Cloudera Runtime 7.3.1

Configuring Streams Messaging Manager

Date published: 2023-09-25
Date modified: 2024-12-10

CLOUDERA

Legal Notice

Contents

Installing Streams Messaging Manager in Cloudera on cloud

Learn how Streams Messaging Manager is installed and set up in Cloudera on cloud with the Streams Messaging cluster templates.

In Cloudera on cloud, Streams Messaging Manager is automatically installed on clusters that you deploy with the Streams Messaging Light Duty, Heavy Duty, and High Availability templates available in the Cloudera Data Hub service. A single instance of the Streams Messaging Manager service (Rest Admin Server and UI Server roles) is installed with each of the templates.

In the Light and High Availability templates, Streams Messaging Manager is installed in the **Master Nodes** host group, while in Heavy Duty, Streams Messaging Manager is installed in the **Streams Messaging Manager Nodes** host group. For more information regarding the Streams Messaging cluster templates and cluster deployment with Cloudera Data Hub, see the *Related Information* section below.

Integration with other services

Streams Messaging Manager is capable of integrating with other services. Enabling integration enables additional features and functions on the Streams Messaging Manager UI.

In Cloudera on cloud, integration with other, eligible Cloudera Streams Messaging services is set up and configured automatically when you deploy the cluster using one of the Streams Messaging templates. For example, if your Streams Messaging cluster includes Kafka Connect, Streams Messaging Manager is automatically configured

to integrate with the Kafka Connect instances running on the cluster. The **Connect** section of the Streams Messaging Manager UI will be available by default, enabling you to manage and monitor Kafka Connect using Streams Messaging Manager.

For Cruise Control, even though the integration with Streams Messaging Manager is automatic, users must have at least Viewer level authorization in Cruise Control. Otherwise, the Cruise Control UI will not be visible in Streams Messaging Manager. For more information, see *Authorizing users to access Cruise Control in Streams Messaging Manager*.

Related Information
Setting up your Streams Messaging cluster
Streams Messaging cluster layout
Monitoring Streams Replication Manager

Setting up Prometheus for Streams Messaging Manager

Get started with setting up and using Prometheus for Streams Messaging Manager. You can use Prometheus as the metrics store for Streams Messaging Manager. Its use is recommended over Cloudera Manager if your deployment includes a significantly large number Kafka entities that you want to monitor.

Cloudera Manager metrics overview

A metric is a property that can be measured to quantify the state of an entity or activity. They include properties such as the number of open file descriptors or CPU utilization percentage across your cluster.

Cloudera Manager monitors a number of performance metrics for services and role instances running on your clusters. These metrics are monitored against configurable thresholds and can be used to indicate whether a host is functioning as expected or not. You can view these metrics in the Cloudera Manager Admin Console.

Cloudera agents collect the metrics from individual brokers and report them to Cloudera Manager once in a minute (through Heartbeat). The collected metrics are stored in the Cloudera Manager database to query or search for historical data.

Cloudera Manager provides a default metric store for Streams Messaging Manager. Streams Messaging Manager fetches the required metrics from Cloudera Manager whenever required and caches the subset of metrics in the Streams Messaging Manager server for which the historical values appear in the Streams Messaging Manager UI. The cache refresh is configurable, the default is 50 seconds.

Prometheus overview

Prometheus is a metrics store that pulls metrics from different endpoints which you configure. Prometheus is not the default metric store for Streams Messaging Manager. If you want to use Prometheus as the metrics store for Streams Messaging Manager, you need to download and configure it. Prometheus supports a larger number of time series entities compared to the Cloudera Manager metrics store.

If you use Prometheus, you can configure the roll up policy, delete specific time series entities, and configure scrape interval and metrics retention period.

For Streams Messaging Manager, you need to configure the following endpoints for Prometheus to pull metrics from:

- Kafka

  Kafka exposes a Prometheus metrics endpoint for Kafka metrics to be pulled.
- Kafka Connect

  Kafka Connect, through configuration, exposes a Prometheus endpoint for Kafka connect metrics to be pulled.
- Prometheus Node Exporter

  You need to configure a separate Node Exporter on each Kafka broker host and enable Prometheus to pull the system metrics.

Streams Messaging Manager queries Prometheus for metrics over time. Prometheus fetches the metrics from the endpoints.

Prometheus relies on external tools for security. For example, you can secure your Prometheus with Nginx in the following scenarios:

- TLS
- TLS with basic authentication
- mTLS

The following image shows the architecture of Prometheus configured for Streams Messaging Manager and secured with Nginx:
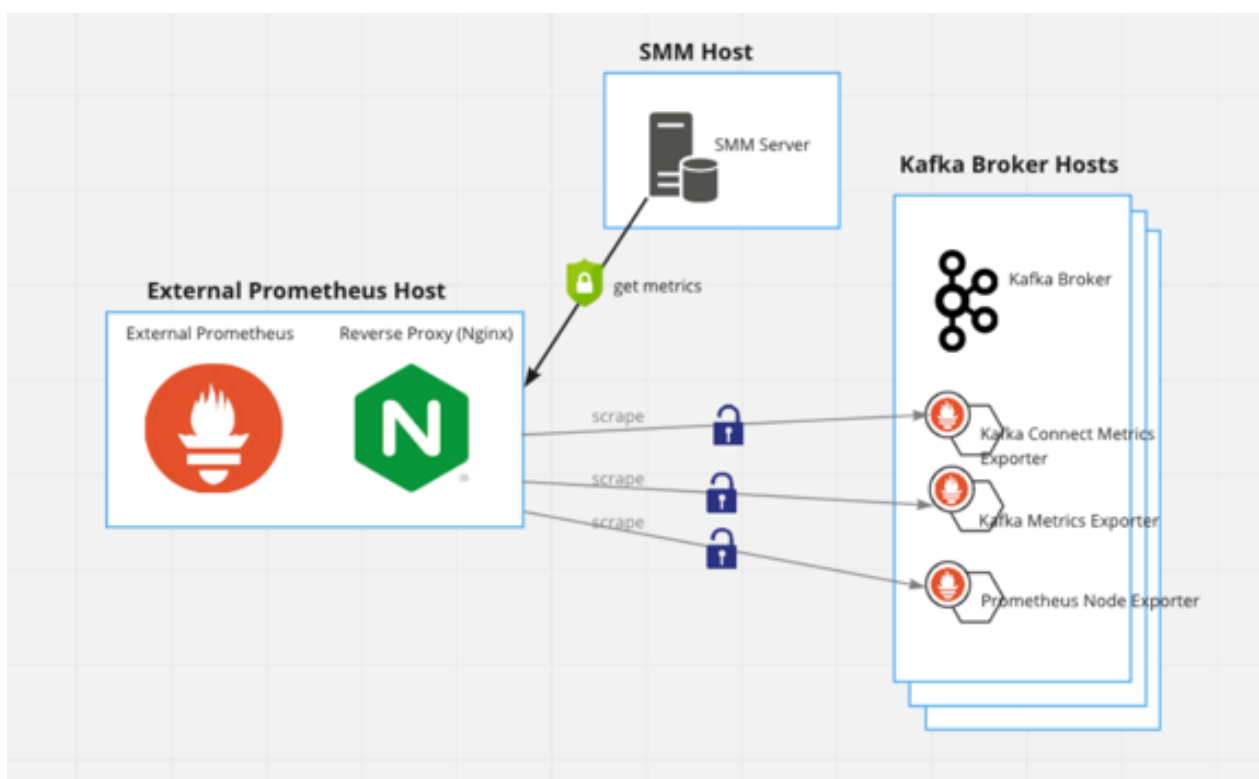
Prometheus configuration for Streams Messaging Manager

Prometheus is not the default metric store for Streams Messaging Manager. If you want to use Prometheus as the metric store for Streams Messaging Manager, you need to download and configure it.

Prerequisites for Prometheus configuration

Learn the prerequisites before you configure Prometheus for Streams Messaging Manager.

- You must download Prometheus and install Prometheus and Prometheus node exporters for each Kafka broker. Streams Messaging Manager requires system-level metrics for each Kafka broker and therefore, Prometheus node exporter must be configured for each Kafka broker. Cloudera only supports Linux node exporters.
- Cloudera recommends using a dedicated Prometheus instance for Streams Messaging Manager because other services querying Prometheus could use the capacity causing Streams Messaging Manager query to timeout.

Prometheus properties configuration

Learn the properties that you need to configure in the prometheus.yml file before you start using the Prometheus metric store for Streams Messaging Manager.

Configure the following properties in the prometheus.yml file:

- Set the scrape_interval property value to 60 seconds in the prometheus.yml file.

```
scrape_interval: 60s
```

- Prometheus automatically assigns the instance label to metrics. However, in case partition leader change reassignment happens, it means an excessive amount of metrics being created. This property is recommended for large clusters.

```
    metric_relabel_configs:
    - source_labels: [__name__]
```

```
        regex: ^(broker_producer_messagesinpersec_total|topic_partition_me
ssagesinpersec_total|topic_partition_bytesinpersec_total|topic_partition
_bytesoutpersec_total)$
        target_label: instance
        replacement: 'no-instance'
```

- Set Kafka host and metrics port values.

```
['luigi-1.luigi.root.hwx.site:24042','luigi-2.luigi.root.hwx.site:24042'
,'luigi-3.luigi.root.hwx.site:24042']
```

    You can find the metric port used by Kafka in Cloudera Manager Kafka service Configuration HTTP Metric
    Report Port .

- Set Kafka Connect host (deployed on same hosts as Kafka) and the metrics port values.

```
['luigi-1.luigi.root.hwx.site:28087','luigi-1.luigi.root.hwx.site:28087'
,'luigi-1.luigi.root.hwx.site:28087']
```

    The metrics port used by Kafka Connect is set in Cloudera Manager Kafka service Configuration Secure Jetty
    Metrics Port and Jetty Metrics Port. Use Secure Jetty Metrics Port if the Kafka Connect metrics reporter uses
    TLS/SSL, otherwise, use Jetty Metrics Port. The value of 28087 in this example is the default secure port.

- Set Prometheus node exporter host (deployed on same hosts as Kafka) and the Prometheus node exporter metrics
    port values.

```
['luigi-1.luigi.root.hwx.site:9100','luigi-2.luigi.root.hwx.site:9100','
luigi-3.luigi.root.hwx.site:9100']
```

- update=true parameter should be only used by Prometheus. Querying Kafka Prometheus endpoint with this flag
    set to true updates the internal metrics cache for stateful metrics.

```
update: ['true']
```

- If Basic Authentication is enabled for the Kafka Connect metrics reporter, add the following parameters:

```
basic_auth:
  username: 'email@username.me'
  password: 'password'
```

    Ensure that you specify the username and password that is configured for the Kafka Connect metrics reporter. You
    can find the username and password in Cloudera Manager Kafka service Configuration Jetty Metrics User Name
    and Jetty Metrics Password.

Configuration example

The following is an example of a Prometheus configuration YAML. You can use this example as a template and
make changes as necessary. Ensure that you replace host and port values as well as the Basic Authentication
credentials with your own values. This example uses the default ports.

```
# my global config
global:
  scrape_interval: 60s
  scrape_timeout: 55s

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093
```

```
# Load rules once and periodically evaluate them according to the global '
evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries s
craped from this config.
  - job_name: 'prometheus'
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
    - targets: ['localhost:9090']

  - job_name: 'kafka'

    metrics_path: '/api/prometheus-metrics'

    params:
      update: ['true']

    static_configs:
    - targets: ['luigi-1.luigi.root.hwx.site:24042','luigi-2.luigi.root.hwx.
site:24042','luigi-3.luigi.root.hwx.site:24042']
    metric_relabel_configs:
    - source_labels: [__name__]
      regex: ^(broker_producer_messagesinpersec_total|topic_partition_messa
gesinpersec_total|topic_partition_bytesinpersec_total|topic_partition_byteso
utpersec_total)$
      target_label: instance
      replacement: 'no-instance'

  - job_name: 'kafka_connect'

    metrics_path: '/prometheus-metrics'

    basic_auth:
      username: 'email@username.me'
      password: 'password'

    static_configs:
    - targets: ['luigi-1.luigi.root.hwx.site:28087','luigi-1.luigi.root.hwx.
site:28087','luigi-1.luigi.root.hwx.site:28087']

  - job_name: 'system_metrics'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['luigi-1.luigi.root.hwx.site:9100','luigi-2.luigi.root.hwx
.site:9100','luigi-3.luigi.root.hwx.site:9100']
```

Where:

- ['luigi-1.luigi.root.hwx.site:9100','luigi-2.luigi.root.hwx.site:9100','luigi-3.luigi.root.hwx.site:9100'] = Kafka host + Node exporter metrics port
- ['luigi-1.luigi.root.hwx.site:24042','luigi-2.luigi.root.hwx.site:24042','luigi-3.luigi.root.hwx.site:24042'] = Kafka host + HTTP Metric Report Port

- ['luigi-1.luigi.root.hwx.site:28087','luigi-1.luigi.root.hwx.site:28087','luigi-1.luigi.root.hwx.site:28087'] = Kafka Connect host + Secure Jetty Metrics Port

Streams Messaging Manager property configuration in Cloudera Manager for Prometheus

Learn about the Streams Messaging Manager properties that you need to configure in Cloudera Manager before you start using the Prometheus metric store.

Configure the following Streams Messaging Manager properties in Cloudera Manager:

- metrics.fetcher.class

    Configures Streams Messaging Manager to fetch metrics from Prometheus. Set it to com.hortonworks.smm.kafka.services.metric.prometheus.PrometheusMetricsFetcher.

- prometheus.metrics.url

    Prometheus metrics URL should be configured here in the format of scheme://prometheus_host:prometheus_port. If HTTPS is configured for Prometheus, HTTPS should be specified as the scheme.

- prometheus.metrics.user

    Should be set if Prometheus is configured for TLS with basic authentication.

- prometheus.metrics.password

    Should be set if Prometheus is configured for TLS with basic authentication.

Kafka property configuration in Cloudera Manager for Prometheus

Learn about the Kafka properties that you need to configure in Cloudera Manager before you start using the Prometheus metric store.

Configure the following Kafka property in Cloudera Manager:

- kafka.http.metrics.port

    Used for exposing Cloudera Manager metrics. This port is now also shared to expose Prometheus metrics for Kafka.

Kafka Connect property configuration in Cloudera Manager for Prometheus

Learn about the Kafka Connect properties that you need to configure in Cloudera Manager before you start using the Prometheus metric store.

In order to set up Prometheus as the Streams Messaging Manager metrics store, you need to configure the metrics reporter of Kafka Connect. Configuration is done in Cloudera Manager by setting Kafka Connect properties. Configuration differs depending on whether you want to enable security for the Kafka Connect metrics reporter. All of the following properties are found in Cloudera Manager Kafka service Configuration . Completing the following makes Prometheus-compatible metrics available on the /prometheus-metrics API path on each Kafka Connect worker host.

Do the following to configure an unsecured metrics-scraping endpoint:

1. Add metrics.jetty.server.prometheus.metrics.enable=true to the Kafka Connect Advanced Configuration Snippet (Safety Valve) for connect-distributed.properties advanced configuration snippet.
2. Look up or configure the value of Jetty Metrics Port.

    This is the port where Kafka Connect exposes its metrics when TLS/SSL is not enabled. This is the port that you need to add to prometheus.yml. Only configure this property if you want to change the port.

Do the following to configure a secure metrics scraping endpoint:

1. Add metrics.jetty.server.prometheus.metrics.enable=true to the Kafka Connect Advanced Configuration Snippet (Safety Valve) for connect-distributed.properties advanced configuration snippet.

2. Enable TLS/SSL, Basic Authentication (BA), or both for the metrics reporter by configuring the following properties:

- Enable TLS/SSL for Kafka Connect
- Enable Basic Authentication for Metrics Reporter
- Jetty Metrics User Name
- Jetty Metrics User Password

Note the following about these properties:

- The Enable TLS/SSL for Kafka Connect property is not specific to the metrics reporter. It enables TLS/SSL for Kafka Connect roles including their metrics reporter.
- You can enable both TLS/SSL and BA on their own, however, Cloudera recommends that you enable both.
- Jetty Metrics User Name and Jetty Metrics User Password set the username and password that you need to add to prometheus.yml if you enable BA.

3. Look up or configure the value of Secure Jetty Metrics Port or Jetty Metrics Port

These are the ports where Kafka Connect exposes its metrics. Secure Jetty Metrics Port is only used if TLS/SSL is enabled, otherwise Kafka Connect uses Jetty Metrics Port. You need to add the port being used to prometheus.yml. Only configure this property if you want to change the port.

Start Prometheus

You should start Prometheus with the startup flag options that you need.

For example,

```
./prometheus --storage.tsdb.retention.time 30d --web.enable-admin-api --query.max-samples 5000000000
```

- --storage.tsdb.retention.time 30d

  Prometheus defaults to 15 days. Streams Messaging Manager monitors metrics up to 30 days. Therefore, this flag must be configured.
- --web.enable-admin-api

  Optional. If Admin APIs are needed, such as deleting time series entities, this flag should be set.
- --query.max-samples

  Optional. When many entities or samples exist, bulk query might reach the maximum sample limit.

Secure Prometheus for Streams Messaging Manager

Streams Messaging Manager supports connecting to Prometheus servers behind a TLS proxy (for example, Nginx).

You can connect to Prometheus servers behind a TLS proxy in the following scenarios:

- TLS

  Streams Messaging Manager verifies the proxy's certificate.
- TLS with basic authentication

  Streams Messaging Manager verifies the proxy's certificate and authenticates itself with username and password.
- mTLS

  Streams Messaging Manager verifies the proxy's certificate and authenticates itself with its TLS Certificate (TLS proxy should recognize it).

Nginx proxy configuration over Prometheus

You need to install and configure Nginx before using it to configure proxy over Prometheus.

Nginx installtion

To use Nginx, you need to install it.

For information about how to install Nginx, see Nginx documentation.

Nginx configuration for Prometheus

Prometheus does not, by default, support TLS encryption for connections to Prometheus instances. If you want to enforce TLS encryption for the connections, you can use Prometheus in conjunction with a reverse proxy and apply TLS at the proxy layer. You can use any reverse proxy, but in this guide you see an Nginx example.

For details about securing Prometheus using TLS encryption, see Prometheus documentation.

> **Note:** In the above linked documentation it is presumed that you want to run Ningx in the 443 port. If it is not the case (for example, it runs on 9443 port), it is not necessary to run Nginx as root. However, you must pay attention to the following things:
>
> • Ensure that the nginx user has access to the TLS certificates.
> • The web.external-url parameter of the Prometheus start command must contain the port number. For example,
>
> ```
> --web.external-url=https://myprometheus.com:9443/prometheus
> ```

After you configure a server for Prometheus, you may want to disable the default server in Nginx configuration. The following example shows the default server commented out:

```
#     server {
#         listen        80 default_server;
#         listen        [::]:80 default_server;
#         server_name  _;
#         root          /usr/share/nginx/html;
#
#         # Load configuration files for the default server block.
#         include /etc/nginx/default.d/*.conf;
#
#         location / {
#         }
#
#         error_page 404 /404.html;
#         location = /404.html {
#         }
#
#         error_page 500 502 503 504 /50x.html;
#         location = /50x.html {
#         }
#     }
```

Setting up TLS for Prometheus

You need to configure Streams Messaging Manager when a TLS proxy is configured over Prometheus.

You need to configure the following:

• Ensure that Cloudera Manager or Streams Messaging Manager recognizes Nginx's TLS certificate. For details, see *Configure Streams Messaging Manager to recognize Prometheus's TLS certificate*.
• Update the Prometheus URL in Streams Messaging Manager settings. You must update the prometheus.metrics.url property to point to the TLS proxy's endpoint (for example, https://myprometheus.com:9443/prometheus) and restart Streams Messaging Manager.

Related Information
Configuring Streams Messaging Manager to recognize Prometheus's TLS certificate

Configuring Streams Messaging Manager to recognize Prometheus's TLS certificate
You can configure Streams Messaging Manager either to use its own keystore or truststore, or to use the auto-TLS feature in Cloudera Manager. Cloudera recommends using the auto-TLS feature for Cloudera clusters.

About this task

If the TLS proxy certificate is not recognized by Streams Messaging Manager, it must be added to the Streams Messaging Manager truststore. The process is different for auto-TLS and the manual TLS setups.

Auto-TLS

If the TLS proxy certificate is not recognized by the cluster, you can add the TLS proxy certificate to the CA truststore of the cluster by triggering a certificate regeneration. This involves restarting the services in the cluster.

1. Go to Administration Security from the left menu bar.
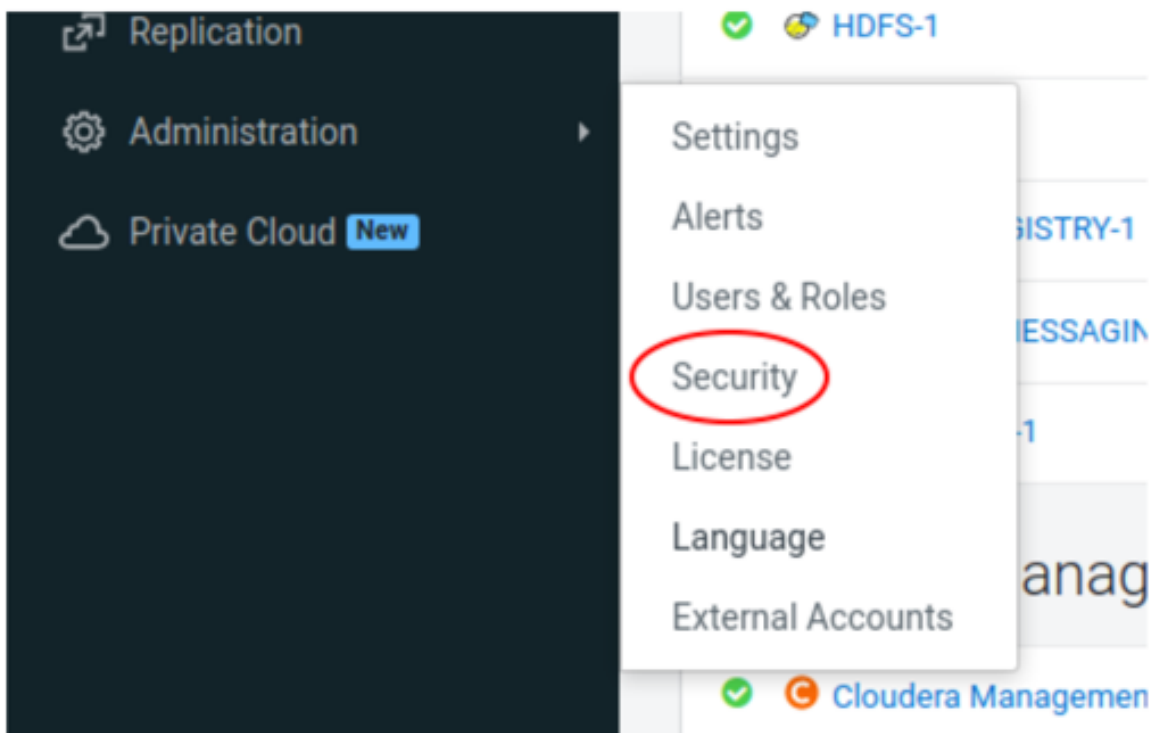
2. Click Rotate Auto-TLS Certificates.
3. In the Trusted CA Certificates Location field, enter the path to the Nginx server's certificate. For example, /etc/ngi
nx/certs/ca-certificate.pem. Ensure that the file is accessible by the cloudera-scm user.
4. Specify the authentication method with other nodes of the cluster (password or certificate).
5. Click Next and follow the instructions in the wizard.

Manual TLS

You can use the keytool command to configure the manual TLS settings.

Keytool is a tool provided by the Java Runtime Environment to manipulate JKS type keystores. You can find it in the bin folder of your JRE installation. For example, /usr/java/default/jre/bin/keytool.

1. Use the following command to add the TLS proxy certificate to the Streams Messaging Manager truststore:

```
keytool -import -file <TLS PROXY OR CA CERTIFICATE> -alias Nginx_for_Pro
metheus -keystore <STREAMS MESSAGING MANAGER TRUSTSTORE> -storepass <TRU
STSTORE PASSWORD>
```

For example,

```
keytool -import -file /etc/nginx/certs/ca-certificate.pem -alias Nginx_f
or_Prometheus -keystore smm_trusstore.jks
```

This command creates the truststore if it does not already exist.

2. Create a keystore for Streams Messaging Manager if it does not already exist:

```
keytool -genkey -keystore smm_keystore.jks -alias smm -keyalg RSA -sigalg
 SHA256withRSA -validity 365 -keysize 3072
```

It creates a keystore with a self-signed key.

3. Set the following Streams Messaging Manager properties in Cloudera Manager:

   • streams.messaging.manager.ssl.keyStorePath/ssl_server_keystore_location
   • ssl_server_keystore_password
   • ssl_server_keystore_keypassword (by default it is the same as the keystore file password)
   • streams.messaging.manager.ssl.trustStorePath/ssl_client_truststore_location
   • ssl_client_truststore_password

Related Information

Setting up TLS for Prometheus

Setting up basic authentication with TLS for Prometheus

To set up TLS with basic authentication, you need to configure Nginx and Streams Messaging Manager.

Configuring Nginx for basic authentication

To configure Nginx for basic authentication, you need to create a file for user and password, update Nginx configurations, and restart Nginx.

Procedure

1. Create an user-password file for Nginx.

```
htpasswd -c /etc/nginx/.htpasswd admin
```

This requires the Apache HTTP package to be installed on the system.

2. Update your Nginx configuration (/etc/nginx/nginx.conf or a custom configuration file in the /etc/nginx/conf.d directory) with the highlighted portion in the code below:

```
    server {
        listen              9443 ssl;
        server_name         _;
        ssl_certificate     /<PATH TO CERTIFICATE>/nginx-server-crt.pem;
        ssl_certificate_key /<PATH TO CERTIFICATE>/nginx-server-key.pem;
        location /prometheus/ {
            auth_basic          "Prometheus";
            auth_basic_user_file /etc/nginx/.htpasswd;
            proxy_pass http://localhost:9090/;
        }
    }
```

3. Restart Nginx.

Configuring Streams Messaging Manager for basic authentication

To configure Streams Messaging Manager, you need to configure Prometheus username and password, and restart Streams Messaging Manager.

Procedure

1. Set the following configurations:

- Prometheus User (admin)
- Prometheus Password (the password you gave in the htpassword tool)

2. Restart Streams Messaging Manager.

Setting up mTLS for Prometheus

Along with or instead of basic authentication, mTLS can also be used for client authentication.

About this task

When using mTLS, both the server and the client authenticate themselves with a TLS certificate. As Streams Messaging Manager is configured to recognize Nginx's certificate, it needs to be configured the other way around.

Procedure

1. Export the certificate or CA certificate of Streams Messaging Manager.

- In case of Auto-TLS, it is

```
/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
```

or

```
/var/lib/cloudera-scm-agent/agent-cert/cm-auto-in_cluster_ca_cert.pem
```

- In case of manual TLS, you can use keytool to export the certificate. For example,

```
keytool -exportcert -rfc -keystore /opt/cloudera/smm_keystore.jks -alias
 smm -file smm.cer
```

2. Add the highlighted lines to Nginx server configuration (/etc/nginx/nginx.conf or a custom configuration file in the /etc/nginx/conf.d directory).

```
    server {
        listen              9443 ssl;
        server_name         _;
        ssl_certificate     /<PATH TO CERTIFICATE>/nginx-server-crt.pem;
        ssl_certificate_key /<PATH TO CERTIFICATE>/nginx-server-key.pem;
        ssl_client_certificate /<PATH TO STREAMS MESSAGING MANAGER
  CERTIFICATE>;
        ssl_verify_client       on;

        location /prometheus/ {
            proxy_pass http://localhost:9090/;
        }
    }
```

3. Restart Nginx.

Prometheus for Streams Messaging Manager limitations

Learn about the known issues and limitations, the areas of impact, and workaround while using Prometheus for Streams Messaging Manager.

- Prometheus is not managed by Cloudera Manager. You should start Prometheus right after Kafka startup because certain metrics are stateful within Kafka and might cause the initial scrape metric values for Prometheus to be high. The stateful metrics accumulate the values from several Kafka restarts until the initial scrape, and accumulate the values between consecutive scrapes.
- You need to configure the scrape interval to 60 seconds.
- Streams Messaging Manager supports Prometheus 2.25 and above versions.
- Streams Messaging Manager supports Linux 1.1.2 and above versions of node exporters.
- Depending on the number of entities (topics, partitions, consumers, and producers), memory requirements for Prometheus might vary.

Troubleshooting Prometheus for Streams Messaging Manager

Troubleshooting Prometheus for Streams Messaging Manager requires being able to diagnose and debug problems related to performance, network connectivity, out-of-memory conditions, disk space usage, and crash or non-responsive conditions.

This section provides some troubleshooting solutions for Prometheus issues for Streams Messaging Manager.
Issue: If no metrics are available.

Solution: Examine if all configured Prometheus targets are running.

Issue: You observe incorrect metrics in Streams Messaging Manager.

Solution: Examine up metrics to determine if scraping failed at certain points in time.

Issue: If Prometheus does not start up or scrape right after Kafka startup, or Prometheus is down for some time, scrapped metric values might be unusually high.

Solution: This situation can be avoided by querying the endpoints manually with the update=true parameter and then starting Prometheus. If you already encountered this situation, you can delete that particular time series with the delete API of Prometheus.

Performance comparison between Cloudera Manager and Prometheus

Learn the performance comparison between Cloudera Manager and Prometheus metric stores.

- Prometheus can handle more than twice the amount of TimeSeries entities than Cloudera Manager while maintaining the consistency of the metrics.
- Metrics queries for shorter time periods (30 minutes, 1 hour) take about half the time in case of Prometheus compared to Cloudera Manager.
- Metrics queries for larger time periods (2 weeks, 1 month), Prometheus seems to be about 30% slower than Cloudera Manager.
- Metric inconsistencies are not observed while testing Prometheus.