

# Getting Metrics for Streams Messaging Manager

Date published: 2021-07-12

Date modified: 2021-08-05



# Legal Notice

© Cloudera Inc. 2024. 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

<b>Cloudera Manager metrics overview.....</b>	<b>4</b>
<b>Prometheus metrics overview.....</b>	<b>4</b>
<b>Prometheus configuration for SMM.....</b>	<b>5</b>
Prerequisites for Prometheus configuration.....	5
Prometheus properties configuration.....	5
SMM property configuration in Cloudera Manager for Prometheus.....	7
Kafka property configuration in Cloudera Manager for Prometheus.....	8
Kafka Connect property configuration in Cloudera Manager for Prometheus.....	8
<b>Start Prometheus.....</b>	<b>8</b>
<b>Secure Prometheus for SMM.....</b>	<b>8</b>
Nginx proxy configuration over Prometheus.....	9
Nginx installtion.....	9
Nginx configuration for Prometheus.....	9
Setting up TLS for Prometheus.....	10
Configuring SMM to recognize Prometheus's TLS certificate.....	10
Setting up basic authentication with TLS for Prometheus.....	11
Configuring Nginx for basic authentication.....	11
Configuring SMM for basic authentication.....	12
Setting up mTLS for Prometheus.....	12
<b>Prometheus for SMM limitations.....</b>	<b>13</b>
<b>Troubleshooting Prometheus for SMM.....</b>	<b>13</b>
<b>Performance comparison between Cloudera Manager and Prometheus.....</b>	<b>13</b>

## Cloudera Manager metrics overview

Cloudera Manager provides a default metric store for Streams Messaging Manager (SMM). SMM fetches the required metrics from Cloudera Manager whenever required and caches the subset of metrics in the SMM server for which the historical values appear in the SMM UI. The cache refreshes every minute.

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 level database to query or search for historical data.

## Prometheus metrics 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 (SMM). If you want to use Prometheus as the metric store for SMM, you need to download and configure it.

Prometheus supports a larger number of time series entities compared to the Cloudera Manager metric 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 SMM, you need to configure the following endpoints for Prometheus to pull metrics from:

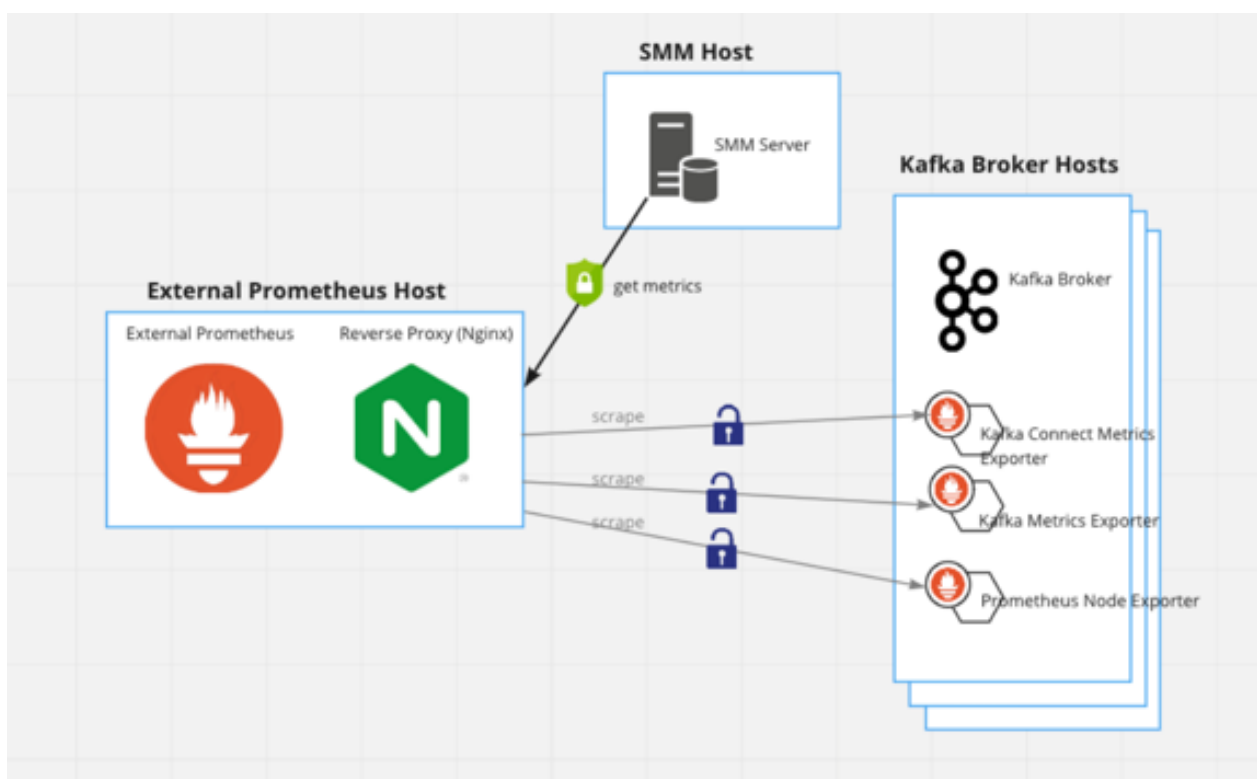
- Kafka  
Kafka exposes a Prometheus metrics endpoint for Kafka metrics to be pulled.
- Kafka Connect  
Kafka Connect, through configuration, exposes a Prometheus endpoint for Kafka connect metrics to be pulled.
- Prometheus Node Exporter  
You need to configure a separate Node Exporter on each Kafka broker host and enable Prometheus to pull the system metrics.

SMM 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 SMM and secured with Nginx:



## Prometheus configuration for SMM

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

### Prerequisites for Prometheus configuration

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

- You must download Prometheus and install Prometheus and Prometheus node exporters for each Kafka broker. SMM 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 SMM because other services querying Prometheus could use the capacity causing SMM 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 (SMM).

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_messagesinpersec_total|topic_partition_bytesinpersec_total|topic_partition_bytesoutpersec_total)$
  target_label: instance
  replacement: 'no-instance'
```

- Set Kafka hosts + metrics port (Kafka's Cloudera Manager configuration kafka.http.metrics.port) values.

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

- Set Kafka Connect hosts (deployed on same hosts as Kafka) + metrics port (Kafka Connect's Cloudera Manager configuration connect.prometheus.metrics.port) values.

```
['luigi-1.luigi.root.hwx.site:28086', 'luigi-1.luigi.root.hwx.site:28086',
 'luigi-1.luigi.root.hwx.site:28086']
```

- Set Prometheus node exporter hosts (deployed on same hosts as Kafka) + 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']
```

For example, configure Prometheus with the following YAML file:

```
# 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
  scraped 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']
- job_name: 'kafka_connect'

metrics_path: '/api/prometheus-metrics'
static_configs:
- targets: ['luigi-1.luigi.root.hwx.site:28086','luigi-1.luigi.root.hwx.site:28086','luigi-1.luigi.root.hwx.site:28086']
- 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']
- job_name: 'kafka'
  metric_relabel_configs:
  - source_labels: [__name__]
    regex: ^(broker_producer_messages_in_persec_total|topic_partition_messages_in_persec_total|topic_partition_bytes_in_persec_total|topic_partition_bytes_out_persec_total)$
    target_label: instance
    replacement: 'no-instance'

```

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 + Kafka metrics port
- ['luigi-1.luigi.root.hwx.site:28086','luigi-1.luigi.root.hwx.site:28086','luigi-1.luigi.root.hwx.site:28086'] = Kafka Connect host name
- 28086 = connect.prometheus.metrics.port

You need to replace Kafka and Kafka Connect host names and set Kafka Connect's Cloudera Manager configuration property `connect.prometheus.metrics.port` in the `connect-distributed.properties` file with values from your cluster.

## SMM property configuration in Cloudera Manager for Prometheus

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

Configure the following SMM properties in Cloudera Manager:

- `metrics.fetcher.class`

Configures SMM 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.

Configure the following Kafka Connect property in Cloudera Manager:

- `connect.prometheus.metrics.port`  
Exposes Kafka Connect Prometheus metrics.

## 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. SMM 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 SMM

Streams Messaging Manager (SMM) 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  
SMM verifies the proxy's certificate.
- TLS with basic authentication  
SMM verifies the proxy's certificate and authenticates itself with username and password.
- mTLS  
SMM 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 installation

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 Nginx 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 (SMM) when a TLS proxy is configured over Prometheus.

You need to configure the following:

- Ensure that Cloudera Manager or SMM recognizes Nginx's TLS certificate. For details, see *Configure SMM to recognize Prometheus's TLS certificate*.
- Update the Prometheus URL in SMM 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 SMM.

### Related Information

[Configuring SMM to recognize Prometheus's TLS certificate](#)

## Configuring SMM to recognize Prometheus's TLS certificate

You can configure Streams Messaging Manager (SMM) 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 CDP clusters.

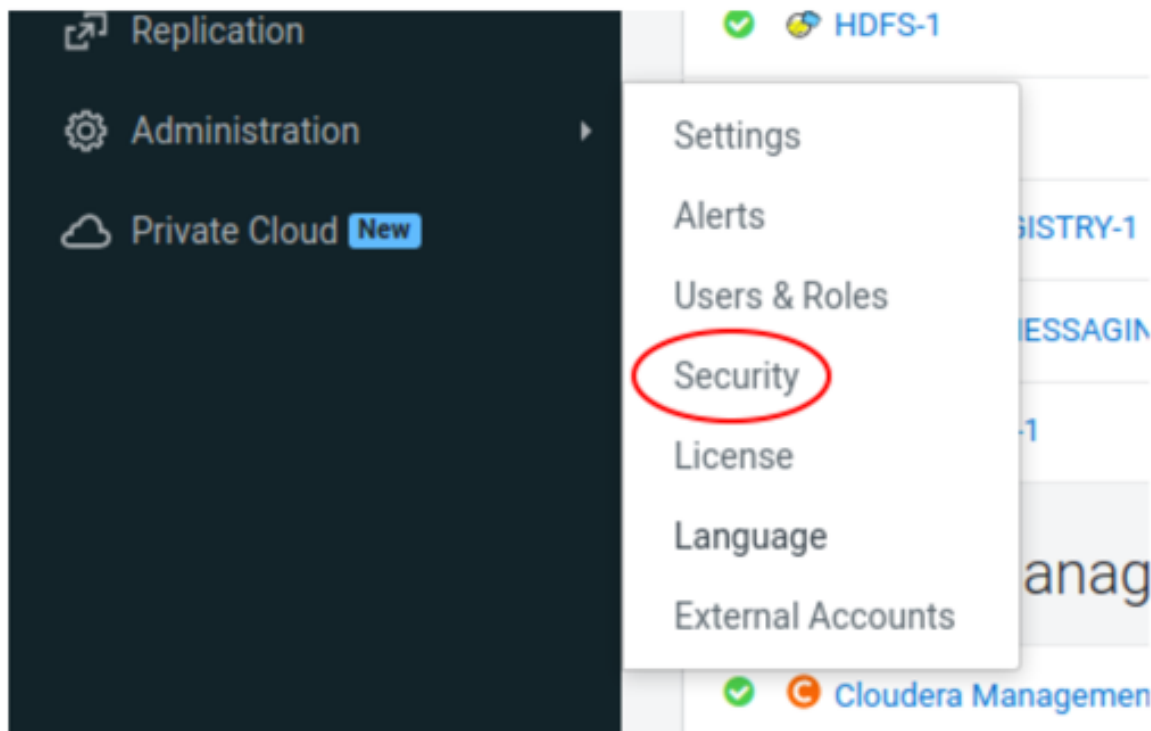
### About this task

If the TLS proxy's certificate is not recognized by SMM, it must be added to SMM 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/nginx/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's certificate to SMM's truststore:

```
keytool -import -file <TLS PROXY OR CA CERTIFICATE> -alias Nginx_for_Prometheus -keystore <SMM_TRUSTSTORE> -storepass <TRUSTSTORE PASSWORD>
```

For example,

```
keytool -import -file /etc/nginx/certs/ca-certificate.pem -alias Nginx_for_Prometheus -keystore smm_truststore.jks
```

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

2. Create a keystore for SMM 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 SMM 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 (SMM).

### 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 part:

```
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 SMM for basic authentication

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

### Procedure

1. Set the following configurations:
  - Prometheus User (admin)
  - Prometheus Password (the password you gave in the htpasswd tool)
2. Restart SMM.

## 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 (SMM) is configured to recognize Nginx's certificate, it needs to be configured the other way around.

### Procedure

1. Export SMM's certificate or its CA certificate.

- 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 SMM'S CERTIFICATE>;  
ssl_verify_client      on;  
  
location /prometheus/ {  
    proxy_pass http://localhost:9090/;  
}
```

3. Restart Nginx.

## Prometheus for SMM limitations

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

- 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.
- SMM supports Prometheus 2.25 and above versions.
- SMM 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 SMM

Troubleshooting Prometheus for SMM 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 SMM.

**Issue: If no metrics are available.**

Solution: Examine if all configured Prometheus targets are running.

**Issue: You observe incorrect metrics in SMM.**

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.