Cloudera DataFlow for Data Hub 7.2.18

Getting Started with Streams Messaging Cluster on CDP Public Cloud

Date published: 2019-12-16
Date modified: 2024-04-03

CLOUDERA

https://docs.cloudera.com/

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

Introducing streams messaging cluster on CDP Public Cloud..... 5

Meet the prerequisites to create streams messaging cluster.....5

Creating Machine User..... 6

Granting Machine User access to environment.....7

Creating Kafka topic.....8

Create Ranger policies for Machine User account.....9
Create topic policy.....10
Create consumer group policy..... 11

Produce data to Kafka topic..... 12
Setting workload password.....12
Connecting to Kafka host.....13
Configuring LDAP authentication.....14
Producing data to Kafka topic..... 14

Consuming data from Kafka topic..... 15

Use Kerberos authentication..... 16
Kerberos authentication using the ticket cache.....17
Kerberos authentication using a keytab..... 18

Monitoring Kafka activity in Streams Messaging Manager..... 19

Use Schema Registry..... 22
Gather configuration information.....23
Finding list of brokers..... 23
Finding Schema Registry endpoint..... 24
Creating TLS truststore..... 24
Defining Schema Registry access policies.....25
Producing data in Avro format..... 27
Checking schema registration.....29
Checking producer activity.....30
Consuming data from Kafka topics using stored schemas..... 32

Monitor end-to-end latency..... 33
Setting up authorization policies..... 33
Enabling end-to-end latency monitoring..... 34

Evolve your schema..... 36
Reconfiguring the Kafka consumer..... 36
Reconfiguring the Kafka producer..... 37

What to do next..... 38

Introducing streams messaging cluster on CDP Public Cloud

You can create a streams messaging cluster in CDP Public Cloud and produce data to and consume data from an Apache Kafka topic by using that cluster. The concepts and usage of streams messaging cluster on CDP Public Cloud are explained using a simple workflow.

In this example you use a simple application (vmstat) to generate some raw text data and a kafka-console-producer command to send the data to a Kafka topic. You then use a kafka-console-consumer command to read data from that Kafka topic. After you have the producer and consumer running, you use Stream Messaging Manager (SMM), which is also available on CDP Public Cloud, to monitor the functionality of the workflow that you create.

Alternately, you can also structure the data in an object format and send it to Kafka in Avro format. To do this, you store a schema in CDP's Schema Registry and understand how to use a simple Java Kafka client to send and read data using that schema.

You can also update your schema to fit your requirements, and reconfigure the consumer and producer to use the latest schema.

The complete workflow consists of the following tasks:

1. Meeting the prerequisites
2. Creating a Machine User
3. Granting Machine User access to environment
4. Creating a Kafka topic
5. Creating Ranger policies
6. Producing data to Kafka topic
7. Consuming data from Kafka topic
8. Using Kerberos authentication
9. Monitoring Kafka activity in SMM
10. Using Schema Registry
11. Monitoring end-to-end latency in SMM
12. Evolving your schema

Meet the prerequisites to create streams messaging cluster

You need to ensure that your user is assigned the required role and privilege to create the workflow with Kafka on CDP Public Cloud.

Ensure that you have met the prerequisites outlined in the *Creating your First Streams Messaging Cluster in CDP Public Cloud*, in the *Meet the prerequisites* section.

Note: As stated in the prerequisites document, your user must have the EnvironmentAdmin role. This is required so that you have admin privileges on Kafka and Ranger to perform the tasks required in this tutorial. If you do not have this role, request your environment administrator for assistance.

After you meet the prerequisites, you need to create your streams messaging cluster using a default cluster definition.

To complete this tutorial you need a running Kafka cluster on CDP Public Cloud. For instructions, see *Create your cluster*.

Related Information

Meet the prerequisites

Create your cluster

Creating Machine User

You must create a dedicated service account called Machine User to use in the production environment. You create a Machine User in the Management Console with a unique name.

Before you begin

You must have created a Kafka cluster on CDP Public Cloud.

About this task

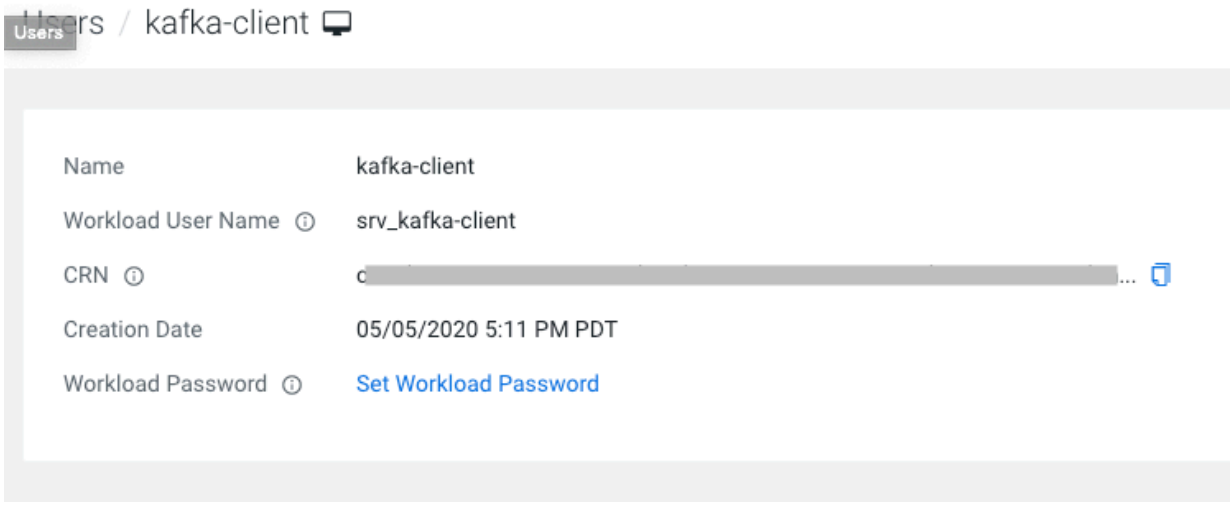
You can use your personal user account to connect to CDP for all the examples in this tutorial. However, in a production environment, you must create a dedicated service account to use ingestion processes or pipelines. So, instead of using your own account, this tutorial takes you through creating a service account.

Machine User is the CDP nomenclature for a service account. For more information on Machine User, see *CDP machine user*.

Procedure

1. Navigate to Management Console User Management .
2. Click Actions Create Machine user .
3. Enter a unique name for the user and click Create.
   After you create a machine user, you see a page like the following one with the user details.

   Note: The Workload User Name (srv_kafka-client, in the example below) is different from the actual user name (kafka-client). The Workload User Name is the identifier you use to configure the access for your workload or application later.



What to do next

Click Set Workload Password and follow the prompts to set a password for the Machine User. You can leave the Environment field empty.

Related Information

CDP machine user

Granting Machine User access to environment

Before you start working with the Machine User account, to access the Kafka cluster on CDP Public cloud, you must grant the user access to the environment that contains the cluster.

Before you begin

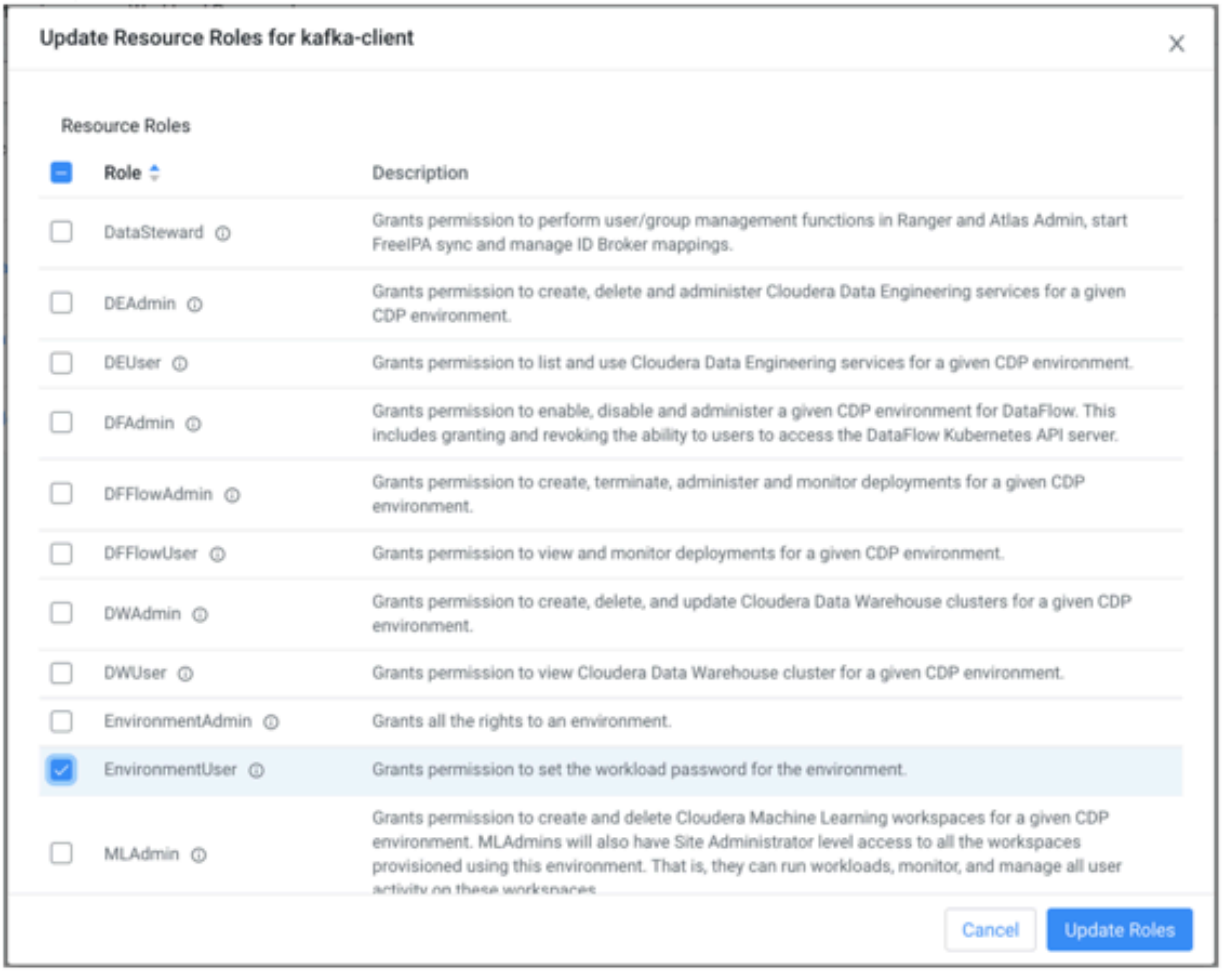
You must have created a Machine User account.

About this task

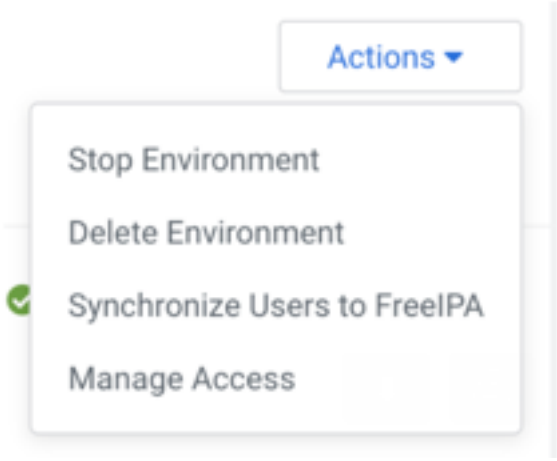
For more information on providing the user access to your cluster, see *Give users access to your cluster*.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click Actions Manage Access .
3. In the search field, type the name of the Machine User and select it from the search results.
4. Check the EnvironmentUser role and click Update Roles.



5. Go to the environment page and click Actions Synchronize Users to FreeIPA , to ensure that the role assignment is in effect for the environment.



6. On the Synchronize Users to FreeIPA page, click Synchronize`Users.

What to do next
You need to create Kafka topics.

Related Information
Give users access to your cluster

Creating Kafka topic

After you create a Kafka cluster, you can create Kafka topics. You create Kafka topics in the environment where your Kafka cluster is running by using the Streams Messaging Manager (SMM) web UI.

Before you begin
You must have created a Kafka cluster and a Machine User account, and granted environment access to the Machine User.

About this task
You create two Kafka topics in the tutorial:

- machine-data

  Topic containing machine usage data in plain text format.
- machine-data-avro

  Topic containing machine usage data in Avro format.

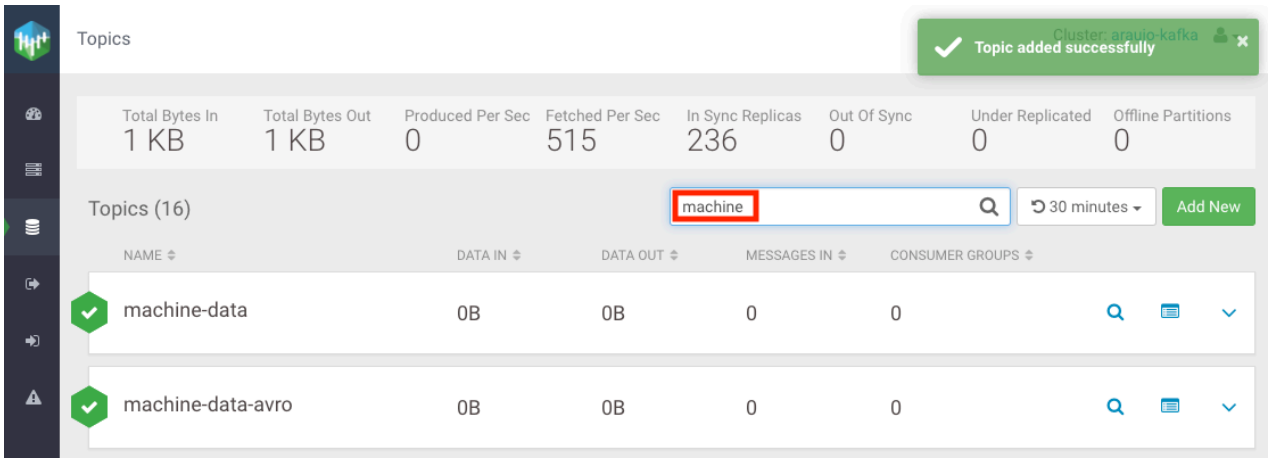
Procedure
1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Streams Messaging Manager (SMM) on the Services pane to open the SMM web UI.

   SMM is CDP's tool to monitor and manage Kafka. You use it to create the topics you need.

4. In the SMM web UI, click the topics icon ( ) on the left-hand bar.
5. Click Add New.
6. Enter the following details and click Save:
   - Topic name: machine-data
   - Partitions: 10
   - Availability: Maximum
   - Cleanup Policy: delete
7. Click Add New to add a second topic.
8. Enter the following details for the new topic and click Save:
   - Topic name: machine-data-avro
   - Partitions: 10
   - Availability: Maximum
   - Cleanup Policy: delete

Results

In the SMM Topics page, type machine in the search field to filter the topics. You must see both topics you created.



Create Ranger policies for Machine User account

To read data from or write data to Kafka topics, you must grant certain privileges to your Machine User. To do this, you create new access policies for the Kafka topics in Ranger. You create one Topic policy and one Consumer Group policy.

Because your personal user is an environment administrator, you already have privileges to access and modify Kafka topics. This is not the case, though, with the Machine User account you created.

Authorization policies in CDP are maintained and enforced by Apache Ranger. To learn more about Apache Ranger authorization policies, see *Using Ranger to Provide Authorization in CDP*.

To allow the Machine User account to read from and write data to the Kafka topics you created, it is necessary that you define new access policies for those topics in Ranger. You need to create the following policies:

- Topic policy to grant access to the two Kafka topics.
- Consumer Group policy to define which consumer groups the Machine User can use to consume data from the Kafka topics.

Related Information
Using Ranger to Provide Authorization in CDP

Create topic policy

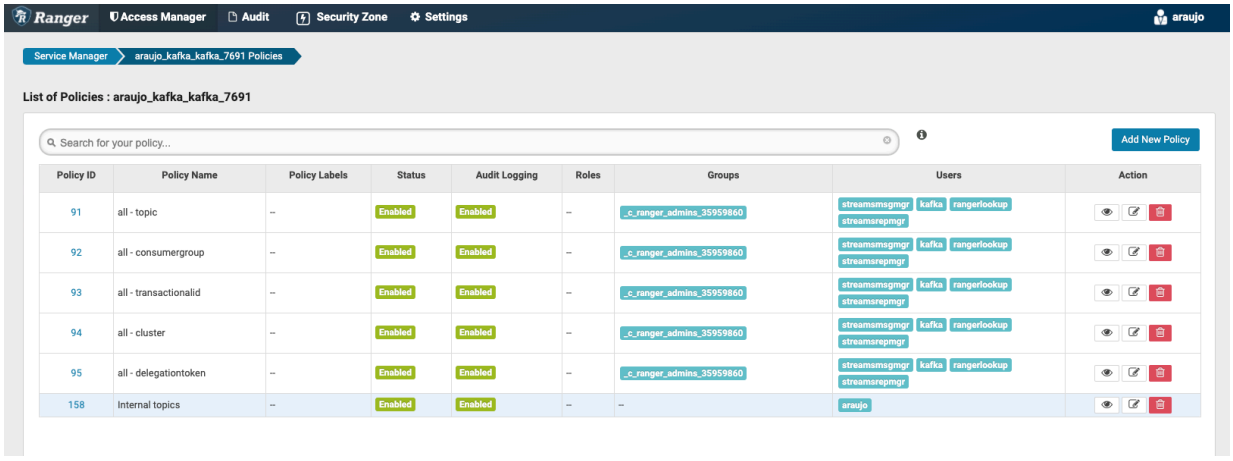
Learn how to grant privileges to a Machine User for accessing and modifying Kafka topics using the Ranger web UI. You must create a new policy, select the Machine User, and add permissions for the Machine User.

Before you begin
You must have created a Machine User, granted environment access to the Machine User, and created two Kafka topics.

Procedure

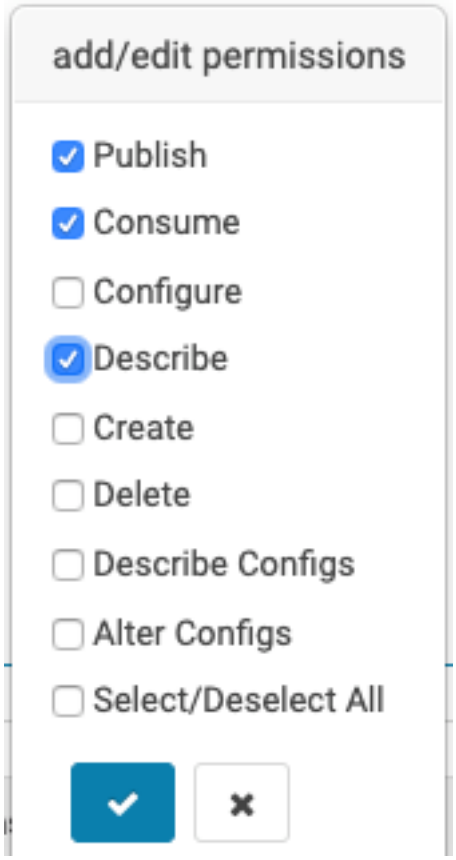
1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click Ranger on the top pane to open the Ranger web UI.
3. In the Ranger web UI, click Access Manager Resource Based Policies .
4. Under the KAFKA group, select the policy of your Kafka cluster.

   To select it, click on the policy name, and not on the icon. You see the list of pre-defined policies for your Kafka cluster.



5. Click Add New Policy to create a new policy and enter the following details:
   - Policy Name: Machine Data topics
   - Topic: machine-data*
   - Description: All topics prefixed with machine-data
6. Below the Allow Conditions section, click the empty Select Users box and select the Machine User you created.

7. Click Add Permissions, select the Publish, Consume, and Describe permissions, and then click the tick mark.



8. Scroll to the bottom of the page and click Add to save the policy.

Create consumer group policy

Learn how to grant privileges to a Machine User for using consumer groups to consume data from the Kafka topics by using the Ranger UI. You create a new policy, select the Machine User, and add permissions for the Machine User.

Before you begin

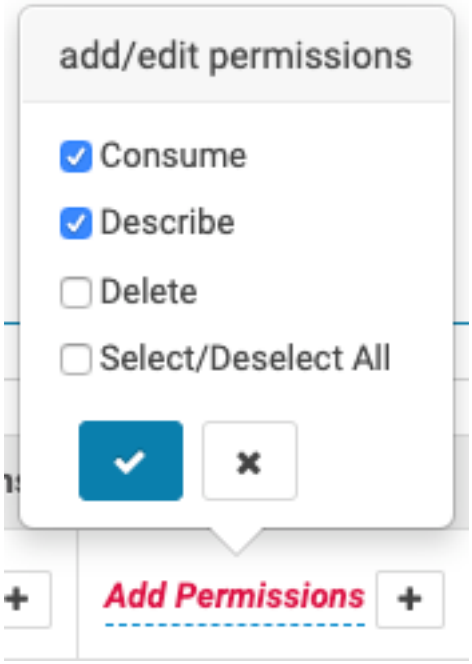
You must have created a Machine User, granted environment access to the Machine User, and created two Kafka topics.

Procedure

1. Click Add New Policy, to create a new one, and enter the following details:
   - Policy Name: Machine Data consumer groups
   - Consumer group: machine-data*
   - Description: All consumer groups prefixed with machine-data
2. Below the Allow Conditions section, click on the empty Select User box and select the Machine User you had created.

   You can type the name of the user to filter the list of users. Also note that the user name appears with the srv_ prefix, indicating that it is a Machine User.

3. Click Add Permissions, select the Consume and Describe permissions and then click the tick mark.



4. Scroll to the bottom of the page and click Add to save the policy.

Results
Now you should see your new policies listed in Ranger. Your Machine User is ready for use.

What to do next
You can start producing data to the Kafka topic.

Produce data to Kafka topic

Learn how to produce data to a Kafka topic. To do this, you need to set the workload password, connect to a Kafka host, provide LDAP authentication, and finally produce data to a Kafka topic.

The applications that produce data to or consume data from Kafka can run on any of the servers in your network, provided they have connectivity to the Kafka cluster in CDP. These applications should not be executed on the hosts of the Kafka cluster, because this could affect the Kafka operation.

To produce data to a Kafka topic, you use the kafka-console-producer command line application because the command line application is already installed on the Kafka hosts and helps you to understand the application configuration without having to build an application. In a later step you configure and run an application on a separate host to connect to the cluster.

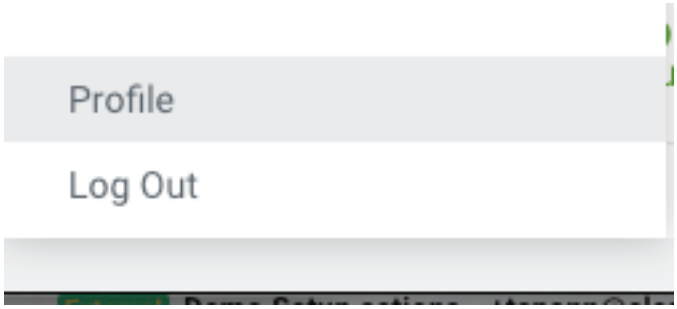
Setting workload password

Learn how to set your personal workload password.

Before you begin
You must have created a Machine User account.

Procedure

1. Click Profile in your user menu on the CDP console.



2. On your profile page, click Set Workload Password and follow the prompts to set a password for your user. You can leave the Environment field empty.

Connecting to Kafka host

Learn how to connect to a Kafka host from the Management Console. You need to select the Kafka cluster you created, select a Kafka broker, and then connect to the broker host.

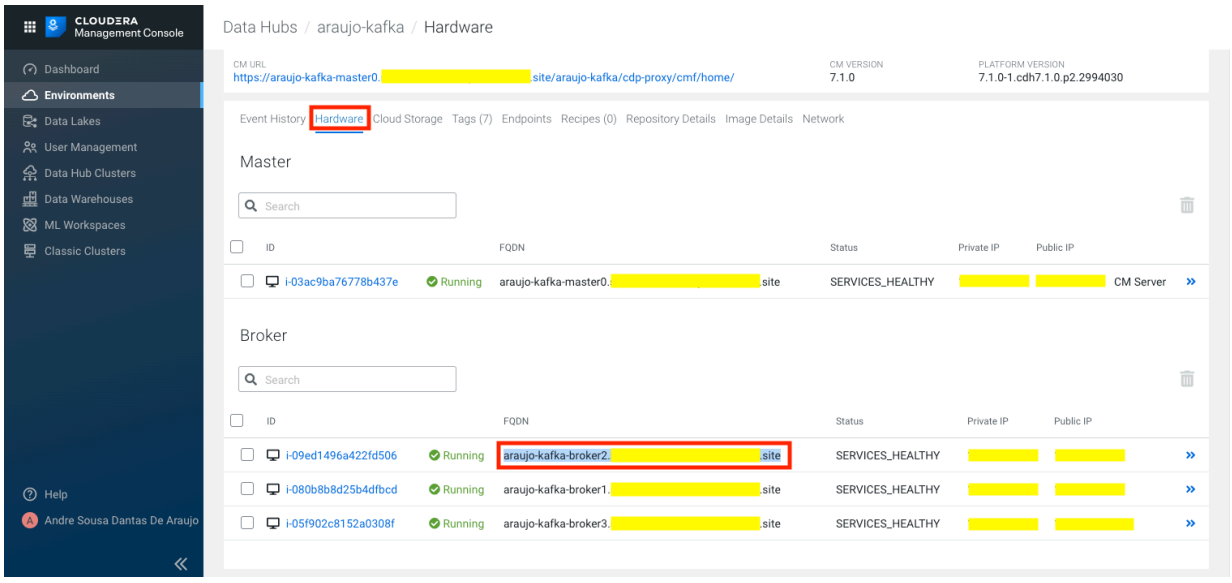
Before you begin

You must have created a Machine User account, granted environment access to the Machine User, created Kafka topics, and created Ranger policies.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. On the Kafka cluster page, click Hardware and select one of the Kafka brokers to run your producer test.

   Select and copy the full hostname of the broker.



4. Using the SSH client of your preference, connect to the broker host using your CDP user and your workload password for authentication.

   For example, if connecting from a terminal, you can use:

   ```
   ssh [***YOUR_USER_NAME***]@[***BROKER_HOST_NAME***]
   ```

Configuring LDAP authentication

All the Kafka clusters created in CDP are secure because TLS is enabled to encrypt communications between clients and the cluster, and strong authentication is enforced, requiring clients to authenticate either through Kerberos or LDAP.

About this task

In this example, you use LDAP authentication. LDAP authentication is not enabled by default.

Before you begin

You must have connected to the Kafka host.

Procedure

1. Provide the client with the security.protocol and sasl.mechanism information to indicate the security protocol and authentication mechanism to use.
2. Provide the client with the ssl.truststore.location information which is required for the client to trust the brokers' certificates.

   All the Kafka hosts deployed by CDP have a valid truststore deployed at /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks. You need to use this truststore. Later you look at how to create a truststore from scratch.
3. Provide the client with the sasl.jaas.config information which is the LDAP credentials to use for authentication.

   For a detailed explanation on how to configure TLS/SSL and LDAP authentication for Kafka clients, see *Configure Kafka clients*.

Related Information

Configure Kafka clients for LDAP authentication

Producing data to Kafka topic

To produce data to Kafka topics, you create a property file, ensure that its permissions are set restrictively, list all the topics that the Machine User has access to, and produce a text message to the Kafka topic.

About this task

Here, you produce data to a Kafka topic on your SSH session.

Before you begin

You must have created a Kafka cluster, created a Machine User, granted environment access to the Machine User, created Kafka topics, and created Ranger policies.

Procedure

1. Create a new client-ldap.properties file with the following content:

   ```
   security.protocol=SASL_SSL
   sasl.mechanism=PLAIN
   ```

```
ssl.truststore.location=/var/lib/cloudera-scm-agent/agent-cert/cm-auto-g
lobal_truststore.jks
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
 required username="srv_kafka-client" password="SuperSecretPassword";
```

Note: Replace the values of the username and password to match with the machine user credentials that you created.

2. The client-ldap.properties file contains sensitive information. Ensure its permissions are set restrictively.

```
chmod 400 client-ldap.properties
```

With this configuration you can start running Kafka client applications.

3. Run the following command to list all the topics that the Machine User has access to.

```
BROKER_HOST=$(hostname -f)

kafka-topics \
  --bootstrap-server $BROKER_HOST:9093 \
  --command-config client-ldap.properties \
  --list
```

The output include a number of INFO lines, with the two topics you created at the end of the output:

```
machine-data
machine-data-avro
```

If you do not see them, go back and review the previous steps.

4. Run the following command to produce a simple text message to the Kafka topic by using the kafka-console-producer command.

```
echo 'Hello, Kafka!' | kafka-console-producer \
  --broker-list $BROKER_HOST:9093 \
  --producer.config client-ldap.properties \
  --topic machine-data
```

Results

If the output of the command above contains only INFO lines and no ERROR lines, it indicates that data was produced correctly to the Kafka topic.

What to do next

In the next section, you use the kafka-console-consumer command to verify the contents of the Kafka topic.

Consuming data from Kafka topic

After you produce data to a Kafka topic, you can consume data from the Kafka topic.

About this task

The configuration that you used previously to produce data to Kafka topics is the same that you use to consume data from Kafka topics. You need to verify that the data was written correctly to the Kafka topic and can also be read by the client.

Before you begin
You must have produced data to your Kafka topic.

Procedure
Run the following command to consume the data produced previously:

```
kafka-console-consumer \
  --bootstrap-server $BROKER_HOST:9093 \
  --consumer.config client-ldap.properties \
  --topic machine-data \
  --group machine-data-1 \
  --from-beginning
```

Results
You see the following output, which tells you that the client successfully read it from the topic:

```
Hello, Kafka!
```

Note:

If you rerun the same command, reusing the consumer group name (machine-data-1 in the previous example), the data is not shown because the client continues from where it left off in the previous execution. To read the data again, change the group name to something else (for example, machine-data-2).

Use Kerberos authentication

Learn how and when to configure Kerberos authentication.

LDAP authentication is generally easier to configure for remote clients, because it does not require Kerberos libraries or clients to be installed and the remote clients do not need direct connectivity to your Kerberos or LDAP server.

In some situations, though, Kerberos authentication may be required and/or preferred. In this section, you run the same kafka-console-consumer command you used for LDAP authentication, but this time using Kerberos authentication to demonstrate the required configuration.

When using Kerberos, you can use two different sources for the authentication credentials: a ticket cache or a keytab. The ticket cache is the place where the Kerberos ticket for a user is stored after the user authenticates successfully. You can use the ticket cache when the user has been already authenticated using a username and password. The keytab is a special and sensitive file that contains the user credentials. Keytabs do not require that the user authenticates previously.

In this example, you create a new configuration file called client-kerberos.properties, which is similar to the client-ldap.properties file that you used in the previous example but with Kerberos-specific parameters. You also need an additional configuration file, which you call either jaas-cache.conf or jaas-keytab.conf, depending on the credentials source you use. This JAAS configuration file is used to communicate the authentication credentials source (ticket cache or keytab) to the Kafka client.

For more details on how to enable Kerberos authentication for Kafka, see *Enable Kerberos authentication* for Kafka clients.

Related Information
Enable Kerberos authentication

Kerberos authentication using the ticket cache

You can configure Kerberos authentication using a ticket cache. To do so, you authenticate with Kerberos to acquire a valid Kerberos ticket. Then you connect to Kafka authenticating with the Kerberos ticket.

Before you begin

You must have produced data to a Kafka topic and consumed data from the Kafka topic.

Procedure

1. Create a file called client-kerberos.properties with the following content:

```
security.protocol=SASL_SSL
sasl.mechanism=GSSAPI
sasl.kerberos.service.name=kafka
ssl.truststore.location=/var/lib/cloudera-scm-agent/agent-cert/cm-auto-glo
bal_truststore.jks
sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required us
eTicketCache=true;
```

2. Before connecting to Kafka, you need to authenticate successfully with Kerberos to acquire a valid Kerberos ticket. You can use kinit for that. Run the following commands, replacing srv_kafka-client with your Machine User name. When prompted for the password, provide the Workload Password that you had set.

```
kdestroy

kinit srv_kafka-client
```

3. If the command is successful, you should have a valid ticket in your ticket cache, as shown by the following klist command:

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_1501600556
Default principal: srv_kafka-client@XYZ.SITE

Valid starting       Expires              Service principal
05/07/2020 03:10:58  05/08/2020 03:10:52  krbtgt/XYZ.SITE@XYZ.SITE
 renew until 05/14/2020 03:10:52
```

   If the command is not successful, your ticket cache should be empty, as shown in the following example:

```
$ klist
klist: No credentials cache found (filename: /tmp/krb5cc_1501600556)
```

4. After you have a valid ticket in the ticket cache you can run the following commands to connect to Kafka, authenticating with the Kerberos ticket.

```
BROKER_HOST=$(hostname -f)

kafka-console-consumer \
  --bootstrap-server $BROKER_HOST:9093 \
  --consumer.config client-kerberos.properties \
  --topic machine-data \
  --group machine-data-$RANDOM \
  --from-beginning
```

   You should see the contents of the machine-data topic as follows:

```
Hello, Kafka!
```

5. After testing, destroy the ticket in your ticket cache:

```
kdestroy
```

If you try to run the kafka-console-consumer command again, without the ticket, you see authentication errors like the following example:

```
Caused by: javax.security.auth.login.LoginException:
Could not login: the client is being asked for a password, but the Kafka c
lient code does not
currently support obtaining a password from the user. not available to
garner  authentication
information from the user
```

Kerberos authentication using a keytab

You can configure Kerberos authentication using a keytab. To do so, you download the keytab from the environment where your Kafka cluster is running, copy the keytab to your broker host, create a configuration file with required properties, and connect to Kafka.

About this task

As mentioned previously, instead of using a ticket from the ticket cache you can also authenticate using a keytab.

Before you begin

You must have produced data to a Kafka topic and consumed data from the Kafka topic.

Procedure

1. Navigate to Management Console User Management .
2. Select the Machine User.
3. Click Actions Get Keytab .
4. On the Get Keytab dialog box, select the environment where the Kafka cluster is running and click Download.
5. Copy the keytab that was downloaded on your computer to the broker host you were connected to.
6. The keytab file contains sensitive information. Ensure that its permissions are set restrictively:

```
chmod 400 ./kafka-client.keytab
```

7. Create the jaas-keytab.conf file with the following content:

```
KafkaClient {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="./kafka-client.keytab"
  principal="srv_kafka-client@XYZ.SITE";
};
```

Where the value of the keyTab property is the path of the keytab copied to the host and the value of the principal property is the Kerberos principal name of the Machine User account. You can find the principal name by listing the content of the keytab, as shown in the following example:

```
$ klist -kt ./kafka-client.keytab
Keytab name: FILE:kafka-client.keytab
KVNO Timestamp           Principal
---- ------------------- ------------------------------------------------
------
   0 05/07/2020 03:32:01 srv_kafka-client@XYZ.SITE
```

```
    0 05/07/2020 03:32:01 srv_kafka-client@XYZ.SITE
```

8. Run the following commands to connect to Kafka, authenticating with the Kerberos keytab.

```
BROKER_HOST=$(hostname -f)

export KAFKA_OPTS="-Djava.security.auth.login.config=./jaas-keytab.conf"

kafka-console-consumer \
  --bootstrap-server $BROKER_HOST:9093 \
  --consumer.config client-kerberos.properties \
  --topic machine-data \
  --group machine-data-$RANDOM \
  --from-beginning
```

The KAFKA_OPTS environment variable is used to communicate the location of the the jaas-keytab.conf configuration file to kafka-console-consumer command.

Results

You see the content of the machine-data topic as follows:

```
Hello, Kafka!
```

Monitoring Kafka activity in Streams Messaging Manager

You can use Streams Messaging Manager (SMM) to monitor Kafka activity. To do this, you run two SSH sessions, one for producing data to a Kafka topic and the other to consume the data from the Kafka topic. Then you go to the SMM UI from the Management Console to track the producer and consumer activity.

About this task

Now that you know how to configure a Kafka client to connect to the Kafka cluster in CDP, generate some activity and monitor it using SMM. To do this, you collect machine usage data from the broker host using the vmstat command and stream that data into the machine-data topic.

For detailed and comprehensive information about SMM, see *Streams Messaging Manager*.

Procedure

1. Start by opening two SSH connections to the same broker host.
2. On the first session, run the following command.

```
BROKER_HOST=$(hostname -f)

vmstat 1 1000 | kafka-console-producer \
  --broker-list $BROKER_HOST:9093 \
  --producer.config client-ldap.properties \
  --topic machine-data
```

This command runs vmstat and sends each line from its output as one message every second to the machine-data topic.

This example assumes that you are using LDAP authentication but you can use the authentication method of your preference.

Leave the session running.

3. On the second SSH session, run the following command to consume the data:

```
BROKER_HOST=$(hostname -f)

kafka-console-consumer \
  --bootstrap-server $BROKER_HOST:9093 \
  --consumer.config client-ldap.properties \
  --topic machine-data \
  --group machine-data-$RANDOM \
  --from-beginning
```

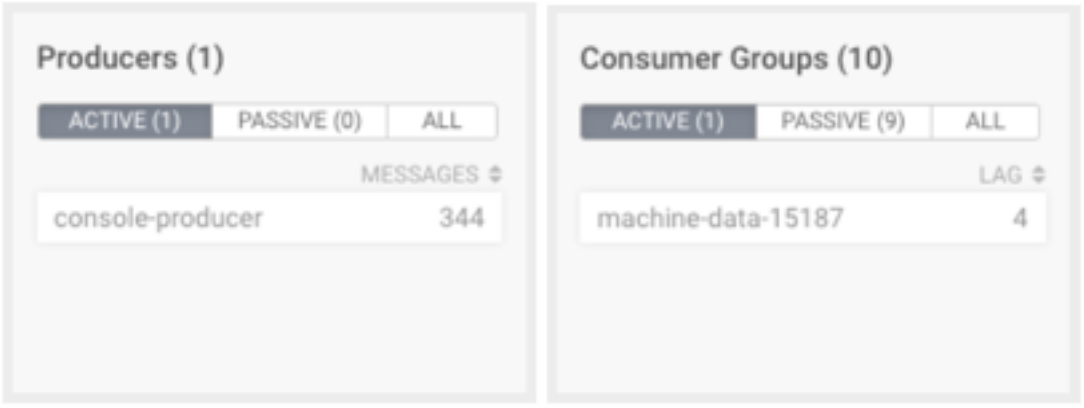
   You see the data produced by the first session continuously appear on the screen as output.

   Leave the second session also running.

4. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
5. On the Data Hubs tab of your environment, select the Kafka cluster you created.
6. Click Streams Messaging Manager on the services pane to open the SMM web UI.

   It may take a few minutes for the consumer or producer activity to start showing in the SMM UI. You may have to refresh your page a few times until the data starts to appear.

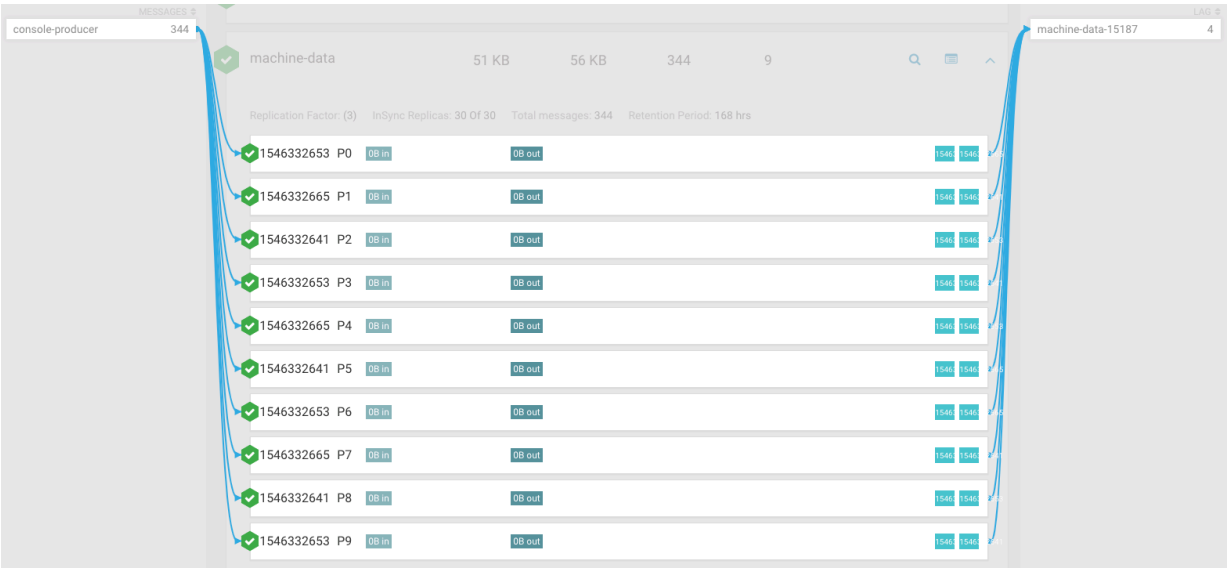
   Once it does, you see one active producer on the left-hand side of the page and at least one active consumer listed on the right-hand side of the page.



Note: SMM shows consumers and producers with some activity in the last few minutes, as active. Because of this, you may see those producers and consumers that finished recently, as active.

You also see the LAG metrics listed besides each consumer. This metric indicates the number of messages the consumer is behind the latest message produced to the Kafka topic. You can use this to quickly identify slow consumers that are lagging behind producers.

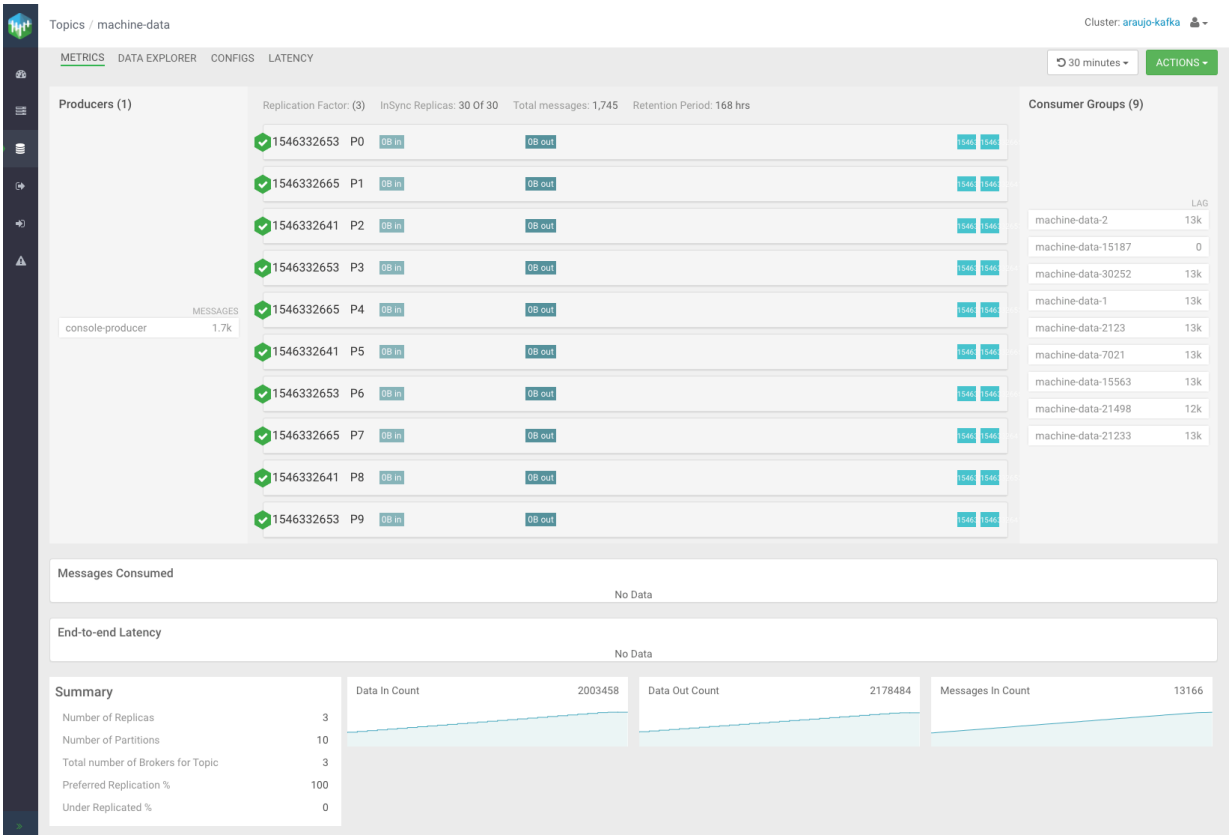
7. Click either on the producer or on the consumer, and SMM shows their activity, highlighting all the topics and partitions they are writing to or reading from, respectively.



In this page you can identify metrics for the overall Kafka topic activity and partition-level. The In Sync Replica (ISR) set for each partition can be identified on the screen. The ID of the LEADER replica for each partition is shown on the left-hand side, while the FOLLOWER replicas are represented as teal-colored boxes on the right-hand side.

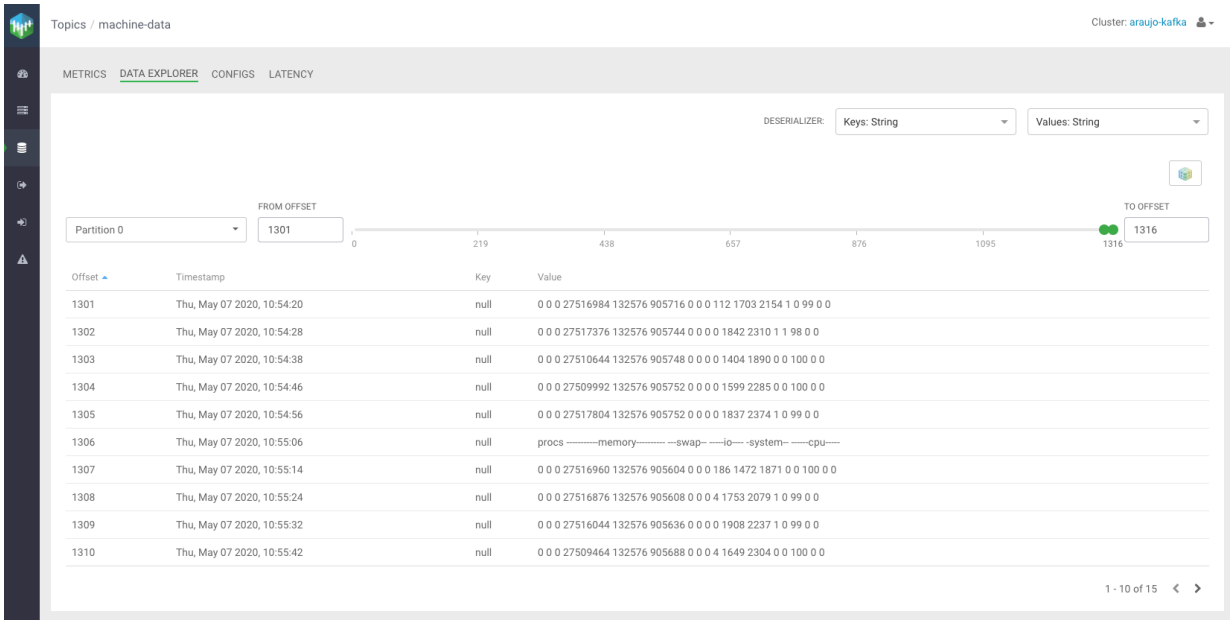
8. Click on an empty part of the page to clear the consumer or producer selection, and then click the profile icon for the machine-data topic.

This opens the Kafka topic details page where you can find all the topic-related metrics and utilization charts.



9. Click the DATA EXPLORER tab.

   In the Data Explorer page you can sample the data that is flowing through the Kafka topic. Select different partitions and adjust the offset sliders to become familiar with those controls.



Related Information

Streams Messaging Manager

Use Schema Registry

You can use a sample Kafka client application written in Java, to produce data to and consume data from Kafka topics using schemas stored in the Schema Registry.

Schema Registry provides a shared repository of schemas that allows applications to flexibly interact with each other. Applications frequently need a way to share metadata across data format, schema, and semantics. Schema Registry addresses these challenges by evolving schemas such that a producer and consumer can understand different versions of the schemas but still read all information shared between both versions and safely ignore the rest.

For more information on the topics covered in this section, check the following Cloudera online documentation resources:

- *Connecting Kafka clients to Data Hub provisioned clusters*
- *Schema Registry*
- *Developing Apache Kafka applications*

Note:

In this section, you build a sample Java application. To do this you must have the following installed on your local computer:

- Java 8 SDK (or later)
- Maven
- Git

Related Information

Connecting Kafka clients to Data Hub provisioned clusters

Schema Registry

Developing Apache Kafka applications

Gather configuration information

To produce data to and consume data from Kafka topics using schemas stored in the Schema Registry, you need to collect some configuration information such as the list of brokers and Schema Registry endpoint. You must also create TLS truststore.

Besides the Machine User name and password that you used in previous examples, you also need the following additional information to configure the client application:

- Broker list

  Because the Kafka client application is a remote application running on your computer, you need to configure it with a list of brokers to which the application can connect.
- Schema Registry endpoint

  You need to provide the application with the Schema Registry endpoint so that the application can store and retrieve schemas from it.
- TLS truststore

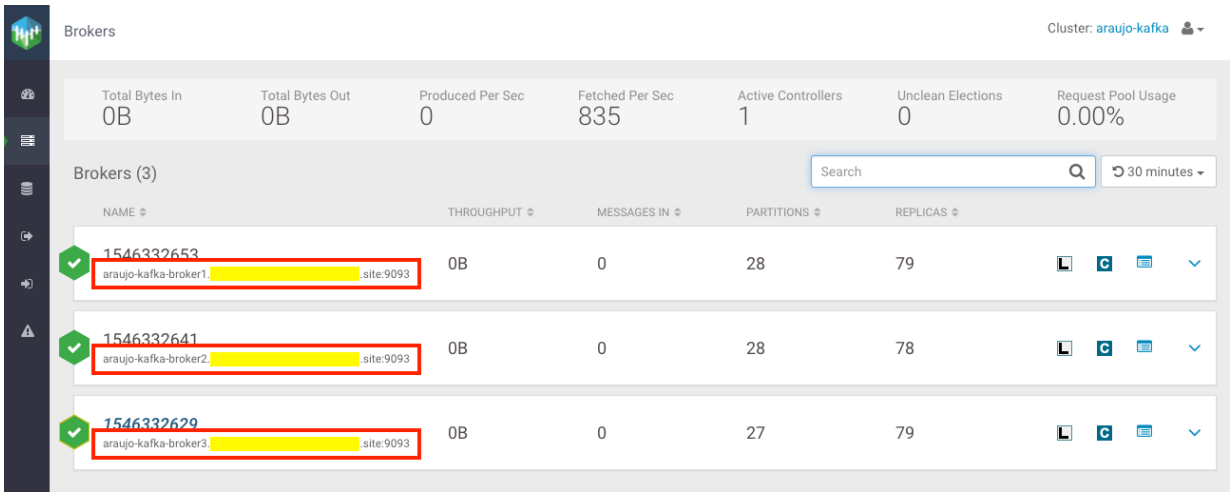
  Because this is a remote application connecting to a secure CDP cluster, for which TLS encryption is enabled, you need to provide the client with a truststore that can be used to validate the cluster certificates to establish secure communication channels.

Finding list of brokers

Learn how to find the list of brokers in your Kafka cluster from the Brokers page in the Streams Messaging Manager (SMM) UI.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Streams Messaging Manager on the services pane to open the SMM web UI.
4. Click the Brokers tab ( ).
5. Take note of the broker endpoints (host and port) listed in the Brokers page.



Note: If your cluster contains more than three brokers, you need to select a few to provide to the client. Pick only three of them.

Finding Schema Registry endpoint

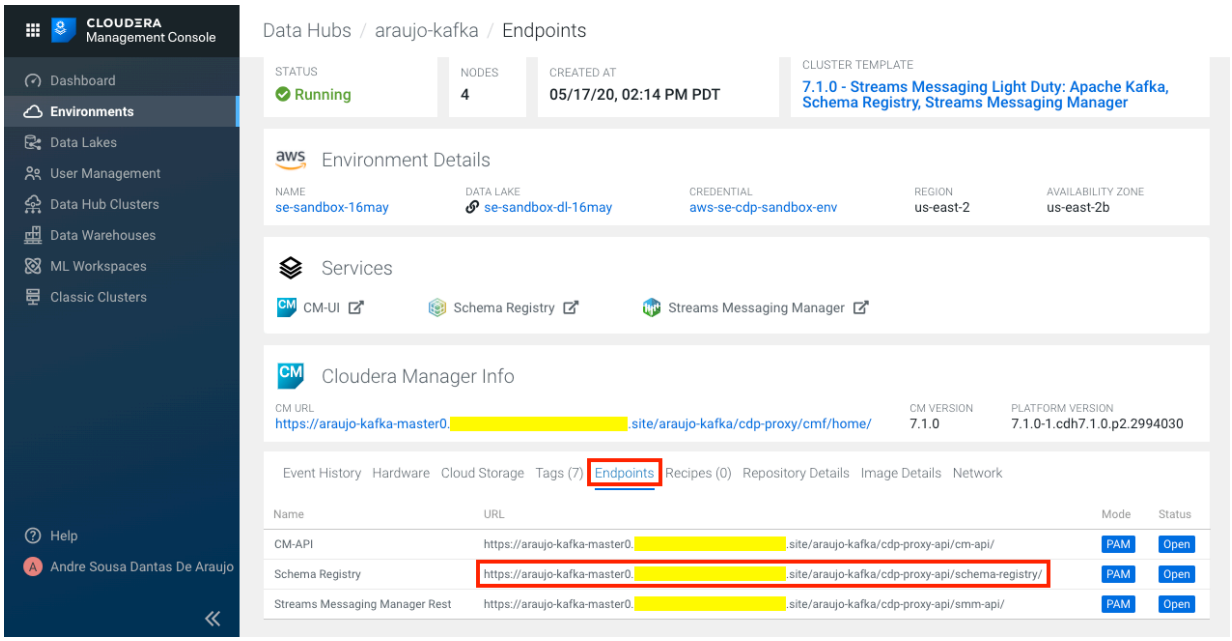
You can find the Schema Registry endpoints by selecting the Kafka cluster, you created, from the Management Console.

Before you begin

You must have created a list of brokers in your Kafka cluster.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click the Endpoints tab and make a note of the Schema Registry endpoint.



Creating TLS truststore

You can create a TLS truststore that the Kafka client needs to successfully connect to the secure Kafka cluster. You need to go to the environment where your Kafka cluster is running, download the FreeIPA certificate file, and create the TLS truststore by using the certificate file.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click the Summary tab, scroll to the FreeIPA section, and then click Actions Get FreeIPA Certificate .

   This downloads the FreeIPA certificate file (<environment_name>.crt) to your computer.
3. Run the following command to create the truststore:

```
keytool \
    -importcert \
    -noprompt \
    -storetype JKS \
    -keystore truststore.jks \
    -storepass changeit \
    -alias freeipa-ca \
```

```
    -file /path/to/ca.crt
```

Defining Schema Registry access policies

You must grant your application certain privileges to use the Schema Registry by creating the appropriate policies in Ranger. Learn how to define access policies and permissions for using Schema Registry.

About this task

In this example, you want to allow the application to create and manage schemas in Schema Registry. You can adjust the permissions granted to the applications in your environment to avoid granting privileges that are not necessary for each use case.

Before you begin

You must have gathered configuration information for using Schema Registry.

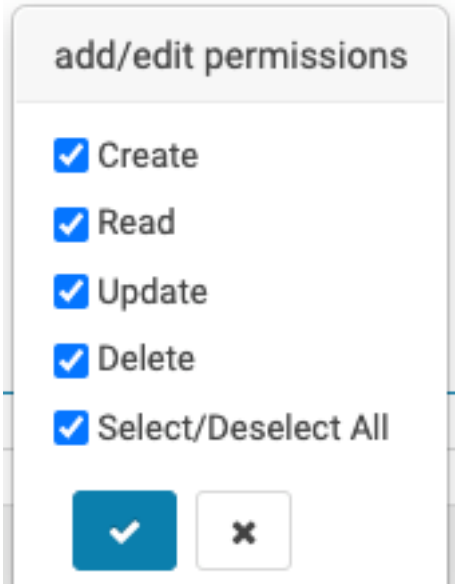
Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click the Ranger icon (

) on the top pane, to open the Ranger web UI.
3. On the Ranger UI, click Access Manager Resource Based Policies .
4. Under the SCHEMA REGISTRY group, select the policy associated to your Schema Registry service.

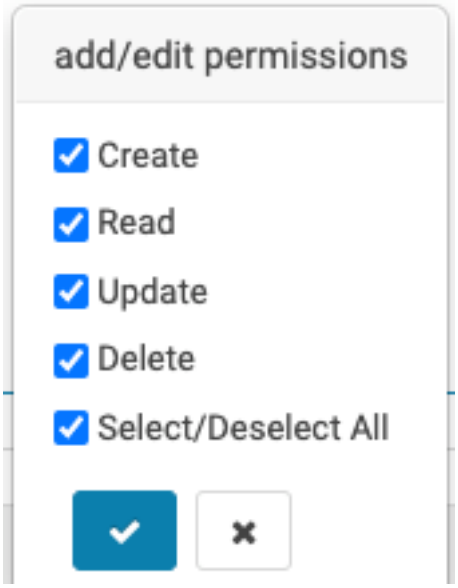
   To select it, click on the policy name, not on the icons. You should see the list of predefined policies for your Schema Registry.

5. Create a policy to allow access to the schema metadata.
   a) Click Add New Policy, to create a new one, and enter the following details:
      - Policy Name: Machine Data Schema Metadata
      - schema-group: kafka
      - Schema Name: machine-data-avro
      - schema-branch: Click on the schema-branch option and select none
      - Description: Access to machine-data-avro schema metadata
   b) Below the Allow Conditions section, click on the empty Select Users box and select the Machine User you created previously.
   c) Click Add Permissions, select all the permissions and click on the tick mark button.



   d) Scroll to the bottom of the page and click Add to save the policy.

6. Create a policy to allow access to the schema versions:
   a) Click Add New Policy, to create a new one, and enter the following details:
      - Policy Name: Machine Data Schema Versions
      - schema-group: kafka
      - Schema Name: machine-data-avro
      - schema-branch: MASTER
      - schema-version: *
      - Description: Access to machine-data-avro schema versions
   b) Below the Allow Conditions section, click on the empty Select Users box and select the Machine User you created previously.
   c) Click Add Permissions, select all the permissions and click on the tick mark button.



   d) Scroll to the bottom of the page and click Add to save the policy.

Producing data in Avro format

You can run the sample producer application to produce data to a Kafka topic in Avro format. You can send data to a Kafka topic in Avro format. Learn how to store a schema in CDP#s Schema Registry and use a simple Java Kafka client to send and read data using that schema.

Before you begin

You must have defined access policies for Schema Registry.

Procedure

1. On your computer, clone this repository using git and change to the kafka-client-avro directory.

```
git clone https://github.com/asdaraujo/cdp-examples
cd cdp-examples/kafka-client-avro
```

2. Build the client binaries.

```
mvn clean package
```

3. Create a copy of the producer properties template file.

```
cp src/main/resources/producer.properties.template producer.properties
```

4. Edit the producer.properties file and replace the following placeholders with the corresponding values:
   - [***MACHINE_USER_NAME***]

     The Machine User name (prefixed with srv_). Note that this needs to be replaced at two locations in the template.
   - [***MACHINE_USER_PASSWORD***]

     The Machine User's Workload Password. Note that this needs to be replaced at two locations in the template.
   - [***BROKER1***], [***BROKER2***], [***BROKER3***]

     The hostnames of three cluster brokers. Also, ensure that the broker port numbers match the numbers seen in the Streams Messaging Manager (SMM) Brokers page.
   - [***TRUSTSTORE_PATH***]

     The path of the truststore.jks file created previously.
   - [***SCHEMA_REGISTRY_ENDPOINT***]

     The Schema Registry endpoint found previously.
5. Run the producer application with the following command:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataProducer \
  ./producer.properties \
  ./src/main/avro/MachineData.v1.avsc
```

   The producer application (MachineDataProducer class) takes the following arguments:
   - The producer.properties file.
   - An Avro schema definition file that the producer uses to produce messages.

   If the producer is working correctly, you see several messages similar to the following:

```
... INFO  c.c.examples.MachineDataProducer - Successfully produced message
 to partition [machine-data-avro-8], offset [0]
```

6. Leave the SSH session open and the producer running for now.

Note: If you browse the producer code, you see that there is no need for explicit serialization of the Avro objects in the code. The objects are sent directly to the producer:

```
for (GenericData.Record data :newMachineDataCollector(schema)) {
    ProducerRecord<String, GenericData.Record> producerRecord =newP
roducerRecord<>(topicName, data);
    producer.send(producerRecord, newProducerCallback());
}
```

The serialization is handled automatically by the KafkaAvroSerializer class that you used to configure the Kafka client's value.serializer property in the properties file:

```
value.serializer=com.hortonworks.registries.schemaregistry.serdes
.avro.kafka.KafkaAvroSerializer
```

Checking schema registration

When the KafkaAvroSerializer is used to produce data to a Kafka topic, the serializer first checks with the configured Schema Registry service to ensure that the schema is compatible with the one registered for that Kafka topic. You can view the registered schema and its the versions in the Schema Registry web UI.

About this task

If the Kafka topic does not have the schema registered yet, the serializer registers the schema on behalf of the client, provided it has permissions to do so. If there is a schema registered for the Kafka topic, the serializer checks if the schema being used by the client is either the same or a compatible version of that schema.

If the schema is the same, the producer uses it for producing messages. If the schema is a new version of the currently registered schema that is compatible with the previous versions, the serializer registers it with a new version number and then uses it for producing messages. If the schema being used by the client is not compatible with the currently registered schema, the serializer issues an exception.

In this example, because this is the first time that you used that topic, there was previously no schema registered for it. If everything works as expected, the schema defined in the MachineData.v1.avsc file is successfully registered in Schema Registry.

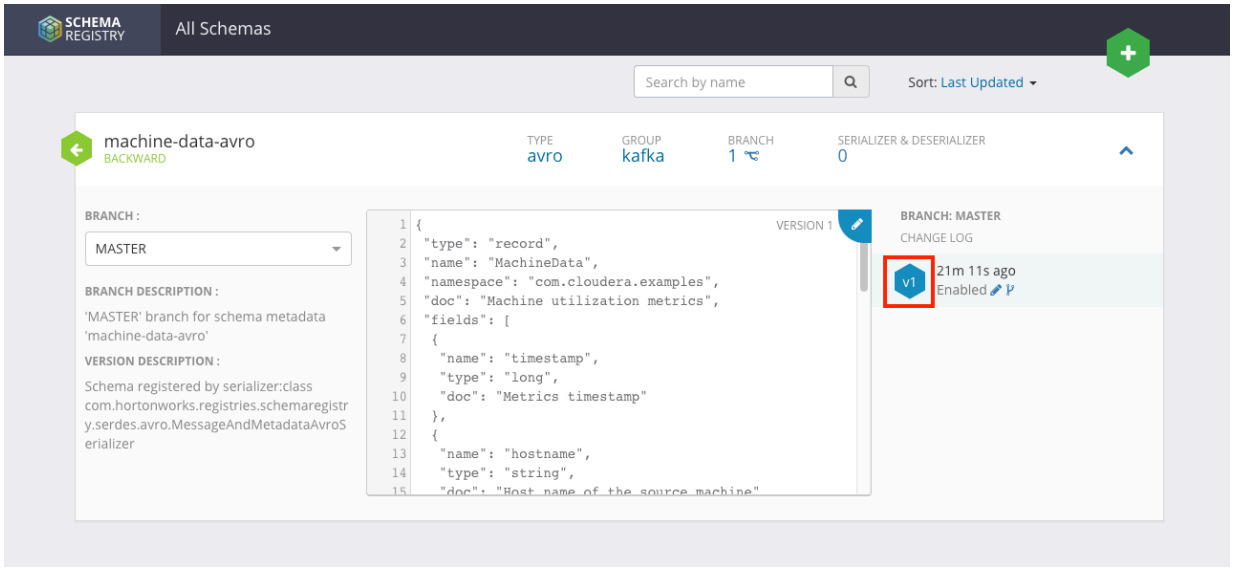
Before you begin

You must have produced data in Avro format.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Schema Registry on the Services pane to open the Schema Registry web UI.

   One schema registered with the name of the topic being used is displayed.
4. Click on the machine-data-avro schema to expand:



In this page, you can view all the versions of the schema. You only see version v1, which is the version just registered by the producer's KafkaAvroSerializer.

Checking producer activity

You can also use Streams Messaging Manager (SMM) to check the activity generated by the producer on the Kafka cluster. Learn how to check the producer activity in SMM after you send data to a Kafka topic in Avro format.

Before you begin

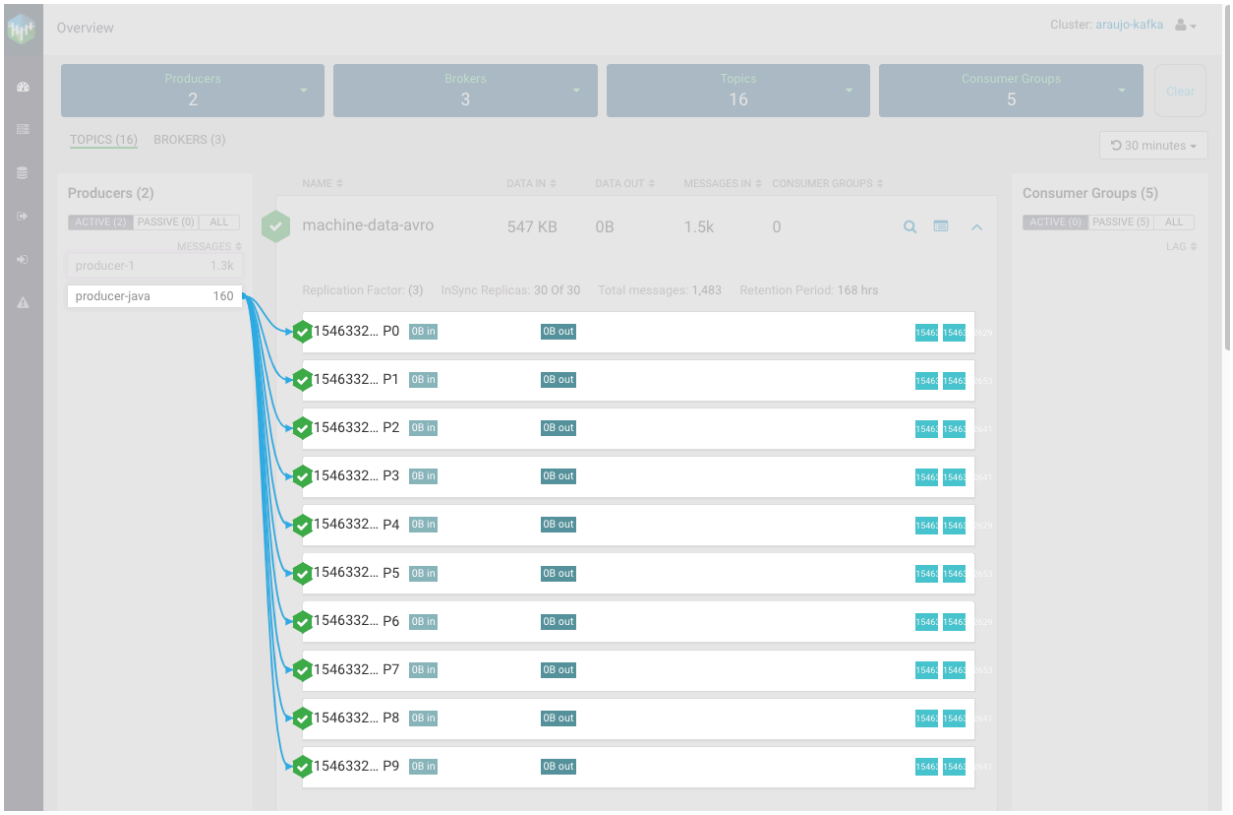
You must have sent data to a Kafka topic in Avro format and checked the schema registration.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Streams Messaging Manager on the Services pane to open the SMM web UI.
4. On the SMM UI, click the Overview tab ( ).
5. Click the Topics dynamic filter, type machine in the Search field, and select the checkbox next to the Name header to include all the filtered topics.

   You should be able to see the machine-data-avro topic and verify that it has some inbound activity (DATA IN > 0).
6. The producer.properties file sets the producer name with the property client.id=producer-java. Find this producer from the list and click on it.

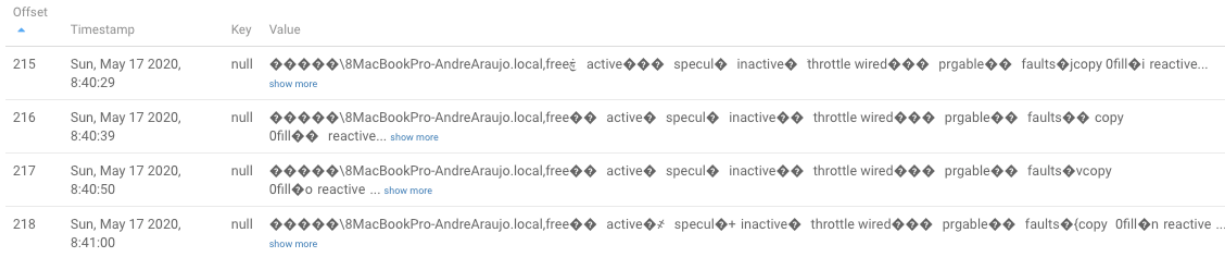
   All the partitions that are receiving data from the producer is displayed.



7. Click on an empty part of the page to clear the consumer or producer selection, and then click the profile icon ( ) for the machine-data-avro topic to open the topic page.

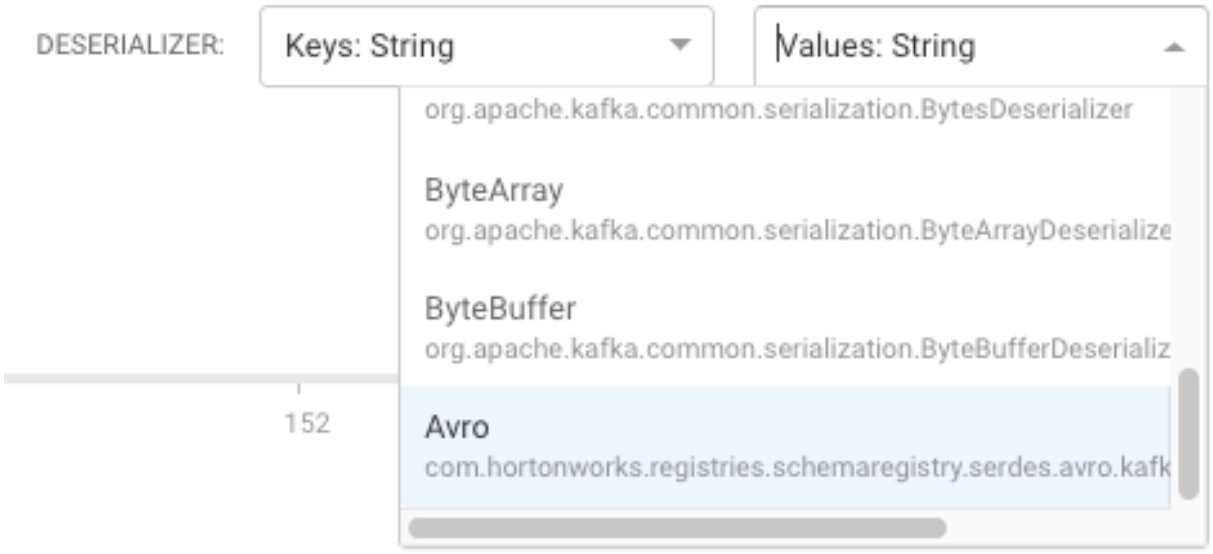
8. Click the DATA EXPLORER tab.

   Because the data you are ingesting now is binary data (Avro serialization), you see that the message contents shown by SMM contain a lot of garbage:

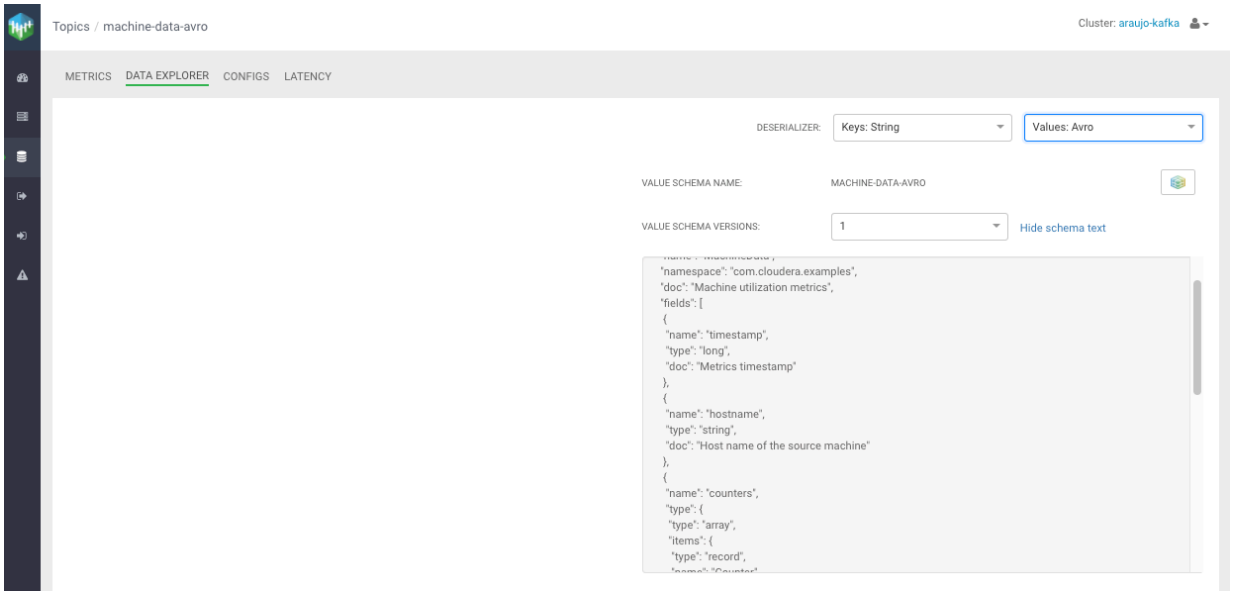


   SMM integrates with Schema Registry and is able to decode the binary messages using the schema registered previously by the client.

9. Click the Values drop-down for DESERIALIZER, and select the Avro format.



   SMM automatically retrieves the correct schema from the Schema Registry and deserializes the messages shown in SMM, showing them as JSON strings:



Consuming data from Kafka topics using stored schemas

You can consume data from the machine-data-avro topic where you have produced data in Avro format. You need to create a copy of the consumer properties template file, replace the values of some properties in that file, and run the consumer application.

About this task

While the producer is running on the original terminal, open a new terminal and start consuming data from the mach ine-data-avro topic.

Before you begin

You must have produced data to a Kafka topic in Avro format.

Procedure

1. On the second terminal, ensure that you are on the same directory you used before.

```
cd cdp-examples/kafka-client-avro
```

2. Create a copy of the consumer properties template file.

```
cp src/main/resources/consumer.properties.template consumer.properties
```

3. Edit the consumer.properties file and replace the following placeholders with the respective values:
   - [***MACHINE_USER_NAME***]

     The Machine User name (prefixed with srv_). Note that you need to replace the machine username at two locations in the template.
   - [***MACHINE_USER_PASSWORD***]

     The Machine User's Workload Password. Note that you need to replace the machine user password at two locations in the template.
   - [***BROKER1***], [***BROKER2***], [***BROKER3***]

     The hostnames of three cluster brokers. Also, ensure that the broker port numbers match the numbers seen on the Brokers page in SMM.
   - [***TRUSTSTORE_PATH***]

     The path of the truststore.jks file created in the previous section.
   - [***SCHEMA_REGISTRY_ENDPOINT***]

     The Schema Registry endpoint discovered in a previous section.

   The consumer application (MachineDataConsumer class) takes one argument which is the consumer.properties file.

   Unlike the producer application, the consumer does not require a schema to be provided. Because the schema is already registered in Schema Registry by the producer, the consumer can retrieve the correct schema from the registry.

   Each message send to Kafka by the producer has a few additional bytes that contain the reference to the correct schema version to be used from Schema Registry. These bytes are either prefixed to the message payload or, optionally, stored in the message header.
4. Run the consumer application with the following command:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataConsumer \
```

```
  ./consumer.properties
```

If the consumer is working correctly, you should see several messages similar to the following:

```
... INFO  c.c.examples.MachineDataConsumer - Consumed 1 records
... INFO  c.c.examples.MachineDataConsumer - Received message: (null, {"
timestamp": ..., }) at partition [machine-data-avro-4], offset [407], with
 headers: [RecordHeader(key = value.schema.version.id, value = [3, 0, 0, 0
, 1])]
```

Note:

- The consumer group being used by the consumer application, configured in the consumer.properties file as group.id=machine-data-1 must match the pattern configured in the Consumer Group policy in Ranger at the beginning of this tutorial. If you chose a consumer group that does not match the policy, the consumer application fails with an authorization error.
- The producer properties template configured the producer to add the schema reference in the message header (store.schema.version.id.in.header=true). You can see this reference in the consumer output with key value.schema.version.id and value [3, 0, 0, 0, 1]. The first byte (3) is the protocolId used by the KafkaAvroSerializer and the next 4 bytes are an integer representing the schema version ID (1). Note that this is not the actual version number, but an internal Schema Registry ID that identifies that schema version object.

5. Go to Streams Messaging Manager (SMM) and verify that now you can see a consumer reading from the topic, besides the producer you had seen before.
6. Stop the producer and consumer.

Monitor end-to-end latency

With Streams Messaging Manager (SMM) you can monitor end-to-end latency for your applications, which is the time taken by a consumer to consume a message that is produced in a Kafka topic.

To find more about end-to-end latency, see *End to end latency overview*.

Learn how to enable end-to-end latency monitoring for your sample Java application.

Related Information

End to end latency overview

Setting up authorization policies

For Streams Messaging Manager (SMM) to show end-to-end latency data for an application, configure the application to generate additional metrics and enable monitoring. This includes sending metadata to internal SMM Kafka topics.

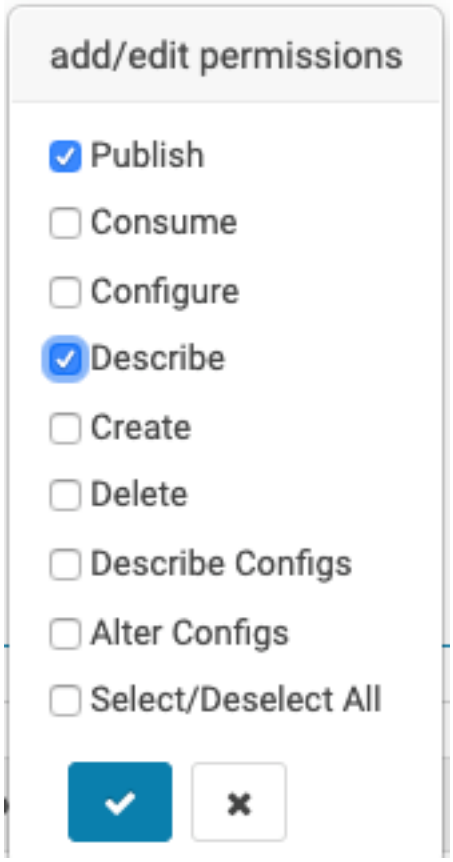
About this task

To do this in a secure Kafka cluster you must grant permissions on those topics through Ranger. Perform the following steps to grant the permissions to the Machine User account you are using:

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click the Ranger icon ( ) on the top pane, to open the Ranger web UI.
3. On the Ranger UI, click Access Manager Resource Based Policies .

4. Under the KAFKA group, select the policy of your Kafka cluster.
   To select it, click on the policy name, and not on the icons.
5. Click Add New Policy, to create a new one, and enter the following details:
   - Policy Name: SMM Metrics topics
   - Topic: __smm_consumer_metrics and __smm_producer_metrics
   - Description: SMM topics for end-to-end latency monitoring
6. Under the Allow Conditions section, click on the empty Select Users box and select the Machine User you created previously.
7. Click Add Permissions, select the Publish and Describe permissions, and click the tick mark.



8. Scroll to the bottom of the page and click Add to save the policy.

Enabling end-to-end latency monitoring

Learn how to enable end-to-end latency monitoring of Kafka topics using Streams Messaging Manager (SMM). You need to edit the consumer.property and producer.property files, start the producer in one terminal and the consumer on another, and then open the SMM web UI for monitoring end-to-end latency.

About this task

To enable the application start logging end-to-end latency metrics you use an interceptor for the Kafka client, which can be set through the configuration file.

Perform the following steps to enable the monitoring for the sample Java application and verify the latency in SMM.

Before you begin

You must have set Ranger authorization policies.

Procedure

1. Edit the consumer.properties file and uncomment the following line:

```
interceptor.classes=com.hortonworks.smm.kafka.monitoring.interceptors
.MonitoringConsumerInterceptor
```

2. Edit the producer.properties file and uncomment the following line:

```
interceptor.classes=com.hortonworks.smm.kafka.monitoring.interceptors
.MonitoringProducerInterceptor
```

3. Start the producer on one terminal window with the following command:

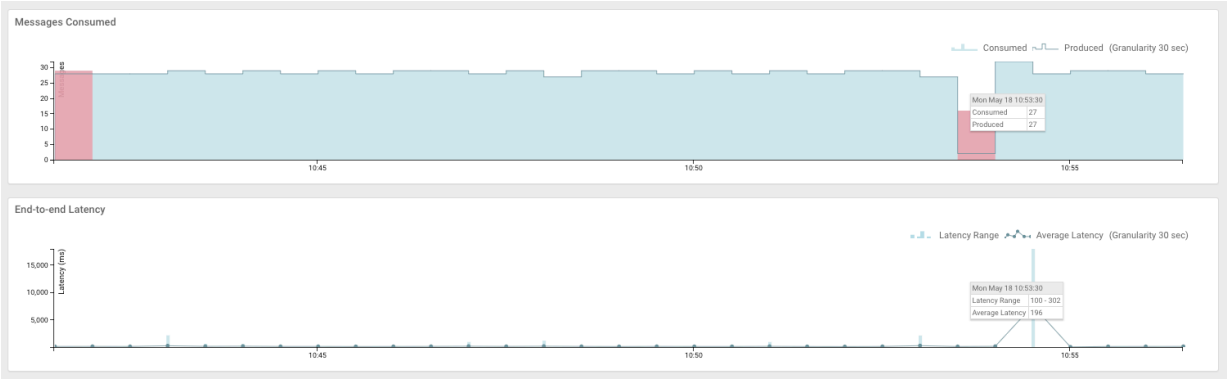
```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataProducer \
  ./producer.properties \
  ./src/main/avro/MachineData.v1.avsc
```

4. Start the consumer on another terminal window with the following command:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataConsumer \
  ./consumer.properties
```

5. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
6. On the Data Hubs tab of your environment, select the Kafka cluster you created.
7. Click Streams Messaging Manager on the Services pane to open the SMM web UI.
8. On the SMM UI, click Overview ( ).
9. Click on the profile icon ( ) for the machine-data-avro topic to open the topic page.

   You see data in the charts Messages Consumed and End-to-end latency. You may have to wait a few seconds until the charts appear.



Results

Refresh the page after some time to view the updated data.

Evolve your schema

You can update the schema to keep track of the type of operating system (OS) from the host(s) you are collecting data. After you update the schema, you need to reconfigure the consumer and producer to enable them to use the updated schema.

You have the application producing and consuming data and being monitored successfully, you identified a new requirement for your application and now you want to keep track of the type of OS from the host(s) you are collecting data.

For this, you must add the following field to your schema:

```
{
  "name": "os_type",
  "doc": "OS type of the source machine",
  "type": "string",
  "default": "UNKNOWN"
}
```

Because this is a new field in the schema, you need to specify a default value, as shown above, so that you can maintain backward compatibility, which is being enforced by Schema Registry. If you do not do this, Schema Registry does not allow the new schema to be used. You can check the complete modified schema in the https://github.com/asdaraujo/cdp-examples/blob/master/src/main/avro/MachineData.v2.avsc file.

This section covers the modification of the schema and verification of what changed in Schema Registry and Streams Messaging Manager (SMM).

Reconfiguring the Kafka consumer

After you update a schema stored in the Schema Registry, you must reconfigure the consumer to use the new schema.

About this task

Because the latest schema version is backward compatible, it can be used to deserialize messages produced with the older versions of the schema. Hence, if you first reconfigure the consumer to use the new version of the schema, it can continue handling messages serialized with the old schema. Furthermore, it can also handle the new message format as soon as the producer starts sending them. By reconfiguring the consumers first, you avoid incompatibility between producers and consumers.

Perform the following steps to get the consumer to use the new schema:

Procedure

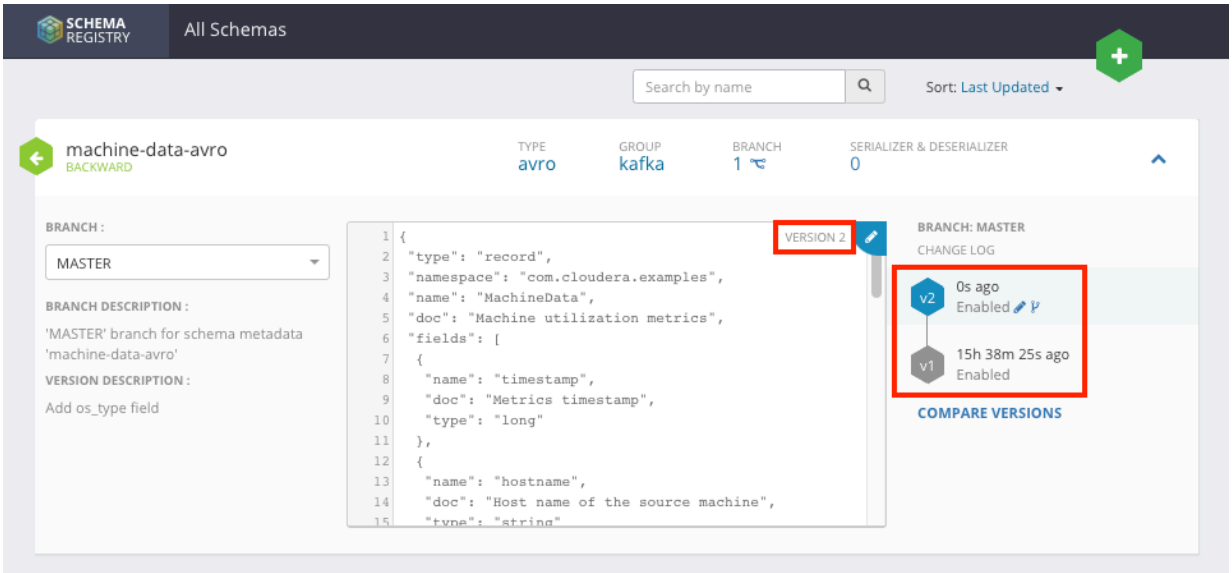
1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Schema Registry on the Services pane to open the Schema Registry web UI.
4. Click on the machine-data-avro schema to expand it, and then click the pencil icon to edit the schema.
5. In the DESCRIPTION field enter a description for this modification: Add os_type field.
6. In the SCHEMA TEXT field, click CLEAR.

   This displays the BROWSE button.

7. Click BROWSE and select the schema definition file for the new version of the schema, located at cdp-examples/kafka-client-avro/src/main/avro/MachineData.v2.avsc, and then click SAVE.

   Schema Registry performs compatibility validations on the new schema and if the outcome is positive, it saves the schema and displays a success message: Schema validated successfully.

   You can view both versions of the schema in Schema Registry.



   Now that you have the new version of the schema stored in Schema Registry, you need to reconfigure the consumer to use it.

8. If the consumer application is still running, stop it.
9. Edit the consumer.properties file, uncomment the following line and ensure it refers to the correct version number for the schema, as shown in the previous screenshot.

```
schemaregistry.reader.schema.versions={"machine-data-avro": 2}
```

   This property is a JSON representation of a Map, associating a topic name to the version of the schema that the consumer should use.

10. Start the consumer on one terminal window with the same command used earlier:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataConsumer \
  ./consumer.properties
```

   The consumer displays messages which includes the new field, as shown in the following example:

```
... INFO  c.c.examples.MachineDataConsumer - Received message: (null, {.
.., "os_type": "UNKNOWN", ...}) at partition ...
```

Note: The value shown for the field is UNKNOWN. This is because the producer is unaware of the new schema and is not producing messages with that field. When the consumer deserializes the message, it uses the default value defined in the schema.

Reconfiguring the Kafka producer

After you update a schema stored in the Schema Registry, you must reconfigure the producer to use the new schema.

About this task

Now that the consumer is already aware of the new schema, you can reconfigure the producer to send messages using the new schema.

Your producer has been created in a way that the schema file argument is optional. If that file is not provided, the producer tries to retrieve the latest version of the schema from Schema Registry with the topic name as the schema key.

Because you already uploaded the new schema to the Schema Registry, you only need to restart the producer without the schema file argument. Perform the following steps to get the producer to use the new schema:

Before you begin

You must have reconfigured the consumer to use the new schema.

Procedure

1. If the producer application is still running, stop it.
2. Start the producer with the following command using the new schema:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataProducer \
  ./producer.properties
```

The producer displays messages with the correct values assigned to the new os_type field, as shown in the following example:

```
... INFO  c.c.examples.MachineDataConsumer - Received message: (null, {.
.., "os_type": "mac", ...}) at partition ...
```

What to do next

After you create your first streams messaging cluster in CDP Public Cloud, you can explore more about how Apache Kafka, Schema Registry, and Streams Messaging Manager (SMM) work in CDP Public Cloud.

For more information on Apache Kafka in CDP Public Cloud, see *Apache Kafka*.

For more information on Schema Registry in CDP Public Cloud, see *Schema Registry*.

For more information on Streams Messaging Manager in CDP Public Cloud, see *Streams Messaging Manager*.

Related Information

Apache Kafka
Schema Registry
Streams Messaging Manager