Integrating data from OpenShift Streams for Apache Kafka

Use data streamed through Red Hat OpenShift Streams for Apache Kafka
Red Hat OpenShift Data Science 1 Integrating data from OpenShift Streams for Apache Kafka

Use data streamed through Red Hat OpenShift Streams for Apache Kafka
Abstract

Learn how to use data streamed through Red Hat OpenShift Streams for Apache Kafka.
Table of Contents

PREFACE ......................................................................................................................... 3

PART I. USING DATA FROM OPENSPLIT STREAMS FOR APACHE KAFKA ......................... 4

CHAPTER 1. ENABLING SERVICES CONNECTED TO OPENSPLIT DATA SCIENCE .............. 5

CHAPTER 2. CREATING A KAFKA INSTANCE ..................................................................... 7

CHAPTER 3. CREATING A KAFKA TOPIC .......................................................................... 8

CHAPTER 4. CREATING A KAFKA SERVICE ACCOUNT .................................................... 9

CHAPTER 5. ASSIGNING ACCESS PERMISSIONS TO THE KAFKA SERVICE ACCOUNT .......... 10

CHAPTER 6. CONFIGURING YOUR NOTEBOOK SERVER FOR KAFKA ............................... 12

CHAPTER 7. EXAMPLE OF A KAFKA CONSUMER ........................................................... 14

CHAPTER 8. EXAMPLE OF A KAFKA PRODUCER ............................................................ 16

CHAPTER 9. LAUNCHING JUPYTERHUB AND STARTING A NOTEBOOK SERVER .............. 18

CHAPTER 10. ADDITIONAL RESOURCES ......................................................................... 20
PREFA E

See the following documents for service and life cycle information related to this release:

- OpenShift Data Science Service Definition
- OpenShift Data Science Life Cycle
PART I. USING DATA FROM OPENSIFHT STREAMS FOR
APACHE KAFKA

Red Hat OpenShift Streams for Apache Kafka is a publish-subscribe messaging system for creating fault-tolerant, real-time data feeds. Connecting Red Hat OpenShift Data Science to a Red Hat managed instance of OpenShift Streams for Apache Kafka lets data scientists integrate these data feeds into machine learning models and intelligent applications.

To use these data feeds in your notebooks on OpenShift Data Science, you must:

1. Enable the OpenShift Streams for Apache Kafka service on Red Hat OpenShift Data Science.

2. Create a Kafka instance and save its configuration details.

3. Provide information about your Kafka instance to the notebook server using environment variables.

4. Create a producer notebook and a consumer notebook to connect your notebook server to your Kafka instance.

You can then develop your machine learning model using data streamed to and from your Kafka instance.

When you are satisfied with your model, you can create an OpenShift application and provide information about your Kafka instance as environment variables in the Deployment configuration section.

Prerequisites

- You have created and configured a working Kafka instance in the OpenShift Streams for Apache Kafka web console. See Getting Started with Red Hat OpenShift Streams for Apache Kafka for details.

- You can log in to the web consoles for OpenShift Dedicated and OpenShift Streams for Apache Kafka.

- You have credentials for logging in to JupyterHub on OpenShift Data Science.
CHAPTER 1. ENABLING SERVICES CONNECTED TO OPENSHIFT DATA SCIENCE

You must enable SaaS-based services, such as Red OpenShift Streams for Apache Kafka and Anaconda, before using them with Red Hat OpenShift Data Science. On-cluster services are enabled automatically.

Typically, you can install services, or enable services connected to OpenShift Data Science using one of the following methods:

- Enabling the service from the Explore page on the OpenShift Data Science dashboard, as documented in this procedure.

- Installing the service’s operator from OperatorHub. OperatorHub is a web console for cluster administrators to discover and select Operators to install on their cluster. It is deployed by default in OpenShift Container Platform (Installing from OperatorHub using the web console).

  **NOTE**
  Deployments containing operators installed from OperatorHub may not be fully supported by Red Hat.

- Installing the service’s operator from Red Hat Marketplace (Install operators).

- Installing the service as an Add-on to your Red Hat OpenShift Dedicated cluster using Red Hat OpenShift Cluster Manager (Installing OpenShift Data Science on OpenShift Dedicated).

For most services, the service endpoint is available on the service’s tile on the Enabled page of OpenShift Data Science. Certain services cannot be accessed directly from their tiles, for example, OpenVINO and Anaconda provide notebook images for use in JupyterHub and do not provide an endpoint link from their tile. Additionally, for services such as OpenShift Streams for Apache Kafka, it may be useful to store these endpoint URLs as environment variables for easy reference in a notebook environment.

Some independent software vendor (ISV) applications must be installed in specific OpenShift Data Science Add-on namespaces. However, do not install ISV applications in namespaces associated with OpenShift Data Science Add-ons unless you are specifically directed to do so on the application’s card on the dashboard.

To help you get started quickly, you can access the service’s learning resources and documentation on the Resources page, or by clicking the relevant link on the service’s tile on the Enabled page.

Prerequisites

- You have logged in to OpenShift Data Science.

- Your administrator has installed or configured the service on your OpenShift Dedicated cluster.

Procedure

1. On the OpenShift Data Science home page, click Explore. The Explore page opens.

2. Click the card for Red Hat OpenShift Streams for Apache Kafka
3. Click **Enable** on the drawer for the service.

4. If prompted, enter the service’s key and click **Connect**.

5. Click **Enable** to confirm that you are enabling the service.

**Verification**

- The service that you enabled appears on the **Enabled** page.
- The service endpoint is displayed on the service’s tile on the **Enabled** page.
CHAPTER 2. CREATING A KAFKA INSTANCE

To connect Red Hat OpenShift Data Science to Red Hat OpenShift Streams for Apache Kafka, you must create and configure a Kafka instance.

A Kafka instance in OpenShift Streams for Apache Kafka includes a Kafka cluster, bootstrap server, and the configurations needed to connect to producer and consumer services. After you have created the Kafka instance, you can obtain the connection information required to connect to Red Hat OpenShift Streams for Apache Kafka.

Prerequisites

- You have enabled Red Hat OpenShift Streams for Apache Kafka on your Red Hat OpenShift Dedicated cluster.
- You have logged in to the OpenShift Streams for Apache Kafka web console at https://console.redhat.com/beta/application-services/streams/.

Procedure

1. In the OpenShift Streams for Apache Kafka web console, go to Streams for Apache Kafka → Kafka Instances and click Create Kafka instance. The Create a Kafka instance page opens.

2. Enter a unique Instance name.

3. Select your cloud provider.

4. Select the cloud region from the list.

5. Click Create instance. The new Kafka instance is displayed in the Kafka Instances table.

6. Click the action menu (⋮) and select Connection from the list. A drawer opens with details of your instance's connection information.

7. Obtain the connection information.

   a. Copy the Bootstrap server endpoint to a secure location. This is the bootstrap server endpoint that you’ll need for connecting to this Kafka instance.

   b. For the Authentication method, copy the SASL/PLAIN Token endpoint URL to a secure location. This is the endpoint that you’ll use with your service account credentials to authenticate the connection to this Kafka instance.

Verification

- The instance Status is Ready.
- The instance’s bootstrap server and token endpoint URL are available.
CHAPTER 3. CREATING A KAFKA TOPIC

After you create a Kafka instance, you can create Kafka topics to start producing and consuming messages from your Red Hat OpenShift Data Science applications.

Prerequisites

- You have enabled Red Hat OpenShift Streams for Apache Kafka on your OpenShift Dedicated cluster.
- You have logged in to the OpenShift Streams for Apache Kafka web console at https://console.redhat.com/beta/application-services/streams/.
- You have created a Kafka instance and the instance is in Ready state.

Procedure

1. In the OpenShift Streams for Apache Kafka web console, go to Streams for Apache Kafka → Kafka Instances.
2. Click the name of the Kafka instance that you want to add a topic to.
3. Select the Topics tab, click Create topic and follow the guided steps to define the topic details. Click Next to complete each step and click Finish to complete the setup.
   - **Topic name**: Enter a unique topic name.
   - **Partitions**: Set the number of partitions for this topic.
   - **Message retention**: Set the message retention time and size to the relevant value and increment.
   - **Replicas**: For this release of OpenShift Streams for Apache Kafka, replicas are preconfigured. After you complete the topic setup, the new Kafka topic is listed in the topics table.

Verification

- The new Kafka topic is listed in the Topics table.
CHAPTER 4. CREATING A KAFKA SERVICE ACCOUNT

To connect your OpenShift Data Science to a Kafka instance in the OpenShift Streams for Apache Kafka web console, create a service account and associate it with the Kafka instance.

Prerequisites

- You have enabled Red Hat OpenShift Streams for Apache Kafka on your Red Hat OpenShift Dedicated cluster.
- You have logged in to the OpenShift Streams for Apache Kafka web console at https://console.redhat.com/beta/application-services/streams/.
- You have created a working Kafka instance.
- You have the bootstrap server endpoint for the instance and the authentication token endpoint available and saved to a secure location.

Procedure

1. Navigate to the Hybrid Cloud Console (http://console.redhat.com) and click Application Services. The OpenShift Application Services web console opens.
2. Click Service Accounts. The Service Accounts page opens.
3. Click Create service account. The Create a service account dialog opens.
4. Enter a short, descriptive name for your service account, and click Create.
5. Copy the Client ID and Client Secret to a secure location. Your application requires these credentials to authenticate the connection to the Kafka instance.
6. Select the I have copied the client ID and secret check box.
7. Click Close. The new service account is displayed in the Service Accounts table.

Verification

- The bootstrap server, client credentials, and authentication token endpoint are saved to a secure location.
- Your service account was successfully created on the Service Accounts page in the OpenShift Application Services web console.
CHAPTER 5. ASSIGNING ACCESS PERMISSIONS TO THE KAFKA SERVICE ACCOUNT

To permit your service account to interact with your Kafka instance, and any associated Kafka resources, you must configure the Kafka instance’s Access Control List (ACL). Access Control Lists (ACL) contain a set of permissions that manage how applications associated with the service account interact with your Kafka resources. You can set these permissions to allow applications to create and delete topics in the instance, to produce and consume messages in any topic in the instance, or to use any consumer group and any producer.

Prerequisites

- You have enabled Red Hat OpenShift Streams for Apache Kafka on your Red Hat OpenShift Dedicated cluster.
- You have logged in to the OpenShift Streams for Apache Kafka web console at https://console.redhat.com/beta/application-services/streams/.
- You have created a Kafka service account.
- You have created a Kafka instance and the instance is in Ready state.

Procedure

1. In the OpenShift Streams for Apache Kafka web console, go to Streams for Apache Kafka→Kafka Instances.

2. Click the name of the Kafka instance that you want your service account to access.

3. Click the Access tab.
   The Kafka instance’s Access Control List (ACL) appears.

4. Click the Manage access button.
   The Manage access dialog opens.

5. Select the service account that you want to interact with the Kafka instance from the Account drop-down menu.

6. Click Next.
   The Manage access dialog refreshes to display existing permissions assigned to the selected service account.

7. In the Assign permissions section, use the drop-down menus to assign permissions to resources associated with the service account.
   a. Select the resource type and associated conditions from the Resource drop-down menus.
   b. Select the access permission and associated operation from the Permissions drop-down menus.
   c. Optional: Click Add to add another set of permissions to a resource.

8. Click Save.

Verification
Your newly configured permissions appear in the Kafka instance’s Access Control List (ACL) on the Access tab.

Your service account can now interact with your resources as permitted.

Additional resources

- Managing account access in Red Hat OpenShift Streams for Apache Kafka
CHAPTER 6. CONFIGURING YOUR NOTEBOOK SERVER FOR KAFKA

To connect to a Red Hat OpenShift Streams for Apache Kafka instance, you need to store some information about the instance in your notebook server. Rather than hard-coding this information in your notebooks, you can store the information in environment variables, and import those environment variables into your notebooks.

Prerequisites

- You have enabled Red Hat OpenShift Streams for Apache Kafka on your Red Hat OpenShift Dedicated cluster.
- You have logged in to the OpenShift Streams for Apache Kafka web console at https://console.redhat.com/beta/application-services/streams/.
- You have created a working Kafka instance and service account.
- You have the bootstrap server endpoint for the instance, the client credentials, and the authentication token endpoint available and saved to a secure location.
- You have credentials for logging in to OpenShift Dedicated.
- If you are using specialized OpenShift Data Science groups, you are part of the user group (for example, rhods-users) in OpenShift Dedicated.

Procedure

1. Log in to JupyterHub and navigate to Starting a notebook server.
2. Configure JupyterHub to connect to your Kafka instance.
   a. Select the Notebook image to use for your server.
   b. Select the Container size for your server.
   c. Select and specify values for the Environment variables required to connect to OpenShift Streams for Apache Kafka. Use the connection information that you obtained earlier. The interface remembers previously entered environment variables
   d. Click Add more variables.
   e. In the Variable name field, enter KAFKA_BOOTSTRAP_SERVER.
   f. In the Variable value field, enter the value for the Kafka external server.
   g. Click Add more variables.
   h. In the Variable name field, enter KAFKA_USERNAME.
   i. In the Variable value field, enter the client ID number.
   j. Click Add more variables.
   k. In the Variable name field, enter KAFKA_PASSWORD.
I. Select the **Secret** check box.

m. In the **Variable value** field, enter the client secret.

n. Click **Start server**.
   The **Starting server** progress indicator appears. Depending on the deployment size and resources you requested, starting the server can take up to several minutes. After the server starts, the JupyterLab interface opens.

**Verification**

- Create a producer and a consumer for an existing Kafka topic to verify that your notebook server can work with OpenShift Streams for Apache Kafka.
CHAPTER 7. EXAMPLE OF A KAFKA CONSUMER

In Red Hat OpenShift Streams for Apache Kafka, a Kafka consumer subscribes to a topic and reads messages in that topic according to partition and offset. Here is an example of how to write a simple Kafka consumer that receives and prints messages. You can create this consumer on Red Hat OpenShift Data Science, using JupyterHub.

- Import and declare values for the location of the Kafka bootstrap server, the topic that the consumer subscribes to, and the consumer group that the consumer belongs to.

Example declaration of variable values

```python
import os
KAFKA_BOOTSTRAP_SERVER = os.environ.get('KAFKA_BOOTSTRAP_SERVER')
KAFKA_USERNAME = os.environ.get('KAFKA_USERNAME')
KAFKA_PASSWORD = os.environ.get('KAFKA_PASSWORD')
KAFKA_TOPIC = 'kafka-topic-name'
KAFKA_CONSUMER_GROUP = 'notebook-consumer-print'
```

The value for `KAFKA_TOPIC` must match the name of your topic in OpenShift Streams for Apache Kafka. You can name it explicitly, as shown here, or you can refer to a previously set `KAFKA_TOPIC` environment variable by using `os.environ.get('KAFKA_TOPIC')`.

- Create a Kafka consumer using the defined variables.
The following consumer uses the declared environment variables to provide the bootstrap server endpoint and the service account username and password for authentication using SASL.

Example function for creating a consumer

```python
from kafka import KafkaConsumer

def create_consumer_print():
    consumer = KafkaConsumer(KAFKA_TOPIC,
        group_id=KAFKA_CONSUMER_GROUP,
        bootstrap_servers=[KAFKA_BOOTSTRAP_SERVER],
        security_protocol='SASL_SSL',
        sasl_mechanism='PLAIN',
        sasl_plain_username=KAFKA_USERNAME,
        sasl_plain_password=KAFKA_PASSWORD,
        auto_offset_reset='earliest',
        api_version_auto_timeout_ms=30000,
        request_timeout_ms=450000)
    print(f'Subscribed to "{KAFKA_BOOTSTRAP_SERVER}" consuming topic "{KAFKA_TOPIC}"...')

    try:
        for record in consumer:
            msg = record.value.decode('utf-8')
            topic = record.topic
            print((f'Received the following message on the 
                f"topic "{topic}": {msg}'))

    finally:
```
print("Closing consumer...")
consumer.close()
print("Kafka consumer stopped.")

- Call the consumer creation function to create and start the consumer.

Example function for starting the consumer

```python
try:
    create_consumer_print()
except KeyboardInterrupt:
    print("Stopped")
```
In Red Hat OpenShift Streams for Apache Kafka, a producer sends messages to a broker topic to be written to a partition. Here is an example of a simple Kafka producer that sends JSON-formatted strings. You can create this producer on Red Hat OpenShift Data Science, using JupyterHub.

- Declare variables and values for the location of the Kafka bootstrap server and the name of the topic that the producer sends messages to.

Example declaration for variable values

```python
import os
KAFKA_BOOTSTRAP_SERVER = os.environ.get('KAFKA_BOOTSTRAP_SERVER')
KAFKA_USERNAME = os.environ.get('KAFKA_USERNAME')
KAFKA_PASSWORD = os.environ.get('KAFKA_PASSWORD')
KAFKA_TOPIC = 'kafka-topic-name'
```

The value for `KAFKA_TOPIC` must match the name of your topic in OpenShift Streams for Apache Kafka. You can name it explicitly, as shown here, or you can refer to a previously set `KAFKA_TOPIC` environment variable by using `os.environ.get('KAFKA_TOPIC')`.

- Create a Kafka producer. The following example producer uses the declared environment variables to provide the bootstrap server endpoint and the service account username and password for authentication using SASL.

Example function for creating a producer

```python
import json
import time
from kafka import KafkaProducer
def produce_messages(start=1, end=100, delay=1):
    """Sends a number of messages in JSON format '{"txt": "hello 1"}'
    Keyword arguments:
    start -- start number (default 0)
    end -- last number to send (default 100)
    delay -- number of seconds between messages (default 1)
    """
    # create the producer
    producer = KafkaProducer(bootstrap_servers=[KAFKA_BOOTSTRAP_SERVER],
                             security_protocol='SASL_SSL',
                             sasl_mechanism='PLAIN',
                             sasl_plain_username=KAFKA_USERNAME,
                             sasl_plain_password=KAFKA_PASSWORD,
                             api_version_auto_timeout_ms=30000,
                             max_block_ms=900000,
                             request_timeout_ms=450000,
                             acks='all')
    # send messages
    for x in range(start, end+1):
```

Red Hat OpenShift Data Science 1 Integrating data from OpenShift Streams for Apache Kafka

16
```python
time.sleep(delay)
jsonpayload = json.dumps({'txt': f'hello {x}'}))
print(f'sending {jsonpayload}')
producer.send(KAFKA_TOPIC, jsonpayload.encode('utf-8'))

producer.flush()  # Important, especially if message size is small
```

- Call the producer function with appropriate arguments to create and start the producer. The following call sends 100 messages with a two second delay.

**Example call to start the producer**

```
produce_messages(1, 100, 2)
```
CHAPTER 9. LAUNCHING JUPYTERHUB AND STARTING A NOTEBOOK SERVER

Launch JupyterHub and start a notebook server to start working with your notebooks.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- You know the names and values you want to use for any environment variables in your notebook server environment, for example, AWS_SECRET_ACCESS_KEY.
- If you want to work with a very large data set, work with your administrator to proactively increase the storage capacity of your notebook server.

Procedure

1. Locate the JupyterHub card on the Enabled applications page.
2. Click Launch application.
   a. If prompted, select your identity provider.
   b. Enter your credentials and click Log in (or equivalent for your identity provider).

   If you see Error 403: Forbidden, you are not in the default user group or the default administrator group for OpenShift Data Science. Contact your administrator so that they can add you to the correct group using Adding users for OpenShift Data Science.

   If you have not previously authorized the jupyterhub-hub service account to access your account, the Authorize Access page appears prompting you to provide authorization. Inspect the permissions selected by default, and click the Allow selected permissions button.

3. Start a notebook server.
   This is not required if you have previously opened JupyterHub.
   a. Select the Notebook image to use for your server.
   b. If the notebook image contains multiple versions, select the version of the notebook image from the Versions section.

   NOTE

   When a new version of a notebook image is released, the previous version remains available and supported on the cluster. This gives you time to migrate your work to the latest version of the notebook image.

   Notebook images can take up to 40 minutes to install. Notebooks images that have not finished installing are not available for you to select. If an installation of a notebook image has not completed, an alert is displayed.

   c. Select the Container size for your server.
   d. Optional: Select the Number of GPUs (graphics processing units) for your server.
IMPORTANT

Using GPUs to accelerate workloads is only supported with the PyTorch, TensorFlow, and CUDA notebook server images.

e. Optional: Select and specify values for any new **Environment variables**. For example, if you plan to integrate with Red Hat OpenShift Streams for Apache Kafka, create environment variables to store your Kafka bootstrap server and the service account username and password here.

The interface stores these variables so that you only need to enter them once. Example variable names for common environment variables are automatically provided for frequently integrated environments and frameworks, such as Amazon Web Services (AWS).

IMPORTANT

Ensure that you select the **Secret** checkbox for any variables with sensitive values that must be kept private, such as passwords.

f. Click **Start server**.

The **Starting server** progress indicator appears. If you encounter a problem during this process, an error message appears with more information. Click **Expand event log** to view additional information about the server creation process. Depending on the deployment size and resources you requested, starting the server can take up to several minutes. After the server starts, the JupyterLab interface opens.

**WARNING**

You can be logged in to JupyterHub for a maximum of 24 hours. After 24 hours, your user credentials expire, you are logged out of JupyterHub, and your notebook server pod is stopped and deleted regardless of any work running in the notebook server. To help mitigate this, your administrator can configure OAuth tokens to expire after a set period of inactivity. See **Configuring the internal OAuth server** for more information.

**Verification**

- The JupyterLab interface opens in a new tab.

**Additional resources**

- **Options for notebook server environments**

**Troubleshooting**

- If you see the "Unable to load notebook server configuration options" error message, contact your administrator so that they can review the logs associated with your JupyterHub pod and determine further details about the problem.
CHAPTER 10. ADDITIONAL RESOURCES

- Try Red Hat OpenShift Streams for Apache Kafka
- Creating an OpenShift application from a Git repository