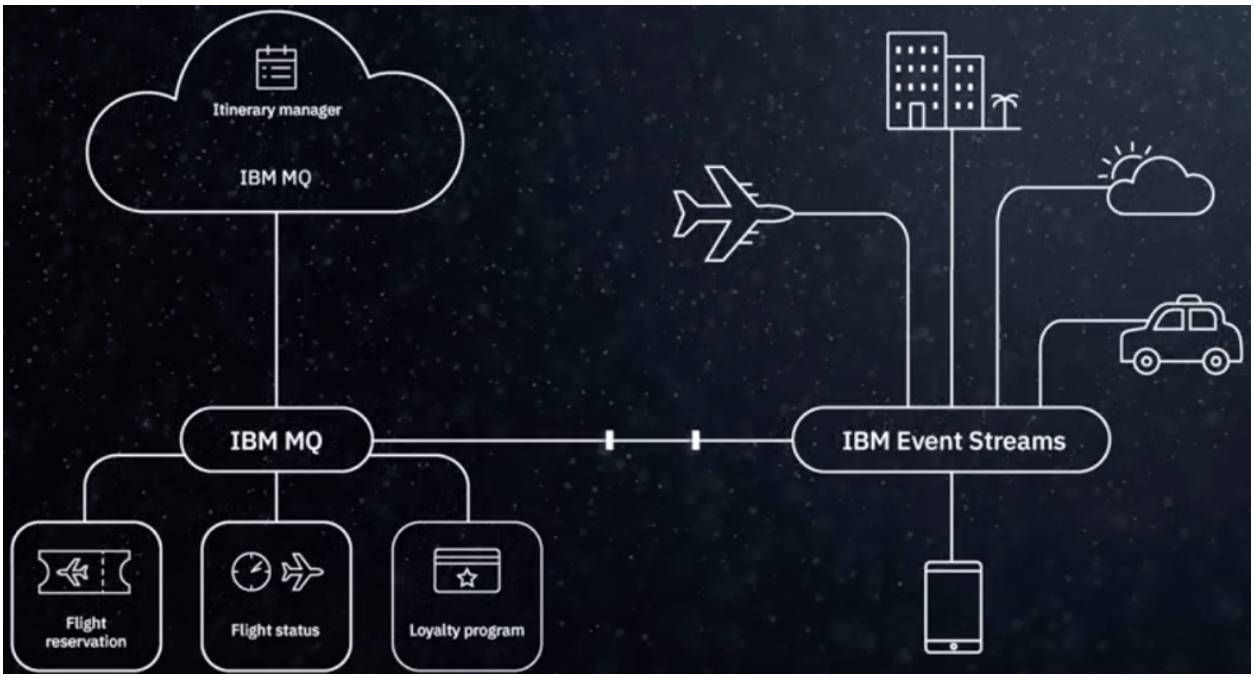
Connecting Message Queues and Kafka to stream data in real-time

In this lab exercise, you will learn how to connect IBM MQ and Kafka and exchange messages between them.

Improve your customer experience by reacting to situations in real time

The most interesting and impactful new applications in an enterprise are those that provide new ways of interacting with existing systems by reacting in real-time to mission-critical data. Leverage your existing investments, skills and even existing data, and use event-driven techniques to offer more-responsive and more-personalized experiences. Kafka event streams has supported connectivity to the systems you're already using. By combining the capabilities of Kafka event streams and message queues, you can combine your transaction data with real-time events to create applications and processes which allow you to react to situations quickly and provide a greater personalized experience.



In this tutorial, you will create a bi-directional connection between IBM MQ and IBM Event Streams (Kafka) by creating two message queues and two event stream topics. One is for sending and one for receiving. You will then configure the IBM Kafka Source and Sink Connectors for MQ in order to connect between the two providers.

In this tutorial, you will explore the following key capabilities:

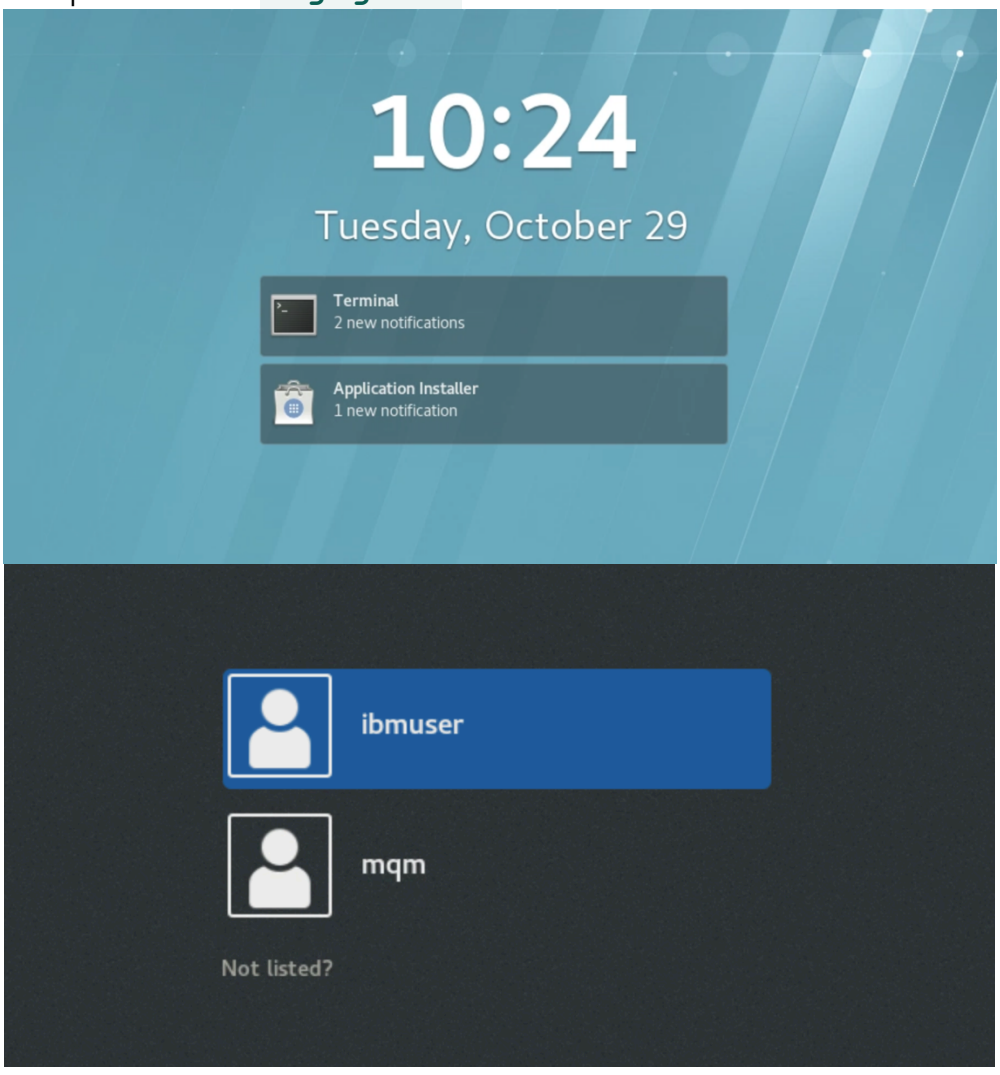
- Configure MQ to send and receive messages and events
- Configure Event Streams Topics: **mqtoevent** and **eventtomq**
- Configure MQ Source and Sync Connectors.
- Test connectors MQ source and sink (send a message and receive events).

Task 1 – Initial Setup Steps

IBM Event Streams is a component of the IBM Cloud Pak for Integration. As this is a new deployment of the IBM Cloud Pak for Integration, you need to execute some steps to prepare the environment. Initial setup steps are only needed for a fresh installation of the IBM Cloud Pak. They do not need to be repeated.

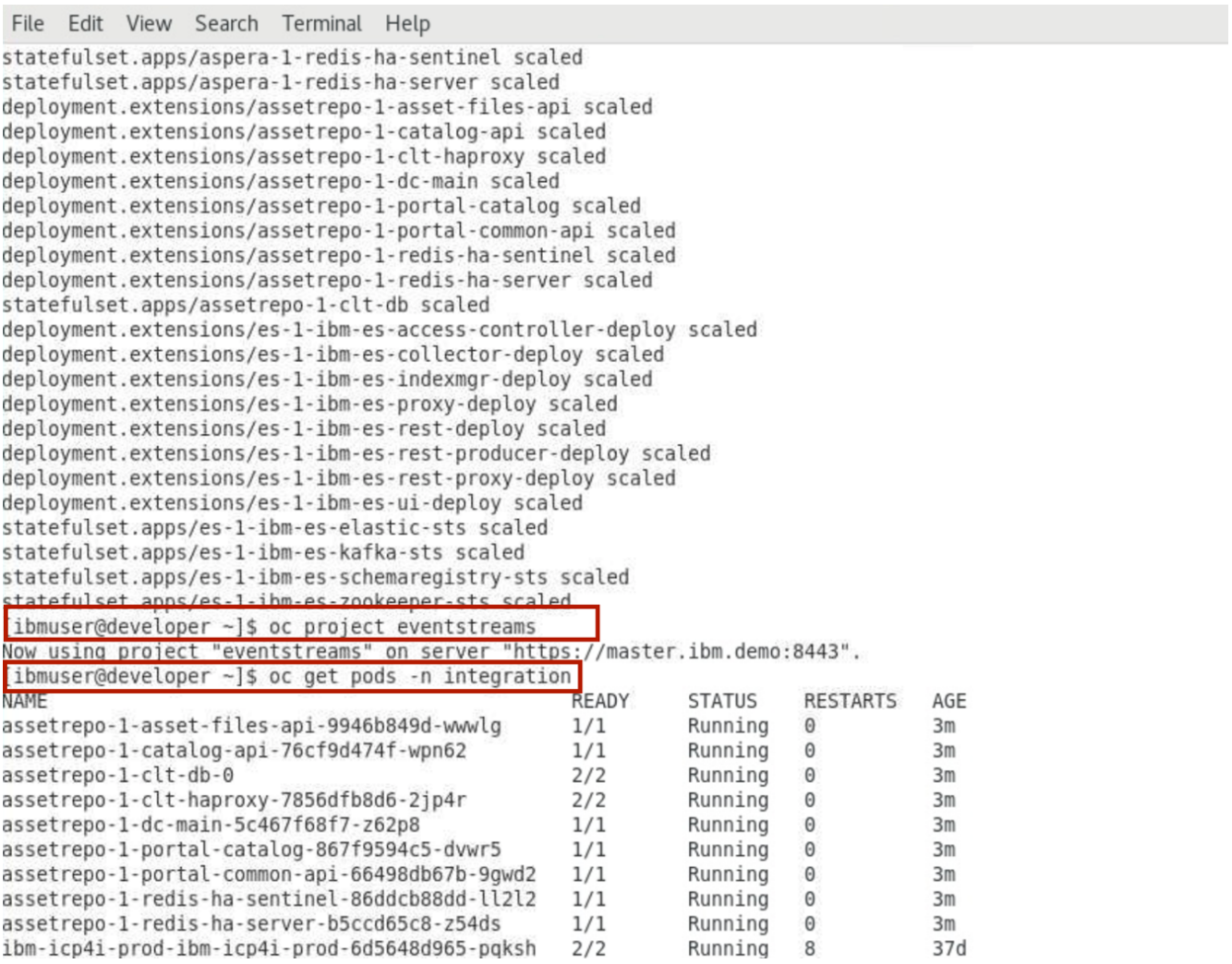
Start the Environment

1. Log into the Linux desktop with the following steps and credentials
    a. Click **`ENTER`** to view the Linux desktop.
    b. userid: **`ibmuser`**
    c. password: **`engageibm`**



Confirm the Cloud Pak for Integration is up and running

1. The next step is to check if the environment is done loading. Open a terminal console window by clicking the Terminal icon in the Desktop.
    a. Type `./startup-scripts/oc-startup.sh`
    b. Type `oc project eventstreams`
    c. Type `oc get pods -n integration`

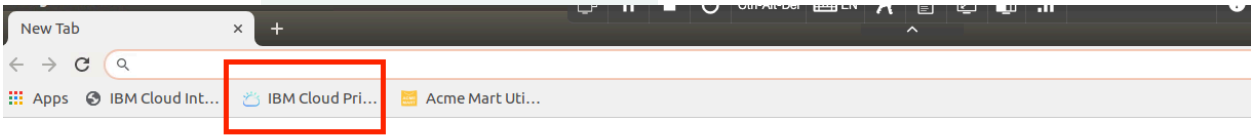


Note: The third command ensure the Cloud Pak is running. After the third command, once all the pods show 1/1 or equivalent proceed to the next step.

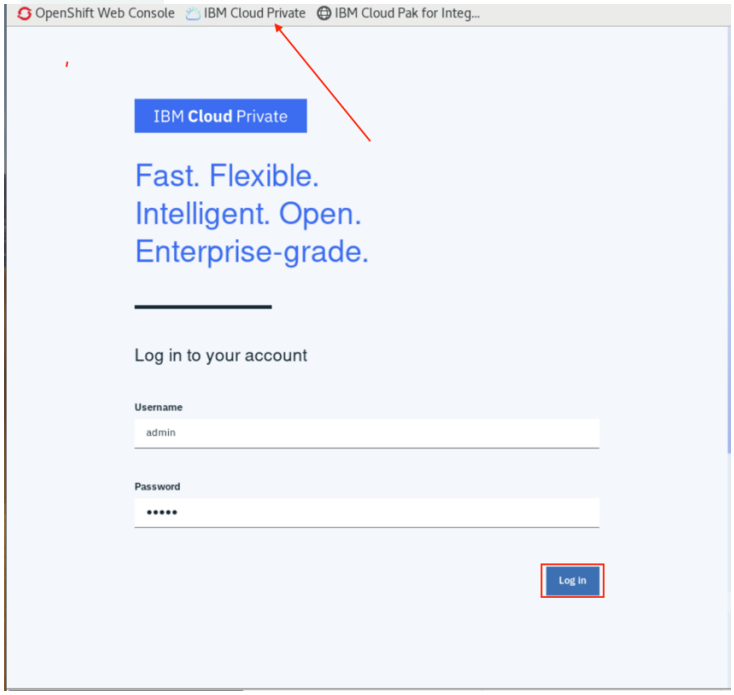
Sync Helm Repositories

1. The Helm repositories must be resynchronized between the repository and the server. Open a Chrome browser to do this:

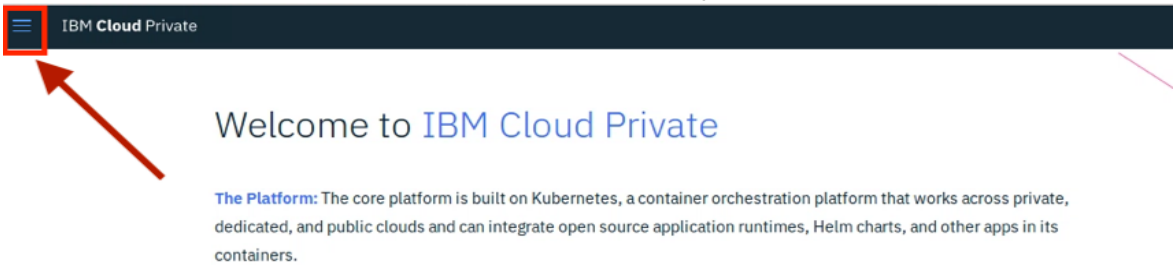
2. Click the **`IBM Cloud Private`** bookmark.



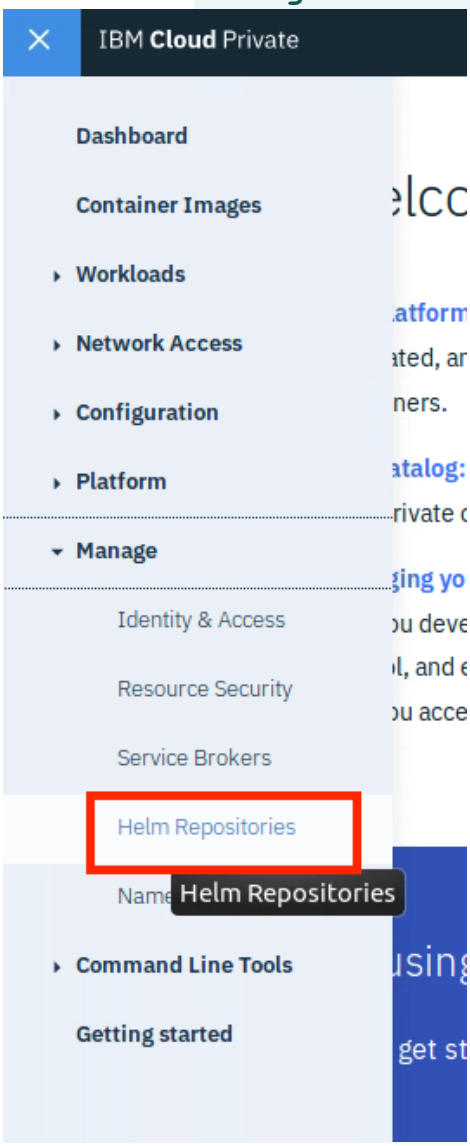
3. Log into IBM Cloud Private with the following credentials:
   a. username: **`admin`**
   b. password **`admin`**
   c. Click **`Log in`**



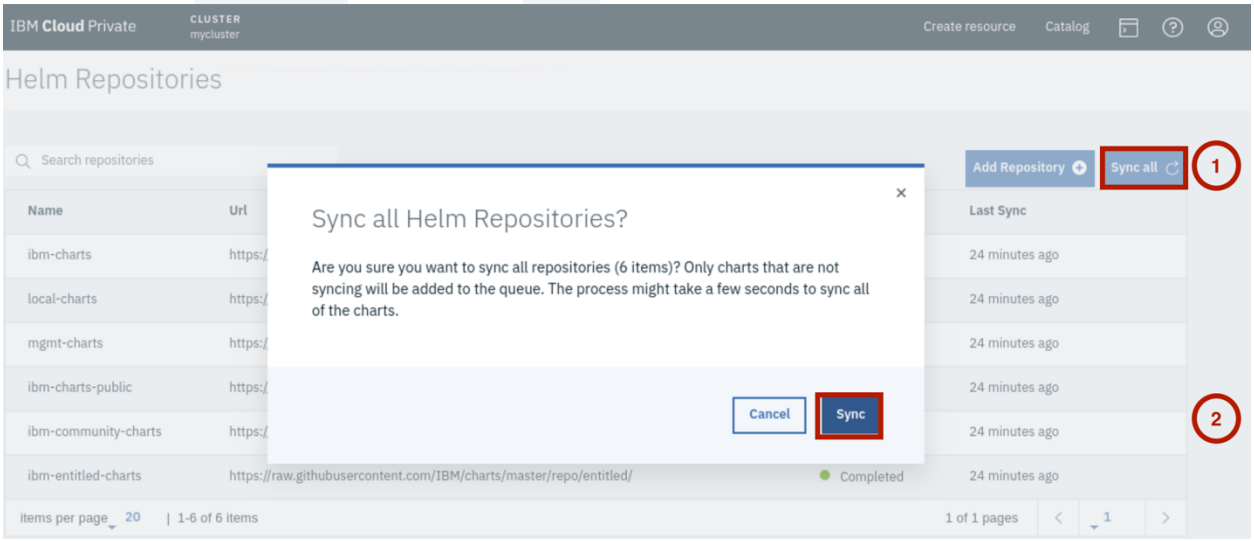
2. Click the main menu icon in the top left.



3. Click **`Manage -> Helm Repositories`**.



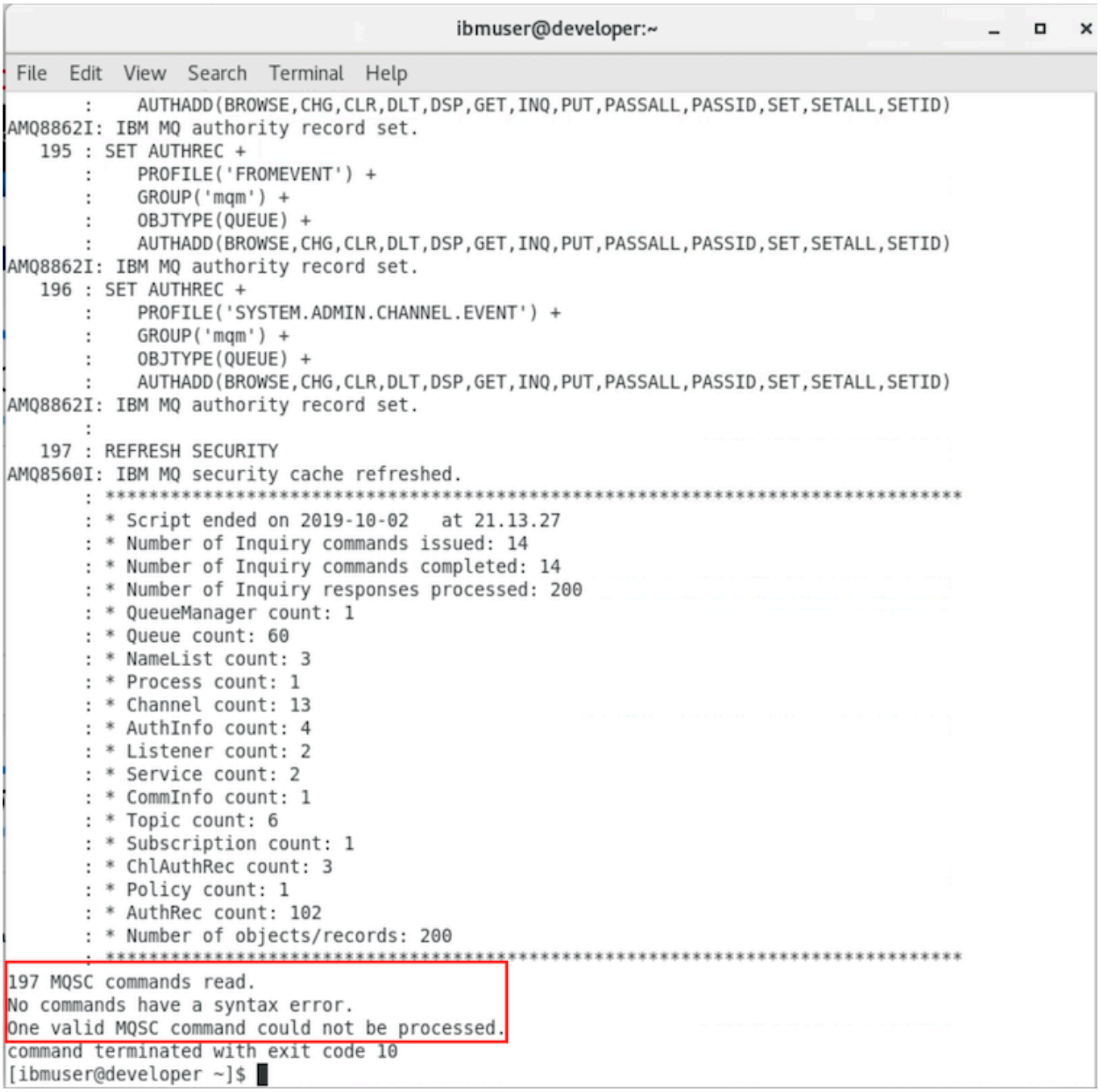
4. Click `Sync all` and click `Sync` in the new window to confirm.



Configure IBM MQ for security and to create MQ objects

1. You need to prep the IBM MQ queue manager to be authorized to accept connections and data coming from the Kafka MQ connectors.

   From a terminal window, type `./loadmqes.sh` and check the results.



Explore the platform capabilities.

IBM Cloud Pak for Integration provides a single solution for all of your enterprise integration needs. The platform provides a comprehensive set of industry-leading **capabilities.** Combine the powerful integration capabilities to create, manage, and monitor all of your integrations across applications, messaging, events, APIs, and more.

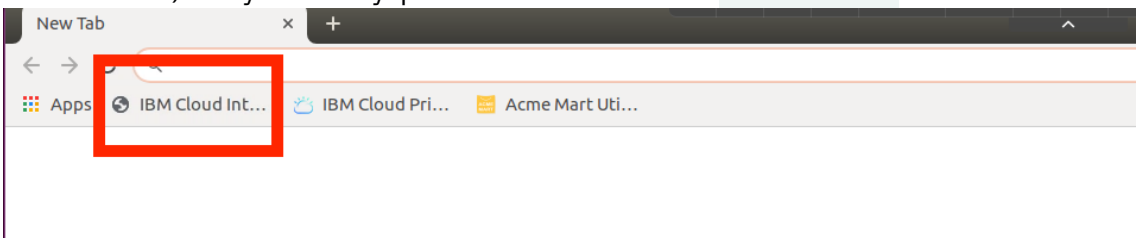
Unlock the power of your data and support the scale required for all of your integration and digital transformation initiatives.

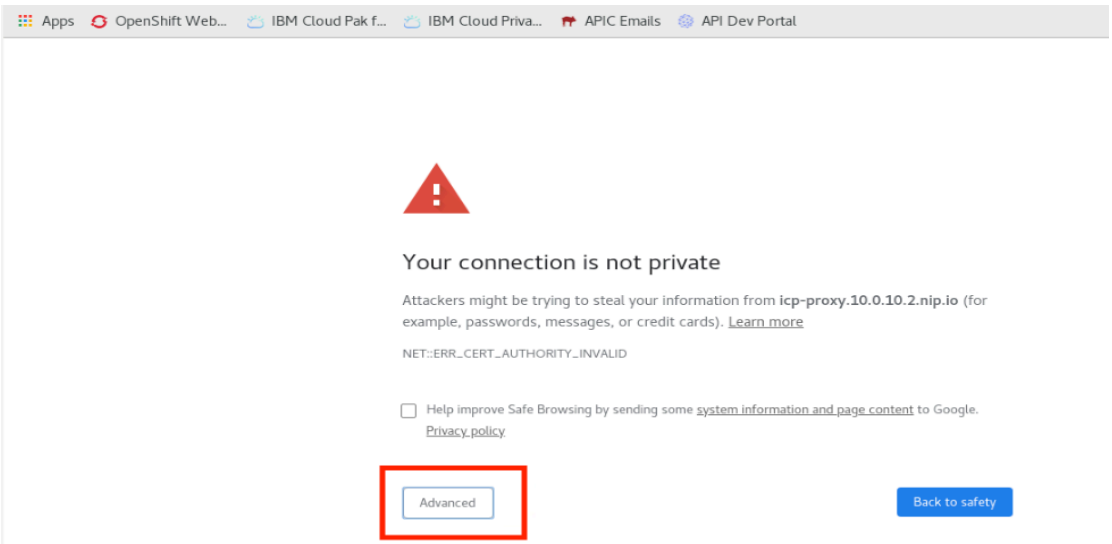
The home page of the **Cloud Pak for Integration** is referred to as the **Platform Navigator**. From the Platform Navigator you are able to navigate to all the integration and development technology contained within the platform.

As of today, the technology included is API management, application integration, message queues, and Kafka event streams. For this lab, we will work with IBM MQ and IBM Event Streams.

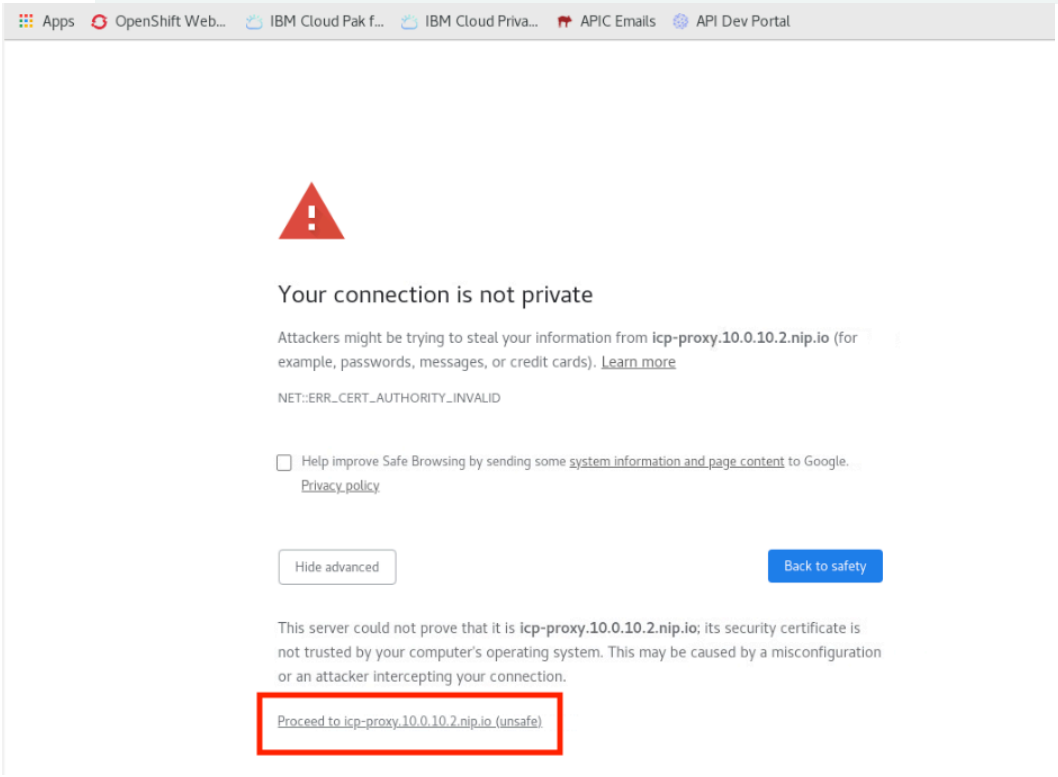
1. Click the **`IBM Cloud Pak for Integration`** bookmark in the bookmarks bar. If you see the login page, please continue to Task 2 (p. 11)

   If instead you see the **`Your connection is not private`** page, there is an issue reading a certificate. The certificate will not affect this lab, so you may proceed and click **`Advanced`**.





2. Click **`Proceed to icp-proxy.10.0.10.2.nip.io (unsafe)`**.

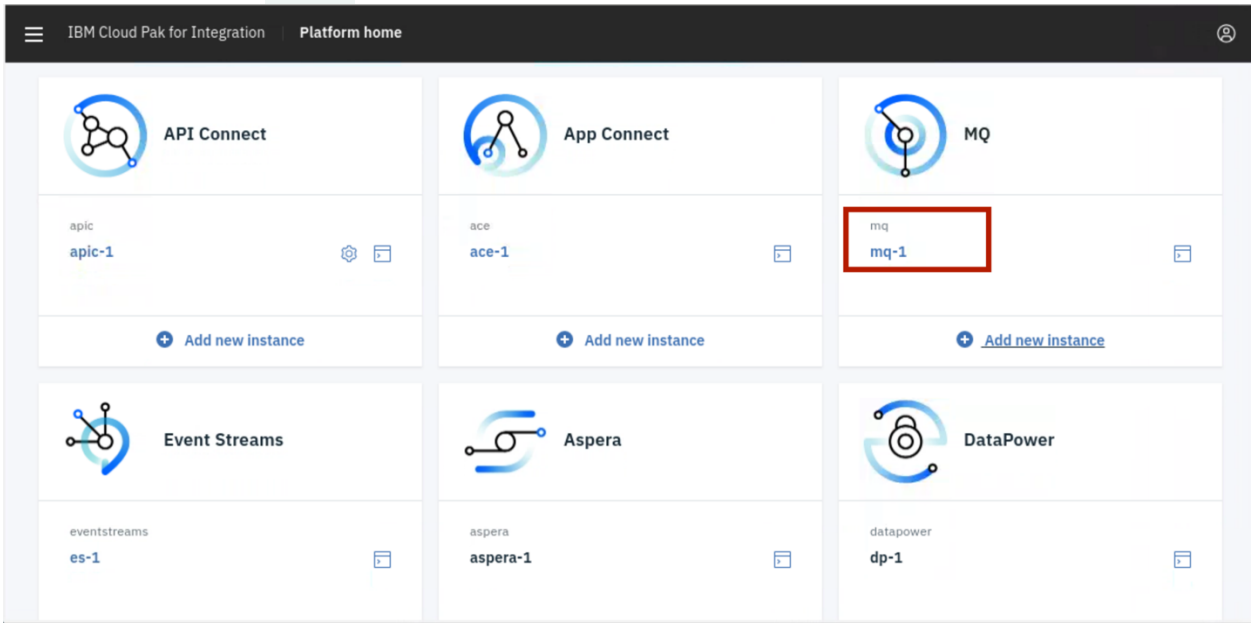


Task 2 – Configuring MQ

In this task you will work with the MQ Console, create a *Queues* widget for later use, and verify the two queues needed for this lab were created.

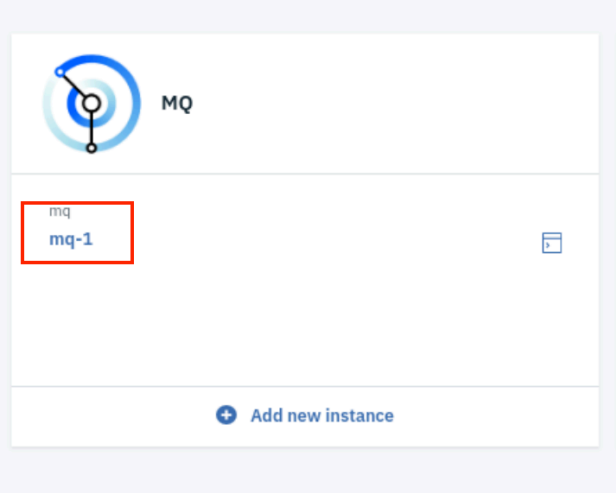
1. In your browser click the **`IBM Cloud Pak`** bookmark.
    a. To view the Cloud Pak for Integration main page, click on the **`Skip Welcome`** button.

2. On the Cloud Pak for Integration main page, locate the **`MQ`** service and click **`mq-1`**.

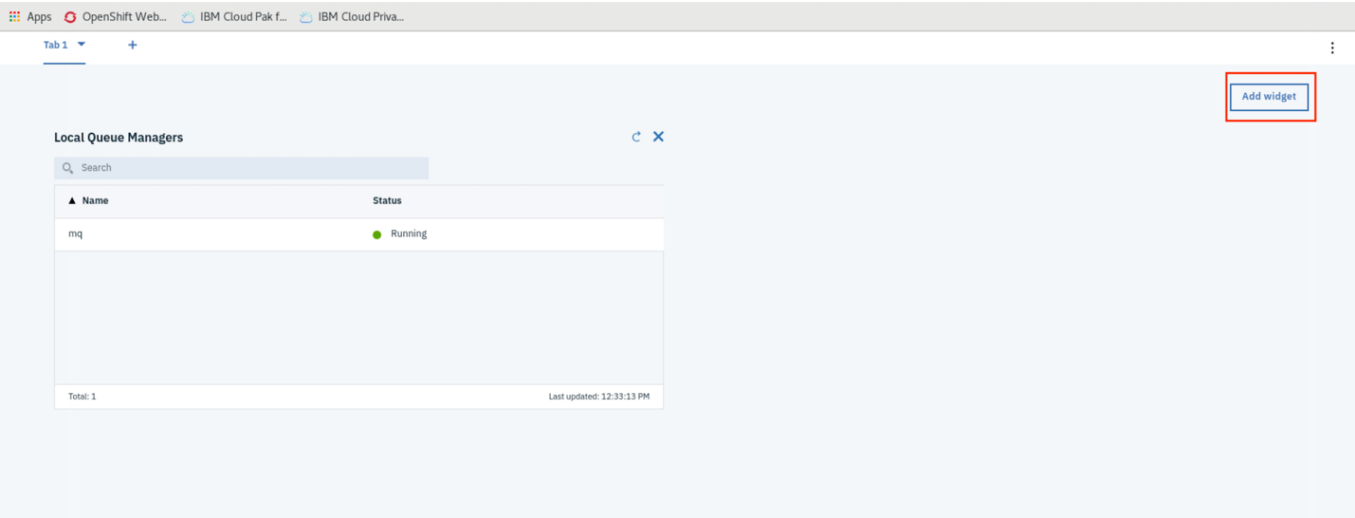


Note:
If you see the message "*mq-1 did not load correctly*", click **`Open mq-1`**. This issue occurs when using a self-signed certificate such as in this demo environment. This will load mq-1 in a new tab. Accept the self-sign certificate. Afterwards come back to the original tab and click **`Try again`**.

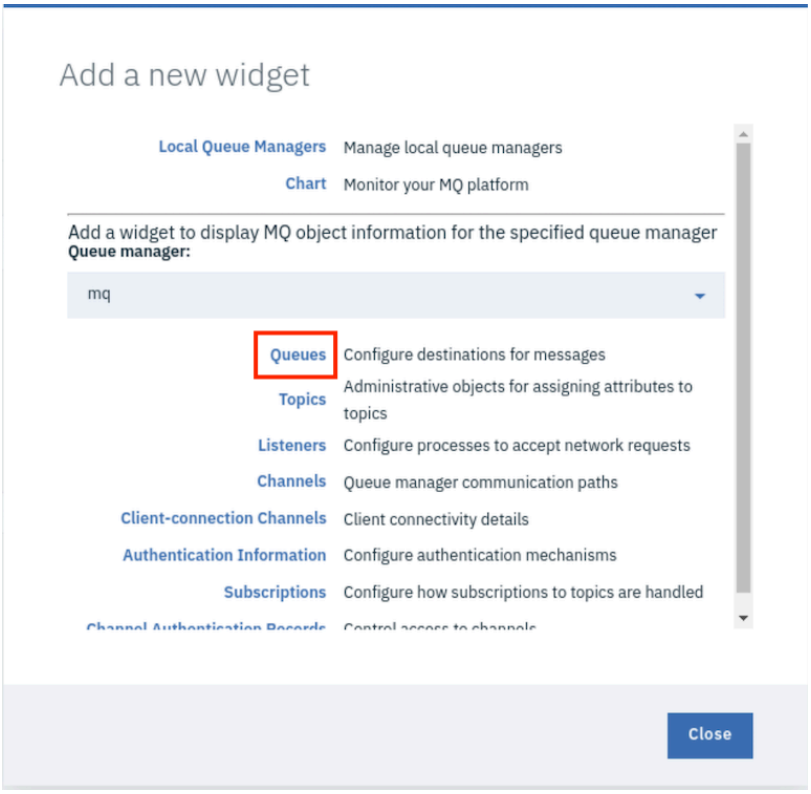
3. Click **`mq-1`** to open the MQ Console.



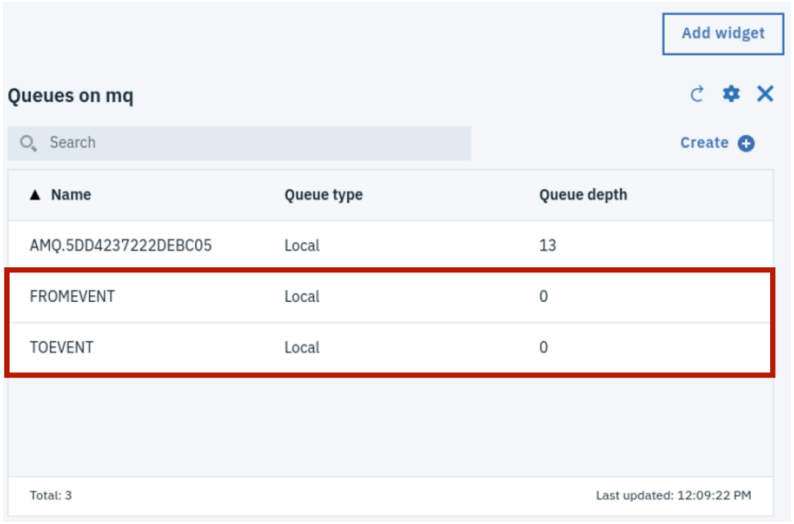
4. You will be signed on the IBM MQ console as an Administrator. You can create and configure MQ objects. Click the **`Add Widget`** icon.



5. Select **`Queues`** to create the widget and display a list of queues on the MQ console.

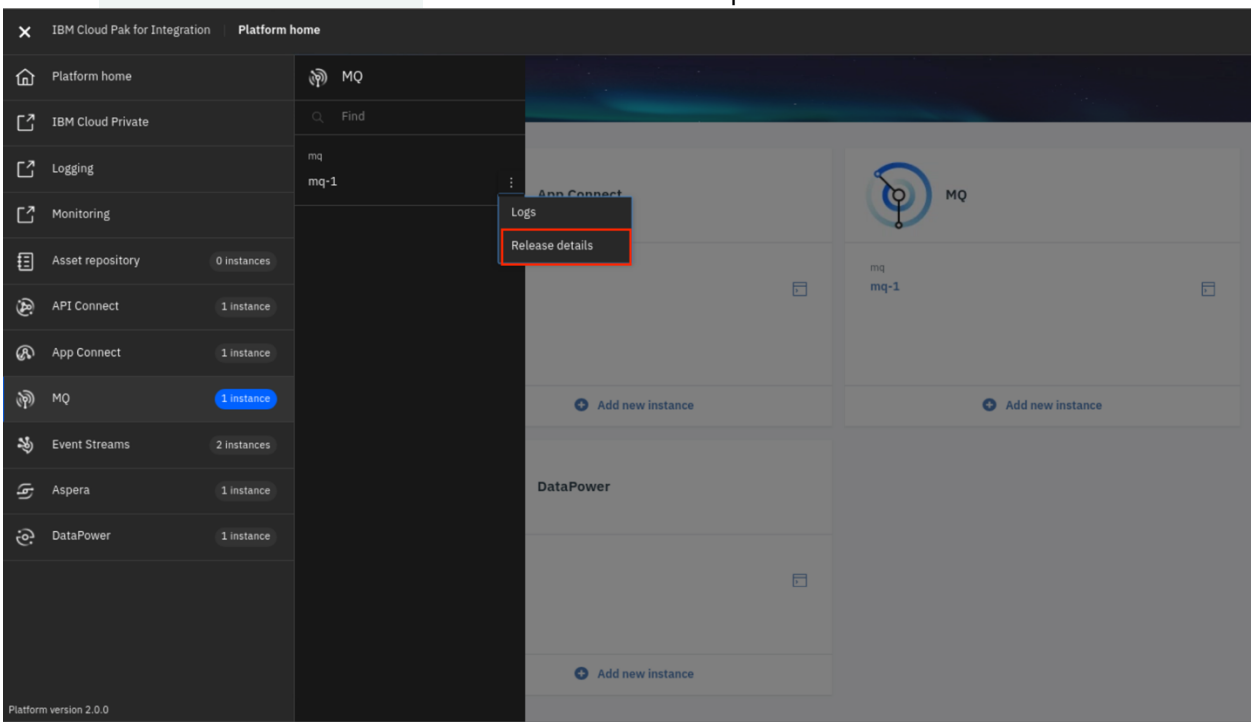


6. In the *Queues on mq* **widget** check the queues created for this lab: **FROMEVENT** and **TOEVENT**.

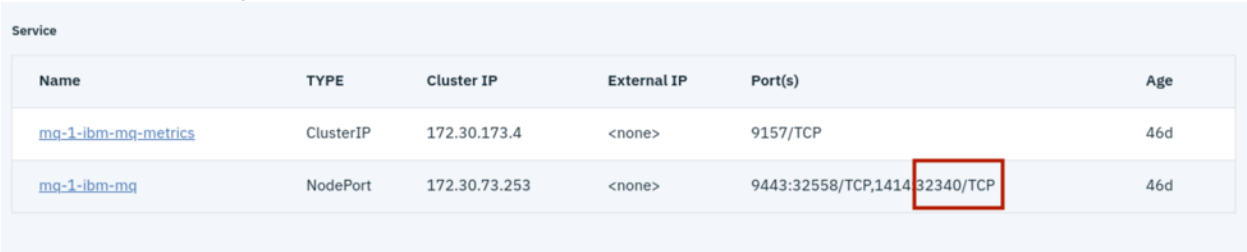


Note: The **`loadmqes`** script that was run earlier in this lab created the FROMEVENT and TOEVENT queues.

7. Click the Menu icon in the top left corner of the website. Followed by: **`MQ (1 instance) -> mq-1 -> 3 dots -> Release Details`** as shown on the picture.



8. Locate service and make a note of the connection information: listener port: **32340**



Service

Name	TYPE	Cluster IP	External IP	Port(s)	Age
mq-1-ibm-mq-metrics	ClusterIP	172.30.173.4	<none>	9157/TCP	46d
mq-1-ibm-mq	NodePort	172.30.73.253	<none>	9443:32558/TCP,1414:32340/TCP	46d

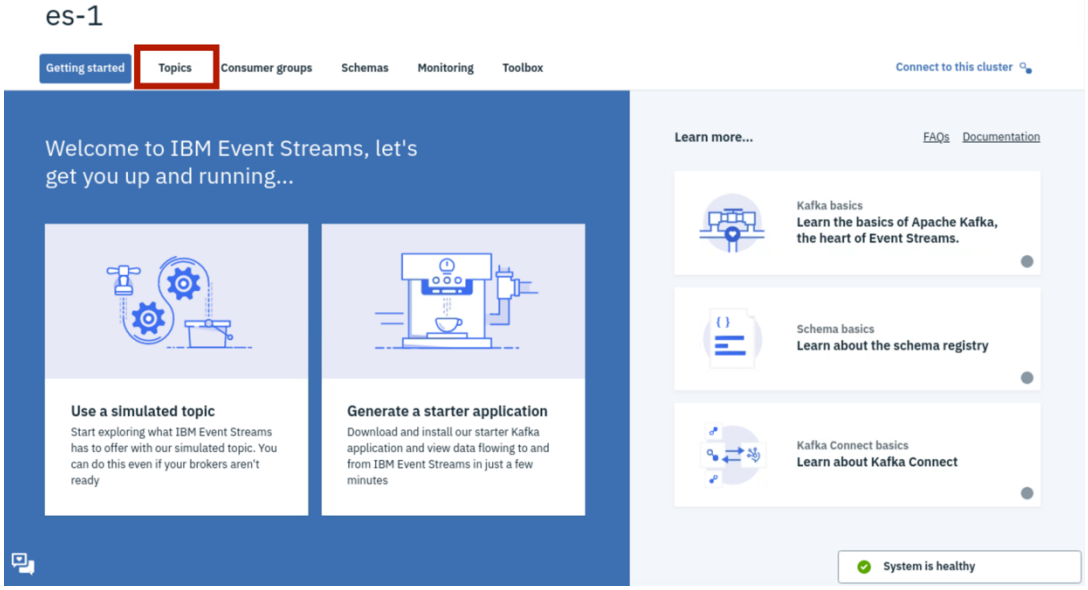
Task 3 – Configuring Event Streams

In this task, you will configure two topics on Event Streams. You will use the Event Streams instance created for this lab.

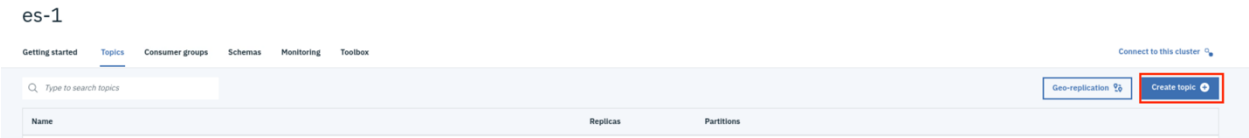
1. In your browser, click on the **IBM Cloud Pak** bookmark.
2. On the Cloud Pak for Integration main page, locate the **Event Streams** service and click **es-1**.
   Note:
   If you see the message "*es-1 did not load correctly*", click **Open es-1**. This issue occurs when using a self-signed certificate such as in this demo environment. This will load es-1 in a new tab. Accept the self-sign certificate. Afterwards come back to the original tab and click **Try again**.

3. Verify the event streams instance deployed successfully. In the bottom-right corner a **System is healthy** icon is shown.

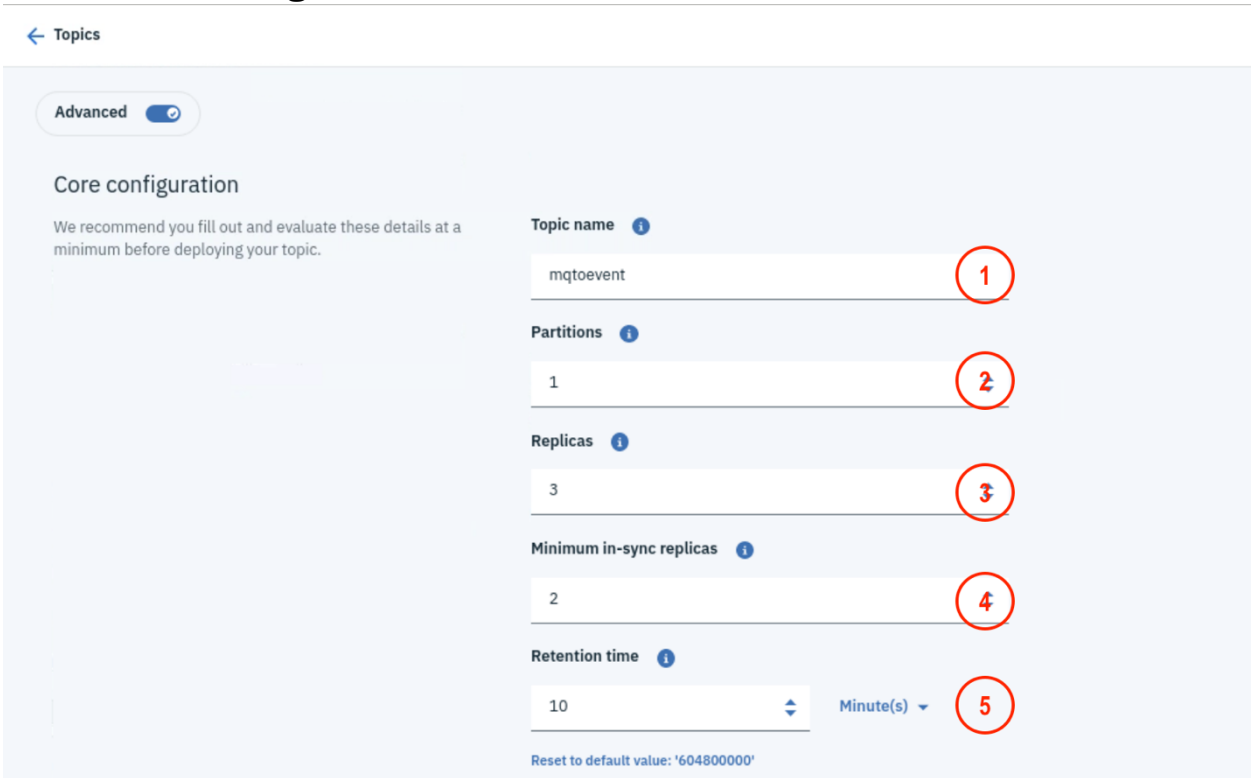
4. Next, you will create two Event Streams topics. Click on **Topics**.



5. Click on `Create Topic`



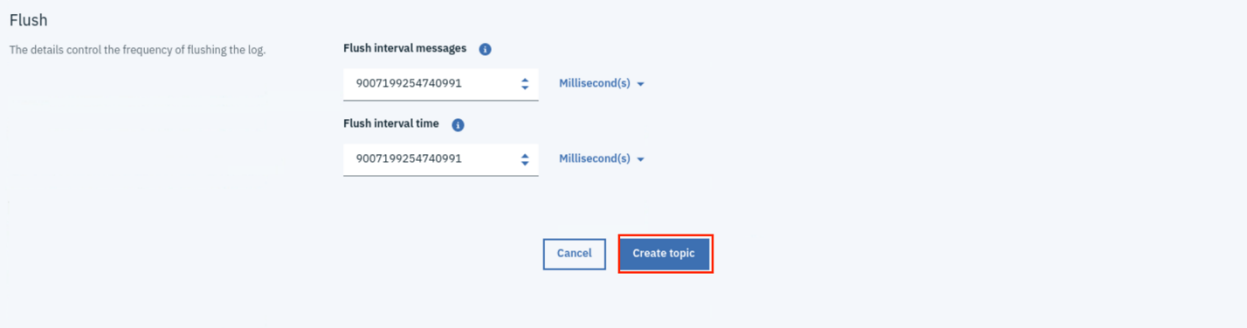
6. For our lab we will need two (2) topics. Click on the "Advanced" switch and type:
    1. **Topic Name: mqtoevent**
    2. Keep Partitions: **1**
    3. Keep Replicas: **3**
    4. Keep Minimum in-sync replicas: **2**
    5. Change **Retention time: 10 minutes**



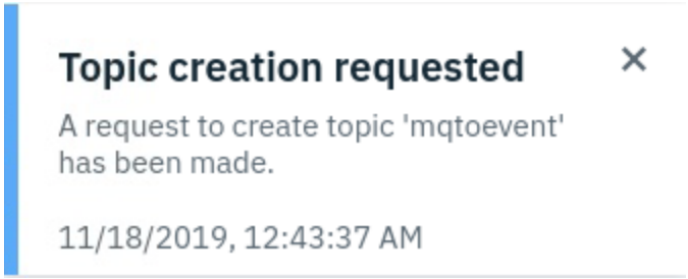
Note:

a. A *partition* is an ordered list of messages.
b. In order to improve availability, *replicas* of each topic can be maintained on multiple brokers. Hence the number of replicas.
c. *In-sync replicas* determine the reliability achievable for this topic.
d. *Retention time* is how long messages are retained before they are deleted automatically by Event Streams.

7. Scroll down and click `Create Topic`.

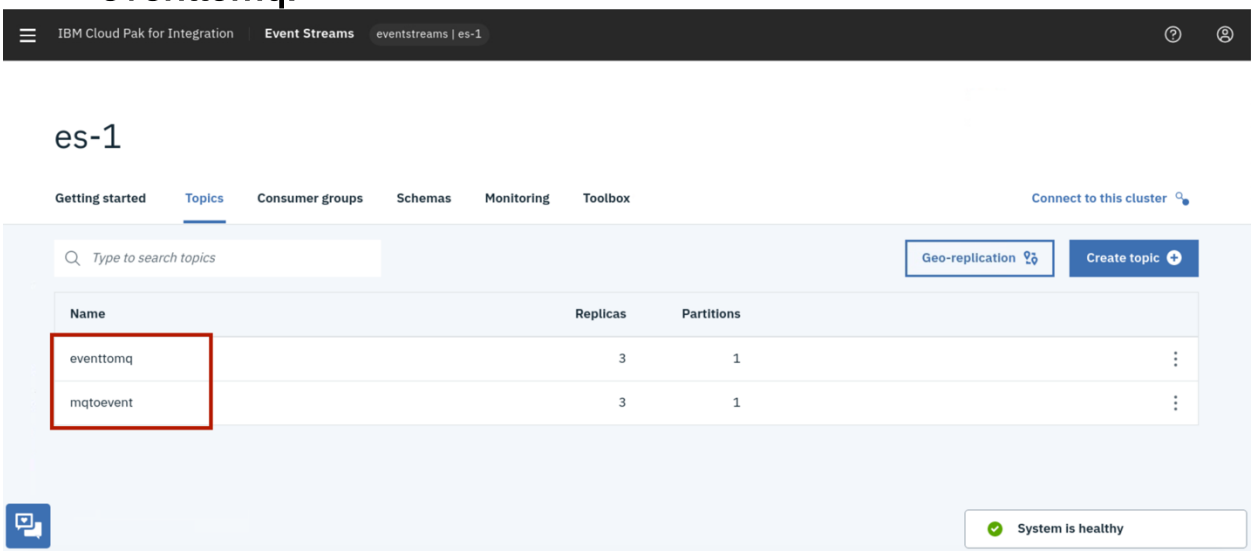


8. A **"Topic creation requested"** icon will appear to validate the creation of the topic mqtoevent.



9. One more topic is needed for this lab. Click `Create Topic`. Topic configuration:
    a. **Topic Name eventtomq**
    b. Keep Partitions: **1**
    c. Keep Replicas: **3**
    d. Keep Minimum in-sync replicas**: 2**
    e. Change **Retention time: 10 minutes**
    f. Click **Create topic**

9. You have created and configured two topics: **mqtoevent** and **eventtomq.**



Task 4 – Installing Kafka MQ connectors

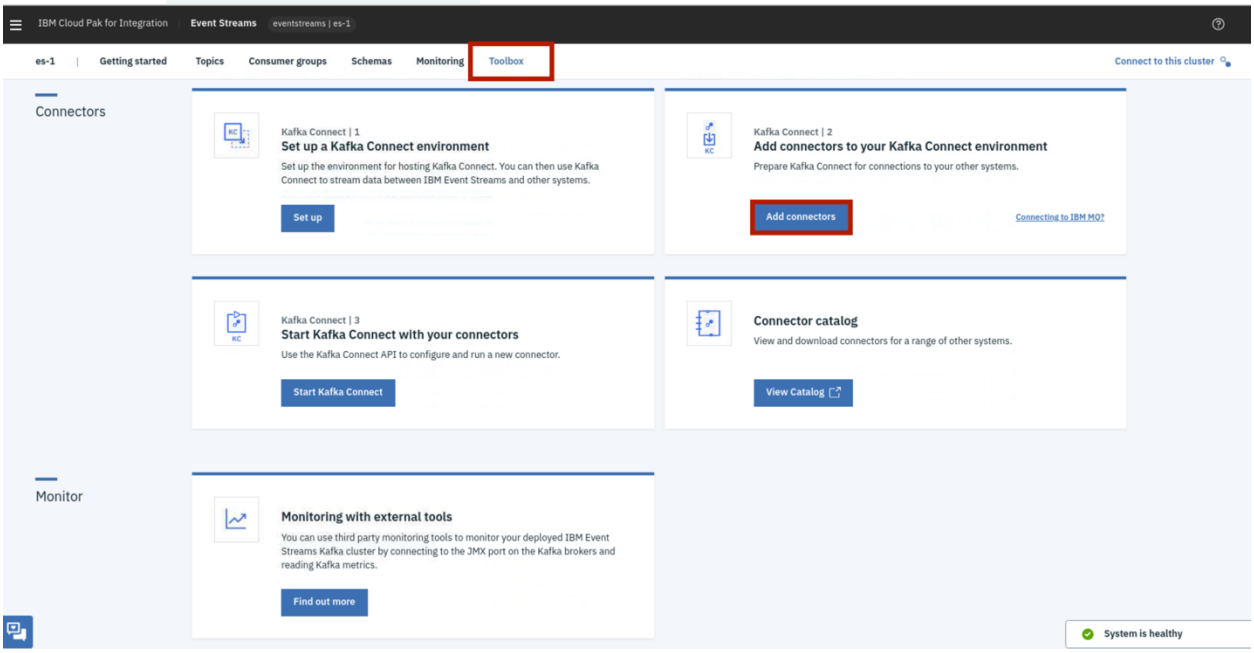
Kafka Connect connectors run inside a Java process called a ***worker***. These instructions focus on the standalone mode because it's easier to see what's going on. You will use a standalone Kafka connector that has already been installed for this lab.

Both the sink and source connector configuration files contain the properties needed for each connector. For this lab, you need to configure the MQ Kafka Connectors.

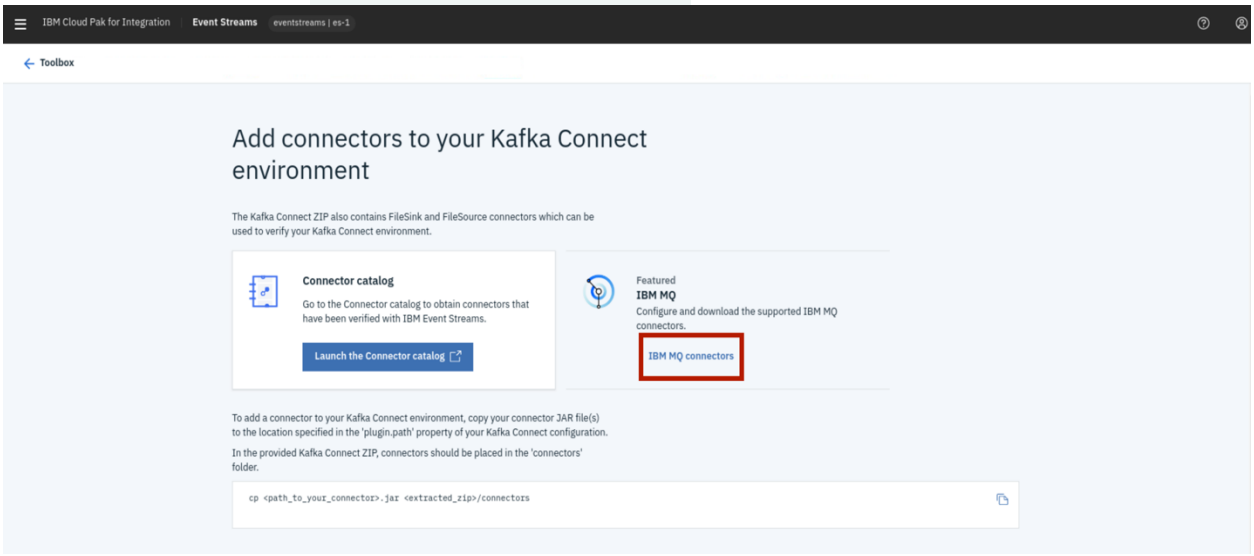
For reference both connectors can be found in Github.

- Source connector https://github.com/ibm-messaging/kafka-connect-mq-source/tree/master/config
- Sink connector https://github.com/ibm-messaging/kafka-connect-mq-sink/tree/master/conf...

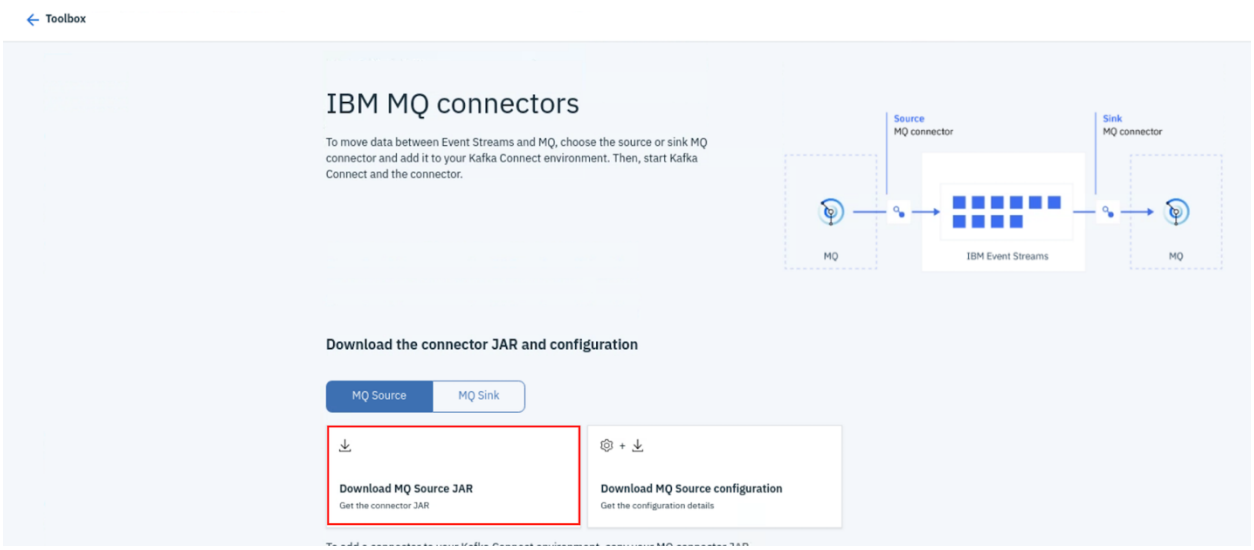
1. Select **`Toolbox`** and scroll down to the **Connectors** tile. Click on the **`Add connectors`** tile.



2. On the page Add Connectors to your Kafka Connect environment, click on the **`IBM MQ connectors`** link.

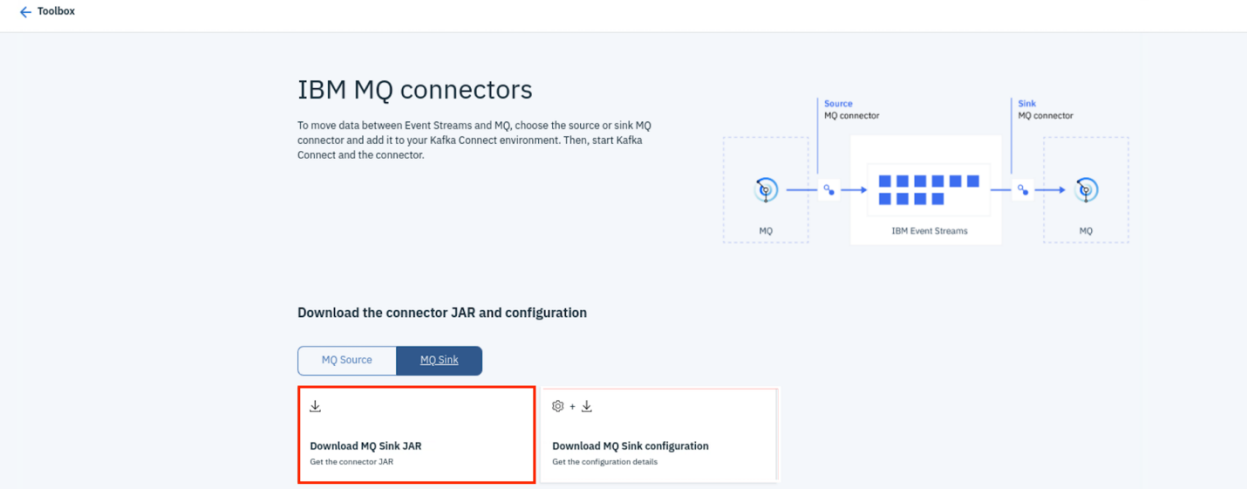


3. On the IBM MQ connectors, you can download the MQ source and MQ sink connectors. Click **`Download MQ Source JAR`**. The file ***kafka-connect-mq-source-1.1.0-jar-with-dependencies.jar*** will be located in ***/home/ibmuser/Downloads***.



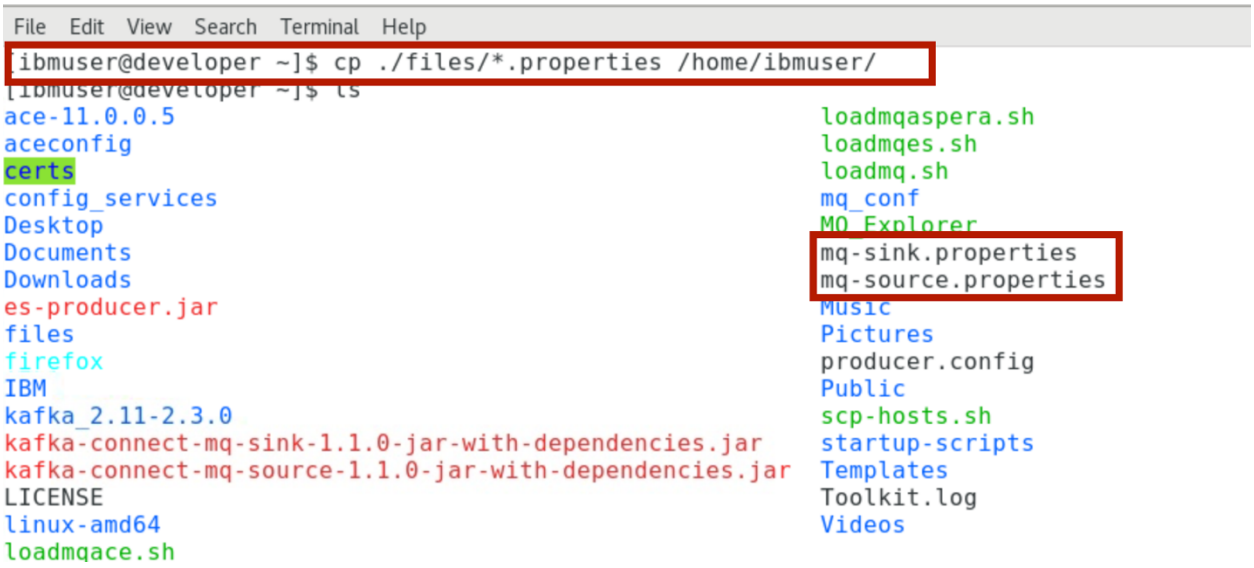
Note: The MQ Source connector moves messages from MQ to Event Streams while MQ Sink connector moves events from Event Streams to MQ.

4. Click on the **`MQ Sink`** icon. Click **`Download MQ Sink JAR`**. The file *kafka-connect-mq-sink-1.1.0-jar-with-dependencies.jar* will be located on */home/ibmuser/Downloads*.



5. Open a terminal and copy the recently downloaded JAR files to the */home/ibmuser* directory:
    a. **`cp ./Downloads/*.jar /home/ibmuser/`**

6. You will need to configure the sink and source connectors.
    a. `cp ./files/*.properties /home/ibmuser/`



For reference the configuration files for the connectors can be found in Github. Download the MQ sink and MQ source configuration connectors from Github.

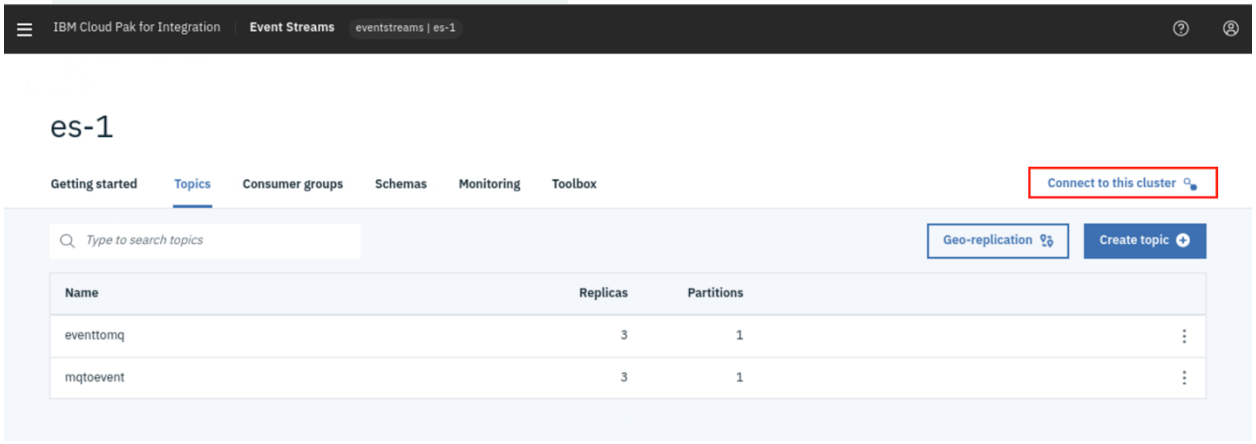
- Sink connector https://github.com/ibm-messaging/kafka-connect-mq-sink/tree/master/config
- Source connector https://github.com/ibm-messaging/kafka-connect-mq-source/tree/master/config

Task 5 – Create API Key and Event Streams certificate (Event Streams Operations)

You downloaded the Kafka sink and source connectors JAR files and MQ sink and source configuration.

You will need some security parameters to execute Kafka Connectors.

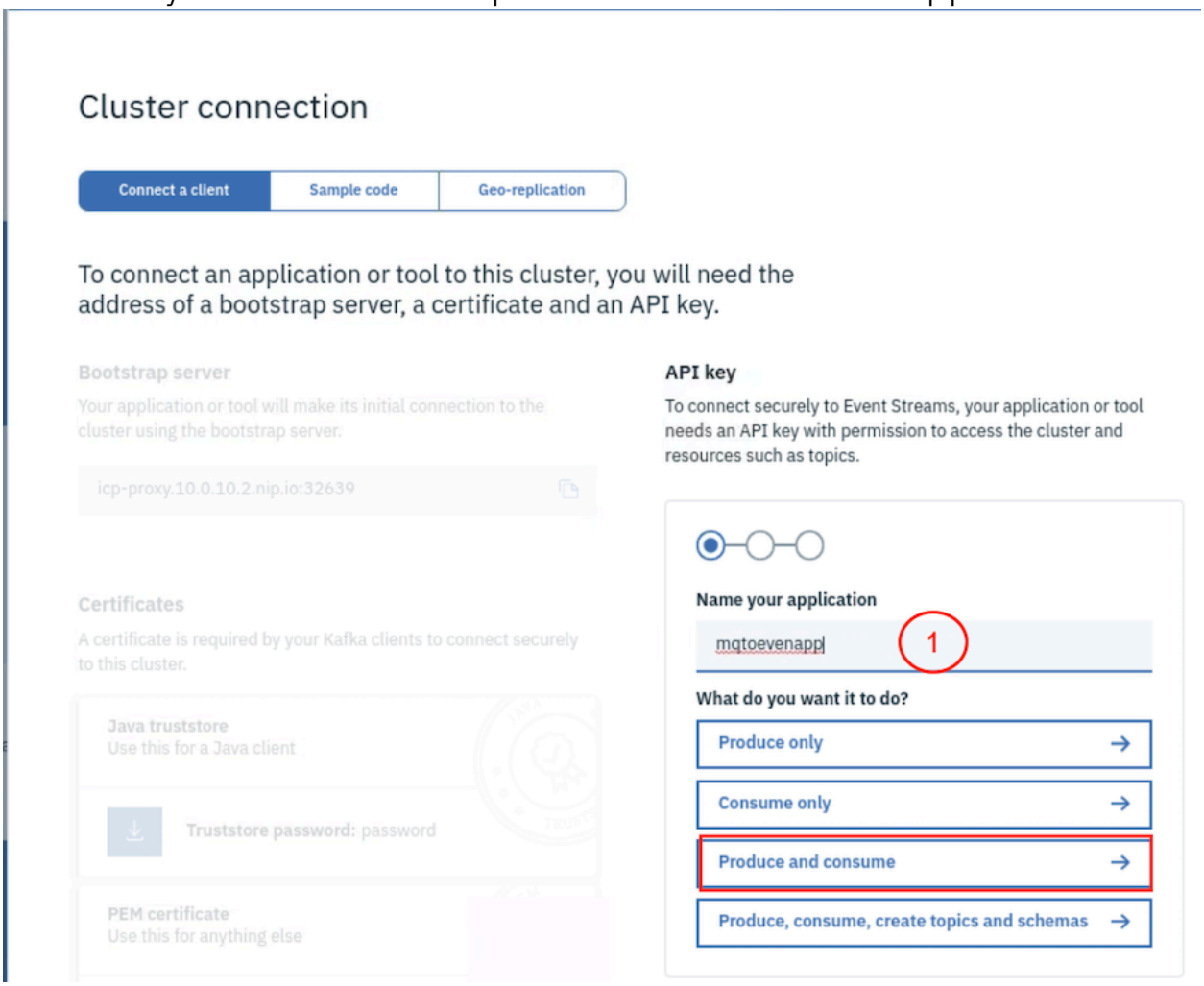
1. In the browser, click on **`IBM Cloud Pak`** bookmark. Open the **`es-1`** instance. Click the **`Topics`** link. Click on the **`Connect to this cluster`** link.



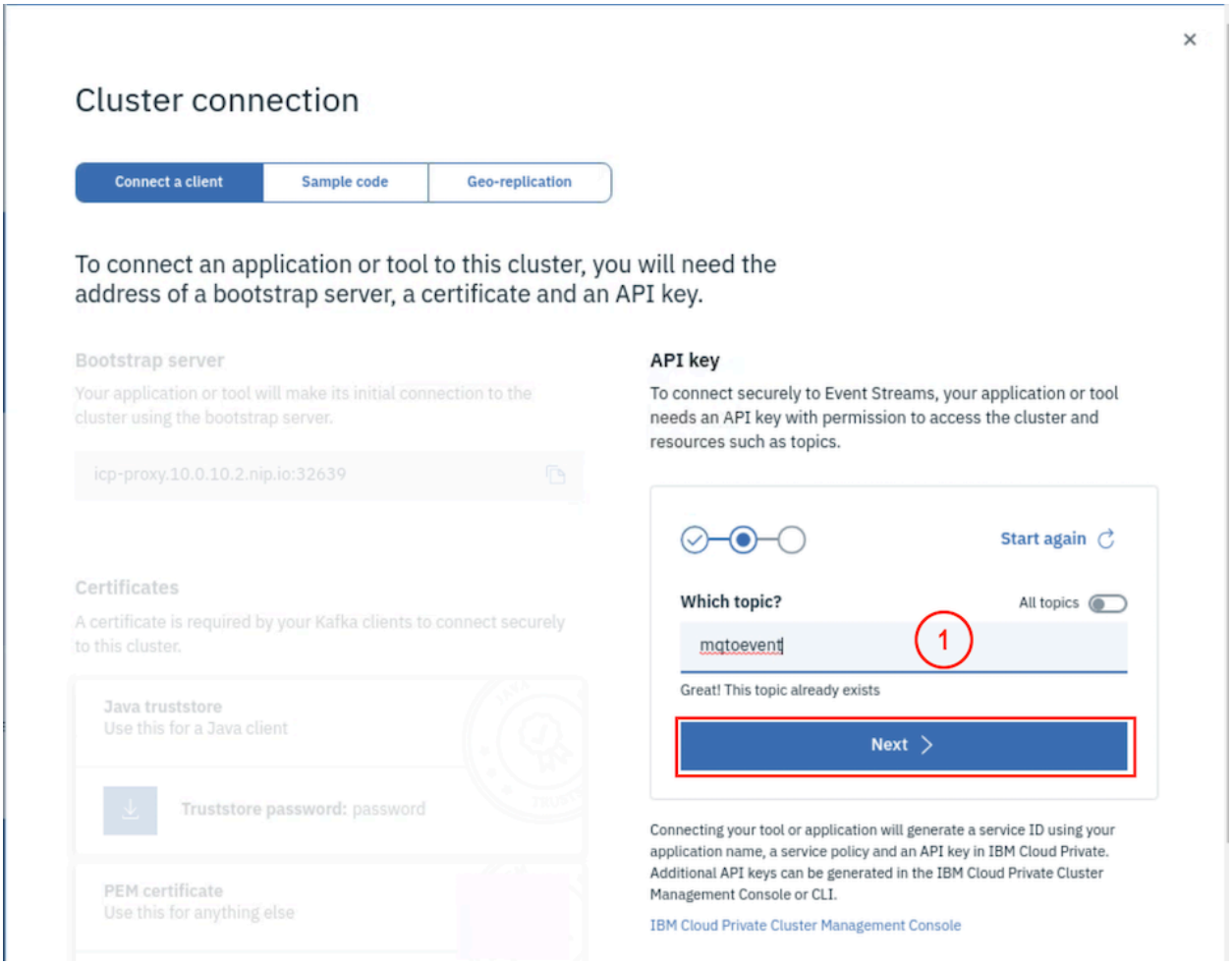
2. In order to connect to MQ, the following are needed:
   - a bootstrap server
   - a certificate
   - an API Key

3. Let’s create an API key.

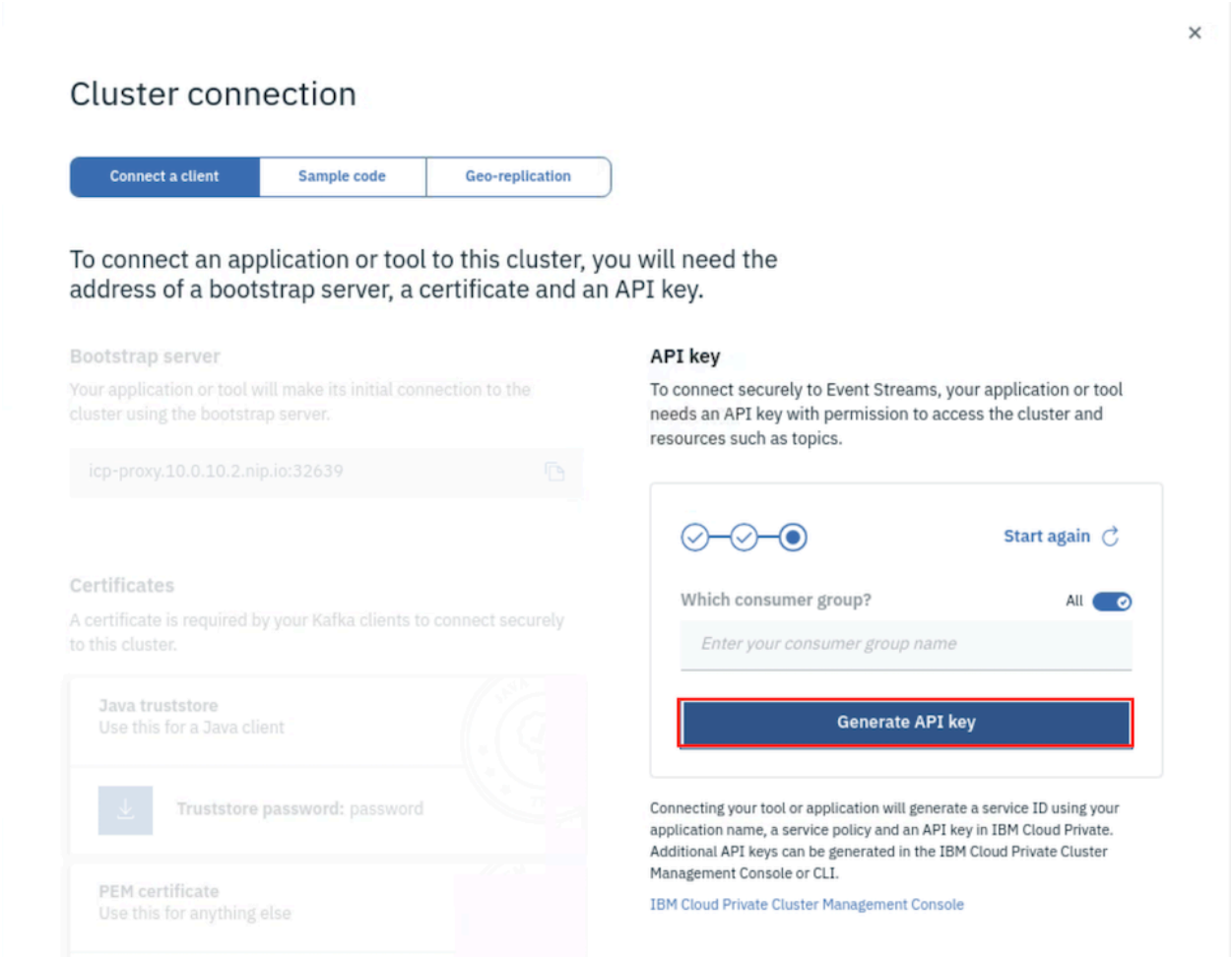
Insert the name of your application. You can type **mqtoeventapp**. Click on `Produce and consume`. This means this API Key can be used for a producer and consumer application.



4. Type the name of topic that you will access under **Which topic?** In this case the topic is **mqtoevent.** Click `Next` to confirm the topic.

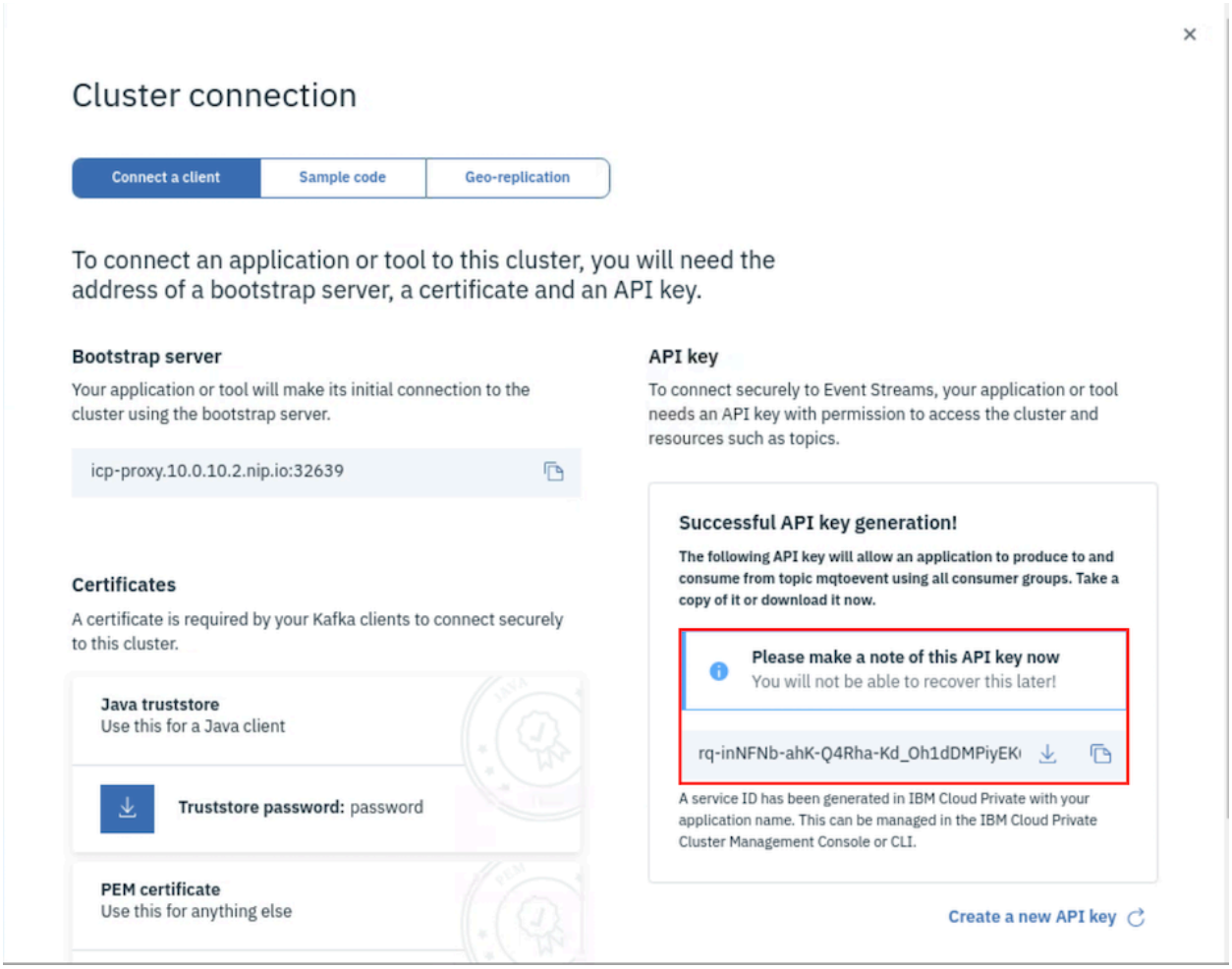


5. Click the **`Generate API Key`** icon. This API Key will have the permission to access the topic and all external applications will use this Key as their userid when accessing this topic.



6. Event Streams generates an API Key to connect our application (MQ).

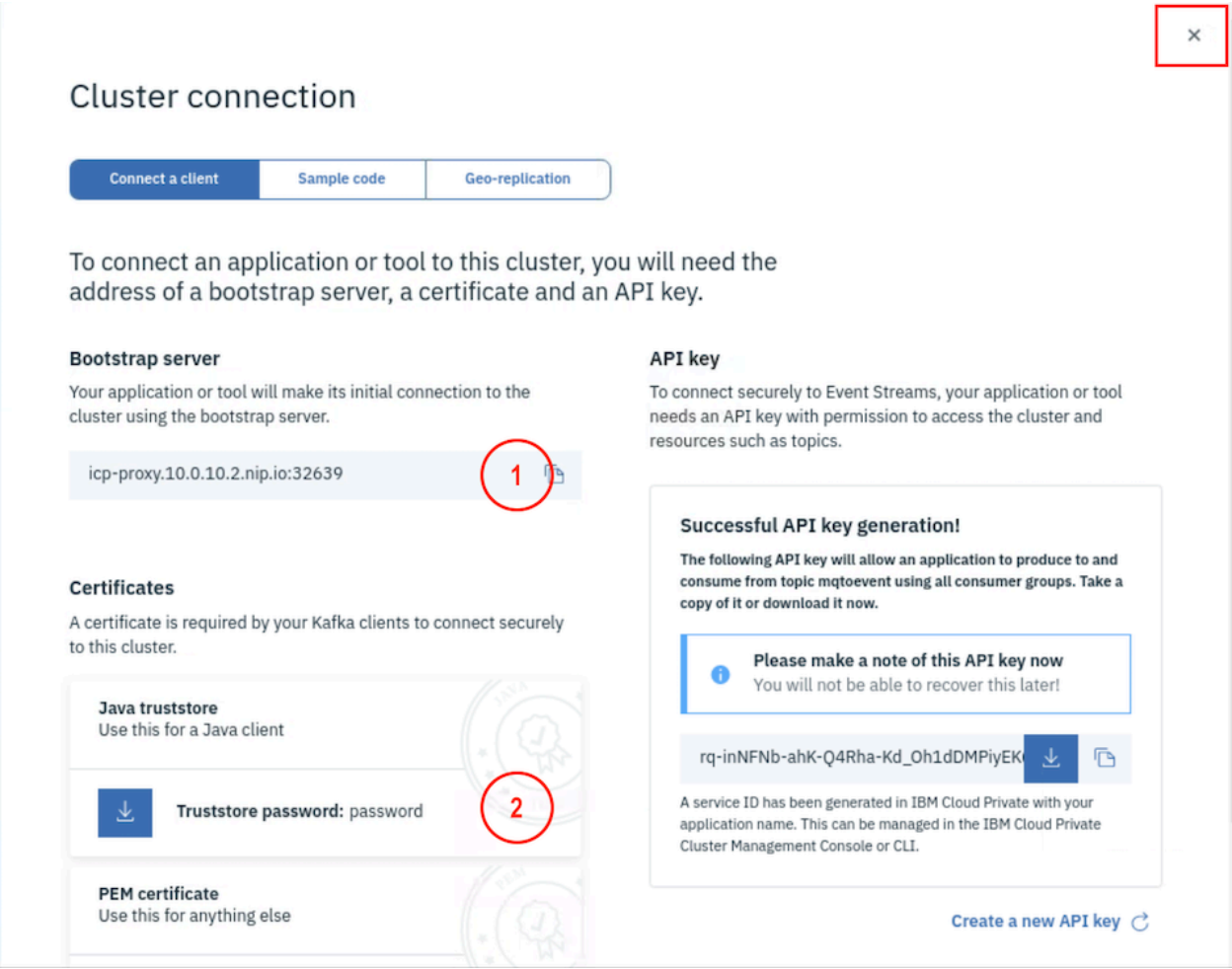
   Click the **download** icon.



7. In a terminal window, go to */home/ibmuser/Downloads* to change the name of the API Key file from es-api-key.json to mqtoevent.json

   Command: `mv es-api-key.json mqtoevent.json`

8. In the Event Streams console, go to **Certificates** and Java truststore. Click the **Truststore password icon** to download. Copy the **bootstrap server**



For reference

a. The es-api-key will be on /home/ibmuser/Downloads directory
b. The es-cert.jks will be on /home/ibmuser/Downloads directory

9. Go back to `Topics` and click the `Connect to cluster` icon. Repeat steps 3,4,5, and 6. Create:
   a. Application: **eventtomqapp**
   b. Select **Produce and consume**
   c. Which topic?: **eventtomq**
   d. Generate the API key. Go to */home/ibmuser/Downloads* and save the es-api-key.json as eventtomq.json
      Command: `mv es-api-key.json eventtomq.json`
   e. Do not download the Certificate **Truststore password** again (you will reuse the one you downloaded for the previous topic)

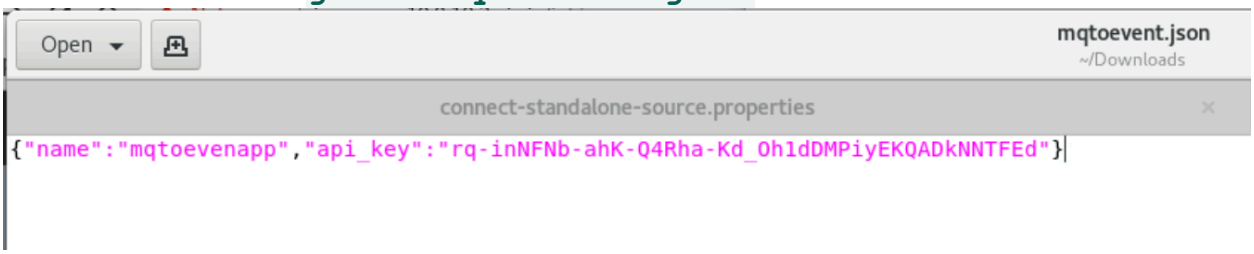
Task 6 – Configuring Kafka Connectors properties

There are two configuration files for each connector. The Kafka connector configuration file contains the properties needed to connect to Kafka. The MQ connector configuration file contains the properties needed for the connector to connect to MQ.

Configuring the Kafka source connector

1. The API key is necessary to connect MQ to Event Streams. Open the terminal window and go to */home/ibmuser/Downloads* and copy the API key. *Do not copy the quotation marks.
   NOTE! Your API key will be different than the one shown below.

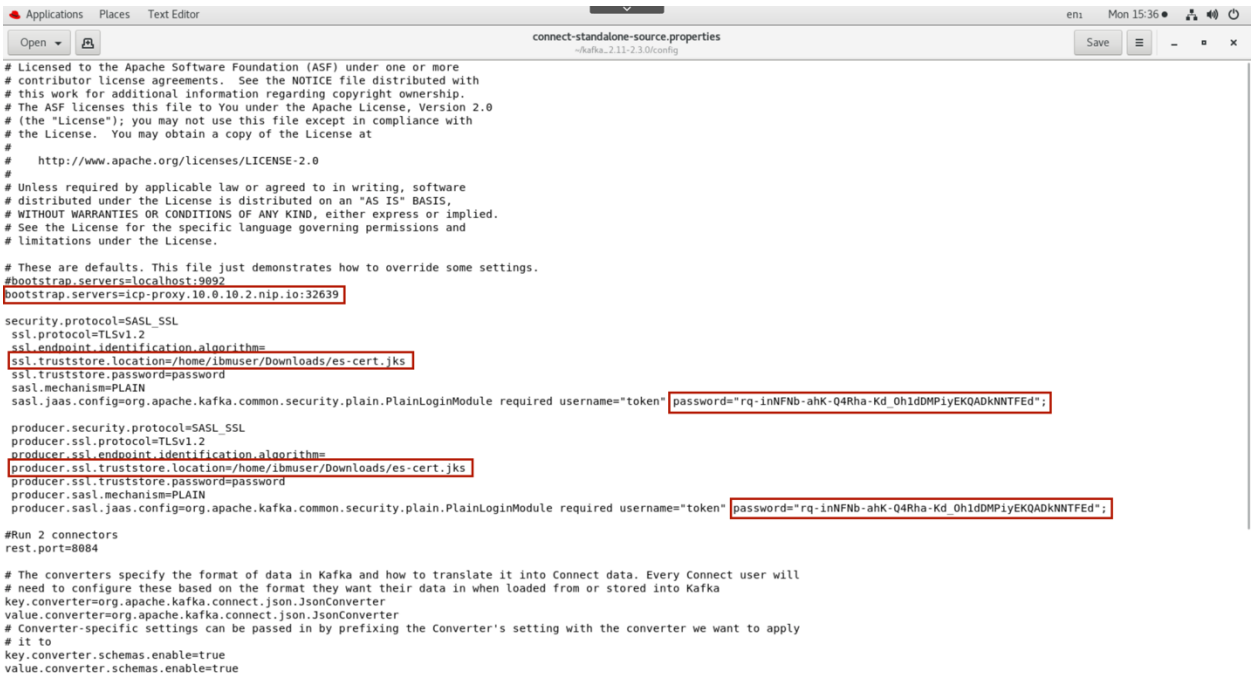
   Command: `gedit mqtoevent.json`



2. From the terminal window, go to the *home/ibmuser/kafka_2.11-2.3.0/config/* directory. A file has been prepared for our lab. There are two sections that you need to configure within the file: **ssl** and **producer.**

   Command: `gedit connect-standalone-source.properties`

a. The first section you configure is security:
   - Set the bootstrap.servers=**icp-proxy.10.0.10.2.nip.io:32639**
   - Set the ssl.truststore.location=/**home/ibmuser/Downloads/es-cert.jks.**
   - The sasl.jaas.config variable, **update the API Key password** to the API key copied from the mqtoevent.json file (Step 1).



b. The second section you configure is the producer**:**
   - Set the producer.ssl.truststore.location=/**home/ibmuser/Downloads/es-cert.jks.**
   - The producer.sasl.jaas.config variable, **update the API Key password** to the API Key copied from the mqtoevent.json file (Step 1). Click **Save** and close the file.

Configuring the Kafka sink connector:

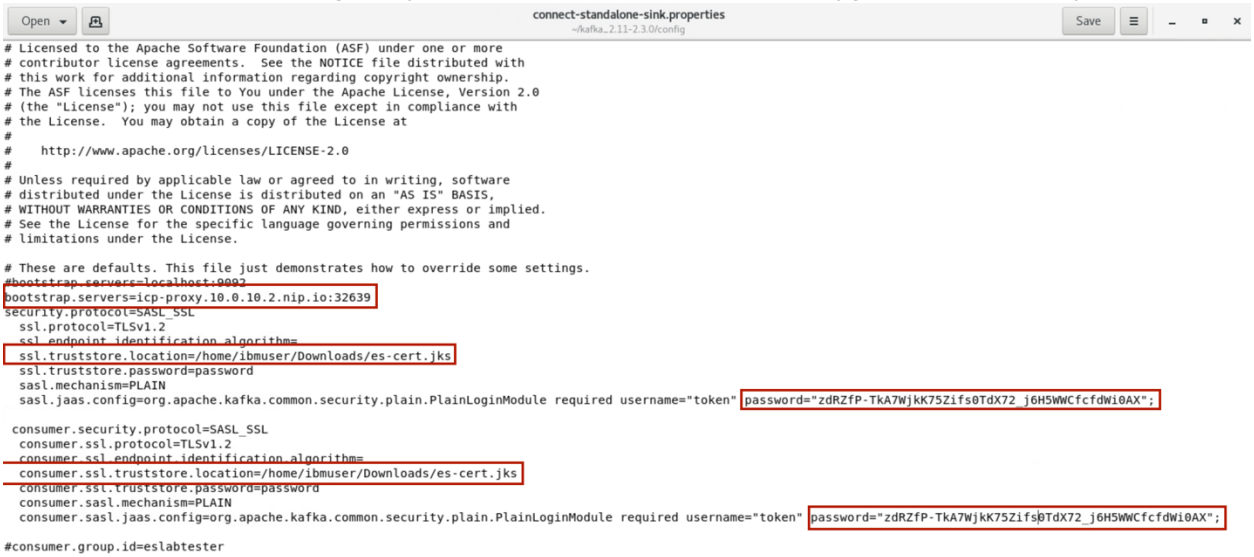
3. Open a terminal window and go to */home/ibmuser/Downloads*
   NOTE! Your API key will be different than the one shown below.

   Command: `gedit eventtomq.json`

4. From the terminal window, go to *home/ibmuser/kafka_2.11-2.3.0/config/* directory. A file has been prepared for our lab. There are two sections that you need to configure within the file: ssl and consumer.

   Command: `gedit connect-standalone-sink.properties`

   a. The first section you configure is security:
   - Set the bootstrap.servers=**icp-proxy.10.0.10.2.nip.io:32639**
   - Set ssl.truststore.location=/**home/ibmuser/Downloads/es-cert.jks**
   - The sasl.jaas.config variable, **update the API Key password** to the API Key copied from the eventtomq.json file (Step 3).



b. The second section you configure is the consumer:
   - Set consumer.ssl.truststore.location=/**home/ibmuser/Downloads/es-cert.jks**
   - The consumer.sasl.jaas.config variable, **update the API Key password** to the API Key copied from the eventtomq.json file (Step 3). Click **Save** and close the file.

Note

a. Each connector has a different API key.
b. The es-cert.jks location will be the same for consumer and producer.

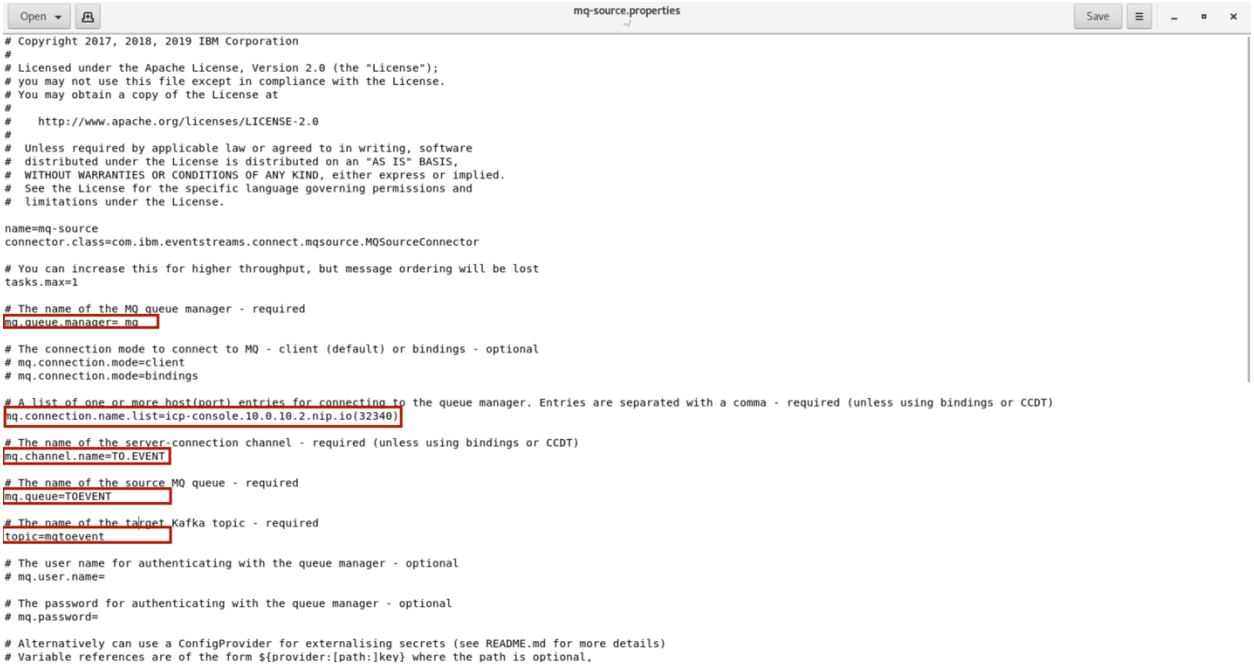
Task 7 – Configuring MQ Connectors

The connector copies messages from a source MQ queue to a target Kafka topic. There is also an MQ sink connector that takes messages from a Kafka topic and transfers them to an MQ queue.

We installed a local Kafka in the Developer Machine. You can check on **/home/ibmuser/kafka_2.11-2.3.0**. It is referred to as the Kafka root directory. It contains several directories including bin for the Kafka executables and config for the configuration files.

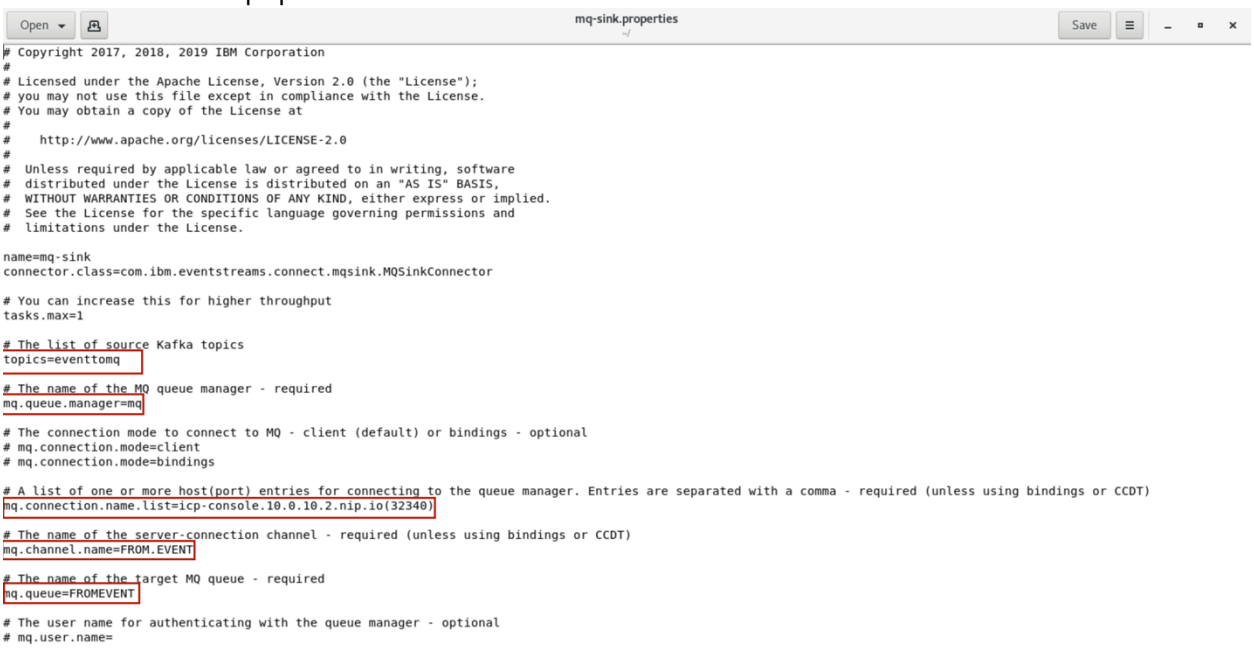
You need to update the connector config files, mq-source.properties and mq-sink.properties, for the values required to connect to your queue manager.

1. View the configure **mq-source.properties** (MQ to Event Streams). Open a terminal window and go to */home/ibmuser* directory. Type `gedit mq-source.properties`. The following variables are pre-set in this lab.
    a. mq.queue.manager=**mq**
    b. mq.connection.name.list=**icp-console.10.0.10.2.nip.io(32340)**
    c. mq.channel.name=**TO.EVENT**
    d. mq.queue=**TOEVENT**
    e. topic=**mqtoevent**



2. Click **Save** and close the file.

3. View the configured **mq-sink.properties** (Event Streams to MQ). Open a terminal window and go to */home/ibmuser* directory. Type `gedit mq-sink.properties`. The following variables are pre-set in this lab.
    a. topics=**eventtomq**
    b. mq.queue.manager=**mq**
    c. mq.connection.name.list=**icp-console.10.0.10.2.nip.io(32340)**
    d. mq.channel.name=**FROM.EVENT**
    e. mq.queue=**FROMEVENT**

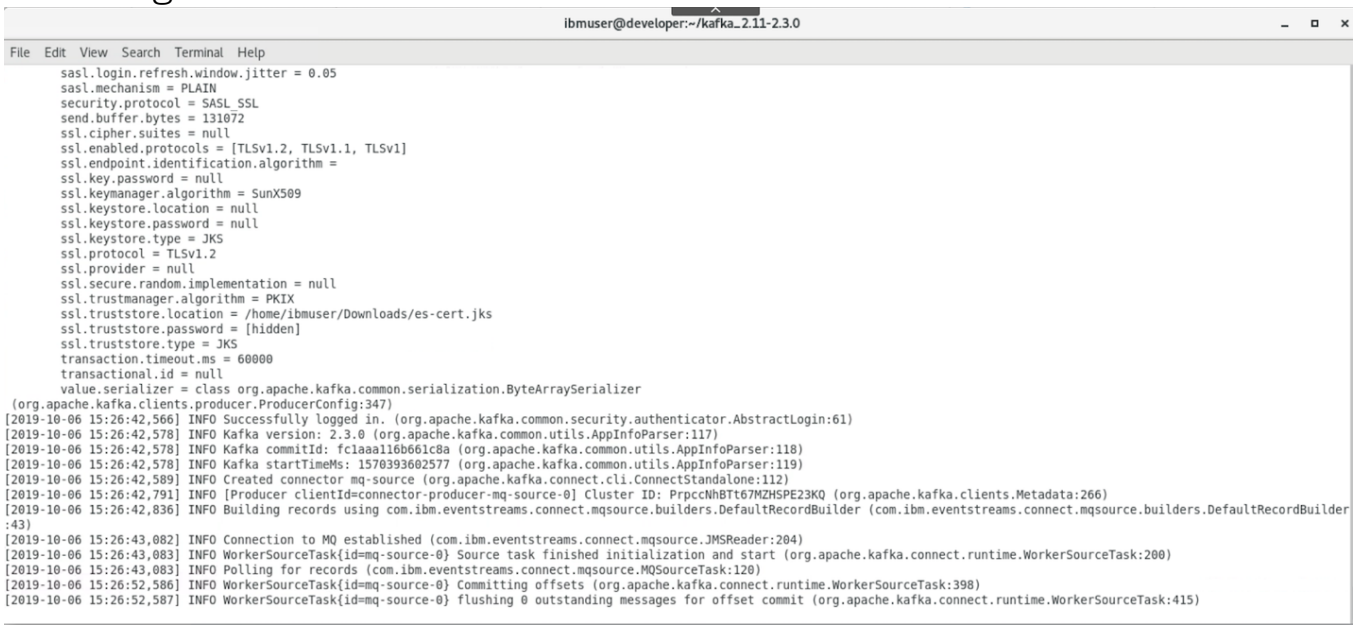


4. Click **Save** and close the file.

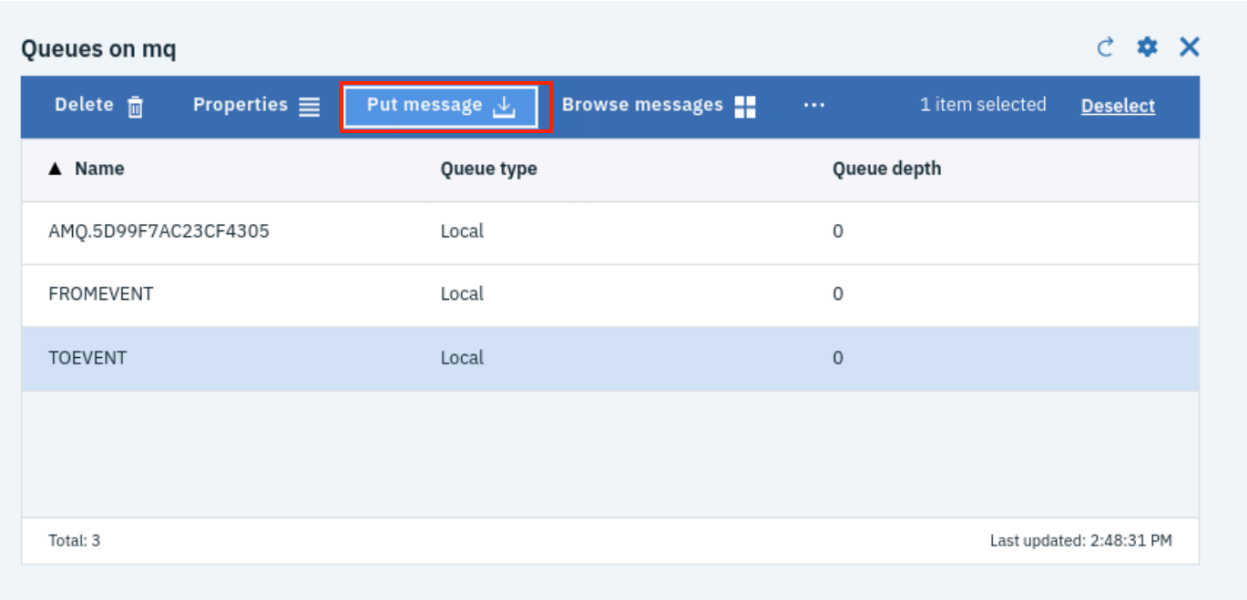
Task 8 – Executing and Testing MQ Connectors

To use an existing Kafka cluster, specify the connection information in the connector configuration file.

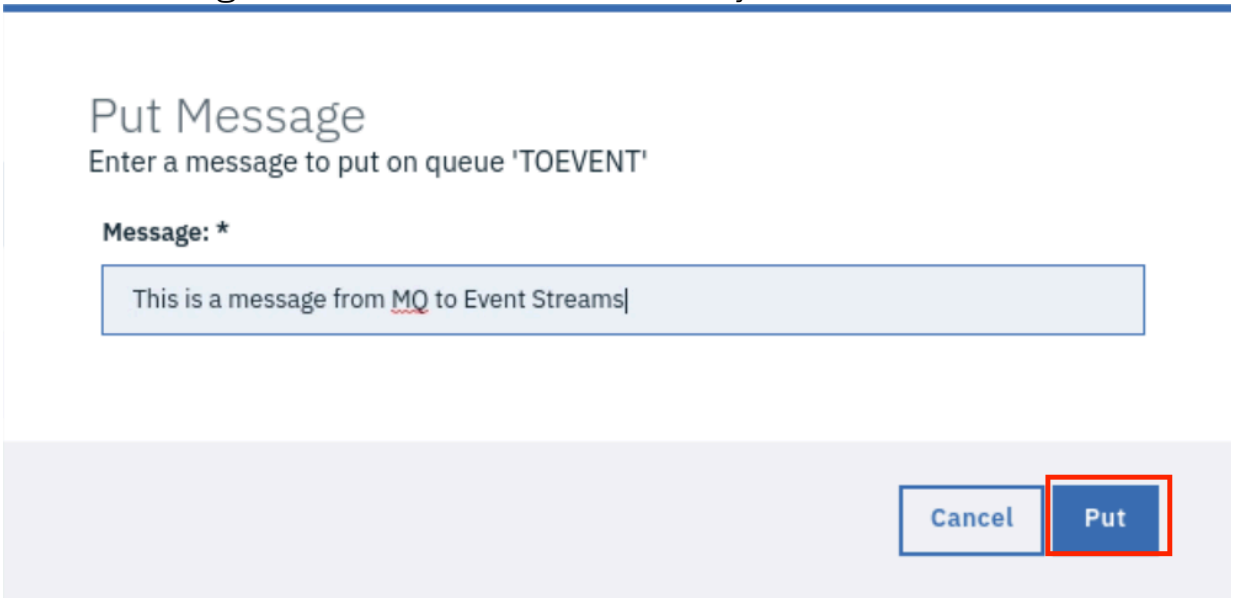
1. Start the connector by ensuring that you point at the new configuration. Open a terminal window and go to */home/ibmuser/kafka_2.11-2.3.0*. Type `./mqtoevent.sh` Check the log for errors.



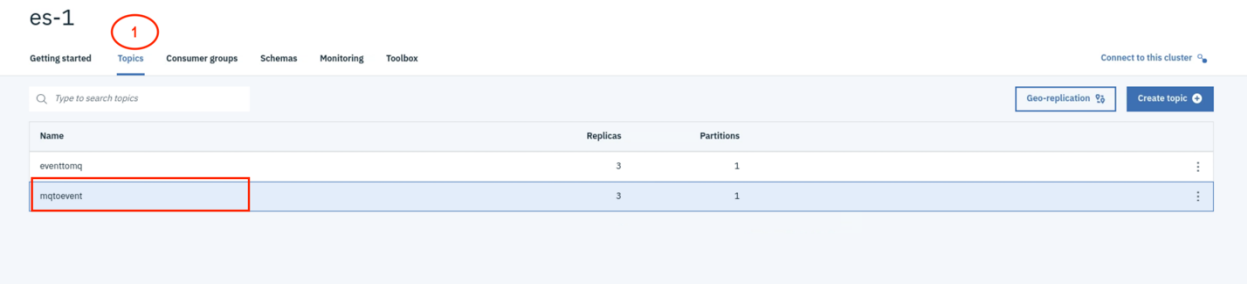
2. Go to the MQ console and in the ***Queues on mq*** widget, click the **TOEVENT** queue entry, then click the **Put message** button.



3. Type any text into **Message** and click **Put.** Notice that there is no message on the **TOEVENT** queue. As soon as you put a message, the connector immediately sends it to Event Streams.

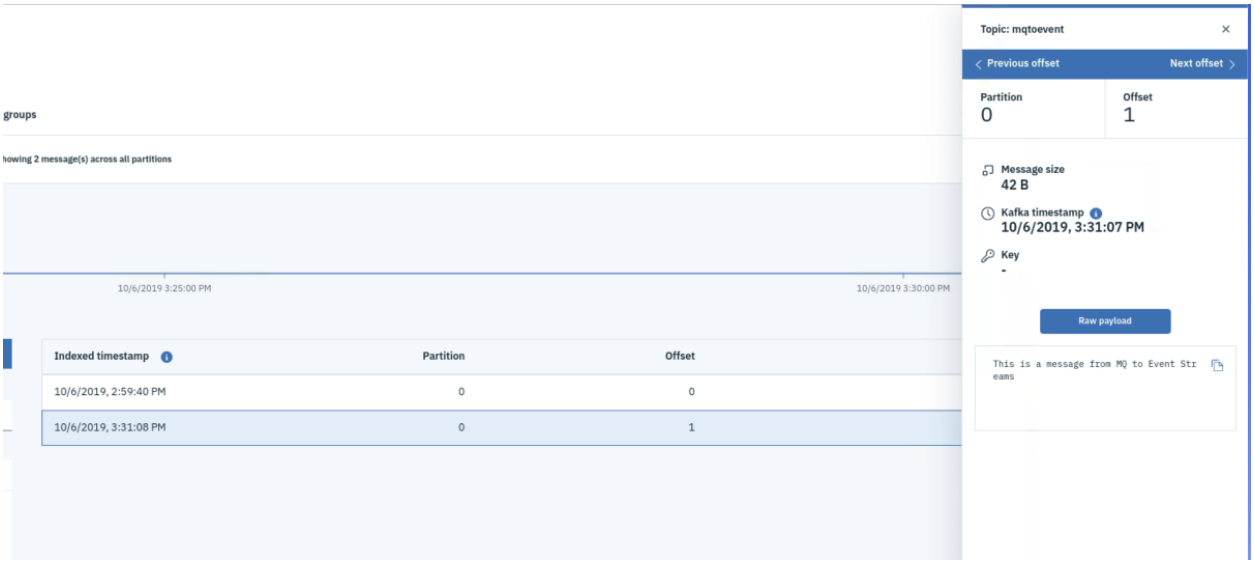


4. Return to the Event Streams main page and click **`Topics`**. Click the topic **mqtovent** to see the mqtovent messages.

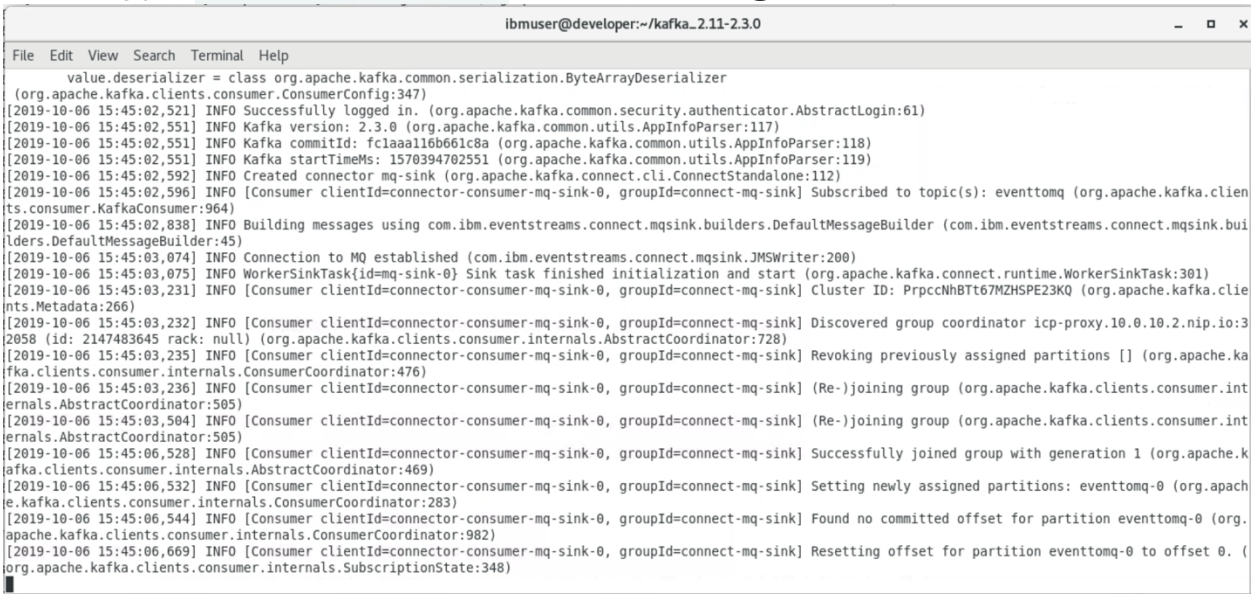


5. Click on **`Messages`**. You can view the timeframe of data to display (Hours). You can change this to Days, Hours, Minutes and Seconds. Click **on the message**.

6. You will see the message that you sent from MQ to Event Streams.



7. Keep the source connector running. Start the sink connector by ensuring that you point to the new configuration. Open a new terminal window and go to */home/ibmuser/kafka_2.11-2.3.0*. Type **`./eventtomq.sh`**. Check the log for errors.

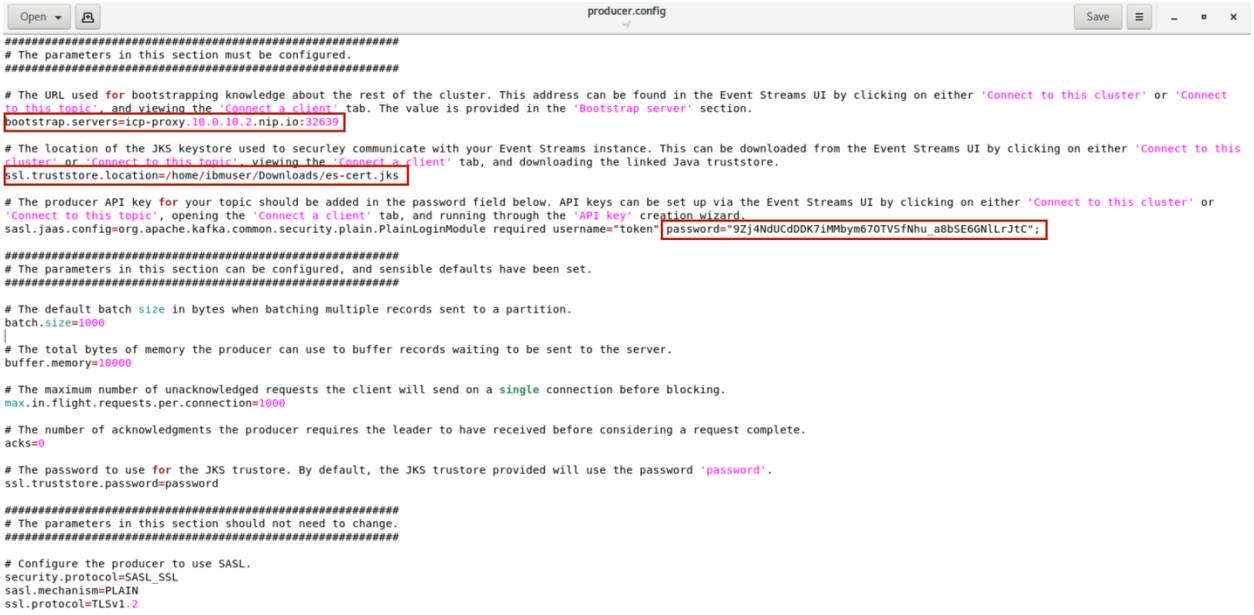


8. We will use a file to simulate events from Event Streams to MQ. The file is located in /home/ibmuser/Downloads/file_example_XLS_50.csv

9. To simulate a stream of events, we will use a **es-producer.jar**. Go to */home/ibmuser* directory and edit the file producer.config.

   Command: **`gedit producer.config`**

   The following parameters need to be set:
   a. Bootstrap server: **icp-proxy.10.0.10.2.nip.io:32639**
   b. ssl.truststore.location: **/home/ibmuser/Downloads/es-cert.jks**
   c. Update the password= It is the same API Key password from the **eventtomq.json** file

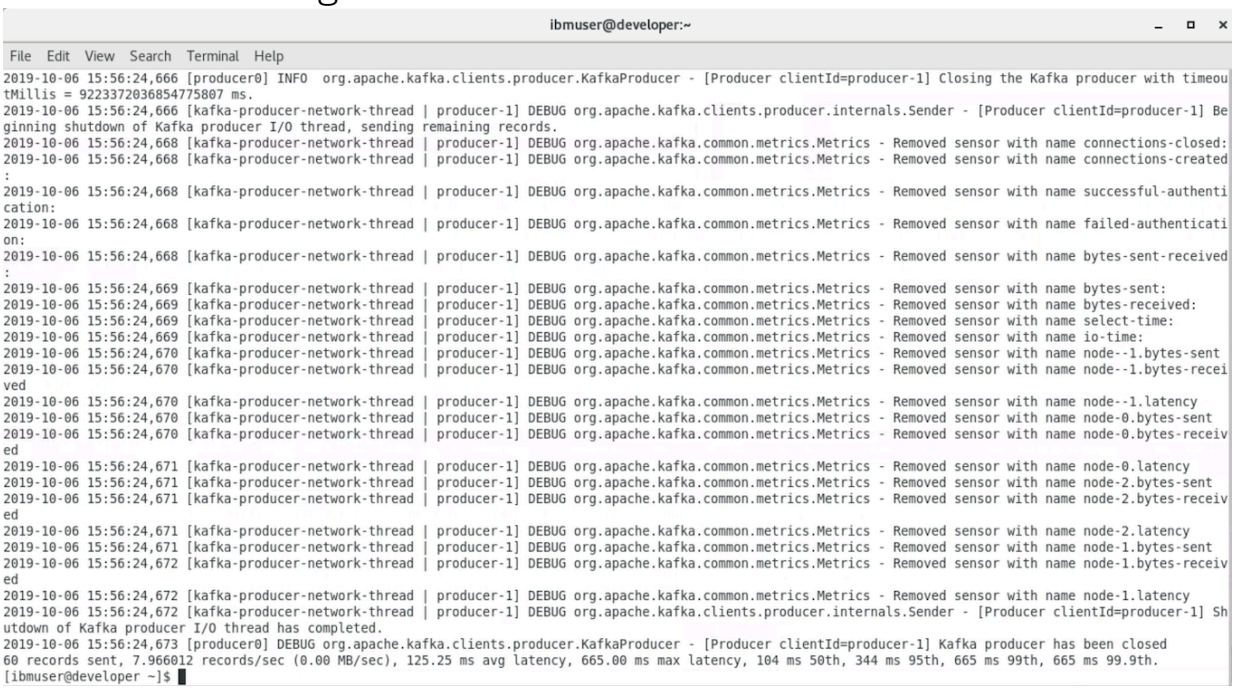


10. Now you will send messages to MQ. Let's use the java application, **es-producer.jar**, to produce events from Event Streams to MQ. We will send 60 messages.

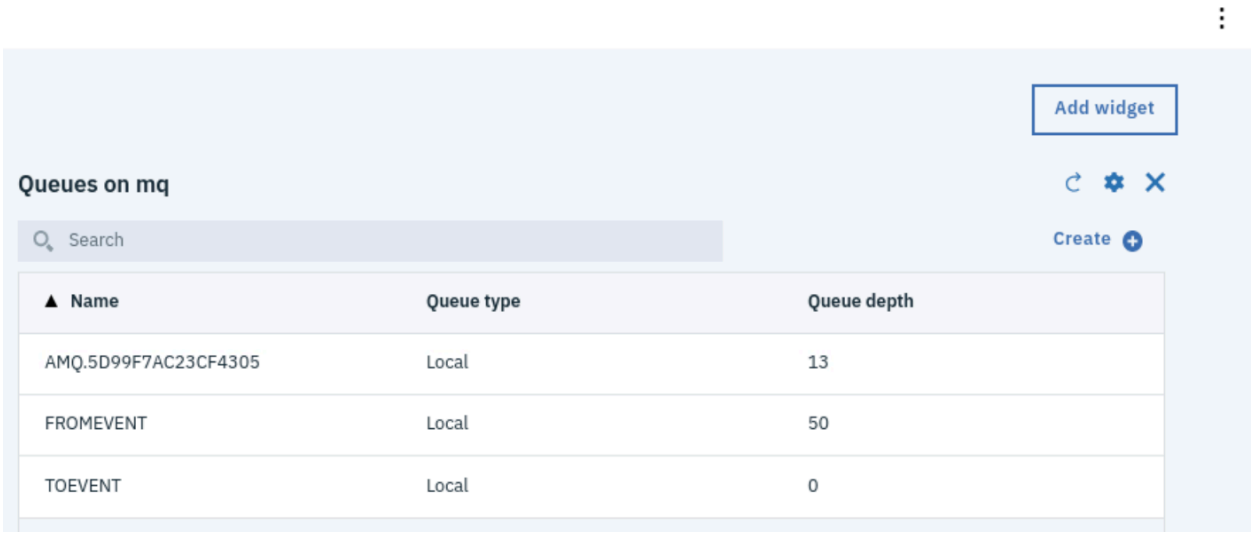
11. Open a new terminal window (we have three windows opened).

Command: `java -jar es-producer.jar -t eventtomq -T 1000 -n 50 -f /home/ibmuser/Downloads/file_example_XLS_50.csv`

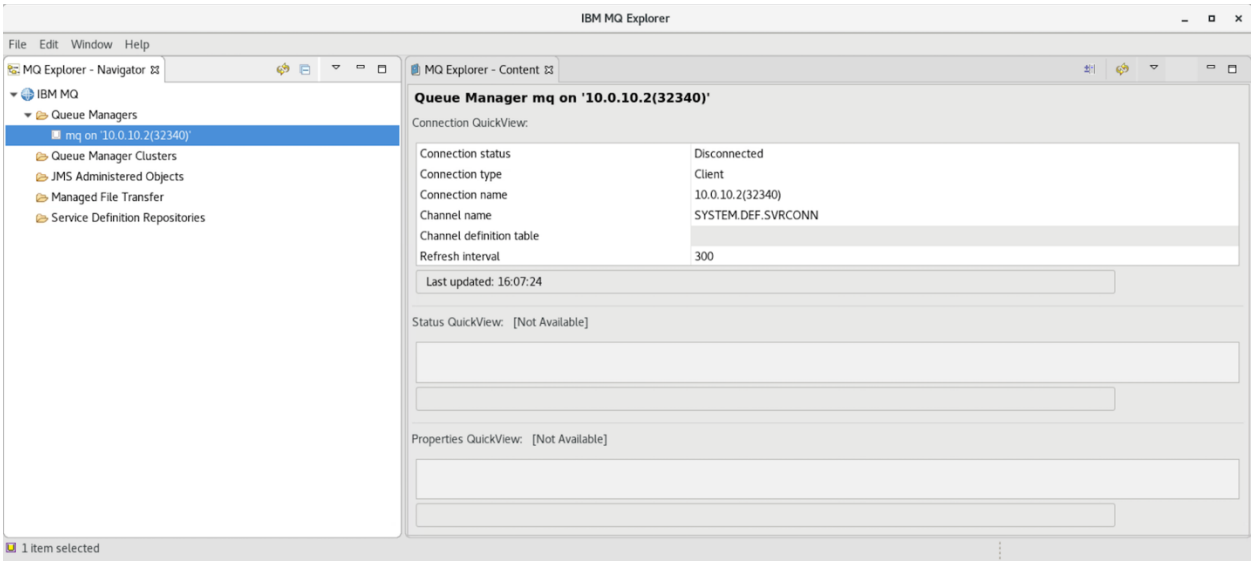
Have a look the log and see the results:



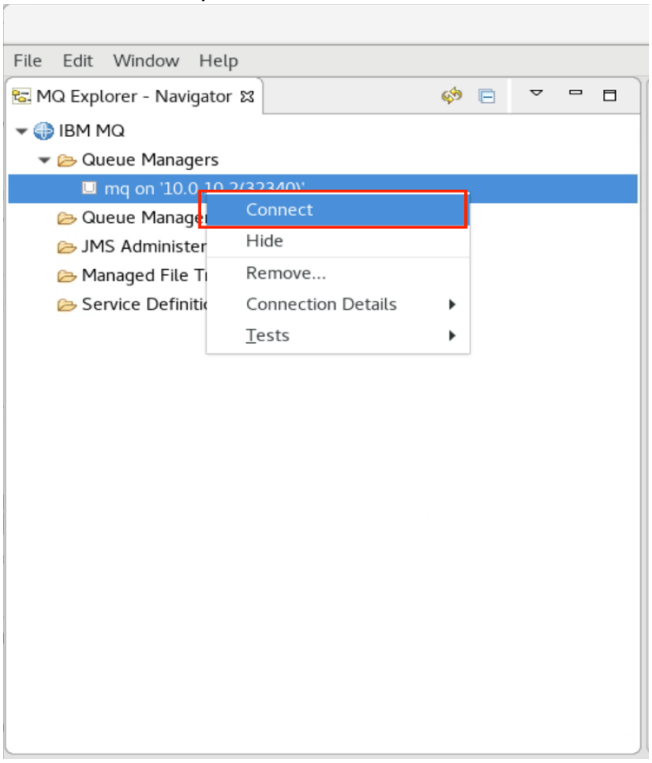
12. Go to the MQ Console. On the ***Queues on mq*** widget, click the **Refresh** icon and check the **Queue Depth.**



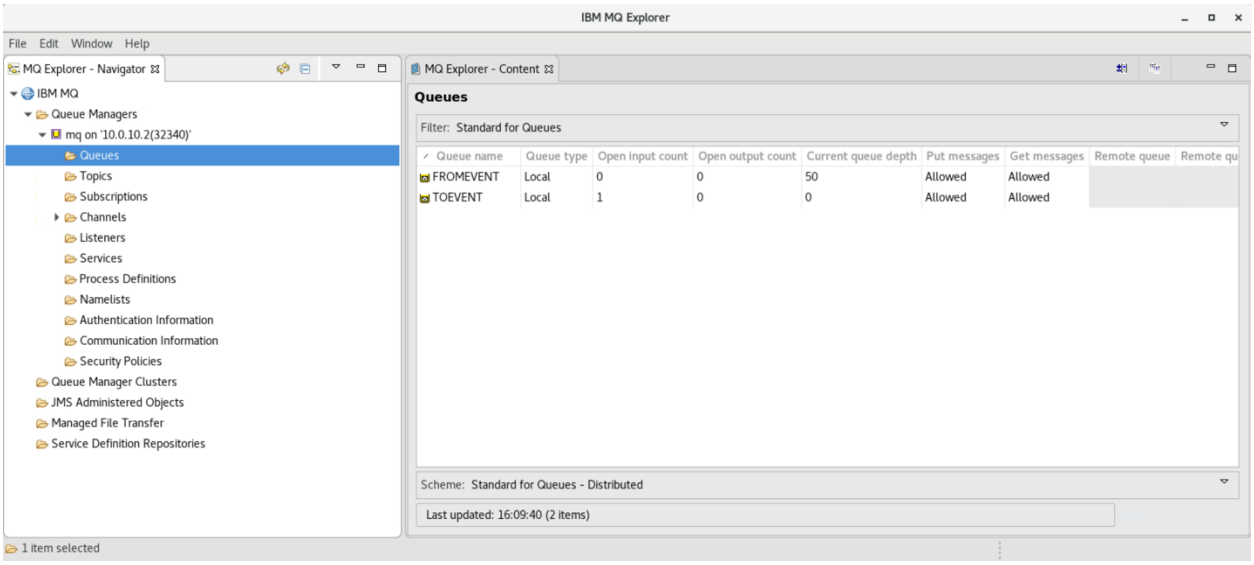
13. You can also use MQ Explorer to **Browse** message. Open a terminal window and type **MQExplorer.**



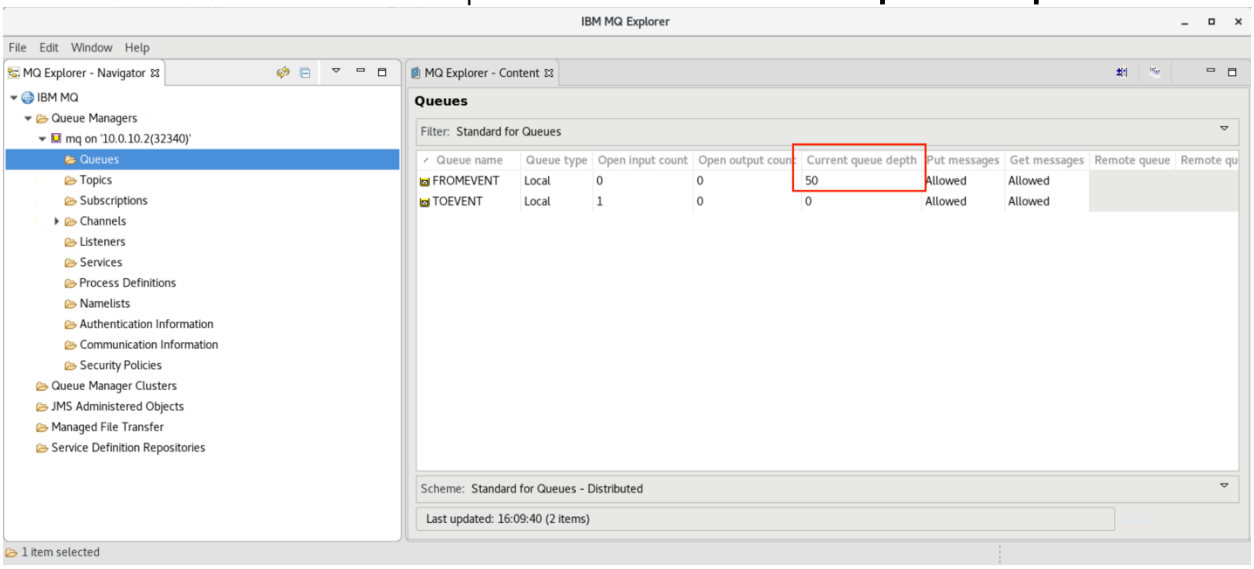
14. Click **connect** (right click) the queue manager **mq.** Click **Yes** when asked to replace the current connection.



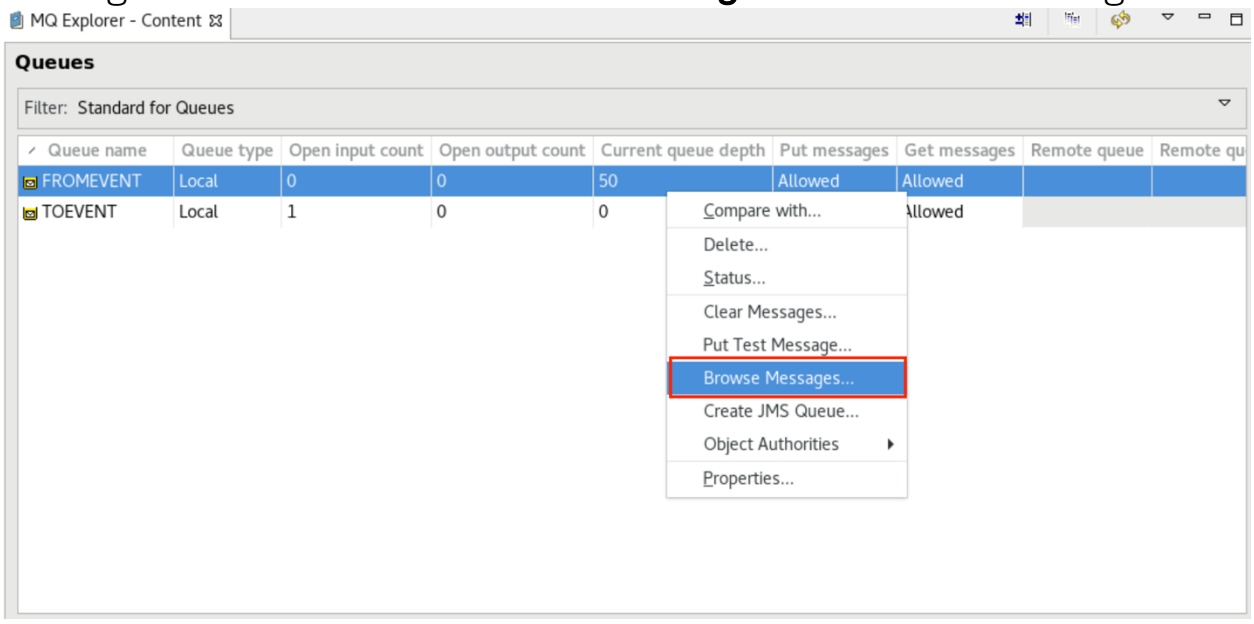
15. Click to open the Queues list, check the queues on the right window.

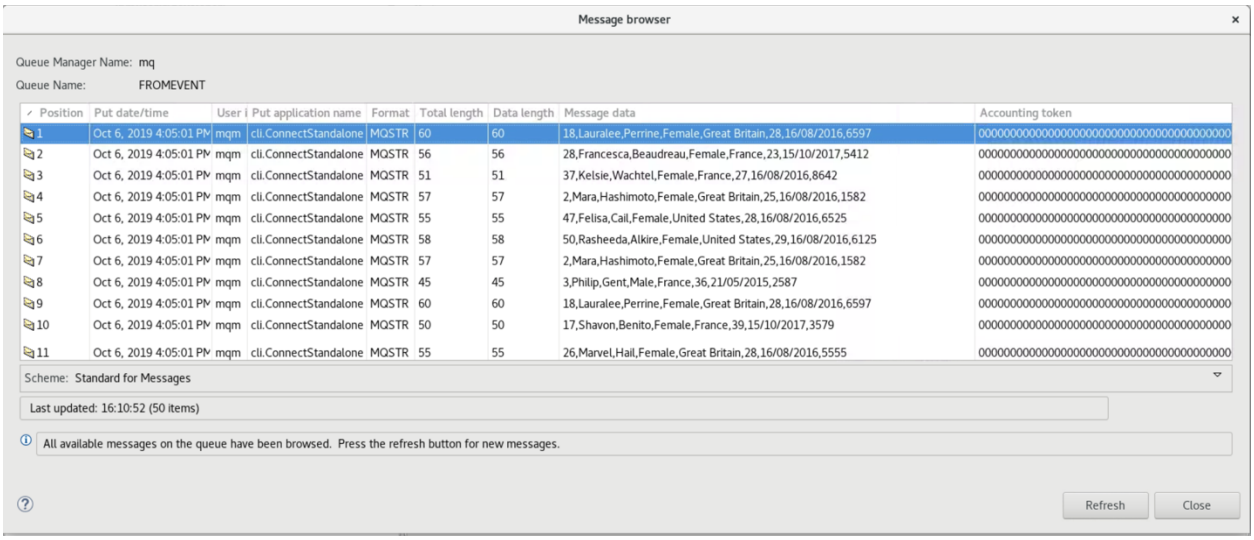


16. Select **FROMEVENT** queue. Look at **Current queue depth** column.



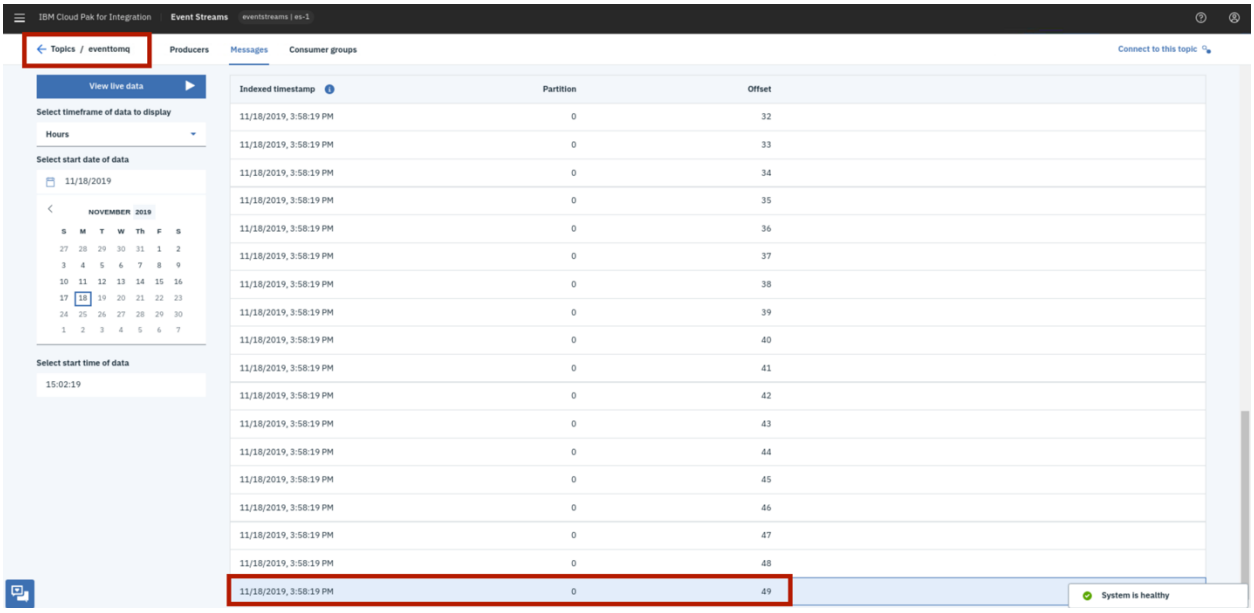
17. Right click and select **Browse Messages**. Check the messages.





18. Event Streams stored the messages that have been sent to MQ. Open **Event Streams** and select the **eventtomq topic.**

19. Click on **Messages**. Scroll through the messages. Note the latest offset number is 49.



Summary

Congratulations! You've successfully completed the tutorial. You were able to add a layer of secure, reliable, event-driven, and real-time data which can be re-used across applications in your enterprise. You learned how to:

- Configure message queues
- Create event streams topics
- Configure message queue connectors (sink and source)
- Execute a test run of the flow and view the data

To try out more labs, go to Cloud Pak for Integration Demos. For more information about the Cloud Pak for Integration, go to https://www.ibm.com/cloud/cloud-pak-for-integration.