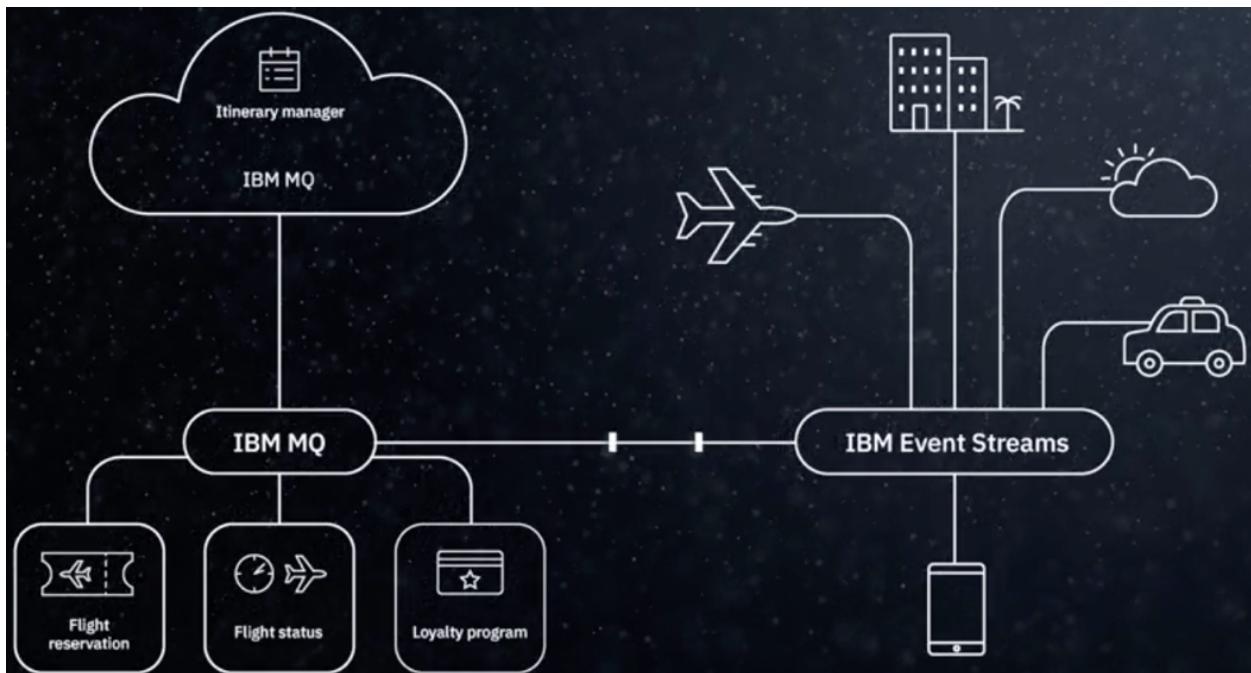


Connecting Message Queues and Kafka to stream data in real-time

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:

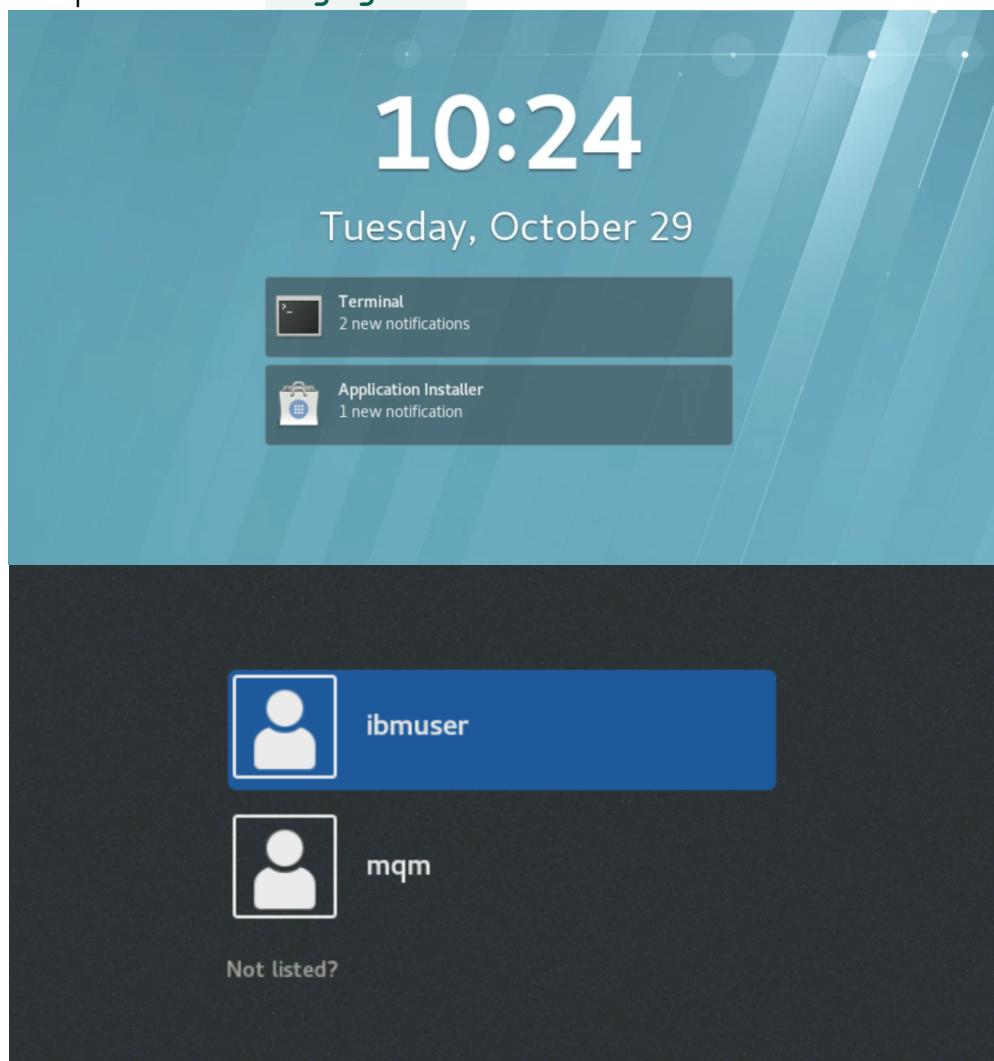
- Configure MQ to send and receive messages and events
- Configure Event Streams Topics: **mqtoevent** and **eventtomq**
- Configure MQ Source and Sync Connectors.
- Test connectors MQ source and sink (send a message and receive events).

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`

```
File Edit View Search Terminal Help
statefulset.apps/aspera-1-redis-ha-sentinel scaled
statefulset.apps/aspera-1-redis-ha-server scaled
deployment.extensions/assetrepo-1-asset-files-api scaled
deployment.extensions/assetrepo-1-catalog-api scaled
deployment.extensions/assetrepo-1-clt-haproxy scaled
deployment.extensions/assetrepo-1-dc-main scaled
deployment.extensions/assetrepo-1-portal-catalog scaled
deployment.extensions/assetrepo-1-portal-common-api scaled
deployment.extensions/assetrepo-1-redis-ha-sentinel scaled
deployment.extensions/assetrepo-1-redis-ha-server scaled
statefulset.apps/assetrepo-1-clt-db scaled
deployment.extensions/es-1-ibm-es-access-controller-deploy scaled
deployment.extensions/es-1-ibm-es-collector-deploy scaled
deployment.extensions/es-1-ibm-es-indexmgr-deploy scaled
deployment.extensions/es-1-ibm-es-proxy-deploy scaled
deployment.extensions/es-1-ibm-es-rest-deploy scaled
deployment.extensions/es-1-ibm-es-rest-producer-deploy scaled
deployment.extensions/es-1-ibm-es-rest-proxy-deploy scaled
deployment.extensions/es-1-ibm-es-ui-deploy scaled
statefulset.apps/es-1-ibm-es-elastic-sts scaled
statefulset.apps/es-1-ibm-es-kafka-sts scaled
statefulset.apps/es-1-ibm-es-schemaregistry-sts scaled
statefulset.apps/es-1-ibm-es-zookeeper-sts scaled
[ibmuser@developer ~]$ oc project eventstreams
Now using project "eventstreams" on server "https://master.ibm.demo:8443".
[ibmuser@developer ~]$ oc get pods -n integration
NAME                               READY   STATUS    RESTARTS   AGE
assetrepo-1-asset-files-api-9946b849d-wwlg   1/1     Running   0          3m
assetrepo-1-catalog-api-76cf9d474f-wpn62   1/1     Running   0          3m
assetrepo-1-clt-db-0                      2/2     Running   0          3m
assetrepo-1-clt-haproxy-7856dfb8d6-2jp4r   2/2     Running   0          3m
assetrepo-1-dc-main-5c467f68f7-z62p8       1/1     Running   0          3m
assetrepo-1-portal-catalog-867f9594c5-dvwr5   1/1     Running   0          3m
assetrepo-1-portal-common-api-66498db67b-9gwd2 1/1     Running   0          3m
assetrepo-1-redis-ha-sentinel-86ddcb88dd-ll2l2 1/1     Running   0          3m
assetrepo-1-redis-ha-server-b5cccd65c8-z54ds   1/1     Running   0          3m
ibm-icp4i-prod-ibm-icp4i-prod-6d5648d965-pqksh 2/2     Running   8          37d
```

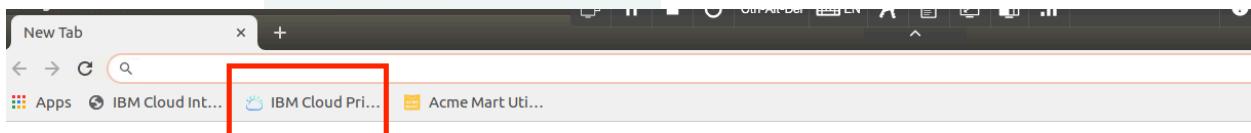
Note: The third command ensure the Cloud Pak is running. After the third command, once all the pods show 1/1 or equivalent proceed to the next step.

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.

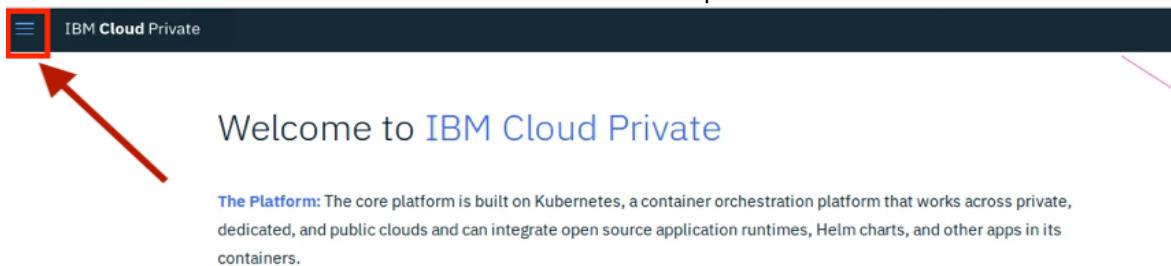


3. Log into IBM Cloud Private with the following credentials:

- a. username: **admin**
- b. password **admin**
- c. Click **Log in**

A screenshot of the IBM Cloud Private login page. At the top, there is a navigation bar with links for "OpenShift Web Console", "IBM Cloud Private" (which is highlighted with a red arrow), and "IBM Cloud Pak for Integ...". The main content features the text "Fast. Flexible. Intelligent. Open. Enterprise-grade.". Below this is a login form with fields for "Username" (containing "admin") and "Password" (containing "*****"). A red box highlights the "Log In" button at the bottom right of the form.

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



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

- X IBM Cloud Private
- [Dashboard](#)
- [Container Images](#)
- [Workloads](#)
- [Network Access](#)
- [Configuration](#)
- [Platform](#)
- [Manage](#)
 - [Identity & Access](#)
 - [Resource Security](#)
 - [Service Brokers](#)
 - [Helm Repositories](#)
- [Command Line Tools](#)
- [Getting started](#)

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

The screenshot shows the Helm Repositories page in the IBM Cloud Private interface. A modal dialog box is open, asking "Sync all Helm Repositories?". The dialog contains a message: "Are you sure you want to sync all repositories (6 items)? Only charts that are not syncing will be added to the queue. The process might take a few seconds to sync all of the charts." At the bottom of the dialog are two buttons: "Cancel" and "Sync", with "Sync" highlighted by a red circle labeled "1". In the background, the main table lists six repositories with their URLs. To the right of the table, a "Last Sync" column shows times ranging from 24 minutes ago. A red circle labeled "2" highlights the "Sync all" button in the top right corner of the main interface area.

Name	Url	Last Sync
ibm-charts	https://	24 minutes ago
local-charts	https://	24 minutes ago
mgmt-charts	https://	24 minutes ago
ibm-charts-public	https://	24 minutes ago
ibm-community-charts	https://	24 minutes ago
ibm-entitled-charts	https://raw.githubusercontent.com/IBM/charts/master/repo/entitled/	Completed

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.

```
ibmuser@developer:~ ibmuser@developer:~
```

```
File Edit View Search Terminal Help
```

```
: AUTHADD(BROWSE,CHG,CLR,DLT,DSP,GET,INQ,PUT,PASSALL,PASSID,SET,SETALL,SETID)
AMQ8862I: IBM MQ authority record set.
195 : SET AUTHREC +
: PROFILE('FROMEVENT') +
: GROUP('mqm') +
: OBJTYPE(QUEUE) +
: AUTHADD(BROWSE,CHG,CLR,DLT,DSP,GET,INQ,PUT,PASSALL,PASSID,SET,SETALL,SETID)
AMQ8862I: IBM MQ authority record set.
196 : SET AUTHREC +
: PROFILE('SYSTEM.ADMIN.CHANNEL.EVENT') +
: GROUP('mqm') +
: OBJTYPE(QUEUE) +
: AUTHADD(BROWSE,CHG,CLR,DLT,DSP,GET,INQ,PUT,PASSALL,PASSID,SET,SETALL,SETID)
AMQ8862I: IBM MQ authority record set.
:
197 : REFRESH SECURITY
AMQ8560I: IBM MQ security cache refreshed.
: ****
: * Script ended on 2019-10-02 at 21.13.27
: * Number of Inquiry commands issued: 14
: * Number of Inquiry commands completed: 14
: * Number of Inquiry responses processed: 200
: * QueueManager count: 1
: * Queue count: 60
: * NameList count: 3
: * Process count: 1
: * Channel count: 13
: * AuthInfo count: 4
: * Listener count: 2
: * Service count: 2
: * CommInfo count: 1
: * Topic count: 6
: * Subscription count: 1
: * ChlAuthRec count: 3
: * Policy count: 1
: * AuthRec count: 102
: * Number of objects/records: 200
: ****
197 MQSC commands read.
No commands have a syntax error.
One valid MQSC command could not be processed.
command terminated with exit code 10
[ibmuser@developer ~]$
```

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.

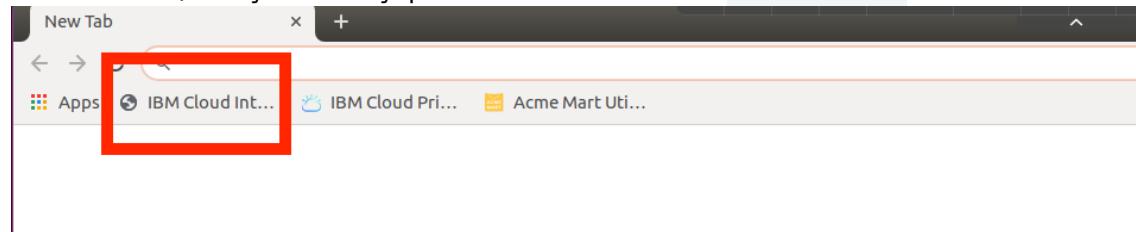
Unlock the power of your data and support the scale required for all of your integration and digital transformation initiatives.

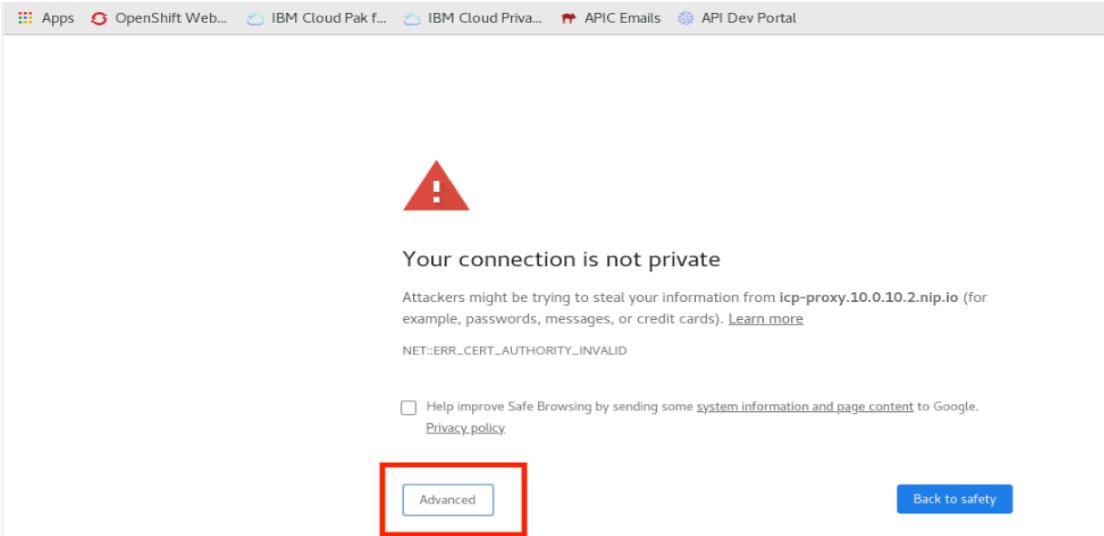
The home page of the **Cloud Pak for Integration** is referred to as the **Platform Navigator**. From the Platform Navigator you are able to navigate to all the integration and development technology contained within the platform.

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.

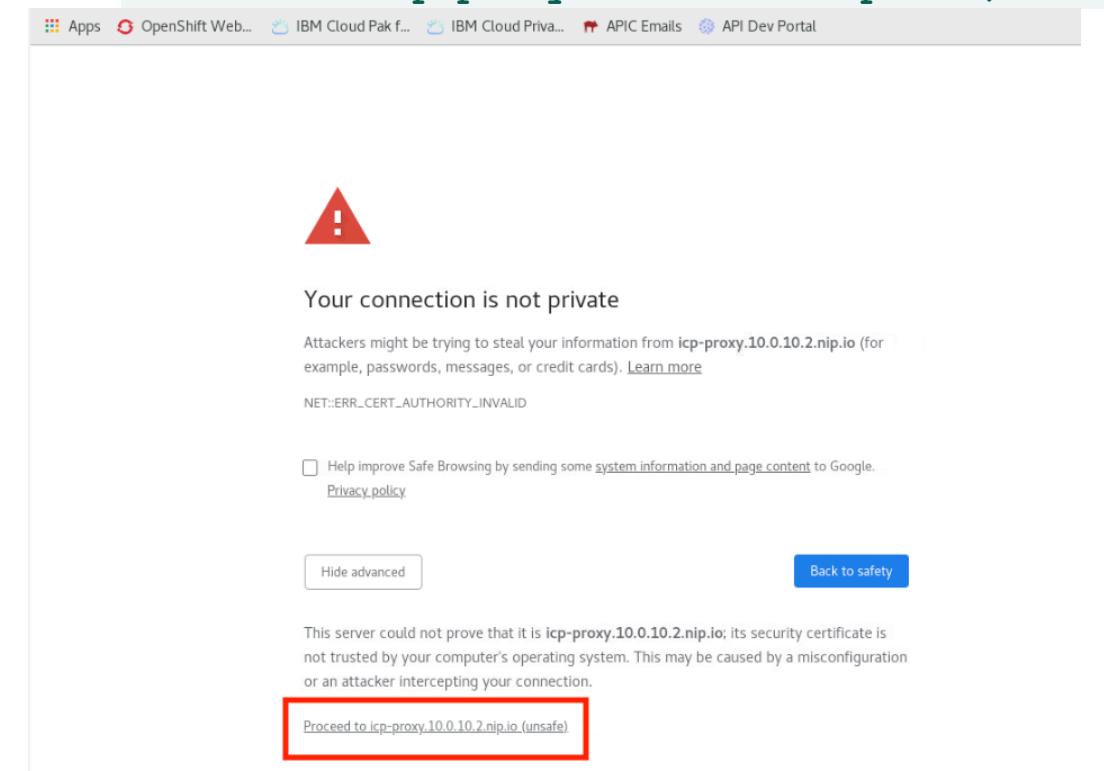
1. Click the **IBM Cloud Pak for Integration** bookmark in the bookmarks bar. If you see the login page, please continue to Task 2 (p. 11)

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**.





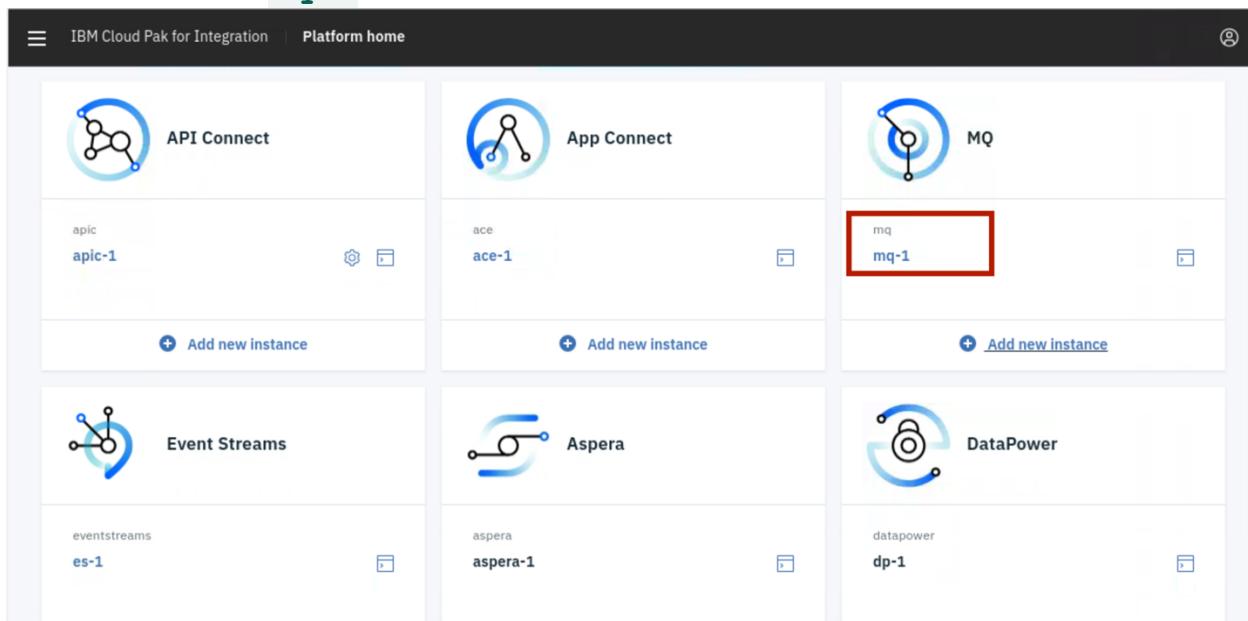
2. Click **Proceed to icp-proxy.10.0.10.2.nip.io (unsafe)**.



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.

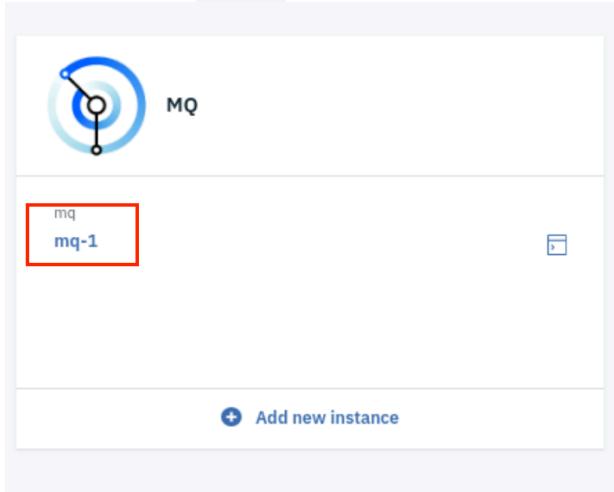
1. In your browser click the **IBM Cloud Pak** bookmark.
 - a. To view the Cloud Pak for Integration main page, click on the **Skip Welcome** button.
2. On the Cloud Pak for Integration main page, locate the **MQ** service and click **mq-1**.



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.

A screenshot of the IBM MQ Local Queue Managers page. The title bar shows 'Apps', 'OpenShift Web...', 'IBM Cloud Pak f...', and 'IBM Cloud Priv...'. Below the title, there's a search bar and a table titled 'Local Queue Managers'. The table has two columns: 'Name' and 'Status'. It shows one entry: 'mq' with a status of 'Running' and a green dot icon. At the bottom of the table, it says 'Total: 1' and 'Last updated: 12:33:13 PM'. In the top right corner of the main content area, there's a red box highlighting a blue rectangular button labeled 'Add widget'.

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

Add a new widget

Local Queue Managers Manage local queue managers

Chart Monitor your MQ platform

Add a widget to display MQ object information for the specified queue manager

Queue manager:

mq

Queues Configure destinations for messages

Topics Administrative objects for assigning attributes to topics

Listeners Configure processes to accept network requests

Channels Queue manager communication paths

Client-connection Channels Client connectivity details

Authentication Information Configure authentication mechanisms

Subscriptions Configure how subscriptions to topics are handled

Channel Authentication Records Control access to channels

Close

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

Add widget

Queues on mq

Search

Create +

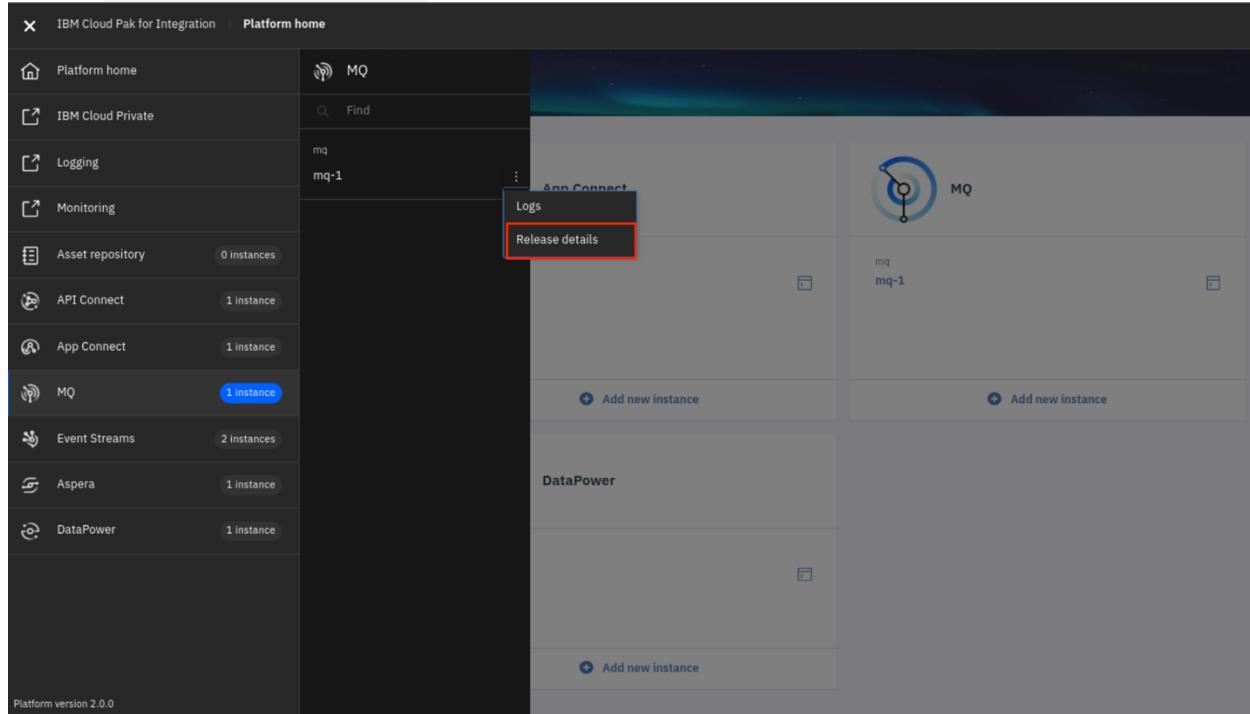
Name	Queue type	Queue depth
AMQ.5DD4237222DEBC05	Local	13
FROMEVENT	Local	0
TOEVENT	Local	0

Total: 3 Last updated: 12:09:22 PM

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**.

The screenshot shows the IBM Event Streams interface. At the top, there are tabs: 'Getting started' (blue), 'Topics' (red border, white text), 'Consumer groups', 'Schemas', 'Monitoring', and 'Toolbox'. On the right, there's a 'Connect to this cluster' button. Below the tabs, a message says 'Welcome to IBM Event Streams, let's get you up and running...'. There are two main sections: 'Use a simulated topic' (with an icon of a gear and pipe) and 'Generate a starter application' (with an icon of a coffee machine). To the right, there's a sidebar with 'Learn more...' and links to 'Kafka basics', 'Schema basics', and 'Kafka Connect basics'. At the bottom right, there's a green box with a checkmark and the text 'System is healthy'.

5. Click on **Create Topic**

The screenshot shows the Event Streams interface with the 'Topics' tab selected. At the top right, there is a 'Create topic' button with a plus sign, which is highlighted with a red box.

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**

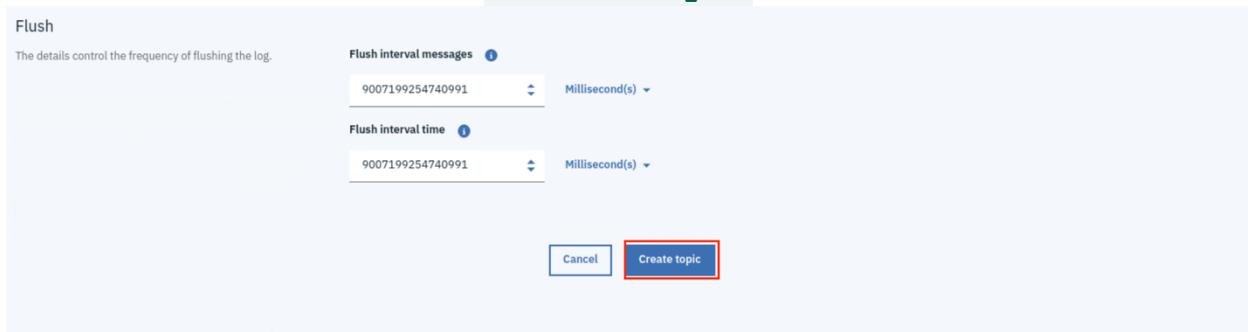
The screenshot shows the 'Core configuration' section of the 'Create topic' form. It includes fields for Topic name, Partitions, Replicas, Minimum in-sync replicas, and Retention time. Each field is circled with a red number corresponding to the steps in the list above:

- 1. Topic name: mqtoevent
- 2. Partitions: 1
- 3. Replicas: 3
- 4. Minimum in-sync replicas: 2
- 5. Retention time: 10 Minute(s)

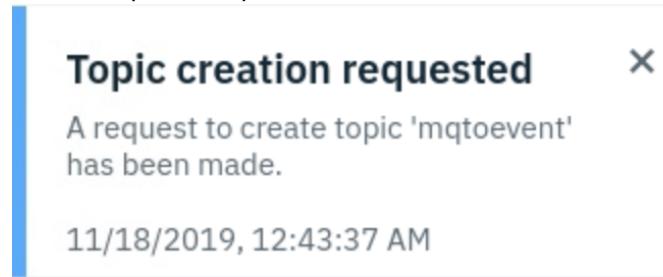
Note:

- 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**.

The screenshot shows the IBM Cloud Pak for Integration Event Streams interface. The top navigation bar includes 'IBM Cloud Pak for Integration' and 'Event Streams' with the path 'eventstreams | es-1'. On the right are help and user icons. Below the navigation is a search bar with placeholder 'Type to search topics' and a 'Create topic +' button. The main area displays a table of topics:

Name	Replicas	Partitions	⋮
eventtomq	3	1	⋮
mqtoevent	3	1	⋮

A blue icon with a speech bubble is on the left, and a green checkmark with 'System is healthy' is on the right.

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.

The screenshot shows the IBM Cloud Pak for Integration interface for a cluster named 'es-1'. The top navigation bar includes 'IBM Cloud Pak for Integration', 'Event Streams', 'eventstreams | es-1', and a 'Connect to this cluster' button. Below the navigation is a horizontal menu with tabs: 'Getting started', 'Topics', 'Consumer groups', 'Schemas', 'Monitoring', and 'Toolbox' (which is highlighted with a red box). On the left, there's a sidebar with sections for 'Connectors' and 'Monitor'. The main content area contains four tiles under the 'Connectors' section:

- Kafka Connect | 1 Set up a Kafka Connect environment**: A brief description of setting up Kafka Connect for streaming data between Event Streams and other systems. Includes a 'Set up' button.
- Kafka Connect | 2 Add connectors to your Kafka Connect environment**: A brief description of preparing Kafka Connect for connections to other systems. Includes a 'Add connectors' button and a 'Connecting to IBM MQ?' link.
- Kafka Connect | 3 Start Kafka Connect with your connectors**: A brief description of using the Kafka Connect API to configure and run new connectors. Includes a 'Start Kafka Connect' button.
- Connector catalog**: A brief description of viewing and downloading connectors for other systems. Includes a 'View Catalog' button.

At the bottom right of the main content area, there's a green checkmark icon with the text 'System is healthy'.

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

The screenshot shows a web interface for adding connectors to a Kafka Connect environment. At the top, there's a navigation bar with 'IBM Cloud Pak for Integration', 'Event Streams', and 'eventstreams | es-1'. Below the navigation is a 'Toolbox' button. The main content area has a title 'Add connectors to your Kafka Connect environment'. It includes a note about the Kafka Connect ZIP containing FileSink and FileSource connectors. Two sections are shown: 'Connector catalog' (with a 'Launch the Connector catalog' button) and 'Featured IBM MQ' (with a 'Launch' button). A red box highlights the 'IBM MQ connectors' link. Below these sections, there's a command-line instruction: `cp <path_to_your_connector>.jar <extracted_zip>/connectors`. The entire screenshot is framed by a red border.

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*.

The screenshot shows the 'IBM MQ connectors' page. At the top, it says 'To move data between Event Streams and MQ, choose the source or sink MQ connector and add it to your Kafka Connect environment. Then, start Kafka Connect and the connector.' Below this is a diagram showing a flow from MQ to IBM Event Streams and back to MQ through Source and Sink MQ connectors. The 'MQ Source' tab is selected. A red box highlights the 'Download MQ Source JAR' button. Another button, 'Download MQ Source configuration', is also visible. The entire screenshot is framed by a red border.

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*.

The screenshot shows the 'IBM MQ connectors' interface. At the top, there's a diagram illustrating the flow from an 'MQ' source to 'IBM Event Streams' through a 'Source MQ connector', and from 'IBM Event Streams' to another 'MQ' through a 'Sink MQ connector'. Below the diagram, there are two buttons: 'MQ Source' (disabled) and 'MQ Sink' (selected). Underneath these buttons, there are two red-bordered boxes: 'Download MQ Sink JAR' (with the sub-instruction 'Get the connector JAR') and 'Download MQ Sink configuration' (with the sub-instruction 'Get the configuration details').

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

a. **`cp ./Downloads/*.jar /home/ibmuser/`**

The terminal window shows the command `cp ./Downloads/*.jar /home/ibmuser/` being run and its output. The output lists several files copied into the */home/ibmuser* directory, including *ace-11.0.0.5*, *aceconfig*, *certs*, *config_services*, *Desktop*, *Documents*, *Downloads*, *es-producer.jar*, *files*, *firefox*, *IBM*, *kafka 2.11-2.3.0*, *kafka-connect-mq-sink-1.1.0-jar-with-dependencies.jar*, *kafka-connect-mq-source-1.1.0-jar-with-dependencies.jar*, *LICENSE*, *linux-amd64*, and *loadmqace.sh*. The *certs* folder and the two Kafka connector JAR files are highlighted with red boxes.

6. You will need to configure the sink and source connectors.

a. `cp ./files/*.properties /home/ibmuser/`

```
File Edit View Search Terminal Help
[ibmuser@developer ~]$ cp ./files/*.properties /home/ibmuser/
[ibmuser@developer ~]$ ls
ace-11.0.0.5
aceconfig
certs
config_services
Desktop
Documents
Downloads
es-producer.jar
files
firefox
IBM
kafka_2.11-2.3.0
kafka-connect-mq-sink-1.1.0-jar-with-dependencies.jar
kafka-connect-mq-source-1.1.0-jar-with-dependencies.jar
LICENSE
linux-amd64
loadmqace.sh
loadmqaspera.sh
loadmqes.sh
loadmq.sh
mq_conf
MQ Explorer
mq-sink.properties
mq-source.properties
Music
Pictures
producer.config
Public
scp-hosts.sh
startup-scripts
Templates
Toolkit.log
Videos
```

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

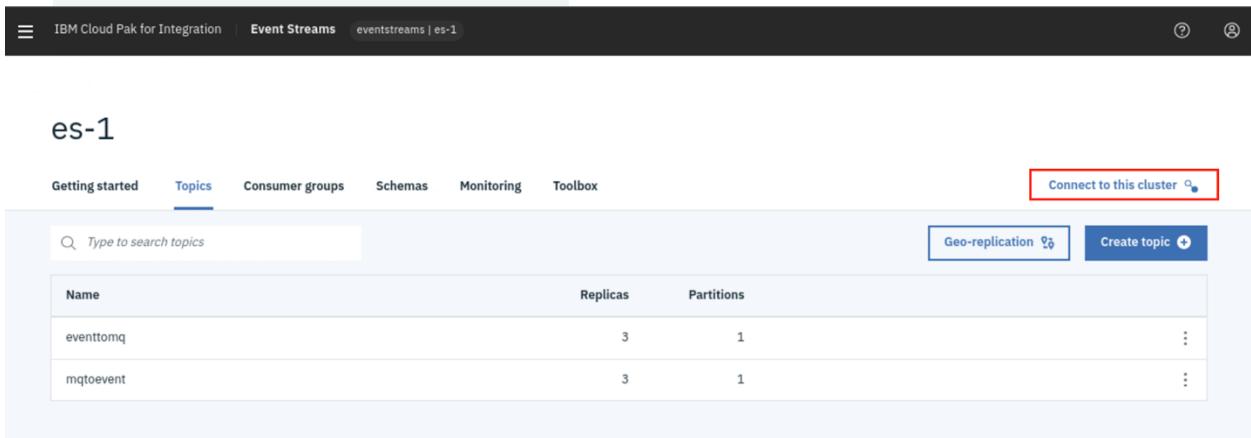
<https://github.com/ibm-messaging/kafka-connect-mq-source/tree/master/config>

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.



The screenshot shows the IBM Cloud Pak for Integration interface. At the top, there's a navigation bar with 'IBM Cloud Pak for Integration' and 'Event Streams' selected. Below it, the instance name 'eventstreams | es-1' is shown. The main area is titled 'es-1'. It has a navigation bar with 'Getting started' (disabled), 'Topics' (selected), 'Consumer groups', 'Schemas', 'Monitoring', and 'Toolbox'. A red box highlights the 'Connect to this cluster' button in the top right corner of the main content area. The 'Topics' section contains a search bar 'Type to search topics', a 'Geo-replication' button, and a 'Create topic' button. A table lists two topics: 'eventtomq' and 'mqtoevent'. Both topics have 3 replicas and 1 partition. Each row has a three-dot menu icon on the far right.

Name	Replicas	Partitions	⋮
eventtomq	3	1	⋮
mqtoevent	3	1	⋮

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.

Cluster connection

Connect a client Sample code Geo-replication

To connect an application or tool to this cluster, you will need the address of a bootstrap server, a certificate and an API key.

Bootstrap server
Your application or tool will make its initial connection to the cluster using the bootstrap server.
icp-proxy.10.0.10.2.nip.io:32639

Certificates
A certificate is required by your Kafka clients to connect securely to this cluster.

Java truststore
Use this for a Java client

Truststore password: password

PEM certificate
Use this for anything else

API key
To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.

Name your application: **mqtoevenapp** (1)

What do you want it to do?

- Produce only →
- Consume only →
- Produce and consume →**
- Produce, consume, create topics and schemas →

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.

Cluster connection

Bootstrap server
Your application or tool will make its initial connection to the cluster using the bootstrap server.

icp-proxy.10.0.10.2.nip.io:32639

Certificates
A certificate is required by your Kafka clients to connect securely to this cluster.

Java truststore
Use this for a Java client

Truststore password: password

PEM certificate
Use this for anything else

API key
To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.

Which topic?
mqtoevent! (1)

All topics

Great! This topic already exists

Start again

Next >

Connecting your tool or application will generate a service ID using your application name, a service policy and an API key in IBM Cloud Private. Additional API keys can be generated in the IBM Cloud Private Cluster Management Console or CLI.

IBM Cloud Private Cluster Management Console

- 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.

The screenshot shows the 'Cluster connection' interface. At the top, there are three tabs: 'Connect a client' (selected), 'Sample code', and 'Geo-replication'. Below the tabs, a message states: 'To connect an application or tool to this cluster, you will need the address of a bootstrap server, a certificate and an API key.' On the left, under 'Certificates', there are two options: 'Java truststore' (selected) and 'PEM certificate'. Both options include a download link and a note about their use. On the right, under 'API key', there is a section titled 'Which consumer group?' with a dropdown menu and a 'Start again' button. A red box highlights the 'Generate API key' button, which is located below the consumer group selection. A note at the bottom explains that connecting a tool or application will generate a service ID using the application name, service policy, and API key.

Cluster connection

Connect a client Sample code Geo-replication

To connect an application or tool to this cluster, you will need the address of a bootstrap server, a certificate and an API key.

Bootstrap server
Your application or tool will make its initial connection to the cluster using the bootstrap server.

icp-proxy.10.0.10.2.nip.io:32639

Certificates
A certificate is required by your Kafka clients to connect securely to this cluster.

Java truststore
Use this for a Java client

Truststore password: password

PEM certificate
Use this for anything else

API key
To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.

Start again ⏪

Which consumer group? All

Enter your consumer group name

Generate API key

Connecting your tool or application will generate a service ID using your application name, a service policy and an API key in IBM Cloud Private. Additional API keys can be generated in the IBM Cloud Private Cluster Management Console or CLI.

IBM Cloud Private Cluster Management Console

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

Click the **download** icon.

The screenshot shows the 'Cluster connection' interface. At the top, there are three tabs: 'Connect a client' (selected), 'Sample code', and 'Geo-replication'. Below the tabs, a message states: 'To connect an application or tool to this cluster, you will need the address of a bootstrap server, a certificate and an API key.' On the left, under 'Bootstrap server', the URL 'icp-proxy.10.0.10.2.nip.io:32639' is displayed with a download icon. Under 'Certificates', it says 'A certificate is required by your Kafka clients to connect securely to this cluster.' It lists two options: 'Java truststore' (use for Java client) and 'PEM certificate' (use for anything else). On the right, under 'API key', it says 'To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.' A success message box is open, stating 'Successful API key generation!' and 'The following API key will allow an application to produce to and consume from topic mqtoevent using all consumer groups. Take a copy of it or download it now.' It contains a note: 'Please make a note of this API key now. You will not be able to recover this later!' with a red border around it. The API key value is 'rq-inNFNb-ahK-Q4Rha-Kd_Oh1dDMPiyEKi' with download and copy icons. Below the message box, it says 'A service ID has been generated in IBM Cloud Private with your application name. This can be managed in the IBM Cloud Private Cluster Management Console or CLI.' At the bottom right of the message box is a link 'Create a new API key'.

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**

The screenshot shows the 'Cluster connection' section of the Event Streams console. It includes tabs for 'Connect a client', 'Sample code', and 'Geo-replication'. Below these tabs, a message states: 'To connect an application or tool to this cluster, you will need the address of a bootstrap server, a certificate and an API key.' Under the 'Bootstrap server' heading, there is a text input field containing 'icp-proxy.10.0.10.2.nip.io:32639' with a red circle around the number '1' next to it. Under the 'Certificates' heading, there are two options: 'Java truststore' and 'PEM certificate', each with a download icon and a red circle around the number '2' next to it. To the right, under the 'API key' heading, a success message says 'Successful API key generation!' followed by instructions to make a note of the key. A generated service ID 'rq-inNFNb-ahK-Q4Rha-Kd_Oh1dDMPiyEK' is shown with download and copy icons, also circled with a red '2'. A 'Create a new API key' button is at the bottom.

For reference

- The es-api-key will be on /home/ibmuser/Downloads directory
- 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

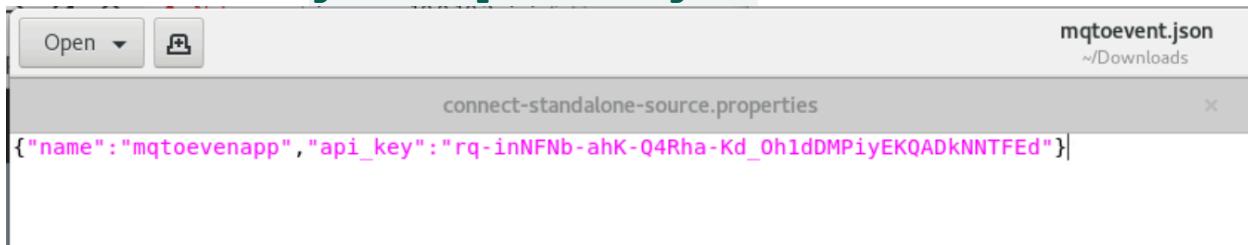
There are two configuration files for each connector. The Kafka connector configuration file contains the properties needed to connect to Kafka. The MQ connector configuration file contains the properties needed for the connector to connect to MQ.

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`



A screenshot of a terminal window titled "mqtoevent.json" located in the "/Downloads" directory. The window shows the command "connect-standalone-source.properties". Inside the terminal, the JSON configuration is displayed:

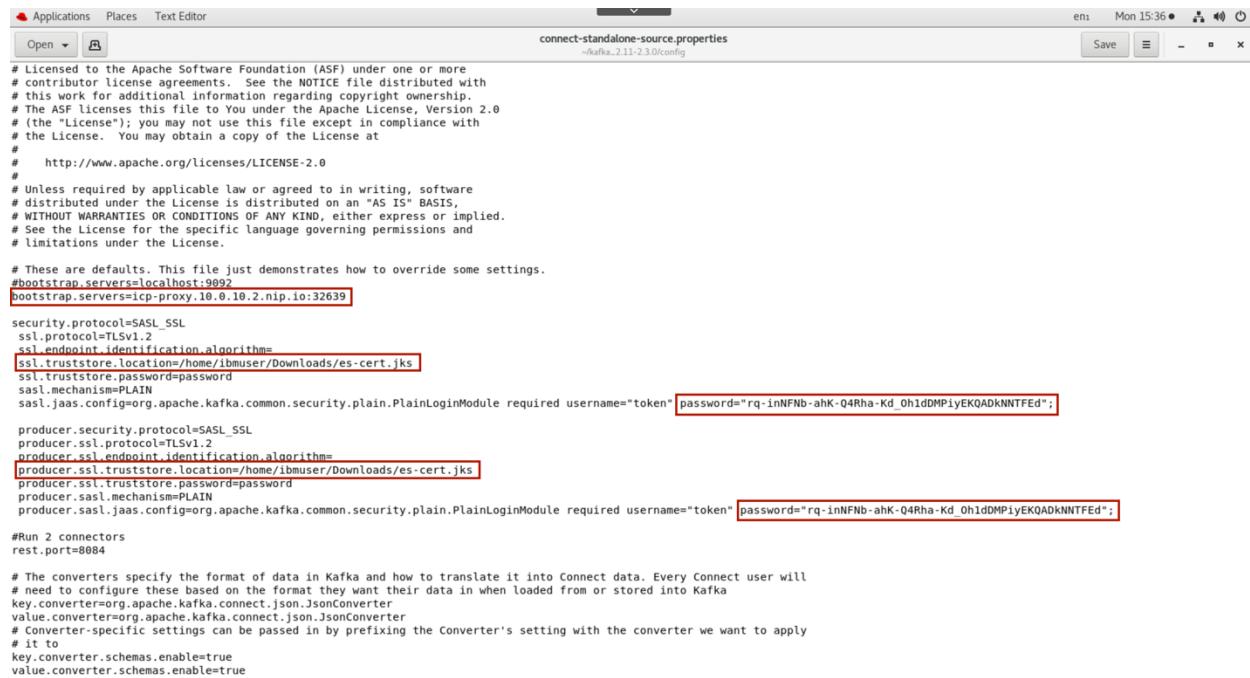
```
{"name": "mqtoevenapp", "api_key": "rq-inNFNb-ahK-Q4Rha-Kd_Oh1dDMPiyEKQADkNNTFEd"}
```

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=**tcp-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).



```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# These are defaults. This file just demonstrates how to override some settings.
#bootstrap.servers=localhost:9092
bootstrap.servers=tcp-proxy.10.0.10.2.nip.io:32639

security.protocol=SASL_SSL
ssl.protocol=TLSv1.2
ssl.endpoint.identification.algorithm=
ssl.truststore.location=/home/ibmuser/Downloads/es-cert.jks
ssl.truststore.password=password
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="token" password="rq-inNFNb-ahK-Q4Rha-Kd_OhldDMPiyEKQADkNNTFEd";

producer.security.protocol=SASL_SSL
producer.ssl.protocol=TLSv1.2
producer.ssl.endpoint.identification.algorithm=
producer.ssl.truststore.location=/home/ibmuser/Downloads/es-cert.jks
producer.ssl.truststore.password=password
producer.sasl.mechanism=PLAIN
producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="token" password="rq-inNFNb-ahK-Q4Rha-Kd_OhldDMPiyEKQADkNNTFEd";

#Run 2 connectors
rest.port=8084

# The converters specify the format of data in Kafka and how to translate it into Connect data. Every Connect user will
# need to configure these based on the format they want their data in when loaded from or stored into Kafka
key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
# Converter-specific settings can be passed in by prefixing the Converter's setting with the converter we want to apply
# it to
key.converter.schemas.enable=true
value.converter.schemas.enable=true
```

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`



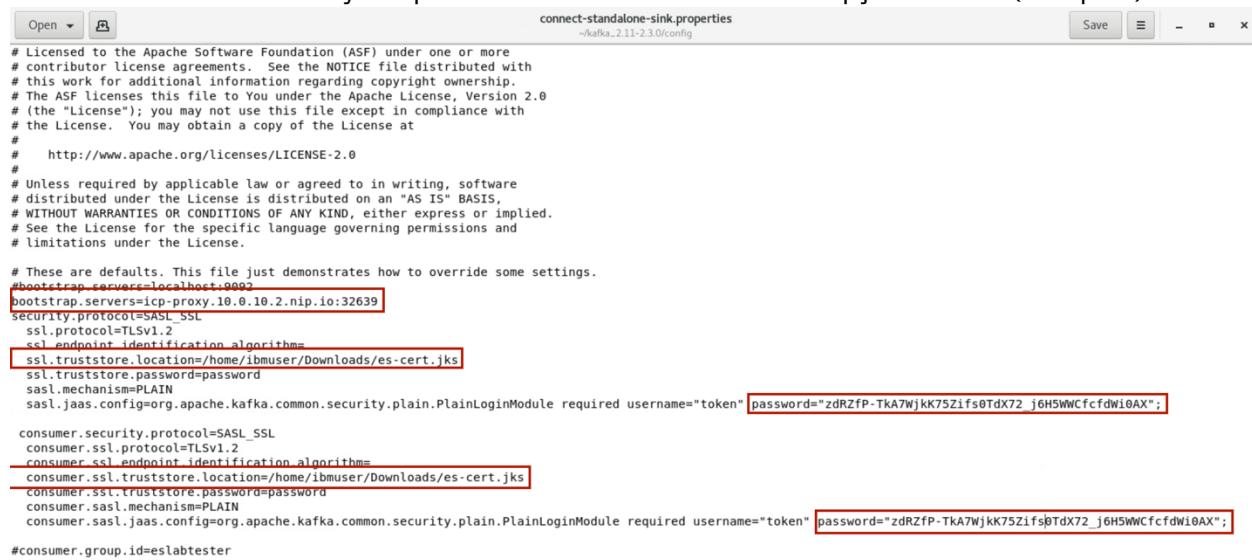
```
{"name": "eventtomq", "api_key": "DFjgYR1lwKLb4ot5XafabNckKhWdmzUl-r90Uw8TQ51nd"}
```

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).



```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to you under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# These are defaults. This file just demonstrates how to override some settings.
#bootstrap.servers=localhost:9092
bootstrap.servers=icp-proxy.10.0.10.2.nip.io:32639
security.protocol=SASL_SSL
ssl.protocol=TLSv1.2
ssl.endpoint.identification.algorithm=
ssl.truststore.location=/home/ibmuser/Downloads/es-cert.jks
ssl.truststore.password=password
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="token" password="zdRZfp-TkA7Wjkk75Z1fs0TdX72_j6H5WWCfcfdWi0AX";

consumer.security.protocol=SASL_SSL
consumer.ssl.protocol=TLSv1.2
consumer.ssl.endpoint.identification.algorithm=
consumer.ssl.truststore.location=/home/ibmuser/Downloads/es-cert.jks
consumer.ssl.truststore.password=password
consumer.sasl.mechanism=PLAIN
consumer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="token" password="zdRZfp-TkA7Wjkk75Z1fs0TdX72_j6H5WWCfcfdWi0AX";

#consumer.group.id=eslabtester
```

- 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**

mq-source.properties

```

# Copyright 2017, 2018, 2019 IBM Corporation
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

name=MQ-SOURCE
connector.class=com.ibm.eventstreams.connect.mqsource.MQSourceConnector

# You can increase this for higher throughput, but message ordering will be lost
tasks.max=1

# The name of the MQ queue manager - required
mq.queue.manager= mq

# The connection mode to connect to MQ - client (default) or bindings - optional
# mq.connection.mode=client
# mq.connection.mode=bindings

# A list of one or more host:port entries for connecting to the queue manager. Entries are separated with a comma - required (unless using bindings or CCDT)
mq.connection.name.list=icp-console.10.0.10.2.nip.io(32340)

# The name of the server-connection channel - required (unless using bindings or CCDT)
mq.channel.name=T0.EVENT

# The name of the source MQ queue - required
mq.queue=T0EVENT

# The name of the target Kafka topic - required
topic=mgtosevent

# The user name for authenticating with the queue manager - optional
# mq.user.name=

# The password for authenticating with the queue manager - optional
# mq.password=

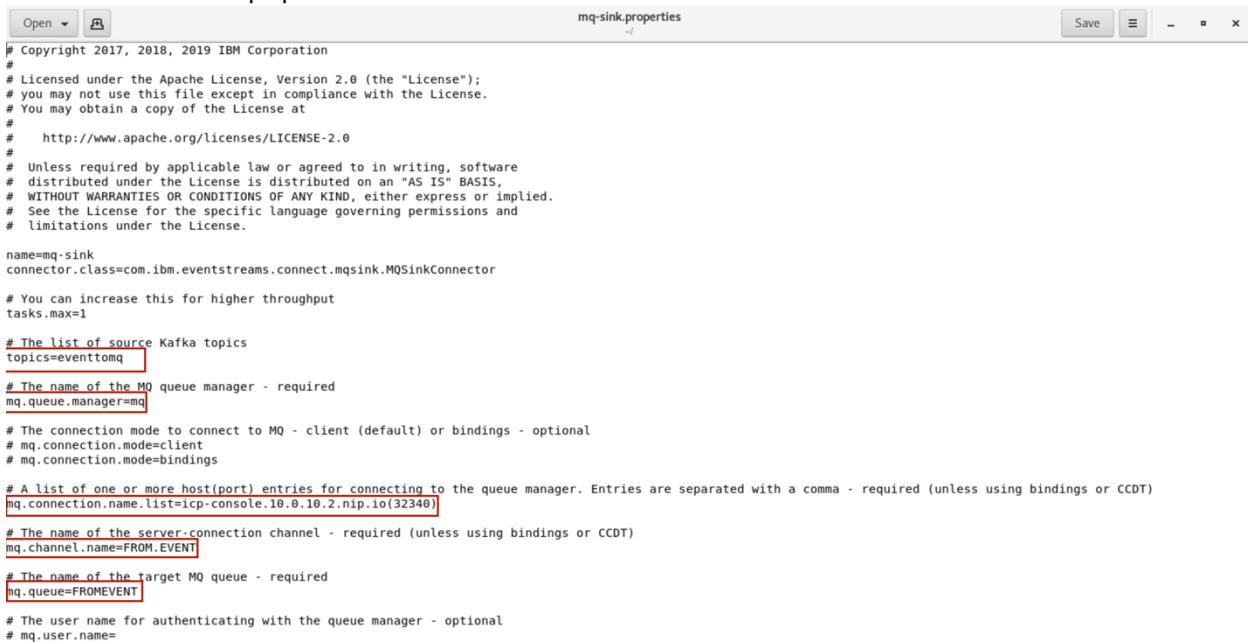
# Alternatively can use a ConfigProvider for externalising secrets (see README.md for more details)
# Variable references are of the form ${provider:[path:]key} where the path is optional,

```

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**



```

# Copyright 2017, 2018, 2019 IBM Corporation
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

name=mc-sink
connector.class=com.ibm.eventstreams.connect.mqsink.MQSinkConnector

# You can increase this for higher throughput
tasks.max=1

# The list of source Kafka topics
topics=eventtomq

# The name of the MQ queue manager - required
mq.queue.manager=mc

# The connection mode to connect to MQ - client (default) or bindings - optional
# mq.connection.mode=client
# mq.connection.mode=bindings

# A list of one or more host(port) entries for connecting to the queue manager. Entries are separated with a comma - required (unless using bindings or CCDT)
mq.connection.name.list=icp-console.10.0.10.2.nip.io(32340)

# The name of the server-connection channel - required (unless using bindings or CCDT)
mq.channel.name=FROM.EVENT

# The name of the target MQ queue - required
mq.queue=FROMEVENT

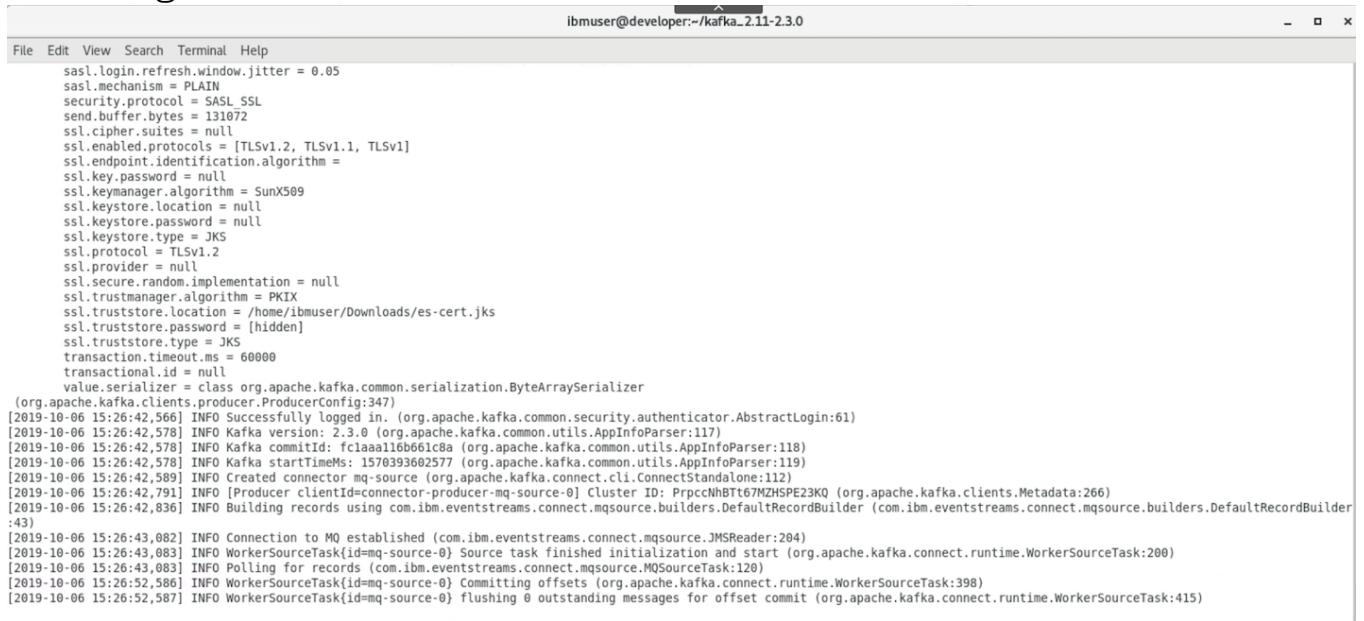
# The user name for authenticating with the queue manager - optional
# mq.user.name=
  
```

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.



A screenshot of a terminal window titled "ibmuser@developer:~/kafka_2.11-2.3.0". The window shows the command `./mqtoevent.sh` being run. The log output displays various Kafka configuration parameters and connector startup messages. Key log entries include:

```
ibmuser@developer:~/kafka_2.11-2.3.0
File Edit View Search Terminal Help
sasl.login.refresh.window.jitter = 0.05
sasl.mechanism = PLAIN
security.protocol = SASL_SSL
send.buffer.bytes = 131072
ssl.cipher.suites = null
ssl.enabled.protocols = [TLSv1.2, TLSv1.1, TLSv1]
ssl.endpoint.identification.algorithm =
ssl.key.password = null
ssl.keystore.algorithm = SunX509
ssl.keystore.location = null
ssl.keystore.password = null
ssl.keystore.type = JKS
ssl.protocol = TLSv1.2
ssl.provider = null
ssl.secure.random.implementation = null
ssl.trustmanager.algorithm = PKIX
ssl.truststore.location = /home/ibmuser/Downloads/es-cert.jks
ssl.truststore.password = [hidden]
ssl.truststore.type = JKS
transaction.timeout.ms = 60000
transactional.id = null
value.serializer = class org.apache.kafka.common.serialization.ByteArraySerializer
(org.apache.kafka.clients.producer.ProducerConfig:347)
[2019-10-06 15:26:42,566] INFO Successfully logged in. (org.apache.kafka.common.security.authenticator.AbstractLogin:61)
[2019-10-06 15:26:42,578] INFO Kafka version: 2.3.0 (org.apache.kafka.common.utils.AppInfoParser:117)
[2019-10-06 15:26:42,578] INFO Kafka commitId: fc1aa11b661c18a (org.apache.kafka.common.utils.AppInfoParser:118)
[2019-10-06 15:26:42,578] INFO Kafka startTimeMs: 1570393602577 (org.apache.kafka.common.utils.AppInfoParser:119)
[2019-10-06 15:26:42,589] INFO Created connector mq-source (org.apache.kafka.connect.cli.ConnectStandalone:112)
[2019-10-06 15:26:42,791] INFO [Producer clientId=connector-producer-mq-source-0] Cluster ID: PrpcchNhBT67MZH5PE23KQ (org.apache.kafka.clients.Metadata:266)
[2019-10-06 15:26:42,836] INFO Building records using com.ibm.eventstreams.connect.mqsource.builders.DefaultRecordBuilder (com.ibm.eventstreams.connect.mqsource.builders.DefaultRecordBuilder:43)
[2019-10-06 15:26:43,082] INFO Connection to MQ established (com.ibm.eventstreams.connect.mqsource.JMSReader:204)
[2019-10-06 15:26:43,083] INFO WorkerSourceTask{id=mq-source-0} Source task finished initialization and start (org.apache.kafka.connect.runtime.WorkerSourceTask:200)
[2019-10-06 15:26:43,083] INFO Polling for records (com.ibm.eventstreams.connect.mqsource.MQSourceTask:120)
[2019-10-06 15:26:52,586] INFO WorkerSourceTask{id=mq-source-0} Committing offsets (org.apache.kafka.connect.runtime.WorkerSourceTask:398)
[2019-10-06 15:26:52,587] INFO WorkerSourceTask{id=mq-source-0} flushing 0 outstanding messages for offset commit (org.apache.kafka.connect.runtime.WorkerSourceTask:415)
```

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

Name	Queue type	Queue depth
AMQ.5D99F7AC23CF4305	Local	0
FROMEVENT	Local	0
TOEVENT	Local	0

Total: 3 Last updated: 2:48:31 PM

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.

Put Message

Enter a message to put on queue 'TOEVENT'

Message: *

This is a message from MQ to Event Streams|

Cancel

Put

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

Name	Replicas	Partitions
eventtomq	3	1
mqtoevent	3	1

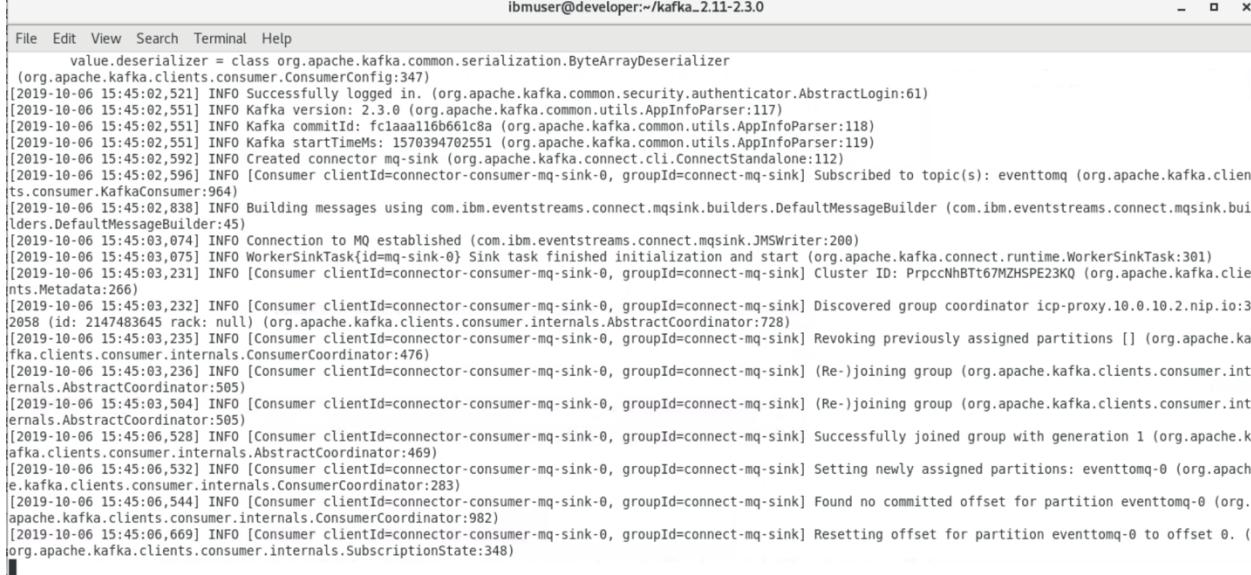
5. Click on **Messages**. You can view the timeframe of data to display (Hours). You can change this to Days, Hours, Minutes and Seconds. Click **on the message**.

Indexed timestamp	Partition	Offset
10/6/2019, 2:59:40 PM	0	0
10/6/2019, 3:31:08 PM	0	1

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

Indexed timestamp	Partition	Offset
10/6/2019, 2:59:40 PM	0	0
10/6/2019, 3:31:08 PM	0	1

- 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.



```
ibmuser@developer:~/kafka_2.11-2.3.0
File Edit View Search Terminal Help
    value.deserializer = class org.apache.kafka.common.serialization.ByteArrayDeserializer
(org.apache.kafka.clients.consumer.ConsumerConfig:347)
[2019-10-06 15:45:02,521] INFO Successfully logged in. (org.apache.kafka.common.security.authenticator.AbstractLogin:61)
[2019-10-06 15:45:02,551] INFO Kafka version: 2.3.0 (org.apache.kafka.common.utils.AppInfoParser:117)
[2019-10-06 15:45:02,551] INFO Kafka commitId: fc1aaal16b661c8a (org.apache.kafka.common.utils.AppInfoParser:118)
[2019-10-06 15:45:02,551] INFO Kafka startTimeMs: 1570394702551 (org.apache.kafka.common.utils.AppInfoParser:119)
[2019-10-06 15:45:02,592] INFO Created connector mq-sink (org.apache.kafka.connect.cli.ConnectStandalone:112)
[2019-10-06 15:45:02,596] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] Subscribed to topic(s): eventtomq (org.apache.kafka.clients.consumer.KafkaConsumer:964)
[2019-10-06 15:45:02,838] INFO Building messages using com.ibm.eventstreams.connect.mqsink.builders.DefaultMessageBuilder (com.ibm.eventstreams.connect.mqsink.builders.DefaultMessageBuilder:45)
[2019-10-06 15:45:03,074] INFO Connection to MQ established (com.ibm.eventstreams.connect.mqsink.JMSWriter:200)
[2019-10-06 15:45:03,075] INFO WorkerSinkTask{id=mq-sink-0} Sink task finished initialization and start (org.apache.kafka.connect.runtime.WorkerSinkTask:301)
[2019-10-06 15:45:03,231] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] Cluster ID: PrpcNhBTt67MZHSPe23KQ (org.apache.kafka.clients.Metadata:266)
[2019-10-06 15:45:03,232] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] Discovered group coordinator icp-proxy.10.0.10.2.nip.io:2058 (id: 2147483645 rack: null) (org.apache.kafka.clients.consumer.internals.AbstractCoordinator:728)
[2019-10-06 15:45:03,235] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] Revoking previously assigned partitions [] (org.apache.kafka.clients.consumer.internals.ConsumerCoordinator:476)
[2019-10-06 15:45:03,236] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] (Re-)joining group (org.apache.kafka.clients.consumer.internals.AbstractCoordinator:505)
[2019-10-06 15:45:03,504] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] (Re-)joining group (org.apache.kafka.clients.consumer.internals.AbstractCoordinator:505)
[2019-10-06 15:45:06,528] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] Successfully joined group with generation 1 (org.apache.kafka.clients.consumer.internals.AbstractCoordinator:469)
[2019-10-06 15:45:06,532] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] Setting newly assigned partitions: eventtomq-0 (org.apache.kafka.clients.consumer.internals.ConsumerCoordinator:283)
[2019-10-06 15:45:06,544] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] Found no committed offset for partition eventtomq-0 (org.apache.kafka.clients.consumer.internals.ConsumerCoordinator:982)
[2019-10-06 15:45:06,669] INFO [Consumer clientId=connector-consumer-mq-sink-0, groupId=connect-mq-sink] Resetting offset for partition eventtomq-0 to offset 0. (org.apache.kafka.clients.consumer.internals.SubscriptionState:348)
```

- 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`

- 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:

- Bootstrap server: **icp-proxy.10.0.10.2.nip.io:32639**
- ssl.truststore.location: **/home/ibmuser/Downloads/es-cert.jks**
- Update the password= It is the same API Key password from the **eventtomq.json** file

```

producer.config
#####
# The parameters in this section must be configured.
#####

# The URL used for bootstrapping knowledge about the rest of the cluster. This address can be found in the Event Streams UI by clicking on either 'Connect to this cluster' or 'Connect to this topic', and viewing the 'Connect a client' tab. The value is provided in the 'Bootstrap server' section.
bootstrap.servers=tcp://10.0.10.2:9092

# The location of the JKS keystore used to securely communicate with your Event Streams instance. This can be downloaded from the Event Streams UI by clicking on either 'Connect to this cluster' or 'Connect to this topic', viewing the 'Connect a client' tab, and downloading the linked Java truststore.
ssl.truststore.location=/home/ibmuser/Downloads/es-cert.jks

# The produced API key for your topic should be added in the password field below. API keys can be set up via the Event Streams UI by clicking on either 'Connect to this cluster' or 'Connect to this topic', opening the 'Connect a client' tab, and running through the 'API key' creation wizard.
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="token" password="9Zj4NdUcdDDK71MMbym670TVsfNhu_a8bSE6GNLrJtC";

#####
# The parameters in this section can be configured, and sensible defaults have been set.
#####

# The default batch size in bytes when batching multiple records sent to a partition.
batch.size=1000
|
# The total bytes of memory the producer can use to buffer records waiting to be sent to the server.
buffer.memory=10000

# The maximum number of unacknowledged requests the client will send on a single connection before blocking.
max.in.flight.requests.per.connection=1000

# The number of acknowledgments the producer requires the leader to have received before considering a request complete.
acks=0

# The password to use for the JKS trustore. By default, the JKS trustore provided will use the password 'password'.
ssl.truststore.password=password

#####
# The parameters in this section should not need to change.
#####

# Configure the producer to use SASL.
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
ssl.protocol=TLSv1.2

```

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:

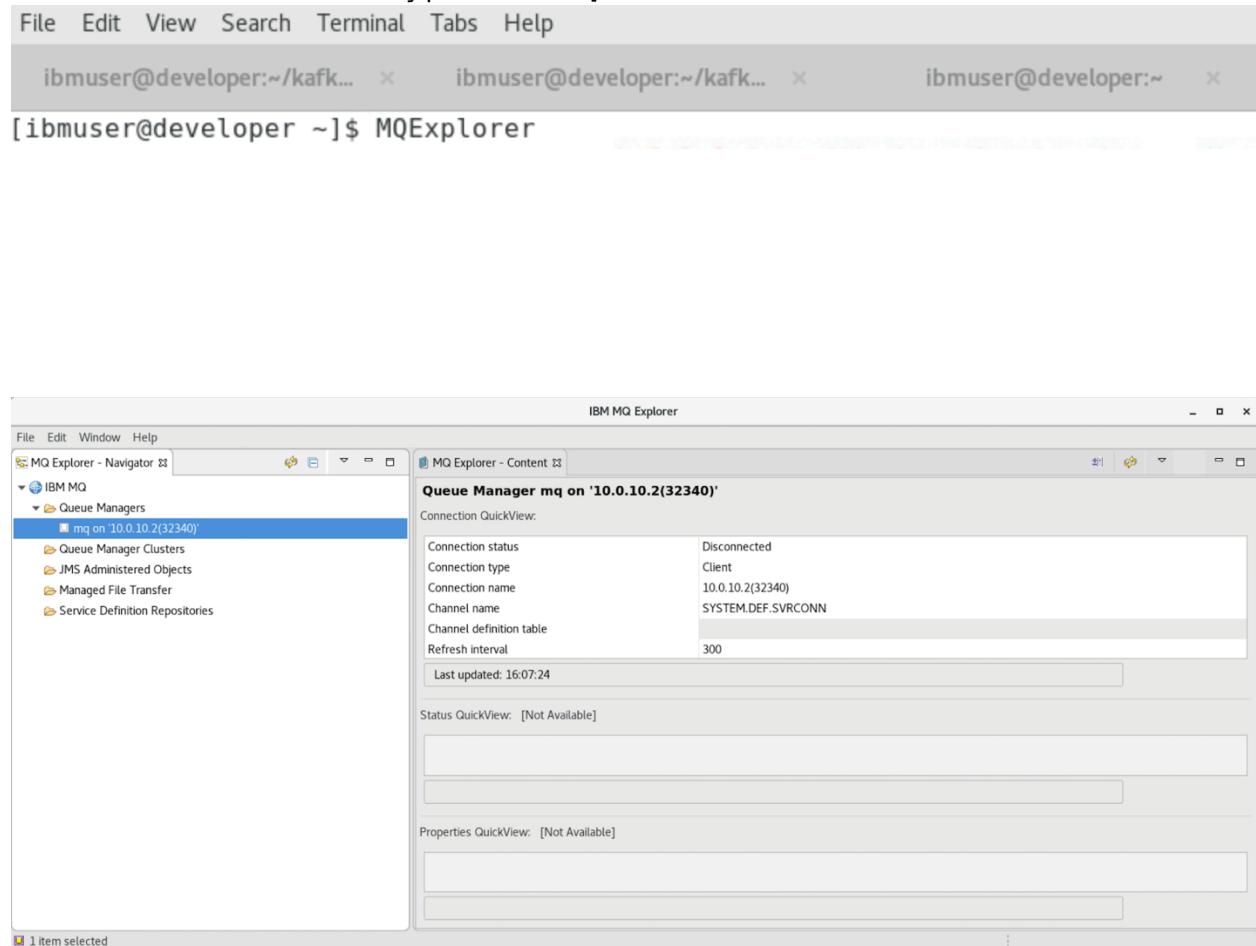
```
ibmuser@developer:~
```

```
File Edit View Search Terminal Help
2019-10-06 15:56:24,666 [producer0] INFO org.apache.kafka.clients.producer.KafkaProducer - [Producer clientId=producer-1] Closing the Kafka producer with timeoutMillis = 9223372036854775807 ms.
2019-10-06 15:56:24,666 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.clients.producer.internals.Sender - [Producer clientId=producer-1] Beginning shutdown of Kafka producer I/O thread, sending remaining records.
2019-10-06 15:56:24,668 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name connections-closed:
2019-10-06 15:56:24,668 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name connections-created:
2019-10-06 15:56:24,668 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name successful-authentication:
2019-10-06 15:56:24,668 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name failed-authentication:
2019-10-06 15:56:24,668 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name bytes-sent-received:
2019-10-06 15:56:24,669 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name bytes-sent:
2019-10-06 15:56:24,669 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name bytes-received:
2019-10-06 15:56:24,669 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name select-time:
2019-10-06 15:56:24,669 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name io-time:
2019-10-06 15:56:24,670 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-1.bytes-sent:
2019-10-06 15:56:24,670 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-1.bytes-received:
2019-10-06 15:56:24,670 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-1.latency:
2019-10-06 15:56:24,670 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-0.bytes-sent:
2019-10-06 15:56:24,670 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-0.bytes-received:
2019-10-06 15:56:24,671 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-0.latency:
2019-10-06 15:56:24,671 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-2.bytes-sent:
2019-10-06 15:56:24,671 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-2.bytes-received:
2019-10-06 15:56:24,671 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-2.latency:
2019-10-06 15:56:24,671 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-1.bytes-sent:
2019-10-06 15:56:24,672 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.common.metrics.Metrics - Removed sensor with name node-1.bytes-received:
2019-10-06 15:56:24,672 [producer0] DEBUG org.apache.kafka.clients.producer.KafkaProducer - [Producer clientId=producer-1] Kafka producer has been closed
60 records sent, 7.966012 records/sec (0.00 MB/sec), 125.25 ms avg latency, 665.00 ms max latency, 104 ms 50th, 344 ms 95th, 665 ms 99th.
[ibmuser@developer ~]$
```

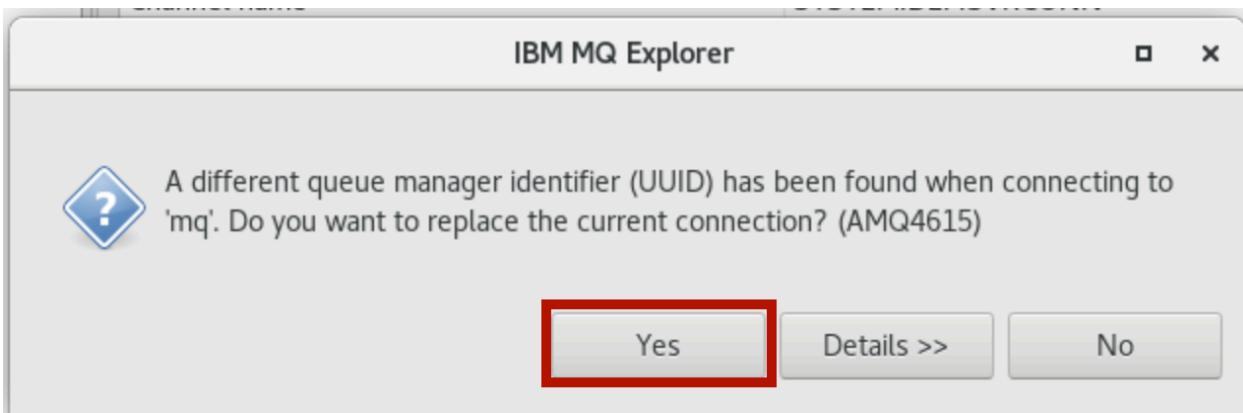
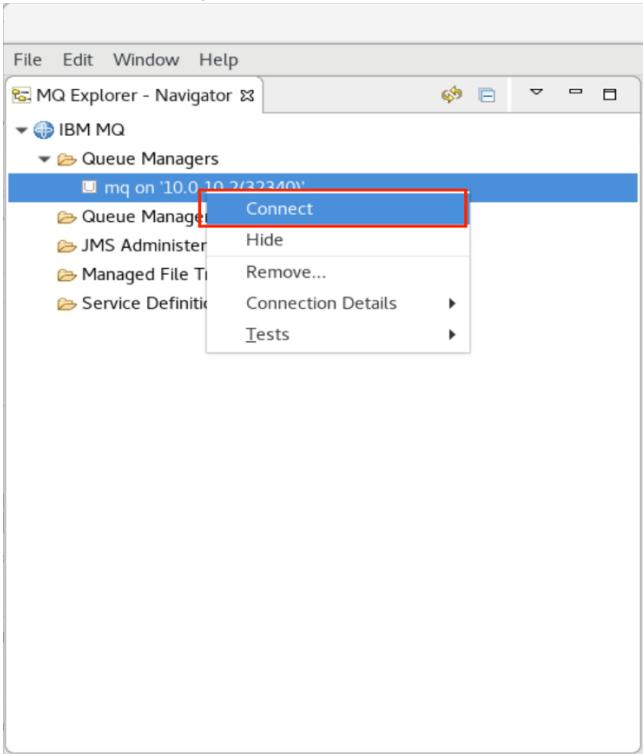
12. Go to the MQ Console. On the **Queues on mq** widget, click the **Refresh** icon and check the **Queue Depth**.

Name	Queue type	Queue depth
AMQ.5D99F7AC23CF4305	Local	13
FROMEVENT	Local	50
TOEVENT	Local	0

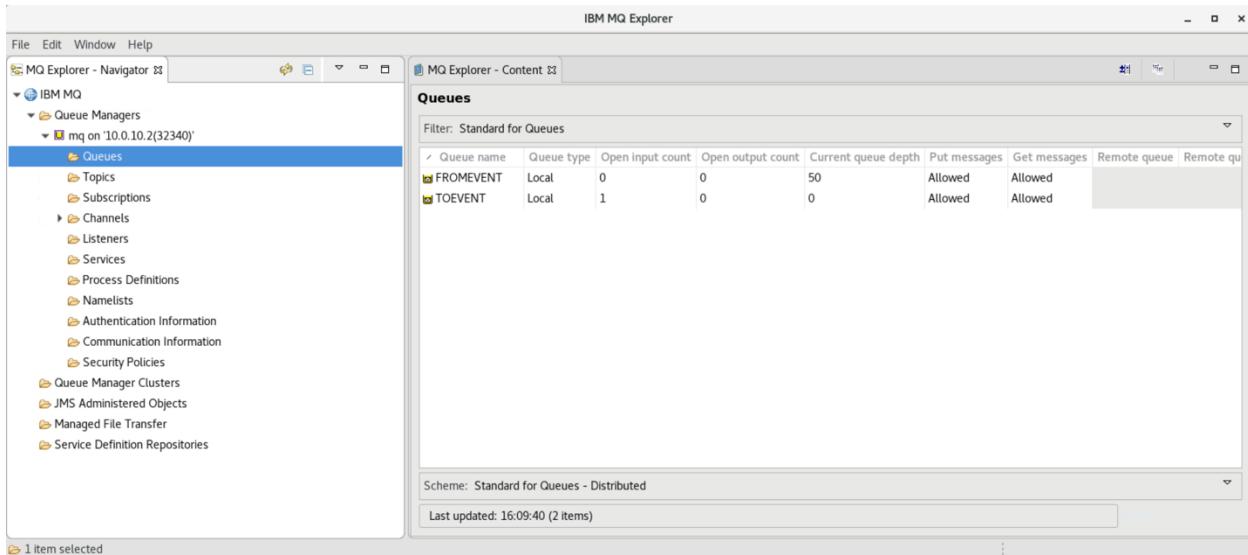
13. You can also use MQ Explorer to **Browse** message. Open a terminal window and type **MQExplorer**.



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.

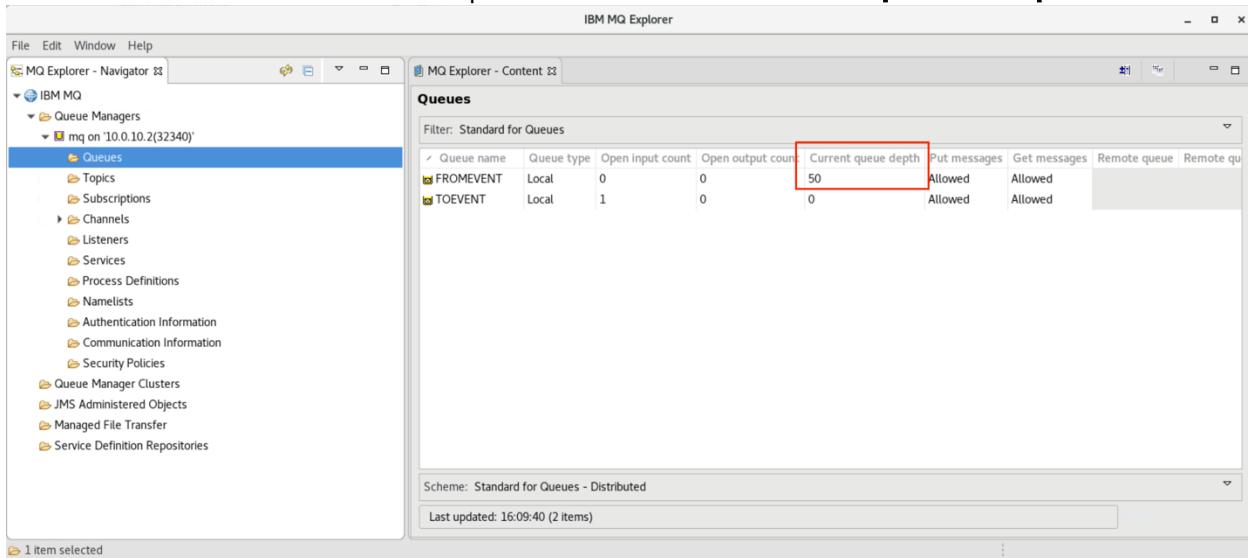


The screenshot shows the IBM MQ Explorer interface. On the left, the Navigator pane displays a tree structure under 'Queue Managers' with 'mq on 10.0.10.2(32340)' selected. Under this, the 'Queues' node is expanded, showing various queue categories like Topics, Subscriptions, Channels, etc. The 'Content' tab on the right is titled 'Queues' and contains a table with two rows:

Queue name	Queue type	Open input count	Open output count	Current queue depth	Put messages	Get messages	Remote queue	Remote qu
FROMEVENT	Local	0	0	50	Allowed	Allowed		
TOEVENT	Local	1	0	0	Allowed	Allowed		

Below the table, a message states 'Scheme: Standard for Queues - Distributed' and 'Last updated: 16:09:40 (2 items)'. A status bar at the bottom indicates '1 item selected'.

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

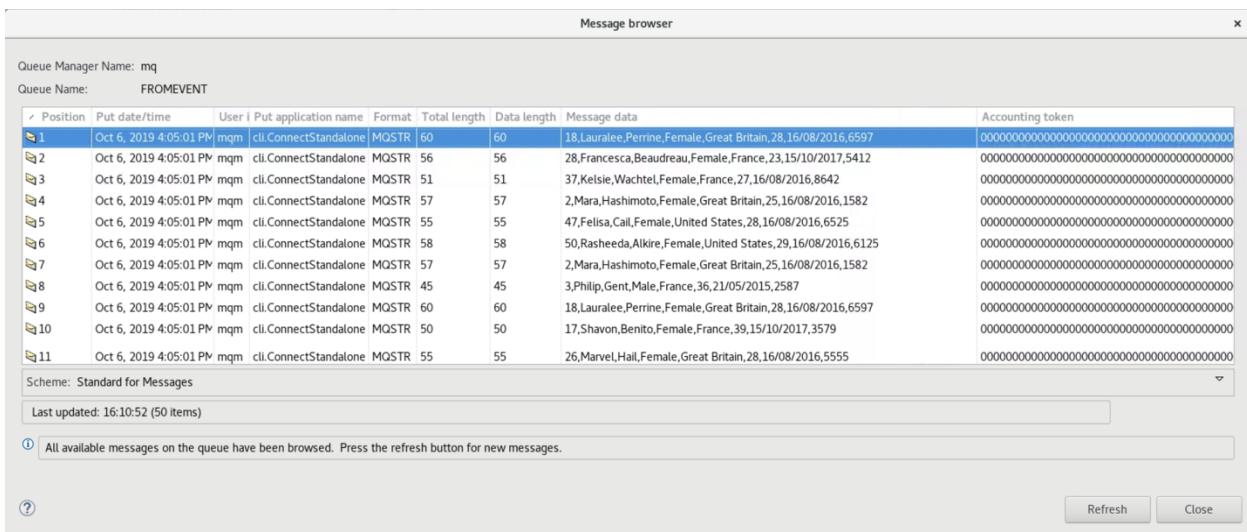
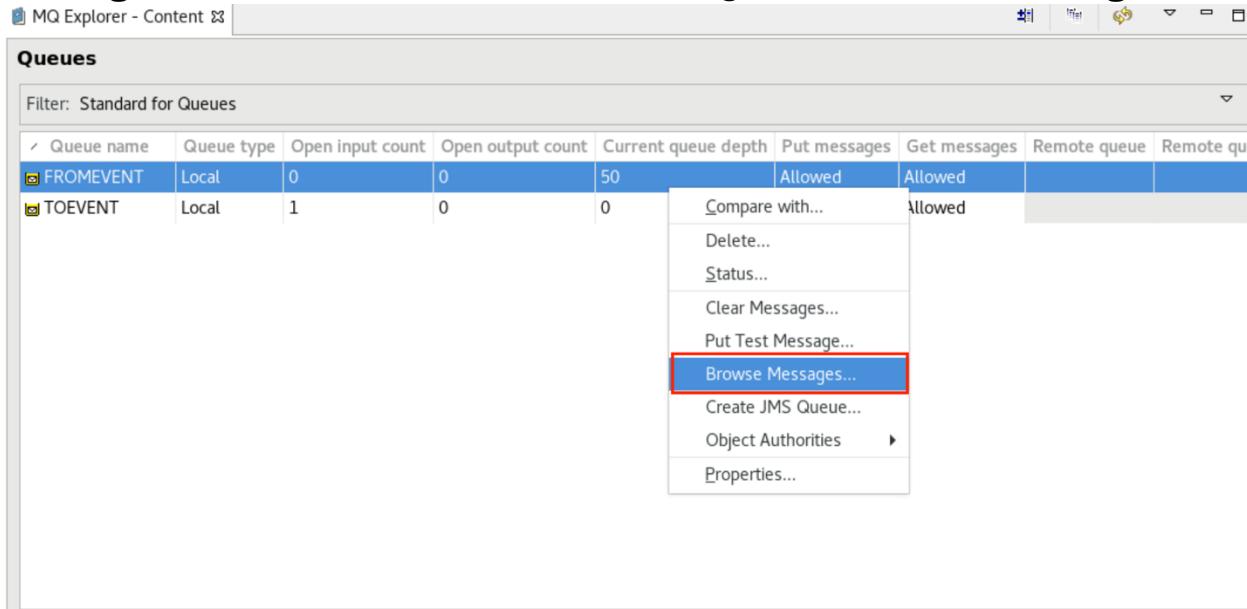


This screenshot is similar to the previous one, showing the IBM MQ Explorer interface. The 'FROMEVENT' queue has been selected in the Navigator pane. The 'Content' tab's 'Queues' table now highlights the 'Current queue depth' column with a red box, specifically pointing to the value '50' for the FROMEVENT row.

Queue name	Queue type	Open input count	Open output count	Current queue depth	Put messages	Get messages	Remote queue	Remote qu
FROMEVENT	Local	0	0	50	Allowed	Allowed		
TOEVENT	Local	1	0	0	Allowed	Allowed		

The status bar at the bottom still shows '1 item selected'.

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.

The screenshot shows the IBM Cloud Pak for Integration interface with the 'Event Streams' tab selected. Under the 'Messages' tab, a table displays the following data:

Indexed timestamp	Partition	Offset
11/18/2019, 3:58:19 PM	0	32
11/18/2019, 3:58:19 PM	0	33
11/18/2019, 3:58:19 PM	0	34
11/18/2019, 3:58:19 PM	0	35
11/18/2019, 3:58:19 PM	0	36
11/18/2019, 3:58:19 PM	0	37
11/18/2019, 3:58:19 PM	0	38
11/18/2019, 3:58:19 PM	0	39
11/18/2019, 3:58:19 PM	0	40
11/18/2019, 3:58:19 PM	0	41
11/18/2019, 3:58:19 PM	0	42
11/18/2019, 3:58:19 PM	0	43
11/18/2019, 3:58:19 PM	0	44
11/18/2019, 3:58:19 PM	0	45
11/18/2019, 3:58:19 PM	0	46
11/18/2019, 3:58:19 PM	0	47
11/18/2019, 3:58:19 PM	0	48
11/18/2019, 3:58:19 PM	0	49

A red box highlights the last row where the offset is 49. The status bar at the bottom right indicates 'System is healthy'.

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>.