

# Lab 4: Use the Kafka Connect source connector for IBM MQ

Duration: 2 hours

In this exercise, you connect IBM Event Streams to IBM MQ by using the Kafka Connect source connector for IBM MQ. You can use this source connector to copy data from MQ into Event Streams or Apache Kafka. The connector copies messages from a source MQ queue to a target Topic.

You can also transfer messages from a Topic to an MQ queue by using a sink connector, which is covered later in this course.

You must complete Labs 1-3 before proceeding with this lab.

## Step 1. Install MQ in IBM Cloud Private

### A. Create a secret to store queue manager credentials

When using the IBM MQ Advanced for Developers certified container, you must create a secret in the target namespace. This secret must contain the admin user password and, optionally, the app user password to use for messaging.

This can be done either by using the ICP Console or the command line interface. The following steps use the console.

The secret is a key:value pair with the value being a base64 encoded password string.

1. Convert the admin user password to base64 by entering the following command in a terminal window:

```
$ echo -n admin | base64
```

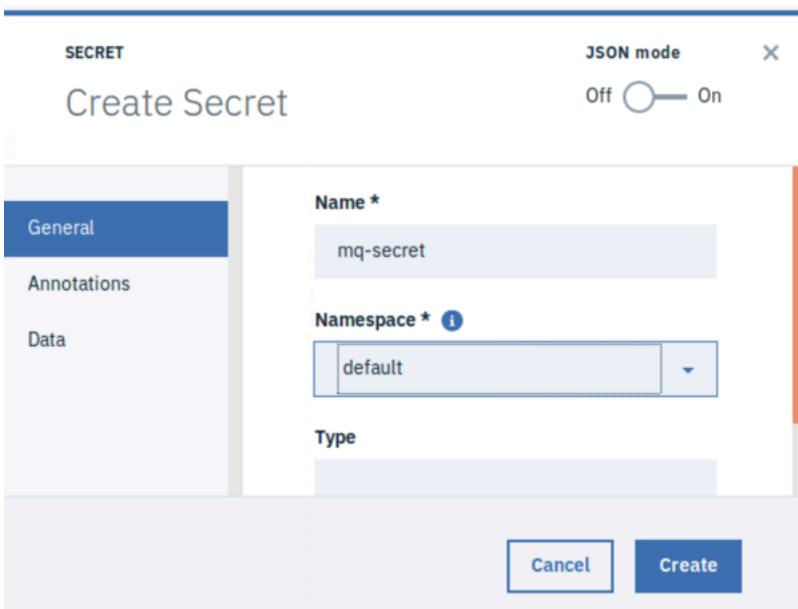
Copy the output to the clipboard or to a file.

2. In the IBM Cloud Private console, select Configuration > Secrets from the menu.

3. Click **Create Secret**.

4. Enter a name for the secret, for example, `mq-secret`.

5. Select **default** as the target namespace.



6. Click the Data tab, and enter a name for the password, for example, `adminpw`, and paste the encoded password into the value field.

**SECRET**

**Create Secret**

**JSON mode**  Off  On

General	Name <small>i</small>	Value
Annotations	adminPassword	YWRtaW4=
<b>Data</b>	Add data <small>+</small>	

**Cancel** **Create**

7. Click **Create**.

## B. Create a Persistent Volume (PV)

This example requires a Persistent Volume (PV) that uses ReadWriteOnce (RWO) mode to store data in IBM Cloud Private. That means that only one node can mount the PV with read/write permissions.

1. On the ICP Master virtual machine image, open Firefox and click the **IBM Cloud Private** bookmark tab, or enter the following address in a browser:  
`https://mycluster.icp:8443/`
2. On the IBM Cloud Private login page, log in with the user ID **admin** and password **admin**.
3. From the hamburger menu, select **Platform > Storage**.
4. Click **Create PersistentVolume**.

**IBM Cloud Private** Create resource Catalog Docs Support

**Storage**

**PersistentVolume** **PersistentVolumeClaim**

**Create PersistentVolume +**

Name	Type	Capacity	Access mode	Reclaim policy	Status	Claim	Created
<a href="#">mgmt-repo-pv</a>	Hostpath	5Gi	RWO	Delete	Bound	kube-system/mgmt-repo-pvc	30 days ago
<a href="#">helm-repo-pv</a>	Hostpath	5Gi	RWO	Delete	Bound	kube-system/helm-repo-pvc	30 days ago
<a href="#">logging-data-node-10.0.0.1</a>	LocalVolume	20Gi	RWO	Retain	Bound	kube-system/data-logging-elk-data-0	30 days ago

In this example, you create a PV by using the host path file system.

5. Complete the **General** section as follows:

- Name: **mqvol**
- Capacity: **2 GB**
- Storage type: **Host path**

PERSISTENTVOLUME      JSON mode  On  Off

### Create PersistentVolume

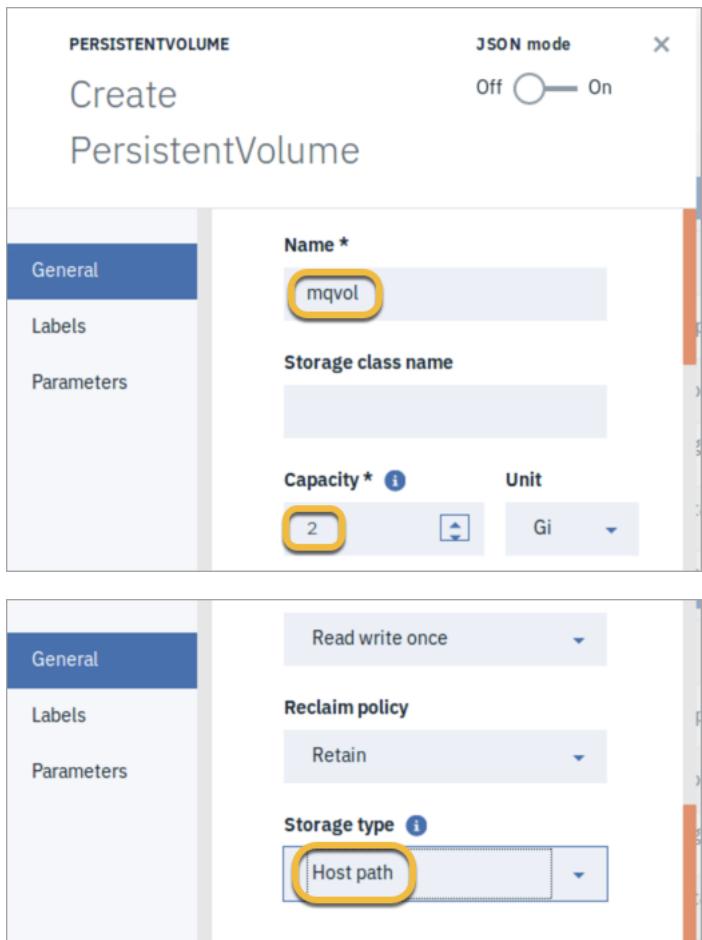
**General**

Name *	mqvol
Storage class name	
Capacity * <small>i</small>	2
Unit	Gi

**General**

Read write once	
Reclaim policy	Retain
Storage type <small>i</small>	Host path



NOTE: Host path is used for the purpose of this lab exercise, but it is not recommended for production environments. Other storage options are available, and you can find more information in the IBM Cloud Private Knowledge Center.

6. Click **Parameters** and complete the form as follows:

- Key: **path**
- Value: **/home/student/icpvols/mqvols/vol1**

7. Click **Create**.

PERSISTENTVOLUME      JSON mode

### Create PersistentVolume

General      Labels      Parameters

**Key \*** path      **Value \*** [icpvols/mqvol/vol1](#)

Add parameter [+](#)

[Cancel](#) [Create](#)

The PV now appears in the list.

Storage

[PersistentVolume](#) [PersistentVolumeClaim](#)

Name	Type	Capacity	Access mode	Reclaim policy	Status	Claim	Created
<a href="#">mqvol</a>	Hostpath	2Gi	RWO	Retain	Available		0 minutes ago
<a href="#">mgmt-repo-pv</a>	Hostpath	5Gi	RWO	Delete	Bound	kube-system/mgmt-repo-pvc	30 days ago
<a href="#">helm-repo-pv</a>	Hostpath	5Gi	RWO	Delete	Bound	kube-system/helm-repo-pvc	30 days ago

- Verify creation of the host path `/home/student/icpvols/mqvol/vol1`.

## C. Install the MQ Helm chart

- In the IBM Cloud Private console, select **Manage > Helm Repositories** from the menu.
- Click **Sync repositories** to make sure that the Helm charts are up to date, and then click **OK** to confirm.
- After synchronization is complete, click **Catalog** to display the list of Helm charts.
- Search for **mq**, and select **ibm-mqadvanced-server-dev**.

## Catalog

The screenshot shows a catalog interface with a search bar at the top containing the text 'mq'. Below the search bar, there is a list of categories on the left: Blockchain, Business Automation, Data, Data Science & Analytics, DevOps, Integration, and IoT. On the right, there are several chart entries. One entry, 'ibm-mqadvanced-server-dev', is highlighted with a yellow border. This entry includes a circular icon with a purple MQ logo, the name 'ibm-mqadvanced-server-dev', the description 'IBM MQ queue manager', and the source 'ibm-charts'. To the right of this are other entries: 'ibm-rabbitmq-dev' (with a red icon), 'rabitmq' (with a blue icon), and 'rabbitmq-ha' (with a blue icon). Each entry has its name, a brief description, and the source it came from.

5. Click **Configure** and complete the form as follows:

- Helm release name: **mymq**
- Target namespace: **default**
- Select the License checkbox

The screenshot shows the 'Configuration' tab of the IBM MQ queue manager setup. At the top, there is a warning box titled 'Pod Security Warning' stating: 'Your ICP cluster is running all namespaces Unrestricted (ibm-anyuid-hostpath-psp) by default. This could pose a security risk.' Below this, the 'Configuration' section is titled 'IBM MQ queue manager. Edit these parameters for configuration.' It contains two main input fields: 'Helm release name \*' with the value 'mymq' and 'Target namespace \*' with the value 'default'. Under 'License \*', there is a checked checkbox labeled 'I have read and agreed to the License agreement'. A small note next to the checkbox says '(1)'.

6. Expand the **All parameters** section, scroll down to **Service**, and select **NodePort** for the Service type.

The screenshot shows the 'All parameters' section expanded. Under the 'Service' heading, there is a description: 'Configuration settings for exposing ports through a service'. Below this, the 'Service type \*' field is set to 'NodePort', which is highlighted with a yellow circle.

7. Under **Security**, select the checkbox to **Initialize volume as root**.

## Security

Configuration settings for security

Service account name \*

default

File system group

Enter value

## Supplemental groups

Enter array in YAML syntax:  
- item1  
- item2

Initialize volume as root \* ⓘ

8. Scroll down to **Queue manager** and enter **QM1** for the Queue manager name, the secret name (**mq-secret**) for the **Password secret name**, and the password name (**adminpw**) for the **Admin password key**.

## Queue manager

Configuration settings for the Queue Manager

### Queue manager name

QM1

Enable multi-instance queue manager \*

### Password secret name \*

mq-secret

### Admin password key \* ⓘ

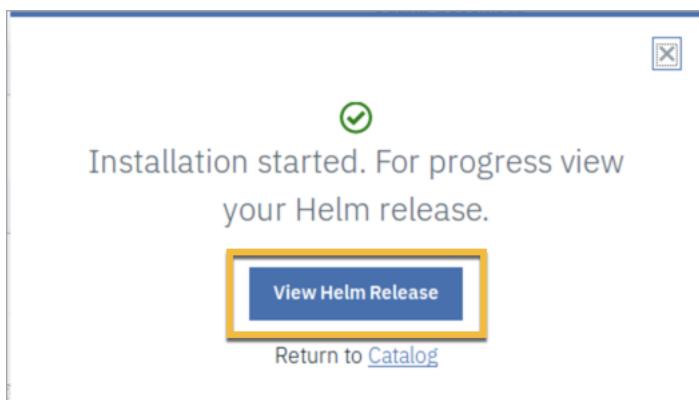
adminpw|

### App password key

Enter value

9. Accept the remaining default values and click **Install**.

10. Click **View Helm release** to view the progress.



The status changes to **Deployed**, and the pod status changes to **Running** when MQ is ready.

The screenshot shows the deployment details for 'mymq'. The 'CURRENT VERSION' is 3.0.1 and the 'AVAILABLE VERSION' is also 3.0.1. The pod 'mymq-ibm-mq-0' is listed as 'Running'. The 'Status' column for the pod table is highlighted with a yellow box.

NAME	READY	STATUS	RESTARTS	AGE
mymq-ibm-mq-0	1/1	Running	0	8m

11. Scroll down to the **Service** section, and you see the NodePort service listed there with its address and port information.

The screenshot shows the 'Service' section with two entries. The second entry, 'mymq-ibm-mq', is highlighted with a yellow oval. It is a NodePort service with a cluster IP of 10.0.0.246 and external ports 9443:31144/TCP, 1414:31730/TCP.

NAME	TYPE	CLUSTER IP	EXTERNAL IP	PORT(S)	AGE
mymq-ibm-mq-metrics	ClusterIP	10.0.0.160	<none>	9157/TCP	5m
mymq-ibm-mq	NodePort	10.0.0.246	<none>	9443:31144/TCP,1414:31730/TCP	5m

12. Under **StatefulSet** click the StatefulSet listed there to view its status.

The screenshot shows the 'StatefulSet' section with one entry, 'mymq-ibm-mq', which is highlighted with a yellow box. The desired and current count is 1, and it is 5 minutes old.

NAME	DESIRED	CURRENT	AGE
mymq-ibm-mq	1	1	5m

13. Scroll down to **Pods** and note the **Status**.

The screenshot shows the IBM Cloud Private console interface. At the top, it displays the service information for 'mymq-ibm-mq' with the following details:

Service	qm
Desired replicas	1
Current replicas	1
Created	9 minutes ago

Below this, under the 'Pods' section, is a table showing the current pod status:

Name	Namespace	Status	Host IP	Pod IP	Ready	Start Time
mymq-ibm-mq-0	default	Running	10.0.0.2	10.1.102.136	0/1	10 minutes ago

At the bottom of the table, there are pagination controls: 'items per page: 20 ▾ | 1-1 of 1 items', '1 of 1 pages', and navigation arrows.

NOTE: You can use the pod name listed here to refer to this resource in the command-line interface (after you configure the `kubectl` client). For example,

```
sudo kubectl describe po mymq-ibm-mq-0
```

You can find more details about using `kubectl` in the [IBM Cloud Private Knowledge Center](#), and in the [Kubernetes documentation](#).

## D. Open the MQ console

1. In the IBM Cloud Private console, go to **Network Access > Services**.
2. Click the **mymq-ibm-mq** service.

The screenshot shows the 'Services' list in the IBM Cloud Private console. A yellow arrow points to the 'Name' column of the table, highlighting the entry 'mymq-ibm-mq' which is enclosed in a yellow box.

Name	Namespace	Created
mymq-ibm-mq	default	19 minutes ago
mymq-ibm-mq-metrics	default	19 minutes ago
eslab-ibm-es-access-controller-svc	es	1 day ago

Links for the MQ console and MQ traffic (listener) display next to **NodePort**. Your ports might differ from those displayed here. Make a note of the MQ listener port for future reference.

3. Click the link for the MQ console.

Services	/	mymq-ibm-mq	/
mymq-ibm-mq			
<u>Overview</u>			
<b>Created</b>	20 minutes ago		
<b>Type</b>	NodePort		
<b>Labels</b>	app=ibm-mq,chart=ibm-mqadvanced-server-dev,heritage=Tiller,release=mymq		
<b>Selector</b>	app=ibm-mq,release=mymq		
<b>Cluster IP</b>	10.0.0.246		
<b>External IP</b>	-		
<b>Load balancer IP</b>	-		
<b>Port</b>	console-https 9443/TCP; qmgr 1414/TCP		
<b>Node port</b>	console-https 31144/TCP qmgr 31730/TCP	←	
<b>Session affinity</b>	None		

NOTE: If you see warnings about using an insecure connection, click **Advanced > Accept the risk and continue**.

4. Log in to the console with user ID **admin** and password **admin**.

The Queue Manager displays a status of **Running**.

### Queue Manager

Name	Status
QM1	Running

Total: 1      Last updated: 8:28:01 AM

### Channels on QM1

Name	Type	Overall channel sta
DEV.ADMIN.SVRCC	Server-connection	Inactive
DEV.APP.SVRCONN	Server-connection	Inactive

Total: 2      Last updated: 8:19:01 AM

### Queues on QM1

Name	Queue type	Queue depth
AMQ.5CDD7CF424!	Local	13

### Topics on QM1

Name	Topic String
DEV.BASE.TOPIC	dev/

Some queues and channels are created for you by default, and the default security settings are sufficient for this lab exercise.

## Step 2. Install the connector

### A. Download connector files

1. Open the Event Streams console.

HINT: To access the Event Streams admin console, in the IBM Cloud Private console, select **Workloads > Helm Releases** from the console menu, click **eslab**, and then click **Launch** in the upper right corner, and select **admin-ui-https**.

2. Click **Toolbox** and scroll down to **Connectors**.

3. Under **Kafka Connect source connector for IBM MQ**, click **Find out more**.

The screenshot shows the 'Connectors' section of the Event Streams admin console. It displays two cards: 'Kafka Connect source connector for IBM MQ' on the left and 'Kafka Connect sink connector for IBM MQ' on the right. Both cards feature a blue icon with a document and a double-headed arrow. Below each icon is the connector's name in bold. Underneath the names is a brief description. At the bottom of each card is a purple 'Find out more' button. A yellow arrow points to the 'Find out more' button on the left card.

4. Click the links to download the **Connector JAR** and **Sample connector properties** file, and click **Save file**.

The screenshot shows a download interface with two rectangular boxes. The top box contains the text 'Download Connector JAR'. The bottom box contains the text 'Download Sample connector properties'.

NOTE: You can also obtain the connector from [GitHub](#) if you want to build it yourself.

5. Copy the files into `/home/student`.

NOTE: The connector runs inside a Java process called a worker. It can run in either standalone or distributed mode. Standalone mode is intended for testing, and temporary connections between systems. Distributed mode is appropriate for production use. In this lab exercise, you use standalone mode.

With a standalone worker, there are two configuration files:

- The worker configuration file contains the properties that are required to connect to Kafka.
- The connector configuration file contains the properties that are used by the connector, so that is where the MQ configuration goes.

The simplest scenario is to run only one connector per standalone worker. A worker sends out a lot of messages, and they are easier to read for just one connector, rather than having to distinguish messages for multiple connectors.

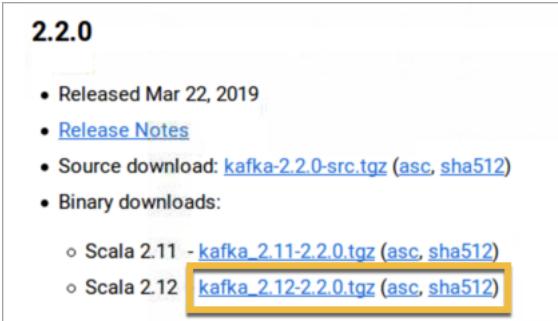
### B. Install Kafka locally

In this part of the exercise, you install a local copy of Kafka before moving on to Event Streams.

1. In a browser, go to the Apache Kafka download site:

```
http://kafka.apache.org/downloads
```

2. Under **Binary downloads**, click the link for the latest supported version of Scala.



Your screen might look different than the image shown here.

If the web site suggests a mirror site for the download, click one of the recommended links, and then click **Save file**.

3. In a command terminal window, run the following command to unpack the compressed file:

```
tar -zxf ~/Downloads/kafka_<version>.tgz
```

NOTE: Be sure to substitute your version for `<version>` in the command. The version used in this course is 2.12-2.2.0, but yours might be different.

The command creates the Kafka root directory, which contains a `bin` directory for the Kafka executable files, and a `config` directory for the configuration files. Feel free to explore the Kafka directory structure.

```
student@master:~/Downloads$ cd kafka_2.12-2.2.0
student@master:~/Downloads/kafka_2.12-2.2.0$ ls
bin config libs LICENSE NOTICE site-docs
student@master:~/Downloads/kafka_2.12-2.2.0$ cd bin
student@master:~/Downloads/kafka_2.12-2.2.0/bin$ ls
connect-distributed.sh      kafka-dump-log.sh          kafka-topics.sh
connect-standalone.sh       kafka-log-dir.sh        kafka-verifiable-consumer.sh
kafka-acls.sh               kafka-mirror-maker.sh    kafka-verifiable-producer.sh
kafka-broker-api-versions.sh kafka-preferred-replica-election.sh trogrodor.sh
kafka-configs.sh            kafka-producer-perf-test.sh windows
kafka-console-consumer.sh   kafka-reassign-partitions.sh zookeeper-security-migration.sh
kafka-console-producer.sh   kafka-replica-verification.sh zookeeper-server-start.sh
kafka-consumer-groups.sh   kafka-run-class.sh        zookeeper-server-stop.sh
kafka-consumer-perf-test.sh kafka-server-start.sh     zookeeper-shell.sh
kafka-delegation-tokens.sh kafka-server-stop.sh
kafka-delete-records.sh    kafka-streams-application-reset.sh
```

4. Change to `/home/student` and open the `mq-source.properties` file in an editor:

```
cd /home/student
gedit mq-source.properties
```

5. Update the properties as follows:

- `mq.queue.manager=QM1`
- `mq.connection.name.list=10.0.0.1(port)`
- `mq.channel.name=DEV.APP.SVRCONN`
- `mq.queue=DEV.QUEUE.1`
- `topic=eslab`

NOTE: For the `port`, use your MQ listener port, which is displayed in the IBM Cloud Private console under **Network Access > Services > mymq-ibm-mq**.

Node port	console-https 30753/TCP gmg 32053/TCP
-----------	--

The number might be different than the ones shown here.

```

# The name of the MQ queue manager - required
mq.queue.manager=QM1

# 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=10.0.0.1(30753)

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

# The name of the source MQ queue - required
mq.queue=DEV.QUEUE.1

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

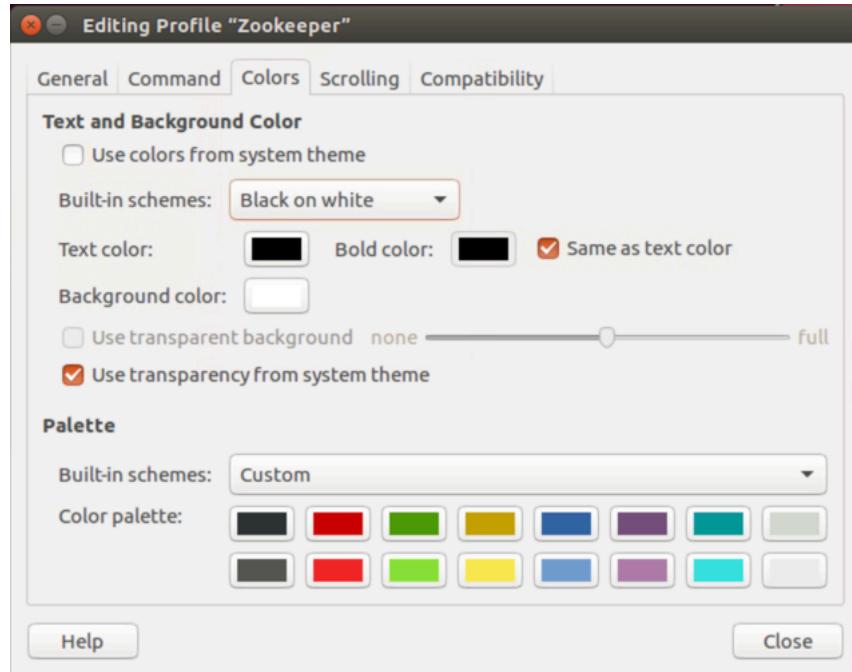
```

- Save and close the file.

## C. Test the connector

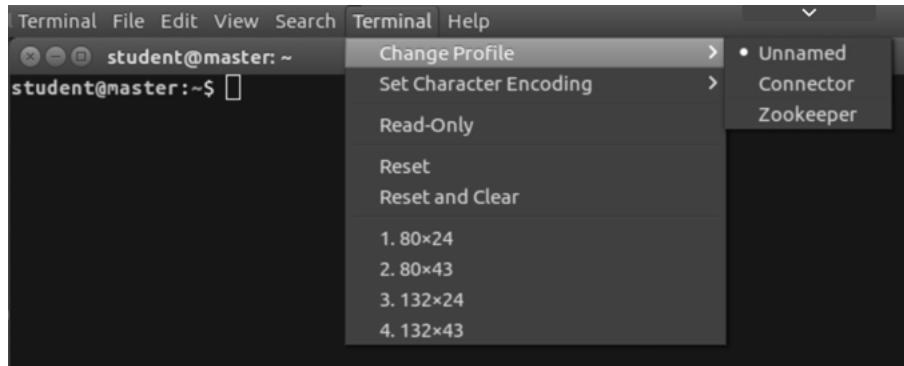
Several components are required to run a minimal Kafka cluster, and it is recommended to run each of them in a separate command terminal window. In this exercise, you open a new command terminal window (right-click **Terminal**, and select **New Terminal**), and change to the Kafka root directory to run the command for each process that is required. You must keep all the windows open. You can minimize a window to get it out of the way by clicking the - (minus sign) in the upper left corner, but do not close the window, or the process will stop.

HINT: You might want to arrange all the windows on the desktop in such a way that you can keep track of them, and tell them apart. You can change the color scheme of each window by creating different terminal profiles. Select **File > New profile** from the Terminal menu, and enter a name for the profile on the **General** tab. Click the **Colors** tab, and choose a different color scheme.



Create as many profiles as you prefer. You might have 5-6 active terminal windows in this exercise.

To switch profiles, select **Terminal > Change profile**, and select the profile.



1. In a new terminal window, change to the Kafka root directory, and start the Zookeeper server:

```
cd /home/student/Downloads/kafka_<version>
bin/zookeeper-server-start.sh config/zookeeper.properties
```

NOTE: Once again, be sure to substitute your version for `<version>` in the command. The version used in this course is 2.12-2.2.0, but yours might be different.

2. In another new terminal window, in the Kafka root directory, run the Kafka server:

```
bin/kafka-server-start.sh config/server.properties
```

Wait until you see the message `INFO [KafkaServer id=0] started` before proceeding with the next step.

3. In another new terminal window, in the Kafka root directory, enter the following command to create a topic:

```
bin/kafka-topics.sh --zookeeper localhost:2181 --create --topic eslab --partitions 1 --replication-factor 1
```

4. Run the following command to verify that the topic was created:

```
bin/kafka-topics.sh --zookeeper localhost:2181 --describe
```

```
student@master:~/Downloads/kafka_2.12-2.2.0$ bin/kafka-topics.sh --zookeeper localhost:2181 --describe
Topic:eslab      PartitionCount:1      ReplicationFactor:1      Configs:
          Topic: eslab      Partition: 0      Leader: 0      Replicas: 0      Isr: 0
```

The details of this single-node configuration follow:

- Kafka bootstrap server: `localhost:9092`
- ZooKeeper server: `localhost:2181`
- Topic name: `eslab`
- Kafka writes data to `/tmp/kafka-logs`
- Zookeeper uses `/tmp/zookeeper`
- Kafka Connect uses `/tmp/connect.offsets`

5. In another new terminal window, in the Kafka root directory, run the following command to start the connector:

```
CLASSPATH=/home/student/kafka-connect-mq-source-1.0.1-jar-with-dependencies.jar bin/connect-standalone.sh config/connect-standalone.properties
```

This process is the Kafka Connect worker. Two messages in the output indicate that the connector is running properly: "Connection to MQ established," and "Polling for records."

```
[2019-05-16 12:31:46,052] INFO Connection to MQ established (com.ibm.eventstreams.connect.mqsource.JMSReader:197)
[2019-05-16 12:31:46,053] INFO WorkerSourceTask{id=mq-source-0} Source task finished initialization and start (org.apache.kafka.connect.runtime.WorkerSourceTask:200)
[2019-05-16 12:31:46,053] INFO Polling for records (com.ibm.eventstreams.connect.mqsource.MQSourceTask:93)
```

6. In another new terminal window, in the Kafka root directory, run the following command to start the consumer:

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic eslab
```

You will not see any message output in this window until the consumer starts to consume messages.

7. In the MQ console, under Queues, select **DEV.QUEUE.1** and click the button to "Put message."

Queues on QM1		
Name	Queue type	Queue depth
AMQ.5CDD7CF424!	Local	0
DEV.DEAD.LETTER.	Local	0
DEV.QUEUE.1	Local	0
DEV.QUEUE.2	Local	0
DEV.QUEUE.3	Local	0

8. Enter a test message, and click **Put**.

Put Message  
Enter a message to put on queue 'DEV.QUEUE.1'

Message:\*

This is a test message

Cancel Put

Now you see the message in the terminal window that is running the consumer.

```
student@master:~/Downloads/kafka_2.12-2.2.0$ bin/kafka-console-consumer.sh --boo
tstrap-server localhost:9092 --topic eslab
This is a test message
```

The local Kafka instance successfully consumed the message that you published from MQ.

9. In the terminal window that is running the consumer, press Ctrl-C.

The process ends, and displays a message indicating the number of messages that were processed.

Now you can stop the Kafka processes.

10. In the terminal window that is running the Kafka Connect worker, press Ctrl-C.

11. In any terminal window that is not otherwise busy, in the Kafka root directory, enter the following commands to stop the Kafka server, and then stop Zookeeper:

```
bin/kafka-server-stop.sh
bin/zookeeper-server-stop.sh
```

12. Close the extra terminal windows.

### Step 3. Test the connector with Event Streams

You can use an existing MQ or Kafka installation, either locally or in the cloud. For performance reasons, it is recommended to run the Kafka Connect worker close to the MQ queue manager to minimize the effect of network latency. So, if you have a queue manager in a datacenter and Kafka in the cloud, it is best to run the Kafka Connect worker in the datacenter.

In this part of the exercise, you use the existing local Kafka cluster, and specify connection details in the Kafka Connect worker configuration file. You need to have the following information on hand:

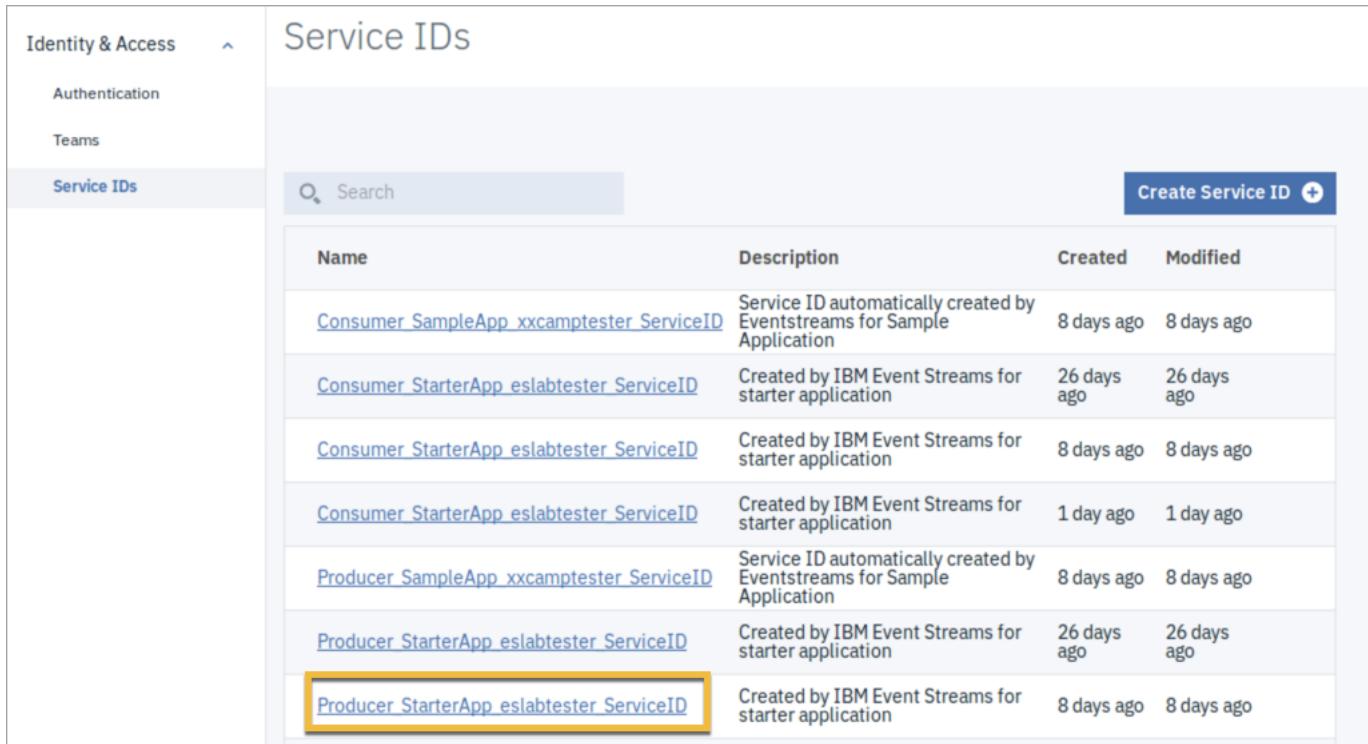
- A list of one or more servers for bootstrapping connections
- Cluster connection requirements (SSL or TLS)
- Authentication credentials, if required

You also run the Kafka Connect worker as you did previously.

## A. Create API keys for the producer and consumer

Create API keys, one each for the consumer and producer. In this example, Kafka Connect uses the producer key, and the console consumer uses the consumer key.

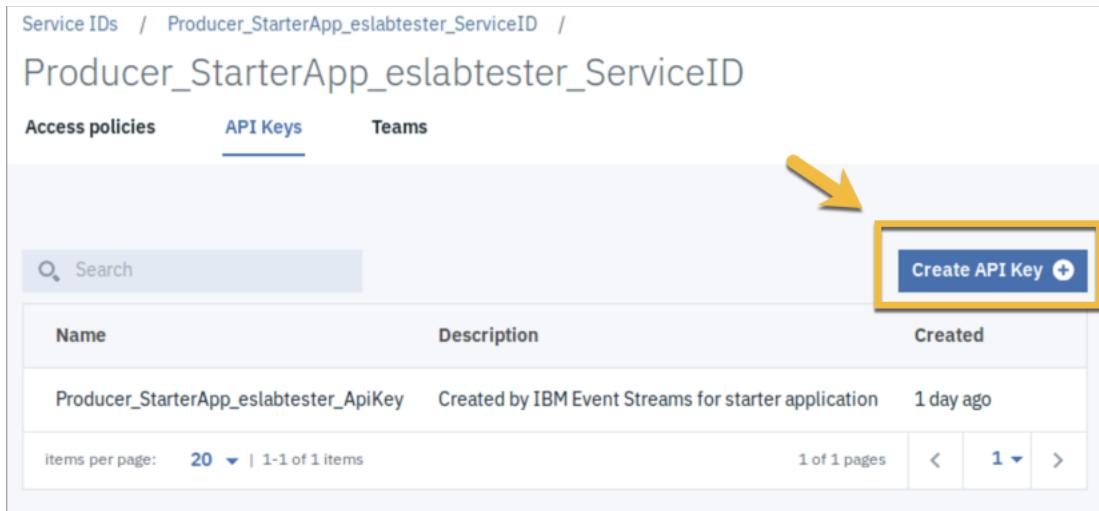
1. In the IBM Cloud Private console menu, select **Manage > Identity & Access**, and click **Service IDs**.
2. Click **Producer\_StarterApp\_eslabtester\_ServiceID**.



Name	Description	Created	Modified
<a href="#">Consumer SampleApp_xcampsteller_ServiceID</a>	Service ID automatically created by Eventstreams for Sample Application	8 days ago	8 days ago
<a href="#">Consumer StarterApp_eslabtester_ServiceID</a>	Created by IBM Event Streams for starter application	26 days ago	26 days ago
<a href="#">Consumer StarterApp_eslabtester_ServiceID</a>	Created by IBM Event Streams for starter application	8 days ago	8 days ago
<a href="#">Consumer StarterApp_eslabtester_ServiceID</a>	Created by IBM Event Streams for starter application	1 day ago	1 day ago
<a href="#">Producer SampleApp_xcampsteller_ServiceID</a>	Service ID automatically created by Eventstreams for Sample Application	8 days ago	8 days ago
<a href="#">Producer_StarterApp_eslabtester_ServiceID</a>	Created by IBM Event Streams for starter application	26 days ago	26 days ago
<a href="#">Producer_StarterApp_eslabtester_ServiceID</a>	Created by IBM Event Streams for starter application	8 days ago	8 days ago

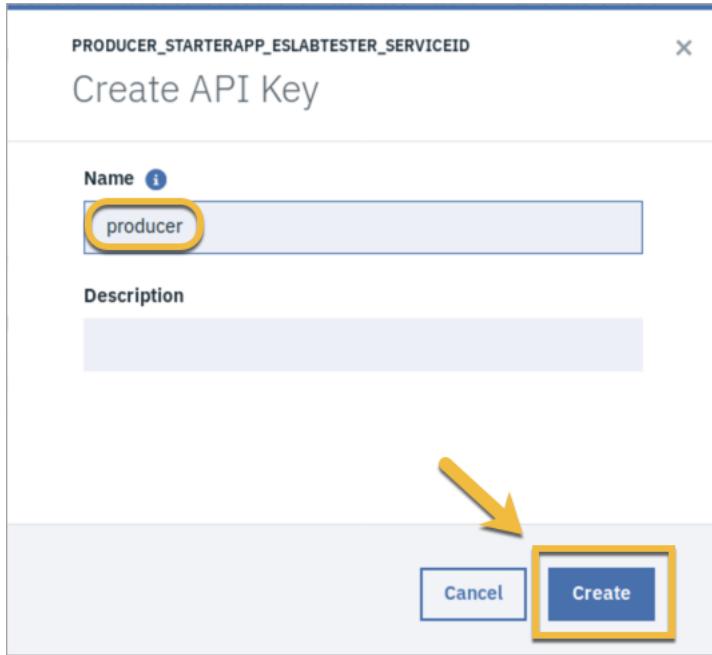
If there are multiple instances of the service ID listed, click the latest one.

3. Click **API Keys**, then **Create API Key**.

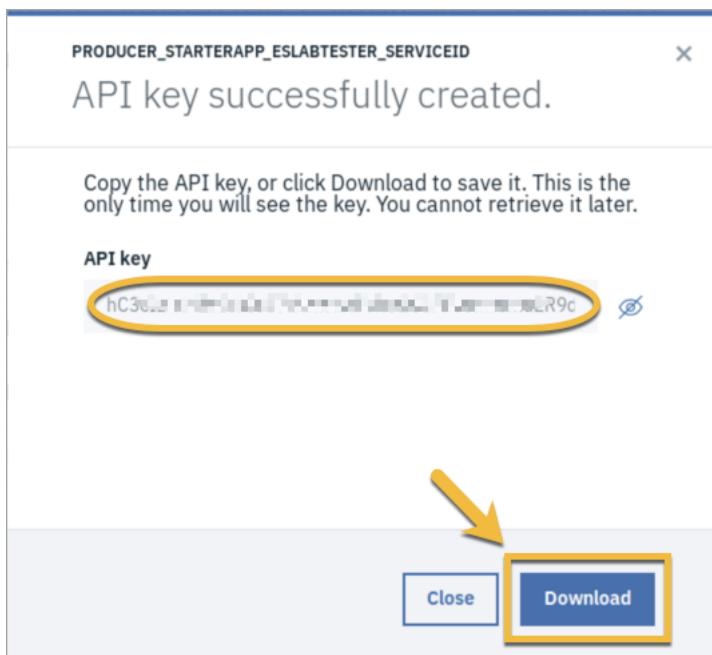


Name	Description	Created
Producer_StarterApp_eslabtester_ApiKey	Created by IBM Event Streams for starter application	1 day ago

4. Enter **producer** for the name, and click **Create**.



5. Click **Download**, and then **Save file**.



The file is named **apikey.json**, by default.

6. In a command terminal, run the following commands to change the name to `producer.json`:

```
cd Downloads  
mv apikey.json producer.json
```

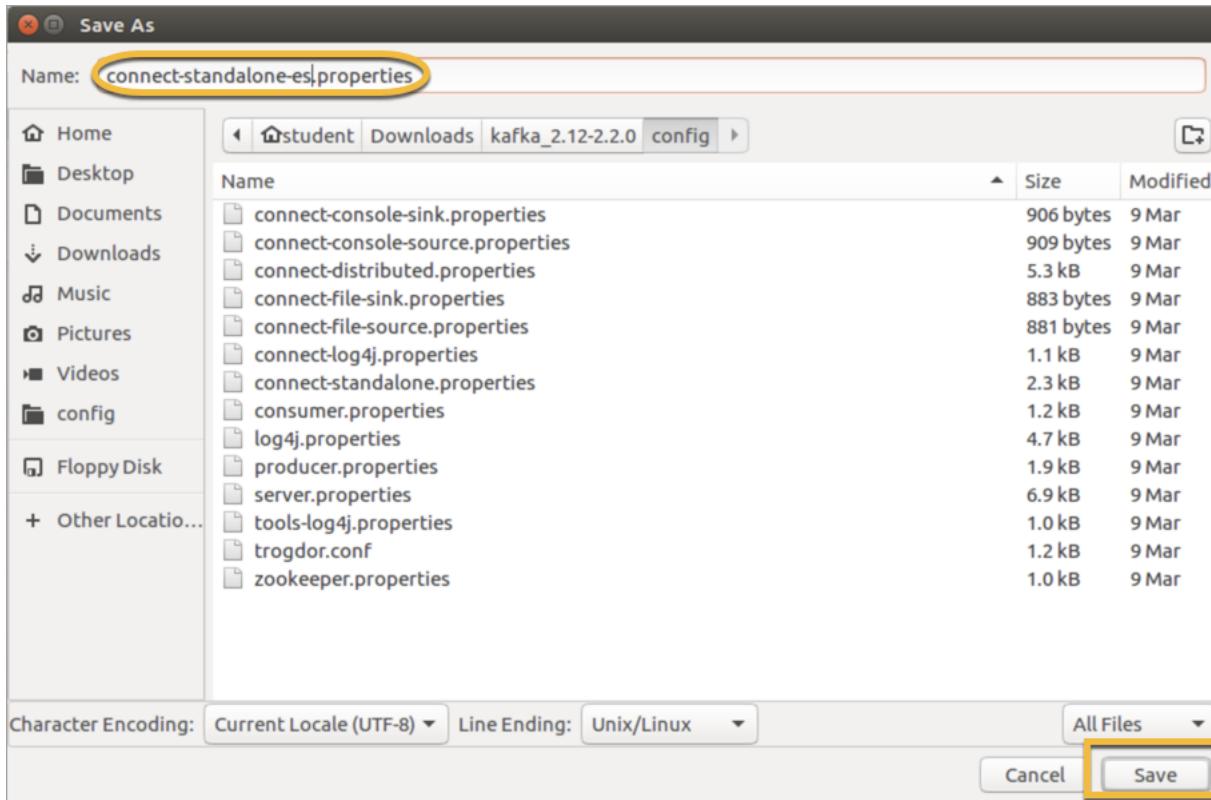
7. Repeat the above steps to create an API key for **Consumer\_StartApp\_eslabtester\_ServiceID**. Name the key **consumer**, and rename the key file `consumer.json`. If there are multiple instances of the service ID listed, click the latest one.

## B. Update the configuration file

1. In the Kafka root `config` directory, make a copy of the `connect-standalone.properties` file, and name it `connect-standalone-es.properties`.

```
cd /home/student/Downloads/kafka_2.12-2.2.0/config  
gedit connect-standalone.properties
```

2. In the text editor menu, select **File > Save As**, edit the name of the file, and click **Save**.



3. Edit the `connect-standalone-es.properties` file. Add the following stanzas:

```
security.protocol=SASL_SSL  
ssl.protocol=TLSv1.2  
ssl.endpoint.identification.algorithm=  
ssl.truststore.location=  
ssl.truststore.password=password  
sasl.mechanism=PLAIN  
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="token" password="";
```

```
producer.ssl.protocol=TLSv1.2  
producer.ssl.endpoint.identification.algorithm=  
producer.ssl.truststore.location=  
producer.ssl.truststore.password=password  
producer.sasl.mechanism=PLAIN  
producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="token" password="";
```

Update values as follows:

- `bootstrap.servers` : the Event Streams bootstrap server (IP address and port number)
- `ssl.truststore.location` : the location of the truststore JKS file that you downloaded in a previous exercise ( `/home/student/Downloads/es-cert.jks` )
- `ssl.truststore.password` : `password`
- `sasl.jaas.config password` : the producer API key that you just created
- `producer.ssl.truststore.location` : the location of the truststore JKS file that you downloaded in a previous exercise ( `/home/student/Downloads/es-cert.jks` )
- `producer.ssl.truststore.password` : `password`
- `producer.sasl.jaas.config password` : the producer API key that you just created

HINT: To find the Event Streams bootstrap server address, in the Event Streams console, click the **Topics** tab, and then click **eslab**. Click **Connect to this topic**, and copy the **Bootstrap server** address and port number. In this example, it is `10.0.0.5:32307`. Be sure to substitute this value with your own bootstrap server address and port.

Copy the API key from the `producer.json` file, and paste it between the double quotes for the `sasl.jaas.config` and `producer.sasl.jaas.config` passwords.

The file should look similar to this:

```
# 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=10.0.0.5:32307

security.protocol=SASL_SSL
ssl.protocol=TLSv1.2
ssl.endpoint.identification.algorithm=
ssl.truststore.location=/home/student/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="g455X...[REDACTED]...;"

producer.security.protocol=SASL_SSL
producer.ssl.protocol=TLSv1.2
producer.ssl.endpoint.identification.algorithm=
producer.ssl.truststore.location=/home/student/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="g455X...[REDACTED]...;"

# The converters specify the format of data in Kafka and how to translate it into Connect data. Every
Connect user will
```

# Producer API key

4. Save and close the file. Note that the MQ properties do not need to be updated in this case.

#### **C. Test the connector by using the Starter application**

1. In a command terminal, in the Kafka root directory, enter the following command to start the connector:

```
CLASSPATH=/home/student/kafka-connect-mq-source-1.0.1-jar-with-dependencies.jar bin/connect-standalone.sh config/connect-standalone-es.properties
```

Look for the two messages in the output that indicate the connector is running properly: "Connected" and "Syncing".



3. Run the eslabtester starter application again (from a previous lab exercise). If it is not already open, in a new browser tab, enter the following URL:

<http://localhost:9080/elabtester>

HINT: If you need to run the starter application again, in a command terminal, change to the `/home/student/Downloads` directory and enter the following commands:

```
cd /home/student/Downloads  
export _JAVA_OPTIONS=-Djdk.net.URLClassPath.disableClassPathURLCheck=true  
mvn install liberty:run-server
```

Wait until you see the message, "The server defaultServer is ready to run a smarter planet" before you proceed to the next step.

You might see some old messages left over from when you ran the application before. Click the most recent message to see more details.

A screenshot of the Kafka console consumer interface. It shows a table with two columns: 'Partition' and 'Offset'. The first partition has an offset of 1. Below the table, there's a section for 'Message size' (24 B), 'Kafka timestamp' (5/21/2019, 8:06:11 AM), and 'Key'. A blue button labeled 'Raw payload' is visible. The message payload is displayed in a box: 'This message is from MQ.' A yellow box highlights this payload text. A small icon of a person is next to it.

#### D. Test the connector by using a console consumer.

1. Make a copy of the `connect-standalone-es.properties` file, and save it as `mqlab.properties`.
2. In two places in the `mqlab.properties` file, where you have the producer API key, change it to the consumer API key (copy the key from `consumer.json` ).
3. Change the `producer` properties to `consumer` properties.

```
# These are defaults. This file just demonstrates how to override some settings.
bootstrap.servers=10.0.0.5:32307

security.protocol=SASL_SSL
ssl.protocol=TLSv1.2
ssl.endpoint.identification.algorithm=
ssl.truststore.location=/home/student/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="D_EUJ...";
```

consumer.security.protocol=SASL\_SSL  
consumer.ssl.protocol=TLSv1.2  
consumer.ssl.endpoint.identification.algorithm=  
consumer.ssl.truststore.location=/home/student/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="D\_EUJ...";

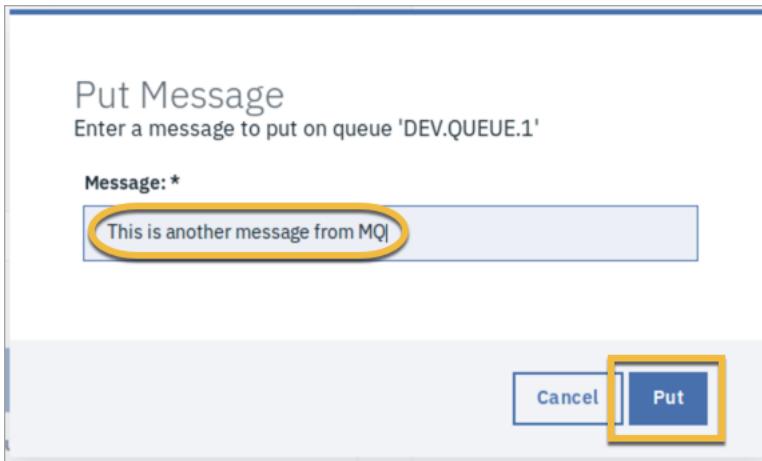
Consumer API key

4. Save and close the file.
5. In the Kafka root directory, run the command for the console consumer:

```
bin/kafka-console-consumer.sh --bootstrap-server 10.0.0.5:32307 --consumer.config config/mqlab.properties --topic eslab --group eslabetest
```

This example assumes that you saved `mqlab.properties` in the Kafka root `config` directory. If you saved the file somewhere else, specify the path to the file as appropriate. Also, be sure to use your bootstrap server address and port in place of the one that is used here.

6. In the MQ console, put another new message in the queue.



In the terminal window where the consumer is running, you see the message.

```
[2019-05-21 09:41:54,040] WARN The configuration 'key.converter.schemas.enable' was supplied but isn't a known config. (org.apache.kafka.clients.consumer.ConsumerConfig)
[2019-05-21 09:41:54,040] WARN The configuration 'value.converter.schemas.enable' was supplied but isn't a known config. (org.apache.kafka.clients.consumer.ConsumerConfig)
[2019-05-21 09:41:54,040] WARN The configuration 'consumer.ssl.protocol' was supplied but isn't a known config. (org.apache.kafka.clients.consumer.ConsumerConfig)
This is another message from MQ
```

7. Press Ctrl-C to stop the console consumer.
8. You can also stop the Kafka Connect worker (press Ctrl-C in its terminal window). You can leave the eslabtester application running because you use it again in the next exercise.

## End of exercise

June 2019 Edition

### Notices

This information was developed for products and services offered in the US. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to: IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 United States of America INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you. This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice. Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk. IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental. **Trademarks** IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml). © Copyright International Business Machines Corporation 2019. This document may not be reproduced in whole or in part without the prior written permission of IBM. US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

### Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies: IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide: IBM Cloud™ z/OS® Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates. VMware is a registered trademark or trademark of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions. Other product and service names might be trademarks of IBM or other companies.-->

