Lab 5: Use Kafka Connect sink connector for IBM MQ

Duration: 1 hour

In this exercise, you connect IBM Event Streams to IBM MQ by using the Kafka Connect sink connector for IBM MQ. You can use this connector to transfer messages from an Event Streams or Kafka Topic to an MQ queue.

In the previous lab exercise, you installed MQ and configured the Kafka Connect source connector, which takes messages from an MQ queue and transfers them to a Topic in Event Streams. In this exercise, you use the same MQ queue manager that you configured previously, and test the sink connector by using the eslabtester application that you used before.

You must complete Labs 1-4 before proceeding with this lab.

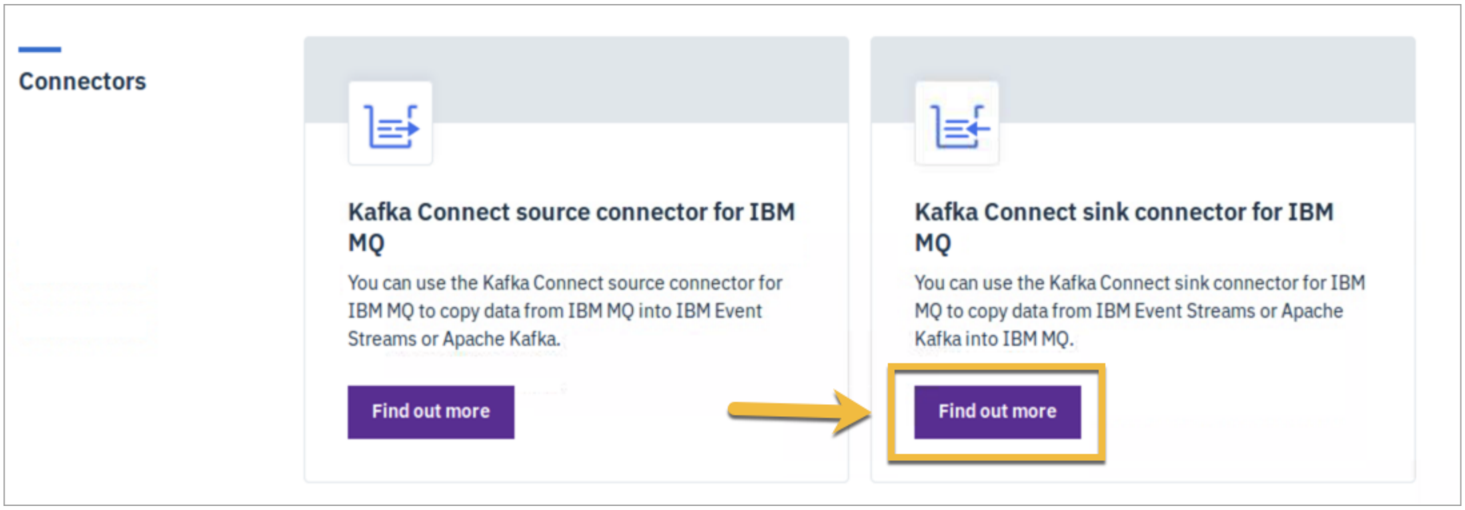
Step 1. Install the connector

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 sink connector for IBM MQ**, click **Find out more**.



Connectors

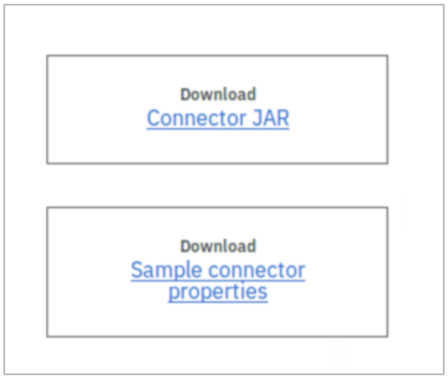
Kafka Connect source connector for IBM MQ

You can use the Kafka Connect source connector for IBM MQ to copy data from IBM MQ into IBM Event Streams or Apache Kafka.

Kafka Connect sink connector for IBM MQ

You can use the Kafka Connect sink connector for IBM MQ to copy data from IBM Event Streams or Apache Kafka into IBM MQ.

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



   NOTE: You can also obtain the connector from GitHub if you want to build it yourself.

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

Step 2. Configure the connector

REMINDER: There are two configuration files: the worker configuration file contains the properties that are required to connect to Kafka, and the connector configuration file contains the properties that are used by the connector, so that is where the MQ configuration goes.

As you did in a previous exercise, you must create a new API key for the consumer application.

A. Update sink connector properties

1. In a command terminal window, change to `/home/student` and open the `mq-sink.properties` file in an editor:

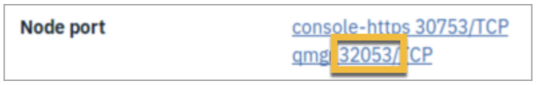
```
cd /home/student
gedit mq-sink.properties
```

2. Update the properties as follows:

   - `topic=eslab`
   - `mq.queue.manager=QM1`
   - `mq.connection.name.list=10.0.0.1(port)`
   - `mq.channel.name=DEV.APP.SVRCONN`
   - `mq.queue=DEV.QUEUE.2`

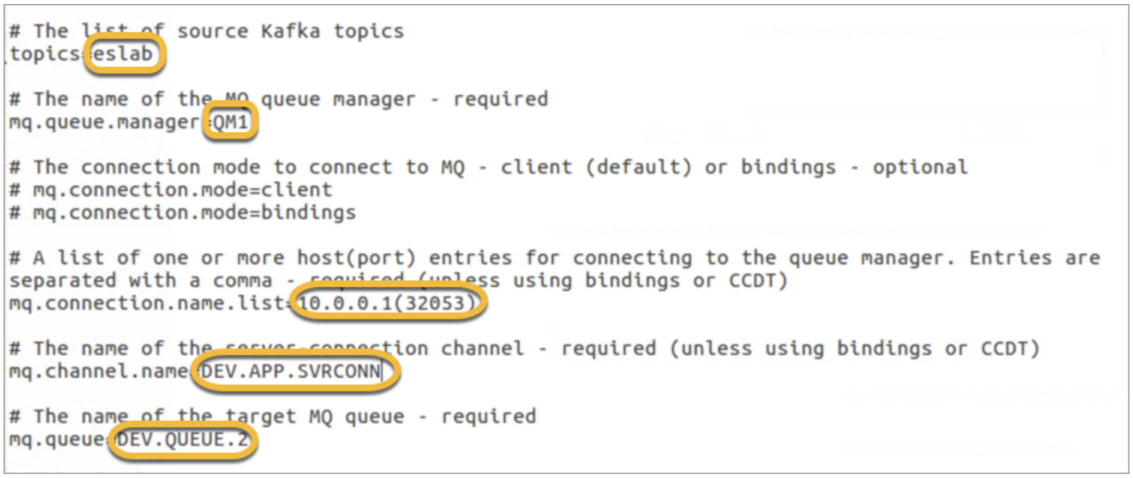
   In this case, the target queue is DEV.QUEUE.2.

   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 qmgr 32053/TCP

   The number might be different than the one shown here.

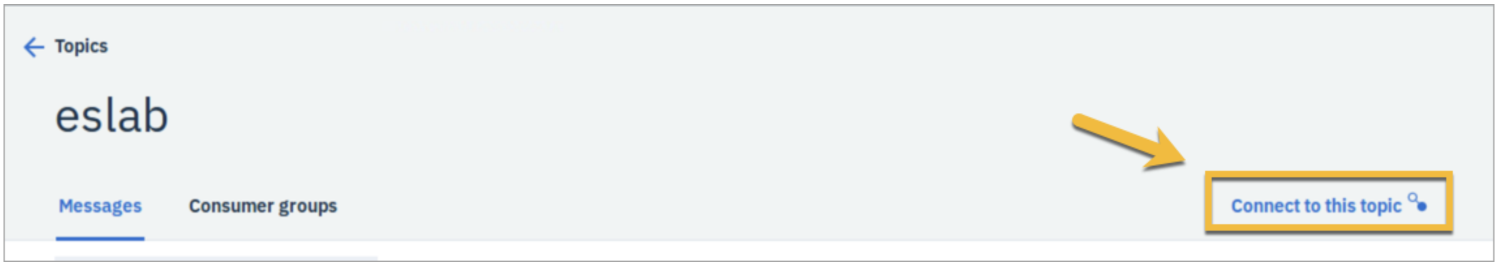


```
# The list of source Kafka topics
topics=eslab

# 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(32053)

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

# The name of the target MQ queue - required
mq.queue=DEV.QUEUE.2
```

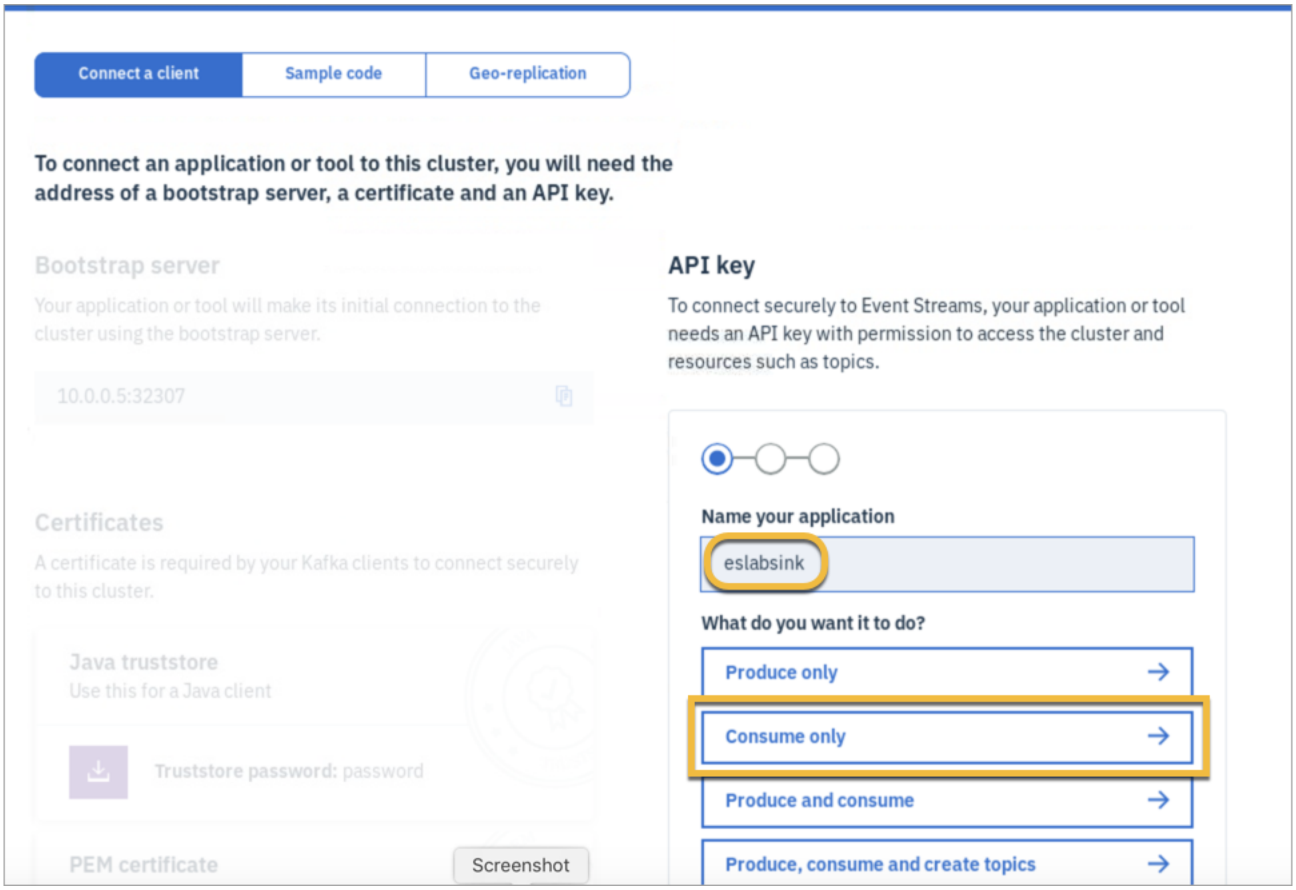
3. Save and close the file.

B. Create an API key for the consumer application

1. In the Event Streams console, select **Topics > eslab > Connect to this topic**.



2. Under **API key**, enter a name in **Name your application**, for example, **eslabsink**, and click **Consume only**.



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.

10.0.0.5:32307

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

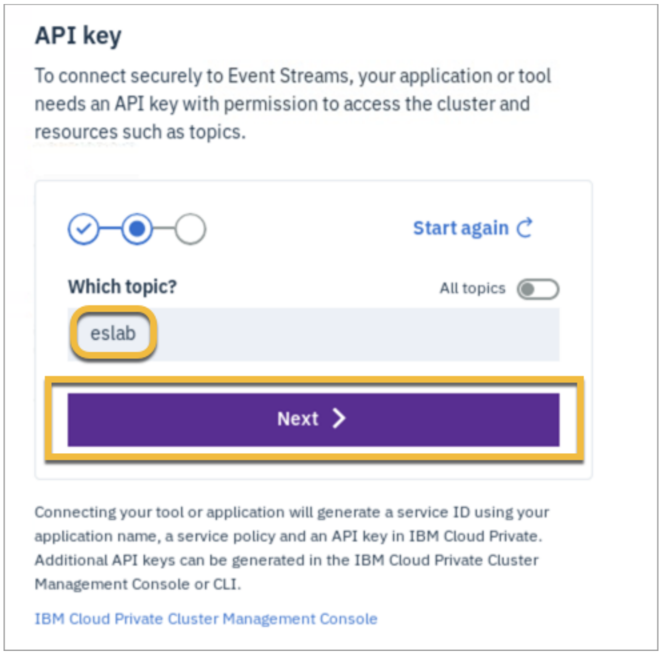
API key

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: eslabsink

What do you want it to do?

3. Enter **eslab** for **Which topic?**, and click **Next**.



API key

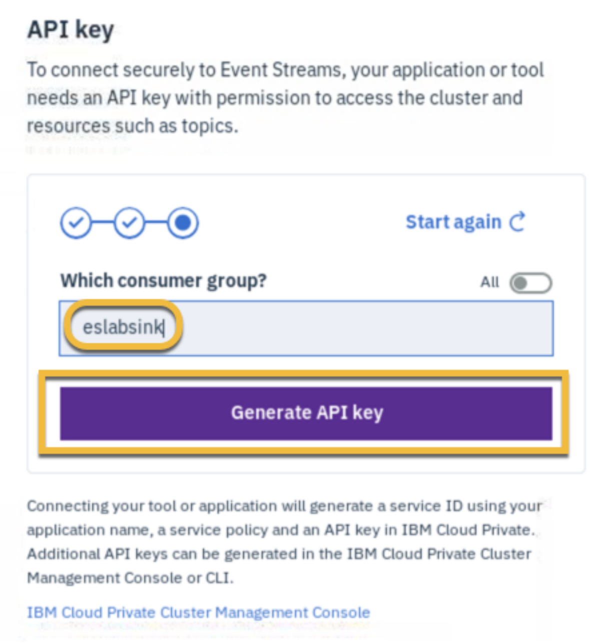
To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.

Which topic? eslab

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

4. Enter **eslabsink** for the **Which consumer group?**, and click **Generate API key**.



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.

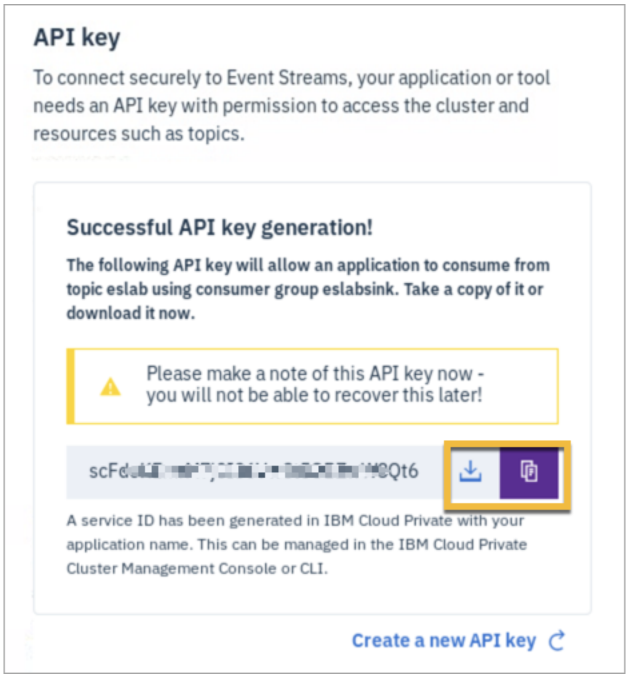
Which consumer group?

eslabsink

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

5. Click the button to either copy the key to the clipboard, or save it to a file.



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.

Successful API key generation!

The following API key will allow an application to consume from topic eslab using consumer group eslabsink. Take a copy of it or download it now.

Please make a note of this API key now - you will not be able to recover this later!

scFd[redacted]8Qt6

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.

C. Update worker properties

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

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

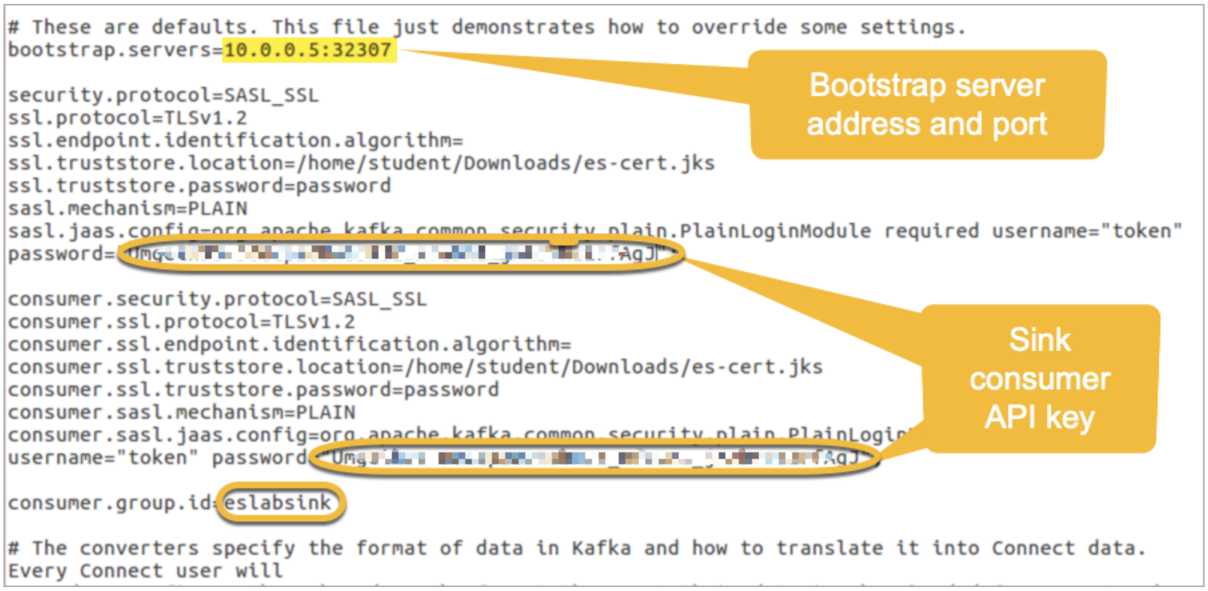
2. In the text editor menu, select **File > Save As**, edit the name of the file, and click **Save**.

3. Replace the consumer API keys with the one you just generated.

4. Add the following line after the `security` and `consumer` code stanzas:

`consumer.group.id=eslabsink`

NOTE: The file should still contain your bootstrap server address and port, from the previous lab exercise.



```
# 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=[redacted]

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.PlainLogin
username="token" password=[redacted]

consumer.group.id=eslabsink

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

Bootstrap server address and port

Sink consumer API key

5. Save and close the file.

Step 3. Test the sink connector

1. In a command terminal window, change to the Kafka root directory, and run the following command to start the connector:

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

   The connector starts, establishes a connection to the MQ queue manager, and starts listening for messages on topic eslab. Look for the message "Connection to MQ established," in the output.

2. Run the eslabtester starter application again. If it is not already open, in a new browser tab, enter the following URL:

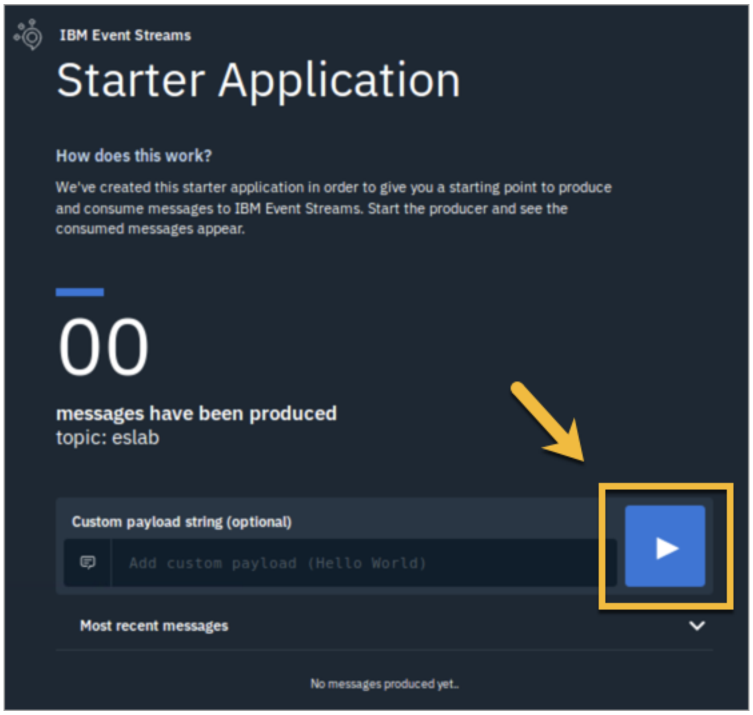
   `http://localhost:9080/elabtester`

   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.

3. In the eslabtester application, click the arrow to start producing messages.



Starter Application

How does this work?

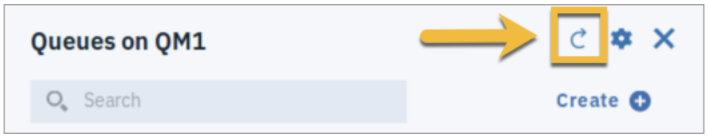
We've created this starter application in order to give you a starting point to produce and consume messages to IBM Event Streams. Start the producer and see the consumed messages appear.

00

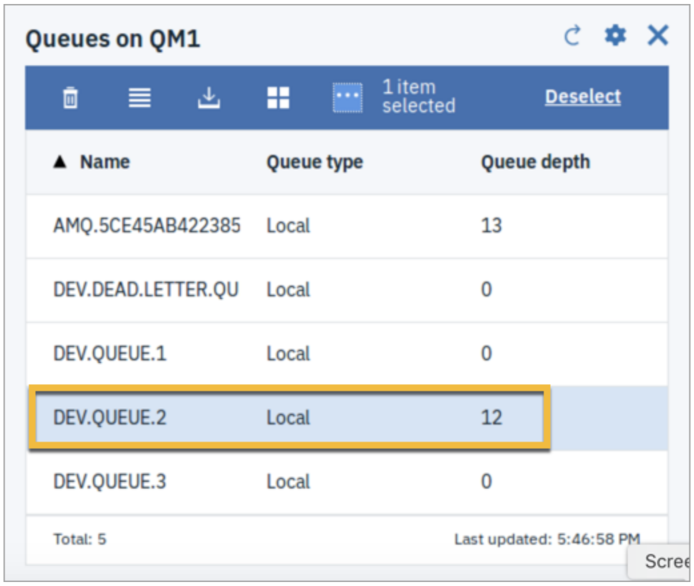
messages have been produced
topic: eslab

Custom payload string (optional)
Add custom payload (Hello World)
Most recent messages
No messages produced yet..

4. After the number of messages begins to increase, check the MQ console. Click the refresh icon for **Queues on QM1**.



The **Queue depth** for **DEV.QUEUE.2** increases as the number of messages increases.



▲ Name	Queue type	Queue depth
AMQ.5CE45AB422385	Local	13
DEV.DEAD.LETTER.QU	Local	0
DEV.QUEUE.1	Local	0
DEV.QUEUE.2	Local	12
DEV.QUEUE.3	Local	0

Total: 5   Last updated: 5:46:58 PM

5. In the eslabtester application, click the run button again to stop producing messages.

6. Check the MQ console again to verify that the messages were received.

7. Stop the connector and eslabtester application by pressing Ctrl-C in each terminal window.

End of exercise

End of Lab