Lab 3: Work with a load-producing application on Event Streams

Duration: 30 minutes

In this exercise, you install another sample application that you can use to generate workloads of a specific size. You can use one of the predefined load sizes, or you can specify your own settings to test throughput.

You must complete Labs 1-2 before proceeding with this lab.

Step 1. Install and configure the workload producer application

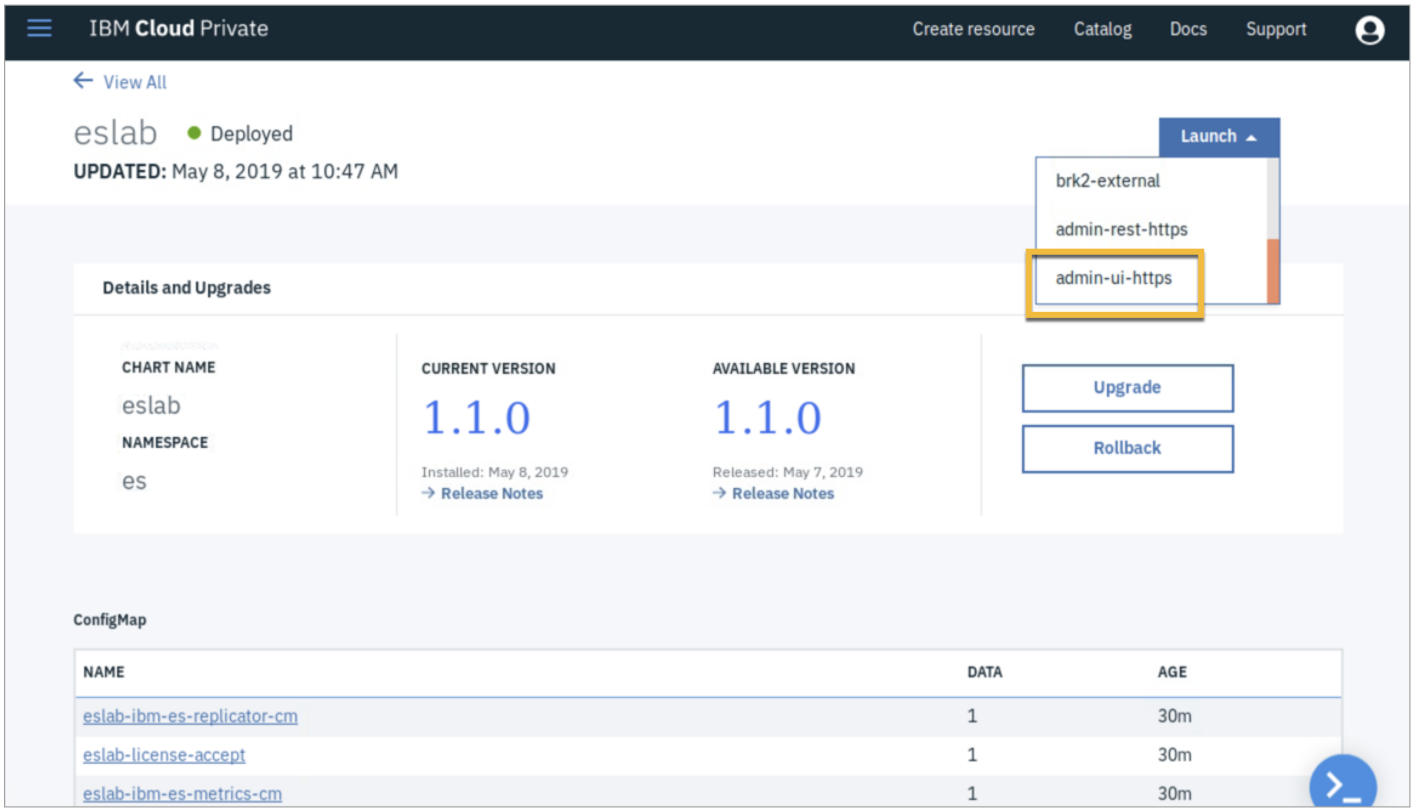
1. On the ICP Master virtual machine image, open Firefox and click the **IBM Cloud Private** bookmark tab, or enter the following address in a browser:

   `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 **Workload > Helm Releases > eslab**.

4. Click **Launch** in the upper right corner, and then select **admin-ui-https**.



eslab ● Deployed

UPDATED: May 8, 2019 at 10:47 AM

Details and Upgrades

CHART NAME	CURRENT VERSION	AVAILABLE VERSION
eslab	1.1.0	1.1.0
NAMESPACE	Installed: May 8, 2019	Released: May 7, 2019
es	→ Release Notes	→ Release Notes

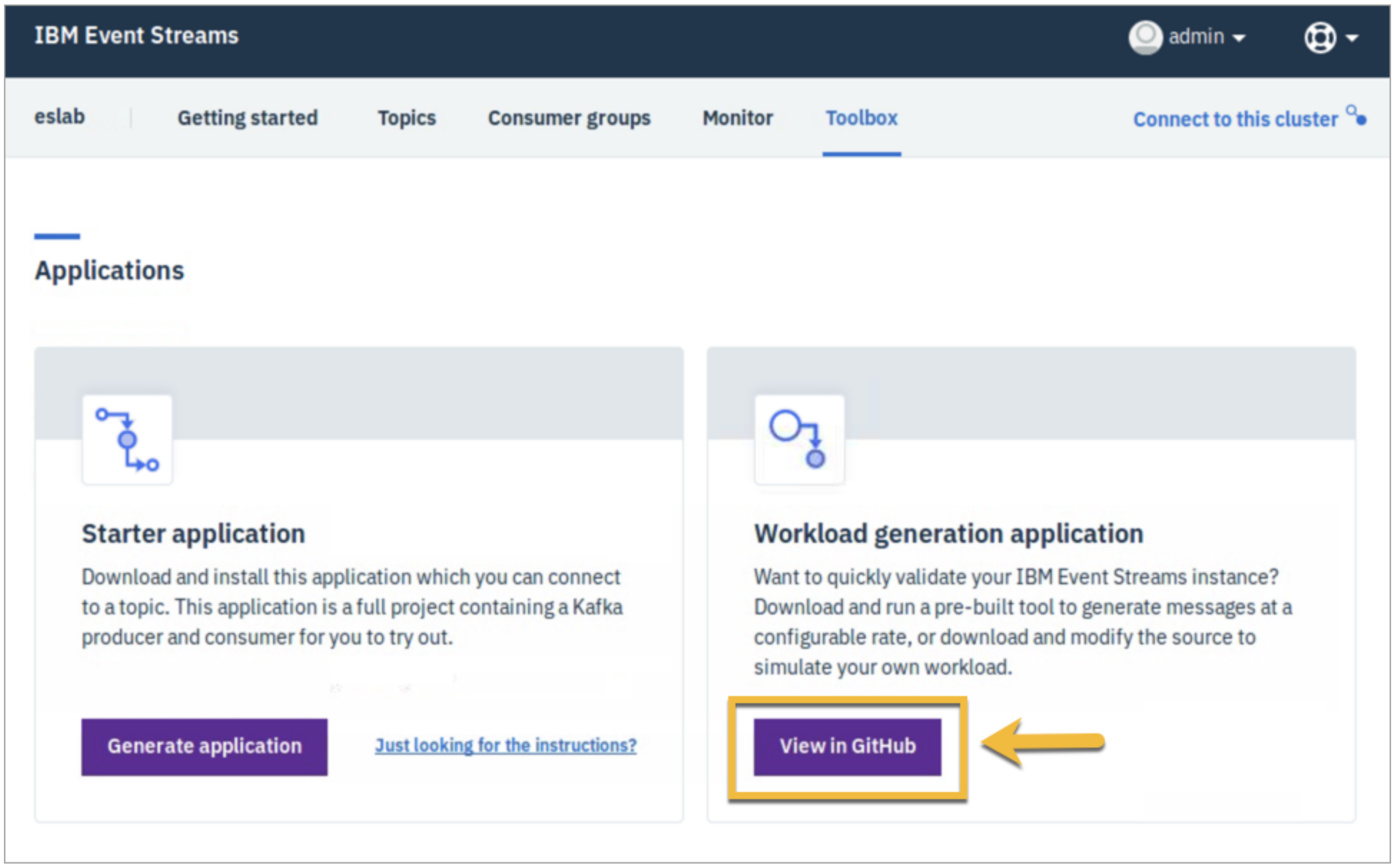
ConfigMap

NAME	DATA	AGE
eslab-ibm-es-replicator-cm	1	30m
eslab-license-accept	1	30m
eslab-ibm-es-metrics-cm	1	30m

5. Log in with the user ID **admin** and password **admin**.

6. Click the **Toolbox** tab to access tools.

7. Click **View on GitHub** in the Workload generation application.



Applications

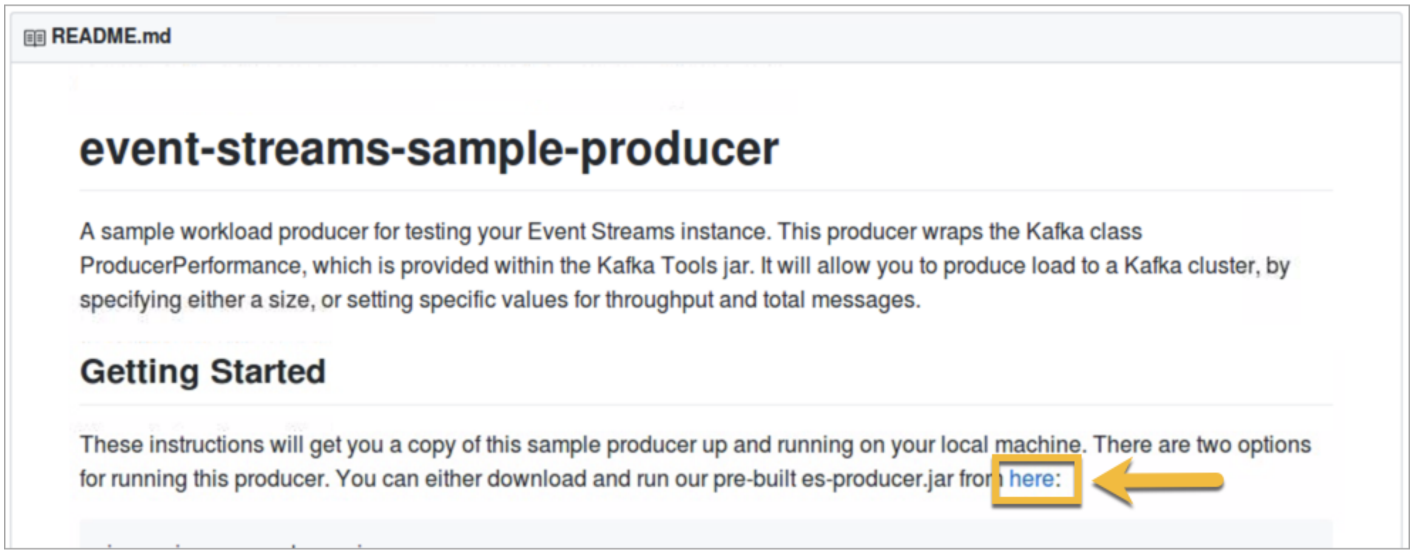
Starter application

Download and install this application which you can connect to a topic. This application is a full project containing a Kafka producer and consumer for you to try out.

Workload generation application

Want to quickly validate your IBM Event Streams instance? Download and run a pre-built tool to generate messages at a configurable rate, or download and modify the source to simulate your own workload.

8. GitHub opens in a new browser tab. Scroll down to the README.md page and click **here**.



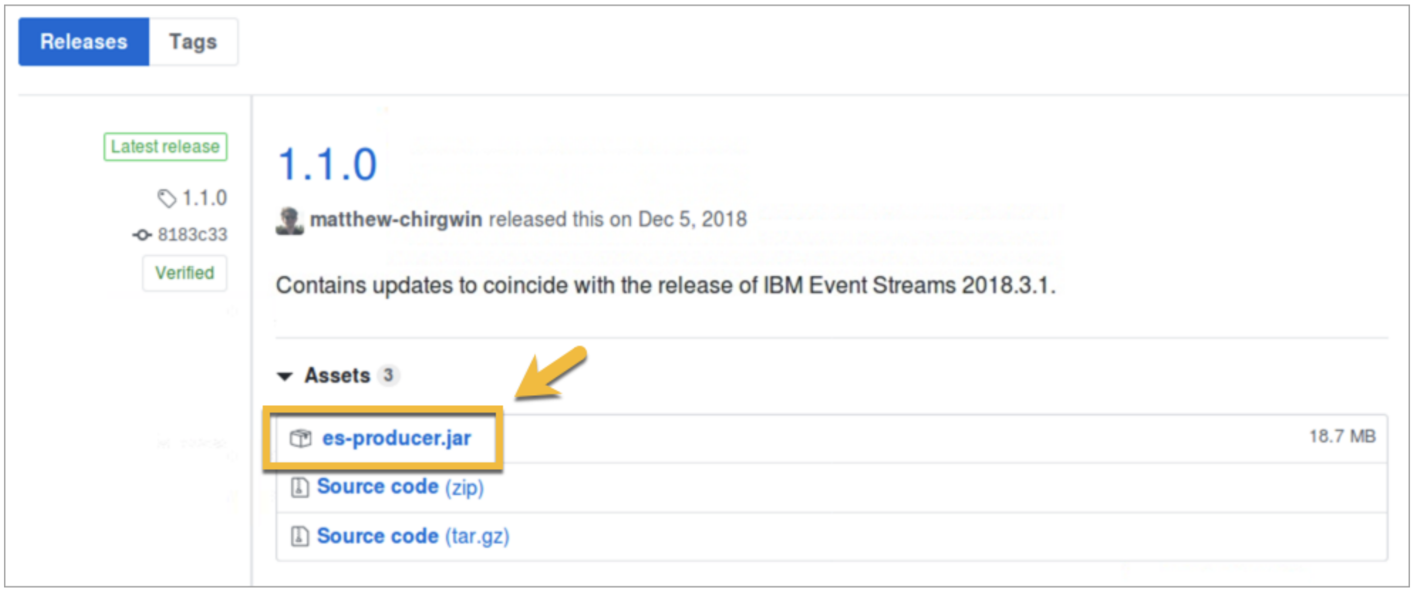
event-streams-sample-producer

A sample workload producer for testing your Event Streams instance. This producer wraps the Kafka class ProducerPerformance, which is provided within the Kafka Tools jar. It will allow you to produce load to a Kafka cluster, by specifying either a size, or setting specific values for throughput and total messages.

Getting Started

These instructions will get you a copy of this sample producer up and running on your local machine. There are two options for running this producer. You can either download and run our pre-built es-producer.jar from here:

9. Under **Latest Release**, click **es-producer.jar** to download it to /home/student/Downloads.

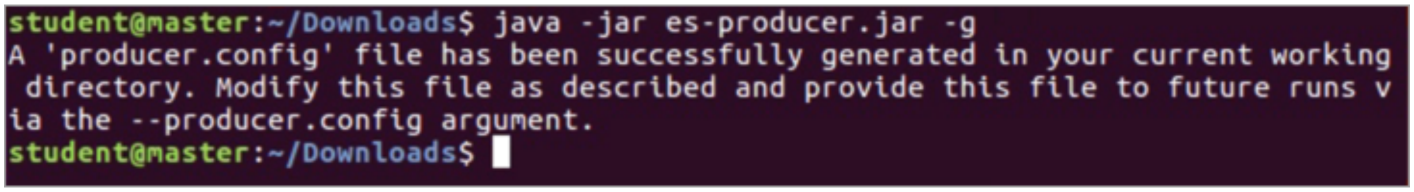


Contains updates to coincide with the release of IBM Event Streams 2018.3.1.

Select **Save file**, and click **OK**.

10. In a command terminal window, change to the Downloads directory (`cd Downloads`) and run the following command:

    ```
    java -jar es-producer.jar -g
    ```



```
student@master:~/Downloads$ java -jar es-producer.jar -g
A 'producer.config' file has been successfully generated in your current working
 directory. Modify this file as described and provide this file to future runs v
ia the --producer.config argument.
student@master:~/Downloads$ 
```

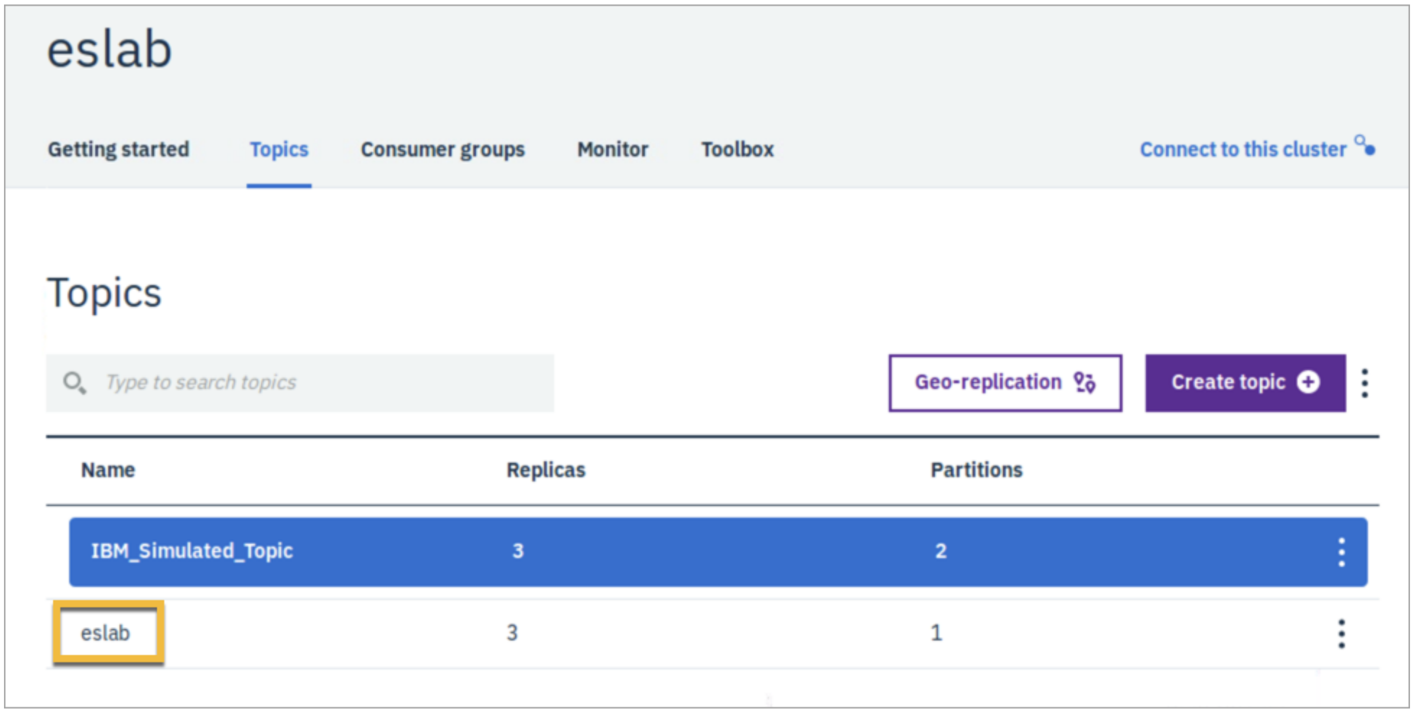
This command creates the configuration file, `producer.config` .

11. Run the following command to open this file in an editor:

    ```
    gedit producer.config
    ```

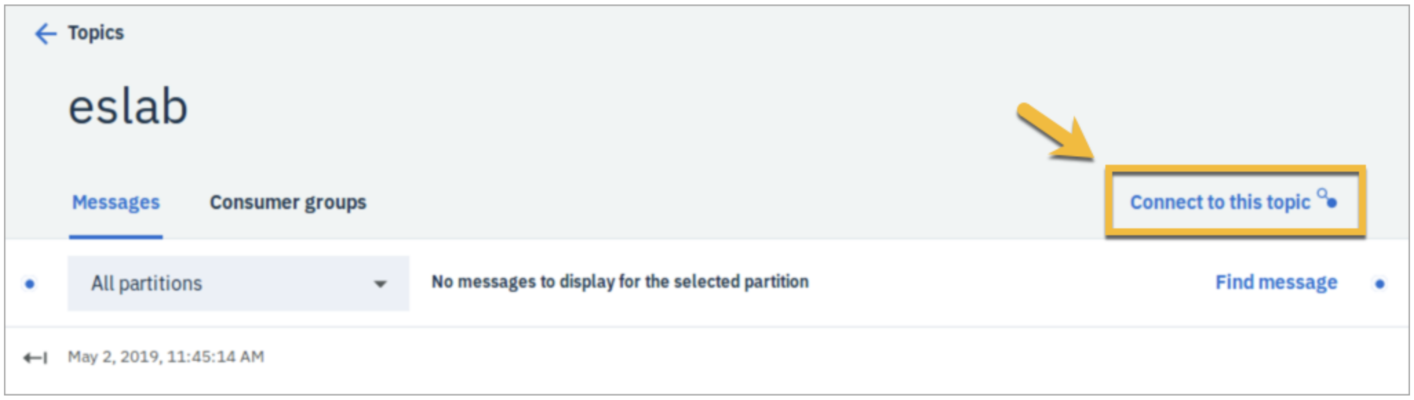
    You must add some information to this file that you can find in the Event Streams console.

12. In the Event Streams console, click the **Topics** tab, and then click **eslab**.

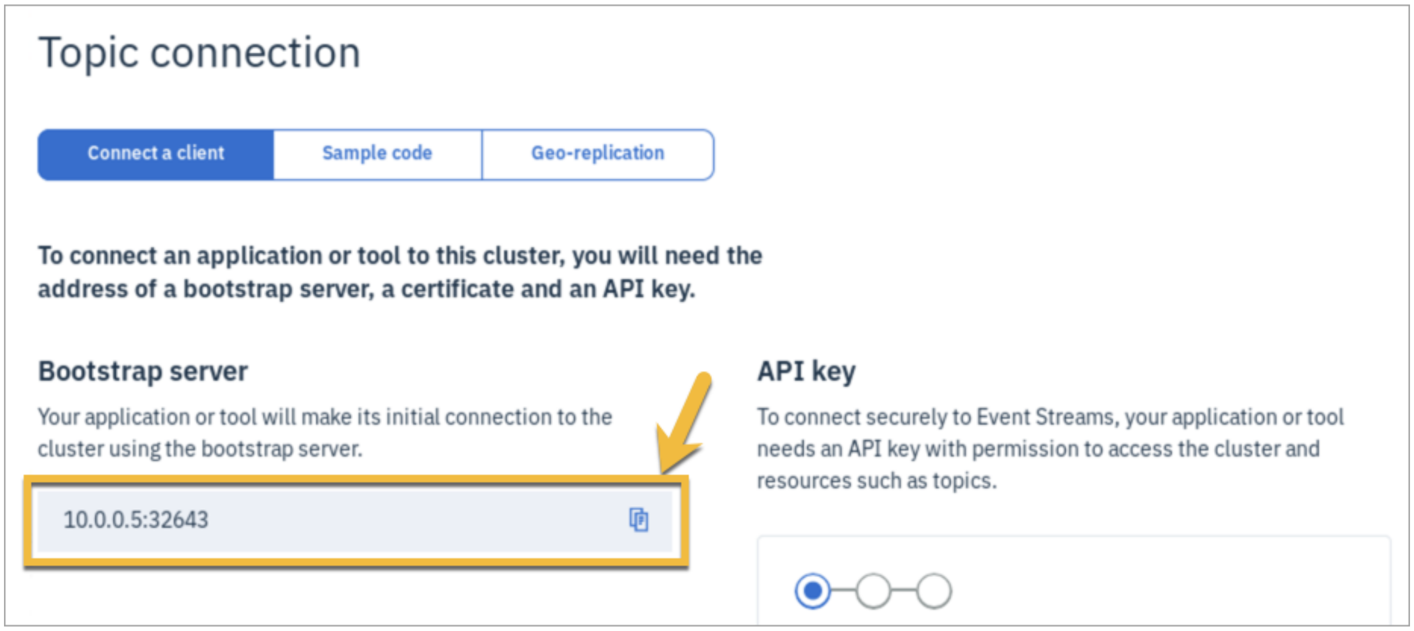


Name	Replicas	Partitions
IBM_Simulated_Topic	3	2
eslab	3	1

13. Click **Connect to this topic**.



14. Copy the **Bootstrap server** address.



Topic connection

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

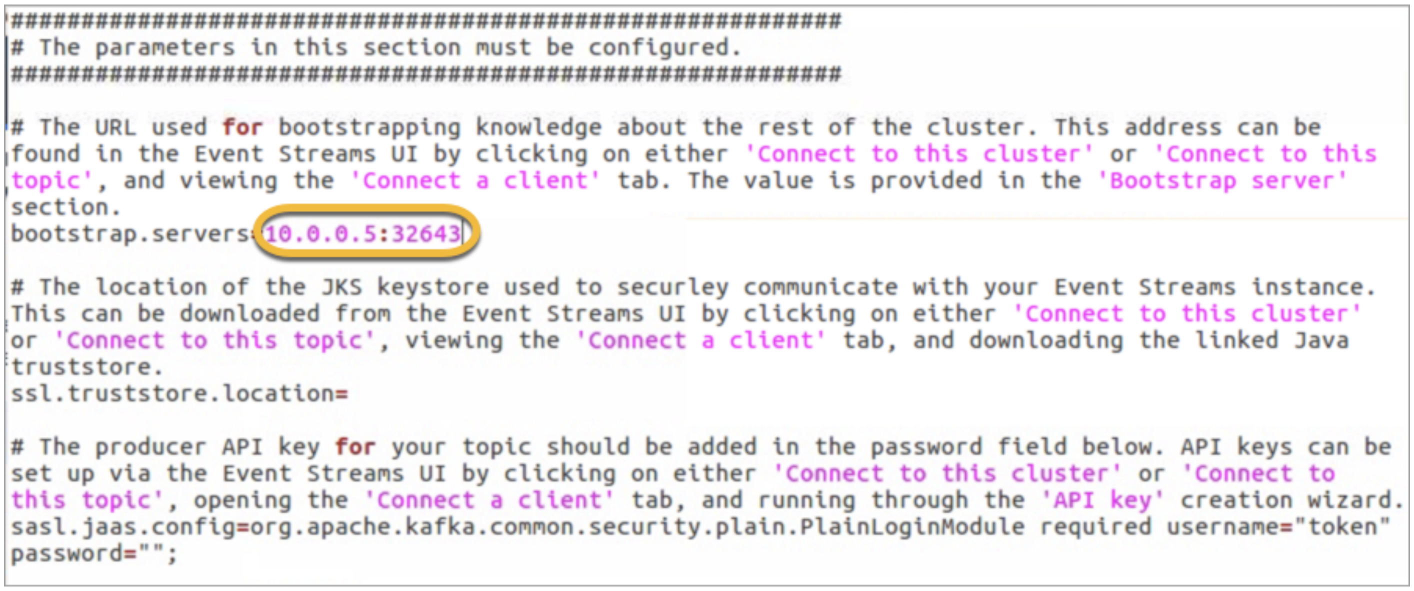
Your application or tool will make its initial connection to the cluster using the bootstrap server.

10.0.0.5:32643

API key

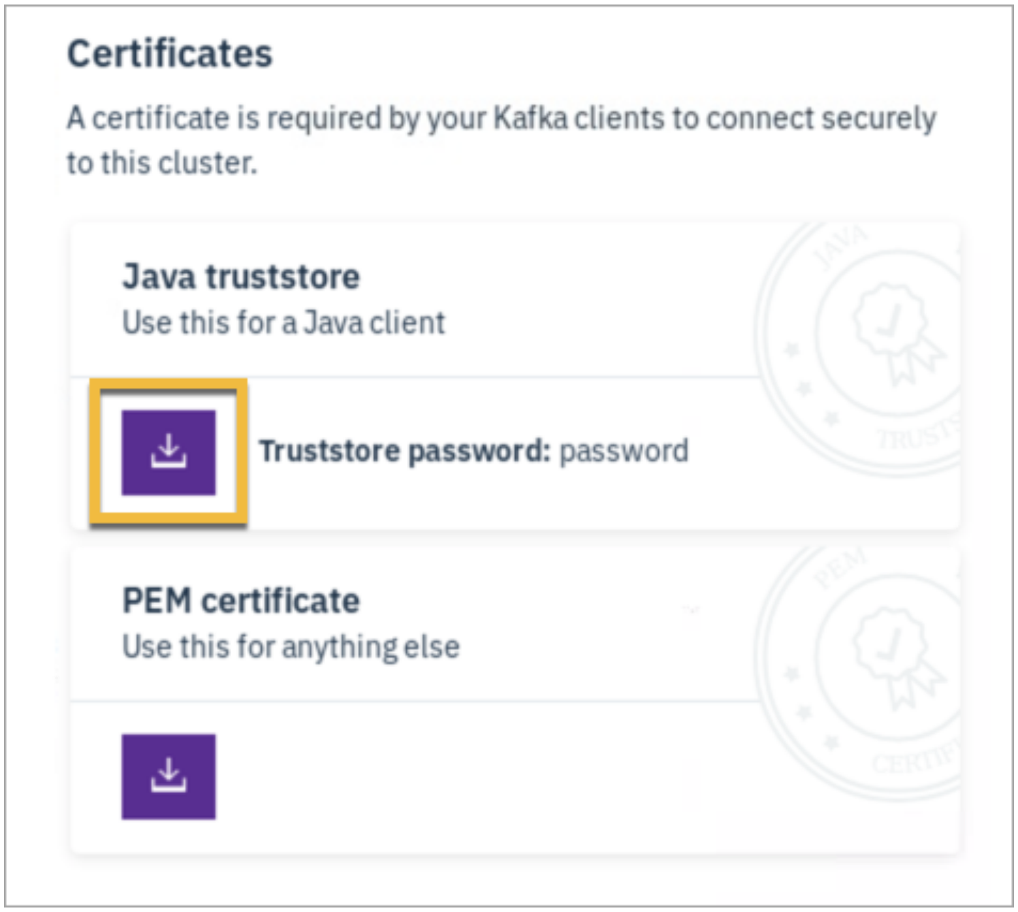
To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.

15. Paste this address in to the editor as the value for `bootstrap.servers` .



```
################################################################
# 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=10.0.0.5:32643

# The location of the JKS keystore used to securley 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=

# The producer 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="";
```

16. Go back to the **Topic connection** page and, under **Certificates**, click the icon to download the **Java truststore**.


Certificates

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

Java truststore

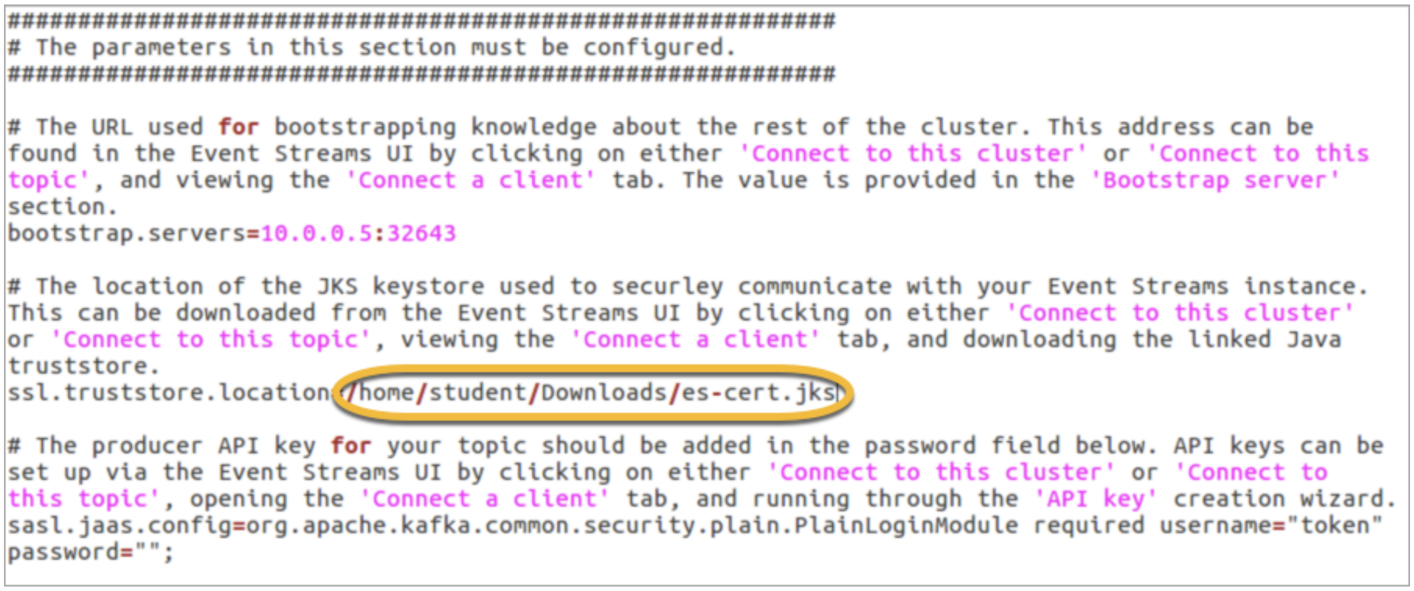
Use this for a Java client

Truststore password: password

PEM certificate

Use this for anything else

17. Choose **Save file**, and then enter the full pathname of the file in to the editor as the value for `ssl.truststore.location`.

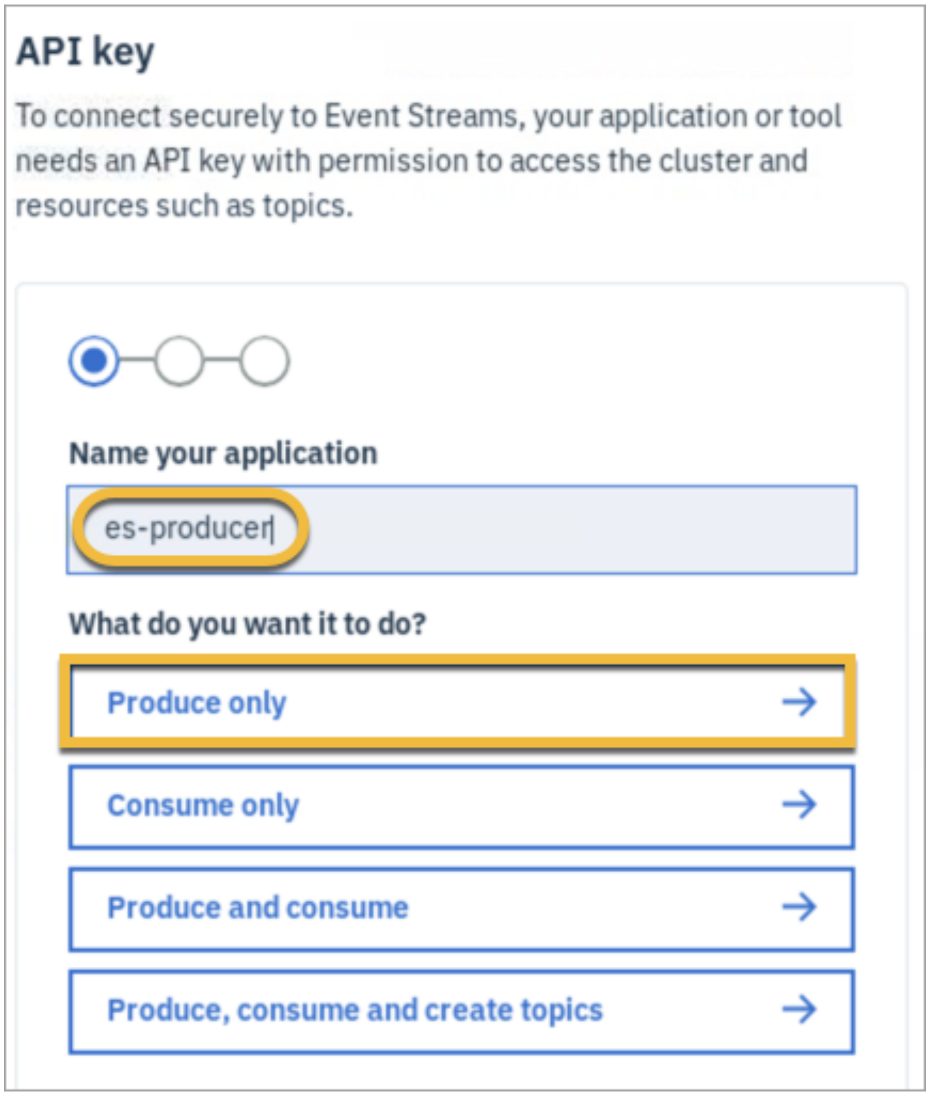


```
################################################################
# The parameters in this section must be configured.
################################################################

# 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=10.0.0.5:32643

# The location of the JKS keystore used to securley 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/student/Downloads/es-cert.jks

# The producer 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="";
```

In this case, the pathname is `/home/student/Downloads/es-cert.jks`.

18. Back on the Topic connection page, under **API key**, enter **es-producer** for the application name, and click

Produce only.



API key

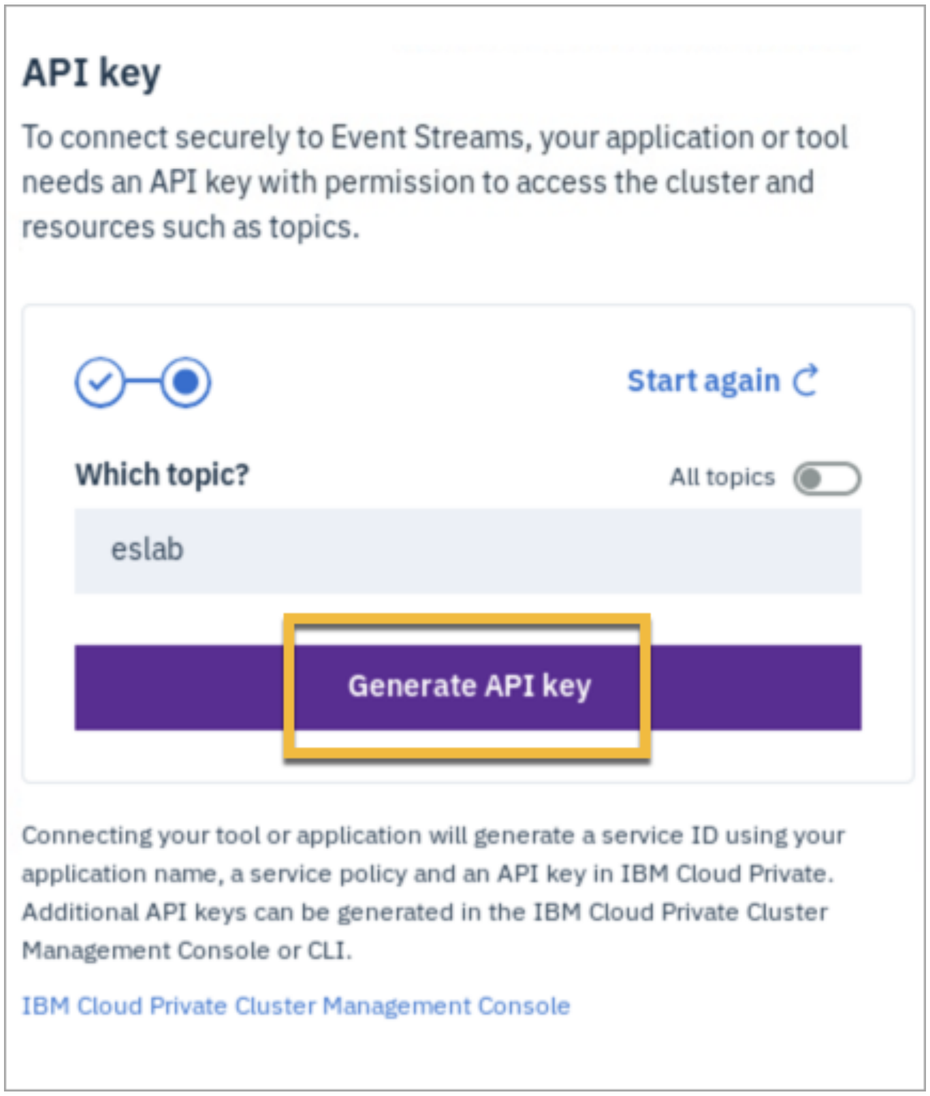
To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.

Name your application

es-producer

What do you want it to do?

19. Click **Generate API key**.



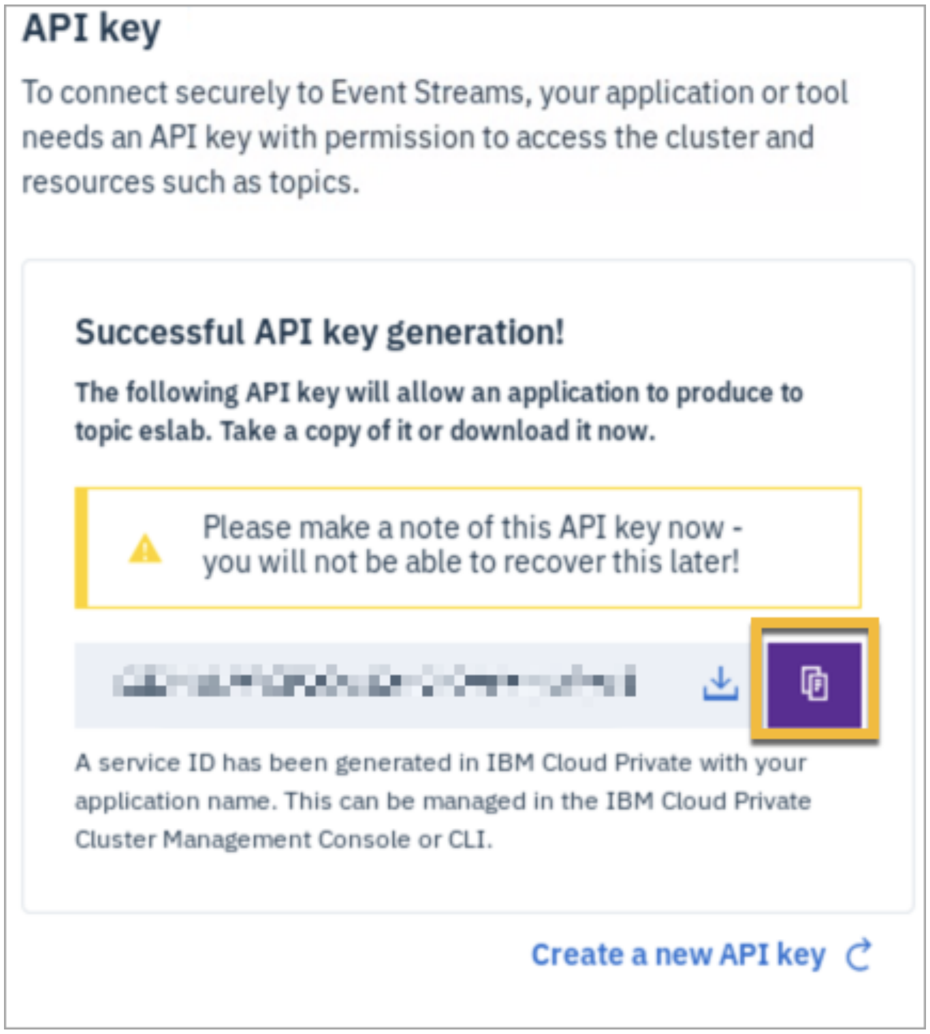
API key

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.

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

20. Click the icon to copy the 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.

Successful API key generation!

The following API key will allow an application to produce to topic eslab. Take a copy of it or download it now.

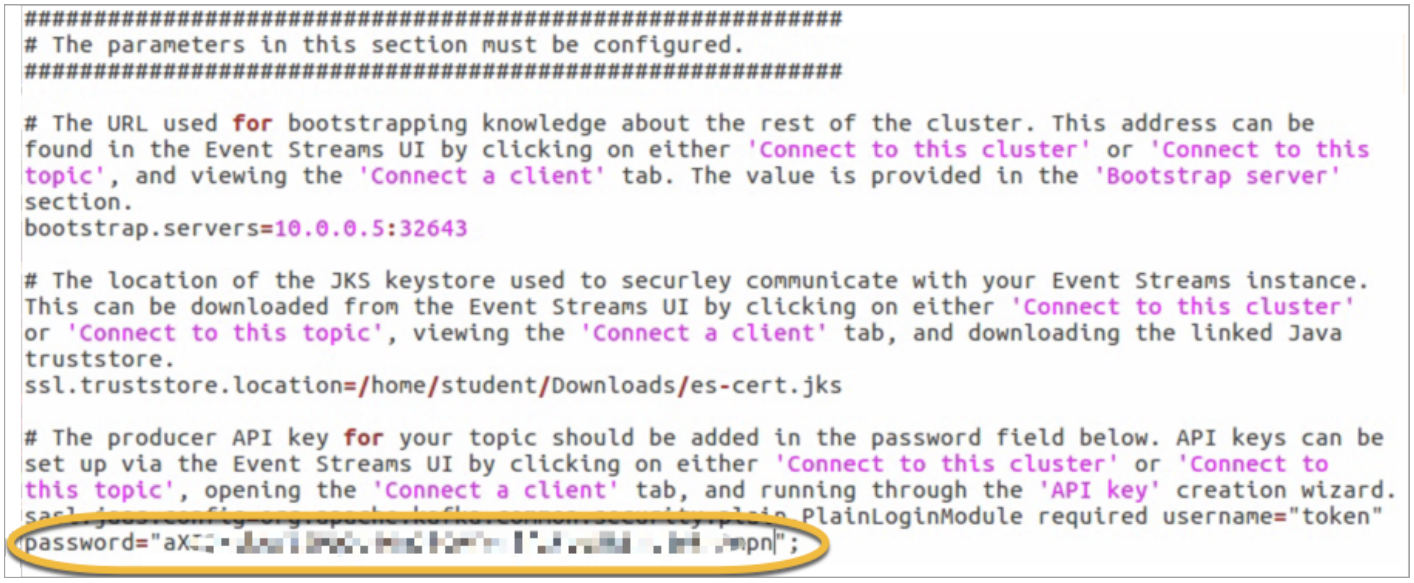
Please make a note of this API key now - you will not be able to recover this later!

[redacted]

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.

Create a new API key

21. Paste the API key in to the editor as the value for password. Paste between the double quotations marks.



```
################################################################
# 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=10.0.0.5:32643

# The location of the JKS keystore used to securley 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/student/Downloads/es-cert.jks

# The producer 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="aX[redacted]pn";
```

22. Save and close the `producer.config` file.

23. In the Event Streams console, click the **X** on the Topic connection page to close it.

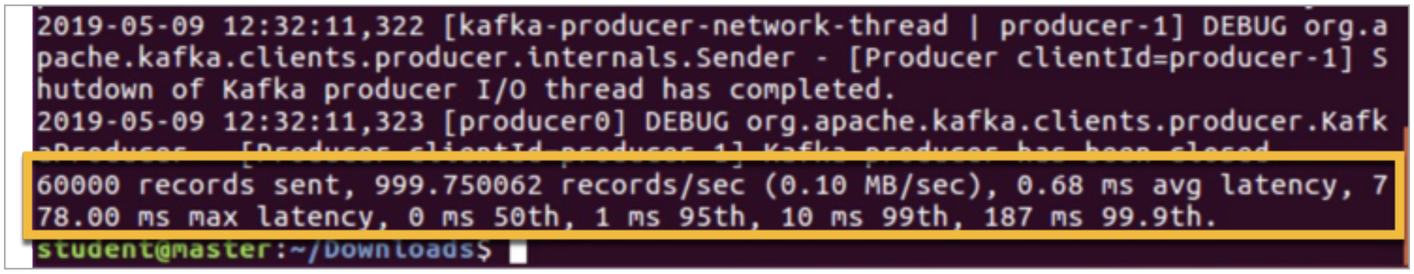
Step 2. Run the application with load

1. In a command terminal window, in the directory where you saved the es-producer.jar file (/home/student/Downloads), run the following command:

   ```
   java -jar es-producer.jar -t eslab -s small
   ```

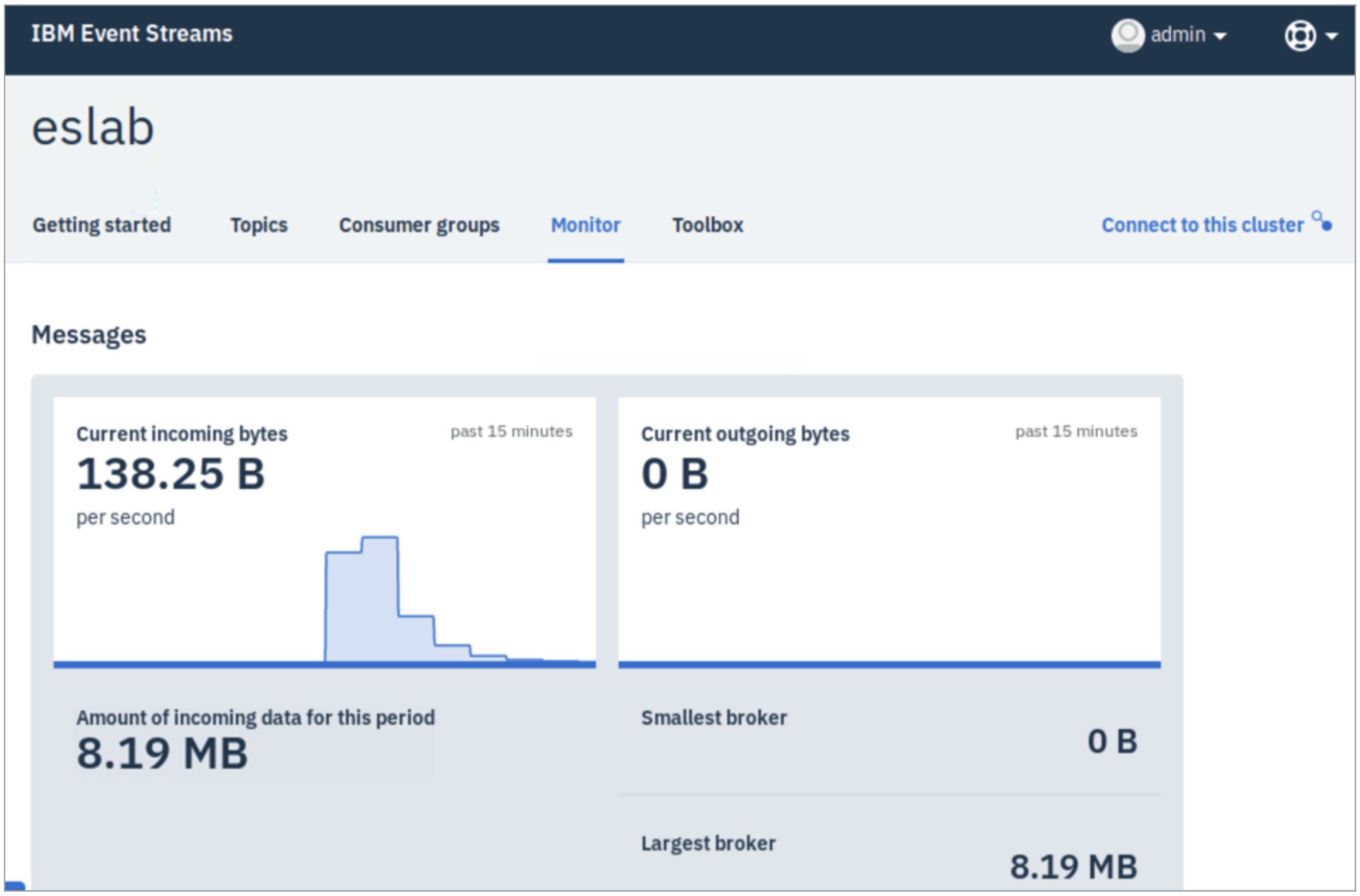
   This command specifies `eslab` for the Topic, and a predefined `small` size load.

   The output scrolls by very quickly, and then concludes with a message that reports the number of messages sent and some details about performance.

   ```
   2019-05-09 12:32:11,322 [kafka-producer-network-thread | producer-1] DEBUG org.a
   pache.kafka.clients.producer.internals.Sender - [Producer clientId=producer-1] S
   hutdown of Kafka producer I/O thread has completed.
   2019-05-09 12:32:11,323 [producer0] DEBUG org.apache.kafka.clients.producer.Kafk
   aProducer - [Producer clientId=producer-1] Kafka producer has been closed
   60000 records sent, 999.750062 records/sec (0.10 MB/sec), 0.68 ms avg latency, 7
   78.00 ms max latency, 0 ms 50th, 1 ms 95th, 10 ms 99th, 187 ms 99.9th.
   student@master:~/Downloads$
   ```

2. Go back to the Event Streams console and click the **Monitor** tab.



Here you see some metrics for the small load. Scroll down to see all the information that is available. The data refreshes every few seconds. You learn more about monitoring later in this course.

3. Run the test again, but this time with a larger load. In a command terminal window, run the following command:

```
java -jar es-producer.jar -t eslab -s large
```

4. Go back to the **Monitor** tab in the Event Streams console and notice what happens while the test runs. Notice the differences between running the small load and the large load.

   If you want, you can also go back to the previous lab and run the starter application (consumer) again, and monitor the results in the console.

End of exercise