Building Streaming Data Analytics Solutions on AWS -

Lab 3: Introduction to Access Control with Amazon MSK

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Lab overview

You are a data engineer at AnyCompany Online Shopping. AnyCorp Online Shopping sells a variety of consumer goods through e-commerce transactions in five categories: books, food, apparel, electronics, and home goods.

Your company wants a low latency solution in real-time to analyze web clickstream events from its online customers who interact with their retail website. Your executive team wants to see top sales by category updated in near real time. It wants a solution with no setup costs and without having to manage servers. The team decides to build its solution on Amazon Web Services (AWS) using Amazon Managed Streaming for Apache Kafka (Amazon MSK) and Amazon Managed Service for Apache Flink services.

You are responsible to build a real-time streaming analytics pipeline in Managed Apache Flink Studio to meet the requirements. Before you build the streaming pipeline, you want to understand AWS Identity and Access Management (IAM) access control for Amazon MSK. You can use this to handle both authentication and authorization for your MSK cluster. The Amazon MSK IAM access control method avoids the need to use one mechanism for authentication and another for authorization.

In this lab, you will learn about the IAM method to authenticate and authorize users of an MSK cluster.

OBJECTIVES

After completing this lab, you will be able to do the following:

- Publish to and consume from an MSK cluster using IAM authenticated broker URLs with a Java demo producer and Java demo consumer.
- Learn about the IAM method to authenticate and authorize users of an MSK cluster.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

- **Command:** A command that you must run.
- **Expected output:** A sample output that you can use to verify the output of a command or edited file.
- **Note: Note:** A hint, tip, or important guidance.
- **Additional information:** Where to find more information.
- **Warning:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
- **Solution:** An answer to a question or challenge.
- **Task complete:** A conclusion or summary point in the lab.

Start lab

1. To launch the lab, at the top of the page, choose Start lab.

Caution: You must wait for the provisioned AWS services to be ready before you can continue.

2. To open the lab, choose Open Console.

You are automatically signed in to the AWS Management Console in a new web browser tab.

WARNING: Do not change the Region unless instructed.

COMMON SIGN-IN ERRORS

Error: You must first sign out

If you see the message, **You must first log out before logging into a different AWS account:**

- Choose the **click here** link.
- Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
- Choose Open Console again.

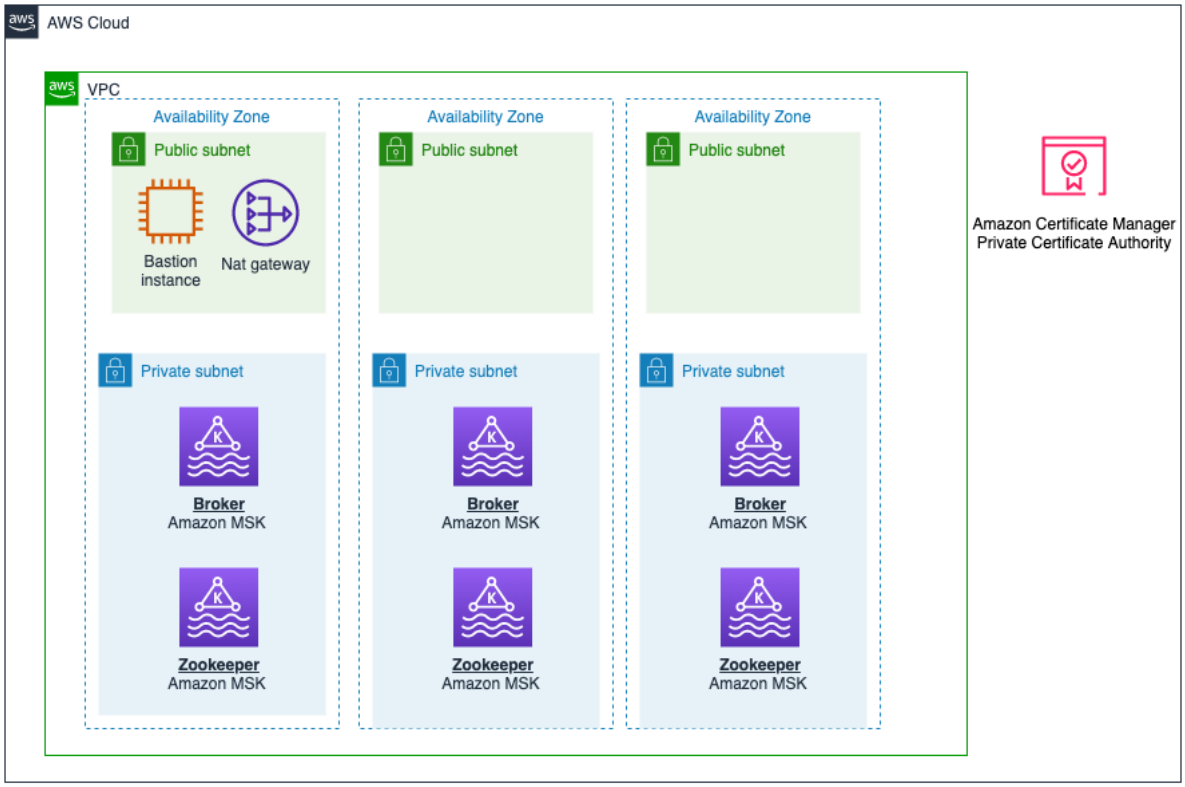
Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

LAB ENVIRONMENT

The following diagram shows the basic architecture of the lab environment:



The lab environment includes an Amazon MSK environment configured in a typical six-subnet (three public, three private) virtual private cloud (VPC). One public subnet contains a NAT gateway and a bastion host, which you use in the lab.

The provisioned MSK cluster has been configured for the IAM authentication method.

AWS SERVICES NOT USED IN THIS LAB

AWS service capabilities used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

Task 1: Inspecting the MSK cluster

In this task, you will navigate to the Amazon Elastic Compute Cloud (Amazon EC2) instance that serves as the Amazon MSK bastion host through a terminal session window. You will run the

command to display the output of a shell script that has been created to populate your environment with the various network addresses needed to access the MSK cluster.

Session Manager, a capability of AWS Systems Manager, provides secure instance management without the need to open inbound ports, maintain bastion hosts, or manage Secure Shell (SSH) keys.

Note: If the command shell is unresponsive when you type a command,to return to the shell prompt, press **Ctrl+C**.

3. To access Session Manager, in the list to the left of these instructions, copy the **CommandHostSessionManagementUrl** value. To display the Amazon EC2 (Amazon MSK bastion host) terminal window, paste the URL into a new web browser tab and press **Enter**.
4. **Command:** To output the various network addresses needed to access the MSK cluster, run the following command.

```
cat /opt/kafka_2.12-2.2.1/msk.env
```

Expected output:

```
****************************
**** This is OUTPUT ONLY. ****
****************************

xport PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/opt/kafka_2.12-
2.2.1/bin:/opt/apache-maven-3.8.8/bin:/usr/local/bin/
export MSK_ARN=arn:aws:kafka:us-west-2:398349607143:cluster/MSK-Demo/3519b212-
c2f5-49d2-a283-209785647a5e-10
export MSK_BOOTSTRAP="b-2.mskdemo.865my8.c10.kafka.us-west-
2.amazonaws.com:9094,b-1.mskdemo.865my8.c10.kafka.us-west-
2.amazonaws.com:9094,b-3.mskdemo.865my8.c10.kafka.us-west-
2.amazonaws.com:9094"
export MSK_ZOOKEEPER="z-1.mskdemo.865my8.c10.kafka.us-west-
2.amazonaws.com:2181,z-3.mskdemo.865my8.c10.kafka.us-west-
2.amazonaws.com:2181,z-2.mskdemo.865my8.c10.kafka.us-west-
2.amazonaws.com:2181"
export MSKIAM_BOOTSTRAP="b-2.mskdemo.865my8.c10.kafka.us-west-
2.amazonaws.com:9098,b-3.mskdemo.865my8.c10.kafka.us-west-
2.amazonaws.com:9098,b-1.mskdemo.865my8.c10.kafka.us-west-
2.amazonaws.com:9098"
```

The output is a series of export statements that include the cluster Amazon Resource Names (ARNs), bootstrap brokers, and the Apache Zookeeper address for the cluster.

Note: The broker URLs in **MSK_BOOTSTRAP** uses TLS authenticated **port 9094** and the broker URLs in **MSKIAM_BOOTSTRAP** uses IAM authenticated **port 9098** for communication with the MSK cluster.

Warning: Continue to the next task in the same session window.

Task complete: You have successfully inspected the MSK cluster.

Task 2: Publish to and consume from an IAM authenticated MSK cluster

IAM access control for Amazon MSK handles both authentication and authorization of producers and consumers that communicate with the cluster.

Note: The following information explains the steps to enable IAM authentication in your own environment. **This lab has been pre-configured with these settings and no additional action is required. You create a topic and publish to the topic in Task 2.1.**

As part of enabling IAM authentication from a bastion host, an IAM role needs to be attached to the bastion host. This role should include permissions for all MSK cluster actions.

As part of enabling IAM authentication for the producers and consumers written in Java, the following lines of code needs to be added to the Kafka client configuration.

```
**********************************
**** This is an EXAMPLE ONLY. ****
**********************************
```

```
props.put("security.protocol", "SASL_SSL");
props.put("sasl.mechanism", "AWS_MSK_IAM");
props.put("sasl.jaas.config", "software.amazon.msk.auth.iam.IAMLoginModule
required;");
props.put("sasl.client.callback.handler.class",
"software.amazon.msk.auth.iam.IAMClientCallbackHandler");
```

- The following lines of code needs to be added to the

  `<dependencies>` in the **pom.xml** file.

```
**********************************
**** This is an EXAMPLE ONLY. ****
**********************************
```

```
 <dependency>
   <groupId>software.amazon.msk</groupId>
   <artifactId>aws-msk-iam-auth</artifactId>
   <version>1.0.0</version>
 </dependency>
```

For more information about IAM access control, see IAM Access Control in the *Amazon MSK Developer Guide*.

TASK 2.1: CREATE A TOPIC, COMPILE A JAVA PRODUCER/CONSUMER AND PUBLISH TO AN IAM AUTHENTICATED MSK CLUSTER

Create and list a topic.

5. **Command:** To load the environment variable that references the MSK cluster into your shell environment, run the following command:

```
source /opt/kafka_2.12-2.2.1/msk.env
```

Expected output:

None, unless there is an error.

6. **Command:** To create a topic (**ExampleTopic**), run the following command:

```
kafka-topics.sh --create --topic ExampleTopic --partitions 5 --replication-
factor 3 --bootstrap-server $MSK_BOOTSTRAP  --command-config
/opt/client.properties
```

Expected output:

None, unless there is an error.

7. **Command:** To list the topics on the cluster to verify successful creation, run the following command:

```
kafka-topics.sh --list --bootstrap-server $MSK_BOOTSTRAP --command-config
/opt/client.properties
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

ExampleTopic
__amazon_msk_canary
__consumer_offsets
```

You see a list of topics, which includes a couple that were created at cluster creation, and the topic that we created (**ExampleTopic**).

TASK 2.2: COMPILE JAVA PRODUCER AND CONSUMER

Communication with an IAM authenticated broker would require a Java based Kafka client. The source code for a Java Kafka producer and consumer has been loaded onto the bastion host. It is located at `/opt/msk-java`.

8. **Command:** To navigate to `/opt/msk-java`, run the following command:

```
cd /opt/msk-java
```

Expected output:

None, unless there is an error.

Within this directory is a Maven project that has already been set up.

9. **Command:** To compile the project, run the following command:

```
mvn package
```

Expected output: Output text has been truncated.

```
INFO] Scanning for projects...
[INFO]
[INFO] ----------------< com.amazonaws.examples:msk-auth-demo >---------------
-
[INFO] Building msk-auth-demo 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
-
INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing /opt/msk-java/target/msk-auth-demo-1.0-SNAPSHOT.jar with
/opt/msk-java/target/msk-auth-demo-1.0-SNAPSHOT-shaded.jar
[INFO] Dependency-reduced POM written at: /opt/msk-java/dependency-reduced-
pom.xml
[INFO] Dependency-reduced POM written at: /opt/msk-java/dependency-reduced-
pom.xml
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  19.099 s
[INFO] Finished at: 2023-10-20T14:29:45Z
[INFO] ------------------------------------------------------------------------
```

When this task is complete, you see

```
BUILD SUCC
```

and are returned back to the bash shell.

TASK 2.3: PUBLISH TO THE TOPIC

10. **Command:** To publish to **ExampleTopic** with the Java program that you complied in the previous step using IAM authenticated broker URLS, run the following command:

```
java -cp target/msk-auth-demo-1.0-SNAPSHOT.jar
com.amazonaws.examples.DemoProducer $MSKIAM_BOOTSTRAP ExampleTopic
```

Expected output:

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************

[main] INFO org.apache.kafka.clients.producer.ProducerConfig - ProducerConfig
values:
        acks = all
        batch.size = 16384
        bootstrap.servers = [b-1.mskdemo.7zfurd.c7.kafka.us-east-
1.amazonaws.com:9098, b-2.mskdemo.7zfurd.c7.kafka.us-east-
1.amazonaws.com:9098, b-3.mskdemo.7zfurd.c7.kafka.us-east-
1.amazonaws.com:9098]
        buffer.memory = 33554432
        client.dns.lookup = default
```

```
        client.id =
        compression.type = none
        connections.max.idle.ms = 540000
        delivery.timeout.ms = 120000
        enable.idempotence = false
        interceptor.classes = []
        key.serializer = class
org.apache.kafka.common.serialization.StringSerializer
        linger.ms = 1
        max.block.ms = 60000
        max.in.flight.requests.per.connection = 5
        max.request.size = 1048576
        metadata.max.age.ms = 300000
        metric.reporters = []
        metrics.num.samples = 2
        metrics.recording.level = INFO
        metrics.sample.window.ms = 30000
        partitioner.class = class
org.apache.kafka.clients.producer.internals.DefaultPartitioner
        receive.buffer.bytes = 32768
        reconnect.backoff.max.ms = 1000
        reconnect.backoff.ms = 50
        request.timeout.ms = 30000
        retries = 0
        retry.backoff.ms = 100
        sasl.client.callback.handler.class = class
software.amazon.msk.auth.iam.IAMClientCallbackHandler
        sasl.jaas.config = [hidden]
        sasl.kerberos.kinit.cmd = /usr/bin/kinit
        sasl.kerberos.min.time.before.relogin = 60000
        sasl.kerberos.service.name = null
        sasl.kerberos.ticket.renew.jitter = 0.05
        sasl.kerberos.ticket.renew.window.factor = 0.8
        sasl.login.callback.handler.class = null
        sasl.login.class = null
        sasl.login.refresh.buffer.seconds = 300
        sasl.login.refresh.min.period.seconds = 60
        sasl.login.refresh.window.factor = 0.8
        sasl.login.refresh.window.jitter = 0.05
        sasl.mechanism = AWS_MSK_IAM
        security.protocol = SASL_SSL
        send.buffer.bytes = 131072
        ssl.cipher.suites = null
        ssl.enabled.protocols = [TLSv1.2, TLSv1.1, TLSv1]
        ssl.endpoint.identification.algorithm = https
        ssl.key.password = null
        ssl.keymanager.algorithm = SunX509
        ssl.keystore.location = null
        ssl.keystore.password = null
        ssl.keystore.type = JKS
        ssl.protocol = TLS
        ssl.provider = null
        ssl.secure.random.implementation = null
        ssl.trustmanager.algorithm = PKIX
        ssl.truststore.location = null
        ssl.truststore.password = null
```

```
        ssl.truststore.type = JKS
        transaction.timeout.ms = 60000
        transactional.id = null
        value.serializer = class
org.apache.kafka.common.serialization.StringSerializer

[main] INFO org.apache.kafka.common.security.authenticator.AbstractLogin -
Successfully logged in.
[main] WARN org.apache.kafka.clients.producer.ProducerConfig - The
configuration 'min.insync.replicas' was supplied but isn't a known config.
[main] INFO org.apache.kafka.common.utils.AppInfoParser - Kafka version: 2.2.1
[main] INFO org.apache.kafka.common.utils.AppInfoParser - Kafka commitId:
55783d3133a5a49a
>[kafka-producer-network-thread | producer-1] INFO
org.apache.kafka.clients.Metadata - Cluster ID: lycb5mcpTfid-G4Hg5CR7Q
```

Add a message to the queue.

11. From the blinking cursor, enter a sample message

    `Hello World!`, and at the **>** prompt press **ENTER**.

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

Hello World!
sent message to topic:EcampleTopic partition:1 offset:0
>
```

Note:

- Keep the terminal window open as we proceed to the next sub-task.
- The broker URLs in *MSKIAM_BOOTSTRAP* uses IAM authenticated *port 9098* to publish to and consume from a Java client.

You have successfully created a topic on the cluster and published a message to the topic using a Java demo producer.

TASK 2.4: CONSUME FROM AN IAM AUTHENTICATED MSK CLUSTER

In this sub-task, you consume the messages from the created topic.

Setup a new terminal window in a new browser tab.

12. To access Session Manager, in the list to the left of these instructions, copy the **CommandHostSessionManagementUrl** value, and to display the Amazon EC2 (Amazon MSK bastion host) terminal window, paste the URL into a new web browser tab and press **Enter**.
13. **Command:** To load the environment variable that references the MSK cluster into your shell environment,run the following command:

```
source /opt/kafka_2.12-2.2.1/msk.env
```

Expected output:

None, unless there is an error.

Warning: Continue the rest of the sub-task in the same session window.

Note: The source code for a Java Kafka producer and consumer has been loaded onto the lab basti on host. It is located at `/opt/msk-java`.

14. **Command:** To navigate to `/opt/msk-java`, run the following command:

```
cd /opt/msk-java
```

Expected output:

None, unless there is an error.

15. **Command:** To consume the messages using a Java client with IAM authenticated broker URLs, run the following command:

```
java -cp target/msk-auth-demo-1.0-SNAPSHOT.jar
com.amazonaws.examples.DemoConsumer $MSKIAM_BOOTSTRAP ExampleTopic
```

Expected output:

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************

[main] INFO org.apache.kafka.clients.consumer.ConsumerConfig - ConsumerConfig
values:
        auto.commit.interval.ms = 5000
        auto.offset.reset = earliest
        bootstrap.servers = [b-1.mskdemo.7zfurd.c7.kafka.us-east-
1.amazonaws.com:9098, b-2.mskdemo.7zfurd.c7.kafka.us-east-
1.amazonaws.com:9098, b-3.mskdemo.7zfurd.c7.kafka.us-east-
1.amazonaws.com:9098]
        check.crcs = true
        client.dns.lookup = default
        client.id =
        connections.max.idle.ms = 540000
        default.api.timeout.ms = 60000
        enable.auto.commit = false
        exclude.internal.topics = true
        fetch.max.bytes = 52428800
        fetch.max.wait.ms = 500
        fetch.min.bytes = 1
        group.id = foobar
        heartbeat.interval.ms = 3000
```

```
        interceptor.classes = []
        internal.leave.group.on.close = true
        isolation.level = read_uncommitted
        key.deserializer = class
org.apache.kafka.common.serialization.StringDeserializer
        max.partition.fetch.bytes = 1048576
        max.poll.interval.ms = 300000
        max.poll.records = 500
        metadata.max.age.ms = 300000
        metric.reporters = []
        metrics.num.samples = 2
        metrics.recording.level = INFO
        metrics.sample.window.ms = 30000
        partition.assignment.strategy = [class
org.apache.kafka.clients.consumer.RangeAssignor]
        receive.buffer.bytes = 65536
        reconnect.backoff.max.ms = 1000
        reconnect.backoff.ms = 50
        request.timeout.ms = 30000
        retry.backoff.ms = 100
        sasl.client.callback.handler.class = class
software.amazon.msk.auth.iam.IAMClientCallbackHandler
        sasl.jaas.config = [hidden]
        sasl.kerberos.kinit.cmd = /usr/bin/kinit
        sasl.kerberos.min.time.before.relogin = 60000
        sasl.kerberos.service.name = null
        sasl.kerberos.ticket.renew.jitter = 0.05
        sasl.kerberos.ticket.renew.window.factor = 0.8
        sasl.login.callback.handler.class = null
        sasl.login.class = null
        sasl.login.refresh.buffer.seconds = 300
        sasl.login.refresh.min.period.seconds = 60
        sasl.login.refresh.window.factor = 0.8
        sasl.login.refresh.window.jitter = 0.05
        sasl.mechanism = AWS_MSK_IAM
        security.protocol = SASL_SSL
        send.buffer.bytes = 131072
        session.timeout.ms = 10000
        ssl.cipher.suites = null
        ssl.enabled.protocols = [TLSv1.2, TLSv1.1, TLSv1]
        ssl.endpoint.identification.algorithm = https
        ssl.key.password = null
        ssl.keymanager.algorithm = SunX509
        ssl.keystore.location = null
        ssl.keystore.password = null
        ssl.keystore.type = JKS
        ssl.protocol = TLS
        ssl.provider = null
        ssl.secure.random.implementation = null
        ssl.trustmanager.algorithm = PKIX
        ssl.truststore.location = null
        ssl.truststore.password = null
        ssl.truststore.type = JKS
        value.deserializer = class
org.apache.kafka.common.serialization.StringDeserializer
```

```
[main] INFO org.apache.kafka.common.security.authenticator.AbstractLogin -
Successfully logged in.
[main] WARN org.apache.kafka.clients.consumer.ConsumerConfig - The
configuration 'batch.size' was supplied but isn't a known config.
[main] WARN org.apache.kafka.clients.consumer.ConsumerConfig - The
configuration 'acks' was supplied but isn't a known config.
[main] WARN org.apache.kafka.clients.consumer.ConsumerConfig - The
configuration 'buffer.memory' was supplied but isn't a known config.
[main] WARN org.apache.kafka.clients.consumer.ConsumerConfig - The
configuration 'retries' was supplied but isn't a known config.
[main] WARN org.apache.kafka.clients.consumer.ConsumerConfig - The
configuration 'linger.ms' was supplied but isn't a known config.
[main] INFO org.apache.kafka.common.utils.AppInfoParser - Kafka version: 2.2.1
[main] INFO org.apache.kafka.common.utils.AppInfoParser - Kafka commitId:
55783d3133a5a49a
[main] INFO org.apache.kafka.clients.consumer.KafkaConsumer - [Consumer
clientId=consumer-1, groupId=foobar] Subscribed to topic(s): ExampleTopic
[main] INFO org.apache.kafka.clients.Metadata - Cluster ID: lycb5mcpTfid-
G4Hg5CR7Q
[main] INFO org.apache.kafka.clients.consumer.internals.AbstractCoordinator -
[Consumer clientId=consumer-1, groupId=foobar] Discovered group coordinator b-
2.mskdemo.7zfurd.c7.kafka.us-east-1.amazonaws.com:9098(id: 2147483645 rack:
null)
[main] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator -
[Consumer clientId=consumer-1, groupId=foobar] Revoking previously assigned
partitions []
[main] INFO org.apache.kafka.clients.consumer.internals.AbstractCoordinator -
[Consumer clientId=consumer-1, groupId=foobar] (Re-)joining group
[main] INFO org.apache.kafka.clients.consumer.internals.AbstractCoordinator -
[Consumer clientId=consumer-1, groupId=foobar] (Re-)joining group
[main] INFO org.apache.kafka.clients.consumer.internals.AbstractCoordinator -
[Consumer clientId=consumer-1, groupId=foobar] Successfully joined group with
generation 1
[main] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator -
[Consumer clientId=consumer-1, groupId=foobar] Setting newly assigned
partitions: ExampleTopic-0, ExampleTopic-2, ExampleTopic-1, ExampleTopic-4,
ExampleTopic-3
[main] INFO org.apache.kafka.clients.consumer.internals.Fetcher - [Consumer
clientId=consumer-1, groupId=foobar] Resetting offset for partition
ExampleTopic-0 to offset 0.
[main] INFO org.apache.kafka.clients.consumer.internals.Fetcher - [Consumer
clientId=consumer-1, groupId=foobar] Resetting offset for partition
ExampleTopic-3 to offset 0.
[main] INFO org.apache.kafka.clients.consumer.internals.Fetcher - [Consumer
clientId=consumer-1, groupId=foobar] Resetting offset for partition
ExampleTopic-1 to offset 0.
[main] INFO org.apache.kafka.clients.consumer.internals.Fetcher - [Consumer
clientId=consumer-1, groupId=foobar] Resetting offset for partition
ExampleTopic-4 to offset 0.
[main] INFO org.apache.kafka.clients.consumer.internals.Fetcher - [Consumer
clientId=consumer-1, groupId=foobar] Resetting offset for partition
ExampleTopic-2 to offset 0.
Hello World!
```

It takes a few seconds to see the first sample message *Hello World*.

16. Navigate to the **Java producer session window** and enter further sample messages at the **>** prompt.
17. Enter

    It is a great day. and press **Enter**.

Expected output:

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************

sent message to topic:ExampleTopic partition:4  offset:3
>
```

18. Enter

    I got this. and press **Enter**.

Expected output:

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************

sent message to topic:ExampleTopic partition:4  offset:4
>
```

19. Switch to the **Java consumer session window**.

Expected output:

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************

It is a great day.
I got this.
```

You can see the new messages consumed in the *Java consumer session window*.

Note: You can close out of both the producer and consumer session windows by choosing **Terminate** in the top right corner of the session window.

Task complete: You have successfully published and consumed the messages from the topic using a console consumer.

Conclusion

You have done the following:

- Published to and consumed from an MSK cluster using IAM authenticated broker URLs with a Java demo producer and Java demo consumer.
- Learned about the IAM method to authenticate and authorize users of an MSK cluster.

End lab

Follow these steps to close the console and end your lab.

20. Return to the **AWS Management Console**.
21. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
22. Choose End lab and then confirm that you want to end your lab.