Lab 2: Message Fan-Out with Amazon EventBridge v1.0.2

Monday, March 29, 2021 10:55 AM

Lab 2: Message Fan-Out with Amazon EventBridge

1 hour 30 minutes Free ★★★★★ Rate Lab

aws training and certification

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

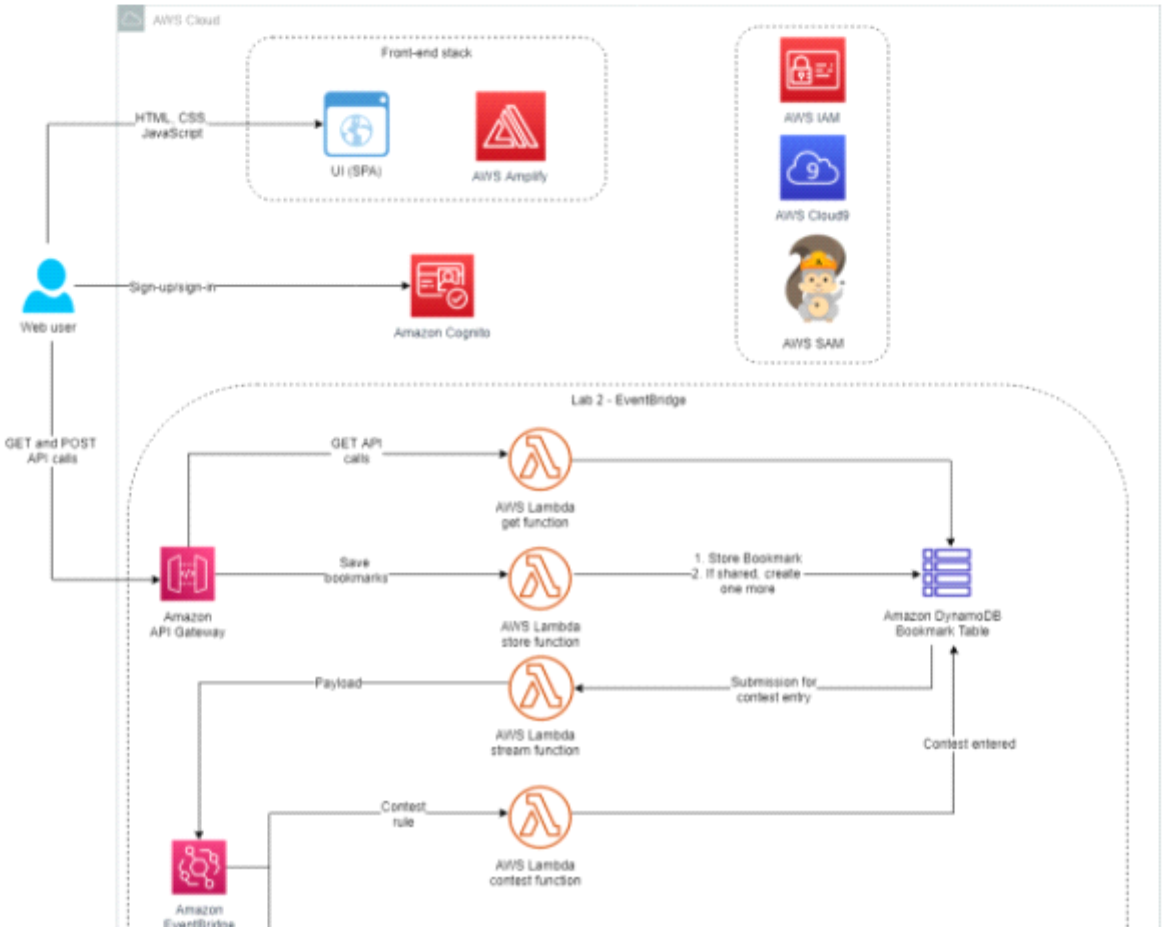
Overview

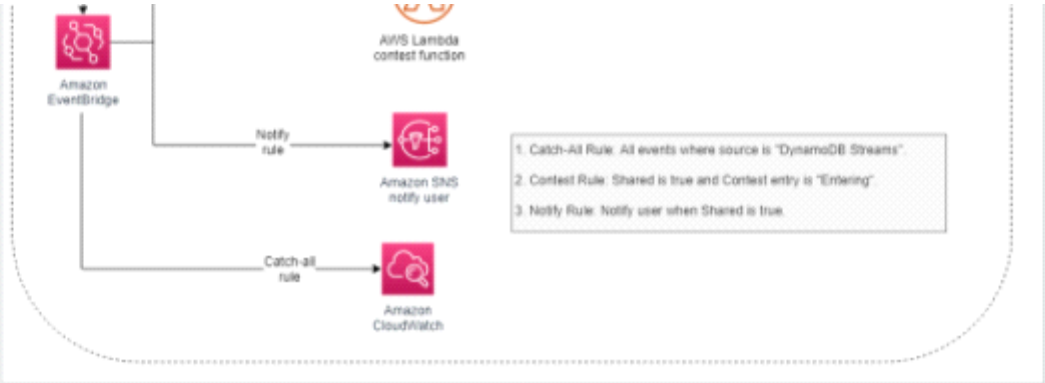
Overview

Now that you have verified your serverless proof of concept with the first iteration, you need to add the capability for sharing bookmarks with the team. To get the managers on board with your new application, you need to align to the existing knowledge base process. To get the team on board, your product manager has authorized a contest so that each new submission through the application enters the submitter into a drawing to win a prize.

In this lab, you add a new feature to the bookmark application to address the contest use case using Amazon EventBridge. When someone shares a bookmark, the bookmark application should automatically notify the mailbox that is being used to monitor submissions and enter the bookmark into the shareable bookmarks contest.

The following diagram shows the architecture components that have been or will be deployed in this lab.





Objectives

After completing this lab, you will be able to:

- Enable Amazon DynamoDB Streams as an event source for an AWS Lambda function that is invoked when new items are added to a DynamoDB table
- Configure an EventBridge event bus with a Lambda function as its event source and Lambda, Amazon Simple Notification Service (Amazon SNS), and Amazon CloudWatch as targets
- Configure EventBridge rules that route events to your targets based on the criteria that you specify
- Configure an SNS topic that notifies an email subscriber

Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi and Microsoft Windows, macOS X, or Linux (Ubuntu, SUSE, or Red Hat)
- For Microsoft Windows users, administrator access to the computer
- An internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported)

⚠ **Note** The lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the lab guide.

Duration

Duration

This lab requires approximately **60 minutes** to complete.

Start Lab

1. At the top of your screen, launch your lab by choosing **Start Lab**

   This starts the process of provisioning your lab resources. An estimated amount of time to provision your lab resources is displayed. You must wait for your resources to be provisioned before continuing.

   ℹ If you are prompted for a token, use the one distributed to you (or credits you have purchased).

2. Open your lab by choosing **Open Console**

   This opens an AWS Management Console sign-in page.

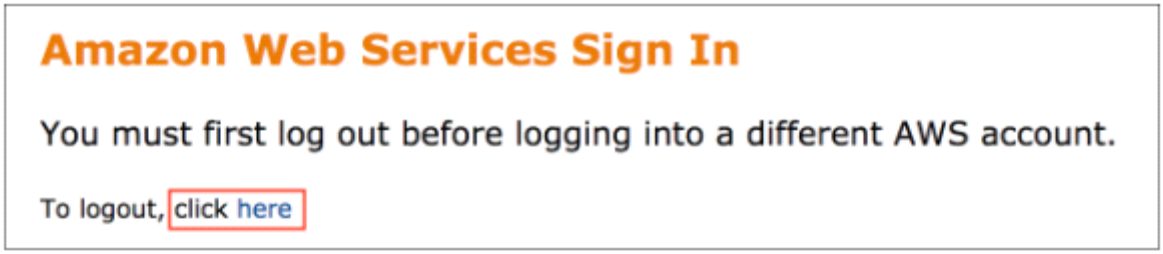
3. On the sign-in page, configure:

   - **IAM user name:** `awsstudent`
   - **Password:** Paste the value of **Password** from the left side of the lab page
   - Choose **Sign In**

   ⚠ **Do not change the Region unless instructed.**

Common Login Errors

Error: You must first log out



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

- Choose **click here**
- Close your browser tab to return to your initial lab window
- Choose Open Console again

Task 1: Understanding key services and application setup

In this task, you open the AWS Cloud9 integrated development environment (IDE) and download the application code. Once the code is unzipped, a startup script automates the launching of the front-end and backend application code. You then launch and deploy the application via the AWS Amplify console.

- **Amazon EventBridge** makes it easy to build event-driven applications because it takes care of event ingestion, delivery, security, authorization, and error handling for you. To achieve the promises of serverless technologies with event-driven architecture, such as being able to individually scale, operate, and evolve each service, the communication

technologies with event-driven architecture, such as being able to individually scale, operate, and evolve each service, the communication between the services must happen in a loosely coupled and reliable environment. Event-driven architecture is a fundamental approach for integrating independent systems or building up a set of loosely coupled systems that can operate, scale, and evolve independently and flexibly. In this lab, you use EventBridge to address the contest use case.

- **Amazon DynamoDB Streams** is an ordered flow of information about changes to items in a DynamoDB table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table.

- **Amazon Simple Notification Service (Amazon SNS)** is a fully managed messaging service for both system-to-system and app-to-person communication. It enables you to communicate between systems through publish/subscribe (pub/sub) patterns that enable messaging between decoupled microservice applications or to communicate directly to users via SMS, mobile push, and email. The system-to-system pub/sub functionality provides topics for high-throughput, push-based, many-to-many messaging. Using SNS topics, your publisher systems can fan out messages to a large number of subscriber systems or customer endpoints, including Amazon Simple Queue Service (Amazon SQS) queues, Lambda functions, and HTTP and HTTPS, for parallel processing. The app-to-person messaging functionality enables you to send messages to users at scale using either a pub/sub pattern or direct-publish messages using a single API.

4. Choose **Services** and select **Cloud9**.

5. For the **BookmarkAppDevEnv** environment, choose **Open IDE**

   Within a few seconds, the AWS Cloud9 environment launches.

6. In the AWS Cloud9 terminal, run the following commands. These commands download and run the startup script, which contains the application code:

download and run the startup script, which contains the application code:

```
wget https://us-west-2-tcprod.s3-us-west-
2.amazonaws.com/courses/ILT-TF-200-SVDVSS/v1.0.2/lab-2-
EventBridge/scripts/app-code.zip
unzip app-code.zip
cd app-code
chmod +x startupscript.sh
./startupscript.sh
```

⚠ **Note** The script takes a couple of minutes to run. Once it is finished, you will deploy your bookmark application through Amplify. **Be sure to let the script finish running before moving on to the next step.**

What the script is doing: This script modifies the **samconfig.toml** file within the backend portion of the application code. The script replaces values such as AWS Region, stack name, and role Amazon Resource Name (ARN). Next, the script updates the **aws-exports.js** file with the Amazon Cognito metadata that was launched in the lab's AWS CloudFormation template. The script then runs the build, deploys the bookmark application, and uploads the **app.zip** file to the **samserverless** bucket.

7. In the AWS Management Console, choose **Services** and select **AWS Amplify**.

8. Choose the menu ☰ icon at the top-left corner of the page, and then choose **All apps**.

9. Choose **New app** and select **Host web app** from the dropdown list.

   You can choose either **New app** button on the page.

10. Select **Deploy without Git provider**, and then choose **Continue**

11. On the **Manual deploy** page, configure the following information:

    - **App name:** Enter `BookmarkApp`

11. On the **Manual deploy** page, configure the following information:

    - **App name:** Enter `BookmarkApp`
    - **Environment name:** Enter `dev`
    - **Method:** Select **Amazon S3**
    - **Bucket:** Select the bucket name that includes **samserverless**
    - **Zip file:** Select **app.zip** (When the **Bucket** is selected, this dropdown menu auto-populates.)

12. Choose **Save and deploy**

13. Once you see the message **Deployment successfully completed** in the Amplify console, choose the **Domain** URL to open the bookmark application.

14. From the bookmark application page, choose **Create account**

15. Fill in the fields with your information, and choose **CREATE ACCOUNT**

    Note Leave this browser tab open.

Task 2: Enabling DynamoDB Streams and setting up a Lambda trigger

In this task, you enable DynamoDB Streams on **bookmarksTable**.

16. From the AWS Management Console, choose **Services** and select and open **DynamoDB** in a new browser tab.

17. On the left side of the DynamoDB dashboard, choose **Tables**.

18. Choose the table with the **Name** that includes **bookmarksTable**.

18. Choose the table with the **Name** that includes **bookmarksTable**.

19. Under **DynamoDB stream details**, choose Manage DynamoDB stream

20. Select **New and old images**, and choose Enable

    DynamoDB Streams helps ensure that each stream record appears exactly once in the stream. Also, for each item that is modified in a DynamoDB table, the stream records appear in the same sequence as the actual modifications to the item.

    Now, create the Lambda function that is invoked by the DynamoDB table and alerts the EventBridge event bus.

21. From the AWS Management Console, choose Services and select and open **Lambda** in a new browser tab.

22. On the left side of the Lambda dashboard, choose **Functions**.

23. Choose Create function

24. On the **Create function** page, configure the following information:

    - **Function name:** Enter `StreamTrigger`
    - **Runtime:** Select **Node.js 14.x**

25. Under **Permissions**, expand the **Change default execution role** section, and then select **Use an existing role**.

26. From the **Existing Role** dropdown menu, choose the role name that includes **EventBridgeLambdaRole**.

27. Choose Create function

28. Choose + Add trigger

29. On the **Add trigger** page, configure the following information:

29. On the **Add trigger** page, configure the following information:

    - In the **Trigger configuration** section, from the **Select a trigger** dropdown menu, choose **DynamoDB**.
    - In the **DynamoDB table** search box, select the table with **bookmarksTable** in the name.
    - Decrease the **Batch size** to `5`

30. Leave the rest of the defaults the same, and then choose **Add**

31. Choose the **Code** tab to bring up the function again.

32. In the **Code source** section, select **index.js**, open the context (right-click) menu and choose **Open**.

33. Delete the existing code and paste the following code:

```
const EventBridge = require('aws-sdk/clients/eventbridge')
const ev = new EventBridge();

exports.handler = async (event) => {
    console.log(JSON.stringify(event, null, 2));
    try
    {
        for(let i=0; i< event.Records.length; i++) {
            const record = event.Records[i]
            console.log(record.eventID);
            console.log(record.eventName);
            if(record.eventName === 'INSERT' || record.eventName
=== 'MODIFY') {
                console.log('DynamoDB Record: %j',
record.dynamodb);
                console.log('share flag:',
record.dynamodb.NewImage.shared.BOOL);
                console.log('contest value: ',
record.dynamodb.NewImage.contest.S);

                var pk = record.dynamodb.NewImage.id.S;
                var sharedFlag =
record.dynamodb.NewImage.shared.BOOL;
                var contestValue =
```

```
                var sharedFlag =
record.dynamodb.NewImage.shared.BOOL;
                var contestValue =
record.dynamodb.NewImage.contest.S;

                const bookmarkDetails = {
                    id: pk,
                    shared: sharedFlag,
                    contest: contestValue,
                    payload: record.dynamodb.NewImage
                }

                const params = {
                Entries: [
                  {
                    Source: 'DynamoDB Streams',
                    DetailType: 'Shared Bookmarks',
                    EventBusName: 'bookmarks-bus',
                    Detail: JSON.stringify(bookmarkDetails)
                  }
                ]
              };
              const response =  await
ev.putEvents(params).promise();
              console.log("response:", response);
              //We can optimize the code by calling the putEvents
outside of the loop with promise all option. where all the records
              //will put in the bus in parallel.
            }
        }
    } catch (error) {
          throw new Error(JSON.stringify(error));
    }

}
```

34. Choose **Deploy**

You should see a message that says **Changes deployed**

This code sends updates to EventBridge only if there is an UPDATE or INSERT event, kicking off the next phase in the event-driven architecture.

Task 3: Subscribing to bookmark contest notifications

In this task, you create and subscribe to an SNS topic that sends notifications to you and your manager when a bookmark has been shared.

35. From the AWS Management Console, choose **Services** and select and open **Simple Notification Service** in a new browser tab.

36. On the right side of the page, in the **Create topic** box, for **Topic name**, enter `BookmarkTopic`

37. Choose **Next step**

38. Confirm that **Type** is set to *Standard* and choose **Create topic**

39. Choose **Create subscription**

40. On the **Create subscription** page, configure the following information:

    - **Topic ARN:** Confirm that it is the ARN with **BookmarkTopic** in the name
    - **Protocol:** Select **Email**
    - **Endpoint:** Enter a valid email address

41. Choose **Create subscription**

    After a few moments, you should receive an email to confirm the subscription. You must confirm the subscription to activate it.

42. To confirm the subscription, choose the **Confirm subscription** link in the email that you receive.

12. To confirm the subscription, choose the **Confirm subscription** link in the email that you receive.

    💬 **Note** For this scenario, this email address is considered the email address for the manager who will receive the notifications for the bookmarking contest.

    You can move onto the next task as you await the Amazon SNS email confirmation.

Task 4: Setting up an event bus and configuring rules

In this task, you create rules in EventBridge to attach to an event bus. This event bus receives events from a Lambda stream trigger and then matches them to the applicable rules.

Rules watch for specific types of events. When a matching event occurs, the event is routed to the targets that are associated with the rule. A rule can be associated with one or more targets.

13. In the AWS Management Console, choose **Services** and select **Amazon EventBridge**.

14. On the left side of the page, choose the menu ☰ icon.

15. Choose **Event buses**.

16. Choose **Create event bus**

17. On the **Create event bus** page, in the **Name** field, enter `bookmarks-bus`

47. On the **Create event bus** page, in the **Name** field, enter `bookmarks-bus`

48. Choose **Create**

49. On the left side of the page, choose the menu ☰ icon.

50. Choose **Rules**.

51. From the **Event bus** dropdown menu, select **bookmarks-bus**.

    The first rule that you need to create is the **catch-all-rule**. This rule sends the event payload to Amazon CloudWatch Logs after being invoked by the **StreamTrigger** Lambda function and passing through the event bus.

52. Choose **Create rule**

53. On the **Create rule** page, in the **Name and description** section, configure the following information:

    - **Name:** Enter `catch-all-rule`
    - **Description:** Enter `catch-all rule for cloudwatch logs`

54. In the **Define pattern** section, configure the following information:

    - Select **Event pattern**.
    - For **Event matching pattern**, select **Custom pattern**.
    - In the **Event pattern** code box, copy and paste in the following code:

```
{
  "source": [
    "DynamoDB Streams"
  ],
  "detail-type": [
    "Shared Bookmarks"
  ]
}
```

```
}
```

55. Choose **Save**

    Note EventBridge rules use event patterns to match AWS events on an event bus. When a pattern matches, the rule routes that event to a target. This event pattern is using DynamoDB Streams as the source and identifying the **Shared Bookmarks** value as the detail to invoke the CloudWatch catch-all log.

56. In the **Select event bus** section, configure the following information:

    - **Select an event bus for this rule:** Select **Custom or partner event bus**
    - Make sure that **bookmarks-bus** is selected in the dropdown menu.

57. In the **Select targets** section, configure the following information:

    - **Target:** Select **CloudWatch log group**
    - **/aws/events/:** Enter `catch-all`

58. Choose **Create**

    The next rule that you need to create is the **Notification rule**. When invoked, this rule sends a notification via Amazon SNS to the email address that you used earlier to register with the bookmarks site.

59. Choose **Create rule**

60. On the **Create rule** page, in the **Name and description** section, configure the following information:

    - **Name:** Enter `notify-rule`
    - **Description:** Enter `rule to invoke SNS`

61. In the **Define pattern** section, configure the following information:

    - Select **Event pattern**.

- Select **Event pattern**.
- For **Event matching pattern**, select **Custom pattern**.
- In the **Event pattern** code box, copy and paste in the following code:

```
{
  "source": [
    "DynamoDB Streams"
  ],
  "detail-type": [
    "Shared Bookmarks"
  ],
  "detail": {
    "shared": [
      true
    ],
    "contest": [
      {
        "anything-but": [
          "Entering"
        ]
      }
    ]
  }
}
```

52. Choose Save

Note This event pattern is again using DynamoDB Streams as the source and **Shared Bookmarks** as the detail-type. The third level to this pattern is **detail: shared: true** along with **contest: anything-but:Entering**, which sends an Amazon SNS message when someone shares a bookmark. The condition without **Entering** will send only one message that the bookmark has been entered into the contest.

53. In the **Select event bus** section, configure the following information:

- For **Select an event bus for this rule**, select **Custom or partner event bus**.
- Make sure that **bookmarks-bus** is selected in the dropdown menu.

54. In the **Select targets** section, configure the following information:

- Make sure that **bookmarks-bus** is selected in the dropdown menu.

54. In the **Select targets** section, configure the following information:

    - **Target:** Select **SNS topic**
    - **Topic:** Select **BookmarkTopic**

55. Choose **Create**

    The final rule that you need to create is the **contest-rule**. This rule invokes the **Contest** Lambda function, which adds the relevant item into the **sam-bookmark-app-bookmarksTable**.

56. Choose **Create rule**

57. On the **Create rule** page, in the **Name and description** section, configure the following information:

    - **Name:** Enter `contest-rule`
    - **Description:** Enter `rule to invoke contest function`

58. In the **Define pattern** section, configure the following information:

    - Select **Event pattern**.
    - For **Event matching pattern**, select **Custom pattern**.
    - In the **Event pattern** code box, copy and paste in the following code:

```
{
  "source": [
    "DynamoDB Streams"
  ],
  "detail-type": [
    "Shared Bookmarks"
  ],
  "detail": {
    "shared": [
      true
    ],
  "contest": [
      "Entering"
```

```
    "contest": [
        "Entering"
    ]
  }
}
```

59. Choose **Save**

Note This event pattern uses the same structure and details but invokes the Lambda contest function instead of the Amazon SNS message.

70. In the **Select event bus** section, configure the following information:

- For **Select an event bus for this rule**, select **Custom or partner event bus**.
- Make sure that **bookmarks-bus** is selected in the dropdown menu.

71. In the **Select targets** section, configure the following information:

- **Target:** Select **Lambda function**
- **Function:** Select the function that contains **contest** in the name

72. Choose **Create**

Task 5: Testing EventBridge rules

In this task, you add and share bookmarks. This kicks off EventBridge and the corresponding rules.

Note Before continuing, make sure that you have confirmed your email registration and email subscriptions from earlier in the lab.

73. Go to the browser tab with the bookmark application

registration and email subscriptions from earlier in the lab.

73. Go to the browser tab with the bookmark application.

74. Choose the plus **+** icon at the top-right corner of the page.

75. On the **Add New Bookmark** page, add and share a bookmark of your choice. Make sure that the **Share Bookmark** toggle is set to **On**.

76. Choose **ADD BOOKMARK**.

    Note When a user shares a bookmark, the bookmark application navigates to the shared bookmark page that lists all of the bookmarks that have been shared. The bookmark application also updates the value of the **shared** column for that particular row from false to true in the **bookmarks-app-bookmarksTable** in DynamoDB.

77. Check your email for the Amazon SNS notification.

    Note Within the payload of the Amazon SNS email, you should see **"shared":true**. This email confirms that the SNS topic and **notify-rule** worked as intended.

78. From the AWS Management Console, choose **Services** and select **CloudWatch**.

79. On the left side of the page, choose **Log groups**.

80. Locate and open the **/aws/events/catch-all** log group.

81. Open the most recent **Log stream**.

82. Next to the timestamp for the log stream, expand the message.

    Here you can see important details from the catch-all rule that you created earlier, such as time, ARN of the user who added the bookmark, account ID, and username.

and username.

33. From the AWS Management Console, choose **Services** and select **DynamoDB**.

34. Choose the ▸ icon in the left navigation pane.

35. Choose **Tables**.

36. Choose the **sambookmark-app-bookmarksTable** table.

    This is the **sambookmark-app-bookmarksTable** table that was created when deploying your backend code via the AWS Serverless Application Model (AWS SAM).

37. Choose the **Items** tab.

    Here are the bookmarks that have been added. Notice the **shared** column, which shows either true or false, and the **contest** column, which shows **Entered**.

    This information confirms that all three EventBridge rules worked and that the event-driven architecture was a success!

Conclusion

👍 Congratulations! You now have successfully:

- Enabled DynamoDB Streams as an event source for a Lambda function that is invoked when new items are added to a DynamoDB table
- Configured an EventBridge event bus with a Lambda function as its event

that is invoked when new items are added to a DynamoDB table
- Configured an EventBridge event bus with a Lambda function as its event source and Lambda, Amazon SNS, and CloudWatch as targets
- Configured EventBridge rules that route events to your targets based on the criteria that you specify
- Configured an SNS topic that notifies an email subscriber

End Lab

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

38. Return to the AWS Management Console.

39. On the navigation bar, choose **awsstudent@<AccountNumber>**, and then choose **Sign Out**.

40. Choose **End Lab**

41. Choose **OK**

42. (Optional):

    - Select the applicable number of stars ☆
    - Type a comment
    - Choose **Submit**

        - 1 star = Very dissatisfied
        - 2 stars = Dissatisfied
        - 3 stars = Neutral
        - 4 stars = Satisfied

- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You may close the window if you don't want to provide feedback.
For more information about AWS Training and Certification, see *http://aws.amazon.com/training/*.

Your feedback is welcome and appreciated.

If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form*.

Additional resources

- For more information about EventBridge, see https://pages.awscloud.com/Deep-Dive-on-Amazon-EventBridge_2019_0919-SRV_OD.html.
- For more information about DynamoDB Streams, see https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Str