

## Lab 3: Workflow Orchestration Using AWS Step Functions v1.0.2

Monday, March 29, 2021 10:56 AM

# Lab 3: Workflow Orchestration Using AWS Step Functions

1 hour 30 minutes Free ★★★★★ [Rate Lab](#)



© 2021 Amazon Web Services, Inc. and its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Corrections, feedback, or other questions? Contact us at [AWS Training and Certification](#).

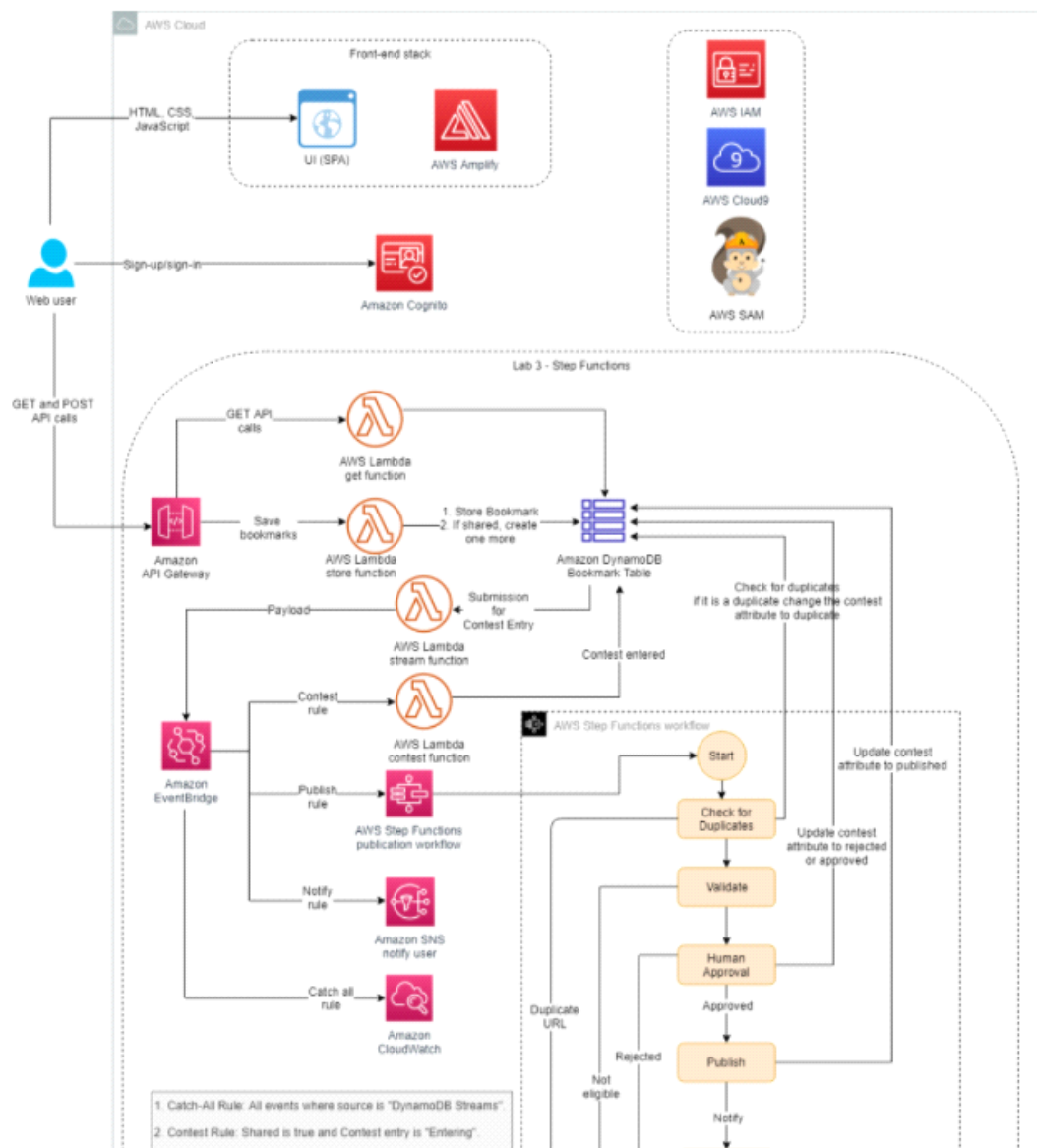
## Overview

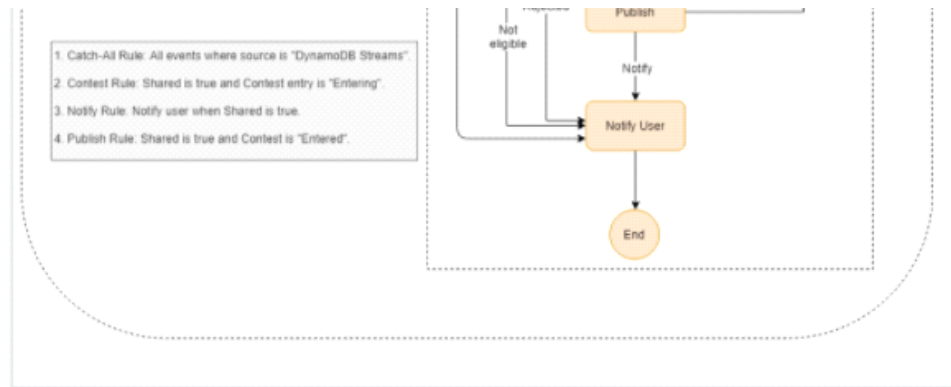
You deployed your serverless bookmark application, and then updated it to support bookmark sharing, routing of requests to the shared mailbox, and a contest feature. The managers who review the knowledge base submissions recognize that submissions are going to increase a lot with the application, so they want you to scale their submissions process by automating as much of it as possible. As part of their manual review, the managers check to make sure the submission isn't a duplicate, validate the URL and make sure it meets eligibility requirements to be

scale their submissions process by automating as much of it as possible. As part of their manual review, the managers check to make sure the submission isn't a duplicate, validate the URL and make sure it meets eligibility requirements to be shared, and then publish the resource. Then, the managers let the submitter know whether or not the item was added. When the item is a duplicate or is invalid, the submitter does not get a contest entry for that item, so the managers need to inform the contest administrator of duplicates and bad submissions. Given the variety of sources and type of materials that might be shared, the managers still want to maintain a manual approval step so that they can review the material and make sure it follows practices and technologies that they want to promote.

In this lab, you use AWS Step Functions to incorporate a workflow to automatically evaluate and publish new submissions, while still including a manual approval step as part of the process.

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





This lab uses the following software stack:

- AWS Amplify
- AWS Serverless Application Model (AWS SAM)
- Amazon Cognito
- Vue JavaScript framework
- AWS Cloud9
- Swagger API
- Amazon DynamoDB
- Amazon EventBridge
- Amazon Simple Notification Service (Amazon SNS)
- AWS Step Functions
- AWS Lambda
- Amazon CloudWatch
- Amazon API Gateway

## Objectives

After completing this lab, you will be able to:

- Configure EventBridge to target a Step Functions workflow
- Use a Step Functions Standard Workflow to orchestrate tasks
- Use Lambda for tasks within a Step Functions state machine

## Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi and Microsoft Windows, macOS, 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)
- A text editor

versions of Internet Explorer are not supported)

- A text editor

**⚠ 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

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**

**Amazon Web Services Sign In**

## Amazon Web Services Sign In

You must first log out before logging into a different AWS account.

To logout, [click here](#)

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 the environment setup

In this task, you look at the different AWS services that are used in this lab. The services include Step Functions and Amazon SNS.

- **AWS Step Functions** is a serverless function orchestrator that makes it easy to sequence Lambda functions and multiple AWS services into business-critical applications. Through its visual interface, you can create and run a series of checkpointed and event-driven workflows that maintain the application state. The output of one step acts as input to the next. Each step in your application runs in order, as defined by your business logic.

Orchestrating a series of individual serverless applications, managing retries, and debugging failures can be challenging. As your distributed applications become more complex, the complexity of managing them also grows. Step Functions automatically manages error handling, retry logic, and state. With its built-in operational controls, Step Functions manages sequencing, error handling, retry logic, and state, removing a significant operational burden from your team.


- **Amazon Simple Notification Service (Amazon SNS)** is a fully managed messaging service for both system-to-system and app-to-person (A2P) communication. The service enables you to communicate between systems through publish/subscribe (pub/sub) patterns that enable messaging between



communication. The service 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 Amazon 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/S, for parallel processing. The A2P 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.

In this task, you use the AWS Cloud9 integrated development environment (IDE) and download the application code. The front-end and backend code combine the workflows and resources from Labs 1 and 2. The code also contains new Lambda functions, a new API Gateway endpoint, and an SNS topic for the manual approval of the bookmark publishing process.

After downloading the code, you launch and deploy the application via the Amplify console.

4. In the AWS Management Console, on the **Services** ▾ menu, choose **Cloud9**.
5. For the **BookmarkAppDevEnv** environment, choose [Open IDE](#) 

Within a few seconds, the AWS Cloud9 environment launches.

6. In the terminal pane, run the following commands to download the application code and run the startup script:

```
wget https://us-west-2-tcprod.s3-us-west-2.amazonaws.com/courses/ILT-TF-200-SVDVSS/v1.0.2/lab-3-Step-Functions/scripts/app-code.zip
unzip app-code.zip
cd app-code
chmod +x startupscript.sh
./startupscript.sh
```

The script takes a couple of minutes to run. When it is finished, you will deploy your bookmark application through Amplify.

## What is the script doing?

- This script modifies the `serverless.yml` file within the backend portion of the

## what is the script 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).
- The script then updates the **aws-exports.js** file with the Amazon Cognito metadata that was launched in the lab AWS CloudFormation template.
- As part of the AWS SAM deployment, a new API Gateway resource and a few Lambda functions are deployed. These functions are needed for the Step Functions workflow.
- The script then runs **npm build**, deploys the bookmark application, and uploads the **app.zip** file to the **samserverless** bucket.

## Review the resources that have been deployed

7. Return to the AWS Management Console tab.

8. On the **Services** menu, choose **Amazon EventBridge**.

9. In the left navigation pane, choose **Rules**.

**Note** To expand the left navigation pane, choose the menu icon (≡).

10. From the **Event bus** dropdown menu, choose **bookmarks-bus**.

Review the rules that have been created under **bookmarks-bus**.

11. On the **Services** menu, choose **Lambda**.

Review the functions with **bookmark** in their names. These functions were deployed to create and share the bookmark URLs.

**Note** To review the code for a function, choose the name of the function, and scroll down to the **Code source** section.

12. On the **Services** menu, choose **API Gateway**.

13. In the **APIs** list, choose the name of **Bookmark App**.

14. In the **Resources** pane, under the **/approval** endpoint, choose **GET**.

This endpoint invokes a Lambda function that handles the human approval process.

This endpoint invokes a Lambda function that handles the human approval process.

15. In the left navigation pane, choose **Stages**.
16. In the **Stages** pane, expand the ▶ **dev** stage.
17. Under the **bookmarks/approval** endpoint, choose **GET**.
18. Copy the **Invoke URL** value to a text editor to use later. You will use this value later in the Step Functions state machine.

**Note** The endpoint URL should end with **/bookmarks/approval**.

19. On the **Services** ▾ menu, choose **Simple Notification Service**.
20. In the left navigation pane, choose **Topics**.

Review the topics that have been deployed.

## Task 2: Creating SNS subscriptions, updating the EventBridge rule, and deploying the bookmark application

In this task, you create subscriptions for the SNS topics, update the EventBridge rule to create CloudWatch logs, and deploy the front-end application using Amplify.

### Create subscriptions for the SNS topics

21. In the Amazon SNS console, under **Topics**, choose the name of the **BookmarkTopic** topic.
22. Choose **Create subscription**
23. In the **Details** section, configure the following:
  - **Protocol:** Choose **Email**
  - **Endpoint:** Enter an email address where you can receive messages



- **Protocol:** Choose **Email**
- **Endpoint:** Enter an email address where you can receive messages

24. Choose **Create subscription**

25. Check your email and confirm the subscription.

**Note** It may take a few minutes for the subscription confirmation to arrive. This subscription sends an email with the contents of a bookmark as it is entered in the contest.

26. In the left navigation pane, choose **Topics**.

27. Choose the name of the **ContestTopic** topic.

28. Copy the **ARN** of the **ContestTopic** to a text editor. You will use this value later in the Step Functions state machine.

29. Choose **Create subscription**

30. In the **Details** section, configure the following:

- **Protocol:** Choose **Email**
- **Endpoint:** Enter an email address where you can receive messages

31. Choose **Create subscription**

32. Check your email and confirm the subscription.

**Note** It may take a few minutes for the subscription confirmation to arrive. This subscription sends an email with a link for human approval or rejection of a bookmark.

## Update the catch-all-rule to create CloudWatch logs

33. On the **Services** menu, choose **Amazon EventBridge**.

34. In the left navigation pane, choose **Rules**.

35. From the **Event bus** dropdown menu, choose **bookmarks-bus**.

36. Choose the **catch-all-rule**.

36. Choose the **catch-all-rule**.

37. Choose **Edit**.

38. Scroll to the bottom of the page, and choose **Update**

**Note** Updating this rule is necessary to create the permissions to allow EventBridge to write logs to the `\aws\events\catch-all` CloudWatch log group whenever the catch-all-rule is invoked.

## Deploy the front-end application using Amplify

39. On the **Services** menu, choose **AWS Amplify**.

40. In the left navigation pane, choose the menu icon (≡), and then choose **All apps**.

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

42. Choose **Deploy without Git provider**, and then choose **Continue**

43. On the **Manual deploy** page, configure the following:

- **App name:** Enter `BookmarkApp`
- **Environment name:** Enter `dev`
- **Method:** Choose **Amazon S3**
- **Bucket:** Choose `xxxx-samserverless-xxxx`
- **Zip file:** Choose `app.zip`

44. Choose **Save and deploy**

## Create a user, add and share a bookmark, and check the EventBridge rules

45. When the deployment has successfully completed, choose the **Domain** URL in the middle of the Amplify console page.


46. To create an account for the bookmark application, choose **Create account**.

47. Fill in the provided fields with your information and choose **CREATE ACCOUNT**

48. To add a bookmark, choose the plus **+** icon at the top right of the page.

48. To add a bookmark, choose the plus **+** icon at the top right of the page.
49. Enter the **Name**, **Description**, and **Bookmark URL** of any website. Ensure the **Share Bookmark** toggle is set to **On**.
50. Choose **ADD BOOKMARK**.

Now, check that the CloudWatch logs are configured for the **catch-all** EventBridge rule.



51. Return to the AWS Management Console tab.
52. On the **Services**  menu, choose **CloudWatch**.
53. In the left navigation pane, choose **Log groups**.
54. In the log groups list, choose the name of the **/aws/events/catch-all** log group.

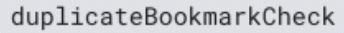
Notice the log streams listed for this log group. This shows that logs are being generated for the **catch-all** EventBridge rule.

Check your email for an Amazon SNS notification with the details of the bookmark that you entered in the bookmark contest.

## Task 3: Deploying the Lambda functions for the Step Functions state machine

In this task, you create a Lambda function that the Step Functions state machine needs. You also review the remaining Lambda functions that were set up with the AWS SAM deploy process.

55. In the AWS Management Console, on the **Services**  menu, choose **Lambda**.
56. Choose **Create function** .
57. Choose **Author from scratch**, and configure the following:

- **Function name:** Enter `duplicateBookmarkCheck` 
- **Runtime:** Choose **Node.js 14.x**

- **Function name:** Enter `duplicateBookmarkCheck`
- **Runtime:** Choose **Node.js 14.x**

58. Expand ► **Change default execution role**, and configure:

- **Execution role:** Choose **Use an existing role**
- **Existing role:** Choose **xxxx-SamDeploymentRole-xxxx**

59. Choose **Create function**

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

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

```
const AWS = require('aws-sdk');
const docClient = new AWS.DynamoDB.DocumentClient();
const dynamodb = new AWS.DynamoDB();

exports.handler = async message => {
  console.log(message);
  let bookmark = message;
  const bookmarkDetails = JSON.stringify(bookmark);
  console.log("bookmarkDetails are "+bookmarkDetails);
  const bookmarkItem = JSON.parse(bookmarkDetails);
  console.log("bookmarkItem "+bookmarkItem);
  console.log("url is "+bookmarkItem.detail.payload.bookmarkUrl.S);
  var exists = false;
  try{
    if(message != null)
    {
      var params = {
        TableName: process.env.TABLE_NAME,
        IndexName: process.env.INDEX_NAME,
        KeyConditionExpression: "bookmarkUrl = :keyurl",
        ExpressionAttributeValues: {
          ":keyurl": {"S":
bookmarkItem.detail.payload.bookmarkUrl.S}
        }
      };
      console.log("exists "+exists);
      var result = await dynamodb.query(params).promise();
      console.log("result is "+JSON.stringify(result.Items));
      var data = JSON.parse(JSON.stringify(result.Items));

      data.forEach(function(item) {
        console.log("db username", item.username.S+"
"+bookmarkItem.detail.payload.username.S);
        if (item.username.S !=
bookmarkItem.detail.payload.username.S)
          exists = true;
      });
    }
  }
```

```

bookmarkItem.detail.payload.userName);
        exists = true;
    });

    console.log(exists);
    if (exists === true)
    {
        console.log("in here");
        var updateParams = {
            TableName: process.env.TABLE_NAME,
            Key:{
                "id": bookmarkItem.detail.payload.id.S
            },
            UpdateExpression: "set contest=:c",
            ExpressionAttributeValues:{
                ":c": "duplicate"
            },
            ReturnValues:"UPDATED_NEW"
        };
        await docClient.update(updateParams, function(err, data) {
            if (err) {
                console.log("Unable to update item. Error JSON:",
                    JSON.stringify(err, null, 2));
            }
            else {
                console.log("UpdateItem succeeded:", JSON.stringify(data,
                    null, 2));
            }
        }).promise();
        return "Duplicate";
    }
}

catch(e){
    console.log(e);
}

return "NotDuplicate";
};

```

62. Choose **Deploy**

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

63. Choose the **Configuration** tab to configure the environment variables.

64. In the left navigation pane, choose **Environment variables**.

65. In the **Environment variables** section, choose **Edit**

66. In the **Edit environment variables** page, configure the following details:

- Choose **Add environment variable**



- Choose **Add environment variable**
  - **Key:** Enter `INDEX_NAME`
  - **Value:** Enter `bookmarkUrl-index`
- Choose **Add environment variable**
  - **Key:** Enter `TABLE_NAME`
  - **Value:** Run the following command in the AWS Cloud9 terminal to retrieve the bookmarks table name. Copy the table name into the **Value** field.

```
aws dynamodb list-tables
```

67. Choose **Save**

**Note** This Lambda function is the first function that runs as part of the Step Functions state machine. The function checks whether a new bookmark URL is a duplicate.

68. In the breadcrumbs at the top left of the page, choose **Functions**.

Review the following functions, which are used in the Step Functions workflow:

- **validateURL:** This function validates if the user entered a valid URL.
- **userApprovalEmail:** This function generates a URL for human approval or rejection and notifies the **ContestTopic**.
- **userApproval:** This function gets the approve or reject value after the user chooses the URL generated from the **userApprovalEmail** function.
- **publishApproval:** If the URL is approved, this function updates the DynamoDB table with the approved value.

**Note** To see all of these functions, you may need to go to the second page of the functions list.

## Task 4: Setting up the state machine

## Task 4: Setting up the state machine workflow invoked by the EventBridge rule

In this task, you set up a Step Functions state machine and an EventBridge rule. The EventBridge rule will invoke the state machine that you create.

69. In the AWS Management Console, on the **Services** menu, choose **Step Functions**.

70. In the left navigation pane, choose the menu icon (☰), and then choose **State machines**.

71. Choose **Create state machine**

72. Configure the following:

- Choose **Author with code snippets**
- **Type**: Choose **Standard**

73. In the **Definition** section, replace the sample code with the following code:

```
{
  "Comment": "Publish Rule workflow",
  "StartAt": "CheckDuplicates",
  "States": {
    "CheckDuplicates": {
      "Type": "Task",
      "Resource": "(duplicateBookmarkCheckArn)",
      "InputPath": "$",
      "ResultPath": "$.extractedMetadata",
      "Catch": [
        {
          "ErrorEquals": [ "States.ALL" ],
          "Next": "NotifyUser"
        }
      ],
      "Next": "DuplicateChoiceState"
    },
    "DuplicateChoiceState": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.extractedMetadata",
          "StringEquals": "NotDuplicate",
          "Next": "NotDuplicate"
        },
        {
          "Variable": "$.extractedMetadata",
          "StringEquals": "Duplicate",
          "Next": "Duplicate"
        }
      ]
    }
  }
}
```

```

        "Variable": "$.extractedMetadata",
        "StringEquals": "Duplicate",
        "Next": "Duplicate"
    }
}
},
"NotDuplicate": {
    "Type": "Pass",
    "Next": "ValidateURL"
},
"Duplicate": {
    "Type": "Pass",
    "Next": "NotifyUser"
},
"ValidateURL": {
    "Type": "Task",
    "Resource": "(validateURLArn)",
    "InputPath": "$",
    "ResultPath": "$.extractedMetadata",
    "Catch": [
        {
            "ErrorEquals": [ "States.ALL" ],
            "Next": "NotifyUser"
        }
    ],
    "Next": "ValidateChoiceState"
},
"ValidateChoiceState": {
    "Type": "Choice",
    "Choices": [
        {
            "Variable": "$.extractedMetadata",
            "StringEquals": "Valid",
            "Next": "UserApprovalEmail"
        },
        {
            "Variable": "$.extractedMetadata",
            "StringEquals": "Invalid",
            "Next": "NotifyUser"
        }
    ]
},
"UserApprovalEmail": {
    "Type": "Task",
    "Resource": "arn:aws:states:::lambda:invoke.waitForTaskToken",
    "InputPath": "$",
    "ResultPath": "$.extractedMetadata",
    "Parameters": {
        "FunctionName": "(userApprovalEmailArn)",
        "Payload": {
            "ExecutionContext.$": "$$",
            "APIGatewayEndpoint": "(APIGatewayEndpointURL)"
        }
    },
    "Next": "ManualApprovalChoiceState"
},
"ManualApprovalChoiceState": {
    "Type": "Choice",
    "Choices": [

```

```

    },
    "ManualApprovalChoiceState": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.extractedMetadata.Status",
          "StringEquals": "Approved",
          "Next": "PublishApproval"
        },
        {
          "Variable": "$.extractedMetadata.Status",
          "StringEquals": "Rejected",
          "Next": "NotifyUser"
        }
      ]
    },
    "PublishApproval": {
      "Type": "Task",
      "Resource": "(publishApprovalArn)",
      "InputPath": "$",
      "ResultPath": "$.extractedMetadata",
      "End": true
    },
    "NotifyUser": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "InputPath": "$",
      "Parameters": {
        "TopicArn": "(ContestTopicArn)",
        "Message.$": "$",
        "MessageAttributes": {
          "msg": {
            "DataType": "String",
            "StringValue": "additional instructions!"
          }
        }
      },
      "End": true
    }
  }
}

```

74. In the AWS Cloud9 IDE, choose the **AWS** tab in the left navigation pane to view the **AWS: Explorer**.

**Note** The resources deployed by the CloudFormation template as part of the build process are listed here. View the components of each resource by expanding the resource.

75. Choose ► **Lambda**

76. Right-click the Lambda function with the name **duplicateBookmarkCheck**, and choose **Copy ARN** to copy the ARN of the Lambda function.

77. In the **State Functions** definition, replace **duplicateBookmarkCheckArn** with the

choose **Copy ARN** to copy the ARN of the Lambda function.

77. In the Step Functions definition, replace (**duplicateBookmarkCheckArn**) with the value copied from the previous step.
78. Repeat the previous steps to copy the ARNs for the following Lambda functions and replace the placeholders in the Step Functions definition:
  - Replace (**validateURLArn**) with the ARN of the **validateURL** function.
  - Replace (**userApprovalEmailArn**) with the ARN of the **userApprovalEmail** function.
  - Replace (**publishApprovalArn**) with the ARN of the **publishApproval** function.
79. In the definition, replace (**APIGatewayEndpointURL**) with the API Gateway invoke URL that you noted earlier.
80. In the definition, replace (**ContestTopicArn**) with the **ContestTopic** ARN that you noted earlier.
81. Choose **Next**
82. On the **Specify details** page, configure the following:
  - **State machine name:** Enter `PublishStateMachine`
  - **Permissions:** Choose **Choose an existing role**
  - **Existing roles:** Choose **xxxx-SamDeploymentRole-xxxx**
83. Choose **Create state machine**

The state machine setup is now complete. Now, create a new event rule in EventBridge to invoke the new state machine.
84. In the AWS Management Console, on the **Services** menu, choose **Amazon EventBridge**.
85. In the left navigation pane, choose **Rules**.
86. From the **Event bus** dropdown menu, choose **bookmarks-bus**.
87. Choose **Create rule**
88. On the **Create rule** page, in the **Name and description** section, configure the following:
  - **Name:** Enter `publish-rule`



- **Name:** Enter `publish-rule`
- **Description:** Enter `Publish rule to invoke PublishStateMachine`

89. In the **Define pattern** section, configure the following:

- **Define pattern:** Choose **Event pattern**
- **Event matching pattern:** Choose **Custom pattern**
- Copy and paste the following code in the **Event pattern** code block:

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

90. Choose **Save**

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

- **Select an event bus for this rule:** Choose **Custom or partner event bus**
- Choose **bookmarks-bus**

92. In the **Select targets** section, configure the following:

- **Target:** Choose **Step Functions state machine**
- **State machine:** Choose **PublishStateMachine**
- Choose **Use existing role**
- Choose **xxxx-EventBridgeStateMachineRole-xxxx**

93. Choose **Create**

## Task 5: Reviewing the deployment and running test cases using the application

In this task, you take a quick overview of all of the components that have been deployed so far. You then run a few test cases to test the Step Functions workflow and the EventBridge rule. Note the following:

- The backend and front end have been integrated for end-to-end testing.
- You created a Lambda function and a state machine manually.
- You reviewed the Lambda functions that the AWS SAM deploy process created for the Step Functions state machine workflow.
- You updated the state machine with the Lambda function ARNs, API Gateway endpoint, and SNS topic ARN.

94. Open and log in to the bookmark application, if you are not already logged in.

### Test case 1: Add a new bookmark without the shared flag

In this test case, add a new *unshared* bookmark. Check the Step Functions state machine to see if it was invoked or not. Then, share the new bookmark and approve

### Test case 2: Add a new bookmark with the shared flag

In this test case, add a new *shared* bookmark. Then, approve the new bookmark.

102. In the bookmark application, add a new bookmark with the **Share Bookmark** toggle set to **On**. Make sure you enter a valid bookmark URL.

This invokes the **PublishStateMachine** workflow.

103. In the AWS Management Console, on the **Services** menu, choose **Step Functions**.

104. Choose the name of the **PublishStateMachine** state machine.

105. Choose the name of the **Running** state.

Scroll down to the **Graph inspector** section, and review the state machine steps.

Because this is a new bookmark and no duplicates are found, the

Scroll down to the **Graph inspector** section, and review the state machine steps.

Because this is a new bookmark and no duplicates are found, the **duplicateBookmarkCheck** function sends a **NotDuplicate** message and invokes the **validateURL** function.

Because this is a valid URL, the **validateURL** function sends a **Valid** message and invokes the **userApprovalEmail** function.

106. Check your email for approve and reject links from the **userApprovalEmail** function.

107. Choose the approval URL, and check the Step Functions workflow.

Because you manually approved the URL, the **publishApproval** function is invoked, and the contest value in the DynamoDB table is updated to **Approved**.

### Test case 3: Add a bookmark with an invalid URL

In this test case, add a new *shared* bookmark that is *invalid*. Then, share the new bookmark.

108. In the bookmark application, add a new bookmark **without** "http", "www", or ".com" in the URL field. Make sure the **Share Bookmark** toggle is set to **On**.

This invokes the **PublishStateMachine** workflow.

109. In the AWS Management Console, on the **Services** menu, choose **Step Functions**.

110. Choose the name of the **PublishStateMachine** state machine.

111. Choose the name of the most recently started state.

Scroll down to the **Graph inspector** section, and review the state machine steps.

Because this is a new bookmark and no duplicates are found, the **duplicateBookmarkCheck** function sends a **NotDuplicate** message and invokes the **validateURL** function.

Because this is *not* a valid URL, the **validateURL** function sends an **Invalid** message and invokes the **notifyUser** function.

### Test case 4: Add a duplicate bookmark as a different user

## Test case 4: Add a duplicate bookmark as a different user

In this test case, add a *new* user account in the bookmark application. Then, add a *shared* bookmark that is a duplicate of a bookmark that another user added.

112. In the bookmark application, create a second user account.

113. Add a bookmark that is a duplicate of a bookmark that another user added. Make sure the **Share Bookmark** toggle is set to **On**.

This invokes the **PublishStateMachine** workflow.

114. In the AWS Management Console, on the **Services** menu, choose **Step Functions**.

115. Choose the name of the **PublishStateMachine** state machine.

116. Choose the name of the most recently started state.

Scroll down to the **Graph inspector** section, and review the state machine steps.

Because this is a duplicate bookmark that was already shared, the **duplicateBookmarkCheck** function sends a **Duplicate** message and invokes the **notifyUser** function to send an email saying the bookmark is a duplicate.

**Note** If you have lab time remaining, run a few test cases for the human rejection process.

## Conclusion

👍 Congratulations! You now have successfully:

- Configured EventBridge to target a Step Functions workflow
- Used a Step Functions Standard workflow to orchestrate tasks
- Used Lambda for tasks within a Step Functions state machine


## End Lab

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

117. Return to the AWS Management Console.

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

119. Choose 

120. Choose 

121. (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
  - 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



## Additional resources

- For more information about asynchronous messaging for microservices, see <https://aws.amazon.com/blogs/compute/understanding-asynchronous-messaging-for-microservices/>.
  - For more information about Step Functions, see <https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html>.
- 