

# Try-it-out Exercises - Day 2



© 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

During the course, you use this environment to interact with the AWS Management Console and some of the services that the course discusses. The environment will be available to you throughout the day but will be reset for the following day's class.

### 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)
- 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.

## 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.

**Note** If you are prompted for a token, use the one distributed to you (or credits you have

be provisioned before continuing.

❗ 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.

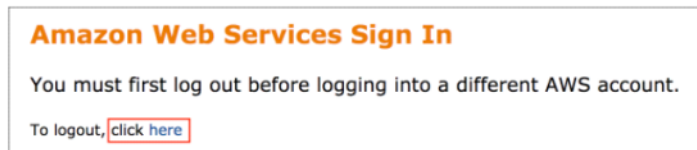
3. On the sign-in page, configure:

- **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

## Module 7

### Build two serverless applications from examples

To get started in this module, you deploy applications that help to illustrate topics discussed later in this module and the next module. These examples are also available for use in your own account if you want to review or work with them later.

For both examples, you download and unzip each application using your AWS Cloud9 environment. The random error generator application has been scoped for use in this course, but you can get a copy of the application on which it is based from <https://docs.aws.amazon.com/lambda/latest/dg/samples-errorprocessor.html>. The serverless API application is available as a sample application in the AWS Lambda applications console.

4. In the AWS Cloud9 environment, run the following commands to download the code, unzip it, and deploy both applications:

```
wget https://us-west-2-tcprod.s3-us-west-2.amazonaws.com/courses/ILT-TF-200-SVDVSS/v1.0.2/try-it-out-Day2/scripts/day-2-apps.zip
unzip day-2-apps.zip
chmod +x *.sh
./create-bucket.sh
./build-layer.sh
./deploy.sh
```

---

5. Run the **invoke** script:

```
./invoke.sh
```

**Note** The invoke script continues to run until you stop it. If you run the command by right-clicking the *invoke.sh* file in the file navigation window in AWS Cloud9, it opens in its own terminal window with an option to stop it. You come back and stop it after it has generated data for the course examples.

## Try-it-out exercise: Lambda console

Open your environment and navigate to the Lambda console. Use the details below to follow your instructor.

### Review the resource-based policy

6. Open the **putItemFunction** function. Choose the **Configuration** tab, then choose **Permissions** in the left navigation pane, and scroll to the section titled **Resource-based policy** to review who or what services have permission to invoke your function.

### Review the execution role

7. Choose the **Configuration** tab, then choose **Permissions** in the left navigation pane and find the role name in the **Execution role** section. Use the dropdown option in the **Resource summary** section to review the services and permissions that the function has through its execution role.

### Review requirements for connecting to a VPC

8. Within the **Configuration** tab of your function, choose **VPC** in the left navigation pane to find the characteristics of the VPC that you would need to configure.

### Add an OnFailure destination

9. Navigate to the **Function overview** section of your function.

10. Choose **Add destination**. Select the desired target (for example, an Amazon Simple Notification Service (SNS) topic).

### Modify the memory setting

11. Choose the **Configuration** tab, and select **General configuration** in the left navigation pane to modify the memory for your function.

### Modify timeout setting

12. In the same section, you can also modify the timeout.

### Review Lambda function code

13. Choose the **Code** tab, and navigate to the **Code source** section. This section lets you directly review and edit functions depending on how they were written and deployed.

### Review code to separate the business logic

14. Navigate to the **error-processor-randomerror** function. In the **Code source** section, select **index.js**, open the context (right-click) menu and choose **Open**.

The handler consists of a single line of code:

```
exports.handler = myFunction
```

All of the other logic happens in the **myFunction** function starting with the following line:

```
var myFunction = async function(event, context) {
```

#### Review where layers are added

15. The **error-processor-randomerror** function uses a layer. In the **Code** tab of the function, scroll to the **Layers** section. You can add a layer in two ways: Within the **Code** tab, use the **Add a layer** option, or go to the Lambda navigation menu on the left, and use the **Layers** option.

#### Enable Amazon CloudWatch Logs Insights

16. While you have the **error-processor-randomerror** function open, choose the **Configuration** tab, select the **Monitoring and operation tools** section, and choose **Edit**. Under **CloudWatch Lambda Insights**, turn on **Enhanced monitoring** and **Save**.

#### Review environment variable example

17. The **error-processor-processor** function has an example of an environment variable set for the bucket used in the function. Open the function, choose the **Configuration** tab, select the **Environment variables** section, and you'll see the bucket variable. You can see the variable being used in the function in the following line:

```
const bucket = process.env.bucket
```

#### Publish a new version of a function

18. Choose the **Actions** dropdown, choose **Publish new version**, and then choose **Publish**.
19. Navigate back to the **error-processor-processor** function overview page. Choose the **Versions** tab. You can see 1 in the **Versions** section as an option.

#### Create an alias

20. Choose the **Actions** dropdown, and then choose **Create alias**. Name the alias, select the version to associate it with, and choose **Save**. Choose the **Aliases** tab, and you'll be able to see the alias as an option.

#### Configure the test event

21. Choose the **error-processor-randomerror** function. Choose the **Test** tab to create a test event.
22. Name the test event, and paste in the following code snippet:

```
{
  "max-depth": 50,
  "current-depth": 0,
  "error-rate": 0.05
}
```

#### Save and test

23. Choose **Save changes** to save the test event. Choose **Invoke**. A panel above the main console sections displays your results.

## Module 8

## Try-it-out exercise: AWS Step Functions

24. Open your environment, and navigate to the Step Functions console. Use the details below to follow your instructor.

### Start creation of a state machine

25. In the Step Functions console navigation menu, select **State machines**. Choose **Create state machine**. Select **Author with code snippets**. Select **Standard** for the type.

### Preview code for a task state

26. In the **Definition** section, choose **Generate code snippet**. From the list of code snippets, choose **AWS Lambda: Invoke a function**. The preview shows you the code that you would use to incorporate this as a step in your state machine.

### Preview the code for a Choice state

27. In the console, change the option from **Author with code snippets** to **Start with a template**.
28. Select the **Choice state** template. Review the code and visualization populated in the **Definition** section.

### Preview the code for a Parallel state

29. Select the **Parallel** template. Review the code and visualization populated in the **Definition** section.

### Preview the code for a Wait state

30. Select the **Wait state** template. Review the code and visualization populated in the **Definition** section.

### Preview the code to wait for a batch job to complete

31. In the **Definition** section, from the list of code snippets, choose **AWS Batch: Manage a job**. Toggle the check box for **Wait for batch job to complete**, and review the resulting code changes in the preview panel.

### Preview the code to use a task token with callback

32. Select one of the code snippets that calls an AWS service. Then select the check box for **Wait for callback with task token**, and review the code in the preview panel.

### Preview the code for a Map state

33. Select the **Map state** template. Review the code and visualization populated in the **Definition** section.

### Select the Retry failure template

34. Select the **Retry failure** template. In the code sample it provides, you'll find the array of retriers within the Task state.

### Select the Catch failure template

35. Select the **Catch failure** template. In the code sample it provides, you'll find the catch field and its array of catchers and, beneath that, the states that are used for fallback from the catchers. You can also review the visualization of the catch failure flow.

## Troubleshoot a Step Functions express workflow

36. In the Step Functions console, select the **CheckName-Express** express state

machine.

37. Choose **Start execution**

38. In the **Input** field, enter the payload below:

```
{ "application": { "name": "evil Spock" } }
```

39. Choose **Start execution**

This produces an error. With express workflows, you can review CloudWatch logs to troubleshoot the error. (In the **Results** panel, use the CloudWatch Logs link that tells you that the state machine has failed.)

To take advantage of the visual results available in standard workflows, copy the express state machine definition to a new standard state machine.

40. On the **CheckName-Express** state machine, from the **Actions** menu, choose **Copy to new**.

41. Change the type to **Standard**, and choose **Next**

42. Name the new state machine **CheckName-Standard**

43. In the **Permissions** section, choose **Choose an existing role**. Do not modify the role that is presented. Choose **Create state machine**

44. Choose **Start execution** again, and use the same payload in the input field:

```
{ "application": { "name": "evil Spock" } }
```

45. When the standard state machine fails, use the **Graph inspector** to isolate where the failure occurred.

46. Choose the **Denied** task, and look at the details associated with the failure.

## Module 9

### Configure Amazon API Gateway logging and AWS X-Ray tracing and generate sample traffic

#### Enable CloudWatch Logs and X-Ray for the REST API created with your serverless API backend application

**i** If you ended this lab at any point during the day, repeat the first two steps in the Module 7 section to deploy and invoke the serverless apps again before proceeding with the following steps.

47. From the Lambda console navigation menu, choose **Applications**, and select **serverless-api**.

48. Scroll down on the **Overview** tab to find the resources created with this application. Choose **ServerlessRestApi**.

49. Choose **Stages**, and then choose the **Prod** stage to open the Prod stage editor. Choose the **Logs/Tracing** tab.


50. Select the **Enable CloudWatch Logs** check box. For the **log level**, choose **INFO**. (For production APIs, you would typically use the **ERROR** level rather than logging



everything, but for this example, use the **INFO** option to produce more detailed logs.)

51. Select the **Enable X-Ray Tracing** check box.

52. Choose **Save Changes**

 **Note** While saving these changes, if you see an error message that says **CloudWatch Logs role ARN must be set in account settings to enable logging**, then API Gateway is missing the CloudWatch Logs service role to write these logs into CloudWatch Logs. In order to fix this issue, do the following:

- In the left navigation pane of the lab page, copy the **APIGatewayInvocationRole** value.
- At the bottom of the left navigation pane of the API Gateway console, choose **Settings**.
- In the **CloudWatch log role ARN** box, paste the **APIGatewayInvocationRole** value you copied from the lab page.
- Choose **Save**
- Repeat the steps in this task to successfully enable X-Ray and CloudWatch Logs.

#### Generate log data for the serverless API backend application

53. In the Lambda console, go back to the **Overview** tab for the application, and copy the API endpoint that is displayed above the **Resources** section.

54. In your AWS Cloud9 environment, run the below command, replacing *{paste-your-endpoint-here}* with your API endpoint:

```
ENDPOINT={paste-your-endpoint-here}
```

55. Run the following curl command to send POST requests to the application endpoint. When you run the POST requests, API Gateway proxies them to the **putItemFunction** function, which writes them to the Amazon DynamoDB table called **SampleTable**.

```
curl -d '{"id":"1234ABCD", "name":"Apple Pie"}' -H "Content-Type: application/json" -X POST $ENDPOINT
curl -d '{"id":"2234ABCD", "name":"Glazed Donut"}' -H "Content-Type: application/json" -X POST $ENDPOINT
```

56. Run the following command to send a GET request to the endpoint to get a list of items:

```
curl $ENDPOINT
```

57. Run the following command to send a GET request with individual item IDs to get single items:

```
curl $ENDPOINT/1234ABCD
curl $ENDPOINT/2234ABCD
```

## Try-it-out exercise: CloudWatch logs

Open your environment and navigate to the CloudWatch console. Use the details below to follow your instructor.

#### Review API Gateway logs for your application

58. Find the log group for the serverless backend API associated with the serverless backend application that you created. It will have the words **API-Gateway-Execution-Logs** in its name and the unique API identifier that AWS generated for the API in API Gateway.

59. Select the latest log stream, and look at the information included in the logs.

#### Review Lambda logs for a function

60. Find the log group for the **getAllItemsFunction** associated with the serverless backend application that you created.

61. Select the latest log stream, and look at the most recent REPORT entry.

#### Run a query

62. With your Lambda log group selected, choose the option to **View in Logs Insights**.

63. Run the default query that is included in the query box.

#### Use a Lambda sample query

64. In the right navigation, choose the **Queries** option, and choose one of the Lambda sample queries. Apply the option, and run the query again.

### Try-it-out exercise: X-Ray console

Open your environment and navigate to the X-Ray console. Use the details below to follow your instructor.

#### Drill down to a trace

65. In X-Ray, choose **Service map** from the left navigation pane. Find and select the **error-processor-randomerror** node that says **AWS::Lambda::Function**.

66. In the **Service details** section, choose **View traces**.

67. Select one of the traces in the trace list, and review the kind of information that is available.

#### Segments and subsegments

68. Within the trace you've selected, scroll to the line that has **AWS::Lambda::Function** in the title, and choose the first item that is listed. This represents a segment. In the **Overview** tab, you'll find the segment ID.

69. Underneath the segment, choose the **Invocation** item. On the **Overview** tab, you'll find that the parent ID matches the segment ID of the top-level segment.

#### Review annotations

70. Using the same trace, choose the **annotations** item in the list under the Invocation. Choose the **Annotations** tab to view the annotations that have been created for this trace.

### Try-it-out exercise: CloudWatch metrics

Open your environment, and navigate to the Lambda console. Use the details below to follow your instructor.

#### Lambda

71. On the Lambda console, choose the **error-processor-randomerror** function, and choose the **Monitor** tab. In the corner of the **Error count and success rate (%)** metric, choose the ellipsis (...), and then choose **View in metrics**.

#### API Gateway

72. Rest APIs have a metrics dashboard. In the API Gateway console, choose the



serverless backend API in your account.

73. Choose the **Dashboard** option from the menu. Choose any of the graphs to view the graphed metrics in CloudWatch.

#### Add metric to dashboard

74. From the CloudWatch metrics console, choose the **Actions** dropdown for the graph, and then choose **Add to dashboard**.

#### Review Lambda Insights for a function

75. From the CloudWatch console menu, choose **Lambda Insights**. Choose **Multi-function** to review a summary of Lambda Insights for all functions.
76. Scroll to the **Function summary** section for quick stats on the number of invocations and cold starts.
77. Select the **error-processor-randomerror** function from the list to drill down into details about the function performance.

## Try-it-out exercise: CloudWatch ServiceLens

Open your environment and navigate to the CloudWatch console. Use the details below to follow your instructor.

#### Open ServiceLens

78. From within CloudWatch in the left navigation menu, locate **ServiceLens**, and choose **Service Map**.

#### Review data for one function

79. Choose one of the Lambda function representations for the **error-processor-randomerror** function that indicates errors.

#### Review details

80. The metrics are graphed below the service map, and you can choose **View logs**, **View traces**, or **View dashboard** for the item you chose.

## End Lab

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

81. Return to the AWS Management Console.
82. On the navigation bar, choose **awsstudent@<AccountNumber>**, and then choose **Sign Out**.
83. Choose **End Lab**
84. Choose **OK**
85. (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](#).

```
wget https://us-west-2-tcpred.s3-us-west-2.amazonaws.com/courses/ILT-TF-200-SVDVSS/v1.0.2/try-it-out-Day2/scripts/day-2-apps.zip unzip day-2-apps.zip chmod +x *.sh ./create-bucket.sh ./build-layer.sh ./deploy.sh
```

```
./invoke.sh
```

ENDPOINT={paste-your-endpoint-here}

```
curl -d '{"id": "1234ABCD", "name": "Apple Pie"}' -H "Content-Type: application/json" -X POST $ENDPOINT
curl -d '{"id": "2234ABCD", "name": "Glazed Donut"}' -H "Content-Type: application/json" -X POST $ENDPOINT
```

curl \$ENDPOINT

curl \$ENDPOINT/1234ABCD curl \$ENDPOINT/2234ABCD