Lab 4 - Developing Event-Driven Solutions with AWS Lambda

Friday, December 25, 2020 10:45 AM

Developing on AWS - Lab 4 - Developing Event-Driven Solutions with AWS Lambda

2 hours 30 minutes Free ★★★★½ Rate Lab

.Net version

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

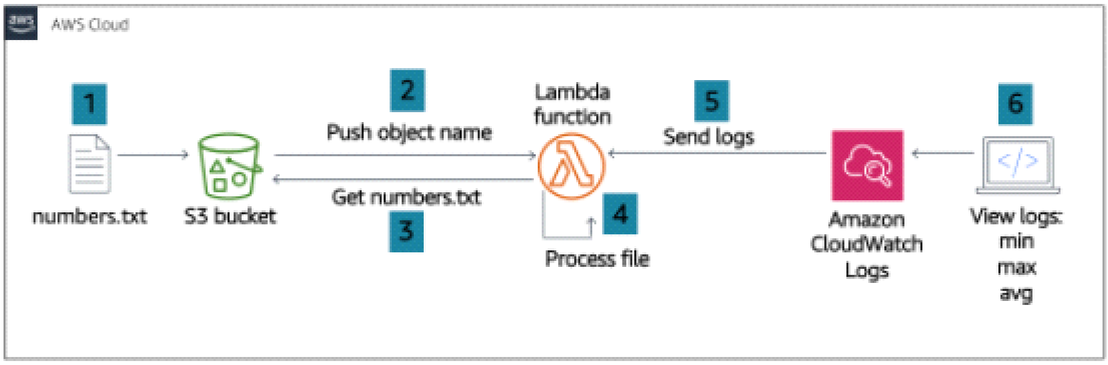
Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Overview

In this lab, you will learn how to use AWS Lambda to run a Lambda function when objects are uploaded into an Amazon S3 bucket.

The Lambda function will calculate the minimum, maximum, and average of the numbers contained in an object uploaded to the S3 bucket.

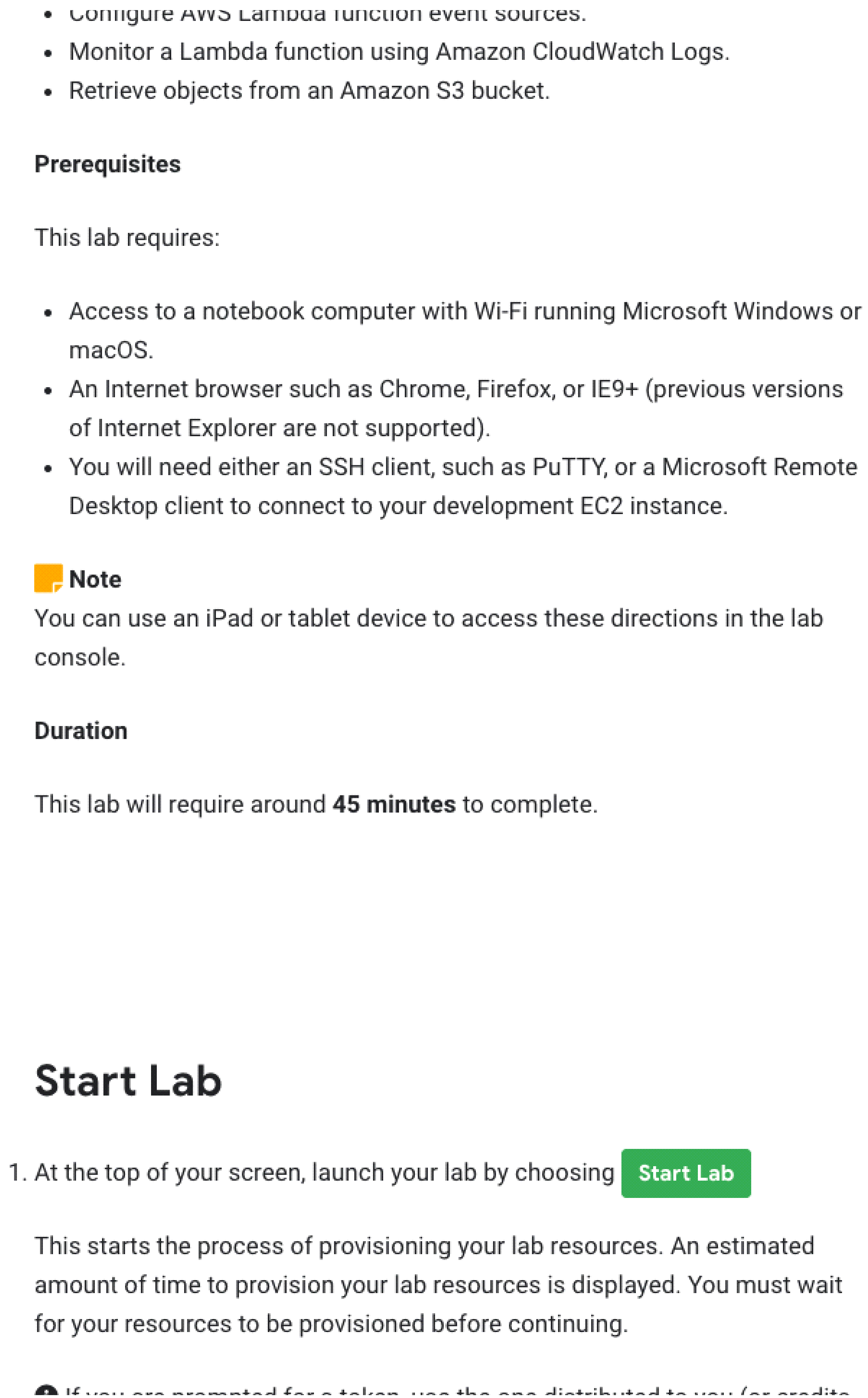
The terms *file* and *object* are used interchangeably when referring to the contents of Amazon S3 buckets.



Objectives

After completing this lab, you will be able to:

- Create an Amazon IAM Lambda Execution Role.
- Create an AWS Lambda function.
- Configure AWS Lambda function event sources.
- Monitor a Lambda function using Amazon CloudWatch Logs.



- Configure AWS Lambda function event sources.
- Monitor a Lambda function using Amazon CloudWatch Logs.
- Retrieve objects from an Amazon S3 bucket.

Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi running Microsoft Windows or macOS.
- An Internet browser such as Chrome, Firefox, or IE9+ (previous versions of Internet Explorer are not supported).
- You will need either an SSH client, such as PuTTY, or a Microsoft Remote Desktop client to connect to your development EC2 instance.

Note
You can use an iPad or tablet device to access these directions in the lab console.

Duration

This lab will require around **45 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.

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

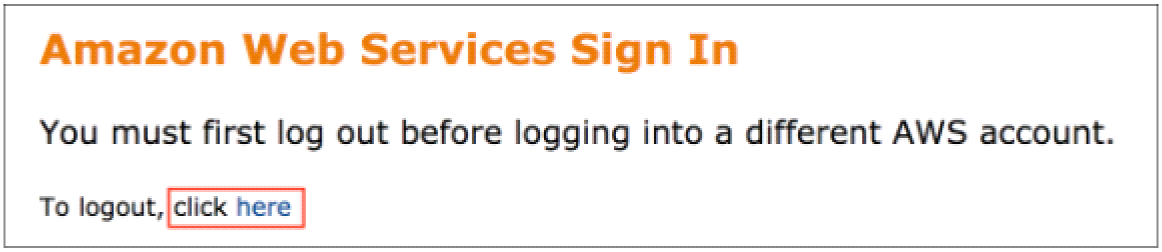
3. On the sign-in page, configure:

   - **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

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: Connecting to Your Development Environment

4. To connect to your Dev instance, see the Appendix: Connecting to Your Development Environment.

Task 2: Creating Your Input Bucket in the AWS Management Console

In this section, you will create an input Amazon S3 bucket for your Lambda function to use.

5. Choose **Services ⌄** and select **S3**.

6. Select **Create bucket**.

7. **Bucket name**: `calculator-input-<your-initials>`

   Note
   For example, if your name is John Smith, your bucket name would be **calculator-input-js**.

8. **Region**: make sure that you have selected the region your lab instance is running in. If you are unsure of the region, verify the value with your instructor.

running in. If you are unsure of the region, verify the value with your instructor.

9. Select **Create bucket**.

10. Download the [numbers.txt]() to your local computer. You will use this S3 object to test your Lambda function in a following step.

11. In the S3 Console, select the newly created **calculator-input-<your-initials>** bucket.

12. Next, add the *numbers.txt* file to your bucket with the following steps:

    - Select the **Upload** button.
    - Select **Add files**.
    - Navigate to the *numbers.txt* file that you just downloaded and select it.
    - Choose **Open**.
    - Leave the remaining cards with their default values.
    - Navigate to the bottom of the page and choose **Upload**.

    You should see that your file is uploaded to your S3 bucket.

Task 3: Create an Execution Role for Your Lambda Function

In this section, you will walk through steps create an execution role for the Lambda function.

Note

The role has already been created for you. This will be an exercise in learning the steps necessary to create a role and attach policies to it.

The role has already been created for you. This will be an exercise in learning the steps necessary to create a role and attach policies to it.

13. Choose **Services ⌄** and select **IAM**.

14. In the navigation pane, select **Roles**.

15. Select **Create role**.

16. Under **Select type of trusted entity**, select **AWS service**, then select **Lambda**.

17. Select **Next: Permissions**.

18. For this role, select the box next to *both* of the following permissions:

    - `AmazonS3ReadOnlyAccess`
    - `AWSLambdaBasicExecutionRole`

    Note
    You may need to type `S3` and then `Lambda` into the **Filter** text box on the page to find these permissions more quickly.

19. Once you have selected *both* permissions, select **Next: Tags**, then select **Next: Review**.

20. For **Role name**, enter: `lambda_execution_role`

21. Given the was already created, you will see the following error message.

    A role named "lambda_execution_role" already exists.

22. Select Cancel.

Task 4: Create the AWS Lambda Function

In this section, you will create a Lambda function using the AWS Management Console.

By using this configuration, the Lambda function will be invoked in response to Amazon S3 notifications.

23. Choose **Services ⌄** and select **Lambda**.

    Note
    Make sure that you have selected the right region. Verify the lab region with your instructor if you are unsure.

    For example, in the console `us-west-2` will be displayed as `US West (Oregon)`. For more region mappings, see: Available Regions

24. Select **Create function**.

    Note
    Blueprints are code templates for writing Lambda functions. Blueprints are provided for standard Lambda triggers such as creating Alexa skills and processing Amazon Kinesis Firehose streams. This lab provides you with a pre-written Lambda function, so you will **Author from scratch**.

25. Select ◉ **Author from scratch** and then, make the following changes:

    - **Function name**: `DotnetCalculator`
    - **Runtime**: `.NET Core 2.1 (C#/Powershell)`.
    - **Permissions:**

        - Expand ▸ **Change default execution role**

- Expand ▸ **Change default execution role**
- **Execution role:** ● Use an existing role
- **Existing role:** `lambda_execution_role`

26. Select **Create function**.

    The lambda function page will be displayed with your function configuration.

    Note
    If you see the following message in the **Function code** section, it can be ignored.

    "The code editor does not support the .NET Core 2.1 (C#/PowerShell) runtime."

27. Scroll down to the **Runtime settings** card and click on **Edit** and then enter the following details.

    - **Handler**: `Calculator::Calculator.Function::FunctionHandler`

    This matches the *ASSEMBLY::TYPE::METHOD* format for the C# handler you will build and package on your developer instance. For more details see Lambda Function Handler (C#)

28. Select **Save**.

29. Go up to the **Designer** section. Select **+ Add trigger**.

30. Under **Trigger configuration**, select **S3** and enter the following settings:

    - **Bucket**: `calculator-input-<your-initials>`.
    - **Event type**: `All object create events`.

    The Lambda function will run whenever an object is created in your Amazon S3 bucket.

S3 bucket.

- **Recursive invocation**: ☑ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

31. Select **Add**.

    At the top of the window, you should see a message stating ✓ the trigger was added successfully and that the function is now receiving events from the trigger.

Task 5: Building, Packaging, and Deploying Your Lambda Function

In this section, you will build, package, deploy, and test a Lambda function. This function retrieves an object from an Amazon S3 bucket, calculates the minimum, maximum, and average of the numbers from the object.

Task 5.1: Test Your Lambda Function

In this section you test and debug code on your DEV instance. When you are happy with the code, exactly the same code is packaged and deployed to Lambda.

32. Open your Development Environment as was described in a previous task. [Appendix: Connecting to Your Development Environment](#).

33. Open the `C:\temp\workdir\LambdaCSharpLab\Calculator.sln` solution

Appendix: Connecting to Your Development Environment.

33. Open the `C:\temp\workdir\lambdaCSharpLab\Calculator.sln` solution in Visual Studio.

    Take a moment to familiarize yourself with *Function.cs*. The code is designed to respond to an S3 event in Lambda. The code retrieves a file from S3, as you did in a previous lab. A regular expression is used to locate all the numbers in the file. From this array the code calculates the minimum, maximum and average of the numbers. There are `Console.WriteLine` statements in a few places in the code.

34. Locate `TODO1` and update the `S3EventNotification.S3Entity` class with your bucket name and save your changes.

    This is a simulated event you use to test the C# code in local development.

35. Run the C# solution. In Visual Studio select **Debug -> Start Without Debugging**. The output from running the solution will be visible in a console application. You can also set breakpoints and run the debugging as you would with any console application.

    Note
    The project that you are working with was created with the Amazon.Lambda.Templates nuget package. For more information on the templates and the deployment `dotnet lambda` commands see Creating a Deployment Package (C#) » .NET Core CLI

    Check the output. Do you see the logging output from the *Console.WriteLine* statements?

Task 5.2: Deploy Your Lambda Function

Now that the code has been developed and tested on the DEV instance, you will deploy your Lambda function.

will deploy your Lambda function.

36. Open a PowerShell window and run the following command:

```
cd C:\temp\workdir\lambdaCSharpLab
```

37. To package and deploy your Lambda function, run the following command:

```
dotnet lambda deploy-function DotnetCalculator
```

The *dotnet lambda* command task has created your zip file of your function's compiled assembly along with all of its assembly dependencies. For more details on this process see: [Creating a Deployment Package (C#)]()

Task 5.3: Test Your Lambda Function's Invocation

38. Choose **Services ⌄** and select **Lambda**.

39. Select the **DotnetCalculator** function.

40. Select Test at the top right corner.

41. On the **Configure test event** page, for **Event name**, enter: `CalcTest`.

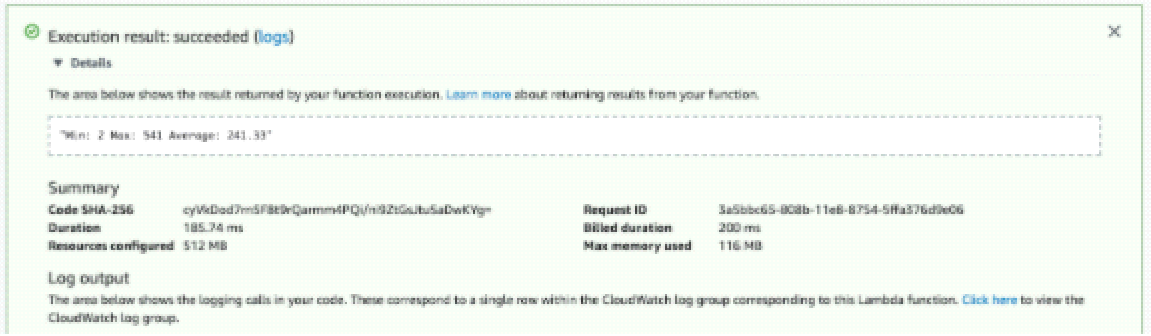
42. Enter the following JSON test payload and replace the bucket name with the name of the bucket you created earlier:

```
{
  "Records": [
    {
      "s3": {
        "object": {
          "key": "numbers.txt"
        },
        "bucket": {
          "key": "numbers.txt"
        },
        "bucket": {
          "name": "calculator-input-<your_initials>"
        }
      }
    }
  ]
}
```

43. Scroll to the bottom and select **Create**.

44. Scroll to the top and select **Test**. You should see that the execution result succeeded.

    After the AWS Lambda function has successfully run, under **Execution result: succeeded(logs)**, expand the **Details** section.



Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
"Min: 2 Max: 541 Average: 241.33"
```

Summary

Code SHA-256	cyVkDod7m5F8t9rQamm4PQi/n9ZtGsJtu5aDwKYg=	**Request ID**	3a5bbc65-808b-11e8-8754-5ffa376d9e06
Duration	185.74 ms	**Billed duration**	200 ms
Resources configured	512 MB	**Max memory used**	116 MB

Log output

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.

45. In the **Log Output** section, select the Click here link to see the CloudWatch Logs log group.

46. Select the log stream to see the logging output from the test run. Leave this tab open, you will return to the logs later in the exercise.

    Next, you will test the operation of your Lambda function trigger by uploading test case files to the S3 bucket.

47. Download the following test case files and inspect them on your local computer:

    - empty.txt
    - mixed_numbers_text.txt

- empty.txt
- mixed_numbers_text.txt
- text_only.txt
- Use a text editor to create your own test case.

48. Choose **Services ⌄** and select **S3**.

49. Select the input bucket that you created earlier, **calculator-input-<your_initials>**.

    - Select **Upload**.
    - Select **Add files**.
    - Navigate to the files that you just downloaded and select them.
    - Choose **Open**.
    - Leave the remaining cards with their default values.
    - Navigate to the bottom of the page and choose **Upload**.

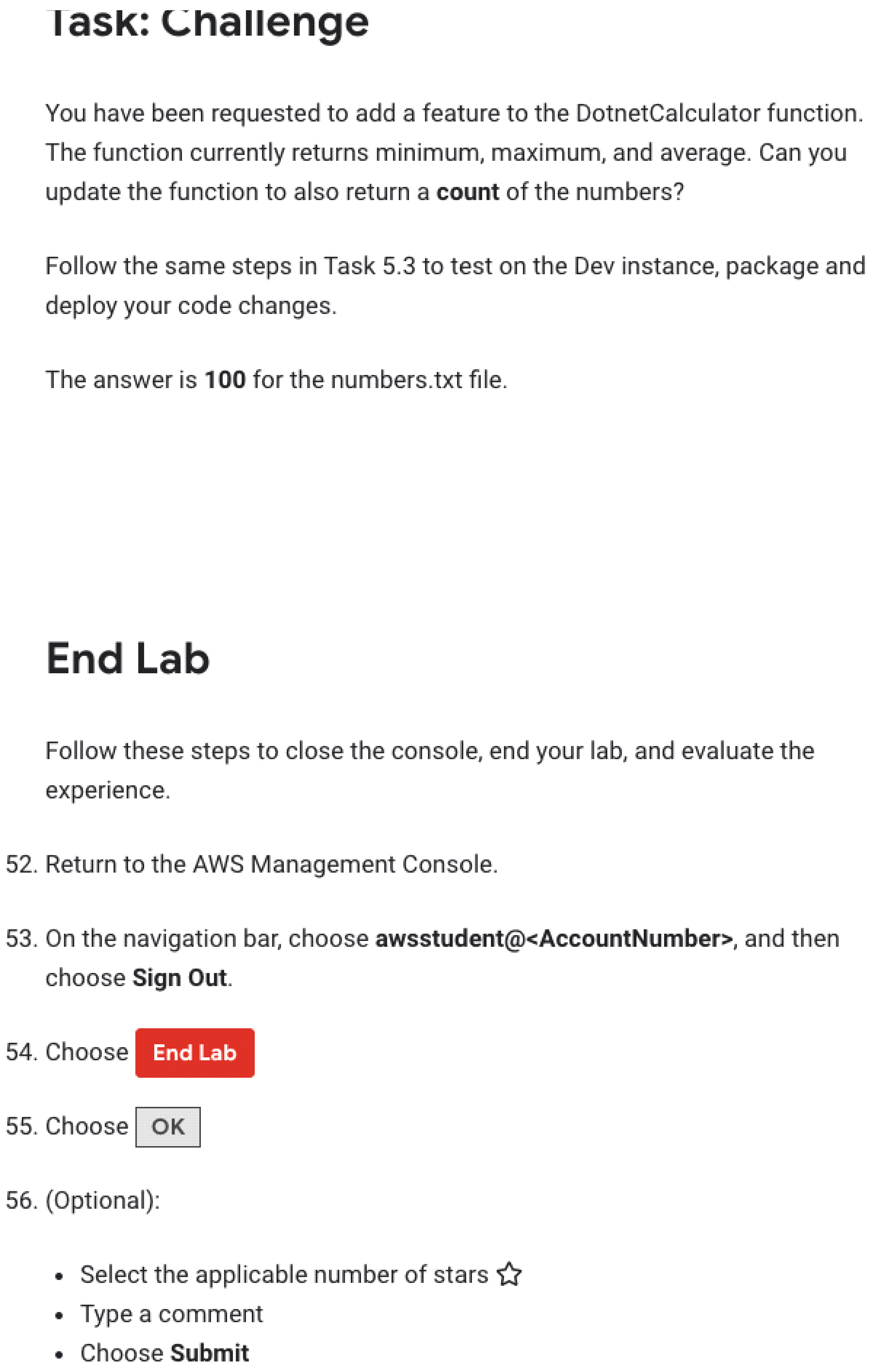
    You should see that your files are uploaded to your S3 bucket.

50. Return to the CloudWatch Logs log stream tab. Select the **/aws/lambda/DotnetCalculator** log group.

51. Inspect the contents of all the recent entries of all the log streams. Confirm you see the output from your test case files.

    Note
    You can also see the logs by selecting **Lambda -> Functions -> PythonCalculator -> Monitoring ->** **View logs in CloudWatch**.

Task: Challenge



Task: Challenge

You have been requested to add a feature to the DotnetCalculator function. The function currently returns minimum, maximum, and average. Can you update the function to also return a **count** of the numbers?

Follow the same steps in Task 5.3 to test on the Dev instance, package and deploy your code changes.

The answer is **100** for the numbers.txt file.

End Lab

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

52. Return to the AWS Management Console.

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

54. Choose End Lab

55. Choose OK

56. (Optional):

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

- 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. Congratulations! You are done!

Additional Resources

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*.

Appendix: Connecting to your Development Environment

You can connect to your Dev instance by using one of the following

You can connect to your Dev instance by using one of the following methods:

- Use Apache Guacamole to connect to your Windows Dev instance
- Use Remote Desktop to connect to your Windows Dev instance

To connect to the **Windows EC2 instance** by using Guacamole (Recommended), see the following directions:

- Connect to Your Windows Dev Instance by Using Apache Guacamole

To connect to the **Windows EC2 instance** by using RDP, see the following directions:

- Connect to Your Windows Dev Instance from a Windows Machine
- Connect to Your Windows Dev Instance from a macOS Machine

Connect to Your Windows Dev Instance by Using Apache Guacamole

57. In the **Connection Details** section in the lab console, go to the bottom for the Guacamole information. Copy the **GuacamoleLink** and paste it into a browser.

58. Go back to the lab console and copy the **WindowsPassword** to the clipboard.

59. Go to the Apache Guacamole sign in in the browser. Sign in by using the following steps:

- For **Username**, enter: `student`

- For **Username**, enter: `student`
- For **Password**, paste the **WindowsPassword** from the clipboard.
- Select **Log In**.

Your connection to your remote instance should start momentarily. Once you open a connection, you will see an image of the Dev instance desktop. You can interact with this image just as you would your normal desktop, or any remote desktop client.

You are now connected to your Windows Dev instance in the browser via Guacamole.

Tip Web browsers don't provide access to clipboard data, which means synchronization between your local clipboard and the remote clipboard is impossible. To copy and paste when using Guacamole, you must use the Clipboard editor. To open the Clipboard editor, press **Ctrl -> Alt -> Shift**.

Copy your text and paste it to the Clipboard editor. This will set the clipboard of your Dev instance to what you just pasted. You can also edit the text that you place in the Clipboard editor before pasting into your remote desktop. To close the Clipboard editor, press **Ctrl -> Alt -> Shift**.

To continue this lab, move on to Task 2: Creating Your Input Bucket in the AWS Management Console.

Connect to Your Windows Dev Instance from a Windows Machine

In this task, you will connect to a Windows EC2 instance from your Windows

In this task, you will connect to a Windows EC2 instance from your Windows machine.

Note

Perform the steps in this task only if you are connecting to **Windows Dev Instance** from a Windows machine.

60. In the lab console, go to the **Connection Details** section and copy the **WindowsInstanceIP** to the clipboard.

61. Open the Remote Desktop Connection application on your computer.

    - On Windows 7, select the **Start** icon, and in the **Search programs and files** textbox, enter: `Remote Desktop Connection`. Select the application when it appears in the **Programs** list.
    - On Windows 8, activate the Charms menu by moving the cursor into the lower right corner of the screen, and select the **Search** icon. Enter: `Remote Desktop Connection`. Select the application when it appears in the **Programs** list.
    - On Windows 10, select the **Start** icon, and then, select the **Search** icon. Enter: `Remote Desktop Connection`. Select the application when it appears in the **Programs** list.

62. In Remote Desktop Connection, for **Computer**, paste the IP of your Windows instance that you copied.

63. Select **Connect**.

64. Remote Desktop Connection will prompt you with a Login dialog asking for your username and password. By default, the application will use your current Windows username and domain. To change this, select **Use another account**.

Note

On Windows 10, select **More Choices** before selecting **Use a different account**.

On Windows 10, select **More Choices** before selecting **Use a different account**.

65. Go back to the lab console and copy the **WindowsPassword** to the clipboard.

66. For your login credentials, use the following values:

    - For **User name**, enter: `\Administrator`
    - For**Password**, paste the password from the clipboard.

    Note
    The `\` in the user name is important, as it tells Remote Desktop Connection that you are logging in as the local Administrator, and not as a domain user.

67. To connect to your instance, select **OK**. If you receive a prompt that the certificate used to verify the connection was not a known, trusted root certificate, select **Yes**.

    Result

    Your connection to your remote instance should start momentarily. When lab instructions in subsequent sections require a command window, open or use a Powershell window.

    To continue this lab, move on to Task 2: Creating Your Input Bucket in the AWS Management Console.

Connect to Your Windows Dev Instance from a macOS Machine

In this section, you will connect to a Windows EC2 instance from your

In this section, you will connect to a Windows EC2 instance from your macOS machine.

68. In the lab console, go to the **Connection Details** section and copy the **WindowsInstanceIP** to the clipboard.

69. Install Microsoft Remote Desktop if it is not already installed.

    - From the Dock, launch **App store**.
    - Search for the following string: `Microsoft Remote Desktop`
    - Select **Install**.

70. To open **Microsoft Remote Desktop**, on the Dock, select **Launchpad**. Then, select **Microsoft Remote Desktop**.

71. To create a new connection, select **New**.

72. Use the following values:

    - For **Connection name**, enter: `Windows Dev Instance`
    - For*PC Name*, paste in the IP address of your Windows Server instance that you copied to the clipboard.
    - For*User name*, enter: `\Administrator`

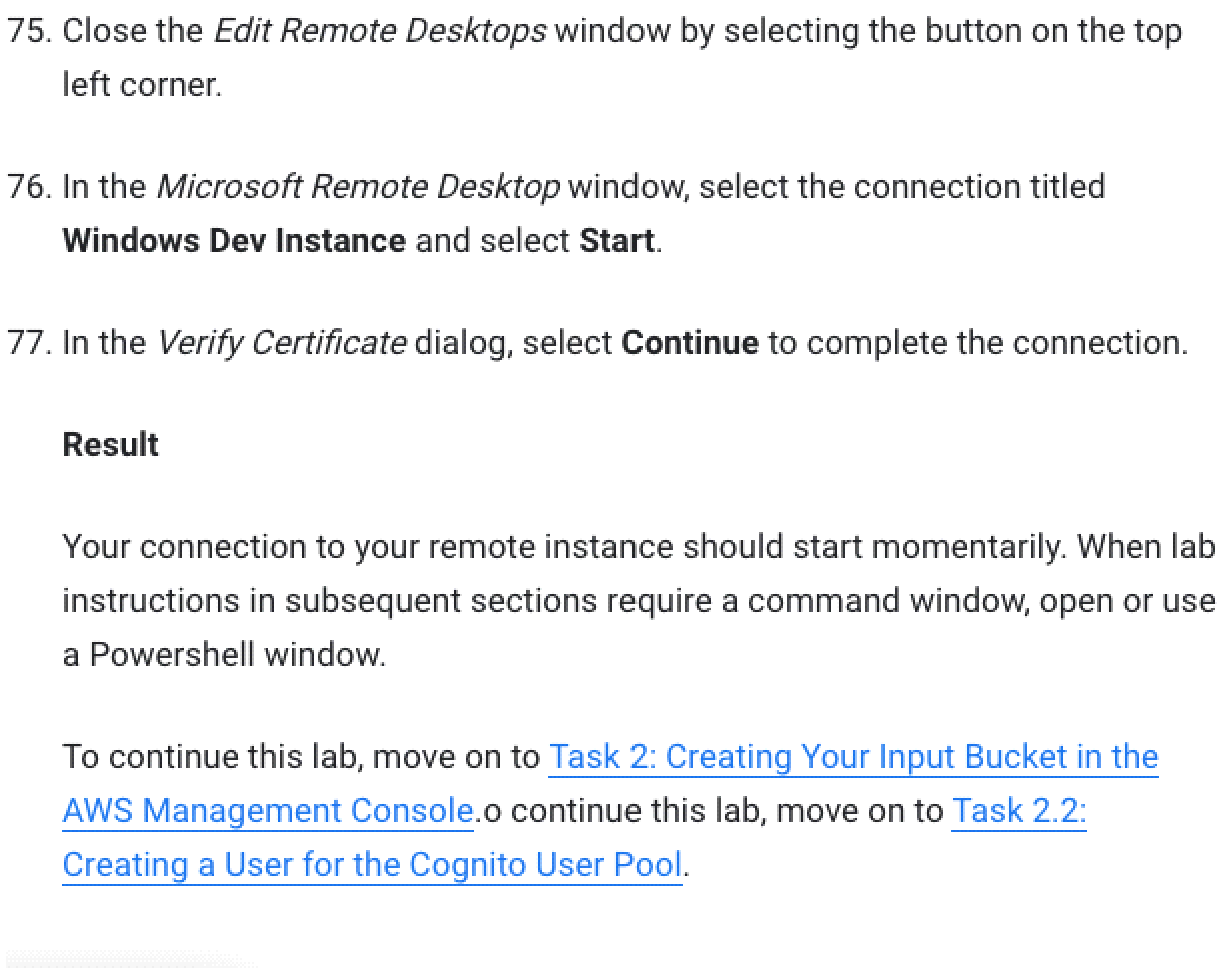
73. Go back to the lab console and copy the **WindowsPassword** to the clipboard.

74. Go back to your Microsoft Remote Desktop connection window and enter the following value:

    - For*Password*, paste in the password that you copied to the clipboard.

    Note
    The `\` in the user name is important, as it tells Remote Desktop Connection that you are logging in as the local Administrator, and not as a domain user.

75. Close the *Edit Remote Desktops* window by selecting the button on the top



75. Close the *Edit Remote Desktops* window by selecting the button on the top left corner.

76. In the *Microsoft Remote Desktop* window, select the connection titled **Windows Dev Instance** and select **Start**.

77. In the *Verify Certificate* dialog, select **Continue** to complete the connection.

    Result

    Your connection to your remote instance should start momentarily. When lab instructions in subsequent sections require a command window, open or use a Powershell window.

    To continue this lab, move on to Task 2: Creating Your Input Bucket in the AWS Management Console.o continue this lab, move on to Task 2.2: Creating a User for the Cognito User Pool.