Lab Information

Contents

Prerequisites
Overview
Start lab
Task 1: Create an AWS Lambda function
Task 2: Add processing logic to the Lambda function code
Task 3: Publish an AWS Lambda function
Task 4: Invoke an AWS Lambda function
Optional Challenge Task 5: Create the remaining AWS Lambda functions used in the course application
Summary
End lab
Additional Resources
Code Challenge Solutions
Appendix

AWS Services are loading.

aws training and certification

# Lab 4 (Python) - Develop Solutions Using AWS Lambda



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

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

**Duration**

This lab will require **60 minutes** to complete.

## Prerequisites

This lab requires:

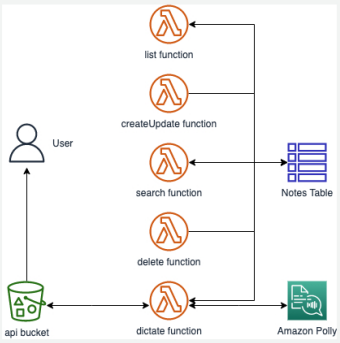
- Access to a Microsoft Windows or MacOS notebook computer with a WI-FI connection.
- An Internet browser such as Chrome, Firefox, or IE9+.
- **Note:** Previous versions of Internet Explorer are not supported.
- **Note:** You can use an iPad or tablet device to access these directions in the lab console.
- **Additional information:** Review additional lab environment specific details in the Appendix.

## Overview

Now that you have notes stored in an Amazon DynamoDB table, you can create the application logic to interact with them. In prior labs you have used the AWS SDK to interact with AWS services. Now you need to look at where you can run the code you have been developing.

With AWS you can use the Amazon Elastic Compute Cloud (Amazon EC2) to run your code when you need full control over the operating system and instance size. If you don't need that level of control, you can run your code in a prebuilt run-time environment that supports your language. If you need to control the orchestration of those containers, you could use Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS). But, if you just want to run your code with minimum configuration, you want to use AWS Lambda.

In this lab, you will create all of the backend logic for your application by deploying your code with AWS Lambda functions.




### OBJECTIVES

After completing this lab, you will be able to:

- Create AWS Lambda functions and interact programmatically using AWS SDKs and the AWS CLI.
- Configure Lambda functions to use the environment variables and integrate with other services.
- Generate Amazon S3 pre-signed URLs using AWS SDKs and verify the access to bucket objects.
- Deploy the Lambda functions with .zip file archives through your IDE and test as needed.
- Invoke AWS Lambda functions using the AWS Console and AWS CLI.

---

## Start lab

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

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

2. To open the lab, choose Open Console .

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

**WARNING: Do not change the Region unless instructed.**

### COMMON SIGN-IN ERRORS

**Error: You must first sign out**


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

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

**Error: Choosing Start Lab has no effect**

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

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

---

### Task 1: Create an AWS Lambda function

In the first function you create an audio version of your note on demand. This function will use Amazon DynamoDB, Amazon Polly, and Amazon S3.

You will start the lab work by using the AWS Console so that you can see a representation of the options that are available. However, most of the work in this lab will be performed in the AWS CLI and your code.

**Consider:** This lab is designed for both experienced and newer developers:

- For more experienced developers who enjoy a challenge, there are **High-Level Instructions** before each task that should provide you enough information to help you complete the task.
- Once you complete the updates test your code to ensure it works, troubleshoot if needed, and then move on to the next task.
- For newer developers, there are **Detailed Instructions** to guide you through each step of the lab.

**High-Level Instructions**

- Use the AWS Console to create the `dictate-function`.
- Set the runtime to **python3.11**.
- Use **lambdaPollyRole** to grant the Lambda function permissions

**Detailed Instructions**

3. From the **AWS Management Console**, use the **AWS search bar** to search for `Lambda` and then choose the service from the list of results.
4. Choose **Create function**.
5. For **Function name** enter `dictate-function`
6. For **Runtime**, select **Python 3.11**.
7. Expand **Change default execution role** and choose **Use an existing role**.
8. For **Existing role**, select **lambdaPollyRole**.
9. Choose **Create function**.

The AWS Lambda Console allows you to create, configure, monitor, and invoke Lambda functions. You can even view and modify the function code as long as it is less than 3 MB. The **lambdaPollyRole** provides permissions for the function to access Amazon DynamoDB, Amazon S3, Amazon Polly, and Amazon CloudWatch Logs services.

**Congratulations!** You have created the Lambda function named **dictate-function**.

### Task 2: Add processing logic to the Lambda function code

In your AWS Cloud9 Environment, you have been provided with a skeleton function that has all dependencies imported, clients and resources created, and user parameters created. This function will return a signed URL that grants access to an MP3 file hosted in an Amazon S3 bucket. The MP3 file is an audio version of a note stored in your database.

This function will flow through 3 main steps:

- Retrieve the text of a note from Amazon DynamoDB.
- Convert that text to speech with Amazon Polly and save that audio stream as a local file.
- Upload that file to Amazon S3 and generate a pre-signed URL to return from the function.

**High-Level Instructions**

**Note:** Make sure the Cloud9 environment is **ready**. You can verify this by checking the folder structure for a folder named **api** and a folder named **Lab-Is-Ready**. If you don't see them, just wait a few moments for the final configurations to complete.

**Note:** If you see the following message choose **Accept**.

- **.c9/project.settings have been changed on disk**
  - The project settings file (.c9/project.settings) was updated outside the IDE. If you did not make the changes we suggest reviewing the file before accepting. Do you want to accept the new settings?

---

- **Command:** Add two environment variables, **MP3_BUCKET_NAME** and **TABLE_NAME**, to the **dictate-function**. Be sure to replace *APIBUCKETNAME* with the **APIBucketName** value from the left side of the lab page.

```
apiBucket=APIBUCKETNAME
notesTable='Notes'
```

- In AWS Cloud9 complete **app.py** in **api/dictate-function** folder using the parameters passed to each function:
  - Add code to TODO1 that returns the note text from Amazon DynamoDB.
  - Add code to TODO2 that creates a local MP3 file in the `/tmp` folder with an audio version of that note using Amazon Polly and returns the file location.
  - Add code to TODO3 that uploads the MP3 file to Amazon S3, removes the local file, and returns a pre-signed URL to that object.

**Detailed Instructions**

#### TASK 2.1: ADD CODE TO RETRIEVE THE NOTE TEXT FROM AMAZON DYNAMODB

Open the **AWS Cloud9 IDE** to access your development environment.

10. From the **AWS Management Console**, use the **AWS search bar** to search for `Cloud9` and then choose the service from the list of results.
11. To open the AWS Cloud9 environment, locate **Lab4** and choose **Open** in the **Cloud9 IDE** column.

**Note:** Make sure the Cloud9 environment is **ready** . You can verify this by checking the folder structure for a folder named **api** and a folder named **Lab-Is-Ready** . If you don't see both of them, just wait a few moments for the final configurations to complete.

**Note:** If you see the following message choose **Accept** .

- **.c9/project.settings have been changed on disk**
  - The project settings file (.c9/project.settings) was updated outside the IDE. If you did not make the changes we suggest reviewing the file before accepting. Do you want to accept the new settings?

12. Expand the **api/dictate-function** folder and open **app.py** .

The **app.py** file has the structure you need to build the Lambda function for this lab.

The file starts with all required dependencies imported. The next section initializes the SDK clients and resources outside of the handler function, which follows AWS best practices for improving performance. Next, is the handler function that is called when your Lambda function is invoked. After the handler function, all of the application core logic is broken up into three separate functions.

When you inspect the handler function, notice that it first prints the event, which helps to troubleshoot if you have an issue. Next, all of the user parameters needed for the Lambda function are declared and retrieved from the event and environment.

Since the function requires two environment variables, unique to your lab, you need to add them to the Lambda function configuration.

13. **Command:** In the AWS Cloud9 terminal, run the command below to set a value to the **apiBucket** variable. Be sure to replace *APIBUCKETNAME* with the **APIBucketName** value from the left side of the lab page:

```
apiBucket=APIBUCKETNAME
```

**Expected output:**

None, unless there is an error.

14. **Command:** Run the command below to set a value to the **notesTable** variable.

```
notesTable='Notes'
```

**Expected output:**

None, unless there is an error.

15. **Command:** Run the command below to add the two required Lambda environment variables:

```
aws lambda update-function-configuration \
--function-name dictate-function \
--environment Variables="{MP3_BUCKET_NAME=$apiBucket, TABLE_NAME=$notesTable}"
```

**Expected output:**

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

{
    "FunctionName": "dictate-function",
    "FunctionArn": "arn:aws:lambda:ap-southeast-2:710606863198:function:dictate-function",
    "Runtime": "python3.11",
    "Role": "arn:aws:iam::710606863198:role/lambdaPollyRole",
    "Handler": "lambda_function.lambda_handler",
    "CodeSize": 299,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2022-04-11T20:07:06.000+0000",
    "CodeSha256": "fI06ZlRH/KN6Ra3twvdRllUYaxv182Tjx0qNWNlKIhI=",
    "Version": "$LATEST",
    "Environment": {
        "Variables": {
            "TABLE_NAME": "Notes",
            "MP3_BUCKET_NAME": "labstack-hornmarc-q2xzm8gvouu-pollynotesapibucket-zna9l72ij0vl"
        }
    },
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "5958a616-ff31-4185-b56b-c31df4dc79f6",
    "State": "Active",
    "LastUpdateStatus": "InProgress",
    "LastUpdateStatusReason": "The function is being created.",
    "LastUpdateStatusReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    }
}
```

16. Press the **space bar** to see more of the command output. Type **q** if the prompt does not return automatically.

After the function has all of the required values, it then processes the core logic. You will have to decide which choice is correct for each of the remaining functions. Use the Boto3 DynamoDB Table resource documentation for the next steps.

The first task is to retrieve the note text requested by the event. Scroll to the **getNote** function to complete your first coding challenge.

17. **Copy/Paste:** For **TODO 1** , choose and insert the correct choice to retrieve the note text from the DynamoDB table.

- Choice A

```
return records['Item']
```

- Choice B

```
return records['Item']['Note']
```

**Answer:** You can find the solution to this step in the TODO 1 solution section at the bottom of these instructions.

## TASK 2.2: ADD CODE TO USE AMAZON POLLY TO CONVERT THE NOTE TEXT TO AN MP3 FILE

With the note text, you are ready to use Amazon Polly to convert it from text to audio. Amazon Polly will return an audio stream if successful. You will need to write the stream to an MP3 file.

To complete the TODO 2 section, use the provided parameters that are passed to the function.

Use the Boto3 Polly Client Documentation for the next step.

18. **Copy/Paste:** For **TODO 2** , choose and insert the correct choice to convert the note text to a local MP3 file.

- Choice A

```
pollyResponse = pollyClient.synthesize_speech(
        OutputFormat='mp3',
        Text = text,
        VoiceId = VoiceId
    )
```

- Choice B

```
pollyResponse = pollyClient.synthesize_speech(
        Text = text,
        VoiceId = VoiceId
    )
```

**Answer:** You can find the solution to this step in the TODO 2 solution section at the bottom of these instructions.

### TASK 2.3: ADD CODE TO UPLOAD THE MP3 FILE TO AMAZON S3 AND GENERATE A PRE-SIGNED URL

Finally, complete TODO 3 to upload the file then generate and return a url variable. The URL needs to allow access to anyone who has the link and only be valid for a limited amount of time. This access needs to be granted without making the bucket or object public.

19. **Copy/Paste:** For **TODO 3** , choose and insert the correct choice to upload the MP3 file to Amazon S3 and generate a pre-signed URL.

- Choice A

```
s3Client.put_file(filePath,
    mp3Bucket,
    UserId+'/'+NoteId+'.mp3')
```

- Choice B

```
s3Client.upload_file(filePath,
    mp3Bucket,
    UserId+'/'+NoteId+'.mp3')
```

**Answer:** You can find the solution to this step in the TODO 3 solution section at the bottom of these instructions.

20. Save and close **app.py** .

## Task 3: Publish an AWS Lambda function

Your code for the AWS Lambda function consists of scripts or compiled programs and their dependencies. You use a deployment package to deploy your function code to Lambda. Lambda supports two types of deployment packages: container images and .zip file archives.

A .zip file archive includes your application code and its dependencies. When you author functions, using the Lambda console or a toolkit, Lambda automatically creates a .zip file archive of your code.

When you create functions with the Lambda API, command line tools, or the AWS SDKs, you must create a deployment package. You also must create a deployment package if your function uses a compiled language, or to add dependencies to your function. To deploy your function's code, you upload the deployment package from Amazon Simple Storage Service (Amazon S3) or your local machine.

**Note:** All dependencies used in this lab are already included in the AWS Lambda Python 3.11 runtime and do not need to be added to the deployment package manually.

**High-Level Instructions**

- Create a zip file that contains your app.py as a deployment package.
- Update the code of the function, created in task 1, with your deployment package.
- Update the function handler to match the path to the handler function in your deployment package.

**Detailed Instructions**

21. **Command:** In the AWS Cloud9 terminal, change to the **dictate-function** folder with the following command:

```
cd ~/environment/api/dictate-function
```

**Expected output:**

None, unless there is an error.

22. **Command:** Zip the app.py file with the following command:

```
zip dictate-function.zip app.py
```

**Expected output:**

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

    adding: app.py (deflated 61%)
```

23. **Command:** Upload the new code to your Lambda function with the command below:

```
aws lambda update-function-code \
--function-name dictate-function \
--zip-file fileb://dictate-function.zip
```

**Expected output:**

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

{
    "FunctionName": "dictate-function",
    "FunctionArn": "arn:aws:lambda:ap-southeast-2:710606863198:function:dictate-function",
    "Runtime": "python3.11",
    "Role": "arn:aws:iam::710606863198:role/lambdaPollyRole",
    "Handler": "lambda_function.lambda_handler",
    "CodeSize": 1265,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2022-04-11T20:14:00.000+0000",
    "CodeSha256": "8w0UfF47gTtmqhw9wOOGF0pZubXFH5o6EMPugOgDwCU=",
    "Version": "$LATEST",
    "Environment": {
        "Variables": {
            "TABLE_NAME": "Notes",
            "MP3_BUCKET_NAME": "labstack-hornmarc-q2xzm8gvouu-pollynotesapibucket-zna9l72ij0vl"
        }
    },
    "TracingConfig": {
        "Mode": "PassThrough"
{
    "FunctionName": "dictate-function",
    "FunctionArn": "arn:aws:lambda:ap-southeast-2:710606863198:function:dictate-function",
    "Runtime": "python3.11",
    "Role": "arn:aws:iam::710606863198:role/lambdaPollyRole",
    "Handler": "lambda_function.lambda_handler",
    "CodeSize": 1265,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2022-04-11T20:14:00.000+0000",
    "CodeSha256": "8w0UfF47gTtmqhw9wOOGF0pZubXFH5o6EMPugOgDwCU=",
    "Version": "$LATEST",
    "Environment": {
        "Variables": {
            "TABLE_NAME": "Notes",
            "MP3_BUCKET_NAME": "labstack-hornmarc-q2xzm8gvouu-pollynotesapibucket-zna9l72ij0vl"
        }
    },
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "4633553d-99d9-4b5a-833e-da3516fcf39a",
    "State": "Active",
    "LastUpdateStatus": "InProgress",
    "LastUpdateStatusReason": "The function is being created.",
    "LastUpdateStatusReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    }
}
```

24. Press the **space bar** to see more of the command output. Type **q** if the prompt does not return automatically.

This Lambda function has the handler function in an app.py file instead of the path that was defined in the console. You need to change the handler of the Lambda function so that the service knows what python function to run when it is invoked.

25. ⌨ **Command:** Run the command below to update the function handler:

```
aws lambda update-function-configuration \
--function-name dictate-function \
--handler app.lambda_handler
```

**Expected output:**

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

{
    "FunctionName": "dictate-function",
    "FunctionArn": "arn:aws:lambda:ap-southeast-2:710606863198:function:dictate-function",
    "Runtime": "python3.11",
    "Role": "arn:aws:iam::710606863198:role/lambdaPollyRole",
    "Handler": "app.lambda_handler",
    "CodeSize": 1265,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2022-04-11T20:15:55.000+0000",
    "CodeSha256": "8w0UfF47gTtmqhw9wOOGF0pZubXFH5o6EMPugOgDwCU=",
    "Version": "$LATEST",
    "Environment": {
        "Variables": {
            "TABLE_NAME": "Notes",
            "MP3_BUCKET_NAME": "labstack-hornmarc-q2xzm8gvouu-pollynotesapibucket-zna9l72ij0vl"
        }
    },
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "c3b649a9-686c-41f9-b520-78895190cee6",
    "State": "Active",
    "LastUpdateStatus": "InProgress",
    "LastUpdateStatusReason": "The function is being created.",
    "LastUpdateStatusReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    }
}
```

26. Press the **space bar** to see more of the command output. Type **q** if the prompt does not return automatically.

## Task 4: Invoke an AWS Lambda function

It is time to test your function and make sure it works. Functions can be tested using the AWS console, the AWS explorer that is included with AWS Cloud9, or the AWS CLI. In this task, you will use the AWS CLI and Console to invoke your function.

**High-Level Instructions**

- Invoke the Lambda function with the following event using the AWS CLI and AWS Management Console.

```
{
  "UserId": "newbie",
  "NoteId": "2",
  "VoiceId": "Joey"
}
```

**Detailed Instructions**

### TASK 4.1: INVOKE THE AWS LAMBDA FUNCTION IN YOUR AWS CLOUD9 TERMINAL

You can use the AWS CLI to invoke your Lambda functions as a test or to offload processing from your scripts. The AWS CLI can invoke a function synchronously (and wait for the response), or asynchronously. To invoke a function asynchronously, use the **–invocation-type Event** option. For type **Event** , the command will succeed if the event was successfully submitted.

There is one more option for the **Invocation Type** option and that is **DryRun** . Using **DryRun** is helpful to validate parameter values and verify that the user or role has permission to invoke the function.

**CAUTION:** If you need additional guidance to complete this task, you can view the complete **app.py** in the **Solution-dictate-function** folder.

27. **Context:** Open the context menu on the **dictate-function** folder and choose the option to create a new file. Name it `event.json`

28. Open **event.json** and paste the following json object to it.

```
{
  "UserId": "newbie",
  "NoteId": "2",
  "VoiceId": "Joey"
}
```

29. Save and close **event.json** .

30. **Command:** Run the command below to change into the **~/environment/api/dictate-function** directory if not already in that directory:

```
cd ~/environment/api/dictate-function
```

**Expected output:**

None, unless there is an error.

31. **Command:** Run the command below to invoke the Lambda function:

```
aws lambda invoke \
--function-name dictate-function \
--payload fileb://event.json response.txt
```

**Expected output:**

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

{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```

The command should return a 200 status saying that it was successful. There is now a new **response.txt** file in the **dictate-function** folder. That is the output that was returned from the Lambda function.

32. Open the **response.txt** file.

This is a pre-signed URL, excluding the quotes, that links to an MP3 file. If you paste the URL into a browser tab, you can listen to it on your computer. You will hear the note that you requested.

**Note:** If your **response.txt** file does not contain a URL in it, re-run Tasks 3 and 4 and correct your code.

### TASK 4.2: INVOKE THE AWS LAMBDA FUNCTION IN THE AWS CONSOLE

You can also manually invoke functions with the AWS console. This method is graphical and gives you immediate access to all logs generated by the function. The only drawback is that it is not easily automated.

33. From the **AWS Management Console** , use the **AWS search bar** to search for `Lambda` and then choose the service from the list of results.

34. Choose the function name of **dictate-function** .

35. Choose the **Test** tab and make the following selections under **Test event** :

- **Test event action:** Create new event
- **Event name:** `testPolly`
- **Event sharing settings:** Private
- **Template - optional:** hello-world
- **Event JSON:** Replace the default JSON object with the following:

```
{
  "UserId": "newbie",
  "NoteId": "2",
  "VoiceId": "Joey"
}
```

36. Choose `Save` to save the test event to your browser cache.

37. Choose **Test** .

38. In the **Execution result: succeeded (logs)** box, expand **Details**

39. Copy the URL returned (without the surrounding quotes) and browse to that URL.

Once again you see a URL to the MP3 file that you can test in your browser.

### Optional Challenge Task 5: Create the remaining AWS Lambda functions used in the course application

Subsequent labs will access DynamoDB with Lambda functions. In this task, you will create those functions. You have already explored how to access DynamoDB with the SDK in the previous lab, so the function code is provided for you. Review the app.py files to make sure you understand what is happening as you use them to create and test the functions.

**High-Level Instructions**

- Using the `aws lambda create-function` command and the code in the api folder create the remaining functions:
  - createUpdate-function
  - search-function
  - delete-function
  - list-function
- Use the `lambdaPollyRole` for all functions
- Test all the functions with the provided **event.json** files in each folder.

**Detailed Instructions**

#### TASK 5.1: CREATE THE FUNCTIONS

Lambda requires permissions to interact with other AWS services. This is granted with an AWS IAM role. You already know the name of the role created for use in this lab from task 1. Use the AWS CLI to create a variable with that role's ARN.

**Note:** You will use these commands to create several functions. You should copy them to a new file in your AWS Cloud9 environment so that it is easy to modify them for each additional function.

40. **Command:** In the AWS Cloud9 environment, run the command below to create a variable for the ARN of the **lambdaPollyRole** that the functions will use:

```
roleArn=$(aws iam list-roles --output text --query 'Roles[?contains(RoleName, `lambdaPollyRole`) == `true`].Arn')
```

**Expected output:**

None, unless there is an error.

41. **Command:** Set the **folderName** variable for the **createUpdate-function** folder with the following command:

```
folderName=createUpdate-function
```

**Expected output:**

None, unless there is an error.

42. **Command:** Change directories to the function code folder with the command below:

```
cd ~/environment/api/$folderName
```

**Expected output:**

None, unless there is an error.

43. **Command:** Create a zip file containing the function code with the command below:

```
zip $folderName.zip app.py
```

**Expected output:**

```
adding: app.py (deflated 54%)
```

In the AWS Cloud9 terminal, use the `aws lambda create-function help` command to complete the next step.

**Note:** You may need to confirm that the **$notesTable** variable from task 2 is still valid with `echo $notesTable`. If it is not recreate it like you did in Task 1.

44. **Copy/Paste:** Choose the correct command and use it to create the new function:

- Choice A

```
aws lambda create-function \
--function-name $folderName  \
--role $roleArn \
--environment Variables={TABLE_NAME=$notesTable} \
--zip-file fileb://$folderName.zip
```

- Choice B

```
aws lambda create-function \
--function-name $folderName  \
--handler app.lambda_handler \
--runtime python3.11 \
--role $roleArn \
--environment Variables={TABLE_NAME=$notesTable} \
--zip-file fileb://$folderName.zip
```

**Expected output:**

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

{
    "FunctionName": "createUpdate-function",
    "FunctionArn": "arn:aws:lambda:ap-southeast-2:710606863198:function:createUpdate-function",
    "Runtime": "python3.11",
    "Role": "arn:aws:iam::710606863198:role/lambdaPollyRole",
    "Handler": "app.lambda_handler",
    "CodeSize": 731,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2022-04-11T21:18:08.891+0000",
    "CodeSha256": "KIT3a8a76eElBEwHyACoQsayQa8GYlr1LRP6d/HN7+g=",
    "Version": "$LATEST",
    "Environment": {
        "Variables": {
            "TABLE_NAME": "Notes"
```

```
            TABLE_NAME : Notes
        }
    },
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "cc12a134-7a5f-492d-87d5-da6b1037317b",
    "State": "Pending",
    "StateReason": "The function is being created.",
    "StateReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    }
}
```


45. Press the **space bar** to see more of the command output. Type **q** if the prompt does not return automatically.

**Answer:** You can find the solution to this step in the TODO 4 solution section at the bottom of these instructions.

46. Use the steps in this task to create the remaining functions:

- search-function
- delete-function
- list-function

**Answer:** You can find an example script to this step in the Optional task 5.1 solution section at the bottom of these instructions.

### TASK 5.2: TEST THE NEW FUNCTIONS

Using what you learned in task 4 and the included event.json files, complete the following tests on your new functions. Test them in the following order:

- List all notes.
- Create a new note.
- Search for that new note.
- Delete the note.

**Answer:** You can find an example script in the Optional task 5.2 solution section at the bottom of these instructions.

## Summary

**Congratulations on completing the python version of the lab!** You can now:

- Create AWS Lambda functions and interact programmatically using AWS SDKs and the AWS CLI.
- Configure Lambda functions to use the environment variables and integrate with other services.
- Generate Amazon S3 pre-signed URLs using AWS SDKs and verify the access to bucket objects.
- Deploy the Lambda functions with .zip file archives through your IDE and test as needed.
- Invoke AWS Lambda functions using the AWS Console and AWS CLI.

## End lab

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

47. Return to the **AWS Management Console** .

48. At the upper-right corner of the page, choose **AWSLabsUser** , and then choose **Sign out** .

49. Choose End lab and then confirm that you want to end your lab. For more information about AWS Training and Certification, see *https://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

- AWS Lambda Documentation.
- Deploy Python Lambda functions with .zip file archives
- Best practices for working with AWS Lambda functions

---

## Code Challenge Solutions

### TODO 1 SOLUTION

- Choice A is incorrect because you only want to return the note attribute from the selected item, not the entire item.
- **Choice B is the correct code snippet.**

```
return records['Item']['Note']
```

Return to instructions

### TODO 2 SOLUTION

- **Choice A is the correct code snippet.**

```
pollyResponse = pollyClient.synthesize_speech(
        OutputFormat='mp3',
        Text = text,
        VoiceId = VoiceId
    )
```

- Choice B is incorrect because you must specify the OutputFormat when using polly to synthesize_speech.

Return to instructions

### TODO 3 SOLUTION

- Choice A is incorrect because put_file is not a valid method.
- **Choice B is the correct code snippet.**

```
s3Client.upload_file(filePath,
    mp3Bucket,
    UserId+'/'+NoteId+'.mp3')
```

Return to instructions

### TODO 4 SOLUTION

- Choice A is incorrect because the runtime and handler are required.
- **Choice B is the correct code snippet.**

```
aws lambda create-function \
--function-name $folderName  \
--handler app.lambda_handler \
--runtime python3.11 \
--role $roleArn \
--environment Variables={TABLE_NAME=$notesTable} \
--zip-file fileb://$folderName.zip
```

Return to instructions

### OPTIONAL TASK 5.1 SOLUTION

For the optional tasks, you can use the script below to create the additional functions. In the order below, replace **[ReplaceName]** with the function name and run the provided script.

- search-function
- delete-function
- list-function

**Note:** The code below is a script, therefore there are multiple commands.

```
functionName=[ReplaceName]

folderName=$functionName
cd ~/environment/api/$folderName

## Bundle and create the function
zip $folderName.zip app.py
aws lambda create-function \
--function-name $functionName  \
--handler app.lambda_handler \
--runtime python3.11 \
--role $roleArn \
--environment Variables={TABLE_NAME=$notesTable} \
--zip-file fileb://$folderName.zip
```

Return to instructions

### OPTIONAL TASK 5.2 SOLUTION

You can use the script below to test the functions. In the order below, replace **[ReplaceName]** with the function name and run the provided script. The results will be placed in the response.txt file in each folder.

- list-function
- createUpdate-function
- search-function
- delete-function

**Note:** The code below is a script, therefore there are multiple commands.

```
functionName=[ReplaceName]
folderName=$functionName
cd ~/environment/api/$folderName

## Test the function with the provided test event
aws lambda invoke \
--function-name $functionName \
--payload fileb://event.json response.txt
```

Return to the instructions

## Appendix

### AWS SERVICES NOT USED IN THIS LAB

AWS services that are not used in this lab are deactivated in the lab environment. In addition, the capabilities of the services used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

### ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. While not all of the icons will be used, the following list explains the purpose for each icon:

- **Command:** A command that you must run.
- **Expected output:** A sample output that you can use to verify the output of a command or edited file.
- **Note:** A note, tip, or important guidance.
- **Additional information:** Where to find more information.
- **Caution:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- **WARNING:** An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).
- **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
- **Copy/Paste:** A code block that displays the contents of a script or file you need to copy and paste that has been pre-created for you. When you need to copy only a certain part of a code block, there will be numbered **TODO** comments in the code.
- **Knowledge check:** An opportunity to check your knowledge and test what you have learned.
- **Security:** An opportunity to incorporate security best practices.
- **Refresh:** A time when you might need to refresh a web browser page or list to show new information.
- **Copy command:** A time when copying a command, script, or other text to a text editor (to edit specific variables within it) might be easier than editing directly in the command line or terminal.
- **Hint:** A hint to a question or challenge.
- **Answer:** An answer to a question or challenge.
- **Group effort:** A time when you must work together with another student to complete a task.

Return to the instructions