



Lab 4 (Python) - Develop Solutions Using AWS Lambda

1. Download the zip file for lab4, unzip it, and open it in VS Code.
2. Now go to AWS Console and open Lambda then click on create function.
3. Then give your function the same name and choose the runtime as python 3.11.

Basic information

Function name
Enter a name that describes the purpose of your function.
 C

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
 ▼ C

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

4. After that choose an existing role which is lambda polly role. In case if you didn't have this role created. Then you can copy the JSON code below and create this role.

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#) C.

Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
 ▼ C

View the lambdapollyrole role C on the IAM console.

{
"Version": "2012-10-17",
"Statement": [

```

{
  "Action": [
    "dynamodb>DeleteItem",
    "dynamodb>PutItem",
    "dynamodb>GetItem",
    "dynamodb>Query",
    "dynamodb>Scan",
    "dynamodb>UpdateItem",
    "dynamodb>DescribeTable",
    "polly>SynthesizeSpeech",
    "s3>PutObject",
    "s3>GetObject",
    "logs>CreateLogGroup",
    "logs>CreateLogStream",
    "logs>PutLogEvents",
    "lambda>TagResource"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
]
}

```

5. After creating the function in your Amazon console search for Cloud9 and create an environment.
6. For EC2 instance choose t3.medium as instance type and the platform should be Amazon Linux 2.

New EC2 instance

Instance type [Info](#)
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.

<input type="radio"/> t2.micro (1 GiB RAM + 1 vCPU) Free-tier eligible. Ideal for educational users and exploration.	<input type="radio"/> t3.small (2 GiB RAM + 2 vCPU) Recommended for small web projects.	<input type="radio"/> m5.large (8 GiB RAM + 2 vCPU) Recommended for production and most general-purpose development.
-------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------

Additional instance types
Explore additional instances to fit your need.

Additional instance types

Platform [Info](#)
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.

Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.

7. In the network settings choose SSH and create your environment.

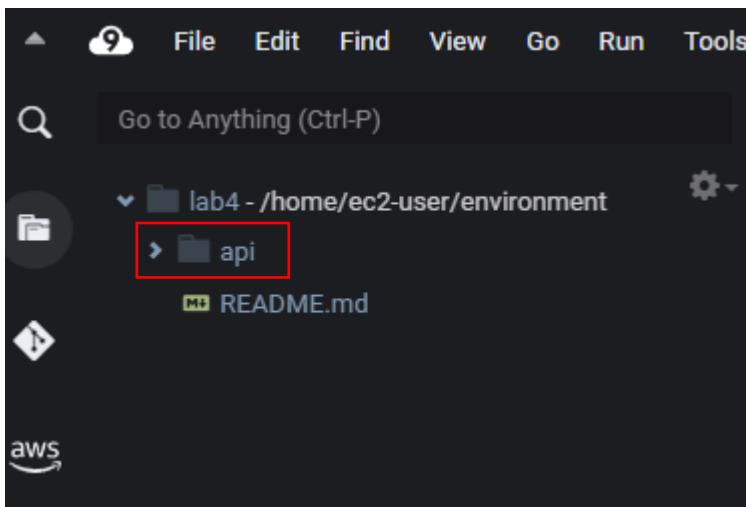
Network settings [Info](#)

Connection
How your environment is accessed.

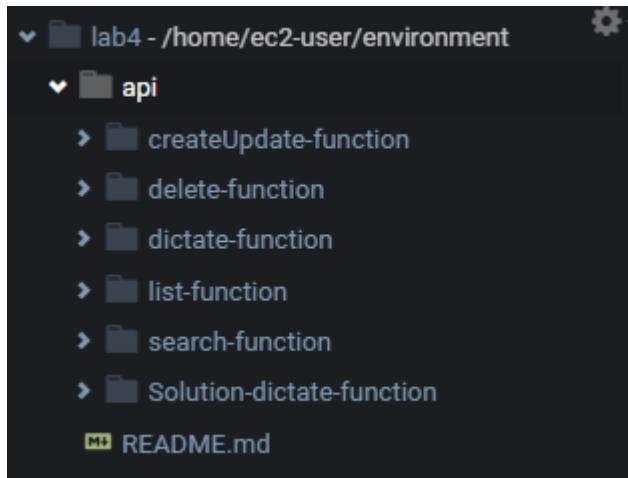
<input type="radio"/> AWS Systems Manager (SSM) Accesses environment via SSM without opening inbound ports (no ingress).	<input checked="" type="radio"/> Secure Shell (SSH) Accesses environment directly via SSH, opens inbound ports.
-----------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------

VPC settings [Info](#)

8. Once your environment is ready open the IDE and there you need to create a folder with the name api.
9. Then in this api folder you need to drag and drop all the files from lab4 folder.



10. It would look like this with all files pasted inside the api folder.



11. Now you are going to run some commands in the cloud9 IDE terminal below you can see the commands run one by one. Also, you will find a text file for commands.

```
$apiBucket='YOURBUCKETNAME'

$notesTable='Notes'

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

12. Now run the below commands to create a zip file of dictate function code so that we can publish this to our lambda function.

```
cd ~/environment/api/dictate-function
zip dictate-function.zip app.py
```

13. Then run the command below to upload the code to your lambda function. If you go to the lambda function you will see that the code has been uploaded.

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

14. Run this command to update the handler of your lambda function.

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

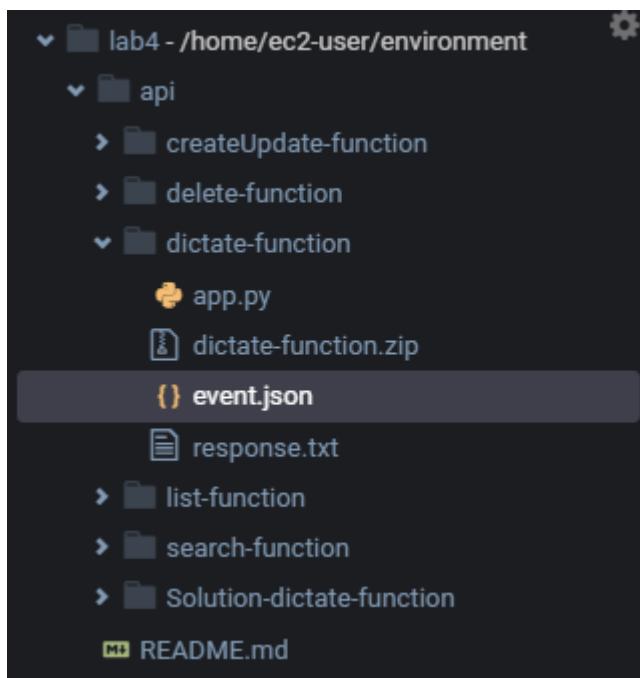
Runtime settings [Info](#)

Runtime Python 3.11	Handler Info app.lambda_handler	Architecture Info x86_64
------------------------	----------------------------------------------------	---------------------------------------------

[Edit](#) [Edit runtime management configuration](#)

15. Now, in cloud9, right-click on the dictate function and choose to create a new file with the name event.json, paste the script below, and then save it.

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



16. Then you need to run this command to invoke your lambda function.

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

```
ec2-user:~/environment/api/dictate-function (main) $ cd ~/environment/api/dictate-function
ec2-user:~/environment/api/dictate-function (main) $ aws lambda invoke \
> --function-name dictate-function \
> --payload fileb://event.json response.txt

{
  "StatusCode": 200,
  "ExecutedVersion": "$LATEST"
}
ec2-user:~/environment/api/dictate-function (main) $
```

17. Now come to your lambda function, go to test and create a new test and paste the below statements in it. Then click on the test.
18. If you get the execution result like you are seeing below then it means that your have completed the lab. Just copy this link and paste it into a new tab. A file will be downloaded.

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

The screenshot shows the AWS Lambda Test Execution Log interface. At the top, there are tabs for 'Code', 'Test' (which is selected), 'Monitor', 'Configuration', 'Aliases', and 'Versions'. Below the tabs, a green checkmark icon indicates 'Executing function: succeeded (logs)'. The log output area starts with the message 'The area below shows the last 4 KB of the execution log.' followed by a long URL string. Below this, there are two columns: 'Summary' on the left and 'Execution time' on the right. The 'Summary' column contains details like Code SHA-256, Request ID, Duration, and Resources configured. The 'Execution time' column shows '23 minutes ago (August 28, 2024 at 04:03 PM GMT+5:30)' and 'Function version \$LATEST'. At the bottom, there is a 'Log output' section.

19. Now there are some optional tasks to create other remaining functions.

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

20. So, I'll be creating one function and similarly you can create other functions.
21. There are some commands which you need to run in your cloud9 IDE if you want to run create those function.
22. Below you can see the commands to create a lambda function. Now you need to execute them one by one. Just remember to change the role name with your given name for your role.
23. Also, when you will be creating other function you just need to change the folder name command rest of all the commands are the same.

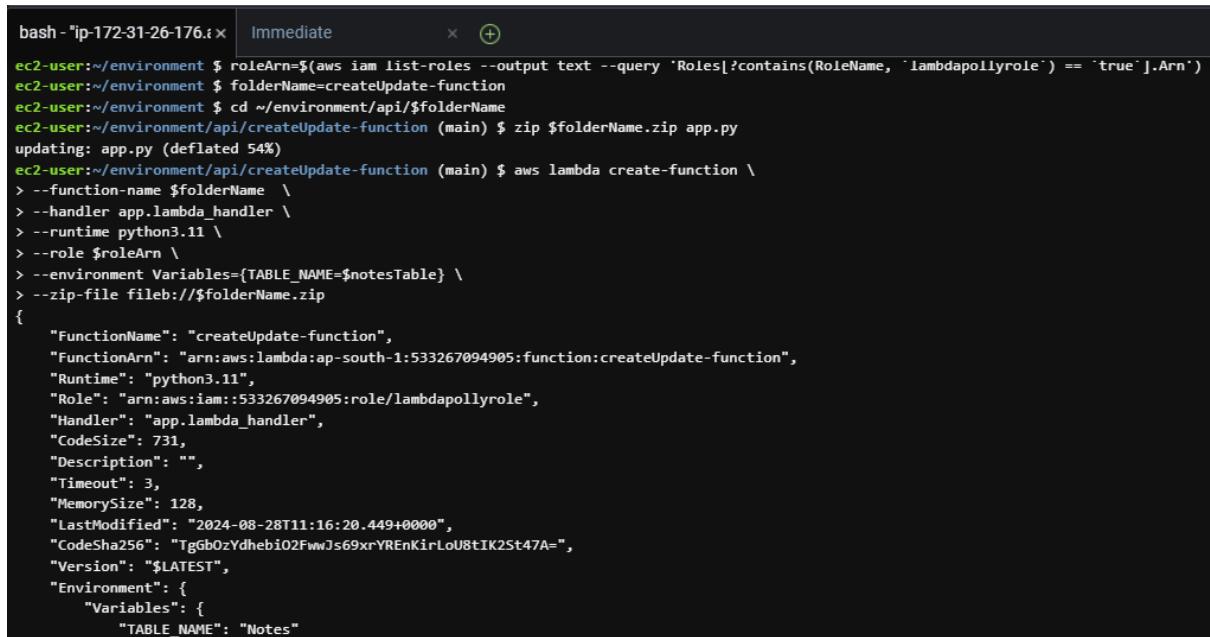
```
roleArn=$(aws iam list-roles --output text --query 'Roles[?contains(RoleName, `lambdaPollyrole`)]==`true`].Arn')
```

```
folderName=createUpdate-function //////
folderName=search-function //////
folderName=delete-function //////
folderName=list-function //////
folderName=delete-function
```

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

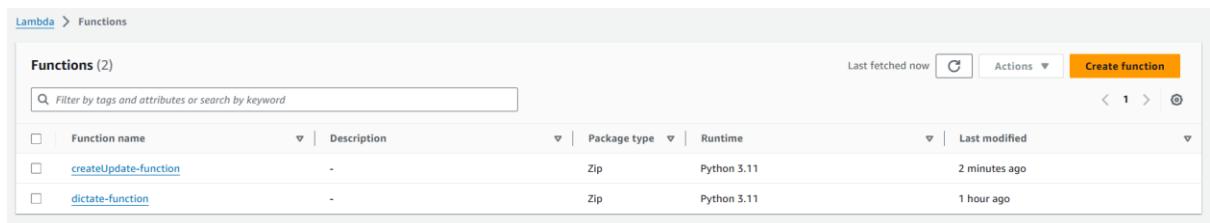
```
zip $folderName.zip app.py
```

```
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
```



```
bash - "ip-172-31-26-176.t" ✘ Immediate × ( + 
ec2-user:~/environment $ roleArn=$(aws iam list-roles --output text --query 'Roles[?contains(RoleName, `lambdapollyrole`)] == `true`].Arn')
ec2-user:~/environment $ folderName=createUpdate-function
ec2-user:~/environment $ cd ~/environment/api/$folderName
ec2-user:~/environment/api/createUpdate-function (main) $ zip $folderName.zip app.py
updating: app.py (deflated 54%)
ec2-user:~/environment/api/createUpdate-function (main) $ aws lambda create-function \
> --function-name $folderName \
> --handler app.lambda_handler \
> --runtime python3.11 \
> --role $roleArn \
> --environment Variables={TABLE_NAME=$notesTable} \
> --zip-file fileb://$folderName.zip
{
  "FunctionName": "createUpdate-function",
  "FunctionArn": "arn:aws:lambda:ap-south-1:533267094905:function:createUpdate-function",
  "Runtime": "python3.11",
  "Role": "arn:aws:iam::533267094905:role/lambdapollyrole",
  "Handler": "app.lambda_handler",
  "CodeSize": 731,
  "Description": "",
  "Timeout": 3,
  "MemorySize": 128,
  "LastModified": "2024-08-28T11:16:20.449+0000",
  "CodeSha256": "TgGbOzYdhebi02FwJ569xrYREnKirLoU8tIK2St47A=",
  "Version": "$LATEST",
  "Environment": {
    "Variables": {
      "TABLE_NAME": "Notes"
    }
  }
}
```

24. Likewise, you can create the other lambda functions. Just repeat the steps as you can see above.



Function name	Description	Package type	Runtime	Last modified
createUpdate-function	-	Zip	Python 3.11	2 minutes ago
dictate-function	-	Zip	Python 3.11	1 hour ago

Lambda > Functions

Last fetched 40 seconds ago Actions Create function

Filter by tags and attributes or search by keyword

< 1 >

	Function name	Description	Package type	Runtime	Last modified
<input type="checkbox"/>	list-function	-	Zip	Python 3.11	1 minute ago
<input type="checkbox"/>	createUpdate-function	-	Zip	Python 3.11	8 minutes ago
<input type="checkbox"/>	delete-function	-	Zip	Python 3.11	5 minutes ago
<input type="checkbox"/>	dictate-function	-	Zip	Python 3.11	1 hour ago
<input type="checkbox"/>	search-function	-	Zip	Python 3.11	2 minutes ago