

Developing Event-Driven Solutions with AWS Lambda

(LAB-M08-01)

Version Control	
Document	Developing Event-Driven Solutions with AWS Lambda
Owner	Ahmad Majeed Zahoory
Version	2.2
Last Change	27 th May 2024
Description of Change	Task steps updated

Lab duration: 40 minutes

Lab scenario

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

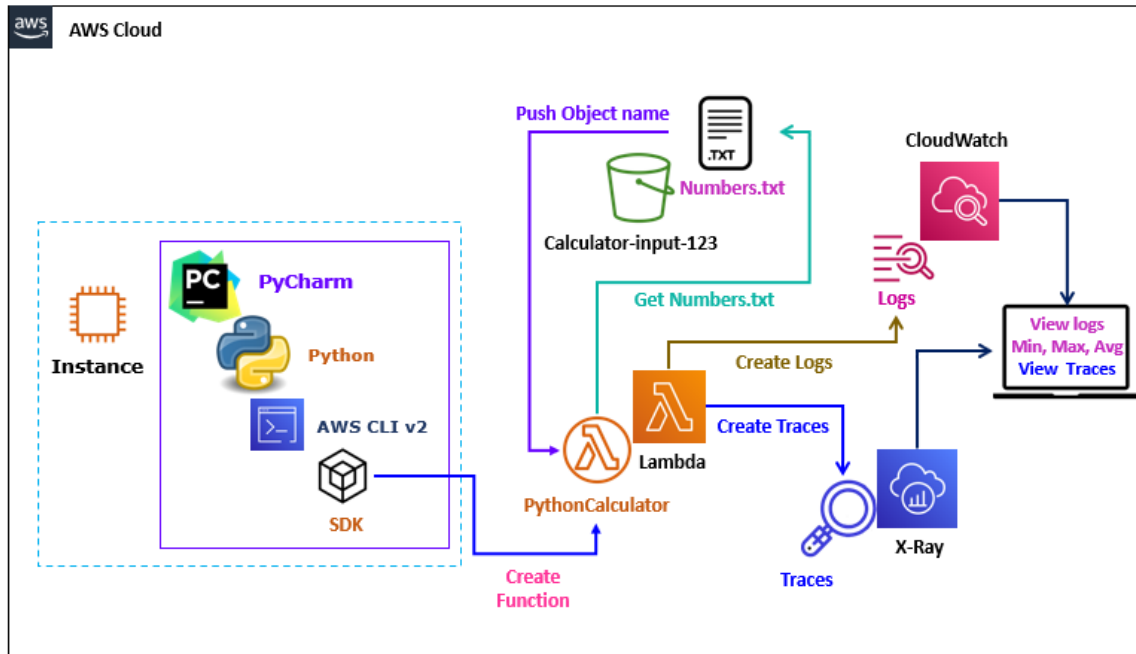
The Lambda function will calculate the minimum, maximum, and average of the numbers contained in an object uploaded to the Amazon S3 bucket.

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

Objectives

After you complete this lab, you will be able to:

- Create new lambda function.
- Create a new bucket.
- Upload object in the bucket.
- Get object from the bucket.
- Process object using lambda function.
- View the lambda logs in cloud watch.
- View the lambda traces in x-Ray.



Task 1: Manage IAM user and Role

In this task, you will update the AWS IAM role with permission to manage the S3 and Lambda.

Step 1: Update the IAM User Permission

1. In the **AWS Management Console**, on the **Services** menu, search and select **IAM**.
2. Select **Users**.
 - a. Open the **Dev-User-YOUR-ID**.
 - i. Select **Permissions**.
 - a) Select **AmazonEC2FullAccess**.
 - b) Select **AmazonDynamoDBFullAccess**.
 - 1) Select **Remove**.
 - I. Select **Remove Policies**.
3. From the **Dev-User** console:
 - a. Select **Permissions**.
 - i. Select **Add permissions**.
 - a) Select **Add permissions**.
 - b. In the **Add permissions** page:
 - i. **Permissions options**: Select **Attach policies directly**.
 - ii. **Permissions policies**:
 - a) Search and select **AmazonS3ReadOnlyAccess**.
 - b) Select **Next**.

1) Select **Add permissions**.

c. In the **Add permissions** page:

i. **Permissions options**: Select **Attach policies directly**.

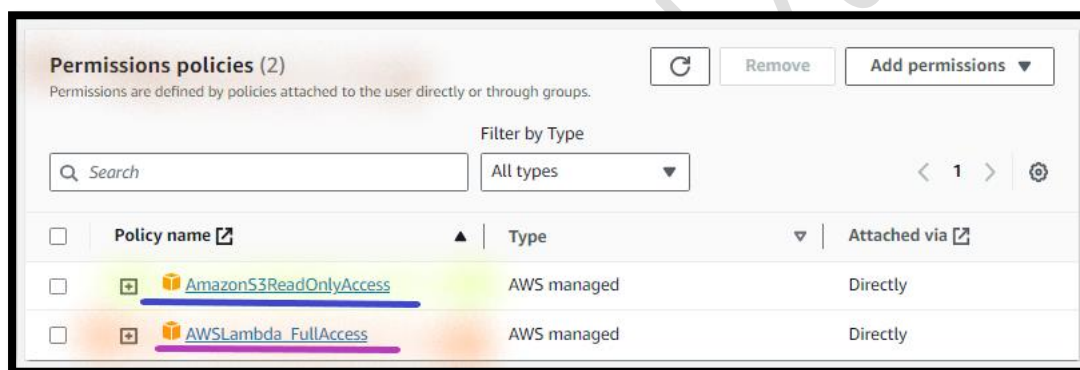
ii. **Permissions policies**:

a) Search and select **AWSLambda_FullAccess**.

b) Select **Next**.

1) Select **Add permissions**.

Note: You can see the **AmazonS3ReadOnlyAccess** and **AWSLambda_FullAccess** under the **Permissions policies**.



Step 2: Create IAM Role

4. From the **IAM** console.

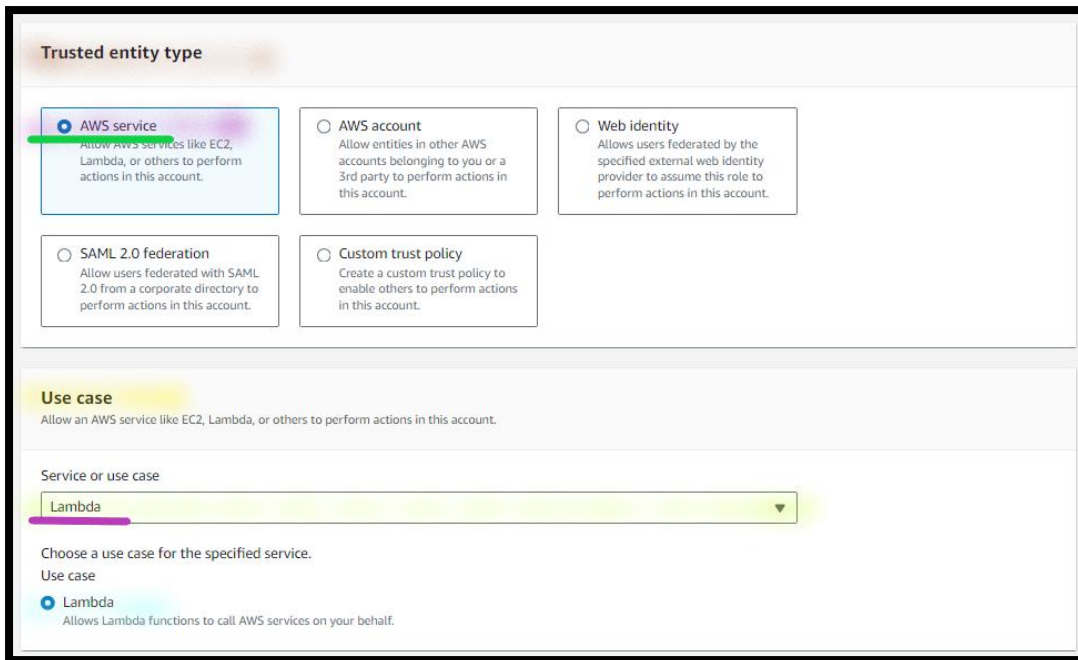
5. Select **Roles**.

a. Click on **Create role**.

i. In the **Select trusted entity** section.

a) **Trusted entity type**: Select **AWS service**.

b) **Use cases**: Dropdown and select **Lambda**.



Trusted entity type

- ☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
Lambda

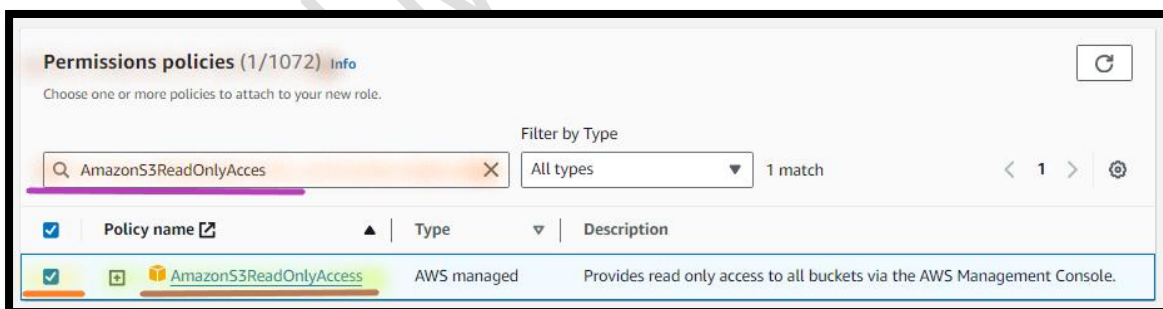
Choose a use case for the specified service.
Use case
☒ **Lambda**
Allows Lambda functions to call AWS services on your behalf.

c) Select **Next**.

ii. In the **Add permissions** section.

a) In the **Search box**, write **AmazonS3ReadOnlyAccess** and select **Enter Key**.

1) Select **AmazonS3ReadOnlyAccess**.



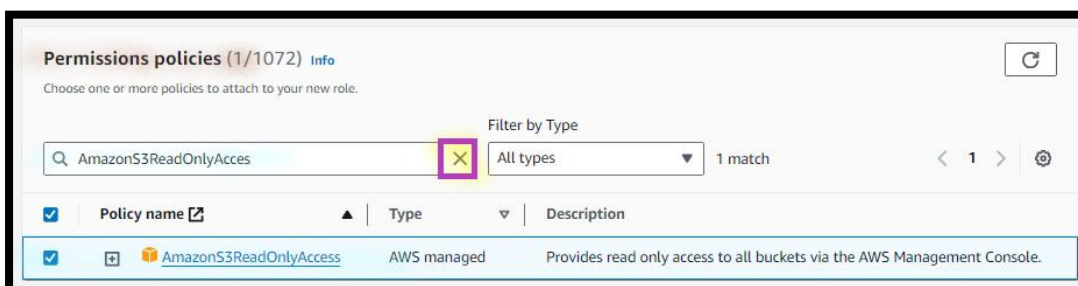
Permissions policies (1/1072) Info

Choose one or more policies to attach to your new role.

Filter by Type
All types 1 match

Policy name	Type	Description
AmazonS3ReadOnlyAccess	AWS managed	Provides read only access to all buckets via the AWS Management Console.

2) Select **Clear search query**.



Permissions policies (1/1072) Info

Choose one or more policies to attach to your new role.

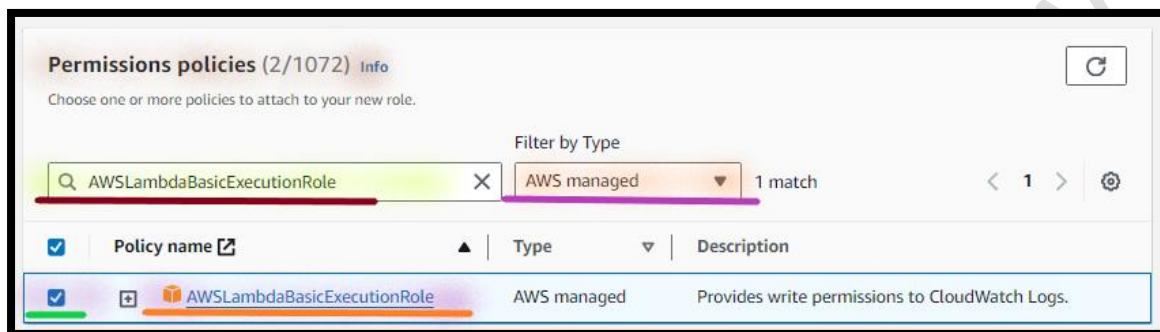
Filter by Type
All types 1 match

Policy name	Type	Description
AmazonS3ReadOnlyAccess	AWS managed	Provides read only access to all buckets via the AWS Management Console.

b) In the **Search box**, write **AWSLambdaBasicExecutionRole** and select **Enter Key**.

1) **Filter by type**: Dropdown and select **AWS managed**.

2) Select the **AWSLambdaBasicExecutionRole**.



c) Select **Next**.

ii. In the **Name, review, and create** section.

a) **Role name**: Write **Lambda-S3-Role-YOUR-ID**.

Note: You can see the **AmazonS3ReadOnlyAccess** and **AWSLambdaBasicExecutionRole** policy under the **Permissions Policy summary** section.

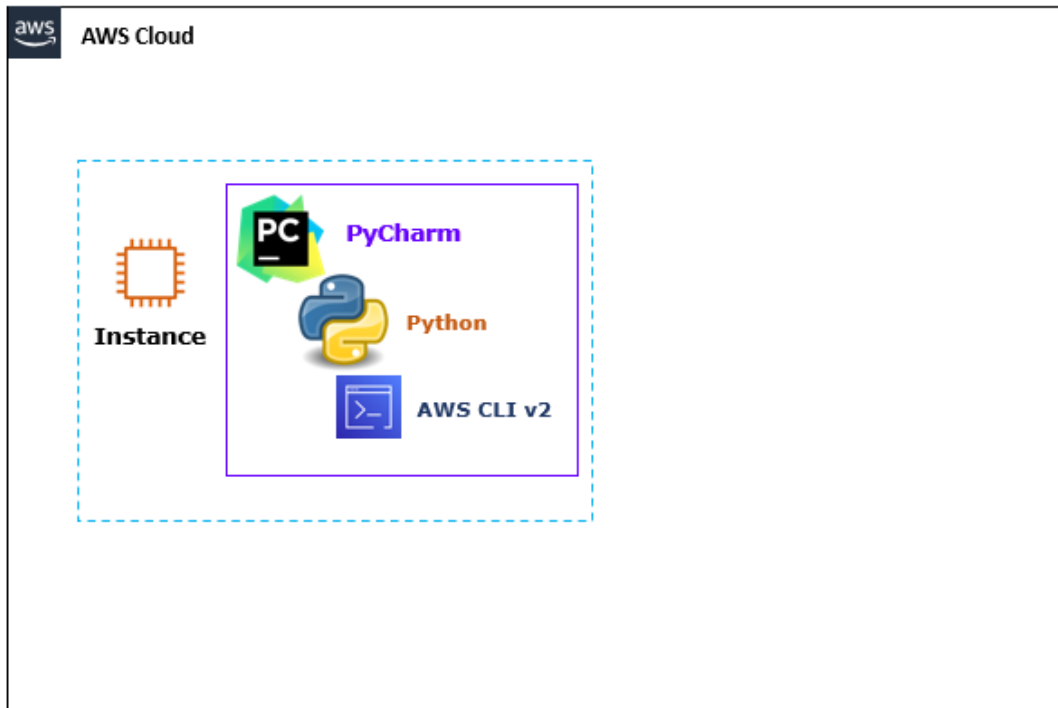
Permissions policy summary		
Policy name	Type	Attached as
AWSLambdaBasicExecutionRole	AWS managed	Permissions policy
AmazonS3ReadOnlyAccess	AWS managed	Permissions policy

b) Click **Create role**.

Note: **Wait**, till you can see the **message "Role Lambda-S3-Role created"**.

Task 2: Build Server for Development Environment

In this task, you will build the development environment with Python, PyCharm and AWS CLI.



Step 1: Create EC2 Instances

6. In the **AWS Management Console**, on the **Services** menu Search and Select **CloudFormation**.
7. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
8. Select **Create stack** and configure:
 - a. In the **Create stack** page:
 - i. **Prepare template**: Select **Template is ready**.

The screenshot shows the 'Prerequisite - Prepare template' section of the AWS CloudFormation console. It includes a description: 'Prepare template. Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.' Below this, there are three radio button options: 'Template is ready' (which is selected), 'Use a sample template', and 'Create template in Designer'.

- ii. **Template source**: Select **Upload a template file**.
- iii. **Choose file**: Click on **Choose file**.

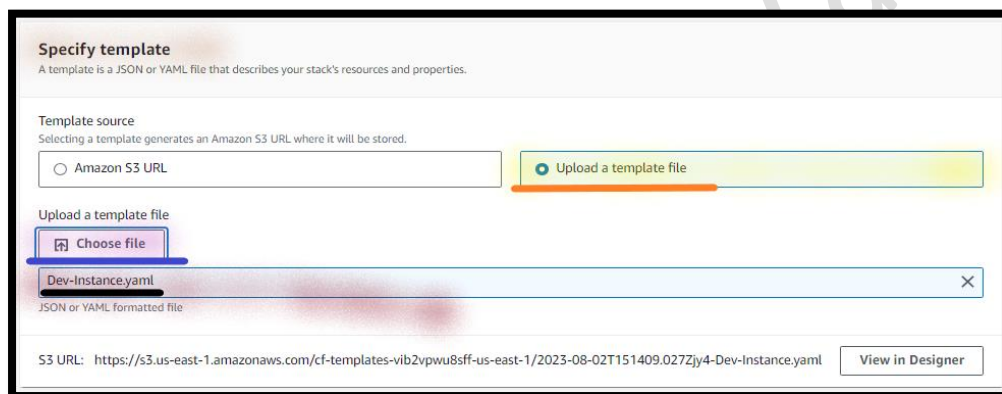
- a) **Navigate** and **select** the **Dev-Instance.yaml** file.

Note: **Dev-Instance.yaml** template is provided with the Lab manual.

Note: AWS template **performing** the **following** tasks:

1. **Creating Windows instances.**
2. **Creating t2.medium** instance (2 vCPU and 4 GB) [*This instance type attract charges*].
3. **Set** the **Administrator's Password.**

Note: You can also use **t2.micro**, but the **performance will be low** to build development environment.



Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL ☒ Upload a template file

Upload a template file


Dev-Instance.yaml
JSON or YAML formatted file

S3 URL: <https://s3.us-east-1.amazonaws.com/cf-templates-vib2vpwu8sff-us-east-1/2023-08-02T151409.027Zjy4-Dev-Instance.yaml>

- iv. Select **Next**.

- b. In the **Specify stack details** page:

- i. **Stack name:** Write **Dev-Instance-PY**.



Stack name

Stack name

Dev-Instance-PY

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Note: Leave other details as default.

- ii. Select **Next**.

- c. In the **Configure stack options** page:

Note: Leave all the details as default.

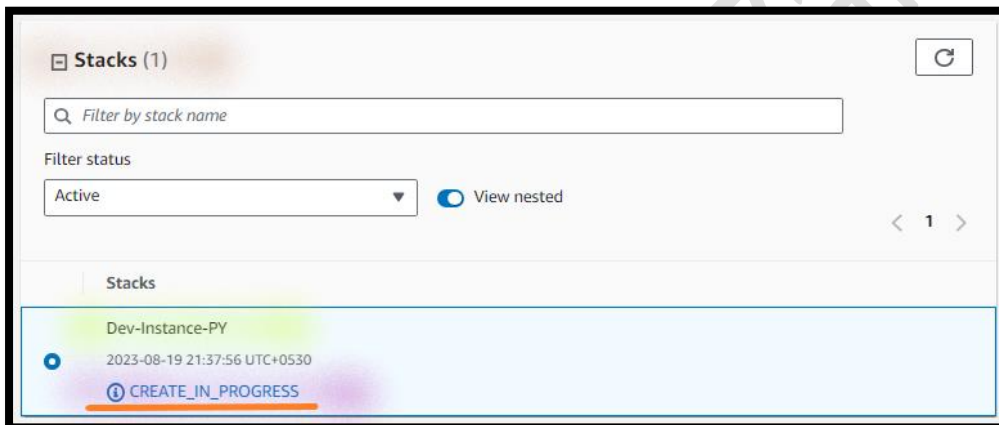
- i. Select **Next**.

- d. In the **Review Dev-Instance-PY** page:

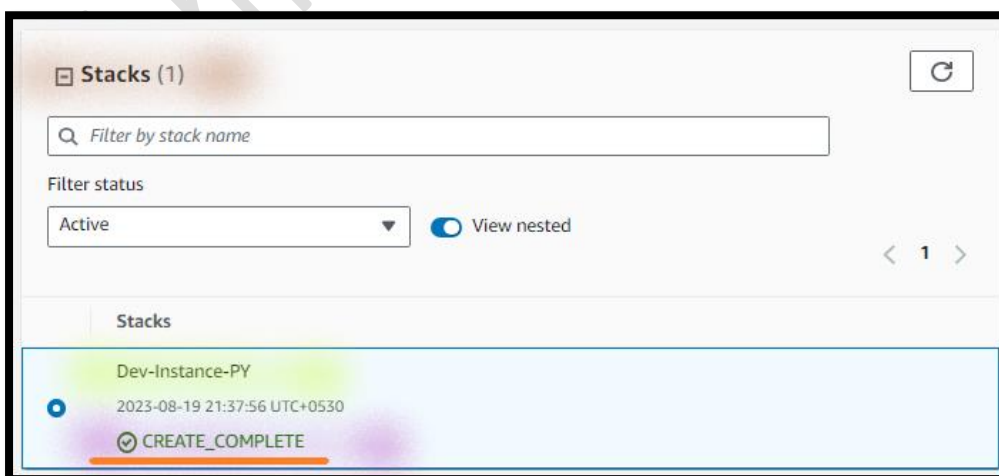
Note: Review all the details.

- i. Select **Submit**.

Note: You can see the **Stack** status as **CREATE_IN_PROGRESS**.



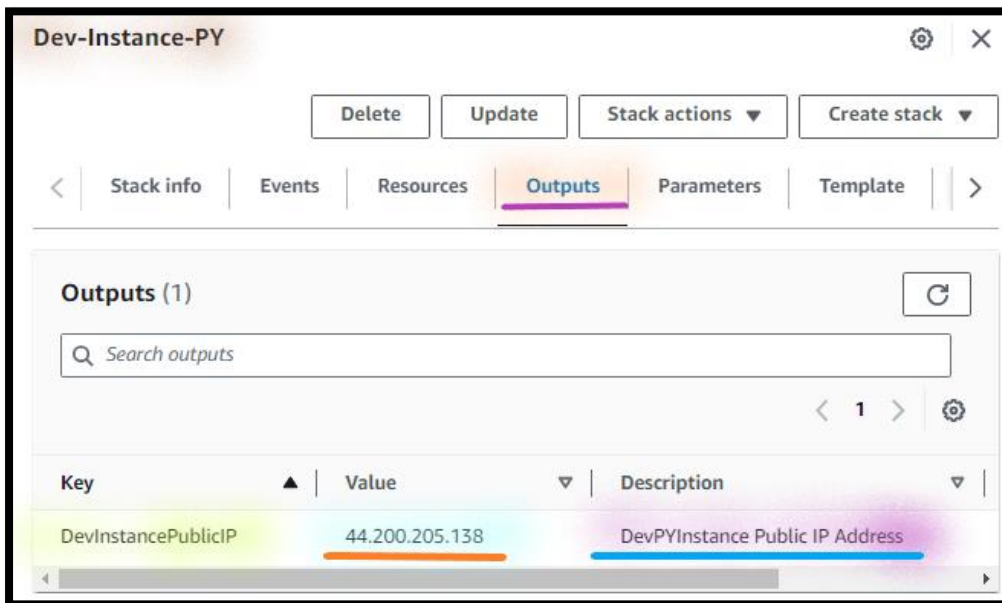
Note: **Wait**, till you can see the **Stack** status as **CREATE_COMPLETE**. You can **Refresh** your screen



Step 2: View the Output

9. From the **Dev-Instance-PY CloudFormation** console:
 - a. Select **Outputs**.

Note: Copy the **DevPYInstance Public IP** address in the **Notepad**.



Step 3: Connect to Instance

10. From the **Local Desktop/ Laptop** (*Windows Desktop*), right click on **Start** & **Run**.
 - a. In the **Open**, write **mstsc**.
 - b. Select **Ok**.
 - i. From the **Remote Desktop Connection**:
 - a) **Computer**: Write the **Public IP Address** of the **DevPYInstance**.
 - b) Select **Connect**.

Note: You can **get the prompt** to enter the **Username** and **Password**.

- 1) **Username:** Write **Administrator**.
- 2) **Password:** Write **lab-password@123**.
- 3) Select **Ok**.

Step 4: Install the Python

11. From the **DevPYInstance** (*Windows Server 2022*).

- a. **Download** the **Python 3.11** for **Windows x64**.

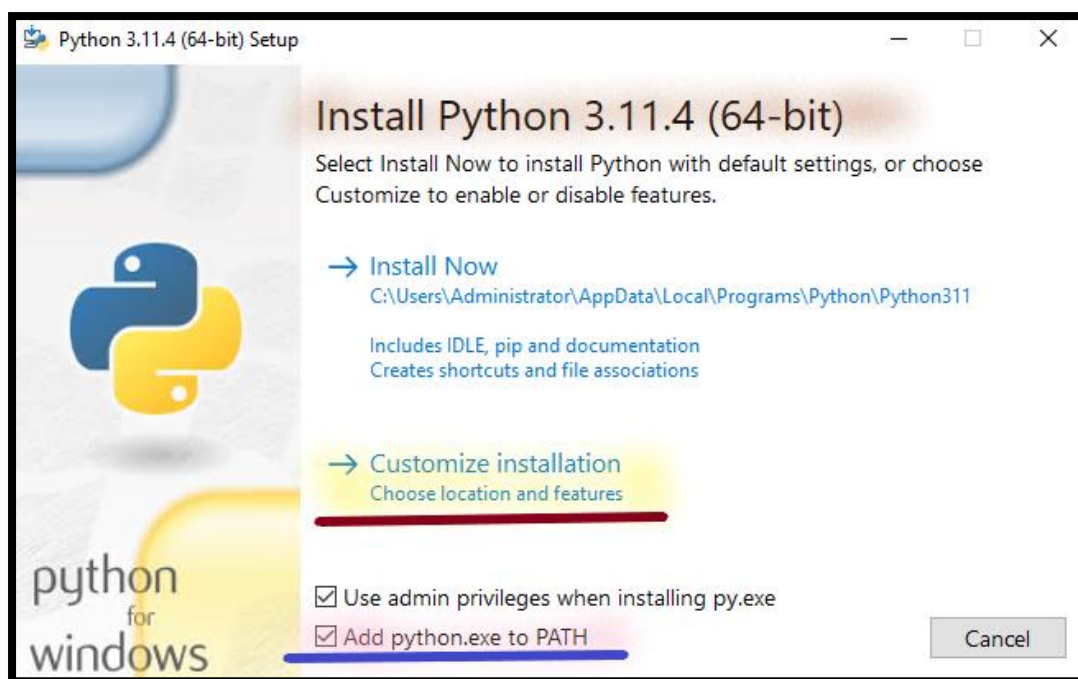
Note: Use the below URL to download the **Python 3.11** for **Windows**.

<https://bitbucket.org/ahmadzahoory/dev/downloads/python-3.11.4-amd64.exe>

Info: You can also download the Python 3.11 from the Python.org site.

<https://www.python.org/downloads/release/python-3119/>

- b. **Install** the **Python** for **Windows x64**.
 - i. From the **Install Python 3.11** section:
 - a) **Add python.exe to PATH:** **Enable** the **Checkmark**.
 - b) Select **Custom installation**.

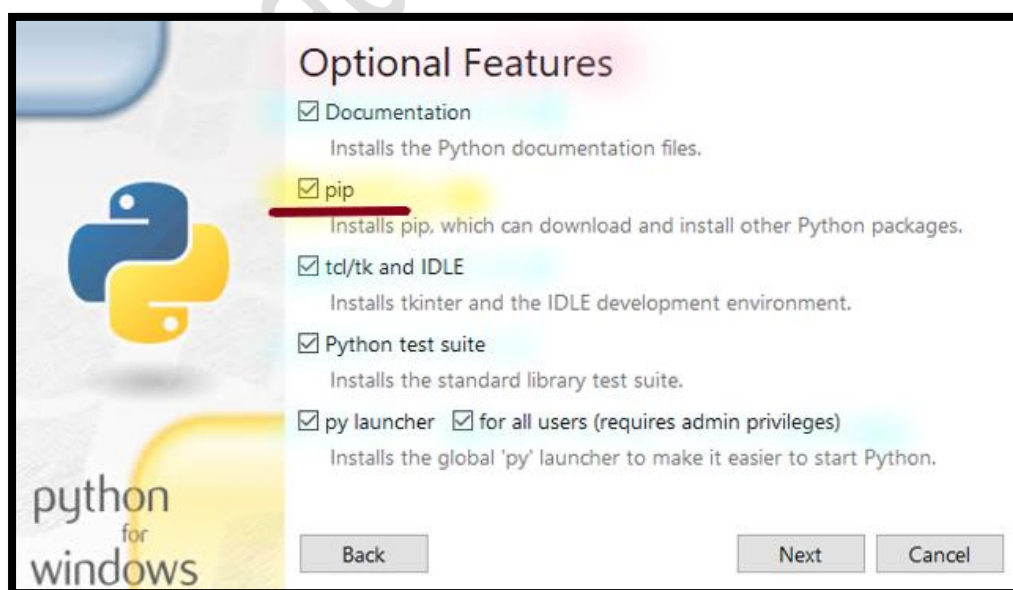


ii. From the **Optional Features** section:

a) Select **All Options**.

Note: Leave all the options as default.

Note: Ensure **pip** must be **selected**.



a) Select **Next**.

iii. From the **Advanced Options** section:

Note: Leave all the options as default.

1) Select **Install**.

Note: Wait, till **Python 3.11** install **successfully**.

Step 5: Check the Python and Pip Version

12. From the **DevPYInstance**, right click on **Start** & **Run**.

a. In the **Open**, write **cmd**.

b. Select **Ok**.

i. From the **Command line interpreter**:

a) **Execute** the **below command** to **verify** the **Python version**:

```
py --version
```

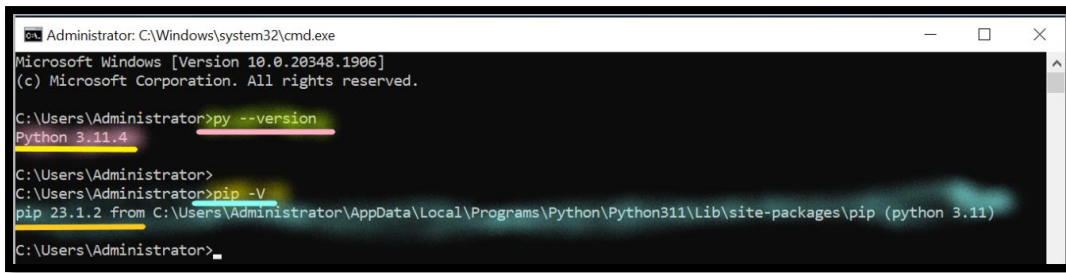
Note: You can see the **Python** installed **version**.

ii. From the **Command line interpreter**:

a) **Execute** the **below command** to **verify** the **PIP version**:

```
pip -V
```

Note: You can see the **Pip** installed **version**.



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.20348.1906]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>py --version
Python 3.11.4

C:\Users\Administrator>
C:\Users\Administrator>pip -V
pip 23.1.2 from C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packages\pip (python 3.11)

C:\Users\Administrator>
```

Step 6: Install the PyCharm IDE

13. **Download** and **Install** the **PyCharm IDE 2024** for **Community Edition**.

Note: Use the below **URL** to download the **PyCharm IDE**.

<https://bitbucket.org/ahmadzahoory/devenv/downloads/pycharm-community-2024.1.1.exe>

Info: You can also download the Pycharm 2024 from the jetbrains.com site.

<https://www.jetbrains.com/pycharm/download/>

Note: **Wait**, till **PyCharm IDE** install **successfully**.

Note: **Don't launch** the **PyCharm IDE**.

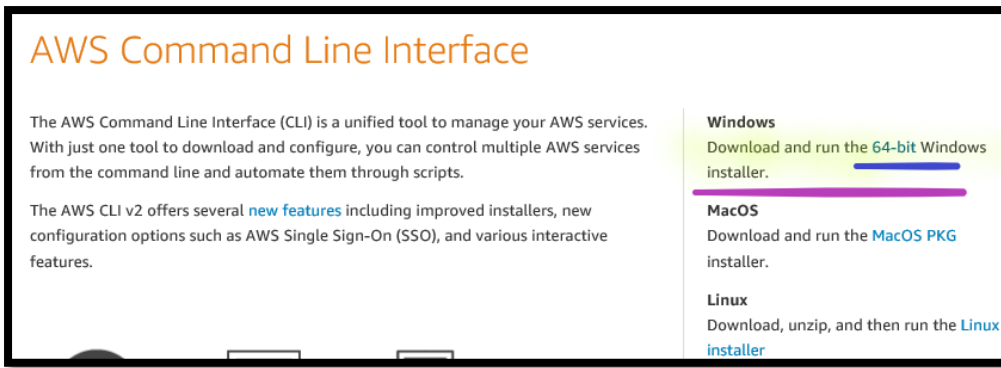
Step 7: Install the AWS CLI V2

14. From the **DevPYInstance**.

- a. **Download** and **install** the **AWS CLI v2**.

Note: Use the below URL to download the **AW CLI v2**.

<https://aws.amazon.com/cli/>



Note: Wait, till **AWS CLI v2** install **successfully**.

Step 8: Check the AWS CLI Version

15. From the **DevPYInstance**, right click on **Start** & **Run**.

a. In the **Open**, write **cmd**.

b. Select **Ok**.

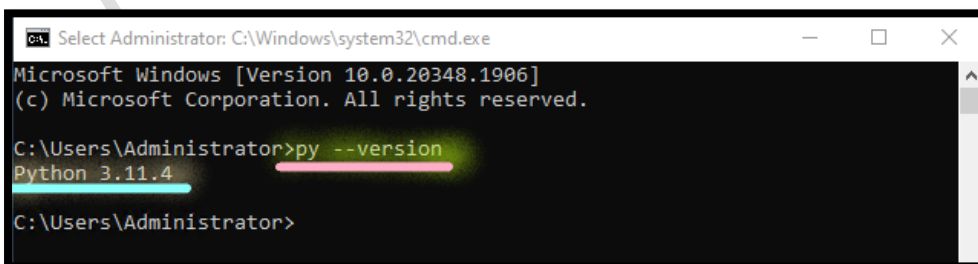
i. From the **Command line interpreter**:

a) **Execute** the **below command** to **verify** the **AWS version**.

```
aws --version
```

Note: You can see the **AWS CLI** installed **version**.

Note: If you can see the "'aws' is not recognized as an internal or external command" message, **Restart** the **DevPYInstance**.



Step 9: Configure the Credentials and Configuration

16. From the **DevPYInstance**, right click on **Start** & **Run**.

a. In the **Open**, write **cmd**.

b. Select **Ok**.

i. From the **Command line interpreter**:

a) **Execute** the **below command** to **configure** the **AWS credentials**.

```
aws configure
```

a) **AWS Access Key ID**: Type **Dev-User-YOUR-ID's**, **access key** (copy from the .csv file), press **Enter** key to continue.

b) **AWS Secret Access Key**: Type **Dev-User-YOUR-ID's**, **secret access key** (copy from the .csv file), press **Enter** key to continue.

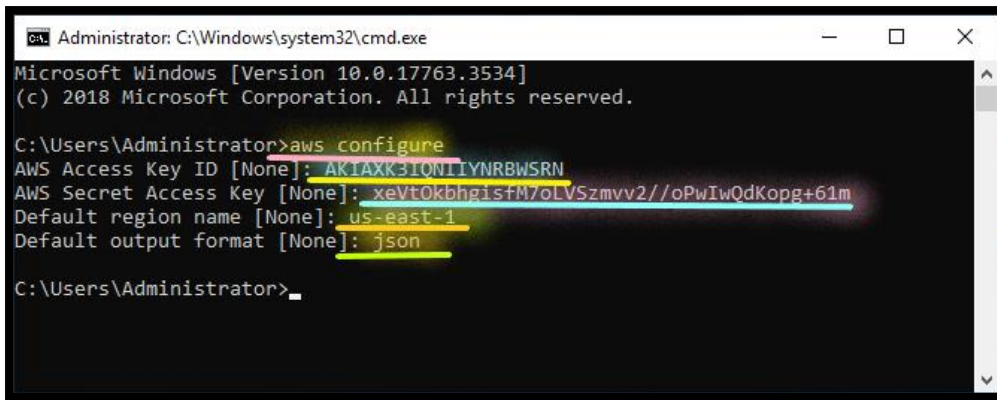
Note: Copy the **access key** and **secret access key** of the IAM user **Dev-User** from **.csv file** which you have downloaded in the previous step.

c) **Default region name**: Type **YOUR ALLOCATED REGION-ID**, press **Enter** key to continue.

Note: **Replace** the **region-identifier**.

Refer the **link** to know your **respective region identifier**
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html>

d) **Default output format**: Type **json**, press **Enter** key to continue.



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.3534]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>aws configure
AWS Access Key ID [None]: AKIAXK3IQNIYNRBWSRN
AWS Secret Access Key [None]: xeVtOkbhgiStM7oLVSzmvv2//oPwIwQdKopg+61m
Default region name [None]: us-east-1
Default output format [None]: json

C:\Users\Administrator>
```

- b) **Execute** the **below command** to **exit**.

exit

Step 10: Verify the Configuration

17. From the **DevJPyInstance**, right click on **Start** & **Run**.

- In the **Open**, write **C:\Users\Administrator**.
- Select **Ok**.
 - From the **File explorer**:
 - Open** the **.aws** folder.
 - Open** the **Credentials** file in **Notepad**.

Note: You can see the **access key** and **secret access key** details.

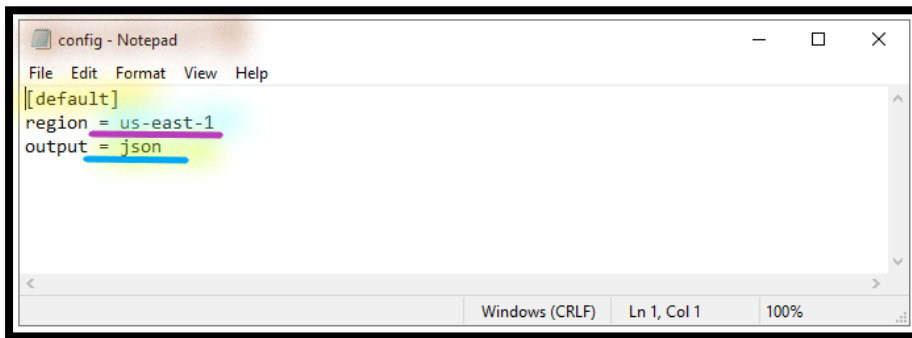


```
credentials - Notepad
File Edit Format View Help
[[default]]
aws_access_key_id = AKIAXK3IQNIYNRBWSRN
aws_secret_access_key = xeVtOkbhgiStM7oLVSzmvv2//oPwIwQdKopg+61m
```

- 1) Select **File**.
- 2) Select **Exit**.

b) **Open** the **Config** file in **Notepad**.

Note: You can see the **region** and **output** format details.



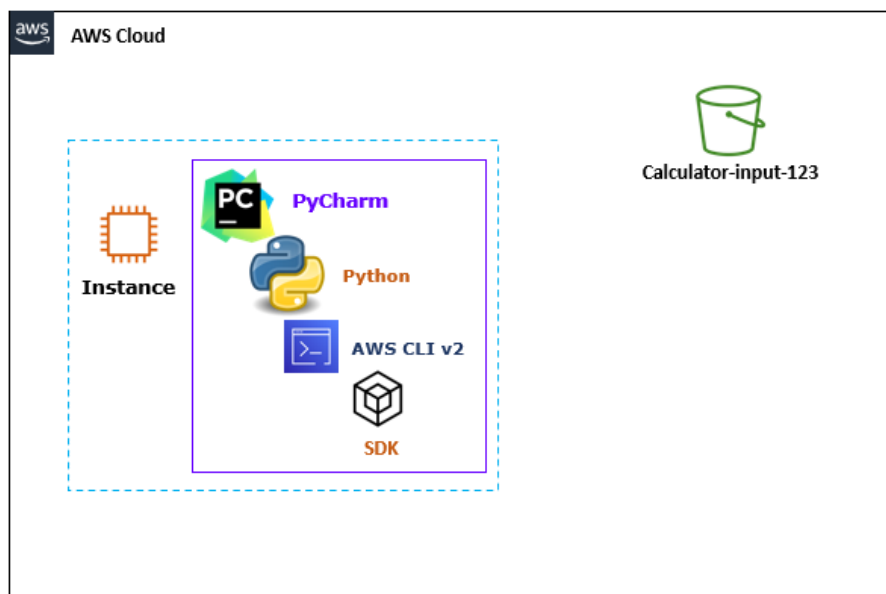
1) Select **File**.

2) Select **Exit**.

c) **Close** the **File explorer**.

Task 3: Create AWS S3 Bucket

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



Step 1: Create a Bucket

18. In the **AWS Management Console**, on the **Services** menu, search and select **S3**.
19. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
20. Select **Create bucket**.
 - a. In the **General Configuration** section:
 - i. **Bucket type**: Select **General purpose**.
 - ii. **Bucket name**: Write **calculator-input-123**.

Note: Replace **123** with a **random number** to make bucket name unique.

Note: Leave other details as default.

- b. Select **Create bucket**.

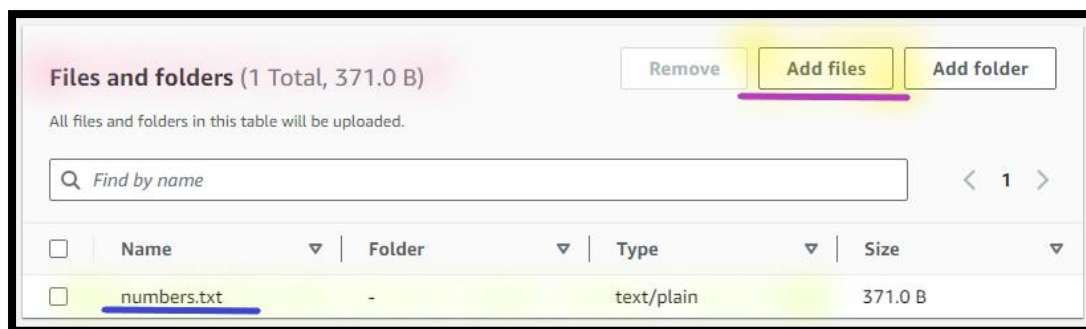
Note: **Wait**, till bucket **calculator-input-123** gets **created**.

Step 2: Upload Content in the Bucket

21. From the **S3** console.
22. Open **calculator-input-123** bucket.
 - a. Select **Objects**.
 - i. Click **Upload**.
 - a) Click **Add files**.
 - 1) **Navigate** and select **numbers.txt** file.
 - I. Select **Open**.

Note: **numbers.txt** file is provided with the **Lab manual**.

Note: Once uploaded you can see the **numbers.txt** file under **files and folders** section.

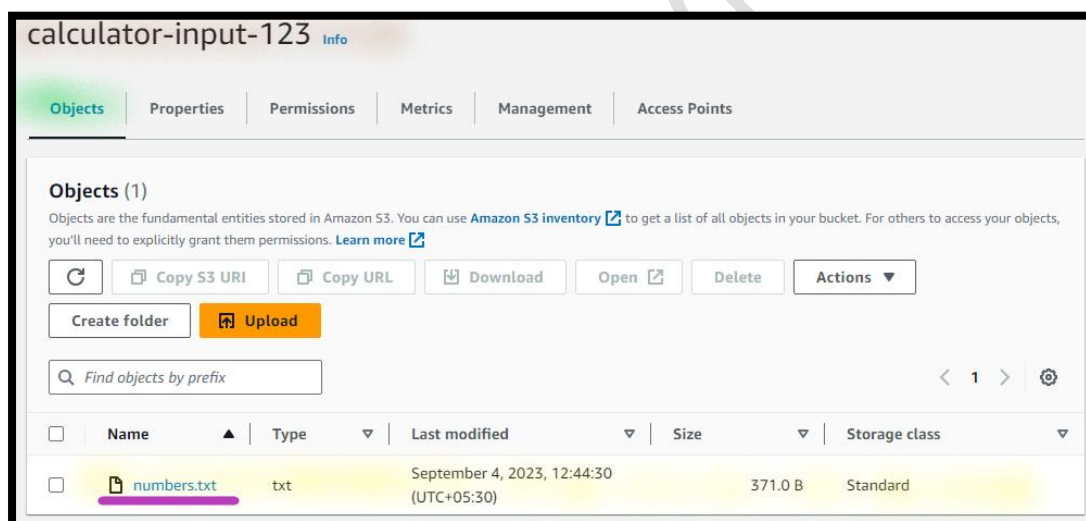


i. Select **Upload**.

Note: Wait, till file gets **uploaded successfully**.

ii. Select **Close**.

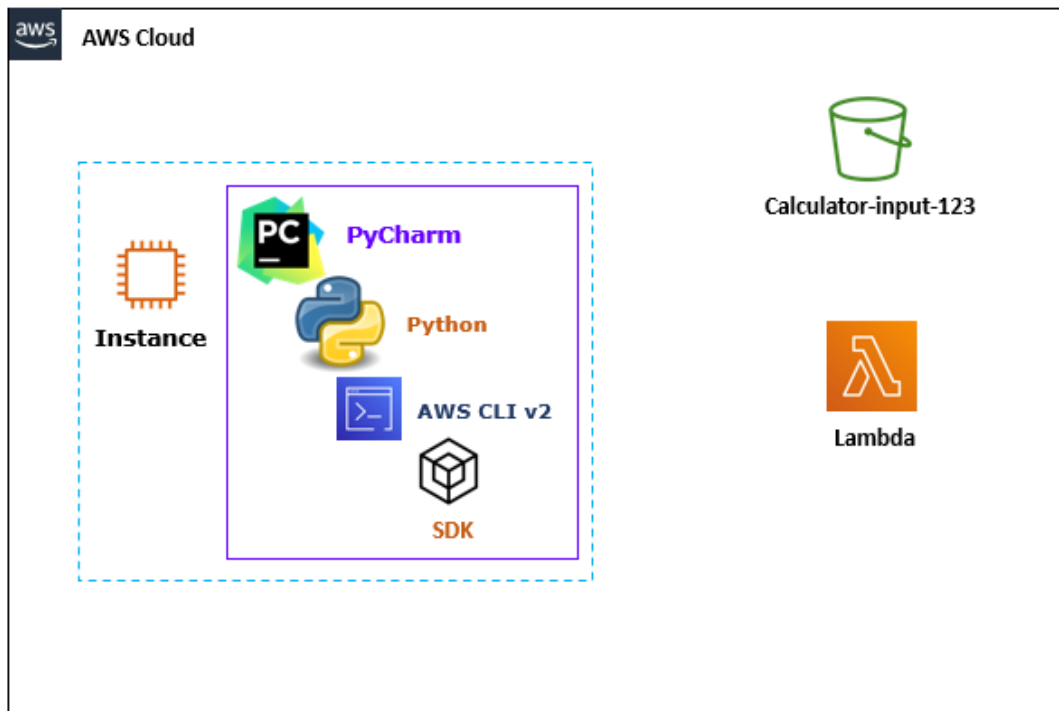
Note: You can see the **numbers.txt** file under Objects.



Task 4: Create Lambda Function

In this task, 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.



Step 1: Create Lambda Function

23. In the **AWS Management Console**, on the **Services** menu, search and select **Lambda**.
24. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
25. Select **Create a function**.
26. Select **Author from scratch** and configure:
 - a. In the **Basic Information** section:
 - i. **Name:** Write **PythonCalculator**.
 - ii. **Runtime:** Dropdown and select **Python 3.11**.

Basic information

Function name
Enter a name that describes the purpose of your function.
PythonCalculator
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.
Python 3.11

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

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.

iii. **Expand Change default execution role.**

a) **Execution Role:** Select **Use an existing role.**

1) **Existing role:** Dropdown and select **Lambda-S3-Role.**

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](#).
☐ 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.
Lambda-S3-Role
[View the Lambda-S3-Role role](#) on the IAM console.

b. Select **Create function.**

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

Step 2: Configure Lambda Function

27. From the **PythonCalculator** lambda function.

a. In the **Code** section:

Code | Test | Monitor | Configuration | Aliases | Versions

- i. Select **Runtime settings** sub section:



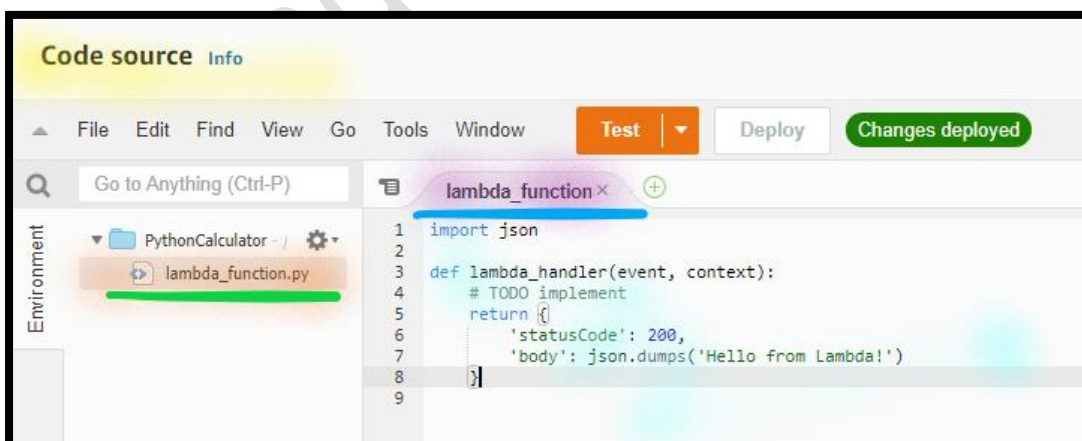
Note: You can see the **Handler** name as `lambda_function.lambda_handler`. If **name is different**, update the name to `lambda_function.lambda_handler`.

Info: Lambda console is `lambda_function.lambda_handler`. This function **handler name** reflects the function name (`lambda_handler`) and the file where the **handler code** is stored (`lambda_function.py`).

- ii. In the **Code source** sub section:

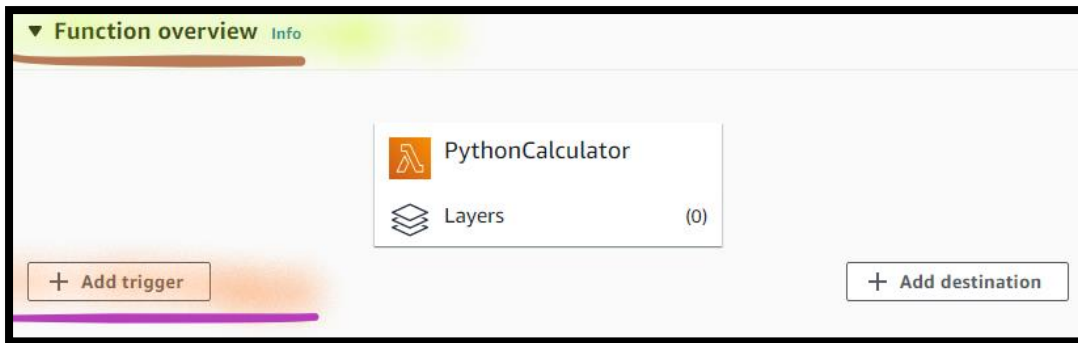
- a) **Double-click** on `lambda_function.py`.

Note: You can see the **Default code**.

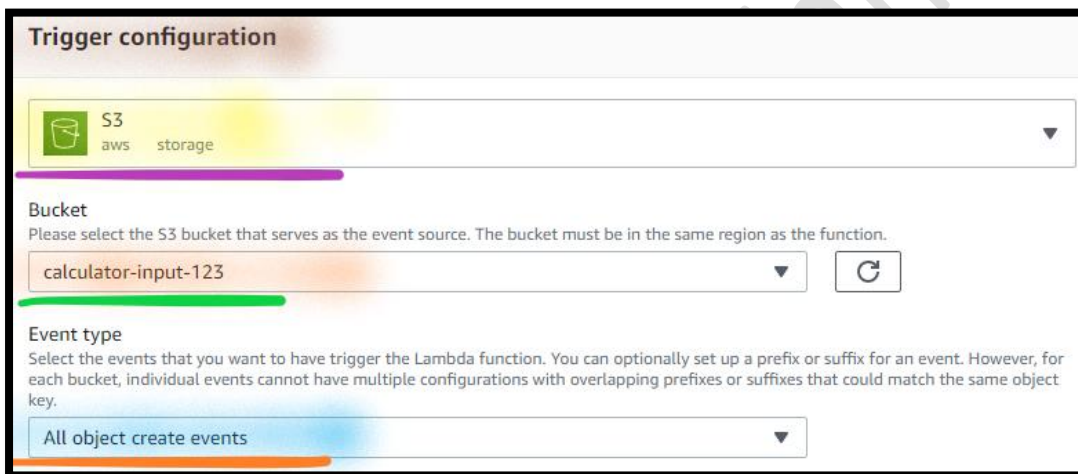


- b. In the **Function overview** section.

- i. Select **+ Add trigger**.

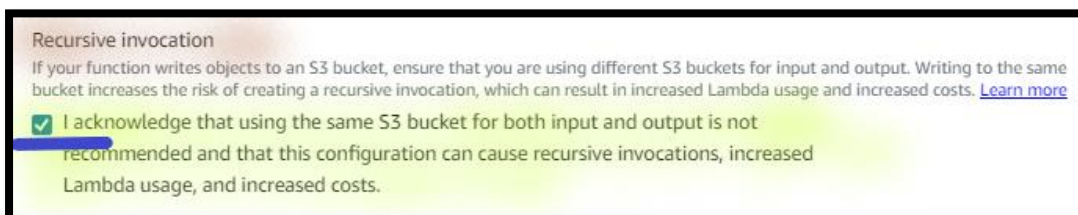


- a) **Trigger configuration:** Dropdown and select **S3**.
- b) **Bucket:** Dropdown and select **calculator-input-123**.
- c) **Event type:** Dropdown and select **All object create events**.



Info: For these **trigger settings**, the **lambda function** will **run** whenever an **object** is **created** in your **S3** bucket.

- d) **Recursive invocation:** **Enable** *I acknowledge that using the same Amazon S3 bucket*



- e) Select **Add**.

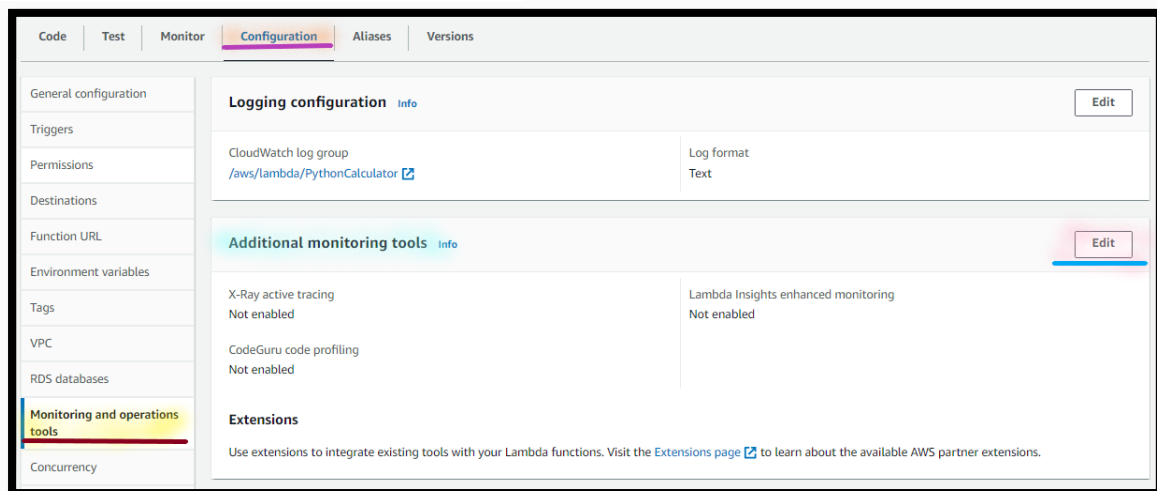
Note: You can see the **message "The function is now receiving events from the trigger"**.

c. In the **Code** section:

i. Select **Configuration** sub section:

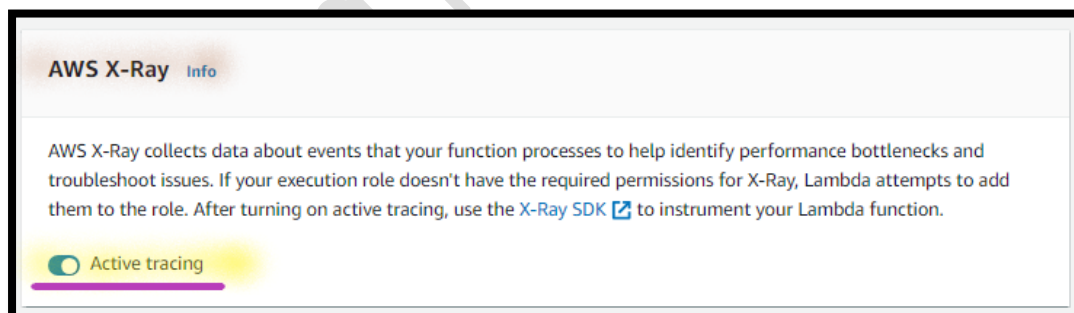
a) Select **Monitoring and operations tools**.

1) **Additional monitoring tool:** Select **Edit**.



I. **AWS X-Ray:**

A. **Active tracing:** **Enable** the same.



b) Select **Save**.

Note: **Go to next Task**, But **Don't Close** the **lambda console**.

Task 5: Developing the Python Application

In this task, you will develop the Python application for Lambda.

Step 1: Check the Python and PIP Version

28. **Return** to the **DevPYInstance** console.

29. **From** the **DevPYInstance**, right click on **Start** & **Run**.

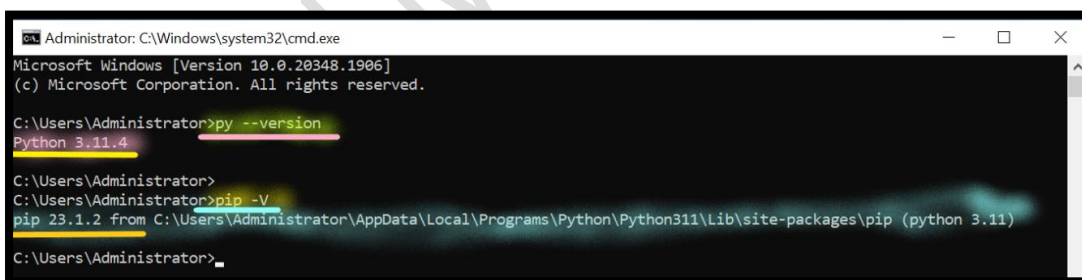
- a. In the **Open**, write **cmd**.
 - i. Press **Ok**.
 - ii. From the **command line interpreter**:
 - a) Write **py --version**.

Note: You can see the **Python version**.

iii. From the **command line interpreter**:

- a) Write **pip -V**.

Note: You can see the **pip version**.



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.20348.1906]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>py --version
Python 3.11.4

C:\Users\Administrator>
C:\Users\Administrator>pip -V
pip 23.1.2 from C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packages\pip (python 3.11)

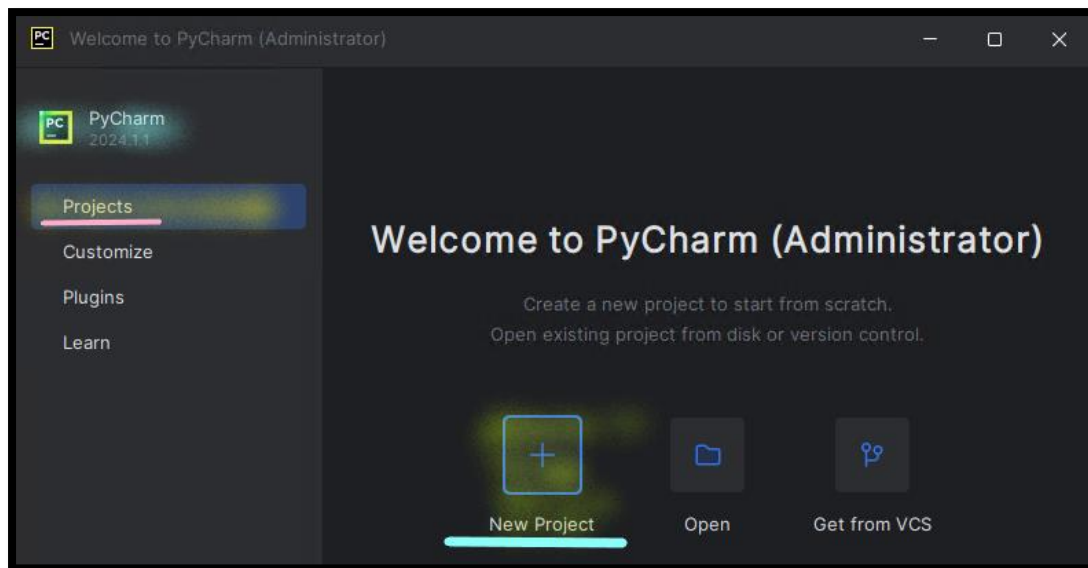
C:\Users\Administrator>
```

Step 2: Launch the PyCharm IDE

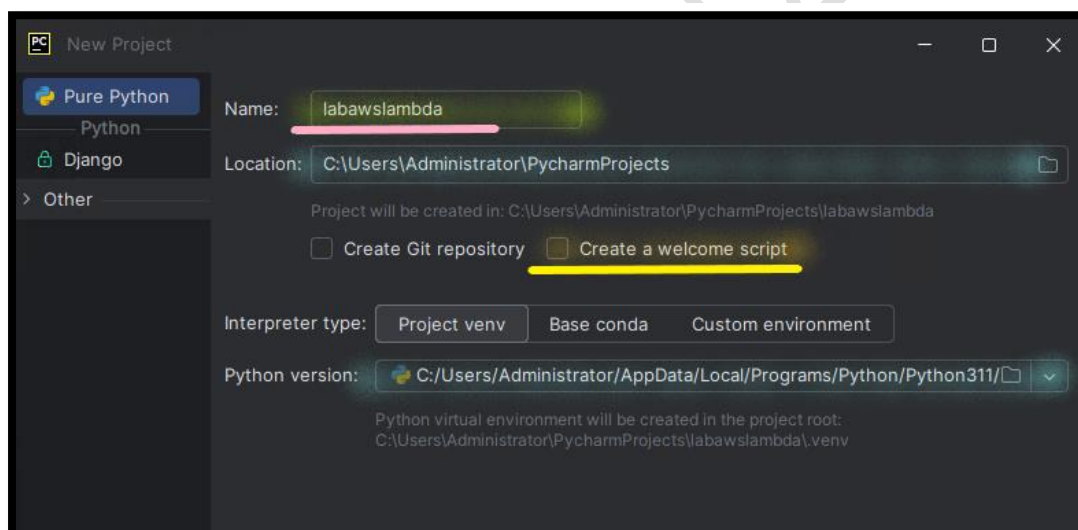
30. From the **DevPYInstance**.

31. Open the **PyCharm IDE**.

- a. In the **PyCharm IDE**:
 - i. Select the **New Project**.



- a) **Name:** Write **labawslambda**.
- b) **Uncheck** the **Create a welcome script**.



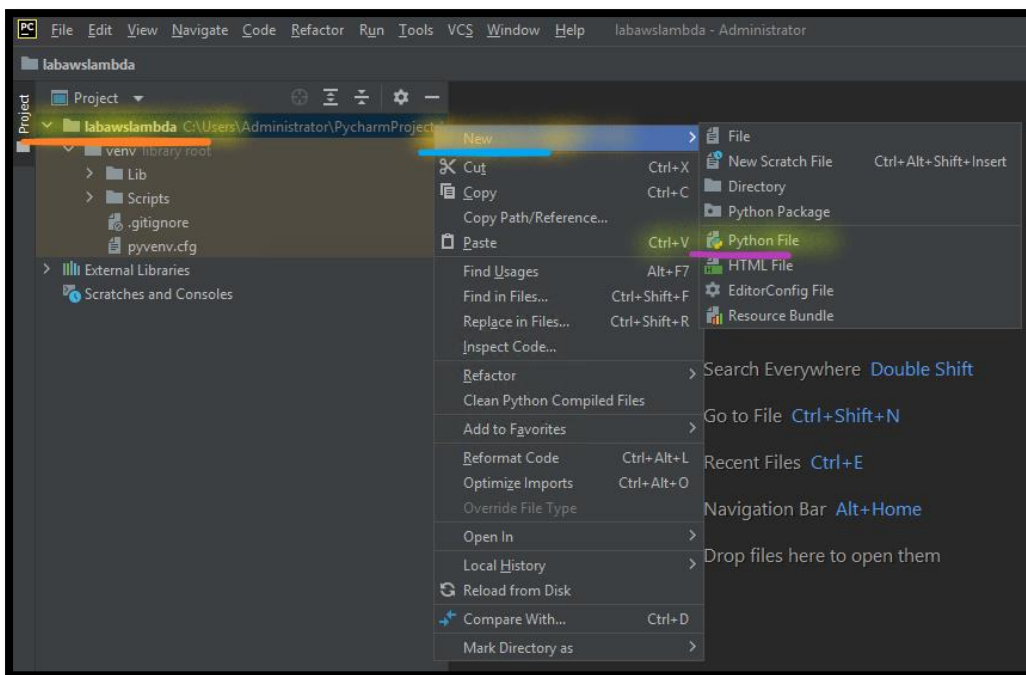
- ii. Select **Create**.

Note: **Wait**, till **Virtual environment** gets **created**.

Step 3: Create the File in the Python Project

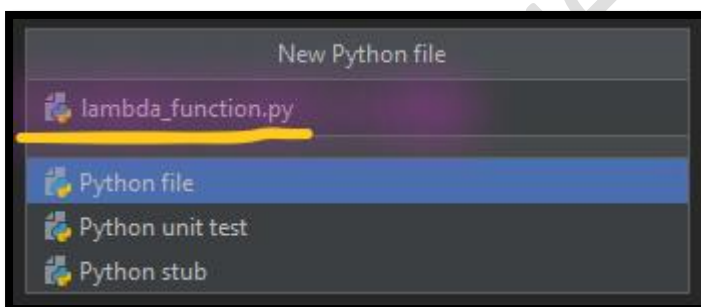
32. **Expand** the **labawslambda** Python project.
 - a. **Right-click** on the **labawslambda** Python project.
 - i. Select **New**.

a) Select **Python File**.



b) In the **New python file** page:

1) **File name:** Write **lambda_function.py**.



2) Select **Enter**.

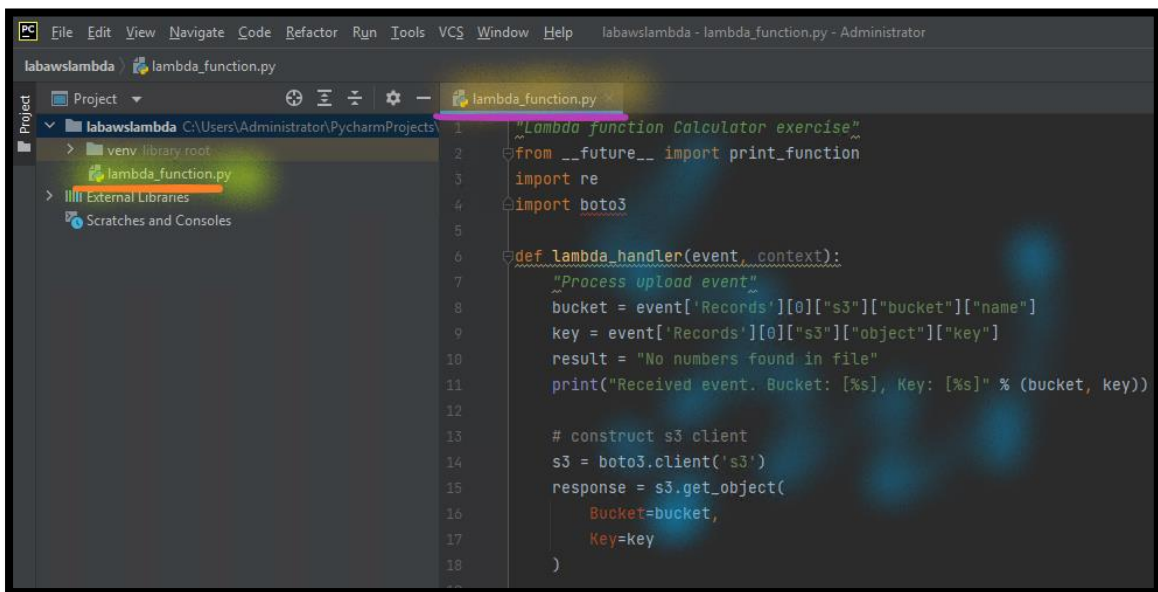
Note: You can see the **lambda_function.py** under Python package.

Step 4: Update the Python Code

33. **Double-click** on the **lambda_function.py** Python file.
 - a. **Paste** the **Code** from **lambda_function.py** file.

Note: **lambda_function.py** file is available with **Lab manual**.

Note: **Ignore** (if) **Python version compatibility** notification.



```
1  """Lambda function Calculator exercise"""
2  from __future__ import print_function
3  import re
4  import boto3
5
6  def lambda_handler(event, context):
7      """Process upload event"""
8      bucket = event['Records'][0]['s3']['bucket']['name']
9      key = event['Records'][0]['s3']['object']['key']
10     result = "No numbers found in file"
11     print("Received event. Bucket: [%s], Key: [%s]" % (bucket, key))
12
13     # construct s3 client
14     s3 = boto3.client('s3')
15     response = s3.get_object(
16         Bucket=bucket,
17         Key=key
18     )
```

Info: Take a moment to **familiarize** yourself with the file. The **code** is designed to:

1. **Respond** to an Amazon **S3 bucket event**.
2. **Retrieve** a **file** from Amazon **S3 bucket**.

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 statements in a few places in the code.

- b. **From** the **PyCharm IDE**:
 - i. Press **CTRL + S**.

Task 6: Build and Test the Lambda function locally

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

Step 1: Update the Lambda Function

34. From the **PyCharm IDE**:

- a. In the **lambda_function.py** file, locate **TODO 1** section:
 - i. Update the **REPLACE WITH BUCKET NAME** with the **Bucket name**, which you have created in the previous step.

Note: Don't remove the starting and end double quote.

Note: Object name **numbers.txt** is already mentioned in the **TODO 1** section.

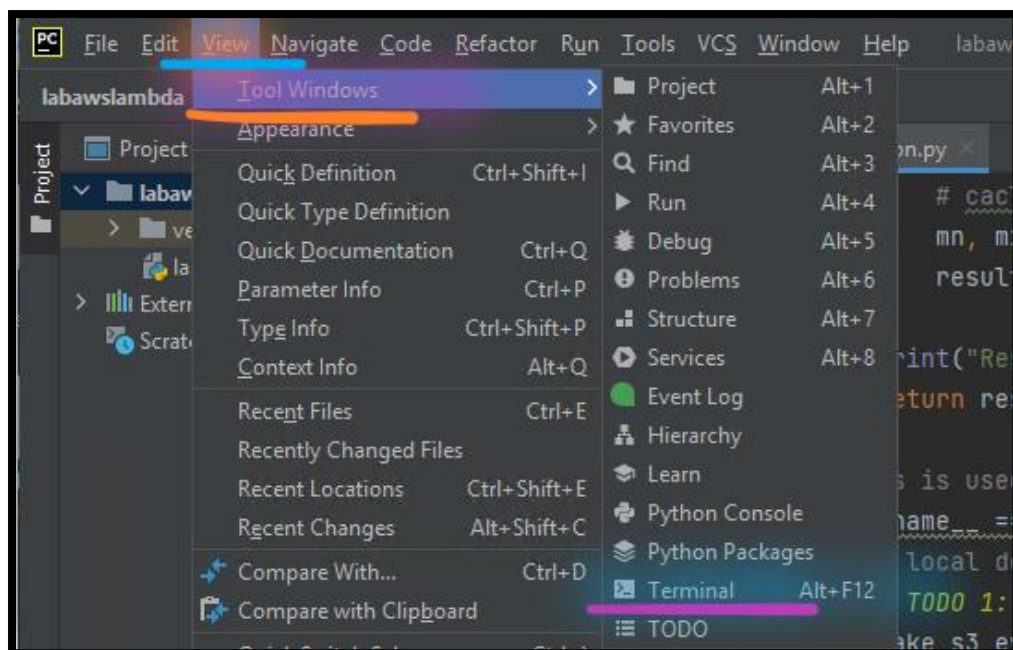
- b. From the **PyCharm IDE**:
 - i. Press **CTRL + S**.

Note: This is a **simulated event** you will use to test the **Python code** in **local development**.

Step 2: Install the Python SDK

35. From the **PyCharm IDE**.

- a. Select **Menu**.
 - i. Select **View**.
 - a) Select **Tool Windows**.
 - 1) Select **Terminal**.

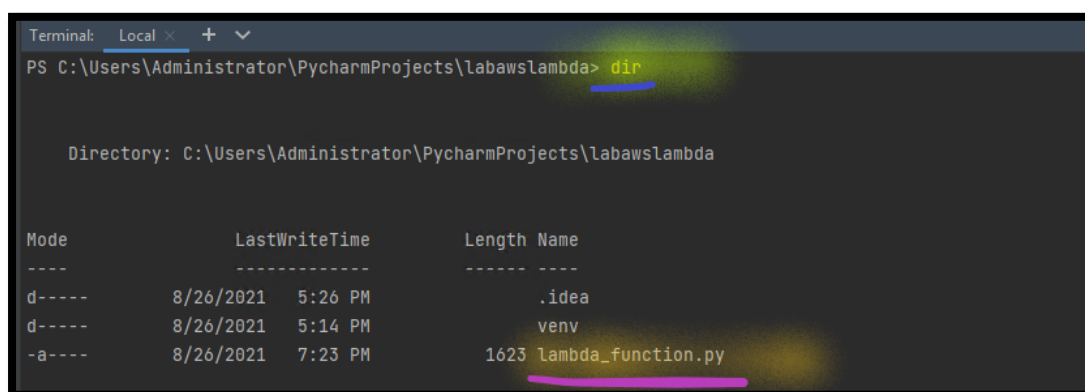


Note: This will **Open** the **Terminal** in your **PyCharm IDE**.

b) From the **Terminal**:

1) **Type Dir.**

Note: You can see the **lambda_function.py** file.



c) From the **Terminal**:

1) **Type python lambda_function.py** (to run the Python script).

Note: You can see the error **'boto3' module not found**.


```
Terminal: Local x + v
PS C:\Users\Administrator\PycharmProjects\labaws\lambda>
PS C:\Users\Administrator\PycharmProjects\labaws\lambda> python lambda_function.py
Traceback (most recent call last):
  File "C:\Users\Administrator\PycharmProjects\labaws\lambda\lambda_function.py", line 4, in <module>
    import boto3
ModuleNotFoundError: No module named 'boto3'
PS C:\Users\Administrator\PycharmProjects\labaws\lambda>
```

Info: The AWS SDK for Python (Boto3) provides a Python API for AWS services. You use the AWS SDK for Python (Boto3) to create, configure, and manage AWS services.

d) From the **Terminal**:

1) **Type** `pip install boto3`.

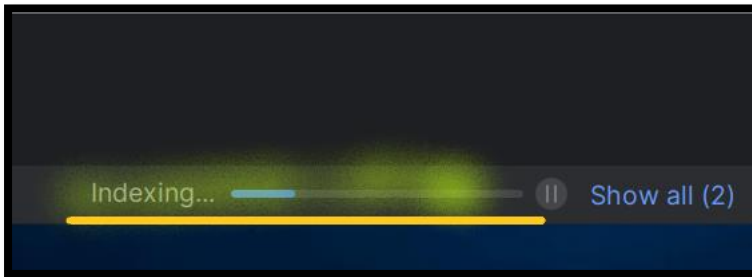
```
(venv) C:\Users\Administrator\PycharmProjects>pip install boto3
```

Note: **Wait**, till **boto3** sdk **install** successfully.

```
Terminal: Local x +
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
  | 227 kB ...
Collecting six>=1.5
  Downloading six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: six, urllib3, python-dateutil, jmespath, botocore, s3t
Successfully installed boto3-1.17.54 botocore-1.20.54 jmespath-0.10.0 python-dateutil
(venv) C:\Users\Administrator\PycharmProjects>
```

Note: **Ignore** the pip upgrade warnings.

Note: **Wait**, till **Indexing** gets **completed**.



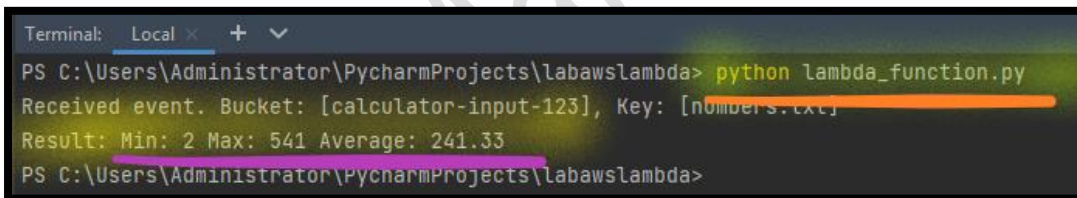
Step 3: Execute the Lambda Function

36. From the **PyCharm IDE**.

a) From the **Terminal**:

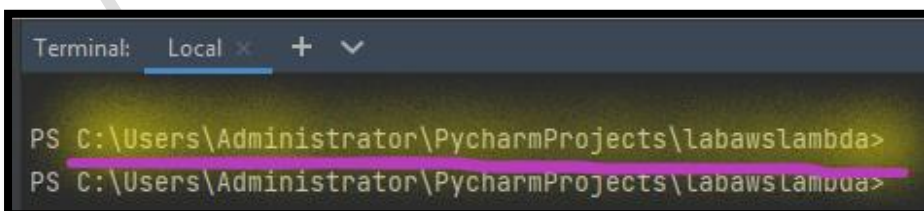
- 1) **Type** **python lambda_function.py** (to run the Python script).

Note: You can see the **logging output** from the **print statements**.



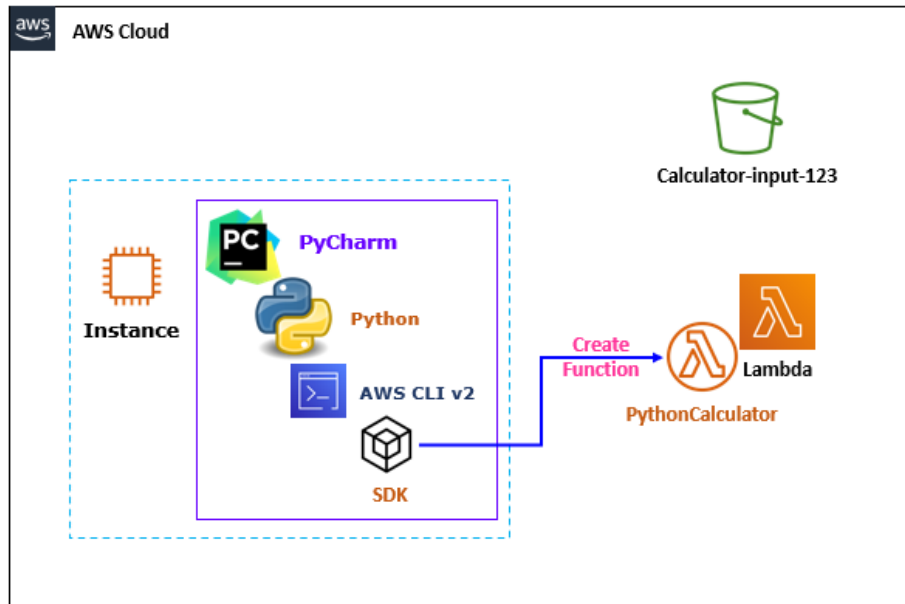
b) From the **Terminal**:

- 1) **Copy** the **Current working directory path** in **Notepad**.



Task 7: Deploy the Lambda function

Now that the code has been developed and tested on the Windows Dev Instance, you will deploy your Lambda function.



Step 1: Update the Lambda Function

37. From the **PyCharm IDE**.

- In the **lambda_function.py** file, locate the **Debugging** section and **remove all the code lines** from **Row number 32** to the **end**.

```
lambda_function.py
18     )
19
20     # get the object contents
21     file_contents = response['Body'].read().decode("utf-8").strip()
22     # find matches of all positive or negative numbers
23     numbers = [int(n) for n in re.findall(r"-?\d+", file_contents)]
24     if numbers:
25         # caculate min/max/average
26         mn, mx, avg = min(numbers), max(numbers), sum(numbers)/len(numbers)
27         result = "Min: %s Max: %s Average: %s" % (mn, mx, avg)
28
29     print("Result: %s" % result)
30     return result
31
```

b. From the **PyCharm IDE**:

i. Press **CTRL + S**.

Step 2: Compress the Python Source Code

38. From the **DevPYInstance**, right click on **Start** & **Run**.

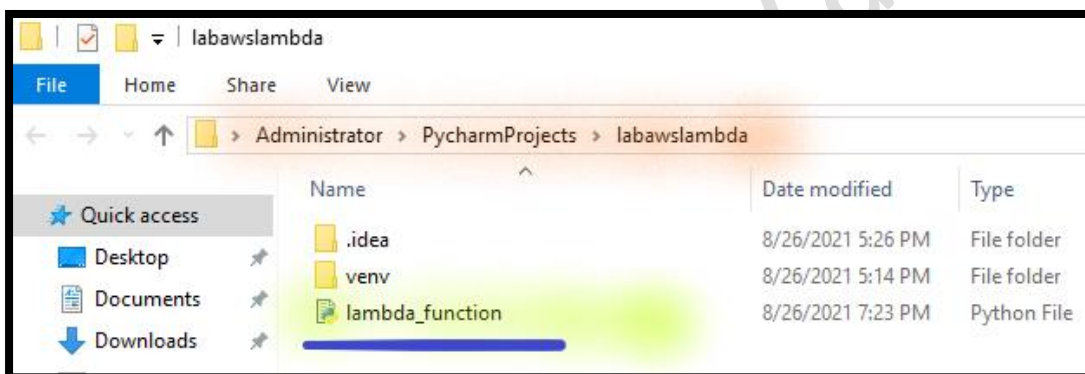
a. In the **Open**, write

C:\Users\Administrator\PycharmProjects\labawslambda,

Press **Ok**.

Note: Change the **directory path** if you have **different working directory path**.

Note: You can see the **lambda_function.py** file.

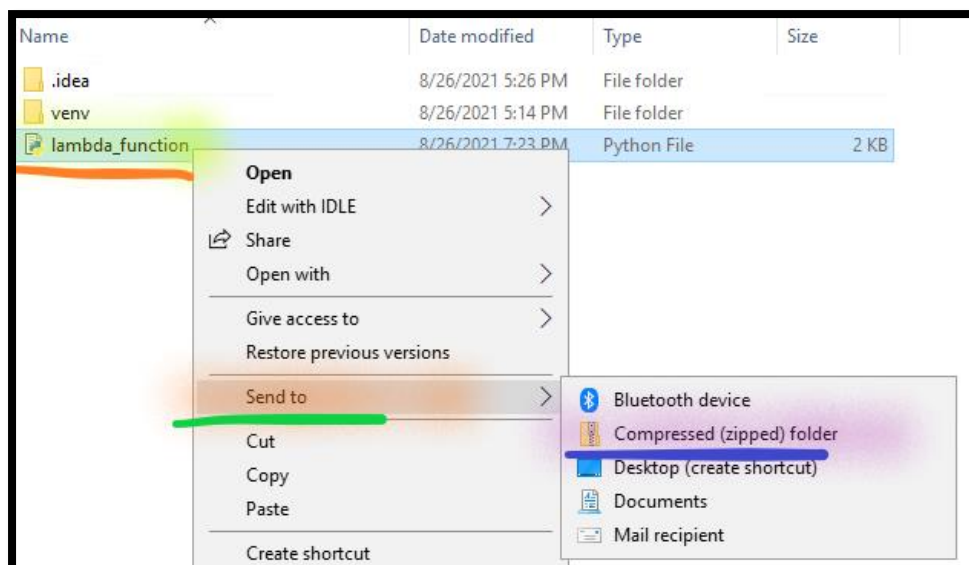


Note: Compress the **lambda_function.py** file.

i. **Right-click** on the **lambda_function.py** file.

a) Select **Send to**.

1) Select **Compressed (zipped) folder**.



Note: Ensure **lambda_function.py** should be in the **root of .Zip file**, not in the sub folder.

Step 3: Deploy the Lambda function from PyCharm

39. From the **PyCharm IDE**.

a. From the **Terminal**:

1) **Type Dir.**

Note: You can see the **lambda_function.py** and **lambda_function.zip** file.

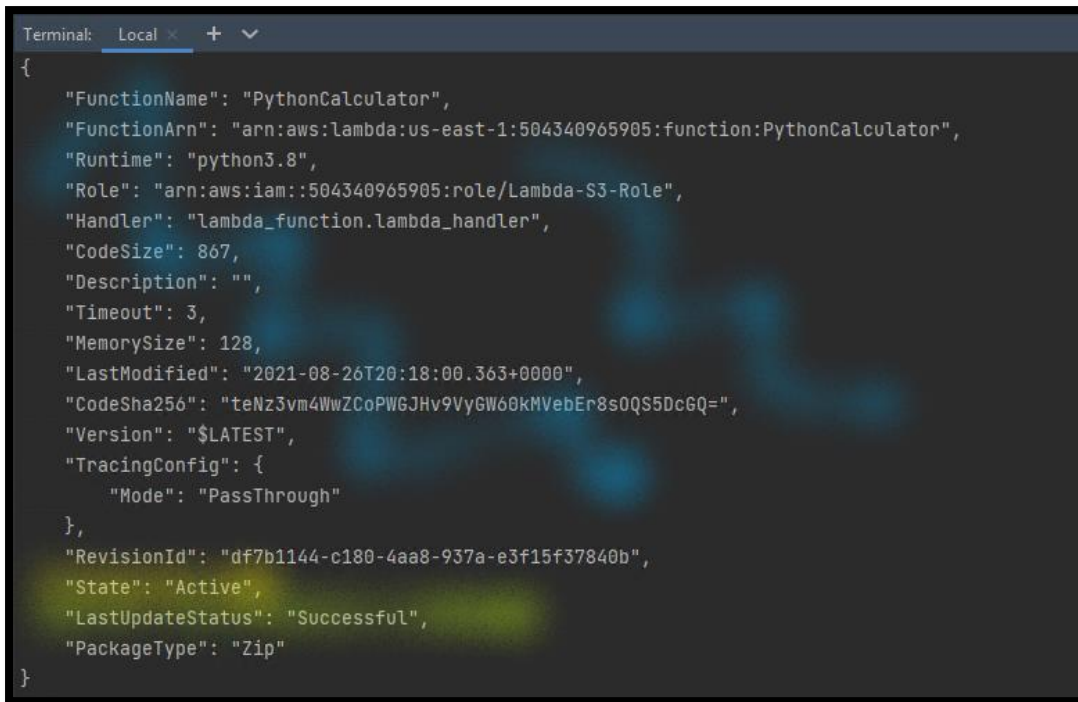
b. From the **Terminal**:

1) **Type**

aws lambda update-function-code --function-name PythonCalculator --zip-file fileb://lambda_function.zip

```
\labaws\lambda>
\labaws\lambda> aws lambda update-function-code --function-name PythonCalculator --zip-file fileb://lambda_function.zip
```

Note: Check the **Output**.



```
Terminal: Local + -
{
  "FunctionName": "PythonCalculator",
  "FunctionArn": "arn:aws:lambda:us-east-1:504340965905:function:PythonCalculator",
  "Runtime": "python3.8",
  "Role": "arn:aws:iam::504340965905:role/Lambda-S3-Role",
  "Handler": "lambda_function.lambda_handler",
  "CodeSize": 867,
  "Description": "",
  "Timeout": 3,
  "MemorySize": 128,
  "LastModified": "2021-08-26T20:18:00.363+0000",
  "CodeSha256": "teNz3vm4WwZCoPWGJHv9VyGW60KMVebEr8s0QS5DcGQ=",
  "Version": "$LATEST",
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "RevisionId": "df7b1144-c180-4aa8-937a-e3f15f37840b",
  "State": "Active",
  "LastUpdateStatus": "Successful",
  "PackageType": "Zip"
}
```

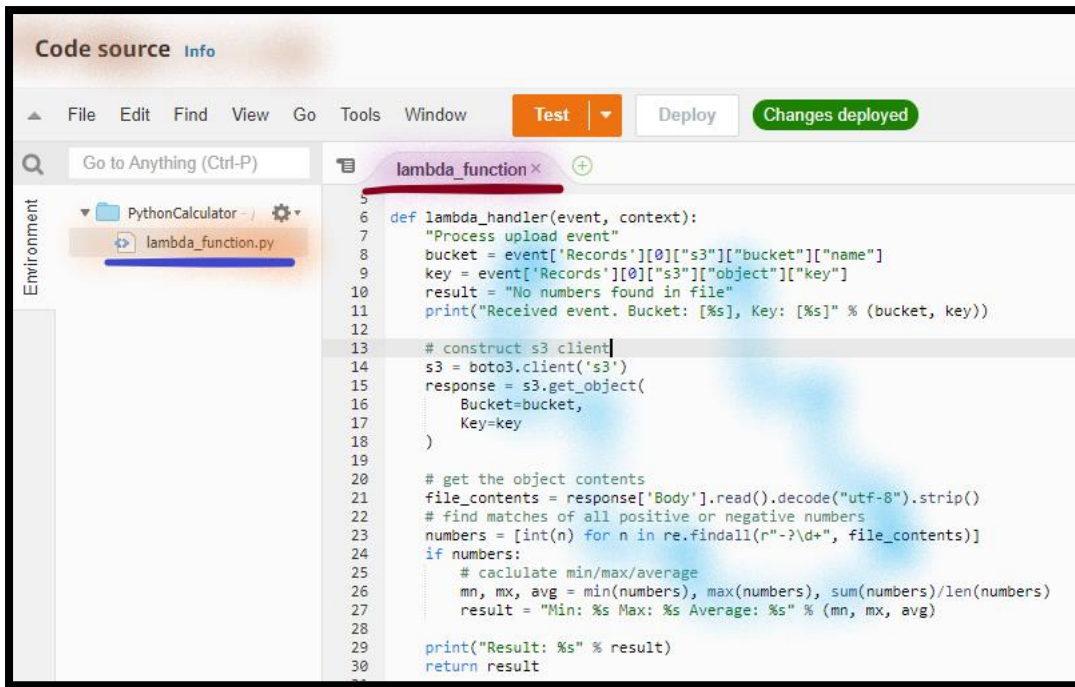
Note: Go to next Task, But Don't Close the DevPYInstance console.

Step 4: Verify the Code from the Lambda Function

40. Return to the Lambda console.
 - a. In the Code section.
 - i. Double click on `lambda_function.py`.

Note: You can see the Deployed code.

Note: If you are not seeing the Lambda Code, Refresh your Browser.

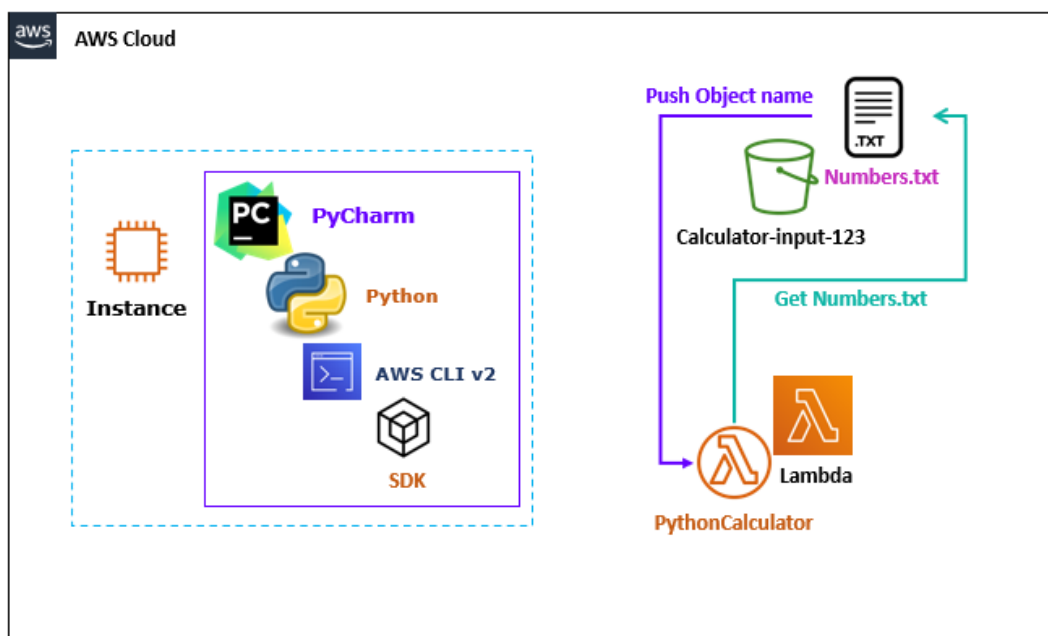


```

5
6 def lambda_handler(event, context):
7     "Process upload event"
8     bucket = event['Records'][0]['s3']['bucket']['name']
9     key = event['Records'][0]['s3']['object']['key']
10    result = "No numbers found in file"
11    print("Received event. Bucket: [%s], Key: [%s]" % (bucket, key))
12
13    # construct s3 client
14    s3 = boto3.client('s3')
15    response = s3.get_object(
16        Bucket=bucket,
17        Key=key
18    )
19
20    # get the object contents
21    file_contents = response['Body'].read().decode("utf-8").strip()
22    # find matches of all positive or negative numbers
23    numbers = [int(n) for n in re.findall(r"-?\d+", file_contents)]
24    if numbers:
25        # calculate min/max/average
26        mn, mx, avg = min(numbers), max(numbers), sum(numbers)/len(numbers)
27        result = "Min: %s Max: %s Average: %s" % (mn, mx, avg)
28
29    print("Result: %s" % result)
30    return result
  
```

Task 8: Invoke the Lambda Function

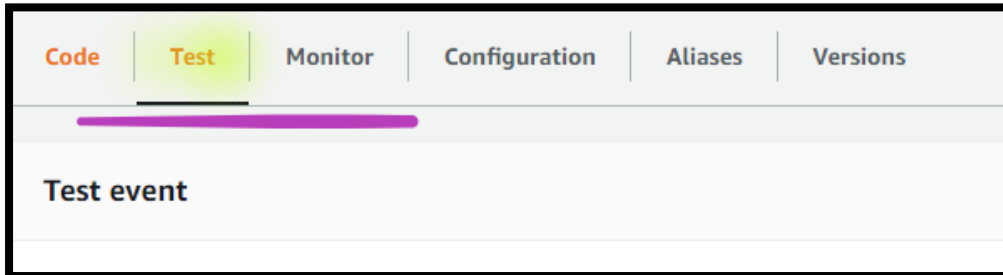
Now that the code has been deployed, invoke the lambda function.



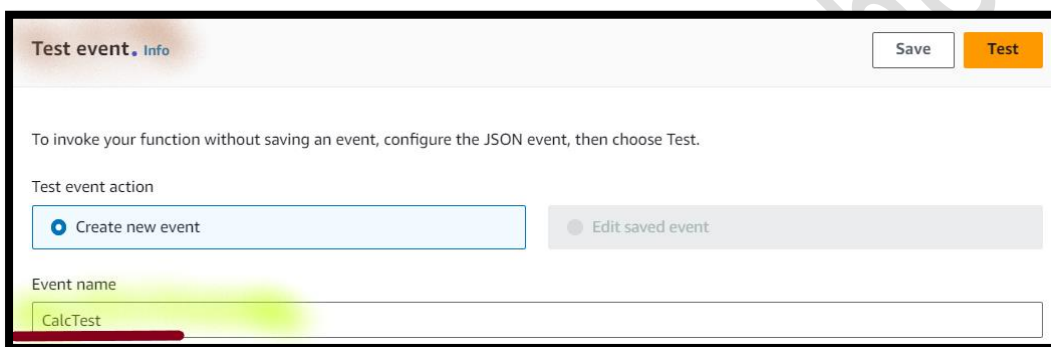
Step 1: Update the Test Events

41. From the **Lambda** console.

a. In the **Test** section.



i. **Event Name:** Write **CalcTest**.



ii. In the **Event JSON:**

a) **Remove** the **existing event**.

b) **Copy** the **below event**:

```
{
  "Records": [
    {
      "s3": {
        "object": {
          "key": "numbers.txt"
        },
        "bucket": {
          "name": "REPLACE WITH BUCKET NAME"
        }
      }
    }
  ]
}
```

1) **Update** the **REPLACE WITH BUCKET NAME** with the **Bucket name**, *which you have created* in the previous step.

Note: Don't remove the starting and end double quote.

Note: Object name **numbers.txt** is already mentioned in the **Test** event.



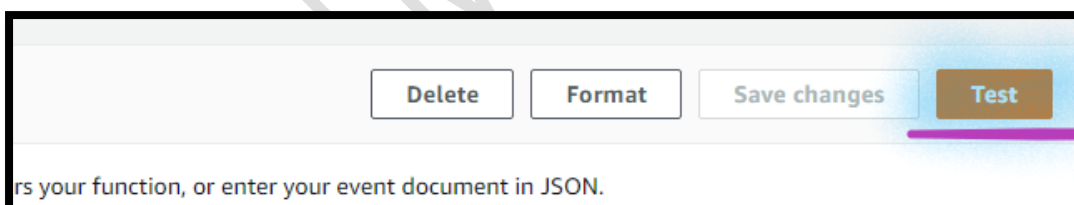
```
1 {  
2   "Records": [  
3     {  
4       "s3": {  
5         "object": {  
6           "key": "numbers.txt"  
7         },  
8         "bucket": {  
9           "name": "calculator-input-123"  
10        }  
11      }  
12    ]  
13  }  
14 }
```

iii. Select **Save**.

Step 2: Validate Your Implementation

42. **From** the **Test** section:

a. Select the **Test**.



Note: Once you **invoked** the **function** and code executed successfully you can see the **Execution result** as **Succeeded**.

i. **Expand** the **Details** section of the **execution result** section.

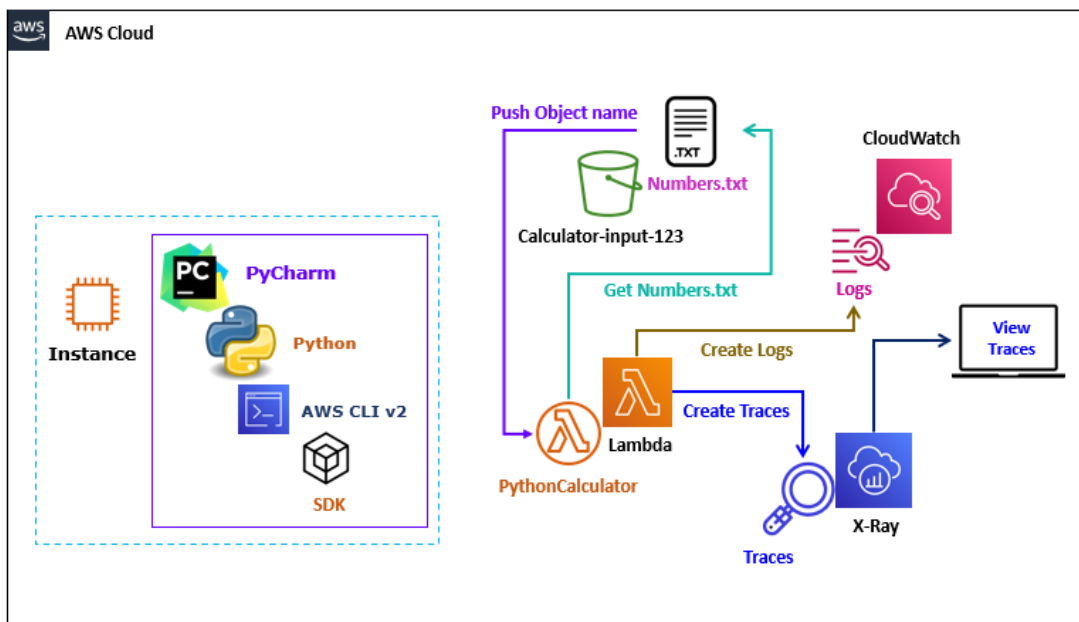
Note: You can view the **Min**, **Max** and **Average** count.



Note: Execute the Test multiple times.

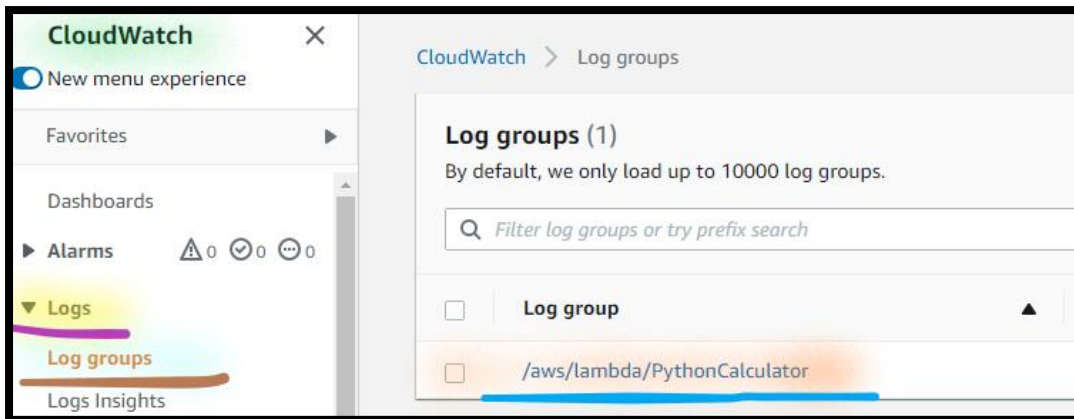
Task 9: Monitor the Lambda Execution

In this task, you will monitor the Events and Traces generated by Lambda.



Step 1: Monitor the CloudWatch Events

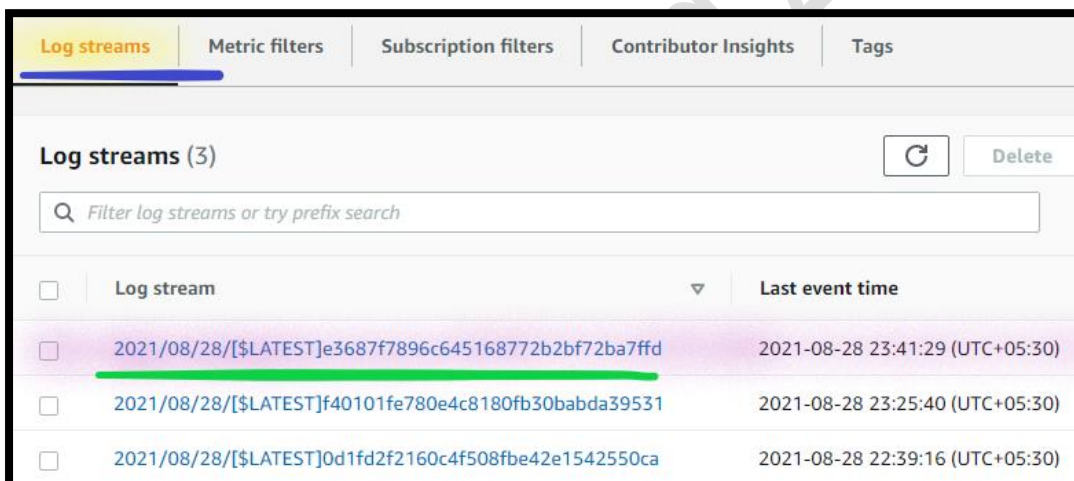
43. In the **AWS Management Console**, on the **Services** menu, search and select **CloudWatch**.
44. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
45. **Expand** the **Logs**.
 - a. Click on the **Log groups**.
 - i. Open the **/aws/lambda/PythonCalculator** log group.



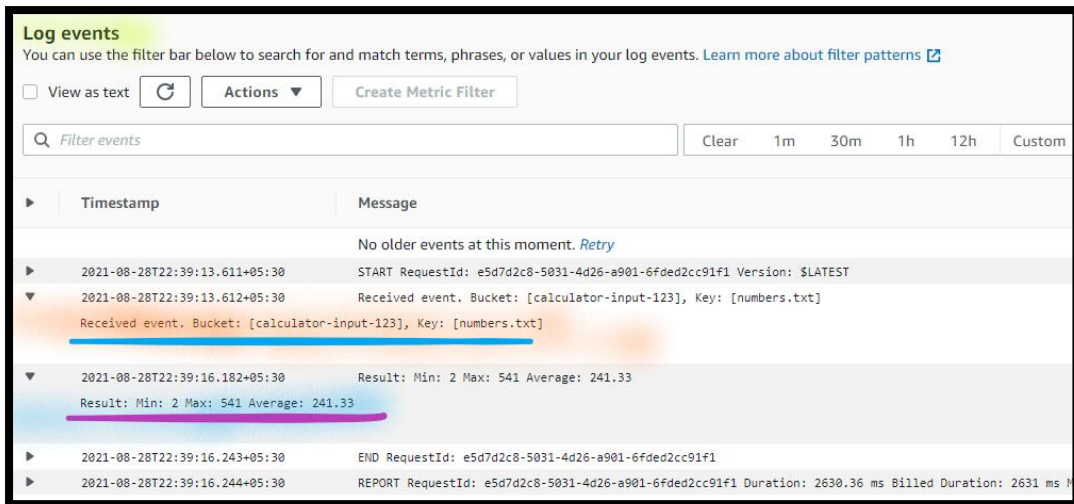
Note: You can see the **Log streams** generated by Lambda with the **Date** and **Time**.

a) Select the **Log streams**.

1) Open the **latest Log streams**.



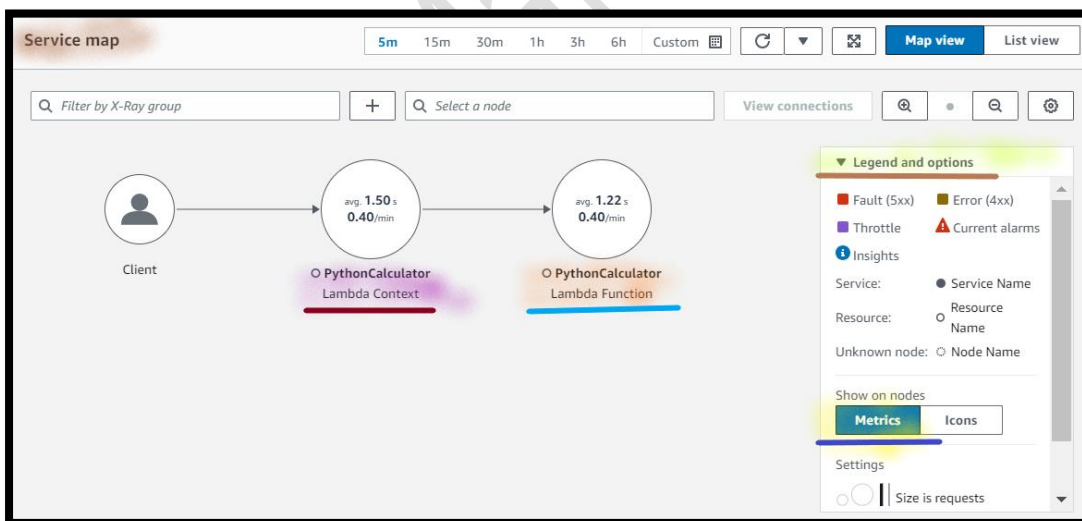
Note: **Inspect the contents** of all the recent **entries** of all the log streams. Confirm you see the **output** from your **test case** files.



Step 2: Monitor the X-Ray Traces

46. From the **CloudWatch** console.

- a. Select the **X-Ray traces**.
 - i. Select the **Trace map**.
 - a) **Expand Legend and options**.
 - 1) Select **Metrics**.



Info:


1. Lambda context: When Lambda runs your function, it **passes a context object to the handler**. This object provides methods and properties that provide information about the invocation, function, and execution environment.

2. Lambda function: The Lambda function handler is the method in your **function code that processes events**. When your function is invoked, Lambda runs the handler method. Your function runs until the handler returns a response, exits, or times out.

47. **From** the **CloudWatch** console.

- a. Select the **X-Ray traces**.
 - i. Select the **Traces**.

Note: You can view the **Trace IDs**.

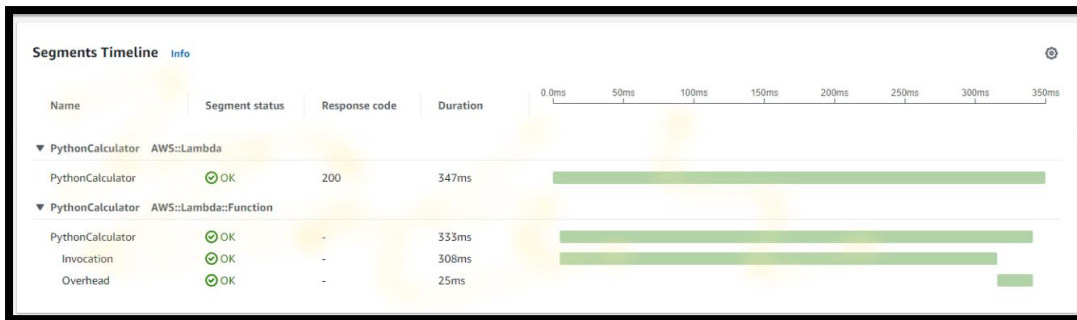


The screenshot shows the 'Traces (3)' section in the AWS CloudWatch console. It includes a search bar and a table with the following data:

ID	Trace status	Timestamp	Response code	Response Time	Duration
...54d5d3054c948b2765cddff3	OK	1.1min (2023-09-04 13:46:16)	200	0.347s	0.347s
...57a08f9f283271546d803623	OK	2.9min (2023-09-04 13:44:25)	200	0.327s	0.327s
...120429815b7711bc6cd21565	OK	3.0min (2023-09-04 13:44:18)	200	2.465s	2.465s

- i. Open **Any Trace ID**.

Note: You can see **Timelines**.



Info: Overhead – commonly called a cold-start, consists of two components.

1. The **first** is the time taken to set up the execution environment for your function's code, which is entirely controlled by AWS.
2. The **second** is the code initialization duration, which is managed by the developer. This is impacted by code outside of the main Lambda handler function and is often responsible for tasks like setting up database connections, initializing objects, downloading reference data, or loading heavy frameworks.

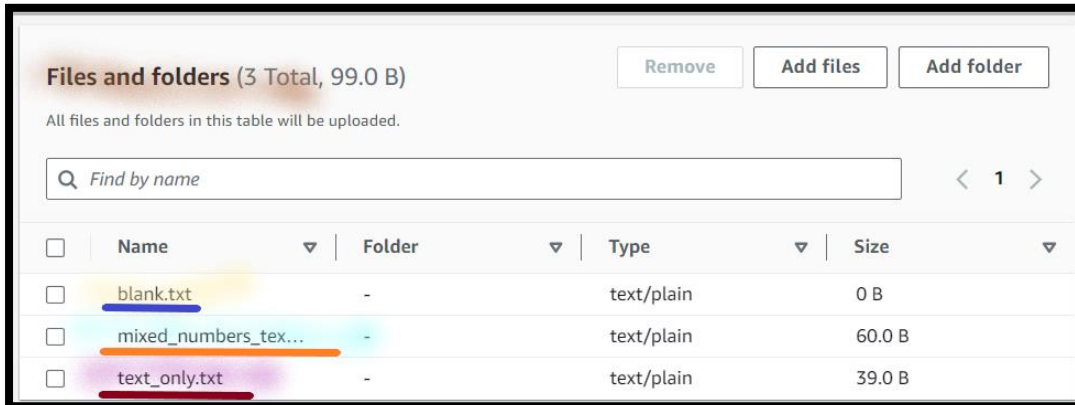
Task 10: Upload Additional Objects

Step 1: Upload Content in the Bucket

48. In the **AWS Management Console**, on the **Services** menu, search and select **S3**.
49. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
50. Click the **Buckets**.
51. Open **calculator-input-123** bucket.
 - a. Select **Objects**.
 - i. Click **Upload**.
 - a) Click **Add files**.
 - 1) **Navigate** and select **blank.txt**, **mixed_numbers_text.txt** and **text_only.txt** file.

I. Select **Open**.

Note: **blank.txt**, **mixed_numbers_text.txt** and **text_only.txt** file is provided with the **Lab manual**.



ii. Select **Upload**.

Note: Once uploaded you can see the **blank.txt**, **mixed_numbers_text.txt** and **text_only.txt** file under **files and folders** section.

Step 2: Monitor the CloudWatch Events

52. In the **AWS Management Console**, on the **Services** menu, search and select **CloudWatch**.

53. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.

54. Expand the **Logs**.

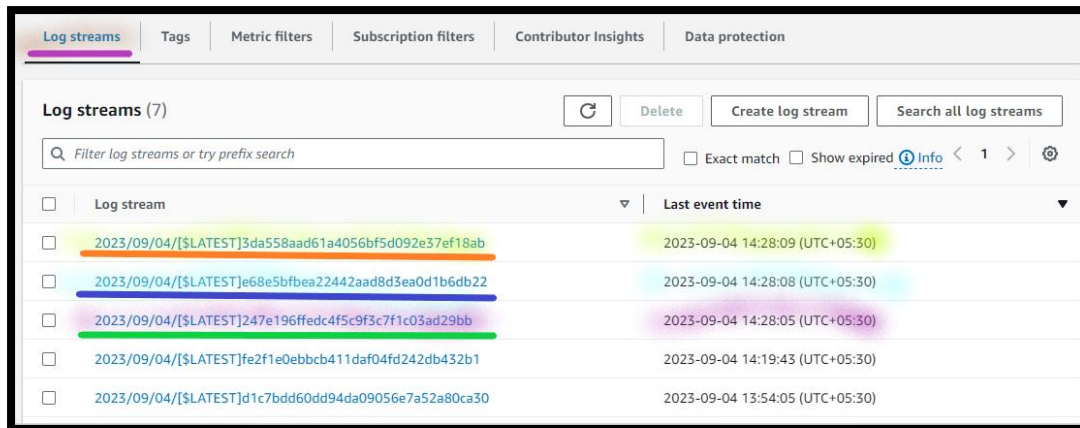
a. Select the **Log groups**.

i. Open the **/aws/lambda/PythonCalculator** log group.

a) Select the **Log streams**.

1) Open the **last three latest Log streams (One by one)**.

Note: **Inspect the contents** of all the recent **entries** of all the log streams. Confirm you see the **Output** from your **test case** files.



Task 11: Delete the Environment

Step 1: Close the Project

55. From the **PyCharm IDE**.

- a. Select the **Menu**.
 - i. Select **File**.
 - a) Select **Close project**.

Step 2: Delete the Bucket

56. In the **AWS Management Console**, on the **Services** menu, search and select **S3**.

57. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.

58. Select the **Buckets**.

- a. Select **calculator-input-123** bucket.
 - i. Select **Empty**.
 - a) Type **permanently delete** to delete all the objects.
 - b) Select **Empty**.
 - c) Select **Exit**.
- b. Select **calculator-input-123** bucket.
 - i. Select **Delete**.
 - a) Type **calculator-input-123** bucket name to delete bucket.
 - b) Select **Delete bucket**.

Step 3: Delete the Lambda Function

59. In the **AWS Management Console**, on the **Services** menu, search and select **Lambda**.
60. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
61. Select the **Functions**.
 - a. Select the **PythonCalculator**.
 - i. Select **Actions**.
 - a) Select **Delete**.
 - 1) When you **get prompt**, write **delete**.
 - I. Select **Delete**.
 - A. Select **Close**.

Step 4: Delete the Log Groups

62. In the **AWS Management Console**, on the **Services** menu, click **CloudWatch**.
63. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
64. Expand the **Logs**.
 - a. Select the **Log groups**.
 - i. Select **/aws/lambda/PythonCalculator**.
 - a) Select **Actions**.
 - 1) Select **Delete log groups(s)**.
 - I. Select **Delete**.