# Developing Event-Driven Solutions with AWS Lambda (LAB-M06-01)

## 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.

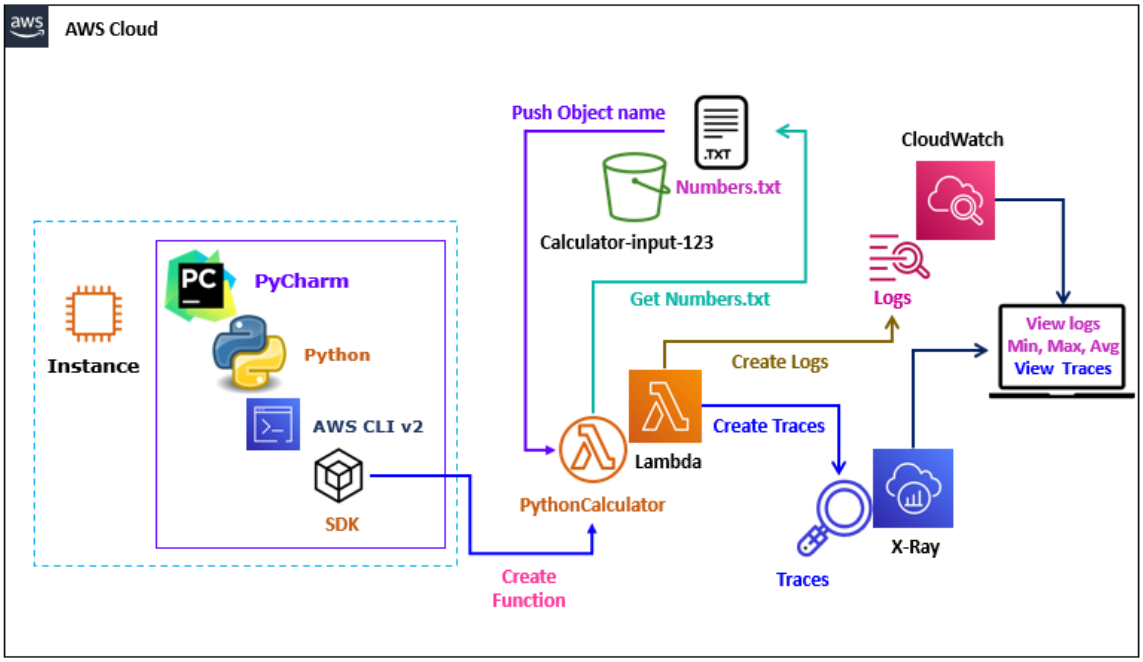
The Lambda function will calculate the minimum, maximum, and average of the numbers contained in an object uploaded to the Amazon S3 bucket.

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 new Bucket.
- Upload Object in the bucket.
- Get Object from the bucket.
- Process Object using Lambda function.
- View the Lambda Logs in CloudWatch.
- View the Lambda Traces in X-Ray.

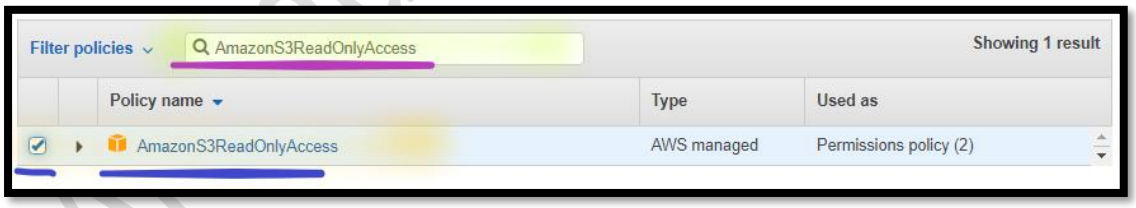


# Task 1: Create IAM Role

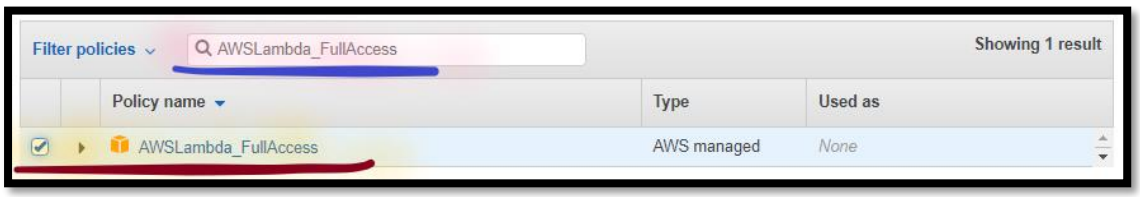
In this task, you will create AWS IAM Role with Permission to manage S3 and Lambda.

## Step 1: Create IAM User

1. In the **AWS Management Console**, on the **Services** menu, click **IAM**.
2. Select **Users**.
    a. Select **Add users**.
        i. In the In the **Set user details** section:
            a) **Username**: Write **Lambda-S3-User**.
        ii. In the In the **Select AWS access type** section:
            a) **Select AWS credential type**: Select **Access key - Programmatic access**.
            b) Select **Next: Permissions**.
        ii. In the **Set permissions** section:
            a) Select **Attach existing policies directly**.
                1) In the **Search box**, write **AmazonS3ReadOnlyAccess** and select **Enter Key**.
                I. Select the **AmazonS3ReadOnlyAccess**.




                2) In the **Search box**, write **AWSLambda_FullAccess** and select **Enter Key**.
                I. Select the **AWSLambda_FullAccess**.



3) Select **Next: Tags**.

iii. In the **Add tags** section:

**Note**: Leave the other details as default.

a) Select **Next: Review**.

iv. In the **Review** section.

**Note**: In the **Permission summary**, you can see the **AmazonS3ReadOnlyAccess** and **AWSLambda_FullAccess** policies.


Permissions summary

The following policies will be attached to the user shown above.

| Type | Name |
|---|---|
| Managed policy | AmazonS3ReadOnlyAccess |
| Managed policy | AWSLambda_FullAccess |

a) Select **Create users**.

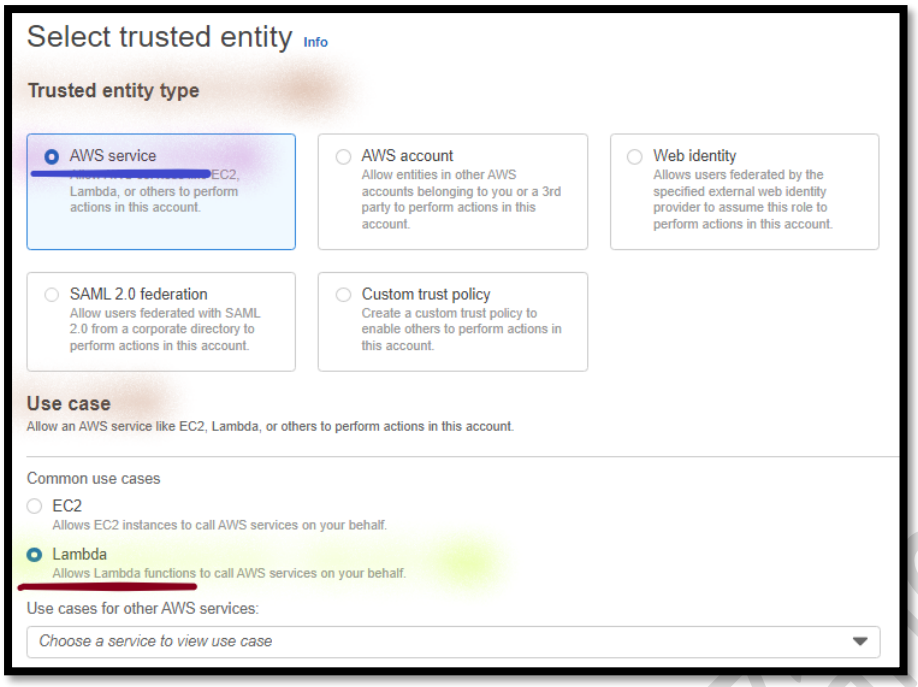
**Note**: **Wait**, till you can see the **message** "**You successfully created the users**".

**Note**: Select **Download .csv**, to **download** the **Access key ID** and **Secret access key** details in your local desktop/ laptop.

b) Select **Close**.

## Step 2: Create IAM Role

3. **From** the **IAM** console.
4. Select **Roles**.
   a. Click on **Create role**.
      i. In the **Select trusted entity** section.
         a) **Trusted entity type**: Select **AWS service**.
         b) **Common use cases**: Select **Lambda**.

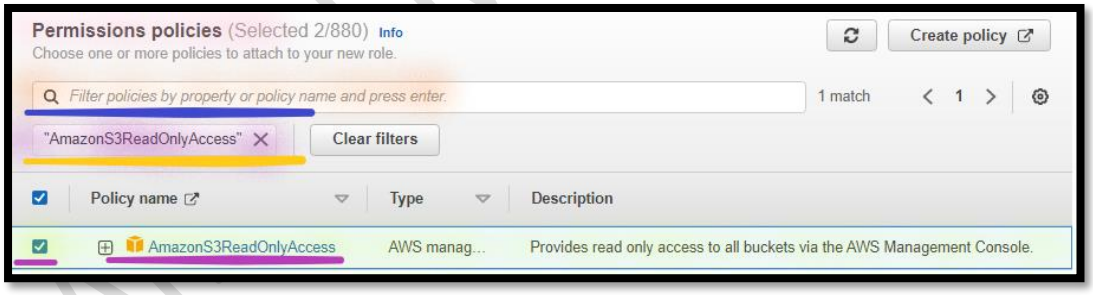



c) Select **Next**.

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

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

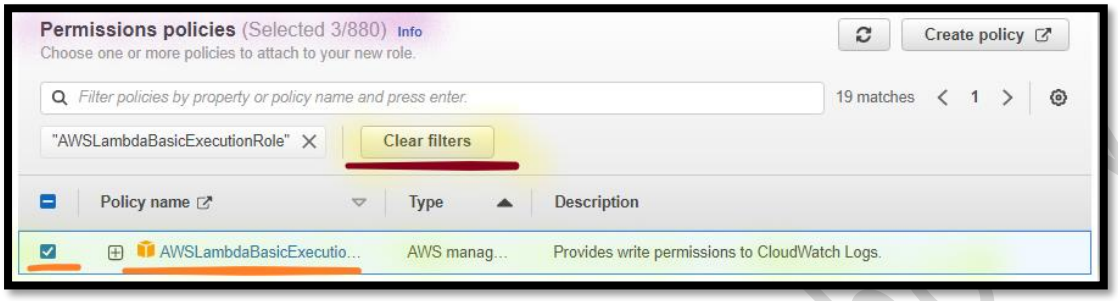
1) Select the **AmazonS3ReadOnlyAccess**.




2) Select **Clear filters**.

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

1) Select the **AWSLambdaBasicExecutionRole**.

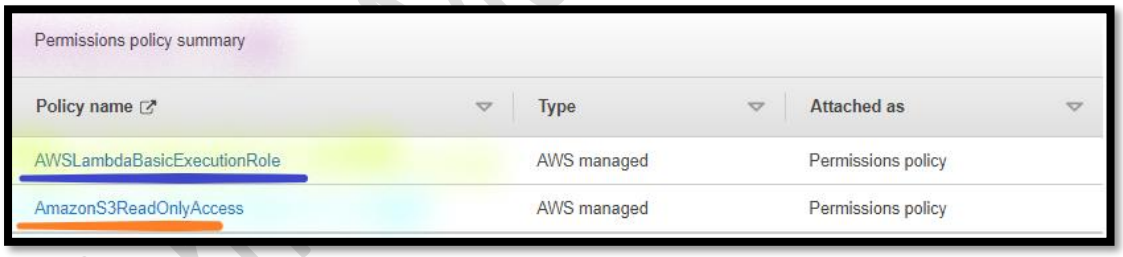



c) Select **Next**.

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

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

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

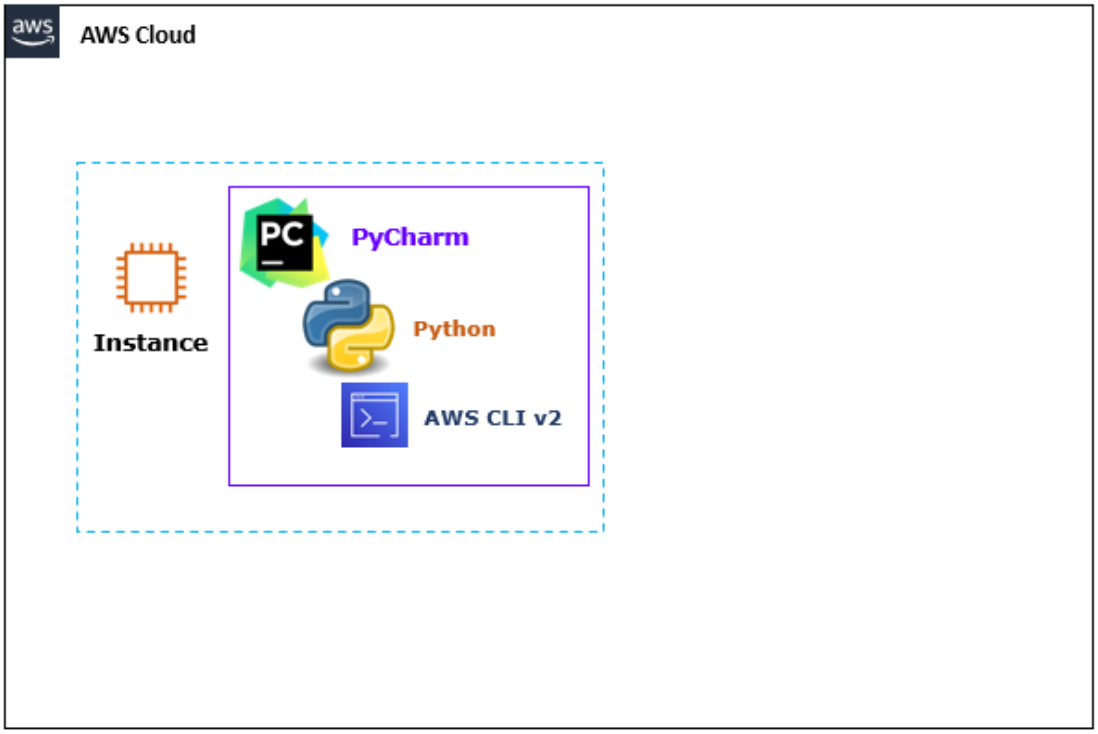



b) Click **Create role**.

**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 AWS Virtual machine to build development environment and install the Python, PyCharm and AWS CLI.




## Step 1: Create EC2 Instance

5. In the **AWS Management Console**, on the **Services** menu, search and select **EC2**.
6. Choose the **US East (N. Virginia)** region list to the right of your account information on the navigation bar.
7. Select **Instances**.
8. Select **Launch Instances**.
    a. In the **Name and tags** section:
        i. **Name**: Write **Dev-Python-Instance**.
    b. In the In the **Application and OS Images** section:
        i. In the **Search box**:
            a) Type **Microsoft Windows Server 2019 Base**.
            b) Press **Enter** ***key***.

**Note**: You can see the **Choose an Amazon Machine Image** page.

c) **From** the **Choose an Amazon Machine Image** page:

1) Select **Microsoft Windows Server 2019 Base**.

**Note**: You can see the **Launch an Instance** page.

c. In the **Instance Type** section:

i. **Instance type**: Dropdown and in the **Search box**:

a) Type **t2.medium**.

b) Select **t2.medium**.

**Note**: You can also use **t2.micro**, but may cause the **poor performance** because of low processor and memory.

d. In the **Key pair (login)** section:

i. **Key pair name**: Dropdown and select **My-Dev-LAB-KP**.

e. In the **Network settings** section:

i. Click on **Select Create security group**.

a) Click on **Select Allow RDP traffic from**.

1) Dropdown and select **Anywhere**.

**Note**: Leave the other details as default.

f. In the **Summary** section:

i. Select **Launch Instances**.

**Note**: **Wait**, till you can see the **message** "**Successfully initiated launch of instance**".

g. Select **View all instances**

**Note**: **Wait**, till you can see the **Dev-Python-Instance** Instance **State** is **Running**.

**Note**: **Wait**, till you can see the **Dev-Python-Instance**Instance **Status check** is **2/2 check passed**.

### Step 2: Copy the IP Address of Instance

9. **From** the **EC2** console.
10. Select the **Dev-Python-Instance**.
    a. Select the **Details**.

**Note**: **Copy** the **Public IP address** of **Dev-Python-Instance** in the **Notepad**.

### Step 3: Generate the Password of Instance

11. **From** the **Dev-Python-Instance** console.
    a. Select **Actions**.
        i. Select **Security**.
        ii. Select **Get Windows Password**.
            a) From the **Get Windows Password** console:
                1) **Browse**: **Click**, **Navigate** and **select** the **My-Dev-LAB-KP**.pem key pair (which you have downloaded in the previous step).
                2) Click on **Decrypt Password**.

**Note**: **Copy** the **Dev-Python-Instance Password** in the **Notepad**.

                3) Select **Ok**.

## Step 4: Connect to Instance

12. From the **Local Desktop/ Laptop** (Windows server 2019), right click on **Start** & **Run**.

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

    b. Select **Ok**.

        i. **From** the **Remote Desktop Connection**:

            i. **Computer**: Write the **Public IP Address** of the **Dev-Python-Instance**.

            ii. Select **Connect**.

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

1) **Username:** Write **Administrator**.

2) **Password**: Write the **Password** (*which you have copied in the previous step*).

3) Select **Ok**.

## Step 5: Update the Security Settings

13. From the **Dev-Python-Instance** (Windows server 2019), right click on **Start** & **Run**.

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

    b. Select **Ok**.

        i. **From** the **Server Manager**:

            a) Select the **Local Server**.

                1) **IE Enhanced Security Configuration**: Select **On**.

**Note**: You can see the **Internet Explorer Enhanced Security Configuration** page.

I. **Administrators**: Select **Off**.

II. Select **Ok**.

ii. Select **Cross** to **close** the Server manager.

## Step 6: Install the Python

14. From the **Dev-Python-Instance** (Windows server 2019).

    a. **Download** and **Install** the ***Python 3.8*** for **Windows x64**.

**Note**: Use the below URL to download the **Python 3.8**.

https://www.python.org/ftp/python/3.8.10/python-3.8.10-amd64.exe

**Note**: **Wait**, till **Python 3.8** install **succesfully**.

## Step 7: Check the Java Development Kit version

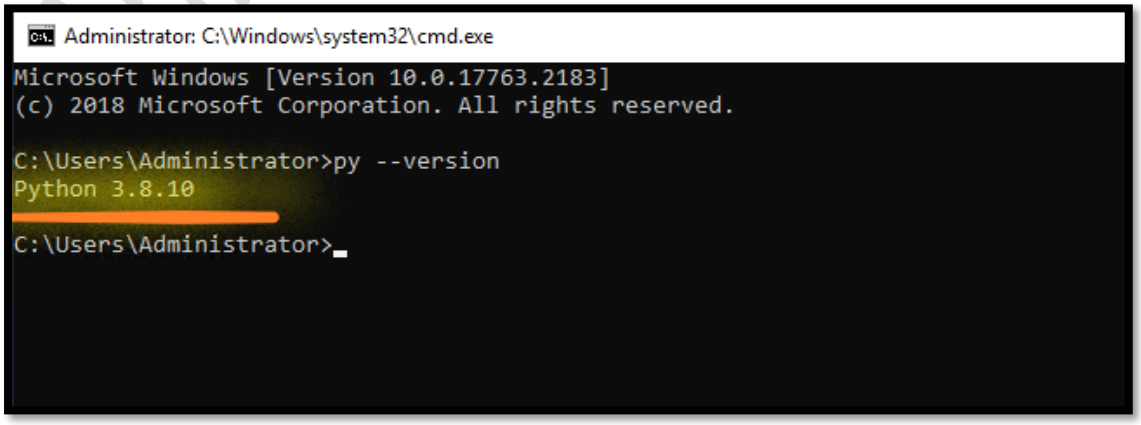
15. From the **Dev-Java-Instance** (Windows server 2019), right click on **Start** & **Run**.

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

    b. Select **Ok**.

        i. From the **command line interpreter**, write **py --version**, press **Enter** key.

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



## Step 8: Install the PyCharm IDE

16. From the **Dev-Python-Instance** (Windows server 2019).

    a. **Download** and **Install** the ***PyCharm IDE***.

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

https://www.jetbrains.com/pycharm/

i. In the **PyCharm**, select **Windows** and **Community edition**.


**Note**: **Wait**, till ***PyChram IDE*** install **succesfully**.

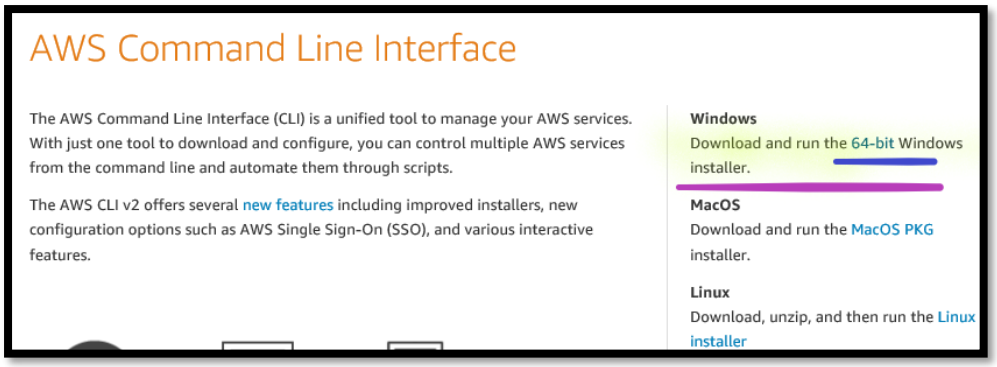
## Step 9: Install the AWS CLI v2

17. From the **Dev-Java-Instance** (Windows server 2019).

    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 **succesfully**.

## Step 10: Check the AWS CLI version

18. From the **Dev-Java-Instance** (Windows server 2019), right click on **Start** & **Run**.

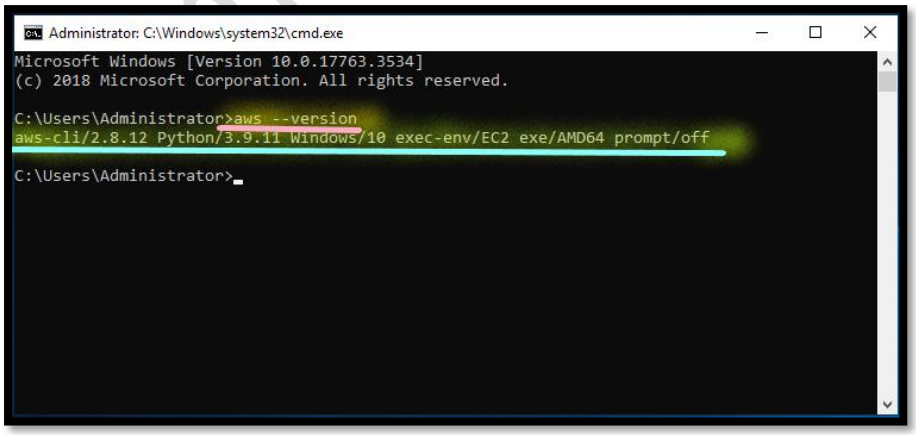
    a. In the **Open**, write **cmd**.

    b. Select **Ok**.

        i. From the **command line interpreter**, write **aws --version**, press **Enter** key.

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

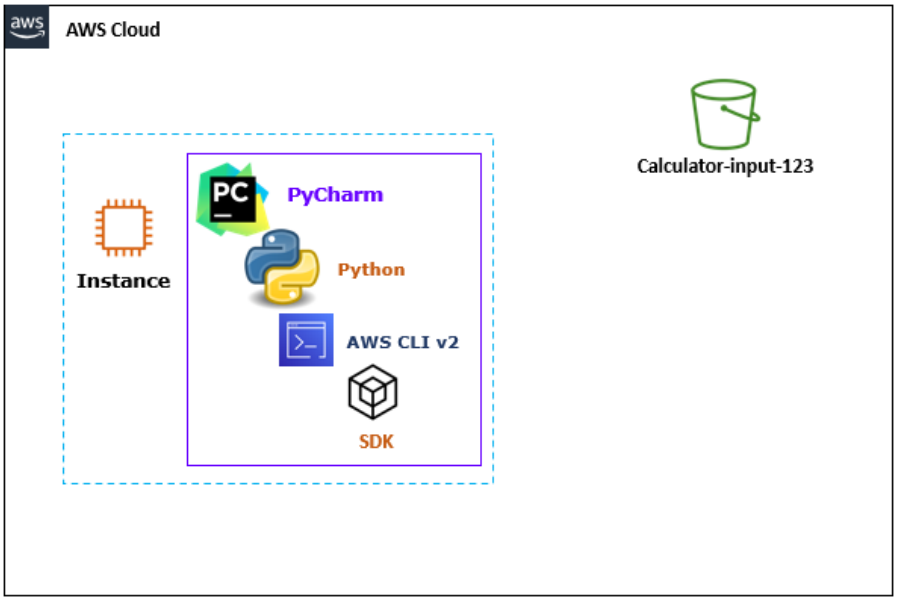
**Note:** If you can't see the version, **restart the virtual machine**.




**Note**: Go to the next task. But **Don't close** the **Dev Python Instance**.

# 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

19. In the AWS Management Console, on the **Services** menu, click **S3**.
20. Click **Create bucket**.
    a. **Bucket name**: Write **calculator-input-123**.

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

    b. **Region**: Dropdown and Select **US East (N. Virginia)**.

**Note**: Leave other details as default.

    c. Click **Create bucket**.

## Step 2: Upload Content in the Bucket

21. In the **AWS Management Console**, on the **Services** menu, click **S3**.
22. Click the **Buckets** tab.
23. Open **calculator-input-123** bucket.
    a. Select **Objects**.

b. Click **Upload**.

c. Click **Add files**.

d. Navigate and select **numbers**.txt file.

   i. Select **Open**.

**Note**: **numbers.txt** file is provided with the LAB Manual.

e. Select **Upload**.

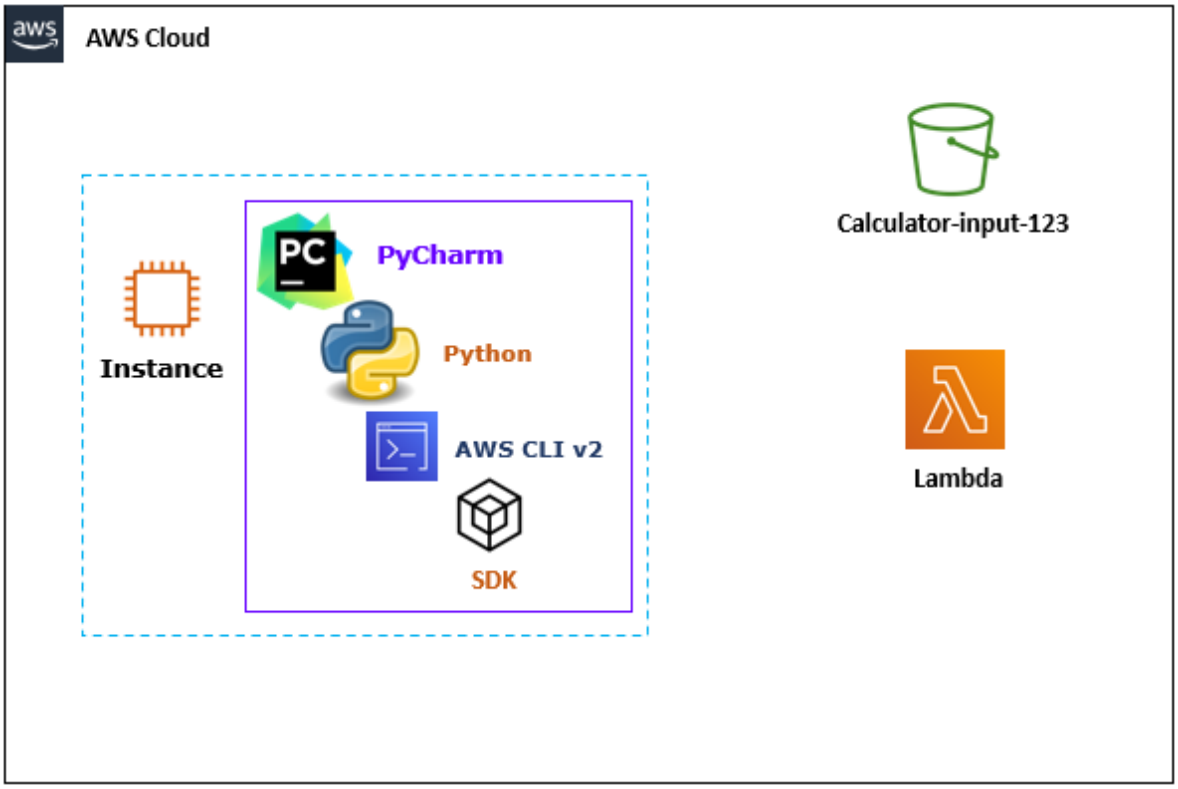
**Note**: Once uploaded you can see the **numbers.txt** file under **files and folders** section.

f. Select **Close**.

# Task 4: Create Lambda Function

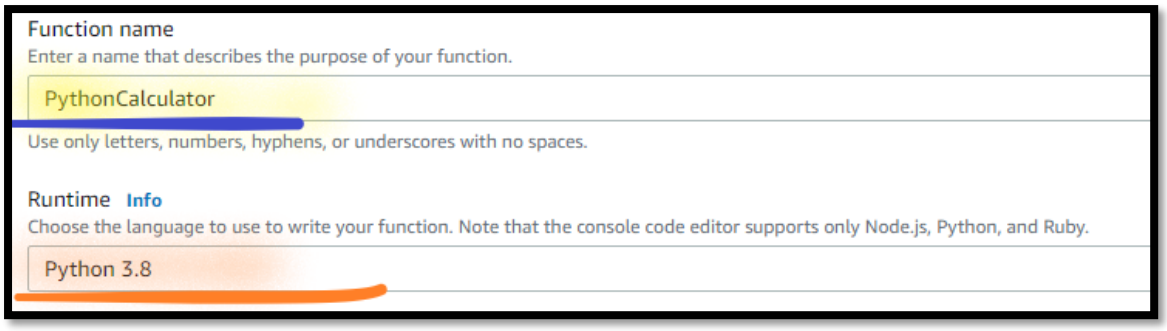
In this task, you will create a Lambda function using the AWS Management Console.

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




## Step 1: Create Lambda Function

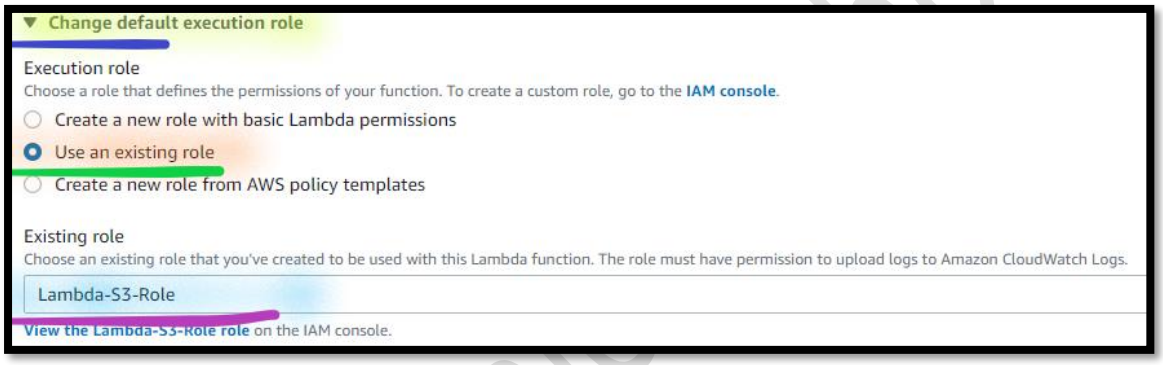
24. In the **AWS Management Console**, on the **Services** menu, click **Lambda**.
25. Click **Create a function**.
26. Select **Author from scratch** and configure:
    a. **Name**: Write **PythonCalculator**.
    b. **Runtime**: Dropdown and Select **Python 3.8**.




c. **Expand Change default execution role**.

d. **Role**: Select **Use an existing role**.

i. **Existing role**: Dropdown and Select **Lambda-S3-Role**.




e. 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. ***Go below*** in the Console, in the **Runtime settings** section.

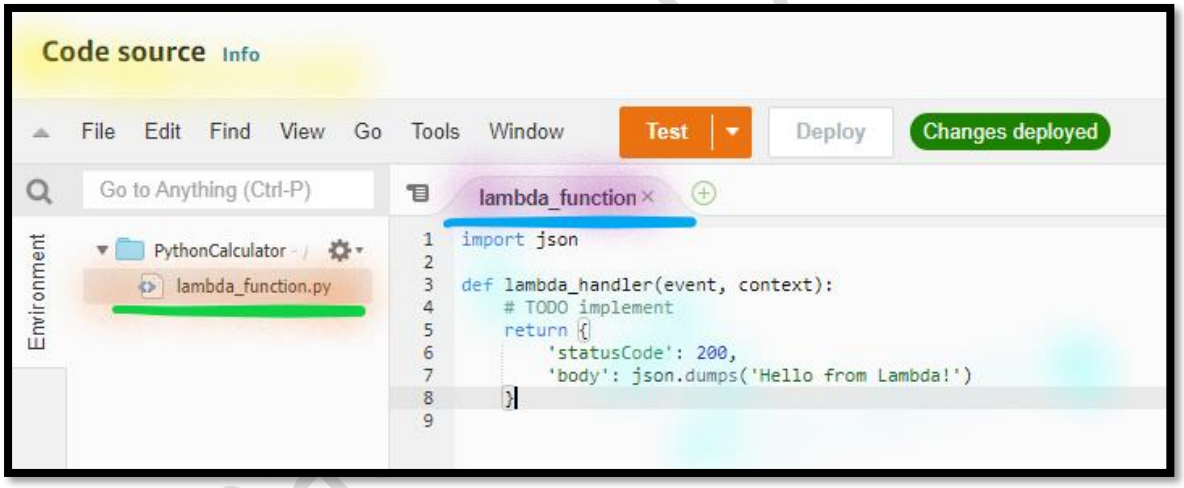
**Note**: You can see the **Handler** name as **lambda_function.lambda_handler**. If name is different, update the name to **lambda_function.lambda_handler**.

**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**).

b. ***Go above*** in the Console, in the **Code source** section.

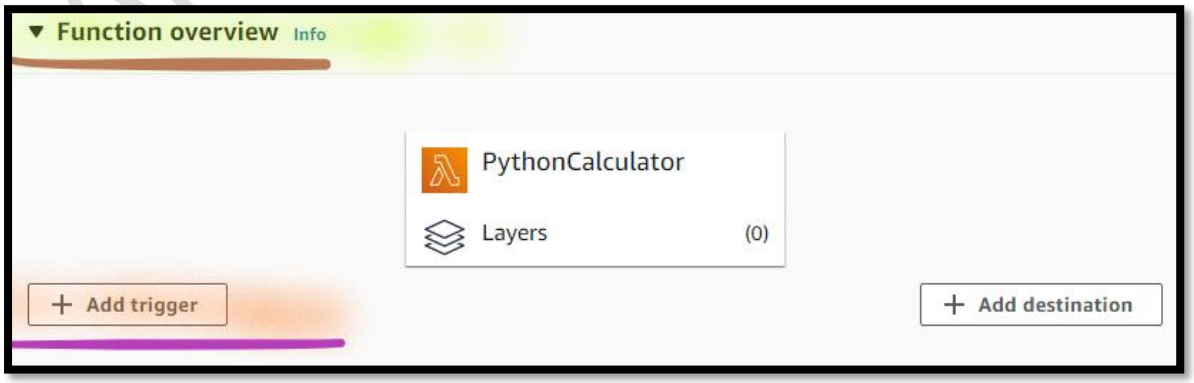
i. Double click on **lambda_function.py**.

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




c. ***Go above*** in the Console, in the **Function overview** section.

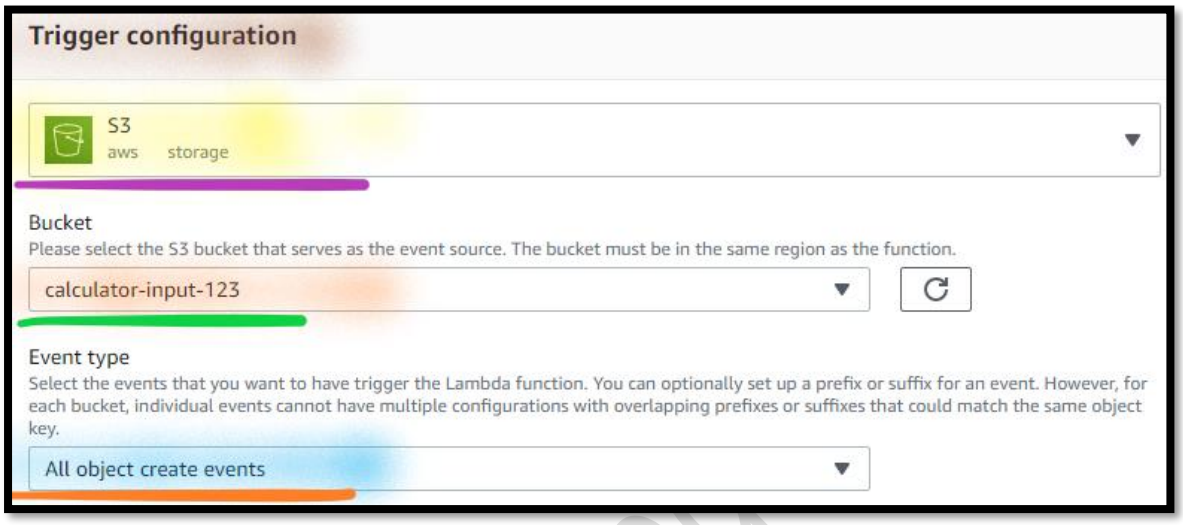
i. Select **+ Add trigger**.



1) ***Under Trigger configuration***, Dropdown and select **S3**.

2) **Bucket**: Dropdown and select **calculator-input-123**.

3) **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 Amazon S3 bucket.

4) **Recursive invocation**: **Enable** *I acknowledge that using the same Amazon S3 bucket …...*

5) Select **Add**.

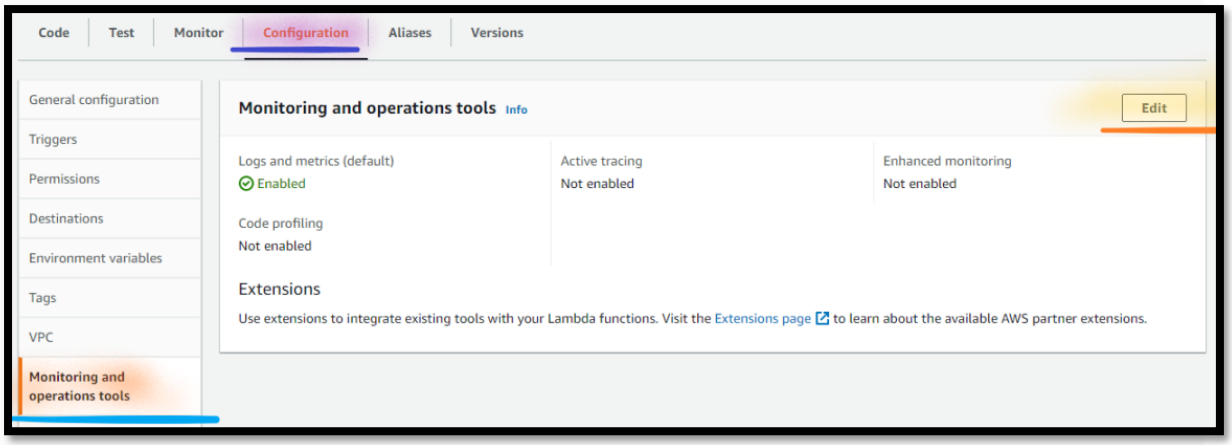
**Note**: At the top of the window, you should see a message stating the trigger was added successfully and that the function is now receiving events from the trigger.

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

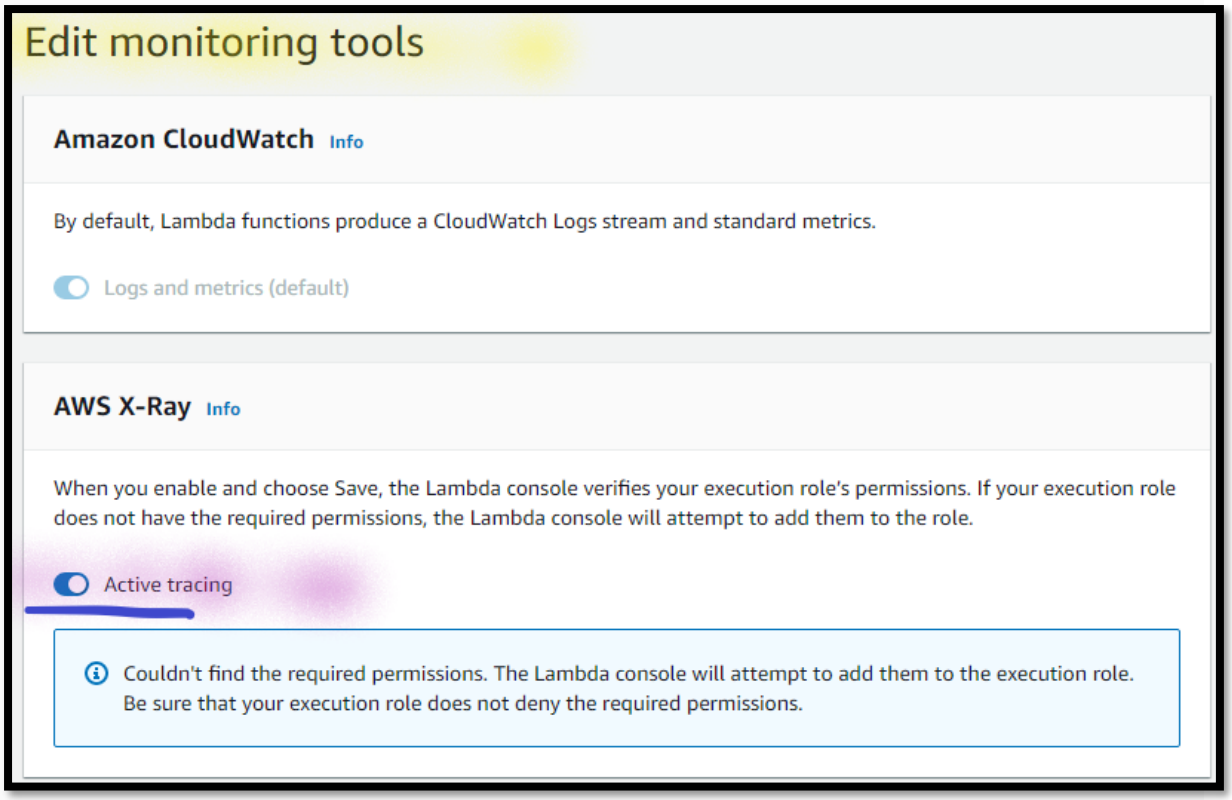
d. ***Go below*** in the Console, in the **Configuration** section.

i. Select **Monitoring and operations tools**.

ii. Select **Edit**.




iii. Enable **Active tracing** in ***AWS X-Ray***.




iv. Select **Save**.

# Task 5: Developing the Python Application

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

## Step 1: Configure the Credentials and Configuration Settings

28. **From** the **Dev Instance**, right click on **Start** & **Run**.

29. In the **Open**, write **cmd**, press **Ok**.

    a. From the ***command line interpreter***, write **aws configure**, press **Enter**.

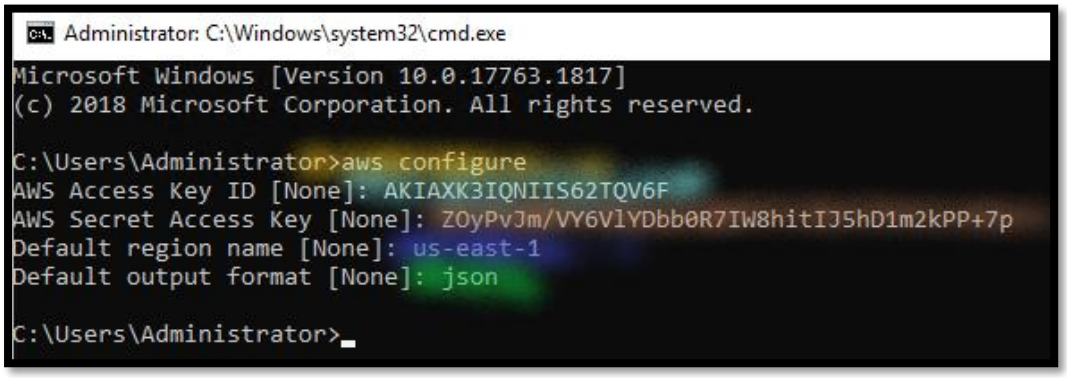
        i. **AWS Access Key ID**: **Type** the IAM User **Lambda-User** **access key** and **enter** to continue.

        ii. **AWS Secret Access Key**: **Type** the IAM User **Lambda-User** **secret access key** and **enter** to continue.

**Note:** You can **copy** the **access key** and **secret access key** of the IAM user **S3-User** from **.csv file** which you have downloaded in the previous step.

        i. **Default region name**: **Type** **us-east-1** and **enter** to continue.

        iii. **Default output format**: **Type** **json** and **enter** to continue.




30. **From** the **Dev Instance**, right click on **Start** & **Run**.

    a. In the **Open**, write **C:\Users\Administrator**, press **Ok**.

    b. Open the, **.aws** folder.

    c. Open the **Credentials** file in **Notepad**.

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

d. Open the **Config** file in **Notepad**.

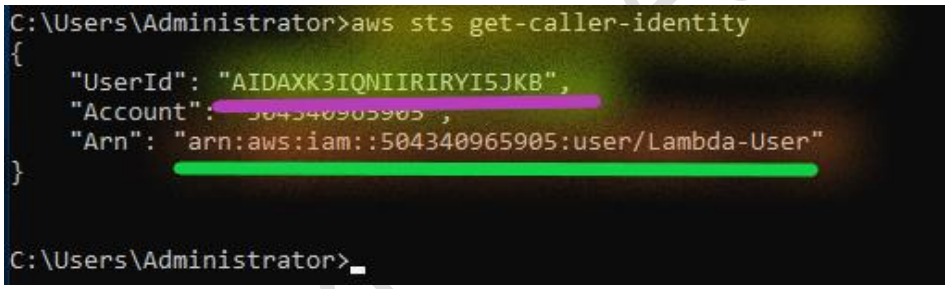
**Note:** You can see the **region** and **output** format details.

### Step 3: Testing the AWS CLI from the Windows Dev Instance

In this task, you would test to see if the policy attached to the role grants the required permissions to list Amazon S3 buckets in your account.

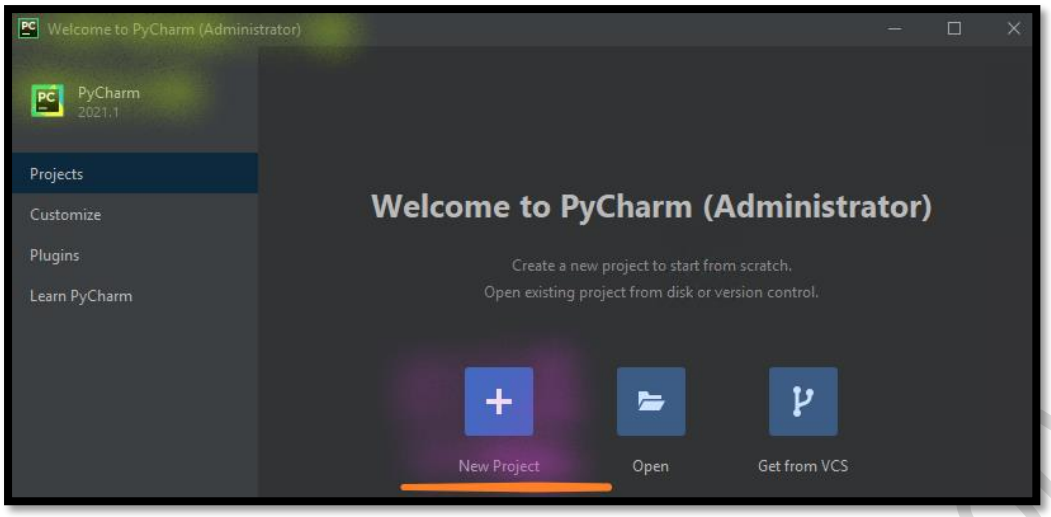
31. From the ***command line interpreter***, write **aws sts get-caller-identity**, press **Enter**.

**Note:** You should see a **below output**.



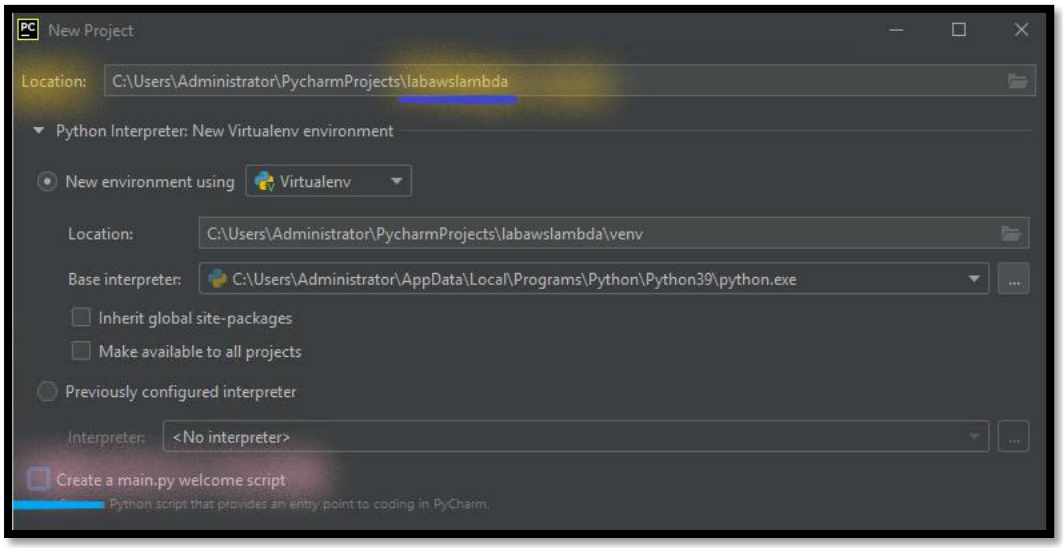

### Step 4: Launch the PyCharm IDE

32. Select and Open the **PyCharm**.
33. In the **PyCharm**:
    a. Select the **New Project**.




b. **Location**: **Replace** the ***existing name*** and write **labawslambda**.

c. **Uncheck** the ***Create a main.py welcome script***.




d. Select the **Create**.

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

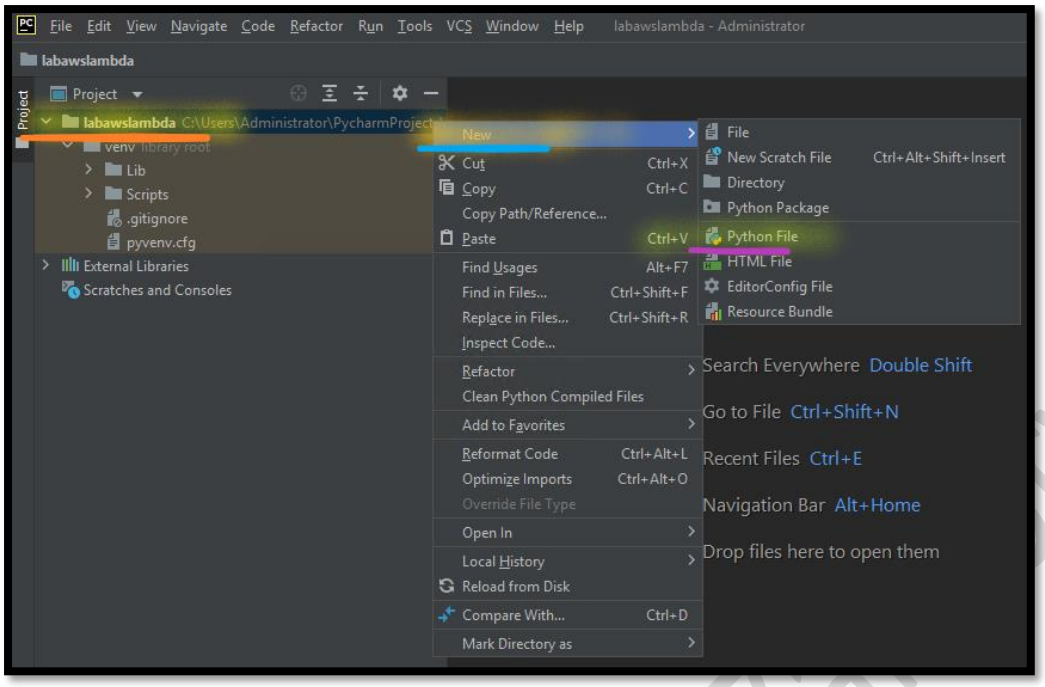
## Step 5: Create the File in the Python Project

34. **Expand** the Python Project ***labawslambda***.

a. **Right-click** on the ***labawslambda*** Python project.

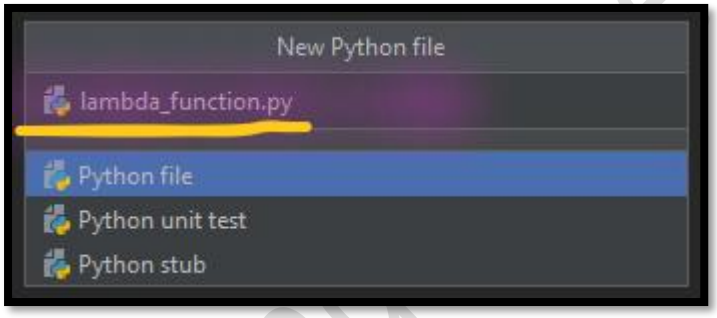
b. Select **New**.

c. Select **Python File**.




d. In the **New python file** page:

i. **File name**: Write **lambda_function.py**.




ii. Select **Enter**.

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

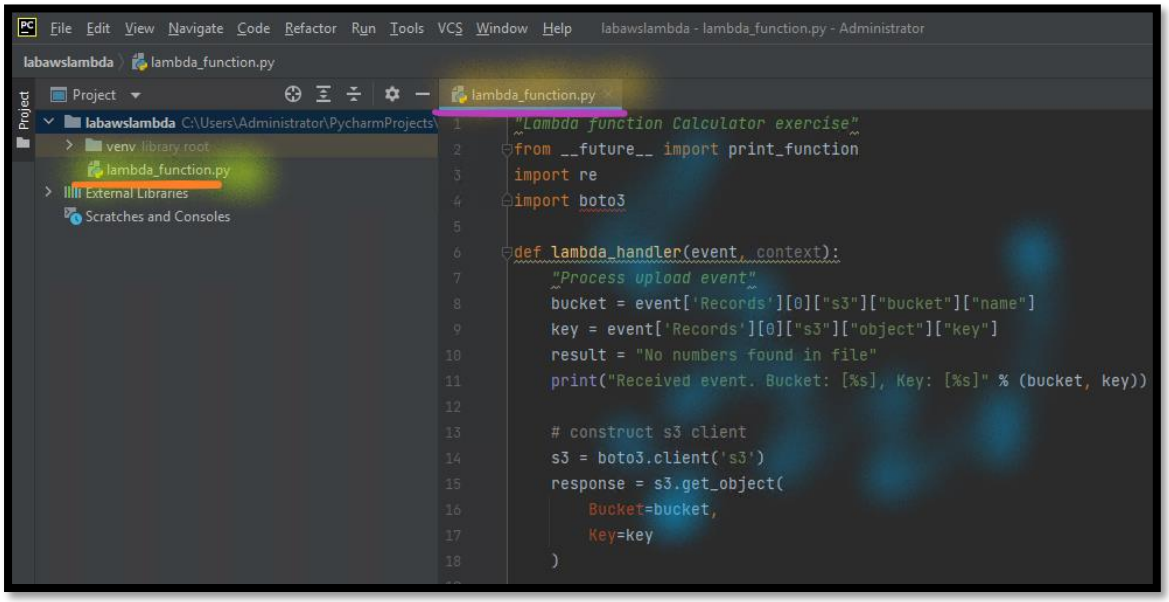
## Step 6: Update the Python Code

35. **Double-click** on the Python file ***lambda_function.py***.

a. **Paste** the Code from ***lambda_function.py*** file.

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

**Note:** Ignore Python version compatibility notification.




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

1. Respond to an Amazon S3 event in Lambda.
2. Retrieve a file from Amazon S3.

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. Select the **File**.

ii. Select the **Save all**.

# 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

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

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

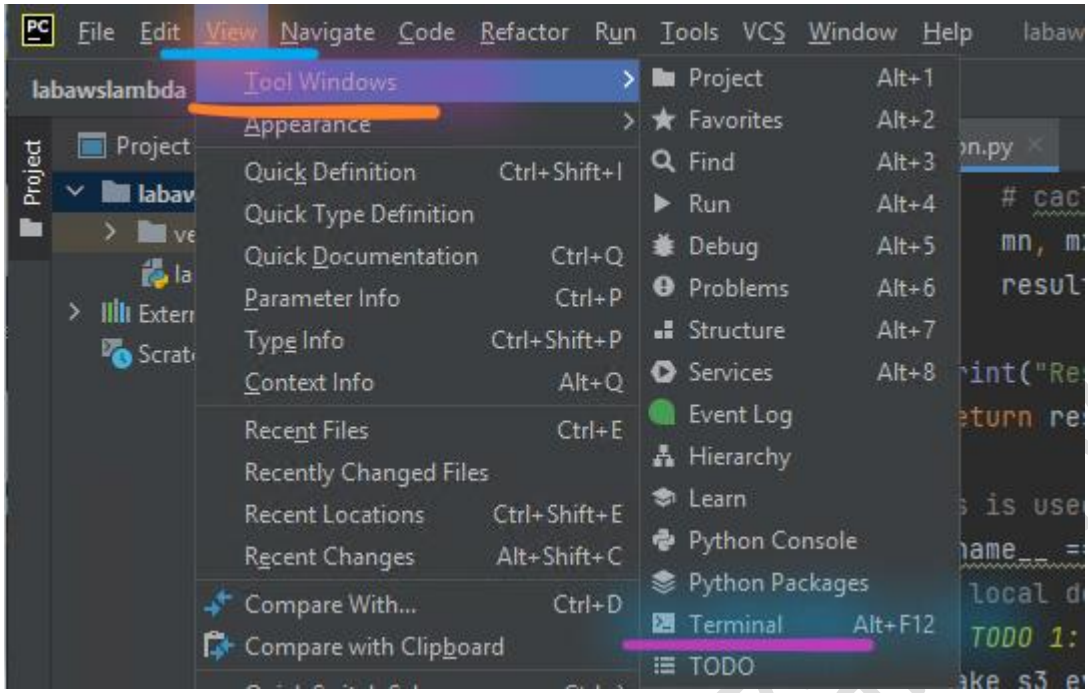
**Note:** Object name **numbers.txt** is already mentioned in the **TODO 1** section.

    b. **From** the **PyCharm IDE**:
        i. Select the **File**.
        ii. Select the **Save all**.

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

## Step 2: Install the Python SDK

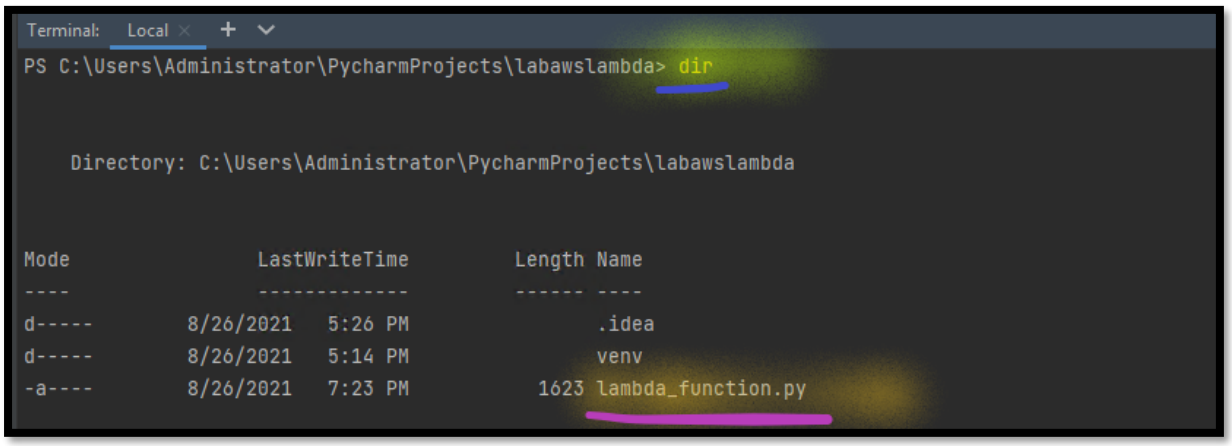
37. **From** the **PyCharm IDE**.
    a. Select **View**.
    b. Select **Tool Windows**.
    c. Select **Terminal**.




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

38. From the ***Terminal***, **Type** **Dir**.

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




39. From the ***Terminal***, **Type** **python lambda_function.py** to run the Python script.

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

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

**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.

40.From the ***Terminal***, **Type** **pip install boto3**.

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

TODO Problems Terminal Python Packages Python Console

Indexing completed in 46 sec. Shared indexes were applied to 2,719 of 5,297 files (51%) (15 minutes ago)

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

```
Terminal: Local
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>
```

TODO Problems Terminal Python Packages Python Console

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

### Step 3: Execute the Lambda function

41. From the ***Terminal***, **Type** **python lambda_function.py** to run the Python script.

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

```
Terminal:  Local  +
PS C:\Users\Administrator\PycharmProjects\labawslambda> python lambda_function.py
Received event. Bucket: [calculator-input-123], Key: [numbers.txt]
Result: Min: 2 Max: 541 Average: 241.33
PS C:\Users\Administrator\PycharmProjects\labawslambda>
```

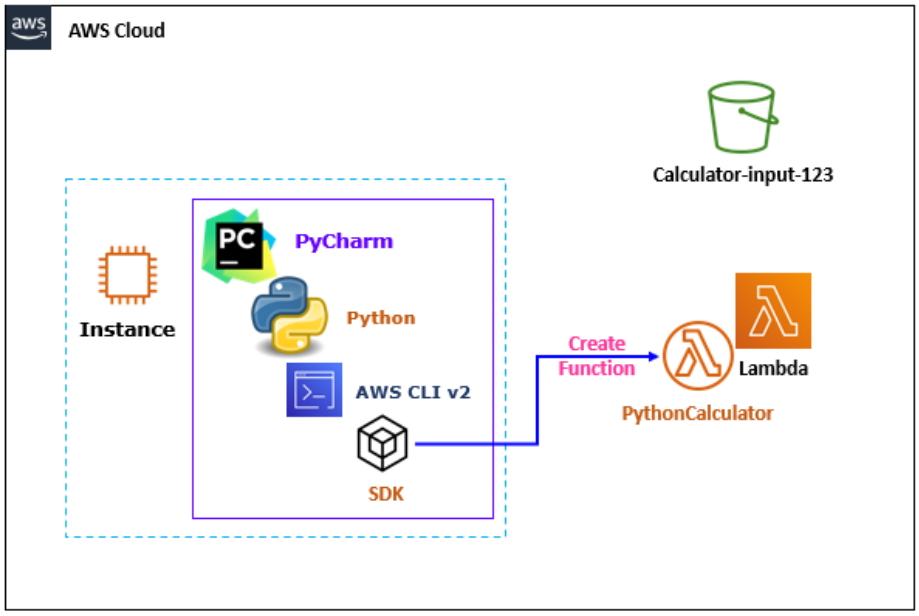
42. From the ***Terminal***, **Copy** the **Current working directory path** in **Notepad**.

```
Terminal:  Local  +

PS C:\Users\Administrator\PycharmProjects\labawslambda>
PS C:\Users\Administrator\PycharmProjects\labawslambda>
```

## 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

43. **From** the **PyCharm IDE**.
    a. 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            # caclulate 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. Select the **File**.
        ii. Select the **Save all**.

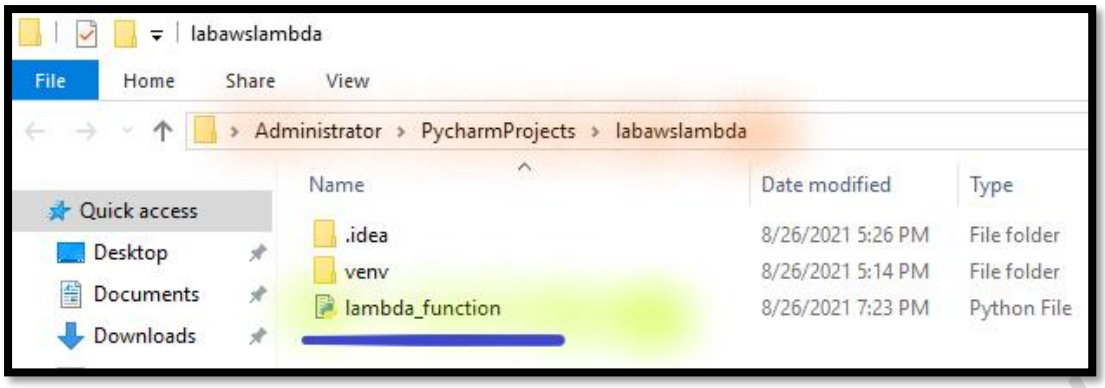
## Step 2: Compress the Python source code file in the Dev Instance

44. **From** the **Dev Instance**, right click on **Start** & **Run**.

45. 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.

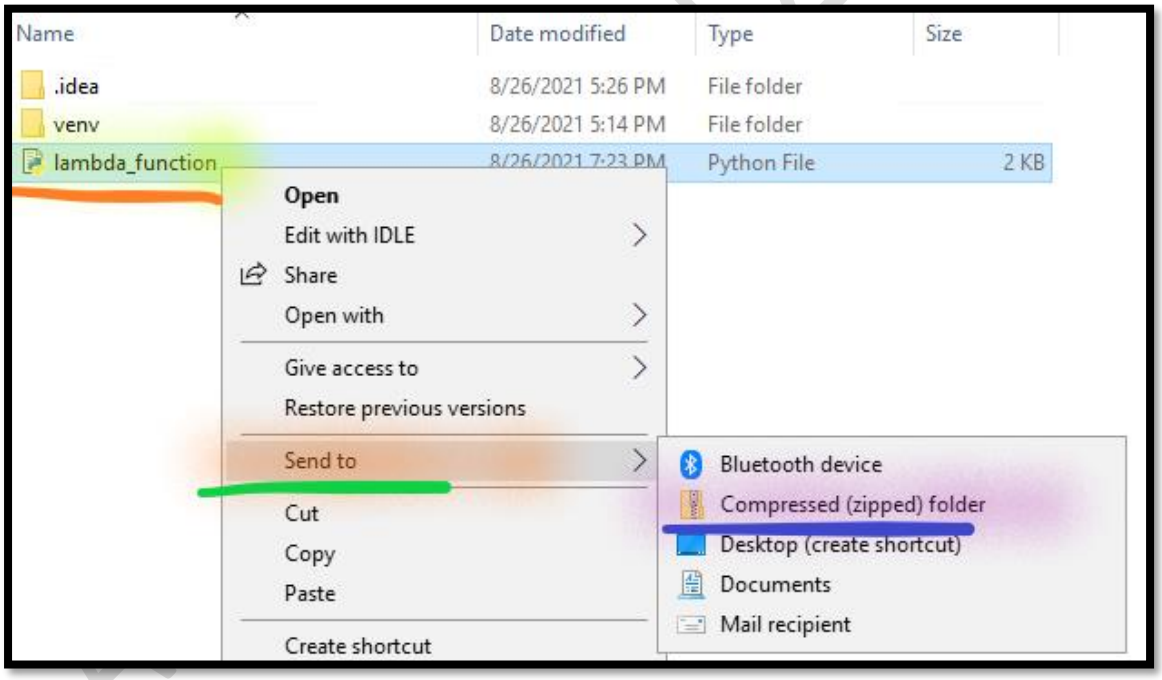



a. **Create** a **zip file** package, **lambda_function.zip**, of your Lambda function:

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

   ii. Select **Send to**.

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




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

## Step 2: Deploy the Lambda function from PyCharm

46. **Return** to the ***PyCharm IDE*** **Terminal**.

    a. From the ***Terminal***, **Type** **Dir**.

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

    b. From the ***Terminal***, **Type**

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

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

**Note:** Check the **output** to make sure that the **code package** has been **deployed successfully**.

```
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": "teNz3vm4WwZCoPWGJHv9VyGW60kMVebEr8sOQS5DcGQ=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "df7b1144-c180-4aa8-937a-e3f15f37840b",
    "State": "Active",
    "LastUpdateStatus": "Successful",
    "PackageType": "Zip"
}
```

## Step 3: Verify the Code from the Lambda

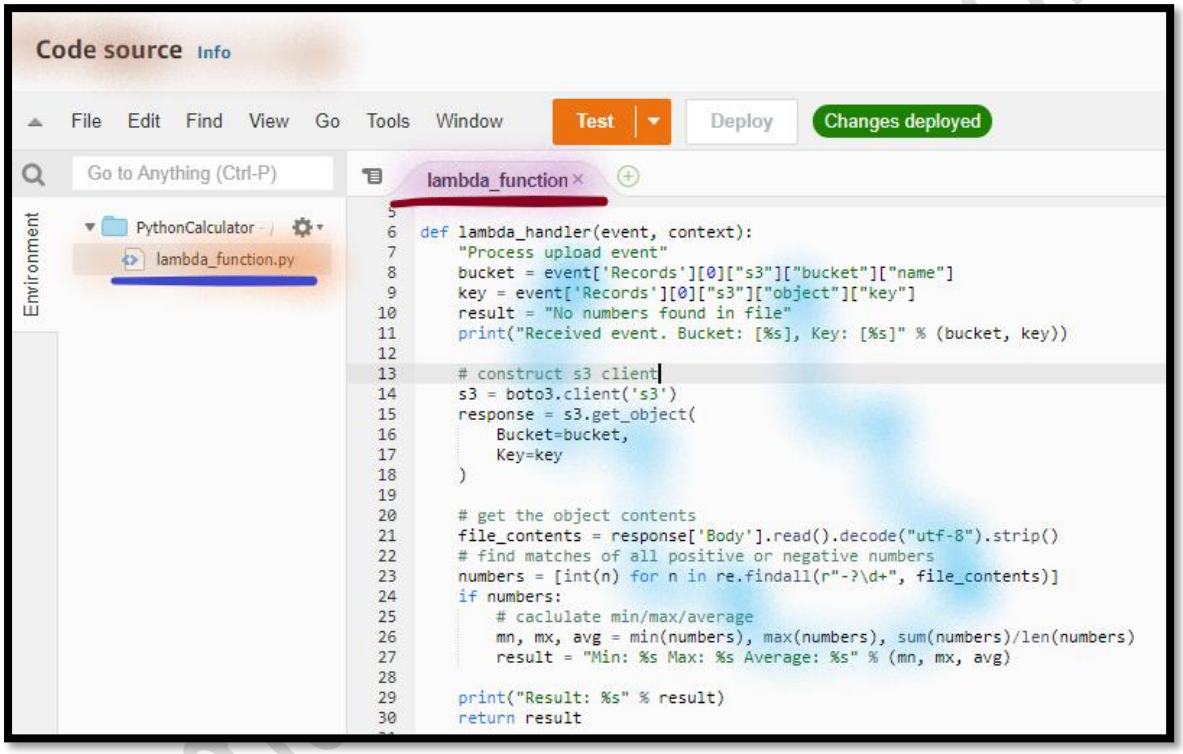
47. **Return** to the ***Lambda Console***.

    a. ***Go below*** in the Console, in the **Code source** section.

        i. Double click on **lambda_function.py**.

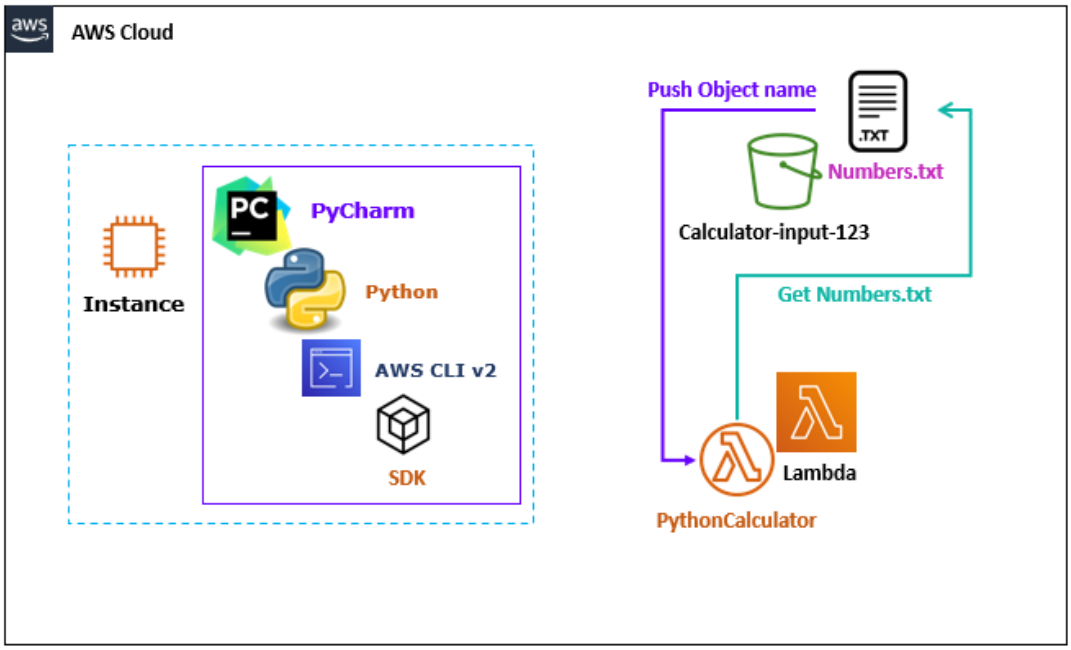
**Note**: You can see the **Deployed code**.

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



## Task 8: Test the Lambda function's invocation

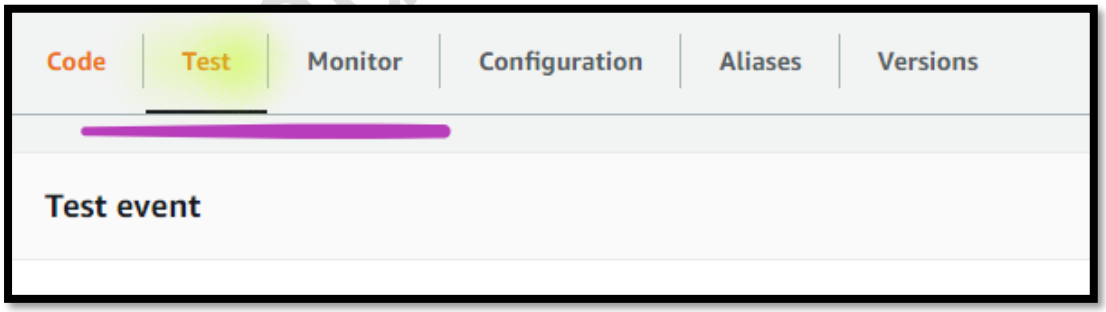
Now that the code has been deployed, test the lambda function invocation.




### Step 1: Update the Test Events

48. **From** the ***Lambda Console***.

    a. ***Go below*** in the Console, **Select** the **Test** section, Next to **Code** section.




    b. Select **Test**.

        i. **Template**: Dropdown and Select **hello-world**.

        ii. **Name**: Write **CalcTest**.

        iii. In the **Event**, **Remove** the existing events and **copy** the below event:

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

iv. **Update** the **REPLACE WITH BUCKET NAME** with the **Bucket name** **you created** in the previous step.

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

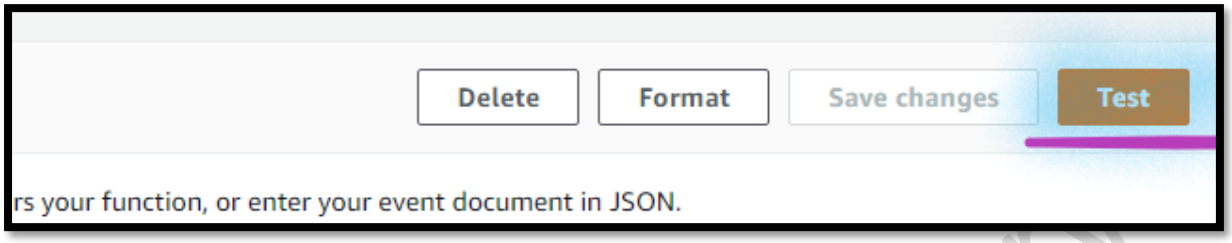
**Note:** Object name **numbers.txt** is already mentioned in the **Test** event.


v. Select **Save changes**.

## Step 2: Validate Your Implementation

49. **From** the ***Test*** *Section:*

a. **Select** the **Test**.




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

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

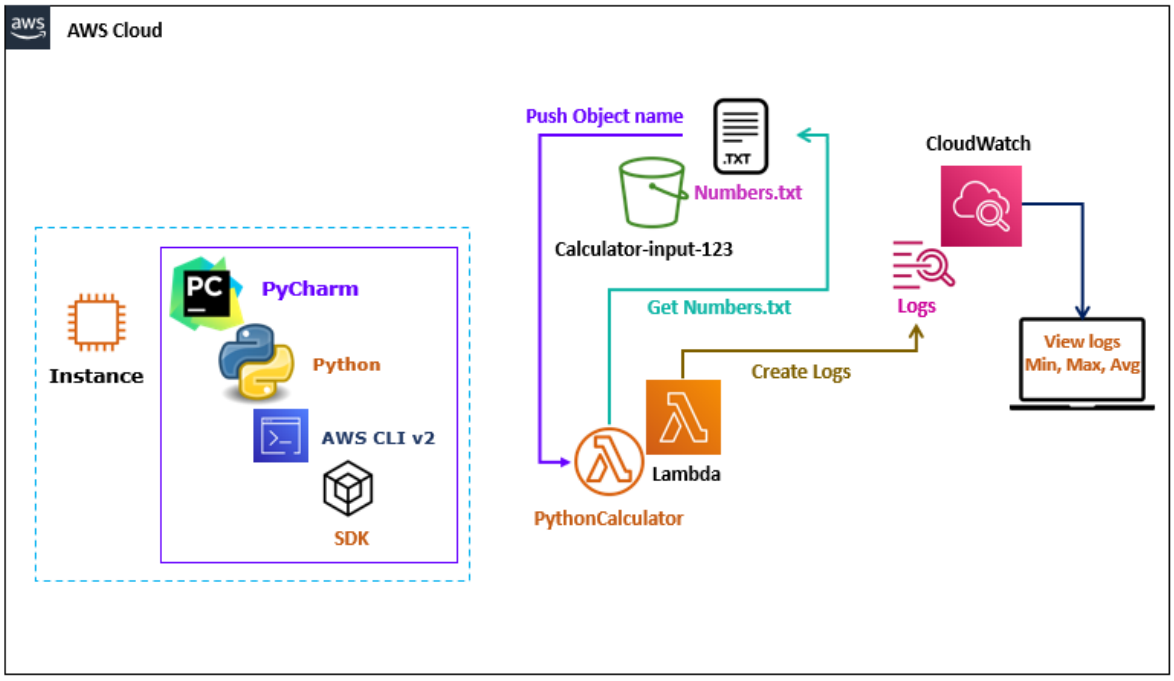
**Note**: You can view the **Min**, **Max** and **Average** count.


**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

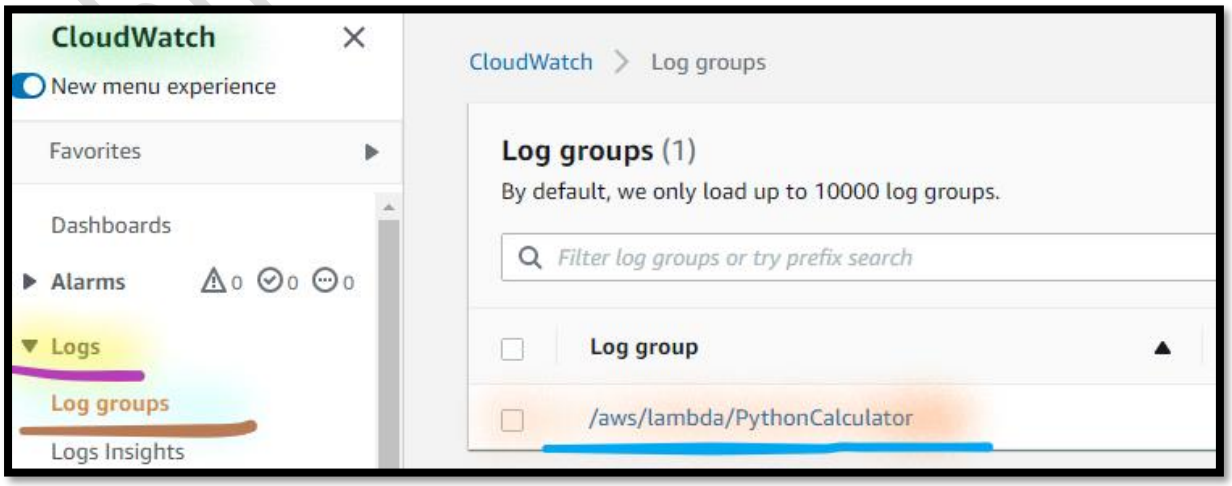



50. In the **AWS Management Console**, on the **Services** menu, click **CloudWatch**.

51. Expand the **Logs**.

    a. Click on the **Log groups**.

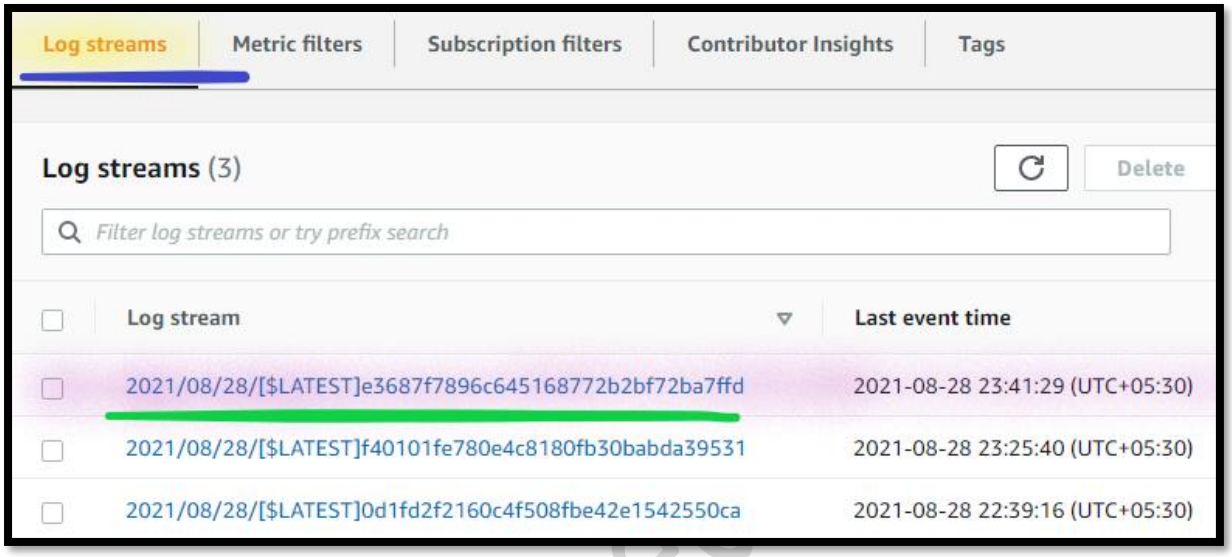
    b. Open the **/aws/lambda/PythonCalculator** log group.



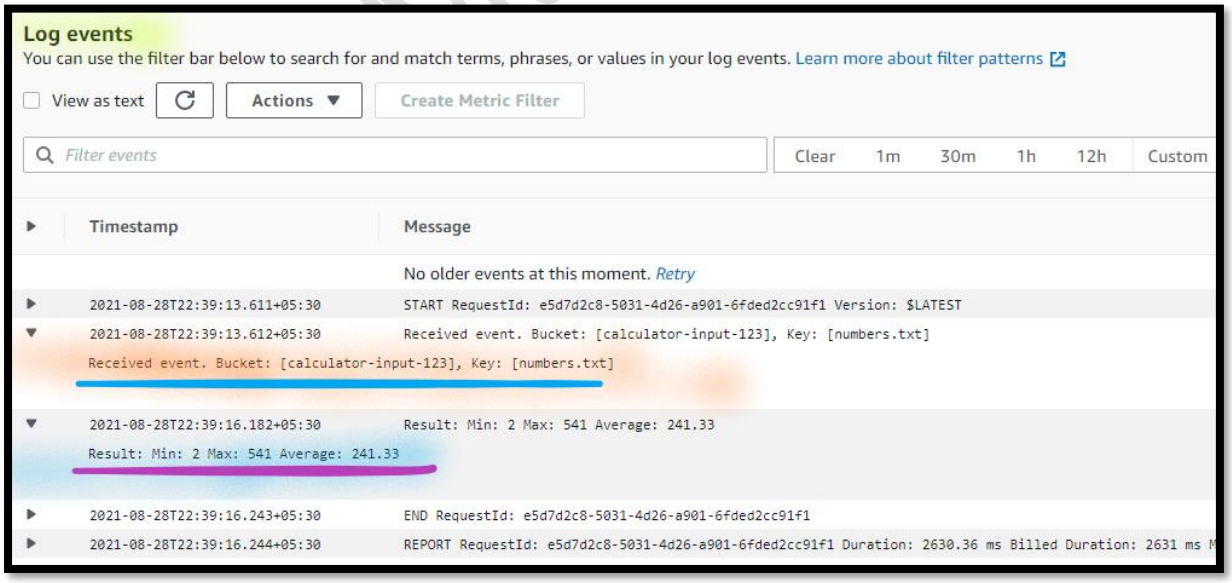
**Note**: You can see the **Log streams** generated by Lambda with the **Date** & **Time**.

c. Select the **Log streams**.

i. Open the **latest** **Log streams**.



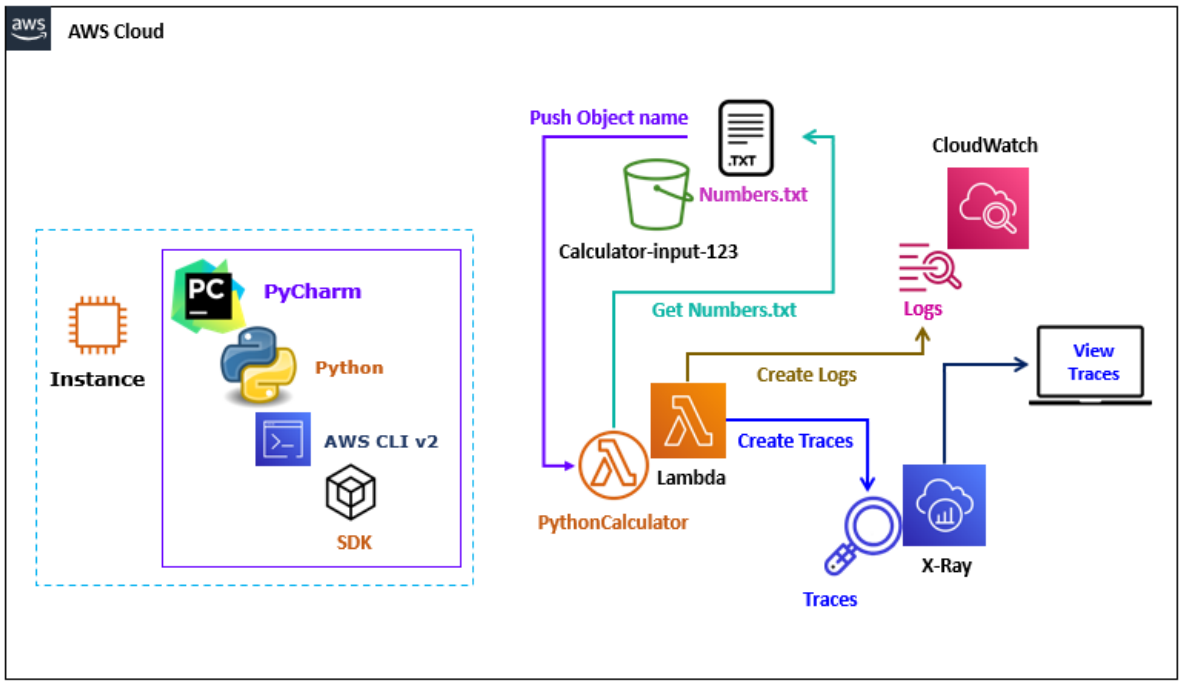

**Note**: **Inspect the contents** of all the recent **entries** of all the log streams. Confirm you see the **output** from your **test case** files.




52. ***Go to left***, Select the **Traces**.

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

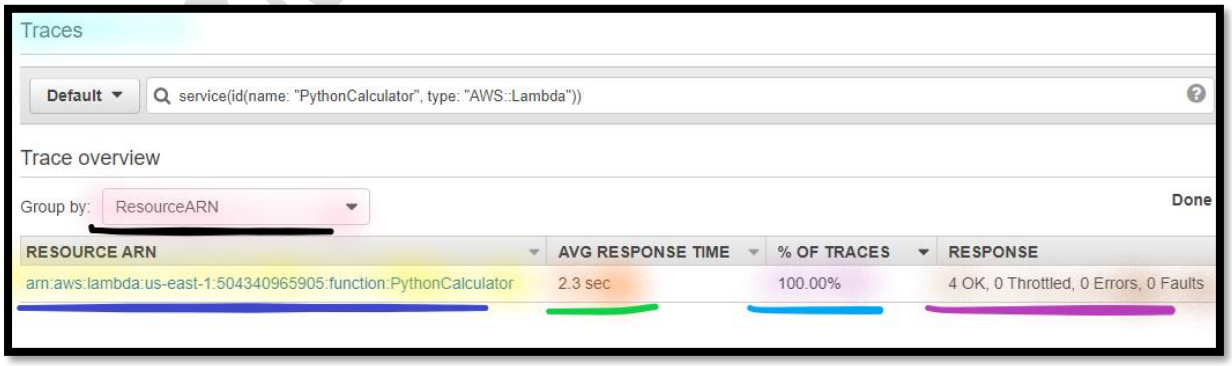
## Step 2: Monitor the X-Ray Traces




53. In the **AWS Management Console**, on the **Services** menu, click **X-Ray**.

54. Select the **Traces**.

    a. **Group by**: Dropdown and Select **ResourceARN**.




    b. **Go below** in the Console, **Open** the **Latest Trace**.

Trace list

| ID | AGE | METHOD | RESPONSE | RESPONSE TIME |
|---|---|---|---|---|
| 1-612a7c4e-68c6c0e40e0468c0687241ec | 31.1 min | | 200 | 3.0 sec |
| 1-612a7899-1cdea91e751d8df957e5a633 | 46.9 min | | 200 | 3.1 sec |
| 1-612a6db9-2d031a5e063a38d4366f6224 | 1.6 hr | | 200 | 3.0 sec |
| 1-612a3211-486dfa3d2ce8064e7f05804c | 5.8 hr | | 202 | 23.0 ms |

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

Traces > Details

1-612a7c4e-68c6c0e40e0468c0687241ec

Timeline | Raw data

| Method | Response | Duration | Age | ID |
|---|---|---|---|---|
| -- | 200 | 3.0 sec | 56.4 min (2021-08-28 18:11:26 UTC) | 1-612a7c4e-68c6c0e40e0468c0687241ec |

Trace Map

Client

3s 1 Request — PythonCalculator AWS::Lambda

2.63s 1 Request — PythonCalculator AWS::Lambda::Function Acct#: 504340965905

| Name | Res. | Duration | Status |
|---|---|---|---|
| PythonCalculator AWS::Lambda | | | |
| PythonCalculator | 200 | 3.0 sec | |
| PythonCalculator AWS::Lambda::Function | | | |
| PythonCalculator | - | 2.6 sec | |
| Initialization | - | 281 ms | |
| Invocation | - | 2.6 sec | |
| Overhead | - | 38.2 ms | |

**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

55. In the **AWS Management Console**, on the **Services** menu, click **S3**.
56. Click the **Buckets** tab.
57. Open **calculator-input-123** bucket.
    a. Select **Objects**.
    b. Click **Upload**.
    c. Click **Add files**.
    d. 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.

    e. 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

58. In the **AWS Management Console**, on the **Services** menu, click **CloudWatch**.
59. Expand the **Logs**.
    a. Click on the **Log groups**.
    b. Open the **/aws/lambda/PythonCalculator** log group.
    c. Select the **Log streams**.
        i. Open the **last three** **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.

## Task 11: Delete the Environment

### Step 1: Delete the Bucket

60. In the **AWS Management Console**, on the **Services** menu, click **S3**.
61. Click the **Buckets** tab.
62. Select **calculator-input-123** bucket.
    a. Select **Empty**.
    b. **Type** **permanently delete** to **Delete all the objects**
    c. Select **Empty**.
    d. Select **Exit**.
63. Select **products-123** bucket.
    a. Select **Delete**.
    b. **Type** **calculator-input-123** bucket name, to **Delete bucket**.
    c. Select **Delete bucket**.

### Step 2: Delete the Lambda Function

64. In the **AWS Management Console**, on the **Services** menu, click **Lambda**.
65. Click the **Functions**.
    a. Select the **PythonCalculator**.
    b. Select **Actions**.
    c. Select **Delete**.

### Step 3: Terminate EC2 Instances

66. In the **AWS Management Console**, on the **Services** menu, click **EC2**.
67. Select **Instances**.
68. Select **Dev-Python-Instance**.
    a. Select **Instance State**.
    b. Select **Terminate instance**.
    c. Select **Terminate**.