## Create a Lambda Function for Addition

### Step 1: Create the Lambda Function

1. **Go to the AWS Lambda Console**:
   - Sign in to the AWS Management Console.
   - Navigate to **Lambda** from the Services menu.
2. **Create a New Function**:

**Step 2: Function Code for Addition**:

```
def lambda_handler(event, context):

    try:

        num1 = event.get('num1')

        num2 = event.get('num2')



        if num1 is None or num2 is None:

            raise ValueError("Both 'num1' and 'num2' must be provided.")

        result = num1 + num2

        return {

            'statusCode': 200,

            'body': {

                'result': result

            }

        }

    except Exception as e:

        return {

            'statusCode': 400,

            'body': {'error': str(e)}

        }
```

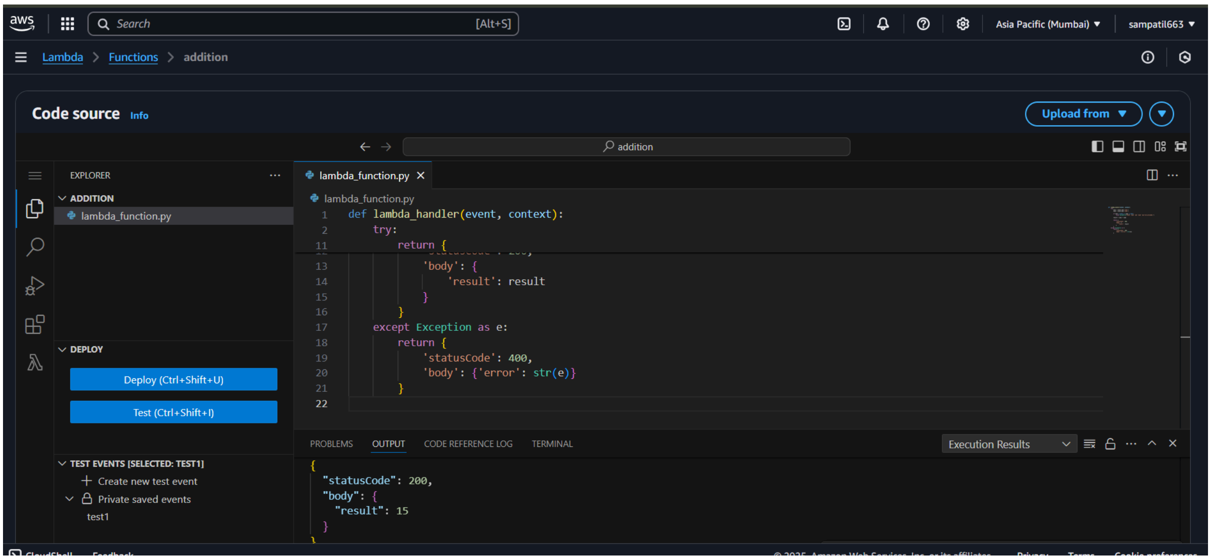
### Step 3: Deploy the Function

### Step 4: Add a Test Event

```
{
  "num1": 5,
  "num2": 10
}
```

### Step 5: Output

```
{
  "statusCode": 200,
  "body": {
    "result": 15
  }
}
```



```
def lambda_handler(event, context):
    try:
        return {
            'body': {
                'result': result
            }
        }
    except Exception as e:
        return {
            'statusCode': 400,
            'body': {'error': str(e)}
        }
```

```
{
  "statusCode": 200,
  "body": {
    "result": 15
  }
}
```

# Create Lambda Function to Save PDF to S3 Bucket

**Step 1: Create an S3 Bucket**
**Step 2: Set Up IAM Policies**
**Step 3: Create Lambda Function to Upload PDF to S3 code**

```
import base64
import boto3
import os
import json

s3_client = boto3.client('s3')

def lambda_handler(event, context):
    try:

        bucket_name = os.environ.get('S3_BUCKET_NAME')
        if not bucket_name:
            raise ValueError("S3_BUCKET_NAME environment variable is not set.")


        file_name = event.get('file_name')
        file_content_base64 = event.get('file_content')

        if not file_name or not file_content_base64:
            raise ValueError("Both 'file_name' and 'file_content' must be provided in the input.")


        missing_padding = len(file_content_base64) % 4
        if missing_padding:
            file_content_base64 += '=' * (4 - missing_padding)


        file_content = base64.b64decode(file_content_base64)


        s3_client.put_object(Bucket=bucket_name, Key=file_name, Body=file_content)

        return {
            'statusCode': 200,
            'body': json.dumps({
                'message': 'File uploaded successfully.',
                'file_name': file_name,
                'bucket_name': bucket_name
            })
        }

    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps({'error': str(e)})
        }
```

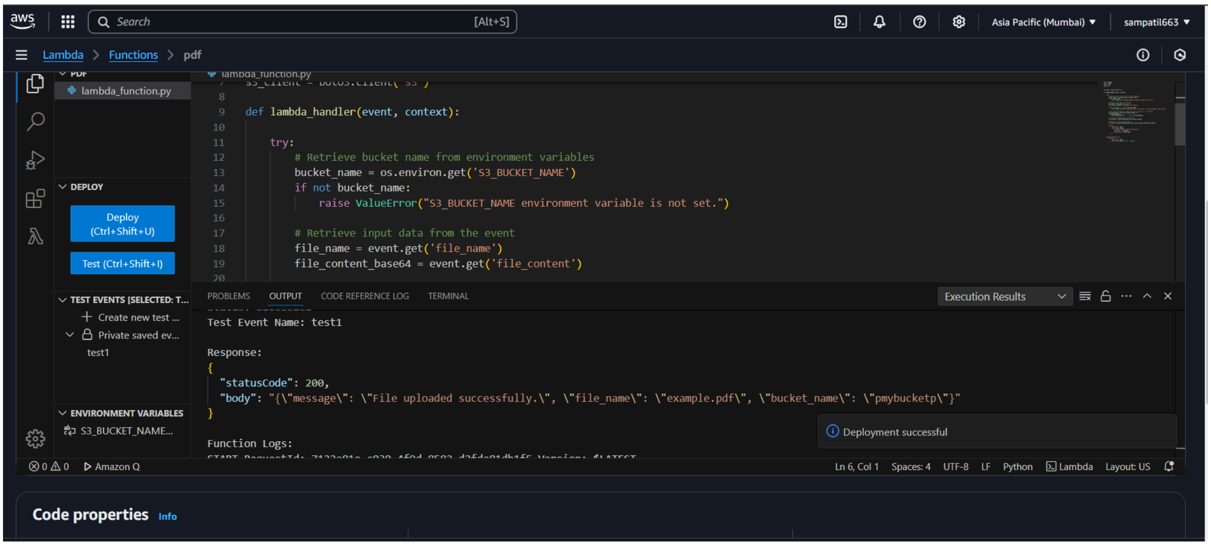
## Step 4: Set the Environment Variable

## Step 5: Test the Lambda Function

```
{
  "file_name": "example.pdf",
  "file_content":
"JVBERi0xLjUKJcTl8uXrp/Og0MTGCjEgMCBvYmoKPDwvTGluZWFyaXplZCAxL0wgNj"
}
```

## Step 6: Output

```
{
  "statusCode": 200,
  "body": "{\"message\": \"File uploaded successfully.\", \"file_name\": \"example.pdf\",
\"bucket_name\": \"pmybucketp\"}"
}
```



```
s3_client = boto3.client('s3')

def lambda_handler(event, context):

    try:
        # Retrieve bucket name from environment variables
        bucket_name = os.environ.get('S3_BUCKET_NAME')
        if not bucket_name:
            raise ValueError("S3_BUCKET_NAME environment variable is not set.")

        # Retrieve input data from the event
        file_name = event.get('file_name')
        file_content_base64 = event.get('file_content')
```

```
Test Event Name: test1

Response:
{
  "statusCode": 200,
  "body": "{\"message\": \"File uploaded successfully.\", \"file_name\": \"example.pdf\", \"bucket_name\": \"pmybucketp\"}"
}

Function Logs:
```