

# Pipeline Scheduling using AWS Lambda

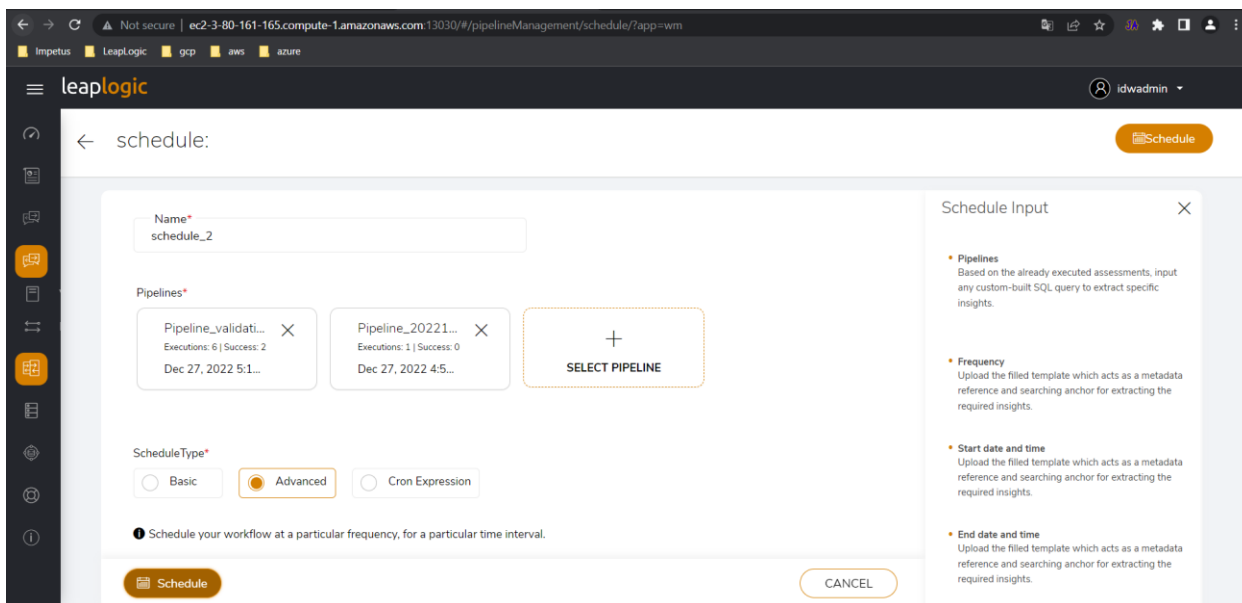
## Contents

|   |    |
|---|----|
| 1. Automatic Push.....                                  | 3  |
| 2. Manual.....  | 5  |
| 2.1 Creating AWS Lambda Manually.....                   | 6  |
| 2.2 Using AWS Lambda to Schedule/Execute Pipeline ..... | 10 |
| 3. Getting Help.....                                    | 13 |

## 1. Automatic Push

This option allows LeapLogic to generate the AWS Lambda code dynamically and push that to the selected AWS cloud environment to generate the AWS Lambda function. It also allows you to trigger the pipeline execution or schedule pipelines. You can provide the credentials of the respective cloud environment in the given format.

1. Go to Operationalization > Parallel Run
2. Select the pipelines that needs to be scheduled

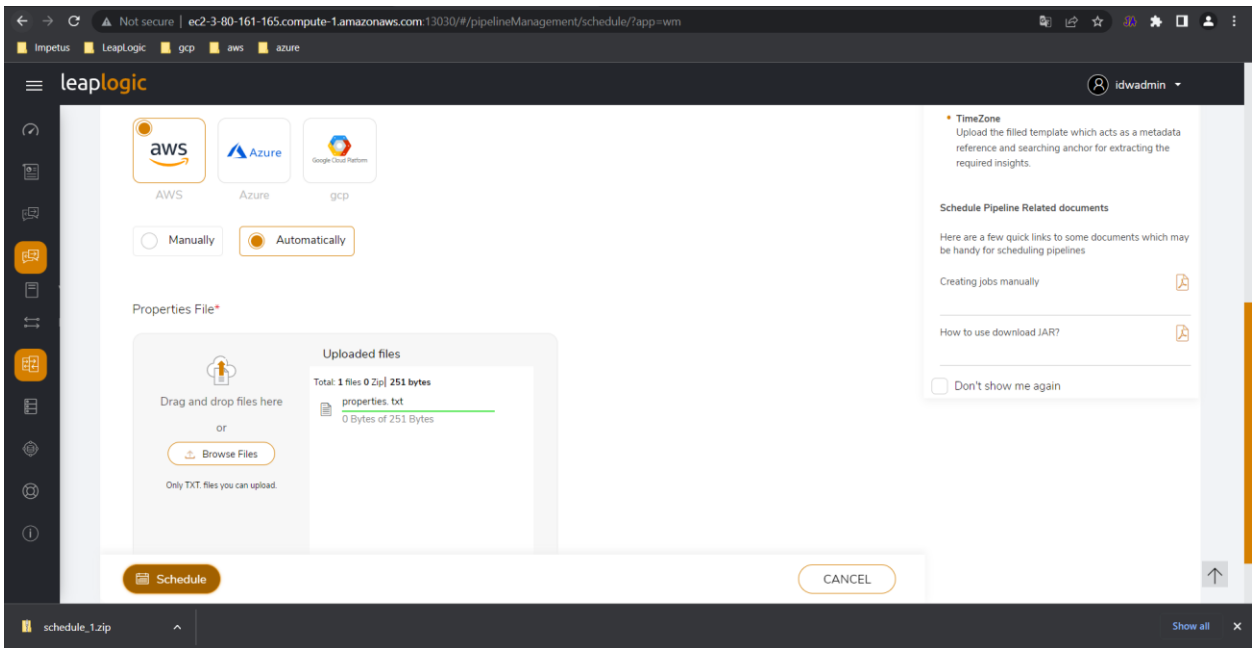


3. Click **Advanced** as schedule type
4. Select Environment for advance trigger. Select **AWS**
5. Click **Automatically**. You can upload properties file as per below format

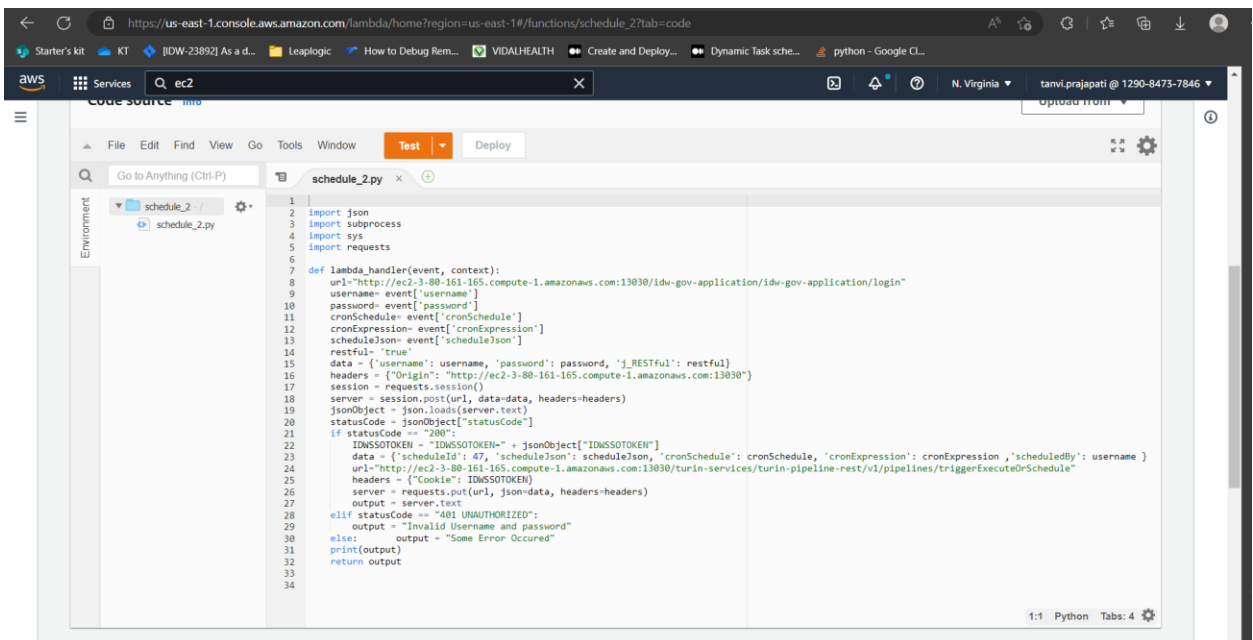
```
accessKeyId=<Access key id with programmatic access to lambda>
secretAccessKey=<Secret Access key with programmatic access to lambda>
arnRole=<ARN Role with lambda creation and execution role>
region=<Region eg. us-east-1>
vpcSubNetId=<VPC subnet Id>
vpcSecurityGroupId=<VPC Security Group Id>
```

### Note

vpcSubNetId and vpcSecurityGroupId are optional. These fields are required if LeapLogic is deployed on AWS cloud and available only in closed network. You must use the same VPC details in which LeapLogic is deployed. If LeapLogic is available in an open network, then do not provide any inputs in these fields.



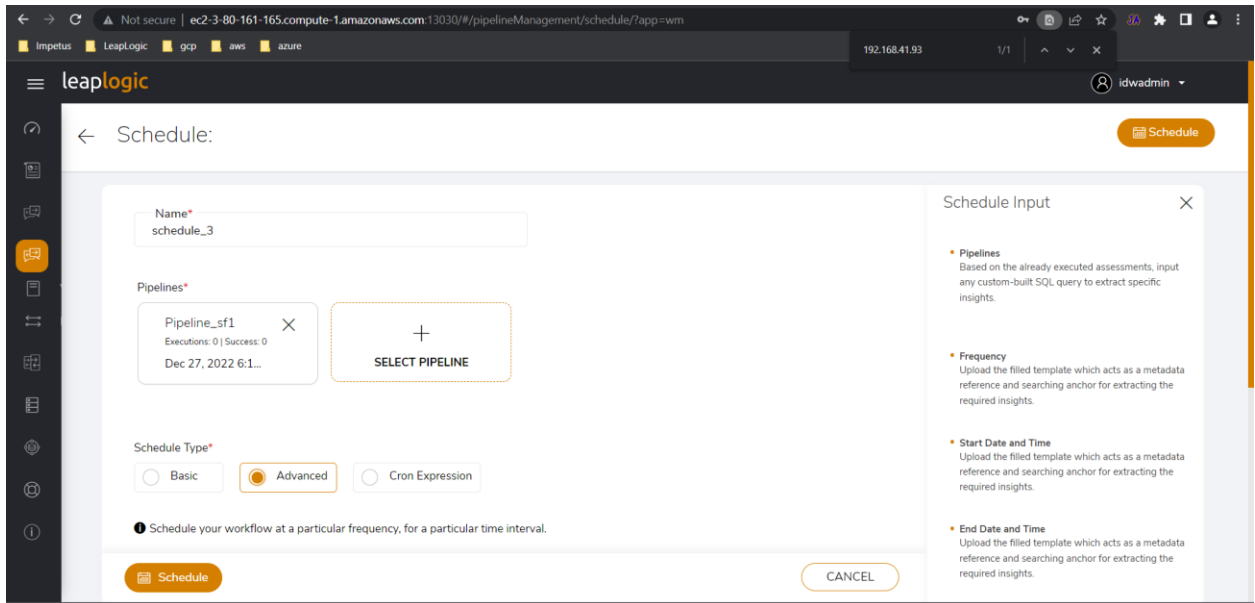
6. Click **Schedule**. This generates AWS Lambda on your cloud environment with the given AWS environment details.



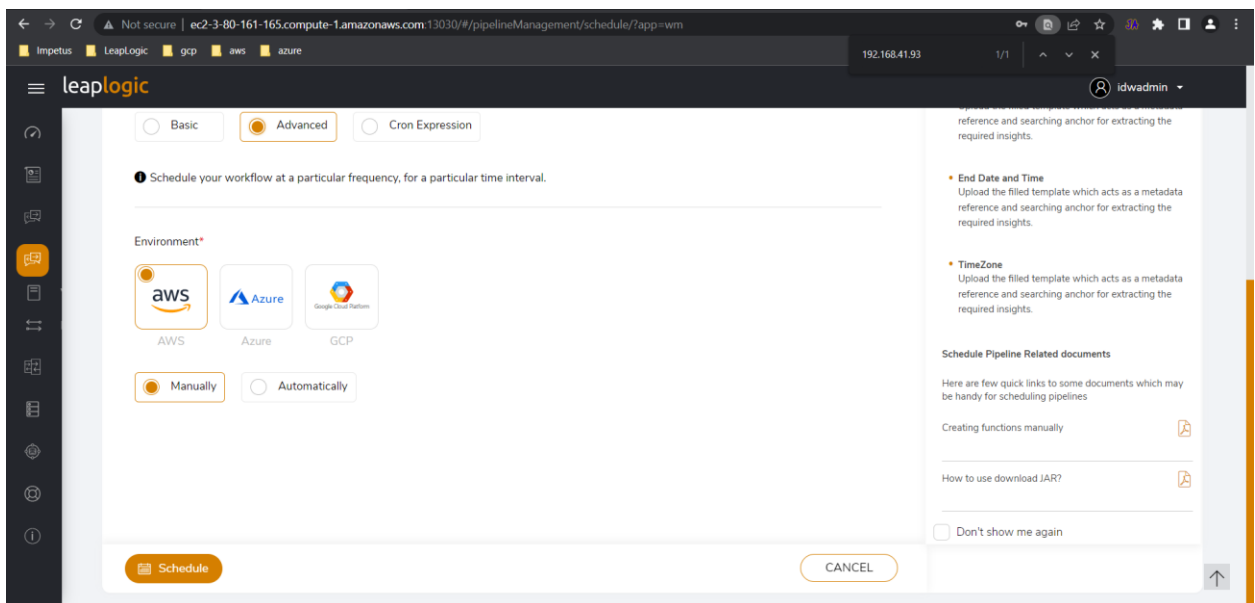
## 2. Manual

This option allows LeapLogic to generate AWS Lambda function code dynamically. You can download the generated code in zip format and generate AWS Lambda manually.

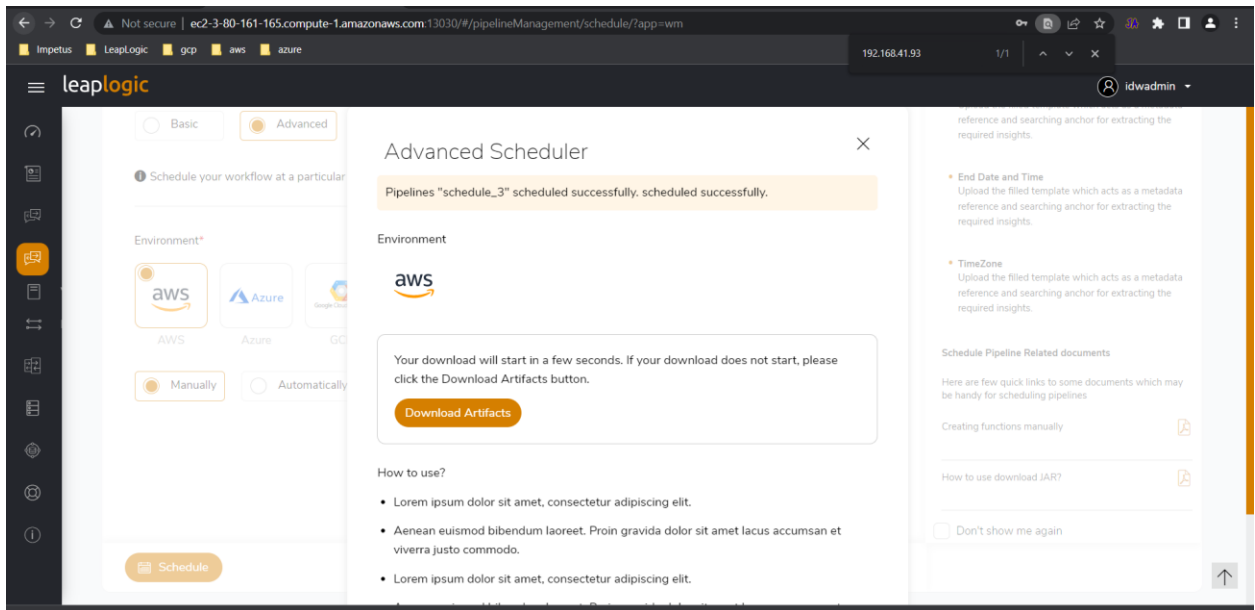
1. Go to Operationalization > Parallel Run
2. Select pipelines that needs to be scheduled



3. Click **Advanced** as schedule type
4. Select Environment for advance trigger and then select **AWS**
5. Click **Manually**



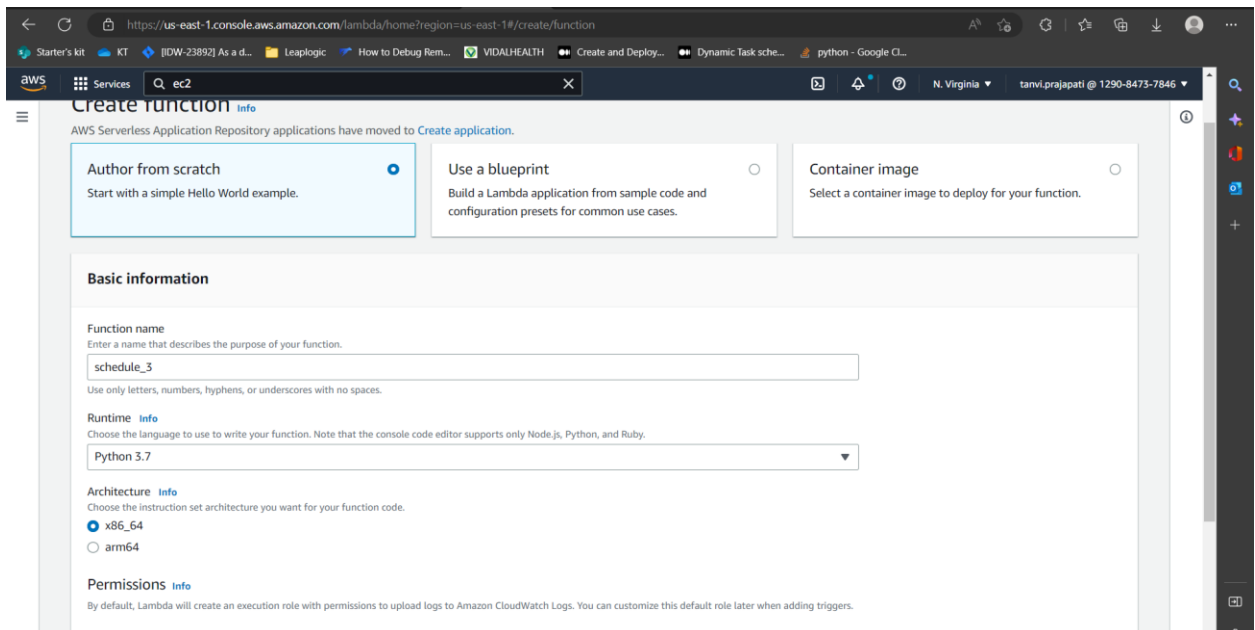
6. Click **Schedule**. The download artifact option appears



7. Download the zip file if not automatically downloaded

## 2.1 Creating AWS Lambda Manually

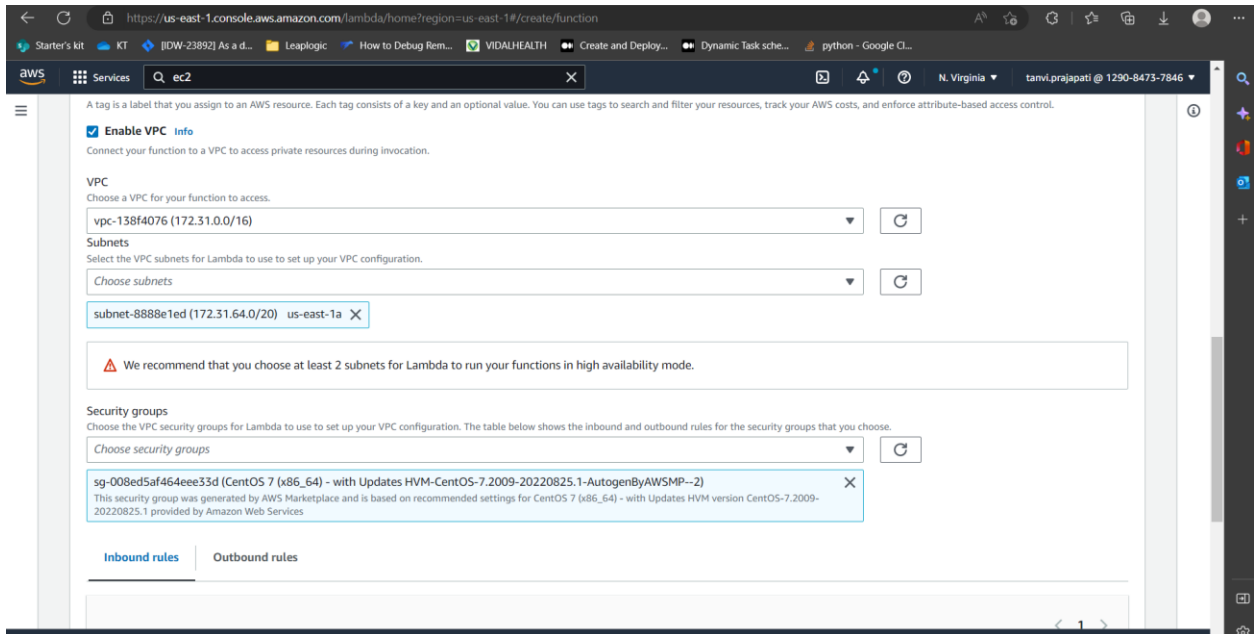
1. Go to AWS Lambda screen from AWS Console
2. Click **Create Function**



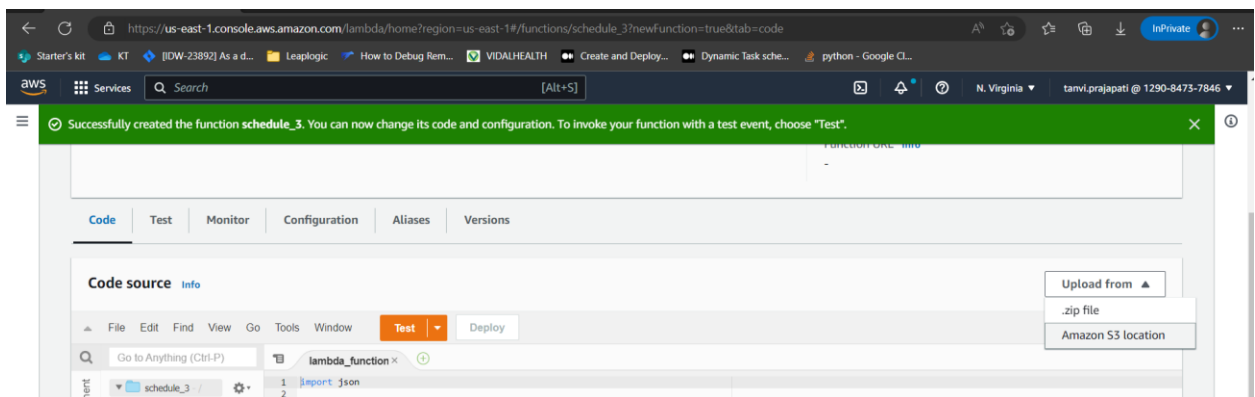
3. Provide appropriate function name and runtime as Python 3.7
4. Choose VPC as required

## Note

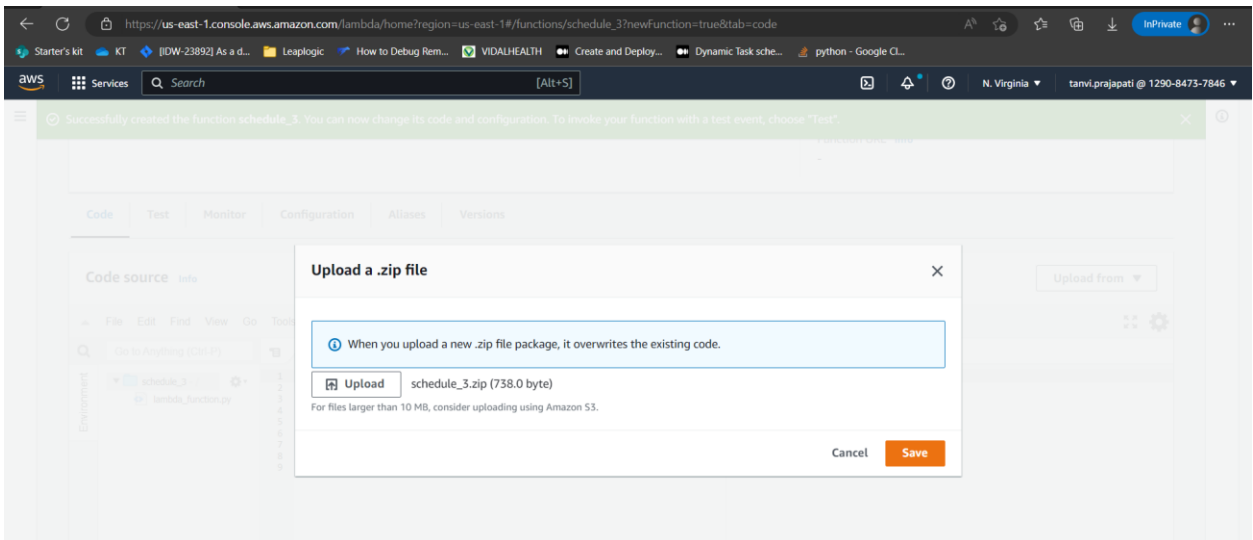
VPC details are optional. These fields are required if LeapLogic is deployed on AWS cloud and available only in closed network. You must use the same VPC details in which LeapLogic is deployed. If LeapLogic is available in open network, then do not provide these two fields.



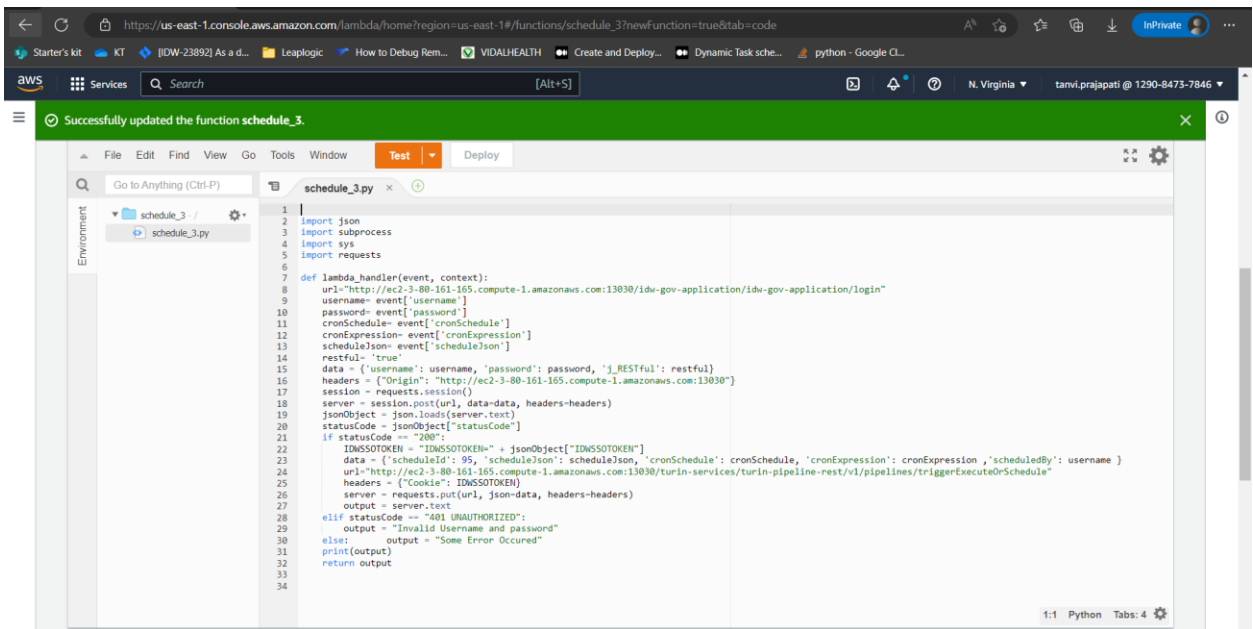
5. Click **Create**. This generates function with dummy details
6. Click **Upload from .zip File**



7. Upload the zip file downloaded at the time of scheduling

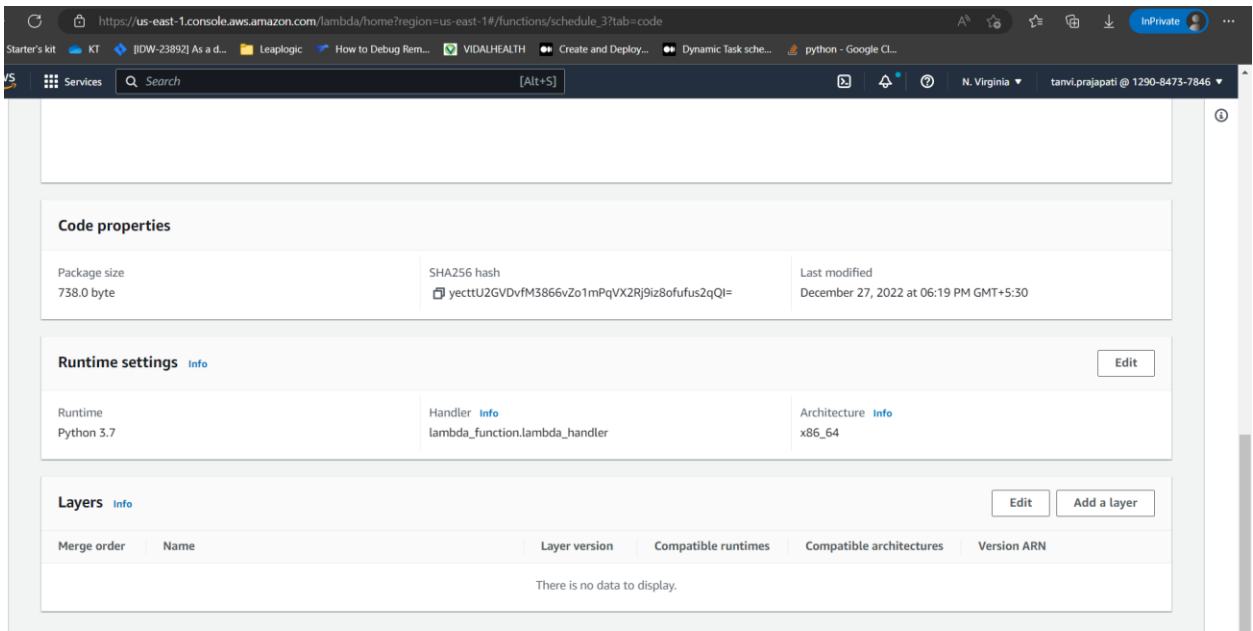


8. This creates the function as shown below

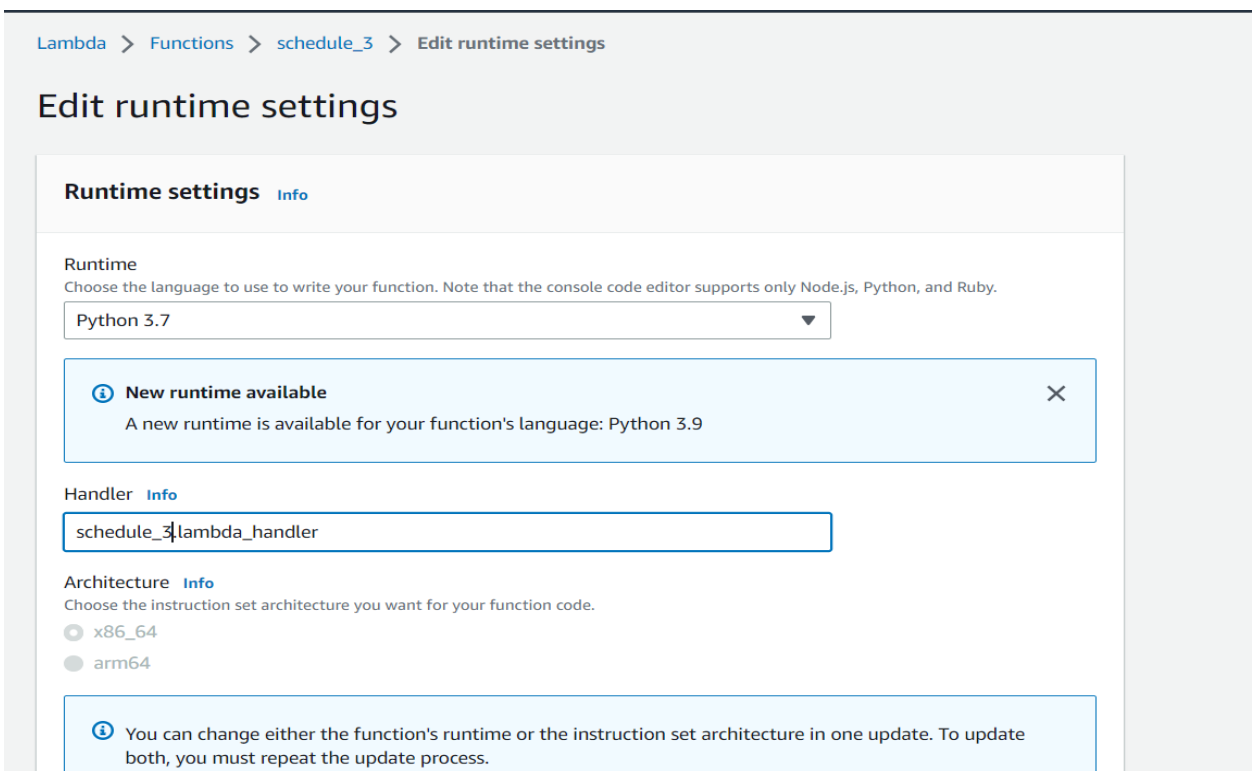


9. You can edit the name of handler as shown below





10. Provide handler name as <ScheduleName>.lambda\_handler (In above example, Schedule name was schedule\_3, so handler name is schedule\_3.lambda\_handler)



11. This completes the process of creating AWS Lambda in a manual way

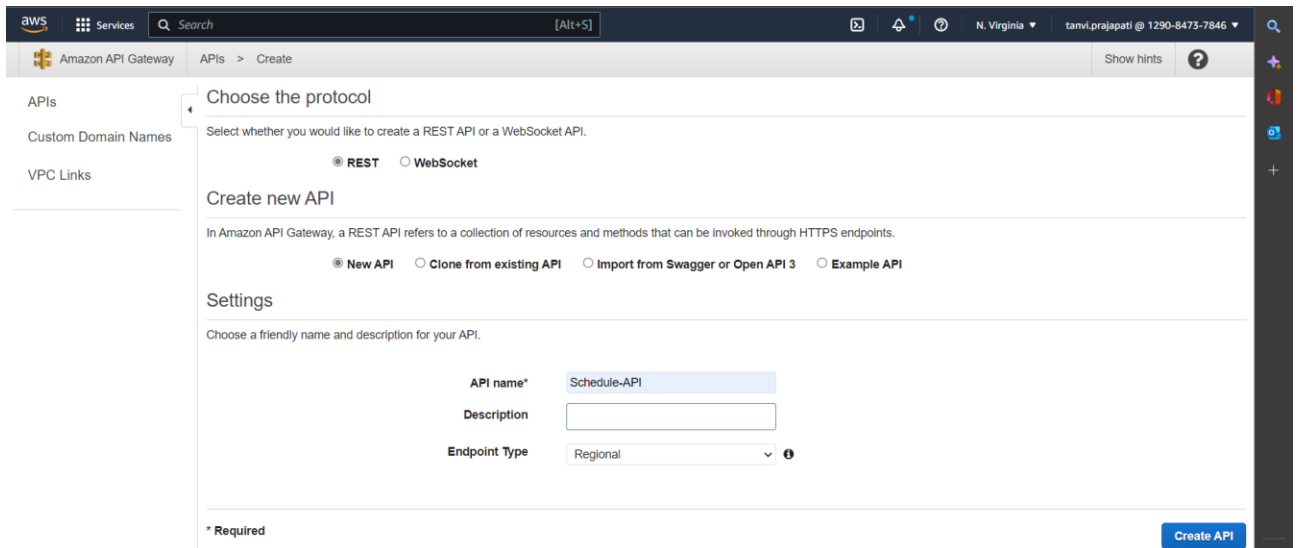
## 2.2 Using AWS Lambda to Schedule/Execute Pipeline

You can now execute or schedule the pipelines as per your requirement by triggering AWS Lambda with appropriate JSON. You can also provide its credentials in JSON to authorize/authenticate beforehand.

You can execute/trigger AWS Lambda by creating trigger events on AWS resources like S3 or by creating API Gateway endpoint.

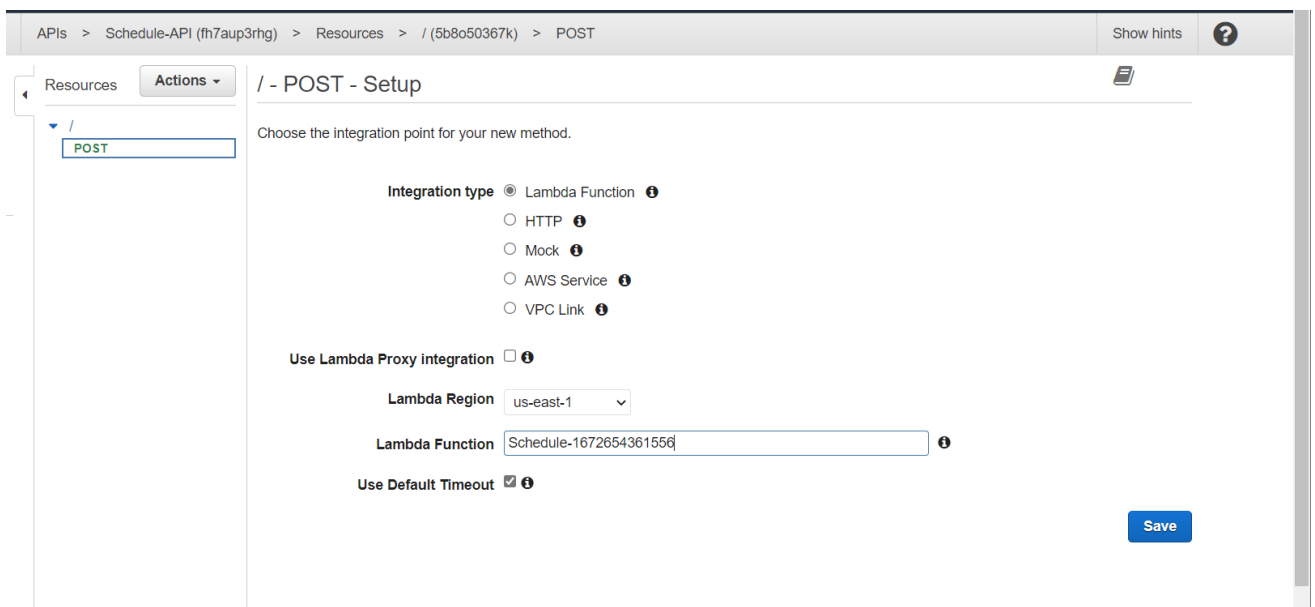
Example: Execute AWS Lambda is by creating API Gateway POST Request

1. Go to API Gateway in AWS Console
2. Create **REST API**



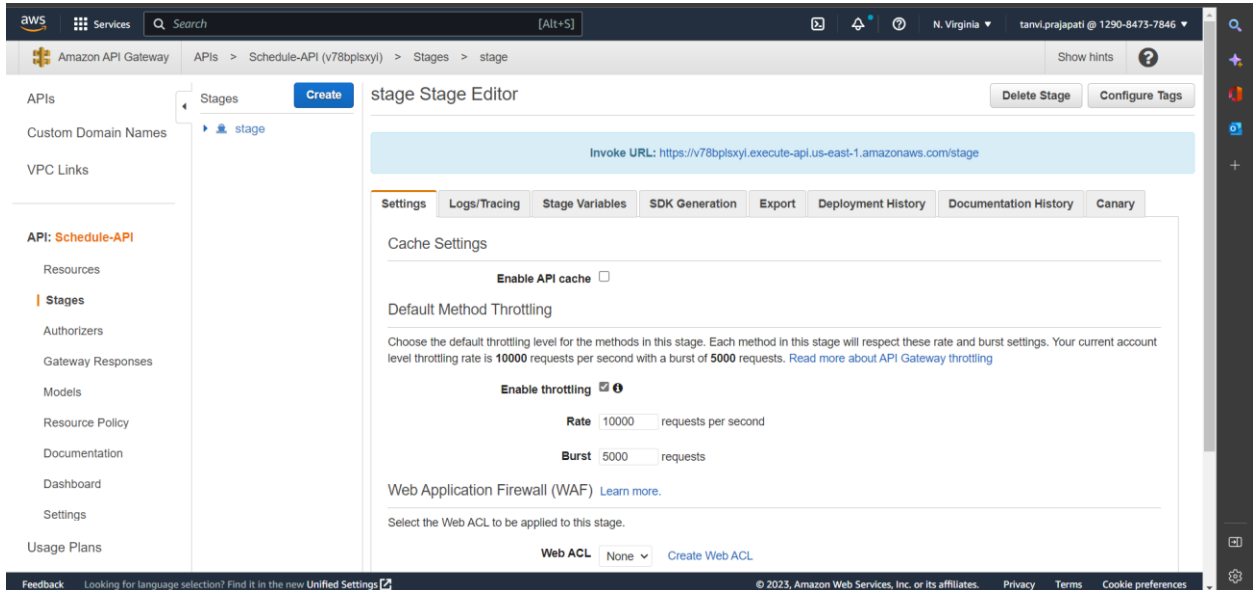
The screenshot shows the AWS API Gateway console interface for creating a new API. The page title is 'Amazon API Gateway' and the breadcrumb is 'APIs > Create'. The main heading is 'Choose the protocol' with a sub-instruction: 'Select whether you would like to create a REST API or a WebSocket API.' There are two radio buttons: 'REST' (selected) and 'WebSocket'. Below this is the 'Create new API' section, which states: 'In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.' It offers four options: 'New API' (selected), 'Clone from existing API', 'Import from Swagger or Open API 3', and 'Example API'. The 'Settings' section asks to 'Choose a friendly name and description for your API.' It includes three input fields: 'API name\*' (containing 'Schedule-API'), 'Description' (empty), and 'Endpoint Type' (a dropdown menu set to 'Regional'). A '\* Required' note is at the bottom left, and a 'Create API' button is at the bottom right.

3. From Actions, click Create Method and choose POST request
4. Choose appropriate region and Lambda function and save

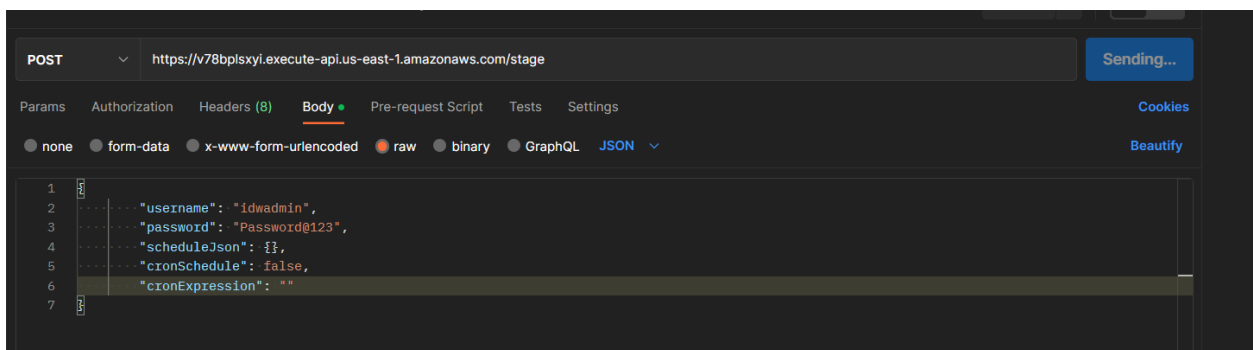


The screenshot shows the AWS API Gateway console interface for setting up a new method. The breadcrumb is 'APIs > Schedule-API (fh7aup3rhg) > Resources > / (5b8o50367k) > POST'. The main heading is '/ - POST - Setup'. The sub-instruction is 'Choose the integration point for your new method.' There are four radio buttons for 'Integration type': 'Lambda Function' (selected), 'HTTP', 'Mock', 'AWS Service', and 'VPC Link'. Below this is the 'Use Lambda Proxy integration' checkbox, which is unchecked. There are three input fields: 'Lambda Region' (a dropdown menu set to 'us-east-1'), 'Lambda Function' (containing 'Schedule-1672654361556'), and 'Use Default Timeout' (a checked checkbox). A 'Save' button is at the bottom right.

5. From Actions, click **Deploy API**



6. Once deployed, you receive an API URL which can be executed from postman with appropriate JSON



i. **JSON to execute pipelines**

```
{
  "username": "idwadmin",
  "password": "Password@123",
  "scheduleJson": {},
  "cronSchedule": false,
  "cronExpression": ""
}
```

ii. **JSON to schedule with basic details**

```
{
  "username": "idwadmin",
  "password": "Password@123",
  "scheduleJson": {
    "startDate": "2022-12-27",
  }
}
```

```
"endDate": "2022-12-27",
"startTime": "18:05",
"endTime": "18:06",
"minutes": 0,
"timezone": "Asia/Calcutta",
"frequency": "NONE"
},
"cronSchedule": false,
"cronExpression": ""
}
```

### Note

Frequency can be NONE(Once), DAILY, WEEKLY, MONTHLY, YEARLY, CUSTOM. With CUSTOM, you can provide minutes to indicate interval of minutes for schedule.

#### iii. **JSON to schedule with cron expression**

```
{
  "username": "idwadmin",
  "password": "Password@123",
  "scheduleJson": {},
  "cronSchedule": true,
  "cronExpression": "0 30 18 27 12 ? 2022"
}
```

7. You can curl the API URL as well
8. You can integrate the API URL in this application as well

### 3. Getting Help

Contact LeapLogic technical support at [info@leaplogic.io](mailto:info@leaplogic.io)