Scheduled Start/Stop of EC2 Instances using Lambda and Cloud Watch Events

1. First Go to **IAM** in AWS Console
2. In this **IAM** go to **Policies** and select **create policy.**
3. Next go to **visual editor** seclect **Service** : **EC2**

**Actions** : **Manual Actions --- Access level :**

* **List,**
* **Read,**
* **Tagging,**
* **Write.**

**Write 🡪 stopinstances** & **startinstances** (select both).

1. **Resources : Specific**

* **All Resources**

1. **OR** copy the code into **JSON** direct

**Lambda start/stop policy :**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Resource": "arn:aws:logs:\*:\*:\*"

},

{

"Effect": "Allow",

"Action": [

"ec2:Start\*",

"ec2:Stop\*"

],

"Resource": "\*"

}

]

}

1. **Review policy :**

**Name** : **Start\_Stop\_Lambda\_Policy** -----> **Create policy**

1. Next go to **create role** in **IAM :**

**Select type of trusted entity** : **AWS Service**  ---->

1. **Choose the service that will use this role** : **Lambda ----->** Next **Permissions .**
2. **Attach permissions policy : Select Start\_Stop\_Lambda\_Policy (Already Created : Step 2-6) and go to tags.**
3. Next go to **Review** enter role **Name** : **Lambda \_Ec2\_Start\_Stop -----> Create Role.**
4. **Go to Lambda select Create Function -----> Author From Scratch**

**Function Name : StartEC2Instance**

**Runtime : Node.js 10.x/Python 3.6**

**Permissions :**

**Execution Role** : Choose an existing role (**Lambda \_Ec2\_Start\_Stop**)

**Hit enter Create function**.

1. Go to the **StartEC2Instance function configuration**
2. **Go to function code section (index.js/index.py) : delete the existing code and replace below code :**

**StartEC2Instance : Node.js code**

const AWS = require('aws-sdk');

exports.handler = (event, context, callback) => {

const ec2 = new AWS.EC2({ region: 'ap-south-1' });

ec2.startInstances({ InstanceIds: ['i-0fc442cc86d2ac356', 'i-03b3ce95558561e2b'] }).promise()

.then(() => callback(null, `Successfully started ${event.instanceId}`))

.catch(err => callback(err));

};

**-------------------------------------------**

**Python code for start instance :**

**-------------------------------------------**

import boto3

# Enter the region your instances are in. Include only the region without specifying Availability Zone; e.g.; 'us-east-1'

region = 'XX-XXXXX-X'

# Enter your instances here: ex. ['X-XXXXXXXX', 'X-XXXXXXXX']

instances = ['X-XXXXXXXX']

def lambda\_handler(event, context):

ec2 = boto3.client('ec2', region\_name=region)

ec2.start\_instances(InstanceIds=instances)

print 'started your instances: ' + str(instances)

**StopEC2Instance : Node.js code**

const AWS = require('aws-sdk');

exports.handler = (event, context, callback) => {

const ec2 = new AWS.EC2({ region: 'ap-south-1' });

ec2.stopInstances({ InstanceIds: ['i-0fc442cc86d2ac356', 'i-03b3ce95558561e2b'] }).promise()

.then(() => callback(null, `Successfully stopped ${event.instanceId}`))

.catch(err => callback(err));

};

---------------------------------------------

**1.2)Python code for stop instance :**

---------------------------------------------

import boto3

# Enter the region your instances are in. Include only the region without specifying Availability Zone; e.g., 'us-east-1'

region = 'XX-XXXXX-X'

# Enter your instances here: ex. ['X-XXXXXXXX', 'X-XXXXXXXX']

instances = ['X-XXXXXXXX']

def lambda\_handler(event, context):

ec2 = boto3.client('ec2', region\_name=region)

ec2.stop\_instances(InstanceIds=instances)

print 'stopped your instances: ' + str(instances)

1. Change **Timeout** 3sec to 10 sec.
2. And Next **Configure Test** event ----> enter the event name and save Test. Properly test it working or not.
3. Repeat steps **11 to 15** **for StopEC2Instance** also.
4. Go to **Cloudwatch** event setup for **cron schedule** ----> **Rules**
5. Select **create rule** ----> **Event Source ----->**

**Event Pattern**

* **Schedule ----> Select Cron Expression :**

|  |  |
| --- | --- |
| **Schedule** | Cron expression  30 2 \* \* ? \* |

**If this is correct expression showing metrics like this below :**

|  |  |
| --- | --- |
| **Next 10 Trigger Date(s)** | 1. Tue, 11 Jun 2019 02:30:00 GMT 2. Wed, 12 Jun 2019 02:30:00 GMT 3. Thu, 13 Jun 2019 02:30:00 GMT 4. Fri, 14 Jun 2019 02:30:00 GMT 5. Sat, 15 Jun 2019 02:30:00 GMT 6. Sun, 16 Jun 2019 02:30:00 GMT 7. Mon, 17 Jun 2019 02:30:00 GMT 8. Tue, 18 Jun 2019 02:30:00 GMT 9. Wed, 19 Jun 2019 02:30:00 GMT 10. Thu, 20 Jun 2019 02:30:00 GMT |

1. Now go to **Target** ----> **add target** ----> select **topic** **: Lambda function**

**Function** : **StartEc2Instance** ---> **configure details**

1. Repeat steps **18 & 19** **for StopEC2Instance** also.