

Here's a detailed step-by-step lab guide based on "Automating EC2 with AWS Lambda Using Boto3 – Lab 2" : stops running EC2 instances across all AWS regions:

Objectives

- Build a Lambda function that lists **all AWS regions**, identifies **running EC2 instances**, and stops them.

Step-by-Step Lab

1. Prerequisites

- AWS account with permissions to **describe and stop EC2 instances** (ec2:DescribeRegions, ec2:DescribeInstances, ec2:StopInstances).

The screenshot shows the AWS IAM Policy Editor interface. At the top, it says "Specify permissions" with an "Info" link. Below that, a note says "Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor." The main area is titled "Policy editor" and has tabs for "Visual", "JSON", and "Actions". The "Actions" tab is selected, showing a list under "EC2" with "Allow" selected for 3 Actions. A note says "Specify what actions can be performed on specific resources in EC2." Below this, "Actions allowed" is expanded, showing a search bar "Filter Actions" and an "Effect" section with "Allow" checked and "Deny" unselected. A "Manual actions" link and an "Add actions" button are also present. At the bottom, a green bar titled "Permissions defined in this policy" shows the JSON code for the policy:

```
1 {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Sid": "VisualEditor0",  
6             "Effect": "Allow",  
7             "Action": [  
8                 "ec2:DescribeInstances",  
9                 "ec2:DescribeInstanceAttribute",  
10                "ec2:DescribeRegions",  
11                "ec2:StopInstances"  
12            ],  
13            "Resource": "*"  
14        }  
15    ]  
16}
```

Review the permissions, specify details, and tags.

Policy details

Policy name

Enter a meaningful name to identify this policy.

ec2lambda

Maximum 128 characters. Use alphanumeric and '+=.,@-_' characters.

Description - optional

Add a short explanation for this policy.

Maximum 1,000 characters. Use alphanumeric and '+=.,@-_' characters.

2. Create IAM Role for Lambda

1. Go to **IAM → Roles → Create role**.
2. Choose **Lambda** as trusted entity.

or

?

external web identity provider to assume this role to perform actions in this account.	a corporate directory to perform actions in this account.
<input type="radio"/> 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.

3. Attach policy with:

- o ec2:DescribeRegions
- o ec2:DescribeInstances
- o ec2:StopInstances

Add permissions [Info](#)

Permissions policies (1/1083) [Info](#)

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

Filter by Type All types 1 match

<input checked="" type="checkbox"/> Policy name	Type	Description
<input checked="" type="checkbox"/> ec2lambda	Customer managed	-

▶ Set permissions boundary - *optional*

[Cancel](#) [Previous](#) [Next](#)

4. Name role: e.g. lambda-ec2-stop-role.
-

3. Create the Lambda Function

1. Open AWS Lambda Console → Create Function.
2. Choose:
 - Author from scratch
 - Runtime: **Python 3.9+**
 - Role: **Use existing role** → select lambda-ec2-stop-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-ec2-stop-role
[View the lambda-ec2-stop-role role](#) on the IAM console.

▶ Additional configurations
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

[Cancel](#) [Create function](#)

3. [Redacted]
 4. Name function: e.g., StopAllRunningEC2s.
-

4. Add the Python Code

Paste this into the Lambda inline editor:

```
import boto3

def lambda_handler(event, context):
```

```

ec2_client = boto3.client('ec2')

regions = [r['RegionName'] for r in ec2_client.describe_regions()['Regions']]


stopped = []

for region in regions:

    ec2 = boto3.resource('ec2', region_name=region)

    running = ec2.instances.filter(Filters=[{
        'Name': 'instance-state-name', 'Values': ['running']
    }])

    for i in running:

        i.stop()

        stopped.append({'Region': region, 'InstanceId': i.id})

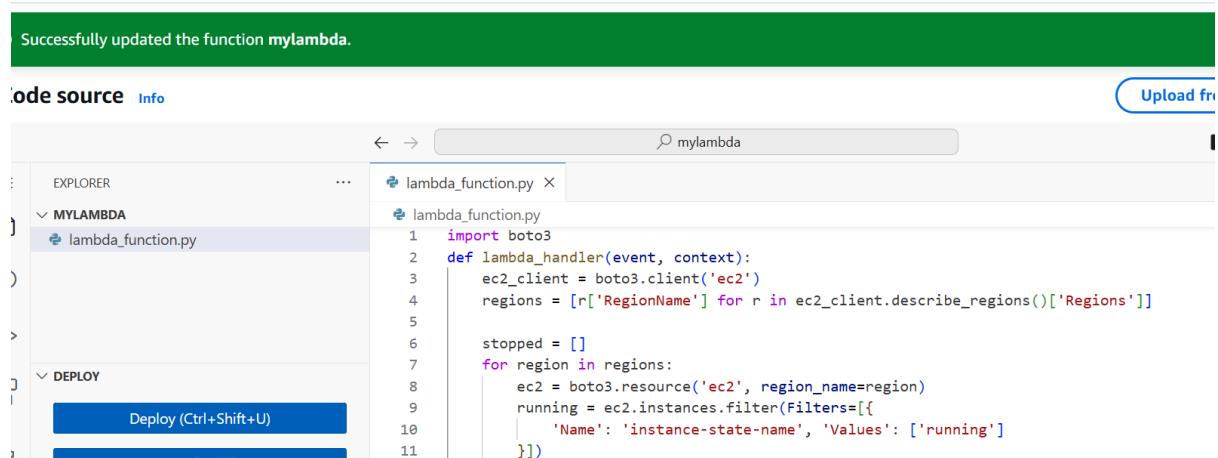
        print(f"Stopped {i.id} in {region}")


return {'StoppedInstances': stopped}

```

5. Deploy the Code

- Click **Deploy** in Lambda console to save the function.



Successfully updated the function mylambda.

Code source [Info](#) [Upload file](#)

EXPLORER **MYLAMBDA** **lambda_function.py**

```

import boto3
def lambda_handler(event, context):
    ec2_client = boto3.client('ec2')
    regions = [r['RegionName'] for r in ec2_client.describe_regions()['Regions']]

    stopped = []
    for region in regions:
        ec2 = boto3.resource('ec2', region_name=region)
        running = ec2.instances.filter(Filters=[{
            'Name': 'instance-state-name', 'Values': ['running']
        }])
        for i in running:
            i.stop()
            stopped.append({'Region': region, 'InstanceId': i.id})
            print(f"Stopped {i.id} in {region}")

    return {'StoppedInstances': stopped}

```

DEPLOY [Deploy \(Ctrl+Shift+U\)](#) [Test \(Ctrl+Shift+I\)](#)

6. Test the Function

- Click **Test → Create Event**.

2. Use a simple template like:

```
{}
```

3. Run the test.
 4. Check logs in **CloudWatch** for output like "Stopped i-0123456789abcdef0 in us-east-1".
-

7. Inspect Results

- Navigate to the EC2 console.
- Verify previously running instances are now in “stopping” or “stopped” states across regions.

The screenshot shows the AWS Lambda execution details page for a successful function execution. The top section displays the status as "Executing function: succeeded" with a link to "Logs". Below this is a "Details" section containing a JSON object with a single entry for a stopped instance in the US East-1 region. The "Summary" section includes the SHA-256 code hash and execution time (60 seconds ago). The execution ID is partially visible as 7fd... .

```
{  
  "StoppedInstances": [  
    {  
      "Region": "us-east-1",  
      "InstanceId": "i-07c371b212149af33"  
    }  
  ]  
}
```

Code SHA-256	Execution time
7fd...CDDA7... / CRV/EEhr020tW/GV/2fcCRHw...ZV-	60 seconds ago

8. Schedule Lambda Execution

(Optional: automate periodic shutdowns)

1. Go to **Lambda** → **Configuration** → **Triggers** → **Add trigger**.
 2. Choose **EventBridge (CloudWatch Events)**.
 3. Create rule:
 - Type: **Schedule expression**
 - Expression: `cron(0 22 * * ? *)` → e.g., 10 PM UTC daily.
 4. Click **Add**.
-