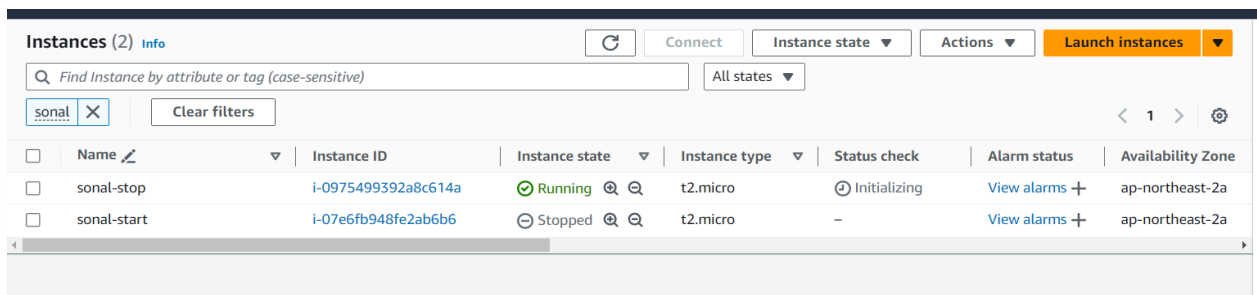


## Assignment 1: Automated Instance Management Using AWS Lambda and Boto3

**Objective:** In this assignment, you will gain hands-on experience with AWS Lambda and Boto3, Amazon's SDK for Python. You will create a Lambda function that will automatically manage EC2 instances based on their tags.

### Step 1: EC2 Setup

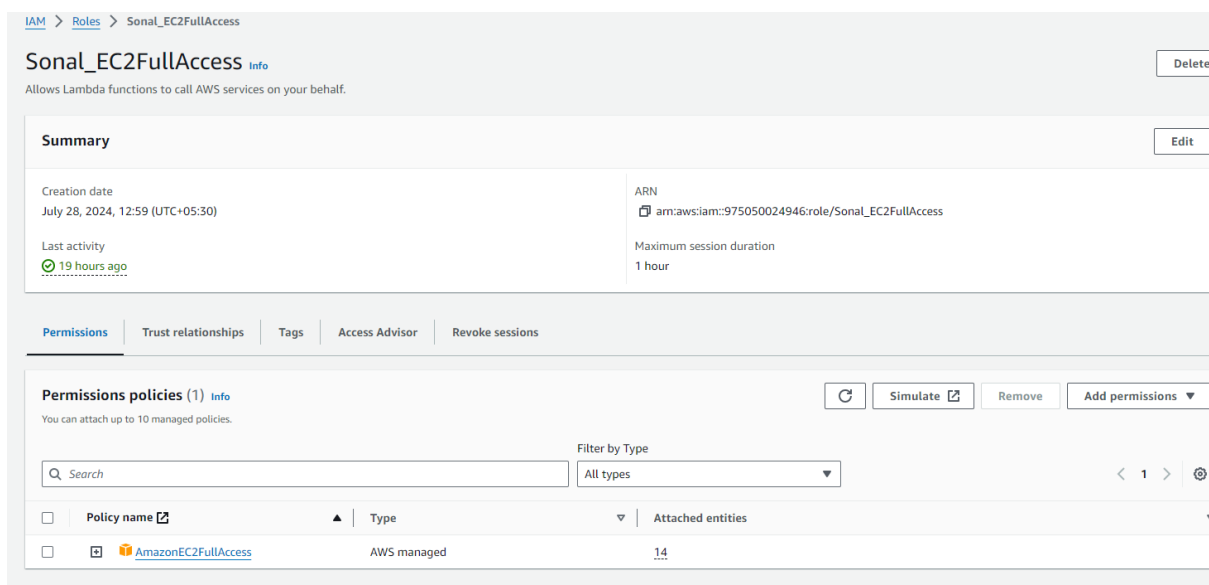
- Navigate to the EC2 Dashboard.
- Create two instances with t2. micro instance type
- Add tags:  
For the first instance: Key = Action, Value = Auto-Stop  
For the second instance: Key = Action, Value = Auto-Start
- Review and Launch the instances.



Instances (2) <a href="#">Info</a>							
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/> <span>All states ▾</span>							
<span>sonal</span> <span>✕</span> <span>Clear filters</span> <span>&lt; 1 &gt; ⚙</span>							
<input type="checkbox"/>	Name <a href="#">✎</a>	Instance ID	Instance state <a href="#">▾</a>	Instance type <a href="#">▾</a>	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	sonal-stop	i-0975499392a8c614a	<span>Running</span> <a href="#">🔍</a> <a href="#">🔍</a>	t2.micro	<span>🕒</span> Initializing	<a href="#">View alarms</a> <a href="#">+</a>	ap-northeast-2a
<input type="checkbox"/>	sonal-start	i-07e6fb948fe2ab6b6	<span>Stopped</span> <a href="#">🔍</a> <a href="#">🔍</a>	t2.micro	-	<a href="#">View alarms</a> <a href="#">+</a>	ap-northeast-2a

### Step 2: Lambda IAM Role

- Navigate to the IAM Dashboard and create new role for Lambda
- Attach the AmazonEC2FullAccess policy to this role.



IAM > Roles > Sonal_EC2FullAccess		<a href="#">Delete</a>
<b>Sonal_EC2FullAccess</b> <a href="#">Info</a>		
Allows Lambda functions to call AWS services on your behalf.		
<b>Summary</b> <a href="#">Edit</a>		
Creation date July 28, 2024, 12:59 (UTC+05:30)	ARN <a href="#">🔗</a> arn:aws:iam::975050024946:role/Sonal_EC2FullAccess	
Last activity <span>🟢</span> 19 hours ago	Maximum session duration 1 hour	
<a href="#">Permissions</a>   <a href="#">Trust relationships</a>   <a href="#">Tags</a>   <a href="#">Access Advisor</a>   <a href="#">Revoke sessions</a>		
<b>Permissions policies (1)</b> <a href="#">Info</a> <span>🔄</span> <a href="#">Simulate</a> <a href="#">Remove</a> <a href="#">Add permissions</a> <a href="#">▾</a>		
You can attach up to 10 managed policies.		
<input type="text" value="Search"/> <span>Filter by Type</span> <span>All types ▾</span> <span>&lt; 1 &gt; ⚙</span>		
<input type="checkbox"/>	Policy name <a href="#">🔗</a>	Attached entities
<input type="checkbox"/>	<a href="#">AmazonEC2FullAccess</a>	AWS managed 14

### Step 3: Lambda Function Creation:

1. **Create Lambda Function:**
  - Navigate to the Lambda Dashboard.

- Click "Create function."
- Select "Author from scratch."
- Name the function
- Choose Python 3.12 as the runtime.
- Under "Permissions," choose "Use an existing role."
- Select the role created in the previous step
- Click "Create function."

Lambda > Functions

Functions (3) Last fetched 2 minutes ago Actions Create function

Q  Filter by tags and attributes or search by keyword Matches: 1

sonal X Clear filters < 1 > ⚙

<input type="checkbox"/>	Function name	Description	Package type	Runtime	Last modified
<input type="checkbox"/>	<a href="#">Sonal_MangageEC2Instance</a>	-	Zip	Python 3.12	19 hours ago

## 2. Write Boto3 Script:

- Initialize a boto3 EC2 client
- Describe instances with 'Auto start' and 'Auto stop' tags
- Detect all EC2 instances with the 'Auto-Stop' tag and stop them
- Detect all EC2 instances with the 'Auto-Start' tag and start them.

Lambda > Functions > Sonal\_MangageEC2Instance

Sonal\_MangageEC2Instance Throttle Copy ARN Actions

Function overview Info Export to Application Composer Download

Diagram Template

+ Add trigger + Add destination

**Sonal\_MangageEC2Instance**

Layers (0)

Description  
-

Last modified  
19 hours ago

Function ARN  
[arn:aws:lambda:ap-northeast-2:975050024946:function:Sonal\\_MangageEC2Instance](#)

Function URL [Info](#)  
-

Code source Info Upload from

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

- Sonal\_MangageEC2
  - lambda\_function.py

```

1 import boto3
2
3 def stop_instance(instance_id):
4     # Create an EC2 client
5     ec2 = boto3.client('ec2')
6
7     # Stop the instance
8     response = ec2.stop_instances(InstanceIds=[instance_id])
9
10    http_code = response['ResponseMetadata']['HTTPStatusCode']
11
12    # Check the response
13    for instance in response['StoppingInstances']:
14        if instance['InstanceId'] == instance_id:
15            current_state = instance['CurrentState']['Name']
16            previous_state = instance['PreviousState']['Name']
17            print(f"Instance {instance_id} state changed from {previous_state} to {current_state}")
18
19            if http_code == 200 and (current_state == 'stopping' or current_state == 'stopped'):
20                return True, http_code
21            else:
22                return False, http_code
23
24 def start_instance(instance_id):
25     # Create an EC2 client
26     ec2 = boto3.client('ec2')
27
28     # Start the instance
29     response = ec2.start_instances(InstanceIds=[instance_id])
30
31    http_code = response['ResponseMetadata']['HTTPStatusCode']
32
33    # Check the response
34    for instance in response['StartingInstances']:
35        if instance['InstanceId'] == instance_id:
36            current_state = instance['CurrentState']['Name']
37            previous_state = instance['PreviousState']['Name']
38            print(f"Instance {instance_id} state changed from {previous_state} to {current_state}")
39
40

```

1:1 Python Spaces: 4

## Python script

```
Assignment_1 > Automated_InstanceManagement.py > stop_instance
1  import boto3
2
3
4  def stop_instance(instance_id):
5      # Create an EC2 client
6      ec2 = boto3.client('ec2')
7
8      # Stop the instance
9      response = ec2.stop_instances(InstanceIds=[instance_id])
10
11     http_code = response['ResponseMetadata']['HTTPStatusCode']
12
13     # Check the response
14     for instance in response['StoppingInstances']:
15         if instance['InstanceId'] == instance_id:
16             current_state = instance['CurrentState']['Name']
17             previous_state = instance['PreviousState']['Name']
18             print(f"Instance {instance_id} state changed from {previous_state} to {current_state}")
19
20             if http_code == 200 and (current_state == 'stopping' or current_state == 'stopped'):
21                 return True, http_code
22             else:
23                 return False, http_code
24
25 def start_instance(instance_id):
26     # Create an EC2 client
27     ec2 = boto3.client('ec2')
28
29     # Start the instance
30     response = ec2.start_instances(InstanceIds=[instance_id])
31
32     http_code = response['ResponseMetadata']['HTTPStatusCode']
33
34     print(response)
35
36     # Check the response
37     for instance in response['StartingInstances']:
38         if instance['InstanceId'] == instance_id:
39             current_state = instance['CurrentState']['Name']
40             previous_state = instance['PreviousState']['Name']
```

### 3. Save the function:

- Click on "Deploy" to save the changes.

## Step 4: Manual Execution

### 1. Execute the Lambda Function:

- Navigate to the Lambda function dashboard.
- Click on "Test."
- Configure a test event (you can use the default test event template).
- Click on "Create."
- Click on "Test" again to manually execute the function.

## 2. Verify EC2 Instance State:

- Navigate to the EC2 Dashboard.
- Check the state of the instances.
- The instance tagged Auto-Stop should be in the "stopping" or "stopped" state.
- The instance tagged Auto-Start should be in the "pending" or "running" state.

The screenshot shows the AWS Lambda console's 'Execution results' page for the function 'sonal\_event'. The status is 'Succeeded' and the max memory used is 89 MB. The response is a JSON object: `{ "statusCode": 200, "body": "Function executed successfully." }`. The function logs show a sequence of events: starting an instance (i-0975499392a8c614a), stopping it, starting another (i-037da9df2f689e42a), stopping it, starting a third (i-07e6fb948fe2ab6b6), and finally stopping it. The logs also include request IDs, version (\$LATEST), and duration (6756.52 ms). The request ID is 7ed01821-be2c-4ae8-9496-6b9d6f43ab66.

The screenshot shows the AWS EC2 console's 'Instances' page. There are two instances listed: 'sonal-stop' and 'sonal-start'. The 'sonal-stop' instance is in the 'Stopped' state, and the 'sonal-start' instance is in the 'Running' state. The table includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
sonal-stop	i-0975499392a8c614a	Stopped	t2.micro	-	View alarms +	ap-northeast-2a	-
sonal-start	i-07e6fb948fe2ab6b6	Running	t2.micro	Initializing	View alarms +	ap-northeast-2a	ec2-43-203-248-80