

Assignment 5: Auto-Tagging EC2 Instances on Launch Using AWS Lambda and Boto3

Objective: Learn to automate the tagging of EC2 instances as soon as they are launched, ensuring better resource tracking and management.

Step 1. EC2 Setup

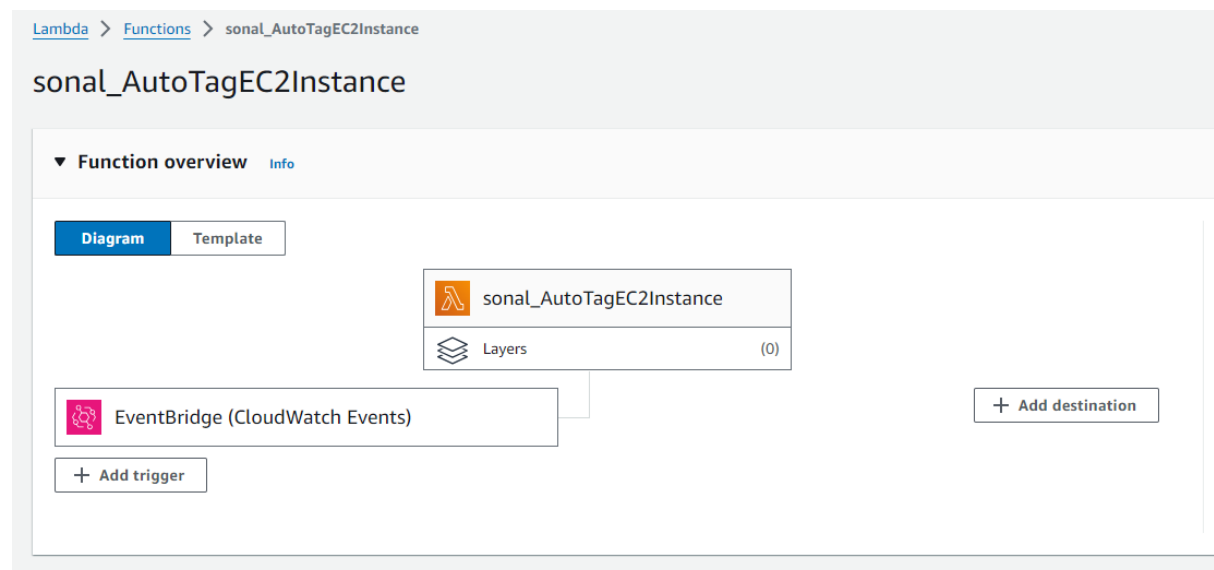
Ensure you have the capability to launch EC2 instances.

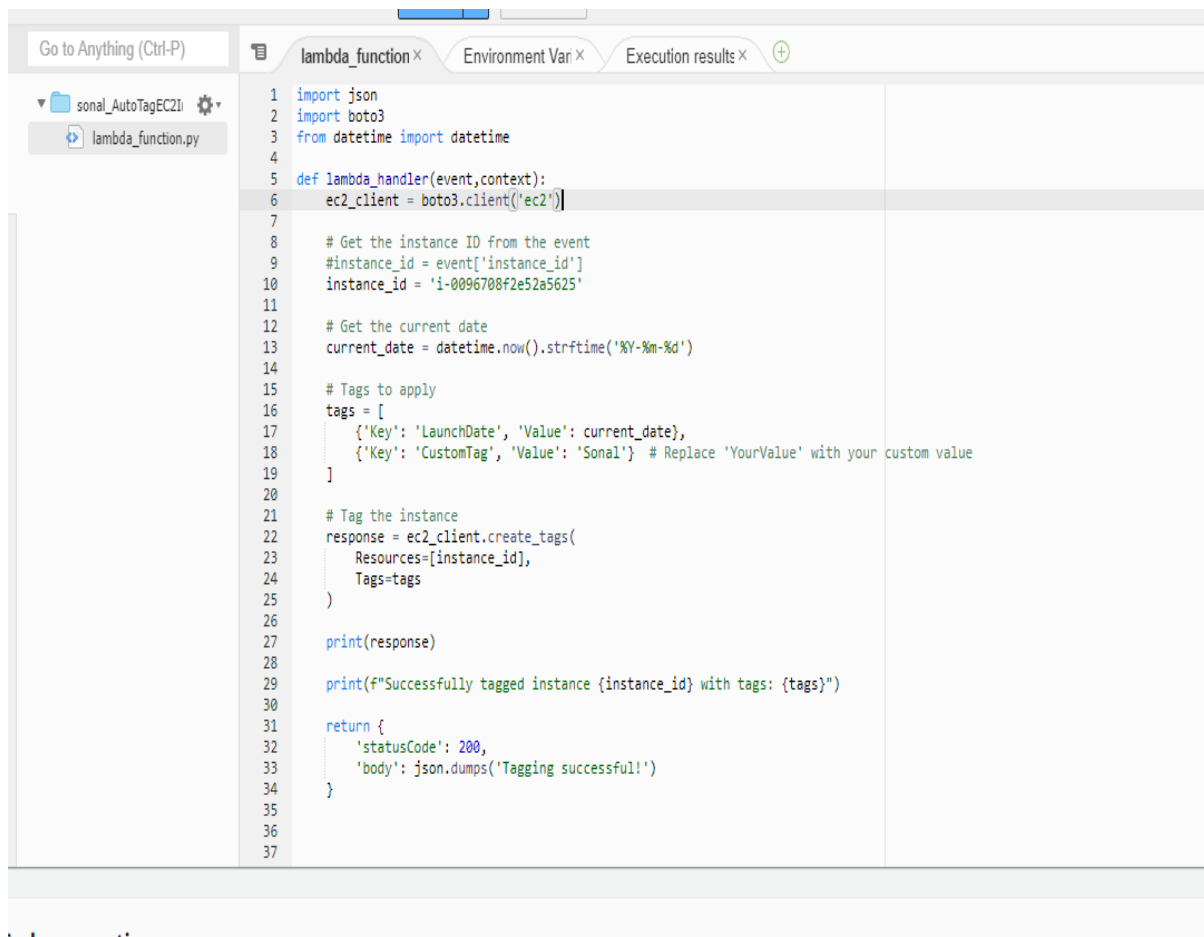
Step 2. Create Lambda IAM Role

- Attach the AmazonEC2FullAccess policy and create role “sonal_AutoTagEC2Instance”

Step 3. Create Lambda Function

- Go to the Lambda dashboard.
- Click **Create function**.
- Choose Python 3.11 as the runtime.
- Under **Permissions**, choose **Use an existing role** and select the IAM role created in the previous step.
- Click **Create function**.





The screenshot shows an IDE window with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'sonal_AutoTagEC2I' containing a file 'lambda_function.py'. The code editor has three tabs: 'lambda_function', 'Environment Vari', and 'Execution results'. The 'lambda_function' tab is active, displaying a Python script. The script imports 'json' and 'boto3', and 'datetime' from the 'datetime' module. It defines a 'lambda_handler' function that takes 'event' and 'context' as arguments. Inside the function, it creates an 'ec2_client' using 'boto3.client("ec2")'. It then extracts the 'instance_id' from the 'event' dictionary. A comment indicates that the 'instance_id' should be replaced with a custom value. The script then gets the current date using 'datetime.now().strftime("%Y-%m-%d")'. It creates a list of tags, including 'LaunchDate' with the current date and 'CustomTag' with the value 'Sonal'. A comment indicates that the 'YourValue' should be replaced with a custom value. The script then calls 'ec2_client.create_tags' with the 'instance_id' and the list of tags. It prints the response and a confirmation message. Finally, it returns a dictionary with 'statusCode' 200 and a 'body' containing 'Tagging successful!'.

```
1 import json
2 import boto3
3 from datetime import datetime
4
5 def lambda_handler(event, context):
6     ec2_client = boto3.client('ec2')
7
8     # Get the instance ID from the event
9     #instance_id = event['instance_id']
10    instance_id = 'i-0096708f2e52a5625'
11
12    # Get the current date
13    current_date = datetime.now().strftime('%Y-%m-%d')
14
15    # Tags to apply
16    tags = [
17        {'Key': 'LaunchDate', 'Value': current_date},
18        {'Key': 'CustomTag', 'Value': 'Sonal'} # Replace 'YourValue' with your custom value
19    ]
20
21    # Tag the instance
22    response = ec2_client.create_tags(
23        Resources=[instance_id],
24        Tags=tags
25    )
26
27    print(response)
28
29    print(f"Successfully tagged instance {instance_id} with tags: {tags}")
30
31    return {
32        'statusCode': 200,
33        'body': json.dumps('Tagging successful!')
34    }
35
36
37
```

Step 4. Write the Boto3 Python script to:

- Initialize a boto3 EC2 client.
- Retrieve the instance ID from the event.
- Tag the new instance with the current date and another tag of your choice.
- Print a confirmation message for logging purposes.

```
Autotag.py X
Autotag.py > ...
1  import json
2  import boto3
3  from datetime import datetime
4
5  def lambda_handler(event, context):
6      ec2_client = boto3.client('ec2')
7
8      # Get the instance ID from the event
9      #instance_id = event['instance_id']
10     instance_id = 'i-0096708f2e52a5625'
11
12     # Get the current date
13     current_date = datetime.now().strftime('%Y-%m-%d')
14
15     # Tags to apply
16     tags = [
17         {'Key': 'LaunchDate', 'Value': current_date},
18         {'Key': 'CustomTag', 'Value': 'YourValue'} # Replace 'YourValue' with your custom value
19     ]
20
21     # Tag the instance
22     response = ec2_client.create_tags(
23         Resources=[instance_id],
24         Tags=tags
25     )
26
27     print(response)
28
29     print(f"Successfully tagged instance {instance_id} with tags: {tags}")
30
31     return {
32         'statusCode': 200,
33         'body': json.dumps('Tagging successful!')
34     }
35
36
37
```

Step 5. Set Up CloudWatch Event Rule

- Set up a CloudWatch Event Rule to trigger the EC2 instance launch event.
- Attach the Lambda function as the target.

Amazon EventBridge > Rules > sonal-rule

sonal-rule

Edit Disable Delete CloudFormation Template ▼

Rule details Info

Rule name sonal-rule	Status Enabled	Event bus name default	Type Scheduled Standard
Description	Rule ARN arn:aws:events:ap-northeast-2:97505024946:rule/sonal-rule	Event bus ARN arn:aws:events:ap-northeast-2:97505024946:event-bus/default	

Event schedule Targets Monitoring Tags

Targets Edit

Details	Target Name	Type	Arn	Input	Role
▼	sonal_AutoTagEC2Instance ↗	Lambda function	arn:aws:lambda:ap-northeast-2:975050024946:function:sonal_AutoTagEC2Instance	Matched event	-

Input to target: Matched event

Amazon EventBridge > Rules > sonal-rule

sonal-rule

Edit Disable Delete CloudFormation Template ▼

Rule details Info

Rule name sonal-rule	Status Enabled	Event bus name default	Type Scheduled Standard
Description	Rule ARN arn:aws:events:ap-northeast-2:975050024946:rule/sonal-rule	Event bus ARN arn:aws:events:ap-northeast-2:975050024946:event-bus/default	

Event schedule Targets Monitoring Tags

Event schedule Info Edit

Cron expression
0/1 * * * * *

Next 10 trigger date(s) Local time zone ▼

Wed, Jul 31, 2024, 09:39 PM GMT+5:30
Wed, Jul 31, 2024, 09:40 PM GMT+5:30
Wed, Jul 31, 2024, 09:41 PM GMT+5:30
Wed, Jul 31, 2024, 09:42 PM GMT+5:30
Wed, Jul 31, 2024, 09:43 PM GMT+5:30
Wed, Jul 31, 2024, 09:44 PM GMT+5:30
Wed, Jul 31, 2024, 09:45 PM GMT+5:30
Wed, Jul 31, 2024, 09:46 PM GMT+5:30
Wed, Jul 31, 2024, 09:47 PM GMT+5:30
Wed, Jul 31, 2024, 09:48 PM GMT+5:30

Step 6. Testing

- Launch a new EC2 instance.
- After a short delay, go to the EC2 dashboard.
- Check the tags of the newly launched instance to confirm they have been automatically applied.

Instances (1/1) [Info](#)

Find Instance by attribute or tag (case-sensitive)

All states ▾

Name = Sonal-Autotag ✕

Clear filters

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	Public IPv4 D
<input checked="" type="checkbox"/>	Sonal-Autotag	i-0096708f2e52a5625	Running	t2.micro	2/2 checks passed	View alarms +	ap-northeast-2a	ec2-3-35-236

The screenshot shows the AWS Lambda console interface. At the top, there are tabs for 'Test', 'Environment Variables', and 'Execution results'. The 'Execution results' tab is selected, showing a status of 'Succeeded'. The response is a JSON object with 'statusCode': 200 and 'body': '"Tagging successful!\\\"'. The function logs show the request ID, version (SLATEST), and the successful tagging of the instance with tags.

Log Events

Tags	
<input type="text"/>	
Key	Value
Name	Sonal-Autotag
CustomTag	Sonal
LaunchDate	2024-07-31