Graded Assignment On Serverless Architecture

Assignment 1: Automated Instance Management Using AWS Lambda and Boto3

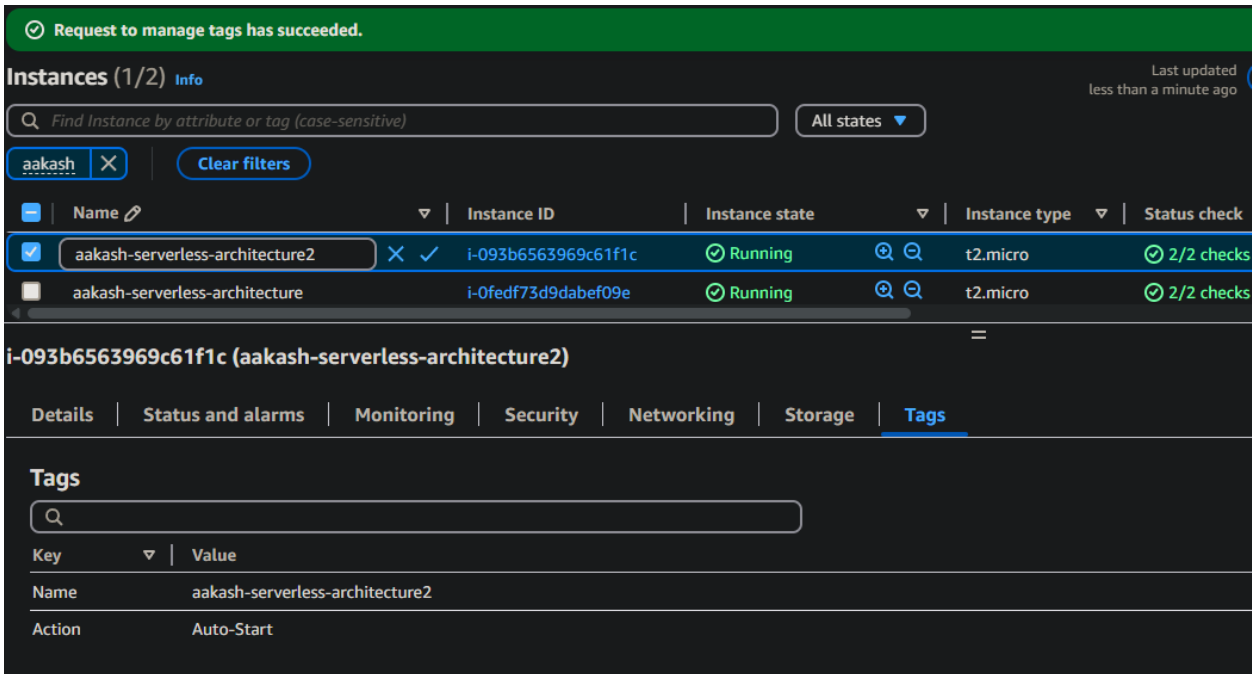
Objective: In this assignment, you will gain hands-on experience with AWS Lambda and Boto3, Amazon's SDK for Python. You will create a Lambda function that will automatically manage EC2 instances based on their tags.

Task: You're tasked to automate the stopping and starting of EC2 instances based on tags. Specifically:

1. Setup:

- Create two EC2 instances.

- Tag one of them as `Auto-Stop` and the other as `Auto-Start`.



Request to manage tags has succeeded.

Instances (1/2) Info

Last updated less than a minute ago

Name	Instance ID	Instance state	Instance type	Status check
aakash-serverless-architecture2	i-093b6563969c61f1c	Running	t2.micro	2/2 checks
aakash-serverless-architecture	i-0fedf73d9dabef09e	Running	t2.micro	2/2 checks

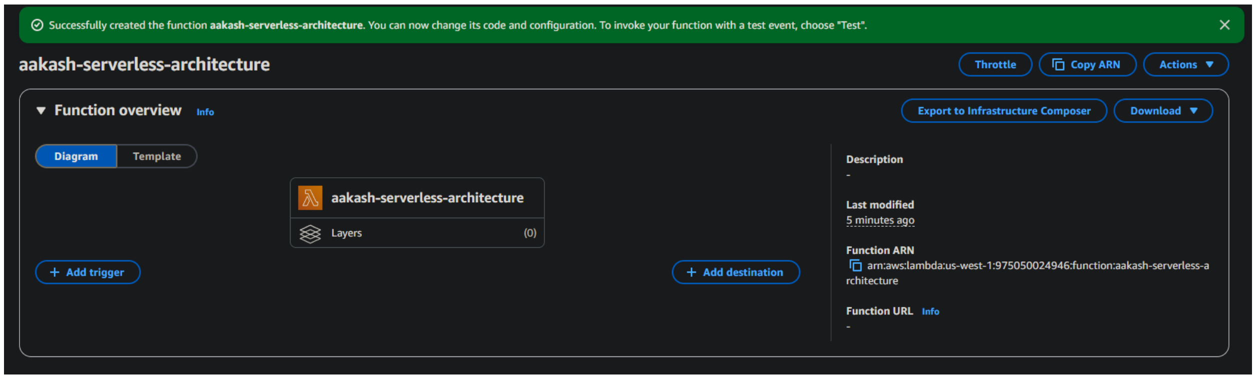
i-093b6563969c61f1c (aakash-serverless-architecture2)

Tags

Key	Value
Name	aakash-serverless-architecture2
Action	Auto-Start

2. Lambda Function Creation:

- Set up an AWS Lambda function.


Successfully created the function **aakash-serverless-architecture**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

aakash-serverless-architecture

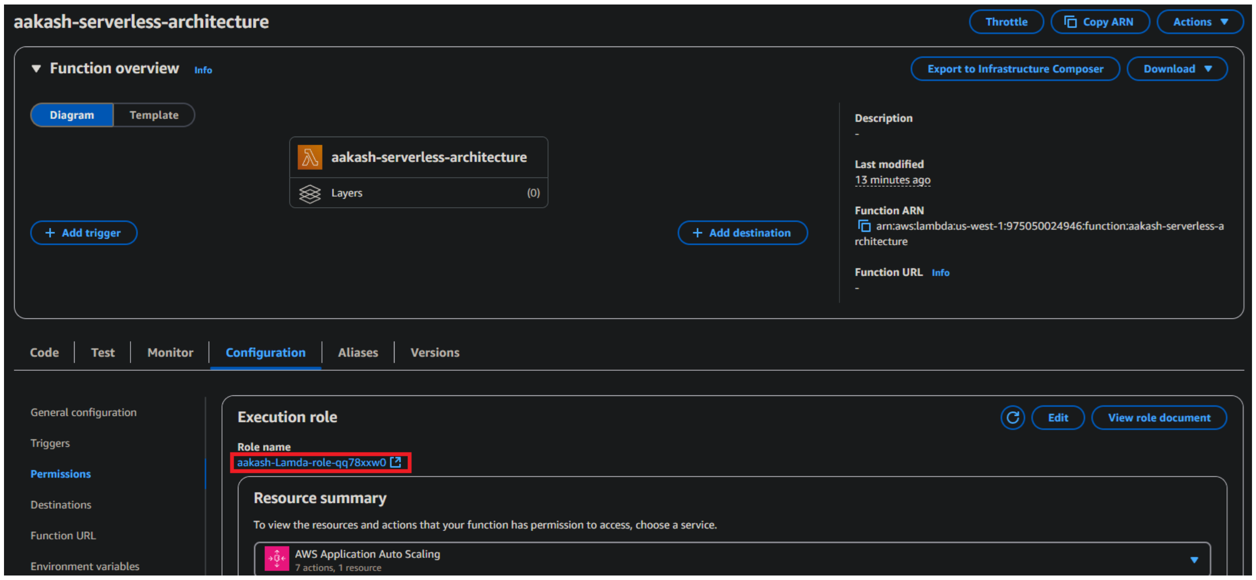
Description
-

Last modified
5 minutes ago

Function ARN
arn:aws:lambda:us-west-1:975050024946:function:aakash-serverless-architecture

Function URL Info
-

- Ensure that the Lambda function has the necessary IAM permissions to describe, stop, and start EC2 instances.


aakash-serverless-architecture

Description
-

Last modified
13 minutes ago

Function ARN
arn:aws:lambda:us-west-1:975050024946:function:aakash-serverless-architecture

Function URL Info
-

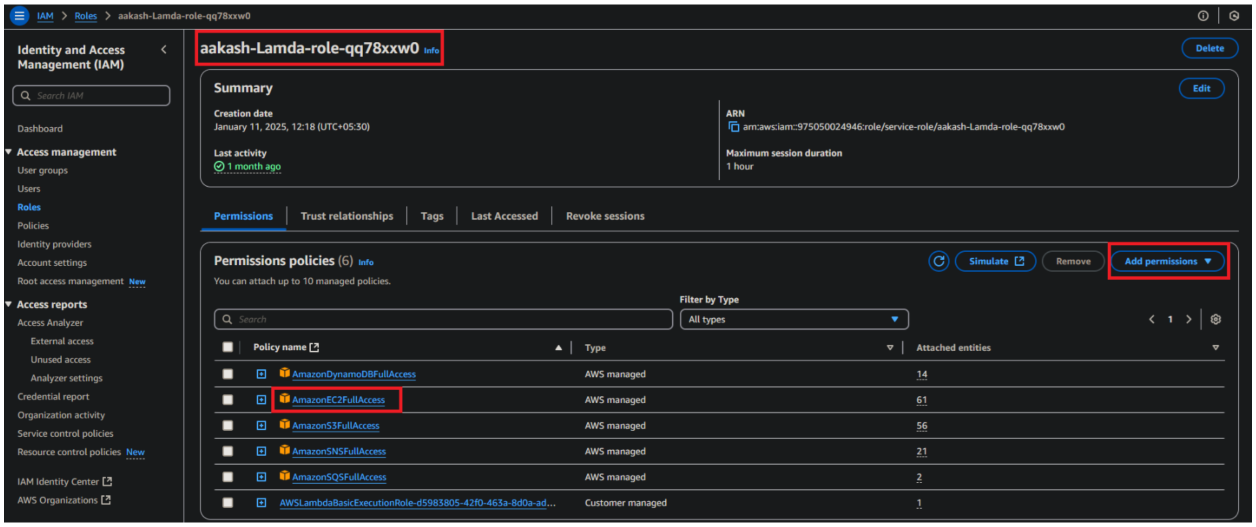
Execution role

Role name
aakash-Lamda-role-qq78xxw0

Resource summary

To view the resources and actions that your function has permission to access, choose a service.

AWS Application Auto Scaling
7 actions, 1 resource


IAM > Roles > aakash-Lamda-role-qq78xxw0

aakash-Lamda-role-qq78xxw0 Info

Summary

Creation date
January 11, 2025, 12:18 (UTC+05:30)

ARN
arn:aws:iam::975050024946:role/service-role/aakash-Lamda-role-qq78xxw0

Last activity
1 month ago

Maximum session duration
1 hour

Permissions policies (6) Info

You can attach up to 10 managed policies.

Filter by Type

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	14
AmazonEC2FullAccess	AWS managed	61
AmazonS3FullAccess	AWS managed	56
AmazonSNSFullAccess	AWS managed	21
AmazonSQSFullAccess	AWS managed	2
AWSLambdaBasicExecutionRole-d5983805-42f0-463a-8d0a-ad...	Customer managed	1

3. Coding:

- Using Boto3 in the Lambda function:
  - Detect all EC2 instances with the `Auto-Stop` tag and stop them.
  - Detect all EC2 instances with the `Auto-Start` tag and start them.

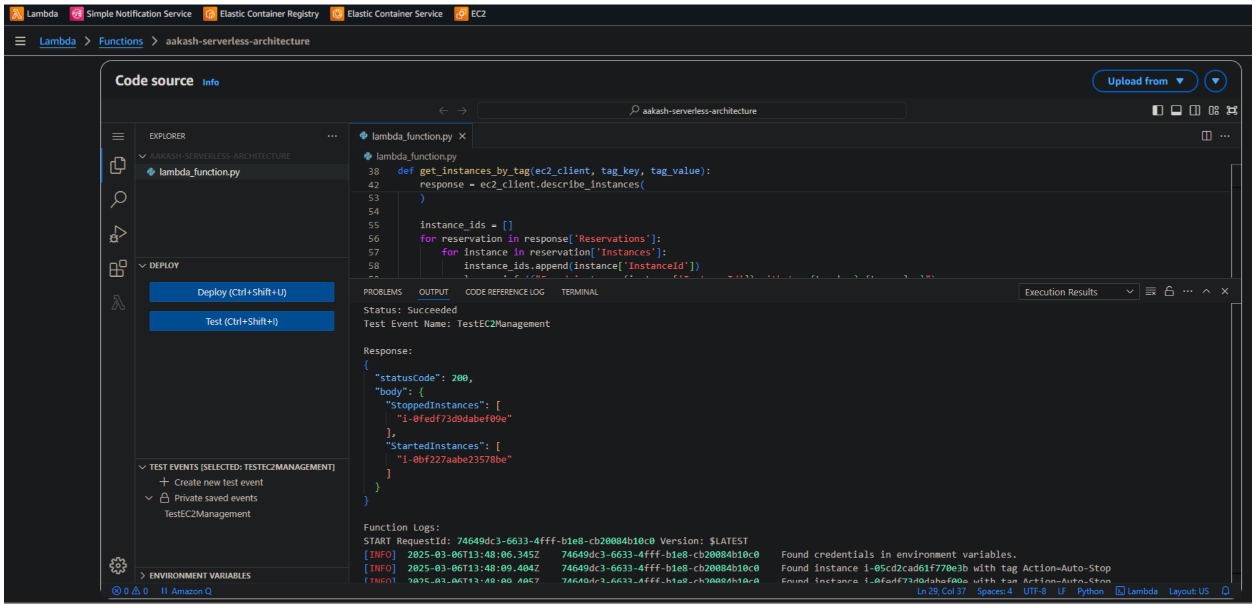
T1.py

```python
import boto3
import logging

# Configure logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    """
    AWS Lambda function to manage EC2 instances based on tags:
    - Stops instances tagged with Action=Auto-Stop
    - Starts instances tagged with Action=Auto-Start
    """
    # Initialize boto3 EC2 client
    ec2_client = boto3.client('ec2')

    # Find instances to stop (Action=Auto-Stop)
    instances_to_stop = get_instances_by_tag(ec2_client, 'Action', 'Auto-Stop')

    # Find instances to start (Action=Auto-Start)
    instances_to_start = get_instances_by_tag(ec2_client, 'Action', 'Auto-Start')

    # Stop instances
    stop_result = stop_instances(ec2_client, instances_to_stop)

    # Start instances
    start_result = start_instances(ec2_client, instances_to_start)

    # Prepare and return the results
    return {
        'statusCode': 200,
        'body': {
            'StoppedInstances': stop_result,
            'StartedInstances': start_result
        }
    }

def get_instances_by_tag(ec2_client, tag_key, tag_value):
    """
    Find EC2 instances that have a specific tag key and value
    """
    response = ec2_client.describe_instances(
        Filters=[
            {
                'Name': f'tag:{tag_key}',
                'Values': [tag_value]
            },
            {
                'Name': 'instance-state-name',
                'Values': ['pending', 'running', 'stopping', 'stopped']
            }
        ]
    )

    instance_ids = []
    for reservation in response['Reservations']:
        for instance in reservation['Instances']:
            instance_ids.append(instance['InstanceId'])
            logger.info(f"Found instance {instance['InstanceId']} with tag 
{tag_key}={tag_value}")

    return instance_ids

def stop_instances(ec2_client, instance_ids):
    """
    Stop the specified EC2 instances if they are in 'running' state
    """
    if not instance_ids:
        logger.info("No instances to stop")
        return []

    # First, check which instances are running
    response = ec2_client.describe_instances(
        InstanceIds=instance_ids,
        Filters=[
            {
                'Name': 'instance-state-name',
                'Values': ['running']
            }
```

```python
        ]
    )

    running_instances = []
    for reservation in response['Reservations']:
        for instance in reservation['Instances']:
            running_instances.append(instance['InstanceId'])

    # Stop only running instances
    if running_instances:
        logger.info(f"Stopping instances: {running_instances}")
        ec2_client.stop_instances(InstanceIds=running_instances)
        return running_instances
    else:
        logger.info("No running instances to stop")
        return []

def start_instances(ec2_client, instance_ids):
    """
    Start the specified EC2 instances if they are in 'stopped' state
    """
    if not instance_ids:
        logger.info("No instances to start")
        return []

    # First, check which instances are stopped
    response = ec2_client.describe_instances(
        InstanceIds=instance_ids,
        Filters=[
            {
                'Name': 'instance-state-name',
                'Values': ['stopped']
            }
        ]
    )

    stopped_instances = []
    for reservation in response['Reservations']:
        for instance in reservation['Instances']:
            stopped_instances.append(instance['InstanceId'])

    # Start only stopped instances
    if stopped_instances:
        logger.info(f"Starting instances: {stopped_instances}")
        ec2_client.start_instances(InstanceIds=stopped_instances)
        return stopped_instances
    else:
        logger.info("No stopped instances to start")
        return []
```

4. Testing:

   - Manually invoke the Lambda function.

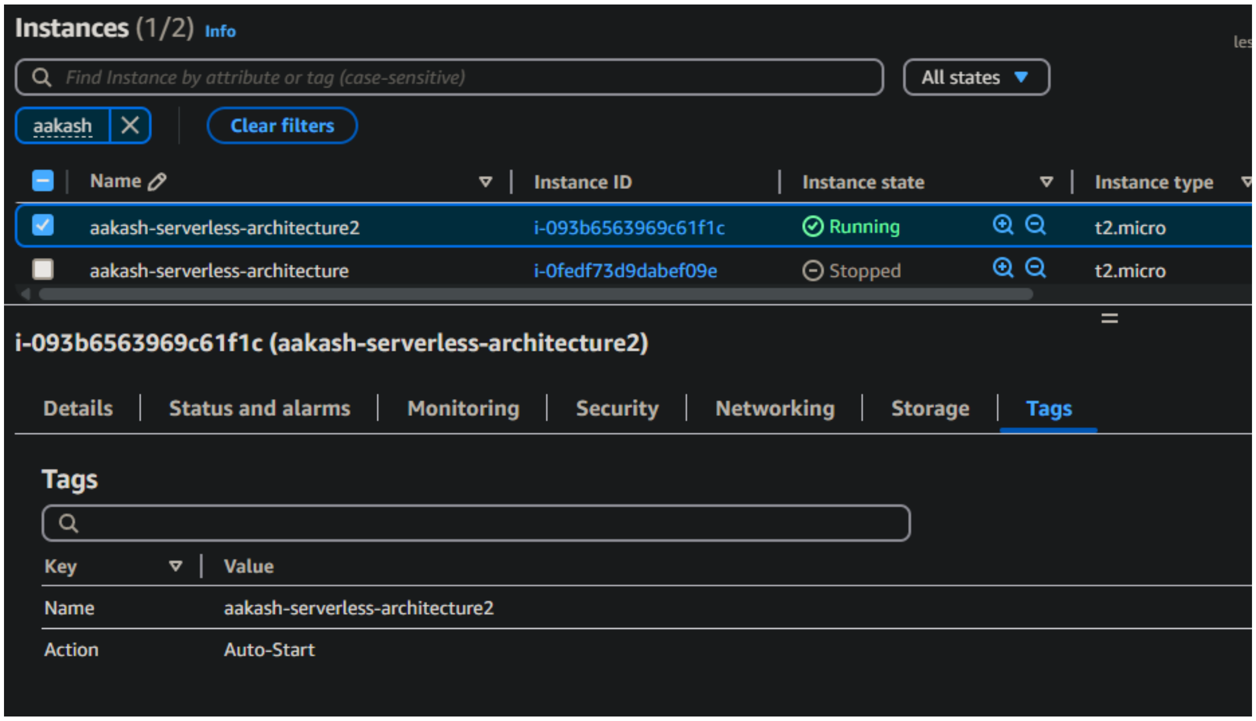


Lambda
Simple Notification Service
Elastic Container Registry
Elastic Container Service
EC2
Lambda > Functions > aakash-serverless-architecture
Code source Info
Upload from
aakash-serverless-architecture
EXPLORER
AAKASH-SERVERLESS-ARCHITECTURE
lambda_function.py
DEPLOY
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [SELECTED: TESTEC2MANAGEMENT]
Create new test event
Private saved events
TestEC2Management
ENVIRONMENT VARIABLES
PROBLEMS
OUTPUT
CODE REFERENCE LOG
TERMINAL
Execution Results
Ln 29, Col 37 Spaces: 4 UTF-8 LF Python Lambda Layout: US

```python
def get_instances_by_tag(ec2_client, tag_key, tag_value):
    response = ec2_client.describe_instances(
    )

    instance_ids = []
    for reservation in response['Reservations']:
        for instance in reservation['Instances']:
            instance_ids.append(instance['InstanceId'])
```

```
Status: Succeeded
Test Event Name: TestEC2Management

Response:
{
  "statusCode": 200,
  "body": {
    "StoppedInstances": [
      "i-0fedf73d9dabef09e"
    ],
    "StartedInstances": [
      "i-0bf227aabe23578be"
    ]
  }
}

Function Logs:
START RequestId: 74649dc3-6633-4fff-b1e8-cb20084b10c0 Version: $LATEST
[INFO]  2025-03-06T13:48:06.345Z    74649dc3-6633-4fff-b1e8-cb20084b10c0    Found credentials in environment variables.
[INFO]  2025-03-06T13:48:09.404Z    74649dc3-6633-4fff-b1e8-cb20084b10c0    Found instance i-05cd2cad61f770e3b with tag Action=Auto-Stop
[INFO]  2025-03-06T13:48:09.405Z    74649dc3-6633-4fff-b1e8-cb20084b10c0    Found instance i-0fedf73d9dabef09e with tag Action=Auto-Stop
```

   - Confirm that the instance tagged `Auto-Stop` stops and the one tagged `Auto-Start` starts.



Name	Instance ID	Instance state	Instance type
aakash-serverless-architecture2	i-093b6563969c61f1c	Running	t2.micro
aakash-serverless-architecture	i-0fedf73d9dabef09e	Stopped	t2.micro

i-093b6563969c61f1c (aakash-serverless-architecture2)

Tags

Key	Value
Name	aakash-serverless-architecture2
Action	Auto-Start