

## 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` .

2. Lambda Function Creation:

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

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.

4. Testing:

- Manually invoke the Lambda function.
- Confirm that the instance tagged `Auto-Stop` stops and the one tagged `Auto-Start` starts.

**Instructions:**

1. EC2 Setup:

- Navigate to the EC2 dashboard and create two new t2.micro instances (or any other available free-tier type).
- Tag the first instance with a key `Action` and value `Auto-Stop` .
- Tag the second instance with a key `Action` and value `Auto-Start` .

2. Lambda IAM Role:

- In the IAM dashboard, create a new role for Lambda.
- Attach the `AmazonEC2FullAccess` policy to this role. (Note: In a real-world scenario, you would want to limit permissions for better security.)

3. Lambda Function:

- Navigate to the Lambda dashboard and create a new function.
- Choose Python 3.x as the runtime.
- Assign the IAM role created in the previous step.
- Write the Boto3 Python script to:
  1. Initialize a boto3 EC2 client.
  2. Describe instances with `Auto-Stop` and `Auto-Start` tags.
  3. Stop the `Auto-Stop` instances and start the `Auto-Start` instances.
  4. Print instance IDs that were affected for logging purposes.
- 4. Manual Invocation:
  - After saving your function, manually trigger it.
- Go to the EC2 dashboard and confirm that the instances' states have changed according to their tags.

The screenshot shows the AWS EC2 Instances page in a Firefox browser. The URL is https://us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#instances. The left sidebar shows navigation links like EC2 Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, Network & Security, and Key Pairs. The main content area displays a table of instances with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Public IP. Two instances are selected: "Vivek Auto-Stop" (instance ID i-0fb51c85613956f3b) and "Vivek Auto-Start" (instance ID i-09558316e709942a0). Both are listed as "Running". The status check for the Auto-Stop instance shows "2/2 checks passec", while for the Auto-Start instance it shows "3/3 checks passec". The alarm status for both is "+". The availability zone is "us-west-2d" for both. The public IP is listed as "ec2-35-84-180-20.us-west-2" for the Auto-Stop instance and "ec2-35-80-7-139.us-west-2" for the Auto-Start instance. A message at the bottom says "2 instances selected".

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
mandarAutoSt...	i-00c21767666df4000	Running	t2.nano	2/2 checks passec	+ View alarms	us-west-2b	ec2-54-188-43-1.us-west-2	54.188.43.1
vish-tm-server0	i-0014718132bf8be71	Running	t4g.micro	3/3 checks passec	+ View alarms	us-west-2d	ec2-35-80-21-8.us-west-2	35.80.21
Ram-Auto-Stop	i-0de80f7babc319838	Running	t4g.micro	3/3 checks passec	+ View alarms	us-west-2d	ec2-44-234-109-87.us-west-2	44.234.109.87
<input checked="" type="checkbox"/> Vivek Auto-Stop	i-0fb51c85613956f3b	Running	t4g.micro	2/2 checks passec	+ View alarms	us-west-2d	ec2-35-84-180-20.us-west-2	35.84.180.20
mandarAutoSt...	i-022e39d2bf934a40f	Running	t4g.nano	Initializing	+ View alarms	us-west-2d	ec2-35-86-192-75.us-west-2	35.86.192.75
<input checked="" type="checkbox"/> Vivek Auto-Start	i-09558316e709942a0	Running	t4g.micro	3/3 checks passec	+ View alarms	us-west-2d	ec2-35-80-7-139.us-west-2	35.80.7.1

Create a two EC2 Instances with the name Auto-Start and Auto-Stop and later stop the EC2 (Auto-start)

## Create an IAM role and the LAMDA Function and add the Full EC2 Access

The screenshot shows the 'Create role' wizard in the AWS IAM console. The current step is 'Step 1: Select trusted entity'. The 'AWS service' option is selected. Other options like 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy' are available. Below this, a 'Use case' section is shown with 'Lambda' selected. At the bottom right, there are 'Cancel' and 'Next' buttons.

The screenshot shows the 'VivekIAM' role details page in the AWS IAM console. A green banner at the top indicates that the 'AmazonEC2FullAccess' policy was successfully attached. The 'Summary' section shows the creation date (September 20, 2024, 20:25 UTC+05:30) and ARN (arn:aws:iam::975050024946:role/VivekIAM). The 'Permissions' tab is selected, showing one managed policy named 'AmazonEC2FullAccess'. The ARN for this policy is arn:aws:iam::975050024946:policy/AmazonEC2FullAccess. The ARN for the role itself is arn:aws:iam::975050024946:role/VivekIAM. At the bottom right, there are 'CloudShell' and 'Feedback' buttons.

Screenshot of the AWS Lambda function creation interface in Firefox.

**Basic information**

**Function name:** VivekAssignment

**Runtime:** Python 3.10

**Architecture:** arm64

**Permissions:** VivekJAM

**Execution role:** Use an existing role (VivekJAM)

**CloudShell:** Feedback

Screenshot of the AWS EC2 Instances page in Firefox, showing the successful stopping of an instance.

**Instances (1/1) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
Vivek Auto-Start	i-09558316e709942a0	Stopping	t4g.micro	Initializing	View alarms	us-west-2d	ec2-35-80-7-139.us-west-2.amazonaws.com	35.80.7.1

**CloudShell:** Feedback

The screenshot shows the AWS EC2 Instances page. A success message at the top says "Request to manage tags has succeeded." The main table displays two instances:

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
<input type="checkbox"/>	Vivek Auto-Stop	i-0fb51c85613956f3b	Running	t4g.micro	3/3 checks passed	<a href="#">View alarms</a> +	us-west-2d	ec2-35-84-180-20.us-west-2.amazonaws.com	35.84.180.20
<input type="checkbox"/>	Vivek Auto-Start	i-09558316e709942a0	Stopped	t4g.micro	-	<a href="#">View alarms</a> +	us-west-2d	-	-

The left sidebar includes sections for EC2 Dashboard, EC2 Global View, Events, Instances (with sub-options like Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs), and CloudShell.

Once done add the lambda function code and changes the tag which is provided earlier in the EC2 - Auto-Start and Auto Stop instances.

The code is provided in the GIT HUB - Assignment 1

Once done with the changes deploy and test the code.

The screenshot shows the AWS Lambda console interface. The top navigation bar includes tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The main area displays the 'Code source' tab with a code editor and a 'Test' button. Below the editor is a log viewer titled 'Execution results' which shows the following log output:

```
Function Log
START RequestId: 38f62a02-2105-45d3-8367-1fa321cb8b76 Version: $LATEST
Stopping instances: ["i-0fb51c85613956f3b"]
Starting instances: ["i-09558316e709942a0"]
END RequestId: 38f62a02-2105-45d3-8367-1fa321cb8b76
REPORT RequestId: 38f62a02-2105-45d3-8367-1fa321cb8b76 Duration: 1441.47 ms Billed Duration: 1442 ms Memory Size: 128 MB Max Memory Used: 86 MB Init Duration: 543.96 ms
Request ID
38f62a02-2105-45d3-8367-1fa321cb8b76
```

The screenshot shows the AWS EC2 Instances page. The left sidebar contains navigation links for EC2 Dashboard, EC2 Global View, Events, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs), and CloudWatch Metrics. The main content area is titled 'Instances (2) Info' and lists two instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
Vivek Auto-Stop	i-0fb51c85613956f3b	Stopping	t4g.micro	-	<a href="#">View alarms</a>	us-west-2d	ec2-35-84-180-20.us-w...	35.84.180...
Vivek Auto-Start	i-09558316e709942a0	Running	t4g.micro	Initializing	<a href="#">View alarms</a>	us-west-2d	ec2-44-234-9-154.us-w...	44.234.9.

## **Assignment 2: Automated S3 Bucket Cleanup Using AWS Lambda and Boto3**

**Objective:** To gain experience with AWS Lambda and Boto3 by creating a Lambda function that will automatically clean up old files in an S3 bucket.

**Task:** Automate the deletion of files older than 30 days in a specific S3 bucket.

### **Instructions:**

#### 1. S3 Setup:

- Navigate to the S3 dashboard and create a new bucket.
- Upload multiple files to this bucket, ensuring that some files are older than 30 days (you may need to adjust your system's date temporarily for this or use old files).

#### 2. Lambda IAM Role:

- In the IAM dashboard, create a new role for Lambda.
- Attach the `AmazonS3FullAccess` policy to this role. (Note: For enhanced security in real-world scenarios, use more restrictive permissions.)

#### 3. Lambda Function:

- Navigate to the Lambda dashboard and create a new function.
- Choose Python 3.x as the runtime.
- Assign the IAM role created in the previous step.
- Write the Boto3 Python script to:
  1. Initialize a boto3 S3 client.
  2. List objects in the specified bucket.
  3. Delete objects older than 30 days.
  4. Print the names of deleted objects for logging purposes.

#### 4. Manual Invocation:

- After saving your function, manually trigger it.
- Go to the S3 dashboard and confirm that only files newer than 30 days remain.

## Create a lambda function with the IAM Role with S3 full access

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Basic information' section is visible, showing the function name 'Viveks3', runtime 'Python 3.10', and architecture 'arm64'. The 'Permissions' section indicates that a default execution role will be created. At the bottom, there's a 'Change default execution role' link.

The screenshot shows the 'Add permissions' wizard in the AWS IAM console. It displays a list of 'Other permissions policies' under 'Current permissions policies'. A search bar shows 'amazonaws3'. The policy 'AmazonS3FullAccess' is selected and highlighted. At the bottom right, there are 'Cancel' and 'Add permissions' buttons.

## Now add the code with the S3 bucket name

The screenshot shows the AWS Lambda function editor for the 'Vivek3' function. The 'Code' tab is selected. The code editor contains the following Python script:

```
import boto3
from datetime import datetime, timezone, timedelta

# Initialize the S3 client
s3 = boto3.client('s3')

# Bucket name
BUCKET_NAME = 'your-bucket-name'

# Time delta for 30 days
DAYS_OLD = 30
now = datetime.now(timezone.utc)
cutoff_date = now - timedelta(days=DAYS_OLD)

def delete_old_objects():
    # List all objects in the bucket
    response = s3.list_objects_v2(Bucket=BUCKET_NAME)

    if 'Contents' not in response:
        print("No objects found in the bucket.")
        return

    deleted_objects = []

    for obj in response['Contents']:
        # Get the object's LastModified time
        last_modified = obj['LastModified']

        # If the object is older than the cutoff date
        if last_modified < cutoff_date:
            deleted_objects.append(obj['Key'])

    if deleted_objects:
        s3.delete_objects(Bucket=BUCKET_NAME, Delete={'Objects': deleted_objects})
```

The browser status bar at the bottom indicates the URL is <https://us-west-2.console.aws.amazon.com/lambda/home?region=us-west-2#/functions/Vivek3?newFunction=true&tab=code>.

The screenshot shows the AWS S3 buckets page. The left sidebar includes links for Buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, and Feature spotlight. The main content area displays an account snapshot and a table of general purpose buckets.

Name	AWS Region	IAM Access Analyzer	Creation date
vivekbatch7	US West (Oregon) us-west-2	<a href="#">View analyzer for us-west-2</a>	July 21, 2024, 11:40:15 (UTC+05:30)

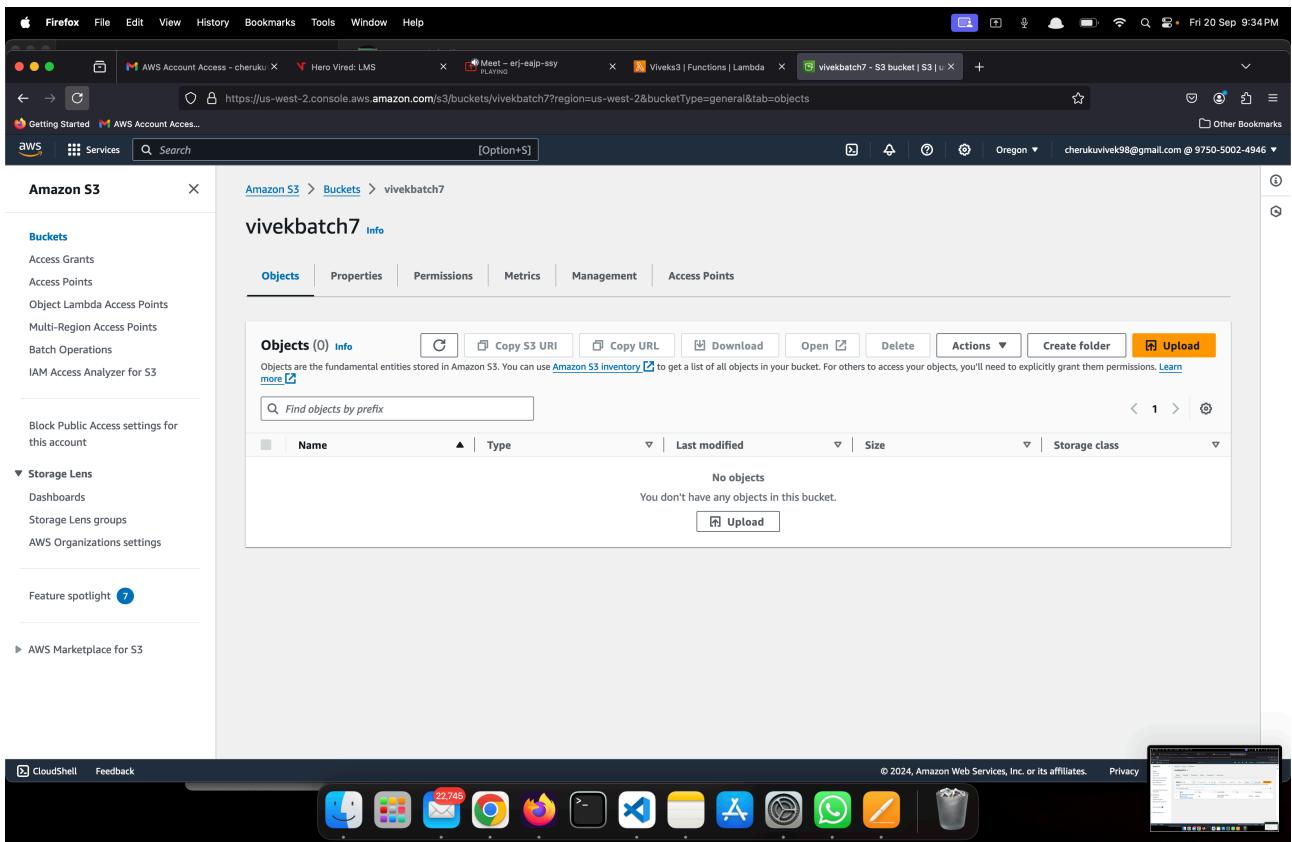
The browser status bar at the bottom indicates the URL is <https://us-west-2.console.aws.amazon.com/s3/home?region=us-west-2#>.

The screenshot shows the AWS Lambda console interface. A green banner at the top indicates that the function 'Viveks3' has been successfully updated. Below this, the 'Code' tab is selected in the navigation bar. The main area displays the 'lambda\_function.py' code and its execution results. The logs show a successful test event named 'Viveks3' with a response of 'null'. The 'Function Logs' section contains detailed log entries, including the start request ID, deleted objects, end request ID, report request ID, and duration. The status bar at the bottom right shows 'Status: Succeeded | Max memory used: 78 MB | Time: 343.03 ms'. The Mac OS Dock is visible at the bottom.

As there is already a image file in the S3 bucket which is more than 30 days

The screenshot shows the AWS S3 console interface. The left sidebar lists various services like Access Grants, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, and Storage Lens. The main area shows the 'vivekbatch7' bucket. Under the 'Objects' tab, a single object is listed: '2024-Formula1-Mercedes-AMG-W15-F1-E-Performance-001-1600.jpg'. The details pane shows the object was last modified on July 21, 2024, at 11:41:16 (UTC+05:30), with a size of 374.9 KB and a storage class of Standard. The Mac OS Dock is visible at the bottom.

Once the code is tested the images or any data which is more than 30 days is automatically deleted



## Assignment 3: Monitor Unencrypted S3 Buckets Using AWS Lambda and Boto3

**Objective:** To enhance your AWS security posture by setting up a Lambda function that detects any S3 bucket without server-side encryption.

**Task:** Automate the detection of S3 buckets that don't have server-side encryption enabled.

### Instructions:

#### 1. S3 Setup:

- Navigate to the S3 dashboard and create a few buckets. Ensure that a couple of them don't have server-side encryption enabled.

#### 2. Lambda IAM Role:

- In the IAM dashboard, create a new role for Lambda.
- Attach the `AmazonS3ReadOnlyAccess` policy to this role.

#### 3. Lambda Function:

- Navigate to the Lambda dashboard and create a new function.
- Choose Python 3.x as the runtime.
- Assign the IAM role created in the previous step.
- Write the Boto3 Python script to:

##### 1. Initialize a boto3 S3 client.

2. List all S3 buckets.
3. Detect buckets without server-side encryption.
4. Print the names of unencrypted buckets for logging purposes.

#### 4. Manual Invocation:

- After saving your function, manually trigger it.
- Review the Lambda logs to identify the buckets without server-side encryption.

Create a lambda function and add the python code with the configuration time upto 15mins Also dd the S3ReadOnlyAccess

The screenshot shows the AWS IAM console in a Firefox browser. The URL is https://us-east-1.console.aws.amazon.com/iam/home?region=us-west-2#/roles/details/VivekIAM/attach-policies. The page title is 'Attach policy to VivekIAM'. Under 'Current permissions policies (2)', there is a list titled 'Other permissions policies (1/990)'. A search bar shows 'amazons3'. The list includes four AWS managed policies: 'AmazonS3ObjectLambdaExecutionRolePolicy', 'AmazonS3OutpostsFullAccess', 'AmazonS3OutpostsReadOnlyAccess', and 'AmazonS3ReadOnlyAccess'. The 'AmazonS3ReadOnlyAccess' policy is selected and highlighted with a blue border. At the bottom right of the list, there are 'Cancel' and 'Add permissions' buttons. The status bar at the bottom of the browser window shows various application icons and the date '© 2024, Amazon Web Services, Inc. or its affiliates.'

The screenshot shows the AWS Lambda console interface. The main title bar indicates the region is 'us-west-2'. The browser address bar shows the URL: <https://us-west-2.console.aws.amazon.com/lambda/home?region=us-west-2#/functions/Vivekbucket?newFunction=true&tab=code>. The browser status bar shows the date as 'Sun 22 Sep 3:32 PM'.

The Lambda function 'Vivekbucket' has been successfully updated. The configuration page shows:

- General configuration:**
  - Description: Vivekbucket
  - Memory: 128 MB
  - Ephemeral storage: 512 MB
  - Timeout: 15 min 0 sec
  - SnapStart: None
- Code:** The 'Code source' tab is selected, showing the code editor with a single file named 'lambda\_function.py'. The code content is:

```
VivekS3bucket
```

The Lambda function ARN is listed as arn:aws:lambda:us-west-2:975050024946:function:Vivekbucket. The function URL is provided as a link.

Once done run the code !

The screenshot shows the AWS Lambda console interface, similar to the previous one but with the 'Test' tab selected under the 'Code' tab.

The Lambda function 'Vivekbucket' has been successfully updated. The test results are displayed:

- Execution results:** Test Event Name: VivekS3bucket, Response: null
- Function Logs:** The logs show a successful execution with the following details:

```
START RequestId: 7e135bfc-9f68-4f30-bce4-5553f830668d Version: $LATEST
All buckets have server-side encryption enabled.
END RequestId: 7e135bfc-9f68-4f30-bce4-5553f830668d
REPORT RequestId: 7e135bfc-9f68-4f30-bce4-5553f830668d Duration: 45321.61 ms Billed Duration: 45322 ms Memory Size: 128 MB Max Memory Used: 79 MB Init Duration: 449.99 ms
Request ID
7e135bfc-9f68-4f30-bce4-5553f830668d
```

The Lambda function ARN is listed as arn:aws:lambda:us-west-2:975050024946:function:Vivekbucket. The function URL is provided as a link.

## **Assignment 4: Automatic EBS Snapshot and Cleanup Using AWS Lambda and Boto3**

**Objective:** To automate the backup process for your EBS volumes and ensure that backups older than a specified retention period are cleaned up to save costs.

**Task:** Automate the creation of snapshots for specified EBS volumes and clean up snapshots older than 30 days.

### **Instructions:**

#### **1. EBS Setup:**

- Navigate to the EC2 dashboard and identify or create an EBS volume you wish to back up.
- Note down the volume ID.

#### **2. Lambda IAM Role:**

- In the IAM dashboard, create a new role for Lambda.
- Attach policies that allow Lambda to create EBS snapshots and delete them (`AmazonEC2FullAccess` for simplicity, but be more restrictive in real-world scenarios).

#### **3. Lambda Function:**

- Navigate to the Lambda dashboard and create a new function.
- Choose Python 3.x as the runtime.
- Assign the IAM role created in the previous step.
- Write the Boto3 Python script to:
  1. Initialize a boto3 EC2 client.
  2. Create a snapshot for the specified EBS volume.
  3. List snapshots and delete those older than 30 days.
  4. Print the IDs of the created and deleted snapshots for logging purposes.

#### **4. Event Source (Bonus):**

- Attach an event source, like Amazon CloudWatch Events, to trigger the Lambda function at your desired backup frequency (e.g., every week).

#### **5. Manual Invocation:**

- After saving your function, either manually trigger it or wait for the scheduled event.
- Go to the EC2 dashboard and confirm that the snapshot is created and old snapshots are deleted.

```

44     # Log the created snapshot
45     print(f"Created snapshot: {snapshot_id}")
46
47     return snapshot_id
48
49 def delete_old_snapshots():
50     # Filter snapshots by volume and created by the user (to avoid deleting system snapshots)
51     response = ec2.describe_snapshots(
52         Filters=[
53             {"Name": "volume-id", "Values": [VOLUME_ID]},
54             {"Name": "owner-id", "Values": ["self"]} # Only own snapshots
55         ]
56     )
57
58     deleted_snapshots = []
59
60     for snapshot in response['Snapshots']:
61         snapshot_id = snapshot['SnapshotId']
62         start_time = snapshot['StartTime']
63
64         # If the snapshot is older than the cutoff date
65         if start_time < cutoff_date:
66             print(f"Deleting snapshot: {snapshot_id} (Created on {start_time})")
67             ec2.delete_snapshot(SnapshotId=snapshot_id)
68             deleted_snapshots.append(snapshot_id)
69
70     # Log the deleted snapshot IDs
71     if deleted_snapshots:
72         print(f"Deleted snapshots: {deleted_snapshots}")
73     else:
74         print("No snapshots older than 30 days found.")
75
76 def lambda_handler(event, context):
77

```

**Code properties**

Add the code in the lambda function and the create and EC2 instance with the EBS

Get the id of the EBS and add it in the code

```

1 import boto3
2
3 # Initialize EC2 resource
4 ec2 = boto3.client('ec2')
5
6 def lambda_handler(event, context):
7     # Stop EC2 instances with 'Auto-Stop' tag
8     stop_instances()
9
10    # Start EC2 instances with 'Auto-Start' tag
11    start_instances()
12
13 def stop_instances():
14    # Filter instances with the tag 'Auto-Stop'
15    response = ec2.describe_instances(
16        Filters=[
17            {
18                'Name': 'tag:Auto-Stop',
19                'Values': ['Auto-Stop']
20            },
21            {
22                'Name': 'instance-state-name',
23                'Values': ['running'] # Only stop running instances
24            }
25        ]
26    )
27
28    instances_to_stop = []
29    for reservation in response['Reservations']:
30        for instance in reservation['Instances']:
31            instances_to_stop.append(instance['InstanceId'])
32
33    # Stop instances if any are found
34

```

**Code properties**

Sun 22 Sep 3:42 PM

AWS Account Access - cherukivek98@gmail.com @ 9750-5002-4946

Getting Started AWS Account Access... Hero Vired: LMS Meet - fzv-ipsr-evw Vivekbucket | Functions | Lamb: Launch an instance | EC2 | us-west-2#LaunchInstances

Services Search Optk You are sharing your entire screen. Stop Sharing

EC2 Instances Launch an instance

**Success** Successfully initiated launch of instance (i-05f89a56bab3064a3)

Launch log

**Next Steps**

What would you like to do next with this instance, for example "create alarm" or "create backup"

1 2 3 4 5 6

Create billing and free tier usage alerts Connect to your instance Connect an RDS database Create EBS snapshot policy

To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.

Connect to instance Learn more

Configure the connection between an EC2 instance and a database to allow traffic flow between them.

Connect an RDS database Create a new RDS database Learn more

Create EBS snapshot policy

Manage detailed monitoring Create Load Balancer Create AWS budget Manage CloudWatch alarms

Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period.

Create Load Balancer

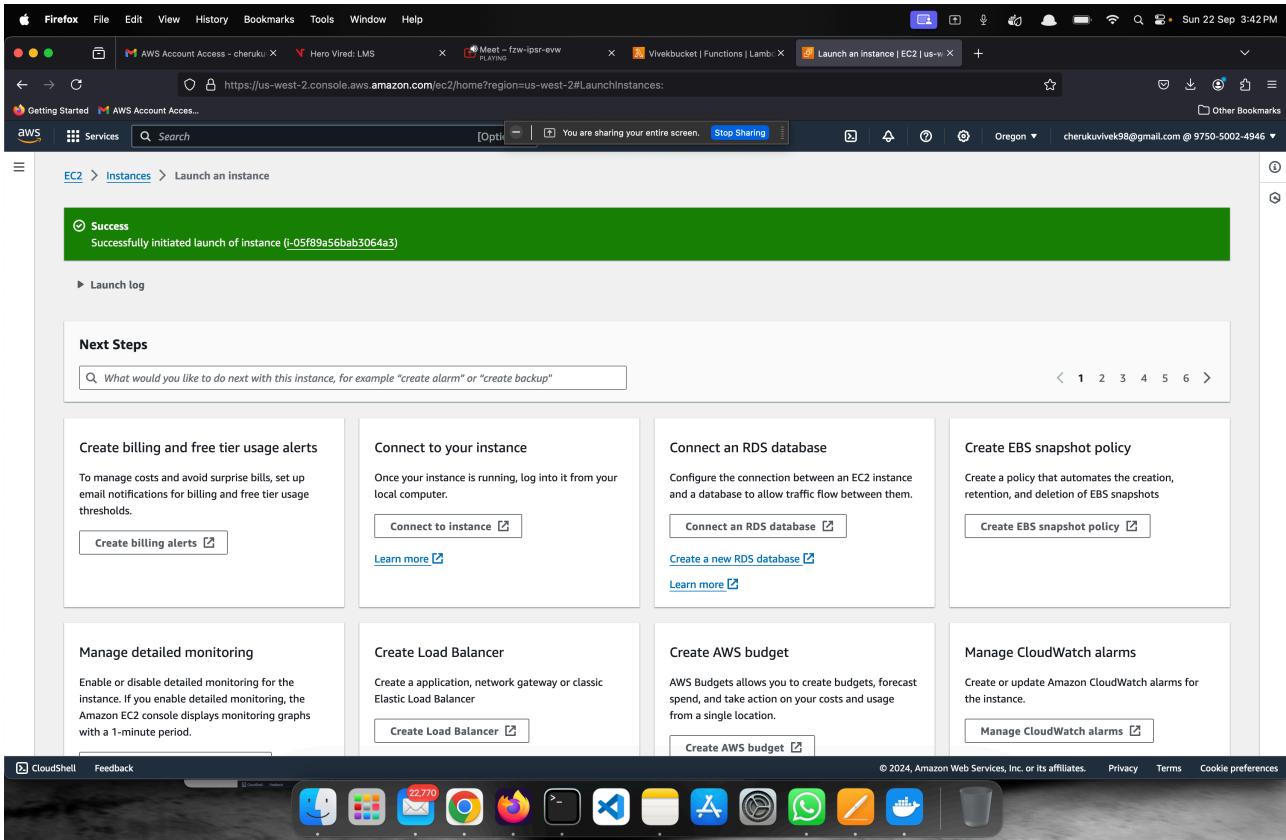
Create a application, network gateway or classic Elastic Load Balancer

Create AWS budget

AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location.

Manage CloudWatch alarms

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Sun 22 Sep 3:44 PM

AWS Account Access - cherukivek98@gmail.com @ 9750-5002-4946

Getting Started AWS Account Access... Hero Vired: LMS Meet - fzv-ipsr-evw Vivekbucket | Functions | Lamb: Volumes | EC2 | us-west-2

Services Search Optk You are sharing your entire screen. Stop Sharing

Volumes (1/1) Info Actions Create volume

Search Clear filters

Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created	Availability Zone
EBSEC2 Vivek	vol-04c230747eb5ca00f	gp3	8 GiB	3000	125	snap-014c752...	2024/09/22 15:42 GMT+5:...	us-west-2d

Volume ID: vol-04c230747eb5ca00f (EBSEC2 Vivek)

Details Status checks Monitoring Tags

Volume ID: vol-04c230747eb5ca00f (EBSEC2 Vivek) Size: 8 GiB Type: gp3 Volume status: Insufficient data

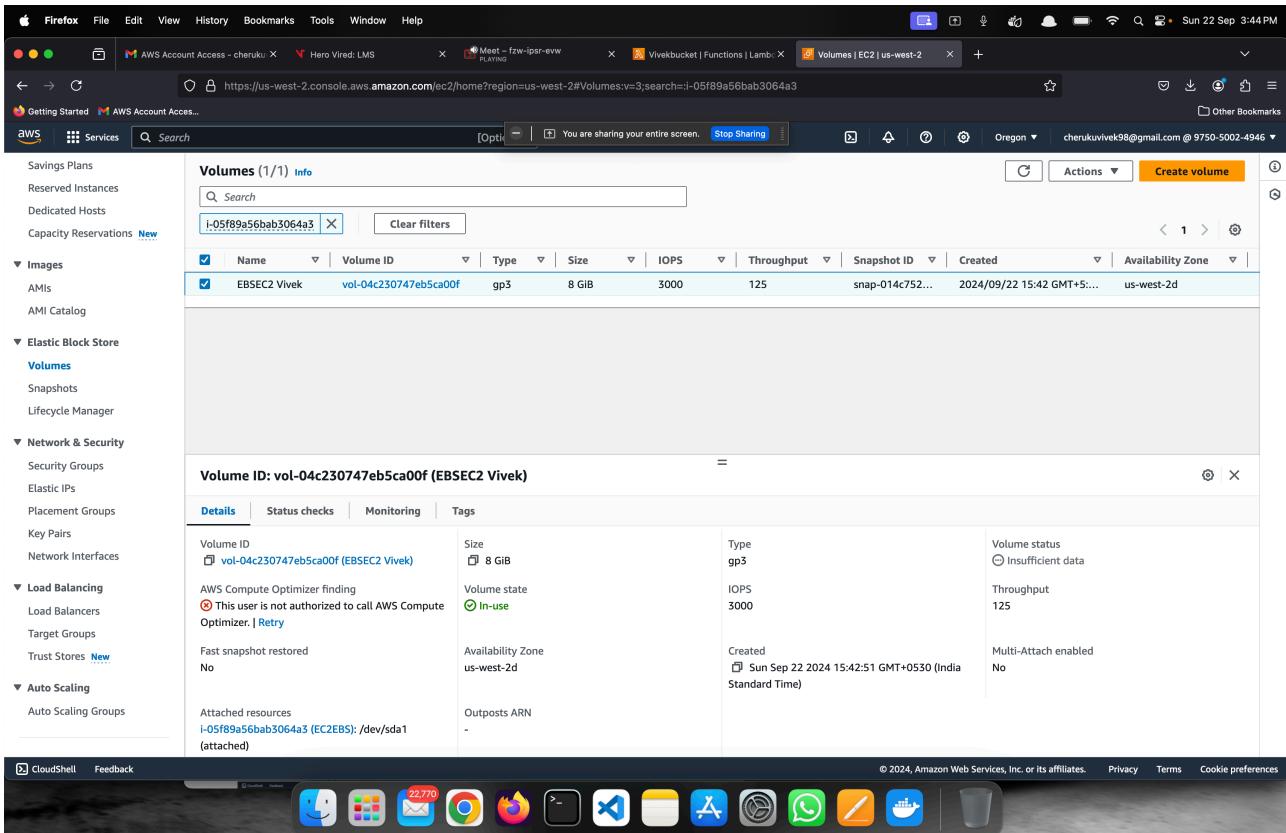
AWS Compute Optimizer finding: This user is not authorized to call AWS Compute Optimizer. | Retry

Volume state: In-use IOPS: 3000 Throughput: 125

Fast snapshot restored: No Availability Zone: us-west-2d Created: Sun Sep 22 2024 15:42:51 GMT+0530 (India Standard Time)

Attached resources: i-05f89a56bab3064a3 (EC2EBS): /dev/sda1 (attached) Outposts ARN: Multi-Attach enabled: No

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Sun 22 Sep 3:45 PM

Firefox File Edit View History Bookmarks Tools Window Help

AWS Account Access - cheruku Hero Vired: LMS Meet - fzr-ipsr-eww Vivekbucket | Functions | Lambda Volumes | EC2 | us-west-2

Getting Started AWS Account Access... Services Search [Opt] You are sharing your entire screen. Stop Sharing

Successfully updated the function Vivekbucket.

Code source Info

File Edit Find View Go Tools Window Test Deploy

Execution result: Status: Succeeded Max memory used: 87 MB Time: 556.53 ms

Environment Go to Anything (⌘ P) lambda\_function Execution results

Test Event Name: Viveks3bucket

Response: null

Function Log:

```
START RequestId: 4296fed4-9c60-4d42-9617-6e4ac23fdf7b Version: $LATEST
Created snapshot: snap-01ec03c09738eb73a
No snapshots older than 30 days found.
END RequestId: 4296fed4-9c60-4d42-9617-6e4ac23fdf7b
REPORT RequestId: 4296fed4-9c60-4d42-9617-6e4ac23fdf7b Duration: 556.53 ms Billed Duration: 557 ms Memory Size: 128 MB Max Memory Used: 87 MB Init Duration: 507.58 ms
RequestID: 4296fed4-9c60-4d42-9617-6e4ac23fdf7b
```

Code properties Info

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

