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.

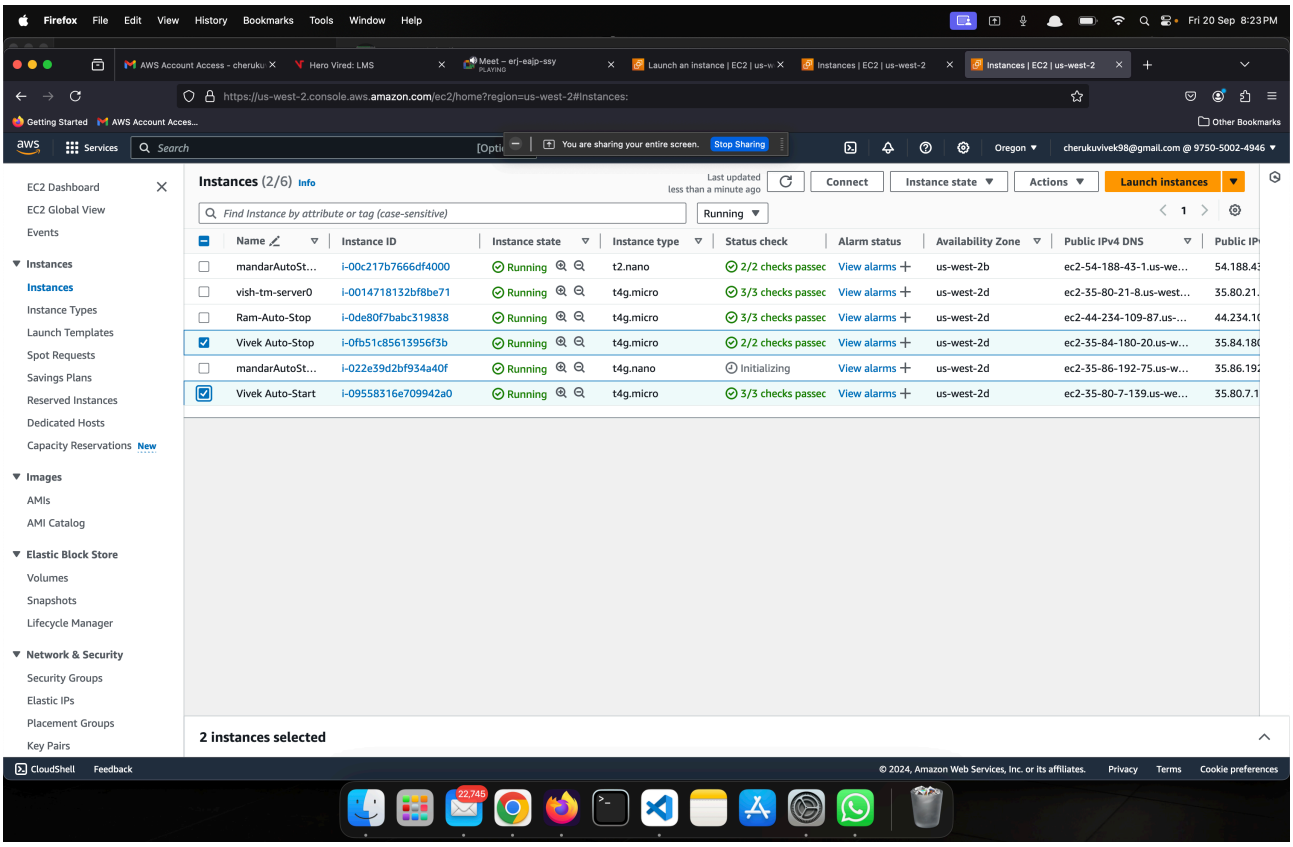
3. Stop the `Auto-Stop` instances and start the `Auto-Start` instances.

4. Print instance IDs that were affected for logging purposes.

4. Manual Invocation:

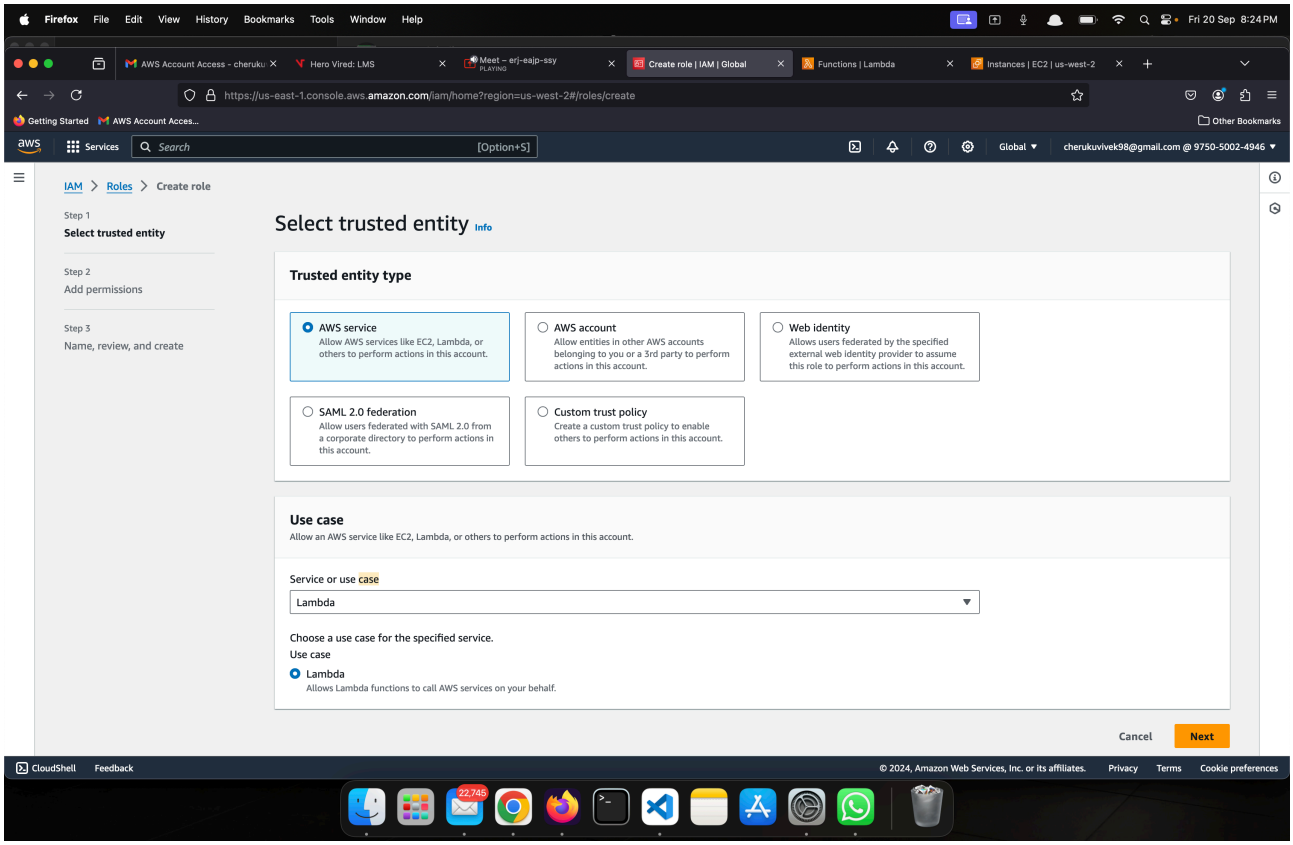
- After saving your function, manually trigger it.

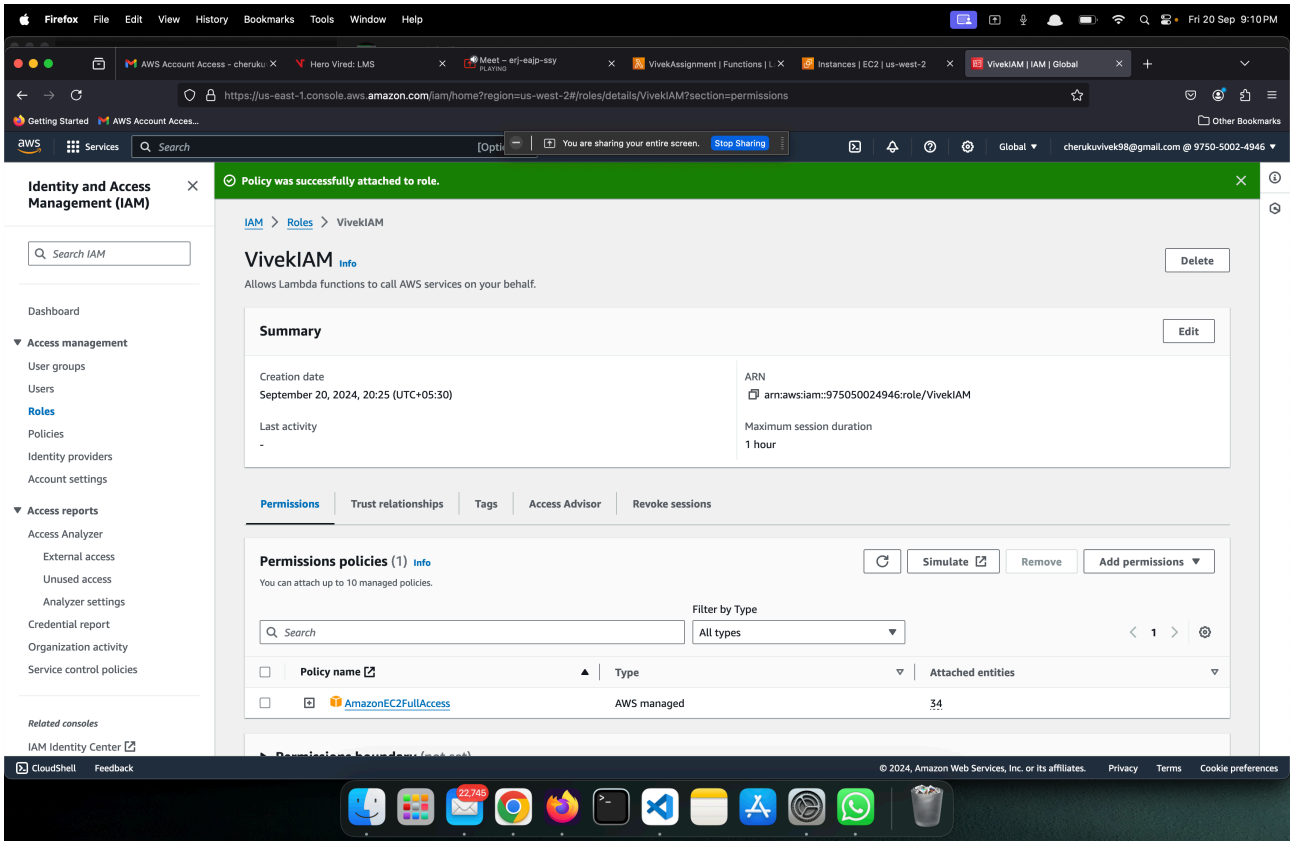
- Go to the EC2 dashboard and confirm that the instances' states have changed according to their tags.

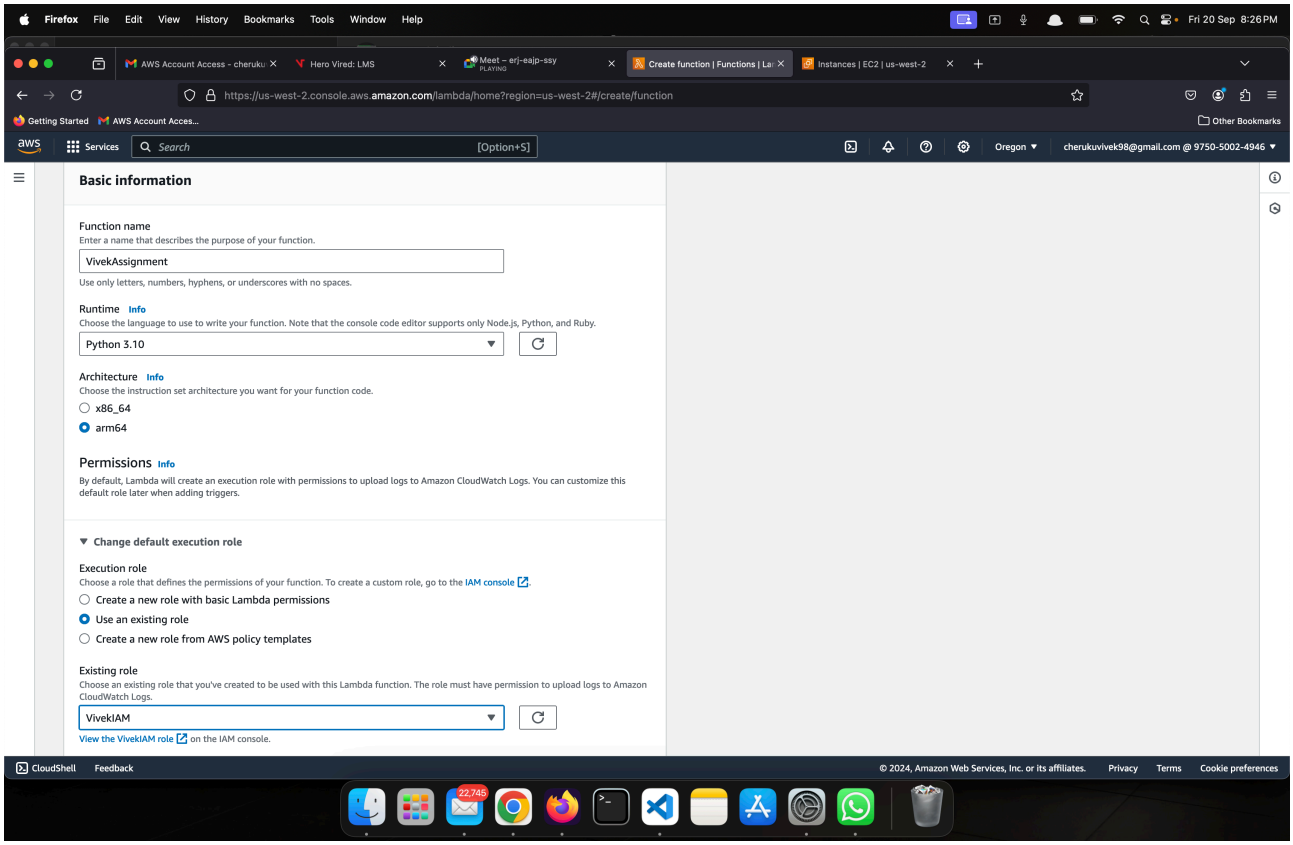


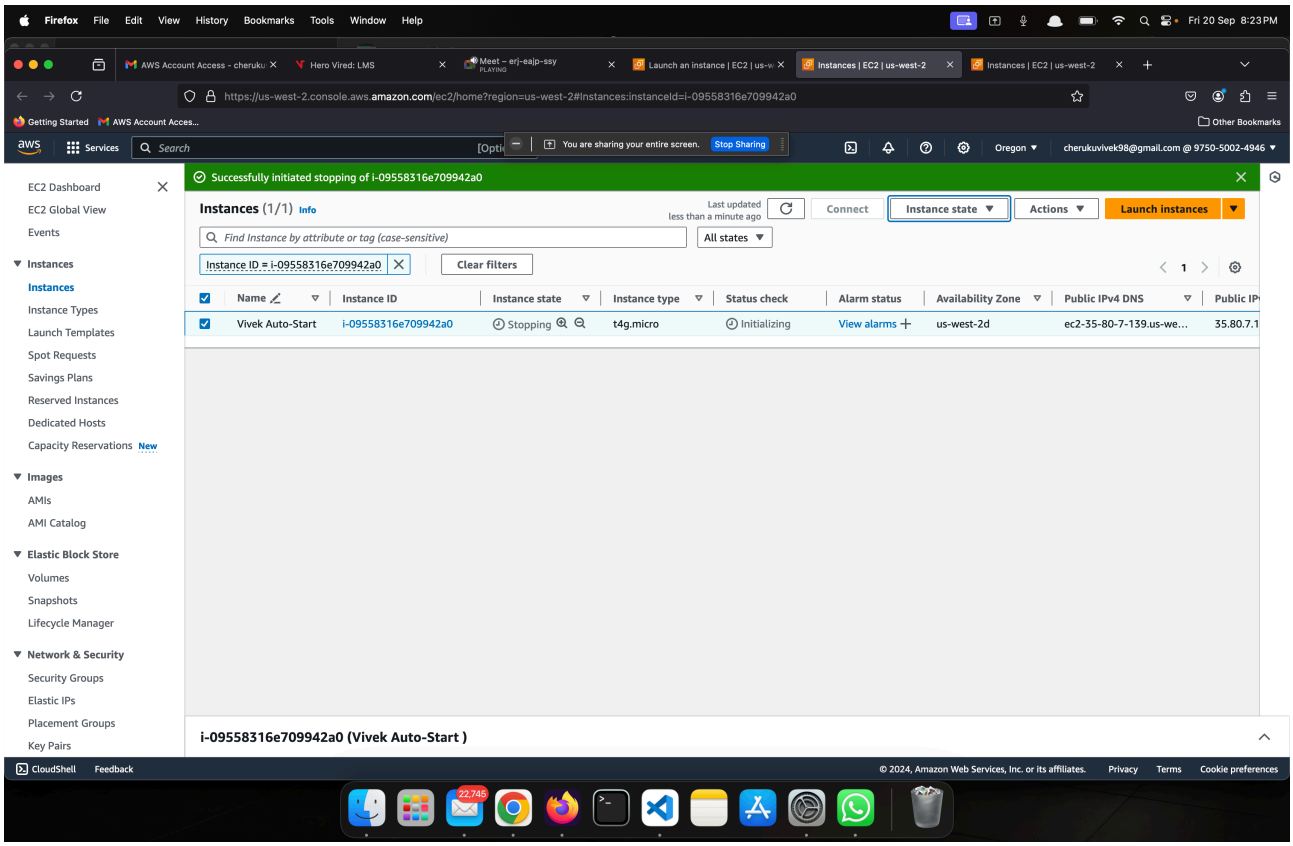
Create a two EC2 Instances with the name Auto-Start and Auto-Stop and later stop the EC2 (Auto-start)

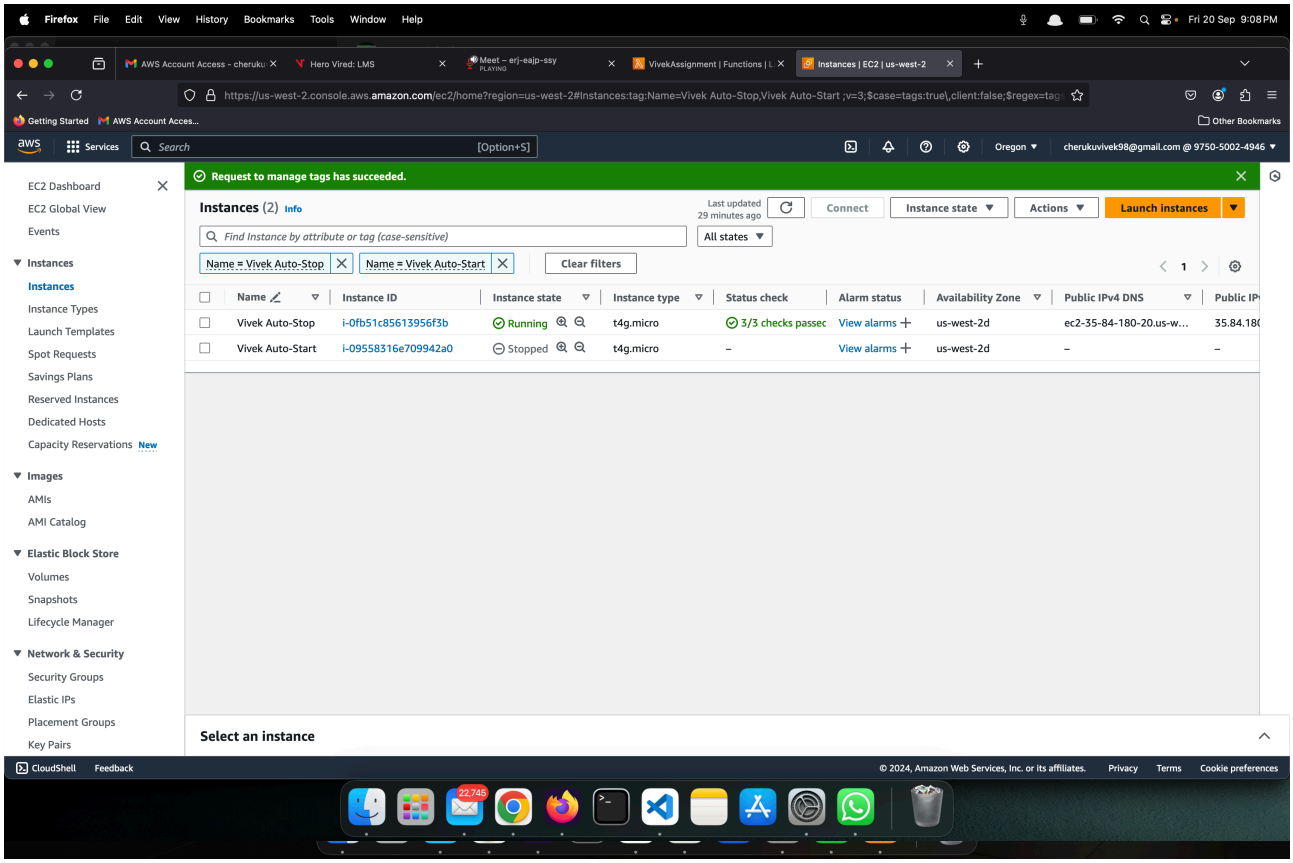
Create an IAM role and the LAMDA Function and add the Full EC2 Access






Basic information
Function name
Enter a name that describes the purpose of your function.
VivekAssignment
Use only letters, numbers, hyphens, or underscores with no spaces.
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.10
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
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.
VivekIAM
View the VivekIAM role on the IAM console.

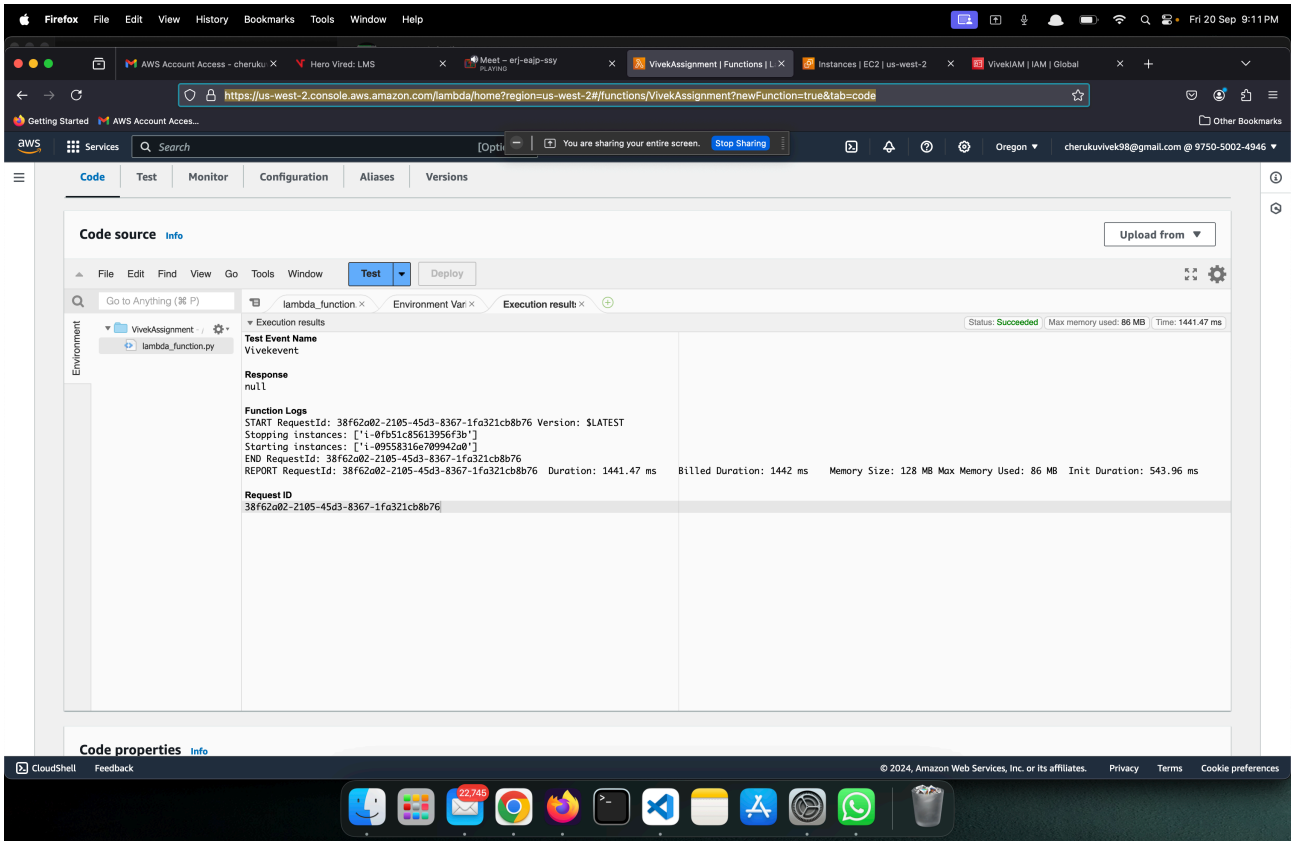

Successfully initiated stopping of i-09558316e709942a0
Instances (1/1) Info
Connect
Instance state
Actions
Launch instances
Vivek Auto-Start
i-09558316e709942a0
Stopping
t4g.micro
Initializing
View alarms
us-west-2d
i-09558316e709942a0 (Vivek Auto-Start)

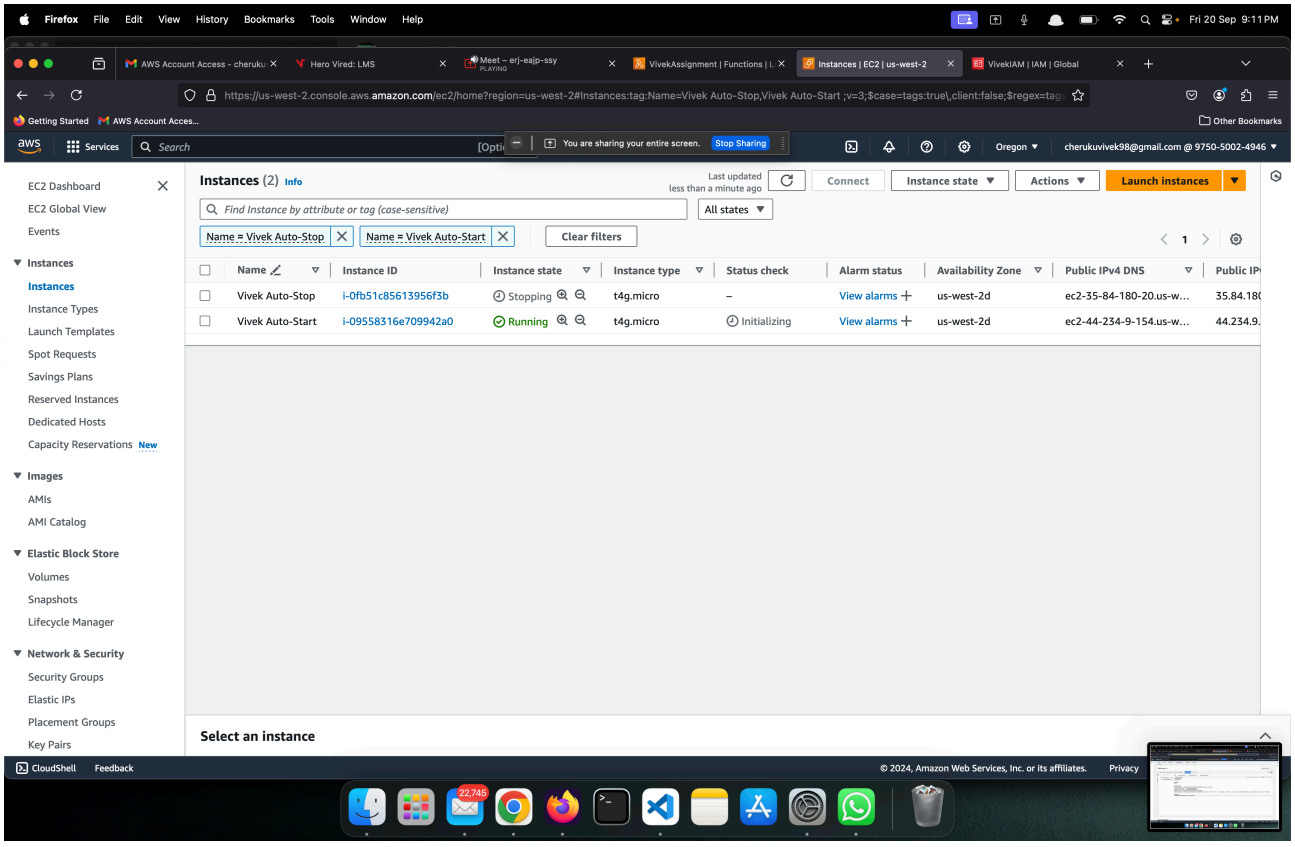


Once done add the lambda function code and changes the tag which is provided earlier in the EC2 - Auto-Start and Auto Stop instances.

The code is provided in the GIT HUB - Assignment 1

Once done with the changes deploy and test the code.


Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
Test
Deploy
lambda_function
Environment Var
Execution result
Execution results
Status: Succeeded
Max memory used: 86 MB
Time: 1441.47 ms
Test Event Name
Vivekevent
Response
null
Function Logs
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
Code properties Info


EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Instances (2) Info
Connect
Instance state
Actions
Launch instances
Name = Vivek Auto-Stop
Name = Vivek Auto-Start
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Vivek Auto-Stop
i-0fb51c85613956f3b
Stopping
t4g.micro
View alarms
us-west-2d
Vivek Auto-Start
i-09558316e709942a0
Running
t4g.micro
Initializing
View alarms
us-west-2d
Select an instance

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.

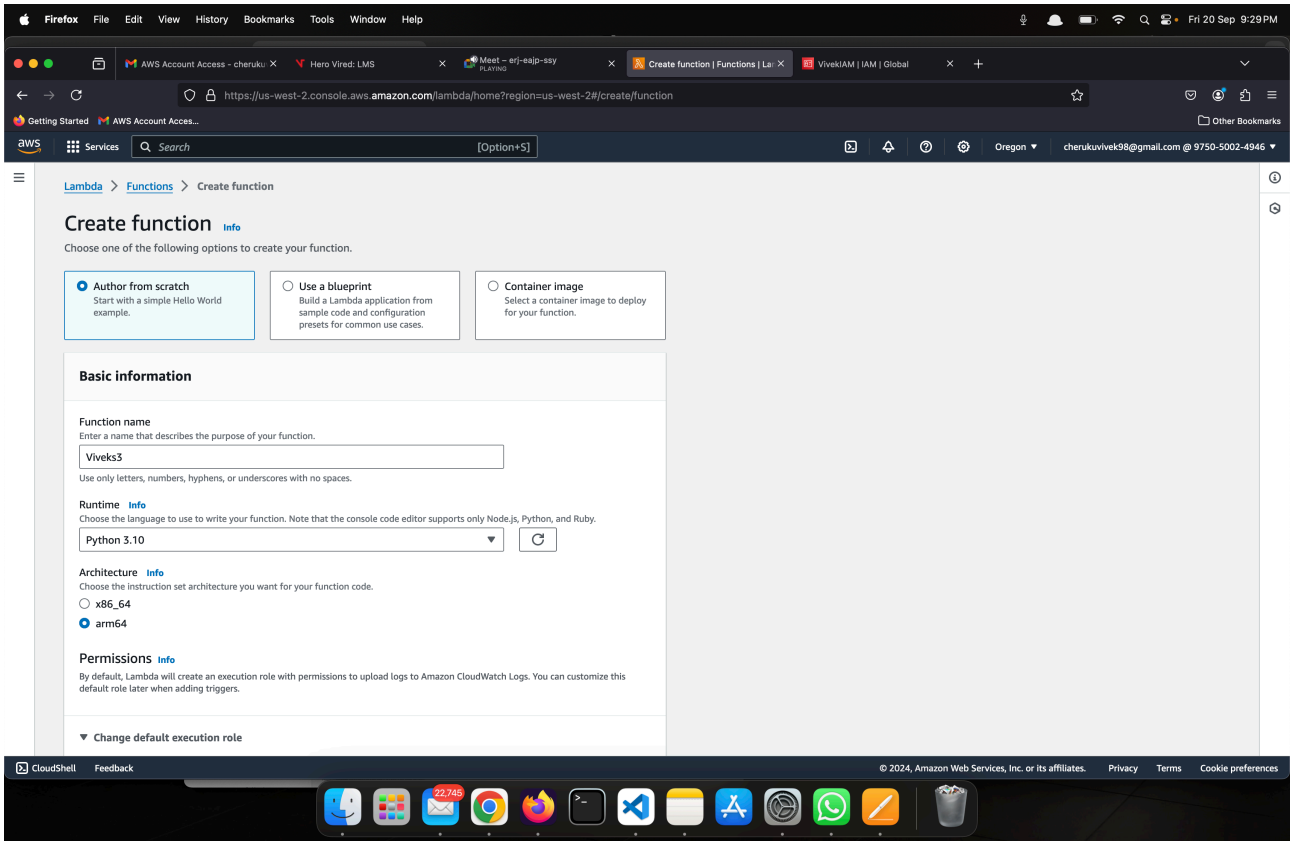
     4. Print the names of deleted objects for logging purposes.

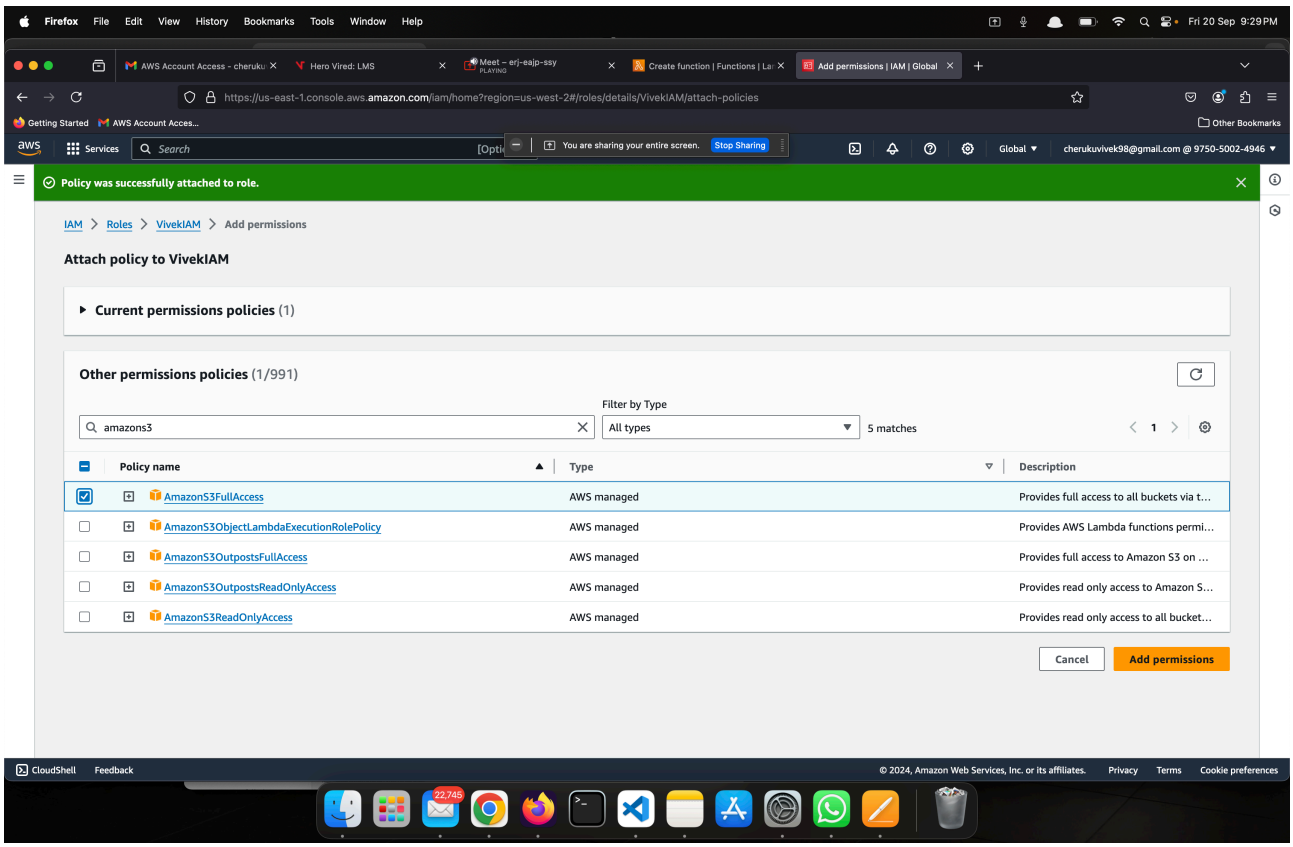
4. Manual Invocation:

   - After saving your function, manually trigger it.

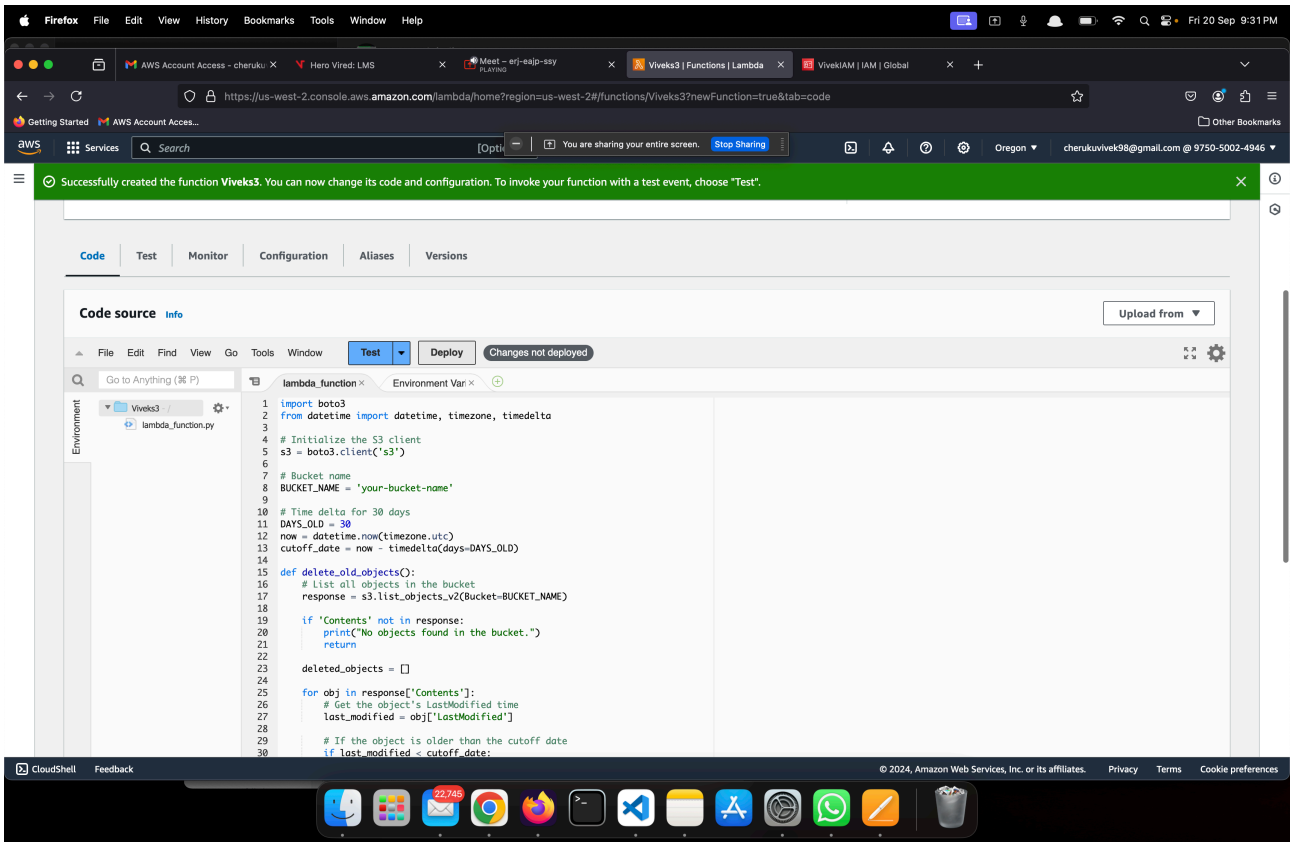
   - Go to the S3 dashboard and confirm that only files newer than 30 days remain.

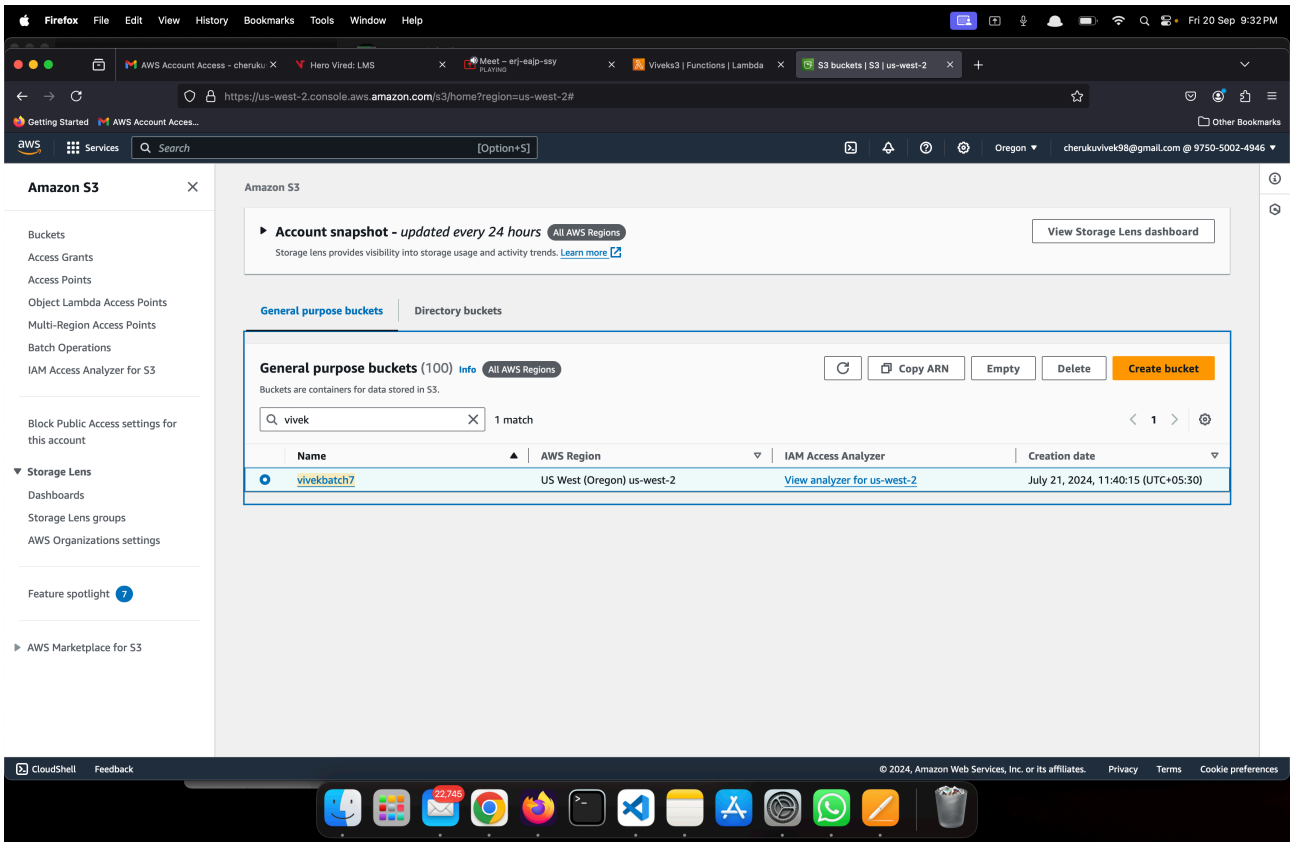
Create a lambda function with the IAM Role with S3 full access

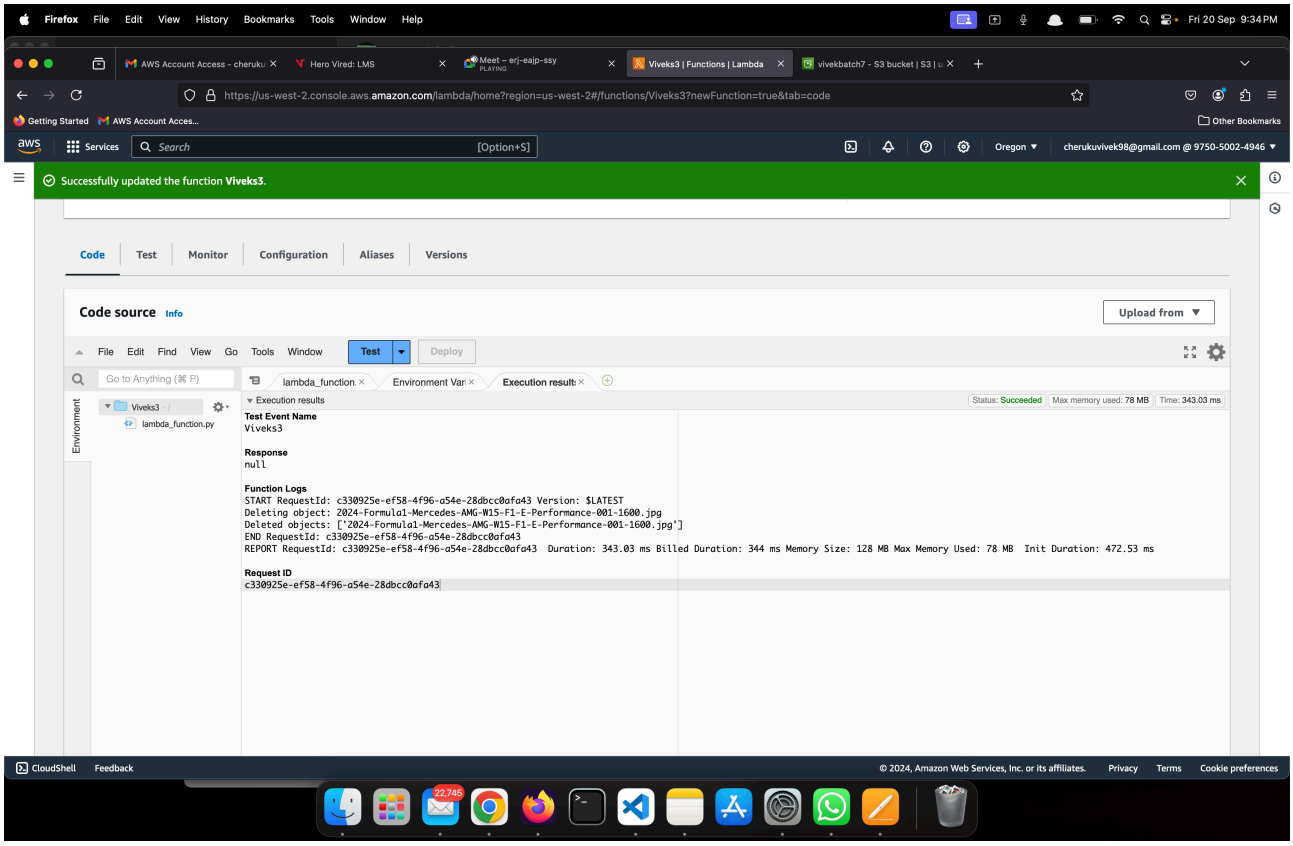




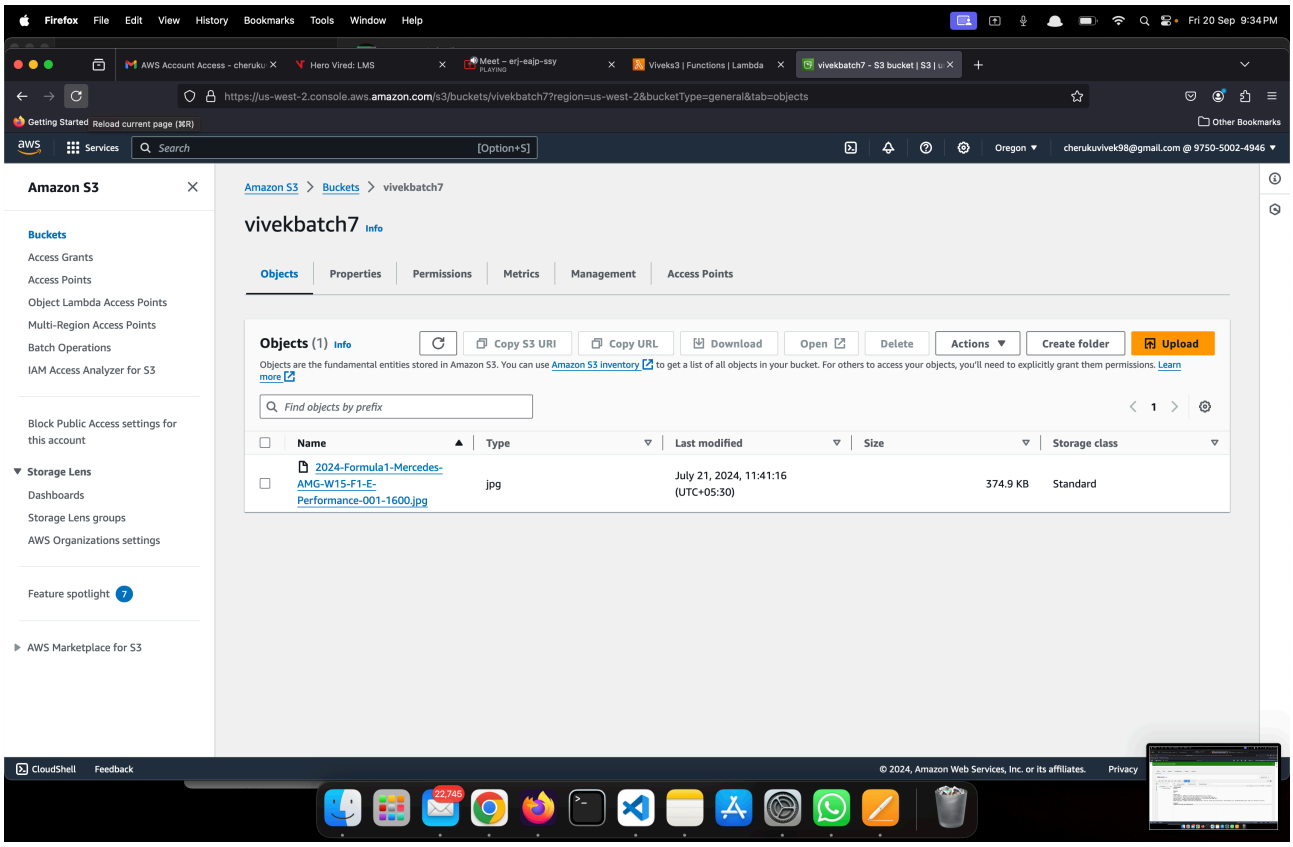
Now add the code with the S3 bucket name



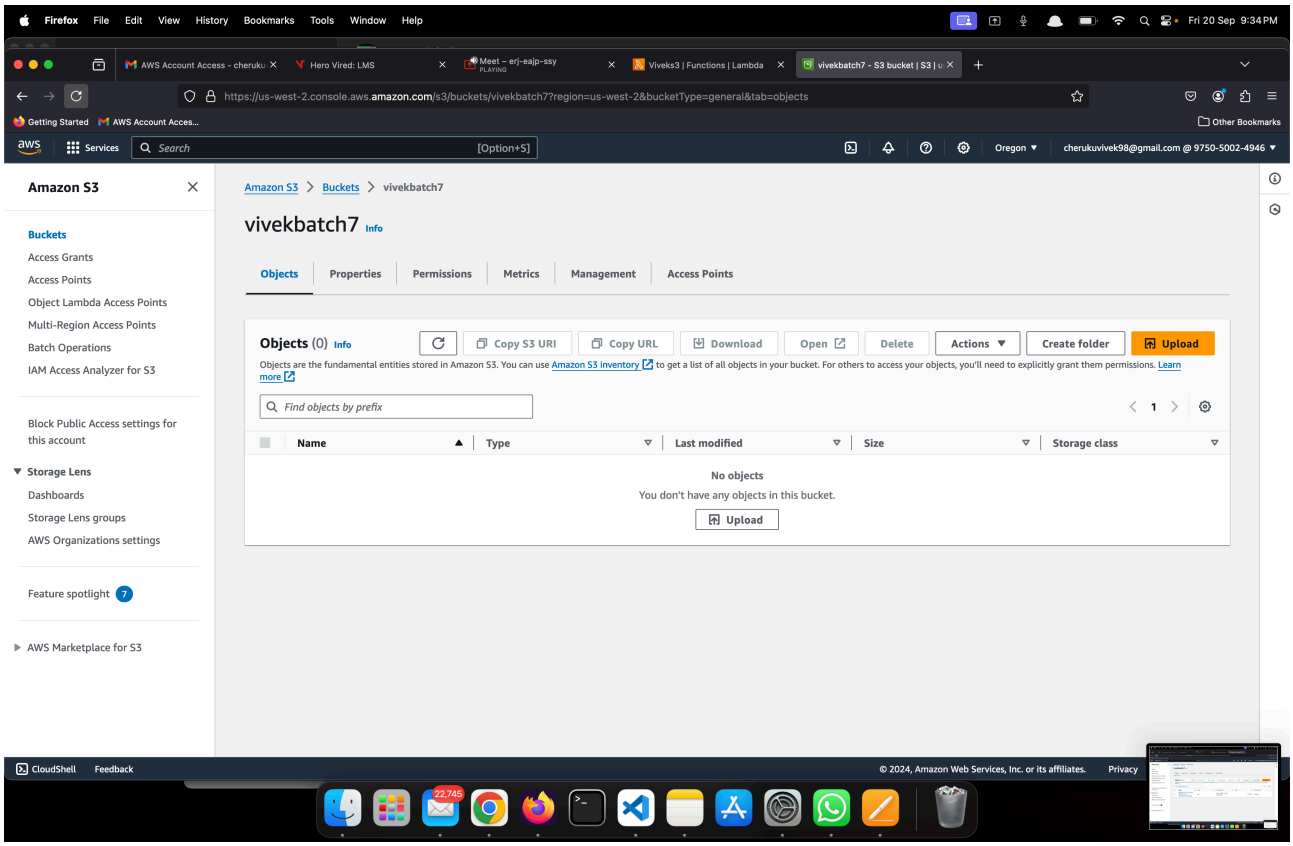




As there is already a image file in the S3 bucket which is more than 30 days



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.

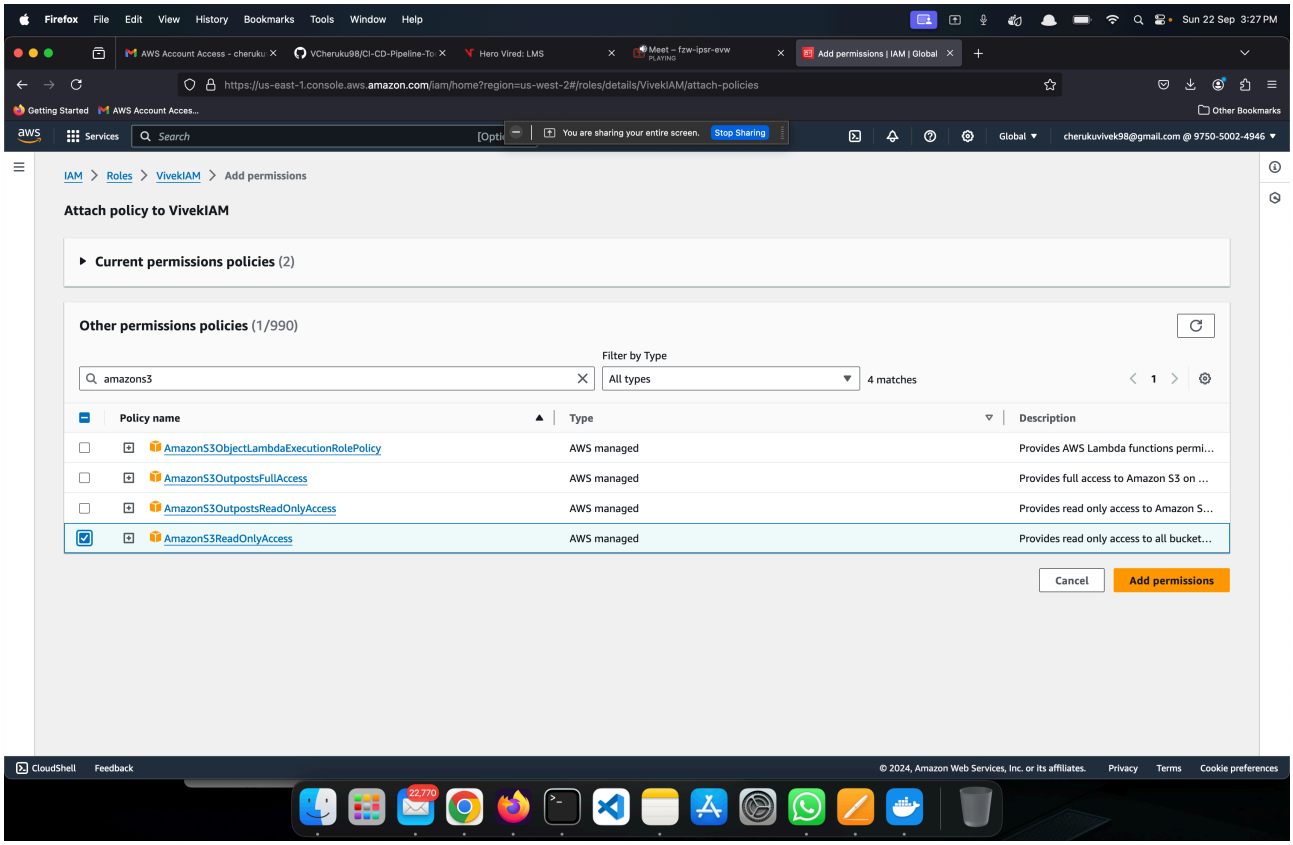
4. Print the names of unencrypted buckets for logging purposes.

4. Manual Invocation:

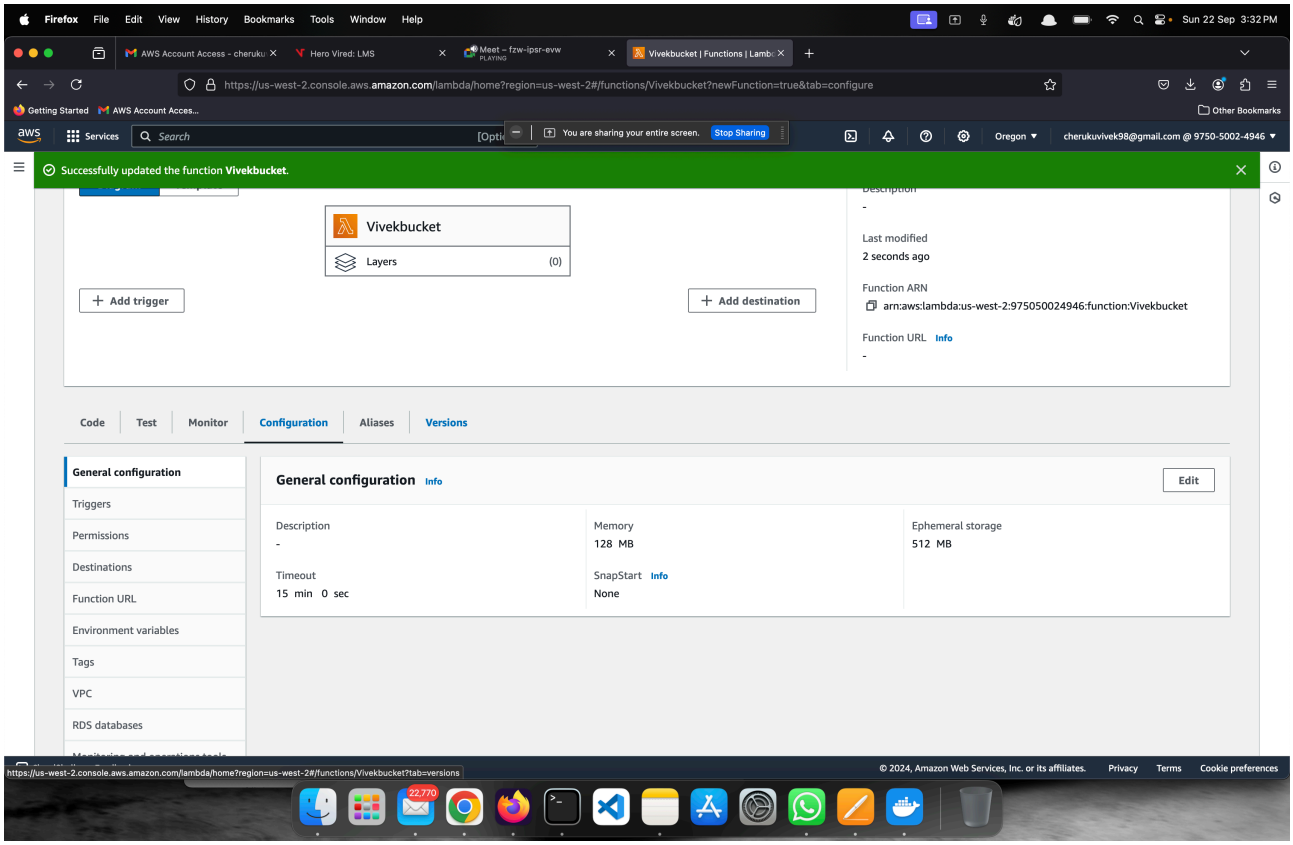
- After saving your function, manually trigger it.

- Review the Lambda logs to identify the buckets without server-side encryption.

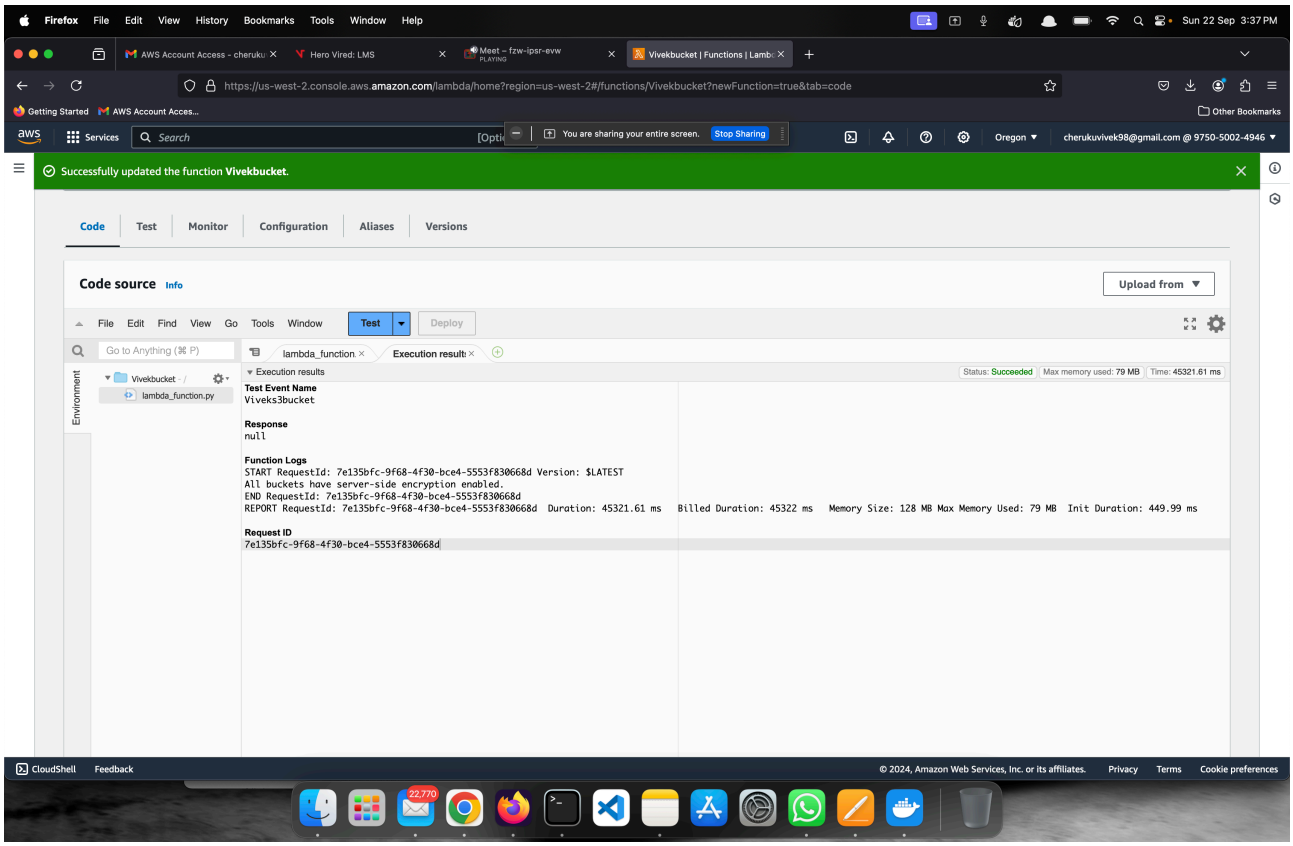
Create a lambda function and add the python code with the configuration time upto 15mins Also dd the S3ReadOnlyAccess



‘



Once done run the code !



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.

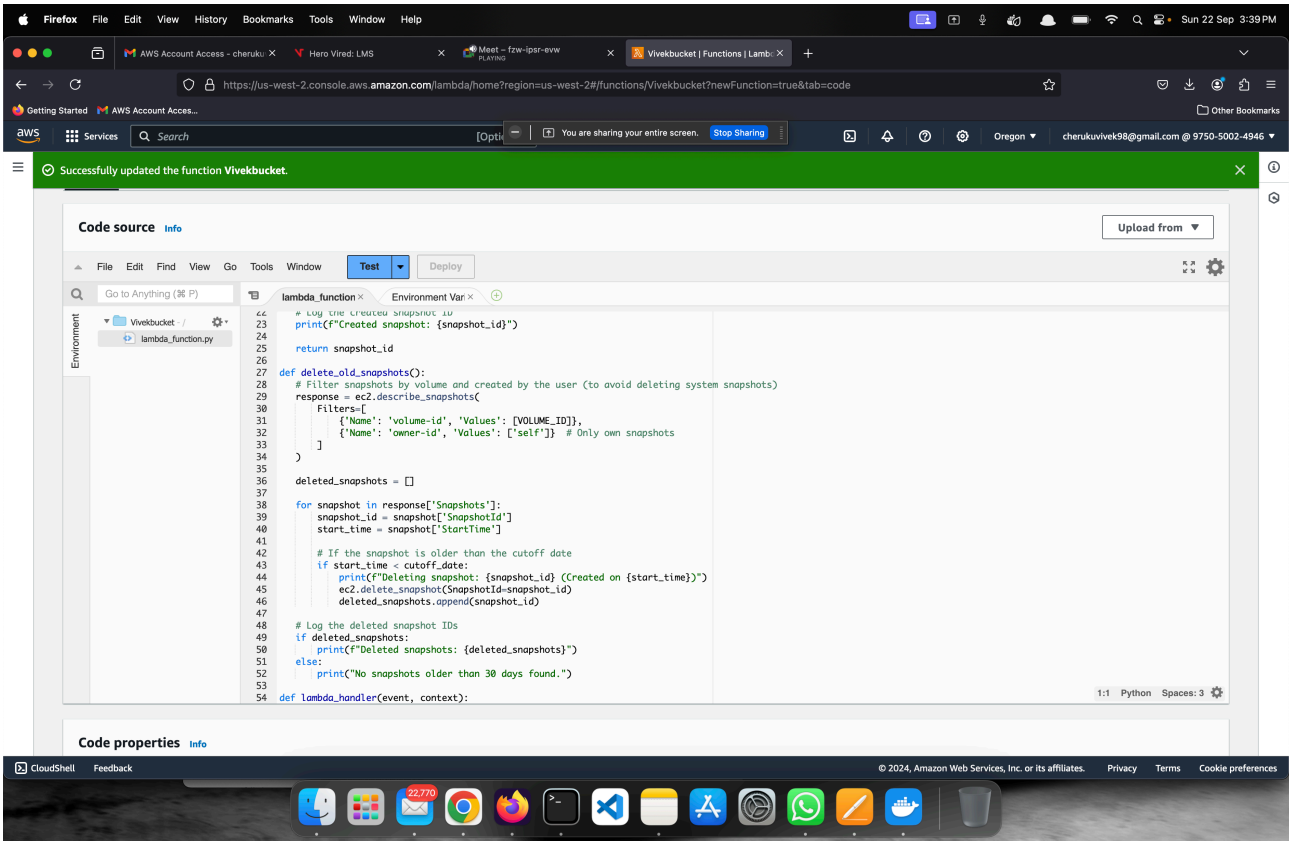
4. Event Source (Bonus):

   - 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:

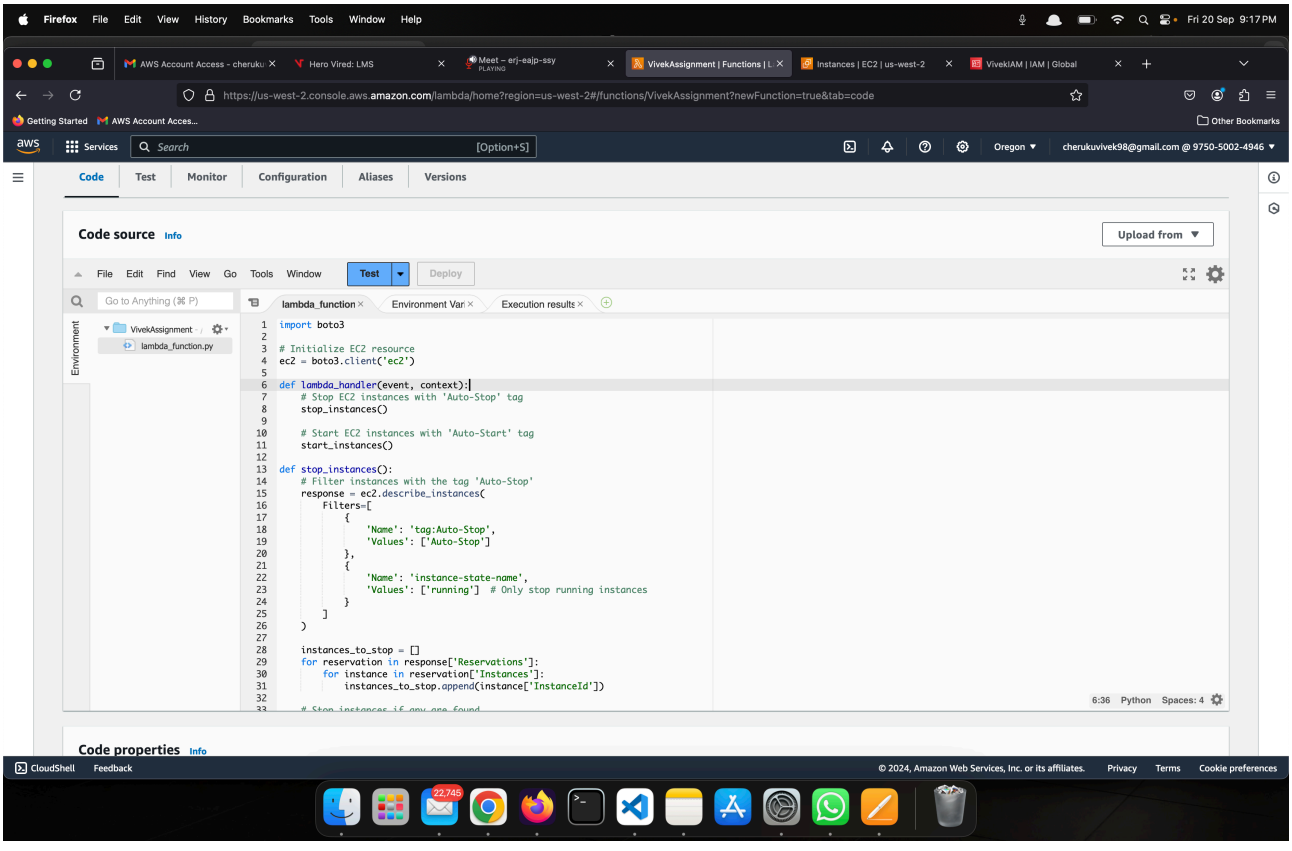
   - After saving your function, either manually trigger it or wait for the scheduled event.

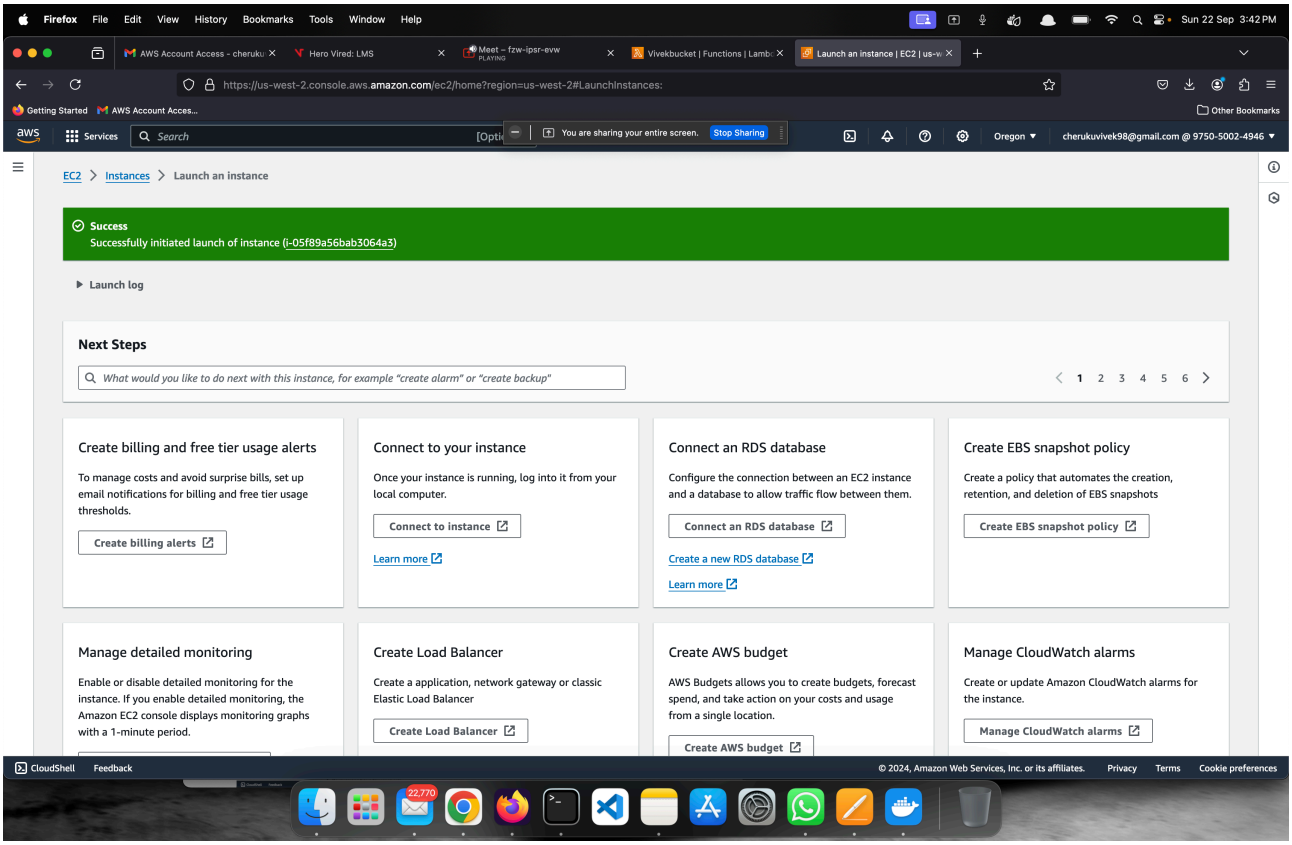
   - Go to the EC2 dashboard and confirm that the snapshot is created and old snapshots are deleted.

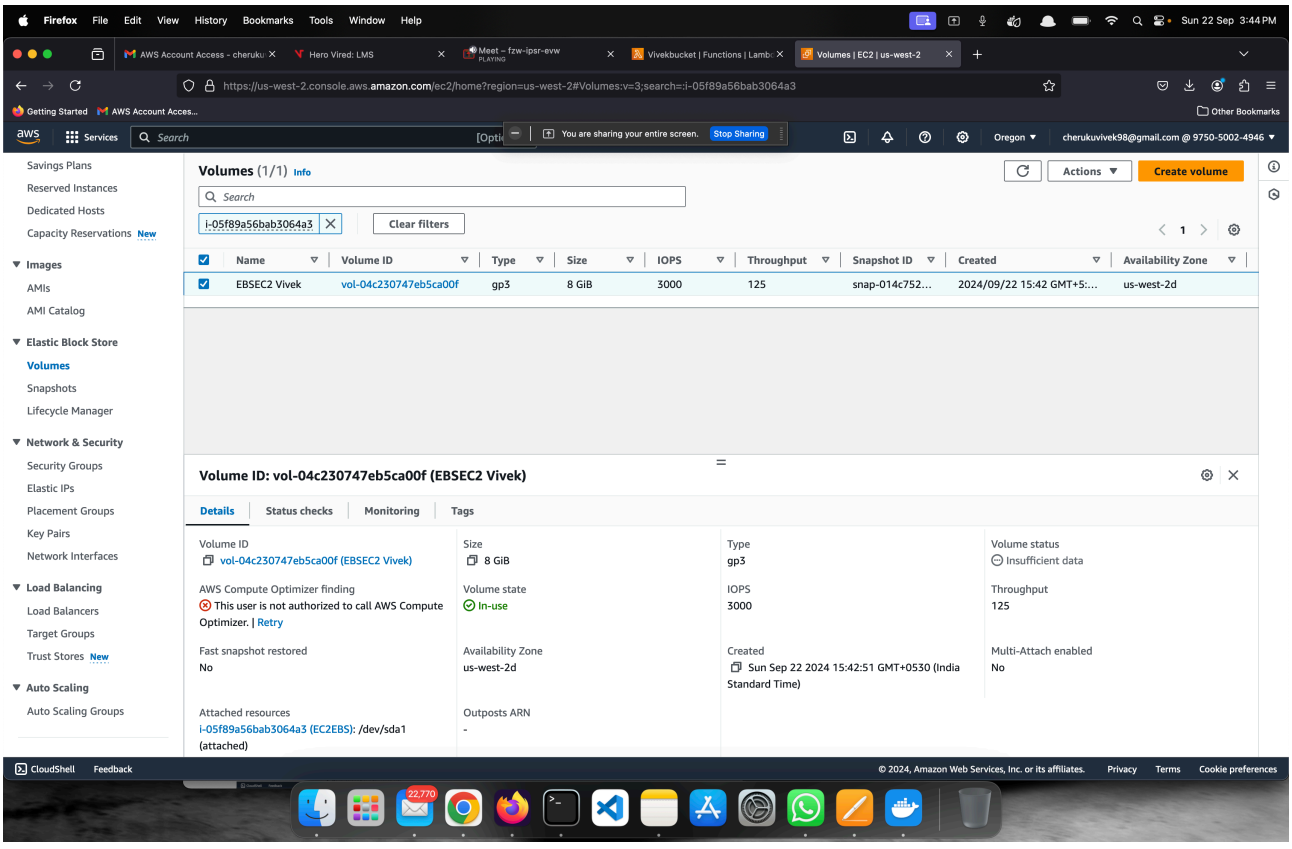


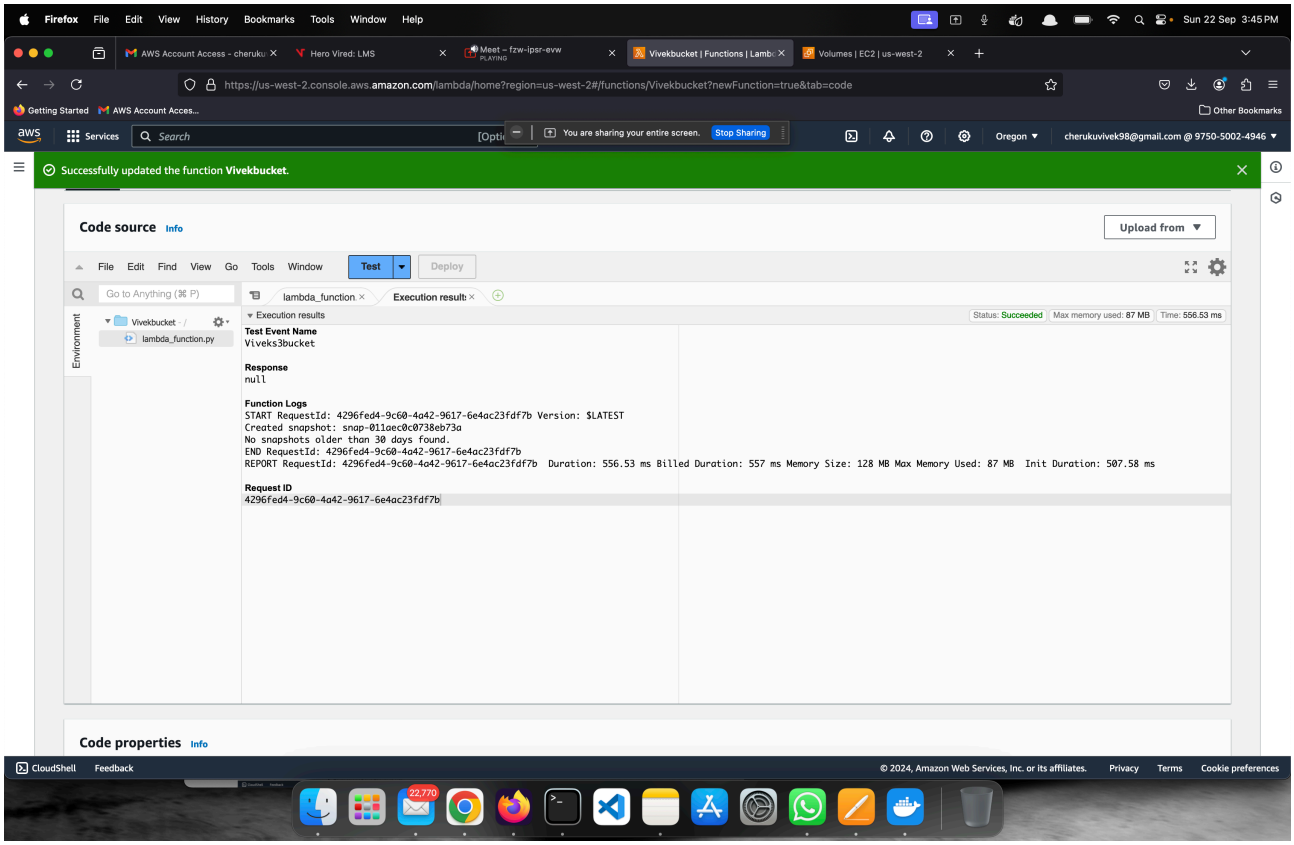
Add the code in the lambda function and the create and EC2 instance with the EBS

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




Launch an instance | EC2 | us-w
https://us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#LaunchInstances:
You are sharing your entire screen.
Stop Sharing
Oregon
cherukuvivek98@gmail.com @ 9750-5002-4946
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"
Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts
Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more
Connect an RDS database
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
Create a policy that automates the creation, retention, and deletion of EBS snapshots
Create EBS snapshot policy
Manage detailed monitoring
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 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.
Create AWS budget
Manage CloudWatch alarms
Create or update Amazon CloudWatch alarms for the instance.
Manage CloudWatch alarms
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Volumes | EC2 | us-west-2
https://us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#Volumes:v=3;search=:i-05f89a56bab3064a3
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
Volumes (1/1) Info
Search
i-05f89a56bab3064a3
Clear filters
Actions
Create volume
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)
Multi-Attach enabled
No
Attached resources
i-05f89a56bab3064a3 (EC2EBS): /dev/sda1 (attached)
Outposts ARN
-
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Firefox File Edit View History Bookmarks Tools Window Help
Sun 22 Sep 3:45 PM
https://us-west-2.console.aws.amazon.com/lambda/home?region=us-west-2#/functions/Vivekbucket?newFunction=true&tab=code
You are sharing your entire screen.
Stop Sharing
Oregon
cherukuvivek98@gmail.com @ 9750-5002-4946
Successfully updated the function Vivekbucket.
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (⌘ P)
lambda_function
Execution result
Environment
Vivekbucket
lambda_function.py
Execution results
Status: Succeeded
Max memory used: 87 MB
Time: 556.53 ms
Test Event Name
Viveks3bucket
Response
null
Function Logs
START RequestId: 4296fed4-9c60-4a42-9617-6e4ac23fdf7b Version: $LATEST
Created snapshot: snap-011aec0c0738eb73a
No snapshots older than 30 days found.
END RequestId: 4296fed4-9c60-4a42-9617-6e4ac23fdf7b
REPORT RequestId: 4296fed4-9c60-4a42-9617-6e4ac23fdf7b Duration: 556.53 ms Billed Duration: 557 ms Memory Size: 128 MB Max Memory Used: 87 MB Init Duration: 507.58 ms
Request ID
4296fed4-9c60-4a42-9617-6e4ac23fdf7b
Code properties Info
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences