Develop a system that automatically manages the lifecycle of a web application hosted on EC2 instances, monitors its health, and reacts to changes in traffic by scaling resources. Furthermore, administrators receive notifications regarding the infrastructure's health and scaling events.

Detailed Breakdown:

1. Web Application Deployment:

- Use `boto3` to:
- Create an S3 bucket to store your web application's static files.
- Launch an EC2 instance and configure it as a web server (e.g., Apache, Nginx). - Deploy the web application onto the EC2 instance.

2. Load Balancing with ELB:

- Deploy an Application Load Balancer (ALB) using `boto3`.
- Register the EC2 instance(s) with the ALB.

3. Auto Scaling Group (ASG) Configuration:

- Using `boto3`, create an ASG with the deployed EC2 instance as a template.
- Configure scaling policies to scale in/out based on metrics like CPU utilization or network traffic.

4. SNS Notifications:

- Set up different SNS topics for different alerts (e.g., health issues, scaling events, high traffic).
- Integrate SNS with Lambda so that administrators receive SMS or email notifications.

5. Infrastructure Automation:

- Create a single script using `boto3` that:
- Deploys the entire infrastructure.
- Updates any component as required.
- Tears down everything when the application is no longer needed.

6. Optional Enhancement – Dynamic Content Handling:

- Store user-generated content or uploads on S3.
- When a user uploads content to the web application, it gets temporarily stored on the EC2 instance. A background process (or another Lambda function) can move this to the S3 bucket and update the application's database to point to the content's new location on S3.

Objectives:

- Gain a comprehensive understanding of key AWS services and their integration. - Understand the lifecycle of a dynamic web application and its infrastructure.
- Learn how to automate infrastructure deployment and management tasks using boto3. - Experience with real-time monitoring and alerting systems.

https://github.com/VCheruku98/Scaling-Automation

1. Web Application Deployment:

- Use `boto3` to:
- Create an S3 bucket to store your web application's static files.
- Launch an EC2 instance and configure it as a web server (e.g., Apache, Nginx). - Deploy the web application onto the EC2 instance.

2. Load Balancing with ELB:

- Deploy an Application Load Balancer (ALB) using `boto3`.
- Register the EC2 instance(s) with the ALB.

3. Auto Scaling Group (ASG) Configuration:

- Using `boto3`, create an ASG with the deployed EC2 instance as a template.

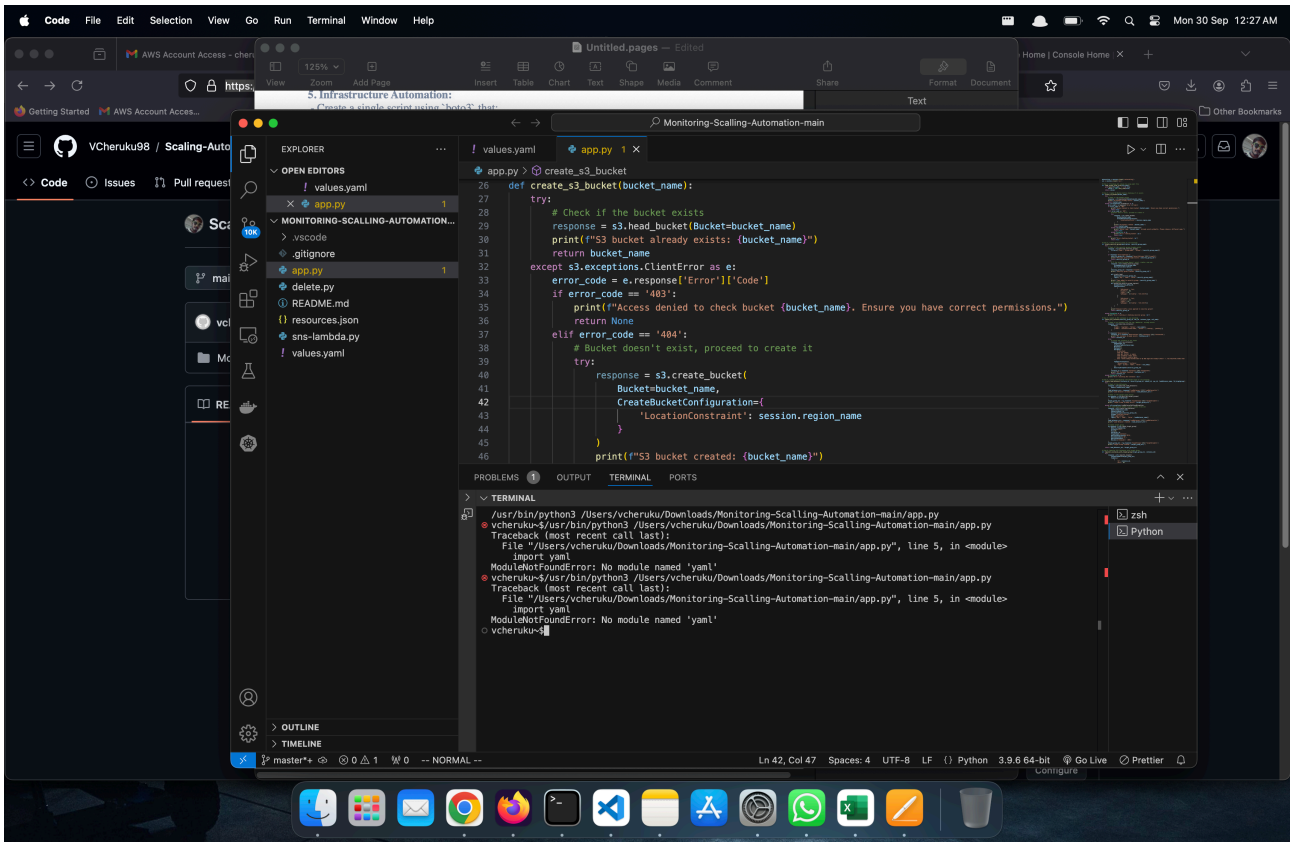
- Configure scaling policies to scale in/out based on metrics like CPU utilization or network traffic.

4. SNS Notifications:

- Set up different SNS topics for different alerts (e.g., health issues, scaling events, high traffic).
- Integrate SNS with Lambda so that administrators receive SMS or email notifications.

Run the python code with all the details and information in the values.ymal
Update your AWS CLI profile ID in the python for the access.
Python code is in the git Repo

Note : Despite of trying to run it in the VS code the Mac OS is raising an error



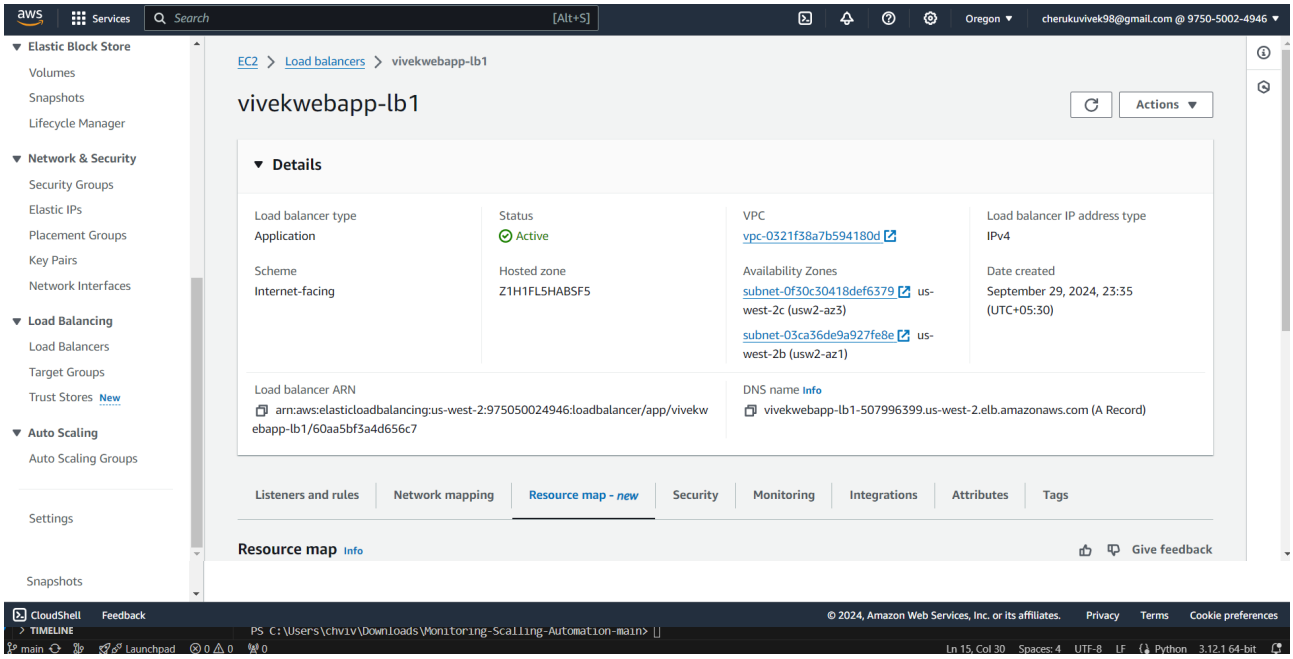
Hence switched to the windows OS to run in the VS Code. However, will another error cannot run on it.
Switched to starting an EC2 with all the nesscary libraries installed
Git, pip, python, AWS CLI, boto3, ymal. Run the python code.

With the python code the following 4 Tasks will be completed where the EC2, Load balancer, Auto Scaling and SNS will be created with the name mentioned earlier in the yaml file

EC2

LoadBalancer

AutoScaling with the computer tracking policies

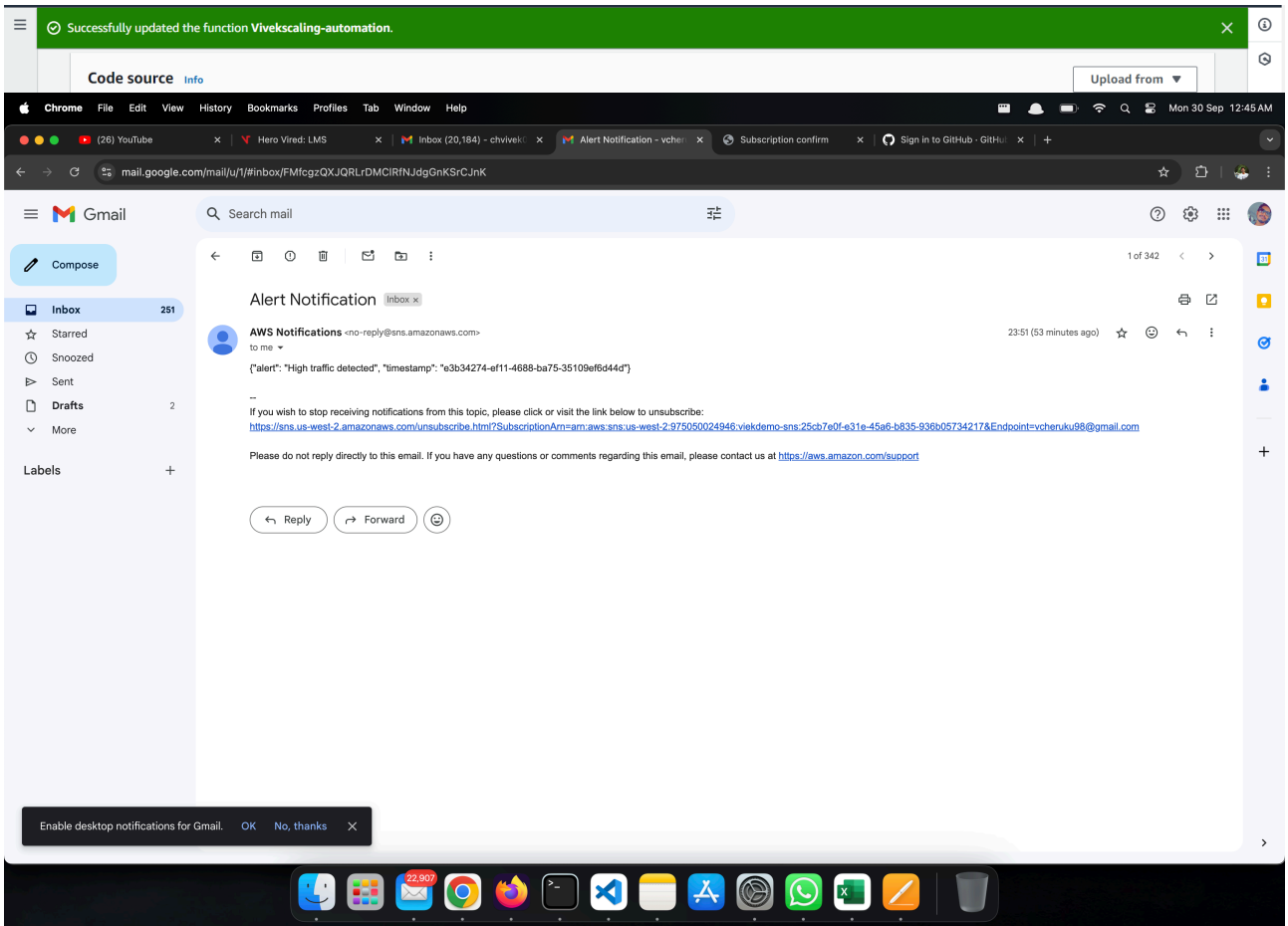


SNS - with your email confirmed for the notifications.

5. Infrastructure Automation:

- Create a single script using `boto3` that:
- Deploys the entire infrastructure.
- Updates any component as required.
- Tears down everything when the application is no longer needed.

Create the lambda function for a script using boto3. The code mentioned in the GIT Repo



us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#AutoScalingGroupDetails:id=vivek-asg1;view=details

Gmail YouTube Maps Translate News

aws Services Search [Alt+S] Oregon cherukuvivek98@gmail.com @ 9750-5002-4946

- 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
- Settings

EC2 > Auto Scaling groups > vivek-asg1

vivek-asg1

Details | Activity | Automatic scaling | Instance management | Monitoring | Instance refresh

Group details

Edit

Auto Scaling group name	Desired capacity	Desired capacity type	Amazon Resource Name (ARN)
vivek-asg1	1	Units (number of instances)	arn:aws:autoscaling:us-west-2:975050024946:autoScalingGroup:bda4577f-28b2-457e-b6a5-0ce87209b6ce:autoScalingGroupName/vivek-asg1
Date created	Minimum capacity	Status	
Sun Sep 29 2024 23:35:19 GMT+0530 (India Standard Time)	1	-	
	Maximum capacity		
	2		

Launch template

Edit

Launch template	AMI ID	Instance type	Owner
lt-0a57364b870079cb5 vivekk	ami-05134c8ef96964280	t2.micro	arn:aws:iam::975050024946:user/cherukuvivek98@gmail.com

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

us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#AutoScalingGroupDetails:id=vivek-asg1;view=scaling

Gmail YouTube Maps Translate News

aws Services Search [Alt+S] Oregon cherukuvivek98@gmail.com @ 9750-5002-4946

- 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
- Settings

application demand changes quickly, but with a recurring pattern, or when your EC2 instances require more time to initialize.

Dynamic scaling policies (1) Info

Actions | Create dynamic scaling policy

1

cpu-target-tracking-policy

Policy type
Target tracking scaling

Enabled or disabled
Enabled

Execute policy when
As required to maintain Average CPU utilization at 70

Take the action
Add or remove capacity units as required

Instances need
300 seconds to warm up before including in metric

Scale in
Enabled

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

Gmail YouTube Maps Translate News

aws Services Search [Alt+S] Oregon cherukuvivek98@gmail.com @ 9750-5002-4946

Amazon SNS

- Dashboard
- Topics
- Subscriptions
- Mobile
  - Push notifications
  - Text messaging (SMS)

Details

Name	Display name
viekdemo-sns	-
ARN	Topic owner
arn:aws:sns:us-west-2:975050024946:viekdemo-sns	975050024946
Type	
Standard	

Subscriptions | Access policy | Data protection policy | Delivery policy (HTTP/S) | Delivery status logging | Encryption | Tags

Subscriptions (1)

Edit | Delete | Request confirmation | Confirm subscription | Create subscription

Search

1

ID	Endpoint	Status	Protocol
25cb7e0f-e31e-45a6-b835-936b...	vcheruku98@gmail.com	Confirmed	EMAIL

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

Tears down everything when the application is no longer needed. Run the delete.py to delete the application.