

CLOUD ARCHITECTURE ASSGINMENT -1

DEEPANSHU

CLOUD ARCHITECTURE

DATE: 30 -10 -2024

Academic Integrity Declaration

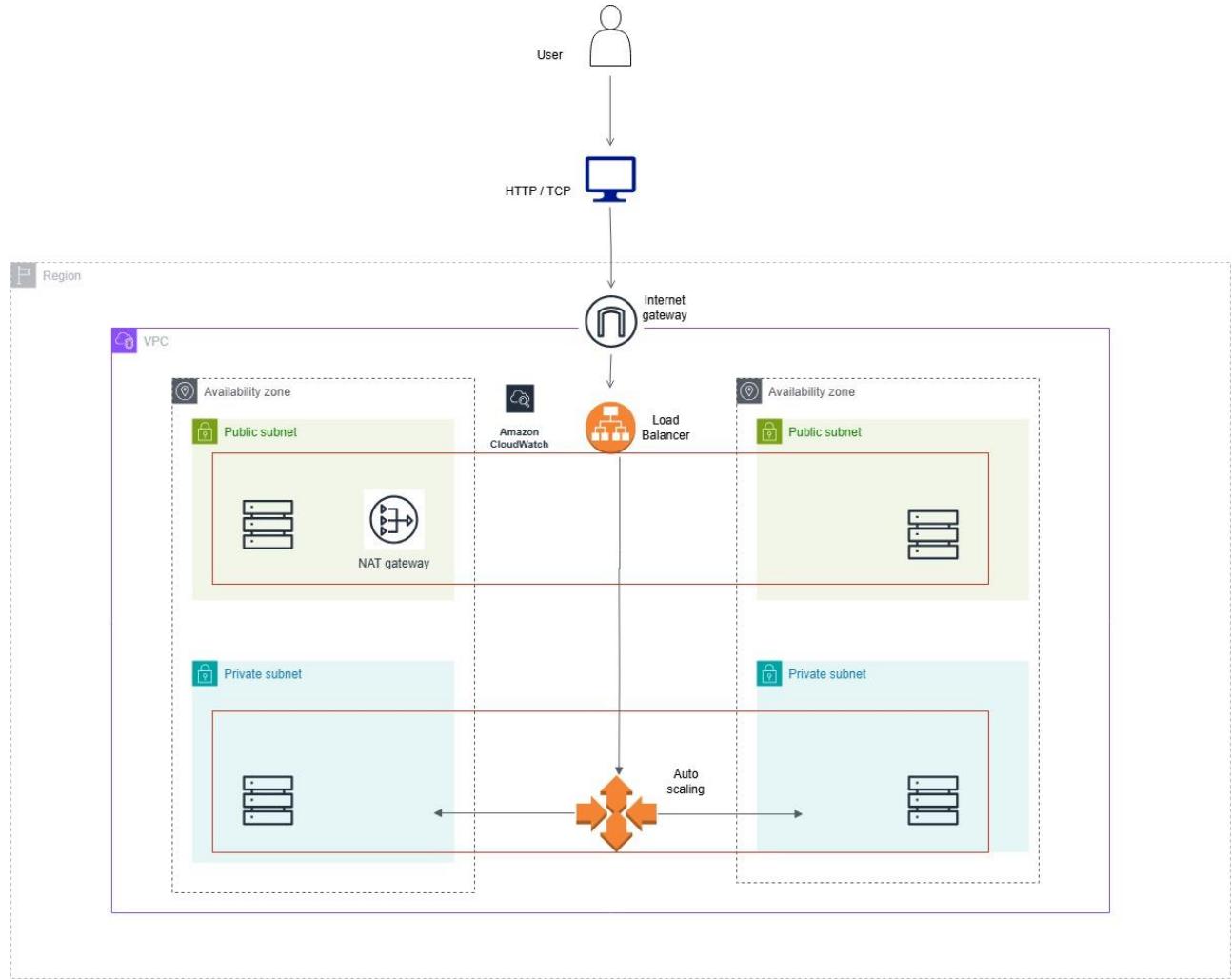
I certify that this assignment is all my own work and contains no plagiarism. By submitting this assignment, I agree to the following terms: Any text, diagrams or other material copied or generated from other sources (including, but not limited to, books, journals and the internet) have been clearly acknowledged and referenced as such in the text by the use of 'quotation marks' (or indented italics for longer quotations) followed by a reference to the original source either in the text or in a footnote/endnote. These details are then confirmed by a fuller reference in the bibliography. I have read the section on plagiarism in the SETU Academic Misconduct Policy (Appendix 1) and I understand that only assignments which are free of plagiarism will be awarded marks. I further understand that breaches of this policy can lead to sanctions up to the suspension or permanent expulsion of students in serious cases.

Signed: Deepanshu

TABLE OF CONTENTS

- Architecture
- Application
- VPC
- Load balancer
- Auto scaling
- Demonstration of load balancing in action using test traffic
- Custom metrics
- Cloud Watch
- Additional functionality 1
- Additional functionality 2
- Discussion/Conclusion
- References

Architecture diagram:



This is a fundamental illustration of my application running on the AWS console.

The user transmits a request to the application via the HTTP/TCP port using the web browser.

Internet gateway: Then the request goes into the internet gateway, which plays an important role in communication between your VPC and the internet.

VPC: The Virtual Private Cloud is the entire network where all your application infrastructure components happened.

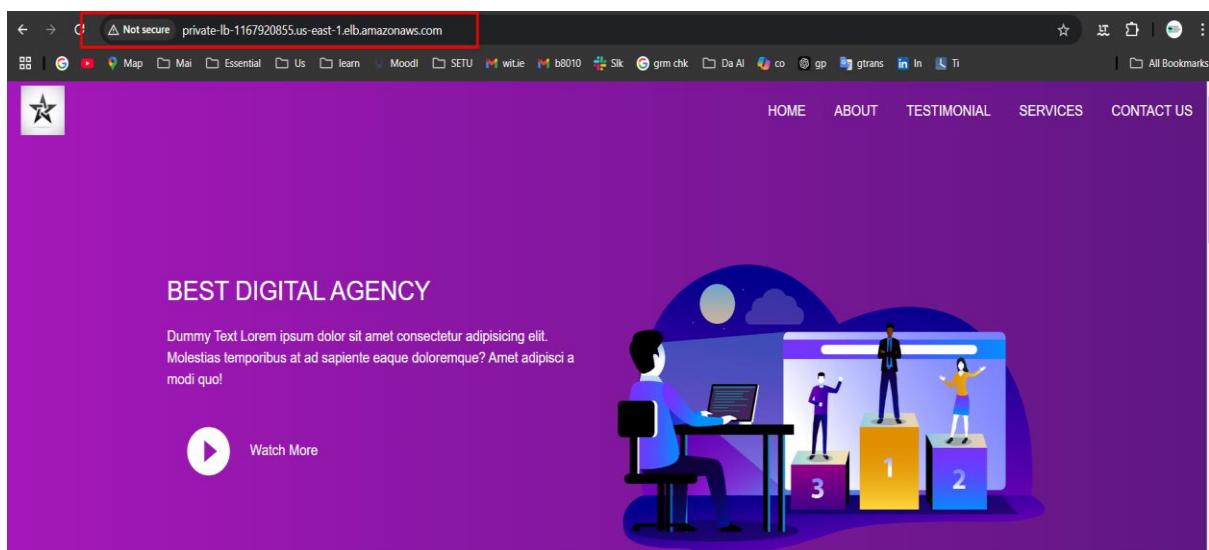
Load Balancer: After that, traffic comes into the load balancer and is distributed across multiple servers within the application according to your configuration. This ensures high availability with auto scaling group policy and manages application failovers.

Auto scaling: Inside that, we assign launch template AMI, this AMI of the EC2 instance where application have when traffic load come on. This AMI defines auto scaling policy rules to optimize and different availability zone and subnets so application handle traffic with the help of load balancer.

Nat Gateway: It provides access to your application resources from your public subnet to your private subnet.

Amazon Cloud Watch: It helps us to monitor our application and provide operational insights by tracking metrics, logging, and triggering alerts for specific conditions.

Application home page



WHAT WE DO ?

AMI

The screenshot shows the AWS CloudWatch Metrics console. A single metric named "AWS/CloudWatchMetrics" is displayed with a value of 1.0. The metric has a unit of "Count" and is associated with the namespace "AWS/CloudWatchMetrics". The time range is set from "Last hour".

VPC : Flow chat

The screenshot shows the AWS VPC console. The VPC dashboard lists two VPCs: "agency-vpc" and "Default". The "agency-vpc" VPC is selected. The "agency-vpc" row shows its VPC ID as "vpc-0a340e1da1570b577", status as "Available", IPv4 CIDR as "10.0.0.0/16", and IPv6 CIDR as "-". It also shows the DHCP option set as "dopt-0b6dcdb187e52ff96c", main route table as "rtb-01aa944a4451bf4aa", and association with "ad-C". The "Default" VPC is listed below it with its own details.

VPC details:

The screenshot shows the AWS VPC console interface. On the left, there's a sidebar with navigation links for EC2 Global View, Virtual private cloud, Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, and Peering connections. The main area displays a table titled "Your VPCs (1/2) Info".

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP
<input checked="" type="checkbox"/> agency-vpc	vpc-0a340e1da1570b577	Available	10.0.0.0/16	-	dopt
<input type="checkbox"/> Default	vpc-0971d191ee1359374	Available	172.31.0.0/16	-	dopt

Below the table, there are tabs for Details, Resource map, CIDRs, Flow logs, Tags, and Integrations. The Details tab is selected, showing the following configuration details for the 'agency-vpc' VPC:

- VPC ID: vpc-0a340e1da1570b577
- State: Available
- Tenancy: Default
- Default VPC: No
- Network Address Usage metrics: Disabled
- State: Available
- DNS hostnames: Enabled
- Main route table: rtb-01aa944a4431bf4aa
- IPv4 CIDR: 10.0.0.0/16
- IPv6 pool: -
- Route 53 Resolver DNS Firewall rule groups: Failed to load rule groups
- Owner ID: 425059748078
- DNS resolution: Enabled
- Main network ACL: acl-0836346f681519053
- IPv6 CIDR (Network border group): -

Route table

The screenshot shows the AWS VPC console interface. The sidebar includes the same navigation links as the previous screenshot. The main area displays a table titled "Route tables (4/6) Info".

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC
-	rtb-0730ea2c5f1fe520	-	-	Yes	vpc-0971d191ee1359374 D
-	rtb-01aa944a4431bf4aa	-	-	Yes	vpc-0a340e1da1570b577 a
<input checked="" type="checkbox"/> agency-rtb-public	rtb-0cc786f727a153f8e	3 subnets	-	No	vpc-0a340e1da1570b577 a
<input checked="" type="checkbox"/> agency-rtb-private3-us-east-1c	rtb-0b4c6ba19db4801c8	subnet-0508c04e6205e0...	-	No	vpc-0a340e1da1570b577 a
<input checked="" type="checkbox"/> agency-rtb-private2-us-east-1b	rtb-0f06103732aa6160d	subnet-0bb632d6aaab56a...	-	No	vpc-0a340e1da1570b577 a
<input checked="" type="checkbox"/> agency-rtb-private1-us-east-1a	rtb-0f011113499170732	subnet-04569f4eb421d...	-	No	vpc-0a340e1da1570b577 a

Below the table, it says "Route tables: rtb-0cc786f727a153f8e, rtb-0b4c6ba19db4801c8, rtb-0f06103732aa6160d, rtb-0f011113499170732". At the bottom right, there are links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Public route table

The screenshot shows the AWS VPC console interface. The left sidebar is titled "VPC dashboard" and includes sections for "Virtual private cloud" (Your VPCs, Subnets, Route tables), "Internet gateways", "Egress-only internet gateways", "Carrier gateways", "DHCP option sets", "Elastic IPs", "Managed prefix lists", "Endpoints", "Endpoint services", "NAT gateways", and "Peering connections". The main content area shows the details of a route table named "rtb-0cc786f727a153f8e / agency-rtb-public". The "Details" tab is selected, displaying information such as Route table ID (rtb-0cc786f727a153f8e), Main (No), Owner ID (vpc-0a340e1da1570b577 | agency-vpc), and Explicit subnet associations (3 subnets). Below this, the "Subnet associations" tab is selected, showing a table of explicit subnet associations:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
agency-subnet-public1-us-east-1a	subnet-02317b8f8f59ebd6d	10.0.0.0/20	-
agency-subnet-public3-us-east-1c	subnet-09235e69d237add96c	10.0.32.0/20	-
agency-subnet-public2-us-east-1b	subnet-0bb371706e59bf54	10.0.16.0/20	-

Private us-east- 1a

The screenshot shows the AWS VPC console interface. The left sidebar is titled "VPC dashboard" and includes sections for "Virtual private cloud" (Your VPCs, Subnets, Route tables), "Internet gateways", "Egress-only internet gateways", "Carrier gateways", "DHCP option sets", "Elastic IPs", "Managed prefix lists", "Endpoints", "Endpoint services", "NAT gateways", and "Peering connections". The main content area shows the details of a route table named "rtb-0f011113499170732 / agency-rtb-private1-us-east-1a". The "Details" tab is selected, displaying information such as Route table ID (rtb-0f011113499170732), Main (No), Owner ID (vpc-0a340e1da1570b577 | agency-vpc), and Explicit subnet associations (2 subnets). Below this, the "Routes" tab is selected, showing a table of routes:

Destination	Target	Status	Propagated
pl-63a5400a	vpc-03695f016699d94b6	Active	No
0.0.0.0/0	nat-05415e1859a2cf8c9	Active	No
10.0.0.0/16	local	Active	No

Private us-east-1b

The screenshot shows the AWS VPC Route Table Details page for route table ID rtb-0f06103732aa6160d. The main details section shows the route table ID, VPC, and explicit subnet associations. The routes section lists three routes:

Destination	Target	Status	Propagated
pl-65a5400a	vpce-03695f016699d94b6	Active	No
0.0.0.0/0	nat-05415e1859a2cf8c9	Active	No
10.0.0.0/16	local	Active	No

Private us-east-1c

The screenshot shows the AWS VPC Route Table Details page for route table ID rtb-0b4c6ba19db4801c8. The main details section shows the route table ID, VPC, and explicit subnet associations. The routes section lists three routes:

Destination	Target	Status	Propagated
pl-65a5400a	vpce-03695f016699d94b6	Active	No
0.0.0.0/0	nat-05415e1859a2cf8c9	Active	No
10.0.0.0/16	local	Active	No

Security group

Private

The screenshot shows the AWS VPC console for a security group named "DBServerSG". The "Details" section displays the following information:

Security group name	Security group ID	Description	VPC ID
DBServerSG	sg-0035ebba26e7c1507	test private	vpc-0a340e1da1570b572
Owner	425059748078	Inbound rules count 2 Permission entries	Outbound rules count 3 Permission entries

The "Inbound rules" tab is selected, showing two rules:

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
-	sgr-0a34fb5147d5956...	IPv4	HTTP	TCP	80	0.0.0.0/0
-	sgr-0745e7d45e005d...	IPv4	SSH	TCP	22	0.0.0.0/0

Public

The screenshot shows the AWS VPC console for a security group named "WebServerSG". The "Details" section displays the following information:

Security group name	Security group ID	Description	VPC ID
WebServerSG	sg-0b23db30280b501f9	test public	vpc-0a340e1da1570b572
Owner	425059748078	Inbound rules count 3 Permission entries	Outbound rules count 2 Permission entries

The "Inbound rules" tab is selected, showing three rules:

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0cd424526e7532cc7	IPv4	HTTPS	TCP	443	0.0.0.0/0	-
-	sgr-0d4ff20f40ec9881	IPv4	SSH	TCP	22	0.0.0.0/0	-
-	sgr-033d06e0825d4f2...	IPv4	HTTP	TCP	80	0.0.0.0/0	-

Load Balancer

The screenshot shows the AWS Cloud Console with the EC2 service selected. In the left navigation pane, under the Load Balancing section, 'Load Balancers' is selected. The main content area displays a table titled 'Load balancers (1/1)'. A single row is listed for a load balancer named 'Private-lb'. The details shown include its ARN, DNS name (Private-lb-1167920855.us-east-1.elb.amazonaws.com), port (80), protocol (HTTP), target type (Instance), and the associated load balancer (Private-lb). The table also shows its state as Active and its VPC ID as vpc-0a340e1da1570b... with 3 Availability Zones.

Target group showing registered targets

The screenshot shows the AWS Cloud Console with the EC2 service selected. In the left navigation pane, under the Load Balancing section, 'Target Groups' is selected. The main content area displays a table titled 'Target groups (1/1)'. A single row is listed for a target group named 'private-tg'. The details shown include its ARN, port (80), protocol (HTTP), target type (Instance), and the associated load balancer (Private-lb). The table also shows its state as healthy.

Auto Scaling Group

The screenshot shows the AWS EC2 Auto Scaling Groups page. On the left, a sidebar navigation includes: 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), and Auto Scaling (Auto Scaling Groups). The main content area displays the 'private-asg' Auto Scaling group. The 'Details' tab is selected, showing the following configuration:

Auto Scaling group name	Desired capacity	Desired capacity type	Amazon Resource Name (ARN)
private-asg	1	Units (number of instances)	arn:aws:autoscaling:us-east-1:425059748078:autoScalingGroup:2df0f30be-30ce-414a-bca1-f0fb48c1fb46:autoScalingGroupName/private-asg
Date created	Minimum capacity	Status	
Mon Oct 28 2024 20:07:56 GMT+0000 (Greenwich Mean Time)	1	-	
	Maximum capacity		
	3		

Below the 'Details' tab, there is a 'Launch template' section with a table:

Launch template	AMI ID	Instance type	Owner

Scaling policies

The screenshot shows the AWS EC2 Dynamic scaling policies page. The sidebar navigation is identical to the previous screenshot. The main content area displays two scaling policies:

- High CPU alarm**:
 - Policy type: Simple scaling
 - Enabled or disabled: Enabled
 - Execute policy when: High CPU alarm
 - breaches the alarm threshold: CPUUtilization > 40 for 2 consecutive periods of 60 seconds for the metric dimensions:
AutoScalingGroupName = private-asg
 - Take the action:
 - Add 1 capacity units
 - And then wait:
30 seconds before allowing another scaling activity
- Low CPU alarm**:
 - Policy type: Simple scaling
 - Enabled or disabled: Enabled
 - Execute policy when: Low CPU alarm
 - breaches the alarm threshold: CPUUtilization <= 40 for 2 consecutive periods of 60 seconds for the metric dimensions:
AutoScalingGroupName = private-asg
 - Take the action:
 - Remove 1 capacity units
 - And then wait:
30 seconds before allowing another scaling activity

After running **while** loop and **top** command auto scale instance trigger

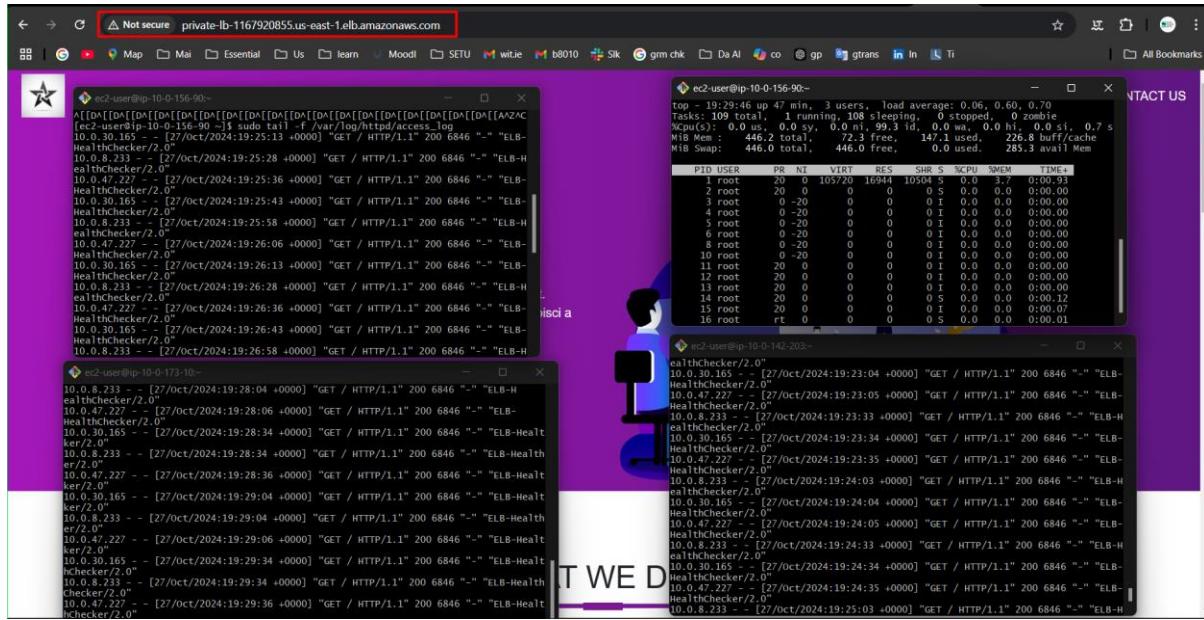
The screenshot shows the AWS EC2 Instances page. The left sidebar includes options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, and Reservations. The main content area displays a table of instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Five instances are listed, all labeled 'Autoscale'. Three instances are selected, indicated by a checked checkbox in the first column. The table header shows 'Instances (3/5) Info' and sorting by Name. A 'Monitoring' tab is open at the bottom, showing CPU utilization (%), Network in (bytes), Network out (bytes), and Network packets in (count). The status bar at the bottom right indicates '© 2024, Amazon Web Services, Inc. or its affiliates.'

Load balancer working

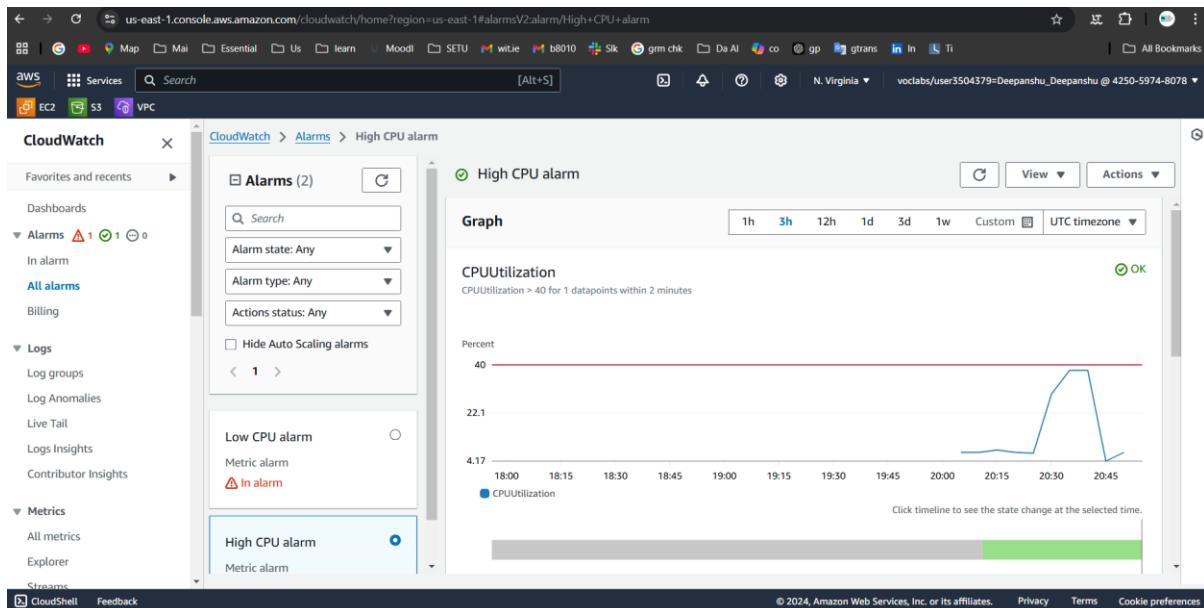
The screenshot shows a web browser displaying a landing page. The URL in the address bar is 'private-lb-116792085.us-east-1.elb.amazonaws.com'. The page has a purple background and features a large graphic on the right showing three people standing on podiums (1st, 2nd, 3rd place) in front of a laptop screen. To the left of the graphic, there is text: 'BEST DIGITAL AGENCY' and 'Dummy Text Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestias temporibus at ad sapiente eaque doloremque? Amet adipisci a modi quo!'. Below this text is a play button icon with the text 'Watch More'.

WHAT WE DO ?

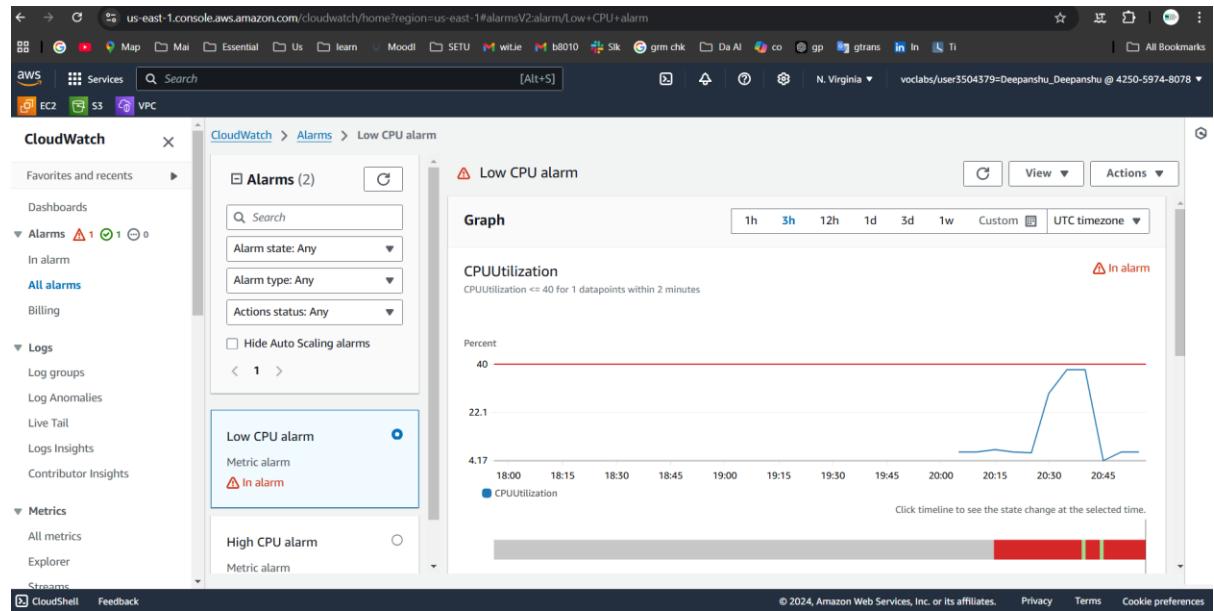
Load balancer working action using test traffic on each auto scale instance trigger.



CloudWatch High CPU alarm

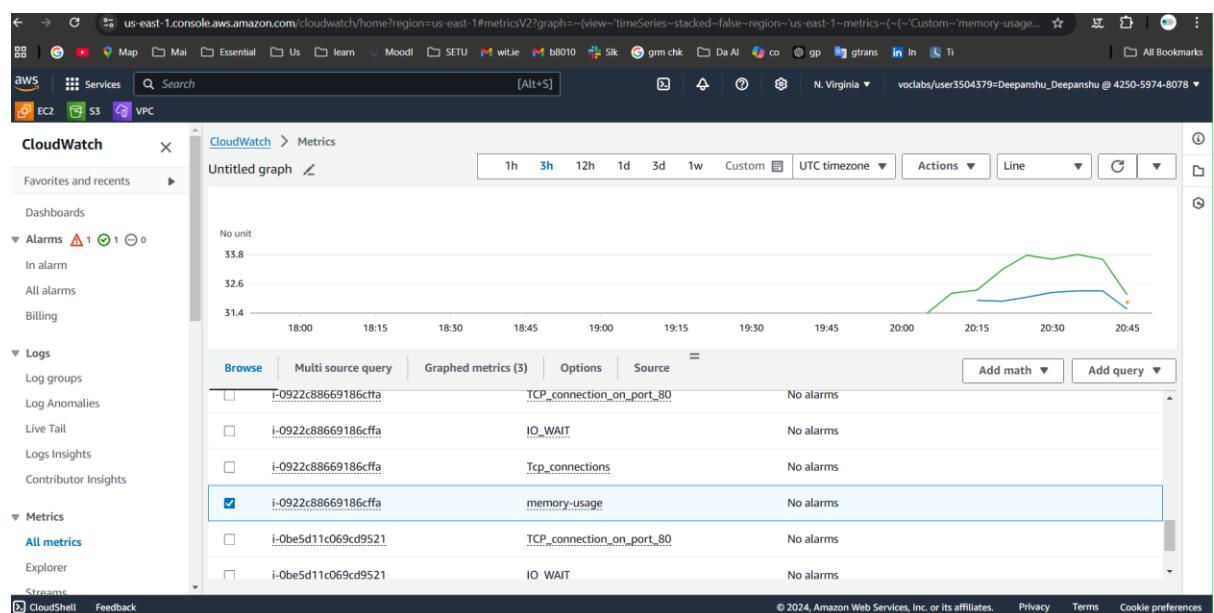


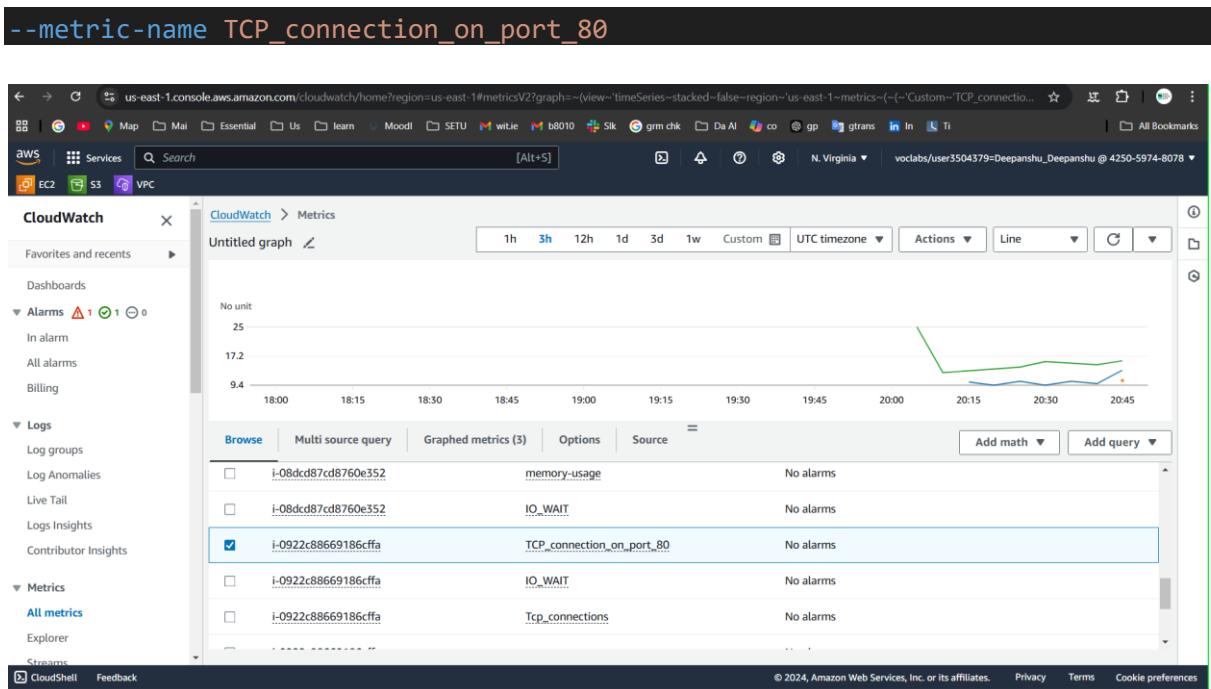
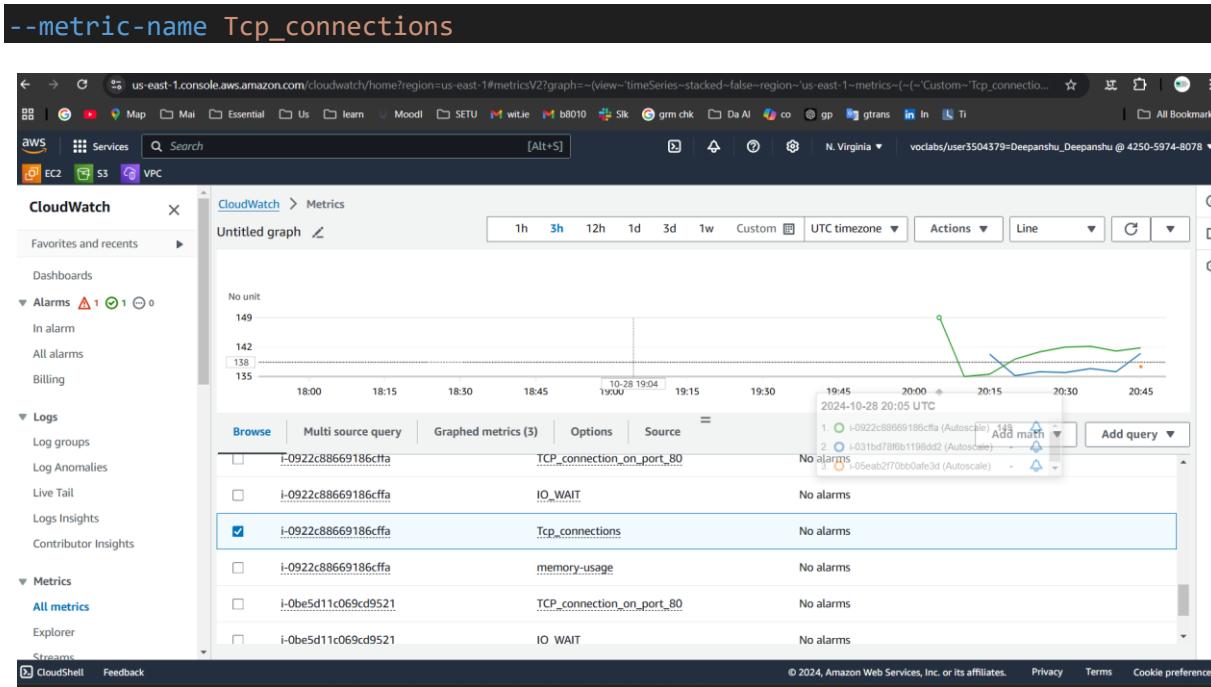
CloudWatch Low CPU alarm

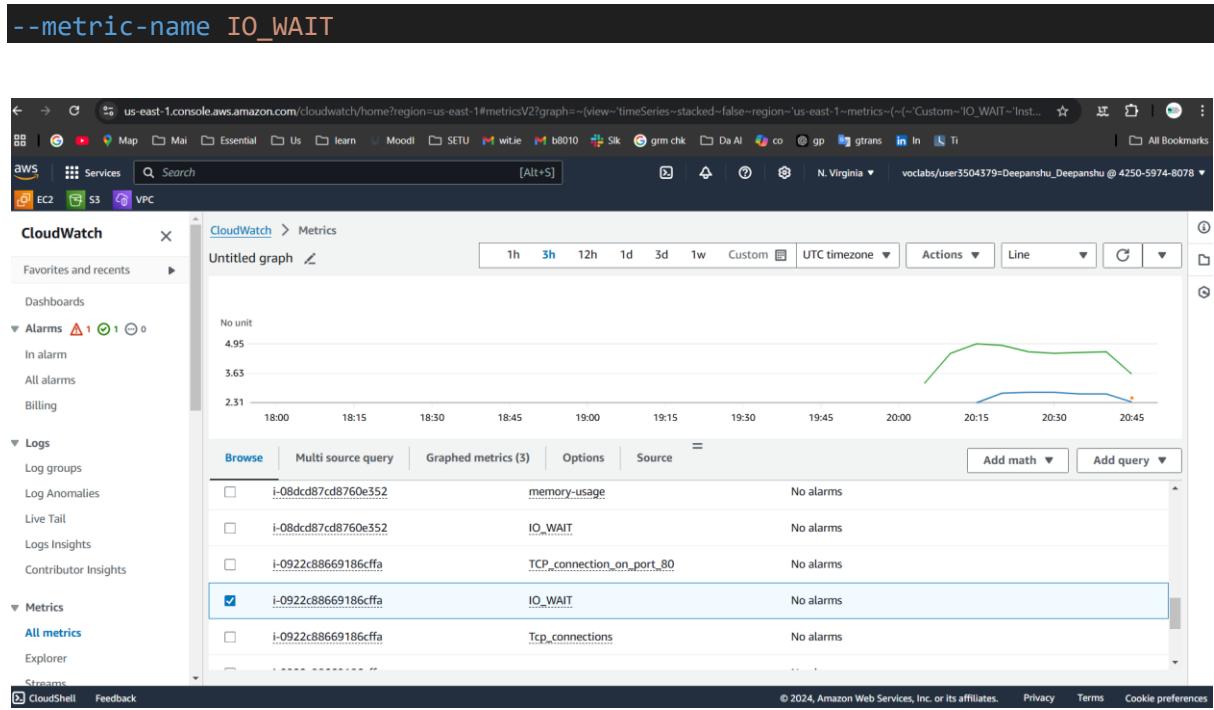


CloudWatch custom metrics data of auto scale instance:

```
--metric-name memory-usage
```







Additional functionality 1

AWS lambda: I integrated the AWS lambda function into my architecture because it provides a powerful way to run code in response to an event or request without having to manage servers. It works for event driven architectures and serverless applications.

```

lambda_function.py
1 import boto3
2
3 ec2 = boto3.client('ec2')
4
5 # Stop Instances
6 ec2.stop_instances(InstanceIds=['i-090c8773e38f6c77'])
7 print("Instance Stopped")
8

```

The screenshot shows the AWS Lambda console. The top bar indicates that the function 'Automate-ec2' has been successfully updated. The main area is a code editor for 'lambda_function.py'. The code is a simple Python script that uses the AWS SDK (boto3) to stop an EC2 instance. A red arrow points to the line of code that stops the instance. The interface includes an 'EXPLORER' sidebar with 'AUTOMATE-EC2' and 'lambda_function.py' listed, and a 'DEPLOY' section with 'Deploy' and 'Test' buttons. At the bottom, there are tabs for 'PROBLEMS', 'OUTPUT', 'CODE REFERENCE LOG', and 'TERMINAL', along with an 'Execution Results' dropdown.

After running test code on the lambda function, it's stopped instance

The screenshot shows the AWS EC2 Instances page. The left sidebar has sections for EC2 Dashboard, EC2 Global View, Events, Instances (with a sub-section for Instances), Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, and Reservations. The main content area shows a table titled 'Instances (1/7) Info' with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. One row is selected, showing 'Autoscale' with Instance ID i-090c87773e38f6c77, which is listed as 'Terminated'. A red box highlights the 'Instance ID' column for this row. Below the table, there's a detailed view for 'i-090c87773e38f6c77 (Autoscale)' with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The 'Details' tab is active, showing the Instance ID i-090c87773e38f6c77.

I used the aws lambda function to manage the EC2 instances because we interact daily with the EC2 instance in our console. If we have a large number of EC2 instances, stopping manually can be challenging. I found python boto3 documentation on the boto3.amazonaws.com website to stop any ec2 instance.

Additional functionality 2:

The **AWS Key Management Service (KMS)** is a security service that facilitates the creation and management of encryption keys to safeguard your data. We can provide a KMS service to encrypt data in S3 buckets, databases, or while transferring between Amazon Web Services.

Create AWS KMS encryption key:

The screenshot shows the AWS KMS Key configuration page for the key '6f88892a-745c-4759-a2f4-a970bc736e67'. The top navigation bar includes AWS, Services, and VPC. The main section is 'General configuration' with fields for Alias (Permission-key), Status (Enabled), Creation date (Oct 29, 2024 17:39 GMT), ARN (arn:aws:kms:us-east-1:425059748078:key/6f88892a-745c-4759-a2f4-a970bc736e67), Description (Enhance security), and Regionality (Single Region). A red arrow points to the ARN field. Below this are tabs for Key policy, Cryptographic configuration, Tags, Key rotation, and Aliases. The 'Key policy' tab is active, showing 'Key administrators (1)' with a table for 'Search Key administrators'. A red arrow points to the 'LabRole' entry in the table.

By default s3 bucket basic key comes so we have to edit and update KMS key inside the bucket permission tab.

The screenshot shows the AWS S3 Bucket Properties page for a bucket named 'mywebapplication-agency'. In the 'Default encryption' section, there is a table with one row. The 'Encryption key ARN' column contains the value 'arn:aws:kms:us-east-1:425059748078:key/6f88892a-745c-4759-a2f4-a970bc736e67'. A red arrow points to this value. Below the table, there is a section titled 'Intelligent-Tiering Archive configurations (0)' with a table showing no configurations.

Define bucket policy who can access this file which account .

The screenshot shows the 'Edit bucket policy' page for the same bucket. The policy JSON is displayed:

```
1 ▼ {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Principal": "arn:aws:iam::425059748078" //Account id
7     },
8   ],
9   "Action": "s3:GetObject",
10  "Resource": "arn:aws:s3:::mywebapplication-agency" //Bucket name
11 }
12 }
13 }
```

A red arrow points to the 'Principal' field in the JSON code, highlighting the account ID '425059748078'. The right side of the screen shows a 'Select a statement' interface with a 'Policy generator' button and a 'Policy examples' button.

I use the AWS Key Management service to secure my S3 bucket when I want to share a file with an account. I want to know who got this file and how they access it. The owner can manage key policies, including who can use the encryption keys and for what purpose. When we compare S3 keys, they do not provide this functionality.

Discussion/Conclusion:

I have created a static HTML website and deployed it on a private server. Therefore, migrating to a private server proved to be challenging. Initially, I uploaded the website to the public instance, then transferred the application to the private EC2 instance on /var/www/html, and finally, I created a custom AMI to utilize an auto scaling group. In this deployment, I created a VPC architecture with public and private subnets, since private application server resource can be accessible from publicly.

I set up load balancer and scaling policies triggered by CloudWatch alarms, because they help us scale our application and safeguard against failover. It ensures that the architecture is capable of automatically scaling up or down based on demand. I also tested load balancing to confirm that traffic was evenly distributed across multiple auto-scale instances that were created in the EC2 instance console.

Security Consideration:

In this Assignment we deployed this application into private subnet to limit direct internet access.

We have first deployed the application on public, and then we have to move the application into private.

Security group: We have created two security groups, WebServerSG for public traffic, and DBServerSG for private traffic. In private security group inbound permissions are less than public security groups.

Custom Watch : The Mem.sh file which contains customized metric parameters, was uploaded on a private subnet and an AMI was created to monitor memory usage, TCP port 80 utilization, CPU utilization, and other related metrics within the cloud metric column within the cloud watch.

Encryption: I use AWS Key Management service to secure sensitive data when we share it, so we can control which account they can use that file from and we have log's of that file in cloud trails.

AWS lambda: We do manually stop or terminate any instance, so I use aws lambda and I created a function we have just run this function and our instance ID is stopped. That instance ID is put in the function.

References:

AWS lambda: Stop instance

https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2/client/stop_instances.html

AWS Key Management services

<https://docs.aws.amazon.com/kms/latest/developerguide/key-policy-overview.html>

AWS S3 bucket policies

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html>