

A Project Report On

TOPIC

Creating a Highly Available Architecture



Submitted By

Name: Kiran K V

Batch: Certified AWS Cloud Practitioner

Batch Code: 2024-11761 Bangalore-CG AWS CP

Enrollment ID: EBEON0424902026

Email: kvkirankumar1201@gmail.com

Mob: +91 7899283024

ACKNOWLEDGEMENT

I would also like to extend my heartfelt thanks to my mentor, **Mr. Prasiddh Saxena**. His invaluable guidance, insightful advice, and constant encouragement have been pivotal throughout this journey. His expertise and mentorship have not only enhanced my understanding of cloud technologies but also inspired me to strive for excellence.

I would also like to extend my thanks to my peers and instructors in the **Certified AWS Cloud Practitioner** course, **Batch 2024-11761 in Bangalore**. The collaborative learning environment and the shared knowledge among batch mates have greatly enhanced my understanding and application of cloud computing concepts.

Special thanks to the administrative staff and technical support team for their assistance in providing the necessary resources and infrastructure to carry out this project.

Kiran K V

Batch Code: 2024-11761 Bangalore

Enrollment ID: EBEON0424902026

Email: kvkirankumar1201@gmail.com

Mob: +91 7899283024

Table of Contents.

1. Introduction.....
2. Architecture.....
3. Execution.....
4. Clean-Up.....
5. Conclusion.....

Chapter 1

INTRODUCTION

Amazon Web Services (AWS) is a comprehensive and widely adopted cloud platform offering over 200 fully-featured services from data centers globally. It provides on-demand cloud computing resources and APIs to individuals, companies, and governments on a metered pay-as-you-go basis. AWS services range from computing, storage, and databases to machine learning, analytics, and IoT. Its flexible infrastructure enables businesses to scale efficiently and deploy applications worldwide with high availability and low latency. AWS also emphasizes robust security measures, compliance certifications, and data protection. With its extensive ecosystem, AWS supports various industries, fostering innovation and transformation.

The primary objective of this project is to create an automated, highly available web application infrastructure that can dynamically scale based on traffic demands. The project encompasses the following key components:

1. **AWS VPC (Virtual Private Cloud):** Amazon Virtual Private Cloud (VPC) is a service that allows you to launch AWS resources in a logically isolated virtual network that you define. It gives you complete control over your virtual networking environment, including resource placement, connectivity, and security.
2. **AWS Simple Notification Service (SNS):** Used for setting up notifications to monitor and manage our infrastructure. SNS provides a reliable way to send alerts for different events within our application, ensuring we stay informed about the system's health and performance.
3. **Custom Amazon Machine Image (AMI):** AMIs are essential for creating pre-configured environments that can be replicated across multiple instances. This ensures consistency and speeds up the deployment process.
4. **Amazon EC2 Instances:** These virtual servers provide the computing power needed for our application. We will launch and configure EC2 instances to host our web application.
5. **Elastic Load Balancer (ELB):** A load balancer is set up to distribute incoming traffic across multiple EC2 instances, ensuring no single instance is overwhelmed and improving the application's availability and fault tolerance.
6. **Security Groups:** Configuring security groups to manage inbound and outbound traffic to ensure our instances are secure and accessible only through predefined rules.

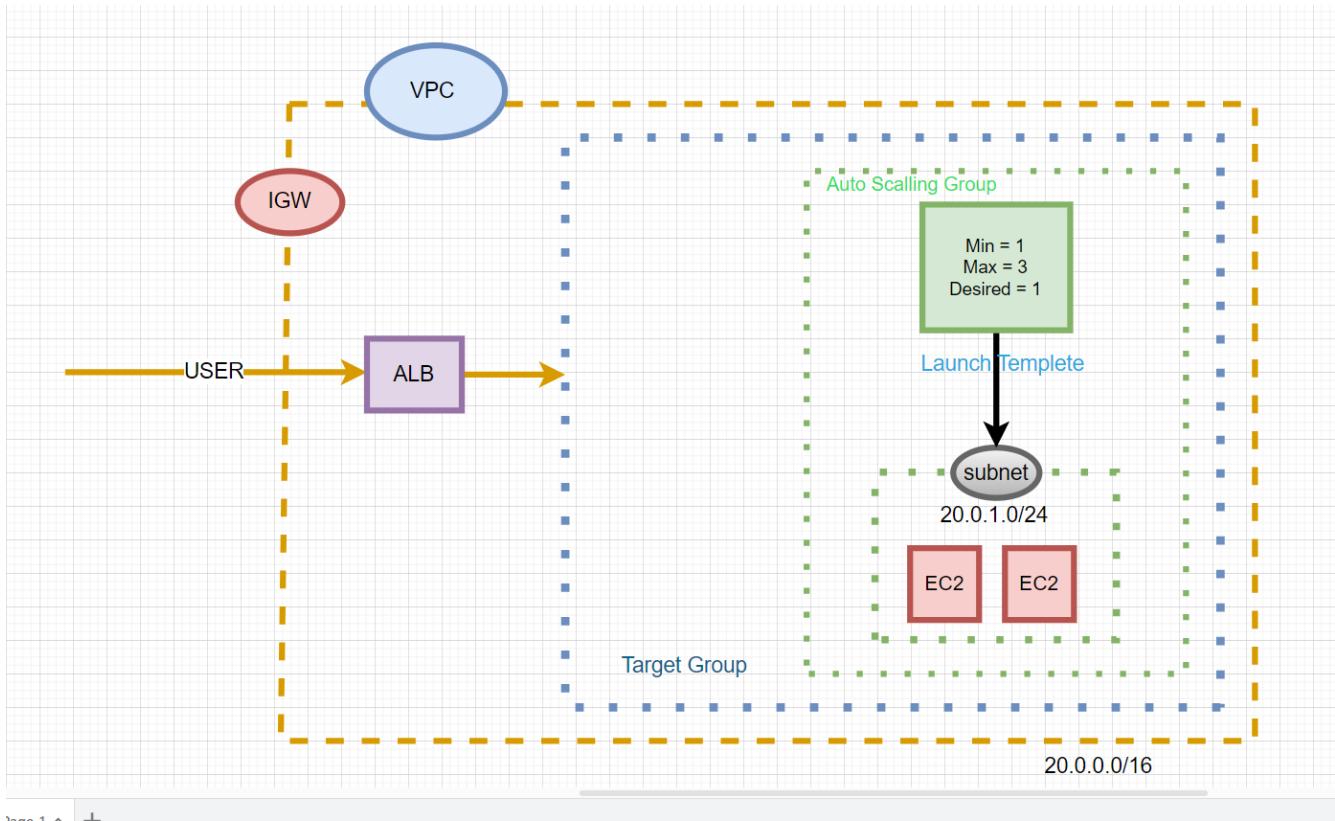
7. **Auto Scaling Groups:** These are configured to automatically adjust the number of EC2 instances based on the current demand. This ensures that the application can handle varying loads efficiently, scaling up during peak times and scaling down when demand is low.
8. **AWS Cloud Watch:** Used for monitoring the infrastructure, setting up alarms, and triggering scaling actions. Cloud Watch provides detailed insights into the system's performance and operational health, enabling proactive management.

The following sections of the report provide a step-by-step guide to implementing each component, starting with setting up virtual private cloud, notifications using SNS, creating a custom AMI, launching and configuring EC2 instances, and setting up security groups, load balancers, auto-scaling groups, and Cloud Watch alarms. By the end of this project, we will have a resilient, scalable, and automated web hosting solution capable of handling fluctuating workloads while maintaining high availability and performance.

Chapter 2

ARCHITECTURE

N.Virginia



Words used in naming conventions:-

VPC – Mproject-Test-VPC (Virtual private cloud)

IGW – Mproject-Test-IGW (Internet gateway)

Subnet - Mproject-test-public-subnet-1a

Simple notification service - Mproject-Test-SNS

AMI- Mproject-Test-AMI

ALb – Mproject-Test-ALB

EC2 – Mproject-Test-Ec2-Instance

ASG - Mproject-Test-AutoScallingGroup

Target Group – Homepage

Alarms = Scal-In, Scal-Out

Chapter 3

EXECUTION

Step 1: VPC configuration.

- a. Once log-in to the AWS Console, click **Services -> Network & Content Delivery -> VPC**.
 - Select resource - VPC only
 - Name- Mproject-Test-VPC
 - Enable Ipv4 manual input,
 - Ipv4 CIDR – 20.0.0.0/16

The screenshot shows the 'Create VPC' wizard. In the 'VPC settings' section, 'Resources to create' is set to 'VPC only'. The 'Name tag' is 'Mproject-Test-VPC'. The 'IPv4 CIDR block' is '20.0.0.0/16'. Under 'IPv6 CIDR block', 'No IPv6 CIDR block' is selected. The 'Tenancy' is 'Default'. A note at the top states: 'A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.'

- Next Create VPC.

The screenshot shows the AWS VPC console for the newly created VPC 'vpc-019e97407464ae05f / Mproject-Test-VPC'. The 'Details' tab is selected, displaying information like VPC ID, State (Available), and DNS resolution (Enabled). The 'Resource map' tab is also visible, showing the VPC's connection to other networks.

- b. Create Internet gateway and attaching it into the VPC, [Services -> Network & Content Delivery -> Internet Gateways](#).

- Name - Mproject-Test-IGW

VPC > [Internet gateways](#) > Create internet gateway

Create internet gateway Info

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings	
Name tag Creates a tag with a key of 'Name' and a value that you specify. <input type="text" value="Mproject-Test-IGW"/>	

Tags - optional	
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.	
Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="Mproject-Test-IGW"/> X
Add new tag	
You can add 49 more tags.	

Cancel Create internet gateway

- Created Internet gateway.

The following internet gateway was created: igw-00235236597fd9134 - Mproject-Test-IGW. You can now attach to a VPC to enable the VPC to communicate with the internet.

Attach to a VPC X

VPC > [Internet gateways](#) > igw-00235236597fd9134

igw-00235236597fd9134 / Mproject-Test-IGW

Actions ▾

Details <small>Info</small>			
Internet gateway ID igw-00235236597fd9134	State Detached	VPC ID -	Owner 905418258809

Tags	
<input type="text" value="Search tags"/> Manage tags	
Key	Value
Name	Mproject-Test-IGW

- Attached to the VPC. Click on [Actions-> attach VPC](#) -> select VPC -> attach

Internet gateway igw-00235236597fd9134 successfully attached to vpc-019e97407464ae05f

VPC > [Internet gateways](#) > igw-00235236597fd9134

igw-00235236597fd9134 / Mproject-Test-IGW

Actions ▾

Details <small>Info</small>			
Internet gateway ID igw-00235236597fd9134	State Attached	VPC ID vpc-019e97407464ae05f Mproject-Test-VPC	Owner 905418258809

Tags	
<input type="text" value="Search tags"/> Manage tags	
Key	Value
Name	Mproject-Test-IGW

c. Creating subnets, Click Services -> Network & Content Delivery -> Subnets->create subnet.

- Select VPC – Mproject-Test-VPC
- Subnet name - Mproject-test-public-subnet-1a
- Ipv4 subnet CIDR block – 20.0.1.0/24

You have successfully created 2 subnets: subnet-0fdfd019a125ef709, subnet-0168c4ddfca9f9921

Subnets (1/1) Info

Find resources by attribute or tag

Subnet ID : subnet-0fdfd019a125ef709 | Clear filters

Last updated less than a minute ago Actions Create subnet

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR
Mproject-test-public-subnet-1a	subnet-0fdfd019a125ef709	Available	vpc-019e97407464ae05f Mpr...	20.0.1.0/24	-

subnet-0fdfd019a125ef709 / Mproject-test-public-subnet-1a

Details

Subnet ID subnet-0fdfd019a125ef709	Subnet ARN arn:aws:ec2:us-east-1:905418258809:subnet/subnet-0fdfd019a125ef709	State Available	IPv4 CIDR 20.0.1.0/24
Available IPv4 addresses 251	Availability Zone us-east-1a	Route table	Availability Zone ID use1-az1
Network border group us-east-1	IPv6 CIDR -	Auto-assign IPv6 address No	Network ACL
Default subnet No	VPC vpc-019e97407464ae05f Mproject-Test-VPC	Auto-assign public IPv4 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool	Auto-assign public IPv4 address No	IPv4 CIDR reservations	IPv6 CIDR reservations

- Subnet name – Mproject-test-public-subnet-1b
- Ipv4 subnet CIDR block – 20.0.3.0/24

subnet-0168c4ddfca9f9921 / Mproject-test-public-subnet-1b

Available

vpc-019e97407464ae05f | Mpr... 20.0.3.0/24

Details Flow logs Route table Network ACL CIDR reservations Sharing Tags

Details

Subnet ID subnet-0168c4ddfca9f9921	Subnet ARN arn:aws:ec2:us-east-1:905418258809:subnet/subnet-0168c4ddfca9f9921	State Available	IPv4 CIDR 20.0.3.0/24
Available IPv4 addresses 251	Availability Zone us-east-1b	Route table	Availability Zone ID use1-az2
Network border group us-east-1	IPv6 CIDR -	Auto-assign IPv6 address No	Network ACL

d. Create Rout table, , Click Services -> Network & Content Delivery -> Route Tables ->create Route table.

- Name – Mpproject-Test-RT
- Select VPC – Mproject-Test-VPC

The screenshot shows the AWS VPC Route Tables page. A success message at the top says "Route table rtb-0c82e12defd93094a | Mpproject-Test-RT was created successfully." Below this, the breadcrumb navigation shows "VPC > Route tables > rtb-0c82e12defd93094a". The main title is "rtb-0c82e12defd93094a / Mpproject-Test-RT". On the right, there is an "Actions" dropdown menu. The "Details" tab is selected, showing the following information:

Route table ID	rtb-0c82e12defd93094a	Main	No	Explicit subnet associations	-	Edge associations	-
VPC	vpc-019e97407464ae05f Mproject-Test-VPC	Owner ID	905418258809				

Below the details, there are tabs for "Routes", "Subnet associations", "Edge associations", "Route propagation", and "Tags". The "Routes" tab is selected, showing one route entry:

Routes (1)		Both		Edit routes	
Filter routes		< 1 >		⚙️	
Destination	Target	Status	Propagated		
20.0.0.0/16	local	Active	No		

- Now associate with public subnets to routable. Click Subnet association -> edit subnet association -> select both the subnets -> save the association.

The screenshot shows the AWS VPC Route Tables page. A success message at the top says "You have successfully updated subnet associations for rtb-0c82e12defd93094a / Mpproject-Test-RT .". Below this, the breadcrumb navigation shows "VPC > Route tables > rtb-0c82e12defd93094a". The main title is "rtb-0c82e12defd93094a / Mpproject-Test-RT". On the right, there is an "Actions" dropdown menu. The "Details" tab is selected, showing the following information:

Route table ID	rtb-0c82e12defd93094a	Main	No	Explicit subnet associations	2 subnets	Edge associations	-
VPC	vpc-019e97407464ae05f Mproject-Test-VPC	Owner ID	905418258809				

Below the details, there are tabs for "Routes", "Subnet associations", "Edge associations", "Route propagation", and "Tags". The "Subnet associations" tab is selected, showing the updated subnet associations:

Routes (1)		Both		Edit routes	
Filter routes		< 1 >		⚙️	
Destination	Target	Status	Propagated		
20.0.0.0/16	local	Active	No		

- This route table also provide the internet access, add the internet route Click route -> edit route -> add route -> select 0.0.0.0/0 for internet access and target type is internet gateway select -> Mproject-Test-IGW -> save the changes.

VPC > Route tables > rtb-0c82e12defd93094a > Edit routes

Edit routes

Destination	Target	Status	Propagated
20.0.0.0/16	local	Active	No
<input type="text" value="Q_ 0.0.0.0"/>	<input type="text" value="local"/>	<input checked="" type="checkbox"/>	
<input type="text" value="Q_ 0.0.0.0/0"/>	<input type="text" value="Internet Gateway"/>	-	No
	<input type="text" value="Q_ igw-00235236597fd9134"/>	<input checked="" type="checkbox"/>	<input type="button" value="Remove"/>
<input type="button" value="Add route"/>			
<input type="button" value="Cancel"/> <input type="button" value="Preview"/> <input type="button" value="Save changes"/>			

⌚ Updated routes for rtb-0c82e12defd93094a / Mpproject-Test-RT successfully

VPC > Route tables > rtb-0c82e12defd93094a

rtb-0c82e12defd93094a / Mpproject-Test-RT

Details	Info
Route table ID <input type="text" value="rtb-0c82e12defd93094a"/>	Main <input type="checkbox"/> No
VPC <input type="text" value="vpc-019e97407464ae05f Mproject-Test-VPC"/>	Owner ID <input type="text" value="905418258809"/>
Explicit subnet associations <u>2 subnets</u>	
Edge associations -	

Routes (2)				
<input type="button" value="Both"/> <input type="button" value="Edit routes"/> <input type="text" value="Filter routes"/> <input type="button" value="Filter routes"/> <input type="button" value="Clear filters"/>				
Destination	Target	Status	Propagated	
0.0.0.0/0	<input type="text" value="igw-00235236597fd9134"/>	<input checked="" type="checkbox"/> Active	No	<input type="button" value="Edit"/>
20.0.0.0/16	local	<input checked="" type="checkbox"/> Active	No	<input type="button" value="Edit"/>

Step 2: AWS Simple Notification Service configuration.

- a. Click **Services -> Application Integration -> Simple Notification Service**.
 - Create Topic name – Mproject-Test-SNS
 - Type – Standard
 - Display Name - Mproject-Test-SNS

The screenshot shows the AWS SNS 'Topics' page. A green banner at the top indicates that the topic has been created successfully. Below the banner, the topic name 'Mproject-Test-SNS' is displayed with options to 'Edit', 'Delete', or 'Publish message'. The 'Details' section shows the following information:

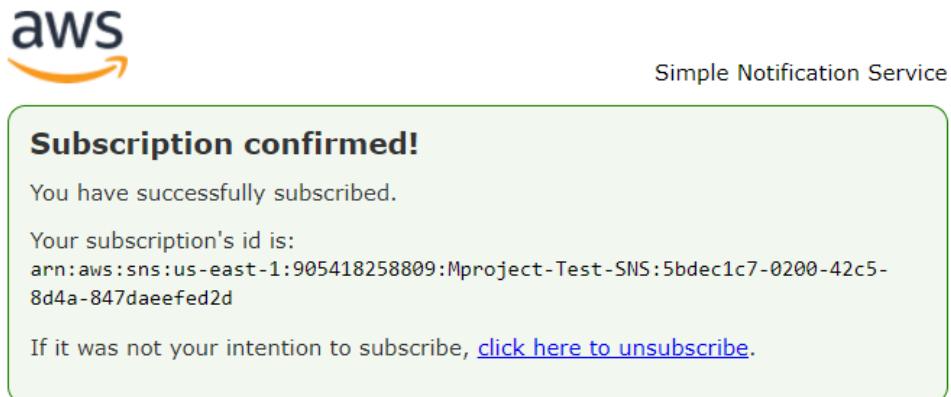
Name	Mproject-Test-SNS	Display name	Mproject-Test-SNS
ARN	arn:aws:sns:us-east-1:905418258809:Mproject-Test-SNS	Topic owner	905418258809
Type	Standard		

- b. Create Subscription, Click create subscription
 - Select Protocol – Email
 - EndPoint – kvkirankumar1201@gmail.com

The screenshot shows the AWS SNS 'Subscription' page for a specific ARN. A green banner at the top indicates that the subscription has been created successfully. Below the banner, the ARN is listed as 'arn:aws:sns:us-east-1:905418258809:Mproject-Test-SNS:5bdec1c7-0200-42c5-8d4a-847daeefed2d'. The 'Subscription: 5bdec1c7-0200-42c5-8d4a-847daeefed2d' section shows the following details:

ARN	arn:aws:sns:us-east-1:905418258809:Mproject-Test-SNS:5bdec1c7-0200-42c5-8d4a-847daeefed2d	Status	Pending confirmation
Endpoint	kvkirankumar1201@gmail.com	Protocol	EMAIL
Topic	Mproject-Test-SNS		
Subscription Principal	arn:aws:iam::905418258809:root		

- c. Confirm the subscription in the Email endpoint.



- d. Refresh the page. Status has been turned to “ Confirmed ”

The screenshot shows the AWS SNS "Topics" page. At the top, there is a banner for a new feature: "Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)". Below the banner, the navigation path is shown: "Amazon SNS > Topics > Mproject-Test-SNS > Subscription: 5bdec1c7-0200-42c5-8d4a-847daefed2d". On the right side of this path, there are "Edit" and "Delete" buttons. The main content area displays a table with the following data:

Details	
ARN	Status
arn:aws:sns:us-east-1:905418258809:Mproject-Test-SNS:5bdec1c7-0200-42c5-8d4a-847daefed2d	<input checked="" type="checkbox"/> Confirmed
Endpoint	Protocol
kvikrankumar1201@gmail.com	EMAIL
Topic	
Mproject-Test-SNS	
Subscription Principal	
arn:aws:iam::905418258809:root	

Simple Notification configuration has done.

Step 3: Custom AMI configuration.

a. Click Services -> Compute -> EC2 -> Launch Instance

- Name - Mproject-Test-Ec2-Instance
- Select Amazon Machine Image as quick start Red Hat (Free tier eligible) enterprises Linux 9.
- Instance Type – t2.micro free tier eligible.
- Click Create key-pair -> name (Linux_key) -> key pair type (RSA) -> file formate(.ppk) -> Create key pair.
- Select Key pair – linux_key
- Network settings VPC – Mproject-Test-VPC.
- Subnet – Mproject-Test-public-subnet-1a
- Auto-assign public ip – enable
- Create security group - > Security group name(Mproject-Test-SG)
- Inbound security group rules - > Allow SSH port:22 protocol(TCP) for instance connection and click add security group rule -> select type HTTP port:80 source type Anywhere for server hosting.
- Configure storage -> 10 GiB and gp2 root volume (Up to 30 GB included in free tier).
- Click Launch Instance
- Wait Instance state turning into the Running state.

Instance summary for i-005fe7ddb1f08a01b (Mproject-Test-Ec2-Instance) Info		
C Connect Instance state Actions		
Updated less than a minute ago		
Instance ID i-005fe7ddb1f08a01b (Mproject-Test-Ec2-Instance)	Public IPv4 address 3.237.178.21 open address	Private IPv4 addresses 20.0.1.130
IPv6 address –	Instance state Running	Public IPv4 DNS –
Hostname type IP name: ip-20-0-1-130.ec2.internal	Private IP DNS name (IPv4 only) ip-20-0-1-130.ec2.internal	Elastic IP addresses –
Answer private resource DNS name –	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address 3.237.178.21 [Public IP]	VPC ID vpc-019e97407464ae05f (Mproject-Test-VPC)	Auto Scaling Group name –
IAM Role –	Subnet ID subnet-0fdfd019a125ef709 (Mproject-test-public-subnet-1a)	–
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:905418258809:instance/i-	–

Instance has been successfully created. Instance state become a “Running”

b. Let's connect the putty.

- Open Putty
- In session -> Host name or IP address -> paste Instance Public IPV4 address (3.237.178.21) -> port : 80, connection type SSH,
- Next window -> appetence -> Change – Courier name -> Terminal.
- Connection -> Enable TCP_Keepalives -> set TCP_Keepalives as (100).
- Connection -> SSH -> Auth -> Credentials -> Browse private key while insance creation we have created private key pair -> upload.
- Click open
- Accept putty alert.
- Login as ec2-user.

```
ec2-user@ip-20-0-1-130:~$ login as: ec2-user
Authenticating with public key "imported-openssh-key"
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
[ec2-user@ip-20-0-1-130 ~]$
```

c. Now let's deploy the any website (Netflix).

- Install httpd and wget servers – sudo yum install httpd wget -y
- Put developed index.html code into - /var/www/html/ directory

```
ec2-user@ip-20-0-1-130:/var/www/html$ cd /var/www/html/
ec2-user@ip-20-0-1-130 html]$ sudo wget https://www.netflix.com/in/
--2024-06-19 16:26:52-- https://www.netflix.com/in/
Resolving www.netflix.com... netflix[3.211.157.115, 3.225.92.8, 54.160.93.182, ...
Connecting to www.netflix.com (www.netflix.com)[3.211.157.115]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: aindex.html[1]

index.html [=>
2024-06-19 16:26:52 (16.9 MB/s) - aindex.html[ saved [1435449]
[ec2-user@ip-20-0-1-130 html]$ ls -l
total 1404
-rw-r--r-- 1 root root 1435449 Jun 19 16:26 index.html
[ec2-user@ip-20-0-1-130 html]$
```

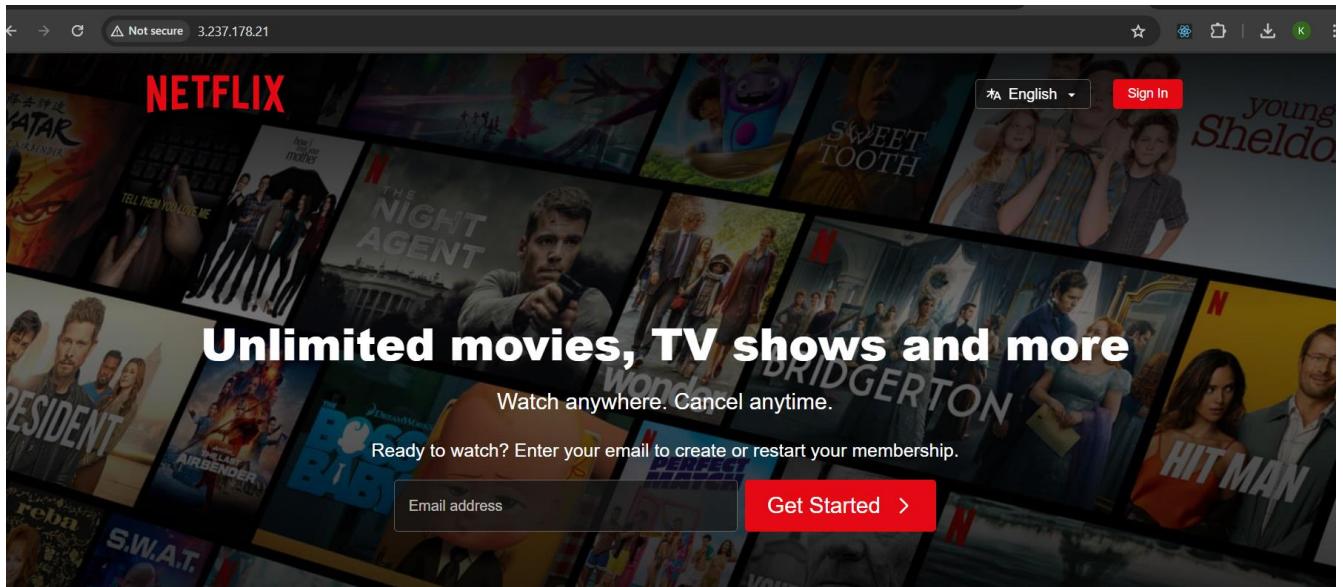
- Start deploying the Netflix website
- Run – sudo systemctl start httpd
- Enable server – sudo systemctl enable httpd
- Check status – sudo systemctl status httpd (Active status)

```
ec2-user@ip-20-0-1-130 html]$ sudo systemctl start httpd
ec2-user@ip-20-0-1-130 html]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-20-0-1-130 html]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
   Active: active (running) since Wed 2024-06-19 16:30:50 UTC; 19s ago
     Docs: man:httpd.service(8)
   Main PID: 14438 (httpd)
      Tasks: 177 (limit: 4400)
     Memory: 22.0M
        CPU: 74ms
       CGroup: /system.slice/httpd.service
           └─14438 /usr/sbin/httpd -DFOREGROUND
               ├─14439 /usr/sbin/httpd -DFOREGROUND
               ├─14440 /usr/sbin/httpd -DFOREGROUND
               ├─14441 /usr/sbin/httpd -DFOREGROUND
               └─14442 /usr/sbin/httpd -DFOREGROUND

Jun 19 16:30:50 ip-20-0-1-130.ec2.internal systemd[1]: Starting The Apache HTTP Server...
Jun 19 16:30:50 ip-20-0-1-130.ec2.internal systemd[1]: Started The Apache HTTP Server.
Jun 19 16:30:50 ip-20-0-1-130.ec2.internal httpd[14438]: Server configured, listening on: port 80
[ec2-user@ip-20-0-1-130 html]$
```

d. Verify the website successfully deployed into the server

- Go to the instance copy the public ipv4 address (3.237.178.21), paste in to the any web browser check server up and running.



e. Creation of custom AMI:

- Select the instance -> click action -> select image and templates -> Create image.
- Name - Mproject-Test-AMI,
- Enable Tag image and snapshot together
- Create Image.
- Click service -> compute -> EC2-> AMI.

A screenshot of the AWS Lambda console. At the top, it says "Lambda Functions (0)" and "Create New Function". Below that, there's a search bar and a "Create New Function" button. The main area shows a table with one row: "HelloWorld" (Function name) and "nodejs14.x" (Runtime). Under "Actions", there are buttons for "Edit", "Delete", "Version History", and "Logs". At the bottom, there's a "Create New Function" button again.

- Select the AMI. You can see that the status is pending. Wait till it becomes Available (take 5-10 mins)

The screenshot shows the AWS EC2 service interface. On the left, there's a navigation sidebar with links like EC2 Dashboard, EC2 Global View, Events, Console-to-Code Preview, Instances, Images, and Elastic Block Store. The main content area is titled "Amazon Machine Images (AMIs) (1/1)" and shows a single item: "1b3f3064e" which is "905418258809/Mproject-Test-AMI". The status is listed as "Available". Below this, a detailed view for "AMI ID: ami-0eb254271b3f3064e" is shown with tabs for Details, Permissions, Storage, and Tags. The Details tab displays information such as AMI ID, Image type, Platform details, Root device type, AMI name, Owner account ID, Architecture, Usage operation, Root device name, Status, Source, Virtualization type, and Kernel ID.

- In the same panel check snapshot created and it's turned into the available state.

The screenshot shows the AWS EC2 service interface, specifically the Snapshots section. The left sidebar includes links for Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, Catalog), and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager). The main content area is titled "Snapshots (1/1)" and shows a single item: "snap-0fe3f5362eb28edb5". The status is listed as "Completed". Below this, a detailed view for "Snapshot ID: snap-0fe3f5362eb28edb5" is shown with tabs for Details, Snapshot settings, Storage tier, and Tags. The Details tab displays information such as Snapshot ID, Volume size, Progress, Snapshot status, Owner, Volume ID, Started, Product codes, Encryption, KMS key ID, KMS key alias, and Description.

Step 4: Configuration of Launch template.

a. Click Service -> Ec2 -> Launch Templates ->Create Launch Templates

- Name – Mproject-Test-Templete
- Version Template description - Mproject-Test-Templete
- AMI -> My AMI -> browse Mproject-Test-AMI -> select.
- Instance type – t2.micro (Free tire eligible).
- Security group – Mproject – Test-SG
- Click create launch templete.

The screenshot shows the AWS EC2 Launch Templates interface. On the left, a navigation sidebar lists various EC2 services like Dashboard, Global View, Instances, Images, and Block Store. The 'Launch Templates' section is selected. The main area displays a table titled 'Launch Templates (1/1)'. A single entry is shown: 'lt-0203bd9efc25ec198' with the name 'Mproject-Test-Templete'. Below this, a detailed view for 'Mproject-Test-Templete (lt-0203bd9efc25ec198)' is expanded. It shows 'Launch template details' with fields: Launch template ID (lt-0203bd9efc25ec198), Launch template name (Mproject-Test-Templete), Default version (1), and Owner (arn:aws:iam::905418258809:root). There are tabs for 'Details', 'Versions', and 'Template tags'. At the bottom, there's a 'Launch template version details' section with 'Actions' and 'Delete template version' buttons.

Launch template configuration has done.

Step 5: Application Load Balancer Configuration.

- a. Click Service -> compute -> Ec2-> Load balancer -> Target Group ->Create Target group.
 - Target type – Instance
 - Target group name – HomePage.
 - Protocol – HTTP, port:80.
 - Ip address type – Ipv4.
 - VPC – Mproject – Test-VPC
 - Protocol version – HTTP1.
 - Health check path - /index.html.
 - Advance health check setting- health threshold(2), Unhealth threshold(2), Time out(2), Interval (5) and success codes as 200.
 - Click Next -> select available instance (Mproject-Test-Ec2-instance) -> click include as pending bellow.
 - Create target group.

The screenshot shows the AWS EC2 Target Groups page. A success message at the top states: "Successfully created the target group: HomePage. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the Targets tab." The main content area displays the details of the 'HomePage' target group. The 'Details' section includes the ARN: am:aws:elasticloadbalancing:us-east-1:905418258809:targetgroup/HomePage/51ff22f7feddc6ba, Target type: Instance, Protocol: Port HTTP: 80, Protocol version: HTTP1, VPC: vpc-019e97407464ae05f, IP address type: IPv4, Load balancer: None associated, and a summary of targets: 1 Total targets (0 Healthy, 0 Unhealthy, 0 Anomalous), 1 Unused, 0 Initial, 0 Draining. Below this is a section titled "Distribution of targets by Availability Zone (AZ)" with a note: "Select values in this table to see corresponding filters applied to the Registered targets table below." At the bottom, there are tabs for Targets (which is selected), Monitoring, Health checks, Attributes, and Tags.

b. Creating Application load balancer

- Click in same panel Load balancer -> create load balancer.
- Select application load balancer -> create.
- Load balancer name - Mproject-Test-ALB
- Scheme – internet – facing
- Ip address type – ipv4.
- Network Mapping – VPC(Mproject-Test-VPC) and Maping select Mproject-Test-public-subnet-1a and Mproject-Test-public-subnet-1b (N.Virginia availability zone).
- Select security group as – Mproject-Test-SG
- Listener routing – protocol(HTTP), port:80, Default action target group (HomePage).
- Create load balancer.
- Wait for status to turn **Active**.

The screenshot shows the AWS Elastic Load Balancing (ELB) console. On the left, there's a navigation sidebar with options like EC2 Dashboard, EC2 Global View, Events, Console-to-Code Preview, Instances, Images, and Elastic Block Store. The main area is titled 'Mproject-Test-ALB' and shows the 'Details' tab for the load balancer. Key details include:

- Load balancer type:** Application
- Status:** Active
- VPC:** [vpc-019e97407464ae05f](#)
- IP address type:** IPv4
- Scheme:** Internet-facing
- Hosted zone:** Z355XDOTRQ7X7K
- Availability Zones:** subnet-0fdfd019a125ef709 (us-east-1a, use1-az1) and subnet-0168c4ddfcfa9f9921 (us-east-1b, use1-az2)
- Date created:** June 19, 2024, 22:51 (UTC+05:30)
- Load balancer ARN:** arn:aws:elasticloadbalancing:us-east-1:905418258809:loadbalancer/app/Mproject-Test-ALB/594fcf7948d8c302
- DNS name:** [Mproject-Test-ALB-998307198.us-east-1.elb.amazonaws.com](#) (A Record)

Below the details, there are tabs for 'Listeners and rules', 'Network mapping', 'Resource map - new', 'Security' (which is selected), 'Monitoring', 'Integrations', 'Attributes', and 'Tags'. Under the 'Security' tab, it shows 'Security groups (1)' with the 'Mproject-Test-SG' selected. There's also an 'Edit' button.

- Copy the DNS name and Check status on web browser
- DNS name - Mproject-Test-ALB-998307198.us-east-1.elb.amazonaws.com

The screenshot shows the AWS ELB console again, focusing on the copied DNS name. The copied URL is pasted into the browser's address bar, and the page is loading. The copied URL is visible in the address bar as 'east-1b (use1-az2)'.

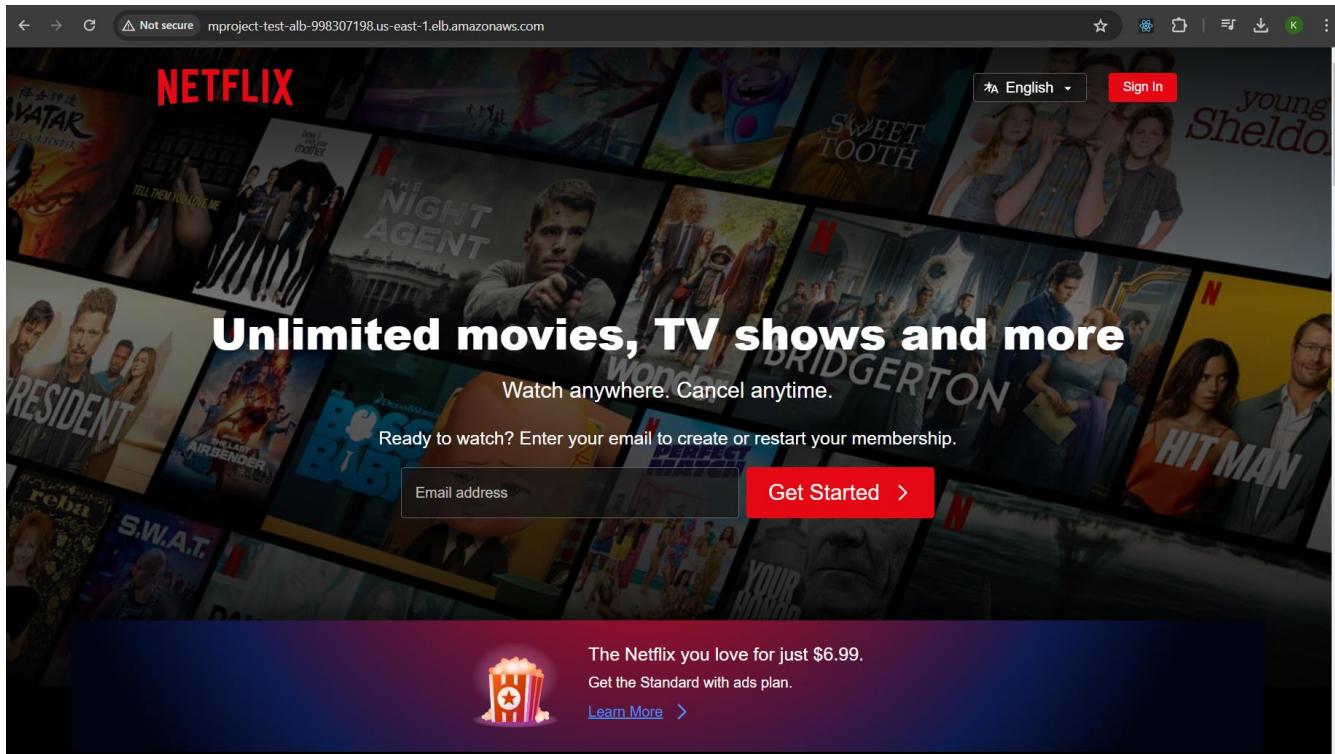
On the left, the copied URL is shown under 'Load balancer ARN':

Load balancer ARN
[arn:aws:elasticloadbalancing:us-east-1:905418258809:loadbalancer/app/Mproject-Test-ALB/594fcf7948d8c302](#)

On the right, the copied URL is shown under 'DNS name' (Info):

DNS name [Info](#)
[Mproject-Test-ALB-998307198.us-east-1.elb.amazonaws.com](#) (A Record)

Application load balancer has successfully configured.



Step 6: Auto Scaling Group Configuration.

- a. Click service -> compute -> ec2 -> Auto scaling -> create auto scaling groups.
 - Name – Mproject-Test-AutoScalingGroup.
 - Select the Launch template – Mproject-Test-Template.
 - Click – next
 - Select VPC – Mproject-Test-VPC.
 - Select availability zones and subnets – Mproject-Test-public-subnet-1a and Mproject-Test-public-subnet-1b.
 - Next.
 - Load balancing – > enable Attach on existing load balancer -> enable chose from your load balancer -> select the existing load balancer target group (HomePage).
 - Enable No VPC lattice service.
 - In additional settings, Checkmark the Enable group metrics collection within Cloud Watch option.
 - Now click Next.
 - Set desired capacity to 1 (because only 1 we can create in free-tier).
 - Set Minimum desired capacity to 1, and max to 3. Select No scaling policies.
 - Select instance maintenance policy as No policy.
 - Checkmark the enable instance scale-in protection and click next.
 - Click Add Notification -> Set SNS topic to Mproject-Test-SNS, allow all event types and click next.
 - Click next.
 - Review the configuration and create auto scaling group.

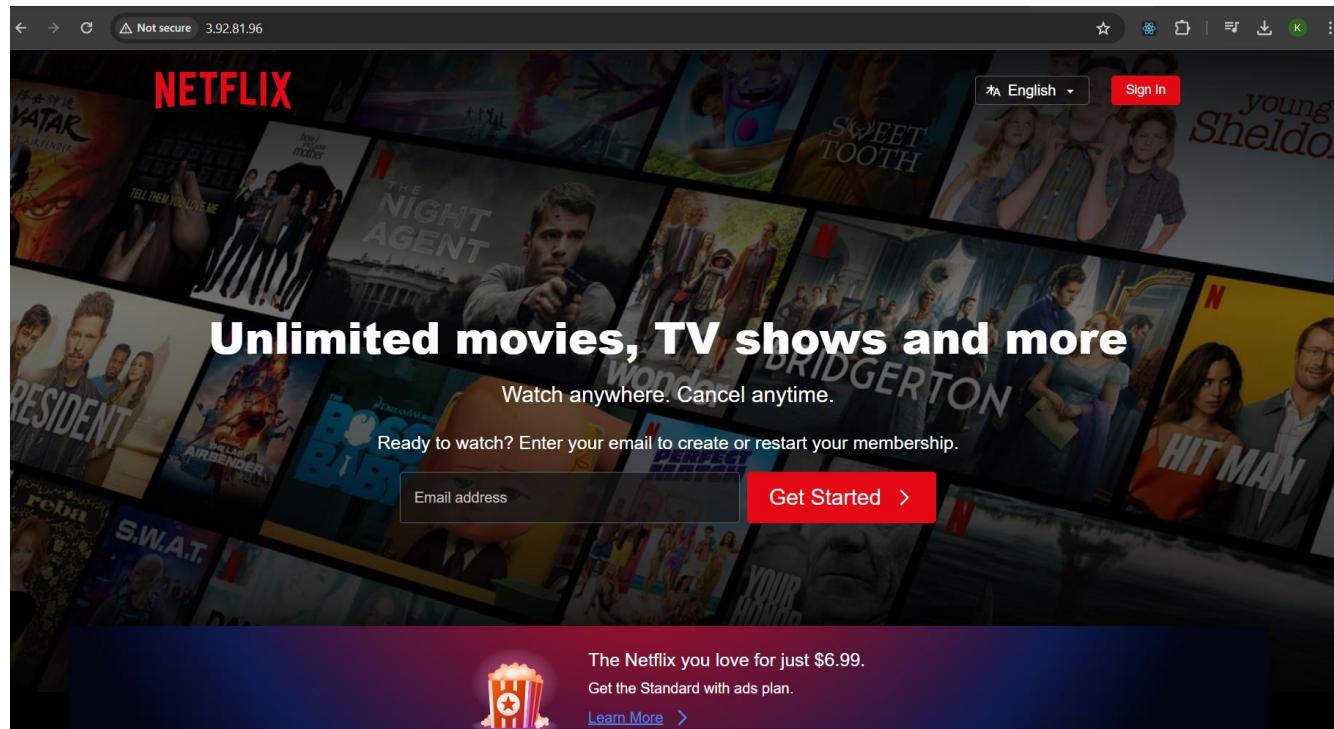
The screenshot shows the AWS EC2 Auto Scaling Groups page. At the top, there's a breadcrumb navigation: EC2 > Auto Scaling groups. Below the navigation, there's a search bar with placeholder text "Search your Auto Scaling groups". To the right of the search bar are navigation icons: back, forward, and refresh. A yellow "Create Auto Scaling group" button is located at the top right. The main area displays a table with one row of data. The columns are: Name, Launch template/configuration, Instances, Status, Desired capacity, Min, Max, and a dropdown arrow. The first column shows a checkbox followed by the name "Mproject-Test-AutoScalingGroup". The second column shows a link to "Mproject-Test-Template". The third column shows "0" instances. The fourth column shows "Updating capacity..." status. The fifth column has a value of "1". The sixth column has a value of "1". The seventh column has a value of "3". The eighth column has a value of "u...".

Auto Scaling Group has been created successfully and running server.

We can observe the auto scaling group activity.

The screenshot shows the AWS EC2 Auto Scaling Groups Activity page. The left sidebar includes sections for 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), and Auto Scaling (Auto Scaling Groups, Settings). The main content area displays two sections: 'Activity notifications (1)' and 'Activity history (1)'. The 'Activity notifications' section shows a single entry for 'Send to Mproject-Test-SNS' with the cause 'Launch, Terminate, Fail to launch, Fail to terminate'. The 'Activity history' section shows a single successful instance launch event from June 19, 2024, at 19:11 PM, with the instance ID i-0882ed87c335a4032.

We can see the newly created server with new Ipv4 address (3.92.81.96)



Step 7: AWS Cloud Watch Configuration.

- a. Click Services -> Management and Governance -> Cloud Watch.
 - Click All alarms -> create alarm.
 - Click select matrix.
 - Matrix → EC2 → per Instance Matrix.
 - Filter the based on instance id
 - Select CPU-Utilization
 - Click select Matrix.
 - Set statistic to maximum and period to 1 minute.
 - Select static, let's create for **scale-out** operation, select greater and value to **30** and click next.
 - Alarm state trigger – In-alarm.
 - Select existing sns topic – Mproject-Test-SNS, then click next.
 - Alarm name – Scale-out-alarm
 - Add alarm description - # When ever CPU-Utilization threshold value crosses 30%, Automatically Auto scaling group create new instance.
 - Click next review configuration and create alarm.

The screenshot shows the AWS CloudWatch Alarms interface. On the left, there is a navigation sidebar with options like Favorites and recents, Dashboards, Alarms (0), In alarm, All alarms, Billing, Logs, Metrics, All metrics, and Explorer. The main area is titled 'CloudWatch > Alarms' and shows 'Alarms (1/1)'. It includes search, filter, and action buttons. A table lists the alarm details: Name (Scale-out-alarm), State (Insufficient data), Last state update (UTC) (2024-06-19 18:08:15), Conditions (CPUUtilization > 30 for 1 datapoints within 1 minute), and Actions (Actions enabled). A prominent orange 'Create alarm' button is at the top right.

- Create a Scal-in- alarm following above steps and Select static, let's create for **scale-in** operation, select lower and value to **15** and click next.

The screenshot shows the AWS CloudWatch Alarms interface after creating a second alarm. The navigation sidebar is identical to the previous screenshot. The main area shows 'Alarms (2)'. The first alarm is 'Scale-out-alarm' (CPUUtilization > 30 for 1 datapoints within 1 minute). The second alarm is 'Scal-in-alarm' (CPUUtilization < 15 for 1 datapoints within 1 minute). Both alarms have 'Actions enabled'. A green banner at the top says 'Successfully created alarm Scal-in-alarm.' A 'View alarm' button is visible at the top right.

All the details regarding the scale-in alarm is given in the email.

ALARM: "Scal-in-alarm" in US East (N. Virginia) Inbox x Print  

Mproject-Test-SNS <no-reply@sns.amazonaws.com> to me 23:47 (0 minutes ago)    

You are receiving this email because your Amazon CloudWatch Alarm "Scal-in-alarm" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [0.483870967741931 (19/06/24 18:11:00)] was less than the threshold (15.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Wednesday 19 June, 2024 18:17:11 UTC".

View this alarm in the AWS Management Console:
<https://us-east-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=us-east-1#alarmsV2:alarm/Scal-in-alarm>

Alarm Details:

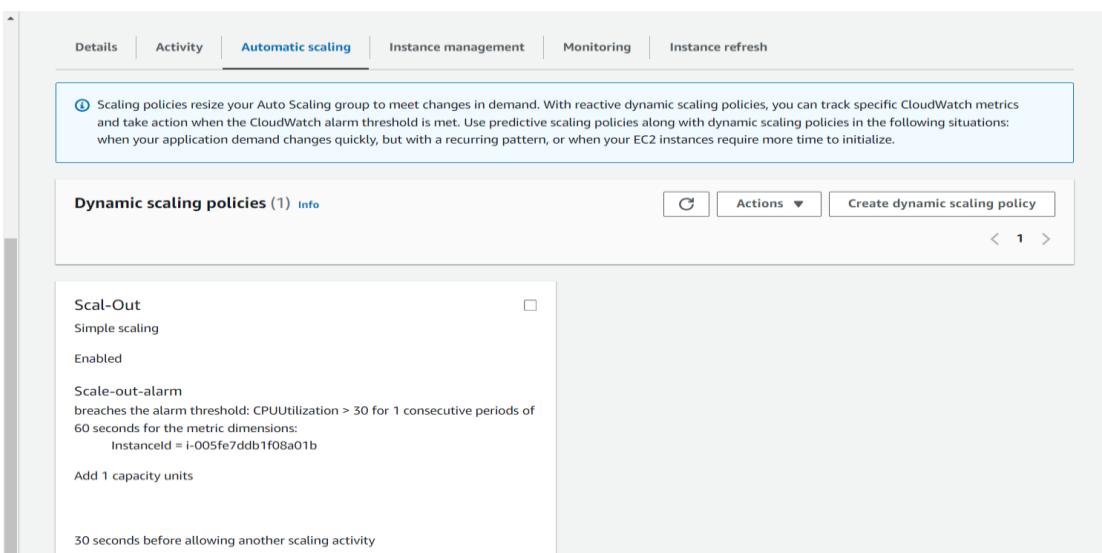
- Name: Scal-in-alarm
- Description: #When ever instance come-down into the 15% automatically instance removed.
- State Change: INSUFFICIENT_DATA -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [0.483870967741931 (19/06/24 18:11:00)] was less than the threshold (15.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Wednesday 19 June, 2024 18:17:11 UTC
- AWS Account: 905418258809
- Alarm Arn: arn:aws:cloudwatch:us-east-1:905418258809:alarm:Scal-in-alarm

Threshold:

- The alarm is in the ALARM state when the metric is LessThanThreshold 15.0 for at least 1 of the last 1 period(s) of 60 seconds.

Step 8: Attach the Scal-in-alarm and Scal-out-alarm into Auto scaling group.

- a. Click **Auto scaling groups** -> select **Mproject-Test-AutoScalingGroup** -> **Automatic scaling** -> **create dynamic scaling policy**.
 - Click automatic scaling -> create dynamic scaling policy.
 - Select policy type – simple scaling.
 - Name – Scal-Out
 - Select the Cloud Watch alarm: Scale-out-alarm
 - Action: Add 1
 - Time: 30 seconds and click create.



The screenshot shows the AWS CloudWatch Metrics console. On the left, there's a sidebar with navigation links: 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), and Auto Scaling (Auto Scaling Groups). The main area has tabs: Details, Activity, Automatic scaling (which is selected), Instance management, Monitoring, and Instance refresh. A callout box on the 'Automatic scaling' tab explains that scaling policies resize your Auto Scaling group to meet changes in demand. Below this, a section titled 'Dynamic scaling policies (1)' shows a single policy named 'Scal-Out'. The policy is set to 'Simple scaling', 'Enabled', and triggered by the 'Scale-out-alarm' which breaches the CPUUtilization > 30 metric dimension over 60 seconds. The action is to 'Add 1 capacity units' with a '30 seconds before allowing another scaling activity' cooldown. There are 'Actions' and 'Create dynamic scaling policy' buttons at the top of the policy list.

- Follow the above steps
- Name – Scal-in
- Select cloud watch alarm as : Scale-in-alarm
- Set Action: Remove 1
- create

The screenshot shows the AWS CloudWatch Metrics console. On the left, there's a sidebar with navigation links for 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), Auto Scaling (Auto Scaling Groups), and Settings.

The main area displays a success message: "Dynamic scaling policy created or edited successfully." Below this, the "Dynamic scaling policies" section shows two entries:

- Scal-In**: Simple scaling, Enabled. Scal-in-alarm: breaches the alarm threshold: CPUUtilization < 15 for 1 consecutive periods of 60 seconds for the metric dimensions: Instanceld = i-005fe7ddb1f08a01b. Actions: Remove 1 capacity units, 30 seconds before allowing another scaling activity.
- Scal-Out**: Simple scaling, Enabled. Scale-out-alarm: breaches the alarm threshold: CPUUtilization > 30 for 1 consecutive periods of 60 seconds for the metric dimensions: Instanceld = i-005fe7ddb1f08a01b. Actions: Add 1 capacity units, 30 seconds before allowing another scaling activity.

Below the scaling policies, the "Predictive scaling policies" section shows 0 entries with an "Info" link and a "Create predictive scaling policy" button. There's also an "Evaluation period" input field.

Now successfully configured scal-in and scal-out policies.

Now let us increase the load on CPU.

Select instance, copy IPv4 (3.237.178.21) public address and set up putty.

Login as ec2-user

```
[ec2-user@ip-20-0-1-130:~]
[ec2-user@ip-20-0-1-130:~] login as: ec2-user
[ec2-user@ip-20-0-1-130:~] Authenticating with public key "imported-openssh-key"
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Wed Jun 19 16:19:48 2024 from 103.5.132.170
[ec2-user@ip-20-0-1-130 ~]$ top
```

Now let us run **top** command to see CPU Usage.

We can see **%Cpu is 0.0**. Let us increase the load now. Press q to quit.

```
[ec2-user@ip-20-0-1-130:~]
top - 18:48:46 up 2:04, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 110 total, 1 running, 109 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 765.7 total, 254.2 free, 295.3 used, 333.0 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 470.4 avail Mem

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
1799 apache   20   0  981516 12628  6272 S  0.3  1.6  0:01.21 httpd
  1 root      20   0  172780 15984 10652 S  0.0  2.0  0:02.21 systemd
  2 root      20   0          0    0   0 S  0.0  0.0  0:00.00 kthreadd
  3 root      0 -20          0    0   0 I  0.0  0.0  0:00.00 rcu_gp
  4 root      0 -20          0    0   0 I  0.0  0.0  0:00.00 rCU_par_gp
  5 root      0 -20          0    0   0 I  0.0  0.0  0:00.00 slub_flushwq
  6 root      0 -20          0    0   0 I  0.0  0.0  0:00.00 netns
  8 root      0 -20          0    0   0 I  0.0  0.0  0:00.00 kworker/0:0H-events_highpri
10 root      0 -20          0    0   0 I  0.0  0.0  0:00.00 mm_percpu_wq
11 root      20   0          0    0   0 I  0.0  0.0  0:00.33 kworker/u30:1-events_unbound
12 root      20   0          0    0   0 I  0.0  0.0  0:00.00 rCU_tasks_kthre
13 root      20   0          0    0   0 I  0.0  0.0  0:00.00 rCU_tasks_rude_
14 root      20   0          0    0   0 I  0.0  0.0  0:00.00 rCU_tasks_trace
15 root      20   0          0    0   0 S  0.0  0.0  0:00.02 ksoftirqd/0
16 root      20   0          0    0   0 S  0.0  0.0  0:01.55 pr/tty0
17 root      20   0          0    0   0 S  0.0  0.0  0:00.68 pr/tty0
18 root      20   0          0    0   0 I  0.0  0.0  0:00.07 rCU_preempt
19 root      rt   0          0    0   0 S  0.0  0.0  0:00.01 migration/0
20 root      -51   0          0    0   0 S  0.0  0.0  0:00.00 idle_inject/0
22 root      20   0          0    0   0 S  0.0  0.0  0:00.00 cpuhp/0
24 root      20   0          0    0   0 S  0.0  0.0  0:00.00 kdevtmpfs
25 root      0 -20          0    0   0 I  0.0  0.0  0:00.00 inet_frag_wq
26 root      20   0          0    0   0 S  0.0  0.0  0:00.00 kauditfd
27 root      20   0          0    0   0 S  0.0  0.0  0:00.00 khungtaskd
28 root      20   0          0    0   0 I  0.0  0.0  0:00.23 kworker/u30:2-events_unbound
```

Execute the below command 5 times to increase the CPU Utilization.

yes > /dev/null &

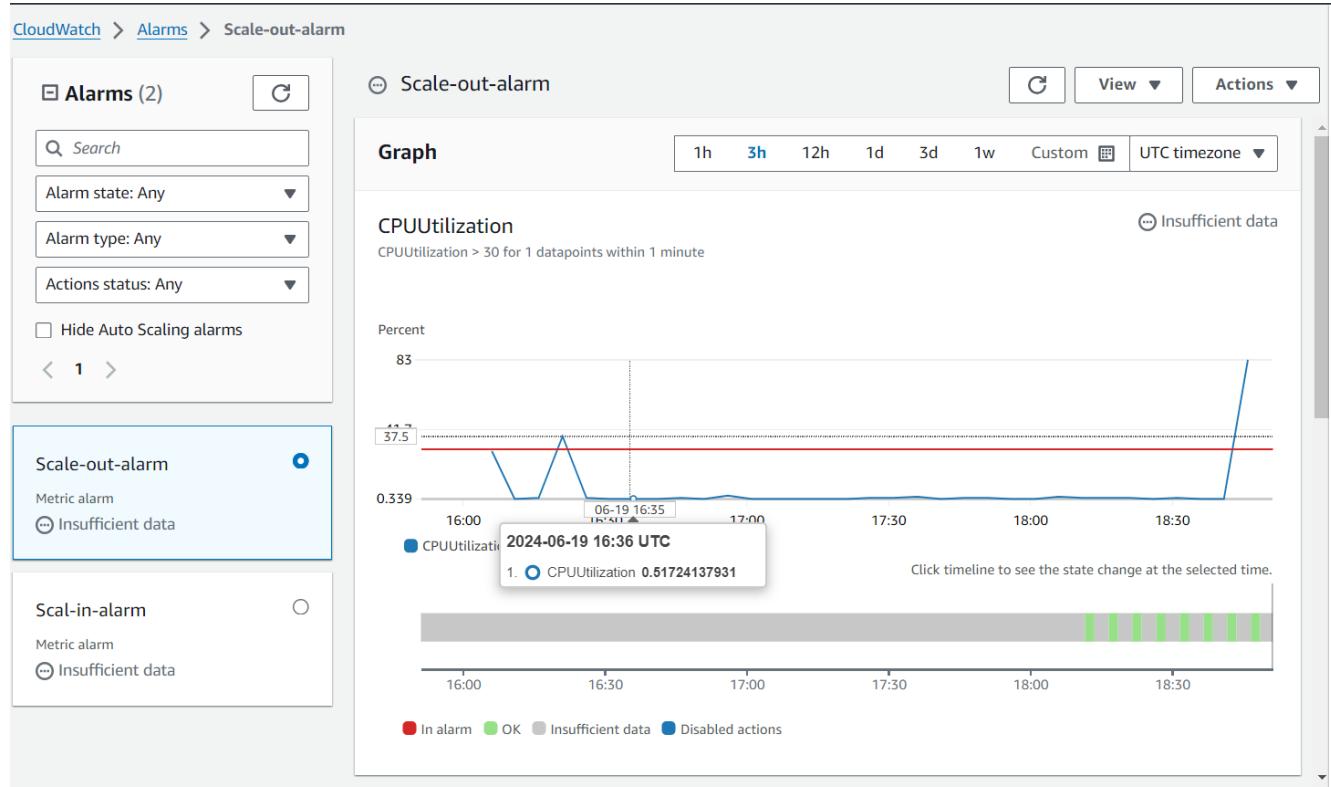
```
[ec2-user@ip-20-0-1-130 ~]$ yes > /dev/null &
[1] 2164
[ec2-user@ip-20-0-1-130 ~]$
[ec2-user@ip-20-0-1-130 ~]$
[ec2-user@ip-20-0-1-130 ~]$ yes > /dev/null &
[2] 2165
[ec2-user@ip-20-0-1-130 ~]$ yes > /dev/null &
[3] 2166
[ec2-user@ip-20-0-1-130 ~]$ yes > /dev/null &
[4] 2167
[ec2-user@ip-20-0-1-130 ~]$ yes > /dev/null &
[5] 2168
[ec2-user@ip-20-0-1-130 ~]$ yes > /dev/null &
[6] 2169
[ec2-user@ip-20-0-1-130 ~]$ 
```

Run the **top** command check the CPU Utilization.

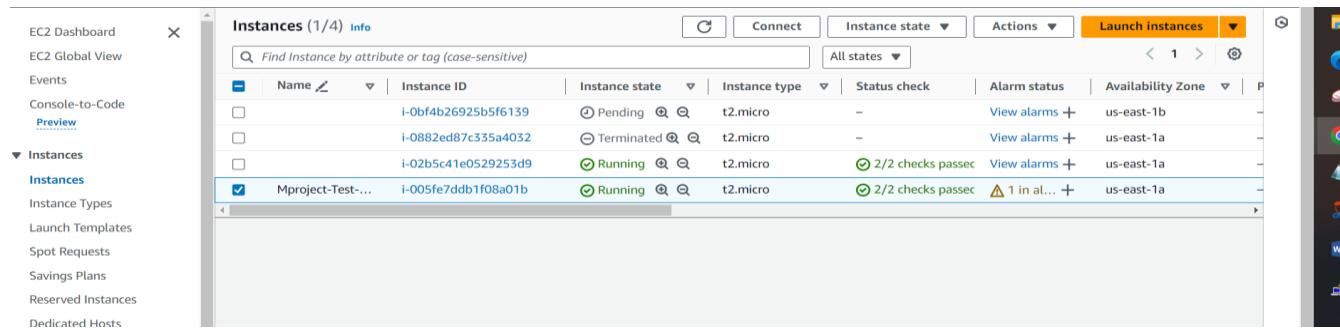
```
ec2-user@ip-20-0-1-130:~$ top - 18:50:53 up 2:06, 1 user, load average: 2.77, 0.71, 0.24
Tasks: 115 total, 7 running, 108 sleeping, 0 stopped, 0 zombie
%Cpu(s): 60.1 us, 39.2 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.7 hi, 0.0 si, 0.0 st
MiB Mem : 765.7 total, 245.5 free, 303.9 used, 333.1 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 461.7 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2164 ec2-user 20 0 5580 1664 1664 R 16.6 0.2 0:10.89 yes
2165 ec2-user 20 0 5580 1664 1664 R 16.6 0.2 0:06.50 yes
2167 ec2-user 20 0 5580 1792 1664 R 16.6 0.2 0:05.83 yes
2168 ec2-user 20 0 5580 1664 1664 R 16.6 0.2 0:05.53 yes
2169 ec2-user 20 0 5580 1664 1664 R 16.6 0.2 0:05.29 yes
2166 ec2-user 20 0 5580 1792 1664 R 16.3 0.2 0:06.12 yes
2076 ec2-user 20 0 19596 6952 5120 S 0.3 0.9 0:00.03 sshd
 1 root 20 0 172780 15984 10652 S 0.0 2.0 0:02.21 systemd
 2 root 20 0 0 0 S 0.0 0.0 0:00.00 kthreadd
 3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
 4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
 5 root 0 -20 0 0 0 T 0.0 0.0 0:00.00 slub_flushwq
```

Now from graph we can see Scale-out has crossed the threshold value of 30%.



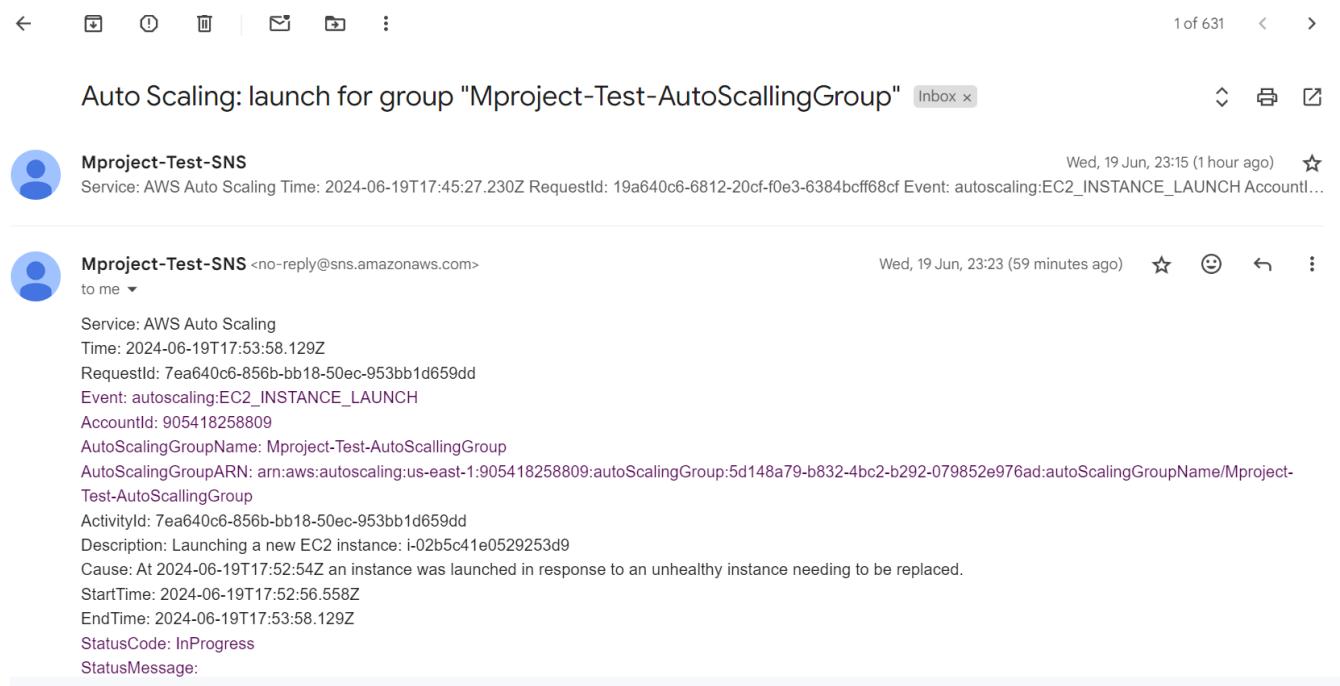
Which means a new instance will be created as demand is crossed. See the instance has changed from 1 to 2.



The screenshot shows the AWS EC2 Instances page with the title "Instances (1/4) Info". The table lists four instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
	i-0bf4b26925b5f6139	Pending	t2.micro	-	View alarms +	us-east-1b
	i-0882ed87c355a4032	Terminated	t2.micro	-	View alarms +	us-east-1a
	i-02b5c41e0529253d9	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a
Mproject-Test-...	i-005fe7ddb1f08a01b	Running	t2.micro	2/2 checks passed	1 in al... +	us-east-1a

We have received 2 emails. One for the scale-out alarm, and other for new instance creation.



The screenshot shows two emails in the inbox:

- Mproject-Test-SNS** (no-reply@sns.amazonaws.com) - Sent: Wed, 19 Jun, 23:15 (1 hour ago)
Service: AWS Auto Scaling Time: 2024-06-19T17:45:27.230Z RequestId: 19a640c6-6812-20cf-f0e3-6384bcff68cf Event: autoscaling:EC2_INSTANCE_LAUNCH AccountId: 905418258809
AutoScalingGroupName: Mproject-Test-AutoScalingGroup
AutoScalingGroupARN: arn:aws:autoscaling:us-east-1:1905418258809:autoScalingGroup:5d148a79-b832-4bc2-b292-079852e976ad:autoScalingGroupName/Mproject-Test-AutoScalingGroup
ActivityId: 7ea640c6-856b-bb18-50ec-953bb1d659dd
Description: Launching a new EC2 instance: i-02b5c41e0529253d9
Cause: At 2024-06-19T17:52:54Z an instance was launched in response to an unhealthy instance needing to be replaced.
StartTime: 2024-06-19T17:52:56.558Z
EndTime: 2024-06-19T17:53:58.129Z
StatusCode: InProgress
StatusMessage:
- Mproject-Test-SNS** (no-reply@sns.amazonaws.com) - Sent: Wed, 19 Jun, 23:23 (59 minutes ago)
to me ▾
Service: AWS Auto Scaling
Time: 2024-06-19T17:53:58.129Z
RequestId: 7ea640c6-856b-bb18-50ec-953bb1d659dd
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 905418258809
AutoScalingGroupName: Mproject-Test-AutoScalingGroup
AutoScalingGroupARN: arn:aws:autoscaling:us-east-1:1905418258809:autoScalingGroup:5d148a79-b832-4bc2-b292-079852e976ad:autoScalingGroupName/Mproject-Test-AutoScalingGroup
ActivityId: 7ea640c6-856b-bb18-50ec-953bb1d659dd
Description: Launching a new EC2 instance: i-005fe7ddb1f08a01b
Cause: At 2024-06-19T17:52:54Z an instance was launched in response to an unhealthy instance needing to be replaced.
StartTime: 2024-06-19T17:52:56.558Z
EndTime: 2024-06-19T17:53:58.129Z
StatusCode: InProgress
StatusMessage:

Completed has been completed and tested successfully.

Chapter-4

CLEAN-UP

1. Delete Auto Scaling Groups.

Select Group (Mproject-Test-AutoScalingGroup) -> Actions -> Delete

The screenshot shows the AWS Auto Scaling Groups page. On the left, a sidebar lists various services: 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), and EC2 Dashboard. The main content area displays a table titled "Auto Scaling groups (1) Info". The table has columns: Name, Launch template/configuration, Instances, Status, and Desired capacity. A single row is selected: "Mproject-Test-AutoScalingGroup" with "Mproject-Test-Template | Version Default" under Launch template/configuration, 3 instances, and a status of "Deleting". Below the table, a message says "0 Auto Scaling groups selected" and "Select an Auto Scaling group".

we can see the instance has also terminated.

The screenshot shows the AWS Instances page. On the left, a sidebar lists EC2 Dashboard, EC2 Global View, Events, Console-to-Code (Preview), Instances (Instances), and Images. The main content area displays a table titled "Instances (4) Info". The table has columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. Three rows are listed, all with the "Terminated" status and t2.micro instance type. The table header includes "Find Instance by attribute or tag (case-sensitive)" and "All states".

2. Delete Launch Templates.

Select Launch Template -> Actions -> Delete template

The screenshot shows the AWS Launch Templates page. On the left, a sidebar lists Instances, Instance Types, Launch Templates (Launch Templates), Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and Images. The main content area displays a table titled "Launch Templates Info". The table has columns: Launch Template ID, Launch Template Name, Default Version, Latest Version, Create Time, and a "C" column. A message at the bottom says "You do not have any Launch Templates in this region". Above the table, a green banner says "Delete Launch Template Request Succeeded".

3. Delete Images

AMI -> select Image -> actions -> de-register AMI

The screenshot shows the AWS Amazon Machine Images (AMIs) interface. On the left, there's a navigation sidebar with sections for Instances (Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations) and Images (AMIs, AMI Catalog). The main content area has a green header bar with the message "Successfully deregistered ami-0eb254271b3f3064e." Below this, the title is "Amazon Machine Images (AMIs) Info". There are buttons for Refresh, Recycle Bin, EC2 Image Builder, Actions, and Launch instance from AMI. A search bar says "Find AMI by attribute or tag" and dropdowns for "Owned by me" and "AMI name". The main table has columns for Name, AMI ID, Source, and Owner. A note says "No AMIs in this Region for: Owned by me." and "You can use the filter to view Owned By Me, Private Images, Public Images, or Disabled Images."

4. Delete Snapshots.

The screenshot shows the AWS Snapshots interface. The left sidebar includes Instances and Images sections. The main area has a green header bar with the message "Successfully deleted snapshot snap-0fe3f5362eb28edb5." The title is "Snapshots Info". It features a "Create snapshot" button and filters for "Owned by me" and "Search". The main table has columns for Name, Snapshot ID, Volume size, Description, Storage tier, Snapshot status, and Started. A note says "You currently have no snapshots in this Region." and "Select a snapshot above".

5. Delete Alarms.

Alarms -> all alarms -> select both alarms -> actions -> delete

The screenshot shows the AWS CloudWatch Alarms interface. The left sidebar lists CloudWatch, Favorites and recents, Dashboards, Alarms (0), In alarm, All alarms, Billing, Logs, Metrics, and Explorer. The main area has a green header bar with the message "Success Alarms were successfully deleted." The title is "CloudWatch > Alarms". It shows a table for "Alarms (0)" with columns for Name, State, Last state update (UTC), Conditions, and Actions. Filter options include Hide Auto Scaling alarms, Clear selection, Create composite alarm, Actions, and Create alarm. A note says "No alarms No alarms to display Read more about Alarms Create alarm".

6. Delete SNS.

Delete Subscription

The screenshot shows the Amazon SNS Subscriptions page. On the left, there's a sidebar with links like Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS), Origination numbers), and Images (AMIs, AMI Catalog). The main area has a green success banner at the top stating "Subscription deleted successfully." Below it is a table titled "Subscriptions (0)" with columns for ID, Endpoint, Status, Protocol, and Topic. A search bar and a "Create subscription" button are also present.

7. Delete Topic

The screenshot shows the Amazon SNS Topics page. The sidebar includes links for Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS), Origination numbers), and Images (AMIs, AMI Catalog). A green success banner at the top says "Topic Mproject-Test-SNS deleted successfully." The main content area shows a table titled "Topics (0)" with columns for Name, Type, and ARN. It displays a message "No topics" and "To get started, create a topic." with a "Create topic" button.

8. Delete Load Balancer

The screenshot shows the AWS EC2 Load balancers page. The sidebar lists Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), and Load Balancing (Load Balancers, Target Groups, Trust Stores). A green success banner at the top says "Successfully deleted load balancer: arn:aws:elasticloadbalancing:us-east-1:905418258809:loadbalancer/app/Mproject-Test-ALB/594fcf7948d8c302." The main area shows a table for "Load balancers" with columns for Name, DNS name, State, VPC ID, Availability Zones, and Type. It displays a message "No load balancers" and "You don't have any load balancers in us-east-1".

9. Delete Target group

The screenshot shows the AWS EC2 Target groups page. A green header bar at the top says "Successfully deleted target group: HomePage." Below it, the main content area has a title "Target groups" with a "Info" link. There is a search bar labeled "Filter target groups". To the right are buttons for "Actions" and "Create target group". A table header row includes columns for Name, ARN, Port, Protocol, Target type, and Load balancer. Below the table, a message says "No target groups" and "You don't have any target groups in us-east-1". At the bottom, a message says "0 target groups selected" and "Select a target group above.".

10. Remove association subnets in route table.

The screenshot shows the AWS VPC Edit subnet associations page. The URL is "VPC > Route tables > rtb-0c82e12defd93094a > Edit subnet associations". The title is "Edit subnet associations" with the sub-instruction "Change which subnets are associated with this route table.". A table titled "Available subnets (2)" lists two subnets: "Mproject-test-public-subnet-1a" and "Mproject-test-public-subnet-1b", each associated with the route table "rtb-0c82e12defd93094a / Mpproject-T". The table has columns for Name, Subnet ID, IPv4 CIDR, IPv6 CIDR, and Route table ID. At the bottom are buttons for "Cancel", "Save associations", and "Delete associations".

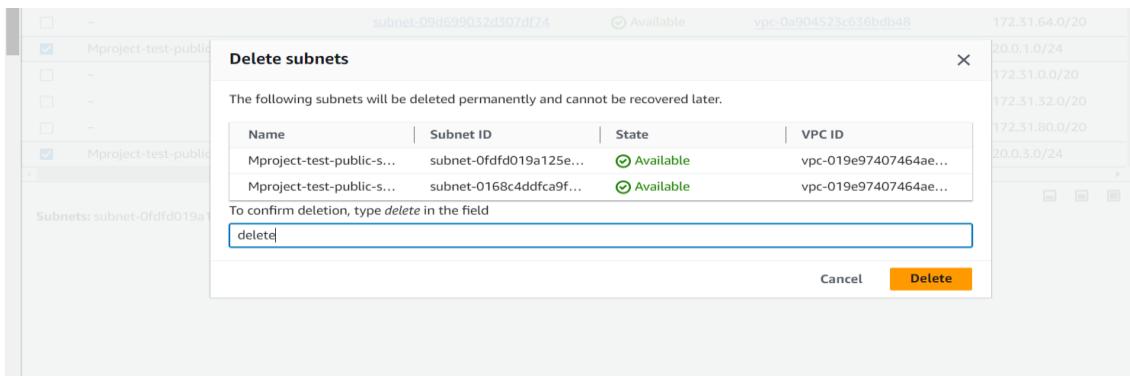
11. Remove the internet route in route table

The screenshot shows the AWS VPC Edit routes page. The URL is "VPC > Route tables > rtb-0c82e12defd93094a > Edit routes". The title is "Edit routes". A table lists routes with columns for Destination, Target, Status, and Propagated. One route has a destination of "0.0.0.0/0" and targets "local" and "Internet Gateway". Both targets are marked as "Active". The "Propagated" column shows "No" for both. A "Remove" button is visible next to the Internet Gateway entry. At the bottom are buttons for "Add route", "Cancel", "Preview", and "Save changes".

12. Delete the route table.

The screenshot shows the AWS VPC dashboard. The URL is "VPC dashboard". The main content area shows a message "You successfully deleted rtb-0c82e12defd93094a / Mpproject-Test-RT." Below it is a "Route tables (2) Info" table. The table has a header row with columns for Name, Route table ID, Explicit subnet associa..., Edge associations, Main, and VPC. It lists two route tables: "rtb-0199c974dfaf9b64e" and "rtb-02b0d6363e0975eab", both associated with the VPC "vpc-019e974074". The "Main" column shows "Yes" for both. At the bottom are buttons for "Actions" and "Create route table".

13. Delete the subnets.



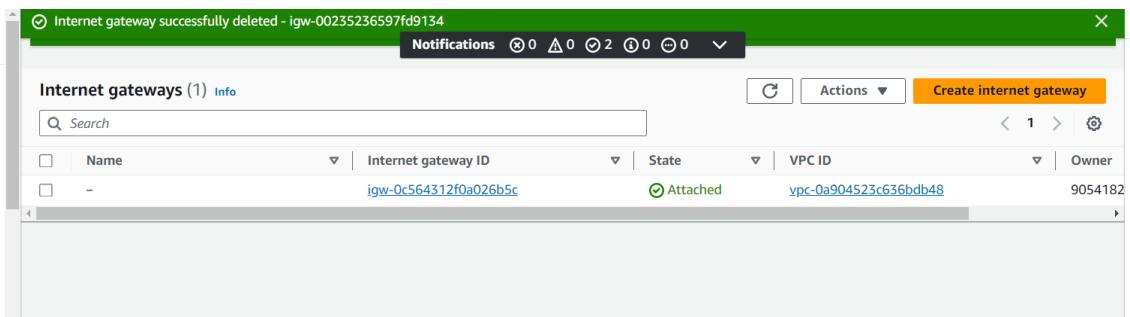
The screenshot shows the AWS VPC console. On the left, the navigation pane is open with 'Virtual private cloud' selected, showing options like 'Your VPCs', 'Subnets', 'Route tables', etc. In the main area, a 'Delete subnets' dialog is open. It lists two subnets: 'Mproject-test-public-s...' and 'Mproject-test-public-s...'. Both are marked as 'Available'. The dialog includes a message: 'The following subnets will be deleted permanently and cannot be recovered later.' Below the table, it says 'To confirm deletion, type `delete` in the field' and has a text input field containing 'delete'. At the bottom right are 'Cancel' and 'Delete' buttons.

14. Detach Internet gateway to VPC.



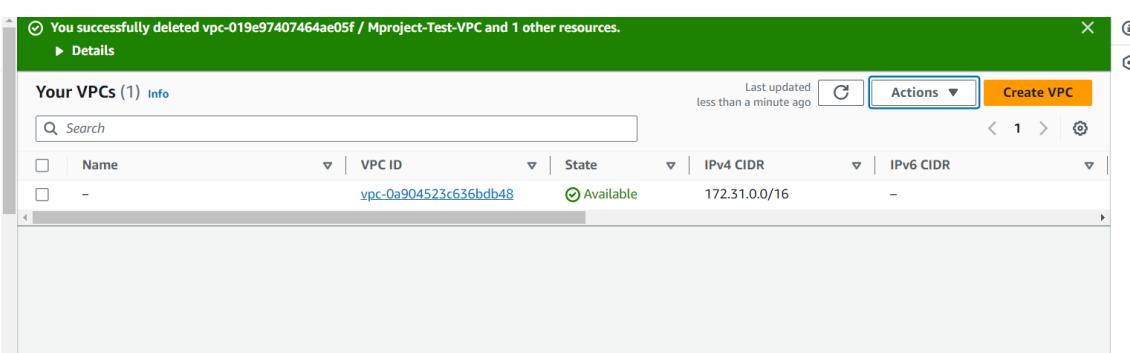
The screenshot shows the AWS VPC console. The left sidebar shows 'Virtual private cloud' selected, with 'Internet gateways' highlighted. In the main area, an 'Detach from VPC' dialog is open over an 'igw-00235236597fd9134 / Mproj...' item. The dialog asks, 'Are you sure that you want to detach internet gateway igw-00235236597fd9134 (Mproject-Test-IGW) from VPC vpc-019e97407464ae05f?' It also states, 'If you detach the internet gateway, resources in the VPC cannot communicate with the internet.' At the bottom are 'Cancel' and 'Detach internet gateway' buttons.

15. Delete internet gateway.



The screenshot shows the AWS VPC dashboard. The left sidebar shows 'Virtual private cloud' selected, with 'Internet gateways' highlighted. A notification bar at the top says 'Internet gateway successfully deleted - igw-00235236597fd9134'. The main area shows a table of 'Internet gateways (1)'. One row is listed: 'igw-0c564312f0a026b5c' with 'Attached' status, 'vpc-0a904523c636bdb48' VPC ID, and owner '9054182'. At the top right of the table are 'Actions' and 'Create internet gateway' buttons.

16. Delete VPC



The screenshot shows the AWS VPC dashboard. The left sidebar shows 'Virtual private cloud' selected, with 'Your VPCs' highlighted. A notification bar at the top says 'You successfully deleted vpc-019e97407464ae05f / Mproject-Test-VPC and 1 other resources.' The main area shows a table of 'Your VPCs (1)'. One row is listed: 'vpc-0a904523c636bdb48' with 'Available' status, '172.31.0.0/16' IPv4 CIDR, and '-' IPv6 CIDR. At the top right of the table are 'Actions' and 'Create VPC' buttons.

17.We have received the email of the termination process.

Auto Scaling: termination for group "Mproject-Test-AutoScallingGroup" [Inbox](#)   

 **Mproject-Test-SNS** Wed, 19 Jun, 23:28 (1 hour ago) 
Service: AWS Auto Scaling Time: 2024-06-19T17:58:05.733Z RequestId: d14640c6-854f-cf2d-83e3-82a41e3564f9 Event: autoscaling:EC2_INSTANCE_TERMINATE Ac...

 **Mproject-Test-SNS** <no-reply@sns.amazonaws.com> 00:34 (22 minutes ago)    
to me ▾

Service: AWS Auto Scaling
Time: 2024-06-19T19:04:01.045Z
RequestId: 6c9ff2e4-618a-4547-9181-93ac39e0293e
Event: autoscaling:EC2_INSTANCE_TERMINATE
AccountId: 905418258809
AutoScalingGroupName: Mproject-Test-AutoScallingGroup
AutoScalingGroupARN: arn:aws:autoscaling:us-east-1:905418258809:autoScalingGroup:5d148a79-b832-4bc2-b292-079852e976ad:autoScalingGroupName/Mproject-Test-AutoScallingGroup
ActivityId: 6c9ff2e4-618a-4547-9181-93ac39e0293e
Description: Terminating EC2 instance: i-0279ca6b3fb17553e
Cause: At 2024-06-19T18:58:11Z a user request force deleted AutoScaling group changing the desired capacity from 3 to 0. At 2024-06-19T18:58:15Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 0. At 2024-06-19T18:58:15Z instance i-0bf4b26925b5f6139 was selected for termination. At 2024-06-19T18:58:15Z instance i-02b5c41e0529253d9 was selected for termination. At 2024-06-19T18:58:15Z instance i-0279ca6b3fb17553e was selected for termination.
StartTime: 2024-06-19T18:58:15.729Z

Chapter-5

CONCLUSION

This project successfully demonstrates the practical application of several AWS services to build a scalable and efficient web hosting solution. By following the detailed steps outlined, we achieved the creation and configuration of AWS VPC SNS for notifications, custom AMIs for tailored server environments, EC2 instances for compute power, and a load balancer to distribute traffic effectively. Additionally, we implemented Auto Scaling Groups and Cloud Watch alarms to ensure the system can dynamically adjust to changing demand, maintaining performance while optimizing costs.

The completion of this project not only highlights the power and flexibility of AWS services but also underscores the importance of automation and monitoring in modern cloud infrastructure. Through careful planning and execution, we have created a resilient web hosting environment capable of adapting to real-world usage patterns, providing a robust foundation for future cloud-based projects