AWS Project Report

Web Server Deployment and Management using AWS Services

Submitted by:

Vaibhao Yenchalwar

A practical implementation using core AWS services

Note: This project demonstrates real-world AWS configuration including EC2, ALB, Auto Scaling, CloudWatch (CPU > 30%), and SNS with practical deployment values.

AWS PROJECT REPORT

Project Title:

Web Server Deployment and Management using AWS Services

(like EC2, ELB, Auto-Scaling, Cloud-Watch, SNS & IAM etc.)

👤 **Submitted by:**

Vaibhao Yenchalwar
👨‍🏫 **Mentor**: Shubham Sir
🏫 **Institute**: Fusion Software Institute, Pune

🎯 1. Objective

To build a **scalable and fault-tolerant web server architecture using AWS services** such as EC2, Auto Scaling, Elastic Load Balancer, CloudWatch, and SNS to **ensure high availability, automatic scaling, and proactive alerting.**

📘 2. Introduction

This project demonstrates how to **deploy and manage a scalable web server** infrastructure using core **AWS services** like EC2, ELB, Auto Scaling, CloudWatch, SNS, and monitoring tools to **ensure high availability and cost-efficiency**. We used a **default VPC**, simulated **CPU usage** with the **stress package**, and **observed Auto Scaling** in action based **on real-time CPU metrics.** This project **demonstrates end-to-end deployment and management of a web server that is:**

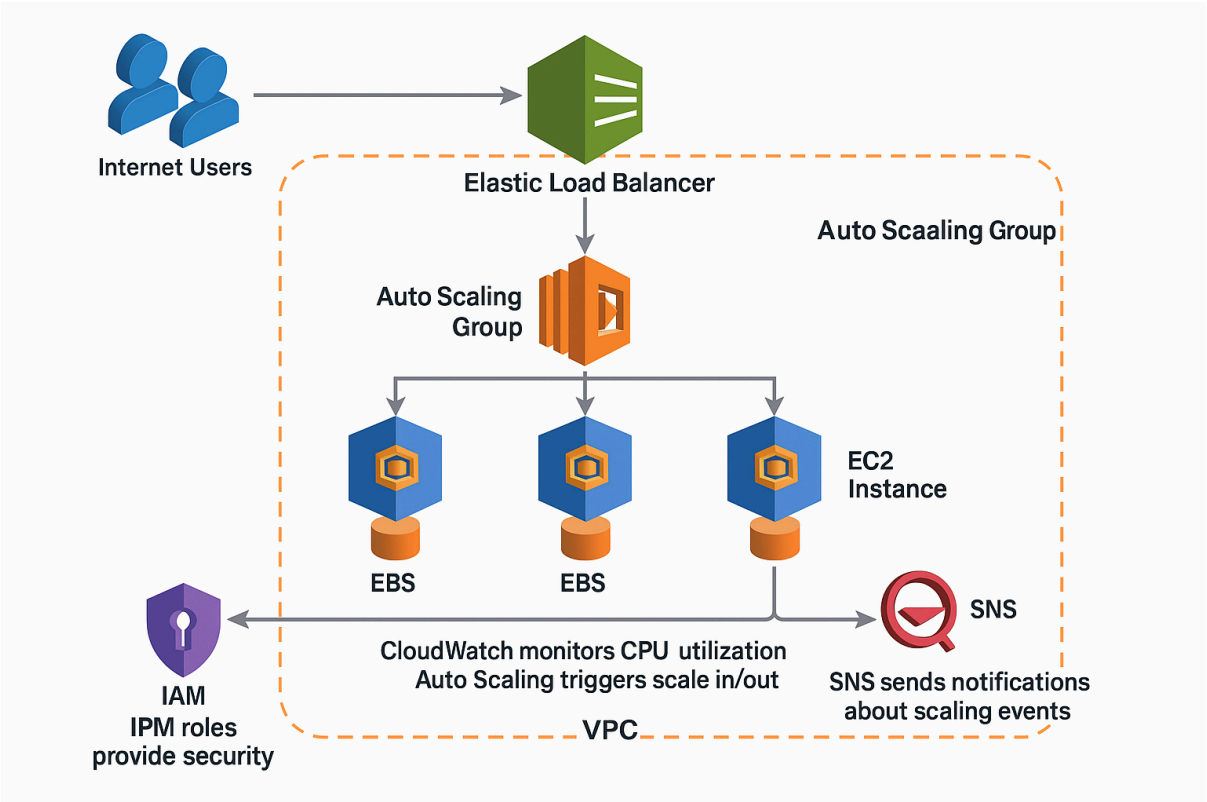
- **Auto-scalable**
- **Highly available**
- **Centrally monitored**
- **Notification-enabled**

🛠️3.AWS Services and Tools Used

- ✅ **Amazon EC2** – Launch virtual servers for web hosting
- ✅ **Amazon EBS** – Block storage attached to EC2
- ✅ **Amazon Machine Image (AMI)** – Snapshot of configured EC2
- ✅ **Elastic Load Balancer (ELB)** – Distributes incoming traffic
- ✅ **Auto Scaling Group** – Adjusts instance count based on load
- ✅ **SNS (Simple Notification Service)** – Sends email notifications
- ✅ **CloudWatch** – Monitors CPU usage and triggers alarms
- ✅ **IAM** – Manages permissions and roles
- ✅ **S3 (Optional)** – Backup storage
- ✅ **Git Bash** – For SSH connection and Linux commands
- ✅ **Stress tool** – To simulate **high CPU utilization**
- ✅ **Top command** – To monitor **live CPU usage**

---

4. Project Architecture 3D Diagram

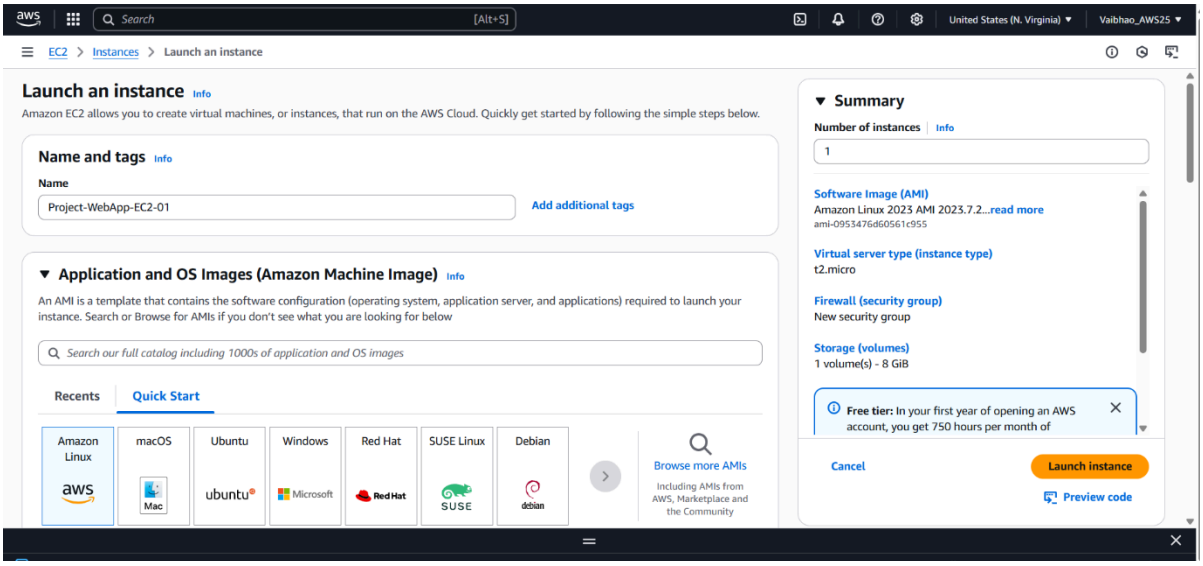


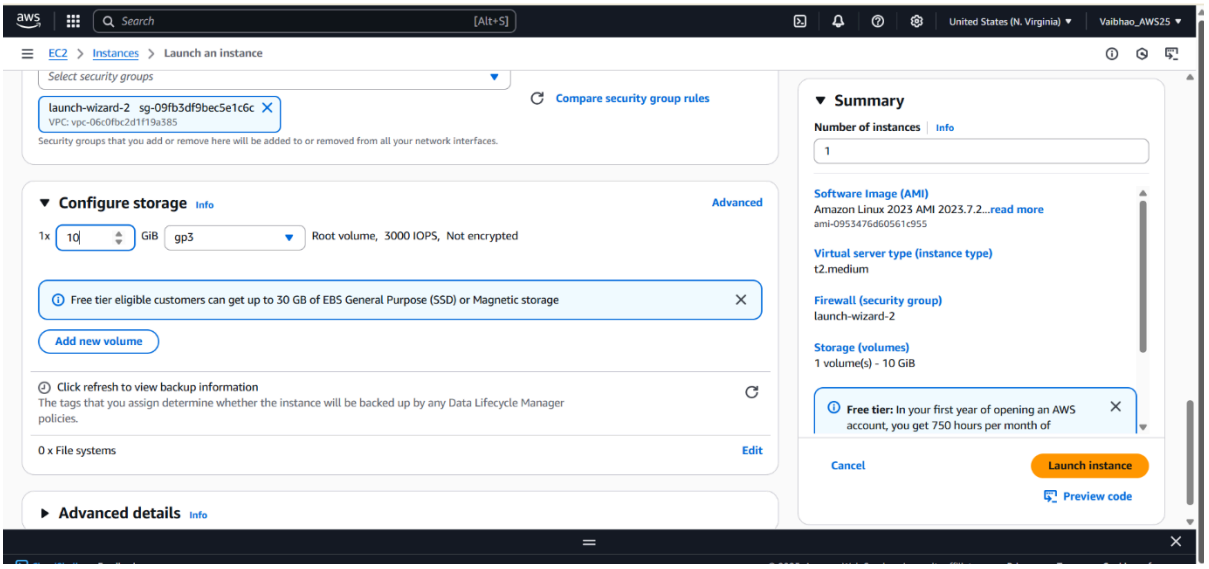
🧩 The Architecture includes EC2 behind a Load Balancer with **Auto Scaling**, monitored via **CloudWatch,** and alerts through **SNS.** A default VPC is used for networking.

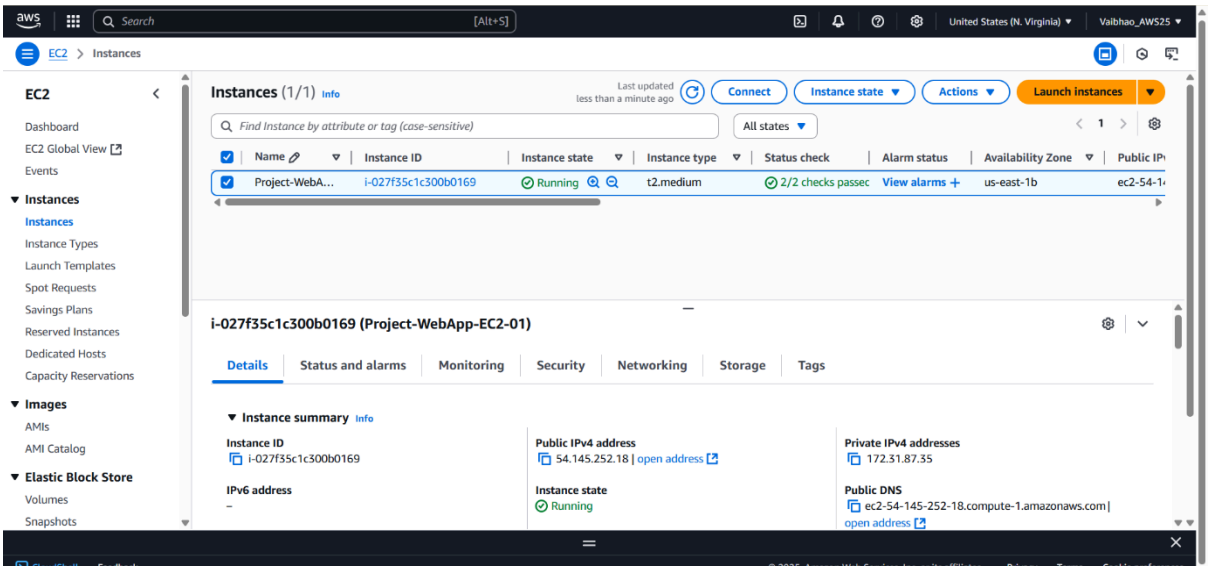
🔧 5. Implementation Steps

1. Create EC2 Instance

- **Type**: **t2.medium, Storage**: **10 GB**, Amazon **Linux OS**

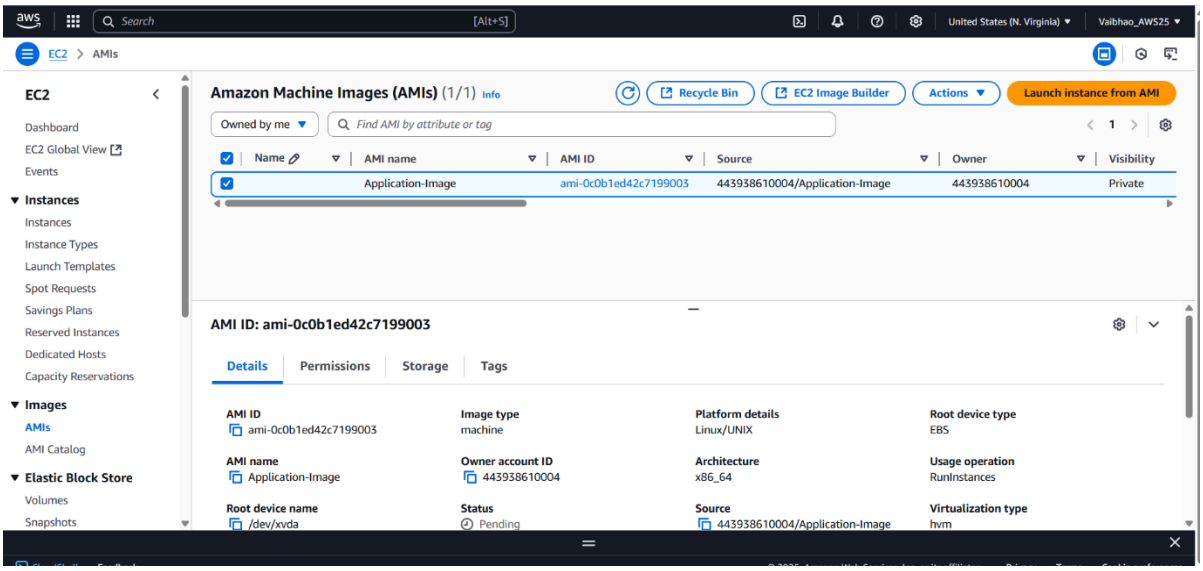






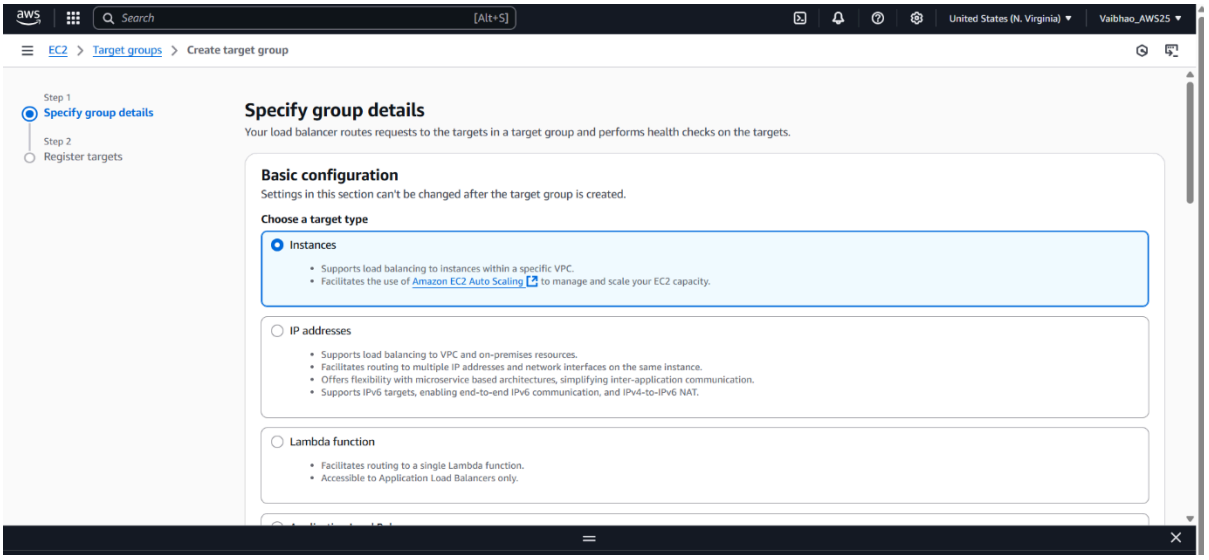
2. Create an AMI

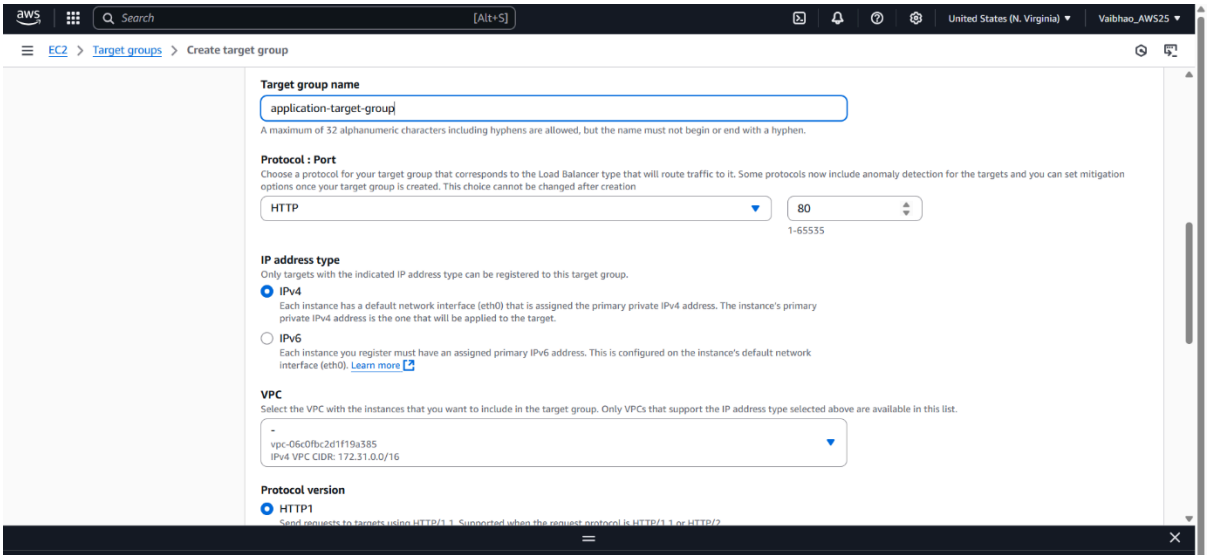
- Create an AMI from the instance to save configuration.
- Saved the EC2 instance as a reusable image.
- AMI will be used later Launch Template.

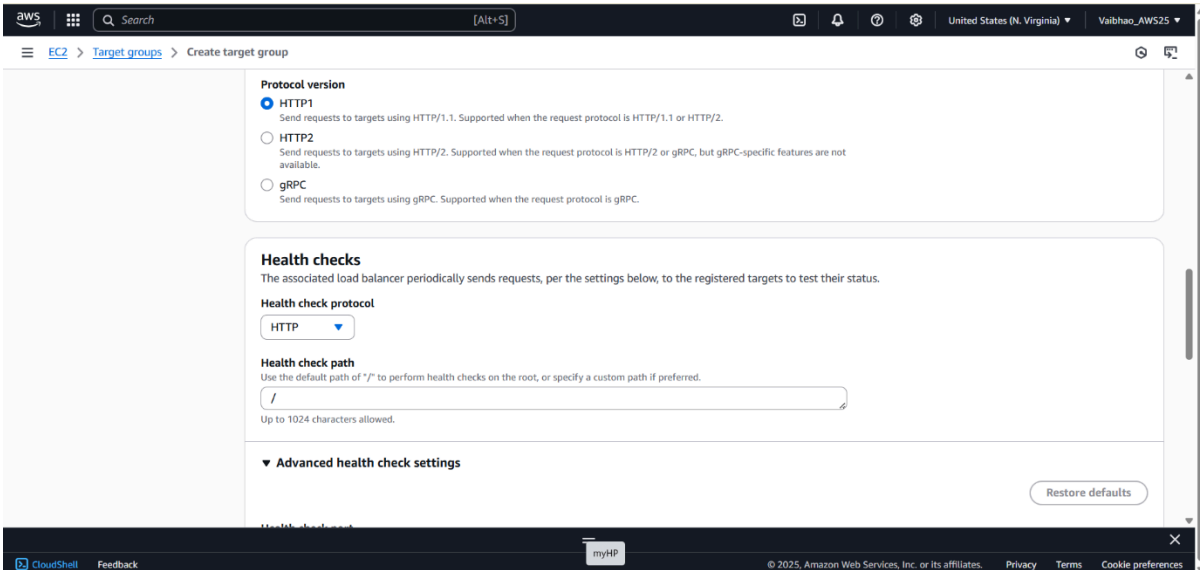


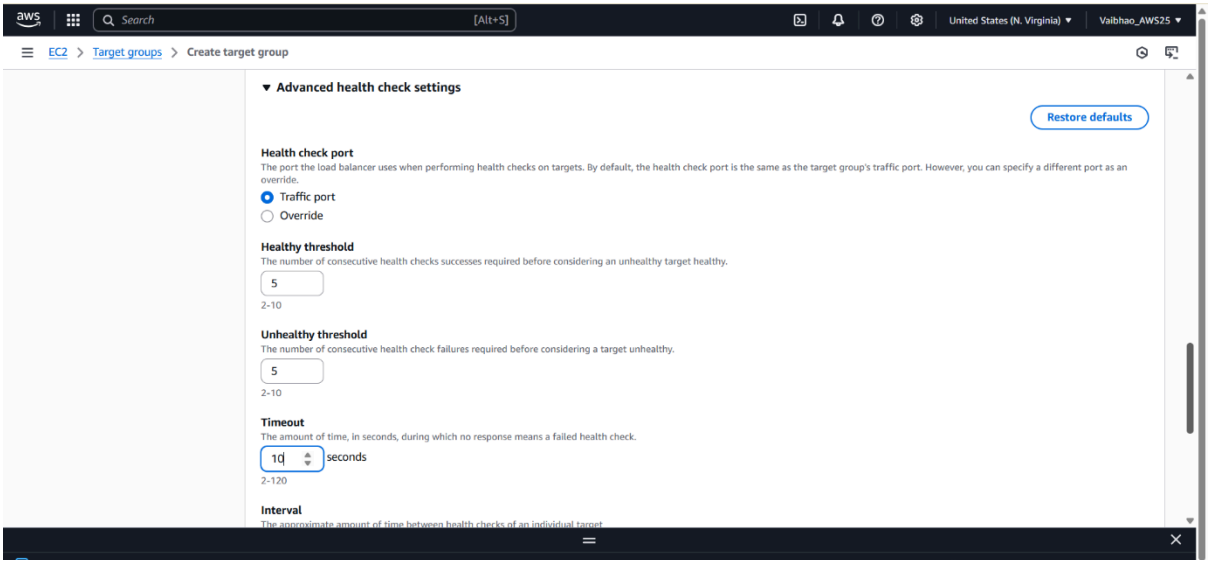
3. Create Target Group

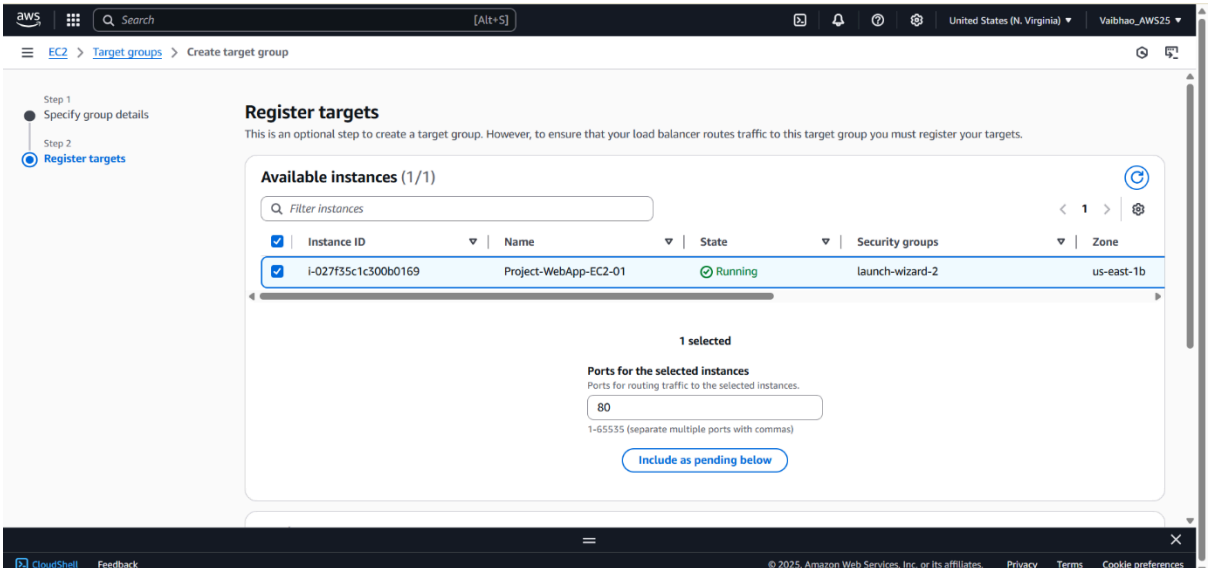
- Configured ELB to **forward requests to targets**

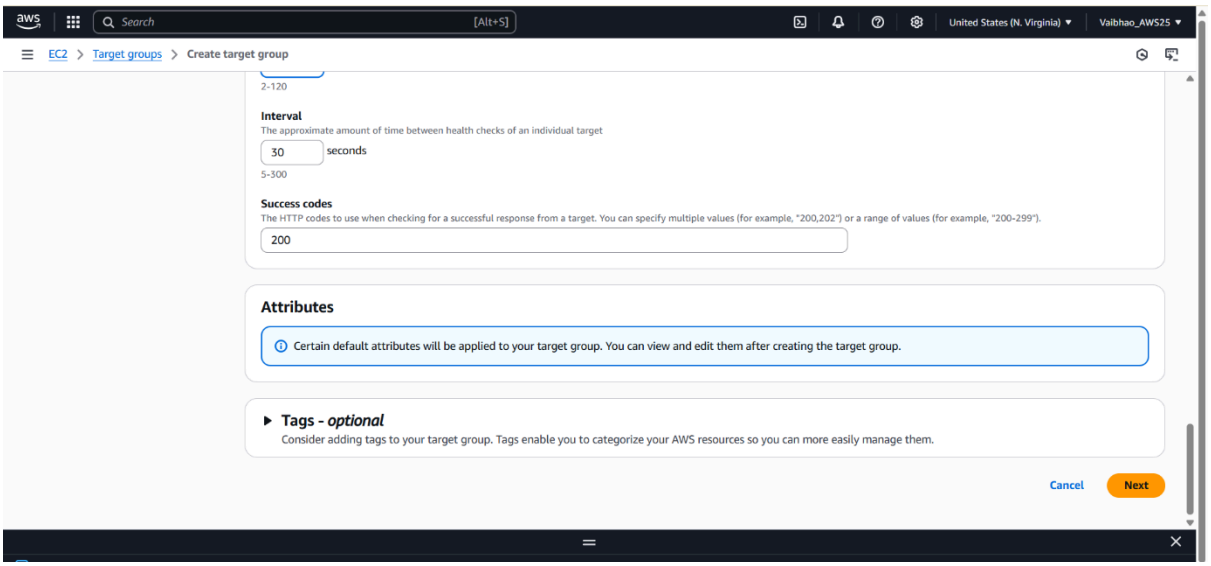


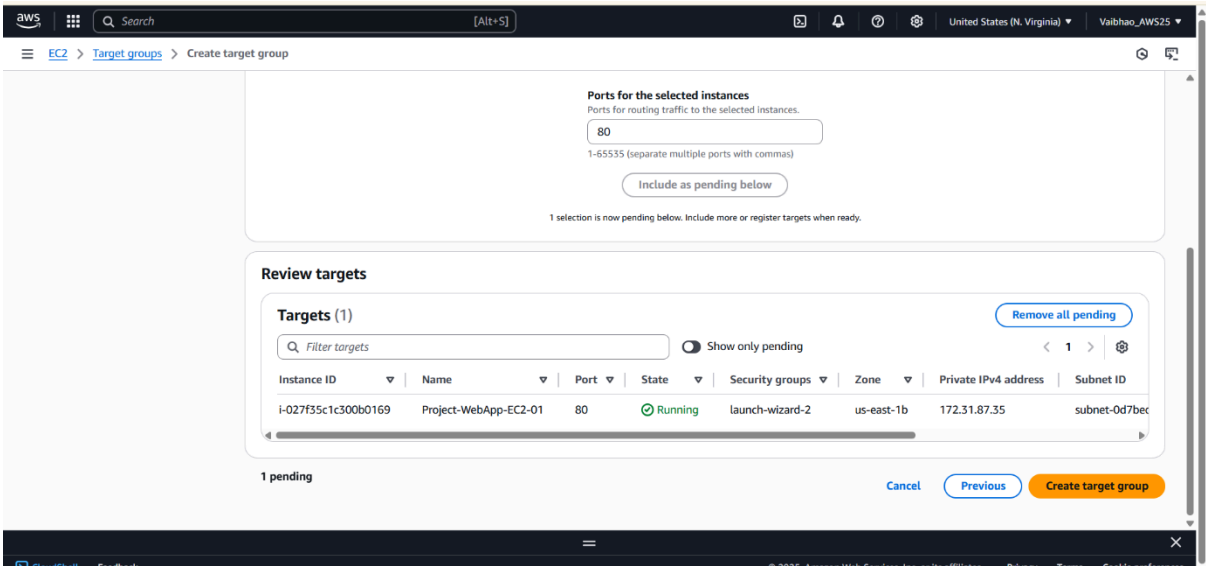


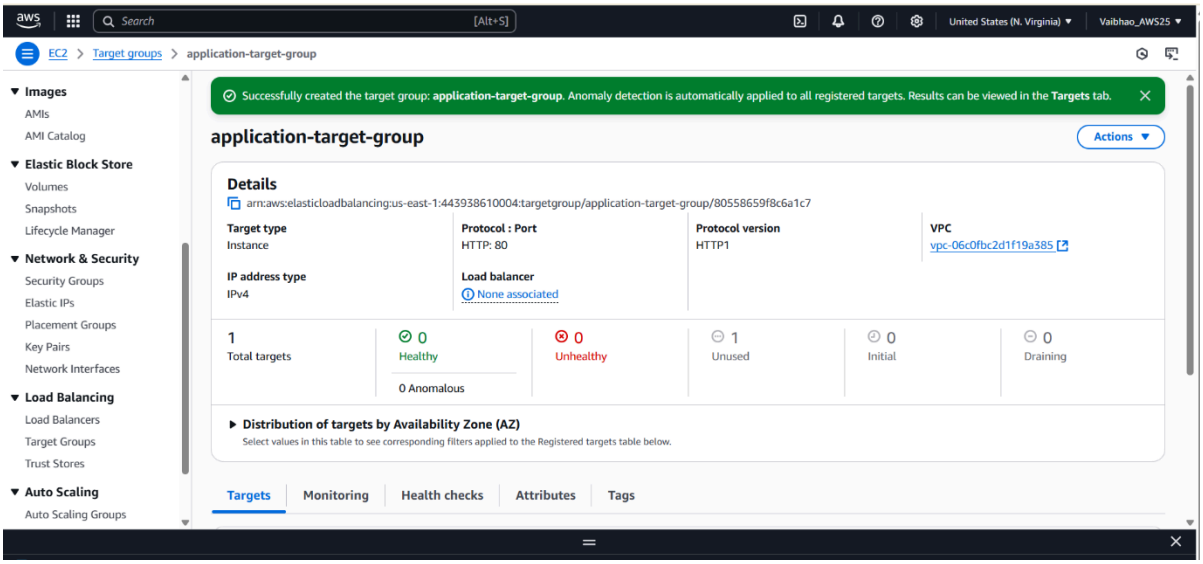

aws
Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2
Target groups
Create target group
Protocol version
HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.
Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.
Health check protocol
HTTP
Health check path
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.
/
Up to 1024 characters allowed.
Advanced health check settings
Restore defaults
CloudShell
Feedback
myHP
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2
Target groups
Create target group
Advanced health check settings
Restore defaults
Health check port
The port the load balancer uses when performing health checks on targets. By default, the health check port is the same as the target group's traffic port. However, you can specify a different port as an override.
Traffic port
Override
Healthy threshold
The number of consecutive health checks successes required before considering an unhealthy target healthy.
5
2-10
Unhealthy threshold
The number of consecutive health check failures required before considering a target unhealthy.
5
2-10
Timeout
The amount of time, in seconds, during which no response means a failed health check.
10
seconds
2-120
Interval


Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2
Target groups
Create target group
Step 1
Specify group details
Step 2
Register targets
Register targets
This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.
Available instances (1/1)
Filter instances
Instance ID
Name
State
Security groups
Zone
i-027f35c1c300b0169
Project-WebApp-EC2-01
Running
launch-wizard-2
us-east-1b
1 selected
Ports for the selected instances
Ports for routing traffic to the selected instances.
80
1-65535 (separate multiple ports with commas)
Include as pending below
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

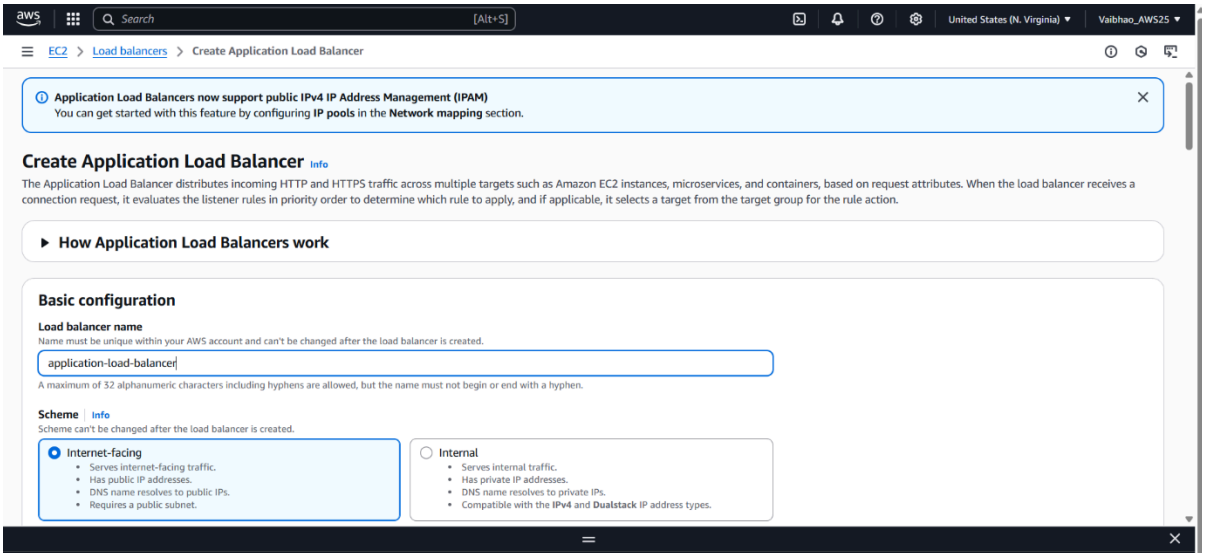

Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2
Target groups
Create target group
2-120
Interval
The approximate amount of time between health checks of an individual target
30
seconds
5-300
Success codes
The HTTP codes to use when checking for a successful response from a target. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-299").
200
Attributes
Certain default attributes will be applied to your target group. You can view and edit them after creating the target group.
Tags - optional
Consider adding tags to your target group. Tags enable you to categorize your AWS resources so you can more easily manage them.
Cancel
Next

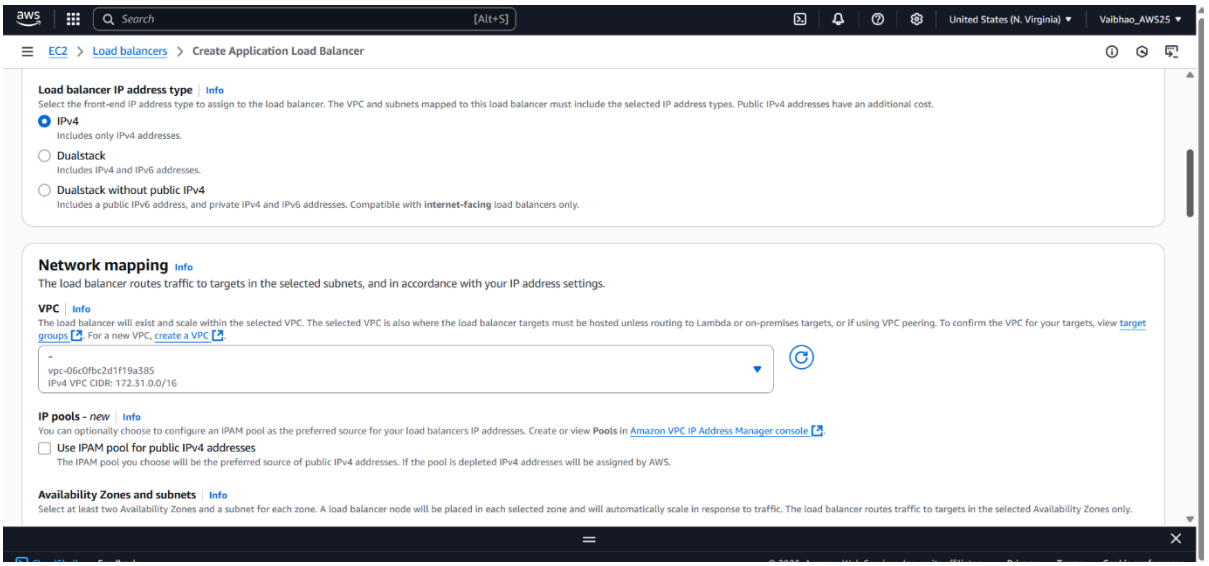

Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Target groups > Create target group
Ports for the selected instances
Ports for routing traffic to the selected instances.
80
1-65535 (separate multiple ports with commas)
Include as pending below
1 selection is now pending below. Include more or register targets when ready.
Review targets
Targets (1)
Remove all pending
Filter targets
Show only pending
Instance ID
Name
Port
State
Security groups
Zone
Private IPv4 address
Subnet ID
i-027f35c1c300b0169
Project-WebApp-EC2-01
80
Running
launch-wizard-2
us-east-1b
172.31.87.35
subnet-0d7bec
1 pending
Cancel
Previous
Create target group

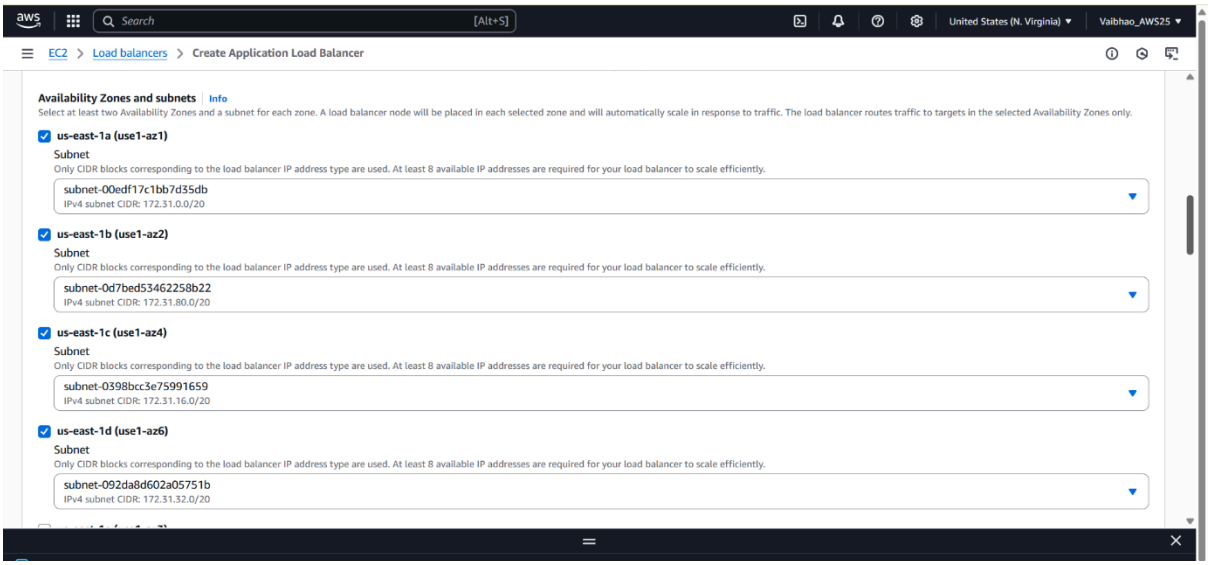

Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Target groups > application-target-group
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores
Auto Scaling
Auto Scaling Groups
Successfully created the target group: application-target-group. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the Targets tab.
application-target-group
Actions
Details
arn:aws:elasticloadbalancing:us-east-1:443938610004:targetgroup/application-target-group/80558659f8c6a1c7
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
vpc-06c0fbc2d1f19a385
IP address type
IPv4
Load balancer
None associated
1
Total targets
0
Healthy
0 Anomalous
0
Unhealthy
1
Unused
0
Initial
0
Draining
Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.
Targets
Monitoring
Health checks
Attributes
Tags

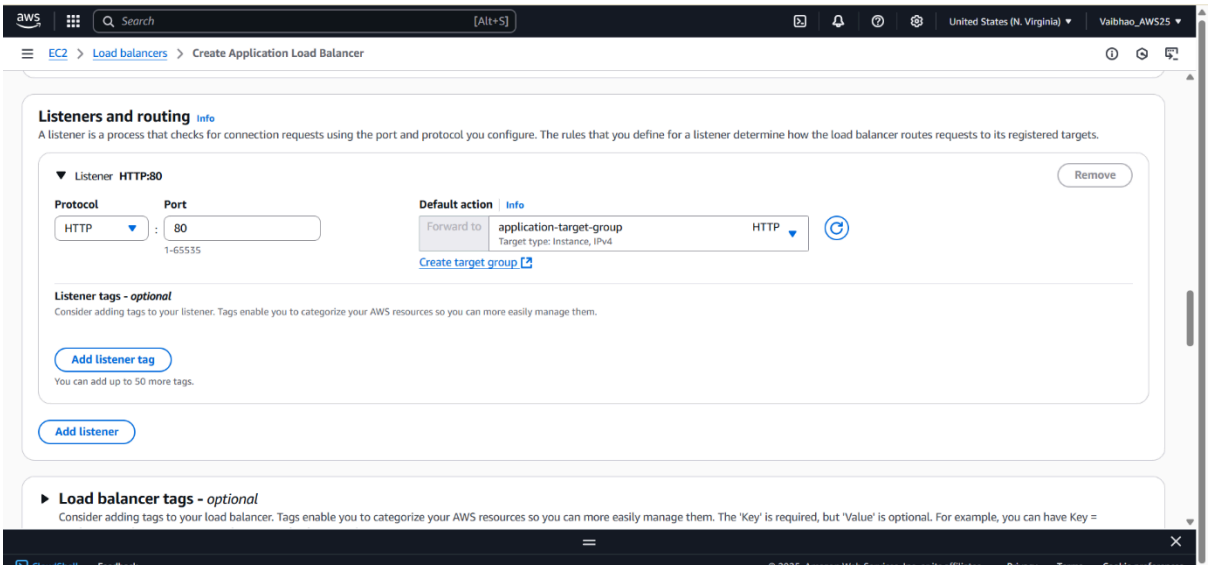
4. Create an ELB (Load Balancer)

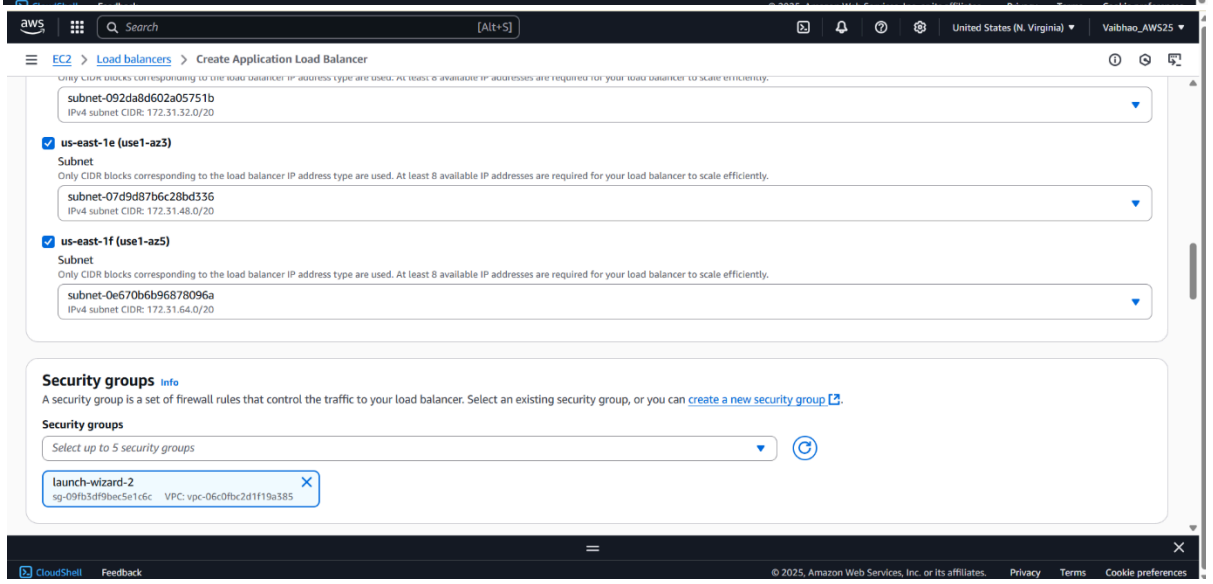
- Create a new **Target Group**, select target type as **Instance**, and protocol as **HTTP**.
- Configure the **Health Check** settings (protocol: HTTP, path: /).
- Register both EC2 instances in the target group.
- Review all configurations and click **"Create"** to launch
- Go to **Load Balancers** under **Load Balancing** and click **"Create Load Balancer."**
- Choose **"Application Load Balancer"** and provide a name.
- Select the **scheme** (internet-facing) and **IP address type** (IPv4).
- Select a **Security Group** to allow all traffic
- Configure listener settings (e.g., **HTTP on port 80**) and selected the created target group.
- Click on create **load balancer** to launch.

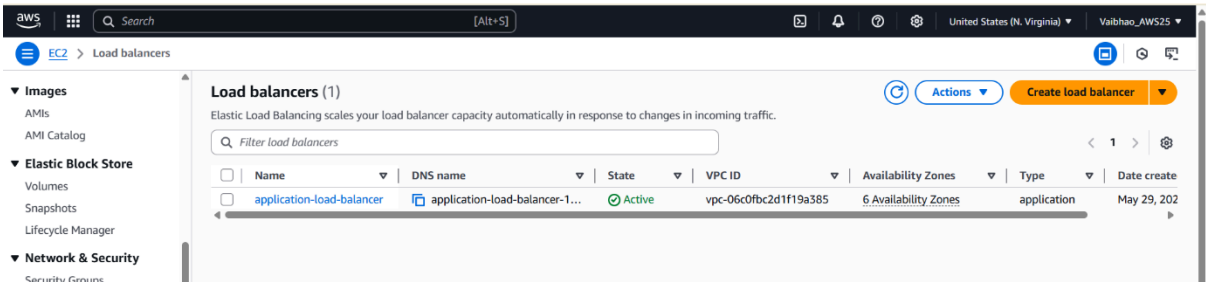



Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Load balancers > Create Application Load Balancer
Load balancer IP address type | Info
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.
IPv4
Includes only IPv4 addresses.
Dualstack
Includes IPv4 and IPv6 addresses.
Dualstack without public IPv4
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with internet-facing load balancers only.
Network mapping Info
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.
VPC | Info
The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view target groups. For a new VPC, create a VPC.
-
vpc-06c0fbc2d1f19a385
IPv4 VPC CIDR: 172.31.0.0/16
IP pools - new | Info
You can optionally choose to configure an IPAM pool as the preferred source for your load balancers IP addresses. Create or view Pools in Amazon VPC IP Address Manager console.
Use IPAM pool for public IPv4 addresses
The IPAM pool you choose will be the preferred source of public IPv4 addresses. If the pool is depleted IPv4 addresses will be assigned by AWS.
Availability Zones and subnets | Info
Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.


Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Load balancers > Create Application Load Balancer
Availability Zones and subnets | Info
Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.
us-east-1a (use1-az1)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-00edf17c1bb7d35db
IPv4 subnet CIDR: 172.31.0.0/20
us-east-1b (use1-az2)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-0d7bed53462258b22
IPv4 subnet CIDR: 172.31.80.0/20
us-east-1c (use1-az4)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-0398bcc3e75991659
IPv4 subnet CIDR: 172.31.16.0/20
us-east-1d (use1-az6)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-092da8d602a05751b
IPv4 subnet CIDR: 172.31.32.0/20

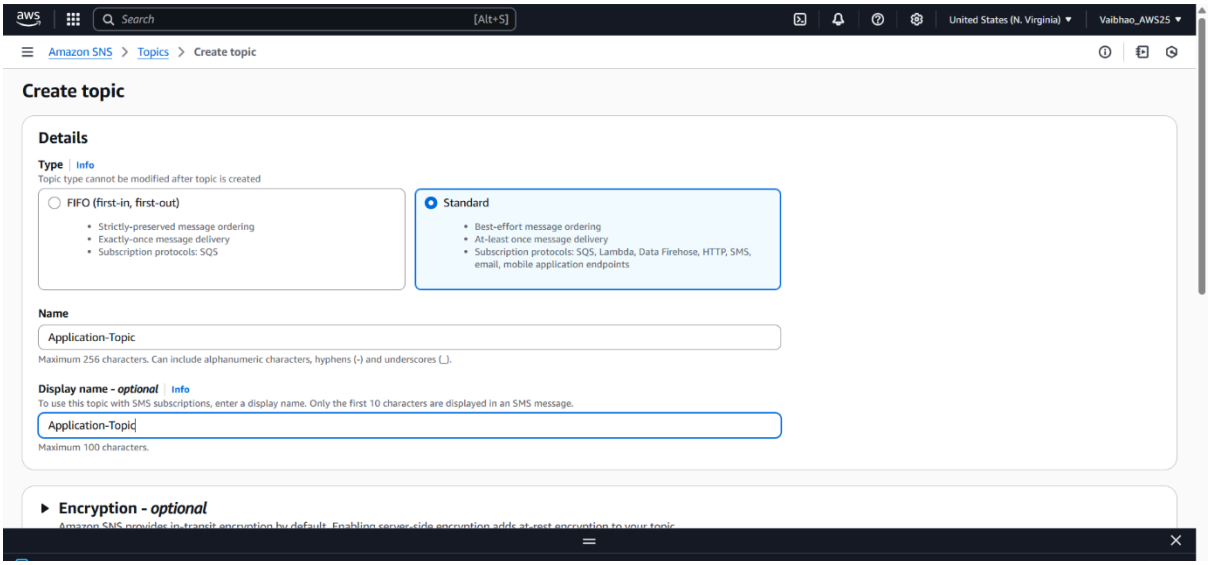

Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Load balancers > Create Application Load Balancer
Listeners and routing Info
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.
Listener HTTP:80
Remove
Protocol
Port
HTTP
80
1-65535
Default action Info
Forward to
application-target-group
HTTP
Target type: Instance, IPv4
Create target group
Listener tags - optional
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.
Add listener tag
You can add up to 50 more tags.
Add listener
Load balancer tags - optional
Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them. The 'Key' is required, but 'Value' is optional. For example, you can have Key =

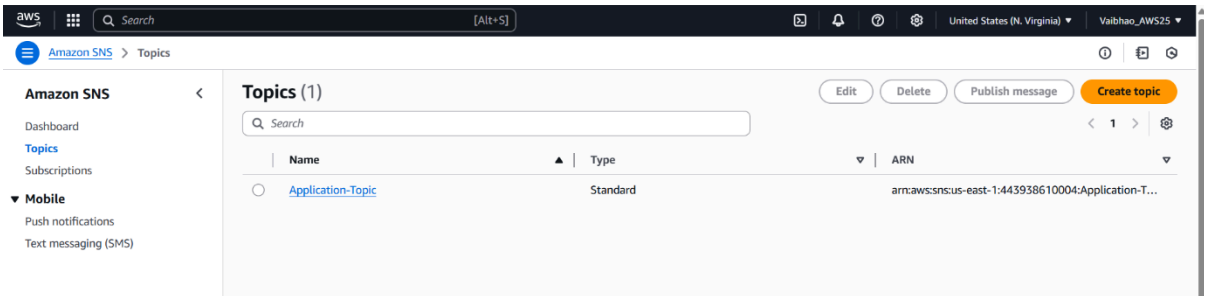

Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Load balancers > Create Application Load Balancer
subnet-092da8d602a05751b
IPv4 subnet CIDR: 172.31.32.0/20
us-east-1e (use1-az3)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-07d9d87b6c28bd336
IPv4 subnet CIDR: 172.31.48.0/20
us-east-1f (use1-az5)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-0e670b6b96878096a
IPv4 subnet CIDR: 172.31.64.0/20
Security groups Info
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group.
Security groups
Select up to 5 security groups
launch-wizard-2
sg-09fb3df9bec5e1c6c VPC: vpc-06c0fbc2d1f19a385
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Load balancers
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Load balancers (1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter load balancers
Name
DNS name
State
VPC ID
Availability Zones
Type
Date create
application-load-balancer
application-load-balancer-1...
Active
vpc-06c0fbc2d1f19a385
6 Availability Zones
application
May 29, 202

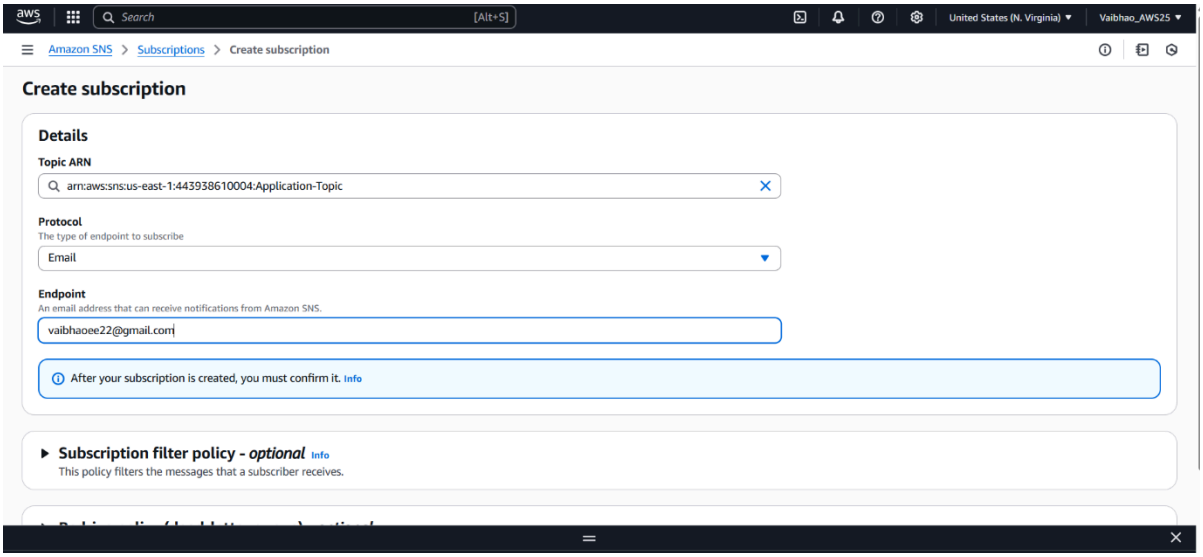
5. Configured SNS

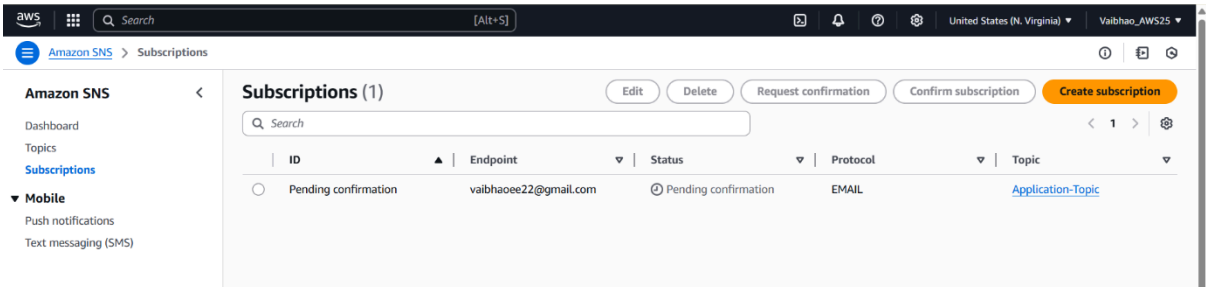
- Created **topic**

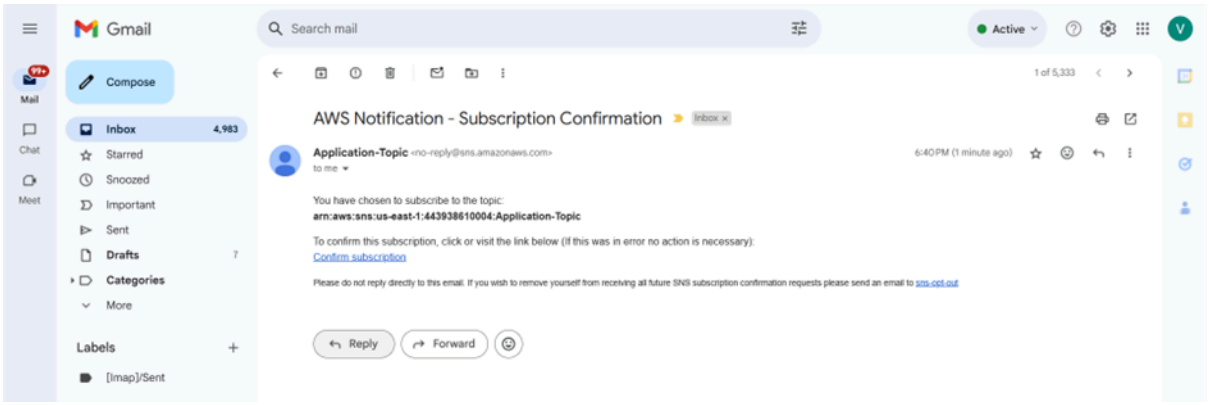


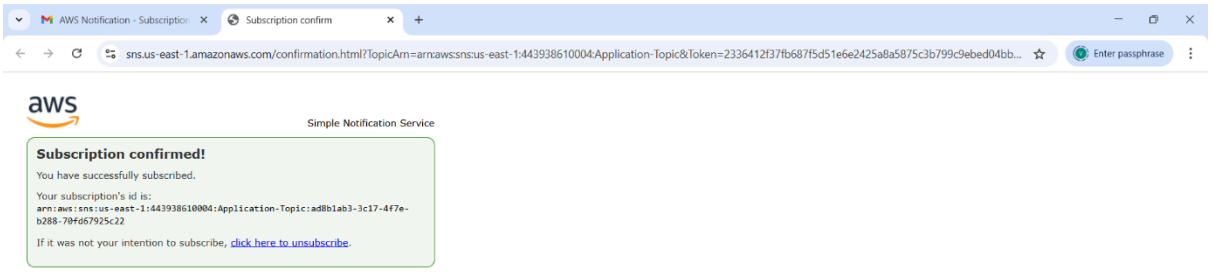


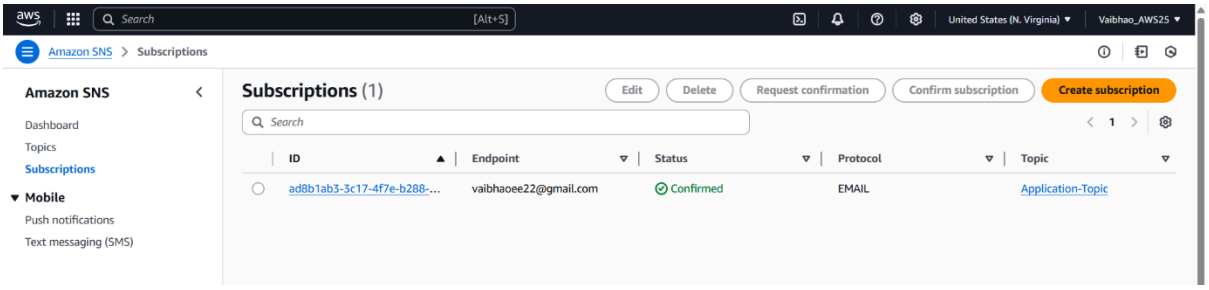
- Subscribed with **email for alerts**


Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
Amazon SNS > Subscriptions > Create subscription
Create subscription
Details
Topic ARN
arn:aws:sns:us-east-1:443938610004:Application-Topic
Protocol
The type of endpoint to subscribe
Email
Endpoint
An email address that can receive notifications from Amazon SNS.
vaibhaoee22@gmail.com
After your subscription is created, you must confirm it. Info
Subscription filter policy - optional Info
This policy filters the messages that a subscriber receives.


Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
Amazon SNS > Subscriptions
Amazon SNS
Dashboard
Topics
Subscriptions
Mobile
Push notifications
Text messaging (SMS)
Subscriptions (1)
Edit
Delete
Request confirmation
Confirm subscription
Create subscription
Search
ID
Endpoint
Status
Protocol
Topic
Pending confirmation
vaibhaoee22@gmail.com
Pending confirmation
EMAIL
Application-Topic

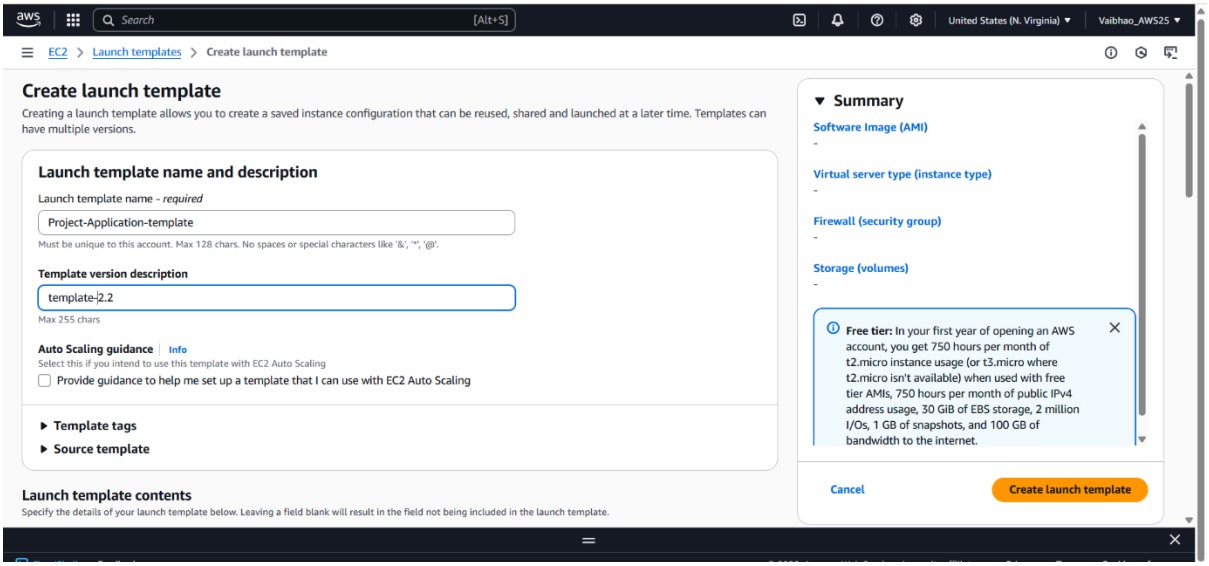

Gmail
Search mail
Active
Compose
Inbox 4,983
Starred
Snoozed
Important
Sent
Drafts
Categories
More
Labels
[Imap]/Sent
AWS Notification - Subscription Confirmation
Application-Topic
You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:443938610004:Application-Topic
To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription
Reply
Forward

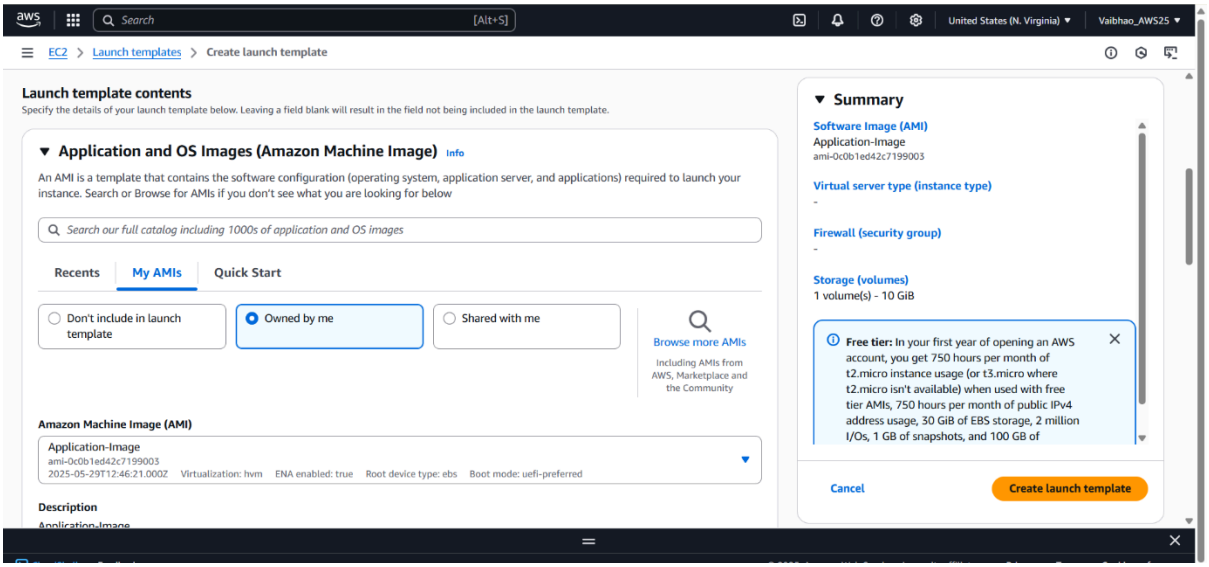


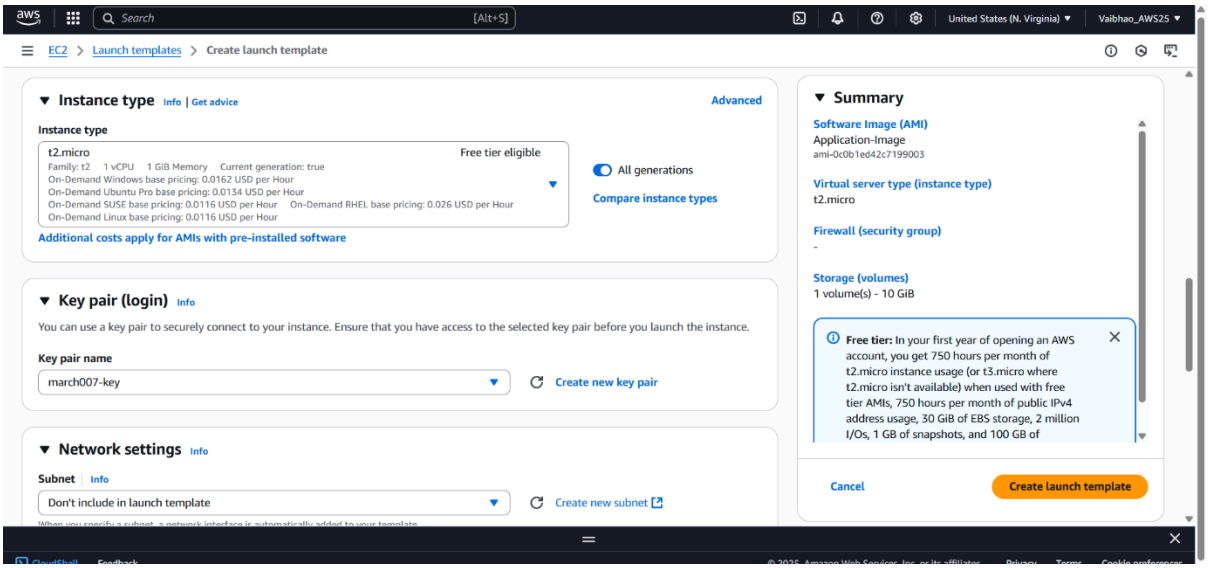


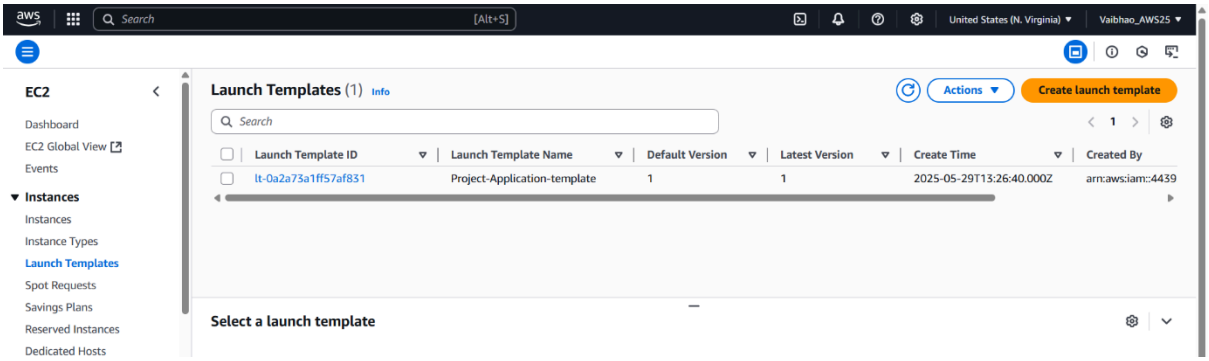
6. Created Launch Template

- Create a Launch Template using previously created AMI.
- Set instance type and configure security group.
- Reference this template in Auto Scaling


Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Launch templates > Create launch template
Create launch template
Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.
Launch template name and description
Launch template name - required
Project-Application-template
Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.
Template version description
template-2.2
Max 255 chars
Auto Scaling guidance Info
Select this if you intend to use this template with EC2 Auto Scaling
Provide guidance to help me set up a template that I can use with EC2 Auto Scaling
Template tags
Source template
Launch template contents
Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.
Summary
Software Image (AMI)
-
Virtual server type (instance type)
-
Firewall (security group)
-
Storage (volumes)
-
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Create launch template

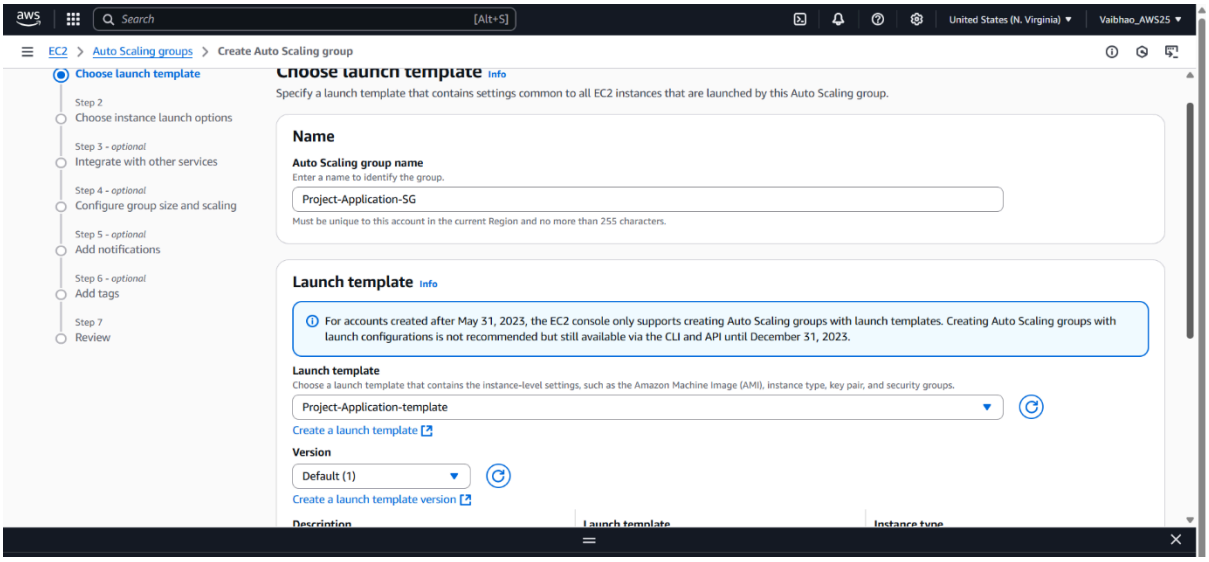

Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Launch templates > Create launch template
Launch template contents
Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.
Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Search our full catalog including 1000s of application and OS images
Recents
My AMIs
Quick Start
Don't include in launch template
Owned by me
Shared with me
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Application-Image
ami-0c0b1ed42c7199003
2025-05-29T12:46:21.000Z Virtualization: hvm ENA enabled: true Root device type: ebs Boot mode: uefi-preferred
Description
Application-Image
Summary
Software Image (AMI)
Application-Image
ami-0c0b1ed42c7199003
Virtual server type (instance type)
-
Firewall (security group)
-
Storage (volumes)
1 volume(s) - 10 GiB
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of
Cancel
Create launch template

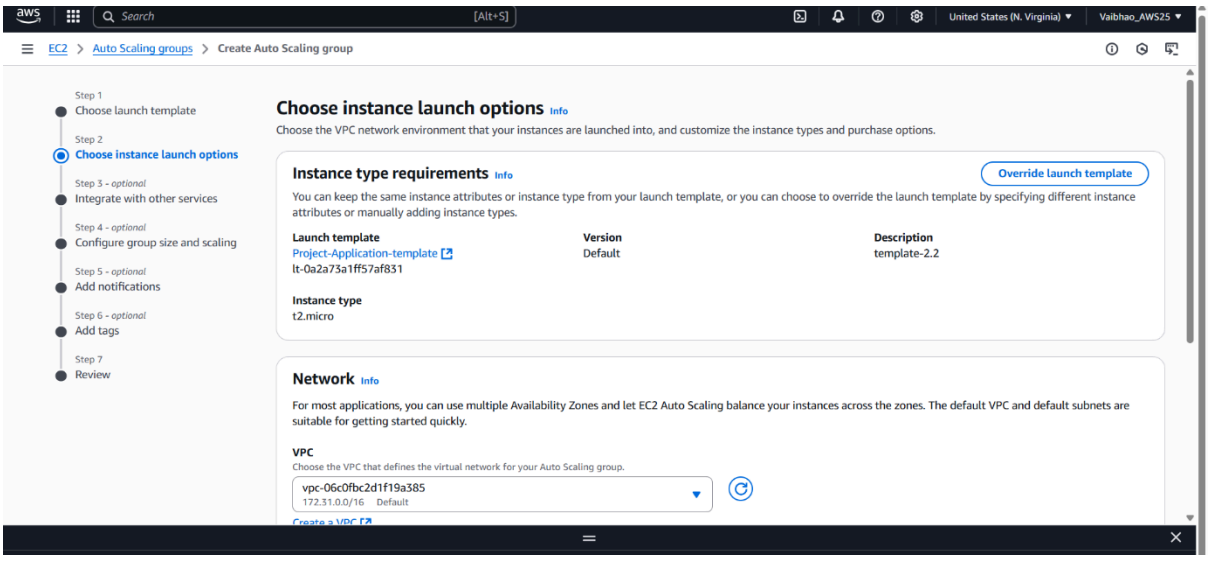

Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Launch templates > Create launch template
Instance type Info | Get advice
Advanced
Instance type
t2.micro
Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name
march007-key
Create new key pair
Network settings Info
Subnet Info
Don't include in launch template
Create new subnet
Summary
Software Image (AMI)
Application-Image
ami-0c0b1ed42c7199003
Virtual server type (instance type)
t2.micro
Firewall (security group)
-
Storage (volumes)
1 volume(s) - 10 GiB
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of
Cancel
Create launch template

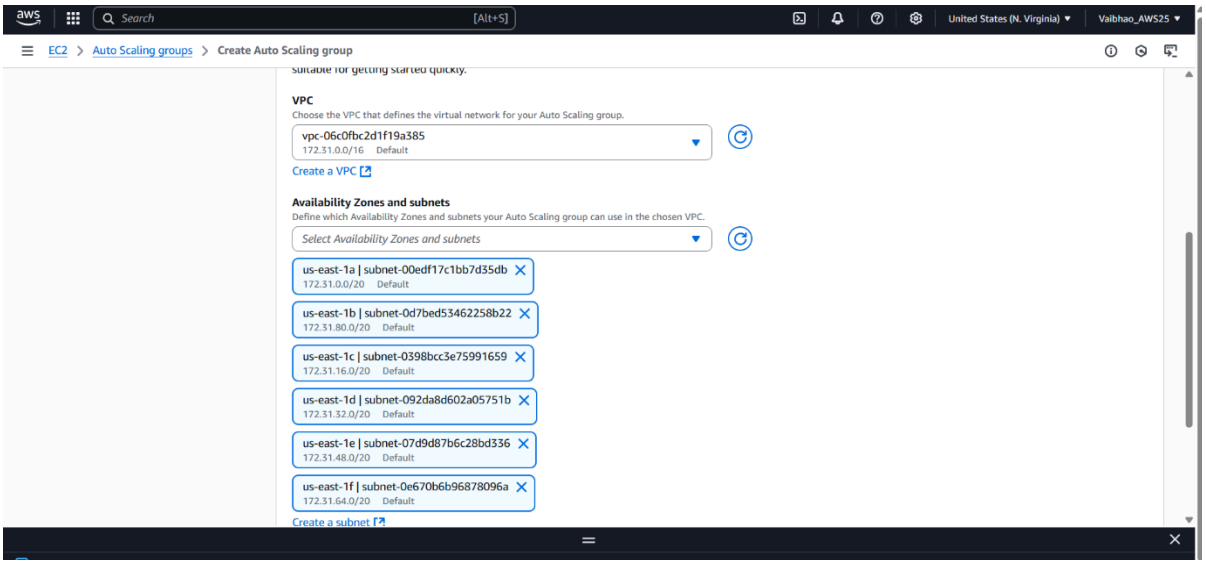

Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Launch Templates (1) Info
Actions
Create launch template
Search
Launch Template ID
Launch Template Name
Default Version
Latest Version
Create Time
Created By
lt-0a2a73a1ff57af831
Project-Application-template
1
1
2025-05-29T13:26:40.000Z
arn:aws:iam::4439
Select a launch template

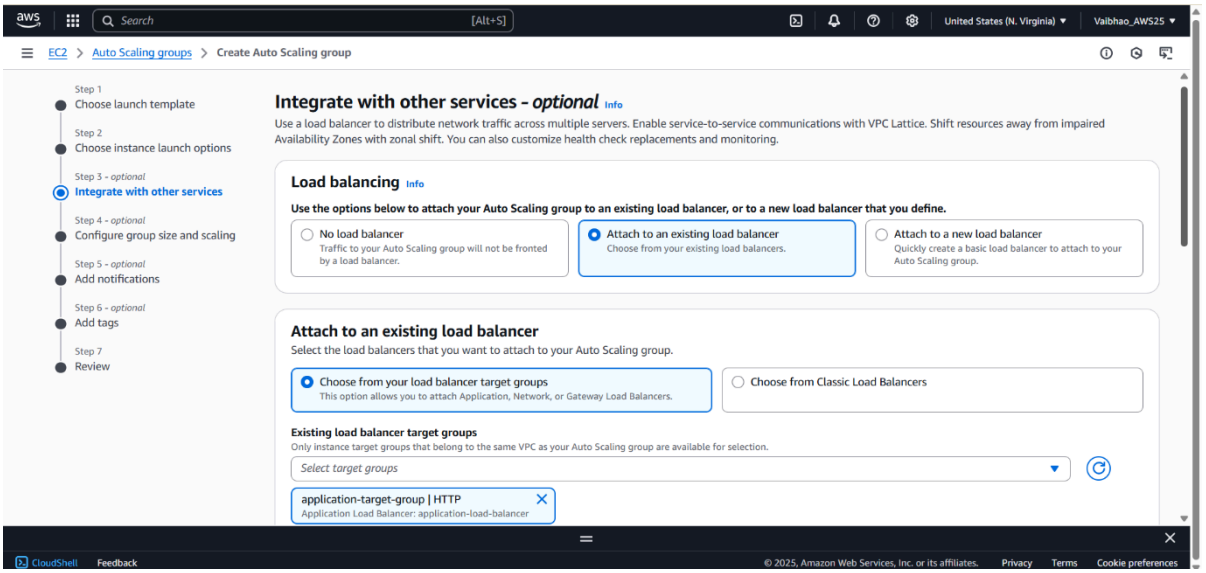
7. Auto Scaling Group

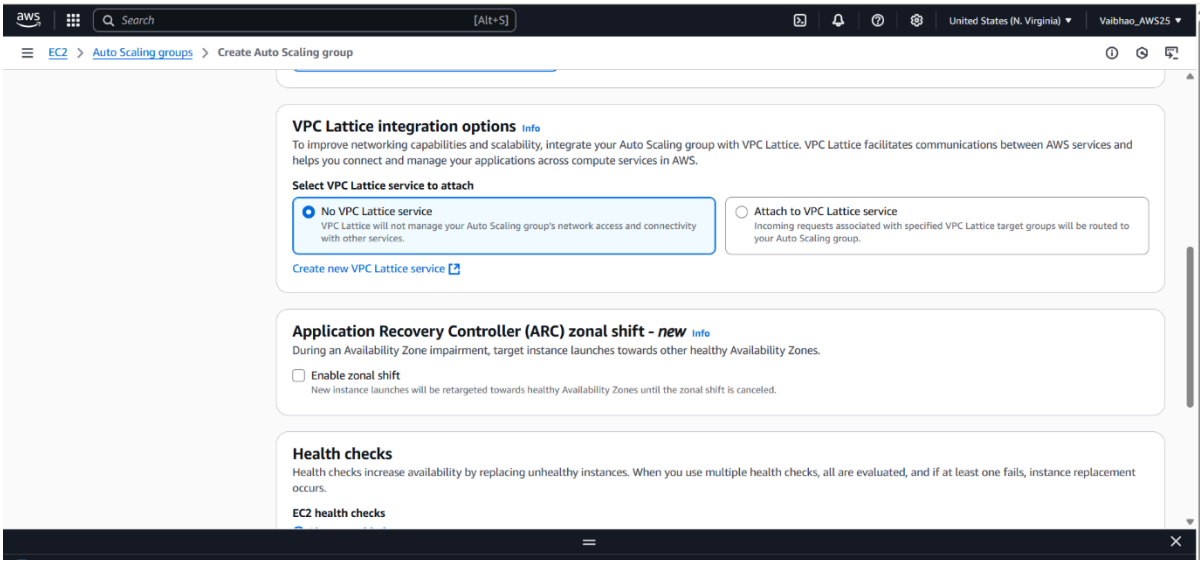
1. Choose launch template
2. Choose launch instance option
3. Integrate with other services
4. Configure group size and scaling
    - **Desired Capacity = 1**
    - **Minimum Desired capacity = 1**
    - **Maximum Desired capacity = 5**
5. Add Notification
6. Add tags (optional)
7. Review
8. Connected to Launch Template & Target Group

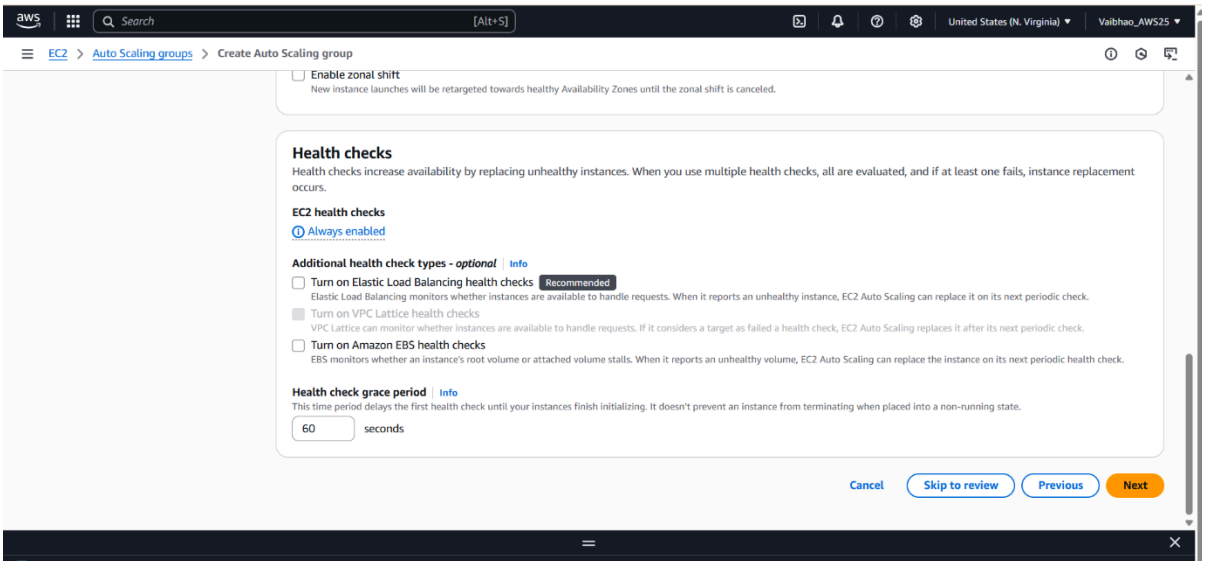


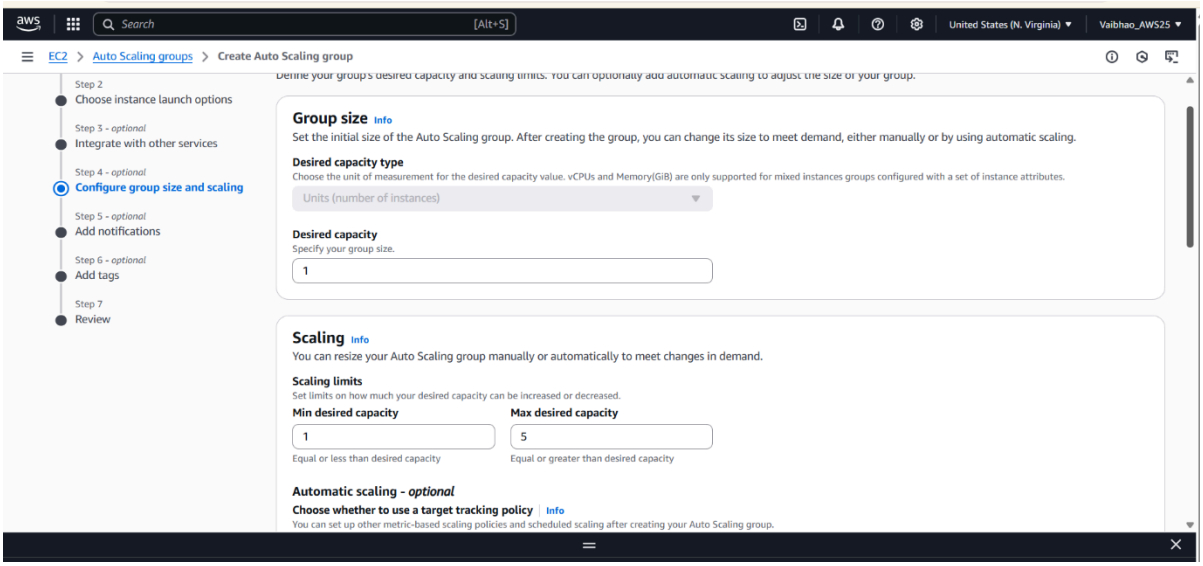



Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Auto Scaling groups > Create Auto Scaling group
VPC
Choose the VPC that defines the virtual network for your Auto Scaling group.
vpc-06c0fbc2d1f19a385
172.31.0.0/16 Default
Create a VPC
Availability Zones and subnets
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.
Select Availability Zones and subnets
us-east-1a | subnet-00edf17c1bb7d35db
172.31.0.0/20 Default
us-east-1b | subnet-0d7bed53462258b22
172.31.80.0/20 Default
us-east-1c | subnet-0398bcc3e75991659
172.31.16.0/20 Default
us-east-1d | subnet-092da8d602a05751b
172.31.32.0/20 Default
us-east-1e | subnet-07d9d87b6c28bd336
172.31.48.0/20 Default
us-east-1f | subnet-0e670b6b96878096a
172.31.64.0/20 Default
Create a subnet

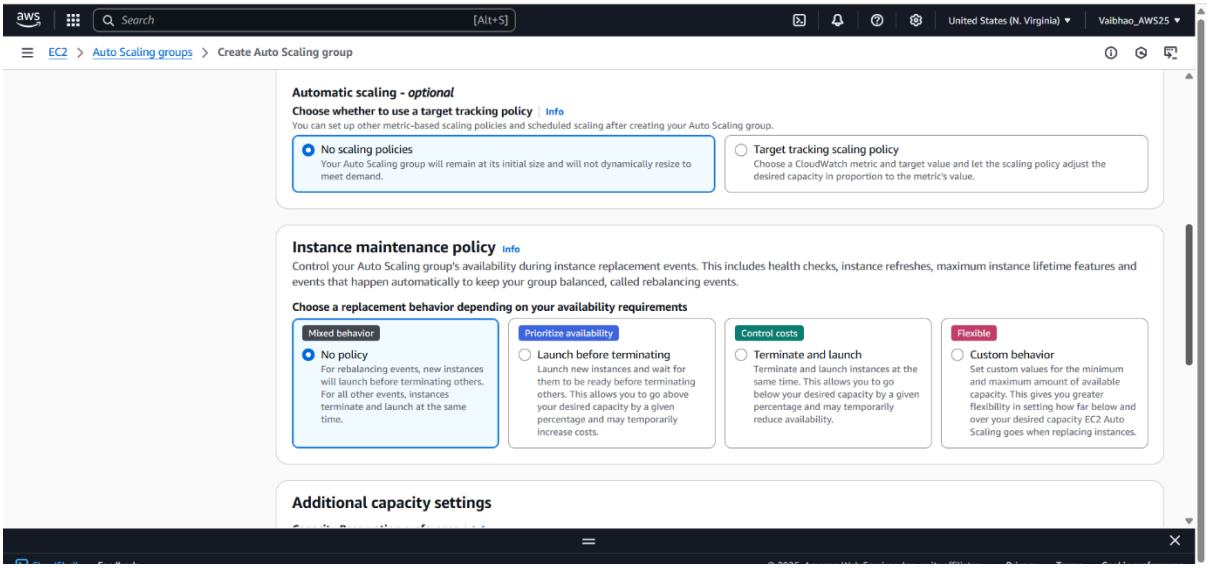

Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Auto Scaling groups > Create Auto Scaling group
Step 1
Choose launch template
Step 2
Choose instance launch options
Step 3 - optional
Integrate with other services
Step 4 - optional
Configure group size and scaling
Step 5 - optional
Add notifications
Step 6 - optional
Add tags
Step 7
Review
Integrate with other services - optional Info
Use a load balancer to distribute network traffic across multiple servers. Enable service-to-service communications with VPC Lattice. Shift resources away from impaired Availability Zones with zonal shift. You can also customize health check replacements and monitoring.
Load balancing Info
Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.
No load balancer
Traffic to your Auto Scaling group will not be fronted by a load balancer.
Attach to an existing load balancer
Choose from your existing load balancers.
Attach to a new load balancer
Quickly create a basic load balancer to attach to your Auto Scaling group.
Attach to an existing load balancer
Select the load balancers that you want to attach to your Auto Scaling group.
Choose from your load balancer target groups
This option allows you to attach Application, Network, or Gateway Load Balancers.
Choose from Classic Load Balancers
Existing load balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.
Select target groups
application-target-group | HTTP
Application Load Balancer: application-load-balancer
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

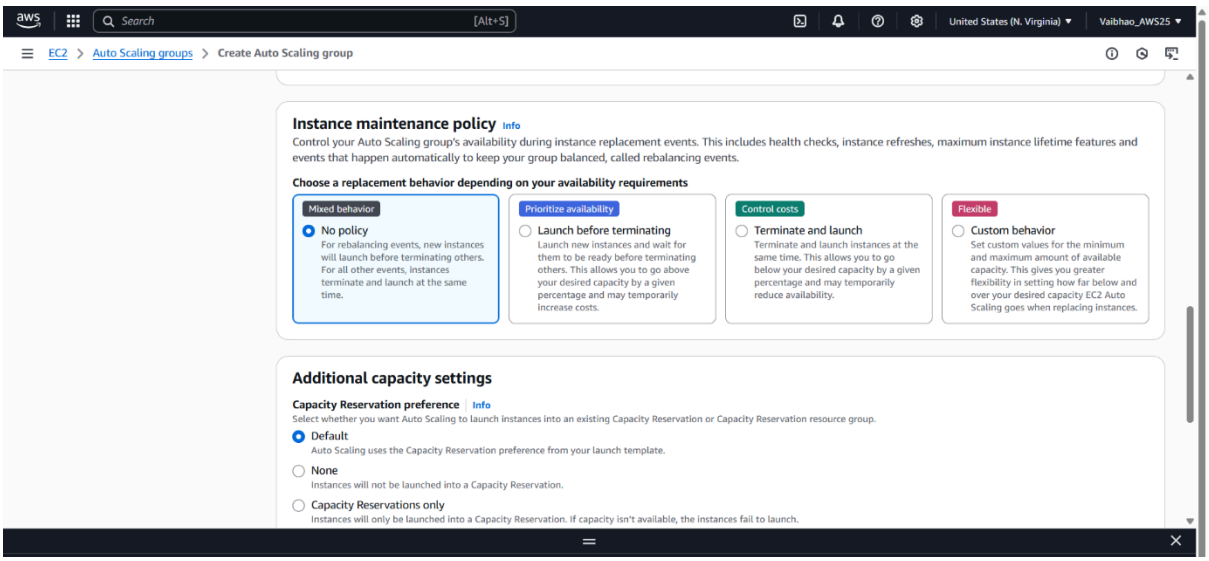

aws
Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2
Auto Scaling groups
Create Auto Scaling group
VPC Lattice integration options Info
To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.
Select VPC Lattice service to attach
No VPC Lattice service
VPC Lattice will not manage your Auto Scaling group's network access and connectivity with other services.
Attach to VPC Lattice service
Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.
Create new VPC Lattice service
Application Recovery Controller (ARC) zonal shift - new Info
During an Availability Zone impairment, target instance launches towards other healthy Availability Zones.
Enable zonal shift
New instance launches will be retargeted towards healthy Availability Zones until the zonal shift is canceled.
Health checks
Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.
EC2 health checks

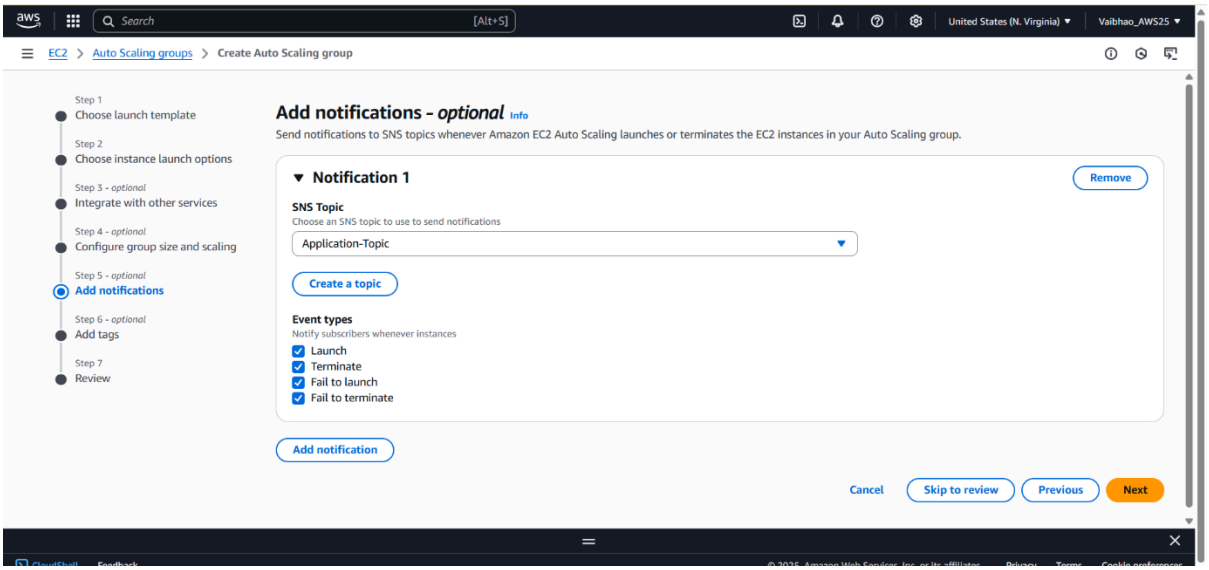

aws
Search
[Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2
Auto Scaling groups
Create Auto Scaling group
Enable zonal shift
New instance launches will be retargeted towards healthy Availability Zones until the zonal shift is canceled.
Health checks
Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.
EC2 health checks
Always enabled
Additional health check types - optional Info
Turn on Elastic Load Balancing health checks
Recommended
Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.
Turn on VPC Lattice health checks
VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.
Turn on Amazon EBS health checks
EBS monitors whether an instance's root volume or attached volume stalls. When it reports an unhealthy volume, EC2 Auto Scaling can replace the instance on its next periodic health check.
Health check grace period Info
This time period delays the first health check until your instances finish initializing. It doesn't prevent an instance from terminating when placed into a non-running state.
60
seconds
Cancel
Skip to review
Previous
Next

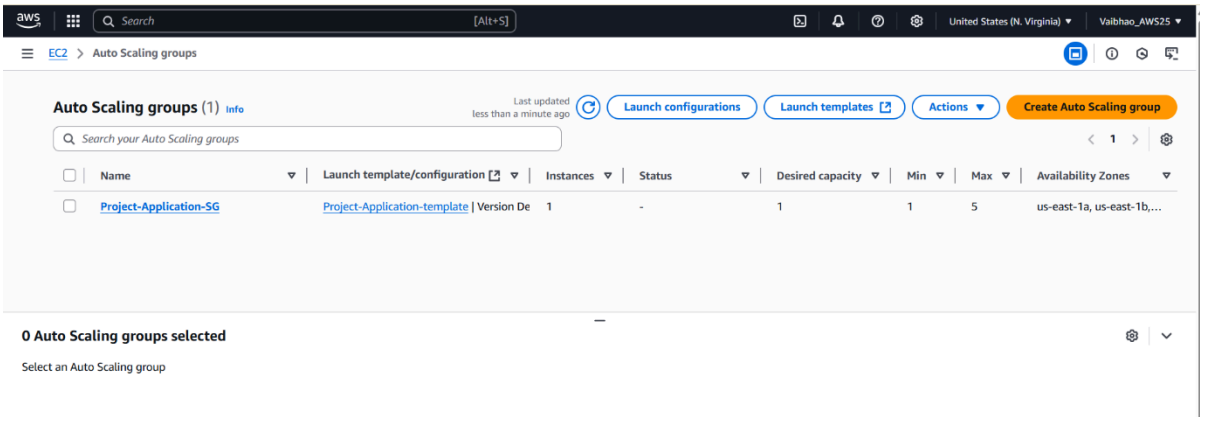


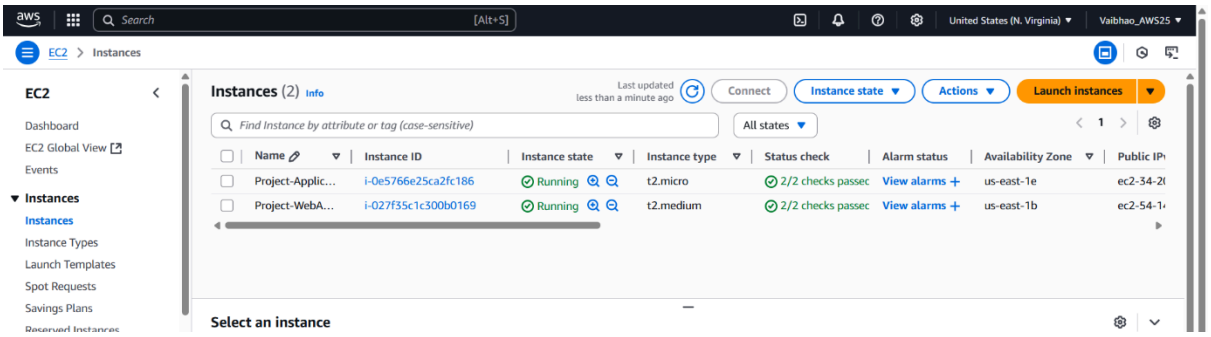
- Configured policy to **scale out if CPU > 30%**




Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Auto Scaling groups > Create Auto Scaling group
Instance maintenance policy Info
Control your Auto Scaling group's availability during instance replacement events. This includes health checks, instance refreshes, maximum instance lifetime features and events that happen automatically to keep your group balanced, called rebalancing events.
Choose a replacement behavior depending on your availability requirements
Mixed behavior
No policy
For rebalancing events, new instances will launch before terminating others. For all other events, instances terminate and launch at the same time.
Prioritize availability
Launch before terminating
Launch new instances and wait for them to be ready before terminating others. This allows you to go above your desired capacity by a given percentage and may temporarily increase costs.
Control costs
Terminate and launch
Terminate and launch instances at the same time. This allows you to go below your desired capacity by a given percentage and may temporarily reduce availability.
Flexible
Custom behavior
Set custom values for the minimum and maximum amount of available capacity. This gives you greater flexibility in setting how far below and over your desired capacity EC2 Auto Scaling goes when replacing instances.
Additional capacity settings
Capacity Reservation preference Info
Select whether you want Auto Scaling to launch instances into an existing Capacity Reservation or Capacity Reservation resource group.
Default
Auto Scaling uses the Capacity Reservation preference from your launch template.
None
Instances will not be launched into a Capacity Reservation.
Capacity Reservations only
Instances will only be launched into a Capacity Reservation. If capacity isn't available, the instances fail to launch.

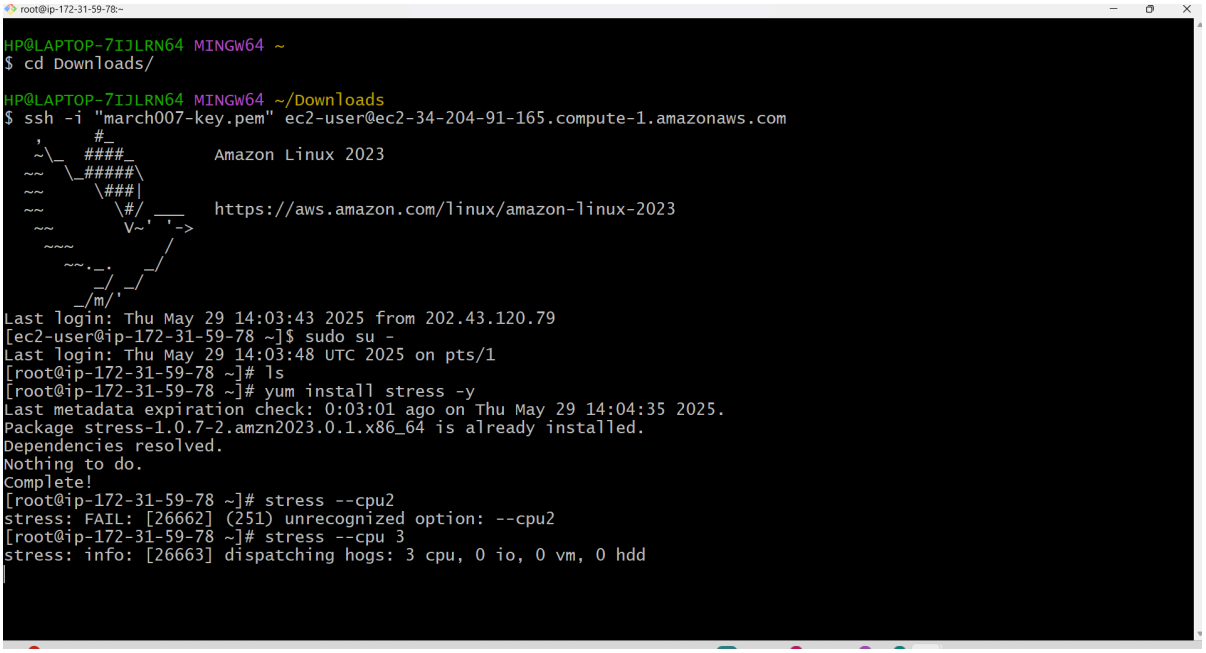

Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Auto Scaling groups > Create Auto Scaling group
Step 1 Choose launch template
Step 2 Choose instance launch options
Step 3 - optional Integrate with other services
Step 4 - optional Configure group size and scaling
Step 5 - optional Add notifications
Step 6 - optional Add tags
Step 7 Review
Add notifications - optional Info
Send notifications to SNS topics whenever Amazon EC2 Auto Scaling launches or terminates the EC2 instances in your Auto Scaling group.
Notification 1
Remove
SNS Topic
Choose an SNS topic to use to send notifications
Application-Topic
Create a topic
Event types
Notify subscribers whenever instances
Launch
Terminate
Fail to launch
Fail to terminate
Add notification
Cancel
Skip to review
Previous
Next

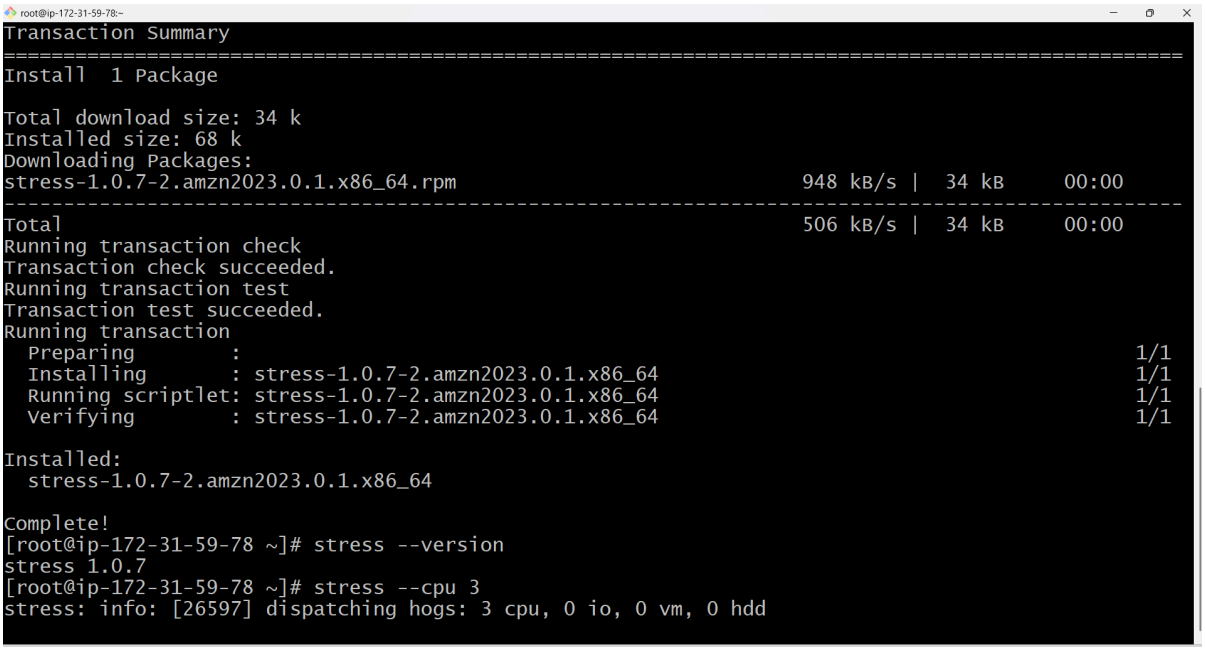

Search [Alt+S]
United States (N. Virginia)
Vaibhao_AWS25
EC2 > Auto Scaling groups
Auto Scaling groups (1) Info
Last updated less than a minute ago
Launch configurations
Launch templates
Actions
Create Auto Scaling group
Search your Auto Scaling groups
Name
Launch template/configuration
Instances
Status
Desired capacity
Min
Max
Availability Zones
Project-Application-SG
Project-Application-template | Version De
1
-
1
1
5
us-east-1a, us-east-1b,...
0 Auto Scaling groups selected
Select an Auto Scaling group



8. Connect via Git Bash

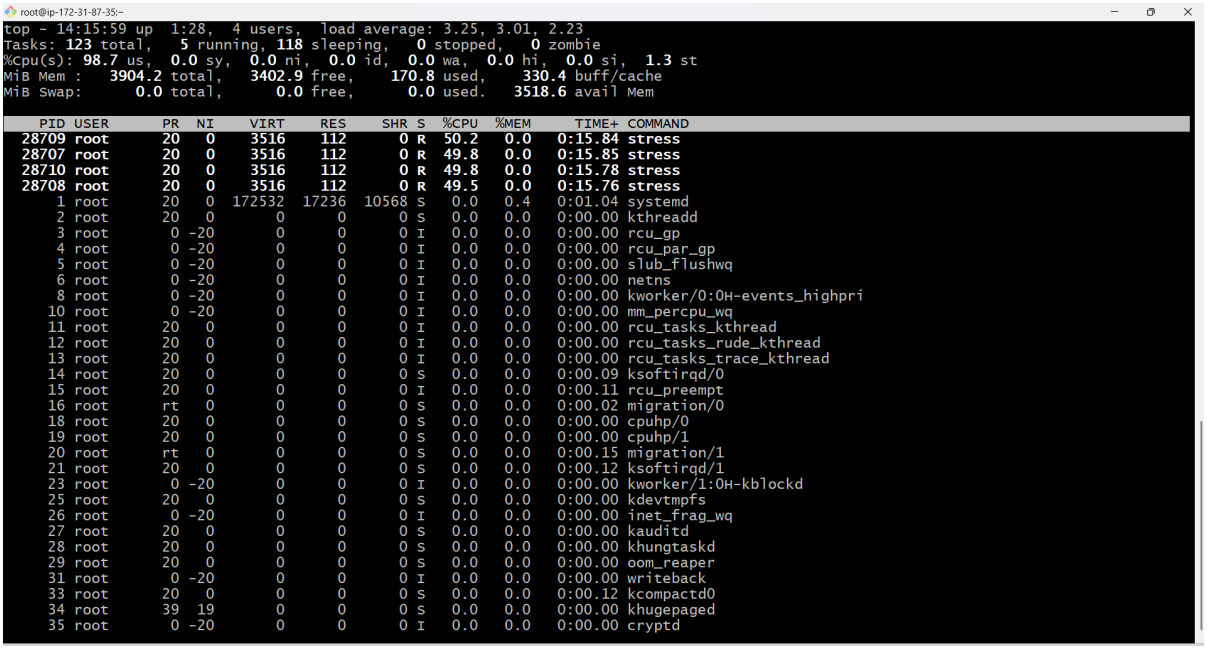
- SSH into EC2 instance using .pem file.
- Installed **“stress”** package to simulate
- **sudo yum install stress -y**
- **stress --version**
- **stress --cpu 3 --timeout 300**





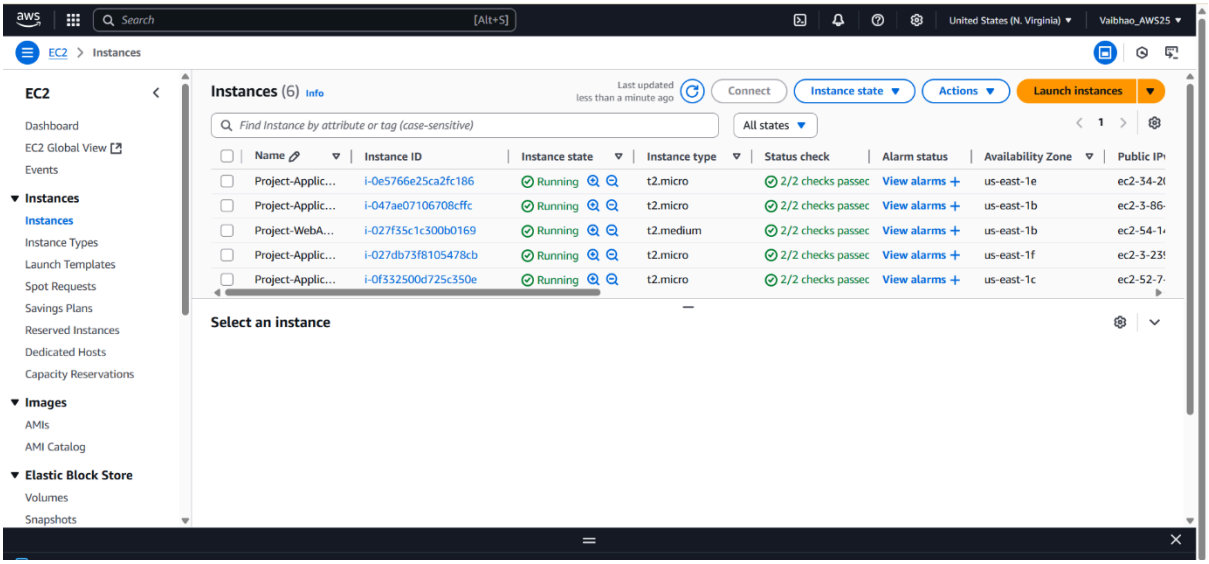
9. Stress Testing & Monitoring

- Ran "**stress**" to **increase CPU load**
- Monitored CPU in real time via "**top**" command
- **Once CPU > 30%,** Auto Scaling launched "**New Instances".**

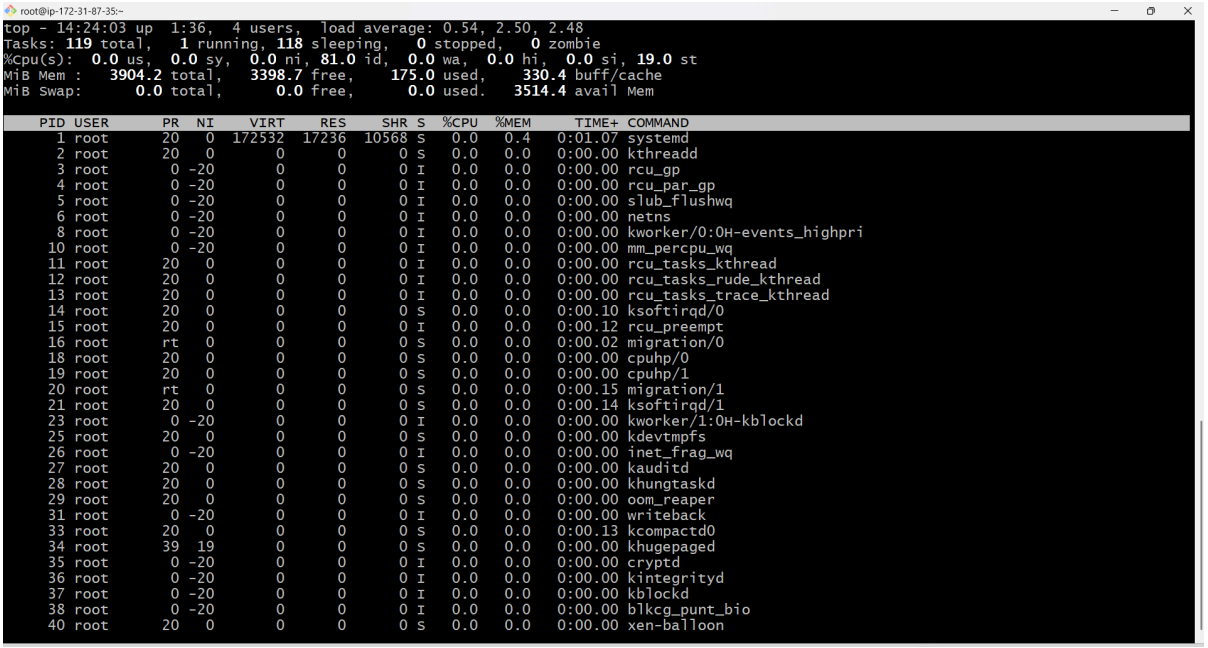


9.1 **Auto Scaling UP**

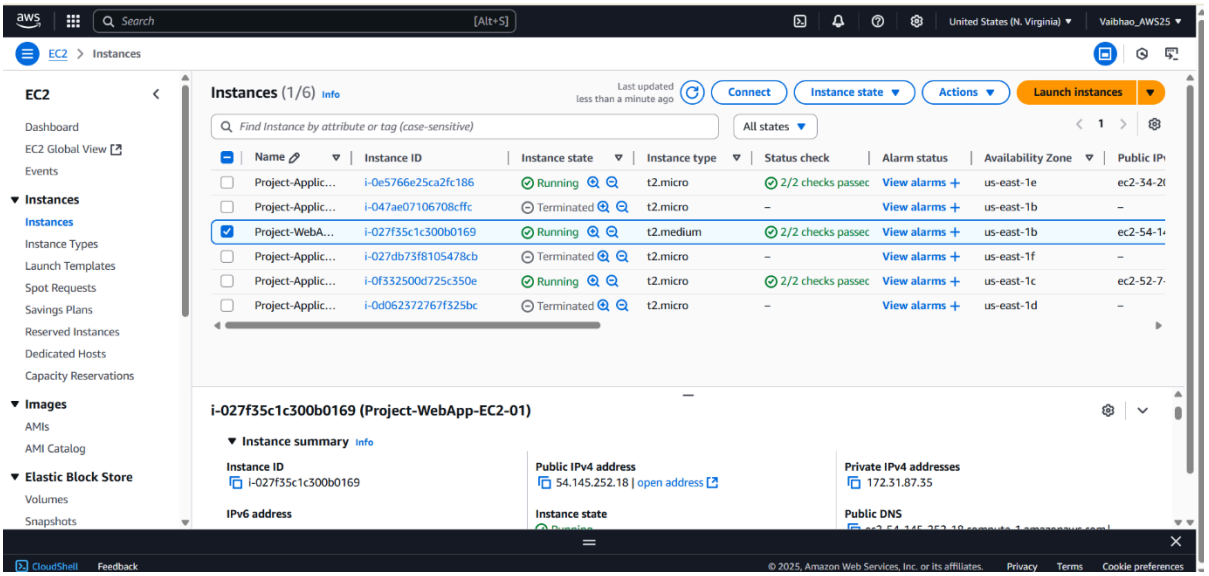
- **As CPU load utilization increases automatically it Launches “New Instances”**



9.2 **Auto Scaling Down**



- **After load reduced, “Extra Instances Terminated”**



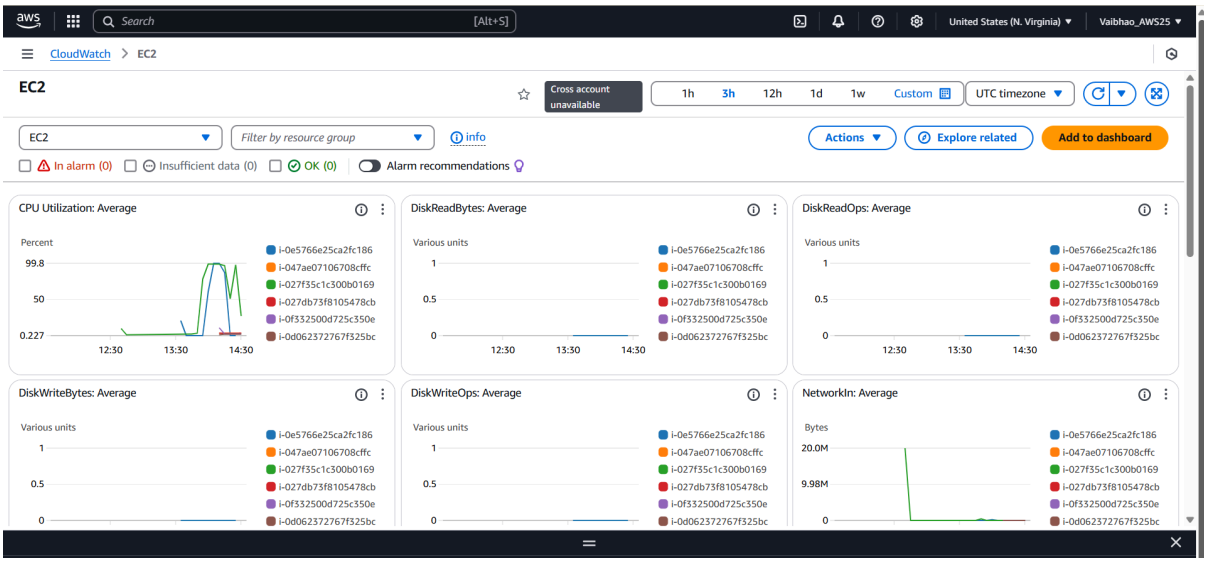
10 Used Default VPC

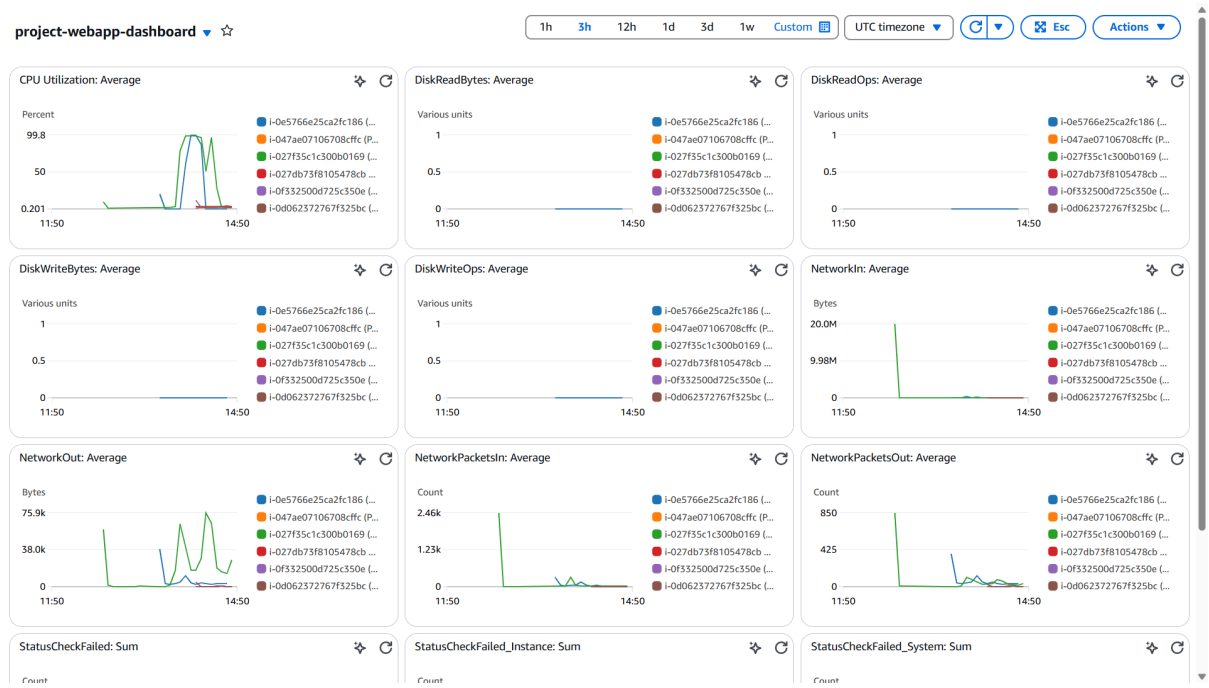
- Avoided custom VPC to keep configuration simple.

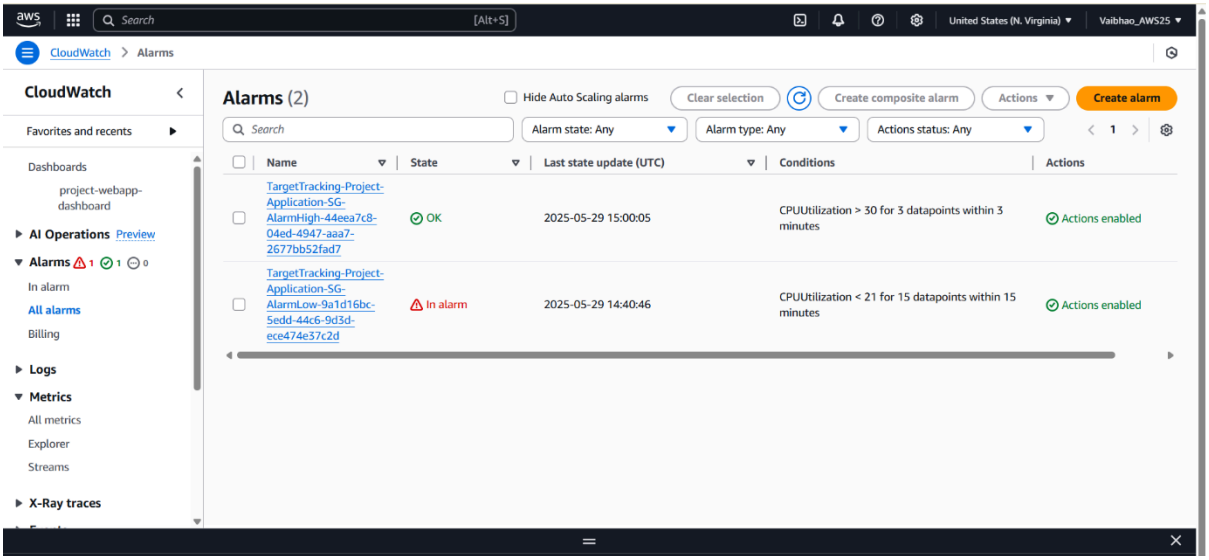
11 Cloud-Watch & Alarm

- Create CloudWatch alarm **for CPU utilization > 30%.**
- Attach alarm to Auto Scaling Policy.
- Link SNS topic for notifications
- Create CloudWatch alarm:
    - Condition: **CPU > 30% for 1 min**
    - Action: Trigger scale-out and notify SNS

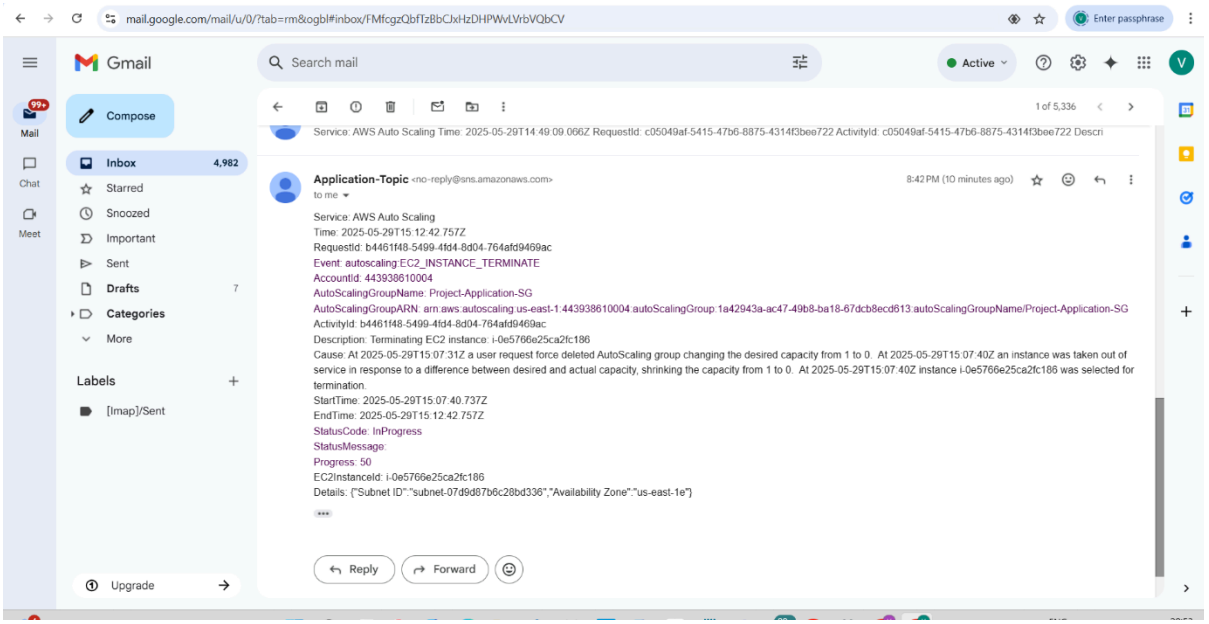
11.1 Cloud-Watch Dashboard







After terminating instance automatically, received email notification



12 S3 Bucket (Optional)

- Used for backup/logging if required

📊 Monitoring and Testing Summary

Test	Expected Result	Status
Load test via stress tool	Auto Scaling triggered	**Passed**
ALB traffic test	Load balanced across instances	**Passed**
EC2 stop and recovery test	Instance replaced automatically	**Passed**
Alarm trigger	SNS email alert received	**Passed**

Scenario	Expected Behaviour	Observed Result
High CPU load	Auto Scaling should launch new EC2	✅ Launched 1–2 more EC2
Load decreases	Extra EC2 should terminate	✅ Terminated after cooldown
Alarm	Email notification via SNS	✅ Email received
App availability	Load balancer should handle traffic	✅ Site always accessible

---

📄 Project Summary

This AWS project efficiently deploys and manages a **web server with scalable, fault-tolerant architecture**. **Auto Scaling** was tested using synthetic load via the **stress tool**, and CloudWatch alarms ensured system health with SNS-based alerts. All services were integrated using the default VPC for ease of setup.

✅ Conclusion

This project successfully demonstrated

- Automated **EC2 provisioning** using AMI and Auto Scaling
- Real-time **monitoring and alerting** with CloudWatch + SNS
- **Load distribution** through ELB
- Infrastructure deployed in **default VPC**
- Efficient **resource usage** through dynamic scaling

AWS provides a flexible platform for deploying web apps that are both **cost-efficient and robust.** This hands-on project builds a solid foundation in **cloud infrastructure management**.

🌟 Key Highlights and Learnings

✔️ Gained hands-on experience with **EC2, ELB, and Auto Scaling**
✔️ Used **CloudWatch alarms** to trigger notifications via **SNS**
✔️ Performed **stress testing** and monitored using Linux tools
✔️ Created and used **AMI** and **Launch Templates**
✔️ Understood **end-to-end deployment flow** on **AWS Cloud**

🔚 End of Report