

# AWS Project Report

## Web Server Deployment and Management using AWS Services

Submitted by:

**Vaibhao Yenchalwarr**

*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:

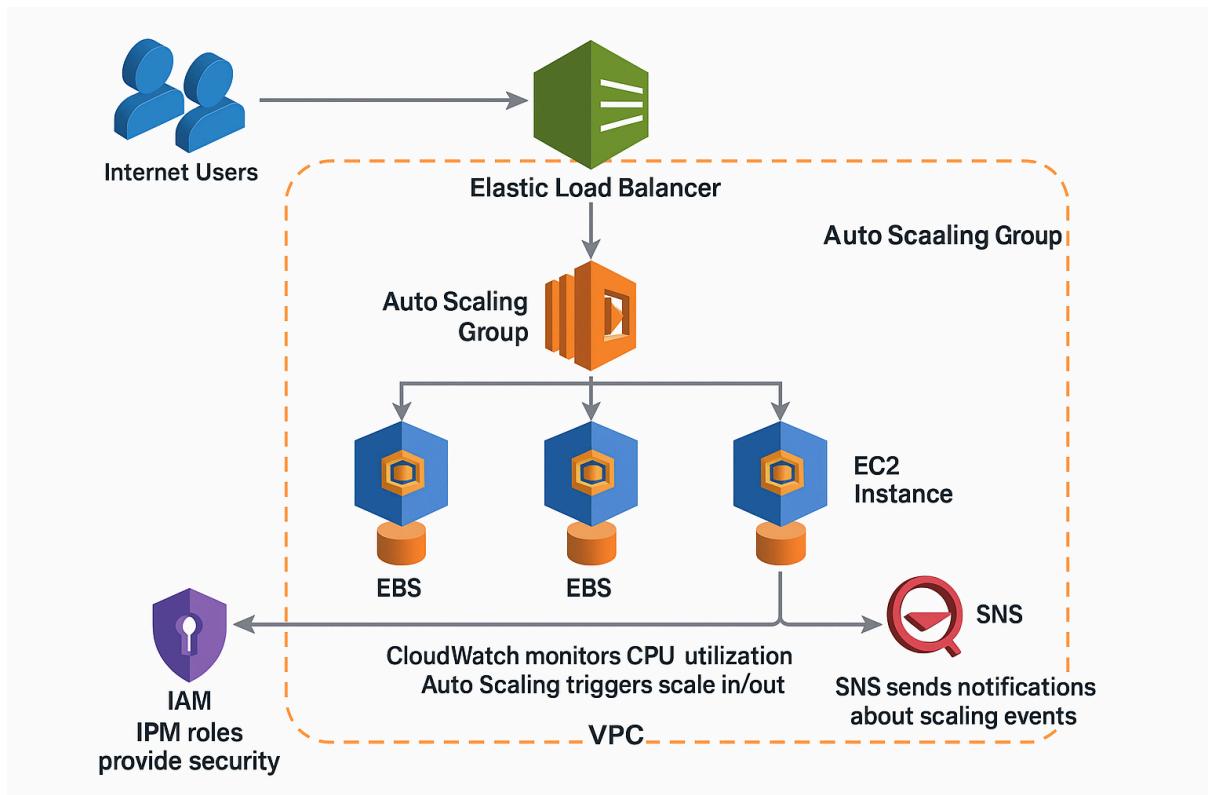
- **Auto-scalable**
  - **Highly available**
  - **Centrally monitored**
  - **Notification-enabled**
-

### 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

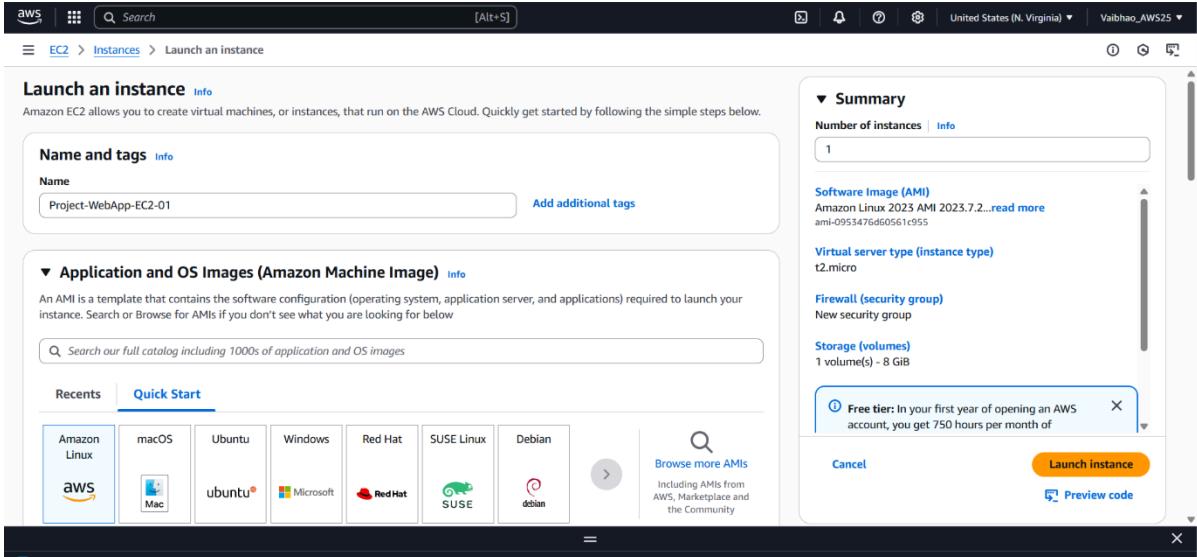


 The Architecture includes EC2 behind a Load Balancer with **Auto Scaling**, monitored via **CloudWatch**, and alerts through **SNS**. A default VPC is used for networking.

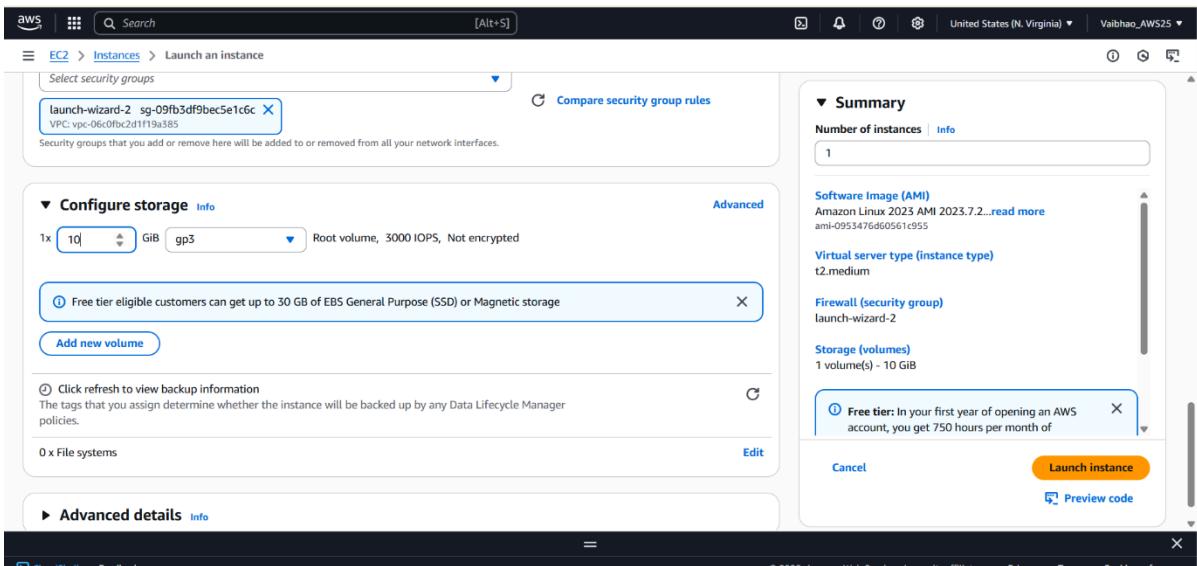
## 5. Implementation Steps

### 1. Create EC2 Instance

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



This screenshot shows the initial steps of the AWS EC2 'Launch an instance' wizard. It includes fields for 'Name and tags' (Project-WebApp-EC2-01), a search bar for 'Application and OS Images (Amazon Machine Image)', and a 'Quick Start' section featuring various AMI icons. On the right, a summary panel shows the selected 'Software Image (AMI)' as Amazon Linux 2023 AMI 2023.7.2, 'Virtual server type (instance type)' as t2.micro, and a note about the free tier.



This screenshot shows the continuation of the 'Launch an instance' wizard, specifically the 'Configure storage' step. It shows a 10 GiB gp3 root volume. A note indicates that free-tier eligible customers can get up to 30 GB of EB5 General Purpose (SSD) or Magnetic storage. Below this, there's a section for 'File systems' and an 'Advanced details' section.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with 'EC2' selected. The main area displays a table of instances with one row selected: 'i-027f35c1c300b0169' (Project-WebApp-EC2-01). The instance is 'Running' on an 't2.medium' type. Below the table, a detailed view for 'i-027f35c1c300b0169 (Project-WebApp-EC2-01)' is shown with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The Details tab is active, showing the instance summary with fields like Public IPv4 address (54.145.252.18), Instance state (Running), and Public DNS (ec2-54-145-252-18.compute-1.amazonaws.com).

## 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.

The screenshot shows the AWS AMIs page. The left sidebar has 'AMIs' selected under 'Images'. The main area shows a table for 'Amazon Machine Images (AMIs) (1/1)' with one entry: 'Application-Image'. The AMI ID is 'ami-0c0b1ed42c7199003'. Below the table, a detailed view for 'AMI ID: ami-0c0b1ed42c7199003' is displayed with tabs for Details, Permissions, Storage, and Tags. The Details tab is active, showing fields such as AMI ID (ami-0c0b1ed42c7199003), Image type (machine), Platform details (Linux/UNIX), Root device type (EBS), AMI name (Application-Image), Owner account ID (443938610004), Architecture (x86\_64), Usage operation (RunInstances), Root device name (/dev/xvda), Status (Pending), Source (443938610004/Application-Image), and Virtualization type (hvm).

### 3. Create Target Group

- Configured ELB to forward requests to targets

The screenshot shows the 'Specify group details' step of creating a target group. It's the first step in a three-step process. The page title is 'Specify group details' and the sub-section is 'Basic configuration'. A note says: 'Your load balancer routes requests to the targets in a target group and performs health checks on the targets.' Under 'Choose a target type', the 'Instances' option is selected, which is highlighted with a blue border. Other options include 'IP addresses' and 'Lambda function'. The 'Instances' section lists benefits such as supporting load balancing to instances within a specific VPC and facilitating the use of Amazon EC2 Auto Scaling.

The screenshot shows the 'Create target group' step. The 'Target group name' field is filled with 'application-target-group'. The 'Protocol : Port' section shows 'HTTP' selected as the protocol and '80' as the port. Below it, the 'IP address type' section has 'IPv4' selected. The 'VPC' section shows a dropdown menu with one item: 'vpc-06c0fb2d1f19a385'. The 'Protocol version' section has 'HTTP1' selected. At the bottom of the form, there is a note: 'Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.'

The screenshot shows the 'Create target group' wizard in the AWS Management Console. The current step is 'Protocol version'. It includes three options: 'HTTP1' (selected), 'HTTP2', and 'gRPC'. Below this is the 'Health checks' section, which contains fields for 'Health check protocol' (set to 'HTTP'), 'Health check path' (set to '/'), and 'Advanced health check settings'. The 'Advanced health check settings' section is expanded, showing fields for 'Health check port' (set to 'Traffic port'), 'Healthy threshold' (set to '5'), 'Unhealthy threshold' (set to '5'), 'Timeout' (set to '10 seconds'), and 'Interval' (set to '10 seconds').

This screenshot shows the 'Register targets' step in the AWS EC2 Target Groups creation wizard. It displays a list of available instances and allows selecting ports for traffic routing.

**Available instances (1/1)**

Instance ID	Name	State	Security groups	Zone
i-027f35c1c300b0169	Project-WebApp-EC2-01	Running	launch-wizard-2	us-east-1b

**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**

This screenshot shows the configuration step of the AWS EC2 Target Groups creation wizard. It includes settings for the target group's interval, success codes, attributes, and optional tags.

**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**

**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)**

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

application-target-group

**Details**

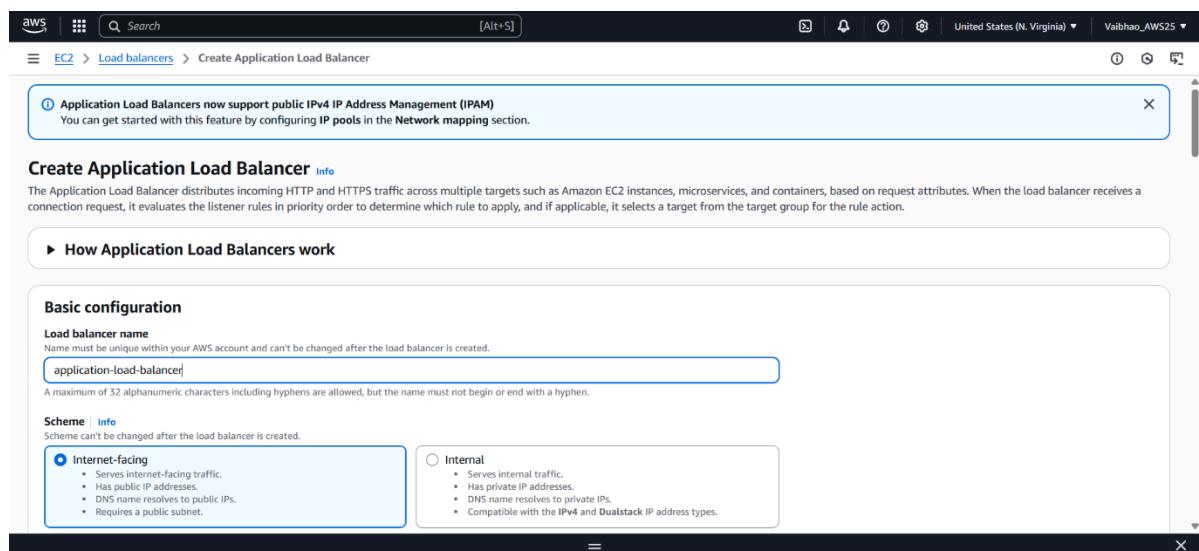
Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC <a href="#">vpc-06c0fb2d1f19a385</a>
IP address type IPv4	Load balancer <a href="#">None associated</a>		
1 Total targets	0 Healthy	0 Unhealthy	1 Unused
0 Anomalous		0 Initial	0 Draining

**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.



**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-06c0fb2d1f19a385](#)  
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.

**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-00edf17c1bb7d55db](#)  
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

**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**

Protocol	Port	Default action	Info
HTTP	80 1-65535	Forward to	application-target-group Target type: Instance, IPv4
		HTTP	

[Create target group](#) [Remove](#)

**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 =

**Subnets**

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-092da8a602a05751b  
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-09fb5df9bec5e1e6c VPC: vpc-06c0fbcd1f19a385

**Load balancers (1)**

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	State	VPC ID	Availability Zones	Type	Date create
application-load-balancer	application-load-balancer-1...	Active	vpc-06c0fbcd1f19a385	6 Availability Zones	application	May 29, 2023

[Actions](#) [Create load balancer](#)

## 5. Configured SNS

- Created topic

The screenshot shows the 'Create topic' wizard in the AWS SNS console. The 'Type' section is selected, showing two options: 'FIFO (first-in, first-out)' and 'Standard'. The 'Standard' option is chosen, highlighted with a blue border. Below the type selection, there are fields for 'Name' (set to 'Application-Topic') and 'Display name - optional' (also set to 'Application-Topic'). A note indicates that a display name is used for SMS subscriptions. At the bottom, there is a section titled 'Encryption - optional' with a note about server-side encryption.

The screenshot shows the 'Topics' page in the AWS SNS console. On the left, a navigation menu includes 'Topics' under 'Amazon SNS'. The main area displays a table with one row for the 'Application-Topic'. The table columns are 'Name' (Application-Topic), 'Type' (Standard), and 'ARN' (arn:aws:sns:us-east-1:443938610004:Application-T...). Action buttons for 'Edit', 'Delete', and 'Publish message' are visible at the top right, along with a 'Create topic' button.

- Subscribed with email for alerts

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.

Amazon SNS > Subscriptions

### Subscriptions (1)

ID	Endpoint	Status	Protocol	Topic
Pending confirmation	vaibhaoee22@gmail.com	Pending confirmation	EMAIL	Application-Topic

Gmail

Inbox 4,983

AWS Notification - Subscription Confirmation

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](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#).

## 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

**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-t2.2  
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.

**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 bandwidth to the internet.

**Create launch template**

Screenshot of the AWS EC2 'Create launch template' wizard.

**Instance type:** t2.micro (Free tier eligible)

- All generations
- Compare instance types

**Key pair (login):** march007-key

**Network settings:** Subnet: Don't include in launch template

**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

**Create launch template**

Screenshot of the AWS EC2 'Launch Templates' page.

**Launch Templates (1):**

Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By
lt-0a2a73a1ff5af831	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

The screenshot shows the 'Choose launch template' step of the 'Create Auto Scaling group' wizard. On the left, a sidebar lists steps from 1 to 7. Step 1 is 'Choose launch template' (selected), Step 2 is 'Choose instance launch options', Step 3 is 'Optional: Integrate with other services', Step 4 is 'Optional: Configure group size and scaling', Step 5 is 'Optional: Add notifications', Step 6 is 'Optional: Add tags', and Step 7 is 'Review'. The main panel is titled 'Choose launch template' and contains a 'Name' section where 'Auto Scaling group name' is set to 'Project-Application-SG'. A note states: 'For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.' Below this is a 'Launch template' section with a dropdown set to 'Project-Application-template' and a 'Version' dropdown set to 'Default (1)'. There are also 'Description', 'Launch template', and 'Instance type' sections.

The screenshot shows the 'Choose instance launch options' step of the 'Create Auto Scaling group' wizard. The sidebar shows steps 1 through 7. Step 2 is 'Choose instance launch options' (selected). The main panel is titled 'Choose instance launch options' and contains an 'Instance type requirements' section with a 'Launch template' dropdown set to 'Project-Application-template', a 'Version' dropdown set to 'Default', and a 'Description' section with 'template-2.2'. It also includes an 'Override launch template' button. Below this is a 'Network' section with a 'VPC' dropdown set to 'vpc-06c0fbcd21f19a385' (172.31.0.0/16, Default). A 'Create a VPC' link is at the bottom.

Suitable for getting started quickly.

**VPC**  
Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-06c0fb2d1f19a385  
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 X  
172.31.0.0/20 Default

us-east-1b | subnet-0d7bed53462258b22 X  
172.31.80.0/20 Default

us-east-1c | subnet-0398bcc5e75991659 X  
172.31.16.0/20 Default

us-east-1d | subnet-092da8d602a05751b X  
172.31.32.0/20 Default

us-east-1e | subnet-07d9d87b6c28bd356 X  
172.31.48.0/20 Default

us-east-1f | subnet-0e670b6b96878096a X  
172.31.64.0/20 Default

Create a subnet

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

**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**

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

Step 2

- Choose instance launch options
- Step 3 - optional
- Integrate with other services
- Configure group size and scaling**
- Step 4 - optional
- Add notifications
- Step 5 - optional
- Add tags
- Step 6 - optional
- Review

**Group size** Info  
Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

**Desired capacity type**  
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

**Units (number of instances)**

**Desired capacity**  
Specify your group size.  
1

**Scaling** Info  
You can resize your Auto Scaling group manually or automatically to meet changes in demand.

**Scaling limits**  
Set limits on how much your desired capacity can be increased or decreased.

<b>Min desired capacity</b> 1 Equal or less than desired capacity	<b>Max desired capacity</b> 5 Equal or greater than desired capacity
-------------------------------------------------------------------------	----------------------------------------------------------------------------

**Automatic scaling - optional**  
**Choose whether to use a target tracking policy** | Info  
You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

- Configured policy to scale out if CPU > 30%

**Automatic scaling - optional**  
**Choose whether to use a target tracking policy** | Info  
You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

**No scaling policies**  
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

**Target tracking scaling policy**  
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

**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**

<b>Mixed behavior</b> <input checked="" type="radio"/> <b>No policy</b> For rebalancing events, new instances will launch before terminating others. For all other events, instances terminate and launch at the same time.	<b>Prioritize availability</b> <input type="radio"/> <b>Launch before terminating</b> Launch new instances and wait for them to ready before terminating others. This allows you to go above your desired capacity by a given percentage and may temporarily increase costs.	<b>Control costs</b> <input type="radio"/> <b>Terminate and launch</b> 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.	<b>Flexible</b> <input type="radio"/> <b>Custom behavior</b> Set custom values for the minimum and maximum 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**

**Summary**

**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** (selected) For rebalancing events, new instances will launch before terminating others. For all other events, instances terminate and launch at the same time.
- Prioritize availability**
- Control costs**
- Flexible**
- Terminate and launch**
- Custom behavior**

**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** (selected) 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.

**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**

**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**

**Auto Scaling groups (1)** Info

Last updated less than a minute ago

**Launch configurations** **Launch templates** **Actions** **Create Auto Scaling group**

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

The screenshot shows the AWS EC2 Instances page. The left sidebar has 'EC2' selected under 'Instances'. The main area displays 'Instances (2) Info' with a table of two running instances. The first instance, 'Project-Applic...', has an ID of i-0e5766e25ca2fc186, is a t2.micro type, and has 2/2 checks passed. The second instance, 'Project-WebA...', has an ID of i-027f35c1c300b0169, is a t2.medium type, and also has 2/2 checks passed. Both instances are in the 'us-east-1e' availability zone with public IP addresses ec2-34-2(1) and ec2-54-1.

## 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**

```
root@ip-172-31-59-78:~#
HP@LAPTOP-7IJLRN64 MINGW64 ~
$ cd Downloads/
HP@LAPTOP-7IJLRN64 MINGW64 ~/Downloads
$ ssh -i "march007-key.pem" ec2-user@ec2-34-204-91-165.compute-1.amazonaws.com
,#
~\_\#\#\#_
~\_\#\#\#\#
~~`#\#\#
~~`#/`-->
https://aws.amazon.com/linux/amazon-linux-2023
~~`..`/`-
~~`/`-
~/`-
Last login: Thu May 29 14:03:43 2025 from 202.43.120.79
[ec2-user@ip-172-31-59-78 ~]$ sudo su -
Last login: Thu May 29 14:03:48 UTC 2025 on pts/1
[root@ip-172-31-59-78 ~]# ls
[root@ip-172-31-59-78 ~]# yum install stress -y
Last metadata expiration check: 0:03:01 ago on Thu May 29 14:04:35 2025.
Package stress-1.0.7-2.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-59-78 ~]# stress --cpu2
stress: FAIL: [26662] (251) unrecognized option: --cpu2
[root@ip-172-31-59-78 ~]# stress --cpu 3
stress: info: [26663] dispatching hogs: 3 cpu, 0 io, 0 vm, 0 hdd
```

```

root@ip-172-31-59-78:~#
Transaction Summary
=====
Install 1 Package

Total download size: 34 k
Installed size: 68 k
Downloading Packages:
stress-1.0.7-2.amzn2023.0.1.x86_64.rpm
Total                                         948 kB/s | 34 kB   00:00
-----[Total]                                     506 kB/s | 34 kB   00:00

Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : 1/1
Installing : stress-1.0.7-2.amzn2023.0.1.x86_64 1/1
Running scriptlet: stress-1.0.7-2.amzn2023.0.1.x86_64 1/1
Verifying   : stress-1.0.7-2.amzn2023.0.1.x86_64 1/1

Installed:
  stress-1.0.7-2.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-59-78 ~]# stress --version
stress 1.0.7
[root@ip-172-31-59-78 ~]# stress --cpu 3
stress: info: [26597] dispatching hogs: 3 cpu, 0 io, 0 vm, 0 hdd

```

## 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**”.

```

root@ip-172-31-87-35:~
top - 14:15:59 up 1:28, 4 users, load average: 3.25, 3.01, 2.23
Tasks: 123 total,  5 running, 118 sleeping,  0 stopped,  0 zombie
%Cpu(s): 98.7 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  1.3 st
Mem:  3904.2 total, 3402.9 free, 170.8 used,   330.4 buff/cache
Swap:     0.0 total,      0.0 free,      0.0 used, 3518.6 avail Mem

 PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM TIME+ COMMAND
28709 root      20   0  3516   112    0 R 50.2  0.0 0:15.84 stress
28707 root      20   0  3516   112    0 R 49.8  0.0 0:15.85 stress
28710 root      20   0  3516   112    0 R 49.8  0.0 0:15.78 stress
28708 root      20   0  3516   112    0 R 49.5  0.0 0:15.76 stress
  1 root      20   0 172532 17236 10568 S  0.0  0.4 0:01.04 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 slob_flushhwq
  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 0:00.00 rcu_tasks_kthread
 12 root      20   0      0      0    0 I  0.0  0.0 0:00.00 rcu_tasks_rude_kthread
 13 root      20   0      0      0    0 I  0.0  0.0 0:00.00 rcu_tasks_trace_kthread
 14 root      20   0      0      0    0 S  0.0  0.0 0:00.09 ksoftirqd/0
 15 root      20   0      0      0    0 I  0.0  0.0 0:00.11 rcu_preempt
 16 root      rt  0      0      0    0 S  0.0  0.0 0:00.02 migration/0
 18 root      20   0      0      0    0 S  0.0  0.0 0:00.00 cpuhp/0
 19 root      20   0      0      0    0 S  0.0  0.0 0:00.00 cpuhp/1
 20 root      rt  0      0      0    0 S  0.0  0.0 0:00.15 migration/1
 21 root      20   0      0      0    0 S  0.0  0.0 0:00.12 ksoftirqd/1
 23 root      0 -20      0      0    0 I  0.0  0.0 0:00.00 kworker/1:0H-kblockd
 25 root      20   0      0      0    0 S  0.0  0.0 0:00.00 kdevtmpfs
 26 root      0 -20      0      0    0 I  0.0  0.0 0:00.00 inet_frag_wq
 27 root      20   0      0      0    0 S  0.0  0.0 0:00.00 kauditfd
 28 root      20   0      0      0    0 S  0.0  0.0 0:00.00 khungtaskd
 29 root      20   0      0      0    0 S  0.0  0.0 0:00.00 oom_reaper
 31 root      0 -20      0      0    0 I  0.0  0.0 0:00.00 writeback
 33 root      20   0      0      0    0 S  0.0  0.0 0:00.12 kcompactd0
 34 root      39  19      0      0    0 S  0.0  0.0 0:00.00 khugepaged
 35 root      0 -20      0      0    0 I  0.0  0.0 0:00.00 cryptd

```

## 9.1 Auto Scaling UP

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

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed, and the main area displays a table titled "Instances (6) Info". The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. All instances are listed as "Running" t2.micro type, located in us-east-1, with 2/2 checks passed and active alarms. A "Launch instances" button is visible at the top right of the table.

## 9.2 Auto Scaling Down

The screenshot shows a terminal window displaying system monitoring output. The top part of the screen shows the command "top - 14:24:03 up 1:36, 4 users, load average: 0.54, 2.50, 2.48" and various system statistics like CPU usage and memory. Below this is a detailed table of processes (PID, USER, PR, NI, VIRT, RES, SHR, S, %CPU, %MEM, TIME+, COMMAND) with over 40 entries, mostly kernel threads and system daemons.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	172532	17236	10568	S	0.0	0.4	0:01.07	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/0:0-H-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	0:00.00	rcu_tasks_kthread
12	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
14	root	20	0	0	0	0	S	0.0	0.0	0:00.10	ksoftirqd/0
15	root	20	0	0	0	0	I	0.0	0.0	0:00.12	rcu_preempt
16	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration/0
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
20	root	rt	0	0	0	0	S	0.0	0.0	0:00.15	migration/1
21	root	20	0	0	0	0	S	0.0	0.0	0:00.14	ksoftirqd/1
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0-H-kblockd
25	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	inet_frag_wq
27	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
28	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
29	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
31	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
33	root	20	0	0	0	0	S	0.0	0.0	0:00.13	kcompactd0
34	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
35	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	cryptd
36	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
37	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
38	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	blkcg_punt_bio
40	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xen-balloon

- After load reduced, “Extra Instances Terminated”

The screenshot shows the AWS EC2 Instances page. The left sidebar includes links for Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes, Snapshots). The main content area displays a table titled "Instances (1/6) info" with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. There are six rows, with the first five being terminated and one being running. Below the table, a specific instance, "i-027f35c1c300b0169 (Project-WebApp-EC2-01)", is selected and expanded. The expanded view shows "Instance summary" with fields for Instance ID (i-027f35c1c300b0169), Public IPv4 address (54.145.252.18), Private IPv4 addresses (172.31.87.35), and Public DNS (ip-172-31-87-35.compute-1.amazonaws.com). The status is listed as Pending.

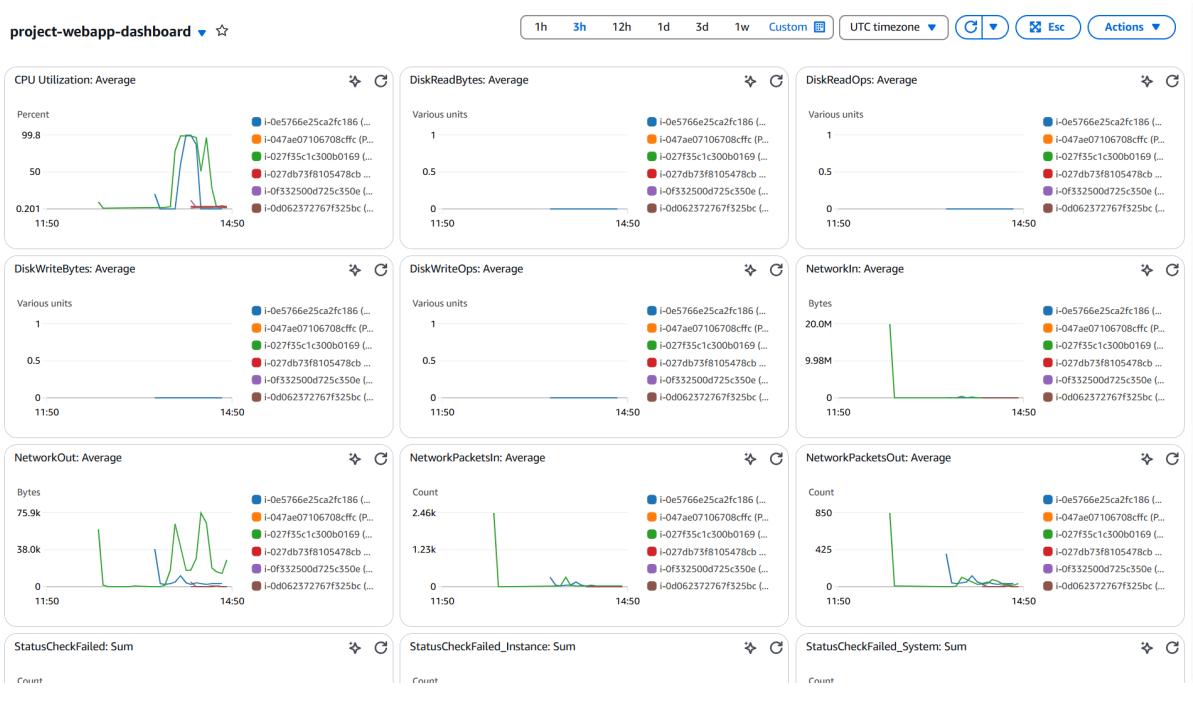
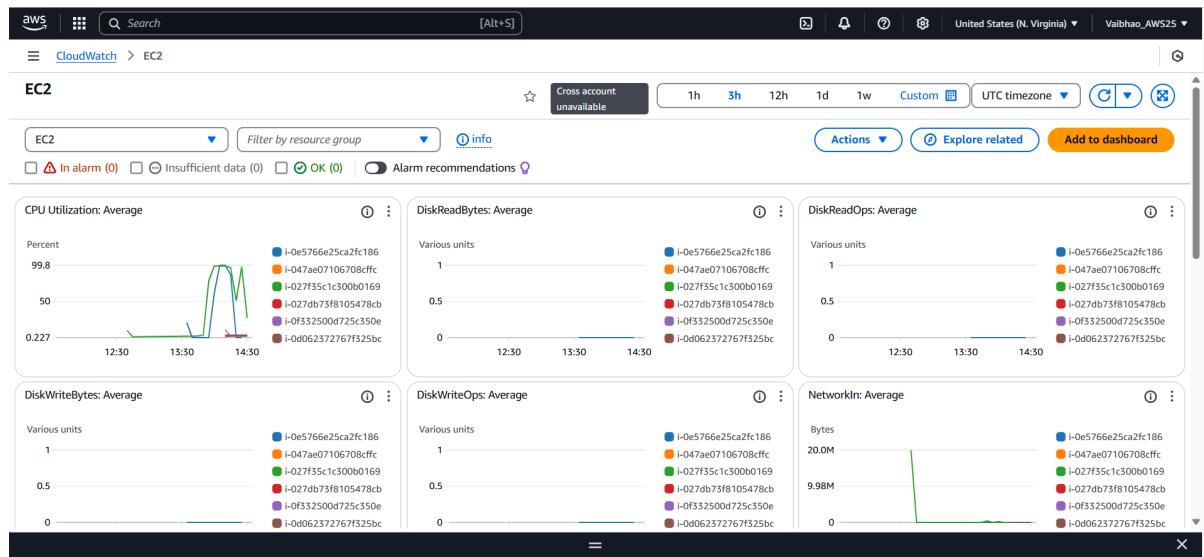
## 10 Used Default VPC

- 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



The screenshot shows the AWS CloudWatch Alarms interface. On the left, there's a navigation sidebar with links like Dashboards, AI Operations, Alarms, Logs, Metrics, and X-Ray traces. The main area is titled 'Alarms (2)' and lists two alarms:

- TargetTracking-Project-Application-SG-AlarmHigh-44ea78-04ed-4947-aaa7-2677bb52fad7**: State OK, Last state update: 2025-05-29 15:00:05. Condition: CPUUtilization > 30 for 3 datapoints within 3 minutes. Actions enabled.
- TargetTracking-Project-Application-SG-AlarmLow-9a1d16bc-5edd-44c6-9d3dece474e57c2d**: State In alarm, Last state update: 2025-05-29 14:40:46. Condition: CPUUtilization < 21 for 15 datapoints within 15 minutes. Actions enabled.

## After terminating instance automatically, received email notification

The screenshot shows a Gmail inbox with 4,982 messages. An email from 'Application-Topic <no-reply@sns.amazonaws.com>' is selected. The subject is 'Service: AWS Auto Scaling Time: 2025-05-29T14:49:09.066Z RequestId: c05049af-5415-47b6-8875-4314f3bee722 ActivityId: c05049af-5415-47b6-8875-4314f3bee722 Description:'. The message body contains detailed information about the termination of an EC2 instance, including the activity ID, instance ID, and cause of termination.

## 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