

Design and Implementation of an Auto-Scalable Web Application on AWS

Using Load Balancer, CloudWatch and SNS

Introduction

This project focuses on designing and implementing an auto-scalable web application infrastructure using Amazon Web Services. The system dynamically adjusts computing resources based on real-time demand, ensuring high availability, performance efficiency, and cost optimization. By integrating load balancing, monitoring, and notification services, the architecture demonstrates a practical cloud solution capable of handling variable traffic loads without manual intervention.

Problem Statement

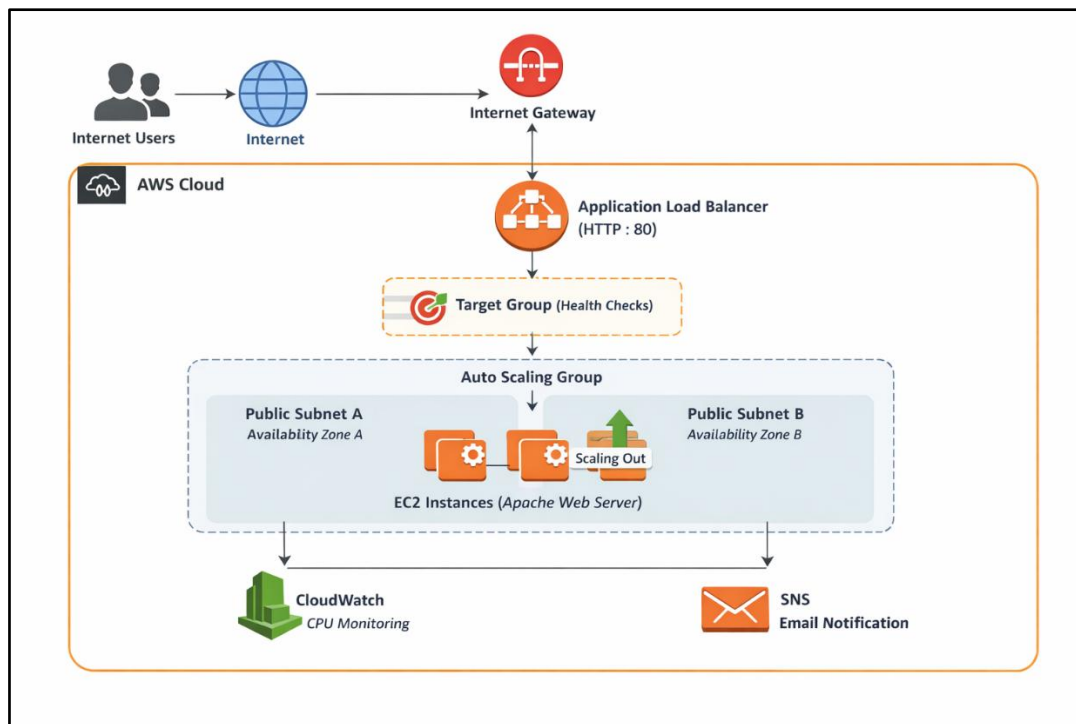
Traditional web application infrastructures struggle to efficiently manage sudden increases or decreases in user traffic. Over-provisioning leads to unnecessary costs, while under-provisioning causes performance degradation and downtime. The problem addressed in this project is to design a cloud-based infrastructure that can automatically scale resources in response to workload changes while maintaining availability, reliability, and minimal operational overhead.

Objective

- Deploy a web application on EC2
- Distribute traffic using an Application Load Balancer
- Automatically scale EC2 instances based on CPU usage
- Monitor system performance using CloudWatch
- Send real-time notifications through SNS

Architecture Overview

The architecture consists of an internet-facing Application Load Balancer that distributes incoming HTTP requests across multiple EC2 instances. These instances are managed by an Auto Scaling Group, which launches or terminates instances based on CPU utilization. CloudWatch continuously monitors performance metrics and triggers scaling actions, while SNS sends notifications during significant scaling events. This design ensures fault tolerance, scalability, and consistent application availability.



AWS Services Used

The following Amazon Web Services were used in the implementation of this project:

1. Amazon EC2 (Elastic Compute Cloud)

Amazon EC2 provides scalable virtual servers to host the web application. EC2 instances run the Apache web server and dynamically scale based on workload requirements. These instances are launched automatically using a predefined launch template.

The screenshot shows the AWS Management Console interface for an EC2 instance. The top navigation bar includes buttons for 'Connect', 'Instance state', 'Actions', and 'Launch instances'. Below the navigation bar, there is a table listing instances. The instance 'web-ec2' with ID 'i-06fd93af501d747ae' is shown in a 'Running' state. The instance details are displayed below the table, including the Instance ID, Public IPv4 address (3.234.223.234), Private IPv4 addresses (10.0.1.28), Instance state (Running), Public DNS (ec2-3-234-223-234.compute-1.amazonaws.com), and Private IP DNS name (ip-10-0-1-28.ec2.internal).

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
web-ec2	i-06fd93af501d747ae	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	ec2-3-234-223-234

i-06fd93af501d747ae (web-ec2)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary

- Instance ID: i-06fd93af501d747ae
- Public IPv4 address: 3.234.223.234 | [open address](#)
- Private IPv4 addresses: 10.0.1.28
- Instance state: Running
- Public DNS: ec2-3-234-223-234.compute-1.amazonaws.com | [open address](#)
- Private IP DNS name (IPv4 only): ip-10-0-1-28.ec2.internal
- IP name: ip-10-0-1-28.ec2.internal

2. Application Load Balancer (ALB)

The Application Load Balancer distributes incoming HTTP traffic across multiple EC2 instances. It improves application availability and fault tolerance by ensuring that traffic is routed only to healthy instances.

web-app-load-balanecer

Actions

▼ Details

Load balancer type Application	Status Active	VPC vpc-0ea52c1769a34f6e8	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-0204f86f5f3553760 us-east-1a (use1-az1) subnet-004953d4ddb38f9f us-east-1b (use1-az2)	Date created January 8, 2026, 21:41 (UTC+05:30)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:098167103976:loadbalancer/app/web-app-load-balanecer/af1fbe3ca09658b9		DNS name Info web-app-load-balanecer-221679503.us-east-1.elb.amazonaws.com (A Record)	

3. Auto Scaling Group (ASG)

Auto Scaling Groups automatically manage the number of EC2 instances. Based on CPU utilization metrics, the ASG launches new instances during high demand and terminates instances when demand decreases, ensuring optimal resource utilization.

web-autoscaling

Edit

web-autoscaling Capacity overview

[arn:aws:autoscaling:us-east-1:098167103976:autoScalingGroup:ff7407a7-89ab-4be8-af31-98f1e49d4920:autoScalingGroupName/web-autoscaling](#)

Desired capacity 1	Scaling limits 1 - 3	Desired capacity type Units (number of instances)	Status -
-----------------------	-------------------------	--	-------------

Date created
Thu Jan 08 2026 21:44:40 GMT+0530 (India Standard Time)

4. Amazon CloudWatch

Amazon CloudWatch monitors system performance metrics such as CPU utilization. CloudWatch alarms are configured to trigger scaling actions when predefined thresholds are crossed.



5. Amazon SNS (Simple Notification Service)

Amazon SNS is used to send real-time email notifications when CloudWatch alarms change state. This helps administrators stay informed about scaling events and system performance.



adithi-sns

Edit

Delete

Publish message

Details

Name

adithi-sns

ARN

arn:aws:sns:us-east-1:098167103976:adithi-sns

Display name

my-sns

Type

Standard

Topic owner

098167103976

<

Subscriptions

Access policy

Data protection policy

Delivery policy (HTTP/S)

Delivery status logging

Encryption

Tags

>

Subscriptions (1)

Edit

Delete

Request confirmation

Confirm subscription

Create subscription

Q Search

< 1 >

⚙

ID	Endpoint	Status	Protocol
d2ea32d0-6010-4d70-ae31-3881c...	keerthi.creative05@gmail.com	Confirmed	EMAIL

6. Amazon VPC (Virtual Private Cloud)

Amazon VPC provides the networking foundation for the project. It enables the creation of subnets, route tables, and internet gateways, allowing secure communication between AWS resources and external users.

vpc-0ea52c1769a34f6e8 / web-vpc

Actions

Details

VPC ID

vpc-0ea52c1769a34f6e8

DNS resolution

Enabled

Main network ACL

acl-0417009841d983b0e

IPv6 CIDR (Network border group)

-

Encryption control ID

-

State

Available

Tenancy

default

Default VPC

No

Network Address Usage metrics

Disabled

Encryption control mode

-

Block Public Access

Off

DHCP option set

dopt-08fbaa4d6070f6dab

IPv4 CIDR

10.0.0.0/16

Route 53 Resolver DNS Firewall rule groups

-

DNS hostnames

Enabled

Main route table

rtb-00ecca8ab07bc6a39

IPv6 pool

-

Owner ID

098167103976

Resource map

CIDRs

Flow logs

Tags

Integrations

Resource map

Show all details

VPC

Your AWS virtual network

web-vpc

Subnets (2)

Subnets within this VPC

us-east-1a

web-publicsubnet-01

us-east-1b

web-publicsubnet-02

Route tables (2)

Route network traffic to resources

rtb-00ecca8ab07bc6a39

web-rt-public

Network Connections (1)

Connections to other networks

web-igwv

7. Internet Gateway

The Internet Gateway enables internet access for resources deployed within the VPC, allowing users to access the application through the Application Load Balancer.

Internet gateways (1/2) Info

Find internet gateways by attribute or tag

Actions

Create internet gateway

Name

Internet gateway ID

State

VPC ID

Owner

-

[igw-01f1d55cc47c26c1a](#)

Attached

[vpc-04f3413b36d6d6481](#)

098167103976

web-igwy

[igw-04a94331ad68b1dff](#)

Attached

[vpc-0ea52c1769a34f6e8](#) | web-vpc

098167103976

igw-04a94331ad68b1dff / web-igwy

Details

Tags

Details

Internet gateway ID

[igw-04a94331ad68b1dff](#)

State

Attached

VPC ID

[vpc-0ea52c1769a34f6e8](#) | web-vpc

Owner

[098167103976](#)

Route tables (3) Info

Last updated 4 minutes ago

Actions

Create route table

Find route tables by attribute or tag

Name

Route table ID

Explicit subnet associ...

Edge associations

Main

VPC

web-rt-public

[rtb-06647fd6946a0f121](#)

2 subnets

-

No

[vpc-0ea52c1769a34f6e8](#) | web-

-

[rtb-00ecca8ab07bc6a39](#)

-

-

Yes

[vpc-0ea52c1769a34f6e8](#) | web-

-

[rtb-0b70c825d5e1c8923](#)

-

-

Yes

[vpc-04f3413b36d6d6481](#)

8. AWS Security Groups

Security Groups act as virtual firewalls that control inbound and outbound traffic to EC2 instances and the load balancer. They ensure that only authorized traffic is allowed.

Security Groups (1/18) Info

Find security groups by attribute or tag

Actions

Export security groups to CSV

Create security group

Name

Security group ID

Security group name

VPC ID

Description

-

[sg-0e1351e5c48bbe21f](#)

web-securitygrp

[vpc-0ea52c1769a34f6e8](#)

Web + SSH access

-

[sg-0eeb56c31691d59f6](#)

launch-wizard-9

[vpc-04f3413b36d6d6481](#)

launch-wizard-9 c

-

[sg-0f915c027522d6472](#)

launch-wizard-8

[vpc-04f3413b36d6d6481](#)

launch-wizard-8 c

-

[sg-0b310adda0829f02c](#)

launch-wizard-7

[vpc-04f3413b36d6d6481](#)

launch-wizard-7 c

sg-0e1351e5c48bbe21f - web-securitygrp

Details

Inbound rules

Outbound rules

Sharing

VPC associations

Tags

Details

Security group name

[web-securitygrp](#)

Security group ID

[sg-0e1351e5c48bbe21f](#)

Description

[Web + SSH access](#)

VPC ID

[vpc-0ea52c1769a34f6e8](#)

Owner

[098167103976](#)

Inbound rules count

2 Permission entries

Outbound rules count

1 Permission entry

9. Subnets

Subnets are used to logically divide the VPC network into smaller segments. In this project, multiple public subnets are created across different Availability Zones to ensure high availability and fault tolerance. These subnets host the Application Load Balancer and EC2 instances managed by the Auto Scaling Group, allowing traffic to be distributed across zones in case of failures.

Subnets (2/8) [Info](#)

Last updated 2 minutes ago [Actions](#) [Create subnet](#)

< 1 >

<input type="checkbox"/>	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
<input checked="" type="checkbox"/>	web-publicsubnet-02	subnet-004953d4ddb38f9f	Available	vpc-0ea52c1769a34f6e8 web-...	Off	10.0.2.0/24
<input checked="" type="checkbox"/>	web-publicsubnet-01	subnet-0204f86f5f3553760	Available	vpc-0ea52c1769a34f6e8 web-...	Off	10.0.1.0/24

10. Target Groups

Target Groups are used by the Application Load Balancer to route incoming traffic to registered EC2 instances. The target group continuously performs health checks on the instances to ensure only healthy resources receive traffic. If an instance becomes unhealthy, it is automatically removed from traffic routing, improving reliability and application uptime.

web-targetgrp [Actions](#)

Details

[arn:aws:elasticloadbalancing:us-east-1:098167103976:targetgroup/web-targetgrp/d0ba31400f39ad77](#)

Target type	Protocol : Port	Protocol version	VPC
Instance	HTTP: 80	HTTP1	vpc-0ea52c1769a34f6e8
IP address type	Load balancer		
IPv4	web-app-load-balancer		

1	1 Healthy	0 Unhealthy	0 Unused	0 Initial	0 Draining
Total targets	0 Anomalous				

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.




11. Launch Template

A Launch Template defines the configuration required to launch EC2 instances automatically. It includes the Amazon Machine Image (AMI), instance type, security group, key pair, and user data script for software installation. The Auto Scaling Group uses this launch template to create consistent and identical EC2 instances during scaling events, ensuring uniform application behaviour.

web-template (lt-09d9f4c869342a2cd)

[Actions](#)[Delete template](#)




Launch template details

Launch template ID lt-09d9f4c869342a2cd**Launch template name** web-template**Default version** 1**Owner** arn:aws:iam::098167103976:root[Details](#)[Versions](#)[Template tags](#)

Launch template version details

[Actions](#)[Delete template version](#)**Version**

1 (Default)

Description Initial**Date created** 2026-01-08T16:08:10.000Z**Created by** arn:aws:iam::098167103976:root[Instance details](#)[Storage](#)[Resource tags](#)[Network interfaces](#)[Advanced details](#)**AMI ID** ami-07ff62358b87c7116**Instance type** t3.micro**Availability Zone**


-

Availability Zone Id

-

Key pair name mykwy**Security groups**

-

Security group IDs sg-0e1351e5c48bbe21f

Security Design

Security is implemented using a layered approach. Security groups act as virtual firewalls, allowing only essential inbound traffic such as HTTP (port 80) for web access and SSH (port 22) for administrative purposes. All outbound traffic is permitted to allow system updates and service communication. Access control is minimized to reduce the attack surface, and resources are automatically managed to avoid prolonged exposure of unused instances.

STRESS TESTING AND VALIDATION

ALARM: "CPU Utilisation" in US East (N. Virginia) inbox x

**my-sns** <no-reply@sns.amazonaws.com>
to me

10:11PM (3 minutes ago) ☆ 😊 ↶ ⋮

You are receiving this email because your Amazon CloudWatch Alarm "CPU Utilisation" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [100.0 (08/01/26 16:40:00)] was greater than the threshold (60.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Thursday 08 January, 2026 16:41:32 UTC".

View this alarm in the AWS Management Console:

<https://us-east-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=us-east-1#alarmsV2:alarm/CPU%20Utilisation>

Alarm Details:

- Name: CPU Utilisation
- Description:
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [100.0 (08/01/26 16:40:00)] was greater than the threshold (60.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Thursday 08 January, 2026 16:41:32 UTC
- AWS Account: 098167103976
- Alarm Arn: arn:aws:cloudwatch:us-east-1:098167103976:alarm:CPU Utilisation

Threshold:

- The alarm is in the ALARM state when the metric is GreaterThanThreshold 60.0 for at least 1 of the last 1 period(s) of 60 seconds.

Monitored Metric:

- MetricNamespace: AWS/EC2
- MetricName: CPUUtilization
- Dimensions: [InstanceId = i-06fd93af501d747ae]
- Period: 60 seconds
- Statistic: Average
- Unit: not specified
- TreatMissingData: missing

State Change Actions:

- OK:
- ALARM: [arn:aws:sns:us-east-1:098167103976:adithi-sns]
- INSUFFICIENT_DATA:

To test auto scaling:

- Connected to EC2 instance via SSH
- Executed stress command to increase CPU utilization
- Observed CPU crossing threshold
- CloudWatch alarm triggered
- Auto Scaling launched a new EC2 instance
- SNS email notification received

This confirms successful dynamic scaling.

```
r.adithi@Adithi MINGW64 ~/Downloads/devops
$ ssh -i mykwy.pem ec2-user@3.234.223.234

#_
~\####_ Amazon Linux 2023
~\#####\
~\###|
~\#/ https://aws.amazon.com/linux/amazon-linux-2023
~V~' '->
~
~._.
~/_/_/
~/m/'

[ec2-user@ip-10-0-1-28 ~]$ sudo apt update
sudo: apt: command not found
[ec2-user@ip-10-0-1-28 ~]$ apt update
-bash: apt: command not found
[ec2-user@ip-10-0-1-28 ~]$ stress --cpu 2 --timeout 300
stress: info: [26464] dispatching hogs: 2 cpu, 0 io, 0 vm, 0 hdd
stress: info: [26464] successful run completed in 300s
[ec2-user@ip-10-0-1-28 ~]$ |
```

Conclusion

The project successfully demonstrates the deployment of an auto-scaling web infrastructure in the cloud. By leveraging AWS services such as EC2, Application Load Balancer, Auto Scaling, CloudWatch, and SNS, the system efficiently adapts to changing workloads while maintaining high availability and cost efficiency. This implementation provides a strong foundation for understanding scalable cloud architectures and real-world infrastructure automation.

By -

R. Adithi

Aspiring DevOps Engineer

Batch - 06