Project Title: Flexible and Scalable Web Application Deployment on AWS

Project Description

The project aims to create a robust infrastructure on AWS for a web application, ensuring high availability and scalability. By utilising AWS services strategically, we'll design a system capable of efficiently managing fluctuating loads and maintaining uptime across multiple Availability Zones.

Objectives

1. High Availability : Ensure minimal downtime by leveraging multiple Availability Zones.
2. Scalability: Utilise AWS Auto Scaling to dynamically adjust resources based on traffic fluctuations.
3. Security: Implement robust security measures, including security groups and secure communication protocols.
4. Resilience: Design application infrastructure to withstand failures and traffic spikes without manual intervention.

Core AWS Services Utilisation :

1.Virtual Private Cloud (VPC): Provides an isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define. Elastic File System (EFS): Offers a simple, scalable, elastic file storage for use with AWS Cloud services and on-premises resources.

2.**Elastic Compute Cloud (EC2):** Provides scalable computing capacity in the AWS cloud. It allows developers to scale up or down based on demand. AWS Auto Scaling: Monitors applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.

3.Application Load Balancer (ALB): Automatically distributes incoming application traffic across multiple targets, such as EC2 instances, containers, and IP addresses.

4.Auto Scaling Group (ASG): in AWS is a tool that automatically adjusts the number of EC2 instances in your application based on demand. It helps keep your application running smoothly during traffic spikes and reduces costs during quieter periods. It also spreads instances across multiple locations for reliability.

5.Security Groups : in AWS are like virtual firewalls for your EC2 instances. They control inbound and outbound traffic based on rules you set. You can specify which protocols, ports, and IP addresses are allowed to communicate with your instances. They're applied at instance launch and can be modified anytime.

6.Amazon Simple Notification Service (SNS): is a messaging service on AWS that enables you to send notifications to various endpoints like email, SMS, or HTTP endpoints. It follows a publish-subscribe model and is highly reliable and scalable.

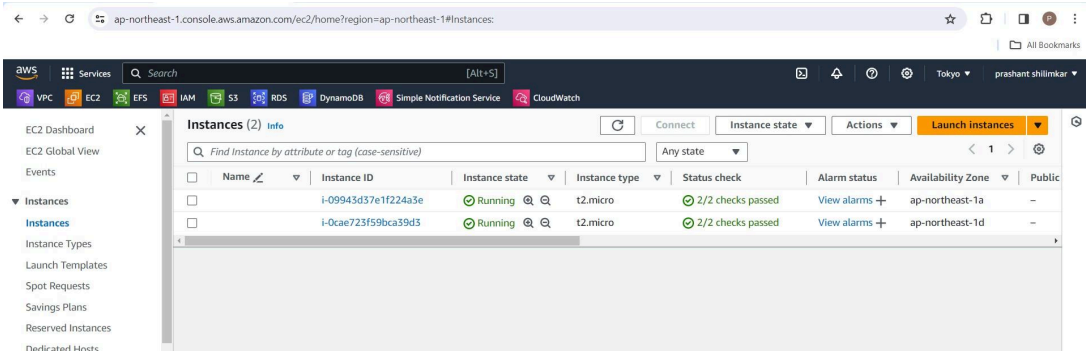
7.CloudWatch: is an AWS service for monitoring your resources and applications on the cloud. It collects data in the form of metrics and logs, sets alarms for specific conditions, and provides dashboards for visualisation. It helps you understand and optimise the performance of your AWS environment.

Deployment strategy

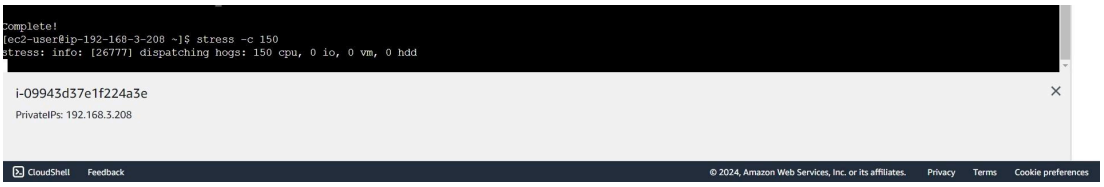
- **VPC Setup:**
    - Create a Virtual Private Cloud (VPC) with two private subnets and two public subnets across different Availability Zones (AZs) for fault tolerance.
    - Configure a public route table with an Internet Gateway (IGW) attached to provide internet access to instances in the public subnets.
    - Ensure that the private subnets do not have direct internet access by not associating them with the public route table. Instead, they can access the internet through a NAT Gateway or NAT Instance in the public subnet.
- **Launch Template Creation:**
    - Develop a launch template with a user data script that installs Apache HTTP Server (httpd) on instances launched by the Auto Scaling Group.
- **Auto Scaling Group Configuration:**
    - Create an Auto Scaling Group (ASG) with a minimum of two instances and a maximum of four instances, spread across the private subnets for high availability.
    - Use the launch template created earlier for configuring instances launched by the ASG.
- **Application Load Balancer (ALB) Setup**:
    - Create an Application Load Balancer (ALB) and place it in one of the public subnets to allow incoming traffic from clients.
    - Configure the ALB's target group to include instances in the private subnets, enabling it to route traffic to them based on health checks and load balancing algorithms.
- **SNS Configuration for Auto Scaling Events:**
    - Set up an SNS topic to receive notifications when CPU utilisation exceeds 75% on instances managed by the ASG.
    - Configure CloudWatch alarms to monitor CPU utilisation and trigger SNS notifications when thresholds are

Results and Performance Metrics

As you can see there are only 2 instance before cpu utilisation below than 75%

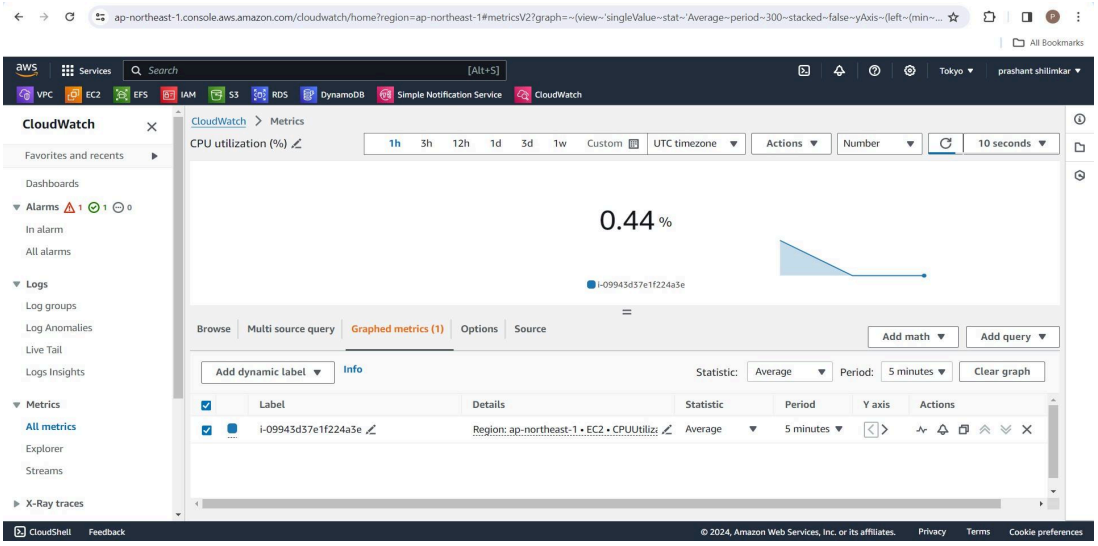


But after increasing the cpu utilisation by using **`#stress -c 150`** command

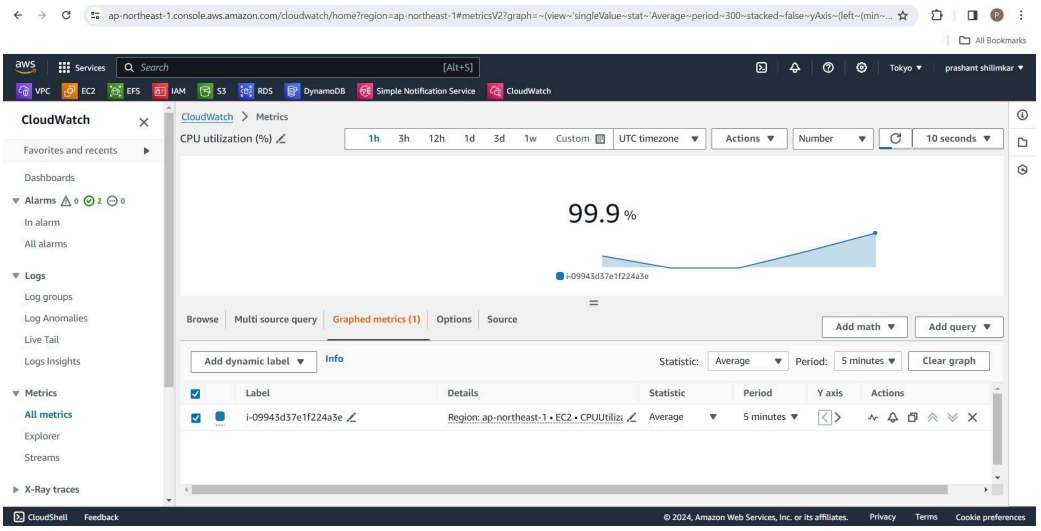


Now you see there is increase in cpu utilisation , here we use Cloud watch series

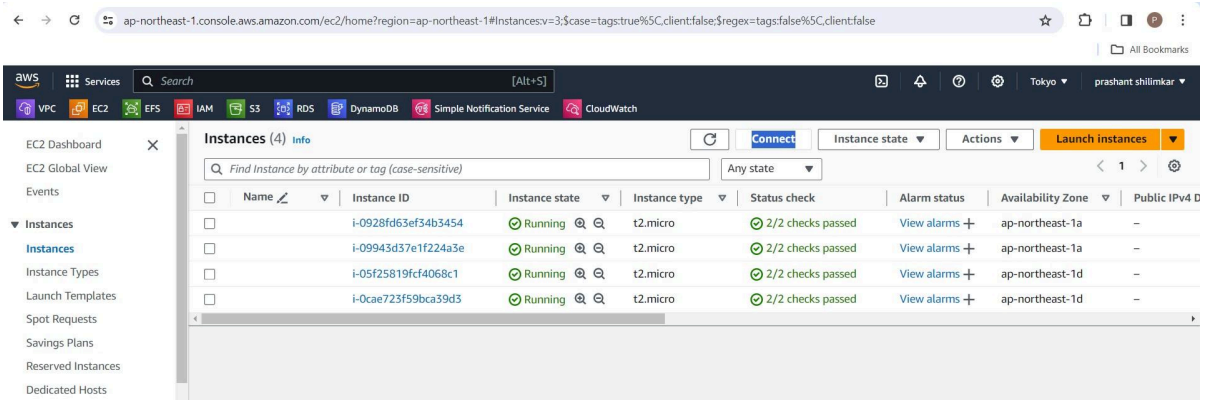
Before :



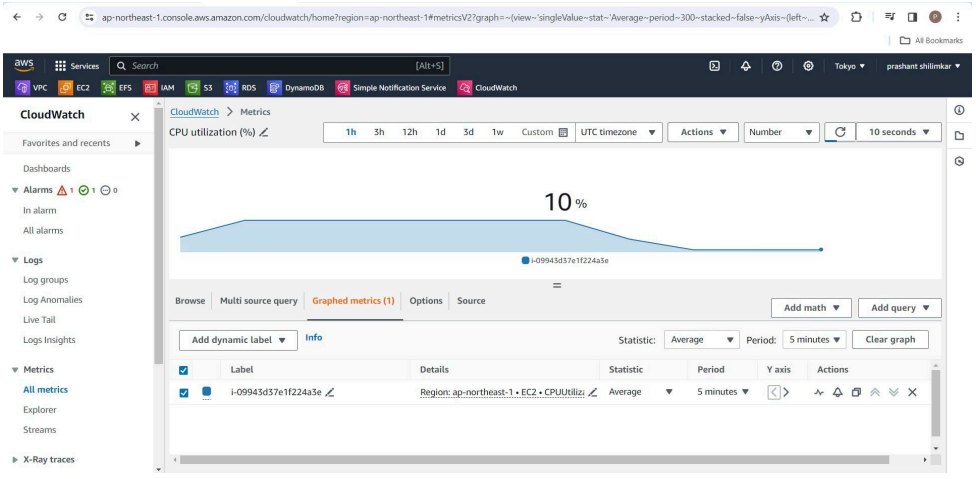
After:



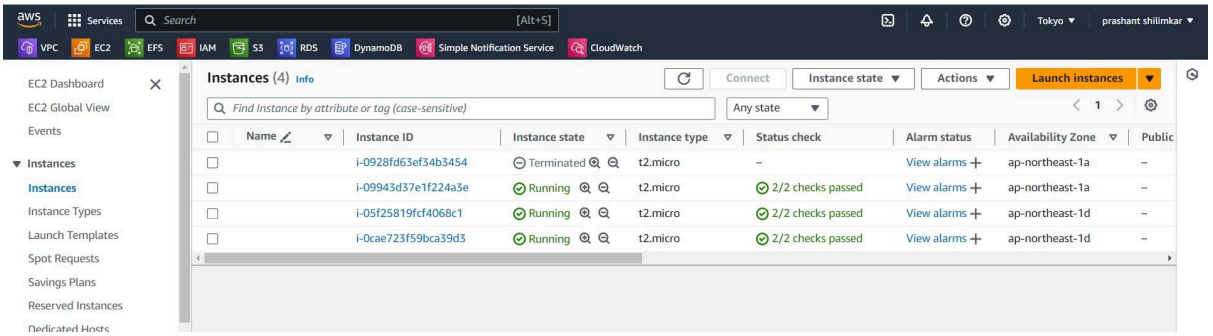
And here you see the number of instance is also increases as set as auto scaling configuration



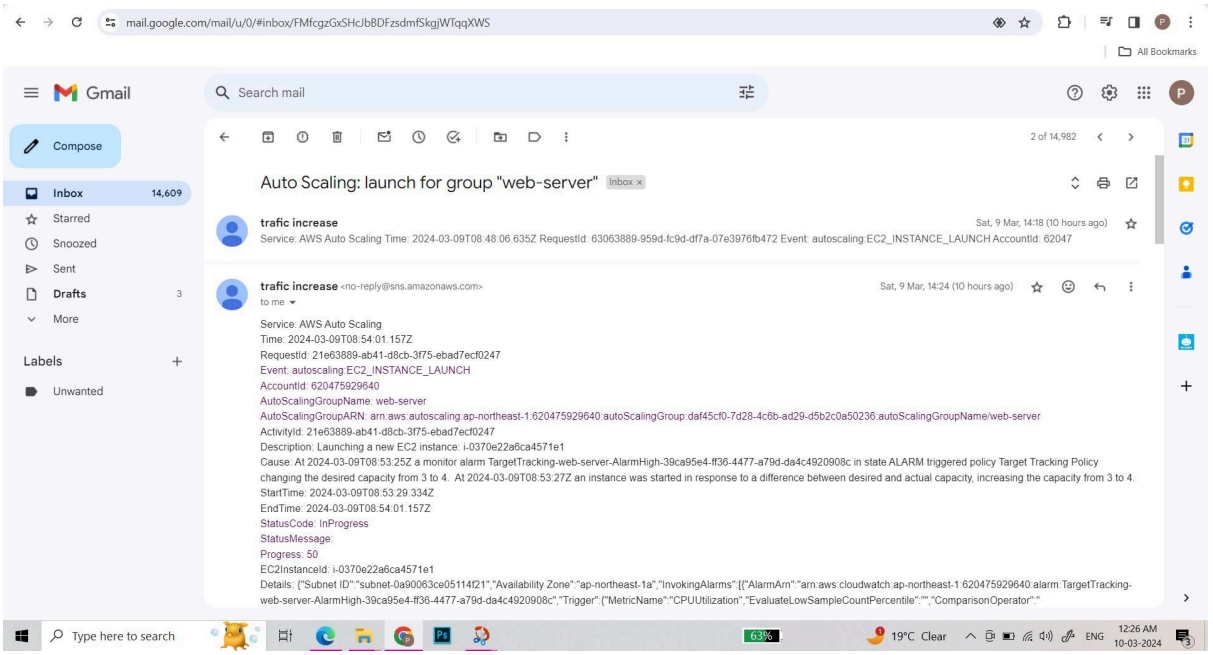
Now after the load (cpu utilisation) is decreased



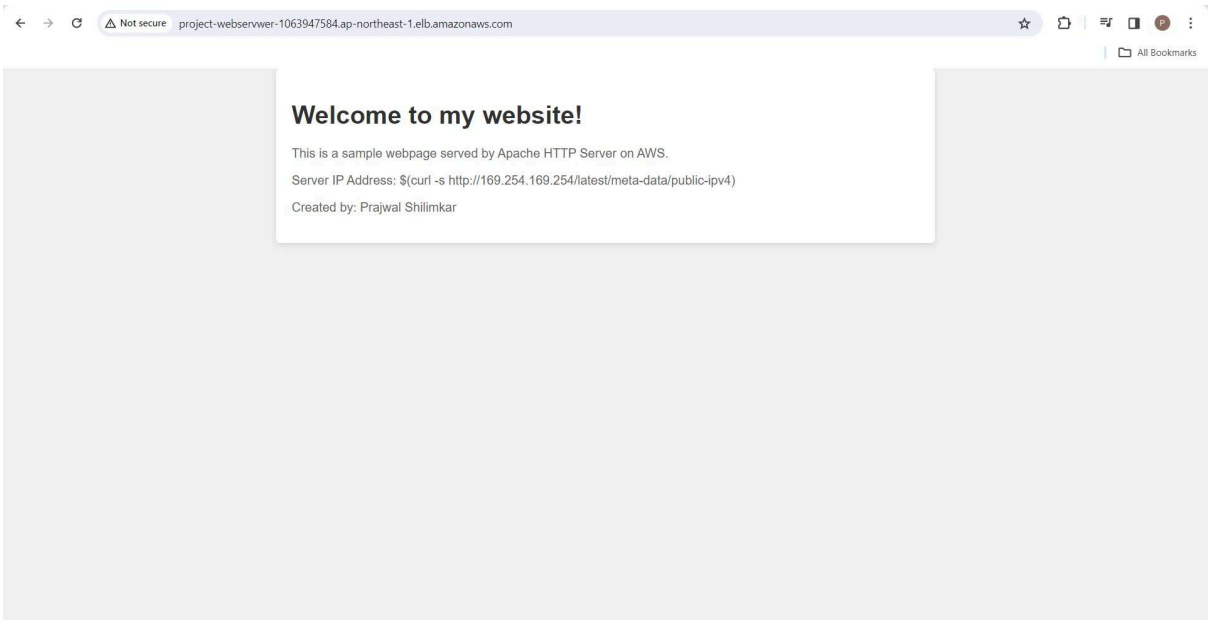
Then number of instance is also decrease



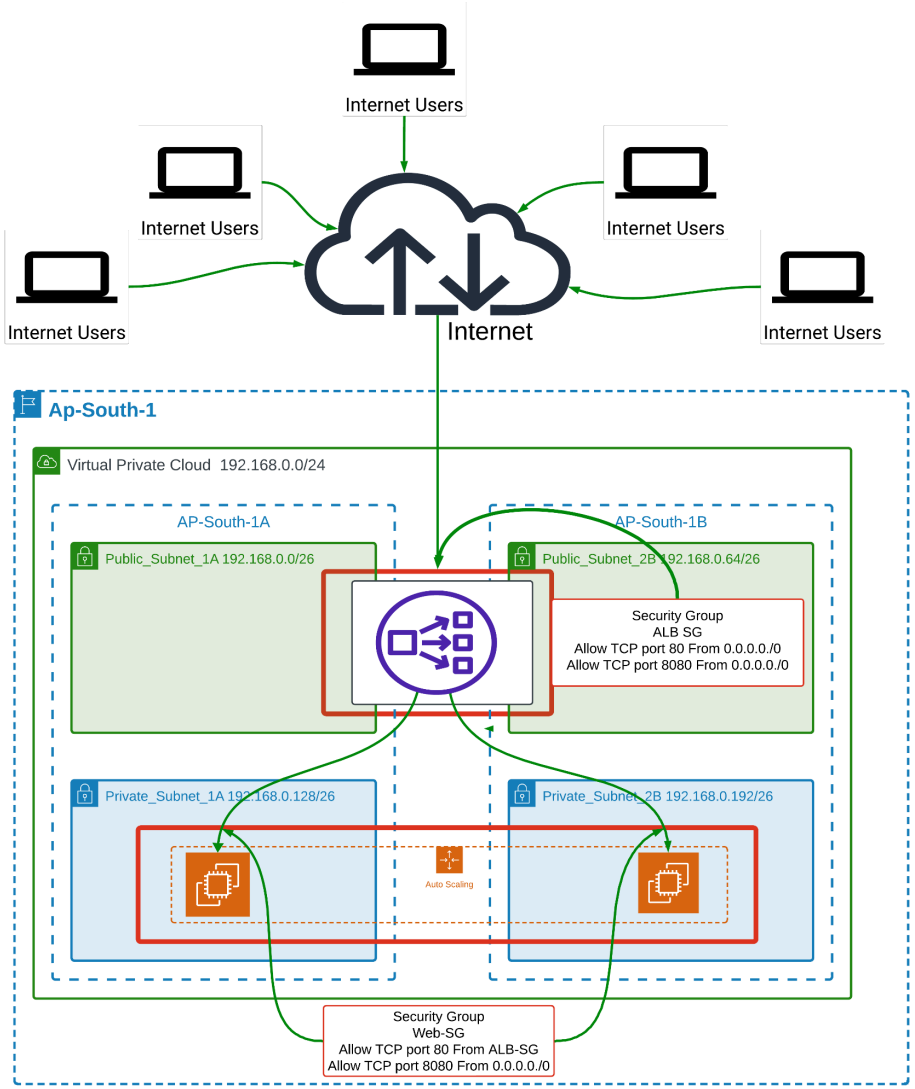
Here because of the SNS services we got an email which alert



Website:


Not secure
project-webservwer-1063947584.ap-northeast-1.elb.amazonaws.com
All Bookmarks
Welcome to my website!
This is a sample webpage served by Apache HTTP Server on AWS.
Server IP Address: $(curl -s http://169.254.169.254/latest/meta-data/public-ipv4)
Created by: Prajwal Shilimkar

FINAL DESIGN:



Architecture design

Conclusion

In conclusion, this deployment strategy outlines a comprehensive approach to deploying a scalable web application on AWS while adhering to best practices for security, availability, and automation. By following the steps outlined above, you can effectively set up a resilient infrastructure capable of handling varying loads, ensure high availability across multiple Availability Zones, and automate the deployment process for efficiency and consistency.

Key components of the strategy include setting up a well-designed VPC with public and private subnets, configuring an Auto Scaling Group to dynamically adjust resources based on demand, utilising an Application Load Balancer for distributing incoming traffic, and implementing monitoring and alerting through AWS CloudWatch and SNS.

Continuous testing, monitoring, and optimization are emphasised to ensure the reliability and performance of the deployed application. Documentation and training are also highlighted to

facilitate knowledge sharing and enable effective management and maintenance of the infrastructure by the operations team.

Overall, this deployment strategy provides a robust framework for deploying and managing a scalable web application on AWS, helping organisations achieve their goals of scalability, reliability, and cost-efficiency in the cloud environment.