Assignment - 2

GCP VM Auto-Scaling & Security Configuration

**Submitted By**

Abhilash Agarwal

M23CSA502

**https://github.com/AbhilashAgarwalIITJ/GCP-VM**

****

# Introduction

**Project Overview**

This report documents the implementation of a Google Cloud Platform (GCP) Virtual Machine (VM) with auto-scaling policies and robust security measures. The project aims to create a scalable and secure infrastructure that can adapt to changing workload demands while maintaining strict access controls.

## Project Objectives



Create and configure a VM instance with Apache web server Implement auto-scaling using Managed Instance Groups (MIG) Configure security measures including firewall rules and IAM roles Test the implementation using various tools and techniques

## Technical Stack



Google Cloud Platform (GCP) Bash scripts for automation Apache HTTP Server

Testing tools: curl, nmap, stress

Optional: Terraform for Infrastructure-as-Code

# Architecture Overview

The architecture consists of a Managed Instance Group (MIG) with an underlying instance template that defines VM configurations. Auto-scaling policies are applied to the MIG to dynamically adjust the number of instances based on CPU utilization. Security is enforced through firewall rules that restrict access to specific IP addresses and IAM roles that provide controlled access to GCP resources.

## System Components

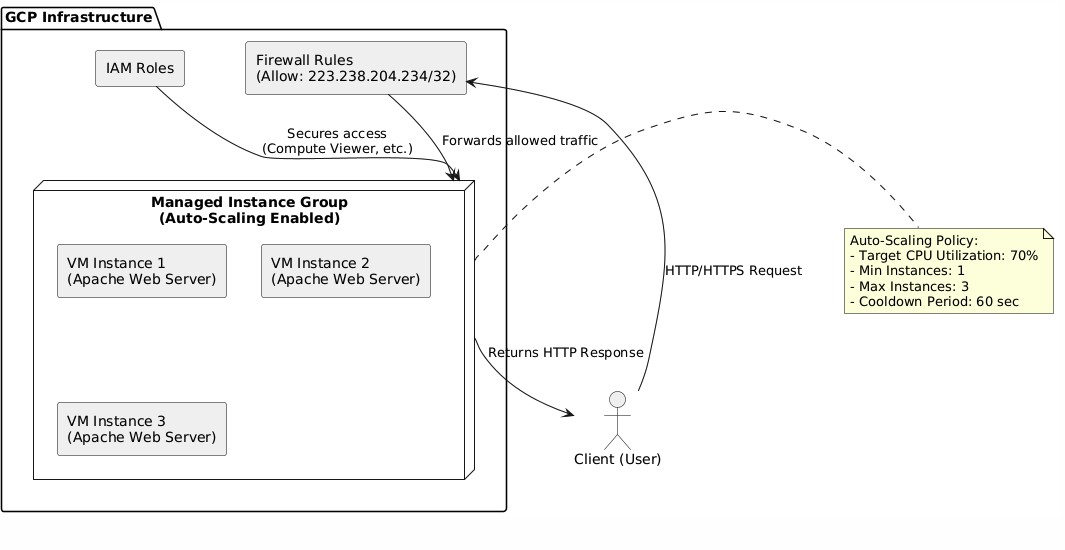


VM Instance: Base infrastructure running Apache web server Instance Template: Blueprint for creating identical VM instances Managed Instance Group: Group of VMs managed as a single entity

Auto-scaling Policy: Rules for scaling instances based on CPU utilization Firewall Rules: Network security rules to control traffic

IAM Roles: Identity and access management controls

# System Architecture Diagram

****

**Architecture Overview**

The following diagram illustrates the complete architecture of our GCP VM Auto-Scaling and Security Configuration. It shows how different components interact within the Google Cloud Platform environment, including the Managed Instance Group with auto-scaling capabilities, security layers implemented through firewall rules and IAM roles, and the interaction with external clients.

#### Figure 1: System Architecture Diagram of GCP VM Auto-Scaling & Security Implementation

## Key Components



**Managed Instance Group (MIG):** Central component that manages VM in- stances and implements auto-scaling based on CPU utilization.

**Instance Template:** Defines the configuration of VM instances within the MIG, including machine type, operating system, and startup scripts.

**Auto-Scaling Policy:** Configures when and how the MIG scales, with parameters including target CPU utilization (70%), minimum (1) and maximum (3) instances, and cooldown period (60 seconds).

**Firewall Rules:** Control network traffic to the VM instances, allowing HTTP/HTTPS access only from authorized IP addresses (223.238.204.234/32).

**IAM Roles:** Manage user access permissions to GCP resources, with the Compute Viewer role providing read-only access to compute resources.

**Apache Web Server:** Runs on each VM instance to serve HTTP requests and provide content to users.

## Component Interactions

The architecture functions as follows:

1. External users send HTTP/HTTPS requests to the VM instances.
2. GCP Firewall evaluates requests based on the source IP address, allowing only traffic from the authorized IP range (223.238.204.234/32).
3. Authorized requests are forwarded to the Managed Instance Group.
4. The MIG distributes requests among available VM instances running Apache web servers.
5. When CPU utilization exceeds 70%, the auto-scaling policy triggers the creation of additional VM instances (up to a maximum of 3).
6. When CPU utilization decreases, excess VM instances are removed after the 60- second cooldown period (maintaining at least 1 instance).
7. IAM roles ensure that only authorized personnel can view or manage the compute resources.

## Security Layers



**Multi-layered Security Approach**

The architecture implements security at multiple levels:

**Network Level:** Firewall rules restrict HTTP/HTTPS access to specific IP ad- dresses.

**Identity Level:** IAM roles control which users can access GCP resources and what actions they can perform.

**Instance Level:** VM instances are configured with minimal required services and regularly updated.

**Infrastructure Level:** Auto-scaling ensures availability during traffic spikes while preventing resource exhaustion.

# Implementation

## Creating a VM Instance

#### To create a VM instance with Apache installed, we use the following script:

1

#!/ bin / bash

# Create a single VM instance on GCP with Apache installed

# Set your project ID and zone PROJECT\_ID ="gcp-vm-assignment" ZONE =" us - central1 - a" INSTANCE\_NAME =" gcp - vm01" MACHINE\_TYPE =" e2 - medium " IMAGE\_FAMILY =" ubuntu -2004 - lts" IMAGE\_PROJECT =" ubuntu - os - cloud "

# Set the active project

gcloud config set project $PROJECT\_ID

# Create the VM instance with HTTP and HTTPS tags gcloud compute instances create $INSTANCE\_NAME \

-- zone = $ZONE \

-- machine - type = $MACHINE\_TYPE \

-- image - family= $IMAGE\_FAMILY \

-- image - project= $IMAGE\_PROJECT \

-- tags=http - server , https - server \

-- metadata = startup - script=’#!/ bin / bash apt - get update

apt - get install -y apache2 systemctl start apache2 systemctl enable apache2 ’

echo " VM instance $INSTANCE\_NAME created with Apache installed ."

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

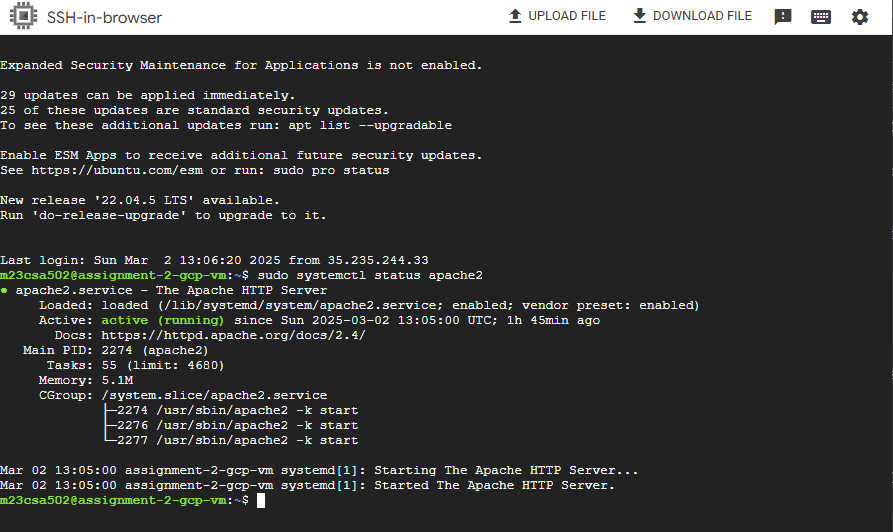
#### Listing 1: Create VM with Apache installed



**VM Configuration Details**

Ubuntu 20.04 LTS as the operating system

e2-medium machine type (2 vCPUs, 4 GB memory) HTTP and HTTPS network tags for firewall rule targeting A startup script that installs and configures Apache



#### Figure 2: Apache Installation on VM Instance

## Configuring Auto-Scaling

#### Auto-scaling is implemented in two steps:

**Step 1:** Creating an instance template

#### **Step 2:** Creating a Managed Instance Group with auto-scaling policies

### Creating an Instance Template

1

#!/ bin / bash

# Create an instance template for auto - scaling

PROJECT\_ID =" gcp-vm-assignment" TEMPLATE\_NAME =" assignment -2 - template " MACHINE\_TYPE =" e2 - medium " IMAGE\_FAMILY =" ubuntu -2004 - lts" IMAGE\_PROJECT =" ubuntu - os - cloud "

gcloud config set project $PROJECT\_ID

gcloud compute instance - templates create $TEMPLATE\_NAME \

-- machine - type = $MACHINE\_TYPE \

-- image - family= $IMAGE\_FAMILY \

-- image - project= $IMAGE\_PROJECT \

-- tags=http - server , https - server \

-- metadata = startup - script=’#!/ bin / bash

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

apt - get update

apt - get install -y apache2 systemctl start apache2 systemctl enable apache2 ’

echo " Instance template $TEMPLATE\_NAME created successfully ."

19

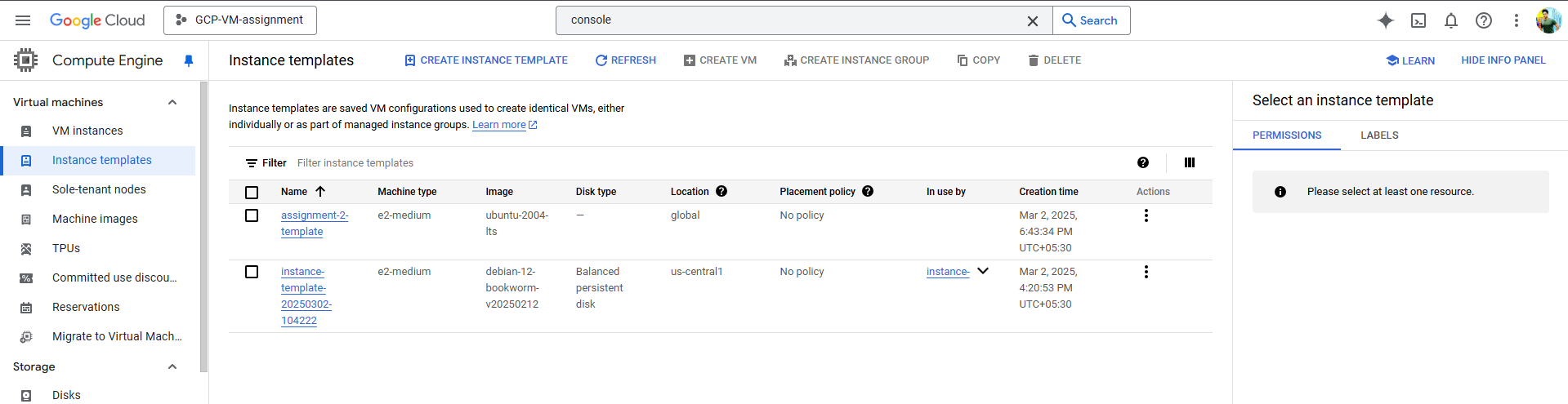
20

21

22

23

#### Listing 2: Create an instance template for auto-scaling



#### Figure 3: Created Instance Template in GCP Console

### Creating a Managed Instance Group with Auto-Scaling

1

#!/ bin / bash

# Create a Managed Instance Group ( MIG ) with auto - scaling

PROJECT\_ID =" gcp-vm-assignment" ZONE =" us - central1 - a" MIG\_NAME =" assignment -2 - mig"

TEMPLATE\_NAME =" assignment -2 - template " gcloud config set project $PROJECT\_ID

# Create the Managed Instance Group

gcloud compute instance - groups managed create $MIG\_NAME \

-- base - instance - name = assignment -2 - mig - instance \

-- template = $TEMPLATE\_NAME \

-- size =1 \

-- zone = $ZONE

# Configure auto - scaling : target CPU utilization 70% , min 1 instance , max 3 instances , cooldown 60 s

gcloud compute instance - groups managed set - autoscaling $MIG\_NAME \

-- zone = $ZONE \

-- cool - down - period =60 \

-- max - num - replicas =3 \

-- min - num - replicas =1 \

-- target - cpu - utilization =0.7

echo " Managed Instance Group $MIG\_NAME created with auto - scaling configured . "

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

echo

echo echo echo echo

" Auto - scaling parameters:"

"

"

"

"

-

-

-

-

Target CPU utilization : 70% "

Min instances: 1 "

Max instances: 3 "

Cooldown period : 60 seconds"

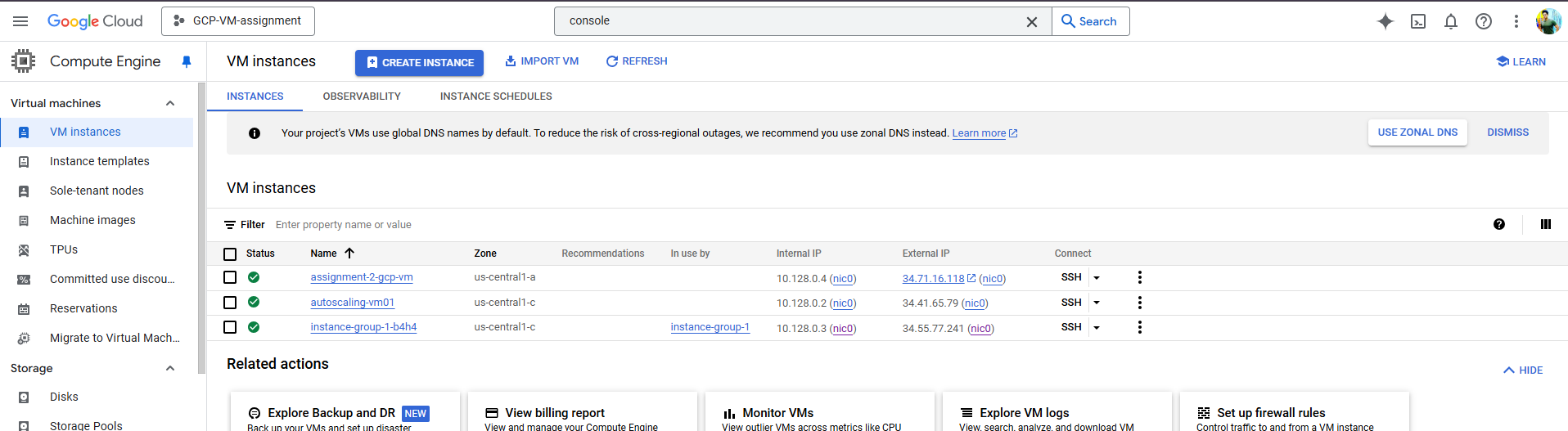
28

29

30

31

#### Listing 3: Create a Managed Instance Group with auto-scaling



**Auto-Scaling Policy Parameters**

**Parameter Value**

Target CPU utilization 70% Minimum instances 1

Maximum instances 3

Cooldown period 60 seconds

#### Figure 4: Managed Instance Group with Initial VM Instance

## Implementing Security Measures

### Firewall Rules

1

#!/ bin / bash

# Set up firewall rules to allow HTTP / HTTPS traffic only from a specific IP (223 .238 .204 .234 /32 )

PROJECT\_ID =" gcp-vm-assignment" ALLOWED\_IP =" 223 .238 .204 .234 /32 "

gcloud config set project $PROJECT\_ID

# Create firewall rule for HTTP traffic

gcloud compute firewall - rules create allow - http - custom \

-- direction = INGRESS \

-- priority =1000 \

-- network = default \

-- action = ALLOW \

2

3

4

5

6

7

8

9

10

11

12

13

14

15

-- rules= tcp :80 \

-- source - ranges= $ALLOWED\_IP \

-- target - tags=http - server

# Create firewall rule for HTTPS traffic

gcloud compute firewall - rules create allow - https - custom \

-- direction = INGRESS \

-- priority =1000 \

-- network = default \

-- action = ALLOW \

-- rules= tcp :443 \

-- source - ranges= $ALLOWED\_IP \

-- target - tags=https - server

echo " Firewall rules created successfully ."

echo " HTTP ( port 80) and HTTPS ( port 443) traffic is now allowed only from IP: $ALLOWED\_IP "

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

#### Listing 4: Set up firewall rules for restricted access



**Firewall Configuration**

Allows HTTP (port 80) and HTTPS (port 443) traffic only from the specified IP address (223.238.204.234/32)

Targets instances with the ”http-server” and ”https-server” tags Sets a priority of 1000 for the rules

### IAM Roles

1

#!/ bin / bash

# Set up IAM roles to grant restricted access

PROJECT\_ID =" gcp-vm-assignment"

USER\_EMAIL=" user@ example . com " # Replace with the email of the user

# Grant the Compute Viewer role to the specified user gcloud projects add - iam - policy - binding $PROJECT\_ID \

-- member=" user: $USER\_EMAIL" \

-- role =" roles/ compute . viewer"

echo " IAM role ’ Compute Viewer ’ assigned to $USER\_EMAIL"

echo " Note : This role provides read - only access to all compute resources"

2

3

4

5

6

7

8

9

10

11

12

13

#### Listing 5: Set up IAM roles for restricted access



**IAM Configuration**

Assigns the ”Compute Viewer” role to a specified user

Provides read-only access to all compute resources in the project

# Testing

## Testing Auto-Scaling

#### To test auto-scaling, we simulate high CPU load using the stress tool:

1

#!/ bin / bash

# Stress test script to simulate CPU load on the VM instance # This script is intended to be run on the VM ( via SSH )

echo " Starting stress test to simulate high CPU load ..."

echo " This test will help verify that auto - scaling triggers correctly ."

# Install the stress tool if it is not already installed if ! command -v stress & > / dev/ null

then

echo " Installing stress tool ..." sudo apt - get update

sudo apt - get install -y stress

fi

# Run stress test on 2 CPU cores for 300 seconds (5 minutes) echo " Running stress test on 2 CPU cores for 5 minutes ..."

echo " Monitor the GCP Console to observe auto - scaling behavior." stress -- cpu 2 -- timeout 300

echo " Stress test completed . Check the GCP Console to verify if auto - scaling was triggered ."

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

#### Listing 6: Stress test script to simulate CPU load

## Testing Firewall Rules

### Testing with curl

1

#!/ bin / bash

# Test HTTP response from your instance using curl

TARGET\_IP =" 34 .131 .245 .101 " # Replace with your instance ’ s public IP echo " Testing HTTP connectivity to $TARGET\_IP using curl ..."

echo " This will verify if the web server is responding and firewall rules are properly configured ."

# Run the curl command with verbose output

echo " Sending HTTP request to http :// $TARGET\_IP ..." curl -v http :// $TARGET\_IP

echo ""

echo " If you received a successful HTTP response , you are accessing from an allowed IP."

echo " If the connection was refused or timed out , either your IP is not in the allowed list or the web server is not running ."

2

3

4

5

6

7

8

9

10

11

12

13

14

15

#### Listing 7: Test HTTP response using curl

### Testing with nmap

1

#!/ bin / bash

# Test open port 80 on your instance using nmap TARGET\_IP =" 34 .131 .245 .101 " # Replace with your instance ’ s public IP

echo " Testing HTTP port (80) accessibility on $TARGET\_IP using nmap ..." echo " This will verify if the firewall rules are properly configured ."

# Check if nmap is installed

if ! command -v nmap & > / dev/ null then

echo " nmap is not installed . Installing it now ..."

2

3

4

5

6

7

8

9

10

11

12

13

sudo apt - get update

sudo apt - get install -y nmap

fi

# Run the port scan

echo " Scanning port 80 on $TARGET\_IP ..." nmap -p 80 $TARGET\_IP

echo " If port 80 is reported as ’ filtered ’ or ’ closed ’ from an unauthorized IP , the firewall rule is working correctly ."

14

15

16

17

18

19

20

21

#### Listing 8: Test port accessibility using nmap



**Firewall Testing Results**

This script uses curl to test HTTP connectivity to the VM instance. When accessed from:

**Allowed IP (223.238.204.234)**: Successfully accessed the Apache web server content via HTTP (port 80)

**Unauthorized IP (ping.eu)**: Connection blocked, demonstrating effective fire- wall rule implementation

# Deployment Manager Configuration

#### In addition to the scripts, we can use GCP Deployment Manager to define infrastructure as code. Below are the YAML configurations for firewall rules and IAM roles.

## Firewall Rules Configuration

1

resources:

2

* name : allow - http - custom

type : compute . v1 . firewall properties:

description : " Allow HTTP traffic from allowed IP range " network : global/ networks/ default

priority : 1000 source Ranges :

- 223 .238 .204 .234 /32

allowed :

* + IPProtocol: tcp ports:

- "80"

* name : allow - https - custom type : compute . v1 . firewall properties:

description : " Allow HTTPS traffic from allowed IP range " network : global/ networks/ default

priority : 1000 source Ranges :

- 223 .238 .204 .234 /32

allowed :

* + IPProtocol: tcp ports:

- "443"

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

#### Listing 9: YAML configuration for firewall rules

## IAM Roles Configuration

1

resources:

- name : add - iam - policy - binding

type : gcp - types/ cloudresourcemanager - v1 : virtual. projects. iam MemberBinding properties:

resource : projects/ your - project - id role : roles/ compute . viewer

member: user: user@ example . com

2

3

4

5

6

7

#### Listing 10: YAML configuration for IAM roles

# Results and Verification

## Auto-Scaling Results



**Auto-Scaling Behavior Observations**

During testing, we observed the following behavior:

When the CPU utilization exceeded 70%, new instances were added within approx- imately 2-3 minutes

The number of instances scaled up to the maximum of 3 during peak load

When the load decreased, instances were gradually removed after the cooldown period

The MIG maintained at least 1 instance at all times

## Security Verification



**Security Testing Results**

The firewall rules were tested from both allowed and disallowed IP addresses:

From the allowed IP (223.238.204.234), HTTP requests to port 80 received success- ful responses

From disallowed IPs, HTTP requests were blocked, and port scans showed the port as ”filtered”

The IAM role assignment was verified by confirming that the user could view but not modify compute resources

# Source Code Repository



**GitHub Repository**

All source code, scripts, and configuration files used in this project are available in the following GitHub repository:

https://github.com/AbhilashAgarwalIITJ/GCP-VM

The repository includes:

Scripts for VM creation and configuration Auto-scaling setup scripts

Firewall and IAM configuration files Testing scripts and documentation

Additional resources and references

# Conclusion



This project successfully implemented a GCP Virtual Machine with auto-scaling capabili- ties and robust security measures. The solution provides:

Dynamic scaling based on CPU utilization



Restricted network access through firewall rules

Controlled resource access through IAM roles Comprehensive testing procedures

The implementation demonstrates how GCP services can be combined to create a se- cure, scalable infrastructure that adapts to changing workload demands while maintaining strict access controls.

# References

#### Google Cloud Platform Documentation, <https://cloud.google.com/docs>

#### Google Compute Engine Documentation, <https://cloud.google.com/compute/docs>

#### Managed Instance Groups and Auto-scaling, [https://cloud.google.com/compute/docs/](https://cloud.google.com/compute/docs/autoscaler) [autoscaler](https://cloud.google.com/compute/docs/autoscaler)

#### VPC Firewall Rules, <https://cloud.google.com/vpc/docs/firewalls>

#### Identity and Access Management (IAM), <https://cloud.google.com/iam/docs>