Assignment - 2

GCP VM Auto-Scaling & Security Configuration

Submitted By

Anuj Chincholikar
B22ES018

March 1, 2025

Contents

1 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.

1.1 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

1.2 Technical Stack

- Google Cloud Platform (GCP)
- Bash scripts for automation
- Apache HTTP Server
- Testing tools: curl, nmap, stress
- Optional: Terraform for Infrastructure-as-Code

2 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.

2.1 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

3 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.

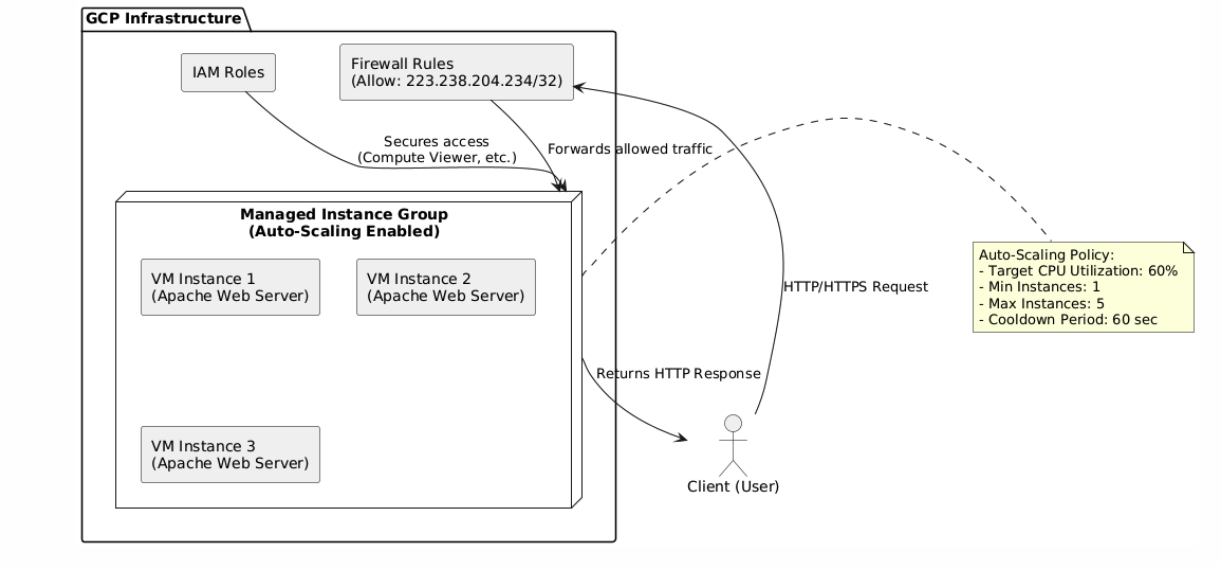

Figure 1: System Architecture Diagram of GCP VM Auto-Scaling & Security Implementation

3.1 Key Components

- **Managed Instance Group (MIG):** Central component that manages VM instances 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 (60%), minimum (1) and maximum (5) 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.

3.2 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 60%, the auto-scaling policy triggers the creation of additional VM instances (up to a maximum of 5).
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.

3.3 Security Layers

Multi-layered Security Approach

The architecture implements security at multiple levels:

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

4 Implementation

4.1 Creating a VM Instance

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

```
#!/bin/bash
# Create a single VM instance on GCP with Apache installed

# Set your project ID and zone
PROJECT_ID="your-project-id"
ZONE="us-central1-a"
INSTANCE_NAME="assignment-2-gcp-vm"
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."
```

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

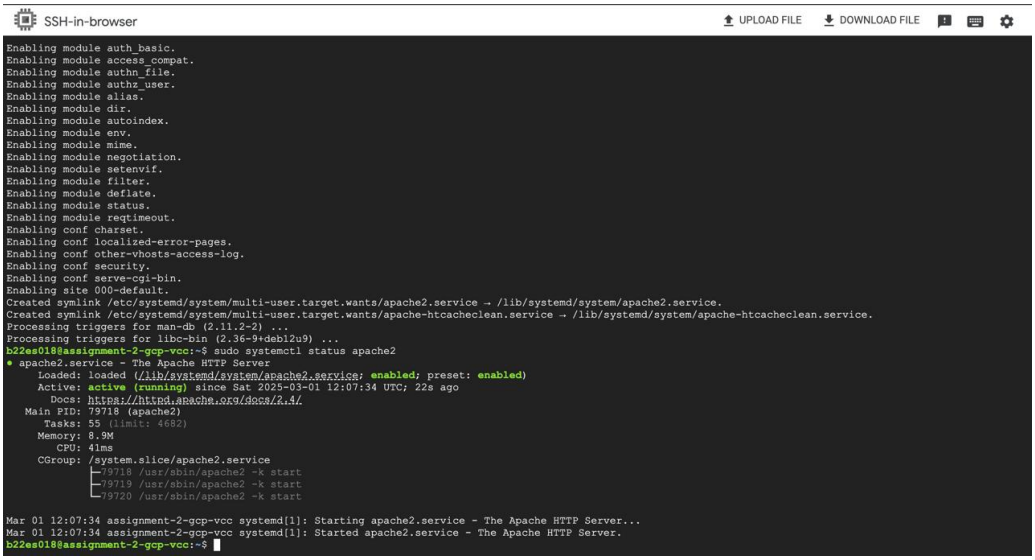

Figure 2: Apache Installation on VM Instance

4.2 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

4.2.1 Creating an Instance Template

```
#!/bin/bash
# Create an instance template for auto-scaling

PROJECT_ID="your-project-id"
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
```

```
    apt-get update
    apt-get install -y apache2
    systemctl start apache2
    systemctl enable apache2'

echo "Instance template $TEMPLATE_NAME created successfully."
```

Listing 2: Create an instance template for auto-scaling

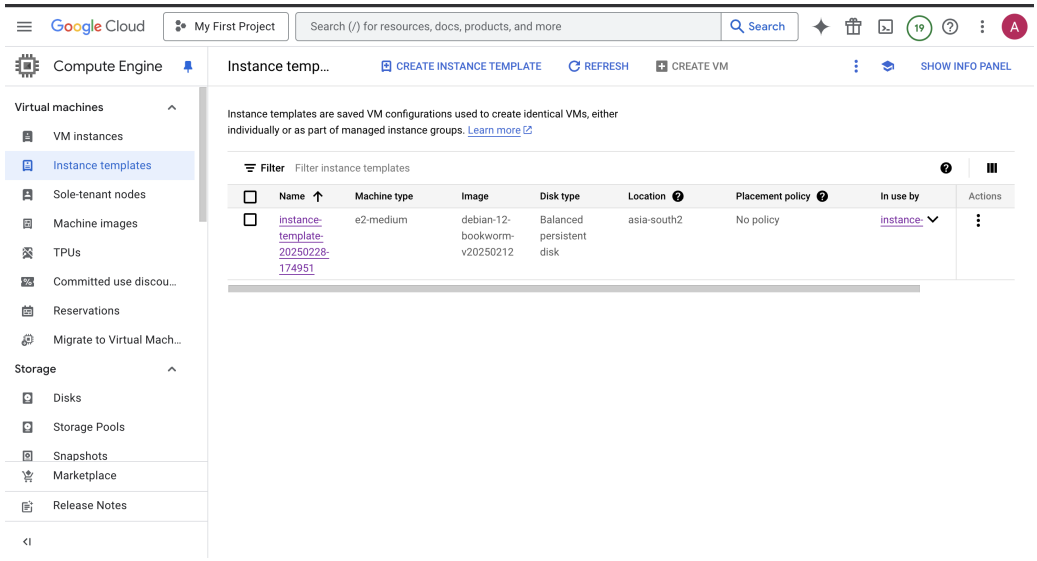

Figure 3: Created Instance Template in GCP Console

4.2.2 Creating a Managed Instance Group with Auto-Scaling

```
#!/bin/bash
# Create a Managed Instance Group (MIG) with auto-scaling

PROJECT_ID="your-project-id"
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 60%, min 1 instance, max 5 instances, cooldown 60s
gcloud compute instance-groups managed set-autoscaling $MIG_NAME \
  --zone=$ZONE \
  --cool-down-period=60 \
  --max-num-replicas=5 \
  --min-num-replicas=1 \
  --target-cpu-utilization=0.6

echo "Managed Instance Group $MIG_NAME created with auto-scaling configured."
```

```
echo "Auto-scaling parameters:"
echo "  - Target CPU utilization: 60%"
echo "  - Min instances: 1"
echo "  - Max instances: 5"
echo "  - Cooldown period: 60 seconds"
```

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

Auto-Scaling Policy Parameters

Parameter	Value
Target CPU utilization	60%
Minimum instances	1
Maximum instances	5
Cooldown period	60 seconds

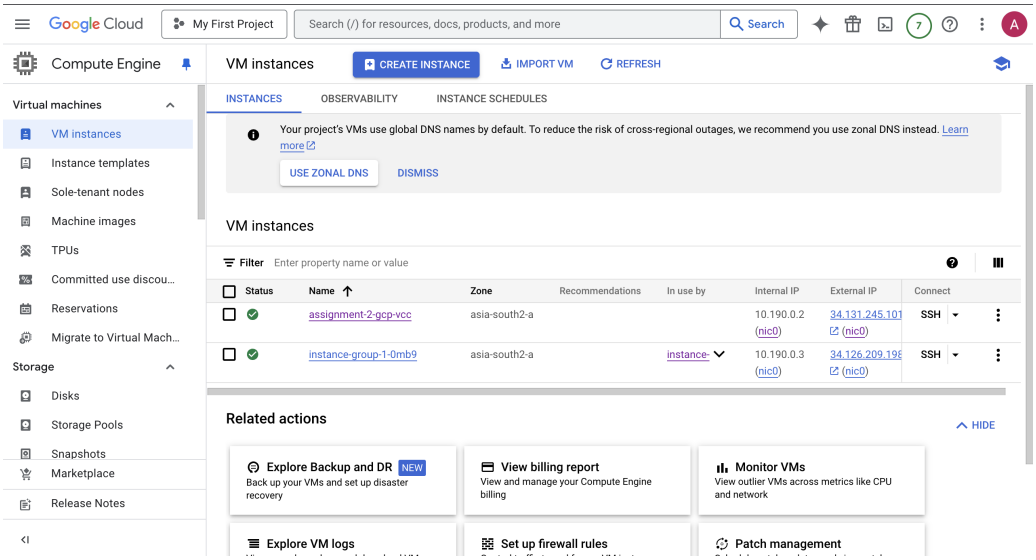

Figure 4: Managed Instance Group with Initial VM Instance

4.3 Implementing Security Measures

4.3.1 Firewall Rules

```
#!/bin/bash
# Set up firewall rules to allow HTTP/HTTPS traffic only from a specific IP
    (223.238.204.234/32)

PROJECT_ID="your-project-id"
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 \
```

```
  --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"
```

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

4.3.2 IAM Roles

```
#!/bin/bash
# Set up IAM roles to grant restricted access

PROJECT_ID="your-project-id"
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"
```

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

5 Testing

5.1 Testing Auto-Scaling

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

```
#!/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."
```

Listing 6: Stress test script to simulate CPU load

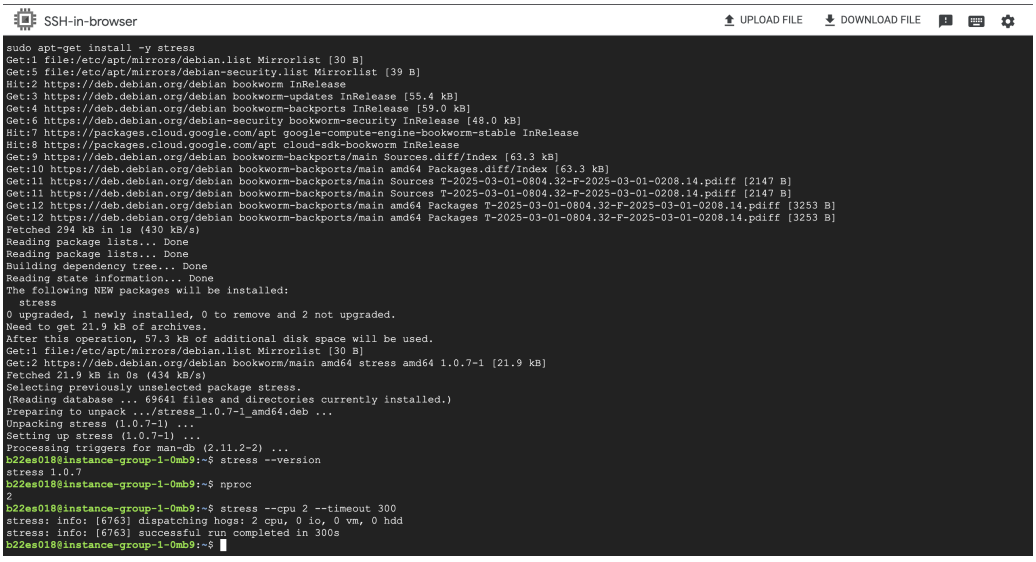

Figure 5: CPU Stress Test Running on VM Instance

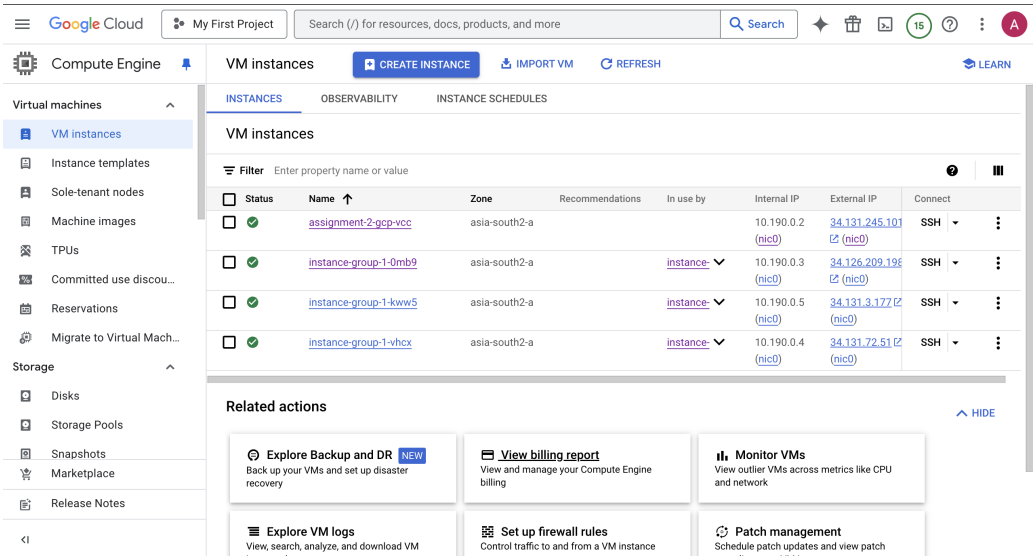

Figure 6: Auto-scaling: New VMs Added to Instance Group During High CPU Load

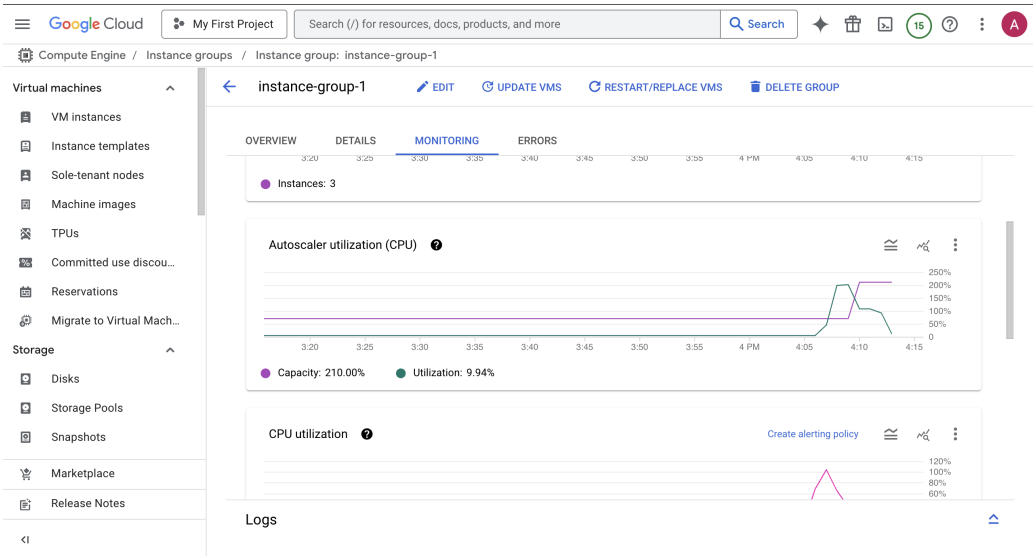

Figure 7: CPU Utilization Graph Showing Auto-scaling Trigger and Cooldown

Auto-Scaling Test Process

1. Installs the stress tool if not already installed
2. Runs a stress test on 2 CPU cores for 5 minutes
3. Monitors the GCP Console for auto-scaling events
4. Observes the addition of new VMs when CPU utilization exceeds 60%
5. After the cooldown period of 60 seconds, excess VMs are removed

5.2 Testing Firewall Rules

5.2.1 Testing with curl

```bash
#!/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."
```

Listing 7: Test HTTP response using curl

```
~ — -zsh
(base) anujchincholikar@Anujs-MacBook-Air ~ % curl -v http://34.131.245.101:80

*   Trying 34.131.245.101:80...
* Connected to 34.131.245.101 (34.131.245.101) port 80
> GET / HTTP/1.1
> Host: 34.131.245.101
> User-Agent: curl/8.2.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< Date: Sat, 01 Mar 2025 13:05:43 GMT
< Server: Apache/2.4.62 (Debian)
< Last-Modified: Sat, 01 Mar 2025 12:07:33 GMT
< ETag: "29cd-62f46c49c50e7"
< Accept-Ranges: bytes
< Content-Length: 10701
< Vary: Accept-Encoding
< Content-Type: text/html
<

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Debian Default Page: It works</title>
    <style type="text/css" media="screen">
  * {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
  }

  body, html {
    padding: 3px 3px 3px 3px;

    background-color: #D8DBE2;

    font-family: Verdana, sans-serif;
```

Figure 8: Successful HTTP Access from Allowed IP (Local Machine)

5.2.2 Testing with nmap

```bash
#!/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..."
```

```
    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."
```

Listing 8: Test port accessibility using nmap

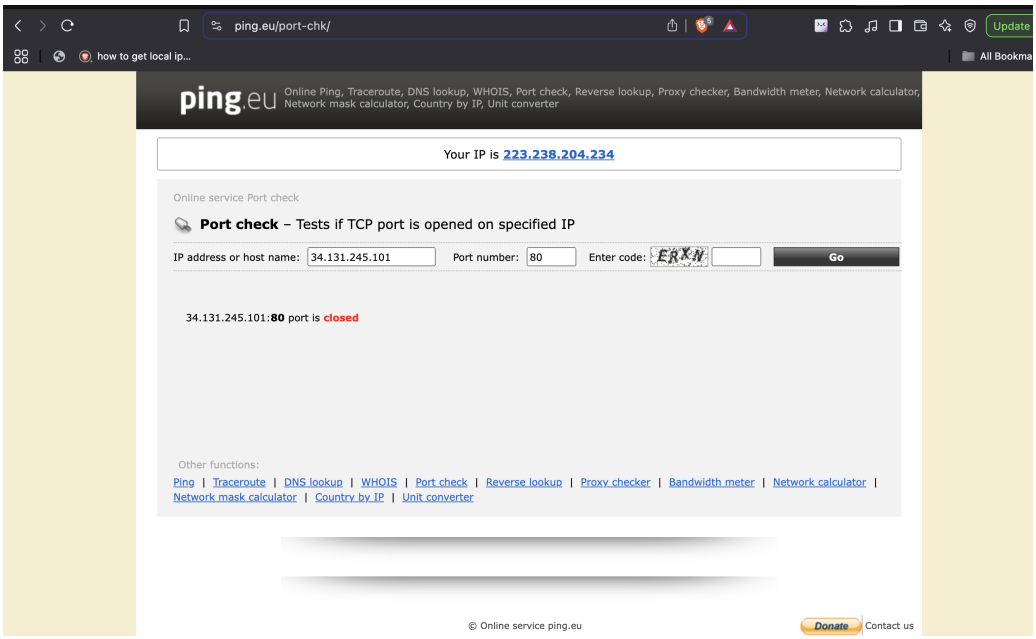

Figure 9: Blocked Access Attempt from Unauthorized IP (ping.eu)

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 firewall rule implementation

6 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.

6.1 Firewall Rules Configuration

```
resources:
```

```
- name: allow-http-custom
  type: compute.v1.firewall
  properties:
    description: "Allow HTTP traffic from allowed IP range"
    network: global/networks/default
    priority: 1000
    sourceRanges:
      - 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
    sourceRanges:
      - 223.238.204.234/32
    allowed:
      - IPProtocol: tcp
        ports:
          - "443"
```

Listing 9: YAML configuration for firewall rules

6.2 IAM Roles Configuration

```
resources:
- name: add-iam-policy-binding
  type: gcp-types/cloudresourcemanager-v1:virtual.projects.iamMemberBinding
  properties:
    resource: projects/your-project-id
    role: roles/compute.viewer
    member: user:user@example.com
```

Listing 10: YAML configuration for IAM roles

7 Results and Verification

7.1 Auto-Scaling Results

Auto-Scaling Behavior Observations

During testing, we observed the following behavior:

- When the CPU utilization exceeded 60%, new instances were added within approximately 2-3 minutes
- The number of instances scaled up to the maximum of 5 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

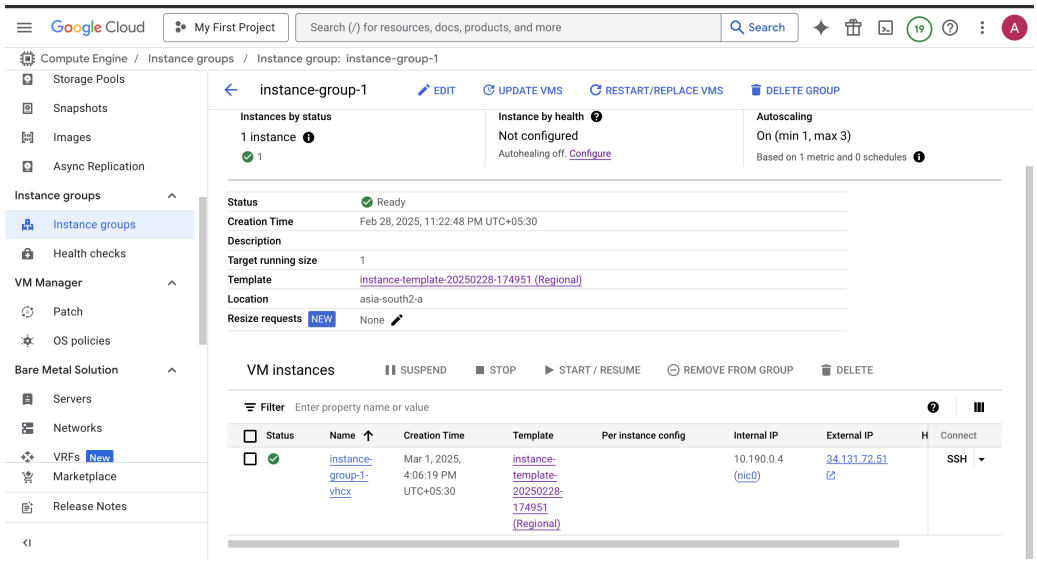

Figure 10: VM Instances Scaled Down After Cooldown Period

7.2 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 successful 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

8 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/chinanuj/gcp-vm-autoscaling-project

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

9 Video Demonstration

Video Demonstration

A comprehensive video demonstration of the project implementation and testing is available at the following Google Drive link:

https://drive.google.com/file/d/
1hb1b8eTiS7tSDILG6O9OofhVpM9a1W0A/view?usp=sharing]

The video covers:

- Step-by-step implementation process
- Auto-scaling demonstration with CPU load testing
- Firewall rules verification and testing
- IAM role configuration and verification
- Explanation of key concepts and best practices

10 Conclusion

This project successfully implemented a GCP Virtual Machine with auto-scaling capabilities 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 secure, scalable infrastructure that adapts to changing workload demands while maintaining strict access controls.

11 Future Enhancements

Potential Future Enhancements

Potential enhancements to the current implementation include:

- Implementing HTTPS with SSL certificates
- Adding load balancing for improved availability
- Configuring monitoring and alerting
- Implementing more sophisticated auto-scaling policies (e.g., based on request rate or memory usage)
- Expanding the infrastructure to multiple regions for disaster recovery

References

[1] Google Cloud Platform Documentation, https://cloud.google.com/docs

[2] Google Compute Engine Documentation, https://cloud.google.com/compute/docs

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

[4] VPC Firewall Rules, https://cloud.google.com/vpc/docs/firewalls

[5] Identity and Access Management (IAM), https://cloud.google.com/iam/docs