Assignment - 2

GCP VM Auto-Scaling & Security Configuration

Submitted By

Abhilash Agarwal
M23CSA502

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

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.

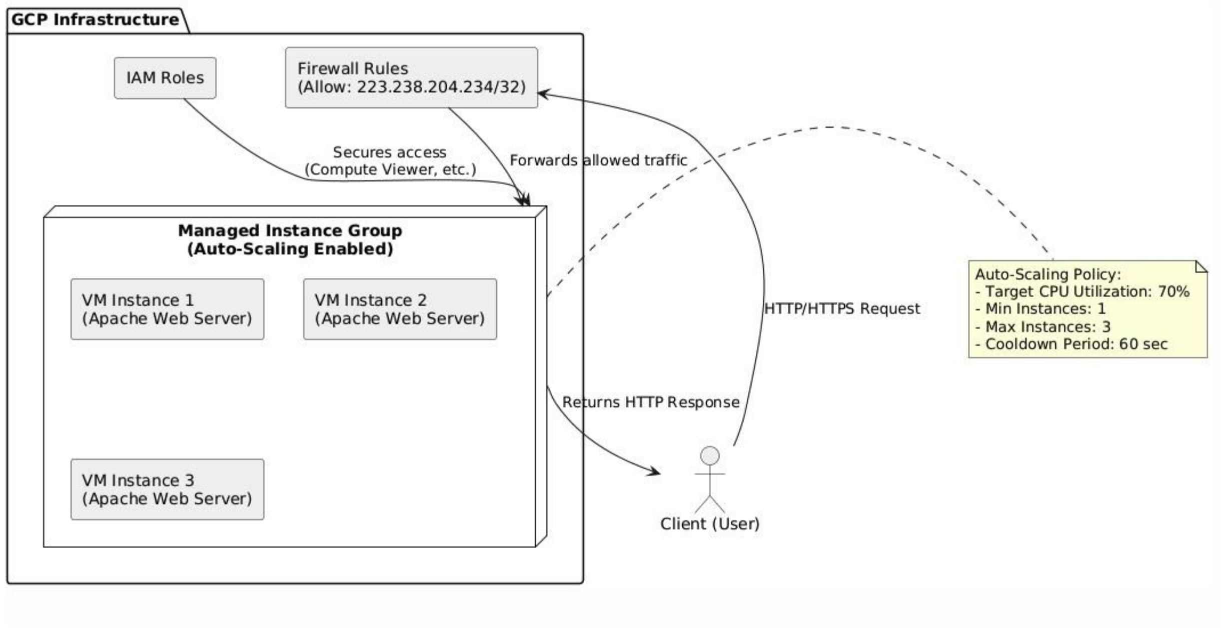


Figure 1: System Architecture Diagram of GCP VM Auto-Scaling & Security Implementation

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 (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.

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 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.

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

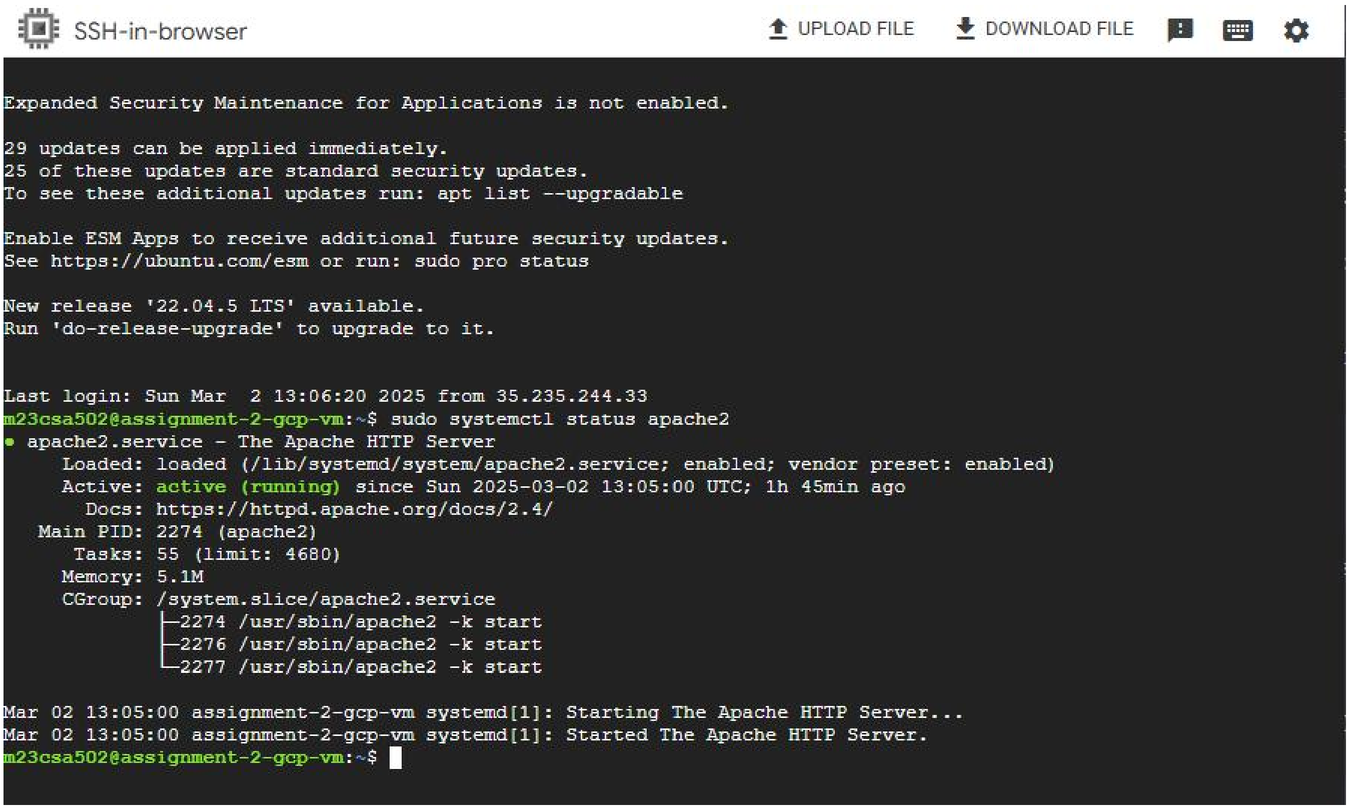
To create a VM instance with Apache installed, we use the following script:

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

# Set your project ID and zone
PROJECT_ID="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."
```

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



```
Expanded Security Maintenance for Applications is not enabled.

29 updates can be applied immediately.
25 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sun Mar  2 13:06:20 2025 from 35.235.244.33
m23csa502@assignment-2-gcp-vm:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2025-03-02 13:05:00 UTC; 1h 45min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 2274 (apache2)
      Tasks: 55 (limit: 4680)
     Memory: 5.1M
     CGroup: /system.slice/apache2.service
             ├─2274 /usr/sbin/apache2 -k start
             ├─2276 /usr/sbin/apache2 -k start
             └─2277 /usr/sbin/apache2 -k start

Mar 02 13:05:00 assignment-2-gcp-vm systemd[1]: Starting The Apache HTTP Server...
Mar 02 13:05:00 assignment-2-gcp-vm systemd[1]: Started The Apache HTTP Server.
m23csa502@assignment-2-gcp-vm:~$ 
```

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

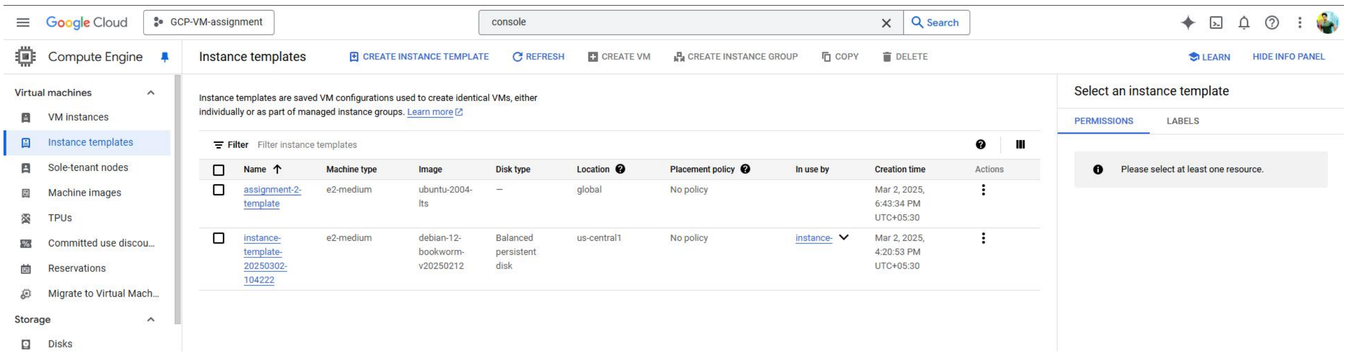
Step 2: Creating a Managed Instance Group with auto-scaling policies

4.2.1 Creating an Instance Template

```bash
#!/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  \
```

```
    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



Instance templates are saved VM configurations used to create identical VMs, either individually or as part of managed instance groups. Learn more

Name	Machine type	Image	Disk type	Location	Placement policy	In use by	Creation time	Actions
assignment-2-template	e2-medium	ubuntu-2004-lts	—	global	No policy		Mar 2, 2025, 6:43:34 PM UTC+05:30	⋮
instance-template-20250302-104222	e2-medium	debian-12-bookworm-v20250212	Balanced persistent disk	us-central1	No policy	instance-	Mar 2, 2025, 4:20:53 PM UTC+05:30	⋮

Please select at least one resource.

Figure 3: Created Instance Template in GCP Console

4.2.2 Creating a Managed Instance Group with Auto-Scaling

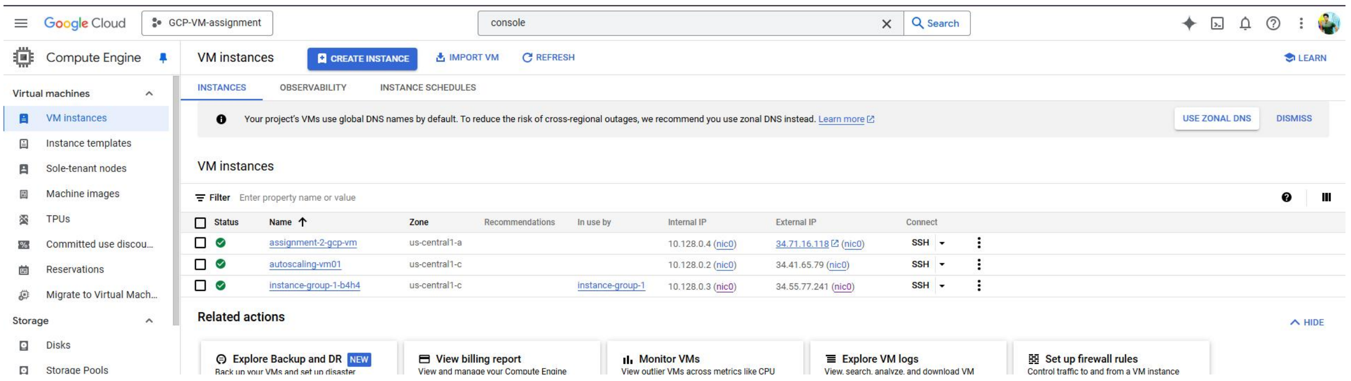
```
#!/bin/bash
# Create a Managed Instance Group (MIG) with auto-scaling

PROJECT_ID="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 60s
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.
    "
```

```
echo "Auto-scaling parameters:"
echo "  - Target CPU utilization: 70%"
echo "  - Min instances: 1"
echo "  - Max instances: 3"
echo "  - Cooldown period: 60 seconds"
```

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



Your project's VMs use global DNS names by default. To reduce the risk of cross-regional outages, we recommend you use zonal DNS instead. Learn more

VM instances

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	assignment-2-gcp-vm	us-central1-a			10.128.0.4 (nic0)	34.71.16.118 (nic0)	SSH
✓	autoscaling-vm01	us-central1-c			10.128.0.2 (nic0)	34.41.65.79 (nic0)	SSH
✓	instance-group-1-b4h4	us-central1-c		instance-group-1	10.128.0.3 (nic0)	34.55.77.241 (nic0)	SSH

Explore Backup and DR NEW
Back up your VMs and set up disaster

View billing report
View and manage your Compute Engine

Monitor VMs
View outlier VMs across metrics like CPU

Explore VM logs
View, search, analyze, and download VM

Set up firewall rules
Control traffic to and from a VM instance

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="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 \
  --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="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"
```

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:

```bash
#!/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

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

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

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:
```

```yaml
- 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

```yaml
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 70%, new instances were added within approximately 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

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/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

9 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.

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