# Automated VM Scaling and Security on GCP

Aryan Kumar M23CSA510

March 1, 2025

**GitHub Repository:** `https://github.com/Aryank47/gcp-autoscaling`

**Abstract**

This report outlines step-by-step instructions for creating a VM on Google Cloud Platform (GCP), configuring auto-scaling based on CPU utilization, and implementing security measures such as IAM roles and firewall rules. The architecture ensures scalability, security, and automated testing of the auto-scaling policy.

## Contents

# 1 Introduction

Google Cloud Platform (GCP) is a robust cloud infrastructure for deploying scalable applications. This document demonstrates how to set up a VM instances with auto-scaling and security policies.

# 2 Step-by-Step Implementation

## 2.1 Step 1: Create a VM Instance on GCP

## 2.2 config

```
ZONE=asia-south1-c
MACHINE_TYPE=e2-micro
TEMPLATE_NAME=test-template
MIG_NAME=test-mig
TARGET_CPU=0.7
MAX_INSTANCES=3
LOAD_DURATION=300
```

## 2.3 Authenticate and Initialize the CLI

```
gcloud auth login
gcloud config set project [PROJECT_ID]
gcloud config set compute/zone [ZONE]
```

## 2.4 Create a VM Instance

```
gcloud compute instance-templates create $TEMPLATE_NAME \
        --machine-type=$MACHINE_TYPE \
        --image-family=debian-11 \
        --image-project=debian-cloud \
        --tags=allow-ssh,allow-http \
        --metadata=startup-script='
#!/bin/bash
apt-update
apt-install-y-stress-ng-python3
(
cat-<<-EOF
import-os

def-generate_stress(duration=300):
----command-=-f"stress-ng-----cpu-2-----timeout-{duration}s"
----os.system(command)

if-__name__-==-"__main__":
----generate_stress()
EOF
)->-/usr/local/bin/load_generator.py
chmod-+x-/usr/local/bin/load_generator.py
--------'
```

## 2.5 Verify the Instance

```
gcloud compute instances list
```

### 2.6 Step 2: Configure Auto-Scaling Policy

### 2.7 Create a Managed Instance Group (MIG)

```
gcloud compute instance-groups managed create $MIG_NAME \
        --template=$TEMPLATE_NAME \
        --base-instance-name=test-vm \
        --zone=$ZONE \
        --size=1
```

### 2.8 Set Auto-Scaling Policy

```
gcloud compute instance-groups managed set-autoscaling $MIG_NAME \
        --max-num-replicas=$MAX_INSTANCES \
        --min-num-replicas=1 \
        --target-cpu-utilization=$TARGET_CPU \
        --zone=$ZONE
```

### 2.9 Verify Auto-Scaling Configuration

```
gcloud compute instance-groups managed describe my-mig --zone=[ZONE]
```

### 2.10 Step 3: Implement Security Measures

### 2.11 Create Firewall Rules

```
gcloud compute firewall-rules create allow-ssh \
  --allow tcp:22 \
  --target-tags=allow-ssh

gcloud compute firewall-rules create allow-http \
  --allow tcp:80 \
  --target-tags=allow-http

gcloud compute firewall-rules create deny-all \
  --deny all \
  --priority=65534
```

### 2.12 Configure IAM Roles

```
gcloud iam service-accounts create compute-reader --display-name="Compute Reader"
gcloud projects add-iam-policy-binding [PROJECT_ID] \
  --member="serviceAccount:compute-reader@[PROJECT_ID].iam.gserviceaccount.com" \
  --role="roles/compute.instanceViewer"
```

### 2.13 Verify IAM Policy

```
gcloud projects get-iam-policy [PROJECT_ID] --flatten="bindings[].members"
```

### 2.14 Step 4: Test Auto-Scaling

### 2.15 Generate CPU Load

```
gcloud compute ssh my-vm --zone=[ZONE] --command="stress-ng --cpu 2 --timeout 300s"
```

### 2.16 Monitor Scaling Activity

```
gcloud compute instance-groups managed list-instances my-mig --zone=[ZONE]
```

## 3 Architecture Diagram

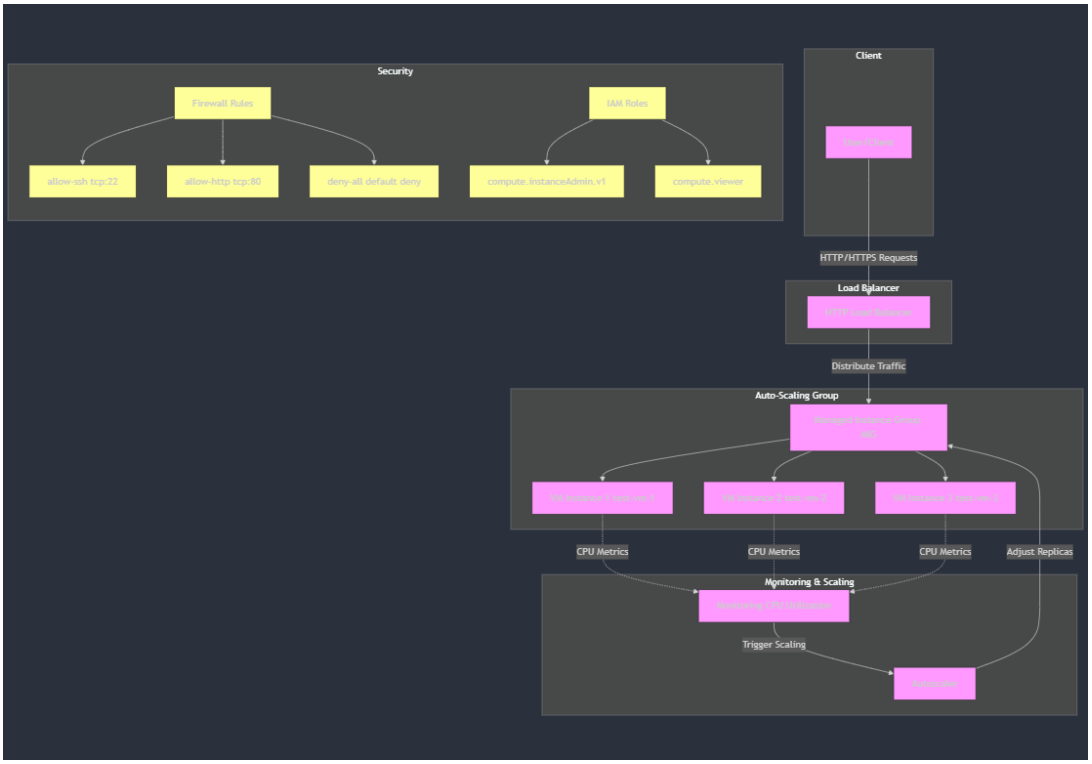


Figure 1: **GCP Auto-Scaling Architecture**

# 4 CPU Utilization Metrics

## 4.1 Initial Load Generation

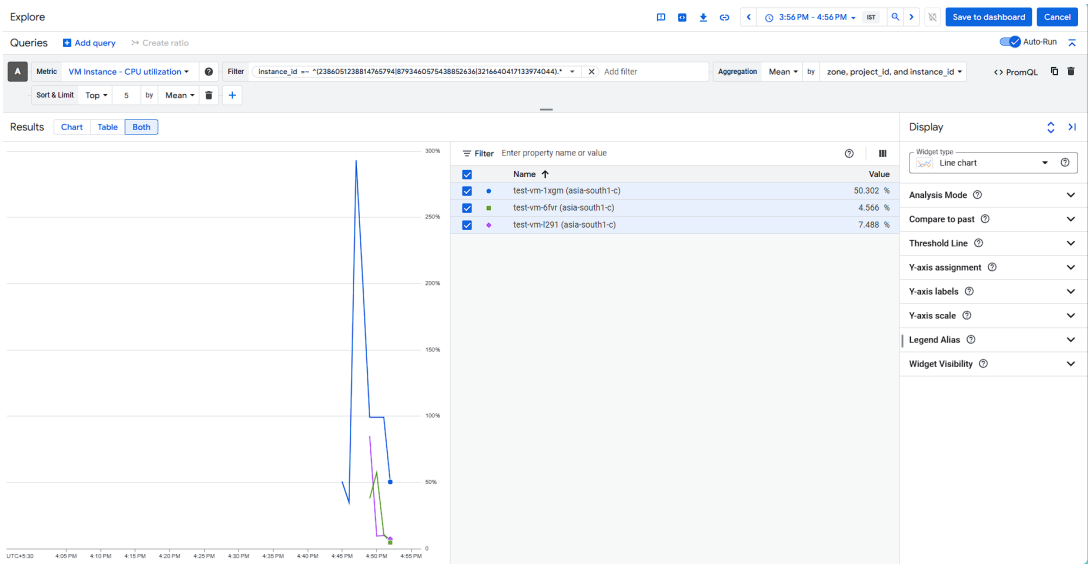


Figure 2: Initial CPU utilization spike during load generation

**Description:** This graph shows the CPU utilization of the VM instances during the initial load generation phase. The blue line represents the primary instance experiencing a significant spike in CPU usage, triggering the auto-scaling policy.

## 4.2 Auto-Scaling Response

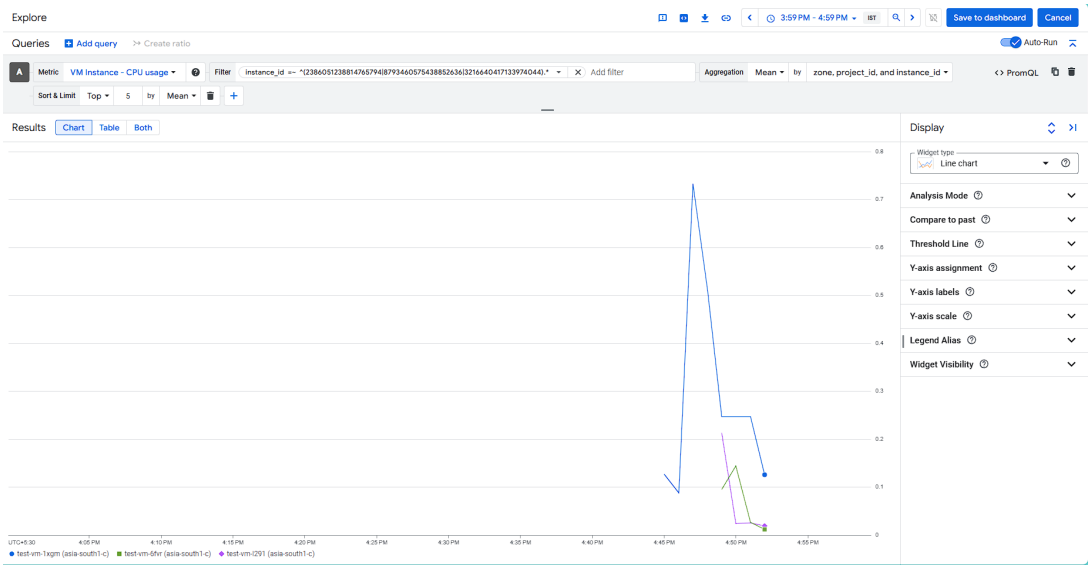


Figure 3: Auto-scaling response and load distribution

**Description:** This graph illustrates the auto-scaling response where additional instances are created to handle the load. The different colored lines represent multiple VM instances sharing the load, demonstrating the effectiveness of the auto-scaling configuration.

# 5 Best Practices

- Use the default VPC network for simplicity.
- Keep zones consistent across resources.
- Delete resources after testing to avoid costs.

# 6 Conclusion

This implementation provides a scalable and secure VM setup on GCP. The auto-scaling policy ensures resource optimization, while IAM and firewall rules enhance security. Always validate your setup by triggering load and monitoring the environment.