Project Report: Virtual Machine Setup with Auto-Scaling

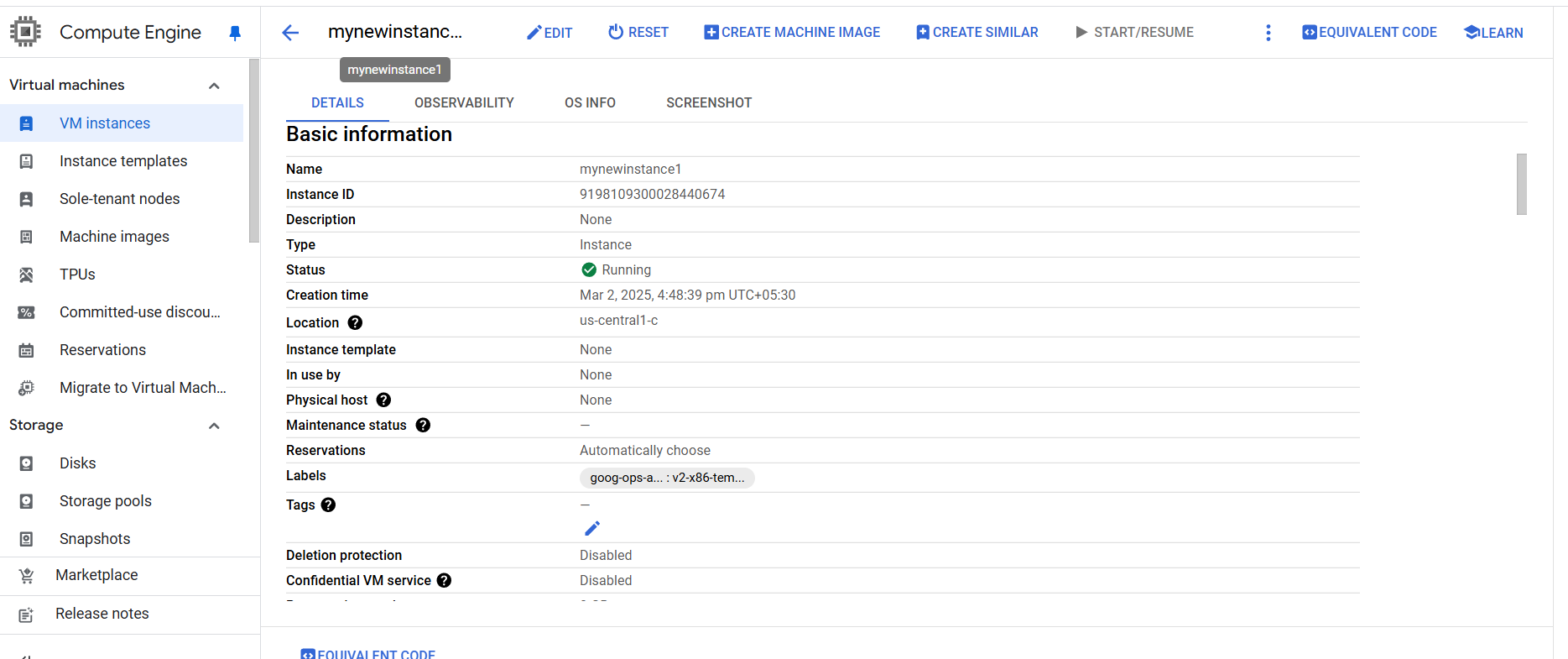
and Security in GCP (Assignment- 2)

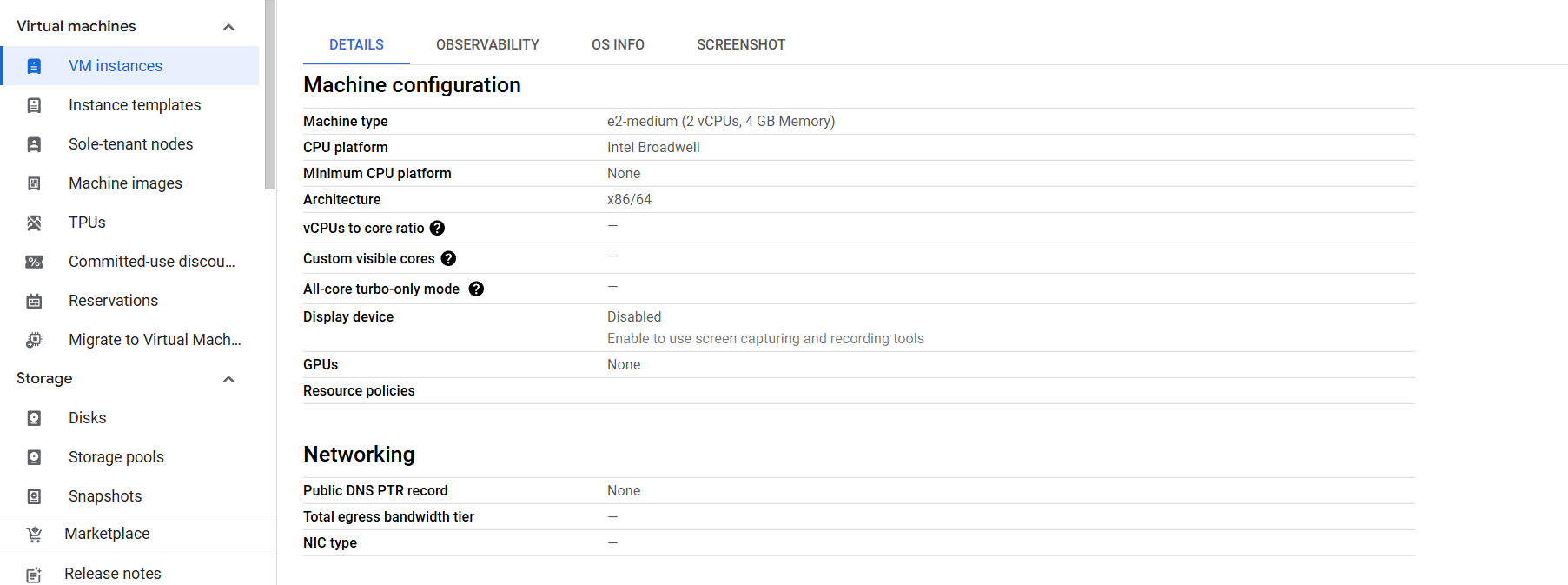
Objective  
This project aims to deploy a virtual machine (VM) on Google Cloud Platform (GCP), configure auto-scaling to handle varying workloads, and establish security protocols like firewall rules and IAM roles to maintain controlled and secure access.

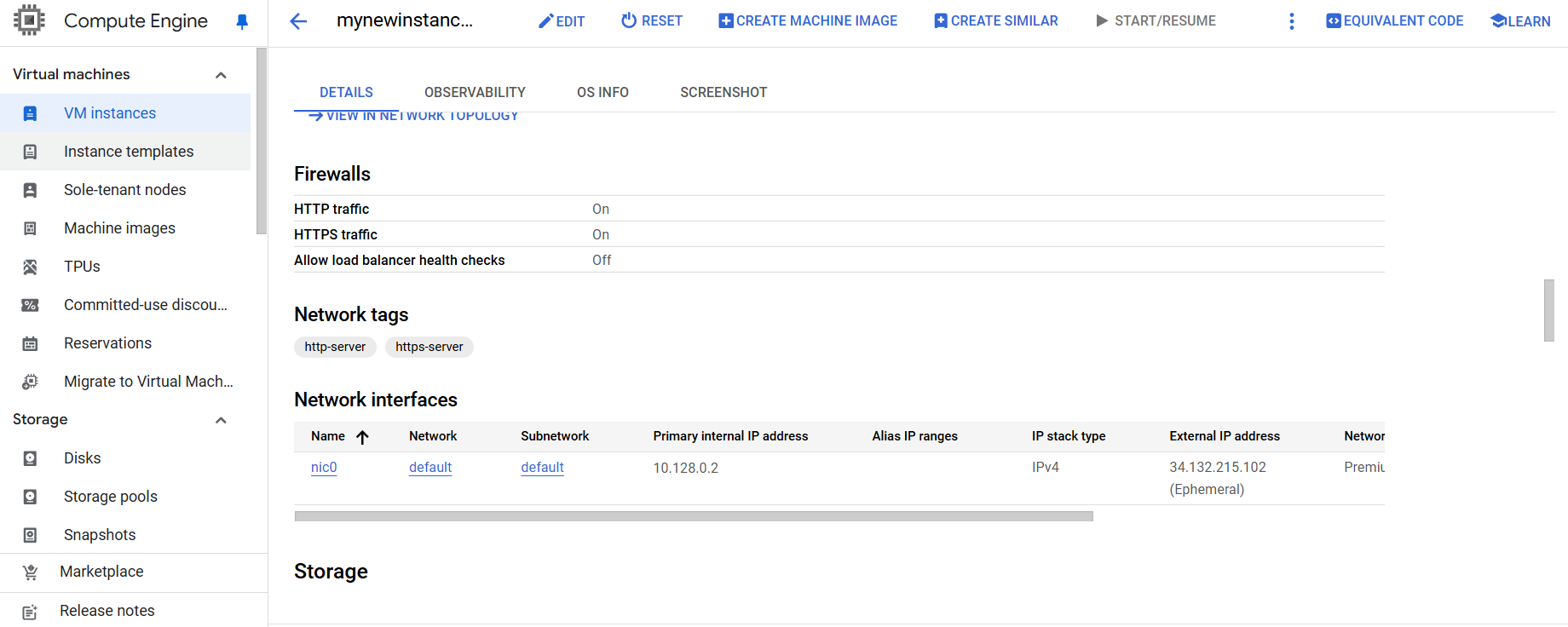
# 1. Step-by-Step Instructions for Implementation

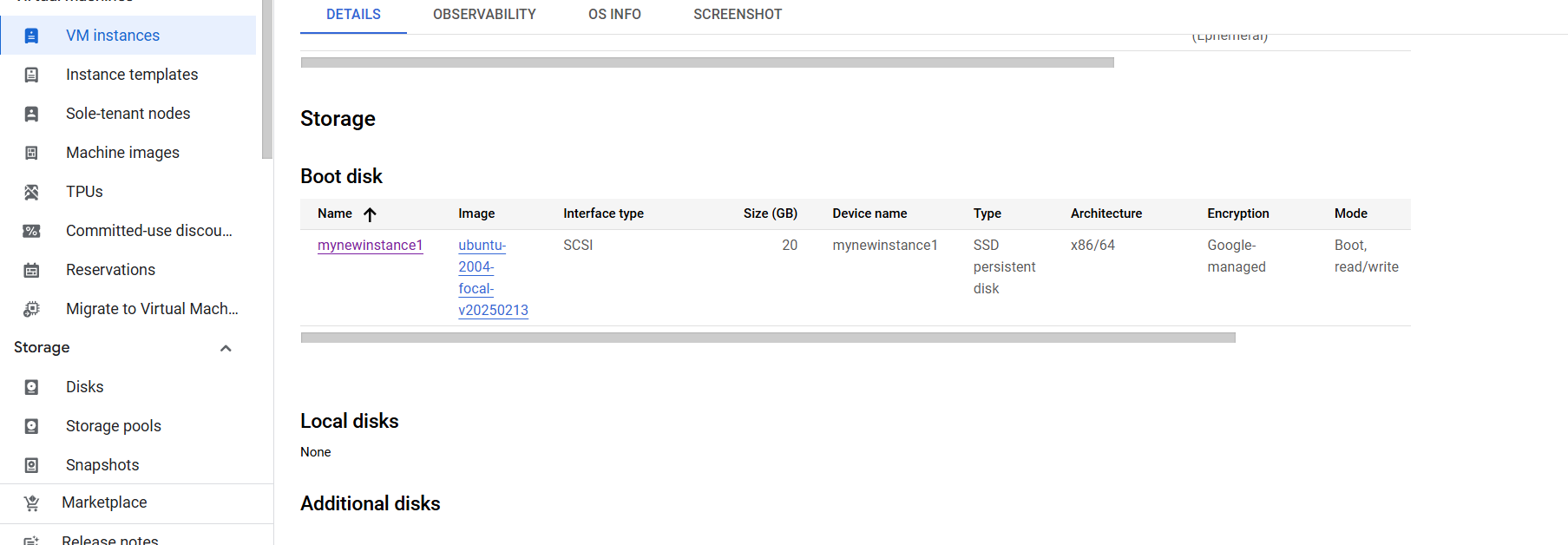
## 1.1 Creation of a VM Instance on GCP:

1. **Sign in to Google Cloud Console:**
   * Navigate to the [Google Cloud Console](https://console.cloud.google.com).
2. **Go to Compute Engine:**
   * In the sidebar, go to **Compute Engine** > **VM instances**.
3. **Create a New VM:**
   * Click on **Create Instance**.
   * Set a **name** for the instance.
   * Choose the **region** and **zone** where the instance will be located.
   * Choose a **machine type** (e.g., e2-medium).
   * Select an **operating system** (e.g., Ubuntu 20.04 LTS).
4. **Configure Boot Disk:**
   * Adjust the boot disk to meet your requirements (size, type, and OS).
5. **Configure Networking (Optional):**
   * Under the **Networking** tab, set the **VPC network** and subnet settings.
6. **Create the Instance:**
   * Click on **Create** to provision the VM.



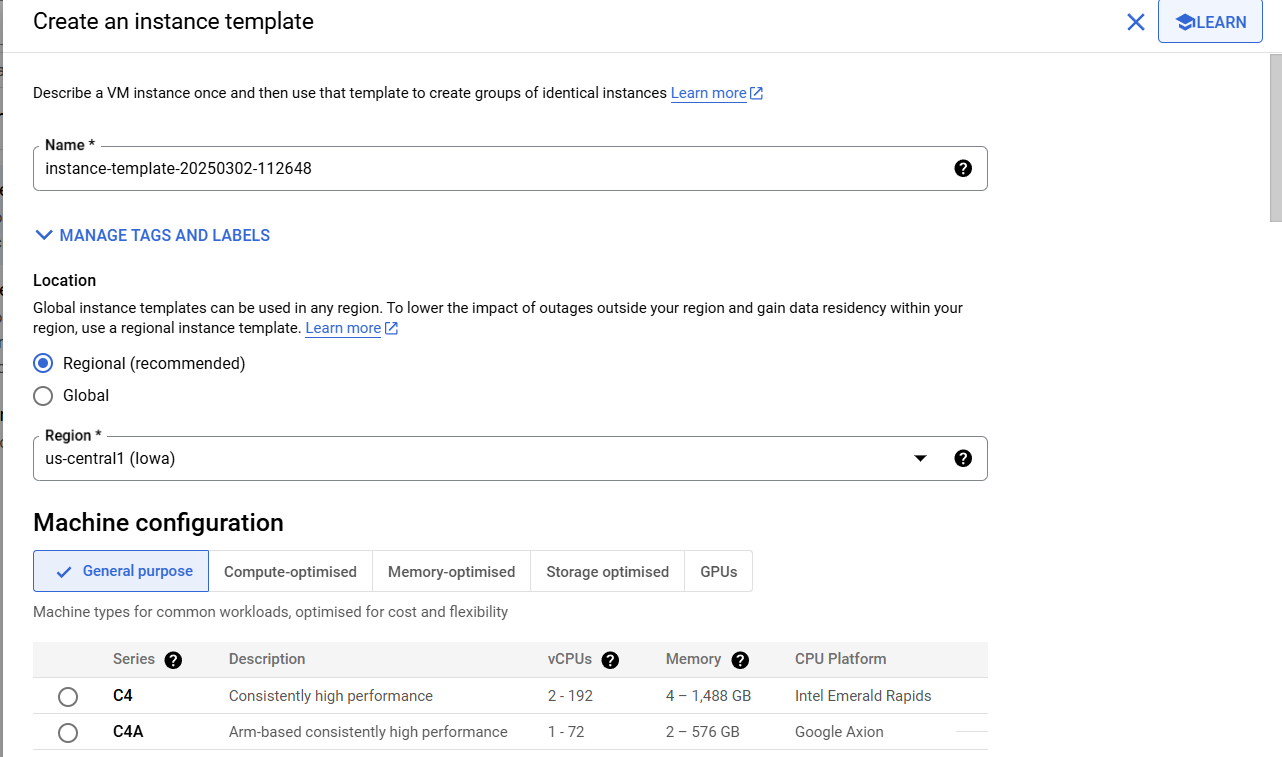


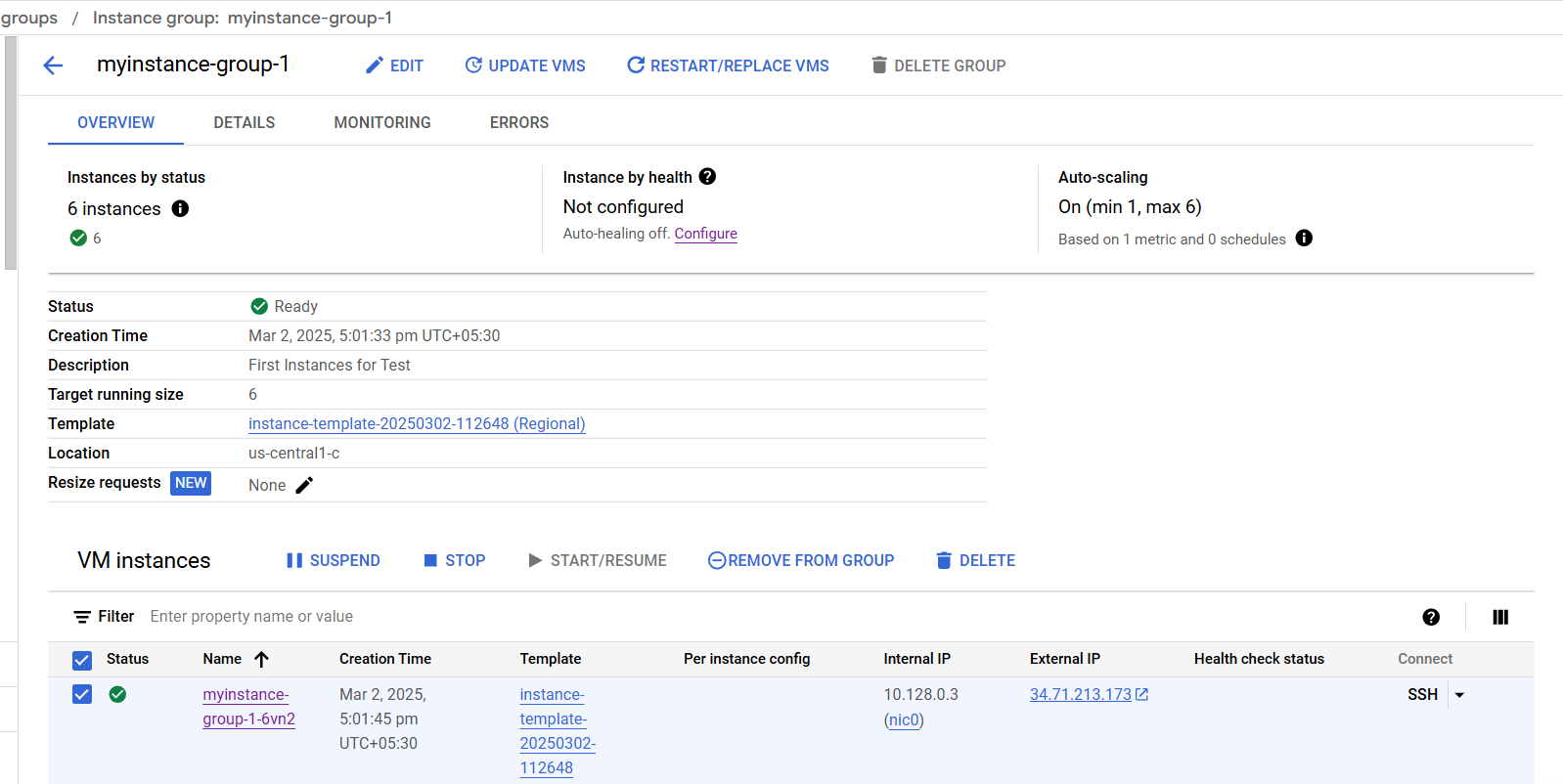


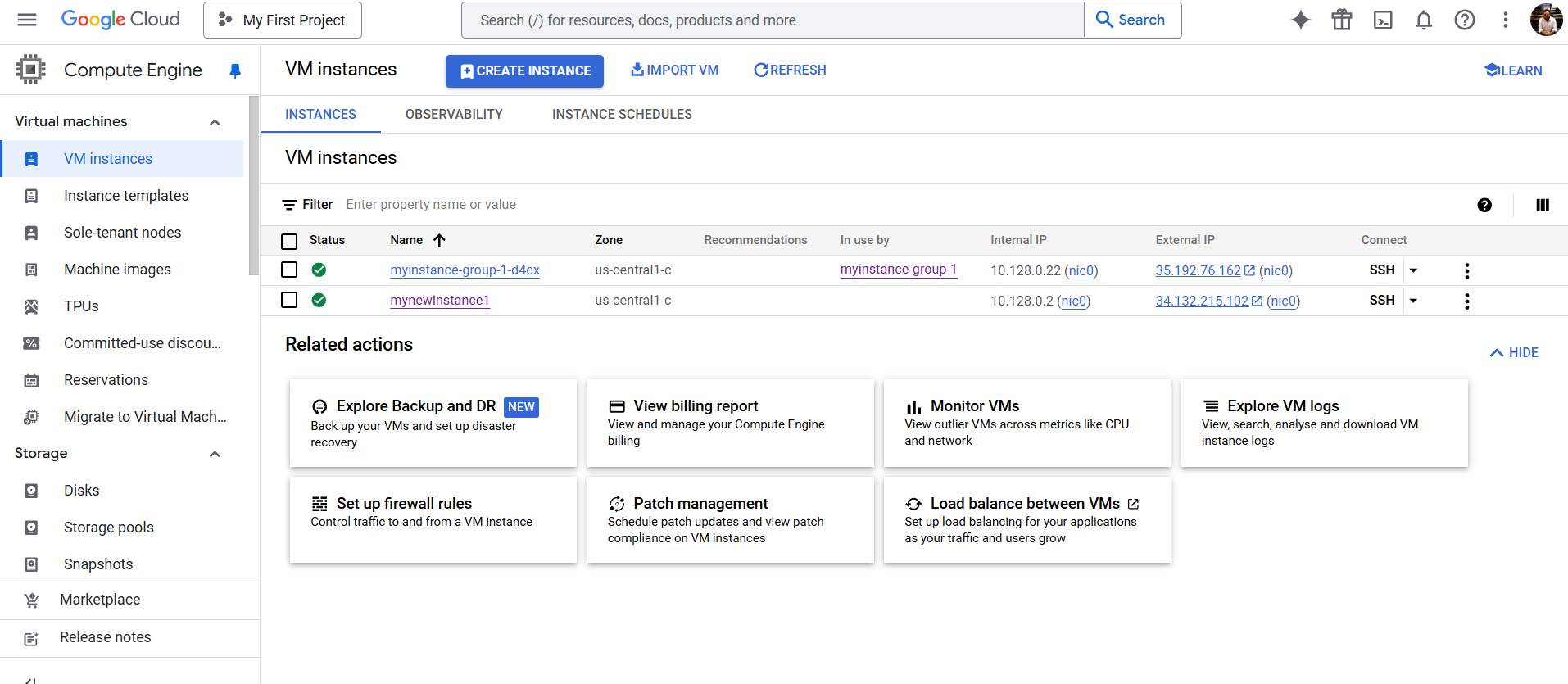


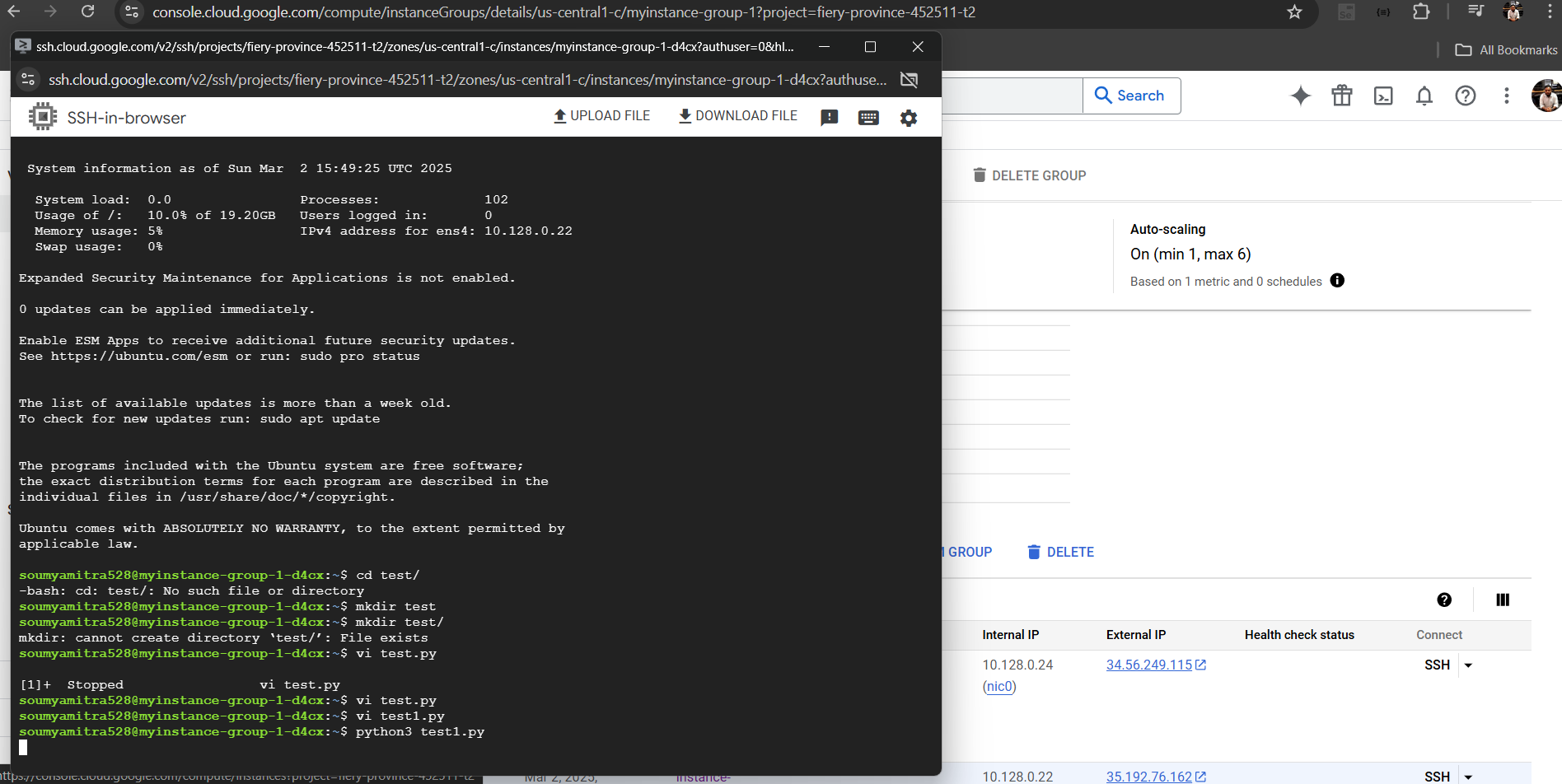
## 1.2 Configuration of Auto-scaling Policies:

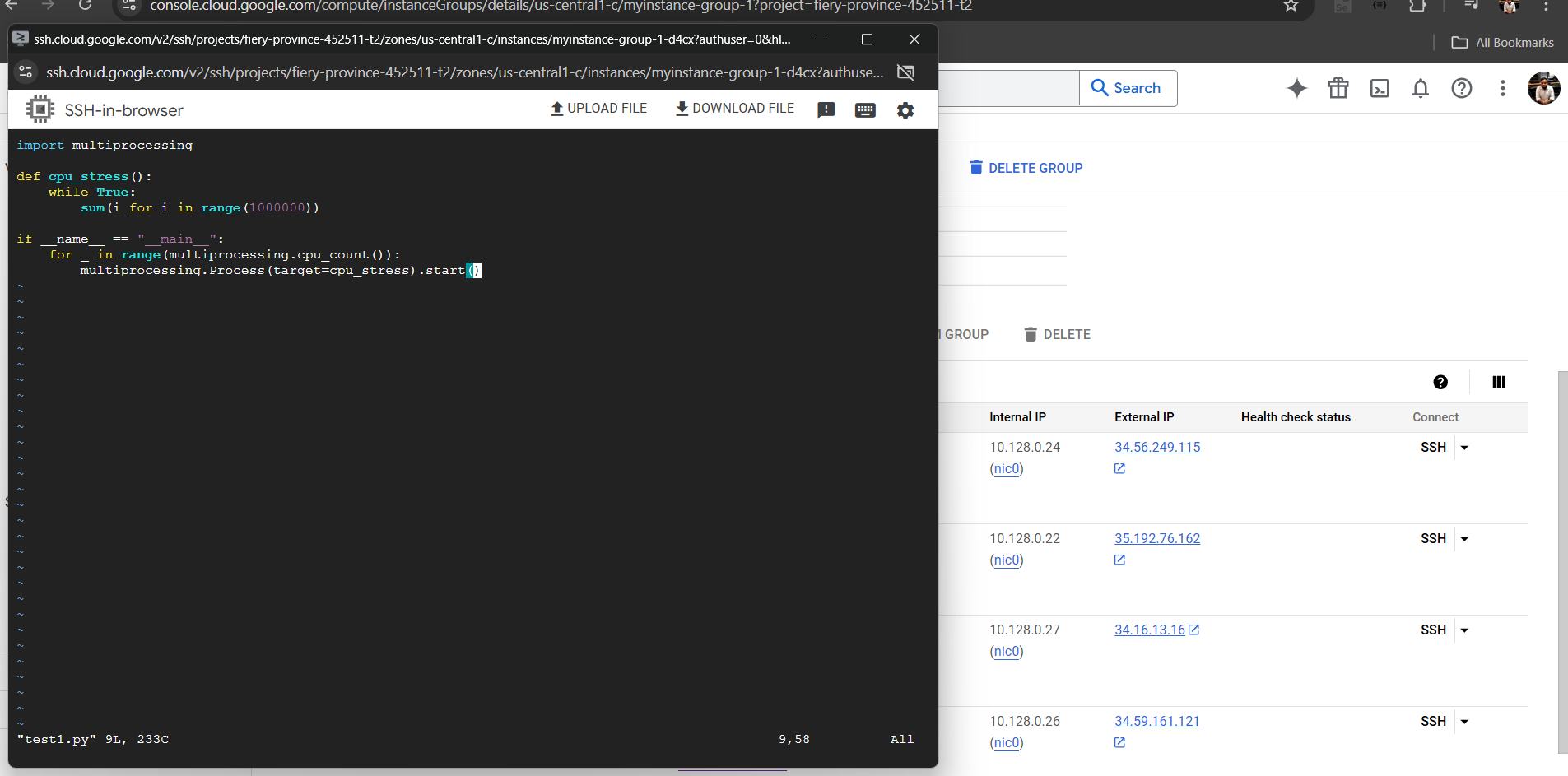
1. **Create an Instance Group:**
   * Navigate to **Compute Engine** > **Instance Groups**.
   * Click on **Create Instance Group**.
   * Select the **Managed Instance Group** option.
   * Specify the **instance template** (you can create one if not already available).
2. **Configure Auto-scaling:**
   * In the auto-scaling section, set the following:
     + **Metric Type:** CPU utilization.
     + **Target CPU Utilization:** Define the percentage of CPU usage that will trigger scaling (here 40% such that VM instances creation is easier)
     + **Minimum Instances:** Set the minimum number of instances.
     + **Maximum Instances:** Set the maximum number of instances to scale to.
3. **Save the Auto-scaling Policy:**
   * Click **Create** to enable auto-scaling based on the defined CPU utilization policy.

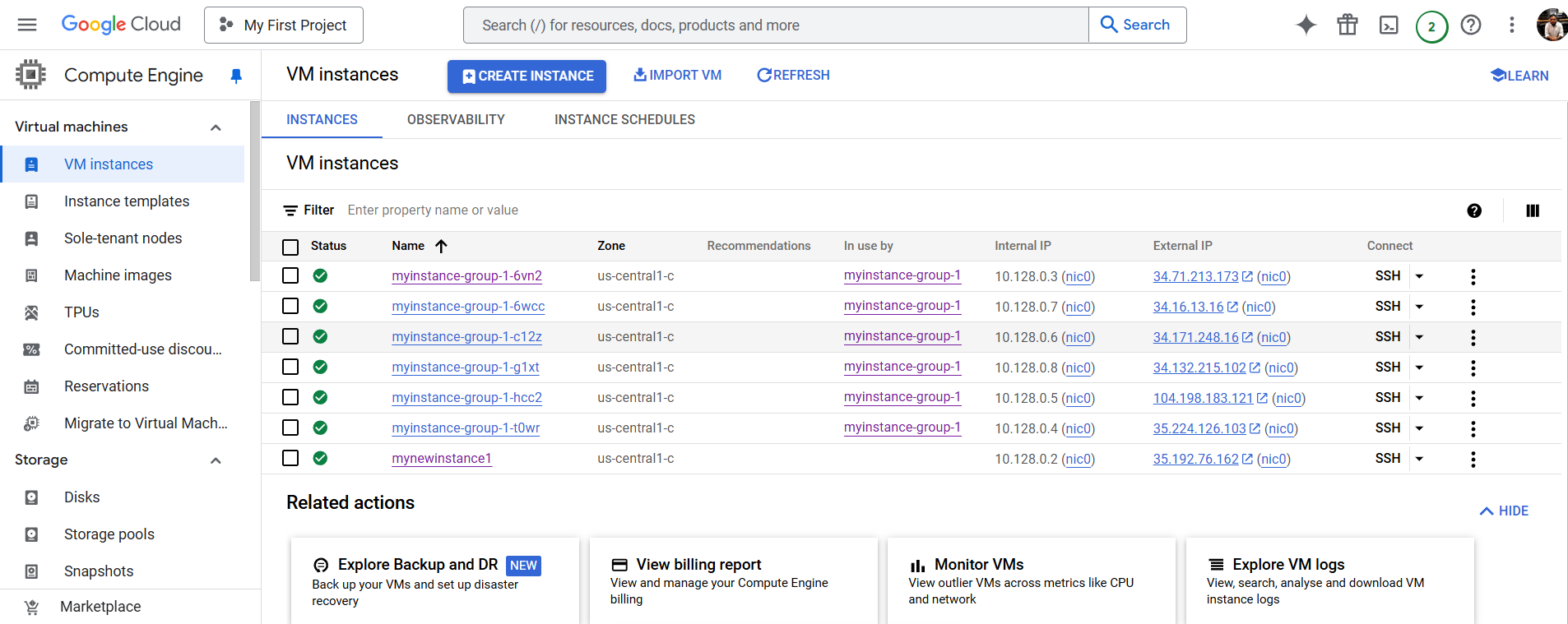




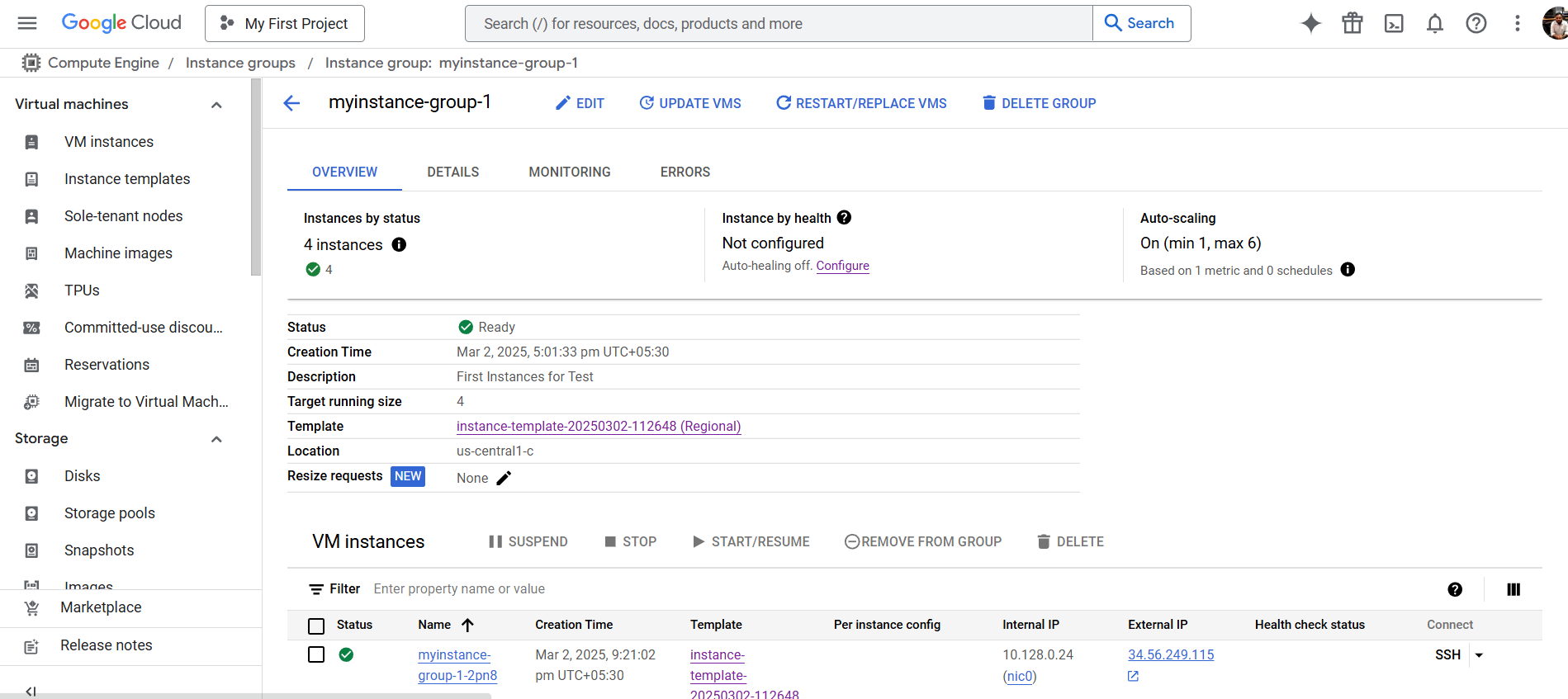


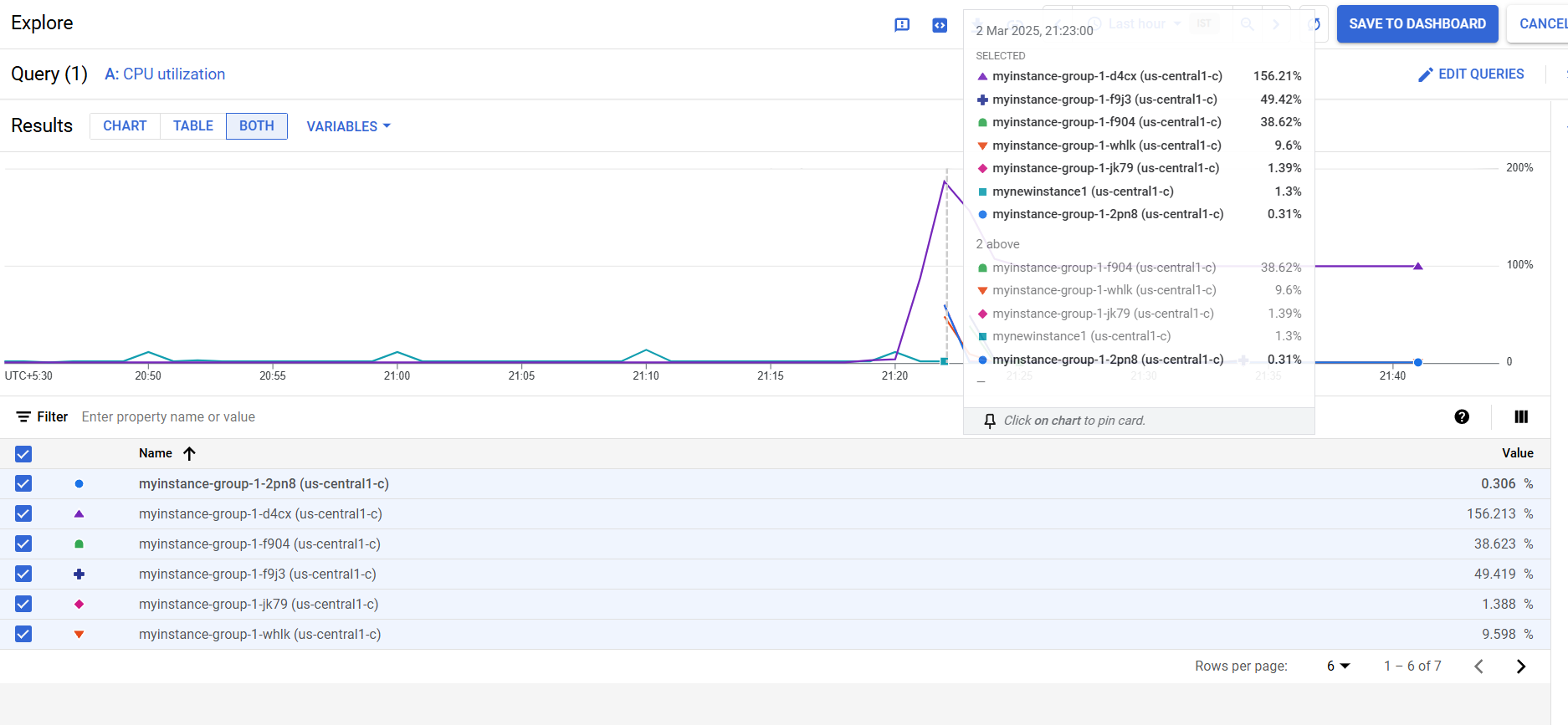






Stress Testing for Autoscaling (Number of instances reduced to 4):

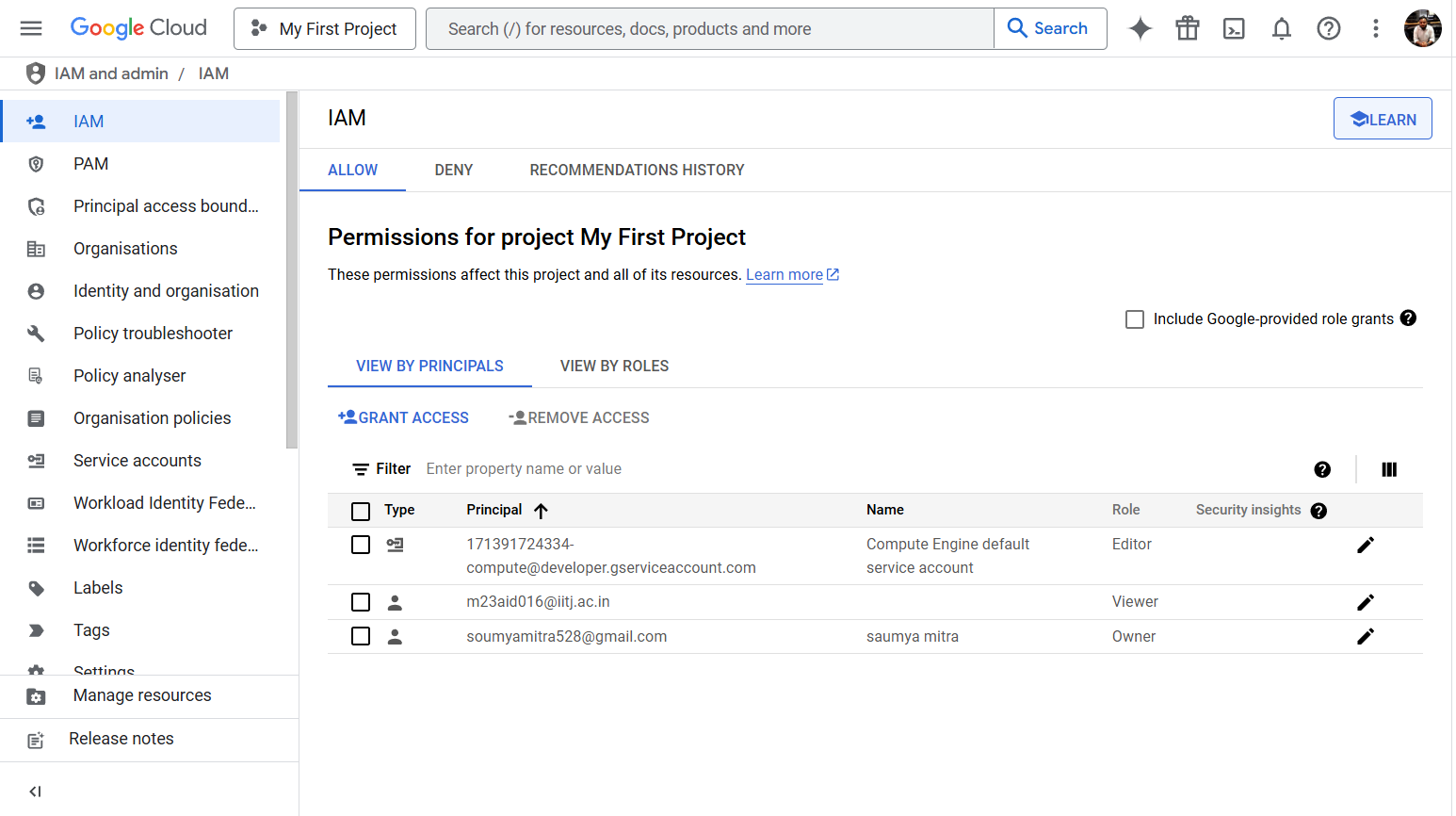




## 1.3 Implementation of Security Measures:

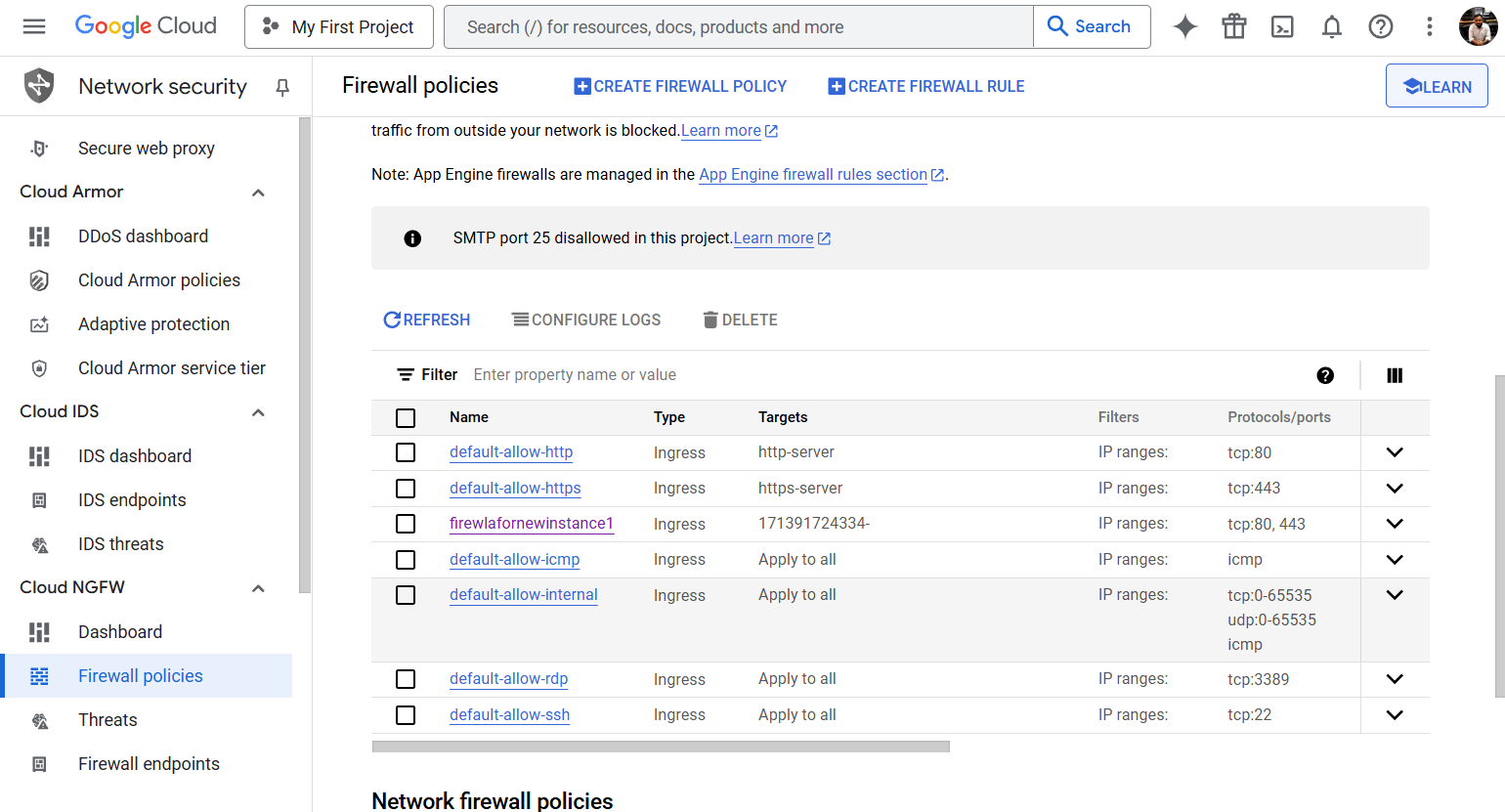
**1.3.1 Setting up IAM Roles:**

1. **Go to IAM & Admin:**
   * Navigate to **IAM & Admin** in the GCP console.
2. **Assign Roles:**
   * Select the user or service account for which you want to assign roles.
   * Click **Edit** and assign appropriate **roles** (e.g., Compute Engine Admin, Viewer).
   * Limit access by assigning only necessary roles for the task.



## 1.3.2 Configuring Firewall Rules:

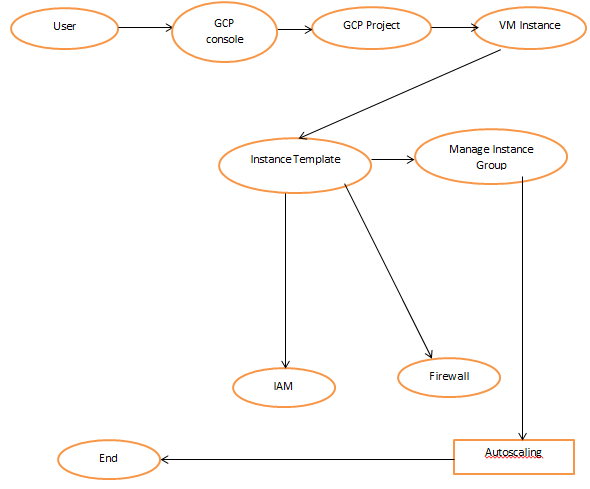
1. **Go to VPC Networks:**
   * Navigate to **VPC Networks** > **Firewall rules**.
2. **Create a New Firewall Rule:**
   * Click **Create Firewall Rule**.
   * Set a **name** for the rule.
   * Select the **network** the rule applies to.
   * Specify the **direction** (inbound or outbound).
3. **Allow/Deny Traffic:**
   * Set **source IP ranges**, **protocols**, and **ports** to control traffic (e.g., allow only traffic on port 80 for HTTP).
4. **Create the Rule:**
   * Click **Create** to apply the firewall rule.



# 2. Architecture Design

Below is the architectural overview of the VM setup, auto-scaling configuration, and security policies applied:

* **VM Instances** are created within a **Managed Instance Group**.
* **Auto-scaling** adjusts the number of instances based on CPU utilization.
* **IAM roles** provide restricted access to specific users or service accounts.
* **Firewall rules** regulate traffic coming into and going out of the VM instances, controlling access based on IP ranges, ports, and protocols.



# Recording Link:

<https://drive.google.com/file/d/162Xisoeeiivb3sib581CKHBpcrrqzf3d/view?usp=sharing>

# Code:

import multiprocessing

def cpu\_stress():

while True:

sum(i for i in range(1000000))

if \_\_name\_\_ == "\_\_main\_\_":

for \_ in range(multiprocessing.cpu\_count()):

multiprocessing.Process(target=cpu\_stress).start()