# Project Report: Virtual Machine Setup with Auto-Scaling and Security in GCP

---

## Objective

The objective of this project is to set up a virtual machine (VM) in Google Cloud Platform (GCP), implement auto-scaling policies based on workload, and configure security measures such as firewall rules and IAM roles to ensure controlled access.

---

## 1. Step-by-Step Instructions for Implementation

This section provides a structured approach to setting up a Virtual Machine, enabling auto-scaling, and configuring security measures in Google Cloud Platform (GCP). By following these steps, users can deploy a scalable and secure cloud-based infrastructure efficiently. The instructions ensure that the system is both cost-effective and highly available.

This section provides a structured approach to setting up a Virtual Machine, enabling auto-scaling, and configuring security measures in Google Cloud Platform (GCP). It ensures an efficient cloud-based infrastructure setup with optimal performance and security.q

### 1.1 Log in to Google Cloud Console

Before provisioning resources, users must log in to the Google Cloud Console, which serves as the interface for managing cloud services. This step is crucial to ensure access permissions, billing setup, and project management.

Before creating any resources in GCP, users must first log in to the Google Cloud Console. This step ensures access to cloud services and enables billing to use necessary computing resources.

1. Open Google Cloud Console.
2. Sign in with IITJ account with coupon provided in classroom

## 1.2 Create a GCP Project

A **GCP project** serves as an organizational container for cloud resources, allowing users to manage resources effectively while maintaining separate billing and security settings. Creating a dedicated project ensures better management and isolation of different cloud workloads.

A project in GCP acts as a container for cloud resources. Each project maintains separate billing, permissions, and configurations, providing organizational clarity.

## 1.3 Create a Virtual Machine (VM) Instance

A Virtual Machine (VM) is the core compute resource in GCP. This step guides users through selecting an appropriate **machine type, region, boot disk, and network configurations** to deploy an instance tailored to specific workloads.

A Virtual Machine (VM) is a fundamental computing resource in GCP. This step involves selecting an appropriate machine type, region, boot disk, and network settings to deploy a cloud-based VM.

1. Navigate to **Compute Engine > VM Instances**.
2. Click **Create Instance**.
3. Configure the instance:
    - **Name**: (e.g., my-vm-instance).
    - **Region & Zone**: Choose a location.
    - **Machine Type**: e2-medium (adjust based on need).
    - **Boot Disk**: Select **Ubuntu 22.04 LTS**.
    - **Firewall**: Check **Allow HTTP and HTTPS traffic**.
4. Click **Create**.

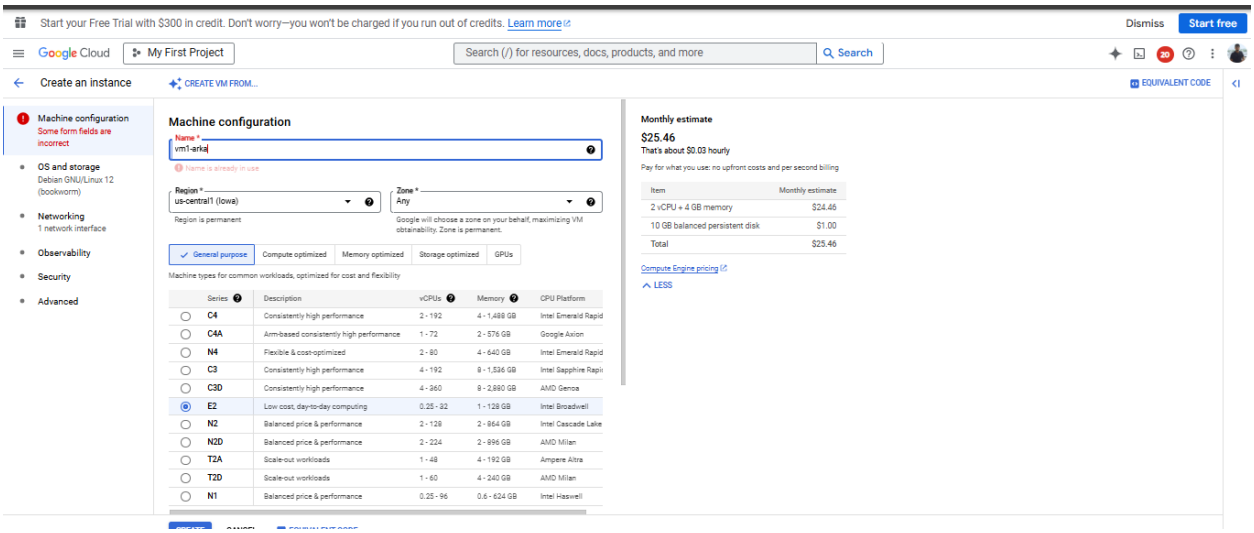



**Figure 1: VM creation**

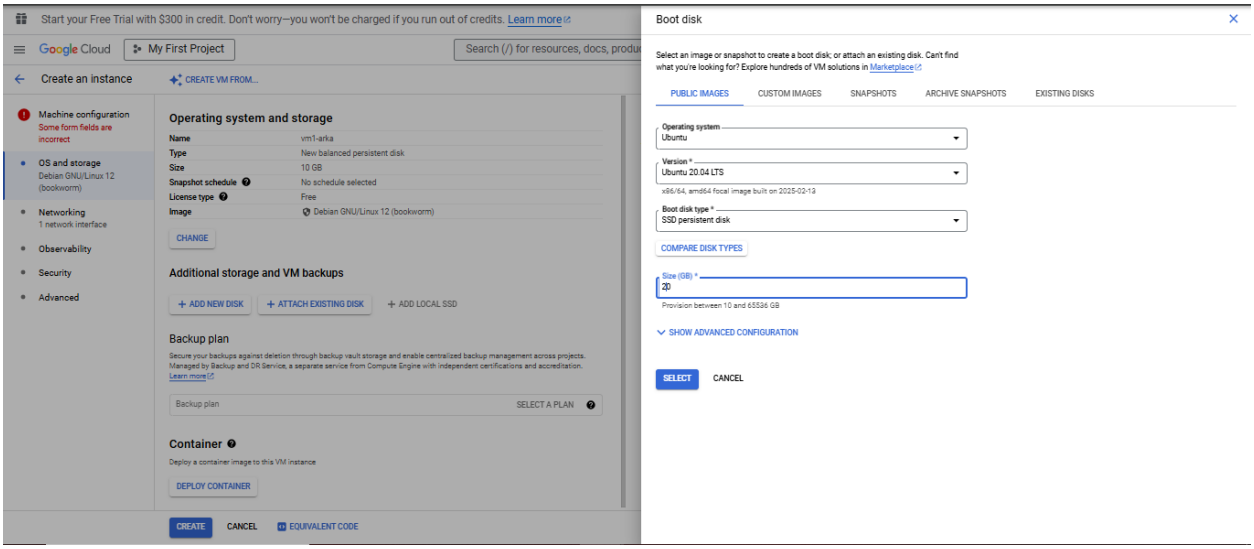



**Figure 2: VM OS setup**

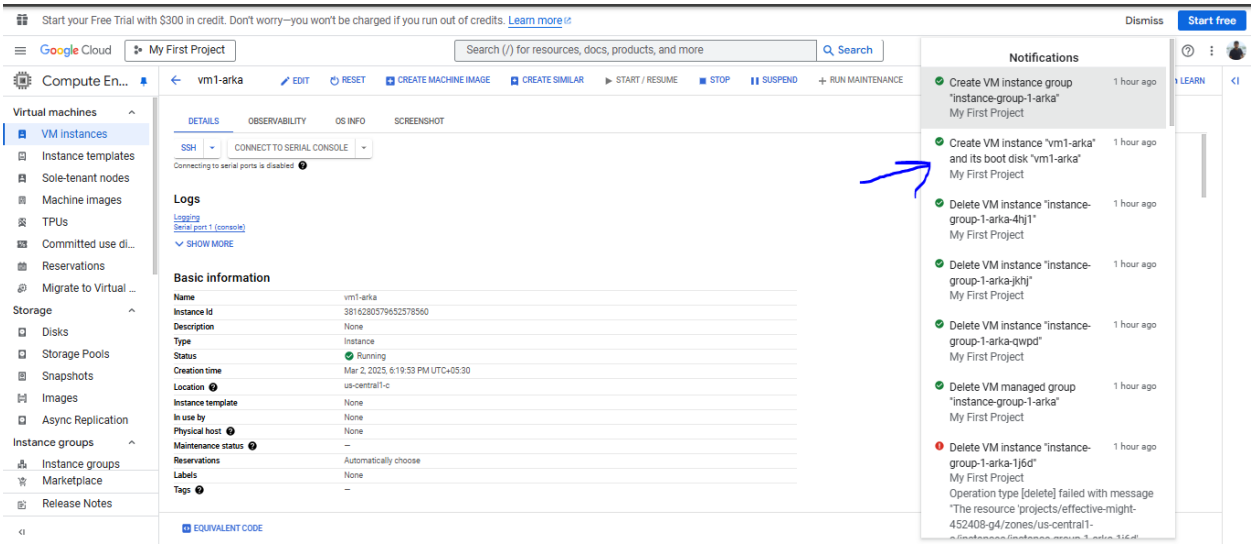



Figure 3: VM after creation

---

# 2. Configure Auto-Scaling

**Auto-scaling** is an essential feature in GCP that dynamically adjusts the number of instances based on workload. This ensures **cost efficiency, high availability, and optimal performance** by automatically scaling up during demand spikes and scaling down when demand decreases.

Auto-scaling in GCP dynamically adjusts the number of instances in response to workload changes. This ensures cost efficiency and high availability by scaling up during demand spikes and scaling down when demand decreases.

## 2.1 Create an Instance Template

An **Instance Template** acts as a predefined blueprint for creating VM instances within a **Managed Instance Group (MIG)**. By defining machine specifications, boot disk settings, and startup scripts, it ensures **consistency and ease of scaling** when new instances are added.

An instance template provides a blueprint for new virtual machines in a managed instance group. It defines machine specifications, boot disk settings, and startup scripts for uniform deployment.

1. Navigate to **Compute Engine > Instance Templates**.
2. Click **Create Instance Template**.
3. Configure it similarly to your existing VM.
4. Click **Create**.

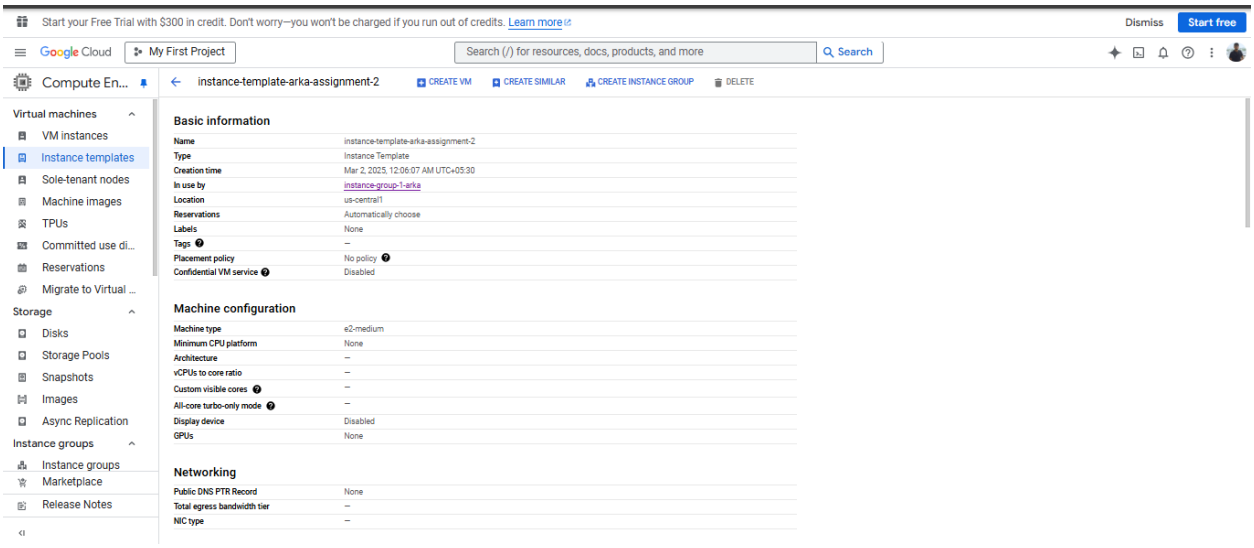



Figure 4: Instance template after creation

## 2.2 Create a Managed Instance Group

A **Managed Instance Group (MIG)** is responsible for maintaining identical instances across the cloud infrastructure. It ensures **fault tolerance, load balancing, and auto-scaling** based on resource utilization, thereby improving system reliability.

A Managed Instance Group (MIG) is responsible for maintaining identical VM instances. It automatically increases or decreases instances based on resource utilization, ensuring application reliability.

1. Navigate to **Compute Engine > Instance Groups**.

2. Click **Create Instance Group**.
3. Provide name & description
4. Select the **Instance Template** created earlier.
5. Set **Auto-scaling**:
    - Enable auto-scaling.
    - **CPU utilization target**: 40%.
    - **Minimum instances**: 1, **Maximum instances**: 10.
6. Click **Create**.

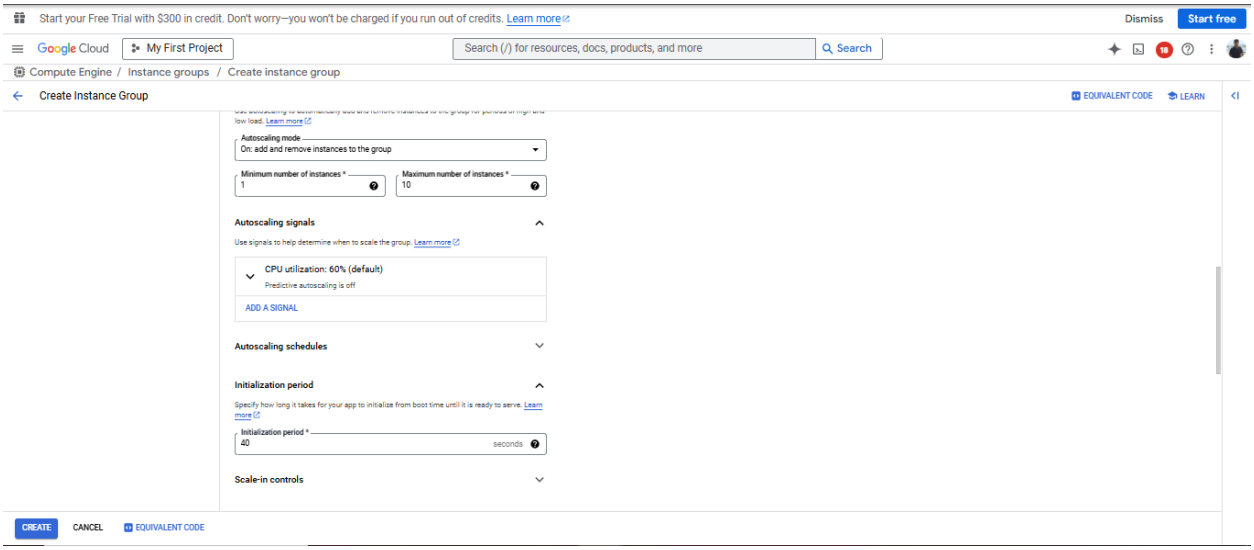



Figure 5: Instance group creation for autoscaling

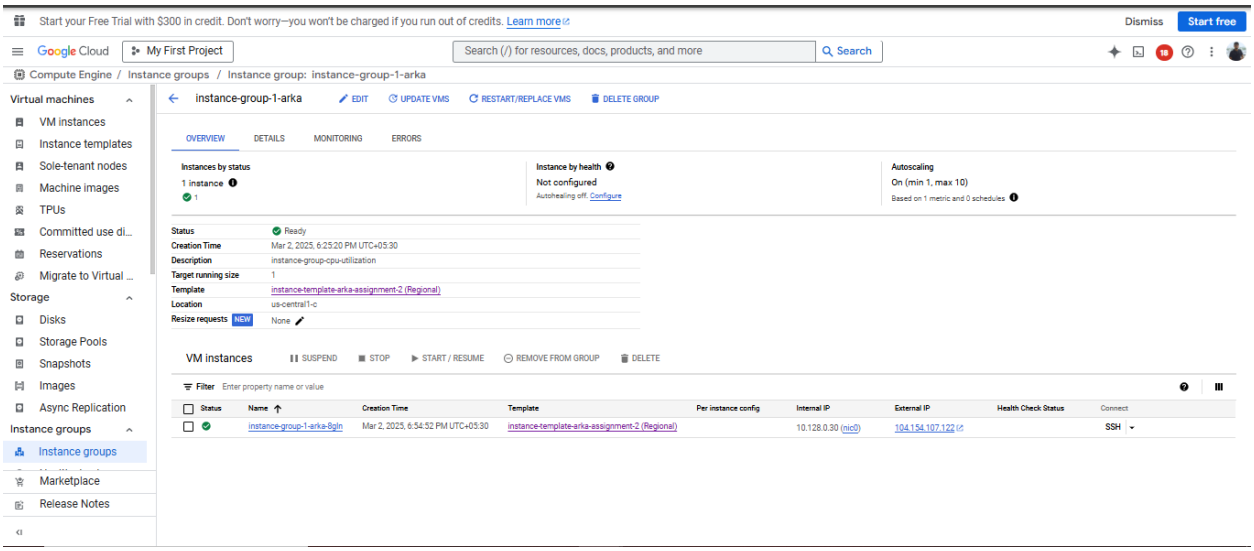



**Figure 6: Instance group after creation**

## 2.3 Stress Test for Auto-Scaling

A **stress test** simulates high CPU usage to validate the auto-scaling configuration. By running computationally intensive tasks, users can monitor the **scaling behavior of GCP**, ensuring that additional instances are created when necessary. This test is essential for verifying that the infrastructure can handle sudden spikes in workload without manual intervention.

A stress test simulates high CPU utilization to validate auto-scaling. By executing computationally intensive tasks, users can monitor whether GCP scales the infrastructure appropriately to handle workload spikes. To test if auto-scaling is working, run a CPU-intensive process on all cores:

```
import multiprocessing
import time

def check_prime(number):
```

```
    """Determine whether a given number is prime."""
    if number < 2:
        return False
    if number in (2, 3):
        return True
    if number % 2 == 0 or number % 3 == 0:
        return False
    divisor = 5
    while divisor * divisor <= number:
        if number % divisor == 0 or number % (divisor + 2) == 0:
            return False
        divisor += 6
    return True

def find_primes():
    """Continuously identify and display prime numbers."""
    candidate = 2
    while True:
        if check_prime(candidate):
            print(candidate, end=" ", flush=True)
        candidate += 1

def cpu_stress_test(core_count):
    """Initiate a CPU-intensive process using multiple cores."""
    workers = []
    for _ in range(core_count):
        process = multiprocessing.Process(target=find_primes)
        process.start()
        workers.append(process)

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
```

```
        for process in workers:
            process.terminate()
        print("¥nStress test halted.")

if __name__ == "__main__":
    available_cores = multiprocessing.cpu_count()  # Detect available CPU cores
    print(f"Initiating CPU stress test on {available_cores} cores...")
    cpu_stress_test(available_cores)
```

Run this script on an instance to generate high CPU usage, and check the **Instance Groups** section in the GCP Console to verify if additional instances are created automatically.

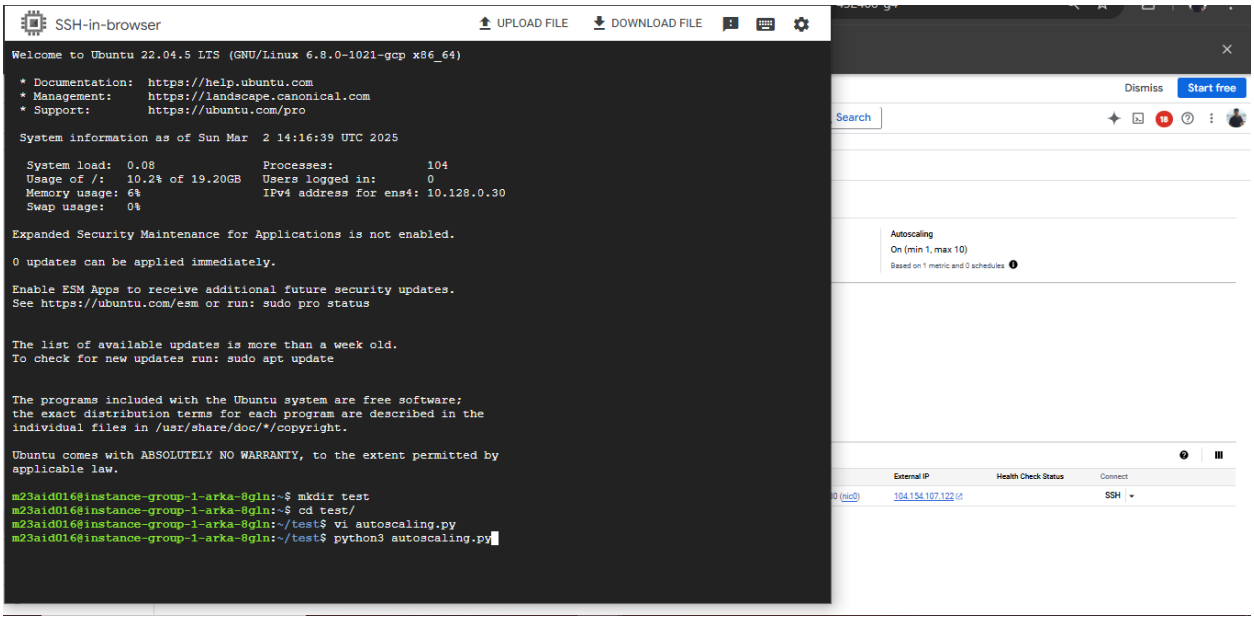



Figure 7: Executing program

## 3. Configure Security Measures

**Security** is a critical aspect of cloud computing. GCP offers robust security mechanisms such as **Identity and Access Management (IAM) roles and firewall rules** to restrict unauthorized access, protect data, and enforce least-privilege policies for users and services.

Security is crucial in cloud environments. GCP provides Identity and Access Management (IAM) roles and firewall rules to restrict unauthorized access and protect resources.

## 3.1 Set Up IAM Roles

**Identity and Access Management (IAM) roles** define user permissions for cloud resources. Implementing IAM ensures that only authorized personnel can access or modify cloud instances, reducing security risks and preventing unauthorized access.

IAM roles define user permissions in GCP. Assigning roles based on the principle of least privilege ensures that only authorized users can manage resources, reducing security risks.

1. Navigate to **IAM & Admin > IAM**.
2. Click **Add Member**.
3. Enter the **user's email ID**.
4. Assign a role:
    - **Compute Viewer**: Read-only access.
    - **Compute Admin**: Full control.
5. Click **Save**.

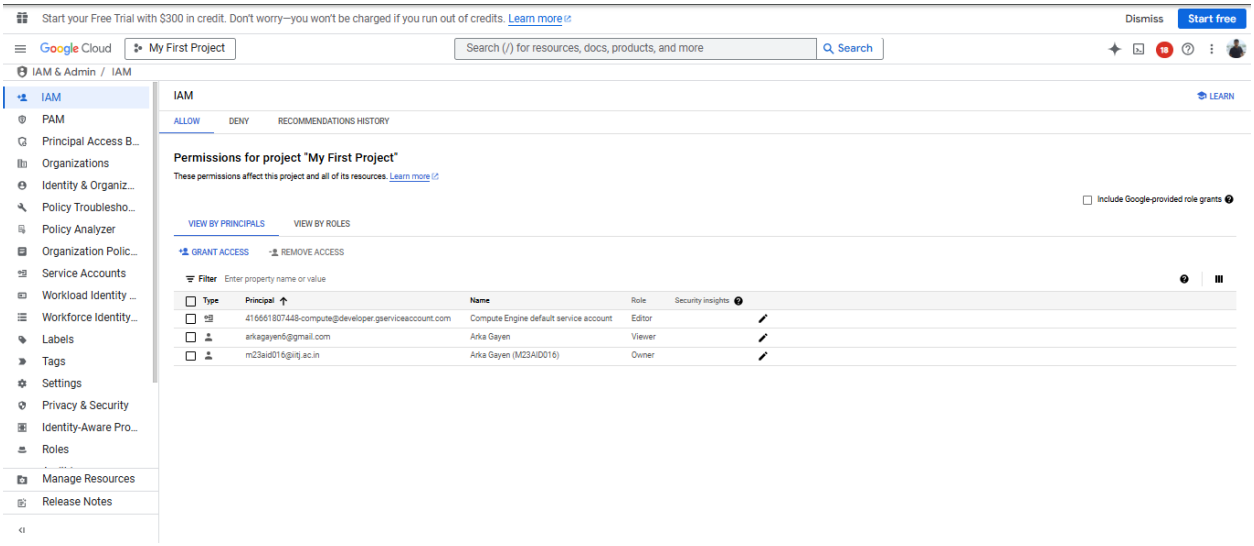



Figure 8: Setting IAM

## 3.2 Configure Firewall Rules

Firewall rules help **control inbound and outbound traffic** to VM instances. Properly configuring these rules ensures that only trusted sources can access cloud resources while blocking unauthorized or potentially harmful connections.

Firewall rules in GCP control network traffic by allowing or denying access to VM instances. This step involves defining ingress and egress rules to protect against unauthorized access and potential cyber threats.

1. Navigate to **VPC Network > Firewall**.
2. Click **Create Firewall Rule**.
3. Configure:
    - **Name**: allow-web.
    - **Direction**: Ingress.
    - **Action**: Allow.
    - **Source**: 0.0.0.0/0 (or restrict to specific IPs).
    - **Protocols & Ports**: TCP 80 (HTTP), 443 (HTTPS).
4. Click **Create**.

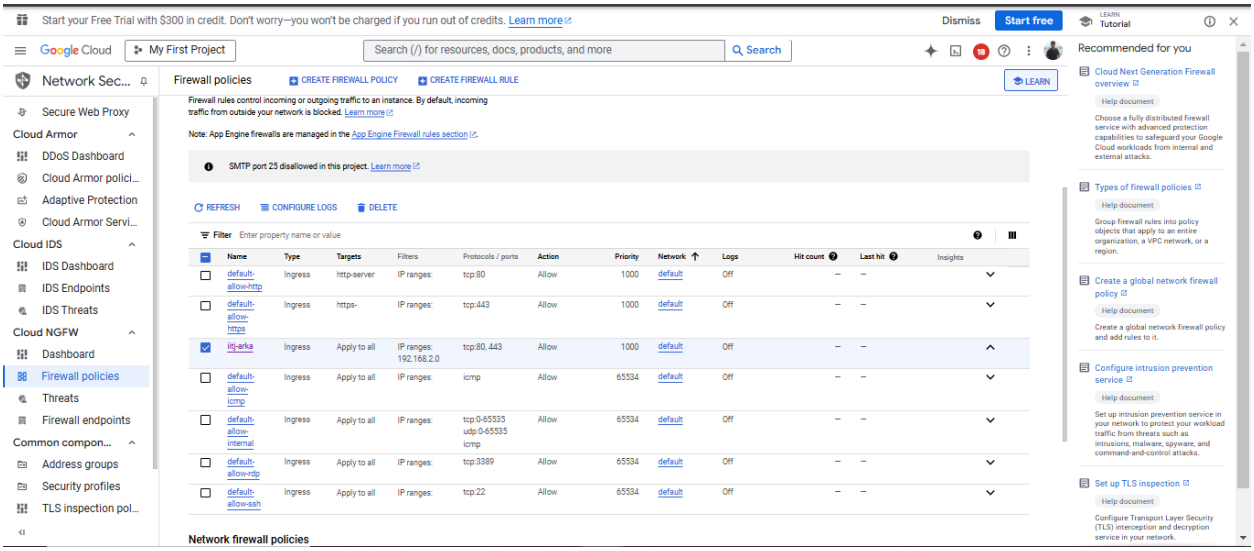



Figure 9: Setting Firewall

---

# 4. Testing and Validation

Once all configurations are completed, it is important to conduct thorough **testing and validation**. This section outlines steps to verify the deployment by checking **instance scaling and system performance under different loads**.

After setting up auto-scaling and security measures, testing is essential to confirm proper functionality. This section outlines validation steps such as monitoring instance scaling.

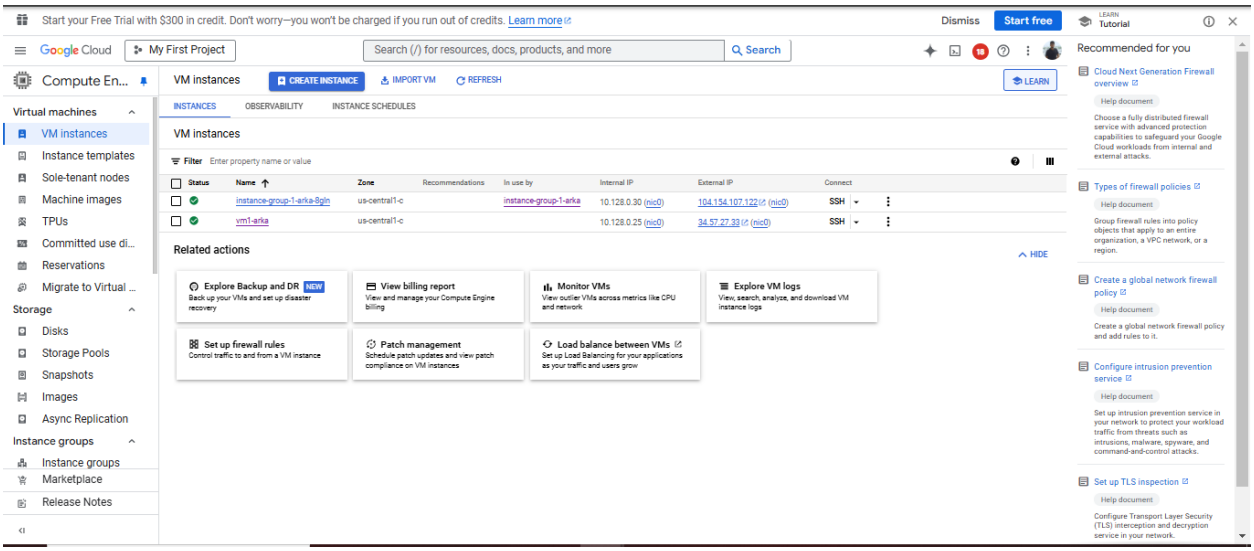



**Figure 10: Initial VM instances**

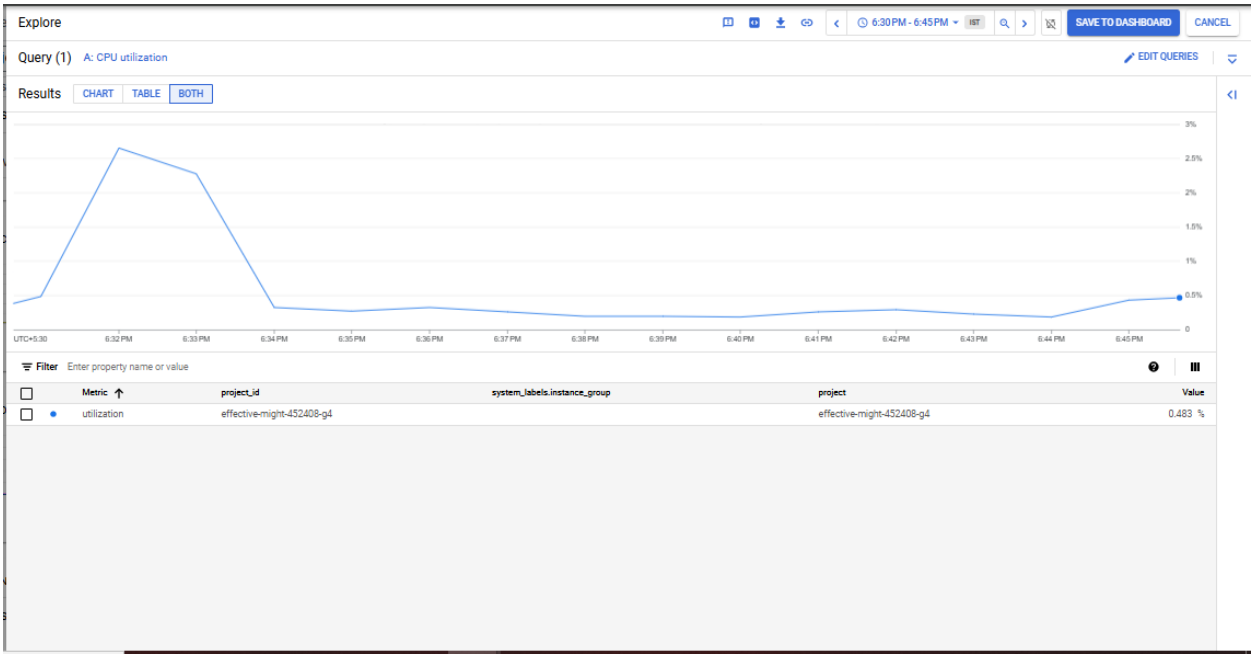



**Figure 11: Initial CPU utilization**

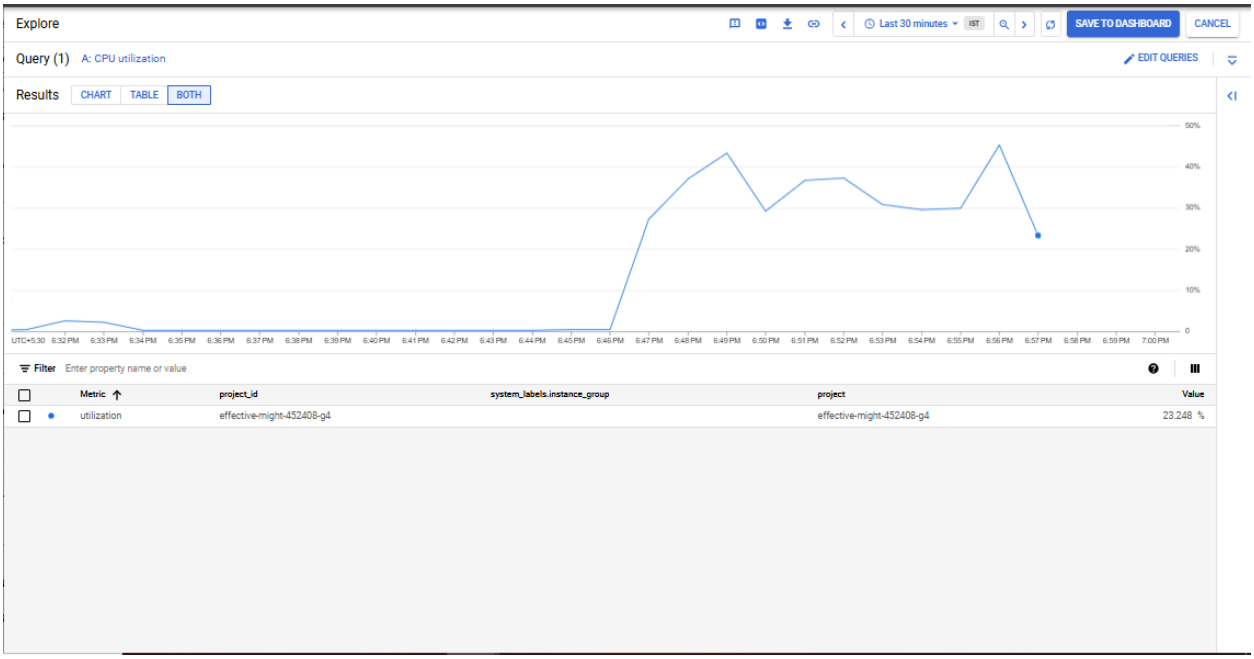



Figure 12: CPU utilization after creating instances

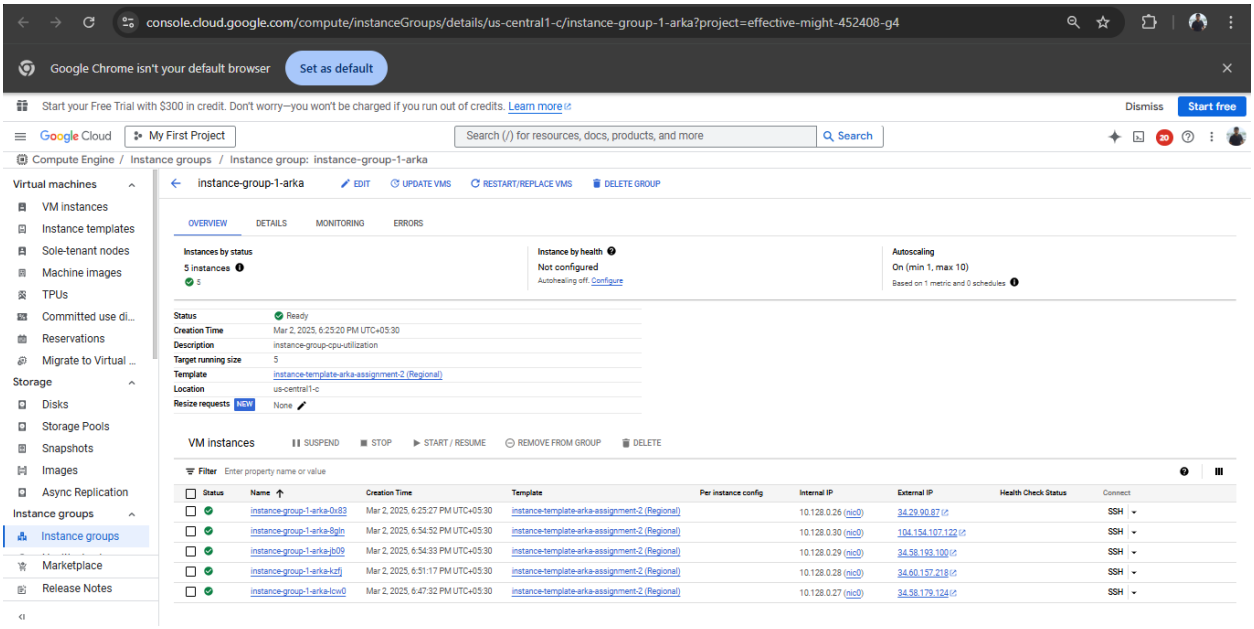
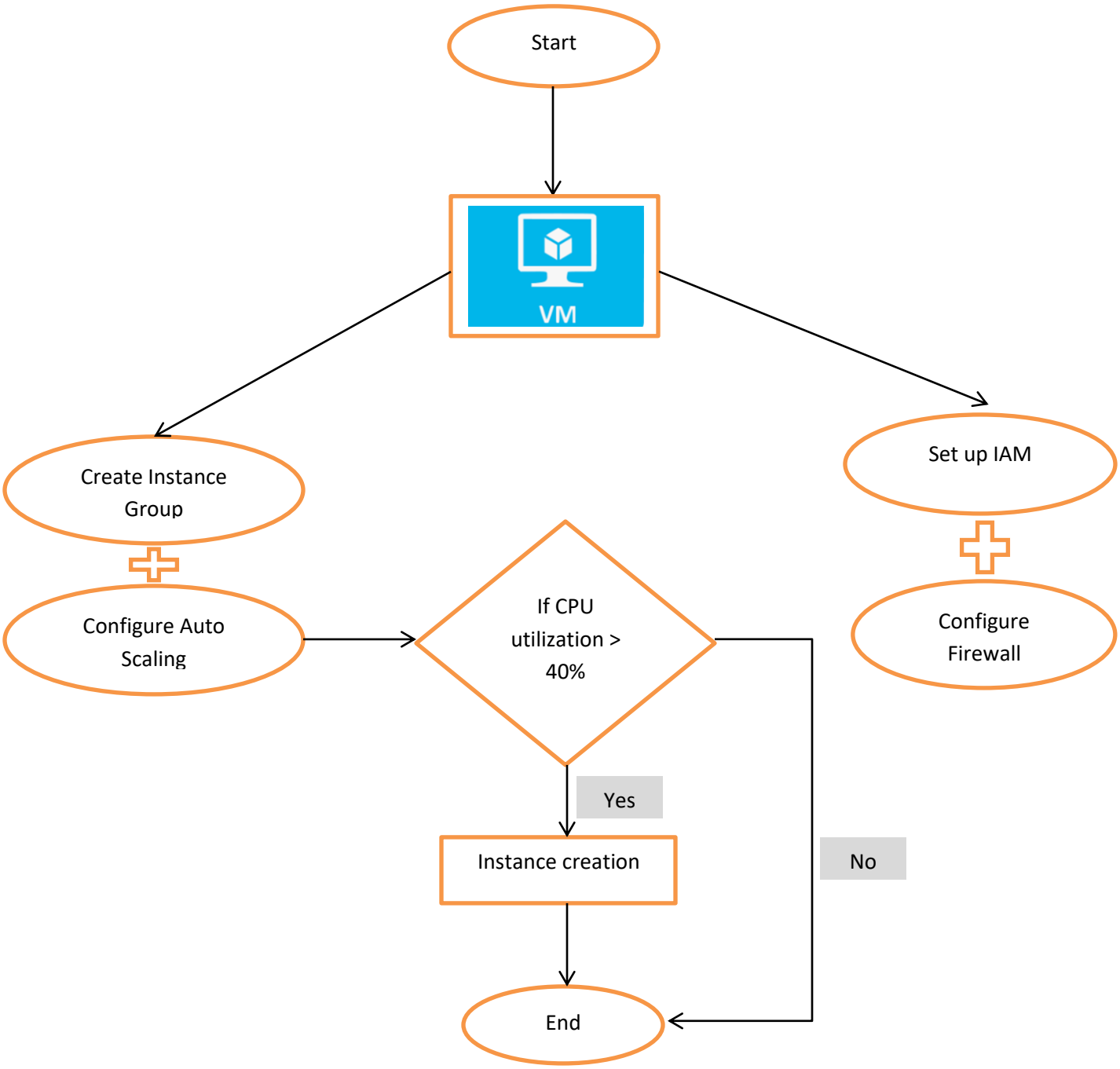



Figure 13: Instances created after CPU utilization increases

## 5. Architechture Diagram



## 6. Source Code Repository

**Github Repository Link**

https://github.com/ArkaGayen16/VccAssignment2.git

---

## 7. Recorded Video Demo

https://drive.google.com/file/d/1KLw-0XSxSSB0Fls3z3xY4APX0uTi1ijZ/view?usp=drive_web

---

## 8. Deliverables

**Included:**

1. **Project Report** (This Document)
2. **Architecture Diagram**
3. **Source Code Repository**
4. **Recorded Video Demo**

## 9. Conclusion

This project establishes a robust cloud infrastructure using GCP. By leveraging **Virtual Machines, Auto-Scaling, IAM roles, and Firewall configurations**, organizations can build a **secure, scalable, and cost-efficient** cloud computing environment.

This project establishes a robust cloud infrastructure using GCP. By leveraging VM instances, auto-scaling, and security configurations, organizations can optimize performance, ensure

reliability, and maintain strong security controls. This setup provides a robust solution with: A **VM instance** running in GCP.

- **Auto-scaling** based on CPU usage.
- **IAM roles** restricting access.
- **Firewall rules** securing the network.

---