

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.

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**.

The screenshot shows the 'Machine configuration' step of the 'Create an instance' wizard in Google Cloud. The machine name is 'vm1-aria'. The region is 'us-central1 (Iowa)' and the zone is 'Any'. The 'General purpose' tab is selected. A table of machine types is displayed, with 'E2' selected. The monthly estimate is \$25.46.

Item	Monthly estimate
2 vCPU + 4 GB memory	\$24.46
10 GB balanced persistent disk	\$1.00
Total	\$25.46

Series	Description	vCPUs	Memory	CPU Platform
<input type="radio"/> C4	Consistently high performance	2 - 192	4 - 1,488 GB	Intel Emerald Rapids
<input type="radio"/> C4A	Arm-based consistently high performance	1 - 72	2 - 576 GB	Google Azion
<input type="radio"/> N4	Flexible & cost-optimized	2 - 80	4 - 640 GB	Intel Emerald Rapids
<input type="radio"/> C3	Consistently high performance	4 - 192	8 - 1,536 GB	Intel Sapphire Rapids
<input type="radio"/> C3D	Consistently high performance	4 - 260	8 - 2,880 GB	AMD Genoa
<input checked="" type="radio"/> E2	Low cost, day-to-day computing	0.25 - 32	1 - 128 GB	Intel Broadwell
<input type="radio"/> N2	Balanced price & performance	2 - 128	2 - 864 GB	Intel Cascade Lake
<input type="radio"/> N2D	Balanced price & performance	2 - 224	2 - 896 GB	AMD Milan
<input type="radio"/> T2A	Scale-out workloads	1 - 48	4 - 192 GB	Ampere Altra
<input type="radio"/> T2D	Scale-out workloads	1 - 60	4 - 240 GB	AMD Milan
<input type="radio"/> N1	Balanced price & performance	0.25 - 96	0.6 - 624 GB	Intel Haswell

Figure 1: VM creation

The screenshot shows the 'Boot disk' step of the 'Create an instance' wizard. The operating system is 'Ubuntu'. The version is 'Ubuntu 20.04 LTS'. The boot disk type is 'SSD persistent disk'. The size is '20 GB'.

Operating system and storage

Name: vm1-aria
Type: New balanced persistent disk
Size: 10 GB
Snapshot schedule: No schedule selected
License type: Free
Image: Debian GNU/Linux 12 (bookworm)

Additional storage and VM backups

+ ADD NEW DISK + ATTACH EXISTING DISK + ADD LOCAL SSD

Backup plan

Secure your backups against deletion through backup vault storage and enable centralized backup management across projects. Managed by Backup and DR Service, a separate service from Compute Engine with independent certifications and accreditation. [Learn more](#)

Backup plan: SELECT A PLAN

Container

Deploy a container image to this VM instance

DEPLOY CONTAINER

Boot disk

Select an image or snapshot to create a boot disk or attach an existing disk. Can't find what you're looking for? Explore hundreds of VM solutions in [Marketplace](#)

PUBLIC IMAGES CUSTOM IMAGES SNAPSHOTS ARCHIVE SNAPSHOTS EXISTING DISKS

Operating system: Ubuntu
Version: Ubuntu 20.04 LTS
x86_64, amd64 focal image built on 2020-02-13
Boot disk type: SSD persistent disk
COMPARE DISK TYPES
Size (GB): 20
Provision between 10 and 65536 GB
SHOW ADVANCED CONFIGURATION
SELECT CANCEL

Figure 2: VM OS setup

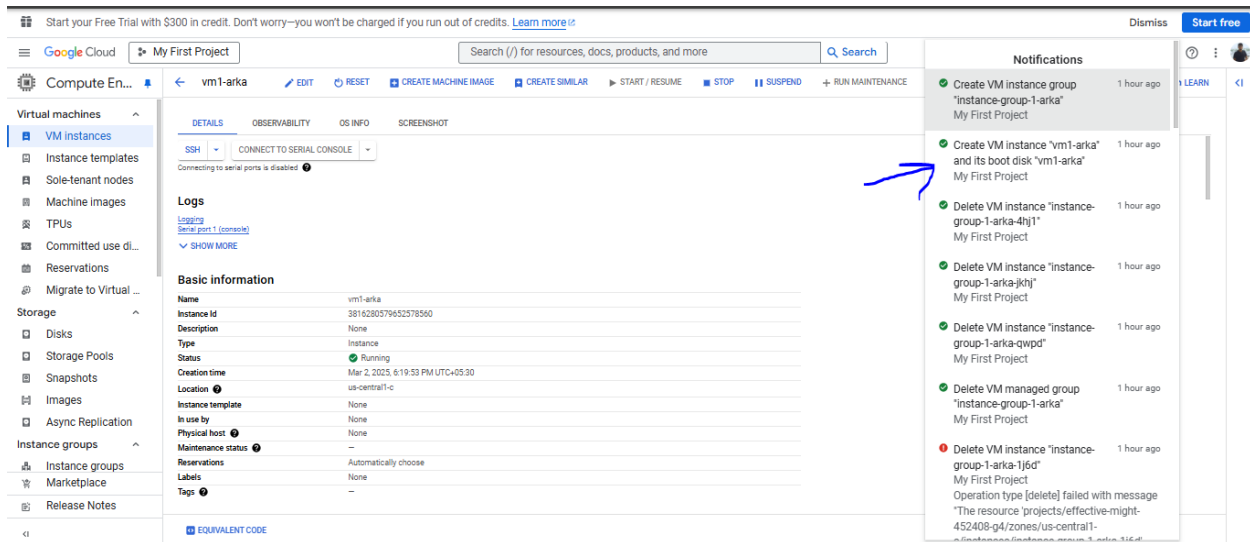


Figure 3: VM after creation

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**.

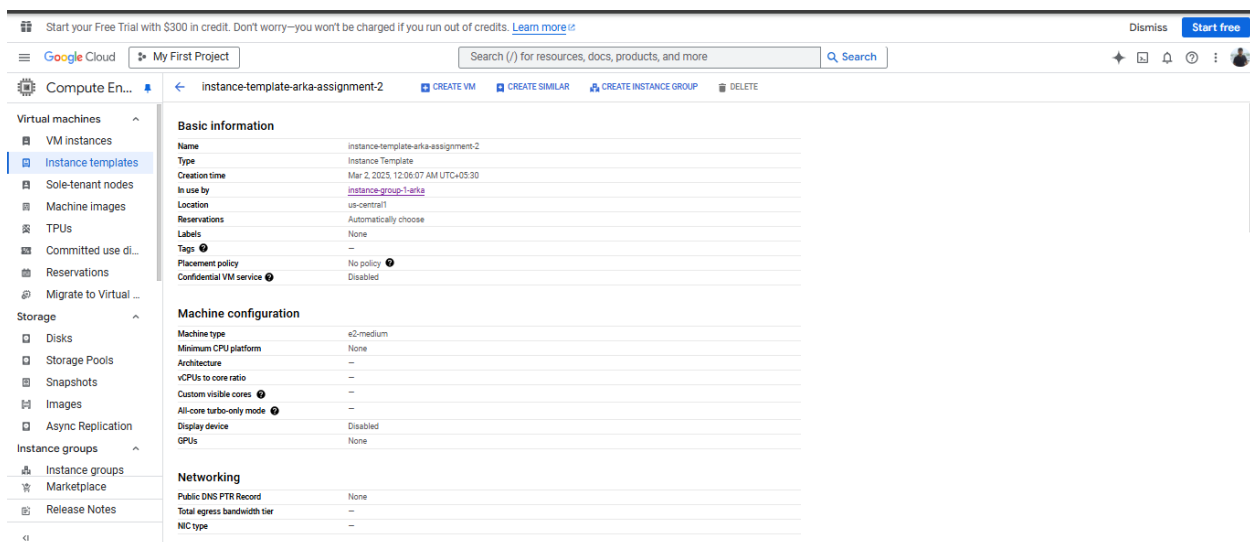


Figure 4: Instance template after creation

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**.

The screenshot shows the 'Create Instance Group' page in the Google Cloud Console. The page is for 'My First Project' and is titled 'Create Instance Group'. It features a search bar and navigation links for 'Compute Engine / Instance groups / Create instance group'. The main content area is divided into several sections:

- Autoscaling mode:** A dropdown menu set to 'On: add and remove instances to the group'.
- Minimum number of instances:** A numeric input field set to '1'.
- Maximum number of instances:** A numeric input field set to '10'.
- Autoscaling signals:** A section with a dropdown menu set to 'CPU utilization: 60% (default)' and a note 'Predictive autoscaling is off'. There is an 'ADD A SIGNAL' button.
- Autoscaling schedules:** A section with a dropdown menu.
- Initialization period:** A section with a note 'Specify how long it takes for your app to initialize from boot time until it is ready to serve.' and a numeric input field set to '60' seconds.
- Scale-in controls:** A section with a dropdown menu.

At the bottom of the page, there are buttons for 'CREATE', 'CANCEL', and 'EQUIVALENT CODE'.

Figure 5: Instance group creation for autoscaling

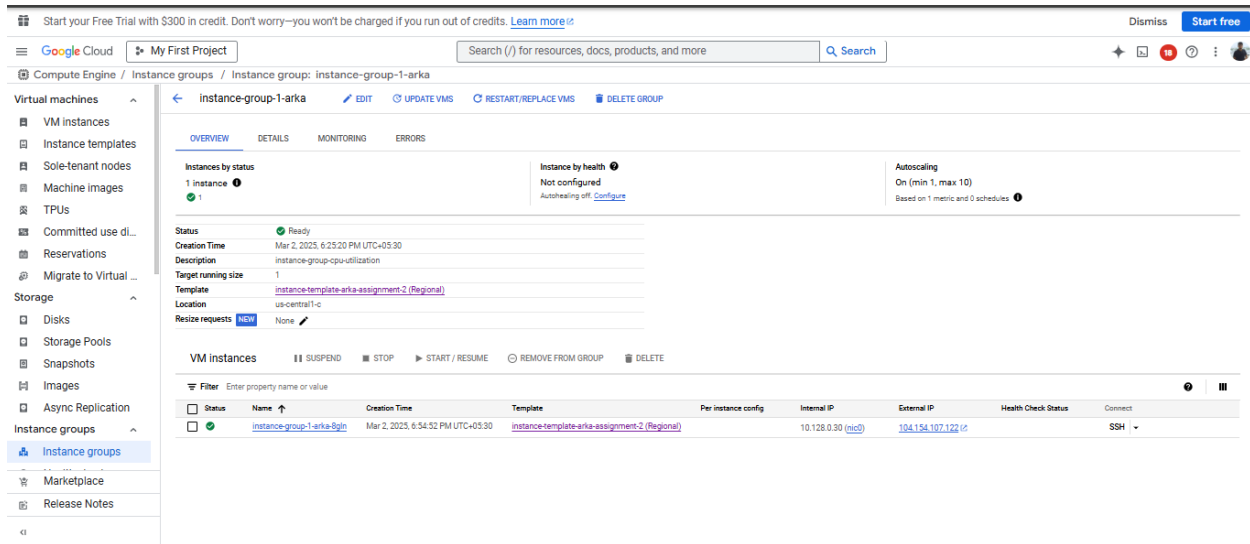


Figure 6: Instance group after creation

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.

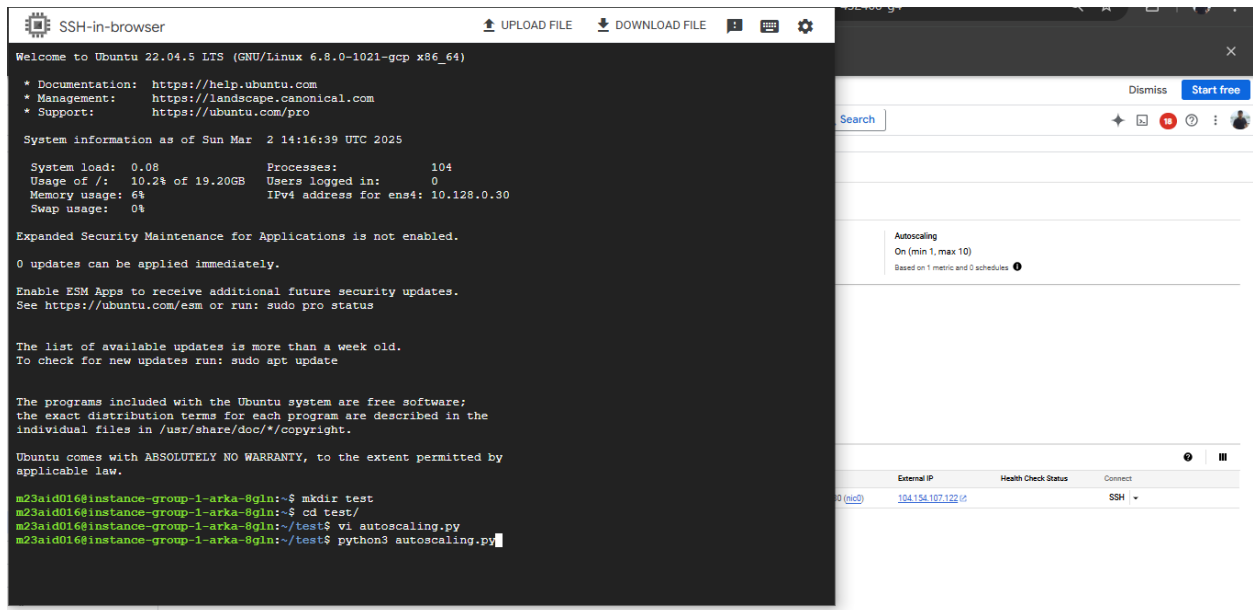


Figure 7: Executing program

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**.

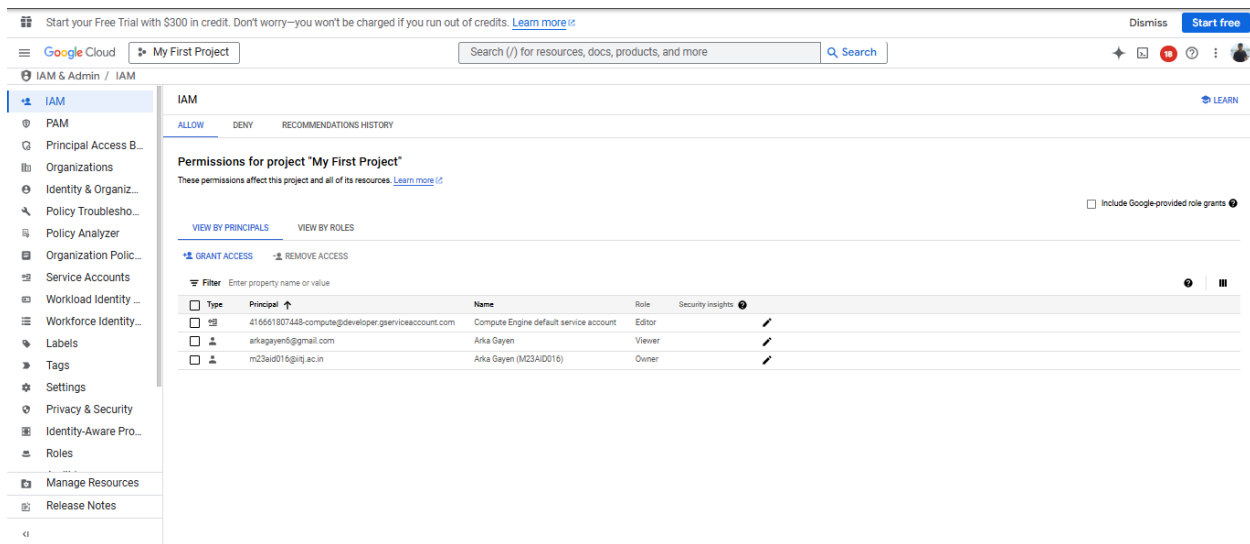


Figure 8: Setting IAM

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**.

The screenshot displays the Google Cloud Network Security console. The left sidebar shows the navigation menu with 'Firewall policies' selected. The main content area shows the 'my-arka' firewall policy details. A table lists the rules for this policy:

Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Network	Logs	Hit count	Last hit	Insights
default-allow-http	Ingress	http-server	IP ranges	tcp:80	Allow	1000	default	Off	--	--	▼
default-allow-https	Ingress	https-	IP ranges	tcp:443	Allow	1000	default	Off	--	--	▼
my-arka	Ingress	Apply to all	IP ranges: 192.168.2.0	tcp:80, 443	Allow	1000	default	Off	--	--	▲
default-allow-icmp	Ingress	Apply to all	IP ranges	icmp	Allow	65534	default	Off	--	--	▼
default-allow-internal	Ingress	Apply to all	IP ranges	tcp:0-65535, udp:0-65535	Allow	65534	default	Off	--	--	▼
default-allow-rdp	Ingress	Apply to all	IP ranges	tcp:3389	Allow	65534	default	Off	--	--	▼
default-allow-ssh	Ingress	Apply to all	IP ranges	tcp:22	Allow	65534	default	Off	--	--	▼

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.

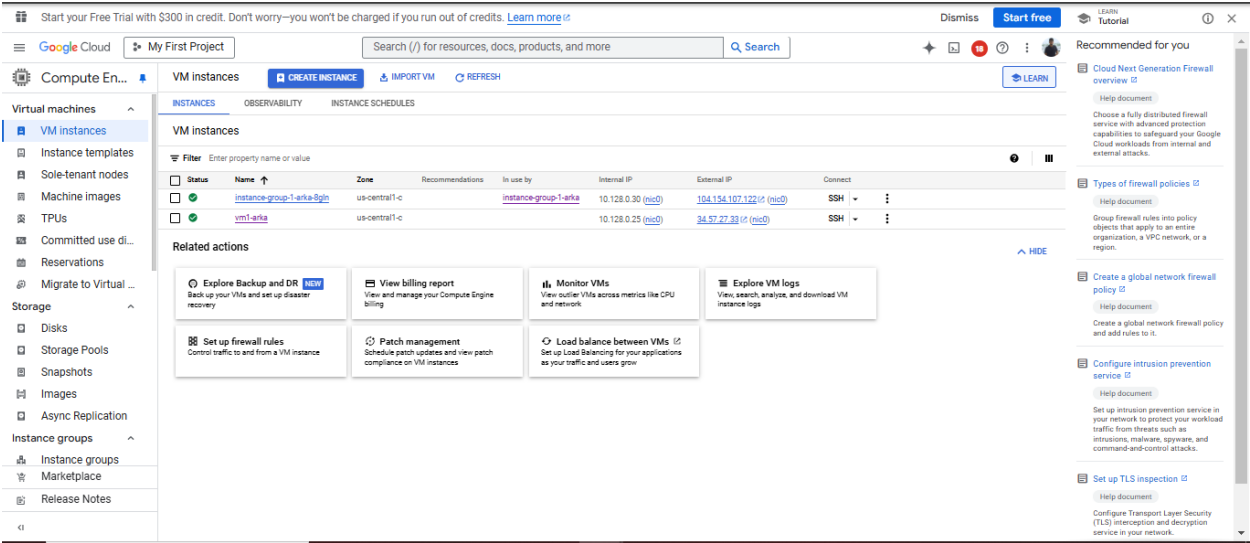


Figure 10: Initial VM instances

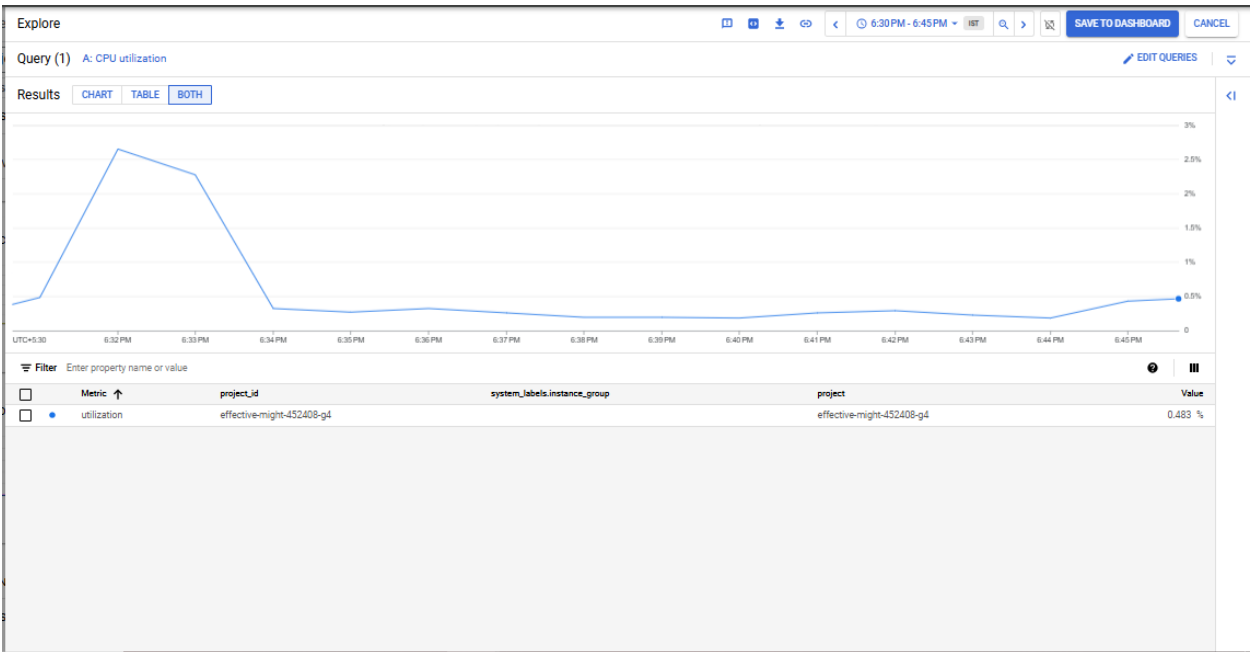


Figure 11: Initial CPU utilization

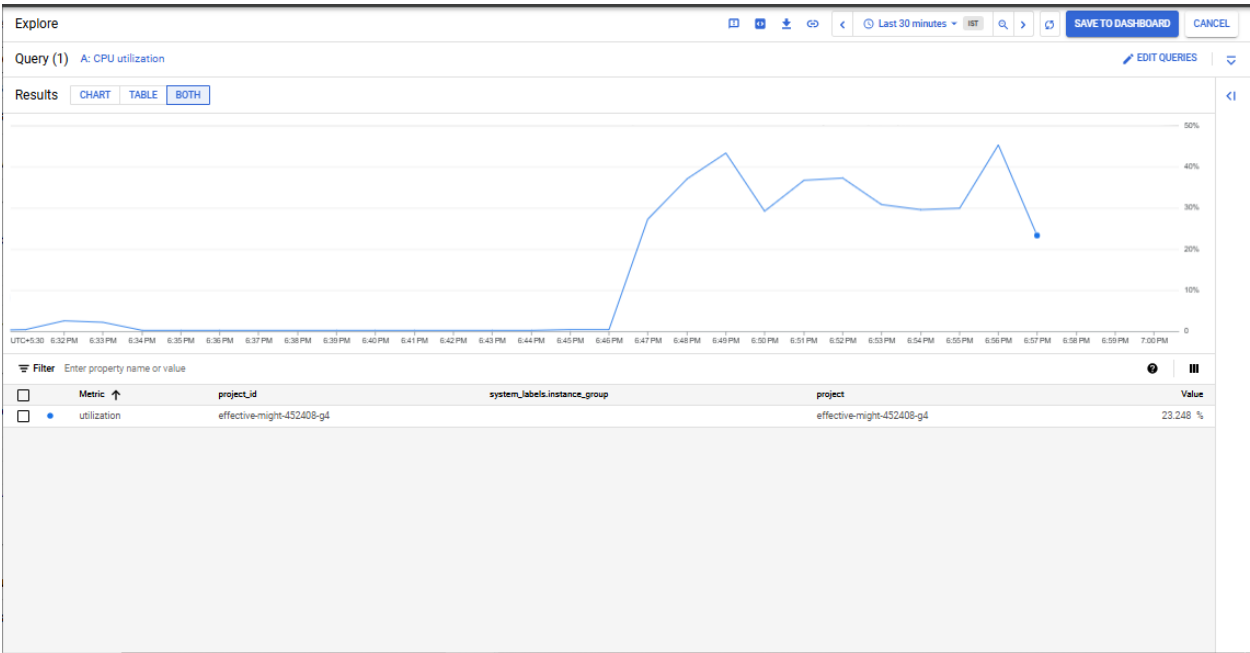


Figure 12: CPU utilization after creating instances

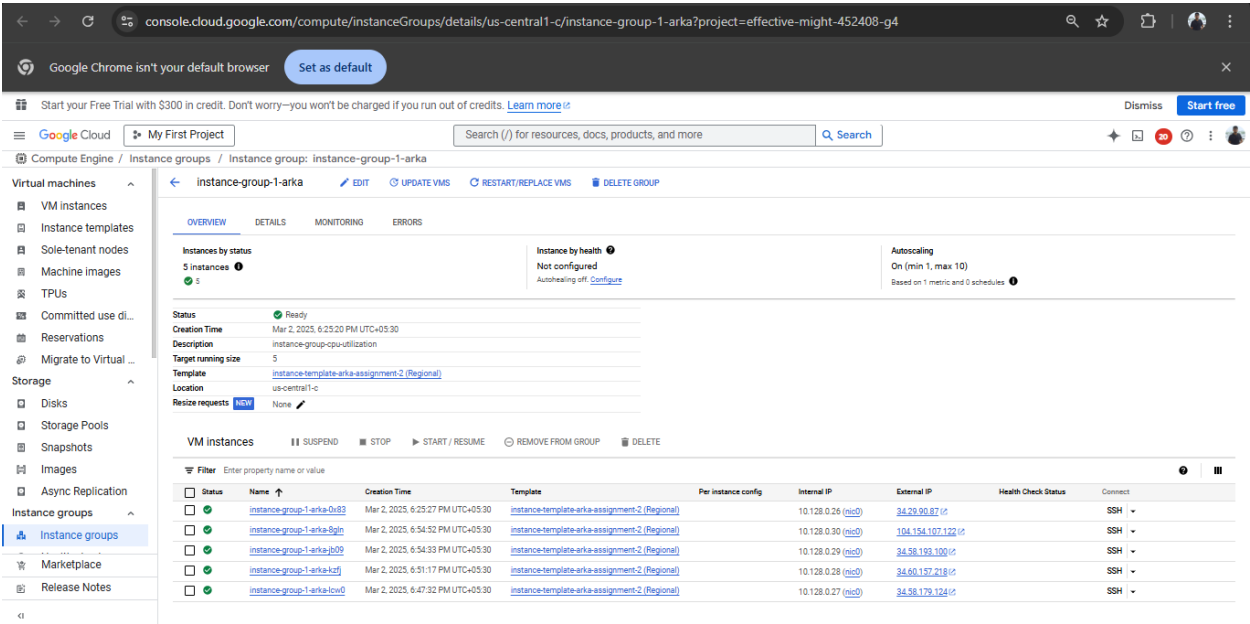
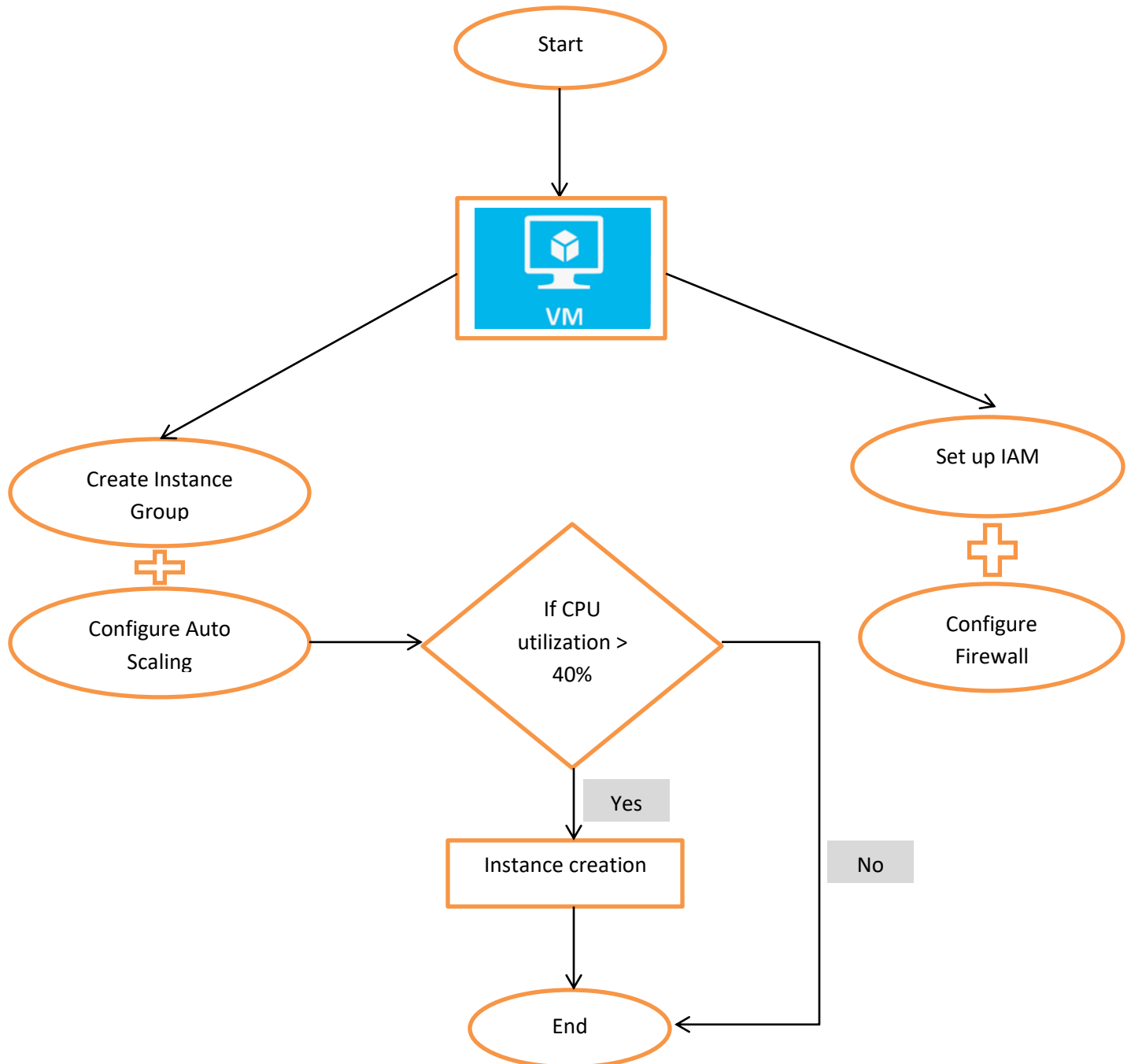


Figure 13: Instances created after CPU utilization increases

5. Architecture 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.
-