Report

on

Auto-Scaling from Local VM to Google Cloud Platform (GCP)

Course Code

CSL7510

submitted by

Ankit Kumar Chauhan (M23CSA509)

M.Tech- Artificial Intelligence (Executive)

Course instructor

Dr. SUMIT KALRA

Department of Computer Science and Engineering

Indian Institute of Technology Jodhpur
NH 65, Nagaur Road, Karwar,
Jodhpur 342030, INDIA

❖ Introduction

The objective of this assignment is to create a local virtual machine (VM) and implement a mechanism to monitor its resource usage (CPU and/or memory). When the resource usage exceeds a threshold of 75%, additional compute resources are automatically provisioned in a public cloud—in this case, Google Cloud Platform (GCP). This ensures that the application can handle the increased load without manual intervention.

- **Key Objectives**

1. **Local VM Creation**: Using VirtualBox.
2. **Resource Monitoring**: Implemented with a custom script.
3. **Auto-Scaling to Cloud**: Automatic provisioning and scaling of resources on GCP once threshold is exceeded.
4. **Sample Application**: Demonstrate the entire flow using a simple web application.

❖ Architecture Overview

The below Fig. 1 shows a high-level summary of the architecture:

1. **Local VM**
   - Runs Apache (web server) and a monitoring script that checks CPU and memory usage.
   - Hosts a simple PHP page (index.php) to demonstrate the application.
   - If usage exceeds 75%, it triggers auto-scaling logic.
2. **Resource Monitoring**
   - A custom script monitors CPU and memory usage on the local VM.
   - Once usage crosses the threshold, the script initiates GCP resource creation or scaling actions.
3. **Google Cloud Platform (GCP)**
   - **Compute Engine**: Houses a Managed Instance Group (MIG) that can scale up/down based on load.
   - **Instance Template**: Defines the configuration for VM instances (machine type, startup script, etc.).

- **Load Balancer (LB)**: Front-ends the MIG instances.
- **Firewall Rules**: Allows inbound traffic (HTTP, health checks) to the MIG.
- **Cloud Storage (GCS)**: Stores web content that each GCP VM can download during startup.

4. **Auto-Scaling Flow**
   - Local VM monitors resource usage.
   - If usage > 75% and no GCP VMs are active, the script:
     1. Uploads local web content to a GCS bucket.
     2. Creates a GCP instance template and a Managed Instance Group (MIG).
     3. Sets up a Load Balancer (LB) in front of the MIG.
     4. Routes traffic to the GCP Load Balancer once GCP instances are live.
   - If usage > 75% again and GCP instances are already active, the script increases the MIG size by 1 to handle the load.

5. **Traffic Routing**
   - A rewrite rule in Apache on the local VM checks how many GCP VMs are active.
   - If there is at least one active GCP VM, traffic is proxied from the local VM to the GCP Load Balancer IP.

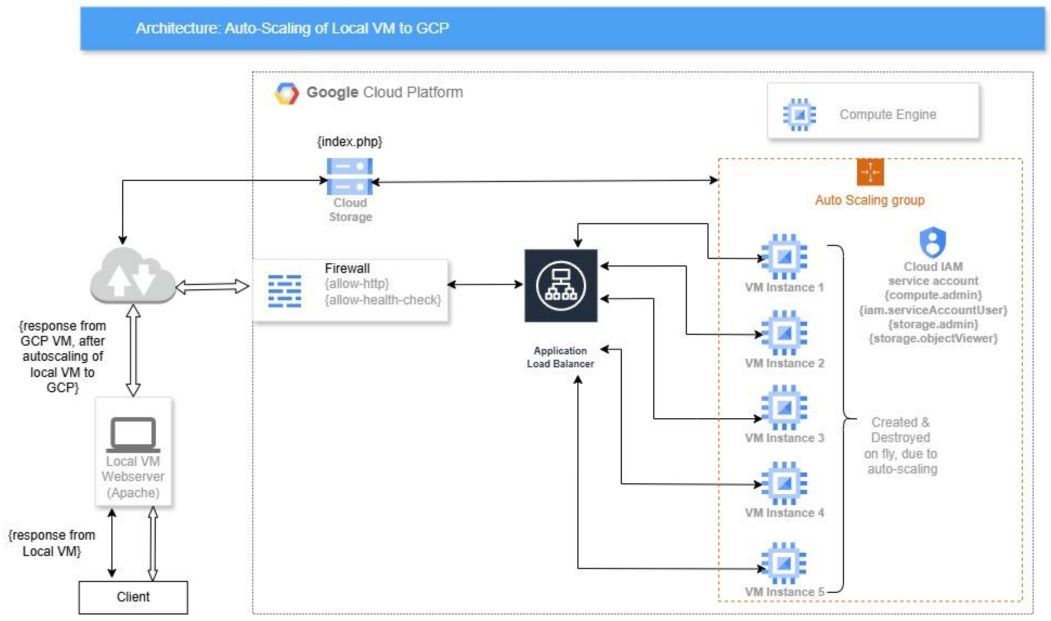


Fig. 1. Architecture of auto-scaling of Local VM to GCP

❖ Step-by-Step Implementation Guide

- **Prerequisites**

  ☑ **Local Environment**:

  - A machine capable of running VirtualBox.
  - Ubuntu, Lubuntu, or another Linux distro as the guest OS on the VM.

  ☑ **Google Cloud Platform Account**:

  - A GCP Project with billing enabled.
  - Sufficient IAM privileges to create Compute Engine resources, service accounts, and Cloud Storage buckets.

  ☑ **Installed Tools**:

  - VirtualBox.
  - Google Cloud SDK (can also be installed via script).

- **Create a Local VM**

1. **Downloaded and Installed Virtual Box**.
2. **Create a New VM**:
   - Allocated sufficient CPU, memory (e.g., 2 vCPUs, 4 GB RAM).
   - Attached an ISO image for a lightweight OS (Lubuntu).
   - Install and boot into the OS.
   - Ensured that VM can access the internet (Bridged Adapter).
   - Update and upgraded the VM packages (e.g., sudo apt update && sudo apt upgrade -y).

- **Configure the Local VM and the Auto-Scaling Script**

One can use the attached `auto_scaling_script_gcs.sh` as the main orchestrator. The following below is an outline of its functionality and how to run it.

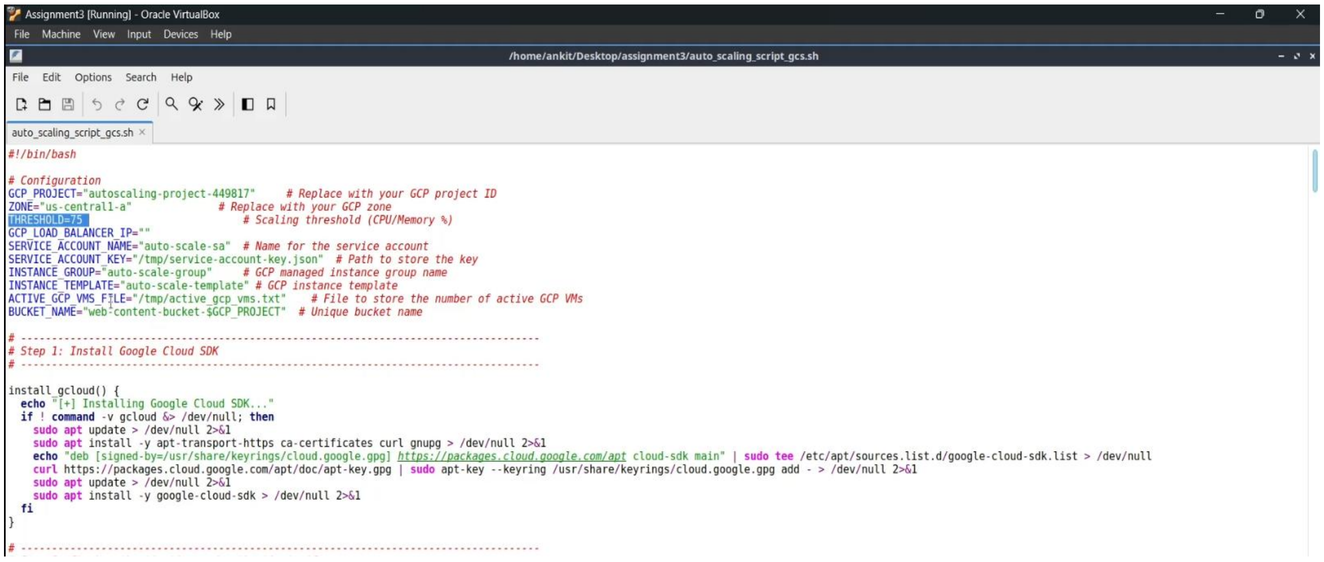
1. **Download the Script**
   - Placed it in my VM, for instance at `/home/user/auto_scaling_script_gcs.sh`.

- Make it executable by running below command:

chmod +x auto_scaling_script_gcs.sh

2. **Edit Configuration Variables**
   Inside auto_scaling_script_gcs.sh, modified the following to match GCP setup (Fig. 2):

   - GCP_PROJECT="PROJECT_ID"
   - ZONE="PREFERRED_GCP_ZONE"
   - Adjust threshold to 75 as desired.



Assignment3 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
/home/ankit/Desktop/assignment3/auto_scaling_script_gcs.sh
File Edit Options Search Help
auto_scaling_script_gcs.sh

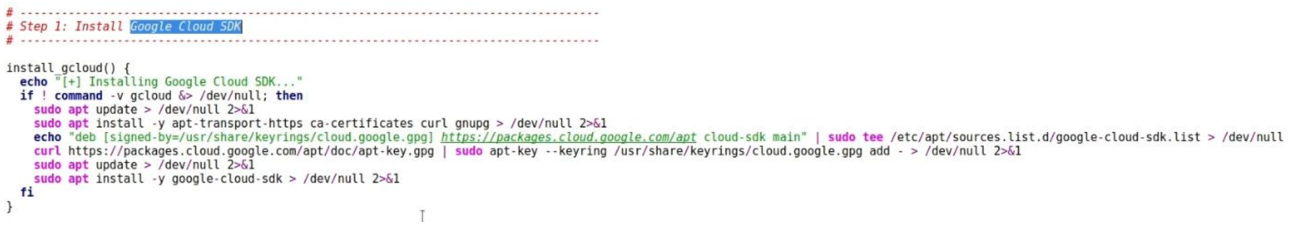
```
#!/bin/bash

# Configuration
GCP_PROJECT="autoscaling-project-449817"     # Replace with your GCP project ID
ZONE="us-central1-a"              # Replace with your GCP zone
THRESHOLD=75                         # Scaling threshold (CPU/Memory %)
GCP_LOAD_BALANCER_IP=""
SERVICE_ACCOUNT_NAME="auto-scale-sa"  # Name for the service account
SERVICE_ACCOUNT_KEY="/tmp/service-account-key.json"  # Path to store the key
INSTANCE_GROUP="auto-scale-group"     # GCP managed instance group name
INSTANCE_TEMPLATE="auto-scale-template" # GCP instance template
ACTIVE_GCP_VMS_FILE="/tmp/active_gcp_vms.txt"    # File to store the number of active GCP VMs
BUCKET_NAME="web-content-bucket-$GCP_PROJECT"  # Unique bucket name

# ---------------------------------------------------------------------------
# Step 1: Install Google Cloud SDK
# ---------------------------------------------------------------------------

install_gcloud() {
  echo "[+] Installing Google Cloud SDK..."
  if ! command -v gcloud &> /dev/null; then
    sudo apt update > /dev/null 2>&1
    sudo apt install -y apt-transport-https ca-certificates curl gnupg > /dev/null 2>&1
    echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee /etc/apt/sources.list.d/google-cloud-sdk.list > /dev/null
    curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring /usr/share/keyrings/cloud.google.gpg add - > /dev/null 2>&1
    sudo apt update > /dev/null 2>&1
    sudo apt install -y google-cloud-sdk > /dev/null 2>&1
  fi
}

# ---------------------------------------------------------------------------
```

Fig. 2. Script section showing variables

3. **Run the Script (using below command)**

`./auto_scaling_script_gcs.sh`

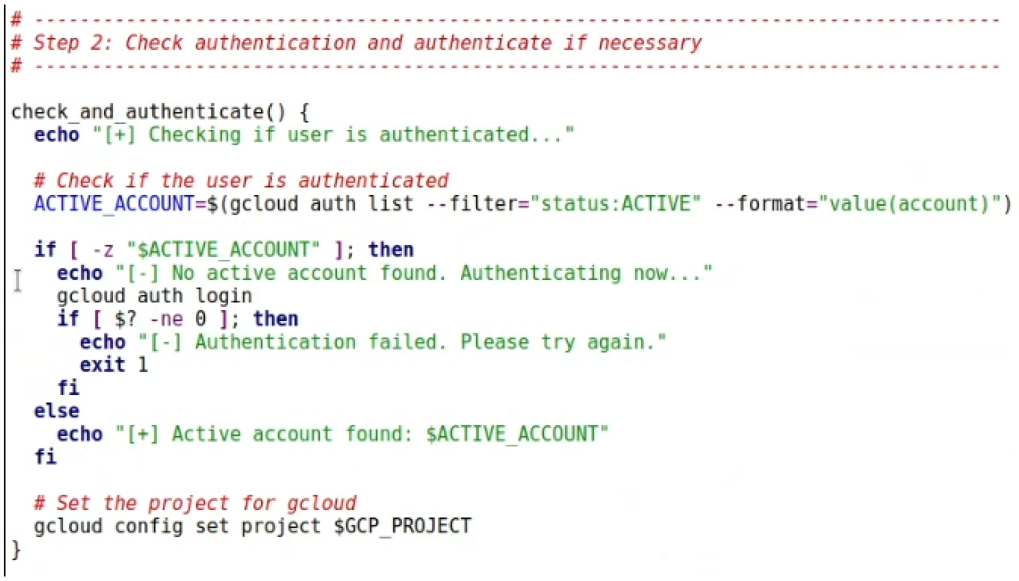
   - The script will:
     1. **Install Google Cloud SDK** (if not installed) (Fig. 3)



```
# ---------------------------------------------------------------------------
# Step 1: Install Google Cloud SDK
# ---------------------------------------------------------------------------

install_gcloud() {
  echo "[+] Installing Google Cloud SDK..."
  if ! command -v gcloud &> /dev/null; then
    sudo apt update > /dev/null 2>&1
    sudo apt install -y apt-transport-https ca-certificates curl gnupg > /dev/null 2>&1
    echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee /etc/apt/sources.list.d/google-cloud-sdk.list > /dev/null
    curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring /usr/share/keyrings/cloud.google.gpg add - > /dev/null 2>&1
    sudo apt update > /dev/null 2>&1
    sudo apt install -y google-cloud-sdk > /dev/null 2>&1
  fi
}
```

Fig. 3. Function (install_gcloud) install google cloud sdk

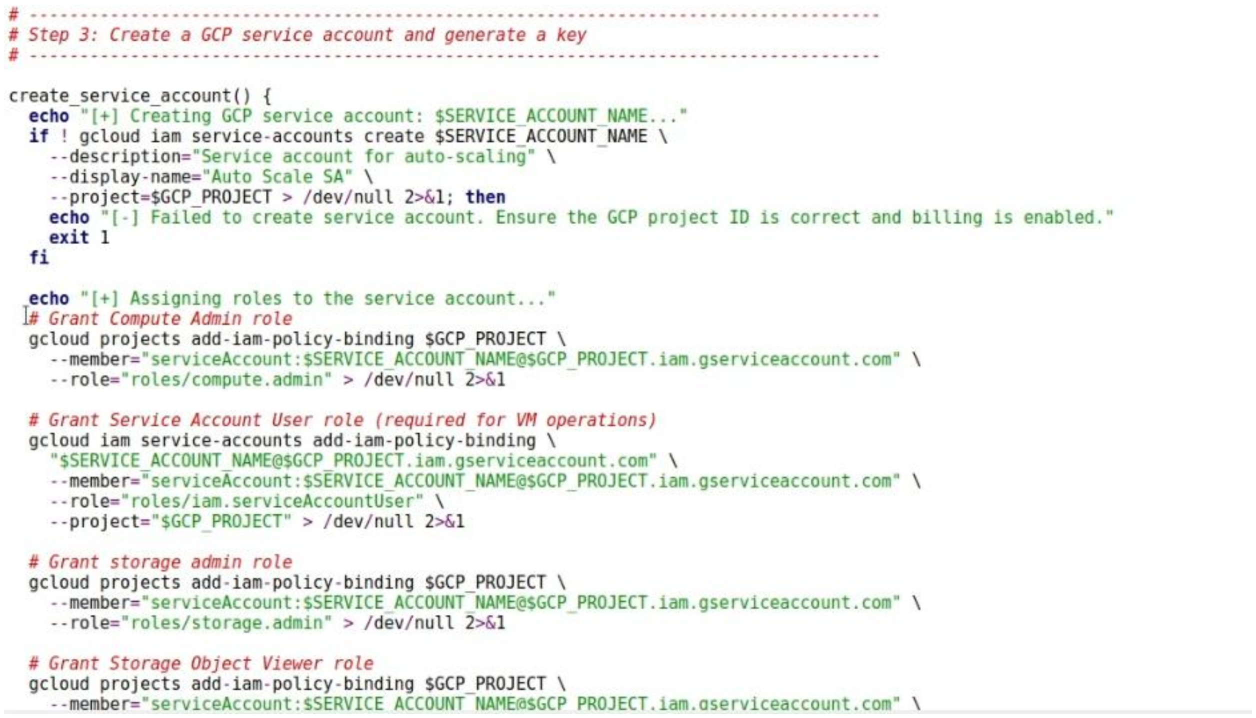
2. **Authenticate** you to GCP (As shown in below Fig. 4).



```
# ------------------------------------------------------------------------------------
# Step 2: Check authentication and authenticate if necessary
# ------------------------------------------------------------------------------------

check_and_authenticate() {
  echo "[+] Checking if user is authenticated..."

  # Check if the user is authenticated
  ACTIVE_ACCOUNT=$(gcloud auth list --filter="status:ACTIVE" --format="value(account)")

  if [ -z "$ACTIVE_ACCOUNT" ]; then
    echo "[-] No active account found. Authenticating now..."
    gcloud auth login
    if [ $? -ne 0 ]; then
      echo "[-] Authentication failed. Please try again."
      exit 1
    fi
  else
    echo "[+] Active account found: $ACTIVE_ACCOUNT"
  fi

  # Set the project for gcloud
  gcloud config set project $GCP_PROJECT
}
```

Fig. 4. Function (check and authenticate) authenticates to GCP

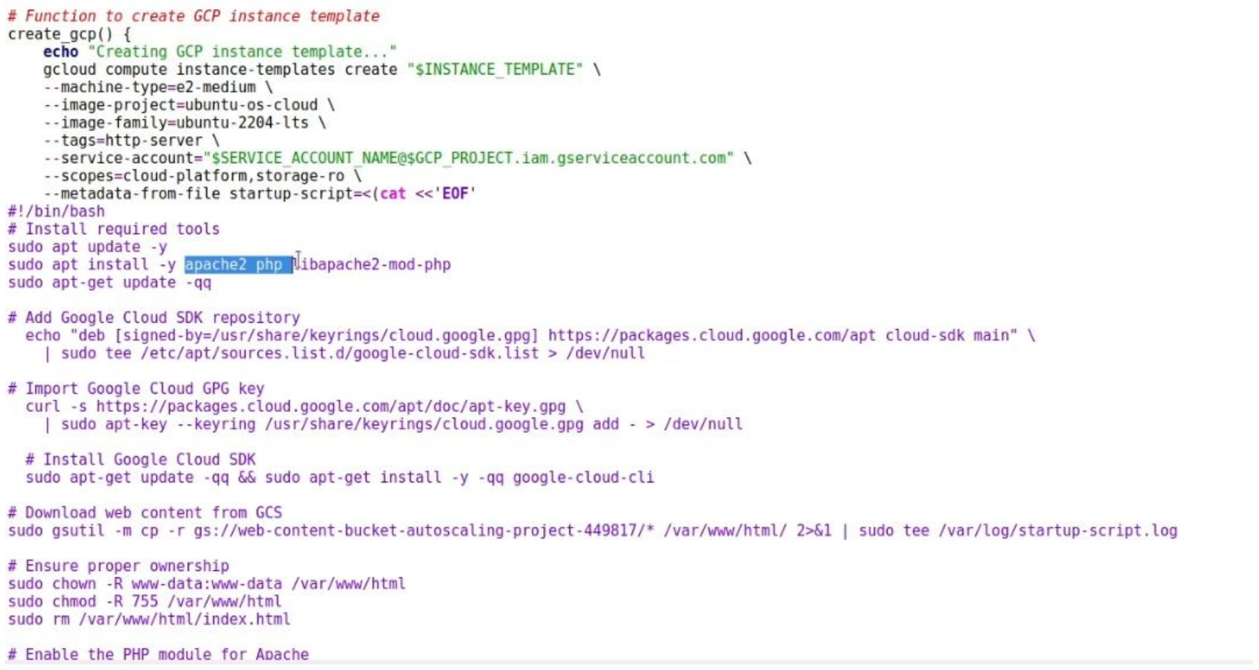
3. **Create a Service Account** and assign necessary roles (Compute Admin, Storage Admin, etc.) (As shown in below Fig. 5).



```
# ------------------------------------------------------------------------------------
# Step 3: Create a GCP service account and generate a key
# ------------------------------------------------------------------------------------

create_service_account() {
  echo "[+] Creating GCP service account: $SERVICE_ACCOUNT_NAME..."
  if ! gcloud iam service-accounts create $SERVICE_ACCOUNT_NAME \
    --description="Service account for auto-scaling" \
    --display-name="Auto Scale SA" \
    --project=$GCP_PROJECT > /dev/null 2>&1; then
    echo "[-] Failed to create service account. Ensure the GCP project ID is correct and billing is enabled."
    exit 1
  fi

  echo "[+] Assigning roles to the service account..."
  # Grant Compute Admin role
  gcloud projects add-iam-policy-binding $GCP_PROJECT \
    --member="serviceAccount:$SERVICE_ACCOUNT_NAME@$GCP_PROJECT.iam.gserviceaccount.com" \
    --role="roles/compute.admin" > /dev/null 2>&1

  # Grant Service Account User role (required for VM operations)
  gcloud iam service-accounts add-iam-policy-binding \
    "$SERVICE_ACCOUNT_NAME@$GCP_PROJECT.iam.gserviceaccount.com" \
    --member="serviceAccount:$SERVICE_ACCOUNT_NAME@$GCP_PROJECT.iam.gserviceaccount.com" \
    --role="roles/iam.serviceAccountUser" \
    --project="$GCP_PROJECT" > /dev/null 2>&1

  # Grant storage admin role
  gcloud projects add-iam-policy-binding $GCP_PROJECT \
    --member="serviceAccount:$SERVICE_ACCOUNT_NAME@$GCP_PROJECT.iam.gserviceaccount.com" \
    --role="roles/storage.admin" > /dev/null 2>&1

  # Grant Storage Object Viewer role
  gcloud projects add-iam-policy-binding $GCP_PROJECT \
    --member="serviceAccount:$SERVICE_ACCOUNT_NAME@$GCP_PROJECT.iam.gserviceaccount.com" \
```

Fig. 5. Function (create_service_account) creates service accounts and assigns necessary roles

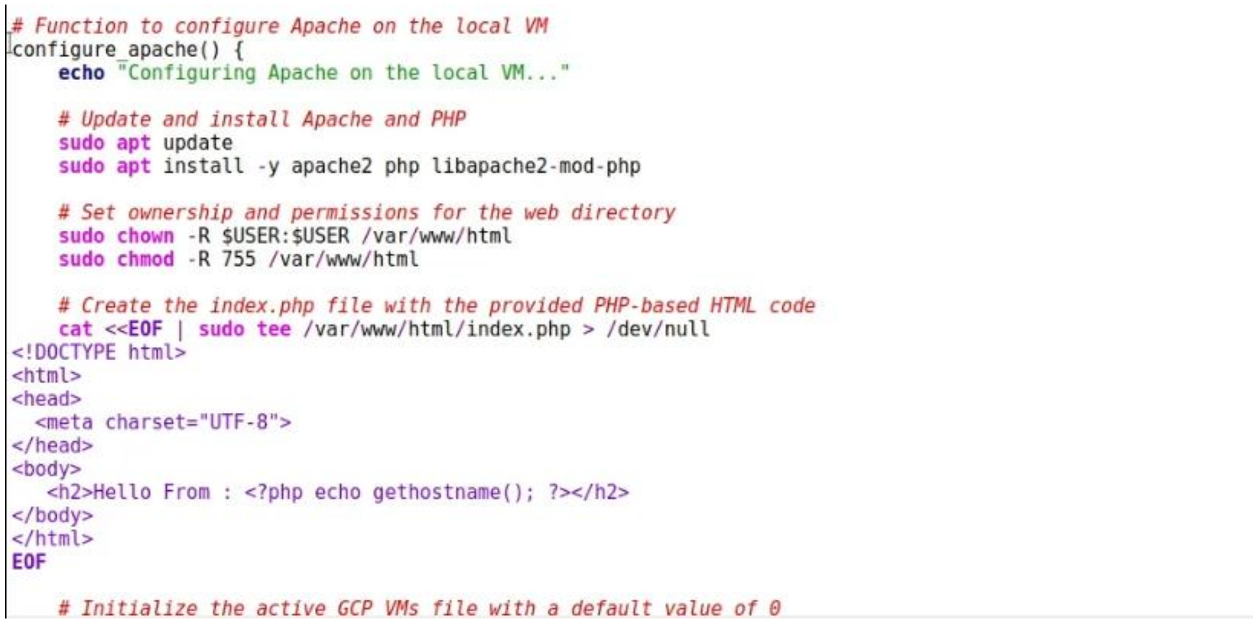
4. **Configure GCP template and Instance group** (as shown in below Fig. 6)



```
# Function to create GCP instance template
create_gcp() {
    echo "Creating GCP instance template..."
    gcloud compute instance-templates create "$INSTANCE_TEMPLATE" \
    --machine-type=e2-medium \
    --image-project=ubuntu-os-cloud \
    --image-family=ubuntu-2204-lts \
    --tags=http-server \
    --service-account="$SERVICE_ACCOUNT_NAME@$GCP_PROJECT.iam.gserviceaccount.com" \
    --scopes=cloud-platform,storage-ro \
    --metadata-from-file startup-script=<(cat <<'EOF'
#!/bin/bash
# Install required tools
sudo apt update -y
sudo apt install -y apache2 php libapache2-mod-php
sudo apt-get update -qq

# Add Google Cloud SDK repository
  echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] https://packages.cloud.google.com/apt cloud-sdk main" \
    | sudo tee /etc/apt/sources.list.d/google-cloud-sdk.list > /dev/null

# Import Google Cloud GPG key
  curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg \
    | sudo apt-key --keyring /usr/share/keyrings/cloud.google.gpg add - > /dev/null

  # Install Google Cloud SDK
  sudo apt-get update -qq && sudo apt-get install -y -qq google-cloud-cli

# Download web content from GCS
sudo gsutil -m cp -r gs://web-content-bucket-autoscaling-project-449817/* /var/www/html/ 2>&1 | sudo tee /var/log/startup-script.log

# Ensure proper ownership
sudo chown -R www-data:www-data /var/www/html
sudo chmod -R 755 /var/www/html
sudo rm /var/www/html/index.html

# Enable the PHP module for Apache
```

Fig. 6. Function (create_gcp) creates template with startup script and instance group

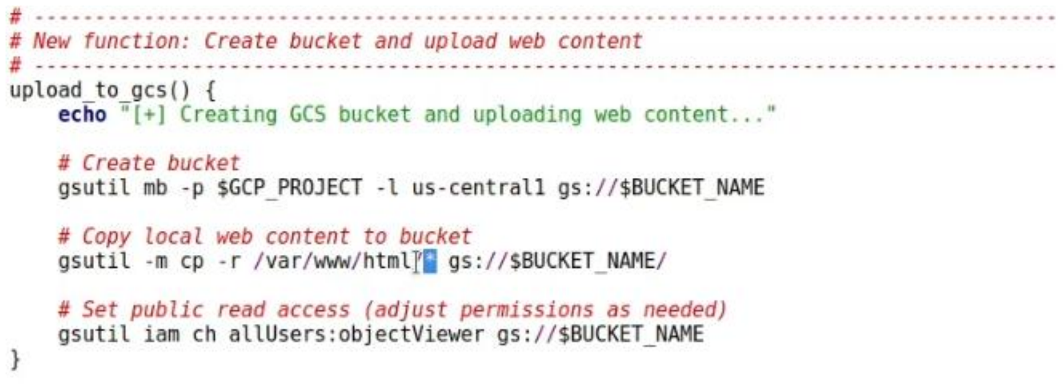
5. **Configure Apache** on the local VM (As shown in below Fig. 7).



```
# Function to configure Apache on the local VM
configure_apache() {
    echo "Configuring Apache on the local VM..."

    # Update and install Apache and PHP
    sudo apt update
    sudo apt install -y apache2 php libapache2-mod-php

    # Set ownership and permissions for the web directory
    sudo chown -R $USER:$USER /var/www/html
    sudo chmod -R 755 /var/www/html

    # Create the index.php file with the provided PHP-based HTML code
    cat <<EOF | sudo tee /var/www/html/index.php > /dev/null
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>
   <h2>Hello From : <?php echo gethostname(); ?></h2>
</body>
</html>
EOF

    # Initialize the active GCP VMs file with a default value of 0
```

Fig 7. Function (configure_apache) configures apache at local VM, with code for index.php

6. **Move local resources to GCP after auto-scaling kicks in** (As shown in below Fig. 8).



```
# ------------------------------------------------------------------------------
# New function: Create bucket and upload web content
# ------------------------------------------------------------------------------
upload_to_gcs() {
    echo "[+] Creating GCS bucket and uploading web content..."

    # Create bucket
    gsutil mb -p $GCP_PROJECT -l us-central1 gs://$BUCKET_NAME

    # Copy local web content to bucket
    gsutil -m cp -r /var/www/html/* gs://$BUCKET_NAME/

    # Set public read access (adjust permissions as needed)
    gsutil iam ch allUsers:objectViewer gs://$BUCKET_NAME
}
```

Fig. 8. Function (upload_to_gcs) uploads all resources to GCS bucket

7. **Begin Monitoring** CPU and memory usage in an infinite loop.

- **Resource Monitoring & Threshold-Based Actions**

☑ **Resource Monitoring**

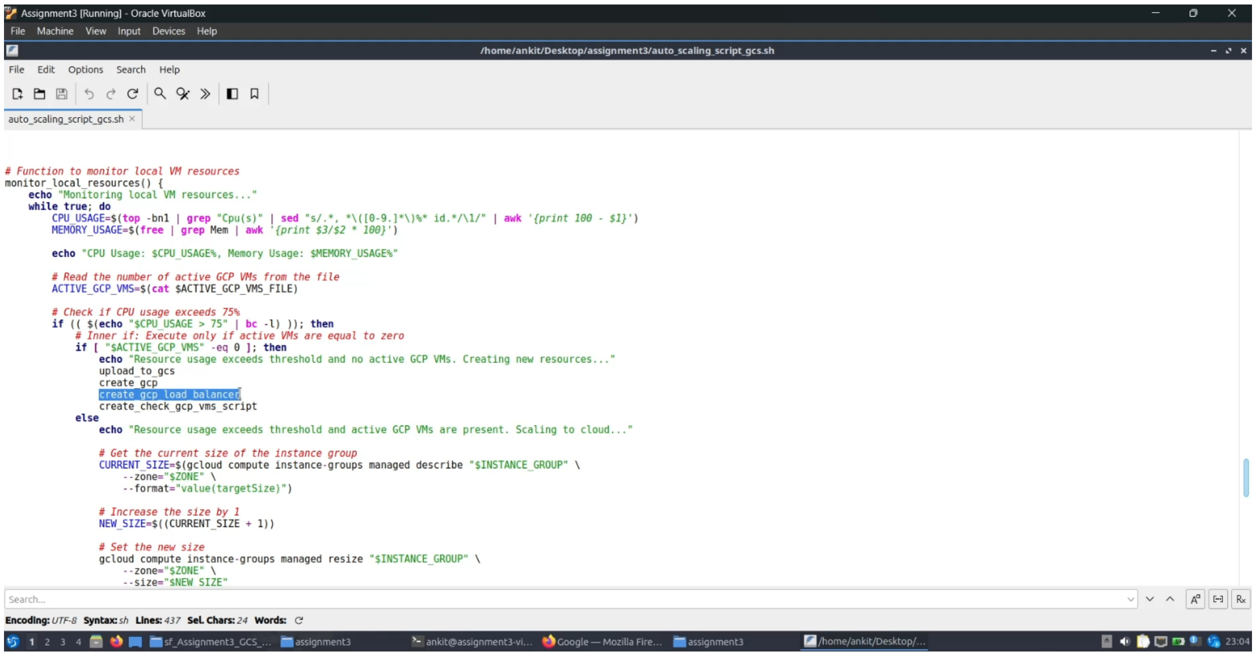
- The script continuously checks CPU and memory usage using standard Linux utilities (top, free). Fig. 9 shows the function (monitor_local_resources) to continuously monitor resources.
- If usage exceeds 75%, it checks how many GCP VMs are active.

☑ **Auto-Scaling Actions**

- **Case 1**: If ACTIVE_GCP_VMS=0, the script:
  1. Uploads the local web content to a GCS bucket.
  2. Creates an **Instance Template** on GCP.
  3. Creates a **Managed Instance Group (MIG)** based on that template.
  4. Sets an auto-scaling policy for GCP (target CPU utilization = 60%).
  5. Configures a **Load Balancer** with health checks and firewall rules.
  6. Updates the local Apache config to forward traffic to the GCP Load Balancer IP if at least one GCP VM is active.
- **Case 2**: If ACTIVE_GCP_VMS >= 1, the script increments the MIG size by 1 to handle additional load.

☑ Active GCP VMs Tracking

- A cron job (check_gcp_vms.sh) runs every minute to:
    1. Query GCP for running instances.
    2. Update a local file (/tmp/active_gcp_vms.txt).
    3. Dynamically rewrite the local Apache config to forward traffic if active_gcp_vms > 0.



```
# Function to monitor local VM resources
monitor_local_resources() {
    echo "Monitoring local VM resources..."
    while true; do
        CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print 100 - $1}')
        MEMORY_USAGE=$(free | grep Mem | awk '{print $3/$2 * 100}')

        echo "CPU Usage: $CPU_USAGE%, Memory Usage: $MEMORY_USAGE%"

        # Read the number of active GCP VMs from the file
        ACTIVE_GCP_VMS=$(cat $ACTIVE_GCP_VMS_FILE)

        # Check if CPU usage exceeds 75%
        if (( $(echo "$CPU_USAGE > 75" | bc -l) )); then
            # Inner if: Execute only if active VMs are equal to zero
            if [ "$ACTIVE_GCP_VMS" -eq 0 ]; then
                echo "Resource usage exceeds threshold and no active GCP VMs. Creating new resources..."
                upload_to_gcs
                create_gcp
                create_gcp_load_balancer
                create_check_gcp_vms_script
            else
                echo "Resource usage exceeds threshold and active GCP VMs are present. Scaling to cloud..."

                # Get the current size of the instance group
                CURRENT_SIZE=$(gcloud compute instance-groups managed describe "$INSTANCE_GROUP" \
                    --zone="$ZONE" \
                    --format="value(targetSize)")

                # Increase the size by 1
                NEW_SIZE=$((CURRENT_SIZE + 1))

                # Set the new size
                gcloud compute instance-groups managed resize "$INSTANCE_GROUP" \
                    --zone="$ZONE" \
                    --size="$NEW_SIZE"
```

Assignment3 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
/home/ankit/Desktop/assignment3/auto_scaling_script_gcs.sh
File Edit Options Search Help
auto_scaling_script_gcs.sh
Search...
Encoding: UTF-8 Syntax: sh Lines: 437 Sel. Chars: 24 Words:
sf_Assignment3_GCS_... assignment3 ankit@assignment3-vi... Google — Mozilla Fire... assignment3 /home/ankit/Desktop/...
23:04

Fig. 9. Function to monitor resource usage and above-mentioned cases of auto-scaling.

• Deployment of the Sample Application

☑ Local VM

- A basic index.php is placed in /var/www/html/ which displays a greeting and the hostname.
- The script automatically configures Apache to serve this file.

☑ GCP VMs

- The instance template's **startup script** installs Apache and PHP, downloads the same web content from the GCS bucket, and serves it.
- Each instance in the MIG will show a similar greeting (with hostname), making it easy to identify which server is responding.

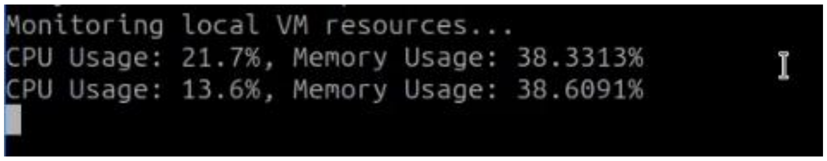
❖ Testing the Auto-Scaling Setup

1. **Run the Main Script**

./auto_scaling_script_gcs.sh

2. **Monitor the Output**

- The script prints CPU/Memory usage every 90 seconds.



```
Monitoring local VM resources...
CPU Usage: 21.7%, Memory Usage: 38.3313%
CPU Usage: 13.6%, Memory Usage: 38.6091%
```

Fig. 10. Custom script monitoring CPU and Memory

3. **Apply Load / Stress**

- To artificially raise CPU usage above 75%, used a tool like stress. Run below commands:

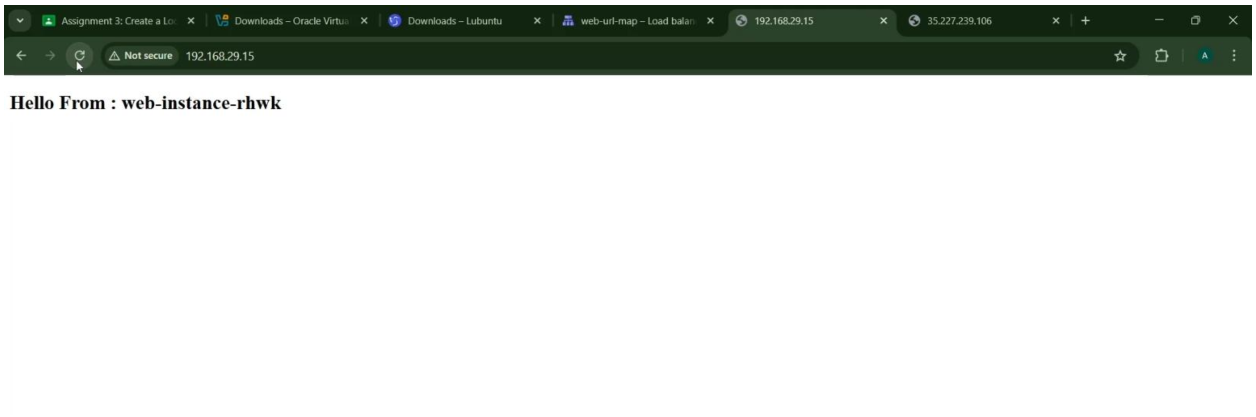
sudo apt install stress

stress -c 2 --timeout 60

- This quickly pushed CPU usage above 75%.

4. **Observe the Autoscaling Behavior**

- When the threshold is crossed, the script uploads content to GCS (Google Cloud Storage), creates the MIG (Managed Instance Group) and LB (Load Balancer), or increments MIG size if it already exists.
- The local Apache configuration rewrites traffic to the GCP LB once at least one GCP VM (Virtual Machine) is active (As shown in below Fig. 11).



Hello From : web-instance-rhwk

Fig. 11. On request from a client to Local VM, output is served from GCP VM

5. Verify in GCP Console

1. Go to **Compute Engine** → **Instance groups** to see new instances being created.
2. Check the **Load Balancer** page to see the forwarding rule and backend services.
3. Confirm that the web application is served from the GCP instance(s).

❖ Cleaning Up Resources

To save costs, delete the created resources:

1. Make the script executable by running:

```
chmod +x delete_resources_gcs.sh
```

2. Run the Delete Script: Execute the provided script to remove all resources:

```
./delete_resources_gcs.sh
```

❖ Source Code Repository

The source codes used for this implementation are available at the following repository:

- **GitHub**: https://github.com/Ankit-IITJ/VCC_Assignment3.git

❖ Recorded Video Link

The setup process is being demonstrated in a recorded video [**Link**], which showcases auto-scaling and security configurations.

➢ References

- https://cloud.google.com/sdk?hl=en
- https://cloud.google.com/cli?hl=en

- https://cloud.google.com/compute/docs/autoscaler
- https://cloud.google.com/firewall/docs/firewalls
- https://cloud.google.com/iam/docs/overview