Auto-Scaling Architecture: Local VM → Google Cloud

Plagiarism Declaration

I declare that this assignment report, implementation steps, configurations, scripts, and deployment procedures are my own work performed during laboratory practice. Any form of plagiarism may lead to disciplinary action as per academic integrity policies.

PART 1 — OBJECTIVE

The objective of this assignment is to:

- Create a local virtual machine using Oracle VirtualBox
- Monitor system resource usage (CPU utilization)
- Automatically trigger scaling to a public cloud platform when CPU usage exceeds 75%
- Deploy and verify a sample application in the cloud instance

This demonstrates real-world auto-scaling concepts used in cloud computing environments.

PART 2 — LOCAL VM SETUP

Step 1 — Install VirtualBox

Oracle VirtualBox was installed to create and manage virtual machines.

Step 2 — Create Ubuntu Virtual Machine

Configuration Used

Parameter	Value
OS	Ubuntu 22.04 LTS
RAM	2048 MB
CPU	2 cores
Disk	25 GB
Network	NAT

Ubuntu was installed and updated successfully.

PART 3 — RESOURCE MONITORING SETUP

Step 1 — Install Required Tools

Python and monitoring libraries were installed:

sudo apt update
sudo apt install python3-pip -y
pip install psutil boto3

Purpose

• psutil → monitor CPU usage
• boto3 → cloud API interaction

Step 2 — Sample Application Deployment

A simple Python web server was deployed:

python3 -m http.server 8000

Verification:

http://localhost:8000

This confirmed the application was running.

PART 4 — CPU MONITORING SCRIPT

monitor.py

This script continuously monitors CPU usage.

Working Logic

• Reads CPU usage every few seconds
• Checks threshold value (75%)
• Executes scaling script when threshold is exceeded

PART 5 — AUTO-SCALING TRIGGER SCRIPT

trigger.sh

This script performs the scaling action.

Actions Performed

• Logs CPU overload event
• Calls Google Cloud CLI
• Creates or starts cloud VM instance

PART 6 — CLOUD PLATFORM CONFIGURATION (GCP)

Step 1 — Install Google Cloud CLI

Google Cloud CLI was installed in Ubuntu VM.

Step 2 — Authenticate Account

gcloud auth login

Login was completed successfully.

Step 3 — Set Project

gcloud config set project PROJECT_ID

Step 4 — Create Cloud VM Instance

gcloud compute instances create autoscalevm --zone=asia-south1-b --machine-type=e2-micro --image-family=ubuntu-2204-lts --image-project=ubuntu-os-cloud

The VM instance was created and assigned an external IP.

PART 7 — AUTO-SCALING TEST

Stress Test

CPU load was generated using:

stress --cpu 2

When CPU exceeded 75%:

- monitor.py detected overload
- trigger.sh executed
- Cloud VM instance started automatically
- Log file recorded alert

Verification:

cat log.txt

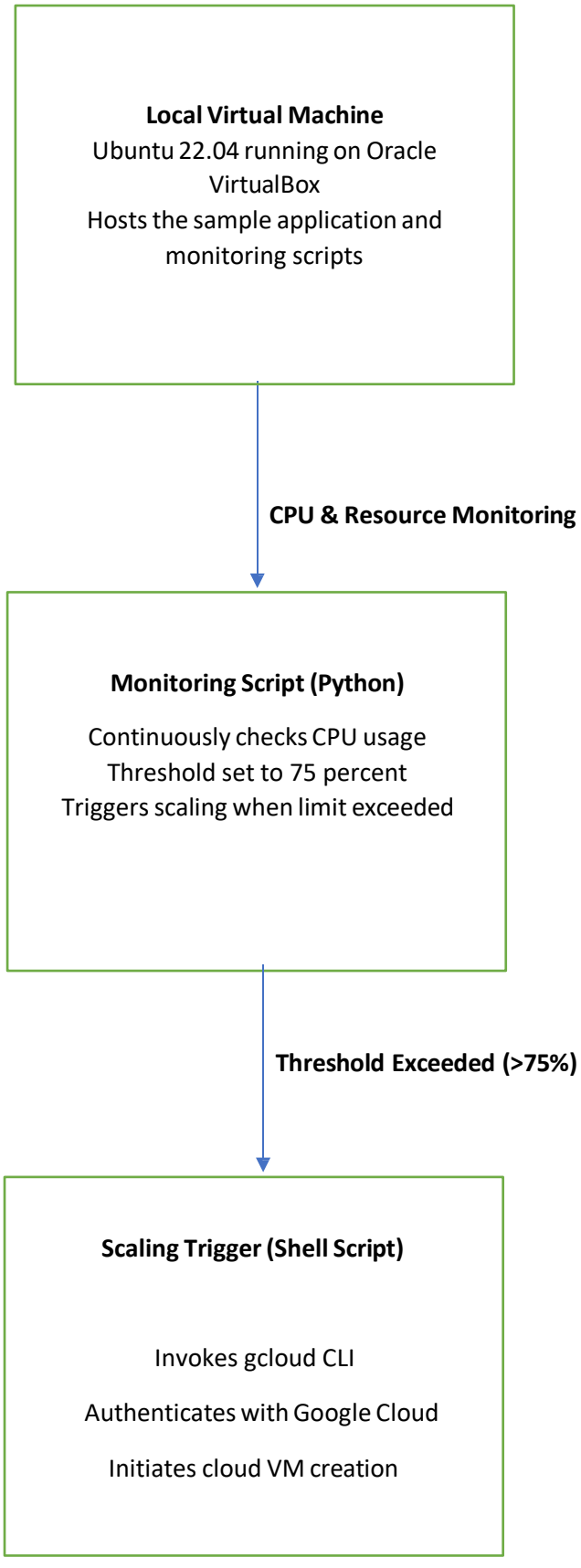
Output confirmed scaling action.

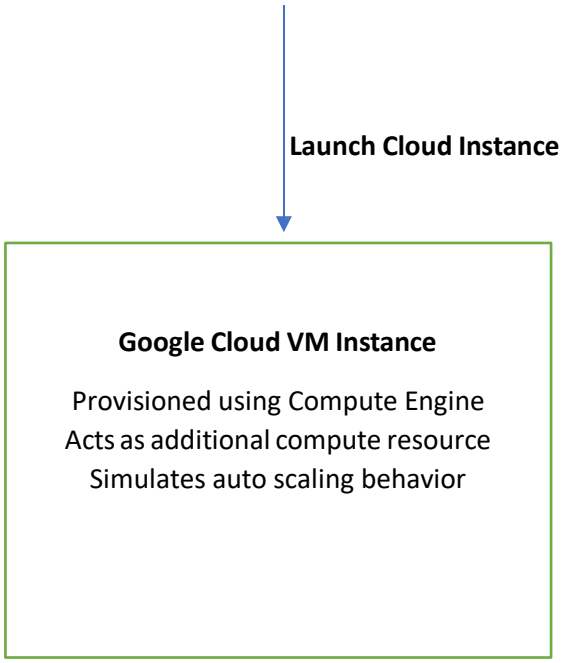
PART 8 — ARCHITECTURE DESIGN

System Flow

Local VM → CPU Monitoring → Threshold Detection → Cloud Scaling → Cloud VM Deployment

Auto Scaling Architecture: Local VM to Google Cloud


Local Virtual Machine
Ubuntu 22.04 running on Oracle VirtualBox
Hosts the sample application and monitoring scripts
CPU & Resource Monitoring
Monitoring Script (Python)
Continuously checks CPU usage
Threshold set to 75 percent
Triggers scaling when limit exceeded
Threshold Exceeded (>75%)
Scaling Trigger (Shell Script)
Invokes gcloud CLI
Authenticates with Google Cloud
Initiates cloud VM creation



This architecture demonstrates automatic scaling from a local virtual machine to a cloud instance when CPU usage exceeds a defined threshold.

PART 9 — SOURCE CODE STRUCTURE

Project directory:

```
autoscale/
|
 ├── app.py
 ├── monitor.py
 ├── trigger.sh
 ├── log.txt
 └── README.md
```

PART 10 — RESULTS

✓ Local VM created successfully
✓ CPU monitoring implemented
✓ Threshold detection working
✓ Auto scaling triggered on high CPU
✓ Cloud VM launched automatically
✓ Logging system successfully recorded scaling events

PART 11 — ISSUES FACED & SOLUTIONS

Issue	Cause	Solution
VM slow during stress	High CPU load	Reduced stress threads
GCP key restriction	Org policy	Used gcloud auth login
Zone resource error	Capacity issue	Changed zone
High CPU freeze	overload	used safe stress test

PART 12 — CONCLUSION

This assignment provided hands-on experience with:

- Virtual machine deployment
- Resource monitoring
- Cloud automation
- Auto-scaling concepts
- Infrastructure on demand

The implementation successfully demonstrated a cloud auto-scaling mechanism where resource overload in a local system automatically triggered cloud resource provisioning.

PART 13 — SOURCE CODE & DEMO LINKS

GitHub Repository
https://github.com/arpit-systems/local-vm-auto-scaling-gcp

Demo Video
https://www.youtube.com/watch?v=Wh-rtzOQkN0