Google Cloud deployment runbook \(Docker Compose app on a small VM\)

This runbook captures the **prerequisites** and the **exact commands** to deploy a multi-container Docker Compose application (UI + API + DB) to Google Cloud using a low-cost Compute Engine VM (fits well within trial credits).

It is written for **Windows PowerShell**.

Prerequisites

1\) Accounts \& billing

• A Google Cloud account with **Billing enabled** (trial credits are OK).

• A Google Cloud **Project ID** (example used below: `iron-tea-482716-v0`).

2\) Local tools \(Windows\)

Install Git

• Install Git for Windows and ensure `git` is available in PowerShell.

Verify:

```
git --version
```

Install Docker Desktop (optional for local run)

• Install Docker Desktop if you want to run locally. (Not required to deploy to GCP VM.)

Install Google Cloud CLI

Install Google Cloud CLI.

After installation, verify `gcloud` is callable.

On some Windows setups, the easiest is to call it via full path:

```
$gcloud = "$env:LOCALAPPDATA\Google\Cloud SDK\google-cloud-sdk\bin\gcloud.cmd"
& $gcloud --version
```

3\) Your application source code \(Git repo\)

You need a Git repo URL for your app, for example:

• `https://github.com/leenaparik/shivam-demo.git`

One\-time setup \(per machine / per account\)

Login to Google Cloud

```
$gcloud = "$env:LOCALAPPDATA\Google\Cloud SDK\google-cloud-sdk\bin\gcloud.cmd"
& $gcloud auth login
& $gcloud auth list
```

\(Optional\) List projects you have access to

```
& $gcloud projects list --format="table(projectId,name)" --limit=50
```

Deployment steps \(the commands\)

0\) Choose your project \+ zone

```
$PROJECT_ID = "iron-tea-482716-v0"
$ZONE = "us-central1-a"
```

1\) Configure gcloud to use the project/zone

```
& $gcloud --quiet config set project $PROJECT_ID
& $gcloud config set compute/zone $ZONE
& $gcloud config list
```

2\) Enable Compute Engine API

```
& $gcloud --quiet services enable compute.googleapis.com
```

3\) Create a firewall rule for your app ports

Recommended: restrict to your IP (safer).

If you *must* open to all, use `0.0.0.0/0`.

Open ports:

• UI: `8080`

• API: `5000`

```
$FW_RULE = "shivam-demo-allow-web"
$TAGS = "shivam-demo"
$SOURCE_RANGES = "0.0.0.0/0"
& $gcloud --quiet compute firewall-rules create $FW_RULE `
```

```
  --direction=INGRESS `
  --priority=1000 `
  --network=default `
  --action=ALLOW `
  --rules=tcp:8080,tcp:5000 `
  --source-ranges=$SOURCE_RANGES `
  --target-tags=$TAGS
```

If the rule already exists and you want to update allowed ports:

```
& $gcloud --quiet compute firewall-rules update $FW_RULE --rules=tcp:8080,tcp:5000
```

4\) Create a small VM \(e2\-micro\) and deploy via startup script

This approach uses a **startup script** to:

• install Docker + docker-compose

• clone your Git repo

• create `.env` with generated keys

• run `docker-compose up -d --build`

In this repo, the startup script file is:

• `gcp-startup.sh`

Create the VM:

```
$VM = "shivam-demo-vm"

& $gcloud --quiet compute instances create $VM `
  --machine-type=e2-micro `
  --image-family=ubuntu-2204-lts `
  --image-project=ubuntu-os-cloud `
  --boot-disk-size=20GB `
  --tags=$TAGS `
  --metadata-from-file=startup-script=gcp-startup.sh
```

5\) Get the VM external IP

```
$IP = & $gcloud compute instances describe $VM --zone $ZONE --format="value(networkInterfaces[0].accessConfi
$IP
```

Open the app:

• UI: `http://$IP:8080`

• API health: `http://$IP:5000/api/health`

6\) Verify endpoints from PowerShell

```
curl.exe -s "http://$IP:5000/api/health"
curl.exe -s -I "http://$IP:8080/" | findstr /I "HTTP"
```

Updating an existing deployment

If you changed code in GitHub and want the VM to redeploy, update the startup script metadata and reboot:

```
& $gcloud --quiet compute instances add-metadata $VM --zone $ZONE --metadata-from-file startup-script=gcp-st
& $gcloud --quiet compute instances reset $VM --zone $ZONE
```

Basic troubleshooting commands

Check VM status

```
& $gcloud compute instances describe $VM --zone $ZONE --format="value(status)"
```

Read startup script logs \(serial port output\)

```
& $gcloud compute instances get-serial-port-output $VM --zone $ZONE --start=0
```

SSH to the VM and check containers

```
& $gcloud compute ssh $VM --zone $ZONE --command "sudo docker ps"
& $gcloud compute ssh $VM --zone $ZONE --command "sudo docker ps -a"
```

Cost control \(stay within free/trial credit\)

Stop the VM when not using it

```
& $gcloud --quiet compute instances stop $VM --zone $ZONE
& $gcloud compute instances describe $VM --zone $ZONE --format="value(status)"
```

While stopped, you may still be billed for:

• **Persistent disks** (boot disk)

• **Reserved static IPs** (if you created any)

Delete the VM \(to remove most costs\)

```
& $gcloud --quiet compute instances delete $VM --zone $ZONE
```

Delete the firewall rule \(optional cleanup\)

```
& $gcloud --quiet compute firewall-rules delete $FW_RULE
```