AI-04750 GCP High Cost Analysis

My first month bill of my GCP account is now hovering around $90, I have a 3 year CUD, US West1-c E2 Standard-4 4vcpu, 16GB RAM and 80GB. Is the billing displayed the non-discounted amount??
Per our conversation, I was suppose to be in under $50 with the discount. Can you explain the billing? How can I lower my monthly cost? Am I allowed to downsize my VM to 2cpu and 8GB RAM and back to 30 GB space? I now concentrating on Docmost and small database MariaDB and PostGres. Help me downsize.

Docker / System Commands

Bash

```
docker compose ls
```

Bash

```
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Status}}\t{{.Ports}}"
```

Bash

```
docker ps -a --format "table {{.Names}}\t{{.Image}}\t{{.Status}}\t{{.Ports}}"
```

Bash

```
docker stats --no-stream
```

Bash

```
docker system df
```

Bash

```
docker inspect openproject-web-1 \
  --format '{{ index .Config.Labels "com.docker.compose.project.working_dir" }}'
```

Bash

```
docker inspect -f \
'{{ index .Config.Labels "com.docker.compose.project" }}' \
openproject-web-1
```

Bash

```
docker compose stop
```

Bash

```
docker compose start
```

Bash

```
free -h
```

Bash

```
df -h
```

Docker Container Management

Bash

```
docker stop <container_name>
```

Bash

```
docker start <container_name>
```

Bash

```
docker restart <container_name>
```

Bash

```
docker compose up -d
```

Bash

```
docker compose down
```

> Avoid unless intentionally removing volumes:
>
> Bash
>
> ```
> docker compose down -v
> ```

Google Cloud CLI Commands

Bash

```
gcloud compute instances list
```

Bash

```
gcloud compute instances describe <INSTANCE_NAME> \
  --zone=<ZONE>
```

Bash

```
gcloud compute machine-types describe e2-standard-4 \
  --zone=us-west1-c
```

Bash

```
gcloud compute disks list
```

Bash

```
gcloud compute addresses list
```

Bash

```
gcloud billing accounts list
```

Database / Backup-Related Checks

Bash

```
docker volume ls
```

Bash

```
docker volume inspect <VOLUME_NAME>
```

Bash

```
docker logs <container_name>
```

Bash

```
docker inspect <container_name>
```

Bash

```
docker exec -it <container_name> /bin/bash
```

Bash

```
docker exec -it <container_name> /bin/sh
```

GCP Cost Review & VM Downsizing Plan

Rigel, let's work through this carefully. Your GCP bill hovering around $90 for the first month, despite your 3-year Committed Use Discount (CUD), deserves an investigation before you start changing your infrastructure.

Based on your current setup:

|
Item
|
Current configuration
|
| --- | --- |
|
Cloud region
|
`us-west1-c` (Oregon)
|
|
VM machine type
|
`e2-standard-4`
|
|
vCPUs
|
4
|
|
Memory
|
16 GB
|
|
Boot disk

|

80 GB

|

|

Commitment

|

3-year CUD

|

|

Main workloads

|

Docmost, MariaDB, PostgreSQL

|

|

Cost target

|

Under $50/month

|

1. Is the $90 bill the non-discounted amount?

Possibly for some line items, but we cannot assume that the entire $90 is the undiscounted price.

Google Cloud billing can show several different values depending on which report or billing view you're looking at:

- List cost: The price before applicable discounts or credits.
- Discounts: CUD discounts and other applicable pricing adjustments.
- Net cost: The amount after discounts and credits.
- Forecast: An estimate of the full month based on usage so far.

The important distinction is that a CUD does not necessarily discount every component of your bill.

Your commitment might apply to the eligible compute usage, while storage, snapshots, IP addresses, network egress, and other services can be billed separately.

What we need to verify

Before downsizing, identify whether the $90 includes:

Compute Engine VM

The e2-standard-4 usage and whether your CUD is applied.

Persistent Disk

80 GB disk charges, disk type, and any additional volumes.

Network and IP

Egress, external IP charges, or other networking-related fees.

CUD commitment

Whether the commitment is resource-based or compute-flexible, and its billing scope.

2. Why your $90 bill might be higher than expected

There are several likely explanations. We need to distinguish what applies to your account from what is merely possible.

Scenario A: Your CUD is active, but you committed to more resources than you need

A resource-based CUD commits you to specified compute resources, such as vCPUs and memory, in a particular region and machine series. Google applies the discount to eligible usage, but your commitment remains a financial obligation.

Compute Engine

+1

For example, if your commitment covers:

- 4 vCPUs
- 16 GB memory
- E2 machine series
- A particular region

and you downsize to 2 vCPUs and 8 GB memory, you may continue paying for the original commitment, depending on its exact configuration. Downsizing the VM is not the same as reducing or canceling the commitment.

This is especially important because you signed a 3-year CUD. We should inspect the commitment before changing the VM size.

Scenario B: The $90 is a projected or gross billing amount

Your billing dashboard might be showing a cost estimate, list cost, or a total that includes services outside the VM.

Don't use the displayed total alone to conclude that your CUD is missing. Check the actual billing report, discounts, and SKU-level charges.

Scenario C: Additional GCP resources are contributing

Your DockerLab has included services such as Nginx Proxy Manager, databases, and other applications. The containers themselves do not each incur a separate VM charge when running on the same Compute Engine VM, but they can contribute to resource consumption and potentially other service usage.

Check for:

- Persistent Disk storage
- Static external IP addresses
- Cloud Storage or snapshots
- Network egress
- Other enabled GCP services

The CUD does not automatically cover every billable service.

3. Can you downsize from 4 vCPU / 16 GB to 2 vCPU / 8 GB?

Yes. You can resize your Compute Engine VM to a smaller machine type, subject to the machine type being available in your zone and your workload's requirements.

Your target would likely be:

Current

E2 Standard-4

CPU

4 vCPU

Memory

16 GB

Proposed

E2 Standard-2

CPU

2 vCPU

Memory

8 GB

The E2 standard-2 configuration is a reasonable candidate for a small, lightly used Docmost installation with MariaDB and PostgreSQL. However, actual memory usage, database size, concurrent users, and background tasks determine whether it will be sufficient.

My recommendation for your workloads

Given your current focus on Docmost and small databases:

|
Workload

|

Initial approach

|
| --- | --- |
|

Docmost

|

Keep running on the main VM

|
|

MariaDB

|

Keep running; monitor memory

|
|
PostgreSQL
|
Keep running; monitor memory
|
|
Nginx Proxy Manager
|
Keep if required for routing
|
|
Portainer
|
Optional; keep if useful
|
|
Other applications
|
Stop or remove if no longer needed
|

I would first measure resource usage and then downsize rather than immediately moving to an even smaller machine.

Important: 2 vCPU / 8 GB RAM is a proposed starting point, not a guarantee that all of your existing containers will operate comfortably at the same time.

4. Can you reduce the disk from 80 GB to 30 GB?

You cannot normally shrink an existing Persistent Disk in place.

A smaller disk would generally require a migration process:

1. Back up your existing data.
2. Create a new 30 GB disk (if the disk type and required capacity support it).
3. Copy or restore the required data to the new disk.
4. Configure the VM to use the new disk.
5. Verify the operating system, Docker data, and databases.

There is a major issue with your proposed 30 GB size: Docker images, container layers, database files, logs, and operating system space can consume it quickly.

For your Docmost + MariaDB + PostgreSQL setup, I would not automatically recommend 30 GB without first inspecting the actual disk usage.

My storage recommendation

|
Disk size
|
Assessment
|
| --- | --- |
|
80 GB
|
Current baseline; investigate actual usage
|
|
50 GB
|
Potential cost-saving target if usage is low
|
|
30 GB
|
Possible for a very small installation, but tight for Docker + databases
|
|
60 GB
|
A possible compromise if 50 GB is unavailable or insufficient
|

The precise cost difference depends on the disk type and provisioned capacity. Storage optimization should be based on actual usage, not simply reducing the number.

5. How to verify your CUD and billing

Let's inspect the billing before making changes. This will help determine whether the original under-$50 target was realistic.

Step 1 — Open the billing report

In Google Cloud Console:

1. Open Billing.
2. Select your billing account.
3. Open Reports.
4. Set the time range to the current month.

5. Group or filter by Service.
6. Inspect Compute Engine and other services.

Look for the difference between gross/list cost, discounts, and net cost.

Step 2 — Check the CUD commitment

Navigate to:

Billing → Commitments (or the Compute Engine commitments view).

Check:

|
Field

|
What to verify

|
| --- | --- |
|
Commitment type

|
Resource-based or compute flexible

|
|
Commitment term

|
3 years

|
|
Region

|
Whether it matches your VM's region

|
|
Machine series

|
E2, if applicable

|
|
Committed vCPU

|
How many are committed

|

|

Committed memory

|

How much memory is committed

|

|

Monthly commitment charge

|

The actual recurring financial obligation

|

Google documents resource-based and compute flexible commitments as different models. The type matters because the flexibility of the commitment differs.

google.com

+1

Please don't cancel or modify the commitment based on the $90 figure alone. We first need to establish exactly what you purchased.

6. How I would lower your monthly cost

Your goal is a small, focused DockerLab rather than a large collection of applications. I would approach it in this order.

1. Verify the bill

First

Confirm CUD discounts and identify all billing line items.

2. Measure VM utilization

Review CPU, RAM, disk usage, and database resource requirements.

3. Downsize compute

Consider E2 standard-2 after backing up and verifying workload needs.

4. Optimize storage

Clean unused Docker data and evaluate a smaller disk through migration.

5. Simplify DockerLab

Run only the services you actively use.

Cost-saving option A — Reduce VM size

Changing the machine type requires stopping the VM. Google documents this as a supported method for changing machine types, and the attached persistent disk data remains intact during a machine-type change.

Compute Engine

+1

Console procedure

1. Go to Compute Engine → VM instances.
2. Select your VM.
3. Click Stop and wait until the VM is stopped.
4. Click Edit.
5. Find the machine configuration.
6. Change the machine type to `e2-standard-2`.
7. Save.
8. Start the VM.
9. Check Docker, Docmost, MariaDB, PostgreSQL, and Nginx Proxy Manager.

The exact menu names may vary slightly with the current Cloud Console interface.

CLI alternative

Once you know your instance name and zone:

Bash

```
gcloud compute instances stop VM_NAME \
  --zone=ZONE

gcloud compute instances set-machine-type VM_NAME \
  --machine-type=e2-standard-2 \
  --zone=ZONE

gcloud compute instances start VM_NAME \
  --zone=ZONE
```

Replace `VM_NAME` and `ZONE` with your actual values. Don't run this until you've confirmed the VM name and zone.

CUD warning: If your existing commitment is specifically for 4 vCPUs and 16 GB of E2 resources, reducing the VM's size could leave some committed resources unused. That means the VM's ordinary consumption charge may decline without a proportional reduction in your total commitment expense.

7. Storage optimization: Don't shrink to 30 GB yet

I would start by checking how much space your Docker environment is actually using.

Run these commands inside your Ubuntu VM

A. Check disk capacity

Bash

```
df -h
```

This shows how much space is used and available on your filesystem.

B. Check Docker disk usage

Bash

```
docker system df
```

This shows usage from images, containers, local volumes, and build cache.

C. Check the largest directories

Bash

```
sudo du -xhd1 / | sort -h
```

Then, if `/var` is large:

Bash

```
sudo du -xhd1 /var | sort -h
```

Don't delete anything based on these results yet. We need to identify what is safe to remove.

D. Check database directories

If you use Docker volumes, inspect them:

Bash

```
docker volume ls
```

You can then identify which volumes belong to Docmost, MariaDB, and PostgreSQL.

Critical: Database volumes contain your data. Never delete or prune Docker volumes without first verifying their contents and having a usable backup.

8. What I recommend for your target configuration

Given your current focus, I would use this as a preliminary target:

Lean DockerLab

Proposed

VM

E2 standard-2

CPU

2 vCPU

RAM

8 GB

Disk

50–80 GB initially

Core services

Docmost

MariaDB

PostgreSQL

NPM

Keep the system lean, monitor resource usage, and avoid reducing storage below the actual operational requirement.

What about a 1 vCPU / 4 GB VM?

I would not choose that as the first target while running Docmost and two databases together. It might work for a very light workload, but it gives you less memory and CPU headroom.

Let's first see whether 2 vCPU / 8 GB meets your needs.

9. How much can you realistically save?

I don't want to give you a false promise that downsizing will reduce your bill to under $50, because your CUD commitment may limit the immediate savings.

There are two separate cost questions:

|
Question

|

Why it matters

|
| --- | --- |
|

What is the cost of running a smaller VM?

|

Helps estimate future resource usage

|
|

What is your fixed CUD obligation?

|

Determines how much of the bill can actually decrease

|

If you have a resource-based CUD for the current configuration, downsizing might not produce the full savings you expect. If you have a flexible commitment, the result may differ.

The first goal is to find the actual commitment charge and the non-commitment charges. After that, we can model the savings accurately.

10. Let's diagnose your GCP account together

Fill in the information below. You can find it in the Google Cloud Console.

GCP Cost & Downsizing Checklist

Enter the information you have. Leave unknown fields blank.

1. CUD type

Resource-based CUD

Compute flexible CUD

Not sure

2. Committed vCPU

3. Committed memory

4. VM name

5. VM zone

6. Disk type

Standard Persistent Disk

Balanced Persistent Disk

SSD Persistent Disk

Not sure

7. Disk usage

8. Billing view

Actual cost

Forecast

List cost / before discounts

Not sure

9. Compute Engine cost this month

10. Persistent Disk cost this month

11. Other costs / services

12. RAM usage

13. Running services

Docmost

MariaDB

PostgreSQL

Nginx Proxy Manager

Portainer

OpenProject

OpenEMR

Inventree

Zammad

WordPress

Other

14. Proposed disk size

Keep 80 GB

60 GB

50 GB

30 GB

15. Backup status

Verified backup exists

Snapshot exists

No backup yet

Not sure

Analyze My GCP Configuration

My bottom line

Rigel, I would downsize the compute to 2 vCPU / 8 GB after verifying utilization, but I would not shrink the disk to 30 GB yet.

The biggest question is your 3-year CUD commitment. If it covers 4 vCPUs and 16 GB memory, reducing the VM may not reduce your fixed commitment expense. Let's confirm that first, then optimize the remaining monthly charges.

For the official documentation:

- Google Cloud: Compute Engine CUD overview
- Google Cloud: Change a VM machine type
- Google Cloud: Resize Persistent Disks

Is my current GCP spending justified as a tuition fee for my skill upgrade. Also, I don't have to run my own Poweredge server at home and the office is so quite now and I can work anywhere.

Yes—your GCP spending can reasonably be viewed as a tuition fee for your skill upgrade

Based on what you've described, your GCP DockerLab is not merely a hosting expense. It is also a personal technology laboratory, a learning environment, and a practical investment in flexibility.

The key distinction is that a tuition fee should produce something valuable: knowledge, skills, completed projects, improved confidence, or a capability you can reuse. Your environment appears to be serving several of those purposes.

I would evaluate the spending through three lenses: learning value, lifestyle value, and financial efficiency.

1. Learning value: You are building practical skills, not just consuming tutorials

Your GCP environment gives you a place to practice with real systems:

- Docker and container orchestration.
- Linux server administration.
- Reverse proxies and secure remote access.
- PostgreSQL and MariaDB.
- OpenProject deployment and administration.
- OpenEMR installation and configuration.
- Monitoring, backups, networking, and troubleshooting.
- Cloud billing and infrastructure management.

This is different from watching a course and forgetting the material. You are repeatedly deploying, breaking, troubleshooting, rebuilding, and managing actual services.

That makes your spending more comparable to a continuing-education expense or a personal laboratory budget than a pure entertainment subscription.

The value is especially relevant because you are interested in technology as an ongoing retirement activity—not necessarily because you need to turn it into a commercial business.

2. Your lifestyle benefit is real

The point you made about the quiet office and being able to work anywhere is significant.

Previously: Physical home server model

PowerEdge homelab

- Physical hardware running at home.
- Electricity consumption.
- Heat and fan noise.
- Hardware maintenance and component failures.
- Dependence on home networking and power.
- Physical location limits where you work.

Now: Cloud DockerLab model

GCP DockerLab

- No need to operate the PowerEdge servers continuously.
- Quieter home or office environment.
- Remote access from different locations.
- A consistent Linux and Docker environment.
- Ability to work from Cebu, Texas, or another location with connectivity.
- Less direct responsibility for physical server hardware.

You are not simply paying for CPU and RAM. You are also paying for convenience, mobility, reduced physical infrastructure, and the freedom to work in a quieter environment.

That benefit is personal, so there is no universal dollar value I can assign to it. But it is legitimate to include it in your evaluation.

3. Is approximately $82 per month justified?

Your earlier billing figure was approximately $82.26 per month. If that level continued for a full year, it would be approximately:

$82.26×12=$987.12\$82.26 \times 12 = \$987.12$82.26×12=$987.12

So the scale is roughly $1,000 per year, assuming the monthly cost remains similar.

I would frame the question this way:

> Am I receiving at least $1,000 per year in learning, convenience, flexibility, and enjoyment from this environment?

You do not need to generate $1,000 in business revenue to justify a personal learning expense. However, it should remain a conscious choice rather than a cost that grows unnoticed.

A practical value test

|
Question

|

Your situation

|
| --- | --- |
|

Am I actively learning?

|

Yes—Docker, databases, OpenProject, OpenEMR, Linux, and cloud infrastructure

|
|

Do I use the environment for meaningful projects?

|

Yes—Docmost and your training lab

|
|

Does it replace or reduce physical-server overhead?

|

Yes, based on your decision not to run the PowerEdge servers

|
|

Does it improve mobility?

|

Yes—you can work from different locations

|
|

Do I enjoy the activity?

|

Your ongoing lab work suggests that this is an important hobby and learning activity

|
|

Is the cost controlled?

|

Needs monthly monitoring, particularly because of the 3-year commitment

|
|

Is the data protected?

|

Must be verified with independent backups

|

My assessment: The expense is defensible as a personal education and technology-lifestyle budget, provided you keep it intentional and protect your data.

That is different from claiming that the investment is guaranteed to pay for itself financially.

4. Your cloud bill is also the cost of convenience

The quiet office and ability to work anywhere may be the most valuable part of your setup.

You have effectively exchanged some physical infrastructure responsibilities for cloud infrastructure costs:

|
You avoid or reduce

|

You accept

|
| --- | --- |
|

Running PowerEdge hardware

|

Monthly cloud commitment and usage charges

|
|

Fan noise and heat

|

Dependence on cloud availability and internet

|
|

Hardware maintenance

|

Cloud configuration and billing management

|
|

Physical-location limitations

|

Cloud security and access-management responsibility

|
|

Some electricity consumption

|

Recurring GCP charges

|
|

Local hardware troubleshooting

|

Provider-level infrastructure dependency

|

This is not necessarily a cost-minimization strategy. It is a trade-off between ownership and convenience.

For someone who enjoys homelab hardware and wants maximum control, local infrastructure may be attractive. For someone who wants to work quietly and remotely, cloud infrastructure can be worth the recurring cost.

Your preference appears to be shifting toward the latter for this particular lab.

5. The commitment deserves one extra financial precaution

Your resource-based CUD is active from September 9, 2026, through September 9, 2029. Google Cloud documentation confirms that resource-based commitments have a monthly commitment fee even when the committed resources are not fully used.

Google Cloud

+1

Therefore, treat the commitment as a three-year infrastructure decision, not merely a month-to-month hosting subscription.

I would create a simple monthly review:

- Total GCP bill.
- Compute commitment fee.
- Compute usage outside the commitment.
- Storage and network charges.
- Whether you used the lab meaningfully that month.

- Any new learning milestone or completed project.

The purpose is not to force every month to produce measurable output. Some months will be experimentation, maintenance, or downtime. The purpose is to ensure that you remain aware of what you are paying for.

6. Establish a personal "tuition budget"

A reasonable approach would be to treat your DockerLab like a continuing-learning subscription.

For example:

DockerLab Tuition Framework

Personal policy

Target monitoring range

About $80–$100/month

Keep the lab if it continues to support meaningful learning and personal enjoyment.

Review the bill monthly rather than obsessing over every small daily charge.

Avoid adding services simply because the VM has available capacity.

Use the existing capacity for OpenProject, OpenEMR, databases, and other planned training.

Reassess at milestones such as three, six, and twelve months.

The range above is a budgeting example, not a claim that your future GCP bill will remain within it. Your actual commitment fee and additional charges should be monitored separately.

7. The one area I would not compromise: backups

Because your AI notes are in Docmost, the value of the environment depends heavily on data preservation.

Your cloud VM is not a substitute for a backup strategy. I would maintain at least:

1. Primary Docmost data on the GCP VM.
2. Independent backup of the Docmost database and uploaded files.
3. A backup location separate from the VM, ideally in another environment or storage provider.
4. Periodic restoration testing to verify that the backup is usable.

You do not need to run a PowerEdge server continuously to maintain backups. The important thing is that your backup does not depend entirely on the same VM, disk, and failure domain as your primary data.

My overall perspective

Yes, your current GCP spending can be justified as a tuition fee for your skill development and a lifestyle upgrade.

The justification rests on several concrete benefits:

- You are developing practical technical skills.
- You have a persistent environment for experimentation.
- You retain OpenProject and OpenEMR for future training.
- Docmost supports your personal knowledge management.
- You have reduced the need to operate noisy physical servers.
- You can work from different locations.

- You are learning cloud cost management as part of the experience.

The important boundary is this:

> Treat the GCP bill as a deliberate personal investment in learning and freedom—not as an expense that must be minimized at all costs, but also not as an expense that should be ignored.

For your stated lifestyle, a quiet office, remote accessibility, and a reusable learning environment may be worth more than the raw cost difference between cloud hosting and running the PowerEdge server. The decision becomes stronger when you maintain backups, review billing monthly, and continue using the environment for meaningful projects.

I want to work on OpenProject and OpenEMR as well, hence I kept it, Do I need to delete those that I stopped? I still want to play with it later but not now. Docmost is a must, all my AI Notes are in there. I also needs training on DB and OpenProject and OpenEMR when I get to Cebu. I think 4vcpu, 16GB RAM is not overkill? How to optimize my billing?

Your DockerLab: Keep It, Optimize the Cost

No, you do not need to delete the OpenProject, OpenEMR, or database containers you stopped. Given your plans to continue training and experimenting when you return to Cebu, deleting them would create unnecessary work and could risk losing configurations, volumes, or application data.

Your objective should be:

> Preserve the complete learning environment, keep Docmost continuously available, and reduce unnecessary infrastructure costs without sacrificing your 4-vCPU/16-GB development capability.

1. What to keep versus what to stop

|
Component
|
Recommendation
|
Reason
|
| --- | --- | --- |
|
Docmost
|
Keep running
|
Your AI notes are important and you need ongoing access
|
|
PostgreSQL
|
Keep running

|

Likely required by Docmost and potentially other applications

|

|

MariaDB

|

Keep running for now

|

Useful for database training and possible application dependencies

|

|

OpenProject

|

Keep installed; stop when not training

|

You want to resume later

|

|

OpenEMR

|

Keep installed; stop when not training

|

You want to use it for future clinical-system learning

|

|

phpMyAdmin / pgAdmin

|

Keep installed; run when needed

|

Useful for database training, but not necessarily required 24/7

|

|

Portainer

|

Keep if you use it

|

Useful for managing Docker remotely

|
|

Nginx Proxy Manager

|

Keep if needed

|

May be required for access to Docmost and other services

|
|

Previously stopped applications

|

Do not delete yet

|

Preserve your learning environment and configurations

|

Stopping a container is not the same as deleting it. Stopped containers generally retain their writable container layer and configuration, while persistent application data should be stored in Docker volumes or bind mounts. You should still verify your backup strategy before relying on this distinction.

2. Is 4 vCPU / 16 GB RAM overkill?

For your intended learning environment, no—it is a reasonable configuration. You are not running just one lightweight application. Your DockerLab includes:

- Docmost, which you want available continuously.
- OpenProject for project-management training.
- OpenEMR for clinical-system training.
- PostgreSQL and MariaDB for database learning.
- Management and administration tools such as Portainer, pgAdmin, and phpMyAdmin.
- Reverse proxy and other supporting services.

However, there are two separate questions:

1. Is 4 vCPU / 16 GB appropriate for running these applications? Yes, it gives you room for simultaneous experimentation, database work, and application testing.

2. Is keeping that VM running 24/7 the cheapest way to operate it? Not necessarily. Your previous `docker stats` snapshot showed relatively low CPU activity, while OpenProject's web service was using substantial memory. That indicates you should optimize the workload and uptime before automatically downsizing the VM.

I would not immediately reduce your VM to 2 vCPU / 8 GB while you are keeping Docmost, OpenProject, OpenEMR, and database services available. First establish your actual memory usage under a realistic training workload.

Also, the `15.61 GiB` limit shown for each container does not mean every container has reserved 15.61 GiB. It generally indicates that the container can see the host's available memory because no smaller container memory limit has been configured.

3. Your biggest billing issue: the VM's billing model

Your previous billing showed approximately $82.26, with Compute Engine accounting for most of the amount. Stopping individual Docker containers can reduce resource pressure, but it generally does not reduce the hourly cost of a running Compute Engine VM.

The important question is whether your 3-year commitment is covering the VM and whether it is a resource-based or another type of commitment.

Google states that resource-based commitments involve paying for the committed resources throughout the commitment term, even when you do not fully use them. Therefore, downsizing or stopping the VM may not reduce your commitment payment in the way you expect.

Google Cloud Documentation

+1

Do not make a machine-type change until we verify the commitment.

4. Recommended billing optimization plan

Priority 1 — Verify your commitment before resizing

In Google Cloud Console:

1. Open Billing.
2. Select the correct billing account.
3. Go to Cost management → Commitments or the Compute Engine commitment view.
4. Check:
   - Commitment type.
   - Machine series covered.
   - Region.
   - Committed vCPUs.
   - Committed memory.
   - Monthly commitment charge.
   - Commitment start and end dates.
   - Utilization.

Your VM is in `us-west1`, so the region is particularly important. Resource-based commitments are tied to a machine series and region, and the commitment itself cannot simply be transferred to another project, type, or region.

Google Cloud Documentation

+1

The result we need: Determine whether your commitment covers the current `e2-standard-4` and whether moving to `e2-standard-2` would reduce your actual monthly bill or merely leave you paying for unused committed resources.

Priority 2 — Keep the VM, but operate applications selectively

I recommend this operating model:

Always available

Keep running

- Docmost
- Its required database and Redis services
- Nginx Proxy Manager, if needed for remote access
- Tailscale or your chosen secure remote-access method

  Training mode

  Start when needed
- OpenProject
- OpenEMR
- Additional test databases
- phpMyAdmin and pgAdmin
- Other experimental applications

  Preserve

  Do not delete yet
- Docker Compose files
- Environment files
- Docker volumes
- Database backups
- Application configurations

This gives you a permanent Docmost knowledge base while retaining the ability to activate your training environment when you need it.

Priority 3 — Stop applications safely, not delete them

For OpenProject, first locate its Compose project:

Bash

```
docker compose ls
```

You can also inspect the directory used to create the OpenProject container:

Bash

```
docker inspect openproject-web-1 \
  --format '{{ index .Config.Labels "com.docker.compose.project.working_dir" }}'
```

Then change into the displayed directory and stop the Compose services:

Bash

```
docker compose stop
```

This stops the services without intentionally deleting the Compose project, images, or named volumes.

When you return to training:

Bash

```
docker compose start
```

If you have changed the Compose configuration or need to recreate services, use the appropriate `docker compose up -d` command—but do not use `docker compose down -v` unless you deliberately intend to remove the associated volumes and understand the data consequences.

Important dependency caution

Do not stop `postgres-shared` or `mariadb-shared` simply because OpenProject or OpenEMR is inactive. Those databases may serve multiple applications. Identify their consumers first:

Bash

```
docker ps --format \
'table {{.Names}}\t{{.Image}}\t{{.Status}}'
```

Then inspect Compose files and environment configurations to establish which database belongs to which application.

5. Optimize memory without unnecessarily downsizing

Your next step should be to measure the host, not just individual containers.

Run:

Bash

```
free -h
```

Bash

```
docker stats --no-stream
```

Bash

```
docker system df
```

Bash

```
df -h
```

Capture the results in three situations:

| Scenario |

What to measure

|
| --- | --- |
|

Docmost only

|

Idle memory and CPU

|
|

Docmost + OpenProject

|

Memory after opening and using OpenProject

|
|

Docmost + OpenEMR + database training

|

Memory during realistic simultaneous work

|

Watch for:

- Swap usage.
- Available memory.
- PostgreSQL and MariaDB memory.
- OpenProject web and worker memory.
- CPU utilization during actual training.
- Disk usage and database growth.

I would avoid imposing aggressive Docker memory limits initially. A limit that is too low can cause database instability, application crashes, or out-of-memory termination. First measure, then tune individual services.

6. Consider a scheduled operating strategy

Because Docmost is a must-have, you probably want the VM running most of the time. But you can still reduce the workload during periods when you are not training.

Normal mode — Cebu daily use

- Docmost: running.
- Database services required by Docmost: running.
- Reverse proxy and remote access: running.
- OpenProject: stopped unless needed.
- OpenEMR: stopped unless needed.
- Training-only database tools: stopped unless needed.

Training mode

- Start OpenProject.
- Start OpenEMR.
- Start the required database and administration tools.
- Perform your training.
- Stop the training-only services afterward.

This approach saves operational resources, reduces background activity, and avoids unnecessary service exposure. It will not necessarily reduce the Compute Engine VM charge while the VM remains powered on.

If you eventually decide that Docmost does not need 24/7 availability, a scheduled VM shutdown could provide more direct savings. But that would make Docmost unavailable during the shutdown period, so I would not use that as your default configuration.

7. Review the other billing components

Your previous billing indicated that Compute Engine was the primary cost, but you should also check the following:

|
Area

|

What to inspect

|

Optimization

|
| --- | --- | --- |
|

Compute Engine

|

VM runtime and machine type

|

Verify commitment before resizing

|
|

Persistent disk

|

80-GB disk type and usage

|

Avoid reducing it prematurely

|
|

Snapshots

|

Old or redundant snapshots

|

Delete only after confirming backups

|
|

External IP

|

Static IP allocation and attachment

|

Remove unused reserved addresses

|
|

Cloud Monitoring

|

Enabled metrics and retention

|

Check whether you need all monitoring features

|
|

Network egress

|

Traffic leaving Google Cloud

|

Avoid unnecessary large transfers

|
|

Logs

|

Log volume and retention

|

Review excessive application logging

|

Do not reduce your 80-GB disk to 30 GB just to save money at this stage. Your applications and databases may grow, and the storage savings may be relatively small compared with the VM cost. First inspect disk utilization and determine whether your database data, Docker images, logs, and backups are growing unexpectedly.

8. My recommendation for your situation

Decision	Recommendation
Delete stopped applications?	No
Keep Docmost online?	Yes
Keep OpenProject installed?	Yes
Keep OpenEMR installed?	Yes
Keep 4 vCPU / 16 GB for now?	Yes
Immediately resize to 2 vCPU / 8 GB?	No—not before testing

Stop unused application services?

|

Yes, selectively

|

|

Reduce disk size now?

|

No

|

|

Verify CUD details?

|

Highest priority

|

|

Target under $50/month?

|

Possible only after confirming commitment charges and actual cost levers

|

Your ideal architecture

You are building a persistent personal learning and knowledge platform, not merely hosting a website. The design should prioritize:

1. Docmost availability and data protection.
2. Retaining OpenProject and OpenEMR for future training.
3. Database experimentation without accidental data loss.
4. Selective service activation.
5. Billing transparency before changing committed infrastructure.

The next practical step is to obtain these two outputs:

Bash

```
docker compose ls
```

Bash

```
free -h
```

And from Google Cloud, identify the exact commitment type, committed vCPU/RAM quantities, region, and monthly commitment charge. Once those are known, we can determine whether your best optimization is service scheduling, VM resizing, commitment adjustment strategy, or a separate always-on Docmost architecture.