🗓️ Day 3 — Google Cloud Study Jam Progress Report

☁️ Labs Completed:

1. **The Basics of Google Cloud Compute: Challenge Lab**
2. **Deploying the Fancy Store Website using Compute Engine**

🧩 Overview

Day 3 focused on deepening my understanding of **Google Cloud Compute Engine** — from building core infrastructure manually to automating deployments with scalability and reliability features.

I completed two labs:

- A **Challenge Lab** testing my applied skills with virtual machines, storage, and web hosting.
- A **deployment lab** showing how to scale a production-ready e-commerce app ("Fancy Store") using instance templates, managed instance groups, and load balancing.

🚀 Lab 1: *The Basics of Google Cloud Compute – Challenge Lab*

🎯 Objectives

- Create a **Cloud Storage bucket**
- Deploy a **Compute Engine instance**
- Attach a **200GB persistent disk**
- Install and configure an **NGINX web server**
- Verify website accessibility via the **External IP**

🛠️ Tasks Completed

1. Created Cloud Storage Bucket

- Bucket name: <PROJECT_ID>-bucket
- Location: US (multi-region)
- Used to store team files and startup scripts.

2. Created Compute Engine Instance

- Instance name: my-instance
- Series: E2, Machine type: e2-medium
- OS: Debian GNU/Linux 12 (bookworm)
- Enabled firewall for HTTP traffic.

3. Attached Persistent Disk

- Disk name: mydisk (200GB Balanced persistent disk)
- Attached to the VM for data storage.

4. Installed NGINX Web Server

- Used SSH to connect and run:
- sudo apt-get update
- sudo apt-get install -y nginx
- sudo systemctl start nginx
- sudo systemctl enable nginx
- Verified using the external IP — saw "Welcome to nginx!" ✅

💡 **Key Learnings**

- Learned how to **manually provision infrastructure** and manage VM-level networking.
- Understood **persistent disk attachment** and web server configuration on Compute Engine.
- Reinforced the idea of using **Compute Engine for flexible, low-level infrastructure control**.

🏪 **Lab 2: *Deploying the Fancy Store Website using Compute Engine***

🎯 **Objectives**

- Create and configure **Compute Engine instances**
- Build **Instance Templates** for automated scaling
- Deploy **Managed Instance Groups (MIGs)**

- Configure **Health Checks** for autohealing
- Set up an **HTTP(S) Load Balancer**
- Enable **Cloud CDN** for caching and performance

🛠️ Steps Completed

1. **Created Base Instance**
   - Configured Compute Engine VM with startup script to serve the Fancy Store web app.
2. **Created Instance Template**
   - Captured the configuration of the base instance for replication.
3. **Deployed Managed Instance Group**
   - Enabled autohealing and autoscaling based on CPU usage.
4. **Configured Health Checks**
   - Ensured only healthy instances serve requests.
5. **Set Up HTTP(S) Load Balancer**
   - Distributed incoming traffic across the instance group.
6. **Enabled Cloud CDN**
   - Cached static assets globally for faster delivery.

💡 Key Learnings

- Managed Instance Groups simplify **autohealing and scaling** of web servers.
- Load Balancers distribute traffic intelligently and ensure **high availability**.
- Cloud CDN improves **latency and performance** globally.
- The combination of Compute Engine + Load Balancer + CDN provides **production-grade architecture**.

📊 Tools & Services Used

- **Compute Engine**
- **Cloud Storage**
- **Instance Templates**
- **Managed Instance Groups**
- **HTTP(S) Load Balancing**
- **Health Checks**
- **Cloud CDN**

📄 Reflection

Today's labs gave me hands-on experience with both **individual infrastructure setup** (Challenge Lab) and **scalable web architecture** (Fancy Store Lab).
I now understand how organizations deploy and maintain web applications using Compute Engine's **VMs, load balancers, and CDNs** for performance and reliability.

🏁 Outcomes

✅ Completed and verified both labs with 100% progress
✅ Successfully deployed a functioning NGINX web app
✅ Achieved automated scaling and global content delivery setup
✅ Earned Google Cloud badges for both labs