Getting Started with Google Cloud Marketplace – LAMP Stack Deployment

📌 Objective
Learn how to use Google Cloud Marketplace to quickly deploy a pre-configured LAMP stack (Linux, Apache, MySQL, PHP) environment on a Compute Engine instance.

🛠️ Tools & Services Used
- **Google Cloud Marketplace** – for one-click deployments
- **Compute Engine** – virtual machine hosting the LAMP stack
- **Linux (OS)** – base operating system
- **Apache HTTP Server** – web server
- **MySQL** – relational database
- **PHP** – application framework
- **phpMyAdmin** – web-based database administration tool

📂 Steps Performed
1. Signed in to the Google Cloud Console using temporary lab credentials.
2. Navigated to **Marketplace** and searched for the Google Click to Deploy LAMP Stack.
3. Accepted the terms and deployed the LAMP stack with:
   - Zone: chosen deployment zone
   - Machine type: E2 series, `e2-medium`
   - Default settings for other configuration options
4. Enabled required APIs: **Compute Engine API** and **Infrastructure Manager API**.
5. Verified deployment by checking the **lamp-1-vm** instance details.

🎯 Key Learnings
- Google Cloud Marketplace provides pre-configured, production-ready solutions that can be deployed in a few clicks.
- The LAMP stack combines key components (Linux, Apache, MySQL, PHP, phpMyAdmin) into a full web development environment.
- Marketplace deployments automatically enable required APIs and create resources within your project.

🚀 Real-World Application
Using Cloud Marketplace speeds up development by reducing setup time for commonly used environments like LAMP. This approach is valuable for quickly spinning up test environments, hosting simple web applications, or deploying pre-built solutions in enterprise settings.

🧠 Reflection
This lab highlighted how efficient Google Cloud Marketplace can be for provisioning environments without manual configuration. It clarified how services like Compute Engine and Marketplace work together. I want to revisit customizing Marketplace deployments (e.g., adjusting network/firewall settings or scaling VM resources) to better understand how to optimize for production environments.