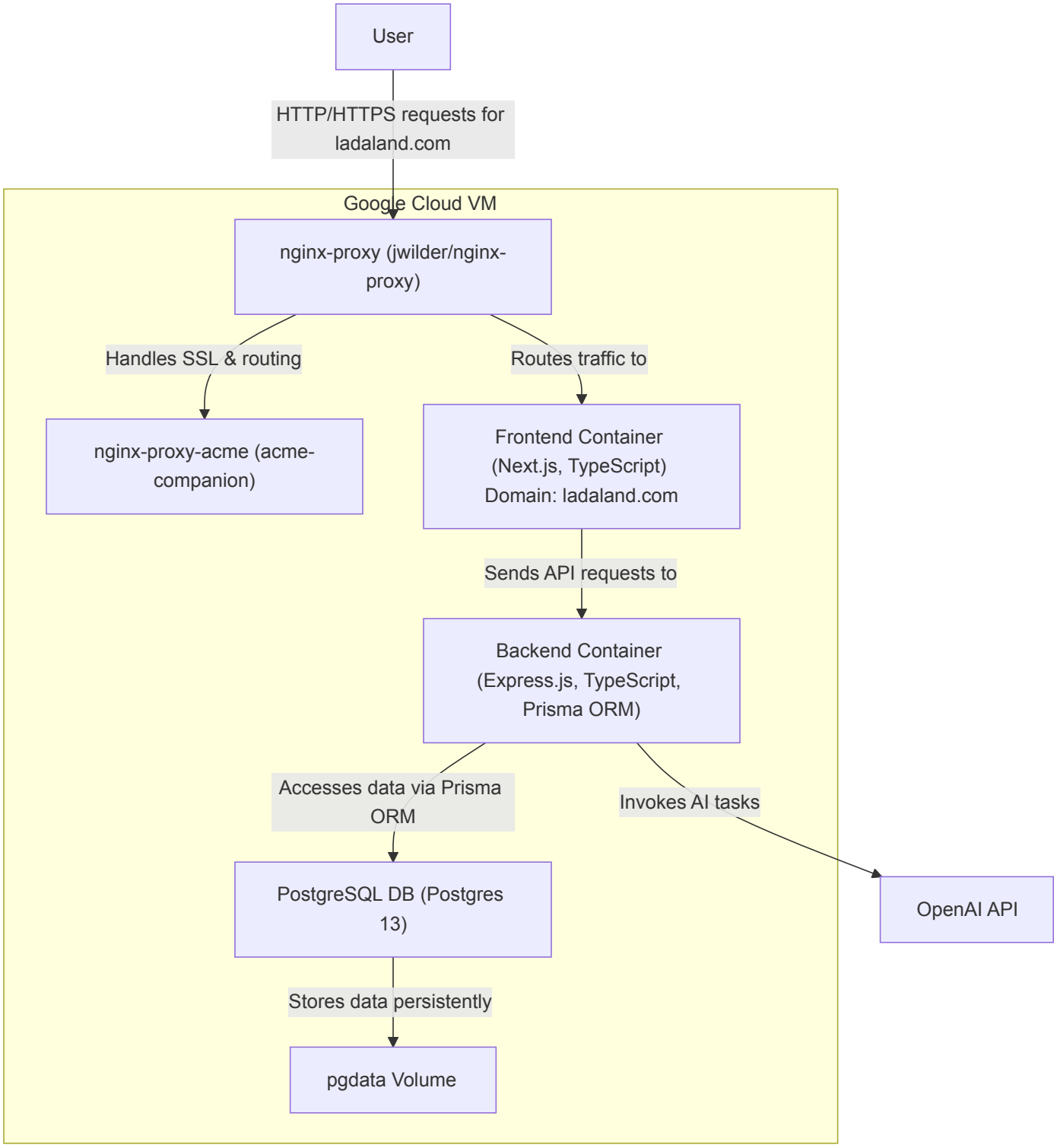
CICD System Design Report

System Design Diagram



System Design Explanation:

- **User Interaction:**
  Users send HTTP/HTTPS requests to **ladaland.com**, which are handled by the **nginx-proxy** running on our GCP

VM.

- **Reverse Proxy & SSL Management:**
  The **nginx-proxy** (jwilder/nginx-proxy) receives the requests and, with SSL certificate management provided by **nginx-proxy-acme** (acme-companion), routes traffic appropriately.
- **Frontend and API Flow:**
  The **Frontend Container** (built with Next.js and TypeScript) serves the website for **ladaland.com**. When the user interacts with the frontend, API requests are sent to the **Backend Container** (built with Express.js, TypeScript, and using Prisma ORM).
- **Backend and AI Tasks:**
  The **Backend Container** handles logic, communicates with the **PostgreSQL DB** (using Prisma as the ORM), and also calls the **OpenAI API** for AI-related tasks (trip suggestions).
- **Data Persistence:**
  The PostgreSQL database (Postgres 13) stores application data, and its persistent data is maintained via the **pgdata Volume**.

CI/CD Pipeline

1. **Checkout repository**
   Uses: `actions/checkout@v2`
   Explanation: This step checks out the repository's code into the workflow/github VM so subsequent steps can access the project files.
2. **Set up Node.js**
   Uses: `actions/setup-node@v2` with `node-version: '18'`
   Explanation: It sets up a Node.js environment using version 18. This is necessary to run npm commands for installing dependencies and running tests.
3. **Install dependencies**
   Commands:
   - `npm install`
   - `npm install --prefix core`
   - `npm install --prefix client`
     Explanation: This step installs all required packages. The first command installs dependencies defined in the root directory, while the subsequent commands install dependencies specific to the `core` and `client` subdirectories.
4. **Create .env file for core**
   Command:
   - `echo "${{ secrets.CORE_ENV }}" > core/.env`
     Explanation: This command creates a `.env` file inside the `core` directory, populating it with environment variables stored securely in the repository secrets.
5. **Create .env file for client**
   Command:
   - `echo "${{ secrets.CLIENT_ENV }}" > client/.env`
     Explanation: Similarly, this creates a `.env` file for the `client` directory with its specific environment variables from the secrets.
6. **Run tests**
   Command:
   - `npm run test --prefix core`
     Explanation: This command runs the test suite of unit tests for the core part of the project by executing the test script defined in its package configuration.

7. **Build core docker image**
   Command:
   - `docker build -t danielalyoshin/lada-core:latest -f core/Dockerfile .`
     Explanation: It builds a Docker image for the core service using the Dockerfile located in the `core` directory, tagging the image as `danielalyoshin/lada-core:latest`.
8. **Build client docker image**
   Command:
   - `docker build -t danielalyoshin/lada-client:latest -f client/Dockerfile .`
     Explanation: It builds a Docker image for the client service using the Dockerfile located in the `client` directory, tagging the image as `danielalyoshin/lada-client:latest`.
9. **Log into Docker Hub**
   Command:
   - `echo "${{ secrets.DOCKER_HUB_PASSWORD }}" | docker login -u "${{ secrets.DOCKER_HUB_USERNAME }}" --password-stdin`
     Explanation: This command logs into Docker Hub using credentials stored as secrets. Logging in is required to push the built Docker images to the Docker Hub repository.
10. **Push docker images to Docker Hub**
    Commands:
    - `docker push danielalyoshin/lada-core:latest`
    - `docker push danielalyoshin/lada-client:latest`
      Explanation: These commands upload the newly built Docker images for both the core and client services to the Docker Hub repository.
11. **SSH into VM and pull docker images and deploy**
    Uses: `appleboy/ssh-action@v1`
    Script executed on the remote VM:
    - `sudo docker pull danielalyoshin/lada-core:latest`
    - `sudo docker pull danielalyoshin/lada-client:latest`
    - `sudo docker compose -f docker-compose.prod.yml up -d`
      Explanation: This final step connects to a remote virtual machine hosted on GCP (using SSH credentials stored as secrets), pulls the latest Docker images from Docker Hub, and deploys the updated application using Docker Compose in detached mode.