Docker Tasks

1. Container Operations

Run an Alpine container interactively, install `curl`, then commit as a new image.

Launch an Nginx container with:

- Port mapping `8080:80`
- Auto-restart policy (`unless-stopped`)
- Hostname `webserver`

Force remove all containers (including running ones) in one command.

2. Image Management

Build a custom image that:

- Uses `ubuntu:22.04`
- Installs `apache2` and `curl`
- Copies `index.html` to `/var/www/html`
- Exposes port `80`

Tag your image as `yourusername/webapp:v1`.

Optimize image size by cleaning unnecessary files.

3. Storage & Volumes

Create a named volume `app-data` and mount it to `/data` in a `busybox` container.

Share a local directory as read-only inside a test container.

Inspect changes between a container's writable layer and its base image.

4. Networking

Create a custom bridge network `dev-net`.

Connect `nginx` and `busybox` to `dev-net` and verify communication.

Run a container with `host` network mode.

5. Debugging & Optimization

Diagnose why a container exits immediately after startup.

Limit a container to:

- `1GB` memory
- `50%` of a single CPU core

Monitor real-time resource usage of running containers.

6. DockerHub & Security

Push your custom image to DockerHub.

Pull a public image and scan it for vulnerabilities.

7. Advanced Challenges

Implement a multi-stage build for a Python app.