# AI-03079 DockerLab Two Compose-ChatGPT

| | |
|---|---|
| Owner | Ⓡ Rigel Arcayan |
| Tags | Project |
| Created time | @October 15, 2025 9:13 AM |

Build a Self-contained Docker Lab in my Ubuntu Server VM in Hyper-V.

Build 2 docker-compose files:

1.docker-compose-core.yml, for nginx npm, portainer, mariadb, postgres, pgadmin, phymyadmin

2.docker-compose-app.yml, for wordpress, wikijs, openemr, openproject, osticket, inventree

HyperVM Host IP = 192.168.1.2

Gateway = 192.168.1.1

DNS = 8.8.8.8, 8.8.4.4

Domain = ubuntuserver3.local

Ubuntu Server Host IP = 192.168.1.3

Ubuntu Server Ethernet Name = eth0

Ubuntu Server OS Version = Ubuntu 25.04 (plucky)

Home Directory of All .yaml, sql, env = /home/Docker

Docker Lab information and objectives:

Provide all the sudo commands I need to install and run Docker, Docker-Compose and Portainer, NGINX Proxy Manager, pgAdmin, phpMyAdmin, MariaDB, PostgreSQL with detailed functional descriptions. Assume that I am fairly new to Ubunbu and Docker but had been functionally educated through hands on with ChatGPT and actual HyperV VM, Ubuntu Server and Docker Compose and Portainer.

I want to use 1 network for the entire Docker Lab (Do not use Macvlan)

Provide commands to create all necessary folders inside my /home/docker directory.

Save all persistent volumes of MariaDB and PostgreSQL inside /home/docker. I want to use 1 Mariadb and 1 Postgresql db to simplify studying of db tables.

Create a unified necessary.env and docker-compose.yaml.

Create a db init script to initialize MariaDB and PostgreSQL on first use.

Provide instructions on how to start the Docker Lab at first use

Provide instructions on how to access the completed apps from my internal network.

Provide instructions on how to access the completed apps using the internet.

Provide instructions on how to stop the Docker Lab safely without losing data.

Provide instructions on how to stop the Docker Lab and wipe out all data to start over.

Provide instructions on how to monitor the Docker Lab using Portainer.

Provide instructions on how to monitor the Docker Lab using command lines.

Provide a summary of sudo commands for simplicity.

Folder Structure

/home/docker (home of each of the applications, persistent volumes and db)

/home/docker/portainer

/home/docker/nginx

/home/docker/mariadb

/home/docker/postgresql

/home/docker/openproject

/home/docker/osticket

/home/docker/openemr

/home/docker/inventree

/home/docker/wordpress

/home/docker/wikijs

Network Ports Assignment:

OpenProject Port=801

osTicket Port=802

OpenEMR Port=803

Inventree Port=804

Wordpress=805

Wikijs=806

DB Credentials: db name, user, passwords

openproject, openproject_user, password=SecurePassword!

osticket, osticket_user, password=SecurePassword!

openemr, openemr_user, password=SecurePassword!

inventree, inventree_user, password=SecurePassword!

wordpress, wordpress_user, password=SecurePassword!

wikijs, wikijs_user, password=SecurePassword!

Nice — I'll build this end-to-end for you. I'll assume the lab root is **/home/docker** (your folder list used lowercase), and I'll create:

- a single `.env` (unified `necessary.env`) with DB credentials and common values
- `docker-compose-core.yml` (nginx-proxy-manager, Portainer, MariaDB, PostgreSQL, phpMyAdmin, pgAdmin)
- `docker-compose-app.yml` (WordPress, Wiki.js, OpenEMR, OpenProject, osTicket, InvenTree)
- DB init scripts for MariaDB and PostgreSQL (to create the named DBs and DB users)
- all commands you need to install Docker, create folders, create the docker network, start/stop, wipe data, and monitor

I follow your port mapping and folder layout, save DB volumes under `/home/docker/mariadb` and `/home/docker/postgresql`, and use a single Docker bridge network `dockerlab_net`. No macvlan. Everything you can copy/paste and run now.

---

# 1) Direct Answer — what I'm giving you (files + commands)

Files (contents below):

- `/home/docker/necessary.env` (unified .env)
- `/home/docker/docker-compose-core.yml`
- `/home/docker/docker-compose-app.yml`
- `/home/docker/db/init-mariadb.sql`
- `/home/docker/db/init-postgres.sql`

Shell steps (single sequence near the end) to install Docker & compose plugin, create folders, create network, place files, and start the lab.

Ports & services (summary)

- OpenProject → `192.168.1.3:801`
- osTicket → `192.168.1.3:802`
- OpenEMR → `192.168.1.3:803`
- InvenTree → `192.168.1.3:804`
- WordPress → `192.168.1.3:805`
- Wiki.js → `192.168.1.3:806`
- Portainer → `192.168.1.3:9000`
- Nginx Proxy Manager UI → `192.168.1.3:81`
- phpMyAdmin → `192.168.1.3:8080`
- pgAdmin → `192.168.1.3:5050`
- MariaDB / PostgreSQL are internal (not exposed to WAN)

---

# 2) Key reasons (concise)

- Single network simplifies inter-service DNS (services can reference `mariadb` / `postgres` as hosts).
- Single MariaDB + single PostgreSQL reduces complexity for learning DB tables while keeping each app its own database and DB user.
- Nginx Proxy Manager (NPM) gives you easy internal & public routing + Let's Encrypt later.
- Portainer provides GUI container management and logs for monitoring.
- Persistent volumes live under `/home/docker/...` so stopping/starting keeps data.

---

# 3) Alternatives (brief)

- Use docker swarm / k8s for production — more complex; not necessary for learning.
- Use separate DB instances per app (safer for real production) — but more complex for studying.
- Use reverse proxy with Caddy or Traefik instead of NPM — both good; you chose NPM.

---

# 4) Action Plan — full step-by-step with files & commands

> Note: I will use /home/docker (lowercase) based on your folder list. If you instead want /home/Docker change the path in the commands and files.

---

## A — Install Docker, docker compose plugin, Portainer prerequisites (paste and run)

```
# 1. Update and install pre-reqs
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg lsb-release apt-transport-https

# 2. Add Docker's official GPG key and repo
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearm
or -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docke
r.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /de
v/null

sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plu
gin

# 3. Add your user to docker group (log out/in required for non-sudo docker)
sudo usermod -aG docker $USER

# 4. Enable and start docker
sudo systemctl enable --now docker

# Quick check
docker --version
```

You can still use `sudo docker` if you do not want to re-login; the commands in this guide use `sudo` for clarity.

---

## B — Create folders and files

Run these commands to create the directory structure and set ownership:

```
# Root folder (change if you used /home/Docker)
sudo mkdir -p /home/docker
cd /home/docker

# App folders & persistent volumes
sudo mkdir -p portainer nginx mariadb postgresql openproject osticket opene
mr inventree wordpress wikijs db

# Ensure correct permissions (you can chown to your user if desired)
sudo chown -R $USER:$USER /home/docker
```

---

## C — Create docker network (single bridge network)

```
# create a single bridge network for the lab
docker network create dockerlab_net
```

(If the network already exists this will error; that's fine.)

---

## D — `necessary.env` (unified .env)

Create `/home/docker/necessary.env` with the following contents. **Save exactly** as `.env` or `necessary.env` and we will reference it from compose files.

```
# /home/docker/necessary.env
TZ=Etc/UTC

# Host / domain info
HOST_IP=192.168.1.3
DOMAIN=ubuntuserver3.local

# MariaDB (single instance)
MARIADB_ROOT_PASSWORD=RootSecurePassword!
MARIADB_PORT=3306
```

```
# Postgres (single instance)
POSTGRES_PASSWORD=PostgresRootSecure!

# Shared DB credentials for the apps (you provided these passwords)
OPENPROJECT_DB=openproject
OPENPROJECT_DB_USER=openproject_user
OPENPROJECT_DB_PASS=SecurePassword!

OSTICKET_DB=osticket
OSTICKET_DB_USER=osticket_user
OSTICKET_DB_PASS=SecurePassword!

OPENEMR_DB=openemr
OPENEMR_DB_USER=openemr_user
OPENEMR_DB_PASS=SecurePassword!

INVENTREE_DB=inventree
INVENTREE_DB_USER=inventree_user
INVENTREE_DB_PASS=SecurePassword!

WORDPRESS_DB=wordpress
WORDPRESS_DB_USER=wordpress_user
WORDPRESS_DB_PASS=SecurePassword!

WIKIJS_DB=wikijs
WIKIJS_DB_USER=wikijs_user
WIKIJS_DB_PASS=SecurePassword!

# Portainer admin password (bcrypt) - optional, you can set initial password o
n first visit
PORTAINER_ADMIN_PASSWORD=admin1234

# NPM DB connection details (we'll use same MariaDB)
NPM_DB_NAME=npm
NPM_DB_USER=npm_user
NPM_DB_PASS=SecurePassword!
```

```
# phpMyAdmin login (uses MariaDB root or users)
PMA_USER=root
PMA_PASSWORD=${MARIADB_ROOT_PASSWORD}
```

Save this file as `/home/docker/necessary.env`. The compose files below will reference it.

---

## E — DB init scripts

Create `/home/docker/db/init-mariadb.sql`:

```
-- /home/docker/db/init-mariadb.sql
-- runs at first MariaDB container start (placed into docker volume)
CREATE DATABASE IF NOT EXISTS openproject CHARACTER SET utf8mb4 C
OLLATE utf8mb4_general_ci;
CREATE DATABASE IF NOT EXISTS osticket CHARACTER SET utf8mb4 COLLA
TE utf8mb4_general_ci;
CREATE DATABASE IF NOT EXISTS openemr CHARACTER SET utf8mb4 COLL
ATE utf8mb4_general_ci;
CREATE DATABASE IF NOT EXISTS inventree CHARACTER SET utf8mb4 COLL
ATE utf8mb4_general_ci;
CREATE DATABASE IF NOT EXISTS wordpress CHARACTER SET utf8mb4 COL
LATE utf8mb4_general_ci;
CREATE DATABASE IF NOT EXISTS npm CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci;

CREATE USER IF NOT EXISTS 'openproject_user'@'%' IDENTIFIED BY 'Secure
Password!';
CREATE USER IF NOT EXISTS 'osticket_user'@'%' IDENTIFIED BY 'SecurePas
sword!';
CREATE USER IF NOT EXISTS 'openemr_user'@'%' IDENTIFIED BY 'SecurePa
ssword!';
CREATE USER IF NOT EXISTS 'inventree_user'@'%' IDENTIFIED BY 'SecurePa
ssword!';
CREATE USER IF NOT EXISTS 'wordpress_user'@'%' IDENTIFIED BY 'SecureP
assword!';
```

```
CREATE USER IF NOT EXISTS 'npm_user'@'%' IDENTIFIED BY 'SecurePasswo
rd!';

GRANT ALL PRIVILEGES ON openproject.* TO 'openproject_user'@'%';
GRANT ALL PRIVILEGES ON osticket.* TO 'osticket_user'@'%';
GRANT ALL PRIVILEGES ON openemr.* TO 'openemr_user'@'%';
GRANT ALL PRIVILEGES ON inventree.* TO 'inventree_user'@'%';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress_user'@'%';
GRANT ALL PRIVILEGES ON npm.* TO 'npm_user'@'%';

FLUSH PRIVILEGES;
```

Create `/home/docker/db/init-postgres.sql`:

```
-- /home/docker/db/init-postgres.sql
-- executed by Postgres container on first startup
CREATE ROLE openproject_user WITH LOGIN PASSWORD 'SecurePassword!';
CREATE DATABASE openproject OWNER openproject_user;

CREATE ROLE wikijs_user WITH LOGIN PASSWORD 'SecurePassword!';
CREATE DATABASE wikijs OWNER wikijs_user;

CREATE ROLE inventree_user_postgres WITH LOGIN PASSWORD 'SecurePass
word!';
CREATE DATABASE inventree_pg OWNER inventree_user_postgres;
-- (Note: this file creates databases Postgres apps will use. Some apps use M
ariaDB instead.)
```

> Notes: MariaDB's Docker image runs .sql files placed in /docker-entrypoint-initdb.d; PostgreSQL also processes /docker-entrypoint-initdb.d/*.sql. Compose files below mount these.

---

## F — `docker-compose-core.yml` (create at `/home/docker/docker-compose-core.yml`)

```
# /home/docker/docker-compose-core.yml
version: "3.8"
services:

  mariadb:
    image: mariadb:10.11
    container_name: dockerlab_mariadb
    restart: unless-stopped
    env_file:
      - ./necessary.env
    environment:
      - MYSQL_ROOT_PASSWORD=${MARIADB_ROOT_PASSWORD}
    volumes:
      - /home/docker/mariadb/data:/var/lib/mysql
      - /home/docker/db/init-mariadb.sql:/docker-entrypoint-initdb.d/init-mariad
b.sql:ro
    networks:
      - dockerlab_net

  postgres:
    image: postgres:15
    container_name: dockerlab_postgres
    restart: unless-stopped
    env_file:
      - ./necessary.env
    environment:
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
    volumes:
      - /home/docker/postgresql/data:/var/lib/postgresql/data
      - /home/docker/db/init-postgres.sql:/docker-entrypoint-initdb.d/init-postg
res.sql:ro
    networks:
      - dockerlab_net

  portainer:
```

```
    image: portainer/portainer-ce:latest
    container_name: dockerlab_portainer
    restart: unless-stopped
    command: -H unix:///var/run/docker.sock
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /home/docker/portainer/data:/data
    ports:
      - "9000:9000"
    networks:
      - dockerlab_net

  nginx-proxy-manager:
    image: jc21/nginx-proxy-manager:latest
    container_name: dockerlab_nginxproxymanager
    restart: unless-stopped
    env_file:
      - ./necessary.env
    environment:
      # NPM will use the MariaDB DB we created above
      - DB_MYSQL_HOST=mariadb
      - DB_MYSQL_PORT=3306
      - DB_MYSQL_USER=${NPM_DB_USER}
      - DB_MYSQL_PASSWORD=${NPM_DB_PASS}
      - DB_MYSQL_NAME=${NPM_DB_NAME}
    volumes:
      - /home/docker/nginx/data:/data
      - /home/docker/nginx/letsencrypt:/etc/letsencrypt
    ports:
      - "81:81"      # NPM app UI
      - "80:80"      # HTTP (optional; required if you want LetsEncrypt HTTP chal
lenge)
      - "443:443"    # HTTPS
    networks:
      - dockerlab_net
    depends_on:
```

```
    - mariadb

  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    container_name: dockerlab_phpmyadmin
    restart: unless-stopped
    env_file:
      - ./necessary.env
    environment:
      - PMA_HOST=mariadb
      - PMA_PORT=3306
      - PMA_USER=${PMA_USER}
      - PMA_PASSWORD=${PMA_PASSWORD}
    ports:
      - "8080:80"
    networks:
      - dockerlab_net
    depends_on:
      - mariadb

  pgadmin:
    image: dpage/pgadmin4:latest
    container_name: dockerlab_pgadmin
    restart: unless-stopped
    env_file:
      - ./necessary.env
    environment:
      - PGADMIN_DEFAULT_EMAIL=admin@${DOMAIN}
      - PGADMIN_DEFAULT_PASSWORD=pgadmin1234
    volumes:
      - /home/docker/pgadmin:/var/lib/pgadmin
    ports:
      - "5050:80"
    networks:
      - dockerlab_net
    depends_on:
```

```
      - postgres

networks:
  dockerlab_net:
    external: true
```

Notes:

- `nginx-proxy-manager` uses the MariaDB DB and the DB user created via init script (`npm`, `npm_user`).
- `pgadmin` default login: `admin@ubuntuserver3.local` / `pgadmin1234` (change if you want).

---

## G — `docker-compose-app.yml` (create at `/home/docker/docker-compose-app.yml`)

```
# /home/docker/docker-compose-app.yml
version: "3.8"
services:

  wordpress:
    image: wordpress:latest
    container_name: dockerlab_wordpress
    restart: unless-stopped
    env_file:
      - ./necessary.env
    environment:
      - WORDPRESS_DB_HOST=mariadb:3306
      - WORDPRESS_DB_NAME=${WORDPRESS_DB}
      - WORDPRESS_DB_USER=${WORDPRESS_DB_USER}
      - WORDPRESS_DB_PASSWORD=${WORDPRESS_DB_PASS}
    volumes:
      - /home/docker/wordpress/html:/var/www/html
    ports:
      - "805:80"
    networks:
```

```
      - dockerlab_net
    depends_on:
      - mariadb

  wikijs:
    image: ghcr.io/requarks/wiki:2
    container_name: dockerlab_wikijs
    restart: unless-stopped
    env_file:
      - ./necessary.env
    environment:
      - DB_TYPE=postgres
      - DB_HOST=postgres
      - DB_PORT=5432
      - DB_USER=${WIKIJS_DB_USER}
      - DB_PASS=${WIKIJS_DB_PASS}
      - DB_NAME=${WIKIJS_DB}
    volumes:
      - /home/docker/wikijs/data:/var/wiki
    ports:
      - "806:3000"
    networks:
      - dockerlab_net
    depends_on:
      - postgres

  openproject:
    image: openproject/community:latest
    container_name: dockerlab_openproject
    restart: unless-stopped
    env_file:
      - ./necessary.env
    environment:
      - DATABASE_URL=postgresql://${OPENPROJECT_DB_USER}:SecurePassw
ord!@postgres:5432/${OPENPROJECT_DB}
    ports:
```

```
    - "801:80"
  volumes:
    - /home/docker/openproject/data:/var/openproject
  networks:
    - dockerlab_net
  depends_on:
    - postgres

osticket:
  image: osticket/osticket:1.17
  container_name: dockerlab_osticket
  restart: unless-stopped
  env_file:
    - ./necessary.env
  environment:
    - DBHOST=mariadb
    - DBUSER=${OSTICKET_DB_USER}
    - DBPASS=${OSTICKET_DB_PASS}
    - DBNAME=${OSTICKET_DB}
  volumes:
    - /home/docker/osticket/data:/data
  ports:
    - "802:80"
  networks:
    - dockerlab_net
  depends_on:
    - mariadb

openemr:
  image: openemr/openemr:latest
  container_name: dockerlab_openemr
  restart: unless-stopped
  env_file:
    - ./necessary.env
  environment:
    - MYSQL_HOST=mariadb
```

```
    - MYSQL_ROOT_PASS=${MARIADB_ROOT_PASSWORD}
    - MYSQL_USER=${OPENEMR_DB_USER}
    - MYSQL_PASSWORD=${OPENEMR_DB_PASS}
    - MYSQL_DATABASE=${OPENEMR_DB}
  volumes:
    - /home/docker/openemr:/var/www/localhost/htdocs
  ports:
    - "803:80"
  networks:
    - dockerlab_net
  depends_on:
    - mariadb

inventree:
  image: inventree/inventree:latest
  container_name: dockerlab_inventree
  restart: unless-stopped
  env_file:
    - ./necessary.env
  environment:
    - DB_TYPE=postgres
    - DB_HOST=postgres
    - DB_PORT=5432
    - DB_USER=${INVENTREE_DB_USER}
    - DB_PASS=${INVENTREE_DB_PASS}
    - DB_NAME=${INVENTREE_DB}
  volumes:
    - /home/docker/inventree/media:/opt/inventree/media
    - /home/docker/inventree/static:/opt/inventree/static
  ports:
    - "804:8000"
  networks:
    - dockerlab_net
  depends_on:
    - postgres
```

```
networks:
  dockerlab_net:
    external: true
```

Notes:

- Some images (OpenProject, InvenTree, Wiki.js) use PostgreSQL; others use MariaDB.
- The ports match your requested mapping. If an app uses a non-default internal port, compose maps to the external port you requested.

---

# H — First startup (order & commands)

From `/home/docker`:

```
# make sure .env is in /home/docker and named necessary.env (and referenc
ed in compose files)
cd /home/docker

# copy or rename necessary.env to .env if you want docker-compose to auto-
load .env
cp necessary.env .env

# start core services first
sudo docker compose -f docker-compose-core.yml up -d

# wait for mariadb & postgres to initialize (20-60s). Check logs if you want:
sudo docker logs -f dockerlab_mariadb
sudo docker logs -f dockerlab_postgres

# then start app services
sudo docker compose -f docker-compose-app.yml up -d

# verify all containers are running
sudo docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"
```

Expect the DB init scripts to run on first MariaDB/Postgres boot and create the databases + users.

---

# I — Accessing the completed apps from your internal network

Open a browser on any machine on your LAN and go to the Ubuntu Server IP:

- `http://192.168.1.3:805` → WordPress initial setup page
- `http://192.168.1.3:806` → Wiki.js (first-run setup)
- `http://192.168.1.3:801` → OpenProject
- `http://192.168.1.3:802` → osTicket
- `http://192.168.1.3:803` → OpenEMR
- `http://192.168.1.3:804` → InvenTree
- `http://192.168.1.3:9000` → Portainer GUI
- `http://192.168.1.3:81` → Nginx Proxy Manager GUI
- `http://192.168.1.3:8080` → phpMyAdmin (login with `root` + `RootSecurePassword!` or app user)
- `http://192.168.1.3:5050` → pgAdmin (login admin@ubuntuserver3.local / pgadmin1234)

For NPM, after logging into NPM UI you can set proxy hosts to point friendly hostnames (e.g., `wordpress.ubuntuserver3.local`) to the internal ports above, add SSL certificates, redirect, etc.

---

# J — Expose to the internet (high level; do it carefully)

**Security preface**: Exposing services to the internet requires careful firewall/router configuration and HTTPS. Use Nginx Proxy Manager to add TLS with Let's Encrypt and to route hostnames to internal services. Recommended method:

1. On your home router/HyperV host:
    - Forward ports 80 and 443 to `192.168.1.3` (Ubuntu Server VM).

- Forward management port if needed (I recommend blocking Portainer UI from the internet; use VPN to access it).

2. In Nginx Proxy Manager:
   - Add a Proxy Host for each app (example `wordpress.ubuntuserver3.com`) mapping to `192.168.1.3:805`.
   - Enable `Force SSL` and request Let's Encrypt certificate (requires port 80/443 forwarded and public DNS pointing to your public IP).
   - For security, restrict admin endpoints (Portainer, phpMyAdmin, pgAdmin) to local network or protect with strong auth & firewall.
3. Use firewall on Ubuntu:

```
# example - allow only http/https and ssh if desired
sudo apt install -y ufw
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable
```

1. Recommend using a VPN (WireGuard) into your LAN for secure remote access rather than exposing management UIs to public Internet.

---

## K — Stop the Docker Lab safely (preserve data)

Stop both compose stacks gracefully:

```
# stop apps
cd /home/docker
sudo docker compose -f docker-compose-app.yml down

# stop core services
sudo docker compose -f docker-compose-core.yml down
```

`down` stops containers and removes the containers/networks created by compose, but **persists volumes** under `/home/docker/*` because we used host-mounted directories. Your MariaDB and Postgres data remain in `/home/docker/mariadb/data` and `/home/docker/postgresql/data`.

If you prefer to stop without removing containers, use `sudo docker compose -f ... stop` to stop (keeps containers).

---

## L — Wipe everything (stop and remove all data) — use with extreme caution

```
# stop & remove containers and volumes
cd /home/docker
sudo docker compose -f docker-compose-app.yml down
sudo docker compose -f docker-compose-core.yml down

# remove host folders (this deletes all persistent data!)
sudo rm -rf /home/docker/mariadb/data
sudo rm -rf /home/docker/postgresql/data
sudo rm -rf /home/docker/portainer
sudo rm -rf /home/docker/nginx
sudo rm -rf /home/docker/pgadmin
sudo rm -rf /home/docker/wordpress
sudo rm -rf /home/docker/wikijs
sudo rm -rf /home/docker/openproject
sudo rm -rf /home/docker/osticket
sudo rm -rf /home/docker/openemr
sudo rm -rf /home/docker/inventree
# remove db init files if you want a fresh DB on next startup
sudo rm -f /home/docker/db/init-*.sql

# remove docker network if desired
docker network rm dockerlab_net
```

After that you can recreate folders, re-populate the init SQL, and `docker compose up` again to start fresh.

---

# M — Monitoring

## Via Portainer (GUI)

1. Open `http://192.168.1.3:9000`.
2. Login (first run will ask you to create an admin user).
3. Use `Containers` → view logs, stats, start/stop/restart, and manage volumes and networks.

## Via CLI (docker)

Useful commands:

```
# list containers
sudo docker ps -a

# follow a container log:
sudo docker logs -f dockerlab_wordpress

# show resource usage (live)
sudo docker stats --all

# inspect a container
sudo docker inspect dockerlab_mariadb

# list volumes and size
sudo docker system df -v

# prune unused objects (careful)
sudo docker system prune -a --volumes
```

---

## N — Summary / Cheat-sheet of sudo commands

```
# install docker
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg lsb-release apt-transport-https
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearm
or -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/do
cker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stabl
e" | sudo tee /etc/apt/sources.list.d/docker.list
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plu
gin
sudo usermod -aG docker $USER
sudo systemctl enable --now docker

# create folders
sudo mkdir -p /home/docker/{portainer,nginx,mariadb,postgresql,openprojec
t,osticket,openemr,inventree,wordpress,wikijs,db}
sudo chown -R $USER:$USER /home/docker

# create network
docker network create dockerlab_net

# start core
cd /home/docker
cp necessary.env .env
sudo docker compose -f docker-compose-core.yml up -d

# start apps
sudo docker compose -f docker-compose-app.yml up -d

# stop safely
sudo docker compose -f docker-compose-app.yml down
```

```
sudo docker compose -f docker-compose-core.yml down

# wipe data (DANGEROUS)
sudo rm -rf /home/docker/mariadb/data /home/docker/postgresql/data /hom
e/docker/portainer /home/docker/nginx /home/docker/pgadmin /home/docke
r/wordpress /home/docker/wikijs /home/docker/openproject /home/docker/os
ticket /home/docker/openemr /home/docker/inventree
docker network rm dockerlab_net
```

---

## O — Extra tips & gotchas

- **Sleep on DB init**: On first start, DB init can take a minute. Check `docker logs -f dockerlab_mariadb` and `docker logs -f dockerlab_postgres`.
- **DNS inside compose**: Services can refer to `mariadb` and `postgres` as the hostnames because they share one network.
- **Backups**: Regularly copy `/home/docker/mariadb/data` and `/home/docker/postgresql/data` (or use `mysqldump` / `pg_dump`) before major changes.
- **Secure passwords**: Replace the example passwords in `necessary.env` with stronger ones. Don't expose Portainer, phpMyAdmin, pgAdmin to the WAN without auth and IP restrictions.
- **NPM & LetsEncrypt**: To get Let's Encrypt certificates NPM must be reachable from the public Internet on port 80 and 443 (for DNS pointing to your home public IP). If you have a dynamic IP, use a dynamic DNS service or don't expose cert issuance to the open net and instead use Cloudflare DNS + DNS challenge (advanced).

---

If you want, I can:

- produce these files in a downloadable `.zip` (I can generate the files here and attach them as files you can download), or
- adapt the stack to use `docker-compose` one-file multi-stack (you asked for two files; I kept them separate)

- harden the install (disable root login, add fail2ban, limit NPM access) — I can add those steps next.

Which would you like next?