AI-02083 Docker Database in Local Ubuntu Server

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	@September 7, 2025 10:53 PM

I’m new to Ubuntu server and Docker, I need help in the following objectives.

1. Use 1 vm for my running Ubuntu server

2. Create a docker container MySQL database as backend db

3. Create a docker container Mariadb as backend db

4. Create a docker container Postgres as backend db

5. Run docker OpenEMR, Wordpress, osTicket, BookStack, OpenProject on each container and connect to each respective backend databases

Write a comprehensive step by step instructions on how to deploy all the applications mentioned as standalone containers connecting to the backend database

Direct answer

Yes — you can run **one Ubuntu VM** and host **3 database containers** (MySQL, MariaDB, PostgreSQL) plus **5 app containers** (OpenEMR, WordPress, osTicket, BookStack, OpenProject) using **Docker Compose**.

Below I give a complete, copy-pasteable step-by-step guide (install Docker, create the project files, `docker compose up`, how to initialize DBs, and quick troubleshooting). The guide uses a single `docker-compose.yml`, a `.env` file, and an init SQL script (for MariaDB) so each app connects to its respective backend DB.

Key reasons (concise)

- Docker + Docker Compose is the simplest, repeatable way to run many containers on one VM. (Docker Documentation)
- Official DB images provide simple environment variables to initialize users/databases (MySQL / MariaDB / Postgres). (Docker Hub)
- OpenEMR expects a MySQL/MariaDB backend; OpenProject is packaged to run with PostgreSQL. (I map them accordingly.) (OpenEMR, OpenProject.org)

Action plan — full step-by-step (copy/pasteables included)

> Assumptions: you have an Ubuntu Server VM (22.04 / 24.04 recommended), you have sudo access, and you want containers to be persistent (volumes) and reachable via different host ports (for initial testing). Production hardening (TLS, firewall rules, secrets) is covered after the steps.

---

0) Recommended VM sizing

- 2–4 vCPU, **4–8 GB RAM** minimum (more for OpenEMR/OpenProject under load). Disk: 50+ GB. (Adjust to your usage.)

---

1) Install Docker Engine & Docker Compose plugin (official method)

Follow Docker's official instructions — example commands below (works on Ubuntu 22.04 / 24.04). See Docker docs for any distro specifics. (Docker Documentation)

```
# remove old conflicting packages (one-liner from docs)
sudo apt-get remove -y docker docker-engine docker.io containerd runc || tru
e
```

```
# Install prerequisites, add Docker repo and GPG key
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg lsb-release

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearm
or -o /usr/share/keyrings/docker-archive-keyring.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/dock
er-archive-keyring.gpg] \
  https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose
-plugin

# Add your user to docker group (log out/login required)
sudo usermod -aG docker "$USER"

# Verify
docker --version
docker compose version
```

---

2) Project directory layout (on VM)

Example structure:

```
/home/ubuntu/multi-app-stack/
├─ docker-compose.yml
├─ .env
├─ mariadb/
│  └─ initdb.d/
│     └─ init.sql
```

Create it:

```
mkdir -p ~/multi-app-stack/mariadb/initdb.d
cd ~/multi-app-stack
```

---

3) Create your `.env` file (do **not** commit to git)

Create `~/multi-app-stack/.env` with strong passwords. Example (change every password):

```
# MySQL (for OpenEMR)
MYSQL_ROOT_PASSWORD=ChangeMeOpenEMRRoot!
OPENEMR_DB=openemr
OPENEMR_DB_USER=openemr
OPENEMR_DB_PASSWORD=ChangeMeOpenEMR!

# MariaDB (for WordPress, osTicket, BookStack)
MARIADB_ROOT_PASSWORD=ChangeMeMariaRoot!
WP_DB=wordpress
WP_DB_USER=wp_user
WP_DB_PASSWORD=ChangeMeWP!
OST_DB=osticket
OST_DB_USER=ost_user
OST_DB_PASSWORD=ChangeMeOST!
BOOK_DB=bookstack
BOOK_DB_USER=book_user
BOOK_DB_PASSWORD=ChangeMeBook!

# PostgreSQL (for OpenProject)
POSTGRES_USER=openproject
POSTGRES_PASSWORD=ChangeMeOP!
POSTGRES_DB=openproject

# Host ports (optional adjust)
WP_PORT=8080
OPENEMR_PORT=8081
```

```
OST_PORT=8082
BOOK_PORT=8083
OPENPROJECT_PORT=8084
```

4) MariaDB init script — create multiple DBs & users

The MariaDB official image will run `*.sql` scripts from `/docker-entrypoint-initdb.d` on first startup. Create `mariadb/initdb.d/init.sql`:

```
-- mariadb/initdb.d/init.sql
CREATE DATABASE IF NOT EXISTS `wordpress` CHARACTER SET utf8mb4 C
OLLATE utf8mb4_unicode_ci;
CREATE DATABASE IF NOT EXISTS `osticket` CHARACTER SET utf8mb4 COLL
ATE utf8mb4_unicode_ci;
CREATE DATABASE IF NOT EXISTS `bookstack` CHARACTER SET utf8mb4 CO
LLATE utf8mb4_unicode_ci;

CREATE USER IF NOT EXISTS 'wp_user'@'%' IDENTIFIED BY 'ChangeMeWP!';
GRANT ALL PRIVILEGES ON `wordpress`.* TO 'wp_user'@'%';

CREATE USER IF NOT EXISTS 'ost_user'@'%' IDENTIFIED BY 'ChangeMeOS
T!';
GRANT ALL PRIVILEGES ON `osticket`.* TO 'ost_user'@'%';

CREATE USER IF NOT EXISTS 'book_user'@'%' IDENTIFIED BY 'ChangeMeBo
ok!';
GRANT ALL PRIVILEGES ON `bookstack`.* TO 'book_user'@'%';

FLUSH PRIVILEGES;
```

> Replace passwords (or mount secrets) before starting.

5) Full `docker-compose.yml` (paste into `~/multi-app-stack/docker-compose.yml`)

This file creates 3 DB containers and 5 app containers. Apps are exposed to different host ports for testing. It uses the **official** DB images and popular app images.

```
version: "3.8"

services:
  # ----------------------
  # Databases (do not publish DB ports publicly)
  # ----------------------
  mysql-db:
    image: mysql:8.0
    container_name: mysql-db
    environment:
      MYSQL_ROOT_PASSWORD: "${MYSQL_ROOT_PASSWORD}"
      MYSQL_DATABASE: "${OPENEMR_DB}"
      MYSQL_USER: "${OPENEMR_DB_USER}"
      MYSQL_PASSWORD: "${OPENEMR_DB_PASSWORD}"
    volumes:
      - mysql_data:/var/lib/mysql
    restart: unless-stopped

  mariadb-db:
    image: mariadb:latest
    container_name: mariadb-db
    environment:
      MARIADB_ROOT_PASSWORD: "${MARIADB_ROOT_PASSWORD}"
    volumes:
      - mariadb_data:/var/lib/mysql
      - ./mariadb/initdb.d:/docker-entrypoint-initdb.d:ro
    restart: unless-stopped

  postgres-db:
```

```
  image: postgres:15
  container_name: postgres-db
  environment:
    POSTGRES_USER: "${POSTGRES_USER}"
    POSTGRES_PASSWORD: "${POSTGRES_PASSWORD}"
    POSTGRES_DB: "${POSTGRES_DB}"
  volumes:
    - postgres_data:/var/lib/postgresql/data
  restart: unless-stopped

# ----------------------
# Applications
# ----------------------
wordpress:
  image: wordpress:latest
  container_name: wordpress
  depends_on:
    - mariadb-db
  environment:
    WORDPRESS_DB_HOST: "mariadb-db:3306"
    WORDPRESS_DB_USER: "${WP_DB_USER}"
    WORDPRESS_DB_PASSWORD: "${WP_DB_PASSWORD}"
    WORDPRESS_DB_NAME: "${WP_DB}"
  volumes:
    - wordpress_data:/var/www/html
  ports:
    - "${WP_PORT}:80"
  restart: unless-stopped

openemr:
  image: openemr/openemr:latest
  container_name: openemr
  depends_on:
    - mysql-db
  environment:
    MYSQL_HOST: "mysql-db"
```

```
    MYSQL_ROOT_PASS: "${MYSQL_ROOT_PASSWORD}"
    # optional: create/app DB credentials (OpenEMR installer can also create)
    MYSQL_USER: "${OPENEMR_DB_USER}"
    MYSQL_PASS: "${OPENEMR_DB_PASSWORD}"
    MYSQL_DATABASE: "${OPENEMR_DB}"
  volumes:
    - openemr_data:/var/www/html
  ports:
    - "${OPENEMR_PORT}:80"
  restart: unless-stopped

osticket:
  image: tiredofit/osticket:latest
  container_name: osticket
  depends_on:
    - mariadb-db
  environment:
    MYSQL_HOST: "mariadb-db"
    MYSQL_USER: "${OST_DB_USER}"
    MYSQL_PASSWORD: "${OST_DB_PASSWORD}"
    MYSQL_DATABASE: "${OST_DB}"
  volumes:
    - osticket_data:/data
  ports:
    - "${OST_PORT}:80"
  restart: unless-stopped

bookstack:
  image: lscr.io/linuxserver/bookstack:latest
  container_name: bookstack
  depends_on:
    - mariadb-db
  environment:
    - PUID=1000
    - PGID=1000
    - APP_URL=http://localhost:${BOOK_PORT:-8083}
```

```
      - DB_HOST=mariadb-db
      - DB_USER=${BOOK_DB_USER}
      - DB_PASS=${BOOK_DB_PASSWORD}
      - DB_DATABASE=${BOOK_DB}
    volumes:
      - bookstack_data:/config
    ports:
      - "${BOOK_PORT}:80"
    restart: unless-stopped

  openproject:
    image: openproject/community:latest
    container_name: openproject
    depends_on:
      - postgres-db
    environment:
      # OpenProject uses DATABASE_URL for external PostgreSQL
      DATABASE_URL: "postgres://${POSTGRES_USER}:${POSTGRES_PASSWO
RD}@postgres-db:5432/${POSTGRES_DB}"
      # other env vars (email, secrets) can be added to .env or override files
    volumes:
      - openproject_data:/var/db
    ports:
      - "${OPENPROJECT_PORT}:8080"
    restart: unless-stopped

volumes:
  mysql_data:
  mariadb_data:
  postgres_data:
  wordpress_data:
  openemr_data:
  osticket_data:
  bookstack_data:
  openproject_data:
```

> Notes:
>
> - The `depends_on` simply ensures start order; it does **not** wait for DB readiness. I strongly advise the DB-first start recommended below.
> - Images used: `mysql`, `mariadb`, `postgres`, `wordpress`, `openemr/openemr`, `tiredofit/osticket`, `lscr.io/linuxserver/bookstack`, `openproject/community`. (Official DB images documented here.) (Docker Hub)

---

6) Start only the databases first (recommended)

Start DBs, wait until they are ready, then bring up the apps — this avoids app installer failures while DB still initializing.

```
# from ~/multi-app-stack
docker compose up -d mysql-db mariadb-db postgres-db

# watch logs until each DB reports ready:
docker compose logs -f mysql-db
docker compose logs -f mariadb-db
docker compose logs -f postgres-db
```

You can also test DB connectivity from the host:

```
# MySQL (from host) -- optional if port is not published use docker exec into mysql container:
docker exec -it mysql-db mysql -u root -p"${MYSQL_ROOT_PASSWORD}" -e "SELECT VERSION();"

# MariaDB (test)
docker exec -it mariadb-db mysql -u root -p"${MARIADB_ROOT_PASSWORD}" -e "SHOW DATABASES;"

# Postgres:
docker exec -it postgres-db psql -U "${POSTGRES_USER}" -d "${POSTGRES_DB}" -c "SELECT version();"
```

7) Start the applications

Once DBs are ready:

```
docker compose up -d wordpress openemr osticket bookstack openproject
```

Check status:

```
docker compose ps
docker compose logs -f wordpress
```

8) Finish web installers (first-time)

Open in browser (replace `VM_IP` with your VM address):

- WordPress → `http://VM_IP:${WP_PORT}` → run WordPress web installer. (Docker Hub)
- OpenEMR → `http://VM_IP:${OPENEMR_PORT}` → OpenEMR web UI; OpenEMR will attempt automatic install if env vars present (or guide through DB settings). (OpenEMR)
- osTicket → `http://VM_IP:${OST_PORT}` → follow installer (it will create config file using DB credentials). (Most osticket Docker images expect MySQL/MariaDB). (Docker Hub, GitHub)
- BookStack → `http://VM_IP:${BOOK_PORT}` → first-run creates APP_KEY and runs migrations; check `/config` for `.env`. (linuxserver/bookstack docs). (LinuxServer)
- OpenProject → `http://VM_IP:${OPENPROJECT_PORT}` → initial admin login (docs show using their docker-compose stack for full setup). (OpenProject.org)

9) Quick backup & restore tips

- Backup volumes by creating a tar of the volume mount folder (or use `docker run --rm -v mysql_data:/data -v $(pwd):/backup alpine tar czf /backup/mysql_data.tar.gz -C /data .`).
- For Postgres, `pg_dump` from inside the `postgres-db` container.

10) Troubleshooting (common problems)

- **App cannot connect to DB** — DB might still be initializing. Re-check DB container logs and run the test queries from Step 6.
- **Passwords / wrong env var name** — use `docker compose config` to see resolved env vars and inspect `docker compose logs <service>`.
- **Permissions errors for BookStack** — ensure `PUID` / `PGID` match and `/config` volume is writable. See linuxserver docs. (LinuxServer)

11) Security & production notes (do before exposing to public)

- Do **not** publish DB ports to the public internet — avoid `ports:` on DB services; keep them internal to the Docker network (this compose doesn't publish DB ports).
- Use Docker **secrets** for DB passwords or mount a protected `.env` file; do not store secrets in Git.
- Use a reverse proxy (Nginx, Traefik, or Nginx Proxy Manager) with TLS (Let's Encrypt) in front of your apps.
- Use `ufw` / iptables to allow only required ports (e.g., 80/443).
- For production, consider managed databases or a proper backup strategy and monitoring.

Short references (officials / helpful docs)

- Docker Engine on Ubuntu (install & compose plugin). (Docker Documentation)
- MySQL official image (env vars: `MYSQL_ROOT_PASSWORD`, `MYSQL_DATABASE`, `MYSQL_USER`, `MYSQL_PASSWORD`). (Docker Hub)
- MariaDB official image (init scripts in `/docker-entrypoint-initdb.d` for multiple DBs/users). (Docker Hub)

- PostgreSQL official image (`POSTGRES_USER`, `POSTGRES_PASSWORD`, `POSTGRES_DB`). (Docker Hub)
- OpenEMR Docker / compatibility notes (OpenEMR prefers MariaDB; requires a MySQL/MariaDB backend). (OpenEMR)
- WordPress official image docs (WordPress DB env usage). (Docker Hub)
- OpenProject Docker Compose docs / example stack. (OpenProject.org)

Quick checklist (what to run right now)

1. Install Docker (run commands in Step 1).
2. Create project dir and `.env`, `mariadb/initdb.d/init.sql`.
3. Save `docker-compose.yml` from Step 5.
4. `docker compose up -d mysql-db mariadb-db postgres-db` → wait until DBs are ready.
5. `docker compose up -d` (start apps) → visit the ports listed and finish web installers.

If you want, I can:

- produce a **ready-to-run** single tarball (env + compose + init.sql) you can download and SCP to your VM, or
- adapt this compose for **Traefik + Let's Encrypt** reverse proxy so the apps run on subdomains with automatic TLS, or
- convert the stack into separate per-app compose files or systemd templates.

Which of those would you like me to prepare next?