

# AI-03031 Docker Deployment of Wikijs osTicket OpenEMR OpenProject

👤 Owner	Ⓜ Rigel Arcayan
☰ Tags	Project
🕒 Created time	@September 25, 2025 12:40 AM

I would like to consolidate my Ubuntu Server application via Docker setup with the following information. Create a unified necessary.env and docker-compose.yaml. Also, I want the Network Container IP to be in the same network as my Ubuntu Server: 192.168.1.3, 255.255.255.0 and the Gateway is 192.168.1.1 using macvlan type. For the Mariadb and Postgresql, include the necessary init scripts and file location to run at first db use. For the persistent volumes, I want all the files inside my home folder so that I have 1 folder to back up and make my Docker safe and redeployable for later use.

~/home/rarcayan/docker (home of each of the applications and db)

~/home/rarcayan/docker/mariadb

~/home/rarcayan/docker/postgresql

~/home/rarcayan/docker/wikijs

~/home/rarcayan/docker/osTicket

~/home/rarcayan/docker/OpenEMR

~/home/rarcayan/docker/OpenProject

1. Mariadb Container for Wordpress, osTicket, OpenEMR Database
2. Postgresql Container for OpenProject and Wikijs Database
3. Network Container for Wikijs osTicket, OpenEMR, Openproject (in one subnet)
4. Wikijs Container
5. osTicket Container

6. **OpenEMR Container**
7. **OpenProject Container**
8. **Wikijs Port=801**
9. **osTicket Port=802**
10. **OpenEMR Port=803**
11. **OpenProject Port=804**
12. **WikijsDB, wikijs\_user, password=@SecurePassword!**
13. **osTicketDB, osticket\_user, password=@SecurePassword!**
14. **OpenEMRDB, openemr\_user, password=@SecurePassword!**
15. **OpenProjectDB, openproject\_user, password=@SecurePassword!**

Got it 👍 You want a **single consolidated Docker setup** with `docker-compose.yml` + `necessary.env`, using **macvlan** networking so your containers share the same LAN as your Ubuntu Server. All persistent volumes will sit inside `~/home/rarcayan/docker/...`, making backup and redeployment straightforward.

Here's the structured solution:



### `necessary.env`

```
# Database Credentials
MARIADB_ROOT_PASSWORD=@SecurePassword!
MARIADB_USER=appuser
MARIADB_PASSWORD=@SecurePassword!

POSTGRES_USER=appuser
POSTGRES_PASSWORD=@SecurePassword!
POSTGRES_DB=appdb

# Wikijs
WIKIJS_DB=wikijsdb
```

```
WIKIJS_USER=wikijs_user
WIKIJS_PASSWORD=@SecurePassword!

# osTicket
OSTICKET_DB=osticketdb
OSTICKET_USER=osticket_user
OSTICKET_PASSWORD=@SecurePassword!

# OpenEMR
OPENEMR_DB=openemrdb
OPENEMR_USER=openemr_user
OPENEMR_PASSWORD=@SecurePassword!

# OpenProject
OPENPROJECT_DB=openprojectdb
OPENPROJECT_USER=openproject_user
OPENPROJECT_PASSWORD=@SecurePassword!

# Network
PARENT_INTERFACE=eth0
DOCKER_SUBNET=192.168.1.0/24
DOCKER_GATEWAY=192.168.1.1
```



## docker-compose.yaml

```
version: "3.9"

networks:
  macvlan_net:
    driver: macvlan
    driver_opts:
      parent: ${PARENT_INTERFACE}
    ipam:
      config:
```

```
- subnet: ${DOCKER_SUBNET}
  gateway: ${DOCKER_GATEWAY}
```

#### services:

##### mariadb:

```
image: mariadb:11
container_name: mariadb
restart: unless-stopped
environment:
  MYSQL_ROOT_PASSWORD: ${MARIADB_ROOT_PASSWORD}
volumes:
- ~/home/rarcayan/docker/mariadb/data:/var/lib/mysql
- ~/home/rarcayan/docker/mariadb/init:/docker-entrypoint-initdb.d
networks:
  macvlan_net:
```

##### postgres:

```
image: postgres:17
container_name: postgres
restart: unless-stopped
environment:
  POSTGRES_USER: ${POSTGRES_USER}
  POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
volumes:
- ~/home/rarcayan/docker/postgresql/data:/var/lib/postgresql/data
- ~/home/rarcayan/docker/postgresql/init:/docker-entrypoint-initdb.d
networks:
  macvlan_net:
```

##### wikijs:

```
image: requarks/wiki:2
container_name: wikijs
restart: unless-stopped
ports:
- "801:3000"
environment:
```

```
DB_TYPE: postgres
DB_HOST: postgres
DB_PORT: 5432
DB_USER: ${WIKIJS_USER}
DB_PASS: ${WIKIJS_PASSWORD}
DB_NAME: ${WIKIJS_DB}
```

depends\_on:

- postgres

volumes:

- ~/home/rarcayan/docker/wikijs:/var/wiki

networks:

- macvlan\_net:

osticket:

image: campbellsoftwaresolutions/osticket

container\_name: osticket

restart: unless-stopped

ports:

- "802:80"

environment:

MYSQL\_HOST: mariadb

MYSQL\_DATABASE: \${OSTICKET\_DB}

MYSQL\_USER: \${OSTICKET\_USER}

MYSQL\_PASSWORD: \${OSTICKET\_PASSWORD}

depends\_on:

- mariadb

volumes:

- ~/home/rarcayan/docker/osTicket:/var/www/html

networks:

- macvlan\_net:

openemr:

image: openemr/openemr

container\_name: openemr

restart: unless-stopped

ports:

```

- "803:80"
environment:
  MYSQL_HOST: mariadb
  MYSQL_ROOT_PASS: ${MARIADB_ROOT_PASSWORD}
  MYSQL_USER: ${OPENEMR_USER}
  MYSQL_PASS: ${OPENEMR_PASSWORD}
  MYSQL_DATABASE: ${OPENEMR_DB}
depends_on:
  - mariadb
volumes:
  - ~/home/rarcayan/docker/OpenEMR:/var/www/localhost/htdocs/openemr
networks:
  macvlan_net:

openproject:
  image: openproject/community:14
  container_name: openproject
  restart: unless-stopped
  ports:
    - "804:80"
  environment:
    DATABASE_URL: "postgresql://${OPENPROJECT_USER}:${OPENPROJECT_PASSWORD}@postgres:5432/${OPENPROJECT_DB}"
  depends_on:
    - postgres
  volumes:
    - ~/home/rarcayan/docker/OpenProject:/var/openproject
  networks:
    macvlan_net:

```

## Init Script Locations

- **MariaDB init:** `~/home/rarcayan/docker/mariadb/init/01-init.sql`
- **Postgres init:** `~/home/rarcayan/docker/postgresql/init/01-init.sql`

## Example: MariaDB init ( 01-init.sql )

```
CREATE DATABASE IF NOT EXISTS osticketdb;
CREATE USER 'osticket_user'@'%' IDENTIFIED BY '@SecurePassword!';
GRANT ALL PRIVILEGES ON osticketdb.* TO 'osticket_user'@'%';

CREATE DATABASE IF NOT EXISTS openemrdb;
CREATE USER 'openemr_user'@'%' IDENTIFIED BY '@SecurePassword!';
GRANT ALL PRIVILEGES ON openemrdb.* TO 'openemr_user'@'%';
```

## Example: Postgres init ( 01-init.sql )

```
CREATE DATABASE wikisdb;
CREATE USER wikis_user WITH ENCRYPTED PASSWORD '@SecurePassword!';
GRANT ALL PRIVILEGES ON DATABASE wikisdb TO wikis_user;

CREATE DATABASE openprojectdb;
CREATE USER openproject_user WITH ENCRYPTED PASSWORD '@SecurePassword!';
GRANT ALL PRIVILEGES ON DATABASE openprojectdb TO openproject_user;
```

## Action Plan

1. Save `necessary.env` and `docker-compose.yaml` in `~/home/rarcayan/docker/`.
2. Create init SQL scripts in their respective `init` directories.
3. Run:

```
cd ~/home/rarcayan/docker
docker compose --env-file necessary.env up -d
```

4. Verify each service at:

- Wiki.js → `http://192.168.1.xxx:801`

- osTicket → `http://192.168.1.xxx:802`
- OpenEMR → `http://192.168.1.xxx:803`
- OpenProject → `http://192.168.1.xxx:804`

**Do you want me to also include fixed IPs per container (e.g., `192.168.1.10` for Wiki.js, `192.168.1.11` for osTicket, etc.), or do you prefer Docker to assign automatically within the subnet?**

I forgot, this is a fresh Ubuntu Server setup. Please provide all the sudo commands I need to run Docker, Docker-Compose and Portainer.

Perfect 👍 Since this is a **fresh Ubuntu Server**, let's go step by step and set up **Docker, Docker Compose v2, and Portainer** so your environment is ready for the `docker-compose.yaml` and `necessary.env` we built.

## Step 1: Update Your Server

```
sudo apt update && sudo apt upgrade -y
```

## Step 2: Install Docker Engine

```
# Install prerequisites
```

```
sudo apt install -y apt-transport-https ca-certificates curl software-properties  
-common gnupg lsb-release
```

```
# Add Docker's official GPG key
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearm  
or -o /usr/share/keyrings/docker.gpg
```

```
# Add Docker repository
```

```
echo \  
"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/dock
```

```
er.gpg] https://download.docker.com/linux/ubuntu \  
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /de  
v/null
```

```
# Update and install Docker  
sudo apt update  
sudo apt install -y docker-ce docker-ce-cli containerd.io
```

## Step 3: Enable Docker Service

```
# Start Docker on boot  
sudo systemctl enable docker  
sudo systemctl start docker
```

```
# Check version  
docker --version
```

## Step 4: Install Docker Compose v2

Docker Compose is now a **plugin**.

```
sudo apt install -y docker-compose-plugin  
docker compose version
```

## Step 5: Add Your User to Docker Group (so you don't need sudo each time)

```
sudo usermod -aG docker $USER  
newgrp docker
```

(Log out and back in if needed.)

---

## Step 6: Install Portainer (Docker GUI)

```
# Create Portainer volume
sudo docker volume create portainer_data

# Run Portainer (latest version)
sudo docker run -d \
  -p 9000:9000 -p 9443:9443 \
  --name portainer \
  --restart=always \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v portainer_data:/data \
  portainer/portainer-ce:latest
```

Access Portainer at:

👉 [https://YOUR\\_SERVER\\_IP:9443](https://YOUR_SERVER_IP:9443) (secure, recommended)

👉 [http://YOUR\\_SERVER\\_IP:9000](http://YOUR_SERVER_IP:9000) (HTTP, less secure)

First login: create an **admin password**.

---

## Step 7: Create Your Project Directory

```
mkdir -p ~/home/rarcayan/docker/{mariadb,postgresql,wikijs,osTicket,OpenEMR,OpenProject}
```

## Step 8: Deploy Your Stack

```
cd ~/home/rarcayan/docker
nano necessary.env    # paste env file
```

```
nano docker-compose.yaml # paste yaml file

# Deploy
docker compose --env-file necessary.env up -d
```

## ✓ At This Point

- Docker is installed
- Docker Compose v2 is available
- Portainer GUI is running at port `9443`
- Your `docker-compose.yaml` stack is ready to launch

Would you like me to also include **macvlan setup commands** (create the custom Docker macvlan network manually with fixed IPs) before launching the stack, or do you prefer to let `docker-compose` handle network creation automatically?

That means a lot to hear 🙏 — thank you! You're diving into a big ecosystem (Ubuntu + Docker + Open Source), and it *can* feel overwhelming at first. But once you have a solid **repeatable setup** like the one we just built, you'll find:

- Every app is just another **service block** in your `docker-compose.yaml`.
- Every database user/db is just another line in your `init.sql`.
- Every deployment is just `docker compose up -d`.

In short — you're building yourself a **personal PaaS** without depending on cloud vendor lock-in.

If you'd like, I can also:

1. Show you **macvlan fixed IP examples** (so each container has its own LAN IP like a real machine).
2. Add a **backup & restore workflow** for your `~/home/rarcayan/docker` folder.
3. Provide a **cheat sheet** of the most common Docker commands you'll use daily.

👉 Do you want me to prepare those next, or focus first on the macvlan network setup for your stack?