

AI-02074 Docker in Hyper-V Ubuntu









■ Owner	Ⓒ Rigel Arcayan
■ Tags	Project
■ Created time	@September 6, 2025 11:59 AM

I'm new to Ubuntu, Azure and Docker. I'm starting fresh in my home lab and I have a Hyper-v server running Ubuntu Server I want to deploy 4 docker applications such as OpenEMR, Wordpress, OpenProject, osTicket. Provide a step by step planning, deployment and installation guide for testing and evaluation.

Great! You're setting up a home lab using **Hyper-V**, **Ubuntu Server**, **Docker**, and deploying applications like **OpenEMR**, **WordPress**, **OpenProject**, and **osTicket** – all solid tools for various use cases (healthcare, blogging, project management, ticketing). I'll walk you through everything step-by-step assuming you're a beginner.

OVERVIEW

You will:

1.  Set up Ubuntu Server in Hyper-V
2.  Install Docker and Docker Compose
3.  Set up basic firewall and security
4.  Create Docker Compose files for each application
5.  Deploy and test OpenEMR, WordPress, OpenProject, and osTicket
6.  Access each app via browser
7.  Optionally set up volumes for persistent storage
8.  Evaluate and manage containers

PART 1: Setup Ubuntu Server on Hyper-V

1.1 Install Ubuntu Server in Hyper-V

1. Download the latest **Ubuntu Server ISO**: <https://ubuntu.com/download/server>
2. Create a **new VM in Hyper-V**:
 - Generation 2 (UEFI), 4GB RAM minimum
 - At least 2 vCPUs
 - 40GB virtual hard disk
3. Mount the ISO and boot the VM
4. Follow the prompts to install Ubuntu Server (use defaults or minimal install)

1.2 Set Static IP (Optional but Recommended)

```
sudo nano /etc/netplan/00-installer-config.yaml
```

Example config:

```
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: no
      addresses: [192.168.1.100/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [1.1.1.1, 8.8.8.8]
```

Then apply:

```
sudo netplan apply
```



PART 2: Install Docker & Docker Compose

2.1 Install Docker Engine

```
sudo apt update
sudo apt install -y docker.io
sudo systemctl enable --now docker
```

2.2 Install Docker Compose Plugin

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

2.3 Add Your User to Docker Group (to avoid using `sudo`)

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



PART 3: Security & Updates

3.1 Enable UFW (Uncomplicated Firewall)

```
sudo ufw allow OpenSSH
sudo ufw enable
```

Later you'll add:

```
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
```

3.2 System Update

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

PART 4: Create App Deployment Structure

Directory Structure (optional but tidy)

```
mkdir ~/docker-apps  
cd ~/docker-apps  
mkdir openemr wordpress openproject osticket
```

PART 5: Docker Compose Files

5.1 WordPress

Create `~/docker-apps/wordpress/docker-compose.yml`

```
version: '3.8'  
  
services:  
  wordpress:  
    image: wordpress:latest  
    ports:  
      - 8080:80  
    environment:  
      WORDPRESS_DB_HOST: db  
      WORDPRESS_DB_NAME: wordpress  
      WORDPRESS_DB_USER: wpuser  
      WORDPRESS_DB_PASSWORD: wppass  
    volumes:  
      - wp_data:/var/www/html
```

```
depends_on:
  - db

db:
  image: mysql:5.7
  environment:
    MYSQL_DATABASE: wordpress
    MYSQL_USER: wuser
    MYSQL_PASSWORD: wppass
    MYSQL_ROOT_PASSWORD: rootpass
  volumes:
    - db_data:/var/lib/mysql

volumes:
  wp_data:
  db_data:
```

Then deploy:

```
cd ~/docker-apps/wordpress
docker compose up -d
```

Access via: <http://<ubuntu-ip>:8080>

5.2 OpenEMR

Create <~/docker-apps/openemr/docker-compose.yml>

```
version: '3.8'

services:
  openemr:
    image: openemr/openemr
    restart: always
    ports:
      - 8081:80
```

```
environment:
  OE_USER: admin
  OE_PASS: pass
volumes:
  - openemr_sites:/var/www/localhost/htdocs/sites
```

```
mysql:
  image: mariadb
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: openemr
    MYSQL_USER: openemr
    MYSQL_PASSWORD: openemr
  volumes:
    - mysql_data:/var/lib/mysql
```

```
volumes:
  openemr_sites:
  mysql_data:
```

Deploy:

```
cd ~/docker-apps/openemr
docker compose up -d
```

Access via: <http://<ubuntu-ip>:8081>

5.3 OpenProject

Create <~/docker-apps/openproject/docker-compose.yml>

```
version: '3.8'

services:
  openproject:
```

```
image: openproject/community:latest
ports:
  - 8082:80
volumes:
  - openproject_data:/var/openproject/assets
restart: always
```

```
volumes:
  openproject_data:
```

Deploy:

```
cd ~/docker-apps/openproject
docker compose up -d
```

Access via: <http://<ubuntu-ip>:8082>



5.4 osTicket

Create <~/docker-apps/osticket/docker-compose.yml>

```
version: '3.8'

services:
  osticket:
    image: osticket/osticket
    ports:
      - 8083:80
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: osticket
      MYSQL_PASSWORD: osticketpass
      MYSQL_DATABASE: osticketdb
    depends_on:
      - mysql
```

```
mysql:
  image: mysql:5.7
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: osticketdb
    MYSQL_USER: osticket
    MYSQL_PASSWORD: osticketpass
  volumes:
    - mysql_data:/var/lib/mysql
```

```
volumes:
  mysql_data:
```

Deploy:

```
cd ~/docker-apps/osticket
docker compose up -d
```

Access via: `http://<ubuntu-ip>:8083`

PART 6: Accessing Applications

Application	URL
WordPress	<code>http://:8080</code>
OpenEMR	<code>http://:8081</code>
OpenProject	<code>http://:8082</code>
osTicket	<code>http://:8083</code>

- Replace `<ubuntu-ip>` with your server's IP address (e.g., `192.168.1.100`)

PART 7: Persistence and Backups

- Each Docker Compose file uses **named volumes** to persist data

- You can back them up like so:

```
docker volume ls
docker run --rm -v volume_name:/volume -v $(pwd):/backup ubuntu tar czf /
backup/volume_name.tar.gz /volume
```

PART 8: Managing Containers

Check logs:

```
docker compose logs -f
```

Restart or rebuild:

```
docker compose down
docker compose up -d
```

See all running containers:

```
docker ps
```

BONUS: Useful Tools

- **Portainer** (web GUI for managing containers):

```
docker volume create portainer_data
docker run -d -p 9000:9000 -p 8000:8000 \
  --name=portainer \
  --restart=always \
  -v /var/run/docker.sock:/var/run/docker.sock \
```

```
-v portainer_data:/data \
portainer/portainer-ce
```

Visit: <http://<ubuntu-ip>:9000>

Summary

Step	Task
1	Install Ubuntu in Hyper-V
2	Install Docker & Docker Compose
3	Configure firewall and security
4	Create Docker Compose files
5	Deploy and test apps
6	Access apps via browser
7	Backup volumes and manage containers

Would you like me to generate a downloadable version of this as a PDF or Markdown guide for your reference?

Versio 2

Direct answer

Yes — you can run all four apps (OpenEMR, WordPress, OpenProject, osTicket) on your Hyper-V Ubuntu Server by installing **Docker Engine + Docker Compose**, then running each app as its own `docker compose` stack (each stack has a DB container + app container + named volumes). Start simple (host port mapping) for evaluation; later you can add a reverse-proxy (Traefik or Nginx-Proxy-Manager) or move a stack to Azure. (How-to + example `docker compose` files, commands and troubleshooting below.) ([Docker Documentation](#))

Key reasons (short)

- Docker isolates each app and makes installs reproducible, easy to start/stop and to back up (volumes + DB dumps).
 - `docker compose` keeps app + DB + volumes in one directory/stack — ideal for testing.
 - OpenProject provides a maintained docker-compose repo (recommended) because it expects PostgreSQL and more services; OpenEMR & WordPress have official Docker images. ([OpenProject.org](https://openproject.org), [OpenEMR](https://openemr.org), [Docker Documentation](https://docs.docker.com))
-

Alternatives

- Use a reverse proxy (Traefik / Nginx Proxy Manager) to expose multiple apps on standard ports and hostnames.
 - Move to Azure for cloud testing: Azure VM (run Docker just like local) or Azure Container Instances for ephemeral/testing containers. For production-scale, consider AKS. ([Microsoft Learn](https://learn.microsoft.com))
-

Action Plan — step-by-step (planning → deploy → evaluate)

Assumptions I'll use for the guide: you have an Ubuntu Server VM on Hyper-V with SSH access and sudo privileges. If that's different, the steps are the same but paths and users may vary.

1) Plan / resource checklist (do this before you start)

- Host CPU: 2+ cores (4 recommended).
- RAM: **8 GB minimum** for light local testing; **16 GB** recommended if you run OpenProject + OpenEMR + others simultaneously. OpenProject in particular can need more for production. ([OpenProject.org](https://openproject.org))

- Disk: 50–200 GB depending on attachments and DB size. Use an SSD if possible.
- Network: static IP for your Ubuntu VM or set `/etc/hosts` entries for convenience.
- Backups: plan where to store DB dumps (another disk, NAS, or backup VM).

2) Prep the Ubuntu VM (packages, users)

Run these commands on your Ubuntu VM:

```
# update + basic tools
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg lsb-release git vim
```

3) Install Docker Engine + Compose plugin (official recommended method)

Follow Docker's official steps (I'll give the commands used most often):

```
# 1) Add Docker's GPG key & 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.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docke
r.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENA
ME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 2) Install Docker Engine + compose plugin
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

```
# 3) Optional: allow your user to run Docker without sudo
sudo usermod -aG docker $USER
# then logout/login or run:
newgrp docker
```

Verify:

```
docker run hello-world
docker compose version # shows the "docker compose" plugin
```

(These are the official Docker installation instructions for Ubuntu.) ([Docker Documentation](#))

4) Create a project folder structure (good practice)

Example structure:

```
/home/ubuntu/apps/
  wordpress/
    docker-compose.yml
  openemr/
    docker-compose.yml
  openproject/
    (clone the official repo here)
  osticket/
    docker-compose.yml
```

5) App stacks — example **docker compose** files

Important: these are starter examples for testing only. Change the passwords and persistent volume locations. For production, use TLS, secure passwords, and do NOT expose DB ports to the internet.

A) WordPress (quick test)

```
/home/ubuntu/apps/wordpress/docker-compose.yml
```

```
version: "3.8"
services:
  db:
    image: mysql:8.0
    restart: unless-stopped
    command: --default-authentication-plugin=mysql_native_password
    environment:
      MYSQL_ROOT_PASSWORD: wp_root_pass
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress_pass
    volumes:
      - db_data:/var/lib/mysql

  wordpress:
    image: wordpress:latest
    depends_on:
      - db
    ports:
      - "8080:80" # access http://<host-ip>:8080
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress_pass
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - wordpress_data:/var/www/html
    restart: unless-stopped

volumes:
  db_data:
  wordpress_data:
```

This follows the official Docker Compose WordPress sample. ([Docker Documentation](#))

B) OpenEMR (starter)

</home/ubuntu/apps/openemr/docker-compose.yml>

```
version: "3.8"
services:
  db:
    image: mariadb:10.5
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: openemr_root_pass
      MYSQL_DATABASE: openemr
      MYSQL_USER: openemr
      MYSQL_PASSWORD: openemr_pass
    volumes:
      - openemr_db:/var/lib/mysql

  openemr:
    image: openemr/openemr:latest
    depends_on:
      - db
    ports:
      - "8082:80" # access http://<host-ip>:8082
    environment:
      MYSQL_HOST: db
      MYSQL_ROOT_PASS: openemr_root_pass
      MYSQL_USER: openemr
      MYSQL_PASS: openemr_pass
      OE_USER: admin
      OE_PASS: pass
    volumes:
      - openemr_data:/var/www/localhost/htdocs
    restart: unless-stopped

volumes:
```

```
openemr_db:
openemr_data:
```

OpenEMR publishes official Docker images and example compose files — check their repo/wiki for full production options and extra services (redis, backups). ([OpenEMR](#), [GitHub](#))

C) osTicket (starter)

```
/home/ubuntu/apps/osticket/docker-compose.yml
```

```
version: "3.8"
services:
  db:
    image: mysql:5.7
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: ost_root_pass
      MYSQL_DATABASE: osticket
      MYSQL_USER: osticket
      MYSQL_PASSWORD: ost_pass
    volumes:
      - ost_db:/var/lib/mysql

  osticket:
    image: osticket/osticket:latest
    depends_on:
      - db
    ports:
      - "8084:80" # http://<host-ip>:8084
    environment:
      MYSQL_HOST: db
      MYSQL_DATABASE: osticket
      MYSQL_USER: osticket
      MYSQL_PASSWORD: ost_pass
    volumes:
```

```
- ost_uploads:/var/www/html/upload
restart: unless-stopped
```

```
volumes:
  ost_db:
  ost_uploads:
```

There are community images and docs for osTicket on Docker Hub — review those if you need plugin/language pack support. ([Docker Hub](#), [Mike Polinowski](#))

D) OpenProject (recommended: use official repo)

OpenProject maintains an **official docker-compose repository** that bundles all required services. For evaluation **clone and run their repo** rather than inventing your own compose:

```
cd /home/ubuntu/apps
git clone https://github.com/opf/openproject-docker-compose.git --depth=1 -
-branch=stable/16 openproject
cd openproject
cp .env.example .env # edit .env if you want (PORT, OPDATA path, etc.)
OPENPROJECT_HTTPS=false docker compose up -d --build --pull always
# then visit http://<host-ip>:8080 (default admin/admin)
```

OpenProject's docs recommend disabling HTTPS on first startup for local testing and include backup/upgrade commands. ([OpenProject.org](#))

6) Start the stacks

From each app folder:

```
cd /home/ubuntu/apps/wordpress
docker compose up -d

cd /home/ubuntu/apps/openemr
docker compose up -d
```

```
# For openproject use the commands shown earlier (repo)
```

Check status and logs:

```
docker compose ps
docker compose logs -f # or docker compose logs -f service-name
docker ps -a
```

7) Backups & maintenance (basic)

- MySQL/MariaDB backup (example):

```
# run from host
docker compose exec db mysqldump -u root -p'rootpassword' wordpress > w
ordpress.sql
# for osticket/openemr: change DB name
```

- PostgreSQL backup for OpenProject (uses the repo control scripts; see repo backup commands). ([OpenProject.org](https://openproject.org))
- To stop & remove containers but keep volumes:

```
docker compose down
# remove volumes too:
docker compose down -v
```

8) Troubleshooting tips

- If `docker compose up` fails: `docker compose logs` and `docker compose ps` show errors.
- DB connection errors: check env vars and `depends_on` (may need to wait for DB to finish init). Use `docker compose logs db`.
- Permission/volume issues: check file ownership (some containers expect UID 1000 for mounted folders).

- Port conflicts: map to free host ports (8080, 8081, 8082, 8084) or use a reverse proxy for hostnames.
-

9) Next steps / improvements (after you test)

- Add a reverse proxy (Traefik or Nginx-Proxy-Manager) so you can use friendly hostnames (e.g., `openemr.local`) and HTTPS.
 - Centralize monitoring: add Portainer for container GUI, and consider Prometheus + Grafana for metrics.
 - If you move to Azure: quick options are Azure VM with Docker installed (same steps), Azure Container Instances for short tests, or AKS for orchestration. ([Microsoft Learn](#))
-

10) Security & production notes (short)

- Never use example passwords in production.
 - Use TLS and a reverse proxy for exposed apps. OpenProject expects TLS behind a proxy in production (docs note). ([OpenProject.org](#))
 - Harden DB & container user permissions and secure backups.
-

If you want, I can:

- Produce fully populated `docker-compose.yml` files with secure `.env` examples and suggested volumes for your actual VM paths, **or**
- Provide a single combined `docker compose` with a Traefik reverse proxy and example `hosts` entries so you can access the apps via hostnames (e.g., `wordpress.test`, `openemr.test`).

Which of those two would you like right now? (I can generate the files and commands immediately.)

Cheatsheet for Commands

Got it — here's a **cheatsheet** style summary of all the key commands you'll need for your Ubuntu + Docker home lab setup (WordPress, OpenEMR, OpenProject, osTicket).

Ubuntu + Docker Lab Cheatsheet

◆ System Prep

```
# Update system
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg lsb-release git vim
```

◆ Install Docker + Compose

```
# Add Docker repo + key
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
or -o /etc/apt/keyrings/docker.asc
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/do
cker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAM
E}") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
# Install Docker + Compose plugin
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

```
# Allow user to run docker without sudo
sudo usermod -aG docker $USER
newgrp docker
```

◆ Verify Install

```
docker run hello-world
docker compose version
```

Folder Setup

```
mkdir -p ~/apps/{wordpress,openemr,openproject,osticket}
cd ~/apps
```

Deploy Applications

WordPress

```
cd ~/apps/wordpress
nano docker-compose.yml # paste sample compose
docker compose up -d
```

Access: <http://<host-ip>:8080>

OpenEMR

```
cd ~/apps/openemr
nano docker-compose.yml # paste sample compose
docker compose up -d
```

Access: <http://<host-ip>:8082>

osTicket

```
cd ~/apps/osticket
nano docker-compose.yml # paste sample compose
docker compose up -d
```

Access: <http://<host-ip>:8084>

OpenProject (official repo)

```
cd ~/apps
git clone https://github.com/opf/openproject-docker-compose.git --branch=stable/16 openproject
cd openproject
cp .env.example .env
OPENPROJECT_HTTPS=false docker compose up -d --build --pull always
```

Access: <http://<host-ip>:8080> (default admin/admin)



Management Commands

Show running containers

```
docker ps
```

Stop containers (in project folder)

```
docker compose down
```

Stop & remove volumes too

```
docker compose down -v
```

View logs

```
docker compose logs -f
```

```
docker compose logs -f service-name
```

Restart containers

```
docker compose restart
```

List docker images

```
docker images
```

List docker volumes

```
docker volume ls
```

Backup (examples)

```
# MySQL/MariaDB dump (WordPress/OpenEMR/osTicket)
docker compose exec db mysqldump -u root -p'rootpass' dbname > backup.sql
```

```
# OpenProject (PostgreSQL) backup
docker compose exec db pg_dump -U postgres openproject > openproject.sql
```

Troubleshooting

```
docker compose ps      # check status
docker compose logs    # check logs
docker inspect <container> # details
```

👉 That's the **minimum cheat sheet** you can keep beside you while testing.

Do you want me to also generate **ready-to-paste compose files** (WordPress, OpenEMR, osTicket) with `.env` files for passwords so you don't edit inside YAML? That way your cheat sheet stays clean.