

AI-02073 Docker in Azure Deployment

Owner	© Rigel Arcayan
Tags	Project
Created time	@September 6, 2025 10:58 AM

I'm new to Ubuntu, Azure and Docker. I'm starting fresh in my Azure Account and 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. Be cost conscious with the lowest possible per month operating expenses.

Welcome to the world of Ubuntu, Azure, and Docker! You're starting strong with useful open-source applications. I'll guide you **step-by-step** through planning, deployment, and testing — with a strong focus on **cost minimization**. The apps you want to deploy are:

- **OpenEMR** (electronic medical records)
- **WordPress** (CMS/blog)
- **OpenProject** (project management)
- **osTicket** (support ticket system)

Overview of the Deployment Strategy

Key Goals:

- Keep it **simple, low-cost**, and easy to manage
- Use **1 Ubuntu VM** with **Docker** and **Docker Compose**
- Each app runs in its own container stack using **Docker Compose**
- Use **Azure B1s VM** (low-cost option with enough for testing)

- Use **basic disk + minimal bandwidth**
- Optional: **Free-tier Azure Database for MySQL/Postgres** or run DBs in **containers**

Monthly Cost Estimation (as of 2025)

Service	Description	Cost (approx)
Azure B1s VM (Linux)	1 vCPU, 1 GB RAM	~\$5/month
Standard SSD Disk (30GB)	OS + Docker volumes	~\$2/month
Static IP (1 included)	1 Public IP	Free
Bandwidth	First 5GB/month	Free
Total	Base cost for 1 VM	~\$7/month

 You can shut down the VM when not testing to save more.

Step-by-Step Guide

1. Planning

App	Default Ports	DB	Notes
WordPress	80/443	MySQL	Well-supported
OpenEMR	80/443	MySQL	PHP-based
OpenProject	8080	PostgreSQL	Heavy RAM usage
osTicket	80/443	MySQL	Also PHP-based

Because OpenProject is heavier, the B1s instance will be **tight on RAM**, but OK for testing. Avoid running all 4 **at once** if RAM is exhausted.

2. Provision Ubuntu VM in Azure

1. **Login to Azure Portal:** <https://portal.azure.com>
2. Go to **"Virtual Machines" > "Create VM"**
3. Choose:

- **Region:** Near you (to reduce latency)
- **Image:** Ubuntu 22.04 LTS
- **Size:** **B1s** (1vCPU, 1GB RAM)
- **Authentication:** SSH key (recommended)
- **Public IP:** Yes (static)
- **Disk:** Standard SSD (30 GB)

4. Networking:

- Open ports: 22 (SSH), 80 (HTTP), 443 (HTTPS), 8080 (OpenProject)

5. Review and create.

3. Connect to VM via SSH

```
ssh azureuser@your_vm_ip
```

4. Install Docker & Docker Compose

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Allow current user to run Docker
sudo usermod -aG docker $USER
newgrp docker

# Install Docker Compose plugin
sudo apt install docker-compose-plugin -y
```

Test it:

```
docker --version
docker compose version
```

5. Setup Directory Structure

```
mkdir ~/docker-apps
cd ~/docker-apps

# Create a folder per app
mkdir wordpress openemr openproject osticket
```

6. Docker Compose Files

Here's a sample `docker-compose.yml` per app:

a) WordPress (`~/docker-apps/wordpress/docker-compose.yml`)

```
version: '3.8'
services:
  wordpress:
    image: wordpress
    restart: always
    ports:
      - "8081:80"
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wpuser
      WORDPRESS_DB_PASSWORD: wppassword
      WORDPRESS_DB_NAME: wpdb
    volumes:
      - wordpress_data:/var/www/html
  db:
    image: mysql:5.7
    restart: always
```

```
environment:
  MYSQL_DATABASE: wpdb
  MYSQL_USER: wpuser
  MYSQL_PASSWORD: wppassword
  MYSQL_ROOT_PASSWORD: rootpass
volumes:
  - db_data:/var/lib/mysql
volumes:
wordpress_data:
db_data:
```

b) OpenEMR (`~/docker-apps/openemr/docker-compose.yml`)

```
version: '3.8'
services:
  openemr:
    image: openemr/openemr
    ports:
      - "8082:80"
    environment:
      MYSQL_HOST: mysql
      MYSQL_ROOT_PASS: root
      OE_USER: admin
      OE_PASS: pass
    depends_on:
      - mysql
  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: openemr
    volumes:
      - openemr_db:/var/lib/mysql
```

```
volumes:  
  openemr_db:
```

c) OpenProject (`~/docker-apps/openproject/docker-compose.yml`)

```
version: '3.8'  
services:  
  openproject:  
    image: openproject/community:latest  
    ports:  
      - "8083:8080"  
    volumes:  
      - openproject_pgdata:/var/openproject/pgdata  
      - openproject_assets:/var/openproject/assets  
    restart: always  
volumes:  
  openproject_pgdata:  
  openproject_assets:
```

d) osTicket (`~/docker-apps/osticket/docker-compose.yml`)

```
version: '3'  
services:  
  osticket:  
    image: osticket/osticket  
    ports:  
      - "8084:80"  
    environment:  
      MYSQL_HOST: osticket-db  
      MYSQL_USER: osticket  
      MYSQL_PASSWORD: osticketpass  
      MYSQL_DATABASE: osticketdb  
    depends_on:  
      - osticket-db
```

```
osticket-db:
  image: mariadb
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: osticketdb
    MYSQL_USER: osticket
    MYSQL_PASSWORD: osticketpass
  volumes:
    osticket-db:
```

7. Running Applications

In each app folder:

```
docker compose up -d
```

Check if it's running:

```
docker ps
```

Then access in your browser:

App	URL
WordPress	http://your_vm_ip:8081
OpenEMR	http://your_vm_ip:8082
OpenProject	http://your_vm_ip:8083
osTicket	http://your_vm_ip:8084

8. Testing and Evaluation Tips

- Check logs: `docker compose logs -f`
- Stop/start app: `docker compose stop` / `docker compose start`
- Test one or two apps at a time to save RAM
- You can use `swap` to prevent crashes from low memory:

```
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
echo "/swapfile none swap sw 0 0" | sudo tee -a /etc/fstab
```

9. Shutdown for Cost Savings

To avoid charges when not using:

```
# In Azure Portal
Stop the VM (preserves data but stops billing for compute)
```

Or on terminal:

```
sudo shutdown now
```

10. Secure Your Setup (Optional but Recommended)

- Install **fail2ban**, **ufw**
- Use **Let's Encrypt** and Nginx reverse proxy to expose apps on HTTPS
- Set up **basic monitoring** with Netdata or Glances

Clean Up (When Done Testing)

```
docker system prune -a --volumes
```

In Azure:

- Deallocate or delete VM
- Delete resource group if you want to remove everything

✓ Summary

Step	Task
1	Create low-cost Ubuntu VM
2	Install Docker and Compose
3	Prepare Docker Compose files
4	Run apps one by one for evaluation
5	Shut down VM when idle to reduce cost

Would you like a downloadable script or GitHub repo with all these Compose files and a setup script?

Version 2

Direct answer — short plan + outcome

Yes — you can run **OpenEMR, WordPress, OpenProject and osTicket** on your Ubuntu Server VM in Hyper-V using **Docker + Docker Compose**.

Below I give a complete, copy-pasteable **planning checklist**, the exact **Ubuntu/Docker install commands, production-like docker compose stacks** (with persistent volumes and networks), verification steps and quick **backup/upgrade** recipes so you can test and evaluate safely. I also recommend a simple reverse-proxy (Nginx Proxy Manager) so all apps can share ports and get HTTPS later. ([Docker Documentation](#))

Key reasons (concise)

1. **Docker + Compose** isolates each app, simplifies upgrades/backups, and is the fastest way to test web apps. ([Docker Documentation](#))
2. Use a **reverse proxy** (Nginx Proxy Manager or Traefik) when you want multiple web apps accessible on standard 80/443 ports and to add HTTPS (LetsEncrypt) later. ([nginxproxymanager.com](#), [Traefik Docs](#))
3. Give **OpenProject >=4GB RAM** (it is the heaviest here); OpenEMR needs a compatible PHP/MariaDB stack; osTicket needs PHP 8.x. Plan VM sizing

accordingly. ([OpenProject.org](https://openproject.org), [OpenEMR](https://openemr.org), docs.osticket.com)

4. **Do not expose PHI (OpenEMR) to the open Internet** — test on LAN or secure VPN only. ([OpenEMR](https://openemr.org))

Alternatives (short)

- If you want orchestration and plan to scale: use **k3s / k8s** (more complex).
- If you prefer GUI management: add **Portainer** to manage containers.
- If you want no containers: install each app on a LAMP/LEMP stack (manual, slower to maintain).

Action plan — step-by-step (copy/paste ready)

Notes before you start

Work from the Ubuntu Server VM terminal (sudo or your non-root sudo user).

Use Ubuntu 22.04 LTS or later if possible. Allocate at least **4 vCPU / 8–16 GB RAM / 100+ GB SSD** for comfortable testing (OpenProject needs ~4GB min).

([Docker Documentation](https://docs.docker.com), [OpenProject.org](https://openproject.org))

0 — Hyper-V / VM checklist

1. Create an **External Virtual Switch** in Hyper-V so the VM is reachable from your LAN.
2. Give the Ubuntu VM a **static IP** on your LAN (or DHCP reservation).
3. Assign CPU (2–4) and RAM (8 GB recommended) and an SSD virtual disk ($\geq 100\text{GB}$).
4. Snapshot the VM before big changes (Hyper-V checkpoint) so you can revert.

1 — Ubuntu base setup (commands)

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

2 — Install Docker Engine (official method)

(Using Docker's apt repository — recommended). See Docker docs. ([Docker Documentation](#))

```
# add Docker official GPG, repo and install
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
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 > /dev/null

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

- Verify: `docker run --rm hello-world`
- If `docker-compose-plugin` isn't available on your release, install the Compose CLI as a user plugin (alternate method in Compose docs). ([Docker Documentation](#))

Make Docker usable without sudo:

```
sudo usermod -aG docker $USER
# then log out & log in again (or use `newgrp docker` in the current shell)
```

3 — Networking & directories

```
# base folders
sudo mkdir -p /srv/docker/{nginx-proxy-manager,wordpress,openemr,openproject,osticket}
sudo chown -R $USER:$USER /srv/docker

# docker bridge network for proxy+apps
docker network create proxy
```

4 — Install a reverse proxy (Nginx Proxy Manager) — recommended for home lab

Why: simple web UI to forward domains/subdomains to containers and manage HTTPS later. See setup docs. (nginxproxymanager.com)

`/srv/docker/nginx-proxy-manager/docker-compose.yml` :

```
version: "3"
services:
  app:
    image: jc21/nginx-proxy-manager:latest
    restart: unless-stopped
    ports:
      - "80:80" # HTTP
      - "81:81" # admin UI
      - "443:443" # HTTPS
    environment:
      DB_SQLITE_FILE: "/data/database.sqlite"
    volumes:
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt
    networks:
      - proxy
networks:
```

```
proxy:
  external: true
```

Start:

```
cd /srv/docker/nginx-proxy-manager
docker compose up -d
# Default admin: admin@example.com / changeme (change immediately). :co
ntentReference[oaicite:9]{index=9}
```

5 — WordPress stack (quick test)

Official example (Docker sample). Use MariaDB and WordPress images. If using proxy, you can **omit** `ports:` and expose only on the `proxy` network, then create a proxy host in Nginx Proxy Manager. For quick local test with port mapping, use the `ports` version.

`/srv/docker/wordpress/docker-compose.yml` (port-mapped quick test):

```
version: "3.8"
services:
  db:
    image: mariadb:10.6
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: example_root_pw
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wp_user
      MYSQL_PASSWORD: wp_pass
    volumes:
      - db_data:/var/lib/mysql

  wordpress:
    image: wordpress:php8.1-fpm
    depends_on:
      - db
```

```
restart: unless-stopped
environment:
  WORDPRESS_DB_HOST: db:3306
  WORDPRESS_DB_USER: wp_user
  WORDPRESS_DB_PASSWORD: wp_pass
  WORDPRESS_DB_NAME: wordpress
ports:
  - "8081:80"
volumes:
  - wp_data:/var/www/html
networks:
  - proxy
```

```
volumes:
  db_data:
  wp_data:
```

```
networks:
  proxy:
    external: true
```

Start:

```
cd /srv/docker/wordpress
docker compose up -d
# WordPress reachable at http://<vm-ip>:8081 (or setup proxy host for e.g. w
ordpress.local). :contentReference[oaicite:10]{index=10}
```

6 — OpenEMR stack (recommended: MariaDB + OpenEMR image)

Use the official OpenEMR Docker image. For production or PHI, **do not** expose to Internet unsecured. Official image docs and examples on Docker Hub / GitHub. ([Docker Hub](#), [OpenEMR](#))

/srv/docker/openemr/docker-compose.yml :

```
version: "3.8"
services:
  db:
    image: mariadb:10.5
    restart: unless-stopped
    command: ['--character-set-server=utf8mb4', '--collation-server=utf8mb4
_unicode_ci']
    environment:
      MYSQL_ROOT_PASSWORD: openemr_root_pw
      MYSQL_DATABASE: openemr
      MYSQL_USER: openemr
      MYSQL_PASSWORD: openemr_pw
    volumes:
      - openemr_db:/var/lib/mysql

  openemr:
    image: openemr/openemr:latest
    depends_on:
      - db
    restart: unless-stopped
    environment:
      MYSQL_HOST: db
      MYSQL_ROOT_PASS: openemr_root_pw
      MYSQL_USER: openemr
      MYSQL_PASS: openemr_pw
      OE_USER: admin
      OE_PASS: pass
    ports:
      - "8300:80" # quick test; remove if behind proxy
    volumes:
      - openemr_data:/var/www/localhost/htdocs
    networks:
      - proxy
```

```
volumes:
  openemr_db:
  openemr_data:

networks:
  proxy:
    external: true
```

Start:

```
cd /srv/docker/openemr
docker compose up -d
# OpenEMR web installer should be available; default web login often admin/p
ass in test mode. Verify official docs for the image. :contentReference[oaicite:
12]{index=12}
```

7 — OpenProject (simple quick test)

OpenProject provides a docker-compose installation method and an **all-in-one** container for quick testing — recommended for evaluation. OpenProject is heavier; ensure at least 4GB RAM allocated. ([OpenProject.org](https://openproject.org))

Quick start (recommended from OpenProject docs):

```
cd /srv/docker
git clone --depth=1 https://github.com/opf/openproject-deploy.git openproject
cd openproject
# Follow the README in that repo; usually:
docker compose up -d
# OpenProject default admin/admin for first login (change immediately). :cont
entReference[oaicite:14]{index=14}
```

(If you prefer the single all-in-one container, see the official OpenProject "all-in-one" doc in their install docs.) ([OpenProject.org](https://openproject.org))

8 — osTicket

There are community Docker images (tiredofit, devinsolutions). They usually auto-run the installer on first boot. Use a stable image and mount persistent volumes. Example below uses a commonly used image (adjust tags/version to the latest stable). ([GitHub](#))

`/srv/docker/osticket/docker-compose.yml` :

```
version: "3.8"
services:
  db:
    image: mariadb:10.6
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: ost_root_pw
      MYSQL_DATABASE: osticket
      MYSQL_USER: ost
      MYSQL_PASSWORD: ost_pw
    volumes:
      - ost_db:/var/lib/mysql

  osticket:
    image: devinsolutions/osticket:latest
    depends_on:
      - db
    restart: unless-stopped
    environment:
      MYSQL_HOST: db
      MYSQL_DATABASE: osticket
      MYSQL_USER: ost
      MYSQL_PASSWORD: ost_pw
      INSTALL_EMAIL: admin@example.com
    ports:
      - "7777:80"
    volumes:
      - ost_data:/data
```

```
networks:  
  - proxy
```

```
volumes:  
  ost_db:  
  ost_data:
```

```
networks:  
  proxy:  
    external: true
```

Start:

```
cd /srv/docker/osticket  
docker compose up -d  
# Default admin credentials are container-specific; many images set ostadmin/Admin1 or ostadmin/Admin1 — check the image README. :contentReference [oaicite:17]{index=17}
```

9 — Basic verification / logs

```
docker ps  
docker compose -f docker-compose.yml logs -f <service-name>  
# Example: docker compose logs -f openemr
```

10 — Backups & upgrades (quick recipes)

MySQL/MariaDB logical dump

```
# dump DB (replace container name)  
docker exec -i $(docker ps -qf "name=<db-container-name>") mysqldump -u root -p'ROOTPW' wordpress > /srv/backups/wordpress.sql
```

Volume tarball (file-level)

```
docker run --rm -v openemr_db:/volume -v /srv/backups:/backup busybox \
sh -c "cd /volume && tar -czvf /backup/openemr_db.tgz ."
```

Upgrade containers

```
docker compose pull
docker compose up -d
# Review app upgrade docs (esp. OpenEMR and OpenProject) before major v
ersion upgrades. :contentReference[oaicite:18]{index=18}
```

11 — Housekeeping & security notes

- **Do not** expose OpenEMR or other PHI-containing apps to the public Internet without full security/hardening and legal compliance. Keep them behind VPN or LAN for testing. ([OpenEMR](#))
- Use firewall (`ufw`) to allow SSH and proxy ports only: `sudo ufw allow ssh; sudo ufw allow 80,443,81/tcp; sudo ufw enable` .
- Regularly snapshot the VM before upgrades.

Quick troubleshooting hints

- If `docker` commands require sudo, re-login after `usermod -aG docker $USER` .
- If `docker compose` is not found, try `docker-compose` (older), or install Compose plugin as documented. ([Docker Documentation](#), [DigitalOcean](#))
- Check container logs: `docker logs -f <container>` .

Helpful references (official docs I used)

- Docker Engine install on Ubuntu (official). ([Docker Documentation](#))
- Docker Compose install & usage. ([Docker Documentation](#))
- OpenEMR Docker Hub / Docker README. ([Docker Hub](#))

- OpenProject Docker Compose docs & system requirements. ([OpenProject.org](https://openproject.org))
- osTicket Docker community images (Devinsolutions / tiredofit). ([GitHub](https://github.com))
- WordPress Docker sample (official). ([Docker Documentation](https://docs.docker.com))
- Nginx Proxy Manager setup docs. (nginxproxymanager.com)

If you want I can:

- produce **ready-to-use** `docker-compose.yml` **files** (I already included one-off examples) tuned to specific versions and hostnames you want, or
- provide a **single combined docker compose repo** with all stacks and a README that you can `git clone` on your VM, or
- walk you through converting the stacks to be **behind the proxy** (remove ports and register proxy hosts) and set up **local DNS / /etc/hosts** entries so `openemr.local`, `wordpress.local`, etc. resolve.

Which of those would you like next? (I can generate a complete repo of compose files and a one-file deployment script right away.)

Got it 👍 — here's a **clean Docker + Ubuntu Home Lab Cheatsheet** tailored to your OpenEMR, WordPress, OpenProject, and osTicket test setup.

Ubuntu + Docker Home Lab Cheatsheet

◆ System basics

```
# Update packages
```

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

```
# Install essentials
```

```
sudo apt install -y ca-certificates curl gnupg lsb-release git vim
```

```
# Reboot the VM
sudo reboot
```

◆ Docker install & setup

```
# Add Docker GPG + 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/do
cker.gpg] \
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" \
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

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

```
# Test
docker run --rm hello-world
```

```
# Run Docker without sudo
sudo usermod -aG docker $USER
newgrp docker
```

◆ Docker basics

```
# Run container (test)
docker run -it ubuntu bash
```

```
# List containers (running / all)
docker ps
```

```
docker ps -a

# Stop / Start / Restart container
docker stop <id>
docker start <id>
docker restart <id>

# Remove container
docker rm <id>

# List images
docker images

# Remove image
docker rmi <image-id>

# Container logs
docker logs -f <container-name>

# Enter a container shell
docker exec -it <container-name> bash
```

◆ Docker Compose basics

```
# Start stack
docker compose up -d

# Stop stack
docker compose down

# View logs
docker compose logs -f

# Restart after config change
```

```
docker compose up -d --build
```

```
# Update to latest images
```

```
docker compose pull
```

```
docker compose up -d
```

◆ Volumes & networks

```
# List volumes / networks
```

```
docker volume ls
```

```
docker network ls
```

```
# Inspect volume
```

```
docker volume inspect <volume-name>
```

```
# Create custom network
```

```
docker network create proxy
```

◆ Backups

```
# Dump database (example: WordPress DB)
```

```
docker exec -i <db-container> mysqldump -u root -p'ROOTPW' wordpress >  
wordpress.sql
```

```
# Backup a volume
```

```
docker run --rm -v <volume-name>:/volume -v $(pwd):/backup busybox \  
sh -c "cd /volume && tar -czf /backup/<volume-name>.tgz ."
```

◆ Reverse Proxy (Nginx Proxy Manager)

```
# Start Nginx Proxy Manager stack
```

```
cd /srv/docker/nginx-proxy-manager
```

```
docker compose up -d
```

```
# Access web UI
```

```
http://<vm-ip>:81
```

```
# Default login: admin@example.com / changeme
```

◆ App access (default test ports in your setup)

- **WordPress** → `http://<vm-ip>:8081`
- **OpenEMR** → `http://<vm-ip>:8300`
- **OpenProject** → `http://<vm-ip>:8080` (or proxy host if set)
- **osTicket** → `http://<vm-ip>:7777`
- **Nginx Proxy Manager** → `http://<vm-ip>:81`

◆ Useful Linux system commands

```
# Check IP of VM
```

```
ip a
```

```
# Check open ports
```

```
ss -tulnp
```

```
# Check disk space
```

```
df -h
```

```
# Check running processes
```

```
top # or htop if installed
```

👉 Would you like me to compress this into a **one-page printable PDF cheatsheet** (landscape, boxes for each section) so you can keep it by your terminal while you work?


I need assistance on which plan to setup in Azure with the following objectives
1. Ubuntu server 2. Max Azure spending per month=\$10 3. Docker OpenEMR, Wordpress, osTicket, openproject. Tell me the configuration and then step by step from start to finish

Thanks for waiting, Rigel! Based on your goals and budget, here's a complete recommendation and step-by-step guide to set up an **Ubuntu server on Azure** with **Docker containers** for OpenEMR, WordPress, osTicket, and OpenProject — all within a **\$10/month budget**.

 Recommended Azure Configuration

VM Type: Standard_B1ls

- **Specs:** 1 vCPU, 0.5 GiB RAM
- **Cost:** ~\$3.80/month (Linux VM) [1](#)
- **OS:** Ubuntu 20.04 or 22.04 LTS
- **Disk:** Use the smallest possible OS disk (30 GB Standard HDD)
- **Docker:** Install Docker Engine manually or use a preconfigured image [2](#)

 **Note:** This VM is very lightweight. Running all four apps simultaneously may be slow. Consider running only one or two at a time, or using Docker Compose to manage resource allocation.

 Step-by-Step Setup Guide

Step 1: Create the VM

1. Go to Azure Portal
2. Click "**Create a resource**" > "**Virtual Machine**"
3. Choose:
 1. **Image:** Ubuntu Server 20.04 LTS
 2. **Size:** Standard_B1ls
 3. **Authentication:** SSH public key or password

4. **Disk:** Standard HDD (30 GB)
 5. **Networking:** Allow SSH (port 22)
 4. Click **Review + Create**, then **Create**
-

Step 2: Install Docker

SSH into your VM:

ssh yourusername@your-vm-ip Install Docker:

```
sudo apt update sudo apt install docker.io -y sudo systemctl start docker sudo  
systemctl enable docker Verify Docker:
```

```
docker --version
```

Step 3: Pull and Run Containers

You can use Docker Compose or run containers individually. Here's a basic example using Docker CLI:

OpenEMR

```
docker run -d -p 8080:80 --name openemr openemr/openemr WordPress
```

```
docker run -d --name wordpress -p 8081:80 -e  
WORDPRESS_DB_HOST=yourdbhost -e WORDPRESS_DB_USER=user -e  
WORDPRESS_DB_PASSWORD=pass -e WORDPRESS_DB_NAME=dbname  
wordpress osTicket
```

```
docker run -d -p 8082:80 --name osticket osticket/osticket OpenProject
```

```
docker run -d -p 8083:80 --name openproject openproject/community:latest You  
may need to configure volumes and databases separately for each app. For  
production use, Docker Compose is recommended.
```

Step 4: Access Your Apps

Use your VM's public IP and the ports you mapped:

- OpenEMR: <http://your-vm-ip:8080>
- WordPress: <http://your-vm-ip:8081>

- osTicket: <http://your-vm-ip:8082>
 - OpenProject: <http://your-vm-ip:8083>
-

Step 5: Optimize for Budget

- **Turn off VM when not in use** to save costs.
 - **Monitor usage** via Azure Cost Management.
 - **Use Azure Spot Instances** if acceptable for non-critical workloads.
-

Would you like help creating a **Docker Compose file** to manage all four apps together? Or do you want to explore **alternative hosting options** (like Oracle Cloud Free Tier or Linode) that might offer more resources for free or under \$10/month?

1: https://cloudprice.net/vm/Standard_B1ls 2:

<https://cloudinfrastructureservices.co.uk/how-to-setup-docker-engine-containers-on-linux-in-azure-aws-gcp/>