AI-02084 OpenEMR in Azure Docker

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	@September 8, 2025 11:14 AM

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 Mariadb database as backend db

3. Run docker OpenEMR as Frontend

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

Direct answer — short

Yes — you can run **one Azure Ubuntu VM** and host **two standalone Docker containers** on it: a **MariaDB** container (backend) and the **OpenEMR** container (frontend).

Below I give a single, copy-pasteable **production-style** `docker-compose.yml`, the exact shell commands to create the VM, install Docker + Compose, deploy the stack, verify it, and recommended security/backup steps. The OpenEMR repo provides an official `docker-compose.yml` example you can base this on. (repo.or.cz)

Key reasons (concise)

1. Docker keeps OpenEMR and MariaDB isolated (easier upgrades/backups).

2. Use Docker volumes so database & site files persist across container restarts.
3. Don’t expose the DB port to the internet — only expose HTTP/HTTPS for the app.
4. Use Docker Compose to declare both services and `depends_on: condition: service_healthy` so OpenEMR waits for MariaDB. (OpenEMR’s production compose does this.) (repo.or.cz)

Alternatives

- **Run MariaDB as an Azure managed database** (better for production/backup/HA).
- **Use a reverse proxy (nginx/Traefik) + Let’s Encrypt** container for automatic TLS.
- **Run each service on separate VMs** if you need stronger isolation or different scaling.

Action plan — step-by-step (copy-paste friendly)

> Notes before you start
>
> - This guide assumes Ubuntu 22.04 LTS (or similar). Use strong passwords and **do not** expose port 3306 (MariaDB) to the public internet.
> - Commands use `docker compose` (Compose V2 plugin). Follow Docker’s Ubuntu install instructions when needed. (Docker Documentation)

---

0) Pre-work: create an Azure VM (portal or CLI)

Quick Azure CLI example (replace names):

```
# login first if needed
az login
```

```
# create resource group
az group create -n myRG -l eastus

# create VM (Ubuntu LTS), auto-generate SSH keys
az vm create \
  -g myRG \
  -n openemr-vm \
  --image UbuntuLTS \
  --size Standard_B1ms \
  --admin-username azureuser \
  --generate-ssh-keys
```

After the VM is created, open ports 80 and 443 (do **not** open 3306). Example CLI to open ports:

```
az vm open-port -g myRG -n openemr-vm --port 80
az vm open-port -g myRG -n openemr-vm --port 443
# keep SSH (22) open by default, but restrict source IP if possible
```

(You can also add more restrictive NSG rules via the Portal.) (Microsoft Learn)

---

1) SSH into the VM

From your workstation:

```
ssh azureuser@<VM_PUBLIC_IP>
```

---

2) Install Docker Engine + Docker Compose plugin (official steps)

Run these commands on the Ubuntu VM (taken from Docker's official Ubuntu installation guidance):

```
# update & install prerequisites
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg lsb-release

# add Docker 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-get update

# install docker engine + compose plugin
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-pl
ugin docker-compose-plugin

# add your user to docker group so you can run docker without sudo (you will
need to logout/login)
sudo usermod -aG docker $USER
newgrp docker
```

Verify:

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

(See Docker docs for the canonical procedure and troubleshooting.) (Docker Documentation)

3) Create a project directory and `.env` file (separate secrets)

On the VM:

```
mkdir ~/openemr && cd ~/openemr
```

Create a file named `.env` (example — **change all passwords to strong values**):

```
# .env
MYSQL_ROOT_PASSWORD=SuperSecretRootPass!
MYSQL_DATABASE=openemr
MYSQL_USER=openemr
MYSQL_PASSWORD=OpenEmrDBpass!
OE_USER=admin
OE_PASS=pass
```

> Security tip: for production, use a secrets manager or Azure Key Vault, not plaintext .env.

---

4) `docker-compose.yml` (recommended production example)

Create `docker-compose.yml` in `~/openemr` using the official production pattern (this example is adapted from OpenEMR's production compose). Save the file exactly as below (it uses the `.env` variables created above):

```
version: '3.8'

services:
  mysql:
    image: mariadb:11.4
    command: ['mariadbd','--character-set-server=utf8mb4']
    restart: always
    volumes:
```

```
    - databasevolume:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MARIADB_DATABASE: ${MYSQL_DATABASE}
    MARIADB_USER: ${MYSQL_USER}
    MARIADB_PASSWORD: ${MYSQL_PASSWORD}
  healthcheck:
    test:
      - CMD
      - /usr/local/bin/healthcheck.sh
      - --su-mysql
      - --connect
      - --innodb_initialized
    start_period: 1m
    interval: 1m
    timeout: 5s
    retries: 3

openemr:
  image: openemr/openemr:7.0.3
  restart: always
  ports:
    - "80:80"
    - "443:443"
  volumes:
    - logvolume01:/var/log
    - sitevolume:/var/www/localhost/htdocs/openemr/sites
  environment:
    MYSQL_HOST: mysql
    MYSQL_ROOT_PASS: ${MYSQL_ROOT_PASSWORD}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASS: ${MYSQL_PASSWORD}
    OE_USER: ${OE_USER}
    OE_PASS: ${OE_PASS}
  depends_on:
    mysql:
```

```
        condition: service_healthy
    healthcheck:
      test:
        - CMD
        - /usr/bin/curl
        - --fail
        - --insecure
        - --location
        - --show-error
        - --silent
        - https://localhost/
      start_period: 3m
      interval: 1m
      timeout: 5s
      retries: 3

volumes:
  logvolume01: {}
  sitevolume: {}
  databasevolume: {}
```

This compose file mirrors the official OpenEMR production example; OpenEMR expects a companion MySQL/MariaDB container and environment variables such as `MYSQL_HOST` and `MYSQL_ROOT_PASS`. (repo.or.cz, Docker Hub)

---

5) Start the stack

From `~/openemr`:

```
# pull images (optional)
docker compose pull

# bring up services in background
docker compose up -d
```

Watch logs while the database initializes:

```
docker compose ps
docker compose logs -f mysql
docker compose logs -f openemr
```

Important: the first startup can take several minutes while MariaDB initializes and OpenEMR runs initial migrations/installs.

---

6) Visit OpenEMR and finish setup

Open a browser to: `http://<VM_PUBLIC_IP>/` (or `https://<VM_PUBLIC_IP>/` if you configure TLS). The `OE_USER` / `OE_PASS` you put in `.env` should let you log in (the compose example sets those values so it can auto-provision an admin account). If the UI prompts you for DB settings, use:

- Host: `mysql` (service name)
- DB user / password: the values from your `.env` (e.g. `openemr` / `OpenEmrDBpass!`)
- Root password: the `MYSQL_ROOT_PASSWORD` value

---

7) Test the DB connection from the VM

```
# open a shell in the MariaDB container
docker compose exec mysql mysql -u root -p"${MYSQL_ROOT_PASSWORD}"
-e "SHOW DATABASES;"
```

---

8) Backups & restore (basic)

Backup DB (dump):

```
docker compose exec mysql sh -c 'exec mysqldump --all-databases -u root -
p"$MYSQL_ROOT_PASSWORD"' > ~/openemr-db-backup.sql
```

Restore (example):

```
docker compose exec -T mysql sh -c 'mysql -u root -p"$MYSQL_ROOT_PASSWORD"' < ~/openemr-db-backup.sql
```

Also snapshot the `databasevolume` if you need block-level recovery.

---

9) Security & production notes (must-read)

- **Never** expose port **3306** to the internet. Keep it internal to the Docker network (the example above does not publish 3306). (repo.or.cz)
- Use HTTPS in production — add a reverse proxy (nginx/Traefik) + Let's Encrypt, or use a load balancer with TLS.
- For production PHI, follow legal/compliance requirements (HIPAA, local laws) and consider Azure managed DB services (Azure Database for MariaDB or MySQL) for better backups and HA.
- Rotate passwords & use Azure Key Vault for secrets.

---

10) Troubleshooting tips

- If OpenEMR says "cannot connect to DB", check `docker compose logs mysql` and that `mysql` service is `healthy` (`docker compose ps` shows health). The compose above uses a healthcheck so OpenEMR waits until MariaDB is ready. (repo.or.cz)
- If containers fail to start, run `docker compose up` (no `d`) to see errors.
- If you changed `.env`, re-create stack: `docker compose down -v` then `docker compose up -d` (warning: `v` will remove volumes — **do not** use `v` in prod unless you mean to delete DB).

Quick checklist (final)

- ☐ Create Azure VM (Ubuntu LTS). (Microsoft Learn)
- ☐ Open inbound 80 & 443 (and SSH) only. (Microsoft Learn)

☐ SSH into VM, install Docker & Compose (official instructions). (Docker Documentation)

☐ Create `.env` with strong passwords.

☐ Paste `docker-compose.yml` and `docker compose up -d`. (repo.or.cz)

☐ Verify containers, visit app URL, secure with TLS and backups.

If you want, I can:

- Produce a ready-to-run shell script that creates the project folder, writes `.env` and `docker-compose.yml` with placeholders (so you only need to fill secrets), **or**
- Produce a separate `nginx` + `certbot` compose snippet to enable automatic HTTPS, **or**
- Walk you through migrating an existing OpenEMR backup into this container stack.

Which of those would you like me to generate next?