AI-03101 Docker Volumes Explained

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	@October 24, 2025 10:25 AM

Here's a **one-page** `README.md` summarizing all key information from this conversation in a clear, reusable format:

```
# 📌 Docker OpenEMR + MariaDB: Volumes, Persistence & Backup Guide

This document summarizes how Docker volumes work in your OpenEMR setu
p, how data persistence is handled, and how to safely back up or recycle data
bases when testing applications.

---

## ✅ 1. How Docker Volumes Work

In your `docker-compose.yml`, you used **named volumes**:

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

These named volumes are **not saved inside your project folder** (e.g. `/home/dockerlab/openemr`).

Instead, Docker stores them here on the host:
```

```
/var/lib/docker/volumes/<volume_name>/_data/
```

✔️ For data to persist:

You must **define** the volume and also **attach it** inside each service:

```
services:
  mysql:
    volumes:
      - databasevolume:/var/lib/mysql

  openemr:
    volumes:
      - logvolume01:/var/log
      - sitevolume:/var/www/localhost/htdocs/openemr/sites
```

✅ 2. Named Volumes vs Bind Mounts

Method	Stored Where?	Good For
Named Volume	`/var/lib/docker/volumes/.../_data`	Safe, automatic persistence
Bind Mount	`./data/` inside `/home/dockerlab/openemr`	Easy access from host
No Volume	Inside container → lost when container is removed	Never use for databases

Example of using bind mounts (if you want data inside your project folder):

```
volumes:
  - ./data/mysql:/var/lib/mysql
  - ./data/sites:/var/www/localhost/htdocs/openemr/sites
```

✅ 3. Backing Up & Moving Database Volumes

You **can copy named volume folders** for backup or transfer to another VM — but only if containers are stopped first.

📦 Backup a MariaDB volume:

```
docker compose down
sudo tar -czvf mariadb_backup.tar.gz /var/lib/docker/volumes/databasevolume/_data
```

📂 Restore on another system:

```
docker compose down
sudo tar -xzvf mariadb_backup.tar.gz -C /var/lib/docker/volumes/databasevolume/_data
docker compose up -d
```

✅ 4. Safer Database Backup Methods

Method	Command Example
MariaDB Dump	`docker exec mysql mysqldump -u root -proot --all-databases > backup.sql`
Restore	`cat backup.sql
Postgres	`docker exec postgres pg_dumpall -U postgres > backup.sql`

This method is safer and portable across systems.

✅ 5. Recycling / Resetting Database for Testing

If you test Docker apps often and want a fresh database quickly:

- **Snapshot method (tar volume)**
- **SQL dump seed with** `/docker-entrypoint-initdb.d/`
- **Automation script:**

```
#!/bin/bash
docker compose stop

echo "Backing up..."
docker exec mysql mysqldump -u root -proot --all-databases > backup.sql

echo "Restoring..."
cat backup.sql | docker exec -i mysql mysql -u root -proot

docker compose start
echo "Database recycled."
```

✅ 6. Key Takeaways

✔️ Named volumes = persistent, stored in `/var/lib/docker/volumes/...`

✔️ Must be **defined + referenced** in services to work

✔️ You can **back up & transfer** volumes safely if containers are stopped

✔️ Use **SQL dumps or tar archives** for database snapshots

✔️ Great for developers who reset Docker often and want repeatable DB states

---

Let me know if you want this turned into a GitHub-ready folder with scripts and compose files 📁🚀

```
---

Would you like me to:
- ✅ Include backup/restore scripts in the README?
- ✅ Add docker commands section?
- ✅ Provide downloadable `.zip` with folders set up?

Just let me know!
```

Docker OpenEMR .yml Explained

```
# ---------------------------
# Docker Compose for OpenEMR + MariaDB
# ---------------------------
# Credentials hint:
# Default credentials to log in to OpenEMR:
#   Username = admin (OE_USER)
#   Password = pass  (OE_PASS)
# Database connection is handled via environment variables below.

services:
  mysql:                          # Defines the MariaDB (MySQL-compatible) datab
ase service
    restart: always                 # Automatically restart the container if it stops
    image: mariadb:11.8               # Uses MariaDB version 11.8 as database eng
ine
    command: ['mariadbd','--character-set-server=utf8mb4']
                                  # Overrides default command to set UTF-8 encoding
(full Unicode support)
    volumes:
    - databasevolume:/var/lib/mysql      # Persists database data on local volum
e (so data isn't lost on container restart)
    environment:
      MYSQL_ROOT_PASSWORD: root         # Sets the root (admin) password fo
r MariaDB
    healthcheck:                    # Health check to ensure MariaDB is working be
fore OpenEMR connects
      test:
      - CMD
      - /usr/local/bin/healthcheck.sh    # Runs this script to test if database is re
ady
      - --su-mysql                  # Checks as MySQL user
```

```
      - --connect                    # Ensures the DB is accepting connections
      - --innodb_initialized         # Verifies InnoDB storage engine is initialized
      start_period: 1m               # Wait up to 1 minute before first health check
attempt
      start_interval: 10s            # Time between checks during the start period
      interval: 1m                   # Regular health check every 1 minute
      timeout: 5s                    # Each check must respond within 5 seconds
      retries: 3                     # Mark container as unhealthy after 3 failures

  openemr:                           # OpenEMR service (web-based medical record
s system)
    restart: always                  # Automatically restart on crash or reboot
    image: openemr/openemr:7.0.3     # Uses OpenEMR version 7.0.3
    ports:
    - 80:80                          # Maps container port 80 to host port 80 (HTTP)
    - 443:443                        # Maps container port 443 to host port 443 (HT
TPS)
    volumes:
    - logvolume01:/var/log           # Persists log files outside container
    - sitevolume:/var/www/localhost/htdocs/openemr/sites
                                     # Stores site-specific OpenEMR settings and configur
ations
    environment:                     # Environment variables used for initial setup &
DB connection
      MYSQL_HOST: mysql              # Name of the database service (Docker i
nternal hostname)
      MYSQL_ROOT_PASS: root          # Root password of MariaDB (must matc
h mysql service)
      MYSQL_USER: openemr            # Database username (optional; defaults
to 'openemr' if omitted)
      MYSQL_PASS: openemr            # Database user password
      OE_USER: admin                 # OpenEMR login username (default: admin)
      OE_PASS: pass                  # OpenEMR login password (default: pass)
    depends_on:                      # Ensure database is healthy before starting Op
enEMR
      mysql:
```

```
      condition: service_healthy      # Only starts after the MySQL container pa
sses its health check
    healthcheck:                      # Health check to ensure OpenEMR web server
is working
      test:
      - CMD
      - /usr/bin/curl                 # Uses curl to check if the web page loads succ
essfully
      - --fail                      # Makes curl return a non-zero exit code on errors
      - --insecure                    # Ignores SSL certificate issues (self-signed cert
s)
      - --location                  # Follows redirect responses
      - --show-error                  # Displays errors if any
      - --silent                    # Runs quietly (no extra output)
      - https://localhost/            # Test URL to ensure OpenEMR is serving HTT
PS
      start_period: 3m                # Give up to 3 minutes for OpenEMR to fully ini
tialize
      start_interval: 10s           # Time between checks during startup
      interval: 1m                  # Regular health check every 1 minute
      timeout: 5s                   # Must respond within 5 seconds
      retries: 3                    # Mark container unhealthy after 3 failed checks

volumes:                            # Define named volumes to persist data
  logvolume01: {}                   # Stores logs from OpenEMR
  sitevolume: {}                    # Stores OpenEMR site configurations and uplo
aded data
  databasevolume: {}                # Stores MariaDB database files
```

To ensure **all data—including MariaDB database files and volumes—reside inside** `/home/dockerlab/zammad`, you'll want to explicitly define **volume paths** in your `docker-compose.yml` file that point to subdirectories within that folder. Here's how to structure it:

🛠️ Step-by-Step Setup

1. **Create subfolders for persistent data**

Inside `/home/dockerlab/zammad`, create folders for each service's data:

```
mkdir -p /home/dockerlab/zammad/data/mariadb
mkdir -p /home/dockerlab/zammad/data/zammad
```

2. **Update your `docker-compose.yml`**

Here's a simplified example showing how to bind volumes to those folders:

```
version: '3.7'

services:
  mariadb:
    image: mariadb:10.5
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: zammad
      MYSQL_USER: zammad
      MYSQL_PASSWORD: zammadpass
    volumes:
      - ./data/mariadb:/var/lib/mysql
    restart: always

  zammad:
    image: zammad/zammad
    depends_on:
      - mariadb
    ports:
      - "8080:80"
    volumes:
      - ./data/zammad:/opt/zammad
    restart: always
```

```
volumes:
  # Optional if you want named volumes, but not needed for bind mounts
```

> ✅ Note: ./data/mariadb and ./data/zammad are relative paths, so they resolve to /home/dockerlab/zammad/data/... when you run docker-compose from that directory.

3. **Run from the correct directory**

Make sure you're inside `/home/dockerlab/zammad` when you run:

```
cd /home/dockerlab/zammad
docker-compose up -d
```

📦 Why This Works

- **Bind mounts** (`./data/...`) ensure data is stored directly in your project folder.
- This setup makes your DockerLab fully self-contained and easy to zip, back up, or migrate.

Would you like me to help you generate a zipped README or restore script for this setup?

Here's a clear example of how to use **named volumes** in a `docker-compose.yml` file for your Zammad + MariaDB setup. Named volumes are managed by Docker and stored in Docker's default volume directory (usually `/var/lib/docker/volumes`), but you can still control their names and reuse them across services.

📥 Example: Using Named Volumes

```
version: '3.8'

services:
  mariadb:
    image: mariadb:10.5
    environment:
```

```
    MYSQL_ROOT_PASSWORD: rootpass
    MYSQL_DATABASE: zammad
    MYSQL_USER: zammad
    MYSQL_PASSWORD: zammadpass
  volumes:
    - mariadb_data:/var/lib/mysql
  restart: always

 zammad:
  image: zammad/zammad
  depends_on:
    - mariadb
  ports:
    - "8080:80"
  volumes:
    - zammad_data:/opt/zammad
  restart: always

volumes:
 mariadb_data:
 zammad_data:
```

📌 Key Points

- `mariadb_data` and `zammad_data` are **named volumes** declared at the bottom under `volumes:`.
- These volumes are **not tied to a specific path** on your host unless you explicitly map them.
- Docker stores them in `/var/lib/docker/volumes/`, but you can inspect them with:

```
docker volume ls
docker volume inspect mariadb_data
```

🧭 When to Use Named Volumes

Use named volumes when:

- You want Docker to manage storage paths.
- You plan to reuse the volume across multiple containers or compose files.
- You want easier portability without worrying about host paths.

If you want to **combine the clarity of named volumes with the control of bind mounts**, I can show you how to alias named volumes to specific host paths too. Just say the word.

Great! Here's how to **alias named volumes to specific host paths**—giving you the **clarity of named volumes** and the **control of bind mounts** inside `/home/dockerlab/zammad`.

🔄 Hybrid Approach: Named Volumes with Explicit Host Paths

You can define named volumes in the `volumes:` section and map them to host directories using the `driver_opts` key. Here's how:

```
version: '3.8'

services:
  mariadb:
    image: mariadb:10.5
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: zammad
      MYSQL_USER: zammad
      MYSQL_PASSWORD: zammadpass
    volumes:
      - mariadb_data:/var/lib/mysql
    restart: always

  zammad:
    image: zammad/zammad
    depends_on:
```

```
      - mariadb
    ports:
      - "8080:80"
    volumes:
      - zammad_data:/opt/zammad
    restart: always

volumes:
  mariadb_data:
    driver: local
    driver_opts:
      type: none
      device: /home/dockerlab/zammad/data/mariadb
      o: bind

  zammad_data:
    driver: local
    driver_opts:
      type: none
      device: /home/dockerlab/zammad/data/zammad
      o: bind
```

🧠 Why Use This?

- **Named volume benefits**: Easy to reference, portable across services.
- **Bind mount control**: You know exactly where the data lives on disk (`/home/dockerlab/zammad/data/...`).
- **Zippable & restorable**: Perfect for your legacy gifting and DockerLab reproducibility goals.

Would you like a `README.md` template or a `restore.sh` script to go with this setup? I can tailor it for Brent or your zipped archive.

Perfect, Rigel. Here's a zipped legacy-grade starter kit for your DockerLab setup using **named volumes aliased to `/home/dockerlab/zammad`**. This includes a `README.md` and a `restore.sh` script tailored for gifting, mentoring, and reproducibility.

📘 README.md (Zammad + MariaDB with Named Volume Bind Mounts)

```
# Zammad + MariaDB Docker Setup (Legacy-Grade Bind Mounts)

This setup ensures all persistent data is stored inside `/home/dockerlab/zammad`, making it easy to back up, zip, and gift.

## 📁 Directory Structure
```

```
/home/dockerlab/zammad/
├── docker-compose.yml
└── data/
    ├── mariadb/
    └── zammad/
```

```
## 🚀 Quick Start

1. Ensure the following directories exist:

```bash
mkdir -p /home/dockerlab/zammad/data/mariadb
mkdir -p /home/dockerlab/zammad/data/zammad
```

1. Run the stack:

```
cd /home/dockerlab/zammad
docker-compose up -d
```

🧱 Volume Mapping

Named volumes are aliased to host paths:
```

```
volumes:
  mariadb_data:
    driver: local
    driver_opts:
      type: none
      device: /home/dockerlab/zammad/data/mariadb
      o: bind

  zammad_data:
    driver: local
    driver_opts:
      type: none
      device: /home/dockerlab/zammad/data/zammad
      o: bind
```

🔁 Restore Instructions

Use `restore.sh` to bring the stack back online after a backup or migration.

```
---

### 🔄 `restore.sh` (Simple Restore Script)

```bash
#!/bin/bash

echo "🛠️ Restoring Zammad + MariaDB stack..."

cd /home/dockerlab/zammad || {
  echo "❌ Directory /home/dockerlab/zammad not found."
  exit 1
}

echo "📦 Ensuring data directories exist..."
```
```

```
mkdir -p data/mariadb data/zammad

echo "🚀 Starting Docker stack..."
docker-compose up -d

echo "✅ Restore complete. Access Zammad at http://localhost:8080"
```

> 🧠 Tip: You can zip /home/dockerlab/zammad including this README and script for Brent or mentees. It's reproducible, debug-friendly, and legacy-grade.

Would you like to add a `backup.sh` script next, or embed this into a zipped gifting protocol with version notes and changelog?