AI-04023 OpenEMR Backup and Restore

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	July 12, 2026 3:40 PM

What is the enterprise BKM for OpenEMR Backup and Restore. In OpenEMR, I created a backup, where is the backup saved? What's in it? How to restore the backup? Provide a restore step by step on OpenEMR running on Docker, I have phpmyadmin running. Can i restore using phpmyadmin? The Docker OpenEMR is in GCP.

In a production environment—especially on a cloud infrastructure like Google Cloud Platform (GCP)—relying on the built-in browser-based backup utility is highly discouraged. For an enterprise-grade setup, your backup and restore architecture must decouple data handling from the user interface.

1. Enterprise BKM (Best Known Method)

The enterprise standard for OpenEMR requires capturing two distinct components simultaneously to prevent data skew: the relational database and the physical persistent storage volume.

- **Database:** Use native utility tools like `mysqldump` or `mariadb-dump` executed directly on the host or inside the container. Do not run heavy SQL dumps via the web server worker processes.
- **Persistent Storage (The Web Directory):** Take cold snapshots or use block-level file synchronization tools like `rsync` or `tar` on the `sites/` directory. This houses critical patient documents, uploaded PDFs, scanned charts, and custom configuration fields (`sqlconf.php`).
- **Automation & Cloud Native Approach (GCP):**

- **Stateful Volumes:** Ensure your Docker containers mount local host directories or persistent disks (`gcePersistentDisk`) for `/var/www/localhost/htdocs/openemr/sites` and the database storage directory (`/var/lib/mysql`).
- **Snapshots:** The ultimate BKM in GCP is automating standard disk snapshots of your persistent disks using GCP Cloud Storage lifecycle policies, alongside a secondary automated shell cron script that outputs to a secure Google Cloud Storage (GCS) bucket.

2. Built-in Browser Backup: Where & What

If you executed a backup via the OpenEMR UI (`Administration -> System -> Backup`), here is how it behaves:

- **Where is it saved?** It does not stay on the server. The script compiles the archives in a temporary location and immediately pushes a download payload (`emr_backup.tar`) directly to your client web browser.
- **What is in it?** If you extract `emr_backup.tar`, you will find two nested compressed archives:
  1. `openemr.sql.gz`: A compressed SQL dump of the entire MariaDB/MySQL database instance.
  2. `openemr.tar.gz`: A compressed archive of the web application directory containing configuration parameters, certificates, and the all-important `sites/default/documents/` folder where patient media files reside.

3. Can I restore using phpMyAdmin?

Yes, partially. You can use phpMyAdmin to restore the relational database component (`openemr.sql.gz`), but **you cannot restore the entire application with it.**

> ⚠️ **Critical Risk:** phpMyAdmin knows nothing about your persistent files. If you only restore the database, your tables will point to patient documents, signed consent forms, and lab results that do not exist on the file system, breaking data integrity. Additionally, phpMyAdmin often hits PHP execution time limits (`max_execution_time`) or upload size limits (`upload_max_filesize`) when dealing with large production database files.

4. Step-by-Step Production Restore on Docker (GCP Host)

This procedure completely bypasses the fragile web interface and uses the native system architecture to cleanly restore both your database dump and application files.

Prerequisites

- SSH access to the GCP Compute Engine VM instance hosting the containers.
- The `emr_backup.tar` uploaded to the host VM (e.g., inside your home directory `/home/user/`).
- Your active `docker-compose.yml` environment structure.

Step 1: Extract the Backup Components on the Host

Unpack the master archive to separate the database structural dump from the application assets:

Bash

```
# Extract the master tarball
tar -xf /home/user/emr_backup.tar -C /home/user/

# This yields openemr.sql.gz and openemr.tar.gz
# Decompress the database file for injection
gunzip /home/user/openemr.sql.gz
```

Step 2: Identify and Isolate Container States

Determine the active container IDs or names assigned by the Docker engine.

Bash

```
docker ps
```

(Look for your OpenEMR app container name, e.g., `openemr_openemr_1`, and your database container name, e.g., `openemr_db_1`.)

Step 3: Restore the Web Directory Assets

Copy the web directory archive directly into the running OpenEMR application container space and extract it into the correct web root hierarchy.

Bash

```
# Copy the archive into the container's temporary space
docker cp /home/user/openemr.tar.gz openemr_openemr_1:/tmp/

# Execute container shell commands to unpack into the web root
docker exec -it openemr_openemr_1 sh -c "
  cd /var/www/localhost/htdocs/openemr && \
  tar -xzf /tmp/openemr.tar.gz --strip-components=1 && \
  chown -R apache:apache sites/ && \
  rm /tmp/openemr.tar.gz
"
```

(Note: Adjust `apache:apache` to `www-data:www-data` if your specific underlying base Docker image is Debian/Ubuntu-based instead of Alpine-based).

Step 4: Drop and Recreate the Target Database

To prevent unique key conflicts or dirty database updates, drop the current data schema and initialize an empty target base.

Bash

```
# Access the MariaDB container command line interface to reset schema
docker exec -it openemr_db_1 mysql -u root -p -e "DROP DATABASE IF EXISTS openemr; CREATE
DATABASE openemr CHARACTER SET utf8 COLLATE utf8_general_ci;"
```

Step 5: Inject the Clean SQL Stream

Pipe the decompressed native SQL dump straight into the freshly initialized container database workspace.

Bash

```
docker exec -i openemr_db_1 mysql -u root -p openemr < /home/user/openemr.sql
```

Step 6: Flush Memory Caches and Restart Stack

Force the environment to re-read system states and re-establish clean connection hooks.

Bash

```
docker-compose restart
```

5. Alternative: Restoring via phpMyAdmin

If you must use phpMyAdmin to handle the database layer specifically, execute these steps:

1.Extract the Database Dump Locally:

Desktop Workspace.

Unpack `emr_backup.tar` on your machine using a utility like 7-Zip or terminal commands. Isolate the `openemr.sql.gz` file.

2.Prepare Target Schema via phpMyAdmin:

Web Console Interface.

Log into phpMyAdmin, select the target `openemr` database schema from the left-hand navigation panel, navigate to the **Structure** tab, select all existing tables, and select **Drop** to clear out conflicting structures.

3.Execute Import Stream:

Execution Configuration.

Click on the **Import** tab. Choose the isolated `openemr.sql.gz` file. If the file is large, change the partial import settings to prevent timeout drops before execution completes. Click **Go**.

4.Sync Filesystem Volumes:

Manual Host Maintenance.

You **must** still manually SSH into your GCP host and extract the contents of `openemr.tar.gz` into the designated persistent volume path mapped to your OpenEMR application container's `sites/` directory.