Installation Guide

Installation and deployment guide for Open Clinical History.

- How to Import SNOMED CT
- Setting Up a New Open Clinical History Server

How to Import SNOMED CT

Open Clinical History uses a local **SNOMED CT RF2 Snapshot release** to provide clinical terminology, terminology search, anatomical hierarchy information and clinical concept-to-body-layer mappings.

The SNOMED distribution is large and the initial import can take some time.

Once imported, Open Clinical History builds optimised local lookup tables so the application does not need to process the raw RF2 files during normal operation.

---

1. Obtain a SNOMED CT RF2 Release

Download an authorised SNOMED CT RF2 release from your SNOMED distribution provider.

Open Clinical History expects an **RF2 Snapshot** release.

A typical release contains directories similar to:

```
Content
Language
Map
Metadata
Refset
Terminology
```

For example:

```
root@server:/var/www/private/demo/snomed# ls


Content  Language  Map  Metadata  Refset  Terminology
```

The exact directory structure can vary slightly between SNOMED editions and releases.

Open Clinical History searches the release directories for the required RF2 files rather than depending on exact release filenames.

---

2. Store the Release Outside the Web Root

SNOMED CT source files should **not be publicly accessible through the web server**.

Store the extracted RF2 release in a private server directory outside the application's public `DocumentRoot`.

For example:

```
/var/www/private/demo/snomed
```

Do **not** place the RF2 distribution somewhere such as:

```
/var/www/html/snomed
```

or another directory that Apache or Nginx can serve directly.

The SNOMED files are licensed content and should not be exposed for direct download.

File permissions

The files must not be web-accessible, but the Open Clinical History PHP process and background import worker still need permission to read them.

For example, your final structure may look similar to:

```
/var/www/
├── private/
│   └── demo/
│       └── snomed/
│           ├── Content/
│           ├── Language/
```

```
│           ├── Map/
│           ├── Metadata/
│           ├── Refset/
│           └── Terminology/
│
└── openclinicalhistory/
    └── public application files
```

The important distinction is:

> The SNOMED directory must be readable by the application, but must not be reachable as a public web URL.

---

3. Configure the RF2 Release Path

Open:

Admin → Configuration

Locate:

SNOMED CT RF2 → RF2 release path

Enter the absolute server path containing the extracted SNOMED release.

For example:

```
/var/www/private/demo/snomed
```

Save the configuration.

Open Clinical History accepts either:

- the release root; or
- a compatible RF2 `Snapshot` directory.

The importer searches below the configured directory for the required RF2 files.

---

4. Open the SNOMED Import

From the administration interface, open:

SNOMED Import

The page inspects the configured release before starting an import.

At the top of the page you should see information similar to:

```
Release: AU1000036_20260731
/var/www/private/demo/snomed


Source: app_config
```

The screen also displays detected information about the currently staged SNOMED database, including:

- active concepts
- relationships
- anatomical/site relationships
- presence of the International core

---

5. Check the RF2 File Inventory

Before starting the pipeline, check the **RF2 file inventory** at the bottom of the page.

Open Clinical History currently requires four RF2 Snapshot datasets.

Dataset	Purpose	Required

Concepts	SNOMED concept identifiers and status	Yes
Descriptions	Clinical names, terms and synonyms	Yes
Relationships	SNOMED hierarchy and clinical relationships	Yes
Language refset	Preferred terminology for the selected language edition	Yes

Each required file should show:

```
ready
```

For example:

```
Import concepts          ready
Import descriptions      ready
Import relationships     ready
Import language refset   ready
```

If a required file shows:

```
missing
```

do not force the import. Check the RF2 package and configured release path first.

Are SNOMED Relationships Required?

Yes, in the current Open Clinical History implementation.

The relationship file is not simply retained as unused SNOMED reference data.

Open Clinical History uses SNOMED relationships for several important functions.

Anatomical hierarchy

The application follows SNOMED **IS-A** relationships to build its anatomical hierarchy.

Conceptually:

```
Body structure
      |
      v
Parent anatomical structures
      |
      v
More specific structures
      |
      v
Specific anatomical site
```

This allows a specific SNOMED body structure to inherit the appropriate Open Clinical History anatomical image layer.

Finding-site relationships

Clinical concepts can contain relationships to anatomical structures.

For example, conceptually:

```
Clinical disorder
       |
       | finding site
       v
Body structure
       |
       v
Open Clinical History image layer
```

These relationships are used when building the concept-to-image mapping.

Without the relationship dataset, Open Clinical History cannot currently build these derived anatomical mappings correctly.

Release verification

The importer also verifies that the RF2 release contains:

- a substantial relationship dataset
- SNOMED `IS-A` relationships
- anatomical/site relationships

A release without them will fail the current verification stage.

---

6. Start the Pipeline

Once all required RF2 files show as ready, click:

Start or resume pipeline

The import runs as a background process.

You can leave the page open to watch its progress.

The pipeline consists of the following stages.

Step	Stage	Purpose
1	Import concepts	Load the RF2 Concept Snapshot
2	Import descriptions	Load SNOMED names and synonyms
3	Import relationships	Load hierarchy and clinical relationships
4	Import language refset	Load preferred-language terminology
5	Verify release	Confirm that the imported release is usable
6	Validate body image mappings	Validate configured anatomical SNOMED seeds
7	Build anatomical hierarchy	Build body-structure ancestry
8	Build concept-to-image mappings	Connect clinical concepts to anatomical layers
9	Build health-history concept lookup	Build optimised concept lookup data
10	Build health-history term lookup	Build optimised terminology search data
11	Validate health-history catalogue	Check the resulting clinical catalogue
12	Complete	Pipeline finished

7. Be Patient

SNOMED CT is **large**.

A release can contain millions of rows.

A typical Australian release may contain approximately:

```
Concepts             hundreds of thousands
Descriptions         millions
Relationships        millions
Language refset      millions
```

The raw RF2 files themselves may also be hundreds of megabytes in size.

The first import therefore takes time.

Some later stages must also analyse and build indexes over millions of imported records, so apparent pauses during the process are not necessarily failures.

Avoid restarting the server or database while the import is running.

---

8. Monitoring Progress

While an import is running, the SNOMED Import page displays:

- current pipeline stage
- worker process
- elapsed time
- rows processed
- worker heartbeat
- worker output
- error messages

The page updates as the background worker progresses.

For large RF2 files the row count can increase substantially before a stage completes.

9. Safe Restart and Resume

The SNOMED pipeline is designed to be resumable.

If an import is interrupted, return to:

SNOMED Import

and select:

Start or resume pipeline

The importer checks:

- the persistent pipeline state
- the selected RF2 release
- the source-file information
- existing database tables
- current row counts

Completed and still-valid stages are skipped.

A failed or stale stage is rebuilt, followed by any stages that depend on it.

This avoids unnecessarily re-importing millions of SNOMED rows after an interruption.

Rebuild Every Stage

The SNOMED Import screen also contains:

Rebuild every stage, including completed tables

Selecting this forces the entire pipeline to run again.

This includes re-importing the large RF2 source tables.

Do not select this for a normal restart.

Use a full rebuild when, for example:

- installing a new SNOMED CT release
- deliberately replacing all staged terminology
- troubleshooting potentially inconsistent source tables
- testing changes to the SNOMED import process

For an interrupted import, normally use **Start or resume pipeline** without selecting the rebuild option.

---

10. Successful Completion

The import is complete when all pipeline stages show:

```
complete
```

and the final stage reports:

```
Pipeline complete
```

At this point Open Clinical History has both:

1. the imported SNOMED CT RF2 source tables; and
2. the derived lookup and anatomical mapping tables used by the application.

Normal patient-history processing can then use the local SNOMED terminology database.

Updating to a New SNOMED Release

When a newer SNOMED release becomes available:

1. Download the new authorised RF2 Snapshot release.
2. Extract it into a new private directory.
3. Update **Admin → Configuration → SNOMED CT RF2 → RF2 release path**.

4. Open **SNOMED Import**.
5. Confirm the RF2 inventory identifies the new files.
6. Run the pipeline.

Keeping separate release directories is preferable to overwriting the previous release immediately.

For example:

```
/var/www/private/snomed/
├── AU1000036_20260731/
└── AU1000036_20270131/
```

This makes rollback and troubleshooting considerably easier.

Troubleshooting

No RF2 Snapshot release found

Check:

- the configured RF2 path
- that the release has been extracted
- that the PHP/worker user can read the directory
- that the release contains RF2 Snapshot files

The path must be a filesystem path, not a URL.

RF2 file shows `missing`

Check the file inventory.

The current importer expects files matching the standard Snapshot families:

```
sct2_Concept_Snapshot...
sct2_Description_Snapshot...
```

```
sct2_Relationship_Snapshot...
der2_cRefset_LanguageSnapshot...
```

The release-specific identifiers and dates in the filenames may vary.

Open Clinical History discovers them by their RF2 filename prefixes.

Existing table rows are shown but the RF2 file is missing

The **Table rows** column reports what is currently in the Open Clinical History database.

It does not prove that the corresponding source file exists in the currently selected release.

For example:

```
Import relationships

File:       missing
Status:     missing
Table rows: 5,061,383
```

means that relationship records exist in the database, but the selected RF2 directory does not currently contain a relationship Snapshot file recognised by the importer.

The existing rows may have come from an earlier release.

Do not treat this as a valid new release until the source inventory also reports the relationship file as:

```
ready
```

Important Security Note

SNOMED CT distributions may be subject to licensing and redistribution conditions.

Do not expose the raw RF2 files through the Open Clinical History website or another publicly accessible directory.

Store them in a protected server-side location and provide access only to the application and authorised administrators.

Setting Up a New Open Clinical History Server

This guide describes how to deploy a new Open Clinical History installation.

Open Clinical History includes a temporary first-run setup assistant:

```
setup_delete_after_setup.php
```

The setup assistant is deliberately self-contained so that it can run **before `config.php` exists and before the application database has been configured**.

It guides the administrator through:

```
Server preflight
      |
      v
Private config.php
      |
      v
Database
      |
      v
Application settings
      |
      v
SNOMED CT
      |
      v
Image/SNOMED database
      |
      v
Queue workers
      |
      v
First patient
```

```
      |
      v
Delete setup installer
```

> **Important:** `setup_delete_after_setup.php` must be deleted from the web server after installation is complete.

Requirements

A new installation requires:

- PHP 8.3 or later
- PHP CLI
- MariaDB or MySQL
- Apache, Nginx or another suitable web server
- cURL support
- PDO / PDO MySQL
- mbstring
- JSON support
- `proc_open()`
- a private configuration directory
- a writable runtime directory
- a private SNOMED CT storage directory
- access to the Open Clinical History source repository

The existing installation design separates the private configuration, runtime and SNOMED files from the public application files.

HTTPS is strongly recommended and should be considered mandatory for an installation that will contain real patient information.

Recommended Directory Structure

Sensitive files should remain outside the web-accessible application directory.

For example:

```
/var/www/
├── private/
│   └── demo/
│       ├── config.php
│       ├── config_template.php
│       ├── health_history.sql
│       ├── run/
│       └── snomed/
│
└── openclinicalhistory/
    └── application files
```

For example:

```
root@server:/var/www/private/demo# ls

config.php
config_template.php
health_history.sql
run
snomed
```

The application web directory is public.

The following are **not** public:

```
config.php
database schema/backups
runtime state
SNOMED CT RF2 files
```

The `/var/www/private` directory must never be configured as the Apache or Nginx document root.

---

1. Obtain the Open Clinical History Source

The Open Clinical History source repository is currently private.

Repository:

```
https://github.com/bengoudieparkoch/openclinicalhistory
```

To request access, visit:

```
https://www.openclinicalhistory.org/access.php
```

and complete the access request form.

The existing deployment documentation also identifies repository access as an approved-user process rather than a public source download.

After access has been granted, clone the repository into the web application directory.

For example:

```
cd /var/www


git clone https://github.com/bengoudieparkoch/openclinicalhistory.git
```

For later application updates:

```
cd /var/www/openclinicalhistory


git pull
```

Do not store GitHub credentials in the Open Clinical History source code.

---

2. Configure the Web Server

Create an Apache VirtualHost or equivalent Nginx configuration for the application.

For example:

```
https://clinical.example.org/
```

The web server's document root should point directly to the Open Clinical History application directory.

For example:

```
DocumentRoot /var/www/openclinicalhistory
```

Do **not** use:

```
DocumentRoot /var/www
```

because that could make `/var/www/private` reachable through the web server.

The original installation guide explicitly separates the application directory from `/var/www/private` for this reason.

---

3. Configure HTTPS

Configure SSL/TLS before using Open Clinical History with real clinical information.

Open Clinical History may contain:

- patient identifiers
- dates of birth
- clinical records
- complete medical histories
- API bearer tokens

These should not be transmitted over plain HTTP.

For example:

```
https://clinical.example.org/
```

The Patient Record API also defaults to requiring HTTPS.

For Apache deployments, Let's Encrypt and Certbot can be used where appropriate.

---

4. Open the First-Run Setup Assistant

Once the application files are available through the web server, open:

```
/setup_delete_after_setup.php
```

For example:

```
https://clinical.example.org/setup_delete_after_setup.php
```

If Open Clinical History is installed under a subdirectory such as `/demo`, use:

```
https://clinical.example.org/demo/setup_delete_after_setup.php
```

The setup assistant contains the installation sequence:

```
0 · Preflight
1 · config.php
2 · Database
3 · App settings
4 · SNOMED
5 · Image DB
6 · Workers
7 · Patient import
8 · Finish
```

It is safe for this page to run before the database is working because it deliberately does not depend on the normal application bootstrap.

---

5. Run the Server Preflight

Select:

0 · Preflight

The installer checks the web-server PHP environment.

Checks currently include:

Check	Requirement
PHP version	PHP 8.3+
PDO	Required
PDO MySQL	Required
cURL	Required
mbstring	Required
JSON	Required
`proc_open()`	Required
CLI PHP	Required
Config template	Should be available

The setup assistant searches for a CLI PHP executable in this order:

```
/usr/bin/php8.5
/usr/bin/php8.4
/usr/bin/php8.3
/usr/bin/php
```

A working CLI PHP installation is important because long-running Open Clinical History operations execute outside the browser request.

These include background processing such as:

```
history_job.php
queue_worker.php
SNOMED import workers
```

Additional Preflight Checks

Also verify manually that:

- MariaDB/MySQL is installed and reachable
- HTTPS is working
- the web-server/worker user can read the application
- the web-server/worker user can read the SNOMED directory
- the runtime directory is writable
- the supplied anatomical PNG catalogue exists under `img/`

Do not proceed until the PHP preflight is clean.

---

6. Create the Private Directory

Create the private installation directory.

For example:

```
sudo install -d -o root -g www-data -m 0750 /var/www/private/demo
```

Create the writable runtime directory:

```
sudo install -d -o www-data -g www-data -m 0770 /var/www/private/demo/run
```

Create the SNOMED storage directory if it does not already exist:

```
mkdir -p /var/www/private/demo/snomed
```

The resulting structure should resemble:

```
/var/www/private/demo/
├── config.php
├── config_template.php
├── health_history.sql
├── run/
└── snomed/
```

The existing setup design uses `run` for background process state and `snomed` for the RF2 terminology release.

---

7. Generate `config.php`

Select:

1 · config.php

The setup assistant provides an interactive `config.php` generator.

The form is processed entirely in the browser.

Database credentials entered into this setup form are **not posted back to `setup_delete_after_setup.php`**.

Enter:

- PHP CLI binary
- private runtime directory
- database host
- database port
- database name
- database username
- database password
- database character set
- debug mode

The default runtime directory is:

```
/var/www/private/demo/run
```

The recommended configuration location is:

```
/var/www/private/demo/config.php
```

Example `config.php`

A typical configuration is:

```
<?php

return [

    'php_binary' => '/usr/bin/php8.3',

    'run_dir' => '/var/www/private/demo/run',

    'db' => [

        'host'     => '127.0.0.1',

        'port'     => 3306,

        'database' => 'health_history',

        'username' => 'openclinicalhistory',

        'password' => 'your_password',

        'charset'  => 'utf8mb4',

    ],

    'debug' => false,

];
```

The existing guide uses the same private configuration structure and keeps `config.php` outside version control.

PHP CLI Binary

For example:

```
'php_binary' => '/usr/bin/php8.3',
```

Check it with:

```
/usr/bin/php8.3 -v
```

And verify the important extensions:

```
/usr/bin/php8.3 -m | grep -E 'pdo_mysql|curl|mbstring'
```

The CLI PHP installation can differ from the PHP installation used by Apache or PHP-FPM.

Both must be correctly configured.

Runtime Directory

For example:

```
'run_dir' => '/var/www/private/demo/run',
```

This directory stores runtime information such as:

- job status
- locks
- worker state
- processing state

It must be writable by the web/worker account.

Database Connection

For example:

```
'db' => [

    'host'     => '127.0.0.1',

    'port'     => 3306,

    'database' => 'health_history',
```

```
    'username' => 'openclinicalhistory',

    'password' => 'your_password',

    'charset'  => 'utf8mb4',

],
```

Use:

```
utf8mb4
```

for the database character set.

Clinical records can contain the full Unicode character set.

Database User

Although a `root` database login may be convenient during development, a dedicated application account is recommended.

For example:

```
CREATE USER 'openclinicalhistory'@'localhost'
IDENTIFIED BY 'use-a-strong-password';

GRANT ALL PRIVILEGES
ON health_history.*
TO 'openclinicalhistory'@'localhost';

FLUSH PRIVILEGES;
```

Then use that account in `config.php`.

Debug Mode

Production installations should normally use:

```
'debug' => false,
```

Debug information can expose application or database information and should only be enabled temporarily when troubleshooting.

Save and Protect `config.php`

The setup page allows the generated configuration to be:

- copied
- downloaded as `config.php`

Save it as:

```
/var/www/private/demo/config.php
```

Then protect it.

For example:

```
sudo chown root:www-data /var/www/private/demo/config.php


sudo chmod 640 /var/www/private/demo/config.php
```

The web server needs to read the file.

It should not be writable by the web process.

Do not commit `config.php` to Git. The existing installation guide specifically keeps server credentials and paths outside the repository.

---

8. Create the Database

Select:

2 · Database

Create the application database.

For example:

```
CREATE DATABASE health_history
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

The base database schema is supplied with the Open Clinical History installation.

For example:

```
health_history.sql
```

Import it:

```
mysql -u openclinicalhistory -p health_history \
  < /var/www/private/demo/health_history.sql
```

The existing installation article already requires loading this base schema before the application can operate.

Run All Database Migrations

The base schema is not necessarily the end of database setup.

Run **all migrations supplied with the installed Open Clinical History release, in release order**.

The current setup assistant specifically identifies migrations responsible for features including:

```
app_config
api_token
ingest_queue
```

Known migrations referenced by the current application include:

```
migrations/2026-08-21_config_and_api_tokens.sql
migrations/2026-08-21b_ingest_queue.sql
```

The release package itself should be treated as authoritative.

If additional migrations are supplied, apply those as well.

Do not skip migrations simply because the base application page loads.

Verify the Database

Connect to the database:

```
mysql -u openclinicalhistory -p health_history
```

Then check:

```
SHOW TABLES;
```

In particular, before moving to application configuration, confirm that the required configuration and queue tables exist.

---

9. Configure Open Clinical History

Select:

3 · App settings

or open:

```
/admin_config.php
```

`config.php` contains the infrastructure settings needed to connect to the server and database.

Operational application settings are instead stored in:

```
app_config
```

Configure at least:

- date format
- SNOMED RF2 release path
- LLM enabled
- Gemini API key
- Gemini model
- provider limits
- extraction limits
- grounded SNOMED settings
- API configuration
- ingest queue configuration

The existing server article already places these settings after database setup and before SNOMED processing.

Gemini API Key vs Open Clinical History API Token

These are two different credentials.

Gemini API key

Used for:

```
Open Clinical History
        |
        v
Gemini
```

It authenticates **outbound LLM requests**.

Open Clinical History API token

Used for:

```
External clinical system
        |
        v
Open Clinical History Patient Record API
```

It authenticates **inbound patient-document submissions**.

Do not confuse the two.

API Token

If another system will send patient records through:

```
history_api_import.php
```

create an API token under:

Admin → Configuration

For a normal importing integration, the likely scopes are:

```
import
status
commit
```

Store the token immediately.

The plaintext token is displayed only once.

Only its hash is stored afterwards.

Enable the import API only after HTTPS has been configured.

---

10. Install SNOMED CT

Select:

4 · SNOMED

Open Clinical History does not redistribute SNOMED CT.

The installing organisation must obtain the authorised terminology release under the appropriate SNOMED licensing arrangements.

For an Australian installation, obtain the appropriate **SNOMED CT-AU RF2 Snapshot** release.

Store the extracted files in the protected server directory.

For example:

```
/var/www/private/demo/snomed
```

A release may contain:

```
Full/
Snapshot/
Delta/
```

with directories such as:

```
Terminology/
Refset/
```

The previous server guide already specifies the Snapshot release as the source for normal Open Clinical History imports.

Configure the SNOMED Path

Under:

Admin → Configuration → SNOMED CT RF2

set:

```
/var/www/private/demo/snomed
```

or the root of the specific extracted release.

Run SNOMED Import

Open:

SNOMED Import

or:

```
/snomed_import.php
```

Confirm that the RF2 inventory detects:

```
Concept Snapshot
Description Snapshot
Relationship Snapshot
Language Snapshot
```

Then run:

Start or resume pipeline

The original installation guide also requires all four datasets before proceeding.

SNOMED Import Can Take a Long Time

SNOMED CT is large.

The import involves millions of terminology rows and multiple derived-index stages.

Do not proceed simply because the first RF2 tables have been populated.

Wait until the entire SNOMED pipeline reports completion.

The setup assistant deliberately requires the administrator to confirm:

```
SNOMED import completed and verified
```

before enabling the Image DB step.

---

11. Build the Image / SNOMED Database

Select:

5 · Image DB

or open:

```
/build_image_db.php
```

Do this **only after SNOMED import has completely finished**.

The image build combines:

```
Completed SNOMED terminology
        +
Installed anatomical PNG catalogue
        |
        v
Image / SNOMED Summary Build
        |
        v
Clinical concept ↔ anatomy ↔ image mappings
```

Run:

Clean rebuild image summary database

The existing installation sequence also places this operation immediately after SNOMED import.

Verify the Image Build

Review the resulting coverage, including:

- active PNG layers
- SNOMED seed rows
- anatomy hierarchy rows
- concept/image rows
- mapped clinical concepts
- anatomical layers without verified SNOMED seeds

Do not proceed to production patient ingestion until the result looks sensible.

---

12. Configure Persistent Queue Workers

Select:

6 · Workers

Queued API ingestion requires:

```
queue_worker.php
```

to be running.

Without a queue worker, records can enter:

```
ingest_queue
```

but will remain there waiting for processing.

For testing:

```
cd /var/www/openclinicalhistory


/usr/bin/php8.3 queue_worker.php
```

The existing server documentation also recommends running the queue worker under a persistent service manager rather than relying on a terminal session.

Use a Service Manager

For production, use a host-level process manager such as:

- systemd
- Supervisor
- another appropriate service manager

Workers should restart automatically following:

- process crashes
- application failures
- server reboots

Do not rely on a browser connection to keep a queue worker alive.

Worker Count

The setup assistant recommends starting conservatively and tuning worker concurrency according to:

- CPU capacity
- Gemini concurrency
- configured LLM budgets
- database load
- available memory

More workers are not automatically better.

The LLM provider, database and token budget can become the limiting resources before CPU does.

Monitor Workers

Open:

```
/import_monitor.php
```

Confirm that the expected worker processes are visible before proceeding.

---

13. Import the First Patient

Select:

7 · Patient import

Do not begin with bulk production data.

The recommended first validation is a single known:

```
synthetic
```

or:

```
de-identified
```

patient history.

Open:

```
/history_import.php
```

Upload one plain-text history and allow it to complete.

Validate the Entire Pipeline

For the first patient:

1. Upload the source history.
2. Start extraction.
3. Watch processing through the available monitoring screens.
4. Review the extracted clinical events.
5. Review SNOMED CT classifications.
6. Review anatomical mappings.
7. Check the original source evidence.
8. Review date precision and chronology.

9. Review the audit output.
10. Confirm unresolved terminology appears in the Learning Queue instead of being silently guessed.
11. Complete the import.
12. Open the resulting patient history.

The existing guide already uses manual import as the end-to-end validation of database, worker, Gemini, SNOMED and anatomy configuration.

---

14. Test the Patient Record API

Only after the manual test has succeeded should an external integration be enabled.

Use:

```
/history_api_import.php?action=ping
```

with an Open Clinical History Bearer token.

Then test one non-production record through the API.

Verify:

```
submission
    |
    v
queued
    |
    v
extracting
    |
    v
audit / commit
    |
    v
patient history
```

Do not expose the API token in:

- browser-side JavaScript
- public source code
- Git repositories
- documentation

The original article similarly places API testing after manual patient validation.

---

15. Finish the Installation

Select:

8 · Finish

Before removing the setup assistant, confirm:

- private `config.php` is outside the web root
- `config.php` permissions are restricted
- base database schema is installed
- all database migrations are installed
- Gemini/application settings are configured
- SNOMED CT has completed and verified
- Image/SNOMED database has completed and verified
- queue workers are supervised and running
- a test patient completed successfully end-to-end
- the API has been tested if it will be used
- backups have been configured

---

16. Delete `setup_delete_after_setup.php`

This step is mandatory.

The first-run installer exposes information useful during server configuration and is intentionally not part of the normal running application.

Once installation has been proven:

```
cd /var/www/openclinicalhistory

rm setup_delete_after_setup.php
```

If your application is installed somewhere else, adjust the path accordingly.

For example:

```
cd /var/www/openclinicalhistory/demo

rm setup_delete_after_setup.php
```

Confirm that:

```
https://your-domain.example/setup_delete_after_setup.php
```

now returns:

```
404 Not Found
```

or is otherwise inaccessible.

> **Open Clinical History does not depend on this file after installation.**

Do not leave it available on a production server.

Updating Open Clinical History Later

After the initial installation:

```
cd /var/www/openclinicalhistory

git pull
```

Review the release for:

- database migrations
- changed configuration requirements
- SNOMED rebuild requirements
- Image/SNOMED rebuild requirements
- worker changes

Do not assume that every application update requires a SNOMED or image rebuild.

Follow the release-specific instructions.

If `setup_delete_after_setup.php` is restored by a later Git update, **do not leave it publicly accessible simply because the server is already configured**.

Remove it again unless it is deliberately required for a controlled upgrade procedure.

Recommended File Permissions

Conceptually:

```
Application source
    read by web server

config.php
    root owned
    readable by web-server group
    not writable by web server

run/
    readable/writable by workers

snomed/
    readable by application/workers
    not web accessible
```

For example:

```
sudo chown root:www-data /var/www/private/demo/config.php
sudo chmod 640 /var/www/private/demo/config.php
```

The existing deployment guide also recommends giving the web process write access only where required rather than across the whole application.

Backups

Before using Open Clinical History for important patient information, configure regular backups.

At minimum, protect:

- Open Clinical History database
- private `config.php`
- application-specific customisations
- anatomical image catalogue

The original deployment guidance also recommends retaining the exact SNOMED release where practical because it simplifies recovery and auditing.

Do not rely on Git as a patient-data or database backup.

Security Checklist

A production installation should include:

- HTTPS only
- dedicated database credentials
- restrictive filesystem permissions
- `/var/www/private` inaccessible through HTTP
- no secrets committed to Git
- protected administration pages
- secure API bearer tokens
- current operating-system security updates
- database backups
- appropriate firewall restrictions
- supervised background workers
- deletion of `setup_delete_after_setup.php`

These controls build on the security requirements already identified in the server guide.

New Server Checklist

- [ ] Obtain repository access
- [ ] Clone Open Clinical History
- [ ] Configure Apache/Nginx
- [ ] Configure HTTPS
- [ ] Open `setup_delete_after_setup.php`
- [ ] Pass server preflight
- [ ] Create `/var/www/private/<environment>`
- [ ] Create private `run/`
- [ ] Create private `snomed/`
- [ ] Generate `config.php`
- [ ] Protect `config.php`
- [ ] Create MariaDB/MySQL database
- [ ] Create dedicated database user
- [ ] Import `health_history.sql`
- [ ] Apply all release migrations
- [ ] Confirm `app_config`

- [ ] Confirm `api_token`
- [ ] Confirm `ingest_queue`
- [ ] Configure Admin settings
- [ ] Configure Gemini
- [ ] Install SNOMED CT RF2 Snapshot
- [ ] Run SNOMED Import
- [ ] Verify SNOMED pipeline completion
- [ ] Run Image/SNOMED Summary Build
- [ ] Verify image/SNOMED coverage
- [ ] Configure persistent `queue_worker.php`
- [ ] Confirm workers in Import Monitor
- [ ] Test one synthetic/de-identified patient
- [ ] Review SNOMED, anatomy, evidence and audit
- [ ] Test Patient Record API if required
- [ ] Configure backups
- [ ] **Delete** `setup_delete_after_setup.php`
- [ ] Confirm the setup page is no longer web accessible

Installation Order Summary

The complete first-run sequence is:

```
1. Obtain source access
        |
        v
2. Deploy application
        |
        v
3. Configure domain + HTTPS
        |
        v
4. Open setup_delete_after_setup.php
        |
        v
5. Pass server preflight
        |
        v
6. Create private directories
        |
        v
7. Generate config.php
        |
        v
8. Create database
        |
        v
9. Load base schema + migrations
        |
        v
10. Configure Admin settings + Gemini
        |
        v
11. Install SNOMED RF2
        |
        v
12. Run SNOMED Import
```

```
        |
        v
13. Build Image/SNOMED database
        |
        v
14. Start supervised queue workers
        |
        v
15. Import one test patient
        |
        v
16. Review complete patient history
        |
        v
17. Test API if required
        |
        v
18. Configure backups
        |
        v
19. DELETE setup_delete_after_setup.php
        |
        v
SYSTEM READY
```

Why the Setup Assistant Must Be Deleted

`setup_delete_after_setup.php` exists only to help bootstrap a new installation.

It performs checks and displays server information that is useful to an administrator during deployment but should not remain exposed afterwards.

It can display information such as:

- installed PHP version

- PHP extensions
- detected CLI PHP path
- expected private filesystem paths
- installation sequence
- application administration locations

It also contains a browser-side `config.php` generator.

Although the generator does not submit the entered credentials to the server, there is no operational reason for the installer to remain publicly available once the server has been configured.

The filename is deliberately explicit:

```
setup_delete_after_setup.php
```

When setup is complete, delete it.