Chapter 10: Hosting Overview and Requirements

Part 3 of this manual is for people who want to run their own Ensemble web panel rather than using the hosted instance at ensemble.virtualportal.space. This chapter covers what you need, what the file layout looks like, and how to configure the installation before you put it behind a web server. Chapters 11 and 12 cover the web server setup itself for nginx and Apache respectively.

If you are happy using the hosted instance and have no interest in running your own server, you can skip Part 3 entirely.

10.1 Who This Is For

Self-hosting Ensemble requires:

- A Linux server with shell access, or a shared hosting plan that supports PHP 8.x and SQLite. A VPS is ideal.
- Confidence with basic Linux commands — creating directories, setting file permissions, editing configuration files.
- A domain name or static IP address that your HUD can reach over HTTPS.

You do not need to know PHP, understand databases, or have any prior web development experience. Ensemble is designed to be straightforward to deploy: there is no build step, no package manager, and no database server to install. You put the files in place, edit one configuration file, and point a web server at the directory.

> **Not technical?** If the steps in this chapter feel unfamiliar, the hosted instance at ensemble.virtualportal.space is the right choice for you. It is maintained, backed up, and requires no setup beyond wearing the HUD. Self-hosting is entirely optional.

10.2 What Ensemble Does Not Require

It is worth being clear about what you do not need, because many PHP applications have heavy dependencies that Ensemble deliberately avoids:

- **No MySQL or MariaDB.** Ensemble uses SQLite, a file-based database that requires no database server, no credentials, and no configuration. The database is created automatically on first boot.
- **No Composer or npm.** There are no PHP or JavaScript package dependencies. The application is self-contained.
- **No root access** for a basic installation. All files live within your web root and a data directory. Standard shared hosting file permissions are sufficient.
- **No cron jobs** for normal operation. The cleanup script (Chapter 13) benefits from a cron job, but the application itself runs entirely on demand.
- **No email configuration.** Ensemble communicates with users via in-world instant message, not email.

10.3 Server Requirements

Operating system

Ensemble is developed and tested on Ubuntu 24 with nginx. Any Linux distribution that supports PHP 8.2 or later will work. macOS works for local development. Windows is not tested and not recommended for production.

PHP version

PHP 8.2 or later is required. PHP 8.1 may work but is not tested. Older versions of PHP are not supported.

Ensemble is designed to run under PHP-FPM, which is the standard mode for modern nginx and Apache setups. The mod_php Apache module also works.

PHP extensions

The table below lists the extensions Ensemble uses and whether each is required.

PHP Extension	Used for	Required?
`pdo_sqlite`	Database (SQLite via PDO)	✓ Required
`gd`	Outfit image upload and resizing	✓ Required
`zip`	Backup download and restore (ZipArchive)	✓ Required
`fileinfo`	MIME type detection for uploaded images	✓ Required
`json`	API requests and responses; backup files	✓ Built into PHP 8
`session`	User login sessions	✓ Built into PHP 8
`curl`	Not used directly — HTTP to HUD uses file_get_contents	Not required
`opcache`	Performance — caches compiled PHP bytecode	Recommended

Most of these are enabled by default in standard PHP 8.x packages. On Ubuntu/Debian you can install the commonly needed ones with:

```
sudo apt install php8.2-fpm php8.2-sqlite3 php8.2-gd php8.2-zip
```

Check which are already enabled on your system with `php -m`. Extensions listed as “Built into PHP 8” (json, session) are always present and need no separate installation.

Web server

Either nginx or Apache 2.4+ is supported. Chapter 11 covers nginx and Chapter 12 covers Apache. Both are equally capable for this application; the choice depends on what you are already running or comfortable with.

HTTPS

HTTPS is strongly recommended. The HUD communicates with the web panel over HTTP from inside the OpenSimulator simulator process. On most grids, the simulator will refuse to make outbound HTTP requests to plain HTTP (non-TLS) URLs, or will do so only on specific allowed ports. Without HTTPS your HUD will not be able to connect.

The simplest way to get HTTPS on a Linux server is Let's Encrypt with Certbot:

```
sudo apt install certbot python3-certbot-nginx

sudo certbot --nginx -d yourdomain.example.com
```

Certbot will obtain a certificate and configure your nginx server block automatically. Renewal is handled automatically by a systemd timer installed with the package.

10.4 File Structure

The recommended layout puts the PHP application files inside the web root and keeps the database and uploads directory one level above it, outside the web root. This ensures that the database file and uploaded images cannot be accessed directly via a browser URL, even if misconfigured.

```
/srv/ensemble/              ← parent directory (outside web root)
   data/                    ← SQLite database lives here
      ensemble.db           ← created automatically on first boot
   uploads/                 ← outfit images live here
         <uuid>/            ← one subdirectory per user
   www/                     ← web root (served by nginx/Apache)
         index.php
         api.php
         auth.php
         backup.php
         config.php
         db.php
         image.php
         view.php
         themes/
               default/
                     style.css
                     ensemblelogo.png
                     placeholder.png
               dark/
                     style.css
                     ensemblelogo.png
                     placeholder.png
```

The exact parent directory path (`/srv/ensemble/` in the example above) is up to you. Common alternatives include `/var/www/ensemble/`, `/home/yourusername/ensemble/`, or any path your user account has write access to.

The web root — the `www/` directory in the example — is what you configure in your web server as the document root. The `data/` and `uploads/` directories sit alongside it, not inside it.

> **Shared hosting note:** On shared hosting you may not be able to create directories above your web root (often `public_html/`). In that case, put `data/` and `uploads/` inside `public_html/` but protect them with a `.htaccess` (for Apache) file that denies all HTTP access. Chapter 12 covers this for Apache installations. Nginx installations will need server or location blocks to protect the database and uploads – if you are not able to do that then nginx hosting may not be the best choice for you as .htaccess files are Apache ONLY.

10.5 Configuring config.php

`config.php` is the only file you need to edit. It is a short file with four constants. Open it in a text editor and set each value before starting the web server.

DB_PATH

The full filesystem path to the SQLite database file. The default value is:

```
define('DB_PATH', __DIR__ . '/../data/ensemble.db');
```

This puts the database one level above the PHP files, in the `data/` directory described in Section 10.4. The `data/` directory must exist and be writable by the PHP process before Ensemble is first accessed. If it does not exist, `db.php` will attempt to create it automatically, but this will fail if the parent directory is not writable by PHP.

If you need to use a different path — for example on shared hosting where you have a fixed home directory — change this to an absolute path:

```
define('DB_PATH', '/home/yourusername/ensemble_data/ensemble.db');
```

UPLOADS_DIR

The full filesystem path to the directory where outfit images are stored. The default is:

```
define('UPLOADS_DIR', __DIR__ . '/../uploads/');
```

Like `DB_PATH`, this sits one level above the web root by default. This directory must exist and be writable by the PHP process. Ensemble will create per-user subdirectories inside it automatically.

LINKS_MAX

The maximum number of shared links any single user account can have active at once. The default is `20`. Increase this if you are running a multi-user installation where users need more links, or decrease it to conserve resources.

```
define('LINKS_MAX', 20);
```

ACCESS_CODE

An optional access code that HUDs must present when checking in. If set to a non-empty string, only HUDs whose wearer has entered the matching code via the HUD's Settings menu will be accepted. Leave empty to allow any HUD to connect.

```
define('ACCESS_CODE', '');
```

The access code is useful if you are running a private installation that you want to share only with specific people. Set it to any string you choose, and give that string to the people you want to be able to connect. They enter it in the HUD's **Settings › Access Code** menu (Chapter 8, Section 8.6).

> **Note:** The access code is not a password for the web panel login. It is a gate for HUD-to-server communication only. Users still need their own web panel credentials to log in; the access code simply prevents unknown HUDs from registering new accounts on your installation.

10.6 The Database

Ensemble uses SQLite, a file-based database engine built into PHP. There is no database server to install, start, or configure. The database file is created automatically the first time the web panel is accessed.

WAL mode

The database is opened in WAL (Write-Ahead Log) mode. This allows concurrent reads while a write is in progress, which is important because the HUD sends heartbeats frequently while users are simultaneously browsing the web panel. WAL mode is set automatically by `db.php` on every connection; no manual configuration is needed.

Schema migrations

As Ensemble is updated, the database schema evolves. New columns are added to existing tables using `ALTER TABLE` statements that run on every boot. These statements are safe to run repeatedly: if a column already exists, the error is caught and silently ignored. This means schema updates are applied automatically when you replace the PHP files with a newer version — no manual SQL is required.

The schema is entirely managed by `db.php`. You should never need to edit the database directly.

Backing up the database

The database file is a single file at the path set in `DB_PATH`. To back it up, copy it while the application is not actively writing. Because SQLite uses WAL mode, a simple file copy is safe even while the application is running, though a brief moment of low traffic is ideal.

A simple cron-based backup copies the file to a safe location on a schedule:

```
0 3 * * * cp /srv/ensemble/data/ensemble.db
/srv/ensemble/backups/ensemble_$(date +\%Y\%m\%d).db
```

Ensemble's own Backup feature (Chapter 9) backs up user data within the application, but does not back up the database file itself. For a complete server backup, include the database file and the uploads directory.

10.7 File and Directory Permissions

The PHP process needs read access to all PHP files and the themes directory, and write access to the `data/` and `uploads/` directories. A clean permission setup looks like this:

```
# PHP files — readable by web server, not world-writable chmod 644
/srv/ensemble/www/*.php chmod -R 644 /srv/ensemble/www/themes/ chmod
755 /srv/ensemble/www/ chmod 755 /srv/ensemble/www/themes/ chmod 755
/srv/ensemble/www/themes/default/ chmod 755
/srv/ensemble/www/themes/dark/  # Data and uploads — writable by PHP
process chown -R www-data:www-data /srv/ensemble/data/ chown -R www-
data:www-data /srv/ensemble/uploads/ chmod 755 /srv/ensemble/data/
chmod 755 /srv/ensemble/uploads/
```

Replace `www-data` with the user your PHP-FPM or web server process runs as. On Ubuntu with nginx and PHP-FPM this is typically `www-data`. On other distributions it may be `nginx`, `apache`, or `http`.

> **Tip:** If you are unsure which user PHP runs as, create a temporary PHP file containing `<?php echo posix_getpwuid(posix_geteuid())['name'];` and access it via your browser. Remove the file immediately after checking.

10.8 Security Considerations

Keep data outside the web root

The most important security measure is keeping `data/` and `uploads/` outside the web root, as described in Section 10.4. If these directories are inside the web root, a misconfigured web server could allow anyone to download your SQLite database (containing password hashes, outfit data, and session tokens) or browse uploaded images.

If you are on shared hosting and cannot place them outside the web root, protect them with a `.htaccess` file (only if you are using Apache):

```
# /public_html/data/.htaccess and /public_html/uploads/.htaccess Deny
from all
```

On nginx, if you cannot place the database and uploads above the web root, and cannot edit the nginx server and/or location blocks, then you will be unable to protect the database or uploads!

HTTPS everywhere

Ensure your installation is only accessible over HTTPS. Configure your web server to redirect all HTTP traffic to HTTPS. The HUD communicates with the panel over the open internet; unencrypted connections expose login credentials and session tokens in transit.

config.php

`config.php` contains the access code if you have set one. Ensure this file is not readable by the web server as a text file. PHP files are executed, not served, so under a correctly configured web server `config.php` will never be sent to a browser. If you have any doubt, verify this by trying to access `https://yourdomain.example.com/config.php` in a browser — you should receive an empty page or a PHP error, never the source code.

The uploads directory

Uploaded outfit images are stored in `uploads/` with filenames that are 32-character hex strings plus `.jpg`. The directory should not be executable and should not contain PHP files. Ensemble enforces this in `image.php` — files are validated against the expected filename format before being written — but a belt-and-braces check at the web server level is worthwhile. Chapter 11 shows how to do this in the nginx configuration.

10.9 Upgrading Ensemble

To upgrade Ensemble to a newer version:

- Download the new release from the OpenSimWorld Script Library.
- Replace the PHP files in your web root with the new versions. Do not delete `config.php` — copy it from your existing installation to preserve your settings.
- The `themes/` directory can be replaced wholesale; custom community themes should be backed up first and re-added after the upgrade.
- No database migration is needed. The `db.php` bootstrap runs automatically on first access and applies any new schema changes.

There is no downtime procedure required. PHP-FPM processes the upgrade on the first request after the files are replaced. Users mid-session will see a brief moment of inconsistency at most — the database schema is updated on the first connection after upgrade, which takes milliseconds.

> **Tip:** Before upgrading, download a backup from the web panel (Backup › Download Backup) so you have a safe copy of all user data. Also copy the database file directly as described in Section 10.6. This gives you two layers of recovery if anything goes wrong.

The next chapter walks through setting up Ensemble with nginx, from a fresh VPS to a working installation.