Chapter 12: Installing on Apache

This chapter covers installing Ensemble on Apache 2.4 with PHP 8.2. The instructions follow the same structure as Chapter 11 and reach the same end result — a working Ensemble installation served over HTTPS with the database and uploads directory kept outside the web root. If you read Chapter 11, you will find this chapter familiar; the differences are entirely in the web server configuration.

Apache is a good choice if you are on shared hosting (where it is almost universally available), already running Apache for other sites, or prefer its `.htaccess` override system. The instructions are written for Ubuntu 24; they apply equally to Debian 12 and closely to other distributions.

12.1 Prerequisites

Before starting, make sure you have:

- A Linux server with a public IP address or domain name.
- A domain name pointed at the server with DNS propagated.
- Root or sudo access.
- Port 80 and 443 open in your firewall.

All commands below use `sudo`. The example domain is `ensemble.example.com` — substitute your actual domain everywhere it appears.

12.2 Install Apache and PHP 8.2

1. Update the package list:

```
sudo apt update
```

2. Install Apache:

```
sudo apt install -y apache2
```

3. Install PHP 8.2 and the required extensions. The `libapache2-mod-php8.2` package integrates PHP directly into Apache as a module (the simplest approach for a dedicated server):

```
sudo apt install -y php8.2 libapache2-mod-php8.2 \     php8.2-sqlite3 php8.2-gd php8.2-zip
```

4. Confirm Apache is running:

```
sudo systemctl status apache2
```

5. Enable the required Apache modules:

```
sudo a2enmod rewrite ssl headers
```

6. Restart Apache to load the newly enabled modules:

```
sudo systemctl restart apache2
```

PHP-FPM alternative: If you are running PHP-FPM rather than mod_php (common on servers hosting multiple PHP versions), replace `libapache2-mod-php8.2` with `php8.2-fpm` and enable `mod_proxy_fcgi` and `mod_proxy` instead: `sudo a2enmod proxy_fcgi proxy`. The VirtualHost configuration differs slightly — Section 12.6 includes a note on this.

12.3 Create the Directory Structure

The directory layout is identical to Chapter 11 — `www/` inside a parent directory, with `data/` and `uploads/` sitting alongside it outside the web root:

```
sudo mkdir -p /srv/ensemble/www sudo mkdir -p /srv/ensemble/data sudo
mkdir -p /srv/ensemble/uploads
```

Set ownership so the Apache process (`www-data` on Ubuntu) can write to `data/` and `uploads/`:

```
sudo chown -R $USER:$USER /srv/ensemble/www sudo chown -R www-
data:www-data /srv/ensemble/data sudo chown -R www-data:www-data
/srv/ensemble/uploads sudo chmod 755 /srv/ensemble/data sudo chmod
755 /srv/ensemble/uploads
```

12.4 Deploy the Ensemble Files

Copy the Ensemble PHP files and `themes/` directory into `/srv/ensemble/www/`. The directory should contain:

```
index.php    api.php    auth.php    backup.php config.php   db.php
image.php   view.php themes/
```

Set permissions on the PHP files:

```
chmod 644 /srv/ensemble/www/*.php find /srv/ensemble/www/themes -type
f -exec chmod 644 {} \; find /srv/ensemble/www/themes -type d -exec
chmod 755 {} \;
```

12.5 Configure config.php

Open `config.php`:

```
nano /srv/ensemble/www/config.php
```

The default paths resolve correctly for this layout — `__DIR__ . '/../'` points to `/srv/ensemble/` where `data/` and `uploads/` live. No changes are needed unless you used a different layout:

```
define('DB_PATH',      __DIR__ . '/../data/ensemble.db');
define('UPLOADS_DIR', __DIR__ . '/../uploads/'); define('LINKS_MAX',
20); define('ACCESS_CODE', '');
```

Set `ACCESS_CODE` if you want to restrict HUD connections to users who have been given the code (see Chapter 10, Section 10.5).

12.6 Create the Apache VirtualHost

Create a new VirtualHost configuration file for Ensemble:

```
sudo nano /etc/apache2/sites-available/ensemble.conf
```

Paste in the following configuration, replacing `ensemble.example.com` with your actual domain:

```
<VirtualHost *:80>
    ServerName ensemble.example.com
    # Redirect all HTTP to HTTPS.
    # Comment this out until after Certbot runs.
    RewriteEngine On
    RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
</VirtualHost>

<VirtualHost *:443>
    ServerName ensemble.example.com
    DocumentRoot /srv/ensemble/www

    # SSL certificate — filled in by Certbot
    SSLEngine on
    SSLCertificateFile /etc/letsencrypt/live/ensemble.example.com/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/ensemble.example.com/privkey.pem
    Include /etc/letsencrypt/options-ssl-apache.conf

    <Directory /srv/ensemble/www>
            Options -Indexes -FollowSymLinks
            AllowOverride All
            Require all granted
    </Directory>

    # Block direct browser access to the uploads directory.
    # Uploads live outside the web root so this is a belt-and-braces
    # guard in case the layout is ever changed.
    <DirectoryMatch "^/srv/ensemble/(data|uploads)">
            Require all denied
    </DirectoryMatch>

    # Security headers
    Header always set X-Content-Type-Options "nosniff"
    Header always set X-Frame-Options "SAMEORIGIN"

    # Cache static theme assets
    <FilesMatch "\.(css|js|png|jpg|ico|woff2?)$">
            Header set Cache-Control "public, max-age=604800"
    </FilesMatch>

    ErrorLog  ${APACHE_LOG_DIR}/ensemble_error.log
    CustomLog ${APACHE_LOG_DIR}/ensemble_access.log combined
</VirtualHost>
```

PHP-FPM note: If you are using PHP-FPM instead of mod_php, add the following inside the `<VirtualHost *:443>` block (adjust the socket path for your PHP version): <FilesMatch "\.php$"> SetHandler "proxy:unix:/run/php/php8.2-fpm.sock|fcgi://localhost" </FilesMatch>

First-time setup: If you have not yet run Certbot, comment out the SSL lines in the 443 VirtualHost and temporarily remove the HTTP-to-HTTPS redirect from the port 80 block. Replace the 80 block with a plain VirtualHost serving from `DocumentRoot` on port 80. Get the site working over HTTP first, then run Certbot (Section 12.7) to add HTTPS.

Please note: I do not use Apache and this was entirely generated by Claude.AI. Please check before deplying!

Enable the site and disable the default Apache site if it is active:

```
sudo a2ensite ensemble.conf sudo a2dissite 000-default.conf
```

Test the configuration and reload Apache:

```
sudo apachectl configtest sudo systemctl reload apache2
```

`apachectl configtest` should report `Syntax OK`. If it reports an error, the message will point to the problem line. Fix it before reloading.

12.7 Obtain an SSL Certificate with Certbot

1. Install Certbot and the Apache plugin:

```
sudo apt install -y certbot python3-certbot-apache
```

2. Run Certbot for your domain:

```
sudo certbot --apache -d ensemble.example.com
```

3. Follow the prompts. When asked about the redirect, choose **Redirect** (option 2) to enforce HTTPS.
4. Certbot will update your VirtualHost configuration with the certificate paths and reload Apache automatically.
5. Verify automatic renewal:

```
sudo certbot renew --dry-run
```

A successful dry-run means renewal is working. Let's Encrypt certificates are renewed automatically via a systemd timer; no manual action is needed.

12.8 AllowOverride and .htaccess

The VirtualHost in Section 12.6 sets `AllowOverride All` for the web root. This permits `.htaccess` files inside `/srv/ensemble/www/` to override Apache directives per-directory. Ensemble does not ship with a `.htaccess` file, so this setting has no immediate effect — but it means you can add one if needed for your specific hosting environment without editing the VirtualHost.

If `AllowOverride All` is not available to you (for example on managed hosting where you cannot edit VirtualHost files), you can replace it with `AllowOverride None` and rely entirely on the VirtualHost configuration. Ensemble functions correctly either way.

Shared hosting without VirtualHost access

On shared hosting you typically have a `public_html/` directory (or similar) as your web root, and you cannot create VirtualHost files. To install Ensemble in this environment:

1. Upload the Ensemble PHP files and `themes/` into `public_html/` (or a subdirectory, e.g. `public_html/ensemble/`).
2. Create `data/` and `uploads/` directories. On shared hosting these must usually live inside `public_html/` since you cannot write above it. Create them as subdirectories:

```
public_html/ensemble/data/ public_html/ensemble/uploads/
```

3. Update `config.php` to use absolute paths to those directories:

```
define('DB_PATH',
'/home/yourusername/public_html/ensemble/data/ensemble.db');
define('UPLOADS_DIR',
'/home/yourusername/public_html/ensemble/uploads/');
```

4. Create a `.htaccess` file in each of `data/` and `uploads/` to block direct HTTP access:

```
# public_html/ensemble/data/.htaccess #
public_html/ensemble/uploads/.htaccess Require all denied
```

With this setup, the SQLite database and uploaded images are inside the web root but protected by `.htaccess`. On a correctly configured Apache installation with `AllowOverride` enabled, these directories will return 403 if accessed directly via a browser URL.

> **Verify protection:** After setting up, try visiting `https://yourdomain.example.com/ensemble/data/` in a browser. You should receive a 403 Forbidden response. If you can browse the directory listing or download files, the `.htaccess` protection is not working and your host may have `AllowOverride None` set. Contact your host to enable `.htaccess` overrides, or ask them to add the `Deny from all` rule at the server level.

12.9 First Boot and Verification

Open `https://ensemble.example.com` in a browser. You should see the Ensemble login page. The database is created automatically on the first request.

> *[SCREENSHOT ch12-01-first-boot-login.png: The Ensemble login page loading correctly in a browser over HTTPS]*

Check that the database file has been created:

```
ls -la /srv/ensemble/data/
```

You should see `ensemble.db` (and accompanying WAL files once activity begins). If the file is not present, check the PHP error log (Section 12.10) for a permissions error.

Wear the HUD in-world, set the web URL in the HUD's Settings menu, and follow the first-boot flow in Chapter 2.

12.10 Common Issues

403 Forbidden on the web root

Apache is refusing to serve files from `/srv/ensemble/www/`. Two common causes:

- **`Require all granted` missing or overridden.** Confirm the `<Directory>` block in the VirtualHost contains `Require all granted`. On older Apache setups the equivalent is `Allow from all`.

- **Directory permissions.** The `www-data` user needs execute permission on every directory in the path. Check each level: `/srv/`, `/srv/ensemble/`, `/srv/ensemble/www/` all need at least `755`.

```
sudo chmod 755 /srv/ sudo chmod 755 /srv/ensemble/ sudo chmod 755
/srv/ensemble/www/
```

PHP source displayed instead of executed

If the browser shows raw PHP code, the PHP module is not active or not handling `.php` files. Check:

- mod_php is enabled: `sudo apache2ctl -M | grep php`. If absent, run `sudo a2enmod php8.2 && sudo systemctl restart apache2`.
- If using PHP-FPM, confirm the `SetHandler` directive in the VirtualHost is present and `mod_proxy_fcgi` is enabled.

500 Internal Server Error

A 500 error usually means a PHP error. Check Apache's error log:

```
sudo tail -50 /var/log/apache2/ensemble_error.log
```

Common causes: a missing PHP extension (`php8.2-sqlite3` or `php8.2-gd` not installed), a syntax error in `config.php`, or a path in `DB_PATH`/`UPLOADS_DIR` that does not exist or is not writable.

To see PHP errors directly in the browser during troubleshooting, temporarily add the following to the top of `index.php` (remove before going live):

```
<?php ini_set('display_errors', 1); ini_set('display_startup_errors',
1); error_reporting(E_ALL);
```

Database permission error / blank page after login

If the login page appears but logging in fails, or you see an error about the database, the PHP process cannot write to `data/`. Check ownership:

```
ls -la /srv/ensemble/ # data/ should show owner www-data sudo chown -
R www-data:www-data /srv/ensemble/data/ sudo chmod 755
/srv/ensemble/data/
```

RewriteEngine issues

If the HTTP-to-HTTPS redirect is not working, confirm `mod_rewrite` is enabled:

```
sudo apache2ctl -M | grep rewrite
```

If absent:

```
sudo a2enmod rewrite sudo systemctl restart apache2
```

Also confirm the `<Directory>` block has `AllowOverride All` (or at minimum `AllowOverride FileInfo`) if the redirect is in a `.htaccess` file rather than the VirtualHost.

HUD cannot connect

If the HUD reports it cannot reach the web panel:

- Visit `https://ensemble.example.com` in a browser on a different device to confirm the site is reachable externally.
- Check that port 443 is open: `sudo ufw status`. If it is not listed, run `sudo ufw allow 443/tcp && sudo ufw allow 80/tcp`.
- Verify the SSL certificate is valid and not expired: a browser padlock should show no warnings.
- Confirm the URL entered in the HUD's Settings › Set Web URL is exactly `https://ensemble.example.com` with no trailing slash.

> **Log files:** Apache error logs for this installation are at `/var/log/apache2/ensemble_error.log` (as specified in the VirtualHost). The access log is at `/var/log/apache2/ensemble_access.log`. PHP errors are written to the Apache error log when using mod_php. These are the first places to look when diagnosing any problem.

The next chapter covers the cleanup script and ongoing maintenance tasks for a self-hosted installation.