



Installation & Administration

Version 1.0.0 · 2026-10-10

Release preparation

Online: <https://docs.opencorehr.com>

Contents

1. Installation & Administration
2. Requirements & hosting
3. Prepare your installation
4. Install on cPanel
5. Install on a Linux VPS
6. Plesk, Apache & Windows
7. Local development
8. Configure & initialize
9. Email, queues & real-time services
10. Installation checklist
11. Backups & updates
12. Customization & integrations
13. Troubleshooting & FAQ
14. Support, licences & credits

Installation & Administration

This guide is for the person installing and maintaining Open Core HR. For everyday HR and employee tasks, open the [User Manual](#).

Open Core HR is a self-hosted Laravel application supplied with source code. Its marketing and documentation sites can run on Cloudflare; the HR application needs a PHP application server and database.

This guide accompanies **version 1.0.0 in release preparation**. These deployment recipes are checked against repository configuration. Fresh installations on the named hosting platforms still need verification before those platforms are advertised as supported.

Installation sequence

1. Check [requirements and hosting](#).
2. [Prepare the source package, domain and database](#).
3. Follow the recipe for [cPanel](#), [Linux VPS](#), [another platform](#) or [local development](#).
4. [Configure and initialize](#) the application.
5. Set up [email and background services](#).
6. Complete the [installation checklist](#).

Keep third-party licence notices with the installation. Use the final purchase package for customer licence and support terms.

Requirements & hosting

Application requirements

Component	Requirement
PHP	8.4 or a compatible later version accepted by the supplied Composer lockfile
Composer	2.x; verify with <code>composer check-platform-reqs --no-dev</code>
PHP extensions	All extensions required by <code>composer.lock</code> ; commonly curl, PDO/MySQL, zip, mbstring, XML/DOM, fileinfo and OpenSSL
Database	MySQL; choose a maintained version and validate migrations on that exact version
Web server	Nginx or Apache with its document root set to <code>backend/public</code>
Frontend build	Node.js compatible with the locked Vite dependencies; use Node 22 LTS for the build environment
Background services	A scheduler every minute and a persistent worker for asynchronous jobs
HTTPS	Required for a customer-facing deployment
Real-time features	Persistent Reverb process and WebSocket proxy when enabled

The backend README's MySQL 5.7 and Node 18 entries are historical minimums, not a current hosting recommendation. The supplied lockfiles and target-platform validation are authoritative.

Choose your hosting

Platform	Confirm with the provider
Linux VPS	Compatible PHP/CLI, database, SSH, cron, web server and process manager
cPanel / Plesk	Compatible web and CLI PHP, Composer/SSH, configurable document root, cron and persistent workers
Shared hosting	All requirements above; many plans cannot run workers or WebSockets
Windows / IIS	PHP/MySQL compatibility, URL rewriting, scheduling and service management; separate acceptance required

Prefer a VPS for live chat, calls and other real-time features. Cloudflare DNS/proxy does not provide a PHP runtime or MySQL database.

Verification status

Fresh-installation acceptance on cPanel, Linux VPS, Plesk/Apache and Windows remains pending. Local regression tests do not certify these hosting environments.

Prepare your installation

1. Extract your purchased release package into a private working directory.
2. Locate `backend/` . Run application commands from that directory.
3. Create a dedicated database and a user restricted to that database.
4. Point an application hostname, such as `hr.example.com` , to the application server.
5. Configure HTTPS and set the web document root to the full path of `backend/public` .
6. Arrange SMTP delivery, scheduled tasks and worker services.

Do not expose the repository root, `.env` , source, backups or private uploads through the public document root.

Install dependencies

```
cd backend
composer install --no-dev --prefer-dist --optimize-autoloader
composer check-platform-reqs --no-dev
npm ci
npm run build
```

Use the supplied lockfiles. If Node.js is unavailable on the server, build on a trusted machine and upload the matching `public/build` with the release source. Do not deploy a development `public/hot` file. Composer dependencies must match the server's PHP platform.

Create private configuration

On a new installation only:

```
cp .env.example .env
php artisan key:generate
php artisan jwt:secret
```

Continue with [configuration](#). Do not overwrite an existing `.env` or regenerate encryption/authentication keys during updates.

Install on cPanel

This recipe requires compatible web/CLI PHP, Terminal/SSH, Composer, cron and persistent worker support. cPanel alone does not guarantee these capabilities.

1. In **Select PHP Version** or **MultiPHP Manager**, select compatible PHP and its required extensions. Confirm the Terminal and cron PHP binary uses the same compatible version.
2. In **MySQL Databases**, create the database and user, assign permissions on this database, and record the full prefixed names privately.
3. Extract the backend outside `public_html`, for example `/home/account/open-core-hr/backend`.
4. In **Domains**, set the application hostname's document root to `/home/account/open-core-hr/backend/public`.
5. Enable HTTPS with the host's SSL/AutoSSL controls.
6. Open Terminal, change into `backend`, and follow [preparation](#) and [configuration](#).
7. In **Cron Jobs**, run the scheduler every minute using absolute paths:

```
/path/to/php /home/account/open-core-hr/backend/artisan schedule:run >> /home/account/open-core-hr/backend/storage/logs/scheduler.log 2>&1
```

Use the provider's actual compatible PHP binary. Monitor and rotate this log. Ask the provider to supervise `queue:work` and Reverb if needed; see [services](#).

If the host cannot set `public` as the document root or run required services, use a compatible plan or VPS. Moving the front controller into a public repository root is not an approved workaround.

Finish the [installation checklist](#). cPanel acceptance remains pending.

Install on a Linux VPS

Use a maintained Linux distribution. Install Nginx, compatible PHP-FPM/CLI and extensions, MySQL, Composer and a process manager using supported packages. Package names and socket paths vary by distribution.

1. Create a deployment account and dedicated database/user.
2. Put the application in `/srv/open-core-hr/backend` or your chosen private path.
3. Follow [preparation](#) and [configuration](#).
4. Give PHP-FPM write access to `storage` and `bootstrap/cache`. Keep source and secrets private; do not use world-writable permissions.
5. Configure Nginx, replacing the hostname, path and socket below:

```
server {  
    listen 80;  
    server_name hr.example.com;  
    root /srv/open-core-hr/backend/public;  
    index index.php;  
    client_max_body_size 20m;  
  
    location / {  
        try_files $uri $uri/ /index.php?$query_string;  
    }  
    location ~ ^/index\.php(/|$) {  
        include fastcgi_params;  
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;  
        fastcgi_pass unix:/run/php/php8.4-fpm.sock;  
        fastcgi_hide_header X-Powered-By;  
    }  
    location ~ \.php$ { return 404; }  
    location ~ /\.(!well-known).* { deny all; }  
}
```

6. Test the Nginx configuration and reload it.
7. Issue an HTTPS certificate and configure HTTP-to-HTTPS redirection.
8. Set up [cron and supervised services](#).
9. Complete the [installation checklist](#).

Align upload limits in PHP, Nginx and application validation. This starting configuration still needs acceptance on the exact Linux/PHP stack.

Plesk, Apache & Windows

These environments require their own fresh-installation checks before being advertised as supported.

Plesk

Create the application domain, select compatible PHP for web and CLI, set its document root to `backend/public`, create its database and enable SSL. Use SSH for [preparation](#) and [configuration](#). Configure Scheduled Tasks for `artisan schedule:run` every minute. Arrange persistent queue/Reverb services with the hosting provider.

Apache

Set `DocumentRoot` to `backend/public`. Enable `mod_rewrite`, permit the supplied `public/.htaccess` rules, configure compatible PHP and HTTPS, and deny access to hidden files. Follow common setup and services instructions. If only the homepage works, check rewriting and virtual-host settings.

Windows / IIS

Install compatible PHP/extensions and MySQL. Set the site root to `backend/public`. Configure IIS URL Rewrite rules equivalent to the Apache front controller; IIS does not read `.htaccess`. Adapt Unix commands for PowerShell. Use Task Scheduler for the scheduler and supervised Windows services for queue workers and Reverb. Verify filesystem permissions and storage links.

Windows production hosting is not currently certified by this guide. Arrange deployment-specific validation or use the VPS recipe.

Local development

Install compatible PHP, Composer, MySQL and Node.js. Use a separate local database with demonstration data only.

Follow [preparation](#), using `composer install` with development dependencies instead of `--no-dev`. Set `APP_ENV=local`, configure a new local database, and follow [configuration](#).

```
# Terminal 1, from backend
php artisan serve --host=127.0.0.1 --port=8000

# Terminal 2, from backend
npm run dev
```

Set `APP_URL=http://localhost:8000`. Run additional services when testing asynchronous or real-time features. Use local mail capture instead of customer email accounts.

Use a real web server and compiled assets for customer hosting. Keep local configuration and uploads out of source packages.

Configure & initialize

Edit `.env` privately. Set the real application URL and database credentials. The values below are placeholders:

```
APP_ENV=production
APP_DEBUG=false
APP_DEMO=false
APP_URL=https://hr.example.com
APP_TIMEZONE=UTC
TELESCOPE_ENABLED=false
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=your_database
DB_USERNAME=your_database_user
DB_PASSWORD=your_private_password
QUEUE_CONNECTION=database
CACHE_STORE=file
SESSION_DRIVER=file
DOCUMENTATION_URL=https://docs.opencorehr.com
```

`APP_URL` also controls password reset email links. Configure SMTP before testing recovery. Review locale defaults in `.env.example` ; manage company localization in the app after installation.

Initialize a fresh database

The current initialization uses command-line tools, with no browser installer. On a **new, empty database only**:

```
php artisan migrate --force
php artisan db:seed --force
php artisan storage:link
php artisan optimize
```

Review the supplied seeders and enabled modules before customer use. Current seeders create a known demonstration administrator (`superadmin@demo.com` , password `123456`) and can include module fixtures. `APP_DEMO=false` disables the optional demo seeder; it does not remove every sample record or change this password.

Keep access restricted while initializing. Immediately change the administrator email/password, review seeded data, and verify role/portal access before opening to employees. A secure customer bootstrap and clean seed-data audit remain release acceptance items.

Never reset a database, run `migrate:fresh` , re-run demo seeders or regenerate existing keys during updates.

Files and modules

The web process needs write access to `storage` and `bootstrap/cache` . Keep private documents private; do not link all of storage publicly. Use authorized application downloads.

Review optional features in **Modules**. Disabling a module changes app availability; its public documentation remains visible and labels the required module.

Email, queues & real-time services

Email

Configure the provider's SMTP host, port, credentials and sender address in private configuration. Match transport/encryption settings to the supplied `config/mail.php`. Verify the sender domain and test real inbox delivery. `log` and `array` mailers do not deliver email.

Test password recovery and confirm the email link opens the reachable HTTPS application hostname.

Queue worker

For a database queue, apply its migrations and supervise a worker from `backend`:

```
php artisan queue:work --sleep=3 --tries=3 --timeout=90
```

Ensure queue `retry_after` exceeds the worker timeout. Run with access to storage/configuration. Use systemd, Supervisor or the host's worker manager to restart on failure and reboot. After code deployment:

```
php artisan queue:restart  
php artisan queue:failed
```

The `sync` queue runs jobs inline; it does not demonstrate asynchronous worker health.

Scheduler

Add a cron entry every minute:

```
* * * * * cd /srv/open-core-hr/backend && /path/to/php artisan schedule:run >> /srv/open-core-  
hr/backend/storage/logs/scheduler.log 2>&1
```

Replace paths, monitor/rotate logs, review `php artisan schedule:list` and confirm timezones. Modules can contribute additional schedules when enabled.

Real-time services

Configure `REVERB_*` and public `VITE_REVERB_*` before building assets. Keep private Reverb secrets out of frontend variables. Supervise `php artisan reverb:start` and configure HTTPS/WSS proxying with WebSocket upgrade headers. Restart long-running services after configuration changes.

Microsoft 365, Firebase and other external providers need customer-owned credentials and separate live acceptance. Do not ship configured service accounts.

Installation checklist

Record product version, hosting platform, PHP/database versions, date and outcome. Complete before inviting employees:

- ☐ Composer platform requirements pass and migrations finish.
- ☐ HTTPS works; only `backend/public` is served.
- ☐ Secrets, logs, source, backups and private documents cannot be downloaded publicly.
- ☐ Debug/demo modes are off; keys are private and backed up.
- ☐ Initial administrator email/password have been changed.
- ☐ Seeded users, master data and module fixtures are reviewed.
- ☐ Administrator and employee accounts reach the expected portals.
- ☐ Employee creation, attendance, leave and approvals work with expected roles.
- ☐ Uploads and authorized downloads work; unrelated employees cannot read private files.
- ☐ SMTP reaches an inbox; password recovery uses the correct hostname.
- ☐ Queue jobs run and workers survive reboot; failed jobs are reviewed.
- ☐ Cron and optional-module schedules run.
- ☐ WebSocket features work when included.
- ☐ Mobile API/live integrations pass when included; mocks do not replace acceptance.
- ☐ A backup restores successfully to an isolated installation.
- ☐ Online help and the offline documentation package work.

Documentation publication is separate from customer-installation acceptance, secure bootstrap review and live integration verification.

Backups & updates

Backups

Back up the database, private/uploaded files, `.env`, encryption/OAuth keys, module settings and customizations. Encrypt backups, restrict access, retain an off-server copy and test restoration in isolation. Do not send production backups to support.

Updates

1. Read release notes and review your installed version/customizations.
2. Back up data, keys and configuration; confirm restoration.
3. Test the update on a restored staging copy.
4. Enter maintenance mode and stop/drain workers during the agreed window.
5. Deploy matching source, dependencies and built assets. Preserve `.env`, storage and existing keys.
6. Run only the update migrations and refresh configuration from `backend` :

```
php artisan down
composer install --no-dev --prefer-dist --optimize-autoloader
php artisan migrate --force
php artisan optimize
php artisan queue:restart
php artisan up
```

7. Restart other services and repeat acceptance smoke checks.

Do not run fresh-install seeders unless the release explicitly requires a reviewed data upgrade. Reverting source alone does not reverse database changes; follow the tested restore/rollback plan.

Customization & integrations

Start with company details, localization, roles, approval workflows and module settings. See the User Manual's [settings overview](#) and [roles and permissions](#).

For source changes, keep a private version-controlled copy, document changes and re-test after updates. Keep business rules in the backend and use the shared API across clients. Rebuild assets after changing frontend source/public configuration. Never put secrets in `VITE_*` variables.

Provider integrations require customer accounts, permissions and their own costs. Enable only configured and tested integrations. Distribute examples rather than customer credentials.

For employee-app source builds, use `apps/employee/README.md` in the matching release. Signing, store publication and physical-device acceptance are separate tasks.

Troubleshooting & FAQ

500 error

Read `storage/logs` and web/PHP logs privately. Check PHP compatibility, dependencies, database credentials, keys and writable storage/cache. After configuration changes, run `php artisan optimize` and restart services. Keep public debug mode off. Share sanitized errors only.

Blank page or missing assets

Build the matching frontend source and deploy `public/build`. Remove a development `public/hot` file. Check document root, asset URLs and browser errors.

Homepage works; other pages return 404

Check Nginx `try_files`, Apache rewriting or IIS URL Rewrite. The root must be `backend/public`.

Login fails or returns 419

Check `APP_URL`, HTTPS, sessions, application key, writable session storage and cookies. Clear stale cookies after correcting the hostname. Check account status and roles if the wrong portal opens.

Reset emails do not arrive

Verify SMTP/sender domain, spam folders, queue workers and failed jobs. A log mailer does not deliver. Confirm reset links use the reachable application hostname.

Upload/download errors

Check storage permissions, disk settings and PHP/server limits. Private files require authorized downloads. Do not expose private storage to work around authorization errors.

Live updates fail

Check module enablement, Reverb, WSS hostname, proxy upgrade headers and firewalls. Rebuild after `VITE_REVERB_*` changes. Shared hosting may lack persistent WebSocket support.

Installation without SSH?

Only with a secure provider equivalent for Composer, Artisan, scheduling and workers, validated on that host. Uploading files does not initialize the database.

Can Cloudflare host the HR app?

Cloudflare hosts the static marketing/docs sites. This Laravel app still needs a PHP server and database.

A documented feature is missing from my menu

The manual covers optional modules and different roles. Ask the administrator to check enablement/permissions. Documentation does not grant access.

Support, licences & credits

Asking for help

Use the support channel supplied with your purchase or final listing. Include version, platform, PHP/database versions, sanitized errors, reproduction steps and the failing guide step. Do not send credentials, employee records or production databases.

Installation, customization, infrastructure and mobile publication services are scoped according to purchased terms. This guide does not promise a support period or service entitlement.

Commercial distribution

Open Core HR is a commercial self-hosted product supplied with source code. Customer terms are being finalized. Use the licence included with your purchase and applicable marketplace terms. Third-party components retain their own licences.

Credits

The product uses Laravel, React, Inertia, Flutter and dependencies in its Composer/npm/pub lockfiles. Retain their supplied notices. The dependency list is not a completed redistribution audit; release acceptance must confirm included assets and rights.

Documentation tooling uses [Markdown-it](#), [YAML](#), [Playwright](#) and [Cloudflare Workers](#). The locally served Inter font includes its licence.