



# Coolify Project Structure

The open-source self-hosting alternative to Vercel/Netlify. A massive Laravel application using Livewire for UI and Docker for orchestration.

Updated 2025-12-30

#coolify #laravel #livewire #php #docker #self-hosting #open-source

PNG

PDF

Copy

Prompt

## Project Directory

### coolify/

- app/ Core applicatio...
  - Actions/ Domain logic & ...
    - Application/
    - Server/
    - Service/
  - Jobs/ Async backgroun...
    - ApplicationDeploymentJob.php
    - ServerCheckJob.php
  - Livewire/ Reactive UI con...
    - Project/
    - Server/
    - Dashboard.php
  - Models/ Eloquent ORM de...
  - Http/ Controllers & M...
- resources/ Frontend assets...
  - views/
    - livewire/ UI templates
    - components/ Blade components
  - js/ Vue & Alpine sc...
  - css/ Tailwind styles
- database/ Schema & Migrat...
  - migrations/ Hundreds of sch...
  - seeders/ Initial data se...
- templates/ One-click servi...
  - compose/ Docker Compose ...
  - service-templates.json
- docker/ Internal Docker...
  - coolify-helper/
  - coolify-realtime/
- scripts/ Install & upgra...
  - install.sh
  - upgrade.sh
  - composer.json PHP dependencies
  - package.json JS dependencies
  - docker-compose.yml Self-hosting co...

## Repository Info

Repository - [coollabsio/coolify](#)

Stars - 35k+

License - Apache-2.0

Last Analyzed - December 2025

## Tech Stack

Framework - Laravel 11

UI Library - Livewire + Alpine.js

Database - SQLite (Default) / Postgres

Orchestration - Docker

Styling - Tailwind CSS

Realtime - Laravel Echo + Pusher

## Architecture Notes

Coolify is a 'Modular Monolith' built on Laravel. Instead of a separate frontend SPA, it uses Livewire to deliver a reactive, single-page-like experience directly from the server. This simplifies the architecture significantly. The core logic relies heavily on asynchronous **Jobs** to handle long-running tasks like server provisioning and Docker deployments.

## Key Directories

**app/Actions/** - Contains isolated business logic (e.g., `StopApplication`), keeping controllers thin.

**app/Jobs/** - The engine room. Handles all async tasks like deploying containers, checking server health, and backups.

**templates/compose/** - A massive collection of Docker Compose files that power the 'One-Click' service marketplace.

**app/Livewire/** - Frontend controllers. Each file here corresponds to a reactive UI component or page.

## Why This Structure?

This is a masterclass in modern PHP development. It shows how to build a complex, interactive system without the complexity of a separate React/Vue frontend API. The use of **Actions** for business logic and **Jobs** for orchestration makes the codebase highly readable and maintainable despite its size.