Introduction

Docker with PHP and Caddy for Deployer

Anonymous · 09/09/2026 · #docker

Docker with PHP and Caddy for Deployer

A modern Docker setup for hosting PHP websites with Caddy web server and SSH access for Deployer deployments.

Current PHP version: 8.5.

The latest supported, and recommended, PHP version is 8.5. If you need to use another version, please use the docker-php7.4-caddy-server, docker-php8.3-caddy-server or docker-php8.4-caddy-server repositories. Remember change the PHP version in the examples below.

Features

- **Multiple PHP versions**: Supported PHP versions with common extensions: 7.4, 8.3, 8.4 and 8.5.
- **Caddy**: Modern web server with automatic HTTPS.
- **SSH Access**: For automated deployments with Deployer.
- **Automatic Site Discovery**: Just add your site folder and it works.
- **Development Domains**: Test with .local domains that map to production folders.
- **Automatic WWW Redirection**: For second-level domains (e.g., example.com → www.example.com).
- **Auto-HTTPS**: Certificates are automatically generated on-demand.
- **Environment Separation**: Development and production environments can be managed through Docker Compose override.

Quick Start

Prerequisites

- Docker and Docker Compose installed on your system.
- SSH key for deployment access.

Setup

1. Clone this repository:

   ```
   git clone https://github.com/derafu/docker-php8.5-caddy-server.git
   cd docker-php8.5-caddy-server
   ```

2. Add your SSH public key to `config/ssh/authorized_keys` for admin, and default deployment, access:

   ```
   cat ~/.ssh/id_rsa.pub > config/ssh/authorized_keys
   ```

3. Build and start the container:

   ```
   docker-compose up -d
   ```

   The `-d` parameter runs it in detached mode (background).

Verification

Check that the container is running:

```
docker-compose ps
```

View container logs:

```
docker-compose logs -f
```

The `-f` parameter allows you to follow logs in real-time.

Testing Your First Site

1. Create a test site directory structure:

   ```
   mkdir -p sites/www.example.com/public
   echo "<?php phpinfo();" > sites/www.example.com/public/index.php
   ```

2. Access the site at:

   - Production mode: https://www.example.com (requires DNS configuration).

    - Development mode: https://www.example.com.local:8443 (requires local hosts entry).

For local development, add to your `/etc/hosts` file:

```
127.0.0.1 www.example.com.local
```

Directory Structure

```
docker-php8.5-caddy-server/
├── config/                              # Configuration files.
│   ├── caddy/                           # Caddy configuration.
│   ├── php/                             # PHP configuration.
│   ├── ssh/                             # SSH configuration and authorized keys.
│   └── supervisor/                      # Supervisor configuration.
├── sites/                               # Web sites directory for local development.
│   └── www.example.com/                 # Example site.
│       └── public/                      # Public web files.
├── .env                                 # Docker Compose environment configuration.
├── Dockerfile                           # Container definition.
├── docker-compose.yml                   # Docker services configuration (production).
└── docker-compose.override-example.yml # Development-specific configuration.
```

Development vs Production Environment

This project uses Docker Compose's override functionality to separate different configurations.

Usage:

- **Development**: Rename `docker-compose.override-example.yml` to `docker-compose.override.yml` and then Docker Compose automatically merges both files:

  ```
  docker-compose up -d
  ```

- **Production**: Use only the base configuration (with `-f` or not creating the `docker-compose.override.yml` file):

  ```
  docker-compose -f docker-compose.yml up -d
  ```

Access and Management

SSH Access

Connect to the container via SSH:

```
ssh admin@localhost -p 2222
```

Direct Container Access

Access the container shell:

```
docker exec -it derafu-sites-server-php-caddy bash
```

Restarting Services

Restart Caddy web server:

```
docker exec -it derafu-sites-server-php-caddy supervisorctl restart caddy
```

Stopping the Container

```
docker-compose down
```

Rebuilding After Configuration Changes

Rebuild for development:

```
docker-compose build --no-cache
docker-compose up -d
```

Rebuild for production:

```
docker-compose -f docker-compose.yml build --no-cache
docker-compose -f docker-compose.yml up -d
```

Adding New Sites

1. Create the site directory structure:

```
mkdir -p sites/www.newsite.com/public
touch sites/www.newsite.com/public/index.php
```

2. No server restart required! Caddy automatically detects new sites.

3. For local development, add to your hosts file:

```
127.0.0.1 www.newsite.com.local
```

Environment Variables

Customize behavior through environment variables:

Variable	Description	Default
`SERVER_NAME`	Name for the docker container	derafu-sites-server
`CADDY_DEBUG`	Enable debug mode with `debug`	(empty)
`CADDY_EMAIL`	Email for Let's Encrypt	admin@example.com
`CADDY_HTTPS_ISSUER`	TLS issuer (internal, acme)	internal
`CADDY_HTTPS_ALLOW_ANY_HOST`	Allow any host for TLS	false
`CADDY_LOG_SIZE`	Log file max size	100mb
`CADDY_LOG_KEEP`	Number of log files to keep	5
`WWW_ROOT_PATH`	Web root path	/var/www/sites
`WWW_USER`	WWW and SSH user in the container	admin
`WWW_GROUP`	WWW group in the container	www-data
`HTTP_PORT`	HTTP port in host	8080
`HTTPS_PORT`	HTTPS port in host	8443
`SSH_PORT`	SSH port in host	2222

Domain Logic

The server handles domains in the following way:

1. **Development domains**: Any domain ending with `.local` (e.g., `www.example.com.local`)
    - Maps to the same directory as its production counterpart.

    - Uses internal self-signed certificates.

2. **Production domains**:
    - Redirects from non-www to www for second-level domains.
    - Automatically obtains and manages Let's Encrypt certificates (issuer `acme`).

Troubleshooting

SSL Certificate Issues

If you're having issues with SSL certificates in development:

- Ensure your browser trusts self-signed certificates.
- Try using HTTP instead of HTTPS for local development.

Permissions Issues

If you encounter permission issues:

```
docker exec -it derafu-sites-server-php-caddy chown -R admin:www-data /var/www/sites
```

Logs Location

Logs are available in the container and can be accessed with:

```
docker exec -it derafu-sites-server-php-caddy cat /var/log/caddy/access.log
```

Advanced Usage

Custom Caddy Configuration

For advanced configurations, modify the Caddyfile at `config/caddy/Caddyfile`.

Using with Deployer

This container is designed to work with Deployer for PHP deployments:

1. Set up your `deploy.php` configuration to connect to the SSH server on port 2222.
2. Use the `admin` user for authentication.
3. Set your deployment path to `/var/www/sites/www.yoursite.com`

Last updated on 09/09/2026