Website Project

Base for Derafu's Websites

Anonymous · 21/08/2026 · #php

Base for Derafu’s Websites

This guide explains how to create a new website using the Derafu Website Base, the Docker and PHP Deployer projects for local development and deployment to the production server.

WIP: Work in Progress

The Road to Success is Always Under Construction.

Quick Start

Create a new website based on this base:

```
composer create-project derafu/website www.example.com --stability=dev
```

Review what you need to do, install, or know

Configure ZSH on local macOS machine

If you’re on macOS and use ZSH as your shell, you can configure ZSH to accept comments in commands with:

```
echo "setopt interactivecomments" >> ~/.zshrc
source ~/.zshrc
```

SSH Key on local machine

It’s necessary to have an SSH key on your local machine. This key will be used to configure your Docker container.

With the following command, you’ll get an existing key or create a new one in `$HOME/.ssh/id_rsa` and `$HOME/.ssh/id_rsa.pub`. If a new one is created, you’ll need to enter your email to identify the key owner. If you prefer, you can enter your username and machine name (in case you have multiple SSH

keys in different environments).

```
SSH_KEY="$HOME/.ssh/id_rsa"
if [ -f "$SSH_KEY.pub" ]; then
    cat "$SSH_KEY.pub"
else
    echo -n "Enter your email: "; read COMMENT
    ssh-keygen -t rsa -b 4096 -N "" -C "$COMMENT" -f "$SSH_KEY"
    cat "$SSH_KEY.pub"
fi
```

With this command, the public key will be displayed on screen. If you need to see it again in the future, run:

```
cat $HOME/.ssh/id_rsa.pub
```

Important: The key in `$HOME/.ssh/id_rsa` is **private** and should never be shared.

Add SSH key to GitHub

1. Go to GitHub.
2. In `Title`, enter the same email or comment you chose when creating the key.
3. In `Key`, paste the public key extracted with `cat $HOME/.ssh/id_rsa.pub`.
4. Click on `Add SSH key`.

Docker container with PHP and Caddy

Prepare Docker in `$DOCKER_DIR` on your local machine:

```
DEV_DIR=$HOME/dev
DOCKER_DIR=$DEV_DIR/docker-sites-php
mkdir -p $DEV_DIR
git clone https://github.com/derafu/docker-php-caddy-server.git $DOCKER_DIR
cat $HOME/.ssh/id_rsa.pub > $DOCKER_DIR/config/ssh/authorized_keys
cd $DOCKER_DIR
```

Copy and edit the `.env` file and configure the environment variables as needed.

```
# It's recommended to at least configure the DEPLOYER_HOST variable.
cp .env-dist .env
```

Warning

Don’t proceed until you’ve reviewed and configured the `.env` file.

Build the Docker container:

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

With this configuration, the folder where sites to be developed will be installed will be in `$DOCKER_DIR/sites`. This folder will be shared between your local machine and the Docker container.

Connect to Docker container

Configure the `dev` SSH alias on your local machine:

```
echo "
Host dev
    HostName localhost
    User admin
    Port 2222
    IdentityFile $HOME/.ssh/id_rsa
    StrictHostKeyChecking no
    UserKnownHostsFile /dev/null
" >> $HOME/.ssh/config
```

Then you can enter the Docker container with:

```
ssh dev
```

Note: The alias name can be whatever you want; in this case, `dev` was used.

SSH key in Docker container

To work with private repositories and deploy to the production server, it's necessary that the SSH key from your local machine be added to the Docker container.

On your local machine, run:

```
scp $HOME/.ssh/id_rsa* dev:.ssh/
```

Basic Git configuration

Perform the following configurations inside the Docker container.

First, configure your GitHub email and name:

Warning

Before pasting this command in the Docker container, edit it to use your GitHub email and name.

```
git config --global user.email "you@example.com"  # Your github email.
git config --global user.name "Your Name"         # Your name.
```

Configure case sensitivity and avoid mixing changes:

```
git config --global core.ignorecase false         # Case sensitive.
git config --global pull.rebase false             # Rebase instead of merge.
git config --global merge.ff false                # Fast forward.
```

Configure text editor:

```
git config --global core.editor nano              # Default editor nano or any other.
```

Configure commit signing in Git

First, create the SSH key on your local machine:

```
SSH_KEY="$HOME/.ssh/id_ed25519"
if [ -f "$SSH_KEY.pub" ]; then
    cat "$SSH_KEY.pub"
else
    echo -n "Enter your email: "; read COMMENT
    ssh-keygen -t ed25519 -N "" -C "$COMMENT" -f "$SSH_KEY"
    cat "$SSH_KEY.pub"
fi
```

Then add the SSH key to the Docker container:

```
scp $HOME/.ssh/id_ed25519* dev:.ssh/
```

Finally, inside the Docker container, configure Git to sign commits:

```
git config --global commit.gpgSign true               # Sign commits.
git config --global user.signingkey ~/.ssh/id_ed25519 # Your ssh key.
git config --global gpg.format ssh                    # Use ssh key.
git config --global tag.gpgSign true                  # Sign tags.
```

Develop a website

Create a new website

Enter the container and run:

```
site-create www.example.com
```

Note: Creating the website requires as a subsequent step that you configure its repository on GitHub and add the site to the `$DEPLOYER_DIR/sites.php` file using the `site-add` command.

Clone a website

Enter the container and run:

```
site-clone www.example.com git@github.com:example/example.git
```

Note: When cloning an existing website, the site is automatically added to the `$DEPLOYER_DIR/sites.php` file.

Visit the website in the browser

On your machine, configure `/etc/hosts` by adding the local development domain:

```
echo "127.0.0.1         www.example.com.local" | sudo tee -a /etc/hosts
```

Then you can access the website via the URL https://www.example.com.local:8443

Deploy a website to production

All these instructions are executed in the Docker container.

Add website to configuration file

If you created the site from scratch instead of cloning it, make sure the website is added to the `$DEPLOYER_DIR/sites.php` file. You can validate this by running:

```
site-add www.example.com git@github.com:example/example.git
```

Note: If the site requires special configuration, you'll need to manually edit the `$DEPLOYER_DIR/sites.php` file.

Style tests, code quality, and unit tests

Run style tests, code quality, and unit tests in the Docker container with:

```
site www.example.com
site-check
```

Push changes to GitHub

If everything is correct, push the changes to GitHub:

```
site www.example.com
site-send "Website update."
```

Note: If the GitHub repository has a configured webhook, the website will be deployed automatically when pushing changes and passing the style tests, code quality, and unit tests in the GitHub Actions workflow.

Deployment

Note

It's not necessary to deploy to the production server if the GitHub repository has a configured webhook.

If there are no errors, you can deploy to the production server with:

```
#DEPLOYER_HOST=hosting.example.com # Only if not configured in .env
site-deploy www.example.com
```

If an error occurs when deploying and you try to make a new deploy, it's very likely that the deploy is locked. If this happens, you can unlock and deploy again with:

```
#DEPLOYER_HOST=hosting.example.com # Only if not configured in .env
site-deploy-locked www.example.com
```

Update components

Update Docker

On your local machine, run:

```
DEV_DIR=$HOME/dev
DOCKER_DIR=$DEV_DIR/docker-sites-php
cd $DOCKER_DIR
git pull
docker-compose build --no-cache
docker-compose up -d
```

Warning

You must add the SSH keys to the Docker container again and configure Git inside the container.

Update PHP Deployer

Enter the container and run:

```
cd $DEPLOYER_DIR
git pull
composer update
```

Update website

Enter the container and run:

```
site www.example.com
site-update
```

Using development tools

Composer

Install dependencies:

```
composer install
```

Add development dependency:

```
composer require --dev dependency-name
```

Add production dependency:

```
composer require dependency-name
```

Remove dependency:

```
composer remove dependency-name
```

Update dependencies:

```
composer update
```

Note: Normally only `composer install` is used to install dependencies.

NPM

Install dependencies:

```
npm install
```

Add development dependency:

```
npm install --save-dev dependency-name
```

Add production dependency:

```
npm install --save dependency-name
```

Remove dependency:

```
npm uninstall dependency-name
```

Update dependencies:

```
npm update
```

Note: Normally only `npm install` is used to install dependencies.

Git

View change status:

```
git status
```

Add changes:

```
git add .
```

Make commit:

```
git commit -m "Commit message"
```

Push changes to GitHub:

```
git push
```

Update local repository:

```
git pull
```

Undo changes:

```
git checkout -- .
```

Note: Instead of using dot `.`, you can specify the files you want to add, commit, or revert.

PHP CS Fixer

Check code style:

```
composer phpcs
```

Fix code style:

```
composer phpcs-fix
```

PHP Unit

Run unit tests:

```
composer tests
```

Using terminal in Docker container

Enter a website directory:

```
cd $SITES_DIR/www.example.com
```

Exit directory:

```
cd ..
```

List files:

```
ls -la
```

View file content:

```
cat $DEPLOYER_DIR/sites.php
```

Edit a file:

```
nano $DEPLOYER_DIR/sites.php
```

Save and exit in `nano`:

```
Ctrl + X
```

Last updated on 21/08/2026