

Configuration

Configuration

Anonymous · 10/10/2026

Configuration

The configuration can be stored in multiple files:

- `sites.php`: Legacy file and currently **deprecated**.
- `config/sites.yaml`: New file for the sites configuration when you need only one environment.
- `config/ABC.sites.yaml`: File for separated environments sites configuration.

The `ABC` is called the *source* of the configuration. For example, `dev.sites.yaml` for the development environment, `prod.sites.yaml` for the production environment, etc.

Sites Configuration

Create a YAML file and configure the sites you want to deploy using the following structure in each file:

- A key with the site/domain name and a value with the repository URL.
- A key with the site/domain name and a value with an array of detailed configuration options.

For example, create the file `config/sites.yaml` with the following content:

```
# Simple configuration with just the repository URL.
www.example.com: git@github.com:example/example.git

# Extended configuration with options.
www.complex-site.com:
  repository: git@github.com:example/complex-site.git
  branch: dev
  deploy_path: /var/www/custom/path/complex-site
  shared_files: ['.env', 'config/settings.php']
  shared_dirs: ['var/uploads', 'var/logs']
  writable_dirs: ['var', 'tmp', 'var/cache']
  writable_mode: chmod
  writable_use_sudo: false
  writable_recursive: true
  writable_chmod_mode: 0775
```

Add a site to the configuration via CLI

For simple configurations you can use:

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

This will always add the configuration to the `config/sites.yaml` file.

Using sources

You can use sources to create a configuration for multiple environments. For example, you can create a source for the development environment and a source for the production environment.

- `config/dev.sites.yaml`: Development environment configuration.
- `config/prod.sites.yaml`: Production environment configuration.

This file can have the same sites, but with different configuration options. Then you can use the `--source` option to specify the source of the configuration to select the appropriate environment for the site.

Available Configuration Options

Option	Description	Default
repository	Git repository URL	<i>Required</i>
branch	Git branch to deploy	main
deploy_path	Deployment path on server	<code>/var/www/sites/[site-name]</code>
shared_files	Files to share between releases	<code>[]</code>
shared_dirs	Directories to share between releases	<code>[]</code>
writable_dirs	Directories to make writable	<code>['var', 'tmp']</code>
writable_mode	Mode for writable directories	chmod
writable_use_sudo	Whether to use sudo for writable directories	false
writable_recursive	Apply writable permissions recursively	true
writable_chmod_mode	Chmod mode for writable directories	0777

Deployer Server Configuration

The server configuration is defined in the file `deploy.php`. By default, a local environment (localhost) and a remote environment (if `DEPLOYER_HOST` is set) are configured:

```
// Default local environment.
host('localhost')
  ->setRemoteUser('admin')
  ->setPort(2222)
  ->setLabels(['stage' => 'local']);

// Remote environment (only if DEPLOYER_HOST is set).
if (getenv('DEPLOYER_HOST')) {
  $stage = getenv('DEPLOYER_STAGE') ?: 'prod';
  host(getenv('DEPLOYER_HOST'))
    ->setRemoteUser(getenv('DEPLOYER_USER') ?: 'admin')
    ->setPort(getenv('DEPLOYER_PORT') ?: 2222)
    ->setLabels(['stage' => $stage]);
  set('default_selector', 'stage=' . $stage);
}
```

You can modify these settings or add additional environments as needed.

We recommend to use the environment variables and not to hardcode the values in the file `deploy.php`.

Variable	Description	Default
DEPLOYER_HOST	Remote host	
DEPLOYER_USER	Remote user	admin
DEPLOYER_PORT	Remote port	2222
DEPLOYER_STAGE	Environment stage	prod

Actions (shell scripts)

Actions allow you to run custom commands during the deployment. For example, you can run a custom command to rebuild the cache after the deployment.

There are three types of actions:

- `initial`: Run initial actions before the deployment, just after the code is updated.
- `final`: Run final actions after the deployment, just before the symlink is created.
- `success`: Run success actions after the deployment, just before the success message.

The actions are defined in the `.deployer/actions` directory. Each action is a Shell script file:

- `.deployer/actions/initial.sh`: Initial actions script.
- `.deployer/actions/final.sh`: Final actions script.
- `.deployer/actions/success.sh`: Success actions script.

All the actions are optional, the deployment will continue if the action file is not present.

Last updated on 10/10/2026