

Usage and Deployment

Usage and Deployment

Anonymous · 10/10/2026

Usage and Deployment

Usage

List Available Sites

To see the list of configured sites and usage information use `derafu:sites:list` command.

```
vendor/bin/dep derafu:sites:list
```

You can filter the list of sites by source using the `--source` option.

```
vendor/bin/dep derafu:sites:list --source=prod
```

Deploy a Single Site

Local Environment (localhost)

```
vendor/bin/dep derafu:deploy:single --site=www.example.com
```

Remote Environment

This usually is used for production environments, but can be used for any remote environment.

```
DEPLOYER_HOST=hosting.example.com vendor/bin/dep derafu:deploy:single --  
site=www.example.com
```

You can also specify the SSH user and port with the environment variables `DEPLOYER_USER` and `DEPLOYER_PORT`.

```
DEPLOYER_HOST=hosting.example.com DEPLOYER_USER=deployuser DEPLOYER_PORT=22  
vendor/bin/dep derafu:deploy:single --site=www.example.com
```

You can also specify the source of the configuration to select the appropriate environment for the site.

```
DEPLOYER_HOST=hosting.example.com DEPLOYER_USER=deployuser DEPLOYER_PORT=22  
vendor/bin/dep derafu:deploy:single --source=prod --site=www.example.com
```

Deploy All Sites

You can deploy all sites of the configuration in deployer, or only the sites of a specific source. It's similar to the deploy a single site, but using `derafu:deploy:all`.

Local Environment (localhost)

```
vendor/bin/dep derafu:deploy:all
```

For a specific source:

```
vendor/bin/dep derafu:deploy:all --source=prod
```

Remote Environment

```
DEPLOYER_HOST=hosting.example.com vendor/bin/dep derafu:deploy:all
```

For a specific source:

```
DEPLOYER_HOST=hosting.example.com vendor/bin/dep derafu:deploy:all --source=prod
```

And you can also specify the SSH user and port with the environment variables `DEPLOYER_USER` and `DEPLOYER_PORT`.

Unlock a Deployment

If a deployment gets stuck or locked, you can unlock it `--unlock` option.

```
DEPLOYER_HOST=hosting.example.com vendor/bin/dep derafu:deploy:single --  
site=www.example.com --unlock
```

Using aliases

You can create aliases for easy deployment. For example:

```
DEPLOYER_DIR="$HOME/dev/php/deployer"  
alias site-deploy-stage="cd $DEPLOYER_DIR && DEPLOYER_HOST=example.com  
DEPLOYER_USER=admin DEPLOYER_PORT=2223 vendor/bin/dep derafu:deploy:single --site"
```

```
alias site-deploy-prod="cd $DEPLOYER_DIR && DEPLOYER_HOST=example.com
DEPLOYER_USER=admin DEPLOYER_PORT=2224 vendor/bin/dep derafu:deploy:single --site"
```

Also, you can specify the source of the configuration to select the appropriate environment for the site.

```
DEPLOYER_DIR="$HOME/dev/php/depoyer"
alias site-deploy-stage="cd $DEPLOYER_DIR && DEPLOYER_HOST=example.com
DEPLOYER_USER=admin DEPLOYER_PORT=2223 vendor/bin/dep derafu:deploy:single --
source=dev --site"
alias site-deploy-prod="cd $DEPLOYER_DIR && DEPLOYER_HOST=example.com
DEPLOYER_USER=admin DEPLOYER_PORT=2224 vendor/bin/dep derafu:deploy:single --
source=prod --site"
```

Then you can use the aliases to deploy the site to the appropriate environment.

```
site-deploy-stage www.example.com
site-deploy-prod www.example.com
```

Deployment Process

For each site, the deployment process performs the following steps:

1. **Info** (deploy:info): Shows information about the deployment.
2. **Setup** (deploy:setup): Sets up the deployment environment.
3. **Check Remote** (deploy:check_remote): Verifies SSH connection and deployment path.
4. **Lock** (deploy:lock): Locks the deployment.
5. **Release** (deploy:release): Creates a new release.
6. **Update Code** (deploy:update_code): Fetches code from the Git repository.
7. **Initial Actions** (deploy:initial_actions): Runs custom commands in the initial of the deployment.
8. **Environment** (deploy:env): Sets up the environment variables.
9. **Shared Files/Dirs** (deploy:shared): Links shared files and directories.
10. **Writable Dirs** (deploy:writable): Makes specified directories writable.
11. **Vendors** (deploy:vendors): Installs PHP dependencies with Composer.
12. **Assets** (deploy:assets): Builds frontend assets if package.json exists (npm install && npm run build).
13. **Final Actions** (deploy:final_actions): Runs custom commands in the final of the deployment.
14. **Symlink** (deploy:symlink): Creates a symlink to the new release.
15. **Unlock** (deploy:unlock): Removes the deployment lock.
16. **Cleanup** (deploy:cleanup): Removes old releases (keeps 5 by default).
17. **OPcache Reset** (opcache:reset): Resets the OPcache.
18. **Success Actions** (deploy:success_actions): Runs custom commands in the success of the deployment.
19. **Success Message** (deploy:success): Shows the success message.

Last updated on 10/10/2026