
PHPloy

Version 4.9.2

PHPloy is an incremental Git FTP and SFTP deployment tool. By keeping track of the state of the remote server(s) it deploys only the files that were committed since the last deployment. PHPloy supports submodules, sub-submodules, deploying to multiple servers and rollbacks. PHPloy requires **PHP 7.3+** and **Git 1.8+**.

How it works

PHPloy stores a file called `.revision` on your server. This file contains the hash of the commit that you have deployed to that server. When you run `phploy`, it downloads that file and compares the commit reference in it with the commit you are trying to deploy to find out which files to upload. PHPloy also stores a `.revision` file for each submodule in your repository.

Install

Via Composer

If you have composer installed in your machine, you can pull PHPloy globally like this:

```
1 composer global require "banago/phploy"
```

Make sure to place the `$HOME/.composer/vendor/bin` directory (or the equivalent directory for your OS) in your `$PATH` so the PHPloy executable can be located by your system.

Via Phar Archive

You can install PHPloy Phar globally, in your `/usr/local/bin` directory or, locally, in your project directory. **Rename** `phploy.phar` to `phploy` for ease of use.

1. **Globally:** Move `phploy` into `/usr/local/bin`. Make it executable by running `sudo chmod +x phploy`.
2. **Locally** Move `phploy` into your project directory.

Usage

When using PHPloy locally, proceed the command with `php`

-
1. Run `phploy --init` in the terminal to create the `phploy.ini` file or create one manually.
 2. Run `phploy` in terminal to deploy.

Windows Users: Installing PHPloy globally on Windows

phploy.ini

The `phploy.ini` file holds your project configuration. It should be located in the root directory of the project. `phploy.ini` is never uploaded to server. Check the sample below for all available options:

```
1 ; This is a sample deploy.ini file. You can specify as many
2 ; servers as you need and use normal or quickmode configuration.
3 ;
4 ; NOTE: If a value in the .ini file contains any non-alphanumeric
5 ; characters it needs to be enclosed in double-quotes (").
6
7 [staging]
8     scheme = sftp
9     user = example
10    ; When connecting via SFTP, you can opt for password-based
11    authentication:
12    pass = password
13    ; Or private key-based authentication:
14    privkey = 'path/to/or/contents/of/privatekey'
15    host = staging-example.com
16    path = /path/to/installation
17    port = 22
18    ; You can specify a branch to deploy from
19    branch = develop
20    ; File permission set on the uploaded files/directories
21    permissions = 0700
22    ; File permissions set on newly created directories
23    directoryPerm = 0775
24    ; Deploy only this directory as base directory
25    base = 'directory-name/'
26    ; Files that should be ignored and not uploaded to your server, but
27    still tracked in your repository
28    exclude[] = 'src/*.scss'
29    exclude[] = '*.ini'
30    ; Files that are ignored by Git, but you want to send the the
31    server
32    include[] = 'js/scripts.min.js'
33    include[] = 'directory-name/'
34    ; conditional include - if source file has changed, include file
35    include[] = 'css/style.min.css:src/style.css'
36    ; Directories that should be copied after deploy, from->to
37    copy[] = 'public->www'
```

```
35 ; Directories that should be purged before deploy
36 purge-before[] = "dist/"
37 ; Directories that should be purged after deploy
38 purge[] = "cache/"
39 ; Pre- and Post-deploy hooks
40 ; Use "DQUOTE" inside your double-quoted strings to insert a
    literal double quote
41 ; Use 'QUOTE' inside your quoted strings to insert a literal quote
42 ; For example pre-deploy[] = 'echo "that'QUOTE's nice"' to get a
    literal "that's".
43 ; That workaround is based on http://php.net/manual/de/function.
    parse-ini-file.php#70847
44 pre-deploy[] = "wget http://staging-example.com/pre-deploy/test.php
    --spider --quiet"
45 post-deploy[] = "wget http://staging-example.com/post-deploy/test.
    php --spider --quiet"
46 ; Works only via SSH2 connection
47 pre-deploy-remote[] = "touch .maintenance"
48 post-deploy-remote[] = "mv cache cache2"
49 post-deploy-remote[] = "rm .maintenance"
50 ; You can specify a timeout for the underlying connection which
    might be useful for long running remote
51 ; operations (cache clear, dependency update, etc.)
52 timeout = 60
53
54 [production]
55 quickmode = ftp://example:password@production-example.com:21/path/
    to/installation
56 passive = true
57 ssl = false
58 ; You can specify a branch to deploy from
59 branch = master
60 ; File permission set on the uploaded files/directories
61 permissions = 0774
62 ; File permissions set on newly created directories
63 directoryPerm = 0755
64 ; Files that should be ignored and not uploaded to your server, but
    still tracked in your repository
65 exclude[] = 'libs/*'
66 exclude[] = 'config/*'
67 exclude[] = 'src/*.scss'
68 ; Files that are ignored by Git, but you want to send the the
    server
69 include[] = 'js/scripts.min.js'
70 include[] = 'js/style.min.css'
71 include[] = 'directory-name/'
72 purge-before[] = "dist/"
73 purge[] = "cache/"
74 pre-deploy[] = "wget http://staging-example.com/pre-deploy/test.php
    --spider --quiet"
75 post-deploy[] = "wget http://staging-example.com/post-deploy/test.
```

```
php --spider --quiet"
```

If your password is missing in the `phploy.ini` file or the `PHPLOY_PASS` environment variable, PHPloy will interactively ask you for your password. There is also an option to store the user and password in a file called `.phploy`.

```
1 [staging]
2   user="theUser"
3   pass="thePassword"
4
5 [production]
6   user="theUser"
7   pass="thePassword"
```

This feature is especially useful if you would like to share your `phploy.ini` via Git but hide your password from the public.

You can also use environment variables to deploy without storing your credentials in a file. These variables will be used if they do not exist in the `phploy.ini` file:

```
1 PHPLOY_HOST
2 PHPLOY_PORT
3 PHPLOY_PASS
4 PHPLOY_PATH
5 PHPLOY_USER
6 PHPLOY_PRIVKEY
```

These variables can be used like this;

```
1 $ PHPLOY_PORT="21" PHPLOY_HOST="myftphost.com" PHPLOY_USER="ftp"
   PHPLOY_PASS="ftp-password" PHPLOY_PATH="/home/user/public_html/
   example.com" phploy -s servername
```

Or export them like this, the script will automatically use them:

```
1 $ export PHPLOY_PORT="21"
2 $ export PHPLOY_HOST="myftphost.com"
3 $ export PHPLOY_USER="ftp"
4 $ export PHPLOY_PASS="ftp-password"
5 $ export PHPLOY_PATH="/home/user/public_html/example.com"
6 $ export PHPLOY_PRIVKEY="path/to/or/contents/of/privatekey"
7 $ phploy -s servername
```

Multiple servers

PHPloy allows you to configure multiple servers in the deploy file and deploy to any of them with ease.

By default PHPloy will deploy to *ALL* specified servers. Alternatively, if an entry named 'default' exists in your server configuration, PHPloy will default to that server configuration. To specify one single server, run:

```
1 phploy -s servername
```

or:

```
1 phploy --server servername
```

`servername` stands for the name you have given to the server in the `phploy.ini` configuration file.

If you have a 'default' server configured, you can specify to deploy to all configured servers by running:

```
1 phploy --all
```

Shared configuration (custom defaults)

If you specify a server configuration named `*`, all options configured in this section will be shared with other servers. This basically allows you to inject custom default values.

```
1 ; The special '*' configuration is shared between all other
   configurations (think include)
2 [*]
3     exclude[] = 'src/*'
4     include[] = "dist/app.css"
5
6 ; As a result both shard1 and shard2 will have the same exclude[] and
   include[] "default" values
7 [shard1]
8     quickmode = ftp://example:password@shard1-example.com:21/path/to/
   installation
9
10 [shard2]
11     quickmode = ftp://example:password@shard2-example.com:21/path/to/
   installation
```

Rollbacks

Warning: the `--rollback` option does not currently update your submodules correctly.

PHPloy allows you to roll back to an earlier version when you need to. Rolling back is very easy.

To roll back to the previous commit, you just run:

```
1 phploy --rollback
```

To roll back to whatever commit you want, you run:

```
1 phploy --rollback commit-hash-goes-here
```

When you run a rollback, the files in your working copy will revert **temporarily** to the version of the rollback you are deploying. When the deployment has finished, everything will go back as it was.

Note that there is not a short version of `--rollback`.

Listing changed files

PHPloy allows you to see what files are going to be uploaded/deleted before you actually push them. Just run:

```
1 phploy -l
```

Or:

```
1 phploy --list
```

Updating or “syncing” the remote revision

If you want to update the `.revision` file on the server to match your current local revision, run:

```
1 phploy --sync
```

If you want to set it to a previous commit revision, just specify the revision like this:

```
1 phploy --sync your-revision-hash-here
```

Creating deployment directory on first deploy

If the deployment directory does not exist, you can instruct PHPloy to create it for you:

```
1 phploy --force
```

Manual fresh upload

If you want to do a fresh upload, even if you have deployed earlier, use the `--fresh` argument like this:

```
1 phploy --fresh
```

Submodules

Submodules are supported, but are turned off by default since you don't expect them to change very often and you only update them once in a while. To run a deployment with submodule scanning, add the `--submodules` parameter to the command:

```
1 phploy --submodules
```

Purging

In many cases, we need to purge the contents of a directory after a deployment. This can be achieved by specifying the directories in `phploy.ini` like this:

```
1 ; relative to the deployment path
2 purge[] = "cache/"
```

To purge a directory before deployment, specify the directories in `phploy.ini` like this:

```
1 ; relative to the deployment path
2 purge-before[] = "dist/"
```

Hooks

PHPloy allows you to execute commands before and after the deployment. For example you can use `wget` call a script on my server to execute a `composer update`.

```
1 ; To execute before deployment
2 pre-deploy[] = "wget http://staging-example.com/pre-deploy/test.php --
  spider --quiet"
3 ; To execute after deployment
4 post-deploy[] = "wget http://staging-example.com/post-deploy/test.php
  --spider --quiet"
```

Logging

PHPloy supports simple logging of the activity. Logging is saved in a `phploy.log` file in your project in the following format:

```
1 2016-03-28 08:12:37+02:00 --- INFO: [SHA: 59
    a387c26641f731df6f0d1098aaa86cd55f4382] Deployment to server: "
    default" from branch "master". 2 files uploaded; 0 files deleted.
```

To turn logging on, add this to `phploy.ini`:

```
1 [production]
2     logger = on
```

Contribute

Contributions are very welcome; PHPloy is great because of the contributors. Please check out the issues.

Credits

- Baki Goxhaj
- Contributors

Version history

Please check release history for details.

License

PHPloy is licensed under the MIT License (MIT).