PHP: Getting started

How to start using PHP

> 👋**Welcome on Stackhero's documentation!**
> Stackhero provides PHP instances that are ready for production in just 2 minutes!
> Including TLS encryption (aka HTTPS), customizable domain name, deploys with a simple git push, backups and updates in just a click.
> Try our managed PHP cloud in just 2 minutes

PHP on Stackhero is super easy to deploy and very powerful.
You will find here all the resources to deploy your app in seconds.

Start a PHP service

First, you have to create a PHP service on Stackhero.

Prerequisites

You will need some classical tools to deploy on Stackhero:

1. Git, that you will find here: https://git-scm.com/download/.
2. For windows users, we strongly recommend to use "Windows terminal" as terminal. You will find it in Microsoft Store.

Configure your service

In service's configuration, the only one thing to configure is the public key.

You will find yours with the command `cat ~/.ssh/id_rsa.pub` or `cat ~/.ssh/id_ed25519.pub`.

If you don't have a set of keys yet, you can create them using `ssh-keygen` on Linux and macOS or using `ssh-keygen.exe` on Windows.

Once you have your public key, go to the Stackhero dashboard, select your PHP service, go to its configuration and copy your key there.

> Tip: you can define globally your SSH public key so future services will get it automatically in their configuration.
> To do that, go to Stackhero dashboard, click on your profile picture (on the top right corner) then go to "Your profile" and paste your SSH public key.

Clone the example

We have prepared a simple PHP application to show you how it works on Stackhero.

Simply clone it with the following commands:

```
git clone https://github.com/stackhero-io/phpGettingStarted.git stackhero-php-getting-started
cd stackhero-php-getting-started
```

Configure the remote repository server

With Stackhero, you can deploy your application using git. It's super easy, fast and reliable.
To do this, simply copy/paste the "git remote command" you will find on the first page of your service.

It's a command like this one: `git remote add stackhero ssh://stackhero@XXXXX.stackhero-network.com:222/project.git`

Deploy your PHP code

The final step is to deploy your app.

Just push your code to Stackhero with this command: `git push stackhero master`
The first time, you will be prompt to add the key fingerprint. Just reply "yes".

Wait a few seconds and your application should be online!

Check it directly online by going to the "website" URL you will find on Stackhero dashboard (something like `https://XXXXX.stackhero-network.com`).

That's it, your application is deployed!

You can then modify the `www/index.php` file and redeploy your code:

```
git add -A .
git commit -m "Update www/index.php"
git push stackhero master
```

Deploy an existing application

To deploy an existing application, add the remote repository to it (see Configure the remote repository server above).

Then, every time you want to deploy your app, just push it to your instance: `git push stackhero master`.

Note that by default the public directory is "www". This is the directory where your PHP code will be executed (where you will have your index.php file) and where all your static files will be stored. For example, if you are going to yourdomain.com/myphoto.jpg, the file "myphoto.jpg" will be retrieved from "www/myphoto.jpg".
You can change this directory in your PHP service configuration.

Error "failed to push some refs to '[...]'"

Maybe you will get this kind of error when deploying your application:

```
error: failed to push some refs to '[...]'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

It means that the git repository on your instance isn't synchronized with your local data (it contains some files that you do not have locally).

Do force the push anyway, you can use this command: `git push -f stackhero master`

Deploy another branch than master

If you want to deploy another branch than master, let's say the branch "production", you can use this command: `git push stackhero production:master`

Deploy a tag

If you use tags and want to deploy the tag "v1.0" for example, you can use this command: `git push stackhero 'v1.0^{}:master'`

`^{}` is here to allow the tag (actually the last commit corresponding before the tag) to be pushed.

Revert or deploy a specific commit

To deploy a specific commit, get its hash with `git log`.
Then simply push it like this: `git push -f stackhero <HASH>:master`

Deploy to multiple environments

You can create multiple PHP services for different environments. For example, one for production and another one for staging.

You can rename the current remote named `stackhero` to `stackhero-staging` with this command:
`git remote rename stackhero stackhero-staging`

You can then start a new PHP service in your dashboard and add it as `stackhero-production`:
`git remote add stackhero-production ssh://stackhero@XXXXX.stackhero-network.com:222/project.git`

Finally, you can deploy to production or staging by using `git push stackhero-production master` or `git push stackhero-staging master`

Save your SSH private key password in Apple's keychain

On macOS, every time you want to push your code, your key password will be asked.
Security is great, but typing your password every time you push some code can be very annoying.

Some people may be tempted to remove the password from their SSH private key.
This is, obviously, a very bad idea and you shouldn't even think about it!

A better solution is to keep your password in Apple's keychain. It is a good compromise between security and ease of use.

To add your key password to keychain, simply type this command: `/usr/bin/ssh-add -K ~/.ssh/id_rsa`
You will not be prompted for your key password in the future and you will save a lot of time.

Handle secrets

Sometimes (often), you have to store some secrets, like tokens or passwords, but you can't store them to your repository for security reasons.

A solution is to use the environment variables.

You can add your secrets in environment variables on Stackhero dashboard and get access to them directly in your code.

For example, if youhave created a variable named "mySecret" in Stackhero dashboard, you will retrieve its content by reading `getenv("mySecret")` in your PHP code.

Handle PHP dependencies

When you push your code to your instance, our scripts will read the file `composer.json` and install all the corresponding dependencies via Composer.

Store files

If you need to store files, like users' photos, we really encourage you to use an object storage solution.

It gives you the ability to share these files between multiple services and multiple instances and split your uploaded files from your code. It's a best practice and you definitely should use this solution to handle your storage.

We recommend MinIO, that is an incredibly easy, fast and powerful solution, compatible with the S3 standard.

Nevertheless, if you want to store files locally, you can use the persistent storage offered with your PHP instance.

This storage is located in the directory `/persistent/storage/`.

To store an uploaded file, you can use the function `move_uploaded_file` like this:

```
move_uploaded_file($_FILES['image']['tmp_name'], '/persistent/storage/image.jpg');
```

You will get more informations about files upload on the official PHP documentation: https://www.php.net/manual/en/features.file-upload.php.

⚠️⚠️⚠️**CAUTION:** never store data outside the `/persistent/storage/` folder!

When your instance will be rebooted or when you will push some code, you can loose all your data stored outside the persistent storage!

Other articles about PHP that might interest you

- Connect to MySQL
  How to connect to MySQL from PHP using MySQLi