Architecture & Deployment

2025-2026 v0.1.0 on branch main Rev: 6d218297357081f922e8a7dd1d5cd8471c27fa79

Configure a PHP application through environment variables

The goal of this exercise is to improve the configuration step of the previous exercise by using environment variables instead of hardcoded configuration values.

Table of contents

Cloud server exercise

Parts of this exercise happen on the cloud server you should have created for this course. Log in and make sure you are connected to the internet to see your server's details.

Log in

📜 Legend

Parts of this exercise are annotated with the following icons:

❗ A task you MUST perform to complete the exercise

❓ An optional step that you may perform to make sure that everything is working correctly, or to set up additional tools that are not required but can help you

🏁 The end of the exercise

🏛️ The architecture of the software you ran or deployed during this exercise.

💥 Troubleshooting tips: how to fix common problems you might encounter

❗ Setup

Make sure you have completed the previous exercise first.

Stop your `php -S` command if it is still running.

> 💎 **Tip**
>
> You can use `Ctrl-C` to stop any command currently running in your terminal.

❗ Update the configuration

> 🛠️ Do this part of the exercise on your local machine, not on the server.

Clone the repository if you do not have it already:

```
$> cd /path/to/projects
$> git clone git@github.com:MyUser/php-todo-ex.git
```

> 💎 **Tip**
>
> Make sure to use your own fork of the repository, the same as in the previous exercise.

Modify the first few lines of `index.php` to take configuration values from the environment if available.

For example, instead of:

```
define('BASE_URL', '/');
```

Use this:

```
define('BASE_URL', getenv('TODOLIST_BASE_URL') ?: '/');
```

With this code, the `BASE_URL` variable will be equal to the value of the `TODOLIST_BASE_URL` environment variable if it has been set, or it will default to `/` if the environment variable is not available.

> 📚 **More information**
>
> This is accomplished using the PHP shorthand ternary operator `?:`.

DO NOT set a default value for the password, as it is a bad practice to hardcode sensitive values (as mentioned in the Config section of The Twelve-Factor App). The definition of the `DB_PASS` variable should have no default and look like this:

```
define('DB_PASS', getenv('TODOLIST_DB_PASS'));
```

Make sure to update the definitions of all other variables (`DB_USER`, `DB_NAME`, `DB_HOST` and `DB_PORT`) to take their value from the environment, with an appropriate default value.

> 💎 **Tip**
>
> Regarding the default values, you may assume that for a typical deployment, a MySQL database server is available on the host machine itself (`127.0.0.1`) and

> exposed on the default MySQL port (`3306`).

Commit and push your changes to the remote repository on GitHub.

❗ Pull the latest version from the server

> 🛠️ You may now connect to your server to perform the rest of the exercise.

Go into the cloned repository from the previous exercise (`~/todolist-repo` if you followed the instructions to the letter).

You may have made manual configuration changes during the previous exercise. You must discard them with the `git restore <file>` command. This will remove any uncommitted changes and restore the latest version of the file that was committed in the repository:

```
$> git restore index.php
```

You can now pull the latest version of the code from GitHub.

> 💎 **Tip**
>
> The command to pull the latest changes is `git pull <remote> <branch>`. If you do not remember the name(s) of your remote(s), you can list them with the `git remote` command (or with `git remote -v` to also see their URLs).

❗ Run the PHP development server

Still in the cloned repository, run a PHP development server on port 3000 like in the previous exercise. Note that this time you must provide the appropriate configuration

through environment variables:

- You must provide the `TODOLIST_DB_PASS` environment variable which has no default value.
- If the default values you have hardcoded for other variables are not suitable for your server's environment, you must also provide the corresponding environment variables with suitable values.

> 💎 **Tip**
>
> You can execute a command with additional environment variables using the following syntax: `EXAMPLE='value' ANOTHER='one' command arg1 arg2`.
>
> The single quotes around the variables' values are optional if the value contains no spaces or special characters.

You (and everybody else) should be able to access the application in a browser at the correct IP address and port (e.g. `W.X.Y.Z:3000`) and it should work.

🏁 What have I done?

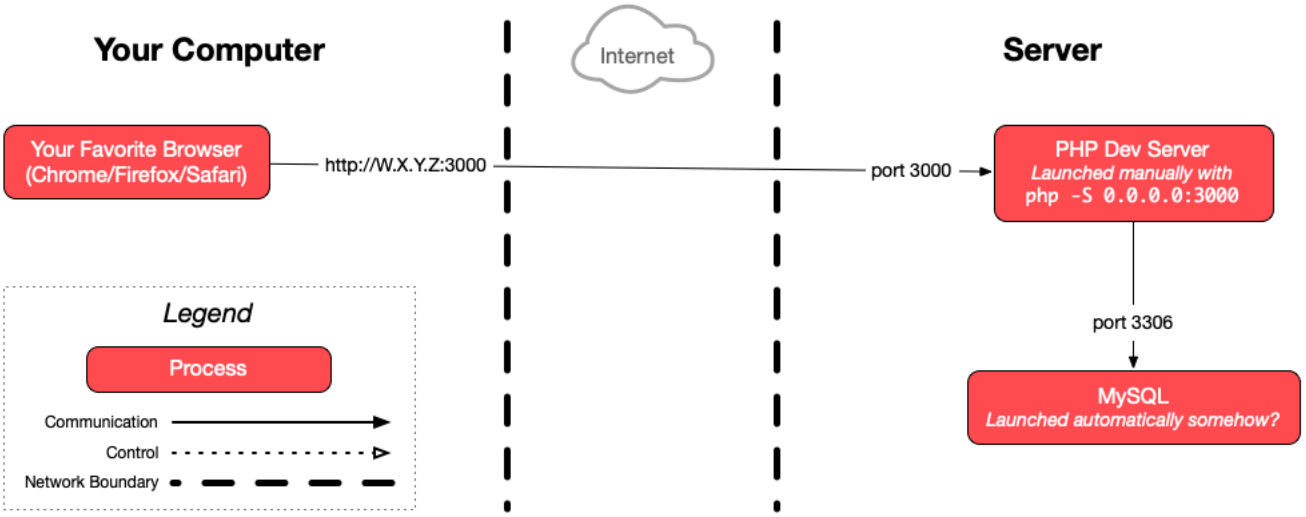
You have made your application configurable through the environment, as recommended in the Config section of The Twelve-Factor App.

This means that you no longer need to make any changes to the code before deploying your application to any new environment. It can now be deployed *anywhere*, on any server or on any developer's local machine, without changing a single line of code.

You simply need to set the appropriate environment variables when running it, and the application will use that configuration instead of the hardcoded defaults. For example, if you are deploying the application on a server where the MySQL database server is exposed on a non-standard port like `5000`, simply set the `TODOLIST_DB_PORT` variable, and the application will happily connect to it.

🏛️ Architecture

This is a simplified architecture of the main running processes and communication flow at the end of this exercise. Note that it has not changed compared to the previous exercises since we have neither created any new processes nor changed how they communicate.



↑ Back to top