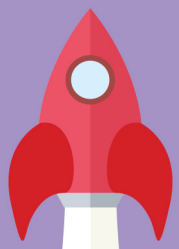
DEPLOYING PHP APPLICATIONS



BY NIKLAS MODESS

Deploying PHP Applications

Niklas Modess

This book is for sale at http://leanpub.com/deploying-php-applications

This version was published on 2017-10-17

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Niklas Modess by spreading the word about this book on Twitter!

The suggested hashtag for this book is #deployphpapps.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#deployphpapps

Contents

Introduction

Background

The PHP community is in an interesting phase right now and it has for a long time lagged behind on the more traditional software development practices. Practices that have been more or less considered as given in other communities. Continuous integration, package management, dependency injection and adopting object oriented programming to its extent to name a few. But as of PHP 5.3 there are no more excuses to why you can't write modern and clean code. The community have responded to this and a great number of people have stepped forward in supporting this through teaching and building tools for accomplishing these practices.

Yet we seem to forget deployment in this and I feel it's time to bring it to the table for discussion. There are a lot of great tools and services that aid you out there but few resources are available on how to build, maintain and optimize your deployment process. In this book I hope to shed some light on what is important and how it can be achieved.

Deploying is not just about pushing changes to a production server but an important part of the software development process[1]. You need to think of every step and every application is unique and need to be dealt with according to that. When working in a team is when you most of all need a great deployment process and the team should work in certain ways to allow for the process to be beneficial for all parties involved.

Every software application has a life cycle that the deployment process supports and is necessary for. You want to be able to maintain the application, roll out new features and do bug fixes without constant pain and headaches while doing it. You should be able to manage your branches, push your code, run the appropriate tests, migrate your database and deploy the changes in a fast and confident manner. With a good deployment process and a work flow that enables this it will be a breeze. If you can do a fast and easy rollback too, your confidence in pushing code will benefit a lot.

In a more and more agile software development world being able to deploy is important. Release cycles are getting shorter and shorter and some organisations even push it to the limit with continuous delivery[2]. In an environment with short release cycles, the importance of the deployment process intensifies. Not being able to deploy or rollback fast enough could end up slowing you down your entire development.

[1] http://en.wikipedia.org/wiki/Software_development_process
[2] http://en.wikipedia.org/wiki/Continuous_delivery

Who it's for

You are already familiar with PHP and you're not afraid of the command line. But you could also be a manager of a software development team that deal with deployment on a regular basis.

Legacy is not only a code issue but a process issue as well. Are you looking to streamline your deployment process or you want to scrap your current one and start of fresh? Then this book is for you.

Outside its scope

I consider server provisioning an almost crucial part of deploying your application but it's too big of a topic for this book. It could without a doubt be a book on its own (and perhaps will be?). I also don't want this book to focus on any framework or tool but keep the content and name broad instead of something like *Deploying PHP application to Amazon Web Services with Chef*.

The tools and commands used will be outside the scope unless it is a deployment tool (then it will have a dedicated section). I will make examples with Git, Composer, Grunt, PHPUnit and various other tools and if you want to learn more about the tools there are a vast amount of books, screencasts and blog posts to find, Google is your friend.

Assumptions

I know it's not nice to make assumptions about people or software. But I'm still going to do it to some extent in this book. I will make them when I approach examples but I'll not judge you or your application in any way.

Where you deploy to

You will need a hosting environment that you are some what in control of. If you are not able to install software or run commands it will limit you in what you can achieve. Whether it is a hosted server, co-location server or a VPS does not matter but to use everything in this book you need some control over it in terms of installing software, changing configuration, etc.

Git

The base of some of the topics discussed will use Git as version control. Why? Because I think it enables work flows that is best suitable for a good deployment process. There is a chapter on the topic as Git about version control and branching strategy for a good deployment process that could've been named *Git version control*. But I'll leave the name without Git in it since there are perhaps a lot in there that you can apply to other version control systems as well. Other than Git I have worked with Subversion and Perforce but when I found Git and started incorporating it in my work flow I have never looked back.

Both ends

There will also be an assumption about your application that it is not only a backend application. If your application is a REST API for example with no frontend what so ever it will not matter though. I will give some general examples on how to manage builds for your frontend as well but all the commands used will be arbitrary.

About the author

I have been developing PHP applications for over 10 years now. During this time I've developed and deployed a great variety of applications. The scale of these applications have been from a few hundred users to over 250 million users. Currently I'm working as a kind of free agent consultant focused on web architecture while spending the rest of my time on my own projects.

I'm one of two co-organizers for the meetup group Laravel Stockholm[3] trying to get some nice people together discussing Laravel and PHP concepts in general. I contribute to open source projects as much as possible. Most that is developing and maintaining Git Pretty Stats[4] which is a self hosted tool for statistics and graphs of Git repositories.

Oh, by the way I'm from Stockholm, Sweden. So I'm writing a book in my second language and I would appreciate all the help I can get when it comes to spelling and grammar. If you find anything, please create an issue in this repository[5] or fork it, fix it and send me a pull request. Thank you!

Code samples

In the repository on github[6], you can find code samples structured by chapter. Any substantial amount of code used in the book is available there for reference and use.

Thanks to

My friend and talented designer extraordinaire **Joakim Unge** for the awesome cover image. Look at it, it's a fucking rocket ship! You can find his portfolio at www.ashbagraphics.com[7].

My sister and **Jenny Modess** the talented copywriter, for proof reading from a non-technical perspective. Keeping my spelling, grammar and storytelling in check!

[3] http://www.meetup.com/Laravel-Stockholm/
[4] https://github.com/modess/git-pretty-stats
[5] https://github.com/modess/deploying-php-applications
[6] https://github.com/modess/deploying-php-applications
[7] http://www.ashbagraphics.com

1. Automate, automate, automate

I want to get this off the bat right away since one of the most crucial ingredients in your deployment process should be to **automate everything** in order to **minimize human errors**. If your deployment process involves manual steps, you're going to have a bad time and shit will hit the inevitable fan. Nobody is perfect. People can, and will, forget if they need to remember.

An automated deployment process will boost the confidence for the person deploying. Knowing that everything will be taken care of is a major contributor to feeling safe during a deploy. Of course other unexpected issues can arise but with a flexible process with logging and notifications you can relax and figure out what went wrong.

1.1 One button to rule them all

You should have **one** button to push or **one** command to run, to deploy your application. If you don't, something is wrong and you should automate all steps. Perhaps the only manual intervention I would consider okay is *"Are you sure? This will push stuff to production. [N/y]"*. This might be a bold statement but I truly believe in it.

Even if you're the only developer on a project and you're deploying the application each time I would say it's a bad practice having manual steps. When it comes to teams it becomes a lot worse since if not all team members are familiar with the deployment steps they won't be able to deploy. Team members come and go and whenever a new team member arrive they will need to learn how to deploy in the correct manner. Sure there can be documentation for it. But whenever someone is familiar enough with it they will most likely start to deploy without it.

1.2 Example of a manual step

Let's take an example where you have a revision number of your assets in code. I would say this is a rather common practice (unfortunately) and I've seen it on many occasions. This revision number handles cache busting so browsers do not keep serving old assets after a deploy.

This definition is a constant in a class somewhere for managing static assets.

```
class Assets
{
    const REVISION = 14;

    // [...]
}
```

Then it's applied to the static assets.

```
<link rel="stylesheet" type="text/css" href="style.css?v=<?=Assets::REVISION?>">
```

This is truly bad manual steps you could have. It is easy to forget since it requires a code change and if it's forgotten it might break the users' experience serving cached and out dated assets. A manual step where you would have to run a command in connection to the deploy would be better since it would be easier to remember. When you are already have the command line in front of you will be more prone to remember it.

Since it requires a code change another issue will likely occur. That is when you remember the step in the middle of the deploy just before pushing changes to production. You stop yourself before pressing the button screaming "Shit, the assets revision number!". Now you have to deal with changing the code, committing it and push the code through a new deploy. In a perfect world you would've already merged a release branch and tagged it (discussed in chapter 2), so how would you approach that now? Reset your repository, remove the tag, commit and create the release branch again? Or would you just commit and push, knowing that it will break traceability[8] for this specific deploy? This is a problem that goes away with automation.

1.3 Time is always a factor

You might be prone to say that you don't have time automating all the trivial steps or commands. If you spend one hour automating a command that takes one second to run you would have to run that command 3601 times before you have saved time on it. Yes, I did the math. While this is true I would say that it isn't the whole truth.

We need to take into account the time spent on dealing with issues arising when forgetting the step or command. Time spent on cleaning up. Can you measure bad user experience when your application breaks or behaves incorrect? Most likely not. In the previous example you can't tell how much time that you will spend on correcting that error and neither can you tell the impact on the users. If the problem is not caught in time it could persist for a long time.

[8] http://en.wikipedia.org/wiki/Traceability#Software_development