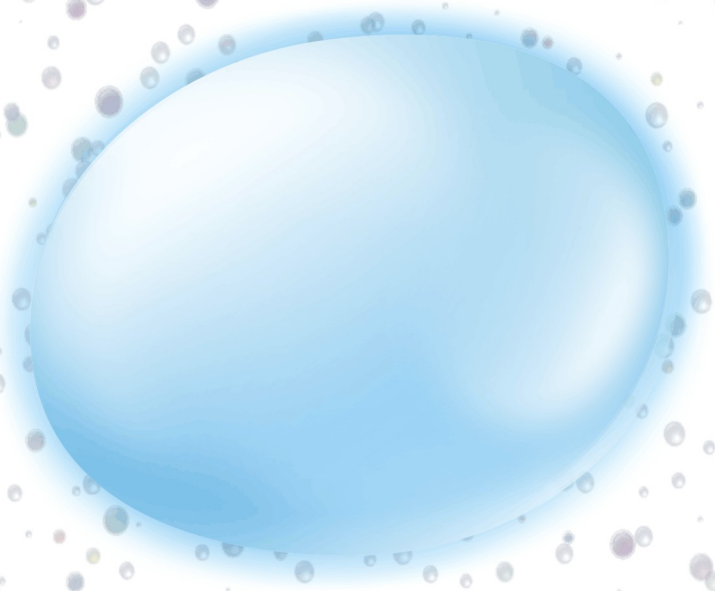
DurableDrupal
Lean



Tool up for solid web apps with a powerful Drupal distro

by

Victor Kane

DurableDrupal Lean

Tool up for solid web apps now and maintain them for years with a powerful easy to install Drupal distro

Victor Kane

Contents

Introduction

Tool up for solid web apps now and maintain them for years with a powerful Drupal distro

This book is for web app development teams, called upon by their clients to move their business model to the internet, their back-office to the cloud.

This book will empower you and your team, no matter what your level of expertise, to simplify the building of solid solutions for your clients' real needs, and to facilitate in a productive and cost-effective manner the maintenance these solutions will be crying out for from the moment they are born.

In the process, you will actually be evolving for yourselves a lean web factory to output durable solutions that can be maintained for years to come.

Change, maintenance, continous delivery and the long haul

Successful web developers and agencies know that the real challenge is how to embrace continuous delivery throughout the web app life cycle.

Delivering websites and web apps is only the beginning. The real question for success in web development is how to continue to ply your client with continual fixes and releases in order to fix problems and satisfy change requests.

Because constant change is inevitable, and ever-present throughout the web app life-cycle:

- First in the minimum of 40% in changes that will be requested in the weeks or months of development.
- Secondly, in the feature and usability requests the solution will have to assimilate in order to maintain its value over time.
- Thirdly in the security threats that will have to be dealt with in as fully an automated way as possible.
- Fourthly in the architectural and deployment changes that will be required as time goes by, in the struggle to scale with growth and cut costs with new technological breakthroughs.

Because it's not about the "single click site generation" and easy scaffolding any decent framework or "hosting" solution offers. We'll soon see how to fire up a framework in seconds, in just a few clicks, but the devil is in how we empower ourselves with a process that is going to facilitate, stabilize and simplify our work in the face of this constant change. By using the most exciting best of breed industry standard tools, like Ansible, Vagrant and Docker, Cloud-based hosting, and the all important Git branch model; just to mention a few of the things we will be working with here.

The theme of embracing change, while not new at all, marks the difference between success and failure, is born of the inevitable failures as well as the successes we all live through as we gain experience, and runs throughout the book so that web development can be fun instead of being a constant source of stress, code and maintenance debt.

We're here for the long haul.

And we're using Drupal[1] now.

Overview of Book Contents

So here's what we'll be doing:

- Chapter 1: Overcoming Choice Panic - What frameworks should we use?

  We need to free ourselves from choice panic which can paralyze us or disrupt our work. Dialog is necessary to ensure that the project stakeholders (all of them) are on board with the major framework optionss. We need to have this discussion sooner rather than later in order to share a common perspective and be ready to get down to work. In order to achieve this, we will be facing some tough questions together:
  - Building from scratch or not? Frameworks or not?
  - Proprietary or not? Open-source or not?
  - WordPress or not?
  - Drupal or not?
  - Drupal 8 or not?
  - Backdrop or not?
  - Vanilla Drupal 7 or not? Distro or not?
  - Presenting DurableDrupalDistro: fork it, tailor it, maintain it, branch and release it over all your projects!
  - Preparing to migrate gradually all along, so we'll be ready when the time comes
- Chapter 2: Setting up just enough shop to deliver our first web app
  - Team consciousness: client reps and dev team together
  - Shop infrastructre
    * Central code repos for all deliverables using the Git branching model
    * VPS MVP instances

[1] https://www.drupal.org/

            - Testing & Integration instance
            - Staging instance
            - Production ready instance
        * Every team member has their own dev environment automatically provisioned on their own laptop
        * Tools for running our agile developoment process
    - Fork the distro and make it our own
    - User stories and doing it with agile
    - Delivery as client empowerment
    - Post mortem: We did it! But…
        * Agile is the new waterfall (pixel perfect design -> "implementation")
        * Team (devs and designers, devs and client reps) couldn't really work in parallel
            - The tangle of User Stories and Design
        * Client took delivery of a monolithic blob he wasn't sure how to use
            - We had to do special training (not to mention train the trainers)
            - Client reps began a slew of change requests that could have come earlier if they had been familiar with the system
- Chapter 3: Agile is the new waterfall - we need Lean!
    - New project!
    - Value hypotheses
    - Design studio for nerds… and for the client too
    - Getting the team working in parallel
- Chapter 4: Functional prototypes
- Chapter 5: Grabbing the legacy content
- Chapter 6: Candidate architecture
    - Choices panic revisited
    - We're evolving on top of best of breed
- Chapter 7: Build the recursive MVP of MVP's
    - Every user story is an MVP
    - No project is ever anything more than an MVP
- Chapter 8: Go team test team go go go The advantage of using frameworks isn't just that they're tried and tested, but that they empower testing and TTD
- Chapter 9: Build, deploy, maintain
    - Rinse and repeat over "development"

Formatting Conventions

This book follows industry standard formatting and typographical conventions[2] for web development books.

[2] https://leanpub.com/help/manual

Obtaining and using the DurableDrupalDistro and other code examples

This is a book we need to work through together, so a starter Drupal Distro has been provided.

> For team dev, integration, staging and production DurableDrupalDistro provisioning, fork and adapt (or directly download and use) the Ansible playbook for setting up Durable Drupal Distro[3].

> Or grab the DurableDrupalDistro[4] itself and build out as you like.

At a minimum

Thanks to all the modern tools that have come along, all you really need is a laptop with some disk space and a decent amount of RAM. And we promise not to dirty it up with a lot of "server" stuff, thanks to Vagrant, Virtual Box, Ansible and Docker. We'll be setting up complete self-contained server replicas that won't affect your regular configuration at all. And you'll be able to follow through all the examples throughout the book.

Contact and feedback

Victor Kaneâ€¦ Book page… mailing lists

Acknowledgements

But more than a book this is a community of teams. Dive in!

[3] https://github.com/victorkane/ansible-vagrant-durable-drupal-distro
[4] https://github.com/victorkane/durable-drupal-distro

Chapter 1 - Overcoming Choice Panic - What frameworks should we use?

In this chapter we'll be putting Choice Panic behind us and making a firm decision on the CMS framework we'll be using for our content-centric projects. First we'll place all our doubts right on the table for all to see (should we be using Drupal for this project? really?), sift through the options, and choose the best one as our initial hypothesis. In making that decision we'll base ourselves on our current backlog of projects and on the information we actually have to go on right now. We'll explore that particular choice and the characteristics that make it the most versatile and dynamic option open to us. We want to be comfortable with the decision, and have the whole team all on the same page.

Now, also, above and beyond the framework, we are adopting a process which bases itself brutally on what we need and what our clients need, and which abandons vested vendor lock-in of any kind. We can't be married to any one set of frameworks any more. So the framework we select must be able to fit into our adopted, evolving and tailored process at a time when the industry is exploding with change and opportunities.

This is what we need and we need it now

Let's suppose we are a new web development team and we have our first job together. In the best agile tradition it's a "light" one, so that we can get to know each other and build some synergy to prepare us for a bigger project we'll be starting in a month or so (an online community of writers, a more challenging project we will be dealing with in successive chapters).

A neighborhood non-profit runs programs for youth in the community. They've had a webmaster who managed the content on their site for them, but he seems to have disappeared and they've lost access to their old stuff. Actually they've been wanting to re-do their site for some time, so they have some ideas, but now they're stuck. They want to be able to manage their content without depending on anyone else for that.

So what to do?

"Building from Scratch" or not? Frameworks or not?

By starting from scratch what is usually meant is not actually starting with a blank page, but rather the use of light-weight frameworks and libraries over larger CMS frameworks. Light-weight frameworks might give you more freedom and flexibility and may be easier to start using, whereas

larger frameworks offer a lot of built-in off-the-shelf functionality together with a large, easy to extend eco-system of plug-ins and add-ons. And using them often constitutes being part of a relatively large community of builders and users, offering possibilities for helping and being helped as well as for networking of many kinds.

Starting from Scratch: Pros

Many developers are attracted towards smaller, light-weight libraries and components because they feel they will be dealing with less complexity, and will be able to be more creative and to "break new ground". They also feel they will not be stuck with a monolithic system and faced with the task of "turning off" a great deal of the built-in functionality not required for many projects, which is something not always easy to do.

- You are building on top of shiny new technology
- Experimentation with new technologies and frameworks is not only fun but necessary in order to know what the options are.
- When a project is all about things that have never been done before, a completely straight-up ground-breaking disruptive Blue Ocean[5] effort, it only makes sense to find a best-fit candidate architecture which doesn't complicate things with its extra baggage and lowest common denominator features which won't bring anything to the table or may even be a nuisance.

Starting from Scratch: Cons

- If your are developing a content-centric project with many use cases that coincide with the larger CMS frameworks, then it really does not make any sense to reinvent the wheel. Countless features will have to be re-created: authentication and authorization (think: access rights for users and groups), forms management, database management, separation of concerns between business logic, data and the presentation layer, etc. It's a long list.
- The larger CMS frameworks are constantly monitored for security threats and constantly offer security updates when threats emerge. So much so that to some (superficial) eyes the sheer number of security fixes seems to constitute evidence of their being more likely to come under attack. But experience shows this nothing could be further from the truth. One-off apps will need to dedicate considerable time and resources towards security issues on many levels.
- Starting from scratch may need a larger team, with more specialists compared to using a CMS framework.
- You will have a harder time finding replacement team than you would if the project were being implemented with a large, well-known and highly adopted CMS framework like WordPress or Drupal, or Sitepoint.

[5] http://en.wikipedia.org/wiki/Blue_Ocean_Strategy

Note: In this author's opinion, the learning curve necessary to acquire the skills required for the development of a modern website or web application tends to actually be the same, no matter whether you "start from scratch" or decide to use a large CMS framework. Even though learning a large CMS framework like Drupal, for example, may take several months in order to be productive, the same is equally true when it comes to getting up to speed with other frameworks. In the case of Drupal you may spend a lot of time learning "how things are done". But if you go the other route, you will be learning "how things are done" in a whole series of smaller frameworks; while needing to find out "how to do things" quite a bit more than you would have to in the former case. So, starting from scratch may take even longer. I would say it's six of one and half a dozen of the other. Furthermore, much of what "learning Drupal" is often said to entail actually includes things proper to the web application domain itself, such as server provisioning, e-commerce, email management, maintenance, testing frameworks, and methods of deployment, etc. So this isn't a pro or a con, again, in this author's opinion and experience.

"Frameworks don't scale!" Well, this is simply just not true. The Emmys, Weather.com, and many other examples abound. Of course, it's going to call for a lot of work to get a framework to scale. It's not the kind of work you expect to be doing when you signed up for a CMS framework either: Load balancers, Content Delivery Networks, HTTP and Database optimizations, accelerators, are just the start. Bud did you seriously think you were going to avoid all that just by using a lighter framework? Actually, it's all the same work. So this isn't a pro or con either.

Proprietary or not? Open-source or not?

Once the choice of framework has been made for a given project or series of projects, it must then be decided to go either with a proprietary or an open source solution.

On this subject, let's first take a look at the title of a fairly recent article: ["Open Source vs. Proprietary Software: There is No Clear Winner"[(http://www.cmswire.com/cms/information-management/open-source-vs-proprietary-software-there-is-no-clear-winner-021752.php#null). This might inflame the hearts and minds of many of us who have unquestionably supported open source over the years. But it's time to coldly ask ourselves: which has better code, and what's the low-down about true cost to market?

The article itself begins by asking why anyone would want to ever use anything but Open Source Software: "It's free (or almost free); it's built by passionate communities of developers; you can "look under the hood"; and there's no vendor lock-in. Add to that, that the rate of innovation is supposed to be faster." And the short answer is that after a certain point of complexity is reached, it ceases to be true: it's no longer free even if there is no actual price tag on it, since the cost of (professional) use is much higher; it's increasingly built by company employees rather than a broad based passionate

community; the rate of innovation ceases to be fast; and a kind of vendor lock-in arises out of the sheer scale becoming necessary for development, deployment and maintenance.

- According to Software development testing leader Coverity[6], defect densities are practically identical. Where the number of lines of code[7] exceed 1 million, proprietary quality is better. The reason seems to be that not all open source projects have the budget to deal with the infrastructure required by the added complexity. Interestingly enough Drupal has 970,000[8], just under the limit.
- "The true value in open source is the open, collaborative community of like-minded professionals who come together to crowd-source ideas and build great things" (Kevin Cochrane, Alfresco). But proprietary software vendors have huge community support to, as is undeniably the case with Sharepoint Saturday's[9]. They have the "speakers", the "sponsors" and the T-shirts!
- In fact, there is a growing tendency within all open source frameworks for the number of very key core contributors to form part of an inner circle. In this way, the gap is closing between the relatively small number of "employee" committers on a proprietary project, on the one hand, and the ever-decreasing number of core committers on open source projects that grow in complexity.
- Cost to market actually converges in both cases also, even though of course results may vary: Proprietary solutions charge an up-front and/or licensing price, but Open Source frameworks require considerable development, training and administratation costs to accomplish what the former does out-of-the-box.

In this author's opinion technical support and infrastructre service levels may be obtained for both kinds of frameworks; and there are more and more consultants, agencies and companies with an enormously high level of domain knowledge for either kind of framework also. These areas cannot really be considered to be an advantage for only proprietary frameworks.

Rather than the choice being between Proprietary and Open Source, it is more about the project's real needs. "Paying developers to massively customise an open source CMS may end up costing you more than paying for a proprietary solution that more closely meets your requirements. On the flip side, paying license fees for an ill-suited proprietary solution doesn't make any sense either." (See Choosing Between an Open Source or Proprietary Website CMS[10]).

[6] http://www.coverity.com
[7] http://www.ohloh.net
[8] http://www.ohloh.net/p?ref=homepage&q=drupal
[9] http://www.spsevents.org
[10] http://www.netxtra.net/insights/choosing-between-an-open-source-or-proprietary-website-cms/

WordPress or not?

The stark fact here is that, from a CMS framework perspective, WordPress is less than Drupal, unless what you really want is an elegant industry-standard blog. While Drupal is a superset of WordPress, and one may migrate to Drupal comfortably without expecting to miss any features, the opposite is certainly not the case.

Unless your use cases coincide with the functionality offered by WordPress, in which case you still have a choice to make, the following points point consistently in Drupal's favor:

- It's when you contemplate porting a Drupal web application to a WordPress website that the latter's shortcomings really stand out.
- Despite Easy Post Types[11] Drupal wins hands down in terms of being able to create business objects with content types, having support for all kinds of field types, plus field collections and other features.
- Users are just another entity in Drupal and can use the same patterns that any old business object (taxonomy term, content type item, user, etc.) uses.
- Separation of concerns between the dynamic data model, the semantic HTML containing the dynamic content of the business object data model, and the styling to be applied to this basic unadorned un-styled HTML (that's called semantic HTML) is very well observed in Drupal, but badly supported in Wordpress, which means parrallel team development work is greatly impaired.
- WordPress has nothing like Drupal's Views: a complete and intuitive SQL Query Generator that's so intuitive most users don't realize it's an SQL Query Generator. It's really easy to list your stuff in all kinds of different ways.
- Rules module: built in support for UI based business rules, plus the ability to roll your own in code.
- Huge amount of very rich extensions (Drupal modules):
  - Complete layout and theming development system alternatives
    * Page Manager and Panels[12] (either with Panopoly[13] or do it yourself (as we do in our DurableDrupalDistro[14])
    * Display Suite[15]
    * Context module[16]
  - Workflow modules
    * Workbench[17]

[11] https://wordpress.org/plugins/easy-post-types/
[12] https://www.drupal.org/project/panels
[13] https://www.drupal.org/project/panopoly
[14] https://github.com/victorkane/drupal-lean
[15] https://www.drupal.org/project/ds
[16] https://www.drupal.org/project/context
[17] https://www.drupal.org/project/workbench

    * Workflow[18]
- Everything in code alternatives allow us to break the back of configuration in the database, and help our teamwork. We'll go into much more detail on this later.

Drupal or not?

Drupal 8 or not?

Backdrop or not?

Drupal 7 or not?

Won't everything we build in Drupal 7 be obsolete in a few months? Well no, actually Drupal 7 is an extremely mature project, with all kinds of tested and true themes and modules, including options for including much of the hotness of Drupal 8 right now and without the learning curve.

The thing is, every time we start a new project, we can't always be downloading the latest version, working to disable the cruft, adding in the modules and themes we always use again, configuring them again, testing everything again. We need a rich starting point. Fortunately we can get that in a distribution profile, a Drupal distro.

Drupal 7 Distro or not?

Use a distro that gives us a powerful and versatile starter kit, so we don't have to go combing through all the alternatives for modules, themes, plugins (unless we want to)

Everything in code with install profile and configuration management

Built in migration of content between legacy sites, development, test and staging, and live instances.

Easy to set up complete development environments for all team members no matter what their skill set.

A distro that can be part and parcel of a repeatable Agile and Lean process that can evolve together with changes in the industry as well as the needs our clients have and realize that they have over time.

It's not just about building, we have to be experts in maintenance, for the long haul. There are going to be security releases, upgrades of all kinds, new technology to integrate with, varying backends…

And a distro is something we can evolve, maintain over time as we grow and better identify a productive baseline we can build on.

[18] https://www.drupal.org/project/workflow

> The devil is in the maintenance

And we don't have to get married to any of the frameworks we're using. In fact, we want to be preparing to migrate gradually all along, so we'll be ready when the time comes

What we want is a re-usable distro that we can evolve over time and even migrate… something that will last us several years

Presenting DurableDrupalDistro with Lean process

Fork it, tailor it, maintain it, branch and release it over all your projects!

….. in this chapter we did …. In Chapter 2 - DurableDrupalDistro, we'll …

Chapter 2 - DurableDrupalDistro

Having made ourselves comfortable with the decision to fork, tailor and maintain a ready-to-use Drupal 7 distro as our workhorse, this chapter will actually relive the entire process of using it for the community website project.

On the one hand we want a cool bird's eye overview of everything that happens and how the team works together from a process point of view, so that the following chapters can drill down in the different aspects and components of applying the Lean process to Drupal site building and development.

On the other hand, we don't want to miss out on detailed instructions for actually doing it ourselves as we read along! These detailed instructions can be found in Appendix 1, and will be referenced from the text of this Chapter, which will be the method we use throughout this book. We can jump over to the appendix for details, while at the same time it will stand alone as a kind of categorized cookbook for DurableDrupalDistro. Fork it, add your own recipes and send us a pull request!

This chapter will explain how each team member can actually create a working environment for themselves. In this way the team will tackle a simple initial project while following a modern agile and lean development process.

Exactly how we built this thing

(see Lean Book for methodology)

Do it! and document it!

Basic team workflow

clone pull, branch, work, pull request rinse and repeat

"We'll first show you how to set up a workspace[19], getting a folder somewhere you can configure as a Drupal doc root, that is, the main Drupal directory that can be picked up by a LAMP stack and made available to a browser as a URL.

This can be done on a dedicated server, on a VPS, on a virtual machine or directly on your laptop, even on a good shared hosting, but you've got to have ssh access to the command line.

We all need to be able to do this no matter what our role on the team, so we'll show you a couple of examples.

Then we'll see how to grab our very own DurableDrupalDistro and fire it up[20].

[19] a_workspace_in_a_lamp_stack.html

[20] obtaining_and_installing_the_durabledrupaldistro.html

Finally, we'll check out what we have going for us[21] under the hood right off-the-shelf, we'll take our first steps[22], and then I'll give an example of updating and maintaining the distro itself[23]".

Getting a workspace in a LAMP stack

El Viejo shows here how to set up a workspace from scratch, in his usual opinionated fashion.

- on a VPS like Linode or Digital Ocean,
- on personal laptop/workstation alternatives
  - Water Works stack
  - Bitnami
    * Drupal App
    * Drupal Server
  - Kalabox
- Roll your own
- on a shared hosting account with command line access and drush

VPS

TODO linode

TODO digital ocean

Water Works stack

Well, Water Works never got around to offering one, or else they did and then started charging for it, so nobody used it and it died. But Acquia has one! Very convenient for running Drupal apps on your laptop, but it isn't a virtual machine, it's an app only, so that means that while you can run it and everything, it doesn't emulate the target live environment (a GOOD idea) the app will be running on when it is launched. Get it here[24].

Bitnami

Bitnami[25] swings both ways, with the app approach and the VirtualBox VM alternative (recommended but a lot of work[26] and then more work[27]).

[21] whats_included_in_durabledrupaldistro.html
[22] durabledrupaldistro_first_steps.html
[23] updating_durabledrupaldistro_itself.html
[24] http://www.acquia.com/downloads
[25] https://bitnami.com/stack/drupal
[26] http://awebfactory.com/node/524
[27] http://awebfactory.com/node/525

Pantheon

Single click …..

And you can have the best of both worlds, with pantheon in the cloud and on your desk… Kalabox!

Kalabox

Ah, the amigos at Kalamuna[28] are a breath of fresh air. They've not only come up with Kalabox[29] but are now developing Kalabox 2[30], the NodeWebkit[31] and Docker[32] wonder, as a completely open source project after a successful Kickstarter campaign[33].

I've tried it out quite a bit. See how I installed Kalabox 1[34] and used it as a Pantheon workflow solution. Integration with Pantheon hosting[35] and their one-click Drupal Distro site builder is a hallmark of Kalabox 1. Very useful considering its revolutionary architecture[36] and standardizing workflow[37] (at least as far as Drupal hosting is concerned). Alternative hosting support plug-ins are expected with Kalabox 2.

Ansible

(study up on Ansible for Dev-Ops book first! buy the bundle!) * Ansible on a Mac to run a server * Ansible on a Mac to run a local vagrant/vbox instance

Roll your own

sthg about virtual box & vagrant, with puppet, chef, ansible…

Shared hosting account

We shouldn't even be talking about this, but recent improvements in shared hosting accounts have made something like a reseller account somewhat (only just) viable, at least for hosting small to middling projects.

First thing to do is get Drush on there, once you've gotten ssh command-line access, if you can't get that switch hosting now (and if you're even reading this section, second thing is to create a subdomain for each site: don't go running Drupal from a sub-directory).

[28] http://www.kalamuna.com/
[29] http://www.kalamuna.com/products/kalabox
[30] https://github.com/kalabox
[31] https://github.com/rogerwang/node-webkit
[32] https://www.docker.com/
[33] https://www.kickstarter.com/projects/kalabox/kalabox-advanced-web-tools-for-the-people
[34] http://awebfactory.com/node/522
[35] https://www.getpantheon.com/
[36] https://www.getpantheon.com/how-it-works
[37] https://www.getpantheon.com/how-it-works

Bringing the whole team up with the DurableDrupalDistro

What's included in DurableDrupalDistro

DurableDrupalDistro for our first project

So we're in oue workspace, let's say on a VPS.

What we want to do

- Create virtual host
- Create database
- Clone and set up distro
- Install with drush on the command line
- Check it out

Create virtual host

Create database

Clone and set up distro #### Install with drush on the command line #### Check it out

Updating DurableDrupalDistro itself

```
$ drush pm-refresh
Refreshing update status information ...
Done.
$ drush pm-update
 Name                 Installed  Proposed  Message
                      Version    version
 Drupal               7.31       7.32      SECURITY UPDATE available
 Calendar (calendar)  7.x-3.4    7.x-3.5   Update available
 CKEditor (ckeditor)  7.x-1.15   7.x-1.16  SECURITY UPDATE available
 Profiler Builder     7.x-1.1    7.x-1.2   Update available
 (profiler_builder)


Update information last refreshed: Tue, 10/21/2014 - 13:20
```

```
NOTE: A security update for the Drupal core is available.
Drupal core will be updated after all of the non-core projects are updated.

Security and code updates will be made to the following projects: Calendar [cale\
ndar-7.x-3.5], CKEditor - WYSIWYG HTML editor [ckeditor-7.x-1.16], Profiler Buil\
der [profiler_builder-7.x-1.2]

Note: A backup of your project will be stored to backups directory if it is not \
managed by a supported version control system.
Note: If you have made any modifications to any file that belongs to one of thes\
e projects, you will have to migrate those modifications after updating.
Do you really want to continue with the update process? (y/n):
Project calendar was updated successfully. Installed version is now 7.x-3.5.
Backups were saved into the directory                                [ok]
/home/drupallean/drush-backups/drupal_lean/20141021132817/modules/calendar.
Project ckeditor was updated successfully. Installed version is now 7.x-1.16.
Backups were saved into the directory                                [ok]
/home/drupallean/drush-backups/drupal_lean/20141021132817/modules/ckeditor.
Project profiler_builder was updated successfully. Installed version is now 7.x-\
1.2.
Backups were saved into the directory                                [ok]
/home/drupallean/drush-backups/drupal_lean/20141021132817/modules/profiler_build\
er.

Code updates will be made to drupal core.
WARNING:  Updating core will discard any modifications made to Drupal core files\
, most noteworthy among these are .htaccess and robots.txt.  If you have made an\
y modifications to these files, please back them up before updating so that you \
can re-create your modifications in the updated version of the file.
Note: Updating core can potentially break your site. It is NOT recommended to up\
date production sites without prior testing.

Do you really want to continue? (y/n):
Project drupal was updated successfully. Installed version is now 7.32.
Backups were saved into the directory                                [ok]
/home/drupallean/drush-backups/drupal_lean/20141021132817/drupal.
No database updates required                                         [success]
'all' cache was cleared.                                             [success]
Finished performing updates.                                         [ok]
$ git status
$ git add .
$ git commit -am "Maintenance update to drupal core 7.32 plus updated contrib mo\
```

```
dules"
$ git tag
$ git tag -a "October2014" -m "Maintenance core and contrib update"
$ git push --tags
```

Now we need to prepare the site (see Quick install for developers (command line)[38]

```
cp sites/default/default.settings.php sites/default/settings.php
```

Give the web server write privileges (666 or u=rw,g=rw,o=rw) to the configuration file.

```
chmod a+w sites/default/settings.php
```

Give the web server write privileges to the sites/default directory.

```
chmod a+w sites/default
```

so we can re-install from scratch to test. Check out incredible options available with drush help:

```
$ drush help site-install
Install Drupal along with modules/themes/configuration using the specified
install profile.

Examples:
 drush site-install expert --locale=uk     (Re)install using the expert install
                                           profile. Set default language to
                                           Ukranian.
 drush site-install                        Install using the specified DB
 --db-url=mysql://root:pass@localhost:por  params.
 t/dbname
 drush site-install                        Install using SQLite (D7+ only).
 --db-url=sqlite://sites/example.com/file
 s/.ht.sqlite
 drush site-install --account-name=joe     Re-install with specified uid1
 --account-pass=mom                        credentials.
 drush site-install standard               Pass additional arguments to the
 install_configure_form.site_default_coun  profile (D7 example shown here - for
 try=FR                                    D6, omit the form id).
 my_profile_form.my_settings.key=value
```

[38] https://www.drupal.org/documentation/install/developers

```
 drush site-install                          Disable email notification during
 install_configure_form.update_status_mod    install and later. If your server
 ule='array(FALSE,FALSE)'                    has no smtp, this gets rid of an
                                             error during install.

Arguments:
 profile                                     the install profile you wish to run.
                                             defaults to 'default' in D6,
                                             'standard' in D7+
 key=value...                                any additional settings you wish to
                                             pass to the profile. Fully supported
                                             on D7+, partially supported on D6
                                             (single step configure forms only).
                                             The key is in the form [form
                                             name].[parameter name] on D7 or just
                                             [parameter name] on D6.

Options:
 --account-mail                              uid1 email. Defaults to
                                             admin@example.com
 --account-name                              uid1 name. Defaults to admin
 --account-pass                              uid1 pass. Defaults to a randomly
                                             generated password. If desired, set
                                             a fixed password in drushrc.php.
 --clean-url                                 Defaults to 1
 --db-prefix                                 An optional table prefix to use for
                                             initial install.  Can be a key-value
                                             array of tables/prefixes in a
                                             drushrc file (not the command line).
 --db-su=<root>                              Account to use when creating a new
                                             database. Must have Grant permission
                                             (mysql only). Optional.
 --db-su-pw=<pass>                           Password for the "db-su" account.
                                             Optional.
 --db-url=<mysql://root:pass@host/db>        A Drupal 6 style database URL. Only
                                             required for initial install - not
                                             re-install.
 --locale=<en-GB>                            A short language code. Sets the
                                             default site language. Language
                                             files must already be present. You
                                             may use download command to get
                                             them.
```

```
 --site-mail                              From: for system mailings. Defaults
                                          to admin@example.com
 --site-name                              Defaults to Site-Install
 --sites-subdir=<directory_name>          Name of directory under 'sites'
                                          which should be created. Only needed
                                          when the subdirectory does not
                                          already exist. Defaults to 'default'

Aliases: si
```

So let's install in Spanish, specifying db credentials, uid1 user name and password, uid1 user email, site email and site name

```
site-install drupallean --locale=es --db-url=mysql://drupal_lean:lean@localhost/\
drupal_lean --account-name=admin --account-pass=omg --account-mail=victorkane@gm\
ail.com --site-mail=drupallean@awebfactory.com.ar --site-name=DrupalLean

You are about to DROP all tables in your 'drupal_lean' database. Do you want to \
continue? (y/n): y
No tables to drop.                                                 [ok]
Starting Drupal installation. This takes a few seconds ...         [ok]
WD php: Warning: Invalid argument supplied for foreach() in        [warning]
_features_restore() (line 966 of
/home/drupallean/drupal-lean/sites/all/modules/contrib/features/features.module).
Installation complete.  User name: admin  User password: omg     [ok]
```

Post Mortem (or Agile is the new Waterfall)

So in this chapter we ...

For the rest of the book, we are going to drill down into the different moments of a more challenging project. Turn the page and find out what it's gonna be...

Chapter 3 - Value Hypotheses for the project

Chapter 4 - Functional prototypes

Chapter 5 - Grabbing the legacy content

Chapter 6 - Candidate architecture

Chapter 7 - Build the recursive MVP of MVPs

Chapter 8 - Go team test team go go go

> If I hear “Get out of the building” one more time I think I’ll scream

It’s not just about code that runs right, even though it must.

It’s about value! People’s needs!

Chapter 9 - Build, deploy, rinse, repeat

If I hear “Rinse and repeat” one more time I think I’ll scream

“Whatever happened to getting it right the first time?” muttered Jason, sick and tired of stuff he thought was already done being “postponed” to “the next iteration”.

Appendix 1 - How To #1

how to

Appendix 2 - How To #2

how to