Chef by example

Practical Exercises in a Successful Chef Deployment.

Carlos Camacho

Contents

Preface

The source code of the book is hosted in Github, everyone can fork and make pull requests; you are all invited.

This is my Chef Style DevOps Kung-fu implementation, inspired in the Adam Jacob keynote from Chefconf 2015. A practical roadmap to deal with software developers, infrastructure, continuous delivery, automation and the lack of coffee in the office…

There isnt an agreed definition for DevOps yet. DevOps is a cultural and professional movement, focused on how we build and operate high velocity organizations, born from the experience of its practitioners.

Principles (Universal)

- Based on prioritize people over products over companies.
- Design for the safety, contentment, knowledge and freedom of both your peers and your customers.
- Eliminate non-value-added actions and processes.
- Continuously improve your processes.
- Adapt to needs.
- Small improvements over the time.
- Fail faster to learn faster. Be calm, fix things and improve your processes.
- Workflows automation.

- Diversity, gets feedback, have different opinions, argue, make demonstrations on your points of view.
- Demo all the things you have or you are working in.
- Be the owner of your work, love your work, and find reasons to do your work.
- Improve and do things right even if is hard (At first).
- Make decisions based on your experience and proof your point of view.

Forms (Shared)
Applications (Unique)
A unique approach, based in previous experience focused on customers.
The excellence achieved thought long practice in ones skill

Chapter 1

The first chapter

- General configuration structure for the chef-repo:
  - chef-repo/environments/banana.rb
  - chef-repo/environments/potato.rb
  - chef-repo/environments/kiwi.rb
  - chef-repo/data_bags/banana.rb
  - chef-repo/data_bags/potato.rb
  - chef-repo/data_bags/kiwi.rb:
  - chef-repo/roles/base.rb
  - chef-repo/roles/web.rb
  - chef-repo/roles/db.rb
- Banana cookbook structure:
  - chef-repo/cookbooks/banana/templates/default/*.erb
  - chef-repo/cookbooks/banana/attributes/default.rb
- Potato cookbook structure:
  - chef-repo/cookbooks/potato/templates/default/*.erb

  – chef-repo/cookbooks/potato/attributes/default.rb

-Kiwi cookbook structure: * chef-repo/cookbooks/kiwi/templates/default/*.erb * chef-repo/cookbooks/kiwi/attributes/default.rb

A node belongs to an environment in which case, will override the default configuration per the corresponding one.

Override app attributes for kiwi (Non sensitive info) Override app attributes for kiwi (Sensitive info) Define the recipes for the xxx role

Default configuration templates for kiwi Default configuration values according the templates for kiwi

This is the first paragraph of the Softcover Markdown template produced with the `softcover` command-line interface. It shows how to write a document in Markdown, a lightweight markup language, augmented with the kramdown converter and some custom extensions, including support for embedded PolyTEX, a subset of the powerful LATEX typesetting system.[1] For more information, see *The Softcover Book*. To learn how to easily publish (and optionally sell) documents produced with Softcover, visit Softcover.io.

This is the *second* paragraph, showing how to emphasize text.[2] You can also make text **bold** or *emphasize a second way*. Via embedded PolyTEX, Softcover also supports colored text, such as red, cornflower blue, and arbitrary HTML colors.

1.1 A section

This is a section. You can refer to it using the LATEX cross-reference syntax, like so: Section 1.1.

1.1.1 Source code

This is a subsection.

[1]Pronunciations of "LaTeX" differ, but *lay*-tech is the one I prefer.

[2]This is a footnote. It is numbered automatically.

You can typeset code samples and other verbatim text using four spaces of indentation:

```
def hello
  puts "hello, world"
end
```

Softcover also comes with full support for syntax-highlighted source code using kramdown's default syntax, which combines the language name with indentation:

```ruby
def hello
  puts "hello, world"
end
```

Softcover's Markdown mode also extends kramdown to support "code fencing" from GitHub-flavored Markdown:

```ruby
def hello
  puts "hello, world!"
end
```

The last of these can be combined with PolyTeX's `codelisting` environment to make code listings with linked cross-references (Listing 1.1).

Listing 1.1: Hello, world.

```ruby
def hello
  puts "hello, world!"
end
```

1.1.2 Mathematics

Softcover's Markdown mode supports mathematical typesetting using LaTeX syntax, including inline math, such as $\phi^2 - \phi - 1 = 0$, and centered math, such as

$$\phi = \frac{1+\sqrt{5}}{2}.$$

It also supports centered equations with linked cross-reference via embedded PolyTEX (Eq. (1.1)).

$$\phi = \frac{1+\sqrt{5}}{2} \tag{1.1}$$

Softcover also supports an alternate math syntax, such as $\phi^2 - \phi - 1 = 0$, and centered math, such as

$$\phi = \frac{1+\sqrt{5}}{2}.$$

The LaTeX syntax is strongly preferred, but the alternate syntax is included for maximum compatibility with other systems.

1.2 Images and tables

This is the second section.

Softcover supports the inclusion of images, like this:

Using LaTeX labels, you can also include a caption (as in Figure 1.1) or just a figure number (as in Figure 1.2).

Figure 1.1: Some dude.

1.2.1 Tables

Softcover supports raw tables via a simple table syntax:

Figure 1.2

HTTP request	**URL**	**Action**	**Purpose**
`GET`	/users	`index`	page to list all users
`GET`	/users/1	`show`	page to show user with id `1`
`GET`	/users/new	`new`	page to make a new user
`POST`	/users	`create`	create a new user
`GET`	/users/1/edit	`edit`	page to edit user with id `1`
`PATCH`	/users/1	`update`	update user with id `1`
`DELETE`	/users/1	`destroy`	delete user with id `1`

See *The Softcover Book* to learn how to make more complicated tables.

1.3 Command-line interface

Softcover comes with a command-line interface called `softcover`. To get more information, just run `softcover help`:

```
$ softcover help
Commands:
  softcover build, build:all       # Build all formats
  softcover build:epub         # Build EPUB
  softcover build:html         # Build HTML
  softcover build:mobi         # Build MOBI
  softcover build:pdf          # Build PDF
  softcover build:preview          # Build book preview in all formats
  .
  .
  .
```

You can run `softcover help <command>` to get additional help on a given command:

```
$ softcover help build
Usage:
  softcover build, build:all

Options:
  -q, [--quiet]   # Quiet output
  -s, [--silent]  # Silent output
```

```
Build all formats
```

1.4 Miscellanea

This is the end of the template—apart from two mostly empty chapters. In fact, lets include the last chapter in its entirety, just to see how mostly empty it is:

```
# Yet *another* chapter

*This chapter left intentionally blank*
```

Visit *The Softcover Book* to learn more about what Softcover can do.

Chapter 2

Another chapter

This is another chapter.[1] It also includes a little code fencing, mainly to test an edge case for the sake of the Softcover test suite:[2]

```
$ find . \( -name \*.gemspec -or -name \*.jpg \) -type f
```

[1]Footnotes are numbered on a per-chapter basis.

[2]The test suite uses the template files to stress-test the build system. In this case, there used to be a bug when math syntax appeared in a non-math context. Including the code fencing as above ensures that any regressions will cause the test suite to fail.

Chapter 3

Yet *another* chapter

This chapter left intentionally blank