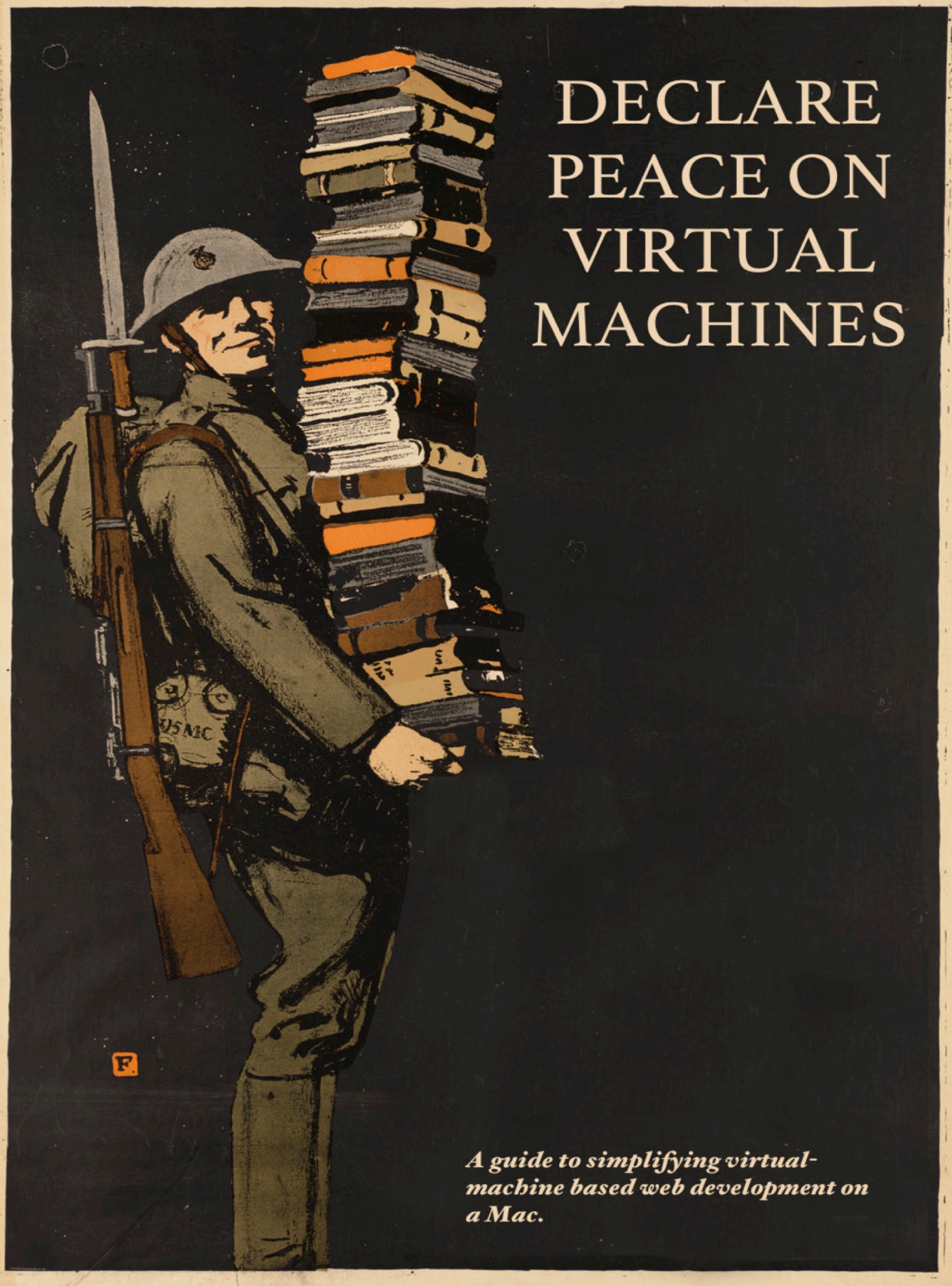

DECLARE PEACE ON VIRTUAL MACHINES

A guide to simplifying virtual-machine based web development on a Mac.

Declare Peace On Virtual Machines

A guide to simplifying virtual-machine based web development on a Mac

Murphy J. C. Randle

This book is for sale at http://leanpub.com/declarepeaceonvms

This version was published on 2013-12-03

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 Murphy J. C. Randle by spreading the word about this book on Twitter!

The suggested hashtag for this book is #declarepeaceonvms.

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

https://twitter.com/search?q=#declarepeaceonvms

To my wife, Laura, whom I love. And to our expected son.

Contents

Introduction

You may be a Web developer working on a Macbook. You may work for a company whose back-end depends upon scripts that are written only to work on Debian Linux. You may find out that the practical way for you to work with your company's tools is by installing Linux on a VM (Virtual Machine), and running the company code inside that VM.

You may be reluctant to spend two whole days setting up the VM and figuring out how to comfortably communicate with it.

This book is meant to help you set up and interact with a new VM running Linux in a matter of hours rather than days. The goal is to make the experience of working with, and working within, your VM as close as possible to the experience of working from your native shell.

A working knowledge of the command-line is essential to understanding the examples in this book. If you are a full-stack or a back-end developer you will have no problem. If you are a front-end developer who sweats at the mere mention of the command-line, I promise you, you're going to be okay :). Learning to use the command-line is not only essential to your work as a Web developer, but will expand your abilities to succeed in front-end development immensely. There are plenty of tutorials that can be found on Google to teach basic command-line usage. Here's one from Tuts+[1]

I've distilled the frustrations I've experienced when working with VMs in to 3 basic problems:

- Configuration
- File Sharing
- Integration

The next three chapters of this book will explain each of these problems, and then present a solution to the problem with an example of that solution applied.

About Me

I work as a full-stack Web developer at Space Monkey, Inc[2]. Our back-end private API is written mostly in Python, and it has many dependencies which we've prepared for distribution to the servers as Debian packages. The most straightforward way for me to get our API (Application Programming Interface) up and running locally for development was to do it on a Virtual Machine; I work on a Macbook Pro running OS X which isn't compatible with Debian packages. (Debian is a Linux

[1] http://webdesign.tutsplus.com/tutorials/workflow-tutorials/quick-tip-a-designers-introduction-to-the-command-line/
[2] http://www.spacemonkey.com

operating system, and Debian packages are a standardized method of installing, configuring, and managing software). I got so frustrated with the large number of steps necessary not only to set up a VM, but to work within one that I dedicated a number of days to simplifying the process. Now when I need to make a new VM it's a matter of minutes rather than hours.

Conventions

Most of the figures in this book will be examples of commands that should be issued into your terminal. All terminal commands will appear in code blocks, and will be preceded by the '>' character at the beginning of the line like so:

```
> echo "Hello, World!"
```

Lines that end with a backslash: \ will be joined by the shell with the line that comes after. This happens when a single command is too long to fit on one line:

```
> this command is so very long that I can\
barely keep it in my head!
```

I'll accompany code with comments that are meant solely for clarification and should **not** be run in the terminal. These comments will be preceded by a '#' character like so:

```
# This is a comment.
# Don't enter it into the command line!
```

If a line is not begun with the characters >, or #, and is not preceded by a line ending in \, then it shows the shell's response to the command entered:

```
> print "I'm printing a command!"
I'm printing a command!
```

Terminology

I'll use lots of abbreviations and acronyms in this book. The first time I use an acronym, I'll write its expanded form in parenthesis, like this: AAG (Acronyms Are Great). For your convenience, here's a short list of acronyms, abbreviations, and potentially confusing terms that I use in this book:

CLI Command-line interface. A textual interface to a tool that is meant to run on the command-line.

guest
: An operating system running in a VM.

GUI A graphical user interface.

host An operating system running VM software.

OS Operating System.

provider
: Software that runs a VM on a host.

repo
: A source-code repository. A location on a local computer or on a server where source code changes are tracked and stored.

VM Virtual Machine. An emulated environment to run one OS inside of another.

Prerequisites

I'll be making regular use of the Git revision control system in this book. If you want to follow along with the examples I suggest visiting the Git Web site.[3] to learn how to install and use Git.

Homebrew is a package manager built for OS X that I will also be using in the examples to install the dependencies that Apollo has. To learn about installation and basic usage of Homebrew I refer you to the Homebrew Web site[4].

As I mentioned above in the introduction, the tools in this book are all used from the command-line. If you aren't comfortable in the command-line, I suggest reading a few tutorials and practicing a little before diving into this book.

In order to run a VM, software is necessary that emulate the hardware that a certain OS (Operating System) expects to encounter. There are a variety of VM software packages (providers) available for Mac. This guide will use Virtual Box[5], a free provider that's actively developed and supported by Oracle. However, this book should work with any providers that are supported by Vagrant[6]. More information about Vagrant can be found in Chapter 1.

The Scenario

The scenario for the examples in this book is a simplified version of a Web development task. Explaining the complex back-end we use at Space Monkey wouldn't be appropriate for this short

[3] http://git-scm.com/
[4] http://brew.sh/
[5] https://www.virtualbox.org/
[6] http://docs.vagrantup.com/v2/providers/

book. Instead, I'll show how to use a VM to run a new tool called Docker[7], which will help me set up a fresh instance of the Ghost blogging platform[8] in a Docker container[9].

This book is **not** a Docker tutorial. It only aims to make working with VMs easier. I'm using Docker in the examples, however, because it's a new tool that's rapidly growing in popularity. Since it only runs on Linux, developers who use a Mac need run Docker within a VM.

Docker containers are similar to VMs in that a container is a self-contained copy of a Linux OS. But containers can only be used to run a Linux OS within another Linux OS.

In the end, we'll have a Docker container running inside of a VM with Ubuntu[10] installed, which will be managed by Vagrant, running on OS X. See Figure A for a graphic that shows the final machine configuration that the example will lead to.

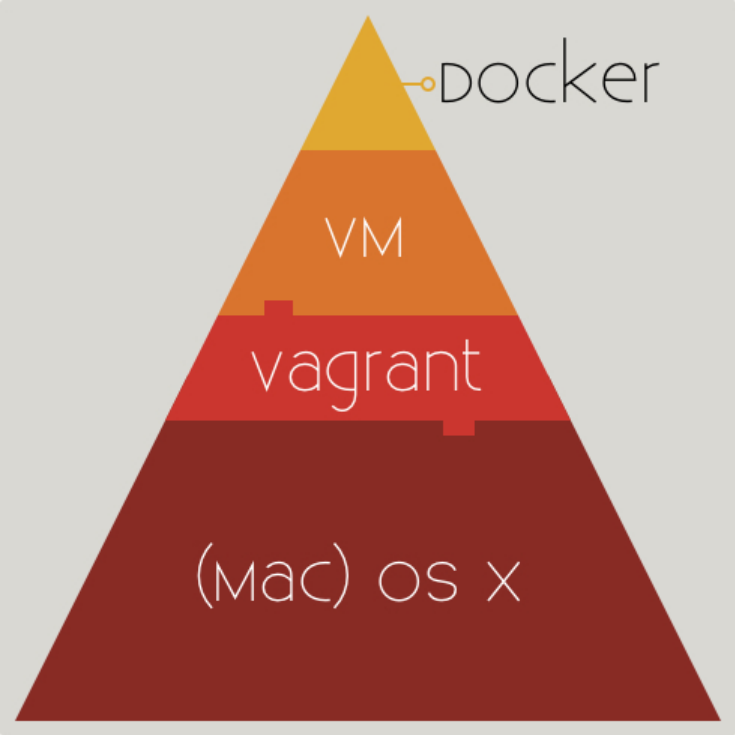

Figure A: The example will end with a Docker container running inside a Ubuntu VM, managed by Vagrant, and running on OS X

[7] http://www.docker.io/
[8] https://ghost.org/
[9] http://docs.docker.io/en/latest/terms/container/#container-def
[10] http://www.ubuntu.com/