Quick answers to common problems

Vagrant Virtual Development Environment Cookbook

Over 35 hands-on recipes to help you master Vagrant, and create and manage virtual computational environments

Chad Thompson

[PACKT] open source
PUBLISHING
community experience distilled

Table of Contents

Vagrant Virtual Development Environment Cookbook

Vagrant Virtual Development Environment Cookbook

First published: February 2015

Production reference: 1210215

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham B3 2PB, UK.

ISBN 978-1-78439-374-8

www.packtpub.com

Credits

Author

Chad Thompson

Reviewers

Emilien Kenler

Darius Krištapavičius

Marcelo Pinheiro

Commissioning Editor

Usha Iyer

Acquisition Editor

Richard Brookes-Bland

Content Development Editor

Arwa Manasawala

Technical Editors

Vijin Boricha

Humera Shaikh

Copy Editor

Relin Hedly

Project Coordinator

Danuta Jones

Proofreaders

Simran Bhogal

Bridget Braund

Indexer

Hemangini Bari

Production Coordinator

Aparna Bhagat

Cover Work

Aparna Bhagat

About the Author

Chad Thompson is a software developer, architect, and DevOps specialist in Central Iowa. He has 15 years of experience in creating and deploying applications for the Web. Chad began using Vagrant 3 years ago when he was trying to solve a tough problem in legacy application development. Since then, he has made use of Vagrant and configuration management tools to support the development and deployment of several web applications i data centers and cloud platforms. He holds certifications in Puppet and Oracle technologies and has enjoyed the pleasure of speaking before several technical conferences and camps. Chad holds two degrees in physics and can be found playing low brass instruments in ensembles around the state of Iowa.

Chad has written articles for O'Reilly web publications and the IOUG SELECT Journal (where he briefly worked as an executive editor). Recently, he reviewed the book *Creating Development Environments with Vagrant* for Packt Publishing, and recorded a set of video presentations titled *Learning Git* by Infinite Skills.

I owe a great measure of gratitude to many people for helping me with the production of this book. I would like to thank my colleagues at Dice Holdings Inc. for their support and feedback during the development of the book. I would like to thank Zach Arlen of FullContact in Denver, CO, for introducing me to Vagrant as a solution to a problem years ago. Mostly, I would like to thank my family for their continued love and support.

With the publication of this book, I would also like to offer my gratitude to Dr. Robert Merlino and the late Dr. Nicola D'Angelo of the University of Iowa. They both taught me a great deal about formulating ideas and teaching others, which I hope serves the readers of this book.

About the Reviewers

Emilien Kenler, after working on small web projects, began focusing on game development in 2008 while he was in high school. Until 2011, he worked for different groups and specialize in system administration.

In 2011, he founded a company that sold Minecraft servers while studying computer science engineering. Emilien created a lightweight IaaS (https://github.com/HostYourCreeper/) based on new technologies (such as Node.js and RabbitMQ).

Thereafter, he worked at TaDaweb as a system administrator, building its infrastructure and creating tools to manage deployments and monitoring.

In 2014, he began a new adventure at Wizcorp, Tokyo. In 2014, Emilien graduated from the University of Technology of Compiègne.

For Packt Publishing, Emilien has also contributed as a reviewer on other books:

- *Learning Nagios 4*, *Wojciech Kocjan* (http://www.packtpub.com/learning-nagios-4/book)
- *MariaDB High Performance*, *Pierre MAVRO* (https://www.packtpub.com/big-data-and-business-intelligence/mariadb-high-performance)
- *OpenVZ Essentials*, *Mark Furman*, (https://www.packtpub.com/virtualization-and-cloud/openvz-essentials)

Darius Krištapavičius attended Vilnius University and studied software engineering as his major subject. In 2009, Darius started working with web application development and since then, he gained considerable experience and particularly developed various e-commerce systems. While working in this field, he learned the PHP programming language and different frameworks (such as CodeIgniter and Symfony2). At present, Darius is working as a professional web developer and is actively engaged in DevOps method, process automation, principles of Agile, and other associated subjects of web development.

Marcelo Pinheiro is a software engineer from Porto Alegre, Brazil. In 2000, he started to work as a web designer and programmer with ASP and PHP. Marcelo is still in touch with Microsoft .NET Framework and Java to run their respective choice of databases for web applications. Since 2003, he has been using Linux- and Unix-related operational systems, from Slackware to GoboLinux, Arch Linux, CentOS, and Debian. At present, he uses OS X, and he also uses FreeBSD to some extent. Marcelo lost a few nights compiling and applying patches on the Linux kernel to make their desktop work. He is an open source enthusiast an acts as a problem solver, irrespective of the programming language, database, or platform.

After a few years, he moved to São Paulo to work with newer technologies (such as NoSQL, cloud computing, and Ruby), where he began to present tech talks with these technologies ir Locaweb, and RS on Rails, the biggest Ruby conference in South Brazil. As an observer, he created some tools to standardize development using tools (such as Vagrant and Ruby

gems), some of these in their GitHub, in Locaweb to ensure fast application packaging and reduce deployment rollbacks. In 2013, Marcelo shifted his career focus to become a full-stac developer and began to follow the DevOps movement. In 2014, he attended QConSP- International Software Development Conference as a speaker and spoke about Packer and i use in Locaweb. Currently, Marcelo works as a DevOps engineer at Moip Pagamentos, wher he is responsible for creating continuous deployment solutions, which cover non-PCI or PCI compliance environments. He is currently using Go as a preferable programming language.

He loves playing the guitar and spending time with his beloved wife and his cats, apart from traveling and drinking beer. He can be found on his blog (http://salizzar.net), Twitter (https://twitter.com/salizzar), GitHub (https://github.com/salizzar), and Linkedin (https://www.linkedin.com/in/salizzar).

First, I want to thank my wife for her patience, especially on days when I came home from work, ate something quickly, and went straight to my office, returning only to sleep. Secondly, I want to thank my friends, who believed in my potential since the beginning and kept in touch with me despite the distance, and lastly, my mentors Gleicon Moraes and Roberto Gaiser for the incentive and tips that helped me become a better software engineer.

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit www.PacktPub.com.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at <service@packtpub.com> for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

https://www2.packtpub.com/books/subscription/packtlib

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktL today and view 9 entirely free books. Simply use your login credentials for immediate access

Preface

If you have written software on a desktop computer and attempted to deploy your code to another computer (a server), you have already encountered the challenges presented when deploying software. Developers and administrators frequently struggle with errors and defect when development environments are different from the eventual production machines. There can be a number of differences introduced when the environments are different at the operating system level. Development with desktop operating systems (such as Windows or OS X) can introduce many issues when deploying to production environments that run a Unix (or Linux) environment.

The introduction of desktop hypervisor software allowed developers to develop and test software using virtual machines. A virtual machine is essentially a system within a system, wherein developers working on a desktop operating system can develop and deploy with a copy of the operating system and environment that closely mimics the eventual production environment. When desktop hypervisors became available, development teams found that they could share development environments by sharing the files used by the hypervisors to store the state of virtual machines. In many cases, sharing a virtual machine involved passin around copies of files on a portable hard drive or a shared network folder.

A few years ago, I encountered this specific example when working on a project that involved adding new features to software that ran on an environment, which we could not support with our modern desktop hardware. As many projects reveal, technical debt was introduced to the application by using some very specific features of the Java Development Kit (version 1.5), a environment that was impossible to work on with a 64-bit OS X machine. This machine had dual problems of being a 64-bit machine and it also lacked native support for Java 1.5 XML libraries. The solution to this problem was the creation of a single virtual machine that was shared between developers, passing around a copy of the machine created by a team lead and using it locally to compile and test our modifications.

As time passed by, changes to the environment became an issue, as we began struggling with the differences between not only the development and production environments, but als between our individual development environments as changes were made, making sure that each developer was working on the *latest version* of the virtual machine on that portable hard drive, which soon had a few different versions itself.

Eventually, the problem of maintaining development environments was large enough to begin looking for new solutions. Configuration management approaches helped us to start defining our environment in code, but we still had issues with sharing and maintaining our *base* environment. We found immediate use of an open source project called *Vagrant,* which was gaining some traction.

Vagrant (http://vagrantup.com) is a tool that allows you to define a virtual environment with code. A single file allows you to define a basic environment for a virtual machine as well as a series of provisioning actions that prepare the environment for use. Vagrant works by running

code (Vagrantfiles) on top of packaged operating system images called *boxes*. The Vagrant code and box files can be versioned and distributed using automated tooling. This allows you to share virtual machines, which is not much different than the process of software development that uses source control.

Using Vagrant boxes and provisioning controlled by Vagrantfiles not only simplified the process of distributing virtual machines (and updates to virtual machines), but it also made th virtual machines we were working with *inexpensive* in terms of effort to rebuild. The amazing thing that we found was that Vagrant not only made it simple to distribute virtual machines, but also gave developers more freedom to experiment and make deeper modifications to the code without losing time due to changes in the development environment that could not be rolled back. This flexibility and a simplified on-boarding process for new developers made it much simpler for the team to spend more time doing software development (and tackling tha technical debt!), rather than attempting to fix and find problems due to environments.

I've found Vagrant to be an invaluable tool in my work. I hope that this book can be a valuabl resource for you in getting started with Vagrant, or perhaps, using Vagrant in new and different ways.

What this book covers

Chapter 1, *Setting Up Your Environment*, covers a few basics about hypervisor technology, the installation of Vagrant and VirtualBox, and some simple recipes to get started with Vagra machines.

Chapter 2, *Single Machine Environments*, contains recipes to get started with writing single machine Vagrantfiles, including booting machines, forwarding ports, and customizing the virtual machine environment.

Chapter 3, *Provisioning a Vagrant Environment*, introduces the concept of provisioning Vagrant machines, installing software, and customizing the environment to develop and deploy software. This chapter focuses on using shell (bash) scripting to modify the Vagrant environment.

Chapter 4, *Provisioning With Configuration Management Tools*, contains simple recipes to provision Vagrant machines with four common configuration management tools: Puppet, Chef, Ansible, and Salt. These tools allow easier configuration of machines that have more complex environments. They also allow Vagrant machines to share the same provisioning instructions as other environments.

Chapter 5, *Networked Vagrant Environments*, contains recipes focused on networking Vagrant machines with external hosts and with each other. We cover a few topics from the basics of assigning host entries to networking a cluster of Vagrant machines with Consul.

Chapter 6, *Vagrant in the Cloud*, contains recipes to use Vagrant with cloud providers (specifically, Amazon Web Services and DigitalOcean). It also contains the use of Hashicorp' Atlas tool to share Vagrant environments with remote users.

Chapter 7, *Packaging Vagrant Boxes*, introduces methods to package Vagrant boxes for others to use. Recipes include the packaging of boxes using manual and automated tools an tips to share your box with others on Atlas.

Appendix A, *Vagrant Plugins*, gives a short introduction on how to extend the capabilities of Vagrant by developing plugins.

Appendix B, *A Puppet Development Environment*, expands on the introduction in Chapter 4, *Provisioning With Configuration Management Tools*, to set up a more robust configuration environment to develop Puppet scripts. While the focus is on using Puppet to provision, similar environments can be created to support the configuration management environment your choice.

Appendix C, *Using Docker With Vagrant*, is an introduction to use Vagrant to create, deploy, and test Docker (http://docker.io) containers. This appendix introduces techniques to launch Docker containers with Vagrant as well as build and test a complete Docker environment.

What you need for this book

To use the recipes in this book, you will need:

- A development machine capable of running virtual machines with hypervisor software, such as VirtualBox (http://virtualbox.org) or VMware desktop products (http://vmware.com). You would want to get started with the freely available VirtualBox product and later on purchase the plugin to support VMware desktop products. Keep in mind that you will need a machine that is capable of running both your host operating system and also the guest operating systems that you will be creating with Vagrant. You will also want to ensure that you have enough storage (disk space) for virtual machine files. The disks created by Vagrant machines will typically be approximately the size required to operate the guest operating systems (approximately, 5-20 GB of disk space)
- If you plan on running 64-bit guests, you will also want to ensure that your processor is capable of Intel hardware virtualization (VT-x). In most cases, processors that support 64-bit operating systems already have this support built-in (with some exceptions, such as older Intel Celeron processors). See https://www.virtualbox.org/manual/ch10.html for more background on the requirements for hardware virtualization.
- Using cloud recipes (particularly, recipes involving Amazon Web Services and DigitalOcean) will require accounts with cloud providers. Running the examples might incur charges to your account, so make sure that you understand the financial impacts o running the examples and how to ensure that all created instances have been stopped o terminated to avoid extra charges for the use of computational resources. The recipes in this book are not expensive to run, but they are also not free. Machines that are left running for a period of time could also end up costing more than you had planned on, so make sure that any instance created with Vagrant is eventually destroyed.

Who this book is for

This book is for developers and administrators of nearly all skill levels. Throughout the book, make a general assumption that you are creating Vagrant machines to support the development of other software. Vagrant itself does not become interesting or useful until you use it to support the deployment and development of other software. Vagrant makes it simple to create local environments that mimic production environments and takes advantage of the same provisioning techniques used on production servers. If you have a mature and robust deployment pipeline, Vagrant allows you to reproduce this process on development machines. If you do not have a robust development pipeline, Vagrant can help you begin developing the scripts and processes, making your development and deployment environments more consistent. Consistent environments will help you to reduce the problems associated with the deployment process, which allows you to focus on producing better software.

Sections

In this book, you will find several headings that appear frequently (Getting ready, How to do i How it works, There's more, and See also).

To give clear instructions on how to complete a recipe, we use these sections as follows:

Getting ready

This section tells you what to expect in the recipe, and describes how to set up any software or any preliminary settings required for the recipe.

How to do it…

This section contains the steps required to follow the recipe.

How it works…

This section usually consists of a detailed explanation of what happened in the previous section.

There's more…

This section consists of additional information about the recipe in order to make the reader more knowledgeable about the recipe.

See also

This section provides helpful links to other useful information for the recipe.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "The Vagrant installer will extract, copy files, and add the `vagrant` command to the executable path."

A block of code is set as follows:

```
-rw-------  0 cothomps staff 1960775680 Jul 24 20:42 ./box-disk1.vmdk
-rw-------  0 cothomps staff      12368 Jul 24 20:38 ./box.ovf
-rw-r--r--  0 cothomps staff        505 Jul 24 20:42 ./Vagrantfile
```

When we wish to draw your attention to a particular part of a code block, the relevant lines o items are set in bold:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
VAGRANTFILE_API_VERSION = "2"
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
config.vm.box = "chad-thompson/ubuntu-trusty64-gui"
config.vm.provider "virtualbox" do |vbox|
    vbox.gui = true
  end
end
```

Any command-line input or output is written as follows:

```
vagrant box add http://servername/boxes/environment.box
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "A new installation of VirtualBox will display a welcome message in a window titled **Oracle VM VirtualBox Manager**."

> ## Note
>
> Warnings or important notes appear in a box like this.

> ## Tip

Tips and tricks appear like this.