

Extracted from:

Deploying Rails

Automate, Deploy, Scale, Maintain, and Sleep at Night

This PDF file contains pages extracted from *Deploying Rails*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printer versions; the content is otherwise identical.

Copyright © 2010 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The
Pragmatic
Programmers

Deploying Rails

Automate, Deploy,
Scale, Maintain,
and Sleep at Night



Anthony Burns
and Tom Copeland

Edited by Brian P. Hogan

Deploying Rails

Automate, Deploy, Scale, Maintain, and Sleep at Night

Anthony Burns

Tom Copeland

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at <http://pragprog.com>.

Copyright © 2012 Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Printed in the United States of America.

ISBN-13: 978-1-93435-695-1

Printed on acid-free paper.

Book version: B1.0—January 19, 2012

Preface

Ruby on Rails has taken the web application development world by storm. Those of us who have been writing web apps for a few years remember the good old days when the leading contenders for web programming languages were PHP and Java, with Perl, Smalltalk, and even C++ as fringe choices. Either PHP or Java could get the job done, but millions of lines of legacy code attest to the difficulty of using either of those languages to deliver solid web applications that are easy to evolve.

But Ruby on Rails changed all that. Now thousands of developers around the world are writing and delivering high quality web applications on a regular basis. Lots of people are programming in Ruby. And there are plenty of books, screencasts, and tutorials for almost every aspect of bringing a Rails application into being.

We say “almost every aspect” because there’s one crucial area in which Rails applications are not a joy; that area is deployment. The most elegant Rails application can be brought low by runtime environment issues that make adding new servers an adventure, unexpected downtime a regularity, scaling a difficult task, and frustration a constant. There are good tools out there for deploying, running, monitoring, and measuring Rails applications, but pulling them together into a coherent whole is no small effort.

In a sense, we as Rails developers are spoiled. Since Rails has such excellent conventions and practices, we expect deploying and running a Rails application to be a similarly smooth and easy path. And while there are a few standard components for which most Rails developers will reach when rolling out a new application, there are still plenty of choices to make and decisions which can affect application’s stability.

And that’s why we’ve written this book. After several years of full-time consulting for companies who were writing and fielding Rails applications, we’ve learned a few things about running busy Rails applications in production environments. Throughout this book we’ll explore various aspects of deploying

Rails applications, and we'll review and apply the practices and tools that helped us keep our consulting clients happy by making their Rails applications reliable, predictable, and generally speaking, successful. When you finish reading this book you'll have a firm grasp on what's needed to deploy your application and keep it running. You'll also learn pick up valuable techniques and principles on constructing a production environment that watches for impending problems and alerts you before things go wrong.

Who Should Read This Book?

This book is for Rails developers who, while comfortable with coding in Ruby and using Rails conventions and best practices, may be less sure of how to get a completed Rails application deployed and running on a server. Just as you learned the Rails conventions for structuring an application's code using REST and MVC, you'll now learn how to keep your application faithfully serving hits, how to know when your application needs more capacity, and how to add new resources in a repeatable and efficient manner so you can get back to adding features and fixing bugs.

This book is also for system administrators who are running a Rails application in production for the first time, or for those who have a Rails application or two up and running but would like to improve the runtime environment. You probably already have solid monitoring and metrics systems; this book will help you monitor and measure the important parts of your Rails applications. In addition, you may be familiar with Puppet, the open source system provisioning tool. If so, by the end of this book you'll have a firm grasp on using Puppet and you'll have a solid set of Puppet manifests. Even if you're already using Puppet you may pick up a trick or two from the manifests that we've compiled.

Finally, this book is for project managers who are overseeing a project where the primarily deliverable is a Rails application that performs some business functionality. You can use the major sections in this book as a checklist. There's a chapter on monitoring; what kind of monitoring is being done on your application, and what kind of situations might occur where would you like to trigger some sort of alert? There's a chapter on metrics; what kind of charts and graphs would best tell you how the application is meeting the business' needs? If your application has some basic story for each chapter

in this book you'll know that you're covering the fundamentals of a solid Rails application environment.

What is in the Book?

This book is centered around an example social networking application “MassiveApp”. While MassiveApp may not have taken the world by storm just yet, we're confident that it's going to be a winner, and we want to build a great environment in which MassiveApp can grow and flourish. This book will take us through that journey.

We'll start with [Chapter 2, *Getting Started with Vagrant*, on page ?](#), where we'll learn how to set up our own virtual server with Vagrant, an open-source tool that makes it easy to configure and manage VirtualBox virtual machines.

In [Chapter 3, *Rails on Puppet*, on page ?](#), we'll introduce what's arguably the most popular open source server provisioning tool, Puppet. We'll explain Puppet's goals, organization, built-in capabilities, and syntax, and we'll build Puppet manifests for MassiveApp. This chapter will provide you with a solid grasp on more in-depth Puppet topics such as writing your own Puppet types, using Puppet defines, and getting better visibility into Puppet status with the Puppet Dashboard.

In [Chapter 4, *Basic Capistrano*, on page ?](#) we'll explore the premier Rails deployment utility, Capistrano. We'll build a deployment file for MassiveApp, and we'll describe how Capistrano features like hooks, roles, and custom tasks can make Capistrano a linchpin in your Rails development strategy.

[Chapter 5, *Advanced Capistrano*, on page ?](#) is a deeper dive into more advanced Capistrano topics. We'll make deployments faster, we'll use the Capistrano multistage extension to ease deploying to multiple environments, we'll explore error handling and recovery, and we'll look at capturing output from remote commands. Along the way we'll explain more of the intricacies of Capistrano variables and roles. This chapter will get you even further down the road to Capistrano mastery.

In [Chapter 6, *Monitoring with Nagios*, on page ?](#) we'll look at monitoring principles and how they apply to Rails applications. We'll build a Nagios Puppet module that monitors system, service, and application-level thresholds. We'll build a custom Nagios check that monitors Passenger memory process size, and we'll build a custom Nagios check that's specifically for checking aspects of MassiveApp's data model.

[Chapter 7, *Collecting Metrics with Ganglia*, on page ?](#) will cover the nuts and bolts of gathering metrics around a Rails application, both from an infrastruc-

ture and an application level. We'll install and configure Ganglia and we'll explore the Ganglia plugin ecosystem. Then we'll write a new Ganglia metric collection plugin for collecting MassiveApp user activity.

[Chapter 8, *Maintaining the Application*, on page ?](#) discusses the ongoing care and feeding of a production Rails application. We'll talk about backups, recovering from hardware failures, managing log files, and handling downtime both scheduled and unscheduled. This chapter is devoted to items that might not arise during the first few days that an application is deployed, but will definitely arise as the application weathers the storms of user activity.

[Chapter 9, *Running Rubies with RVM*, on page ?](#) covers the Ruby Version Manager (RVM). RVM is becoming more and more common as a Rails development tool, and it has its uses in the Rails deployment arena as well. We'll cover a few common use cases for RVM as well as some tricky issues that arise due to the way that RVM modifies a program's execution environment.

[Chapter 10, *Special Topics*, on page ?](#) discusses a few topics which don't fit nicely into any of the previous chapters but are nonetheless interesting parts of the Rails deployment ecosystem. You don't need to use these tools in every Rails deployment, but they're good items to be familiar with.

How to Read This Book

If you're new to Rails deployment you can read most of this book straight through. The exception would be [Chapter 5, *Advanced Capistrano*, on page ?](#), which you can come back to once you've gotten your application deployed and have used the basic Capistrano functionality for a while.

If you're an advanced Rails developer you may be more interested in the system administration sections of this book. You can get a working knowledge of Puppet by reading [Chapter 3, *Rails on Puppet*, on page ?](#), and you may pick up some useful monitoring and metrics tips from those chapters. The [Chapter 8, *Maintaining the Application*, on page ?](#) chapter is also worth a read to ensure you have your bases covered with applications that you've already deployed.

If you're a system administrator, you may want to read the Rails-specific parts of [Chapter 6, *Monitoring with Nagios*, on page ?](#) and [Chapter 7, *Collecting Metrics with Ganglia*, on page ?](#) to see how to hook monitoring and metrics tools up to a Rails application. You might also be interested in [Chapter 8, *Maintaining the Application*, on page ?](#) to see what a few solutions to problems that come up with a typical Rails application. If you've got a number of Rails applications running on a single server, you will also be interested in [Chapter](#)

[9, Running Rubies with RVM, on page ?](#) to see how to isolate those applications' runtime environments.

Throughout this book we'll have short "For Future Reference" sections which display a summary of the configuration files and scripts presented in the chapter. You can use these as a quick reference guide to how we recommend using the tools that were discussed; they'll contain the final product without all the explanations.

Tools and Online Resources

Throughout the book we'll be building a set of configuration files. Since we want to keep track of changes to those files, we're storing them in a revision control sytem, and since Git is a popular revision control system we're using that. There's a variety of documentation on Git's home page ¹ and there are some excellent practical exercises on <http://gitready.com/>. Using Git isn't strictly necessary, but you'll want to have it installed to get the most out of the examples.

This book has a companion website ² where we post articles and interviews and anything else that we think would be interesting to folks who are deploying Rails applications. You can also follow us on Twitter at <http://twitter.com/deployingrails/>.

1. FIXME
2. <http://deployingrails.com/>