Deploy Rails BlueBook

AN AUTOMATED, QUICKSTART BOOK TO DEPLOY TODAY

John B. Wyatt IV

Deploy Rails BlueBook 2014 Edition

A Quick Start, Best Practices Guide to Deploy Rails Automatically with Chef Solo!

John B. Wyatt IV

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

This version was published on 2015-10-11

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 John B. Wyatt IV by spreading the word about this book on Twitter!

The suggested hashtag for this book is #deployrailsbluebook.

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=#deployrailsbluebook

To My Mother: Susana E. Wyatt

Contents

1. Introduction

Greetings!

This is a quickstart book designed to guide you through the steps to get an automated deploy to a VPS. The book is geared towards Rails beginners who have never deployed before and would like some hand holding. The first part proceeds step by step to build a working Rails deploy. The second part details how to deploy automatically. If you already have deployed Rails you can skip to the Automation part.

Please note that this guide expects you to use Ubuntu on both the local machine and the server. Ubuntu 12.04 for the server and the local machine can be any recent version. This guide was written and tested on an Ubuntu 13.04 machine. You should be familiar with the terminal, with simple server adminstration, the ruby language and simple ruby developer tasks such as installing gems.

Also...

Typical Legal Disclaimer!

There is no warranty or expectation of warranty of any kind. You use this information at your own risk and are advised to take precautions while you learn the environment.

I call this book a 'Bluebook' because I noticed that a few of the instructions for setting up Rails would break within a year.[1] According to Wikipedia, a Bluebook is like an Almanac. It contains changing facts and data, which seems to be like server setup instructions. My goal with this book, is to produce a new guide every year, updated to the latest Rails stack. Even in the draft stage, this book has proven to be an excellent little quick reference while I was researching the automation scripts. I hope you find it as useful as I have.

Text in this format are commands to be executed on the local machine or the server.

```
echo "Hello, world!"
```

Blocks of code with a filepath are usually are meant to be copied to the computer. Links to github are avaiable for you to copy and paste.

https://gist.github.com/jbwyatt4/7115255

[1] This book already expects you to be able to code a basic Rails app. That means you have read Micheal Hartz's excellent Rails Tutorial at the very least.

foo/bar/hello.rb

```
1 print "Hello, world!"
```

I will do my best to keep this book up to date with working instructions for the current stack until Dec 1 2014 (or when a new edition is released). I hope this gives you an incentive to pay for it. A few bucks, hopefully is a small price to pay for the hours you would save going through different forums/tutorials/references looking for answers. Please email me if the instructions do not work out for you and you are using the correct stack that I cover.

Cover art done by Ernesto Mora[2].

You can purchase this book at:

Leanpub (PDF, mobi, epub)

https://leanpub.com/deployrailsbluebook

Amazon (Kindle)

http://www.amazon.com/Deploy-Rails-BlueBook-2014-Edition-ebook/dp/B00GZ9SNKY

Smashwords (iPad, Nook, Kobo, Oyster, etc)

Coming Soon

Thankyou for reading this book!

[2] http://www.ernestomora.com

2. Setup a VPS Manually

To serve web pages you need to setup a web server and secure it. This involves updating your server, installing the rails specific parts, securing it by denying brute force against the ssh and using least privileged users to run the app. If you missed it from the description this book will use a Ubuntu/Nginx/Rails/Postgres stack.

Feel free to glance at the steps if your more experienced.

The dummy ip I will use is 148.211.114.67 and I gave it the hostname SleepyKitten1303.

```
user@local:~$ ssh root@148.211.114.67
The authenticity of host '148.211.114.67 (148.211.114.67)' can't be established.
ECDSA key fingerprint is 79:95:46:1a:ab:37:11:8e:96:54:36:38:bb:3c:fa:c0.
Are you sure you want to continue connecting (yes/no)? yes
```

That scary message is just warning you it has never connected to this server before.

You should see this. If not make sure you have the correct ip and no spaces.

```
root@148.211.114.67's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-24-virtual i686)

 * Documentation: https://help.ubuntu.com/
```

Now that you are logged in as the root adminstrator you need to secure the vps. Change the root password.

```
root@SleepyKitten1303:~# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Now we add a non root user as a layer of additional security.

```
root@SleepyKitten1303:~# adduser user
```

Adding user ‘user’ …

Adding new group ‘user’ (1000) …

Adding new user `user' (1000) with group` user’ …

Creating home directory ‘/home/user’ …

Copying files from ‘/etc/skel’ …

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

Changing the user information for user

Enter the new value, or press ENTER for the default

Full Name []: Room Number []: Work Phone []: Home Phone []: Other []: Is the information correct? [Y/n] Y

root@SleepyKitten1303:~#

Reload the ssh service and exit.

```
reload ssh
exit
ssh user@148.211.114.67
```

Now to update the system.

```
sudo apt-get update
sudo apt-get upgrade -y
```

You may get an error warning about overriding one of Grub’s files. Choose the default option. It will happen twice. Grub is the boot manager for GNU systems that boots to Linux or other kernels.

```
sudo apt-get dist-upgrade -y
```

Restart the system

```
sudo reboot
```

Wait a minute and log in again.

Now let's secure the SSH port. Malware, viruses, worms… whatever you call them will scan your server's ports for SSH services. If they detect them they will repeatly try to access them with different passwords until they guess the correct one. This is called brute forcing. You can install software called fail2ban to block repeated attempts to access your site.

```
sudo apt-get install fail2ban -y
```

Fail2ban works by keeping track of the ip addresses that try to access your system. By default if an ip address enters an incorrect password 3 times it is banned for one hour.

The default configuration file is at /etc/fail2ban/jail.conf. It is a good idea to copy the configuration file in case you decide to modify it and make a mistake.

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

(Optional) Whenever you make changes to fail2ban (and most services) you need to restart them.

```
sudo service fail2ban restart
```

user@SleepyKitten1303:~$ sudo service fail2ban restart * Restarting authentication failure monitor fail2ban [OK]

Now you actually get to install Ruby and Rails. The default version that ships with Ubuntu is usually horribly out of date. So we will use rvm, a Ruby manager to download and build a recent Ruby for us!

```
curl -L get.rvm.io | bash -s stable
```

You should get a nice flurry of text. One of the more important ones was to 'source', reload a file for your (bash) terminal.

```
source /etc/profile.d/rvm.sh
```

or as a user (Don't copy and paste the text. My book generator formats the ~ copies as a different character):

```
source ~/.profile
```

Now we ask RVM to check our system for the requirements to build Ruby.

```
rvm requirements
```

If you get an error complaining about missing required patches see the Gotcha Appendix.

Now install Ruby 1.9.3

```
rvm install 1.9.3
```

RVM allows mutiple Ruby instances. Even more, it allows mutiple gemsets per each instance. You can find more information on this in the rails book.

For now set 1.9.3 as the default.

```
rvm use 1.9.3 --default
```

You may get this error.

> RVM is not a function, selecting rubies with 'rvm use ...' will not work.
>
> You need to change your terminal emulator preferences to allow login shell.
>
> Sometimes it is required to use `/bin/bash --login` as the command.
>
> Please visit https://rvm.io/integration/gnome-terminal for an example.

Relogin (Or you can simply exit the ssh and log back in)

```
/bin/bash --login
rvm use 1.9.3 --default
```

If you get this message you're good. (or `Using '/home/user/.rvm/gems/ruby-1.9.3-pXXX'` if installing to a user.)

```
Using '/usr/local/rvm/gems/ruby-1.9.3-pXXX'
```

By the time you read this Ruby 2.0 and Rails 4 will be out. However, as a new user you will want to stick with this older, more mature versions until the community documentation catches up with these new releases.

Now tell RVM to handle gems.

```
rvm rubygems current
```

Congratuations. You now have a working Ruby environment on a vps.

3. Setup the Web Server Manually

Now we have a fully working, modern ruby setup. We just don't have a web server, rails or a database up. Let's remedy the first two by using passenger.

Passenger is an application server. You need to use it to allow Nginx to communicate with the Rails app (like mod_php). Otherwise you would need to use the almost obselete cgi (Common Gateway Interface implemented in Nginx) to communicate. For this book I chose Passenger because it's easier for newcomers to understand and get started. Unicorn is an alternate application server that offers 100% uptime. However there are drawbacks. Unicorn takes alot more memory and it can only host one Rails app per instance. Stick with Passenger for now to get confortable and you can switch to Unicorn in the future.

Install passenger with our newly setup rubygems

```
gem install passenger --no-ri --no-rdoc
```

No rdocs Reminder!

`--no-ri --no-rdoc` skips the generation of ruby documentation that you would probably just lookup online anyway.

Now this is the fun part. With RVM and passenger you can automate the installation and (most of the) configuration of nginx.

```
rvmsudo passenger-install-nginx-module
```

Too bad it fails on stock Ubuntu 12.04. It will tell you to use 3 different terminal commands to install the necessary packages. I have condensed them for your convenience.

```
sudo apt-get install build-essential libcurl4-openssl-dev zlib1g-dev -y
rvmsudo passenger-install-nginx-module
```

Choose option 1. Hit enter for the default directory.

Now start nginx because it will not start on its own.

```
sudo service nginx start
```

Now it will give another error. Passenger nginx doesn't come with the nice Debian/Ubuntu scripts. Let's remedy that.

```
wget -O init-deb.sh http://library.linode.com/assets/660-init-deb.sh
sudo mv init-deb.sh /etc/init.d/nginx
sudo chmod +x /etc/init.d/nginx
sudo /usr/sbin/update-rc.d -f nginx defaults
```

Kudos to spavbg from the Digital Ocean commuinity for this.

Now any of these 3 commands will work. Use nginx start.

```
sudo /etc/init.d/nginx stop
sudo /etc/init.d/nginx start
sudo /etc/init.d/nginx restart
```

Type your remote vps's ip address into the broswer url to see your new server's index page.

You now have a functioning webserver. You now just need to configure it and upload your rails app.

Let's open the nginx configration.

```
sudo nano /opt/nginx/conf/nginx.conf
```

There should be a server bracket. Change the text inside to this.

```
server {
  listen 80;
  server_name example.com;
  passenger_enabled on;
  root /var/www/your_rails_app/public;
}
```

Don't get confused by root. It means webroot. Which is the root of the website on your server.

Nginx should now give you an error when you visit your ip address.

Nginx also gives it's version number which is really bad. An attacker could look up exploits for your version of nginx with this info. Let's fix that.

Before the server add this line like so.

```
server_tokens off;
server {
  listen 80;
  server_name example.com;
  passenger_enabled on;
  root /var/www/your_rails_app/public;
}
```

Restart the server

```
sudo /etc/init.d/nginx restart
```

Check again to see the version number was removed.

Now let's finally get passenger and your rails app working.

You need to install a web server within rails to run the app.

```
sudo apt-get install nodejs
```

Create the directory.

```
sudo mkdir /var/www
```

Now for laughs and giggles create a new rails app on the vps.

```
rails new your_rails_app
```

Add some scaffolding.

```
rails g scaffold sales
```

Copy it over to /var/www

```
cp -r your_rails_app /var/www/
```

Go to your server's public page to see your app in production (it won't work yet, but we will get to that).

But what if you want to copy over your own local version to the vps? Like a real developer? We use rsync to accomplish that.

Rsync is a tool intended to automate file uploading. It's used all the time in backups and other server admin tasks. Let's use it to upload the local rails project.

Make sure to exit the vps's ssh.

By default passenger will run an app in production. If you have a real database configuration setup skip ahead to database chapter or revert back to sqlite3 for production for the time being.

Now rsync your app to the vps.

On your local machine, make sure your in the directory that contains the folder your rails app is stored in.

```
rsync -avz -e ssh your_rails_app user@148.211.114.67:/var/www/
```

Rsync may not get the permissions correct. Use the following commands to fix it if you get a 403 error from passenger.

```
sudo chmod -R 755 your_rails_app
sudo chown -R 755 your_rails_app
sudo chown -R user your_rails_app
```

Now ssh back into the vps and cd into /var/www

```
cd your_rails_app
```

Run the bundler.

```
bundle install
```

Try visiting a dynamic part of your website (or /sales of the one created). You should get an error. But what could it be? You can check log/production.log in your rails app to find out.

If you get a table error make sure you didn't forget to migrate the db (for production):

```
rake db:migrate RAILS_ENV="production"
```

If you want to enter the console don't forget to specify the production flag (and we use bundle exec to take care of some rare issues when the gemset is confused).

```
bundle exec rails c production
```

If you see an assets error you need to run a command to compile them.

```
rake assets:precompile
```

If you're wondering, yes, you need to rerun the assets command everytime you upload a new version (specifically whenever you change javascript, css or other assets). Thankfully, we can automate this as told in the automation chapter.

Your website should be working by now.

4. Books by John B. Wyatt IV

4.0.1 Automate PHP Deploys

4.0.2 The PHP Companion Guide for Deploy Rails Bluebook

Are you a PHP programmer looking for a great way to automate your server? Are you a Rails programmer that has PHP applications to maintain? Do you simply want to automate setting up a Wordpress blog? Look no further than this great companion guide to Deploy Rails Bluebook.

In one single book you can find how to setup a Ruby environment for PHP programmers and for Rails programmers you can have Chef automatically setup Nginx, PHP and APC to work together on Ubuntu 12.04.

Get the book from:

Leanpub (PDF, mobi, epub)

http://leanpub.com/automatephpdeploys

Amazon (Kindle)

http://www.amazon.com/Automate-Deploys-2014-John-Wyatt-ebook/dp/B00HTO717A

Smashwords (iPad, Nook, Kobo, Oyster, etc)

Coming Soon

4.0.3 Rails API and Android Guide

Android and Rails, Java and Ruby, learn how to create API's in Rails that can be accessed by a Android mobile app.

Get the book from:

Leanpub (PDF, mobi, epub)

http://leanpub.com/railsapiandandroidbluebook2014

Amazon (Kindle)

Coming Soon

Smashwords (iPad, Nook, Kobo, Oyster, etc)

Coming Soon

5. Appendix - Vagrant

Vagrant is a tool to easily create and manage headless virtual machines for development. It can be used for servers as well but the performance will be poor. One of Vagrant's best features is in testing Chef recipes. All the commands in this book 'could' be used on a Vagrant virtual machine with a few changes, but the purpose of this book is to get confortable with deploying to a real server.

Vagrant is just a wrapper for the headless VirtualBox interface, a full virtualization solution that itself acts as a wrapper for executed CPU instructions. To establish both security and exclusion VirtualBox traps dangerous instructions that could give a virtualized program elevated privileges. This in theory makes it safer to run a virtualized machine. However, widespread solutions like VirtualBox and VMWare are widly tested by Black Hat hackers and are likely to have vunerabilities that can be exploited in certain conditions. The x86/AMD64 instruction set has many obscure and obsolete instructions that adds to the complications of ensuring a secure system. Do not blindly trust virtualization to protect your computer from random programs on the internet.

First, you need to install the Oracle build of VirtualBox to your local machine since the version in the Ubuntu repos is out of date.

Echo to your sources list. Precise is Ubuntu 12.04. Please check virtualbox.org for the latest instructions for your installation (If you use the more recent releases you can skip this step and just install use `sudo apt-get install virtualbox`).

Get the public key:

```
wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -
O- | sudo apt-key add -
```

Add the apt link:

```
sudo bash -c "echo 'deb http://download.virtualbox.org/virtualbox/debian
precise contrib' >> /etc/apt/sources.list"
```

Install VirtualBox on your local machine.

```
sudo apt-get update
sudo apt-get install virtualbox-4.3
```

This package name *will* change in the future. Ubuntu 12.04 has a 5 year support cycle.

Use `sudo apt-cache search "virtualbox"` to find it in the future.

Make sure to reboot to avoid any kernel module errors.

```
ssh-keygen -t rsa
```

and

```
ssh add
```

Download the Vagrant package from vagrantup.com and install it.

Do not install Vagrant through Rubygems. It is no longer supported.

You need to download the server image. Goto Vagrantbox.es[1] and pick out an image. I recommend this Ubuntu precise 64 VirtualBox[2] image but it can become out of date. Either download it manually or through Vagrant.

```
vagrant box add ubuntu1204 http://files.vagrantup.com/precise64.box
```

You can get help on any vagrant command with.

```
vagrant box help
```

Vagrant relies on a configuration file to setup the box.

Create a new directory and enter it.

```
mkdir VM && cd VM
```

Intialize the directory with Vagrant

```
vagrant init
```

Now check the Vagrantfile.

```
nano Vagrantfile
```

There are a wild variety of options to investigate in the future. For now, I recommend deleting all the text and pasting in this.

[1] http://www.vagrantbox.es/

[2] http://files.vagrantup.com/precise32.box

```
Vagrant::Config.run do |config|
  config.vm.define :chefbox do |mconfig|
    mconfig.vm.box = "ubuntu1204"
    mconfig.vm.network :hostonly, "33.33.13.39"
    mconfig.vm.customize ["modifyvm", :id, "--name",
      "chefbox", "--memory", "512"]
  end
end
```

Let's go through each of the options in the inner block. The first sets the name of the image to use (Vagrant will try to download if it hasn't already). You should change this to whatever image you downloaded. The second sets the machine to be accessible only by the host and at the ip address 33.33.13.39. The third option sets the name of the vm and the memory in megabytes.

Once the download is finished create and start the server.

```
vagrant up
```

You can ssh into a server quickly with:

```
vagrant ssh
```

Or you can ssh in by ip like a regular server. 'vagrant' is the user and password.

Stop the server with:

```
vagrant halt
```

To avoid Chef repeatly asking for a password you can use an SSH key. Now where do you get the key for a box downloaded from the internet? Thankfully (in this case), most boxes use the same SSH key which you can fetch from here while in your Chef directory:

```
curl https://raw.github.com/mitchellh/vagrant/master/keys/vagrant > va-
grant.key
chmod 600 vagrant.key
knife solo prepare --identity-file vagrant.key vagrant@33.33.13.39
knife solo cook --identity-file vagrant.key vagrant@33.33.13.39
```

Thanks to Micheal Grosser![3]

There are plenty more options with Vagrant to explore that are outside the scope of this small guide (including having Vagrant automatically use Chef to build the VM).

Accessing Vagrant with SSH

```
ssh -i vagrant.key vagrant@33.33.13.39
```

This part of the book is in the sample edition. You can test the entire recipe with:

git clone https://github.com/jbwyatt4/railsbluebook2014.git

Run `bundle update` in the root railsbluebook folder, then `knife solo prepare` first and then the cook command. I have left the 33.33.13.39.json file for you to use.

If you wish to use this in a production setting make sure to change secret_token in railsbluebook2014/cookbooks/deployserver/templates/nginx.erb and the password defined for the user.

[3] http://grosser.it/2013/02/09/passwordless-ssh-auth-into-your-vagrant-box/