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 2014-07-02

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

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

I Part 1: Manual Deploy

1 Choose a VPS and Domain Registar

First you have to pick a VPS. Obvious, but you may not know all your options. The most well known is Amazon AWS. What you may not know is the closest competitor is Linode which offers better documentation and cheaper prices. Then you have a very easy setup such as Digital Ocean. Finally you have discount VPS specials listed on lowendbox[1].

Digital Ocean as of 2013 March is my favorite. SSD disk drives, one TB of bandwidth, 512 MB of ram; enough to run a Linux Rails server, all for $5 a month. They are very newbie friendly, especially with all of the help guides on their site. Linode has upped their offerings to 2 TB transfer and 8 CPU's for $20 a month to be more competitive since Digital Ocean came on the scene.

Keep in mind the devil's in the details. Both Digital Ocean and Linode do not charge for incoming bandwidth, Amazon does. You have to consider this if you are the target of a DDOS attack. You also have to consider location. Digital Ocean, despite being a Colorado startup, only recently began offering hosting in San Francisco apart from it's main American servers in New York. Linode offers load balancers, making itself comparable to AWS. Discount VPS's vary by their terms and locations. They are often competitive with Linode.

You need to choose a Domain Registar. You should be looking at two items as far as costs go. The domain and the SSL cert.

Namecheap[2] is well known for both customer service and price. Which you can get both on domain names and SSL certificates.

Never have the same company host both your server and your domain name. Hosting companies can close your account for a number of reasons. The legimate reasons include users posting racist content, etc. Or the ridiculous such as being spammed or being DDOS'ed (i.e. taking a large number of resources). Keep your options open!

Enough of the free advertising.

Signup for a VPS and a domain name. With the domain name host setup an A record with the IP address provided by the VPS. It takes 24 hours for the DNS to update. In the meantime we will use that IP address to connect to the server.

[1] www.lowendbox.com

[2] namecheap.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 Secure the Deployed Server

Alas, our little rails app is not secure. If someone got a hold of your git log they could figure out the secret_token buried in your app. The secret_token prevents spoofing attacks (used in the cookie) and is randomly generated with the creation of each new rails app. There is nothing wrong with that, except the secret_token is stored in your rails source code which git records.

To plug this security hole you need to move the secret_token out of the source code and into an environment variable. You *could* set the environment variables with a Bash startup script and the export keyword. That could, well, would, have issues. A better way would be to use figaro gem that this chapter will focus on. A final way would be to add your own code to your app.[1]

Go to your rails app and then go to config/initializers/secret_token.rb

You should see the following line:

```
Your_App::Application.config.secret_token = "XX...XX"
```

Replace the secret_key with:

```
Your_App::Application.config.secret_token = ENV['YOUR_APP_SECRET_TOKEN']
```

Add figaro to your gemfile file

```
gem 'figaro'
```

Initalize figaro

```
rails generate figaro:install
```

This will setup figaro and create an application.yml file in the config folder.

Add the following line and use your own secret_token:

```
YOUR_APP_SECRET_TOKEN: XXX..XXX
```

[1] http://railsapps.github.io/rails-environment-variables.html

If you don't get any 'Something went wrong' errors, you're good.

You can use this technique to set a variety of variables you don't want in the source code. An example is the admin name and password for developing apps or a database username and password, which is what we will do in the next section.

Whenever you store passwords in plaintext it's a security vulnerability. The reason we are moving the secret_token is to take advantage of version control and to make it easier to automate the deploy. For example, you can have the server automatically pull the latest version off a private repo without worrying about each's machine's configuration or exposing your passwords to other developers.

As an optional improvement, you could add the following code to config/initializers/secret_token.rb. See the Gist[2] for the full code.

your_rails_app/config/initializers/secret_token.rb

```
Myapp::Application.config.secret_token = ENV['YOUR_APP_SECRET_TOKEN']
unless ENV['RAILS_ENV'] == "production"
  Myapp::Application.config.secret_token = 'XXX...XXX'
end
```

[2] https://gist.github.com/jbwyatt4/6480532

5 Setup the Database

We will install PostgreSQL for our database. Why? MySQL and PostgreSQL are the two biggest *Free Software* databases on the planet. Why not use MySQL? Half is for political reasons. Oracle, the company that currently develops MySQL, has been locking down the project[1].

The second half is that PostgreSQL has been improving in both features and speed. It is a decent, yet, not very beginner friendly, replacement. If you must use MySQL, use MariaDB. MariaDB is a fork by the founder of MySQL. It has several improvements over MySQL.

Remember, we can not use the current database, SQLite, for concurrency reasons (mutiple users cannot not query the database at once).

Add the pg gem to your gemfile.

```
gem 'pg'
```

Install the database and the development libraries on the server:

```
sudo apt-get install postgresql libpq-dev
```

PostgreSQL will create a postgres user to securly access the psql terminal. Use

```
sudo su
```

to login as the root user and

```
su postgres
```

to login as the postgres user.

Use this command to change the password:

```
sudo passwd postgres
```

Now change the postgres role's password:

[1] http://monty-says.blogspot.com/2011/09/oracle-adding-close-source-extensions.html

```
psql -U postgres -c "ALTER USER postgres WITH ENCRYPTED PASSWORD 'some_-
new_password';"
```

Use psql to access the terminal interface.

```
psql
```

You can use `help` and `\?` to get information on the terminal commands. I will go over some commands but I will not actually use the PostgreSQL terminal later in this book-it is a useful skill to have.

Now before we setup Rails to handle PostgreSQL we need to configure it. By default, and a major fustration for new users, the documentation does not explain how to easily allow password login to the database.

Change the following config file. Note the only difference is the second line (actualy the sixth line) last's attribute is being changed from peer to md5:

/etc/postgresql/9.1/main/pg_hba.conf

```
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres        127.0.0.1/32            md5
#host    replication     postgres        ::1/128                 md5
```

This file governs *roles* in PostgreSQL, the equivelent of users in the database. The first line is for a local connection by the postgres user. The sixth line governs all users except postgres and by default it's set to peer. By default, a *peer* connection does not accept passwords, even if we set a password.

You can use the trust attribute instead of md5 for development machines. Make sure any installation programs do not punch a hole in your firewall for PostgreSQL.

Restart PostgreSQL for the changes to take effect.

```
service postgresql restart
```

or

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

Create a *role* with this single terminal command as the postgres user:

```
psql -U postgres -c "CREATE ROLE deploy LOGIN ENCRYPTED PASSWORD 'role_-
password';"
```

Enter the psql terminal and type

```
postgres=# \du
```

to see if the deploy role was created.

Now create the database that will be owned by the deploy role.

```
createdb -O deploy your_rails_db
```

Check to see if the database was created in psql.

```
postgres=# \l
```

Remember, you could add `gem 'pg'` to your gemfile. That will produce a conflict with the sqlite3 gem. You have two ways of handling this. Comment out sqlite3 or seperate the sqlite3 and pg gems. The latter is shown for your convience.

your_rails_app/Gemfile

```
group :developement, :test do
  gem 'sqlite3'
end

group :production do
  gem 'pg'
end
```

For now on whenever you need to run bundle install on your local development machine and *if* your local development machine does not have PostgreSQL installed, use the following command:

```
bundle install --without production
```

Make sure to configure the database.yml

your_rails_app/config/database.yml

```
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

test:
  adapter: sqlite3
  database: db/test.sqlite3
  pool: 5
  timeout: 5000

production:
  adapter: postgresql
  encoding: unicode
  database: your_rails_db
  pool: 5
  username: deploy
  password: ImAPasswordInTheGitRepo-hackMe
```

Of course, we do need to make sure not to record our password in the git repo. Change the database.yml to use a few variables.

your_rails_app/config/database.yml

```
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

test:
  adapter: sqlite3
  database: db/test.sqlite3
  pool: 5
  timeout: 5000

production:
  adapter: postgresql
  encoding: unicode
  database: <%= ENV['YOUR_RAILS_DB_NAME'] %>
```

```
  pool: 5
  username: <%= ENV['YOUR_ROLE_NAME'] %>
  password: <%= ENV['YOUR_ROLE_PASSWORD'] %>
```

Let's set these variables:

your_rails_app/config/application.yml

```
YOUR_APP_SECRET_TOKEN: XXX..XXX

YOUR_ROLE_NAME: deploy
YOUR_ROLE_PASSWORD: role_password
YOUR_RAILS_DB_NAME: your_rails_db
```

Your application should now be able to run on the database. Try `rails db:migrate` to test it.

II Part 2: Automatic Deploy

6 Introducing Chef

Chef is a tool that allows us to setup and configure machines. Chef can configure your local machine, a remote machine or can manage a fleet of servers.

The real downside of Chef is the documentation is terrible for the solo features. Opscode, the company that manages Chef, makes money from enterprise deployments and Chef solo, which runs only a single server which I imagine doesn't rake in as much cash. A major reason behind writing this book was to record all the little things I had to do to setup a server and use Chef that wasn't in one, nice convient place instead of being scattered across the internet in tutorials, references, etc.

6.1 Terminology

Let's get started with some terminology.

- Node - a computer where Chef client will run the cookbooks (web server, database server or another server).
- Chef Client - a command-line tool that configures the servers.
- Chef Server - a server that controls the Chef clients
- Chef Solo - a version of Chef that we use with knife solo. This version does not rely on a server or clients to manage recipes and keys.
- Recipes - Ruby code that configures a node (nginx ssl module, apache php module).
- Resources - a node's resources include files, directories, users and services that will be set on the main machine.
- Cookbook - a collection of Chef recipes (nginx cookbook, postgresql cookbook).
- Role - reusable configuration for multiple nodes-often a collection of recipes and attributes (web role, database role, etc).
- Attribute - variables that are passed through Chef and used in recipes and templates (like the version number of Rails to install).
- Template - a file with Ruby variable blocks to be filled in by logic or attributes, used to create and set configuration files like a Rails Erb view.

You can read more in the Glossary[1].

[1] https://wiki.opscode.com/display/chef/Glossary

6.2 Password-less Login

One of the most irriating things about Chef is that it will not allow you to communicate with the server once you start a Chef run. If you haven't setup password-less login Chef will repeatily prompt you for a password during it's run. Let's take care of that now.

If you're using Vagrant, switch over to the Vagrant Appendix chapter to get the key.

If you have never generated a public-private key pair before on your local machine (use this if your key has a passphrase):

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

You will be asked where to save the key. Hit enter for the default:

```
Enter file in which to save the key (/home/user/.ssh/id_rsa):
```

Now you will be asked for a passphrase-a password to use the public key. Since Chef won't allow us to enter a passphrase hit enter for no passphrase.

```
Enter passphrase (empty for no passphrase):
```

You may get this error:

```
Agent admitted failure to sign using the key
```

That means your local key is not yet managed by ssh-agent.

Add the key with this command:

```
ssh-add
```

You will have two keys created in `~/.ssh/` , `id_rsa` and `id_rsa.pub`

Now upload your public key to root:

```
cat ~/.ssh/id_rsa.pub | ssh root@148.211.114.67 "cat >> /root/.ssh/authorized_keys"
```

If you get:

Key-Agent

```
bash: /root/.ssh/authorized_keys: No such file or directory
```

Login as root and create the directory (do not have root create the directory in user's folder):

```
mkdir .ssh && touch .ssh/authorized_keys
```

Your VPS should be setup.

Add your public key to your VPS

Some VPS's allow you to upload your public key through their admin console to be automatically store on any new servers. Remember, by default, your public key is in `~/.ssh/id_rsa.pub`

(Optional) Disable the Password!

You can now disable password login for SSH if you want to, which means people cannot brute force your password. Goto `/etc/ssh/sshd_config` and change PermitRootLogin to `PermitRootLogin No`. Make sure to keep your private key backed up-if you lose your private key you can not login… ever!

6.3 Install Chef

We will use a gemfile in a local directory to install Chef and Knife Solo. While I often use the word Chef to describe the program we will be using Knife most of the time in the terminal. Knife is a tool that can handle bootstrapping (installing Ruby/Chef, interfacing with Amazon, Digital Ocean, etc) and runs Chef Solo.

```
mkdir firstcookout && cd firstcookout
```

Create a Gemfile and add the following to it:

```
touch Gemfile && nano Gemfile
```

https://gist.github.com/jbwyatt4/6664617

Gemfile

```
source "https://rubygems.org"

gem 'chef', '10.26'
gem 'knife-solo'
gem 'knife-solo_data_bag'
gem 'ffi'
gem 'vagrant'
gem 'multi_json'

gem 'ruby-shadow'
gem 'net-ssh'
gem 'net-ssh-multi'
gem 'fog'
gem 'highline'
```

Install the gems:

```
bundle
```

6.4 Your First Cookout!

Let's start out with Chef by intializing it in the cheffolder directory.

```
knife solo init .
```

6.4.1 Configure Knife

Now before we get started Chef will generate any recipes with the following header.

The default header in a recipe

```
#
# Cookbook Name:: somerecipe
# Recipe:: default
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#
```

No one likes this template. Let's customize your Chef install to fix this.

At the root of the cookbook is the .chef folder with a knife configuration file.

```
cd .chef && nano knife.rb
```

Add the following text to your knife.rb file to give cookbook an Apache license. Customize to how you see fit.

https://gist.github.com/jbwyatt4/6665119

firstcookout/.chef/knife.rb

```
# Already generated text

cookbook_copyright "John B. Wyatt IV"
cookbook_license   "apachev2" # If you want an Free Software license.
                              # Otherwise, just delete this line
cookbook_email     "deployrailsbluebook@jbwyatt4.com"
```

Now create a cookbook called setupserver in the cookbooks folder (run the command in the firstcookout folder):

```
knife cookbook create deployserver -o cookbooks
```

Check out `firstcookout/cookbooks/deployrails/recipes/default.rb` for the beautiful license template. (Also checkout the metadata.rb in recipes.)

Now we need to prepare the server.

Shut down your old server and create a brand new server (with the above ssh-key instructions).

Use this command to install chef and a makeshift version of ruby on the system:

```
knife solo prepare root@148.211.114.67
```

If you already have a domain name setup; you can use that instead of an IP.

Chef may give you a Host key mismatch error if you previously connected to the same ip, but recreated the VPS.

Try ssh root@your_ip_address, use the SSH removal command and try to SSH again.

You will see a script installing Ruby/Chef onto the server.

The command will end with a new file being created in nodes/148.211.114.67.json

firstcookout/nodes/148.211.114.67.json

```
1 {"run_list":[]}
```

Change the file to:

firstcookout/nodes/148.211.114.67.json

```
1 {"run_list":["recipe[deployrails]"]}
```

`recipe[deployrails]` is the same as pointing directly towards the default file: `recipe[deployrails::default]`

Run this command to test out Chef:

Make sure you installed the SSH-key before you run this command!

```
knife solo cook root@148.211.114.67
```

It shouldn't do anything because you have nothing in your recipe.

Let's change that-from the manual section the first thing we do is update apt's cache.

firstcookout/cookbooks/deployrails/recipes/default.rb

```
execute "apt-get update" do
  user "root"
end
```

This is how Chef takes advantage of Ruby's DSL: Domain Specific Language. We have a block with a keyword and a top element and elements within the block describe further parts. For example, we want the command to be executed as root so we specify the subelement `user "root"`. By default with the execute keyword the top element will be the command we execute.

firstcookout/cookbooks/deployrails/recipes/default.rb

```
execute "update apt" do # this becomes just a name with command
  command 'apt-get update'
  user "root"
end
```

In the above case we use the top element as a title and specify the command to be executed as an element in the block.

Chef will NOT use more than two usages of the command keyword in a single block. The last command will be the one executed. You can combine commands in a single 'command' keyword instance using Bash syntax, but you want to be able to catch a command when it fails. The last command exit code will be reported to Chef which will report it as a success or failure.

Now this will nicely update all the packages, but apt-get update is very slow. When you are developing a Chef recipe do you really want to recheck the same package repos over and over again? No, let's change that by making apt-get wait 24 hours:

firstcookout/cookbooks/deployrails/recipes/default.rb

```
execute "update apt" do # this becomes just a name with command
  command 'apt-get update'
  user "root"
  ignore_failure true # always a chance that it fails to connect with server
  only_if do
    File.exists?('/var/lib/apt/periodic/update-success-stamp') &&
    File.mtime('/var/lib/apt/periodic/update-success-stamp') < Time.now - 86400
  end
end
```

This demonstrates how to use logic in Chef's DSL. `only_if` and `not_if` allows us to set conditionals if a block executes or not. Normally, Chef will run each block one by one, from top to bottom, unless it's told not to by conditionals, actions such as `action :nothing` or referenced by the node's json file.

Now there is a problem with the above. Ubuntu 12.04 is not set to create the update timestamp file referenced above. We have to tell it to by creating a configuration file. This serves as an excellent way to introduce templates.

firstcookout/cookbooks/deployrails/recipes/default.rb

```
template "/etc/apt/apt.conf.d/15update-stamp" do
  source "15update-stamp"
  mode 0440
  owner "root"
  group "root"
end

execute "update apt" do # this becomes just a name with cmd
  command 'apt-get update'
  user "root"
  ignore_failure true
  not_if do
    File.exists?('/var/lib/apt/periodic/update-success-stamp') &&
    File.mtime('/var/lib/apt/periodic/update-success-stamp') > Time.now - 86400
  end
end
```

Templates allow you to use erb files to create files on the server based on data in the attributes. Although in this case we just use it as a simple way to copy a system file.

https://gist.github.com/jbwyatt4/7037805

firstcookout/cookbooks/deployrails/templates/default/15update-stamp

```
APT::Update::Post-Invoke-Success {"touch /var/lib/apt/periodic/update-success-sta\
mp 2>/dev/null || true";};
```

The server needs to be restarted for apt to generate update timestamps. However, we only want it to restart the server once. We solve this by creating a file with touch in /etc to mark a milestone in the Chef recipe. Since it's a Bash command we can seperate the command with a semicolon and put in restart.

https://gist.github.com/jbwyatt4/6804692

firstcookout/cookbooks/deployrails/recipes/default.rb

```
1  template "/etc/apt/apt.conf.d/15update-stamp" do
2    source "15update-stamp"
3    mode 0440
4    owner "root"
5    group "root"
6  end
7
8  execute "update apt" do # this becomes just a name with cmd
9    command 'apt-get update'
10   user "root"
11   ignore_failure true
12   not_if do
13     File.exists?('/var/lib/apt/periodic/update-success-stamp') &&
14     File.mtime('/var/lib/apt/periodic/update-success-stamp') > Time.now - 86400
15   end
16 end
17
18 execute "Restart after first time" do
19   command "touch /etc/chefflag-reboot; reboot"
20   user "root"
21   not_if do
22     ::File.exists?('/etc/chefflag-reboot')
23   end
24 end
```

Let's add apt-get upgrade, dist-upgrade and autoremove for the complete set. The upgrade command was the most annoying since Grub needed special commands to be sent to Dpkg (Debian's package manager) to select yes to the package manager version of Grub. See [this link]http://askubuntu.

com/questions/146921/how-do-i-apt-get-y-dist-upgrade-without-a-grub-config-prompt for more details about the command.

https://gist.github.com/jbwyatt4/6804913

firstcookout/cookbooks/deployrails/recipes/default.rb

```
template "/etc/apt/apt.conf.d/15update-stamp" do
  source "15update-stamp"
  mode 0440
  owner "root"
  group "root"
end

execute "update apt" do # this becomes just a name with cmd
  command 'apt-get update'
  user "root"
  ignore_failure true
  not_if do
    File.exists?('/var/lib/apt/periodic/update-success-stamp') &&
    File.mtime('/var/lib/apt/periodic/update-success-stamp') > Time.now - 86400
  end
end

execute 'sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--for\
ce-confdef" -o Dpkg::Options::="--force-confold" upgrade' do
  user "root"
end

execute 'sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--for\
ce-confdef" -o Dpkg::Options::="--force-confold" dist-upgrade' do
  user "root"
end

execute "apt-get autoremove -y" do
  user "root"
end

execute "Restart after first time" do
  command "touch /etc/chefflag-reboot; reboot"
  user "root"
  not_if do
    ::File.exists?('/etc/chefflag-reboot')
  end
```

```
38 end
```

Install fail2ban with the package command which will determine the proper package manager on your system. Use an execute block to copy the configuration file. You can use a template after this to set the contents of the config file if you modified it. Also add the Ubuntu Firewall configuration.

https://gist.github.com/jbwyatt4/6804918

firstcookout/cookbooks/deployrails/recipes/default.rb

```
1  template "/etc/apt/apt.conf.d/15update-stamp" do
2    source "15update-stamp"
3    mode 0440
4    owner "root"
5    group "root"
6  end
7
8  execute "update apt" do # this becomes just a name with cmd
9    command 'apt-get update'
10   user "root"
11   ignore_failure true
12   not_if do
13     File.exists?('/var/lib/apt/periodic/update-success-stamp') &&
14     File.mtime('/var/lib/apt/periodic/update-success-stamp') > Time.now - 86400
15   end
16 end
17
18 execute 'sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--for\
19 ce-confdef" -o Dpkg::Options::="--force-confold" upgrade' do
20   user "root"
21 end
22
23 execute 'sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--for\
24 ce-confdef" -o Dpkg::Options::="--force-confold" dist-upgrade' do
25   user "root"
26 end
27
28 execute "apt-get autoremove -y" do
29   user "root"
30 end
31
32 package "fail2ban"
33
```

```
34 execute "fail2ban" do
35   command 'cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local'
36   not_if do
37     ::File.exists?("/etc/fail2ban/jail.local")
38   end
39   user "root"
40 end
41
42 execute "Setup the Firewall" do
43   command "ufw allow ssh; ufw allow 80; ufw allow 443; ufw enable; touch /etc/che\
44 fflag-ufw"
45   user "root"
46   not_if do
47     ::File.exists?('/etc/chefflag-ufw')
48   end
49 end
50
51 execute "Restart after first time" do
52   command "touch /etc/chefflag-reboot; reboot"
53   user "root"
54   not_if do
55     ::File.exists?('/etc/chefflag-reboot')
56   end
57 end
```

Now we will add the user and add a SSH public key with a template. From the Chef documentation[2] Chef needs a hashed password to be set. Chef will not accept a plaintext password. Use the following command to set it. Please read the documentation for information on the other items in the user DSL.

`openssl passwd -1 "password"` outputs: `$1$.MZ8xPWB$/e/nAWc4C2zidbSVN9M/2/`

Change password to whatever you like and rehash.

https://gist.github.com/jbwyatt4/6805036

[2] http://docs.opscode.com/resource_user.html

firstcookout/cookbooks/deployrails/recipes/default.rb

```
template "/etc/apt/apt.conf.d/15update-stamp" do
  source "15update-stamp"
  mode 0440
  owner "root"
  group "root"
end

execute "update and upgrade" do # this becomes just a name with cmd
  command 'apt-get update'
  user "root"
  ignore_failure true
  not_if do
    File.exists?('/var/lib/apt/periodic/update-success-stamp') &&
    File.mtime('/var/lib/apt/periodic/update-success-stamp') > Time.now - 86400
  end
end

execute 'DEBIAN_FRONTEND=noninteractive apt-get upgrade -y -o Dpkg::Options::="--\
force-confdef" -o Dpkg::Options::="--force-confold" dist-upgrade' do
  user "root"
end

execute "apt-get dist-upgrade -y" do
  user "root"
end

execute "apt-get autoremove -y" do
  user "root"
end

package "fail2ban"

execute "fail2ban" do
  command 'cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local'
  not_if do
    ::File.exists?("/etc/fail2ban/jail.local")
  end
  user "root"
end

execute "Setup the Firewall" do
```

```
42    command "ufw allow ssh; ufw allow 80; ufw allow 443; ufw enable; touch /etc/che\
43  fflag-ufw"
44    user "root"
45    not_if do
46      ::File.exists?('/etc/chefflag-ufw')
47    end
48  end
49
50  user "user" do
51    supports :manage_home => true
52    comment "Our user!"
53    uid 1235
54    gid "users"
55    home "/home/user"
56    shell "/bin/bash"
57    password "$1$.MZ8xPWB$/e/nAWc4C2zidbSVN9M/2/" #password - must be hashed
58  end
59
60  directory "/home/user/.ssh" do
61    owner "user"
62  end
63
64  template "/home/user/.ssh/authorized_keys" do
65    source "authorized_keys"
66    mode 0400
67    owner "user"
68  end
```

This is now a fully secured SSH server. Now we just need to install Ruby, Nginx and the database.

7 Writing a Production Recipe

7.1 Breakup your Recipe

Now before I show you the code to setup Ruby-which is quite tricky-let's use a seperate file and modualize the recipe.

Add the following line before the restart block:

```
require "deployserver::rubyinstall.rb"
```

and add the following file to your deployrails cookbook right next to `default.rb`.

```
touch rubyinstall.rb
```

You don't have to go this route. You could instead move the restart command to its own file and change the code in the node's json file to look like this:

https://gist.github.com/jbwyatt4/7037400

firstcookout/nodes/148.211.114.67.json

```
{"run_list":["recipe[deployrails::default, deployrails::installruby, deployrails:\
:restart]"]}
```

Now copy the apt_package commands into rubyinstall.rb. apt_package is as you might guess the Debian/Ubuntu specific package commands that work with apt to install. The only particular reason why I am using apt_package over package is that some package names in Ubuntu/Debian are different than on Fedora (git-core for example) or don't exist (build-essential).

https://gist.github.com/jbwyatt4/7037463

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
apt_package "nodejs"
apt_package "curl"
apt_package "git-core"
apt_package "build-essential"
apt_package "libcurl4-openssl-dev"
apt_package "zlib1g-dev"
```

7.2 Install Ruby

There are two versions of Ruby in a Chef install. The Chef's version of Ruby installed in the /opt directory and the version actually recognized by the user's shell. By default Chef will use it's own, which is bad, especially since it's often out of date!

Let's install our own version which simply fetches the script (don't run Chef yet):

https://gist.github.com/jbwyatt4/7037498

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

execute "curl -L get.rvm.io | bash -s stable --ruby=1.9.3-p448" do
  not_if do
    ::File.directory?("/usr/local/rvm")
  end
  user "root"
end
```

Since we will install RVM as root it will be installed system wide. This time we check if a directory has been created before we install.

Now what happens when a new verson of Ruby comes out? We will need to use the version number and patch number mutiple times in this cookbook. We need to use variables which in Chef are set through Attributes. Notice I already have quite a few variables set in the double array-I will use these later in the book.

https://gist.github.com/jbwyatt4/7037581

firstcookout/cookbooks/deployrails/attributes/default.rb

```
#--------IMPORTANT! CHANGE OR OVERRIDE-----------
default[:bluebook][:user_password] = "$1$.MZ8xPWB$/e/nAWc4C2zidbSVN9M/2/" #passwo\
rd
default[:bluebook][:postgres_password] = "$1$.MZ8xPWB$/e/nAWc4C2zidbSVN9M/2/" #pa\
ssword
default[:bluebook][:db_role_postgres_password] = "postgres_role_password"
default[:bluebook][:db_role_deploy_password] = "role_password"
default[:bluebook][:app1_secret_token] = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\
XXXXXXXXX"
# Vagrant public key-please, please, please, change this or set to empty string.
default[:bluebook][:authorized_keys] = "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8i\
allvQVp22WDkTkyrtvp9eWW6A8YVr+kz4TjGYe7gHzIw+niNltGEFHzD8+v1I2YJ6oXevct1YeS0o9HZy\
N1Q9qgCgzUFtdOKLv6IedplqoPkcmF0aYet2PkEDo3MlTBckFXPITAMzF8dJSIFo9D8HfdOV0IAdx4O7P\
tixWKn5y2hMNG0zQPyUecp4pzC6kivAIhyfHilFR61RGL+GPXQ2MWZWFYbAGjyiYJnAmCP3NOTd0jMZEn\
DkbUvxhMmBYSdETk1rRgm+R4LOzFUGaHqHDLKLX+FIPKcF96hrucXzcWyLbIbEgE98OHlnVYCzRdK8jlq\
m8tehUc9c9WhQ== vagrant insecure public key
"
#-----------------

# Versions
default['bluebook']['ruby_version'] = '1.9.3'
default['bluebook']['package_version'] = 'p484'
default['bluebook']['rails_version'] = '3.2.16'
default['bluebook']['passenger_version'] = '4.0.29'

# Locations and other settings
default['bluebook']['combined'] = "#{default['bluebook']['ruby_version']}-#{defau\
lt['bluebook']['package_version']}"
default['bluebook']['gem_binary'] = "/usr/local/rvm/bin/gem-ruby-#{node.default['\
bluebook']['combined']}"

#default['bluebook']['source'] = "source /usr/local/rvm/scripts/rvm"
default['bluebook']['source'] = "source /etc/profile.d/rvm.sh"
default['bluebook']['gem_options'] = "--no-ri --no-rdoc"

default['bluebook']['app1'] = "your_rails_app"
default['bluebook']['webroot'] = "/var/www/"
# The current directory is used by Mina to link to the current version of the web\
 app.
default['bluebook']['total_url_app1'] = "#{default['bluebook']['webroot']}#{defau\
```

```
42 lt['bluebook']['app1']}/current"
43
44 default[:bluebook][:db_role_deploy_username] = "deploy"
45 default[:bluebook][:db_role_deploy_database] = "your_rails_db"
```

Now you can use some actual Ruby to set the version numbers like so:

https://gist.github.com/jbwyatt4/7039656

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

execute "curl -L get.rvm.io | bash -s stable --ruby=#{default['bluebook']['ruby_v\
ersion']}-#{default['bluebook']['package_version']}" do
  not_if do
    ::File.directory?("/usr/local/rvm")
  end
  user "root"
end
```

Or you can make it even nicer by combining them into one variable:

https://gist.github.com/jbwyatt4/7039689

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

execute "curl -L get.rvm.io | bash -s stable --ruby=#{node.default['bluebook']['c\
ombined']}" do
  not_if do
    ::File.directory?("/usr/local/rvm")
  end
  user "root"
end
```

Run the above, you now need to source the Ruby install so Bash knows where to look for Ruby.

https://gist.github.com/jbwyatt4/7051976

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

execute "source" do
  command "echo 'source /usr/local/rvm/scripts/rvm' >> /etc/bash.bashrc; echo 'ex\
port rvmsudo_secure_path=0' >> /etc/bash.bashrc; source /usr/local/rvm/scripts/rv\
m; rvm use #{node.default['bluebook']['combined']} --default; /bin/bash --login; \
touch /etc/chefflag-source"
  not_if do
    ::File.exists?('/etc/chefflag-source')
  end
  user "root"
end
```

This command is complex. First we echo an additional line to the /etc/bash.bashrc file, the system wide Bash configuration. We echo a second additional line to the same file to take care of a RVM warning on Ubuntu. These two lines will setup the machine from a login or a restart will load RVM's Ruby automatically when we actually login to the server, but that does not help us during our Chef install. The next command manually sources RVM's Ruby as a one time command. The next line as you know sets the default version of Ruby. The next line reintializes Bash. The last line sets a flag for our recipe.

Let's make sure we are `Not` using Chef's version of Ruby; paste in the following command. Notice, I have to source again to gain access to the system's Ruby install. Your version of Ruby should print out the version specified in the attributes.

https://gist.github.com/jbwyatt4/7051993

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

execute "source" do
  command "echo 'source /usr/local/rvm/scripts/rvm' >> /etc/bash.bashrc; echo 'ex\
port rvmsudo_secure_path=0' >> /etc/bash.bashrc; source /usr/local/rvm/scripts/rv\
m; rvm use #{node.default['bluebook']['combined']} --default; /bin/bash --login; \
touch /etc/chefflag-source"
  not_if do
    ::File.exists?('/etc/chefflag-source')
  end
  user "root"
end
```

```
execute "ruby -v" do
  command "bash -c 'source /usr/local/rvm/scripts/rvm; ruby -v'"
  user "root"
end
```

Use variables to further cleanup the code.

https://gist.github.com/jbwyatt4/7052003

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

execute "source" do
  command "echo '#{node.default['bluebook']['source']}' >> /etc/bash.bashrc; echo\
 'export rvmsudo_secure_path=0' >> /etc/bash.bashrc; #{node.default['bluebook']['\
source']}; rvm use #{node.default['bluebook']['combined']} --default; /bin/bash -\
-login; touch /etc/chefflag-source"
  not_if do
    ::File.exists?('/etc/chefflag-source')
  end
  user "root"
end

execute "ruby -v" do
  command "bash -c '#{node.default['bluebook']['source']}; ruby -v'"
  user "root"
end
```

We could download a cookbook to install Ruby instead of this hackish method. I chose this way because I didn't want to be reliant on other people's recipes to setup my own server. The recipes on opscode.com tend to be very difficult to get into for new Chef developers. RVM's recipe actually overrides part of Chef to install Ruby and will likely need to be changed with major point changes to Chef.

7.3 Install the Web Server and Rails

The gem_package part is no longer used as of April 2014 due to a problem with RVM and the latest versions of Ruby. This code has been left for as an example. You are recommended to use the immediate code below, instead. Rails will be installed regardless because of the Gemfile in the later sample app.

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
execute "Install Passenger" do
  command "bash -c '#{node.default['bluebook']['source']}; gem install passenger \
-v #{node.default['bluebook']['passenger_version']} #{node.default['bluebook']['g\
em_options']}'"
  user "root"
end
```

Now to install the gems, we need to state the gem_binary directory to install like so:

How to install a gem into the system Ruby environment

```
gem_package "passenger" do
  package_name "passenger"
  version "0.0.0.1"
  gem_binary "/usr/local/rvm/bin/gem-ruby-#{node.default['bluebook']['combined']}"
  options "--no-ri --no-rdoc"
end
```

We add options to prevent the rdocs from being generated.

This an example of using the attributes with package to install the Passenger and Rails:

https://gist.github.com/jbwyatt4/7052102

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

gem_package "passenger" do
  package_name "passenger"
  version node.default['bluebook']['passenger_version']
  gem_binary node.default['bluebook']['gem_binary']
  options node.default['bluebook']['gem_options']
end

gem_package "rails" do
  package_name "rails"
  version node.default['bluebook']['rails_version']
  gem_binary node.default['bluebook']['gem_binary']
  options node.default['bluebook']['gem_options']
end
```

Now we issue the command for Passenger to install Nginx:

https://gist.github.com/jbwyatt4/7052045

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

execute "Use Passenger to Install Nginx" do
  command "bash -c 'source /usr/local/rvm/scripts/rvm; passenger-install-nginx-mo\
dule --auto --auto-download --prefix=/opt/nginx'"
  user "root"
  not_if do
    ::File.directory?("/opt/nginx")
  end
end
```

The second command calls passenger with their automation commands which all three have to specified. Passenger will compile a custom version of Nginx which will take quite a long coffee break.

The following command that fetches the init.d scripts isn't any different than the manual chapter.

https://gist.github.com/jbwyatt4/7052128

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

execute "Fetch Init Scripts" do
  command "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; touch /etc/chefflag-fetch"
  user "root"
  not_if do
    ::File.exists?('/etc/chefflag-fetch')
  end
end
```

Next setup and restart the Nginx server. Here is the template for the nginx.conf:

https://gist.github.com/jbwyatt4/7038041

<<firstcookout/cookbooks/deployrails/templates/nginx.conf.erb[1]

and the actual code:

https://gist.github.com/jbwyatt4/7052132

[1] https://gist.github.com/jbwyatt4/7038041

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

template "/opt/nginx/conf/nginx.conf" do
  source "nginx.conf.erb"
  mode 0440
  owner "root"
  group "root"
  notifies :run, 'execute[nginx-restart]', :immediately
end

template "/etc/default/nginx" do
  source "nginx.erb"
  mode 0440
  owner "root"
  group "root"
  notifies :run, 'execute[nginx-restart]', :immediately
end

execute "nginx-restart" do
  command "/etc/init.d/nginx restart; true" # we add true to end to return 0
                                            # since this script keeps
                                            # returning 1
  user "root"
  action :nothing
end
```

So your server now works and is configured.

7.4 Install the Database

Installing PostgreSQL with Chef is pretty straight forward.

Create a dbinstall.rb int he recipes folder and the following line to the default.rb file; right after `include_recipe "deployserver::rubyinstall"`

```
include_recipe "deployserver::dbinstall"
```

Now install the software. Unlike MySQL, Postgres's installation is relatively painless. MySQL's installation requires you to use dpkg to override the installation script. For an example see this.[2]

[2] https://github.com/opscode-cookbooks/mysql/blob/master/recipes/_server_debian.rb

firstcookout/cookbooks/deployrails/recipes/binstall.rb

```
1 apt_package "postgresql"
2 apt_package "libpq-dev"
```

Set the template file:

firstcookout/cookbooks/deployrails/recipes/binstall.rb

```
{Above Code Not Shown}

template "/etc/postgresql/9.1/main/pg_hba.conf" do
  source "pg_hba.conf.erb"
  mode 00600
  owner "postgres"
  group "postgres"
  # We need to restart postgresql
  notifies :run, 'execute[postgresql-restart]', :immediately
end
```

Change the postgres user's password and the postgres's role password.

https://gist.github.com/jbwyatt4/7543348

firstcookout/cookbooks/deployrails/recipes/binstall.rb

```
{Above Code Not Shown}

# Change the automatically created postgres user's password
user "postgres" do
  password node.default[:bluebook][:postgres_password]
  action :modify
end

# Now change the postgres role password
execute "Change postgres role password" do
  command "psql -U postgres -c \"ALTER USER postgres WITH ENCRYPTED PASSWORD '#{n\
ode.default[:bluebook][:db_role_postgres_password]}';\""
  user "postgres"
end
```

Now setting up the role and creating the database is tricker. We need to check first if the role or database has been created or we will get an error. We can do this by querying with psql tool. We can query the psql with a block of code being passed to the command line. You need to specify the user a second time or the block will be passed to a different user and the command will trigger everytime.

https://gist.github.com/jbwyatt4/7543375

firstcookout/cookbooks/deployrails/recipes/binstall.rb

```
{Above Code Not Shown}

# You should create a new role for each new database
# As an exercise for the reader try to DRY out creating the role and the db
execute "Setup deploy role" do
  # I'm a block of code!
  check = <<-EOH
  psql -U postgres -c "select * from pg_roles where rolname='#{node.default[:blue\
book][:db_role_deploy_username]}'" | grep -cw #{node.default[:bluebook][:db_role_\
deploy_username]}
  EOH
  # End block.
  user "postgres"
  command "psql -U postgres -c \"CREATE ROLE #{node.default[:bluebook][:db_role_d\
eploy_username]} LOGIN ENCRYPTED PASSWORD \'#{node.default[:bluebook][:db_role_de\
ploy_password]}\';\"" # postgres -c does not like '' and you need to use "" to fi\
lter.
  not_if check, :user => 'postgres' # Without this :user part the command will be\
 ignored
end

execute "Setup your_rails_db database" do
  checked = <<-EOH
  psql -U postgres -c "select * from pg_database WHERE datname='#{node.default[:b\
luebook][:db_role_deploy_database]}'" | grep -cw #{node.default[:bluebook][:db_ro\
le_deploy_database]}
  EOH
  user "postgres"
  command "createdb -O #{node.default[:bluebook][:db_role_deploy_username]} #{nod\
e.default[:bluebook][:db_role_deploy_database]}"
  not_if checked, :user => 'postgres' # Without this :user part the command will \
be ignored
  notifies :run, 'execute[postgresql-restart]', :immediately
end
```

The actual command that is passed is an sql query that is filtered by grep. -c returns 1 on a successful match. -w requires the whole word to match.

Finally, write a command to restart the server, but do not activate it with `action :nothing`.

firstcookout/cookbooks/deployrails/recipes/binstall.rb

```
{Above Code Not Shown}

execute "postgresql-restart" do
  command "service postgresql reload"
  user "root"
  action :nothing
end
```

One last note for the chapter. Write the application.yml template:

https://gist.github.com/jbwyatt4/7540521

7.5 Deploy your_rails_app the Simple Way

Let's start with a simplistic deploy similar to the manual deploy in Part 1.

Create the webroot directory and instead of copying your webapp from your local machine we will instead pull from a public git repository. This is a very useful practice when you deploy mutiple servers. A sample one at my github[3] is used. We only need to do this once.

https://gist.github.com/jbwyatt4/7052191

firstcookout/cookbooks/deployrails/recipes/deploysimpleway.rb

```
1  directory "/var/www" do
2    owner "user"
3    recursive true
4  end
5
6  # do not use ~... in vagrant, will goto vagrant user
7  execute "git clone git://github.com/jbwyatt4/#{node.default['bluebook']['app1']}.\
8  git /home/user/#{node.default['bluebook']['app1']}" do
9    user "user"
10   not_if do
11     ::File.directory?("/home/user/#{node.default['bluebook']['app1']}")
12   end
13 end
```

We pull the latest changes from the repo for future runs-useless for the first run since it was already pulled:

https://gist.github.com/jbwyatt4/7052192

[3] www.github.com/jbwyatt4/your_rails_app

firstcookout/cookbooks/deployrails/recipes/deploysimpleway.rb

```
directory "/var/www" do
  owner "user"
  recursive true
end

# do not use ~... in vagrant, will goto vagrant user
execute "git clone git://github.com/jbwyatt4/#{node.default['bluebook']['app1']}.\
git /home/user/#{node.default['bluebook']['app1']}" do
  user "user"
  not_if do
    ::File.directory?("/home/user/#{node.default['bluebook']['app1']}")
  end
end

execute "cd /home/user/#{node.default['bluebook']['app1']}; git pull" do
  user "user"
end
```

We remove the old version of the app at the webroot if it exists before copying over the new version.

https://gist.github.com/jbwyatt4/7052194

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

template "/opt/nginx/conf/nginx.conf" do
  source "nginx.conf.erb"
  mode 0440
  owner "root"
  group "root"
  notifies :run, 'execute[nginx-restart]', :immediately
end

template "/etc/default/nginx" do
  source "nginx.erb"
  mode 0440
  owner "root"
  group "root"
  notifies :run, 'execute[nginx-restart]', :immediately
end
```

```
execute "nginx-restart" do
  command "/etc/init.d/nginx restart; true" # we add true to end to return 0
                                            # since this script keeps
                                            # returning 1
  user "root"
  action :nothing
end
```

Finally we fix the permissions, run bundle install, migrate, precompile assets and restart the server.

https://gist.github.com/jbwyatt4/7052195

firstcookout/cookbooks/deployrails/recipes/rubyinstall.rb

```
{Above Code Not Shown}

template "/opt/nginx/conf/nginx.conf" do
  source "nginx.conf.erb"
  mode 0440
  owner "root"
  group "root"
  notifies :run, 'execute[nginx-restart]', :immediately
end

template "/etc/default/nginx" do
  source "nginx.erb"
  mode 0440
  owner "root"
  group "root"
  notifies :run, 'execute[nginx-restart]', :immediately
end

execute "nginx-restart" do
  command "/etc/init.d/nginx restart; true" # we add true to end to return 0
                                            # since this script keeps
                                            # returning 1
  user "root"
  action :nothing
end
```

Your application should be running now. Check your ip (148.211.114.67) or domain name to see if you get the Rails static page. If you do then check the dynamic part.

7.6 Deploy your_rails_app the Mina Way

Mina is a lightweight deployment tool that you can use to deploy a web app. Mina is stil in early development so it does not match all the features of Capistrano, but it is much easier to use.

Before you install Mina, on your *local* machine change your ssh configuration to whitelist your ip 148.211.114.67:

~/.ssh/config

```
1 Host 148.211.114.67
2 ForwardAgent yes
```

Comment out `include_recipe` "deployserver::deploysimpleway". `Add` include_recipe "deployserver::deployminaway"'

firstcookout/cookbooks/deployrails/recipes/deployminaway.rb

```
directory "#{node.default['bluebook']['webroot']}" do
  owner "user"
  recursive true
end

template "#{node.default['bluebook']['webroot']}/application.yml" do
  source "application.yml.erb"
  mode 0440
  owner "user"
end
```

Now install Mina by adding it to Gemfile.

```
gem 'mina'
```

Run bundle install without production.

```
bundle install --without production
```

In the root of your rails app on your local machine, initialize Mina:

```
mina init
```

Copy the following code to the Mina deploy file:

https://gist.github.com/jbwyatt4/7638465

your_rails_app/config/deploy.rb

```
require 'mina/bundler'
require 'mina/rails'
require 'mina/git'
# require 'mina/rbenv'  # for rbenv support. (http://rbenv.org)
require 'mina/rvm'    # for rvm support. (http://rvm.io)

set :domain, '148.211.114.67'
set :user, 'user'
set :webroot, '/var/www'
set :app_name, 'your_rails_app'
set :deploy_to, "#{webroot}/#{app_name}"
set :repository, 'http://github.com/jbwyatt4/your_rails_app.git'
set :rvm_path, '/etc/profile.d/rvm.sh'
set :rvm_gemset, 'ruby-1.9.3-p448@default'
set :shared_paths, ['log', 'config/application.yml']
set :term_mode, nil # needed to solve a login problem

# If your using the rails bluebook recipe with Mina
# for the first time, use this task with:
# mina cold_start --verbose
tast :cold_start do
  # You call other Mina (Rake) tasks with invoke.
  invoke :'setup'
  invoke :'deploy'
  invoke :'hard_restart'
  queue! %[echo "Mina has deployed #{app_name}!"]
end

# Setup a shared folder so logs and the application.yml
# can be shared between different versions of the app.
# If you attempt to copy the application.yml w/o a shared
# link you will get permission errors.
task :setup => :environment do
  # create shared folders and set permissions
  # The ! tells Mina to print out the terminal output.
  queue! %[mkdir -p "#{deploy_to}/shared/log"]
  queue! %[chmod g+rx,u+rwx "#{deploy_to}/shared/log"]

  queue! %[mkdir -p "#{deploy_to}/shared/config"]
  queue! %[chmod g+rx,u+rwx "#{deploy_to}/shared/config"]
```

```
  queue! %[touch "#{deploy_to}/shared/config/application.yml"]
  queue! %[chmod g+rx,u+rwx "#{deploy_to}/shared/config/application.yml"]
end

# Wipe out the deploy
task :wipe do
  queue! %[rm -rf "#{deploy_to}"]
end

# Sometimes a deploy can be corrupted when
# your testing it out.
# Use this command to restart fresh.
task :rebuild do
  invoke :'wipe'
  invoke :'setup'
  invoke :'deploy'
  invoke :'hard_restart'
  queue! %[echo "Mina rebuilt #{app_name}!"]
end

# Sets our gem environment
task :environment do
  invoke :"rvm:use[#{rvm_gemset}]"
end

task :restart do
  # How to restart Passenger
  queue 'sudo touch /tmp/restart'
end

# When Nginx and Passenger are stubborn
task :hard_restart do
  queue 'sudo reboot'
end

# The => notation means 'environment' gets run before the deploy task.
task :deploy => :environment do
  deploy do
    # Put things that prepare the empty release folder here.
    # Commands queued here will be ran on a new release directory.
    invoke :'git:clone'
```

```
    # We linked the shared folder based on the shared_paths variable.
    invoke :'deploy:link_shared_paths'
    # Copy the application.yml file
    queue! %[cp "#{webroot}/application.yml" "#{deploy_to}/shared/config/applicat\
ion.yml"]
    queue! %[chmod g+rx,u+rwx "#{deploy_to}/shared/config/application.yml"]

    invoke :'bundle:install'
    invoke :'rails:db_migrate'
    invoke :'rails:assets_precompile'
  end
end

desc "Shows logs."
task :logs do
  queue %[cd #{deploy_to!} && tail -f logs/error.log]
end
```

In the first few lines we include a few necessary libraries. In the next few lines we set a few variables. The rest of the file is dedicated to tasks. You can run these tasks with `mina task_name`

This configuration ties into the Vagrant appendix in using 33.33.13.39 as the domain name with the box that vagrant sets up. You must change this to whatever ip/domain name you are using. Also note the ruby version and gemset.

Now you get to actually use Mina. Setup Mina on the server with this command in your_rails_app's root:

```
mina setup --verbose
```

Mina should connect with the server and performs the setup task. The setup task sets up shared files and folders that the different version of your web app will use. `queue` send commands to the terminal while `invoke` executes Ruby code.

Now deploy your web application:

```
mina deploy
```

This command is the largest one so let's walk through. First RVM is sourced by going to environment command. Mina than clones the public git tree wth `invoke :'git:clone'` to store in the target directory. The target directory is set by the :deploy_to string. The next command links the shared paths setup in the setup task. The next few commands are the familar ones that setup Rails.

Then restart Passenger if you wish:

```
mina restart
```

Your app should be running on the new server.

Try running `mina deploy` again. You will see it notifies you of v2 of the app.

Books by John B. Wyatt IV

Automate PHP Deploys

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

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

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[4] and pick out an image. I recommend this Ubuntu precise 64 VirtualBox[5] 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.

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

[5] 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 > vagrant.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![6]

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

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.

Appendix - FAQ / Troubleshooting

RVM autolibs error

Back around 2013-03 I got this nice error.

> Installing requirements for ubuntu, might require sudo password.
>
> Skipping `apt-get update` make sure your system is up to date.
>
> RVM autolibs is now configured with mode '2' ⇒ 'check and stop if missing', please run `rvm autolibs enable` to let RVM do it's job or run and read `rvm autolibs [help]` or visit https://rvm.io/rvm/autolibs for more information.
>
> Missing required packages: git-core, patch.
>
> RVM autolibs is now configured with mode '2' ⇒ 'check and stop if missing', please run `rvm autolibs enable` to let RVM do it's job or run and read `rvm autolibs [help]`
>
> or visit https://rvm.io/rvm/autolibs for more information.

As of this time part of RVM is broken but easily fixed with this terminal command.

```
rvm autolibs enable
```

Rerun the requirements.

```
rvm requirements
```

It will automatically download all the packages you need to build ruby.

Knife complains about a private key

```
ERROR: Your private key could not be loaded from /etc/chef/client.pem Check
your configuration file and ensure that your private key is readable
```

This means your trying to access a feature of chef client which expects a key to be able to communicate with a fleet of servers.

Password to user is not being set correctly by Chef

Make sure you install the ruby-shadow gem and hashed the password with

Namespace Error!

Chef will often give * which really means you have a statement that is legal Ruby but Chef doesn't know how to handle it in it's DSL.

Server not responding / Nginx doesn't seem to be running

If the response is immediate, that means the server refused to communicate with you (a prolongue delay means the server may not be up). Try `sudo /etc/init.d/nginx start` in the console and see if Nginx complains about it's configuration file.

Nginx gives 404 Not Found error, the app folder was copied and the nginx webroot was setup

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

If you get the text below check your Passenger path in the Nginx configuration file.

```
Restarting nginx: nginx: [alert] Unable to start the Phusion Passenger watchdog b\
ecause its executable (/usr/lib/phusion-passenger/agents/PassengerWatchdog) does \
not exist. This probably means that your Phusion Passenger installation is broken\
 or incomplete, or that your 'passenger_root' directive is set to the wrong value\
. Please reinstall Phusion Passenger or fix your 'passenger_root' directive, whic\
hever is applicable. (-1: Unknown error)
```

Passenger gives error: Ruby (Rack) application could not be started

Make sure you set the permissions as in the deploy chapter.

production.log complains about not being able to find the sales table

Try `rake db:migrate RAILS_ENV='production'` in your rails app folder. If your using Mina and deploying a Rails app w/o migrating the database; Mina thinks since the schema hasn't changed the database has already been migrated. Use `mina wipe` and then trying setting it up or use `mina rebuild` to rebuild the deploy in one step.

Nginx restart doesn't seem to start nginx when a fixed error caused it to not start.

Yeah, you have to use `sudo /etc/init.d/nginx start` because the script is picky about it.

Host key mismatch

Delete the entry cited in your .ssh/known_hosts file on the machine initiating the connection (likely your local machine). Use `ssh-keygen -f "/home/yourusername/.ssh/known_hosts" -R XXX.XXX.XXX.XXX`

Mina reports a bad password.

First, make sure you properly hashed the password and have that hash set in the attributes of the recipe. Second, make sure `set :term_mode, nil` is in the mina config.

Appendix - Common Commands

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

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

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

```
knife solo prepare --identity-file vagrant.key vagrant@33.33.13.39
```

```
knife solo cook --identity-file vagrant.key vagrant@33.33.13.39 -V
```

Bootstrap combines both prepare and cook.

```
knife solo bootstrap --identity-file vagrant.key vagrant@33.33.13.39 -V
```

Appendix - Changelog

Version 1.3 - 14 April 2014

Fixed the RVM problem by just using the terminal command. Heartbleed required a fix to all potential packages that were outside the Ubuntu repos.

You can check to see if your OpenSSL package is vulnerable with:

```
sudo openssl version -a
```

```
OpenSSL 1.0.1 14 Mar 2012 built on: Mon Apr 7 20:33:29 UTC 2014
```

If it was built before Apr 7'th, it's vulnerable.

The following packages have had their instructions changed because of Heartbleed:

Chef

Vagrant

Ruby

RVM

Passenger (w/ Nginx)

Vagrant is no longer supported in Rubygems; you must download it manually from www.vagrantup.com/downloads

Chef had to be moved to version 11. Chef 11 behaves differently in the kind of output it gives. Chef 11 no longer reports terminal output, but does report template file changes by default. To show some console output from remote shell add the -V option to the knife cook and knife bootstrap commands.

Version 1.2 - 9 March 2014

A change in the way RVM handles gemsets has broken the RVM recipe for all newer versions of RVM and Ruby 1.9.3 patchlevels. This is the second time a change in RVM has broken the recipe which is naturally unacceptable and i'm looking for a replacement. Please see the Github repo for more details. The manual installation instructions still work.

Version 1.1 - 23 Dec 2013

Removed ECDSA instructions. Use the following if you want to generate a ECC key.

```
ssh-keygen -t ecdsa
```

Version 1.0 - 29 Nov 2013

Minor corrections. Amazon release.

Version 0.9.0 - 21 Nov 2013

Added database and Mina sections. New cover image. Spelling and grammar fixes.

Version 0.8.1 - 22 Oct 2013

Spelling, grammar and cover image fixes.

Vagrant chapter: box was taken down. Use the default box from vagrantup.com

Updated public-private key instructions to use Elliptic Curve Cryptography (ECDSA). The previous RSA instructions will become obselete within 5 years since that is the time RSA is expected to be cracked.

For more details, please read this Ars Technica article.

http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/1/

Note that both Github and BitBucket at this time, do not accept ECDSA keys. Use:

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

and

```
ssh add
```

to generate the `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`

Version 0.8 - 20 Oct 2013

Leanpub release!

How to Contact Me

My blog is at http://sageofredondo.blogspot.com. You can find additional tutorials or social musings.

My Twitter is @sageofredondo

Feel free to ask me questions or just tell me what you think of my books.