[1.4 Deploying](http://ruby.railstutorial.org/chapters/beginning" \l "sec-deploying)

Even at this early stage, we’re already going to deploy our (still-empty) Rails application to production. This step is optional, but deploying early and often allows us to catch any deployment problems early in our development cycle. The alternative—deploying only after laborious effort sealed away in a development environment—often leads to terrible integration headaches when launch time comes.[20](http://ruby.railstutorial.org/chapters/beginning#fn-1_20)

Deploying Rails applications used to be a pain, but the Rails deployment ecosystem has matured rapidly in the past few years, and now there are several great options. These include shared hosts or virtual private servers running [Phusion Passenger](http://www.modrails.com/) (a module for the Apache and Nginx[21](http://ruby.railstutorial.org/chapters/beginning#fn-1_21) web servers), full-service deployment companies such as [Engine Yard](http://engineyard.com/) and [Rails Machine](http://railsmachine.com/), and cloud deployment services such as [Engine Yard Cloud](http://cloud.engineyard.com/) and [Heroku](http://heroku.com/).

My favorite Rails deployment option is Heroku, which is a hosted platform built specifically for deploying Rails and other web applications. Heroku makes deploying Rails applications ridiculously easy—as long as your source code is under version control with Git. (This is yet another reason to follow the Git setup steps in [Section 1.3](http://ruby.railstutorial.org/chapters/beginning#sec-version_control) if you haven’t already.) The rest of this section is dedicated to deploying our first application to Heroku.

[1.4.1 Heroku setup](http://ruby.railstutorial.org/chapters/beginning#sec-heroku_setup)

Heroku uses the [PostgreSQL](http://www.postgresql.org/) database (pronounced “post-gres-cue-ell”, and often called “Postgres” for short), which means that we need to add the pg gem in the production environment to allow Rails to talk to Postgres:

group :production **do**

gem 'pg', '0.15.1'

gem 'rails\_12factor', '0.0.2'

**end**

Note also the addition of the rails\_12factor gem, which is used by Heroku to serve static assets such as images and stylesheets.

As mentioned in [Section 1.2.4](http://ruby.railstutorial.org/chapters/beginning#sec-bundler), it’s also a good idea to specify explictly which version of Ruby our applications expects:

ruby '2.0.0'

*#ruby-gemset=railstutorial\_rails\_4\_0*

(Here I’ve also added the optional RVM gemset line for convenience. You should substitute**’1.9.3’** if that’s the version of Ruby you’re using, though for this tutorial the difference shouldn’t ever matter.) Applying these changes to the **Gemfile** from [Listing 1.5](http://ruby.railstutorial.org/chapters/beginning#code-gemfile_sqlite_version) yields [Listing 1.9](http://ruby.railstutorial.org/chapters/beginning#code-gemfile_pg_gem).

**Listing 1.9.** A **Gemfile** with added gems and explicit Ruby version.

source 'https://rubygems.org'

ruby '2.0.0'

*#ruby-gemset=railstutorial\_rails\_4\_0*

gem 'rails', '4.0.2'

group :development **do**

gem 'sqlite3', '1.3.8'

**end**

gem 'sass-rails', '4.0.1'

gem 'uglifier', '2.1.1'

gem 'coffee-rails', '4.0.1'

gem 'jquery-rails', '3.0.4'

gem 'turbolinks', '1.1.1'

gem 'jbuilder', '1.0.2'

group :doc **do**

gem 'sdoc', '0.3.20', require: false

**end**

group :production **do**

gem 'pg', '0.15.1'

gem 'rails\_12factor', '0.0.2'

**end**

To install it, we run **bundle install** with a special flag:

**$** bundle install --without production

The --without production option prevents the local installation of any production gems, which in this case consists of pg and rails\_12factor. (If Bundler complains about a readlineerror, try adding gem 'rb-read\-line', '~> 0.4.2' to your **Gemfile**.) Because the only gems we’ve added are restricted to a production environment, right now this command doesn’t actually install any additional local gems, but it’s needed to update **Gemfile.lock** with the pgand rails\_12factor gems and the specific Ruby version. We can commit the resulting change as follows:

**$** git commit -a -m "Update Gemfile.lock for Heroku"

(Some readers have reported that they need one last bit of configuration at this point, namely, creating the files Heroku needs to serve static assets like images and CSS:

**#** This should only be used **if** your Heroku deploy fails without it.

**$** rake assets:precompile

**$** git add .

**$** git commit -m "Add precompiled assets for Heroku"

(This uses the **rake** command, which we’ll cover in more detail in [Section 2.2](http://ruby.railstutorial.org/chapters/a-demo-app#sec-demo_users_resource).) The asset precompile step shouldn’t be necessary, and I have been unable to reproduce the issue, but the reports are common enough that I include it here for reference.)

Next we have to create and configure a new Heroku account. The first step is to [sign up for Heroku](http://api.heroku.com/signup); after checking your email to complete the creation of your account, install the necessary Heroku software using the [Heroku Toolbelt](https://toolbelt.heroku.com/).[22](http://ruby.railstutorial.org/chapters/beginning#fn-1_22) Then use the **heroku** command to log in at the command line (you may have to exit and restart your terminal program first):

**$** heroku login

Finally, navigate back to your Rails project directory and use the **heroku** command to create a place on the Heroku servers for the sample app to live ([Listing 1.10](http://ruby.railstutorial.org/chapters/beginning#code-heroku_create)).

**Listing 1.10.** Creating a new application at Heroku.

**$** cd ~/rails\_projects/first\_app

**$** heroku create

Created http://stormy-cloud-5881.herokuapp.com/ |

git@heroku.com:stormy-cloud-5881.herokuapp.com

Git remote heroku added

The **heroku** command creates a new subdomain just for our application, available for immediate viewing. There’s nothing there yet, though, so let’s get busy deploying.

[1.4.2 Heroku deployment, step one](http://ruby.railstutorial.org/chapters/beginning#sec-heroku_step_one)

To deploy the application, the first step is to use Git to push it up to Heroku:

**$** git push heroku master

[1.4.3 Heroku deployment, step two](http://ruby.railstutorial.org/chapters/beginning#sec-1_4_3)

There is no step two! We’re already done ([Figure 1.10](http://ruby.railstutorial.org/chapters/beginning#fig-heroku_app)). To see your newly deployed application, you can visit the address that you saw when you ran **heroku create** (i.e., [Listing 1.10](http://ruby.railstutorial.org/chapters/beginning#code-heroku_create), but with the address for your app, not the address for mine). You can also use an argument to the **heroku**command that automatically opens your browser with the right address:

**$** heroku open

Unfortunately, the resulting page is an error; as of Rails 4.0, for technical reasons the default Rails page doesn’t work on Heroku. The good news is that the error will go away (in the context of the full sample application) when we add a root route in [Section 5.3.2](http://ruby.railstutorial.org/chapters/filling-in-the-layout#sec-rails_routes).

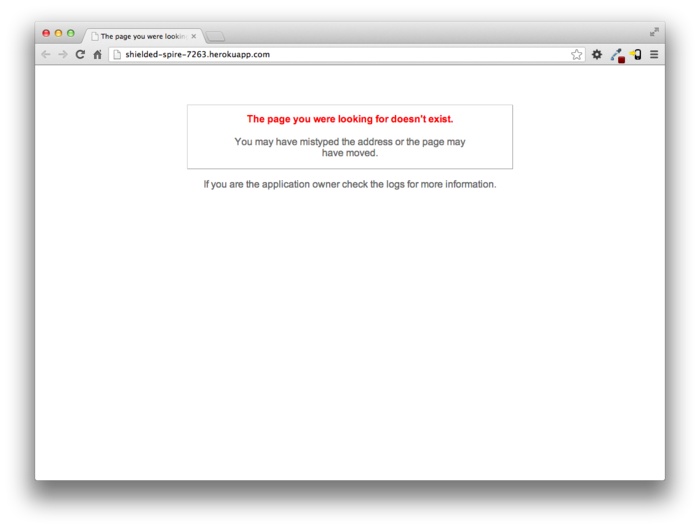


Figure 1.10: The first Rails Tutorial application running on Heroku. [(full size)](http://railstutorial.org/images/figures/heroku_app_4_0-full.png)

Once you’ve deployed successfully, Heroku provides a beautiful interface for administering and configuring your application ([Figure 1.11](http://ruby.railstutorial.org/chapters/beginning#fig-heroku_info)).

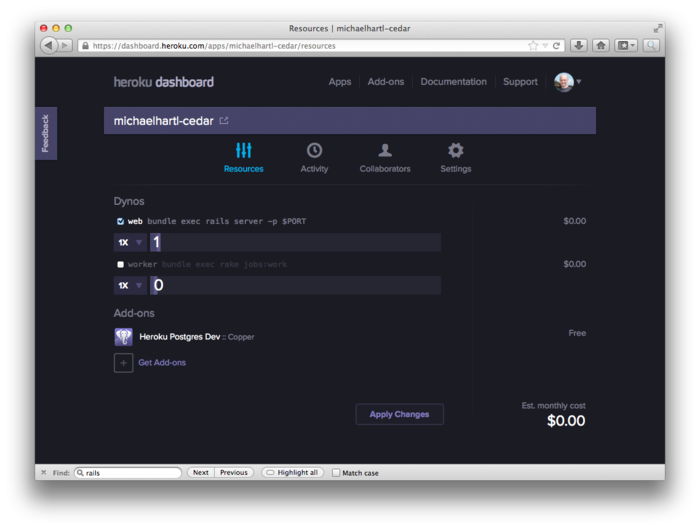


Figure 1.11: The beautiful interface at Heroku. [(full size)](http://railstutorial.org/images/figures/heroku_info_4_0-full.png)

[1.4.4 Heroku commands](http://ruby.railstutorial.org/chapters/beginning#sec-heroku_commands)

There are many [Heroku commands](http://devcenter.heroku.com/heroku-command), and we’ll barely scratch the surface in this book. Let’s take a minute to show just one of them by renaming the application as follows:

**$** heroku rename railstutorial

Don’t use this name yourself; it’s already taken by me! In fact, you probably shouldn’t bother with this step right now; using the default address supplied by Heroku is fine. But if you do want to rename your application, you can arrange for it to be reasonably secure by using a random or obscure subdomain, such as the following:

hwpcbmze.herokuapp.com

seyjhflo.herokuapp.com

jhyicevg.herokuapp.com

With a random subdomain like this, someone could visit your site only if you gave them the address. (By the way, as a preview of Ruby’s compact awesomeness, here’s the code I used to generate the random subdomains:

('a'..'z').to\_a.shuffle[0..7].join

Pretty sweet.)

In addition to supporting subdomains, Heroku also supports custom domains. (In fact, the [Ruby on Rails Tutorial site](http://railstutorial.org/) lives at Heroku; if you’re reading this book online, you’re looking at a Heroku-hosted site right now!) See the [Heroku documentation](http://devcenter.heroku.com/) for more information about custom domains and other Heroku topics.

[1.5 Conclusion](http://ruby.railstutorial.org/chapters/beginning#sec-beginning_conclusion)

We’ve come a long way in this chapter: installation, development environment setup, version control, and deployment. If you want to share your progress at this point, feel free to send a tweet or Facebook status update with something like this:

[I’m learning Ruby on Rails with @railstutorial! http://railstutorial.org/](http://twitter.com/?status=I%27m%20learning%20Ruby%20on%20Rails%20with%20@railstutorial!%20http://railstutorial.org/)

All that’s left is to actually start learning Rails! Let’s get to it.

[Chapter 2 A demo app »](http://ruby.railstutorial.org/chapters/a-demo-app#top)

1. *URI* stands for Uniform Resource Identifier, while the slightly less general *URL* stands for Uniform Resource Locator. In practice, the URL is usually equivalent to “the thing you see in the address bar of your browser”. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_1)
2. http://tryruby.org/ [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_2)
3. http://railsforzombies.org/ [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_3)
4. http://railstutorial.org/screencasts [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_4)
5. When reading the *Rails Tutorial*, you may find it convenient to follow an internal section link to look at the reference and then immediately go back to where you were before. This is easy when reading the book as a web page, since you can just use the Back button of your browser, but both Adobe Reader and OS X’s Preview allow you to do this with the PDF as well. In Reader, you can right-click on the document and select “Previous View” to go back. In Preview, use the Go menu:Go > Back. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_5)
6. Many people erroneously believe that **sudo** stands for “superuser do” because it runs commands as the superuser (root) by default. In fact, **sudo** is a concatenation of the **su** command and the English word “do”, and **su** stands for “substitute user”, as you can verify by typing **man su** in your shell. This etymology also suggests the pronunciation “SOO-doo” (because the word “do” is pronounced “doo”), although the alternate pronunciation “SOO-doh” is also common. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_6)
7. http://railstutorial.org/help [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_7)
8. https://github.com/perfectionist/sample\_project/wiki [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_8)
9. As of this writing, [Sublime Text 3](http://www.sublimetext.com/3) is in beta. I recommend trying the newest Sublime Text only if you really want to be on the bleeding edge. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_9)
10. The vi editor is one of the most ancient yet powerful weapons in the Unix arsenal, and Vim is “vi improved”. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_10)
11. https://github.com/mhartl/rails\_tutorial\_sublime\_text [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_11)
12. https://developer.apple.com/downloads/ [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_12)
13. http://strandcode.com/2013/07/11/ruby-version-manager-rvm-overview-for-rails-newbs/ [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_13)
14. This step is necessary only if you’ve changed the version of the Rails gem, which would likely happen only if you’re using Rails Installer, but it does no harm to run it in other cases as well. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_14)
15. Normally, websites run on port 80, but this usually requires special privileges, so Rails picks a less restricted higher-numbered port for the development server. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_15)
16. Normally this is a feature, since it lets you continue to use the command line after launching your editor, but Git interprets the detachment as closing the file with an empty commit message, which prevents the commit from going through. I only mention this point because it can be seriously confusing if you try to set your editor to **subl** or **gvim** without the flag. (If you find this note confusing, it is safe to ignore it.) [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_16)
17. If you can’t see the **.gitignore** file in your directory, you may need to configure your directory viewer to show hidden files. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_17)
18. If in the future any unwanted files start showing up when you type **git status**, just add them to your **.gitignore** file from [Listing 1.7](http://ruby.railstutorial.org/chapters/beginning#code-gitignore). [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_18)
19. See the chapter [Git Branching in *Pro Git*](http://git-scm.com/book/en/git-branching) for details. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_19)
20. Though it shouldn’t matter for the example applications in the *Rails Tutorial*, if you’re worried about accidentally making your app public too soon there are several options; see [Section 1.4.4](http://ruby.railstutorial.org/chapters/beginning#sec-heroku_commands) for one. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_20)
21. Pronounced “Engine X”. [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_21)
22. https://toolbelt.heroku.com/ [**↑**](http://ruby.railstutorial.org/chapters/beginning#fnref-1_22)