Getting Started on Heroku with Rails 5.x

Last updated 15 August 2018

Table of Contents

Ruby on Rails is a popular web framework written in Ruby (http://www.ruby-lang.org/). This guide covers using Rails 5 on Heroku. For information on running previous versions of Rails on Heroku, see the tutorial for Rails 4.x (https://devcenter.heroku.com/articles/getting-started-with-rails4) or Rails 3.x (https://devcenter.heroku.com/articles/getting-started-with-rails3).

For this guide you will need:

- Basic familiarity with Ruby/Rails and Git
- A locally installed version of Ruby 2.2.0+, Rubygems, Bundler, and Rails 5+
- A Heroku user account: Signup is free and instant (https://signup.heroku.com/devcenter).

Local setup

Install the Heroku CLI (https://devcenter.heroku.com/articles/heroku-cli#download-and-install) on your development machine.

Once installed, the `heroku` command is available from your terminal. Log in using your Heroku account's email address and password:

```
$ heroku login
heroku: Enter your Heroku credentials
Email: schneems@example.com
Password:
Could not find an existing public key.
Would you like to generate one? [Yn]
Generating new SSH public key.
Uploading ssh public key /Users/adam/.ssh/id_rsa.pub
```

Press Enter at the prompt to upload your existing `ssh` key or create a new one, used for pushing code later on.

Create a new Rails app (or upgrade an existing one)

If you are starting with an existing app that uses a previous version of Rails, upgrade it to Rails 5 (http://edgeguides.rubyonrails.org/upgrading_ruby_on_rails.html#upgrading-from-rails-4-2-to-rails-5-0) before continuing. If you're not starting from an existing app at all, a vanilla Rails 5 app works great as a sample app.

To create a new app, first make sure that you're using Rails 5.x by running `rails -v` . If necessary, you can get the new version of rails by running the following:

```
$ gem install rails --no-ri --no-rdoc
Successfully installed rails-5.2.1
1 gem installed
```

Then create a new app and move into its root directory:

```
$ rails new myapp --database=postgresql
$ cd myapp
```

Add the pg gem

If you're using an existing app that was created without specifying `--database=postgresql` , you need to add the `pg` gem to your Rails project. Edit your `Gemfile` and change this line:

```
gem 'sqlite3'
```

To this:

```
gem 'pg'
```

We highly recommend using PostgreSQL during development. Maintaining **parity between your development (http://www.12factor.net/dev-prod-parity)** and deployment environments prevents subtle bugs from being introduced because of differences between your environments. **Install Postgres locally (https://devcenter.heroku.com/articles/heroku-postgresql#local-setup)** now if it is not already on your system.

Now re-install your dependencies (to generate a new `Gemfile.lock`):

```
$ bundle install
```

For more information on why Postgres is recommended instead of Sqlite3, see why you cannot use Sqlite3 on Heroku (https://devcenter.heroku.com/articles/sqlite3).

In addition to using the `pg` gem, ensure that your `config/database.yml` file is using the `postgresql` adapter. The development section of your `config/database.yml` file should look something like this:

```
$ cat config/database.yml
# PostgreSQL. Versions 9.1 and up are supported.
#
# Install the pg driver:
#   gem install pg
# On OS X with Homebrew:
#   gem install pg -- --with-pg-config=/usr/local/bin/pg_config
# On OS X with MacPorts:
#   gem install pg -- --with-pg-config=/opt/local/lib/postgresql84/bin/pg_config
# On Windows:
#   gem install pg
#       Choose the win32 build.
#       Install PostgreSQL and put its /bin directory on your path.
#
# Configure Using Gemfile
# gem 'pg'
#
default: &default
  adapter: postgresql
  encoding: unicode
  # For details on connection pooling, see Rails configuration guide
  # http://guides.rubyonrails.org/configuring.html#database-pooling
  pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>

development:
  <<: *default
  database: myapp_development

  # The specified database role being used to connect to postgres.
  # To create additional roles in postgres see `$ createuser --help`.
  # When left blank, postgres will use the default role. This is
  # the same name as the operating system user that initialized the database.
  #username: myapp

  # The password associated with the postgres role (username).
  #password:

  # Connect on a TCP socket. Omitted by default since the client uses a
  # domain socket that doesn't need configuration. Windows does not have
  # domain sockets, so uncomment these lines.
  #host: localhost

  # The TCP port the server listens on. Defaults to 5432.
  # If your server runs on a different port number, change accordingly.
  #port: 5432

  # Schema search path. The server defaults to $user,public
  #schema_search_path: myapp,sharedapp,public

  # Minimum log levels, in increasing order:
  #   debug5, debug4, debug3, debug2, debug1,
  #   log, notice, warning, error, fatal, and panic
  # Defaults to warning.
  #min_messages: notice

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test:
  <<: *default
  database: myapp_test

# As with config/secrets.yml, you never want to store sensitive information,
# like your database password, in your source code. If your source code is
# ever seen by anyone, they now have access to your database.
#
# Instead, provide the password as a unix environment variable when you boot
# the app. Read http://guides.rubyonrails.org/configuring.html#configuring-a-database
# for a full rundown on how to provide these environment variables in a
# production deployment.
#
# On Heroku and other platform providers, you may have a full connection URL
# available as an environment variable. For example:
#
#   DATABASE_URL="postgres://myuser:mypass@localhost/somedatabase"
#
# You can use this database configuration with:
#
#   production:
#     url: <%= ENV['DATABASE_URL'] %>
#
production:
  <<: *default
  database: myapp_production
  username: myapp
  password: <%= ENV['MYAPP_DATABASE_PASSWORD'] %>
```

Be careful here. If you omit the `sql` at the end of `postgresql` in the `adapter` section, your application will not work.

Create a welcome page

Rails 5 no longer has a static index page in production. When you're using a new app, there will not be a root page in production, so we need to create one. We will first create a controller called `welcome` for our home page to live:

```
$ rails generate controller welcome
```

Next we'll add an index page:

In file `app/views/welcome/index.html.erb` write:

```
<h2>Hello World</h2>
<p>
  The time is now: <%= Time.now %>
</p>
```

Now we need to make Rails route to this action. We'll edit `config/routes.rb` to set the index page to our new method:

In file `config/routes.rb`, on line 2 add:

```
  root 'welcome#index'
```

You can verify that the page is there by running your server:

```
$ rails server
```

And visiting http://localhost:3000 (http://localhost:3000) in your browser. If you do not see the page, use the logs that are output to your server to debug.

Heroku gems

Previous versions of Rails required you to add a gem to your project rails_12factor (https://github.com/heroku/rails_12factor) to enable static asset serving and logging on Heroku. If you are deploying a new application, this gem is not needed. If you are upgrading an existing application, you can remove this gem provided you have the appropriate configuration in your `config/environments/production.rb` file:

```
# config/environments/production.rb
config.public_file_server.enabled = ENV['RAILS_SERVE_STATIC_FILES'].present?
if ENV["RAILS_LOG_TO_STDOUT"].present?
  logger           = ActiveSupport::Logger.new(STDOUT)
  logger.formatter = config.log_formatter
  config.logger = ActiveSupport::TaggedLogging.new(logger)
end
```

Specify your Ruby version

Rails 5 requires Ruby 2.2.0 or above. Heroku has a recent version of Ruby installed by default, however you can specify an exact version by using the `ruby` DSL in your `Gemfile` :

At the end of `Gemfile` add:

```
ruby "2.5.1"
```

You should also be running the same version of Ruby locally. You can check this by running `$ ruby -v` . You can get more information on specifying your Ruby version on Heroku here (https://devcenter.heroku.com/articles/ruby-versions).

Store your app in Git

Heroku relies on Git (http://git-scm.com/), a distributed source control management tool, for deploying your project. If your project is not already in Git, first verify that `git` is on your system:

```
$ git --help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]
```

If you don't see any output or get `command not found` you need to install Git on your system.

Once you've verified that Git works, first make sure you are in your Rails app directory by running `$ ls` :

The output should look like this:

```
$ ls
Gemfile
Gemfile.lock
README.md
Rakefile
app
bin
config
config.ru
db
lib
log
package.json
public
storage
test
tmp
vendor
```

Now run these commands in your Rails app directory to initialize and commit your code to Git:

```
$ git init
$ git add .
$ git commit -m "init"
```

You can verify everything was committed correctly by running:

```
$ git status
On branch master
nothing to commit, working tree clean
```

Now that your application is committed to Git you can deploy to Heroku.

Deploy your application to Heroku

Make sure you are in the directory that contains your Rails app, then create an app on Heroku:

```
$ heroku create
Creating app... done, secret-reef-24871
https://secret-reef-24871.herokuapp.com/ | https://git.heroku.com/secret-reef-24871.git
```

You can verify that the remote was added to your project by running:

```
$ git config --list | grep heroku
remote.heroku.url=https://git.heroku.com/secret-reef-24871.git
remote.heroku.fetch=+refs/heads/*:refs/remotes/heroku/*
```

If you see `fatal: not in a git directory` then you are likely not in the correct directory. Otherwise you can deploy your code. After you deploy your code, you need to migrate your database, make sure it is properly scaled, and use logs to debug any issues that come up.

Deploy your code:

```
$ git push heroku master
remote: Compressing source files... done.
remote: Building source:
remote:
remote:  !     Warning: Multiple default buildpacks reported the ability to handle this app. The first buildp
remote:             Detected buildpacks: Ruby,Node.js
remote:             See https://devcenter.heroku.com/articles/buildpacks#buildpack-detect-order
remote: -----> Ruby app detected
remote: -----> Compiling Ruby/Rails
remote: -----> Using Ruby version: ruby-2.5.1
remote: -----> Installing dependencies using bundler 1.15.2
remote:        Running: bundle install --without development:test --path vendor/bundle --binstubs vendor/bund
remote:        Warning: the running version of Bundler (1.15.2) is older than the version that created the lo
remote:        Fetching gem metadata from https://rubygems.org/.........
remote:        Fetching version metadata from https://rubygems.org/..
remote:        Fetching dependency metadata from https://rubygems.org/.
remote:        Fetching rake 12.3.1
remote:        Fetching concurrent-ruby 1.0.5
remote:        Fetching minitest 5.11.3
remote:        Installing minitest 5.11.3
remote:        Installing rake 12.3.1
remote:        Installing concurrent-ruby 1.0.5
remote:        Fetching thread_safe 0.3.6
remote:        Installing thread_safe 0.3.6
remote:        Fetching builder 3.2.3
remote:        Installing builder 3.2.3
remote:        Fetching erubi 1.7.1
remote:        Installing erubi 1.7.1
remote:        Fetching mini_portile2 2.3.0
remote:        Fetching crass 1.0.4
remote:        Installing mini_portile2 2.3.0
remote:        Installing crass 1.0.4
remote:        Fetching rack 2.0.5
remote:        Installing rack 2.0.5
remote:        Fetching nio4r 2.3.1
remote:        Installing nio4r 2.3.1 with native extensions
remote:        Fetching websocket-extensions 0.1.3
remote:        Installing websocket-extensions 0.1.3
remote:        Fetching mini_mime 1.0.0
remote:        Installing mini_mime 1.0.0
remote:        Fetching arel 9.0.0
remote:        Installing arel 9.0.0
remote:        Fetching mimemagic 0.3.2
remote:        Installing mimemagic 0.3.2
remote:        Fetching msgpack 1.2.4
remote:        Installing msgpack 1.2.4 with native extensions
```

```
remote:        Using bundler 1.15.2
remote:        Fetching coffee-script-source 1.12.2
remote:        Installing coffee-script-source 1.12.2
remote:        Fetching execjs 2.7.0
remote:        Installing execjs 2.7.0
remote:        Fetching method_source 0.9.0
remote:        Installing method_source 0.9.0
remote:        Fetching thor 0.20.0
remote:        Installing thor 0.20.0
remote:        Fetching ffi 1.9.25
remote:        Installing ffi 1.9.25 with native extensions
remote:        Fetching multi_json 1.13.1
remote:        Installing multi_json 1.13.1
remote:        Fetching pg 1.0.0
remote:        Installing pg 1.0.0 with native extensions
remote:        Fetching puma 3.12.0
remote:        Installing puma 3.12.0 with native extensions
remote:        Fetching rb-fsevent 0.10.3
remote:        Installing rb-fsevent 0.10.3
remote:        Fetching tilt 2.0.8
remote:        Installing tilt 2.0.8
remote:        Fetching turbolinks-source 5.1.0
remote:        Installing turbolinks-source 5.1.0
remote:        Fetching tzinfo 1.2.5
remote:        Installing tzinfo 1.2.5
remote:        Fetching i18n 1.1.0
remote:        Installing i18n 1.1.0
remote:        Fetching nokogiri 1.8.4
remote:        Installing nokogiri 1.8.4 with native extensions
remote:        Fetching websocket-driver 0.7.0
remote:        Installing websocket-driver 0.7.0 with native extensions
remote:        Fetching mail 2.7.0
remote:        Installing mail 2.7.0
remote:        Fetching rack-test 1.1.0
remote:        Installing rack-test 1.1.0
remote:        Fetching sprockets 3.7.2
remote:        Installing sprockets 3.7.2
remote:        Fetching marcel 0.3.2
remote:        Installing marcel 0.3.2
remote:        Fetching coffee-script 2.4.1
remote:        Installing coffee-script 2.4.1
remote:        Fetching uglifier 4.1.17
remote:        Installing uglifier 4.1.17
remote:        Fetching bootsnap 1.3.1
remote:        Installing bootsnap 1.3.1 with native extensions
remote:        Fetching rb-inotify 0.9.10
remote:        Installing rb-inotify 0.9.10
remote:        Fetching turbolinks 5.1.1
remote:        Installing turbolinks 5.1.1
remote:        Fetching activesupport 5.2.1
remote:        Installing activesupport 5.2.1
remote:        Fetching loofah 2.2.2
remote:        Installing loofah 2.2.2
remote:        Fetching sass-listen 4.0.0
remote:        Fetching rails-html-sanitizer 1.0.4
remote:        Installing sass-listen 4.0.0
remote:        Installing rails-html-sanitizer 1.0.4
remote:        Fetching rails-dom-testing 2.0.3
remote:        Installing rails-dom-testing 2.0.3
remote:        Fetching globalid 0.4.1
remote:        Fetching activemodel 5.2.1
remote:        Installing globalid 0.4.1
remote:        Installing activemodel 5.2.1
remote:        Fetching jbuilder 2.7.0
remote:        Installing jbuilder 2.7.0
remote:        Fetching sass 3.5.7
remote:        Fetching actionview 5.2.1
remote:        Fetching activejob 5.2.1
remote:        Installing activejob 5.2.1
remote:        Installing actionview 5.2.1
remote:        Installing sass 3.5.7
remote:        Fetching activerecord 5.2.1
remote:        Installing activerecord 5.2.1
remote:        Fetching actionpack 5.2.1
remote:        Installing actionpack 5.2.1
remote:        Fetching actioncable 5.2.1
remote:        Fetching actionmailer 5.2.1
remote:        Fetching activestorage 5.2.1
remote:        Installing actionmailer 5.2.1
remote:        Installing actioncable 5.2.1
remote:        Installing activestorage 5.2.1
remote:        Fetching railties 5.2.1
remote:        Installing railties 5.2.1
remote:        Fetching sprockets-rails 3.2.1
remote:        Installing sprockets-rails 3.2.1
```

```
remote:        Fetching rails 5.2.1
remote:        Fetching sass-rails 5.0.7
remote:        Fetching coffee-rails 4.2.2
remote:        Installing rails 5.2.1
remote:        Installing sass-rails 5.0.7
remote:        Installing coffee-rails 4.2.2
remote:        Bundle complete! 18 Gemfile dependencies, 61 gems now installed.
remote:        Gems in the groups development and test were not installed.
remote:        Bundled gems are installed into ./vendor/bundle.
remote:        Post-install message from sass:
remote:
remote:        Ruby Sass is deprecated and will be unmaintained as of 26 March 2019.
remote:
remote:        * If you use Sass as a command-line tool, we recommend using Dart Sass, the new
remote:          primary implementation: https://sass-lang.com/install
remote:
remote:        * If you use Sass as a plug-in for a Ruby web framework, we recommend using the
remote:          sassc gem: https://github.com/sass/sassc-ruby#readme
remote:
remote:        * For more details, please refer to the Sass blog:
remote:          http://sass.logdown.com/posts/7081811
remote:
remote:        Bundle completed (39.49s)
remote:        Cleaning up the bundler cache.
remote:        Warning: the running version of Bundler (1.15.2) is older than the version that created the lo
remote:        The latest bundler is 1.16.3, but you are currently running 1.15.2.
remote:        To update, run `gem install bundler`
remote: -----> Installing node-v8.10.0-linux-x64
remote: -----> Detecting rake tasks
remote: -----> Preparing app for Rails asset pipeline
remote:        Running: rake assets:precompile
remote:        Yarn executable was not detected in the system.
remote:        Download Yarn at https://yarnpkg.com/en/docs/install
remote:        I, [2018-08-08T22:16:55.417611 #1422]  INFO -- : Writing /tmp/build_385041339af5fe9074ca3c2d6c
remote:        I, [2018-08-08T22:16:55.418320 #1422]  INFO -- : Writing /tmp/build_385041339af5fe9074ca3c2d6c
remote:        I, [2018-08-08T22:16:55.430527 #1422]  INFO -- : Writing /tmp/build_385041339af5fe9074ca3c2d6c
remote:        I, [2018-08-08T22:16:55.430685 #1422]  INFO -- : Writing /tmp/build_385041339af5fe9074ca3c2d6c
remote:        Asset precompilation completed (3.97s)
remote:        Cleaning assets
remote:        Running: rake assets:clean
remote: -----> Detecting rails configuration
remote:
remote: ###### WARNING:
remote:
remote:        You set your `config.active_storage.service` to :local in production.
remote:        If you are uploading files to this app, they will not persist after the app
remote:        is restarted, on one-off dynos, or if the app has multiple dynos.
remote:        Heroku applications have an ephemeral file system. To
remote:        persist uploaded files, please use a service such as S3 and update your Rails
remote:        configuration.
remote:
remote:        For more information can be found in this article:
remote:          https://devcenter.heroku.com/articles/active-storage-on-heroku
remote:
remote:
remote: ###### WARNING:
remote:
remote:        We detected that some binary dependencies required to
remote:        use all the preview features of Active Storage are not
remote:        present on this system.
remote:
remote:        For more information please see:
remote:          https://devcenter.heroku.com/articles/active-storage-on-heroku
remote:
remote:
remote: ###### WARNING:
remote:
remote:        No Procfile detected, using the default web server.
remote:        We recommend explicitly declaring how to boot your server process via a Procfile.
remote:        https://devcenter.heroku.com/articles/ruby-default-web-server
remote:
remote:
remote: -----> Discovering process types
remote:        Procfile declares types     -> (none)
remote:        Default types for buildpack -> console, rake, web, worker
remote:
remote: -----> Compressing...
remote:        Done: 47M
remote: -----> Launching...
remote:        Released v5
remote:        https://secret-reef-24871.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/secret-reef-24871.git
 * [new branch]      master -> master
```

It is always a good idea to check to see if there are any warnings or errors in the output. If everything went well you can migrate your database.

Migrate your database

If you are using the database in your application, you need to manually migrate the database by running:

```
$ heroku run rake db:migrate
```

Any commands after the `heroku run` are executed on a Heroku dyno (https://devcenter.heroku.com/articles/dynos). You can obtain an interactive shell session by running `$ heroku run bash`.

Visit your application

You've deployed your code to Heroku. You can now instruct Heroku to execute a process type. Heroku does this by running the associated command in a dyno (https://devcenter.heroku.com/articles/dynos), which is a lightweight container that is the basic unit of composition on Heroku.

Let's ensure we have one dyno running the `web` process type:

```
$ heroku ps:scale web=1
```

You can check the state of the app's dynos. The `heroku ps` command lists the running dynos of your application:

```
$ heroku ps
Free dyno hours quota remaining this month: 887h 34m (88%)
For more information on dyno sleeping and how to upgrade, see:
https://devcenter.heroku.com/articles/dyno-sleeping

=== web (Free): bin/rails server -p $PORT -e $RAILS_ENV (1)
web.1: starting 2018/08/08 17:17:08 -0500 (~ 12s ago)
```

Here, one dyno is running.

We can now visit the app in our browser with `heroku open`.

```
$ heroku open
```

You should now see the "Hello World" text we inserted above.

Heroku gives you a default web URL for simplicity while you are developing. When you are ready to scale up and use Heroku for production you can add your own custom domain (https://devcenter.heroku.com/articles/custom-domains).

View logs

If you run into any problems getting your app to perform properly, you will need to check the logs.

You can view information about your running app using one of the logging commands (https://devcenter.heroku.com/articles/logging), `heroku logs`:

```
$ heroku logs
2018-08-08T22:16:02.280576+00:00 app[api]: Initial release by user developer@example.com2018-08-08T22:16:02.2
2018-08-08T22:17:13.055248+00:00 heroku[web.1]: Starting process with command `bin/rails server -p 10023 -e p
2018-08-08T22:17:23.846730+00:00 heroku[web.1]: State changed from starting to up
2018-08-08T22:17:25.320681+00:00 heroku[router]: at=info method=GET path="/" host=secret-reef-24871.herokuapp
2018-08-08T22:17:25.516862+00:00 heroku[router]: at=info method=GET path="/assets/application-e3b0c44298fc1c1
2018-08-08T22:17:25.517006+00:00 heroku[router]: at=info method=GET path="/assets/application-8dbcde118d1b2a1
2018-08-08T22:17:25.251543+00:00 app[web.1]: => Booting Puma
2018-08-08T22:17:25.251584+00:00 app[web.1]: => Rails 5.2.1 application starting in production
2018-08-08T22:17:25.251586+00:00 app[web.1]: => Run `rails server -h` for more startup options
2018-08-08T22:17:25.251588+00:00 app[web.1]: Puma starting in single mode...
2018-08-08T22:17:25.251589+00:00 app[web.1]: * Version 3.12.0 (ruby 2.5.1-p57), codename: Llamas in Pajamas
2018-08-08T22:17:25.251591+00:00 app[web.1]: * Min threads: 5, max threads: 5
2018-08-08T22:17:25.251592+00:00 app[web.1]: * Environment: production
2018-08-08T22:17:25.251594+00:00 app[web.1]: * Listening on tcp://0.0.0.0:10023
2018-08-08T22:17:25.251595+00:00 app[web.1]: Use Ctrl-C to stop
2018-08-08T22:17:25.251611+00:00 app[web.1]: I, [2018-08-08T22:17:25.251328 #4]  INFO -- : [cf68c09d-75ad-4c4
2018-08-08T22:17:25.277214+00:00 app[web.1]: I, [2018-08-08T22:17:25.272975 #4]  INFO -- : [cf68c09d-75ad-4c4
2018-08-08T22:17:25.296688+00:00 app[web.1]: I, [2018-08-08T22:17:25.296521 #4]  INFO -- : [cf68c09d-75ad-4c4
2018-08-08T22:17:25.297933+00:00 app[web.1]: I, [2018-08-08T22:17:25.297824 #4]  INFO -- : [cf68c09d-75ad-4c4
2018-08-08T22:17:25.303653+00:00 app[web.1]: I, [2018-08-08T22:17:25.303494 #4]  INFO -- : [cf68c09d-75ad-4c4
2018-08-08T22:17:25.804943+00:00 heroku[router]: at=info method=GET path="/favicon.ico" host=secret-reef-2487
```

You can also get the full stream of logs by running the logs command with the `--tail` flag option like this:

```
$ heroku logs --tail
```

Dyno sleeping and scaling

By default, new applications are deployed to a free dyno. Free apps will “sleep” to conserve resources. You can find more information about this behavior by reading about free dyno behavior (https://devcenter.heroku.com/articles/free-dyno-hours).

To avoid dyno sleeping, you can upgrade to a hobby or professional dyno type as described in the Dyno Types (https://devcenter.heroku.com/articles/dyno-types) article. For example, if you migrate your app to a professional dyno, you can easily scale it by running a command telling Heroku to execute a specific number of dynos, each running your web process type.

Run the Rails console

Heroku allows you to run commands in a one-off dyno (https://devcenter.heroku.com/articles/one-off-dynos) - scripts and applications that only need to be executed when needed - using the `heroku run` command. Use this to launch a Rails console process attached to your local terminal for experimenting in your app’s environment:

```
$ heroku run rails console
irb(main):001:0> puts 1+1
2
```

Another useful command for debugging is `$ heroku run bash` which will spin up a new dyno and give you access to a bash session.

Run Rake commands

Rake can be run as an attached process exactly like the console:

```
$ heroku run rake db:migrate
```

Configure your webserver

By default, your app’s web process runs `rails server`, which uses Puma in Rails 5. If you are upgrading an app you’ll need to add `puma` to your application `Gemfile`:

```
gem 'puma'
```

Then run

```
$ bundle install
```

Now you are ready to configure your app to use Puma. For this tutorial we will use the default `config/puma.rb` of that ships with Rails 5, but we recommend reading more about configuring your application for maximum performance by reading the Puma documentation (https://devcenter.heroku.com/articles/deploying-rails-applications-with-the-puma-web-server).

Finally you will need to tell Heroku how to run your Rails app by creating a `Procfile` in the root of your application directory.

Create a Procfile

Change the command used to launch your web process by creating a file called Procfile (https://devcenter.heroku.com/articles/procfile) and entering this:

In file `Procfile` write:

```
web: bundle exec puma -t 5:5 -p ${PORT:-3000} -e ${RACK_ENV:-development}
```

Note: This file must be named `Procfile` exactly.

We recommend generating a Puma config file based on our Puma documentation (https://devcenter.heroku.com/articles/deploying-rails-applications-with-the-puma-web-server) for maximum performance.

To use the Procfile locally, you can use `heroku local`.

In addition to running commands in your `Procfile` `heroku local` can also help you manage environment variables locally through a `.env` file. Set the local `RACK_ENV` to development in your environment and a `PORT` to connect to. Before pushing to Heroku you'll want to test with the `RACK_ENV` set to production since this is the environment your Heroku app will run in.

```
$ echo "RACK_ENV=development" >>.env
$ echo "PORT=3000" >> .env
```

Note: Another alternative to using environment variables locally with a `.env` file is the dotenv (https://github.com/bkeepers/dotenv) gem.

You'll also want to add `.env` to your `.gitignore` since this is for local environment setup.

```
$ echo ".env" >> .gitignore
$ git add .gitignore
$ git commit -m "add .env to .gitignore"
```

Test your Procfile locally using Foreman. You can now start your web server by running:

```
$ heroku local
[OKAY] Loaded ENV .env File as KEY=VALUE Format
11:06:35 AM web.1   |  [18878] Puma starting in cluster mode...
11:06:35 AM web.1   |  [18878] * Version 3.8.2 (ruby 2.5.1-p111), codename: Sassy Salamander
11:06:35 AM web.1   |  [18878] * Min threads: 5, max threads: 5
11:06:35 AM web.1   |  [18878] * Environment: development
11:06:35 AM web.1   |  [18878] * Process workers: 2
11:06:35 AM web.1   |  [18878] * Preloading application
```

Looks good, so press `Ctrl+C` to exit and you can deploy your changes to Heroku:

```
$ git add .
$ git commit -m "use puma via procfile"
$ git push heroku master
```

Check `ps`. You'll see that the web process uses your new command specifying Puma as the web server.

```
$ heroku ps
Free dyno hours quota remaining this month: 887h 34m (88%)
For more information on dyno sleeping and how to upgrade, see:
https://devcenter.heroku.com/articles/dyno-sleeping

=== web (Free): bundle exec puma -t 5:5 -p ${PORT:-3000} -e ${RACK_ENV:-development} (1)
web.1: starting 2018/08/08 17:18:06 -0500 (~ 7s ago)
```

The logs also reflect that we are now using Puma.

```
$ heroku logs
```

Rails asset pipeline

There are several options for invoking the Rails asset pipeline (http://guides.rubyonrails.org/asset_pipeline.html) when deploying to Heroku. For general information on the asset pipeline please see the Rails 3.1+ Asset Pipeline on Heroku Cedar (https://devcenter.heroku.com/articles/rails-asset-pipeline) article.

The `config.assets.initialize_on_precompile` option has been removed is and not needed for Rails 5. Also, any failure in asset compilation will now cause the push to fail. For Rails 5 asset pipeline support see the Ruby Support (https://devcenter.heroku.com/articles/ruby-support#rails-5-x-applications) page.

Troubleshooting

If you push up your app and it crashes (`heroku ps` shows state `crashed`), check your logs to find out what went wrong. Here are some common problems.

Runtime dependencies on development/test gems

If you're missing a gem when you deploy, check your Bundler groups. Heroku builds your app without the `development` or `test` groups, and if your app depends on a gem from one of these groups to run, you should move it out of the group.

One common example is using the RSpec tasks in your `Rakefile`. If you see this in your Heroku deploy:

```
$ heroku run rake -T
Running `bundle exec rake -T` attached to terminal... up, ps.3
rake aborted!
no such file to load -- rspec/core/rake_task
```

Then you've hit this problem. First, duplicate the problem locally:

```
$ bundle install --without development:test
…
$ bundle exec rake -T
rake aborted!
no such file to load -- rspec/core/rake_task
```

Note: The `--without` option on bundler is sticky. You can get rid of this option by running `bundle config --delete without`.

Now you can fix it by making these Rake tasks conditional on the gem load. For example:

```
begin
  require "rspec/core/rake_task"
  desc "Run all examples"
  RSpec::Core::RakeTask.new(:spec) do |t|
    t.rspec_opts = %w[--color]
    t.pattern = 'spec/**/*_spec.rb'
  end
rescue LoadError
end
```

Confirm it works locally, then push to Heroku.

Next steps

Congratulations! You have deployed your first Rails 5 application to Heroku. Here's some recommended reading:

- Visit the Ruby support category (https://devcenter.heroku.com/categories/ruby-support) to learn more about using Ruby and Rails on Heroku.
- The Deployment category (https://devcenter.heroku.com/categories/deployment) provides a variety of powerful integrations and features to help streamline and simplify your deployments.