

Hosting Ruby on Rails On Heroku Cloud

Albert Chiang
2023

1. Ruby, Rail

Ruby is an Object-Oriented Programming scripting language and Rails is the Model View Controller web app framework developed using the Ruby language. The purpose of this document is to show how to deploy a Ruby on Rails production website to Heroku cloud. This will be demonstrated via a “Thanksgiving Potluck Signup” app running on Heroku (a Platform-As-A-Service).



Learn about Ruby on Rails from class

Ruby - the language
Rails - a framework that makes creating web application requiring database access very easy



Develop and Test project on local lanton

install Ruby, Rails on my Macbook Pro
use Rails command to create scaffold
MySQL db for dev & test, PostgreSAL for prod
use rake to migrate, seed
use rails to host a local web server
use rake to test



Push to production on Heroku

modify Gemfile to point prod to PostgreSQL
git pushes to Heroku production on Heroku

2. How I did it & source code

2.1. Concept of the “Thanksgiving Potluck App”

Every year, my dear friend Neil invites others and I to his house for a Thanksgiving dinner potluck. Neil usually brines and bakes a turkey, but he needs help from his guests to bring other dishes. He also wanted a way for guests to sign up for dishes, but did not want the dishes to overlap. He used Evite to manage the email list and track who is coming, but tracking who was bringing what dishes was not easy to use. And being able to access the list via web or mobile web instantly via its own URI will make the app much easier to use.



Figure 1 Thanksgiving Potluck at Neil's

2.2. Development environment

My initial development was on Cloud9 cloud IDE, which basically allowed me to use my browser to development and run my Rails web app. But many of the Ruby on Rails books suggested that a Linux development environment with a traditional terminal is the best way to go, so I followed the collective advice and switched to developing the entire project on my Macbook Pro.

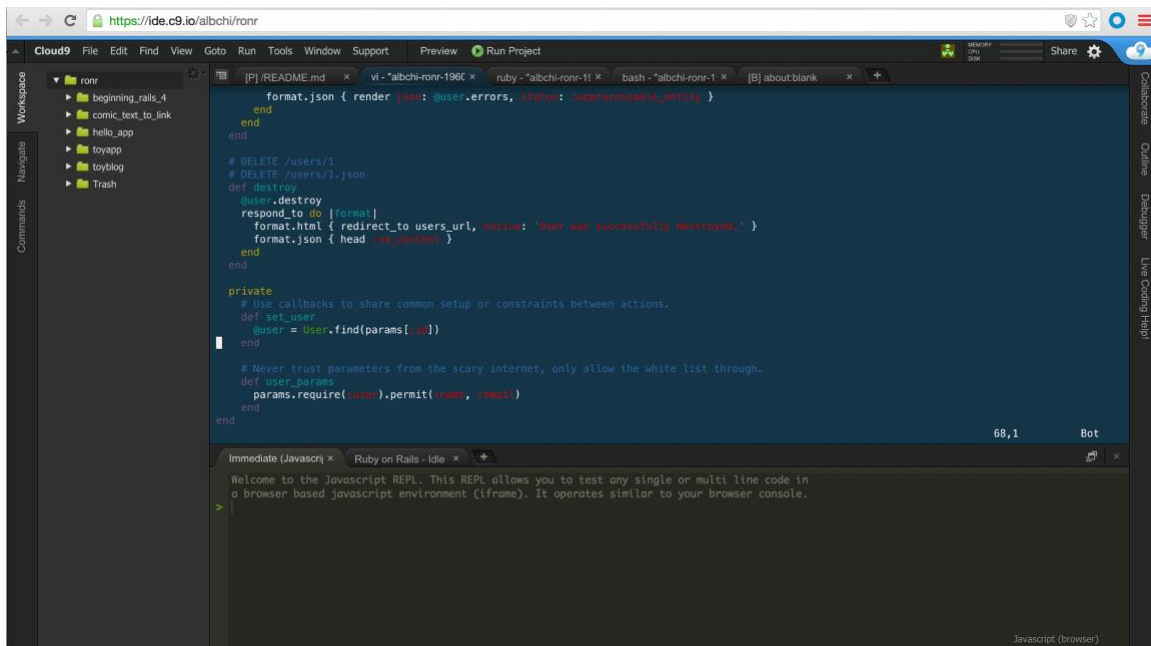


Figure 2 Initially developed on Cloud9 Cloud IDE

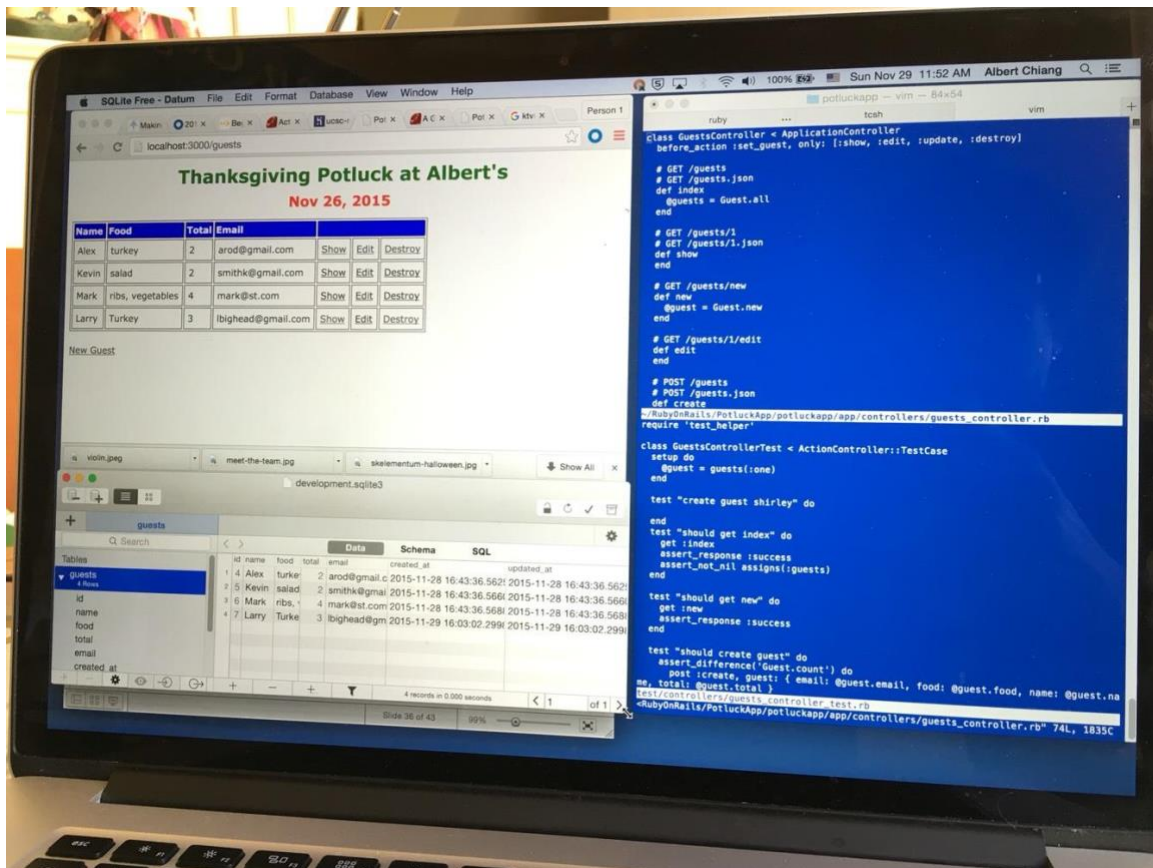


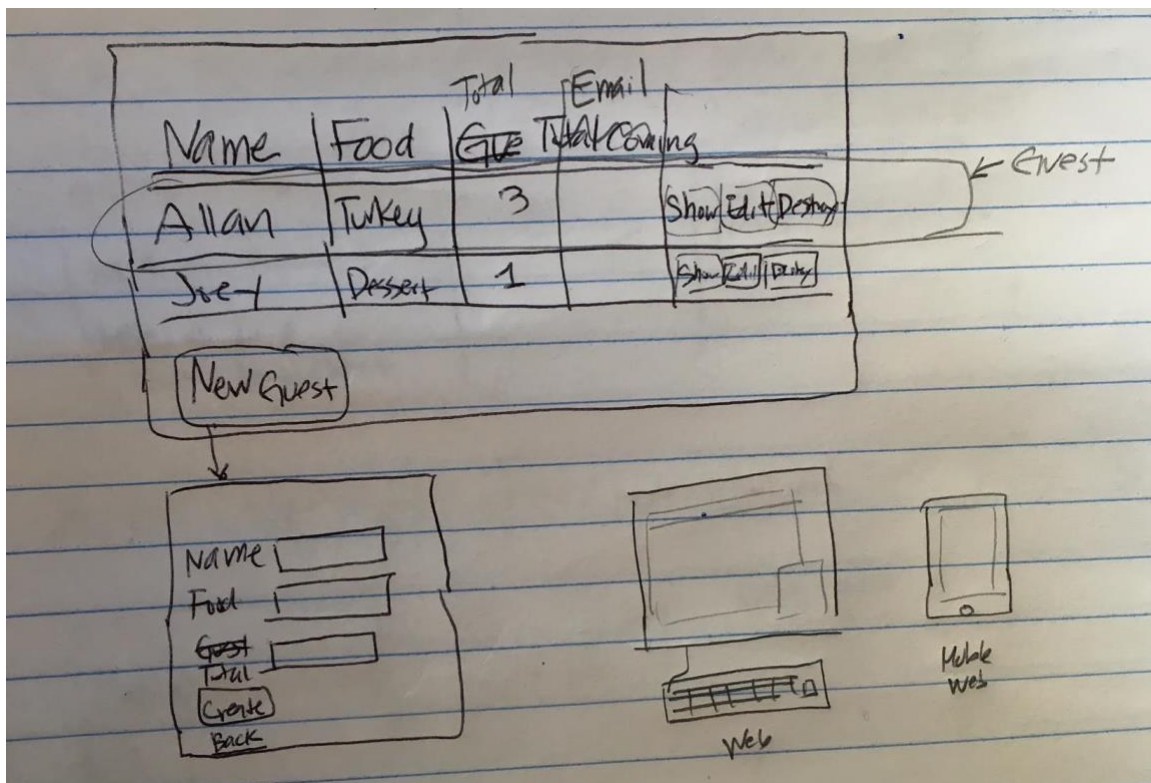
Figure 3 Development and test done on the Macbook Pro. A Linux terminal (R) was the main editor (vi) and Rails/Ruby command invocation. A web page hosted by the Rails web server (UL). SQLite Free - Datum can be used to look at the SQLite database (LR).

2.3. Initial data model

For Neil's Thanksgiving potluck, Neil wanted a simple guest registration system for the guests to use to register themselves. He wanted it to be accessible on both web and mobile. For each guest, Neil wanted to know these about his guests:

1. Name of guest
2. What food is being brought by the guest
3. Total number of people coming
4. Email of the guest

Further more, Neil said that it would be nice if the registration system can check for overlap in food brought.



2.4. Initial scaffold creation using Rails

Rails is a *framework* that eases web app development and testing. The framework was designed to bring order to a potentially messy process of developing and testing web apps. But the framework also potentially adds extra baggage, especially for simple web apps. One way Rails eases the creation of the framework is via automated scaffold generation. A Rails scaffold creates the necessary directory structure and pre-populated files.

```
el-MBP15-achiang-C02PX4X2G8WP-00014:PotluckApp albertchiang$ rails new potluckproj
create
create  README.rdoc
create  Rakefile
create  config.ru
create  .gitignore
```

Figure 4 Creating a new Rails project

```
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ rails generate scaffold
d Guest name:string food:string total:integer email:string
  invoke  active_record
  create  db/migrate/20151129202310_create_guests.rb
  create  app/models/guest.rb
  invoke  test_unit
  create  test/models/guest_test.rb
  create  test/fixtures/guests.yml
  invoke  resource_route
  route   resources :guests
  invoke  scaffold_controller
```

Figure 5 Creating a Rails scaffold for the data model mentioned above

```
[el-MBP15-achiang-C02PX4X2G8WP-00014:~/RubyOnRails/PotluckApp/potluckproj] albertchi
ang% rails server
=> Booting WEBrick
=> Rails 4.2.4 application starting in development on http://localhost:3000
=> Run 'rails server -h' for more startup options
=> Ctrl-C to shutdown server
[2015-11-29 12:26:18] INFO  WEBrick 1.3.1
[2015-11-29 12:26:18] INFO  ruby 2.2.1 (2015-02-26) [x86_64-darwin14]
[2015-11-29 12:26:18] INFO  WEBrick::HTTPServer#start: pid=80786 port=3000
```

Figure 6 Starting a local web server using Rails

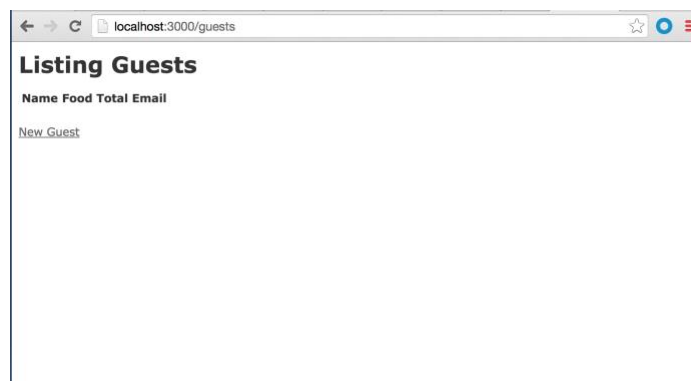


Figure 7 A working web app just using 1 Rails command!

2.5. Seeding the database

Now that we have a web app running, the framework of what to show (guest information) and how to manage it (edit, destroy, show) is built in and ready to use. But now we would like to have data in this web app. What is the fastest way to pre-populate data into it? We can populate the web app with data by using seed.

```
# This file should contain all the record creation needed to seed the database with
its default values.
# The data can then be loaded with the rake db:seed (or created alongside the db wit
h db:setup).
#
# Examples:
#
#   cities = City.create([ { name: 'Chicago' }, { name: 'Copenhagen' } ])
#   Mayor.create(name: 'Emanuel', city: cities.first)
Guest.delete_all
Guest.create(name: "Al", food: "Turkey", total: 2, email: "alz@gmail.com")
Guest.create!(name: "Beth", food: "pie and dessert", total: 4, email: "bhone@gmail.c
om")
Guest.create!(name: "Zack", food: "beer", total: 3, email: "zackary@gmail.com")

"db/seeds.rb" 11L, 605C
```

Figure 8 Create initial guest seed data

```
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ rake db:seed
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$
```

Figure 9 Rake is a Rails “make” utility to read db/seeds.rb into db/development.sqlite3

Figure 10 The seed data is now seen in the web app

2.6. Data migration

The data model was created using Rails and resides in the db/migrate directory as a Ruby file. But the Rails framework eventually will interact with a MySQL database. Rails has one built-in called SQLite3. In order to “migrate” the Ruby version of the data into SQLite3, the “rake db:migrate” is invoked. To double check the output of the migration into an sqlite3 file, I used SQLite Free – Datum.

	id	name	food	total	email	created_at	updated_at
1	4	Alex	turkey		2 arod@gmail.com	2015-11-28 16:43:36.562535	2015-11-28 16:43:36.562535
2	5	Kevin	salad		2 smithk@gmail.com	2015-11-28 16:43:36.566038	2015-11-28 16:43:36.566038
3	6	Mark	ribs, vegeta		4 mark@st.com	2015-11-28 16:43:36.568888	2015-11-28 16:43:36.568888
4	7	Larry	Turkey		3 lbighead@gmail.com	2015-11-29 16:03:02.299846	2015-11-29 16:03:02.299846

Figure 11 SQLite Free - Datum Running On Macbook Pro to Read sqlite3 file

2.7. Customize the User Interface View

The Rails scaffold command creates a basic look on the View. Neil wanted a bit more customization in the web app – which means changes to HTML and CSS. In Rails, HTML is produced from processing Embedded Ruby files (erb) and CSS is produced from processing Sassy CSS (SCSS) files.

```
<p id="notice"><%= notice %></p>

<h1>Thanksgiving Potluck at Neil's </h1>
<h2>Nov 26, 2015</h2>

<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Food</th>
      <th>Total</th>
      <th>Email</th>
      <th colspan="3"></th>
    </tr>
  </thead>

  <tbody>
    <tr>
      <td>Al</td>
      <td>Turkey</td>
      <td>2</td>
      <td>alz@gmail.com</td>
      <td>Show</td>
      <td>Edit</td>
      <td>Destroy</td>
    </tr>
    <tr>
      <td>Beth</td>
      <td>pie and dessert</td>
      <td>4</td>
      <td>bhome@gmail.com</td>
      <td>Show</td>
      <td>Edit</td>
      <td>Destroy</td>
    </tr>
    <tr>
      <td>Zack</td>
      <td>beer</td>
      <td>3</td>
      <td>zackary@gmail.com</td>
      <td>Show</td>
      <td>Edit</td>
      <td>Destroy</td>
    </tr>
  </tbody>
</table>

<a href="#new_guest">New Guest</a>
```

Figure 12 Add a better title with dates into the Embedded RB (which will become HTML later)

```
$h1_color: green;
$h2_color: red;

h1 {
  color: $h1_color;
  text-align: center;
}
h2 {
  color: $h2_color;
  text-align: center;
}

table, th, td {
  border: 1px solid black;
}

a {
  color: blue;
  text-decoration: underline;
}
```

Figure 13 SCSS is a super CSS that is pre-processed by Ruby on Rails.

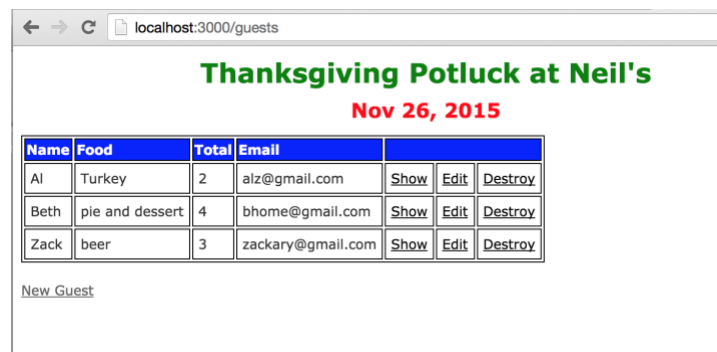


Figure 14 Results of enhancing the ERB and SCSS files

2.8. Validating the data model

Neil preferred that guests not bring duplicate food. He wanted the web app to warn a guest if the food to be brought was already registered. This is easily done in Rail using the “validates” concept. Neil also want to limit the number of total guests, so a limit of 4 per guest is imposed by Neil.

```
class Guest < ActiveRecord::Base
  validates_uniqueness_of :food, :case_sensitive => false
  validates :email, uniqueness: true
  validates_numericality_of :total, :greater_than => 0, :less_than_or_equal_to => 4
end
~
~
~
~
~
~
~
~
~
~
~/RubyOnRails/PotluckApp/potluckproj/app/models/guest.rb" 5L, 226C written
```

Figure 15 Neil's request to not have overlapping food can be implemented here

The screenshot shows a web browser window with the address bar displaying 'localhost:3000/guests'. The page title is 'New Guest'. A red error message box at the top states: '1 error prohibited this guest from being saved:'. Below this, a list of errors shows 'Food has already been taken'. The form fields are: 'Name' (containing 'Steve'), 'Food' (containing 'turkey' and highlighted with a red border), 'Total' (containing '1'), and 'Email' (containing 'scox@yahoo.com'). A 'Create Guest' button is at the bottom. A 'Back' link is visible at the very bottom of the page.

Figure 16 Guests will be told that the food they will bring has already been claimed

2.9. Route configuration

Ruby on Rails uses a Models View Control (MVC) design pattern. Which means that web app requests from users are sent to the Controller. But in Rails, before the user request is sent to the Controller, it is first routed through a Rails router.

```
Rails.application.routes.draw do
  resources :guests
  # The priority is based upon order of creation: first created -> highest priority.
  # See how all your routes lay out with "rake routes".

  # You can have the root of your site routed with "root"
  root 'guests#index'

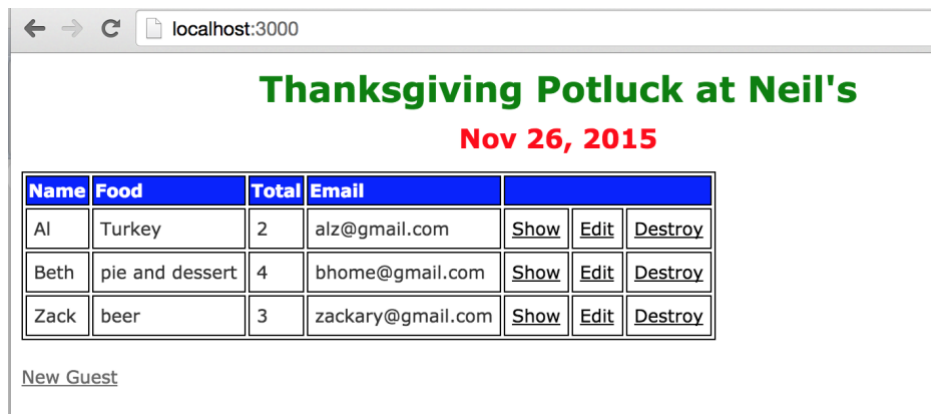
  # Example of regular route:
  # get 'products/:id' => 'catalog#view'

  # Example of named route that can be invoked with purchase_url(id: product.id)
  # get 'products/:id/purchase' => 'catalog#purchase', as: :purchase

  # Example resource route (maps HTTP verbs to controller actions automatically):
  # resources :products

"config/routes.rb" [Modified] line 7 of 57 --12%-- col 13
```

Figure 17 User request to root (localhost:3000/) is directed to guest controller, index action



← → ↻ localhost:3000

Thanksgiving Potluck at Neil's

Nov 26, 2015

Name	Food	Total	Email			
Al	Turkey	2	alz@gmail.com	Show	Edit	Destroy
Beth	pie and dessert	4	bhome@gmail.com	Show	Edit	Destroy
Zack	beer	3	zackary@gmail.com	Show	Edit	Destroy

[New Guest](#)

Figure 18 Users can now go to localhost:3000 instead of localhost:3000/guests... needed later for Heroku

2.10. App testing

Rails was design with test in mind. With Rails, tests are easy to write, easy to run, and offers powerful abilities for test automation.

```
require 'test_helper'

class GuestsControllerTest < ActionController::TestCase
  setup do
    @guest = guests(:one)
  end

  test "should get index" do
    get :index
    assert_response :success
    assert_not_nil assigns(:guests)
  end

  test "should get new" do
    get :new
    assert_response :success
  end

  test "should create guest" do
    assert_difference('Guest.count') do
      post :create, guest: { email: @guest.email, food: @guest.food, name: @guest.name, total: @guest.total }
    end

    assert_redirected_to guest_path(assigns(:guest))
  end

  test "should show guest" do
    get :show, id: @guest
    assert_response :success
  end

  test "should get edit" do
    get :edit, id: @guest
    assert_response :success
  end

  test "should update guest" do
    patch :update, id: @guest, guest: { email: @guest.email, food: @guest.food, name: @guest.name, total: @guest.total }
    assert_redirected_to guest_path(assigns(:guest))
  end

  test "should destroy guest" do
    assert_difference('Guest.count', -1) do
      delete :destroy, id: @guest
    end

    assert_redirected_to guests_path
  end
end

~
~
~
~
~
~

"test/controllers/guests_controller_test.rb" 49L, 1128C
```

Figure 19 A Rails test class for the controller

```
(See full trace by running task with --trace)
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ rake test test/controllers/guests_controller_test.rb test_should_get_new
Run options: -n test_should_get_new --seed 36524

# Running:
.

Finished in 0.169070s, 5.9147 runs/s, 5.9147 assertions/s.

1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$
```

Figure 20 Rail test case invocation and results

3. Deployment to The World - Using Heroku PAAS

Now that we have developed and tested our web app on my Macbook Pro, it is time to let other use it – by pushing the web app to production. Which means Neil and his guests can starting using the app from any web browser. But in order to push my Ruby on Rails environment to production, I need a way to host my Ruby on Rails application. This is where Heroku can help.

3.1. What is Git

Git is a source code revision control system. Which allows you to store your source code into a safe place – called a repository. Git in itself is a great software development tool to keep backups of your project , as well as keep revision control so that you can revert to previous version of your project. But must we use Git?

The answer is yes – because we are using Heroku to push our project to production.

```
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ git init
Initialized empty Git repository in /Users/albertchiang/RubyOnRails/PotluckApp/potluckproj/.git/
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ git add .
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ git commit -m "initial"
[master (root-commit) a97f82d] initial
75 files changed, 1391 insertions(+)
create mode 100644 .gitignore
create mode 100644 Gemfile
create mode 100644 Gemfile.lock
create mode 100644 README.rdoc
```

Figure 21 Using git to create a repository on my Macbook Pro

3.2. Pushing to Heroku

Heroku offers Platform As A Service (PAAS) to deploy applications. In order to productize my web app, I first applied for a free Heroku account. Once established, the Linux command line can be used to control interaction with the Heroku Command Line Interface (CLI).

```
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ heroku login
Enter your Heroku credentials.
Email: albchi@gmail.com
Password (typing will be hidden):
Logged in as albchi@gmail.com
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ heroku create neils-xgiving
! You've reached the limit of 5 apps for unverified accounts.
! Delete some apps or add a credit card to verify your account.
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ heroku create neils-xgiving
Creating neils-xgiving... done, stack is cedar-14
https://neils-xgiving.herokuapp.com/ | https://git.heroku.com/neils-xgiving.git
Git remote heroku added
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ vi Gemfile
```

Figure 22 Heroku CLI from my Macbook Pro terminal

3.

```
Source 'https://rubygems.org'

# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '4.2.4'
# Use sqlite3 as the database for Active Record
gem 'sqlite3'
# Use SCSS for stylesheets
gem 'sqlite3', group: [:development, :test]
gem 'rails_12factor', group: :production
gem 'pg', group: :production

"Gemfile" 52L, 1618C
```

Figure 23 Heroku does not support SQLite, so the Gemfile needs to be modified

```
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ git add Gemfile
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ git add Gemfile.lock
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ git commit -m "Gemfile for Heroku"
[master 389de72] Gemfile for Heroku
 2 files changed, 13 insertions(+)
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ git push heroku master
Counting objects: 91, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (84/84), done.
Writing objects: 100% (91/91), 21.94 KiB | 0 bytes/s, done.
Total 91 (delta 5), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
```

Figure 24 Pushing your local git repository to Heroku

4.

```
remote:
remote: -----> Discovering process types
remote:      Procfile declares types      -> (none)
remote:      Default types for buildpack -> console, rake, web, worker
remote:
remote: -----> Compressing... done, 29.2MB
remote: -----> Launching... done, v5
remote:      https://neils-xgiving.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy.... done.
To https://git.heroku.com/neils-xgiving.git
* [new branch]      master -> master
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$
```

Figure 25 Success upload into Heroku

```
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$ heroku run rake db:migrate
Running rake db:migrate on neils-xgiving... up, run.7646
(14.1ms) CREATE TABLE "schema_migrations" ("version" character varying NOT NULL)
(4.6ms) CREATE UNIQUE INDEX "unique_schema_migrations" ON "schema_migrations" ("version")
ActiveRecord::SchemaMigration Load (1.2ms) SELECT "schema_migrations".* FROM "schema_migrations"
Migrating to CreateGuests (20151129202310)
(1.1ms) BEGIN
== 20151129202310 CreateGuests: migrating =====
-- create_table(:guests)
(8.2ms) CREATE TABLE "guests" ("id" serial primary key, "name" character varying, "food" character varying, "total" integer
, "email" character varying, "created_at" timestamp NOT NULL, "updated_at" timestamp NOT NULL)
-> 0.0093s
== 20151129202310 CreateGuests: migrated (0.0094s) =====

SQL (1.3ms) INSERT INTO "schema_migrations" ("version") VALUES ($1) [{"version", "20151129202310"}]
(2.8ms) COMMIT
el-MBP15-achiang-C02PX4X2G8WP-00014:potluckproj albertchiang$
```

Figure 26 Rail migration once development environment is in Heroku

3.3. Differences Between Development and Production - PostgreSQL instead of SQLite

During the development of our Rails web app on my Macbook Pro, SQLite3 was the default Relational Database System (RDB) used with Ruby on Rails. SQLite3 is an embedded RDB, in contrast to MySQL – a full RDB environment. MySQL is part of Oracle, which leaves the Open Source community a bit nervous because Oracle makes money from selling its own Oracle Database.

For those looking for pure open source RDB, PostgreSQL is the popular choice. Moreover, PostgreSQL is a Object Relational Database. Hence it is not surprising that Heroku supports PostgreSQL over SQLite.


```
Source 'https://rubygems.org'
```

```
# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'  
gem 'rails', '4.2.4'  
# Use sqlite3 as the database for Active Record  
gem 'sqlite3'  
# Use SCSS for stylesheets  
gem 'sqlite3', group: [:development, :test]  
gem 'rails_12factor', group: :production  
gem 'pg', group: :production
```

```
"Gemfile" 52L, 1618C
```

3.4. Production Web View



Figure 27 Web app view

Thanksgiving Potluck at Neil's

Nov 26, 2015

Name	Food	Total	Email			
Al	Turkey	2	alz@gmail.com	Show	Edit	Destroy
Beth	pie and dessert	4	bhome@gmail.com	Show	Edit	Destroy
Zack	beer	3	zackary@gmail.com	Show	Edit	Destroy

[New Guest](#)

Figure 28 iPhone web app

4. Conclusion

Ruby is an object oriented scripting language targeted for web applications. Rails add a framework around Ruby so that code with specification functionality has a nature home in the framework. Rails adopt the MVC design pattern. Ruby on Rails provides a proven and robust framework for web app development, testing, and production. Within a few short Rails command, a full web app is created. Heroku cloud was picked as the public cloud hosting platform.