TP Final

Laboratorio de Sistemas Operativos y Redes

Alumnos: Leandro Gómez, Mariano Linares
Profesor: José Luis Di Biase
Fecha: 05 de Diciembre de 2013

Introducción

Vamos a montar una pequeña aplicación en Heroku, hecha en Rails. Para eso, vamos a tener un repo en Github, el cual tendría un servidor de integración continua como Travis para correr algunos tests que tenga la app. Además, para la base de datos vamos a usar MongoDB.

Para la app vamos a hacer algo mínimo (quizas un juego como Travian) y centrarnos en configurar todo el stack en el que corre la aplicación documentando el proceso.

Entonces, lo que abarcaría sería:

- Github
    - Cómo crear y configurar una cuenta de github (claves ssh por ej)
    - Qué cliente de git utilizar y como configurarlo

- Travis
    - Cómo configuro travis para mi aplicación (que variantes hay, que puedo ver en travis, como se usa)

- Heroku
    - Cómo me consigo una cuenta de heroku

- Rails + MongoDB (minimal app)
    - Que necesito descargar, como habilito ej mongo, que modificaciones hago en mi aplicación, como hago el deploy, etc.

Github

Cómo crear y configurar una cuenta de github (claves ssh por ej)

Para crear una cuenta se debe ir a **https://github.com/join** y completar el formulario para registrarse.
Para este ejemplo, creamos la cuenta con username: **labo2013** y password: **2013labo**

Qué cliente de git utilizar y como configurarlo

Como cliente vamos a usar **git** desde una terminal en linux. Para instalarlo escribimos el comando **sudo apt-get install git** . Una vez instalado, debemos configurar nuestra cuenta en este cliente. Para eso, seteamos nuestro username y mail de la siguiente manera:

- ➢ git config --global user.name “labo2013”
- ➢ git config --global user.email nuestromail@example.com

Cómo crear un repositorio nuevo para subir el código

Ir a **https://github.com/new** y llenar los campos *'**Repository name**'* y *'**Description**'* y dar a *'crear repositorio'.* En este caso usaremos `rainer` como nombre del repositorio.

Creamos una carpeta en nuestro directorio, donde alojar el repositorio local y corremos los siguientes comandos en una consola para inicializar el repositorio.

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/labo2013/railner.git
git push -u origin master
```

Al momento del 'push', github nos pedirá nuestras credenciales (user y password) para autorizarnos. Una vez hecho esto, ya estará inicializado nuestro repositorio.

Rails + MongoDB (minimal app)

Que necesito descargar y como empiezo

Para empezar, descargamos desde la terminal el intérprete de ruby: **sudo apt-get install ruby**
Instalamos rails para armar nuestra aplicación web: **gem install rails**
Para crear la base de la aplicación rails correr los siguientes comandos sobre el directorio que creamos anteriormente.

rails new ../railner --skip-active-record
rails server

Ahora agregamos subimos los archivos creados al repositorio de github:

git add .
git commit -m "Add rails app structure"
git push

Cómo habilito Mongo en Rails

Primero, agregar al Gemfile las gems:

gem 'mongo_mapper'
gem 'bson_ext'

Crear los siguientes archivos:

config/initialize/mongo.rb

```
MongoMapper.connection = Mongo::Connection.new('localhost', 27017)
MongoMapper.database = "#myapp-#{Rails.env}"

if defined?(PhusionPassenger)
  PhusionPassenger.on_event(:starting_worker_process) do |forked|
    MongoMapper.connection.connect if forked
  end
end
```

lib/tasks/mongo.rake

```
namespace :db do
  namespace :test do
    task :prepare do
      # Stub out for MongoDB
    end
  end
end
```

Configurar la aplicacion sin mongodb:

rails new ../railner
rails server

Ahora agregamos subimos los archivos creados al repositorio de github:

git add .
git commit -m "Add rails app structure without mongodb"
git push

Test

Instalamos rspec (https://relishapp.com/rspec/rspec-rails/docs/)

Agregamos la siguiente linea al Gemfile;

```
group :test, :development do
  gem "rspec-rails", "~> 2.4"
end
```

Y luego corremos el comando:

```
$ rails generate rspec:install
```

Y para correr los test:

```
$ rspec
```

Travis

Travis CI is a hosted continuous integration service for the open source community

Cómo configuro travis para mi aplicación rails hosteada en github

Vamos a https://travis-ci.org/ y tocamos en 'Sign in with github' y dar a ***allow access*** en la página de github.

Luego en la página https://travis-ci.org/profile, donde se nos mostrará cada uno de nuestros repositorios en Github, junto con un botón **On/Off** a su lado.

Cambiar el boton de estado **OFF** a **ON** si es necesario del repositorio a integrar.

Luego hacer click en configuración () lo cual redireccionará a una página de github (https://github.com/labo2013/<nombre_del_projecto>/settings/hooks).

Buscar en la lista de ***Available Service Hooks*** la entrada 'travis' y hacer click. Esto nos mostrará un formulario.

Llenar los campos:
user: nombre de usuario de github
token: lo pueden obtener https://travis-ci.org/profile/<nombre del repo>/profile
Nos aseguramos de que la casilla **active** este checkeada.
Click en **Update settings**

Click **Test Hook**

Para aplicaciones rails, travis-ci conoce cómo los distintos steps standards. Pero lo mejor es configurar un archivo ***.travis.yml*** segun se indica para proyectos ruby (ver aqui)

Qué variantes hay

Sauce Labs para correr la app y tests en distintas plataformas. Site: https://saucelabs.com/

Heroku

Cómo me consigo una cuenta

Crear una cuenta en https://id.heroku.com/signup
Ir a https://toolbelt.heroku.com/debian yseguir los pasos:
$ wget -qO- https://toolbelt.heroku.com/install-ubuntu.sh | sh

Una vez instalado, desde la consola:

```
$ heroku login
```

En este momento, ingresamos nuestros datos que pusimos al registrarnos.

```
$ heroku create
$ heroku keys:add ~/.ssh/id_rsa.pub
```

Agregar la gema de postgres en el environment de production, el cual usará heroku:

```
gem pg in Gemfile
$ heroku config:set RACK_ENV=production RAILS_ENV=production

$ git push heroku master
```

Problemas

Integración de MongoDB con Rails.

Nos trajo problemas para crear modelos nuevos. No tuvimos tiempo de probar más configuraciones para hacerlo andar, aunque nos quedamos con las ganas de seguir probando.

Decidimos usar PostgreSQL, ya que simplifica la integración en Heroku.

Asignamos Sqlite para correr en modo desarrollo, y postgresql para deployar en heroku, ya que sino implicaba instalar postgresql en la computadora en la que probamos.
http://stackoverflow.com/questions/18864112/getting-error-while-installing-postgresql-gem

Fuentes

http://docs.mongodb.org/ecosystem/tutorial/getting-started-with-ruby-on-rails-3/