Proyecto Ruby on Rails

Eleazar Borja Hernández Padilla [1]

11 de julio de 2012

[1]DEIOC Universidad de La Laguna

Índice general

1. Instalación del entorno **7**
1.1. Instalación de RVM (Ruby Version Manager) 7
1.2. Instalación de Ruby 7
1.3. Instalación de la gema “Rails“ 8

2. Despliegue en Heroku **9**
2.1. Empezar con Heroku 9
2.1.1. Crear cuenta en Heroku 9
2.1.2. Instalar Heroku Toolbelt 9
2.2. Despliegue con Heroku 10
2.2.1. 10

Índice de figuras

Índice de cuadros

A Juana

For it is in teaching that we learn
And it is in understanding that we are understood

Agradecimientos/Acknowledgments

I'd like to thank Tom Christiansen, Damian Conway, Simon Cozens, Francois Desarmenien, Richard Foley, Jeffrey E.F. Friedl, Joseph N. Hall, Tim Jennes, Andy Lester, Allison Randall, Randal L. Schwartz, Michael Schwern, Peter Scott, Sriram Srinivasan, Linconl Stein, Dan Sugalski, Leopold Tötsch, Nathan Torkington and Larry Wall for their books and/or their modules. To the Perl Community.

Special thanks to Larry Wall for giving us Perl.

A mis alumnos de Programación Paralelo II en el segundo curso de la Ingeniería Superior Informática en la Universidad de La Laguna

Capítulo 1

Instalación del entorno

1.1. Instalación de RVM (Ruby Version Manager)

Para instalar RVM en un sistema Linux teclearemos en un terminal:

```
bash <<(curl -s https://rvm.beginrescueend.com/install/rvm)
```

Se trata de un script en bash que clona RVM de su repositorio en GitHub. Requiere que algunas dependencias estén instaladas. Para ello teclearemos previamente en un terminal

```
sudo aptitude install build-essential git-core curl
```

El contenido de RVM se ubicara en el directorio `~/.rvm`. Para que nuestra terminal pueda trabajar con RVM sin problemas debemos agregar la siguiente línea al fichero `~/.bash_profile`:

```
'[[ -s "$HOME/.rvm/scripts/rvm" ]] && . "$HOME/.rvm/scripts/rvm" # Load RVM function'
```

Posteriormente habrá que recargar la shell. Para ello abrimos una nueva terminal y tecleamos:

```
source .bash_profile
source ~/.rvm/scripts/rvm
```

Si la instalación y la configuración han sido exitosas, al teclear

```
type rvm | head -1
```

nos debe aparecer rvm: es una función. En caso contrario la instalación no habrá finalizado correctamente.

1.2. Instalación de Ruby

Una vez instalado RVM procedemos a instalar una versión de Ruby. En nuestro caso instalamos la 1.9.3 tal que así:

```
rvm install 1.9.3
```

Este comando descargará las fuentes de Ruby, las extraerá en `~/.rvm`, lo compilará, y finalmente lo instalará. Ahora hay que indicarle a RVM la versión de Ruby a usar, en este caso la 1.9.3

```
rvm use 1.9.3
```

Para comprobar la versión que estamos usando tecleamos en un terminal:

```
ruby -v
```

y nos deberá aparecer un mensaje similar a este:

```
ruby 1.9.3p194 (2012-04-20 revision 35410) [i686-linux]
```

Si queremos usar esta versión por defecto debemos teclear:

```
rvm use 1.9.3 --default
```

Para que estos cambios sean permanentes en el sistema debemos editar el fichero `~/.bashrc` y colocar al final del mismo la siguiente línea:

```
source $HOME/.rvm/scripts/rvm
```

1.3. Instalación de la gema "Rails"

La versión 1.9.3 de Ruby incluye rubygems con lo cual lo primero que debemos hacer es actualizar tecleando en un terminal:

```
gem update --system<
```

Una vez actualizado rubygems, procederemos a instalar Rails, en nuestro caso lo instalaremos como una gema más:

```
gem install rails
```

Finalmente si queremos ver la versión de Rails instalada lo podremos comprobar en un terminal mediante el siguiente comando:

```
rails -v
```

Capítulo 2

Despliegue en Heroku

2.1. Empezar con Heroku

2.1.1. Crear cuenta en Heroku

Lo primero que tenemos que hacer es crearnos una cuenta en Heroku. Para ello nos dirigimos a la web de Heroku y pulsamos en **Login** y pulsar en **Sign up**.Nos pedirá que introduzcamos una cuenta de correo y pulsemos en **Sign up**. Ahora tendremos que revisar nuestra bandeja de entrada y pulsar el link que aparece en el correo recibido para poder confirmar nuestra cuenta. Una vez confirmada, tendremos nuestra cuenta creada.

2.1.2. Instalar Heroku Toolbelt

El siguiente paso es instalar Heroku Toolbelt en nuestro Sistema Operativo. Heroku Toolbelt está compuesto de un cliente de Heroku y Foreman (opción que nos facilita correr nuestras aplicaciones localmente). En nuestro caso, que estamos trabajando sobre Linux y también sobre Ruby podemos instalarlo de la siguiente manera:

```
gem install heroku foreman
```

Una vez instalado, para tener acceso a la línea de comando de heroku desde tu shell habrá que hacer un ¨login¨:

```
$ heroku login
Enter your Heroku credentials.
Email: adam@example.com
Password:
```

Si no tenemos ninguna clave de ssh, nos preguntará que si deseamos crear una:

```
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
```

Ahora ya estamos preparados para crear nuestra primera aplicación en Heroku. Para ello nos tendremos que colocar en el directorio donde tengamos la aplicación:

```
$ cd ~/myapp
```

y una vez en dicho directorio teclear:

```
$ heroku create
Creating stark-fog-398... done, stack is cedar
http://stark-fog-398.herokuapp.com/ | git@heroku.com:stark-fog-398.git
Git remote heroku added
```

2.2. Despliegue con Heroku

Heroku nos permite desplegar aplicaciones desarrolladas en lenguajes tales como Java, Phyton, Ruby, etc. En nuestro caso, haremos el despligue en Heroku para una aplicación desarrollada en Rails. Una vez hecho el login en Heroku (`heroku login`), nos colocaremos en el directorio donde esté nuestra aplicación y modificaremos nuestro `Gemfile` cambiando esta línea:

```
gem 'sqlite3'
```

por esta otra:

```
gem 'pg'
```

Ahora reinstalamos nuestras dependencias tecleando:

```
 $ bundle install
```

Esta modificación se hace debido a que Rails suele trabajar con **Sqlite3** y Heroku lo hace con **PostgreSQL**.

*Nota: Para que no falle la instalación de `pg` hay que tener instalada la librería libpq-dev

2.2.1.

Continuaremos almacenando nuestra aplicación en Git, para ello teclearemos:

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

Ahora creamos la aplicación en la pila Cedar de Heroku:

```
$ heroku create
Creating severe-mountain-793... done, stack is cedar
http://severe-mountain-793.herokuapp.com/ | git@heroku.com:severe-mountain-793.git
Git remote heroku added
```

Y realizamos el despliegue de nuestro código tecleando:

```
$ git push heroku master
```

y nos aparecerá algo similar a esto:

```
Counting objects: 67, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (52/52), done.
Writing objects: 100% (67/67), 86.33 KiB, done.
Total 67 (delta 5), reused 0 (delta 0)

-----> Heroku receiving push
-----> Rails app detected
-----> Installing dependencies using Bundler version 1.1
       Checking for unresolved dependencies.
       Unresolved dependencies detected.
       Running: bundle install --without development:test --path vendor/bundle --deployment
       Fetching source index for http://rubygems.org/
       Installing rake (0.8.7)
       ...
```

```
       Installing rails (3.0.5)
       Your bundle is complete! It was installed into ./vendor/bundle
-----> Rails plugin injection
       Injecting rails_log_stdout
       Injecting rails3_serve_static_assets
-----> Discovering process types
       Procfile declares types -> (none)
       Default types for Rails -> console, rake, web, worker
-----> Compiled slug size is 8.3MB
-----> Launching... done, v5
       http://severe-mountain-793.herokuapp.com deployed to Heroku

To git@heroku.com:severe-mountain-793.git
 * [new branch]      master -> master
```

Antes de comprobar nuestra aplicación en la web, debemos comprobar los procesos de la aplicación de la siguiente manera:

```
 $ heroku ps
=== web: ‘bundle exec rails server -p $PORT‘
web.1: up for 5s
```

Ahora es el momento de realizar la migración de la base de datos, para que empieze a funcionar con PostgreSQL:

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

Por defecto la aplicación correrá con el `rails server` que usa Webrick. Esto es conveniente para el testeo, pero para producción necesitaremos el uso de un servidor más robusto. Para ello incluimos en nuestro `Gemfile`

```
 gem 'thin'
```

ya que en la propia página de Heroku está recomendado. Completamos la instalación local de la gema tecleando en consola `bundle install`

Ahora para cambiar el comando que usamos para lanzar los procesos web creamos un fichero llamado P̈rocfileë insertamos esto:

```
 web: bundle exec rails server thin -p $PORT -e $RACK_ENV
```

configuramos la variable de entorno:

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

Ahora probaremos el `Procfile` localmente usando **“Foreman”**. Para ello tecleamos:

```
 $ foreman start
11:35:11 web.1     | started with pid 3007
11:35:14 web.1     | => Booting thin
11:35:14 web.1     | => Rails 3.0.4 application starting in development on http://0.0.0.0:5000
11:35:14 web.1     | => Call with -d to detach
11:35:14 web.1     | => Ctrl-C to shutdown server
11:35:15 web.1     | >> Thin web server (v1.2.8 codename Black Keys)
11:35:15 web.1     | >> Maximum connections set to 1024
11:35:15 web.1     | >> Listening on 0.0.0.0:5000, CTRL+C to stop
```

Si nos sale algo como lo mostrado habrá ido todo correcto. Presionamos Ctrl + C para salir y desplegamos los cambios en heroku:

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

Al comprobar el log, veremos que ahora usamos **“Thin”**

```
 $ heroku logs
2012-07-11T09:39:36+00:00 heroku[web.1]: State changed from down to starting
2012-07-11T09:39:38+00:00 heroku[web.1]: Starting process with command ‘bundle exec rails serv
2012-07-11T09:39:44+00:00 app[web.1]: => Booting Thin
2012-07-11T09:39:44+00:00 app[web.1]: => Rails 3.2.6 application starting in production on htt
2012-07-11T09:39:44+00:00 app[web.1]: => Call with -d to detach
2012-07-11T09:39:44+00:00 app[web.1]: => Ctrl-C to shutdown server
2012-07-11T09:39:44+00:00 app[web.1]: Connecting to database specified by DATABASE_URL
2012-07-11T09:39:44+00:00 app[web.1]: >> Thin web server (v1.4.1 codename Chromeo)
2012-07-11T09:39:44+00:00 app[web.1]: >> Maximum connections set to 1024
2012-07-11T09:39:44+00:00 app[web.1]: >> Listening on 0.0.0.0:22617, CTRL+C to stop
2012-07-11T09:39:45+00:00 heroku[web.1]: State changed from starting to up
```

Apéndice

Instrucciones Para la Carga de Módulos en la ETSII

Cuando entra a una de las máquinas de los laboratorios de la ETSII encontrará montado el directorio `/soft/perl5lib/` y en él algunas distribuciones de módulos Perl que he instalado.

```
export MANPATH=$MANPATH:/soft/perl5lib/man/
export PERL5LIB=/soft/perl5lib/lib/perl5:/soft/perl5lib/lib/perl/5.10.0/:/soft/perl5lib/share/
export PATH=.:/home/casiano/bin:$PATH:/soft/perl5lib/bin
```

Visite esta página de vez en cuando. Es posible que añada algún nuevo camino de búsqueda de librerías y/o ejecutables.

Usando Subversion

En esta sección indicamos los pasos para utilizar subversión bajo el protocolo SSH. Se asume que ha configurado sub conexión SSH con el servidor de subversion para que admita autentificación automática. Véase El capítulo sobre la SSH en los apuntes de Programación en Paralelo II (http://nereida.deioc.ul para aprender a establecer autentificación automática vía SSH.

Use Subversion: Creación de un Repositorio Parece que en `banot` esta instalado subversion. Para crear un repositorio emita el comando `svnadmin create`:

```
-bash-3.1$ uname -a
Linux banot.etsii.ull.es 2.6.24.2 #3 SMP Fri Feb 15 10:39:28 WET 2008 i686 i686 i386 GNU/Linux
-bash-3.1$ svnadmin create /home/loginname/repository/
-bash-3.1$ ls -l repository/
total 28
drwxr-xr-x 2 loginname apache 4096 feb 28 11:58 conf
drwxr-xr-x 2 loginname apache 4096 feb 28 11:58 dav
drwxr-sr-x 5 loginname apache 4096 feb 28 12:09 db
-r--r--r-- 1 loginname apache    2 feb 28 11:58 format
drwxr-xr-x 2 loginname apache 4096 feb 28 11:58 hooks
drwxr-xr-x 2 loginname apache 4096 feb 28 11:58 locks
-rw-r--r-- 1 loginname apache  229 feb 28 11:58 README.txt
```

Una alternativa a considerar es ubicar el repositorio en un dispositivo de almacenamiento portable (pendriver)

Añadiendo Proyectos

Ahora esta en condiciones de añadir proyectos al repositorio creado usando `svn import`:

```
[loginname@tonga]~/src/perl/> uname -a
Linux tonga 2.6.24.2 #1 SMP Thu Feb 14 15:37:31 WET 2008 i686 i686 i386 GNU/Linux
[loginname@tonga]~/src/perl/> pwd
/home/loginname/src/perl
[loginname@tonga]~/src/perl/> ls -ld /home/loginname/src/perl/Grammar-0.02
drwxr-xr-x 5 loginname Profesor 4096 feb 28  2008 /home/loginname/src/perl/Grammar-0.02
[loginname@tonga]~/src/perl/> svn import -m 'Grammar Extended Module' \
                                      Grammar-0.02/ \
                                      svn+ssh://banot/home/loginname/repository/Grammar
Añadiendo      Grammar-0.02/t
Añadiendo      Grammar-0.02/t/Grammar.t
Añadiendo      Grammar-0.02/lib
Añadiendo      Grammar-0.02/lib/Grammar.pm
Añadiendo      Grammar-0.02/MANIFEST
Añadiendo      Grammar-0.02/META.yml
Añadiendo      Grammar-0.02/Makefile.PL
Añadiendo      Grammar-0.02/scripts
```

```
Añadiendo      Grammar-0.02/scripts/grammar.pl
Añadiendo      Grammar-0.02/scripts/Precedencia.yp
Añadiendo      Grammar-0.02/scripts/Calc.yp
Añadiendo      Grammar-0.02/scripts/aSb.yp
Añadiendo      Grammar-0.02/scripts/g1.yp
Añadiendo      Grammar-0.02/Changes
Añadiendo      Grammar-0.02/README

Commit de la revisión 2.
```

En general, los pasos para crear un nuevo proyecto son:

```
* mkdir /tmp/nombreProyecto
* mkdir /tmp/nombreProyecto/branches
* mkdir /tmp/nombreProyecto/tags
* mkdir /tmp/nombreProyecto/trunk
* svn mkdir file:///var/svn/nombreRepositorio/nombreProyecto -m 'Crear el proyecto nombreProye
* svn import /tmp/nombreProyecto \
            file:///var/svn/nombreRepositorio/nombreProyecto \
            -m "Primera versión del proyecto nombreProyecto"
```

Obtener una Copia de Trabajo

La copia en `Grammar-0.02` ha sido usada para la creación del proyecto, pero no pertenece aún al proyecto. Es necesario descargar la copia del proyecto que existe en el repositorio. Para ello usamos `svn checkout`:

```
[loginname@tonga]~/src/perl/> rm -fR Grammar-0.02
[loginname@tonga]~/src/perl/> svn checkout svn+ssh://banot/home/loginname/repository/Grammar G
A    Grammar/t
A    Grammar/t/Grammar.t
A    Grammar/MANIFEST
A    Grammar/META.yml
A    Grammar/lib
A    Grammar/lib/Grammar.pm
A    Grammar/Makefile.PL
A    Grammar/scripts
A    Grammar/scripts/grammar.pl
A    Grammar/scripts/Calc.yp
A    Grammar/scripts/Precedencia.yp
A    Grammar/scripts/aSb.yp
A    Grammar/scripts/g1.yp
A    Grammar/Changes
A    Grammar/README
Revisión obtenida: 2
```

Ahora disponemos de una copia de trabajo del proyecto en nuestra máquina local:

```
[loginname@tonga]~/src/perl/> tree Grammar
Grammar
|-- Changes
|-- MANIFEST
|-- META.yml
|-- Makefile.PL
|-- README
|-- lib
```

```
|   `-- Grammar.pm
|-- scripts
|   |-- Calc.yp
|   |-- Precedencia.yp
|   |-- aSb.yp
|   |-- g1.yp
|   `-- grammar.pl
`-- t
    `-- Grammar.t

3 directories, 12 files
[loginname@tonga]~/src/perl/>
[loginname@tonga]~/src/perl/> cd Grammar
[loginname@tonga]~/src/perl/Grammar/> ls -la
total 44
drwxr-xr-x 6 loginname Profesor 4096 feb 28  2008 .
drwxr-xr-x 5 loginname Profesor 4096 feb 28  2008 ..
-rw-r--r-- 1 loginname Profesor  150 feb 28  2008 Changes
drwxr-xr-x 3 loginname Profesor 4096 feb 28  2008 lib
-rw-r--r-- 1 loginname Profesor  614 feb 28  2008 Makefile.PL
-rw-r--r-- 1 loginname Profesor  229 feb 28  2008 MANIFEST
-rw-r--r-- 1 loginname Profesor  335 feb 28  2008 META.yml
-rw-r--r-- 1 loginname Profesor 1196 feb 28  2008 README
drwxr-xr-x 3 loginname Profesor 4096 feb 28  2008 scripts
drwxr-xr-x 6 loginname Profesor 4096 feb 28  2008 .svn
drwxr-xr-x 3 loginname Profesor 4096 feb 28  2008 t
```

Observe la presencia de los subdirectorios de control `.svn`.

Actualización del Proyecto

Ahora podemos modificar el proyecto y hacer públicos los cambios mediante `svn commit`:

```
loginname@tonga]~/src/perl/Grammar/> svn rm META.yml
D         META.yml
[loginname@tonga]~/src/perl/Grammar/> ls -la
total 40
drwxr-xr-x 6 loginname Profesor 4096 feb 28  2008 .
drwxr-xr-x 5 loginname Profesor 4096 feb 28 12:34 ..
-rw-r--r-- 1 loginname Profesor  150 feb 28 12:34 Changes
drwxr-xr-x 3 loginname Profesor 4096 feb 28 12:34 lib
-rw-r--r-- 1 loginname Profesor  614 feb 28 12:34 Makefile.PL
-rw-r--r-- 1 loginname Profesor  229 feb 28 12:34 MANIFEST
-rw-r--r-- 1 loginname Profesor 1196 feb 28 12:34 README
drwxr-xr-x 3 loginname Profesor 4096 feb 28 12:34 scripts
drwxr-xr-x 6 loginname Profesor 4096 feb 28  2008 .svn
drwxr-xr-x 3 loginname Profesor 4096 feb 28 12:34 t
[loginname@tonga]~/src/perl/Grammar/> echo "Modifico README" >> README
[loginname@tonga]~/src/perl/Grammar/> svn commit -m 'Just testing ...'
Eliminando     META.yml
Enviando       README
Transmitiendo contenido de archivos .
Commit de la revisión 3.
```

Observe que ya no es necesario especificar el lugar en el que se encuentra el repositorio: esa información esta guardada en los subdirectorios de administración de subversion `.svn`

El servicio de subversion parece funcionar desde fuera de la red del centro. Véase la conexión desde una maquina exterior:

```
pp2@nereida:/tmp$  svn checkout svn+ssh://loginname@banot.etsii.ull.es/home/loginname/reposito
loginname@banot.etsii.ull.es's password:
loginname@banot.etsii.ull.es's password:
A    Grammar/t
A    Grammar/t/Grammar.t
A    Grammar/MANIFEST
A    Grammar/lib
A    Grammar/lib/Grammar.pm
A    Grammar/Makefile.PL
A    Grammar/scripts
A    Grammar/scripts/grammar.pl
A    Grammar/scripts/Calc.yp
A    Grammar/scripts/Precedencia.yp
A    Grammar/scripts/aSb.yp
A    Grammar/scripts/g1.yp
A    Grammar/Changes
A    Grammar/README
Revisión obtenida: 3
```

Comandos Básicos

- Añadir y eliminar directorios o ficheros individuales al proyecto

  ```
  svn add directorio_o_fichero
  svn remove directorio_o_fichero
  ```

- Guardar los cambios

  ```
  svn commit -m "Nueva version"
  ```

- Actualizar el proyecto

  ```
  svn update
  ```

- Ver el estado de los ficheros

  ```
  svn status -vu
  ```

- Crear un tag

  ```
  svn copy svn+ssh://banot/home/logname/repository/PL-Tutu/trunk \
           svn+ssh://banot/home/casiano/repository/PL-Tutu/tags/practica-entregada \
   -m 'Distribución como fué entregada en la UDV'
  ```

Referencias

Consulte http://svnbook.red-bean.com/ . Vea la página de la ETSII http://www.etsii.ull.es/svn . En KDE puede instalar el cliente gráfico KDEsvn.

Autentificación Automática

Para evitar la solicitud de claves cada vez que se comunica con el repositorio establezca autentificación SSH automática. Para ver como hacerlo puede consultar las instrucciones en:

http://search.cpan.org/ casiano/GRID-Machine/lib/GRID/Machine.pod#INSTALLATION

Consulte también las páginas del manual Unix de `ssh`, `ssh-key-gen`, `ssh_config`, `scp`, `ssh-agent`, `ssh-add`, `sshd`

Código de 01MartelloAndTothBook.t

```
lhp@nereida:/tmp/Algorithm-Knap01DP-0.01/t$ cat -n 01MartelloAndTothBook.t
 1  # Before ‘make install’ is performed this script should be runnable with
 2  # ‘make test’. After ‘make install’ it should work as ‘perl Algorithm-Knap01DP.t’
 3
 4  ########################
 5  use strict;
 6  use Test::More  tests => 11;
 7
 8  BEGIN { use_ok('Algorithm::Knap01DP', qw/Knap01DP ReadKnap/); }
 9
10  ### main
11  my @inputfiles = qw/knap21.dat  knap22.dat  knap23.dat  knap25.dat/;
12  my @sol = (280, 107, 150, 900);
13  my $knap21 = ['102', [ '2', '20', '20', '30', '40', '30', '60', '10' ],
14                    [ '15', '100', '90', '60', '40', '15', '10', '1' ]];
15  my $knap22 = ['50',  [ '31', '10', '20', '19', '4', '3', '6' ],
16                    [ '70', '20', '39', '37', '7', '5', '10' ]];
17  my $knap23 = ['190', [ '56', '59', '80', '64', '75', '17' ],
18                    [ '50', '50', '64', '46', '50', '5' ]];
19  my $knap25 = ['104', [ '25', '35', '45', '5', '25', '3', '2', '2' ],
20                    [ '350', '400', '450', '20', '70', '8', '5', '5' ]];
21
22  my $knapsackproblem = [$knap21, $knap22, $knap23, $knap25];
23
24  my $i = 0;
25  my ($M, $w, $p);
26  my @f;
27
28  # Now 2*@inputfiles = 8 tests
29  for my $file (@inputfiles) {
30    ($M, $w, $p) = ReadKnap((-e "t/$file")?"t/$file":$file);
31    is_deeply($knapsackproblem->[$i], [$M, $w, $p], "ReadKnap $file");
32    my $N = @$w;
33    @f = Knap01DP($M, $w, $p);
34    is($sol[$i++], $f[$N-1][$M], "Knap01DP $file");
35  }
36
37  # test to check when weights and profits do not have the same size
38  $M = 100; @$w = 1..5; @$p = 1..10;
39  eval { Knap01DP($M, $w, $p) };
40  like $@, qr/Profits and Weights don't have the same size/;
41
42  TODO: { # I plan to provide a function to find the vector solution ...
```

```
43    local $TODO = "Randomly generated problem";
44    can_ok('Algorithm::Knap01DP', 'GenKnap');
45  }
```

Bibliografía