github github

Search…

Search

- Explore
- Gist
- Blog
- Help

pcordone

- 3
- 
-

- Watch Unwatch
- Fork
- Pull Request
- 
    - 3
    - 2

# spob / Firehoze

- Code
- Network
- Pull Requests 0
- Issues 0
- Wiki 21
- Stats & Graphs

- Home
- Pages
- Wiki History
- Git Access

# Dev Environment OSX

- New Page
- Edit Page
- Page History

This page captures a description of how I setup Ruby and Rails on OSX Lion 10.7.3 using XCode . Keep in mind I was brand new to Ruby and Rails when I created this page. I had a lot of issues getting Ruby working with rvm and mysql. It took me about a week of screwing around. Part of the problem is that when

Apple released XCode 4.2 they changed the compiler from gcc-4.2. These instructions should get you going; there may be variations that work, but I haven't tried them.

## XCode Installation

XCode is a suite of tools developed by Apple for creating application for OS X and iOS. It includes GCC the Gnu Compiler Collection.

1. XCode starting with Lion must be downloaded from the Mac App Store
2. Follow instructions on this link to install gcc-4.2 compiler. You can also install XCode 4.1 (you can have more than one copy of XCode installed) which still uses the gcc-4.2 compiler. There is a way to select which XCode compiler is active at the command line. I believe it is xcode_select. I chose the link below: http://rocksolidwebdesign.com/notes-and-fixes/ruby-xcode/

## Homebrew Installation

Homebrew is a package manager for OS X. For more information here is the website Homebrew Site

1. Install brew by executing the following command in Terminal`/usr/bin/ruby -e "$(curl -fsSL https://raw.github.com/gist/323731)"`

## mySql Installation

mySql is an open source database application. For more information go to mySql.com

1. Install mysql via brew. I tried installing mysql from the binary builds (.dmg) at the mysql site and was not able to get the mysql gem to build. I literally lost 5 days screwing with this. I tried all variations for the ruby environment to find the libraries. I believe when you install mysql from the binary images, the dev libraries aren't include though it does dump files in include and lib folders, or what's more likely is that because the libraries aren't built on your system it causes issues when they try to load. I think all of this has to do with Apple changing compilers from gcc to LLVM. You could also try compiling mysql on your system and not installing via brew. I suspect that will work, though I haven't tried it, and brew does make it pretty convient to install and compile from the source. `brew install -v mysql`

2. Create the mysql database. `mysql_install_db --verbose --user=`whoami`--basedir="$(brew --prefix mysql)" --datadir=/usr/local/var/mysql --tmpdir=/tmp`

3. You can start and stop the server with `mysql.server start` `mysql.server stop`

## RVM Installation

RVM is an application that lets you manage multiple versions of Ruby and Rails on a single system. More information at RVM Homepage. You can install RVM at the system level or user level.

1. I did the installation at the user level. For system level you type sudo before the command. You can not install both or you will run the rvm.sh script in /etc/profile and then in your .bash_rc. Running the script twice screws it up. I lost days figuring this out. You can unistall rvm with rvm implode. RVM will use the gcc-4.2 compiler if it finds it, otherwise it will default to the LLVM that ships with

XCode 4.2. You should see RVM compiling with the gcc-4.2.

## Ruby Installation

1. Install ruby version 1.9.3-p0 through rvm. RVM will compile ruby with the gcc-4.2 compiler. `rvm install ruby-1.8.7-p358`
2. Next you will create gem sets to keep your different ruby instances separate. `rvm gemset create rails235`
3. Use the new gem set with a version of ruby `rvm use 1.8.7-p358@rails235`

If you want rvm to automatically use a gem set for a project then run this to create a .rvmrc file in the project. When you cd into the directory, the rvm command will run automatically. `rvm --rvmrc --create use 1.8.7-p357@rails235` I also put the rvm use command in my .bash_profile so it would select the gemset on a new terminal.

1. Validate that the gem set is current. You will see => indicating the default. `rvm gemset list`

## Setup Ruby and mySql

1. Install the mysql gem `gem install mysql`
2. Validate the connection between ruby and mysql by putting this script in a .rb file and running it. You should see your version of mysql. Mine was "Server version: 5.5.19"

Here is a test

```
require "rubygems"
require "mysql"
begin
# connect to the MySQL server
dbh = Mysql.real_connect("localhost", "testuser", "testpass", "test")
# get server version string and display it
puts "Server version: " + dbh.get_server_info
rescue Mysql::Error => e
puts "Error code: #{e.errno}"
puts "Error message: #{e.error}"
puts "Error SQLSTATE: #{e.sqlstate}" if e.respond_to?("sqlstate")
ensure
# disconnect from server
dbh.close if dbh
end
```

## Rails Installation

1. Install the rails version `gem install rails -v 2.3.5`

## Sphinx Installation

1. Install sphinx via brew `brew install sphinx` Create a data directory under /var (i.e. /var/data needs to exist). Follow the instructions under doc/sphinx.html to validate your installation.

If you go for installing sphinx from source you should configure with: `./configure --with-mysql=<path to mysql>`

1. Check what version of gem that you have by running `gem -v` If you don't have 1.6.2 then run gem update --system 1.6.2. I believe it downgrades gem to a previous version. I tried running rake gems:install with a newer version and got a bunch of deprecated warnings and rake failed. After downgrading to 1.6.2, you can run `rake gems:install` to install all dependent gems. I'm not sure if there needed to be manual installation of the other gems. You could try downgrading to 1.6.2 with the gem update command first then `run rake gems:install` and see if it resolves all dependencies without having to install other gems like shoulda, etc.

2. Install shoulda gem (if rake gems:install doesn't). `gem install shoulda`

3. Install thinking sphinx gem. (if rake gems:install doesn't) `gem install thinking-sphinx -v 1.4.4`

4. Install the gem right_aws. `gem install right_aws` (if `rake gems:install` doesn't)

5. Install the twitter gem. I don't know why this needs to be installed manually since it is in the environment.rb. When I ran rake db:migrate (in the step below) it complained that the twitter gem wasn't installed or it installed a later one. I uninstalled it and ran: `gem install twitter -v 0.7.0`

6. Make sure you have version 1.2.2. of the riddle gem installed or you won't be able to run the spinx configuration task with rake. I had to uninstall riddle and reinstall v 1.2.2. `gem install riddle -v 1.2.2`

7. Create the sphinx configuration file with the command below. Mine dumped it to Firehoze/config/development.sphinx.conf.
   `rake thinking_sphinx:configure`

## Database Setup

1. Next create the database by running `rake db:create` Note: I had issues logging in. I ended up creating the database with a gui tool making sure to pick UTF-8 and then after completing seeding the database didn't have issues with logging in. I'm not sure if they are related.

2. Next create the tables by running `rake db:migrate`

3. Create the test database manually through mysql client (or your favorite tool) firehoze_test.

4. Run: `rake db:test:prepare`

5. Run the unit tests with `rake test` It may complain that you don't have the factory_girl version 1.2.4 gem installed. I had to install it with: `gem install factory_girl -v 1.2.4` `gem install webrat`

6. seed the database with: `rake db:seed`

## Start the Server

1. Start the server with `script/server`

2. Start the task_manager (background daemon) by typing: `ruby script/task_server_control.rb start` to run in the background or `ruby script/task_server_control.rb run` to run it in the foreground.

3. The app is seeded with a first user, login (email) is admin@firehoze.com, password is changeme. Surf to http://localhost:3000 and login.

## Debugging in IDE

1. I was able to configure RubyMine (JetBrains) very quickly. I chose to get the project from github, then switched the branch to northeastern (or whatever branch you are working on) from the right click menu, then went into preferences and set the sdk to be my rvm 1.8.7-p357@rails235. Then I debugged the application. It took me about 15 minutes from download to debugging. If anyone figures out eclipse let me know. RubyMine is $69 for the personal, but I thought it was worth the time saved.

2. If you want to be able to debug javascript, go into your development debug configuration (Run -> Edit Condigurations). Select Development: Firehoze. Click on "Run browser" and then click the checkbox Start JavaScript debugger automatically debugging.

3. Run the Development debug configuration after starting mysql. RubyMine will launch a browser and now you can set breakpoints in the ruby server code, erb files and javascript files.

4. I installed the following two gems as part of trying to get Aptana Studio 3 running. I don't know if RubyMine needs them, but I'm guessing if it does, it will let you know when you try to debug. `gem install ruby-debug-base` `gem install ruby-debug-ide`

## Amazon S3 and Streaming Video Setup

1. Create three buckets in Amazon S3. One for the input to zencoder, one for the output video and one for the output thumbnail.

2. Setup video streaming by:
   Log into Amazon S3 Console and clicking on Cloud Front.
   Click on Create Distribution.
   Pick `Streaming` and select the output bucket (i.e. output.prizetube.com). Click Continue.
   Click Continue
   Click Create Distribution Make sure to change the the value s3l34fgg5q8sq below in the file lessons/_player.html.erb to match the identifier of your video stream you just setup. This value really needs to be moved into the s3.yml file. `netConnectionUrl: 'rtmp://s3l34fgg5q8sq.cloudfront.net/cfx/st'`

3. Make sure the following value is in your constants.rb
   `FLIX_CLOUD_AWS_ID = '6c8583d84664a381db0c6af0e79b285ede571885fbe768e7ea50e5d3760597dd'`

You can get the value form the zencoder documentation that talks about S3 permissions. It's the canonical ID that zencoder will use to access the input video. Firehoze gives this ID read permissions to the source video so that zencoder can download it.

4. Make sure the following values in the S3.yml match your input, output and thumbnails S3 bucket names. Below shoes the entries for prizetube.
```
aws_s3_input_video_bucket: input.prizetube.com
aws_s3_output_video_bucket: output.prizetube.com
aws_s3_thumbs_bucket: thumbs.prizetube.com
```

5. Also make sure the following match your amazon user id. You can get at these from S3 console -> Security Credentials -> Access Key
```
access_key_id: <YOUR KEY VALUE>
secret_access_key: <YOUR SECRET KEY VALUE>
```

## Encoding Videos From Your Local Environment

1. Make sure that Amazon S3 is setup correctly for your environment and that you have the correct values in your config files. See the section above.

2. If you want to encode video's from your local environment, you will have to run two processes from a shell terminal. The first is part of Firehoze and is called task_server. It is a batch job that processes jobs that are scheduled to run. I created an alias called tsstart in my .bashrc file to launch it.
```
ruby script/task_server_control.rb run -f -- -e development
```

3. The second process is called zencoder_fetcher (http://rubygems.org/gems/zencoder-fetcher). You can install the gem with:
```
gem install zencoder-fetcher
```

You will run zencoder_fetcher with the following where the last argument is the FLIX_API_KEY value from constants.rb. You get this by logging into your zendcoder account and selecting the top menu item called API. You will see the Full Access Key value. That should be set in your constants.rb and used as the last argument.
```
zencoder_fetcher --url http://localhost:3000/lessons/conversion_notify --loop --
interval 10 --count 1 <FLIX_API_KEY>
```

or alternately you can choose not to run zencoder_fetcher and run `rake flix:repair`. It looks for video files in the Amazon S3 bucket and simulates the notification call to update the video status in the database.

## Deploying to Server

1. To deploy new code to a server you will check in your changes to github and then run this command on your development laptop (not the staging server).

## Software Versions

Below are versions from the Northeastern staging site that I sat and setup with Bob on 2/19/2012

| Software | Version |
|---|---|

| | |
|---|---|
| ruby | 1.8.7 (2010-01-10 patchlevel 249) [i486-linux] |
| rake | 0.9.2.2 |
| gems | 1.6.2 |
| rvm | no version |
| actionmailer | 2.3.5 |
| actionpack | 2.3.5 |
| active_url | 3.2.1 |
| activemodel | 3.2.1 |
| activerecord | 2.3.14, 2.3.5 |
| activeresource | 2.3.5 |
| activesupport | 3.2.1, 2.3.14, 2.3.5 |
| addressable | 2.2.6 |
| after_commit | 1.0.10 |
| arel | 3.0.0 |
| astrails-safe | 0.2.7 |
| authlogic | 2.1.6 |
| aws-s3 | 0.6.2 |
| builder | 3.0.0 |
| cloudfiles | 1.5.0.1 |
| commonjs | 0.2.0 |
| crack | 0.3.1 |
| crafterm-comma | 0.2.2 |
| daemon_controller | 0.2.6 |
| daemons | 1.1.6 |
| faraday | 0.7.6 |
| fastercsv | 1.5.4 |
| fastthread | 1.0.7 |
| gravtastic | 3.2.6, 2.2.0 |
| hoe | 2.13.0 |
| hpricot | 0.8.6 |
| httparty | 0.4.3 |
| i18n | 0.6.0 |
| jrails | 0.6.0 |
| json | 1.6.5 |
| jviney-acts_as_taggable_on_steroids | 1.1 |
| less | 2.0.9 |
| libv8 | 3.3.10.4 x86-linux |
| mash | 0.0.3 |
| mime-types | 1.17.2 |
| multi_json | 1.0.4 |
| multipart-post | 1.1.4 |
| mysql | 2.8.1 |

| | |
|---|---|
| net-sftp | 2.0.5 |
| net-ssh | 2.3.0 |
| newrelic_rpm | 2.12.3 |
| oauth | 0.4.5 |
| paperclip | 2.3.11 |
| passenger | 3.0.11 |
| rack | 1.4.1, 1.0.1 |
| rails | 2.3.5 |
| rake | 0.9.2.2 |
| riddle | 1.2.2 |
| right_aws | 3.0.0 |
| right_http_connection | 1.3.0 |
| rubygems-update | 1.8.15, 1.6.2 |
| searchlogic | 2.3.16 |
| shoulda | 2.11.3 |
| simple_oauth | 0.1.5 |
| spob_browser_detector | 1.1.1 |
| therubyracer | 0.9.9 |
| thinking-sphinx | 1.4.4 |
| thoughtbot-shoulda | 2.11.1 |
| twitter | 0.7.0 |
| tzinfo | 0.3.31 |
| xml-simple | 1.1.1 |
| zencoder | 2.1.15 |

Last edited by pcordone, 4 days ago

Delete this Page

## GitHub Links

### GitHub

- About
- Blog
- Features
- Contact & Support
- Training
- GitHub Enterprise
- Site Status

### Tools

- Gauges: Analyze web traffic
- Speaker Deck: Presentations

- Gist: Code snippets
- GitHub for Mac
- Issues for iPhone
- Job Board

**Extras**

- GitHub Shop
- The Octodex

**Documentation**

- GitHub Help
- Developer API
- GitHub Flavored Markdown
- GitHub Pages

- Terms of Service
- Privacy
- Security

# Markdown Cheat Sheet

## Format Text

Headers

```
# This is an <h1> tag
## This is an <h2> tag
###### This is an <h6> tag
```

Text styles

```
*This text will be italic*
_This will also be italic_
**This text will be bold**
__This will also be bold__

*You **can** combine them*
```

## Lists

Unordered

```
* Item 1
* Item 2
  * Item 2a
  * Item 2b
```

Ordered

```
1. Item 1
2. Item 2
3. Item 3
   * Item 3a
   * Item 3b
```

## Miscellaneous

Images

```
![GitHub Logo](/images/logo.png)
Format: ![Alt Text](url)
```

Links

```
http://github.com - automatic!
[GitHub](http://github.com)
```

Blockquotes

```
As Kanye West said:

> We're living the future so
> the present is our past.
```

## Code Examples in Markdown

Syntax highlighting with [GFM]

~~~
```javascript
function fancyAlert(arg) {
  if(arg) {
    $.facebox({div:'#foo'})
  }
}
```
~~~

Or, indent your code 4 spaces

```
Here is a Python code example
without syntax highlighting:

    def foo:
      if not bar:
        return true
```

Inline code for comments

```
I think you should use an
`<addr>` element here instead.
```

Something went wrong with that request. Please try again. Dismiss

## Looking for the GitHub logo?

- **GitHub Logo**

  Download

- **The Octocat**

  Download