# Installing Rails 2.3 Under CentOS/RHEL 5 and Apache 2.2

Scott Taylor
Tailor Made Software
July 24, 2011
Version 1.2

## 1.0 Introduction

Ruby On Rails (aka just "Rails") is a modern scripting system that allows easy development of "Web 2.0" type applications. It relies heavily on Dynamic HTML (dHTML) and AJAX, a form of JavaScript that provides asynchronous communication between server and client.

The following instructions will install Rails 2.3.x under Apache 2.2 on CentOS or Red Hat Enterprise Linux 5. They should work for any version of CentOS/RHEL 5 but were tested using CentOS 5.2 and 5.5. CentOS is the "community supported" version of RHEL and contains essentially the same packages as RHEL but without the commercial support.

The normal operation of installing linux packages is with yum, downloading the packages automatically over the internet. However, some companies have restrictive firewalls that preclude this method. As a result we will describe loading using pre-downloaded packages. The basic effect of this is that only a given version will be available and therefore will be loaded instead of the version that is current. However, since we want to work with specific versions that have been tested, we would be installing specific versions in any case.

Instructions are given for two methods of file serving/load balancing – using Passenger and Mongrel. One of these needs to be installed. Each has

## 2. Prerequisites

There are several packages that are required to run and install Rails, including the Ruby Interpreter (the system that processes the Ruby language files), RubyGems (the installation system for Ruby modules) and Apache 2.2 (the web server).

### 2.1 Non-requisites

It is possible that certain software that is not desired may be already installed by default on your system, even if it is a clean system. For instance, CentOS 5.6 installs Rails 3.0 by default. Not only do we not want Rails 3.X (we are dependent on Rails 2.3) but

we may also want to replace the version of Ruby that is installed by default. At the appropriate time we will instruct you to check for these "non-requisite" packages.

## 2.2 Apache

First we will install Apache 2 or verify that Apache 2 is installed correctly. You will need the Development headers, not just the normal installation.

The standard commands for installing Apache on CentOS/RHEL are:

```
yum install httpd
yum install httpd-devel
yum install apr-devel
yum install apr-util-devel
```

However, just installing httpd-devel will update or install the other three if Apache was not previously loaded on the machine.

## 2.2 Ruby

There are two options for installing Ruby: "standard Ruby" and "Enterprise Ruby". Standard Ruby is the normal Ruby interpreter that may be installed with your CentOS installation. Enterprise Ruby is a free version that is developed by Phusion, a Dutch software company. It is faster and manages memory better than standard Ruby. We recommend its use.

### 2.2.1 Enterprise Ruby

The current source code for Enterprise Ruby is contained in ruby-enterprise-1.8.7-2011.03.tar.gz

After copying the source to the desired location extract the code using:

```
tar xzvf ruby-enterprise-X.X.X.tar.gz
```

then install the code using:

```
./ruby-enterprise-X.X.X/installer
```

So for the current code those commands would be:

```
tar xzvf ruby-enterprise-1.8.7-2011.03.tar.gz
./ruby-enterprise-1.8.7-2011.03/installer
```

The installer will ask for the location to place Enterprise Ruby. We used:

```
/opt/ruby-enterprise-1.8.7/
```

### 2.2.2 Replace Ruby

If Ruby is already installed on the system you can either place the Enterprise Ruby path on the front of the system path, or replace the symlink in the /usr/bin or usr/sbin directory with a symlink to the Enterprise Ruby directory

```
export PATH=/opt/ruby-enterprise-1.8.7/bin:$PATH

sn -l
```

## 2.3 RubyGems

RubyGems is an installation system, sort of like yum or apt, that is used by Ruby modules.

There are two versions of RubyGems that you can use: "normal" and Enterprise Ruby. It does not matter which you use, but, of course, if you do not have "standard Ruby" installed then you would use the version that comes with Enterprise Ruby.

### 2.3.1 Normal RubyGems

If you are not using Enterprise Ruby, and do not have RubyGems already installed, you will need to install it.

You can check whether it is already installed using whereis:

```
whereis gem
```

Installation of RubyGems is straightforward:

1) Copy the compressed tarball to the desired location
2) Extract the source
3) Switch to the rubygems directory
4) Run the ruby command setup.rb

```
tar -xzvf rubygems-1.5.0.tgz
cd rubygems-1.5.0
/opt/ruby-enterprise-1.8.7/bin/ruby setup.rb
```

This should result in "RubyGems 1.5.0 installed"

### 2.3.2 Enterprise Ruby

Enterprise Ruby installs its own version of RubyGems. If you have "normal" ruby installed on the system and did not switch the PATH environment variable or create a Symlink to Enterprise Ruby, you can create an alias to the Enterprise Ruby version.

```
alias reegem=/opt/ruby-enterprise-1.8.7/bin/gem
```

## 2.4 Uninstall Rails 3.0

A base installation of CentOS may install an incorrect version of Rails. We are using Rails 2.3.8. Version 2.3.11 will also work. Version 2.3.6 will work but contains a security bug that is fixed in 2.3.8, so 2.3.8 should be used instead. CentOS 5.2 will install Rails 3.0.9 which is totally incompatible and therefore must be removed or worked around. Removal is the easiest.

You will use gem to uninstall the existing version of Rails and dependant packages. The command is:

```
gem uninstall abc -v=x.y.z
```

where "abc" is the name of the package to be removed and "x.y.z" is the version. If there is more than one version of the package installed and you do not list the version you want uninstalled, it will prompt you for which version to remove, and give you a choice to remove all versions.

The command "gem list" will give you a list of all gems installed. A sample from base CentOS 5.2 after rails 2.3.8 has been installed:

```
actionmailer (3.0.9, 2.3.8)
actionpack (3.0.9, 2.3.8)
activemodel (3.0.9)
activerecord (3.0.9, 2.3.8)
activeresource (3.0.9, 2.3.8)
activesupport (3.0.9, 2.3.8)
arel (2.0.10)
builder (2.1.2)
bundler (1.0.15)
daemon_controller (0.2.6)
erubis (2.6.6)
fastthread (1.0.7)
formtastic (1.2.4)
i18n (0.6.0, 0.5.0)
mail (2.2.19)
mime-types (1.16)
mysql (2.8.1)
passenger (3.0.7)
polyglot (0.3.1)
rack (1.3.0, 1.2.3, 1.1.2)
rack-mount (0.6.14)
rack-test (0.5.7)
rails (3.0.9, 2.3.8)
railties (3.0.9)
rake (0.9.2)
```

Each item lists the name followed by the version number(s). For instance, activerecord has versions 3.0.9 and 2.3.8 installed.

You need to remove the following if present:

```
actionmailer 3.0.9
actionpack 3.0.9
activemodel 3.0.9
activerecord 3.0.9
activeresource 3.0.9
activesupport 3.0.9
builder 2.1.2
```

```
bundler 1.0.15
passenger 3.0.7
rack 1.3.0 and 1.2.3
rack-mount 0.6.14
rack-test 0.5.7
rails 3.0.9
railties 3.0.9
```

# 3. Install Rails

## 3.1 Installation

Installation of Rails is straightforward. The easiest method is to copy the gem files to a directory and run RubyGems from that directory. You will need to install the following gem files:

- activerecord-2.3.8.gem
- activeresource-2.3.8.gem
- actionmailer-2.3.8.gem
- activesupport-2.3.8.gem
- actionpack-2.3.8.gem
- rack-1.1.0.gem
- rake-0.8.7.gem

These are installed in two steps. First run:

```
gem install rack-1.1.0.gem
gem install rake-0.8.7.gem
```

then run:

```
gem install rails -v=2.3.8
```

The rails install will install all of the "active" and "action" gems.

Do NOT use Rack V1.3.0. Passenger will not recognize it.

You should then install the other gems needs by Visual Query:

```
gem install i18n-0.6.0.gem
gem install formtastic-1.2.4.gem
```

*Note*: *when you install formtastic on a Rails 2.3 system it will complain that activesupport 3.0 or higher is required. That is a problem with the gem installer and is not really true. Activesupport is part of Rails 3 and is not required for Rails 2. If you run "gem list" you will see that formtastic 1.2.4 was actually installed despite the "error" notice.*

You will need to install the database connector unless you are using SqlLite 3. For MySQL install use:

```
gem install mysql
```

If you are using Oracle you will need to install the Oracle Enhanced Driver:

```
gem install activerecord-oracle_enhanced-adapter-1.3.2.gem
```

## 3.2 Testing Rails

With Rails and Enterprise Ruby installed it is a good idea to test the installation so far. That is easiest by creating a simple rails application and running it. The instructions below use mysql for the database and call the project "rtest". You should modify them as needed.

Create a directory for the rails project and then run:

```
rails -d mysql rtest
```

Create a database in the Mysql database:

```
mysql -u root
mysql> create database rtest;
```

Modify the config/database.yml file to point to the new database:

```
development:
          adapter: mysql
          encoding: utf8
          reconnect: false
          database: rtest
          pool: 5
          username: root
          password:  jpmc
          host: backset
```

Create the database table using rails:

```
script/generate scaffold simple name:string
```

This will create the table "simple" in the "rtest" database.

If then you start the rails server using "ruby script/server" and go to the main page "localhost:3000" it will display the Rails equivalent of "Hello world!". Click on the "About your applications environment" link and it should display the information about your setup of rails. It should be something like this:

```
Ruby version   1.8.7 (x86_64-linux)
RubyGems version       1.5.0
Rack version   1.1
Rails version  2.3.8
Active Record version  2.3.8
Active Resource version       2.3.8
Action Mailer version  2.3.8
Active Support version 2.3.8
Application root       /www/rtest
Environment    development
Database adapter       mysql
Database schema version       0
```

# 4. Load Balancing Software

In order to run more than one instance of the Rails application we will need to install multiple server instances and load balancing software. There are three methods of accomplishing this: Passenger, Mongrel and Thin. We will describe the first two.

## 4.1Phusion Passenger

Phusion Passenger is a modification of an older system known as mod_rails. Passenger allows multiple instances of an application to be run and takes care of balancing the load between the instances so no one instance is being overtaxed or consumes all the resources.

Passenger works with Apache or ngenix to serve the pages for the website. Apache and ngenix serve the static pages while Passenger handles the dynamic pages from the Rails application. Unlike mongrel, Passenger is very easy to configure and works with Apache.

### 4.1.1 Installation

The easiest way to install Passenger is using the gem. From the directory where the gem is located run:

```
gem install passenger
```

This will install the version that is resident in that directory. If it does not then use:

```
gem install passenger -v=3.0.5
```

You then need to install the Apache specific code for Passenger. Run:

```
/opt/ruby-enterprise-1.8.7/bin/passenger-install-apache2-module
```

This should result in "The Apache 2 module was successfully installed."

Note: the path may differ depending on where you installed Enterprise Ruby.

### 4.1.2 Environment Variables

If your installation of linux uses a non-default setup you may need to define one or more environment variables or modify the path to enable Passenger to find the dependant software.

The following may need to be defined:

```
export APXS2=/apps/fast/tyger/fcBase/bin/fc-osgi-apache/linux-rh5-x64/bin/apxs
export APR_CONFIG=/apps/fast/tyger/fcBase/bin/fc-osgi-apache/linux-rh5-x64/bin/apr-1-config
export APU_CONFIG=/apps/fast/tyger/fcBase/bin/fc-osgi-apache/linux-rh5-x64/bin/apu-1-config
export APACHE=/apps/fast/tyger/fcBase/bin/fc-osgi-apache/linux-rh5-x64/bin/
export APACHE2=/apps/fast/tyger/fcBase/bin/fc-osgi-apache/linux-rh5-x64/bin/
```

```
export HTTPD=/apps/fast/tyger/fcBase/bin/fc-osgi-apache/linux-rh5-x64/bin/httpd
export PATH=/opt/ruby-enterprise-1.8.7-2011.03/bin:$PATH
```

### 4.1.3 Configuring Apache

The final step is to configure Apache to use Passenger.

Edit your Apache configuration file (the default location is `/etc/httpd/conf/httpd.conf`), and add these lines (or their equivalent since the path will be different):

```
   LoadModule passenger_module /opt/ruby-enterprise-1.8.7/lib/ruby/gems/1.8/gems/passenger-
3.0.5/ext/apache2/mod_passenger.so
   PassengerRoot /opt/ruby-enterprise-1.8.7/lib/ruby/gems/1.8/gems/passenger-3.0.5
   PassengerRuby /opt/ruby-enterprise-1.8.7/bin/ruby

# Virtual host Default Virtual Host

# Virtual host jpmcvq

<VirtualHost *:80>
        ServerName  www.jpmcvq.com
        DocumentRoot /www/rtest/public
         <Directory /www/rtest/public >
             RailsEnv development
            AllowOverride all
             Options -MultiViews
         </Directory>
        DirectoryIndex index.html index.htm index.shtml
</VirtualHost>
```

Make sure you set the RailsEnv variable to "production", "development" or "test" as appropriate. Also, note that the directory statements point to the "public" subdirectory of the rails application.

You will also need to change the "ServerName" setting higher in the httpd.conf file to a valid server name.

Now, restart Apache and try the "Hello, world!" page again.

## 4.2 Mongrel

Mongrel, and its associated process Mongrel Cluster, work in conjunction with Apache to provide load balancing and multiple process support. Mongrel is actually a web server written in Ruby that is optimized for running Ruby-based processes like Rails. Mongrel Cluster allows multiple instances of Mongrel to run in parallel. Apache is used to provide load balancing using *mod_proxy_balancer* and to serve static pages, while Mongrel is used to serve Ruby-based dynamic pages.

### 4.2.1 Installation

Mongrel and Mongrel Cluster are installed from their gems. Use the commands:

```
gem install mongrel -v=1.1.5
gem install mongrel_cluster -v=1.0.5
```

### 4.2.2 Configuration

Mongrel has no specific configuration requirements. Everything is configured using Mongrel Cluster.

To configure Mongrel Cluster you need to know the port to use (we will using 3000), the number of instances (we will start with 3) and which environment is being run (we will use development for now).

First change directory to the home directory for the application and then run mongreal cluster configure:

```
cd /www/public_html/testapp
mongrel_rails cluster::configure -e development -p 3000 -N 3 -c /www/public_html/testapp -a
192.168.1.104
```

Of course you should use the appropriate values for the www directory and the URL.

Note that this will create the file `config/mogrel_cluster.yml`. Make sure you have write permission to the config directory.

You can manually start, restart and stop the cluster (from the document root directory) using:

```
mongrel_rails cluster::start
mongrel_rails cluster::restart
mongrel_rails cluster::stop
```

Next create an init script and add it to start automatically:

```
mkdir /etc/mongrel_cluster
ln -s /home/demo/public_html/testapp/config/mongrel_cluster.yml /etc/mongrel_cluster/testapp.yml
cp /usr/lib/ruby/gems/1.8/gems/mongrel_cluster-1.0.5/resources/mongrel_cluster /etc/init.d/
chmod +x /etc/init.d/mongrel_cluster
chkconfig -add mongrel_cluster
```

You can check the cluster status using:

```
mongrel_cluster_ctl status
```

### 4.2.3 Apache Configuration

The final step is to configure apache to use mongrel.

```
<VirtualHost *:80>
    ServerAdmin webmaster@www.jpmcvq.com
    ServerName jpmcvq.com
    ServerAlias www.jpmcvq.com

    DocumentRoot /www/rtest/public

    RewriteEngine On

    <proxy balancer://mongrel1>
        BalancerMember http://192.168.1.104:3000
```

```
        BalancerMember http://192.168.1.104:3001
        BalancerMember http://192.168.1.104:3002
    </proxy>

    RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} !-f
    RewriteRule ^/(.*)$ balancer://mongrel1%{REQUEST_URI} [P,QSA,L]

    ProxyPass / balancer://mongrel1/
    ProxyPassReverse / balancer://mongrel1/
    ProxyPreserveHost on

    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>

    ErrorLog /www/rtest/log/error.log
    CustomLog /www/rtest/log/access.log combined

</VirtualHost>
```