How to Install Ruby On Rails, using Mongrel and Lighttpd, on Fedora Core 6 (FC6)

Supplemental FC6/Rails Articles

- How To Connect To SQL2005 From Ruby On Rails on Fedora Core 6 (Linux)

Introduction ¶

The goal of this article is to create a "Production-Quality" Rails Server. Thus, the best possible technologies (strictly my opinion) have been selected at the time of this writing to achieve this goal.

Unlike other developers, I prefer to install my Rails Applications under Web Subdirectories, such as http://www.not404.com/MyRailsApp, instead of running it as a Root Application of a Web Root, such as http://MyRailsApp.not404.com/. These instructions are geared for how I lay things out, but will let you know what to adjust in order to run your Rails Apps as traditional Web-Root Applications.

You may also notice that these instructions are SQLite3-oriented. This is intentional. IMHO, it's better to use the simplest-case database to prove that everything else is properly stitched together. Then, once you're satisfied that everything is properly locked down and performance-tuned, you can focus your attention on tying your Rails Application to a **real** database.

- Fedora Core 6
- Ruby on Rails
- Mongrel Ruby Application Server
- Lighttpd Web Server
- Various Plumbing and Configuration "Best Practices" to stitch it all together. :-)

Fedora Core 6: The Base Server ¶

- Install Fedora Core 6. Enable the "Development Tools" group in the initial setup screens.
- do a **yum -y update** to get the latest update patches
- Optionally, do a **visudo** to configure "sudo" access.

If you missed the "Development Tools" checkbox, do a **yum -y groupinstall "Development Tools"** to install it now. You'll need the GCC compiler to build some Ruby Gems, it makes sense to have it ready here. After everything's configured, you may decide to do a **yum -y groupremove "Development Tools"** to remove the C compiler and other tools from your production box.

Ruby On Rails ¶

- yum -y install ruby ruby-devel ruby-irb ruby-libs ruby-rdoc ruby-ri rubygems
- yum -y install ruby-sqlite3
- Now, we can use Gem to get the latest Rails Framework:
    - sudo gem install rails --include-dependencies

Optional (But Recommended) Yum Packages ¶

- ruby-sqlite3
- ruby-mysql
- ruby-postgres
- ruby-clearsilver
- ruby-racc
- subversion-ruby
- ruby-docs

Mongrel Ruby Application Server ¶

- sudo gem install gem_plugin daemons capistrano --include-dependencies
- sudo gem install mongrel mongrel_cluster railsmachine --include-dependencies
- sudo /usr/sbin/adduser -r mongrel (This creates a **mongrel** user, as suggested BryanThompson?'s Blog).

I prefer to install my Rails Applications under Web Subdirectories, such as http://www.not404.com/MyRailsApp, instead of running it as a Root Application of a Web Root, such as http://MyRailsApp.not404.com/.

To correctly handle Rails Applications running under Web Subdirectories , we need to use Mongrel's **--prefix** support, recently added in Mongrel Cluster 0.2.1. At this writing though, Mongrel Cluster 0.2.1 is still a pre-release version, so it needs to be installed from their "trunk" repository, instead of from the standard Gem Repositories.

To get the Pre-Release version of Mongrel Cluster to let us run Rails Applications under Web Subdirectories, run this command:

- sudo gem install mongrel_cluster --source http://railsmachine.rubyforge.org/releases/
- sudo gem cleanup

Test Mongrel and Rails As a Non-Root User ¶

- cd ~
- rails testapp
- cd testapp
- mongrel_rails start

Launch firefox, and go to http://localhost:3000 -- you should get the "Welcome Aboard" web page. You can now stop Mongrel, so we can configure it as a Service.

Configure Mongrel for Production ¶

Because everyone lays out their Production Directories differently, I'll just call the Production Root Directory **$PRODUCTION** in this article, and I'll assume that Application Instances are in subdirectories, which I'll call **$APP_ROOT**. For my Production Servers, I like to lay things out this way -- it makes things easier to create master startup scripts that can iterate over all $APP_ROOT instances in $PRODUCTION. I also configure $APP_PORT to a unique service port for each instance on my Production Server.

Please substitute my variables with your own directory structures as appropriate.

If you're following along with my layout, now's a good time to copy ~/testapp to your Production Area:

- sudo mv ~/testapp /$PRODUCTION
- sudo chown -R mongrel:mongrel /$PRODUCTION/testapp

Install Mongrel_Cluster a Startup Service ¶

- sudo mkdir /etc/mongrel_cluster
- find /usr/lib/ruby -type f -name "mongrel_cluster" -exec sudo cp -ap {} /etc/init.d/ \;
- sudo chmod +x /etc/init.d/mongrel_cluster
- sudo /sbin/chkconfig --level 345 mongrel_cluster on

Install Mongrel_Cluster Controller Tool ¶

(I've not found a use for this, but since others think it's important enough to document, I've put it in . . .)

- find /usr/lib/ruby -type f -name "mongrel_cluster_ctl" -exec sudo ln -s {} /usr/bin; \;

Configure Mongrel_Cluster For Each Application Instance ¶

(NOTE: The following hunk is a pseudo-script, just to give you an idea of what I do on my machines. It's not a complete bash script)

- export APP_PORT=8000 # Change this as needed
- export APP_NODES=3 # Change this as needed
- cd $PRODUCTION/$APP_ROOT
- sudo mongrel_rails cluster::configure -e production -c $PRODUCTION/$APP_ROOT -p $APP_PORT -N $APP_NODES -a 127.0.0.1 --user mongrel --group mongrel --prefix /$APP_ROOT
- sudo ln -s $PRODUCTION/$APP_ROOT/config/mongrel_cluster.yml /etc/mongrel_cluster/$APP_ROOT.yml

Note that my instructions above include the new **--prefix $APP_ROOT** command, which allows Mongrel Applications to properly "ignore" the prefix. Prior to this Mongrel Enhancement, we needed to configure Apache or Lighttpd to strip out this prefix.

If you're running your Rails Apps as the web server root application, remove the **--prefix /$APP_ROOT** additions.

Lighttpd Web Server ¶

At this point, you now have your Mongrel-Rails Applications properly configured to run as Startup Services. (They'll automatically startup when your machine reboots). Now it's time to stitch together the Lighttpd Web Server as our front-end.

Install Lighttpd ¶

- sudo yum -y install lighttpd
- sudo /sbin/chkconfig --level 345 lighttpd on

Configure Lighttpd For Mongrel ¶

- sudo nano /etc/lighttpd/lighttpd.conf

Uncomment the **mod_proxy** module, as we'll need that to dispatch requests to our Mongrel Serves.

Additional (Recommended) Modules For Lighttpd ¶

Out of the Box, Fedora's Lighttpd configuration is rather light. You may want to uncomment these additional modules to get more functionality.

- mod_rewrite
- mod_redirect
- mod_access
- mod_accesslog
- mod_compress

Configure A Mongrel-Cluster In Lighttpd ¶

Add a hunk of code similar to the following to the tail end of /etc/lighttpd/lighttpd.conf:

```
proxy.balance = "fair"
proxy.server  = ( "/testapp" =>
  ( ( "host" => "127.0.0.1", "port" => 8001 ),
    ( "host" => "127.0.0.1", "port" => 8002 ),
    ( "host" => "127.0.0.1", "port" => 8003 ) ) )
```

You will need to change the **/testapp** prefix to the name of your Rails Application. Remember that **$APP_ROOT** variable that we passed to Mongrel as **--prefix**? Yes, the value you input here must match that **--prefix** value. Obviously, you will also need to change the Mongrel Server-Ports to match your $APP_PORT, up to $APP_NODES instances for this server pool.

If you are running your apps in the root-directory, change "/testapp" to "/"

That should be it! Now you can fire up the whole shebang and cross your fingers:

- sudo service mongrel_cluster start
- sudo service lighttpd start

Ruby on Rails
RailsOnFedora

Here's a step-by-step guide for making Rails work with FastCGI? on Fedora Core 3 from scratch. A much simpler approach can be found at [http://linuxweblog.com/ruby-on-rails-install linuxweblog]

I began with a FC3 "Server" install. This tutorial is suitable to get Rails up and running on Apache2 with PostgreSQL

- Install your prerequisites. You're going to have to build some things from source on this voyage.

```
$ sudo yum install gcc
$ sudo yum install httpd-devel readline-devel postgresql php-pgsql zlib-devel apr
apr-devel apr-util-devel
```

- You're responsible for setting up PostgreSQL yourself. Go ahead and take care of that.
- Install more prerequisites by hand. FedoraCore3 seems to have some zlib files, but they're not actually installed? Under FC1, the zlib-devel package will work fine. (Not sure about FC3 however…) Anyway, download zlib from http://www.zlib.net/

```
$ ./configure
$ sudo make
$ sudo make install
```

If you are using Apache you will need mod_fcgi, and the FastCGI Development kit; These can be obtained from http://www.fastcgi.com . You will also need ruby-fcgi.

On Fedora Core 5 mod_fastcgi doesn't compile, because it uses some deprecated symbols that have gone in Apache httpd 2.2. I found the rpm available at http://www.city-fan.org/ftp/contrib/websrv/ to be very helpful.

- Install the FastCGI? Devel kit:

```
$ wget http://fastcgi.com/dist/fcgi-2.4.0.tar.gz
$ tar zxvf fcgi-2.4.0.tar.gz
$ cd fcgi-2.4.0
$ ./configure
$ sudo make
$ sudo make install
```

*NB:*FastCGI installs to `/usr/local/lib`, so if it isn't there already you will need to add `/usr/local/lib` to `/etc/ld.so.conf` and run:

```
$ ldconfig -v
```

- Install mod_fastcgi:

```
$ wget http://fastcgi.com/dist/mod_fastcgi-2.4.2.tar.gz
$ tar zxvf mod_fastcgi-2.4.2.tar.gz
$ cd mod_fastcgi-2.4.2
```

At this point, read INSTALL.AP2. You'll need to copy Makefile.AP2 to Makefile, and then edit the

Makefile.AP2 to point to an Apache base dir, probably `/etc/httpd` if you have the httpd package installed.

```
$ sudo make
$ sudo make install
```

*NB:*If you get the following error, install the httpd-devel package:

```
/usr/local/httpd/build/special.mk: No such file or directory
```

*NB:*If you use Apache2.2, and encounter following error:

```
'ap_null_cleanup' undeclared (first use in this function)
```

Please apply this patch to fcgi.h:

```
@@ -73,6 +73,36 @@
 #define ap_reset_timeout(a)
 #define ap_unblock_alarms()

+/* starting with apache 2.2 the backward-compatibility defines for
+ * 1.3 APIs are not available anymore. Define them ourselves here.
+ */
+#ifndef ap_copy_table
+
+#define ap_copy_table apr_table_copy
+#define ap_cpystrn apr_cpystrn
+#define ap_destroy_pool apr_pool_destroy
+#define ap_isspace apr_isspace
+#define ap_make_array apr_array_make
+#define ap_make_table apr_table_make
+#define ap_null_cleanup apr_pool_cleanup_null
+#define ap_palloc apr_palloc
+#define ap_pcalloc apr_pcalloc
+#define ap_psprintf apr_psprintf
+#define ap_pstrcat apr_pstrcat
+#define ap_pstrdup apr_pstrdup
+#define ap_pstrndup apr_pstrndup
+#define ap_push_array apr_array_push
+#define ap_register_cleanup apr_pool_cleanup_register
+#define ap_snprintf apr_snprintf
+#define ap_table_add apr_table_add
+#define ap_table_do apr_table_do
+#define ap_table_get apr_table_get
+#define ap_table_set apr_table_set
+#define ap_table_setn apr_table_setn
+#define ap_table_unset apr_table_unset
+
+#endif /* defined(ap_copy_table) */
+
 #if (defined(HAVE_WRITEV) && !HAVE_WRITEV && !defined(NO_WRITEV)) || defined WIN32
 #define NO_WRITEV
 #endif
```

- Install Ruby # From source: Download Ruby from http://www.ruby-lang.org/ # From RPM: (As of 02/10/2004, 1.8.2-1.FC3.1 packages are required)

  ```
  <code>$ sudo yum install ruby
  $ sudo yum install ruby-devel
  ```

```
$ sudo yum install rdoc
$ sudo yum install irb</code>
```

- Install rubygems Download “RubyGems”: from http://www.rubyforge.org/
- Install Rails and all its dependencies

  ```
  <code>$ sudo gem install rails</code>
  ```

- Install ruby-fcgi

  ```
  <code>$ sudo gem install fcgi</code>
  ```

  Note if the above command does not install fcgi version 0.8.6 or greater then unistall it and install 0.8.6 from http://raa.ruby-lang.org/list.rhtml?name=fcgi manually

- Install PostgreSQL bindings

  ```
  <code>$ sudo gem install postgres-pr</code>
  ```

- Create a text file at `/etc/httpd/conf.d/fastcgi.conf` with the following:

```
<code>LoadModule fastcgi_module modules/mod_fastcgi.so
<IfModule mod_fastcgi.c>
    FastCgiIpcDir /tmp/fcgi_ipc/
    AddHandler fastcgi-script .fcgi
</IfModule></code>
```

1. Restart apache with:

   ```
   <code>$ /etc/init.d/httpd restart</code>
   ```

1. At this point I followed the directions for Non VHost Installation (now HowToSetTheBaseURLsOfYourRailsApps) and it worked like a champ.
2. Don’t forget to change in `public/.htaccess`

   ```
   <code>RewriteRule ^(.*)$ /dispatch.cgi?$1 [QSA,L]</code>
   ```

   to

   ```
   <code>RewriteRule ^(.*)$ /dispatch.fcgi?$1 [QSA,L]</code>
   ```

Installing Ruby on Rails on Fedora Core 5

Submitted by sandip on Sat, 04/08/2006 - 23:17. Linux

Quick notes on installing Ruby on Rails on Fedora Core 5 (It should be similar for other linux distros as well)

1. Install ruby rpms via yum:

   ```
   # yum install ruby ruby-libs ruby-mode ruby-rdoc ruby-irb ruby-ri ruby-docs
   ```

2. Download and install rubygems from rubygems.org.
   Change to the extracted directory and run:

   ```
   # ruby setup.rb
   ```

3. Now use gem to install rails. It will ask about installing dependencies. Answer "Y" or just hit return.

   ```
   # gem install rails
   ```

4. Test it by creating a skeleton rails app in your home directory:

   ```
   $ cd ~
   $ rails testapp
   ```

5. Start the WEBrick server.

   ```
   $ cd ~/testapp
   $ ruby script/server
   ```

   The WEBrick server should now be started and listening to the default port - 3000 .
   Point your browser to:

   ```
   http://localhost:3000/
   ```

   You should see a welcome page with some additional getting started info.