

Mongrel, Apache, and Nginx

ENDAX MANUALS

Content is freely available for non commercial, non Profit, educational purposes provided the content is not modified and the author / publisher are given full credit for their work. Permission to reproduce for any other use in any format is strictly forbidden without the copyright holders permission.

©2008 ENDAX

Mongrel

While there are a number of configurations for serving production Rails applications such as FastCGI, SCGI, Mongrel has proven to be simple, easily configurable, and powerful option. Mongrel is fundamentally a HTTP library and server for Ruby applications. It isn't a single process server but due to the fact that Rails is not thread safe it therefore must be load balanced by either a TCP/HTTP balancer, a supporting HTTP server, or a hardware solution. We will review both Nginx and Apache in the following chapter to explain this in greater detail. But for now, simply install the Gem

Installing

From you command line, simply enter:

```
gem install mongrel -y
```

This will install the base Gem after possibly asking you which distribution. You should of course specify the latest **non-Win32** version. For example, at the writing of this manual, I had the following options:

```
Select which gem to install for your platform
1. mongrel 1.1.3 (java)
2. mongrel 1.1.3 (i386-mswin32)
3. mongrel 1.1.3 (ruby)
```

Select #3 in this case. Installation can take a few minutes due to the compilation necessary, etc.

Finally ensure that a symbolic link has been created for you:

```
ln -s /var/lib/gems/1.8/bin/mongrel_rails \
/usr/bin/mongrel_rails
```

Running

Mongrel comes with a host of options for starting, stopping and restarting and can be summary with the following:

```
mongrel_rails [command] -h
```

For **start**, here is break down of the command line options:

-e, --environment ENV	Rails environment to run as
-d, --daemonize	Run daemonized in the background
-p, --port PORT	Which port to bind to
-a, --address ADDR	Address to bind to
-l, --log FILE	Where to write log messages
-P, --pid FILE	Where to write the PID
-n, --num-procs INT	Number of processors active before clients denied
-t, --timeout TIME	Timeout all requests after 100th seconds time
-m, --mime PATH	A YAML file that lists additional MIME types
-c, --chdir PATH	Change to dir before starting (will be expanded)
-r, --root PATH	Set the document root (default 'public')
-B, --debug	Enable debugging mode
-C, --config PATH	Use a config file
-S, --script PATH	Load the given file as an extra config script
-G, --generate PATH	Generate a config file for use with -C
--user USER	User to run as
--group GROUP	Group to run as
--prefix PATH	URL prefix for Rails app
-h, --help	Show this breakdown for start command
--version	Show version

For `stop` and `restart`, here is break down of the command line options:

-c, --chdir PATH	Change to dir before starting (will be expanded).
-f, --force	Force the shutdown (kill -9) [<code>stop</code> only]
-s, --soft	Do a soft restart rather than a process exit restart [<code>restart</code> only]
-w, --wait SECONDS	Wait SECONDS before forcing shutdown

<code>-P, --pid FILE</code>	Where the PID file is located.
<code>-h, --help</code>	Show this message
<code>--version</code>	Show version

Mongrel can be run as a single instance from inside your project's root directory:

```
cd /project/root
mongrel_rails start
```

or simply use the server script which will default to Mongrel if it is installed:

```
cd /project/root
./script/server
```

Of course you want to eventually use a Mongrel cluster as part of a production deployment; fortunately, there is a gem for just that.

Mongrel_Cluster

Mongrel Cluster was developed to simplify a mongrel cluster configuration. First you will need to install the gem:

```
gem install mongrel_cluster -y
```

Once installed we will need a configuration file in the config directory of your application:

```
cd /project/root
mongrel_rails cluster::configure -e production -p \
8000 -N 3 -c /var/www/myproject -a 127.0.0.1
```

This will create a `mongrel_cluster.yml` file in your config directory:

```
---
cwd: /var/www/rv
log_file: log/mongrel.log
port: "8000"
environment: production
address: 127.0.0.1
pid_file: tmp/pids/mongrel.pid
servers: 3
```

that will be used to starting the Mongrel Cluster. At this point we going to stop here and return to the final stages of the Mongrel Cluster configuration after we have introduced Capistrano in a subsequent manual. This is primarily due to the fact that Capistrano introduces a slightly different directory structure and there's no reason to configure this twice.

Apache / Nginx

Both legitimate, powerful solutions, Nginx is very easy to configure but has some limitations when compared to Apache; notably it doesn't support streaming media when used as a load balancer as of the current release. Regardless, we will cover both here since it is likely that your requirements will dictate one or the other. Also be sure to choose one option, running both doesn't make any sense.

Apache

Apache needs no introduction as a the most prolific HTTP server of the Internet. As with Nginx, what we really need is a load balancer and not necessarily a HTTP server.

Apache Installing

It's likely that Apache is intalled in your distribution, and likely running already but just in case:

```
sudo apt-get install apache2
```

It's also worth ensuring that is is started on restart using the System-5 `update-rc.d` [specific to Debian, varies by distribution, e.g. Red Hat is `chkconfig`].

```
update-rc.d apache2 defaults
```

It is now possible to stop, start, and restart the apache server by using the provided script

```
/etc/init.d/apache2 start
```

Apache Configuration

At this point you are ready to configure you application as a load balancer using Apache. The first thing to do is stop Apache if it is running:

```
/etc/init.d/apache2 stop
```

Now enable the necessary modules in Apache by moving to the directory

and making a series of symbolic links to enable them:

```
cd /etc/apache2/mods-enabled/  
  
ln -s ../mods-available/rewrite.load rewrite.load  
ln -s ../mods-available/proxy.load proxy.load  
ln -s ../mods-available/proxy_balancer.load \  
proxy_balancer.load  
ln -s ../mods-available/proxy_http.load \  
proxy_http.load  
ln -s ../mods-available/deflate.load deflate.load  
ln -s ../mods-available/deflate.conf deflate.conf
```

[do we need the deflate?]

Also we are going to set up your application as the default application and not use a traditional VirtualHost; this is, of course, the likely scenario that you will face.

Regardless though, you still should create the necessary configuration files specific to you application. This is accomplished by create a series of file with your application's name as the prefix. For example, if your application is named `myapp` then:

1. `myapp.common`
2. `myapp.conf`
3. `myapp.proxy_cluster.conf`

Change to you conf directory:

```
cd /etc/apache2/conf
```

For `myapp.common` use:

```
# myapp.common  
ServerName myserver  
DocumentRoot /var/www/myapp/current/public  
  
<Directory "/var/www/myapp/current/public">  
  Options FollowSymLinks  
  AllowOverride None  
  Order allow,deny  
  Allow from all  
</Directory>  
  
RewriteEngine On  
  
# Check for maintenance file and redirect all requests  
RewriteCond %{DOCUMENT_ROOT}/system/maintenance.html
```

```

-f
RewriteCond %{SCRIPT_FILENAME} !maintenance.html
RewriteRule ^.*$ /system/maintenance.html [L]

# Rewrite index to check for static
RewriteRule ^/$ /index.html [QSA]

# Rewrite to check for Rails cached page
RewriteRule ^([\^.]*)$ $1.html [QSA]

# Redirect all non-static requests to cluster
RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} !-f
RewriteRule ^/(.*)$
balancer://mongrel_cluster%{REQUEST_URI} [P,QSA,L]

# Deflate
AddOutputFilterByType DEFLATE text/html text/plain
text/xml application/xml application/xhtml+xml
text/javascript text/css
BrowserMatch ^Mozilla/4 gzip-only-text/html
BrowserMatch ^Mozilla/4.0[678] no-gzip
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

```

For `myapp.conf` use:

```

<VirtualHost *:80>
  Include /etc/apache2/conf.d/myapp.common
  ErrorLog /etc/apache2/logs/myapp_errors_log
  CustomLog /etc/apache2/logs/myapp_log combined
</VirtualHost>

```

For `myapp.proxy_cluster.conf` use:

```

<Proxy balancer://mongrel_cluster>
  BalancerMember http://127.0.0.1:8000
  BalancerMember http://127.0.0.1:8001
  BalancerMember http://127.0.0.1:8002
</Proxy>

```

Also be sure to create a log directory if necessary:

```
mkdir /etc/apache2/logs
```

And finally, disable the default site by editing your Apache config file:

```
vi /etc/apache2/apache2.conf
```

and comment out the last line:

```
#Include /etc/apache2/sites-enabled/
```

In the following chapter we will introduce starting Apache as part of a deployment.

Nginx

Nginx (<http://nginx.net/>) is high performance HTTP server / reverse proxy and is known (mostly in Russia) for its simple configuration, stability, low consumption, and has been in production use for a number of years now. Nginx is pronounced /En-Gin-X/.

Nginx Installing

As root (or `sudo`), stop the running version of Apache:

```
/etc/init.d/apache2 stop
```

And install nginx:

```
apt-get install nginx
```

This will install a startup script in your `init.d` directory and usage is similar to most other scripts, for example to start:

```
/etc/init.d/nginx start
```

Also ensure that it is started on restart [and necessary to stop Apache from restarting as well] using the `System-5 update-rc.d`. To remove the existing Apache:

```
update-rc.d -f apache2 remove
```

Your new nginx installation should have created this start up links [check in the `/etc/rc3.d` directory] but if not:

```
update-rc.d nginx defaults
```

Now you are ready to configure it for the Mongrel Cluster (and other options).

Nginx Configuration

What you are going to setup is similar to the Apache configuration above, essentially a reverse proxy that handle and coordinate the distribution of incoming requests to available mongrel instances.

It's always a good idea to backup the existing configuration file though with the following:

```
mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.old
```

Now you are ready to create a new configuration file. Using vi or you favorite editor create a new nginx.conf file:

```
vi /etc/nginx/nginx.conf
```

And insert the following, taking note to change `server_name` and `root` values to the correct name of you server and you root PATH:

```
#
# NGINX.conf

worker_processes 1;
error_log /var/log/nginx/error.log;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    access_log /var/log/nginx/access.log;
    sendfile on;
    keepalive_timeout 65;
    tcp_nodelay on;
    gzip on;
    # the cluster of mongrels
    upstream mongrel {
        server 127.0.0.1:8000;
        server 127.0.0.1:8001;
        server 127.0.0.1:8002;
    }
    server {
        listen 80;
        server_name myserver.com; #EDIT!
        root /var/www/myapp/current/public; #EDIT!
        access_log /var/log/nginx/localhost.access.log;
        location / {
```

```
# forward user's real IP address
proxy_set_header X-Real-IP $remote_addr;
# required for HTTPS
proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
proxy_set_header Host $http_host;
proxy_redirect false;
proxy_max_temp_file_size 0;

# Avoid the rewrite tests if static file..
if (-f $request_filename) {
    break;
}

# index.html present..
if (-f $request_filename/index.html) {
    rewrite (.*) $1/index.html break;
}

# page caching..
if (-f $request_filename.html) {
    rewrite (.*) $1.html break;
}
if (!-f $request_filename) {
    proxy_pass http://mongrel;
    break;
}
}

error_page 500 502 503 504 /50x.html;
location = /50x.html {
    root /var/www/nginx-default;
}

}

}
```