

[Ultimate Open Source Web Server Part 3](#)

de [Articles and Tutorials | Lehi Sanchez](#) de Lehi Sanchez

In [Part 1](#), we installed Ubuntu Server 8.04, OpenSSH and configured a basic firewall with the iptables. In [Part 2](#), we installed our databases, server-side languages, web servers, the Rails framework, and version control. In this article, Part 3, we'll configure NGINX and FastCGI. After we're done with that we'll be installing [phpMyAdmin](#) and [SQLBuddy](#) as well as setting up some basic Hello World! websites using content management systems and frameworks such as [Wordpress](#), [Joomla!](#), [Drupal](#), [CakePHP](#) and [Ruby on Rails](#).

In this article, I am using the server username 'lsanchez' as well as port '8888' for ssh. When you see this in the code be sure to replace it with YOUR server username and the port you chose in the earlier tutorials.

NGINX

We'll need to tweak NGINX before we start installing applications.

```
sudo kill `cat /usr/local/nginx/logs/nginx.pid`
```

```
sudo nano /etc/init.d/nginx
```

Copy and paste the code below into the window and save the file.

```
1  #! /bin/sh
2
3  ### BEGIN INIT INFO
4  # Provides:          nginx
5  # Required-Start:    $all
6  # Required-Stop:    $all
7  # Default-Start:    2 3 4 5
8  # Default-Stop:     0 1 6
9  # Short-Description: starts the nginx web server
10 # Description:       starts nginx using start-stop-daemon
11 ### END INIT INFO
12
13 PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
14 DAEMON=/usr/local/sbin/nginx
15 NAME=nginx
16 DESC=nginx
17
18 test -x $DAEMON || exit 0
19
20 # Include nginx defaults if available
21 if [ -f /etc/default/nginx ] ; then
22     . /etc/default/nginx
23 fi
24
25 set -e
26
27 case "$1" in
28     start)
29         echo -n "Starting $DESC: "
30         start-stop-daemon --start --quiet --pidfile /usr/local/nginx/logs/$NAME.pid \
31             --exec $DAEMON -- $DAEMON_OPTS
32         echo "$NAME."
33         ;;
34     stop)
35         echo -n "Stopping $DESC: "
36         start-stop-daemon --stop --quiet --pidfile /usr/local/nginx/logs/$NAME.pid \
37             --exec $DAEMON
38         echo "$NAME."
39     *)
40         echo "Usage: $0 {start|stop}"
41         exit 1
42     esac
```

```

38     echo $NAME.
39     ;;
40 restart|force-reload)
41     echo -n "Restarting $DESC: "
42     start-stop-daemon --stop --quiet --pidfile \
43         /usr/local/nginx/logs/$NAME.pid --exec $DAEMON
44     sleep 1
45     start-stop-daemon --start --quiet --pidfile \
46         /usr/local/nginx/logs/$NAME.pid --exec $DAEMON -- $DAEMON_OPTS
47     echo "$NAME."
48     ;;
49 reload)
50     echo -n "Reloading $DESC configuration: "
51     start-stop-daemon --stop --signal HUP --quiet --pidfile /usr/local/nginx/logs/$NAME.pid
52 \
53     --exec $DAEMON
54     echo "$NAME."
55     ;;
56 *)
57 N=/etc/init.d/$NAME
58 echo "Usage: $N {start|stop|restart|reload|force-reload}" >&2
59 exit 1
60 ;;
61 esac
62
63 exit 0

```

```
sudo chmod +x /etc/init.d/nginx
```

```
sudo /usr/sbin/update-rc.d -f nginx defaults
```

These are the commands to start, stop, or restart NGINX. We'll be using these a few times throughout the tutorial.

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

```
sudo /etc/init.d/nginx stop
```

```
sudo /etc/init.d/nginx restart
```

Setting up Virtual Hosting in NGINX

```
sudo mkdir /usr/local/nginx/sites-available
```

```
sudo mkdir /usr/local/nginx/sites-enabled
```

```
sudo nano /usr/local/nginx/conf/nginx.conf
```

Copy and paste the code below into the window and save the file.

```

1 user www-data www-data;
2
3 # max_clients = worker_processes * worker_connections
4 worker_processes 4;
5 events {
6     worker_connections 1024;
7 }
8
9 http {
10     include mime.types;
11     default_type application/octet-stream;
12
13     sendfile on;
14
15     tcp_nopush on;
16     tcp_nodelay off;

```

```

17 keepalive_timeout 3;
18
19 gzip on;
20 gzip_comp_level 2;
21 gzip_proxied any;
22 gzip_types          text/plain text/html text/css application/x-javascript text/xml
23 application/xml
24 application/xml+rss text/javascript;
25
26 include /usr/local/nginx/sites-enabled/*;
27 }

```

```
sudo mkdir /home/lsanchez/public_html
```

```
sudo mkdir /home/lsanchez/public_html/default
```

```
sudo nano /home/lsanchez/public_html/default/index.html
```

Copy and paste the code below into the window and save the file.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
1  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
2
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
4   <head>
5     <title>Apollo Server</title>
6     <style type="text/css" media="screen">
7       body {font: 62.5% "Lucida Grande", "Trebuchet MS", Verdana, sans-serif;text-align:
8 center; padding-top:20px;background-color: #2288bb;}
9       #container {text-align: left; margin:0px auto; width:800px; background-color: #fff;
10 border: 3px solid #000; padding: 15px;}
11     </style>
12   </head>
13
14   <body>
15     <div id="container">
16       <div id="content">
17         <h1>Apollo</h1>
18       </div>
19     </div>
20 </body>
</html>

```

```
sudo nano /usr/local/nginx/sites-available/default
```

Copy and paste the code below into the window and save the file.

```

1 server {
2   listen 80;
3   server_name localhost;
4
5   location / {
6     root /home/lsanchez/public_html/default;
7     index index.html;
8   }
9 }

```

```
sudo ln -s /usr/local/nginx/sites-available/default /usr/local/nginx/sites-enabled/default
```

```
sudo /etc/init.d/nginx restart
```

If you go to your slice ip address in your browser you should now see your default site (Apollo). We just configured NGINX and setup a default site. Now let's get PHP working so we can begin installing some bigger sites and applications. First, let's modify our default site to display php files.

```
sudo nano /home/lsanchez/public_html/default/index.php
```

Copy and paste the code below into the window and save the file.

```
<?php echo phpinfo(); ?>
```

Let's modify our NGINX settings for the default site.

```
sudo nano /usr/local/nginx/sites-available/default
```

On the line where you see:

```
index index.html;
```

Change it to read

```
index index.php
```

Restart NGINX

```
sudo /etc/init.d/nginx restart
```

If you refresh your browser - the server will want you to download the index.php file. Let's fix that using FastCGI.

FastCGI

```
sudo nano /usr/local/nginx/fastcgi_params
```

Copy and paste the code below into the window and save the file.

```
1 fastcgi_param QUERY_STRING      $query_string;
2 fastcgi_param REQUEST_METHOD    $request_method;
3 fastcgi_param CONTENT_TYPE      $content_type;
4 fastcgi_param CONTENT_LENGTH    $content_length;
5
6 fastcgi_param SCRIPT_NAME        $fastcgi_script_name;
7 fastcgi_param REQUEST_URI        $request_uri;
8 fastcgi_param DOCUMENT_URI       $document_uri;
9 fastcgi_param DOCUMENT_ROOT      $document_root;
10 fastcgi_param SERVER_PROTOCOL    $server_protocol;
11
12 fastcgi_param GATEWAY_INTERFACE  CGI/1.1;
13 fastcgi_param SERVER_SOFTWARE    nginx;
14
15 fastcgi_param REMOTE_ADDR        $remote_addr;
16 fastcgi_param REMOTE_PORT        $remote_port;
17 fastcgi_param SERVER_ADDR        $server_addr;
18 fastcgi_param SERVER_PORT        $server_port;
19 fastcgi_param SERVER_NAME        $server_name;
20
21 # PHP only, required if PHP was built with --enable-force-cgi-redirect
22 # fastcgi_param REDIRECT_STATUS  200;
```

Again, let's modify NGINX's file for the default site

```
sudo nano /usr/local/nginx/sites-available/default
```

- Add another location parameter to make it look like the following as well as change the index to point to the new **index.php** file.

```

1 server {
2     listen 80;
3     server_name localhost;
4
5     location / {
6         root /home/lsanchez/public_html/default;
7         index index.php;
8     }
9
10    location ~ /\.php$ {
11        fastcgi_pass 127.0.0.1:9999;
12        fastcgi_index index.php;
13        fastcgi_param SCRIPT_FILENAME
14 /home/lsanchez/public_html/default$fastcgi_script_name;
15        include /usr/local/nginx/fastcgi_params;
16    }

```

```
sudo /etc/init.d/nginx stop
```

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

This still doesn't get us what we want - but we're close!

```
sudo nano /usr/bin/php-fastcgi
```

Copy and paste the code below into the window and save the file.

```

1 #!/bin/sh
2 /usr/bin/spawn-fcgi -a 127.0.0.1 -p 9999 -u www-data -f /usr/bin/php5-cgi

```

```
sudo nano /etc/init.d/init-fastcgi
```

Copy and paste the code below into the window and save the file.

```

1 #!/bin/bash
2 PHP_SCRIPT=/usr/bin/php-fastcgi
3 RETVAL=0
4 case "$1" in
5 start)
6 $PHP_SCRIPT
7 RETVAL=$?
8 ;;
9 stop)
10 killall -9 php
11 RETVAL=$?
12 ;;
13 restart)
14 killall -9 php
15 $PHP_SCRIPT
16 RETVAL=$?
17 ;;
18 *)
19 echo "Usage: php-fastcgi {start|stop|restart}"
20 exit 1
21 ;;
22 esac
23 exit $RETVAL

```

```
sudo chmod 755 /usr/bin/php-fastcgi
```

```
sudo chmod 755 /etc/init.d/init-fastcgi
```

```
/etc/init.d/init-fastcgi start  
sudo update-rc.d init-fastcgi defaults
```

Go ahead and refresh the browser. You should see PHP information now!

Local Virtual Hosts

Let's make it easier for us by setting up virtual hosts on our computers so we can access all of these applications that we're going to install in a simplified manner. So far, we only have a default web site that shows us PHP information, instead of typing in the IP Address into the browser when we want to navigate to each application, let's assign domain names for each one. Let's give our server a domain name that we never actually use - something like 'slice.com'. We can give each application an appropriate domain name like so - phpmyadmin.slice.com, sqlbuddy.slice.com, wordpress.slice.com, etc.

On your **local** machine type the following

```
sudo pico /etc/hosts
```

then scroll down to the next available line and add the following (REPLACE '127.0.0.1' WITH THE IP ADDRESS OF YOUR SERVER)

```
1 127.0.0.1 slice.com  
2 127.0.0.1 phpmyadmin.slice.com  
3 127.0.0.1 sqlbuddy.slice.com  
4 127.0.0.1 cakephp.slice.com  
5 127.0.0.1 wordpress.slice.com  
6 127.0.0.1 joomla.slice.com  
7 127.0.0.1 drupal.slice.com  
8 127.0.0.1 rubyonrails.slice.com
```

Save that file and then in your browser type in "slice.com". You should still see the PHP information. Let's start installing stuff.

SFTP

We'll be uploading files to our server. This is easy to do if you have an FTP program that has an SFTP setting.

1. Host: IP Address of your server
2. Username: This one is obviously your server login username
3. Password: This is your server login password
4. Port: MAKE SURE PORT IS SET TO THE SAME PORT YOU LOGIN BY (eg. '-8888')
5. Protocol: Set this to 'SFTP'

When you connect just navigate to the public_html folder. All of our sites will be located in there.

Database Management

We're going to setup 2 web based database management systems. PHPMYAdmin and a little known system I've been looking at called SQLBuddy. We're going to install both of these and you can judge which one you like more.

SQLBuddy

Download and then upload the [SQLBuddy](#) source files to a new folder in public_html. You can name the

folder what you wish but I like to keep the folder names consistent with the domain names I'll be using with the site. In this case I have a folder name of 'sqlbuddy.slice.com'

Setting up a new site

*** It would be helpful to memorize the following sequence when setting up new sites - we'll be repeating it quite a few times.**

After creating the sites folder in 'public_html' do the following for each (we'll use SQLBuddy as our model to follow from this point on with all of our PHP sites)

```
sudo nano /usr/local/nginx/sites-available/sqlbuddy.slice.com

1 server {
2     listen 80;
3     server_name sqlbuddy.slice.com;
4
5     location / {
6         root    /home/lsanchez/public_html/sqlbuddy.slice.com;
7         index  index.php;
8     }
9
10    location ~ /\.php$ {
11        fastcgi_pass 127.0.0.1:9999;
12        fastcgi_index index.php;
13        fastcgi_param SCRIPT_FILENAME
14 /home/lsanchez/public_html/sqlbuddy.slice.com$fastcgi_script_name;
15        include /usr/local/nginx/fastcgi_params;
16    }
17 }
```

```
sudo ln -s /usr/local/nginx/sites-available/sqlbuddy.slice.com
/usr/local/nginx/sites-enabled/sqlbuddy.slice.com

sudo /etc/init.d/nginx restart
```

That's all. If you now go to sqlbuddy.slice.com you should see the login screen - all you need to do is type in the root user's password you used when setting up MySQL.

phpMyAdmin

This will be a little different since we're going to let Ubuntu install PHP for us. Let's let it do that and then we'll set up the site afterward since we don't know where Ubuntu's going to put it.

```
sudo apt-get install phpmyadmin

sudo nano /usr/local/nginx/sites-available/phpmyadmin.slice.com

server {
    listen 80;
    server_name phpmyadmin.slice.com;

    location / {
        root    /usr/share/phpmyadmin;
        index  index.php;
    }

    location ~ /\.php$ {
        fastcgi_pass 127.0.0.1:9999;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME /usr/share/phpmyadmin$fastcgi_script_name;
        include /usr/local/nginx/fastcgi_params;
    }
}
```

```
}
```

```
sudo ln -s /usr/local/nginx/sites-available/phpmyadmin.slice.com  
/usr/local/nginx/sites-enabled/phpmyadmin.slice.com
```

```
sudo rm /usr/share/phpmyadmin/config.inc.php
```

```
sudo nano /usr/share/phpmyadmin/config.inc.php
```

Copy and paste the code below into the window and save the file and be sure to fill in the variable for the blowfish_secret variable.

```
1 <?php  
2 /* vim: set expandtab sw=4 ts=4 sts=4: */  
3 /**  
4  * phpMyAdmin sample configuration, you can use it as base for  
5  * manual configuration. For easier setup you can use scripts/setup.php  
6  *  
7  * All directives are explained in Documentation.html and on phpMyAdmin  
8  * wiki <http://wiki.cihar.com>.  
9  *  
10 * @version $Id: config.sample.inc.php 10142 2007-03-20 10:32:13Z cybot_tm $  
11 */  
12  
13 /*  
14 * This is needed for cookie based authentication to encrypt password in  
15 * cookie  
16 */  
17 $cfg['blowfish_secret'] = ''; /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */  
18  
19 /*  
20 * Servers configuration  
21 */  
22 $i = 0;  
23  
24 /*  
25 * First server  
26 */  
27 $i++;  
28 /* Authentication type */  
29 $cfg['Servers'][$i]['auth_type'] = 'cookie';  
30 /* Server parameters */  
31 $cfg['Servers'][$i]['host'] = 'localhost';  
32 $cfg['Servers'][$i]['connect_type'] = 'tcp';  
33 $cfg['Servers'][$i]['compress'] = false;  
34 /* Select mysqli if your server has it */  
35 $cfg['Servers'][$i]['extension'] = 'mysql';  
36 /* User for advanced features */  
37 // $cfg['Servers'][$i]['controluser'] = 'pma';  
38 // $cfg['Servers'][$i]['controlpass'] = 'pmapass';  
39 /* Advanced phpMyAdmin features */  
40 // $cfg['Servers'][$i]['pmadb'] = 'phpmyadmin';  
41 // $cfg['Servers'][$i]['bookmarktable'] = 'pma_bookmark';  
42 // $cfg['Servers'][$i]['relation'] = 'pma_relation';  
43 // $cfg['Servers'][$i]['table_info'] = 'pma_table_info';  
44 // $cfg['Servers'][$i]['table_coords'] = 'pma_table_coords';  
45 // $cfg['Servers'][$i]['pdf_pages'] = 'pma_pdf_pages';  
46 // $cfg['Servers'][$i]['column_info'] = 'pma_column_info';  
47 // $cfg['Servers'][$i]['history'] = 'pma_history';  
48 // $cfg['Servers'][$i]['designer_coords'] = 'pma_designer_coords';  
49  
50 /*  
51 * End of servers configuration  
52 */  
53  
54 /*  
55 * Directories for saving/loading files from server  
56 */  
57 $cfg['UploadDir'] = '';  
58 $cfg['SaveDir'] = '';  
59  
...
```

```
60 ?>
```

```
sudo /etc/init.d/nginx restart
```

Navigate to phpmyadmin.slice.com in your browser - you should now see the login page. Login in using 'root' for the username and then the password you chose for the MySQL installation.

PHP Content Management Systems

Wordpress

This should be as easy as the SQLBuddy installation. Download the [Wordpress](#) source files and upload them to a new folder in the 'public_html' folder. I'll name my folder 'wordpress.slice.com'. Now that we have that taken care of - let's repeat the steps we took for our SQLBuddy installation.

```
sudo nano /usr/local/nginx/sites-available/wordpress.slice.com
```

```
1 server {
2     listen            80;
3     server_name      wordpress.slice.com;
4
5     location / {
6         root          /home/lsanchez/public_html/wordpress.slice.com;
7         index         index.php;
8
9         # this serves static files that exist without running other rewrite tests
10        if (-f $request_filename) {
11            expires 30d;
12            break;
13        }
14
15        # this sends all non-existing file or directory requests to index.php
16        if (!-e $request_filename) {
17            rewrite ^(.+)$ /index.php?q=$1 last;
18        }
19    }
20
21    location ~ /\.php$ {
22        fastcgi_pass 127.0.0.1:9999;
23        fastcgi_index index.php;
24        fastcgi_param SCRIPT_FILENAME
25        /home/lsanchez/public_html/wordpress.slice.com$fastcgi_script_name;
26        include /usr/local/nginx/fastcgi_params;
27    }
```

```
sudo ln -s /usr/local/nginx/sites-available/wordpress.slice.com
/usr/local/nginx/sites-enabled/wordpress.slice.com
```

```
sudo /etc/init.d/nginx restart
```

Wordpress will need you to setup the configuration file.

```
sudo rm /home/lsanchez/public_html/wordpress.slice.com/wp-config-sample.php
```

```
sudo nano /home/lsanchez/public_html/wordpress.slice.com/wp-config.php
```

Copy and paste the code below into the window and save the file and be sure to fill in the password with the one you chose when installing MySQL.

```
<?php
// ** MySQL settings ** //
```

```

// mysql settings //
1 define('DB_NAME', 'wordpress'); // The name of the database
2 define('DB_USER', 'root'); // Your MySQL username
3 define('DB_PASSWORD', ''); // ...and password
4 define('DB_HOST', 'localhost'); // 99% chance you won't need to change this value
5 define('DB_CHARSET', 'utf8');
6 define('DB_COLLATE', '');
7
8 // Change each KEY to a different unique phrase. You won't have to remember the
9 phrases later,
10 // so make them long and complicated. You can visit
11 http://api.wordpress.org/secret-key/1.1/
12 // to get keys generated for you, or just make something up. Each key should have a
13 different phrase.
14 define('AUTH_KEY', 'put your unique phrase here'); // Change this to a unique phrase.
15 define('SECURE_AUTH_KEY', 'put your unique phrase here'); // Change this to a unique
16 phrase.
17 define('LOGGED_IN_KEY', 'put your unique phrase here'); // Change this to a unique
18 phrase.
19
20 // You can have multiple installations in one database if you give each a unique prefix
21 $table_prefix = 'wp_'; // Only numbers, letters, and underscores please!
22
23 // Change this to localize WordPress. A corresponding MO file for the
24 // chosen language must be installed to wp-content/languages.
25 // For example, install de.mo to wp-content/languages and set WPLANG to 'de'
26 // to enable German language support.
27 define ('WPLANG', '');
28
29 /* That's all, stop editing! Happy blogging. */
30
31 if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');
    require_once(ABSPATH . 'wp-settings.php');
    ?>

```

Before we navigate to wordpress.slice.com - we need to create a database for Wordpress to use. Open up either [SQLBuddy](#) or [phpMyAdmin](#) - login - and then create a new database called 'wordpress'. After we've created the database you can now go to wordpress.slice.com and you should see the installation page.

Permalinks

You might have noticed this extra code in Wordpress' virtual server file. This helps Wordpress keep pretty urls when you change the permalinks settings in the admin section. Let's hope it works with the other php installations!

```

9 # this serves static files that exist without running other rewrite tests
10 if (-f $request_filename) {
11     expires 30d;
12     break;
13 }
14
15 # this sends all non-existing file or directory requests to index.php
16 if (!-e $request_filename) {
17     rewrite ^(.+)$ /index.php?q=$1 last;
18 }

```

Joomla!

Download and upload the [Joomla!](#) source code files and follow the steps we took with Wordpress.

```
sudo nano /usr/local/nginx/sites-available/joomla.slice.com
```

```

1 server {
2     listen 80;
    server name joomla.slice.com;

```

```

3  server_name joomla.slice.com;
4
5  location / {
6      root    /home/lsanchez/public_html/joomla.slice.com;
7      index  index.php;
8
9      # this serves static files that exist without running other rewrite tests
10     if (-f $request_filename) {
11         expires 30d;
12         break;
13     }
14
15     # this sends all non-existing file or directory requests to index.php
16     if (!-e $request_filename) {
17         rewrite ^(.+)$ /index.php?q=$1 last;
18     }
19 }
20
21 location ~ /\.php$ {
22     fastcgi_pass 127.0.0.1:9999;
23     fastcgi_index index.php;
24     fastcgi_param SCRIPT_FILENAME
25 /home/lsanchez/public_html/joomla.slice.com$fastcgi_script_name;
26     include /usr/local/nginx/fastcgi_params;
27 }

```

```

sudo ln -s /usr/local/nginx/sites-available/joomla.slice.com
/usr/local/nginx/sites-enabled/joomla.slice.com

```

```

sudo /etc/init.d/nginx restart

```

Let's go to joomla.slice.com in our browser and see what happened. Follow the installation instructions. When you come to the database setup page - just go into [SQLBuddy](#) or [phpMyAdmin](#) and create a database for Joomla!. When you've created the database - on the Joomla! db setup page enter the following:

1. Database type: MySQL
2. Host Name: localhost
3. Username: root
4. Password: *your password you chose when installing MySQL
5. Database Name: *name of the database you just created for Joomla!

Follow Joomla!'s instructions. You should be set.

Drupal

You know what to do. Download and then upload [Drupal's](#) source code to a new folder in the 'public_html' folder. I called my folder 'drupal.slice.com'. Then follow the usual pattern (eg. Wordpress installation steps)

```

sudo nano /usr/local/nginx/sites-available/drupal.slice.com

```

```

1  server {
2      listen          80;
3      server_name    drupal.slice.com;
4
5      location / {
6          root    /home/lsanchez/public_html/drupal.slice.com;
7          index  index.php;
8
9          # this serves static files that exist without running other rewrite tests
10         if (-f $request_filename) {
11             expires 30d;
12             break;

```

```

12     }
13 }
14 # this sends all non-existing file or directory requests to index.php
15 if (!-e $request_filename) {
16     rewrite ^(.+)$ /index.php?q=$1 last;
17 }
18 }
19 }
20 location ~ /\.php$ {
21     fastcgi_pass 127.0.0.1:9999;
22     fastcgi_index index.php;
23     fastcgi_param SCRIPT_FILENAME
24 /home/lsanchez/public_html/drupal.slice.com$fastcgi_script_name;
25     include /usr/local/nginx/fastcgi_params;
26 }
27 }

```

```

sudo ln -s /usr/local/nginx/sites-available/drupal.slice.com
/usr/local/nginx/sites-enabled/drupal.slice.com

```

```

sudo /etc/init.d/nginx restart

```

Drupal needs to be configured just a little bit more.

```

sudo mkdir /home/lsanchez/public_html/drupal.slice.com/sites/default/files

```

```

sudo chmod 777 /home/lsanchez/public_html/drupal.slice.com/sites/default/files

```

```

sudo cp /home/lsanchez/public_html/drupal.slice.com/sites/default/default.settings.php
/home/lsanchez/public_html/drupal.slice.com/sites/default/settings.php

```

We'll need to create a database for Drupal as well before navigation to 'drupal.slice.com'. Once a database has been created - navigate to drupal.slice.com and follow the installation instructions.

CakePHP

Download and then upload the [CakePHP](http://cakephp.org) source files to a new folder in the 'public_html' folder. I called it 'cakephp.slice.com'.

```

sudo nano /usr/local/nginx/sites-available/cakephp.slice.com

```

***notice in the code below that we need to point to CakePHP's webroot folder as the 'root' directory.**

```

1  server {
2      listen      80;
3      server_name  cakephp.slice.com;
4
5      location / {
6          root     /home/lsanchez/public_html/cakephp.slice.com/app/webroot;
7          index   index.php;
8
9          # this serves static files that exist without running other rewrite tests
10         if (-f $request_filename) {
11             expires 30d;
12             break;
13         }
14
15         # this sends all non-existing file or directory requests to index.php
16         if (!-e $request_filename) {
17             rewrite ^(.+)$ /index.php?q=$1 last;
18         }
19     }

```

```

1~
20 location ~ /\.php$ {
21     fastcgi_pass 127.0.0.1:9999;
22     fastcgi_index index.php;
23     fastcgi_param SCRIPT_FILENAME
24 /home/lsanchez/public_html/cakephp.slice.com$fastcgi_script_name;
25     include /usr/local/nginx/fastcgi_params;
26 }
27 }

```

```

sudo ln -s /usr/local/nginx/sites-available/cakephp.slice.com
/usr/local/nginx/sites-enabled/cakephp.slice.com

```

```

sudo /etc/init.d/nginx restart

```

CakePHP needs to be configured just a little bit more.

```

sudo chmod 777 -R /home/lsanchez/public_html/cakephp.slice.com/app/tmp

```

```

sudo nano /home/lsanchez/public_html/cakephp.slice.com/app/config/core.php

```

Find the following line and then change the value of 'Security.salt'

```

Configure::write('Security.salt', 'DYhG93b0qyJfIxfS2guVoUubWwvniR2G0FgaC9mi');

```

You'll need to create 2 databases for CakePHP - a default and test database. I've created 2 named 'cakephp_default' and 'cakephp_test'.

```

sudo cp /home/lsanchez/public_html/cakephp.slice.com/app/config/database.php.default
/home/lsanchez/public_html/cakephp.slice.com/app/config/database.php

```

```

sudo nano /home/lsanchez/public_html/cakephp.slice.com/app/config/database.php

```

My configuration file looks like this:

```

class DATABASE_CONFIG {

    var $default = array(
        'driver' => 'mysql',
        'persistent' => false,
        'host' => 'localhost',
        'login' => 'root',
        'password' => '*****',
        'database' => 'cakephp_default',
        'prefix' => '',
    );

    var $test = array(
        'driver' => 'mysql',
        'persistent' => false,
        'host' => 'localhost',
        'login' => 'root',
        'password' => '*****',
        'database' => 'cakephp_test',
        'prefix' => '',
    );
}

```

And there you have it. Wordpress, Joomla!, Drupal and CakePHP. Let's do one more. Ruby on Rails.

Thin

We'll need to tweak Thin web server first.

```
cd ~
sudo thin install
sudo /usr/sbin/update-rc.d -f thin defaults
```

Ruby on Rails

```
cd ~/public_html
rails todo
cd ~/public_html/todo
sudo thin config -C /etc/thin/todo.yml -c /home/lsanchez/public_html/todo/ --servers 3 -e
production
sudo /etc/init.d/thin start
```

Now let's set up the site.

```
sudo nano /usr/local/nginx/sites-available/rubyonrails.slice.com
```

This code will be a little different from the PHP files we did earlier.

```
1 upstream domain1 {
2     server 127.0.0.1:3000;
3     server 127.0.0.1:3001;
4     server 127.0.0.1:3002;
5 }
6
7 server {
8     listen 80;
9     server_name rubyonrails.slice.com;
10
11     access_log /home/lsanchez/public_html/todo/log/access.log;
12     error_log /home/lsanchez/public_html/todo/log/error.log;
13
14     root /home/lsanchez/public_html/todo/public/;
15     index index.html;
16
17     location / {
18         proxy_set_header X-Real-IP $remote_addr;
19         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
20         proxy_set_header Host $http_host;
21         proxy_redirect false;
22
23         if (-f $request_filename/index.html) {
24             rewrite (.*) $1/index.html break;
25         }
26
27         if (-f $request_filename.html) {
28             rewrite (.*) $1.html break;
29         }
30
31         if (!-f $request_filename) {
32             proxy_pass http://domain1;
33             break;
34         }
35     }
36 }
```

```
sudo ln -s /usr/local/nginx/sites-available/rubyonrails.slice.com
```

```
/usr/local/nginx/sites-enabled/rubyonrails.slice.com
```

```
sudo /etc/init.d/nginx restart
```

Navigate to rubyonrails.slice.com in your browser. You should see the default Ruby on Rails welcome page. Welcome to the wonderful world of Ruby on Rails!

I'm not a perfect programmer nor do I have the credentials to say that this is the most perfect setup for a web server - but it suits my needs and it's easy for me to setup and understand what's going on. Leave some comments below if you have any suggestions or have a sweet setup of your own.