FreeBSD Administration - Support #622

Setup a Nginx, PostgreSQL, PHP 5.6 Web Server on FreeBSD

06/06/2015 10:44 AM - Daniel Curtis

Status:	Closed	**Start date:**	05/02/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Web Server	**Estimated time:**	3.00 hours
Target version:	FreeBSD 9	**Spent time:**	6.00 hours

Description

Here is a procedure to install a FreeBSD with Nginx, PostgreSQL and PHP server stack. If any version of the packages needs to be changed, replace the versions in the commands accordingly.

There are also extra additions for running Ruby web applications with Phusion Passenger, and Perl CGI web applications using FastCGI.

Prepare the Environment

- Before installation of the components, make sure everything is up to date using the following command:

```
pkg update -f && pkg upgrade
```

- Install portmaster:

```
cd /usr/ports/ports-mgmt/portmaster
make install clean
pkg2ng
```

Install Nginx

- Install Nginx

```
portmaster www/nginx
```

- Start and enable nginx at boot:

```
echo 'nginx_enable="YES"' >> /etc/rc.conf
service nginx start
```

- Create a configuration directory to make managing individual server blocks easier

```
mkdir /usr/local/etc/nginx/conf.d
```

- Edit the main nginx config file:

```
vi /usr/local/etc/nginx/nginx.conf
```

    - And strip down the config file and add the include statement at the end to make it easier to handle various server blocks:

```
    load_module /usr/local/libexec/nginx/ngx_mail_module.so;
    load_module /usr/local/libexec/nginx/ngx_stream_module.so;

    worker_processes  1;
    error_log  /var/log/nginx-error.log;

    events {
        worker_connections  1024;
    }

    http {
        include       mime.types;
        default_type  application/octet-stream;
        sendfile        on;
        keepalive_timeout  65;

        # Load config files from the /etc/nginx/conf.d directory
        include /usr/local/etc/nginx/conf.d/*.conf;
    }
```

Default Static Website

Start by setting up a simple static website, no server-side stuff PHP or Ruby; just plain HTML, CSS, JavaScript, etc.

- Create a directory for the web site:

```
mkdir /usr/local/www/www.example.com
```

- Add a **default site server block**:

```
vi /usr/local/etc/nginx/conf.d/www.example.com.conf
```

  - Add the following:

```
    server {
        listen       80 default_server;
        server_name  www.example.com;

        access_log  /var/log/www.example.com.log  main;

        location / {
            root   /usr/local/www/www.example.com;
            index  index.html index.htm;
        }

        # redirect server error pages to the static page /50x.html
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   /usr/local/www/nginx-dist;
        }
    }
```

Install PostgreSQL 9.4

- Install PostgreSQL:

```
portmaster databases/postgresql94-server
```

- Enable PostgreSQL at boot:

```
echo 'postgresql_enable="YES"' >> /etc/rc.conf
```

- Initialize the database:

```
service postgresql initdb
```

- Start PostgreSQL:

```
service postgresql start
```

- Edit the postgres config file:

```
vi /usr/local/pgsql/data/postgresql.conf
```

  - And modify the following:

```
listen_addresses = '*'
```

- Edit the pg_hba config file:

```
vi /usr/local/etc/pgsql/data/pg_hba.conf
```

  - And modify the following:

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD only

# Local connections
local   all         all                               trust

# IPv4 local connections:
host    all         all         127.0.0.1/32          trust

# IPv6 local connections:
host    all         all         ::1/128               trust

# IPv4 connections:
host    all         all         192.168.10.0/24       md5

# IPv6 local connections:
host    all         all         1234::abcd/64         md5
```

- And wrap up by restarting the nginx and postgresql servers:

```
service nginx restart
service postgresql restart
```

Create a new user and database

- Switch to the pgsql user:

```
su pgsql
```

- Create the somepgdb database:

```
createdb somepgdb
```

- Then create the somepguser user:

```
createuser -P
```

- Grant all privileges to somepgdb to somepguser:

```
psql template1
GRANT ALL PRIVILEGES ON DATABASE "somepgdb" to somepguser;
\q
```

- Exit from the pgsql user

```
exit
```

Install PHP

The PHP support in FreeBSD is extremely modular so the base install is very limited. It is very easy to add support using the *lang/php5-extensions* port. This port provides a menu driven interface to PHP extension installation. Alternatively, individual extensions can be installed using the appropriate port.

- Install PHP 5.6 and other supporting packages:

```
portmaster lang/php56
```

- Install PHP extensions and a few modules:

```
portmaster lang/php56-extensions databases/php56-pgsql databases/php56-pdo_pgsql www/php56-ses
sion
```

- **NOTE**: There are many more PHP modules, to search for more PHP modules run:

```
find /usr/ports/ -name "php56-*"
```

- **NOTE**: PHP capabilities can be further extended by using PECL packages, to search for more PECL packages run:

```
find /usr/ports/ -name "pecl-*"
```

- Configure the default PHP settings

```
cp /usr/local/etc/php.ini-production /usr/local/etc/php.ini
```

Configure PHP-FPM

- Edit /usr/local/etc/php-fpm.conf:

```
vi /usr/local/etc/php-fpm.conf
```

  - Make the following changes:

```
listen = /var/run/php-fpm.sock
listen.owner = www
listen.group = www
listen.mode = 0660
```

- Start and enable PHP-FPM at boot:

```
echo 'php_fpm_enable="YES"' >> /etc/rc.conf
service php-fpm start
```

- Restart nginx:

```
service nginx restart
```

PHP Website

- Create a directory for the web application:

```
mkdir /usr/local/www/phpapp.example.com
```

- Add a **phpapp.example.com server block**:

```
vi /usr/local/etc/nginx/conf.d/phpapp.example.com.conf
```

  - Add the following:

```
server {
    listen       80;
    server_name  phpapp.example.com;
    root         /usr/local/www/phpapp.example.com;
    access_log   /var/log/phpapp.example.com-access.log;
    error_log    /var/log/phpapp.example.com-error.log;

    location / {
        index  index.php index.html index.htm;
    }

    # For all PHP requests, pass them on to PHP-FPM via FastCGI
    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_script_name;
        include fastcgi_params;
    }
}
```

Install Passenger (Ruby)

- Install Passenger

```
portmaster www/rubygem-passenger
```

  - **NOTE**: Make sure to enable **[X]NGINX** while configuring rubygem-passenger
  - **NOTE**: Enabling **[X]SYMLINK** makes upgrading passenger easier later on.
  - **NOTE**: Ruby capabilities can be further extended by using rubygem packages, to search for more packages run:

```
find /usr/ports/ -name "rubygem-*"
```

- Reinstall nginx with passenger support:

```
cd /usr/ports/www/nginx
make config
portmaster www/nginx
```

  - **NOTE**: Make sure to enable **[X]PASSENGER** while configuring *nginx*

Configure Passenger

- Edit the main nginx config file:

```
vi /usr/local/etc/nginx/nginx.conf
```

  - And add the Passenger config parameters:

```
#user  nobody;
worker_processes  1;
error_log  /var/log/nginx-error.log;

events {
    worker_connections  1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

    # Load Phusion Passenger module globally
    passenger_root /usr/local/lib/ruby/gems/2.1/gems/passenger;
    passenger_ruby /usr/local/bin/ruby21;
    passenger_max_pool_size 15;
    passenger_pool_idle_time 300;

    # Load config files from the /etc/nginx/conf.d directory
    include /usr/local/etc/nginx/conf.d/*.conf;
}
```

- Restart nginx to load passenger:

```
service nginx restart
```

Ruby Website

- Create a directory for the web application:

```
mkdir /usr/local/www/rubyapp.example.com
```

- Add a **rubyapp.example.com server block**:

```
vi /usr/local/etc/nginx/conf.d/rubyapp.example.com.conf
```

  - Add the following:

```
server {
    listen       80;
    server_name  rubyapp.example.com;
    root         /usr/local/www/rubyapp.example.com/public;
    access_log   /var/log/rubyapp.example.com-access.log;
    error_log    /var/log/rubyapp.example.com-error.log;

    passenger_enabled on;
    passenger_user    www;
    passenger_group   www;
}
```

- Restart nginx to load the website config:

```
service nginx restart
```

Install FastCGI (Perl)

I occasionally need to run a perl CGI script, FastCGI is capable of handling this task.

- Install fcgiwrap

```
pkg install fcgiwrap p5-FCGI
```

- Start and enable fcgiwrap at boot:

```
echo 'fcgiwrap_enable="YES"' >> /etc/rc.conf
echo 'fcgiwrap_profiles="main"' >> /etc/rc.conf
echo 'fcgiwrap_main_socket="unix:/tmp/fcgiwrap.sock"' >> /etc/rc.conf
echo 'fcgiwrap_main_user="www"' >> /etc/rc.conf
service fcgiwrap start
```

CGI Website

- Create a directory for the web application:

```
mkdir /usr/local/www/cgi-app.example.com
```

- Add a **cgi-app.example.com server block**:

```
vi /usr/local/etc/nginx/conf.d/cgi-app.example.com.conf
```

  - Add the following:

```
upstream fcgiwrap {
   server unix:/tmp/fcgiwrap.sock;
}

server {
    listen       80;
    server_name  cgi-app.example.com;
    root         /usr/local/www/cgi-app.example.com;
    access_log   /var/log/cgi-app.example.com-access.log;
    error_log    /var/log/cgi-app.example.com-error.log;

    location / {
            try_files $uri $uri/ =404;
    }

    location ~ \.cgi$ {
        include /usr/local/etc/nginx/fastcgi_params;
        fastcgi_index index.cgi;
        fastcgi_pass fcgiwrap;
        fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
    }
}
```

- Restart nginx to load the website config:

```
service nginx restart
```

Install uWSGI (Python)

- Install a couple dependencies:

```
pkg install bash  py27-{pip,virtualenv} sqlite3 uwsgi
```

- Create a user for the python app to run a virtual environment under:

```
pw group add pydev
pw user add pydev -m -g pydev -s /usr/local/bin/bash -c "Python Dev"
```

- Switch to the pydev user:

```
su - pydev
```

- Create a virtualenv for python_app:

```
mkdir ~/venv && cd ~/venv
virtualenv python_app
```

- Activate the python_app virtualenv:

```
source ~/venv/python_app/bin/activate
```

- Install your python app, this example just installs django:

```
pip install Django
django-admin.py startproject python_app
cd python_app
```

- Initialize the SQLite database:

```
pip install pysqlite
./manage.py migrate
```

- Create an administrative user for the python_app:
  ./manage.py createsuperuser
- Create a wsgi config file for the python_app:

```
vi ~/python_app/wsgi.ini
```

  - And add the following:

```
[uwsgi]
master          = true
processes       = 10
chdir           = /home/pydev/python_app
module          = python_app.wsgi
home            = /home/pydev/venv/python_app
vacuum          = true
```

- Exit from the pydev user session, back into the root session:

```
exit
```

- Start and enable uwsgi at boot with additional arguments to specify a www profile:

```
echo 'uwsgi_enable="YES"' >> /etc/rc.conf
echo 'uwsgi_profiles="www"' >> /etc/rc.conf
echo 'uwsgi_www_socket="/tmp/www_uwsgi.sock"' >> /etc/rc.conf
echo 'uwsgi_www_uid="1004"' >> /etc/rc.conf
echo 'uwsgi_www_gid="1004"' >> /etc/rc.conf
echo 'uwsgi_www_flags="-M -L --ini /home/pydev/python_app/wsgi.ini"' >> /etc/rc.conf
service uwsgi start
```

Python Website

- Add a **python.example.com server block**:

```
vi /usr/local/etc/nginx/conf.d/python.example.com.conf
```

- Add the following:

```
upstream django {
   server unix:/tmp/www_uwsgi.sock;
}

server {
    listen       80;
    server_name  python.example.com;
    access_log   /var/log/python.example.com-access.log;
    error_log    /var/log/python.example.com-error.log;

    location /media  {
        alias /home/pydev/python_app/media;
    }

    location /static {
        alias /home/pydev/python_app/static;
    }

    location / {
        uwsgi_pass  django;
        include     /usr/local/etc/nginx/uwsgi_params;
    }
}
```

- Restart nginx to apply the configuration:

```
service nginx restart
```

Securing Nginx With SSL

Enabling SSL in Nginx is simple. First add the ssl directive in the server listen option, then add the SSL certificate and key paths.

- Install OpenSSL:

```
pkg install openssl
```

- Setup the Diffie-Hellman Key Exchange Parameters

```
cd /usr/local/etc/nginx
openssl dhparam -out dhparam.pem 4096
```

- Generate a strong SSL key and a CSR to send for signing by a CA:

```
cd /usr/local/etc/nginx
openssl req -sha512 -out www.example.com.csr -new -newkey rsa:4096 -nodes -keyout www.example.com.key
```

  - If the received SSL certificate requires additional bundle certificates, like a CA intermediate bundle, add them together like so:

```
cd /usr/local/etc/nginx
cat www.example.com.crt startcom.class1.bundle > www.example.com.chained.crt
```

- Setup the default site configuration:

```
vi /usr/local/etc/nginx/conf.d/www.example.com.conf
```

  - Then add or modify the configuration to look similar to the following:

```
server {
    listen 80;
    listen 443 default ssl;
    server_name www.example.com;

    # Turn on ans set SSL key/cert
    ssl on;
    ssl_certificate /usr/local/etc/nginx/www.example.com.crt;
    ssl_certificate_key /usr/local/etc/nginx/www.example.com.key;

    # Strong SSL configuration
    ssl_ciphers 'AES128+EECDH:AES128+EDH:!aNULL';
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_session_cache  builtin:1000  shared:SSL:10m;
    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_prefer_server_ciphers on;
    ssl_dhparam /usr/local/etc/nginx/dhparam.pem;
    add_header Strict-Transport-Security max-age=63072000;
    add_header X-Frame-Options SAMEORIGIN;
    add_header X-Content-Type-Options nosniff;

    root /usr/local/www/;
    index index.html index.htm;
    autoindex on;
}
```

- Restart nginx to load the new website config:

```
service nginx restart
```

Resources

- http://www.bsdnow.tv/tutorials/nginx
- http://forums.freebsd.org/viewtopic.php?t=30268
- https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html
- http://uwsgi-docs.readthedocs.io/en/latest/tutorials/Django_and_nginx.html

History

#1 - 06/06/2015 10:44 AM - Daniel Curtis

- Copied from Support #433: Setup a FreeBSD, Nginx, MariaDB 5.5, PHP5 Web Server added

#2 - 06/06/2015 10:44 AM - Daniel Curtis

- Copied from deleted (Support #433: Setup a FreeBSD, Nginx, MariaDB 5.5, PHP5 Web Server)

#3 - 06/06/2015 10:47 AM - Daniel Curtis

- Description updated

#4 - 06/06/2015 11:51 AM - Daniel Curtis

- Subject changed from Setup a FreeBSD, Nginx, PostgreSQL, PHP5 Web Server to Setup a FreeBSD, Nginx, PostgreSQL, PHP 5.6 Web Server

- Description updated

#5 - 06/08/2015 12:26 PM - Daniel Curtis

- Description updated

- % Done changed from 0 to 50

#6 - 06/08/2015 12:29 PM - Daniel Curtis

- Description updated

- Status changed from New to In Progress

- % Done changed from 50 to 80

#7 - 06/09/2015 04:57 PM - Daniel Curtis

- Description updated

- Status changed from In Progress to Resolved

- % Done changed from 80 to 100

#8 - 06/09/2015 04:57 PM - Daniel Curtis

- Description updated

#9 - 06/09/2015 05:01 PM - Daniel Curtis

- Description updated

#10 - 06/10/2015 02:06 PM - Daniel Curtis

- Description updated

#11 - 06/12/2015 10:32 AM - Daniel Curtis

- Status changed from Resolved to Closed

#12 - 06/14/2015 04:30 PM - Daniel Curtis

- Description updated

#13 - 07/29/2015 03:58 PM - Daniel Curtis

- Description updated

#14 - 10/03/2015 12:53 PM - Daniel Curtis

- Description updated

#15 - 10/03/2015 12:55 PM - Daniel Curtis

- Description updated

#16 - 10/03/2015 12:56 PM - Daniel Curtis

- Description updated

#17 - 10/03/2015 12:56 PM - Daniel Curtis

- Description updated

#18 - 10/03/2015 01:29 PM - Daniel Curtis

- Description updated

#19 - 10/03/2015 01:52 PM - Daniel Curtis

- Description updated

#20 - 10/03/2015 02:17 PM - Daniel Curtis

- Description updated

#21 - 10/03/2015 04:14 PM - Daniel Curtis

- Description updated

#22 - 10/03/2015 04:18 PM - Daniel Curtis

- Description updated

#23 - 01/26/2016 06:32 PM - Daniel Curtis

- Description updated

#24 - 03/01/2016 12:04 PM - Daniel Curtis

- Description updated

#25 - 03/01/2016 12:07 PM - Daniel Curtis

- Description updated

#26 - 03/01/2016 02:34 PM - Daniel Curtis

- Description updated

#27 - 03/01/2016 03:03 PM - Daniel Curtis

- Subject changed from Setup a FreeBSD, Nginx, PostgreSQL, PHP 5.6 Web Server to Setup a Nginx, PostgreSQL, PHP 5.6 Web Server on FreeBSD

- Description updated

#28 - 04/10/2016 12:27 PM - Daniel Curtis

- Description updated

#29 - 11/30/2016 12:27 PM - Daniel Curtis

- Description updated

#30 - 11/30/2016 04:00 PM - Daniel Curtis

- Description updated

#31 - 11/30/2016 04:00 PM - Daniel Curtis

- Description updated