GNU/Linux Administration - Support #692

Install an Nginx, PostgreSQL, PHP 5.6 Web Server on Arch

11/06/2015 02:09 PM - Daniel Curtis

Status:	Closed	**Start date:**	11/06/2015
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Web Server	**Estimated time:**	3.00 hours
Target version:	Arch Linux	**Spent time:**	9.00 hours

Description

Here is a procedure to install Nginx, PostgreSQL and PHP web server stack on Arch Linux.

Prepare the Environment

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

```
pacman -Syu
```

- Install a few dependencies:

```
pacman -S sudo git curl ntp
```

- Add the sudo group

```
groupadd sudo
```

- Edit the sudoers file:

```
visudo
```

  - And uncomment the %sudo line to allow users in the sudo group access to sudo:

```
%sudo    ALL=(ALL) ALL
```

- Add a webuser

```
useradd -m -g users -s /bin/bash webuser
```

  - And add webuser to the sudo group:

```
usermod -aG sudo webuser
```

- Start and enable ntpd:

```
systemctl enable ntpd
systemctl start ntpd
```

Install Nginx

- Install Nginx

```
pacman -S nginx
```

- Start and enable nginx at boot:

```
systemctl enable nginx
systemctl start nginx
```

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

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

- Edit the main nginx config file:

```
vi /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:

```
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;

    include /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 -p /var/www/www.example.com
```

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

```
vi /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   /var/www/www.example.com;
        index  index.html index.htm;
    }
}
```

Install PostgreSQL 9.4

- Make sure to enable en_US.UTF-8 in the locale generation file:

```
sed -i -e 's/\#en_US.UTF-8 UTF-8/en_US.UTF-8 UTF-8/' /etc/locale.gen
```

  - And regenerate the local locales:

```
locale-gen
```

- Install PostgreSQL:

```
pacman -S postgresql
```

- Enable PostgreSQL at boot:

```
systemctl enable postgresql
```

  - **NOTE**: If the postgresql database is on a btrfs filesystem, make sure to disable Copy-on-Write for the postgresql database folder:

```
chattr +C /var/lib/postgres/data
```

- Switch to the postgresql user:

```
su - postgres
```

  - Initialize the database:

```
initdb --locale en_US.UTF-8 -E UTF8 -D '/var/lib/postgres/data'
```

  - Then exit the postgresql user:

```
exit
```

- Start PostgreSQL:

```
systemctl start postgresql
```

- Edit the postgres config file:

```
vi /var/lib/postgres/data/postgresql.conf
```

  - And modify the following:

```
listen_addresses = '*'
```

- Edit the pg_hba config file:

```
vi /var/lib/postgres/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 postgresql server:

```
systemctl restart postgresql
```

Create a new user and database

- Switch to the pgsql user:

```
su - postgres
```

  - Create the somepgdb database:

```
createdb somepgdb
```

  - Then create the somepguser user:

```
createuser -P
```

  - Grant all privileges to somepgdb to somepguser:

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

- Exit from the postgresql user

```
exit
```

Install PHP

- Install PHP 5.6 and php-fpm:

```
pacman -S php php-fpm
```

- Install PHP extensions and a few modules:

```
pacman -S php-pgsql php-pear
```

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

```
pacman -Ss php
```

- Enable php-fpm at boot:

```
systemctl enable php-fpm
```

- Edit the php-fpm config:

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

  - Make the following changes:

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

- Start php-fpm:

```
systemctl start php-fpm
```

PHP Website

- Create a directory for the web application:

```
mkdir /var/www/phpapp.example.com
```

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

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

  - Add the following:

```
server {
    listen       80;
    server_name  phpapp.example.com;
    root         /var/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;
    }

    location ~ \.php$ {
        fastcgi_pass unix:/run/php-fpm/php-fpm.sock;
        fastcgi_param SCRIPT_FILENAME /var/www/phpapp.example.com$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_script_name;
        include fastcgi_params; # include extra FCGI params
    }
}
```

- Add the PHP files for the site and change the ownership to the http user:

```
chown R http:http /var/www/phpapp.example.com
```

- Restart nginx:

```
systemctl restart nginx
```

(Extra) Install Phusion Passenger

- Remove the system nginx package:

```
pacman -R nginx
```

  - Remove the old nginx configuration directory

```
rm -rf /etc/nginx
```

- Install the base-devel package:

```
pacman -S base-devel
```

- Import the RVM GPG signing key:

```
gpg2 --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3
```

- Install RVM:

```
curl -L get.rvm.io | bash -s stable
```

- Then add the following line to the end of your ~/.bash_login file

```
echo '[[ -s "$HOME/.rvm/scripts/rvm" ]] && source "$HOME/.rvm/scripts/rvm"' >> ~/.bash_login
```

  - Now logout and log back in for rvm to take effect

- Add the http user to the rvm group:

```
usermod -aG rvm http
```

- See if there are any dependency requirements for your installation by running:

```
rvm requirements
```

- Install ruby 2.1.5:

```
rvm install 2.1.5
```

- Run the following to allow passenger install nginx:

```
rvm 2.1.5
gem install passenger
rvmsudo passenger-install-nginx-module
```

- Link the new nginx configuration folder in the /etc folder:

```
ln -s /opt/nginx/conf /etc/nginx
```

  - Don't forget to remake the conf.d folder in the nginx configuration folder:

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

- Edit the main nginx config file:

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

  - And add the Passenger config parameters:

    ```
    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;

        passenger_root /usr/local/rvm/gems/ruby-2.1.5@global/gems/passenger-5.0.21;
        passenger_ruby /usr/local/rvm/wrappers/ruby-2.1.5/ruby;
    ```

```
    passenger_max_pool_size 15;
    passenger_pool_idle_time 300;

    include /etc/nginx/conf.d/*.conf;
}
```

- Create a new nginx systemd init script:

```
vi /etc/systemd/system/nginx.service
```

  - And add the following:

```
[Unit]
Description=Nginx
After=syslog.target network.target

[Service]
Type=forking
ExecStart=/opt/nginx/sbin/nginx
ExecReload=/opt/nginx/sbin/nginx -s reload

[Install]
WantedBy=multi-user.target
```

- Start and enable nginx at boot::

```
systemctl daemon-reload
systemctl enable nginx
systemctl start nginx
```

Ruby Website

- Create a directory for the web application:

```
mkdir /var/www/rubyapp.example.com
```

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

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

  - Add the following:

```
server {
    listen       80;
    server_name  rubyapp.example.com;
    root         /var/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    http;
    passenger_group   http;
}
```

- Restart nginx to load the website config:

```
service nginx restart
```

Securing Nginx With SSL

- Install OpenSSL:

```
pacman -S openssl
```

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

- The basic SSL server block should be look similar to the following:

```
server {
    listen              443 ssl;
    server_name         www.example.com;
    ssl_certificate     www.example.com.crt;
    ssl_certificate_key www.example.com.key;
}
```

- Setup the Diffie-Hellman Key Exchange Parameters

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

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

```
cd /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, add them together like so:

```
cd /etc/nginx
cat www.example.com.crt www.example.com.bundle > www.example.com.chained.crt
```

- Setup the default site configuration:

```
vi /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;

    ssl on;
    ssl_certificate /etc/nginx/www.example.com.crt;
    ssl_certificate_key /etc/nginx/www.example.com.key;

    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 /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 /var/www/;
        index index.html index.htm;
        autoindex on;
    }
```

- Restart nginx to load the new website config:

```
service nginx restart
```

Certificate Bundles

Some browsers may complain about a certificate signed by a well-known certificate authority, while other browsers may accept the certificate without issues. This occurs because the issuing authority has signed the server certificate using an intermediate certificate that is not present in the certificate base of well-known trusted certificate authorities which is distributed with a particular browser. In this case the authority provides a bundle of chained certificates which should be concatenated to the signed server certificate.

- The server certificate must appear before the chained certificates in the combined file:

```
cat www.example.com.crt bundle.crt > www.example.com.chained.crt
```

- The resulting file should be used in the ssl_certificate directive:

```
server {
    listen              443 ssl;
    server_name         www.example.com;
    ssl_certificate     www.example.com.chained.crt;
    ssl_certificate_key www.example.com.key;
}
```

Resources

- https://wiki.archlinux.org/index.php/Ruby_on_Rails#The_Perfect_Rails_Setup
- https://wiki.archlinux.org/index.php/RVM
- https://wiki.archlinux.org/index.php/PostgreSQL

History

#1 - 11/06/2015 02:12 PM - Daniel Curtis

- Status changed from New to In Progress

- % Done changed from 0 to 30

#2 - 11/06/2015 03:28 PM - Daniel Curtis

- Description updated

- % Done changed from 30 to 60

#3 - 11/06/2015 04:10 PM - Daniel Curtis

- Description updated

- Status changed from In Progress to Resolved

- % Done changed from 60 to 100

#4 - 11/06/2015 04:48 PM - Daniel Curtis

- Description updated

#5 - 11/16/2015 04:38 PM - Daniel Curtis

- Description updated

#6 - 11/17/2015 08:40 AM - Daniel Curtis

- Description updated

#7 - 11/27/2015 03:43 PM - Daniel Curtis

- Status changed from Resolved to Closed