FreeBSD Administration - Support #641

Install CloudLog on FreeBSD

08/15/2015 08:56 PM - Daniel Curtis

Status:	Suspended	**Start date:**	08/15/2015
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	0%
Category:	Web Server	**Estimated time:**	2.00 hours
Target version:	FreeBSD 9	**Spent time:**	0.00 hour

Description

Here is a guide to install CloudLog on FreeBSD with Nginx, MariaDB and PHP server stack.

Pre-installation requirements

- 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:

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

}
```

Install PHP

- Install PHP5 and other supporting packages:

```
portmaster lang/php5
```

- Install PHP extensions and a few modules:

```
portmaster lang/php5-extensions databases/php5-mysql databases/php5-mysqli databases/php5-pdo_
mysql www/php5-session php5-curl
```

- 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:

```
events.mechanism = kqueue
listen = /var/run/php-fpm.sock
listen.owner = www
listen.group = www
listen.mode = 0666
```

- Start and enable PHP-FPM at boot:

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

```
service php-fpm start
```

- Restart nginx:

```
service nginx restart
```

Install MariaDB

- Install MariaDB 5.5 server and client

```
portmaster databases/mariadb55-server databases/mariadb55-client
```

Configure MariaDB server

- Configure the MariaDB server

```
cp /usr/local/share/mysql/my-small.cnf /usr/local/etc/my.cnf
```

- Enable MariaDB to start at boot:

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

- Start MariaDB

```
service mysql-server start
```

- Set password for mysql using the following command

```
mysqladmin -uroot password
```

- Restart mysql using the following commands:

```
service mysql-server restart
```

Install and configure phpMyAdmin

- Install phpmyadmin:

```
pkg install phpmyadmin
```

- Setup phpMyAdmin for nginx by adding the following to the server{ } block in /usr/local/etc/nginx/nginx.conf:

```
## phpMyAdmin
location ^~ /phpmyadmin {
  access_log  off;
  rewrite ^  /phpMyAdmin/ permanent;
```

```
}

location /phpMyAdmin {
  root /usr/local/www/phpMyAdmin;
  index index.php index.html;

  ## Only Allow connections from localhost
  allow 127.0.0.1;
  deny all;

    location ~ ^/phpMyAdmin/(.*\.php)$ {
      root /usr/local/www/phpMyAdmin;
      fastcgi_pass unix:/var/run/php-fpm.sock;
      fastcgi_param SCRIPT_FILENAME /usr/local/www/phpMyAdmin$fastcgi_script_name;
      fastcgi_param PATH_INFO $fastcgi_script_name;
      include fastcgi_params; # include extra FCGI params
    }
}
```

Now its time to configure phpMyAdmin. Do this by creating the file /usr/local/www/phpMyAdmin/config.inc.php, the basic configuration file for phpMyAdmin. Traditionally, users have manually created or modified /usr/local/www/phpMyAdmin/config.inc.php, but now phpMyAdmin includes a nice setup script, making it much easier to create this file with the settings you want.

- Start by creating the directory /usr/local/www/phpMyAdmin/config and make it writable by the phpMyAdmin setup script:

```
mkdir /usr/local/www/phpMyAdmin/config
chmod o+w /usr/local/www/phpMyAdmin/config
```

- Then make /usr/local/www/phpMyAdmin/config.inc.php readable by the phpMyAdmin setup script:

```
chmod o+r /usr/local/www/phpMyAdmin/config.inc.php
```

- Now open your web browser and navigate to http://www.example.com/phpmyadmin/setup where you will see the phpMyAdmin setup *Overview* page.
- Select **New Server** and then select the **Authentication** tab.
  1. Under the **Authentication type** choose http from the drop-down list (using HTTP-Auth to sign-in into phpMyAdmin will avoid storing login/password credentials directly in config.inc.php)
  2. And remove root from the **User for config auth**.
- Now select **Apply** and you will be returned you to the Overview page where you should see a new server listed.
- Select **Save** again in the Overview page to save your configuration as /usr/local/www/phpMyAdmin/config/config.inc.php.
- Now let's move that file up one directory to /usr/local/www/phpMyAdmin where phpMyAdmin can make use of it.

```
mv /usr/local/www/phpMyAdmin/config/config.inc.php /usr/local/www/phpMyAdmin
```

- Now let's try out phpMyAdmin to make sure it works. Point your web browser to http://www.example.com/phpmyadmin where you will be presented with a pop-up box requesting you to log in.
  Use "root" and the MySQL password you set up previously, then you should be directed to the phpMyAdmin administration page.
- We no longer need the /usr/local/www/phpMyAdmin/config directory so let's remove it, and the read permission we added previously to /usr/local/www/phpMyAdmin/config.inc.php:

```
rm -r /usr/local/www/phpMyAdmin/config
chmod o-r /usr/local/www/phpMyAdmin/config.inc.php
```

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

```
service nginx restart
service mysql-server restart
```

Create the User and Database

- Log into the MySQL console:

```
mysql -h localhost -u root -p
```

  - Create the **cloudloguser** user with the **SuperSecretPassword** password and the **cloudlogdb** database:

```
CREATE USER 'cloudloguser'@'localhost' IDENTIFIED BY 'SuperSecretPassword';
CREATE DATABASE IF NOT EXISTS  `cloudlogdb` CHARACTER SET utf8 COLLATE utf8_general_ci;
GRANT ALL PRIVILEGES ON `cloudlogdb`.* TO 'cloudloguser'@'localhost';

flush privileges;
exit
```

Install CloudLog

- Create a directory for the web application:

```
cd /usr/local/www/
git clone https://github.com/magicbug/Cloudlog.git cloudlog.example.com
```

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

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

  - Add the following:

```
server {
  listen       80;
  server_name  cloudlog.example.com;
  root         /usr/local/www/cloudlog.example.com;
  access_log   /var/log/cloudlog.example.com-access.log;
  error_log    /var/log/cloudlog.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 /usr/local/www/phpapp.example.com$fastcgi_script_name;
    fastcgi_param PATH_INFO $fastcgi_script_name;
    include fastcgi_params; # include extra FCGI params
  }

}
```

- Import the CloudLog SQL tables:

```
cd /usr/local/www/cloudlog.example.com/sql/tables
mysql -u cloudloguser -p cloudlogdb < *.sql
```

- Now the install the rest using a web browser by going to http://cloudlog.example.com/install

---

Securing Nginx With SSL

- Install OpenSSL:

```
portmaster security/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         cloudlog.example.com;
        ssl_certificate     cloudlog.example.com.crt;
        ssl_certificate_key cloudlog.example.com.key;
        ...
    }
```

- 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 cloudlog.example.com.csr -new -newkey rsa:4096 -nodes -keyout cloudlo
g.example.com.key
```

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

```
cd /usr/local/etc/nginx
cat cloudlog.example.com.crt cloudlog.example.com.bundle > cloudlog.example.com.chained.cr
t
```

- Setup the default site configuration:

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

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

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

```
  # Turn on ans set SSL key/cert
  ssl on;
  ssl_certificate /usr/local/etc/nginx/cloudlog.example.com.crt;
  ssl_certificate_key /usr/local/etc/nginx/cloudlog.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 DENY;
  add_header X-Content-Type-Options nosniff;

  root /usr/local/www/cloudlog.example.com;
  index index.html index.htm;
  autoindex on;

  # Uncomment to force HTTPS
#  if ($scheme = http) {
#    return 301 https://$server_name$request_uri;
#  }

}
```

Resources

- http://www.bsdnow.tv/tutorials/nginx
- http://forums.freebsd.org/viewtopic.php?t=30268

History

#1 - 04/15/2016 09:11 PM - Daniel Curtis

- Status changed from New to Suspended