Serveur web Apache httpd 2.4, php7.0, MariaDB, et HTTPS par Let's Encrypt

Information

Au 09/07/2017 : Debian 9, Apache 2.4.25, PHP-FPM 7.0.19, MariaDB 10.1.23, OpenSSL 1.1.0f.

Testé sur Debian 9. Cet article implique d'avoir déjà des connaissances sur les éléments utilisés, car tout n'est pas détaillé. Le score sur SSLLABS est A+.

Cette page indique l'installation de fail2ban, voir cette page du wiki pour le configurer.

Installation des packages

Installation de Apache2, PHP7.0-FPM, MariaDB

```
apt install apache2 php7.0-fpm php7.0 php7.0-sqlite php7.0-mysql php7.0-gd
php7.0-curl php7.0-xml php7.0-zip php7.0-mbstring php7.0-apcu mariadb-server
openssl fail2ban
```

Activation de modules apache2

```
a2enmod rewrite    #pour réécriture d'url
a2enmod ssl        #pour SSL
a2enmod headers    #pour entêtes SSL
a2enmod expires    #pour le cache
a2enmod proxy      #pour php7.0-fpm
a2enmod proxy_fcgi #pour php7.0-fpm
```

Configuration de MariaDB

La commande suivante est recommandée en environnement de production. Elle permet de sécuriser l'installation de MariaDB.

```
mysql_secure_installation
```

Configuration php7.0-fpm

Editer le fichier php.ini

```
nano /etc/php/7.0/fpm/php.ini
```

Et ajouter/modifier les infos suivantes :

```
post_max_size = 20M
upload_max_filesize = 20M
max_execution_time = 60
memory_limit = 128M
```

Configurer le cache OpCode pour php7.0-fpm :

```
nano /etc/php/7.0/mods-available/opcache.ini
```

Ajouter/modifier :

```
; configuration for php ZendOpcache module
; priority=05
zend_extension=opcache.so
opcache.memory_consumption = 256
opcache.interned_strings_buffer = 16
opcache.max_accelerated_files = 4000
opcache.revalidate_freq = 10
opcache.fast_shutdown=1
```

Il est possible de surveiller l'utilisation du cache avec opcache.php disponible sur github.

Si les sites hébergés sont à faible ou moyenne fréquentation, on peut optimiser php7.0-fpm en passant le nombre de process géré à la demande.

Editer le fichier :

```
nano /etc/php/7.0/fpm/pool.d/www.conf
```

Ajouter/modifier :

```
pm = ondemand
pm.max_children = 20
pm.process_idle_timeout = 10s
pm.max_requests = 500
```

Redémarrer php7.0-fpm pour charger la nouvelle configuration :

```
service php7.0-fpm restart
```

Il est possible de faire cohabiter php7.0-fpm avec d'autres versions.

Configuration apache2

Désactiver php7.0 pour mpm_prefork :

```
a2dismod php7.0
```

Activer php7.0-fpm :

```
a2enconf php7.0-fpm
```

Attention, cette configuration sera globale à tout vos vhosts !

Alternativement, on peux ajouter php7.0-fpm directement au fichier vhost avec le code suivant :

```
##### PHP7.0-FPM #####

<IfModule !mod_php7.c>
<IfModule proxy_fcgi_module>
    # Enable http authorization headers
    <IfModule setenvif_module>
    SetEnvIfNoCase ^Authorization$ "(.+)" HTTP_AUTHORIZATION=$1
    </IfModule>

    <FilesMatch ".+\.ph(p[3457]?|t|tml)$">
        SetHandler "proxy:unix:/run/php/php7.0-
fpm.sock|fcgi://localhost"
    </FilesMatch>
    <FilesMatch ".+\.phps$">
        # Deny access to raw php sources by default
        # To re-enable it's recommended to enable access to the
files
        # only in specific virtual host or directory
        Require all denied
    </FilesMatch>
    # Deny access to files without filename (e.g. '.php')
    <FilesMatch "^\.ph(p[3457]?|t|tml|ps)$">
        Require all denied
    </FilesMatch>
</IfModule>
</IfModule>
```

On peut alors le mettre dans la section <VirtualHost>, s'appliquant ainsi seulement au domaine désiré. On peut donc faire cohabiter plusieurs versions de php en mode fpm, en fonction des besoins pour chaque domaine.

Passage en mpm_worker pour apache2 :

```
apachectl -V | grep -i mpm
a2dismod mpm_prefork
a2enmod mpm_worker
service apache2 restart
apachectl -V | grep -i mpm
```

Désactiver le fichier de configuration par défaut d'apache :

```
a2dissite 000-default
service apache2 reload
```

Exemple de fichier vhost

Cet exemple de vhost est pour apache 2.4 ! Il contiens une partie HTTP et une partie HTTPS (facultative). N'ajoutez que le nécessaire.

```
############################################################
#     domain.info                                          #
############################################################

##### HTTP

<VirtualHost 192.168.10.10:80>
    ServerName domain.info
    ServerAlias domain.info www.domain.info
    DocumentRoot "/home/www-data/domain.info/www"
    #Redirection automatique vers HTTPS
    #Redirect "/" "https://domain.info/"
    CustomLog /home/www-data/logs/apache2/domain.info-access.log combined
    ErrorLog /home/www-data/logs/apache2/domain.info-error.log
    AccessFileName .htaccess
    <Directory "/home/www-data/domain.info/www">
        DirectoryIndex index.php index.html
        Options FollowSymLinks
        AllowOverride All
        Order Deny,Allow
        Require all granted
    </Directory>
</VirtualHost>
```

On peut changer `Require all granted` par `Require ip x.x.x.x` pour une interface d'administration privée, par exemple. Seule l'IP client indiquée pourra charger les pages, les autres auront une erreur 403.

La ligne commentée `Redirect` forcera l'utilisation de HTTPS. La décommenter quand HTTPS sera configuré et son fonctionnement validé.

Copier ou rendre le vhost disponible dans `/etc/apache2/sites-available/domain.info.conf` :

```
ln -s /home/www-data/domain.info/domain.info.conf /etc/apache2/sites-
```

```
available/domain.info.conf
```

Puis :

```
a2ensite domain.info
```

Le domaine doit être accessible par http. Https n'est pas disponible sans la génération de certificat par Let's Encrypt et CertBot (gratuit) ou un autre fournisseur tier.

Let's Encrypt et CertBot

Commencer par récupérer CertBot dans un dossier.

```
wget https://dl.eff.org/certbot-auto
chmod a+x certbot-auto
./certbot-auto
```

Les dépendances vont s'installer.

Puis générer les certificats, sans auto-configuration d'apache par CertBot (on le fais manuellement dans le vhost).

```
./certbot-auto certonly --webroot -w /home/www-data/domain.info/www -d
domain.info
```

Le paramètre `-w` indique une racine web et `-d` le domaine associé. On peut spécifier plusieurs `-d` après `-w`. On peux recommencer cet enchaînement pour plusieurs racines web, dans la même commande.

Pour renouveler les certificats :

```
./certbot-auto renew
```

Pour automatiser, mettre la commande en tâche cron.

Pensez à modifier le fichier virtualhost pour pointer vers `cert.pem`, `privkey.pem` et `chain.pem` (ou fichiers équivalents).

Modifier le fichier de configuration ssl d'apache :

```
nano /etc/apache2/mods-available/ssl.conf
```

Modifier/ajouter pour obtenir ces paramètres :

```
SSLCipherSuite "AES256+EECDH:AES256+EDH:AES128+EECDH:AES128+EDH"
SSLHonorCipherOrder on
SSLProtocol all -SSLv3 -SSLv2
SSLStrictSNIVHostCheck Off
SSLCompression Off
SSLUseStapling On
SSLStaplingCache "shmcb:${APACHE_RUN_DIR}/stapling_cache(128000)"
```

Ajoutez ensuite les informations au VirtualHost de votre site pour HTTPS :

```
##### HTTPS

<VirtualHost 192.168.10.10:443>
    ServerName domain.info
    ServerAlias domain.info www.domain.info
    DocumentRoot "/home/www-data/domain.info/www"
    #Rewrite pour enlever "www"
    <IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteCond %{HTTP_HOST} !=domain.info [NC]
        RewriteRule ^(.*)$ https://domain.info$1 [L,R=301]
    </IfModule>
    CustomLog /home/www-data/logs/apache2/domain.info-ssl-access.log
combined
    ErrorLog /home/www-data/logs/apache2/domain.info-ssl-error.log
    AccessFileName .htaccess

    <Directory "/home/www-data/domain.info/www">
        DirectoryIndex index.php index.html
        Options FollowSymLinks
        AllowOverride All
        Order Deny,Allow
        Require all granted
    </Directory>
    <IfModule mod_ssl.c>
        SSLEngine on
        SSLCertificateFile "/etc/letsencrypt/live/domain.info/cert.pem"
        SSLCertificateKeyFile
"/etc/letsencrypt/live/domain.info/privkey.pem"
```

```
                SSLCertificateChainFile
"/etc/letsencrypt/live/domain.info//chain.pem"
        Header always set Strict-Transport-Security "max-age=31556926;
preload"
    </IfModule>
</VirtualHost>
```

Appliquez tout vos changements en redémarrant le service apache2 :

```
service apache2 restart
```

Logrotate sur vos logs apache2 custom

Créer le fichier `/etc/logrotate.d/www-data-rotate`

```
/home/www-data/logs/apache2/*.log {
        daily
        missingok
        rotate 30
        notifempty
        create 640 root
        sharedscripts
        postrotate
                if /etc/init.d/apache2 status > /dev/null ; then \
                    /etc/init.d/apache2 reload > /dev/null; \
                fi;
        endscript
        prerotate
                if [ -d /etc/logrotate.d/www-data-prerotate ]; then \
                        run-parts /etc/logrotate.d/www-data-prerotate; \
                fi; \
        endscript
}
```

On peut aussi ajouter les éléments `compress` (compresse les anciens logs) et `delaycompress` (ne compresse pas l'avant dernier fichier de logs). Ne les ajoutez pas si vos logs sont surveillés par fail2ban.

Sources

Source 1 - Source 2 - Source 3 - Source 4 - Source 5 - Source 6 - Source 7 - Source 8 - Source 9 - Source 10 - Source 11 - Source 12 - Source 13 - Source 14 - Source 15