

Trusty, BROUILLON

LLSP : un serveur HTTP Lighty + PHP + SQLite (LLSP)

Un serveur **LLSP** (lighty+sqlite+php) combine sous **L**inux :

- **L**ighttpd ("**lighty**"), beaucoup moins gourmand en ressources que Apache
- **S**QLite, beaucoup moins gourmand en ressources que MySQL.
- **P**HP

Voir aussi <http://ezvz.blogspot.fr/2010/05/lighttpd-how-to-fast-and-secure-web.html>, retranscrit et traduit ici : [Lighttpd "how to" - serveur Web rapide et sécurisé](#)

Pré-requis

Par défaut, la racine du site est située en **/var/www**, emplacement qui n'est accessible qu'au super-utilisateur pour permettre à l'utilisateur de mettre à jour les pages Web sans avoir besoin d'être root.

Nous allons déplacer cette racine vers un disque dur ¹⁾ et lui donner des droits commodes.

La racine sera en **[DISQUE]/srv/www/** et montée sur **/srv/www** via le fichier **fstab**, sans toucher au fichier **lighttpd.conf**.

Ce disque supportera tous les serveurs. Nous allons créer l'arborescence :

DISQUE

```
└─ srv
   └─ ftp
   └─ www — /var/www
```

Pour créer cette arborescence, exécuter les commandes :

```
...@...:~$ sudo mkdir -p [DISQUE]/srv/{ftp,www}
```

Éditez avec les droits d'administration le fichier **/etc/fstab** pour y ajouter la ligne :

[/etc/fstab](#)

```
...
# Mes montages
[DISQUE]/srv /srv none bind 0 0
[DISQUE]/srv/www /var/www none bind 0 0
```

Activez ce montage :

```
...@...:~$ sudo mount /srv
```

La racine du site est désormais accessible de trois façons : **/srv/www**, **/var/www** et **[DISQUE]/srv/www**.

Créez le groupe et l'utilisateur **www-data:www-data** ; ajoutez l'utilisateur **\$USER** au groupe **www-data** :

- ```
sudo addgroup --system www-data
sudo adduser www-data www-data
sudo usermod -a -G www-data $USER
```



N'oubliez pas l'option **-a** ! Sinon, l'utilisateur perdra son appartenance aux groupes dont il faisait partie.

Changez les autorisations sur le répertoire racine :

- ```
sudo chown -R www-data:www-data /srv/www
sudo chmod -R 2770 /srv/www
```



Si **[DISQUE]** est en **ntfs**, il doit être monté avec l'option **permissions** dans **/etc/fstab** pour que les droits soient réglables.

Désormais,



- le répertoire **[DISQUE]/srv/www** est la racine du site.
- l'utilisateur **\$USER** (qui fait partie du groupe **www-data**) a aussi accès à ce répertoire.
- tout nouveau sous-répertoire fait partie du groupe **www-data** (bit setgid=2)

Installation

Installation de Lighttpd

Installez le paquet  **lighttpd-mod-webdav,lighttpd** ou en ligne de commande :

```
$ sudo apt install lighttpd-mod-webdav lighttpd
```

Si vous allez sur l'adresse <http://localhost> (ou <http://192.168.0.31> pour un Raspberry Pi), vous devriez voir la page d'accueil par défaut de lighttpd :

Placeholder page

The owner of this web site has not put up any web pages yet. Please come back later.

You should replace this page with your own web pages as soon as possible.

Unless you changed its configuration, your new server is configured as follows:


- Configuration files can be found in `/etc/lighttpd`. Please read `/etc/lighttpd/conf-available/README` file.
- The DocumentRoot, which is the directory under which all your HTML files should exist, is set to `/var/www`.
- CGI scripts are looked for in `/usr/lib/cgi-bin`, which is where Ubuntu packages will place their scripts. You can enable cgi module by using command "**lighty-enable-mod cgi**".
- Log files are placed in `/var/log/lighttpd`, and will be rotated weekly. The frequency of rotation can be easily changed by editing `/etc/logrotate.d/lighttpd`.
- The default directory index is `index.html`, meaning that requests for a directory `/foo/bar/` will give the contents of the file `/var/www/foo/bar/index.html` if it exists (assuming that `/var/www` is your DocumentRoot).
- You can enable user directories by using command "**lighty-enable-mod userdir**".

About this page

This is a placeholder page installed by the Ubuntu release of the **Lighttpd server package**.

This computer has installed the Ubuntu operating system, but it has nothing to do with the Ubuntu Project. Please do not contact the Ubuntu Project about it.


If you find a bug in this Lighttpd package, or in Lighttpd itself, please file a bug report on it. Instructions on doing this, and the list of known bugs of this package, can be found in the **Ubuntu Bug Tracking System**.



Montez la racine :

- `sudo mount /var/www`

Installation de PHP5

installez les paquets  **php5-cgi,php5-curl,php5-gd,php5-idn,php-pear,php5-imagick,php5-imap,php5-mcrypt,php5-memcache,php5-ming,php5-ps,php5-pspell,php5-recode,php5-snmpphp5-tidy,php5-xmlrpc,php5-xsl,php5-dev**, ou en ligne de commande :

- `sudo aptitude install -y php5-cgi php5-curl php5-gd php5-idn php-pear php5-imagick php5-imap php5-mcrypt php5-memcache php5-ming php5-ps php5-pspell php5-recode php5-snmpphp5-tidy php5-xmlrpc php5-xsl php5-`

dev

Nous en profitons pour installer d'autres modules utiles de PHP5.



Pour lister les modules PHP5 disponibles, tapez ceci:

- `apt-cache search php5`

Installez ceux qui vous intéressent.

Pour activer PHP5 dans Lighttpd, [ouvrez avec les droits d'administration](#) le fichier **/etc/php5/cgi/php.ini** pour le modifier comme ceci :

| action | modifier la ligne | pour obtenir |
|--------------------|--|---|
| dé-commentez | <code>;cgi.fix_pathinfo=1</code> | <code>cgi.fix_pathinfo=1</code> |
| mettez à On | <code>zlib.output_compression = Off</code> | <code>zlib.output_compression = On</code> |

Pour activer PHP via fastcgi, exécutez la commande suivante :

- `sudo lighty-enable-mod fastcgi fastcgi-php`

Redémarrez Lighty

Lancez :

- `sudo service lighttpd restart`

Test

Pour vérifier l'installation de PHP5, [créez](#) le fichier de test **/var/www/test.php** avec le contenu suivant :

[/var/www/test.php](#)


```
<html>
  <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
<head>
  <title>Installation de Lighttpd et de PHP</title>
```

```
</head>
<body>
  <h1>Resultats du test</h1>
  <h2>Lighttpd</h2>
  <p>Lighttpd fonctionne.</p>
  <h2>Dossier courant</h2>
  <p>Dossier courant : <?php echo getcwd();?></p>
  <h2>PHP</h2>
  <p><?php phpinfo();?></p>
</body>
</html>
```

qui teste le fonctionnement de lighty et php et affiche le répertoire en cours.

Pour la lancer : <http://localhost/test.php> ou <http://192.168.0.31/test.php>

Installation de SQLite, du module webdav et de PECL

Pour mettre en place SQLite et webdav, installez les paquets  **php5-sqlite,sqlite3,libsqlite3-dev,lighttpd-mod-webdav,apache2-utils,libpcre3-dev,build-essential** ou en ligne de commande :

- `sudo aptitude install -y php5-sqlite sqlite3 libsqlite3-dev lighttpd-mod-webdav apache2-utils libpcre3-dev build-essential`

Activation de quelques modules




Les modules **compress** et **alias** sont activés par défaut à l'installation de lighty.

Le vérifier en lisant le fichier **/etc/lighttpd/lighttpd.conf** : dans la liste de modules au début, leurs lignes ne sont pas commentées.

Si ce n'est pas le cas, dé-commentez-les.

Pour activer les modules **auth**, **accesslog**, **expire**, **evhost**, **no-www** et **webdav**, lancez les commandes :

- `sudo lighty-enable-mod auth accesslog expire evhost no-www webdav`
`sudo service lighttpd restart`



| | |
|--------------------------------------|--|
| accesslog | active le log des accès dans le fichier /var/log/lighttpd/access.log |
| expire | limite le temps de conservation des fichiers statiques (c'est-à dire sans php) par le navigateur de l'utilisateur. Cela allège le travail du Raspberry Pi en diminuant les demandes et le transfert de données. La date de validité du fichier fait que le navigateur ne demande plus un fichier qu'il a déjà. |
| evhost | active les hôtes virtuels |
| no-www | redirige www.xxx.tld → xxx.tld |
| webdav, auth | serveur webdav |
| sudo service lighttpd restart | relance lighty |

Redémarrez Lighty

Installation de PHP APC

Lancez la commande :

- `sudo pecl install apc`

Pour désinstaller :



- `sudo pecl uninstall apc`

L'installation de APC prend un peu de temps. Répondre aux questions :

| | |
|------------------------------------|--|
| questions concernant Apache | répondre no |
| autres questions | accepter la réponse par défaut (touche ↵ Entrée) |

Créez avec les droits d'administration le fichier **/etc/php5/cgi/conf.d/apc.ini** pour y écrire les lignes suivantes :

/etc/php5/cgi/conf.d/apc.ini

```
extension=apc.so
apc.enabled=1
```

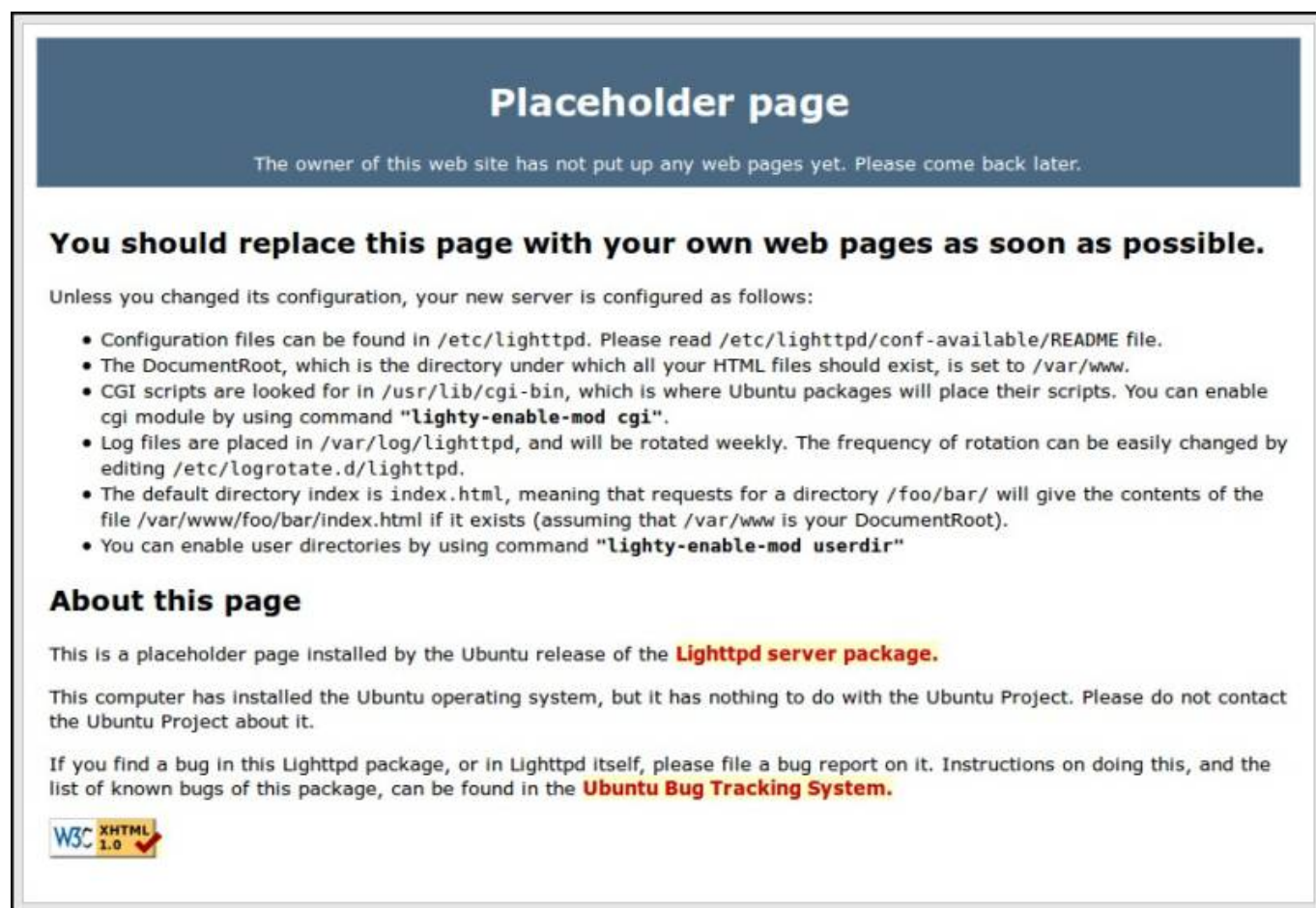
```
apc.shm_size=30
```

Redémarrez Lighty

Vérification

- Sur un PC en local : Ouvrir dans un navigateur l'adresse <http://localhost>
- ou <http://server.exemple.com> sur internet
- ou, sur un PC du réseau, ouvrir l'adresse (ex. d'un Raspberry Pi) <http://framboise.local> (exemple de nom réseau du Raspberry Pi en utilisant avahi).

La page d'accueil par défaut de lighttpd s'affiche :



En lançant <http://localhost/test.php> ou <http://framboise.local/test.php>, notre page de test s'affiche, qui teste le fonctionnement de lighty et php et affiche le répertoire en cours.

Configuration



- Pour retrouver le fichier lighttpd.conf d'origine :
 - télécharger le paquet deb sur le site <https://packages.debian.org/fr/sid/lighttpd#Télec>

harger `lighttpd`

- l'ouvrir avec le gestionnaire d'archives
- et l'extraire.

Pour éviter de modifier le fichier de configuration **`/etc/lighttpd/lighttpd.conf`** livré avec l'application, nous utiliserons un fichier de configuration **`/etc/lighttpd/conf-available/40-config.conf`** que nous activerons.

Ainsi, les réglages ne seront pas affectés par les mises à jour et les migrations seront simplifiées (il suffira de récupérer le fichier de configuration).

sur le même principe, il est possible de créer et d'activer des fichiers **`/etc/lighttpd/conf-available/50-xxx.conf`** pour des configurations particulières (par exemple **`/etc/lighttpd/conf-available/50-dokuwiki.conf`** pour configurer l'accès à dokuwiki)

Configuration de base : fichier **`/etc/lighttpd/conf-available/40-config.conf`**



- Dans notre fichier **`40-config.conf`** :
 - nous ne pouvons pas modifier une variable globale (comme **`server.document-root`**) : on aurait une erreur de double définition.
 - mais dans le cas d'une liste, nous pouvons y ajouter ce que nous voulons par **`+=`** (exemple ici pour **`compress.filetype`**), mais pas la redéfinir.



Pour les directives de configuration, voir cette page de la documentation officielle :

<http://redmine.lighttpd.net/projects/lighttpd/wiki/Docs>

Créez avec les droits d'administration le fichier **/etc/lighttpd/conf-available/40-config.conf** pour y placer les réglages voulus, par exemple comme ceci :

[/etc/lighttpd/conf-available/40-config.conf](#)



```
#
Quelques
définitio
ns de
variables
pour se
faciliter
la vie.
var.based
ir =
server.do
cument-
root
var.conf
d
ir =
"/etc/lig
httpd"
var.logro
ot =
"/var/log
/lighttpd
"
var.dokud
ir =
"doc"

etag.use-
inode =
"enable"
etag.use-
mtime =
"enable"
etag.use-
size =
"enable"
static-
file.etag
```



```
S =  
"enable"  
  
# Pour  
éviter  
les  
erreurs  
417  
server.re  
ject-  
expect-10  
0-  
with-417  
=  
"disable"  
  
#  
sécurisat  
ion  
#  
interdict  
ion  
d'accès  
aux  
fichiers  
.htaccess  
$HTTP["ur  
l"] =~  
"/(\\.|_)h  
t" {  
url.acces  
s-deny =  
( "" ) }  
  
#  
Sécurisat  
ion de  
dokuwiki  
#  
domaine  
doc.* :  
$HTTP["ho  
st"] =~  
"(^" +  
dokudir +  
"\\.)" {  
$HTTP["ur  
l"] =~  
"^/(bin|d  
ata|inc|c  
onf)/" {
```



```
url.access-
deny =
( "" )
}

#
domaine
*/doc :
$HTTP["ur
l"] =~
"^/" +
dokudir +
"/(data|c
onf|bin|i
nc)/+.*"
{
url.access-
deny =
( "" )
}

# Format
du log :
accesslog
.format =
"%h %V %u
%t \"%r\"
%>s %b
\"%{Refer
er}i\"
\"%{User-
Agent}i\"
"

# module
expire :
date
limite 7
jours
$HTTP["ur
l"] =~
"\.(jpg|g
if|png|cs
s|js|svg)
$" {
expire.ur
l = ( ""
=>
"access 7
days" )
```



```
}  
  
# module  
compress  
compress.  
filetype  
+=  
("text/xml", "application/x-javascript", ,  
"application/javascript",  
"text/javascript",  
"text/x-js",  
"text/css",  
,"text/html",  
"text/plain",  
"image/png",  
"image/gif",  
"image/jpeg",  
"image/svg+xml",  
"application/xml")  
  
# module  
awstats  
alias.url  
+= (  
"/icon/"  
=>  
"/var/www/  
awstats/  
wwwroot/i  
con/" )  
alias.url  
+= (  
"/stat"  
=>  
"/var/www
```



```
/awstats/  
wwwroot/c  
gi-bin/"  
)  
$HTTP["ur  
l"] =~  
"^/stat($  
|/)" {  
    # Pour  
que  
http://..  
./stat/  
renvoie  
sur  
http://..  
./stats/a  
wstats.pl  
    index-  
file.name  
s =  
("awstats  
.pl")  
    # perl  
et cgi  
cgi.assig  
n = (  
    ".pl"  
=>  
"/usr/bin  
/perl",  
".cgi" =>  
"/usr/bin  
/perl"  
    )  
}  
  
# module  
WebDAV  
alias.url  
+= (  
"/webdav"  
=>  
"/var/www  
" )
```



| | |
|---------------|--|
| sécurisation | on interdit l'accès aux fichiers .htaccess |
| Format du log | format du fichier access.log |
| module expire | paramétrage de ce module |

module
compress

on ajoute les
filetypes
voulus (avec
+=)

Pour le module **alias**, ouvrez avec les droits d'administration le fichier **/etc/lighttpd/lighttpd.conf** et vérifiez, dans la liste de modules au début, que `mod_alias` est décommenté. Si ce n'est pas le cas, ajoutez à **/etc/lighttpd/conf-available/40-config.conf** la ligne suivante :



`/etc/lighttpd/conf-available/40-config.conf`

```
server.modules +=  
("mod_alias")
```

Activez cette configuration :

```
$ sudo lighty-enable-mod config
```

Pour en savoir plus sur le format du fichier de log `access.log`, voir [Access.log](#) : fichier journal de Lighty



Module accesslog : journaux d'accès

Chaque semaine, le fichier en cours sera automatiquement sauvegardé dans un fichier "access.log.2.gz" et un nouveau fichier "access.log" sera démarré.

Format par défaut de lighty :



```
accesslog
.format =
"%h %V %u
 %t \"%r\"
 %>s %b
 \"%{Refer
er}i\"
 \"%{User-
Agent}i\"
"
```

soit :

%h → adresse
du visiteur

%V → adresse
du serveur
demandé

%u →
utilisateur

%t →
horodatage

\"%r\" →
"requête"

%>s → code


```
192.168.0
.1
doc.framb
oise -
[14/Sep/2
014:15:32
:21
+0200]
"GET /
HTTP/1.1"
200 1558
"_"

"Mozilla/
5.0 (X11;
Ubuntu;
Linux
x86_64;
rv:32.0)
Gecko/201
00101
Firefox/3
2.0"
```

| | | |
|-----------------|---|---|
| %h | 192.168.0.1 | adresse du visiteur |
| %V | doc.framboise | adresse du serveur demandé |
| %a | | utilisateur |
| %t | [14/Sep/2014:13:21:21 +0200] | (ici un bref exemple) horodatage |
| %r | "GET / HTTP/1.1" | requête : méthode GET URL protocole HTTP/1.1 |
| %s | code retour | code statut par 2 : succès |
| %S | 200 | code statut par 3 : réussite |
| %b | 1558 | octets envoyés par 4 : taille |
| %(Referer) % | - | origine |
| %(User-Agent) % | "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; Ubuntu 12.04 LTS; Glibc/2.15; 201303030001 ;)" | type de navigateur |



On peut
ajouter :

Champ **X-Forwarded-For**
(du HTTP-header
field (%))

Pour info, le
format
standard de
Apache est :

```
"%h %l %u  
%t \"%r\"  
%>s %b"
```

donnant :

```
"127.0.0.  
1 - frank  
[10/0ct/2  
000:13:55  
:36  
-0700]  
"GET  
/apache_p  
b.gif  
HTTP/1.0"  
200 2326"
```

avec le sens
suivant :

| élément | valeur | signification |
|---------|-------------------------------|--|
| %h | 127.0.0.1 | adresse IP de l'hôte distant qui a fait la demande au serveur. Ce n'est pas forcément l'adresse IP de l'utilisateur, ce peut être celle d'un proxy. |
| %l | - | Un tiret dans la sortie indique que l'information n'est pas disponible. |
| %u | frank | ID utilisateur (au moment de l'authentification HTTP) de la personne qui demande le document. |
| %t | [10/0ct/2000:13:55:36-0700] | Horodatage de réception de la demande. Format : [jjmm/aaaa:hh:mm:ss zone]. |
| %"%r" | "GET /apache_pb.gif HTTP/1.0" | ligne de demande du client : - méthode : GET - ressource demandée : /apache_pb.gif - protocole : HTTP/1.0 |
| %>s | 200 | code d'état que le serveur envoie au client : - code commençant par 2 : succès - code commençant par 3 : redirection - code commençant par 4 : erreur |
| %b | 2326 | taille de l'objet renvoyé au client, sans compter les en-têtes de réponse. Un tiret si rien n'a été retourné. |

Awstats : surveillance du serveur

Je recommande d'utiliser **awstats** directement sur le serveur, sans installer le paquet, voir la page [Awstats : Surveillance d'un serveur HTTP](#).

- [Awstats : Surveillance d'un serveur HTTP](#)

Module WebDAV

Nous l'avons déjà activé.

Pour en savoir plus, voir la page [Configurer WebDAV Avec Lighttpd](#)



Sous-domaines

D'après <http://wiki.lavilotte-rolle.fr/doku.php?id=debian:sous-domaines>

Pour ajouter un sous domaine **wiki.mondomaine.fr** dont la racine est **/var/wiki** :

[ouvrez avec les droits d'administration](#) le fichier **/etc/lighttpd/lighttpd.conf** et ajoutez la section suivante :

```
$HTTP["host"] ==  
"wiki.mondomaine.fr"  
{  
    server.document-  
root =  
"/var/wiki/wikiroll
```

```
e"
accesslog.filename
=
"/var/log/lighttpd/
wiki.mondomaine.fr.
log"
}
```

Relancez lighttpd :

- #
/etc/init.d/li
ghttpd force-
reload

Maintenant, configurez les dns pour les machines sur le réseau local.

ouvrez avec les droits d'administration le fichier **/etc/hosts** sur les machines concernées et ajoutez la ligne suivante (en remplaçant 192.168.1.101 par l'ip de votre serveur web sur le réseau local) :

```
192.168.1.101
wiki.mondomaine.fr
```

Si vous avez plusieurs domaines pour la même machine il est possible de tous les mettre sur la même ligne :

```
192.168.1.101
www.mondomaine.fr
mondomaine.fr
wiki.mondomaine.fr
```

Pour vérifier que tout fonctionne, ouvrez l'url suivante dans un navigateur :
<http://wiki.mondomaine.fr>

Pour ouvrir le sous-domaine au monde, éditez le fichier de zones correspondant au domaine chez le fournisseur de nom de domaine :

Le fichier doit ressembler à ceci :

```
mon_serveur 10800  
IN A 87.89.21.88
```

Ajoutez la ligne suivante à la fin du fichier :

```
wiki 10800 IN CNAME  
mon_serveur.mondoma  
ine.fr
```

.

Hôtes virtuels (vhost)



Hôtes virtuels simples

Soit à héberger le domaine **domaine.tld**.

Créez l'arborescence pour les fichiers de ce domaine, par exemple **/var/www/domaine.tld** :

- ```
mkdir -p
/var/www/doma
ine.tld
```

Créez avec les droits d'administration un fichier **/etc/lighttpd/conf-available/90-vhost-domaine.tld.conf** contenant le code suivant :

[/etc/lighttpd/conf-](#)

[available/90-vhost-domaine.tld.conf](#)

```
$HTTP["host"] =~
"^(www\\.?)
?domaine\\
.tld$" {
 server.document-
root =
 "/var/www/
domaine.
tld/"
}
```

→ Les URLs de type  
<http://domaine.tld> et  
<http://www.domaine.tld>  
sont dirigées vers le  
répertoire  
**/var/www/domaine.tld**

Activez le domaine en  
lançant :

- ```
sudo lighty-  
enable-mod  
vhost-  
domaine.tld  
sudo service  
lighttpd  
force-reload
```

Pour accéder à ce domaine
via l'adresse locale du
serveur (127.0.0.1 ou
localhost) :

De manière semblable,
[créez avec les droits
d'administration](#) un fichier
**/etc/lighttpd/conf-
available/90-vhost-
domaine.tld** contenant le
code suivant :

[/etc/lighttpd/conf-
available/90-vhost-](#)

domaine.tld

```
$HTTP["host"] =~  
"^xxx\\.yy  
y\\.zzz\\.t  
tt$" {  
  server.docu-  
ment-  
root =  
  "/var/www  
/domaine.  
tld/"  
}
```

```
$HTTP["host"] =~  
"^domaine  
.tld$" {  
  server.docu-  
ment-  
root =  
  "/var/www  
/domaine.  
tld/"  
}
```



Activez ce domaine par :

- ```
sudo lighty-
enable-mod
vhost-
localhost\
&& sudo
service
lighttpd
force-reload
```

### Sous-domaines automatisés

Soit à héberger le domaine  
**domaine.tld** et tous ses  
sous-domaines  
**\*.domaine.tld =**  
**doc.domaine.tld,**  
**toto.domaine.tld,** etc.



Chacun renverra au sous-répertoire correspondant (doc, toto, etc.) de la racine du serveur → Les URLs de type <http://xxx.domaine.tld> sont dirigées vers le répertoire **/var/www/domaine.tld/xx**



Le module **evhost** construit la racine de document selon un modèle qui contient des jokers.

Ces jokers peuvent représenter les parties du **hostname** soumis dans l'appel de la page HTTP, numérotées à partir de la fin :



| paramètre | signification                   | exemple 1                   |
|-----------|---------------------------------|-----------------------------|
| %         | le symbole %                    | "%"                         |
| %0        | nom de domaine + pd             | "ramboise.home"             |
| %1        | pd                              | "home"                      |
| %2        | nom de domaine sans "ramboise"  | "ramboise"                  |
| %3        | nom de sous-domaine de niveau 1 | "111"                       |
| %4        | nom de sous-domaine de niveau 2 | "222"                       |
| %         | nom de domaine complet          | "333.222.111.ramboise.home" |

```
evhost.path-pattern =
"/home/www/servers/
/%3/pages/
/"
```

evhost.path-pattern  
modèle avec des jokers



pour  
construire  
une  
documentroot

Créez avec les droits  
d'administration le fichier  
**/etc/lighttpd/conf-  
available/90-vhost-  
domaine.tld.conf** avec le  
contenu suivant :

[90-vhost-domaine.tld.conf](#)



```
#define a
pattern
for the
host url
finding
ex :
http://xx
x.yyy.zzz
.frambois
e.tld
%% =>
% sign ()
%0 =>
nom de
domaine +
tld
=>
framboise
.home
%1 =>
tld
=> home
%2 =>
nom de
domaine
sans tld
=>
framboise
%3 =>
nom de
sous-
domaine
de niveau
1 =>
zzz
%4 =>
```



```
nom de
sous-
domaine
de niveau
2 =>
yyy
%5 =>
nom de
sous-
domaine
de niveau
3 =>
xxx
etc.
(numérotation de
droite à gauche)

#
Configura
tion pour
le
domaine
framboise
.local
(défini
avec
avahi sur
le réseau
local)
$HTTP["ho
st"] =~
"^(^|\.)fr
amboise\
.local$" {
include
"include/
config.co
nf"
evhost.pa
th-
pattern =
document-
root +
"/%3"
}

#
Configura
tion pour
```



```
le
domaine
domaine.t
ld :
$HTTP["ho
st"] =~
"^(|\.)do
maine\.tl
d$" {
include
"include/
config.co
nf"
evhost.pa
th-
pattern =
document-
root +
"/%3"
}


#
Configura
tion pour
le
domaine
localhost
:
$HTTP["ho
st"] =~
"^(|\.)lo
calhost$"
{
include
"include/
config.co
nf"
evhost.pa
th-
pattern =
document-
root +
"/%2"
}
```

Il suffit de créer un nouveau sous-répertoire **xxx** de la racine du serveur pour que les domaines

**xxx.mondomaine.com**,  
**xxx.framboise.local** et  
**xxx.localhost** existent et  
fonctionnent  
immédiatement.

Exemple pour que le  
domaine  
**www.domaine.tld** soit  
équivalent au domaine  
**domaine.tld**

90-vhost-domaine.tld.conf



```
$HTTP["ho
st"] =~
"^(www\\.)
?domaine\
.tld$" {
 server.do
cument-
root =
document-
root +
"/domaine
.tld"
}
```

Activer le domaine en  
lançant :

- ```
sudo lighty-  
enable-mod  
vhost-  
domaine.tld  
sudo service  
lighttpd  
force-reload
```

**Simplifier les
fichiers de
configuration avec
des includes**

Traduction de
[http://blog.lighttpd.net/arti
cles/2005/11/25/simplify-
your-configfiles-with-](http://blog.lighttpd.net/articles/2005/11/25/simplify-your-configfiles-with-)

[includes/](#)

Ou, autrement dit :
"hébergement virtuel
facile".

Si la configuration de vos
vhosts est complexe,
comme avoir différentes
options (static only,
support php, rails
applications pré-installées,
...) mais une configuration
similaire pour chacun,
vous ne pouvez pas utiliser
les modules vhost et tout
écrire à la main. Mais les
includes et les variables
peuvent aider.

L'idée est de modulariser
le fichier de configuration
et de n'y mettre que les
parties de la configuration
dont le vhost a besoin.



Mais d'abord, définissons
notre configuration :

Tous les vhosts sont sous
/var/www/servers/pages/
certains vhosts ont des
dossiers protégés
certains ont le support de
PHP, certains utilisent des
applications pré-installées

La méthode classique est :

```
$HTTP["host"] ==  
"www.example.org" {  
    server.document-  
    root =  
    "/var/www/servers/w  
    ww.example.org/page  
    s/"  
    auth.backend =  
    "htpasswd"  
    auth.backend.htpass  
    wd.userfile =  
    "/var/www/servers/w
```

```
www.example.org/httpa
sswd"
    auth.require =
...
}
```

Nous spécifions deux fois
le chemin complet.
Améliorons d'abord cela :

```
$HTTP["host"] ==
"www.example.org" {
    var.basedir =
"/var/www/servers/w
ww.example.org/"
server.document-
root = basedir +
"pages/"
    auth.backend =
"httpasswd"
auth.backend.htpass
wd.userfile =
basedir +
"/httpasswd"
    auth.require =
...
}
```



Mais ce n'est seulement
qu'un hôte, ajoutons-en un
autre, qui sert des fichiers
statiques :

```
$HTTP["host"] ==
"www.example.com" {
    var.basedir =
"/var/www/servers/w
ww.example.com/"
server.document-
root = basedir +
"pages/"
}
```

Tous les deux ont le même
répertoire racine,
/var/www/servers/.
Sortons-le :

```
var.basedir =
"/var/www/servers/"
$HTTP["host"] ==
```



```
"www.example.org" {
    var.servername
= "www.example.org"
server.document-
root = basedir +
servername +
"/pages/"
    auth.backend =
"httpasswd"
auth.backend.htpass
wd.userfile =
basedir +
servername +
"/htpasswd"
    auth.require =
...
}

$HTTP["host"] ==
"www.example.com" {
    var.servername
= "www.example.com"

server.document-
root = basedir +
servername +
"/pages/"
}
```



Le réglage

server.document-root

est maintenant

exactement le même pour

les deux serveurs. Sortons-

le de nouveau, cette fois

dans un fichier include

nommé **incl-docroot.conf**

:

[incl-docroot.conf](#)

```
## set
the
docroot
based on
basedir
and
servernam
e
## both
have to
```

```
be
defined
before
server.docu-
ment-root =
basedir +
servername +
"/pages/"
```

et voici notre nouveau
fichier de configuration :

```
var.basedir =
"/var/www/servers/"
$HTTP["host"] ==
"www.example.org" {
    var.servername
    = "www.example.org"

    include "incl-
docroot.conf"

    auth.backend =
    "htpasswd"
    auth.backend.htpass-
    wd.userfile =
    basedir +
    servername +
    "/htpasswd"
    auth.require =
    ( "/download" =>
    ... )
}

$HTTP["host"] ==
"www.example.com" {
    var.servername
    = "www.example.com"

    include "incl-
docroot.conf"
}
```

La dernière étape déplacer
la partie auth dans un
fichier include nommé
incl-auth-htpasswd.conf
:

incl-auth-htpasswd.conf

```
## set
authentif
icate for
a
directory
auth.back
end =
"htpasswd
"
auth.back
end.htpas
swd.userf
ile =
basedir +
servername
+
"/htpasswd"
auth.requ
ire = (
authdir
=> ... )
```



et notre fichier de configuration :

```
var.basedir =
"/var/www/servers/"
$HTTP["host"] ==
"www.example.org" {
    var.servername
= "www.example.org"
    var.authdir =
"/download/"

    include "incl-
docroot.conf"
    include "incl-
auth-htpasswd.conf"
}

$HTTP["host"] ==
"www.example.com" {
    var.servername
= "www.example.com"

    include "incl-
docroot.conf"
```

```
}
```

Ok, dernière étape :
FastCGI pour un hôte.
Nous créons un fichier
include dès le début
appelé **incl-fastcgi-
php.conf** :

```
fastcgi.server = (  
  ".php" => ((  
    "bin-path" =>  
      "/usr/bin/php-cgi",  
    "socket" =>  
      basedir +  
      servername +  
      "/tmp/php-" + PID +  
      ".socket"  
  )))
```

Si un hôte veut le support
PHP, nous incluons
simplement ce fichier :



```
var.basedir =  
"/var/www/servers/"  
$HTTP["host"] ==  
"www.example.org" {  
  var.servername  
= "www.example.org"  
  var.authdir =  
"/download/"  
  
  include "incl-  
docroot.conf"  
  include "incl-  
auth-htpasswd.conf"  
}  
  
$HTTP["host"] ==  
"www.example.com" {  
  var.servername  
= "www.example.com"  
  
  include "incl-  
docroot.conf"  
  include "incl-  
fastcgi-php.conf"  
}
```

Ce qui nous amène à la

dernière question :
plusieurs noms pour le
même vhost. Disons que
www.example.org et
example.org sont le
même vhost sous
différents noms.

```
var.basedir =  
"/var/www/servers/"  
$HTTP["host"] =~  
"^(www\.)?example\  
org$" {  
    var.servername  
= "www.example.org"  
    var.authdir =  
"/download/"  
  
    include "incl-  
docroot.conf"  
    include "incl-  
auth-htpasswd.conf"  
}  
$HTTP["host"] ==  
"www.example.com" {  
    var.servername  
= "www.example.com"  
  
    include "incl-  
docroot.conf"  
    include "incl-  
fastcgi-php.conf"  
}
```



Sous-domaines automatisés

Soit à héberger le domaine
domaine.tld et tous ses
sous-domaines
***.domaine.tld =**
doc.domaine.tld,
toto.domaine.tld, etc.

Chacun renverra au sous-
répertoire correspondant
(doc, toto, etc.) de la
racine du serveur → Les
URLs de type
<http://xxx.domaine.tld> sont

dirigées vers le répertoire
/var/www/domaine.tld/x
xx



Le module **evhost** construit la racine de document selon un modèle qui contient des jokers.

Ces jokers peuvent représenter les parties du **hostname** soumis dans l'appel de la page HTTP, numérotées à partir de la fin :

| paramètre | signification | exemple : |
|-----------|-------------------------------------|------------------------------|
| % | le symbole % | "%" |
| %0 | nom de domaine + tld | "ramboisie.home" |
| %1 | td | "home" |
| %2 | nom de domaine sans "ramboisie" tld | |
| %3 | nom de sous-domaine de niveau 1 | "111" |
| %4 | nom de sous-domaine de niveau 2 | "222" |
| % | nom de domaine complet | "333.222.111.ramboisie.home" |

```
evhost.path-pattern =  
"/home/www/servers/  
%3/pages/  
/"
```

evhost.path-pattern
modèle avec des jokers
pour construire une
documentroot

Créez avec les droits d'administration le fichier **/etc/lighttpd/conf-available/90-vhost-domaine.tld.conf** avec le contenu suivant :

90-vhost-domaine.tld.conf



```
#define a
pattern
for the
host url
finding
# ex :
http://xx
x.yyy.zzz
.frambois
e.tld
#   %% =>
% sign ( )
#   %0 =>
nom de
domaine +
tld
=>
framboise
.home
#   %1 =>
tld
=> home
#   %2 =>
nom de
domaine
sans tld
=>
framboise
#   %3 =>
nom de
sous-
domaine
de niveau
1      =>
zzz
#   %4 =>
nom de
sous-
domaine
de niveau
2      =>
yyy
```



```
# %5 =>
nom de
sous-
domaine
de niveau
3 =>
xxx
# etc.
(numérotation de
droite à gauche)

#
Configura
tion pour
le
domaine
framboise
.local
(défini
avec
avahi sur
le réseau
local)
$HTTP["ho
st"] =~
"(\|\. )fr
amboise\
.local$" {
include
"include/
config.co
nf"
evhost.pa
th-
pattern =
document-
root +
"/%3"
}

#
Configura
tion pour
le
domaine
domaine.t
ld :
$HTTP["ho
st"] =~
```





```
"(^|\.)do
maine\.tl
d$" {
include
"include/
config.co
nf"
evhost.pa
th-
pattern =
document-
root +
"/%3"
}

#
Configura
tion pour
le
domaine
localhost
:
$HTTP["ho
st"] =~
"(^|\.)lo
calhost$"
{
include
"include/
config.co
nf"
evhost.pa
th-
pattern =
document-
root +
"/%2"
}
```

Il suffit de créer un nouveau sous-répertoire **xxx** de la racine du serveur pour que les domaines **xxx.mondomaine.com**, **xxx.framboise.local** et **xxx.localhost** existent et fonctionnent immédiatement.

Exemple pour que le domaine **www.domaine.tld** soit équivalent au domaine **domaine.tld**

90-vhost-domaine.tld.conf

```
$HTTP["host"] =~  
"^(www\\.?)  
?domaine\\  
.tld$" {  
    server.document-  
root =  
document-  
root +  
"/domaine  
.tld"  
}
```




Activer le domaine en lançant :

- ```
sudo lighty-
enable-mod
vhost-
domaine.tld
sudo service
lighttpd
force-reload
```

## Comment configurer lightty pour le https

### Configuration préalable

Installez le paquet  **openssl** ou en ligne de commande :

- ```
sudo apt-get  
install  
openssl
```

Vérifiez que lighttpd
supporte le ssl :

- `lighttpd -v`

```
lighttpd-1.4.19
(ssl) - a light and
fast webserver
Build-Date: Jul 29
2008 18:58:09
```

Si ce n'est pas le cas, il
faut recompiler lighttpd
avec le support ssl.

Créez un certificat ssl
comme indiqué ici :
<http://dev.petitchevalroux.net/linux/generation-certificat-ssl-pour-https-linux.229.html>



Configuration du https pour lighttpd

Créez un nouveau fichier
de configuration : [Créez
avec les droits
d'administration](#) un fichier
**/etc/lighttpd/conf-
available/90-ssl.conf** et
y écrire ce qui suit.

Deux choix possibles :

Activer le ssl pour tous les domaines

Pour activer le ssl pour
tous les domaines
hébergés sur la machine,
écrire ceci dans le fichier
**/etc/lighttpd/conf-
available/90-ssl.conf** :

[/etc/lighttpd/conf-
available/90-ssl.conf](#)

```
$SERVER["socket"] == ":443" {  
    ssl.engine =  
    "enable"  
    ssl.pemfile =  
    "/chemin/vers/certificat/serveur.pem"  
}
```

Activer le ssl pour un seul domaine

Pour activer le ssl sur un seul domaine (ici, *ssl.petitchevalroux.net*), commencez par rediriger toutes les requêtes http vers la version https.


Pour cela, ajoutez dans le fichier `ssl.conf` la règle de redirection suivante :

[ssl.conf](#)

```
...  
#  
Redirection des  
requêtes  
non https  
pour les  
domaines  
https  
$SERVER["socket"] == ":80" {  
    $HTTP["host"] ==  
    "ssl.petitchevalroux.net" {
```

```
url.redirect = (  
    ".*" =>  
    "https://  
ssl.petitchevalrou  
x.net$0"  
)  
    }  
}
```

Puis configurez le port
https de lightty :



```
# Configuration du  
ssl  
$SERVER["socket"]  
== ":443" {  
    ssl.engine =  
    "enable"  
    ssl.pemfile =  
    "/chemin/vers/certi  
ficat/ssl/server.pe  
m"  
    # On met le  
document root à  
/dev/null par  
défaut pour que les  
domaines non ssl  
répondent en 404  
server.document-  
root = "/dev/null"  
    # On surcharge  
le document root  
pour les domaines  
ayant le ssl activé  
    $HTTP["host"]  
==  
    "ssl.petitchevalrou  
x.net" {  
server.document-  
root =  
    "/chemin/vers/htdoc  
s/"  
    }  
}
```

Activation du ssl

Activez le fichier de configuration **ssl.conf** avec la commande :

- `lighttpd-enable-mod ssl`

Et rechargez **lightty** :

- `/etc/init.d/lighttpd force-reload`

Pour aller plus loin, voir la doc du **mod ssl** de **lightty** :

<http://redmine.lighttpd.net/wiki/lighttpd/Docs:SSL>

Utilisation



Lancez l'application via le tableau de bord dash (Unity) ou via le terminal (toutes versions d'Ubuntu) avec la commande suivante :

```
machin-chose
```

Désinstallation

Pour supprimer cette application, il suffit de supprimer son paquet. Selon la méthode choisie, la configuration globale de l'application est conservée ou supprimée. Les journaux du système, et les fichiers de préférence des utilisateurs dans leurs dossiers personnels sont toujours conservés.

Voir aussi

- **(en)** [lighty](#)
- **(fr)** <http://dev.petitchevalroux.net/linux/lighttpd-ssl-linux.237.html>



Contributeurs principaux :
[Jamaïque](#).

Basé sur « *Titre original de l'article* » par Auteur Original.

1)

que nous noterons
[DISQUE]

From:
<https://nfrappe.fr/doc-0/> - Documentation du Dr Nicolas Frappé
Permanent link:
<https://nfrappe.fr/doc-0/doku.php?id=tutoriel:internet:llsp:start>
Last update: 2022/08/13 21:57