Trusty, BROUILLON

Lighttpd "how to" - serveur Web rapide et sécurisé

Le serveur web Lighttpd, aussi appelé “Lighty” par l'auteur, est un excellent outil pour les sites web de taille petite et moyenne.

Ce serveur est un excellent moyen de décharger du contenu statique comme des images ou des téléchargements binaires sur un serveur Apache surchargé et il est parfait pour les serveurs web maison ou un réseau local d'entreprise.

Il est incroyablement rapide pour servir des pages, léger et très facile à installer.

Lighttpd est beaucoup plus efficace que le gourmand serveur web Apache, plus rapide sur le même matériel en utilisant moins de 1/10 des ressources : Lighttpd est un serveur web rapide, sûr et efficace.

Sécurité, vitesse, adaptabilité et flexibilité décrivent lighttpd (prononcer Lighty).

Avec une faible empreinte mémoire par rapport à d'autres serveurs web, une gestion efficace de la charge CPU, un ensemble avancé de fonctionnalités (FastCGI, SCGI, Auth, Output-Compression, URL-Rewriting et beaucoup plus), lighttpd est la solution idéale pour les serveurs qui souffrent de problèmes de charge.

Et il est Open Source sous licence BSD révisée.

Plusieurs sites Web 2.0 populaires comme YouTube, Wikipedia et Meebo utilisent la puissance de Lighttpd. Sa haute vitesse leur permet d'évoluer mieux que d'autres serveurs web avec le même matériel. Son architecture événementielle est optimisée pour un grand nombre de connexions parallèles (keep-alive), ce qui est important pour les applications hautes performances AJAX.

Lisez également Nginx web server "how to". Il est plus rapide que Lighttpd, offre plus d'options de configuration et dispose d'une équipe de développement très active. Tout ce qui peut être fait avec Lighty peut être fait plus efficacement avec Nginx.

Dans l'exemple suivant, nous allons mettre en place un serveur web simple, expliquer les bases et vous aider à démarrer avec un serveur de travail. Le démon charge quelques modules pour

compresser les données et limiter la durée de validité pour réduire la bande passante. La journalisation complète est active, au format par défaut d'Apache et la page interne de rapports générés est active. Le module de sécurité “evasive” sera utilisé pour limiter l'accès par adresse IP pour éviter une attaque DDOS multi-connexion. Enfin, nous allons mettre en place des filtres de restriction par IP pour limiter l'accès à la structure de répertoire «hidden_dir“ où vous pourriez mettre des données non publiques plus sensibles.

Notre objectif est de mettre en place un serveur web rapide et efficace, mais surtout très sécurisé. Pour cet exemple, nous utilisons Lighttpd 1.4.x.

Pré-requis

Installation

Pour commencer, installez le paquet **lighttpd** ou en ligne de commande :

```
$ sudo apt install lighttpd
```

Le fichier principal de configuration **/etc/lighttpd/lighttpd.conf** contient toutes les directives nécessaires pour que lighttpd fonctionne comme vous le souhaitez. Sauvegardez le fichier par défaut en le renommant et le remplacer par la config fournie ci-dessus.

Configuration

Voici un exemple de fichier de configuration **/etc/lighttpd/lighttpd.conf**, entièrement fonctionnel à l'exception de la mise en place de quelques variables d'environnement.

/etc/lighttpd/lighttpd.conf

```
############################################################
###  Calomel.org  lighttpd.conf  BEGIN
############################################################
#
#### modules to load
server.modules           = ( "mod_expire",
        "mod_auth",
                             "mod_access",
                             "mod_evasive",
                             "mod_compress",
                             "mod_status",
                             "mod_redirect",
                             "mod_accesslog" )

#### performance options (aggressive timeouts)
server.max-keep-alive-requests = 6
server.max-keep-alive-idle = 15
```

```
server.max-read-idle     = 15
server.max-write-idle    = 15

## number of child worker processes to spawn (0 for lightly loaded
sites)
# server.max-worker      = 0

## number of file descriptors (leave off for lighty loaded sites)
# server.max-fds         = 512

## maximum concurrent connections the server will accept (1/2 of
server.max-fds)
# server.max-connections = 256

## single client connection bandwidth limit in kilobytes
(0=unlimited)
connection.kbytes-per-second = 0

## global server bandwidth limit in kilobytes (0=unlimited)
server.kbytes-per-second = 0

#### bind to interface (default: all interfaces)
server.bind              = "127.0.0.1"

#### bind to port (default: 80)
server.port              = 80

#### run daemon as uid (default: don't care)
server.username          = "lighttpd"

#### run daemon as gid (default: don't care)
server.groupname         = "lighttpd"

#### set the pid file (newsyslog)
server.pid-file          = "/var/run/lighttpd.pid"

#### name the server daemon publicly displays
server.tag               = "lighttpd"

#### static document-root
server.document-root     = "/var/www/htdocs/"

#### chroot() to directory (default: no chroot() )
server.chroot            = "/"

#### files to check for if .../ is requested
index-file.names         = ( "index.html" )

#### disable auto index directory listings
dir-listing.activate     = "disable"
```

```
#### disable ssl if not needed
ssl.engine                = "disable"

#### compress module
compress.cache-dir        = "/tmp/lighttpd_tmp/"
compress.filetype         = ("text/plain", "text/html", "text/css",
"image/png")

#### expire module
expire.url                = ( "" => "access plus 6 hours")

#### accesslog format (enable for using a proxy, like Pound, in
front of Lighttpd)
# accesslog.format        = "%{X-Forwarded-For}i %V %u %t \"%r\"
%>s %b \"%{Referer}i\" \"%{User-Agent}i\""

#### accesslog module
accesslog.filename        = "/var/log/lighttpd/access.log"

#### error log
server.errorlog           = "/var/log/lighttpd/error.log"

#### error pages
server.errorfile-prefix   = "/var/www/htdocs/errors/errorcode-"

#### enable debugging (un-comment to debug server problems)
# debug.log-request-header   = "enable"
# debug.log-response-header  = "enable"
# debug.log-request-handling = "enable"
# debug.log-file-not-found   = "enable"

#### mod_evasive
evasive.max-conns-per-ip = 250

#### limit request method "POST" size in kilobytes (KB)
server.max-request-size  = 1

#### disable multi range requests
server.range-requests    = "disable"

#### disable symlinks
server.follow-symlink    = "disable"

#### ONLY serve files with all lowercase file names
server.force-lowercase-filenames = "disable"

#### server status
status.status-url         = "/hidden_dir/server-status"

#### password protected area
# auth.backend                  = "htdigest"
```

```
# auth.backend.htdigest.userfile =
"/var/www/htdocs/hidden_dir/passwd_file"
# auth.require                   = ( "/hidden_dir" =>
#                                    (
#                                      "method"  => "digest",
#                                      "realm"   => "REALM_NAME",
#                                      "require" =>
"user=USER_NAME"
#                                    )
#                                  )

##
#### Blocks Section: The order is important.
#### Test all block rules before going live.
##

#### request method restrictions (v1.5.x ONLY)
# $HTTP["request-method"] !~ "^(GET|HEAD)" {
#     url.access-deny = ( "" )
#  }

#### deny access to unwanted bots or bad clients
# $HTTP["useragent"] =~ "(Google|BadGuy)" {
#   url.access-deny = ( "" )
# }

#### access control list for hidden_dir (not for use behind
proxies)
# $HTTP["remoteip"] !~ "127.0.0.1|10.10.10.2|20.10.20.30" {
#     $HTTP["url"] =~ "^/hidden_dir/" {
#       url.access-deny = ( "" )
#     }
#  }

#### url redirect requests for calomel.org to www.calomel.org
# $HTTP["host"] =~ "^(calomel.org)$" {
#         url.redirect = ( "/(.*)" => "http://www.%1/$1")
# }

#### stop image hijacking (anti-hotlinking)
# $HTTP["referer"] !~
"^(http://calomel\.org|http://www\.calomel\.org)" {
#     url.access-deny = ( ".jpg", ".jpeg", ".png", ".avi", ".mov"
)
# }

#### virtual host limits
# $HTTP["host"] !~ "^(calomel\.org|www\.calomel\.org)" {
#     url.access-deny = ( "" )
#  }
```

```
#### stop referer spam
# $HTTP["referer"] =~ "(tarotathome|casinospam)" {
#     url.access-deny = ( "" )
#  }

#### mimetype mapping
mimetype.assign             = (
  ".pdf"          =>      "application/pdf",
  ".sig"          =>      "application/pgp-signature",
  ".spl"          =>      "application/futuresplash",
  ".class"        =>      "application/octet-stream",
  ".ps"           =>      "application/postscript",
  ".torrent"      =>      "application/x-bittorrent",
  ".dvi"          =>      "application/x-dvi",
  ".gz"           =>      "application/x-gzip",
  ".pac"          =>      "application/x-ns-proxy-autoconfig",
  ".swf"          =>      "application/x-shockwave-flash",
  ".tar.gz"       =>      "application/x-tgz",
  ".tgz"          =>      "application/x-tgz",
  ".tar"          =>      "application/x-tar",
  ".zip"          =>      "application/zip",
  ".mp3"          =>      "audio/mpeg",
  ".m3u"          =>      "audio/x-mpegurl",
  ".wma"          =>      "audio/x-ms-wma",
  ".wax"          =>      "audio/x-ms-wax",
  ".ogg"          =>      "application/ogg",
  ".wav"          =>      "audio/x-wav",
  ".gif"          =>      "image/gif",
  ".jpg"          =>      "image/jpeg",
  ".jpeg"         =>      "image/jpeg",
  ".png"          =>      "image/png",
  ".xbm"          =>      "image/x-xbitmap",
  ".xpm"          =>      "image/x-xpixmap",
  ".xwd"          =>      "image/x-xwindowdump",
  ".css"          =>      "text/css",
  ".html"         =>      "text/html",
  ".htm"          =>      "text/html",
  ".js"           =>      "text/javascript",
  ".asc"          =>      "text/plain",
  ".c"            =>      "text/plain",
  ".cpp"          =>      "text/plain",
  ".log"          =>      "text/plain",
  ".conf"         =>      "text/plain",
  ".text"         =>      "text/plain",
  ".txt"          =>      "text/plain",
  ".dtd"          =>      "text/xml",
  ".xml"          =>      "text/xml",
  ".mpeg"         =>      "video/mpeg",
  ".mpg"          =>      "video/mpeg",
  ".mov"          =>      "video/quicktime",
  ".qt"           =>      "video/quicktime",
```

```
  ".avi"          =>      "video/x-msvideo",
  ".asf"          =>      "video/x-ms-asf",
  ".asx"          =>      "video/x-ms-asf",
  ".wmv"          =>      "video/x-ms-wmv",
  ".bz2"          =>      "application/x-bzip",
  ".tbz"          =>      "application/x-bzip-compressed-tar",
  ".tar.bz2"      =>      "application/x-bzip-compressed-tar"
 )
#
############################################################
###  Calomel.org lighttpd.conf  END
############################################################
```

Explication des directives de lighttpd.conf

Maintenant, nous devons modifier le fichier de configuration. Jetons un oeil à chacune des directives.

- **options de performance (temps morts agressifs)** :
  Ce sont les délais d'attente pour toutes les connexions au serveur. Nous utilisons des délais agressifs pour fermer les processus enfants rapidement pour économiser de la mémoire et du temps CPU. Pas besoin de garder une connexion ouverte si elle n'est pas utilisée.
  - **server.max-keep-alive-requests** : nombre maximum de demandes dans une session avant que le serveur ne termine la connexion. (Par défaut: 16) Ne pas mettre cette valeur à moins que le plus grand nombre d'objets servi sur une seule page si le serveur est chargé (> 500 demandes / sec). Par exemple, si vous avez 3 photos, un fichier CSS, un favicon.ico et la Page principale alors cela fait 6 objets → ici, la valeur ne doit pas être inférieure à 6. Si vous avez un serveur à faible trafic (<1 000 demandes / minute), vous pouvez définir cette valeur à zéro pour servir une demande et fermer la connexion. Cela va forcer les clients à se connecter une fois pour chaque objet sur la page. Cela signifie que vous pouvez utiliser votre pare-feu pour bloquer les agresseurs multi-connexion.
  - **server.max-keep-alive-idle** nombre maximum de secondes avant fermeture de la connexion pour une connexion en attente (Par défaut: 30)
  - **server.max-read-idle** nombre maximum de secondes avant fermeture de la connexion pour une lecture en attente (Par défaut: 60)

    - **server.max-write-idle** nombre maximum de secondes avant fermeture de la connexion pour un appel d'écriture en attente (Default: 360)
- **server.max-worker** : nombre de processus à gérer. (Default: 0) NOTE: Si vous utilisez server.max-worker, le module mod_status ne montrera pas les statistiques de tous les enfants du serveur combinés. Vous verrez des statistiques différentes à chaque demande.
- **server.max-fds** : nombre maximal de descripteurs de fichiers. Pour voir le nombre en cours, tapez

```
ulimit
```

  La limite de descripteurs de fichier est le nombre maximal de fichiers qu'un processus ou un utilisateur peut ouvrir en même temps. En termes de serveur Web, c'est le montant total de fonctions, connexions utilisateurs reconnus, ou autres tâches, y compris php mysql.

- **server.max-connections** : nombre total de connexions simultanées que le serveur accepte. Une valeur sûre est la moitié la valeur spécifiée pour “server.max-fds”.

- **connection.kbytes-per-second** : limite en kilo-octets (Ko) pour chaque connexion. Gardez à l'esprit qu'une limite en dessous de 32kb/s pourrait effectivement limiter le trafic à 32kb/s. Cela est dû à la taille de la mémoire tampon d'envoi TCP.
- **server.kbytes-per-second** : limite en kilo-octets (Ko) pour le serveur Web entier. Gardez à l'esprit qu'une limite en dessous de 32kb/s pourrait effectivement limiter le trafic à 32kb/s. Cela est dû à la taille de la mémoire tampon d'envoi TCP.
- **server.bind** : interface à laquelle se reliera le démon. Si vous souhaitez le lier à une adresse IP spécifique, la mettre ici. Si vous voulez que lighttpd se lie à toutes les interfaces alors commenter cette ligne.
- **server.port** : Par défaut, le démon se lie au port 80. Si vous avez un autre port à lier comme 8080, saisissez-le ici.
- **server.username** and **server.groupname** : utilisateur et groupe sous lesquels travaillera le démon. Si vous voulez que lighttpd tourne en tant qu'utilisateur non privilégié, c'est l'endroit pour le mettre. Si vous commentez ces lignes, le démon tournere comme l'utilisateur qui a lancé le processus ou “nobody” selon la distribution du système d'exploitation que vous utilisez.

- **server.pid-file** : Le fichier d'id de processus permet de HUP le serveur ou de faire tourner les logs. Assurez-vous que le fichier de pid peut être fait en vérifiant le répertoire de travail de votre choix.
- **server.tag** : nom du serveur que lighttpd enverra aux clients distants dans les en-têtes HTTP. Normalement, il est utilisé par les robots collecteurs pour classer chaque type de serveur et établir une liste des serveurs Web en cours d'exécution sur le web. L'auteur de lighttpd voudrait que chacun mette “lighttpd”, mais vous pouvez mettre n'importe quoi dans cette chaîne.
- **server.document-root** : racine de l'arborescence web.
- **server.chroot** : Cela signifie que le démon “chroote” l'accès des utilisateurs à la racine du serveur.
- **index-file.names** : nom du fichier d'index que le démon doit rechercher dans un répertoire si le client ne le précise pas. Par défaut, index.html.
- **dir-listing.activate** : active ou désactive la liste de répertoire pour un répertoire qui ne contient pas de fichier index.html ou de tout fichier répertorié dans “index-file.names”.

- **compress.cache-dir** and **compress.filetype** : Le réglage de compression permettra au démon de compresser avec gzip les types de fichiers que vous spécifiez et de les placer dans le cache-dir. Cela peut économiser beaucoup de bande passante sortante. Comme bonus supplémentaire, les clients distants obtiennent les données plus rapidement et votre site semble nettement plus rapide. Il est important de se rappeler que tous les types de fichiers ne sont PAS à comprimer. JPG par exemple, est nativement comprimé et le comprimer à nouveau avec gzip va en réalité le faire grossir.
- **expire.url** : La balise d'en-tête “expire” dit aux clients qu'ils doivent conserver une copie de l'objet qu'ils ont déjà téléchargé pendant un laps de temps spécifié. Dans notre exemple, nous disons au client de conserver les données dans la mémoire cache du navigateur local à partir du moment où ils ont accédé aux données jusqu'à au moins 6 heures plus tard. Cela permet d'économiser aussi beaucoup de bande passante de téléchargement pour vous. Au lieu de clients qui télécharge la même image bannière, encore et encore, ils peuvent garder une copie locale et ne charger que les changements sur votre site. Imaginez un client qui charge 5 pages de votre site. Chaque page a une bannière de 15KB. Avec expire, le client ne télécharge la bannière qu'une fois au lieu de 5 (15KB

par rapport à 75Ko). Cela accroît votre bande passante rend votre site beaucoup plus rapide à répondre.

- **accesslog.format** : Si vous utilisez un proxy en amont de Lighttpd, l'adresse IP du proxy sera affichée dans les journaux au lieu de l'ip du client distant. Décommentez cette ligne si vous utilisez un proxy, pour lire le champ “X-Forwarded-For” de l'en-tête et entrer l'adresse ip du client distant dans le fichier journal d'accès Lighttpd. Si vous n'utilisez pas un proxy alors gardez cette ligne commentée. La valeur par défaut de accesslog.format est le bon format et travaillera avec Webalizer, Analog et AwStats.
- **accesslog.filename** and **server.errorlog** : emplacement du journal d'accès et les journaux d'erreurs pour le démon. Assurez-vous que l'utilisateur que le démon est en cours d'exécution comme on peut accéder et écrire dans ce répertoire.
- **server.errorfile-prefix** : Cette directive est pour le répertoire de la page d'erreur personnalisée si vous souhaitez en utiliser une. Sinon, mettez cette ligne en commentaire et lighttpd va générer sa propre page d'erreur. Dans l'exemple que nous faisons que le répertoire ”/var/www/htdocs/errors/“ inclue les fichiers d'erreur html. Ensuite, pour chacun des codes d'erreur (400-417 and 500-505) nous faisons une page d'erreur personnalisée. Les noms de chaque page d'erreur commencent par la chaîne “errorcode-” comme nous l'avons précisé dans la directive. Ainsi, le fichier de l'erreur 404 serait “errorcode-404.html” et celui de l'erreur 505 serait “errorcode-505.html”.
- **debug.log-???** : options à décommenter si vous souhaitez voir toutes les réponses d'erreur et les demandes des clients. Très utile pour trouver des bugs ou si quelque chose va mal dans un site web.
- **mod_evasive** : Ce module va limiter le nombre de fois où une adresse IP simple peut se connecter à notre serveur et recevoir la bonne page. Dans l'exemple nous avons mis la limite à 25 connexions par ip. Après cette limite, le client obtiendra la page d'erreur 404.
- **limit request method “POST” size in kilobytes (KB)** : The server.max-request-size is the limit in kilobytes for data accepted through the request method “POST”. If you run a web server that does not accept uploads then you can set the POST request method data limit to one(1) kilobyte, in effect POST request will always be denied. This is not really necessary, but it will help deny confused browsers and malicious bots. The maximum size in kbytes of the request (header + body). Only applies to POST

requests. Default: 2097152 (2GB)

- **server.range-requests** : defines if range requests are allowed or not. Range request are requests of one or more sub-ranges of a file. Range requests are very helpful for interrupted downloads and fetching small portions of huge files. The problem is download accelerators car put a huge load on your server by downloading many different parts of a file at the same time. Most of the time the download accelerator is broke and is not helping anyone by using up extra bandwidth. Internet Explorer (IE) also has problems opening up some files like PDF's is the range option is enabled. The example conf disables server.range-requests just to be safe. (Default: enabled)
- **server.follow-symlink** : If you do not use, and do not expect to use sym links then disable them.
- **server.force-lowercase-filenames** : This directive will force lighttpd to _ONLY_ serve files with lower case file names. Be careful if you enable this one. A file with mixed case, like “TeSt.html” will not be served due to upper case letters if this directive is enabled. (Default: disabled)
- **status.status-url** : This is the lighttpd dynamically generated status page. It will show the time the server has been running, total traffic served and some time based stats like traffic for the last 5 minutes. Notice we put this page into the “hidden_dir” which is explain below.

- **password protected area** : If you have an area of the web site you only want authorized personnel to see then you can protect it. This set of directives will password protect the /hidden_dir and all files and directories under it, like the server-status page. We will use the htdigest method which is the best choice to use especially for non-https sites. It will not send any of the passwords in clear text. Check “man htdigest” for details.
  To make a password file use the binary htdigest in the form:

```
htdigest -c
/var/www/htdocs/hidden_dir/passwd_file
REALM_NAME USER_NAME
```

- **request method restrictions** : Request Method restrictions were added in v1.5.x. This allows you to filter on GET, HEAD, POST, SEARCH, etc. We will be limiting access to our example server to GET and HEAD requests only as we do not allow uploads or any other

options due to security concerns. All other request methods will get an error.

- **deny access to unwanted bots or bad clients** : If you wanted to deny access to certain user agents like the Google bot or perhaps all old versions of IE you can do that here. Put in part of the string of the user agent you want to deny.
- **access control list** : This is a way you can define a directory and only allow clients coming from the specified ips to have access. This might be nice to allow internal LAN clients access to the status pages or employee contact information and deny other clients. In our example we will allow the clients coming from localhost 127.0.0.1, an internal LAN ip 10.10.10.2 and some external ip 20.10.20.30 to access the protected “hidden_dir” directory.
- **url redirect** : If you prefer clients who connect to your server to instead use the sub domain www use this rule. For example, if a browser connects to calomel.org they will be redirected to the url www.calomel.org . If they then save the bookmark it will show up as the www sub domain.
- **stop image hijacking** : Image hijacking is when someone makes a link to your site to one of your pictures or videos, but displays it on their site as their own content. The reason this is done is to send a browser to your server to use your bandwidth and make the content look like part of the hijacker's site. This is most common as people make links to pictures and add them to a public forum or blog listing. They get to use your picture in their content and not have to use their bandwidth or server to host the file. In order to keep your bandwidth usage low you should block access to images from those clients who are not requesting the connection from your site. Note, this function can be used for any kind on content. Just add the file types to the url.access-deny list. If would like more ideas on lowering bandwidth usage check out our Saving Webserver Bandwidth (Tips).
- **virtual host limits** : A virtual host is the hostname of your web server. For example this site is called calomel.org. Some bots and scanners will try to access your site using the ip address or no hostname header at all to bypass virtual host limitations. We can block this type of behavior by requiring all clients who want to access our server to reference us by our official host name. This will block anyone who is scanning ip addresses or trying to be mischievous, but allow normal clients like browsers and bots like Google.

- **stop referer spam** : Referer spam is more of an annoyance than a problem. A web site will connect to your server with the referer field referencing their web site. The idea is that if you publish your web logs or statistics then their hostname will show up on your page. When a search bot like Google comes by it will see the link from your site to theirs and give them more PageRank credit. First, never make your weblogs public. Second, block access to referer spammers with these lines.
- **mimetype.assign** : These directive simply allow our server to send the the proper file type and application type to the clients. The list in the example is the same as the default lighttpd config options.

Utilisation

Starting the daemon

Now that you have the config file installed and configured you can start the daemon by hand with ”/usr/local/sbin/lighttpd -f /etc/lighttpd.conf“ or add the following into /etc/rc.local to start the lighttpd daemon on boot.

```
#### lighttpd start in /etc/rc.local
if [ -x /usr/local/sbin/lighttpd ]; then
   echo -n ' lighttpd'; /usr/local/sbin/lighttpd -f /etc/lighttpd.conf
fi
```

Conclusion

Lighttpd has many more options if you wish to use fast-cgi or password authentication. I would highly suggest taking some time to read the lighttpd.net documentation if you need more web functionality. If you are happy with the options we have started out with then at this point all that is left is finding some content to serve and setup your web tree.

SECURITY NOTE: We highly recommend using a reverse proxy like Pound to front end your web servers. Pound offers an added layer of security and can be used to filter most malicious requests from ever hitting the web server. Use Pound to stop the bad bots and allow your web server to focus on good clients. For a complete explanation and examples check out the Calomel.org Pound reverse proxy “how to”.

Désinstallation

Voir aussi

- **(en)** lighty
- **(fr)** lighty

Contributeurs principaux : Jamaique.

Basé sur <Titre original de l'article> par <Auteur Original>.