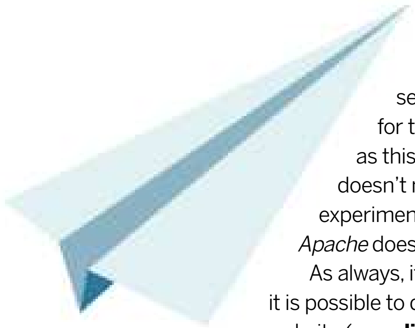
# Serve pages at the speed of Lighttpd

Lighttpd is a fast, lightweight alternative to Apache designed for high-traffic sites. It's also a doddle to set up, as **Juliet Kemp** demonstrates.

*Apache* is pretty much the default web server for anyone running Linux these days, and it's an excellent piece of software. But for some very very high-throughput, high-load sites, it can run a little slower than is ideal. *Lighttpd* is specifically designed to have minimal memory footprint and to operate as quickly as possible. It's particularly good for serving up static content, scaling very well across multiple servers, and it works well with *FastCGI* (you can set it up to use an array of *FastCGI* servers, with *Lighttpd* acting as the front-end for them). It's also fairly straightforward to set up, as this tutorial will demonstrate. Even if your site doesn't need the speed of *Lighttpd*, it's fun to experiment with – it even feels faster to configure than *Apache* does!



"Lighttpd even feels faster to configure than Apache does."

As always, if you want to run the most up to date version, it is possible to download the source from the project's website (**www.lighttpd.net**), which will give you version 1.4.23. However, it's almost always a better idea to use the package available for your distro (as this is a more maintainable setup in the long run). In Debian/Ubuntu the package name is *Lighttpd* (the version in Debian Lenny and Ubuntu 9.04 is 1.4.19, so it's not too far behind the main release); it's also available as an RPM.

Unless you have another web server already running on port 80 (see the box below if you are), the default configuration setup as created by the package install should enable you to start the server immediately, with **/etc/init.d/lighttpd start**. If you'd prefer to run it in debug mode while you're setting up (so it will stay in the foreground of the terminal and output any log messages straight to the screen), use **lighttpd -D -f /etc/lighttpd/lighttpd.conf**.

Go to **http://localhost** and you should get a basic placeholder page (this may vary according to who packaged your version). The basic default config that the install sets up looks a bit like this:

```
# Your config file may have more lines than this and they
may not be in this order: that's OK!
server.modules          = (
            "mod_access",
            "mod_alias",
            "mod_accesslog",
            "mod_compress",
)
server.document-root            = "/var/www/"
server.port                     = 80
server.errorlog         = "/var/log/lighttpd/error.log"
accesslog.filename  = "/var/log/lighttpd/access.log"
index-file.names        = ( "index.php", "index.html",
                        "index.htm", "default.htm",
                        "index.lighttpd.html" )
url.access-deny         = ( "~", ".inc" )
static-file.exclude-extensions = ( ".php", ".pl", ".fcgi" )
include_shell "/usr/share/lighttpd/create-mime.assign.pl"
```

This should all be fairly self-explanatory! There's a very small number of modules enabled by default, the logs are set up (the access log requires the **mod_accesslog** module line in the **server.modules** section), and index filenames are set (these are the filenames to look for if a broswer requests **http://example.com/directory/**).

The **url.access-deny** setting sets which files should not be served at all: here, it's just files with a **~ postfix** (this usually indicates a backup version of a file, and you probably don't want your old content being accessible to the web at large),

## Changing the port

Want to test out *Lighttpd* alongside an existing *Apache* (or other) webserver? Change the **server.port** line in **/etc/lighttpd/lighttpd.conf**, eg:

```
server.port             = 81
```

(here saved at **index.lighttpd.html** because there was an existing *Apache*-provided **index.html** in the **/var/www** directory). Also note the server running on port 81. Restart *Lighttpd* with **/etc/init.d/lighttpd** restart, and then check out **http://localhost:81**.

and **.inc** files, which are source files for generating dynamic content. Note that at the moment our webserver won't handle any dynamic content, since it doesn't have the necessary modules enabled and configured; but even when it's set up to do so, you don't want the source files to be viewable by the public, as this can be a security problem.

The Debian setup generates MIME types using a script that is included (and run) using the **include_shell** line; otherwise you can specify them manually (add more lines if you like!):

```
# Replace the include_shell line with these lines
mimetype.assign = (
  ".html" => "text/html",
  ".txt" => "text/plain",
  ".jpg" => "image/jpeg",
  ".png" => "image/png"
)
```

To test your config changes before you run the server, it's a good idea to use

```
lighttpd -t -f /etc/lighttpd/lighttpd.conf
```

Another thing you may want to do early on is to enable user directories. This is easy:

```
lighty-enable-mod userdir
/etc/init.d/lighttpd force-reload
```

## PHP and FastCGI

Currently your webserver deals only with static content – which is in fact one of the things that *Lighttpd* is particularly good at. Some large sites use *Lighttpd* for their static content and an alternative for dynamic content, because serving up static files is so much simpler. However, *Lighttpd* also works well with dynamic content, and in particular with the FastCGI protocol, which provides an interface between your web server and external applications: it's much like CGI, but it scales far better. FastCGI is platform-independent, so any FastCGI programs that you've already set up to run on *Apache* or any other web server will also work fine with *Lighttpd*. FastCGI aims to get rid of some of the performance limitations of CGI, and *Lighttpd* improves that further by doing internal FastCGI load balancing as well, leading to improved performance compared with *Apache* and **mod_php**.

To set this up on your new lighttpd server, you need **mod_fastcgi**. This should be available in **/etc/lighttpd/mods-available/:** to enable it, use

```
# /usr/sbin/lighty-enable-mod
# /etc/init.d/lighttpd force-reload
```

The **lighty-enable-mod** script will give you a list of available modules and ask you which to enable; after that you reload the server to load the changes in.

We're looking here at running FastCGI with PHP (like CGI, it's language-independent, so you can also use it with Perl or another language). If you're doing this, you'll also need to install PHP5-CGI if you don't already have it installed (on Debian Lenny and Ubuntu, this is the **php5-cgi** package).

If you're setting up **php5-cgi**, you'll need to edit the **/etc/php5/cgi/php.ini** file, to add this line at the end:

```
cgi.fix_pathinfo = 1
```

Restart the server, and that's it: you're all set up for PHP support via FastCGI! (Unfortunately there's no module yet to actually write the PHP for you...)

## SSL configuration

Another useful piece of web server setup is SSL, to allow you to deliver secure pages with **https://** links. The default Debian/Ubuntu *Lighttpd* has SSL enabled: to check that this is the case for your install, run **lighttpd -v** and check that your output looks something like this:

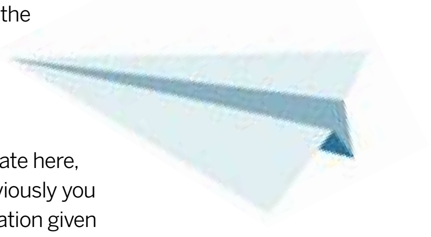
```
lighttpd-1.4.19 (ssl) - a light and fast webserver
```

The **(ssl)** is the important bit. Then use **lighty-enable-mod** to enable the SSL module, as described above in the PHP section. You'll need to edit the **/etc/lighttpd/sites-available/10-ssl.conf file** to set the correct parameters for your server certificate:

```
$SERVER["socket"] == "10.0.0.9:443" {
    ssl.engine  = "enable"
    ssl.pemfile = "/etc/lighttpd/www.server.org.pem"
    server.name = "www.server.org"
}
```

You should edit the address on the top line, and the **ssl.pemfile** and **server.name lines**. Then reload the server with **/etc/init.d/lighttpd force-reload** and you should be able to access **https://** URLs on your server.

We won't cover setting up an SSL server certificate here, as that's outside the scope of this tutorial – but obviously you will need to have a valid server certificate at the location given by **ssl.pemfile** for this to work. Have a look at the *openssl* command for creating a certificate.



## Setting up a virtual server

*Lighttpd* happily supports multiple virtual servers based on the URL that is used to access the page. You set up a conditional expression in the config file, and *Lighttpd* will use that to determine what content you serve (and how you serve it). Here's an example that you can add to your config file to handle **http://www.myothersite.net** URLs (you will of course also need to have the domain registered and DNS set up properly for this to work!):

```
$HTTP["host"] == "www.myothersite.net" {
  server.document-root = "/var/www/myothersite/"
  $HTTP["url"] =~ "^/usefulscripts/" {
    dir-listing.activate = "enable"
  }
}
```

If the URL being served contains **www.myothersite.net**, *Lighttpd* will use the configuration inside this conditional instead of the server default. In this case, that means changing the document root of the site (ie if someone requests **http://www.myothersite.net/test.html**, the file **/var/www/myothersite/test.html** will be served up); and setting up the **usefulscripts** directory to allow directory listing (so asking for **http://www.myothersite.net/usefulscripts/** will give you a directory listing –this is turned off by default for security reasons). You can put whatever options you like within the conditional setup, and you can also run as many of these virtual hosts as you like. LXF

## Rewrite rules

You'll need to create new rewrite rules if you have existing ones in **.htaccess** files, as *Lighttpd* does not use **.htaccess**. Here's an example of a *Lighttpd* rewrite rule:

```
url.rewrite-once = ( "^/faq/([0-9]+)$" => "/faq.php?id=$1" )
```

This will turn http://www.example.com/faq/71 into http://www.example.com/faq.php?id=71.

Another option is to Canonicalise filename extensions:

```
url.rewrite-once = ( "/(.*)\.htm" => "/$1.html" )
```

This will mean that if someone uses **.htm** instead of **.html**, they'll be rerouted to the correct page. To enable rewriting, uncomment the **mod_rewrite** line in the **server.modules** section of the config file and reload the server.