Setting up Apache webserver in Proxmox VE (part 2)

In our example, we will be using apachehustler as our domain name

We would be doing:

1. Creating virtual host file
2. Configure SSL
3. Setup website files
4. Configure security settings

Basic directory structure:

* Web files go in: /var/www/html/
* Configuration files in: /etc/apache2/
* Main config file: /etc/apache2/apache2.conf

Create a folder with your domain name

A screen shot of a computer

Description automatically generated

Now change ownership to your own user for all items created in your domain web file



Ensure new files created can also be executed



Create index.html inside which will be the default page for your webpage



Add some content inside just to see if the virtual host works

A screenshot of a computer

Description automatically generated

Create the virtual host configuration



Now add the config below

<VirtualHost \*:80>

ServerAdmin webmaster@localhost

ServerName apachehustler ServerAlias www.apachehustler

DocumentRoot /var/www/apachehustler

ErrorLog ${APACHE\_LOG\_DIR}/apachehustler\_error.log CustomLog ${APACHE\_LOG\_DIR}/apachehustler\_access.log combined

</VirtualHost>

A screenshot of a computer

Description automatically generated

Now run sudo a2ensite apachehustler.conf (creates symbolic link in sites-enabled)

A black background with white text

Description automatically generated

The workflow:

1. Create config in sites-available
2. Enable site using a2ensite which creates symbolic link in sites-enabled
3. Disable site using a2dissite which removes the symbolic link

Now we will need to enable the virtual host

Add this part into your apachehustler.conf

A screen shot of a computer

Description automatically generated

EXPLANATION:

**1. <VirtualHost \*:80> ... </VirtualHost>**

* **Meaning**: This block defines a virtual host that listens on port 80 on all available network interfaces (\*).
* **Port 80**: This is the default port for HTTP (non-encrypted) traffic.

**2. ServerAdmin webmaster@localhost**

* **Meaning**: Sets the email address for the server administrator. This is mostly used for error messages and documentation (e.g., shown in server-generated error pages).

**3. ServerName apachehustler**

* **Meaning**: Specifies the primary hostname of this virtual host. When a client tries to reach a server with this name (or via DNS pointing to this server), Apache will use this configuration.

**4. ServerAlias www.apachehustler**

* **Meaning**: Adds an additional name for this virtual host. If someone visits www.apachehustler, Apache will treat it the same as if they visited apachehustler.

**5. DocumentRoot /var/www/apachehustler**

* **Meaning**: The directory on the server’s filesystem where the files for this site are located. Apache will serve the files from here when users visit http://apachehustler/ (or the aliased domains).

**6. ErrorLog ${APACHE\_LOG\_DIR}/apachehustler\_error.log**

* **Meaning**: Sets the path to the error log file for this virtual host.
* **${APACHE\_LOG\_DIR}**: Typically, this is defined in Apache’s environment variables and usually points to /var/log/apache2 or a similar path, depending on the OS.

**7. CustomLog ${APACHE\_LOG\_DIR}/apachehustler\_access.log combined**

* **Meaning**: Sets the path and format of the access log for this virtual host.
* **combined**: A standard log format that includes additional details (like user agents, referrers, etc.).

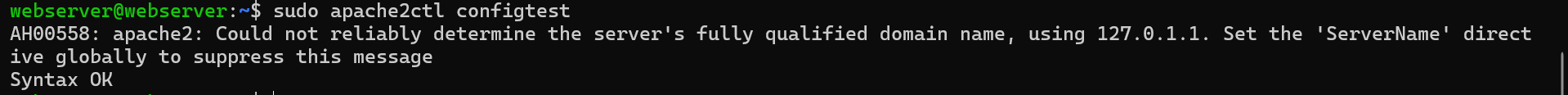
**8. <Directory /var/www/apachehustler> ... </Directory>**

This is a <Directory> directive block that applies directives to the given filesystem path.

Within this block:

1. **Options Indexes FollowSymLinks**
   * **Indexes**: Allows Apache to generate an index (list) of the files in a directory if no default file (e.g., index.html) is present.
   * **FollowSymLinks**: Allows Apache to follow symbolic links in the filesystem.
2. **AllowOverride All**
   * **Meaning**: Permits .htaccess files in this directory (and subdirectories) to override Apache configurations. This means you can put rewrite rules or other directives in .htaccess.
3. **Require all granted**
   * **Meaning**: Allows all clients (any IP address) to access the directory’s contents, as opposed to restricting it.

Run sudo apache2ctl configtest to test the config file



That is just a warning which could be supressed by adding this:

A screen shot of a computer

Description automatically generated

If you run configtest again should be ok now

A black background with white text

Description automatically generated

Now if you are trying to access this from a windows PC you can add these to your host file in C:\Windows\System32\drivers\etc\host. If linux can use sudo vim /etc/hosts.

Hosts file acts like a local DNS where it maps the ip to domain name. so instead of typing your apache server vm ip you can type http://apachehustler

A close-up of a computer

Description automatically generated

As we did not remove the default page we can still see the default apache webpage loaded if you type in the ip address. If you type in the domain name however you can see apache hustler working now. To disable default site can run sudo a2dissite 000-default.conf.

A screen shot of a computer

Description automatically generated

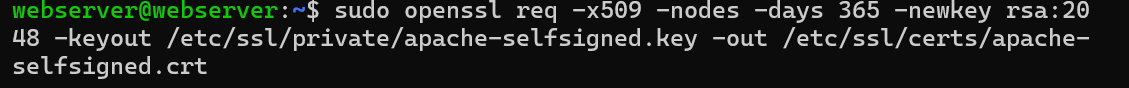
Next, we will setup SSL for secure connections over HTTPS.

Firstly enable ssl module in Apache and then restart apache2

A screen shot of a computer

Description automatically generated

Then create the SSL certificate with: sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt



EXPLANATION TIME:

**req**

* Indicates that you’re making a certificate request (which includes generating a key and certificate).

**-x509**

* Tells OpenSSL to create a self-signed **X.509** certificate (as opposed to creating a Certificate Signing Request, or CSR, that would be signed by a Certificate Authority).

**-nodes**

* Short for “no DES” encryption on the private key. This means the private key is **not** password-protected or encrypted, so Apache can read it at startup without prompting for a passphrase.

**-days 365**

* The number of days for which the certificate will be valid (here, 1 year).

**-newkey rsa:2048**

* Creates a new 2048-bit **RSA** key pair at the same time as generating the certificate.

**-keyout /etc/ssl/private/apache-selfsigned.key**

* The path where the **private key** file will be saved.

**-out /etc/ssl/certs/apache-selfsigned.crt**

* The path where the **self-signed certificate** file will be saved.

You can fill up the fields accordingly or you can just key in . to leave it blank. However for Common name fill up same as domain name which is apachehustler

A computer screen shot of white text

Description automatically generated

Create SSL virtual host config with: sudo vim /etc/apache2/sites-available/apachehustler-ssl.conf

A black screen with white text

Description automatically generated

Add these in:

A screenshot of a computer screen

Description automatically generated

EXPLANATION TIME

**<VirtualHost \*:443>**

* Defines a Virtual Host that listens for HTTPS traffic on port 443.

**ServerAdmin webmaster@localhost**

* Email address for the server administrator (shown in error pages).

**ServerName apachehustler / ServerAlias www.apachehustler**

* The main domain (apachehustler) and its alias (www.apachehustler) served by this virtual host.

**DocumentRoot /var/www/apachehustler**

* The directory containing the website files.

**SSLEngine on**

* Enables SSL/TLS for this virtual host.

**SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt**

* Specifies the path to the SSL certificate.

**SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key**

* Specifies the path to the matching private key for the certificate.

**<Directory /var/www/apachehustler>**

* Configuration for that directory:
  + **Options Indexes FollowSymLinks**: Allows directory listings (if no index file) and following symbolic links.
  + **AllowOverride All**: Lets .htaccess files override Apache settings.
  + **Require all granted**: Permits all IPs to access this directory.

**ErrorLog ${APACHE\_LOG\_DIR}/apachehustler\_error.log**

* The file location for error logs specific to this site.

**CustomLog ${APACHE\_LOG\_DIR}/apachehustler\_access.log combined**

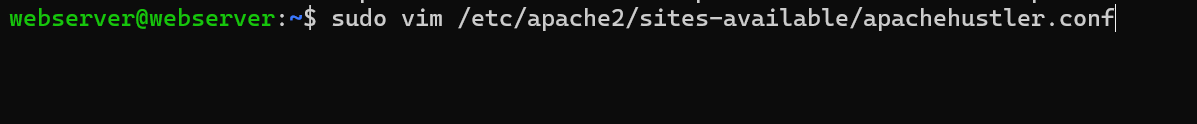
* The file location for access logs using the “combined” log format (includes referrers, user agents, etc.).

Enable Apache SSL site

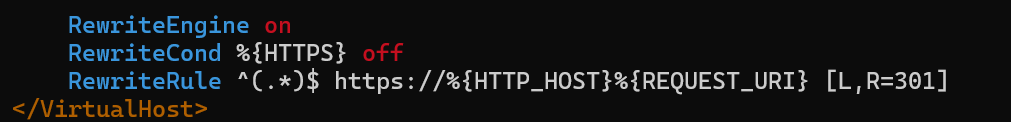
A black screen with white text

Description automatically generated

Modify your non-SSL config



Add these in just before the closing virtualhost tag



EXPLANATION

**9. RewriteEngine on**

* **Meaning**: Activates Apache’s mod\_rewrite engine, used for rewriting URLs, redirects, etc.

**10. RewriteCond %{HTTPS} off**

* **Meaning**: A rewrite condition that checks whether HTTPS is off (meaning the request is using HTTP and not SSL/TLS).

**11. RewriteRule ^(.\*)$ https://%{HTTP\_HOST}%{REQUEST\_URI} [L,R=301]**

* **Meaning**: When the above condition (HTTPS off) is true, this rule sends a 301 redirect to the https version of the same URL.
* **^(.\*)$**: Captures the entire path (any URL path after the domain).
* **https://%{HTTP\_HOST}%{REQUEST\_URI}**: This rebuilds the URL using the same domain (%{HTTP\_HOST}) and the requested URI/path (%{REQUEST\_URI}), but changes the protocol from http to https.
* **[L,R=301]**:
  + R=301 is an HTTP status code for a “permanent redirect.”
  + L means this is the “last rule” to process if this rule matches.

**Put simply**: If someone visits the site using HTTP, Apache automatically redirects them to the same URL but over HTTPS.

Enable rewrite module

A black background with white text

Description automatically generated

Do a configtest then reload the new config and restart services

A black background with white text

Description automatically generated

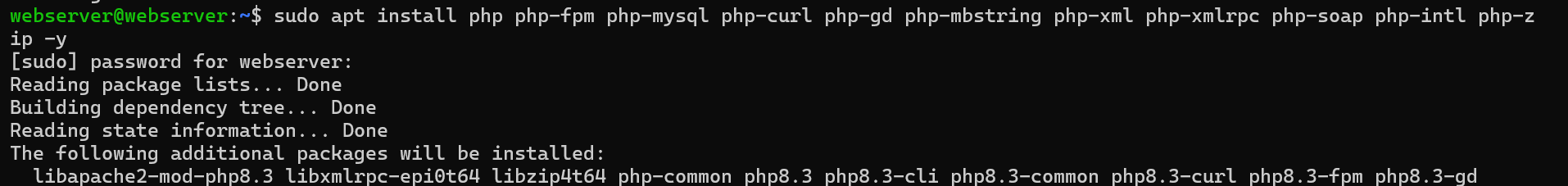
Now if you try to use <http://apachehustler> it will show this prompt just click on advanced then proceed to… this warning is due to we are using a self signed cert but the redirection to https worked.

A screenshot of a computer

Description automatically generated

Now, we can start with setting up PHP

sudo apt install php php-fpm php-mysql php-curl php-gd php-mbstring php-xml php-xmlrpc php-soap php-intl php-zip -y



php: Core PHP package

php-fpm: FastCGI Process Manager (better performance than mod\_php)

php-mysql: For database connections

php-curl: For making HTTP requests php-gd: For image processing

php-mbstring: For handling multi-byte strings

php-xml, php-xmlrpc: For XML processing

php-soap: For SOAP web services

php-intl: For internationalization

php-zip: For handling ZIP files

Enable PHP-FPM in Apache. PHP-FPM manages pools of PHP worker processes more efficiently. It can create more processes when traffic is high and scale down when traffic is low, optimising server resource usage.

A black screen with white text

Description automatically generated

A black background with white text

Description automatically generated

Now configure the PHP settings



We use \* incase there are multiple php versions so you open all the php.ini files inside all versions. (e.g. /etc/php/7.4/fpm or /etc/php/8.0/fpm)

Adjust these paarmeters:

Max memory a script can use



Maximum size for file uploads



Max post request size

A screen shot of a computer

Description automatically generated

Max time (in seconds) a script can run

A black background with green and blue letters

Description automatically generated

Next, create a PHP info file to test

Sudo vim /var/www/apachehustler/info.php



Add this in <?php phpinfo(); ?>

A close-up of a logo

Description automatically generated

To test it, type inside your browser

A blue rectangle with black text

Description automatically generated

Should see the config page now

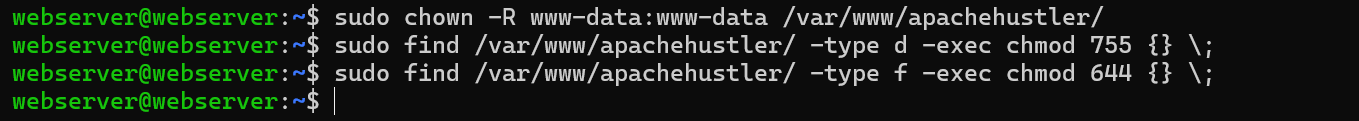


Now that its tested you can remove it first

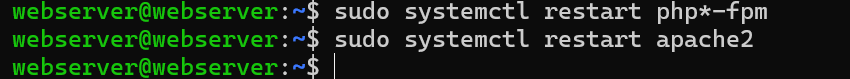
We go to set ownership for our web files folder

Change ownership to Apache user first which is www-data

Then apply rwx permissions for files and folders



Restart services



Now, we can proceed to setup our database

Install mariadb server

A screen shot of a computer

Description automatically generated

Proceed to run security script

A screen shot of a computer screen

Description automatically generated

Since already have root password can press enter for the Enter current password for root.

Switch to unix socket authentication: n

Change root password: n

Remove anonymous users: Y

Disallow root login remotely: Y

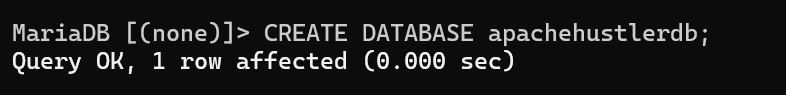
Remove test database and access to it: Y

Reload privilege tables now: Y

Now run sudo mysql -u root -p to connect to the database as root

In the mysql prompt:

Create your database



Create username and password that will be used



If they prompted QUERY OK 0 rows affected it means it is successful.

Grant DB privileges to the created user



Flush privileges to apply changes

A black background with white text

Description automatically generated

Exit then test



In the mysql prompt you can try display the created database

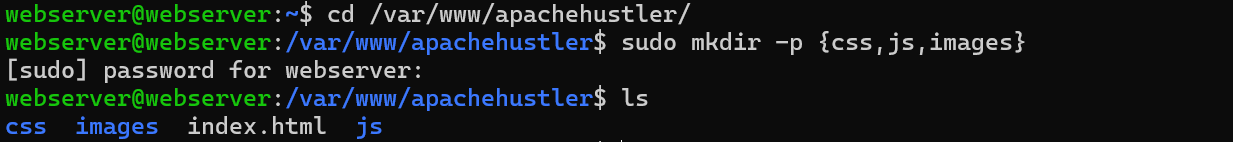
A screen shot of a computer

Description automatically generated

However if you try to run SELECT USER FROM mysql.user you should not be able to do so as by default only root user is able to view from system tables like mysql.user or if you run grant select on mysql.user to your username.

Now that we are done with the basic database setup, we can try to kickstart our website setup.

Cd to your website files folder and create folders for css, javascript and images.



Create another file called index.php inside the same directory and then edit it with content below

<!DOCTYPE html>

<html>

<head>

<title>Apache Hustler</title>

</head>

<body>

<?php

// Test database connection

$conn = mysqli\_connect('localhost', 'your\_username', 'your\_password', 'apachehustler');

if ($conn) {

echo "Database connected successfully!";

}

?>

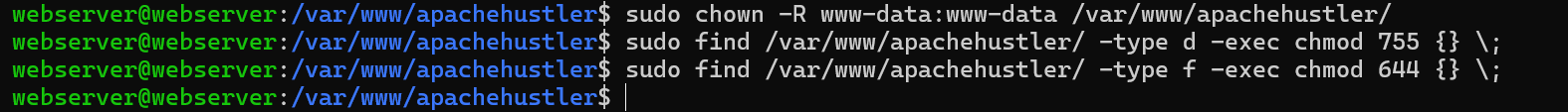
</body>

</html>

A screenshot of a computer

Description automatically generated

Ensure proper permissions applied for all the newly created files



If it displayed as database connected successfully when you access http://your-webserver-domain-name/index.php means it is ok

A close-up of a word

Description automatically generated