LAMP on Debian 12

Apache 2.4 Installation

1. Install apache 2.4

```
$ sudo apt update
$ sudo apt install apache2
```

2. Add one of the two commands to add the user to apache's user group:

```
$ sudo adduser <user> www-data
$ sudo usermod -a -G www-data <user>
```

3. Setup your virtual hosts
4. Create sub folders in `/var/log/apache2` if you setup log files for the virtual hosts in sub folders
5. Install and configure Let's Encrypt Certbot

Settings

- Harden apache
  - change *ServerTokens* and *ServerSignature* in /etc/apache2/conf.d/security.conf
  - add *Require all granted* to your web space, possibly exclude black listed ip addresses, and restrict access to phpmyadmin etc. Put a respective conf file into /etc/apache2/conf.d.
- `MaxRequestedWorkers`: modify /etc/apache2/mods-available/mdm-prefork.conf and restart apache2

```
$ sudo apache2ctl -V | grep MPM
vim /etc/apache2/mods-available/mdm-prefork.conf
  MaxRequestedWorkers 400
  ServerLimit 400
$ sudo service apache2 restart
```

Links

- Multiple servers one IP address
- What is the size limit of a post request?
- Securing Apache and blocking a list of ip addresses
- Apache keeps going down on a Plesk server: server reached MaxRequestWorkers setting

PHP Installation

1. Install packages

```
$ sudo apt update
$ sudo apt install -y curl wget gnupg2 ca-certificates lsb-release apt-transport-https software-properties-common
```

2. Add the SURY repository to your system

```
$ echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" | sudo tee /etc/apt/sources.list.d/sury-php.list
```

3. Import the repository key

```
$ wget -qO- https://packages.sury.org/php/apt.gpg | sudo tee /etc/apt/trusted.gpg.d/sury.gpg
```

4. Install the desired PHP version, where V is the major and v is the minor version number, for example 5.6 or 7.4

```
$ sudo apt update
$ sudo apt install phpV.v
```

5. Enable modules:

```
$ sudo a2enmod ssl
$ sudo a2enmod proxy
$ sudo a2enmod proxy_http
$ sudo a2enmod rewrite
```

Settings

- `php.ini` for production:

```
max_execution_time = 120
max_input_vars = 2000
memory_limit = 512M
post_max_size = 32M
sys_temp_dir = "/tmp"
upload_tmp_dir = "/tmp"
upload_max_filesize = 16M
date.timezone = Asia/Bangkok
```

- `php.ini` for development:

```
output_buffering=off
short_open_tag = On
max_execution_time=120
max_input_time=150
max_input_vars = 2000
memory_limit = 512M
error_reporting=E_ALL & ~E_DEPRECATED & ~E_STRICT
display_errors=On
post_max_size = 32M
```

```
sys_temp_dir = "/tmp"
upload_tmp_dir = "/tmp"
upload_max_filesize=128M
date.timezone=Asia/Bangkok
```

- Check ini files
  - apache2: load a php file with the following content

    ```
    <?php phpinfo();?>
    ```

  - cli:

    ```
    php --ini
    ```

Links

- DEB.SURY.ORG
- php.net - The configuration file
- Fix the apt-key deprecation error in Linux
- How To Specify A Custom php.ini For A Web Site

PHP Extensions

- PHP extensions for Joomla:

  ```
  $ sudo apt install phpV.v-{bz2 curl gd mbstring mysql xml zip bcmath}
  phpV.v-{json}
  ```

- PHP extensions for Wiki:

  ```
  $ sudo apt install phpV.v-{bz2 curl gd mbstring mysql xml zip sqlite3}
  phpV.v-{json}
  ```

- Restart the service with one of the 2 commands below:

  ```
  $ sudo service apache2 restart
  ```

Set or change PHP version

1. Set the desired PHP version for Apache2 and restart the service with one of the 2 commands below:

   ```
   sudo a2dismod phpV.v
   sudo a2enmod phpV.v
   sudo systemctl restart apache2
   sudo service apache2 restart
   ```

2. Check PHP cli Version

```
php -v
```

3. Set the desired PHP version for CLI (interactive):

```
sudo update-alternatives --config php
sudo update-alternatives --config php-config
sudo update-alternatives --config phar
sudo update-alternatives --config phar.phar
sudo update-alternatives --config phpize
```

4. Set the desired PHP version for CLI (manual):

```
sudo update-alternatives --set php /usr/bin/phpV.v
sudo update-alternatives --set php-config /usr/bin/php-configV.v
sudo update-alternatives --set phar /usr/bin/pharV.v
sudo update-alternatives --set phar.phar /usr/bin/phar.pharV.v
sudo update-alternatives --set phpize /usr/bin/phpizeV.v
```

5. Check PHP apache2 Version: call phpinfo(); in a script
6. Once you have installed a required extension, use the below command to verify it

```
php -m | grep -i mysql
```

Links

- How to Switch between Multiple PHP Version on Debian 12/11/10
- How to Install Multiple PHP Version with Apache on Debian 11/10

Xdebug

1. Start the web server and *View page source* of the output of `phpinfo();`
2. Copy the entire page
3. Go to the Xdebug: Installation Wizard, and paste the output inside the text box on the page. It will analyze the output and will recommend the most suited package of Xdebug.
4. Download that package from the output before by clicking on it's name, for example xdebug-3.3.2.tgz
5. Install the pre-requisites for compiling PHP extensions

```
sudo apt install phpV.v-dev autoconf automake
```

6. If you have more than one php version installed you need to Set or change PHP version first
7. Unpack the downloaded file with `tar -xvzf xdebug-3.3.2.tgz` within a temp folder, then change to that folder, run `phpize` and check it's output:

```
cd xdebug-3.3.2
phpize
Configuring for:
```

```
PHP Api Version:         20230831 (8.3)
Zend Module Api No:      20230831
Zend Extension Api No:   420230831
```

8. If it does not, you are using the wrong phpize. Please follow this FAQ entry and skip the next step.
9. Run:

```
./configure
make
```

10. Copy the module to `20230831`, the date-string taken from the output of phpize:

```
sudo cp modules/xdebug.so /usr/lib/php/20230831
```

11. Add the configuration in a new file to `/etc/php/<V.v>/[apache2|cli]/conf.d/80-xdebug.ini` (V.v is the php version, e.g. `8.3`):
    - **Xdebug 2 config**

```
zend_extension = /usr/lib/php/20230831/xdebug.so
xdebug.remote_enable=1
xdebug.profiler_enable=0
xdebug.profiler_enable_trigger=1
xdebug.trace_output_dir=PATH_TO_OUTPUT
xdebug.profiler_output_dir=PATH_TO_OUTPUT
xdebug.gc_stats_output_dir=PATH_TO_OUTPUT
xdebug.remote_log=PATH_TO_LOG/xdebug.log
xdebug.remote_port=9000
xdebug.remote_host=127.0.0.1
```

    - **Xdebug 3 config**

```
zend_extension = /usr/lib/php/20230831/xdebug.so
xdebug.mode=debug,develop
xdebug.start_with_request=trigger
xdebug.start_upon_error=no
xdebug.output_dir=PATH_TO_OUTPUT
xdebug.log=/PATH/TO/LOG/xdebug.log
xdebug.log_level=10
xdebug.var_display_max_children=-1
xdebug.var_display_max_data=-1
xdebug.var_display_max_depth=-1
xdebug.client_port=9003
xdebug.client_host=127.0.0.1
```

12. `PATH_TO_OUTPUT` and `PATH_TO_LOG` must be writable for the php process (apache or cli)
13. Make sure the xdebug configuration is loaded last (e.g., `zend_extension` directive must be below the line for OPcache)
14. Also update php.ini files in adjacent directories (*apache2* and *cli*), as your system might be configured with a separate php.ini file for the web server and command line.
15. Read WSL (Windows Subsystem for Linux) if you are running your LAMP server inside WSL
16. Read Upgrading from Xdebug 2 to 3 if you are upgrading

17. Restart your webserver after configuration changes
18. Create a PHP page with `<?php phpinfo(); ?>`. Load it in a browser and look for the info on the Xdebug module. If you see it next to the Zend logo, you have been successful!
19. Create a PHP page with `<?php xdebug_info(); ?>`. Load it in a browser to verify the Xdebug configuration
20. On the command line, you can also `php -m`. This lists all loaded modules. Xdebug should appear twice there (once under 'PHP Modules' and once under 'Zend Modules')

Links

- Xdebug: Installation
- Xdebug: all settings
- Upgrading from Xdebug 2 to 3

MariaDB 10.11.6 Installation

1. Install

```
$ sudo apt install mariadb-server
```

2. Start MariaDB

```
$ sudo service mariadb start
```

3. Secure the installation following How to Install MariaDB on Debian 12 Bookworm Distribution

```
$ sudo mariadb-secure-installation
```

4. Assign a password to root
5. Dump all databases on the old server

```
$ sudo mysqldump --all-databases > sql_file.sql
```

6. Copy `sql_file.sql` to the new server and remove databases `mysql` and `phpmyadmin` from the dump file, then import with

```
$ sudo mysql < sql_file.sql
```

phpMyAdmin

- phpMyAdmin

WSL (Windows Subsystem for Linux)

- For Windows Subsystem for Linux, create a **Virtual Host** file with document root in */mnt/<drive>/htdocs* or similar, if you need to access it through the Windows file system.

- You should set the apache user to the one who owns the files in the document root, which helps avoiding problems with permissions on the Windows NTFS file system:

```
export APACHE_RUN_USER=<user>
export APACHE_RUN_GROUP=<user>
```

- Restart apache and remove session variables if any:

```
/etc/init.d/apache2 restart
rm /var/lib/php/sessions/*
```

- You can check the current apache user with:

```
ps -ef | egrep '(httpd|apache2|apache)' | grep -v `whoami` | grep -v
root | head -n1 | awk '{print $1}'
```

- Replace the apache default user and group permissions of *www-data* with the one of <user>. run:

```
sudo chown root:<user> /var/lib/phpmyadmin/blowfish_secret.inc.php
sudo chown -R <user>:<user> /var/lib/tmp
sudo chown root:<user> /etc/phpmyadmin/config-db.php
```

- Check permissions of the folder containing the http files according to "Failed to Enumerate Objects in the Container" Windows 10 Error. Most importantly, make sure all files are owned by the same user. Run a (windows) command shell on the windows path of that folder as administrator and run:

```
takeown /F X:\FULL_PATH_TO_FOLDER
takeown /F X:\FULL_PATH_TO_FOLDER /r /d y
icacls X:\FULL_PATH_TO_FOLDER /grant Administrators:F
icacls X:\FULL_PATH_TO_FOLDER /grant Administrators:F /t
```

WSL1 with Xdebug

- WSL1 runs networking in `bridged` mode
- You do not need to set `xdebug.client_host` (Xdebug2: `xdebug.remote_host`)
- Will work with default configuration settings

WSL2 (NAT)

- WSL2 set to NAT networking mode (default)
- Your Linux box runs within a Hyper-V Virtual Machine
- You can access the Windows host through the IP address stored in `/etc/resolv.conf`
- `xdebug.client_host` must be configured to use this IP address. Separate the directive from the xdebug settings by copying it into a separate file to be loaded after the xdebug settings, then run a script to update this IP address before starting or restarting the LAMP server:

```
#!/bin/bash
IP=$(ip route show | grep -i default | awk '{ print $3}')
```

```
INI="/home/bco/conf/xdebug.host.ini"
echo -e "[Xdebug]" > $INI
echo -e "xdebug.client_host=$IP" >> $INI
```

- Check the xdebug logfile with `tail -f /path/to/log/xdebug.log` until this is working properly

Windows host IP address

- Alternatively you could use the Windows host IP address (feasible if it remains the same through re-boots) in which case you need to add a firewall rule to allow access through the xdebug port. Command to run in an elevated PowerShell:

```
New-NetFirewallRule -DisplayName "xdebug" -InterfaceAlias "vEthernet
(WSL (Hyper-V firewall))" -Direction Inbound -Protocol TCP -LocalPort
9003 -Action Allow
```

- Commands to check or remove the firewall rule:

```
Get-NetFirewallRule -DisplayName "xdebug"
Remove-NetFirewallRule -DisplayName "xdebug"
```

WSL2 (mirrored)

- WSL2 set to `mirrored` networking mode
- This networking mode is available for Windows 11 23H2 and later
- To enabled `mirrored` mode, save the following content as file `.wslconfig` into folder `C:\Users\<user>` on your Windows host

```
[wsl2]
networkingMode=mirrored
```

- Your Linux box runs within a Hyper-V Virtual Machine
- You can access the Windows host through the IP address `127.0.0.1`, no firewall rule is necessary
- `xdebug.client_host` must be configured to use this IP address
- Check the xdebug logfile with `tail -f /path/to/log/xdebug.log` until this is working properly
- Note that WSL in `mirrored` mode is less secure and can be directly accessed from your LAN

Links

- How to forward WSL2 port to Windows port and vice versa

SSL for localhost

Ignore invalid certificates

- You can just use the default ssl conf file in /etc/apache2/sites-available which makes use of the snakeoil certificate. Modify DocumentRoot and add Directory permissions.
- Paste this in chrome, enable, and chrome will ignore invalid certificates for localhost:

```
chrome://flags/#allow-insecure-localhost
```

Create certificate for localhost

1. Make a folder to keep your certificate files and change to that folder, for example ~/certs/ssl.
2. Generate RootCA.pem, RootCA.key & RootCA.crt:

```
openssl req -x509 -nodes -new -sha256 -days 1024 -newkey rsa:2048 -
keyout RootCA.key -out RootCA.pem -subj "/C=US/CN=Example-Root-CA"
openssl x509 -outform pem -in RootCA.pem -out RootCA.crt
```

3. Create a file domains.ext that lists all your local domains:

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment,
dataEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = localhost
DNS.2 = localhost.yourdomain.tld
DNS.3 = machine1.yourdomain.tld
DNS.4 = machine2.yourdomain.tld
```

4. Generate localhost.key, localhost.csr, and localhost.crt:

```
openssl req -new -nodes -newkey rsa:2048 -keyout localhost.key -out
localhost.csr -subj "/C=US/ST=YourState/L=YourCity/O=Example-
Certificates/CN=localhost.local"
openssl x509 -req -sha256 -days 1024 -in localhost.csr -CA RootCA.pem -
CAkey RootCA.key -CAcreateserial -extfile domains.ext -out
localhost.crt
```

5. Configure Apache:

```
SSLEngine on
SSLCertificateFile "/home/user/certs/ssl/localhost.crt"
SSLCertificateKeyFile "/home/user/certs/ssl/localhost.key"
```

6. Restart Apache
7. At this point, the site would load with a warning about self-signed certificates. In order to get a green lock, your new local CA has to be added to the trusted Root Certificate Authorities in your OS or browser.
    - For Windows 10 Chrome & Edge: Windows 10 recognizes .crt files, so you can right-click

and open *RootCA.crt*.
    - Select *Install Certificate...*, select *Local Machine*, then select *Trusted Root Certification Authorities* and confirm.
    - You might need to clear cookies and cache for the browser to pick up the certificate from the server
8. If you want to utilitze the certificate for an Endian Firewall, do the following:
    - Rename the files *server.crt*, *server.csr*, and *server.key* in folder /etc/httpd and etc/httpd/cert
    - Copy the newly generate certificate files *localhost.crt*, *localhost.csr*, and *localhost.key* to *server.crt*, *server.csr*, and *server.key* in folder /etc/httpd
    - Copy the newly generate certificate file *localhost.crt* to *server.crt* in folder /etc/httpd/certs and append the parameters from the renamed original *server.crt* file
    - Restart httpd
    - You can check the domain names included in the original certificate:

```
openssl x509 -text < $CERT_FILE
```

Trust localhost certificate on Windows 11

- Run `certmgr.msc`, then select *Trusted Root Certification Authorities*
- Right-click on *Certificates* and select *All Tasks –> Import...*. Import the `RootCA.crt` file and accept all questions with default answer.

Links

- SSL Checker
- Correct steps to add another domain to existing certificate
- Revoking Let’s Encrypt Certificates
- Correct Way to Delete a Certbot SSL Certificate
- Remove a domain in Let’s Encrypt
- How to Install MariaDB on Debian 12 Bookworm Distribution
- How to create an HTTPS certificate for localhost domains (reference)
- Let's Encrypt: Certificates for localhost
- How to Get SSL HTTPS for localhost
- Troubleshooting Apache SSL Certificate Errors
- Correct Way to Delete a Certbot SSL Certificate
- Remove certbot-auto
- Getting Chrome to accept self-signed localhost certificate
- How to create a self-signed certificate with OpenSSL
- How to install an SSL Certificate on Apache

Proxy

* Setup a VirtualHost on your main apache server, which for this example is now called “proxy”. * There needs to be another (regular) VirtualHost file on the backuppc server, which for this example is now called “host”. * The SSL certificates are served from the “proxy” through access to https://sub.domain.tld * The “host” serves an unencrypted site through port 80. This assumes your

local network is secure.

VirtualHost on the "proxy" server

* sub.domain.tld: external domain name with which you access the “host” behind the “proxy” * host.yourdomain.tld: internal domain name of your “host”. You may also choose to have both names the same.

```
<VirtualHost *:80>
    ServerName sub.domain.tld
    Redirect 301 / https://sub.domain.tld
<VirtualHost>
<VirtualHost *:443>
    ServerName sub.domain.tld
    ServerAdmin you@domain.tld
    DocumentRoot /var/www/html/yoursite
    SSLEngine on
    RedirectMatch ^/$ /yourapp/ # use this if backuppc is not the default
app, or if you need to access another app on the same server
    <Location "/yourapp/">
          ProxyPass "http://host.yourdomain.tld/yourapp/"
          ProxyPassReverse "http://host.yourdomain.tld/yourapp/"
          Require all granted
    </Location>
    # add other options such as Files and Directory permissions
    Include /etc/letsencrypt/options-ssl-apache.conf
    SSLCertificateFile /etc/letsencrypt/live/sub.domain.tld/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/sub.domain.tld/privkey.pem
</VirtualHost>
```

Links

- Use apache as a HTTPS to HTTP Proxy
- Reverse Proxy Guide
- Simple Apache reverse proxy example

From:
https://wiki.condrau.com/ - **Bernard's Wiki**

Permanent link:
https://wiki.condrau.com/deb12:lamp

Last update: **2025/11/25 08:08**