LAMPf: Linux / Apache / MariaDB / PHP- FPM- Serverguide

Note: The here explained way of setting up a webserver is very performant plus very slim and basic and may be a very good way if your machine is old and/or your ressources are low and you want to deliver solid, fast and small services based on native distribution- liked builds. But it needs some manual maintanance and a well working, full featured Linux- Distribution, delivering all those Services (Apache, php, fpm, mariadb) as binary build packages. As complexity raises and more services are build up on that server, you may run into high maintainance loads and the need to spend a lot of time to configure more and more services always having the complexity in mind.

For example: if you want to make those Services available on the internet, you should definitively want to make webservices SSL- secured. Getting automatic SSL-Certificates and renew them in a good way, will work if you know how to set it up, but the setup is just another thing to care about.

So nowadays there are solutions to make complexity for that more handy which would be e.g. using Docker or Podman - which would be another, very different approach for large system, delivering a full, extremly powerful service- infrastructure in a pre- build way (it also delivers apache with php-fpm and automatic ssl-acme-challanges out of the box in many packages!).

So if you plan to set up a new fully flagged Server which will have many needed services, don't use this setup. Just after installation go straight into installing and setting up e.g. Docker and go on and use it for all services you can find there.

You can find the new way how to setup a server here: SoHO Serverbasics

So this for me is deprecated as i bought a new server here. I won't spend any time here. If you wann know more about nowadays setups, feel free and write a mail to me, maybe i will deliver docs to this than. HF and GL obel1x

Howto setup a LAMP-Server in 2018 - 2020+

This documentation is about how to set up a LAMP- Server (Linux, Apache, MySql aka MariaDB, PhP) in the current, most stable way for home office usage. The reason for me writing this is, that default Installations of common Distruibutions are often based on an old styled configuration, which is not the way it could be done today, leading to instability and complex configuration and dependencies that make it hard to update components individually.

To get the Differences, this is how my Distro (OpenSuSE) delivers the Packages by default and which disadvanteages it has:

- Apache-Prefork. That way, Apache is one Application with many threads - which is slower, consumes more memory and doesn't scale good
- PHP-Module (mod_php) loaded as Module in the Apache- Server which is not released as stable and may crash the whole Apache- Server on Errors. This module must be compiled to match the Apache- Version, so upgrading Apache means upgrading mod_php
- PHP- Modules, which must be compiled againt the apache-mod_php- module. Updating mod_php means updating all modules.

    - Using Network Connections with overhead to connect to
- MariaDB

And here is what this guide will set up:

- Apache- Event. That way, Apache is one small Apache- Manager- Application which will spawn as many Apache- Servers as needed to handle the incoming connections dynamically. This is stable, as one Error may only crash the one Instance, which will be respawned by the Apache- Manager dynamically
    - Connection to php is done by Proxy- Handling in Apache
- PHP-FPM will also Spawn PHP-Instances dynamically for each script beeing run
    - Using Socket Connections to connect to
- MariaDB

Choosing Installation-Media & Install Basic System

First, starting from Windows, you should make sure to have enough harddisk space free (i recommend at least 60 GB). Then get the installation-medium of the Linux your choice. I will stick to OpenSuSE as to get from OpenSuSE (use Leap 15.1 currently as stable Distro. Tumbleweed may be instable). Follow the instructions to

1. Download the DVD-Image from the Webpage as described there
2. Create the Installation- Media
3. Install the System with standard Desktop- Packages (KDE)
4. Boot into new Linux and Set Up Desktop as you like

Basic System- Scaling thoughts

The most important thing to consider when making performant LAMP is to **not overextend memory-usage** of your System. That means, that the amount of memory used by all Applications, should normally never exeed the system-memory space. If the settings are too high for your setup, the system will start to swap o lot of data, not working fast enough any more. As basic thumb-based Values, you need: **1 GByte Memory for linux Base- System and additionally 1 GByte Memory if you plan to have the graphical Desktop running** (you can run that server in Textmode, which will not consume Memory) **+ at least 1 GByte free** (this will be used by System for filecache)

The remaining Memory should be Split around this Values:

- 1/2 to Mysql
- 1/4 to PhP
- 1/4 to Apache

Those values are only for initial setup. After watching your System some time, you can adjust them to your needs. Mostly, when the system is growing, the Database will need even much more Memory than the Webserver, but that depens on your needs.

This is a very tight setup - having no more space for other Applications. So maybe if possible, spend some more GB and leave them free or dedicated to other things (e.g. In- Memory-DB like redis for special jobs).

Again in short: **Don't use more Memory as your system can deliver, or you will have no fun with it**!

How to tell that your System is set up right

As the System is still usable also with a bad configuration, here is how to check if the Memory is set up right:

Open a terminal and type in „top“ as command. In the 4th line you should see „KiB Mem...“ . The important Values in that line are: XXX free ⇒ if this Value is to low (<90000 is very low), the System has no space left to start new Tasks. This Value should always be higher. 256000 or more is a good Value. XXX buff/cache ⇒ this is the value, that the system has allocated for filebuffering. A low value indicates that it may not performe well. The Value should be around 1024000 or more for best performance.

If those values are both high, you can go and set swappiness to 0, which means that the system will tray to stay in memory as long as it will be possible.

Use „systemctl vm.swappiness = 0“ and set it in /etc/sysctl.conf by adding line „vm.swappiness=0“.

Install MariaDB (MySql)

In OpenSuSE MariaDB is available and working out of the Box. Check with:

```
sudo zypper install mariadb
```

MySQL is tuneable in /etc/my.cnf. You should check the Parameters align with the memory of your machine (see above): Settings for 4 GB Memory (at all), are:

- innodb_buffer_pool_size = 768M
- innodb_log_file_size = 96M
- #Hint: innodb_log_file_size * 2/innodb_buffer_pool_size should be equal 25%)
- max_connections = 24
- join_buffer_size = 12M
- sort_buffer_size = 1M

- readn_rnd_buffer_size = 1M

Memory-Usage will be: innodb_buffer_pool_size + (join_buffer_size + sort_buffer_size + readn_rnd_buffer_size) * max_connections. If your System has more memory, use some tuning script (like MySQLTuner-perl) to see what makes most sense to put the memory to.

For a local setup, you should use Sockets and disable networking. To do this, set

```
socket = /run/mysql/mysql.sock
```

You should than deactivate TCP/IP with „skip-networking“ and comment out the bind-address.

Start Mysql with

```
systemctl start mariadb
```

at the command line as root, it should work now.

To setup passwords, run /usr/bin/mysql_secure_installation as root on the system.

Install Apache

In SuSE 15.X the apache-prefork is installed by default as MPM, which means having one single Apache- Programm in memory. This is not very well scaleable (not multithreaded) and not very stable, as one hangig Request can stop the Server.

In modern setups, apache-event (which is the successor of the apache-worker MPM) is used. This is the most stable and best multithreaded webserver commonly used. If you experience problems with it, you can switch back to apache-worker, which is basically the same.

To switch to that MPM:

- Open the Software Store
- Install apache2-event
- Remove apache2-prefork (if installed)
- Commit the Changes
- in /etc/apache2/server-tuning.conf the module will be configured. Event and Worker is nearby the same. I use the following parameters for the event/worker module:

```
#This Config is for event or worker MPMs.
```

```
#ServerLimit is the maximum number of apache-servers running beside the one controlling server. So 31 will make max. 32
Processes in total.
ServerLimit 31
# http://httpd.apache.org/docs/2.4/mod/mpm_common.html#startservers
StartServers        2
#This should be set to threadsperchild * serverlimit
MaxRequestWorkers  496
# http://httpd.apache.org/docs/2.4/mod/mpm_common.html#minsparethreads
MinSpareThreads     32
MaxSpareThreads     64
# number of worker threads created by each child process
# http://httpd.apache.org/docs/2.4/mod/mpm_common.html#threadsperchild
ThreadsPerChild     16
# maximum number of requests a server process serves
# http://httpd.apache.org/docs/2.4/mod/mpm_common.html#maxrequestsperchild
MaxRequestsPerChild  5000
#unsued Parameters, while not needed or obsolete
#MaxClients         512
#ThreadLimit         16
```

I would suggest to remove any mpm-specific configurations and use only those settings. You can leave the other settings as defined by initial setup.

After that, open yast and go to sysconfig- editor. Search for APACHE_MPM and select event.

Test running apache with

```
systemctl start apache2.service

systemctl status apache2.service
● apache2.service - The Apache Webserver
    Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
    Active: active (running) since Tue 2022-11-22 19:43:17 CET; 1s ag
```

If thats fine, please stop apache once again to configure php-fpm first.

Install PhP-FPM

If installed, remove mod_php (see beneath)! The Module for apache is known to make it slow and instable - here we will set up PhP-FPM, which is much more stable and much faster. Here you can find a good Documentation for changing to php-fpm, but we will extend them a bit.

PhP-FPM is a Server for running the PhP-Instances in a controlled way. It will manage the maximum amount of running instances and take care of errors like hanging scripts. To get it:

- Uninstall mod_php and remove it from apache- starting parameters:

```
sudo zypper remove apache2-mod_php8
sudo a2dismod php8
```

- I would suggest using a newer Version of php-fpm than in the default Repositories. E.g. using the Version of Repository „Apache Modules“. Check out other „Experimental Packages“ in https://software.opensuse.org/package/php7-fpm?search_term=php+fpm
- Either install the new Version with 1-Click-Install there **or** for the default Version, use

```
sudo zypper install php8-fpm
```

- Copy the configuration-files for php-fpm:

```
sudo cp /etc/php8/fpm/php-fpm.conf.default /etc/php8/fpm/php-fpm.conf
sudo cp /etc/php8/fpm/php-fpm.d/www.conf.default /etc/php8/fpm/php-fpm.d/www.conf
```

- Than go to /etc/php8/fpm and briefly check if php-fpm.conf is ok for you
- Explanation: In php-fpm.d directory you need to set up at least one pool. This is one Instance for Apache to speak to.

The „pm“-setting in www.conf controls how much memory will be used at the end. Start with:

- pm = dynamic
- pm.max_children = 120
- pm.start_servers = 12
- pm.min_spare_servers = 6
- pm.max_spare_servers = 18

Using Sockets

Whenever you can - you should use unix sockets instead of TCP/IP, because of less overhead. If you are on the same machine (apache and php-fpm), than you can.

So this here is new for the setup: in „/etc/php8/fpm/php-fpm.d/www.conf“ set

```
listen = /run/php-fpm/php-fpm.sock
listen.owner = wwwrun
listen.group = www
listen.mode = 0660
```

You need to make the file be created by systemd, so create a file /usr/lib/tmpfiles.d/php-fpm.conf and paste this line there:

```
d /run/php-fpm 0700 wwwrun root -
```

PHP Configuration

I do not recommend using php.ini in /etc/php8/fpm, but to put it in /etc/php8/conf.d With that Setup, the whole php-configuration will be the same for cli- and web(f)cgi- php execution. Check to move all php.ini files to conf.d. After that, go through the ini-files in conf.d an see if they fit your needs. Especially each Parameter should only be defined once.

After that, start php-fpm:

```
sudo systemctl start php-fpm
sudo systemctl enable php-fpm
```

and check, if the socket-file has been created.

About PHP- Modules

many modules for PHP are offered in the Distrubution. I **would not recommend using those** - as all php-modules need to be compiled against your php. If you update PHP and your modules are not copiled to that version, they may brake your PHP!

Better use pearl / pecl and install modules with it! Here, i have found no other way, than to search for php-pear and php-pecl in the distribution and use them.

For me, i needed: php8-pear, php8-pecl, php8-devel (for command phpize)

Which can be found in the Repo: https://build.opensuse.org/project/show/devel:languages:php

After that, modules can be installed by e.g. „pecl install imagick“. They also need to be loaded in php.conf. I would make an new config named e.g. /etc/php8/conf.d/pear_pecl.ini and include the modules there. E.g. „extension=imagick.so“

To make pecl/paer alter the ini automagically, use e.g.

```
pear config-set php_ini /etc/php8/conf.d/pear_pecl.ini
pecl config-set php_ini /etc/php8/conf.d/pear_pecl.ini
```

Restart php-fpm for the changes and check the log of php-fpm (usually in /var/log/php-fpm.log) for errors when loading modules.

Tell Apache to use php-fpm

For making Apache use php-fpm as php-server, you use the module „proxy_fcgi“, which should be included in the apache MPM- Package.

Caution: this has noting to do with „mod_fcgi“ or „fastcgi“! You will not need those, as this would execute php itself in the webserver, which we dont' want! „proxy_fcgi“ offers the fcgi- interface and tunnels it to php-fpm via socket or ip-interface. Thus, it will be a small wrapper, not having to manage something as big as php.

- To enable this and all its dependencies, use

```
sudo a2enmod setenvif
sudo a2enmod rewrite
sudo a2enmod proxy
sudo a2enmod proxy_fcgi
```

Now, tell proxy_fcgi to use php:

- Create /etc/apache2/conf.d/mod_proxy_fcgi.conf and add:

```
# Don't use "ProxyPassMatch", while non-ascii-urls will not work!
# This is to forward all PHP to php-fpm
<FilesMatch \.php$>
```

```
  SetHandler "proxy:unix:/run/php-fpm/php-fpm.sock|fcgi://localhost:9000"
</FilesMatch>
DirectoryIndex index.php

# Don't use "Reuse" cause of timeouts and php-fpm manages reuse of php automagically!
# <Proxy fcgi://localhost enablereuse=on max=10>
<Proxy fcgi://localhost>
   #6 Hours = 21600
   #Make this high, as upload will stop after that amount of time
   ProxySet connectiontimeout=30 timeout=21600
</Proxy>

# If the php file doesn't exist, disable the proxy handler.
# This will allow .htaccess rewrite rules to work and
# the client will see the default 404 page of Apache
RewriteCond %{REQUEST_FILENAME} \.php$
RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_URI} !-f
RewriteRule (.*) - [H=text/html]
```

Start and check Apache

Now you can start and enable apache2

```
sudo systemctl start apache2
sudo systemctl enable apache2
```

check if the modules have beend loaded:

```
apache2ctl -M
```

This should include proxy_fcgi_module now.

Create a PHP- Test- File

Create the File /srv/www/htdocs/phpinfo.php (with read-permissions for user wwwrun) and paste this into it:

```
<?php

// Show all information, defaults to INFO_ALL
phpinfo();

?>
```

Now open your Web- Browser and go to: http://localhost/phpinfo.php

This should give you the complete Info of your php-configuration. If something fails, check if the above services are started an/or the logfiles.

If you get Permission denied even if your file is world- readable and the user wwwrun can access the content, in Leap 15.4 there are strict permissions in AppArmor. So check Yast / AppArmor and Check the Protocols. You will most likely find entries that needs to be changed (affects also: executing programs with proc_open() in php). After that, Access should work.

In Production, you should not run a plain http-server, but switch to SSL. Therefore, you can get ssl-certificates from let's encrypt and follow the instructions there to switch to ssl. You need some internet Name like www.myname.com registered for your server to get this (e.g. via Dyndns). The process to make your server visible is something to be explained a bit more, but thats basically what you need to do (official internet-name/DNS- entry and ssl-encryption). as long as you don't want to make the server world-reachable and use it nly fr testing, your are also fine without DNS and SSL, but you should make sure, that your firewall blocks http(80) and https(443)-ports.

Well: You are done. Now its up to you to fill Apache with content. Have fun!

Manage Database with phpMyAdmin

To manage your local Database, it would be nice to have phpMyAdmin installed first (via Package-manager). After that, copy the config.sample.inc.php to config.inc.php under /srv/www/htdocs/phpMyAdmin to use the socket /run/mysql/mysql.sock you specified before for mysql.

You can finish the setup of your phpMyAdmin by visiting http://localhost/phpMyAdmin/index.php

Hint: phpMyAdmin in Opensuse seems to be reduced in functionality. For example generating config.inc.php with http://localhost/phpMyAdmin/setup/ will not work. I would recommend to delete contents of /usr/share/phpMyAdmin/ and download a full version here: https://www.phpmyadmin.net/downloads/

About security of your WEB- Page (Scripts)

There are really a lot of important Documentations about security of your Webserver going in Details for each functionality. They are important - as functions should be set as tight as possible.

But there are more basic security settings that may prevent damage if the functional security is breached. So the basic security might be very important - and here especially the file permissions. One big security-hole that is very common misunderstand at permissions is this one:

If you make a file ONLY readable for the user of the webserver (400) and make that user own that file, you may expect this user to not be able to write to that file. THIS IS WRONG !

Instead, in Linux/Unix the user owning a file can ALWAYs change its permissions. So the user will be able to make it writeable again and write to that file. The ONLY way to prevent the webservers- user form writing to files is not to make this user own that file. So you should make your file owned by root and use groups for controlling the access.

If you need more detailled file permissions, you may have a look at file acls, which are very powerful and can solve permission- restrictions tha ugo may cause.

Installing Eclipse

Get Eclipse with PDT here: https://www.eclipse.org/pdt/

You may download the file, extract the contents (e.g. to ~/eclipse) and run the installer there in userspace (no superuser is required).

Filling Content to your Server

This is a demo to install some small Software to your Server. I will use the github- Project EP3-BS for testing.

Install git-web

This is a nice tool, to view your git-repositorys. Install it, and create a directory named /srv/git and make it writeable to users. Restart your webserver.

You should already be able to go to http://localhost/git/ and see an empty project- Directory.

Download some Project

To get the git-project, open a terminal, change to /srv/git and execute „git clone https://github.com/obel1x/ep3-bs.git".

You should already be set to open the project in eclipse. If you rightclick on the Project, you can add Composer-support to automate the installation of composer-modules while setup of ep3.

Than create the Apache configuration to point to that directory:

```
Alias /ep3 "/srv/git/ep3-bs/public/"
<Directory "/srv/git/ep3-bs/">
require local
Options FollowSymLinks
AllowOverride All
DirectoryIndex index.php
</Directory>
```

After that, follow the instructions in https://github.com/tkrebs/ep3-bs/blob/master/data/docs/install.txt. Remember to use eclipse for installing composer-modules as written above!

As Database you can create a new user (e.g. named ep3) with corresponid database and fill the configuration to fit.

Using Eclipse to work on that Project

In Eclipse now add a project and use GIT smart import from that directory location - it should configure git and eclipse + php the right way and you should be ready to go programming!

Setup Debugging of PHP

To setup Debugging of code, install xdebug for php via pecl:

```
pecl install xdebug
```

Now, also add

```
xdebug.mode=debug
```

Now restart php-fpm and check the php_info() if xdebug is enabled and if Step debugging is active. Otherwise check logs.

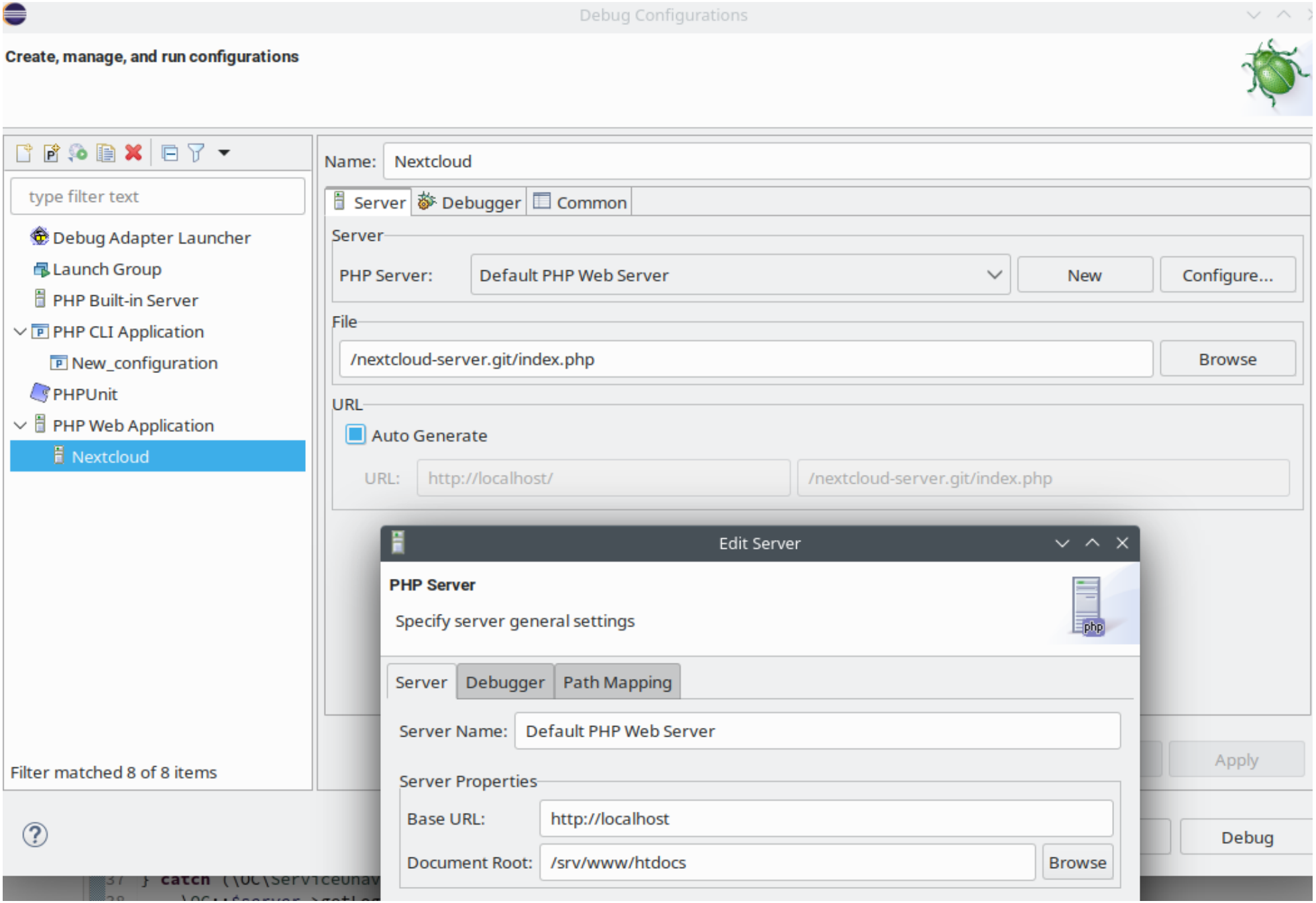
Adjusting timeouts in apache fcgi: Add Timeouts to /etc/apache2/conf.d/mod_proxy_fcgi.conf of apache:

```
...
 Timeout 600
 ProxyTimeout 600
...
```

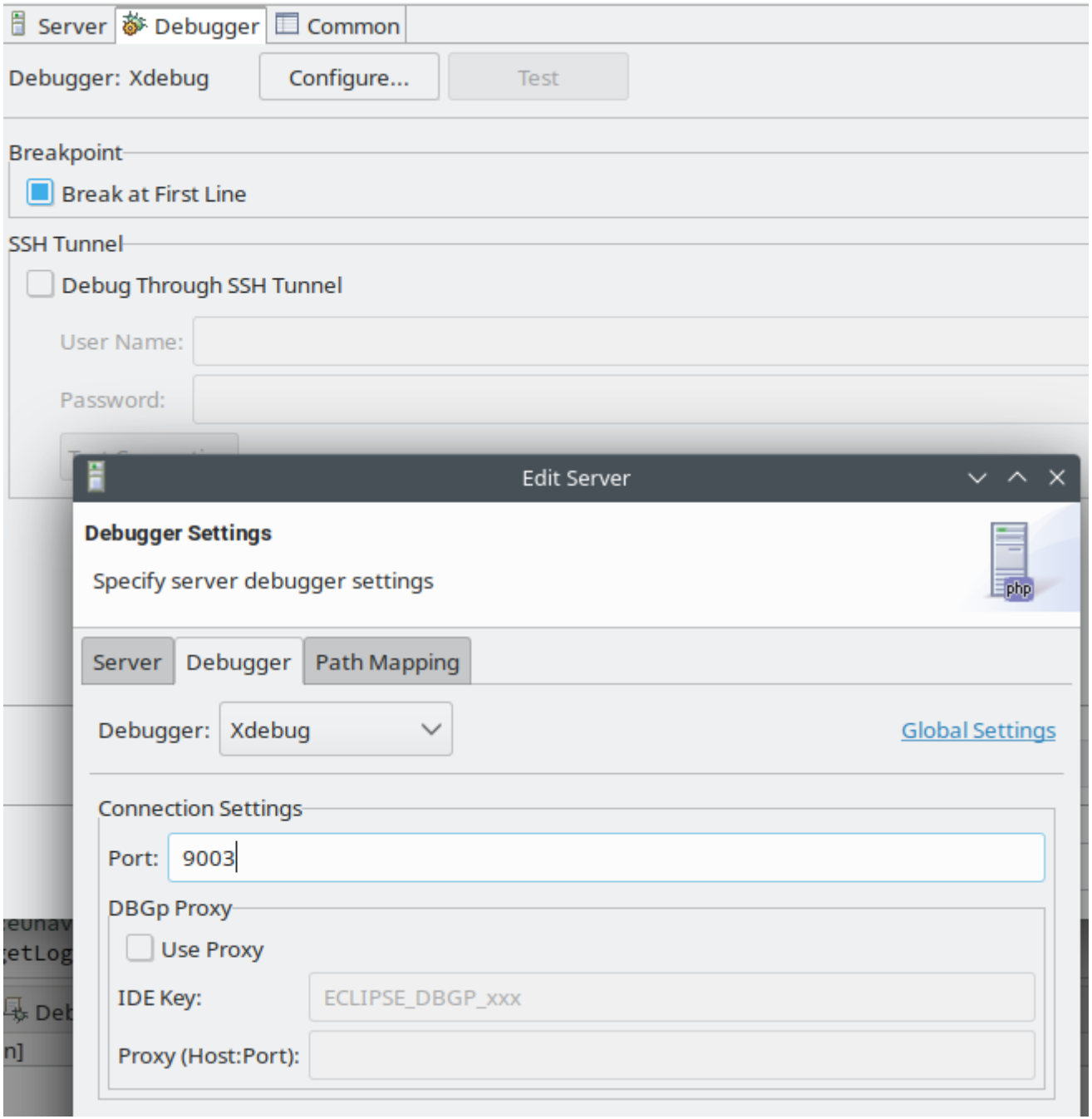
And of course restart apache2.

Setup of Eclipse

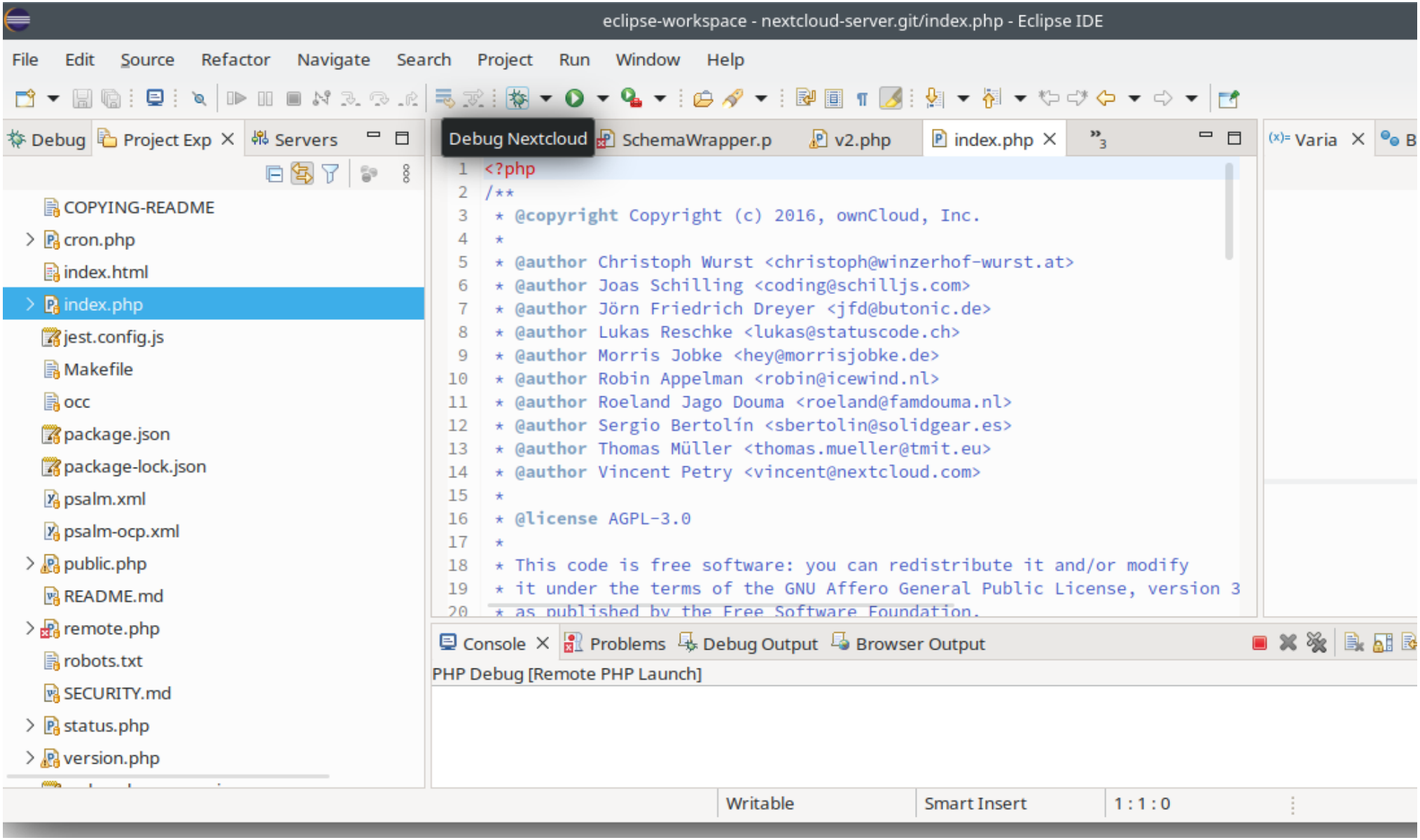
You need to add the PHP- Environement and the Debugger in Eclipse.


Debug Configurations
Create, manage, and run configurations
type filter text
Debug Adapter Launcher
Launch Group
PHP Built-in Server
PHP CLI Application
New_configuration
PHPUnit
PHP Web Application
Nextcloud
Filter matched 8 of 8 items
Name: Nextcloud
Server
Debugger
Common
Server
PHP Server: Default PHP Web Server
New
Configure...
File
/nextcloud-server.git/index.php
Browse
URL
Auto Generate
URL: http://localhost/
/nextcloud-server.git/index.php
Edit Server
PHP Server
Specify server general settings
Server
Debugger
Path Mapping
Server Name: Default PHP Web Server
Server Properties
Base URL: http://localhost
Document Root: /srv/www/htdocs
Browse
Apply
Debug

and


Server
Debugger
Common
Debugger: Xdebug
Configure...
Test
Breakpoint
Break at First Line
SSH Tunnel
Debug Through SSH Tunnel
User Name:
Password:
Edit Server
Debugger Settings
Specify server debugger settings
php
Server
Debugger
Path Mapping
Debugger:
Xdebug
Global Settings
Connection Settings
Port:
9003
DBGp Proxy
Use Proxy
IDE Key:
ECLIPSE_DBGP_xxx
Proxy (Host:Port):

Mind to use the Port phpinfo tells you. Thats it start debugging with e.g. index.php of the project:



That should fire up the Browser, connect php in debugging to eclipse and break at the first line.

You can walk through the code with F5 (dive into), F6 (step over) or just let the work go on with F8.

You are done setting up webserver, php-fpm, mariadb and eclipse plus your new git- repository. hafe fun!