INSTALLATION

LAMPPP STACK INSTALLATION

About LAMPPP

LAMPP stack is a group of open source software used to get web servers up and running. The acronym stands for Linux, Apache, MySQL, and PHP. Since the virtual private server is already running Ubuntu, the linux part is taken care of. Here is how to install the rest.

Step 1: Install Apache

Apache is a free open source software which runs over 50% of the world's web servers.

To install apache, open terminal and type in these commands:

```
sudo apt-get update
sudo apt-get install apache2
```

That's it. To check if Apache is installed, direct wer browser to wer server's IP address (eg. http://12.34.56.789). The page should display the words "It works!" .

Step 2: Install MySQL

MySQL is a powerful database management system used for organizing and retrieving data

To install MySQL, open terminal and type in these commands:

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql php7.0-mysql
```

During the installation, MySQL will ask we to set a root password. If we miss the chance to set the password while the program is installing, it is very easy to set the password later from within the MySQL shell.

Once we have installed MySQL, we should activate it with this command:

```
sudo mysql_install_db
```

Finish up by running the MySQL set up script:

```
sudo /usr/bin/mysql_secure_installation
```

The prompt will ask we for wer current root password.

Type it in.

```
Enter current password for root (enter for none):
```

```
OK, successfully used password, moving on...
```

Then the prompt will ask we if we want to change the root password. Go ahead and choose N and move on to the next steps.

It’s easiest just to say Yes to all the options. At the end, MySQL will reload and implement the new changes.

Once we're done with that we can finish up by installing PHP.

Step 3: Install PHP

PHP is an open source web scripting language that is widely use to build dynamic webpages.

To install PHP, open terminal and type in this command.

```
sudo apt-get install php7.0 libapache2-mod-php7.0 php7.0-mcrypt php7.0-curl
```

Step 4: RESULTS — See PHP on wer Server

Although LAMPP is installed, we can still take a look and see the components online by creating a quick php info page

To set this up, first create a new file:

```
sudo nano /var/www/info.php
```

Add in the following line:

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

Then Save and Exit.

Restart apache so that all of the changes take effect:

```
sudo service apache2 restart
```

DATABASE IMPORT PHPMYADMIN

Standard way is to export the database using “export” tab phpmyadmin. It will work only for limited size databases. In our case size of database exceeds the limit. We have used the following commands.

sudo su
cd /var/lib/mysql/
cp -r <database name> <destination>
scp -r <database name> user@<vpsip>
<password for user in vpsip>
login to vpsip with user name <user>
sudo su
cp -r <database name> /var/lib/mysql
cd /var/lib/mysql
chown -R mysql:mysql <database name>

KEEP THE JOOMLA FILES IN THE DIRECTORY VAR/WWW/HTML

ADD CRONE JOB ON LINUX SERVER WHICH CHECKS FOR UPDATE

Before we start, lets have a look at a few examples of cron job time scheduling and the allowed cron job operators:

```
┌──────────────────── min (0 – 59)
│ ┌────────────────── hour (0 – 23)
│ │ ┌──────────────── day of month (1 – 31)
│ │ │ ┌────────────── month (1 – 12)
│ │ │ │ ┌── day of week (0 – 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat)
│ │ │ │ │
│ │ │ │ │
* * * * *  /path/to/wer/script.sh
```

* * * * * => Run every minute
0 * * * * => Run every Hour
0 0 * * * => Run every midnight
0 0 0 * * => Run every Month
0 0 0 0 * => Run every Weekdays

Step 1: Access our Server via SSH

we need to connect our server via SSH.

Step 2: Create a Cron Job (Scheduled Task)

Once we are connected to our server through SSH, type the following command to open a crontab file.

 # crontab –e

A crontab screen will appear. Now append the entry to schedule a task of wer choice.

For example, run /path/to/command at 6:00 AM, every day, enter:

0 6 * * * /path/to/command

Make wer desired changes and save by pressing “Ctr+X” and then type “Y”.

Note that all times are in UTC, so set wer desired time of execution accordingly.

In our case the we have created a file name server_poll.txt in the home directory in VPS. The content of the server_poll.txt is

* * * * * wget “http://localhost/checkNewNotice.php”

This will trigger checkNewNotice.php in every minute and checkNewNotice.php will compare the currently fetched record with previously stored record(record.txt) and call fcm_notify.php (it uses the fcm server api for sending the push notification to all the registered users.)

FIREBASE CLOUD MESSAGING SERVICE

go to https://firebase.google.com/docs/cloud-messaging/ and sign in there.
create a project
create an app in the project(package name should be same as android studio package name)
download google-service.json from firebase and place that file into the root folder of the android studio project(/app)
include all the nessary dependencies that can be found from FCM webconsole and app should be connected with the FCM through android studio (it will automatically include all the necessary dependencies in the gradle file)

HOW IT WORKS

1) app gets a registration token from google only after first time installation
2) the registration token along with the email id will be stored in mysql database
3) crone job checks for new notification update in a fixed interval(in our case in every minute)
4) the crone job will trigger necessary php script through which FCM server api will be called for sending push notifications to all the registered users (from the mysql database)