

Production installation guide for an HEIG-VD Ubuntu virtual machine

Here you can see the steps I followed to install production ubuntu web server.

Versions

Tool	Version
Ubuntu	20.04.4 LTS
Apache	
Supervisord	
Supervisord	
PHP	8.1.8
Composer	2.3.9
Docker	4.10.1
Docker Compose	
Curl	
Open SSL	

Requirements

Before every thing, you need to follow the deployment guide available on the project README: \ <https://github.com/heig-fablab/fablab-manager/blob/main/README.md>

Usefull commands

Know hostname:

```
hostname -f
```

Apache server commands:

```
sudo systemctl stop apache2
sudo systemctl start apache2
sudo systemctl restart apache2
sudo systemctl reload apache2
```

Apache host configuration commands:

```
sudo a2ensite your_domain.conf
sudo a2dissite your_domain.conf
sudo apache2ctl configtest
```

Production architecture schema

Here is a schema that represents production architecture:

SSH Connection

To start, connect yourself in ssh to the server. To do this, you need to be in HEIG-VD network. \ If you are not, use a VPN and login yourself with your credentials. Here is the command:

```
ssh <user>@<hostname>
```

Update packages

First thing to do is to update and upgrade package managers:

```
sudo apt-get update
sudo apt update
apt list --upgradable
sudo apt update
sudo apt upgrade -y
```

Setup for ssh and Creating a new user

I followed the following tutorial: \ <https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04>

Create user

```
adduser fablab-admin
sudo usermod -aG sudo fablab-admin
```

Enable firewall

```
ufw allow OpenSSH
ufw enable
ufw status
```

Add usefull tools

Open ssl

Verify if already installed:

```
openssl version -a
```

If not, run:

```
sudo apt-get update  
sudo apt-get upgrade
```

Then:

```
sudo apt-get install libssl-dev
```

Curl

To install curl run:

```
sudo apt install curl
```

Project setup

Git installation

First install git:

```
sudo apt-get install git-all
```

Verify if it is installed:

```
git version
```

Clone of GitHub project

Get project git url here (HTTPS):

```
https://github.com/heig-fablab/fablab-manager
```

Create future folder that will contain all apache websites:

```
sudo mkdir /srv/apache2  
cd /srv/apache2
```

Clone project:

```
sudo git clone https://github.com/heig-fablab/fablab-manager.git  
cd fablab-manager/
```

If asked, run this command:

```
git config --global --add safe.directory /var/www/fablab-manager
```

Pull project:

```
sudo git pull
```

PHP installation

PHP is required to run the project. To do this, you can follow this tutorial: \ <https://computingforgeeks.com/how-to-install-php-on-ubuntu-linux-system/>

Install needs to get PHP:

```
sudo apt update
sudo apt install lsb-release ca-certificates apt-transport-https software-properties-common -y
sudo add-apt-repository ppa:ondrej/php
```

Update package manager:

```
sudo apt update
```

Install PHP 8.1:

```
sudo apt install php8.1
```

Verify php is installed:

```
php --version
```

See all extension available:

```
sudo apt install php8.1-<TAB> -> to watch extension
```

Install following extensions:

```
sudo apt install php8.1-{mbstring,xml,bcmath,mysql,curl}
sudo apt-get install php8.1 libapache2-mod-php8.1
```

See all modules installed with following command:

```
php --modules
```

Verify you have installed the following extensions: * Bcmath * CType * Fileinfo * JSON * Mbstring * OpenSSL * PDO * Tokenizer * XML

Composer installation

Verify you are in the project folder:

```
cd /srb/apache2/fablab-manager
```

Install composer:

```
sudo apt-get install composer
```

Install project dependencies:

```
composer install
```

Generate self-signed certificate

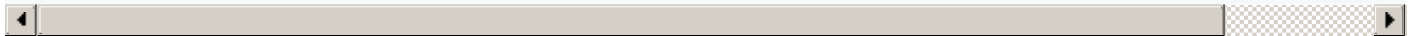
We only need a self-signed certificate in this project because HEIG-VD already provides a valid SSL / TLS certificate. \

First create the folder that will contain the certificate and the key:

```
mkdir /etc/apache2/certificate/  
cd /etc/apache2/certificate/
```

Then generate the certificate and the key:

```
openssl req -new -newkey rsa:4096 -x509 -sha256 -days 365 -nodes -out apache-certificate.crt -keyout a
```



Docker & Docker Compose Installation

I followed these tutorials to install Docker: \ <https://docs.docker.com/engine/install/ubuntu/> \ <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-fr> \ <https://docs.docker.com/compose/install/compose-plugin/#install-using-the-repository>

Upgrade dependencies:

```
sudo apt update  
sudo apt upgrade  
sudo apt-get update
```

Install Docker:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Verify Docker is installed:

```
sudo docker run hello-world
```

Verify Docker Compose version:

```
sudo docker compose version
```

DB MySQL docker compose

Database is running inside a docker container. I created a docker composer containing: * A MySQL database * An Adminer * An MySQL container that perform backups

We need to start this Docker-compose, to do this, first go in mysql folder:

```
cd /srv/apache2/fablab-manager/mysql
```

Prepare Docker-compose with env file:

```
sudo cp .env.example .env
sudo nano .env
```

Change these entries to match the Laravel app configuration:

```
DB_DATABASE=<bd name>
DB_USERNAME=<username | fablab-admin>
DB_PASSWORD=<password>
DB_ADMINER_PORT=8080
DB_VOLUME_PATH=/home/<example-user | fablab-admin>/mysqldata:/var/lib/mysql
DB_DUMP_TARGET=/home/<example-user | fablab-admin>/mysqldump:/var/lib/mysql
```

Run containers:

```
sudo docker compose up -d
```

Setup Laravel project

Copy production .env to create it:

```
sudo cp .env.prod.example .env
```

Modify it with:

```
sudo nano .env
```

The following entries are the one to change (To obtain the values, contact Yves Chevallier):

```
DB_PASSWORD=<password from fablab-admin>
LARAVEL_WEBSOCKETS_SSL_LOCAL_CERT=/etc/apache2/certificate/apache-certificate.crt
LARAVEL_WEBSOCKETS_SSL_LOCAL_PK=/etc/apache2/certificate/apache.key
KEYCLOAK_REALM_PUBLIC_KEY=<realm public key>
```

Add some rights:

```
sudo chown -R root:www-data storage
sudo chown -R fablab-admin:www-data storage
sudo chmod -R g+rwX storage
sudo chmod g+s storage
```

Generate project keys:


```
sudo php artisan key:generate
```

Add job category images to storage for seeding:

```
sudo mkdir storage/app/public/images
```

```
cd job-categories-images
```

```
sudo cp cat1.jpg cat2.jpg cat3.jpg cat4.jpg cat5.jpg cat6.jpg cat7.jpg cat8.jpg cat9.jpg ../storage/ap
```



```
cd ..
```

Run all base migrations:

```
sudo php artisan migrate --seed
```

Clear config:

```
sudo php artisan config:clear
```

Run project:

```
sudo php artisan up
```

Apache

Now we arrive in a big part, we will setup Apache.

Apache installation

First, I followed the following tutorial to install Apache: \ <https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04>

I followed these steps.

1. Update package manager

```
sudo apt update
```

2. Install Apache 2

```
sudo apt install apache2
```

3. Adjust firewall

```
sudo ufw app list
sudo ufw allow 'Apache'
sudo ufw status
```

4. Check web server

```
sudo systemctl status apache2
```

Apache Create Virtual Host file

We create a Virtual Host to keep Apache config intact. \ It also provides the possibility to have more than one configuration at a time. \ We can also have multiple websites running.

1. Add rights to the project folder

```
sudo chown -R $USER:$USER /srv/apache2/fablab-manager
sudo chmod -R 755 /srv/apache2/fablab-manager
```

2. Create configuration file .conf

```
sudo touch /etc/apache2/sites-available/fablab-manager.conf
```

Apache configuration

For SSL / TLS configuration, I followed this tutorial: \ <https://techexpert.tips/apache/enable-https-apache/>

For reverse proxy configuration, I mainly followed these docs: \ <https://ghostscyper.medium.com/running-laravel-websockets-in-production-def368a84d56> \ <https://ghostscyper.medium.com/running-laravel-websockets-in-production-setting-up-websockets-for-a-single-application-9d800c27926> \ https://httpd.apache.org/docs/2.4/en/mod/mod_proxy_wstunnel.html

1. To configure your Apache Virtual Host, open .conf file:

```
sudo nano /etc/apache2/sites-available/fablab-manager.conf
```

TODO: update

2. Add the following configuration


```

<VirtualHost *:80>
    # Redirection to port 443
    RewriteEngine On
    RewriteCond %{HTTPS} !=on
    RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R=301,L]
</VirtualHost>
<VirtualHost *:443>
    # Server information
    ServerAdmin alec.berney@heig-vd.ch
    ServerName tb22-berney.einet.ch
    ServerAlias tb22-berney.heig-vd.ch

    # Inform where the website is stored
    DocumentRoot /srv/apache2/fablab-manager/public

    # Give rights and options to website folder
    <Directory /srv/apache2//fablab-manager/public>
        Require all granted
        AllowOverride All
        Options -Indexes +FollowSymLinks +MultiViews
    </Directory>

    # Setup Proxy for websockets
    ProxyPass "/app/" "ws://127.0.0.1:6000/app/"
    ProxyPass "/apps/" "http://127.0.0.1:6000/apps/"

    # Setup Apache logs
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # Setup SSL / TLS with certificate and key
    SSLEngine on
    SSLCertificateFile /etc/apache2/certificate/apache-certificate.crt
    SSLCertificateKeyFile /etc/apache2/certificate/apache.key
</VirtualHost>

```

3. Reload Apache config

```
sudo systemctl reload apache2
```

4. Open ports

```

sudo ufw allow 443
sudo ufw allow 80

```

TODO

Extern processes

We will use to the **Supervisor** tool to manage external processes to website such as Laravel Websockets and Laravel queue. I chose this tool because he can easily be integrated to a docker later.

Supervisord installation

I followed these tutorial to install Supervisor: \ <http://supervisord.org/installing.html>
<https://www.digitalocean.com/community/tutorials/how-to-install-and-manage-supervisor-on-ubuntu-and-debian-vps>

You just need to run:

```
sudo apt install supervisor
```

You will perhaps have some issue with supervisor when trying to running **supervisorctl** commands, so here is a ref: \ <https://github.com/Supervisor/supervisor/issues/491>

Here is a way to fix it:

```
sudo nano /etc/supervisor/supervisord.conf
```

Remove comments at following line: \ serverurl=unix:///tmp/supervisor.sock ; use a unix:// URL for a unix socket

Remove:

```
; use a unix:// URL for a unix socket
```

Laravel email queue

Create a new file .conf file:

```
sudo nano /etc/supervisor/conf.d/laravel-queue.conf
```

Add these lines:

```
[program:laravel-queue]
process_name=%(program_name)s_%(process_num)02d
command=sudo /usr/bin/php /srv/apache2/fablab-manager/artisan queue:work --daemon
user=fablab-admin
autostart=true
autorestart=true
numprocs=1
startsecs=0
redirect_stderr=true
stdout_logfile=/srv/apache2/fablab-manager/storage/logs/worker.log
```

Laravel Websockets

To add Laravel Websockets program to Supervisor, I followed this doc: \ <https://beyondco.de/docs/laravel-websockets/basic-usage/starting>

Create a new file .conf file:

```
sudo nano /etc/supervisor/conf.d/laravel-websockets.conf
```

Add these lines:

```
[program:laravel-websockets]
process_name=%(program_name)s_%(process_num)02d
command=sudo /usr/bin/php /srv/apache2/fablab-manager/artisan websockets:serve --port=6000
user=fablab-admin
numprocs=1
autostart=true
autorestart=true
```

Reload Supervisor config:

```
sudo supervisorctl reread
sudo supervisorctl update
```

Verify that everything is working:

```
sudo supervisorctl status
```