Doc Anglais

Install Fabmanager app in production with Docker

This README tries to describe all the steps to put a fabmanager app into production on a server, based on a solution using Docker and Docker-compose. We recommend DigitalOcean, but these steps will work on any Docker-compatible cloud provider or local server.

In order to make it work, please use the same directories structure as described in this guide in your fabmanager app folder. You will need to be root through the rest of the setup.

Table of contents

1. [Preliminary steps](#preliminary-steps)
 1.1. setup the server
 1.2. buy a domain name and link it with the droplet
 1.3. connect to the droplet via SSH
 1.4. prepare server
 1.5. setup folders and env file
 1.6. setup nginx file
 1.7. SSL certificate with LetsEncrypt
 1.8. requirements 2. [Install Fabmanager](#install-fabmanager)
 2.1. Add docker-compose.yml file
 2.2. pull images
 2.3. setup database
 2.4. build assets
 2.5. prepare Elasticsearch (search engine)
 2.6. start all services 3. [Generate SSL certificate by Letsencrypt](#generate-ssl-certificate-by-letsencrypt) 4. [Docker utils](#docker-utils) 5. [Update Fabmanager](#update-fabmanager)
 5.1. Steps
 5.2. Good to know

Preliminary steps

setup the server

Go to [DigitalOcean](https://www.digitalocean.com/) and create a Droplet with One-click apps **“Docker on Ubuntu 16.04 LTS”** (Docker and Docker-compose are preinstalled). You need at least 2GB of addressable memory (RAM + swap) to install and use FabManager. We recommend 4 GB RAM for larger communities. Choose a datacenter. Set the hostname as your domain name.

buy a domain name and link it with the server

1. Buy a domain name on [OVH](https://www.ovh.com/fr/) 2. Replace the IP address of the domain with the droplet's IP (you can enable the flexible ip and use it) 3. **Do not** try to access your domain name right away, DNS are not aware of the change yet so **WAIT** and be patient.

connect to the server via SSH

You can already connect to the server with this command: `ssh root@server-ip`. When DNS propagation will be done, you will be able to connect to the server with `ssh root@your-domain-name`.

prepare server

We recommend you to : - ugprade your system - add at least 2GB of swap - verify that you are using a connection via an SSH key. If so, you can set the root passord (for the debug console) and disable password connection. To do this, you can use the following script :

```bash cd /root git clone https://github.com/sleede/lazyscripts.git cd lazyscripts/ chmod a+x prepare-vps.sh ./prepare-vps ```
```

setup folders and env file

Create the config folder: ```bash mkdir -p /apps/fabmanager/config ```

Make a copy of the **docker/env.example** file and use it as a starting point. Set all the environment variables needed by your application. Please refer to the [FabManager README](https://github.com/LaCasemate/fab-manager/blob/master/README.md#environment-configuration) for explanations about those variables.

Then, copy the previously customized `env.example` file as `/apps/fabmanager/config/env`

setup nginx file

Create the nginx folder: ```bash mkdir -p /apps/fabmanager/config/nginx ```

Customize the docker/nginx_with_ssl.conf.example file * Replace **MAIN_DOMAIN** (example: fab-manager.com). * Replace **URL_WITH_PROTOCOL_HTTPS** (example: https://www.fab-manager.com). * Replace **ANOTHER_URL_1**, **ANOTHER_URL_2** (example: .fab-manager.fr)

Use nginx.conf.example if you don't want SSL for your app.

Then, Copy the previously customized `nginx_with_ssl.conf.example` as `/apps/fabmanager/config/nginx/fabmanager.conf`

OR

Copy the previously customized `nginx.conf.example` as `/apps/fabmanager/config/nginx/fabmanager.conf` if you do not want to use ssl (not recommended !).

SSL certificate with LetsEncrypt

FOLLOW THOSE INSTRUCTIONS ONLY IF YOU WANT TO USE SSL.

Let's Encrypt is a new Certificate Authority that is free, automated, and open. Let’s Encrypt certificates expire after 90 days, so automation of renewing your certificates is important. Here is the setup for a systemd timer and service to renew the certificates and reboot the app Docker container:

```bash mkdir -p /apps/fabmanager/config/nginx/ssl ``` Run `openssl dhparam -out dhparam.pem 4096` in the folder /apps/fabmanager/config/nginx/ssl (generate dhparam.pem file) ```bash mkdir -p /apps/fabmanager/letsencrypt/config/ ``` Copy the previously customized `webroot.ini.example` as `/appsfabmanager/letsencrypt/config/webroot.ini` ```bash mkdir -p /apps/fabmanager/letsencrypt/etc/webrootauth ```

Run `docker pull quay.io/letsencrypt/letsencrypt:latest`

Create file (with sudo) /etc/systemd/system/letsencrypt.service and paste the following configuration into it:

```bash [Unit] Description=letsencrypt cert update oneshot Requires=docker.service

[Service] Type=oneshot ExecStart=/usr/bin/docker run –rm –name letsencrypt -v “/apps/fabmanager/log:/var/log/letsencrypt” -v “/apps/fabmanager/letsencrypt/etc:/etc/letsencrypt” -v “/apps/fabmanager/letsencrypt/config:/letsencrypt-config” quay.io/letsencrypt/letsencrypt:latest -c “/letsencrypt-config/webroot.ini” certonly ExecStartPost=-/usr/bin/docker restart fabmanager_nginx_1

```

Create file (with sudo) /etc/systemd/system/letsencrypt.timer and paste the following configuration into it: ```bash [Unit] Description=letsencrypt oneshot timer Requires=docker.service

[Timer] OnCalendar=*-*-1 06:00:00 Persistent=true Unit=letsencrypt.service

[Install] WantedBy=timers.target ```

That's all for the moment. Keep on with the installation, we'll complete that part after deployment in the [Generate SSL certificate by Letsencrypt](#generate-ssl-cert-letsencrypt).

Requirements

Verify that Docker and Docker-composer are installed : (This is normally the case if you used a pre-configured image.)

```bash docker info docker-compose -v ```

Otherwise, you can install docker to ubuntu with the following instructions : https://docs.docker.com/engine/installation/linux/ubuntu/#install-using-the-repository

To install docker-compose :

```bash curl -L https://github.com/docker/compose/releases/download/1.13.0/docker-compose-`uname -s`-`uname -m` > ./docker-compose sudo mkdir -p /opt/bin sudo mv docker-compose /opt/bin/ sudo chmod +x /opt/bin/docker-compose ```

Install Fabmanager

Add docker-compose.yml file

Copy docker-compose.yml to your app folder `/apps/fabmanager`. The docker-compose commands must be launched from the folder `/apps/fabmanager`.

pull images

```bash docker-compose pull ```

setup database

```bash docker-compose run –rm fabmanager bundle exec rake db:create # create the database docker-compose run –rm fabmanager bundle exec rake db:migrate # run all the migrations docker-compose run –rm -e ADMIN_EMAIL=xxx ADMIN_PASSWORD=xxx fabmanager bundle exec rake db:seed # seed the database ```

build assets

`docker-compose run –rm fabmanager bundle exec rake assets:precompile`

prepare Elasticsearch (search engine)

`docker-compose run –rm fabmanager bundle exec rake fablab:es_build_stats`
```

start all services

`docker-compose up -d`

Generate SSL certificate by Letsencrypt

Important: app must be run on http before starting letsencrypt

Start letsencrypt service : ```bash sudo systemctl start letsencrypt.service ```

If the certificate was successfully generated then update the nginx configuration file and activate the ssl port and certificate editing the file `/apps/fabmanager/config/nginx/fabmanager.conf`.

Remove your app container and run your app to apply the changes running the following commands: ```bash docker-compose down docker-compose up -d ```

Finally, if everything is ok, start letsencrypt timer to update the certificate every 1st of the month :

```bash sudo systemctl enable letsencrypt.timer sudo systemctl start letsencrypt.timer (check) sudo systemctl list-timers ```

Docker utils with docker-compose

Restart app

`docker-compose restart fabmanager`

Remove app

`docker-compose down fabmanager`

Restart all containers

`docker-compose restart`

Remove all containers

`docker-compose down`

Start all containers

`docker-compose up -d`

Open a bash in the app context

`docker-compose run –rm fabmanager bash`

Show services status

`docker-compose ps`

Restart nginx container

`docker-compose restart nginx`
```

Example of command passing env variables

docker-compose run –rm -e ADMIN_EMAIL=xxx ADMIN_PASSWORD=xxx fabmanager bundle exec rake db:seed

update Fabmanager

This procedure updates fabmanager to the most recent version by default.

Steps

When a new version is available, this is how to update fabmanager app in a production environment, using docker-compose :

1. go to your app folder

```
`cd /apps/fabmanager`
```

2. pull last docker images

```
`docker-compose pull`
```

3. stop the app

```
`docker-compose stop fabmanager`
```

4. remove old assets

```
`rm -Rf public/assets/`
```

5. compile new assets

```
`docker-compose run --rm fabmanager bundle exec rake assets:precompile`
```

6. run specific commands

- *Do not forget **to check if there are commands to run for your upgrade. Those commands are always specified in the [CHANGELOG](https://github.com/LaCasemate/fab-manager/blob/master/CHANGELOG.md) and prefixed by** [TODO DEPLOY]**.

They are also present in the [releases page](https://github.com/LaCasemate/fab-manager/releases).

Those commands execute specific tasks and have to be run by hand.

7. restart all containers

```
```bash
  docker-compose down
  docker-compose up -d
```
```

You can check that all containers are running with `docker ps`.

Good to know

Is it possible to update several versions at the same time ?

Yes, indeed. It's the default behaviour as `docker-compose pull` command will fetch the latest versions of the docker images. Be sure to run all the specific commands listed in the [CHANGELOG](https://github.com/LaCasemate/fab-manager/blob/master/CHANGELOG.md) between your actual and the new version in sequential order. (Example: to update from 2.4.0 to 2.4.3, you will run the specific commands for the 2.4.1, then for the 2.4.2 and then for the 2.4.3).