Check-List Manager

Dossier d'exploitation

Version 1.0

Auteur
LE MAGOROU Jean-Martial

Nothing-consulting
www.nothing.net

55 rue du Faubourg St-Honoré – 75 008 Paris – jmlm74@gmail.com

SARL au capital de 000,00 € enregistrée au RCS de Paris – SIREN 999 999 999 – Code APE : 6202A

TABLE DES MATIÈRES

1 - Versions

Auteur	Date	Description	Version
JMLM	11/11/2020	Création du document	V 0.1
JMLM	12/11/2020	Fin + correction	V 1.0

2 - Introduction

2.1 - Objet du document

Le présent document constitue le volet “dossier d’exploitation” de la réponse au P13 du parcours Développeur d’Applications Python d’OPENCLASSROOMS.

Ce document a pour objectif de présenter les différentes informations techniques permettant de comprendre et de refaire la mise en production de l’application CheckListMgr. Il doit permettre à toute personne ayant le bagage technique suffisant de maintenir, arrêter et redémarrer l’application

2.2 - Références

Pour de plus amples informations, se référer :

1. **P13 - Dossier de conception fonctionnelle** : Dossier de conception fonctionnelle de l’application
2. **P13 - Dossier de conception technique** : Dossier de conception technique de l’application

3 - Pré-requis

3.1 - Système

3.1.1 - Serveur

Le serveur hébergeant l’application est serveur VPS de la plate-forme Digital-Ocean (Offre droplets). Ce serveur est sous le système d ‘exploitation LINUX/DEBIAN. Ce serveur a été installé via une installation automatisée fournie par Digital-Ocean.

L’application sera accessible via l’adresse internet checklistmgr.jmhayons74.fr.

L’application sera sous un conteneur Docker. Cela permet entre autre :

- Une gestion des mises à jour et du déploiement simplifiée
- Une administration de l’application simplifiée et standardisée

3.1.1.1 - Caractéristiques techniques

Le serveur est un serveur Debian 10 (Buster) installé depuis une installation Digital Ocean. Les mises à jour systèmes sont automatiquement installées et un mail est automatiquement envoyé à chaque mise à jour. Les versions des images docker sont des images standards et officielles. Les modifications qui ont pu y être apportées l’ont été par mes seuls soins.

Les éventuelles mises à jours ainsi que le déploiement de l'application sont traitées plus loin dans ce document.

Un schéma de l'architecture technique est situé en annexe et permet d'avoir une vision globale des services de l'application et de leurs liens.

3.1.2 - Serveur de Base de données

Le serveur de base de données est un serveur PostgreSQL dans un conteneur DOCKER. Le conteneur utilisé est directement fourni par PosgreSQL directement. Aucune modification de configuration n'a été apportée Seul l'ajout de la base de données CheckListMgr est à effectuer (cf procédure ci-dessous)

3.1.3 - Serveur Web

Le serveur WEB est un serveur Nginx lui aussi dans un conteur DOCKER standard. Le conteneur utilisé est lui aussi fourni par l'éditeur soir NGINX directement. Là encore, aucune modification de configuration n'a été effectuée sauf la déclaration du site checklistmgr et le SSL afin de protéger les échanges clients/serveur (cf procédure ci-dessous)

3.1.4 - Serveur applicatif

Le serveur applicatif est fourni par Gunicorn. Là encore, c'est un conteneur DOCKER fourni par l'éditeur. Toutefois, des modifications ont été apportées à la configuration afin de prendre en compte l'application. Ce nouveau conteneur est hébergé sur la plate-forme DOCKER-HUB et est automatiquement mis à jour à chaque modification validée de l'application. Les procédures de génération du conteneur ainsi que la mise à jour automatique du serveur de production sont expliquées plus loin dans le document.

3.2 - Web-services

Les web services utilisés par l'application sont au nombre de 3 :
- OPENSTREET MAP –> Géolocalisation
- OPENWEATHER MAP –> Météo
- MAILGUN –> Envoi des mails

Les connexions utilisées sont des connexions directes en HTTPS soit du client vers le serveur tiers (OPENSTREET MAP et OPENWEATHER MAP) soit du serveur vers le serveur tiers (MAILGUN). Ce type de connexion ne demande aucun paramétrage particulier. Aucune clé/abonnement ou autre n'est nécessaire pour les 2 premières. Pour la dernière (MAILGUN) la clé est directement mise dans les variables d'environnement

3.3 - Autres Ressources

La clé SSL permettant de sécuriser les échanges entre les utilisateurs et le serveur est fournie par zeroSSL. Ce fournisseur permet d'obtenir une clé gratuitement valable 3 mois facilement

renouvelable.

Les ressources utilisées pour le développement sont listées dans la page « Crédits » du site.

4 - Procédure de déploiement et mise à jour

4.1 - Arborescence de l'application

```
checklistmgr
├── app_checklist
│   ├── migrations
│   ├── static
│   │   └── app_checklist
│   │       ├── css
│   │       ├── img
│   │       └── js
│   ├── templates
│   │   └── app_checklist
│   └── test
├── app_create_chklst
│   ├── migrations
│   ├── static
│   │   └── app_create_chklst
│   │       ├── css
│   │       ├── img
│   │       └── js
│   ├── templates
│   │   └── app_create_chklst
│   │       ├── dialogboxes
│   │       └── partials
│   └── test
├── app_home
│   ├── migrations
│   │   └── __pycache__
│   ├── __pycache__
│   ├── static
│   │   └── app_home
│   │       ├── css
│   │       ├── img
│   │       └── js
│   ├── templates
│   │   └── app_home
│   │       └── partials
│   └── test
│       └── __pycache__
├── app_input_chklst
│   ├── migrations
│   │   └── __pycache__
│   ├── __pycache__
│   ├── static
│   │   └── app_input_chklst
│   │       ├── css
│   │       ├── img
```

```
–   │   │       └── js
–   │   ├── templates
–   │   │   └── app_input_chklst
–   │   │       ├── dialogboxes
–   │   │       └── partials
–   │   └── test
–   │       └── __pycache__
–   ├── app_user
–   │   ├── migrations
–   │   │   └── __pycache__
–   │   ├── __pycache__
–   │   ├── static
–   │   │   └── app_user
–   │   │       ├── css
–   │   │       ├── img
–   │   │       └── js
–   │   ├── templates
–   │   │   └── app_user
–   │   │       ├── dialogboxes
–   │   │       └── registration
–   │   └── test
–   │       └── __pycache__
–   ├── app_utilities
–   │   ├── migrations
–   │   │   └── __pycache__
–   │   ├── __pycache__
–   │   ├── static
–   │   │   └── app_utilities
–   │   │       ├── css
–   │   │       ├── img
–   │   │       └── js
–   │   ├── templates
–   │   │   └── app_utilities
–   │   ├── templatetags
–   │   │   └── __pycache__
–   │   └── test
–   ├── checklistmgr
–   │   └── __pycache__
–   ├── media
–   │   ├── checklists
–   │   │   └── 2020
–   │   │       ├── 10
–   │   │       └── 11
–   │   ├── images
–   │   └── photos
–   │       └── 2020
–   │           └── 10
–   ├── __pycache__
–   ├── static
–   │   ├── css
–   │   ├── img
–   │   └── js
–   └── templates
–       ├── errors
–       └── partials
```

Cette arborescence peut évoluer suivant l'ajout de fonctionnalités.

4.2 - Déploiement ou mise à jour de l'application

Le déploiement de l'application se fait suivant la procédure suivante :

ATTENTION :
- Un utilisateur checklistmgr ou autre doit être créé pour contenir les fichiers. Il ne doit toutefois pas pouvoir se connecter pour des raisons de sécurité.
- A partir de maintenant **toutes** les commandes doivent se faire sous le compte root soit par une connexion directe (su) soit par sudo et dans le repertoire de l'utilisateur précédement crée (ex : /*home*/checklistmgr)

L'installation du serveur est donc une installation minimum « Digital Ocean) avec en plus les paquets suivants :
- sudo
- net-tools
- openssh-server
- unattended-upgrade
- python3-pip
- docker
- docker-compose
- postgresql-client
- msmtp

Pour ajouter ces paquets (sauf docker*), taper la commande suivante : ***apt-get -y install nom_du_paquet***

ATTENTION : *pour les paquets docker et docker-compose la toute dernière version a été installée. Pour les installer, se référer à la documentation suivante :*
- Docker : https://docs.docker.com/engine/install/debian/
- Docker-compose : https://docs.docker.com/compose/install/

Le ficher */etc/environment* doit être mis a jour.Cela permet de mettre des paramètres de sécurité en variable d'environnement du serveur et de ne pas les mettre dans un fichier source qui pourrait se retrouver sur les différents repositories et donc à la vue de tout le monde. Un exemple est mis en annexe. Les variables doivent bien entendu être modifiées.
ATTENTION : Afin de prendre en compte le fichier */etc/environment* sur un serveur DEBIAN il faut rajouter la ligne suivante au fichier /etc/pam.d/sudo
session required pam_env.so readenv=1 user_readenv=0

Afin de pouvoir envoyer les mails, le paquet MSMTP a été installé. Se référer à la documentation pour l'envoi de mails via les scripts shell ou autre pour la mise en place d'un fichier SMTP.

Tout les reste (serveur web, connexion base de données…) est directement mis dans le conteneur Docker et ne nécessite donc pas d'installation lourde sur le serveur

Toutefois, la base de données ainsi que les volumes Docker nécessaires au serveur web (fichiers statiques et photos/images/pdf…) nécessitent une première initialisation.

Un paquet python est nécessaire pour la mise en place du script de monitoring des services : psutil. Pour l'installer : ***pip3 install psutil***

4.2.1 - Initialisation de la base de données et des volumes Docker

4.2.1.1 - Volumes

Il y a 3 volumes Docker à initialiser :
- 2 volumes pour le serveur web : 1 volume pour les fichiers statiques et 1 volume pour les fichiers « medias » (photos, avatars… et checklists en PDF).
- 1 volume pour la base de données.

docker volume create db_data
docker volume create media_data
docker volume create static_data

4.2.1.2 - Base de données

– Créer un ficher pg_env (exemple en annexe) qui va contenir les paramètre nécessaires au démarrage et à l'initialisation de la base.

– Initialiser un container Postgresql standard tout en initialisant la base dans le volume créé :
docker run --name db2 -h db2 --env-file ./pg_env -d -v db_data:/var/lib/postgresql/data/pgdata -p 5432:5432 postgres:12

VERIFICATION : La commande ***docker ps*** doit renvoyer quelque chose qui ressemble à cela :
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
aefe256e4376 postgres:12 "docker-entrypoint.s…" 4 seconds ago Up 3 seconds 0.0.0.0:5432->5432/tcp db2

– Il faut ensuite initialiser la base de données et l'utilisateur qui sera utilisé par l'application en tapant les commandes suivantes :
psql -Upostgres -h127.0.0.1 –> mot de passe du fichier pg_env
create database checklistmgr;
create user UTILISATEUR *with createdb encrypted password* 'MotdePasse*';* –> utilisateur et mot de passe du fichier */etc*/enviroment
grant all privileges on database checklistmgr to UTILISATEUR;
alter role UTILISATEUR set client_encoding to 'utf_8';
alter role UTILISATEUR set default_transaction_isolation TO 'read committed';
alter role UTILISATEUR set timezone TO 'UTC';
exit ;

Il faut alors restorer la base de données pour au moins récupérer les langages et les traductions. Tout d'abord rapatrier un dump de la base (via ssh ou autre) par exemple ***/var/tmp/cheklistmgr.dmp*** puis la restorer par la commande :
pg_restore -U*UTILISATEUR *-dchecklistmgr -h127.0.0.1 -c /var/tmp/checklistmgr.dmp —> Utilisateur et mot de passe stipulés plus haut –> ne pas faire attention aux warnings qui ne concernent que les contraintes non encore créées.

VERIFICATION : La commande suivante doit vous afficher une liste de tables :
psql -U*UTILISATEUR *-dchecklistmgr -h127.0.0.1 -c "\dt"

Arrêter le conteneur DOCKER mini de la base de données :
d*ocker stop db2*

4.2.2 - Premier démarrage et fin de déploiement

Il faut dans un premier temps terminer la configuration de NGINX (adresse et nom du serveur + clés SSL). Il faut donc modifier le fichier ***www.conf*** se trouvant dans le répertoire ***$NGINX_CONFDIR*** (positionné dans ***/etc/environment***).

Bien positionner les valeurs des fichiers SSL (key et crt) et mettre les fichiers a l'endroit ou ils ont été stipulés dans le fichier de configuration. Attention, il faut bien faire attention à ce que cet endroit soit bien accessible par le conteneur.

Récupérer le fichier ***docker-compose.yml***. Le mettre dans un répertoire bien précis. Se positionner dans le répertoire et taper la commande : ***docker-compose up*** –> Cela affiche TOUS les messages. Au premier lancement, les images DOCKER sont dans un 1er temps, téléchargées et décompressées puis exécutées. Vous pouvez voir les messages d'initialisation. Pour arrêter les serveurs taper Ctrl-C.

Enfin pour lancer les serveurs en mode service taper la commande : ***docker-compose up -d***.

Le conteneur NGINX étant connu pour ne pas redémarrer seul malgré le paramêtre *restart always…*, le mode de démarrage automatique au reboot du serveur est le suivant : Aller dans la crontab de root et y ajouter la ligne suivante :
@reboot (sleep 30s;cd /home/checklistmgr/;/usr/local/bin/docker-compose up -d)&

4.2.3 - Script de déploiement/mise à jour.

Le même script peut être utilisé pour le déploiement et les mises à jour.

– Dans un 1er temps, le script arrête l'application CheckListMgr et met à la place la page d'indisponibilité.
- Ensuite le script supprime tous les conteneurs utilisés par l'application puis récupère depuis le repository de Docker (Docker-Hub)les dernières versions des conteneurs.
- Enfin le script retire la page d'indisponibilité puis redémarre l'application CheckListMgr.

Le script dure actuellement entre 3 et 4 minutes.

4.2.4 - Déploiement des fichiers

Comme dit ci-dessus : AUCUNE manipulation spéciale n'est nécessaire à la mise à jour normale de l'application. Une mise à jour hebdomadaire est effectuée. Cette mise à jour permet aussi de vider les éventuels caches ou autres qui se seraient créés durant la semaine. En cas de nécessité (correction de bugs par exemple), pour lancer manuellement la mise à jour :

- Se connecter au serveur et taper la commande : ***/home/checklistmgr/scripts/maj.sh***
Le programme de mise à jour dure environ 3 à 4 mn et se trouve en annexe.

4.2.5 - Variables d'environnement

Voici les variables d'environnement pour la bonne exécution de l'application CheckListMgr ainsi que la génération des conteneurs :

Nom	Description et valeur
DJANGO_SETTINGS_MODULE	Positionne le fichier settings.py de production **Valeur :** prod_settings.py
SECRET_KEY	Nécessaire à la sécurité des échanges client/serveur **Valeur :** une suite aléatoire de 50 caractères
MAILGUN_APIKEY	Utilisée pour envoi mails monitoring espace disque Clé appartenant à Pharos Consulting
PG_PSW	Mot de passe de l'utilisateur POSTGRES (base de données)
DATABASE_NAME	Nom de la base de données **Valeur :** checklistmgr
DATABASE_USER	Utilisateur base de données ayant les droits sur la base checklistmgr. Utilisateur spécifié à la création de la base de données.
DATABASE_PSW	Mot de passe de l'utilisateur ci-dessus.
DATABASE_HOST	Nom du serveur de base de données. **valeur :** db
NGINX_CONFDIR	Répertoire de configuration de nginx. **Valeur :** /home/checklistmgr/config/nginx/conf.d

Ces variables sont initialisées dans le fichier /etc/environnement .

4.2.6 - Vérifications

La commande sudo docker ps doit vous renvoyer une réponse similaire. La ligne concernant le do-agent est pour la supervision (cf supervision)

```
CONTAINER ID        IMAGE                              COMMAND                  CREATED             STATUS              PORTS                                      NAMES
b3b7ba65e36f        nginx:latest                       "/docker-entrypoint.…"   3 hours ago         Up 3 hours          0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp   nginx
42e5e9b2c1ce        jmlm74/checklistmgr:latest         "bash -c '/home/jmlm…"   3 hours ago         Up 3 hours          0.0.0.0:8000->8000/tcp                     gunicorn
8bc41ae24788        postgres:12                        "docker-entrypoint.s…"   3 hours ago         Up 3 hours          0.0.0.0:5432->5432/tcp                     db
bae2b833f734        digitalocean/do-agent:stable       "/usr/bin/dumb-init …"   18 hours ago        Up 18 hours                                                    do-agent
```

La commande suivante : ***docker exec -ti gunicorn /home/jmlm/env/bin/python3 /home/jmlm/checklistmgr/manage.py test*** va exécuter une série de tests qui doivent se passer sans erreur.

Enfin, vous devez aussi pouvoir vous connecter au site.

4.2.7 - Mise en indisponibilité et inverse manuellement

Il est possible de mettre le site en indisponibilité (page prévue à cet effet).
Pour cela 2 scripts ont été réalisés.

Pour mettre en indisponibilité : **/home/checklistmgr/script*s/indispo.sh***

Pour remettre en disponibilité : **/home/checklistmgr/script*s/dispo.sh***

Le fait de lancer plusieurs fois l'une ou l'autre des commandes ne gêne en rien !

A noter que le « site de maintenance » est un conteneur Docker, ce qui évite toute configuration autre sur le serveur. La page d'attente est dans le fichier
/home/checklistmgr/sitedattente/index.html

5 - *SUPERVISION*

5.1 - Supervision par la plate-forme DIGITAL-OCEAN

L'hébergeur Digital-Ocean dispose d'une supervision intégrée. Cette supervision nécessite l'installation d'un logiciel sur le serveur. Une version « dockerisée » existant, c'est celle-ci qui a été choisie. Pour la lancer, il suffit de taper la commande suivante : ***`docker run -v /proc:/host/proc:ro -v /sys:/host/sys:ro digitalocean/do-agent:stable`***

Cette supervision contrôle touts les métriques du serveur CPU, RAM, DISQUE… et permet d'obtenir des graphiques de ces métriques. Afin de relancer cette supervision au redémarrage du serveur, une ligne a été ajoutée dans la crontab de root (cf crontab en annexe).

Il existe aussi un monitoring permettant d'alerter l'administrateur par mail en cas de dépassement de seuil. Ces seuils ont les mêmes métriques que la supervision et ne surveillent donc pas que les services prévus soient bien disponibles.

5.2 - Monitoring serveurs

Digital-Ocean ne gérant pas les alertes concernant le bon fonctionnement des services, une procédure permettant cela a été mise en place :
Elle se trouve dans le répertoire ***/home/checklistmgr/scripts*** et se nomme monitor.py. Cette procédure est lancée par la crontab de root toutes les 20 mn et envoie un mail si un des services n'est pas disponible.

6 - DIVERS

6.1 - Sauvegarde et restauration

Une sauvegarde est effectuée tous les matins à 04:30 par le script suivant :

/home/checklistmgr/scripts/backup_db.sh Ce script (en annexe) fait une sauvegarde de la base dans un fichier situé dans ***/home/checklistmgr/backup*** et dont le nom est daté : backup-YYY-MM-DD.dmp. À noter qu'une commande lancée par crontab supprime tous les fichiers de plus de 30 jours dans ce répertoire.

quelconque.

6.2 - Procédure de démarrage / arrêt

L'arrêt du serveur n'est normalement jamais nécessaire. Toutefois, le tableau de bord Digital-Ocean permet d'arrêter ou de redémarrer le serveur. Le redémarrage de l'application est automatique grâce à une ligne de la crontab : ***@reboot (sleep 30s;cd /home/checklistmgr/;/usr/local/bin/docker-compose up -d)&***

Il est aussi possible de mettre le site en indisponibilité en exécutant la commande suivante : ***/home/checklistmgr/scripts/indispo.sh***

La remise en disponibilité se fait par la commande suivante : ***/home/checklistmgr/scripts/dispo.sh***

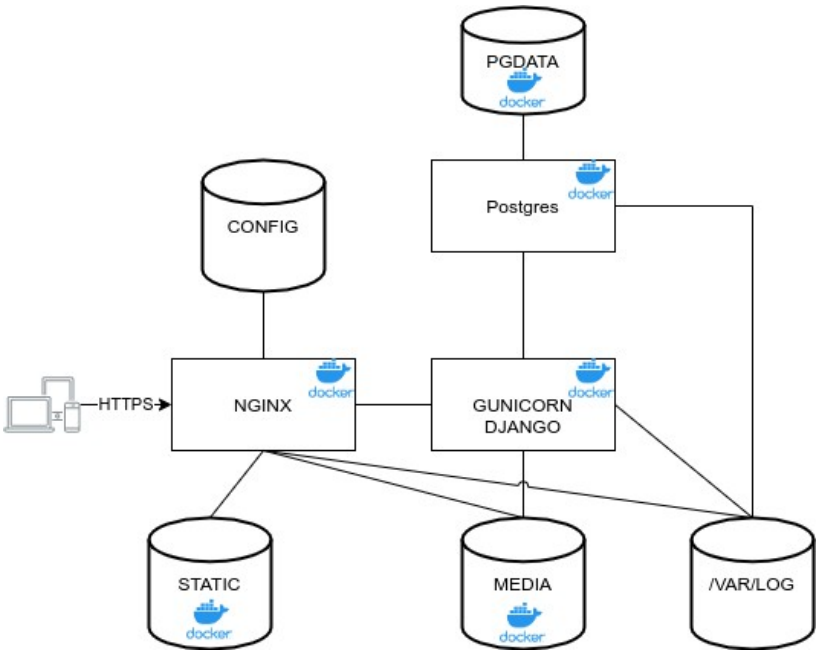
Les 2 scripts se trouvent en annexe

6.3 - Fichiers logs

Les fichiers logs se trouvent dans le répertoire : /var/log/ du serveur (docker pour postgres, django pour django/gunicorn et nginx pour nginx).

7 - Annexes

7.1 - Schéma architecture globale



7.2 - Fichier www.conf → Configuration NGINX

```
server{
 listen 80;
 server_name www.jm-hayons74.fr;
 return 301  https://www.jm-hayons74.fr;
}

server {
    listen 443 ssl;
    server_name www.jm-hayons74.fr;

    ssl_certificate     /etc/nginx/conf.d/certificate.crt;
    ssl_certificate_key     /etc/nginx/conf.d/private.key;

    rewrite "/static/\d+/(.*)" /static/$1 last;

    location /static/ {alias /home/checklistmgr/staticfiles/;}
    location /media/ {alias /home/checklistmgr/media/;}

    location / {
            sendfile off;
            proxy_cache_bypass $http_secret_header;
            add_header X-Cache-Status $upstream_cache_status;

            proxy_headers_hash_max_size 512;
            proxy_headers_hash_bucket_size 128;



        proxy_set_header Host $http_host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_redirect off;
        proxy_pass http://www.jm-hayons74.fr;
    }

        add_header Last-Modified $date_gmt;
        add_header Cache-Control 'no-store, no-cache, must-revalidate, proxy-revalidate, max-age=0';
        if_modified_since off;
        expires off;
        etag off;

            error_page 404 /error;
            access_log /var/log/nginx/checklistmgr-access.log;
            error_log /var/log/nginx/checklistmgr-error.log;
}
```

7.3 - Procédure monitoring.

```
import json
import os
import psutil
import requests

ports = {4321: 'ssh',
        5432: 'postgres',
        80: 'http',
        443: 'https',
        }

def go():
    connections = psutil.net_connections('tcp')
    tab_ports = set(sc.laddr.port for sc in connections if sc.laddr.port in list(ports.keys()))
    missed_port = {key: value for (key, value) in ports.items() if key not in tab_ports}
```

```
    if missed_port:     # dict is not empty !
        api_key = os.getenv("MAILGUN_APIKEY")
        text = 'Attention des ports semblent ne plus répondre sur le serveur www.jm-hayons74.fr!' \
               ' ' \

        text += json.dumps(missed_port)

        rc = requests.post(
            "https://api.mailgun.net/v3/sandbox1f42285ff9e446fa9e90d34287cd8fee.mailgun.org/messages",
            auth=("api", api_key),
            data={"from": "Serveur www.jm-hayons74.fr <root@jm-hayons74.fr>",
                  "to": ["jmlm74@gmail.com"],
                  "subject": "Ports serveur www.",
                  "text": text})
        return
    else:
        return

if __name__ == '__main__':
    go()
```

7.4 - Script de mise à jour le l'application

```
#! /bin/bash
echo "docker-compose stop"
docker-compose stop
docker-compose down
echo "lancement site d attente"
docker run --name nginx2 -p 80:80 -v /home/jmlm/sitedattente/:/usr/share/nginx/html:ro -d nginx
echo "docker-compose rm -f"
docker-compose rm -f
echo "docker system prune -a -f"
docker system prune -a -f
echo "docker-compose pull"
docker-compose pull
echo "arret site d attente"
docker stop nginx2
echo "docker-compose up -d"
docker-compose up -d
echo "arret site d attente"
docker stop ocp
echo "docker-compose up -d"
docker-compose up -d
```

7.5 - Scripts mise à dispo-indispo du site

```
#! /bin/bash
echo "docker-compose stop"
docker-compose stop
docker-compose down
echo "lancement site d attente"
docker run --name nginx2 -p 80:80 -v /home/jmlm/sitedattente/:/usr/share/nginx/html:ro -d nginx
```

```
#! /bin/bash
echo "arret site d attente"
docker stop nginx2
echo "docker-compose up -d"
docker-compose up -d
```

7.6 - Configurations Docker

Dockerfile → Création du conteneur

```
# pull official base image
FROM python:3.8.6

# set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

#ENV DIR1=P13-CheckListMgr
ENV DIR1=./
ENV DIR2=checklistmgr

# install dependencies
RUN apt-get update
RUN apt-get install -y apt-utils
RUN apt-get install -y build-essential python3-dev python3-pip python3-setuptools python3-wheel python3-cffi libcairo2
libpango-1.0-0 libpangocairo-1.0-0 libgdk-pixbuf2.0-0 libffi-dev shared-mime-info

# set work directory
WORKDIR /home/jmlm

# create and activate virtual environment
RUN python3 -m venv env

# copy and install pip requirements
RUN ./env/bin/pip3 install --upgrade pip setuptools
COPY ./${DIR1}/requirements.txt /home/jmlm/requirements/requirements.txt
RUN ./env/bin/python3 -m pip install --upgrade pip
RUN ./env/bin/pip3 install wheel
RUN ./env/bin/pip3 install -r /home/jmlm/requirements/requirements.txt

# copy Django project files
COPY ./${DIR1}/${DIR2}/ /home/jmlm/${DIR2}
WORKDIR /home/jmlm/${DIR2}

# create django-log dir
RUN mkdir -m 766 /var/log/django

EXPOSE 8000

# Command
# CMD /home/jmlm/env/bin/gunicorn --workers 3 --access-logfile - --bind 0.0.0.0:8000 checklistmgr.wsgi:application
```

Docker-compose.yml → Execution du conteneur

```
version: '3.7'
services:
  db:
    image: postgres:12
    volumes:
      - db_data:/var/lib/postgresql/data/pgdata
      - /var/run/postgresql:/var/run/postgresql
      - /var/log/docker:/var/log/postgresql/
      - /home/checklistmgr/backup:/backup
    environment:
      - POSTGRES_PASSWORD=${PG_PSW}
      - POSTGRES_HOST_AUTH_METHOD=md5
      - PGDATA=/var/lib/postgresql/data/pgdata
    ports:
      - 5432:5432
```

```
    container_name: "db"
    hostname: "db"
    domainname: "jm-hayons74.fr"

  gunicorn:
    image: jmlm74/checklistmgr:latest
    command: bash -c "/home/jmlm/env/bin/python manage.py makemigrations &&
                      /home/jmlm/env/bin/python manage.py migrate &&
                      /home/jmlm/env/bin/python manage.py collectstatic --no-input &&
                      /home/jmlm/env/bin/gunicorn --workers 3 --access-logfile /var/log/django/gunicorn.log --bind 0.0.0.0:8000
checklistmgr.wsgi:application"
    depends_on:
      - db
    volumes:
      - static_data:/home/jmlm/checklistmgr/staticfiles/
      - media_data:/home/jmlm/checklistmgr/media/
      - /var/log/django:/var/log/django/
    ports:
      - 8000:8000
    environment:
      DJANGO_SETTINGS_MODULE: ${DJANGO_SETTINGS_MODULE}
      SECRET_KEY: ${SECRET_KEY}
      MAILGUN_APU-KEY: ${MAILGUN_APIKEY}
      DATABASE_USER: ${DATABASE_USER}
      DATABASE_PSW: ${DATABASE_PSW}
      DATABASE_NAME: ${DATABASE_NAME}
      DATABASE_HOST: ${DATABASE_HOST}
    hostname: "gunicorn"
    container_name: "gunicorn"
    domainname: "jm-hayons74.fr"

  nginx:
    image: nginx:latest
    ports:
      - 80:80
      - 443:443
    volumes:
      - ${NGINX_CONFDIR}:/etc/nginx/conf.d
      - static_data:/home/checklistmgr/staticfiles/
      - media_data:/home/checklistmgr/media/
      - /var/log/nginx:/var/log/nginx/
    depends_on:
      - gunicorn
    hostname: "nginx"
    container_name: "nginx"
    domainname: "jm-hayons74.fr"

volumes:
  static_data:
    external:
      name: static_data
  media_data:
    external:
      name: media_data
  db_data:
    external:
      name: db_data
```

7.7 - Crontab

La crontab est celle de l'utilisateur root :

```
MAILTO=jmlm74@gmail.com

@reboot (sleep 30s;cd /home/checklistmgr/;/usr/local/bin/docker-compose up -d)&
*/20 * * * * cd /home/checklistmgr/scripts/; python3 -m monitor
@reboot (sleep 40s;docker run --name do-agent -v /proc:/host/proc:ro  -v /sys:/host/sys:ro -d digitalocean/do-agent:stable)&
# backup
00 05 * * * find /home/checklistmgr/backup/ -name "*.dmp" -type f -mtime +30 -exec rm -f {} \;
30 04 * * * /home/checklistmgr/scripts/backup_db.sh
# maj
00 03 * * 1 /home/checklistmgr/scripts/maj.sh
```

7.8 - Ficher */etc/*environement

MAILGUN_APIKEY=*la-clé-mailgun*
SECRET_KEY=*la-clé-secrete : Attention ne pas mettre de $*
DJANGO_SETTINGS_MODULE=checklistmgr.prod_settings
PG_PSW=*mot-de-passe-utilisateur postgres*
*D*ATABASE_NAME=checklistmgr
DATABASE_USER=*utilisateur BdD*
DATABASE_PSW=*mot-de-passe*
DATABASE_HOST=db Fichier pg_env
POSTGRES_PASSWORD=*mot_de_passe utilisateur postgres (cf /etc/environment)*
POSTGRES_HOST_AUTH_METHOD=md5
PGDATA=/var/lib/postgresql/data/pgdata
NGINX_CONFDIR=/home/checklistmgr/config/nginx/conf.d

8 - Glossaire

WSGI	La Web Server Gateway Interface (WSGI) est une spécification qui définit une interface entre des serveurs et des applications web pour le langage python.
Virtualhost	Dispositif permettant à un serveur web d'héberger plusieurs sites web
SSL/TLS	Protocoles de sécurisation des échanges en réseau et notamment par le réseau Internet. Permet de crypter les échanges entre 2 machines.
Crontab	Outil Unix permettant de lancer des taches à horaire fixe
Docker	Logiciel permettant de lancer des applications dans des conteneurs logiciels.