|  |
| --- |
| **OC PIZZA**  **ocpizza.com**  Dossier d'exploitation  Version 1.0 |
| **Auteur**  Thibaut Vuillaume  *Développeur junior* |

Table des matières

1 - Versions 4

2 - Introduction 5

2.1 - Objet du document 5

2.2 - Références 5

3 - Pré-requis 6

3.1 - Système 6

3.1.1 - Fournisseur de service 6

3.1.2 - Administration du serveur hôte 7

3.1.2.1 - Installation de MobaXterm 7

3.1.2.2 - Paramétrage de MobaXterm 8

3.1.3 - Installation de Docker et Docker Compose sur le serveur hôte 10

3.1.3.1 - Docker 10

3.1.3.2 - Docker Compose 12

3.1.4 - Serveur de base de données 12

3.1.4.1 - Caractéristiques techniques 13

3.1.4.2 - Configuration 13

3.1.5 - Serveur Backend 13

3.1.5.1 - Caractéristiques techniques 14

3.1.6 - Serveur Frontend 14

3.1.6.1 - Caractéristiques techniques 14

3.2 - Service de paiement externe : Stripe 14

4 - Procédure de déploiement 16

4.1 - Import des différents composants sur le serveur hôte 16

4.1.1 - Artefacts 16

4.1.2 - Dockerfiles 16

4.1.3 - Docker Compose 16

4.1.4 - Scripts 18

4.1.5 - Variables d’environnement 18

4.2 - Déploiement du Backend de l’application (service) 19

4.2.1 - Artefacts 19

4.2.2 - Dockerfile 19

4.2.3 - Variables d’environnement 19

4.3 - Déploiement du Frontend de l’application (client) 20

4.3.1 - Artefacts 20

4.3.2 - Dockerfile 20

4.3.3 - Variables d’environnement 21

4.4 - Déploiement de la base de données 22

4.4.1 - Dockerfile 22

4.4.2 - Création de la base de données et des schémas 22

4.4.3 - Variables d’environnement 22

4.5 - Vérification 23

5 - Procédure de démarrage / arrêt 24

5.1 - Démarrage des composants 24

5.2 - Arrêt des composants 24

5.3 - Arrêt d’un conteneur 24

5.4 - Redémarrage d’un conteneur 24

6 - Procédure de mise à jour 26

6.1 - Mise à jour d’un composant 26

6.2 - Mise à jour de la config Docker (Dockerfile / docker-compose.yml) 26

7 - Supervision/Monitoring 27

7.1 - Monitoring du serveur hôte 27

7.2 - Supervision des conteneurs Docker 28

7.3 - Supervision du Backend 29

8 - Procédure de sauvegarde et restauration 30

8.1 - Mise en place du backup automatique périodique 30

9 - Glossaire 32

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Thibaut Vuillaume | 09/10/2019 | Finalisation du document | 1.0 |
| Thibaut Vuillaume | 02/10/2019 | Rédaction du document | 05 |
| Thibaut Vuillaume | 25/09/2019 | Création du document | 0.1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Objet du document

Le présent document constitue le dossier d’exploitation de l'application ocpizza.com

Objectif du document : préciser la démarche de déploiement de l’application ocpizza.com.

## Références

Pour de plus amples informations concernant l’application, se référer à :

1. **ocpizza.com - dossier de conception fonctionnelle.docx**
2. **ocpizza.com - dossier de spécifications techniques.pdf**

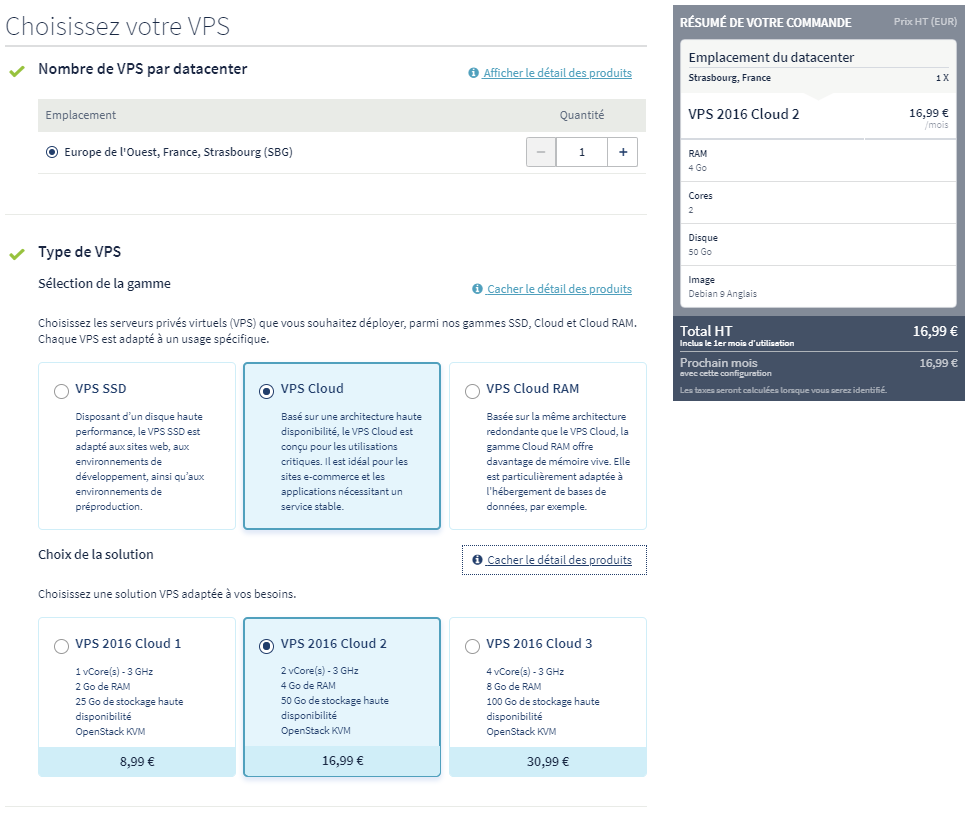
# Pré-requis

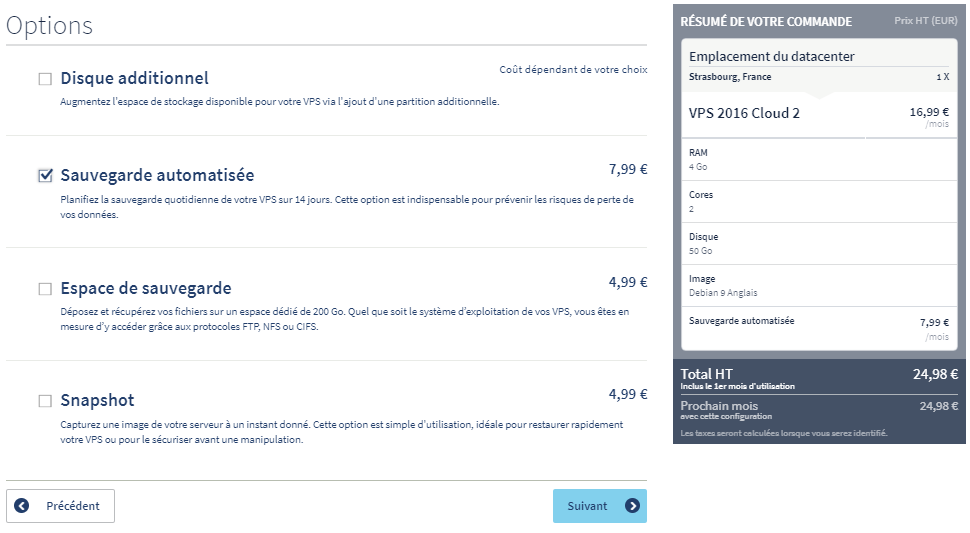
## Système

### Fournisseur de service

Le serveur qui hébergera l’application ocpizza.com et ses composants [[1]](#endnote-1)sera fourni par ovh.com.

L’offre retenue sera [VPS Cloud 2](https://www.ovh.com/fr/vps/vps-cloud.xml), proposée à [16.99€ HT/mois](https://www.ovh.com/fr/order/vps/#/vps/build?selection=~(build~(family~'ceph-nvme~quantity~1~offer~'vps_ceph-nvme_model2_2018v1~osType~'linux~datacenter~'_default))&npm=1).





Une telle infrastructure sera suffisante pour garantir un bon niveau de performance de l’application, avec une disponibilité garantie à 99.99%. La sauvegarde automatisée préviendra tout risque de perte de données.

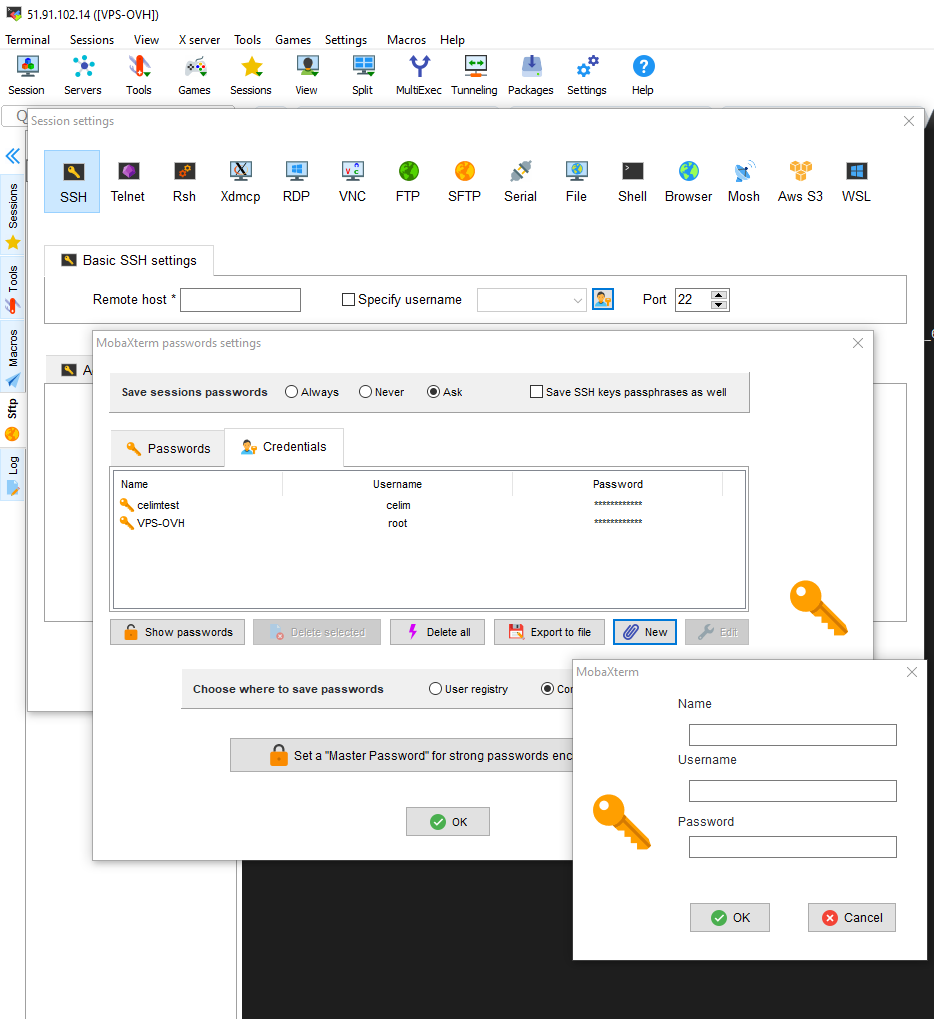
### Administration du serveur hôte

#### Installation de MobaXterm

Télécharger puis installer MobaXterm : <https://mobaxterm.mobatek.net/download.html>

#### Paramétrage de MobaXterm

1



Suivre la séquence ci-dessus et entrer le Username et le Password fourni par le fournisseur de service.

Suivre la séquence ci-dessus : saisir l’adresse IP fourni par le fournisseur de service, sélectionner le Username précédemment enregistré puis valider.

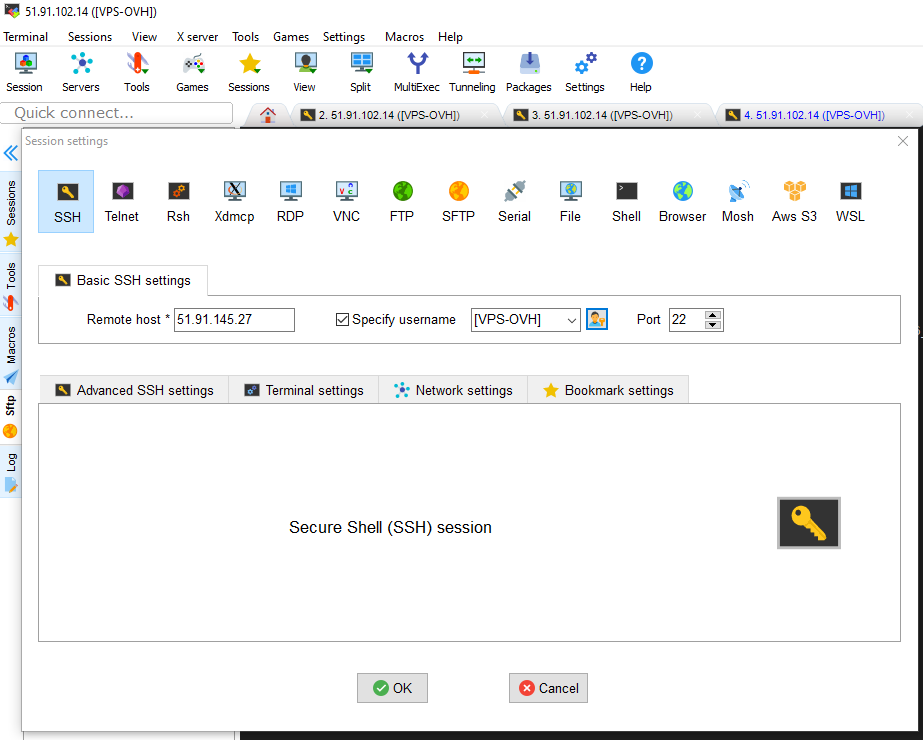
1

1

1

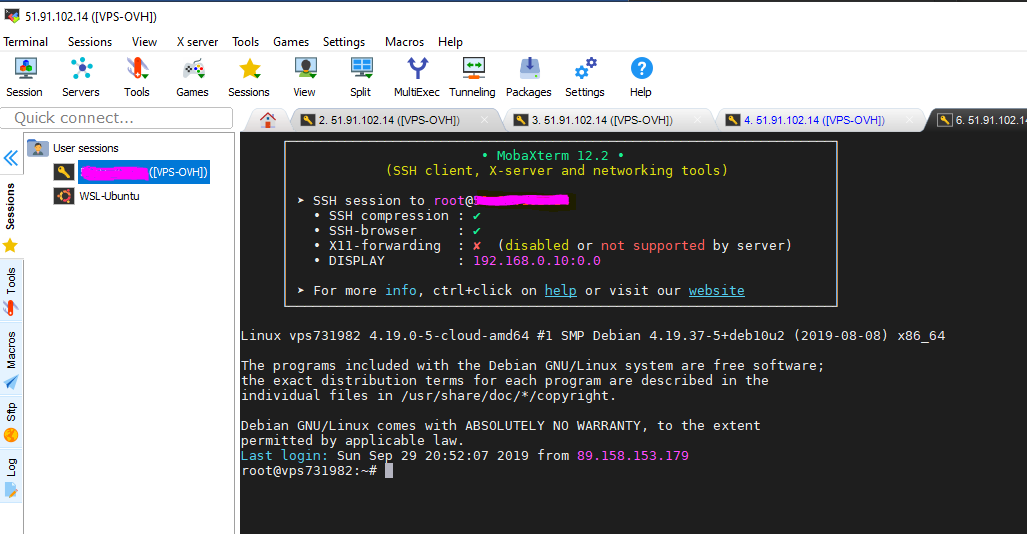
1

1



Accéder ensuite au terminal en double cliquant sur la session qui vient d’être créée.

1



### Installation de Docker et Docker Compose sur le serveur hôte

#### Docker[[2]](#endnote-2)

Accéder au terminal du serveur hôte[[3]](#endnote-3), et exécuter les commandes suivantes :

*# mise à jour de la liste des packages*

$ sudo apt update

*# installation des packages prérequis qui autorisent apt à utiliser des packages via HTTPS*

$ sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common

*# ajout de la clé GPG pour le dépôt Docker officiel sur le serveur hôte*

$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add –

*# ajout du dépôt Docker aux sources APT*

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb\_release -cs) stable"

*# mise à jour de la base de données des packages avec les packages Docker depuis le dépôt nouvellement ajouté*

$ sudo apt update

*# garantie une installation de Docker depuis le dépôt Docker et non depuis le dépôt Debian par défaut*

$ apt-cache policy docker-ce

*# output attendu*

> docker-ce:

Installed: (none)

Candidate: 18.06.1~ce~3-0~debian

Version table:

18.06.1~ce~3-0~debian 500

500 https://download.docker.com/linux/debian stretch/stable amd64 Packages

*# installation de Docker*

$ sudo apt install docker-ce

*# verification de l’installation*

$ sudo systemctl status docker

*# output attendu*

> docker.service - Docker Application Container Engine

Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)

Active: active (running) since Thu 2018-07-05 15:08:39 UTC; 2min 55s ago

Docs: https://docs.docker.com

Main PID: 21319 (dockerd)

CGroup: /system.slice/docker.service

├─21319 /usr/bin/dockerd -H fd://

└─21326 docker-containerd --config /var/run/docker/containerd/containerd.toml

*# ajout du username au groupe docker afin de ne plus avoir besoin de la commande sudo*

$ sudo usermod -aG docker ${USER}

*# application des changements*

$ su - ${USER}

*# verification*

$ id -nG

*# output attendu*

> root docker

#### Docker Compose

Accéder au terminal du serveur hôte, et exécuter les commandes suivantes :

*# téléchargement de la release actuelle stable de Docker Compose*

$ sudo curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

*# application des permissions executables*

$ sudo chmod +x /usr/local/bin/docker-compose

### Serveur de base de données

La base de données sera sous PostgreSQL 11.

En vue des 3 environnements Dev, Test et Prod, 3 schémas seront créés :

* [ocpizza\_dev](https://github.com/thibaut54/ocpizza.com/blob/master/create_db_ocpizza_dev.sql)
* [ocpizza\_test](https://github.com/thibaut54/ocpizza.com/blob/master/create_db_ocpizza_test.sql)
* [ocpizza](https://github.com/thibaut54/ocpizza.com/blob/master/create_db_ocpizza.sql)

L’insertion des données sera faite par le client dans l’interface prévue à cet effet. Cette interface ne sera accessible que par les collaborateurs dont le compte dispose du rôle « admin ».

#### Caractéristiques techniques

La base de données sera déployée dans un conteneur Docker[[4]](#endnote-4).

L’image Docker [[5]](#endnote-5)qui sera construite est décrite dans un dockerfile : [Dockerfile-db](https://github.com/thibaut54/ocpizza.com/blob/master/Dockerfile-db).

L’image utilise une distribution Linux : Debian GNU/Linux 9.11 (stretch).

L’allocation des ressources du serveur hôte au conteneur est gérée automatiquement par Docker. On peut toutefois modifier les paramètres par défauts, en se référant à cette documentation : <https://docs.docker.com/config/containers/resource_constraints/>.

#### Configuration

Nom de la base de données : db\_ocpizza.

Host : xxx-xxx-xxx-xxx.

Port : 5432.

URL : jdbc:postgresql:// xxx-xxx-xxx-xxx:5432/db\_ocpizza

### Serveur Backend

Le Backend[[6]](#endnote-6) de l’application est développé en Java 8. Le framework Springboot 2 est utilisé, avec une architecture REST.

Le backend de l’application sera packagé dans un fichier .war. Ce dernier sera à importer sur le serveur hôte, à l’emplacement : /opt/ocpizza/docker/build/back. Il sera déployé dans un conteneur Docker.

#### Caractéristiques techniques

L’image Docker qui sera construite est décrite dans un dockerfile : [Dockerfile-back](https://github.com/thibaut54/ocpizza.com/blob/master/build/back/Dockerfile-back).

L’image utilise une distribution Linux : Debian GNU/Linux 9 (stretch).

L’allocation des ressources du serveur hôte au conteneur est gérée automatiquement par Docker.

### Serveur Frontend

Le Frontend [[7]](#endnote-7)de l’application est développé en Angular 7. Le contenu de l’archive « ocpizza-front.zip » sera à importer sur le server host, à l’emplacement : /opt/ocpizza/front/. Il sera déployé dans un conteneur Docker.

#### Caractéristiques techniques

L’image Docker qui sera construite est décrite dans un dockerfile : [Dockerfile-front](https://github.com/thibaut54/ocpizza.com/blob/master/build/front/Dockerfile-front).

L’image utilise une distribution de Linux : Debian GNU/Linux 9.9 (stretch)

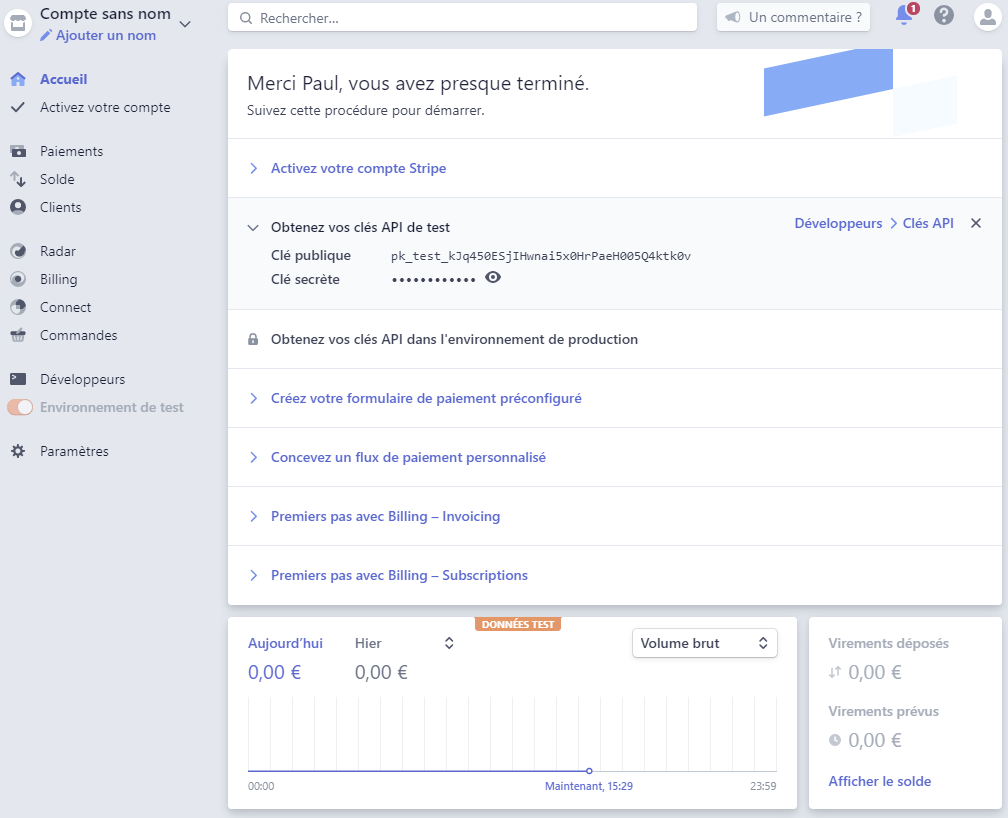
L’allocation des ressources du serveur hôte au conteneur est gérée automatiquement par Docker.

## Service de paiement externe : Stripe

Un service de paiement externe est utilisé. Il s’agit de [Stripe](https://stripe.com/fr).

Un compte test a été créé. Il est accessible à cette adresse : <https://dashboard.stripe.com/test/dashboard>.

Identifiant : tvuillaume10@gmail.com  
Mot de passe : 71eult1IZKuc



# Procédure de déploiement

## Import des différents composants sur le serveur hôte

### Artefacts

* **ocpizza-front.zip** : fichier obtenu en archivant le Frontend de l’application avec Winzip. Extraire et importer à l’emplacement < /opt/ocpizza/front >.
* **ocpizza-back.war** : fichier obtenu en packageant le Backend de l’application avec Maven. Importer à l’emplacement < /opt/ocpizza/front/back >.

### Dockerfiles

* [Dockerfile-back](https://github.com/thibaut54/ocpizza.com/blob/master/build/back/Dockerfile-back) : fichier qui décrit l’image Docker utilisée par le conteneur du Backend. Importer à l’emplacement < /opt/ocpizza/front >.
* [Dockerfile-front](https://github.com/thibaut54/ocpizza.com/blob/master/build/front/Dockerfile-front) : fichier qui décrit l’image Docker utilisée par le conteneur du Frontend. Importer à l’emplacement < /opt/ocpizza/front >.
* [Dockerfile-db](https://github.com/thibaut54/ocpizza.com/blob/master/db/Dockerfile-db) : fichier qui décrit l’image Docker utilisée par le conteneur de la base de données. Importer à l’emplacement < /opt/ocpizza/front >.

### Docker Compose

* [docker-compose.yml](https://github.com/thibaut54/ocpizza.com/blob/master/docker-compose.yml): ce fichier définit les services et le réseau qui seront créés par Docker. Importer à l’emplacement < /opt/ocpizza/front >

version: '3.7'  
  
services:  
 db-ocpizza:  
 build:  
 context: .  
 dockerfile: db/Dockerfile-db  
 container\_name: cont-ocpizza-db  
 restart: unless-stopped  
 ports:  
 - 5432:5432  
 expose:  
 - 5432  
 environment:  
 POSTGRES\_USER: admin\_ocp  
 POSTGRES\_PASSWORD: 123  
 POSTGRES\_DB: db\_ocpizza  
 volumes:  
 - db\_data:/var/lib/postgres/data  
 networks:  
 - ocp-network  
  
 back-ocpizza:  
 depends\_on:  
 - db-ocpizza  
 build:  
 context: .  
 dockerfile: build/back/Dockerfile-back  
 container\_name: cont-ocpizza-back  
 ports:  
 - 8080:8080  
 expose:  
 - 8080  
 environment:  
 SPRING\_DATASOURCE\_USERNAME: admin\_ocp  
 SPRING\_DATASOURCE\_PASSWORD: 123  
 SPRING\_DATASOURCE\_URL: jdbc:postgresql://db-ocpizza:5432/db\_ocpizza  
 networks:  
 - ocp-network  
  
 front-ocpizza:  
 build:  
 context: .  
 dockerfile: build/front/Dockerfile-front  
 container\_name: cont-ocpizza-front  
 volumes:  
 - 'front:/app'  
 - '/app/node\_modules'  
 ports:  
 - 4200:4200  
 networks:  
 - ocp-network  
  
networks:  
 ocp-network:  
  
volumes:  
 db\_data:  
 driver: local  
 driver\_opts:  
 type: 'none'  
 o: 'bind'  
 device: '/opt/ocpizza/docker/volumes/postgres'  
 front:

### Scripts

Scripts SQL :

* [create\_db\_ocpizza](https://github.com/thibaut54/ocpizza.com/blob/master/db/create_db_ocpizza.sql) : ce fichier contient le script SQL de création du schéma de destiné à l’environnement de production.
* [create\_db\_ocpizza\_test](https://github.com/thibaut54/ocpizza.com/blob/master/db/create_db_ocpizza_test.sql) : ce fichier contient le script SQL de création du schéma de destiné à l’environnement de test.
* [create\_db\_ocpizza\_dev](https://github.com/thibaut54/ocpizza.com/blob/master/db/create_db_ocpizza_dev.sql) : ce fichier contient le script SQL de création du schéma de destiné à l’environnement de développement.

Ces fichiers sont à importer à l’emplacement < /opt/ocpizza/docker/db >.

Scripts Shell :

* [backup-db.sh](https://github.com/thibaut54/ocpizza.com/blob/master/sh/backup-db.sh) : ce fichier contient le script Shell qui lancera la commande de backup de la base de données PostgreSQL.
* [extract-backup.sh](https://github.com/thibaut54/ocpizza.com/blob/master/sh/extract-backup.sh) : ce fichier contient le script Shell qui lancera la commande de récupération du backup de la base de données depuis son conteneur.

Ces fichiers sont à importer à l’emplacement < /opt/ocpizza/docker/sh >.

### Variables d’environnement

Le serveur hôte dispose des variables d’environnement suivantes :

|  |  |
| --- | --- |
| **NOM DE VARIABLE** | **CONTENU** |
| **SHELL** | **/bin/bash** |
| **PWD** | **/opt/ocpizza** |
| **LOGNAME** | **root** |
| **XDG\_SESSION\_TYPE** | **tty** |
| **HOME** | **/root** |
| **LANG** | **en\_US.UTF-8** |
| **SSH\_CONNECTION** | **xxx.xxx.xxx.xxx 63434 xxx.xxx.xxx.xxx** |
| **XDG\_SESSION\_CLASS** | **user** |
| **TERM** | **xterm** |
| **USER** | **root** |
| **SHLVL** | **1** |
| **XDG\_SESSION\_ID** | **6921** |
| **XDG\_RUNTIME\_DIR** | **/run/user/0** |
| **SSH\_CLIENT** | **xxx.xxx.xxx.xxx 63434 22** |
| **PATH** | **/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin** |
| **MAIL** | **/var/mail/root** |
| **SSH\_TTY** | **/dev/pts/4** |
| **\_** | **/usr/bin/printenv** |
| **OLDPWD** | **/root** |

## Déploiement du Backend de l’application (service)

### Artefacts[[8]](#endnote-8)

Le Backend de l’application est packagée dans un fichier .war : ocpizza-back.war.

### Dockerfile

FROM openjdk:8  
ADD build/back/ocpizza-back.war ocpizza-back.war  
ENTRYPOINT ["java", "-jar", "ocpizza-back.war"]

### Variables d’environnement

Le conteneur Docker contiendra les variables d’environnements suivantes :

|  |  |
| --- | --- |
| **NOM DE VARIABLE** | **CONTENU** |
| **LANG** | **C.UTF-8** |
| **HOSTNAME** | **a53f91a1da6b** |
| **SPRING\_DATASOURCE\_URL** | **jdbc:postgresql://db-ct:5432/db\_contacts** |
| **JAVA\_HOME** | **/usr/local/openjdk-8** |
| **JAVA\_VERSION** | **8u222** |
| **PWD** | **/** |
| **HOME** | **/root** |
| **SPRING\_DATASOURCE\_PASSWORD** | **123** |
| **TERM** | **xterm** |
| **JAVA\_BASE\_URL** | **https://github.com/AdoptOpenJDK/openjdk8-upstream-binaries/releases/download/jdk8u222-b10/OpenJDK8U-jdk\_** |
| **SHLVL** | **1** |
| **JAVA\_URL\_VERSION** | **8u222b10** |
| **SPRING\_DATASOURCE\_USERNAME** | **admin\_ct** |
| **PATH** | **/usr/local/openjdk-8/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin** |
| **\_** | **/usr/bin/printenv** |

## Déploiement du Frontend de l’application (client)

### Artefacts

Le Frontend de l’application est packagée dans l’archive « ocpizza-front.zip ». Son contenu doit être uploadé sur le serveur hôte à l’emplacement </opt/ocpizza/front>

### Dockerfile

# image de base  
FROM node:latest

# installe chrome pour les tests protractor  
RUN wget -q -O - https://dl-ssl.google.com/linux/linux\_signing\_key.pub | apt-key add -  
RUN sh -c 'echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main" >> /etc/apt/sources.list.d/google.list'  
RUN apt-get update && apt-get install -yq google-chrome-stable  
  
# définit le repertoire de travail  
WORKDIR /app  
  
# ajoute `/app/node\_modules/.bin` au $PATH  
ENV PATH /app/node\_modules/.bin:$PATH  
  
# installe et met en cache les dépendances de l’application  
COPY build/front/package.json /app/package.json  
RUN npm install  
RUN npm install -g @angular/cli@7.3.9  
  
# copie app  
COPY build/front/. /app  
  
# démarre l’application  
CMD ng serve --host 0.0.0.0

### Variables d’environnement

Le conteneur Docker contiendra les variables d’environnements suivantes :

|  |  |
| --- | --- |
| **NOM DE VARIABLE** | **CONTENU** |
| **YARN\_VERSION** | **1.15.2** |
| **HOSTNAME** | **a057d3e67fd0** |
| **PWD** | **/app** |
| **HOME** | **/root** |
| **NODE\_VERSION** | **12.2.0** |
| **TERM** | **xterm** |
| **SHLVL** | **1** |
| **PATH** | **/app/node\_modules/.bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin** |
| **\_** | **/usr/bin/printenv** |

## Déploiement de la base de données

### Dockerfile

FROM postgres:latest  
COPY db/create\_db\_ocpizza.sql /docker-entrypoint-initdb.d/init.sql  
COPY db/insert\_data\_ocpizza.sql /opt/insert\_data\_ocpizza.sql  
COPY sh/backup-db.sh /opt/backup-db.sh  
RUN apt-get update \  
&& apt-get install vim -y \  
&& export EDITOR=/usr/bin/vim \  
&& chmod +x /opt/backup-db.sh \  
&& cd /opt \  
&& mkdir db\_backup \

### Création de la base de données et des schémas

La base de données et les schémas sont construits lorsque le conteneur Docker démarre.

Cela est rendu possible par la commande du Dockerfile :

COPY db/create\_db\_ocpizza.sql /docker-entrypoint-initdb.d/init.sql

Cette commande importe le fichier < create\_db\_ocpizza.sql > dans le conteneur. Il sera exécuté à la construction de l’image.

### Variables d’environnement

Le conteneur Docker contiendra les variables d’environnements suivantes :

|  |  |
| --- | --- |
| **NOM DE VARIABLE** | **CONTENU** |
| **LANG** | **en\_US.utf8** |
| **HOSTNAME** | **f9e4161aef0a** |
| **PG\_MAJOR** | **11** |
| **PWD** | **/** |
| **HOME** | **/root** |
| **PG\_VERSION** | **11.5-1.pgdg90+1** |
| **GOSU\_VERSION** | **1.11** |
| **PGDATA** | **/var/lib/postgresql/data** |
| **POSTGRES\_DB** | **db\_contacts** |
| **TERM** | **xterm** |
| **POSTGRES\_PASSWORD** | **123** |
| **POSTGRES\_USER** | **admin\_ct** |
| **SHLVL** | **1** |
| **PATH** | **/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/lib/postgresql/11/bin** |
| **\_** | **/usr/bin/printenv** |

## Vérification

Afin de vérifier le bon déploiement de l’application, accéder au terminal du serveur hôte et exécutez la commande <docker-compose ps>.

Output attendu :

Name Command State Ports

---------------------------------------------------------------------------------

cont-ocpizza-back java -jar ocpizza-back.jar Up 0.0.0.0:8080->8080/tcp

cont-ocpizza-front /bin/sh -c ng serve --host ... Up 0.0.0.0:4200->4200/tcp

cont-ocpizza-db docker-entrypoint.sh postgres Up 0.0.0.0:5432->5432/tcp

# Procédure de démarrage / arrêt

## Démarrage des composants

Accéder au terminal du serveur hôte, et exécuter la commande suivante :

$ docker-compose up

Cela construira toutes les images en suivant le fichier de configuration   
< docker-compose.yml > puis démarrera tous les conteneurs et services décrits dans ce dernier.

## Arrêt des composants

Accéder au terminal du serveur hôte, et exécuter la commande suivante :

$ docker-compose down

Cela arrêtera tous les services du < docker-compose.yml > et donc tous les composants.

## Arrêt d’un conteneur

Accéder au terminal du serveur hôte, et exécuter les commandes suivantes :

$ docker-compose stop <nom-du-service>

Cela arrêtera le conteneur Docker désigné par < nom-du- service >.

## Redémarrage d’un conteneur

Accéder au terminal du serveur hôte, et exécuter les commandes suivantes :

$ docker-compose restart <nom-du-service>

Cela redémarrera le conteneur Docker désigné par < nom-du- service >.

# Procédure de mise à jour

Pendant une procédure de mise-à-jour, un service Docker doit être stoppé pour être redémarré. Une interruption de service est donc ici nécessaire.

Il est indispensable de n’exécuter ces procédures que lors des périodes les plus creuses en sollicitation de ocpizza.com.

## Mise à jour d’un composant

Quel que soit le composant concerné, Docker facilite grandement les choses.

Il suffit de :

* Importer le composant mis-à-jour sur le serveur hôte (le .war pour le Backend, le contenu du projet Angular pour le Frontend, le script SQL pour la base de données). Si besoin, modifier le Dockerfile concerné.
* Arrêter le conteneur concerné (cf. partie 5.2).
* Reconstruire l’image du conteneur concerné :

$ docker-compose build <nom-du-service>

* Redémarrer le conteneur concerné (cf. partie 5.3).
* (mise à dispo des artefact sur git)

## Mise à jour de la config Docker (Dockerfile / docker-compose.yml)

Modifier le ou les fichiers concernés.

Saisir la commande suivante dans le terminal du serveur hôte :

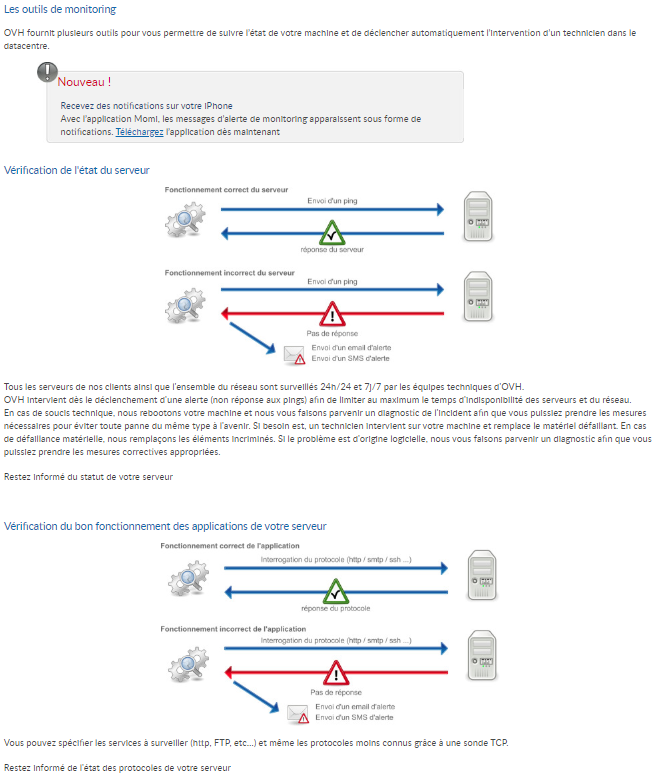
$ docker-compose up --force-recreate --no-deps --build <nom-du-service>

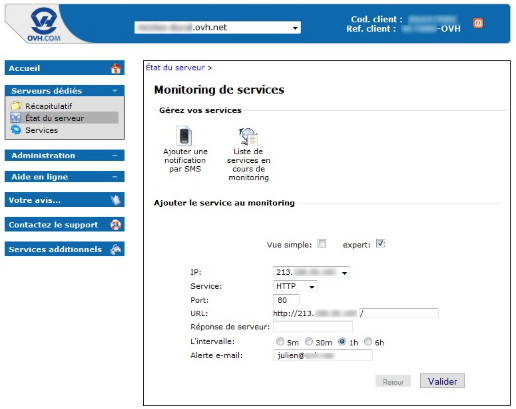
Cela reconstruira l’image du service et relancera son conteneur.

# Supervision/Monitoring

## Monitoring du serveur hôte

OVH fourni un service de monitoring, au niveau du serveur et des applications, intégrant une service d’alerte email et sms en cas de défaillance.





## Supervision des conteneurs Docker

Accéder au terminal du serveur hôte, et exécuter les commandes suivantes :

$ docker stats <OPTIONS> <CONTAINER…>

Cela affichera différents indicateurs relatifs aux conteneurs démarrés sur le serveur :

* ID du conteneur
* Nom du conteneur
* % d’utilisation du CPU
* Utilisation de RAM - Total disponible - % d’utilisation
* NET – I/O
* BLOCK – I/O
* PIDS

## Supervision du Backend

Springboot Actuator est implémenté. Il fournira les informations ci-dessous :



Pour plus d’information, une documentation détaillée est disponible : <https://docs.spring.io/spring-boot/docs/2.1.8.RELEASE/actuator-api/html/>.

# Procédure de sauvegarde et restauration

## Mise en place du backup automatique périodique

Accéder au terminal du serveur hôte, et exécuter les commandes suivantes :

$ docker exec -it cont-ocpizza-db /bin/bash

Cela donne accès au terminal du conteneur Docker « cont-ocpizza-db ».

Exécuter la commande :

$ crontab -e

Le fichier de CRON de l’OS est ouvert dans VIM.

Entrer en mode édition avec la touche <i>.

Ajouter la chaine de caractères suivante à la suite des commentaires :

0 23 \* \* \* /opt/backup-db.sh

Quitter le mode édition avec la touche <Echap> puis quitter en sauvegardant avec la commande < :wq > puis la touche <Entrée>.

Le script shell suivant sera exécuté tous les jours à 23h :

**#!/bin/bash**HOST="localhost"  
DB="db\_ocpizza"  
USER="admin\_ocp"  
PGPASSWORD="admin"  
DATE=*`date* +%Y%m%d*`*FILENAME="/opt/db\_backup/${DB}\_${DATE}.tar"  
#FILENAME="/opt/db\_backup/${DB}\_${DATE}.sql"  
  
PGPASSWORD=${PGPASSWORD} pg\_restore -d ${DB} ${FILENAME} -c -U ${USER}  
#PGPASSWORD=${PGPASSWORD} pg\_dump -Fc -U ${USER} -h ${HOST} ${DB} > ${FILENAME}

Accéder au terminal du serveur hôte, et exécuter les commandes suivantes :

$ apt-get update && apt-get install vim -y && export EDITOR=/usr/bin/vim

$ mkdir /opt/ocpizza && mkdir opt/ocpizza/db\_backup

$ crontab -e

Le fichier de CRON de l’OS est ouvert dans VIM.

Entrer en mode édition avec la touche <i>.

Ajouter la chaine de caractères suivante à la suite des commentaires :

10 23 \* \* \* /opt/extract-backup.sh

Quitter le mode édition avec la touche <Echap> puis quitter en sauvegardant avec la commande < :wq > puis la touche <Entrée>.

Le script shell suivant sera exécuté tous les jours à 23h10 :

**#!/bin/bash**DB="db\_ocpizza"  
DATE=*`date* +%Y%m%d*`*CONT\_PATH="/opt/db\_backup/${DB}\_${DATE}.tar"  
CONT\_NAME="cont-ocpizza-db"  
DESTINATION="/opt/ocpizza/db\_ocpizza/${DB}\_${DATE}.tar"  
# -E UTF-8  
*docker* cp ${CONT\_NAME}:${CONT\_PATH} ${DESTINATION}

A son exécution, le fichier de backup de la base de données est copié depuis le conteneur Docker <cont-ocpizza-db> vers le serveur hôte, à l’emplacement attribué à la variable < DESTINATION >.

## Restauration à partir d’une sauvegarde

Accéder au terminal du serveur hôte, et exécuter les commandes suivantes :

$ docker exec -it cont-ocpizza-db pg\_restore -d db\_contacts /opt/db\_backup/<nom\_du\_fichier.tar> -c -U admin\_ocp

Cela restaurera le contenu de la sauvegarde de l’archive TAR désignée dans la base de données du conteneur cible.

# Glossaire

1. **Composant** : définit un élément constitutif d’une solution applicative. [↑](#endnote-ref-1)
2. **Docker** : plateforme de conteneurisation utilisée pour déployer les composants de l’application sur le serveur hôte. [↑](#endnote-ref-2)
3. **Serveur hôte** : définit l’entité physique qui hébergera les composants de l’application ocpizza. [↑](#endnote-ref-3)
4. **Conteneur docker** : un conteneur enveloppe un composant dans une boîte virtuelle et isolée avec tout ce dont il a besoin pour s’exécuter. [↑](#endnote-ref-4)
5. **Image docker** : définit une collection ordonnée de changements au niveau d’un système de fichier racine et des paramètres correspondant pour une utilisation à l’intérieur d’un conteneur au runtime. [↑](#endnote-ref-5)
6. **Backend :** dans une solution applicative,désigne l’ensemble des composants logiciels responsables, directement ou non, des traitements effectués en arrière-plan par celle-ci. [↑](#endnote-ref-6)
7. **Frontend :** dans une solution applicative, désigne l’ensemble des composants logiciels qui seront reçues par le client (souvent un navigateur internet tel que Chrome ou Firefox). [↑](#endnote-ref-7)
8. **Artefact :** représente une partie physique de l'information, utilisée ou produite par le processus de développement d'un logiciel (ex : [modèle](https://dico.developpez.com/html/2601-Conception-modele.php), [fichier](https://dico.developpez.com/html/2937-Conception-fichier.php), table, etc.). [↑](#endnote-ref-8)