OC Pizza

Développement du système informatique d'un groupe de pizzerias

Dossier d'exploitation

Version 1.0

Auteur
J. Rouzic
Développeur

Table des matières

1 - Versions

Auteur	Date	Description	Version
J. Rouzic	06/05/2019	Création du document	1
J. Rouzic	09/06/2019	Finalisation du document	1
J. Rouzic	14/06/2019	Réprise / complétion du document	1

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier d'exploitation du système informatique de OC pizza. Les applications et outils couvert par cette documentation sont :

- L'application du **Webservice REST.**
- La **Webapp** MVC statique.
- La base de données **postgreSQL**
- Les deux **applications Angular**

2.2 - Références

Pour de plus amples informations, se référer :

1. **Doc 1 - Dossier de conception fonctionnelle.pdf :** Dossier de conception fonctionnelle de l'application
2. **Doc 2 - Dossier de conception technique.pdf** : Dossier de conception technique de l'application
3. **Doc 4 - PV Livraison.pdf**

3 - Pré-requis

3.1 - Système

	Hébergeur	Type	Nom
Base de données	Clever-cloud.com	Automation platform	XS-bigspace
Webservice REST	Clever-cloud.com	Automation platform	Auto-scalable service
Webapp Spring MVC	Clever-cloud.com	Automation platform	Auto-scalable service
Webapp Angular	Firebase.google.com	Backend as a service	Flame

3.1.1 - Serveur de Base de données

3.1.1.1 - Hébergement

L'hébegeur choisit est clever-cloud : **http://clever-cloud.com/**

Serveur de base de données : addon clever-cloud pour postgreSQL

3.1.1.2 - Caractéristiques techniques

Nom de la solution chez clever-cloud : XS Big Space

Type de serveur : serveur dédié

Système de log : oui

Taille maximale de la base de donnée : 15GB

Maximum de connections : 75

Mémoire vive : 1GB

3.1.2 - Serveur d'application pour la Webapp Spring MVC

3.1.2.1 - Hébergement

Hébergé chez Clever-cloud.com.

3.1.2.2 - Caractéristiques techniques

Nom de la solution chez clever-cloud : Auto-scalable service

Cette solution nous laisse définir le nombre d'instances minimum et maximum de l'application. Clever-cloud lance des instances lorsque la charge l'exige et les ferme

quand la charge redevient raisonnable.

Nombre d'instance minimum : 1

Nombre d'instance maximum : 10

Type de serveur : serveur dédié

Système de log : oui

Système de metrics : oui

Type de serveur : S

Nombre de processeurs : 2

Mémoire vive : 2GB

3.1.3 - Serveur d'application pour le Webservice REST

3.1.3.1 - Hébergement

Étant donné que ce Webservice expose un API REST aux autres composants du SI, on privilégira un herbergement chez clever-cloud.com, afin de bénéficier d'une meilleure réponse à la montée en charge.

3.1.3.2 - Caractéristiques techniques

Nom de la solution chez clever-cloud : Auto-scalable service

Cette solution nous laisse définir le nombre d'instances minimum et maximum de l'application. Clever-cloud lance des instances lorsque la charge l'exige et les ferme quand la charge redevient raisonnable.

Nombre d'instance minimum : 2

Nombre d'instance maximum : 10

Type de serveur : serveur dédié

Système de log : oui

Système de metrics : oui

Type de serveur : S

Nombre de processeurs : 2

Mémoire vive : 2GB

3.1.4 - Serveur d'application pour les applications angular.

3.1.4.1 - Hébergement

L'hébergeur retenu est **google firebase**.

3.1.4.2 - Caractéristiques techniques

Nous n'avons pas de données techniques. Google prenant en charge l'aspect technique.

3.2 - Bases de données

Les bases de données et schémas suivants doivent être accessibles et à jour :

- **db-ocpizza :** version 1.0
- **schéma :** public

4 - Procédure de déploiement

4.1 - Déploiement du Webservice REST

4.1.1 - Artefact

ocpizza-webservice-1.0.0-RELEASE.jar

4.1.2 - Répertoire de configuration applicatif

Fichier application.properties

Le fichier application.properties regroupe l'ensemble des propiétés relatives à cette application. Il est interne au fichier jar.

Cependant, si nécessaire, on pourra écraser certaines propriétés en plaçant un fichier application.properties dans le répertoire du jar. Spring Boot analysera les propriétés trouvées au démarrage du jar, et ces dernières auront la priorité sur les propriétés du application.properties interne. Ceci devrait permettre d'adapter, par exemple, l'URI vers la base de données en cas de changements.

4.1.3 - DataSources

La datasource est décrite dans le fichier application.properties

Le fichier de driver **postgresql** est managé par Maven : il télécharge le driver lors d'un **mvn package** et les propriétés décrive le nom correspondant. Le driver se retrouve au sein des dépendances à l'intérieur du fichier jar.

4.1.4 - Ressources

Les ressources sont internes au fichier jar.

4.1.5 - Déployement

Pour déployer sur Clever-cloud :

Placez vous dans le repertoire de base du projet Java/Spring. Créez le repertoire **./clevercloud**

Créez un fichier **jar.json**. Copiez le code suivant :

```
{
  "build": {
    "type": "maven",
    "goal": "package"
  },
  "deploy": {
    "jarName": "target/ocpizza-webservice-1.0.0-RELEASE.jar"
  }
}
```

Commitez, et faites un **git push origin master** afin de mettre à jour votre repository.

Dans la console de clever-cloud choisissez **create an application,** là, sélectionnez le repository github en question, puis sélectionnez JAR. À l'étape suivante vous devez choisir les spécifications du serveur, ces configurations sont données au paragraphe 3.
Clever-cloud automatisera la build et le déploiement de votre JAR.

4.1.6 - Vérifications

Afin de vérifier le bon déploiement de l'application lancez une invite de commande et saisissez la commande suivante :

curl -X GET "http://api.ocpizza.com/testmethod"

Vous avez également la possibilité d'accèder à la vue d'ensemble sur clever-cloud.com et vous rendre sur Activity.

4.2 - Déploiement des applications Angular

La procédure de déploiement est similaire pour les deux applications Angular (cf 4.2.5).

4.2.1 - Artefacts

On retrouve les fichiers suivant dans {project-root}/dist/{project-name} :

index.html

main.{hash}.js

polyfills.{hash}.js

runtime.{hash}.js

style.{hash}.css

le {hash} est généré lors du build de l'application, sa valeur depend du contenu du fichier, ceci permet de différencier différentes versions du fichier.

On retrouve également différents fichiers d'assets.

4.2.2 - Répertoire de configuration applicatif

Nous avons le fichier d'environnement : **environnement.prod.ts** dont le contenu est :

```
export const environment = {
  production: true,
  apiLocation : 'api.ocpizza.com',
  logOutput : 'console'
};

```

4.2.3 - Ressources

Les ressources sont les fichiers d'assets.

4.2.4 - Déployement de l'application cliente

Rendez-vous sur https://console.firebase.google.com/ et sélectionnez **Nouveau projet**. Nommez le **ocpizza**.

Votre application doit être construite : lancez une invite de commandes dans le repertoire du projet et lancez

```
ng build --prod
```

Vous pouvez alors vérifier qu'un repertoire **dist** à bien été créé dans le projet.

Les étapes suivantes requièrent **npm** et **node.js.**

Installez firebase avec la commande suivante :

```
npm install -g firebase-tools
```

Cette commande est également utilisée pour mettre à jour les outils firebase.

Connectez vous à Firebase :

```
firebase login
```

Initialisez un projet Firebase :

```
firebase init
```

Répondez par **Y** à **"are you ready to proceed?"**.

Ensuite choisissez **Hosting** :

```
You're about to initialize a Firebase project in this directory:

  C:\Users\vaalp\workspace-angularjs\deploymentTest

Before we get started, keep in mind:

  * You are initializing in an existing Firebase project directory

? Are you ready to proceed? Yes
? Which Firebase CLI features do you want to set up for this folder? Press Space to select features, then Enter to confirm your choices.
 ( ) Database: Deploy Firebase Realtime Database Rules
 ( ) Firestore: Deploy rules and create indexes for Firestore
 ( ) Functions: Configure and deploy Cloud Functions
>(*) Hosting: Configure and deploy Firebase Hosting sites
 ( ) Storage: Deploy Cloud Storage security rules
```

Choisissez le repertoire par défaut : **ocpizza**.

Puis designez le répertoire public à utiliser (c'est le repertoire contenu dans votre **/dist**), saisissez :

dist/ocpizza.

```
? What do you want to use as your public directory? dist/deploymentTest
? Configure as a single-page app (rewrite all urls to /index.html)? Yes
? File dist/deploymentTest/index.html already exists. Overwrite? (y/N) N
```

Répondez par **Y** à **Configure as a single-page app?**

Répondez par **N** à **Overwrite?**

Losrque **Firebase initialisation complete !** s'affiche lancez :

firebase deploy

Après quelques secondes l'opération devrait être réalisée (cf figure suivante)

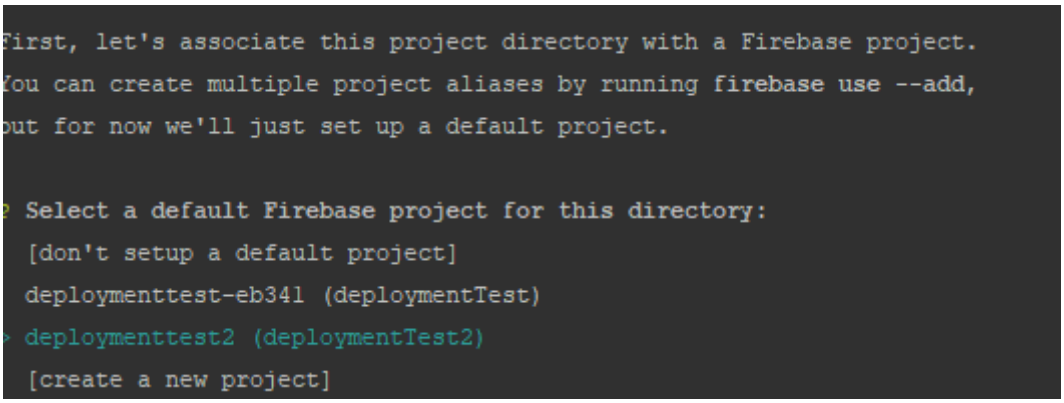
```
C:\Users\vaalp\workspace-angularjs\deploymentTest>firebase deploy

=== Deploying to 'deploymenttest-eb341'...

i  deploying hosting
i  hosting[deploymenttest-eb341]: beginning deploy...
i  hosting[deploymenttest-eb341]: found 7 files in dist/deploymentTest
+  hosting[deploymenttest-eb341]: file upload complete
i  hosting[deploymenttest-eb341]: finalizing version...
+  hosting[deploymenttest-eb341]: version finalized
i  hosting[deploymenttest-eb341]: releasing new version...
+  hosting[deploymenttest-eb341]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/deploymenttest-eb341/overview
Hosting URL: https://deploymenttest-eb341.firebaseapp.com
```

Le déploiement est terminé. Des liens d'accès vers la console de gestion et vers l'application vous sont alors donnés. Il peuvent différer de ceux donnés dans le paragraphe 4.2.6. Dans ce cas la, notez les : ce sont eux qu'il faut utiliser.

4.2.5 - Déployement de l'application professionnelle

Répétez la procedure pour en commençant par crée un nouveau projet **ocpizza-pro**. Remplacez les occurences de **ocpizza** par **ocpizza-pro**.

Après l'étape de hosting, vous aurez à choisir le projet firebase sur lequel travailler :



Choisissez **ocpizza-pro**.

La procèdure se déroule comme au 4.2.4.

4.2.6 - Vérifications

Afin de vérifier le bon déploiement de l'application, se rendre sur https://ocpizza.firebaseapp.com, l'application de type "single page application" devrait être disponible.

Vous avez également la possibilité d'accèder à la vue d'ensemble de l'applicaction sur console.firebase.google.com, ici vous avez le status qui dévrait être à l'état **déployé**.

4.3 - Déploiement de l'application Web Spring MVC

4.3.1 - Artefact

ocpizza-webapp-1.0.0-RELEASE.jar

4.3.2 - Répertoire de configuration applicatif

Fichier application.properties

Le fichier application.properties regroupe l'ensemble des propiétés relatives à cette application. Il est interne au fichier jar.

Si nécessaire, on pourra écraser certaines propriétés en plaçant un fichier application.properties dans le répertoire du jar. Spring Boot analysera les propriétés trouvées au démarrage du jar, et ces dernières auront la priorité sur les propriétés du application.properties interne.

Ceci devrait permettre d'adapter, par exemple, l'URI vers le WebService REST.

4.3.3 - Ressources

Les ressources sont internes au fichier jar.

4.3.4 - Déploiement

Se référer au 4.1.4, il suffit d'ajuster le contenu du fichier **jar.json**.

4.3.5 - Vérifications

Afin de vérifier le bon déploiement de l'application, se rendre sur https://ocpizza.com, la page d'index devrait être disponible.

Vous avez également la possibilité d'accèder à la vue d'ensemble sur clever-cloud.com et vous rendre sur Activity.

5 - Procédure de démarrage / arrêt

5.1 - Base de données

La base de données est automatiquement démarrée par Clever-cloud.com.

On ne peut arrêter de façon simple la base de données.

5.2 - Webservice REST

Démarrage : sur la vue d'ensemble de la console clever-cloud.com, choisissez "START" .

Arrêt : choisissez "STOP APP".

5.3 - Application web Spring MVC

Démarrage : sur la vue d'ensemble de la console clever-cloud.com, choisissez "START" .

Arrêt : choisissez "STOP APP".

5.4 - Applications Angular pour clients et professionnels

Le démarrage est automatique lors du déploiement.

Pour stoppez le projet souhaité, dans une invite de commandes :

```
firebase use --add
```

Maintenant choisissez le projet firebase à arrêter. Puis :

```
firebase hosting::disable
```

Pour redémarrer l'application il faut refaire un **deploy**.

6 - Procédure de mise à jour

6.1 - Base de données

La mise à jour du système de base de données (postgresql) est assurée par l'hébergeur.

Pour mettre à jour le schéma de la base de données, rendez-vous dans la console de gestion de Clever-cloud.com, sélectionnez "Personnal space", puis la base de données en question (nom par défaut : "ocpizza"). Ici vous pouvez administrer la base de données.

Ensuite, l'option PG Studio vous permettra d'executer des requêtes SQL sur la base de données.

On peut également accèder au CLI psql grace à la commande suivante :

```
psql -h db-ocpizza-postgresql.services.clever-cloud.com -p 5432 -U admin -d changeit
```

6.2 - Webservice REST

Dans la console de gestion, sélectionnez "START LAST PUSHED COMMIT".

6.3 - Application web Spring MVC

Dans la console de gestion, sélectionnez "START LAST PUSHED COMMIT".

6.4 - Applications Angular

On effectura la mise à jour de la même façon que décrit au 5.4.

7 - Supervision/Monitoring

7.1 - Base de données

Dans la console de gestion clever-cloud.com vous pouvez consulter les logs et les metrics de la base de données.

7.2 - Supervision du Webservice REST

Dans la console de gestion vous pouvez consulter les logs et les metrics de l'application.

7.3 - Supervision de l'application web Spring MVC

Dans la console de gestion vous pouvez consulter les logs et les metrics de l'application.

7.4 - Supervision des applications Angular

Dans la console de gestion vous pouvez consulter l'activité de l'application.

8 - Procédure de sauvegarde et restauration

8.1 - Sauvegardes

Concernant la base de données, une sauvegarde quotidienne est effectuée, chaque sauvegarde est conservée 5 jours (durée configurable). Un sauvegarde peut être effectuée à n'importe quel moment en sélectionnant “Request new backup” dans “addon dashboard”.

8.2 - Restaurations

Pour la restauration de la bdd, dans une invite de commande, entrez la commande suivante :

```
pg_restore -h db-ocpizza-postgresql.services.clever-cloud.com -p 5432 -U admin -d changeit --format=c YOUR_BACKUP_FILE
```