OC Pizza

IT Consulting & Development

# OC PIZZA

# Système de gestion pour pizzerias

## Dossier d'exploitation

## Version 1.1

**Auteur**
Yann Rouzaud
Analyste - Programmeur

# TABLE DES MATIÈRES

# 1. VERSIONS

| Auteur | Date | Description | Version |
|---|---|---|---|
| Yann Rouzaud | 01/01/2023 | Création du document | 1.0 |
| Yann Rouzaud | 17/01/2023 | Modification du document | 1.1 |
| | | | |
| | | | |

# 2. INTRODUCTION

## 2.1. Objet du document

Le présent document constitue le dossier d'exploitation de l'application d'OC PIZZA. Il est destiné aux équipes technique et de maintenance du client

Ce document a pour objectif de décrire la mise en exploitation du système informatique d'OC PIZZA. Il contient la description du déploiement et des méthodes de maintenance de la solution.

## 2.2. Références

Pour de plus amples informations, se référer :

1. **DCT - 001** : Dossier de conception fonctionnelle de l'application
2. **DCT - 002** : Dossier de conception technique de l'application

# 3.PRÉ-REQUIS

## 3.1.Système

### *3.1.1.Prestataire*

Que ce soit pour le serveur web, le serveur de base de données ou le serveur de fichier, le service choisi est celui d'Heroku détenu par Salesforce. Heroku est un plateforme d'hébergement cloud qui permet de déployer, exécuter et gérer des applications web. Les spécifications techniques varient en fonction des différents types d'applications et des différents plans de tarification. Heroku prend en charge plusieurs bases de données, notamment PostgreSQL.

### *3.1.2.Tarification*

L'option la plus adaptée pour une application web de gestion des commandes d'un restaurant est l'offre "Standard" de Heroku. L'offre Standard permet de gérer un volume de commandes élevé.

L'offre Standard de Heroku comprend plusieurs options différentes, notamment Standard 1X, Standard 2X et Performance M.

L'offre Standard 2X est puissante et convient aux applications qui gèrent un volume de commandes et de trafic élevé. Le tarif de cette offre est de **0.06$ par heure, soit 50$ par mois**.

Il est possible de mettre à niveau ou de rétrograder facilement entre les différentes offres en fonction des besoins de l'application.

### *3.1.3.Caractéristiques techniques*

Voici les caractéristiques techniques de l'option Standard 2X d'offre Standard de Heroku :

| | **Standard 2X** |
|---|---|
| **RAM** | 1GB |
| **Deploy with Git** | Yes |
| **Automated OS patching** | Yes |
| **Unified logs** | Yes |
| **Number of process types** | Yes |
| **Always on** | Yes |

| | Standard 2X |
|---|---|
| **Custom domains** | Yes |
| **Free SSL on custom domains** | Yes |
| **Automated Certificate Management on custom domains** | Yes |
| **Horizontal scaling** | Yes |
| **Preboot** | Yes |
| **Language runtime metrics** | Yes |
| **Treshold alerting** | Yes |
| **Combining multiple dyne types** | Can combine with Performance dynes |
| **Price** | Max $50 per month |

### *3.1.4.Informations de connexion*

Pour se connecter au compte client, rendez-vous à l'dresse suivante :

*https://id.heroku.com/login*

Les identifiants de connexion demandés par le site sont :

*Identifiant : OC_pizza_system_2023*
*Mot de passe : HTdfo-36529-te%op4*

## 3.2.Bases de données

Les bases de données et schémas suivants doivent être accessibles et à jour sur le repository du compte Github de la pizzeria :

*https://github.com/ocpizzasystem/dumps*

Pour installer la base de données, vous pouvez vous référer à la documentation du prestataire choisi pour la location du serveur, à l'adresse suivante :

*https://devcenter.heroku.com/categories/heroku-postgres*

## 3.3.Web-services

Pour le bon fonctionnement du système il sera nécessaire d'implémenter certains web-services. Ces web-services inclu l'utilisation de « plug-ins ». Voici les web-services important :

- Heroku utilise le service de paiement Stripe pour prendre en charge les paiements des clients pour ses services.
- Heroku prend en charge l'utilisation de services d'authentification tels que Auth0 ou Firebase Auth pour gérer les utilisateurs et les sessions.
- Heroku prend en charge l'utilisation de services d'analyse tels que Google Analytics ou Mixpanel pour suivre les utilisateurs et les performances de l'application.

# 4.PROCÉDURE DE DÉPLOIEMENT

## 4.1.Déploiement de la base de données

### *4.1.1.Artefacts*

#### *4.1.1.1.Dumps*

Les dumps de la base de données sont disponibles à l'adresse suivante :

*https://github.com/ocpizzasystem/dumps*

Les dumps fournis via ce lien sont les suivants :

- **data.sql** : fausses données de tests
- **structure.sql** : structure de la base
- **dataAndStructure.sql** : structure et fausses données de la basse de données

#### *4.1.1.2.Serveur Vapor*

Les fichiers Vapor pour le back-end sont disponibles à l'adresse suivante :

*https://github.com/ocpizzasystem/vaporapi*

#### *4.1.1.3.Importation de la base de données*

Il existe plusieurs façons d'importer une base de données PostgreSQL qui est disponible dans un dépôt GitHub avec Heroku. Voici une méthode courante :

1. Téléchargez le fichier de base de données à partir du dépôt GitHub.

2. Installez Homebrew en utilisant la commande :
   ```
   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)
   ```

3. Installez le CLI Heroku :
   ```
   brew tap heroku/brew && brew install heroku
   ```

4. Connectez-vous à Heroku en utilisant la commande :
   ```
   heroku login
   ```

5. Initialisez un dépôt git pour votre application en utilisant la commande :
   ```
   git init
   ```

6. Créez une nouvelle application Heroku en utilisant la commande :
   `heroku create <nom_application>`

7. Ajoutez le dossier de l'application Vapor au dépôt git

8. Configurer le buildpack pour apprendre à heroku comment gérer Vapor :
   `heroku buildpacks:set vapor/vapor`

9. Ajoutez un fichier **Procfile** à la racine du projet qui contient la commande pour lancer votre serveur Vapor :
   `web: Run --env=production --hostname=0.0.0.0 --port=$PORT`
   (0.0.0.0 utilise toutes les addresses IP disponibles et PORT est fourni par Heroku)

10. Ajoutez un fichier .build-env à la racine du projet Vapor qui contient les informations de configuration de votre application Vapor. Il doit être formaté comme ceci :
```
CLE_DATABASE_URL=postgres://user:password@host:port/database
CLE_API_KEY=ma_cle_api
CLE_SECRET=mon_secret
```

11. Déployez votre application en utilisant la commande :
   `git push heroku master`

12. Configurez les variables d'environnement en utilisant la commande :
   `heroku config:set`

13. Ajoutez l'addon Heroku Postgres en utilisant la commande :
   `heroku addons:create heroku-postgresql:<plan>`
   (remplacez `<plan>` par le plan `standard-4`).

14. Connectez-vous à la base de données Heroku en utilisant la commande :
   `heroku pg:psql`

15. Utilisez la commande `\i <nom_fichier.sql>` pour importer le fichier SQL dans la base de données Heroku.

16. Vérifiez que les données ont été importées correctement en utilisant des commandes SQL pour sélectionner des données de la base de données.

Voici la documentation détaillée de Heroku pour déployer une base de données :

*https://devcenter.heroku.com/categories/reference#data-management*

## 4.2.Déploiement du serveur de fichiers

### *4.2.1.Caractéristiques techniques*

#### *4.2.1.1.S3 Bucket de AWS*

Un serveur de fichiers S3 Bucket est utilisé pour stocker des fichiers statiques et des données de médias dans un projet web. Il est généralement utilisé pour stocker des images, des vidéos, des fichiers audio, des documents et d'autres types de fichiers qui ne nécessitent pas de traitement côté serveur.

Ce serveur de fichiers servira donc à stocker, entre autre, les cartes de chaque restaurant, les images des plats, les images des restaurants, les images des chefs pizzaiolos, les vidéos de présentation des restaurants, etc.

#### *4.2.1.2.Informations de connexion*

Pour se connecter au compte client, rendez-vous à l'dresse suivante :

*https://signin.aws.amazon.com/*

Les identifiants de connexion demandés par le site sont :

*Identifiant : OC_pizza_system_2023*
*Mot de passe : JTdfo-36529-te%op0*

### *4.2.2.Artefacts*

#### *4.2.2.1.Fichiers*

Tous les fichiers fournis par OC PIZZA sont centraliser dans le repository Github dédié à l'adresse suivante :

*https://github.com/ocpizzasystem/files*

#### *4.2.2.2.Ajouter des fichiers au serveur S3 Bucket*

Il existe plusieurs façons d'ajouter des fichiers à un serveur de fichiers S3 Bucket, voici une méthode possible :

1. Connectez vous à votre compte AWS
2. Accédez à la console S3 en cliquant sur "Services" puis "S3" dans le menu de gauche.
3. Sélectionnez le bucket dans lequel vous souhaitez ajouter des fichiers.
4. Cliquez sur le bouton "Téléverser" pour ouvrir la fenêtre de téléchargement.
5. Utilisez le bouton "Ajouter des fichiers" pour sélectionner les fichiers à ajouter à partir de votre ordinateur ou glissez-déposez les fichiers directement dans la fenêtre de téléchargement.
6. Cliquez sur le bouton "Téléverser" pour commencer le téléchargement des fichiers.
7. Une fois le téléchargement terminé, vous devriez voir les fichiers ajoutés dans votre bucket S3.

Il est également possible d'ajouter des fichiers à un serveur de fichiers S3 Bucket en utilisant l'API S3 ou un outil de ligne de commande tel que aws-cli. Ces outils permettent de gérer les fichiers de manière automatisée et de les transférer en masse dans S3.

#### *4.2.2.3.Informations d'accès à S3 Bucket*

Pour obtenir les informations d'accès à un compte S3 Bucket, vous devrez suivre les étapes ci-dessous :

1. Connectez-vous à votre compte AWS.
2. Accédez à la page de gestion des utilisateurs IAM en cliquant sur "Services" puis "IAM" dans le menu de gauche.
3. Dans la section "Utilisateurs", cliquez sur "Ajouter un utilisateur ».
4. Donnez un nom à l'utilisateur et cochez la case "Accès programmatique" pour générer des clés d'accès.
5. Attribuez les autorisations nécessaires à l'utilisateur pour accéder à S3 en utilisant les rôles ou les stratégies.
6. Cliquez sur "Créer utilisateur" pour créer l'utilisateur.

7. Une fois l'utilisateur créé, vous devriez voir les clés d'accès et la clé secrète qui ont été générées pour cet utilisateur. Téléchargez ou copiez-collez ces informations dans un endroit sûr, car vous ne pourrez plus les consulter une fois que vous quitterez cette page.

8. Utilisez ces informations d'accès (clé d'accès et clé secrète) pour accéder au compte S3 Bucket via l'API ou un outil de ligne de commande comme aws-cli.

#### *4.2.2.4.Importer le serveur de fichiers S3 Bucket*

Il existe plusieurs façons de déployer un serveur de fichiers S3 Bucket avec Heroku, mais voici une méthode possible :

1. Rendez vous dans le projet Heroku.

2. Accédez à la section "Settings" de votre projet et cliquez sur "Reveal Config Vars" pour afficher les variables d'environnement actuellement configurées.

3. Ajoutez les variables d'environnement nécessaires pour votre projet, comme les informations d'accès à votre compte S3 (clé d'accès, clé secrète, nom de l'espace de stockage, etc.). Utilisez les noms de variables appropriés pour votre application, comme S3_ACCESS_KEY, S3_SECRET_KEY, S3_BUCKET et S3_REGION

4. Cliquez sur "Save" pour enregistrer les modifications.

5. Ajoutez un fichier `config.ru` à votre projet pour définir la configuration de la pile Rack pour votre serveur de fichiers S3. Il devrait ressembler à ceci:

```
require 'aws-sdk-s3'

s3 = Aws::S3::Resource.new(
  region: ENV['S3_REGION'],
  access_key_id: ENV['S3_ACCESS_KEY'],
  secret_access_key: ENV['S3_SECRET_KEY']
)

bucket = s3.bucket(ENV['S3_BUCKET'])

run lambda { |env|
  [200, { 'Content-Type' => 'text/plain' },
[bucket.objects.map(&:key).join("\n")]]
}
```

6. Ajoutez un fichier `Gemfile` au projet pour définir les dépendances nécessaires, comme `aws-sdk-s3`. Il devrait ressembler à ceci:

```
source 'https://rubygems.org'

gem 'aws-sdk-s3'
```

7. Déployez votre projet sur Heroku en utilisant la commande `git push heroku master`.

8. Vérifiez que votre serveur de fichiers S3 est en cours d'exécution en accédant à l'URL de votre application Heroku. Il devrait afficher la liste des objets stockés dans votre espace de stockage S3.

Remarque: Il est important de protéger les données stockées dans S3 en utilisant des stratégies de sécurité appropriées (comme les règles de sécurité de l'objet S3 et les stratégies de contrôle d'accès) pour éviter les accès non autorisés.

## 4.2.Déploiement de l'Application Web

### *4.2.1.Artefacts*

Le code source de l'application est disponible à l'adresse suivante :

*https://github.com/ocpizzasystem/frontend*

### *4.2.2.Importation du code de l'application*

Pour déployer le front-end de votre application sur un S3 Bucket en utilisant Heroku, vous pouvez suivre les étapes suivantes:

1. Se rendre dans le bucket S3 dans la console AWS.

2. Configurez les paramètres du bucket pour rendre les fichiers publics et permettre les requêtes cross-origin.

3. Se rendre dans le dépôt Git de l'application Heroku.

4. Utiliser Git pour envoyer le code sur Heroku.

5. Utilisez la commande `heroku buildpacks:set` pour définir les buildpacks nécessaires à votre application, comme `heroku-buildpack-aws-cli` pour installer l'outil CLI AWS.

6. Utilisez la commande `heroku run` pour exécuter une tâche de déploiement personnalisée qui utilise l'outil CLI AWS pour téléverser les fichiers de votre application sur le bucket S3.

### *4.2.3.Environnement de l'application web*

#### *4.2.3.1.Variables d'environnement*

Aucune variable d'environnement n'a été créée. Pour créer les variables d'environnement, vous devez utiliser la commande `heroku config:set`.

Vous pouvez également définir des variables d'environnement pour votre application en utilisant l'interface web Heroku. Allez dans le dashboard de votre application, cliquez sur "Settings" puis sur "Reveal Config Vars" pour ajouter ou modifiez les variables existantes.

Voici les de variables d'environnement à définir :

- **`DATABASE_URL`**: l'URL de votre base de données PostgreSQL. Cette variable est nécessaire pour se connecter à la base de données depuis votre application.
- **`AWS_ACCESS_KEY_ID`** et **`AWS_SECRET_ACCESS_KEY`**: les identifiants d'accès à utiliser pour se connecter à l'API AWS pour le stockage de fichiers statiques.

### 4.2.4.Répertoire de configuration applicatif

Les fichiers de configuration qui se situent dans l'application Heroku, et l'arborescence de celle-ci serait la suivante :

```
.
├── Procfile
├── Package.swift
├── Sources/
│   ├── App/
│   │   ├── Controllers/
│   │   ├── Models/
│   │   ├── Middleware/
│   │   ├── Public/
│   │   ├── Resources/
│   │   ├── Routes/
│   │   ├── Utils/
│   │   ├── configure.swift
│   │   └── main.swift
│   ├── Run/
│   └── Tests/
├── Public/
│   ├── styles/
│   ├── scripts/
│   └── images/
├── Config/
│   ├── nginx.conf
│   ├── postgresql.conf
│   ├── secrets/
│   └── ssl/
├── Resources/
│   ├── Views/
│   └── migrations/
├── .gitignore
└── README.md
```

### 4.2.4.DataSources

Il existe plusieurs façons d'accéder aux sources de données d'une application web déployée sur Heroku avec Vapor, voici quelques exemples :

1. Utiliser l'interface web de Heroku : Allez dans le dashboard de votre application, cliquez sur "Resources" puis sur "Heroku Postgres" pour accéder à la base de données de votre application. Vous pouvez utiliser cette interface pour effectuer des opérations CRUD (create, read, update, delete) sur vos données.

2. Utiliser la ligne de commande : Utilisez la commande `heroku pg:psql` pour se connecter à votre base de données Postgres et effectuer des requêtes SQL.

3. Utiliser un outils de visualisation de données : Il existe des outils tel que **DBeaver** pour se connecter à votre base de données Heroku et vous permettent de visualiser et de manipuler vos données de manière graphique. Voici le lien vers la documentation :

*https://dbeaver.com/2022/03/03/how-to-create-database-connection-in-dbeaver/*

### *4.2.5. Vérifications*

Le déploiement s'est correctement déroulé si vous avez accès au système informatique à partir de l'adresse :

*https://www.ocpizza.fr*

# 5.PROCÉDURE DE DÉMARRAGE / ARRÊT

## 5.1.Base de données

### *5.1.1.Démarrage*

Comme vous utilisez Heroku pour gérer votre base de données, il n'y a rien à faire de particulier, la base de données sera automatiquement démarrée lorsque vous déployez votre application.

### *5.1.2.Arrêt*

Si vous utilisez Heroku pour gérer votre base de données, vous pouvez utiliser la commande `heroku pg:stop` pour arrêter la base de données. Cela vous permettra de réduire les coûts liés à l'utilisation de la base de données lorsque vous n'utilisez pas votre application.

## 5.2.Application Web

### *5.2.1.Démarrage*

Placer dans l'application Heroku et tapez la commande suivante :
`heroku restart`

Cela va redémarrer toutes les dynos de l'application, y compris les dynos web et les dynos de tâche en arrière-plan. Cela peut être utile pour rafraîchir l'application après des mises à jour de code ou pour réinitialiser des états de l'application qui ont été modifiés.

Il est important de noter que cela ne va pas réinitialiser les variables d'environnement ou les configurations de l'application.

### *5.2.2.Arrêt*

Placer dans l'application Heroku et tapez la commande suivante :
`heroku ps:scale web=0` (Cela va redimensionner la dyno web à 0)

Vous pouvez également utiliser la commande `heroku apps:destroy` pour supprimer complètement l'application de Heroku.

# 6.PROCÉDURE DE MISE À JOUR

## 6.1.Base de données

Pour mettre à jour la base de données, vous pouvez suivre les étapes suivantes:

1. Tout d'abord, créez une sauvegarde de votre base de données actuelle en utilisant la commande `Heroku pg:backups:capture`
2. Ensuite, effectuez les modifications nécessaires à votre schéma de base de données.
3. Utilisez la commande `Heroku run` pour exécuter les migrations de base de données sur votre application déployée.
4. Testez votre application pour vous assurer qu'elle fonctionne correctement avec la nouvelle base de données.
5. Enfin, si tout fonctionne bien, utilisez la commande `Heroku pg:backups:restore` pour restaurer votre sauvegarde de base de données sur votre base de données en direct.

## 6.2.Application web

Pour mettre à jour l'application web, voici les étapes générales à suivre :

1. Mettez à jour le code de votre application en local en utilisant Git pour gérer les versions.
2. Utilisez la commande `git push` pour envoyer les mises à jour sur votre dépôt distant Heroku.
3. Utilisez la commande `heroku logs --tail` pour surveiller les erreurs pendant le déploiement.
4. Utilisez la commande `heroku run` pour démarrer des tâches de maintenance, comme la mise à jour de la base de données.
5. Utilisez la commande `heroku restart` pour redémarrer l'application après les mises à jour.

Il est vivement conseillé de revenir vers nous pour effectuer une modification

# 7.SUPERVISION/MONITORING

## 7.1.Supervision de l'application web

Heroku propose plusieurs outils intégrés pour superviser et surveiller les applications web hébergées sur sa plateforme.

1. **Heroku Dashboard**: Il permet de surveiller les erreurs, les performances, les requêtes et les utilisateurs sur votre application. Il vous donne une vue d'ensemble de l'état de votre application, y compris le nombre de requêtes, les temps de réponse, le taux d'erreur, l'utilisation de la mémoire et du CPU, et bien plus encore.

2. **Heroku Metrics**: Il permet de surveiller l'utilisation des ressources de votre application, telles que la mémoire et la CPU, pour détecter les problèmes potentiels de performance. Il vous permet également de surveiller les tendances d'utilisation pour aider à planifier les mises à niveau de ressources.

3. **Heroku Logs**: Il permet de surveiller les journaux de votre application pour détecter les erreurs et résoudre les problèmes rapidement. Il vous permet de rechercher, filtrer et consulter les journaux de votre application en temps réel.

4. **Heroku Add-ons**: Il permet d'intégrer des outils tiers de surveillance tels que New Relic, Loggly, Papertrail, et Uptime Robot pour surveiller les performances, les journaux et la disponibilité de votre application.

En utilisant ces outils, vous pouvez surveiller régulièrement l'état de votre application, détecter les problèmes potentiels et les résoudre rapidement afin de garantir une expérience utilisateur optimale. Pour plus d'informations, vous pouvez vous référer à la documentation de Heroku aux adresses suivantes :

*https://devcenter.heroku.com/articles/heroku-dashboard*

*https://devcenter.heroku.com/categories/monitoring-metrics*

*https://devcenter.heroku.com/articles/production-check*

*https://devcenter.heroku.com/articles/logging*

*https://elements.heroku.com/addons#monitoring*

# 8. PROCÉDURE DE SAUVEGARDE ET RESTAURATION

Pour sauvegarder une base de données, vous pouvez utiliser la commande `heroku pg:backups capture` pour créer une sauvegarde de la base de données. Cette sauvegarde sera stockée sur les serveurs Heroku et pourra être téléchargée plus tard.

Pour restaurer une base de données, vous pouvez utiliser la commande `heroku pg:backups restore` suivie du nom de la sauvegarde que vous souhaitez restaurer. Cela remplacera la base de données actuelle par la sauvegarde sélectionnée. Il est important de noter que cela effacera toutes les données existantes dans la base de données.

Il est également possible de planifier des sauvegardes automatiques de la base de données en utilisant l'**addon Heroku Scheduler** pour exécuter la commande de sauvegarde à intervalle régulier.

Il est important de noter que ces commandes nécessitent un accès à la ligne de commande de Heroku et les privilèges d'administration pour accéder à la base de données. Il est recommandé de conserver les sauvegardes en dehors de Heroku, comme sur un stockage distant sécurisé.