

# OC PIZZA

## P9

### Dossier d'exploitation

Version 1

**Auteur**

Ewen Jeannenot

*Analyste-programmeur*



**IT Consulting & Development**

## TABLE DES MATIERES

<b>1 - Versions .....</b>	<b>5</b>
<b>2 - Introduction .....</b>	<b>6</b>
2.1 - Objet du document .....	6
2.2 - Références.....	6
<b>3 - Prérequis.....</b>	<b>7</b>
3.1 - Système .....	7
3.1.1 - Hébergement .....	7
3.1.2 - Serveur Web .....	7
3.1.2.1 - Caractéristiques techniques .....	7
3.2 - Bases de données.....	8
3.2.1 - Serveur de Base de données .....	8
3.2.1.1 - Caractéristiques techniques .....	8
3.3 - Web-services .....	8
3.4 - Autres Ressources .....	8
<b>4 - Procédure de déploiement.....</b>	<b>9</b>
4.1 - Déploiement du serveur .....	9
4.1.1 - Console hébergeur Scaleway Dedibox .....	9
4.1.2 - Configuration.....	10
4.1.2.1 - Package.....	10
4.1.2.2 - PostgreSQL.....	11
4.2 - Déploiement de l'Application Web .....	12
4.2.1 - Application .....	12
4.2.2 - Environnement virtuel et dépendances .....	12
4.2.2.1 - Variables d'environnement .....	12
4.2.3 - Configuration applicatif Nginx / Unicorn.....	12
4.2.3.1 - Fichier web.sh.....	13
4.2.4 - Ressources.....	13
4.2.5 - Vérifications .....	16
<b>5 - Procédure de démarrage / arrêt .....</b>	<b>17</b>
5.1 - Serveur .....	17
5.2 - Base de données .....	17
5.3 - Application web .....	17
<b>6 - Procédure de mise à jour .....</b>	<b>18</b>
6.1 - Base de données .....	18
6.2 - Application web .....	18
<b>7 - Supervision/Monitoring.....</b>	<b>19</b>
7.1 - Supervision de l'application web .....	19
<b>8 - Procédure de sauvegarde et restauration .....</b>	<b>20</b>
<b>9 - Glossaire .....</b>	<b>21</b>



**IT Consulting & Development**

# 1 - VERSIONS

Auteur	Date	Description	Version
Ewen	16/01/2022	Création du document	001
Ewen	19/01/2022	Ajout de la procédure de sauvegarde	002

## 2 - INTRODUCTION

### 2.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application « OC Pizza »

L'objectif du document est de présenter un dossier d'exploitation à l'attention de l'équipe technique du client, lui permettant d'assurer l'exploitation de la solution proposée et de pouvoir réagir de manière appropriée lorsqu'un problème surgit.

### 2.2 - Références

Pour de plus amples informations, se référer également aux éléments suivants :

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

## 3 - PREREQUIS

### 3.1 - Système

L'application sera déployée sur un serveur linux avec le système d'exploitation Ubuntu server, sans interface graphique

#### *3.1.1 - Hébergement*

L'hébergement se fera chez Scaleway Dedibox et le nom de domaine pour l'application a aussi été réservé chez eux : **oc-pizza.fr**

#### *3.1.2 - Serveur Web*

Le serveur est un serveur dédié, ce qui laisse plus de liberté pour l'installation de services ou d'applications.

##### *3.1.2.1 - Caractéristiques techniques*

L'offre dedibox Pro-6-XS :

- **Modèle de châssis**  
Supermicro MBD-X10SDE-DF
- **Processeur**  
1× Intel® Xeon® D-1531  
CPU - 6C/12T - 2.2 GHz
- **Stockage**  
2 × 500 Go SSD
- **Mémoire**  
32 Go DDR4 ECC
- **Réseau public**  
Illimité
- **Bande passante**  
500 Mbps

## 3.2 - Bases de données

Pour la base de données et le schéma, se référer au DCT-001 précédemment abordé.

### 3.2.1 - Serveur de Base de données

Le serveur de base de données hébergeant la base de données de l'application sera sur le même serveur que le serveur web.

Nous utiliserons :

- **PostgreSQL** : version 12.9

#### 3.2.1.1 - Caractéristiques techniques

Les caractéristiques du serveur sont les mêmes que pour le serveur hébergeant l'application et couvrent largement les besoins de PostgreSQL.

## 3.3 - Web-services

Les web services suivants doivent être accessibles et à jour :

L'api pour le paiement

- **SQUARE** est une solution qui propose des tarifs raisonnables et une offre presque complète qui conviendra à l'utilisation du client.

L'api pour la géolocalisation

- **Google Maps Api** qui permet la localisation des points de vente et adresses de livraison ainsi que l'affichage des itinéraires.

## 3.4 - Autres Ressources

L'hébergeur Dedibox propose de bon tarif et offre une assistance rapide en cas de problème mais le monitoring reste un peu limité ou en supplément à la charge d'un tiers. C'est pourquoi nous conseillons l'utilisation de deux services supplémentaires :

**New Relic** : comme outil de monitoring et de performance du serveur

**Sentry** : comme outil de monitoring de l'application django



## 4 - PROCEDURE DE DEPLOIEMENT

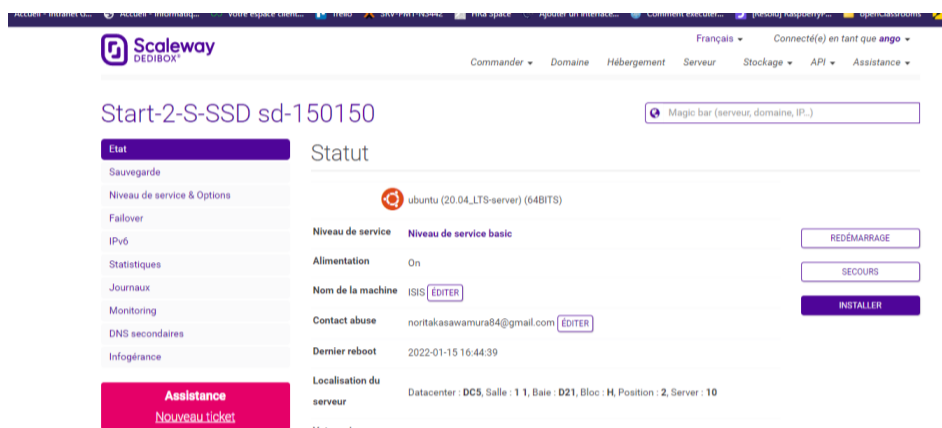
### 4.1 - Déploiement du serveur

#### 4.1.1 - Console hébergeur Scaleway Dedibox

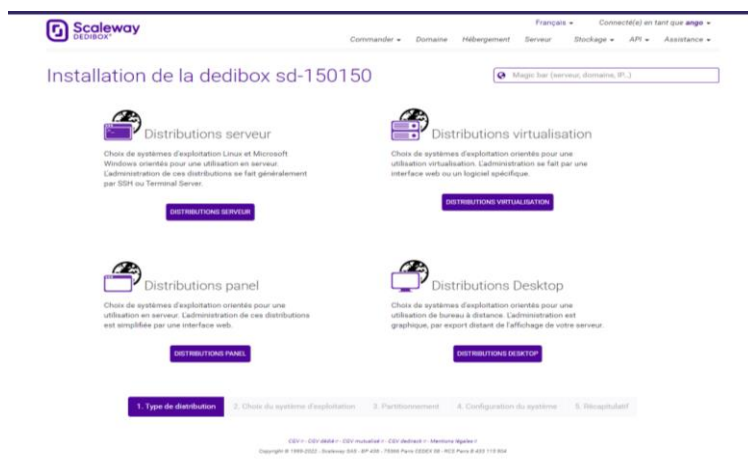
Le choix de laisser la configuration à l'équipe d'IT Consulting & Development appartient au client et si il souhaite laisser cette charge à son équipe technique, il trouvera l'ensemble de la documentation pour l'utilisation et la création d'un serveur sur Dedibox à l'adresse : <https://www.scaleway.com/en/docs/>

Une fois connecté à votre espace Dedibox :

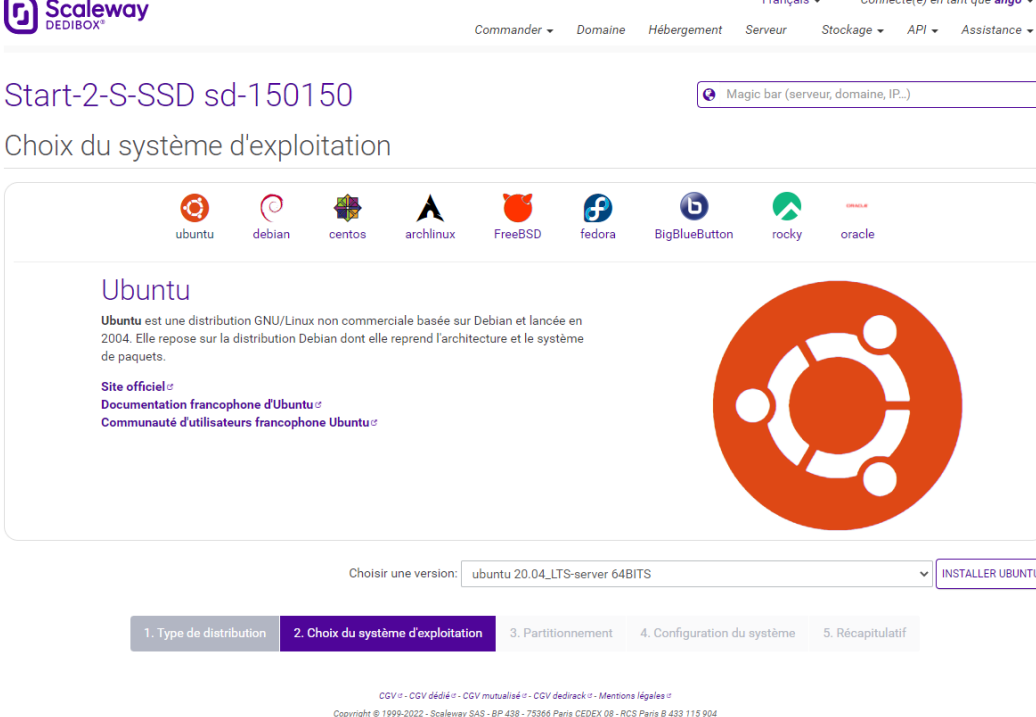
- On commence par la création d'un serveur en choisissant « installer »



- On choisit ensuite le type de distribution (orienté bureau, serveur, serveur avec interface graphique).



- On peut alors choisir la distribution parmi une liste. Pour le projet c'est la distribution Ubuntu Server.



- Dernière étape, la configuration de l'identifiant et du mot de passe pour la connexion SSH. Il sera aussi possible de renseigner une clef public pour permettre la connexion sans mot de passe depuis une machine définie.

## 4.1.2 - Configuration

- Maintenant que le serveur est installé et que l'accès SSH est configuré, on peut installer les services dont l'application va avoir besoin.
- L'application est à télécharger depuis le lien github fourni au client. Elle est au format .zip et est donc à dézipper avant utilisation.

### 4.1.2.1 - Package

On commence par mettre à jour les programmes du système :

```
$ sudo apt-get update
```

Ensuite on va installer les programmes et librairies utiles :

```
$ sudo apt-get install python3-pip python3-dev libpq-dev postgresql postgresql-contrib
```

Et enfin on installe le paquet Nginx (Gunicorn étant une librairie Python, il sera installé automatiquement avec les requirements) et Supervisor

```
$ sudo apt-get install nginx supervisor
```

#### 4.1.2.2 - PostgreSQL

Pour l'utilisation de la base de données avec l'application, il faut créer la base en elle-même et un utilisateur pour y accéder.

Utiliser le script « script\_database.sql » fourni avec l'application pour le créer automatiquement.

Il crée une base de données appelée ocpizza avec un utilisateur ocpizza ayant les droits dessus.

- **ATTENTION** - Si vous préférez créer vous-même la base et l'utilisateur il faudra penser à reporter les noms dans le fichier settings de l'application Django.

## 4.2 - Déploiement de l'Application Web

### 4.2.1 - Application

Pour la suite il faudra se déplacer à la racine du dossier de l'application dézippé

### 4.2.2 - Environnement virtuel et dépendances

Afin de ne pas être parasité par d'autres éléments, il conviendra de travailler dans un environnement virtuel. On installe donc Pipenv et on initialise un environnement virtuel :

```
Pip install pipenv
```

```
Pipenv shell
```

On peut maintenant installer les dépendances utiles à l'application à l'aide du fichier requirements.txt

```
pip install -r requirements.txt
```

#### 4.2.2.1 - Variables d'environnement

Il est fortement recommandé de ne pas mettre les informations d'authentification et autres clefs dans les fichiers de configuration mais plutôt sous forme de variable d'environnement.

Voici les variables d'environnement déclarées dans le fichier .env et reconnues par les fichiers de l'application oc-pizza :

Nom	Obligatoire	Description
'GOOGLEAPIKEY'	Fortement conseillé	Clef fournie par l'API Google lors de la création du projet sur leur plateforme
'DJANGOSECRETKEY'	Fortement conseillé	Clef que l'on peut générer nous même et qui est renseigné dans le fichier setting de l'application

### 4.2.3 - Configuration applicatif Nginx / Gunicorn

Pour le fichier de configuration de l'applicatif web vous pouvez utiliser le script web.sh fourni dans le fichier .zip.

- Le script va copier le fichier de configuration Nginx, ocpizza, dans le répertoire **/etc/nginx/sites-available** puis créer un **lien symbolique** vers le répertoire

### `/etc/nginx/site-enable`

Le fichier de configuration Nginx indique au serveur de rediriger toutes les requêtes web à l'attention de oc-pizza.fr vers l'application django

- Ensuite le script copiera le fichier de configuration de Supervisor dans `/etc/supervisor/conf.d/ocpizza.conf`

-ATTENTION- Il faut changer le chemin vers l'exécutable de Gunicorn dans le fichier de configuration car l'application est installée dans l'environnement virtuel créé pour python et le projet. Copier le chemin qui se trouve ici (en changeant les valeurs en gras) :

```
$ ls /home/nom_user/.virtualenvs/nom_environnement_virtuel/bin/gunicorn
```

et la reporter dans la ligne command du fichier de configuration.

Le fichier de configuration de supervisor va dire à Gunicorn de démarrer l'application django ocpizza et de la redémarrer automatiquement en cas d'arrêt

- Une fois les changements effectués dans les fichiers de configuration de Nginx et de Supervisor, il conviendra de redémarrer les deux services pour qu'ils prennent en compte les modifications.

```
$ sudo service nginx reload
```

```
$ sudo supervisorctl update
```

```
$ sudo supervisorctl reload
```

#### 4.2.3.1 - Fichier `web.sh`

Le fichier `web.sh` est un script shell qui va s'occuper de déplacer les fichiers de configuration de Nginx et Supervisor (pour Gunicorn) dans les bons répertoires après installation des différents paquet.

#### 4.2.4 - Ressources

**Sentry :**

Avant d'installer Sentry il faut se créer un compte sur leur site. Puis choisir « **Create a new project** »

Il faut choisir la plateforme à surveiller, dans notre cas : **Django**

## Create a new Project

Projects allow you to scope error and transaction events to a specific application in your organization. For example, you might have separate projects for your API server and frontend client.

### Choose a platform

Popular

Browser


Server

Mobile


Desktop

Serverless


All




ANDROID




IOS




.NET




ASP.NET CORE




FLUTTER




GO




JAVA




SPRING BOOT




JAVASCRIPT




ANGULAR




NEXT.JS




REACT




VUE




NODE.JS




EXPRESS




PHP




LARAVEL




PYTHON




DJANGO




FLASK



REACT NATIVE



RUBY



RAILS

### Set your default alert settings

☒ I'll create my own alerts later
 ☐ Alert me on every new issue
 ☐ When there are more than  occurrences of

### Give your project a name

Project name

Team

Ensuite Sentry nous fournit la marche à suivre pour initialiser la surveillance de l'application Django.

## Configure Django

This is a quick getting started guide. For in-depth instructions on integrating Sentry with Django, view [our complete documentation](#).

The Django integration adds support for the [Django Web Framework](#) from Version 1.6 upwards.

Install `sentry-sdk`:

```
pip install --upgrade sentry-sdk
```

To configure the SDK, initialize it with the Django integration in your `settings.py` file:

```
import sentry_sdk
from sentry_sdk.integrations.django import DjangoIntegration

sentry_sdk.init(
    dsn="https://a1dd5dd5327c4283a6092dd97a2c1603@o1092036.ingest.sentry.io/6156969",
    integrations=[DjangoIntegration()],

    # Set traces_sample_rate to 1.0 to capture 100%
    # of transactions for performance monitoring.
    # We recommend adjusting this value in production.
    traces_sample_rate=1.0,

    # If you wish to associate users to errors (assuming you are using
    # django.contrib.auth) you may enable sending PII data.
    send_default_pii=True
)
```

The above configuration captures both error and performance data. To reduce the volume of performance data captured, change `traces_sample_rate` to a value between 0 and 1.

You can easily verify your Sentry installation by creating a route that triggers an error:

```
from django.urls import path

def trigger_error(request):
    division_by_zero = 1 / 0

urlpatterns = [
    path('sentry-debug/', trigger_error),
    # ...
]
```

Visiting this route will trigger an error that will be captured by Sentry.

Il faut d'abord installer la librairie sentry :

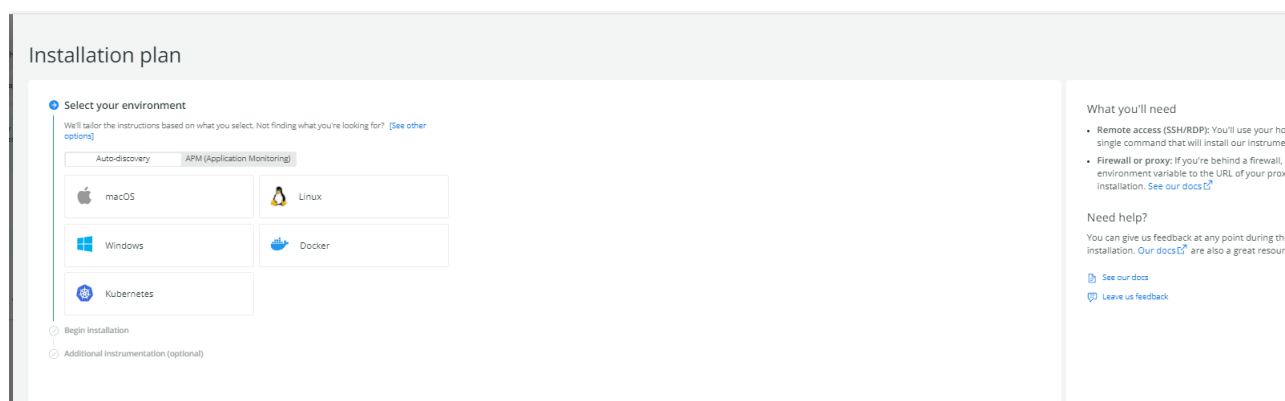
```
pip install --upgrade sentry-sdk
```

La configuration du fichier settings.py est déjà faite dans le livrable. Tout ce qu'il reste à faire est de modifier l'identifiant pour utiliser celui que Sentry vous donne. Pour cela vous devez copier la ligne **dsn** obtenue et remplacer celle dans le fichier **settings.py** de l'application Django

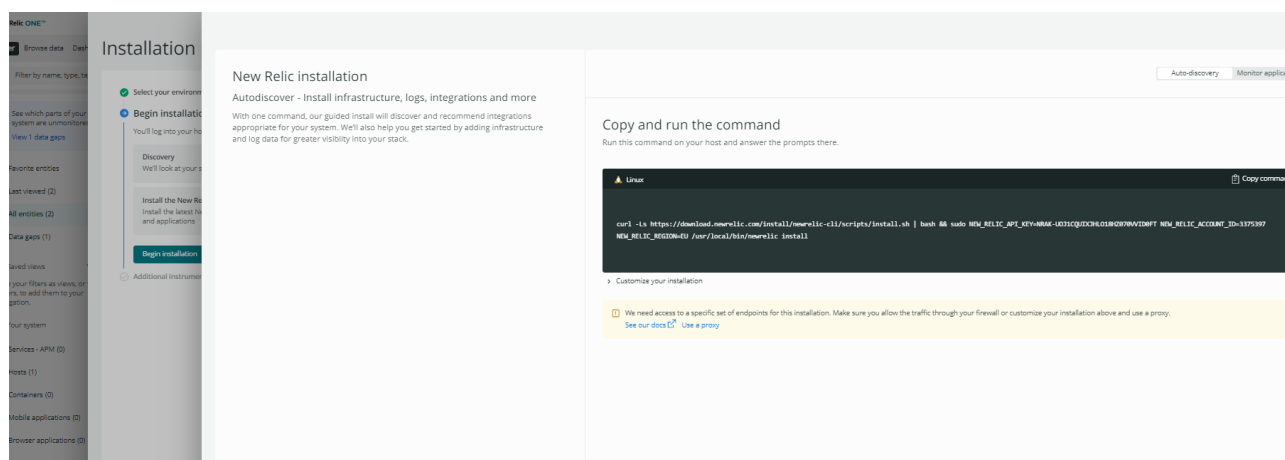
Pour approfondir votre utilisation de Sentry vous pouvez retrouver la documentation à l'adresse : <https://docs.sentry.io/platforms/python/guides/django/>

## New Relic :

Avant d'installer New Relic il faut se créer un compte sur leur site. Puis on choisit de remonter des données.



Après avoir choisi l'environnement de notre serveur, Linux dans notre cas, New Relic nous fournit la commande pour installer un agent sur le serveur qui se chargera de renvoyer les informations.



Il suffit alors de copier la commande sur le serveur et de l'exécuter.

New Relic crée par défaut une surveillance du serveur avec des alertes pour quatre points. Il est conseillé de rajouter une alerte pour l'espace disque et une pour l'utilisation de la mémoire.

Pour approfondir votre utilisation de New Relic, vous pouvez retrouver la documentation à l'adresse

<https://docs.newrelic.com/docs/new-relic-one/use-new-relic-one/cross-product-functions/install-configure/new-relic-guided-install-overview/>

### 4.2.5 - Vérifications

Afin de vérifier le bon déploiement de l'application, vous pouvez :

- Vérifiez que le service Gunicorn est bien lancé :

```
$ sudo supervisorctl status
```

- vous rendre à l'adresse :

<http://www.oc-pizza.fr>



## 5 - PROCEDURE DE DEMARRAGE / ARRET

### 5.1 - Serveur

On peut redémarrer le serveur depuis l'interface d'administration de l'hébergeur Dedibox mais ATTENTION cela a le même effet qu'une coupure de courant, le serveur s'éteint sans couper les différents services au préalable.

On peut aussi éteindre le serveur directement depuis la console en SSH :

Eteindre : `sudo shutdown -h now`

Redémarrer : `sudo shutdown -r now` ou `sudo reboot`

### 5.2 - Base de données

Par défaut, le serveur de base de données est démarré.

Pour le stopper on peut utiliser la commande :

`sudo service postgresql stop`

Pour démarrer le service :

`sudo service postgresql start`

### 5.3 - Application web

Comme nous avons utilisé Supervisor pour gérer le redémarrage automatique de l'application, il conviendra de commenter la ligne **autorestart** du fichier de configuration de supervisor avant d'arrêter l'application.

## 6 - PROCEDURE DE MISE A JOUR

**-ATTENTION-** Toute modification ou mise à jour sur un serveur en production est à préparer en amont avec si possible des tests effectués sur une machine présentant les mêmes caractéristiques afin de s'assurer de la bonne compatibilité des différentes versions de logiciels et de garder une continuité de service.

Tout package installer sur le serveur avec apt-get peut être mis à jour avec apt-get.

On commence par mettre à jour la liste des fichiers disponibles dans les dépôts avec `apt-get update`

Puis on met à jour le paquet : `apt-get install nom_du_paquet`

### 6.1 - Base de données

Nous recommandons au client de passer par l'interface d'administration de l'application Django pour toute intervention sur la base de données.

Si il y avait besoin de mettre à jour PostgreSQL, se référer à la note général ci-dessus en prenant soin de bien sauvegarder la base avant.

### 6.2 - Application web

La procédure de mise à jour de l'application web reprend la démarche de déploiement du document.

Il faudra dézipper le nouveau fichier fourni et exécuter les scripts pour mettre à niveau l'application.

## 7 - SUPERVISION/MONITORING

### 7.1 - Supervision de l'application web

Afin de tester que l'application web est toujours fonctionnelles, nous utiliserons les deux outils de monitoring présentés en « 3.4 autres ressources » et mis en place en « 4.2.4 ».

- **Sentry** pour le monitoring de l'application : en se connectant sur l'interface web de l'outil, <https://sentry.io/> , on peut consulter les logs et le suivi de fonctionnement de l'application.
- **New Relic** pour le monitoring du serveur : en se connectant sur l'interface web de l'outil, <https://one.eu.newrelic.com/> , on peut consulter les logs et l'activité du serveur en temps réel (utilisation de la charge CPU, des disques durs, de la bande passante...).

## 8 - PROCEDURE DE SAUVEGARDE ET RESTAURATION

### Sauvegarde :

Le script **web.sh** fournit avec le livrable et exécuté plus haut en « 4.2.3 configuration applicatif » crée une Tâche CRON qui va exécuter une sauvegarde quotidienne de la base de données avec une rotation sur 10 jours pour que les fichiers ne prennent pas trop de place.

Pour cela on utilise le petit utilitaire de PostgreSQL : `pg_dump`

### Restauration :

Pour restaurer une base de données depuis une sauvegarde il faut taper la commande :

```
Psql ocpizza < nom_de_la_sauvegarde
```

## 9 - GLOSSAIRE
