|  |
| --- |
| **OC PIZZA**  **Boutique en ligne**  **&**  **Gestionnaire centralisé**  Dossier d'exploitation  Version 1.0 |
| **Auteur**  Stephen AOGOLO  *Analyste-programmeur* |

Table des matières

1 - Versions 4

2 - Introduction 5

2.1 - Objet du document 5

2.2 - Références 6

3 - Prérequis 7

3.1 - Description du système 7

3.2 - Serveur de Base de données 8

3.3 - Serveur Web 9

3.4 - Web-services 10

3.4.1 - Serveur de Batches 11

3.4.2 - Serveur de Fichiers 11

3.5 - Bases de données 11

4 - Procédure de déploiement 12

4.1 - Descriptions des artefacts 14

4.2 - Configuration des scripts 15

4.2.1.1 - Script ubuntu-scripts.sh 15

4.2.1.2 - Script postgresql-scripts.sh 15

4.2.1.3 - Script venv-scripts.sh 15

4.2.1.4 - Script django-scripts.sh 16

4.2.1.5 - Script gunicorn-scripts.sh 16

4.2.1.6 - Script ngnix-scripts.sh 16

4.3 - Configuration des fichiers 17

4.3.1.1 - Fichier gunicorn.service 17

4.3.1.2 - Fichier ngnix\_OC\_PIZZA 18

4.4 - Exécution du déploiement 19

4.5 - Vérifications du déploiement 19

4.6 - Déploiement de l'Application Web 20

4.6.1 - Artefacts 20

4.6.2 - Environnement de l’application web 20

4.6.2.1 - Variables d’environnement 20

4.6.3 - Répertoire de configuration applicatif 20

4.6.3.1 - Fichier xxx.yyy 20

4.6.4 - DataSources 20

4.6.5 - Ressources 21

4.6.6 - Vérifications 21

5 - Procédure de démarrage / arrêt 22

5.1 - Désactivation du site web 22

5.2 - Activation du site web 22

5.3 - Rafraichissement du site web 23

5.3.1 - Réactivation du site web 23

5.3.2 - Rechargement du site web 23

5.3.3 - Les cas de maintenances manuelles 23

5.4 - Base de données 23

5.5 - Batches 23

5.6 - Application web 23

6 - Procédure de mise à jour 24

6.1 - Mise à jour catégorique 24

6.1.1 - Description des artefacts 25

6.1.2 - Configuration du script update-engine.sh 26

6.1.3 - Configuration du fichier controller.ini 26

6.1.4 - Exécution de la mise à jour catégorique 27

6.1.5 - Vérification de la mise à jour catégorique 27

6.2 - Mise à jour cyclique 28

6.2.1 - Description des artefacts 28

6.2.2 - Configuration de settings.py 29

6.2.3 - Réglage de la périodicité 29

6.2.4 - Vérification de la périodicité 30

6.3 - Base de données 30

6.4 - Batches 30

6.5 - Application web 30

7 - Supervision/Monitoring 31

7.1 - Supervision de l’application web 31

8 - Procédure de sauvegarde et restauration 32

9 - Glossaire 33

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Stephen AOGOLO | 14/01/2021 | Création du document | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Objet du document

Ce dossier d’exploitation (***version 1.0***) fait suite au dossier de spécifications technique (***version 2.0***).

L’objectif de ce dossier d’exploitation se partage en deux parties.

Dans un premier temps, ce dossier reprend et décrit techniquement le contenu vital du système. Par la suite il renseigne sur les procédures nominales d’installation, d’utilisation et de maintenance du système. Ce document présente et décrit donc le déploiement du produit **OC PIZZA – Boutique en ligne & Gestionnaire centralisé**.

Les points suivants seront donc traités afin de garantir le meilleur niveau d’information quant aux manipulations post-production :

* La description technique des sous-systèmes
* La procédure nominale du déploiement
* La procédure nominale de mise en route et d’extinction
* La procédure nominale de mise à jour
* Les actions et méthodes nominales de supervision
* Les actions et méthodes nominales de sauvegarde
* Les actions et méthodes nominales de restauration

*Il est important de noter que la mention «****nominale****» fait référence à la manière standard et recommandé d’effectuer ces opérations. Dans une version future du système et donc un futur dossier d’exploitation, les procédures, les actions et les méthodes «****d’urgences****» seront disponibles afin d’intervenir dans des cas particuliers de dépannage du système.*

## Références

Pour de plus amples informations, merci de se référer aux documents suivants :

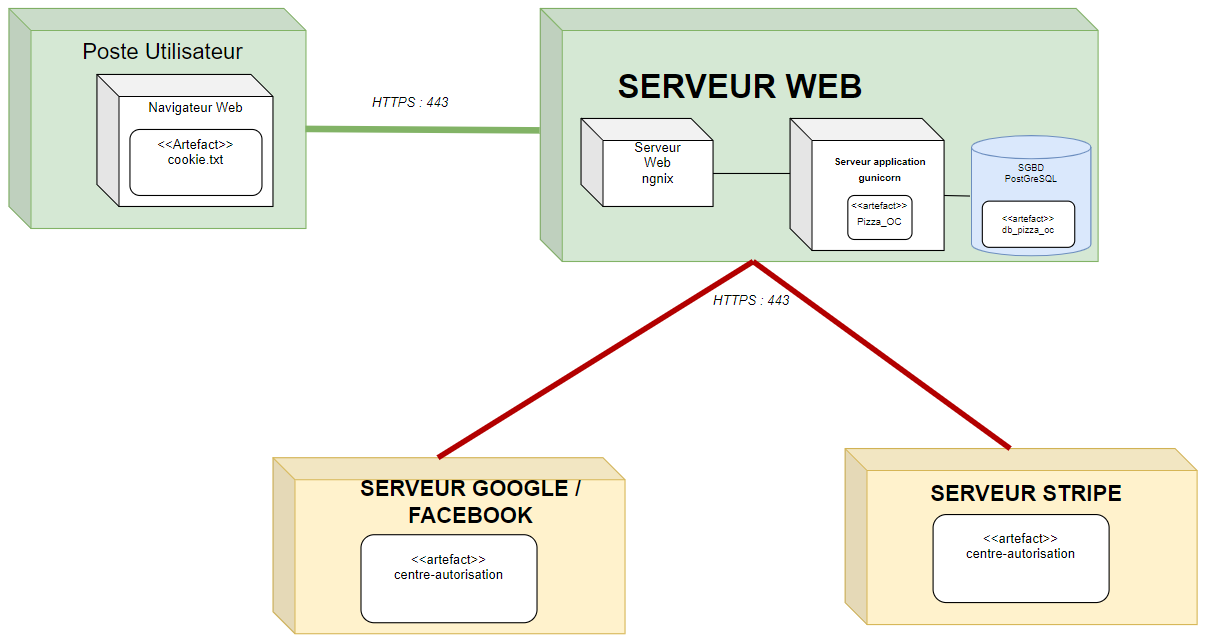
|  |  |
| --- | --- |
| REFERENCES | VERSIONS |
| Dossier de conception fonctionnelle | 1.4 |
| Dossier de conception technique | 2.0 |
| Procès-verbal de livraison | 1.0 |

# Prérequis

## Description du système

Cette partie reprend la description technique des composants du système.

L’ensemble des composants du système seront hébergés chez DIGITAL OCEAN. Ils occuperont un emplacement virtuel. Ci-dessous, le diagramme de déploiement, extrait du dossier de conception.



Ce diagramme reprend donc l’ensemble des composants principaux du système à déployer.

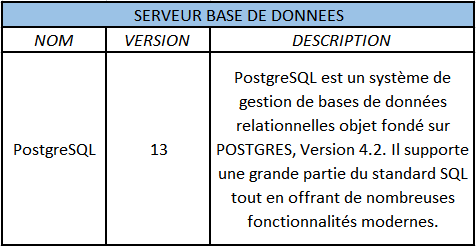
*PS : L’artefact « cookie.txt » sera installé automatiquement après la première connexion de l’utilisateur. Il n’est donc pas concerné par les procédures de déploiement, de démarrage et de maintenance.*

## Serveur de Base de données

Serveur de base de données hébergeant le/les schémas/base Xxxx….

Le serveur de base de données utilisé est de type **Postgresql**.

Le tableau ci-dessous reprend la description technique du serveur de base de données.

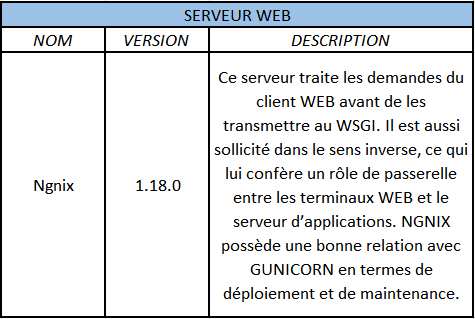


## Serveur Web

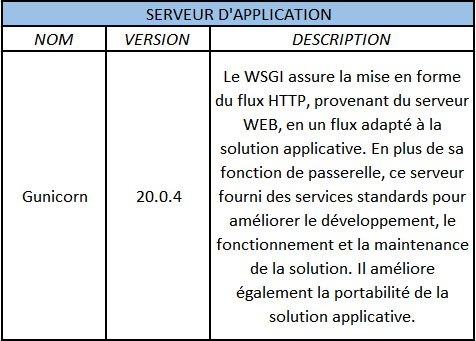
Serveur physique ou virtuel hébergeant l'application web.

Le serveur web est de type **NGNIX**. Il est le support du serveur d’application web de type **Gunicorn**.

Le tableau ci-dessous reprend la description technique du serveur web.



Le tableau ci-dessous reprend la description technique du serveur d’application web.



## Web-services

Les web services suivants doivent être accessibles et à jour :

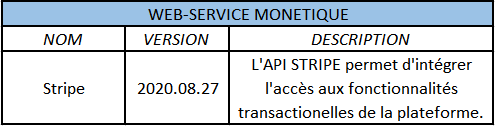
Les web-services sont des composants externes à la solution qui apportent des fonctionnalités supplémentaires au système. Elles sont implémentées via leur API respective.

*Pour de plus d’informations sur le rôle et la mise en place des web-services de la solution, consultez le* ***dossier de conception technique*** *à la partie* ***Règles de gestion Architecture Technique****.*

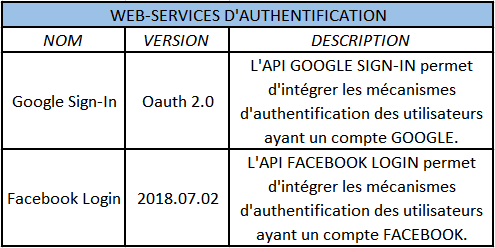
Le système utilise trois web-services classés en deux groupes :

* Un web-service monétique : Stripe
* Deux web-services d’authentification : Google Sign-in et Facebook Login

Le tableau ci-dessous reprend la description technique du web-service monétique.



Le tableau ci-dessous reprend la description technique des web-services d’authentification.



### Serveur de Batches

...

### Serveur de Fichiers

...

## Bases de données

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

* **… :** version XXX

# Procédure de déploiement

La procédure de déploiement suivante doit être respectée, afin de garantir le bon déploiement du site web **OC PIZZA**.

L’étape initiale

Avant de débuter le déploiement, il faut tout d’abord se rendre sur la machine d’hébergement, préalablement choisie. Dans le cas de ce dossier d’exploitation, une machine d’hébergement publique DIGITAL OCEAN. Il est donc essentiel de détenir l’adresse IP publique de la machine d’hébergement afin de s’y connecter et d’y effectuer l’ensemble des procédures à venir. Pour plus d’information sur cette étape initiale, merci de consulter [***https://www.digitalocean.com/docs/getting-started/***](https://www.digitalocean.com/docs/getting-started/).

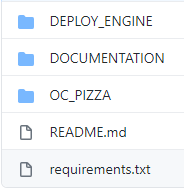
Pour se rendre sur la machine d’hébergement, il faut y accéder via une connexion SSH en ligne de commandes, comme suit :



Les acteurs du déploiement

A l’état initial, l’ensemble de la solution se trouve sur GITHUB à l’adresse suivante : [***https://github.com/StephenAOGOLO/P9\_Documentez\_votre\_systeme\_de\_gestion\_de\_pizzeria.git***](https://github.com/StephenAOGOLO/P9_Documentez_votre_systeme_de_gestion_de_pizzeria.git).

A cette adresse se trouve les éléments suivants :



Le répertoire « DOCUMENTATION » contient l’ensemble des dossiers liés à la conception du projet OC PIZZA.

Le répertoire « OC\_PIZZA » est le projet DJANGO de la solution, soit, le site web à déployer.

Le répertoire « DEPLOY\_ENGINE » contient les scripts de déploiement de la solution.

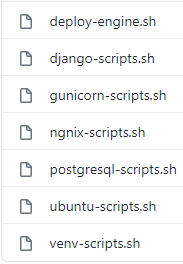
Le fichier « README.md » reprend la description générale du projet et conseille sur son mode utilisation.

Le fichier « requirements.txt » est un fichier utilisé par les scripts « DEPLOY\_ENGINE ».

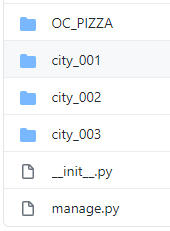
*SOLUTION*



DOCUMENTATION



DEPLOY\_ENGINE



OC\_PIZZA

Les étapes de déploiement

Le déploiement du site web est entièrement géré par les éléments du répertoire « DEPLOY\_ENGINE ».

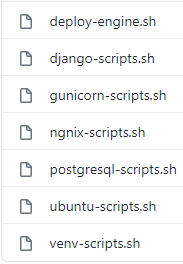
Le site web, de type DJANGO, sera installé sur une machine publique de type UBUNTU. Des composants LINUX-UBUNTU seront donc installés pour prendre en charge et servir les applications DJANGO.

La base de données POSTGRESQL sera configurée afin de gérer les données applicatives.

Le serveur d’application GUNICORN sera installé et configuré pour s’interfacer avec les applications du site.

Enfin, le serveur web NGNIX sera installé et configuré pour gérer la communication entre le service GUNICORN et internet, et ainsi, profiter des fonctionnalités de sécurité et de performance.

Ci-dessous, le contenu du répertoire « DEPLOY\_ENGINE ».

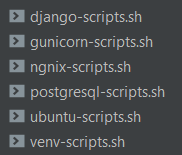


DEPLOY\_ENGINE

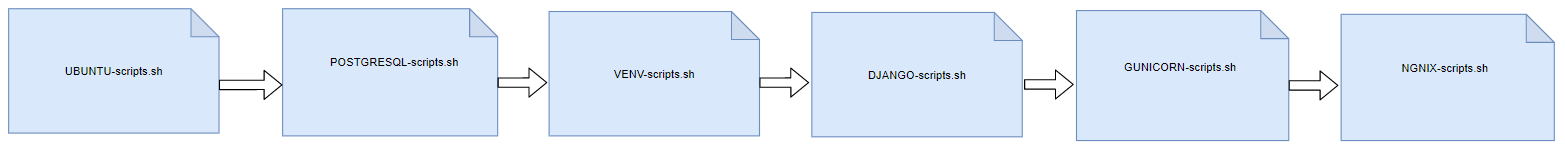
## Descriptions des artefacts

Le déploiement du site web **OC PIZZA** est semi-automatique. Il s’appuie sur l’exécution de plusieurs scripts partiellement rédigés nécessitant des précisions complémentaires.

Il y a six scripts à configurer avant qu’elles ne soient utilisées pour le déploiement semi-automatique du site :

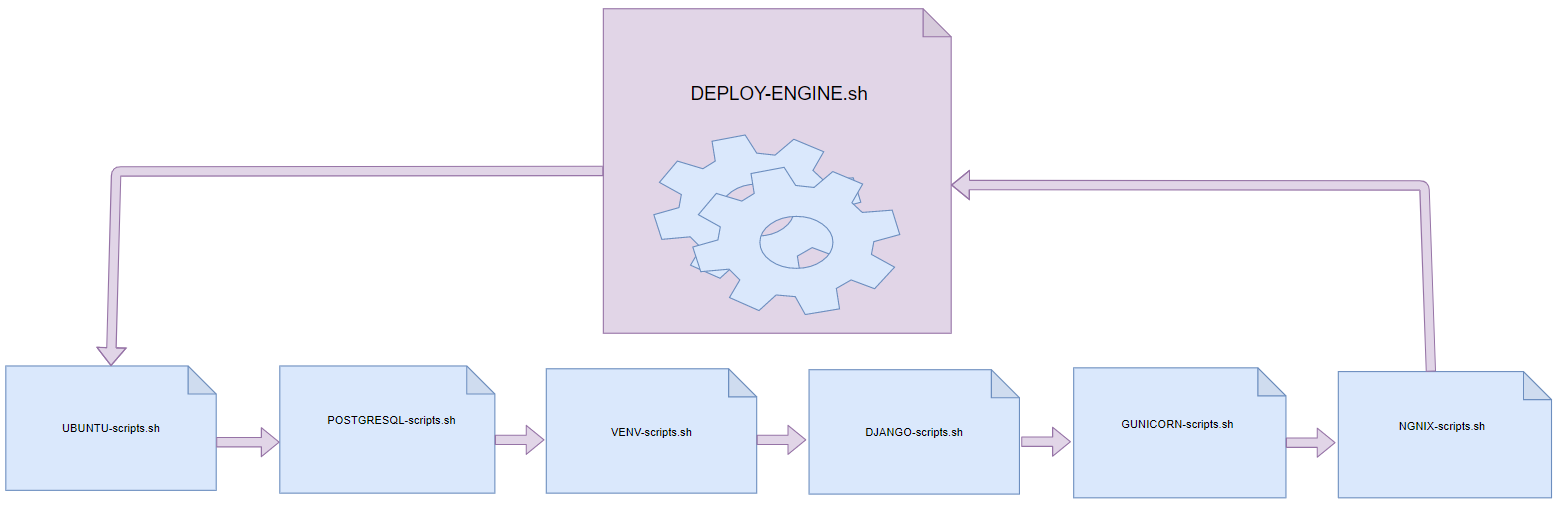


Ces artefacts seront exécutés dans l’ordre précis illustré ci-dessous :



Chaque artefact est une étape de déploiement. Chacun contient les opérations d’installation, de configuration et de lancement. Avant l’exécution de ces scripts, il est important de les édités afin de corriger et confirmer leur contenu.

L’exécution des six scripts cités précédemment, est gérée par un gestionnaire de déploiement « deploy-engine.sh ». Il assure le séquencement de l’installation générale des étapes de déploiement.



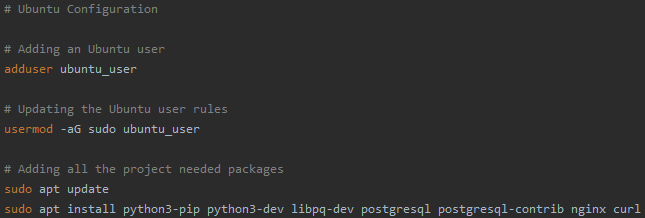
## Configuration des scripts

Sur votre machine locale, récupérez l’ensemble de la solution actuellement disponible sur GITHUB :



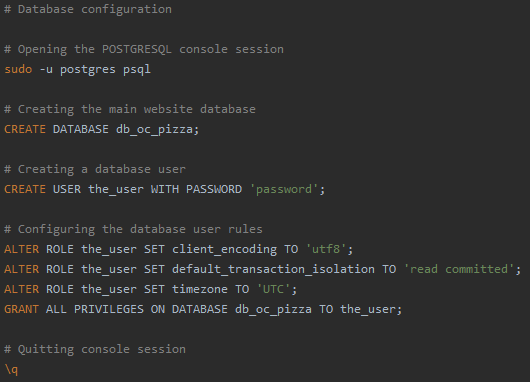
Editez les scripts suivants afin d’y ajouter vos informations personnalisées :

#### Script ubuntu-scripts.sh



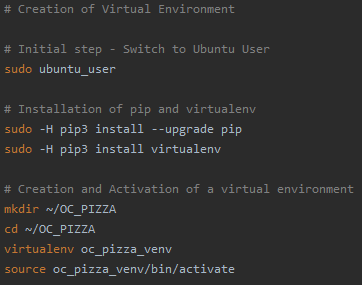
***A DECRIRE***

#### Script postgresql-scripts.sh



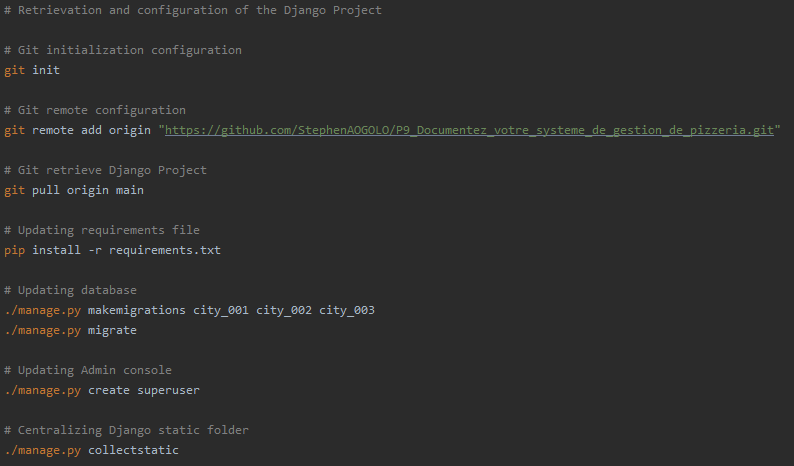
***A DECRIRE***

#### Script venv-scripts.sh



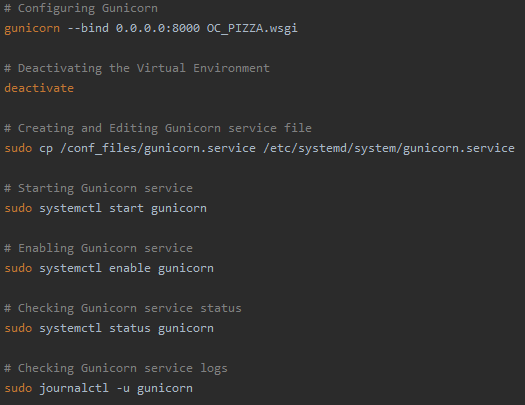
***A DECRIRE***

#### Script django-scripts.sh



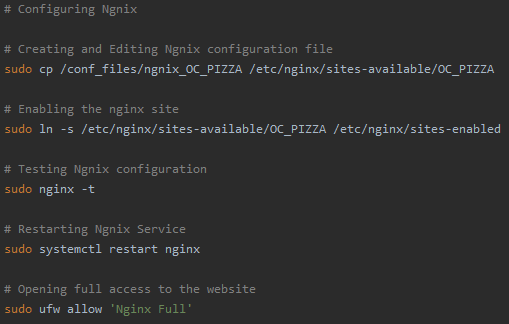
***A DECRIRE***

#### Script gunicorn-scripts.sh



***A DECRIRE***

#### Script ngnix-scripts.sh

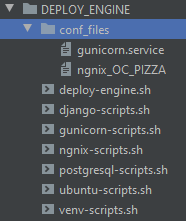


***A DECRIRE***

## Configuration des fichiers

Dans le répertoire « DEPLOY\_ENGINE », se trouve le second répertoire « conf\_files ».

Il contient les deux fichiers de configuration suivants :



gunicorn.service

* Fichier de configuration du service GUNICORN

ngnix\_OC\_PIZZA

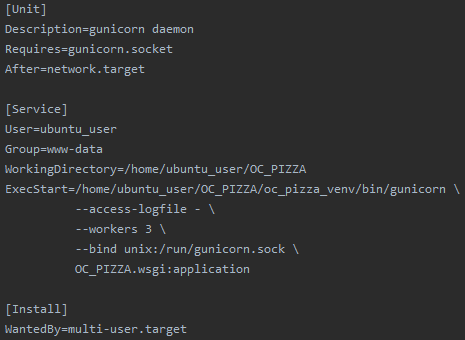
* Fichier de configuration du service NGNIX

Ces fichiers doivent également être modifiés avant l’exécution du gestionnaire de déploiement.

#### Fichier gunicorn.service

Ce fichier contient les informations suivantes :

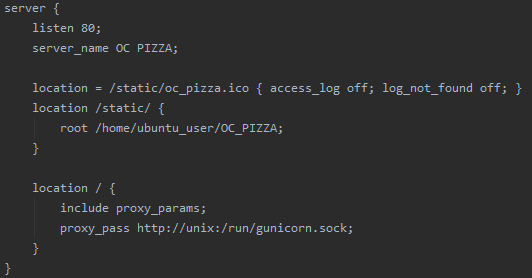
***A DECRIRE***



#### Fichier ngnix\_OC\_PIZZA

Ce fichier contient les informations suivantes :

***A DECRIRE***



## Exécution du déploiement

La configuration des scripts est à présent terminée. Le lancement du déploiement peut alors débuter.

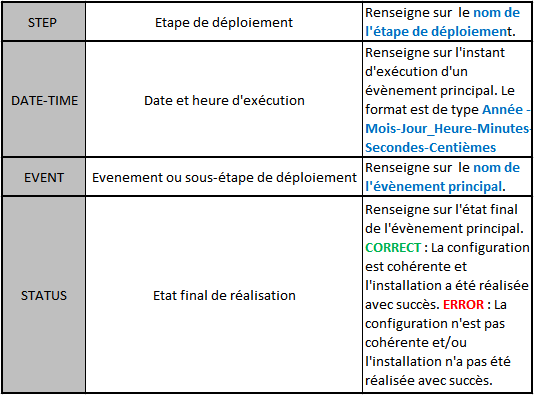
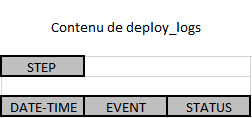
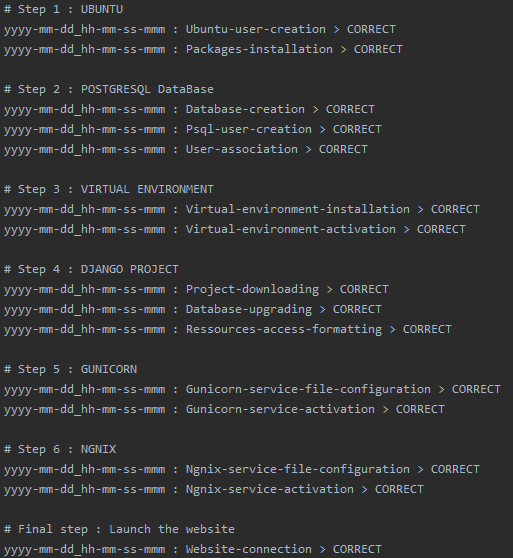
Lancez le gestionnaire de déploiement :



## Vérifications du déploiement

À la suite de l’exécution du gestionnaire de déploiement, le fichier « deploy\_logs » sera automatiquement généré. Ce fichier informe sur les sous-étapes de déploiement, autrement dit, les évènements principaux des six étapes de déploiement.

Ci-dessous, le fichier généré après le bon déploiement du site.



## Déploiement de l'Application Web

### Artefacts

...

### Environnement de l’application web

#### Variables d’environnement

Le serveur d'application JOnAS doit être exécuté avec la variable d'environnement suivante définie au démarrage. Elle est nécessaire afin de récupérer le répertoire contenant les fichiers de configuration de l'application :

**-Dcom.ocpizza.apps.conf=$home\_application\_conf\_directory**

INFO : il ne faut pas mettre de « / » à la fin de la valeur de la variable et ne pas utiliser d'espace dans le chemin.

### Répertoire de configuration applicatif

Le répertoire de configuration applicatif doit être créé sur le système de fichier et définit de la façon suivante :

**$home\_application\_conf\_directory/applicationX**

… fichiers de configuration… :

* ...

#### Fichier xxx.yyy

...

### DataSources

Les accès aux bases de données doivent se configurer à l'aide des fichiers…

Le fichier de drivers **postgresql (postgresql-9.2.x.)** doit être déposé dans le répertoire :

**$home\_server/lib/ext**

...

### Ressources

...

### Vérifications

Afin de vérifier le bon déploiement de l’application, faire ceci…

# Procédure de démarrage / arrêt

Le déploiement du site web a été réalisé avec succès, grâce au gestionnaire de déploiement.

Dans l’exemple de ce dossier d’exploitation, le site web est disponible à l’adresse suivante :

<https://www.ocpizza.fr>

Ainsi, les applications web et la base de données du site, sont opérationnelles et en cours d’exécution.

Certaines opérations de maintenance nécessitent une désactivation du site web. Afin de permettre ces maintenances particulières, des interventions manuelles sont requises pour arrêter et redémarrer le site web. Ces opérations se concentre sur le processus des services GUNICORN et NGNIX.

## Désactivation du site web

Pour désactiver correctement le site web, il faut exécuter la suite de commandes ci-dessous :



***A DECRIRE***











A la suite de ces opérations, le site web s’est désactivé correctement. Des interventions de maintenance peuvent être effectuées en toute sécurité.

## Activation du site web

Pour activer correctement le site web, il faut exécuter la suite de commandes ci-dessous :



***A DECRIRE***















A la suite de ces opérations, le site web s’est activé correctement. Son utilisation peut débuter en toute sécurité.

## Rafraichissement du site web

Dans certains cas, la réactivation ou le rechargement du site web suffisent.

### Réactivation du site web



***A DECRIRE***







### Rechargement du site web



***A DECRIRE***

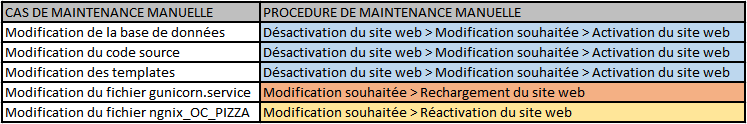




### Les cas de maintenances manuelles

Durant la vie d’hébergement du site web, des interventions de maintenance manuelles pourront être effectuées.

Toutes interventions manuelles nécessitent une procédure de maintenance spécifique. Dans le tableau ci-dessous, sont répertoriés les cas d’interventions manuels les plus fréquents. Ces derniers sont couplés aux procédures qui doivent être respectées pour assurer le bon fonctionnement du site.



## Base de données

## Batches

## Application web

# Procédure de mise à jour

Il existe deux moyens pour mettre à jour le site web :

La mise à jour catégorique

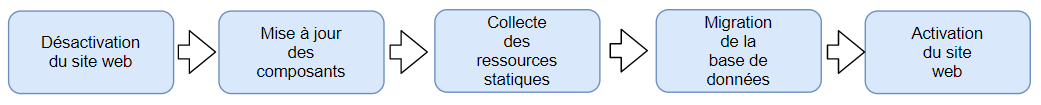
La mise à jour catégorique permet de mettre à jour la plupart des ressources du site web, tel que le code source, le contenu visuel ou encore la structure de la base de données. Cette est une opération non périodique, qui doit être préparée avant d’être exécutée. C’est donc un processus semi-automatique qui est déclenché manuellement.

La mise à jour cyclique

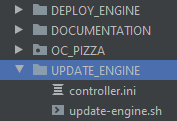
La mise à jour cyclique est un processus automatique qui démarre dès l’activation du site web. Il permet de mettre à jour périodiquement le contenu de la base de données, et non sa structure. Les données collectées durant ce processus proviennent de la base de données de production.

## Mise à jour catégorique

A l’instar de la procédure de déploiement, la procédure de mise à jour catégorique est semi-automatisée. Elle respecte des étapes de mise à jour bien précis :



Elle s’appuie sur l’exécution d’un script nommé « update-engine.sh », disponible dans le répertoire « UPDATE\_ENGINE ».

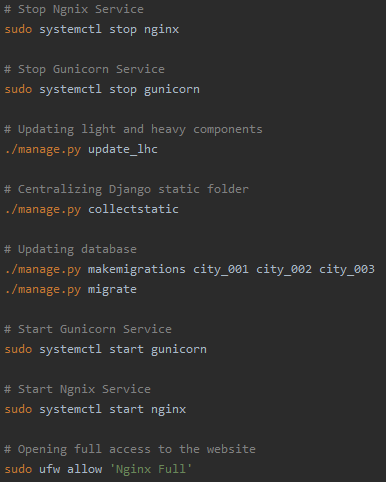


### Description des artefacts

Le répertoire « UPDATE\_ENGINE » contient donc deux artefacts :

* update-engine.sh
* controller.ini

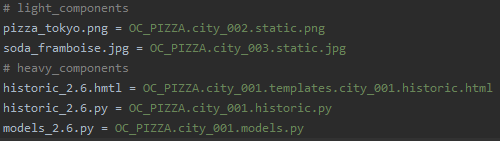
update-engine.sh



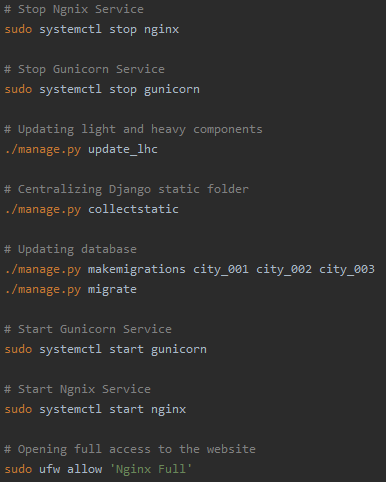
Ce script permet de séquencer les étapes de mise à jour.

Controller.ini

Ce fichier de configuration regroupe les informations d’adressage, relatives aux composants à mettre à jour.

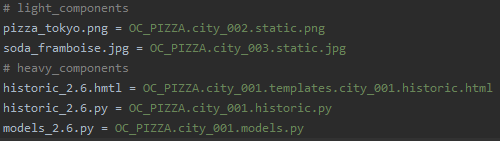


### Configuration du script update-engine.sh



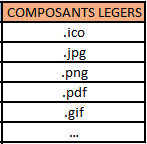
***A DECRIRE***

### Configuration du fichier controller.ini

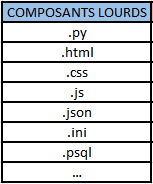


Les « light components » et les « heavy components » sont les composants ajoutés durant la mise à jour. Dans ce fichier, ces composants sont valorisés par leur adresse de destination.

Les « Light components » ou composants légers définissent tous les éléments comportant les extensions suivantes :



Les « Heavy components » ou composants lourds définissent tous les éléments comportant les extensions suivantes :



### Exécution de la mise à jour catégorique

Le projet DJANGO du site web possède une procédure de mise à jour personnalisée et développée par nos soins.

***A DECRIRE***

### Vérification de la mise à jour catégorique

A la fin de la mise à jour automatique. On peut vérifier la prise en compte des nouvelles modifications.

***A DECRIRE***

## Mise à jour cyclique

La mise à jour cyclique est un processus intégré au site web via un module DJANGO :

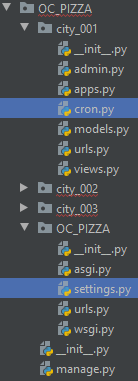
* DJANGO-CRONTAB

Conformément à sa documentation, ce module a été implémenté afin de s’affranchir de toutes interventions humaines, éventuellement utiles à la mise en marche du processus.

Toutefois, il est possible de régler la périodicité de cette opération.

### Description des artefacts

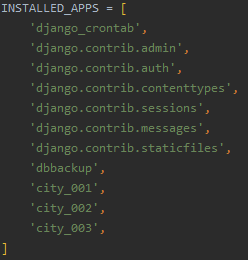
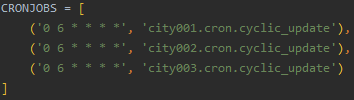
A titre d’information, l’exercice du module « DJANGO-CRONTAB » occupe deux types d’artefacts :



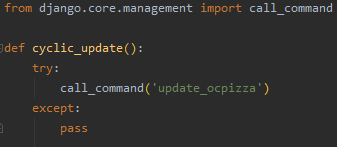
* **settings.py** – Module de paramétrage global du site web. Il contient notamment le réglage de périodicité.
* **cron.py** – Module dédié à l’action de mise à jour. Il est situé dans chaque application du projet DJANGO.

### Configuration de settings.py

Les images ci-dessous illustrent la couverture rédactionnelle du module « DAJNGO-CRONTAB ».



settings.py



cron.py

C’est au niveau du module « settings.py », dans la partie « CRONJOBS » que s’opère le réglage de la périodicité. Par défaut, la mise à jour a lieu tous les jours à 6h du matin.

### Réglage de la périodicité

Le réglage de la périodicité entre dans le cadre des cas de maintenances manuelles.

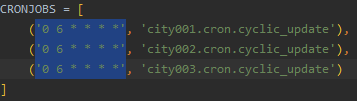
Avant de modifier le contenu du « CRONJOBS », assurez-vous de respecter la procédure de maintenance manuelle dédié aux modifications de code source.



Le « CRONJOBS » se compose essentiellement de deux parties. La première permet de préciser la périodicité des actions à mener. La seconde renseigne sur le type d’action à réaliser.

Dans notre cas, il existe plusieurs actions à réaliser car ce site web possèdent plusieurs applications, une action par application.

Afin de modifier uniquement la périodicité, veillez à apporter des modifications uniquement sur les premières parties du « CRONJOBS ».



« CRONJOBS » reprend les paramétrages et opérations standards du planificateur de tâches « CRON ».

Afin de parfaire le réglage de la périodicité, merci de consulter [la documentation CRON UBUNTU](https://doc.ubuntu-fr.org/cron).

### Vérification de la périodicité

A l’activation du site web, on peut vérifier la prise en compte des nouvelles modifications.

## Base de données

## Batches

## Application web

# Supervision/Monitoring

Le service de surveillance du site web OC PIZZA n’a pas été intégré dans la procédure de déploiement semi-automatisée. Il est donc recommandé d’installer un tel service après le déploiement du site.

Le service retenu pour la surveillance de l’application web est SUPERVISOR.

La procédure de mise en place de la surveillance est automatisée. Cependant, une intervention est nécessaire avant son exécution. Les éléments de surveillance se trouve dans le répertoire « SUPERVISON\_ENGINE ». Ce répertoire contient par défaut deux artefacts :

Son installation et son utilisation sont décrites dans les sous-parties suivantes.

## Description des artefacts

## Configuration du fichier supervisord.conf

Veuillez trouver ci-dessous la procédure d’installation du service SUPERVISOR :

## Utilisation de Supervisor

Afin de tester que l’application web est toujours fonctionnelles, faire ceci…

# Procédure de sauvegarde et restauration

# Glossaire

|  |  |
| --- | --- |
| **API** | Sigle anglais signifiant « Application Programming Interface », interface de programmation applicative en français. C’est une librairie de classes, fonctions et méthodes qui proposent un service WEB particulier. |
| **BATCH** | Terme anglais utilisé en informatique. C’est un proche synonyme de « script informatique ». C’est donc un fichier exécutable par une machine qui contient des instructions rédigées par un humain. Selon sa rédaction, il peut à son tour demander l’exécution d’un ou plusieurs autres scripts, on parle donc de « batch ou script en cascade ». |
| **DIGITAL OCEAN** | Fournisseur d’infrastructure cloud américain, hébergeur américain en d’autres termes dont le siège est basé à New York. La couverture en centre données occupe le monde entier. Cet hébergeur propose la plus part formule standard d’hébergement, du physique au virtuel. |
|  |  |
| **WSGI** | Sigle anglais signifiant « Web Server Gateway Interface ». C’est un type de serveur, voire une spécification qui définit une interface entre des serveurs et des applications web pour le langage Python. |
|  |  |