Installation sur CentOs

Pré-requis

- Installer Nginx (voir commmande plus bas)
- yum install git
- yum install gcc

Installation

A l'heure actuelle, Pod a été installé en test sur une VM spécifique, sous CentOS 7.

Pour réaliser cette installation, je me suis basé sur les packages et documentation de référence, à savoir :

- Sources : https://github.com/EsupPortail/podv2
- Documentation : https://podv2.readthedocs.io/ (??cette doc ne sera plus maintenue, se référer plutot au wiki)

Cependant, pour CentOS 7, il a fallu changer de nombreuses commandes. Ci-dessous les étapes d'installation réalisées :

> Lors de certaines étapes d'installation, il est nécessaire d'être **root**, ou - à minima - d'avoir les droits **sudo.**
>
> Selon l'environnement test / préproduction / production, il faut remplacer **%userpod%** par le bon user.
>
> Lors de l'installation, je parle du fichier **settings_local.py**; la documentation concernant ce fichier se retrouve plus loin, dans la partie Configuration Pod v2.

Installation de Pod

Création d'un nouvel utilisateur

Serveur(s) Pod / Compte root

```
[root@ts-sun-video ~]# adduser %userpod%
[root@ts-sun-video ~]# passwd %userpod%
[root@ts-sun-video ~]# usermod -aG wheel %userpod%
[root@ts-sun-video ~]# # L'utilisateur %userpod% doit faire parti du groupe nginx. A ce moment là, ce groupe
n'existe pas encore, mais il est très important de s'en rappeler par la suite !!!
[root@ts-sun-video ~]# usermod -g nginx %userpod%
```

Utilisation du repository epel

Serveur(s) Pod / Compte root

```
[root@ts-sun-video ~]# yum install epel-release
```

Installation python 3.6.x / pip

Installation de la dernière version de Python stable, en tant que root (ou via sudo).

Cf. https://www.rosehosting.com/blog/how-to-install-python-3-6-4-on-centos-7/

Serveur(s) Pod / Compte root

```
[root ~]# yum install -y https://centos7.iuscommunity.org/ius-release.rpm
[root ~]# yum update
[root ~]# yum install -y python36u python36u-libs python36u-devel python36u-pip
[root ~]# python3.6 -V
```

Requêtes annexes en lien avec pip

Cf. https://stackoverflow.com/questions/50408941/recommended-way-to-install-pip3-on-centos7

Serveur(s) Pod / Compte root

```
# Pour vérifier la version
[root ~]# python3.6 -m pip -V
# Pour mettre à jour pip3
[root ~]# python3.6 -m pip install --upgrade pip
# Pour vérifier la version
[root ~]# pip3 -V
 # Autant installer les autres packages, utiles pour la reprise de l'existant
[root ~]# pip3 install requests
[root ~]# pip3 install request
[root ~]# pip3 install wget
```

Par la suite, pour utiliser la nouvelle version de pip, il est nécessaire d'utiliser - en lieu et place de la commande pip :

- soit *python3.6 -m pip*
- soit *pip3.6*
- soit *pip3*

Mise en place de l'environnement virtuel

L'environnement virtuel sera positionné dans le répertoire **/data/www/%userpod%** (choix pour l'Université de Montpellier).

> *Surtout bien positionner l'environnement virtuel dans le bon répertoire dès le départ. Il est complexe par la suite de déplacer un environnement virtuel d'un répertoire à un autre !*

Serveur(s) Pod / Compte root

```
[root ~]# cd /data/www/%userpod%
[root /data/www/%userpod%]# pip3 install virtualenvwrapper
```

Puis éditer le fichier .bashrc du compte :

Serveur(s) Pod / Compte %userpod%

```
[%userpod% ~]$ cd
[%userpod% ~]$ emacs .bashrc
```

Si besoin, commenter les anciennes lignes, puis ajouter ces quelques lignes en fin de fichier :

Serveur(s) Pod / Emacs .bashrc (compte %userpod%)

```
# POD v2
export WORKON_HOME=/data/www/%userpod%/.virtualenvs

#### A priori, ces lignes ne sont pas utiles ####
# Il est necessaire d'utiliser le python du virtualenv
#export PYTHONPATH=$HOME/.virtualenvs/django_pod/bin/
#export PYTHONPATH=/usr/bin/
#PATH=$PYTHONPATH:$PATH

export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3.6
source /usr/bin/virtualenvwrapper.sh
```

> **Remarque** : le fichier virtualenvwrapper.sh est positionné dans */usr/bin/virtualenvwrapper.sh* et non pas */usr/local/bin/virtualenvwrapper.sh*.

Prendre en compte les modifications :

Serveur(s) Pod / Compte %userpod%

```
[%userpod% ~]$ source .bashrc
```

Et créer un nouvel environnement virtuel :

Serveur(s) Pod / Compte %userpod%

```
[%userpod% ~]$ cd /data/www/%userpod%
[%userpod% /data/www/%userpod%]$ mkvirtualenv --system-site-packages --python=/usr/bin/python3.6 django_pod
```

Récupération des sources

*Pour l'Université de Montpellier, on positionne le projet dans **/data/www/%userpod%** et non /usr/local.*

Serveur(s) Pod / Compte root ou %userpod%

```
En tant que %userpod%:
[%userpod% ~]$ sudo mkdir /data/www/%userpod%/django_projects
OU
En tant que root (A vérifier: existence du groupe nginx)
[root ~]$ mkdir /data/www/%userpod%/django_projects
[root ~]$ chown %userpod%:nginx /data/www/%userpod%/django_projects/
```

Création d'un lien symbolique, dans le home de l'utilisateur :

Serveur(s) Pod / Compte %userpod%

```
[%userpod% ~]$ cd
[%userpod% ~]$ ln -s /data/www/%userpod%/django_projects django_projects
```

?? S'il y a besoin de supprimer un ancien lien symbolique existant, penser à faire - sous root - : unlink django_projects

Récupérer les sources :

Serveur(s) Pod / Compte %userpod%

```
[%userpod% ~]$ cd /data/www/%userpod%/django_projects
[%userpod% /data/www/%userpod%/django_projects]$ git clone https://github.com/esupportail/podv2.git
```

Applications tierces

Installation de toutes les librairies python

On profite pour mettre à jour la version de pip3 avec l'installation des librairies Python :

Serveur(s) Pod / Compte %userpod%

```
[%userpod% ~]$ cd
[%userpod% ~]$ source .bashrc
[%userpod% ~]$ cd /data/www/%userpod%/django_projects/podv2
[%userpod% /data/www/%userpod%/django_projects/podv2]$ workon django_pod
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2] pip3 install --upgrade pip
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2] pip3 install -r requirements.txt
```

? Il faut vérifier que l'on se trouve bien dans l'environnement virtuel (présence de "(django_pod)" au début l'invite de commande. Sinon, il faut lancer la commande, dans le bon répertoire `$> workon django_pod.`

ImageMagick

Utilisé pour la génération des overviews. A installer sur le serveur d'encodage.

```
# Pour CentOS 8, ajoutez ce yum config-manager :
[root ~]# yum config-manager --set-enabled PowerTools


[root ~]# yum install ImageMagick ImageMagick-devel
```

FFMPEG

Utilisé pour l'encodage des vidéos. À installer sur le serveur d'encodage.

```
[root /opt]# mkdir /opt/ffmpeg -p
[root /opt]# cd /opt/ffmpeg
[root /opt]# wget https://johnvansickle.com/ffmpeg/releases/ffmpeg-release-amd64-static.tar.xz
[root /opt]# tar -Jxvf ffmpeg-release-amd64-static.tar.xz
[root /opt]# # ATTENTION : changer les X.X.X par la bonne version !!!
[root /opt]# ln -s ffmpeg-X.X.X-amd64-static ffmpeg
[root /opt]# chown -R %userpod%:nginx /opt/ffmpeg
```

Il faudra modifier le fichier *settings_local.py* pour indiquer les chemins ffmpeg et ffprobe :

```
FFMPEG = '/opt/ffmpeg/ffmpeg/ffmpeg'
FFPROBE = '/opt/ffmpeg/ffmpeg/ffprobe'
```

FFMPEGTHUMBNAILER

Utilisé pour l'encodage des vignettes (plus rapide que ffmpeg dans ce cas là). A installer sur le serveur d'encodage.

Installation décrite à cette adresse : http://linoxide.com/linux-how-to/install-ffmpeg-centos-7/

```
[root ~]# yum -y install epel-release
[root ~]# rpm --import http://li.nux.ro/download/nux/RPM-GPG-KEY-nux.ro
[root ~]# # Attention : modification du 0-1 en 0-5 par rapport à la documentation ci-dessus
[root ~]# rpm -Uvh http://li.nux.ro/download/nux/dextop/el7/x86_64/nux-dextop-release-0-5.el7.nux.noarch.rpm
[root ~]# yum repolist

# Pour CentOS 8, ajouter :
[root ~]# yum localinstall --nogpgcheck https://download1.rpmfusion.org/free/el/rpmfusion-free-release-8.noarch.
rpm -y

[root ~]# yum install --enablerepo=epel ffmpeg ffmpeg-devel
[root ~]# yum install --enablerepo=epel ffmpegthumbnailer
```

Au final, le serveur contient 2 versions différentes de ffmpeg :

- une version installée à la main (cf. paragraphe précédent), qui sera la plus récente possible. C'est cette version qui sera utilisée par *Pod* .
- une version installée via yum. Cette version sera seulement utilisée via *ffmpegthumbnailer*.

Elasticsearch

Pour utiliser Elasticsearch, il faut avoir java8 sur sa machine.

Installation de java

Dans l'environnement applicatif UM, Java est installé de base; cette étape n'est alors pas nécessaire.

```
[root ~]# yum install java-1.8.0-openjdk.x86_64
```

Installation Elasticsearch

1. PGP

Repository pour Elasticsearch (compte root)

```
[root ~]# rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

2. Création du fichier ***/etc/yum.repos.d/elasticsearch.repo***

emacs /etc/yum.repos.d/elasticsearch.repo (compte root)

```
[elasticsearch-6.x]
name=Elasticsearch repository for 6.x packages
baseurl=https://artifacts.elastic.co/packages/6.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

3. Installation via yum

E / Installation Elasticsearch (compte root)

```
[root ~]# yum install elasticsearch
```

? *S'il y a besoin de supprimer une ancienne installation d'Elasticsearch, exécuter la commande suivante, en tant que root : yum erase elasticsearch*

4. Mise en service

Mise en service Elasticsearch (compte root)

```
[root ~]# systemctl daemon-reload
[root ~]# systemctl enable elasticsearch.service
```

6. Gestion des droits des répertoires

E / Droits des répertoires Elasticsearch (compte root)

```
[root ~]# # Création des répertoires de base
[root ~]# mkdir /data/elasticsearch/data -p
[root ~]# mkdir /data/elasticsearch/logs
[root ~]# chown elasticsearch:elasticsearch /data/elasticsearch/ -R
[root ~]# chown elasticsearch:elasticsearch /usr/share/elasticsearch/ -R
[root ~]# chown elasticsearch:elasticsearch /etc/elasticsearch/ -R
```

6. Paramétrage

emacs /etc/elasticsearch/elasticsearch.yml (compte root)

```
cluster.name: %userpod%
node.name: node-1
path.data: /data/elasticsearch/data
path.logs: /data/elasticsearch/logs
# Mettre l'adresse IP du serveur
network.host: 162.38.xx.xx
# Au cas où, il est aussi possible de mettre 0.0.0.0
# network.host: 0.0.0.0
# Adresse IP du ou des serveurs pod
discovery.zen.ping.unicast.hosts: ["162.38.xx.xx"]
```

7. Démarrage et test ES

Serveur(s) Pod / Démarrage et test Elasticsearch (compte root)

```
[root ~]# systemctl start elasticsearch.service
# Mettre l'adresse IP du serveur Elasticsearch
[root ~]# curl -XGET http://162.38.xx.xx:9200/_cat/nodes
```

8. Installation du plugin ICU

Pour utiliser la recherche dans Pod, nous allons avoir besoin également du plugin ICU:

Serveur(s) Pod / Installation plugin ICU (compte root)

```
[root ~]# cd /usr/share/elasticsearch/ ????
[root ~]# bin/elasticsearch-plugin install analysis-icu
[root ~]# systemctl restart elasticsearch
```

Installation et configuration du module H5PP

H5P n'est plus supporté dans Pod. à titre d'archive, vous pouvez toujours consulter l'ancienne doc d'install h5p.

Encodage déporté

Installation de RabbitMQ

À installer uniquement sur le serveur principal (1 seul RabbitMQ est utile).

Installation d'Erlang

Si besoin : pour supprimer les anciennes versions d'Erlang : (yum list installed | grep erl | grep -v perl)

```
yum remove 'erlang-*'
```

La documentation officielle : https://www.erlang-solutions.com/resources/download.html

Voici ce qui a été fait à l'UM :

Serveur principal Pod / Compte root

```
[root ~]# cd
[root ~]# wget https://github.com/rabbitmq/erlang-rpm/releases/download/v21.2.3/erlang-21.2.3-1.el7.centos.
x86_64.rpm
[root ~]# yum install erlang-21.2.3-1.el7.centos.x86_64.rpm
```

Installation de RabbitMQ

Serveur principal Pod / Compte root

```
[root ~]# cd
[root ~]# wget https://github.com/rabbitmq/rabbitmq-server/releases/download/v3.7.10/rabbitmq-server-3.7.10-1.
el7.noarch.rpm
[root ~]# rpm --import https://www.rabbitmq.com/rabbitmq-release-signing-key.asc
[root ~]# yum install rabbitmq-server-3.7.10-1.el7.noarch.rpm
[root ~]# # Démarrage du service la 1° fois
[root ~]# systemctl start rabbitmq-server
[root ~]# # Activation des plugins RabbitMQ (utile pour l'interface Web)
[root ~]# rabbitmq-plugins enable rabbitmq_management
```

Utilisation de Rabbitmq management console

Il est possible d'utiliser une interface Web pour gérer Rabbitmq ; cette interface se nomme Rabbitmq management console.

La version installée contient - de base - cette interface. Il n'y a plus besoin d'installation complémentaire.

Il est ensuite possible d'accéder à cette interface via le port 15672 : http://[serverPod]:15672/

? *Penser à rendre accessible ce port pour les utilisateurs concernés de la DSI.*

Gestion de la sécurité

Serveur principal Pod / Compte root

```
[root ~]# # On crée un nouvel utilisateur pod qui dispose d'un mot de passe xxxxx
[root ~]# rabbitmqctl add_user pod xxxxx
[root ~]# # On donne les droits administrateurs. Commande documentée mais ne fonctionne pas dans mon cas
"rabbitmqctl set_admin pod" remplacée par celle ci-dessous
[root ~]# rabbitmqctl set_user_tags pod administrator
[root ~]# # Commande documentée mais ne fonctionne pas dans mon cas "rabbitmqctl clear_admin guest" remplacée
par celle ci-dessous
[root ~]# rabbitmqctl set_permissions -p / pod ".*" ".*" ".*"
```

Installation de Celery

Configuration Pod & Celery

Celery est intégré à Pod dans le fichier requirements.txt à la racine du projet. Pensez à vérifier que vous avez bien installé les prérequis python au passage (n'oubliez pas d'être en environnement virtuel) :

```
(django_pod) %userPod%:~/django_projects/pod$ pip3 install -r requirements.txt
```

Les fichiers relatifs à Celery sont déjà créés dans Pod. L'installation est donc terminée.

Configurer le fichier **settings_local.py** de Pod, pour utiliser l'encodage déporté.

```
# Encode with Celery
CELERY_TO_ENCODE = True
CELERY_BROKER_URL = "amqp://pod:xxxx@162.38.xxx//"
```

⚠️ *La valeur du celery_broker doit pointer vers le serveur RabbitMQ (installé sur le serveur principal).*

⚠️ *Cette configuration a changé entre Pod v1 et Pod v2. Il faut maintenant utiliser CELERY_BROKER_URL.*

Le paragraphe suivant concerne le serveur d'encodage. Dans son cas, il doit recevoir les ordres d'encodage qui lui ont été envoyés.
La création des fichiers celeryd permettent au serveur d'encodage de recevoir ces ordres !

Installations complémentaires (Celeryd)

À réaliser UNIQUEMENT sur le serveur d'encodage.

- Créer le fichier du service **/etc/init.d/celeryd,** identique à ce fichier https://raw.githubusercontent.com/celery/celery/4.3/extra/generic-init.d/celeryd
- *chmod 755 /etc/init.d/celeryd*
- Créer le fichier **/etc/default/celeryd**

```
# Nom du/des worker(s). Ajoutez autant de workers que de tache à executer en paralelle.
# exemple : CELERYD_NODES="worker1 worker2 worker3 worker4"
CELERYD_NODES="worker1"
# Settings de votre Pod
DJANGO_SETTINGS_MODULE="pod.settings"
# Répertoire source de celery
CELERY_BIN="/data/www/%userpod%/.virtualenvs/django_pod/bin/celery"
# Application où se situe celery
CELERY_APP="pod.main"
# Répertoire du projet Pod (où se trouve manage.py)
CELERYD_CHDIR="/data/www/%userpod%/django_projects/podv2"
# Options à appliquer en plus sur le comportement du/des worker(s)
CELERYD_OPTS="--time-limit=86400 --concurrency=1 --maxtasksperchild=1"
# Fichier log
CELERYD_LOG_FILE="/var/log/celery/%N.log"
# Fichier pid du socket
CELERYD_PID_FILE="/var/run/celery/%N.pid"
# Utilisateur système utilisant celery
CELERYD_USER="%userpod%"
# Groupe système utilisant celery
CELERYD_GROUP="nginx"
# Si celery dispose du droit de création de dossiers
CELERY_CREATE_DIRS=1
# Niveau d'information qui seront inscrit dans les logs
CELERYD_LOG_LEVEL="INFO"
```

- Démarrage si nécessaire du service : *systemctl start celeryd*

Ayant eu des problèmes de droits sur le répertoire contenant les vidéos (qui peuvent être déposés et par le user nginx et par le user Celery), j'ai ajouté l'utilisateur dans le groupe Nginx et j'ai changé quelques droits :

```
[root ~]# mkdir /var/log/celery
[root ~]# chown poduser:nginx /var/log/celery

[root ~]# mkdir /var/run/celery
[root ~]# chown poduser:nginx /var/run/celery



[root ~]# usermod -g nginx %userpod%
[root ~]# chown %userpod%:nginx /data/www/%userpod%/media -R
[root ~]# chmod 755 /data/www/%userpod%/media/ -R
```

- Configurer pour que ce service soit démarré lors d'un reboot :

Mise en service uwsgi-pod (compte root)

```
[root ~]# systemctl daemon-reload
[root ~]# systemctl enable celeryd
```

- Configurer un système de logrotate pour vider régulièrement les logs de Celery, sur le serveur d'encodage. Pour cela, j'ai créé le fichier **/etc /logrotate.d/celery** avec le contenu suivant :

```
/var/log/celery/*.log {
    missingok
    notifempty
    compress
    delaycompress
    copytruncate
    daily
    dateext
    rotate 7
    size 10M
}
```

Configuration Pod v2

Le fichier custom/settings_local.py

La configuration de Pod v2 se réalise au travers d'un fichier de configuration, spécifique à chaque établissement, *settings_local.py*.

Il est alors nécessaire de créer ce fichier via :

Serveur(s) Pod / Création du fichier settings_local.py (compte %userpod%)

```
[%userpod%@ ~]# mkdir /data/www/%userpod%/django_projects/podv2/pod/custom/
[%userpod%@ ~]# cp /data/www/%userpod%/django_projects/podv2/pod/main/settings.py /data/www/%userpod%
/django_projects/podv2/pod/custom/settings_local.py
```

Serveur(s) Pod / Droits du fichier (compte root)

```
[root@ ~]# # En cas de problème de droit
[root@ ~]# chown %userpod%:nginx /data/www/%userpod%/django_projects/podv2/pod/custom/ -R
```

Une fois ce fichier créé, il est nécessaire de réaliser la configuration.

Pour cela, il y a la documentation officielle, accessible à l'adresse suivante : https://podv2.readthedocs.io/configuration/

Le mieux est de partir du fichier de configuration créé dans l'environnement de test, qui configure l'ensemble des éléments, à savoir :

- Base de données,
- Annuaire LDAP,
- Authentification CAS,
- Gestion des affiliations,
- Encodage déporté,
- Langues,
- Emails,
- Templates spécifiques
- ...

Il est indispensable que le fichier de configuration *settings_local.py* soit finalisé avant la création de la base de données. De nombreux paramètres de configuration ont un impact sur la structure des tables de la base de données.

En particulier, il est indispensable de vérifier les paramètres suivants (il ne faut pas changer les valeurs de ces paramètres par la suite) :

- *USE_PODFILE = True*
  Activation du gestionnaire de fichiers.
- *FROM_URL = "https://video.umontpellier.fr/media/"*
  Utile pour la récupération des vidéos depuis ce serveur (serveur de production de podv1).
- *MEDIA_ROOT = '/data/www/%userpod%/media'*
  Utile pour la récupération des vidéos.
- *LANGUAGES :...*
  Ne garder que les valeurs FR et EN : supprimer la valeur NL avant de créer la base de données.

Dans le cas contraire, il sera sûrement nécessaire de réaliser une migration de la base de données via les commandes suivantes :

- *python manage.py makemigrations*
- *python manage.py migrate*

N'oubliez pas de relancer tous les services en lien avec ce fichier de configuration (en cas de cache) : uWSGI, celeryd voire nginx et rabbitmq (cf. document d'exploitation v2).

Je le répète : en cas de changement de configuration, il est préférable de supprimer tout le contenu de la base et de la recréer.

Au final, voici le contenu du fichier custom/settings_local.py :

Serveur(s) Pod / Fichier custom/settings_local.py

```
# -*- coding: utf-8 -*-
from django.utils.translation import ugettext_lazy as _

##
# DEBUG mode activation
#
# https://docs.djangoproject.com/en/1.11/ref/settings/#debug
#
# SECURITY WARNING: MUST be set to False when deploying into production.
DEBUG = False

SECRET_KEY = 'xxxxxxxxxxxxxxxxxxxxxxx'

# Droit des fichiers uploadés
FILE_UPLOAD_PERMISSIONS = 0o644


# Permet d'utiliser la mise en ligne fragmentée (qui permet de reprendre la mise en ligne lors de problèmes de
connexion)
USE_CHUNKED_UPLOAD = True

##
# A list of strings representing the host/domain names
# that this Django site is allowed to serve.
#
# https://docs.djangoproject.com/en/1.11/ref/settings/#allowed-hosts
ALLOWED_HOSTS = ['162.38.xx.xx', 'xxxxx.umontpellier.fr']

##
# A tuple that lists people who get code error notifications
#   when DEBUG=False and a view raises an exception.
#
# https://docs.djangoproject.com/fr/1.11/ref/settings/#std:setting-ADMINS
#
ADMINS = (
    ('Admin Pod', 'xxx.xxx@umontpellier.fr'),
)
##
# A dictionary containing the settings for all databases
# to be used with Django.
#
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases
```

```
"""
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': '/data/www/podtest/django_projects/podv2/db.sqlite',
    }
}
"""

# MySQL settings
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'xxx',
        'USER': 'xxx',
        'PASSWORD': 'xxx',
        'HOST': 'xxxxx.umontpellier.fr',
        'PORT': '',
        'OPTIONS': {'init_command':"SET storage_engine=INNODB, sql_mode='STRICT_TRANS_TABLES',
innodb_strict_mode=1;"}
        #'OPTIONS': {'init_command':"SET storage_engine=INNODB;"}
    }
}

RESTRICT_EDIT_VIDEO_ACCESS_TO_STAFF_ONLY = True

VIDEO_MAX_UPLOAD_SIZE = 4

FILE_ALLOWED_EXTENSIONS = ( 'doc', 'docx', 'odt', 'pdf', 'xls', 'xlsx', 'ods', 'ppt', 'pptx', 'txt', 'html',
'htm', 'vtt', 'srt', 'webm', 'ts', )
IMAGE_ALLOWED_EXTENSIONS = ( 'jpg', 'jpeg', 'bmp', 'png', 'gif', 'tiff', )
FILE_MAX_UPLOAD_SIZE = 20

##
# THIRD PARTY APPS OPTIONNAL
#
USE_PODFILE = True
THIRD_PARTY_APPS = ['live', 'enrichment']

# Nouveaute v2
AUTH_TYPE = (('local', ('local')), ('CAS', 'CAS'))

USE_CAS = True
CAS_VERSION = '3'
CAS_SERVER_URL = 'https://xxx.umontpellier.fr/cas/'
CAS_GATEWAY = False
POPULATE_USER = 'LDAP'
AUTH_CAS_USER_SEARCH = 'user'

CREATE_GROUP_FROM_AFFILIATION = False
AFFILIATION_STAFF = ('faculty', 'employee', 'researcher', 'affiliate')

LDAP_SERVER = {'url': 'xxxxx.umontpellier.fr', 'port': 389, 'use_ssl': False}
AUTH_LDAP_BIND_DN = 'xxx'
AUTH_LDAP_BIND_PASSWORD =  'xxx'
AUTH_LDAP_BASE_DN = 'ou=people,dc=umontpellier,dc=fr'
AUTH_LDAP_USER_SEARCH = (AUTH_LDAP_BASE_DN, "(uid=%(uid)s)")

##
# Internationalization and localization.
#
#   https://docs.djangoproject.com/en/1.8/ref/settings/#globalization-i18n-l10n
#
LANGUAGE_CODE = 'fr'
LANGUAGES = (
    ('fr', 'Fran?§ais'),
    ('en', 'English')
)
MODELTRANSLATION_DEFAULT_LANGUAGE = 'fr'
MODELTRANSLATION_FALLBACK_LANGUAGES = ('fr', 'en')
```

```
##
# A string representing the time zone for this installation.
#
#   https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
#
TIME_ZONE = 'Europe/Paris'

##
# Dynamic files (user managed content: videos, subtitles, documents, etc...)
#
# https://docs.djangoproject.com/en/1.11/ref/settings/#media-url
# https://docs.djangoproject.com/en/1.11/ref/settings/#media-root
#
# WARNING: this folder must have previously been created.
MEDIA_URL = '/media/'
MEDIA_ROOT = '/data/www/%userpod%/media'

FILE_UPLOAD_TEMP_DIR = os.path.join(os.path.sep, 'var', 'tmp')

##
# eMail settings
#
# https://docs.djangoproject.com/en/1.11/ref/settings/#email-host
# https://docs.djangoproject.com/en/1.11/ref/settings/#email-port
# https://docs.djangoproject.com/en/1.11/ref/settings/#default-from-email
#
#   username: EMAIL_HOST_USER
#   password: EMAIL_HOST_PASSWORD
#
EMAIL_HOST = 'smtp-xxx.umontpellier.fr'
EMAIL_PORT = 25
DEFAULT_FROM_EMAIL = 'xxx.xxx@umontpellier.fr'

# https://docs.djangoproject.com/fr/1.11/ref/settings/#std:setting-SERVER_EMAIL
SERVER_EMAIL = 'xxx.xxx@umontpellier.fr'

MENUBAR_SHOW_STAFF_OWNERS_ONLY = True
HOMEPAGE_SHOWS_RESTRICTED = True
##
# List of Elasticsearch urls, like ['host1', 'host2', ...].
#
#   http://elasticutils.readthedocs.io/en/latest/django.html#django.conf.settings.ES_URLS
#
ES_URL = ['http://162.38.xx.xx:9200/']

FFMPEG = '/opt/ffmpeg/ffmpeg/ffmpeg'
FFPROBE = '/opt/ffmpeg/ffmpeg/ffprobe'

# Encode with Celery
CELERY_TO_ENCODE = True
CELERY_NAME = "pod_project"
CELERY_BACKEND = "amqp"
CELERY_BROKER = "amqp://pod:xxxx@162.38.xx.xx//"

# Template Settings
TEMPLATE_VISIBLE_SETTINGS = {
     'TITLE_SITE': 'Vidéo - Université de Montpellier',
     'TITLE_ETB': 'Université de Montpellier',
     'LOGO_SITE': 'custom/um_pod.png',
     'LOGO_COMPACT_SITE': 'custom/logo_black_compact.png',
     'LOGO_ETB': 'custom/um.png',
     'LOGO_PLAYER': 'custom/logo_player.png',
     'FOOTER_TEXT': (
         '163 rue Auguste Broussonnet',
         '34090 Montpellier',
         ('<a href="https://goo.gl/maps/RHsfxME59Fp"'
          ' target="_blank">Google maps</a>')
     ),
     'LINK_PLAYER': 'https://www.umontpellier.fr',
     'CSS_OVERRIDE': 'custom/um.css',
     'FAVICON': 'custom/favicon.png'
```

```
}

SUBJECT_CHOICES = ( ('', '-----'), ('info', _('Request more information')), ('request_password', _('Password
request for a video')), ('inappropriate_content', _('Report inappropriate content')),
('bug', _('Correction or bug report')), ('other', _('Other (please specify)')) )

# Templates spécifiques UM
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['/data/www/%userpod%/django_projects/podv2/pod/custom/templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                # Local contexts
                'pod.main.context_processors.context_settings',
                'pod.main.context_processors.context_navbar'
            ],
        },
    },
]

DEFAULT_THUMBNAIL = 'custom/default.png'

# Utile pour la reprise de l'existant (adresse de Pod v1 en production)
FROM_URL = 'https://video.umontpellier.fr/media/'
```

Droits des fichiers uploadés

Pour que les fichiers vidéos uploadés aient les bons droits (*pour simplifier 644*), j'ai positionné la ligne suivante dans le fichier settings_local.py :

FILE_UPLOAD_PERMISSIONS = 0o644

Base de données

Installation de la librairie

Serveur(s) Pod / Compte %userpod%

```
[%userpod%@ts-sun-video ~]$ pip3 install "mysqlclient==1.3.14"
```

⚠ En cas d'erreur, c'est que le client MySQL / MariaDB n'a pas été installé; il est alors nécessaire de réaliser, sous root, un *yum install mariadb-devel*

Surtout ne pas faire de "*pip3 install mysqlclient*" : il semblerait que la version de Django utilisée ne soit pas compatible avec la dernière version de MySQL.

Cela provoquait une erreur du type "***KeyError: <class 'bytes'>***" lors de la création de la base de données (cf. https://github.com/PyMySQL/mysqlclient-python/issues/306).

Création de la base de données :

?? Attention, si vous préférez ne pas accorder les droits "CREATE DATABASE" à l'utilisateur Pod, vous pouvez créer une base vide en amont, mais assurez-vous que cette dernière soit en "utf8_general_ci" ! Lancez ensuite les scripts ci-dessous (dans tous les cas) :

Serveur(s) Pod / Compte %userpod%

```
[%userpod%][/data/www/%userpod%/django_projects/podv2]# workon django_pod
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2]# sh create_data_base.sh
```

? Il faut vérifier que l'on se trouve bien dans l'environnement virtuel.

Creation de l'index Pod

Nous pouvons enfin vérifier le bon fonctionnement de l'ensemble (l'erreur "404-index_not_found_exception : no such index" affichée lors de la suppression est normale, puisque l'index n'existe pas encore, mais nous devons supprimer avant de créer un index dans ES) :

Serveur(s) Pod / Compte %userpod%

```
[%userpod% ~]$ cd
[%userpod% ~]$ source .bashrc
[%userpod% ~]$ cd /data/www/%userpod%/django_projects/podv2
[%userpod% /data/www/%userpod%/django_projects/podv2]$ workon django_pod
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2] python3.6 manage.py create_pod_index
```

Création du super user

Serveur(s) Pod / Compte %userpod%

```
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2]# python3.6 manage.py createsuperuser
```

? Il faut vérifier que l'on se trouve bien dans l'environnement virtuel.

Gestion des fichiers "static"

Par défaut, chaque application possède ses propres fichiers statiques (images, css, javascript etc.) La commande ci-après permet de les centraliser pour que ces fichiers soient distribués plus facilement par le fontal web.

Serveur(s) Pod / Compte %userpod%

```
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2]# python3.6 manage.py collectstatic
```

?Il faut vérifier que l'on se trouve bien dans l'environnement virtuel.

Gestion des droits des fichiers

Il est nécessaire de donner les droits à l'utilisateur Nginx (cf. configuration Nginx) aux fichiers Web de l'application.

Dans mon cas, voici les droits que j'ai positionné en tant que root (ou sudo) :

Serveur(s) Pod / Compte root

```
# Droits (sinon cela peut provoquer une erreur "ImportError: No module named site")
[root ~]# chmod 755 /data/www/%userpod%/
# Attention, les vidéos vont être uploadées dans le répertoire « media ». Il faut donner le droit en lecture et
en écriture à l'utilisateur web (nginx)
[root ~]# mkdir /data/www/%userpod%/media
[root ~]# chown -R %userpod%:nginx /data/www/%userpod%/media/
[root ~]# chown -R %userpod%:nginx /data/www/%userpod%/.virtualenvs/
[root ~]# chown -R %userpod%:nginx /data/www/%userpod%/django_projects/
[root ~]# chmod 755 /data/www/%userpod%/media -R
# Sécurisation
[root ~]# chmod -R o-rx /data/www/%userpod%/django_projects/podv2/
```

Spécificités pour l'établissement

Traduction

Pour réaliser les traductions, il suffit de modifier le fichier **/data/www/%userpod%/django_projects/podv2/pod/locale/fr/LC_MESSAGES/django.po** puis de compiler :

Serveur(s) Pod / Compte %userpod%

```
(django_pod) [%userpod%@ts-sun-video][/data/www/%userpod%/django_projects/podv2]# django-admin.py
compilemessages
```

? Il faut vérifier que l'on se trouve bien dans l'environnement virtuel.

A l'heure actuelle, j'ai ajouté les traductions suivantes :

- Correction d'une faute : ligne 1535 (msgstr "Signaler un contenu inapproprié"),
- Ajout des traductions :
    - msgid "Help"
      msgstr "Aide"
    - msgid "Legal notice"
      msgstr "Mentions légales"
    - msgid "Top of page"
      msgstr "Haut de page"
    - msgid "enrichment"
      msgstr "Ajouter de l'enrichissement"
    - msgid "interactive"
      msgstr "Ajouter de l'interactivité"

Thème spécifique à l'établissement

La mise en place d'un thème spécifique à chaque établissement est totalement différent que pour la v1.

Fichiers CSS, Javascript et images

Pour l'utilisation de fichiers Javascript, CSS ou d'images, il est nécessaire d'utiliser un répertoire custom, directement dans le répertoire static.

Ainsi, ce répertoire custom est accessible : **/data/www/%userpod%/django_projects/podv2/pod/static/custom**.

Serveur(s) Pod / Compte %userpod%

```
[%userpod%][/]# mkdir /data/www/%userpod%/django_projects/podv2/pod/static/custom
```

Templates

Pour utiliser des templates spécifiques, il est nécessaire de :

- Réaliser une configuration dans le fichier *settings_local.py*

Serveur(s) Pod / Compte %userpod%

```
# Templates spécifiques UM
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['/data/www/%userpod%/django_projects/podv2/pod/custom/templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                # Local contexts
                'pod.main.context_processors.context_settings',
                'pod.main.context_processors.context_navbar'
            ],
        },
    },
]
```

- Utiliser un répertoire templates dans custom (**/data/www/%userpod%/django_projects/podv2/pod/custom/templates**)

Ce répertoire contient les pages spécifiques à l'établissement, sur le modèle **/data/www/%userpod%/django_projects/podv2/pod/main/templates**.

? Par exemple, pour définir une page d'erreur spécifique, il suffit de créer une page /data/www/%userpod%/django_projects/podv2/pod/custom/templates /404.html, à partir de la page /data/www/%userpod%/django_projects/podv2/pod/main/templates/404.html.

Modifications réalisées directement dans le code source de l'application

Icône dans la barre de navigation principale

Pour que l'icône d'ajout de vidéo corresponde à un (plus entouré d'un cercle) et non un plus entouré d'un carré, j'ai préféré directement modifier le code source de la page navbar.html, à savoir */data/www/%userpod%/django_projects/podv2/pod/main/templates/navbar.html* :

- remplacer, dans la ligne 51, *data-feather="plus-square"* par *data-feather="plus-circle"*

Changement de libellés dans le menu de droite, pour les enrichissements et l'interactivité

Par défaut, lorsque l'utilisateur gère une de ses vidéos, il peut ajouter des enrichissements ou de l'interactivité à cette vidéo. Dans le menu de droite, ces fonctionnalités apparaissaient en tant que "enrichment" et "interactive".

Pour plus de clarté, j'ai alors modifié cela pour afficher des libellés plus explicites; pour cela, j'ai modifié la page */data/www/%userpod%/django_projects /podv2/pod/video/templates/videos/video_edit.html* :

- ligne 139 : {% for app in THIRD_PARTY_APPS %}{% if app != "live" %}{% with urledit=app|add:':edit_'|add:app %} {% trans app %} {% endwith %}{% endif %}{% endfor %}
- ajouter les traductions correspondantes (cf. traductions ci-dessus).

Ce n'est pas très propre de modifier directement le code source de l'application, mais je préfère éviter de surcharger des pages (de plus, ce n'est pas toujours possible) juste pour des changements de libellés.

Il faut seulement penser à réaliser ces modifications à chaque mise à jour.

Modification de la ligne de commande d'encodage INUTILE en v2

Après divers tests sur Pod v1, il s'était avéré utile de modifier la ligne de commande par défaut qui permet l'encodage des vidéos (en remplaçant ***-b:v %(bv)s -maxrate %(bv)s -bufsize %(bufsize)s*** par ***-crf 23***)

Pour Pod v2, cela ne semble plus utile; par contre, l'option est positionnée - par défaut - à -crf 20, ce qui traduit une bonne qualité, mais des fichiers vidéos plus lourds !

?? Dans Pod v2, pour l'encodage MP4, il est possible d'utiliser le paramètre FFMPEG_STATIC_PARAMS.

Actions complémentaires

Une fois le serveur lancé, il m'a fallu réaliser quelques éléments de paramétrage via l'interface d'administration du site.

Accès au module d'administration

Le module d'administration n'est accessible qu'aux administrateurs du site.

Au départ, il n'y a que le compte **root** qui est considéré comme administrateur du site.

Pour donner ce droit à d'autres utilisateurs :

- il est nécessaire que l'utilisateur se soit déjà connecté,
- se connecter en tant qu'administrateur (**root** la 1° fois),
- *Administration* / module *Utilisateurs* / modifier l'utilisateur concerné
- Donner le statut *super-utilisateur*.

??Il est possible de mettre directement le statut super-utilisateur à un utilisateur en base de données: *UPDATE `auth_user` SET is_superuser = 1 WHERE email = 'xxx.xxx@umontpellier.fr'*

Mise à jour de l'URL du site en base

Pour obtenir une génération correcte des URLs des vidéos lors d'une recherche, il est obligatoire de positionner l'adresse dans le module des sites de Pod.

Pour ce faire, 2 possibilités :

- soit via l'interface d'*Administration*, module *Sites*, définir un nom de domaine.
- soit directement en mettant à jour la base de données, table *django_site*, colonnes *domain* et *name*.

```
??UPDATE `django_site` SET `domain` = 'video.umontpellier.fr', `name` = 'video.umontpellier.fr' WHERE `django_site`.`id` = 1;
```

Utilisation de Pod avec un serveur Web

Côté serveur Web, je pensais partir sur une configuration similaire à Pod v1, à savoir du Apache + WSGI.

Malheureusement, il semblerait que Pod v2 ne fonctionne pas sous cette configuration (ou du moins, cela nécessiterait plus de temps pour trouver une solution).

Du coup, j'ai suivi les recommandations et mis en place du Nginx + uWSGI.

Ci-dessous, je mets l'ensemble des documentations concernant ces solutions, même si cela n'est pas concluant.

Utilisation du serveur de développement

Serveur(s) Pod / Compte %userpod%
`(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2]# python3.6 manage.py runserver 162.38.xx.xx:8080`

? Il faut vérifier que l'on se trouve bien dans l'environnement virtuel.

Cette solution ne peut être utilisée en production !

Utilisation avec Apache + WSGI (comme pour la v1) NON CONCLUANT

Documentations de référence :

- Réalisation sur Pod v1 : https://wikidsin.umontpellier.fr/x/HQCYBg
- https://exordium.readthedocs.io/en/stable/apache_deployment_howto.html

```
[root]# yum install mod_wsgi
[root]# yum install expat-devel
[root]# yum install httpd-devel
[root]# yum install python36u-mod_wsgi
[root]# # ATTENTION: nouveau fichier créé /etc/httpd/conf.modules.d/10-wsgi-python3.6.conf
[root]# # Configurer 10-wsgi.conf
[root]# # Configurer pod.conf
[root]# chmod 755 /data/www/%userpod%/django_projects/podv2/pod/wsgi.py

[%userpod%][/data/www/%userpod%/django_projects/podv2]# pip3 install requests
```

Utilisation avec Nginx + uWSGI

Documentations de référence :

- https://uwsgi-docs.readthedocs.io/en/latest/tutorials/Django_and_nginx.html#basic-uwsgi-installation-and-configuration
- https://www.digitalocean.com/community/tutorials/how-to-set-up-uwsgi-and-nginx-to-serve-python-apps-on-centos-7

Installation Nginx

Serveur(s) Pod / Compte root

```
[root]# yum install --enablerepo=epel nginx
[root]# emacs /etc/nginx/nginx.conf # vérifier que include /etc/nginx/conf.d/*.conf n'est pas en commentaire.
Ne pas configurer d'adresse IP ni de server_name.
[root]# systemctl start nginx
[root]# systemctl enable nginx
```

Pod permet 2 systèmes de barre de progression :

- Le premier nécessite de recompiler directement les sources d'nginx (je ne vous le recommande pas, cela complique les mises à jour)
  - Au cas où, suivez l'installation du module nginx upload_progress
- Le second ne modifie rien dans nginx, et permet en plus de reprendre un téléchargement en cas de coupure réseau (à condition que l'utilisateur ne recharge pas la page) : c'est le mode "CHUNK" que nous préconisons ici. ("USE_CHUNKED_UPLOAD = True" dans settings_local.py)

Configuration Nginx

Pour configurer Nginx, je me base sur le fichier fourni par Pod v2, que je copie dans *custom* et que je personnalise :

Serveur(s) Pod / Compte %userpod%

```
[%userpod%]# cd /data/www/%userod%/django_projects/podv2/
# Copie du fichier
[%userpod% /data/www/%userpod%/django_projects/podv2]# cp pod_nginx.conf pod/custom/
# Édition du nouveau fichier
[%userpod /data/www/%userpod%/django_projects/podv2]# emacs pod/custom/pod_nginx.conf
# Création d'un lien symbolique, directement dans le répertoire de Nginx
[%userpod /data/www/%userpod%/django_projects/podv2]# sudo ln -s /data/www/%userpod%/django_projects/podv2/pod
/custom/pod_nginx.conf /etc/nginx/conf.d/pod_nginx.conf
[%userpod% /data/www/%userpod%/django_projects/podv2]# sudo systemctl restart nginx
```

Voici un exemple de contenu du fichier ***pod_nginx.conf*** :

Serveur(s) Pod / Fichier pod_nginx.conf

```
# the upstream component nginx needs to connect to
upstream podupstream {
    server unix:///data/www/%userpod%/django_projects/podv2/uwsgi/podv2.sock max_fails=2 fail_timeout=3s;
}

# configuration of the server
server {
    # the port your site will be served on
    listen      80;
    # the domain name it will serve for
    server_name xxxxx.umontpellier.fr; # substitute your machine's IP address or FQDN
    charset     utf-8;

    # max upload size
    client_max_body_size 4G;   # adjust to taste

    # Django media
    location /media  {
        alias /data/www/%userpod%/media;  # your Django project's media files - amend as required
    }

    location /static {
                # your Django project's static files - amend as required
        alias /data/www/%userpod%/django_projects/podv2/pod/static;
    }

    # Finally, send all non-media requests to the Django server.
    location / {
        uwsgi_pass  podupstream;
        include     /data/www/%userpod%/django_projects/podv2/uwsgi_params;
    }
}
```

Installation uWSGI

Serveur(s) Pod / Compte root

```
[root]# yum install pcre-devel
[root]# cd /data/www/%userod%/django_projects/podv2/
[root /data/www/%userpod%/django_projects/podv2]# pip3 install uwsgi
```

Configuration uWSGI

Pour configurer uWSGI, je me base sur le fichier fourni par Pod v2, que je copie dans *custom* et que je personnalise :

Serveur(s) Pod / Compte %userpod%

```
[%userpod%]# cd /data/www/%userod%/django_projects/podv2/
# Copie du fichier
[%userpod% /data/www/%userpod%/django_projects/podv2]# cp pod_uwsgi.ini pod/custom/.
# Edition du nouveau fichier
[%userpod% /data/www/%userpod%/django_projects/podv2]# emacs pod/custom/pod_uwsgi.ini
# Création d'un répertoire, qui contiendra les fichiers de logs, socket... de uWSGI
[%userpod% /data/www/%userpod%/django_projects/podv2]# mkdir /data/www/%userpod%/django_projects/podv2/uwsgi
# Gestion des droits de ce répertoire
[%userpod% /data/www/%userpod%/django_projects/podv2]# chown %userpod%:nginx /data/www/%userpod%/django_projects
/podv2/uwsgi
[%userpod% /data/www/%userpod%/django_projects/podv2]# chmod 776 /data/www/%userpod%/django_projects/podv2/uwsgi
```

Voici le contenu du fichier ***pod_uwsgi.ini*** :

Serveur(s) Pod / Fichier pod_uwsgi.ini

```
# pod_uwsgi.ini file
[uwsgi]

# Django-related settings
# the base directory (full path)
chdir           = /data/www/%userpod%/django_projects/podv2
# Django's wsgi file
module          = pod.wsgi
# the virtualenv (full path)
home            = /data/www/%userpod%/.virtualenvs/django_pod
# process-related settings
# The master uWSGI process is necessary to gracefully re-spawn and pre-fork workers, consolidate logs, and
manage many other features.
master          = true
# maximum number of worker processes
processes       = 10
# the socket (use the full path to be safe
socket          = /data/www/%userpod%/django_projects/podv2/uwsgi/podv2.sock
chown-socket    = %userpod%:nginx
# http               = :8000
# ... with appropriate permissions - may be needed
chmod-socket    = 666
# clear environment on exit
vacuum          = true

daemonize       = /data/www/%userpod%/django_projects/podv2/uwsgi/uwsgi-pod.log

# recommended params by https://www.techatbloomberg.com/blog/configuring-uwsgi-production-deployment/
strict          = true  ; This option tells uWSGI to fail to start if any parameter in the configuration file
isn’t explicitly understood.
die-on-term     = true  ; Shutdown when receiving SIGTERM (default is respawn)
need-app        = true  ; This parameter prevents uWSGI from starting if it is unable to find or load your
application module.

# Limit process address space (vsz) (in megabytes).
# Limits the address space usage of each uWSGI (worker) process using POSIX/UNIX setrlimit(). For example,
limit-as 256 will disallow uWSGI processes to grow over 256MB of address space. Address space is the virtual
memory a process has access to. It does not correspond to physical memory. Read and understand this page before
enabling this option: http://en.wikipedia.org/wiki/Virtual_memory
# limit-as        = 5120
# max-requests    = 5000
```

Démarrage manuel des workers uWSGI

Il faut lancer la commande *uwsgi*, en pointant vers le bon fichier *pod_uwsgi.ini*, via l'environnement *django_pod*. Cela donne :

Serveur(s) Pod / Compte %userpod%

```
[%userpod%]# cd
[%userpod% /home/%userpod%]# source .bashrc
[%userpod% /home/%userpod%]# cd /data/www/%userpod%/django_projects/podv2
[%userpod% /data/www/%userpod%/django_projects/podv2]# workon django_pod
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2]# /usr/bin/uwsgi --ini /data/www/%userpod%
/django_projects/podv2/pod/custom/pod_uwsgi.ini --enable-threads --uid %userpod% --gid nginx --pidfile /data/www
/%userpod%/django_projects/podv2/uwsgi/pod.pid
```

Arrêt manuel des workers uWSGI

Serveur(s) Pod / Compte %userpod%

```
[%userpod%]# cd
[%userpod% /home/%userpod%]# source .bashrc
[%userpod% /home/%userpod%]# cd /data/www/%userpod%/django_projects/podv2
[%userpod% /data/www/%userpod%/django_projects/podv2]# workon django_pod
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2]# /usr/bin/uwsgi --stop /data/www/%userpod%
/django_projects/podv2/uwsgi/pod.pid
```

Mise en place d'un service

Pour éviter de démarrer / arrêter manuellement le système de workers uWSGI, il est nécessaire de créer un service.

Ce service s'appelle **uwsgi-pod.service**, positionné dans */etc/systemd/system/*.

Voici le contenu de ce fichier :

Serveur(s) Pod / Fichier uwsgi-pod.service (compte root)

```
# [Unit]
Description=Pod uWSGI app
After=syslog.target

[Service]
ExecStartPre=-/usr/bin/bash -c 'export WORKON_HOME=/data/www/%userpod%/.virtualenvs; export
VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3.6; cd /data/www/%userpod%/django_projects/podv2; source /usr/bin
/virtualenvwrapper.sh; workon django_pod;'
ExecStart=/usr/bin/bash -c '/usr/bin/uwsgi --ini /data/www/%userpod%/django_projects/podv2/pod/custom/pod_uwsgi.
ini --enable-threads --uid %userpod% --gid nginx --pidfile /data/www/%userpod%/django_projects/podv2/uwsgi/pod.
pid'
ExecStop=/usr/bin/uwsgi --stop /data/www/%userpod%/django_projects/podv2/uwsgi/pod.pid
User=%userpod%
Group=nginx
Restart=on-failure
Type=notify
TimeoutStopSec=15
KillSignal=SIGQUIT
RemainAfterExit=yes
StandardError=syslog
NotifyAccess=all
StartLimitBurst=0

[Install]
WantedBy=multi-user.target
limit-as=512
```

Pensez à créer le répertoire /var/run/podv2, avec les droits adéquats pour pod:nginx.

Configurer pour que ce service soit démarré lors d'un reboot :

Mise en service uwsgi-pod (compte root)

```
[root ~]# systemctl daemon-reload
[root ~]# systemctl enable uwsgi-pod.service
```

Rotation des logs

Configurer un système de logrotate pour vider régulièrement les logs de django et uwsgi. Pour cela, j'ai créé le fichier **/etc/logrotate.d/django-uwsgi** avec le contenu suivant :

/etc/logrotate.d/django-uwsgi

```
/data/www/%userpod%/django_projects/podv2/uwsgi/*.log
/data/www/%userpod%/django_projects/podv2/pod/log/*.log {
    su %userpod% nginx
    daily
    missingok
    rotate 14
    compress
    delaycompress
    notifempty
    create 0640 %userpod% nginx
    sharedscripts
    postrotate
        systemctl restart uwsgi-pod >/dev/null 2>&1
    endscript
}
```

Puis lancez la commande suivante pour vérifier que ca fonctionne :

```
logrotate -d /etc/logrotate.d/django-uwsgi
```

Migrations de Pod v2

Migration v2.0.4 vers v2.1.0

Migration effective

Afin de tests, nous sommes passé d'une version 2.0.4 à une version 2.1.0 de Pod v2 sur l'environnement de test.

Pour ce faire, voici les commandes réalisées :

Serveur(s) Pod / Migration v2.0.4 vers 2.1.0 (compte %userpod%)

```
[%userpod% ~]# cd
[%userpod% ~]# source .bashrc
[%userpod% ~]# cd /data/www/%userpod%/django_projects/podv2
[%userpod% /data/www/%userpod%/django_projects/podv2]# workon django_pod
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2] git pull
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2] python manage.py makemigrations
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2] python manage.py migrate
(django_pod) [%userpod%][/data/www/%userpod%/django_projects/podv2] python manage.py collectstatic
```

En cas d'erreur du type "*Your local changes to the following files would be overwritten by merge*", lors du *git pull*, cela signifie que des fichiers sources - qui n'auraient pas dû être modifiés - ont été modifiés (typiquement main/settings.py).

Il suffit de faire un *git checkout .* avant le *git pull*, pour ne plus avoir cette erreur (et perdre les données modifiées).

À penser

Il est nécessaire de vérifier / réaliser les étapes suivantes à chaque migration :

- la traduction
- la gestion des fichiers statiques,
- les templates spécifiques (footer.html, 404.html....) doivent, dans certains cas, être modifiés, en cas de changement majeur.
- les autres spécificités qui n'ont pas pu être pris en compte dans le système de répertoire *custom*.