installation

Installation de la plateforme

Les commandes suivantes ont été lancées sur une distribution Debian 9.4 "stretch".

Environnement

Creation de l'utilisateur Pod

```
user@pod:~$  sudo adduser pod
user@pod:~$  adduser pod sudo
user@pod:~$  su pod
```

Création de l'environnement virtuel

```
pod@pod:~$ sudo python3 -V
pod@pod:~$ sudo python -V
pod@pod:~$ sudo apt-get install -y python3-pip
pod@pod:~$ pip3 search virtualenvwrapper
pod@pod:~$ sudo pip3 install virtualenvwrapper
```

A la fin du bashrc, il faut ajouter ces deux lignes :

```
pod@pod:~$ vim .bashrc
      [..]
      export WORKON_HOME=$HOME/.virtualenvs
      export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
      source /usr/local/bin/virtualenvwrapper.sh
      [..]
```

Puis prendre en charge ces modifications :

```
pod@pod:$ source .bashrc
```

Et enfin créer l'environnement virtuel :

```
pod@pod:~$ mkvirtualenv --system-site-packages --python=/usr/bin/python3 django_pod
```

Récupération des sources

Concernant l'emplacement du projet, je conseille de le mettre dans /usr/local/django_projects

```
(django_pod)pod@pod:~$ sudo mkdir /usr/local/django_projects
```

Vous pouvez faire un lien symbolique dans votre home pour arriver plus vite dans le répertoire django_projects:

```
(django_pod)pod@pod:~$ ln -s /usr/local/django_projects django_projects
```

Placez vous dans le répertoire django_projects

```
(django_pod)pod@pod:~$ cd django_projects
(django_pod)pod@pod:~/django_projects$
```

Donnez les droits à l'utilisateur Pod de lire et d'écrire dans le répertoire :

```
(django_pod) pod@pod:~/django_projects$ sudo chown pod:pod /usr/local/django_projects
```

Vous pouvez enfin récupérer les sources :

Attention, si vous devez utiliser un proxy, vous pouvez le spécifier avec cette commande :

```
(django_pod) pod@pod:~/django_projects$ git config --global http.proxy http://PROXY:PORT
```

la récupération des sources de la V2 se font via cette commande : **git clone https://github.com/esupportail/podv2.git**

```
(django_pod) pod@pod:~/django_projects$ git clone https://github.com/esupportail/podv2.git
Clonage dans 'podv2'...
remote: Counting objects: 4578, done.
remote: Compressing objects: 100% (378/378), done.
remote: Total 4578 (delta 460), reused 564 (delta 348), pack-reused 3847
Réception d'objets: 100% (4578/4578), 4.40 MiB | 3.88 MiB/s, fait.
Résolution des deltas: 100% (3076/3076), fait.
(django_pod) pod@pod:~/django_projects$ cd podv2/
```

Applications tierces

Installation de toutes les librairies python :

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 `$> workon django_pod`

```
(django_pod) pod@pod:~/django_projects/podv2$ pip3 install -r requirements.txt
```

De même, si vous devez utiliser un proxy :

```
(django_pod) pod@pod:~/django_projects/podv2$ pip3 install --proxy="PROXY:PORT" -r requirements.txt
```

FFMPEG

Pour l'encodage des vidéos et la creation des vignettes, il faut installer **ffmpeg**, **ffmpegthumbnailer** et **imagemagick** (ne pas installer sur le serveur frontal si vous déportez l'encodage)

```
(django_pod) pod@pod:~/django_projects/podv2$ sudo aptitude install ffmpeg
(django_pod) pod@pod:~/django_projects/podv2$ sudo aptitude install ffmpegthumbnailer
(django_pod) pod@pod:~/django_projects/podv2$ sudo aptitude install imagemagick
```

Elasticsearch

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

```
(django_pod) pod@pod:~/django_projects/podv2$ sudo aptitude install openjdk-8-jre
```

Puis pour installer Elasticsearch sur Debian en utilsant les paquets, il faut suivre les instructions situées à cette adresse : https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html

Voici :

```
django_pod) pod@pod:~/django_projects/podv2$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo
apt-key add -
OK
(django_pod) pod@pod:~/django_projects/podv2$ sudo apt-get install apt-transport-https
(django_pod) pod@pod:~/django_projects/podv2$ echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main"
 | sudo tee -a /etc/apt/sources.list.d/elastic-6.x.list
deb https://artifacts.elastic.co/packages/6.x/apt stable main
(django_pod) pod@pod:~/django_projects/podv2$ sudo apt-get update && sudo apt-get install elasticsearch
```

Ensuite il faut paramétrer l'instance :

```
(django_pod) pod@pod:~/django_projects/podv2$ sudo vim /etc/elasticsearch/elasticsearch.yml
```

Pour préciser trois valeurs :

- **cluster.name**: pod-application
- **node.name**: pod-1
- **discovery.zen.ping.unicast.hosts**: ["127.0.0.1"]

Il faut enfin le lancer et vérifier son bon fonctionnement :

```
(django_pod) pod@pod:~/django_projects/podv2$ sudo /etc/init.d/elasticsearch start
(django_pod) pod@pod:~/django_projects/podv2$ curl -XGET "127.0.0.1:9200"
{
  "name" : "pod-1",
  "cluster_name" : "pod-application",
  "cluster_uuid" : "lG2pQ219SHePd4axTKBNZA",
  "version" : {
    "number" : "6.2.4",
    "build_hash" : "ccec39f",
    "build_date" : "2018-04-12T20:37:28.497551Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

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

```
(django_pod) pod@pod:~/django_projects/podv2$ cd /usr/share/elasticsearch/
(django_pod) pod@pod:/usr/share/elasticsearch$ sudo bin/elasticsearch-plugin install analysis-icu
-> Downloading analysis-icu from elastic
[=================================================] 100%
-> Installed analysis-icu
(django_pod) pod@pod:/usr/share/elasticsearch$ sudo /etc/init.d/elasticsearch restart
[ ok ] Restarting elasticsearch (via systemctl): elasticsearch.service.
```

Creation de l'index Pod

Nous pouvons enfin vérifier le bon fonctionnement de l'ensemble (l'erreur affichée lors de la deletion est normal puisque l'indice n'existe pas mais nous devons supprimer avant de créer un index dans ES):

```
(django_pod) pod@pod:~/django_projects/podv2$ python manage.py create_pod_index
DELETE http://127.0.0.1:9200/pod [status:404 request:0.140s]
An error occured during index video deletion: 404-index_not_found_exception : no such index
Successfully create index Video
(django_pod) pod@pod:~/django_projects/podv2$ curl -XGET "127.0.0.1:9200/pod/_search"
{"took":35,"timed_out":false,"_shards":{"total":2,"successful":2,"skipped":0,"failed":0},"hits":{"total":0,"max_sc
ore":null,"hits":[]}}
```

Si vous déportez l'elastic search sur une autre machine, rajouter dans le fichier settings_local.py son URL d'accès :

```
(django_pod) pod@pod:/usr/local/django_projects/podv2$ vim pod/custom/settings_local.py
```

```Python console ES_URL = ['http://elastic.domaine.fr:9200/']


## Mise en route
### Base de données SQLite intégrée

Lancer le script présent à la racine afin de créer les fichier de migration, puis de les lancer afin de créer la
base de données SQLite intégrée

```console
(django_pod) pod@Pod:~/django_projects/podv2$ sh create_data_base.sh
```

(option) Base de donnée MySQL/MariaDB

Vous pouvez utiliser une base de donnée MySQL/MariaDB sur le serveur frontal (ou sur un serveur distant) il faut installer le moteur MySql/Python :

```
(django_pod) pod@pod:/usr/local/django_projects/podv2$ sudo apt-get install python3-dev
(django_pod) pod@pod:/usr/local/django_projects/podv2$ sudo aptitude install default-libmysqlclient-dev
(django_pod) pod@pod:/usr/local/django_projects/podv2$ pip3 install mysqlclient
```

Il faut ensuite spécifier la configuration de votre base de données dans votre fichier de configuration :

```
(django_pod) pod@pod:/usr/local/django_projects/podv2$ vim pod/custom/settings_local.py
```

```Python console DATABASES = { 'default': { 'ENGINE': 'django.db.backends.mysql', 'NAME': 'mydatabase', 'USER': 'mydatabaseuser', 'PASSWORD': 'mypassword', 'HOST': '127.0.0.1', 'PORT': '3306', 'OPTIONS': { 'init_command': "SET storage_engine=INNODB, sql_mode='STRICT_TRANS_TABLES', innodb_strict_mode=1", }, } }


Il faut ensuite relancer le script présent à la racine afin de créer les fichier de migration, puis de les lancer
afin de créer la base de données :

```console
(django_pod) pod@Pod:~/django_projects/podv2$ sh create_data_base.sh
```

SuperUtilisateur

Il faut créer un premier utilisateur qui aura tous les pouvoirs sur votre instance.

```
(django_pod) pod@Pod:~/django_projects/podv2$ python manage.py createsuperuser
```

Tests

Enfin afin de vérifier que votre instance est opérationnelle, il faut lancer les tests unitaires :

```
(django_pod) pod@Pod:~/django_projects/podv2$ python manage.py test
```

Serveur de développement

Le serveur de développement permet de tester vos futurs modifications facilement.

N'hésitez pas à lancer le serveur de développement pour vérifier vos modifications au fur et à mesure.

À ce niveau, vous devriez avoir le site en français et en anglais et voir l'ensemble de la page d'accueil.

```
(django_pod) pod@Pod:~/django_projects/podv2$ python manage.py runserver ADRESSE_IP/NOM_DNS:8080
```

--> exemple : `(django_pod) pod@pod:~/django_projects/podv2$ python manage.py runserver pod.univ.fr:8080`

- Attention -

> *Quand le site est lancé, il faut se rendre dans la partie administration puis dans site pour renseigner le nom de domaine de votre instance de Pod (par défaut 'example.com').*
>
> *Avant la mise en production, il faut vérifier le fonctionnement de la plateforme dont l'ajout d'une vidéo, son encodage et sa suppression.*
>
> *Attention, pour ajouter une vidéo, il doit y avoir au moins un type de vidéo disponible. Si vous avez correctement peuplé votre base de données avec le fichier initial_data.json vous devez au moins avoir other/autres.*
>
> *il faut vérifier l'authentification CAS, le moteur de recherche etc.*

Mise en production

> *Toute la personnalisation et la configuration de votre instance de Pod peut se faire dans le répertoire pod/custom. Par exemple, pour votre configuration, il faut créer et renseigner le fichier settings_local.py :*

```
(django_pod) pod@pod:/usr/local/django_projects/podv2$ vim pod/custom/settings_local.py
```

La liste des paramètre ses trouve dans docs/configuration.md

Frontal Web NGINX / UWSGI et fichiers statiques

Pour plus de renseignement, d'explication que la documentation ci-dessous, voici le tutoriel que j'ai suivi pour mettre en place cette solution : doc{:target="_blank"}

Insallation du serveur Web NGINX et paramétrage :

```
pod@pod:~$ sudo aptitude install nginx [...] juil. 19 08:58:15 pod1 systemd[1]: Failed to start A high
performance web server and a reverse proxy server. [...] pod@pod:~$ sudo vim /etc/nginx/sites-enabled/default
[...] server { listen 80 default_server;
        #listen [::]:80 default_server;
[...]
pod@pod:~$ sudo aptitude install nginx-extras
```

Rajouter les lignes ci-dessous dans le fichier de configuration de nginx :

```
pod@pod:~$ sudo vim /etc/nginx/nginx.conf http {
[...]
# Pod Progress Bar : reserve 1MB under the name 'uploads' to track uploads upload_progress uploadp 1m;
[...]
}
```

Il faut ensuite spécifier le host pour le serveur web :

```
pod@pod:~$ cd django_projects/podv2/
pod@pod:~/django_projects/podv2$ cp pod_nginx.conf pod/custom/.
pod@pod:~/django_projects/podv2$ vim pod/custom/pod_nginx.conf
pod@pod:~/django_projects/podv2$ sudo ln -s /usr/local/django_projects/podv2/pod/custom/pod_nginx.conf /etc/nginx
/sites-enabled/pod_nginx.conf
pod@pod:~/django_projects/podv2$ sudo /etc/init.d/nginx restart
```

UWSGI

Un fichier de configuration est fourni pour faciliter l'usage d'UWSGI.

```
pod@pod:~/django_projects/podv2$ sudo pip3 install uwsgi
pod@pod:~/django_projects/podv2$ sudo uwsgi --ini pod_uwsgi.ini --enable-threads --daemonize /usr/local
/django_projects/podv2/pod/log/uwsgi-pod.log --uid pod --gid www-data --pidfile /tmp/pod.pid
pod@pod:~/django_projects/podv2$ sudo uwsgi --stop /tmp/pod.pid
```

Si vous souhaitez effectuer un changement, une personnalisation :

```
pod@pod:~/django_projects/podv2$ cp pod_uwsgi.ini pod/custom/.
pod@pod:~/django_projects/podv2$ vim pod/custom/pod_uwsgi.ini
pod@pod:~/django_projects/podv2$ sudo uwsgi --ini pod/custom/pod_uwsgi.ini --enable-threads --daemonize /usr/local
/django_projects/podv2/pod/log/uwsgi-pod.log --uid pod --gid www-data --pidfile /tmp/pod.pid
[uWSGI] getting INI configuration from pod/custom/pod_uwsgi.ini
pod@pod:~/django_projects/podv2$
```

Pour lancer le service UWSGI au démarrage de la machine :

```
pod@pod:/usr/local/django_projects/podv2/pod/log$ sudo vim /etc/systemd/system/uwsgi-pod.service

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

[Service]
ExecStart=/usr/local/bin/uwsgi --ini /usr/local/django_projects/podv2/pod/custom/pod_uwsgi.ini \
        --enable-threads \
        --pidfile /tmp/pod.pid
ExecStop=/usr/local/bin/uwsgi --stop /tmp/pod.pid
User=pod
Group=www-data
Restart=on-failure
KillSignal=SIGQUIT
Type=notify
StandardError=syslog
NotifyAccess=all

[Install]
WantedBy=multi-user.target
```

Il faut Ensuite activer le service

```
pod@pod:/usr/local/django_projects/podv2/pod/log$ sudo systemctl enable uwsgi-pod
```

Et le lancer ou l'arréter :

```
pod@pod:/usr/local/django_projects/podv2/pod/log$ sudo systemctl stop uwsgi-pod
pod@pod:/usr/local/django_projects/podv2/pod/log$ sudo systemctl restart uwsgi-pod
```

Fichiers statiques doc

Attention, il faut penser à collecter les fichiers statics pour qu'ils soient servis par le serveur web :

```
(django_pod) pod@pod:~$ python manage.py collectstatic
```

Optionnel : Serveur FTP (enregistreur)

> ***Lors de l'installation de Pod à l'unviersité de Lille, les fichiers vidéos sont stockés sur une partition montée sur "/data". Pour cela, le répertoire "media", qui contient les fichiers "utilisateurs" est créé sur "/data/media" en paramétrant la variable MEDIA_ROOT dans le fichier de configuration. De ce fait, pour des raisons de cohérence, le répertoire du serveur FTP est placé dans "/data/ftp-pod". Au niveau du logiciel, nous proposons d'utiliser vsftpd.***

Voici le détail de son installation :

```
(django_pod) pod@pod:/data$ INSTALLDIR=/data
(django_pod) pod@pod:/data$ NOMFTPUSER="ftpuser"
(django_pod) pod@pod:/data$ PASSFTPUSER="*******"
(django_pod) pod@pod:/data$ sudo mkdir $INSTALLDIR/ftp-pod
(django_pod) pod@pod:/data$ sudo useradd -m -d $INSTALLDIR/ftp-pod/ftp $NOMFTPUSER
(django_pod) pod@pod:/data$ sudo echo "$NOMFTPUSER:$PASSFTPUSER"|sudo chpasswd
(django_pod) pod@pod:/data$ sudo aptitude install vsftpd
```

Pour la configuration, il faut éditer le fichier "/etc/vsftpd.conf"

```
django_pod) pod@pod1:/data$ sudo vim /etc/vsftpd.conf
[...]
listen=YES
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
...
chroot_local_user=YES
```

A la fin du fichier de configuration, nous avons ajouté ceci :

```
local_root=/data/ftp-pod/ftp
pasv_enable=Yes
pasv_min_port=10090
pasv_max_port=10100
allow_writeable_chroot=YES
```

Enfin un petit restart pour mettre tout ca en route !

```
(django_pod) pod@pod:/data$ sudo /etc/init.d/vsftpd restart
```

Déporter l'encodage sur un autre serveur

Pré-requis :

- Il faut que votre répertoire ~/django_projects/podv2/pod/media soit partagé entre vos serveurs (montage NFS par exemple)
- Il faut utiliser une BD Mysql/MariaDB pour qu'elle soit partageable entre les serveurs Pod frontaux et encodages

Installation sur le frontal :

Il ne faut pas avoir installé ffmpeg, ffmpegthumbnailer et imagemagick. Si c'est le cas, les déinstaller :

```
(django_pod) pod@pod:~/django_projects/podv2$ sudo aptitude purge ffmpeg ffmpegthumbnailer imagemagick
```

Il faut installer rabbitmq :

```
(django_pod) pod@pod:~/django_projects/podv2$ sudo apt-get install rabbitmq-server
(django_pod) pod@pod:~/django_projects/podv2$ sudo rabbitmqctl add_user pod *mdp*
(django_pod) pod@pod:~/django_projects/podv2$ sudo rabbitmqctl set_user_tags pod administrator
(django_pod) pod@pod:~/django_projects/podv2$ sudo rabbitmqctl set_user_tags guest
(django_pod) pod@pod:~/django_projects/podv2$ sudo rabbitmqctl add_vhost rabbitpod
(django_pod) pod@pod:~/django_projects/podv2$ sudo rabbitmqctl set_permissions -p rabbitpod pod ".*" ".*" ".*"
```

Rajouter la configuration Celery/rabbitmq dans le fichier settings_local.py

```
(django_pod) pod@pod:/usr/local/django_projects/podv2$ vim pod/custom/settings_local.py
```

```Python console

## Configuration Celery sur le frontal

CELERY_TO_ENCODE = True # Active encode

CELERY_BROKER_URL = "amqp://pod:*mdp*@localhost/rabbitpod" # Define a broker


## Installation sur le serveur d'encodage :

Il faut installer pod sans réinitialiser la base et sans nginx/uwsgi

## Création du user pod
```console
user@pod:~$  sudo adduser pod
user@pod:~$  adduser pod sudo
user@pod:~$  su pod
```

Créer le virtualenv

```
pod@pod:~$ sudo python3 -V
pod@pod:~$ sudo python -V
pod@pod:~$ sudo apt-get install -y python3-pip
pod@pod:~$ pip3 search virtualenvwrapper
pod@pod:~$ sudo pip3 install virtualenvwrapper
```

A la fin du bashrc, il faut ajouter ces deux lignes :

```
pod@pod:~$ vim .bashrc
      [..]
      export WORKON_HOME=$HOME/.virtualenvs
      export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
      source /usr/local/bin/virtualenvwrapper.sh
      [..]
```

Puis prendre en charge ces modifications :

```
pod@pod:$ source .bashrc
```

Et enfin créer l'environnement virtuel :

```
pod@pod:~$ mkvirtualenv --system-site-packages --python=/usr/bin/python3 django_pod
```

Récupérer les sources

```
(django_pod)pod@pod:~$ sudo mkdir /usr/local/django_projects
```

Vous pouvez faire un lien symbolique dans votre home pour arriver plus vite dans le répertoire django_projects:

```
(django_pod)pod@pod:~$ ln -s /usr/local/django_projects django_projects
```
```

Placez vous dans le répertoire django_projects

```
(django_pod)pod@pod:~$ cd django_projects
(django_pod)pod@pod:~/django_projects$
```

Donnez les droits à l'utilisateur Pod de lire et d'écrire dans le répertoire :

```
(django_pod) pod@pod:~/django_projects$ sudo chown pod:pod /usr/local/django_projects
```

Vous pouvez enfin récupérer les sources :

Attention, si vous devez utiliser un proxy, vous pouvez le spécifier avec cette commande :

```
(django_pod) pod@pod:~/django_projects$ git config --global http.proxy http://PROXY:PORT
```

la récupération des sources de la V2 se font via cette commande : **git clone https://github.com/esupportail/podv2.git**

```
(django_pod) pod@pod:~/django_projects$ git clone https://github.com/esupportail/podv2.git
Clonage dans 'podv2'...
remote: Counting objects: 4578, done.
remote: Compressing objects: 100% (378/378), done.
remote: Total 4578 (delta 460), reused 564 (delta 348), pack-reused 3847
Réception d'objets: 100% (4578/4578), 4.40 MiB | 3.88 MiB/s, fait.
Résolution des deltas: 100% (3076/3076), fait.
(django_pod) pod@pod:~/django_projects$ cd podv2/
```

Installation des pré-requis

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 `$> workon django_pod`

```
(django_pod) pod@pod:~/django_projects/podv2$ pip3 install -r requirements.txt
```

De même, si vous devez utiliser un proxy :

```
$> pip install --proxy="PROXY:PORT" -r requirements.txt
```

Installation des librairies MySQL

```
(django_pod) pod@pod:/usr/local/django_projects/podv2$ sudo apt-get install python3-dev
(django_pod) pod@pod:/usr/local/django_projects/podv2$ sudo aptitude install default-libmysqlclient-dev
(django_pod) pod@pod:/usr/local/django_projects/podv2$ pip3 install mysqlclient
```

Installation des librairies d'encodage

```
(django_pod) pod@pod:~/django_projects/podv2$ sudo aptitude purge ffmpeg ffmpegthumbnailer imagemagick
```

Rajouter la configuration de tout ça dans le fichier de configuration

```
(django_pod) pod@pod:/usr/local/django_projects/podv2$ vim pod/custom/settings_local.py
```

```Python console CELERY_TO_ENCODE = True # Active encode CELERY_BROKER_URL = "amqp://pod:**mdp**@ip.serveur.frontal/rabbitpod" # Definit le message broker. TIME_ZONE = 'Europe/Paris' DATABASES = { 'default': { 'ENGINE': 'django.db.backends.mysql', 'NAME': 'database_name', 'USER': 'user_anme', 'PASSWORD': 'password', 'HOST': 'mysql_host_ip', 'PORT': '3306', 'OPTIONS': { 'init_command': "SET storage_engine=INNODB, sql_mode='STRICT_TRANS_TABLES', innodb_strict_mode=1", }, } } ES_URL = ['http://elastic.domaine.fr:9200/'] EMAIL_HOST = 'smtp.domaine.fr' EMAIL_PORT = 25 DEFAULT_FROM_EMAIL = 'noreply@pod.domaine.fr' SERVER_EMAIL = 'noreply@pod.domaine.fr'

ADMINS = ( ('Bob', 'bob@domaine.fr'), )


### Activer Celery sur le serveur d'encodage

Mettre le contenu de https://github.com/celery/celery/blob/4.2/extra/generic-init.d/celeryd dans /etc/init.d
/celeryd
```console
(django_pod) pod@pod-enc:~/django_projects/podv2$ sudo vim /etc/init.d/celeryd
(django_pod) pod@pod-enc:~/django_projects/podv2$ sudo chmod u+x /etc/init.d/celeryd
```

Créer le fichier default associé :

```
(django_pod) pod@pod-enc:/usr/local/django_projects/podv2$ sudo vim /etc/default/celeryd
```
```

```
CELERYD_NODES="worker1"                                                 # Nom du/des worker(s)
DJANGO_SETTINGS_MODULE="pod.settings"                                 # settings de votre Pod
CELERY_BIN="/home/pod/.virtualenvs/django_pod/bin/celery"                 # répertoire source de celery
CELERY_APP="pod.main"                                                # application où se situe celery
CELERYD_CHDIR="/usr/local/django_projects/podv2"                     # répertoire du projet Pod (où se trouve
manage.py)
CELERYD_OPTS="--time-limit=86400 --concurrency=1 --maxtasksperchild=1"         # options à appliquer en plus sur
le comportement du/des worker(s)
CELERYD_LOG_FILE="/var/log/celery/%N.log"                               # fichier log
CELERYD_PID_FILE="/var/run/celery/%N.pid"                               # fichier pid
CELERYD_USER="pod"                                                      # utilisateur système utilisant
celery
CELERYD_GROUP="pod"                                                     # groupe système utilisant celery
CELERY_CREATE_DIRS=1                                                    # si celery dispose du droit de
création de dossiers
CELERYD_LOG_LEVEL="INFO"                                                # niveau d'information qui seront
inscrit dans les logs
```

Démarrer Celeryd

```
(django_pod) pod@pod-enc:~/django_projects/podv2$ sudo /etc/init.d/celeryd start
```