Installation Pod V2

Installation de la plateforme

Les commandes suivantes ont été lancées sur une distribution Debian 9.4 "stretch". Elles fonctionnent également sur une version 10 de debian.

Attention, le dépôt a été renommé en Esup-Pod, la récupération se fait donc maintenant via cette url : https://github.com/EsupPortail/Esup-Pod.git

Pré-requis

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
```

si des problèmes de permissions et/ou de path apparaissent, il suffit de faire un sudo chmod 755 sur /usr/local/bin/* et /usr/local/lib/python3.7/dist-packages/

il est possible, depuis la v3.4 de python de faire autrement pour créer l'env virtuel, voir cette documentation : https://linuxconfig.org/how-to-set-up-a-python-virtual-environment-on-debian-10-buster

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/Esup-Pod.git podv2**

```
(django_pod) pod@pod:~/django_projects$ git clone https://github.com/EsupPortail/Esup-Pod.git podv2
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 apt-get install ffmpeg
(django_pod) pod@pod:~/django_projects/podv2$ sudo apt-get install ffmpegthumbnailer
(django_pod) pod@pod:~/django_projects/podv2$ sudo apt-get install imagemagick
```

Elasticsearch

Installation de Java

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

Sur Debian 9 :

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

Sur Debian 10 :

```
(django_pod) pod@pod:~/django_projects/podv2$ sudo apt-get install openjdk-11-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

Vous pouvez install Elasticsearch en version 6 ou en version 7

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
```

Pour Elasticsearch 6 :

```
(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
```

Pour Elasticsearch 7 :

```
(django_pod) pod@pod:~/django_projects/podv2$ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable
main" | sudo tee /etc/apt/sources.list.d/elastic-7.x.list
(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 ces valeurs :
Pour Elasticsearch 6 :

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

Pour Elasticsearch 7 :

- **cluster.name**: pod-application
- **node.name**: pod-1
- **discovery.seed_hosts**: ["127.0.0.1"]
- **cluster.initial_master_nodes**: ["pod-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 fichiers de migration, puis de les lancer afin de créer la base de données SQLite intégrée

- Pour Pod < 3.0 :

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

- Pour Pod >= 3.0

```
console (django_pod) pod@Pod:~/django_projects/podv2$ make createDB
```

Fichier de configuration settings_local.py

Vous devez créer un fichier de configuration local dans le dossier pod/custom.

Vous mettez dans ce fichier uniquement les variables dont vous voulez changer la valeur par défaut. Vous trouverez ci-dessous un exemple de fichier avec les principales variables à modifier : connexion à la base de donné, un fichier css custom, le thème green de pod, retirer le langage nl, etc ... Vous pouvez adapter ce fichier et le coller dans le votre.

```
"""Django local settings for pod_project.Django version : 1.11.10."""

##
# The secret key for your particular Django installation.
#
# This is used to provide cryptographic signing,
# and should be set to a unique, unpredictable value.
#
# Django will not start if this is not set.
# https://docs.djangoproject.com/en/1.11/ref/settings/#secret-key
#
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'A_CHANGER'

##
# 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 = 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 = ['localhost', '192.168.1.8']

##
# 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 = (
    ('Name', 'adminmail@univ.fr'),
)
# configuration exemple pour utiliser une base de données MySql,
# voir ci-apprès l'installation de lib tierce nécessaire
# il est possible d'utiliser d'autres moteur de bases de données (PostGreSql...)
# https://docs.djangoproject.com/fr/1.11/ref/databases/
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",
         },
    }
}

##
# Internationalization and localization.
#
# https://docs.djangoproject.com/en/1.11/topics/i18n/
# https://github.com/django/django/blob/master/django/conf/global_settings.py
LANGUAGES = (
    ('fr', 'Français'),
    ('en', 'English')
)

#Hide Users in navbar
HIDE_USER_TAB = True
# Hide Types tab in navbar
HIDE_TYPES_TAB = True
# Hide Tags
HIDE_TAGS = True
# Hide disciplines in navbar
HIDE_DISCIPLINES = True

##
# 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.univ.fr'
EMAIL_PORT = 25
DEFAULT_FROM_EMAIL = 'noreply@univ.fr'

# https://docs.djangoproject.com/fr/1.11/ref/settings/#std:setting-SERVER_EMAIL
```

```
SERVER_EMAIL = 'noreply@univ.fr'

##
# THIRD PARTY APPS OPTIONNAL
#
USE_PODFILE = True

##
# TEMPLATE Settings
#
TEMPLATE_VISIBLE_SETTINGS = {

    'TITLE_SITE': 'Lille.Pod',
    'TITLE_ETB': 'Université de Lille',
    'LOGO_SITE': 'img/logoPod.svg',
    'LOGO_COMPACT_SITE': 'img/logoPod.svg',
    'LOGO_ETB': 'img/logo_etb.svg',
    'LOGO_PLAYER': 'img/logoPod.svg',
    'FOOTER_TEXT': (
        '42, rue Paul Duez',
        '59000 Lille - France',
        ('<a href="https://goo.gl/maps/AZnyBK4hHaM2"'
            ' target="_blank">Google maps</a>')
    ),
    'LINK_PLAYER': 'http://www.univ-lille.fr',
    'CSS_OVERRIDE': 'custom/mycss.css',
    'PRE_HEADER_TEMPLATE': ''
}
# Choose a theme for your pod website
# 'default' is the simpliest, bootstrap $enable_rounded is true
# 'green' is with a dark green for primary color, $enable_rounded is false
# 'dark' is black and red, without grey background, $enable_rounded is false
USE_THEME = 'green'


# Custom Bootstrap CSS file. Example : custom/bootstrap-default.min.css
BOOTSTRAP_CUSTOM = ''
```

Vous trouverez l'ensemble des variables disponibles sur cette page :

Configuration de la plateforme

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

SI ce n'est pas encore fait, vous devez spécifier la configuration de votre base de données dans votre fichier de configuration settings_local.py :

```
(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",
         },
    }
}
```

```
Attention : il faut créer la Base en spécifiant CHARACTER SET
CREATE DATABASE pod CHARACTER SET utf8;
sinon une erreur de type "Specified key was too long; max key length is" apparaitra

 Il faut ensuite relancer le script présent à la racine afin de créer les fichiers 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 --settings=pod.main.test_settings
```

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
```

Vous pouvez reprendre le fichier du serveur de dévelopement et à minima supprimer la ligne DEBUG = True. Voir plus haut dans la page.

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 apt-get 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 apt-get 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 apt-get 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 apt-get 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
```

```
# 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

```
root@pod:~$ adduser pod
root@pod:~$ adduser pod sudo
root@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
```

À 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 apt-get install default-libmysqlclient-dev
(django_pod) pod@pod:/usr/local/django_projects/podv2$ pip3 install mysqlclient==2.0.3
```

Installation des librairies d'encodage

```
(django_pod) pod@pod:~/django_projects/podv2$ sudo apt-get install 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
```

```
CELERY_TO_ENCODE = True # Active encode
CELERY_BROKER_URL = "amqp://pod:mdp@ip.serveur.frontal/rabbitpod" # Definit le message broker.
CELERY_TASK_ACKS_LATE=True # permet de ne traiter que une tache à la fois
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'), )
LANGUAGES = (
    ('fr', 'Français'),
    ('en', 'English')
)
MODELTRANSLATION_FALLBACK_LANGUAGES = ('fr', 'en')
USE_PODFILE = True
```

Activer Celery sur le serveur d'encodage

Mettre le contenu de https://raw.githubusercontent.com/celery/celery/4.3/extra/generic-init.d/celeryd dans /etc/init.d/celeryd

```
(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). Ajoutez autant de
workers que de tache à executer en paralelle.
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 --max-tasks-per-child=1  --prefetch-multiplier=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
```

Pour vérifier si Celery fonctionne bien :

```
celery -A pod.main worker -l info
```

Installation de Shibboleth SP pour l'authentification Shibboleth

Vous pouvez suivre le tutoriel suivant pour installer un SP Shibboleth sur une distribution Debian : https://tuakiri.ac.nz/confluence/display/Tuakiri/Install+Shibboleth+SP+on+Debian+Based+linux

Mise en place de l'encodage distant

- Générer une paire de clés pour l'utilisateur pod sur le serveur podv2
- Copier la clé publique sur le serveur d'encodage distant dans le fichier .ssh/authorized_key (hpc)
- L'utilisateur hpc doit aussi pouvoir se connecter via ssh sur le serveur podv2. Generer sa paire de clés puis copier sa clé public dans le fichier authorized_key du serveur podv2.
- Ouvrir les firewall de maniere à ce que les serveurs puissent communiquer.
- Configurer l'encodage distant via les settings de podv2:

```
ENCODE_VIDEO = "start_remote_encode"
"""
# pourl'encodage distant, il faut préciser le login, l'host et la clé pour appeler cet encodage
# REMOTE ENCODING SETTINGS
"""
SSH_REMOTE_USER = "poduser"
SSH_REMOTE_HOST = "encode-gpu.univ.fr"
SSH_REMOTE_KEY = "/home/poduser/.ssh/id_rsa"
SSH_REMOTE_CMD = "./script.sh"
```

- Redemarrez nginx et uwsgi-pod afin que la modif soit prise en compte.
- Copier le fichier encode_gpu.py present dans podv2 (/home/pod/django_projects/pod/video) sur le serveur hpc (donnez les droits d'execution) sur le serveur d'encodage distant.
- Copier le fichier *script.sh* sur le serveur d'encodage distant. C'est ce script qui va créer le job d'encodage

```
# Video encoding automation
#
# Cyrille TOULET <cyrille.toulet@univ-lille.fr>
# Nicolas Can <nicolas.can@univ-lille.fr>

# Define variables
GPU_TYPE="RTX2080"
BASE_DIR="/home/poduser/encoding"
WORK_DIR="/tmp"
HOME_DIR="/home/poduser"
REMOTE_SERVER="pod@pod.univ.fr"
REMOTE_MEDIA_PATH="/usr/local/django_projects/podv2/pod/media"
POD_REMOTE_HOST="pod.univ.fr"
JOB_NAME="undefined-job"
INPUT_FILE="default-input.mp4"
DEBUG=False

while getopts n:i:v:u:d: option
do
case "${option}"
in
n) JOB_NAME=${OPTARG};;
i) INPUT_FILE=${OPTARG};;
v) VIDEO_ID=${OPTARG};;
u) USER_ID=${OPTARG};;
d) DEBUG=${OPTARG};;
esac
done

# Step 1: Create encoding directory
mkdir -p ${BASE_DIR}/${JOB_NAME}

# Step 2: Generate batch file
cp $(dirname $(readlink -f $0))/encode.template ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{GPU_TYPE}|${GPU_TYPE}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{BASE_DIR}|${BASE_DIR}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{WORK_DIR}|${WORK_DIR}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{JOB_NAME}|${JOB_NAME}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{REMOTE_SERVER}|${REMOTE_SERVER}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{REMOTE_MEDIA_PATH}|${REMOTE_MEDIA_PATH}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{INPUT_FILE}|${INPUT_FILE}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{VIDEO_ID}|${VIDEO_ID}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{USER_ID}|${USER_ID}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{POD_REMOTE_HOST}|${POD_REMOTE_HOST}|g" ${BASE_DIR}/${JOB_NAME}/encode.job
sed -i "s|{DEBUG}|${DEBUG}|g" ${BASE_DIR}/${JOB_NAME}/encode.job


# Step 2: Submit slurm job
cd ${HOME_DIR}/slurm-encoding-out
sbatch ${BASE_DIR}/${JOB_NAME}/encode.job
```

- Adapter les parametres suivants (exemple de paramétrages) :

# Define variables
GPU_TYPE="GTX1080" #Le type de carte graphique utilisée
BASE_DIR="/b/home/di/podtest" #dossier racine d'encodage
WORK_DIR="/b/home/di/podtest/tmp" #dossier temporaire de travail
HOME_DIR="/b/home/di/podtest" #dossier racine
REMOTE_SERVER="pod@podv272-test.di.unistra.fr" #connexion ssh au serveur podv2
REMOTE_MEDIA_PATH="/data/www/pod/media" #dossier media sur podv2
POD_REMOTE_HOST="podv272-test.di.unistra.fr" #url podv2
JOB_NAME="undefined-job" #valeur par defaut car au lancement du job un nom sera attribué
INPUT_FILE="default-input.mp4" #valeur par defaut car au lancement du job un nom sera attribué
DEBUG=True

- Copier le fichier encode.template sur le serveur d'encodage distant. Dans le même répertoire de base de script.sh qui va le transformer en encode.job (avec des sed) puis ce job va etre mis en attente

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=01:00:00
#SBATCH --job-name={JOB_NAME}
#SBATCH --mem=16G
#SBATCH --gres=gpu:{GPU_TYPE}:1


# Encoding job for POD platform
# Cyrille TOULET <cyrille.toulet@univ-lille.fr>
# Nicolas CAN <nicolas.can@univ-lille.fr>


# Print job info
echo "Job running at" $(date)" with following specs:"
echo -e " - Job name: ${SLURM_JOB_NAME}"
echo -e " - Job ID: ${SLURM_JOB_ID}"
echo " - Allocated nodes: ${SLURM_JOB_NODELIST}"
echo " - Allocated CPU cores: ${SLURM_NTASKS}"
echo -e " - Allocated GPU: /dev/nvidia${CUDA_VISIBLE_DEVICES}\n"

mkdir -p {WORK_DIR}/{JOB_NAME}

# Step 1: Download input file
echo "$> scp {REMOTE_SERVER}:{REMOTE_MEDIA_PATH}/videos/{USER_ID}/{INPUT_FILE} {WORK_DIR}/{JOB_NAME}/"
scp {REMOTE_SERVER}:{REMOTE_MEDIA_PATH}/videos/{USER_ID}/{INPUT_FILE} {WORK_DIR}/{JOB_NAME}/

# Step 2: Encode video
module load ffmpeg/4.2.3 nvidia/cuda/10.0/compilers
module load anaconda3/2020.02
echo "$> python3 /home/nicolas.can/pod-encoding/encode_gpu.py --device ${CUDA_VISIBLE_DEVICES} --dir
{WORK_DIR}/{JOB_NAME} --input {INPUT_FILE} --job ${SLURM_JOB_ID} --name {JOB_NAME}"
python3 /home/nicolas.can/pod-encoding/encode_gpu.py --device ${CUDA_VISIBLE_DEVICES} --dir {WORK_DIR}/
{JOB_NAME} --input {INPUT_FILE} --job ${SLURM_JOB_ID} --name {JOB_NAME} --debug {DEBUG}
module purge

# Step 3: Upload output files
scp -r {WORK_DIR}/{JOB_NAME}/{VIDEO_ID} {REMOTE_SERVER}:{REMOTE_MEDIA_PATH}/videos/{USER_ID}

# Step 4: Clear video files
SIZE=$(du -sh {BASE_DIR}/{JOB_NAME})
rm -rf {BASE_DIR}/{JOB_NAME}
echo "End of encoding job. Release ${SIZE} for job {JOB_NAME}"
SIZE=$(du -sh {WORK_DIR}/{JOB_NAME})
rm -rf {WORK_DIR}/{JOB_NAME}
echo "End of encoding job. Release ${SIZE} for job {JOB_NAME}"

# Step 5: Call a custom API to report the end of encoding
echo "ssh call import_encoded_video {VIDEO_ID}"
echo "$> ssh {REMOTE_SERVER} \"cd /home/pod/django_projects/podv2 && /home/pod/.virtualenvs/django_pod/bin
/python manage.py import_encoded_video {VIDEO_ID}\""
ssh {REMOTE_SERVER} "cd /home/pod/django_projects/podv2 && /home/pod/.virtualenvs/django_pod/bin/python
manage.py import_encoded_video {VIDEO_ID}"
```

- Adaptez les parametres SBATCH à votre infrastructure slurm dans ce fichier

#SBATCH -p pod # utiliser la file hpc standard
#BATCH -N 1-1 # nombre de noeuds min-max
#SBATCH -t 24:00:00 # temps estimé utilisépar le scheduler slurm
#SBATCH --gres=gpu:4 # on exige 1 GPU
#SBATCH --exclusive

- Adaptez l'etape 2 en fonction des modules disponibles sur votre infrastructure hpc et du chemin du fichier encode_gpu.py

Step 2: Encode video
module load ffmpeg/ffmpeg
module load python/python-3.6.2.i17

```
echo "$> python3 /b/home/di/podtest/encode_gpu.py --device ${CUDA_VISIBLE_DEVICES} --dir {WORK_DIR}/{JOB_NAME} --input {INPUT_FILE} --job ${SLURM_JOB_ID} --name {JOB_NAME}"
python3 /b/home/di/podtest/encode_gpu.py --device ${CUDA_VISIBLE_DEVICES} --dir {WORK_DIR}/{JOB_NAME} --input {INPUT_FILE} --job ${SLURM_JOB_ID} --name {JOB_NAME} --debug {DEBUG}
module purge
```

- Adaptez également les chemins de votre application podv2 pour l'étape 5

```
# Step 5: Call a custom API to report the end of encoding
echo "ssh call import_encoded_video {VIDEO_ID}"
echo "$> ssh {REMOTE_SERVER} \"cd /home/pod/django_projects && /data/www/pod/.virtualenvs/django_pod/bin/python manage.py import_encoded_video {VIDEO_ID}\""
ssh {REMOTE_SERVER} "cd /home/pod/django_projects && /data/www/pod/.virtualenvs/django_pod/bin/python manage.py import_encoded_video {VIDEO_ID}"
```

- Créer le répertoire d'encodage temporaire

```
${HOME_DIR}/tmp
```

- Créer le répertoire
  ${HOME_DIR}/slurm-encoding-out qui contiendra les logs d'encodage des jobs.
- Tester en lancant un encodage depuis votre serveur podv2.