Mise en place de plusieurs tenants

POD V2 POD V3

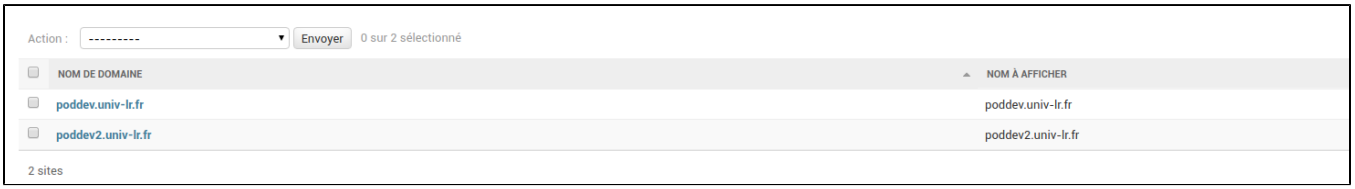
La mise en place de plusieurs tenants sur votre application Pod permettra d'héberger 2 instances de pod tout en n'ayant qu'une seule installation de Pod. Vous pourrez donc avoir un pod1.univ.fr et un pod2.univ.fr. Chaque instance possédera ses propres utilisateurs, ses propres vidéos et ses propres paramètres.

Pré-configuration dans Pod

Avant de mettre en place à proprement parler le muti-tenants vous devez vous assurer de correctement configurer vos sites dans votre espace d'administration.

> **A savoir**
>
> Un "site" dans django est l'équivalent d'un tenant. Chaque objet site est relié à un nom de domaine, un nom et un identifiant unique.

Dans l'administration des sites (dans Administration > Sites > Site) vous devez donc créer un second site (ou plus, selon vos besoins) qui possédera son propre nom de domaine.



Déploiement avec Nginx

Afin de mettre en place le multi-tenants sur pod, il conviendra de correctement paramétrer son serveur nginx. Plusieurs tenants vont donc signifier plusieurs processus uwsgi, un par tenant.

Ainsi, il suffit de prendre la documentation d'installation de pod dans la section "Mise en production" et de refaire la manipulation autant de fois que vous avez de site.

> **Aide**
>
> Lors de la mise en place du deuxième site, il conviendra de renommer les fichiers nécessaires à la configuration Nginx avec des noms différents de ceux du premier site. Par exemple on pourra créer pod2_nginx.conf,
>
> ```
> pod_uwsgi2.ini
>
>
> Dans la commande d'initialisation du processus uwsgi, il faudra également changer le uid, celui ci doit
> être uniquement à chaque processus, il en va de même pour le pidfile.
>
>
> Exemple de commandes qui peuvent être rentrées pour la création d'un second site :
>
> pod@pod:~/django_projects/podv2$ cp pod_nginx.conf pod/custom/pod_nginx2.conf
>
> pod@pod:~/django_projects/podv2$ vim pod/custom/pod_nginx2.conf
>
> pod@pod:~/django_projects/podv2$ sudo ln -s /usr/local/django_projects/podv2/pod/custom/pod_nginx2.conf
> /etc/nginx/sites-enabled/pod_nginx2.conf
>
> pod@pod:~/django_projects/podv2$ sudo /etc/init.d/nginx restart
>
>
> pod@pod:~/django_projects/podv2$ cp pod_uwsgi.ini pod_uwsgi2.ini
>
> pod@pod:~/django_projects/podv2$ cp pod_uwsgi2.ini pod/custom/.
>
> pod@pod:~/django_projects/podv2$ sudo uwsgi --ini pod/custom/pod_uwsgi2.ini --enable-threads --daemonize
> /usr/local/django_projects/podv2/pod/log/uwsgi-pod2.log --uid pod2 --gid www-data --pidfile /tmp/pod2.pid
> ```

Cas particulier : le fichier .ini

Concernant le fichier pod_uwsgi2.ini vous devrez modifier le chemin vers le socket

socket = /home/pod/django_projects/podv2/pod2v2.sock

Vous devez également utiliser un fichier settings personnalisé. Pour cela ajoutez celle ligne à la gin du fichier .ini

env = DJANGO_SETTINGS_MODULE=pod.sites2_settings #Vous pouvez choisir le nom que vous voulez

Le fichier de settings

Une fois le déploiement terminé, il est nécessaire de faire quelques ajustements dans le fichier de settings nouvellement créé

Le fichier doit au minimum contenir ces deux lignes :

from .settings import * #Cette ligne va importer les settings déjà présents dans votre application, vous pourrez les surchager par site

SITE_ID=2 #Ici il faut mettre l'id unique du site qui correspond à l'id présent dans votre espace d'administration sur le site en question

ES_INDEX = pod2 #index pour elasticsearch, il est important de le modifier pour que chaque instance ai son moteur de recherche

Dans ce fichier, il est possible de surcharger n'importe quel setting de pod pour ce site en particulier.

> **Astuce**
>
> Pour pouvoir faire une commande concernant un site en particulier il est nécessaire de préciser le site dans la commande
>
> python manage.py <commande> --settings=pod.sites2_settings
>
> Il est notamment nécessaire de le faire lors de la mise à jour de l'index de ElasticSearch

Création des "Admin site"

Tout utilisateur ayant le statut "super utilisateur" pourra se connecter sur toutes les instances de pod déployées. En revanche, si vous souhaitez avoir des administrateurs de site vous devrez procéder de cette façon :

- Dans l'administration de chaque site, créer un groupe "Admin du site" (ou autre nom au choix), lui donner les permissions souhaitées.
- Ajout les utilisateurs à ce groupe.

Les personnes dans le groupe "Admin du site" n'auront donc les permissions que sur le site du groupe en question.

Commande de mise en place

NGINX-VHOST
-> socket uwsgi
  -> fichier ini uwsgi
    -> fichier de config par tenant (tenant_settings.py) mettre tout en haut "from .settings import *"

Attention, il faut que chaque tenant est son propre identifiant de site : SITE_ID=2

Ensuite, dans ce fichier de settings, surcharger les variables propres au tenant

Ce qui donnerai

from .settings import * #Cette ligne va importer les settings déjà présents dans votre application, vous pourrez les surchager par site

SITE_ID=2 #Ici il faut mettre l'id unique du site qui correspond à l'id présent dans votre espace d'administration sur le site en question

ES_INDEX = 'podtenant' #index pour elasticsearch, il est important de le modifier pour que chaque instance ai son moteur de recherche
# USE_THEME = 'dark'
ALLOWED_HOSTS = ['video.tenant.fr']

DEFAULT_FROM_EMAIL = 'no-reply@tenant.fr'
SERVER_EMAIL = 'no-reply@tenant.fr'
HELP_MAIL = 'no-reply@tenant.fr'
CONTACT_US_EMAIL = ['contact@tenant.fr']

```
TEMPLATE_VISIBLE_SETTINGS = {
    'TITLE_SITE': 'tenant.Video',
    'TITLE_ETB': 'Tenant title',
    'LOGO_SITE': 'img/logoPod.svg',
    'LOGO_COMPACT_SITE': 'img/logoPod.svg',
    'LOGO_ETB': 'tenant/custom/images/tenant-logo-1.png',
    'LOGO_PLAYER': 'img/logoPod.svg',
    'FOOTER_TEXT': (
        ''
    ),
    'LINK_PLAYER': 'https://www.tenant.fr/',
    'CSS_OVERRIDE': 'tenant/custom/override.css',
}
```

CELERY_TO_ENCODE = True # Active encode
CELERY_BROKER_URL = "amqp://pod:p0drabbit@localhost/rabbitpod-tenant" # Define a broker

Attention, pour chaque commande lancée, il faut préciser le fichier de settings du tenant :
(django_pod) pod@pod:/usr/local/django_projects/podv2$ python manage.py runserver tenant:8080 --settings=pod.tenant_settings ^C
(django_pod) pod@pod:/usr/local/django_projects/podv2$ python manage.py index_videos --all --settings=pod.tenant_settings

Sauf pour les données communes, example BDD - il faut le faire que pour le tenant 1 (SITE_ID=1)

RAJOUTER les settings cron task :
0 3 * * * cd /usr/local/django_projects/podv2 && /home/pod/.virtualenvs/django_pod/bin/python manage.py clearsessions &>> /usr/local/django_projects/podv2/pod/log/cron_clearsessions.log 2>&1
0 4 * * * cd /usr/local/django_projects/podv2 && /home/pod/.virtualenvs/django_pod/bin/python manage.py index_videos --all &>> /usr/local/django_projects/podv2/pod/log/cron_index.log 2>&1
0 5 * * * cd /usr/local/django_projects/podv2 && /home/pod/.virtualenvs/django_pod/bin/python manage.py check_obsolete_videos >> /usr/local/django_projects/podv2/pod/log/cron_obsolete.log 2>&1
* * * * * cd /usr/local/django_projects/podv2 && /home/pod/.virtualenvs/django_pod/bin/python manage.py live_viewcounter >> /usr/local/django_projects/podv2/pod/log/cron_viewcounter.log 2>&1
0 6 * * * cd /usr/local/django_projects/podv2 && find pod/media/chunked_uploads -mtime +7 -delete

Pour l'encodage déporté (sur une autre VM) avec celery:
Préciser le nom du broker dans le fichier de configuration du tenant (frontal)
Sur la VM d'encodage, copier le fichier /etc/init.d/celeryd en /etc/init.d/celeryd-tenant
Créer un fichier de configuration pour ce broker dans /etc/default/celeryd-tenant

dans ce fichier de configuration, précier le fichier de settings à utiliser (il devra appeler le même broker), l'emplacement du fichier celery.py de votre tenant et le ou les worker(s) à lancer
$> cp pod/main/celery.py pod/custom/tenant/celery.py

$> /etc/default/cellryd-tenant

```
CELERYD_NODES="worker1tenant"                                   # Nom du/des worker(s). Ajoutez autant de workers que de tache à executer en paralelle.
DJANGO_SETTINGS_MODULE="pod.tenantsettings"                     # settings de votre Pod
CELERY_BIN="/home/pod/.virtualenvs/django_pod/bin/celery"       # répertoire source de celery
CELERY_APP="pod.custom.tenant"                                  # 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
```