

# Reprise de l'existant, entre la version 1.x et 2.x


- Contexte
  - Nomenclature / pré-requis
  - Principe de la solution
- Les étapes à réaliser sur le serveur Pod v1
- Les étapes à réaliser sur le serveur Pod v2
  - Transfert des fichier JSON
  - Pré-requis
    - Installation des packages
    - Initialiser la base de données
    - Configuration minimale en base de données
  - Import effectif des données
  - Téléchargement des vidéos
  - Encodage des vidéos

## Contexte

Un système de reprise de l'existant a été créé pour faciliter la migration d'un Pod en version 1X à un Pod en version 2.x.

La documentation de référence est accessible à l'adresse suivante: <https://oae.esup-portail.org/content/UVHC/BJTqXmPNN>

Dans les faits, il a fallu s'arranger pour que cela fonctionne correctement.

 *Voici les étapes réalisées pour l'Université de Montpellier, à savoir pour le passage d'un Pod v1.7.1 à un Pod v2.0.4, sur CentOS 7. Ainsi l'installation de Pod est réalisée dans le répertoire /data/www/%userpod% selon l'environnement.*

## Nomenclature / pré-requis

Dans cette documentation, nous utiliserons la convention de nomenclature suivante:

- podv1 : VM hébergeant la version 1 de Pod, en production.
- podv2 : VM hébergeant la version 2 de Pod.

Le compte "%userpod%" à utiliser dépend de l'environnement (de test, de pré-production ou de production).

## Principe de la solution

Vous retrouvez dans le dossier /data/www/%userpod%/django\_projects/podv2/pod/video/management/commands, 2 fichiers importants dans le cadre de cette migration de données:

- import\_data.py
- download\_video\_source\_file

La migration des données depuis podv1 vers podv2 s'effectue en 4 grandes étapes :

- Génération de l'ensemble des éléments de la base de données de podv1, sous la forme de fichiers JSON,
- Transfert et intégration de ces fichiers JSON dans la base de données de podv2,
- Rapatriement des fichiers de vidéos,
- Ré-encodage des vidéos.

## Les étapes à réaliser sur le serveur Pod v1

Sur la machine podv1, suivez les étapes suivantes :

### Podv1 / Export des données de la BD (compte %userpod%)

```
[%userpod%@podv1 ~]$ cd
[%userpod%@podv1 ~]$ source .bashrc
[%userpod%@podv1 ~]$ cd /data/www/%userpod%/django_projects/pod/pod_project/
[%userpod%@podv1 /data/www/%userpod%/django_projects/pod/pod_project]$ workon django_pod
# On lance le shell Python
(django_pod) [%userpod%@podv1 /data/www/%userpod%/django_projects/pod/pod_project]$ python manage.py shell
```

Une fois l'interpréteur shell Python lancé, il faut exécuter le script suivant :



Etant du shell Python, il est nécessaire de respecter l'indentation.

De plus, il n'est pas possible de tout lancer d'un seul coup. Pour les traitements les plus longs, il est nécessaire de faire étape par étape entre la création d'un objet et la génération du fichier JSON correspondant.

Pour les blocs pour **les tags**, **les docpods**, **les trackpods** et **les enrichpods**, il est nécessaire d'appuyer sur Entrée - et de laisser le temps de traitement - avant la génération du fichier JSON.

Exemple :

```
for p in Pod.objects.all():
    list_tag["%s" %p.id] = []
    for t in p.tags.all():
        list_tag["%s" %p.id].append(t.name)

----- APPUYER SUR Entrée ET ATTENDRE LE FIN DU TRAITEMENT AVANT DE LANCER LES 2 LIGNES SUIVANTES -----
with open("tags.json", "w") as out:
    out.write(json.dumps(list_tag, indent=2))
```

#### Podv1 / Shell Python (compte %userpod%)

```
from pods.models import Channel
from pods.models import Theme
from pods.models import Type
from pods.models import User
from pods.models import Discipline
from pods.models import Pod
from pods.models import json
from core.models import UserProfile
from django.contrib.auth.models import User
from django.contrib.flatpages.models import FlatPage

from pods.models import ChapterPods
from pods.models import ContributorPods

from django.core import serializers
jsonserializer = serializers.get_serializer("json")
json_serializer = jsonserializer()

with open("Channel.json", "w") as out:
    json_serializer.serialize(Channel.objects.all(), indent=2, stream=out)

with open("Theme.json", "w") as out:
    json_serializer.serialize(Theme.objects.all(), indent=2, stream=out)

with open("Type.json", "w") as out:
    json_serializer.serialize(Type.objects.all(), indent=2, stream=out)

owners = set(Channel.objects.all().values_list("owners", flat=True))
users = set(Channel.objects.all().values_list("users", flat=True))

list_user = owners.union(users)

#with open("User.json", "w") as out:
#    json_serializer.serialize(User.objects.filter(id__in=list_user), indent=2,
#    stream=out)

with open("User.json", "w") as out:
    json_serializer.serialize(User.objects.all(), indent=2,
    stream=out)

with open("Discipline.json", "w") as out:
    json_serializer.serialize(Discipline.objects.all(), indent=2,
    stream=out)

with open("FlatPage.json", "w") as out:
    json_serializer.serialize(FlatPage.objects.all(), indent=2,
```

```

        stream=out)

with open("UserProfile.json", "w") as out:
    json_serializer.serialize(UserProfile.objects.all(), indent=2,
        stream=out, fields=(
            'user', 'auth_type', 'affiliation', 'commentaire', 'image'))

podowner = set(Pod.objects.all().values_list("owner", flat=True))

#with open("User.json", "w") as out:
#    json_serializer.serialize(User.objects.filter(id__in=podowner), indent=2, stream=out)

video_fields = ('video', 'allow_downloading', 'is_360', 'title', 'slug',
    'owner', 'date_added', 'date_evt', 'cursus', 'main_lang', 'description',
    'duration', 'type', 'discipline', 'channel', 'theme', 'is_draft',
    'is_restricted', 'password')

with open("Pod.json", "w") as out:
    json_serializer.serialize(
        Pod.objects.all().order_by("id"), stream=out, indent=2,
        fields=video_fields)

import json

list_tag = {}
for p in Pod.objects.all():
    list_tag["%s" %p.id] = []
    for t in p.tags.all():
        list_tag["%s" %p.id].append(t.name)
with open("tags.json", "w") as out:
    out.write(json.dumps(list_tag, indent=2))

with open("Chapter.json", "w") as out:
    json_serializer.serialize(ChapterPods.objects.all().order_by('video'),
        indent=2, stream=out)

with open("Contributor.json", "w") as out:
    json_serializer.serialize(ContributorPods.objects.all().order_by('video'),
        indent=2, stream=out)

list_doc = {}
for p in Pod.objects.all():
    list_doc["%s" %p.id] = []
    for d in p.docpods_set.all():
        list_doc["%s" %p.id].append(d.document.file.name)
with open("docpods.json", "w") as out:
    out.write(json.dumps(list_doc, indent=2))

list_track = {}
for p in Pod.objects.all():
    if p.trackpods_set.all().count() > 0:
        list_track["%s" %p.id] = []
        for d in p.trackpods_set.all():
            if d.src :
                data = {}
                data['kind'] = d.kind
                data['lang'] = d.lang
                data['src'] = d.src.file.name
                list_track["%s" %p.id].append(data)
with open("trackpods.json", "w") as out:
    out.write(json.dumps(list_track, indent=2))

list_enrich = {}
for p in Pod.objects.all().order_by('id'):
    if p.enrichpods_set.all().count() > 0:
        list_enrich["%s" %p.id] = []
        for d in p.enrichpods_set.all():
            data = {}
            data['title'] = d.title
            data['stop_video'] = d.stop_video
            data['start'] = d.start

```

```

        data['end'] = d.end
        data['type'] = d.type
        data['image'] = d.image.file.name if d.image else ""
        data['richtext'] = d.richtext
        data['weblink'] = d.weblink
        data['document'] = d.document.file.name if d.document else ""
        data['embed'] = d.embed
        list_enrich["%s" % p.id].append(data)

with open("enrichpods.json", "w") as out:
    out.write(json.dumps(list_enrich, indent=2))

```

Si tout s'est bien passé, les fichiers JSON seront générés dans le répertoire courant, à savoir `/data/www/%userpod%/django_projects/pod/pod_project` :

- Channel.json
- Chapter.json
- Contributor.json
- Discipline.json
- docpods.json
- enrichpods.json
- FlatPage.json
- Pod.json
- tags.json
- Theme.json
- trackpods.json
- Type.json
- User.json
- UserProfile.json



Je ne sais pas pourquoi, mais il semblerait que la génération du fichier des User ne soit pas bonne la 1<sup>re</sup> fois. Le fichier ne contient pas tout les users. Il est nécessaire de relancer une 2<sup>e</sup> fois la génération du fichier User.json.

## Les étapes à réaliser sur le serveur Pod v2

### Transfert des fichier JSON

Créer un répertoire nommé `import` sous le répertoire `django_project`, à savoir `/data/www/%userpod%/django_projects/import`.



#### Droits

Si besoin est, positionnez les bons droits / propriétaires. A l'UM (sous root) :

```
chown %userpod%/nginx /data/www/%userpod%/django_projects/import
```

Copier les fichiers JSON générés précédemment - sur `podv1` - dans le répertoire **import**.

### Pré-requis

#### Installation des packages

Avant l'import des données, côté `Podv2`, il est nécessaire d'installer, via `pip`, les packages **requests** et **wget**.

##### Podv2 / Pré-requis (compte root ou %userpod%)

```
[%userpod%@podv2 ~]$ pip3 install requests
[%userpod%@podv2 ~]$ pip3 install wget
```

### Initialiser la base de données


Vérifier que la base est totalement vide.

D'après mes tests, pour éviter tout problème par la suite, le mieux est de supprimer toutes les tables de la base Pod, puis de tout recréer.

Pour cela :

**Podv2 / Création de la BD (compte %userpod%)**

```
[%userpod%@podv2 ~]$ cd
[%userpod%@podv2 ~]$ source .bashrc
[%userpod%@podv2 ~]$ cd /data/www/%userpod%/django_projects/podv2
[%userpod%@podv2 /data/www/%userpod%/django_projects/podv2]$ workon django_pod
# On crée les tables et données nécessaires
(django_pod) [%userpod%@podv2 /data/www/%userpod%/django_projects/podv2]$ sh create_data_base.sh
```

 Il est indispensable que le fichier de configuration *setings\_local.py* soit finalisé. 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://podv1.univ.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.**

**Configuration minimale en base de données**


Une fois la base créée, il est nécessaire d'exécuter les requêtes suivantes :









**Base de données podv2**

```
-- Mise à jour du site et création des 2 groupes (si besoin est) qui existaient déjà en v1 (positionner la
bonne valeur selon l'environnement)
UPDATE `django_site` SET `domain` = 'pod.univ.fr', `name` = 'pod.univ.fr' WHERE `django_site`.`id` = 1;
INSERT INTO `auth_group` (`id`, `name`) VALUES ('1', 'Manager'), ('2', 'Can delete file');
```

**Import effectif des données**

Il est nécessaire de lancer les commandes suivantes, dans l'ordre :

Etape	Commande à exécuter	Statut	Commentaires
1	python manage.py import_data User	OK	 Ne pas oublier de modifier le code pour pouvoir exporter l'ensemble des users.
2	python manage.py import_data Channel	OK	Chez moi, cette commande plantait en plein milieu. Après vérification, cette erreur provient du fait que certains utilisateurs ( <i>propriétaires ou utilisateurs d'une chaîne</i> ) n'ont pas été créés, car ils n'avaient jamais postés de vidéos et n'ont alors pas été repris par le script des users de l'étape 1. <div>SOLUTION J'ai modifié le script, à lancer sur Podv1 pour exporter l'ensemble des utilisateurs.</div>

3	python manage. py import_data Theme	OK	
4	python manage. py import_data Type	OK	
5	python manage. py import_data Discipline	OK	
6	python manage. py import_data FlatPage	OK	
7	python manage. py import_data UserProfile	OK	<div>SOLUTION 1</div> <p>Le fait d'exporter l'ensemble des utilisateurs fait fonctionner cette commande.</p> <div>SOLUTION 2</div> <p>Commande non utilisée.</p> <p>A priori, cette commande devrait mettre à jour les profils des utilisateurs dans la table authentication_owner. Par défaut, tous les utilisateurs créés lors de cette migration sont considérés comme des comptes étudiants locaux.</p> <p>Après vérification, la table authentication_owner est mise à jour à chaque connexion de l'utilisateur (paramétrage UM : <i>POPULATE_USER: ldap</i>).</p> <p>Ainsi, cette commande ne me paraît pas utile et ne sera pas utilisée chez nous.</p>
8	python manage. py import_data Pod	OK	
9	python manage. py import_data tags	OK	<div>SOLUTION</div> <p>Voir les différentes remarques ci-dessous :</p> <p> : le fichier tags.json n'existe que si sa génération s'est bien déroulée (cf. avertissement sur le serveur podv1).</p> <p> : Il peut y avoir des erreurs du style "Data too long for column 'tags' at row 1".</p> <p>Pour éviter cela, il faut modifier son <i>settings_local.py</i>, et remplacer la ligne</p> <pre>'OPTIONS': {'init_command': "SET storage_engine=INNODB, sql_mode='STRICT_TRANS_TABLES', innodb_strict_mode=1;"} par 'OPTIONS': {'init_command': "SET storage_engine=INNODB;"}</pre> <p>Le fait de désactiver le mode <b>STRICT</b> permet que les données soient tronquées, et évite de faire planter l'import. A la place, cela va juste générer un warning du type <i>Warning: (1265, "Data truncated for column 'tags' at row 1")</i>.</p> <p> : certaines lignes peuvent avoir des erreurs, du style 2459 <i>does not exist</i>. En vérifiant, je ne sais pas à quoi cela correspond et cela ne semble pas avoir d'incidence.</p> <p> : des nouveaux mots-clés sont créés via cette migration (Ex: <i>de, l, la, ses...</i>). Il faudra les supprimer manuellement par la suite.</p> <p> <b>PENSEZ A RÉACTIVER le mode STRICT dans <i>settns_local.py</i> par la suite.</b></p>
10	python manage. py import_data Chapter	OK	
11	python manage. py import_data Contributor	OK	
12	python manage. py import_data docpods	OK	 : le fichier docpods.json n'existe que si sa génération s'est bien déroulée (cf. avertissement sur le serveur podv1).
13	python manage. py import_data trackpods	OK	 : le fichier trackpods.json n'existe que si sa génération s'est bien déroulée (cf. avertissement sur le serveur podv1).
14	python manage. py import_data enrichpods	OK	 : le fichier enrichpods.json n'existe que si sa génération s'est bien déroulée (cf. avertissement sur le serveur podv1).

## Téléchargement des vidéos

Avant de pouvoir télécharger les vidéos, il faut vous assurer qu'elles sont toutes accessibles en téléchargement sur podV1. Voici un scripts pour mettre à jour les droits :

```
for dossier in $(ls -d videos/*)
do
    for fichier in $(ls -F $dossier| grep -v '/$')
    do
        if test -f $dossier/$fichier; then
            chmod a+r $dossier/$fichier
        fi
    done
done
```



#### Conseil

Avant de lancer les téléchargements, si vous avez beaucoup de vidéos à télécharger, et que vous utilisez la file d'attente d'encodage sur podv2, nous vous conseillons de mettre chaque fichier en file d'attente d'encodage dès son téléchargement terminé.

Pour cela, il suffit d'ajouter la ligne suivante :

```
vid.launch_encode = True
```

juste avant la ligne "vid.save()" dans le fichier

```
video/management/commands/download_video_source_file
```

Une fois la base de podv2 alimentée, voici le script permettant de récupérer les vidéos :

#### Podv2 / Téléchargement des fichiers vidéos (compte %userpod%)

```
[%userpod%@podv2 ~]$ cd
[%userpod%@podv2 ~]$ source .bashrc
[%userpod%@podv2 ~]$ cd /data/www/%userpod%/django_projects/podv2/
[%userpod%@podv2 /data/www/%userpod%/django_projects/podv2]$ workon django_pod
# Shell Python
(django_pod) [%userpod%@podv2 /data/www/%userpod%/django_projects/podv2]$ python manage.py shell
```

Le code ci-dessous - à exécuter sur Podv2 - permet de récupérer l'ensemble des fichiers vidéos, depuis le serveur Podv1 de production (cf. *settings\_local.py* paramètres *FROM\_URL* et *MEDIA\_ROOT*).

#### Podv2 / Shell Python (compte %userpod%)

```
from django.core.management import call_command
from pod.video.models import Video
Video.objects.all().count()
list_videos = Video.objects.all().order_by('id')
for vid in list_videos[1:3000]:
    call_command('download_video_source_file', vid.id)
```

Pour récupérer une seule vidéo (*exemple : 2184*) :

#### Podv2 / Shell Python (compte %userpod%)

```
from django.core.management import call_command
from pod.video.models import Video
call_command('download_video_source_file', 2184)
```

Pour ne récupérer que quelques vidéos, qui correspondent au filtre (*Exemple de filtre : id=2189*) :

### Podv2 / Shell Python (compte %userpod%)

```
from django.core.management import call_command
from pod.video.models import Video
Video.objects.all().count()
list_videos = Video.objects.filter(id=2189).order_by('id')
for vid in list_videos[1:10]:
    call_command('download_video_source_file',vid.id)
```

## Encodage des vidéos



Avant l'encodage des vidéos, il est indispensable de vérifier le paramètre ***EMAIL\_ON\_ENCODING\_COMPLETION = False*** dans le fichier *settings\_local.py*.

Sinon, des milliers d'emails seront envoyés aux utilisateurs pour rien.

Vérifier également que le paramètre soit bien en compte, et que le fichier de configuration n'ait pas été mis dans le cache (redémarrer les services *rabbitmq-server* et *celeryd* sur les serveurs adéquats).

Une fois la base alimentée et les fichiers vidéos définis, il reste à tout ré-encoder.

En effet, entre la v1 et la v2, le type d'encodage des vidéos a changé : les vidéos sont maintenant encodées et en MP4 et en HLS.

Pour ré-encoder l'ensemble des vidéos, le mieux est d'utiliser l'interface d'administration et de lancer le ré-encodage pour l'ensemble des vidéos.

Avec le système RabbitMQ / Celery, l'encodage est déporté et se réalise via une Job Queue.