Gestion de l'encodage, de la transcription et de l'xAPI

1. En local

Par défaut, l'application exécute les taches d'encodage et de transcription sur la même machine que celle sur laquelle elle tourne.

Si l'encodage et la transcription sont effectués sur la même machine que le frontal, leur execution est "threadée" (exécuté dans un sous processus).

Pour l'xAPI, le nombre de requête oblige à déporter son traitement sur une autre machine.

Pour configurer ces taches et leur execution, vous pouvez vous reporter sur le fichier de configuration disponible à cette adresse : https://github.com/EsupPortail/Esup-Pod/blob/master/CONFIGURATION_FR.md#configuration-application-video-encodage-et-transcription

Redis

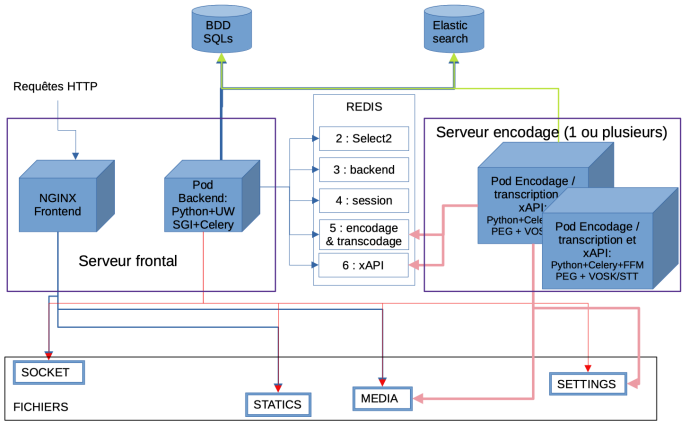
Dans le cas d'un usage déporté (en micro-service ou non), vous aurez besoin d'ouvrir l'accès à votre Redis. (voir page installation de Pod pour son installation).

Vous pouvez, soit le laisser sur le frontal web de Pod, soit l'installer sur une machine dédiée.

2. Déporté sur une machine distante

Vous pouvez déporter l'encodage, la transcription et le traitement de l'xAPI sur des environnements distants (VM ou Docker) avec soit un seul environnement pour les 3 taches, soit "n" environnements pour chaque tache.

le déport actuel se fait selon le schéma suivant :

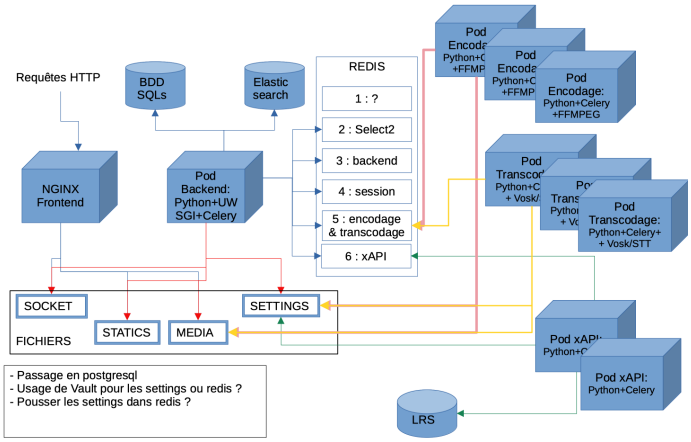


- Déporter l'encodage sur un ou plusieurs serveurs en Pod V3
- Installation de l'autotranscription en Pod V3 (avec possibilité de déport sur le serveur d'encodage)
- Mise en place de l'xAPI sur le serveur d'encodage

3. Déporté sur une machine ou un docker en microservice

Depuis la version 3.4.0, il est possible de déporter l'encodage, la transcription et l'xAPI en micro-service. Ces micro-services sont autonome et ne nécessite pas de lien avec la base de données ou le moteur de recherche comme précédemment.

Cela se faite selon le schéma suivant :



Vous avez des dockerfiles pour chaque micro-service à disposition dans le code source de Pod :

- Encodage : (lien à venir)
- Transcription : (lien à venir)
- xAPI : (lien à venir)

Il faut que chaque service ait accès au même espace de fichier Pod (espace partagé) et accès à Redis qui va jouer de role de file d'attente pour les taches d'encodage, de transcription ou d'envoi xAPI.

Chaque micro-service est lancé via une commance Celery.

3.1 Microservice Encodage

Nous appellerons dans la suite de cette documentation, **serveur Pod backend** le serveur où la partie web serveur est installée et **serveur Pod encodage** le serveur où est déporté l'encodage

Pré-requis :

- Il faut que votre répertoire "`podv3`" du serveur backend soit partagé entre vos serveurs (montage NFS par exemple)

Configuration sur le serveur Pod backend :

Dans le fichier `settings_local.py`

```
(django_pod3) pod@pod:/usr/local/django_projects/podv3$ vim pod/custom/settings_local.py
```

```
# Configuration Celery sur le frontal
USE_REMOTE_ENCODING_TRANSCODING = True
ENCODING_TRANSCODING_CELERY_BROKER_URL = "redis://redis:6379/5"
```

Installation sur le serveur d'encodage :

Installation des lib tierces

```
(django_pod3) pod@pod-encodage:/usr/local/django_projects/podv3$ apt-get update && apt-get install -y ffmpeg \
                          ffmpegthumbnailer \
                          imagemagick
```

Installation des lib python (dans un environnement virtuel)

```
(django_pod3) pod@pod-encodage:/usr/local/django_projects/podv3$ pip3 install --no-cache-dir -r requirements-
encode.txt
```

Il suffit ensuite de lancer Celery via la commande suivante (vous pouvez également créer un fichier init.d pour lancer cette commande ou ajouter --detach pour lancer en mode démon)

```
(django_pod3) pod@pod-encodage:/usr/local/django_projects/podv3$ celery -A pod.video_encode_transcript.
encoding_tasks worker -l INFO -Q encoding --concurrency 1 -n encode
```

3.2 Microservice Transcodage

Nous appellerons dans la suite de cette documentation, **serveur Pod backend** le serveur où la partie web serveur est installée et **serveur Pod transcodage** le serveur où est effectué le transcodage

Pré-requis :

- Il faut que votre répertoire "`podv3`" du serveur backend soit partagé entre vos serveurs (montage NFS par exemple)

Configuration sur le serveur Pod backend :

Dans le fichier `settings_local.py`

```
(django_pod3) pod@pod:/usr/local/django_projects/podv3$ vim pod/custom/settings_local.py
```

```
# Configuration Celery sur le frontal
USE_REMOTE_ENCODING_TRANSCODING = True
ENCODING_TRANSCODING_CELERY_BROKER_URL = "redis://redis:6379/5"
```

Installation sur le serveur de transcodage:

Installation des lib tierces

```
(django_pod3) pod@pod-transcodage:/usr/local/django_projects/podv3$ apt-get update && apt-get install -y sox
libsox-fmt-mp3
```

Installation des lib python (dans un environnement virtuel)

```
(django_pod3) pod@pod-transcodage:/usr/local/django_projects/podv3$ pip3 install --no-cache-dir -r requirements-
transcripts.txt \
    && pip3 install --no-cache-dir -r requirements-encode.txt
```

Il suffit ensuite de lancer Celery via la commande suivante (vous pouvez également créer un fichier init.d pour lancer cette commande ou ajouter --detach pour lancer en mode démon)

```
(django_pod3) pod@pod-transcodage:/usr/local/django_projects/podv3$ celery -A pod.video_encode_transcript.
transcripting_tasks worker -l INFO -Q transcripting --concurrency 1 -n transcript
```

3.2 Microservice xAPI

Nous appellerons dans la suite de cette documentation, **serveur Pod backend** le serveur où la partie web serveur est installée et **serveur Pod xAPI** le serveur où est effectué le traitement xAPI

Pré-requis :

- Il faut que votre répertoire "`podv3`" du serveur backend soit partagé entre vos serveurs (montage NFS par exemple)

Configuration sur le serveur Pod backend :

Dans le fichier `settings_local.py`

```
(django_pod3) pod@pod:/usr/local/django_projects/podv3$ vim pod/custom/settings_local.py
```

```
USE_XAPI = True
XAPI_ANONYMIZE_ACTOR = False
XAPI_LRS_LOGIN = "XXXX"
XAPI_LRS_PWD = "XXXXX"
XAPI_LRS_URL = "http://xapi.univ.fr/xAPI/statements/"
USE_XAPI_VIDEO = True
XAPI_CELERY_BROKER_URL = "redis://redis:6379/6"
```

Installation sur le serveur de traitement xAPI:

Installation des lib python (dans un environnement virtuel) et oui, c'est le même requirement que pour l'encodage

```
(django_pod3) pod@pod-transcodage:/usr/local/django_projects/podv3$ pip3 install --no-cache-dir -r requirements-encode.txt
```

Il suffit ensuite de lancer Celery via la commande suivante (vous pouvez également créer un fichier init.d pour lancer cette commande ou ajouter --detach pour lancer en mode démon)

```
(django_pod3) pod@pod-transcodage:/usr/local/django_projects/podv3$ celery -A pod.xapi.xapi_tasks worker -l INFO -Q xapi --concurrency 1 -n xapi
```

4. Monitoring

Pour monitorer la liste des encodages en cours ou en attente, vous pouvez utiliser l'outil **celery** en ligne de commande.

Placez-vous donc dans l'environnement virtuel django et lancez les commandes suivantes, en remplacant <ID> par le thread Redis voulu (5 pour les encodages, 6 pour xAPI par exemple) :

Tâches en cours

```
# Liste des tâches en cours :

(django_pod3) pod@pod-transcodage:/$ celery --broker=redis://redis:6379/<ID> inspect active
```

Tâches en attente

```
# Liste des tâches en attente :

(django_pod3) pod@pod-transcodage:/$ celery --broker=redis://redis:6379/<ID> inspect reserved
```