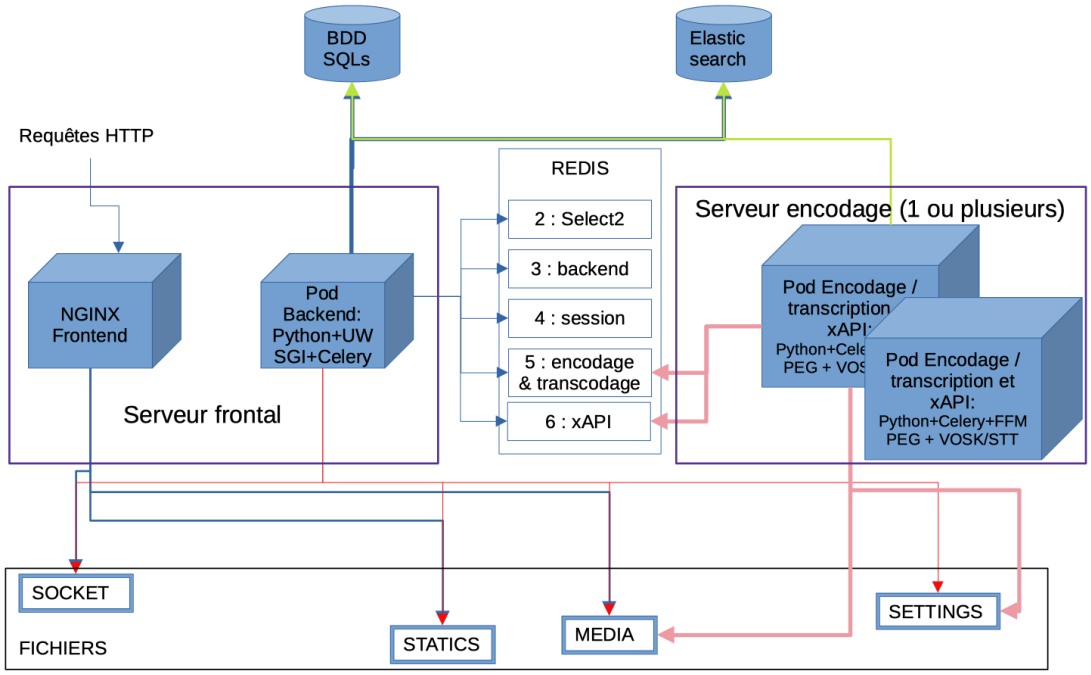
Déporter l'encodage sur un ou plusieurs serveurs en Pod V3

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

Schéma de principe de fonctionnement :



Rabbitmq était utilisé comme gestion de file d'attente, il est remplacé maintenant par Redis

Pré-requis :

- Il faut que votre répertoire `~/django_projects/podv3/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
- Il faut utiliser sur les serveurs d'encodage Elasticsearch installé sur le serveur frontal

Installation sur le serveur frontal :

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

```
(django_pod) pod@pod:~/django_projects/podv3$ sudo apt-get purge ffmpeg ffmpegthumbnailer imagemagick
```

On peut utiliser le même Redis que pour la gestion du cache du frontal. Toutefois, si vous souhaitez continuer à utiliser RabbitMQ, il faut l'installer.

```
(django_pod) pod@pod:~/django_projects/podv3$ sudo apt-get install rabbitmq-server
(django_pod) pod@pod:~/django_projects/podv3$ sudo rabbitmqctl add_user pod *mdp*
(django_pod) pod@pod:~/django_projects/podv3$ sudo rabbitmqctl set_user_tags pod administrator
(django_pod) pod@pod:~/django_projects/podv3$ sudo rabbitmqctl set_user_tags guest
(django_pod) pod@pod:~/django_projects/podv3$ sudo rabbitmqctl add_vhost rabbitpod
(django_pod) pod@pod:~/django_projects/podv3$ sudo rabbitmqctl set_permissions -p rabbitpod pod ".*" ".*" ".*"
```

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

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

```
# Configuration Celery sur le frontal

CELERY_TO_ENCODE = True # Active encode
# Si RabbitMQ
CELERY_BROKER_URL = "amqp://pod:mdp@localhost/rabbitpod" # Define a broker
# Si Redis
CELERY_BROKER_URL = "redis://redis:6379/5" # on utilise la db numéro 5
```

Installation sur le serveur d'encodage :

Il faut installer Pod sans réinitialiser la base et sans nginx/uwsgi/Elasticsearch. Vous pouvez suivre la doc Installation de la plateforme Pod.

Rajouter la configuration de tout ça dans le fichier de configuration

Il faut maintenant dire au serveur d'encodage :

- Que l'on souhaite utiliser CELERY
- Donner l'adresse du serveur front de CELERY BROKER
- De connecter la base de données commune
- De connecter l'ElasticSearch commun

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

```
CELERY_TO_ENCODE = True # Active encode
# si RabbitMQ : CELERY_BROKER_URL = "amqp://pod:mdp@ip.serveur.frontal/rabbitpod" # Definit le message broker.
# si REDIS : CELERY_BROKER_URL = "redis://redis:6379/5" # on utilise la db numéro 5
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
```

Vérifiez que votre base de données, ElasticSearch accepte les communications entrantes avec vos serveurs d'encodage (bind)

Activer Celery sur le serveur d'encodage

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

```
(django_pod) pod@pod-enc:~/django_projects/podv3$ sudo vim /etc/init.d/celeryd
(django_pod) pod@pod-enc:~/django_projects/podv3$ sudo chmod u+x /etc/init.d/celeryd
```

Créer le fichier default associé :

```
(django_pod) pod@pod-enc:/usr/local/django_projects/podv3$ 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/podv3"                        # 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/podv3$ sudo /etc/init.d/celeryd start
```

Pour vérifier si Celery fonctionne bien :

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