Mise en place du Direct Live

Installation d'une nouvelle VM

Nous conseillons de mettre en place le direct sur une autre VM que Pod.

Le live se base sur le module RTMP de Nginx.

Pré-requis

- Installation : Debian 9.4 64 bits

Pour installer nginx en version 1.14, il faut d'abord ajouter les **backports** :

Se placer en tant que root (sudo -s)

```
$> vim /etc/apt/sources.list
```

Ajouter la ligne : *deb http://ftp.debian.org/debian stretch-backports main*

Puis faire une mise à jour :

```
$> apt update
```

Installation de nginx

```
$> apt-get -t stretch-backports install nginx
```

Ensuite, il faut installer le module nginx-rtmp :

```
$> apt-get install libnginx-mod-rtmp
```

ffmpeg

Pour le multibitrate, il faut installer ffmpeg qui encode en temps réel le flux vidéo :

```
$> aptitude install ffmpeg
```

Pour vérifier que tout s'est bien passé, il faut lister le répertoire modules enabled de nginx :

```
$> ls -l /etc/nginx/modules-enabled/
```

Vous devez voir mod-rtmp.conf :

```
total 16

[...]

lrwxrwxrwx 1 root root 48 oct.  17 12:59 50-mod-rtmp.conf -> /usr/share/nginx/modules-available/mod-rtmp.conf

[...]
```

Ensuite, il faut ajouter l'instruction include rtmp dans le nginx.conf et créer le snippets correspondant :

```
$> vim /etc/nginx/nginx.conf
[...]
include /etc/nginx/snippets/rtmp.conf;
[...]
```

Il faut donc ensuite créer le snippet **RTMP** :

```
$> vim /etc/nginx/snippets/rtmp.conf
```

Fichier rtmp.conf originel

Ci-dessous le fichier de configuration originel qui utilise **3 encodages** et n'est pas spécialement optimisé vis-à-vis de la latence (il faut compter entre **15 et 30 secondes de latence** avec cette configuration) :

Fichier /etc/nginx/snippets/rtmp.conf

```
rtmp {
    server {
        listen 1935; # port rtmp par defaut
        chunk_size 4096; # taille des paquets transmis/découpé

         application live { # nom de l'application
            live on;
            #meta copy;
            #record off;

            # publish only from localhost
            allow publish 127.0.0.1; # seulement publier en local
            allow publish all; #tout le monde peut publier
            allow play all; # certaine adresse IP on le droit de lire
            deny publish all; # mettre un deny à la fin pour securiser

            exec ffmpeg -i rtmp://localhost/$app/$name
                -c:v libx264 -preset veryfast -b:v 256k -maxrate 256k -bufsize 512k -vf "scale=480:-2,
format=yuv420p" -g 60 -c:a aac -b:a 64k -ar 44100 -f flv rtmp://localhost/show/$name_low
                -c:v libx264 -preset veryfast -b:v 512k -maxrate 512k -bufsize 1024k -vf "scale=720:-2,
format=yuv420p" -g 60 -c:a aac -b:a 96k -ar 44100 -f flv rtmp://localhost/show/$name_mid
                -c:v libx264 -preset veryfast -b:v 1024k -maxrate 1024k -bufsize 2048k -vf "scale=1280:-2,
format=yuv420p" -g 60 -c:a aac -b:a 128k -ar 44100 -f flv rtmp://localhost/show/$name_high
            >/tmp/ffmpeg.log 2>&1 ;

            exec_publish curl --request PATCH "https://pod.univ.fr/rest/broadcasters/$name/" --header "Content-
Type: application/json" --header "Accept: application/json" --user CHANGE_USERNAME:CHANGE-THIS-STATUS-PASSWORD
--data "{\"status\":true}";

            exec_publish_done curl --request PATCH "https://pod.univ.fr/rest/broadcasters/$name/" --header
"Content-Type: application/json" --header "Accept: application/json" --user CHANGE_USERNAME:CHANGE-THIS-STATUS-
PASSWORD --data "{\"status\":false}";
        }

        # This application is for splitting the stream into HLS fragments
        application show {
            live on; # Allows live input from above
            meta copy;
            record off;

            hls on; # activation du hls
            hls_path /dev/shm/hls; #chemin des fragment ts mettre dans /dev/shm pour eviter de faire trop
travailler le disque
            hls_nested on; # cree un sous repertoire par stream envoye
            hls_fragment 2s; #taille des fragments

            # Instruct clients to adjust resolution according to bandwidth
            hls_variant _low BANDWIDTH=320000; # Low bitrate, sub-SD resolution
            hls_variant _high BANDWIDTH=640000; # High bitrate, higher-than-SD resolution
            hls_variant _src BANDWIDTH=1200000; # Source bitrate, source resolution
        }
    }
}
```

Cette configuration est largement éprouvée dans le temps, mais fait consommer plus de ressources (3 encodages) avec une certaine latence au final.

Fichier rtmp.conf optimisé pour réduire la latence

Ci-dessous le même fichier de configuration qui utilise **2 encodages** et qui est spécialement optimisé vis-à-vis de la latence (il faut compter **moins de 10 secondes de latence** avec cette configuration) .

Ce fichier reprend des éléments de configuration présenté par Ludovic Bouguerra de la société Kalyzée lors de son Webinaire "Mise en place d'une infrastructure de live et réduction de la latence avec Pod" du 23 septembre 2022.

Les éléments de configuration modifiés sont les suivants :

- 2 encodages réalisés
- le preset en ultrafast
- l'option tune zerolatency
- le nombre de keyframes (- g) positionné à 60 (ou 50)

Ce qui donne :

Fichier /etc/nginx/snippets/rtmp.conf

```
rtmp {
    server {
        listen 1935; # port rtmp par defaut
        chunk_size 4000; # taille des paquets transmis/découpé

         application live { # nom de l'application
            live on;
            #meta copy;
            #record off;

            # publish only from localhost
            allow publish 127.0.0.1; # seulement publier en local
            allow publish all; #tout le monde peut publier
            allow play all; # certaine adresse IP on le droit de lire
            deny publish all; # mettre un deny à la fin pour securiser

            exec ffmpeg -i rtmp://localhost/$app/$name
                -c:v libx264 -preset ultrafast -b:v 512k -tune zerolatency -maxrate 512k -bufsize 1024k -vf
"scale=480:-2,format=yuv420p" -g 60 -c:a aac -b:a 96k -ar 44100 -f flv rtmp://localhost/show/$name_low
                -c:v libx264 -preset ultrafast -b:v 1.5M -tune zerolatency -maxrate 1.5M -bufsize 3M -vf
"scale=1280:-2,format=yuv420p" -g 60 -c:a aac -b:a 128k -ar 44100 -f flv rtmp://localhost/show/$name_high
            >/tmp/ffmpeg.log 2>&1 ;

            exec_publish curl --request PATCH "https://pod.univ.fr/rest/broadcasters/$name/" --header "Content-
Type: application/json" --header "Accept: application/json" --user CHANGE_USERNAME:CHANGE-THIS-STATUS-PASSWORD
--data "{\"status\":true}";

            exec_publish_done curl --request PATCH "https://pod.univ.fr/rest/broadcasters/$name/" --header
"Content-Type: application/json" --header "Accept: application/json" --user CHANGE_USERNAME:CHANGE-THIS-STATUS-
PASSWORD --data "{\"status\":false}";
        }

        # This application is for splitting the stream into HLS fragments
        application show {
            live on; # Allows live input from above
            meta copy;
            record off;

            hls on; # activation du hls
            hls_path /dev/shm/hls; #chemin des fragment ts mettre dans /dev/shm pour eviter de faire trop
travailler le disque
            hls_nested on; # cree un sous repertoire par stream envoye
            hls_fragment 2s; #taille des fragments
                        # Mise en commentaire suite a des problemes en lien avec la publication RTMP de SMP 351
                        # hls_max_fragment 3s;
            # hls_playlist_length 10s;

            # Instruct clients to adjust resolution according to bandwidth
            hls_variant _low BANDWIDTH=512000; # Low/Mid bitrate, about sub-SD resolution
            hls_variant _high BANDWIDTH=1500000; # Source bitrate, source resolution
        }
    }
}
```

Cette configuration est plus récente, mais fait consommer moins de ressources (2 encodages), avec une latence réduite au final.

Suite à la mise en place en production, il s'est avéré que, lors de la publication RTMP de la part de SMP 351, cette configuration pouvait provoquer une erreur du type "**force fragment split**".

Finalement, en commentant les paramètres suivants, ce problème n'est plus réapparu :

hls_max_fragment 3s;
hls_playlist_length 10s;

Vous pouvez voir toutes les directives de ce module à cette adresse : https://github.com/arut/nginx-rtmp-module/wiki/Directives

HTTP

Il faut enfin déclarer la route hls pour lire les vidéos :

```
$> vim /etc/nginx/sites-enabled/default
```

Fichier /etc/nginx/sites-enabled/default

```
server {
  listen 80 default_server;
  root /var/www/html;
  index index.html index.htm index.nginx-debian.html;
  server_name _;
  location / {
      try_files $uri $uri/ =404;
   }
 # path to HLS application service
        location /hls {
            types {
                application/vnd.apple.mpegurl m3u8;
                video/mp2t ts;
            }
            alias /dev/shm/hls;
            add_header Cache-Control no-cache;
            add_header 'Access-Control-Allow-Origin' 'https://pod.univ-machin.fr';  #  <--- surtout pas "*" :
risque d'injection de code !!!
        }
  add_header 'Access-Control-Allow-Origin' 'https://pod.univ-machin.fr'; # Hotfix pour diffusion depuis un
autre serveur  <--- surtout pas "*" : risque d'injection de code !!!
}
```

⚠️ **Si votre frontal POD est en HTTPS il faut configurer Nginx sur le serveur live pour servir du HTTPS et non du HTTP**

Sur votre instance de POD

Il faut commencer par activer l'application live en ajoutant "live" dans THIRD_PARTY_APPS dans le fichier settings_local.py

```
...
THIRD_PARTY_APPS = ["xxx","xxx","live"]
...
```

Ensuite il faut se rendre dans l'administration de Pod, créer un bâtiment puis un diffuseur rattaché à ce bâtiment en précisant l'url de lecture du flux de direct.

Mise en place du comptage des spectateurs (A partir de 2.7.0 et supérieur)

Pod inclut une fonction qui permet de compter les spectateurs sur un direct en temps réel.

Cette fonctionnalité doit être mise en place de la façon suivante :

- Ajouter une tâche CRON périodique (chaque minute recommandé, mais il est possible d'augmenter ou de diminuer le temps et régler les settings sur pod en conséquence (voir les points suivants ...) :

  Commande du CRON

  bash -c 'export WORKON_HOME=/home/%userpod%/.virtualenvs; export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3.6; cd /home/%userpod%/django_projects/podv2; source /usr/local/bin/virtualenvwrapper.sh; workon django_pod; python manage.py live_viewcounter'

  ATTENTION : Il est possible que vous ayez à modifier cette commande suivant vos chemins d'installation de POD.
- Réglez le setting VIEWERS_ONLY_FOR_STAFF, celui-ci prend les valeurs True ou False et permet de définir si la liste des spectateurs (disponible en appuyant sur l'oeil) est accessible à tous ou uniquement aux membres du personnel.
- Réglez le setting HEARTBEAT_DELAY, celui ci permet de définir le délai entre lequel un client va remonter son "signal" au serveur. Il est par défaut fixé à 45 secondes.
- Réglez le setting VIEW_EXPIRATION_DELAY, il permet de définir le délai maximum pour lequel une vue va encore considérée comme présente sur le direct.

  Explication !

  Afin de mieux comprendre ces notions de délais voici un exemple avec les paramètres par défaut : Le client (le terminal sur lequel l'utilisateur est en train de regarder le direct) va remonter un signal au serveur pod de manière périodique toutes les **45** (HEARTBEAT_DELAY) secondes. Chaque **minute** (temps du CRON), une tâche vérifie la validité de chaque spectateur pour déterminer si il est encore présent sur le direct. Pour cela, elle élimine tout les spectateurs qui n'ont pas remonté de signal depuis **60** (VIEW_EXPIRATION_DELAY) secondes.

  A noter : Il est laissé la possibilité de modifier les délais pour pallier à d'éventuelles pertes de performances à cause de la remontée des vues. Si vous rencontrez des difficultés à ce niveau, n'hésitez pas à doubler les délais. Le comptage sera moins précis en temps réel mais vous gagnerez en nombre de requêtes.

Diffusion d'un flux video en direct (avec OBS)

Avec **OBS**, dans les paramètres, onglet Flux, je précise ces données :
URL : rtmp://serveur.univ.fr/live
Clé de stream : nico

Dans **Pod**, dans les paramètres de mon diffuseur, dans le champ URL, je vais préciser ceci : **http://serveur.univ.fr/hls/nico.m3u8**

Il faut que le titre court soit le même que le flux/clé de stream (ici **nico**) de manière à ce que le status du live puisse être modifié par l'appel REST (`exec_publish curl ...`)

Nous venons donc de créer un flux diffusé en direct accessible en HTML5, multibitrate et adaptatif, voici ce que contient le fichier nico.m3u8 :

Fichier nico.m3u8

```
#EXTM3U

#EXT-X-VERSION:3

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=320000

nico_low/index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=640000

nico_high/index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1200000

nico_src/index.m3u8
```