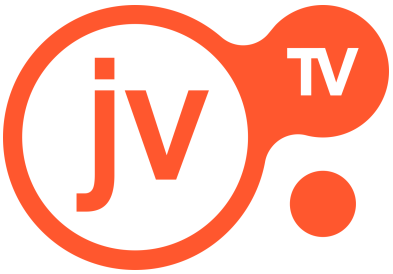

Twitch : Mise en place d'un serveur de *"Restream"*

Guide d'installation

Auteur :
Florian GRANTE

Superviseur :
David LAROSE

1er août 2017

Table des matières

Introduction

L'objectif de ce dossier est de mettre en place un serveur permettant de switch entre différents stream sans couper Twitch.

Actuellement, si un streamer veux reprendre la diffusion sur Twitch, la diffusion doit être coupé pour qu'il puisse reprendre le flux.

L'objectif est donc de créer un serveur relai pour effectuer le schéma suivant :

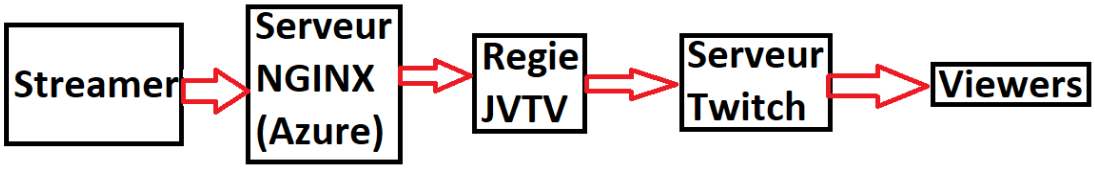

FIGURE 1 – *RIP PAINT 1985-2017*

De cette façon, lorsqu'il y a un changement de streamer, seul le lien N°1 (celui qu'on retient !) est coupé.
Ce dossier va donc expliquer comment créer de A à Z un tel serveur sur le Cloud *Microsoft Azure*.

Chapitre 1

Création et configuration de la VM

Pour ce serveur nous allons utiliser une VM RedHat Enterprise License (RHEL). Rendez vous sur votre plate forme *Azure* puis créer la VM RHEL avec les paramètres souhaités (Utilisateur / mot de passe,...).

1.1 Configuration Azure

Dans un premier temps, il est nécessaire d'ouvrir les ports 1935 et 80 en TCP sur le portail *Microsoft Azure*.

1. Se rendre sur la page principale de la VM sur *Azure*
2. Cliquer sur l'onglet "Endpoint" ("Point de terminaison" en français.)
3. Une liste va alors s'ouvrir, cliquer alors sur "Add" en haut.
4. Indiquez un nom pour votre exception (choix arbitraire).
5. Renseigner le port 80 (entrant et sortant).
6. Choisir le protocole "TCP".
7. Cliquer alors sur "OK".

Une fois que vous aurez la notification vous assurant que cela a bien été prit en compte, recommencer une nouvelle fois la démarche mais cette fois ci pour le port 1935. Il s'agit du port par lequel va transiter notre flux.

Aucune autre étape est nécessaire sur le portail *Azure* si ce n'est récupérer l'adresse IP de votre VM pour faire une connexion en ssh.

1.2 Définition d'un mot de passe root

Microsoft ne nous fournis pas le mot de passe root qui est définie par défaut sur la VM. Nous allons donc passer par l'utilisateur que vous avez créer sur *Azure* pour le modifier.

1. Connecter vous en SSH avec votre utilisateur :

```
1                    ssh utilisateur@ip
```

2. Entrer ensuite la commande suivante pour passer en mode root :

```
1                    sudo su
```

3. Renseigner votre mot de passe utilisateur.
4. Taper ensuite la commande :

```
1                    passwd
```

5. Il vous sera alors demandé de renseigner votre nouveau mot de passe root deux fois.

Maintenant que vous connaissez le mot de passe root, vous pouvez vous connecter en ssh en remplaçant votre utilisateur par "root".

1.3 Installation de NGINX et configuration.

1.3.1 Installation

Avant d'installer NGINX a proprement parler, il est nécessaire d'ajouter d'autre paquets avant.

1. Connecter vous en ssh en mode root.
2. Commencer par entrer la commande :

```
yum group install 'Development Tools'
```

3. Puis enchainer avec la commande :

```
yum install pcre-devel
```

4. Enfin, entrer la commande :

```
yum install openssl-devel
```

Ces quelques paquets risque de mettre un peu de temps à s'installer. Nous allons maintenant pouvoir installer NGINX a proprement parler. Il faut entrer ces commandes une à une :

```
wget http://nginx.org/download/nginx-1.6.2.tar.gz

wget https://github.com/arut/nginx-rtmp-module/archive/master.zip

tar -zxvf nginx-1.6.2.tar.gz

yum install unzip

unzip master.zip

cd nginx-1.6.2/

./configure --with-http_ssl_module --add-module=../nginx-rtmp-module-master

make

make install

rm -r nginx-1.6.2/ nginx-rtmp-module-master/
```

Nginx est maintenant installé. Avant de pouvoir le tester, il faut désactiver le firewall de RHEL.

```
systemctl disable firewalld
```

Puis lancer le serveur NGINX :

```
/usr/local/nginx/sbin/nginx
```

INFORMATIONS : Pour arrêter le serveur, taper la même commande suivi de "-s stop".

Ouvrez un navigateur WEB et taper l'@IP du serveur dans votre barre d'URL. Vous devriez voir apparaître le résultat suivant :

FIGURE 1.1 – *Serveur NGINX*

Félicitation, votre serveur NGINX est maintenant opérationnel.

1.3.2 Configuration du serveur

Nous allons maintenant configurer notre serveur pour qu'il puisse rediriger notre flux RTMP. Ouvrer alors le fichier de configuration en entrant la commande :

```
sudo nano /usr/local/nginx/conf/nginx.conf
```

ATTENTION : Nano n'est pas le meilleur éditeur de texte (je dirais même que c'est le pire, mais il est de base donc on s'en contentera pour le peu qu'il y a à faire). Pour naviguer dans le fichier, utiliser les flèches directionnelles.

Modifier la première ligne en changeant "nobody" par le nom d'utilisateur que vous avez renseigné sur *Azure*.

Puis aller à la fin du ficher et rajouter les lignes suivantes :
Lire sur GitHub

```

rtmp {
        server {
                listen 1935;
                chunk_size 128;

                application live {
                        live on;
                        exec_push omxplayer --live rtmp://localhost:1935/live
/stream;
                        record off;
                }
        }
}
```

NOTE : Ici, le mot "live" après "application" et le mot "stream" sont arbitraire et définissent le lien où envoyer le flux. On peut considérer "stream" comme la clé de stream. Libre à vous de changer ces deux termes dans le fichier de config pour modifier le lien à votre convenance.

Il nous reste maintenant à faire en sorte que notre serveur NGINX se lance tout seul au démarrage du serveur. Cela nous évitera d'y retourner sans cesse si des fois il devait redémarrer.

Pour cela il va falloir créer un script avec différentes options pour que l'OS soit en mesure de le lancer.

Comme le fichier est un petit peu long, je vais vous mettre un lien pour télécharger un fichier texte contenant le code. Il ne vous restera plus qu'à le copier coller sur nano dans le git bash.
Voici donc le code du script : .txt sur GitHub

```
1
2  #!/bin/sh
3  #
4  # nginx - this script starts and stops the nginx daemin
5  #
6  # chkconfig:   - 85 15
7  # description:  Nginx is an HTTP(S) server, HTTP(S) reverse \
8  #               proxy and IMAP/POP3 proxy server
9  # processname: nginx
10 # config:      /usr/local/nginx/conf/nginx.conf
11 # pidfile:     /usr/local/nginx/logs/nginx.pid
12
13 # Source function library.
14 . /etc/rc.d/init.d/functions
15
16 # Source networking configuration.
17 . /etc/sysconfig/network
18
19 # Check that networking is up.
20 [ "$NETWORKING" = "no" ] && exit 0
21
22 nginx="/usr/local/nginx/sbin/nginx"
23 prog=$(basename $nginx)
24
25 NGINX_CONF_FILE="/usr/local/nginx/conf/nginx.conf"
26
27 lockfile=/var/lock/subsys/nginx
28
29 start() {
30     [ -x $nginx ] || exit 5
31     [ -f $NGINX_CONF_FILE ] || exit 6
32     echo -n $"Starting $prog: "
33     daemon $nginx -c $NGINX_CONF_FILE
34     retval=$?
35     echo
36     [ $retval -eq 0 ] && touch $lockfile
37     return $retval
38 }
39
40 stop() {
41     echo -n $"Stopping $prog: "
42     killproc $prog -QUIT
43     retval=$?
44     echo
45     [ $retval -eq 0 ] && rm -f $lockfile
46     return $retval
47 }
48
49 restart() {
50     configtest || return $?
51     stop
52     start
53 }
54
55 reload() {
56     configtest || return $?
57     echo -n $"Reloading $prog: "
58     killproc $nginx -HUP
59     RETVAL=$?
60     echo
```

```
}

force_reload() {
    restart
}

configtest() {
  $nginx -t -c $NGINX_CONF_FILE
}

rh_status() {
    status $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}

case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart|configtest)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
            ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart
|reload|force-reload|configtest}"
        exit 2
esac
```

Ce code doit être placé dans un fichier dans */etc/init.d/*. Il doit de plus porter le même nom que notre script de base, *nginx*.

Ainsi, taper la commande :

```
        nano /etc/init.d/nginx
```

Une fois le fichier complété et sauvegardé (Ctrl + O puis Ctrl + X pour enregistrer puis quitter nano), Rendre le script exécutable avec la commande :

```
        chmod +x /etc/init.d/nginx
```

Enfin, ajouter le script à la list des script à lancer avec la commande :

```
1        /sbin/chkconfig nginx on
```

Et voilà, faire un reboot de la machine et vérifier que votre serveur est bien ON !

Chapitre 2

Envoyer et lire un flux sur le serveur

2.1 envoyer un flux

Pour rester dans des domaines maitrisé, le streamer devra envoyer le flux via *Xsplit*.
Pour cela, rien de plus simple :

1. Cliquer sur *"Ajouter source"*.
2. Choisir *"Custom RTMP server"*.

Une fenêtre s'ouvre alors :

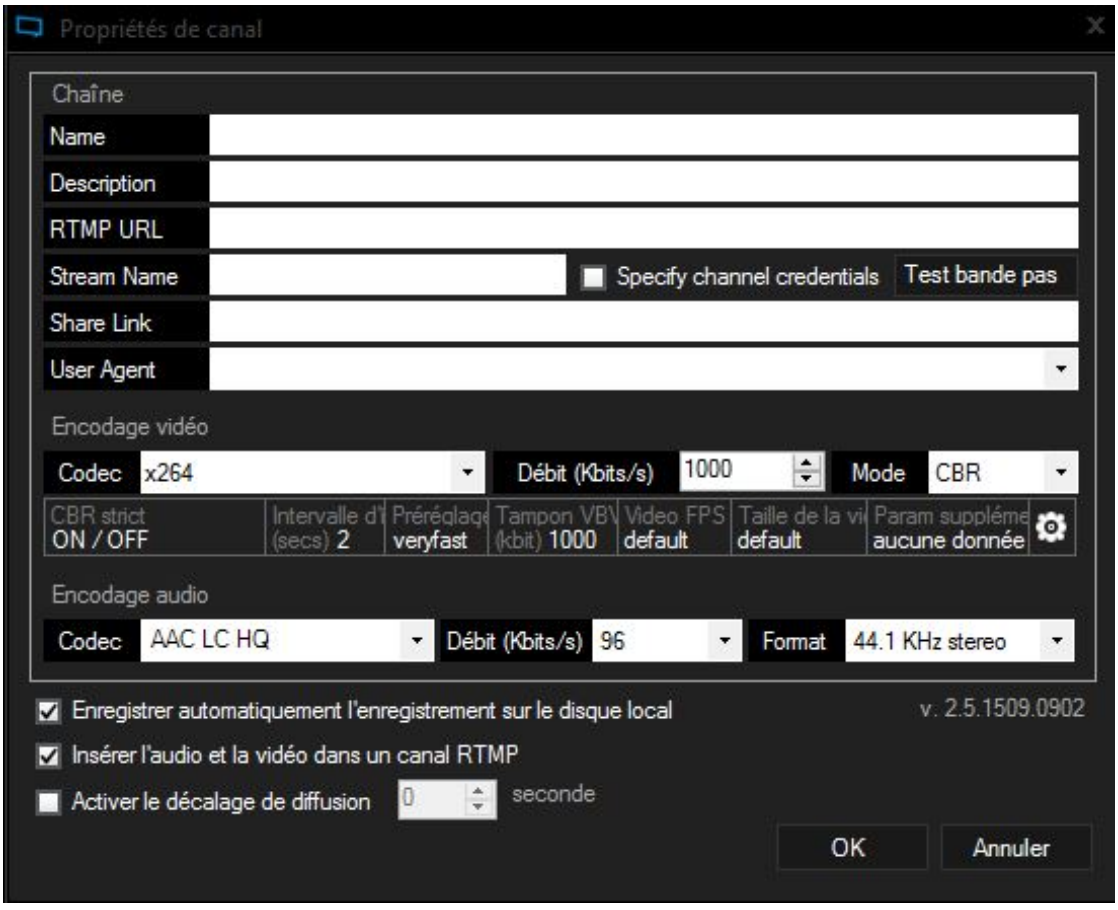

FIGURE 2.1 – *Interface custom RTMP xsplit*

3. RTMP URL : *rtmp ://xxx.xxx.xxx.xxx :1935/live*. Y mettre l'adresse IP de la machine.
4. Stream Name : *stream*.
5. Décocher "Specify channel credentials". (à coté de Stream Name).

NOTE : Penser à remplacer live et stream si vous l'avez fait dans le fichier de config.
Le reste de la configuration n'importe pas et dépend de s contraintes propre au streamer.

Et voilà, tout est prêt pour envoyer le flux comme un streamer pourrait le faire habituellement.
A noter le le délai est faible (de l'ordre de 1 à 3 secondes) et impacte peut par rapport au décalage déjà imposé par Twitch.

2.2 Lire le flux

Pour lire le flux depuis une régie, spécifique, se renseigner auprès du fabriquant et de sa documentation.

Pour un PC classique, il est possible d'ajouter une source RTMP sur Xsplit.

Si vous voulez seulement lire la source sans forcément la diffuser, il suffit d'ouvrir VLC, puis le raccourcis *Ctrl + n* et d'y enter le lien du flux : *rtmp ://xxx.xxx.xxx.xxx :1935/live/stream.*

Vous disposez désormais de toute les informations nécessaire pour effectuer un serveur de redirection de flux. Enjoy!