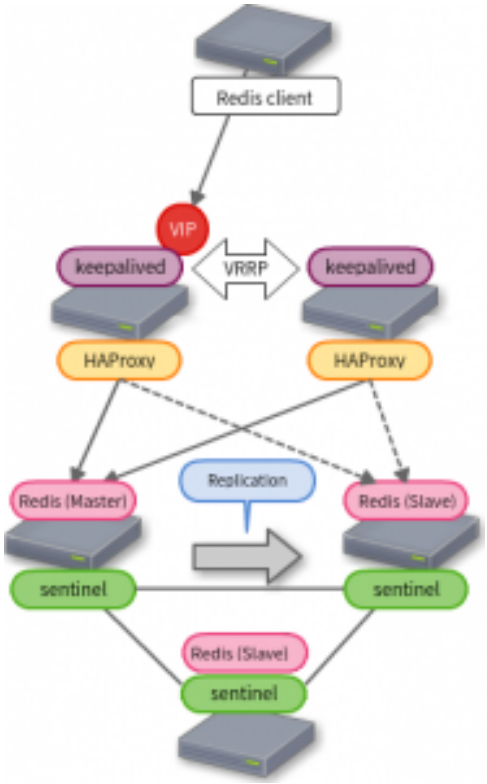
Dans le cadre d'un projet de centralisation des logs avec Ossec + LogStash + ElasticSearch + Kibana, nous avons eu recours à Redis pour faire le tampon entre Ossec et Logstash.
Des contraintes de disponibilité m'ont conduit à mettre en place une plateforme Redis hautement disponible et tolérante….la routine habituelle ;-).
Le schéma de l'architecture mise en place ressemble à ça (merci 1Q77 pour l'inspiration) :



Contrairement à Yteraoka, j'ai opté pour 3 serveurs Redis (1 maitre, 2 esclaves).
En cas de défaillance du Redis Maitre, Sentinel (redis-sentinel) s'occupe de promouvoir un des esclaves en maitre.
Le fidèle Ha-proxy redirigera sur le flux sur le nouveau maitre.
Tandis que KeepAlived s'occupe de présenter toujours la même (V)IP aux clients Redis.
Voici le plan d'adressage des machines et leur rôle :

Nom	IP	Rôle
VIP	192.168.0.1	VIP
haproxy-01	192.168.0.2	haproxy + keepalied
haproxy-02	192.168.0.3	haproxy + keepalied
redis-01	192.168.0.4	Redis maître
redis-02	192.168.0.5	Redis esclave
redis-03	192.168.0.6	Redis esclave

Ha-Proxy
Installation

```
apt-get install software-properties-common
add-apt-repository ppa:vbernat/haproxy-1.5
```

```
apt-get install haproxy hatop
```

Configuration

Editer /etc/haproxy/haproxy.cfg

```
global
    log /dev/log    local0
    log /dev/log    local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

defaults REDIS
    mode tcp
    timeout connect  4s
    timeout server  30s
    timeout client  30s

frontend ft_redis
    bind 192.168.0.1:6379 name redis
    default_backend bk_redis

backend bk_redis
    mode tcp
    option tcplog
    option tcp-check
    tcp-check send AUTH\ 7xJtpLugAyu6hgPbuB3hX4R\r\n
    tcp-check expect string +OK
    tcp-check send PING\r\n
    tcp-check expect string +PONG
    tcp-check send info\ replication\r\n
    tcp-check expect string role:master
    tcp-check send QUIT\r\n
    tcp-check expect string +OK
    server redis01 192.168.0.4:6379 check inter 1s
    server redis02 192.168.0.5:6379 check inter 1s
    server redis03 192.168.0.6:6379 check inter 1s
```

Toute l'intelligence se retrouve dans les lignes « tcp-check ».
Ici ha-proxy va simuler une connexion cliente sur les serveurs Redis en jouant le scénario suivant :
Connexion et authentification avec envoie du mot de passe, en espérant avoir la réponse OK
Demande d'information sur la réplication. Retourne un certain nombre de ligne dont celle indiquant le rôle du serveur. Ici on s'attend à avoir « role:master ».
Déconnexion
Si a une des commandes « send » le résultat retourné correspond à celui la ligne « expect » qui suite, le serveur est considéré comme L7OK, et il prendra les flux.
Sinon L7TOUT et il n'aura pas de flux.

KeepAlived

Installation

```
apt-get install keepalived
```

Configuration

Editer /etc/keepalived/keepalived.conf.
La configuration est identique les 2 machines haproxy-*, à la « priority » près.

```
# Settings for notifications
global_defs {
    notification_email {
        hugues@lepesant.com
    }
    notification_email_from haproxy01@lepesant.com
    smtp_server 212.27.48.4
    smtp_connect_timeout 15
    router_id haproxy01
}

vrrp_sync_group SyncGroup01 {
    group {
        VI_1
    }
}

vrrp_script chk_haproxy {
    script "/usr/bin/killall -0 haproxy"
    script "/usr/sbin/service haproxy restart"
    interval 9
    timeout 3
    weight 20
    rise 2
    fall 4
}

vrrp_instance VI_1 {
    interface eth0
    nopreempt
    virtual_router_id 51
    priority 101          # 101 on master, 100 on backup
    advert_int 5

    virtual_ipaddress {
        192.168.0.1/24 dev eth0
    }

    track_script {
        chk_haproxy
    }

    smtp_alert
}
```

Pour tester :

```
ip address show
```

Redis

Installation

Toutes les machines sont animées par des Ubuntu 14.04.
Les commandes sont exécutées après « sudo -i ».

```
  apt-get install software-properties-common
  add-apt-repository ppa:rwky/redis
  apt-get update
  apt-get install redis-server
```

Configuration

Histoire de ne plus avoir d'alerte dans les log à propos de « Transparent Hugepage »

```
  echo never > /sys/kernel/mm/transparent_hugepage/enabled
  vim /etc/rc.local
```

Ajouter « echo never > /sys/kernel/mm/transparent_hugepage/enabled » dans /etc/rc.local, avant le « exit ».
Par habitude je ne modifie pas les fichiers de configuration si je peux le faire par un fichier dans un répertoire « conf.d/ » de l'application.
Donc :

```
vim /etc/redis/conf.d/local.conf
```

Avec les lignes suivantes :

Redis maître

Sur le maître. Enfin au démarrage, et tant qu'il ne rencontre pas de défaillance c'est le maître.

```
#slaveof 192.168.0.5 6379
masterauth 7xJtpLugAyu6hgPbuB3hX4R
requirepass 7xJtpLugAyu6hgPbuB3hX4R
```

La réplication est un peu sécurisée par la demande d'un mot de passe (masterauth).
Comme à n'importe quel moment le maître peut devenir esclave, il doit connaitre le mot de passe pour se connecter au maître (requirepass).

Redis esclave

Sur le 2 esclaves.

```
slaveof 192.168.0.4 6379
masterauth 7xJtpLugAyu6hgPbuB3hX4R
requirepass 7xJtpLugAyu6hgPbuB3hX4R
```

Sentinel

Création d'un script d'init

C'est balot y'en a pas

```
vim /etc/init.d/redis-sentinel
chmod +x /etc/init.d/redis-sentinel
update-rc.d redis-sentinel defaults
```

En voilà le contenu :

```
#!/bin/sh
```

```
### BEGIN INIT INFO
# Provides:          redis-sentinel
# Required-Start:
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start daemon at boot time
# Description:       Enable service provided by daemon.
### END INIT INFO

NAME="redis-sentinel"
DAEMON="/usr/bin/redis-sentinel"

. /lib/lsb/init-functions
[ -f /etc/default/rcS ] && . /etc/default/rcS

test -x $DAEMON || exit 0

case "$1" in
    stop)
        log_begin_msg "Stoping Redis Sentinel..." "redis-sentinel"
        killall -9 redis-sentinel &> /dev/null
        log_end_msg $?
        ;;
    start)
        log_begin_msg "Starting Redis Sentinel..." "redis-sentinel"
        nohup $DAEMON /etc/redis/sentinel.conf >>
/var/log/redis/sentinel.log &> /dev/null &
        log_end_msg 0
        ;;
        restart)
                $0 stop
                sleep 2
                $0 start
                ;;
        status)
                status_of_proc $DAEMON redis-sentinel
        ;;
    *)
        log_failure_msg "Usage: $0 <stop|start|restart|status>"
        exit 1
        ;;
esac

exit 0
```

Configuration

Création du fichier « /etc/redis/sentinel.conf » avec le contenu suivant :

```
port 26379
daemonize yes
logfile "/var/log/redis/sentinel.log"
```

```
pidfile "/var/run/redis/redis.pid"

sentinel monitor mymaster 192.168.0.4 6379 2
sentinel auth-pass mymaster 7xJtpLugAyu6hgPbuB3hX4R
sentinel down-after-milliseconds mymaster 5000
sentinel failover-timeout mymaster 10000
```

On dit à sentinel de surveiller le noeud « mymaster » qui à l'IP 192.168.0.4, sur le port 6379, et qui nécessite un quorum de 2 pour l'élection d'un master.
Sentinel se charge de détecter à la fois les autres serveurs sentinels et redis.
Il mettra à jour lui-même son fichier de configuration.

Start

```
service redis-server start
service redis-sentinel start
```

Pour monitorer tout ça

Keepalived

```
ip address show
```

Ha-Proxy

```
hatop -s /var/run/haproxy/admin.sock
```

Redis

```
redis-cli -a 7xJtpLugAyu6hgPbuB3hX4R monitor
redis-cli -a 7xJtpLugAyu6hgPbuB3hX4R info replication
```

Sentinel

```
tail -f /var/log/redis/sentinal le viagra en vente libre.log
```

Et avec le client Redis.

```
redis-cli -p 26379 -a 7xJtpLugAyu6hgPbuB3hX4R sentinel masters
redis-cli -p 26379 -a 7xJtpLugAyu6hgPbuB3hX4R sentinel master mymaster
redis-cli -p 26379 -a 7xJtpLugAyu6hgPbuB3hX4R sentinel slaves mymaster
redis-cli -p 26379 -a 7xJtpLugAyu6hgPbuB3hX4R sentinel sentinels
mymaster
```

Si le maître plante

Dans ce cas :
Sentinel va marquer le maître comme « down »
Sentinel va promouvoir un esclave en maître, choisi parmi les esclaves
Ha-Proxy va détecter que le maître a changé et va basculer les connexions sur lui
Si l'ancien maître revient à la vie, Ha-Proxy le détactera à nouveau comme « L7OK », mais n'enverra pas de connexion dessus car il n'y a pas de répartition de charge dessus.
Il nous faudra modifier ça configuration en précisant le nouveau maître.

Références

https://blog.1q77.com/2015/02/redis-ha/
http://engineering.bloomreach.com/the-evolution-of-fault-tolerant-redis-cluster/
http://qiita.com/wellflat/items/8935016fdee25d4866d9
http://bencane.com/2013/11/12/installing-redis-and-setting-up-master-slave-replication/
https://support.pivotal.io/hc/en-us/articles/205309388-How-to-setup-HAProxy-and-Redis-Sentinel-for-automatic-failover-between-Redis-Master-and-Slave-servers
http://www.101tech.net/2014/08/08/highly-available-redis-cluster/

ET bien sure :
http://redis.io/topics/replication
http://redis.io/topics/sentinel