HAProxy

Installing and configuring HAProxy on CentOS 7

1. Install dependencies

```
yum install openssl-devel pcre-devel make gcc -y
```

2. Download HAproxy

Download the stable version of HAproxy, for instance to the /tmp directory

```
cd /tmp
wget http://www.haproxy.org/download/1.7/src/haproxy-1.7.2.tar.gz -O- | tar -zx
```

3. Go to the unpacked directory with sources

```
d haproxy-*
```

4. Run make

```
make TARGET=linux2628 USE_PCRE=1 USE_OPENSSL=1 USE_ZLIB=1 USE_CRYPT_H=1 USE_LIBCRYPT=1
make install
```

5. Create a user named haproxy

```
useradd haproxy
```

6. Create a directory named /var/lib/haproxy/

```
mkdir /var/lib/haproxy/
```

7. Create a .pem file from certificates imported to the WCS server

Examples of certificates from StartSSL
test.flashphoner.com.crt - certificate file
test.flashphoner.com.key - private key file
ca.pem - root certificate
sub.class2.server.ca.pem - intermediate certificate

```
cat test.flashphoner.com.crt ca.pem sub.class2.server.ca.pem test.flashphoner.com.key | tee test.flashphoner.
com.pem
```

8. Create the configuration file /etc/haproxy/haproxy.cfg with the following contents:

```
#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
chroot /var/lib/haproxy
pidfile /var/run/haproxy.pid
maxconn 4000
user haproxy
group haproxy
daemon
# turn on stats unix socket
stats socket /var/lib/haproxy/stats
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
mode http
log global
option httplog
option tcplog
option http-server-close
option redispatch
retries 3
timeout http-request 10s
timeout queue 1m
timeout connect 10s
timeout client 1m
timeout server 1m
timeout http-keep-alive 5s
timeout check 10s
maxconn 3000
http-reuse always
#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend secure
bind SET_YOUR_IP:443 ssl crt /path/to/your/certificate/cert.pem
acl is_websocket hdr(Upgrade) -i WebSocket
acl is_websocket hdr(Sec-WebSocket-Key) -m found
use_backend ws_app if is_websocket
use_backend web_app if { req.proto_http }
default_backend static
backend static
server static 127.0.0.1:8888
# websocket
backend ws_app
server app1 127.0.0.1:8080
# web content
backend web_app
server app1 127.0.0.1:8888 ssl verify none
```

In the line

```
bind SET_YOUR_IP:443 ssl crt /path/to/your/certificate/cert.pem
```

replace
- SET_YOUR_IP - to the public IP of the WCS server
- /path/to/your/certificate/cert.pem - to the .pem file created from certificates imported to the WCS server

9. Create the init file /etc/init.d/haproxy with the following contents:

```
#!/bin/bash
#
# chkconfig: - 85 15
```

```
# description: HA-Proxy is a TCP/HTTP reverse proxy which is particularly suited \
# for high availability environments.
# processname: haproxy
# config: /etc/haproxy/haproxy.cfg
# pidfile: /var/run/haproxy.pid
# Script Author: Simon Matter <simon.matter@invoca.ch>
# Version: 2004060600
# Source function library.
if [ -f /etc/init.d/functions ]; then
. /etc/init.d/functions
elif [ -f /etc/rc.d/init.d/functions ] ; then
. /etc/rc.d/init.d/functions
else
exit 0
fi
# Source networking configuration.
. /etc/sysconfig/network
# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0
# This is our service name
BASENAME=`basename $0`
if [ -L $0 ]; then
BASENAME=`find $0 -name $BASENAME -printf %l`
BASENAME=`basename $BASENAME`
fi
BIN=/usr/local/sbin/$BASENAME
CFG=/etc/$BASENAME/$BASENAME.cfg
[ -f $CFG ] || exit 1
PIDFILE=/var/run/$BASENAME.pid
LOCKFILE=/var/lock/subsys/$BASENAME
RETVAL=0
start() {
quiet_check
if [ $? -ne 0 ]; then
echo "Errors found in configuration file, check it with '$BASENAME check'."
return 1
fi
echo -n "Starting $BASENAME: "
daemon $BIN -D -f $CFG -p $PIDFILE
RETVAL=$?
echo
[ $RETVAL -eq 0 ] && touch $LOCKFILE
return $RETVAL
}
stop() {
echo -n "Shutting down $BASENAME: "
killproc $BASENAME -USR1
RETVAL=$?
echo
[ $RETVAL -eq 0 ] && rm -f $LOCKFILE
[ $RETVAL -eq 0 ] && rm -f $PIDFILE
return $RETVAL
}
restart() {
quiet_check
if [ $? -ne 0 ]; then
echo "Errors found in configuration file, check it with '$BASENAME check'."
return 1
fi
stop
start
}
reload() {
if ! [ -s $PIDFILE ]; then
return 0
fi
quiet_check
if [ $? -ne 0 ]; then
echo "Errors found in configuration file, check it with '$BASENAME check'."
return 1
fi
```

```
$BIN -D -f $CFG -p $PIDFILE -sf $(cat $PIDFILE)
}
check() {
$BIN -c -q -V -f $CFG
}
quiet_check() {
$BIN -c -q -f $CFG
}
rhstatus() {
status $BASENAME
}
condrestart() {
[ -e $LOCKFILE ] && restart || :
}
# See how we were called.
case "$1" in
start)
start
;;
stop)
stop
;;
restart)
restart
;;
reload)
reload
;;
condrestart)
condrestart
;;
status)
rhstatus
;;
check)
check
;;
*)
echo $"Usage: $BASENAME {start|stop|restart|reload|condrestart|status|check}"
exit 1
esac
exit $?
```

10. Add haproxy to autostart

```
chmod a+x /etc/init.d/haproxy
chkconfig --add haproxy
chkconfig haproxy on
```

11. Start haproxy

```
service haproxy start
```

Verifying HAProxy

1. Make sure haproxy listens to the port 443

```
netstat -antp | grep 443
```

Example of the result of executing the command:

```
tcp 0 0 192.168.1.1:443 0.0.0.0:* LISTEN 24083/haproxy
```

If the port is occupied by another service, terminate the corresponding process and restart haproxy:

```
service haproxy restart
```

2. Make sure the certificates used to create the .pem file soecified in haproxy.cfg are imported to the WCS server

You can read more about certificates for the WCS server here: Websocket SSL

3. open the WCS server control panel via HTTPS

```
https://<domain name or IP of the WCS server>:8888/dashboard.xhtml
```

4. Verify operation of the demo example with the port 443

For instance, in the Streamer demo example change the wss port to 443 and start publishing the stream.