

Contents

- [1 Introduction](#)
- [2 Loadbalancer Configuration](#)
 - ◆ [2.1 Heartbeat](#)
 - ◇ [2.1.1 authkeys](#)
 - ◇ [2.1.2 ha.cf](#)
 - ◇ [2.1.3 haresources](#)
 - ◆ [2.2 Sysctl](#)
 - ◆ [2.3 OpenSIPS](#)
 - ◆ [2.4 MySQL Cluster](#)
- [3 FreeSWITCH Configuration](#)
 - ◆ [3.1 Authentication](#)
 - ◆ [3.2 SIP Registrations](#)
 - ◆ [3.3 CDRs](#)
- [4 See Also](#)

Enterprise Deployment

This is a work in progress

Introduction

A load balancer setup can share load between multiple FS hosts according to load. A single ingress IP can be presented to the customer for the entire cluster simplifying the customer configuration (this can be combined with [DNS SRV](#) to give failover between data centres). The load balancer will measure load on each server in the cluster and send calls to the least loaded server. This ensures that the call load is spread among servers evenly, so no server

SIP signalling goes via the load balancer server, but media goes direct between the customer endpoint and the FS host in the cluster. This is similar to the bypass media functionality of FS. Since media does not go through the OpenSIPS server, the hardware requirements are far smaller that for the FS hosts in the cluster.

For the purposes of the example of this page there are 2 load balancer servers lb1 (100.100.100.101) & lb2 (100.100.100.102) and 4 FreeSWITCH servers (fs1-fs4, 100.100.100.103 - 100.100.100.106). All are on the same subnet (255.255.255.0). The virtual IP is 100.100.100.100.

Loadbalancer Configuration

For redundancy you should have two servers that act as load balancers. These will be in an active-passive configuration, i.e. all traffic goes via the active server the passive server is promoted to active if the active server fails or is taken down for maintenance. They each have their own IP so that they can be accessed independantly (e.g. via SSH for maintenance) and share a virtual IP which is assigned to the current active server. SIP traffic is accepted on the virtual IP.

It is recommended that lb1 and lb2 have multiple ways to communicate, to prevent a split brain situation where both lb1 and lb2 think the other is offline and both start listening on the virtual IP.

Heartbeat

Heartbeat is responsible for assigning the virtual IP to a server and detecting when the other server fails.

You will need to create 3 files in /etc/heartbeat: authkeys, ha.cf, haresources. These files must be identical on both lb1 and lb2.

authkeys

This file stores one or more authentication methods, and selects which to use with the auth statement. For example:

```
auth 2
1 crc
2 sha1 thisismysecretkey
3 md5 thisismysecretkey
```

The key here is used to authenticate communication between nodes in the cluster and prevents malicious servers being able to trick a node into going offline. As a result it should be kept secret and very secure. CRC should never be used in production as it provides no authentication, just error checking.

ha.cf

This is only an example. Timings can be adjusted to suit your needs. This uses a serial cable and network broadcast to check the other node but 1 or more of serial, unicast, multicast and broadcast can be used and none is required (apart from there must be at least one communication method).

```
# Logging
logfacility      local0

# Timings
keepalive 100ms
deadtime 2
warntime 1
initdead 120

# Communication between nodes - serial port
baud 19200
serial /dev/ttyS0
```

Enterprise_deployment_OpenSIPS

```
# Communication between nodes - network
udpport 694
bcast    bond0

# What nodes are in the cluster
node lb1 lb2

# Control whether a resource will automatically fail back to its "primary" node
# or remain on the current node until it fails/admin intervenes
auto_failback off

# Processes started and stopped with heartbeat.  Restarted unless they exit with rc=100
respawn hacluster /usr/lib/heartbeat/ipfail

# API Authentication
apiauth ipfail gid=haclient uid=hacluster
```

haresources

This file controls what resources are configured on the load balancer servers, this is where the virtual IP is configured. The format is:

```
preferred_hostname IPaddr2::ip_address/netmask(/interface)
```

Interface is optional, but for our example there are two NICs (eth0 and eth1) bonded to form bond0 for network redundancy.

For example:

```
lb1 IPaddr2::100.100.100.100/255.255.255.0/bond0
```

Sysctl

OpenSIPS will bind to the virtual IP. This will result in an error on the passive server which prevents OpenSIPS starting up since it will be attempting to listen on an IP that does not belong to it. This can be avoided by setting the Linux option `net.ipv4.ip_nonlocal_bind=1`.

On debian this can be achieved by creating the file `/etc/sysctl.d/linuxha.conf`:

```
# allow services to bind to the virtual ip even when this server is the passive machine
net.ipv4.ip_nonlocal_bind = 1
```

OpenSIPS

This example listens for both UDP and TCP SIP calls on the virtual IP.

The dispatcher is used distribute REGISTER requests among the FreeSWITCH hosts. It uses the OPTIONS packet to detect when a FreeSWITCH host is offline and stop sending requests to that host. Dispatcher uses weights and doesn't take server load into account; this is ok for REGISTER requests but unsuitable for actual calls.

Enterprise_deployment_OpenSIPS

The `load_balancer` module is used to distribute INVITE requests (calls). It also uses `OPTIONS` to automatically stop sending traffic to a host if it goes offline. Unlike `dispatcher`, it stores the number of channels available on a host and in memory keeps a count of how many calls are currently in progress to that host, sending the INVITE to the least loaded host.

`load_balance()` takes an algorithm as the 3rd option - either 0 (absolute) or 1 (relative). The example here uses relative, so the "least loaded" server is the one with the lowest percentage load ($\text{current_channels}/\text{max_channels}$) while absolute would send to the one with the most free channels ($\text{maximum_channels}-\text{current_channels}$).

OpenSIPS 1.7 install guide on CentOS 5.6:

```
[root@opensips1~]# yum install gcc-c++ bison lynx subversion flex
[root@opensips1~]# wget http://opensips.org/pub/opensips/1.7.0/src/opensips-1.7.0_src.tar.gz
[root@opensips1~]# tar zxvf opensips-1.7.0_src.tar.gz
[root@opensips1~]# cd opensips-1.7.0-tls
[root@opensips1~]# make all include_modules="db_mysql"
[root@opensips1~]# make include_modules="db_mysql" prefix="/usr/local" install
```

Edit `/usr/local/etc/opensips/opensipsctlrc` and uncomment the line that says `DBENGINE=MYSQL`

```
[root@opensips1~]# vi /usr/local/etc/opensips/opensipsctlrc
DBENGINE=MYSQL
```

Create the OpenSIPS database:

```
[root@opensips1~]# opensipsdbctl create
```

Create the OpenSIPS MySQL user, password and grant privileges to opensips database:

```
[root@opensips1~]# mysql -p
mysql> grant all privileges on opensips.* to opensips@localhost identified by 'opensips';
```

While still in MySQL CLI add the FreeSWITCH machines into the `load_balancer` table:

```
mysql> use opensips;
mysql> insert into load_balancer (group_id, dst_uri, resources, description) values (1,'sip:100.1
mysql> insert into load_balancer (group_id, dst_uri, resources, description) values (1,'sip:100.1
mysql> insert into load_balancer (group_id, dst_uri, resources, description) values (1,'sip:100.1
mysql> insert into load_balancer (group_id, dst_uri, resources, description) values (1,'sip:100.1
mysql> quit;
```

Copy the OpenSIPS startup script:

```
cp /usr/src/opensips-1.7.0-tls/packaging/rpm/opensips.init /etc/init.d/opensips
sed -i "s/\usr/sbin/opensips/\usr/local/sbin/opensips/g" /etc/init.d/opensips
sed -i "s/\etc/opensips/\usr/local/etc/opensips/g" /etc/init.d/opensips
sed -i "s/\etc/default/opensips/\usr/local/etc/opensips/g" /etc/init.d/opensips
sed -i "s/RUN_OPENSIPS=no/RUN_OPENSIPS=yes/g" /etc/init.d/opensips
chmod +x /etc/init.d/opensips
```

Add user 'opensips':

```
useradd -d /usr/local/etc/opensips -s /sbin/nologin opensips
```

Enterprise_deployment_OpenSIPS

Start opensips and add the FreeSWITCH machines into the dispatcher table:

```
[root@opensips1~]# service opensips start
[root@opensips1~]# opensipsctl dispatcher addgw 1 sip:100.100.100.103 0 'FS1'
[root@opensips1~]# opensipsctl dispatcher addgw 1 sip:100.100.100.104 0 'FS2'
[root@opensips1~]# opensipsctl dispatcher addgw 1 sip:100.100.100.105 0 'FS3'
[root@opensips1~]# opensipsctl dispatcher addgw 1 sip:100.100.100.106 0 'FS4'
```

Edit opensips.cfg:

```
vi /usr/local/etc/opensips/opensips.cfg
```

Copy and paste into opensips.cfg:

```
##### Global Parameters #####

debug=3
log_stderr=no
log_facility=LOG_LOCAL0

fork=yes
children=4

disable_tcp=no

dns_try_ipv6=no

auto_aliases=no

/* bind on the machine's virtual ip (note: enable sys.net.ipv4.ip_nonlocal_bind) */
listen=udp:100.100.100.100:5060
listen=tcp:100.100.100.100:5060

##### Modules Section #####

#set module path
mpath="/usr/local/lib/opensips/modules/"

loadmodule "db_mysql.so"
loadmodule "signaling.so"
loadmodule "sl.so"
loadmodule "tm.so"
loadmodule "rr.so"
loadmodule "uri.so"
loadmodule "dialog.so"
loadmodule "maxfwd.so"
loadmodule "textops.so"
loadmodule "mi_fifo.so"
loadmodule "dispatcher.so"
loadmodule "load_balancer.so"

# ----- setting module-specific parameters -----

modparam("mi_fifo", "fifo_name", "/tmp/opensips_fifo")

modparam("dialog", "db_mode", 1)
modparam("dialog", "db_url", "mysql://opensips:opensips@localhost/opensips")

modparam("rr", "enable_double_rr", 1)
modparam("rr", "append_fromtag", 1)
```

Enterprise_deployment_OpenSIPS

```
modparam("tm", "fr_timer", 2)

modparam("dispatcher", "db_url", "mysql://opensips:opensips@localhost/opensips")
modparam("dispatcher", "ds_ping_method", "OPTIONS")
modparam("dispatcher", "ds_ping_interval", 5)
modparam("dispatcher", "ds_probing_threshold", 2)
modparam("dispatcher", "ds_probing_mode", 1)

modparam("load_balancer", "db_url", "mysql://opensips:opensips@localhost/opensips")
modparam("load_balancer", "probing_method", "OPTIONS")
modparam("load_balancer", "probing_interval", 5)

##### Routing Logic #####

# based on http://www.opensips.org/index.php?n=Resources.DocsTutLoadbalancing
route{

    if (!mf_process_maxfwd_header("10")) {
        sl_send_reply("483","Too Many Hops");
        exit;
    }

    if (!has_totag()) {
        # initial request
        record_route();
    }
    else {
        # sequential request - obey the indicated route
        loose_route();
        t_relay();
        exit;
    }

    # handle cancel and re-transmissions
    if ( is_method("CANCEL") ) {
        if ( t_check_trans() )
            t_relay();
        exit;
    }

    # from now on we have only the initial requests

    # select the node that'll handle the call (load balanced)
    # the method used is different for invite/register requests
    # unknown methods are rejected here
    if (is_method("INVITE")) {
        if (!load_balance("1","pstn","1")) {
            send_reply("503","Service Unavailable");
            exit;
        }
    }
    else if (is_method("REGISTER")) {
        if (!ds_select_dst("1", "0")) {
            send_reply("503","Service Unavailable");
            exit;
        }
    }
    else {
        send_reply("405","Method Not Allowed");
        exit;
    }
}
```

```
# route the request
if (!t_relay()) {
    sl_reply_error();
}
}
```

Restart OpenSIPS:

```
service opensips restart
```

Done!

MySQL Cluster

For calls to failover without dropping, the dialogs module must share its information in a shared database. This then means that on failover UDP calls will not be dropped since OpenSIPS will still know where to forward any SIP packets passing through the passive (now active) server as they'll be in its database. TCP calls cannot failover as gracefully in this way because the TCP connection will not be known to Linux/OpenSIPS on the passive server.

The database also provides a source for the list of hosts and weights/channels for the dispatcher and load_balancer modules. These must be told to reload the data from the database via a MI module if the database is updated.

There are multiple methods for setting up a shared database. You can have a single database, but this provides a point of failure. You can also have a pair of database servers with master-master, but this can cause problems since the replication is asynchronous. The MySQL Cluster solution provides a nice solution.

For a fully redundant setup, you should have at least 2 management servers, 2 data nodes and 2 sql nodes. All can be on the same hardware or on dedicated hardware. For this example they'll all run on lb1 and lb2, but in real usage it might be advisable to move them onto separate servers to reduce the load on the load balancer servers. This could be a LAN which lb1 & lb2 are also on, to remove them from the public Internet (this is advisable since MySQL Cluster uses no authentication/encryption).

FreeSWITCH Configuration

FreeSWITCH needs very little additional configuration to function with the load balancer setup. Your existing setup will mostly work just fine. There are a couple of areas that will need attention though.

Authentication

User authentication would just fine, as the 401 Not Authorized and authentication headers are forwarded by the SIP proxy.

ACLs might be trickier however, since the requests will all come from the virtual IP.

TODO: is it possible to get FreeSWITCH to authenticate against an ACL from a value in a header if the request comes from an IP matching a proxy ACL? If so OpenSIPS could add an IP header

SIP Registrations

REGISTER requests will go to a host in FreeSWITCH cluster. For the other hosts to know about the registration they will need to store the mod_sofia data in a shared database (single database, master-master replication or MySQL cluster). All hosts must also be configured to handle calls for the same domain, and the registration must be to this domain.

CDRs

The network IP logged in the CDRs will be that of the load balancer not the client's IP, this should be beared in mind.

See Also

- [Enterprise Deployment](#)
- [OpenSIPS using dispatcher](#) - Using OpenSIPS to provide load balancing using dispatcher.
- [Linux-HA project \(heartbeat\)](#)
- [OpenSIPS project](#)
- [OpenSIPS dispatcher module documentation](#)
- [OpenSIPS load balancer module documentation](#)
- [MySQL Cluster](#)