Linux Firewall Cluster with Pacemaker and Corosync

admin October 7, 2013

GIT – I will be using a template of CentOS 6.3 I created, but you could start from the minimal install of CentOS 6.2 and do a yum update to 6.3.

Some basic things you should get out of the way before you start: (Do on both servers)

Setup your IP's and networking
Setup your hostnames
Make sure you can get out to the public internet
Resolve FQDN -> can you ping google.com

Add our usual repos: (Do on both servers)

```
# EPEL
rpm -Uvh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
# Install CentALT repo for more updated packages (Only if you are doing something that needs newer rpms like Apache or Nginx
rpm -Uvh http://centos.alt.ru/repository/centos/6/x86_64/centalt-release-6-1.noarch.rpm
```

Install some basic software: (Do on both servers)

```
yum install -y vim git make automake gcc gcc-c++ glibc gd gd-devel glib-devel glibc-headers wget curl tar nc libcurl-devel openssl-devel zlib-devel zlib patch readline readline-devel libffi-devel curl-devel zip rsync ruby rubygems ruby-devel  htop mlocate python-dateutil redhat-rpm-config ntp python-lxml
```

Add your two hosts to your hosts file: (Do on both servers)

```
vim /etc/hosts
# Use whatever your ips and hostnames are here
10.1.0.1   server01
10.1.0.2   server02
```

Install Pacemaker: (Do on both servers)

```
yum install -y pacemaker cman ccs
```

Add them to start up: (Do on both servers)

```
chkconfig pacemaker on
```

NOTE: This step takes way to long to finish, so I suggest going and doing something productive (like reading the rest of this guide or checking out my other guides!) while waiting. Literally can take 15-30 minutes depending on the number of processors you have.

Create Corosync Key: (Only do on the first server)

```
corosync-keygen
```

Set permissions on the key: (Do first server)

```
chown root:root /etc/corosync/authkey
chmod 400 /etc/corosync/authkey
```

Rsync the key to your 2nd node:

```
rsync –avh /etc/corosync/authkey [email protected]:/etc/corosync/authkey
```

Get a couple additional files we will need: (One is an admin utility the other is a dependency) (Do on both servers)

```
rpm -Uvh http://download.opensuse.org/repositories/network:/ha-clustering:/Stable/CentOS_CentOS-6/x86_64/pssh-2.3.1-2.1.x86_64.rpm
rpm -Uvh http://download.opensuse.org/repositories/network:/ha-clustering:/Stable/CentOS_CentOS-6/x86_64/crmsh-1.2.6-4.4.x86_64.rpm
```

Note: Log out of your putty session and log back in to all the CRM utility to be ran.

Install a base corosync config: (Do on first server)

```
cp /etc/corosync/corosync.conf.example /etc/corosync/corosync.conf
```

Edit the corosync.conf file: (Do on first server)

```
vim /etc/corosync/corosync.conf
compatibility: whitetank
totem {
        version: 2
        secauth: off
        threads: 0
        interface {
                ringnumber: 0
                # the bindnetaddr is the ip of your base subnet since we are using 10.1.0.0 for an example
                bindnetaddr: 10.0.0.0
        broadcast: yes
        mcastport: 5405
                ttl: 1
        }
}
logging {
        fileline: off
        to_stderr: no
        to_logfile: yes
        to_syslog: yes
        logfile: /var/log/cluster/corosync.log
        debug: off
        timestamp: on
        logger_subsys {
                subsys: AMF
                debug: off
        }
}
amf {
        mode: disabled
}
service {
        name: pacemaker
        ver: 0
}
```

Make the log folder: (Do on both servers)

```
mkdir /var/log/cluster
```

Rsync the config file to the 2nd node in cluster:

```
rsync -avh /etc/corosync/corosync.conf [email protected]:/etc/corosync/corosync.conf
```

Start the service: (Do on first server)

```
/etc/init.d/corosync start
```

Check the log file make sure nothing jumps out for errors: (Do on first server)

```
tail -f /var/log/cluster/corosync.log
```

Create the folder crm configure needs: (Do on both servers)

```
mkdir -p /var/lib/pacemaker/cores/root
```

Stop the service as we will use crm to manage it and pacemaker to start it: (Do on first server)

```
/etc/init.d/corosync stop
```

Note: If for some reason you can't get corosync to stop and it takes forever to unload. Reboot your server, this seems to be an issue on CentOS the first time its started.

Setup the pacemaker cluster: (Do on first server)

```
# At the command line
ccs -f /etc/cluster/cluster.conf –createcluster openvpn
ccs -f /etc/cluster/cluster.conf –addnode server01
ccs -f /etc/cluster/cluster.conf –addnode server02
ccs -f /etc/cluster/cluster.conf –addfencedev pcmk agent=fence_pcmk
ccs -f /etc/cluster/cluster.conf –addmethod pcmk-redirect server01
ccs -f /etc/cluster/cluster.conf –addmethod pcmk-redirect server02
ccs -f /etc/cluster/cluster.conf –addfenceinst pcmk server01 pcmk-redirect port=server01
ccs -f /etc/cluster/cluster.conf –addfenceinst pcmk server02 pcmk-redirect port=server02
echo "CMAN_QUORUM_TIMEOUT=0" >> /etc/sysconfig/cman
```

Rsync cluster file to 2nd server:

```
rsync -avh /etc/cluster/cluster.conf [email protected]:/etc/cluster/cluster.conf
```

Start pacemaker: (On the first server)

```
1    /etc/init.d/pacemaker start
```

Allow Pacemaker through iptables: (On both servers)

```
iptables -I INPUT 1 –protocol udp –dport 5405 -j ACCEPT
iptables -I INPUT 1 –protocol udp –sport 5404 -j ACCEPT
iptables -I OUTPUT 1 –protocol udp –dport 5405 -j ACCEPT
iptables -I OUTPUT 1 –protocol udp –sport 5404 -j ACCEPT
```

Note: If you don't add these iptables rules you may get the error: ERROR: running cibadmin -Ql: Could not establish cib_rw connection: Connection refused (111)

Setup the cluster options: (On the first server)

```
# At the command line
crm configure
> edit
```

Note: This will open the xml file inside vi / vim or whatever your default editor is.

Delete everything in the file and paste in the following: (On the first server)

```
node server01
node server02
# This section is our public ip (or external that your isp gives you)
primitive vip1 ocf:heartbeat:IPaddr2 \
    params ip="192.168.0.1" cidr_netmask="24" nic="eth0" \
    op monitor interval="40s" timeout="20s"
# This section is our internal ip and the gateway we use for all boxes on the network
primitive vip2 ocf:heartbeat:IPaddr2 \
    params ip="10.1.0.1" cidr_netmask="8" nic="eth1" \
    op monitor interval="40s" timeout="20s"
# This section is to raise an additional ip used for public web traffic to something like haproxy or load
balancers
primitive vip3 ocf:heartbeat:IPaddr2 \
    params ip="10.2.0.1" cidr_netmask="8" nic="eth2" \
    op monitor interval="40s" timeout="20s"
property $id="cib-bootstrap-options" \
    dc-version="1.1.8-7.el6-394e906" \
    cluster-infrastructure="cman" \
    expected-quorum-votes="2" \
    stonith-enabled="false" \
    no-quorum-policy="ignore"
rsc_defaults $id="rsc-options" \
    resource-stickiness="100"
```

 # Small additional note, if you are doing a cluster of more than 2 servers, do some research on the no-quorum-policy and stonith-enabled.

Save and commit your changes:

```
# Exit and save
:wq
# At the crm command line
commit
exit
```

Restart Pacemaker and check the logs: (On the first server)

```
/etc/init.d/pacemaker restart
tail -f /var/log/cluster/corosync.log
```

Start Pacemaker on the 2nd server:

```
/etc/init.d/pacemaker start
```

I had some trouble getting my firewall rules (iptables) to stick when the ip floated from one to the other, to resolve this issue make sure you are saving your iptable rules if you are using an init script.

```
service iptables save
```

To test our setup we will launch crm_mon on the passive node, while running a ping in another session to watch for packet loss:

```
crm_mon
```

On the server that currently has the virtual ips:

```
reboot
```

You should see the resources float from the active node to the standby, and as long as your firewall rules are correct, you shouldn’t lose more than 1-2 packets.

Aaaandd we are done! As always let me know if I missed anything or if you run into any issues in the comments section!