How to make MaxScale High Available

Corosync/Pacemaker setup
& MaxScale init script

Massimiliano Pinto

Last Updated: 30th June 2014

Contents

Overview

The proposed solution is with Pacemaker / Corosync cluster software with the following minimum requirements:

- MaxScale process is started/stopped and monitored via /etc/init.d/maxscale script that is LSB compatible in order to be managed by Pacemaker resource manager

- A Virtual IP is set providing the access to the MaxScale process that could be set to one of the cluster nodes

- Pacemaker/Corosync basic knowledge

The document shows a detailed example with a three node cluster based on Linux Centos 6.5

Clustering Software installation

On each node in the cluster do the following steps:

(1) Add clustering repos to yum

```
# vi /etc/yum.repos.d/ha-clustering.repo
```

Add the following to the file

```
[haclustering]
name=HA Clustering
baseurl=http://download.opensuse.org/repositories/network:/ha-cl
ustering:/Stable/CentOS_CentOS-6/
enabled=1
gpgcheck=0
```

(2) Install the software

```
# yum install pacemaker corosync crmsh
```

Package versions used

Package **pacemake**r-1.1.10-14.el6_5.3.x86_64
Package **corosync**-1.4.5-2.4.x86_64
Package **crmsh**-2.0+git46-1.1.x86_64

(3) Assign hostname on each node

In this example the three names used for the nodes are:
node1,node,node3

```
# hostname node1
...
# hostname nodeN
```

(4) For each node add server names in /etc/hosts

```
[root@node3 ~]# vi /etc/hosts

10.74.14.39     node1
10.228.103.72   node2
10.35.15.26     node3 current-node

[root@node1 ~]# vi /etc/hosts

10.74.14.39     node1 current-node
10.228.103.72   node2
10.35.15.26     node3 current-node

...
```

Please note: add **current-node** as an alias for the current node in each of the /etc/hosts files.

(5) Prepare authkey for optional cryptographic use

On one of the nodes, say node2 run the corosync-keygen utility and follow

```
[root@node2 ~]# corosync-keygen

Corosync Cluster Engine Authentication key generator.
       Gathering 1024 bits for key from /dev/random.
       Press keys on your keyboard to generate entropy.
```

After completion the key will be found in /etc/corosync/authkey.

(6) Prepare the corosync configuration file

Using node2 as an example:

```
[root@node2 ~]# vi /etc/corosync/corosync.conf
```

Add the following to the file:

```
# Please read the corosync.conf.5 manual page
compatibility: whitetank

totem {
        version: 2
        secauth: off
        interface {
                member {
                        memberaddr: node1
                }
                member {
                        memberaddr: node2
                }
                member {
                        memberaddr: node3
                }
           ringnumber: 0
                 bindnetaddr: current-node
                 mcastport: 5405
                 ttl: 1
        }
        transport: udpu
}

logging {
        fileline: off
        to_logfile: yes
        to_syslog: yes
        logfile: /var/log/cluster/corosync.log
        debug: off
        timestamp: on
        logger_subsys {
                subsys: AMF
                debug: off

        }
}

# this will start Pacemaker processes
service {
ver: 0
name: pacemaker
}
```

Please note in this example:

- unicast UDP is used
- bindnetaddr for corosync process is current-node, that has the right value on each node due to the alias added in /etc/hosts above
- Pacemaker processes are started by the corosync daemon, so there is no need to launch it via /etc/init.d/pacemaker start

<u>(7) copy configuration files and auth key on each of the other nodes</u>

```
[root@node2 ~]# scp /etc/corosync/*  root@node1:/etc/corosync/
[root@node2 ~]# scp /etc/corosync/*  root@nodeN:/etc/corosync/
...
```

<u>(8) Corosync needs port *5*405 to be opened:</u>

- configure any firewall or iptables accordingly

For a quick start just disable iptables on each nodes:

```
[root@node2 ~]# service iptables stop
…
[root@nodeN ~]# service iptables stop
```

<u>(9) Start Corosyn on each node:</u>

```
[root@node2 ~] #/etc/init.d/corosync start

…

[root@nodeN ~] #/etc/init.d/corosync start
```

and check the corosync daemon is successfully bound to port 5404:

```
[root@node2 ~] #netstat -na | grep 5405

udp        0      0 10.228.103.72:5405        0.0.0.0:*
```

Check if other nodes are reachable with nc utility and option UDP (-u):

```
[root@node2 ~] #echo "check ..."  | nc -u node1 5405
[root@node2 ~] #echo "check ..."  | nc -u node3 5405
...
[root@node1 ~] #echo "check ..."  | nc -u node2 5405
[root@node1 ~] #echo "check ..."  | nc -u node3 5405

…
```

If the following message is displayed

nc: Write error: Connection refused

There is an issue with communication between the nodes, this is most likely to be an issue with the firewall configuration on your nodes. Check and resolve issues with your firewall configuration.

(10) Check the cluster status, from any node

```
[root@node3 ~]# crm status
```

After a while this will be the output:

```
[root@node3 ~]# crm status
Last updated: Mon Jun 30 12:47:53 2014
Last change: Mon Jun 30 12:47:39 2014 via crmd on node2
Stack: classic openais (with plugin)
Current DC: node2 - partition with quorum
Version: 1.1.10-14.el6_5.3-368c726
3 Nodes configured, 3 expected votes
0 Resources configured


Online: [ node1 node2 node3 ]
```

For the basic setup disable the following properties:

- stonith
- quorum policy

```
[root@node3 ~]# crm configure property 'stonith-enabled'='false'
[root@node3 ~]# crm configure property
'no-quorum-policy'='ignore'
```

For more information see:

http://www.clusterlabs.org/doc/crm_fencing.html
http://clusterlabs.org/doc/

The configuration is automatically updated on every node:

Check it from another node, say node1

```
[root@node1 ~]# crm configure show
```

```
node node1
node node2
node node3
property cib-bootstrap-options: \
        dc-version=1.1.10-14.el6_5.3-368c726 \
        cluster-infrastructure="classic openais (with plugin)" \
        expected-quorum-votes=3 \
        stonith-enabled=false \
        no-quorum-policy=ignore \
        placement-strategy=balanced \
        default-resource-stickiness=infinity
```

The Corosync / Pacemaker cluster is ready to be configured to manage resources.

MaxScale init script /etc/init.d/maxscale

The MaxScale /etc/init.d./maxscale script allows to start/stop/restart and monitor maxScale process running in the system.

Edit it and modify the **MAXSCALE_BASEDIR** to match the installation directory you choose when you installed MaxScale.

Note:

It could be necessary to modify other variables, such as MAXSCALE_BIN, MAXSCALE_HOME, MAXSCALE_PIDFILE and LD_LIBRARY_PATH for a non standard setup.

```
[root@node1 ~]# /etc/init.d/maxscale
Usage: /etc/init.d/maxscale
{start|stop|status|restart|condrestart|reload}
```

- Start

```
[root@node1 ~]# /etc/init.d/maxscale start
Starting MaxScale: maxscale (pid 25892) is running...      [ OK ]
```

- Start again

```
[root@node1 ~]# /etc/init.d/maxscale start
Starting MaxScale:  found maxscale (pid  25892) is running.[ OK ]
```

- Stop

```
[root@node1 ~]# /etc/init.d/maxscale stop
Stopping MaxScale:                                          [ OK ]
```

- Stop again

```
[root@node1 ~]# /etc/init.d/maxscale stop
Stopping MaxScale:                                          [FAILED]
```

- Status (MaxScale not running)

```
[root@node1 ~]# /etc/init.d/maxscale status
MaxScale is stopped                                        [FAILED]
```

The script exit code for “status” is 3

- Status (MaxScale is running)

```
[root@node1 ~]# /etc/init.d/maxscale status
Checking MaxScale status: MaxScale (pid  25953) is running.[ OK ]
```

The script exit code for “status” is 0

Note: the MaxScale script is LSB compatible and returns the proper exit code for each action:

For more informations;

http://www.linux-ha.org/wiki/LSB_Resource_Agents

After checking maxScale is well managed by the /etc/init.d/script is possible to configure the MAxScale HA via Pacemaker.

Configure MaxScale for HA with Pacemaker

```
[root@node2 ~]# crm configure primitive MaxScale lsb:maxscale \
op monitor interval=”10s” timeout=”15s” \
op start interval=”0” timeout=”15s” \
op stop interval=”0” timeout=”30s”
```

MaxScale resource will be started:

```
[root@node2 ~]# crm status
Last updated: Mon Jun 30 13:15:34 2014
Last change: Mon Jun 30 13:15:28 2014 via cibadmin on node2
Stack: classic openais (with plugin)
Current DC: node2 - partition with quorum
Version: 1.1.10-14.el6_5.3-368c726
3 Nodes configured, 3 expected votes
1 Resources configured


Online: [ node1 node2 node3 ]

 MaxScale  (lsb:maxscale): Started node1
```

Basic use cases:

1. Resource restarted after a failure:

MaxScale Pid is, $MAXSCALE_PIDFILE=$MAXSCALE_HOME/log/maxscale.pid

In the example is 26114, kill the process immediately:

```
[root@node2 ~]# kill -9 26114
```

```
[root@node2 ~]# crm status
Last updated: Mon Jun 30 13:16:11 2014
Last change: Mon Jun 30 13:15:28 2014 via cibadmin on node2
Stack: classic openais (with plugin)
Current DC: node2 - partition with quorum
Version: 1.1.10-14.el6_5.3-368c726
3 Nodes configured, 3 expected votes
1 Resources configured


Online: [ node1 node2 node3 ]

Failed actions:
    MaxScale_monitor_15000 on node1 'not running' (7): call=19,
status=complete, last-rc-change='Mon Jun 30 13:16:14 2014',
queued=0ms, exec=0ms
```

Note the **MaxScale_monitor** failed action

After a few seconds it will be started again:

```
[root@node2 ~]# crm status
Last updated: Mon Jun 30 13:21:12 2014
Last change: Mon Jun 30 13:15:28 2014 via cibadmin on node1
Stack: classic openais (with plugin)
Current DC: node2 - partition with quorum
Version: 1.1.10-14.el6_5.3-368c726
```

```
3 Nodes configured, 3 expected votes
1 Resources configured


Online: [ node1 node2 node3 ]

 MaxScale  (lsb:maxscale): Started node1
```

2. The resource cannot be migrated to node1 for a failure:

First, migrate the the resource to another node, say node3

```
[root@node1 ~]# crm resource migrate MaxScale node3

...

Online: [ node1 node2 node3 ]




Failed actions:
    MaxScale_start_0 on node1 'not running' (7): call=76,
status=complete, last-rc-change='Mon Jun 30 13:31:17 2014',
queued=2015ms, exec=0ms
```

Note the **MaxScale_start** failed action on node1, and after a few seconds

```
[root@node3 ~]# crm status
Last updated: Mon Jun 30 13:35:00 2014
Last change: Mon Jun 30 13:31:13 2014 via crm_resource on node3
Stack: classic openais (with plugin)
Current DC: node2 - partition with quorum
Version: 1.1.10-14.el6_5.3-368c726
3 Nodes configured, 3 expected votes
1 Resources configured
```

```
Online: [ node1 node2 node3 ]

 MaxScale  (lsb:maxscale): Started node2

Failed actions:
    MaxScale_start_0 on node1 'not running' (7): call=76,
status=complete, last-rc-change='Mon Jun 30 13:31:17 2014',
queued=2015ms, exec=0ms
```

Successfully, MaxScale has been started on a new node: node2.

Note: Failed actions remain in the output of crm status.
With “crm resource cleanup MaxScale” is possible to cleanup the messages:

```
[root@node1 ~]# crm resource cleanup MaxScale
Cleaning up MaxScale on node1
Cleaning up MaxScale on node2
Cleaning up MaxScale on node3
```

The cleaned status is visible from other nodes as well:

```
[root@node2 ~]# crm status
Last updated: Mon Jun 30 13:38:18 2014
Last change: Mon Jun 30 13:38:17 2014 via crmd on node3
Stack: classic openais (with plugin)
Current DC: node2 - partition with quorum
Version: 1.1.10-14.el6_5.3-368c726
3 Nodes configured, 3 expected votes
1 Resources configured


Online: [ node1 node2 node3 ]

 MaxScale  (lsb:maxscale): Started node2
```

Add a Virtual IP (VIP) to the cluster

It’s possible to add a virtual IP to the cluster:
MaxScale process will be only contacted with this IP, that mat move across nodes with maxscale process as well.

Setup is very easy:

assuming an addition IP address is available and can be added to one of the nodes, this i the new configuration to add:

```
[root@node2 ~]# crm configure primitive maxscale_vip
ocf:heartbeat:IPaddr2 params ip=192.168.122.125 op monitor
interval=10s
```

MaxScale process and the VIP must be run in the same node, so it’s mandatory to add to the configuration the group ‘maxscale_service’.

```
[root@node2 ~]# crm configure group maxscale_service maxscale_vip
MaxScale
```

The final configuration is, from another node:

```
[root@node3 ~]# crm configure show
node node1
node node2
node node3
primitive MaxScale lsb:maxscale \
        op monitor interval=15s timeout=10s \
        op start interval=0 timeout=15s \
        op stop interval=0 timeout=30s
primitive maxscale_vip IPaddr2 \
        params ip=192.168.122.125 \
        op monitor interval=10s
group maxscale_service maxscale_vip MaxScale \
        meta target-role=Started
property cib-bootstrap-options: \
        dc-version=1.1.10-14.el6_5.3-368c726 \
        cluster-infrastructure="classic openais (with plugin)" \
        expected-quorum-votes=3 \
        stonith-enabled=false \
        no-quorum-policy=ignore \
        placement-strategy=balanced \
        last-lrm-refresh=1404125486
```

Check the resource status:

```
[root@node1 ~]# crm status
Last updated: Mon Jun 30 13:51:29 2014
Last change: Mon Jun 30 13:51:27 2014 via crmd on node1
Stack: classic openais (with plugin)
Current DC: node2 - partition with quorum
Version: 1.1.10-14.el6_5.3-368c726
3 Nodes configured, 3 expected votes
2 Resources configured

Online: [ node1 node2 node3 ]

 Resource Group: maxscale_service
     maxscale_vip     (ocf::heartbeat:IPaddr2):  Started node2
     MaxScale    (lsb:maxscale): Started node2
```

With both resources on node2, now MaxScale service will be reachable via the configured VIP address 192.168.122.125