Table of Contents

Cluster Creation for RHEL/CentOS 6

This section describe how to create a cluster using *CMAN/Ricci*.

Cluster Creation

Set the password for *ricci*, and configure the appropriate services for startup:

```
[root@onmssrv01 ~]# passwd ricci
Changing password for user ricci.
New password:
passwd: all authentication tokens updated successfully.
[root@onmssrv01 ~]# chkconfig ricci on | service ricci start
Starting oddjobd: [  OK  ]
generating SSL certificates...  done
Generating NSS database...  done
Starting ricci: [  OK  ]
[root@onmssrv01 ~]# chkconfig cman on
[root@onmssrv01 ~]# chkconfig rgmanager on
```

If NetworkManager is installed, it is recommended to turn it off:

```
[root@onmssrv01 ~]# chkconfig NetworkManager off | service NetworkManager stop
```

Repeat the the above process for the standby server (i.e. *onmssrv02*).

From the active node (i.e. *onmssrv01*), run the below command to create a new cluster.

The following command will create the cluster configuration file `/etc/cluster/cluster.conf`. If the file already exists, it will replace the existing `cluster.conf` with the newly created `cluster.conf`.

```
[root@onmssrv01 ~]# ccs -h onmssrv01.local --createcluster onmscluster
onmssrv01.local
```

The command will ask you for the password configured for *ricci*.

```
[root@onmssrv01 ~]# ls -l /etc/cluster/cluster.conf
-rw-r-----. 1 root root 188 Jul 14 17:40 /etc/cluster/cluster.conf
```

Also keep in mind that we are running these commands only from one node on the cluster and we are not yet ready to propagate the changes to the other node on the cluster.

After creating the cluster, the `cluster.conf` file will look like the following:

```
[root@onmssrv01 ~]# cat /etc/cluster/cluster.conf
<?xml version="1.0"?>
<cluster config_version="1" name="onmscluster">
  <fence_daemon/>
  <clusternodes/>
  <cman/>
  <fencedevices/>
  <rm>
    <failoverdomains/>
    <resources/>
  </rm>
</cluster>
```

Once the cluster is created, we need to add the participating nodes to the cluster using the `ccs` command as shown below.

```
[root@onmssrv01 ~]# ccs -h onmssrv01.local --addnode onmssrv01.local
Node onmssrv01.local added.
```

Next, add the second node *rh2* to the cluster as shown below.

```
[root@onmssrv01 ~]# ccs -h onmssrv01.local --addnode onmssrv02.local
Node onmssrv02.local added.
```

Once the nodes are created, you can use the following command to view all the available nodes in the cluster. This will also display the *node id* for the corresponding node.

```
[root@onmssrv01 ~]# ccs --lsnodes
onmssrv01.local: nodeid=1
onmssrv02.local: nodeid=2
```

This above will also add the nodes to the `cluster.conf` file as shown below.

```
[root@onmssrv01 ~]# cat /etc/cluster/cluster.conf
<?xml version="1.0"?>
<cluster config_version="3" name="onmscluster">
  <fence_daemon/>
  <clusternodes>
    <clusternode name="onmssrv01.local" nodeid="1"/>
    <clusternode name="onmssrv02.local" nodeid="2"/>
  </clusternodes>
  <cman/>
  <fencedevices/>
  <rm>
    <failoverdomains/>
    <resources/>
  </rm>
</cluster>
```

Make sure the cluster has the appropriate flags to work with two servers:

```
[root@onmssrv01 ~]# ccs -h onmssrv01.local --setcman two_node=1 expected_votes=1
```

To add a failover domain, execute the following command. In this example, I created domain named as `onms_domain`,

```
[root@onmssrv01 ~]# ccs -h onmssrv01.local --addfailoverdomain onms_domain ordered nofailback
```

Once the failover domain is created, add both the nodes to the failover domain as shown below:

```
[root@onmssrv01 ~]# ccs -h onmssrv01.local --addfailoverdomainnode onms_domain onmssrv01.local 1

[root@onmssrv01 ~]# ccs -h onmssrv01.local --addfailoverdomainnode onms_domain onmssrv02.local 2
```

You can view all the nodes in the failover domain using the following command.

```
[root@onmssrv01 ~]# ccs --lsfailoverdomain
onms_domain: restricted=0, ordered=1, nofailback=1
  onmssrv01.local: priority=1
  onmssrv02.local: priority=2
```

Now it is time to add the resources. This indicates the applications (i.e., *pgpool-II* and *OpenNMS*) that should failover along with ip and the shared filesystem when a node fails.

When you are ready to add resources, there are 2 ways you can do this. You can add them as global resources, or add a resource directly to resource group or service.

The advantage of adding resources as global resource is that if you want to add the resource to more than one service group you can just reference the global resource on your service or resource group.

The global resources we're going to add are the following:

- A shared filesystem for `/opt/opennms/etc`
- A shared filesystem for `/var/opennms`
- A shared filesystem for `/etc/pgpool-II`
- The init script for *pgpool-II*
- The init script for *OpenNMS*
- The floating IP Address

Use the above information for creating the *NFS* resources:

```
[root@onmssrv01 ~]# ccs --addresource netfs name=onms_etc host=nfssrv01.local export=/opt/opennms/etc
mountpoint=/opt/opennms/etc fstype=nfs4
[root@onmssrv01 ~]# ccs --addresource netfs name=onms_var host=nfssrv01.local export=/opt/opennms/share
mountpoint=/var/opennms fstype=nfs4
[root@onmssrv01 ~]# ccs --addresource netfs name=pgpool_etc host=nfssrv01.local export=/opt/opennms/pgpool
mountpoint=/etc/pgpool-II fstype=nfs4
```

Then, add the script resources:

```
[root@onmssrv01 ~]# ccs --addresource script name=pgpool_bin file=/etc/init.d/pgpool
[root@onmssrv01 ~]# ccs --addresource script name=onms_bin file=/etc/init.d/opennms
```

IMPORTANT the last declaration is the reason why START_TIMEOUT=0 cannot be used on /opt/opennms/etc/opennms.conf.

Then, add the floating IP resource:

```
[root@onmssrv01 ~]# ccs --addresource ip address=192.168.205.150 monitor_link=1
```

Finally, create the cluster service:

```
[root@onmssrv01 ~]# ccs --addservice onms_svc domain=onms_domain recovery=relocate autostart=1
```

Then, add the resource references with the desired order to the cluster service created above.

```
[root@onmssrv01 ~]# ccs --addsubservice onms_svc ip ref=192.168.205.150
[root@onmssrv01 ~]# ccs --addsubservice onms_svc netfs ref=onms_etc
[root@onmssrv01 ~]# ccs --addsubservice onms_svc netfs ref=onms_var
[root@onmssrv01 ~]# ccs --addsubservice onms_svc netfs ref=pgpool_etc
[root@onmssrv01 ~]# ccs --addsubservice onms_svc script ref=pgpool_bin
[root@onmssrv01 ~]# ccs --addsubservice onms_svc script ref=onms_bin
```

IMPORTANT as explained before, if you have issues with the NFS server, do not add a subservice for pgpool_etc.

The final `/etc/cluster/cluster.conf` should look like the following:

```
[root@onmssrv01 ~]# cat /etc/cluster/cluster.conf
<?xml version="1.0"?>
<cluster config_version="20" name="onmscluster">
  <fence_daemon/>
  <clusternodes>
    <clusternode name="onmssrv01.local" nodeid="1"/>
    <clusternode name="onmssrv02.local" nodeid="2"/>
  </clusternodes>
  <cman expected_votes="1" two_node="1"/>
  <fencedevices/>
  <rm>
    <failoverdomains>
      <failoverdomain name="onms_domain" nofailback="1" ordered="1" restricted="0">
        <failoverdomainnode name="onmssrv01.local" priority="1"/>
        <failoverdomainnode name="onmssrv02.local" priority="2"/>
      </failoverdomain>
    </failoverdomains>
    <resources>
      <netfs export="/opt/opennms/etc" fstype="nfs4" host="nfssrv01.local" mountpoint="/opt/opennms/etc"
name="onms_etc"/>
      <netfs export="/var/opennms" fstype="nfs4" host="nfssrv01.local" mountpoint="/opt/opennms/share"
name="onms_var"/>
      <netfs export="/opt/opennms/pgpool" fstype="nfs4" host="nfssrv01.local" mountpoint="/etc/pgpool-II"
name="pgpool_etc"/>
      <script file="/etc/init.d/pgpool" name="pgpool_bin"/>
      <script file="/etc/init.d/opennms" name="onms_bin"/>
      <ip address="192.168.205.150" monitor_link="1"/>
    </resources>
    <service autostart="1" domain="onms_domain" name="onms_svc" recovery="relocate">
      <ip ref="192.168.205.150"/>
      <netfs ref="onms_etc"/>
      <netfs ref="onms_var"/>
      <netfs ref="pgpool_etc"/>
      <script ref="pgpool_bin"/>
      <script ref="onms_bin"/>
    </service>
  </rm>
</cluster>
```

NOTE the `config_version` might be different, but that's ok.

Finally we can synchronize the cluster configuration across all members of the cluster:

```
[root@onmssrv01 ~]# ccs -h onmssrv01.local --sync --activate
[root@onmssrv01 ~]# ccs -h onmssrv01.local --checkconf
All nodes in sync.
```

The `sync` command might request the *ricci's* password on all the cluster members that requires synchronization (in this case, *onmssrv02*). Now, the `cluster.conf` file should exist on the standby server.

IMPORTANT Do not edit the `cluster.conf` file directly. Use always the `ccs` command for this purpose.

When we have the external resources ready, and cluster configuration is ready and synchronized with all the members, we are start the cluster.

```
[root@onmssrv01 ~]# css --startall
```

The above command is equivalent to start *cman* and then *rgmanager* on all cluster members.

Because *onmssrv01* has more priority than *onmssrv02*, you should see that the `onms_svc` service should start automatically on *onmssrv01*.

The following command provides a quick overview of the cluster and the status of the cluster services,

```
[root@onmssrv01 ~]# clustat
Cluster Status for onmscluster @ Tue Jul 14 12:20:19 2015
Member Status: Quorate

 Member Name                        ID   Status
 ------ ----                        ---- ------
 onmssrv01.local                       1 Online, Local, rgmanager
 onmssrv02.local                       2 Online, rgmanager

 Service Name                       Owner (Last)               State
 ------- ----                       ----- ------               -----
 service:onms_svc                   onmssrv01.local            started
```

It might show *starting*, as *opennms* could take time to initialize, but eventually, if there are no issues, you should see *started*. If the cluster service is not started check the cluster logs on `/var/log/cluster.`

At this time, if you open a browser and point it to the virtual IP address, you should see *OpenNMS up and running*:

http://192.168.205.150:8980/opennms

Test Failover

There are several ways to check if the cluster works.

1. Manually stop one the critical resources on the active node, the obvious one is the *OpenNMS* application:

```
[root@onmssrv01 ~]# service opennms stop
```

That will trigger the cluster failover operation which is move the `onms_svc` service to the standby node:

```
[root@onmssrv02 ~]# clustat
Cluster Status for onmscluster @ Tue Jul 14 12:20:19 2015
Member Status: Quorate

 Member Name                        ID   Status
 ------ ----                        ---- ------
 onmssrv01.local                       1 Online, Local, rgmanager
 onmssrv02.local                       2 Online, rgmanager

 Service Name                       Owner (Last)               State
 ------- ----                       ----- ------               -----
 service:onms_svc                   onmssrv02.local            started
```

Of course, you should see several states in the middle: *recoverable*, *recovering*, then *starting* and finally *started*.

There's always going to be a small gap when doing a failover on the *DB* cluster or the application cluster, so during that time, the application could miss some *DB* transactions or external events (like *SNMP Traps* or *Syslog* messages).

As you can see, it is not a good idea to manually stop *OpenNMS* on a cluster. Later I'll mention how to properly restart *OpenNMS* within a cluster.

Keep in mind that the `onms_svc` will continue running on *onmssrv02*, as the failover domain was configured to do this.

1. Relocate the cluster service to the standby node:

```
[root@onmssrv01 ~]# clusvcadm -r onms_svc -m onmssrv02
```

That will instruct the cluster to stop the `onms_svc` service on *onmssrv01* and start it on *onmssrv02*.

WebUI for Managing Cluster (luci)

The idea with *luci* is having a *WebUI* to configure and manage the cluster. The procedure is basically the same already explained with the `ccs` command. The *WebUI* itself will guide you for adding all the components.

Keep in mind that this is totally optional, and not necessary. Also, in case you want to experiment with it, the correct procedure is install the application on a dedicated server and not the cluster members, even knowing that this is possible.

In order to use *luci*, you must install the following group of packages on one of the cluster members, or on another machine that can reach both *OpenNMS* machines. For simplicity, I'm going to install *luci* on *onmssrv01*:

```
[root@onmssrv01 ~]# yum groupinstall "High Availability Management"
[root@onmssrv01 ~]# chkconfig luci on | service luci start
```

If the internal firewall is enabled, open the TCP 8084 port.

```
[root@onmssrv01 ~]# iptables -A INPUT -p tcp --dport 8084 -j ACCEPT
[root@onmssrv01 ~]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
```

On your browser open the following URL:

https://onmssrv01:8084/

Replace the *FQDN* with the appropriate one if needed. Use a local administrator account or a user that can run the `ccs` command.

Restart OpenNMS

Because *OpenNMS* is part of a cluster service, the standard way to start, stop and restart *OpenNMS* cannot be used. Otherwise, the cluster will be confused.

The following command must be used from the active node to restart *OpenNMS* on the same cluster node:

```
[root@onmssrv01 ~]# clusvcadm -R onms_svc
```

This operation could take a few minutes as explained before.

To relocate the cluster service to another node:

```
[root@onmssrv01 ~]# clusvcadm -r onms_svc -m onmssrv02
```

To stop the cluster, use the following command from one of the cluster members:

```
[root@onmssrv01 ~]# ccs --stopall
```

To start the cluster, use the following command from one of the cluster members:

```
[root@onmssrv01 ~]# ccs --startall
```

To stop the cluster services on a given node:

```
[root@onmssrv01 ~]# ccs --stop onmssrv02
```

To start the cluster services on a given node:

```
[root@onmssrv01 ~]# ccs --start onmssrv02
```

Any of the above commands might ask you the password of the *ricci* account.