RedHat Cluster (Pacemaker/Corosync)

Chapter 1:- Introduction and basic difference from previous RHEL cluster and latest RHEL Cluster.

Red hat cluster allows you to configure and manage group of resources (computer/servers) to work together to provide high availability and form group called cluster.

From RHEL7 onward red hat uses **pacemaker** as default cluster resource manage. **Corosync** is open source cluster engine which is responsible to manage the cluster interconnect and maintains the same cluster configuration across all the cluster nodes. i.e.

Pacemaker ➔ Cluster resource manager
Corosync ➔ Cluster Engine which help to connect and maintain cluster nodes configuration.

In RHEL5, earlier red hat were using **CMAN** and **RGManger** for cluster configuration. Pacemaker was shipping from REHL 6 onwards but not used widely since ccs /crm was part of it.

➔ Default Cluster Configuration path is **/etc/corosync/corosync.conf**, it contains Membership and Quorum configuration and file **/var/lib/heartbeat/crm/cib.xml** contains Cluster node and Resource configuration.

➔ Default command utility is "pcs" and "pcsd is pacemaker web utility tool.

➔ Default cluster user "hacluster"

➔ Default service name "pcsd" & "corosync"

Chapter 2:- Introduction to Pacemaker & CoroSync

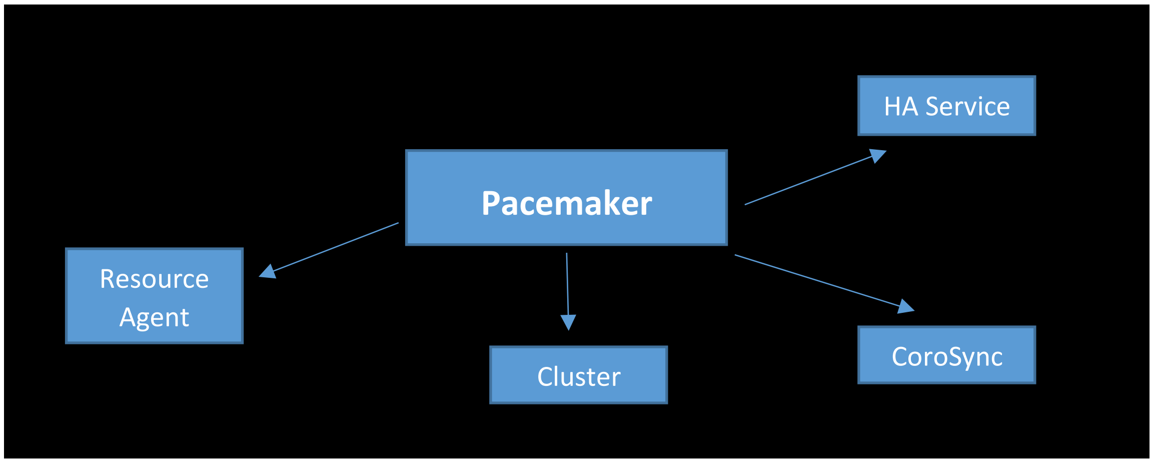
Pacemaker is robust and powerful open source tool/ resource manager in RHEL 6/7 as high availability add-on. Pacemaker is simplified cluster configuration and cluster management is made really simple in it. Corosync and pacemaker look completely different than earlier cluster CMAN and RGManager.

Corosync engine help to maintain cluster configuration in all nodes and maintain consistency.

Red hat cluster core components:-

A – **Resource Agent:** - Resource agent are scrips that help to stop start and monitor resource configured in cluster.

B – **Resource Manager:** - Pacemaker provides brain that processes & reacts to event regarding cluster. This event can be joining or leaving node in cluster. Resource events caused by failures, maintenance and scheduled activities and other administrative actions. Resource manager is nothing but "pacemaker".



Cluster basic structure

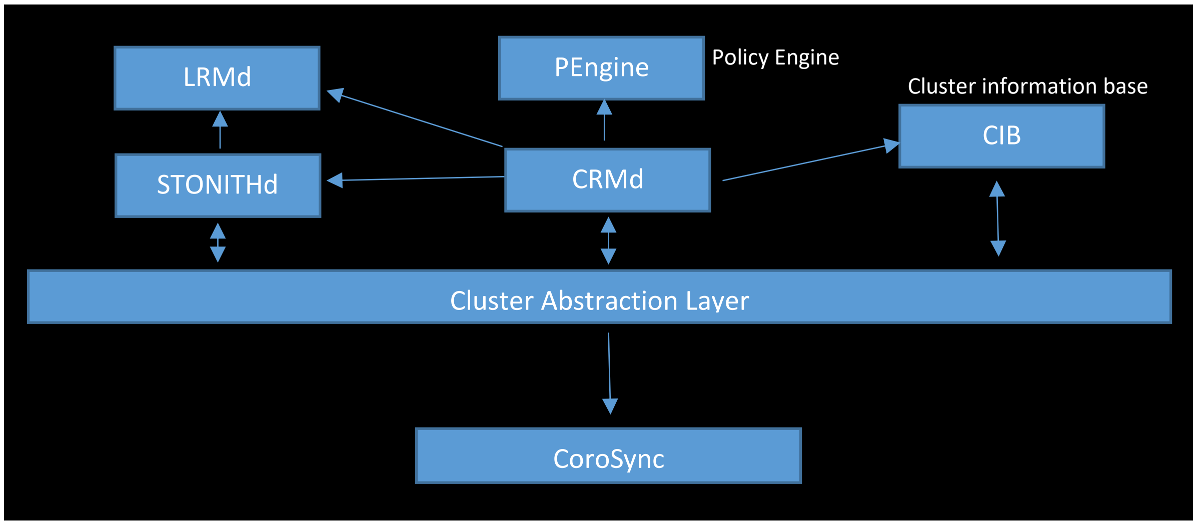
Pacemaker is responsible is to provide maximum availability of cluster resource and service by detecting and recovering from node and resource level failure.

It uses massaging and membership capabilities provided by corosync to keep the resource available on any of the cluster node.

Features:-

- ➔ Detection and recovery of nodes and service level failure
- ➔ Storage agnostic, no need of shared storage
- ➔ Resource agnostic, ay application can be scripted and configured in cluster as resource
- ➔ Supports fencing (STONITH) to ensure data integrity
- ➔ Support larger cluster of 32 nodes and minimum of 2 node cluster
- ➔ Support practically any redundant cluster
- ➔ Automatically replicate cluster configuration across nodes
- ➔ Supports advance service types
    - Clones :- for services those are active on multiple nodes
    - Multi-state:- for services for multiple modes (master / slave).

Pacemaker Key component:-



➔ CIB (Cluster Information Base)

It uses XML file (cib.xml) to represent the cluster configuration and current state of cluster to all the nodes. This file kept in sync with other member node of cluster and it is used by PEngine to compute ideal state of the cluster.

➔ CRMd (Cluster Resource Management daemon)

Pacemaker centralize all cluster decision making by electing one of CRMd instance to act as master. If one CRMd instance fails it automatically start new instance.

➔ LRMd (Local Resource Management daemon)

LRMd responsible to hear the instruction from PEngine.

➔ PEngine (Policy Engine)

PEngine used CIB XML file and determine cluster state and recalculate the ideal cluster state based upon unexpected results.

➔ STONITHd (Fencing Daemon)

If any node misbehave, it better to turnoff instead of corrupting the data on shared storage.

COROSYNC:-

Corosync is an open source cluster engine which communicates with multiple cluster nodes and updates the cluster information database (cib.xml) frequently. In previous red hat cluster release, "cman" was responsible for cluster interconnect, messaging and membership capabilities. Pacemaker also supports "heartbeat" which is another open source cluster engine (Not available in RHEL 7), but can be used in RHEL6.

Type of cluster supported by pacemaker:-

➔ Active / Passive Cluster for DR setup

➔ Active / Passive Cluster for Backup setup

➔ Active / Active Cluster

<u>Chapter 3:- Installing Red hat Cluster Software (Corosync / Pacemaker)</u>

Here we would be creating two node cluster and installing package from yum repo from redhat or from your locally created yum repo server.

Hardware requirement:-

1 – 2 CPU

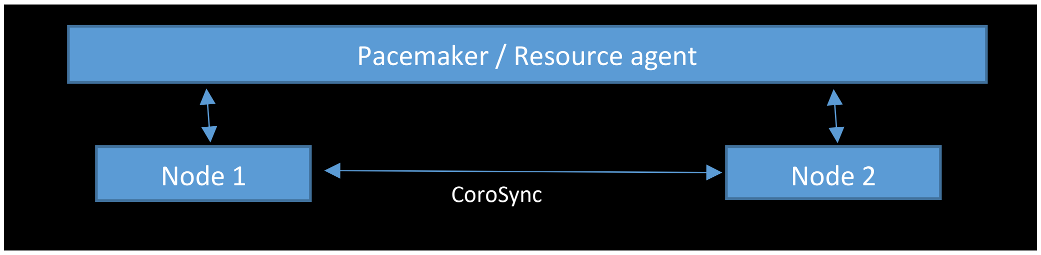
2 – 4 GB Memory

3 – NFS (For shared storage)

Installation:-

```
# yum install pacemaker pcs
```

Stop SELinux and firewalld services for any port blockage.



Enabling and starting services on run time:-

```
# systemctl status pcsd.service
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

Reset default password of "**hacluster**" user.

```
# passwd hacluster
```

<u>Configure Corosync & Create new cluster:</u>

Cluster Autheticate

```
# pcs cluster auth <node1> <nod2>
```

Create Cluster

```
# pcs cluster setup –name <cluster_name>  <node1>  <node2>
```

Cluster status checking

```
# pcs status
```

Start Cluster using pcs command

```
# pcs cluster start --all
```

Verify Corosync communication status

```
# corosync-cfgtool -s
```

Check membership and quorum API's

```
# corosync-cmapctl  | grep members
```

Verify Pacemaker process

```
# ps axf | grep pacemaker
```

Cluster status check

```
# pcs status
```

To view cluster node configuration

```
# pcs cluster cib
```

Verify cluster base information or error

```
# crm_verify –L –V
```

Disable Fencing

```
# pcs property set stonith-enabled=false
# pcs property show stonith-enabled
```

Chapter 4:- Cluster Resource agent

Resource agent plays very important role in cluster resource management. Resource agents are multi thread process that provide logic to manage multiple resource of same type. It mean on resource agent can handle different resource of same agent type.

Pacemaker will have one agent per resource to handle its operation. Resource type could be filesystem, IP address, database, application and more. Resource agent is responsible to monitor, stop, start, migrate, promote, demote resources whenever required. Most of the resource agents are compliant to Open Cluster Framework (OCF).

Let's add one IP resource to the existing cluster and then we will get in to the detailed explanation of command options.

1 - Login to any node using root or hacluster

2 – Check Cluster status

```
# pcs status
```

3 – Add IP which need to be High-available

```
# pcs resource create ClusterIP  ocf:heartbeat:IPaddr2 ip=192.168.XX.XX cidr_nestmask=24  op monitor interval=30s
```

Where,

ClusterIP ➔ Name id IP resource which will have IP running.

ocf:heartbeat:IPaddr2 ➔ Resource Agent Name

Resource Agent =➔ **ocf:heartbeat:IPaddr2**

Where,

ocf is resource standard

heartbeat is resource provider

IPaddr2 is resource agent type

Now let see about Resource Standards:-

The first field “ocr” is standard to which the resource script conform and where to find it. You can see list of resource standard using following command.

```
# pcs resource  standards
```

Common standards are:-

Ocf → Open cluster Framwork

Lsb → Linux standard base

Service → Based on Linux service command

System → systemd based service command

Stonith → Fencing resource standard

Resource Providers:-

The second field when specifying resource agent is resource provider (exp. heartbeat for OCF resource standard). It help the cluster to identify which OCF namespace the resource script is in.

Below command will list you type of resource providers.

```
# pcs resource providers
```

Command providers are:-

heartbeat

openstack

pacemaker

Checking inbuilt resource agents in cluster:-

This is last 3rd parameter in while declaring resource agent. To check available agent for resource use below command.

```
# pcs resource agents ocf:heartbat

# pcs resource agents ocf:openstack

# pcs resource agents ocf:pacemaker
```

Chapter 5:- Pacemaker Cluster Resources and Group Management

To manage n – number of resources pacemaker has concept of resource management and resource group management. In actual environment resources stop and start in specified sequence, this is done by resource group.

Project: - Let create small cluster with 3 resource IP, Filesyestem and Apache.

Filesystem:-

Shared LUN	- /dev/sdb
Volume Group	- webvg01
Volume	- weblv01
FS Directory	- /opt/webhost/web01
FS Type	- ext4

1 - Create filesystem in system:-

```
# pvcreate /dev/sdb
# vgcreate webvg01 /dev/sdb
# lvcreate –L 1G –n weblv01 webvg01
# mkfs.ext4 /dev/webvg01/weblv01
# mkdir /opt/webhost/web01
# mount /dev/webvg01/weblv01 /opt/webhost/web01
```

2 - Apache:-

```
# yum install  httpd –y
```

3 - IP:-

IP addr → 192.168.XX.XX
Subnet → 255.255.255.0

4 - Pre-requisites for LVM:-

You have to change lvm.conf file “use_lvmetad” value to “0”. This is mandatory when you use pacemaker or configure LVM in pacemaker as resource.

```
# grep use_lvmetad /etc/lvm/lvm.conf
```

You can also prevent auto activation of volume group after reboot. This can be done changing parameter “volume_list” in lvm.conf.

Just remove volume group name from list.

```
# grep volumne_list /etc/lvm/lvm.conf
```

You can also rebuilt you initramfs” image to ensure that you volume group wont activate after reboot, use below command. (take backup of previous image)

```
# dracut –H –f /boot/initramfs-$(uname -r).img $(uname -r)
```

5 - Creating LVM Resource:-

Volume Group

```
# pcs resource create vgres01 LVM volgrpname=webvg01 exclusive=true
# pcs resource show vgres01
```

Mount Point:-

# pcs resource create mnt_web01 Filesystem device="/dev/webvg01/weblv01" directory="/opt/webhost/web01" fstype="ext4"

```
# pcs resource show mnt_web01
```

6 - Before adding apache resource please update httpd.conf file for webhosted.

Creating apache resource:-

# **pcs resource create webres01 apache configfile="/etc/httpd/conf/httpd.conf" statusurl="http://92.168.XX.XX/server-status"**

```
# pcs status
```

7- Create resource group to form resource dependency:-

```
# pcs resource group add WEBGroup01 ClusterIP vgres01 mnt_web01 webres01
```

So sequence while startup will be ➔

IP → volume Group → Mount Point → Apache Service

Check Resource status:-

```
# pcs status

# pcs resource
```

[If you find any resource found disable, please enable it using below command]

```
# pcs resource enable vgres01

# pcs resource enable mnt_web01
```

8 - Testing Failover (assume you are on node1)

```
# pcs resource move WEBGroup01 <node2>
```

9 – To Stop start resource services execute

```
# pcs resource disable WEBGroup01

# pcs resource enable WEBGroup01
```

10 – Clear resource error count

```
# pcs resource cleanup
```

Or

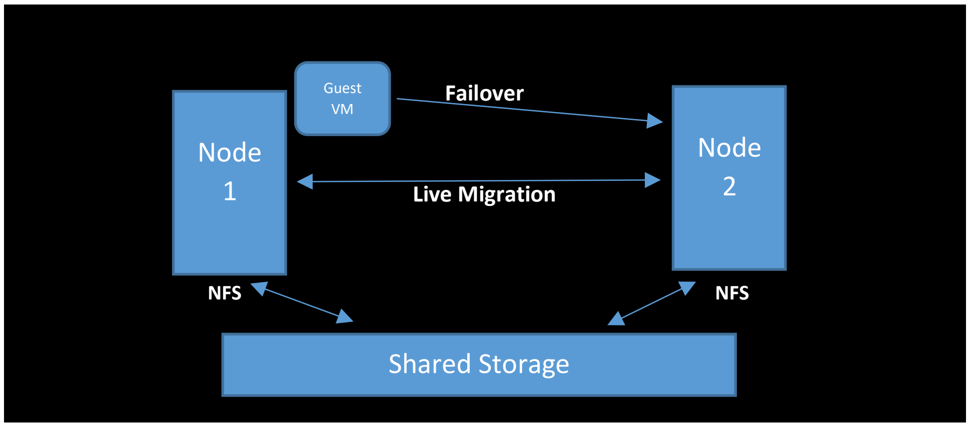
```
# pcs resource clear <resourceName>
```

Project 2:- Pacemaker – Configure HA KVM guest, it should migrate KVM guest VM to another In case f any failure on base node.

Here concept is to provide high availability to KVM guest, like Vmware HA. Migration should be done with minimal downtime from one node to another. Here KVM guest will work like resource in cluster.

Pre-requirements:-

KVM Hypervisor ➔ RHEL6/7
RHEL cluster Node ➔ RHEL6/7 (call it as nod1 / node2)
Shared Storage ➔ NFS (GFS can also be used)
KVM Guest ➔ centos_guest_node1



1 – Login to any node using root privilege

2 – Ensure your KVM configuration is already ready in system. Copy guest domain configuration for (xml) to NFS path.

List GuestVM in KVM.

```
# virsh list --all
```

```
# cd /etc/libvirt/qemu/ ; ls -l

# cp centos_guest_node1.xml  /kvmpool/qemu_config/ ; ls –l
```

Undefined KVM virtual gues

```
# virsh undefined centos_guest_node01

# virsh list --all
```

Verify cluster status

```
# pcs status
```

3 – To manage KVM guest machine we need to use ‘VirtualDomain’ agent resource. Creating resource for KVM guest.

```
# pcs resource create Guest01 VirtualDomain hypervisor=”qemu://system” config=”/kvmpool/qemu_config/centos_guest_node01.xml” migration_transport=ssh op start timeout=120s” op stop timeout=”120s” op monitor timeout=”30” interval=”10” meta allow-migrate=”true” priority=”100” op migrate_from interval=”0” timeout="120s" op migrate_to interval="0" timeout="120" --group KVMGuestVMs
```

```
# pcs status
```

Verify KVM VM using below command:-

```
# virsh list --all
```

Migrate it for testing purpose.

```
# pcs resource move KVMGuestVM node2
```

Perform testing on another node.

Chapter 6:- Cluster Node management

In this chapter we will tell you how to do maintenance activity on cluster nodes.

Project 3:- suppose you have two nodes i.e. node1 and node2 as cluster node and you have to do maintenance on both the nodes with high availability.

Yes, it’s possible to do maintenance activity each node one by one to ensure services running cluster with highly available.

A – One node at a time :-

1 – Check Cluster status and resource

```
# pcs cluster status          🡐 Cluster Status
# pcs resources               🡐 Resource Status
```

2 – If all ok, move Cluster services running from node1 to node2 and bring node1 in maintenance mode(standby).

```
# pcs cluster standby node1
```

As soon as you execute above command services running on node1 will first start migrating on node2.

```
# pcs status
# pcs status corosync         🡐 Cluster membership status
```

OR

```
# corosync-quorumtool
```

3 – Perform maintenance activity and remove node from standby mode.

```
# pcs cluster unstandby node1
# pcs status
```

4 – Stop – Start the cluster services on specific node

```
# pcs cluster stop                 🡐 Stop Cluster service for local machine
# pcs cluster status               🡐 Execute on both node to verify
# pcs cluster start                🡐 Start Cluster service on logged in node
```

Note:- If you don’t want to move resource group automatically, then you can BAN that resource group

```
# pcs resource ban node2 <groupname>
```

B – Both the node need maintenance for some critical software upgrade.

1 – Set the cluster in maintenance mode

```
# pcs property set maintenance-mode=ture          🡐 All resource would be unmapped
# pcs status
```

2 – Verify cluster property

```
# pcs property list
```

3 – Force stop cluster services leaving behind service running.

```
# pcs cluster stop --all           🡐 stop cluster services
# pcs cluster start –all           🡐 start cluster services
```

4 - Once you done with maintenance activity, you can start services and remove maintenance mode

```
# pcs property set maintenance-mode=false
# pcs status
```