**HA Cluster Basics**

A cluster is two or more computers (called nodes or members) that work together to perform a task.

There are four major types of clusters:

- High availability

- Storage

- Load balancing

- High performance

**High availability**

Provides highly available services by eliminating single points of failure and by failing over services from one cluster node to another in case a node becomes inoperative. A high availability cluster must maintain data integrity as one cluster node takes over control of a service from another cluster node. Node failures in a high availability cluster are not visible from clients outside the cluster. The RedHat Enterprise Linux High Availability Add-On provides high availability clustering through its Highly Availability Service Management component, **Pacemaker**. The cluster infrastructure performs the following functions: Cluster management, Lock management, Fencing, Cluster configuration management.

**Quorum**

In order to maintain cluster integrity and availability, cluster systems use a concept known as quorum to prevent data corruption and loss. A cluster has quorum when more than half of the cluster nodes are online. To mitigate the chance of data corruption due to failure, Pacemaker by default stops all resources if the cluster does not have quorum. Quorum is established using a voting system. When a cluster node does not function as it should or loses communication with the rest of the cluster, the majority working nodes can vote to isolate and, if needed, fence the node for servicing.

**Fencing**

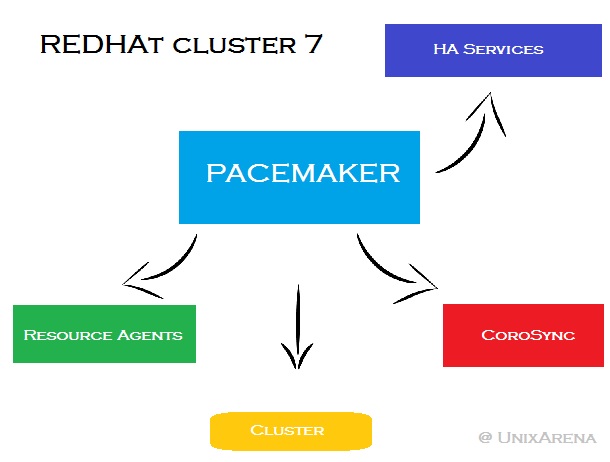
In a cluster system, there can be many nodes working on several pieces of vital production data. Nodes in a busy, multi-node cluster could begin to act erratically or become unavailable, prompting action by administrators. The problems caused by errant cluster nodes can be mitigated by establishing a fencing policy. Fencing is the disconnection of a node from the cluster's shared storage. Fencing cuts off I/O from shared storage, thus ensuring data integrity. The cluster infrastructure performs fencing through the STONITH facility.

**STONITH**

Shoot the Other Node in the Head (STONITH) acts as a cluster resource in Pacemaker that processes fence requests, forcefully powering down nodes and removing them from the cluster to ensure data integrity. It's often deployed in conjunction with a power switch, or an integrated remote management board, or others fence devices, managed by fence agents.

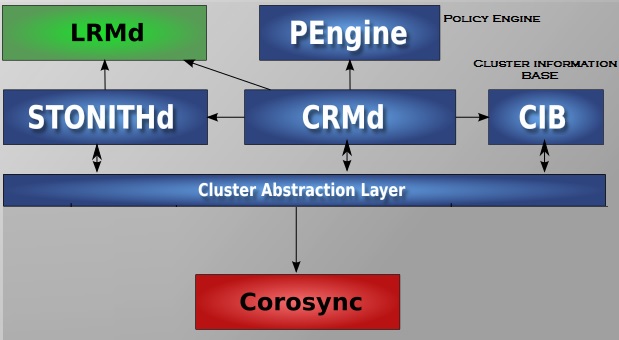
**Redhat Cluster Core Components:**

1. Resource Manager - **Pacemaker** provides the brain that processes and reacts to events regarding the cluster (nodes joining or leaving, failures, maintenance and scheduled activities). Pacemaker reacts moving resources, stopping nodes and even forcing shutdown.
2. Low-level infrastructure - **Corosync** provides reliable messaging, membership and quorum information about the cluster.
3. Resource Agents - **Resource agents** (RA) are nothing but a scripts that start, stop and monitor them.



**Pacemaker’s key components:**

1. Cluster Information Base (**CIB**) It uses XML format file (cib.xml) to represent the cluster configuration and current state of cluster to all the nodes. This file is kept in sync across all the nodes and is used by PEngine to compute ideal state of the cluster and how it should be achieved.
2. Cluster Configuration Management (**PCSd**). The pcsd service manage the configuration of the cluster in CIB, and receives commands by CLI and GUI tools.
3. Cluster Resource Management daemon (**CRMd**) List of instruction will feed to the Designated Controller (DC). Pacemaker centralizes all cluster decision making by electing one of the CRMd instances to act as a master. If one CRMd instance fails, automatically new one will establish.
4. Local Resource Management daemon (**LRMd**) LRMd is responsible to hear the instruction from PEngine.
5. Policy Engine (**PEngine** or PE) PEngine uses the CIB XML file to determine the cluster state and recalculate the ideal cluster state based on the unexpected results.
6. Fencing daemon (**STONITHd**) If any node misbehaves, it better to turned off instead of corrupting the data on shared storage. STONITH offers fencing mechanism in RHEL 7. STONITH is configured in CIB and can be monitored as a normal cluster resource.



**Config files:**

/etc/corosync/corosync.conf - membership and quorum configuration

/var/lib/pacemaker/crm/cib.xml - cluster node and resource configuration.

**Log files:**

/var/log/pacemaker.log

/var/log/cluster/corosync.log

/var/log/pcsd/pcsd.log

/var/log/pcmk\_alert\_file.log

/var/log/messages

### References

#### RedHat documentation:

1. High Availability Add-On Overview <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/high_availability_add-on_overview/>
2. High Availability Add-On Administration Guide <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/high_availability_add-on_administration/>
3. High Availability Add-On Reference Manual <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/high_availability_add-on_reference/>

**CheatSheet**

All operations on the cluster, including installation and configuration, resource management and messagging are made using one command as root

# pcs

where pcs means ‘peacemaker configuration system’.

pcs status - View cluster status.

pcs config - View and manage cluster configuration.

pcs cluster - Configure cluster options and nodes.

pcs resource - Manage cluster resources.

pcs stonith - Manage fence devices.

pcs constraint - Manage resource constraints.

pcs property - Manage pacemaker properties.

pcs node - Manage cluster nodes.

pcs quorum - Manage cluster quorum settings.

pcs alert - Manage pacemaker alerts.

pcs pcsd - Manage pcs daemon.

pcs acl - Manage pacemaker access control lists.

pcs qdevice - Manage quorum device provider on the local host.

pcs booth - Manage booth (cluster ticket manager).

An alternative is using the pcsd Web User Interface to the url:

https://<server-ip>:2224/

with the same features and abilities as the command-line based pcs utility. Login with hacluster user and password

**Cluster Installation and Configuration**

**packages installation:**

# yum install pcs -y

# yum install fence-agents-all -y

# echo CHANGE\_ME | passwd --stdin hacluster

# systemctl start pcsd

# systemctl enable pcsd

**authenticate nodes:**

# pcs cluster auth \

node1.example.com node2.example.com node3.example.com

Username: hacluster

Password:

node2.example.com: Authorized

node1.example.com: Authorized

node3.example.com: Authorized

**create and start new cluster:**

# pcs cluster setup <option> <member> ...

for example

# pcs cluster setup --start --enable --name mycluster \

node1.example.com node2.example.com node3.example.com

**Create a cluster**

pcs cluster setup [--start] [--local] --name \_cluster\_ name\_ \_node1\_ [\_node2\_] [...]

**Enable cluster service (autoboot):**

# pcs cluster enable --all

**Enable cluster service on node[s]**

# pcs cluster enable [--all] [node] [...]

**Disable cluster service on node[s]**

# pcs cluster disable [--all] [node] [...]

**check cluster status:**

# pcs status

# pcs config

# pcs cluster status

# pcs quorum status

# pcs resource show

# crm\_verify -L -V

# crm\_mon (a live monitor for crmd)

**destroy cluster**

# pcs cluster destroy <cluster>

**start/stop cluster**

# pcs cluster start --all

# pcs cluster stop --all

**start/stop node**

# pcs cluster start <node>

# pcs cluster stop <node>

**Start the cluster**

# pcs cluster start [--all] [node] [...]

**Stop the cluster**

# pcs cluster stop [--all] [node] [...]

**Forcebly stop cluster service on a node**

# pcs cluster kill

**Mantainance of a node**

# pcs cluster standby <node>

# pcs cluster unstandby <node>

**Put node in standby**

# pcs cluster standby <node1>

**Remove node from standby**

# pcs cluster unstandby <node1>

**Set cluster property**

# pcs property set <property>=<value>

**disable fencing**

Disabled status is not supported by RedHat. For SOS reports enabled must equal TRUE.

# pcs property set stonith-enabled=false

**Display the configuration in xml style**

# pcs cluster cib

**Display the current status**

# pcs status

**Display the current cluster status**

# pcs cluster status

**Destroy/remove cluster configuration on a node**

# pcs cluster destroy [--all]

**Cluster node authentication**

# pcs cluster auth [node] [...]

**Add a node to cluster**

# pcs cluster node add [node]

**Remove a node to cluster**

# pcs cluster node remove [node]

**RESOURCES**

Resources and services are managed using

# pcs resource

**relocate, enable and disable service groups:**

# pcs resource move <resource>

or

# pcs resource relocate <resource>

to move back

# pcs resource clear <resource>

resource allocation and movement can also be affected by setting up constraints.

# pcs contraint <type> <option>

**Resource creation**

# pcs resource create <resource\_name> <resource\_type> <resource\_options>

To create resources, you need resource agents (RAs). A resource agent is like a service load script.

**list of ocf subclasses (resource type):**

# pcs resource list heartbeat

for example resource type

* ocf:heartbeat:IPaddr2
* ocf:heartbeat:LVM
* ocf:heartbeat:Filesystem
* ocf:heartbeat:oracle
* ocf:heartbeat:apache

**options detail of a resource type or agent:**

# pcs resource describe <resource\_type>

**Ex/ resource creation (vip)**

# pcs resource describe ocf:heartbeat:IPaddr2

# pcs resource create vip\_cluster ocf:heartbeat:IPaddr2 ip=192.168.100.1 --group myservices

# pcs resource create apache-ip ocf:heartbeat:IPaddr2 ip=10.0.2.50 cidr\_netmask=24

# pcs resource show

**Troubleshooting**

# pcs resource debug-start <resource>

# pcs resource debug-stop <resource>

# pcs resource debug-monitor <resource>

# pcs resource failcount show <resource>

**resource update**

update configuration or correct config errors:

# pcs resource update <resource> <options>

**to reset failcount:**

# pcs resource cleanup <resource>

**to move resource offf a node:**

# pcs resource move <resource> [ <node> ]

**to start a resource (or group):**

# pcs resource enable <resource>

**to stop a resource (or group):**

# pcs resource disable <resource>

**GROUPS**

**create group and add a new resource:**

# pcs resource create <resource\_name> <resource\_type> <resource\_options> --group <group>

**delete resource**

# pcs resource delete <resource>

**add a resource in a group**

# pcs resource group add <group> <resource>

# pcs resource group list

# pcs resource list

**CONSTRAINTS**

A constraint is a set of rules that defines how resources (groups) should be loaded.

**add constraint to groups:**

# pcs constraint colocation add apache-group with ftp-group -10000

# pcs constraint order apache-group then ftp-group

If a resource is migrated off from a node using move will enforce constraints -INFINITY on original node. You may have to use pcs resource clear to remove the constraint

**reset constraint on a resource ( or group):**

# pcs resource clear <resource>

**Resource manipulation**

**List Resource Agent (RA) classes**

# pcs resource standards

**List available RAs**

# pcs resource agents ocf

# pcs resource agents lsb

# pcs resource agents service

# pcs resource agents stonith

# pcs resource agents

**Filter by provider**

# pcs resource agents ocf:pacemaker

**List RA info**

# pcs resource describe RA

# pcs resource describe ocf:heartbeat:RA

**Create a resource**

# pcs resource create ClusterIP IPaddr2 ip=192.168.0.120 cidr\_netmask=32

params ip=192.168.122.120 cidr\_netmask=32 op monitor interval=30s

The standard and provider (ocf:heartbeat) are determined automatically since IPaddr2 is unique. The monitor operation is automatically created based on the agent's metadata.

**Delete a resource**

# pcs resource delete resourceid

**Display a resource**

# pcs resource show

# pcs resource show ClusterIP

**Start a resource**

# pcs resource enable ClusterIP

**Stop a resource**

# pcs resource disable ClusterIP

**Remove a resource**

# pcs resource delete ClusterIP

**Modify a resource**

# pcs resource update ClusterIP clusterip\_hash=sourceip

**Delete parameters for a given resource**

# pcs resource update ClusterIP ip=192.168.0.98

**List the current resource defaults**

# pcs resource rsc default

**Set resource defaults**

# pcs resource rsc defaults resource-stickiness=100

**List the current operation defaults**

# pcs resource op defaults

**Set operation defaults**

# pcs resource op defaults timeout=240s

**Set Colocation**

# pcs constraint colocation add ClusterIP with WebSite INFINITY

**With roles**

# pcs constraint colocation add Started AnotherIP with Master WebSite INFINITY

**Set ordering**

# pcs constraint order ClusterIP then WebSite

**With roles:**

# pcs constraint order promote WebSite then start AnotherIP

**Set preferred location**

# pcs constraint location WebSite prefers <node1>=50

**With roles:**

# pcs constraint location WebSite rule role=master 50 \#uname eq <node1>

**Move resources**

# pcs resource move WebSite <node1>

# pcs resource clear WebSite

**A resource can also be moved away from a given node:**

# pcs resource ban Website pcmk-2

Moving a resource sets a stickyness to -INF to a given node until unmoved. Also, pcs deals with constraints differently. These can be manipulated by the command above as well as the following and others

# pcs constraint list --full

# pcs constraint remove cli-ban-Website-on-pcmk-1

**Set a resource failure threshold**

# pcs resource meta RA migration-threshold=3

**Move default resource failure threshold**

# pcs resource meta default migration-threshold=3

**Show a resource failure count**

pcs resource failcount show RA

**Reset a resource failure count**

pcs resource failcount reset RA

**Create a clone**

# pcs resource clone ClusterIP globally-unique=true clone-max=2 clone-node-max=2

**To manage a resource**

pcs resource manage RA

**To UNmanage a resource**

pcs resource unmanage RA

**STONITH**

**ADD FENCING DEVICE**

# ipmitool -H rh7-nodo1-irmc -U admin -P password power on

# fence\_ipmilan --ip=rh7-nodo1-irmc.localdomain --username=admin --password=password --action=status

Status: ON

# pcs stonith

# pcs stonith describe fence\_ipmilan

# pcs stonith create ipmi-fencing1 fence\_ipmilan \

pcmk\_host\_list="rh7-nodo1.localdomain" \

ipaddr=192.168.10.225 \

login=admin passwd=password \

op monitor interval=60s

( pcmk\_host\_list list of machines controlled by this device ) check and enable

# pcs stonith

# pcs property set stonith-enabled=true

# pcs stonith fence pcmk-2

# stonith\_admin --reboot pcmk-2

**Display fencing resources**

# pcs stonith show

pcs treats STONITH devices separately.

**Display Stonith RA info**

# pcs stonith describe fence\_ipmilan

**List available fence agents**

#pcs stonith list

**Add a filter to List available resource agents**

#pcs stonith list <string>

**Setup properties for STONITH**

# pcs property set no-quorum-policy=ignore

# pcs property set stonith-action=poweroff # default is reboot

**Create a fencing device**

#pcs stonith create stonith-rsa-nodeA fence\_rsa action=off ipaddr="nodeA\_rsa" login=<user> passwd=<pass> pcmk\_host\_list=nodeA secure=true

**Display fencing devices**

#pcs stonith show

**Fence a node off**

#pcs stonith fence <node>

**Modify a fencing device**

#pcs stonith update stonithid [options]

**Display a fencing device options**

#pcs stonith describe <stonith\_ra>

**Deleting a fencing device**

#pcs stonith delete stonithid