

# THE OPENSOURCE CLUSTER LAB FOR Pacemaker

CHOI GOOKHYUN

**KOICA**  Korea International  
Cooperation Agency

**GCG** Good Consulting Group Co., Ltd.

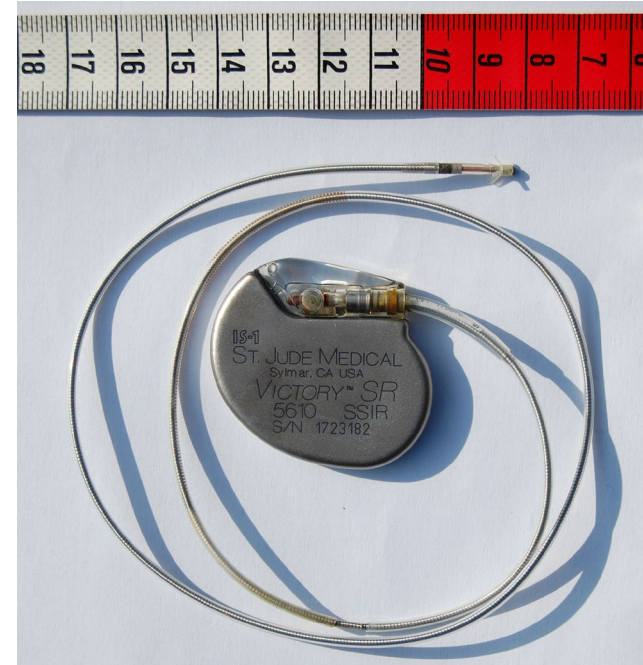
# INTRODUCE



A pacemaker works like an actual pacemaker.

Of course, it is based on Software and Operating systems.

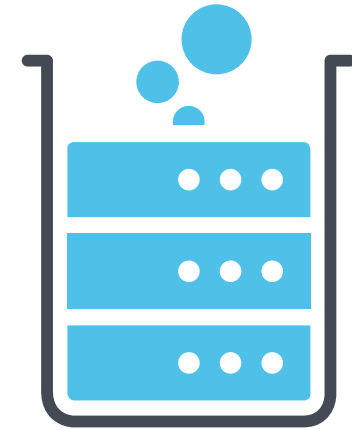
The pacemaker makes it stable for a Service and an Operating System.





The Pacemaker is doing like these.

- Health check to Linux resource and service(**systemD**)
- Fail-Over and Service takeover between host to host
- Small and Lager scale High Availability support



## BACKGROUND



- The oldest and most well-known open-community HA project - providing sophisticated fail over and restart capabilities for Linux
- In existence since 1998; ~ 30k mission-critical clusters in production since 1999
- Active, open development community led by IBM and Novell
- Wide variety of industries; applications supported

## BACKGROUND



- Shipped with most Linux distributions (all but Red Hat)
- No special hardware requirements; no kernel dependencies, all user space
- All releases tested by automated test suite



**Corosync is Cluster Engine.** Group communication System with additional features for implementing high availability within applications.

**\*\*MOST CORE COMPONENT**

<https://clusterlabs.org/corosync.html>

1. Pacemaker
2. DRBD
3. ScanCore



DRBD is Distributed Replicated Storage System. This is helping and implemented as kernel driver, several userspace management application and shell script.

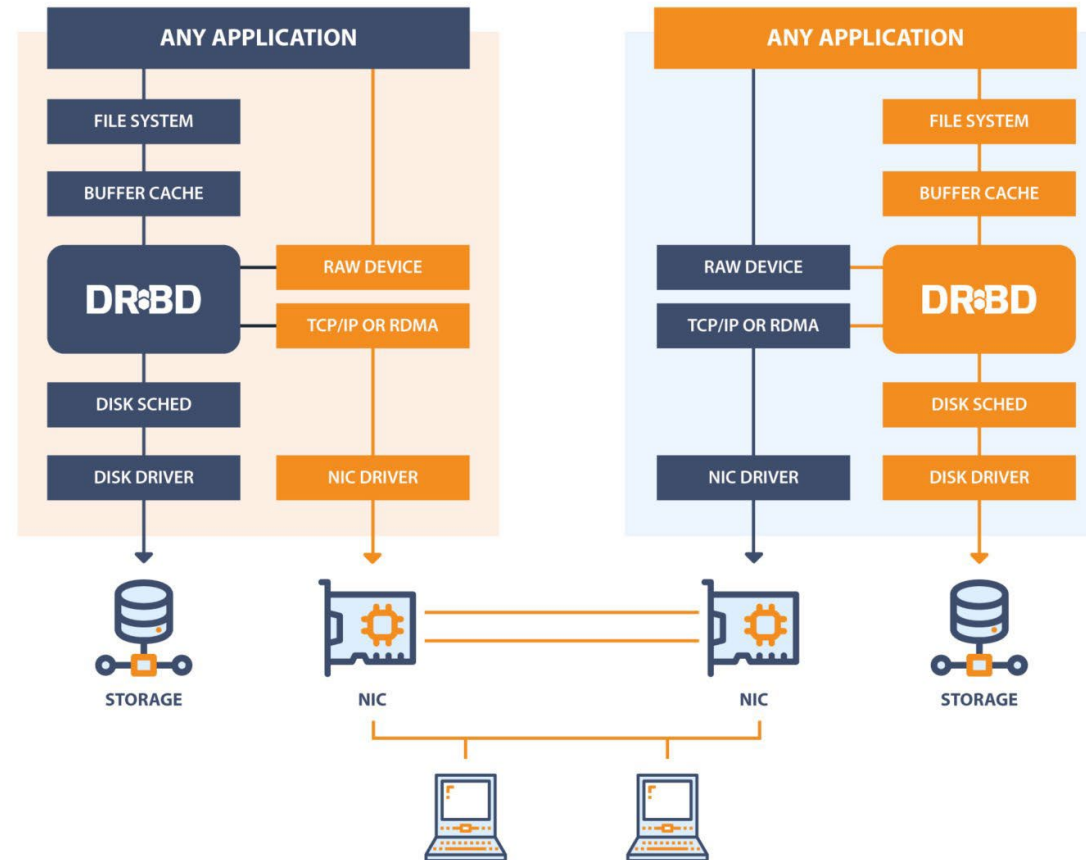
In the Pacemaker, recently adopted this component in Pacemaker system.

<https://linbit.com/drbd/>

But, Don't need to use the DRBD cli command in the Pacemaker.



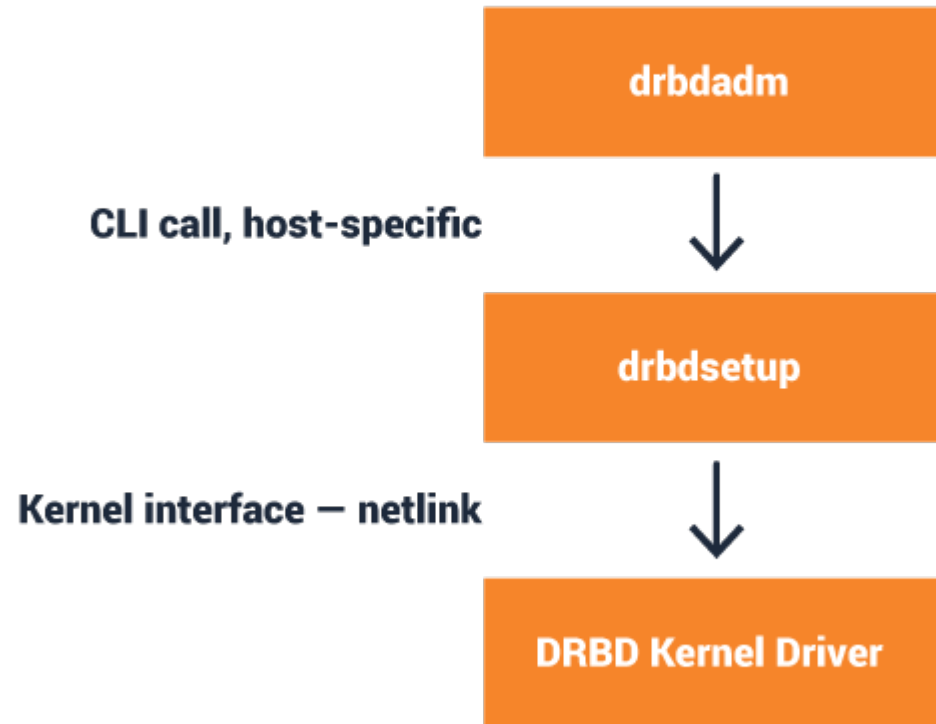
# DRBD



# DRBD



## Configuration File





**ScanCore** is at its core, "decision engine". This component will check to node below condition.

- Over heating
- Loss of input power
- Node Health
- Scan Agents

If you want to know more detail, Please visit this web site.

<https://www.alteeve.com/w/ScanCore>



## Features

---

The ClusterLabs stack, incorporating [Corosync](#) and [Pacemaker](#) defines an [Open Source, High Availability cluster](#) offering suitable for both small and large deployments.

- Detection and recovery of machine and application-level failures
  - Supports practically any [redundancy configuration](#)
  - Supports both [quorate](#) and [resource-driven](#) clusters
  - Configurable [strategies](#) for dealing with [quorum](#) loss (when multiple machines fail)
  - Supports application [startup/shutdown ordering](#), without requiring the applications to run on the same node
  - Supports applications that must or must not run on the [same node](#)
  - Supports applications which need to be active on [multiple nodes](#)
  - Supports applications with dual roles ([promoted and unpromoted](#))
- 
- Provably correct response to any failure or cluster state. The cluster's response to any stimuli can be tested offline *before* the condition exists

# PACEMAKER



## Components

---

A Pacemaker stack is built on five core components:

- libQB - core services (logging, IPC, etc)
- Corosync - Membership, messaging and quorum
- Resource agents - A collection of scripts that interact with the underlying services managed by the cluster
- Fencing agents - A collection of scripts that interact with network power switches and SAN devices to isolate cluster members
- Pacemaker itself

We describe each of these in [more detail](#) as well as other optional components such as CLIs and GUIs.

*"The definitive open-source high-availability stack for the Linux platform builds upon the Pacemaker cluster resource manager."*  
-- *LINUX Journal*, "[Ahead of the Pack: the Pacemaker High-Availability Stack](#)"

# PACEMAKER



## Background

---

Pacemaker has been around since [2004](#) and is primarily a collaborative effort between [Red Hat](#) and [SUSE](#), however we also receive considerable help and support from the folks at [LinBit](#) and the community in general.

Corosync also began life in [2004](#) but was then part of the [OpenAIS project](#). It is primarily a [Red Hat](#) initiative, with considerable help and support from the folks in the community.

The core ClusterLabs team is made up of full-time developers from Australia, Austria, Canada, China, Czech Republic, England, Germany, Sweden and the USA. Contributions to the code or documentation are always welcome.

The ClusterLabs stack ships with most modern enterprise distributions and has been deployed in many critical environments including Deutsche Flugsicherung GmbH ([DFS](#)) which uses Pacemaker to ensure its [air traffic control systems](#) are always available.

# RGMAN VS PACEMAKER



## Under RHEL/CentOS 7

Only Support Resource Manager(RGMAN or CMAN)

## Over RHEL/CentOS 7

Use able to be Pacemaker

# RGMAN VS PACEMAKER



Resource Configuration Management

Resource Management Model

Dependency Models

Event Handling Model

Command-Line Interface Management

Fencing Model

Multi-State Resources

Event Scripts

Maximum Node Count

rgmanager

Manual

Resource Group

Colocation, Start-After

Distributed or Centralized

Status, Control

Assumed

No

Yes

16

Pacemaker

Automatic

Resource-Dependency, Resource Group

User-defined

Centralized

Status, Control, Administration

Flexible

Yes

No

16



# RGMAN VS PACEMAKER



<u>Exclusive Services</u>	Yes	Yes
<u>Failover Domains</u>	Yes	Yes
<u>Resource Exclusion</u>	No	Yes
<u>Time-Based Resource Control</u>	No	Yes
<u>Resource Attribute Inheritance</u>	Yes	Yes
<u>Shared Resources</u>	Yes	Yes
<u>Cloned Resources</u>	No	Yes
<u>Resource Agent APIs</u>	OCF, SysV	OCF, SysV

## RGMAN VS PACEMAKER



Resource Freezing

Yes

Yes

Requires Quorum

Yes

Configurable

Requires DLM

Yes

No

Multi-Partition Resource Management

No

Yes

Non-root Administration

No

Yes

# THE AGENT

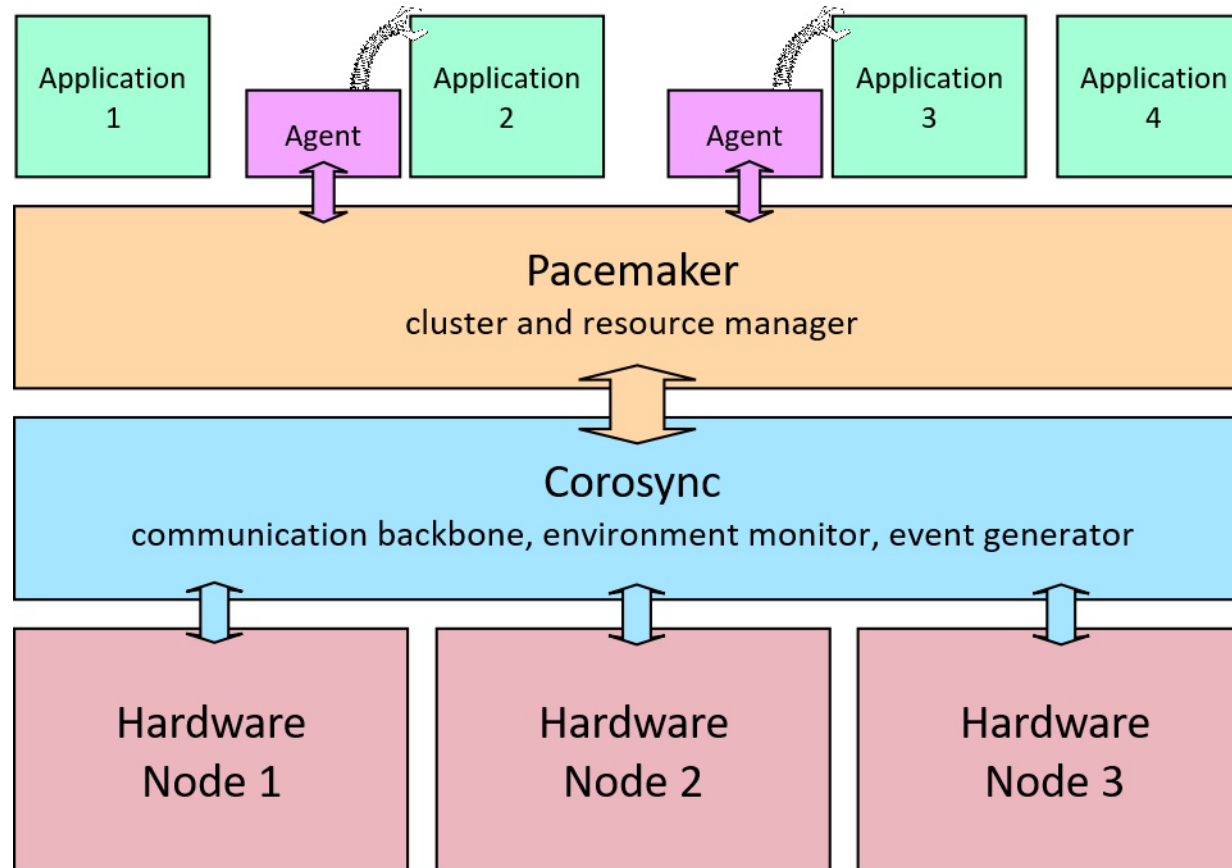


Pacemaker and RGMAN can use OCF Resource Agents.

The OCF Resource Agent can not cover all of the agents features in this training.

<http://www.linux-ha.org/doc/dev-guides/ra-dev-guide.html>

# THE AGENT



## THE AGENT



- H/A, It cannot achieve 100% availability.
- A good HA Cluster systems adds a "9" to your base availability.
- Don't get and drag the complexity into your cluster
  - > Most of time, You will fail the design and High Availability Architecture.

# THE NINES THEORY



99.9999% IN 30SEC

99.999% IN 5MIN

99.99% IN 52 MIN

99.9% IN 9 HOUR

99% IN 3.5 DAY



### DR(DISASTER RECOVERY)

1. FAILOVER IS EXPENSIVE
2. FAILOVER TIMES OFTEN MEASURED IN HOURS
3. UNRELIABLE INTER-NODE COMMUNICATION ASSUMED
4. TOO MUCH COMPLEX AND COMPLICATED DESIGN WITH CLUSTER AND NODES



### HA(HIGH AVAILABILITY)

- FAILOVER IS CHEAP
- FAILOVER TIMES MEASURED IN SECONDS
- RELIABLE INTER-NODE COMMUNICATION
- SIMPLE AND WELL DISGIN WITH CLUSTER AND NODES THROUGH AGENT



# SINGLE POINTS OF FAILURE



## SPOFs

### Good

H/A design eliminates or remove of single point of failure.

### Bad

H/A design is entire system or service can not communicate between nodes.



If the resource is going failure, The fencing guarantees integrity of service

## STONITH

Shoot the Other Node in the Head

## SCSI RELEASE/LOCK AND RESERVE

LVM2, GFS2 or NFS things

## SCSI CHANNEL

iSCSI, Fiber Channel things

## CAPABILITIES



- Supports n-node clusters – where 'n' <= something like 16
- Can use serial, UDP bcast, mcast, ucast comm.
- Fails over on node failure, or on service failure
- Fails over on loss of IP connectivity, or arbitrary criteria
- Active/Passive or full Active/Active
- Built-in resource monitoring
- Support for the OCF resource standard

## CAPABILITIES



- Sophisticated dependency model with rich constraint support (resources, groups, incarnations, master/slave) (needed for SAP)
- XML-based resource configuration
- Configuration and monitoring GUI
- Support for OCFS cluster filesystem
- Multi-state (master/slave) resource support

## CAPABILITIES



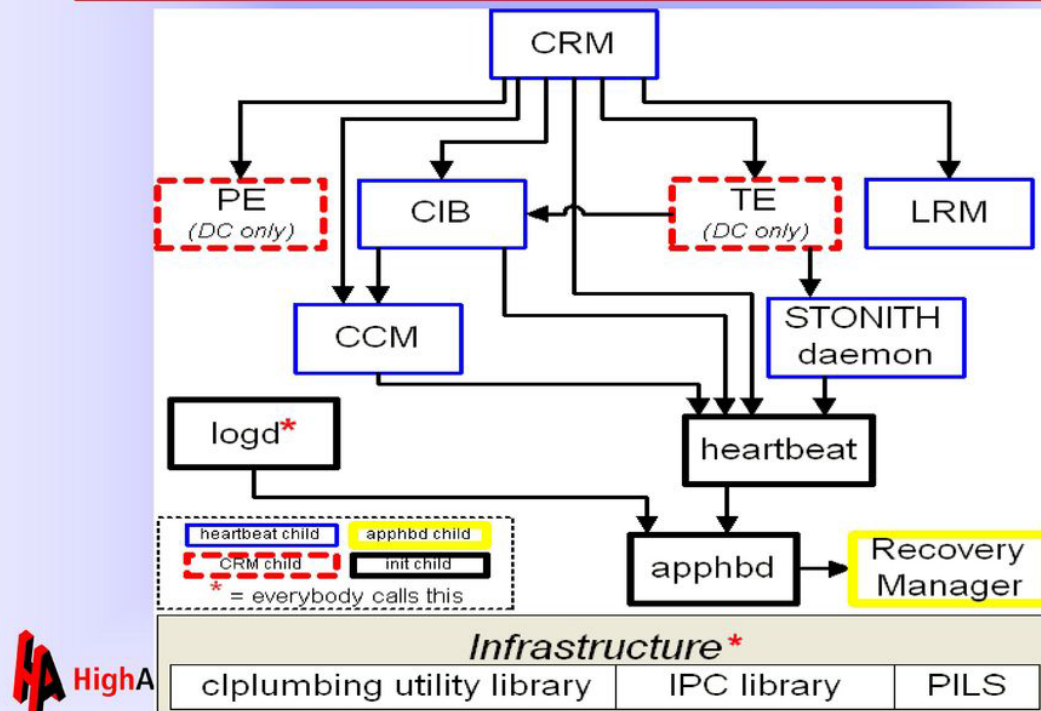
Pacemaker will be support like these.

- NODE
- RESOURCE
- RESOURCE AGENT
- DC(DESIGNATED COORDINATOR), MASTER NODE
- STONITH
- SPLIT BRAIN
- QUORUM

# CAPABILITIES



## Linux-HA Release 2 Architecture



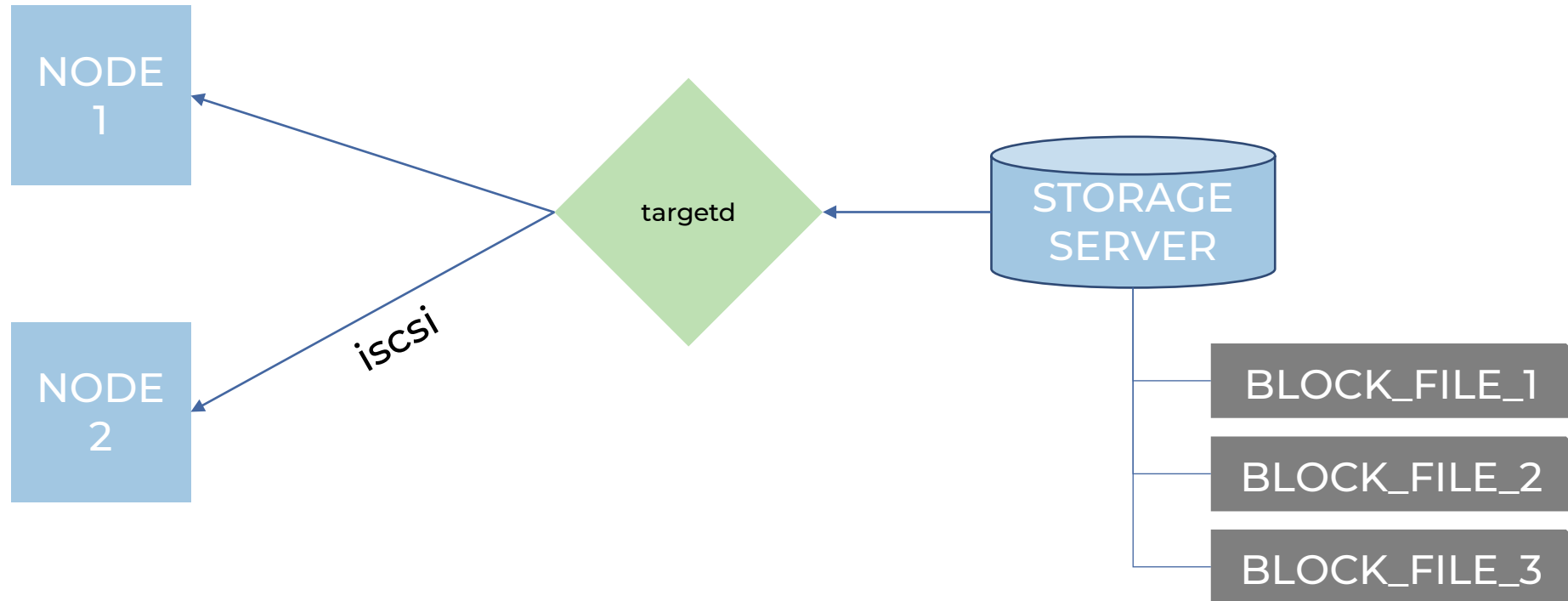
# LAB SETUP

- INSTALL LINUX FOR HOST COMPUTER
- LIBVIRT AND VIRSH COMMAND
- BUILD UP VIRTUAL MACHINE FOR PoC

# FUNDAMENTAL



## LAB DESIGN

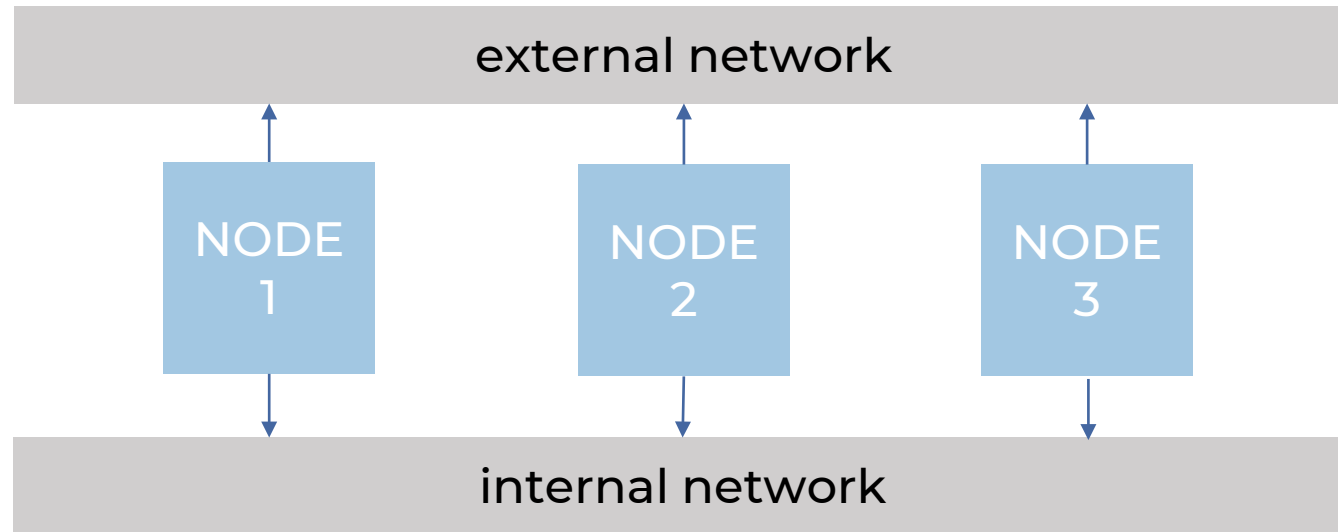




# FUNDAMENTAL



## LAB DESIGN



## LAB



We need those are packages for the lab.

- libvirt
- virsh
- virt-builder

# VIRTUALBMC FOR IPMI



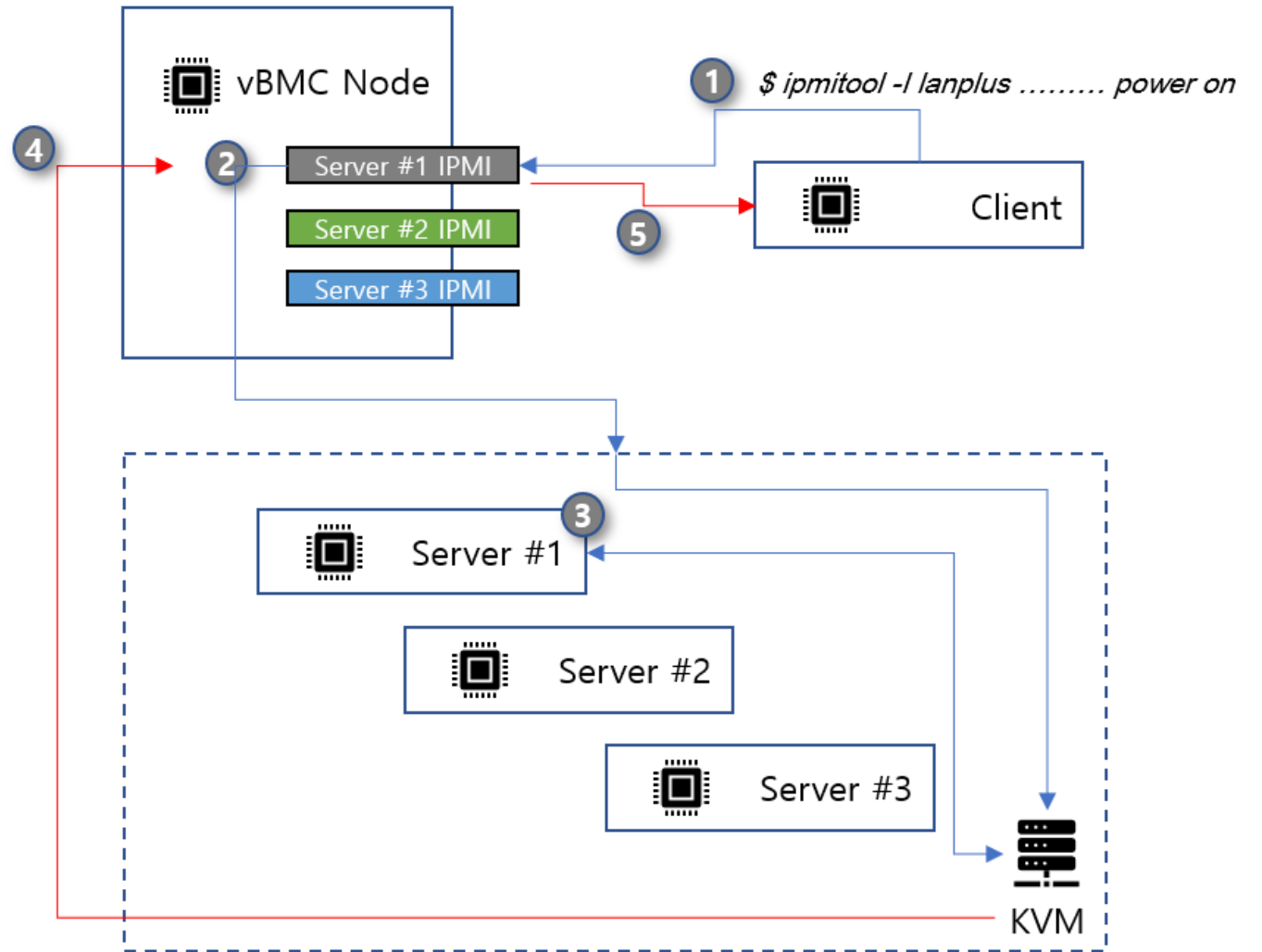
And Maybe need to install for VirtualBMC(For IPMI protocol)

virtualbmc

\*\* The Intelligent Platform Management Interface (IPMI) is a set of computer interface specifications for an autonomous computer subsystem

[https://en.wikipedia.org/wiki/Intelligent\\_Platform\\_Management\\_Interface](https://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface)

# VIRTUALBMC



## VIRTUALBMC FOR IPMI



```
node1# dnf install libvirt libvirt-devel python3-devel gcc  
-y  
node1# pip3 install virtualbmc  
node1# vbmcd  
node1# vbmc add --username centos --password centos --port  
7755 --libvirt-uri qemu+ssh://root@bare/system node2  
node1# vbmc list
```

## BUILD VM AND INSTALL LIBVIRT



```
bare# dnf groupinstall "Virtualization Host" -y
bare# dnf install libguestfs-tools-c -y
bare# virt-builder --list

bare# virsh net-list
bare# cat <<EOF> internal-network.xml
```

# BUILD VM AND INSTALL LIBVIRT



```
<network>
  <name>internal</name>
  <bridge name='virbr10' stp='on' delay='0' />
  <mac address='52:54:00:91:24:b8' />
  <domain name='internal' />
  <ip address="192.168.90.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.90.2" end="192.168.90.254" />
    </dhcp>
  </ip>
</network>
EOF
```

## BUILD VM AND INSTALL LIBVIRT



```
bare# virsh define --file  
bare# virsh net-list
```



## BUILD VM AND INSTALL LIBVIRT



```
bare# virt-builder --size 10G --format qcow2 --root-  
password password:centos -o  
/var/lib/libvirt/images/node1.qcow2 centosstream-8
```

```
bare# virt-builder --size 10G --format qcow2 --root-  
password password:centos -o  
/var/lib/libvirt/images/node2.qcow2 centosstream-8
```

```
bare# virt-builder --size 30G --format qcow2 --root-  
password password:centos -o  
/var/lib/libvirt/images/node3.qcow2 centosstream-8
```

## INSTALL VM ON LIBVIRT BY CLI



```
bare# dnf install virt-install -y
```

```
bare# virt-install --memory 4096 --cpu host-passthrough --vcpu 2 -n  
node1  
--disk /var/lib/libvirt/images/node1.qcow2,cache=none,bus=virtio -w  
network=default,model=virtio -w network=internal,model=virtio --graphics  
none --autostart --noautoconsole --import
```

```
bare# virt-install --memory 4096 --cpu host-passthrough --vcpu 2 -n  
node2 --disk /var/lib/libvirt/images/node2.qcow2,cache=none,bus=virtio  
-w network=default,model=virtio -w network=internal,model=virtio --  
graphics none --autostart --noautoconsole --import
```

## BUILD VM AND INSTALL LIBVIRT



```
bare# virt-install --memory 4096 --cpu host-passthrough --vcpu 2 -n  
node3 --disk /var/lib/libvirt/images/node3.qcow2,cache=none,bus=virtio  
-w network=default,model=virtio -w network=internal,model=virtio --  
graphics none --autostart --noautoconsole --import
```

```
bare# virsh console node1  
bare# virsh console node2  
bare# virsh console node3
```

```
bare# virsh domifaddr node1
```

```
bare# ssh root@<IP>
```

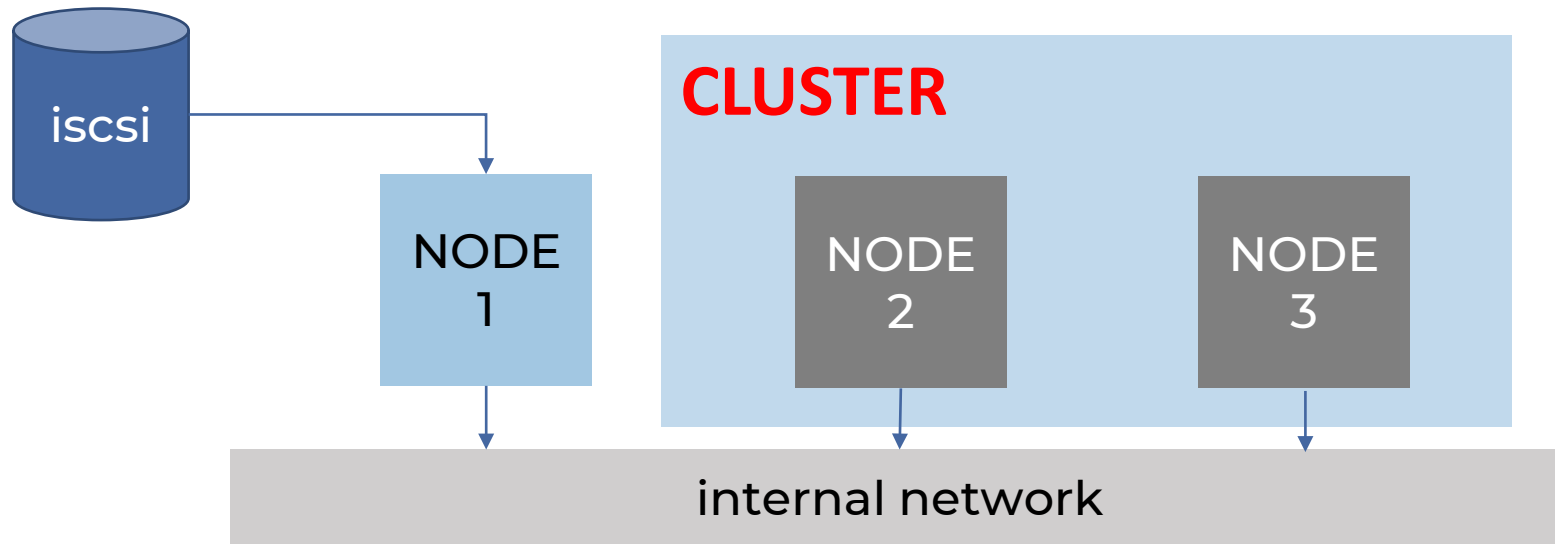
# ISCSI

- CONFIG TO TARGETD SERVER
- FILEIO BLOCKS SETUP

ISCSI



## LAB DESIGN





## STORAGE(TARGET SERVER, node1)

```
node1# dnf install targetd
node1# systemctl enable --now target
node1# firewall-cmd --add-service=iscsi-target
```

```
node1# dnf install iscsi-initiator-utils -y
```

```
node1# mkdir -p /var/lib/iscsi_disks
```

```
node1# targetcli backstores/fileio create iscsi
/var/lib/iscsi_disks/iscsi_disk.img 2G
```

```
node1# targetcli backstores/fileio create nfs /var/lib/iscsi_disks/nfs_disk.img
2G
```

```
node1# targetcli backstores/fileio create gfs2
/var/lib/iscsi_disks/gfs2_disk.img 2G
```

## ISCSI



### STORAGE(TARGET SERVER, node3)

```
node1# targetcli iscsi/ create iqn.2023-02.com.example:blocks
```

```
node1# targetcli iscsi/iqn.2023-02.com.example:blocks/tpg1/luns/ create  
/backstores/fileio/iscsi/
```

```
node1# targetcli iscsi/iqn.2023-02.com.example:blocks/tpg1/luns/ create  
/backstores/fileio/nfs/
```

```
node1# targetcli iscsi/iqn.2023-02.com.example:blocks/tpg1/luns/ create  
/backstores/fileio/gfs2/
```

```
node1# targetcli iscsi/iqn.2023-02.com.example:blocks/tpg1/acls/ create  
iqn.2023-02.com.example.com:node1.init
```

```
node1# targetcli iscsi/iqn.2023-02.com.example:blocks/tpg1/acls/ create  
iqn.2023-02.com.example.com:node2.init
```



## STORAGE(TARGET SERVER, node3)

```
node1# targetcli iscsi/iqn.2023-02.com.example:blocks/tpg1/acls/iqn.2023-02.com.example.com:node1.init  
set auth userid=username
```

```
node1# targetcli iscsi/iqn.2023-02.com.example:blocks/tpg1/acls/iqn.2023-02.com.example.com:node1.init  
set auth password=username
```

```
node1# targetcli iscsi/iqn.2023-02.com.example:blocks/tpg1/acls/iqn.2023-02.com.example.com:node2.init  
set auth userid=username
```

```
node1# targetcli iscsi/iqn.2023-02.com.example:blocks/tpg1/acls/iqn.2023-02.com.example.com:node2.init  
set auth password=password
```



# ISCSI



## STORAGE(TARGET SERVER, node3)

```
# iscsadm -m discoverydb -t sendtargets -p 192.168.100.130  
# iscsadm -m node --login  
# iscsadm -m session --debug 3  
# iscsadm -m session --rescan
```

ISCSI



**DO SELF LAB**

# **ROLL BACK VIRTUAL MACHINE AND REINSTALL TO THE PACEMAKER CLUSTER**

# PACEMAKER INSTALLATION

- LINUX CONFIGURE AND INSTALLATION
- ISCSI TARGET SERVER CONFIGURATION



## SNAPSHOT

before start this lab, we have to make a snapshot for all virtual machines.

```
bare# virsh snapshot-create as --domain node1 --name node1-pcs-setup
```

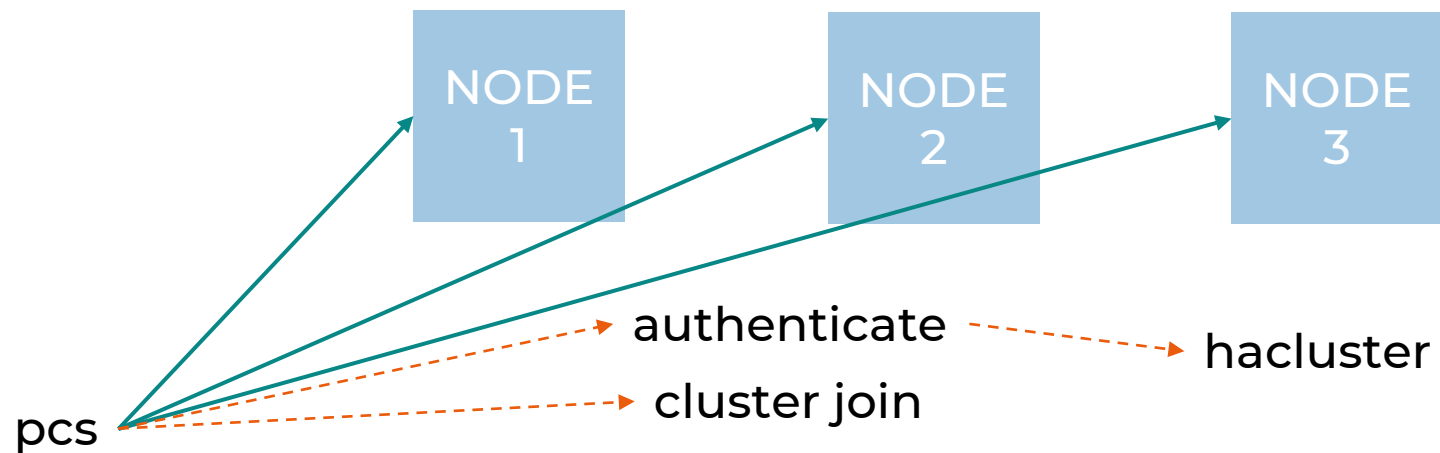
```
bare# virsh snapshot-create-as --domain node2 --name node2-pcs-setup
```

```
bare# virsh snapshot-create-as --domain node3 --name node3-pcs-setup
```

```
bare# virsh snapshot-list node1
```

```
bare# virsh snapshot-revert --domain node1 --snapshotname node1-pcs-setup --running
```

## DESIGN





## NETWORK CONFIG

```
node1# nmcli con add con-name eth1 ipv4.addresses 192.168.90.110/24 ipv4.never-  
default yes method manual autoconnect yes type ethernet ifname eth1  
node1# nmcli con up eth1
```

```
node2# nmcli con add con-name eth1 ipv4.addresses 192.168.90.120/24 ipv4.never-  
default yes method manual autoconnect yes type ethernet ifname eth1  
node2# nmcli con up eth1
```

```
node3# nmcli con add con-name eth1 ipv4.addresses 192.168.90.130/24 ipv4.never-  
default yes method manual autoconnect yes type ethernet ifname eth1  
node3# nmcli con up eth1
```



## PCS

need to set a hostname to each of node.

```
node1# hostnamectl set-hostname node1.example.com
node2# hostnamectl set-hostname node2.example.com
node3# hostnamectl set-hostname node3.example.com
```

Here no have a DNS server. /etc/hosts configure to the A recode instead DNS.

Copy this file to each hosts.

```
# cat <<EOF>> /etc/hosts
192.168.90.110 node1.example.com node1
192.168.90.120 node2.example.com node2
192.168.90.130 node3.example.com node3 storage
EOF
```



## PCS

Make a SSH Private and Public key for access to nodes without password

```
node1# ssh-keygen -t rsa -N'' -f ~/.ssh/id_rsa
node1# dnf install sshpass -y
```





## PCS

**Before install to Pacemaker, we have to update whole nodes as latest packages. Then install to pacemaker package.**

```
# cat <<EOF> ~/.ssh/config
StrictHostKeyChecking=no
EOF
```

```
# for i in node{1..3} ; do sshpass -pcentos ssh root@$i 'dnf update -y' ; done
# for i in node{1..3} ; do sshpass -pcentos scp /etc/hosts root@\$i.example.com:/etc/hosts ; done
```

```
# for i in node{1..3} ; do sshpass -p centos ssh root@$i 'dnf --enablerepo=ha -y install pacemaker pcs
' ; done
# for i in node{1..3} ; do sshpass -p centos ssh root@$i 'dnf install firewalld && systemctl enable --
now firewalld' ; done
```



## PCS

```
# for i in {1..3} ; do sshpass -p centos ssh root@node${i} 'firewall-cmd --add-service=high-availability && firewall-cmd --runtime-to-permanent' ; done
```

```
# for i in {1..3} ; do sshpass -p centos ssh root@node$i 'echo centos | passwd --stdin hacluster' && systemctl enable --now pcs.service ; done
```

```
# ping node1 -c3
```

```
# ping node2 -c3
```

```
# ping node3 -c3
```



## PCS(node1)

```
# pcs host auth -u ha_cluster_lab -p centos node1.example.com node2.example.com  
node3.example.com  
# pcs cluster setup ha_cluster_lab node1.example.com node2.example.com node3.example.com  
  
# pcs cluster start --all  
# pcs cluster enable --all  
# pcs cluster status  
  
# pcs status corosync  
# pcs cluster stop --all  
# pcs cluster destroy --all  
  
# ss -npltu | grep -i corosync
```



## ADD NODE TO THE CLUSTER

If you want to try to add new node into the existing cluster, Please do run these commands in the lab.

```
bare# virt-builder --size 30G --format qcow2 -o --root-password password:centos  
/var/lib/libvirt/images/node4.qcow2 centosstream-8  
bare# virt-install --memory 4096 --vcpu 2 -n node4 \  
--disk /var/lib/libvirt/images/node4.qcow2,cache=none,bus=virtio \  
-w network=default,model=virtio -w network=internal,model=virtio \  
--graphics none --autostart --noautoconsole --import  
  
# pcs cluster auth -u hacluster -p centos node4  
# pcs cluster node add node4 --start --enable  
# pcs cluster start node4  
# pcs cluster enable node4
```

**PACEMAKER**

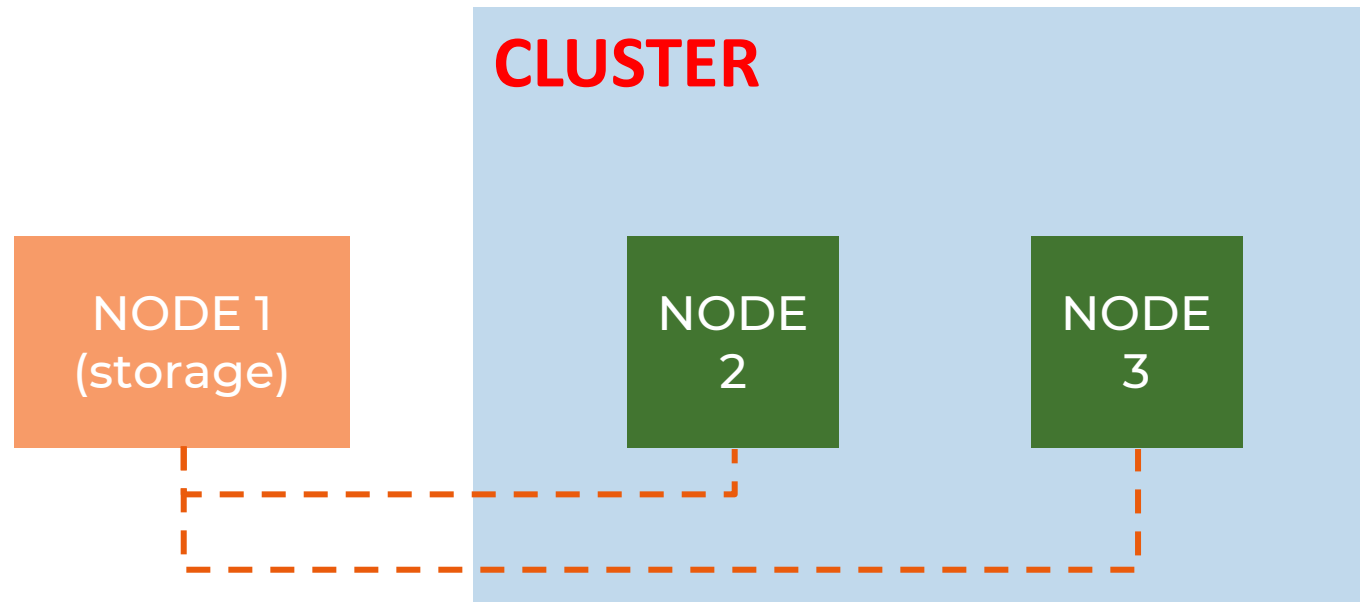


**DO SELF LAB**

# **ROLL BACK VIRTUAL MACHINE AND REINSTALL TO THE PACEMAKER CLUSTER**

# PACEMAKER CLUSTER

- CLUSTER CREATE
- CLUSTER VERIFY





## CLUSTER

```
node2/3# dnf --enablerepo=ha -y install pacemaker pcs
```

```
node2/3# systemctl enable --now pcsd
```

```
node2/3# echo centos | passwd --stdin hacluster
```

```
node2/3# firewall-cmd --add-service=high-availability --permanent
```

```
node2/3# firewall-cmd --reload
```

```
node2# pcs host auth -u hacluster -p centos node2.example.com node3.example.com
```

```
node2# pcs cluster setup ha_cluster_lab node2.example.com node3.example.com
```

```
node2# pcs cluster start --all
```

```
node2# pcs cluster enable --all
```

```
node2# pcs cluster status
```



**PACEMAKER**



## CLUSTER

```
node2# pcs status corosync  
node2# corosync-cfgtools -s
```



## CLUSTER

```
node2# dnf --enablerepo=ha -y install fence-agents-scsi
```

```
node2# ls /dev/disk/by-id
```

```
node2# pcs stonith create scsi-shooter fence_scsi pcmk_host_list="node2.example.com node3.example.com" devices=/dev/disk/by-id/wwn-<ID> meta provides=unfencing
```

```
node2# pcs stonith config scsi-shooter
```

```
node2# pcs status
```

```
node3# pcs status
```

```
node3# pcs stonith fence node2.example.com
```

```
node3# pcs status
```

```
node2# reboot
```

**PACEMAKER**



**DO SELF LAB**

# **ROLL BACK VIRTUAL MACHINE AND REINSTALL TO THE PACEMAKER CLUSTER**

# FUNDAMENTAL

- PACEMAKER RESOURCE
- BASIC COMMANDS



## ACL

Access Control List, This is will be disabled as default value. Pacemaker uses "**root**" or "**hacluster**" group with user account for modification CIB information.

If you want to use normal user as Pacemaker grained of authorization, use to the **ACLs**.

**CIB: Cluster Information Base. The CIB file is normally configured by XML.**

## ACL



```
# adduser rouser
# echo centos | passwd --stdin rouser
# usermod -aG haclient rouser
# pcs acl enable
# pcs acl role create read-only description="Read only access to
cluster" read xpath /cib
# pcs acl user create rouser read-only
# pcs acl
# pcs client local-auth
```



## alert

Pacemaker can support alert agents via script.  
"/usr/share/pacemaker/alerts" in the location have a sample alert agents.

## ALTER



```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_file.sh.sample  
/var/lib/pacemaker/alert_file.sh  
# touch /var/log/pcmkn_alert_file.log  
# chown hacluster:haclient /var/log/pcmkn_alert_file.log  
# chmod 600 /var/log/pcmkn_alert_file.log  
# pcs alert create id=alert_file description="Log events to a file."  
path=/var/lib/pacemaker/alert_file.sh  
# pcs alert recipient add alert_file id=my-alert_logfile  
value=/var/log/pcmkn_alert_file.log  
  
# pcs alert
```



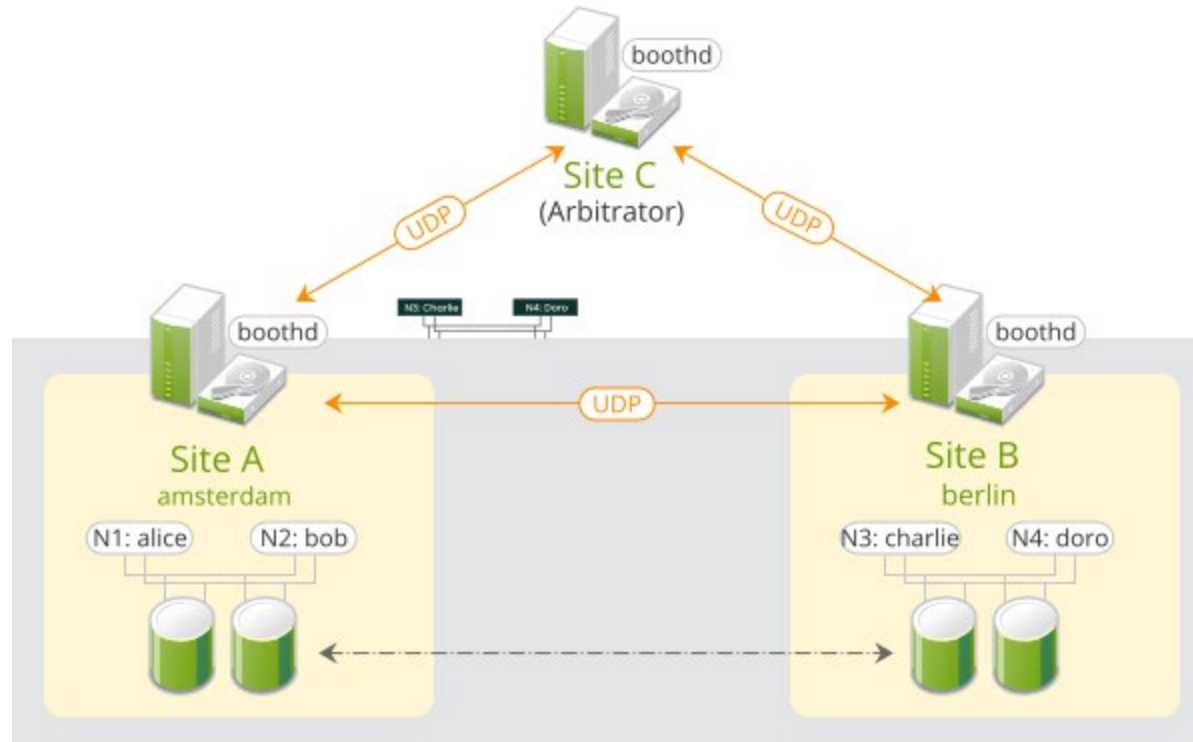


## booth

Different place and cluster are in two site, the arbitrator cluster(or overlay cluster) is managed by the booth cluster. So, In Simple way, the booth cluster role is "Arbitrator"

In this training not cover this part.

# BOOTH



<https://documentation.suse.com/sle-ha/12-SP4/single-html/SLE-HA-geo-guide/index.html>



Different place and cluster are in two site, the arbitrator cluster(or overlay cluster) is managed by the booth cluster.

So, In Simple way, the booth cluster role is  
**"Arbitrator"**



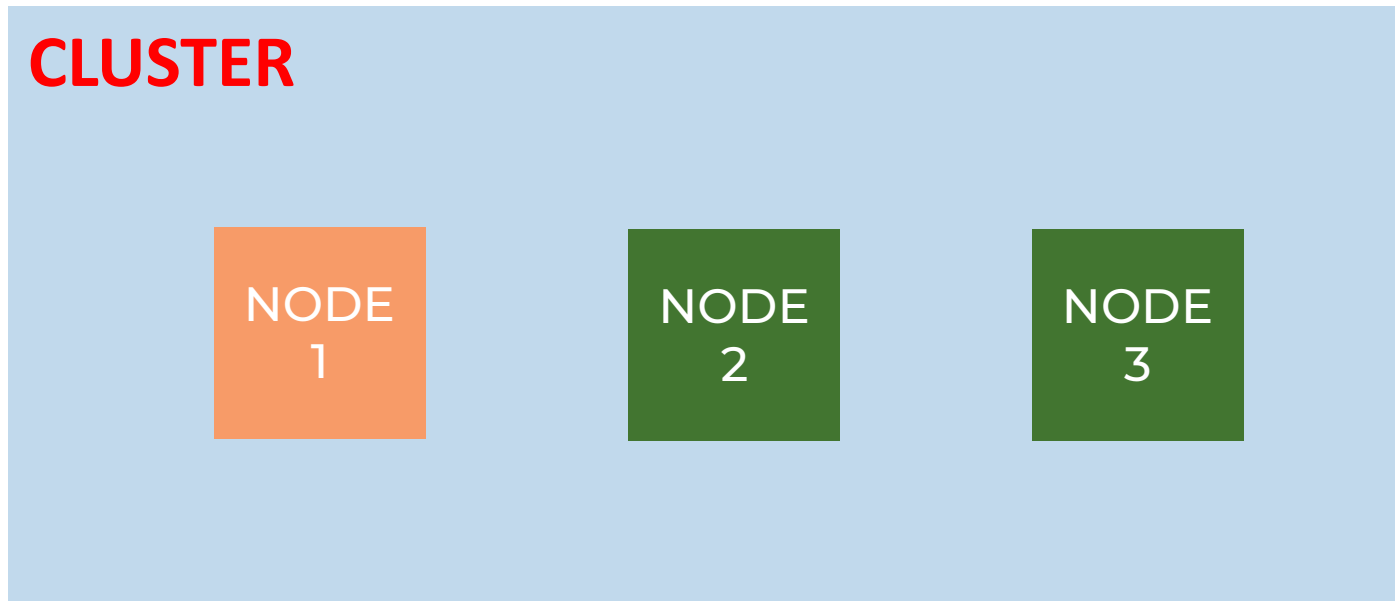
Different place and cluster are in two site, the arbitrator cluster(or overlay cluster) is managed by the booth cluster.

So, In Simple way, the booth cluster role is "Arbitrator"

# CLUSTER



The cluster is group of nodes. In the pacemaker needs at least two nodes in the cluster.



## CLUSTER COMMAND



```
# pcs cluster status  
# pcs cluster config  
# pcs cluster auth  
# pcs cluster enable  
# pcs cluster start
```

## CONFIG/COMMAND



Shows the cluster all configuration about "resource", "stonith", "fence" and OCF agent configuration.

```
# pcs config
```



Set to location for the cluster resource. You can determine the behavior of a resource in a cluster by configuration.

### location

The location is a node. A resource run the location node by configure.

### order

Order is constraint determines the order in which a resources run in ordering.





## colocation

colocation is where resource will locate or placed relative to other resource

DR



This is Disaster recovery for a high availability cluster when primary cluster is not available. This feature can use **RHEL/CentOS 8** version.

## DR COMMAND



```
# pcs host auth -uhacluster -phacluster node1.example.com  
node2.example.com node3.example.com node4.example.com  
# pcs cluster setup node1.example.com node2.example.com  
node3.example.com node4.example.com  
# pcs cluster setup DRsite node3.example.com node4.example.com --start  
# pcs dr set-recovery-site node3.example.com  
# pcs dr config  
# pcs dr status
```

## HOST



In the cluster concept, the cluster have a node at least one. Pacemaker can add a host by 'pcs host' command. Each host need to authenticate with 'hacluster' user account.

## NODE



node is minimum element for cluster. the cluster have at least one node. the node will have resource, constraint with resource group and order.


# NODE COMMAND



```
# pcs cluster status  
# pcs node standby node1.example.com  
# pcs cluster status
```



pcs is Pacemaker configuration daemon for GUI and remote cli. The web GUI web administrator page access via 2224/TPC port.

 HIGH AVAILABILITY  
MANAGEMENT

hacluster ▼

MANAGE CLUSTERS

PERMISSIONS

MANAGE CLUSTERS

[x Remove](#)

[+ Add Existing](#)

[x Destroy](#)

[+ Create New](#)

All	✓	!	!	x
0	0	0	0	0

INFORMATION ABOUT CLUSTERS

Select a cluster to view more detailed cluster information

## PROPERTY



The property is configuration to cluster value such as corosync, quorum values. Cluster properties control how the cluster behavior with property value.



# PROPERTY COMMAND



```
# pcs property
```

```
Cluster Properties:
```

```
cluster-infrastructure: corosync
```

```
cluster-name: ha_cluster_lab
```

```
dc-version: 2.1.5-5.el8-a3f44794f94
```

```
have-watchdog: false
```

```
no-quorum-policy: freeze
```

```
# pcs property set maintenance-mode=true
```

# PROPERTY COMMAND



## # pcs property

### Cluster Properties:

cluster-infrastructure: corosync

cluster-name: ha\_cluster\_lab

dc-version: 2.1.5-5.el8-a3f44794f94

have-watchdog: false

**maintenance-mode: true**

no-quorum-policy: freeze



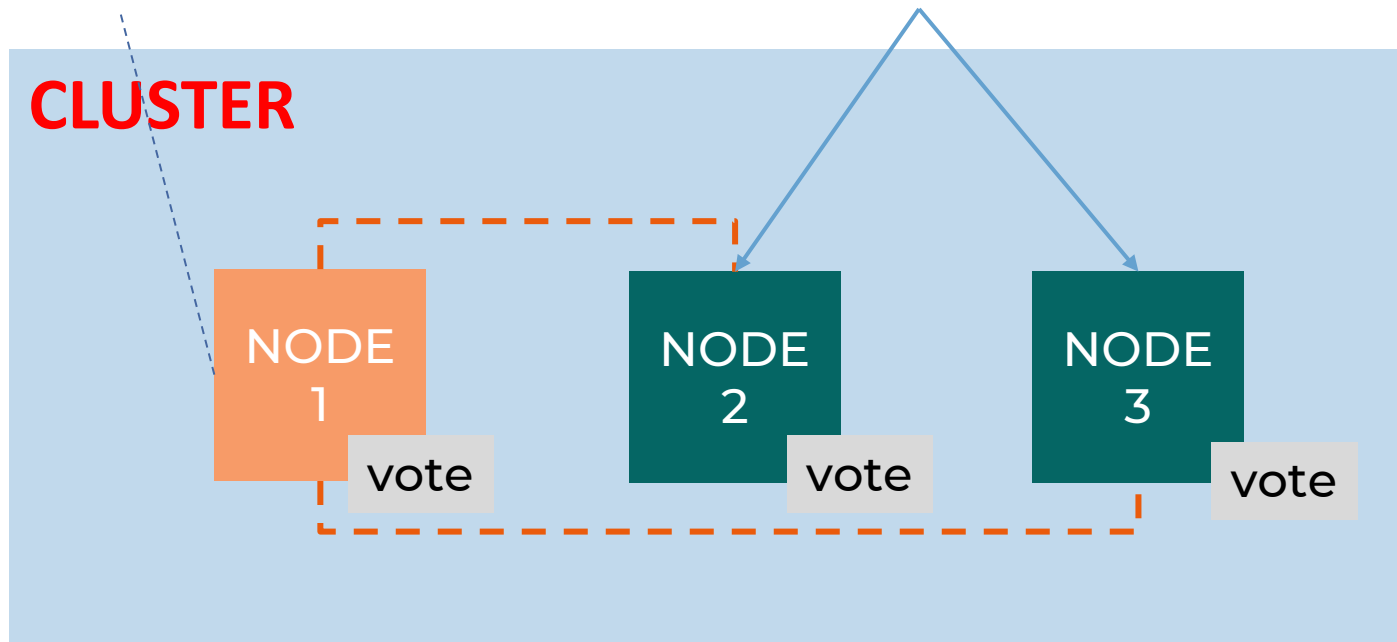
"qdevice" is "**quorum device**". this is not device and agent. It's third-party arbitration device for the cluster. Normally a quorum device is recommended for clusters with an even number of nodes. But, when the cluster is going to be two-node cluster status, the quorum device is better for survive in split-brain situation.

# QDEVICE

```
# pcs qdevice status net --full
```

brain split

**CLUSTER**



# QDEVICE COMMAND



**Install these packages form High Availability repository.**

```
node1# dnf --enablerepo=ha install corosync-qdevice
node1# dnf --enablerepo=ha install pcs corosync-qnetd
```

```
node1# yum install pcs corosync-qnetd
node1# systemctl enable --now pcsd.service
node1# pcs qdevice setup model net --enable --start
```

```
node1# pcs qdevice status net --full
node1# firewall-cmd --permanent --add-service=high-availability
node2# pcs cluster auth qdevice
```

## QDEVICE COMMAND



```
node2# pcs quorum config
node2# pcs quorum status
node2# pcs quorum device add model net host=qdevice
algorithm=ffsplit
node2# pcs quorum config
node2# pcs quorum status
node2# pcs quorum device status
node2# pcs qdevice status net --full
```

## QDEVICE COMMAND



```
node2# pcs qdevice start net
node2# pcs qdeivce stop net
node2# pcs qdevice enable net
node2# pcs qdevice disable net
node2# pcs qdevice kill net
```

## QDEVICE COMMAND



```
node2# pcs quorum device update model algorithm=lms
node2# pcs quorum device remove
node2# pcs quorum device status
node2# pcs qdevice destroy net
```



# QUORUM

- quorum
- The quorum is voting system for cluster nodes.
- Every cluster nodes has a vote for vote-quorum system. If some resources or nodes can't vote the object will be fencing and detached from system.



**"Well, let's get started now we've got a quorum."**

## QUORUM COMMAND



```
node2# pcs quorum status
Quorum information
```

-----

```
Date:                Sun Feb 26 02:09:16 2023
Quorum provider:     corosync_votequorum
Nodes:               2
Node ID:              1
Ring ID:              1.40
Quorate:              Yes
```

# QUORUM COMMAND



## Votequorum information

-----

Expected votes: 2  
Highest expected: 2  
Total votes: 2  
Quorum: 1  
Flags: 2Node Quorate WaitForAll

## Membership information

-----

Nodeid	Votes	Qdevice Name
1	1	NR node2.example.com (local)
2	1	NR node3.example.com

## RESOURCE



The resource is a service managed by Peacemaker. The resource is kind of agent for standardized interface for managing the service. This allows Pacemaker to be agnostic the services it manages.

We don't need to understand about the service works behind of resource agent.

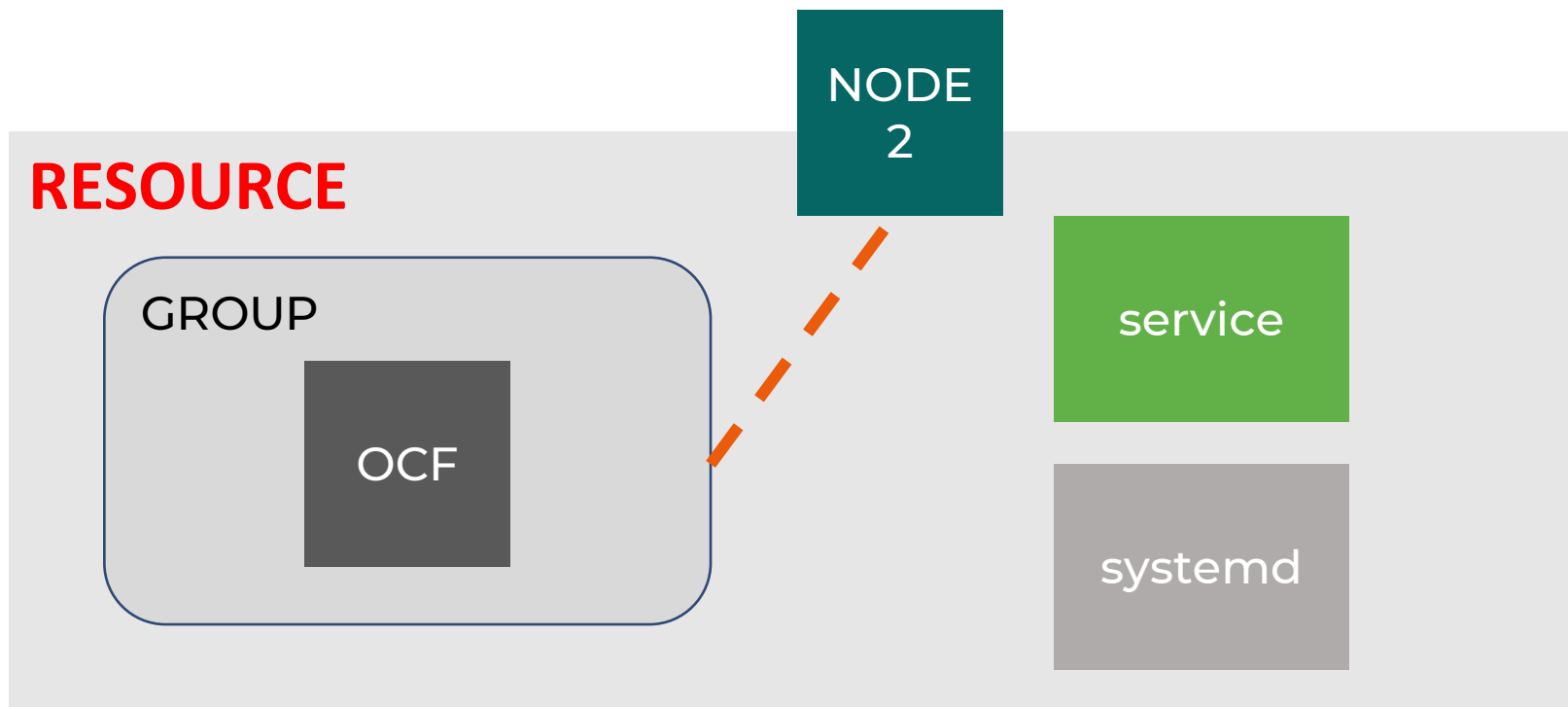
## RESOURCE



```
nodeX# ls /usr/share/resource-agents/ocft/configs
```

```
apache      exportfs-multidir  IPAddr2v6  jboss  mysql-  
proxy  pgsql      SendArp      Xen  
db2      Filesystem      IPsrcaddr  LVM      named  
portblock  sg_persist      Xinetd  
drbd.linbit  IPAddr2      IPv6addr  MailTo  
nfsserver  postfix      tomcat  
exportfs      IPAddr2v4      iscsi      mysql      oracle  
Raid1      VirtualDomain
```

# RESOURCE



## RESOURCE



The resources have a classes below these.

- OCF
- LSB
- system
- Upstart(deprecated)
- Service
- Fencing
- Nagios



# The Open Cluster Framework

The Open Cluster Framework (OCF) Resource Agent API is a ClusterLabs standard for managing services. It is the most preferred since it is specifically designed for use in a Pacemaker cluster.





OCF agents are scripts that support a variety of actions including **start**, **stop**, and **monitor**. They may accept parameters, making them more flexible than other classes. The number and purpose of parameters is left to the agent, which advertises them via the meta-data action.

Unlike other classes, OCF agents have a provider as well as a class and type.



Most Linux distributions use Systemd for system initialization and service management. Unit files specify how to manage services and are usually provided by the distribution.

Pacemaker can manage systemd services. Simply create a resource with systemd as the resource class and the unit file name as the resource type. Do not run `systemctl enable` on the unit.



# Linux Standard Base

LSB resource agents, also known as SysV-style, are scripts that provide start, stop, and status actions for a service.

They are provided by some operating system distributions. If a full path is not given, they are assumed to be located in a directory specified when your Pacemaker software was built (usually /etc/init.d).

In order to be used with Pacemaker, they must conform to the LSB specification as it relates to init scripts.



**The Stonith** class is used for managing fencing devices, discussed later in Fencing.

## STATUS



The status command will show the cluster state.  
The status collect from pcsd, corosync and agent information.

## STATUS COMMAND



```
# pcs status  
# pcs resource status <NAME>  
# pcs status nodes
```



"Shoot the other node in the head" aka fencing. The Stonith for protects your data from being corrupted by rogue nodes.

The command example will not work correctly.

# STONITH COMMAND



```
node2# pcs stonith list
node2# dnf search fence-agents-all
node2# dnf install fence-agents-ipmilan
node2# pcs stonith describe fence_ipmilan
node2# pcs stonith create ipmi-fence-node1 fence_ipmilan
pcmk_host_list="node1" ipaddr="10.0.0.1" login="xxx" passwd="xxx"
lanplus=1 power_wait=4
```



# STONITH COMMAND

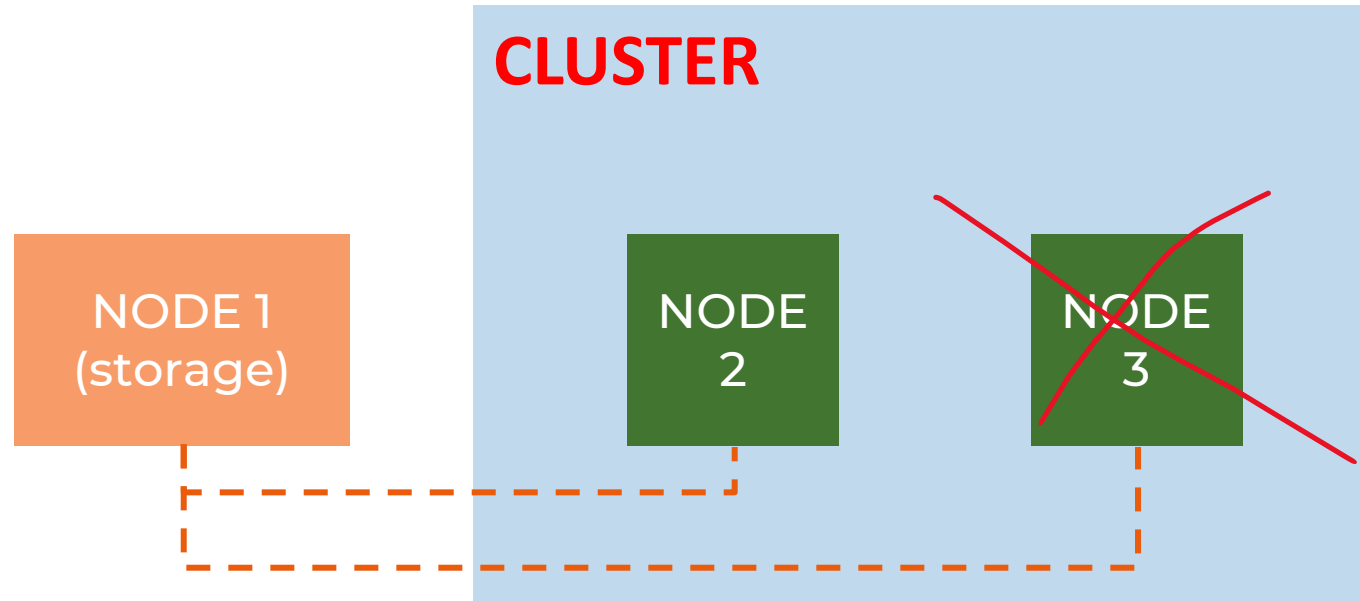


```
# pcs -f stonith_cfg stonith
# pcs -f stonith_cfg property set stonith-enabled=true
# pcs -f stonith_cfg property
# pcs cluster stop node2
# stonith_admin --reboot node2
```

# FENCE

- FENCE TEST WITH ISCIS

# FENCE



## FENCE

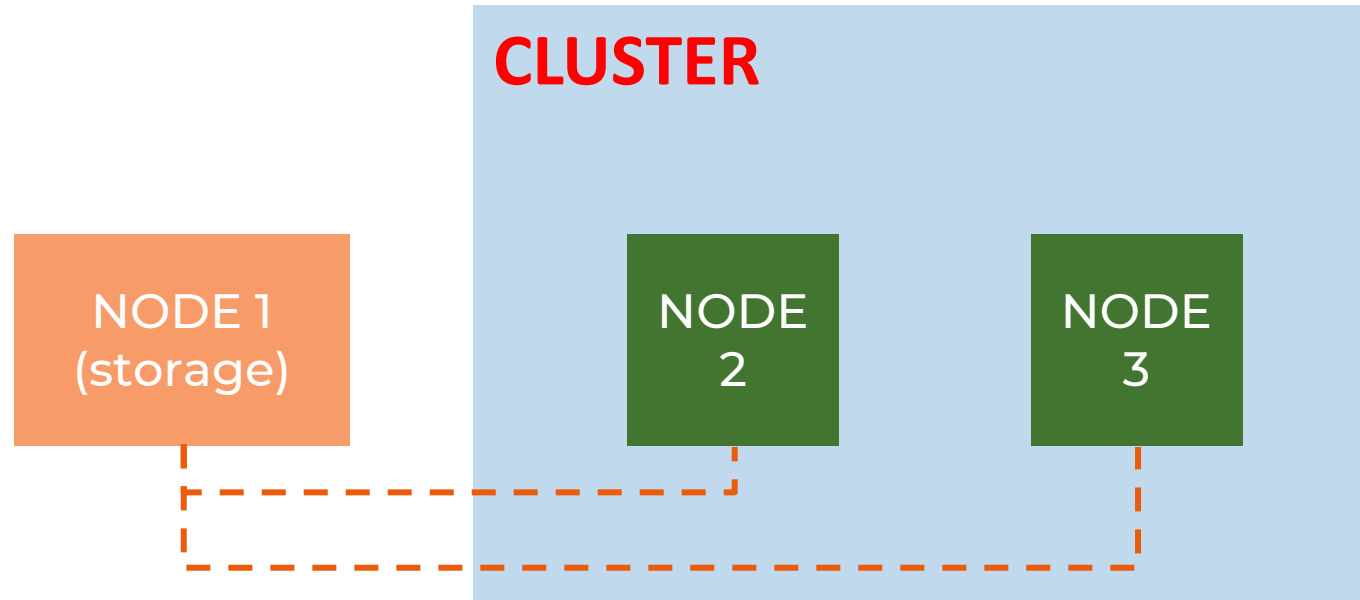


```
node2# dnf install --enablerepo=ha fence-agent-scsi -y
node2# ls -l /dev/disk/by-id | grep sda | grep wwn
node2# pcs stonith create scsi-shooter fence_scsi pcmk_host
_list="node1.example.com node2.example.com" devices=/dev/di
sk/by-id/wwn-<SCSI-DISK-ID> meta provides=unfencing
node2# pcs stonith config scsi-shooter
node2# pcs status
node2# pcs stonith fence node2.example.com
node2# pcs status
```

# LVM

- CLUSTER LVM2

LVM



## LVM



```
node2/3# vi /etc/lvm/lvm.conf  
system_id_source = "uname"
```

```
node2# parted --script /dev/sda "mklabel msdos"  
node2# parted --script /dev/sda "mkpart primary 0% 100%"  
node2# parted --script /dev/sda "set 1 lvm on"
```

```
node2# pvcreate /dev/sda1  
node2# vgcreate vg_ha_lvm /dev/sda1  
node2# vgs -o+systemid  
node2# lvcreate -l 100%FREE -n lv_ha_lvm vg_ha_lvm
```

## LVM



```
node2# mkfs.xfs /dev/vg_ha_lvm/lv_ha_lvm
```

```
node2# vgchange vg_ha_lvm -an
```

```
node2# lvm pvscan --cache --activate ay
```

```
node2# pcs resource create lvm_ha_iscsi ocf:heartbeat:LVM-activate vg_name=vg_ha_lvm vg_access_mode=system_id --group ha_lvm_group
```

```
node2# pcs resource create lvm_ha_mount FileSystem device=/dev/vg_ha_lvm/lv_ha_lvm directory=/home/lvm_directory fstype=xfs --group ha_lvm_group
```

```
node2# pcs status
```



LVM

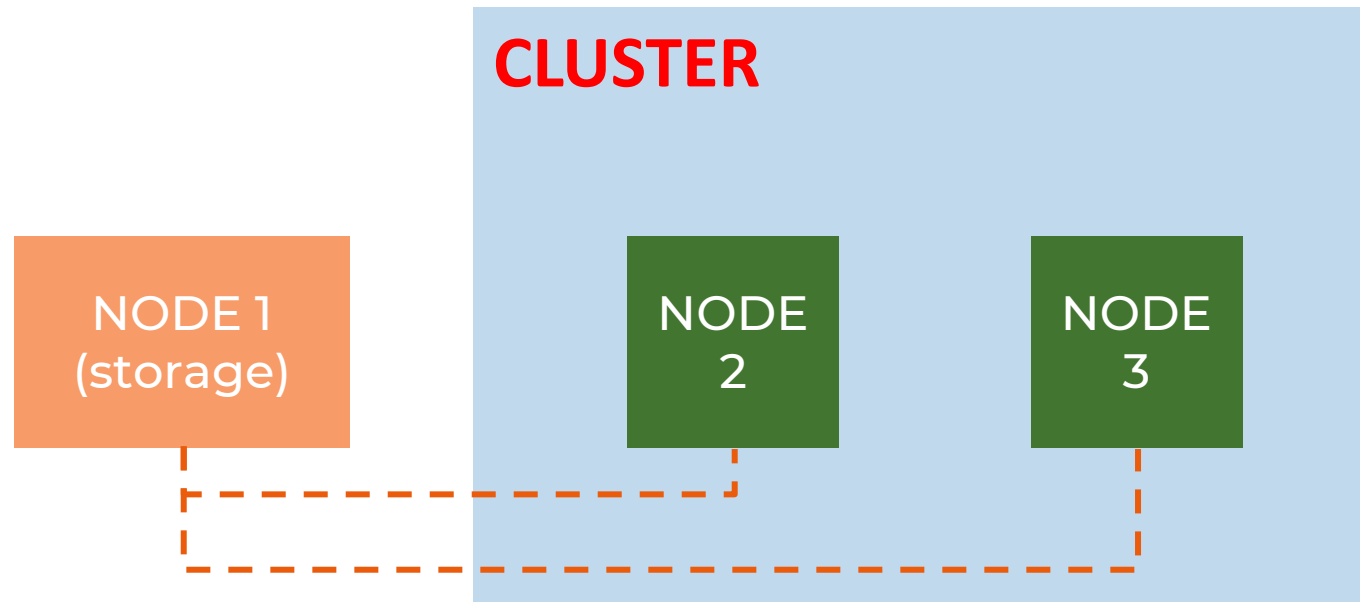


**DO SELF LAB**

# **ROLL BACK VIRTUAL MACHINE AND REINSTALL TO THE PACEMAKER CLUSTER**

**NFS**

## CLUSTER





## NFS

```
node2/3# firewall-cmd --add-service=nfs --permanent
node2/3# firewall-cmd --add-service={nfs3,mountd,rpc-bind} --permanent
node2/3# firewall-cmd --reload
```

```
node2/3# mkdir -p /home/nfs-share
node2# pcs resource create nfs_share_iscsi ocf:heartbeat:Filesystem device=/dev/vg_ha_iscs
i/lv_ha_iscsi directory=/home/nfs-share fstype=xfs --group ha_iscsi_group
```

```
node2# pcs status
node2# mount | grep /home/nfs-share
```

```
node2# pcs resource create nfs_daemon ocf:heartbeat:nfsserver nfs_shared_infodir=/home/nfs
-share/nfsinfo nfs_no_notify=true --group ha_iscsi_group
```

```
node2# pcs resource create nfs_vip ocf:heartbeat:IPaddr2 ip=192.168.100.250 nic=eth1 cidr_
netmask=24 --group ha_iscsi_group
```



## NFS

```
node2# pcs resource create nfs_notify ocf:heartbeat:nfsnotify source_host=192.168.100.250  
--group ha_iscsi_group
```

```
node2# mkdir -p /home/nfs-share/nfs-root/share01
```

```
node2# pcs resource create nfs_root ocf:heartbeat:exportfs clientspec=192.168.100.0/255.255.255.0 options=rw,sync,no_root_squash directory=/home/nfs-share/nfs-root fsid=0 --group ha_iscsi_group
```

```
node1 # pcs resource create nfs_share01 ocf:heartbeat:exportfs clientspec=192.168.100.0/255.255.255.0 options=rw,sync,no_root_squash directory=/home/nfs-share/nfs-root/share01 fsid=1 --group ha_iscsi_group
```

```
node2 # pcs status
```

```
node2 # showmount -e
```

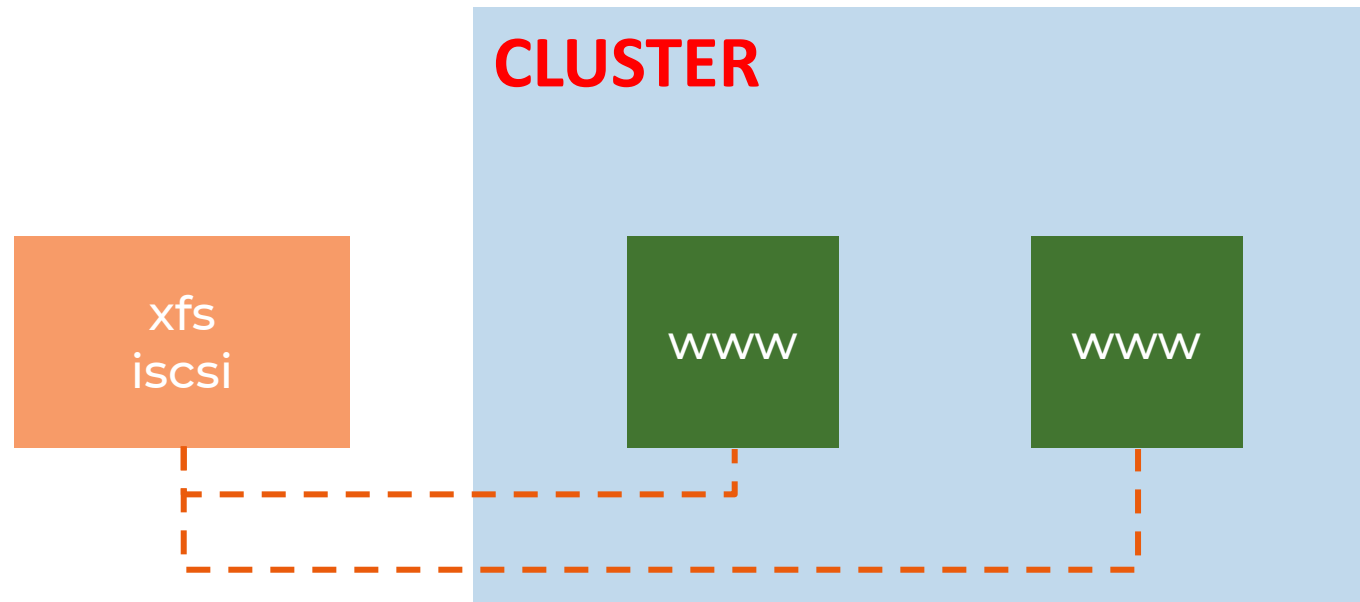
```
node2/3 # mkdir -p /mnt/test_nfs
```

```
node2/3 # mount 192.168.100.250:/home/nfs-share/nfs-root/share01 /mnt
```

# WWW

- APACHE
- TOMCAT

## CLUSTER





## WWW

```
node12/3 dnf install httpd -y
node2/3# vi /etc/httpd/conf.d/server-status.conf
<Location /server-status>
    SetHandler server-status
    Require local
</Location>
```

```
node2/3# firewall-cmd --add-service={http,https} --permanent && firewall-cmd --runtime-to-permanent
node2/3# mkdir -p /mnt/html
node2/3# mount /dev/vg_ha_iscsi/lv_ha_iscsi /mnt/html
node2/3# echo "Hello World" > /mnt/html/index.html && umount /mnt/html/
```

```
node2# pcs resource create httpd_fs ocf:heartbeat:Filesystem device=/dev/vg_ha_iscsi/lv_ha_iscsi directory=/var/www fstype=xfs --group ha_group_iscsi
```





## WWW

```
node2# pcs resource create httpd_vip ocf:heartbeat:IPaddr2 ip=192.168.10  
0.240 cidr_netmask=24 --group ha_group_iscsi
```

```
node2# pcs resource create website ocf:heartbeat:apache configfile=/etc/  
httpd/conf/httpd.conf statusurl=http://127.0.0.1/server-status --group h  
a_group
```

```
node2# pcs status
```

```
node2# restorecon -RFvvv /var/www/
```

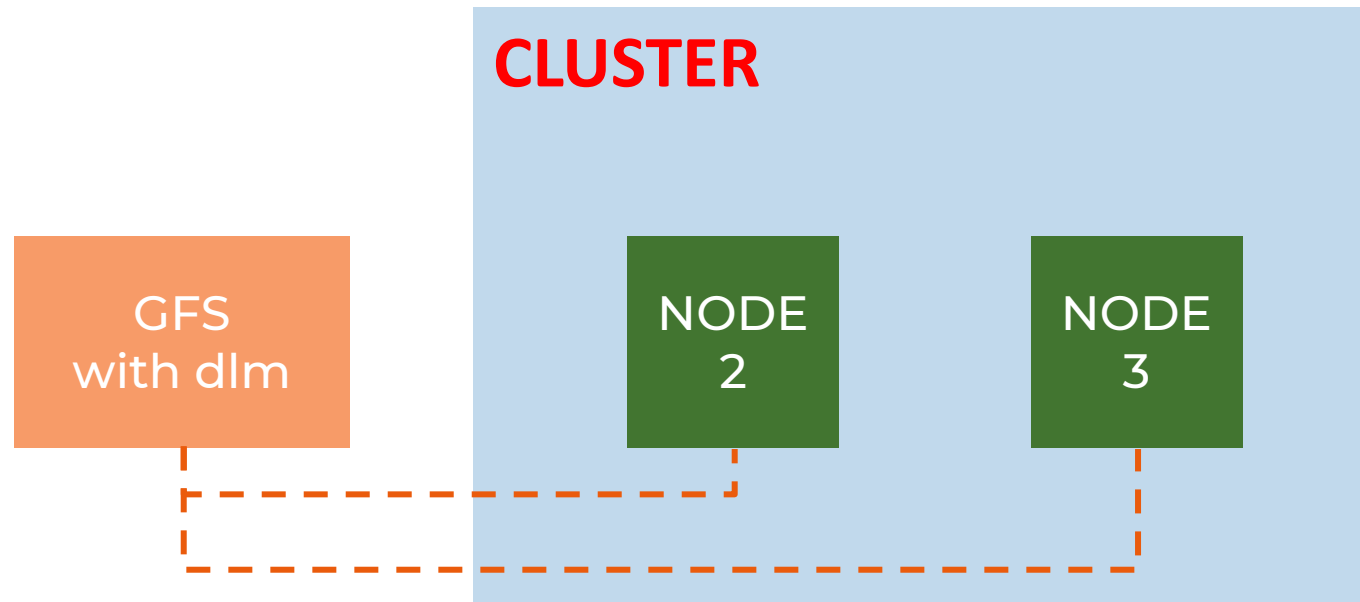
```
node2# curl http://192.168.100.240/index.html
```



A

# GFS2

## CLUSTER



## GFS



```
node2# dnf --enablerepo=ha,resilientstorage -y install lvm
2-lockd gfs2-utils dlm
node2/3# vi /etc/lvm/lvm.conf
use_lvmlockd = 1
node2/3# systemctl enable --now dlm lvmlockd lvmlocks
node2# pcs property set no-quorum-policy=freeze
node2# pcs resource create dlm ocf:pacemaker:controld op mo
nitor interval=30s on-fail=fence --group locking
```

## GFS



```
node2# pcs resource clone locking interleave=true  
node2# pcs resource create lvmlockdd ocf:heartbeat:lvmlockd  
op monitor interval=30s on-fail=fence --group locking  
node2# pcs status
```

## GFS



```
node2# parted --script /dev/sdb "mklabel gpt"
node2# parted --script /dev/sdb "mkpart primary 0% 100%"
node2# parted --script /dev/sdb "set 1 lvm on"

node2# pvcreate /dev/sdb1
node2# vgcreate --shared vg_gfs2 /dev/sdb1

node2# vgs
node2# vgchange --lock-start vg_gfs2
node2# lvcreate -l 100%FREE -n lv_gfs2 vg_gfs2
```

## GFS



```
node2# mkfs.gfs2 -j2 -p lock_dlm -t ha_cluster_lab:gfs2 /dev/vg_gfs2/lv_
gfs2
node2# pcs resource create shared_lv ocf:heartbeat:LVM-activate lvname=l
v_gfs2 vname=vg_gfs2 activation_mode=shared vg_access_mode=lvmlockd --g
roup shared_vg
node2# pcs resource clone shared_vg interleave=true
node2# pcs constraint order start locking-clone then shared_vg-clone
```





```
node2# pcs constraint colocation add shared_vg-clone with locking-clone
node2# pcs resource create shared_fs ocf:heartbeat:Filesystem device="/dev/vg_gfs2/lv_gfs2" directory="/home/gfs2-share" fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence --group shared_vg
node2# pcs status
node2# mount | grep gfs2-share
node3# mount | grep gfs2-share
```

# PACEMAKER



## RHEL 5

cman

rgmanager

openAIS

GFS & GFS2

system-config-cluster

conga  
(ricci & luci)

Red Hat Cluster Suite

## RHEL 6

cman

rgmanager

openAIS/  
Corosync

<http://blog.csdn.net/hsh11214>

GFS & GFS2

CLI tool: CCS

conga  
(ricci & luci)

Red Hat High  
Availability Add-On

## RHEL 7

Pacemaker

Corosync

GFS2

CLI Tool: pcs

pcs-gui

Red Hat High  
Availability Add-On