THE OPENSOURCE CLUSTER LAB FOR Pacemaker

CHOI GOOKHYUN

KOICA Korea International Cooperation Agency

KISCA Korea Information Systems Consulting & Audit Co., Ltd.

DAY 1

What is the High Availability.

The Red hat kind Linux distribution
H/A with Linux relationship
Lab Design and Architect of service for training

INSTALLATION AND BASIC COMMAND

Install the Pacemaker on Rocky Linux 9
Configure to basic configuration for Pacemaker
Fly to basic command

INSTALLATION AND BASIC COMMAND

DAY 2

What is the High Availability.

The Red hat kind Linux distribution
H/A with Linux relationship
Lab Design and Architect of service for training

INSTALLATION

Install the Pacemaker on Rocky Linux 9
Configure to basic configuration for Pacemaker
Fly and Watch to the basic components in Pacemaker

Content 03.

Title Name Title Name Title Name
Title Name Title Name Title Name

DAY 3

What is the High Availability.

The Red hat kind Linux distribution
H/A with Linux relationship
Lab Design and Architect of service for training

INSTALLATION

Install the Pacemaker on Rocky Linux 9
Configure to basic configuration for Pacemaker
Fly and Watch to the basic components in Pacemaker

Content 03.

Title Name Title Name Title Name
Title Name Title Name Title Name

DAY 4

What is the High Availability.

The Red hat kind Linux distribution
H/A with Linux relationship
Lab Design and Architect of service for training

INSTALLATION

Install the Pacemaker on Rocky Linux 9
Configure to basic configuration for Pacemaker
Fly and Watch to the basic components in Pacemaker

Content 03.

Title Name Title Name Title Name
Title Name Title Name Title Name

INTRODUCE

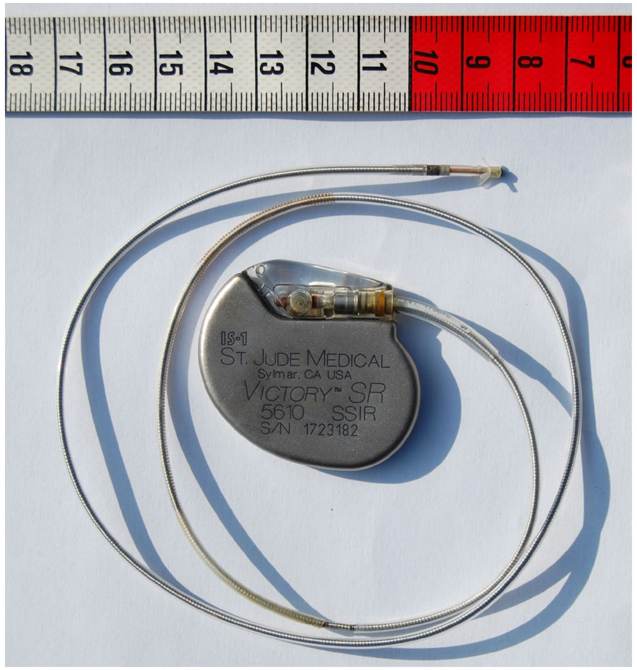
- WHAT IS THE PACEMAKER
- COMPARE WITH CMAN AND PACEMAKER
- WHAT IS DIFFRENCE BETWEEN RHEL CENTOS AND ROCKY FOR H/A

PACEMAKER

THE PACEMAKER IS NOT THE REAL PACEMAKER

:))

But It would be very simulating the Real Pacemaker.



PACEMAKER

The Pacemaker doing like these.

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

COROSYNC

Corosync

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

**MOST CORE COMPOMENT

https://clusterlabs.org/corosync.html

- Pacemaker
- DRBD
- ScanCore

DRBD

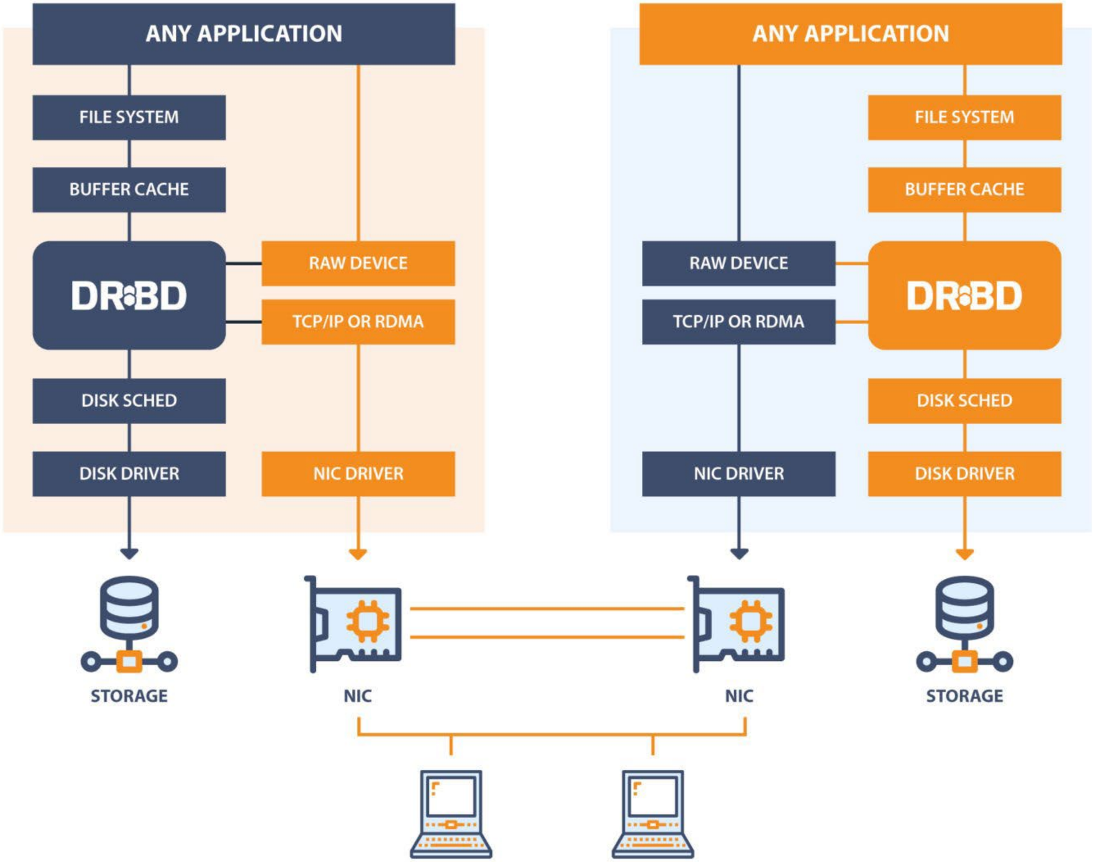
DRBD is Distribute Replicated Storage System. This is helping and implemented as kernel driver, several userspace management application and shell script.

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

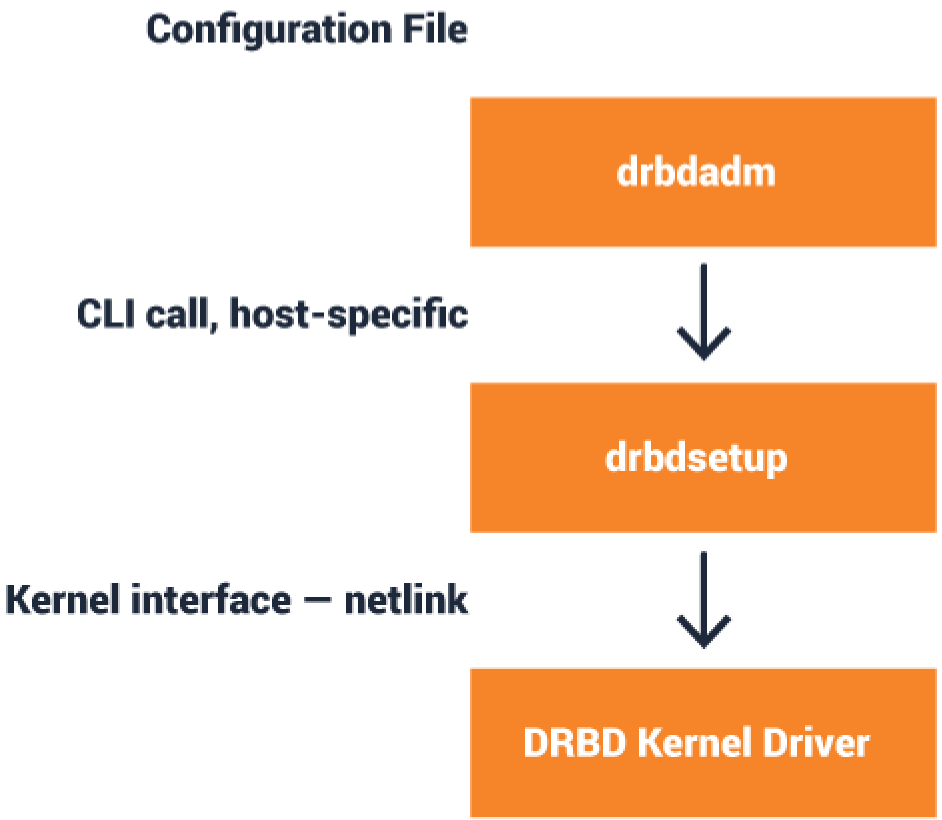
https://linbit.com/drbd/

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

DRBD


ANY APPLICATION
FILE SYSTEM
BUFFER CACHE
DRBD
RAW DEVICE
TCP/IP OR RDMA
DISK SCHED
DISK DRIVER
NIC DRIVER
ANY APPLICATION
FILE SYSTEM
BUFFER CACHE
RAW DEVICE
TCP/IP OR RDMA
DRBD
DISK SCHED
NIC DRIVER
DISK DRIVER
STORAGE
NIC
NIC
STORAGE

DRBD



Configuration File

drbdadm

CLI call, host-specific

drbdsetup

Kernel interface — netlink

DRBD Kernel Driver

ScanCore

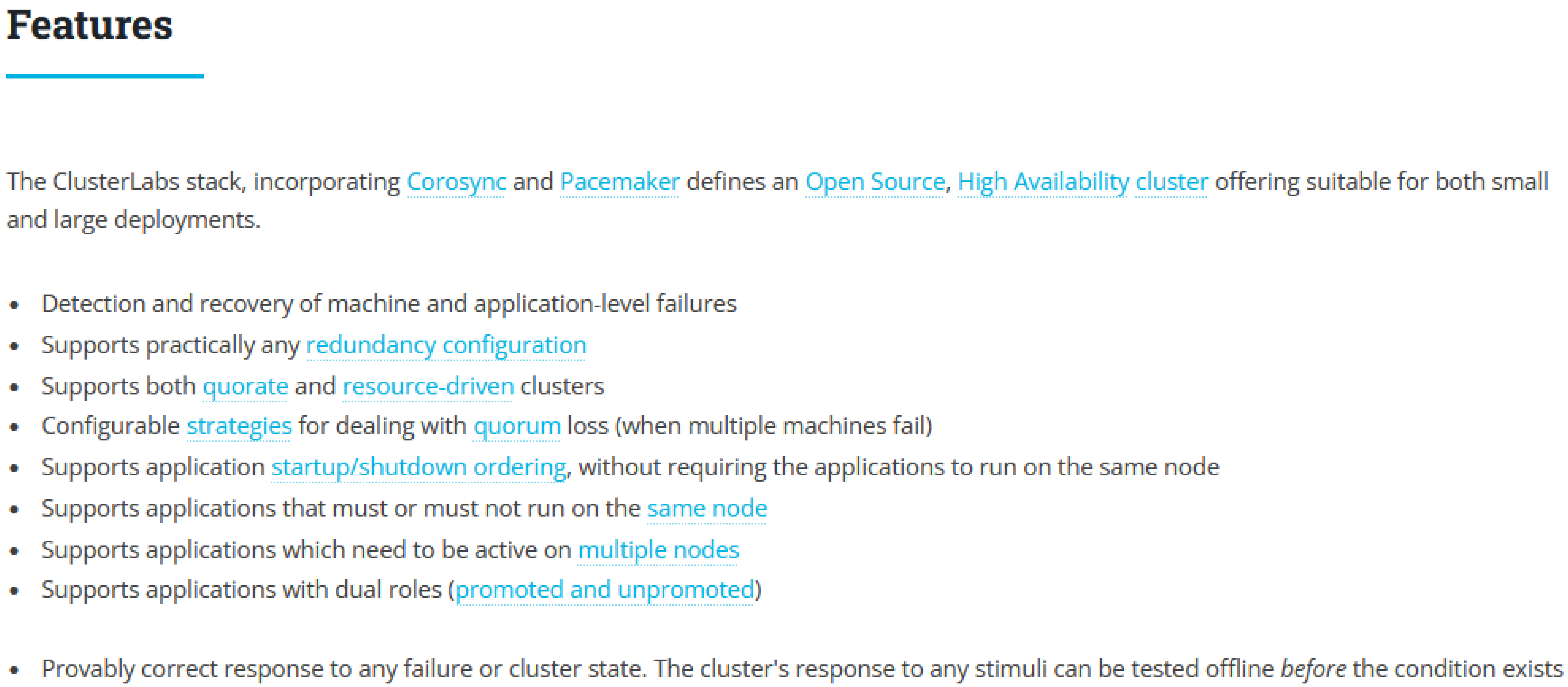
ScanCore is at its core, "decision engine". This component will check to node below condition.

- Overheating
- 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

PACEMAKER



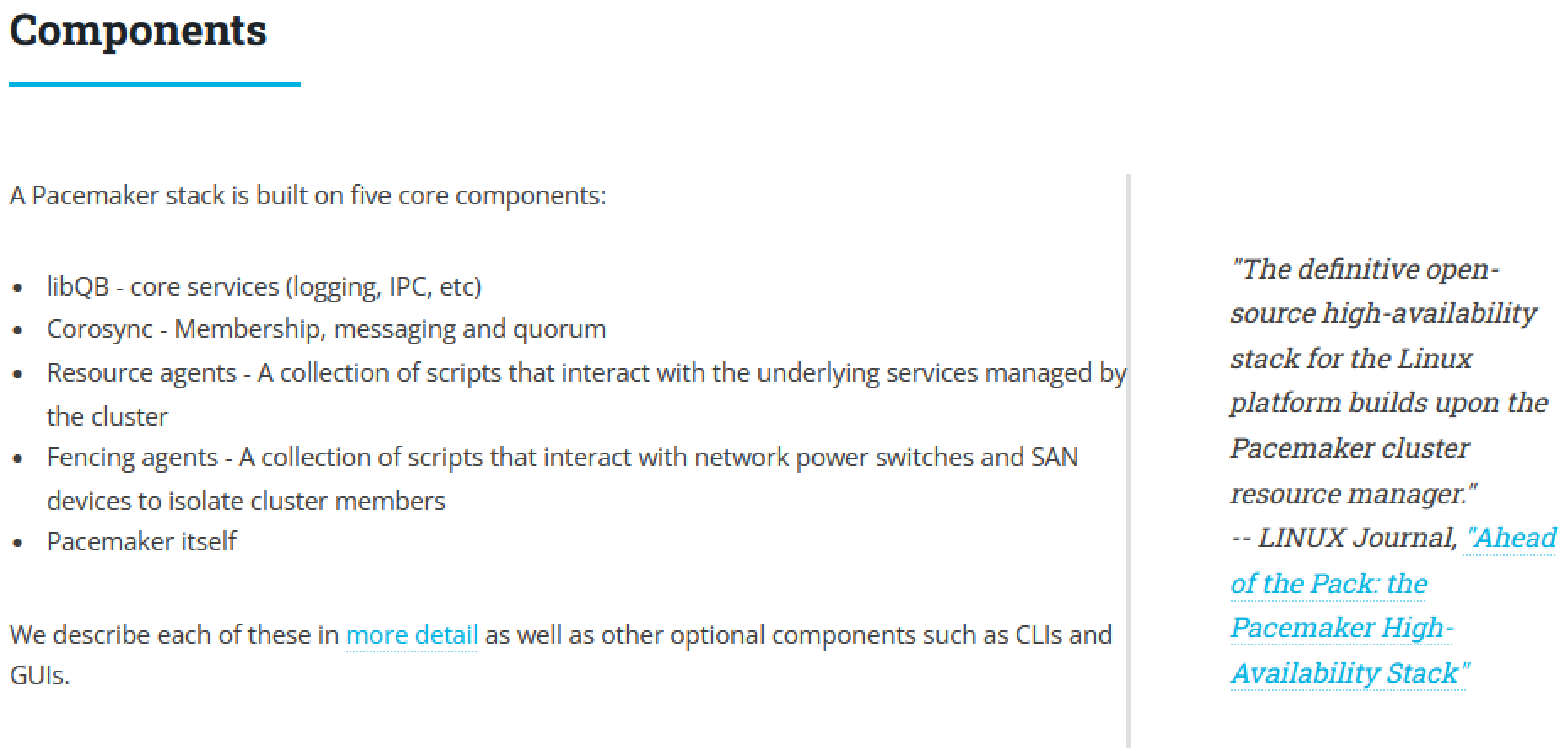
Features

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

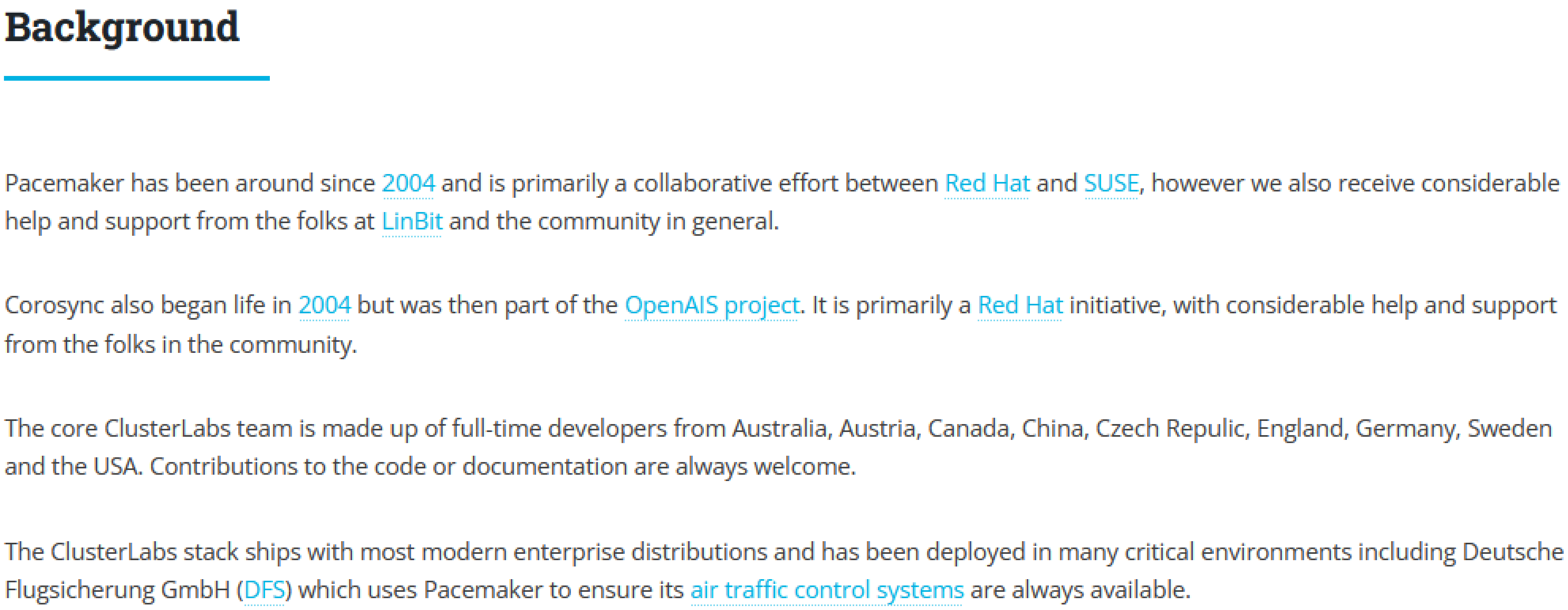
A Pacemaker stack is built on five core components:

- 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 Repulic, 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 RHLE/Centos 7 can use able to be Pacemaker

RGMAN VS PACEMAKER

	rgmanager	Pacemaker
Resource Configuration Management	Manual	Automatic
Resource Management Model	Resource Group	Resource-Dependency, Resource Group
Dependency Models	Colocation, Start-After	User-defined
Event Handling Model	Distributed or Centralized	Centralized
Command-Line Interface Management	Status, Control	Status, Control, Administration
Fencing Model	Assumed	Flexible
Multi-State Resources	No	Yes
Event Scripts	Yes	No
Maximum Node Count	16	16

RGMAN VS PACEMAKER

Exclusive Services	Yes	Yes
Failover Domains	Yes	Yes
Resource Exclusion	No	Yes
Time-Based Resource Control	No	Yes
Resource Attribute Inheritance	Yes	Yes
Shared Resources	Yes	Yes
Cloned Resources	No	Yes
Resource Agent APIs	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

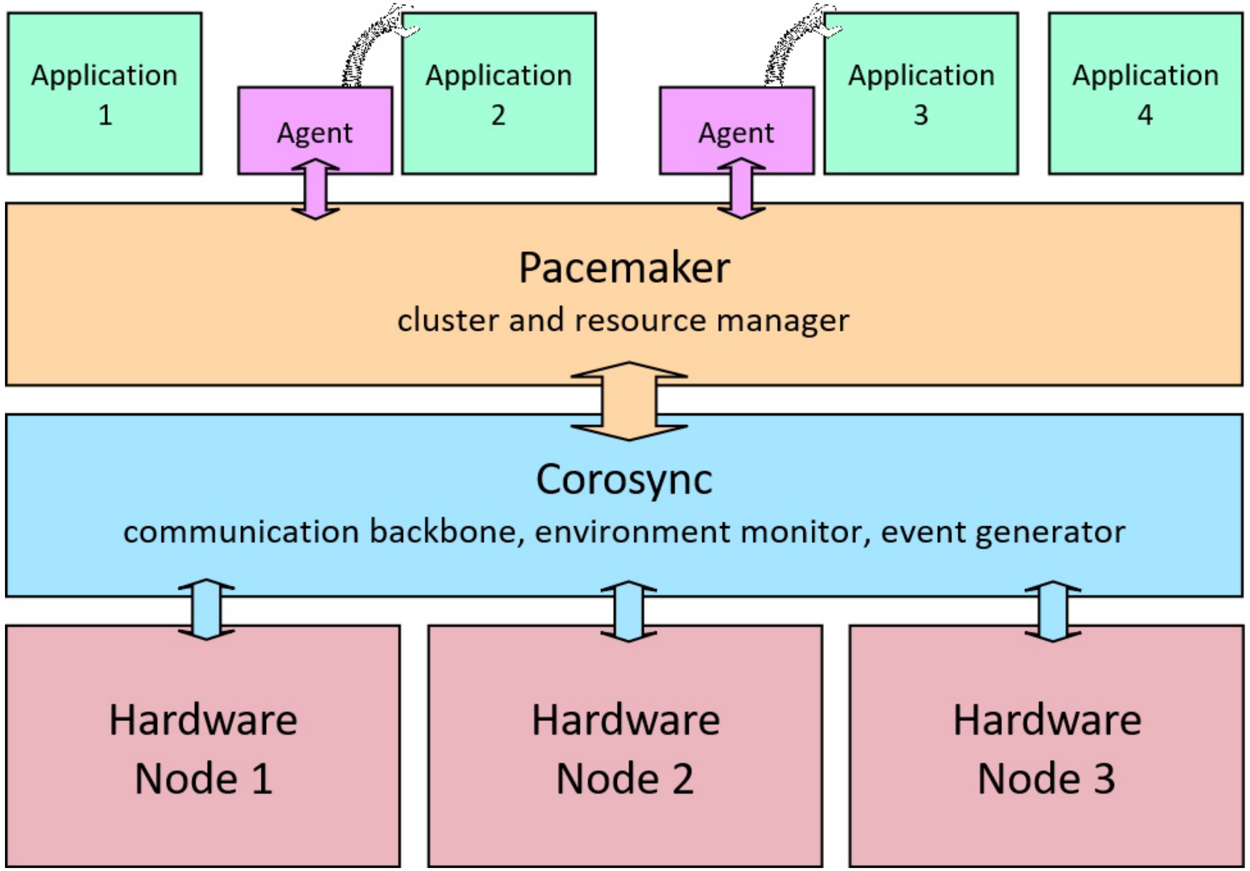
THE AGENT

Pacemaker and RGMan can use OCF Resource Agents.

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

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

THE AGENT


Application 1
Agent
Application 2
Agent
Application 3
Application 4
Pacemaker
cluster and resource manager
Corosync
communication backbone, environment monitor, event generator
Hardware Node 1
Hardware Node 2
Hardware Node 3

THE AGENT

THE AGENT

1. H/A, It cannot achieve 100% availability.
2. A good HA Cluster systems adds a "9" to your base availability.

99->99.9, 99.9 -> 99.99 something like this.

3. Don't get and drag the complexity into your cluster

Most of time, You will fail the design and High Availability Architecture.

THE NINES THORY

99.9999% IN 30SEC
99.999% IN 5MIN
99.99% IN 52 MIN
99.9% IN 9 HOUR
99% IN 3.5 DAY

DR VS HA

DR

FAILOVER IS EXOENSIVE
FAILOVER TIMES OFTEN MEASURED IN HOURS
UNRELIABLE INTER-NODE COMMUNICATION ASSUMED
TOO MUCH COMPLEX AND COMPLICATED DESGIN WITH CLUSTER AND NODES

HA
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.

LAB SETUP

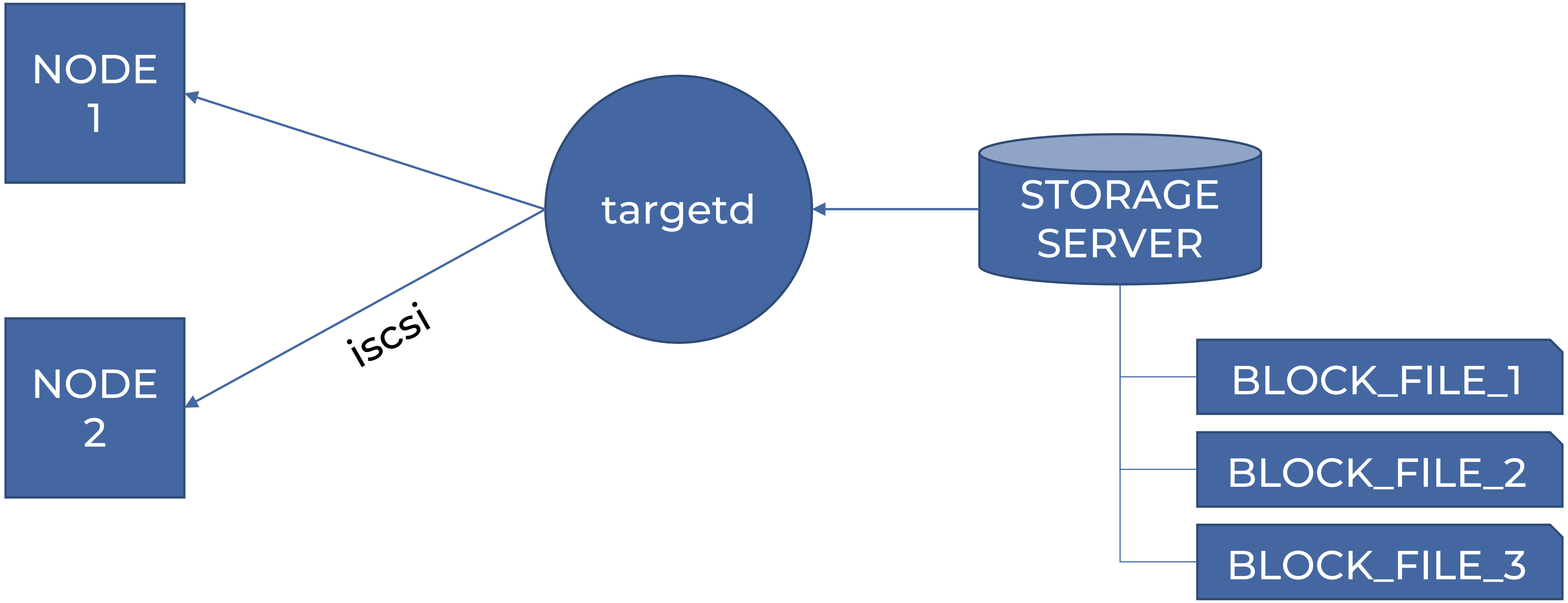
- INSTALL LINUX FOR HOST COMPUTER
- LIBVIRTD AND VIRSH COMMAND
- BUILD UP VIRTUAL MACHINE FOR PoC

THE FUNDAMENTAL

- PACEMAKER RESOURCE
- BASIC COMMANDS

INSTALLATION

LAB DESIGN



INSTALLATION

- LINUX CONFIGURE AND INSTALLATION
- ISCSI TARGET SERVER CONFIGURATION

PACEMAKER

PCS

```
# hostnamectl set-hostname nodeX.example.com

# 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

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

PACEMAKER

PCS

```
# for i in node{1..3} ; do sshpass -procky ssh-copy-id -o "StrictHostKeyChecking=no" root@$i && scp /etc/hosts root@$i.example.com:/etc/hosts ; done

# for i in node{1..2} storage ; do sshpass -p rocky ssh root@$i 'dnf --enablerepo=highavailability -y install pacemaker pcs' ; done

# dnf update -y

# for i in node{1..2} storage ; do dnf install firewalld && systemctl enable --now firewalld ; done
```

PACEMAKER

PCS

```
# for i in {1..2} ; do sshpass -p rocky ssh root@node${i} 'firewall-cmd --add-service=high-
availability && firewall-cmd --runtime-to-permanent' ; done

# for i in {1..2} ; do sshpass -p rocky ssh root@node$i 'echo centos | passwd --stdin hacluster'
&& systemctl enable --now pcs.service ; done

# ping node1 -c3
# ping node2 -c3
# ping node3 -c3
```

PACEMAKER

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
```

KOICA Korea International Cooperation Agency KISCA Korea Information Systems Consulting & Audit Co., Ltd.

ISCSI

- Title Name Title Name Title Name
- Title Name Title Name Title Name

STORAGE(TARGET SERVER, node3)

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

# dnf install iscsi-initiator-utils -y

# mkdir -p /var/lib/iscsi_disks

targetcli backstores/fileio create iscsi /var/lib/iscsi_disks/iscsi_disk.img 2G
targetcli backstores/fileio create nfs /var/lib/iscsi_disks/nfs_disk.img 2G
targetcli backstores/fileio create gfs2 /var/lib/iscsi_disks/gfs2_disk.img 2G
```

STORAGE(TARGET SERVER, node3)

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

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

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

PACEMAKER

STORAGE(TARGET SERVER, node3)

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

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

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
```

PCS

- CLUSTER CREATE
- CLUSTER VERIFY

PACEMAKER

CLUSTER

```
# dnf --enablerepo=ha -y install pacemaker pcs
# systemctl enable --now pcsd
# echo centos | passwd --stdin hacluster

# firewall-cmd --add-service=high-availability --permanent
# firewall-cmd --reload

# pcs host auth -u hacluster -p centos node1.example.com node2.example.com
# pcs cluster setup ha_cluster_lab node1.example.com node2.example.com
# pcs cluster start --all
# pcs cluster enable --all
# pcs cluster status
# pcs status corosync
```

PACEMAKER

CLUSTER

```
node1 #  dnf --enablerepo=highavailability -y install fence-agents-scsi
node1 # ls /dev/disk/by-id

node1 # pcs stonith create scsi-shooter fence_scsi pcmk_host_list="node1.example.com node2.example.c
om" devices=/dev/disk/by-id/wwn-<ID> meta provides=unfencing
node1 # pcs stonith config scsi-shooter
node1 # pcs status

node2 # pcs status
node2 # pcs stonith fence node1.example.com
node2 # pcs status

node1 # reboot
```

LVM

- Title Name Title Name Title Name
- Title Name Title Name Title Name

LVM

```
node1 # vi /etc/lvm/lvm.conf

system_id_source = "uname"

node1 # parted --script /dev/sda "mklabel msdos"
node1 # parted --script /dev/sda "mkpart primary 0% 100%"
node1 # parted --script /dev/sda "set 1 lvm on"

node1 # pvcreate /dev/sda1
node1 # vgcreate vg_ha_iscsi /dev/sda1
node1 # vgs -o+systemid
node1 # lvcreate -l 100%FREE -n lv_ha_iscsi vg_ha_iscsi
```

PACEMAKER

LVM

```
node1 # mkfs.xfs /dev/vg_ha_iscsi/lv_ha_iscsi
node1 # vgchange vg_ha _iscsi -an

node1 # lvm pvscan --cache --activate ay
node1 # pcs resource create lvm_ha_iscsi ocf:hearbeat:LVM-activate vg_name=vg_ha_iscsi vg_access_m
ode=system_id --group ha_iscsi_group

node1 # pcs status
```

NFS

- Title Name Title Name Title Name
- Title Name Title Name Title Name

PACEMAKER

NFS

```
node1 # firewall-cmd --add-service=nfs --permanent
node1 # firewall-cmd --add-service={nfs3,mountd,rpc-bind} --permanent
node1 # firewall-cmd --reload

node1 # mkdir -p /home/nfs-share
node1 # pcs resource create nfs_share_iscsi ocf:heartbeat:Filesystem device=/dev/vg_ha_iscsi/lv_ha_iscsi
directory=/home/nfs-share fstype=xfs --group ha_iscsi_group

node1 # pcs status
node1 # mount | grep /home/nfs-share

node1 # pcs resource create nfs_daemon ocf:heartbeat:nfsserver nfs_shared_infodir=/home/nfs-share/nf
sinfo nfs_no_notify=true --group ha_iscsi_group

node1 # pcs resource create nfs_vip ocf:heartbeat:IPaddr2 ip=192.168.100.250 cidr_netmask=24 --group h
a_iscsi_group
```

PACEMAKER

NFS

```
node1 # pcs resource create nfs_notify ocf:heartbeat:nfsnotify source_host=192.168.100.250 --group ha_iscsi_group

node1 # mkdir -p /home/nfs-share/nfs-root/share01
node1 # 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

node1 # pcs status
node1 # showmount -e

nodeX # mkdir -p /mnt/test_nfs
nodeX # mount 192.168.100.250:/home/nfs-share/nfs-root/share01 /mnt
```

PACEMAKER

NFS

```
node1 # pcs resource create nfs_notify ocf:heartbeat:nfsnotify source_host=192.168.100.250 --group ha_iscsi_group

node1 # mkdir -p /home/nfs-share/nfs-root/share01
node1 # 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

node1 # pcs status
node1 # showmount -e

nodeX # mkdir -p /mnt/test_nfs
nodeX # mount 192.168.100.250:/home/nfs-share/nfs-root/share01 /mnt
```

www

- Title Name Title Name Title Name
- Title Name Title Name Title Name

PACEMAKER

WWW

```
node1 # dnf install httpd -y
node1 # vi /etc/httpd/conf.d/server-status.conf
<Location /server-status>
   SetHandler server-status
   Require local
</Location>

node1 # vi /etc/logrotate.d/httpd
node1 # firewall-cmd --add-service={http,https} --permanent && firewall-cmd --runtime-to-permanent
node1 # mkdir -p /mnt/html
node1 # mount /dev/vg_ha_iscsi/lv_ha_iscsi /mnt/html
node1 # echo "Hello World" > /mnt/html/index.html && umount /mnt/html/

node1 # 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
```

PACEMAKER

www

```
node1 # pcs resource create httpd_vip ocf:heartbeat:IPaddr2 ip=192.168.100.240 cidr_netmask=24 --group ha_group_iscsi

node1 # pcs resource create website ocf:heartbeat:apache configfile=/etc/httpd/conf/httpd.conf statusurl=http://127.0.0.1/server-status --group ha_group

node1 # pcs status

node2 # restorecon -RFvvv /var/www/
node2 # curl http://192.168.100.240/index.html
```

GFS2

- Title Name Title Name Title Name
- Title Name Title Name Title Name

PACEMAKER

GFS

```
node1 # dnf --enablerepo=ha -y install lvm2-lockd gfs2-utils dlm
node1 # vi /etc/lvm/lvm.conf
use_lvmlockd = 1
node1 # systemctl enable --now dlm lvmlockd lvmlocks
node1 # pcs property set no-quorum-policy=freeze
node1 # pcs resource create dlm ocf:pacemaker:controld op monitor interval=30s on-fail=fence --group locking
node1 # pcs resource clone locking interleave=true
node1 # pcs resource create lvmlockdd ocf:heartbeat:lvmlockd op monitor interval=30s on-fail=fence --group locking
node1 # pcs status
```

PACEMAKER

GFS

```
node1 # parted --script /dev/sdb "mklabel gpt"
node1 # parted --script /dev/sdb "mkpart primary 0% 100%"
node1 # parted --script /dev/sdb "set 1 lvm on"

node1 # pvcreate /dev/sdb1
node1 # vgcreate --shared vg_gfs2 /dev/sdb1

node1 # vgs
node1 # vgchance --lock-start vg_gfs2
node1 # lvcreate -l 100%FREE -n lv_gfs2 vg_gfs2

node1 # mkfs.gfs2 -j2 -p lock_dlm -t ha_cluster_lab:gfs2 /dev/vg_gfs2/lv_gfs2
node1 # pcs resource create shared_lv ocf:heartbeat:LVM-activate lvname=lv_gfs2 vgname=vg_gfs2 activation_mode=shared vg_access_mode=lvmlockd --group shared_vg
node1 # pcs resource clone shared_vg interleave=true
node1 # pcs constraint order start locking-clone then shared_vg-clone
```

GFS

```
node1 # pcs constraint colocation add shared_vg-clone with locking-clone
node1 # 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
node1 # pcs status
node1 # mount | grep gfs2-share
node2 # mount | grep gfs2-share
```

Title Area

Sub Title Style

Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area
Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area
Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area

Notice
Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area
Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area
Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area Contents Area

Project Title Area

Title Area

- Sub Title Style

Contents Area Contents Area

1. title	2. title	3. title	4. title	5. title	6. title
text	text	text	text	text	text
text	text	text	text	text	text