#### 425px-DRBD_logo

## What Is Pacemaker?

Pacemaker is a cluster resource manager, that is, a logic responsible for a life-cycle of deployed software — indirectly perhaps even whole systems or their interconnections — under its control within a set of computers (a.k.a. nodes) and driven by prescribed rules.

Pacemaker’s key features include:

* Detection and recovery of node and service-level failures
* Storage agnostic, no requirement for shared storage
* Resource agnostic, anything that can be scripted can be clustered
* Supports large and small clusters
* Automatically replicated configuration that can be updated from any node
* Supports fencing for ensuring data integrity

**Pacemaker Architecture**

At the highest level, the cluster is made up of three pieces:

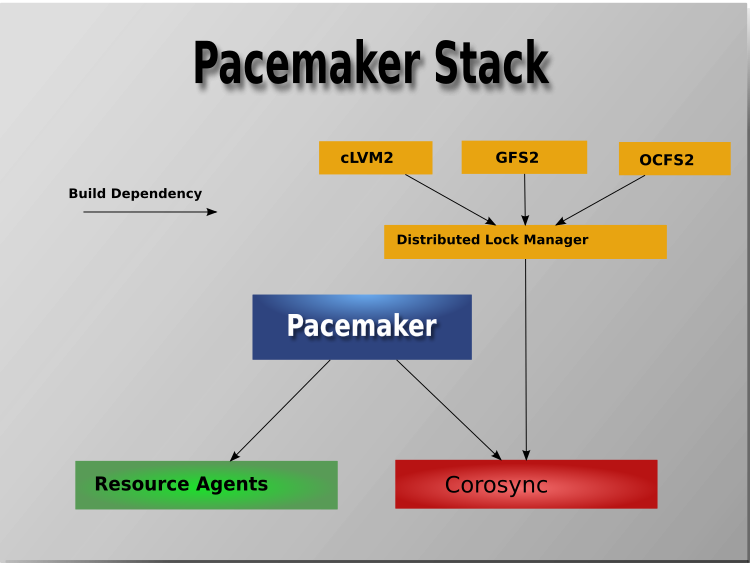
**Non-cluster-aware components**. These pieces include the resources themselves; scripts that start, stop and monitor them; and a local daemon that masks the differences between the different standards these scripts implement. Even though interactions of these resources when run as multiple instances can resemble a distributed system, they still lack the proper HA mechanisms and/or autonomous cluster-wide governance as subsumed in the following item.

**Resource management**. Pacemaker provides the brain that processes and reacts to events regarding the cluster. These events include nodes joining or leaving the cluster; resource events caused by failures, maintenance and scheduled activities; and other administrative actions. Pacemaker will compute the ideal state of the cluster and plot a path to achieve it after any of these events. This may include moving resources, stopping nodes and even forcing them offline with remote power switches.

**Low-level infrastructure**. Projects like Corosync, CMAN and Heartbeat provide reliable messaging, membership and quorum information about the cluster.

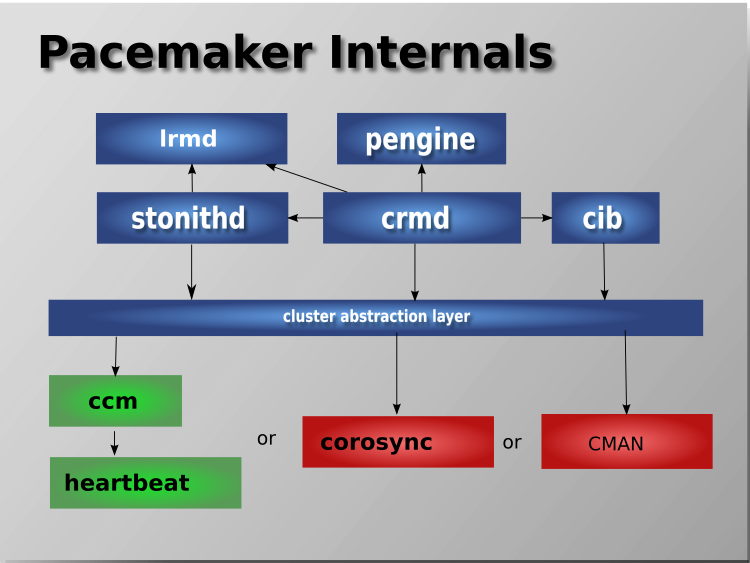
Note:

When combined with Corosync, Pacemaker also supports popular open source cluster filesystems.



**Internal Components**

* Pacemaker itself is composed of five key components:
* Cluster Information Base (CIB)
* Cluster Resource Management daemon (CRMd)
* Local Resource Management daemon (LRMd)
* Policy Engine (PEngine or PE)
* Fencing daemon (STONITHd)



The CIB uses XML to represent both the cluster’s configuration and current state of all resources in the cluster. The contents of the CIB are automatically kept in sync across the entire cluster and are used by the PEngine to compute the ideal state of the cluster and how it should be achieved.

This list of instructions is then fed to the Designated Controller (DC). Pacemaker centralizes all cluster decision making by electing one of the CRMd instances to act as a master. Should the elected CRMd process (or the node it is on) fail, a new one is quickly established.

The DC carries out the PEngine’s instructions in the required order by passing them to either the Local Resource Management daemon (LRMd) or CRMd peers on other nodes via the cluster messaging infrastructure (which in turn passes them on to their LRMd process).

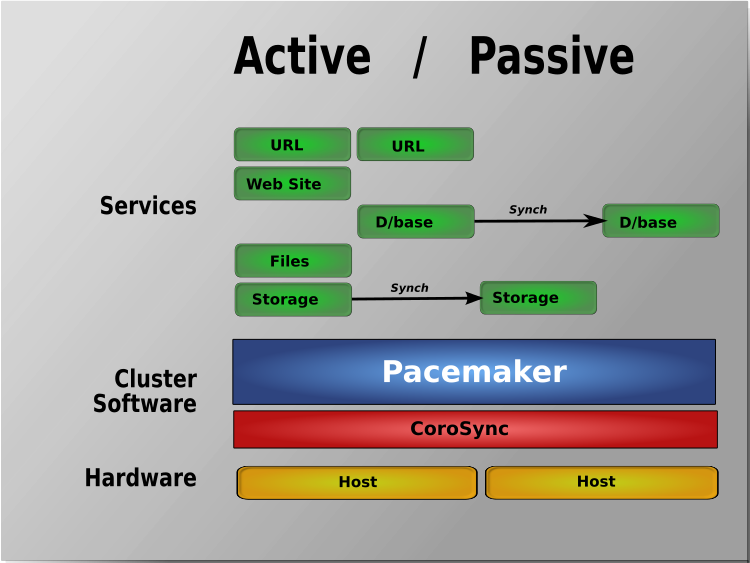
The peer nodes all report the results of their operations back to the DC and, based on the expected and actual results, will either execute any actions that needed to wait for the previous one to complete, or abort processing and ask the PEngine to recalculate the ideal cluster state based on the unexpected results.

In some cases, it may be necessary to power off nodes in order to protect shared data or complete resource recovery. For this, Pacemaker comes with STONITHd.

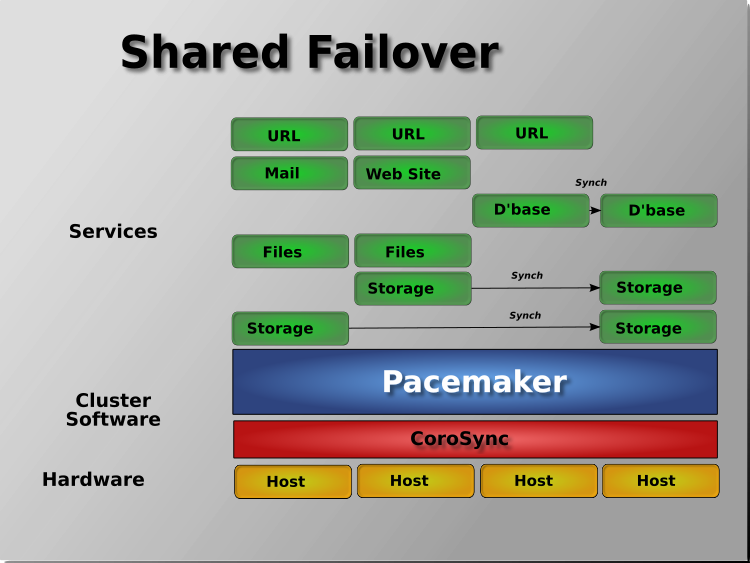
In Pacemaker, STONITH devices are modeled as resources (and configured in the CIB) to enable them to be easily monitored for failure, however STONITHd takes care of understanding the STONITH topology such that its clients simply request a node be fenced, and it does the rest.

## Types of Pacemaker Clusters

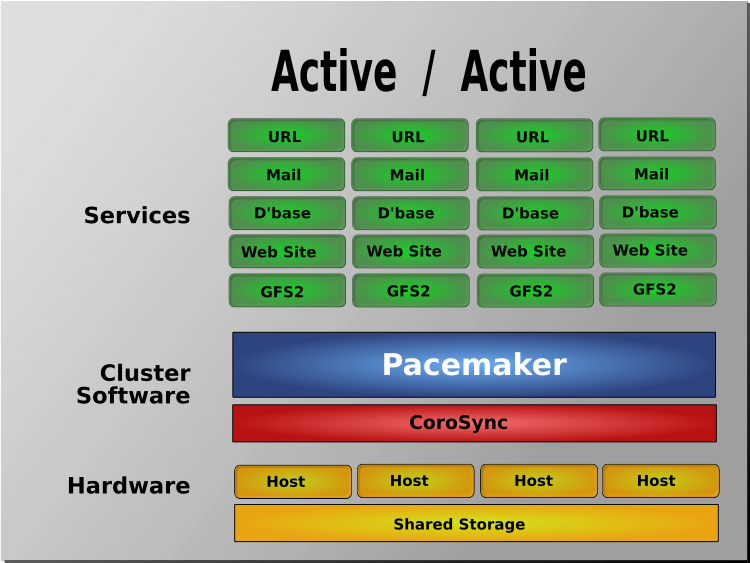
Pacemaker makes no assumptions about your environment. This allows it to support practically any [redundancy configuration](http://en.wikipedia.org/wiki/High-availability_cluster" \l "Node_configurations) including Active/Active, Active/Passive



Two-node Active/Passive clusters using Pacemaker and DRBD are a cost-effective solution for many High Availability situations.



By supporting many nodes, Pacemaker can dramatically reduce hardware costs by allowing several active/passive clusters to be combined and share a common backup node.



When shared storage is available, every node can potentially be used for failover. Pacemaker can even run multiple copies of services to spread out the workload.

**Configuration Steps:**

Now , Configure two machines, prerequisites are:

* Configured system must updated
* They must have entry in /etc/hosts file
* Network configuration
* Both nodes must communicate with each other
* Configure ssh on both node

## Install the Cluster Software

# yum install -y pacemaker pcs psmisc policycoreutils-python

**Note:**

This document will show commands that need to be **executed on both nodes** with a simple **#** prompt. Be sure to run them on each node individually.

This document uses **pcs** for cluster management. Other alternatives, such as **crmsh**, are available, but their syntax will differ from the examples used here.

### Allow cluster services through firewall

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

success

# firewall-cmd --reload

Success

### Enable pcs Daemon

# systemctl start pcsd.service

# systemctl enable pcsd.service

Created symlink from /etc/systemd/system/multi-user.target.wants/pcsd.service to /usr/lib/systemd/system/pcsd.service.

The installed packages will create a **hacluster** user with a disabled password. While this is fine for running **pcs** commands locally, the account needs a login password in order to perform such tasks as syncing the corosync configuration, or starting and stopping the cluster on other nodes.

So now we will set a password for the **hacluster** user, using the same password on both nodes:

# passwd hacluster

Changing password for user hacluster.

New password:

Retype new password:

passwd: all authentication tokens updated successfully.

### Configure Corosync

On either node, use **pcs cluster auth** to authenticate as the **hacluster** user:

[root@pcmk-1 ~]# pcs cluster auth pcmk-1 pcmk-2

Username: hacluster

Password:

pcmk-2: Authorized

pcmk-1: Authorized

Next, use pcs cluster setup on the same node to generate and synchronize the corosync configuration:

[root@pcmk-1 ~]# pcs cluster setup --name mycluster pcmk-1 pcmk-2

Destroying cluster on nodes: pcmk-1, pcmk-2...

pcmk-2: Stopping Cluster (pacemaker)...

pcmk-1: Stopping Cluster (pacemaker)...

pcmk-1: Successfully destroyed cluster

pcmk-2: Successfully destroyed cluster

Sending 'pacemaker\_remote authkey' to 'pcmk-1', 'pcmk-2'

pcmk-2: successful distribution of the file 'pacemaker\_remote authkey'

pcmk-1: successful distribution of the file 'pacemaker\_remote authkey'

Sending cluster config files to the nodes...

pcmk-1: Succeeded

pcmk-2: Succeeded

Synchronizing pcsd certificates on nodes pcmk-1, pcmk-2...

pcmk-2: Success

pcmk-1: Success

Restarting pcsd on the nodes in order to reload the certificates...

pcmk-2: Success

pcmk-1: Success

If you received an authorization error for either of those commands, make sure you configured the hacluster user account on each node with the same password.

**Note:** PCS is a utility to queue up several changes into a file and commit those changes automatically.

## Start the Cluster

Now that corosync is configured, it is time to start the cluster:

[root@pcmk-1 ~]# pcs cluster start --all

pcmk-1: Starting Cluster...

pcmk-2: Starting Cluster…

# pcs cluster start

Starting Cluster…

OR

# systemctl start corosync.service

# systemctl start pacemaker.service

Next, check the membership and quorum APIs:

[root@pcmk-1 ~]# corosync-cmapctl | grep members

runtime.totem.pg.mrp.srp.members.1.config\_version (u64) = 0

runtime.totem.pg.mrp.srp.members.1.ip (str) = r(0) ip(192.168.122.101)

runtime.totem.pg.mrp.srp.members.1.join\_count (u32) = 1

runtime.totem.pg.mrp.srp.members.1.status (str) = joined

runtime.totem.pg.mrp.srp.members.2.config\_version (u64) = 0

runtime.totem.pg.mrp.srp.members.2.ip (str) = r(0) ip(192.168.122.102)

runtime.totem.pg.mrp.srp.members.2.join\_count (u32) = 1

runtime.totem.pg.mrp.srp.members.2.status (str) = joined

[root@pcmk-1 ~]# pcs status corosync

Membership information

\----------------------

Nodeid Votes Name

1 1 pcmk-1 (local)

2 1 pcmk-2

You should see both nodes have joined the cluster.

## Verify Pacemaker Installation

[root@pcmk-1 ~]# ps axf

PID TTY STAT TIME COMMAND

2 ? S 0:00 [kthreadd]

...lots of processes...

1362 ? Ssl 0:35 corosync

1379 ? Ss 0:00 /usr/sbin/pacemakerd -f

1380 ? Ss 0:00 \\_ /usr/libexec/pacemaker/cib

1381 ? Ss 0:00 \\_ /usr/libexec/pacemaker/stonithd

1382 ? Ss 0:00 \\_ /usr/libexec/pacemaker/lrmd

1383 ? Ss 0:00 \\_ /usr/libexec/pacemaker/attrd

1384 ? Ss 0:00 \\_ /usr/libexec/pacemaker/pengine

1385 ? Ss 0:00 \\_ /usr/libexec/pacemaker/crmd

If that looks OK, check the pcs status output:

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

WARNING: no stonith devices and stonith-enabled is not false

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 16:37:34 2018

Last change: Mon Sep 10 16:30:53 2018 by hacluster via crmd on pcmk-2

2 nodes configured

0 resources configured

Online: [ pcmk-1 pcmk-2 ]

No resources

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

Finally, ensure there are no start-up errors from corosync or pacemaker (aside from messages relating to not having STONITH configured, which are OK at this point):

[root@pcmk-1 ~]# journalctl -b | grep -i error

# Create an Active/Passive Cluster

When Pacemaker starts up, it automatically records the number and details of the nodes in the cluster, as well as which stack is being used and the version of Pacemaker being used.

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

**WARNING:** no stonith devices and stonith-enabled is not false

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 16:41:46 2018

Last change: Mon Sep 10 16:30:53 2018 by hacluster via crmd on pcmk-2

2 nodes configured

0 resources configured

Online: [ pcmk-1 pcmk-2 ]

As you can see, the tool has found some WARNING.

In order to guarantee the safety of your data, [[5]](https://clusterlabs.org/pacemaker/doc/deprecated/en-US/Pacemaker/1.1/html/Clusters_from_Scratch/ch05.html" \l "ftn.idm140603909044240) fencing (also called STONITH) is enabled by default. However, it also knows when no STONITH configuration has been supplied and reports this as a problem (since the cluster will not be able to make progress if a situation requiring node fencing arises).

We will disable this feature for now and configure it later.

To disable STONITH, set the **stonith-enabled** cluster option to false:

[root@pcmk-1 ~]# pcs property set stonith-enabled=false

[root@pcmk-1 ~]# crm\_verify -L

**NOTE:** The use of **stonith-enabled=false** is completely inappropriate for a production cluster. It tells the cluster to simply pretend that failed nodes are safely powered off.

## Add a Resource

Our first resource will be a unique IP address that the cluster can bring up on either node. Regardless of where any cluster service(s) are running, end users need a consistent address to contact them on. Here, I will choose 192.168.122.120 as the floating address, give it the imaginative name ClusterIP and tell the cluster to check whether it is running every 30 seconds.

[root@pcmk-1 ~]# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 \

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

Another important piece of information here is ocf:heartbeat:IPaddr2. This tells Pacemaker three things about the resource you want to add:

**The first field** (ocf in this case) is the standard to which the resource script conforms and where to find it.

**The second field** (heartbeat in this case) is standard-specific; for OCF resources, it tells the cluster which OCF namespace the resource script is in.

**The third field** (IPaddr2 in this case) is the name of the resource script.

To obtain a list of the available resource standards (the ocf part of ocf:heartbeat:IPaddr2), run:

[root@pcmk-1 ~]# pcs resource standards

lsb

ocf

service

Systemd

To obtain a list of the available OCF resource providers (the **heartbeat** part of **ocf:heartbeat:IPaddr2**), run:

[root@pcmk-1 ~]# pcs resource providers

heartbeat

openstack

pacemaker

Finally, if you want to see all the resource agents available for a specific OCF provider (the **IPaddr2** part of **ocf:heartbeat:IPaddr2**), run:

[root@pcmk-1 ~]# pcs resource agents ocf:heartbeat

apache

aws-vpc-move-ip

awseip

awsvip

azure-lb

Clvm

.

. (skipping lots of resources to save space)

.

symlink

tomcat

VirtualDomain

Xinetd

Now, verify that the IP resource has been added, and display the cluster’s status to see that it is now active:

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 16:55:26 2018

Last change: Mon Sep 10 16:53:42 2018 by root via cibadmin on pcmk-1

2 nodes configured

1 resource configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

## Perform a Failover

Since our ultimate goal is high availability, we should test failover of our new resource before moving on.

First, find the node on which the IP address is running.

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 16:55:26 2018

Last change: Mon Sep 10 16:53:42 2018 by root via cibadmin on pcmk-1

2 nodes configured

1 resource configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

You can see that the status of the ClusterIP resource is Started on a particular node (in this example, pcmk-1). Shut down Pacemaker and Corosync on that machine to trigger a failover.

[root@pcmk-1 ~]# pcs cluster stop pcmk-1

Stopping Cluster (pacemaker)...

Stopping Cluster (corosync)…

[root@pcmk-1 ~]# pcs status

Error: cluster is not currently running on this node

Go to the other node, and check the cluster status.

[root@pcmk-2 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 16:57:22 2018

Last change: Mon Sep 10 16:53:42 2018 by root via cibadmin on pcmk-1

2 nodes configured

1 resource configured

Online: [ pcmk-2 ]

OFFLINE: [ pcmk-1 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-2

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

Notice that **pcmk-1** is **OFFLINE** for cluster purposes (its **pcsd** is still active, allowing it to receive **pcs** commands, but it is not participating in the cluster).

Also notice that **ClusterIP** is now running on **pcmk-2** — failover happened automatically, and no errors are reported.

## Prevent Resources from Moving after Recovery

In most circumstances, it is highly desirable to prevent healthy resources from being moved around the cluster. Moving resources almost always requires a period of downtime. For complex services such as databases, this period can be quite long.

To address this, Pacemaker has the concept of resource stickiness, which controls how strongly a service prefers to stay running where it is. You may like to think of it as the "cost" of any downtime. By default, Pacemaker assumes there is zero cost associated with moving resources and will do so to achieve "optimal" [[6]](https://clusterlabs.org/pacemaker/doc/deprecated/en-US/Pacemaker/1.1/html/Clusters_from_Scratch/_prevent_resources_from_moving_after_recovery.html" \l "ftn.idm140603923945488) resource placement. We can specify a different stickiness for every resource, but it is often sufficient to change the default.

[root@pcmk-1 ~]# pcs resource defaults resource-stickiness=100

Warning: Defaults do not apply to resources which override them with their own defined values

[root@pcmk-1 ~]# pcs resource defaults

resource-stickiness: 100

# Add Apache HTTP Server as a Cluster Service

# yum install -y httpd wget

# firewall-cmd --permanent --add-service=http

# firewall-cmd --reload

# cat <<-END >/var/www/html/index.html

<html>

<body>My Test Site - $(hostname)</body>

</html>

END

# cat <<-END >/etc/httpd/conf.d/status.conf

<Location /server-status>

SetHandler server-status

Require local

</Location>

END

At this point, Apache is ready to go, and all that needs to be done is to add it to the cluster. Let’s call the resource WebSite.

[root@pcmk-1 ~]# pcs resource create WebSite ocf:heartbeat:apache \

configfile=/etc/httpd/conf/httpd.conf \

statusurl="http://localhost/server-status" \

op monitor interval=1min

[root@pcmk-1 ~]# pcs resource op defaults timeout=240s

Warning: Defaults do not apply to resources which override them with their own defined values

[root@pcmk-1 ~]# pcs resource op defaults

timeout: 240s

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 17:06:22 2018

Last change: Mon Sep 10 17:05:41 2018 by root via cibadmin on pcmk-1

2 nodes configured

2 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-2

WebSite (ocf::heartbeat:apache): Started pcmk-1

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

wget -O - <http://localhost/server-status>

If you see Not Found or Forbidden in the output, then this is likely the problem. Ensure that the <Location /server-status> block is correct.

## Ensure Resources Run on the Same Host

To reduce the load on any one machine, Pacemaker will generally try to spread the configured resources across the cluster nodes. However, we can tell the cluster that two resources are related and need to run on the same host (or not at all). Here, we instruct the cluster that WebSite can only run on the host that ClusterIP is active on.

To achieve this, we use a colocation constraint that indicates it is mandatory for WebSite to run on the same node as ClusterIP. The "mandatory" part of the colocation constraint is indicated by using a score of INFINITY. The INFINITY score also means that if ClusterIP is not active anywhere, WebSite will not be permitted to run.

**Note:** If ClusterIP is not active anywhere, WebSite will not be permitted to run anywhere.

[root@pcmk-1 ~]# pcs constraint colocation add WebSite with ClusterIP INFINITY

[root@pcmk-1 ~]# pcs constraint

Location Constraints:

Ordering Constraints:

Colocation Constraints:

WebSite with ClusterIP (score:INFINITY)

Ticket Constraints:

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 17:08:54 2018

Last change: Mon Sep 10 17:08:27 2018 by root via cibadmin on pcmk-1

2 nodes configured

2 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-2

WebSite (ocf::heartbeat:apache): Started pcmk-2

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

## Ensure Resources Start and Stop in Order

Like many services, Apache can be configured to bind to specific IP addresses on a host or to the wildcard IP address.

All order constraints are mandatory, which means that the recovery of Cluster-IP will also trigger the recovery of WebSite.

[root@pcmk-1 ~]# pcs constraint order ClusterIP then WebSite

Adding ClusterIP WebSite (kind: Mandatory) (Options: first-action=start then-action=start)

[root@pcmk-1 ~]# pcs constraint

Location Constraints:

Ordering Constraints:

start ClusterIP then start WebSite (kind:Mandatory)

Colocation Constraints:

WebSite with ClusterIP (score:INFINITY)

Ticket Constraints:

## Prefer One Node Over Another

Pacemaker does not rely on any sort of hardware symmetry between nodes, so it may well be that one machine is more powerful than the other.

In such cases, you may want to host the resources on the more powerful node when it is available, to have the best performance — or you may want to host the resources on the less powerful node when it’s available, so you don’t have to worry about whether you can handle the load after a failover.

To do this, we create a location constraint.

In the location constraint below, we are saying the WebSite resource prefers the node pcmk-1 with a score of 50. Here, the score indicates how strongly we’d like the resource to run at this location.

[root@pcmk-1 ~]# pcs constraint location WebSite prefers pcmk-1=50

[root@pcmk-1 ~]# pcs constraint

Location Constraints:

Resource: WebSite

Enabled on: pcmk-1 (score:50)

Ordering Constraints:

start ClusterIP then start WebSite (kind:Mandatory)

Colocation Constraints:

WebSite with ClusterIP (score:INFINITY)

Ticket Constraints:

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 17:21:41 2018

Last change: Mon Sep 10 17:21:14 2018 by root via cibadmin on pcmk-1

2 nodes configured

2 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-2

WebSite (ocf::heartbeat:apache): Started pcmk-2

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

Wait a minute, the resources are still on pcmk-2!

Even though WebSite now prefers to run on pcmk-1, that preference is (intentionally) less than the resource stickiness (how much we preferred not to have unnecessary downtime).

To see the current placement scores, you can use a tool called crm\_simulate.

[root@pcmk-1 ~]# crm\_simulate -sL

Current cluster status:

Online: [ pcmk-1 pcmk-2 ]

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-2

WebSite (ocf::heartbeat:apache): Started pcmk-2

Allocation scores:

native\_color: ClusterIP allocation score on pcmk-1: 50

native\_color: ClusterIP allocation score on pcmk-2: 200

native\_color: WebSite allocation score on pcmk-1: -INFINITY

native\_color: WebSite allocation score on pcmk-2: 100

Transition Summary:

## Move Resources Manually

There are always times when an administrator needs to override the cluster and force resources to move to a specific location. In this example, we will force the WebSite to move to pcmk-1.

[root@pcmk-1 ~]# pcs resource move WebSite pcmk-1

[root@pcmk-1 ~]# pcs constraint

Location Constraints:

Resource: WebSite

Enabled on: pcmk-1 (score:50)

Enabled on: pcmk-1 (score:INFINITY) (role: Started)

Ordering Constraints:

start ClusterIP then start WebSite (kind:Mandatory)

Colocation Constraints:

WebSite with ClusterIP (score:INFINITY)

Ticket Constraints:

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 17:28:55 2018

Last change: Mon Sep 10 17:28:27 2018 by root via crm\_resource on pcmk-1

2 nodes configured

2 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

WebSite (ocf::heartbeat:apache): Started pcmk-1

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

Once we’ve finished whatever activity required us to move the resources to pcmk-1 (in our case nothing), we can then allow the cluster to resume normal operation by removing the new constraint. Due to our first location constraint and our default stickiness, the resources will remain on pcmk-1.

We will use the **pcs resource clear** command, which removes all temporary constraints previously created by **pcs resource move** or **pcs resource ban**.

[root@pcmk-1 ~]# pcs resource clear WebSite

[root@pcmk-1 ~]# pcs constraint

Location Constraints:

Resource: WebSite

Enabled on: pcmk-1 (score:50)

Ordering Constraints:

start ClusterIP then start WebSite (kind:Mandatory)

Colocation Constraints:

WebSite with ClusterIP (score:INFINITY)

Ticket Constraints:

Note that the INFINITY location constraint is now gone. If we check the cluster status, we can also see that (as expected) the resources are still active on pcmk-1.

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 17:31:47 2018

Last change: Mon Sep 10 17:31:04 2018 by root via crm\_resource on pcmk-1

2 nodes configured

2 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

WebSite (ocf::heartbeat:apache): Started pcmk-1

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

To remove the constraint with the score of 50, we would first get the constraint’s ID using pcs constraint --full, then remove it with pcs constraint remove and the ID. We won’t show those steps here, but feel free to try it on your own, with the help of the pcs man page if necessary.

# Replicate Storage Using DRBD

# rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org

# rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm

Retrieving http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm

Preparing... ################################# [100%]

Updating / installing...

1:elrepo-release-7.0-3.el7.elrepo ################################# [100%]

Now, we can install the DRBD kernel module and utilities:

# yum install -y kmod-drbd84 drbd84-utils

DRBD will not be able to run under the default SELinux security policies. If you are familiar with SELinux, you can modify the policies in a more fine-grained manner, but here we will simply exempt DRBD processes from SELinux control:

# semanage permissive -a drbd\_t

We will configure DRBD to use port 7789, so allow that port from each host to the other:

[root@pcmk-1 ~]# firewall-cmd --permanent --add-rich-rule='rule family="ipv4" \

source address="192.168.122.102" port port="7789" protocol="tcp" accept'

success

[root@pcmk-1 ~]# firewall-cmd --reload

success

[root@pcmk-2 ~]# firewall-cmd --permanent --add-rich-rule='rule family="ipv4" \

source address="192.168.122.101" port port="7789" protocol="tcp" accept'

success

[root@pcmk-2 ~]# firewall-cmd --reload

success

## Allocate a Disk Volume for DRBD

DRBD will need its own block device on each node. This can be a physical disk partition or logical volume, of whatever size you need for your data. For this document, we will use a 512MiB logical volume, which is more than sufficient for a single HTML file and (later) GFS2 metadata.

[root@pcmk-1 ~]# vgdisplay | grep -e Name -e Free

VG Name centos\_pcmk-1

Free PE / Size 255 / 1020.00 MiB

[root@pcmk-1 ~]# lvcreate --name drbd-demo --size 512M centos\_pcmk-1

Logical volume "drbd-demo" created.

[root@pcmk-1 ~]# lvs

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert

drbd-demo centos\_pcmk-1 -wi-a----- 512.00m

root centos\_pcmk-1 -wi-ao---- 3.00g

swap centos\_pcmk-1 -wi-ao---- 1.00g

Repeat for the second node, making sure to use the same size:

[root@pcmk-1 ~]# ssh pcmk-2 -- lvcreate --name drbd-demo --size 512M centos\_pcmk-2

Logical volume "drbd-demo" created.

**Create LVM shared storage**

# pvcreate /dev/sdb

# vgcreate vg01 /dev/sdb

# vgdisplay vg01

# lvcreate –L 5G(size) –n drbd(name) vg01

# vgdisplay | grep -e Name -e Free

# lvcreate -L 20G -n drbd centos

## Configure DRBD

# cat <<END >/etc/drbd.d/wwwdata.res

resource wwwdata {

protocol C;

meta-disk internal;

device /dev/drbd1;

syncer {

verify-alg sha1;

}

net {

allow-two-primaries;

}

on pcmk-1 {

disk /dev/centos\_pcmk-1/drbd-demo;

address 192.168.122.101:7789;

}

on pcmk-2 {

disk /dev/centos\_pcmk-2/drbd-demo;

address 192.168.122.102:7789;

}

}

END

## Initialize DRBD

With the configuration in place, we can now get DRBD running.

These commands create the local metadata for the DRBD resource, ensure the DRBD kernel module is loaded, and bring up the DRBD resource. Run them on one node:

[root@pcmk-1 ~]# drbdadm create-md wwwdata

--== Thank you for participating in the global usage survey ==--

The server's response is:

you are the 2147th user to install this version

initializing activity log

initializing bitmap (16 KB) to all zero

Writing meta data...

New drbd meta data block successfully created.

success

[root@pcmk-1 ~]# modprobe drbd

[root@pcmk-1 ~]# drbdadm up wwwdata

--== Thank you for participating in the global usage survey ==--

The server's response is:

We can confirm DRBD’s status on this node:

[root@pcmk-1 ~]# cat /proc/drbd

version: 8.4.11-1 (api:1/proto:86-101)

GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@, 2018-04-26 12:10:42

1: cs:WFConnection ro:Secondary/Unknown ds:Inconsistent/DUnknown C r----s

ns:0 nr:0 dw:0 dr:0 al:8 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:524236

Because we have not yet initialized the data, this node’s data is marked as Inconsistent. Because we have not yet initialized the second node, the local state is WFConnection (waiting for connection), and the partner node’s status is marked as Unknown.

Now, repeat the above commands on the second node, starting with creating wwwdata.res. After giving it time to connect, when we check the status, it shows:

[root@pcmk-2 ~]# cat /proc/drbd

version: 8.4.11-1 (api:1/proto:86-101)

GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@, 2018-04-26 12:10:42

1: cs:Connected ro:Secondary/Secondary ds:Inconsistent/Inconsistent C r-----

ns:0 nr:0 dw:0 dr:0 al:8 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:524236

You can see the state has changed to Connected, meaning the two DRBD nodes are communicating properly, and both nodes are in Secondary role with Inconsistent data.

To make the data consistent, we need to tell DRBD which node should be considered to have the correct data. In this case, since we are creating a new resource, both have garbage, so we’ll just pick pcmk-1 and run this command on it:

[root@pcmk-1 ~]# drbdadm primary --force wwwdata

After a while, the sync should finish, and you’ll see something like:

[root@pcmk-1 ~]# cat /proc/drbd

version: 8.4.11-1 (api:1/proto:86-101)

GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@, 2018-04-26 12:10:42

1: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----

ns:524236 nr:0 dw:0 dr:526364 al:8 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0

Both sets of data are now UpToDate, and we can proceed to creating and populating a filesystem for our WebSite resource’s documents.

## Populate the DRBD Disk

On the node with the primary role (pcmk-1 in this example), create a filesystem on the DRBD device:

[root@pcmk-1 ~]# mkfs.xfs /dev/drbd1

meta-data=/dev/drbd1 isize=512 agcount=4, agsize=32765 blks

= sectsz=512 attr=2, projid32bit=1

= crc=1 finobt=0, sparse=0

data = bsize=4096 blocks=131059, imaxpct=25

= sunit=0 swidth=0 blks

naming =version 2 bsize=4096 ascii-ci=0 ftype=1

log =internal log bsize=4096 blocks=855, version=2

= sectsz=512 sunit=0 blks, lazy-count=1

realtime =none extsz=4096 blocks=0, rtextents=0

Mount the newly created filesystem, populate it with our web document, give it the same SELinux policy as the web document root, then unmount it (the cluster will handle mounting and unmounting it later):

[root@pcmk-1 ~]# mount /dev/drbd1 /mnt

[root@pcmk-1 ~]# cat <<-END >/mnt/index.html

<html>

<body>My Test Site - DRBD</body>

</html>

END

[root@pcmk-1 ~]# chcon -R --reference=/var/www/html /mnt

[root@pcmk-1 ~]# umount /dev/drbd1

## Configure the Cluster for the DRBD device

One handy feature pcs has is the ability to queue up several changes into a file and commit those changes all at once. To do this, start by populating the file with the current raw XML config from the CIB.

[root@pcmk-1 ~]# pcs cluster cib drbd\_cfg

Using pcs’s -f option, make changes to the configuration saved in the drbd\_cfg file. These changes will not be seen by the cluster until the drbd\_cfg file is pushed into the live cluster’s CIB later.

Here, we create a cluster resource for the DRBD device, and an additional clone resource to allow the resource to run on both nodes at the same time.

[root@pcmk-1 ~]# pcs -f drbd\_cfg resource create WebData ocf:linbit:drbd \

drbd\_resource=wwwdata op monitor interval=60s

[root@pcmk-1 ~]# pcs -f drbd\_cfg resource master WebDataClone WebData \

master-max=1 master-node-max=1 clone-max=2 clone-node-max=1 \

notify=true

[root@pcmk-1 ~]# pcs -f drbd\_cfg resource show

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

WebSite (ocf::heartbeat:apache): Started pcmk-1

Master/Slave Set: WebDataClone [WebData]

Stopped: [ pcmk-1 pcmk-2 ]

After you are satisfied with all the changes, you can commit them all at once by pushing the drbd\_cfg file into the live CIB.

[root@pcmk-1 ~]# pcs cluster cib-push drbd\_cfg --config

CIB updated

Let’s see what the cluster did with the new configuration:

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 17:58:07 2018

Last change: Mon Sep 10 17:57:53 2018 by root via cibadmin on pcmk-1

2 nodes configured

4 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

WebSite (ocf::heartbeat:apache): Started pcmk-1

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 ]

Slaves: [ pcmk-2 ]

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

We can see that WebDataClone (our DRBD device) is running as master (DRBD’s primary role) on pcmk-1 and slave (DRBD’s secondary role) on pcmk-2.

## Configure the Cluster for the Filesystem

Now that we have a working DRBD device, we need to mount its filesystem.

we will queue our changes to a file and then push the new configuration to the cluster as the final step.

[root@pcmk-1 ~]# pcs cluster cib fs\_cfg

[root@pcmk-1 ~]# pcs -f fs\_cfg resource create WebFS Filesystem \

device="/dev/drbd1" directory="/var/www/html" fstype="xfs"

Assumed agent name 'ocf:heartbeat:Filesystem' (deduced from 'Filesystem')

[root@pcmk-1 ~]# pcs -f fs\_cfg constraint colocation add \

WebFS with WebDataClone INFINITY with-rsc-role=Master

[root@pcmk-1 ~]# pcs -f fs\_cfg constraint order \

promote WebDataClone then start WebFS

Adding WebDataClone WebFS (kind: Mandatory) (Options: first-action=promote then-action=start)

We also need to tell the cluster that Apache needs to run on the same machine as the filesystem and that it must be active before Apache can start.

[root@pcmk-1 ~]# pcs -f fs\_cfg constraint colocation add WebSite with WebFS INFINITY

[root@pcmk-1 ~]# pcs -f fs\_cfg constraint order WebFS then WebSite

Adding WebFS WebSite (kind: Mandatory) (Options: first-action=start then-action=start)

Review the updated configuration.

[root@pcmk-1 ~]# pcs -f fs\_cfg constraint

Location Constraints:

Resource: WebSite

Enabled on: pcmk-1 (score:50)

Ordering Constraints:

start ClusterIP then start WebSite (kind:Mandatory)

promote WebDataClone then start WebFS (kind:Mandatory)

start WebFS then start WebSite (kind:Mandatory)

Colocation Constraints:

WebSite with ClusterIP (score:INFINITY)

WebFS with WebDataClone (score:INFINITY) (with-rsc-role:Master)

WebSite with WebFS (score:INFINITY)

Ticket Constraints:

[root@pcmk-1 ~]# pcs -f fs\_cfg resource show

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

WebSite (ocf::heartbeat:apache): Started pcmk-1

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 ]

Slaves: [ pcmk-2 ]

WebFS (ocf::heartbeat:Filesystem): Stopped

After reviewing the new configuration, upload it and watch the cluster put it into effect.

[root@pcmk-1 ~]# pcs cluster cib-push fs\_cfg --config

CIB updated

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 18:02:24 2018

Last change: Mon Sep 10 18:02:14 2018 by root via cibadmin on pcmk-1

2 nodes configured

5 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

WebSite (ocf::heartbeat:apache): Started pcmk-1

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 ]

Slaves: [ pcmk-2 ]

WebFS (ocf::heartbeat:Filesystem): Started pcmk-1

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

## Test Cluster Failover

Previously, we used pcs cluster stop pcmk-1 to stop all cluster services on pcmk-1, failing over the cluster resources, but there is another way to safely simulate node failure.

We can put the node into standby mode. Nodes in this state continue to run corosync and pacemaker but are not allowed to run resources.

This feature can be particularly useful when performing system administration tasks such as updating packages used by cluster resources.

Put the active node into standby mode, and observe the cluster move all the resources to the other node. The node’s status will change to indicate that it can no longer host resources, and eventually all the resources will move.

[root@pcmk-1 ~]# pcs cluster standby pcmk-1

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 18:04:22 2018

Last change: Mon Sep 10 18:03:43 2018 by root via cibadmin on pcmk-1

2 nodes configured

5 resources configured

Node pcmk-1: standby

Online: [ pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-2

WebSite (ocf::heartbeat:apache): Started pcmk-2

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-2 ]

Stopped: [ pcmk-1 ]

WebFS (ocf::heartbeat:Filesystem): Started pcmk-2

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

Once we’ve done everything we needed to on pcmk-1 (in this case nothing, we just wanted to see the resources move), we can allow the node to be a full cluster member again.

[root@pcmk-1 ~]# pcs cluster unstandby pcmk-1

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-2 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Mon Sep 10 18:05:22 2018

Last change: Mon Sep 10 18:05:21 2018 by root via cibadmin on pcmk-1

2 nodes configured

5 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-2

WebSite (ocf::heartbeat:apache): Started pcmk-2

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-2 ]

Slaves: [ pcmk-1 ]

WebFS (ocf::heartbeat:Filesystem): Started pcmk-2

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

Notice that pcmk-1 is back to the Online state, and that the cluster resources stay where they are due to our resource stickiness settings configured earlier.

# Configure STONITH

STONITH (Shoot The Other Node In The Head aka. fencing) protects your data from being corrupted by rogue nodes or unintended concurrent access.

Just because a node is unresponsive doesn’t mean it has stopped accessing your data. The only way to be 100% sure that your data is safe, is to use STONITH to ensure that the node is truly offline before allowing the data to be accessed from another node.

STONITH also has a role to play in the event that a clustered service cannot be stopped. In this case, the cluster uses STONITH to force the whole node offline, thereby making it safe to start the service elsewhere.

It is crucial that your STONITH device can allow the cluster to differentiate between a node failure and a network failure.

A common mistake people make when choosing a STONITH device is to use a remote power switch (such as many on-board IPMI controllers) that shares power with the node it controls. If the power fails in such a case, the cluster cannot be sure whether the node is really offline, or active and suffering from a network fault, so the cluster will stop all resources to avoid a possible split-brain situation.

Likewise, any device that relies on the machine being active (such as SSH-based "devices" sometimes used during testing) is inappropriate.

## Configure the Cluster for STONITH

* Install the STONITH agent(s). To see what packages are available, run yum search fence-. Be sure to install the package(s) on all cluster nodes.
* Configure the STONITH device itself to be able to fence your nodes and accept fencing requests. This includes any necessary configuration on the device and on the nodes, and any firewall or SELinux changes needed. Test the communication between the device and your nodes.
* Find the correct STONITH agent script: pcs stonith list
* Find the parameters associated with the device: pcs stonith describe agent\_name
* Create a local copy of the CIB: pcs cluster cib stonith\_cfg
* Create the fencing resource: pcs -f stonith\_cfg stonith create stonith\_id stonith\_device\_type [stonith\_device\_options]
* Any flags that do not take arguments, such as --ssl, should be passed as ssl=1.
* Enable STONITH in the cluster: pcs -f stonith\_cfg property set stonith-enabled=true
* If the device does not know how to fence nodes based on their uname, you may also need to set the special pcmk\_host\_map parameter. See man stonithd for details.
* If the device does not support the list command, you may also need to set the special pcmk\_host\_list and/or pcmk\_host\_check parameters. See man stonithd for details.
* If the device does not expect the victim to be specified with the port parameter, you may also need to set the special pcmk\_host\_argument parameter. See man stonithd for details.
* Commit the new configuration: pcs cluster cib-push stonith\_cfg

Once the STONITH resource is running, test it (you might want to stop the cluster on that machine first): stonith\_admin --reboot nodename

## Example

For this example, assume we have a chassis containing four nodes and an IPMI device active on 10.0.0.1. Following the steps above would go something like this:

Step 1: Install the fence-agents-ipmilan package on both nodes.

Step 2: Configure the IP address, authentication credentials, etc. in the IPMI device itself.

Step 3: Choose the fence\_ipmilan STONITH agent.

Step 4: Obtain the agent’s possible parameters:

[root@pcmk-1 ~]# pcs stonith describe fence\_ipmilan

fence\_ipmilan - Fence agent for IPMI

fence\_ipmilan is an I/O Fencing agentwhich can be used with machines controlled by IPMI.This agent calls support software ipmitool (http://ipmitool.sf.net/). WARNING! This fence agent might report success before the node is powered off. You should use -m/method onoff if your fence device works correctly with that option.

Stonith options:

ipport: TCP/UDP port to use for connection with device

hexadecimal\_kg: Hexadecimal-encoded Kg key for IPMIv2 authentication

port: IP address or hostname of fencing device (together with --port-as-ip)

inet6\_only: Forces agent to use IPv6 addresses only

ipaddr: IP Address or Hostname

passwd\_script: Script to retrieve password

method: Method to fence (onoff|cycle)

inet4\_only: Forces agent to use IPv4 addresses only

passwd: Login password or passphrase

lanplus: Use Lanplus to improve security of connection

auth: IPMI Lan Auth type.

cipher: Ciphersuite to use (same as ipmitool -C parameter)

target: Bridge IPMI requests to the remote target address

privlvl: Privilege level on IPMI device

timeout: Timeout (sec) for IPMI operation

login: Login Name

verbose: Verbose mode

debug: Write debug information to given file

power\_wait: Wait X seconds after issuing ON/OFF

login\_timeout: Wait X seconds for cmd prompt after login

delay: Wait X seconds before fencing is started

power\_timeout: Test X seconds for status change after ON/OFF

ipmitool\_path: Path to ipmitool binary

shell\_timeout: Wait X seconds for cmd prompt after issuing command

port\_as\_ip: Make "port/plug" to be an alias to IP address

retry\_on: Count of attempts to retry power on

sudo: Use sudo (without password) when calling 3rd party sotfware.

priority: The priority of the stonith resource. Devices are tried in order of highest priority to lowest.

pcmk\_host\_map: A mapping of host names to ports numbers for devices that do not support host names. Eg. node1:1;node2:2,3 would tell the cluster to use port 1 for node1 and ports 2 and

3 for node2

pcmk\_host\_list: A list of machines controlled by this device (Optional unless pcmk\_host\_check=static-list).

pcmk\_host\_check: How to determine which machines are controlled by the device. Allowed values: dynamic-list (query the device), static-list (check the pcmk\_host\_list attribute), none

(assume every device can fence every machine)

pcmk\_delay\_max: Enable a random delay for stonith actions and specify the maximum of random delay. This prevents double fencing when using slow devices such as sbd. Use this to enable a

random delay for stonith actions. The overall delay is derived from this random delay value adding a static delay so that the sum is kept below the maximum delay.

pcmk\_delay\_base: Enable a base delay for stonith actions and specify base delay value. This prevents double fencing when different delays are configured on the nodes. Use this to enable

a static delay for stonith actions. The overall delay is derived from a random delay value adding this static delay so that the sum is kept below the maximum delay.

pcmk\_action\_limit: The maximum number of actions can be performed in parallel on this device Pengine property concurrent-fencing=true needs to be configured first. Then use this to

specify the maximum number of actions can be performed in parallel on this device. -1 is unlimited.

Default operations:

monitor: interval=60s

Step 5: pcs cluster cib stonith\_cfg

Step 6: Here are example parameters for creating our STONITH resource:

[root@pcmk-1 ~]# pcs -f stonith\_cfg stonith create ipmi-fencing fence\_ipmilan \

pcmk\_host\_list="pcmk-1 pcmk-2" ipaddr=10.0.0.1 login=testuser \

passwd=acd123 op monitor interval=60s

[root@pcmk-1 ~]# pcs -f stonith\_cfg stonith

ipmi-fencing (stonith:fence\_ipmilan): Stopped

Steps 7-10: Enable STONITH in the cluster:

[root@pcmk-1 ~]# pcs -f stonith\_cfg property set stonith-enabled=true

[root@pcmk-1 ~]# pcs -f stonith\_cfg property

Cluster Properties:

cluster-infrastructure: corosync

cluster-name: mycluster

dc-version: 1.1.18-11.el7\_5.3-2b07d5c5a9

have-watchdog: false

stonith-enabled: true

Step 11: pcs cluster cib-push stonith\_cfg --config

Step 12: Test:

[root@pcmk-1 ~]# pcs cluster stop pcmk-2

[root@pcmk-1 ~]# stonith\_admin --reboot pcmk-2

After a successful test, login to any rebooted nodes, and start the cluster (with pcs cluster start).

# Convert Cluster to Active/Active

The primary requirement for an Active/Active cluster is that the data required for your services is available, simultaneously, on both machines. Pacemaker makes no requirement on how this is achieved; you could use a SAN if you had one available, but since DRBD supports multiple Primaries, we can continue to use it here.

On both nodes, install the GFS2 command-line utilities and the Distributed Lock Manager (DLM) required by cluster filesystems:

# yum install -y gfs2-utils dlm

## Configure the Cluster for the DLM

The DLM needs to run on both nodes, so we’ll start by creating a resource for it (using the ocf:pacemaker:controld resource script), and clone it:

[root@pcmk-1 ~]# pcs cluster cib dlm\_cfg

[root@pcmk-1 ~]# pcs -f dlm\_cfg resource create dlm \

ocf:pacemaker:controld op monitor interval=60s

[root@pcmk-1 ~]# pcs -f dlm\_cfg resource clone dlm clone-max=2 clone-node-max=1

[root@pcmk-1 ~]# pcs -f dlm\_cfg resource show

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

WebSite (ocf::heartbeat:apache): Started pcmk-1

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 ]

Slaves: [ pcmk-2 ]

WebFS (ocf::heartbeat:Filesystem): Started pcmk-1

Clone Set: dlm-clone [dlm]

Stopped: [ pcmk-1 pcmk-2 ]

Activate our new configuration, and see how the cluster responds:

[root@pcmk-1 ~]# pcs cluster cib-push dlm\_cfg --config

CIB updated

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-1 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Tue Sep 11 10:18:30 2018

Last change: Tue Sep 11 10:16:49 2018 by hacluster via crmd on pcmk-2

2 nodes configured

8 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ipmi-fencing (stonith:fence\_ipmilan): Started pcmk-1

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

WebSite (ocf::heartbeat:apache): Started pcmk-1

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 ]

Slaves: [ pcmk-2 ]

WebFS (ocf::heartbeat:Filesystem): Started pcmk-1

Clone Set: dlm-clone [dlm]

Started: [ pcmk-1 pcmk-2 ]

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

## Create and Populate GFS2 Filesystem

Before we do anything to the existing partition, we need to make sure it is unmounted. We do this by telling the cluster to stop the WebFS resource. This will ensure that other resources (in our case, Apache) using WebFS are not only stopped, but stopped in the correct order.

[root@pcmk-1 ~]# pcs resource disable WebFS

[root@pcmk-1 ~]# pcs resource

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1

WebSite (ocf::heartbeat:apache): Stopped

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 ]

Slaves: [ pcmk-2 ]

WebFS (ocf::heartbeat:Filesystem): Stopped (disabled)

Clone Set: dlm-clone [dlm]

Started: [ pcmk-1 pcmk-2 ]

You can see that both Apache and WebFS have been stopped, and that pcmk-1 is the current master for the DRBD device.

Now we can create a new GFS2 filesystem on the DRBD device.

**Note:** This will erase all previous content stored on the DRBD device. Ensure you have a copy of any important data.

[root@pcmk-1 ~]# mkfs.gfs2 -p lock\_dlm -j 2 -t mycluster:web /dev/drbd1

It appears to contain an existing filesystem (xfs)

This will destroy any data on /dev/drbd1

Are you sure you want to proceed? [y/n] y

Discarding device contents (may take a while on large devices): Done

Adding journals: Done

Building resource groups: Done

Creating quota file: Done

Writing superblock and syncing: Done

Device: /dev/drbd1

Block size: 4096

Device size: 0.50 GB (131059 blocks)

Filesystem size: 0.50 GB (131056 blocks)

Journals: 2

Resource groups: 3

Locking protocol: "lock\_dlm"

Lock table: "mycluster:web"

UUID: 0bcbffab-cada-4105-94d1-be8a26669ee0

The mkfs.gfs2 command required a number of additional parameters:

-p lock\_dlm specifies that we want to use the kernel’s DLM.

-j 2 indicates that the filesystem should reserve enough space for two journals (one for each node that will access the filesystem).

-t mycluster:web specifies the lock table name. The format for this field is clustername:fsname.

For clustername, we need to use the same value we specified originally with pcs cluster setup --name (which is also the value of cluster\_name in /etc/corosync/corosync.conf). If you are unsure what your cluster name is, you can look in /etc/corosync/corosync.conf or execute the command pcs cluster corosync pcmk-1 | grep cluster\_name.

Now we can (re-)populate the new filesystem with data (web pages). We’ll create yet another variation on our home page.

[root@pcmk-1 ~]# mount /dev/drbd1 /mnt

[root@pcmk-1 ~]# cat <<-END >/mnt/index.html

<html>

<body>My Test Site - GFS2</body>

</html>

END

[root@pcmk-1 ~]# chcon -R --reference=/var/www/html /mnt

[root@pcmk-1 ~]# umount /dev/drbd1

[root@pcmk-1 ~]# drbdadm verify wwwdata

## Reconfigure the Cluster for GFS2

With the WebFS resource stopped, let’s update the configuration.

[root@pcmk-1 ~]# pcs resource show WebFS

Resource: WebFS (class=ocf provider=heartbeat type=Filesystem)

Attributes: device=/dev/drbd1 directory=/var/www/html fstype=xfs

Meta Attrs: target-role=Stopped

Operations: monitor interval=20 timeout=40 (WebFS-monitor-interval-20)

notify interval=0s timeout=60 (WebFS-notify-interval-0s)

start interval=0s timeout=60 (WebFS-start-interval-0s)

stop interval=0s timeout=60 (WebFS-stop-interval-0s)

The fstype option needs to be updated to gfs2 instead of xfs.

[root@pcmk-1 ~]# pcs resource update WebFS fstype=gfs2

[root@pcmk-1 ~]# pcs resource show WebFS

Resource: WebFS (class=ocf provider=heartbeat type=Filesystem)

Attributes: device=/dev/drbd1 directory=/var/www/html fstype=gfs2

Meta Attrs: target-role=Stopped

Operations: monitor interval=20 timeout=40 (WebFS-monitor-interval-20)

notify interval=0s timeout=60 (WebFS-notify-interval-0s)

start interval=0s timeout=60 (WebFS-start-interval-0s)

stop interval=0s timeout=60 (WebFS-stop-interval-0s)

GFS2 requires that DLM be running, so we also need to set up new colocation and ordering constraints for it:

[root@pcmk-1 ~]# pcs constraint colocation add WebFS with dlm-clone INFINITY

[root@pcmk-1 ~]# pcs constraint order dlm-clone then WebFS

Adding dlm-clone WebFS (kind: Mandatory) (Options: first-action=start then-action=start)

## Clone the IP address

There’s no point making the services active on both locations if we can’t reach them both, so let’s clone the IP address.

The IPaddr2 resource agent has built-in intelligence for when it is configured as a clone. It will utilize a multicast MAC address to have the local switch send the relevant packets to all nodes in the cluster, together with iptables clusterip rules on the nodes so that any given packet will be grabbed by exactly one node. This will give us a simple but effective form of load-balancing requests between our two nodes.

[root@pcmk-1 ~]# pcs cluster cib loadbalance\_cfg

[root@pcmk-1 ~]# pcs -f loadbalance\_cfg resource clone ClusterIP \

clone-max=2 clone-node-max=2 globally-unique=true

* clone-max=2 tells the resource agent to split packets this many ways. This should equal the number of nodes that can host the IP.
* clone-node-max=2 says that one node can run up to 2 instances of the clone. This should also equal the number of nodes that can host the IP, so that if any node goes down, another node can take over the failed node’s "request bucket". Otherwise, requests intended for the failed node would be discarded.
* globally-unique=true tells the cluster that one clone isn’t identical to another (each handles a different "bucket"). This also tells the resource agent to insert iptables rules so each host only processes packets in its bucket(s).

Notice that when the ClusterIP becomes a clone, the constraints referencing ClusterIP now reference the clone. This is done automatically by pcs.

[root@pcmk-1 ~]# pcs -f loadbalance\_cfg constraint

Location Constraints:

Ordering Constraints:

start ClusterIP-clone then start WebSite (kind:Mandatory)

promote WebDataClone then start WebFS (kind:Mandatory)

start WebFS then start WebSite (kind:Mandatory)

start dlm-clone then start WebFS (kind:Mandatory)

Colocation Constraints:

WebSite with ClusterIP-clone (score:INFINITY)

WebFS with WebDataClone (score:INFINITY) (with-rsc-role:Master)

WebSite with WebFS (score:INFINITY)

WebFS with dlm-clone (score:INFINITY)

Ticket Constraints:

Now we must tell the resource how to decide which requests are processed by which hosts. To do this, we specify the clusterip\_hash parameter. The value of sourceip means that the source IP address of incoming packets will be hashed; each node will process a certain range of hashes.

[root@pcmk-1 ~]# pcs -f loadbalance\_cfg resource update ClusterIP clusterip\_hash=sourceip

Load our configuration to the cluster, and see how it responds.

[root@pcmk-1 ~]# pcs cluster cib-push loadbalance\_cfg --config

CIB updated

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-1 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Tue Sep 11 10:36:38 2018

Last change: Tue Sep 11 10:36:33 2018 by root via cibadmin on pcmk-1

2 nodes configured

9 resources configured (1 DISABLED)

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ipmi-fencing (stonith:fence\_ipmilan): Started pcmk-1

WebSite (ocf::heartbeat:apache): Stopped

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 ]

Slaves: [ pcmk-2 ]

WebFS (ocf::heartbeat:Filesystem): Stopped (disabled)

Clone Set: dlm-clone [dlm]

Started: [ pcmk-1 pcmk-2 ]

Clone Set: ClusterIP-clone [ClusterIP] (unique)

ClusterIP:0 (ocf::heartbeat:IPaddr2): Started pcmk-2

ClusterIP:1 (ocf::heartbeat:IPaddr2): Started pcmk-1

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

If desired, you can demonstrate that all request buckets are working by using a tool such as arping from several source hosts to see which host responds to each.

## Clone the Filesystem and Apache Resources

Now that we have a cluster filesystem ready to go, and our nodes can load-balance requests to a shared IP address, we can configure the cluster so both nodes mount the filesystem and respond to web requests.

Clone the filesystem and Apache resources in a new configuration. Notice how pcs automatically updates the relevant constraints again.

[root@pcmk-1 ~]# pcs cluster cib active\_cfg

[root@pcmk-1 ~]# pcs -f active\_cfg resource clone WebFS

[root@pcmk-1 ~]# pcs -f active\_cfg resource clone WebSite

[root@pcmk-1 ~]# pcs -f active\_cfg constraint

Location Constraints:

Ordering Constraints:

start ClusterIP-clone then start WebSite-clone (kind:Mandatory)

promote WebDataClone then start WebFS-clone (kind:Mandatory)

start WebFS-clone then start WebSite-clone (kind:Mandatory)

start dlm-clone then start WebFS-clone (kind:Mandatory)

Colocation Constraints:

WebSite-clone with ClusterIP-clone (score:INFINITY)

WebFS-clone with WebDataClone (score:INFINITY) (with-rsc-role:Master)

WebSite-clone with WebFS-clone (score:INFINITY)

WebFS-clone with dlm-clone (score:INFINITY)

Ticket Constraints:

Tell the cluster that it is now allowed to promote both instances to be DRBD Primary (aka. master).

[root@pcmk-1 ~]# pcs -f active\_cfg resource update WebDataClone master-max=2

Finally, load our configuration to the cluster, and re-enable the WebFS resource (which we disabled earlier).

[root@pcmk-1 ~]# pcs cluster cib-push active\_cfg --config

CIB updated

[root@pcmk-1 ~]# pcs resource enable WebFS

After all the processes are started, the status should look similar to this.

[root@pcmk-1 ~]# pcs resource

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 pcmk-2 ]

Clone Set: dlm-clone [dlm]

Started: [ pcmk-1 pcmk-2 ]

Clone Set: ClusterIP-clone [ClusterIP] (unique)

ClusterIP:0 (ocf::heartbeat:IPaddr2): Started pcmk-2

ClusterIP:1 (ocf::heartbeat:IPaddr2): Started pcmk-1

Clone Set: WebFS-clone [WebFS]

Started: [ pcmk-1 pcmk-2 ]

Clone Set: WebSite-clone [WebSite]

Started: [ pcmk-1 pcmk-2 ]

## Test Failover

Testing failover is left as an exercise for the reader. For example, you can put one node into standby mode, use **pcs status** to confirm that its ClusterIP clone was moved to the other node, and use **arping** to verify that packets are not being lost from any source host.

**Note:** You may find that when a failed node rejoins the cluster, both ClusterIP clones stay on one node, due to the resource stickiness. While this works fine, it effectively eliminates load-balancing and returns the cluster to an active-passive setup again. You can avoid this by disabling stickiness for the IP address resource:

[root@pcmk-1 ~]# pcs resource meta ClusterIP resource-stickiness=0

## Final Cluster Configuration

[root@pcmk-1 ~]# pcs resource

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 pcmk-2 ]

Clone Set: dlm-clone [dlm]

Started: [ pcmk-1 pcmk-2 ]

Clone Set: ClusterIP-clone [ClusterIP] (unique)

ClusterIP:0 (ocf::heartbeat:IPaddr2): Started pcmk-2

ClusterIP:1 (ocf::heartbeat:IPaddr2): Started pcmk-1

Clone Set: WebFS-clone [WebFS]

Started: [ pcmk-1 pcmk-2 ]

Clone Set: WebSite-clone [WebSite]

Started: [ pcmk-1 pcmk-2 ]

[root@pcmk-1 ~]# pcs resource op defaults

timeout: 240s

[root@pcmk-1 ~]# pcs stonith

impi-fencing (stonith:fence\_ipmilan): Started pcmk-1

[root@pcmk-1 ~]# pcs constraint

Location Constraints:

Ordering Constraints:

start ClusterIP-clone then start WebSite-clone (kind:Mandatory)

promote WebDataClone then start WebFS-clone (kind:Mandatory)

start WebFS-clone then start WebSite-clone (kind:Mandatory)

start dlm-clone then start WebFS-clone (kind:Mandatory)

Colocation Constraints:

WebSite-clone with ClusterIP-clone (score:INFINITY)

WebFS-clone with WebDataClone (score:INFINITY) (with-rsc-role:Master)

WebSite-clone with WebFS-clone (score:INFINITY)

WebFS-clone with dlm-clone (score:INFINITY)

Ticket Constraints:

[root@pcmk-1 ~]# pcs status

Cluster name: mycluster

Stack: corosync

Current DC: pcmk-1 (version 1.1.18-11.el7\_5.3-2b07d5c5a9) - partition with quorum

Last updated: Tue Sep 11 10:41:53 2018

Last change: Tue Sep 11 10:40:16 2018 by root via cibadmin on pcmk-1

2 nodes configured

11 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ipmi-fencing (stonith:fence\_ipmilan): Started pcmk-1

Master/Slave Set: WebDataClone [WebData]

Masters: [ pcmk-1 pcmk-2 ]

Clone Set: dlm-clone [dlm]

Started: [ pcmk-1 pcmk-2 ]

Clone Set: ClusterIP-clone [ClusterIP] (unique)

ClusterIP:0 (ocf::heartbeat:IPaddr2): Started pcmk-2

ClusterIP:1 (ocf::heartbeat:IPaddr2): Started pcmk-1

Clone Set: WebFS-clone [WebFS]

Started: [ pcmk-1 pcmk-2 ]

Clone Set: WebSite-clone [WebSite]

Started: [ pcmk-1 pcmk-2 ]

Daemon Status:

corosync: active/disabled

pacemaker: active/disabled

pcsd: active/enabled

[root@pcmk-1 ~]# pcs cluster cib --config

## Node List

[root@pcmk-1 ~]# pcs status nodes

Pacemaker Nodes:

Online: pcmk-1 pcmk-2

Standby:

Maintenance:

Offline:

Pacemaker Remote Nodes:

Online:

Standby:

Maintenance:

Offline:

### Fencing

[root@pcmk-1 ~]# pcs stonith show

ipmi-fencing (stonith:fence\_ipmilan): Started pcmk-1

[root@pcmk-1 ~]# pcs stonith show ipmi-fencing

Resource: ipmi-fencing (class=stonith type=fence\_ipmilan)

Attributes: ipaddr="10.0.0.1" login="testuser" passwd="acd123" pcmk\_host\_list="pcmk-1 pcmk-2"

Operations: monitor interval=60s (fence-monitor-interval-60s)

### DRBD - Shared Storage

[root@pcmk-1 ~]# pcs resource show WebDataClone

[root@pcmk-1 ~]# pcs constraint ref WebDataClone

### Cluster Filesystem

[root@pcmk-1 ~]# pcs resource show WebFS-clone

[root@pcmk-1 ~]# pcs constraint ref WebFS-clone

[root@pcmk-1 ~]# pcs resource show WebSite-clone

[root@pcmk-1 ~]# pcs constraint ref WebSite-clone

# Sample Corosync Configuration

totem {

version: 2

cluster\_name: mycluster

secauth: off

transport: udpu

}

nodelist {

node {

ring0\_addr: pcmk-1

nodeid: 1

}

node {

ring0\_addr: pcmk-2

nodeid: 2

}

}

quorum {

provider: corosync\_votequorum

two\_node: 1}

logging {

to\_logfile: yes

logfile: /var/log/cluster/corosync.log

to\_syslog: yes

}

Reference link: <https://clusterlabs.org/pacemaker/doc/deprecated/en-US/Pacemaker/1.1/html/Clusters_from_Scratch/index.html>