Clusters from Scratch

Step-by-Step Instructions for Building Your First High-Availability Cluster

, Written by the Pacemaker project contributors

Clusters from Scratch: Step-by-Step Instructions for Building Your First High-Availability Cluster

by

Abstract

This document provides a step-by-step guide to building a simple high-availability cluster using Pacemaker.

The example cluster will use:

1. CentOS 7.5 as the host operating system
2. Corosync to provide messaging and membership services,
3. Pacemaker 1.1.18 [1]
4. DRBD as a cost-effective alternative to shared storage,
5. GFS2 as the cluster filesystem (in active/active mode)

Given the graphical nature of the install process, a number of screenshots are included. However the guide is primarily composed of commands, the reasons for executing them and their expected outputs.

[1] While this guide is part of the document set for Pacemaker 2.0, it demonstrates the version available in the standard CentOS repositories.

Table of Contents

List of Figures

List of Examples

Preface

Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the **cat my_next_bestselling_novel** command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **Enter** to execute the command.
>
> Press **Ctrl**+**Alt**+**F2** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `mono-spaced bold`. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog-box text; labeled buttons; check-box and radio-button labels; menu titles and submenu titles. For example:

> Choose System → Preferences → Mouse from the main menu bar to launch Mouse Preferences. In the Buttons tab, select the Left-handed mouse check box and click Close to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a gedit file, choose Applications → Accessories → Character Map from the main menu bar. Next, choose Search → Find… from the Character Map menu bar, type the name of the character in the Search field and click Next. The character you sought will be highlighted in the Character Table. Double-click

this highlighted character to place it in the Text to copy field and then click the Copy button. Now switch back to your document and choose Edit → Paste from the gedit menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or *Proportional Bold Italic*

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username@domain.name*** at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the `/home` file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above: username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in `mono-spaced roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `mono-spaced roman` but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
```

```
        Echo           echo   = home.create();

        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
    }
}
```

Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled “Important” will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

We Need Feedback!

You should over ride this by creating your own local Feedback.xml file.

Chapter 1. Read-Me-First

The Scope of this Document

Computer clusters can be used to provide highly available services or resources. The redundancy of multiple machines is used to guard against failures of many types.

This document will walk through the installation and setup of simple clusters using the CentOS distribution, version &DISTRO_VERSION;.

The clusters described here will use Pacemaker and Corosync to provide resource management and messaging. Required packages and modifications to their configuration files are described along with the use of the Pacemaker command line tool for generating the XML used for cluster control.

Pacemaker is a central component and provides the resource management required in these systems. This management includes detecting and recovering from the failure of various nodes, resources and services under its control.

When more in-depth information is required, and for real-world usage, please refer to the Pacemaker Explained [https://www.clusterlabs.org/pacemaker/doc/] manual.

What Is *Pacemaker*?

Pacemaker is a high-availability *cluster resource manager* — software that runs on a set of hosts (a *cluster* of *nodes*) in order to preserve integrity and minimize downtime of desired services (*resources*). [1] It is maintained by the ClusterLabs [https://www.ClusterLabs.org/] community.

Pacemaker's key features include:

- Detection of and recovery from node- and service-level failures
- Ability to ensure data integrity by fencing faulty nodes
- Support for one or more nodes per cluster
- Support for multiple resource interface standards (anything that can be scripted can be clustered)
- Support (but no requirement) for shared storage
- Support for practically any redundancy configuration (active/passive, N+1, etc.)
- Automatically replicated configuration that can be updated from any node
- Ability to specify cluster-wide relationships between services, such as ordering, colocation and anti-colocation
- Support for advanced service types, such as *clones* (services that need to be active on multiple nodes), *stateful resources* (clones that can run in one of two modes), and containerized services
- Unified, scriptable cluster management tools

[1] *Cluster* is sometimes used in other contexts to refer to hosts grouped together for other purposes, such as high-performance computing (HPC), but Pacemaker is not intended for those purposes.

Fencing

Fencing, also known as *STONITH* (an acronym for Shoot The Other Node In The Head), is the ability to ensure that it is not possible for a node to be running a service. This is accomplished via *fence devices* such as intelligent power switches that cut power to the target, or intelligent network switches that cut the target's access to the local network.

Pacemaker represents fence devices as a special class of resource.

A cluster cannot safely recover from certain failure conditions, such as an unresponsive node, without fencing.

Cluster Architecture

At a high level, a cluster can be viewed as having these parts (which together are often referred to as the *cluster stack*):

- **Resources:** These are the reason for the cluster's being — the services that need to be kept highly available.
- **Resource agents:** These are scripts or operating system components that start, stop, and monitor resources, given a set of resource parameters. These provide a uniform interface between Pacemaker and the managed services.
- **Fence agents:** These are scripts that execute node fencing actions, given a target and fence device parameters.
- **Cluster membership layer:** This component provides reliable messaging, membership, and quorum information about the cluster. Currently, Pacemaker supports Corosync [http://www.corosync.org/] as this layer.
- **Cluster resource manager:** Pacemaker provides the brain that processes and reacts to events that occur in the cluster. These events may include nodes joining or leaving the cluster; resource events caused by failures, maintenance, or scheduled activities; and other administrative actions. To achieve the desired availability, Pacemaker may start and stop resources and fence nodes.
- **Cluster tools:** These provide an interface for users to interact with the cluster. Various command-line and graphical (GUI) interfaces are available.

Most managed services are not, themselves, cluster-aware. However, many popular open-source cluster filesystems make use of a common *Distributed Lock Manager* (DLM), which makes direct use of Corosync for its messaging and membership capabilities and Pacemaker for the ability to fence nodes.

Figure 1.1. Example Cluster Stack

Pacemaker Architecture

Pacemaker itself is composed of multiple daemons that work together:

- pacemakerd
- pacemaker-attrd

- pacemaker-based
- pacemaker-controld
- pacemaker-execd
- pacemaker-fenced
- pacemaker-schedulerd

Figure 1.2. Internal Components

The Pacemaker master process (pacemakerd) spawns all the other daemons, and respawns them if they unexpectedly exit.

The *Cluster Information Base* (CIB) is an XML [https://en.wikipedia.org/wiki/XML] representation of the cluster's configuration and the state of all nodes and resources. The *CIB manager* (pacemaker-based) keeps the CIB synchronized across the cluster, and handles requests to modify it.

The attribute manager (pacemaker-attrd) maintains a database of attributes for all nodes, keeps it synchronized across the cluster, and handles requests to modify them. These attributes are usually recorded in the CIB.

Given a snapshot of the CIB as input, the *scheduler* (pacemaker-schedulerd) determines what actions are necessary to achieve the desired state of the cluster.

The *local executor* (pacemaker-execd) handles requests to execute resource agents on the local cluster node, and returns the result.

The *fencer* (pacemaker-fenced) handles requests to fence nodes. Given a target node, the fencer decides which cluster node(s) should execute which fencing device(s), and calls the necessary fencing agents (either directly, or via requests to the fencer peers on other nodes), and returns the result.

The *controller* (pacemaker-controld) is Pacemaker's coordinator, maintaining a consistent view of the cluster membership and orchestrating all the other components.

Pacemaker centralizes cluster decision-making by electing one of the controller instances as the *Designated Controller* (*DC*). Should the elected DC process (or the node it is on) fail, a new one is quickly established. The DC responds to cluster events by taking a current snapshot of the CIB, feeding it to the scheduler, then asking the executors (either directly on the local node, or via requests to controller peers on other nodes) and the fencer to execute any necessary actions.

Old daemon names

The Pacemaker daemons were renamed in version 2.0. You may still find references to the old names, especially in documentation targeted to version 1.1.

Old name	New name
attrd	pacemaker-attrd
cib	pacemaker-based
crmd	pacemaker-controld
lrmd	pacemaker-execd

Old name	New name
stonithd	pacemaker-fenced
pacemaker_remoted	pacemaker-remoted

Node Redundancy Designs

Pacemaker supports practically any node redundancy configuration [https://en.wikipedia.org/wiki/High-availability_cluster#Node_configurations] including *Active/Active*, *Active/Passive*, *N+1*, *N+M*, *N-to-1* and *N-to-N*.

Active/passive clusters with two (or more) nodes using Pacemaker and DRBD [https://en.wikipedia.org/wiki/Distributed_Replicated_Block_Device:] are a cost-effective high-availability solution for many situations. One of the nodes provides the desired services, and if it fails, the other node takes over.

Figure 1.3. Active/Passive Redundancy

Pacemaker also supports multiple nodes in a shared-failover design, reducing hardware costs by allowing several active/passive clusters to be combined and share a common backup node.

Figure 1.4. Shared Failover

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.

Figure 1.5. N to N Redundancy

Chapter 2. Installation

Install CentOS &DISTRO_VERSION;

Boot the Install Image

Download the 4GB CentOS &DISTRO_VERSION; DVD ISO [http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-DVD-1804.iso]. Use the image to boot a virtual machine, or burn it to a DVD or USB drive and boot a physical server from that.

After starting the installation, select your language and keyboard layout at the welcome screen.

Figure 2.1. CentOS &DISTRO_VERSION; Installation Welcome Screen

Installation Options

At this point, you get a chance to tweak the default installation options.

Figure 2.2. CentOS &DISTRO_VERSION; Installation Summary Screen

Ignore the **SOFTWARE SELECTION** section (try saying that 10 times quickly). The **Infrastructure Server** environment does have add-ons with much of the software we need, but we will leave it as a **Minimal Install** here, so that we can see exactly what software is required later.

Configure Network

In the **NETWORK & HOSTNAME** section:

- Edit **Host Name:** as desired. For this example, we will use **pcmk-1.localdomain**.
- Select your network device, press **Configure…**, and manually assign a fixed IP address. For this example, we'll use 192.168.122.101 under **IPv4 Settings** (with an appropriate netmask, gateway and DNS server).
- Flip the switch to turn your network device on, and press **Done**.

Figure 2.3. CentOS &DISTRO_VERSION; Network Interface Screen

Important

Do not accept the default network settings. Cluster machines should never obtain an IP address via DHCP, because DHCP's periodic address renewal will interfere with corosync.

Configure Disk

By default, the installer's automatic partitioning will use LVM (which allows us to dynamically change the amount of space allocated to a given partition). However, it allocates all free space to the `/` (aka. **root**) partition, which cannot be reduced in size later (dynamic increases are fine).

In order to follow the DRBD and GFS2 portions of this guide, we need to reserve space on each machine for a replicated volume.

Enter the **INSTALLATION DESTINATION** section, ensure the hard drive you want to install to is selected, select **I will configure partitioning**, and press **Done**.

In the **MANUAL PARTITIONING** screen that comes next, click the option to create mountpoints automatically. Select the / mountpoint, and reduce the desired capacity by 1GiB or so. Select **Modify…** by the volume group name, and change the **Size policy:** to **As large as possible**, to make the reclaimed space available inside the LVM volume group. We'll add the additional volume later.

Figure 2.4. CentOS &DISTRO_VERSION; Manual Partitioning Screen

Press **Done**, then **Accept changes**.

Configure Time Synchronization

It is highly recommended to enable NTP on your cluster nodes. Doing so ensures all nodes agree on the current time and makes reading log files significantly easier.

CentOS will enable NTP automatically. If you want to change any time-related settings (such as time zone or NTP server), you can do this in the **TIME & DATE** section.

Finish Install

Select **Begin Installation**. Once it completes, set a root password, and reboot as instructed. For the purposes of this document, it is not necessary to create any additional users. After the node reboots, you'll see a login prompt on the console. Login using **root** and the password you created earlier.

Figure 2.5. CentOS &DISTRO_VERSION; Console Prompt

Note

From here on, we're going to be working exclusively from the terminal.

Configure the OS

Verify Networking

Ensure that the machine has the static IP address you configured earlier.

```
[root@pcmk-1 ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP grou
```

```
    link/ether 52:54:00:8e:eb:41 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.101/24 brd 192.168.122.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::e45:c99b:34c0:c657/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Note

If you ever need to change the node's IP address from the command line, follow these instructions, replacing **${device}** with the name of your network device:

```
[root@pcmk-1 ~]# vi /etc/sysconfig/network-scripts/ifcfg-${device} # manually e
[root@pcmk-1 ~]# nmcli dev disconnect ${device}
[root@pcmk-1 ~]# nmcli con reload ${device}
[root@pcmk-1 ~]# nmcli con up ${device}
```

This makes **NetworkManager** aware that a change was made on the config file.

Next, ensure that the routes are as expected:

```
[root@pcmk-1 ~]# ip route
default via 192.168.122.1 dev eth0 proto static metric 100
192.168.122.0/24 dev eth0 proto kernel scope link src 192.168.122.101 metric 100
```

If there is no line beginning with **default via**, then you may need to add a line such as

```
GATEWAY="192.168.122.1"
```

to the device configuration using the same process as described above for changing the IP address.

Now, check for connectivity to the outside world. Start small by testing whether we can reach the gateway we configured.

```
[root@pcmk-1 ~]# ping -c 1 192.168.122.1
PING 192.168.122.1 (192.168.122.1) 56(84) bytes of data.
64 bytes from 192.168.122.1: icmp_seq=1 ttl=64 time=0.254 ms

--- 192.168.122.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.254/0.254/0.254/0.000 ms
```

Now try something external; choose a location you know should be available.

```
[root@pcmk-1 ~]# ping -c 1 www.clusterlabs.org
PING oss-uk-1.clusterlabs.org (109.74.197.241) 56(84) bytes of data.
64 bytes from oss-uk-1.clusterlabs.org (109.74.197.241): icmp_seq=1 ttl=49 time=33

--- oss-uk-1.clusterlabs.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 333.204/333.204/333.204/0.000 ms
```

Login Remotely

The console isn't a very friendly place to work from, so we will now switch to accessing the machine remotely via SSH where we can use copy and paste, etc.

From another host, check whether we can see the new host at all:

```
beekhof@f16 ~ # ping -c 1 192.168.122.101
PING 192.168.122.101 (192.168.122.101) 56(84) bytes of data.
64 bytes from 192.168.122.101: icmp_req=1 ttl=64 time=1.01 ms

--- 192.168.122.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.012/1.012/1.012/0.000 ms
```

Next, login as root via SSH.

```
beekhof@f16 ~ # ssh -l root 192.168.122.101
The authenticity of host '192.168.122.101 (192.168.122.101)' can't be established.
ECDSA key fingerprint is 6e:b7:8f:e2:4c:94:43:54:a8:53:cc:20:0f:29:a4:e0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.101' (ECDSA) to the list of known hosts.
root@192.168.122.101's password:
Last login: Tue Aug 11 13:14:39 2015
[root@pcmk-1 ~]#
```

Apply Updates

Apply any package updates released since your installation image was created:

```
[root@pcmk-1 ~]# yum update
```

Use Short Node Names

During installation, we filled in the machine's fully qualified domain name (FQDN), which can be rather long when it appears in cluster logs and status output. See for yourself how the machine identifies itself:

```
[root@pcmk-1 ~]# uname -n
pcmk-1.localdomain
```

We can use the `hostnamectl` tool to strip off the domain name:

```
[root@pcmk-1 ~]# hostnamectl set-hostname $(uname -n | sed s/\\..*//)
```

Now, check that the machine is using the correct name:

```
[root@pcmk-1 ~]# uname -n
pcmk-1
```

You may want to reboot to ensure all updates take effect.

Repeat for Second Node

Repeat the Installation steps so far, so that you have two nodes ready to have the cluster software installed.

For the purposes of this document, the additional node is called pcmk-2 with address 192.168.122.102.

Configure Communication Between Nodes

Configure Host Name Resolution

Confirm that you can communicate between the two new nodes:

```
[root@pcmk-1 ~]# ping -c 3 192.168.122.102
PING 192.168.122.102 (192.168.122.102) 56(84) bytes of data.
64 bytes from 192.168.122.102: icmp_seq=1 ttl=64 time=0.343 ms
64 bytes from 192.168.122.102: icmp_seq=2 ttl=64 time=0.402 ms
64 bytes from 192.168.122.102: icmp_seq=3 ttl=64 time=0.558 ms

--- 192.168.122.102 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.343/0.434/0.558/0.092 ms
```

Now we need to make sure we can communicate with the machines by their name. If you have a DNS server, add additional entries for the two machines. Otherwise, you'll need to add the machines to `/etc/hosts` on both nodes. Below are the entries for my cluster nodes:

```
[root@pcmk-1 ~]# grep pcmk /etc/hosts
192.168.122.101 pcmk-1.clusterlabs.org pcmk-1
192.168.122.102 pcmk-2.clusterlabs.org pcmk-2
```

We can now verify the setup by again using ping:

```
[root@pcmk-1 ~]# ping -c 3 pcmk-2
PING pcmk-2.clusterlabs.org (192.168.122.101) 56(84) bytes of data.
64 bytes from pcmk-1.clusterlabs.org (192.168.122.101): icmp_seq=1 ttl=64 time=0.1
64 bytes from pcmk-1.clusterlabs.org (192.168.122.101): icmp_seq=2 ttl=64 time=0.4
64 bytes from pcmk-1.clusterlabs.org (192.168.122.101): icmp_seq=3 ttl=64 time=0.1

--- pcmk-2.clusterlabs.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.164/0.275/0.475/0.141 ms
```

Configure SSH

SSH is a convenient and secure way to copy files and perform commands remotely. For the purposes of this guide, we will create a key without a password (using the -N option) so that we can perform remote actions without being prompted.

Warning

Unprotected SSH keys (those without a password) are not recommended for servers exposed to the outside world. We use them here only to simplify the demo.

Create a new key and allow anyone with that key to log in:

Creating and Activating a new SSH Key.

```
[root@pcmk-1 ~]# ssh-keygen -t dsa -f ~/.ssh/id_dsa -N ""
```

```
Generating public/private dsa key pair.
Your identification has been saved in /root/.ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
91:09:5c:82:5a:6a:50:08:4e:b2:0c:62:de:cc:74:44 root@pcmk-1.clusterlabs.org
The key's randomart image is:
+--[ DSA 1024]----+
|==.ooEo..        |
|X O + .o o       |
| * A    +        |
|  +      .       |
| .      S        |
|                 |
|                 |
|                 |
|                 |
+-----------------+
[root@pcmk-1 ~]# cp ~/.ssh/id_dsa.pub ~/.ssh/authorized_keys
```

Install the key on the other node:

```
[root@pcmk-1 ~]# scp -r ~/.ssh pcmk-2:
The authenticity of host 'pcmk-2 (192.168.122.102)' can't be established.
ECDSA key fingerprint is SHA256:63xNPkPYq98rYznf3T9QYJAzlaGiAsSgFVNHOZjPWqc.
ECDSA key fingerprint is MD5:d9:bf:6e:32:88:be:47:3d:96:f1:96:27:65:05:0b:c3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pcmk-2,192.168.122.102' (ECDSA) to the list of known h
root@pcmk-2's password:
id_dsa
id_dsa.pub
authorized_keys
known_hosts
```

Test that you can now run commands remotely, without being prompted:

```
[root@pcmk-1 ~]# ssh pcmk-2 -- uname -n
pcmk-2
```

Chapter 3. Set up a Cluster

Simplify Administration With a Cluster Shell

In the dark past, configuring Pacemaker required the administrator to read and write XML. In true UNIX style, there were also a number of different commands that specialized in different aspects of querying and updating the cluster.

In addition, the various components of the cluster stack (corosync, pacemaker, etc.) had to be configured separately, with different configuration tools and formats.

All of that has been greatly simplified with the creation of higher-level tools, whether command-line or GUIs, that hide all the mess underneath.

Command-line cluster shells take all the individual aspects required for managing and configuring a cluster, and pack them into one simple-to-use command-line tool.

They even allow you to queue up several changes at once and commit them all at once.

Two popular command-line shells are `pcs` and `crmsh`. Clusters from Scratch is based on `pcs` because it comes with CentOS, but both have similar functionality. Choosing a shell or GUI is a matter of personal preference and what comes with (and perhaps is supported by) your choice of operating system.

Install the Cluster Software

Fire up a shell on both nodes and run the following to install pacemaker, pcs, and some other command-line tools that will make our lives easier:

```
# yum install -y pacemaker pcs psmisc policycoreutils-python
```

Important

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.

Note

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

Configure the Cluster Software

Allow cluster services through firewall

On each node, allow cluster-related services through the local firewall:

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

Note

If you are using iptables directly, or some other firewall solution besides firewalld, simply open the following ports, which can be used by various clustering components: TCP ports 2224, 3121, and 21064, and UDP port 5405.

If you run into any problems during testing, you might want to disable the firewall and SELinux entirely until you have everything working. This may create significant security issues and should not be performed on machines that will be exposed to the outside world, but may be appropriate during development and testing on a protected host.

To disable security measures:

```
[root@pcmk-1 ~]# setenforce 0
[root@pcmk-1 ~]# sed -i.bak "s/SELINUX=enforcing/SELINUX=permissive/g" /etc/sel
[root@pcmk-1 ~]# systemctl mask firewalld.service
[root@pcmk-1 ~]# systemctl stop firewalld.service
[root@pcmk-1 ~]# iptables --flush
```

Enable pcs Daemon

Before the cluster can be configured, the pcs daemon must be started and enabled to start at boot time on each node. This daemon works with the pcs command-line interface to manage synchronizing the corosync configuration across all nodes in the cluster.

Start and enable the daemon by issuing the following commands on each node:

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/pcsd.service to /
```

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.

This tutorial will make use of such commands, 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.
```

Note

Alternatively, to script this process or set the password on a different machine from the one you're logged into, you can use the `--stdin` option for `passwd`:

```
[root@pcmk-1 ~]# ssh pcmk-2 -- 'echo mysupersecretpassword | passwd --stdin hac
```

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
```

Note

In Fedora 29 and CentOS 8.0, the command has been changed to `pcs host auth`:

```
[root@pcmk-1 ~]# pcs host 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
```

Note

In Fedora 29 and CentOS 8.0, the syntax has been changed and the `--name` option has been dropped:

```
[root@pcmk-1 ~]# pcs cluster setup mycluster pcmk-1 pcmk-2
No addresses specified for host 'pcmk-1', using 'pcmk-1'
No addresses specified for host 'pcmk-2', using 'pcmk-2'
Destroying cluster on hosts: 'pcmk-1', 'pcmk-2'...
pcmk-1: Successfully destroyed cluster
pcmk-2: Successfully destroyed cluster
Requesting remove 'pcsd settings' from 'pcmk-1', 'pcmk-2'
pcmk-1: successful removal of the file 'pcsd settings'
pcmk-2: successful removal of the file 'pcsd settings'
Sending 'corosync authkey', 'pacemaker authkey' to 'pcmk-1', 'pcmk-2'
```

```
pcmk-2: successful distribution of the file 'corosync authkey'
pcmk-2: successful distribution of the file 'pacemaker authkey'
pcmk-1: successful distribution of the file 'corosync authkey'
pcmk-1: successful distribution of the file 'pacemaker authkey'
Synchronizing pcsd SSL certificates on nodes 'pcmk-1', 'pcmk-2'...
pcmk-1: Success
pcmk-2: Success
Sending 'corosync.conf' to 'pcmk-1', 'pcmk-2'
pcmk-2: successful distribution of the file 'corosync.conf'
pcmk-1: successful distribution of the file 'corosync.conf'
Cluster has been successfully set up.
```

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

If you are not using `pcs` for cluster administration, follow whatever procedures are appropriate for your tools to create a corosync.conf and copy it to all nodes.

The `pcs` command will configure corosync to use UDP unicast transport; if you choose to use multicast instead, choose a multicast address carefully. [1]

The final corosync.conf configuration on each node should look something like the sample in Appendix B, *Sample Corosync Configuration*.

Explore pcs

Start by taking some time to familiarize yourself with what `pcs` can do.

```
[root@pcmk-1 ~]# pcs

Usage: pcs [-f file] [-h] [commands]...
Control and configure pacemaker and corosync.

Options:
    -h, --help         Display usage and exit.
    -f file            Perform actions on file instead of active CIB.
    --debug            Print all network traffic and external commands run.
    --version          Print pcs version information. List pcs capabilities if
                       --full is specified.
    --request-timeout  Timeout for each outgoing request to another node in
                       seconds. Default is 60s.
    --force            Override checks and errors, the exact behavior depends on
                       the command. WARNING: Using the --force option is
                       strongly discouraged unless you know what you are doing.

Commands:
    cluster     Configure cluster options and nodes.
    resource    Manage cluster resources.
```

[1] For some subtle issues, see Topics in High-Performance Messaging: Multicast Address Assignment [http://web.archive.org/web/20101211210054/http://29west.com/docs/THPM/multicast-address-assignment.html] or the more detailed treatment in Cisco's Guidelines for Enterprise IP Multicast Address Allocation [https://www.cisco.com/c/dam/en/us/support/docs/ip/ip-multicast/ipmlt_wp.pdf].

```
    stonith     Manage fence devices.
    constraint  Manage resource constraints.
    property    Manage pacemaker properties.
    acl         Manage pacemaker access control lists.
    qdevice     Manage quorum device provider on the local host.
    quorum      Manage cluster quorum settings.
    booth       Manage booth (cluster ticket manager).
    status      View cluster status.
    config      View and manage cluster configuration.
    pcsd        Manage pcs daemon.
    node        Manage cluster nodes.
    alert       Manage pacemaker alerts.
```

As you can see, the different aspects of cluster management are separated into categories. To discover the functionality available in each of these categories, one can issue the command `pcs *category* help`. Below is an example of all the options available under the status category.

```
[root@pcmk-1 ~]# pcs status help

Usage: pcs status [commands]...
View current cluster and resource status
Commands:
    [status] [--full | --hide-inactive]
        View all information about the cluster and resources (--full provides
        more details, --hide-inactive hides inactive resources).

    resources [<resource id> | --full | --groups | --hide-inactive]
        Show all currently configured resources or if a resource is specified
        show the options for the configured resource.  If --full is specified,
        all configured resource options will be displayed.  If --groups is
        specified, only show groups (and their resources).  If --hide-inactive
        is specified, only show active resources.

    groups
        View currently configured groups and their resources.

    cluster
        View current cluster status.

    corosync
        View current membership information as seen by corosync.

    quorum
        View current quorum status.

    qdevice <device model> [--full] [<cluster name>]
        Show runtime status of specified model of quorum device provider.  Using
        --full will give more detailed output.  If <cluster name> is specified,
        only information about the specified cluster will be displayed.

    nodes [corosync | both | config]
        View current status of nodes from pacemaker. If 'corosync' is
        specified, view current status of nodes from corosync instead. If
        'both' is specified, view current status of nodes from both corosync &
```

```
        pacemaker. If 'config' is specified, print nodes from corosync &
        pacemaker configuration.

    pcsd [<node>]...
        Show current status of pcsd on nodes specified, or on all nodes
        configured in the local cluster if no nodes are specified.

    xml
        View xml version of status (output from crm_mon -r -1 -X).
```

Additionally, if you are interested in the version and supported cluster stack(s) available with your Pacemaker installation, run:

```
[root@pcmk-1 ~]# pacemakerd --features
Pacemaker 1.1.18-11.el7_5.3 (Build: 2b07d5c5a9)
 Supporting v3.0.14:  generated-manpages agent-manpages ncurses libqb-logging libq
```

Chapter 4. Start and Verify Cluster

Start the Cluster

Now that corosync is configured, it is time to start the cluster. The command below will start corosync and pacemaker on both nodes in the cluster. If you are issuing the start command from a different node than the one you ran the `pcs cluster auth` command on earlier, you must authenticate on the current node you are logged into before you will be allowed to start the cluster.

```
[root@pcmk-1 ~]# pcs cluster start --all
pcmk-1: Starting Cluster...
pcmk-2: Starting Cluster...
```

Note

An alternative to using the `pcs cluster start --all` command is to issue either of the below command sequences on each node in the cluster separately:

```
# pcs cluster start
Starting Cluster...
```

or

```
# systemctl start corosync.service
# systemctl start pacemaker.service
```

Important

In this example, we are not enabling the corosync and pacemaker services to start at boot. If a cluster node fails or is rebooted, you will need to run `pcs cluster start` *`nodename`* (or `--all`) to start the cluster on it. While you could enable the services to start at boot, requiring a manual start of cluster services gives you the opportunity to do a post-mortem investigation of a node failure before returning it to the cluster.

Verify Corosync Installation

First, use `corosync-cfgtool` to check whether cluster communication is happy:

```
[root@pcmk-1 ~]# corosync-cfgtool -s
Printing ring status.
Local node ID 1
RING ID 0
 id = 192.168.122.101
 status = ring 0 active with no faults
```

We can see here that everything appears normal with our fixed IP address (not a 127.0.0.x loopback address) listed as the **id**, and **no faults** for the status.

If you see something different, you might want to start by checking the node's network, firewall and SELinux configurations.

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

Now that we have confirmed that Corosync is functional, we can check the rest of the stack. Pacemaker has already been started, so verify the necessary processes are running:

```
[root@pcmk-1 ~]# ps axf
  PID TTY      STAT   TIME COMMAND
    2 ?        S      0:00 [kthreadd]
...lots of processes...
11635 ?        SLsl   0:03 corosync
11642 ?        Ss     0:00 /usr/sbin/pacemakerd -f
11643 ?        Ss     0:00  \_ /usr/libexec/pacemaker/cib
11644 ?        Ss     0:00  \_ /usr/libexec/pacemaker/stonithd
11645 ?        Ss     0:00  \_ /usr/libexec/pacemaker/lrmd
11646 ?        Ss     0:00  \_ /usr/libexec/pacemaker/attrd
11647 ?        Ss     0:00  \_ /usr/libexec/pacemaker/pengine
11648 ?        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
```

Note

Other operating systems may report startup errors in other locations, for example `/var/log/messages`.

Repeat these checks on the other node. The results should be the same.

Explore the Existing Configuration

For those who are not of afraid of XML, you can see the raw cluster configuration and status by using the `pcs cluster cib` command.

Example 4.1. The last XML you'll see in this document

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

<cib crm_feature_set="3.0.14" validate-with="pacemaker-2.10" epoch="5" num_updates
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair id="cib-bootstrap-options-have-watchdog" name="have-watchdog" valu
        <nvpair id="cib-bootstrap-options-dc-version" name="dc-version" value="1.1
        <nvpair id="cib-bootstrap-options-cluster-infrastructure" name="cluster-in
        <nvpair id="cib-bootstrap-options-cluster-name" name="cluster-name" value=
      </cluster_property_set>
    </crm_config>
    <nodes>
      <node id="1" uname="pcmk-1"/>
      <node id="2" uname="pcmk-2"/>
    </nodes>
    <resources/>
    <constraints/>
  </configuration>
  <status>
    <node_state id="1" uname="pcmk-1" in_ccm="true" crmd="online" crm-debug-origin
      <lrm id="1">
        <lrm_resources/>
      </lrm>
    </node_state>
    <node_state id="2" uname="pcmk-2" in_ccm="true" crmd="online" crm-debug-origin
      <lrm id="2">
        <lrm_resources/>
```

```
      </lrm>
    </node_state>
  </status>
</cib>
```

Before we make any changes, it's a good idea to check the validity of the configuration.

```
[root@pcmk-1 ~]# crm_verify -L -V
   error: unpack_resources: Resource start-up disabled since no STONITH resources
   error: unpack_resources: Either configure some or disable STONITH with the ston
   error: unpack_resources: NOTE: Clusters with shared data need STONITH to ensure
Errors found during check: config not valid
```

As you can see, the tool has found some errors. The cluster will not start any resources until we configure STONITH.

Chapter 5. Configure Fencing

What is Fencing?

Fencing protects your data from being corrupted, and your application from becoming unavailable, due to unintended concurrent access by rogue nodes.

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 fencing to ensure that the node is truly offline before allowing the data to be accessed from another node.

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

Fencing is also known as STONITH, an acronym for "Shoot The Other Node In The Head", since the most popular form of fencing is cutting a host's power.

In order to guarantee the safety of your data, [1] fencing is enabled by default.

Note

It is possible to tell the cluster not to use fencing, by setting the **stonith-enabled** cluster option to false:

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

However, this is completely inappropriate for a production cluster. It tells the cluster to simply pretend that failed nodes are safely powered off. Some vendors will refuse to support clusters that have fencing disabled. Even disabling it for a test cluster means you won't be able to test real failure scenarios.

Choose a Fence Device

The two broad categories of fence device are power fencing, which cuts off power to the target, and fabric fencing, which cuts off the target's access to some critical resource, such as a shared disk or access to the local network.

Power fencing devices include:

- Intelligent power switches
- IPMI
- Hardware watchdog device (alone, or in combination with shared storage used as a "poison pill" mechanism)

Fabric fencing devices include:

- Shared storage that can be cut off for a target host by another host (for example, an external storage device that supports SCSI-3 persistent reservations)

[1] If the data is corrupt, there is little point in continuing to make it available

- Intelligent network switches

Using IPMI as a power fencing device may seem like a good choice. However, if the IPMI shares power and/or network access with the host (such as most onboard IPMI controllers), a power or network failure will cause both the host and its fencing device to fail. The cluster will be unable to recover, and must 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, because fencing will be required when the node is completely unresponsive.

Configure the Cluster for Fencing

1. Install the fence agent(s). To see what packages are available, run `yum search fence-`. Be sure to install the package(s) on all cluster nodes.
2. Configure the fence 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.
3. Find the name of the correct fence agent: `pcs stonith list`
4. Find the parameters associated with the device: `pcs stonith describe` *`agent_name`*
5. Create a local copy of the CIB: `pcs cluster cib stonith_cfg`
6. 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`.
7. Enable fencing in the cluster: `pcs -f stonith_cfg property set stonith-enabled=true`
8. If the device does not know how to fence nodes based on their cluster node name, you may also need to set the special **pcmk_host_map** parameter. See `man pacemaker-fenced` for details.
9. 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 pacemaker-fenced` for details.
10. 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 pacemaker-fenced` for details.
11. Commit the new configuration: `pcs cluster cib-push stonith_cfg`
12. Once the fence device 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 a separately powered 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

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 hi
  pcmk_host_map: A mapping of host names to ports numbers for devices that do not
                 3 for node2
  pcmk_host_list: A list of machines controlled by this device (Optional unless pc
  pcmk_host_check: How to determine which machines are controlled by the device. A
                   (assume every device can fence every machine)
  pcmk_delay_max: Enable a random delay for stonith actions and specify the maximu
                  random delay for stonith actions. The overall delay is derived f
  pcmk_delay_base: Enable a base delay for stonith actions and specify base delay
                   a static delay for stonith actions. The overall delay is derive
  pcmk_action_limit: The maximum number of actions can be performed in parallel on
                     specify the maximum number of actions can be performed in par

Default operations:
  monitor: interval=60s
```

Step 5: `pcs cluster cib stonith_cfg`

Step 6: Here are example parameters for creating our fence device 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 fencing 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`).

Chapter 6. Create an Active/Passive Cluster

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.

Warning

The chosen address must not already be in use on the network. Do not reuse an IP address one of the nodes already has configured.

```
[root@pcmk-1 ~]# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 \
    ip=192.168.122.120 cidr_netmask=24 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)...
```

Note

A cluster command such as `pcs cluster stop` *`nodename`* can be run from any node in the cluster, not just the affected node.

Verify that pacemaker and corosync are no longer running:

```
[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.

Quorum

If a cluster splits into two (or more) groups of nodes that can no longer communicate with each other (aka. *partitions*), *quorum* is used to prevent resources from starting on more nodes than desired, which would risk data corruption.

A cluster has quorum when more than half of all known nodes are online in the same partition, or for the mathematically inclined, whenever the following equation is true:

```
total_nodes < 2 * active_nodes
```

For example, if a 5-node cluster split into 3- and 2-node paritions, the 3-node partition would have quorum and could continue serving resources. If a 6-node cluster split into two 3-node partitions, neither partition would have quorum; pacemaker's default behavior in such cases is to stop all resources, in order to prevent data corruption.

Two-node clusters are a special case. By the above definition, a two-node cluster would only have quorum when both nodes are running. This would make the creation of a two-node cluster pointless, but corosync has the ability to treat two-node clusters as if only one node is required for quorum.

The `pcs cluster setup` command will automatically configure **two_node: 1** in `corosync.conf`, so a two-node cluster will "just work".

If you are using a different cluster shell, you will have to configure `corosync.conf` appropriately yourself.

Now, simulate node recovery by restarting the cluster stack on **pcmk-1**, and check the cluster's status. (It may take a little while before the cluster gets going on the node, but it eventually will look like the below.)

```
[root@pcmk-1 ~]# pcs cluster start pcmk-1
pcmk-1: Starting Cluster...
[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 17:00:04 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-2

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

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" [1] 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 def
[root@pcmk-1 ~]# pcs resource defaults
resource-stickiness: 100
```

[1] Pacemaker's definition of optimal may not always agree with that of a human's. The order in which Pacemaker processes lists of resources and nodes creates implicit preferences in situations where the administrator has not explicitly specified them.

Chapter 7. Add Apache HTTP Server as a Cluster Service

Now that we have a basic but functional active/passive two-node cluster, we're ready to add some real services. We're going to start with Apache HTTP Server because it is a feature of many clusters and relatively simple to configure.

Install Apache

Before continuing, we need to make sure Apache is installed on both hosts. We also need the wget tool in order for the cluster to be able to check the status of the Apache server.

```
# yum install -y httpd wget
# firewall-cmd --permanent --add-service=http
# firewall-cmd --reload
```

Important

Do **not** enable the httpd service. Services that are intended to be managed via the cluster software should never be managed by the OS. It is often useful, however, to manually start the service, verify that it works, then stop it again, before adding it to the cluster. This allows you to resolve any non-cluster-related problems before continuing. Since this is a simple example, we'll skip that step here.

Create Website Documents

We need to create a page for Apache to serve. On CentOS &DISTRO_VERSION;, the default Apache document root is /var/www/html, so we'll create an index file there. For the moment, we will simplify things by serving a static site and manually synchronizing the data between the two nodes, so run this command on both nodes:

```
# cat <<-END >/var/www/html/index.html
 <html>
 <body>My Test Site - $(hostname)</body>
 </html>
END
```

Enable the Apache status URL

In order to monitor the health of your Apache instance, and recover it if it fails, the resource agent used by Pacemaker assumes the server-status URL is available. On both nodes, enable the URL with:

```
# cat <<-END >/etc/httpd/conf.d/status.conf
 <Location /server-status>
    SetHandler server-status
    Require local
```

```
 </Location>
END
```

Note

If you are using a different operating system, server-status may already be enabled or may be configurable in a different location. If you are using a version of Apache HTTP Server less than 2.4, the syntax will be different.

Configure the Cluster

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. We need to use an OCF resource script called apache in the heartbeat namespace. [1] The script's only required parameter is the path to the main Apache configuration file, and we'll tell the cluster to check once a minute that Apache is still running.

```
[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
```

By default, the operation timeout for all resources' start, stop, and monitor operations is 20 seconds. In many cases, this timeout period is less than a particular resource's advised timeout period. For the purposes of this tutorial, we will adjust the global operation timeout default to 240 seconds.

```
[root@pcmk-1 ~]# pcs resource op defaults timeout=240s
Warning: Defaults do not apply to resources which override them with their own def
[root@pcmk-1 ~]# pcs resource op defaults
timeout: 240s
```

Note

In a production cluster, it is usually better to adjust each resource's start, stop, and monitor timeouts to values that are appropriate to the behavior observed in your environment, rather than adjust the global default.

After a short delay, we should see the cluster start Apache.

```
[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 ]
```

[1] Compare the key used here, **ocf:heartbeat:apache**, with the one we used earlier for the IP address, **ocf:heartbeat:IPaddr2**

```
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
```

Wait a moment, the WebSite resource isn't running on the same host as our IP address!

Note

If, in the `pcs status` output, you see the WebSite resource has failed to start, then you've likely not enabled the status URL correctly. You can check whether this is the problem by running:

```
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.

Important

Colocation constraints are "directional", in that they imply certain things about the order in which the two resources will have a location chosen. In this case, we're saying that **WebSite** needs to be placed on the same machine as **ClusterIP**, which implies that the cluster must know the location of **ClusterIP** before choosing a location for **WebSite**.

```
[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. If Apache binds to the wildcard, it doesn't matter whether an IP address is added before or after Apache starts; Apache will respond on that IP just the same. However, if Apache binds only to certain IP address(es), the order matters: If the address is added after Apache starts, Apache won't respond on that address.

To be sure our WebSite responds regardless of Apache's address configuration, we need to make sure ClusterIP not only runs on the same node, but starts before WebSite. A colocation constraint only ensures the resources run together, not the order in which they are started and stopped.

We do this by adding an ordering constraint. By default, all order constraints are mandatory, which means that the recovery of ClusterIP 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-actio
[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.

We will use the **pcs resource move** command to create a temporary constraint with a score of INFINITY. While we could update our existing constraint, using **move** allows to easily get rid of the temporary constraint later. If desired, we could even give a lifetime for the constraint, so it would expire automatically — but we don't that in this example.

```
[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.

Chapter 8. Replicate Storage Using DRBD

Even if you're serving up static websites, having to manually synchronize the contents of that website to all the machines in the cluster is not ideal. For dynamic websites, such as a wiki, it's not even an option. Not everyone care afford network-attached storage, but somehow the data needs to be kept in sync.

Enter DRBD, which can be thought of as network-based RAID-1. [1]

Install the DRBD Packages

DRBD itself is included in the upstream kernel,[2] but we do need some utilities to use it effectively.

CentOS does not ship these utilities, so we need to enable a third-party repository to get them. Supported packages for many OSes are available from DRBD's maker LINBIT [http://www.linbit.com/], but here we'll use the free ELRepo [http://elrepo.org/] repository.

On both nodes, import the ELRepo package signing key, and enable the repository:

```
# 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
```

[1] See http://www.drbd.org/ for details.
[2] Since version 2.6.33

Note

In this example, we have only two nodes, and all network traffic is on the same LAN. In production, it is recommended to use a dedicated, isolated network for cluster-related traffic, so the firewall configuration would likely be different; one approach would be to add the dedicated network interfaces to the trusted zone.

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 Cp
  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.
```

Configure DRBD

There is no series of commands for building a DRBD configuration, so simply run this on both nodes to use this sample configuration:

```
# 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
```

Important

Edit the file to use the hostnames, IP addresses and logical volume paths of your nodes if they differ from the ones used in this guide.

Note

Detailed information on the directives used in this configuration (and other alternatives) is available in the DRBD User's Guide [https://docs.linbit.com/docs/users-guide-8.4/#ch-configure]. The **allow-two-primaries** option would not normally be used in an active/passive cluster. We are adding it here for the convenience of changing to an active/active cluster later.

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

 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

 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
```

Note

If you are using a different version of DRBD, the required syntax may be different. See the documentation for your version for how to perform these commands.

If we check the status immediately, we'll see something like this:

```
[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

 1: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-----
    ns:43184 nr:0 dw:0 dr:45312 al:8 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:481052
 [>...................] sync'ed:  8.6% (481052/524236)K
 finish: 0:01:51 speed: 4,316 (4,316) K/sec
```

We can see that this node has the **Primary** role, the partner node has the **Secondary** role, this node's data is now considered **UpToDate**, the partner node's data is still **Inconsistent**, and a progress bar shows how far along the partner node is in synchronizing the data.

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

 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
```

Note

In this example, we create an xfs filesystem with no special options. In a production environment, you should choose a filesystem type and options that are suitable for your application.

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 ]
```

Note

In Fedora 29 and CentOS 8.0, *master* resources have been renamed to *promotable clone* resources and the `pcs` command has been changed accordingly:

```
[root@pcmk-1 ~]# pcs -f drbd_cfg resource promotable WebData \
        promoted-max=1 promoted-node-max=1 clone-max=2 clone-node-max=1 \
        notify=true
```

The new command does not allow to set a custom name for the resulting promotable resource. `Pcs` automatically creates a name for the resource in the form of ***`resource_name`*-clone**, that is **WebData-clone** in this case.

To avoid confusion whether the `pcs resource show` command displays resources' status or configuration, the command has been deprecated in Fedora 29 and CentOS 8.0. Two new commands have been introduced for displaying resources' status and configuration: `pcs resource status` and `pcs resource config`, respectively.

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**.

Important

The resource agent should load the DRBD module when needed if it's not already loaded. If that does not happen, configure your operating system to load the module at boot time. For CentOS &DISTRO_VERSION;, you would run this on both nodes:

```
# echo drbd >/etc/modules-load.d/drbd.conf
```

Configure the Cluster for the Filesystem

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

In addition to defining the filesystem, we also need to tell the cluster where it can be located (only on the DRBD Primary) and when it is allowed to start (after the Primary was promoted).

We are going to take a shortcut when creating the resource this time. Instead of explicitly saying we want the **ocf:heartbeat:Filesystem** script, we are only going to ask for **Filesystem**. We can do this because we know there is only one resource script named **Filesystem** available to pacemaker, and that pcs is smart enough to fill in the **ocf:heartbeat:** portion for us correctly in the configuration. If there were multiple **Filesystem** scripts from different OCF providers, we would need to specify the exact one we wanted.

Once again, 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-ac
```

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 INFINI
[root@pcmk-1 ~]# pcs -f fs_cfg constraint order WebFS then WebSite
Adding WebFS WebSite (kind: Mandatory) (Options: first-action=start then-action=st
```

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. Any resources found active there will be moved elsewhere. 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.

Note

Since Fedora 29 and CentOS 8.0, the commands for controlling standby mode are `pcs node standby` and `pcs node unstandby`.

Chapter 9. Convert Storage 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.

Install Cluster Filesystem Software

The only hitch is that we need to use a cluster-aware filesystem. The one we used earlier with DRBD, xfs, is not one of those. Both OCFS2 and GFS2 are supported; here, we will use GFS2.

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 control daemon 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.

Warning

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

Important

Run the next command on whichever node has the DRBD Primary role. Otherwise, you will receive the message:

```
/dev/drbd1: Read-only file system
```

```
[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=
```

Clone the Filesystem Resource

Now that we have a cluster filesystem ready to go, we can configure the cluster so both nodes mount the filesystem.

Clone the filesystem resource 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 constraint
Location Constraints:
Ordering Constraints:
  start ClusterIP then start WebSite (kind:Mandatory)
  promote WebDataClone then start WebFS-clone (kind:Mandatory)
  start WebFS-clone then start WebSite (kind:Mandatory)
  start dlm-clone then start WebFS-clone (kind:Mandatory)
Colocation Constraints:
  WebSite with ClusterIP (score:INFINITY)
  WebFS-clone with WebDataClone (score:INFINITY) (with-rsc-role:Master)
  WebSite 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 ]
 ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1
 Clone Set: WebFS-clone [WebFS]
     Started: [ pcmk-1 pcmk-2 ]
 WebSite (ocf::heartbeat:apache): Started pcmk-1
```

Test Failover

Testing failover is left as an exercise for the reader.

With this configuration, the data is now active/active. The website administrator could change HTML files on either node, and the live website will show the changes even if it is running on the opposite node.

If the web server is configured to listen on all IP addresses, it is possible to remove the constraints between the WebSite and ClusterIP resources, and clone the WebSite resource. The web server would always be ready to serve web pages, and only the IP address would need to be moved in a failover.

Appendix A. Configuration Recap

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 ]
 ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1
 Clone Set: WebFS-clone [WebFS]
     Started: [ pcmk-1 pcmk-2 ]
 WebSite (ocf::heartbeat:apache): Started pcmk-1

[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 then start WebSite (kind:Mandatory)
  promote WebDataClone then start WebFS-clone (kind:Mandatory)
  start WebFS-clone then start WebSite (kind:Mandatory)
  start dlm-clone then start WebFS-clone (kind:Mandatory)
Colocation Constraints:
  WebSite with ClusterIP (score:INFINITY)
  WebFS-clone with WebDataClone (score:INFINITY) (with-rsc-role:Master)
  WebSite 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 ]
 ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1
 Clone Set: WebFS-clone [WebFS]
     Started: [ pcmk-1 pcmk-2 ]
 WebSite (ocf::heartbeat:apache): Started pcmk-1

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled

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

<configuration>
  <crm_config>
    <cluster_property_set id="cib-bootstrap-options">
      <nvpair id="cib-bootstrap-options-have-watchdog" name="have-watchdog" value=
      <nvpair id="cib-bootstrap-options-dc-version" name="dc-version" value="1.1.1
      <nvpair id="cib-bootstrap-options-cluster-infrastructure" name="cluster-infr
      <nvpair id="cib-bootstrap-options-cluster-name" name="cluster-name" value="m
      <nvpair id="cib-bootstrap-options-stonith-enabled" name="stonith-enabled" va
      <nvpair id="cib-bootstrap-options-last-lrm-refresh" name="last-lrm-refresh"
    </cluster_property_set>
  </crm_config>
  <nodes>
    <node id="1" uname="pcmk-1"/>
    <node id="2" uname="pcmk-2"/>
  </nodes>
  <resources>
    <primitive class="stonith" id="impi-fencing" type="fence_ipmilan">
      <instance_attributes id="impi-fencing-instance_attributes">
        <nvpair id="impi-fencing-instance_attributes-pcmk_host_list" name="pcmk_ho
        <nvpair id="impi-fencing-instance_attributes-ipaddr" name="ipaddr" value="
        <nvpair id="impi-fencing-instance_attributes-login" name="login" value="te
        <nvpair id="impi-fencing-instance_attributes-passwd" name="passwd" value="
      </instance_attributes>
      <operations>
        <op id="impi-fencing-interval-60s" interval="60s" name="monitor"/>
      </operations>
    </primitive>
    <master id="WebDataClone">
      <primitive class="ocf" id="WebData" provider="linbit" type="drbd">
        <instance_attributes id="WebData-instance_attributes">
          <nvpair id="WebData-instance_attributes-drbd_resource" name="drbd_resour
        </instance_attributes>
        <operations>
          <op id="WebData-demote-interval-0s" interval="0s" name="demote" timeout=
          <op id="WebData-monitor-interval-60s" interval="60s" name="monitor"/>
          <op id="WebData-notify-interval-0s" interval="0s" name="notify" timeout=
          <op id="WebData-promote-interval-0s" interval="0s" name="promote" timeou
          <op id="WebData-reload-interval-0s" interval="0s" name="reload" timeout=
          <op id="WebData-start-interval-0s" interval="0s" name="start" timeout="2
          <op id="WebData-stop-interval-0s" interval="0s" name="stop" timeout="100
        </operations>
```

```
  </primitive>
  <meta_attributes id="WebDataClone-meta_attributes">
    <nvpair id="WebDataClone-meta_attributes-master-node-max" name="master-nod
    <nvpair id="WebDataClone-meta_attributes-clone-max" name="clone-max" value
    <nvpair id="WebDataClone-meta_attributes-notify" name="notify" value="true
    <nvpair id="WebDataClone-meta_attributes-master-max" name="master-max" val
    <nvpair id="WebDataClone-meta_attributes-clone-node-max" name="clone-node-
  </meta_attributes>
</master>
<clone id="dlm-clone">
  <primitive class="ocf" id="dlm" provider="pacemaker" type="controld">
    <operations>
      <op id="dlm-monitor-interval-60s" interval="60s" name="monitor"/>
      <op id="dlm-start-interval-0s" interval="0s" name="start" timeout="90"/>
      <op id="dlm-stop-interval-0s" interval="0s" name="stop" timeout="100"/>
    </operations>
  </primitive>
  <meta_attributes id="dlm-clone-meta_attributes">
    <nvpair id="dlm-clone-meta_attributes-clone-max" name="clone-max" value="2
    <nvpair id="dlm-clone-meta_attributes-clone-node-max" name="clone-node-max
  </meta_attributes>
</clone>
<primitive class="ocf" id="ClusterIP" provider="heartbeat" type="IPaddr2">
  <instance_attributes id="ClusterIP-instance_attributes">
    <nvpair id="ClusterIP-instance_attributes-cidr_netmask" name="cidr_netmask
    <nvpair id="ClusterIP-instance_attributes-ip" name="ip" value="192.168.122
    <nvpair id="ClusterIP-instance_attributes-clusterip_hash" name="clusterip_
  </instance_attributes>
  <operations>
    <op id="ClusterIP-monitor-interval-30s" interval="30s" name="monitor"/>
    <op id="ClusterIP-start-interval-0s" interval="0s" name="start" timeout="2
    <op id="ClusterIP-stop-interval-0s" interval="0s" name="stop" timeout="20s
  </operations>
  <meta_attributes id="ClusterIP-meta_attributes">
    <nvpair id="ClusterIP-meta_attributes-resource-stickiness" name="resource-
  </meta_attributes>
</primitive>
<clone id="WebFS-clone">
  <primitive class="ocf" id="WebFS" provider="heartbeat" type="Filesystem">
    <instance_attributes id="WebFS-instance_attributes">
      <nvpair id="WebFS-instance_attributes-device" name="device" value="/dev/
      <nvpair id="WebFS-instance_attributes-directory" name="directory" value=
      <nvpair id="WebFS-instance_attributes-fstype" name="fstype" value="gfs2"
    </instance_attributes>
    <operations>
      <op id="WebFS-monitor-interval-20" interval="20" name="monitor" timeout=
      <op id="WebFS-notify-interval-0s" interval="0s" name="notify" timeout="6
      <op id="WebFS-start-interval-0s" interval="0s" name="start" timeout="60"
      <op id="WebFS-stop-interval-0s" interval="0s" name="stop" timeout="60"/>
    </operations>
  </primitive>
</clone>
<primitive class="ocf" id="WebSite" provider="heartbeat" type="apache">
  <instance_attributes id="WebSite-instance_attributes">
```

```
        <nvpair id="WebSite-instance_attributes-configfile" name="configfile" valu
        <nvpair id="WebSite-instance_attributes-statusurl" name="statusurl" value=
      </instance_attributes>
      <operations>
        <op id="WebSite-monitor-interval-1min" interval="1min" name="monitor"/>
        <op id="WebSite-start-interval-0s" interval="0s" name="start" timeout="40s
        <op id="WebSite-stop-interval-0s" interval="0s" name="stop" timeout="60s"/
      </operations>
    </primitive>
  </resources>
  <constraints>
    <rsc_colocation id="colocation-WebSite-ClusterIP-INFINITY" rsc="WebSite" score
    <rsc_order first="ClusterIP" first-action="start" id="order-ClusterIP-WebSite-
    <rsc_colocation id="colocation-WebFS-WebDataClone-INFINITY" rsc="WebFS-clone"
    <rsc_order first="WebDataClone" first-action="promote" id="order-WebDataClone-
    <rsc_colocation id="colocation-WebSite-WebFS-INFINITY" rsc="WebSite" score="IN
    <rsc_order first="WebFS-clone" first-action="start" id="order-WebFS-WebSite-ma
    <rsc_colocation id="colocation-WebFS-dlm-clone-INFINITY" rsc="WebFS-clone" sco
    <rsc_order first="dlm-clone" first-action="start" id="order-dlm-clone-WebFS-ma
  </constraints>
  <rsc_defaults>
    <meta_attributes id="rsc_defaults-options">
      <nvpair id="rsc_defaults-options-resource-stickiness" name="resource-stickin
    </meta_attributes>
  </rsc_defaults>
  <op_defaults>
    <meta_attributes id="op_defaults-options">
      <nvpair id="op_defaults-options-timeout" name="timeout" value="240s"/>
    </meta_attributes>
  </op_defaults>
</configuration>
```

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:
```

Cluster Options

```
[root@pcmk-1 ~]# pcs property
Cluster Properties:
 cluster-infrastructure: corosync
 cluster-name: mycluster
 dc-version: 1.1.18-11.el7_5.3-2b07d5c5a9
```

```
 have-watchdog: false
 last-lrm-refresh: 1536679009
 stonith-enabled: true
```

The output shows state information automatically obtained about the cluster, including:

- **cluster-infrastructure** - the cluster communications layer in use
- **cluster-name** - the cluster name chosen by the administrator when the cluster was created
- **dc-version** - the version (including upstream source-code hash) of Pacemaker used on the Designated Controller, which is the node elected to determine what actions are needed when events occur

The output also shows options set by the administrator that control the way the cluster operates, including:

- **stonith-enabled=true** - whether the cluster is allowed to use STONITH resources

Resources

Default Options

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

This shows cluster option defaults that apply to every resource that does not explicitly set the option itself. Above:

- **resource-stickiness** - Specify the aversion to moving healthy resources to other machines

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="p
  Operations: monitor interval=60s (fence-monitor-interval-60s)
```

Service Address

Users of the services provided by the cluster require an unchanging address with which to access it.

```
[root@pcmk-1 ~]# pcs resource show ClusterIP
Resource: ClusterIP (class=ocf provider=heartbeat type=IPaddr2)
 Attributes: cidr_netmask=24 ip=192.168.122.120 clusterip_hash=sourceip
 Meta Attrs: resource-stickiness=0
 Operations: monitor interval=30s (ClusterIP-monitor-interval-30s)
             start interval=0s timeout=20s (ClusterIP-start-interval-0s)
             stop interval=0s timeout=20s (ClusterIP-stop-interval-0s)
```

DRBD - Shared Storage

Here, we define the DRBD service and specify which DRBD resource (from /etc/drbd.d/*.res) it should manage. We make it a master clone resource and, in order to have an active/active setup, allow both

instances to be promoted to master at the same time. We also set the notify option so that the cluster will tell DRBD agent when its peer changes state.

```
[root@pcmk-1 ~]# pcs resource show WebDataClone
 Master: WebDataClone
  Meta Attrs: master-node-max=1 clone-max=2 notify=true master-max=2 clone-node-ma
  Resource: WebData (class=ocf provider=linbit type=drbd)
   Attributes: drbd_resource=wwwdata
   Operations: demote interval=0s timeout=90 (WebData-demote-interval-0s)
               monitor interval=60s (WebData-monitor-interval-60s)
               notify interval=0s timeout=90 (WebData-notify-interval-0s)
               promote interval=0s timeout=90 (WebData-promote-interval-0s)
               reload interval=0s timeout=30 (WebData-reload-interval-0s)
               start interval=0s timeout=240 (WebData-start-interval-0s)
               stop interval=0s timeout=100 (WebData-stop-interval-0s)
[root@pcmk-1 ~]# pcs constraint ref WebDataClone
Resource: WebDataClone
  colocation-WebFS-WebDataClone-INFINITY
  order-WebDataClone-WebFS-mandatory
```

Cluster Filesystem

The cluster filesystem ensures that files are read and written correctly. We need to specify the block device (provided by DRBD), where we want it mounted and that we are using GFS2. Again, it is a clone because it is intended to be active on both nodes. The additional constraints ensure that it can only be started on nodes with active DLM and DRBD instances.

```
[root@pcmk-1 ~]# pcs resource show WebFS-clone
 Clone: WebFS-clone
  Resource: WebFS (class=ocf provider=heartbeat type=Filesystem)
   Attributes: device=/dev/drbd1 directory=/var/www/html fstype=gfs2
   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)
[root@pcmk-1 ~]# pcs constraint ref WebFS-clone
Resource: WebFS-clone
  colocation-WebFS-WebDataClone-INFINITY
  colocation-WebSite-WebFS-INFINITY
  colocation-WebFS-dlm-clone-INFINITY
  order-WebDataClone-WebFS-mandatory
  order-WebFS-WebSite-mandatory
  order-dlm-clone-WebFS-mandatory
```

Apache

Lastly, we have the actual service, Apache. We need only tell the cluster where to find its main configuration file and restrict it to running on a node that has the required filesystem mounted and the IP address active.

```
[root@pcmk-1 ~]# pcs resource show WebSite
Resource: WebSite (class=ocf provider=heartbeat type=apache)
 Attributes: configfile=/etc/httpd/conf/httpd.conf statusurl=http://localhost/serv
 Operations: monitor interval=1min (WebSite-monitor-interval-1min)
```

```
              start interval=0s timeout=40s (WebSite-start-interval-0s)
              stop interval=0s timeout=60s (WebSite-stop-interval-0s)
[root@pcmk-1 ~]# pcs constraint ref WebSite
Resource: WebSite
  colocation-WebSite-ClusterIP-INFINITY
  colocation-WebSite-WebFS-INFINITY
  order-ClusterIP-WebSite-mandatory
  order-WebFS-WebSite-mandatory
```

Appendix B. Sample Corosync Configuration

Sample `corosync.conf` for two-node cluster created by `pcs`.

```
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
}
```

Appendix C. Further Reading

- Project Website https://www.clusterlabs.org/
- SuSE has a comprehensive guide to cluster commands (though using the `crmsh` command-line shell rather than `pcs`) at: https://www.suse.com/documentation/sle_ha/book_sleha/data/book_sleha.html
- Corosync http://www.corosync.org/

Appendix D. Revision History

Revision History		
Revision 1-0	Mon May 17 2010	AndrewBeekhof<andrew@beekhof.net>
Import from Pages.app		
Revision 2-0	Wed Sep 22 2010	RaoulScarazzini<rasca@miamammausalinux.c
Italian translation		
Revision 3-0	Wed Feb 9 2011	AndrewBeekhof<andrew@beekhof.net>
Updated for Fedora 13		
Revision 4-0	Wed Oct 5 2011	AndrewBeekhof<andrew@beekhof.net>
Update the GFS2 section to use CMAN		
Revision 5-0	Fri Feb 10 2012	AndrewBeekhof<andrew@beekhof.net>
Generate docbook content from asciidoc sources		
Revision 6-0	Tues July 3 2012	AndrewBeekhof<andrew@beekhof.net>
Updated for Fedora 17		
Revision 7-0	Fri Sept 14 2012	DavidVossel<davidvossel@gmail.com>
Updated for pcs		
Revision 8-0	Mon Jan 05 2015	KenGaillot<kgaillot@redhat.com>
Updated for Fedora 21		
Revision 8-1	Thu Jan 08 2015	KenGaillot<kgaillot@redhat.com>
Minor corrections, plus use include file for intro		
Revision 9-0	Fri Aug 14 2015	KenGaillot<kgaillot@redhat.com>
Update for CentOS 7.1 and leaving firewalld/SELinux enabled		
Revision 10-0	Fri Jan 12 2018	KenGaillot<kgaillot@redhat.com>
Update banner for Pacemaker 2.0 and content for CentOS 7.4 with Pacemaker 1.1.16		
Revision 10-1	Wed Sep 5 2018	KenGaillot<kgaillot@redhat.com>
Update for CentOS 7.5 with Pacemaker 1.1.18		
Revision 10-2	Fri Dec 7 2018	KenGaillot<kgaillot@redhat.com>, JanPokorný<jpokorny@redhat.com>, ChrisLumens<clumens@redhat.com>
Minor clarifications and formatting changes		
Revision 11-0	Thu Jul 18 2019	TomasJelinek<tojeline@redhat.com>
Note differences in pcs 0.10		
Revision 11-1	Thu Nov 21 2019	KenGaillot<kgaillot@redhat.com>
Remove references to obsolete cloned IP usage, and reorganize chapters a bit		

Index

Symbols

A

C

D

F

N

S