Pacemaker 1.1

Clusters from Scratch

Creating Active/Passive and Active/Active Clusters on Fedora

Andrew Beekhof

Pacemaker 1.1 Clusters from Scratch
Creating Active/Passive and Active/Active Clusters on Fedora
Edition 5

Author	Andrew Beekhof	*andrew@beekhof.net*
Translator	Raoul Scarazzini	*rasca@miamammausalinux.org*
Translator	Dan Frîncu	*df.cluster@gmail.com*

The purpose of this document is to provide a start-to-finish guide to building an example active/passive cluster with Pacemaker and show how it can be converted to an active/active one.

The example cluster will use:

1. Fedora 17 as the host operating system

2. Corosync to provide messaging and membership services,

3. Pacemaker to perform resource management,

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 Fedora 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] An explanation of CC-BY-SA is available at *http://creativecommons.org/licenses/by-sa/3.0/*

Table of Contents

List of Figures

List of Examples

Preface

Table of Contents

1. Document Conventions

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

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. 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:

[1] https://fedorahosted.org/liberation-fonts/

> 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.

1.2. 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"));
   }
}
```

1.3. 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.

2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla[2] against the product **Pacemaker.**

When submitting a bug report, be sure to mention the manual's identifier: *Clusters_from_Scratch*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

[2] *http://bugs.clusterlabs.org*

Read-Me-First

Table of Contents

1.1. 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 Fedora distribution, version 20.

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*[1] manual.

1.2. What Is Pacemaker?

Pacemaker is a cluster resource manager.

It achieves maximum availability for your cluster services (aka. resources) by detecting and recovering from node and resource-level failures by making use of the messaging and membership capabilities provided by your preferred cluster infrastructure (either *Corosync*[2] or *Heartbeat*[3]).

Pacemaker's key features include:

- Detection and recovery of node and service-level failures
- Storage agnostic, no requirement for shared storage
- Resource agnostic, anything that can be scripted can be clustered
- Supports STONITH for ensuring data integrity
- Supports large and small clusters

[1] http://www.clusterlabs.org/doc/
[2] http://www.corosync.org/
[3] http://linux-ha.org/wiki/Heartbeat

- Supports both quorate and resource driven clusters
- Supports practically any redundancy configuration
- Automatically replicated configuration that can be updated from any node
- Ability to specify cluster-wide service ordering, colocation and anti-colocation
- Support for advanced service types
  - Clones: for services which need to be active on multiple nodes
  - Multi-state: for services with multiple modes (eg. master/slave, primary/secondary)
- Unified, scriptable, cluster management tools.

1.3. Pacemaker Architecture

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

- Non-cluster aware components. These pieces include the resources themselves, scripts that start, stop and monitor them, and also a local daemon that masks the differences between the different standards these scripts implement.
- Resource management. Pacemaker provides the brain that processes and reacts to events regarding the cluster. These events include nodes joining or leaving the cluster; resource events caused by failures, maintenance, scheduled activities; and other administrative actions. Pacemaker will compute the ideal state of the cluster and plot a path to achieve it after any of these events. This may include moving resources, stopping nodes and even forcing them offline with remote power switches.
- Low level infrastructure. Projects like Corosync, CMAN and Heartbeat provide reliable messaging, membership and quorum information about the cluster.

When combined with Corosync, Pacemaker also supports popular open source cluster filesystems. footnote:[Even though Pacemaker also supports Heartbeat, the filesystems need to use the stack for messaging and membership and Corosync seems to be what they're standardizing on.

Technically it would be possible for them to support Heartbeat as well, however there seems little interest in this.]

Due to past standardization within the cluster filesystem community, they make use of a common distributed lock manager which makes use of Corosync for its messaging and membership capabilities (which nodes are up/down) and Pacemaker for fencing services.

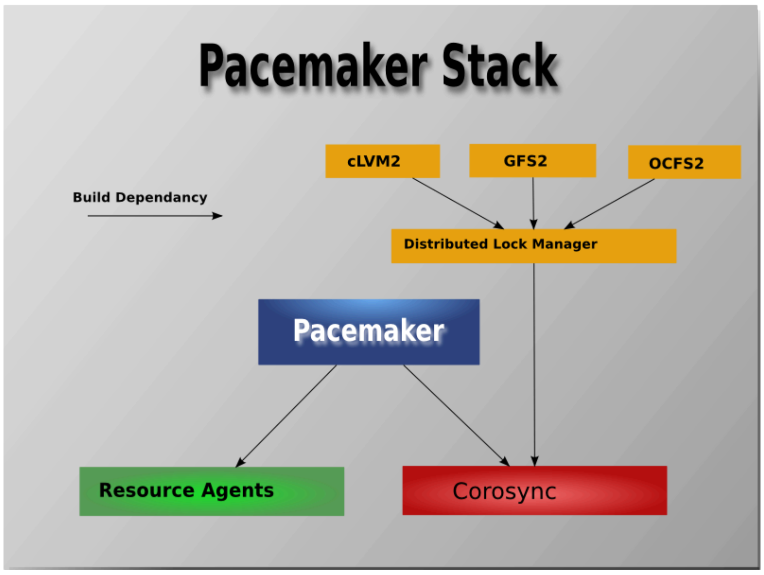


Figure 1.1. The Pacemaker Stack

1.3.1. Internal Components

Pacemaker itself is composed of five key components:

- CIB (aka. Cluster Information Base)
- CRMd (aka. Cluster Resource Management daemon)
- LRMd (aka. Local Resource Management daemon)
- PEngine (aka. PE or Policy Engine)
- STONITHd

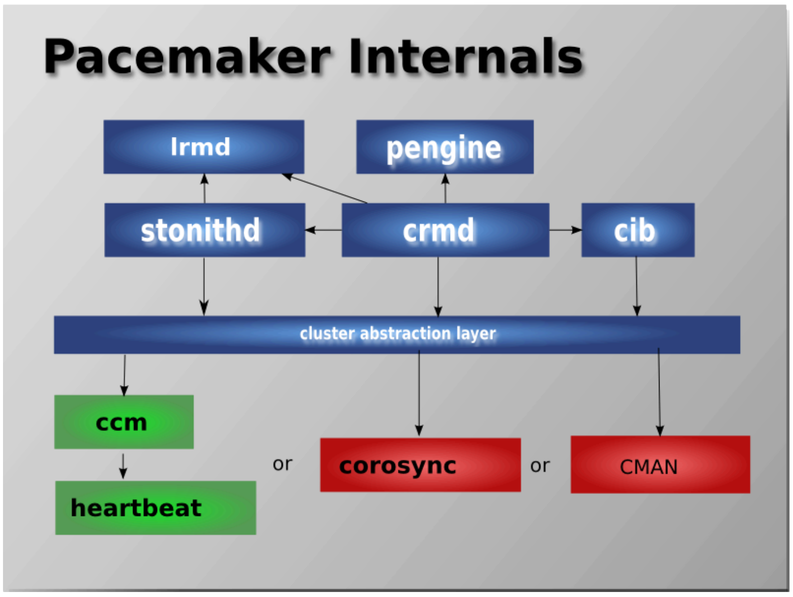


Figure 1.2. Internal Components

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

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

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

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

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

STONITH is an acronym for Shoot-The-Other-Node-In-The-Head and is usually implemented with a remote power switch.

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

1.4. Types of Pacemaker Clusters

Pacemaker makes no assumptions about your environment, this allows it to support practically any *redundancy configuration*[4] including Active/Active, Active/Passive, N+1, N+M, N-to-1 and N-to-N.

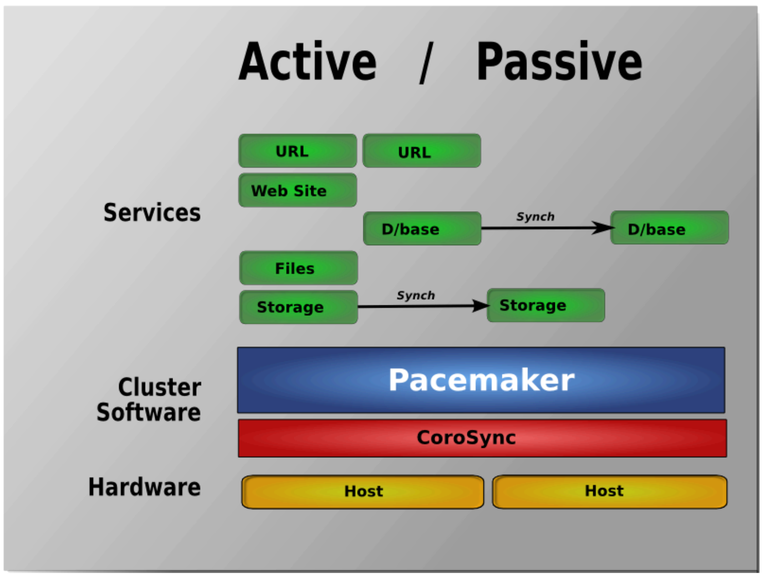


Figure 1.3. Active/Passive Redundancy

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

[4] http://en.wikipedia.org/wiki/High-availability_cluster#Node_configurations

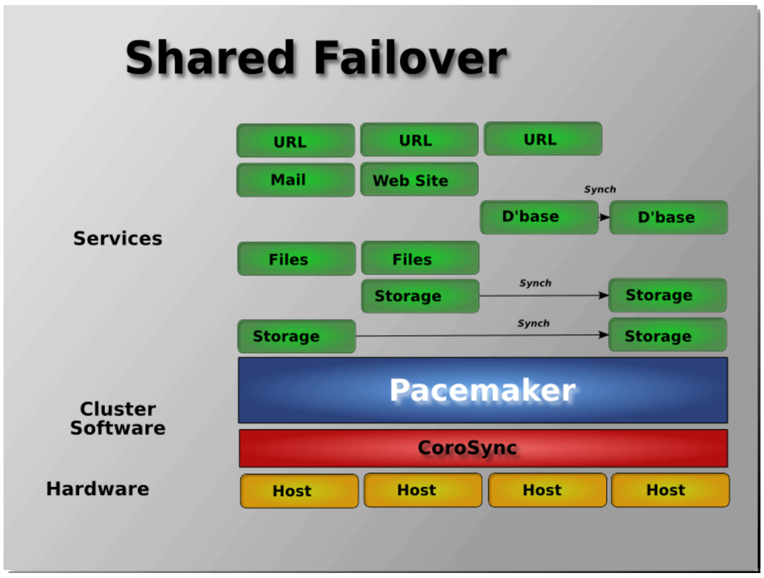


Figure 1.4. Shared Failover

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

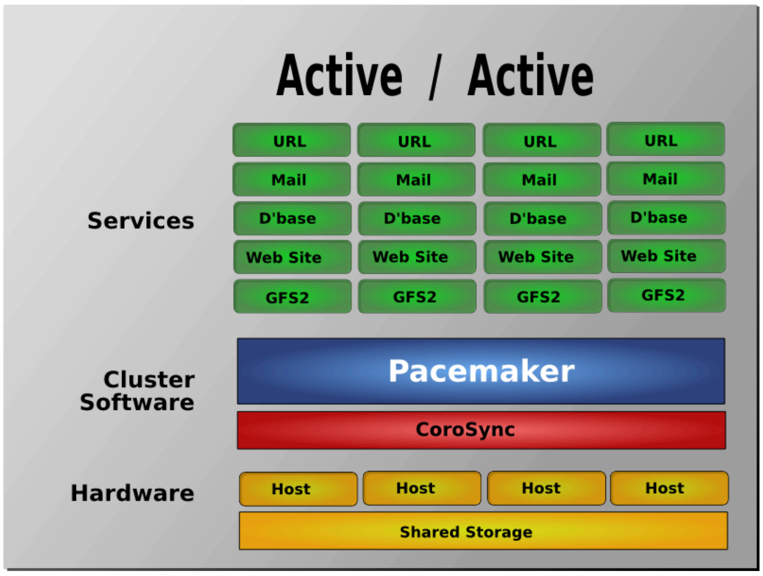


Figure 1.5. N to N Redundancy

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.

Installation

Table of Contents

2.1. OS Installation

Detailed instructions for installing Fedora are available at *http://docs.fedoraproject.org/en-US/Fedora/20/html/Installation_Guide/* in a number of languages. The abbreviated version is as follows…

Point your browser to *http://fedoraproject.org/en/get-fedora-all*, locate the **`Install Media`** section and download the install DVD that matches your hardware.

Burn the disk image to a DVD [1] and boot from it, or use the image to boot a virtual machine.

After clicking through the welcome screen, select your language, keyboard layout [2]

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

In the **`Network Configuration`** section you'll want to:

- Assign your machine a host name I happen to control the clusterlabs.org domain name, so I will use that here.
- Assign a fixed IP address

[1] *http://docs.fedoraproject.org/en-US/Fedora/20/html/Burning_ISO_images_to_disc/index.html*
[2] *http://docs.fedoraproject.org/en-US/Fedora/20/html/Installation_Guide/language-selection-x86.html*

Important

Do not accept the default network settings. Cluster machines should never obtain an IP address via DHCP.

If you miss this step, this can easily be configured after installation. You will have to navigate to **`system settings`** and select **`network`**. From there you can select what device to configure.

In the **`Software Selection`** section (try saying that 10 times quickly), choose **`Minimal Install`** so that we see everything that gets installed. Don't enable updates yet, we'll do that (and install any extra software we need) later.

Important

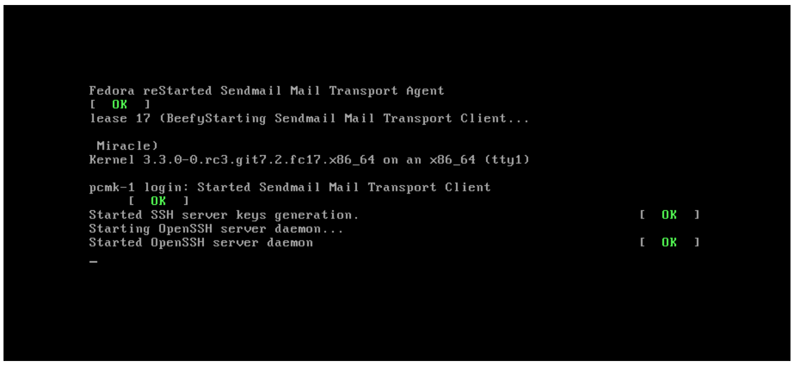
By default Fedora uses LVM for partitioning which allows us to dynamically change the amount of space allocated to a given partition.

However, by default it also allocates all free space to the **`/`** (aka. **`root`**) partition which cannot be dynamically *reduced* in size (dynamic increases are fine by-the-way).

So if you plan on following the DRBD or GFS2 portions of this guide, you should reserve at least 1Gb of space on each machine from which to create a shared volume. To do so, enter the **`Installation Destination`** section where you are be given an opportunity to reduce the size of the **`root`** partition (after chosing which hard drive you wish to install to).

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. You can do this in the **`Date & Time`** section. [3]

Once the node reboots, you'll see a (possibly mangled) login prompt on the console. Login using **`root`** and the password you created earlier.



```
Fedora reStarted Sendmail Mail Transport Agent
[  OK  ]
lease 17 (BeefyStarting Sendmail Mail Transport Client...

 Miracle)
Kernel 3.3.0-0.rc3.git7.2.fc17.x86_64 on an x86_64 (tty1)

pcmk-1 login: Started Sendmail Mail Transport Client
     [  OK  ]
Started SSH server keys generation.                                           [  OK  ]
Starting OpenSSH server daemon...
Started OpenSSH server daemon                                                 [  OK  ]
_
```

[3] http://docs.fedoraproject.org/en-US/Fedora/20/html/Installation_Guide/s1-timezone-x86.html

Note

From here on in we’re going to be working exclusively from the terminal.

2.2. Post Installation Tasks

2.2.1. Networking

Check the machine has the static IP address you configured earlier

```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    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 qlen 1000
    link/ether 52:54:00:d7:d6:08 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.101/24 brd 192.168.122.255 scope global eth0
    inet6 fe80::5054:ff:fed7:d608/64 scope link
       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:

```
# manually edit /etc/sysconfig/network-scripts/ifcfg-${device}
# nmcli dev disconnect ${device}
# nmcli con reload ${device}
# nmcli con up ${device}
```

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

Next, check the routes are ok:

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

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 */etc/sysconfig/network* and restart the network.

Now check for connectivity to the outside world. Start small by testing if we can read the gateway we configured.

```
# 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_req=1 ttl=64 time=0.249 ms

 --- 192.168.122.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.249/0.249/0.249/0.000 ms
```

Now try something external, choose a location you know will be available.

```
# ping -c 1 www.google.com
PING www.l.google.com (173.194.72.106) 56(84) bytes of data.
64 bytes from tf-in-f106.1e100.net (173.194.72.106): icmp_req=1 ttl=41 time=167 ms

 --- www.l.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 167.618/167.618/167.618/0.000 ms
```

2.2.2. Leaving the Console

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

First we check we can see the newly installed 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 we login via SSH

```
beekhof@f16 ~ # ssh -l root 192.168.122.11
root@192.168.122.11's password:
Last login: Fri Mar 30 19:41:19 2012 from 192.168.122.1
[root@pcmk-1 ~]#
```

2.2.3. Security Shortcuts

To simplify this guide and focus on the aspects directly connected to clustering, we will now disable the machine’s firewall and SELinux installation.

Warning

Both of these actions create significant security issues and should not be performed on machines that will be exposed to the outside world.

Important

```
TODO: Create an Appendix that deals with (at least) re-enabling the firewall.
```

```
# setenforce 0
# sed -i.bak "s/SELINUX=enforcing/SELINUX=permissive/g" /etc/selinux/config
# systemctl disable iptables.service
# rm '/etc/systemd/system/basic.target.wants/iptables.service'
# systemctl stop iptables.service
```

2.2.4. 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:

```
# uname -n
pcmk-1.clusterlabs.org
# dnsdomainname
clusterlabs.org
```

The output from the second command is fine, but we really don't need the domain name included in the basic host details. To address this, we need to use the **`hostnamectl`** tool to strip off the domain name.

```
# hostnamectl set-hostname $(uname -n | sed s/\\..*//)'
```

Now check the machine is using the correct names

```
# uname -n
pcmk-1
# dnsdomainname
clusterlabs.org
```

If it concerns you that the shell prompt has not been updated, simply log out and back in again.

2.3. Before You Continue

Repeat the Installation steps so far, so that you have two Fedora 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.

2.3.1. Finalize Networking

Confirm that you can communicate between the two new nodes:

```
# 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* . Below are the entries for my cluster nodes:

```
# 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:

```
# 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=0.164 ms
64 bytes from pcmk-1.clusterlabs.org (192.168.122.101): icmp_seq=2 ttl=64 time=0.475 ms
64 bytes from pcmk-1.clusterlabs.org (192.168.122.101): icmp_seq=3 ttl=64 time=0.186 ms

--- 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
```

2.3.2. 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

```
# 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        |
|                 |
|                 |
|                 |
|                 |
+-----------------+

# cp .ssh/id_dsa.pub .ssh/authorized_keys
```

Install the key on the other nodes and test that you can now run commands remotely, without being prompted

Installing the SSH Key on Another Host

```
# scp -r .ssh pcmk-2:
The authenticity of host 'pcmk-2 (192.168.122.102)' can't be established.
RSA key fingerprint is b1:2b:55:93:f1:d9:52:2b:0f:f2:8a:4e:ae:c6:7c:9a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pcmk-2,192.168.122.102' (RSA) to the list of known
 hosts.root@pcmk-2's password:
id_dsa.pub                                    100%  616     0.6KB/s   00:00
id_dsa                                        100%  672     0.7KB/s   00:00
known_hosts                                   100%  400     0.4KB/s   00:00
authorized_keys                               100%  616     0.6KB/s   00:00
# ssh pcmk-2 -- uname -n
pcmk-2
#
```

2.4. Cluster Software Installation

2.4.1. Install the Cluster Software

Since version 12, Fedora comes with recent versions of everything you need, so simply fire up a shell on all your nodes and run:

```
[ALL] # yum install -y pacemaker pcs
```

Now install the cluster software on the second node.

2.4.2. Install the Cluster Management Software

The pcs cli command coupled with the pcs daemon creates a cluster management system capable of managing all aspects of the cluster stack across all nodes from a single location.

```
[ALL] # yum install -y pcs
```

Make sure to install the pcs packages on both nodes.

2.5. Setup

2.5.1. Enable pcs Daemon

Before the cluster can be configured, the pcs daemon must be started and enabled to boot on startup on each node. This daemon works with the pcs cli command to manage syncing the corosync configuration across all the 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
```

Now we need a way for **`pcs`** to talk to itself on other nodes in the cluster. This is necessary in order to perform tasks such as syncing the corosync config, or starting/stopping the cluster on remote nodes

While **`pcs`** can be used locally without setting up these user accounts, this tutorial will make use of these remote access commands, so we will set a password for the *hacluster* user. Its probably best if password is consistent across all the nodes.

As *root*, run:

```
# passwd hacluster
password:
```

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

```
# ssh pcmk-2 -- 'echo redhat1 | passwd --stdin hacluster'
```

2.5.2. Notes on Multicast Address Assignment

There are several subtle points that often deserve consideration when choosing/assigning multicast addresses for corosync. [4]

1. Avoid *224.0.0.x*

   Traffic to addresses of the form *224.0.0.x* is often flooded to all switch ports. This address range is reserved for link-local uses. Many routing protocols assume that all traffic within this range will be received by all routers on the network. Hence (at least all Cisco) switches flood traffic within this range. The flooding behavior overrides the normal selective forwarding behavior of a multicast-aware switch (e.g. IGMP snooping, CGMP, etc.).

2. Watch for *32:1* overlap

   32 non-contiguous IP multicast addresses are mapped onto each Ethernet multicast address. A receiver that joins a single IP multicast group implicitly joins 31 others due to this overlap. Of course, filtering in the operating system discards undesired multicast traffic from applications, but NIC bandwidth and CPU resources are nonetheless consumed discarding it. The overlap occurs in the 5 high-order bits, so it's best to use the 23 low-order bits to make distinct multicast streams unique. For example, IP multicast addresses in the range *239.0.0.0* to *239.127.255.255* all map

[4] This information is borrowed from, the now defunct, *http://web.archive.org/web/20101211210054/http://29west.com/docs/THPM/multicast-address-assignment.html*

to unique Ethernet multicast addresses. However, IP multicast address *239.128.0.0* maps to the same Ethernet multicast address as *239.0.0.0*, *239.128.0.1* maps to the same Ethernet multicast address as *239.0.0.1*, etc.

3. Avoid *x.0.0.y* and *x.128.0.y*

   Combining the above two considerations, it’s best to avoid using IP multicast addresses of the form *x.0.0.y* and *x.128.0.y* since they all map onto the range of Ethernet multicast addresses that are flooded to all switch ports.

4. Watch for address assignment conflicts

   IANA[5] administers *Internet multicast addresses*[6]. Potential conflicts with Internet multicast address assignments can be avoided by using *GLOP addressing*[7] (*AS*[8] required) or *administratively scoped*[9] addresses. Such addresses can be safely used on a network connected to the Internet without fear of conflict with multicast sources originating on the Internet. Administratively scoped addresses are roughly analogous to the unicast address space for *private internets*[10]. Site-local multicast addresses are of the form *239.255.x.y*, but can grow down to *239.252.x.y* if needed. Organization-local multicast addresses are of the form *239.192-251.x.y*, but can grow down to *239.x.y.z* if needed.

For a more detailed treatment (57 pages!), see *Cisco’s Guidelines for Enterprise IP Multicast Address Allocation*[11] paper.

2.5.3. Configuring Corosync

In the past, at this point in the tutorial an explanation of how to configure and propagate corosync’s /etc/corosync.conf file would be necessary. Using pcs with the pcs daemon greatly simplifies this process by generating *corosync.conf* across all the nodes in the cluster with a single command. The only thing required to achieve this is to authenticate as the pcs user *hacluster* on one of the nodes in the cluster, and then issue the *pcs cluster setup* command with a list of all the node names in the cluster.

```
# pcs cluster auth pcmk-1 pcmk-2
Username: hacluster
Password:
pcmk-1: Authorized
pcmk-2: Authorized

# pcs cluster setup --name mycluster pcmk-1 pcmk-2
pcmk-1: Succeeded
pcmk-2: Succeeded
```

That’s it. Corosync is configured across the cluster. If you received an authorization error for either of those commands, make sure you setup the *hacluster* user account and password on every node in the cluster with the same password.

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

[5] http://www.iana.org/
[6] http://www.iana.org/assignments/multicast-addresses
[7] http://www.ietf.org/rfc/rfc3180.txt
[8] http://en.wikipedia.org/wiki/Autonomous_system_%28Internet%29
[9] http://www.ietf.org/rfc/rfc2365.txt
[10] http://www.ietf.org/rfc/rfc1918.txt
[11] http://www.cisco.com/en/US/tech/tk828/technologies_white_paper09186a00802d4643.shtml

Important

Pacemaker used to obtain membership and quorum from a custom Corosync plugin. This plugin also had the capability to start Pacemaker automatically when Corosync was started.

Neither behavior is possible with Corosync 2.0 and beyond as support for plugins was removed. Instead, Pacemaker must be started as a separate service.

Also, since Pacemaker made use of the plugin for message routing, a node using the plugin (Corosync prior to 2.0) cannot talk to one that isn't (Corosync 2.0+). Rolling upgrades between these versions are therefor not possible and an alternate strategy [12] must be used.

[12] *http://www.clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/ap-upgrade.html*

Pacemaker Tools

Table of Contents

3.1. Using Pacemaker Tools

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.

All of that has been greatly simplified with the creation of unified command-line shells (and GUIs) that hide all the messy XML scaffolding.

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

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

There are currently two command-line shells that people use, **pcs** and **crmsh**. This edition of Clusters from Scratch is based on **pcs**. Start by taking some time to familiarize yourself with what it can do.

Note

The two shells share many concepts but the scope, layout and syntax does differ, so make sure you read the version of this guide that corresponds to the software installed on your system.

Important

Since **pcs** has the ability to manage all aspects of the cluster (both corosync and pacemaker), it requires a specific cluster stack to be in use, (corosync 2.0 with votequorum + Pacemaker version >= 1.1.8).

```
# pcs
```

```
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

Commands:
    cluster     Configure cluster options and nodes
    resource    Manage cluster resources
    stonith     Configure fence devices
    constraint  Set resource constraints
    property    Set pacemaker properties
    status      View cluster status
    config      Print full cluster configuration
```

As you can see, the different aspects of cluster management are broken up into categories: resource, cluster, stonith, property, constraint, and status. 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.

```
# pcs status help
```

```
Usage: pcs status [commands]...
View current cluster and resource status
Commands:
    [status]
        View all information about the cluster and resources

    resources
        View current status of cluster resources

    groups
        View currently configured groups and their resources

    cluster
        View current cluster status

    corosync
        View current membership information as seen by corosync

    nodes [corosync|both|config]
        View current status of nodes from pacemaker. If 'corosync' is
        specified, print nodes currently configured in corosync, if 'both'
        is specified, print nodes from both corosync & pacemaker.  If 'config'
        is specified, print nodes from corosync & pacemaker configuration.

    pcsd <node> ...
        Show the current status of pcsd on the specified nodes

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

Additionally, if you are interested in the Pacemaker version and supported cluster stack(s) available with your current Pacemaker installation, the pacemakerd --features option is available to you.

```
# pacemakerd --features
```

```
Pacemaker 1.1.11 (Build: 9fe61fb)
 Supporting v3.0.9:  generated-manpages agent-manpages ascii-docs publican-docs ncurses
 libqb-logging libqb-ipc upstart systemd nagios  corosync-native snmp acls
```

Note

If the SNMP and/or email options are not listed, then Pacemaker was not built to support them. This may be by the choice of your distribution or the required libraries may not have been available. Please contact whoever supplied you with the packages for more details.

Verify Cluster Installation

Table of Contents

4.1. 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 current node you are logged into before you will be allowed to start the cluster.

```
# pcs cluster start --all
pcmk-1: Starting Cluster...
pcmk-2: Starting Cluster...
```

An alternative to using the *pcs cluster startall* command is to issue either of the below commands on each node in the cluster by hand.

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

or

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

4.2. Verify Corosync Installation

The first thing to check is if cluster communication is happy, for that we use **`corosync-cfgtool`**.

```
# 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 we check the membership and quorum APIs:

```
# corosync-cmapctl  | grep members
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.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

# 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.

All good!

4.3. 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.

```
# ps axf
  PID TTY      STAT   TIME COMMAND
    2 ?        S      0:00 [kthreadd]
...lots of processes...
28019 ?        Ssl    0:03 /usr/sbin/corosync
28047 ?        Ss     0:00 /usr/sbin/pacemakerd -f
28048 ?        Ss     0:00  \_ /usr/libexec/pacemaker/cib
28049 ?        Ss     0:00  \_ /usr/libexec/pacemaker/stonithd
28050 ?        Ss     0:00  \_ /usr/lib64/heartbeat/lrmd
28051 ?        Ss     0:00  \_ /usr/libexec/pacemaker/attrd
28052 ?        Ss     0:00  \_ /usr/libexec/pacemaker/pengine
28053 ?        Ss     0:00  \_ /usr/libexec/pacemaker/crmd
```

If that looks ok, check the pcs status output.

```
# pcs status
Last updated: Fri Sep 14 09:52:25 2012
Last change: Fri Sep 14 09:51:55 2012 via crmd on pcmk-2
Stack: corosync
Current DC: pcmk-2 (2) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
0 Resources configured.

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:
```

Next, check for any ERRORs during startup - there shouldn't be any.

```
# grep -i error /var/log/messages
```

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

Creating an Active/Passive Cluster

Table of Contents

5.1. Exploring the Existing Configuration

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

This is what the base configuration should look like.

```
# pcs status
Last updated: Fri Sep 14 10:12:01 2012
Last change: Fri Sep 14 09:51:55 2012 via crmd on pcmk-2
Stack: corosync
Current DC: pcmk-1 (1) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
0 Resources configured.

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:
```

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

Example 5.1. The last XML you'll see in this document

```
# pcs cluster cib
```

```
<cib epoch="4" num_updates="19" admin_epoch="0" validate-with="pacemaker-1.2"
 crm_feature_set="3.0.6" update-origin="pcmk-1" update-client="crmd" cib-last-written="Wed
 Aug  1 16:08:52 2012" have-quorum="1" dc-uuid="1">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair id="cib-bootstrap-options-dc-version" name="dc-version"
 value="1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0"/>
        <nvpair id="cib-bootstrap-options-cluster-infrastructure" name="cluster-
infrastructure" value="corosync"/>
      </cluster_property_set>
    </crm_config>
    <nodes>
      <node id="1" uname="pcmk-1" type="normal"/>
      <node id="2" uname="pcmk-2" type="normal"/>
    </nodes>
    <resources/>
    <constraints/>
  </configuration>
  <status>
    <node_state id="2" uname="pcmk-2" ha="active" in_ccm="true" crmd="online"
 join="member" expected="member" crm-debug-origin="do_state_transition" shutdown="0">
      <lrm id="2">
        <lrm_resources/>
      </lrm>
      <transient_attributes id="2">
        <instance_attributes id="status-2">
          <nvpair id="status-2-probe_complete" name="probe_complete" value="true"/>
        </instance_attributes>
      </transient_attributes>
    </node_state>
    <node_state id="1" uname="pcmk-1" ha="active" in_ccm="true" crmd="online"
 join="member" expected="member" crm-debug-origin="do_state_transition" shutdown="0">
      <lrm id="1">
        <lrm_resources/>
      </lrm>
      <transient_attributes id="1">
        <instance_attributes id="status-1">
          <nvpair id="status-1-probe_complete" name="probe_complete" value="true"/>
        </instance_attributes>
      </transient_attributes>
    </node_state>
  </status>
</cib>
```

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

```
# crm_verify -L -V
   error: unpack_resources: Resource start-up disabled since no STONITH resources have been
 defined
   error: unpack_resources: Either configure some or disable STONITH with the stonith-enabled
 option
   error: unpack_resources: NOTE: Clusters with shared data need STONITH to ensure data
 integrity
Errors found during check: config not valid
  -V may provide more details
```

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

In order to guarantee the safety of your data [1] , the default for STONITH [2] in Pacemaker is **`enabled`**. However it also knows when no STONITH configuration has been supplied and reports this as a problem (since the cluster would not be able to make progress if a situation requiring node fencing arose).

For now, we will disable this feature and configure it later in the Configuring STONITH section. It is important to note that the use of STONITH is highly encouraged, turning it off tells the cluster to simply pretend that failed nodes are safely powered off. Some vendors will even refuse to support clusters that have it disabled.

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

```
# pcs property set stonith-enabled=false
# crm_verify -L
```

With the new cluster option set, the configuration is now valid.

[1] If the data is corrupt, there is little point in continuing to make it available
[2] A common node fencing mechanism. Used to ensure data integrity by powering off "bad" nodes

Warning

The use of stonith-enabled=false is completely inappropriate for a production cluster. We use it here to defer the discussion of its configuration which can differ widely from one installation to the next. See *Section 9.1, “What Is STONITH”* for information on why STONITH is important and details on how to configure it.

5.2. Adding a Resource

The first thing we should do is configure an IP address. Regardless of where the cluster service(s) are running, we need a consistent address to contact them on. Here I will choose and add 192.168.122.120 as the floating address, give it the imaginative name ClusterIP and tell the cluster to check that its running every 30 seconds.

Important

The chosen address must not be one already associated with a physical node

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 \
    ip=192.168.0.120 cidr_netmask=32 op monitor interval=30s
```

The other 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, is the standard to which the resource script conforms to and where to find it. The second field is specific to OCF resources and tells the cluster which namespace to find the resource script in, in this case heartbeat. The last field indicates the name of the resource script.

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

```
# pcs resource standards
ocf
lsb
service
systemd
stonith
```

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

```
# pcs resource providers
heartbeat
linbit
pacemaker
redhat
```

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

```
# pcs resource agents ocf:heartbeat
AoEtarget
AudibleAlarm
CTDB
ClusterMon
Delay
Dummy
.
. (skipping lots of resources to save space)
.
IPaddr2
.
.
.
symlink
syslog-ng
tomcat
vmware
```

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

```
# pcs status

Last updated: Fri Sep 14 10:17:00 2012
Last change: Fri Sep 14 10:15:48 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-1 (1) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
1 Resources configured.

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

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

5.3. Perform a Failover

Being a high-availability cluster, we should test failover of our new resource before moving on.

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

```
# pcs status

Last updated: Fri Sep 14 10:17:00 2012
Last change: Fri Sep 14 10:15:48 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-1 (1) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
1 Resources configured.

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

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

Shut down Pacemaker and Corosync on that machine.

```
#pcs cluster stop pcmk-1
Stopping Cluster...
```

Once Corosync is no longer running, go to the other node and check the cluster status.

```
# pcs status

Last updated: Fri Sep 14 10:31:01 2012
Last change: Fri Sep 14 10:15:48 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-2 (2) - partition WITHOUT quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
1 Resources configured.

Online: [ pcmk-2 ]
OFFLINE: [ pcmk-1 ]

Full list of resources:

 ClusterIP      (ocf::heartbeat:IPaddr2):       Stopped
```

There are three things to notice about the cluster's current state. The first is that, as expected, **`pcmk-1`** is now offline. However we can also see that **`ClusterIP`** isn't running anywhere!

5.3.1. Quorum and Two-Node Clusters

This is because the cluster no longer has quorum, as can be seen by the text "partition WITHOUT quorum" in the status output. In order to reduce the possibility of data corruption, Pacemaker's default behavior is to stop all resources if the cluster does not have quorum.

A cluster is said to have quorum when more than half the known or expected nodes are online, or for the mathematically inclined, whenever the following equation is true:

```
total_nodes < 2 * active_nodes
```

Therefore a two-node cluster only has quorum when both nodes are running, which is no longer the case for our cluster. This would normally make the creation of a two-node cluster pointless [3] , however it is possible to control how Pacemaker behaves when quorum is lost. In particular, we can tell the cluster to simply ignore quorum altogether.

```
# pcs property set no-quorum-policy=ignore
# pcs property
dc-version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
cluster-infrastructure: corosync
stonith-enabled: false
no-quorum-policy: ignore
```

After a few moments, the cluster will start the IP address on the remaining node. Note that the cluster still does not have quorum.

```
# pcs status
Last updated: Fri Sep 14 10:38:11 2012
```

[3] Actually some would argue that two-node clusters are always pointless, but that is an argument for another time

```
Last change: Fri Sep 14 10:37:53 2012 via cibadmin on pcmk-2
Stack: corosync
Current DC: pcmk-2 (2) - partition WITHOUT quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
1 Resources configured.

Online: [ pcmk-2 ]
OFFLINE: [ pcmk-1 ]

Full list of resources:

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

Now simulate node recovery by restarting the cluster stack on **`pcmk-1`** and check the cluster's status. Note, if you get an authentication error with the *pcs cluster start pcmk-1* command, you must authenticate on the node using the *pcs cluster auth pcmk pcmk-1 pcmk-2* command discussed earlier.

```
# pcs cluster start pcmk-1
Starting Cluster...
# pcs status

Last updated: Fri Sep 14 10:42:56 2012
Last change: Fri Sep 14 10:37:53 2012 via cibadmin on pcmk-2
Stack: corosync
Current DC: pcmk-2 (2) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
1 Resources configured.

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

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

Note

In the dark days, the cluster may have moved the IP back to its original location (**`pcmk-1`**). Usually this is no longer the case.

5.3.2. 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 like Oracle databases, this period can be quite long.

To address this, Pacemaker has the concept of resource stickiness which controls how much 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" [4] resource placement. We can specify a different stickiness for every resource, but it is often sufficient to change the default.

[4] It should be noted that 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

```
# pcs resource defaults resource-stickiness=100
# pcs resource defaults
resource-stickiness: 100
```

Apache - Adding More Services

Table of Contents

6.1. Forward

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 because its a feature of many clusters and relatively simple to configure.

6.2. Installation

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
```

```
Loaded plugins: langpacks, presto, refresh-packagekit
fedora/metalink                                               | 2.6 kB     00:00
updates/metalink                                              | 3.2 kB     00:00
updates-testing/metalink                                      |  41 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.2.22-3.fc17 will be installed
--> Processing Dependency: httpd-tools = 2.2.22-3.fc17 for package:
 httpd-2.2.22-3.fc17.x86_64
--> Processing Dependency: apr-util-ldap for package: httpd-2.2.22-3.fc17.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.2.22-3.fc17.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.2.22-3.fc17.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.4.6-1.fc17 will be installed
---> Package apr-util.x86_64 0:1.4.1-2.fc17 will be installed
---> Package apr-util-ldap.x86_64 0:1.4.1-2.fc17 will be installed
---> Package httpd-tools.x86_64 0:2.2.22-3.fc17 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====================================================================================
 Package                Arch         Version                Repository          Size
=====================================================================================
Installing:
 httpd                  x86_64       2.2.22-3.fc17          updates-testing      823 k
 wget                   x86_64       1.13.4-2.fc17          fedora               495 k
```

```
Installing for dependencies:
 apr                x86_64       1.4.6-1.fc17           fedora                 99 k
 apr-util           x86_64       1.4.1-2.fc17           fedora                 78 k
 apr-util-ldap      x86_64       1.4.1-2.fc17           fedora                 17 k
 httpd-tools        x86_64       2.2.22-3.fc17          updates-testing        74 k

Transaction Summary
=====================================================================================
Install  1 Package (+4 Dependent packages)

Total download size: 1.1 M
Installed size: 3.5 M
Downloading Packages:
(1/6): apr-1.4.6-1.fc17.x86_64.rpm                           |  99 kB     00:00
(2/6): apr-util-1.4.1-2.fc17.x86_64.rpm                      |  78 kB     00:00
(3/6): apr-util-ldap-1.4.1-2.fc17.x86_64.rpm                 |  17 kB     00:00
(4/6): httpd-2.2.22-3.fc17.x86_64.rpm                        | 823 kB     00:01
(5/6): httpd-tools-2.2.22-3.fc17.x86_64.rpm                  |  74 kB     00:00
(6/6): wget-1.13.4-2.fc17.x86_64.rpm                         | 495 kB     00:01
-------------------------------------------------------------------------------------
Total                                             238 kB/s | 1.1 MB     00:04
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : apr-1.4.6-1.fc17.x86_64                                          1/6
  Installing : apr-util-1.4.1-2.fc17.x86_64                                     2/6
  Installing : apr-util-ldap-1.4.1-2.fc17.x86_64                                3/6
  Installing : httpd-tools-2.2.22-3.fc17.x86_64                                 4/6
  Installing : httpd-2.2.22-3.fc17.x86_64                                       5/6
  Installing : wget-1.13.4-2.fc17.x86_64                                        6/6
  Verifying  : apr-util-ldap-1.4.1-2.fc17.x86_64                                1/6
  Verifying  : httpd-tools-2.2.22-3.fc17.x86_64                                 2/6
  Verifying  : apr-util-1.4.1-2.fc17.x86_64                                     3/6
  Verifying  : apr-1.4.6-1.fc17.x86_64                                          4/6
  Verifying  : httpd-2.2.22-3.fc17.x86_64                                       5/6
  Verifying  : wget-1.13.4-2.fc17.x86_64                                        6/6

Installed:
  httpd.x86_64 0:2.2.22-3.fc17              wget.x86_64 0:1.13.4-2.fc17

Dependency Installed:
  apr.x86_64 0:1.4.6-1.fc17                 apr-util.x86_64 0:1.4.1-2.fc17
  apr-util-ldap.x86_64 0:1.4.1-2.fc17       httpd-tools.x86_64 0:2.2.22-3.fc17

Complete!
```

6.3. Preparation

First we need to create a page for Apache to serve up. On Fedora the default Apache docroot is /var/www/html, so we'll create an index file there.

```
# cat <<-END >/var/www/html/index.html
 <html>
 <body>My Test Site - pcmk-1</body>
 </html>
END
```

For the moment, we will simplify things by serving up only a static site and manually sync the data between the two nodes. So run the command again on pcmk-2.

```
[root@pcmk-2 ~]# cat <<-END >/var/www/html/index.html
 <html>
```

```
  <body>My Test Site - pcmk-2</body>
</html>
END
```

6.4. 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. Look for the following in */etc/httpd/conf/httpd.conf* and make sure it is not disabled or commented out:

```
<Location /server-status>
   SetHandler server-status
   Order deny,allow
   Deny from all
   Allow from 127.0.0.1
</Location>
```

6.5. Update the Configuration

At this point, Apache is ready to go, all that needs to be done is to add it to the cluster. Lets call the resource WebSite. We need to use an OCF script called apache in the heartbeat namespace [1] , the 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.

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

```
# pcs resource op defaults timeout=240s
# pcs resource op defaults
timeout: 240s
```

After a short delay, we should see the cluster start apache

```
# pcs status

Last updated: Fri Sep 14 10:51:27 2012
Last change: Fri Sep 14 10:50:46 2012 via crm_attribute on pcmk-1
Stack: corosync
Current DC: pcmk-2 (2) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
2 Resources configured.

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

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

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

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

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 if this is the problem by running:

```
wget http://127.0.0.1/server-status
```

If you see **`Connection refused`** in the output, then this is indeed the problem. Check to ensure that **`Allow from 127.0.0.1`** is present for the **`<Location /server-status>`** block.

6.6. Ensuring 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 **`WebSite`** needs to be placed on the same machine as **`ClusterIP`**, this implies that we must know the location of **`ClusterIP`** before choosing a location for **`WebSite`**.

```
# pcs constraint colocation add WebSite ClusterIP INFINITY
# pcs constraint
Location Constraints:
Ordering Constraints:
Colocation Constraints:
  WebSite with ClusterIP
# pcs status
```

```
Last updated: Fri Sep 14 11:00:44 2012
Last change: Fri Sep 14 11:00:25 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-2 (2) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
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
```

6.7. Controlling Resource Start/Stop Ordering

When Apache starts, it binds to the available IP addresses. It doesn't know about any addresses we add afterwards, so not only do they need to run on the same node, but we need to make sure ClusterIP is already active before we start WebSite. We do this by adding an ordering constraint.

By default all order constraints are mandatory constraints unless otherwise configured. This means that the recovery of ClusterIP will also trigger the recovery of WebSite.

```
# pcs constraint order ClusterIP then WebSite
Adding ClusterIP WebSite (kind: Mandatory) (Options: first-action=start then-action=start)
# pcs constraint
Location Constraints:
Ordering Constraints:
  start ClusterIP then start WebSite
Colocation Constraints:
  WebSite with ClusterIP
```

6.8. Specifying a Preferred Location

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 it makes sense to host the resources there if it is available. 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. The score here indicates how badly we'd like the resource to run somewhere.

```
# pcs constraint location WebSite prefers pcmk-1=50
# pcs constraint
Location Constraints:
  Resource: WebSite
    Enabled on: pcmk-1 (score:50)
Ordering Constraints:
  start ClusterIP then start WebSite
Colocation Constraints:
  WebSite with ClusterIP
# pcs status
Last updated: Fri Sep 14 11:06:37 2012
Last change: Fri Sep 14 11:06:26 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-2 (2) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
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
```

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

Even though we now prefer pcmk-1 over pcmk-2, 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

```
# 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:
```

6.9. Manually Moving Resources Around the Cluster

There are always times when an administrator needs to override the cluster and force resources to move to a specific location. By updating our previous location constraint with a score of INFINITY, WebSite will be forced to move to pcmk-1.

```
# pcs constraint location WebSite prefers pcmk-1=INFINITY
# pcs constraint --full
Location Constraints:
  Resource: WebSite
    Enabled on: pcmk-1 (score:INFINITY) (id:location-WebSite-pcmk-1-INFINITY)
Ordering Constraints:
  start ClusterIP then start WebSite (Mandatory) (id:order-ClusterIP-WebSite-mandatory)
Colocation Constraints:
  WebSite with ClusterIP (INFINITY) (id:colocation-WebSite-ClusterIP-INFINITY)
# pcs status

Last updated: Fri Sep 14 11:16:26 2012
Last change: Fri Sep 14 11:16:18 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-2 (2) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
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
```

6.9.1. Giving Control Back to the Cluster

Once we've finished whatever activity that required us to move the resources to pcmk-1, in our case nothing, we can then allow the cluster to resume normal operation with the unmove command. Since we previously configured a default stickiness, the resources will remain on pcmk-1.

```
# pcs constraint all
Location Constraints:
  Resource: WebSite
    Enabled on: pcmk-1 (score:INFINITY) (id:location-WebSite-pcmk-1-INFINITY)
Ordering Constraints:
  start ClusterIP then start WebSite (Mandatory) (id:order-ClusterIP-WebSite-mandatory)
Colocation Constraints:
  WebSite with ClusterIP (INFINITY) (id:colocation-WebSite-ClusterIP-INFINITY)
# pcs constraint remove location-WebSite-pcmk-1-INFINITY
# pcs constraint
Location Constraints:
Ordering Constraints:
  start ClusterIP then start WebSite
Colocation Constraints:
  WebSite with ClusterIP
```

Note that the 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.

```
# pcs status

Last updated: Fri Sep 14 11:57:12 2012
Last change: Fri Sep 14 11:57:03 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-2 (2) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
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
```

Chapter 7.

Replicated Storage with DRBD

Table of Contents

7.1. Background

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. See *http://www.drbd.org/* for more details.

7.2. Install the DRBD Packages

Since its inclusion in the upstream 2.6.33 kernel, everything needed to use DRBD has shiped with Fedora since version 13. All you need to do is install it:

```
# yum install -y drbd-pacemaker drbd-udev
```

```
Loaded plugins: langpacks, presto, refresh-packagekit
Resolving Dependencies
--> Running transaction check
---> Package drbd-pacemaker.x86_64 0:8.3.11-5.fc17 will be installed
--> Processing Dependency: drbd-utils = 8.3.11-5.fc17 for package: drbd-
pacemaker-8.3.11-5.fc17.x86_64
---> Package drbd-udev.x86_64 0:8.3.11-5.fc17 will be installed
--> Running transaction check
---> Package drbd-utils.x86_64 0:8.3.11-5.fc17 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====================================================================================
 Package                Arch         Version                Repository          Size
=====================================================================================
Installing:
 drbd-pacemaker         x86_64       8.3.11-5.fc17          updates-testing       22 k
 drbd-udev              x86_64       8.3.11-5.fc17          updates-testing      6.4 k
Installing for dependencies:
 drbd-utils             x86_64       8.3.11-5.fc17          updates-testing      183 k

Transaction Summary
=====================================================================================
Install  2 Packages (+1 Dependent package)

Total download size: 212 k
```

```
Installed size: 473 k
Downloading Packages:
(1/3): drbd-pacemaker-8.3.11-5.fc17.x86_64.rpm                  |  22 kB     00:00
(2/3): drbd-udev-8.3.11-5.fc17.x86_64.rpm                       | 6.4 kB     00:00
(3/3): drbd-utils-8.3.11-5.fc17.x86_64.rpm                      | 183 kB     00:00
--------------------------------------------------------------------------------------
Total                                                  293 kB/s | 212 kB     00:00
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : drbd-utils-8.3.11-5.fc17.x86_64                                    1/3
  Installing : drbd-pacemaker-8.3.11-5.fc17.x86_64                                2/3
  Installing : drbd-udev-8.3.11-5.fc17.x86_64                                     3/3
  Verifying  : drbd-pacemaker-8.3.11-5.fc17.x86_64                                1/3
  Verifying  : drbd-udev-8.3.11-5.fc17.x86_64                                     2/3
  Verifying  : drbd-utils-8.3.11-5.fc17.x86_64                                    3/3

Installed:
  drbd-pacemaker.x86_64 0:8.3.11-5.fc17        drbd-udev.x86_64 0:8.3.11-5.fc17

Dependency Installed:
  drbd-utils.x86_64 0:8.3.11-5.fc17

Complete!
```

7.3. Configure DRBD

Before we configure DRBD, we need to set aside some disk for it to use.

7.3.1. Create A Partition for DRBD

If you have more than 1Gb free, feel free to use it. For this guide however, 1Gb is plenty of space for a single html file and sufficient for later holding the GFS2 metadata.

```
# vgdisplay | grep -e Name -e Free
  VG Name               vg_pcmk1
  Free  PE / Size       31 / 992.00 MiB
# lvs
  LV        VG          Attr     LSize   Pool Origin Data%  Move Log Copy%  Convert
  lv_root   vg_pcmk1 -wi-ao--   8.56g
  lv_swap   vg_pcmk1 -wi-ao-- 960.00m
# lvcreate -n drbd-demo -L 1G vg_pcmk1
Logical volume "drbd-demo" created
# lvs
  LV        VG          Attr     LSize   Pool Origin Data%  Move Log Copy%  Convert
  drbd-demo vg_pcmk1 -wi-a--- 1.00G
  lv_root   vg_pcmk1 -wi-ao--   8.56g
  lv_swap   vg_pcmk1 -wi-ao-- 960.00m
```

Repeat this on the second node, be sure to use the same size partition.

```
# ssh pcmk-2 -- lvs
LV   VG    Attr  LSize  Origin Snap% Move Log Copy% Convert
  lv_root   vg_pcmk1 -wi-ao--   8.56g
  lv_swap   vg_pcmk1 -wi-ao-- 960.00m
# ssh pcmk-2 -- lvcreate -n drbd-demo -L 1G vg_pcmk1
Logical volume "drbd-demo" created
# ssh pcmk-2 -- lvs
LV    VG    Attr  LSize  Origin Snap% Move Log Copy% Convert
  drbd-demo vg_pcmk1 -wi-a--- 1.00G
  lv_root   vg_pcmk1 -wi-ao--   8.56g
  lv_swap   vg_pcmk1 -wi-ao-- 960.00m
```

7.3.2. Write the DRBD Config

There is no series of commands for building a DRBD configuration, so simply copy the configuration below to /etc/drbd.conf

Detailed information on the directives used in this configuration (and other alternatives) is available from *http://www.drbd.org/users-guide/ch-configure.html*

Warning

Be sure to use the names and addresses of your nodes if they differ from the ones used in this guide.

```
global {
 usage-count yes;
}
common {
 protocol C;
}
resource wwwdata {
 meta-disk internal;
 device  /dev/drbd1;
 syncer {
  verify-alg sha1;
 }
 net {
  allow-two-primaries;
 }
 on pcmk-1 {
  disk   /dev/vg_pcmk1/drbd-demo;
  address  192.168.122.101:7789;
 }
 on pcmk-2 {
  disk   /dev/vg_pcmk1/drbd-demo;
  address  192.168.122.102:7789;
 }
}
```

Note

TODO: Explain the reason for the allow-two-primaries option

7.3.3. Initialize and Load DRBD

With the configuration in place, we can now perform the DRBD initialization

```
# drbdadm create-md wwwdata
Writing meta data...
initializing activity log
NOT initialized bitmap
New drbd meta data block successfully created.
```

```
success
```

Now load the DRBD kernel module and confirm that everything is sane

```
# modprobe drbd
# drbdadm up wwwdata
# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: 0D2B62DEDB020A425130935

 1: cs:Connected ro:Secondary/Secondary ds:Inconsistent/Inconsistent C r-----
    ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:1015740
```

Repeat on the second node

```
# ssh pcmk-2 -- drbdadm --force create-md wwwdata
Writing meta data...
initializing activity log
NOT initialized bitmap
New drbd meta data block successfully created.
success
# ssh pcmk-2 -- modprobe drbd
WARNING: Deprecated config file /etc/modprobe.conf, all config files belong into /etc/
modprobe.d/.
# ssh pcmk-2 -- drbdadm up wwwdata
# ssh pcmk-2 -- cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: 0D2B62DEDB020A425130935

 1: cs:Connected ro:Secondary/Secondary ds:Inconsistent/Inconsistent C r-----
    ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:1015740
```

Now we need to tell DRBD which set of data to use. Since both sides contain garbage, we can run the following on pcmk-1:

```
# drbdadm -- --overwrite-data-of-peer primary wwwdata
# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: 0D2B62DEDB020A425130935

 1: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-----
    ns:8064 nr:0 dw:0 dr:8728 al:0 bm:0 lo:0 pe:1 ua:0 ap:0 ep:1 wo:f oos:1007804
        [>....................] sync'ed:  0.9% (1007804/1015740)K
        finish: 0:12:35 speed: 1,320 (1,320) K/sec
```

After a while, the sync should finish and you'll see:

```
# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: 0D2B62DEDB020A425130935

 1: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
    ns:1015740 nr:0 dw:0 dr:1016404 al:0 bm:62 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

pcmk-1 is now in the Primary state which allows it to be written to. Which means it's a good point at which to create a filesystem and populate it with some data to serve up via our WebSite resource.

7.3.4. Populate DRBD with Data

```
# mkfs.ext4 /dev/drbd1
```

```
mke2fs 1.42 (29-Nov-2011)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
63488 inodes, 253935 blocks
12696 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=260046848
8 block groups
32768 blocks per group, 32768 fragments per group
7936 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

Now mount the newly created filesystem so we can create our index file

```
# mount /dev/drbd1 /mnt/
# cat <<-END >/mnt/index.html
 <html>
  <body>My Test Site - drbd</body>
 </html>
END
# umount /dev/drbd1
```

7.4. Configure the Cluster for DRBD

One handy feature pcs has is the ability to queue up several changes into a file and commit those changes atomically. To do this, start by populating the file with the current raw xml config from the cib. This can be done using the following command.

```
# pcs cluster cib drbd_cfg
```

Now using the pcs -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 on.

```
# pcs -f drbd_cfg resource create WebData ocf:linbit:drbd \
         drbd_resource=wwwdata op monitor interval=60s
# pcs -f drbd_cfg resource master WebDataClone WebData \
         master-max=1 master-node-max=1 clone-max=2 clone-node-max=1 \
         notify=true
```

```
# pcs -f drbd_cfg resource show
 ClusterIP      (ocf::heartbeat:IPaddr2) Started
 WebSite        (ocf::heartbeat:apache) Started
 Master/Slave Set: WebDataClone [WebData]
     Stopped: [ WebData:0 WebData:1 ]
```

After you are satisfied with all the changes, you can commit all the changes at once by pushing the drbd_cfg file into the live cib.

```
# pcs cluster cib-push drbd_cfg
```

```
CIB updated

# pcs status

Last updated: Fri Sep 14 12:19:49 2012
Last change: Fri Sep 14 12:19:13 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-2 (2) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
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 ]
```

Note

TODO: Include details on adding a second DRBD resource

Now that DRBD is functioning we can configure a Filesystem resource to use it. In addition to the filesystem's definition, 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 though. 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 to use.

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

```
# pcs cluster cib fs_cfg
# pcs -f fs_cfg resource create WebFS Filesystem \
         device="/dev/drbd/by-res/wwwdata" directory="/var/www/html" \
         fstype="ext4"
```

```
# pcs -f fs_cfg constraint colocation add WebFS WebDataClone INFINITY with-rsc-role=Master
# pcs -f fs_cfg constraint order promote WebDataClone then start WebFS
Adding WebDataClone WebFS (kind: Mandatory) (Options: first-action=promote then-action=start)
```

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

```
# pcs -f fs_cfg constraint colocation add WebSite WebFS INFINITY
# pcs -f fs_cfg constraint order WebFS then WebSite
```

Now review the updated configuration.

```
# pcs -f fs_cfg constraint
Location Constraints:
Ordering Constraints:
  start ClusterIP then start WebSite
  WebFS then WebSite
  promote WebDataClone then start WebFS
Colocation Constraints:
  WebSite with ClusterIP
  WebFS with WebDataClone (with-rsc-role:Master)
  WebSite with WebFS

# pcs -f fs_cfg resource show
 ClusterIP      (ocf::heartbeat:IPaddr2) Started
 WebSite        (ocf::heartbeat:apache) Started
 Master/Slave Set: WebDataClone [WebData]
     Masters: [ pcmk-1 ]
     Slaves: [ pcmk-2 ]
 WebFS  (ocf::heartbeat:Filesystem) Stopped
```

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

```
# pcs cluster cib-push fs_cfg
CIB updated
# pcs status
 Last updated: Fri Aug 10 12:47:01 2012

 Last change: Fri Aug 10 12:46:55 2012 via cibadmin on pcmk-1
 Stack: corosync
 Current DC: pcmk-1 (1) - partition with quorum
 Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
 2 Nodes configured, unknown expected votes
 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
```

7.4.1. Testing Migration

We could shut down the active node again, but another way to safely simulate recovery is to put the node into what is called "standby mode". Nodes in this state tell the cluster that they are not allowed to run resources. Any resources found active there will be moved elsewhere. This feature can be particularly useful when updating the resources' packages.

Put the local node into standby mode and observe the cluster move all the resources to the other node. Note also that the node’s status will change to indicate that it can no longer host resources.

```
# pcs cluster standby pcmk-1
# pcs status

Last updated: Fri Sep 14 12:41:12 2012
Last change: Fri Sep 14 12:41:08 2012 via crm_attribute on pcmk-1
Stack: corosync
Current DC: pcmk-1 (1) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
```

```
2 Nodes configured, unknown expected votes
5 Resources configured.

Node pcmk-1 (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: [ WebData:1 ]
WebFS   (ocf::heartbeat:Filesystem):    Started pcmk-2
```

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.

```
# pcs cluster unstandby pcmk-1
# pcs status

Last updated: Fri Sep 14 12:43:02 2012
Last change: Fri Sep 14 12:42:57 2012 via crm_attribute on pcmk-1
Stack: corosync
Current DC: pcmk-1 (1) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
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
```

Notice that our resource stickiness settings prevent the services from migrating back to pcmk-1.

Conversion to Active/Active

Table of Contents

8.1. Requirements

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, however since DRBD supports multiple Primaries, we can also use that.

The only hitch is that we need to use a cluster-aware filesystem. The one we used earlier with DRBD, ext4, is not one of those. Both OCFS2 and GFS2 are supported, however here we will use GFS2 which comes with Fedora 17.

8.1.1. Installing the required Software

```
# yum install -y gfs2-utils dlm kernel-modules-extra
```

```
Loaded plugins: langpacks, presto, refresh-packagekit
Resolving Dependencies
--> Running transaction check
---> Package dlm.x86_64 0:3.99.4-1.fc17 will be installed
---> Package gfs2-utils.x86_64 0:3.1.4-3.fc17 will be installed
---> Package kernel-modules-extra.x86_64 0:3.4.4-3.fc17 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch       Version          Repository           Size
================================================================================
Installing:
 dlm                    x86_64     3.99.4-1.fc17    updates               83 k
 gfs2-utils             x86_64     3.1.4-3.fc17     fedora               214 k
 kernel-modules-extra   x86_64     3.4.4-3.fc17     updates              1.7 M

Transaction Summary
================================================================================
Install  3 Packages

Total download size: 1.9 M
Installed size: 7.7 M
Downloading Packages:
(1/3): dlm-3.99.4-1.fc17.x86_64.rpm                          |  83 kB     00:00
(2/3): gfs2-utils-3.1.4-3.fc17.x86_64.rpm                    | 214 kB     00:00
(3/3): kernel-modules-extra-3.4.4-3.fc17.x86_64.rpm          | 1.7 MB     00:01
```

```
--------------------------------------------------------------------------------
Total                                              615 kB/s | 1.9 MB     00:03
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : kernel-modules-extra-3.4.4-3.fc17.x86_64                 1/3
  Installing : gfs2-utils-3.1.4-3.fc17.x86_64                           2/3
  Installing : dlm-3.99.4-1.fc17.x86_64                                 3/3
  Verifying  : dlm-3.99.4-1.fc17.x86_64                                 1/3
  Verifying  : gfs2-utils-3.1.4-3.fc17.x86_64                           2/3
  Verifying  : kernel-modules-extra-3.4.4-3.fc17.x86_64                 3/3

Installed:
  dlm.x86_64 0:3.99.4-1.fc17
  gfs2-utils.x86_64 0:3.1.4-3.fc17
  kernel-modules-extra.x86_64 0:3.4.4-3.fc17

Complete!
```

8.2. Create a GFS2 Filesystem

8.2.1. Preparation

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.

```
# pcs resource disable WebFS
# pcs resource
 ClusterIP      (ocf::heartbeat:IPaddr2) Started
 WebSite        (ocf::heartbeat:apache) Stopped
 Master/Slave Set: WebDataClone [WebData]
     Masters: [ pcmk-2 ]
     Slaves: [ pcmk-1 ]
 WebFS  (ocf::heartbeat:Filesystem) Stopped
```

Note

Note that both Apache and WebFS have been stopped.

8.2.2. Create and Populate an GFS2 Partition

Now that the cluster stack and integration pieces are running smoothly, we can create an GFS2 partition.

Warning

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

We need to specify a number of additional parameters when creating a GFS2 partition.

First we must use the -p option to specify that we want to use the the Kernel's DLM. Next we use -j to indicate that it should reserve enough space for two journals (one per node accessing the filesystem).

Lastly, we use -t to specify the lock table name. The format for this field is **`clustername:fsname`**. For the **`fsname`**, we need to use the same value as specified in *corosync.conf* for **`cluster_name`**. If you setup corosync with the same cluster name we used in this tutorial, cluster name will be *mycluster*. If you are unsure what your cluster name is, open up /etc/corosync/corosync.conf, or execute the command *pcs cluster corosync pcmk-1* to view the corosync config. The cluster name will be in the **`totem`** block.

Important

We must run the next command on whichever node last had */dev/drbd* mounted. Otherwise you will receive the message:

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

```
# ssh pcmk-2 -- mkfs.gfs2 -p lock_dlm -j 2 -t mycluster:web /dev/drbd1
This will destroy any data on /dev/drbd1.
It appears to contain: Linux rev 1.0 ext4 filesystem data, UUID=dc45fff3-c47a-4db2-96f7-
a8049a323fe4 (extents) (large files) (huge files)
Are you sure you want to proceed? [y/n]y
Device:                    /dev/drbd1
Blocksize:                 4096
Device Size                0.97 GB (253935 blocks)
Filesystem Size:           0.97 GB (253932 blocks)
Journals:                  2
Resource Groups:           4
Locking Protocol:          "lock_dlm"
Lock Table:                "mycluster"
UUID:                      ed293a02-9eee-3fa3-ed1c-435ef1fd0116
```

```
# pcs cluster cib dlm_cfg
# pcs -f dlm_cfg resource create dlm ocf:pacemaker:controld op monitor interval=60s
# pcs -f dlm_cfg resource clone dlm clone-max=2 clone-node-max=1
# pcs -f dlm_cfg resource show
 ClusterIP      (ocf::heartbeat:IPaddr2) Started
 WebSite        (ocf::heartbeat:apache) Stopped
 Master/Slave Set: WebDataClone [WebData]
     Masters: [ pcmk-2 ]
     Slaves: [ pcmk-1 ]
 WebFS  (ocf::heartbeat:Filesystem) Stopped
 Clone Set: dlm-clone [dlm]
     Stopped: [ dlm:0 dlm:1 ]
# pcs cluster cib-push dlm_cfg
CIB updated
# pcs status

Last updated: Fri Sep 14 12:54:50 2012
Last change: Fri Sep 14 12:54:43 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-1 (1) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
7 Resources configured.
```

```
Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

 ClusterIP      (ocf::heartbeat:IPaddr2):       Started pcmk-2
 WebSite        (ocf::heartbeat:apache):        Stopped
 Master/Slave Set: WebDataClone [WebData]
     Masters: [ pcmk-2 ]
     Slaves: [ pcmk-1 ]
 WebFS  (ocf::heartbeat:Filesystem):    Stopped
 Clone Set: dlm-clone [dlm]
     Started: [ pcmk-1 pcmk-2 ]
```

Then (re)populate the new filesystem with data (web pages). For now we'll create another variation on our home page.

```
# mount /dev/drbd1 /mnt/
# cat <<-END >/mnt/index.html
<html>
<body>My Test Site - GFS2</body>
</html>
END
# umount /dev/drbd1
# drbdadm verify wwwdata
```

8.3. Reconfigure the Cluster for GFS2

With the WebFS resource stopped, lets update the configuration.

```
# pcs resource show WebFS
Resource: WebFS
  device: /dev/drbd/by-res/wwwdata
  directory: /var/www/html
  fstype: ext4
  target-role: Stopped
```

The fstype option needs to be updated to gfs2 instead of ext4.

```
# pcs resource update WebFS fstype=gfs2
# pcs resource show WebFS
Resource: WebFS
  device: /dev/drbd/by-res/wwwdata
  directory: /var/www/html
  fstype: gfs2
  target-role: Stopped
CIB updated
```

8.4. Reconfigure Pacemaker for Active/Active

Almost everything is in place. Recent versions of DRBD are capable of operating in Primary/Primary mode and the filesystem we're using is cluster aware. All we need to do now is reconfigure the cluster to take advantage of this.

This will involve a number of changes, so we'll want work with a local cib file.

```
# pcs cluster cib active_cfg
```

There's no point making the services active on both locations if we can't reach them, so lets first clone the IP address. Cloned IPaddr2 resources use an iptables rule to ensure that each request only gets processed by one of the two clone instances. The additional meta options tell the cluster how many instances of the clone we want (one "request bucket" for each node) and that if all other nodes fail, then the remaining node should hold all of them. Otherwise the requests would be simply discarded.

```
# pcs -f active_cfg resource clone ClusterIP \
     globally-unique=true clone-max=2 clone-node-max=2
```

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

```
# pcs -f active_cfg constraint
Location Constraints:
Ordering Constraints:
  start ClusterIP-clone then start WebSite
  WebFS then WebSite
  promote WebDataClone then start WebFS
Colocation Constraints:
  WebSite with ClusterIP-clone
  WebFS with WebDataClone (with-rsc-role:Master)
  WebSite with WebFS
```

Now we must tell the ClusterIP how to decide which requests are processed by which hosts. To do this we must specify the clusterip_hash parameter.

```
# pcs -f active_cfg resource update ClusterIP clusterip_hash=sourceip
```

Next we need to convert the filesystem and Apache resources into clones.

Notice how pcs automatically updates the relevant constraints again.

```
# pcs -f active_cfg resource clone WebFS
# pcs -f active_cfg resource clone WebSite
# pcs -f active_cfg constraint
Location Constraints:
Ordering Constraints:
  start ClusterIP-clone then start WebSite-clone
  WebFS-clone then WebSite-clone
  promote WebDataClone then start WebFS-clone
Colocation Constraints:
  WebSite-clone with ClusterIP-clone
  WebFS-clone with WebDataClone (with-rsc-role:Master)
  WebSite-clone with WebFS-clone
```

The last step is to tell the cluster that it is now allowed to promote both instances to be Primary (aka. Master).

```
# pcs -f active_cfg resource update WebDataClone master-max=2
```

Review the configuration before uploading it to the cluster, quitting the shell and watching the cluster's response

```
# pcs cluster cib-push active_cfg
# pcs resource enable WebFS
```

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

```
# pcs resource
 Master/Slave Set: WebDataClone [WebData]
     Masters: [ pcmk-2 pcmk-1 ]
 Clone Set: dlm-clone [dlm]
     Started: [ pcmk-2 pcmk-1 ]
 Clone Set: ClusterIP-clone [ClusterIP] (unique)
     ClusterIP:0        (ocf::heartbeat:IPaddr2) Started
     ClusterIP:1        (ocf::heartbeat:IPaddr2) Started
 Clone Set: WebFS-clone [WebFS]
     Started: [ pcmk-1 pcmk-2 ]
 Clone Set: WebSite-clone [WebSite]
     Started: [ pcmk-1 pcmk-2 ]
```

8.4.1. Testing Recovery

Note

TODO: Put one node into standby to demonstrate failover

Configure STONITH

Table of Contents

9.1. What Is STONITH

STONITH is an acronym for Shoot-The-Other-Node-In-The-Head and it protects your data from being corrupted by rogue nodes or concurrent access.

Just because a node is unresponsive, this doesn't mean it isn't accessing your data. The only way to be 100% sure that your data is safe, is to use STONITH so we can be certain that the node is truly offline, before allowing the data to be accessed from another node.

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

9.2. What STONITH Device Should You Use

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

The biggest mistake people make in choosing a STONITH device is to use remote power switch (such as many on-board IMPI controllers) that shares power with the node it controls. In such cases, the cluster cannot be sure if the node is really offline, or active and suffering from a network fault.

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

9.3. Configuring STONITH

1. Find the correct driver: **`pcs stonith list`**
2. Find the parameters associated with the device: **`pcs stonith describe <agent name>`**
3. Create a local config to make changes to **`pcs cluster cib stonith_cfg`**
4. Create the fencing resource using **`pcs -f stonith_cfg stonith create <stonith_id> <stonith device type> [stonith device options]`**
5. Set stonith-enable to true. **`pcs -f stonith_cfg property set stonith-enabled=true`**
6. If the device does not know how to fence nodes based on their uname, you may also need to set the special **`pcmk_host_map`** parameter. See **`man stonithd`** for details.
7. If the device does not support the list command, you may also need to set the special **`pcmk_host_list`** and/or **`pcmk_host_check`** parameters. See **`man stonithd`** for details.

8. If the device does not expect the victim to be specified with the port parameter, you may also need to set the special **`pcmk_host_argument`** parameter. See **`man stonithd`** for details.

9. Commit the new configuration. **`pcs cluster cib-push stonith_cfg`**

10. Once the stonith resource is running, you can test it by executing: **`stonith_admin --reboot nodename`**. Although you might want to stop the cluster on that machine first.

9.4. Example

Assuming we have an chassis containing four nodes and an IPMI device active on 10.0.0.1, then we would chose the fence_ipmilan driver in step 2 and obtain the following list of parameters

Obtaining a list of STONITH Parameters

```
# pcs stonith describe fence_ipmilan
Stonith options for: fence_ipmilan
  auth: IPMI Lan Auth type (md5, password, or none)
  ipaddr: IPMI Lan IP to talk to
  passwd: Password (if required) to control power on IPMI device
  passwd_script: Script to retrieve password (if required)
  lanplus: Use Lanplus
  login: Username/Login (if required) to control power on IPMI device
  action: Operation to perform. Valid operations: on, off, reboot, status, list, diag,
 monitor or metadata
  timeout: Timeout (sec) for IPMI operation
  cipher: Ciphersuite to use (same as ipmitool -C parameter)
  method: Method to fence (onoff or cycle)
  power_wait: Wait X seconds after on/off operation
  delay: Wait X seconds before fencing is started
  privlvl: Privilege level on IPMI device
  verbose: Verbose mode
```

from which we would create a STONITH resource fragment that might look like this

Sample STONITH Resource

```
# pcs cluster cib stonith_cfg
# pcs -f stonith_cfg stonith create impi-fencing fence_ipmilan \
      pcmk_host_list="pcmk-1 pcmk-2" ipaddr=10.0.0.1 login=testuser \
      passwd=acd123 op monitor interval=60s
```

```
# pcs -f stonith_cfg stonith
 impi-fencing   (stonith:fence_ipmilan) Stopped
```

And finally, since we disabled it earlier, we need to re-enable STONITH. At this point we should have the following configuration.

```
# pcs -f stonith_cfg property set stonith-enabled=true
# pcs -f stonith_cfg property
dc-version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
cluster-infrastructure: corosync
no-quorum-policy: ignore
stonith-enabled: true
```

Now push the configuration into the cluster.

```
# pcs cluster cib-push stonith_cfg
```

Appendix A. Configuration Recap

Table of Contents

A.1. Final Cluster Configuration

```
# pcs resource
 Master/Slave Set: WebDataClone [WebData]
     Masters: [ pcmk-2 pcmk-1 ]
 Clone Set: dlm-clone [dlm]
     Started: [ pcmk-2 pcmk-1 ]
 Clone Set: ClusterIP-clone [ClusterIP] (unique)
     ClusterIP:0        (ocf::heartbeat:IPaddr2) Started
     ClusterIP:1        (ocf::heartbeat:IPaddr2) Started
 Clone Set: WebFS-clone [WebFS]
     Started: [ pcmk-1 pcmk-2 ]
 Clone Set: WebSite-clone [WebSite]
     Started: [ pcmk-1 pcmk-2 ]
# pcs resource defaults
resource-stickiness: 100
# pcs resource op defaults
timeout: 240s
# pcs stonith
 impi-fencing   (stonith:fence_ipmilan) Started
# pcs property
dc-version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
cluster-infrastructure: corosync
no-quorum-policy: ignore
stonith-enabled: true
# pcs constraint
Location Constraints:
Ordering Constraints:
  ClusterIP-clone then WebSite-clone
  WebDataClone then WebSite-clone
  WebFS-clone then WebSite-clone
Colocation Constraints:
  WebSite-clone with ClusterIP-clone
  WebFS-clone with WebDataClone (with-rsc-role:Master)
  WebSite-clone with WebFS-clone
#
# pcs status

Last updated: Fri Sep 14 13:45:34 2012
Last change: Fri Sep 14 13:43:13 2012 via cibadmin on pcmk-1
Stack: corosync
Current DC: pcmk-1 (1) - partition with quorum
Version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
2 Nodes configured, unknown expected votes
```

```
11 Resources configured.

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

 Master/Slave Set: WebDataClone [WebData]
     Masters: [ pcmk-2 pcmk-1 ]
 Clone Set: dlm-clone [dlm]
     Started: [ pcmk-1 pcmk-2 ]
 Clone Set: ClusterIP-clone [ClusterIP] (unique)
     ClusterIP:0        (ocf::heartbeat:IPaddr2):       Started pcmk-1
     ClusterIP:1        (ocf::heartbeat:IPaddr2):       Started pcmk-2
 Clone Set: WebFS-clone [WebFS]
     Started: [ pcmk-1 pcmk-2 ]
 Clone Set: WebSite-clone [WebSite]
     Started: [ pcmk-1 pcmk-2 ]
 impi-fencing   (stonith:fence_ipmilan):        Started
```

In xml it should look similar to this.

```
<cib admin_epoch="0" cib-last-written="Fri Sep 14 13:43:13 2012" crm_feature_set="3.0.6"
 dc-uuid="1" epoch="47" have-quorum="1" num_updates="50" update-client="cibadmin" update-
origin="pcmk-1" validate-with="pacemaker-1.2">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair id="cib-bootstrap-options-dc-version" name="dc-version"
 value="1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0"/>
        <nvpair id="cib-bootstrap-options-cluster-infrastructure" name="cluster-
infrastructure" value="corosync"/>
        <nvpair id="cib-bootstrap-options-no-quorum-policy" name="no-quorum-policy"
 value="ignore"/>
        <nvpair id="cib-bootstrap-options-stonith-enabled" name="stonith-enabled"
 value="true"/>
      </cluster_property_set>
    </crm_config>
    <nodes>
      <node id="1" type="normal" uname="pcmk-1"/>
      <node id="2" type="normal" uname="pcmk-2"/>
    </nodes>
    <resources>
      <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_resource"
 value="wwwdata"/>
          </instance_attributes>
          <operations>
            <op id="WebData-interval-60s" interval="60s" name="monitor"/>
          </operations>
        </primitive>
        <meta_attributes id="WebDataClone-meta_attributes">
          <nvpair id="WebDataClone-meta_attributes-master-node-max" name="master-node-max"
 value="1"/>
          <nvpair id="WebDataClone-meta_attributes-clone-max" name="clone-max" value="2"/>
          <nvpair id="WebDataClone-meta_attributes-clone-node-max" name="clone-node-max"
 value="1"/>
          <nvpair id="WebDataClone-meta_attributes-notify" name="notify" value="true"/>
          <nvpair id="WebDataClone-meta_attributes-master-max" name="master-max" value="2"/>
        </meta_attributes>
      </master>
      <clone id="dlm-clone">
        <primitive class="ocf" id="dlm" provider="pacemaker" type="controld">
          <instance_attributes id="dlm-instance_attributes"/>
          <operations>
```

```
            <op id="dlm-interval-60s" interval="60s" name="monitor"/>
          </operations>
        </primitive>
        <meta_attributes id="dlm-clone-meta">
          <nvpair id="dlm-clone-max" name="clone-max" value="2"/>
          <nvpair id="dlm-clone-node-max" name="clone-node-max" value="1"/>
        </meta_attributes>
      </clone>
      <clone id="ClusterIP-clone">
        <primitive class="ocf" id="ClusterIP" provider="heartbeat" type="IPaddr2">
          <instance_attributes id="ClusterIP-instance_attributes">
            <nvpair id="ClusterIP-instance_attributes-ip" name="ip" value="192.168.0.120"/>
            <nvpair id="ClusterIP-instance_attributes-cidr_netmask" name="cidr_netmask"
 value="32"/>
            <nvpair id="ClusterIP-instance_attributes-clusterip_hash" name="clusterip_hash"
 value="sourceip"/>
          </instance_attributes>
          <operations>
            <op id="ClusterIP-interval-30s" interval="30s" name="monitor"/>
          </operations>
        </primitive>
        <meta_attributes id="ClusterIP-clone-meta">
          <nvpair id="ClusterIP-globally-unique" name="globally-unique" value="true"/>
          <nvpair id="ClusterIP-clone-max" name="clone-max" value="2"/>
          <nvpair id="ClusterIP-clone-node-max" name="clone-node-max" value="2"/>
        </meta_attributes>
      </clone>
      <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/drbd/by-
res/wwwdata"/>
            <nvpair id="WebFS-instance_attributes-directory" name="directory" value="/var/
www/html"/>
            <nvpair id="WebFS-instance_attributes-fstype" name="fstype" value="gfs2"/>
          </instance_attributes>
          <meta_attributes id="WebFS-meta_attributes"/>
        </primitive>
        <meta_attributes id="WebFS-clone-meta"/>
      </clone>
      <clone id="WebSite-clone">
        <primitive class="ocf" id="WebSite" provider="heartbeat" type="apache">
          <instance_attributes id="WebSite-instance_attributes">
            <nvpair id="WebSite-instance_attributes-configfile" name="configfile" value="/
etc/httpd/conf/httpd.conf"/>
            <nvpair id="WebSite-instance_attributes-statusurl" name="statusurl"
 value="http://localhost/server-status"/>
          </instance_attributes>
          <operations>
            <op id="WebSite-interval-1min" interval="1min" name="monitor"/>
          </operations>
        </primitive>
        <meta_attributes id="WebSite-clone-meta"/>
      </clone>
      <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_host_list"
 value="pcmk-1 pcmk-2"/>
          <nvpair id="impi-fencing-instance_attributes-ipaddr" name="ipaddr"
 value="10.0.0.1"/>
          <nvpair id="impi-fencing-instance_attributes-login" name="login" value="testuser"/>
          <nvpair id="impi-fencing-instance_attributes-passwd" name="passwd" value="acd123"/>
        </instance_attributes>
        <operations>
          <op id="impi-fencing-interval-60s" interval="60s" name="monitor"/>
        </operations>
      </primitive>
```

```
    </resources>
    <constraints>
      <rsc_colocation id="colocation-WebSite-ClusterIP-INFINITY" rsc="WebSite-clone"
 score="INFINITY" with-rsc="ClusterIP-clone"/>
      <rsc_order first="ClusterIP-clone" first-action="start" id="order-ClusterIP-WebSite-
mandatory" then="WebSite-clone" then-action="start"/>
      <rsc_colocation id="colocation-WebFS-WebDataClone-INFINITY" rsc="WebFS-clone"
 score="INFINITY" with-rsc="WebDataClone" with-rsc-role="Master"/>
      <rsc_colocation id="colocation-WebSite-WebFS-INFINITY" rsc="WebSite-clone"
 score="INFINITY" with-rsc="WebFS-clone"/>
      <rsc_order first="WebFS-clone" id="order-WebFS-WebSite-mandatory" then="WebSite-clone"/
>
      <rsc_order first="WebDataClone" first-action="promote" id="order-WebDataClone-WebFS-
mandatory" then="WebFS-clone" then-action="start"/>
    </constraints>
    <rsc_defaults>
      <meta_attributes id="rsc_defaults-options">
        <nvpair id="rsc_defaults-options-resource-stickiness" name="resource-stickiness"
 value="100"/>
      </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>
</cib>
```

A.2. Node List

The list of cluster nodes is automatically populated by the cluster.

```
Pacemaker Nodes:
 Online: [ pcmk-1 pcmk-2  ]
```

A.3. Cluster Options

This is where the cluster automatically stores some information about the cluster

- dc-version - the version (including upstream source-code hash) of Pacemaker used on the DC
- cluster-infrastructure - the cluster infrastructure being used (heartbeat or openais)
- expected-quorum-votes - the maximum number of nodes expected to be part of the cluster

and where the admin can set options that control the way the cluster operates

- stonith-enabled=true - Make use of STONITH
- no-quorum-policy=ignore - Ignore loss of quorum and continue to host resources.

```
# pcs property
dc-version: 1.1.8-1.el7-60a19ed12fdb4d5c6a6b6767f52e5391e447fec0
cluster-infrastructure: corosync
no-quorum-policy: ignore
stonith-enabled: true
```

A.4. Resources

A.4.1. Default Options

Here we configure cluster options that apply to every resource.

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

```
# pcs resource defaults
resource-stickiness: 100
```

A.4.2. Fencing

```
# pcs stonith show
 impi-fencing   (stonith:fence_ipmilan) Started
# pcs stonith show impi-fencing
Resource: impi-fencing
  pcmk_host_list: pcmk-1 pcmk-2
  ipaddr: 10.0.0.1
  login: testuser
  passwd: acd123
```

A.4.3. Service Address

Users of the services provided by the cluster require an unchanging address with which to access it. Additionally, we cloned the address so it will be active on both nodes. An iptables rule (created as part of the resource agent) is used to ensure that each request only gets processed by one of the two clone instances. The additional meta options tell the cluster that we want two instances of the clone (one "request bucket" for each node) and that if one node fails, then the remaining node should hold both.

```
# pcs resource show ClusterIP-clone
Resource: ClusterIP-clone
  ip: 192.168.0.120
  cidr_netmask: 32
  clusterip_hash: sourceip
  globally-unique: true
  clone-max: 2
  clone-node-max: 2
  op monitor interval=30s
```

Note

TODO: The RA should check for globally-unique=true when cloned

A.4.4. DRBD - Shared Storage

Here we define the DRBD service and specify which DRBD resource (from drbd.conf) it should manage. We make it a master/slave resource and, in order to have an active/active setup, allow both instances to be promoted by specifying master-max=2. We also set the notify option so that the cluster will tell DRBD agent when it’s peer changes state.

```
# pcs resource show WebDataClone
Resource: WebDataClone
  drbd_resource: wwwdata
  master-node-max: 1
  clone-max: 2
  clone-node-max: 1
  notify: true
  master-max: 2
  op monitor interval=60s
# pcs constraint ref WebDataClone
Resource: WebDataClone
  colocation-WebFS-WebDataClone-INFINITY
  order-WebDataClone-WebFS-mandatory
```

A.4.5. 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 gfs-control and drbd instances.

```
# pcs resource show WebFS-clone
Resource: WebFS-clone
  device: /dev/drbd/by-res/wwwdata
  directory: /var/www/html
  fstype: gfs2
# pcs constraint ref WebFS-clone
Resource: WebFS-clone
  colocation-WebFS-WebDataClone-INFINITY
  colocation-WebSite-WebFS-INFINITY
  order-WebFS-WebSite-mandatory
  order-WebDataClone-WebFS-mandatory
```

A.4.6. Apache

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

```
# pcs resource show WebSite-clone
Resource: WebSite-clone
  configfile: /etc/httpd/conf/httpd.conf
  statusurl: http://localhost/server-status
  master-max: 2
  op monitor interval=1min
# pcs constraint ref WebSite-clone
Resource: WebSite-clone
  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 using a node list.

```
# Please read the corosync.conf.5 manual page
totem {
version: 2
secauth: off
cluster_name: mycluster
transport: udpu
}

nodelist {
  node {
        ring0_addr: pcmk-1
        nodeid: 1
  }
  node {
        ring0_addr: pcmk-2
        nodeid: 2
  }
}

quorum {
  provider: corosync_votequorum
}

logging {
  to_syslog: yes
}
```

Appendix C. Further Reading

- Project Website *http://www.clusterlabs.org*
- Cluster Commands A comprehensive guide to cluster commands has been written by SuSE and can be found at: *http://www.suse.com/documentation/sle_ha/book_sleha/?page=/documentation/sle_ha/book_sleha/data/book_sleha.html*
- Corosync *http://www.corosync.org*

Appendix D. Revision History

Revision 1-0	**Mon May 17 2010**	**Andrew Beekhof** *andrew@beekhof.net*
Import from Pages.app		
Revision 2-0	**Wed Sep 22 2010**	**Raoul Scarazzini** *rasca@miamammausalinux.org*
Italian translation		
Revision 3-0	**Wed Feb 9 2011**	**Andrew Beekhof** *andrew@beekhof.net*
Updated for Fedora 13		
Revision 4-0	**Wed Oct 5 2011**	**Andrew Beekhof** *andrew@beekhof.net*
Update the GFS2 section to use CMAN		
Revision 5-0	**Fri Feb 10 2012**	**Andrew Beekhof** *andrew@beekhof.net*
Generate docbook content from asciidoc sources		
Revision 6-0	**Tues July 3 2012**	**Andrew Beekhof** *andrew@beekhof.net*
Updated for Fedora 17		
Revision 7-0	**Fri Sept 14 2012**	**David Vossel** *dvossel@redhat.com*
Updated for pcs		

Index