CHAPTER 7. GFS2 FILE SYSTEMS IN A CLUSTER

This section provides administration procedures for configuring GFS2 file systems in a Red Hat high availability cluster.

This section includes:

- A procedure to set up a Pacemaker cluster that includes GFS2 file systems file systems
- A procedure to set up a Pacemaker cluster with an encrypted GFS2 file system
- A procedure to migrate RHEL 7 logical volumes that contain GFS2 file systems to a RHEL 8 cluster

7.1. Configuring a GFS2 file system in a cluster

This procedure is an outline of the steps required to set up a Pacemaker cluster that includes GFS2 file systems. This example creates three GFS2 file systems on three logical volumes.

Prerequisities

- Install and start the cluster software on all nodes and create a basic two-node cluster.
- Configure fencing for the cluster.

For information on creating a Pacemaker cluster and configuring fencing for the cluster, see Creating a Red Hat High-Availability cluster with Pacemaker.

Procedure

1. On both nodes in the cluster, enable the repository for Resilient Storage that corresponds to your system architecture. For example, to enable the Resilient Storage repository for an x86_64 system, you can enter the following `subscription-manager` command:

```
# subscription-manager repos --enable=rhel-8-for-x86_64-resilientstorage-rpms
```

Note that the Resilient Storage repository is a superset of the High Availability repository. If you enable the Resilient Storage repository you do not also need to enable the High Availability repository.

2. On both nodes of the cluster, install the `lvm2-lockd`, `gfs2-utils`, and `dlm` packages. To support these packages, you must be subscribed to the AppStream channel and the Resilient Storage channel.

```
# yum install lvm2-lockd gfs2-utils dlm
```

3. Set the global Pacemaker parameter `no-quorum-policy` to `freeze`.

NOTE

By default, the value of `no-quorum-policy` is set to `stop`, indicating that once quorum is lost, all the resources on the remaining partition will immediately be stopped. Typically this default is the safest and most optimal option, but unlike most resources, GFS2 requires quorum to function. When quorum is lost both the applications using the GFS2 mounts and the GFS2 mount itself cannot be correctly stopped. Any attempts to stop these resources without quorum will fail which will ultimately result in the entire cluster being fenced every time quorum is lost.

To address this situation, set `no-quorum-policy` to `freeze` when GFS2 is in use. This means that when quorum is lost, the remaining partition will do nothing until quorum is regained.

```
# pcs property set no-quorum-policy=freeze
```

4. Set up a `dlm` resource. This is a required dependency for configuring a GFS2 file system in a cluster. This example creates the `dlm` resource as part of a resource group named `locking`.

```
root@z1 ~]# pcs resource create dlm --group locking ocf:pacemaker:controld op monitor interval=30s on-fail=fence
```

5. Clone the `locking` resource group so that the resource group can be active on both nodes of the cluster.

```
[root@z1 ~]# pcs resource clone locking interleave=true
```

6. Set up an `lvmlockd` resource as part of the group `locking`.

```
z1 ~]# pcs resource create lvmlockd --group locking ocf:heartbeat:lvmlockd op monitor interval=30s on-fail=fence
```

7. Check the status of the cluster to ensure that the `locking` resource group has started on both nodes of the cluster.

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

Online: [ z1.example.com (1) z2.example.com (2) ]

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
     Resource Group: locking:0
         dlm    (ocf::pacemaker:controld):      Started z1.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
     Resource Group: locking:1
         dlm    (ocf::pacemaker:controld):      Started z2.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
     Started: [ z1.example.com z2.example.com ]
```

8. On one node of the cluster, create two shared volume groups. One volume group will contain two GFS2 file systems, and the other volume group will contain one GFS2 file system.

   The following command creates the shared volume group `shared_vg1` on `/dev/vdb`.

```
[root@z1 ~]# vgcreate --shared shared_vg1 /dev/vdb
  Physical volume "/dev/vdb" successfully created.
  Volume group "shared_vg1" successfully created
  VG shared_vg1 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

   The following command creates the shared volume group `shared_vg2` on `/dev/vdc`.

```
[root@z1 ~]# vgcreate --shared shared_vg2 /dev/vdc
  Physical volume "/dev/vdc" successfully created.
  Volume group "shared_vg2" successfully created
  VG shared_vg2 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

9. On the second node in the cluster, start the lock manager for each of the shared volume groups.

```
[root@z2 ~]# vgchange --lock-start shared_vg1
  VG shared_vg1 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
[root@z2 ~]# vgchange --lock-start shared_vg2
  VG shared_vg2 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

10. On one node in the cluster, create the shared logical volumes and format the volumes with a GFS2 file system. One journal is required for each node that mounts the file system. Ensure that you create enough journals for each of the nodes in your cluster.

```
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv1 shared_vg1
  Logical volume "shared_lv1" created.
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv2 shared_vg1
  Logical volume "shared_lv2" created.
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv1 shared_vg2
  Logical volume "shared_lv1" created.

[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo1 /dev/shared_vg1/shared_lv1
[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo2 /dev/shared_vg1/shared_lv2
[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo3 /dev/shared_vg2/shared_lv1
```

11. Create an `LVM-activate` resource for each logical volume to automatically activate that logical volume on all nodes.

    i. Create an `LVM-activate` resource named `sharedlv1` for the logical volume `shared_lv1` in volume group `shared_vg1`. This command also creates the resource group `shared_vg1` that includes the resource. In this example, the resource group has the same name as the shared volume group that includes the logical volume.

    ```
    [root@z1 ~]# pcs resource create sharedlv1 --group shared_vg1 ocf:heartbeat:LVM-activate lvname=shared_lv1 vg
    ```

    ii. Create an `LVM-activate` resource named `sharedlv2` for the logical volume `shared_lv2` in volume group `shared_vg1`. This resource will also be part of the resource group `shared_vg1`.

    ```
    [root@z1 ~]# pcs resource create sharedlv2 --group shared_vg1 ocf:heartbeat:LVM-activate lvname=shared_lv2 vg
    ```

    iii. Create an `LVM-activate` resource named `sharedlv3` for the logical volume `shared_lv1` in volume group `shared_vg2`. This command also creates the resource group `shared_vg2` that includes the resource.

    ```
    [root@z1 ~]# pcs resource create sharedlv3 --group shared_vg2 ocf:heartbeat:LVM-activate lvname=shared_lv1 vg
    ```

12. Clone the two new resource groups.

```
[root@z1 ~]# pcs resource clone shared_vg1 interleave=true
[root@z1 ~]# pcs resource clone shared_vg2 interleave=true
```

13. Configure ordering constraints to ensure that the `locking` resource group that includes the `dlm` and `lvmlockd` resources starts first.

```
[root@z1 ~]# pcs constraint order start locking-clone then shared_vg1-clone
Adding locking-clone shared_vg1-clone (kind: Mandatory) (Options: first-action=start then-action=start)
[root@z1 ~]# pcs constraint order start locking-clone then shared_vg2-clone
Adding locking-clone shared_vg2-clone (kind: Mandatory) (Options: first-action=start then-action=start)
```

14. Configure colocation constraints to ensure that the `vg1` and `vg2` resource groups start on the same node as the `locking` resource group.

```
[root@z1 ~]# pcs constraint colocation add shared_vg1-clone with locking-clone
[root@z1 ~]# pcs constraint colocation add shared_vg2-clone with locking-clone
```

15. On both nodes in the cluster, verify that the logical volumes are active. There may be a delay of a few seconds.

```
[root@z1 ~]# lvs
  LV         VG          Attr       LSize
  shared_lv1 shared_vg1  -wi-a----- 5.00g
  shared_lv2 shared_vg1  -wi-a----- 5.00g
  shared_lv1 shared_vg2  -wi-a----- 5.00g

[root@z2 ~]# lvs
  LV         VG          Attr       LSize
  shared_lv1 shared_vg1  -wi-a----- 5.00g
  shared_lv2 shared_vg1  -wi-a----- 5.00g
  shared_lv1 shared_vg2  -wi-a----- 5.00g
```

16. Create a file system resource to automatically mount each GFS2 file system on all nodes.

    You should not add the file system to the `/etc/fstab` file because it will be managed as a Pacemaker cluster resource. Mount options can be specified as part of the resource configuration with `options=*options*`. Run the `pcs resource describe Filesystem` command for full configuration options.

    The following commands create the file system resources. These commands add each resource to the resource group that includes the logical volume resource for that file system.

```
[root@z1 ~]# pcs resource create sharedfs1 --group shared_vg1 ocf:heartbeat:Filesystem device="/dev/shared_vg1/s
[root@z1 ~]# pcs resource create sharedfs2 --group shared_vg1 ocf:heartbeat:Filesystem device="/dev/shared_vg1/s
[root@z1 ~]# pcs resource create sharedfs3 --group shared_vg2 ocf:heartbeat:Filesystem device="/dev/shared_vg2/s
```

Verification steps

1. Verify that the GFS2 file systems are mounted on both nodes of the cluster.

```
[root@z1 ~]# mount | grep gfs2
/dev/mapper/shared_vg1-shared_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg1-shared_lv2 on /mnt/gfs2 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg2-shared_lv1 on /mnt/gfs3 type gfs2 (rw,noatime,seclabel)

[root@z2 ~]# mount | grep gfs2
/dev/mapper/shared_vg1-shared_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg1-shared_lv2 on /mnt/gfs2 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg2-shared_lv1 on /mnt/gfs3 type gfs2 (rw,noatime,seclabel)
```

2. Check the status of the cluster.

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
     Resource Group: locking:0
         dlm    (ocf::pacemaker:controld):      Started z2.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
     Resource Group: locking:1
         dlm    (ocf::pacemaker:controld):      Started z1.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
     Started: [ z1.example.com z2.example.com ]
 Clone Set: shared_vg1-clone [shared_vg1]
     Resource Group: shared_vg1:0
         sharedlv1      (ocf::heartbeat:LVM-activate):  Started z2.example.com
         sharedlv2      (ocf::heartbeat:LVM-activate):  Started z2.example.com
         sharedfs1      (ocf::heartbeat:Filesystem):    Started z2.example.com
         sharedfs2      (ocf::heartbeat:Filesystem):    Started z2.example.com
     Resource Group: shared_vg1:1
         sharedlv1      (ocf::heartbeat:LVM-activate):  Started z1.example.com
         sharedlv2      (ocf::heartbeat:LVM-activate):  Started z1.example.com
         sharedfs1      (ocf::heartbeat:Filesystem):    Started z1.example.com
         sharedfs2      (ocf::heartbeat:Filesystem):    Started z1.example.com
     Started: [ z1.example.com z2.example.com ]
 Clone Set: shared_vg2-clone [shared_vg2]
     Resource Group: shared_vg2:0
         sharedlv3      (ocf::heartbeat:LVM-activate):  Started z2.example.com
         sharedfs3      (ocf::heartbeat:Filesystem):    Started z2.example.com
     Resource Group: shared_vg2:1
         sharedlv3      (ocf::heartbeat:LVM-activate):  Started z1.example.com
         sharedfs3      (ocf::heartbeat:Filesystem):    Started z1.example.com
     Started: [ z1.example.com z2.example.com ]

...
```

Additional resources

- Configuring a Red Hat High Availability cluster on Microsoft Azure
- Configuring a Red Hat High Availability cluster on AWS
- Configuring Red Hat High Availability Cluster on Google Cloud Platform
- Configuring shared block storage for a Red Hat High Availability cluster on Alibaba Cloud

7.2. Configuring an encrypted GFS2 file system in a cluster

(RHEL 8.4 and later) This procedure creates a Pacemaker cluster that includes a LUKS encrypted GFS2 file system. This example creates one GFS2 file systems on a logical volume and encrypts the file system. Encrypted GFS2 file systems are supported using the `crypt` resource agent, which provides support for LUKS encryption.

There are three parts to this procedure:

- Configuring a shared logical volume in a Pacemaker cluster
- Encrypting the logical volume and creating a `crypt` resource
- Formatting the encrypted logical volume with a GFS2 file system and creating a file system resource for the cluster

7.2.1. Configure a shared logical volume in a Pacemaker cluster

Prerequisities

- Install and start the cluster software on all nodes and create a basic two-node cluster.
- Configure fencing for the cluster.

For information on creating a Pacemaker cluster and configuring fencing for the cluster, see Creating a Red Hat High-Availability cluster with Pacemaker.

Procedure

1. On both nodes in the cluster, enable the repository for Resilient Storage that corresponds to your system architecture. For example, to enable the Resilient Storage repository for an x86_64 system, you can enter the following `subscription-manager` command:

```
# subscription-manager repos --enable=rhel-8-for-x86_64-resilientstorage-rpms
```

   Note that the Resilient Storage repository is a superset of the High Availability repository. If you enable the Resilient Storage repository you do not also need to enable the High Availability repository.

2. On both nodes of the cluster, install the `lvm2-lockd`, `gfs2-utils`, and `dlm` packages. To support these packages, you must be subscribed to the AppStream channel and the Resilient Storage channel.

```
# yum install lvm2-lockd gfs2-utils dlm
```

3. Set the global Pacemaker parameter `no-quorum-policy` to `freeze`.

   NOTE

   By default, the value of `no-quorum-policy` is set to `stop`, indicating that once quorum is lost, all the resources on the remaining partition will immediately be stopped. Typically this default is the safest and most optimal option, but unlike most resources, GFS2 requires quorum to function. When quorum is lost both the applications using the GFS2 mounts and the GFS2 mount itself cannot be correctly stopped. Any attempts to stop these resources without quorum will fail which will ultimately result in the entire cluster being fenced every time quorum is lost.

   To address this situation, set `no-quorum-policy` to `freeze` when GFS2 is in use. This means that when quorum is lost, the remaining partition will do nothing until quorum is regained.

```
# pcs property set no-quorum-policy=freeze
```

4. Set up a `dlm` resource. This is a required dependency for configuring a GFS2 file system in a cluster. This example creates the `dlm` resource as part of a resource group named `locking`.

```
[root@z1 ~]# pcs resource create dlm --group locking ocf:pacemaker:controld op monitor interval=30s on-fail=fenc
```

5. Clone the `locking` resource group so that the resource group can be active on both nodes of the cluster.

```
[root@z1 ~]# pcs resource clone locking interleave=true
```

6. Set up an `lvmlockd` resource as part of the group `locking`.

```
[root@z1 ~]# pcs resource create lvmlockd --group locking ocf:heartbeat:lvmlockd op monitor interval=30s on-fail
```

7. Check the status of the cluster to ensure that the `locking` resource group has started on both nodes of the cluster.

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

Online: [ z1.example.com (1) z2.example.com (2) ]

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
     Resource Group: locking:0
         dlm    (ocf::pacemaker:controld):      Started z1.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
     Resource Group: locking:1
         dlm    (ocf::pacemaker:controld):      Started z2.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
     Started: [ z1.example.com z2.example.com ]
```

8. On one node of the cluster, create a shared volume group.

   The following command creates the shared volume group `shared_vg1` on `/dev/sda1`.

```
[root@z1 ~]# vgcreate --shared shared_vg1 /dev/sda1
  Physical volume "/dev/sda1" successfully created.
  Volume group "shared_vg1" successfully created
  VG shared_vg1 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

9. On the second node in the cluster, start the lock manager for the shared volume group.

```
[root@z2 ~]# vgchange --lock-start shared_vg1
  VG shared_vg1 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
[root@z2 ~]# vgchange --lock-start shared_vg2
  VG shared_vg2 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

10. On one node in the cluster, create the shared logical volume.

```
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv1 shared_vg1
  Logical volume "shared_lv1" created.
```

11. Create an `LVM-activate` resource for the logical volume to automatically activate the logical volume on all nodes.

    The following command creates an `LVM-activate` resource named `sharedlv1` for the logical volume `shared_lv1` in volume group `shared_vg1`. This command also creates the resource group `shared_vg1` that includes the resource. In this example, the resource group has the same name as the shared volume group that includes the logical volume.

```
[root@z1 ~]# pcs resource create sharedlv1 --group shared_vg1 ocf:heartbeat:LVM-activate lvname=shared_lv1 vgnam
```

12. Clone the new resource group.

```
[root@z1 ~]# pcs resource clone shared_vg1 interleave=true
```

13. Configure an ordering constraints to ensure that the `locking` resource group that includes the `dlm` and `lvmlockd` resources starts first.

```
[root@z1 ~]# pcs constraint order start locking-clone then shared_vg1-clone
Adding locking-clone shared_vg1-clone (kind: Mandatory) (Options: first-action=start then-action=start)
```

14. Configure a colocation constraints to ensure that the `vg1` and `vg2` resource groups start on the same node as the `locking` resource group.

```
[root@z1 ~]# pcs constraint colocation add shared_vg1-clone with locking-clone
```

Verification steps

On both nodes in the cluster, verify that the logical volume is active. There may be a delay of a few seconds.

```
[root@z1 ~]# lvs
  LV         VG          Attr       LSize
  shared_lv1 shared_vg1  -wi-a----- 5.00g

[root@z2 ~]# lvs
  LV         VG          Attr       LSize
  shared_lv1 shared_vg1  -wi-a----- 5.00g
```

7.2.2. Encrypt the logical volume and create a crypt resource

Prerequisites

- You have configured a shared logical volume in a Pacemaker cluster.

Procedure

1. On one node in the cluster, create a new file that will contain the crypt key and set the permissions on the file so that it is readable only by root.

```
[root@z1 ~]# touch /etc/crypt_keyfile
[root@z1 ~]# chmod 600 /etc/crypt_keyfile
```

2. Create the crypt key.

```
[root@z1 ~]# dd if=/dev/urandom bs=4K count=1 of=/etc/crypt_keyfile
1+0 records in
1+0 records out
4096 bytes (4.1 kB, 4.0 KiB) copied, 0.000306202 s, 13.4 MB/s
[root@z1 ~]# scp /etc/crypt_keyfile root@z2.example.com:/etc/
```

3. Distribute the crypt keyfile to the other nodes in the cluster, using the `-p` parameter to preserve the permissions you set.

```
[root@z1 ~]# scp -p /etc/crypt_keyfile root@z2.example.com:/etc/
```

4. Create the encrypted device on the LVM volume where you will configure the encrypted GFS2 file system.

```
[root@z1 ~]# cryptsetup luksFormat /dev/shared_vg1/shared_lv1 --type luks2 --key-file=/etc/crypt_keyfile
WARNING!
========
This will overwrite data on /dev/shared_vg1/shared_lv1 irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
```

5. Create the crypt resource as part of the `shared_vg1` volume group.

```
[root@z1 ~]# pcs resource create crypt --group shared_vg1 ocf:heartbeat:crypt crypt_dev="luks_lv1" crypt_type=lu
```

Verification steps

Ensure that the crypt resource has created the crypt device, which in this example is `/dev/mapper/luks_lv1`.

```
[root@z1 ~]# ls -l /dev/mapper/
...
lrwxrwxrwx 1 root root 7 Mar 4 09:52 luks_lv1 -> ../dm-3
...
```

7.2.3. Format the encrypted logical volume with a GFS2 file system and create a file system resource for the cluster.

Prerequisites

- You have encrypted the logical volume and created a crypt resource.

Procedure

1. On one node in the cluster, format the volume with a GFS2 file system. One journal is required for each node that mounts the file system. Ensure that you create enough journals for each of the nodes in your cluster.

```
[root@z1 ~]# mkfs.gfs2 -j3 -p lock_dlm -t my_cluster:gfs2-demo1 /dev/mapper/luks_lv1
/dev/mapper/luks_lv1 is a symbolic link to /dev/dm-3
This will destroy any data on /dev/dm-3
Are you sure you want to proceed? [y/n] y
Discarding device contents (may take a while on large devices): Done
Adding journals: Done
Building resource groups: Done
Creating quota file: Done
Writing superblock and syncing: Done
Device:                    /dev/mapper/luks_lv1
Block size:                4096
Device size:               4.98 GB (1306624 blocks)
Filesystem size:           4.98 GB (1306622 blocks)
Journals:                  3
Journal size:              16MB
Resource groups:           23
Locking protocol:          "lock_dlm"
Lock table:                "my_cluster:gfs2-demo1"
UUID:                      de263f7b-0f12-4d02-bbb2-56642fade293
```

2. Create a file system resource to automatically mount the GFS2 file system on all nodes.

   Do not add the file system to the `/etc/fstab` file because it will be managed as a Pacemaker cluster resource. Mount options can be specified as part of the resource configuration with `options=*options*`. Run the `pcs resource describe Filesystem` command for full configuration options.

   The following command creates the file system resource. This command adds the resource to the resource group that includes the logical volume resource for that file system.

```
[root@z1 ~]# pcs resource create sharedfs1 --group shared_vg1 ocf:heartbeat:Filesystem device="/dev/mapper/luks_
```

Verification steps

1. Verify that the GFS2 file system is mounted on both nodes of the cluster.

```
[root@z1 ~]# mount | grep gfs2
/dev/mapper/luks_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)

[root@z2 ~]# mount | grep gfs2
/dev/mapper/luks_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)
```

2. Check the status of the cluster.

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

Full list of resources:

  smoke-apc      (stonith:fence_apc):    Started z1.example.com
  Clone Set: locking-clone [locking]
      Resource Group: locking:0
          dlm    (ocf::pacemaker:controld):      Started z2.example.com
          lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
      Resource Group: locking:1
          dlm    (ocf::pacemaker:controld):      Started z1.example.com
          lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
     Started: [ z1.example.com z2.example.com ]
  Clone Set: shared_vg1-clone [shared_vg1]
     Resource Group: shared_vg1:0
             sharedlv1      (ocf::heartbeat:LVM-activate):  Started z2.example.com
             crypt       (ocf::heartbeat:crypt) Started z2.example.com
             sharedfs1      (ocf::heartbeat:Filesystem):    Started z2.example.com
    Resource Group: shared_vg1:1
             sharedlv1      (ocf::heartbeat:LVM-activate):  Started z1.example.com
             crypt      (ocf::heartbeat:crypt)  Started z1.example.com
             sharedfs1      (ocf::heartbeat:Filesystem):    Started z1.example.com
          Started:  [z1.example.com z2.example.com ]
...
```

7.3. Migrating a GFS2 file system from RHEL7 to RHEL8

This procedure allows you to use your existing Red Hat Enterprise 7 logical volumes when configuring a RHEL 8 cluster that includes GFS2 file systems.

In Red Hat Enterprise Linux 8, LVM uses the LVM lock daemon `lvmlockd` instead of `clvmd` for managing shared storage devices in an active/active cluster. This requires that you configure the logical volumes that your active/active cluster will require as shared logical volumes. Additionally, this requires that you use the `LVM-activate` resource to manage an LVM volume and that you use the `lvmlockd` resource agent to manage the `lvmlockd` daemon. See Configuring a GFS2 file system in a cluster for a full procedure for configuring a Pacemaker cluster that includes GFS2 file systems using shared logical volumes.

To use your existing Red Hat Enterprise Linux 7 logical volumes when configuring a RHEL8 cluster that includes GFS2 file systems, perform the following procedure from the RHEL8 cluster. In this example, the clustered RHEL 7 logical volume is part of the volume group `upgrade_gfs_vg`.

NOTE

The RHEL8 cluster must have the same name as the RHEL7 cluster that includes the GFS2 file system in order for the existing file system to be valid.

Procedure

1. Ensure that the logical volumes containing the GFS2 file systems are currently inactive. This procedure is safe only if all nodes have stopped using the volume group.
2. From one node in the cluster, forcibly change the volume group to be local.

```
[root@rhel8-01 ~]# vgchange --lock-type none --lock-opt force upgrade_gfs_vg
Forcibly change VG lock type to none? [y/n]: y
  Volume group "upgrade_gfs_vg" successfully changed
```

3. From one node in the cluster, change the local volume group to a shared volume group

```
[root@rhel8-01 ~]# vgchange --lock-type dlm upgrade_gfs_vg
   Volume group "upgrade_gfs_vg" successfully changed
```

4. On each node in the cluster, start locking for the volume group.

```
[root@rhel8-01 ~]# vgchange --lock-start upgrade_gfs_vg
  VG upgrade_gfs_vg starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
[root@rhel8-02 ~]# vgchange --lock-start upgrade_gfs_vg
  VG upgrade_gfs_vg starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

After performing this procedure, you can create an `LVM-activate` resource for each logical volume.

Chapter 6. Configuring an active/passive NFS server in a Red Hat High Availability cluster

Chapter 8. Getting started with the pcsd Web UI