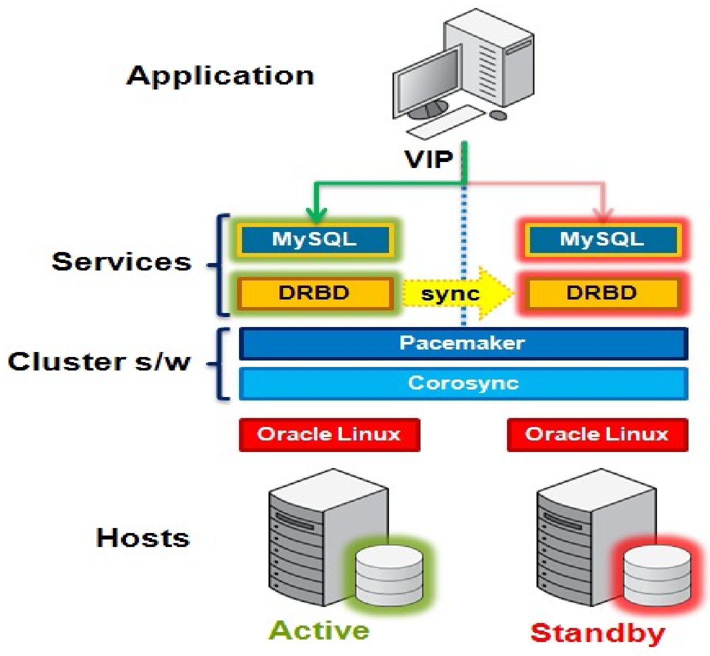
MySQL with DRBD/Pacemaker/Corosync on Linux

Definition of DRBD :- **DRBD** (**Distributed Replicated Block Device**)

DRBD synchronizes data at the block device (typically a spinning or solid state disk) – transparent to the application, database and even the file system. DRBD requires the use of a journaling file system such as ext3 or ext4. For this solution it acts in an active-standby mode – this means that at any point in time the directories being managed by DRBD are accessible for reads and writes on exactly one of the two hosts and inaccessible (even for reads) on the other. Any changes made on the active host are synchronously replicated to the standby host by DRBD.



-: The Concept :-

The concept of an active/passive fail-over Cluster is the following:

- Two servers (nodes).
- They communicate over a cluster software (Heartbeat,Corosync,Pacemaker)
- They are running on DRBD failover storage system.
- MySQL is only running in MASTER node (active), the other is the PASIVE node.
- You reach MySQL over a Virtual IP (ClusterIP)
- In case of a problem the cluster fail-over the resources including the VIP to the passive node.
- This fail-over is transparent for the application (a lite SERVICEDOWN).

Network and Server settings :-

- Network static ip set on Both Node
- Deactivate selinux (**/etc/sysconfig/selinux**)
- To simplification the configuration use host names. (**/etc/sysconfig/network**)
- Add the two nodes in the **/etc/hosts** file on Both Node

like this :-

```
127.0.0.1       localhost
192.168.0.251   Cluster1
 192.168.0.252   Cluster2
```

- Password-less communication with Both Node
- Install **mysql** with the help of Yum (**On Both Node for mounting DRBD created device**)

DRBD Installation & Configuration Step by Step :-

Simple Download **elrepo-release-6-5.el6.elrepo.noarch.rpm** for DRBD package installation .

Required Packages :-

1. drbd83-utils.x86_64
2. kmod-drbd83.x86_64

Note :- This package installation with the help of yum

Copy the dist configuration :-

```
# cp -pr /etc/drbd.conf /etc/drbd.conf-DIST
```

The **/etc/drbd.conf** is the configuration file for drbd, here my configuration:
look like this :-

```
# You can find an example in  /usr/share/doc/drbd.../drbd.conf.example
include "drbd.d/global_common.conf";
resource DISK1 {
  protocol C;
  net {
     cram-hmac-alg sha1;
     shared-secret "8a6cxxxxxxxxxxxxxxxxxxxxx49xxxxxxxxfb3";
     after-sb-0pri discard-zero-changes;
```

```
        after-sb-1pri discard-secondary;
        after-sb-2pri disconnect;
        rr-conflict disconnect;
    }
    device    /dev/drbd0;
    disk      /dev/sda5;
    meta-disk internal;
    on Cluster1 {
        address   192.168.0.251:7789;
    }
    on Cluster2 {
        address   192.168.0.252:7789;
    }
}
```

As you can see I specify this parameters :-

RESOURCE: The name of the resource

- PROTOCOL:In this case **C** means synchronous
- NET: The SHA1 key, **that have the same in the two nodes**
- **after-sb-0pri** : When a Split Brainocurrs, and no data have changed, the two nodes connect normally.
- **after-sb-1pri** : If some data have been changed, discard the secondary data and synchronize with the primary
- **after-sb-2pri** : If the previous option is impossible disconnect the two nodes, in this case manually Split-Brain solution is required
- **rr-conflict:**In case that the previous statements don't apply and the drbd system have a role conflict, the system disconnect automatically.
- DEVICE: Virtual device, the patch to the fisical device.
- DISK: Fisical device
- META-DISK: Meta data are stored in the same disk (sdc1)
- ON <NODE>: The nodes that form the cluster

Creating the resource

This commands **in both nodes**

```
# fdisk /dev/sda
```

Create partition

Create the partition **without format**

```
[root@node1 ~]# drbdadm create-md DISK1
Writing meta data...
initializing activity log
NOT initializing bitmap
New drbd meta data block successfully created.
[root@node2 ~]# drbdadm create-md DISK1
Writing meta data...
initializing activity log
NOT initializing bitmap
New drbd meta data block successfully created.
```

Activate the Resource

Be sure that the drbd module is load (lsmod), if not load it:

```
# modprobe drbd
```

(On Both Node)

Now activate the resource DISK1:

```
[root@node1 ~]# drbdadm up DISK1
[root@node2 ~]# drbdadm up DISK1
```

Synchronize

Only in the master node, we'll say that the node1 is the primary:

```
# drbdadm -- --overwrite-data-of-peer primary DISK1
```

We'll see that the disks synchronization are in progress, adn the state is **UpToDate/Inconsistent**

Check Synchronization status

```
# /etc/init.d/drbd status
```

(On Both node)

Format the Resource

Only in the master node:

```
# mkfs.ext4 /dev/drbd0
```

Testing

Mount the resource in the node1

```
[root@node1 ~]# mount /dev/drbd0 /var/lib/mysql
```

Ok, now umount and mark the node1 like secondary

```
[root@node1 ~]# umount /var/lib/mysql
```

```
[ node1 ~]# drbdadm secondary DISK1
```

Mark Node2 like Primary and mount:

```
[root@node2 ~]# drbdadm primary DISK1
[root@node2 ~]# mount /dev/drbd0 /var/lib/mysql
```

Introduction of Corosync & Pacemaker

Pacemaker and Corosync combine to provide the clustering layer that sits between the services and the underlying hosts and operating systems. Pacemaker is responsible for starting and stopping services – ensuring that they're running on exactly one host, delivering high availability and avoiding data corruption. Corosync provides the underlying messaging infrastructure between the nodes that enables Pacemaker to do its job; it also handles the nodes membership within the cluster and informs Pacemaker of any changes.

Installation of Corosync & Pacemaker step by step

- **elrepo-release-6-5.el6.elrepo.noarch.rpm** **(.repo file)** for corosync & Pacemaker package installation
- Package install with the help of yum

```
# yum install pacemaker.x86_64 corosync.x86_64
```

- configure corosync to use cryptographic techniques to ensure authenticity and privacy of the messages, you will need to generate a private key.Corosync Cluster Engine Authentication key generator.Gathering 1024 bits for key from /dev/random.

- Corosync Key In one node create the corosync security comunication key.
- Corosync for the first time, you need to create the authkey-file for authentication within cluster communication.

```
[root@node1]# corosync-keygen
```

```
Corosync Cluster Engine Authentication key generator.
Gathering 1024 bits for key from /dev/random.
Press keys on your keyboard to generate entropy.
Writing corosync key to /etc/corosync/authkey.
```

- need to copy that file to all of your nodes and put it in /etc/corosync/ with user=root, group=root and mode 0400. permissions to **400**

```
[root@node1]# scp /etc/corosync/authkey node2:/etc/corosync/
```

```
[root@node1]# ll /etc/corosync/authkey
-r-------- 1 root root 128 july  17 10:26 /etc/corosync/authkey
[root@node2]# ll /etc/corosync/authkey
-r-------- 1 root root 128 july  17 10:27 /etc/corosync/authkey
```

- Now configure the **/etc/corosync/corosync.conf**

Look like this -:

```
[root@node1]# vi /etc/corosync/corosync.conf

 # Please read the corosync.conf.5 manual page

compatibility: whitetank

totem {

      version: 2

        secauth: off

        interface {

                member {

                        memberaddr: 192.168.0.251

                }

                member {

                        memberaddr: 192.168.0.252

                }

                ringnumber: 0

        mcastaddr: 239.255.255.255

        bindnetaddr: 192.168.0.0

                mcastport: 5405

                ttl: 1

        }

        transport: udpu

}

logging {

        fileline: off

        to_logfile: yes

        to_syslog: yes
        debug: on

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

        debug: off

        timestamp: on

        logger_subsys {

                subsys: AMF

                debug: off

        }

}
```

- Now Copy **corosync.conf configured file on Node02**

  [root@node1]# scp /etc/corosync/corosync.conf node2:/etc/corosync/.

- Corosync to load the quorum and messaging interfaces needed by pacemaker, create */etc/corosync/service.d/pcmk* with the following fragment.

```
 service {
   # Load the Pacemaker Cluster Resource Manager
   name: pacemaker
   ver:  1
 }
```

- Now copy on Node02

  [root@node1]# ***scp*** */etc/corosync/service.d/pcmk* **node2:***/etc/corosync/service.d/.*

- crm command(cluster managment for pacemaker) not found in latest Centos 6

```
# yum install crmsh.x86_64 ( On Both Node )
```

- Now create Log Directory and file under /var/log/   (Both Node)

```
# mkdir -p /var/log/cluster

# vi /var/log/cluster/corosync.log
```

- Corosync process would launch pacemaker, this is no longer the case. Pacemaker must be launched after Corosync has successfully started. (Both Node)

```
# service corosync start

# service pacemaker start

# chkconfig corosync on

# chkconfig pacemaker on
```

- crm_mon is run with the -1 option to indicate that it should report once and then return. A recommendation would be to also run it without the option (on both servers) so that you get a continually refreshed view of the state of the cluster – including any managed resources.

- Check Online configured Cluster node with crm command (On Both Node)

```
[root@node01]# crm_mon -1
 ============
```

Output show look like this :-

```
Last updated: Mon Feb 27 17:51:10 2012

Last change: Mon Feb 27 17:50:25 2012 via crmd on host1.localdomain

Stack: openais

Current DC: host1.localdomain - partition with quorum

Version: 1.1.6-3.el6-a02c0f19a00c1eb2527ad38f146ebc0834814558

2 Nodes configured, 2 expected votes

0 Resources configured.

 ============

 Online: [ Node01 Node02 ]
```

Configuring and Managing Cluster Resources (Command Line)

- Use the following commands to set the options for a two-node clusters only:

```
[root@node01 ~]# crm configure property no-quorum-policy=ignore
```

- Pacemaker uses “resource stickiness” parameters to determine when resources should be migrated between nodes – the absolute values are not important, rather how they compare with the values that will subsequently be configured against specific events; here we set the stickiness to 100:

```
[root@node01 ~]# crm configure rsc_defaults resource-stickiness=100
```

- STONITH (Shoot The Other Node In The Head) – otherwise known as fencing – refers to one node trying to kill another in the even that it believes the other has partially failed and should be stopped in order to avoid any risk of a split-brain scenario. We turn this off as this solution will rely on each node shutting itself down in the event that it loses connectivity with the independent host:

```
[root@node01 ~]# crm configure property stonith-enabled=false
```

- Now configure Virtual IP resource for this nodes. Nodes have to check each other every 20 seconds:

```
[root@node1]#crm configure
crm(live)configure# primitive ClusterIP ocf:heartbeat:IPaddr2 params ip=192.168.1.100
cidr_netmask=32
```

DRBD

- Now the filesystem, add **DISK1** to the cluster

```
crm(live)configure# primitive drbd_mysql ocf:linbit:drbd params drbd_resource="DISK1" op
monitor interval="15s" op start timeout="240s"
```

Define the mount point

```
crm(live)configure# primitive fs_mysql ocf:heartbeat:Filesystem params device="/dev/drbd0"
directory="/var/lib/mysql" fstype="ext4"
```

Define only one Master node

```
crm(live)configure# ms ms_drbd_mysql drbd_mysql meta master-max="1" master-node-max="1"
clone-max="2" clone-node-max="1" notify="true"
```

MySQL

Now the mysql server

```
crm(live)configure# primitive mysqld ocf:heartbeat:mysql params binary="/usr/bin/mysqld_safe"
config="/etc/my.cnf" user="mysql" group="mysql" log="/var/log/mysqld.log"
pid="/var/run/mysqld/mysqld.pid" datadir="/var/lib/mysql" socket="/var/lib/mysql/mysql.sock" op
monitor interval="60s" timeout="60s" op start interval="0" timeout="180" op stop interval="0"
timeout="240"
```

Groups & Colocations

- With this group we ensure that the drbd, mysql and VIP are in the same node (master) and the order to stop and start is correctly:

  start: fs_mysql–>mysqld–>ClusterIP
  stop: ClusterIP–>mysqld–>fs_mysql

```
crm(live)configure# group group_mysql fs_mysql mysqld ClusterIP meta migration-threshold="5"
```

The group group_mysql allways in the MASTER node

```
crm(live)configure# colocation mysql_on_drbd inf: group_mysql ms_drbd_mysql:Master
```

Mysql start allways after drbd MASTER

```
crm(live)configure# order mysql_after_drbd inf: ms_drbd_mysql:promote group_mysql:start
```

How to configure pacemaker apache fail-over system with pacemaker

```
crm(live)configure# crm configure primitive APACHE ocf:heartbeat:apache params
configfile="/etc/httpd/conf/httpd.conf" statusurl="http://localhost/server-status" op monitor
interval="40s"
```

### APACHE – resource name
### configfile – path to apache configuration file
### statusurl – url to status page(below how to configures one)
### interval – time between checks

### In Apache Configuration file :-

```
# vi /etc/httpd/conf/httpd.conf

ExtendedStatus On

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

- **To prevent situation when resource apache migrate to node002 and resource IP stays at node001(It happens when apache at node001 hung but network stack works well) we need to make colocation**

```
[root@node1]# crm configure colocation WEB_SITE inf: APACHE ClusterIP
```

- **To make pacemaker start up apache only after IP is set up. In other words describe start up order run:**

```
[root@node1]# crm configure order START_ORDER inf: ClusterIP APACHE
```

- **Now check configuration on both Node**

```
# crm_mon -1
# crm configure show
# crm_mon
```

By :- Alok Raj

System Administrator

***Saigun Technologies* Pvt Ltd**