4. 고 가용성 – High Availability 클러스터 시스템 구축

4.1. Open Source를 이용한 Linux HA 시스템 구축 준비

High Availability 는 흔히 고 가용성이라고 얘기 하며, 시스템이 정상적으로 작동 하는 시간과 문제 발생 시 복구 되는 시간을 기준으로 측정한 수치로 가용성의 한 레벨로 볼 수 있다. 흔히 99.999% 의 가용성 수치가 나왔을 때 이 시스템을 고 가용성 시스템이라 칭한다.
이 단원에서는 Linux 가 지원하는 기능을 이용하여 실무에서 사용 할 수 있는 고 가용성 시스템을 구축 하는 방법에 대해 알아보도록 한다.

4.1.1 High Availability 시스템 구축 준비

고 가용성을 보장하기 위해서 반드시 필요한 요소는 서비스 서버에 문제 발생 시 서비스의 접속 포인트를 정상 작동 중인 시스템으로 자동 전환 시키는 fail over 기능과 문제 서버가 다시 정상 작동 되었을 경우 본 서비스 구조로 전환 하는 fail back 기능이 필요하다.

또한 메인 서비스 서버와 백업 서비스 서버의 데이터가 항상 동기화가 되어서 데이터 일관성이 유지 되어야 메인 서비스 서버에서 백업 서비스 서버로 시스템이 전환 될 때 서비스가 정상적으로 지속이 되어 질 것이다. 데이터 동기화 방식은 HA 대상 어플리케이션에 따라 Pvfs, Rsync, Csync 와 같은 비 동기적 File Level replication 방식을 사용할 수도 있고, Drbd, GFS, Gnbd, enbd 와 같은 low block level real time replication 방식을 사용 할 수도 있다.
여기서는 먼저 Oracle 과 같이 데이터의 일관성이 중요한 어플리케이션에서도 무난히 데이터 일관성을 보장하는 real time data sync를 지원하는 DRBD 와 Linux HA 에서 주로 사용되는 heartbeat 란 대표적인 HA 프로그램을 이용하여 HA 시스템을 구축하는 방법에 대해 알아 보도록 한다. 그리고 국내 개발의 대표적인 HA 상용 어플리케이션으로 실무에서 주로 요구하는 HA 시스템을 구축하여 보도록 한다.

- DRBD 최신 버전 다운로드

http://oss.linbit.com/drbd/

당시 최신 버전 drbd-0.7.5.tar.gz를 받는다

drbd 는 high availability cluster를 구현하는데 필요한 network block device를 제공하는 kernel

modules 과 scripts로 구성되어 있다. network를 통한 block device를 통해
remote disk 간의 mirroring 기능을 제공한다. 그러므로 현 시스템의 kernel version 과의
의존성이 크다. 설치 되는 Makefile 에서 현 시스템의 kernel version 과 modules version 을
check 하게 되는데 Redhat 기본 Kernel에서 잘 안 되는 경우가 종종 발생한다. 발생 되는 에러
메시지를 자세히 체크하면 문제를 해결 할 수 있을 것이다.

original kernel source version 2.4.27로 테스트를 했다.

- Kernel 환경 준비하기

먼저 linux-2.4.27.tar.bz2를 /usr/src 밑에 놓고 kernel compile를 한다.

Kernel 컴파일 시 꼭 포함해 줘야 할 것은

Kernel 모듈 인식

CONFIG_MODULES=y
CONFIG_MODVERSIONS=y
CONFIG_KMOD=y

NBD 모듈 인식

CONFIG_BLK_DEV_LOOP=y
CONFIG_BLK_DEV_NBD=y
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_SIZE=4096
CONFIG_BLK_DEV_INITRD=y
CONFIG_BLK_STATS=y

reiserfs 파일 시스템 인식

CONFIG_REISERFS_FS=y

정도 이다.

```
# cd /usr/src/linux

# make dep && make clean && make bzImage && make modules && make modules_install
# cp System.map /boot/System.map-2.4.27
# cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.27
# cd /boot
# ln -sf System.map-2.4.27 System.map
# ln -sf vmlinuz-2.4.27 vmlinuz

# cd /lib/modules
# depmod -a 2.4.27
```

/etc/lilo.conf 에 new building kernel 을 인식할수 있도록 적용 후 reboot

4.1.2. DRBD 구축 하기

- DRBD 설치 하기

먼저 /usr/src 및에 drbd source 를 copy 해 둔다.

```
# cp drbd-0.7.5.tar.gz /usr/src
# tar xzvf drbd-0.7.5.tar.gz
# cd drbd-0.7.5
# cd drbd
# make clean all
# cd ../user
# make
# cd ..
# make all
# make install
```

이로써 drbd 컴파일이 완료된다. 아무런 error 없이 컴파일이 되었으면 정상적으로 drbd modules 생성되었는지 확인한다.

```
# ls -al /lib/modules/2.4.27/kernel/drivers/block/drbd.o
/lib/modules/2.4.27/kernel/drivers/block/drbd.o
```

그런후 drbd device 를 생성한다.

```
# vi mkdrbddev
----------------------------------------------------------------------
#!/bin/sh

for i in $(seq 0 15)
do mknod /dev/drbd$i b 147 $i
done

----------------------------------------------------------------------

# sh mkdrbddev
# ls -al /dev/drbd0
brw-r--r--      1 root       root       147,     0 Oct 31 15:47 /dev/drbd0
```

이로써 설치가 완료 되었다. 이제 지금껏 생성한 DRBD 장치를 이용하여 설정을 해보도록 하자.

먼저 DRBD 시스템 설정 환경을 정하도록 하자.

master node :

hostname : test01
ip address : 192.168.123.165

slave node :

hostname : test02
ip address : 192.168.123.166

- DRBD 설정하기

vi /etc/drbd.conf 설정 하기

vi /etc/drbd.conf

```
---------------------------------------------------------------------------

skip {
   As you can see, you can also comment chunks of text
   with a 'skip[optional nonsense]{ skipped text }' section.
   This comes in handy, if you just want to comment out
   some 'resource <some name> {...}' section:
   just precede it with 'skip'.

   The basic format of option assignment is
   <option name><linear whitespace><value>;

   It should be obvious from the examples below,
   but if you really care to know the details:

   <option name> :=
         valid options in the respective scope
   <value>   := <num>|<string>|<choice>|...
                depending on the set of allowed values
                for the respective option.
   <num>     := [0-9]+, sometimes with an optional suffix of K,M,G
   <string> := (<name>|₩"([^₩"₩₩₩n]*|₩₩.)*₩")+
   <name>    := [/_.A-Za-z0-9-]+
}


resource drbd0 {
   protocol C;
```


클루닉스 서 진우(alang@clunix.com)

```
incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
startup {
   degr-wfc-timeout 120;
}

disk {
   on-io-error     detach;
}

net {
   # sndbuf-size 512k;
   # timeout           60;      #   6 seconds   (unit = 0.1 seconds)
   # connect-int     10;      # 10 seconds   (unit = 1 second)
   # ping-int          10;      # 10 seconds   (unit = 1 second)
   # max-buffers        2048;
   # max-epoch-size    2048;
   # ko-count 4;
   # on-disconnect reconnect;

}

syncer {
   rate 10M;
   group 1;
   al-extents 257;
}

on test01 {
   device        /dev/drbd0;
   disk          /dev/sda7;
   address       192.168.123.165:7788;
   meta-disk     internal;
}

on test02 {
```

```
    device      /dev/drbd0;
    disk        /dev/sda7;
    address     192.168.123.166:7788;
    meta-disk internal;
  }
}
```

위의 drbd.conf 파일을 test01, test02 의 /etc 에 copy 한다.

test01, test02 Node 에서 아래 command 를 실행한다.

;; test01

```
# modprobe drbd; drbdadm up all; dmesg | tail; cat /proc/drbd
```

;; test02

```
# modprobe drbd; drbdadm up all; dmesg | tail; cat /proc/drbd
```

```
drbd: SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
drbd: registered as block device major 147
drbd0: resync bitmap: bits=5535754 words=172994
drbd0: size = 21 GB (22143016 KB)
drbd0: 17 GB marked out-of-sync by on disk bit-map.
drbd0: Found 2 transactions (2 active extents) in activity log.
drbd0: Marked additional 1024 KB as out-of-sync based on AL.
drbd0: drbdsetup [1364]: cstate Unconfigured --> StandAlone
drbd0: drbdsetup [1366]: cstate StandAlone --> Unconnected
drbd0: drbd0_receiver [1367]: cstate Unconnected --> WFConnection
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:WFConnection st:Secondary/Unknown ld:Inconsistent
```

```
    ns:0 nr:0 dw:0 dr:0 al:0 bm:2 lo:0 pe:0 ua:0 ap:0
```

test01, test02 Node 에서 차례로 drbd 데몬을 실행한다.

;; test01

```
# /etc/rc.d/init.d/drbd start

 DRBD's startup script waits for the peer node(s) to appear.
 - In case this node was already a degraded cluster before the
    reboot the timeout is 120 seconds. [degr-wfc-timeout]
 - If the peer was available before the reboot the timeout will
    expire after 0 seconds. [wfc-timeout]
    (These values are for resource 'drbd0'; 0 sec -> wait forever)
 To abort waiting enter 'yes' [     8]: To abort waiting enter 'yes' [ -- ]:
```

위의 메세지는 test02 노드의 drbd 데몬이 실행이 안되어서 통신이 안되기 때문에 test02 노드가 drbd 통신을 할때까지 대기 하고 있다는 메세지다. test02 의 drbd 데몬을 실행하면 자동으로 없어진다.

;; test02

```
# /etc/rc.d/init.d/drbd start
```

이제 drbd 가 정상적으로 작동하는지 확인 하자

```
[root@test01 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test01, 2004-10-31 16:13:21
 0: cs:SyncSource st:Secondary/Secondary ld:Consistent
    ns:824336 nr:0 dw:0 dr:825060 al:0 bm:51 lo:74 pe:36 ua:181 ap:0
        [>.....................] sync'ed:   4.6% (16891/17696)M
        finish: 0:42:23 speed: 6,716 (5,720) K/sec
```

```
[root@test02 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:SyncTarget st:Secondary/Secondary ld:Inconsistent
    ns:0 nr:1016336 dw:1016336 dr:0 al:0 bm:65 lo:6 pe:256 ua:6 ap:0
        [=>.................] sync'ed:   5.7% (16703/17696)M
        finish: 0:35:38 speed: 7,996 (5,840) K/sec
```

현재 두 Node 모두 Secondary 상태로 실행되고 있다는 것을 알수 있다.
이제 test01 노드를 Primary 로 전환해 보도록 하자.

;; test01

```
# drbdadm primary all
```

만일 아래와 같은 메세지가 출력 되면

```
ioctl(,SET_STATE,) failed: Input/output error
Local replica is inconsistent (--do-what-I-say ?)
```

```
# drbdadm -- --do-what-I-say primary all
```

같이 실행하면 된다.
다시 확인 해보도록 하자.

```
[root@test01 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test01, 2004-10-31 16:13:21
 0: cs:SyncSource st:Primary/Secondary ld:Consistent
    ns:2204520 nr:0 dw:0 dr:2205544 al:0 bm:136 lo:256 pe:0 ua:256 ap:0
        [==>................] sync'ed: 12.3% (15543/17696)M
        finish: 11:03:10 speed: 36 (5,696) K/sec
```

;; test02

```
[root@test02 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:SyncTarget st:Secondary/Primary ld:Inconsistent
    ns:0 nr:2263064 dw:2263064 dr:0 al:0 bm:141 lo:7 pe:256 ua:7 ap:0
        [==>.................] sync'ed: 12.6% (15486/17696)M
        finish: 0:40:39 speed: 6,404 (5,640) K/sec
```

이제 Primary Node 에 drbd 를 device 에 mount 를 시켜 보도록 하자.

;; test01

```
# mkreiserfs /dev/drbd0
# mkdir /data1
# mount /dev/drbd0 /data1
```

-> /dev/drbd0 device 에 파일 시스템을 재생성 할때 물리적 device 가 format 되어 버린다.

이제 test01 에 /data1 에서 발생하는 모든 file system 의 writing 작업은 drbd dev를 통해서 test02 의 /dev/sda7 에도 일어나게 된다.

- DRBD 수동 전환 방식

위의 과정으로 DRBD 실제 셋팅이 완료 되었다. 이제 실제 노드에 failed status 를 가정할때 Node 전환 방식을 수동으로 확인해 보자. 실제 HA 프로그램을 이용하여 이런 일련의 과정을 자동화 할수 있을 것이다.

* Primary/Secondary Switching

test01 Node를 primary , test02 Node 를 secondary 로 가정. test01, test02의 역할을 수동 으로 전환 하는 방식이다.

;; test01

```
[root@test01 root]# umount /data1
[root@test01 root]# /etc/rc.d/init.d/drbd stop
```

* 즉 test01 node 가 정상 작동 안하는 것을 가정한다.

;; test02

```
[root@test02 /]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:WFConnection st:Secondary/Unknown ld:Consistent
    ns:0 nr:696 dw:696 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
```

test02 상태를 보면 현재 Secondary status 이고 primary 상태의 노드는 인식할 수 없는 상태 이다.

test02 를 primary 로 전환 한다.

```
[root@test02 /]# drbdadm primary all

[root@test02 /]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:WFConnection st:Primary/Unknown ld:Consistent
    ns:0 nr:696 dw:696 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0

[root@test02 /]# mount /dev/drbd0 /data1
```

```
[root@test02 /]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda2              2016044    149888   1763744   8% /
/dev/sda1               202220     17642    174138  10% /boot
none                    515760         0    515760   0% /dev/shm
/dev/sda3              7052496   2001624   4692624  30% /usr
/dev/sda6              2016016    242608   1670996  13% /var
/dev/drbd0            22142336     67440  22074896   1% /data1
```

* test02 node 는 test01 node를 주기적으로 checking 하다 문제 발생 시 위 일련의 가정을 거쳐서 Secondary 상태에서 Primary 상태로 전환되어 파일 서비스를 진행하게 된다.

이제 test01 node 가 다시 살아 나서 정상적인 역할 상태로 전환 한다고 가정해 보자

;; test01

```
[root@test01 root]# /etc/rc.d/init.d/drbd start

[root@test01 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test01, 2004-10-31 16:13:21
 0: cs:Connected st:Secondary/Primary ld:Consistent
    ns:0 nr:40 dw:40 dr:0 al:0 bm:3 lo:0 pe:0 ua:0 ap:0
```

test01 node 가 살아 나게 되면 drbd status 가 Secondary 상태로 있게 된다.
이를 다시 Primary 로 전환 한다.

;; test02

```
[root@test02 /]# umount /data1
[root@test02 /]# drbdadm secondary all
[root@test02 /]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
```

```
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:Connected st:Secondary/Primary ld:Consistent
    ns:80 nr:696 dw:780 dr:848 al:0 bm:8 lo:0 pe:0 ua:0 ap:0

;; test01

[root@test01 root]# drbdadm primary all
[root@test01 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test01, 2004-10-31 16:13:21
 0: cs:Connected st:Primary/Secondary ld:Consistent
    ns:0 nr:80 dw:80 dr:0 al:0 bm:3 lo:0 pe:0 ua:0 ap:0
[root@test01 root]# mount /dev/drbd0 /data1
```

- DRBD Multi Instance 구성하기

/etc/drbd.conf 를 아래와 같이 구성한다.

```
# vi /etc/drbd.conf
-------------------------------------------------------------------------
resource drbd0 {
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
  startup {
    degr-wfc-timeout 120;      # 2 minutes.
  }


  disk {
    on-io-error     detach;
  }

  net {
```

```
  }

  syncer {
    rate 10M;
    group 0;
    al-extents 257;
  }

  on test01 {
    device      /dev/drbd0;
    disk        /dev/sda7;
    address     192.168.123.165:7788;
    meta-disk   internal;
  }

  on test02 {
    device      /dev/drbd0;
    disk        /dev/sda7;
    address     192.168.123.166:7788;
    meta-disk internal;
  }

resource drbd1 {

  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
  startup {
    degr-wfc-timeout 120;    # 2 minutes.
  }


  disk {
    on-io-error    detach;
  }
```

```
  net {
  }

  syncer {
    rate 10M;
    group 0;
    al-extents 257;
  }

  on test01 {
    device      /dev/drbd1;
    disk        /dev/sda8;
    address     192.168.123.165:7799;
    meta-disk   internal;

  }

  on test02 {
    device      /dev/drbd1;
    disk        /dev/sda8;
    address     192.168.123.166:7799;
    meta-disk internal;
  }

}
```

---

drbd0, drbd1 두개의 resource 설정을 한 후 /etc/rc.d/init.d/drbd start 를 해주면 된다.

각 노드에 대한 primary, secondary 설정은 해당 노드의 정책에 맞게 구성해 주면 된다. 즉

| | test01 | test02
---
drbd0 | primary | secondary

---

drbd1 | secondary | primary

와 같은 구성을 위해서는..

;; test01

drbdadmin primary drbd0

;; test02

drbdadmin primary drbd1

명령을 해주면 된다.

4.1.3. Heartbeat 구축 하기

- Heartbeat 설치 하기

http://www.linux-ha.org

test01, test02 node 에 heartbeat package를 설치 한다.

heartbeat-stonith-1.2.2-8.rh.9.i386.rpm
heartbeat-pils-1.2.2-8.rh.9.i386.rpm
heartbeat-1.2.2-8.rh.9.i386.rpm

[root@test01 src]# rpm -Uvh heartbeat-*

- Heartbeat 설정 하기

heartbeat 의 설정 파일은 크게 3개로 나누어 진다. 이 3개의 설정파일의 설정만 완료 되면 정상적인 HA 가 이루어 진다.

ha.cf -> ha 의 기본 환경 설정 파일이다.
authkeys -> ha 노드간의 인증 방식을 지정하는 파일이다.
haresources -> heartbeat 구성에서 vip 와 실제 node failed 시 실행 스크립트를 지정하는 설정 파일이다.

[root@test01 src]# cd /etc/ha.d

[root@test01 ha.d]# vi ha.cf

```
---------------------------------------------------------------------------

debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 5
hopfudge 1
udpport 1001
udp     eth0
node    test01
node    test02

---------------------------------------------------------------------------
```

[root@test01 ha.d]# vi authkeys

```
---------------------------------------------------------------------------
auth 1
1 sha1 HI!
```

[root@test01 ha.d]# chmod 600 authkeys

;; 반드스 인증 파일은 퍼미션 600 으로 해주어야 한다.

[root@test01 ha.d]# vi haresources

---

test01 192.168.123.170 mysqld

---

haresources 설정 파일의 형식은 다음과 같다.

<< mater node >> << vip >> << init scripts >>

- Multi Heartbeat 구성 하기

4.1.2 에서 multi Drbd instance 구성하기에 대해 잠시 알아 보았다. 여기서는 Multi instance로 구성된 DRBD를 Heartbeat 에서도 dual로 구성하여 사용 할 수 있는 방법을 알아보도록 한다.

Heartbeat를 multi 로 구성하는 방법은 아주 간단하다.

[root@test01 root]# vi /etc/ha.d/haresources

---

test01 192.168.123.170 drbddisk::drbd0 Filesystem::/dev/drbd0::/data1::reiserfs mysqld

- 아래 추가 -

test02 192.168.123.171 drbddisk::drbd1 Filesystem::/dev/drbd1::/data2::reiserfs mysqld

---

와 같이 해주면 된다.

이 단원의 방법으로 both HA on both Drbd System 이 구축이 되는 것이다.

간단히 위 시스템 설계에 대해 설명하면..

test01, test02 시스템에 각각 두 개의 data volume (data1, data2) 이 존재하고 이 두 개의 volume 을 drbd0, drbd1 로 구성한다.

그런 후 아래와 같이 각 시스템의 drbd 권한을 준다.

test01 ->
    drbd0(/data1) - primary
    drbd1(/data2) - secondary

test02 ->
    drbd0(/data1) - secondary
    drbd1(/data2) - primary

그런 후 mysql datafile path를 아래와 같이 구성하여 시스템을 작동시킨다.

test01 ->
    /data1 -> mysql datafile of test01 node

test02 ->
    /data2 -> mysql datafile of test02 node

이제 시스템을 test01 에 /data1 에 datafile 을 둔 mysql 이, test02 에 /data2 에 datafile 을 둔 mysql 이 각각 작동하면서 test01 에서는 /data2 에 test02 의 mysql datafile 을 real time volum mirroring 을 되어지고, test02 에서는 /data1 에 test01 의 mysql datafile 이 real time volum mirroring 이 되어진다.

test01 노드에 이상이 생기면 test02 에서 자동으로 /data1 을 mount 시키며 test01 에서의 mysql 서비스가 구동되어진다.

test02 노드에 이상이 생기면 test01 에서 자동으로 /data2 을 mount 시키며 test02 의 mysql 서비스가 지속적으로 구동이 되어지는 것이다.

주의 할 점은 위의 구성으로 시스템을 구성하기 위해서 mysql 의 기본 instance

가 두 개가 될 수 있는 설정을 해야 한다. --datefile --port 등의 옵션을 주고
컴파일을 해서 한 시스템에 두 개의 mysql 을 동작 시켜야 할 것이다.

이 방법의 각자의 엔지니어링 역량에 맡기도록 하겠다.

4.2 Heartbeat 와 DRBD를 이용한 File System HA 하기

여기서는 앞에서 살펴본 Heartbeat 와 Drbd를 이용하여 서비스 상태에 따라 자동으로 Mount 와 Unmount를 시켜주는 방식에 대해 알아 보도록 한다. 이 방식을 이용하여 Samba, NFS, PVFS 와 같은 HA 파일 시스템을 구축 할 수 있을 것이다.

test01, test02 node 에 heartbeat package를 설치 한다.

heartbeat-stonith-1.2.2-8.rh.9.i386.rpm
heartbeat-pils-1.2.2-8.rh.9.i386.rpm
heartbeat-1.2.2-8.rh.9.i386.rpm

[root@test01 src]# rpm -Uvh heartbeat-*

[root@test01 src]# cd /etc/ha.d

[root@test01 ha.d]# vi ha.cf

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 5
hopfudge 1
udpport 1001
udp     eth0
node    test01
```

node test02

[root@test01 ha.d]# vi authkeys

---

auth 1
1 sha1 HI!

[root@test01 ha.d]# chmod 600 authkeys

[root@test01 ha.d]# vi haresources

---

test01 192.168.123.170 drbddisk Filesystem::/dev/drbd0::/data1::reiserfs mysqld

---

위 설정을 test02 에도 동일하게 해준다. 그냥 copy 해도 된다.
haresources 설정을 간단히 설명하면 다음과 같다.

test01 -> ha active node
192.168.123.170 -> service ip address

Filesystem::/dev/drbd0::/data1::reiserfs ->

/etc/ha.d/resource.d/Filesystem 이란 scripts 가 존재한다.
이 스크립트의 역할은 /dev/drbd0 device 를 mount point 에 mount 시켜 주는 역할을 한다.

사용 방법은 ./Filesystem <device> <directory> <fstype> {start|stop|status}
이다.

drbd 에서는 "./Filesystem /dev/drbd0 /data1 reiserfs start"
하면 drbd device 에 mount 가 되는 것이다.
"./Filesystem /dev/drbd0 /data1 reiserfs stop" 하면 drbd device 에 mount 된 경로
가 umount 가 되어 진다.

drbddisk ->

drbddisk scripts 는 실제 drbd 의 역할을 drbd node 간의 역할을 정하는 것이다.
drbd Node 는 각각 primary 와 secondary 로 각각의 역할이 있다. 만일 특정 노드에서
"drbddisk start" 를 하면 이 node 가 primary 로 존재하게 된다. stop 시는 secondary
로 존재하게 된다.

이제 drbd 의 역할 전환 수동 방식에 대해서는 위에서 자세히 설명하였다.
단계를 보면 ..

A. umount test01 drbd filesystem
B. apply test01 "drbdadm secondary all"
C. mount test02 drbd filesystem mounting
D. applay test02 "drbdadmin primary all"

위와 같다.

heartbeat 의 working logic 과 위 과정을 자동화 해줄수 있는 scripts 로 drbd 를
ha 할수 있다.

정확한 작동을 위해서 ha 에 필요한 scripts 를 /etc/rc.d/init.d 밑에 copy 한다.

[root@test01 rc3.d]# cd /etc/ha.d/resource.d/
[root@test01 resource.d]# cp Filesystem drbddisk /etc/rc.d/init.d/

node reboot 시에도 이상 없이 작동 할 수 있게 heartbeat 와 drbd 를 booting 시 자동으로
실행하게 설정한다.

```
[root@test01 rc3.d]# cd /etc/rc.d/rc3.d/
[root@test01 rc3.d]# ln -s /etc/rc.d/init.d/drbd S70drbd
```

이로써 DRBD + heartbeat를 이용한 클러스터 파일 시스템 환경이 준비가 되었다.

4.3. 고 가용성 웹 서버 구축 하기

여기서는 실무에서 흔히 사용되는 apache로 구성된 인터넷 시스템을 고 가용성 시스템으로 구축하는 방법이다. 이는 호스팅 업체나 실제 인터넷 서비스를 운영하는 사이트에서 대표적으로 사용하는 방법이다.

test01, test02 Node 에 apache + mysql + php + tomcat 을 설치한다.
설치 관련 자료는 클루닉스 기술부 2차 세미나 자료를 참고 한다.

httpd.conf 에서 ServerName , NameVirtualHost 를 Heartbeat 의 haresources 에 정의된 service ip address 로 지정한다.

DocumentRoot 는 drbd device 에 mount 된 /data1 아래에 생성하도록 한다.

test01, test02 node 의 apache init script 를 /etc/rc.d/init.d 밑에 복사한다.

```
[root@test01 root]# cp /usr/local/apache/bin/apachectl /etc/rc.d/init.d/
[root@test02 root]# cp /usr/local/apache/bin/apachectl /etc/rc.d/init.d/
```

heartbeat 의 haresources 설정에 apache init script 를 추가한다.

```
[root@test01 root]# cd /etc/ha.d
[root@test01 ha.d]# vi haresources
--------------------------------------------------------------------------------

test01 192.168.123.170 drbddisk Filesystem::/dev/drbd0::/data1::reiserfs ₩
  mysqld apachectl
```

---

haresources 를 test02 node 에도 copy 한다.

test01, test02 node 의 heartbeat 를 다시 시작한다.

4.4. 고 가용성 Mysql DB 서버 구축 하기

mysql 을 DRBD 에 적용하는 방법에 대해 간단히 설명한다.

일단 mysql 을 클루닉스 기술부 2차 세미나 방식으로 설치 했을 경우에는 mysql db file 이 /usr/local/mysql/data 밑에 설치 되게 된다. 이 db file path를 DRBD 가 적용되는 /data1 및에 path 로 변경해 주어야 한다.

먼저 실행 중인 mysql daemon 을 stop 시킨다.

[root@test01 data1]# vi /etc/rc.d/init.d/mysqld stop

기존에 생성된 기본 DB file 을 모두 DRBD 가 적용된 volum 으로 Copy 한다.

[root@test01 data1]# cp -a /usr/local/mysql/data /data1/mysqldb
[root@test01 data1]# chown mysql. mysqldb

그런후 mysql init script에서 --datadir 의 부분을 변경해 준다.

[root@test01 data1]# vi /etc/rc.d/init.d/mysqld

...

/// --datadir 로 해당 문자열을 찾는다 ///

...

$bindir/mysqld_safe --datadir=/data1/mysqldb --pid-file=$pid_file --language=korean ₩
--safe-show-database >/dev/null 2>&1 &

..

[root@test01 data1]# vi /etc/rc.d/init.d/mysqld start

이로써 DRBD를 이용한 mysql database HA 가 구현된다.

4.5. 고 가용성 Oracle DB 서버 구축하기

여기서는 Oracle 9i 이상에서 제공하는 RAC를 이용하지 않고 Linux 운영체제에서 제공하는 기능만으로 고가의 고 가용성 Oracle HA 시스템 구축을 순수 기술만으로 구현하는 방법에 대해 알아 보도록 한다.

4.5.1 DRBD 를 이용한 HA 구성으로 Oracle 설치 시 유의할 점

oracle 을 DRBD 로 HA 설계를 하기 위해서는 몇 가지 초기 Oracle 시스템 설계 시 고려 해야 할 사항이 있다. Oracle DB 의 구동 방식은 Mysql 가 같은 file 위주 형식의 작동 방식이 아니다.

즉 시스템의 file data 에 관련된 설정과 oracle 내부 엔진에 관련된 설정들이 자체 dba db tables 에서 관리 하도록 구성이 되어져 있기 때문에 DB 의 주요 구성 요소의 정보를 HA 백업을 시킬 노드에도 동일하게 적용이 되어져 있어야 한다.

만일 실제 DB file 의 내용과 시스템의 구조, Oracle의 내부 구성등의 정보가 하나라도 일치 하지 않으면 정상적으로 oracle daemon 을 start 시킬 수 없다.

이렇기에 Oracle을 설치 시 control file, db file, log file, paramate file 등의 경로 및 정보들을 모두 DRBD 가 작동하는 volum 에 위치 시켜야 한다.

Oracle DB 의 File structure 에 대해 이해만 하고 있으면 초기 설치 시에 HA에 필요한 file 의 경로를 DRBD Volum 으로 target 을 정해 생성할 수 있다.

하지만 기존에 이미 만들어진 DB 인 경우는 수동으로 data path를 모두 수정해야 할것이다.

그럼 Oracle 설정 시 필요한 파일에 대해 알아 보도록 하자

* control db file (default path : {ORACLE_BASE}/oradata)
* data db file (non system db, system db) (default path : {ORACLE_BASE}/oradata)
* redo log db file (default path : {ORACLE_BASE}/oradata)
* spfile<SID>.ora (default path : {ORACLE_HOME}/dbs)
* orapw<SID> (default path : {ORACLE_HOME}/dba)

위 DB 파일들은 초기 DB 생성 프로그램인 dbca 를 통해 DB 생성 시 위치를 정해 줄 수도 있고 DB Create scripts 수정 등을 통해서 변경 할 수도 있다.

spfile<SID>.ora 파일은 서버 초기화 파일로 db file 의 기본 정보 및 변경된 정보 기타 parameter 정보를 관리하는 파일이다. oracle 8i 에서는 pfile 이라 했는데 oracle 9i부터 pfile 과 spfile 두 가지 방식으로 관리가 가능하다.

orapw<SID> 파일은 Oracle에 sysdba 권한으로 접속을 할 수 있는 계정을 관리 하는 파일이다. 만일 이 문제로 oracle db mount 시 error 가 발생 할때는 그냥 primary 시스템의 orapw<SID> 파일을 옮겨 그대로 사용하면 된다.

위에서 언급한 내용으로 단순 active / standby 형식의 oracle HA 구성이 가능하다.
하지만 oracle multi instance 구성으로 두 대의 노드가 각 instance 의 both HA 을 구성할 때는 oracle 의 file architecture 뿐만 아니라 process, log architecture 즉 .. Oracle Instance Structure 에 대해 이해를 해야 한다.
이 문서는 Oracle 을 설명하는 문서가 아니니 oracle 에 관련된 사항들은 개인적인 노력이 필요할 것이다.

4.5.2 Oracle File Data Structure 설계 시 고려 해야 할 사항

Control File, Redo Log File, Data File
분류 시 고려해야 할 사항들

Control Files
- Control File은 크기가 작으므로 중요하므로 여러 디스크에 저장 되면 좋다.
- Control File은 디스크 마다 이름이 같은 것이 좋다. (추 후 문제시 복구 용이)

Redo Log Files
- 그룹 단위이며 한 그룹에는 적어도 2개 이상의 Redo Log File(Member)이 있어야 한다.
- 같은 그룹에 속한 Redo Log File 들은 다른 디스크에 저장되는 것이 좋다.
- Datafile 과 Redo Log File은 같은 디스크에 위치하지 않도록 한다.
  (복구 시 필요하므로 따로 둔다 - 같이 오류,에러시 복구 불가능)

Data Files
- System, Undo, Temporary 및 Oracle 디폴트 제공 테이블 스페이스 영역은 같은 디스크에 존재하는 것이 좋다.
- System, Undo, Temporary 등 테이블 스페이스 영역은 Oracle이 사용하므로 사용자 영역과 다른 디스크에 위치하는 것이 좋다.
- 같은 테이블 스페이스에 존재하는 Data File도 다른 디스크에 위치하게 만드는 것이 좋다.
- Datafile 과 Redo Log File은 같은 디스크에 위치하지 않도록 한다.
  (복구시 필요하므로 따로 둔다 - 같이 오류, 에러시 복구 불가능)

예) disk1 ~ disk5까지의 디스크가 존재한다.
각각의 디스크에 Control File, Reod Log File, Data File을 배치하라.
단, 사용자 Tablespace는 happy 한 개 존재한다.

DISK 1
---
control.ctl
happy01.dbf

DISK 2
---
control.ctl
happy02.dbf

DISK 3
---
control.ctl
system01.dbf, undotbs01.dbf, temp01.dbf, cwlite01.dbf, drsys01.dbf,
example01.dbf, indx01.dbf, odm01.dbf, tools01.dbf, users01.dbf, xdb01.dbf
---
DISK 4
---
control.ctl
redo01.log, redo02.log, redo03.log

DISK 5
---
control.ctl
redo01.log, redo02.log, redo03.log

Oracle 성능을 위해서는 위 사항을 충분히 고려 해서 설계하는 것이 좋을 것이다. 하지만 HA 설계 시에는 Data File Managing Point 가 많아 진다는 단점이 있을 것 이다. 이는 성능과 관리의 상반된 관점에서의 설계자가 결정해야 할 문제라 생각한다.

4.5.3 Oracle HA Migration 을 위한 Oracle File Data Control 하기

Oracle Install 단계에서 미리 HA를 고려한 DB Structure 로 구성을 했다면 필요 없겠지만.. 이미 서비스를 하고 있는 Oracle 을 HA 구성으로 변경할 경우 반드시 알아야 하는 기본 Oracle DBA 기술이기에 간단히 설명하겠다.

기본적으로 Oracle DB 를 생성했다면 {ORACLE_BASE}/oradata 밑에 아래와 같은 DB file 이 있을 것이다.

---------- control file
control01.ctl
control02.ctl
control03.ctl
---------- data file
----------non-system data file
cwmlite01.dbf
drsys01.dbf
example01.dbf
indx01.dbf
odm01.dbf
tools01.dbf
users01.dbf
xdb01.dbf
oem_repository.dbf
----------system data file
system01.dbf
undotbs01.dbf
----------temporary data file
temp01.dbf
---------- log file
redo01.log
redo02.log
redo03.log

여기에는 크게 control file, data file, log file 로 분류를 할수 있다.
여기서 data file 의 경우에는 다시 system data file, non-system data file,
temporary data file 로 나눌 수 있다.

각각의 파일 별로 데이터 경로를 변경하는 방법이 조금씩 다르다.

+ Control File Data Path Modify

기본적인 Control file 위치는 초기화 파라메터 파일 에서 확인 할수 있다.

$ORACLE_HOME/dbs/init<SID>.ora 파일 안에 아래과 같이 존재

```
control_files=("/oracle/oradata/disk3/control.ctl",
               "/oracle/oradata/disk4/control.ctl",
               "/oracle/oradata/disk5/control.ctl")
```

Oracle 9i 에서는 기본적으로 init<SID>.ora 파일이 없을 것이다.
초기 설치 시에는 관련 정보가 모두 server parameter 파일인 spfile<SID>.ora 에 저장이 되기 때문에 별도로 초기화 파라메터 파일을 생성해야 한다.

생성 구문은 다음과 같다.

```
$ sqlplus /nolog
SQL> conn / as sysdba
SQL> create pfile from spfile;
SQL> shutdown normal
```

그럼 {ORACLE_HOME}/dbs 밑에 init<SID>.ora 파일이 생성이 되어져 있을 것이다.
이제 각 control file 의 현재 위치가 확인 되면 옮겨 보도록 하자

먼저 init<SID>.ora 파일의 열어서 control_file 의 control file 경로를 옮길 위치로 변경을 한다.

그런 후 기존의 control0X.ctl file 을 OS 상에서 물리적으로 옮길 경로로 copy 한다.

그런 후 DB 를 변경된 pfile 로 start 시킨다.

```
$ sqlplus /nolog
SQL> conn / as sysdba
SQL> startup pfile='{ORACLE_HOME}/dbs/init<SID>.ora'
```

-- 수정된 정보 확인

SQL> col NAME format a35
SQL> select * from v$controlfile;

정상적으로 변경이 완료되면 기존 control file 을 삭제한다.

-- 변화된 정보를 spfile 에 새로 적용 시킨다.
기존 spfile 을 잠시 다른곳을 백업해 두고 지운다.

SQL> create spfile from pfile;

이로써 control file 변경이 완료된다.

+ DB DATA File Path Modify

DB DATA file 은 system file, non-system file, temp file 로 나누어진다.

-- system file 변경 방법

- Database가 Mount 단계이어야 한다.
- 이동하고자하는곳에 이동하고자 하는 Data File이 존재해야 한다.
- 4 번은 Control File을 갱신시키는 역할을 한다.
- GAD는 Global Area Database 명이다.

1. Oracle을 shutdown 시킨다.

2. startup mount상태로 만든다.

3. !mv $ORACLE_BASE/oradata/<GAD>/system01.dbf
$NEW_VOL/oradata/<GAD>/system01.dbf

4. ALTER DATABASE RENAME
FILE '$ORACLE_BASE/oradata/<SID명>/system01.dbf'
TO '$NEW_VOL/oradata/<SID명>/system01.dbf'

5. alter database open;

-- non system file 변경 방법

- Tablespace는 offline 이어야한다.
- 이동하고자하는곳에 이동하고자 하는 Data File이 존재해야 한다.
- 4 번은 Control File을 갱신시키는 역할을 한다.
- GAD는 Global Area Database 명이다.

1. startup 상태에서 수행한다.

2. alter tablespace <tablespace명> offline; (mount 상태과 같다)

3. SQL> !mv $ORACLE_BASE/oradata/<GAD>/userdata01.dbf
$NEW_VOL/oradata/<GAD>/userdata01.dbf

4. ALTER TABLESPACE userdata RENAME
DATAFILE '$ORACLE_BASE/oradata/<GAD>/userdata01.dbf'
TO '$NEW_VOL/oradata/<GAD>/userdata01.dbf';

5. alter tablespace <tablespace명> online;

+ Temporary File Path Modify

Temporary Tablespace 관리시 주의사항

Oracle 설치 시의 temp 테이블 스페이스는 Default Temporary Tablespace는
삭제되지 않는다. 이동 할수도 없다. 그렇기에 새로운 Temporary 를 원하는 위치에

만들고, 새로운 Temporary tablespace 를 default temporary 로 지정한 후 기존적을 삭제해 버리면 된다.

Temporary Tablespace 생성
---
temp1라는 이름(논리적인 이름)으로
'$ORACLE_BASE/oradata/<SID명>/temp1.dbf' 라는 (물리적) 위치에 생성 할 경우

-> 기본적으로 Oracle 생성 시 temp 라는 이름으로 기본 temporary 가 생성 됨..

```
CREATE TEMPORARY TABLESPACE temp1
TEMPFILE '$NEW_VOL/oradata/<SID명>/temp1.dbf' SIZE 20M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4M;
```

Default Temporary Tablespace로 바꾸기
---
temp1 라는 Temporary Tablespace를 생성 후 다음을 실행한다.

```
ALTER DATABASE
DEFAULT TEMPORARY TABLESPACE temp1;
```

Temporary Tablespace 삭제

```
DROP TABLESPACE temp
INCLUDING CONTENTS AND DATAFILES;
```

+ Log File Path Modify

Redo Log File 이름 바꾸기 및 이동 시키기

---

a. Database를 shutdown 한다.

b. 옮길 위치에 Redo Log File을 복사한다.

```
!cp ${ORACLE_BASE}/oradata/redo01.log
    ${NEW_VOL}/oradata/redo01.log
```

c. Database를 MOUNT 모드로 실행시킨다.

d. 다음의 명령어를 실행한다.

```
ALTER DATABASE RENAME FILE
'${ORACLE_BASE}/oradata/redo01.log'
TO '${NEW_VOL}/oradata/redo01.log';
```

e. Database를 Open 모드로 변경한다.

```
ALTER DATABASE open;
```

4.5.4 Oracle 과 Web Program 연동하기

Oracle을 웹서버에 연동하는 방법으로 크게 Jsp 환경과 PHP 환경이 있다.
JSP 환경은 JDBC 드라이브를 설치하면 간단히 된다. 여기서는 PHP 연동에 대해
간단히 알아보도록 하겠다.

그냥 기존 APM 연동 시 PHP configure option 중 마지막에 아래 구문을 추가한다.

```
--with-oci8=${ORACLE_HOME}
```

4.5.5 Oracle HA를 위해 필요한 스크립트

Oracle HA 를 위해서는 Oracle Daemon 을 자동으로 실행 시켜 주는 start script 와 shutdown script 그리고 이를 제어 할수 있는 init script 가 필요하다.

이 3가지 script 를 이용하여 heartbeat 의 적용 하면 된다. 물론 앞에서 설명한 DRBD 설정 역시 포함이 되어야 한다. 일단 여기서는 Heartbeat 에서 사용될
3개의 스크립트와 1개의 oracle profile 에 대해 알아보자

먼저 heartbeat 의 구동이 root 상에서 이루어 진다는 것을 알아야 한다.
그리고 대부분의 Oracle Profile 은 oracle 계정에만 적용하도록 셋팅을 한다.

Oracle을 구동을 하기 위해서는 반드시 oracle profile 설정이 필요하다.

oracle dba 계정이 아닌 root 가 oracle 을 start 시키기 위해서는 root 역시 oracle profile 정보를 가지고 있어야 한다. 그냥 root의 .bash_profile 에 정보를 입력해도 되지만, 구동 시키는 console 에서만 oracle profile 이 적용 되게 하는 방법으로 oracle init scripts 를 만들었다. - 이게 FM 이다.

[root@test01 root]# cat /etc/sysconfig/oracle

```
# Oracle Environment
ORACLE_BASE=/oracle/app
ORACLE_HOME=$ORACLE_BASE/product/9.2.0
ORACLE_TERM=xterm
TNS_ADMIN=$ORACLE_HOME/network/admin
NLS_LANG=AMERICAN_AMERICA.KO16KSC5601
ORA_NLS33=$ORACLE_HOME/ocommon/nls/admin/data
PATH=$PATH:$ORACLE_HOME/bin
export PATH ORACLE_BASE ORACLE_HOME ORACLE_TERM TNS_ADMIN NLS_LANG ORA_NLS33
```

// Oracle startup , shutdown Script //

[root@test01 root]# cat /etc/rc.d/init.d/ora_start.sql

```
connect sys/sys@TEST01 as sysdba
startup
quit
-------------------------------------------------------------------------------
        --> ㅋㅋ 민망하군..  간단히 설명하면
        connect  <user_id>/<password>@DB명 as sysdba
        이런 형식으로 사용하면 된다. 접속 이후필요한 여러 DBA 명령을 이곳에
        상황에서 맞게 만들어 주면 된다.

[root@test01 root]# cat /etc/rc.d/init.d/ora_shutdown.sql
-------------------------------------------------------------------------------
connect sys/sys@TEST01 as sysdba
shutdown
quit
-------------------------------------------------------------------------------

// Oracle Init Scripts //

[root@test01 root]# cat /etc/rc.d/init.d/oractl
-------------------------------------------------------------------------------
#!/bin/sh

. /etc/init.d/functions

if [ -f "/etc/sysconfig/oracle" ]; then
                source /etc/sysconfig/oracle
fi

RETVAL=0

start () {
        echo -n $"Starting Daemon of Oracle : "
        if [ -f "/etc/sysconfig/oracle" ]; then
                . /etc/sysconfig/oracle
                echo $"(Starting .....)"
```

```
                sqlplus /nolog @/etc/rc.d/init.d/ora_start.sql

        else
                echo $"(No oracle config file)"
                exit 0
        fi

        RETVAL=$?
        echo
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/oracle

        }

stop () {
        echo -n $"Shutdown Daemon of Oracle: "
        sqlplus /nolog @/etc/rc.d/init.d/ora_shutdown.sql

        RETVAL=$?
         echo
        [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/oracle
}

case "$1" in

  start)
        start
        ;;

  stop)
        stop
        ;;

  restart)
        stop
        start
```

```
        RETVAL=$?
        ;;

  *)

        echo $"Usage: $0 {start|stop|restart}"
        exit 1
esac
exit $RETVAL
--------------------------------------------------------------------------------
```

핵심만 설명하면 아래 구문으로 Oracle 내부 scripts 파일을 include 할수 있다.

sqlplus /nolog @/etc/rc.d/init.d/ora_start.sql

4.5.6 DRBD + Heartbeart 를 이용하여 Oracle HA 하기

a. DRBD 설치한다.

b. Heartbeat를 설치한다.

c. Node01, Node02 Oracle 설치
(File Data Path, File Data Name, SID, DB_name, Oracle 환경 설정 등을 모두 동일하게)

d. oracle File Data를 모두 DRBD Volume 상의 디렉토리로 이동한다.

e. oracle Control File 상의 Data Path 정보를 모두 DRBD 볼륨상의 경로로 변경
(설치 시 HA 설계를 고려 하여 경로 지정 권장)

f. DRBD 설정
설정이 완료된 이후 drbd device로 file system 을 새로 만들어야 한다. 그렇기 때문에 Oracle DATA 를 모두 다른곳에 빽업 한 후 mkreiserfs /dev/drbd0 등으로 filesystem 생성 후 다시 그
위치로 Data File 을 옮겨 놓는다.

g. Heartbeat 설정

[root@test01 root]# vi /etc/ha.d/ha/haresources

---

test01 192.168.123.170 drbddisk::drbd0 Filesystem::/dev/drbd0::/data1::reiserfs oractl

---

h. Oracle DBA 인증 파일 Sync

${ORACLE_HOME}/dbs/orapw<SID> 파일을 양 노드 모두 동일하게 해주어야 한다.
Primary Node 의 ${ORACLE_HOME}/dbs/orapw<SID> 파일을 Secondary 에 복사해 준다.

4.5.7 두 개의 Instance를 이용한 Oracle Both HA 구축하기

* 시스템 디자인 :

test01, test02 노드에 각각 TEST01, TEST02 라는 Oracle DB 를 별도의 Instance로 운영을 하고자 한다. test01 노드에 문제 발생 시 test02 노드에서 test01 의 TEST01 DB를 HA 하고 test02
노드에서 문제 발생 시는 test01 에서 TEST02 노드를 HA 하여 서비스의 고 가용성을 추구한다.

- DRBD 설계 다자인 :

test01, test02 노드에 각각 /data1, /data2 라는 물리적 볼륨을 만든다.
/data1 볼륨은 test01 에선 primary, test02 에서는 secondary 로 설정
/data2 볼륨은 test01 에선 secondary, test02 에서는 primary 로 설정

test01 에서는 /data1 에 oracle file data 가 위치 하도록
test02 에서는 /data2 에 oracle file data 가 위치 하도록 설계한다.

-- drbd config

[root@test01 root]# vi /etc/drbd.conf

```
----------------------------------------------------------------------------------------
skip {
   As you can see, you can also comment chunks of text
   with a 'skip[optional nonsense]{ skipped text }' section.
   This comes in handy, if you just want to comment out
   some 'resource <some name> {...}' section:
   just precede it with 'skip'.

   The basic format of option assignment is
   <option name><linear whitespace><value>;

   It should be obvious from the examples below,
   but if you really care to know the details:

   <option name> :=
           valid options in the respective scope
   <value>   := <num>|<string>|<choice>|...
                depending on the set of allowed values
                for the respective option.
   <num>     := [0-9]+, sometimes with an optional suffix of K,M,G
   <string> := (<name>|₩"([^₩"₩₩₩n]*|₩₩.)*₩")+
   <name>    := [/_.A-Za-z0-9-]+
}


resource drbd0 {
   protocol C;
   incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
   startup {
      degr-wfc-timeout 120;
   }
```

```
disk {
   on-io-error     detach;
}

net {
   # sndbuf-size 512k;
   # timeout           60;      #   6 seconds   (unit = 0.1 seconds)
   # connect-int     10;      # 10 seconds   (unit = 1 second)
   # ping-int          10;      # 10 seconds   (unit = 1 second)
   # max-buffers        2048;
   # max-epoch-size    2048;
   # ko-count 4;
   # on-disconnect reconnect;

}

syncer {
   rate 10M;
   group 1;
   al-extents 257;
}

on test01 {
   device        /dev/drbd0;
   disk          /dev/sda8;
   address       192.168.123.165:7788;
   meta-disk     internal;
}

on test02 {
   device       /dev/drbd0;
   disk         /dev/sda8;
   address      192.168.123.166:7788;
   meta-disk internal;
```

```
  }
}

resource drbd1 {

  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
  startup {
    degr-wfc-timeout 120;      # 2 minutes.
  }


  disk {
    on-io-error     detach;
  }

  net {
  }

  syncer {
    rate 10M;
    group 0;
    al-extents 257;
  }

  on test01 {
    device        /dev/drbd1;
    disk          /dev/sda9;
    address       192.168.123.165:7799;
    meta-disk     internal;

  }

  on test02 {
    device       /dev/drbd1;
```

```
    disk        /dev/sda9;
    address     192.168.123.166:7799;
    meta-disk internal;
  }

}
~
```

- HA 설계 디자인 :

test01 노드가 Fail 시 test02 노드에서는 /data1 volum 의 Drbd status 를 Secondary 에서 primary 로 전환하고 /data1 을 mount 시킨다. 그런 후 TEST01 Oracle Instance Start script 를 실행한다.

test02 노드가 fail 시엔 test01 노드에서 /data2 volum 의 Drbd status 를 secondary 에서 primary 로 전환하고 /data2 를 mount 한다. 그리고 TEST02 Oracle Instance start script 를 실행한다.

[root@test01 root]# vi /etc/ha.d/haresources

```
test01 192.168.123.170 drbddisk::drbd0 Filesystem::/dev/drbd0::/data1::reiserfs oractl
test02 192.168.123.171 drbddisk::drbd1 Filesystem::/dev/drbd1::/data2::reiserfs oractl1
```

[root@test01 root]# vi /etc/rc.d/init.d/drbddisk

```
#!/bin/bash
#
# This script is inteded to be used as resource script by heartbeat
#
# Jan 2003 by Philipp Reisner.
#
###
```

```
DEFAULTFILE="/etc/default/drbd"
DRBDADM="/sbin/drbdadm"

if [ -f $DEFAULTFILE ]; then
    . $DEFAULTFILE
fi

if [ "$#" -eq 2 ]; then
    RES="$1"
    CMD="$2"
else
    RES="all"
    CMD="$1"
fi

case "$CMD" in
        start)
                $DRBDADM primary $RES
                ;;
        stop)
                $DRBDADM secondary $RES
                ;;
        status)
                if [ "$RES" = "all" ]; then
                        echo "A resource name is required for status inquiries."
                        exit 10
                fi
                ST=$( $DRBDADM state $RES 2> /dev/null )
                ST=${ST%/*}
                if [ "$ST" = "Primary" ]; then
                        echo "running"
                else
                        echo "stopped"
                fi
                ;;
```

```
    *)
        echo "Usage: drbddisk [resource] {start|stop|status}"
        exit 1
        ;;
esac

exit 0
```

[root@test01 root]# vi /etc/rc.d/init.d/Filesystem

```
#!/bin/sh

unset LC_ALL; export LC_ALL
unset LANGUAGE; export LANGUAGE

prefix=/usr
exec_prefix=/usr
#. /etc/ha.d/shellfuncs
. /etc/ha.d/shellfuncs

# Utilities used by this script
MODPROBE=/sbin/modprobe
FSCK=/sbin/fsck
FUSER=/sbin/fuser
MOUNT=/bin/mount
UMOUNT=/bin/umount
BLOCKDEV=/sbin/blockdev

check_util () {
    if [ ! -x "$1" ] ; then
        ha_log "ERROR: setup problem: Couldn't find utility $1"
        exit 1
    fi
}
```

```
usage() {

cat <<-EOT;
        usage: $0 <device> <directory> <fstype> [<options>] {start|stop|status}

        <device>      : name of block device for the filesystem. e.g. /dev/sda1, /dev/md0
                        OR -LFileSystemLabel OR -Uuuid or an NFS specification
        <directory> : the mount point for the filesystem
        <fstype>      : name of the filesystem type. e.g. ext2
        <options>     : options to be given as -o options to mount.

        $Id: Filesystem.in,v 1.10 2003/07/03 02:14:14 alan Exp $
        EOT
}

#
#       Make sure the kernel does the right thing with the FS buffers
#       This function should be called after unmounting and before mounting
#       It may not be necessary in 2.4 and later kernels, but it shouldn't hurt
#       anything either...
#
#       It's really a bug that you have to do this at all...
#
flushbufs() {
  if
    [ "$BLOCKDEV" != "" -a -x "$BLOCKDEV" ]
  then
    case $1 in
      -*|[^/]*:/*)      ;;
      *)                $BLOCKDEV --flushbufs $1;;
    esac
  fi
}
# Check the arguments passed to this script
```


클루닉스 서 진우(alang@clunix.com)

```
DEVICE=$1
MOUNTPOINT=$2
FSTYPE=$3

case $DEVICE in
  -*) # Oh... An option to mount instead...  Typically -U or -L
        ;;
  [^/]*:/*)       # An NFS filesystem specification...
        ;;
  *)    if [ ! -b "$DEVICE" ] ; then
          ha_log "ERROR: Couldn't find device $DEVICE. Expected /dev/??? to exist"
          usage
          exit 1
        fi;;
esac

if [ ! -d "$MOUNTPOINT" ] ; then
        ha_log "ERROR: Couldn't find directory  $MOUNTPOINT to use as a mount point"
        usage
        exit 1
fi


# Check to make sure the utilites are found
check_util $MODPROBE
check_util $FSCK
check_util $FUSER
check_util $MOUNT
check_util $UMOUNT

case $# in
  4)    operation=$4; options="";;
  5)    operation=$5; options="-o $4";;
  *)    usage; exit 1;;
esac
```

```
# Look for the 'start', 'stop' or status argument
case "$operation" in

#
# START: Start up the filesystem
#
start)

        # See if the device is already mounted.
        $MOUNT | cut -d' ' -f3 | grep -e "^$MOUNTPOINT$" >/dev/null
        if [ $? -ne 1 ] ; then
            ha_log "ERROR: Filesystem $MOUNTPOINT is already mounted!"
            exit 1;
        fi

        # Insert SCSI module
        $MODPROBE scsi_hostadapter >/dev/null 2>&1

        # Insert Filesystem module
        $MODPROBE $FSTYPE >/dev/null 2>&1
        grep -e "$FSTYPE"'$' /proc/filesystems >/dev/null
        if [ $? != 0   ] ; then
                ha_log "ERROR: Couldn't find filesystem $FSTYPE in /proc/filesystems"
                usage
                exit 1
        fi

        # Check the filesystem & auto repair.
        # NOTE: Some filesystem types don't need this step...  Please modify
        #           accordingly

        if
          case $FSTYPE in
```


클루닉스 서 진우(alang@clunix.com)

```
            ext3|reiserfs|xfs|jfs|vfat|fat|nfs) false;;
            *)                                  true;;
          esac
        then
          ha_log "info: Starting filesystem check on $DEVICE"
          $FSCK -t $FSTYPE -a $DEVICE

          # NOTE: if any errors at all are detected, it returns non-zero
          # if the error is >4 then there is a big problem
          if
            [ $? -ge 4 ]
          then
            ha_log "ERROR: Couldn't sucessfully fsck filesystem for $DEVICE"
            exit 1
          fi
        fi

        flushbufs $DEVICE
        # Mount the filesystem.
        if
          $MOUNT -t $FSTYPE $options $DEVICE $MOUNTPOINT
        then
          : Mount worked!
        else
          ha_log "ERROR: Couldn't mount filesystem $DEVICE on $MOUNTPOINT"
          exit 1
        fi

# end of start)
;;

#
# STOP: Unmount the filesystem
#
stop)
```


클루닉스 서 진우(alang@clunix.com)

```
        # See if the device is currently mounted
        if
          $MOUNT | grep -e " on $MOUNTPOINT " >/dev/null
        then
                # Kill all processes open on filesystem
                $FUSER -mk $MOUNTPOINT

                # Get the current real device name...
                # (specified devname could be -L or -U...)
                DEV=`$MOUNT | grep "on $MOUNTPOINT " | cut -d' ' -f1`
                # Unmount the filesystem
                $UMOUNT $MOUNTPOINT
                if [ $? -ne 0 ] ; then
                        ha_log "ERROR: Couldn't unmount $MOUNTPOINT"
                        exit 1
                fi
                flushbufs $DEV
        else
                ha_log "WARNING: Filesystem $MOUNTPOINT not mounted?"
        fi

# end of stop)
;;

#
# STATUS: is the filesystem mounted or not?
#
status)

        $MOUNT | grep -e "on $MOUNTPOINT " >/dev/null
        if [ $? = 0 ] ; then
                echo "$MOUNTPOINT is mounted (running)"
        else
                echo "$MOUNTPOINT is unmounted (stopped)"
```


클루닉스 서 진우(alang@clunix.com)

```
            fi
# end of status)
;;


*)
      echo "This script should be run with a fourth argument of 'start', 'stop', or 'status'"
      usage
      exit 1
;;

esac

# If you got to this point, chances are everything is O.K.
exit 0;
```
