# DRBD

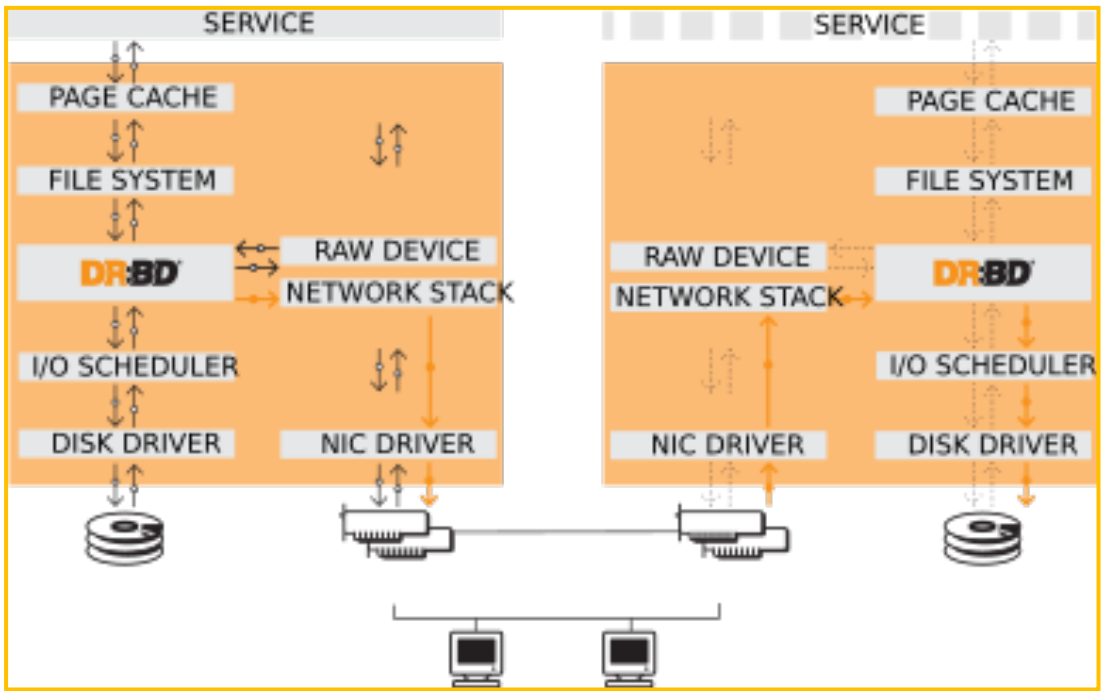
# (Distributed Replicated Block Device)

| | |
|---|---|
| Author | 윤이상 |
| Creation Date | 2019-03-01 |
| Last Updated | 2019-03-01 |
| Version | 1.0 |
|  | |

| Version | 변경일자 | 변경자(작성자) | 주요내용 |
|---|---|---|---|
| 1 | 2019-03-01 | 윤이상 | 문서 최초 작성 |
| 2 | | | |
| 3 | | | |

Contents

# 1. DRBD Introduce




- DRBD(Distributed Replicated Block Device)는, TCP/IP 네트워크를 통해서, 복수의 서버간의 Disk(파티션)를 Mirroring (Raid 1) 하는 소프트웨어입니다.
- 네트워크를 통해서 Raid1의 환경을 구축하는 것이 가능합니다.
- Raid 1 의 일반적인 구성은 데이터 백업(실시간) Mirroring 하는 구성이지만 여기서는 Network를 통해 Mirroring을 하게 됩니다.
- 보통 DRBD 는 HA-Cluster 시스템의 Heartbeat 를 통해 제어되는 것이 일반적이며, 8.0.x 대 버전으로 넘어오면서 OFCS,GFS 등 HA-Cluster 에서의 확장된 기능이 추가되었습니다.

## 1.1. 동작환경

DRBD는 Linux OS상에서 동작하는 소프트웨어입니다. Linux의 커널 모듈이 필요하기 때문에, 같은 유닉스 계 OS인 Open Solaris와 FreeBSD에서는 동작 할 수 없습니다.
현재 패키지가 제공되는 디스트리뷰션은 이하와 같습니다.

- Red Hat Enterprise Linux
- SUSE Linux Enterprise Serve
- Debian GNU/Linux
- Ubuntu Server Edition

패키지가 제공되지 않더라도, 소스코드가 공개되어있기 때문에 Linux 기반이라면 컴파일해서 인스톨하는 것이 가능합니다.

### 1.1.1. 특징

- DRBD 는 2 대의 서버간의 파티션을 네트워크를 통해 복제(Replication)하기 때문에, 같은 스펙 서버가 2 대 이상 필요하게 됩니다.
- 최대 특징은, HDD(파티션)의 복제를 TCP/IP 네트워크를 이용해서 실행하는 것입니다. 네트워크를 이용하는 것으로, 특별한 하드웨어를 필요로 하지 않습니다. 리눅스 환경과 미러링으로 사용하기 위한 네트워크 카드가 있다면, 바로 DRBD 의 기능을 사용할 수 있습니다.
- DRBD 는 파일 시스템보다도 낮은 레이어에서 동작해서, Replication 된 HDD 를 블록 디바이스로써 사용하는 것이 가능합니다. 쉬게 말하면 복제되어 있는 DRBD 디바이스를 보통의 HDD 와 같은 느낌으로 사용하는 것이 가능합니다.
- 보통의 HDD 를 같은 느낌으로 사용하는 것이 가능하기 때문에, 데이터를 보존하는 어플리케이션이 미러링에 대응할 필요는 없습니다. 거의 모든 어플리케이션이 DRBD 에 의해 실시간으로 데이터를 복제해가면서, 동작 가능합니다.
- DRBD 와 Pacemaker 를 결합하면, 간단하고 안전한 고가용성 환경 (HA 클라스터 환경)을 구축하는 것이 가능합니다.
- DRBD 를 사용하지 않는 많은 HA 클러스터 환경은 데이터 영역을 복제 서버에 공유하는 [공유스토리지]를 이용하고 있습니다. 하지만, 공유스토리지에 장애가 발생한 경우에 서비스가 계속 되지 않을 뿐만 아니라, 중요한 데이터의 손실로 연결될 가능성이 있습니다.
- DRBD 는 여러 서버에 네트워크를 통해 미러링을 제공하고 있기 때문에, 공유 스토리지를 사용하지 않습니다.
- DRBD 는 서버 수 감소, 공유 스토리지 단의 단일 장애 지점 (Single point of failer)을 없앨 수 있는, 저비용 기반의 안전한 HA 클러스터 환경을 제안합니다.

## 1.2. 요약

- **특징**
  - 장애 노드 복구 후 Data auto-resynchronization
  - Transactional Storage Engine 지원 가능
  - Shared-Nothing 구조.
  - Failover time 이 30 초 이내
  - Network latency 에 의해 성능이 영향을 받음
  - Write-intensive system 의 경우 부적합
  - HA(High Availability) & TB 단위 데이터 처리
  - SAN Storage 대체 방안
  - Data Redundancy 요구

- **장점**
  - Application Level + H/W Level 장애 감지 가능
  - Pacemaker(Heartbeat)를 통해 auto-failover 지원 가능

## 1.3. 기능

### 1.3.1. Single-primary mode

단일 기본 모드에서 모든 자원은 단 하나의 클러스터 구성에만 primary role 이 있습니다. 하나의 클러스터 노드 만 데이터를 조작 할 수 있으므로, Single-primary mode 는 기존 파일 시스템 (ext3, ext4, XFS 등)과 함께 사용할 수 있습니다.
Single-primary mode 는 고가용성 클러스터를 구성을 위한 DRBD 표준 접근 방식입니다.

### 1.3.2. Dual-primary mode

이 기능은 DRBD 8.0 이상에서 사용할 수 있습니다.
Dual-primary mode 에서 모든 자원은 두 클러스터 노드에 primary role 이 에 있습니다. 데이터에 대한 동시 액세스가 가능하기 때문에이 모드에서는 분산 잠금 관리자를 사용하는 공유 클러스터 파일 시스템을 사용해야합니다. (예) GFS,OCFS2
Dual-primary mode 는 두 노드의 동시 데이터 액세스를 필요로 하는 클러스터 로드 균형 조정에 적합한 방법입니다. 이 모드는 기본적으로 비활성화되어 있으므로 DRBD 의 구성 파일에서 반드시 명시적으로 활성화해야 합니다.

### 1.3.3. Replication modes

DRBD 는 3 가지 복제 모드를 지원하여 3 가지 복제 동시성을 허용합니다.
DRBD 설정에서 가장 일반적으로 사용되는 복제 프로토콜은 프로토콜 C 입니다.

**Protocol A. - 비동기 복제 프로토콜**

primary 노드의 로컬 쓰기 작업은 로컬 디스크 쓰기가 발생하자마자 완료되고, 복제 패킷이 로컬 TCP 송신 버퍼에 저장됩니다. 강제 페일오버의 경우 데이터 손실이 발생할 수 있습니다. 대기 노드의 데이터는 장애 조치 후 일관성이 유지되지만, 충돌 전에 수행된 가장 최근의 업데이트는 손실 될 수 있습니다.

**Protocol B. - 메모리 동기식(반동기식) 복제 프로토콜**

primary 노드의 로컬 쓰기 작업은 로컬 디스크 쓰기가 발생하자마자 완료되어 복제 패킷이 피어 노드에 도달하게 됩니다. 일반적으로, 강제 실패 시 쓰기는 손실되지 않습니다. 그러나, 양쪽 노드에서 동시 정전이 발생하고, primary 노드의 데이터 저장소가 동시에 돌이킬 수 없는 파괴되는 경우, primary 노드의 데이터 저장소에서 완료한 가장 최근의 쓰기가 손실 될 수 있습니다.

**Protocol C. - 동기식 복제 프로토콜**

primary 노드의 로컬 쓰기 작업은 로컬 디스크와 원격 디스크 쓰기가 모두 확인된 후에만 완료됩니다. 결과적으로, 단일 노드의 손실은 데이터 손실을 초래하지 않도록 보장이 됩니다.. 데이터 손실은 물론 두 노드가 동시에 치명적인 문제로 파괴되는 경우 이 복제 프로토콜 또한 데이터 손실이 불가피 합니다.

### 1.3.4. Multiple replication transports

이 기능은 DRBD 8.2.7 이상에서 사용할 수 있습니다.
DRBD 의 복제 및 동기화 프레임 워크 소켓 계층은 여러 하위 수준 전송을 지원합니다.

TCP over IPv4
이것은 표준 구현이며 DRBD 의 기본값입니다. IPv4 가 사용 가능한 모든 시스템에서 사용될 수 있습니다.

TCP over IPv6
복제 및 동기화에 표준 TCP 소켓을 사용하도록 구성된 경우 DRBD 는 네트워크 프로토콜로 IPv6 도 사용할 수 있습니다. 비록 다른 주소 지정 체계를 사용하지만 IPv4 와 의미 및 성능면에서 동일합니다.

SuperSockets
스택의 TCP / IP 부분을 단일 모 놀리 식 고효율 및 RDMA 가능 소켓 구현으로 대체합니다. DRBD 는 매우 낮은 대기 시간 복제에 이 소켓 유형을 사용할 수 있습니다. SuperSockets 는 현재 단일 공급 업체 인 Dolphin Interconnect Solutions 에서 제공하는 특정 하드웨어에서 실행해야만 합니다.

### 1.3.5. Efficient synchronization

복제 링크가 중단 된 경우, 기본 노드의 장애, 2 차 노드의 장애 또는 복제 링크의 중단으로 인해 동기화가 필요합니다. 동기화는 DRBD 가 원래 작성된 순서대로 수정 된 블록을 선형 순서대로 동기화하지 않는다는 점에서 효율적입니다.

- 여러 번의 연속 쓰기 작업이 발생한 블록은 한 번만 동기화되므로 동기화가 빠릅니다.
- 디스크상의 블록 레이아웃에 따라 동기화되므로, 디스크 검색과 관련이 없습니다.
- 동기화 중에 대기 노드의 데이터 세트는 부분적으로 폐기되고 부분적으로 업데이

트됩니다.

### 1.3.6. On-line device verification

이 기능은 DRBD 8.2.5 이상에서 사용할 수 있습니다.

온라인 장치 확인을 통해 사용자는 노드간에 매우 효율적인 방식으로 블록 단위의 데이터 무결성 검사를 수행 할 수 있습니다.

### 1.3.7. Replication traffic integrity checking

이 기능은 DRBD 8.2.0 이상에서 사용할 수 있습니다.

DRBD 는 선택적으로 MD5, SHA-1 또는 CRC-32C 와 같은 암호화 메시지 다이제스트 알고리즘을 사용하여 종단 간 메시지 무결성 검사를 수행합니다.

### 1.3.8. Split brain notification and automatic recovery

DRBD 8.0 및 이후 버전에서 Automatic split brain recovery 가 사용이 가능합니다. split brain 은 클러스터 노드 간의 모든 네트워크 링크가 일시적으로 실패하고 클러스터 관리 소프트웨어 또는 사람의 오류로 인해 두 노드가 연결이 끊긴 상태에서 기본 역할로 전환 된 상황을 의미합니다.

복제되지 않은 상태에서 두 노드 모두에서 데이터 수정이 수행되었음을 의미하므로 잠재적으로 위험한 상태이며 두 개의 데이터 집합이 만들어지게 되어 병할 할 수 없게 됩니다.

이런 문제를 감지하기 위한 통보기능(DRBD 8.2.1 부터 사용가능)과 프로세스 자동화를 복구기능을 사용 할 수 있습니다.

### 1.3.9. Support for disk flushes

하드 드라이브나 RAID 논리 디스크와 같은 로컬 블록 장치가 쓰기 캐싱을 사용할 수 있게되면 이러한 장치에 대한 쓰기 는 휘발성 캐시에 도달하자 마자 완료된 것으로 간주 됩니다. 컨트롤러 제조사는 일반적으로 이것을 write-back 모드라고 부르며 그 반대는 write-through 모드라고 합니다. write-back 모드에서 컨트롤러에서 정전이 발생하면 가장 최근의 보류중인 쓰기 마지막 쓰기가 디스크에 커밋되지 않으므로 데이터가 손실 될 수 있습니다.

이를 방지하기 위해 DRBD 는 디스크 플러시를 사용합니다. 디스크 플러시는 관련 데이터가 안정적 (비 휘발성) 저장소에 커밋 된 경우에만 완료되는 쓰기 작업입니다. 즉, 캐시가 아닌 디스크에 효율적으로 기록됩니다. 이는 정전시에도 데이터 신뢰도를 높일 수 있습니다.

### 1.3.10. Disk error handling strategies

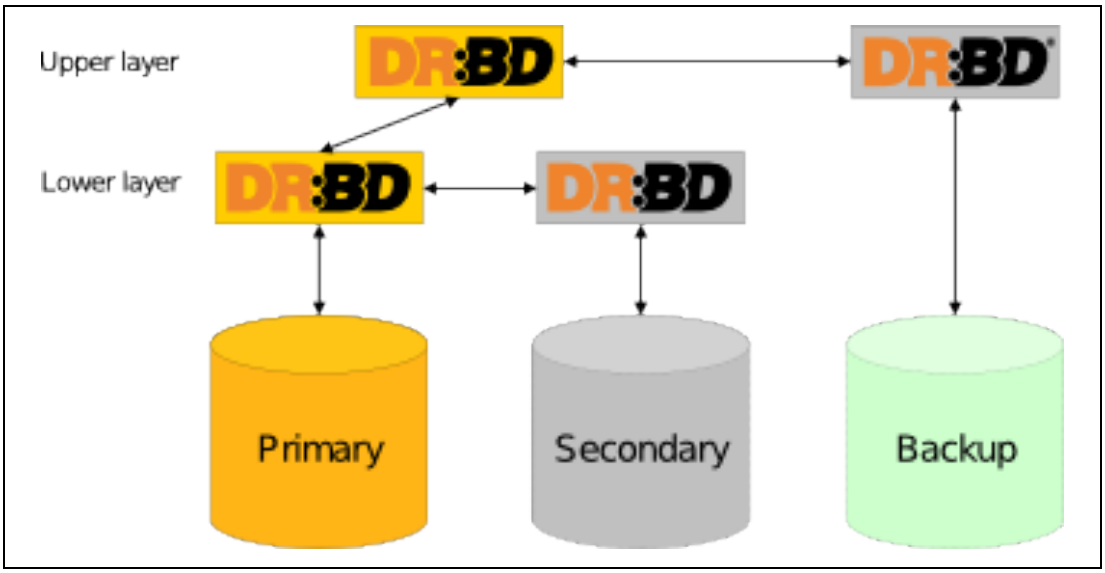
하나의 노드에서 DRBD 의 백업 블록 장치로 사용되는 하드 드라이브가 실패하면 DRBD 는 상위 계층 (일반적으로 파일 시스템)으로 I / O 오류를 전달하거나 I / O 오류를 마스크 할 수 있습니다.

### 1.3.11. Strategies for dealing with outdated data

동기화 대상의 노드 데이터중 일부 데이터의 경우 쓸모없거나, 식별할 수 없는 데이터가 존재하게 됩니다.
이런 불필요한 데이터의 복제 링크가 재설정 될 때마다, 자동으로 백그라운드 동기회가 수행됩니다.
전략적으로 오래된 데이터를 사용하지 못하게 하는 설정 또한 가능합니다.

### 1.3.12. Three-way replication

3 방향 복제를 사용할 때 DRBD 는 기존 2 노드 클러스터에 세 번째 노드를 추가하고 해당 노드에 데이터를 복제하므로 백업 및 재해 복구 용도로 사용할 수 있습니다.




### 1.3.13. Long-distance replication with DRBD Proxy

이 기능은 DRBD 8.2.7 이상에서 사용할 수 있습니다.

DRBD Proxy 는 전달하는 데이터를 압축하고 압축을 풀수 있도록 구성할 수 있으며, 이로 인한 약간의 대기 시간이 늘어날 수 있습니다.
하지만 네트워크 링크의 대역폭에 대한 제한이나 요소에 대한 극복이나, 전송시간의 단축의 장점이 더욱 큰 요소 중 하나입니다.

주의사항으로는 DRBD Proxy 는 오픈소스 라이선스로 사용 하실 수 없으며, 별도의 라이선스를 linbit 에서 제공합니다.

### 1.3.14. Truck based replication

원격 사이트에 스토리지 미디어를 물리적으로 설치하면서 복제 될 데이터로 원격 사이트를 미리 설정하는 수단입니다. 이것은 특히 다음과 같은 경우에 적합합니다.

- 복제 할 데이터의 총량이 상당히 많은 경우(수백 기가 바이트 이상).
- 복제 될 데이터의 예상 변화율이 큰경우.
- 사이트 간 사용 가능한 네트워크 대역폭이 제한적인 경우.

이러한 상황에서 Truck based replication 이 없으면 DRBD 는 매우 긴 초기 장치 동기화 (며칠 또는 몇 주 정도)가 필요합니다. 또한, 원격 사이트에 데이터 시드를 제공하고 초기 동기화 시간을 대폭 단축 할 수 있습니다.

### 1.3.15. Floating peers

이 기능은 DRBD 8.3.2 이상에서 사용할 수 있습니다.
DRBD 피어는 특정 명명된 호스트에 연결되지 않고, 실제 호스트에 연결되지 않은 가상 IP 주소에 바인딩 되어 사용하게 됩니다.
이 구성에서는 DRBD 는 호스트명이 아닌 IP 주소로 피어를 식별합니다.

## 2. DRBD Installation

- DRBD 의 기본 구성은, 2 대의 서버간 하드디스크(파티션)의 복제 환경입니다. 2 대의 서버간, DRBD 전용으로 준비한 네트워크 카드를 사용해서, LAN 케이블에 직접 연결합니다.
- DRBD 로 미러링된 디바이스는 통상의 HDD 와 같이 취급하는 것이 가능하기 때문에, DRBD 디바이스(/dev/drbd0)로 파일 시스템을 작성해서 마운트합니다.
- DRBD 디바이스의 관리는 관리 툴에서 전부 행해지며, 접속의 제어와 디바이스의 상태의 제어를 수행합니다.

- H/W
  1. 동일 성능의 서버 2 대
  2. 미러링에 사용하는 네트워크 카드
  3. LAN 케이블
  4. 하드디스크

- S/W
  1. Linux 계 OS
  2. DRBD 커널모듈
  3. DRBD 유틸리티군(관리 tool)

| 설치 사전 환경 정보 |
| --- |
| OS : CentOS 7.6<br>Linux node1 3.10.0-957.1.3.el7.x86_64<br><br>CPU : Virtual 2 Core<br>Memory : 4G<br>Network Card 2EA<br><br>Public<br>Node1 : 172.40.40.49<br>Node2 : 172.40.40.50<br><br>Physical Disk 2EA<br>/dev/sda : 50G (Base Disk)<br>/dev/sdb : 30G (DRBD Sync Disk) |

## 2.1. DRBD Installation

### 2.1.1. elrepo 저장소 등록

```
shell> rpm -import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
shell> rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-
2.el7.elrepo.noarch.rpm
```

### 2.1.2. selinux 설정 변경

SELINUX 가 활성/해제 상태인지 확인합니다. Enforcing 으로 출력된다면, 활성화 된 상태이므로 아래 명령어를 수행하여 disable 로 변경합니다.

```
shell> setenforce 0
setenforce: SELinux is disabled
```

또는

shell> vi /etc/sysconfig/selinux

...

SELINUX=disabled

### 2.1.3. drbd 패키지 설치

shell> yum install -y drbd84-utils kmod-drbd84

....

| Package | Arch | Version | Repository | Size |
|---|---|---|---|---|
| Installing: | | | | |
| drbd84-utils | x86_64 | 9.6.0-1.el7.elrepo | elrepo | 591 k |
| kernel | x86_64 | 3.10.0-957.1.3.el7 | updates | 48 M |
| kmod-drbd84 | x86_64 | 8.4.11-1.1.el7_6.elrepo | elrepo | 209 k |
| Updating: | | | | |
| selinux-policy-targeted | noarch | 3.13.1-229.el7_6.6 | updates | 6.9 M |
| Updating for dependencies: | | | | |
| libselinux | x86_64 | 2.5-14.1.el7 | base | 162 k |
| libselinux-python | x86_64 | 2.5-14.1.el7 | base | 235 k |
| libselinux-utils | x86_64 | 2.5-14.1.el7 | base | 151 k |
| libsemanage | x86_64 | 2.5-14.el7 | base | 151 k |
| libsemanage-python | x86_64 | 2.5-14.el7 | base | 113 k |
| libsepol | x86_64 | 2.5-10.el7 | base | 297 k |
| linux-firmware | noarch | 20180911-69.git85c5d90.el7 | base | 49 M |
| policycoreutils | x86_64 | 2.5-29.el7 | base | 916 k |
| policycoreutils-python | x86_64 | 2.5-29.el7 | base | 456 k |
| selinux-policy | noarch | 3.13.1-229.el7_6.6 | updates | 483 k |
| setools-libs | x86_64 | 3.3.8-4.el7 | base | 620 k |

Transaction Summary

### 2.1.4. drbd 설치 확인

shell> drbdadm -V

DRBDADM_BUILDTAG=GIT-

hash:W d458166f5f4740625e5ff215f62366aca60ca37bW buildW byW mockbuild@W,W 2018-11-

03W 01:32:40

DRBDADM_API_VERSION=1

DRBD_KERNEL_VERSION_CODE=0x08040b

DRBDADM_VERSION_CODE=0x090600

DRBDADM_VERSION=9.6.0

[특이사항]

modinfo: ERROR: Module drbd not found.

경고메시지가 있을 경우 OS 커널 업데이트를 진행해야 합니다.

가이드 문서상 커널 버전 확인

shell> uname -a

Linux node2 3.10.0-957.1.3.el7.x86_64

## 2.2. DRBD Configration

### 2.2.1. Hostname 변경

[Master - Node1]

```
shell> hostnamectl set-hostname node1
```

[Slave - Node2]

```
shell> hostnamectl set-hostname node2
```

/etc/host 파일 설정

```
shell> vi /etc/hosts
172.40.40.49 node1
172.40.40.50 node2
```

### 2.2.2. 파티션 생성

신규 디스크 추가하여 구축한다는 가정하에, 디스크 전체 영역 할당합니다.

```
shell> fdisk /dev/sdb
....
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p):
Using default response p
Partition number (1-4, default 1):
First sector (2048-62914559, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-62914559, default 62914559):
Using default value 62914559
Partition 1 of type Linux and of size 30 GiB is set

Command (m for help): wq
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

### 2.2.3. 설정파일 생성

DRBD 다음 경로의 구성 파일에서 제어됩니다. /etc/drbd.conf.
일반적으로 이 설정 파일은 아래와 같은 내용이 기본적으로 들어 있으며, 하위 디렉토리의 파일을 생성하거나 직접 기본 Configration 파일을 변경하여도 가능합니다.

| /etc/drbd.conf |
| --- |
| include "/etc/drbd.d/global_common.conf";<br>include "/etc/drbd.d/*.res"; |

**Global 섹션**

일반적으로 다음 경로에 있습니다. /etc/drbd.d/global_common.conf
이 섹션의 옵션은 대부분은 공통 사용자와 관련된 옵션을 설정 합니다.

usage-count : DRBD 사용 통계를 보관하는 옵션입니다. 기본값은 ask 이며, no, yes 로 설정이 가능합니다.

**Common 섹션**

일반적으로 다음 경로에 있습니다. /etc/drbd.d/global_common.conf
Common 섹션은 필수 요구사항은 아니지만 둘 이상의 리소스를 사용할 경우에는 사용을 권고하고 있습니다.

**Resource 섹션**

리소스별 구성파일의 이름은 사용자 요구사항에 맞게 작성한 뒤 /etc/drbd.conf 파일의 Include 형식에 맞게 해당경로에 .res 파일을 생성하면 사용이 가능합니다.

임의의 식별자를 사용할 수 있지만 이름에는 US-ASCII 문자 집합에있는 문자 이외의 자나 공백을 포함해서는 안됩니다.

리소스 구성은각 노드에 하나씩 구성을 설정하여야 하며, 다른 구성 설정은 common 섹션에서 상속되거나 DRBD 의 기본설정을 따름니다.

각 항목별 정의

| 구분 | 설명 |
|---|---|
| resource | resource 를정의하는 블록. Drbd 리소스 사용 명칭을 정의 |
| protocol | 데이터 전송 프로토콜을지정 |
| protocol A | 로컬 디스크에 쓰기가 끝나고 TCP 버퍼에 데이터를 송신한 시점에서 쓰기 작업을 완료.<br>(성능중시, 비동기 전송) |
| protocol B | 로컬 디스크에 쓰기가 끝나고 원격 호스트로 데이터가 도달한 시점에서 쓰기 작업을 완료. (A 와 C 의 중간) |
| protocol C | 원격 호스트의 디스크에도 쓰기가 완료된 시점에서 쓰기 작업을 완료로 한다. (신뢰성 중시, 동기 전송) |
| device | drbd 의 논리 블록 디바이스를 지정 |
| disk | 미러링할 물리 디바이스를 지정 |
| address | 데이터를 동기화하기위해 수신 대기할 IP 주소와 포트를 지정한다. 포트는 리소스마다 고유해야함. |
| meta-disk | 메타 데이터를 저장할디바이스를 지정.<br>Internal(내부 disk) / External (외부 disk) |

**리소스 설정파일 생성 (참고)**

drbddata 라는 별도의 리소스 설정파일을 수동으로 생성합니다. 양쪽 노드에 동일하게 생성하거나 복사합니다.

```
shell> vi /etc/drbd.d/drbddata.res

resource "drbddata" {
  protocol C;
  startup {
  # cluster 로부터 connection 이 떨어진 후 wait timeout 설정
  wfc-timeout       60;     # 1 분
  degr-wfc-timeout  120;    # 2 분.
  }
  disk {
  # lower level device 가 io-error 를 상위 layer 로 report 하는 옵션
  (detach : 에러 발생시 disk 분리)
```

```
on-io-error detach;
resync-rate 300M;          # bandwidth 제한 설정
}
net { timeout    60;       # node fail 체크 시간 설정 (6 초)
connect-int      10;       # 10 seconds  (unit = 1 second)
ping-int         10;       # keep-alive 체크 간격 설정 (10 초)
max-buffers      20000;    # 허용되는 request 수 설정
max-epoch-size      20000;
# split brain 이후의 작업 설정 (split brain 발생시 작업 절차)
      after-sb-0pri disconnect;
      after-sb-1pri disconnect;
      after-sb-2pri disconnect;
      rr-conflict disconnect;
}
on node1 { device /dev/drbd0;
disk /dev/sdb1;            # 동기화 될 device 명
address 172.40.40.49:7788; # 동기화 될 disk 명
# disk 위치 (로컬디스크 internal 외부디스크 external)
meta-disk internal;
}
on node2 { device /dev/drbd0;
disk /dev/sdb1;            # 동기화 될 device 명
address 172.40.40.50:7788; # 동기화 될 disk 명
# disk 위치 (로컬디스크 internal 외부디스크 external)
meta-disk internal;
}
}
```

### 2.2.4. 방화벽 오픈

설정파일에서 설정된 방화벽 포트 오픈하거나, 임시 테스트 진행을 위해 방화벽 기능을 중지합니다.

```
shell>  firewall-cmd --permanent --zone=public --add-port=7788/tcp
or
shell>  systemctl stop firewalld.service
```

### 2.2.5. 메타데이터 생성

초기 장치 생성시에만 수행합니다. DRBD 의 메타 데이터를 초기화합니다.

```
shell> drbdadm create-md drbddata
md_offset 32211202048
al_offset 32211169280
bm_offset 32210186240
Found ext3 filesystem
    31455260 kB data area apparently used
    31455260 kB left usable by current configuration
Even though it looks like this would place the new meta data into
unused space, you still need to confirm, as this is only a guess.
Do you want to proceed?
[need to type 'yes' to confirm] yes
initializing activity log
initializing bitmap (960 KB) to all zero
Writing meta data...
New drbd meta data block successfully created.
```

## 2.3. DRBD Management

### 2.3.1. DRDB 서비스 구동

**Node1(Master) DRBD 서비스 구동**

```
node1> systemctl start drbd.service
```

**서비스 구동 확인**

```
node1> cat /proc/drbd
version: 8.4.11-1 (api:1/proto:86-101)
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@, 2018-11-03 01:26:55
 0: cs:WFConnection ro:Secondary/Unknown ds:Inconsistent/DUnknown C r----s
    ns:0 nr:0 dw:0 dr:0 al:8 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:31455260
```

**Node2(Slave) DRBD 서비스 시작**

```
node2> systemctl start drbd.service
```

**서비스 시작 확인**

```
node2> cat /proc/drbd
version: 8.4.11-1 (api:1/proto:86-101)
```

```
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@, 2018-11-03 01:26:55
 0: cs:SyncSource ro:Secondary/Secondary ds:UpToDate/Inconsistent C r-----
    ns:1591768 nr:0 dw:0 dr:1591768 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:29863492
```

### 2.3.2. DRDB Sync 동기화

아래 명령어를 통해 두 노드의 디스크를 강제로 동기화 합니다.

```
node1> drbdadm primary --force drbddata
node1> cat /proc/drbd
version: 8.4.11-1 (api:1/proto:86-101)
version: 8.4.11-1 (api:1/proto:86-101)
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@, 2018-11-03 01:26:55
 0: cs:SyncTarget ro:Primary/Secondary ds:Inconsistent/UpToDate C r-----
    ns:0 nr:3390228 dw:3390228 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:1600296
         [============>.......] sync'ed: 68.1% (1560/4872)M
         finish: 0:00:18 speed: 85,668 (91,624) want: 83,920 K/sec
```

### 2.3.3. 파티션 마운트

```
node1> mkfs.ext4 /dev/drbd0
...
Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
node1> mkdir /data
node1> mount /dev/drbd0 /data
```

**마운트 확인**

```
node1> df -h | grep data
/dev/drbd0                30G   45M   28G   1% /data
```

### 2.3.4. 상태확인

**drbdadm status** 명령어를 통해서도 상태를 확인 할 수 있습니다.

```
node1> drbdadm status
drbddata role:Primary
```

```
disk:UpToDate
peer role:Secondary
  replication:Established peer-disk:UpToDate
```

# 3. Take Over Test

작동테스트를 위해 테스트 파일 생성 후 수동으로 Master 와 Slave 의 역할을 변경하여 테스트를 진행합니다.

[시나리오]

- node1 에서 DRBD 용 디렉토리 마운트
- 해당 디렉토리에 임의의 파일 생성
- node1 을 primary -> secondary 로 변경
- node2 를 secondary -> primary 로 변경
- DRBD 디렉토리를 마운트하여 확인

## 3.1. 장애처리 테스트

[Master]

node1> touch /data/test.txt → DRBD 디렉터리에 테스트 파일 생성

node1> umount /data

※ 사용자 및 프로세스 사용으로 umount 실패시 fuser -ck /data 명령 실행으로 강제 프로세스를 종료합니다.

node1> drbdsetup /dev/drbd0 secondary

node1> cat /proc/drbd → 구동상태 확인

```
version: 8.4.11-1 (api:1/proto:86-101)
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@, 2018-11-03 01:26:55
 0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
    ns:17004 nr:32225556 dw:32242560 dr:49298 al:27 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f
oos:0
```

[Slave]

node2> drbdsetup /dev/drbd0 primary

node2> mount /dev/drbd0 /data

node2> cat /proc/drbd → 구동상태 확인

```
version: 8.4.11-1 (api:1/proto:86-101)
```

```
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@, 2018-11-03 01:26:55
 0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r-----
    ns:0 nr:16996 dw:16996 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

```
node1> df -h | grep data                                    → 마운트 확인
```

### 3.2. 장애처리 원복

[Slave]

```
node2> umount /data
node2> drbdsetup /dev/drbd0 secondary
```

[Master]

```
Node1> drbdsetup /dev/drbd0 primary
Node1> mount /dev/drbd0 /data
```

## 4. 장애처리 절차

다양한 장애유형에 맞추어 아래의 절차로 장애를 처리합니다.

### 4.1. 정상상태

```
node1> cat /proc/drbd
version: 8.4.11-1 (api:1/proto:86-101)
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@,
2018-11-03 01:26:55
 0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r-----
    ns:0 nr:31455264 dw:31455264 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1
wo:f oos:0
```

### 4.2. DRBD Connection Fail (1 Node)

한쪽 노드의 DRBD 커넥션만 끊어진 상태.
link down 및 drbd crash 상태는 아닌 경우로, 끊어진 StandAlone 노드에서 아래의 명령어를 수행합니다.

```
node1>  cat /proc/drbd
```

```
version: 8.4.11-1 (api:1/proto:86-101)
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@,
2018-11-03 01:26:55
 0: cs:WFConnection ro:Primary/Unknown ds:UpToDate/DUnknown C r-----
    ns:0 nr:31455268 dw:31455268 dr:2088 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
ep:1 wo:f oos:0
```

```
node2> drbdadm connect all
```

or

```
node2> systemctl restart drbd.service
```

→ 서비스 재기동

## 4.3. DRBD Connection Fail (ALL Node)

모든 노드의 DRBD 커넥션만 끊어진 상태.
link down 및 drbd crash 상태는 아닌 경우로, 끊어진 모든 노드에서 아래의 명령어를 수행합니다.

**[상태확인]**

```
node1>  cat /proc/drbd
version: 8.4.11-1 (api:1/proto:86-101)
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@,
2018-11-03 01:26:55
 0: cs:StandAlone ro:Primary/Unknown ds:UpToDate/DUnknown C r-----
    ns:0 nr:31455268 dw:31455268 dr:2088 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
ep:1 wo:f oos:0
```

```
node2>  cat /proc/drbd
version: 8.4.11-1 (api:1/proto:86-101)
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@,
2018-11-03 01:26:55
 0: cs:StandAlone ro:Secondary/Unknown ds:UpToDate/DUnknown C r-----
    ns:0 nr:31455268 dw:31455268 dr:2088 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
ep:1 wo:f oos:0
```

**[DRBD 커넥션 재연결]**

```
node1> drbdadm connect all
```

or

```
node1> systemctl restart drbd.service          → 서비스 재기동

node2> drbdadm connect all
or
node2> systemctl restart drbd.service          → 서비스 재기동
```

## 4.4. Split Brain (데이터 싱크 실패)

DRBD 의 동기화 상태가 깨진 상태를 Split Brain 이라고 합니다. 보통 데이터 확인 후 수동으로 복구하는 것을 권장하나, 자동 복구를 위한 설정도 가능합니다.

StnadAlone 모드에서 drbdadm connect all 명령어 수행시에도 연결이 되지 않는 경우 아래의 절차로 수행합니다.

수동으로 해당노드에서 아래의 명령어 수행시에는 데이터의 유실이 발생할 수 있습니다.

```
node1> drbdadm disconnect all
node1> drbdadm -- --discard-my-data connect all
node1> drbdadm connect all
```

위의 절차 수행으로도 되지 않는 경우에는 하기 명령어로 수행합니다.

```
node1> drbdadm invalidate all
```

아래와 같이 리소스간 데이터 싱크를 맞추는 작업을 재수행합니다.

```
node1>  cat /proc/drbd
GIT-hash: 66145a308421e9c124ec391a7848ac20203bb03c build by mockbuild@,
2018-11-03 01:26:55
 0: cs:SyncSource ro:Secondary/Primary ds:UpToDate/Inconsistent C r-----
    ns:735232 nr:24 dw:24 dr:735232 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1
wo:f oos:30720028
        [>....................] sync'ed:  2.4% (30000/30716)M
        finish: 0:12:31 speed: 40,844 (40,844) K/sec
```

### 4.5. DRBD 수동 분리

Disk 교체나 데이터 싱크 실패등의 사유로 수동관리를 위해 DRBD 리소스를 분리해야 하는 경우 아래와 같이 수행 합니다.

Syntax> drbdadm detach [리소스명 or all]
node1> drbdadm detach drbddata

디스크 분리 확인
Syntax> drbdadm dstat [리소스명 or all]
node1> drbdadm dstat drbddata

## 5. Reference

해당자료는 아래의 Site 정보를 토대로 작성되었습니다.

DRBD 공식사이트
https://docs.linbit.com/docs/users-guide-9.0/