Report

오픈소스를 활용한 3-Tier 환경 구축 #2주차

- IT 인프라 기초(2) -

작성자	코더 – 정지호, 최예진, 김재현
검수자	송인섭 이사, 윤상훈 수석
작성일	2022-09-25

목차

CLOIT

개요

CLOIT

1) RAID란?

- RAID 레벨
- RAID 구현

2) 블록 스토리지 증설 및 마운트

- 단일 구성
- 확장(lvm) 구성

3) NFS 서버 구축

4) NTP 서버 구축

- 타임서버
- NTP 서버 구축

RAID란?

CLOIT

네트워크를 구성하려면 많은 트래픽을 견뎌야 하고, 어마어마한 용량을 필요로 한다.
일반 스토리지 하나로는 그것을 감당할 수 없기에, 여러 개의 스토리지를 사용해 하나의 디스크처럼 사용하게 되는데 이러한 개념을
Redundant Array of Inexpensive/Independent Disk라고 부른다.

기대 효과로는

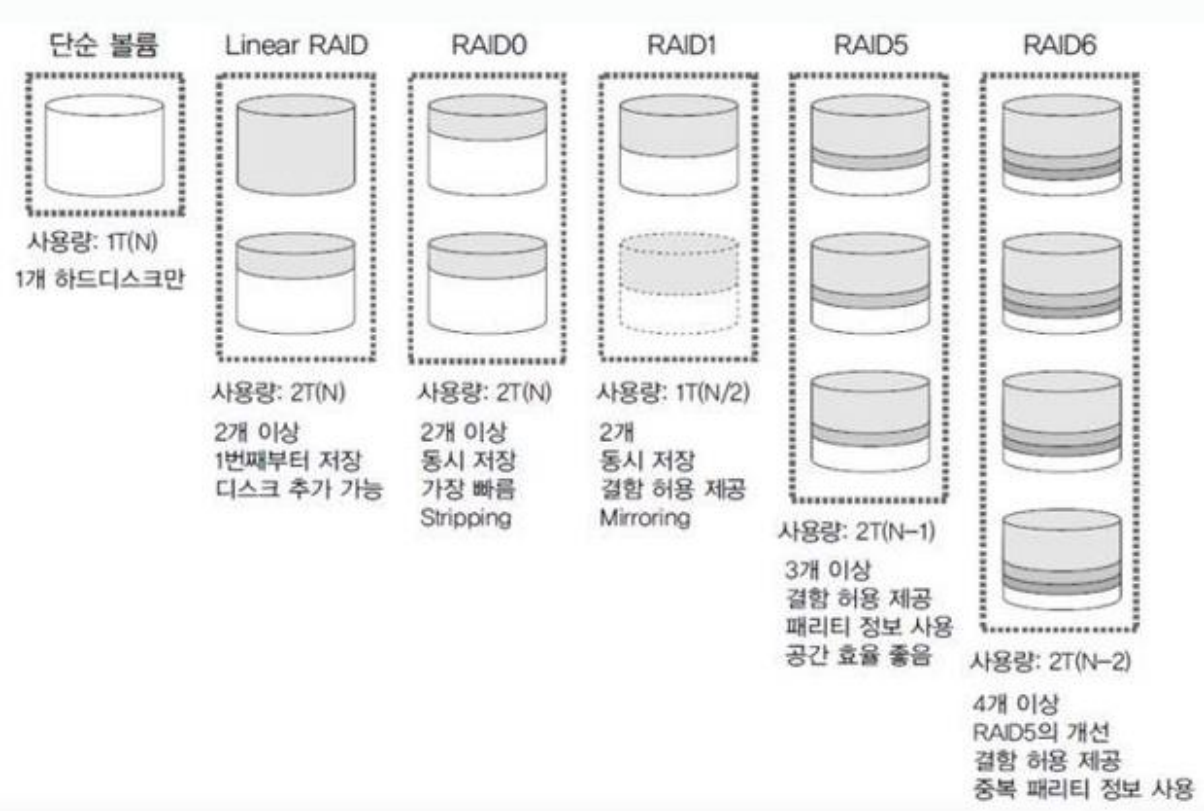
- 대용량의 단일 볼륨을 사용하는 효과
- 디스크 I/O 병렬화로 인한 성능 향상
- 데이터 복제로 인한 안정성 향상.

주 사용 목적은 (고가용성을 보장하는) 무정지 구현과 고성능 구현으로 구분되며, 무정지 구현은 RAID 1, 고성능은 RAID 0로 대표된다.

하드디스크의 느린 속도를 보완하기 위해 만든 기술로, RAID의 구성 방식에 따라 성능, 용량이 바뀌게 된다.
일반적으로 하드웨어 RAID라고 부르는 하드웨어 제조업체에서 여러 개의 하드디스크를 가지고 장비를 만들어 그 자체를 공급하는 방식은 안정적이지만, 매우 비싸서 함부로 이용하기 부담스럽고, 따라서 구성을 위해 좀 더 저렴하고 안전하게 데이터의 저장이 가능한 소프트웨어 RAID 방식을 많이 사용한다.
고가의 하드웨어 RAID의 대안이며 운영체제에서 자체적으로 지원하는 방식이기도 하다, 라고 한다.

RAID 레벨

Standard RAID 레벨에는 대표적으로 `Linear RAID, RAID 0, 1, 5, 6`이 있다.
디스크의 개수는 N으로 표시한다.



Linear RAID

말 그대로 '이어 붙인 하드디스크'. 1TB 하드가 3개라면 정보를 저장할 때 위에서부터 순차적으로 저장하여 3TB를 사용하는 방식으로, 가장 고전적이라고 할 수 있다.

RAID 0 (Striping)

최소 2개의 하드디스크를 사용, 하나의 정보를 2개 이상의 디스크에 나눠서 저장한다. Linear RAID보다 저장속도의 효율이 좋다.
이론적으로 단일 디스크를 사용하는 것에 비해 N배의 성능을 보장한다.
하지만 정보를 나눠 담은 하나의 디스크라도 고장나면 모든 정보를 사용할 수 없게 되므로 신뢰성이 낮다고 할 수 있다.

RAID 시키는 하드디스크 용량이 각각 다르게 되면 공간 효율이 100%가 될 수 없는데, 가장 작은 하드디스크의 용량에 맞춰 저장되기 때문.
때문에 이 방식을 사용하게 되면 서로 호환이 되는 동일 제조사, 동일 용량의 디스크를 사용하는 것이 일반적이다.

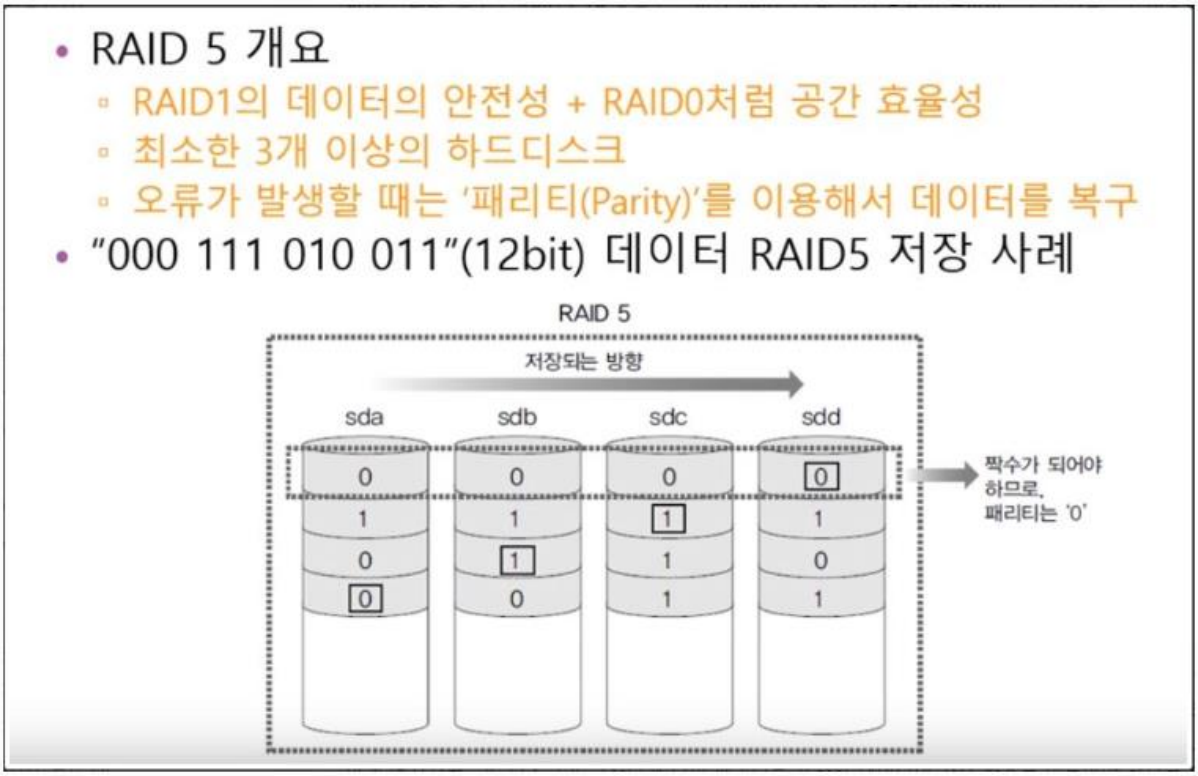
RAID 1 (Mirroring)

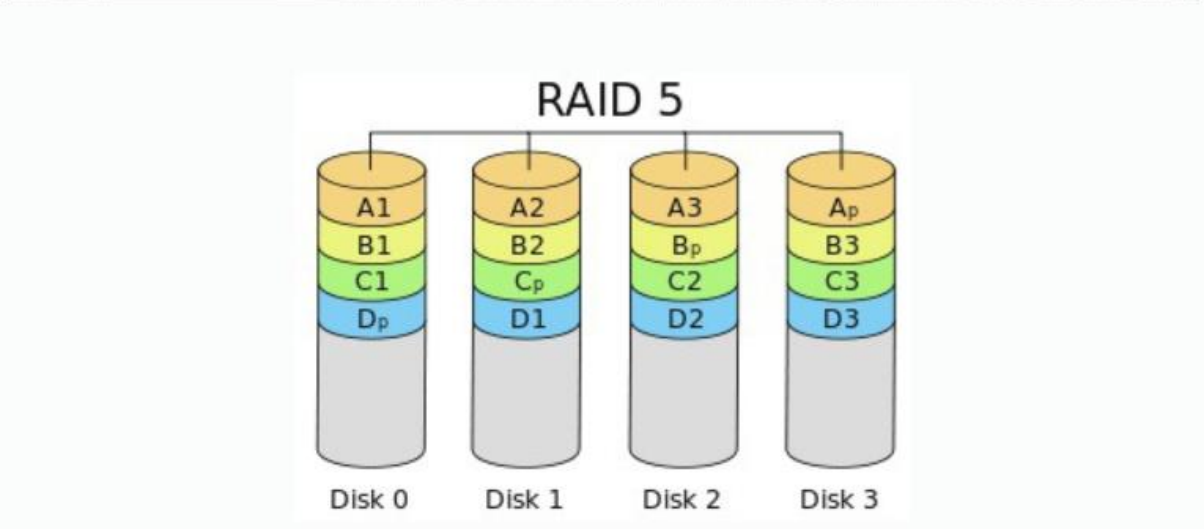
미러링 방식이라고 부르며 동일한 데이터를 2개 이상의 하드디스크에 동일하게 저장하므로 저장효율이 반토막 난다.
공간 효율은 좋지 않지만, 하나의 디스크가 고장나도 다른 디스크는 문제 없이 사용가능 하므로 정보 저장의 신뢰성이 높다고 할 수 있다.
멤버 디스크가 늘더라도 저장 공간은 증가하지 않지만, 가용성이 크게 증가하며 서버에서 끊김 없이 지속적으로 서비스를 제공하기 위해 사용한다.
백업의 목적 보다는 가용성에 초점을 맞추고 사용하는 것이 옳다. 물론, 디스크 고장 상황에서 백업과 유사한 보호 기능을 제공하며, 가장 최신화 된 데이터를 항상 유지할 수 있으나 랜섬웨어 등에 대응할 수 없다는 점에서 반쪽짜리 백업에 불과하다.
비용효율이 좋지 않다.

RAID 5

RAID 0와 RAID 1의 장점을 조합해서 사용하는 방식.
RAID 1의 데이터 안전성 + RAID 0의 빠른 저장속도와 공간효율.
패리티(Parity)를 사용하게 되는데 일반적으로 하드디스크 7~10개를 연결하게 된다.

패리티 :
Block 단위로 스트라이핑을 하고, error correction을 위해 패리티를 한개의 디스크에 저장하는데, 패리티를 저장하는 디스크를 고정하지 않고, 매 번 다른 디스크에 저장한다.
패러티를 이용해서 하나의 디스크가 문제가 생겨도 잃어버린 데이터를 복구 할 수 있다.
하나의 멤버 디스크 고장에는 견딜 수 있지만, 두개 이상 고장나면 데이터가 모두 손실된다.





DB 서버 등 큰 용량과 무정지 복구 기능을 동시에 필요로 하는 환경에서 주로 사용된다.

가용성은 높은 편이나, 패리티 연산을 통해 데이터를 저장한다는 특징 때문에 까다롭다.

RAID 6

RAID 5에서 성능, 용량을 좀 더 줄이고, 안정성을 높인 레벨.
패리티 디스크를 두개 사용한다.
패리티를 2개 사용하므로 RAID 5 보다 가용성이 높지만, 정보를 저장하지 않은 디스크 2개나 생겨 공간효율과 성능이 떨어지므로 활용 빈도는 조금 떨어진다.
5가 1개까지의 고장을 허용했다면 6은 2개까지는 허용된다.

Nested RAID (복합)

위의 0~6 레벨의 Standard RAID를 여러 개 중첩하여 사용한다.
RAID 볼륨의 멤버로 다른 레이드 볼륨을 사용하는 형태.
RAID 1+ RAID 0, RAID 5 + RAID 0, RAID 5 + RAID 1 등의 방식이 있다.

RAID 구현

RAID 레벨이 논리적 구성 방법이었다면, RAID의 구현은 물리적 구성 방법에 대해 설명한다.

하드웨어 RAID

별도의 RAID 카드를 장착하여 구현하는 방법.

카드에는 RAID를 관리하는 컨트롤러 칩셋과 방열판, 캐시로 사용하기 위한 메모리가 달려 있고, 디스크를 연결하기 위한 인터페이스(주로 SAS)가 달려있다.

전원 장애 시 캐시의 내용을 잠시나마 유지시키기 위해 배터리가 달려 있기도 하다.

RAID 카드는 대부분 PCI-E 슬롯에 꽂아서 사용하며, OS부팅 이전 RAID 카드의 설정 페이지로 진입하여 관리하게 된다.

RAID 구성 후 부팅을 하게 되면 OS에서는 구성이 완료된 RAID 볼륨만 확인할 수 있다. 즉, RAID를 구성하고 있는 단일 디스크의 정보는 RAID 카드를 통해서만 알 수 있는 셈.

카드에 달려 있는 별도의 컨트롤러 칩셋이 RAID를 관리하기 때문에 다른 방식에 비해 성능이 월등히 높다. 특히 별도의 패리티 연산이 필요한 5나 6에서 매우 높은 성능을 보여주며, RAID를 구성할 수 있는 최대 디스크 개수도 월등히 높다.

RAID 카드와 연결되어 있는 디스크를 그대로 제거하여 다른 PC, 서버로 이동시키면 기존 운영하던 RAID 볼륨을 거의 100% 그래도 유지할 수 있다는 것도 장점.

별도의 RAID 카드를 사야하기 때문에 구축 비용이 많이 들고 RAID를 구성하는 단일 디스크의 분석을 거의 할 수 없다는 것이 단점이다.

펌웨어(드라이버) RAID

드라이버 RAID, On-board RAID, 임베디드 RAID 등으로 불림.

RAID 카드 대신 기능을 간략화한 RAID 칩을 탑재하고 펌웨어(드라이버)로 제어하여 구현하는 방법.

보통 OS 진입 전 BIOS 메뉴에서 RAID를 구현한다. OS에 관계 없이 작동하며, OS에서는 원래 장착한 디스크 대신 가상의 BIOS RAID 하드웨어가 표시된다. 즉, 별도의 드라이버 소프트웨어를 통한 관리가 불가능.

펌웨어 RAID는 OS부팅 전에 RAID를 구성하기 때문에 OS 변경에는 영향을 미치지 않는다. 따라서 OS를 바꿔도 RAID는 유효한 대신, 메인보드를 바꾸게 되면 더 이상 사용하지 못할 가능성이 크다.
원래 디스크를 가상 디스크가 대체하는 방식이다 보니, 용량이 다른 두 하드웨어를 묶었을 때 남은 공간은 활용 못하고 버려지는 단점이 있다.

소프트웨어 RAID

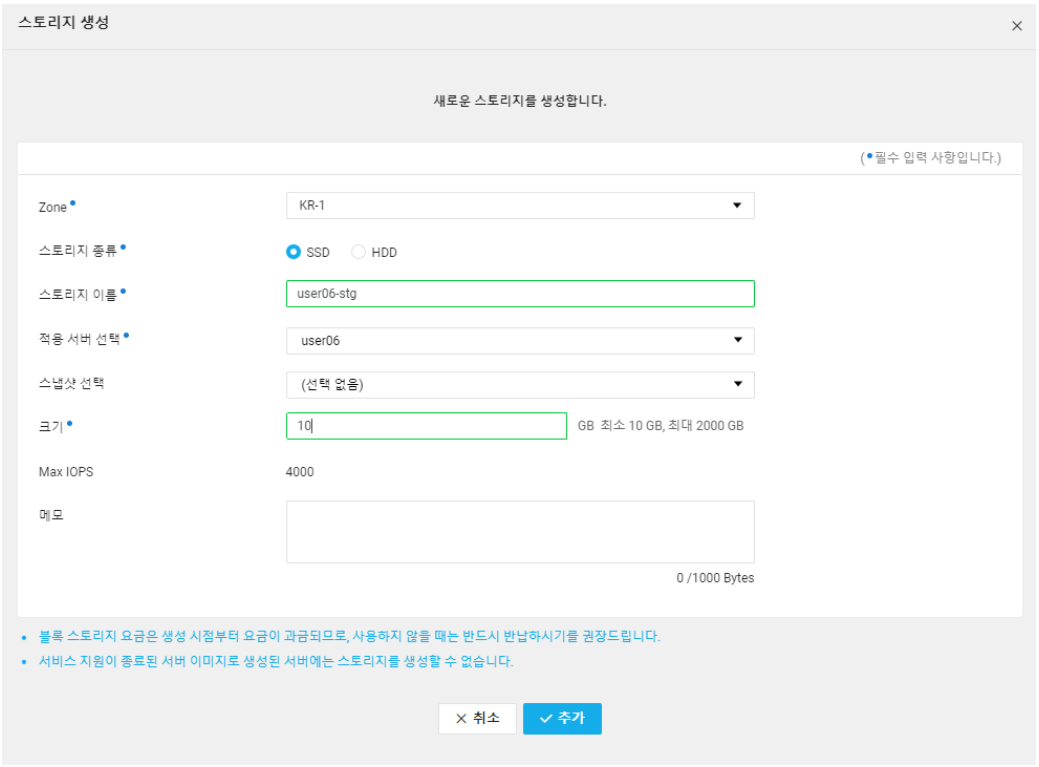
OS RAID라고도 함.
RAID 소프트웨어(프로그램)을 이용하여 RAID를 구성하는 방식. OS부팅 이후 관련 프로그램을 실행하여 RAID를 구축하게 된다.
OS가 인식하고 있는 블록 디바이스를 사용해서 구축하게 되며, OS의 디스크 관리메뉴에서 RAID를 구현하는 방법.
하드웨어 RAID와 다르게 RAID를 구성하고 있는 단일 디스크에 대한 분석을 할 수 있다.
단점으로는 OS 위에서 동작하는 프로그램이 관리하기 때문에 상대적으로 속도가 매우 느리고, 다른 프로그램들과 리소스(CPU, 메모리 등)을 같이 사용하기 때문에 전체적인 시스템의 성능이 떨어질 수도 있다.
OS RAID는 메인보드를 바꾸더라도 해당 디스크만 제대로 꽂아주면 계속 레이드를 사용할 수 있지만, OS를 바꾸면 보통 사용하지 못한다.
OS에서 관리하므로 다양한 방법으로 RAID를 구성할 수 있으며, 특히 용량이 다른 두 제품이 경우 RAID를 구성하고 남는 공간에 단일 파티션, 또는 또 다른 RAID Array를 구성할 수도 있다.

블록 스토리지 증설 및 마운트

CLOIT

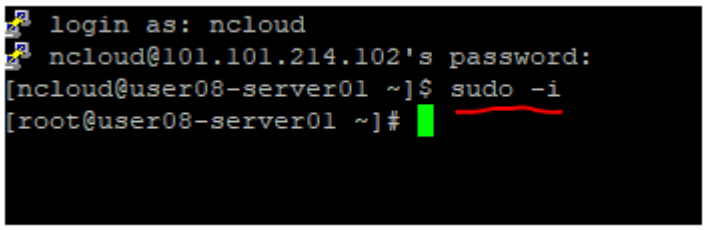
1. 단일 구성

1. Storage 콘솔 접속 후 스토리지 생성



2. sudo -i 를 입력하여 관리자 권한으로 변경

$ 에서 #으로 변경된다.



3. Fdisk -l 명령어로 추가된 스토리지 확인

```
[root@user06 ncloud]# fdisk -l

Disk /dev/xvda: 53.7 GB, 53687091200 bytes, 104857600 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000a38bc

    Device Boot      Start         End      Blocks   Id  System
/dev/xvda1   *        2048   104857599    52427776   83  Linux

Disk /dev/xvdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

4. fdisk /dev/xvdb 명령어로 디스크 파티션

```
[root@user06 ncloud]# fdisk /dev/xvdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x9c64fea3.

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): w
Partition number (1-4, default 1):
First sector (2048-20971519, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default 20971519):
Using default value 20971519
Partition 1 of type Linux and of size 10 GiB is set

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

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

(n 입력 -> p 입력 -> Enter 3번 -> w입력)

5. mkfs.xfs /dev/xvdb1 명령어로 스토리지 포맷

```
[root@user06 ncloud]# mkfs.xfs /dev/xvdb1
meta-data=/dev/xvdb1             isize=512    agcount=4, agsize=655296 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=2621184, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

6. 스토리지 마운트

mkdir /data 명령어로 data 마운트 포인트 생성.

mount /dev/xvdb1 /data 명령어로 /data에 생성한 스토리지(/dev/xvdb1)마운트

```
[root@user06 ncloud]# mkdir /data
[root@user06 ncloud]# mount /dev/xvdb1 /data
```

7. blkid, lsblk 명령어로 스토리지 확인

```
[root@user06 ncloud]# blkid
/dev/xvda1: UUID="79ec23f4-ea9a-4f2c-a2da-8be4ff4a4f09" TYPE="xfs"
/dev/xvdb1: UUID="d184b9ae-c5c6-4777-ae1d-52183de4576c" TYPE="xfs"
[root@user06 ncloud]# lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  50G  0 disk
└─xvda1 202:1    0  50G  0 part /
xvdb    202:16   0  10G  0 disk
└─xvdb1 202:17   0  10G  0 part /data
```

8. vi /etc/fstab 명령어로 마운트 정보 유지 설정

#vi /etc/fstab

/dev/xvdb1 /data xfs defaults,nofail 0 0

```
[root@user06 ncloud]# vi /etc/fstab

#
# /etc/fstab
# Created by anaconda on Mon Oct 12 19:30:22 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=79ec23f4-ea9a-4f2c-a2da-8be4ff4a4f09 /                       xfs     defaults        0 0
/dev/xvdb1                                /data                   xfs     defaults,nofail 0 0
```

- 볼륨설정 값

열 값	나타내는 설정	설명
/dev/xvdb1	볼륨 이름	해당 볼륨의 이름을 표시
/mnt/a	마운트 포인트	볼륨이 마운트될 위치
ext3	파일 시스템 종류	ext3 : CentOS 5.x ext4 : CentOS 6.x, Ubuntu Server/Desktop xfs : CentOS 7.x
defaults	옵션	defaults : 아래 5개 옵션을 모두 적용 auto : 부팅 시 자동으로 마운트 rw : 읽기와 쓰기가 가능하도록 마운트 nouser : root 계정만 마운트가 가능하도록 설정 exec : 파일 실행을 허용 suid : SetUID 와 SetGID 를 허용
1	덤프 설정	0 : 덤프되지 않는 파일 시스템 1 : 덤프 가능한 파일 시스템
2	fsck 설정	0 : 부팅 시 fsck 를 실행하지 않음 1 : 부팅 시 root 파일 시스템을 우선하여 확인 2 : 부팅 시 root 이외의 파일 시스템을 우선하여 확인

- 언마운트

umount /data

```
[root@user06 ~]# umount /data
```

2. 확장 구성 (LVM)

1. 추가 스토리지 생성.

user06-test1, user06-test2 이름으로 생성된 스토리지 2개를 user06서버에 연결

2. 패키지 설치

yum install lvm*

```
[root@user08-server01 ~]# yum install lvm*
Loaded plugins: fastestmirror, langpacks
Determining fastest mirrors
 * base: mirror.navercorp.com
 * extras: mirror.navercorp.com
 * updates: mirror.navercorp.com
base                                                     | 3.6 kB     00:00
extras                                                   | 2.9 kB     00:00
updates                                                  | 2.9 kB     00:00
(1/4): base/7/x86_64/group_gz                              | 153 kB   00:00
(2/4): extras/7/x86_64/primary_db                          | 250 kB   00:00
(3/4): base/7/x86_64/primary_db                            | 6.1 MB   00:00
(4/4): updates/7/x86_64/primary_db                         |  17 MB   00:00
Resolving Dependencies
--> Running transaction check
---> Package lvm2.x86_64 7:2.02.186-7.el7_8.2 will be updated
---> Package lvm2.x86_64 7:2.02.187-6.el7_9.5 will be an update
---> Package lvm2-cluster.x86_64 7:2.02.187-6.el7_9.5 will be installed
--> Processing Dependency: device-mapper = 7:1.02.170 for package: 7:lvm2-clus
r-2.02.187-6.el7_9.5.x86_64
--> Processing Dependency: resource-agents >= 3.9.5-25 for package: 7:lvm2-clu
```

```
Updated:
  lvm2.x86_64 7:2.02.187-6.el7_9.5      lvm2-libs.x86_64 7:2.02.187-6.el7_9.5

Dependency Updated:
  device-mapper.x86_64 7:1.02.170-6.el7_9.5
  device-mapper-event.x86_64 7:1.02.170-6.el7_9.5
  device-mapper-event-libs.x86_64 7:1.02.170-6.el7_9.5
  device-mapper-libs.x86_64 7:1.02.170-6.el7_9.5
  libblkid.x86_64 0:2.23.2-65.el7_9.1
  libmount.x86_64 0:2.23.2-65.el7_9.1
  libsmartcols.x86_64 0:2.23.2-65.el7_9.1
  libuuid.x86_64 0:2.23.2-65.el7_9.1
  systemd.x86_64 0:219-78.el7_9.7
  systemd-libs.x86_64 0:219-78.el7_9.7
  systemd-python.x86_64 0:219-78.el7_9.7
  systemd-sysv.x86_64 0:219-78.el7_9.7
  util-linux.x86_64 0:2.23.2-65.el7_9.1

Complete!
[root@user08-server01 ~]#
```

3. fdisk -l 명령어로 스토리지 확인

```
[root@user08-server01 ~]# fdisk -l

Disk /dev/xvda: 53.7 GB, 53687091200 bytes, 104857600 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000a38bc

   Device Boot      Start         End      Blocks   Id  System
/dev/xvda1   *        2048   104857599    52427776   83  Linux

Disk /dev/xvdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xe073324a

   Device Boot      Start         End      Blocks   Id  System
/dev/xvdb1            2048    20971519    10484736   8e  Linux LVM

Disk /dev/xvdc: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
```

```
Disk /dev/xvdc: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

4. fdisk /dev/xvdc 명령어로 디스크 파티션

```
root@user08-server01 ~]# fdisk /dev/xvdc
elcome to fdisk (util-linux 2.23.2).

hanges will remain in memory only, until you decide to write them.
e careful before using the write command.

evice does not contain a recognized partition table
uilding a new DOS disklabel with disk identifier 0x0a65e96b.

ommand (m for help):
```

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-20971519, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default 20971519):
Using default value 20971519
Partition 1 of type Linux and of size 10 GiB is set

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

Calling ioctl() to re-read partition table.
Syncing disks.
[root@user08-server01 ~]#
```

```
   Device Boot      Start         End      Blocks   Id  System
/dev/xvdc1            2048    20971519    10484736   83  Linux
```

/dev/xvdc1 디바이스가 생성된 것을 확인할 수 있다.

다시 한 번 fdisk /dev/xvdc 명령어를 사용한다. 하지만 다른 방식으로 진행한다.

```
Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): L

 0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
 1  FAT12           27  Hidden NTFS Win 82  Linux swap / So c1  DRDOS/sec (FAT-
 2  XENIX root      39  Plan 9          83  Linux           c4  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     85  Linux extended  c7  Syrinx
 5  Extended        41  PPC PReP Boot   86  NTFS volume set da  Non-FS data
 6  FAT16           42  SFS             87  NTFS volume set db  CP/M / CTOS / .
 7  HPFS/NTFS/exFAT 4d  QNX4.x          88  Linux plaintext de  Dell Utility
 8  AIX             4e  QNX4.x 2nd part 8e  Linux LVM       df  BootIt
 9  AIX bootable    4f  QNX4.x 3rd part 93  Amoeba          e1  DOS access
 a  OS/2 Boot Manag 50  OnTrack DM      94  Amoeba BBT      e3  DOS R/O
 b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedStor
 c  W95 FAT32 (LBA) 52  CP/M            a0  IBM Thinkpad hi eb  BeOS fs
```

t 와 L 을 입력해준다.

```
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

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

Calling ioctl() to re-read partition table.
Syncing disks.
[root@user08-server01 ~]#
```

8e 를 입력하면 파티션 타입이 Linux LVM 으로 변한다.

다음 w 를 입력해준다.

5. Physical Volume 생성

물리적인 디스크가 LVM 데이터 구조를 사용할 수 있도록 생성해준다.

pvcreate /dev/xvdb

```
[root@user06 ~]# pvcreate /dev/xvdb1
  Physical volume "/dev/xvdb1" successfully created.
[root@user06 ~]# pvcreate /dev/xvdc1
  Physical volume "/dev/xvdc1" successfully created.
```

pvdisplay 로 제대로 생성되었는지 확인할 수 있다.

```
[root@user06 ~]# pvdisplay
  "/dev/xvdc1" is a new physical volume of "<10.00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/xvdc1
  VG Name
  PV Size               <10.00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               cCzCys-Z6Ax-pX8o-euMO-KUs5-zfkP-jAqHEb

  "/dev/xvdb1" is a new physical volume of "<10.00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/xvdb1
  VG Name
  PV Size               <10.00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               QxnXm9-rpQM-p6y6-mT5x-fNQL-ILFa-IR6tYs
```

6. Volume Group 생성

한 개 이상의 PV 가 속해 있는 그룹을 말한다.

vgcreate vg0 /dev/xvdb1

```
[root@user08-server01 ~]# vgcreate vg0 /dev/xvdb1
  Volume group "vg0" successfully created
```

PV 를 추가해서 볼륨 확장

vgextend vg0 /dev/xvdc1

```
[root@user08-server01 ~]# vgextend vg0 /dev/xvdc1
  Volume group "vg0" successfully extended
[root@user08-server01 ~]#
```

처음부터 두개의 Volume 으로 하나의 VG 를 생성하는 모습.

vgcreate vg01 /dev/xvdb1 /dev/xvdc1

```
[root@user06 ~]# vgcreate VG01 /dev/xvdb1 /dev/xvdc1
  Volume group "VG01" successfully created
```

vgdisplay 로 생성된 Volume Group 을 확인할 수 있다.

```
[root@user06 ~]# vgdisplay
  --- Volume group ---
  VG Name               VG01
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               19.99 GiB
  PE Size               4.00 MiB
  Total PE              5118
  Alloc PE / Size       0 / 0
  Free  PE / Size       5118 / 19.99 GiB
  VG UUID               dUvP9m-zB44-e992-0Nj7-9o7X-T97v-CEoc3S
```

7. Logical Volume 생성

PV, VG 로 구성되어 있는 공간을 전체 또는 분할하여 사용할 수 있도록 논리적으로 할당한 공간

볼륨 그룹처럼, 논리 볼륨에 사용하는 이름은 관리자가 결정할 수 있다.

lvcreate –extents 100%FREE -n LV01 VG01

```
[root@user06 ~]# lvcreate --extents 100%FREE -n LV01 VG01
  Logical volume "LV01" created.
```

생성된 Logical Volume 확인

lvdisplay

```
[root@user06 ~]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/VG01/LV01
  LV Name                LV01
  VG Name                VG01
  LV UUID                3XVFtU-2jF0-3std-sU5m-XQd6-SAXV-aoTYjH
  LV Write Access        read/write
  LV Creation host, time user06, 2022-09-24 18:19:53 +0900
  LV Status              available
  # open                 0
  LV Size                19.99 GiB
  Current LE             5118
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:0
```

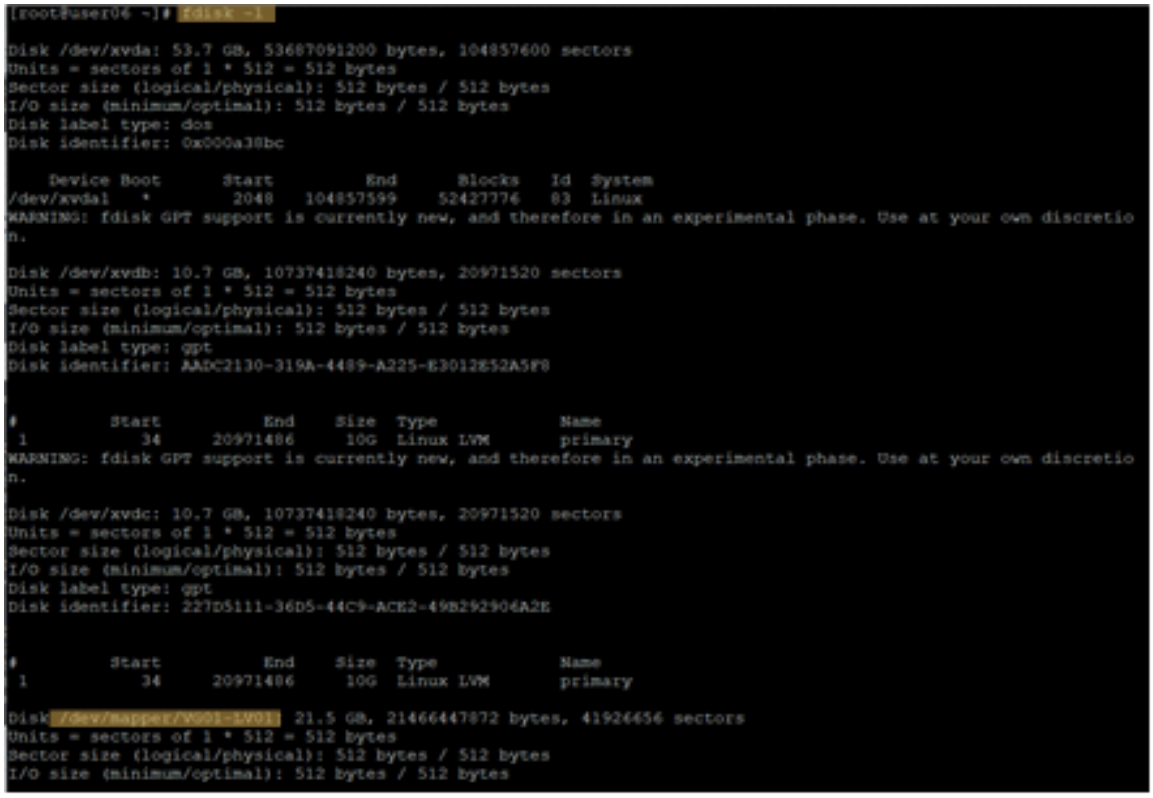
8. 포맷

mkfs.xfs /dev/VG01/LV01

```
[root@user06 ~]# mkfs.xfs /dev/VG01/LV01
meta-data=/dev/VG01/LV01         isize=512    agcount=4, agsize=1310208 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=5240832, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

9. 마운트

fdisk -l 으로 장치명 확인



디스크를 마운트 할 포인트, 즉 디렉토리를 원하는 이름으로 생성하고 마운트

mkdir /data

mount /dev/mapper/VG01-LV01 /data

```
[root@user06 ~]# mkdir /data
mkdir: cannot create directory '/data': File exists
[root@user06 ~]# mount /dev/mapper/VG01-LV01 /data
[root@user06 ~]# df -Th
Filesystem              Type      Size  Used Avail Use% Mounted on
devtmpfs                devtmpfs  1.9G     0  1.9G   0% /dev
tmpfs                   tmpfs     1.9G     0  1.9G   0% /dev/shm
tmpfs                   tmpfs     1.9G  9.1M  1.9G   1% /run
tmpfs                   tmpfs     1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/xvda1              xfs        50G  2.3G   48G   5% /
tmpfs                   tmpfs     378M     0  378M   0% /run/user/11000
/dev/mapper/VG01-LV01 xfs         20G   33M   20G   1% /data
```

10. 마운트 정보 유지 설정

새로 생성된 디스크를 부팅 후에도 인식할 수 있게 blkid 명령으로 UUID를 확인하고, fstab에 등록한다.

blkid |grep /dev/mapper/VG01-LV01

```
[root@user06 ~]# blkid |grep /dev/mapper/VG01-LV01
/dev/mapper/VG01-LV01: UUID="6e66968e-fb98-4203-91ec-2265efe807c2" TYPE="xfs"
```

vi /etc/fstab

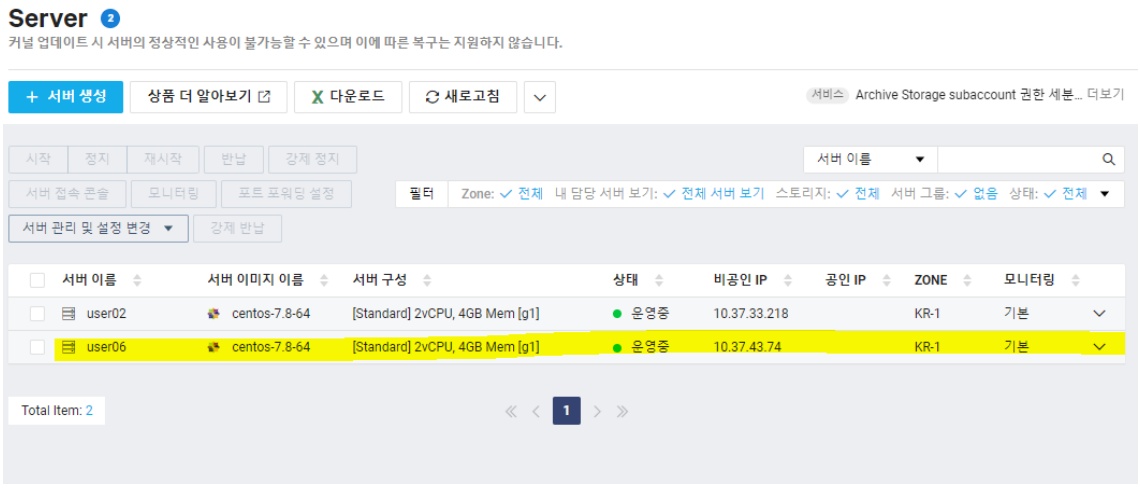
```
[root@user06 ~]# vi /etc/fstab

#
# /etc/fstab
# Created by anaconda on Mon Oct 12 19:30:22 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=79ec23f4-ea9a-4f2c-a2da-8be4ff4a4f09 /                       xfs     defaults        0 0
/dev/xvdb1                                /data                   xfs     defaults,nofail 0 0
UUID=6e66968e-fb98-4203-91ec-2265efe807c2 /data                   xfs     defaults        0 0
```

NFS 서버 구축

CLOIT

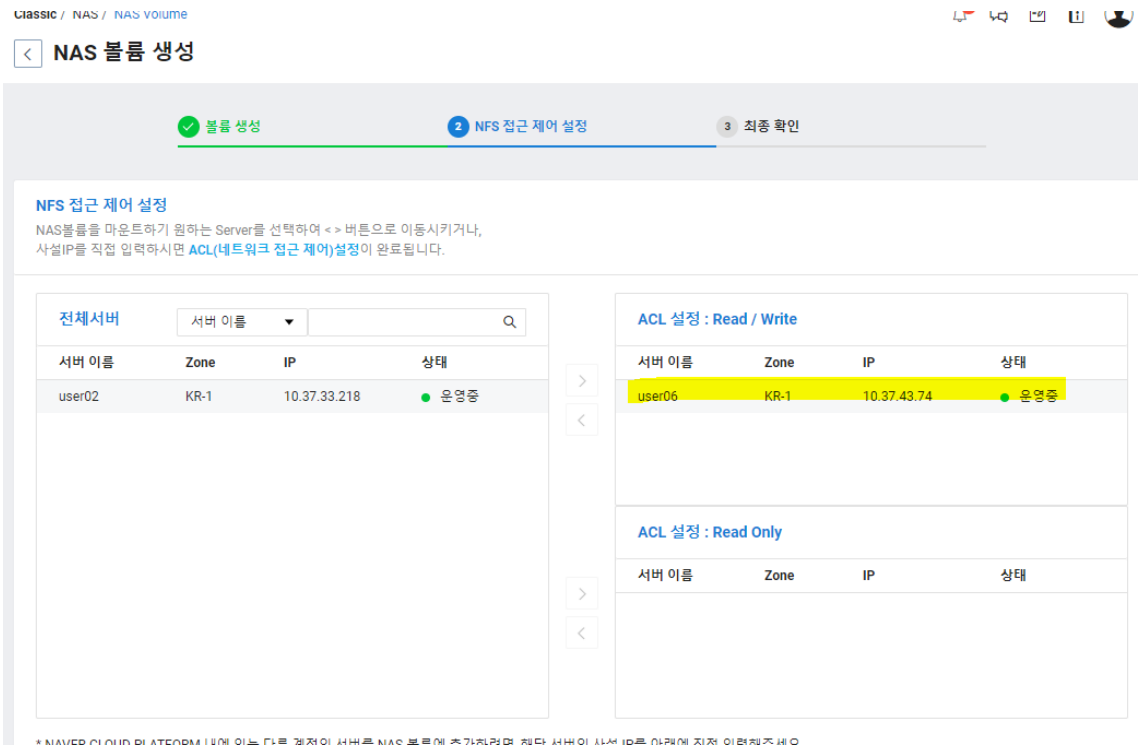
NFS 서버 구축하고, 클라이언트에서 NAS 연결해보기.

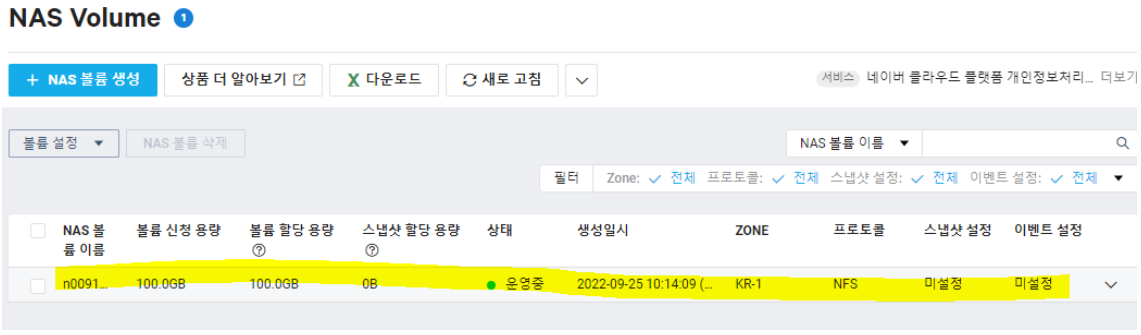
1. 서버 생성



2. NAS 볼륨 생성

3. NFS에 접근 가능한 서버 선택.





4. PuTTY 접속 후 NFS 관련 패키지를 설치한다.

yum install nfs-utils -y

```
[root@user06 ~]# yum install nfs-utils -y
Loaded plugins: fastestmirror, langpacks
Determining fastest mirrors
 * base: mirror.navercorp.com
 * extras: mirror.navercorp.com
 * updates: mirror.navercorp.com
base                                                     | 3.6 kB     00:00
extras                                                   | 2.9 kB     00:00
updates                                                  | 2.9 kB     00:00
(1/4): base/7/x86_64/group_gz                              | 153 kB   00:00
(2/4): extras/7/x86_64/primary_db                          | 250 kB   00:00
(3/4): base/7/x86_64/primary_db                            | 6.1 MB   00:00
(4/4): updates/7/x86_64/primary_db                         |  17 MB   00:00
Resolving Dependencies
--> Running transaction check
---> Package nfs-utils.x86_64 1:1.3.0-0.68.el7.2 will be installed
--> Processing Dependency: gssproxy >= 0.7.0-3 for package: 1:nfs-utils-1.3.0-0.
68.el7.2.x86_64
```

5. RPC 데몬 기동

systemctl start rpcbind

systemctl enable rpcbind

```
[root@user06 ~]# systemctl start rpcbind
[root@user06 ~]# systemctl enable rpcbind
```

6. 마운트 포인트 생성

mkdir /nas 디렉토리 생성

```
[root@user08-01 ~]# mkdir /nas
[root@user08-01 ~]#
```

7. NAS 볼륨 마운트

mount -t nfs -o vers=3 {NAS 볼륨 마운트 정보} /nas

```
[root@user06 ~]# mkdir /nas
[root@user06 ~]# mount -t nfs -o vers=3 10.250.116.32:/n009149_user06nas /nas
[root@user06 ~]# df -Th
Filesystem                      Type      Size  Used Avail Use% Mounted on
devtmpfs                        devtmpfs  1.9G     0  1.9G   0% /dev
tmpfs                           tmpfs     1.9G     0  1.9G   0% /dev/shm
tmpfs                           tmpfs     1.9G  9.0M  1.9G   1% /run
tmpfs                           tmpfs     1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/xvda1                      xfs        50G  2.2G   48G   5% /
tmpfs                           tmpfs     378M     0  378M   0% /run/user/11000
10.250.116.32:/n009149_user06nas nfs       100G  320K  100G   1% /nas
```

df -Th 로 확인할 수 있다.

```
[root@user06 ~]# mkdir /nas
[root@user06 ~]# mount -t nfs -o vers=3 10.250.116.32:/n009149_user06nas /nas
[root@user06 ~]# df -Th
Filesystem                      Type      Size  Used Avail Use% Mounted on
devtmpfs                        devtmpfs  1.9G     0  1.9G   0% /dev
tmpfs                           tmpfs     1.9G     0  1.9G   0% /dev/shm
tmpfs                           tmpfs     1.9G  9.0M  1.9G   1% /run
tmpfs                           tmpfs     1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/xvda1                      xfs        50G  2.2G   48G   5% /
tmpfs                           tmpfs     378M     0  378M   0% /run/user/11000
10.250.116.32:/n009149_user06nas nfs       100G  320K  100G   1% /nas
```

8. vi /etc/fstab

vi /etc/fstab

fstab을 수정해서 마운트 정보 유지 설정.

```
[root@user06 ~]# vi /etc/fstab


#
# /etc/fstab
# Created by anaconda on Mon Oct 12 19:30:22 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=79ec23f4-ea9a-4f2c-a2da-8be4ff4a4f09 /                       xfs     defaults        0 0
10.250.116.32:/n009149_user06nas /nas                             xfs     defaults        0 0
```

NTP 서버 구축

CLOIT

1. 타임 서버

- 시계에서 실제 시간(원자 시계 등이 기반)을 읽고, 컴퓨터 네트워크를 사용하여 시간 정보를 클라이언트에 배포하는 서버 컴퓨터이다.
- 인터넷을 통한 시간 정보를 배포하는데 널리 사용되는 프로토콜이 NTP(Network Time Protocol)이다.
- 최대 256계급까지 존재하고, 16계급까지 사용된다고 한다.
- 계급 순위가 낮을수록 타임 서버의 우선순위가 높아진다.

2. NTP 서버 구축하기

1. 타임 서버 ip 주소 파악.

ip addr

```
[root@user08-server01 ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group defa
ult qlen 1000
    link/ether f2:20:cd:37:57:6e brd ff:ff:ff:ff:ff:ff
    inet 10.37.39.78/23 brd 10.37.39.255 scope global dynamic eth0
       valid_lft 946079749sec preferred_lft 946079749sec
[root@user08-server01 ~]#
```

Ip = 10.37.39.78

2. ntp 패키지 설치

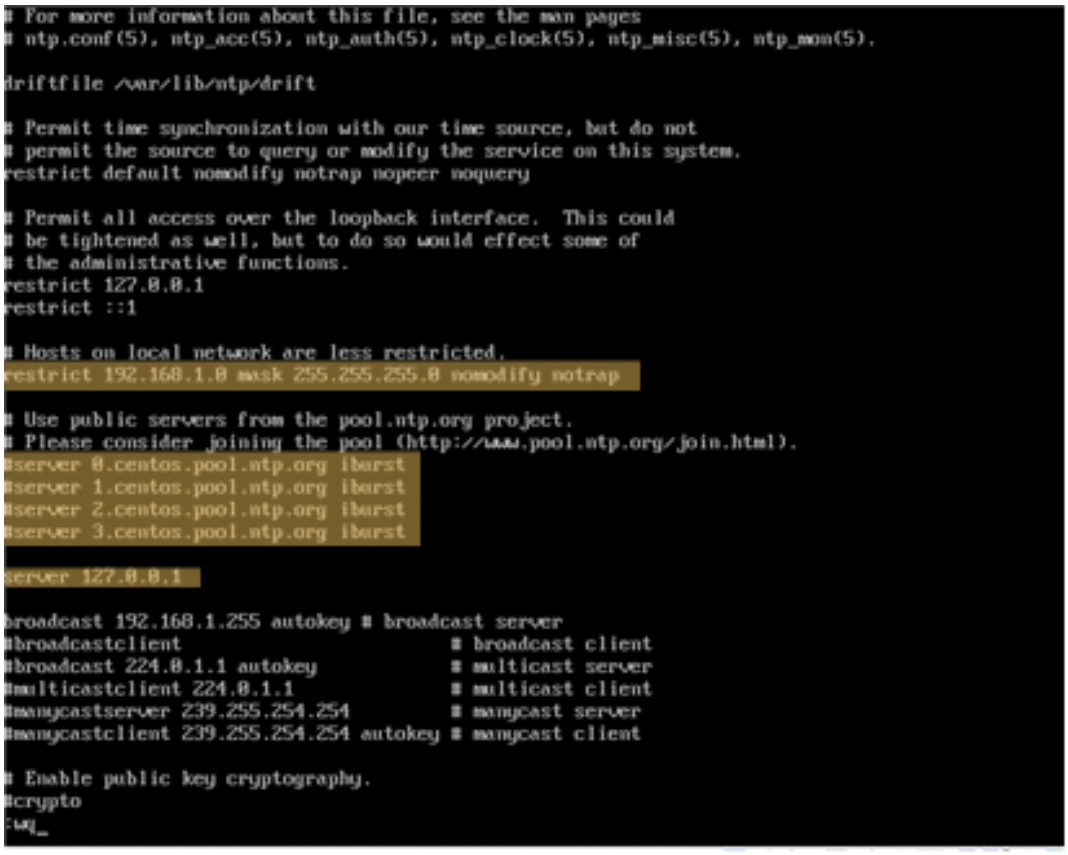
yum install -y ntp

```
[root@user08-server01 ~]# yum install -y ntp
Loaded plugins: fastestmirror, langpacks
Determining fastest mirrors
 * base: mirror.kakao.com
 * extras: mirror.kakao.com
 * updates: mirror.kakao.com
base                                                     | 3.6 kB     00:00
extras                                                   | 2.9 kB     00:00
updates                                                  | 2.9 kB     00:00
(1/4): base/7/x86_64/group_gz                              | 153 kB   00:00
(2/4): extras/7/x86_64/primary_db                          | 250 kB   00:00
(3/4): base/7/x86_64/primary_db                            | 6.1 MB   00:00
(4/4): updates/7/x86_64/primary_db                         |  17 MB   00:00
Resolving Dependencies
--> Running transaction check
---> Package ntp.x86_64 0:4.2.6p5-29.el7.centos.2 will be installed
```

3. ntp 서버 설정

vi /ect/ntp.conf

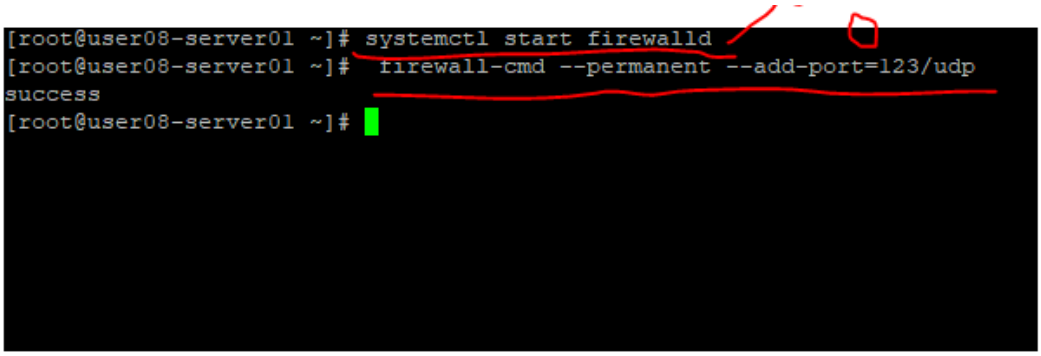
1) conf 파일에 접속한다.



2) restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap 부분의 주석을 해제해준다. 현재 클라이언트가 존재하는 네트워크 서버들이 ntp 서버에 시간을 요청할 수 있도록 한다.

3)우선 순위가 높은 타임서버의 주석처리.

4. 방화벽 ID 실행 후 포트 추가.



영구적으로 포트 아이디 123 -u에 프로토콜을 add 해준다.

5. 포트 추가/변경은 --reload 옵션을 실행시켜야 반영된다.

```
[root@localhost ~]# firewall-cmd --reload
success
```

6. systemctl을 통해 서비스를 실행해준다.

systemctl start ntpd

systemctl enable ntpd

```
[root@localhost ~]# systemctl start ntpd
[root@localhost ~]# systemctl enable ntpd
```

7. 다시 .conf 파일에 들어가서 타임 서버 IP 주소를 기입하고 편집기에서 나간다.

```
#server 0.centos.pool.ntp.org iburst
#server 1.centos.pool.ntp.org iburst
#server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst
server 10.37.39.78
#broadcast 192.168.1.255 autokey        # broadcast server
#broadcastclient                        # broadcast client
```

8. 현재 클라이언트에서 어떤 타임 서버를 쓰는지 파악

ntpq -p

```
[root@user08-server01 ~]# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 localhost       .INIT.          16 l    -  256    0    0.000    0.000   0.000
[root@user08-server01 ~]#
```

localhost를 쓰는 것을 확인할 수 있음.