

파일시스템



다룰 내용

- 데이터와 파일시스템
- 파일 시스템 설치 및 관리

데이터와 파일 시스템



컴퓨터에서 다루는 데이터라는 게 물리적으로 뭘 의미하는 거지?



컴퓨터에서 다루는 데이터는 물리적으로는 전기 신호의 형태로 저장되고 전송됩니다.



컴퓨터는 데이터를 이진수(binary)의 형태로 표현합니다. 이진수는 0과 1의 조합으로 이루어져 있으며, 컴퓨터의 전자 회로에서는 0과 1에 대응하는 전기 신호를 생성하고 이를 저장하거나 전송합니다.

데이터를 저장하는 기억 장치(메모리)는 전기 신호를 이용하여 데이터를 기록하고 읽어들 수 있습니다. 하드 디스크, USB 드라이브 등의 저장 장치도 비슷한 원리로 동작합니다.

데이터를 전송하는 경우에도 전기 신호가 사용됩니다. 인터넷을 통해 데이터를 전송할 때는 전기 신호가 네트워크 케이블을 통해 전송되며, 무선 통신을 이용할 때는 전파가 데이터를 전송합니다.

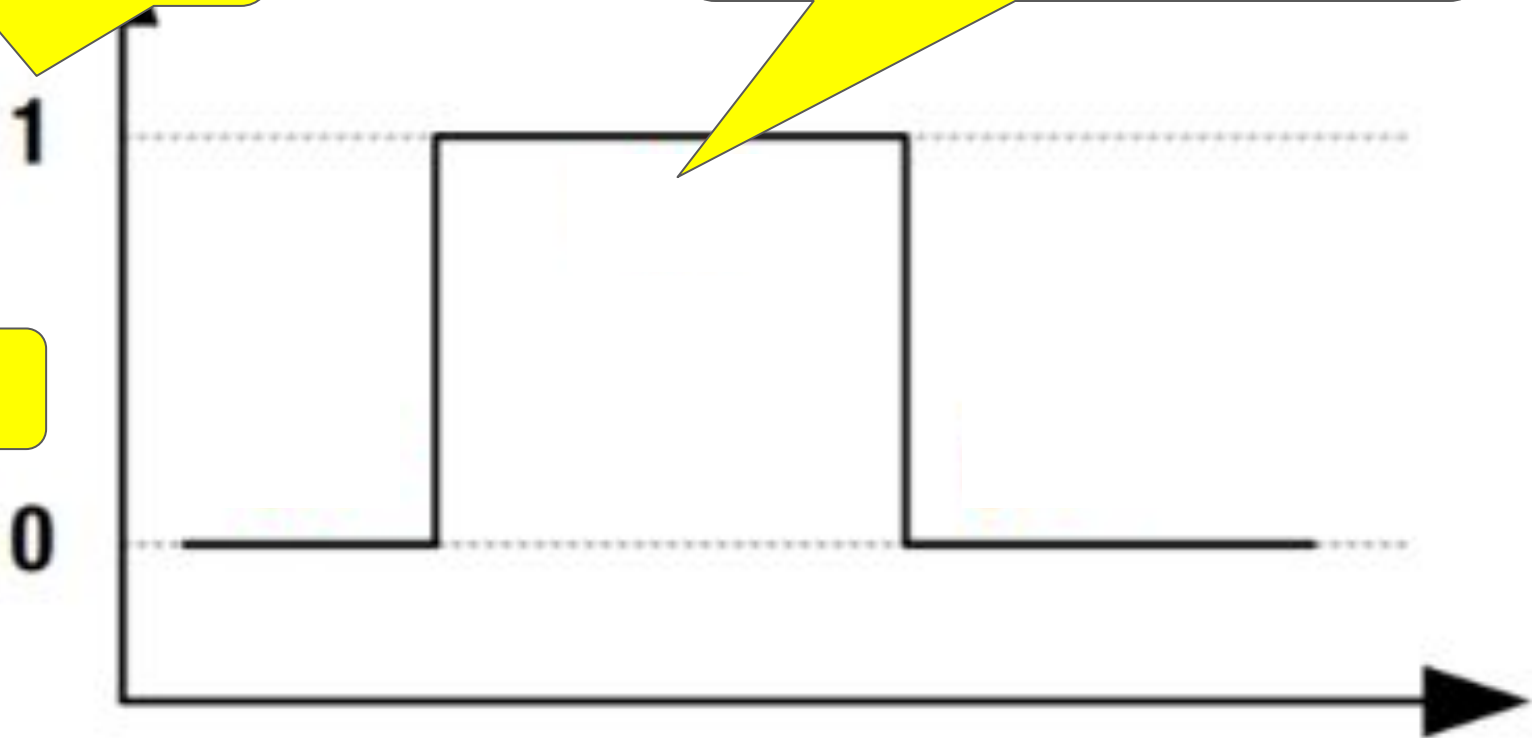
따라서 컴퓨터에서 다루는 데이터는 물리적으로는 전기 신호의 형태로 처리되며, 이를 이진수로 표현하여 저장하고 전송합니다.

컴퓨터의 데이터

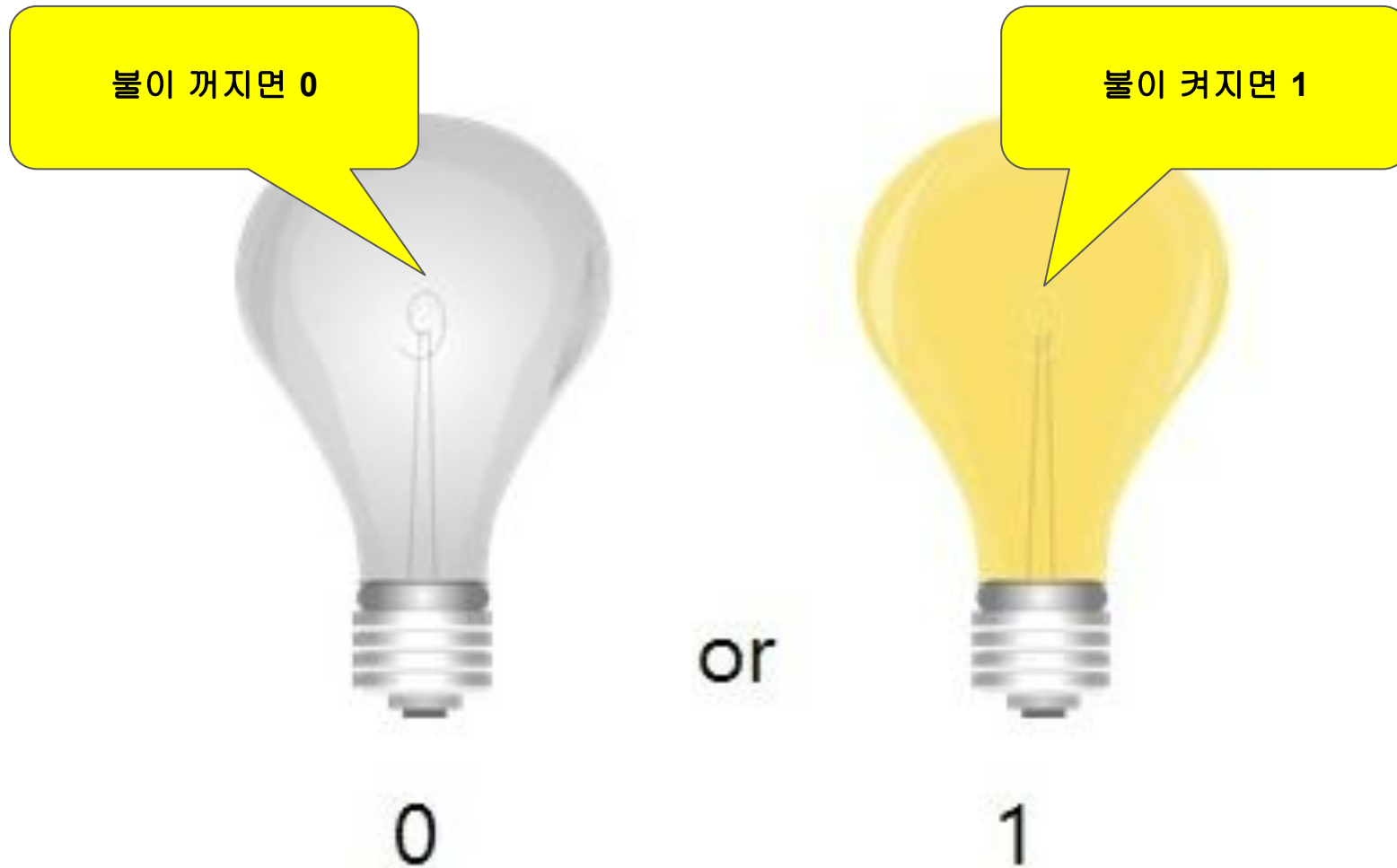
5V 또는 3.3V 또는 1.8V의 전기 신호를
1로 표현

0V의 전기 신호를
0으로 표현

이러한 전기 신호가
컴퓨터 내부의 어딘가에 저장되거나
다른 어딘가로 전송



컴퓨터의 데이터



컴퓨터의 데이터



이 한자리가 1bit

8,796,093,022,208(약 8조 8천억)개의
전기 신호(비트)가 저장

사람이 보는 컴퓨터의 데이터

파일이라는 단위로 접근하고 활용
=> **이름**을 통한 데이터 접근

내 PC

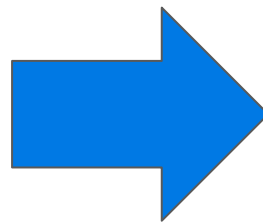
SSD 또는 HDD



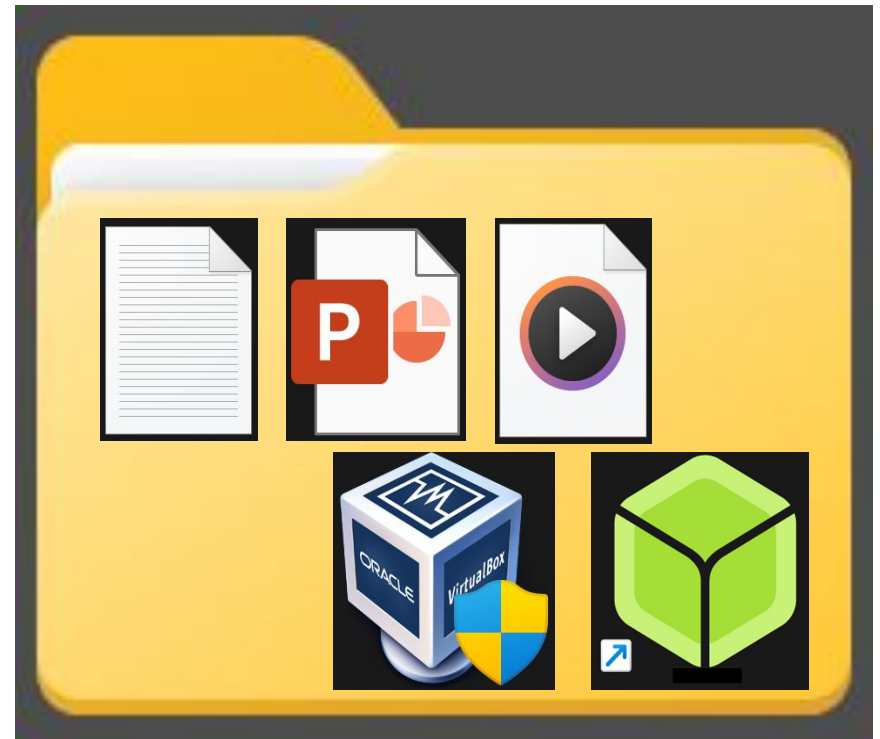
사람이 보는 컴퓨터의 데이터



이게



어떻게



이렇게?

무수히 많은 0과 1의 집합

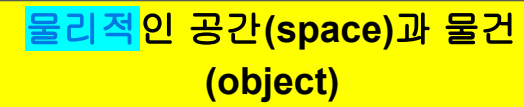
컴퓨터에 존재하는 **물리적 데이터**를

파일이라는 단위로 **접근**하고 **관리**하기 위해 필요한 **논리적 구조**
(이름)

사람을 위한 **인터페이스** 제공

1. 파일 단위의 **데이터 관리**
2. **[저장공간]**이라는 컴퓨터 **자원 관리**

쉽게 떠올릴 수 있는 것은?

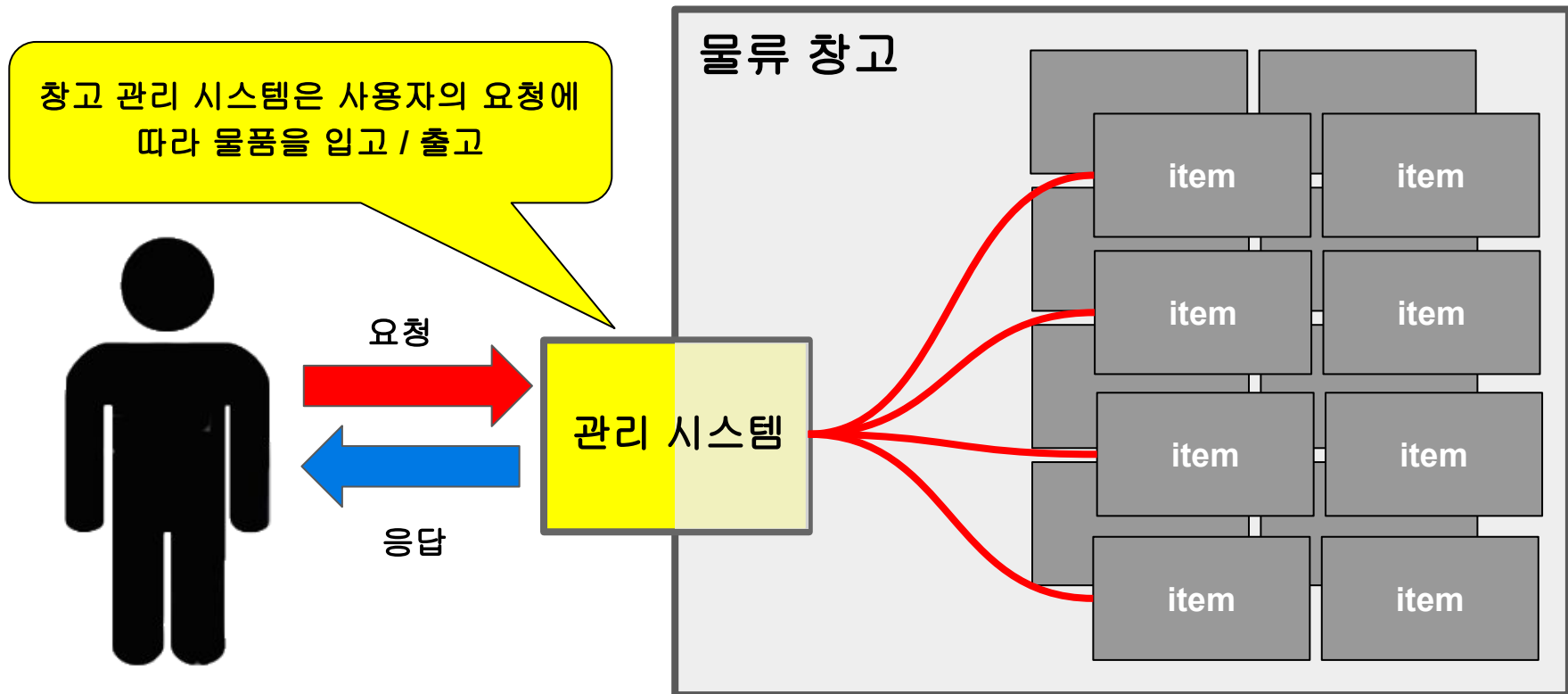


논리적 구조(시스템)

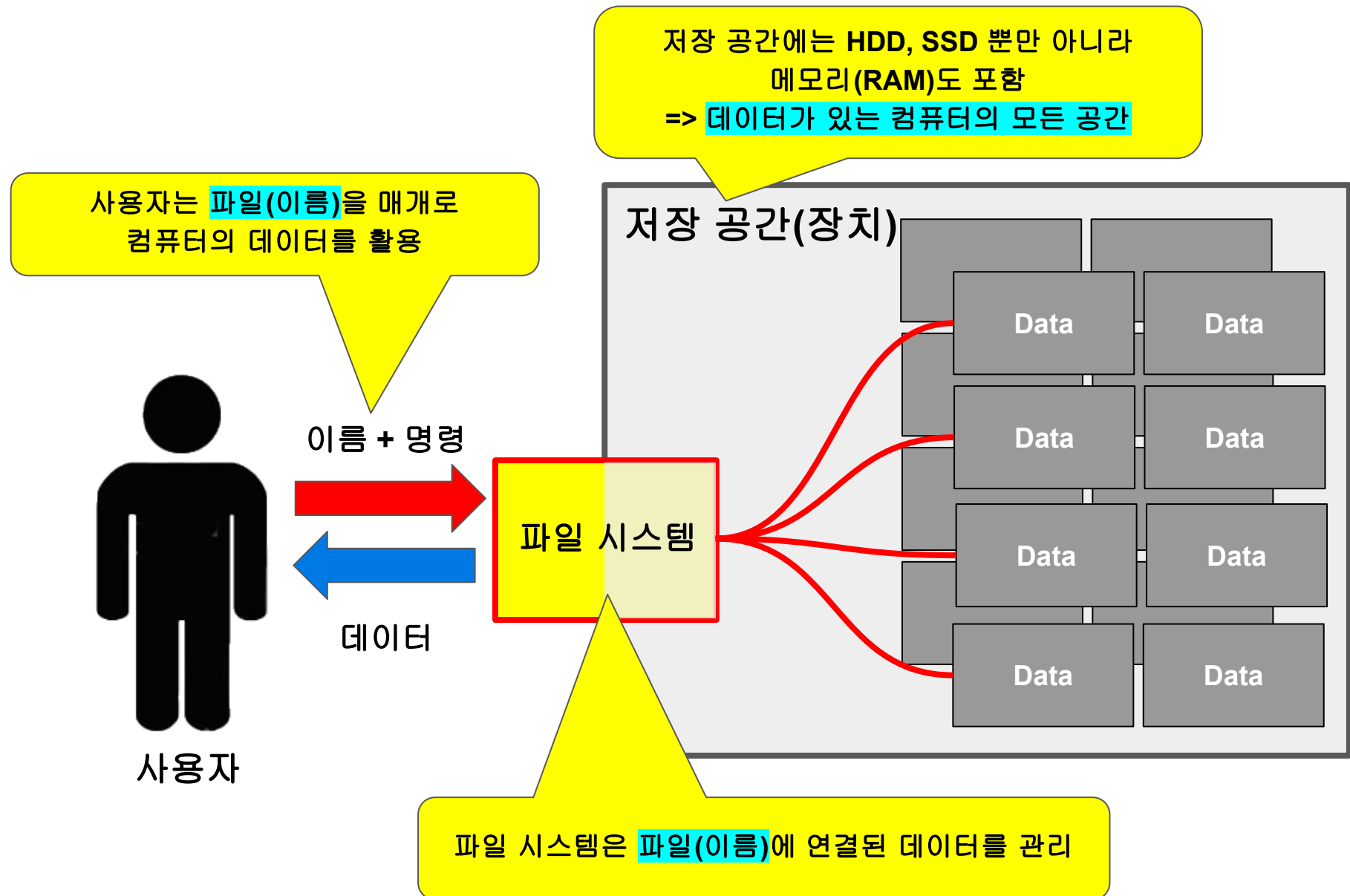
=> 파일 시스템과 비슷



파일 시스템 - 물류 창고와 비교



파일 시스템 - 물류 창고와 비교



리눅스 파일 시스템의 구분

왜 “블록”이라는 이름이 붙었을까?

HDD, SSD, SD 카드,
또는 가상 블록 장치(loop device) 등에 생성

디스크(블록 장치) 기반
파일시스템

메모리(RAM)에 생성

의사(Pseudo)
파일시스템

장치에 저장된 데이터를 파일 단위로 관리
사람이 흔히 접하는 파일 시스템

ext2, 3, 4	리눅스 고유 파일 시스템
NTFS	윈도우 OS 기본 파일 시스템
FAT	주로 SD 카드, USB 메모리 등에 사용

등등...

각 파일 시스템 별 특수한 목적

proc	시스템 상태 정보 제공
sysfs	하드웨어 관련 정보 제공
tmpfs	일시적인 파일 저장 (Power-Off시 삭제)

등등...

리눅스에서 지원하는 파일 시스템

```
root@linux-test:/run# cat /proc/filesystems
```

```
nodev sysfs
nodev rootfs
nodev ramfs
nodev bdev
nodev proc
nodev cpuset
nodev cgroup
nodev cgroup2
nodev tmpfs
nodev devtmpfs
nodev configfs
nodev debugfs
nodev tracefs
nodev securityfs
nodev sockfs
nodev dax
nodev bpf
nodev pipefs
nodev hugetlbfs
nodev devpts
nodev ext3
nodev ext2
nodev ext4
nodev squashfs
nodev vfat
nodev ecryptfs
nodev fuseblk
nodev fuse
nodev fusectl
nodev pstore
nodev mqueue
nodev btrfs
nodev autofs
nodev xenfs
nodev binfmt_misc
```

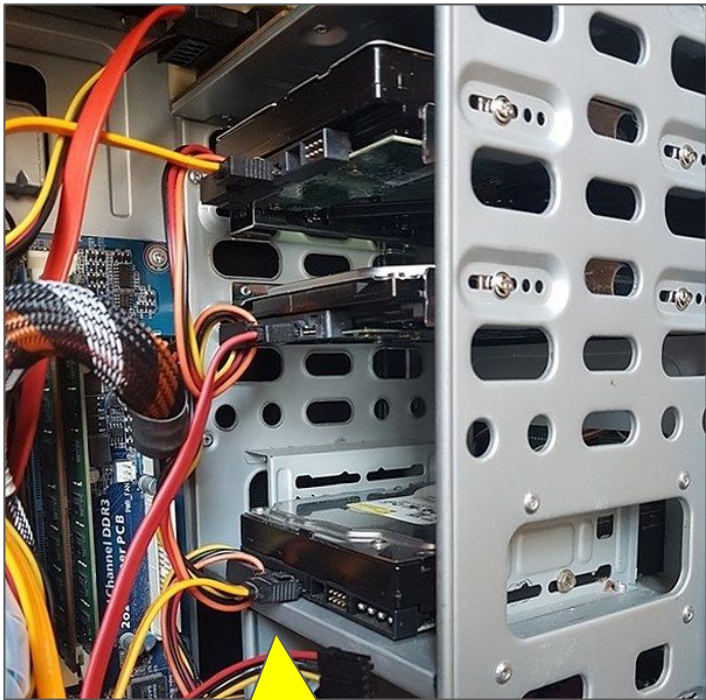
nodev (no-device)
: 의사 파일 시스템을 의미

현재 리눅스 시스템에서 지원하는 파일 시스템 확인
: **cat /proc/filesystems**

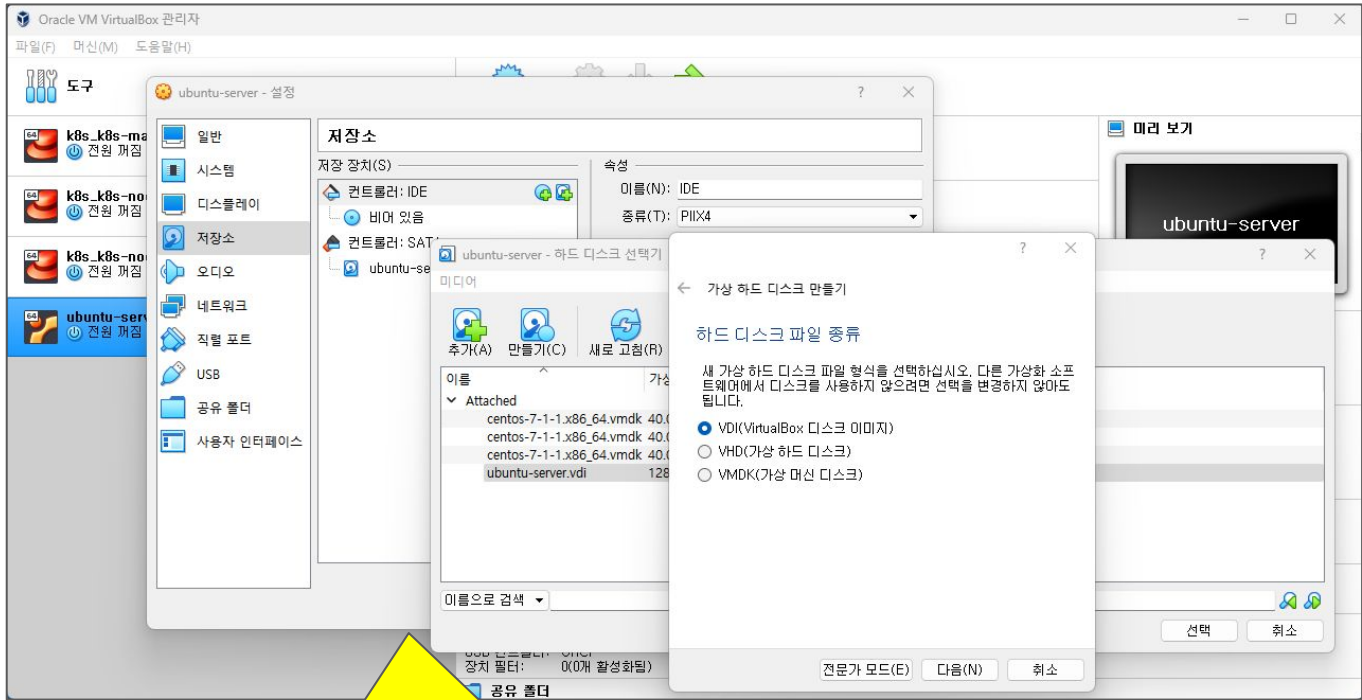
파일 시스템명

파일 시스템 설치 및 관리

디스크 설치

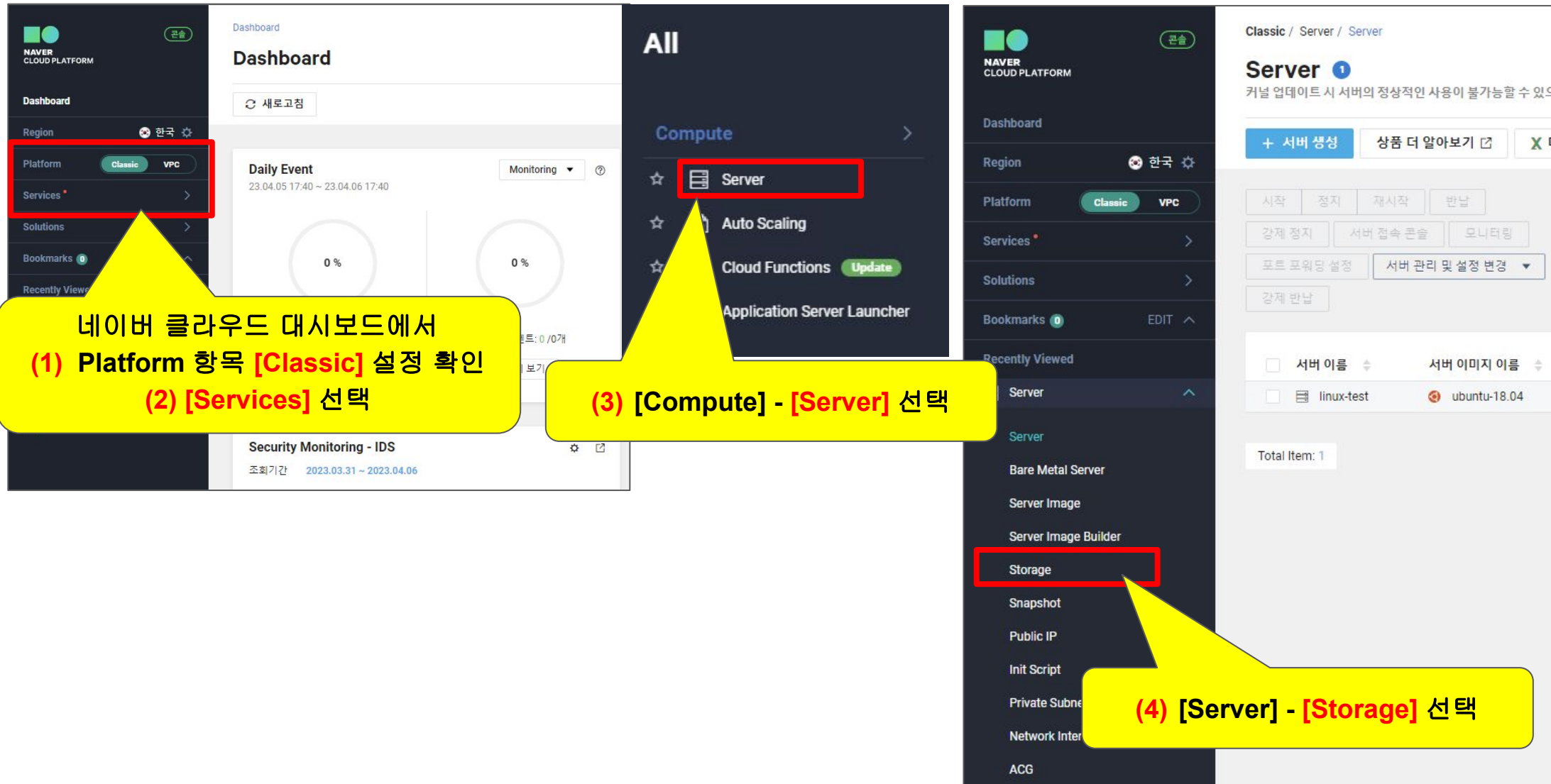


PC에 설치



VirtualBox 등의 가상 머신에 추가

네이버 클라우드 서버에 디스크 추가 (실습)



네이버 클라우드 대시보드에서
(1) Platform 항목 [Classic] 설정 확인
(2) [Services] 선택

(3) [Compute] - [Server] 선택

(4) [Server] - [Storage] 선택

네이버 클라우드 서버에 디스크 추가 (실습)

1) + 스토리지 생성

2) 저렴한 HDD로 선택

3) 이름 지정

4) 적용 서버 꼭 확인

5) 크기는 20GB로 설정 (요금 월 1150원)

6) 추가

7) 확인

스토리지 생성

스토리지 종류

스토리지 이름

Zone

적용 서버 선택

스냅샷 선택

크기

SSD

HDD

hdd1

KR-2

linux-test

(선택 없음)

20

GB 최소 10 GB, 최대 2000 GB

해당 스토리지가 생성됩니다.

생성이 완료되면 서버에 마운트하여 활용하세요. (참조: 스토리지 추가 가이드)

스토리지 이름	hdd1	적용 서버	linux-test
스토리지 종류	HDD	스냅샷	(선택 없음)
스토리지 크기	20 GB	Zone	KR-2
메모			

저장되는 스토리지에 대해서는 요금이 부과됩니다.

점부터 요금이 과금되므로, 사용하지 않을 때는 반드시 반납하시기를 권장드립니다.
미지로 생성된 서버에는 스토리지를 생성할 수 없습니다.

네이버 클라우드 서버에 디스크 추가 (실습)

Classic / Server / Storage

Storage 2

블록 스토리지를 생성하고 관리합니다.

+ 스토리지 생성 상품 더 알아보기 다운로드 새로고침 ▼

서비스 Tibero 서비스 유상 기술 지원 변경 안내 더보기

스토리지 삭제 스냅샷 생성 서버에 연결 서버에 연결 해제 크기 변경

스토리지 이름 필터 Zone: ✓ 전체 스토리지: ✓ 전체 상태: ✓ 전체

스토리지 이름	종류	크기	IOPS	생성 일시	상태	ZONE	연결정보
hdd1	HDD	20 GB		2023-04-06 오후 6:28 (UTC+09:00)	사용중	KR-2	linux-test:/dev/xvdb
linux-test의 기본 스토리지	SSD	50 GB	4000	2023-02-09 오전 9:39 (UTC+09:00)	사용중	KR-2	linux-t

Total Item: 2

1

디스크 생성 확인
=> 서버명 (linux-test) 및 블록 장치 경로 (/dev/xvdb) 확인

```
root@linux-test:~# ls -al /dev/xvd*  
brw-rw---- 1 root disk 202, 0 Apr 6 14:09 /dev/xvda  
brw-rw---- 1 root disk 202, 1 Apr 6 14:09 /dev/xvda1  
brw-rw---- 1 root disk 202, 16 Apr 6 18:28 /dev/xvdb  
root@linux-test:~#
```

기존 디스크 및
파티션

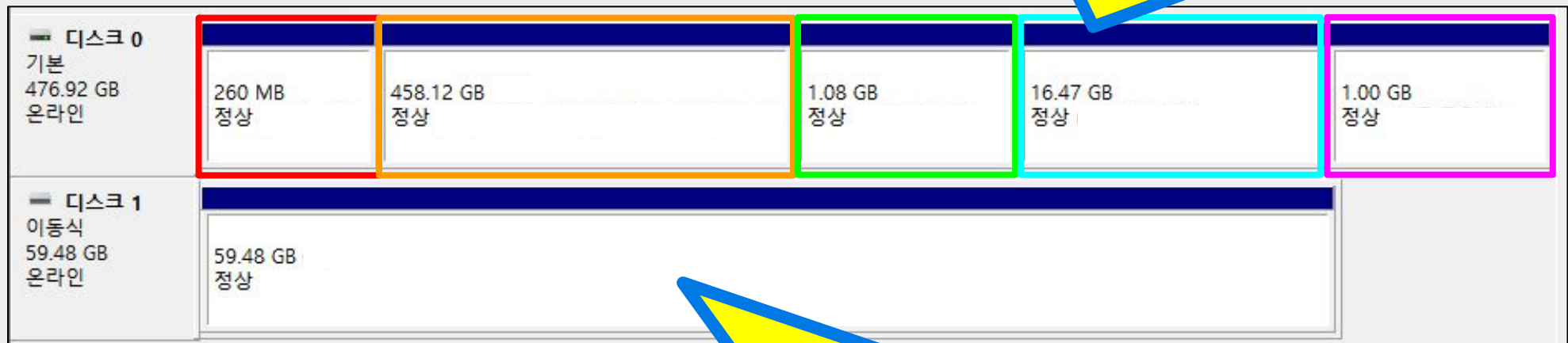
추가된 디스크

디스크의 파티션

파티션(Partition) : 논리적으로 **분할된** 디스크의 저장 공간

=> 디스크에 파일 시스템 생성을 위해 필요

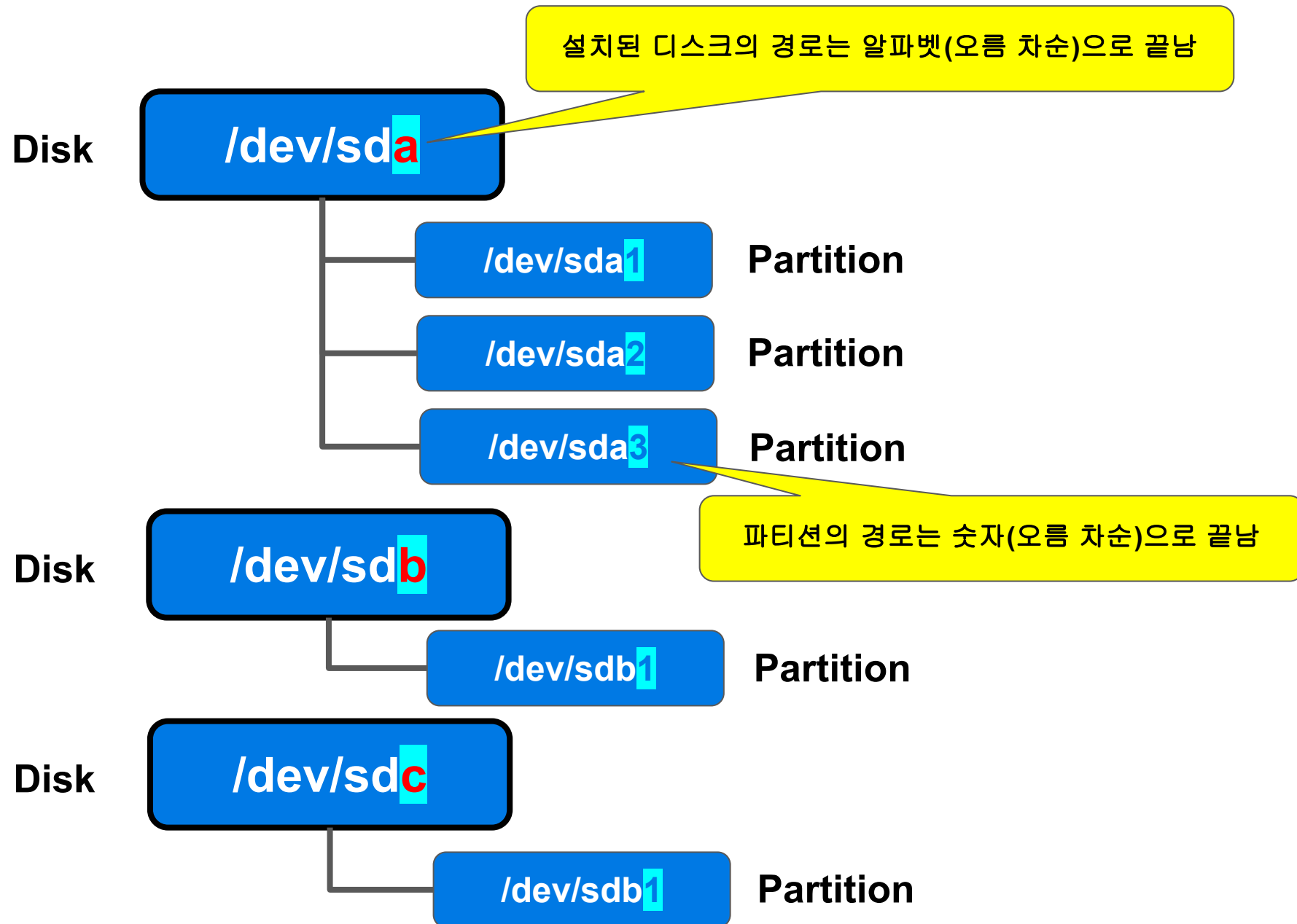
하나의 디스크가 **5개 파티션**으로 구성



하나의 디스크가 **단일 파티션**으로 구성

파티션 단위로 파일 시스템 생성 (포맷)

리눅스의 디스크 및 파티션 표현



파티션 생성

● fdisk

- **기능** : 디스크의 파티션을 관리(생성, 삭제, 조회 등)
- **형식** : **fdisk** [옵션] [디스크 장치 경로]

1. 현재 시스템에 연결된 디스크와 파티션 정보 출력 : **fdisk -l**

```
root@linux-test:~# fdisk -l
```

```
Disk /dev/xvda: 50 GiB, 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
Disklabel type: dos
Disk identifier: 0xae4f48bba
```

디스크(/dev/xvda) 정보

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/xvda1	*	2048	104855551	104853504	50G	83	Linux

디스크의 파티션(/dev/xvda1) 정보

```
Disk /dev/xvdb: 20 GiB, 21474836480 bytes, 41943040 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
root@linux-test:~#
```

디스크의 파티션(/dev/xvdb) 정보

파티션 생성

• fdisk

2. /dev/xvdb 디스크에 2개 파티션 생성 : **fdisk /dev/xvdb**

```
root@linux-test:~# fdisk /dev/xvdb

Welcome to fdisk (util-linux 2.31.1).
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.
Created a new DOS disklabel with disk identifier 0xbc1d7134.

Command (m for help): m
```

m : 도움말 출력

```
Partition type
  p  primary (0 primary, 0 extended, 4 free)
  e  extended (container for logical partitions)
Select (default p):
```

파티션 타입 선택
p : Primary 파티션 선택

```
Help:

DOS (MBR)
a  toggle a bootable flag
b  edit nested BSD disklabel
c  toggle the dos compatibility flag

Generic
d  delete a partition
F  list free unpartitioned space
l  list known partition types
n  add a new partition
p  print the partition table
t  change a partition type
v  verify the partition table
i  print the partition table

Misc
m  print this help message
u  change display/entry units
x  extra functionality (experts only)

Script
I  load disk layout from sfdisk script file
O  dump disk layout to sfdisk script file

Save & Exit
w  write table to disk and exit
q  quit without saving changes

Create a new label
g  create a new empty GPT partition table
G  create a new empty SGI (IRIX) partition table
o  create a new empty DOS partition table
s  create a new empty Sun partition table

Command (m for help):
```

[n : 새로운 파티션 추가] 선택

파티션 생성

● fdisk

```
Partition number (1-4, default 1): 1
First sector (2048-41943039, default 2048): 2048
Last sector, +sectors or +size{K,M,G,T,P} (2048-41943039, default 41943039): +10G
Created a new partition 1 of type 'Linux' and of size 10 GiB.
Command (m for help):
```

첫번째 파티션의 번호 입력
(기본값 1)

첫번째 파티션의 시작 섹터 입력
(기본값 2048)

파티션 1 생성

GB와 GiB의 차이는?

파티션에 할당할 크기 입력
(+10G)

```
Command (m for help): n
Partition type
  p   primary (1 primary, 0 extended, 3 free)
  e   extended (container for logical partitions)
Select (default p): Enter!
Using default response p.
Partition number (2-4, default 2): Enter!
First sector (20973568-41943039, default 20973568): Enter!
Last sector, +sectors or +size{K,M,G,T,P} (20973568-41943039, default 41943039): Enter!
Created a new partition 2 of type 'Linux' and of size 10 GiB.
```

남은 공간에 대한 파티션 생성

나머지 입력 항목은 모두 Enter
=> 모두 기본값(default)으로 셋팅

파티션 2 생성

파티션 생성

- fdisk

p : 파티션 테이블 출력

```
Command (m for help): p
```

```
Disk /dev/xvdb: 20 GiB, 21474836480 bytes, 41943040 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
Disklabel type: dos
Disk identifier: 0xbbd924d5
```

디스크 및 파티션 정보 출력

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/xvdb1		2048	20973567	20971520	10G	83	Linux
/dev/xvdb2		20973568	41943039	20969472	10G	83	Linux

파티션 테이블 저장 후 종료

```
Command (m for help): w
```

```
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```


파일 시스템 생성

디스크 기반 파일 시스템 생성 명령

```
root@linux-test:~# ls -al /sbin/mk*fs*
lrwxrwxrwx 1 root root      8 Jan 25  2017 /sbin/mkdosfs -> mkfs.fat
-rwxr-xr-x 1 root root 129344 Jun  2  2022 /sbin/mke2fs
-rwxr-xr-x 1 root root  10312 Sep 17  2020 /sbin/mkfs
-rwxr-xr-x 1 root root  30800 Sep 17  2020 /sbin/mkfs.bfs
-rwxr-xr-x 1 root root  34824 Sep 17  2020 /sbin/mkfs.cramfs
lrwxrwxrwx 1 root root      6 Jun  2  2022 /sbin/mkfs.ext2 -> mke2fs
lrwxrwxrwx 1 root root      6 Jun  2  2022 /sbin/mkfs.ext3 -> mke2fs
lrwxrwxrwx 1 root root      6 Jun  2  2022 /sbin/mkfs.ext4 -> mke2fs
-rwxr-xr-x 1 root root  35328 Jan 25  2017 /sbin/mkfs.fat
-rwxr-xr-x 1 root root  79960 Sep 17  2020 /sbin/mkfs.minix
lrwxrwxrwx 1 root root      8 Jan 25  2017 /sbin/mkfs.msdos -> mkfs.fat
lrwxrwxrwx 1 root root      6 Nov  1 20:57 /sbin/mkfs.ntfs -> mkntfs
lrwxrwxrwx 1 root root      8 Jan 25  2017 /sbin/mkfs.vfat -> mkfs.fat
-rwxr-xr-x 1 root root 433688 Apr 18  2018 /sbin/mkfs.xfs
-rwxr-xr-x 1 root root  79984 Nov  1 20:57 /sbin/mkntfs
```

각 파일 시스템에 따른 명령어 구성

파일 시스템 생성

• mkfs

- **기능** : 옵션으로 지정한 파일 시스템을 디스크 파티션에 생성
- **형식** : **mkfs** **[옵션]** **[파티션 장치 경로]**

-t 옵션으로 파일 시스템 지정
(default는 ext2로 지정)

1. /dev/xvdb1에 ext4 파일 시스템 생성 : **mkfs -t ext4 /dev/xvdb1**

mkfs.ext4 /dev/xvdb1
mke2fs -t ext4 /dev/xvdb1
도 같은 동작 수행

```
root@linux-test:~# mkfs -t ext4 /dev/xvdb1
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 2621440 4k blocks and 655360 inodes
Filesystem UUID: 07cc02b7-fe7d-4181-9ae1-fe7c124d2b47
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

파일 시스템 생성

- mkfs

mkfs -t vfat -F 32 /dev/xvdb2 와 동일

2. /dev/xvdb2에 FAT32파일 시스템 생성 : **mkfs.vfat -F 32 /dev/xvdb2**

```
root@linux-test:~# mkfs.vfat -F 32 /dev/xvdb2
mkfs.fat 4.1 (2017-01-24)
root@linux-test:~#
```


리눅스의 파티션 및 파일 시스템 확인

● lsblk

- **기능** : 사용 가능한 블록 장치의 리스트 출력
- **형식** : **lsblk** **[옵션]** **[블록 장치 경로]**

디스크와 파티션

1. 현재 시스템에 연결된 디스크와 파티션 정보 출력 : **lsblk**

```
root@linux-test:~# 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   20G  0 disk
├─xvdb1     202:17   0   10G  0 part
└─xvdb2     202:18   0   10G  0 part
root@linux-test:~#
```

2. 파티션의 파일시스템 관련 정보 출력 : **lsblk -f**

```
root@linux-test:~# lsblk -f
NAME        FSTYPE LABEL  UUID                                  MOUNTPOINT
xvda
└─xvda1     ext4              8d992262-a389-4a90-b69a-8c010594a44a /
xvdb
├─xvdb1     ext4              f7239dab-5d1e-4771-9b48-6eca96efd95e
└─xvdb2     vfat              61CB-3924
```

Ext 파일 시스템 - 메타데이터

메타 데이터(Metadata) : 파일 시스템, 파일 관련 정보(Info) 저장

실제 구조는 이거보다 복잡

Ext 파일 시스템의 추상적 구조

- Meta Area -

- Data Area -

Super Block
: 파일 시스템 관련 정보

Super Block

inode
: 개별 파일 관련 정보

inode

다른 메타 데이터도 있지만
일단 이 정도만 기억하자

/dev/xvdb

/dev/xvdb1 (ext4)

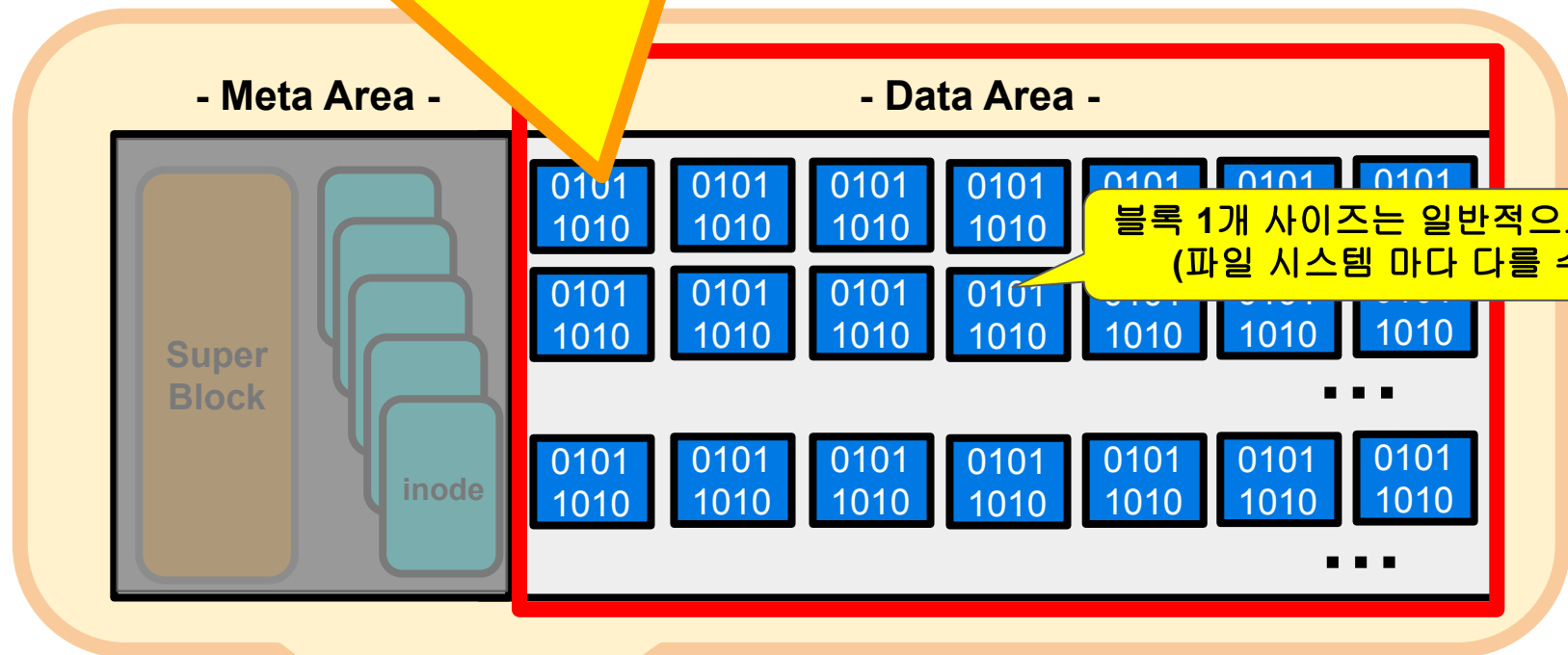
/dev/xvdb2 (vfat)

Ext 파일 시스템 - 데이터 블록

데이터 블록(Data Block)

: 파일 시스템이 디스크(저장장치)에 데이터를 저장하는 단위

왜 [블록 장치]라고 불리는가에
대한 이유



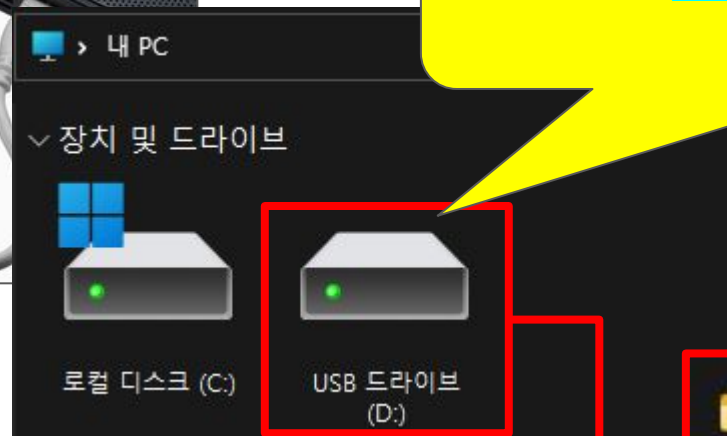
/dev/xvdb

/dev/xvdb1 (ext4)

/dev/xvdb2 (vfat)

파일 시스템의 마운트 (mount)

윈도우 PC에서 디스크를 추가하는 경우



추가된 디스크에 **드라이브 문자 (D, E ...)**를 할당
[내 PC]에 연결된 디스크로 표시
=> 상당히 직관적

디렉토리1
 디렉토리2
 파일1
 파일2

파일 시스템의 마운트 (mount)

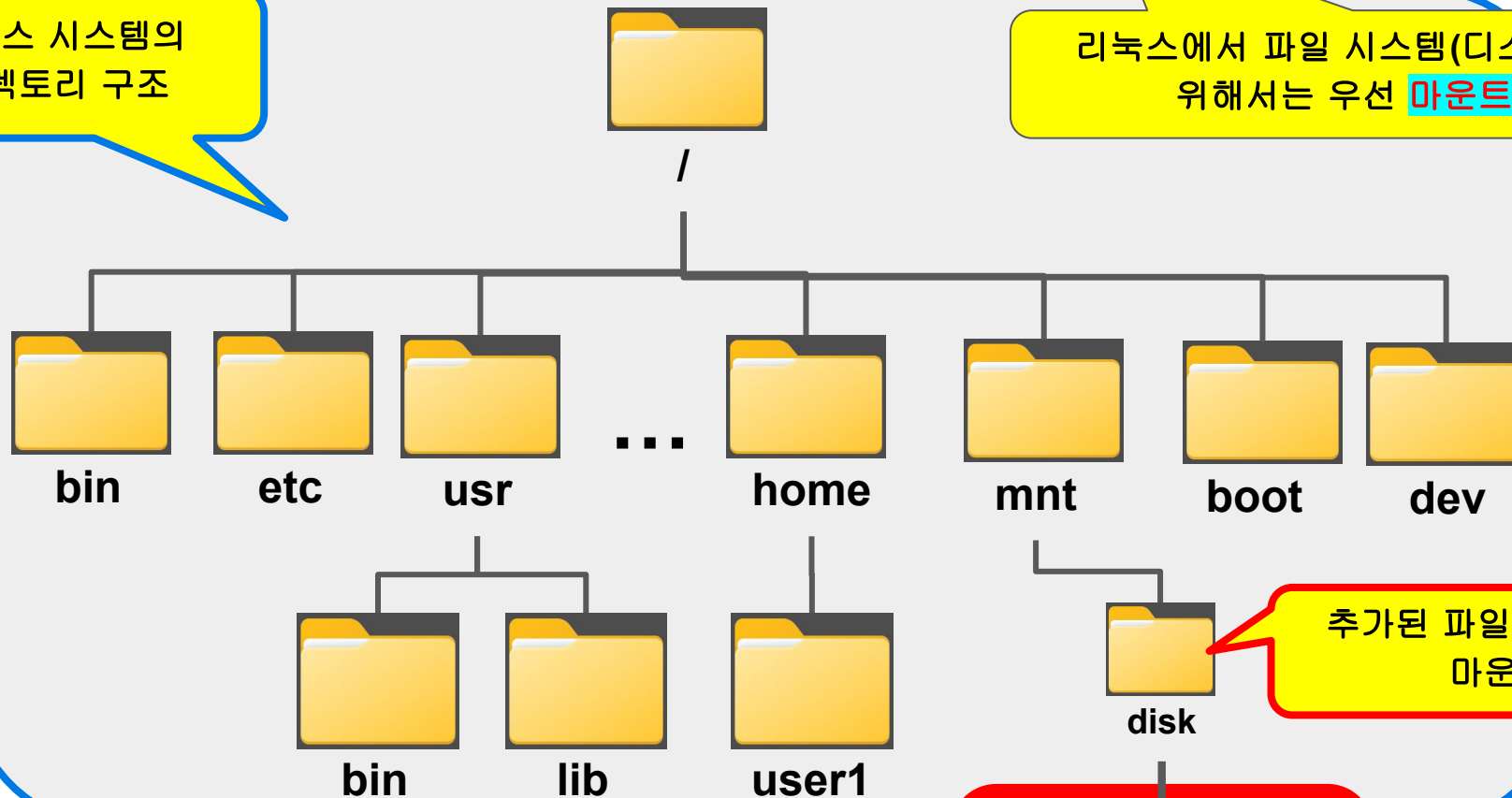
처음 리눅스를 접할 때
어려운 부분 중 하나

마운트 (mount) : 파일 시스템(이 적용된 디스크)를 디렉토리 계층 구조에 연결

=> 사용자가 파일 시스템에 접근하기 위해 필요

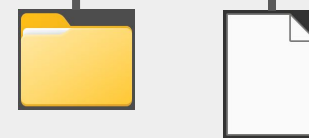
리눅스 시스템의
디렉토리 구조

리눅스에서 파일 시스템(디스크)를 사용하기
위해서는 우선 **마운트**해야 한다



추가된 파일 시스템(디스크)의
마운트 포인트

추가된 파일 시스템(디스크)



파일 시스템의 마운트 (mount)

● mount

- **기능** : 파일 시스템을 디렉토리에 연결
- **형식** : **mount** [옵션] [장치 경로] [디렉토리 경로]

1. 현재 마운트 되어 있는 파일 시스템 출력 : **mount**

```
root@linux-test:~# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=978108k,nr_inodes=244527,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
/dev/xvda1 on / type ext4 (rw,mode=755)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime)
```

/dev/xvda1은 ext4 타입이며
root directory (/)에 마운트 되어 있음

(하락)

파일 시스템의 마운트 (mount)

- mount

2. /dev/xvdb1, /dev/xvdb2를 마운트 : `mount -t ext /dev/xvdb1 /mnt/disk1`
`mount -t vfat /dev/xvdb2 /mnt/disk2`

```
root@linux-test:~# mkdir -p /mnt/disk1 /mnt/disk2
root@linux-test:~# ls /mnt
disk1 disk2
root@linux-test:~#
root@linux-test:~# mount -t ext4 /dev/xvdb1 /mnt/disk1
root@linux-test:~# mount -t vfat /dev/xvdb2 /mnt/disk2
root@linux-test:~#
root@linux-test:~# 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   20G  0 disk
├─xvdb1     202:17   0   10G  0 part /mnt/disk1
└─xvdb2     202:18   0   10G  0 part /mnt/disk2
root@linux-test:~#
```

파일 시스템을 마운트 할
디렉토리(마운트 포인트) 생성

파일 시스템을 마운트
(파일 시스템 타입과 장치/ 마운트 경로에 주의)

xvdb1,2가 디렉토리에 마운트 됨

파일 시스템의 마운트 (mount)

2. 마운트된 디렉토리로 이동하여 파일 복사 확인

```
root@linux-test:~# cd /mnt/disk1
```

마운트된 디렉토리(/mnt/disk1)로 이동

```
root@linux-test:/mnt/disk1# ls
```

```
lost+found
```

```
root@linux-test:/mnt/disk1# cp /etc/hosts ./
```

/mnt/disk1에 파일 복사

```
root@linux-test:/mnt/disk1# ls
```

```
hosts lost+found
```

```
root@linux-test:/mnt/disk1#
```

```
root@linux-test:/mnt/disk1# cd /mnt/disk2
```

마운트된 디렉토리(/mnt/disk2)로 이동

```
root@linux-test:/mnt/disk2# ls
```

```
root@linux-test:/mnt/disk2#
```

```
root@linux-test:/mnt/disk2# cp /etc/hosts ./
```

/mnt/disk2에 파일 복사

```
root@linux-test:/mnt/disk2# ls
```

```
hosts
```

```
root@linux-test:/mnt/disk2# ls -al
```

```
total 20
```

```
drwxr-xr-x 2 root root 8192 Apr 10 11:43 .
```

```
drwxr-xr-x 4 root root 4096 Apr 10 10:57 ..
```

```
-rwxr-xr-x 1 root root 189 Apr 10 11:43 hosts
```

```
root@linux-test:/mnt/disk2#
```

파일 시스템의 마운트 해제 (umount)

● umount

- **기능** : 파일 시스템의 마운트 해제
- **형식** : **umount** [옵션] [마운트 포인트 경로]

1. /mnt/disk1, /mnt/disk2를 마운트 해제 : **umount /mnt/disk1**
umount /mnt/disk2

```
root@linux-test:/mnt/disk2# cd ~
root@linux-test:~#
root@linux-test:~# umount /mnt/disk1
root@linux-test:~# umount /mnt/disk2
root@linux-test:~#
root@linux-test:~# 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   20G  0 disk
├─xvdb1      202:17   0   10G  0 part
└─xvdb2      202:18   0   10G  0 part
```

umount 실행 전에 먼저 마운트 된 디렉토리에서 벗어나야 함

umount 실행 (마운트 해제)

xvdb1,2에 대한 마운트 포인트가 없어짐 확인

마운트 설정 파일

/etc/fstab : 파일 시스템의 마운트 설정 정보

/etc/fstab에 등록된 정보에 따라
리눅스 부팅 시 자동으로 마운트

```
root@linux-test:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>          <dump>  <pass>
# / was on /dev/xvda1 during installation
UUID=8d992262-a389-4a90-b69a-8c010594a44a /          ext4      errors=remount-ro 0      1
root@linux-test:~#
```

/dev/xvda1의 UUID(Universally Unique Identifier)

/dev/xvda1는 부팅 시에 root directory(/) 에 마운트 됨

마운트 설정 파일

/dev/xvdb1,2가 부팅 시 자동으로 마운트 되도록 /etc/fstab 추가 (실습)

```
root@linux-test:~# nano /etc/fstab
```

/etc/fstab 파일 편집

GNU nano 2.9.3 /etc/fstab

/etc/fstab: static file system information.

<file system>	<mount point>	<type>	<options>	<dump>	<pass>
마운트할 장치 파일 (또는 UUID)	마운트할 디렉토리 경로	파일시스템 타입	마운트시 적용 옵션	덤프 설정	파일 점검 옵션

```
# / was on /dev/xvda1 during installation
UUID=8-002262-3280-4b00-b60a-8e01050-344b / ext4 defaults,errors=remount-ro 0 1
/dev/xvdb1 /mnt/disk1 ext4 defaults 1 2
/dev/xvdb2 /mnt/disk2 vfat defaults 0 0
```

<options> 주요 항목

defaults : 다음 5개 항목 모두 적용
auto : 부팅 시 자동으로 마운트
rw : 읽기/쓰기 모두 가능하도록 마운트
nouser : root 계정만 마운트 가능하도록 설정
exec : 파일 실행 허용
suid : SetUID와 SetGID 허용

<dump> 설정 값

0 : 덤프되지 않는 파일 시스템
1 : 덤프 가능한 파일 시스템

<pass> 설정 값

0 : 부팅 시 파일 시스템 점검하지 않음
1 : root 파일 시스템 점검
2 : root 이외의 파일 시스템 점검

마운트 설정 파일

시스템 재부팅 후 파일 시스템 자동 마운트 확인

```
root@linux-test:~# shutdown -r now
Connection to 106.10.32.76 closed by remote host.
Connection to 106.10.32.76 closed.
```

시스템 재부팅 명령
(ssh 로그아웃)

<input type="checkbox"/> 서버 이름	서버 이미지 이름	서버 구성	상태	비공인 IP	공인 IP	ZONE	모니터링	Network 모니터링
<input type="checkbox"/> linux-test	ubuntu-18.04	[Compact] 1vCPU, ...	● 재시작	10.41.134.235		KR-2	기본	해제

<input type="checkbox"/> 서버 이름	서버 이미지 이름	서버 구성	상태	비공인 IP	공인 IP	ZONE	모니터링	Network 모니터링
<input type="checkbox"/> linux-test	ubuntu-18.04	[Compact] 1vCPU, ...	● 운영중	10.41.134.235		KR-2	기본	해제

```
PS C:\Users\magicnotebook> ssh root@106.10.32.76 -p 1024
root@106.10.32.76's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-118-generic x86_64)
```

ssh 로그인

(중략)

```
root@linux-test:~# lsblk -f
NAME        FSTYPE LABEL UUID                                MOUNTPOINT
xvda
└─xvda1 ext4      8d992262-a389-4a90-b69a-8c010594a44a /
xvdb
├─xvdb1 ext4      f7239dab-5d1e-4771-9b48-6eca96efd95e /mnt/disk1
└─xvdb2 vfat       61CB-3924 /mnt/disk2
root@linux-test:~#
```

파일 시스템 마운트 확인

디스크 관리 명령

● df (Disk Free)

- **기능** : 파일 시스템의 용량 정보를 출력
- **형식** : df [옵션]

1. 파일 시스템 용량 정보를 옵션 추가하여 출력: **df -hT**

```
root@linux-test:~# df -hT
Filesystem      Type      Size  Used Avail Use% Mounted on
udev            devtmpfs  956M   0    956M   0% /dev
tmpfs           tmpfs     198M  700K   197M   1% /run
/dev/xvda1      ext4      49G   3.1G   44G    7% /
tmpfs           tmpfs     988M   0    988M   0% /dev/shm
tmpfs           tmpfs     5.0M   0     5.0M   0% /run/lock
tmpfs           tmpfs     988M   0    988M   0% /sys/fs/cgroup
/dev/xvdb1      ext4      9.8G   37M   9.3G   1% /mnt/disk1
/dev/xvdb2      vfat      10G   16K   10G    1% /mnt/disk2
tmpfs           tmpfs     198M   0    198M   0% /run/user/1000
tmpfs           tmpfs     198M   0    198M   0% /run/user/0
```

-T : 파일 시스템 타입 표시

-h : 용량 단위 표시

(Size : 총 용량
Used : 사용량
Avail : 사용 가능 용량)

주요 옵션

옵션	설명
-h	용량 단위(KB, MB, GB) 표시
-k	KB 단위로 표시
-m	MB 단위로 표시
-T	각 파티션에 대한 파일 시스템 타입 표시
-i	아이노드(i-node) 사용량 표시

디스크 관리 명령

• du (Disk Usage)

- **기능** : 파일 및 디렉토리의 파일 시스템(디스크) 사용량을 출력
- **형식** : **du** [옵션] [파일 또는 디렉토리]

1. 현재 디렉토리 내의 파일 및 디렉토리별 디스크 사용량 출력 : **du -h**

```
root@linux-test:~# du -h
4.0K  ./nsight
4.0K  ./gnupg/private-keys-v1.d
8.0K  ./gnupg
4.0K  ./cache
8.0K  ./vim
8.0K  ./local/share/nano
12K   ./local/share
16K   ./local
8.0K  ./ssh
```

2. /boot 디렉토리의 전체 디스크 사용량 출력 : **du -sh /boot**

```
root@linux-test:~# du -sh /boot
147M  /boot
```

주요 옵션

옵션	설명
-h	용량 단위(KB, MB, GB) 표시
-b	Byte 단위로 표시
-k	KB 단위로 표시
-m	MB 단위로 표시
-s	파일들의 전체 크기를 합한 값만 표시

디스크 관리 명령

● fsck (File System Check)

- **기능** : 리눅스의 파일 시스템을 점검하고 복구
- **형식** : **fsck** [옵션] [파티션 장치 경로]

1. /dev/xvdb1에 대한 파일 시스템 점검 : **fsck /dev/xvdb1**

파일 시스템이 마운트 되어 있는 경우 먼저
해제

```
root@linux-test:~# umount /mnt/disk1
```

```
root@linux-test:~# fsck /dev/xvdb1
```

/dev/xvdb1에 대한 파일 시스템 점검

```
fsck from util-linux 2.31.1
```

```
e2fsck 1.44.1 (24-Mar-2018)
```

```
/dev/xvdb1: clean, 12/655360 files, 66754/2621440 blocks
```

```
root@linux-test:~#
```

```
root@linux-test:~# fsck -f /dev/xvdb1
```

-f : 강제 실행 (파일 시스템이 손상된 경우에만 사용 권장)

```
fsck from util-linux 2.31.1
```

```
e2fsck 1.44.1 (24-Mar-2018)
```

```
Pass 1: Checking inodes, blocks, and sizes
```

```
Pass 2: Checking directory structure
```

```
Pass 3: Checking directory connectivity
```

```
Pass 4: Checking reference counts
```

```
Pass 5: Checking group summary information
```

```
/dev/xvdb1: 12/655360 files (0.0% non-contiguous), 66754/2621440 blocks
```

```
root@linux-test:~#
```

스왑(swap) 영역 생성 및 관리

스왑(swap) 영역



swap에 대해서 좀 더 자세히 알려줘



Swap은 시스템의 메모리 부족 상황에서 사용되는 보조 메모리입니다. 메모리가 부족해지면 운영체제는 더 이상 사용되지 않는 메모리 페이지를 디스크의 스왑 공간에 저장하여 빈 메모리 공간을 확보합니다. 이러한 swap 공간을 사용하여 시스템이 메모리 부족 상황에서도 작동할 수 있도록 합니다.

하지만, swap은 메모리보다 디스크에 저장되므로, 디스크의 입출력 속도가 매우 느리기 때문에 메모리보다 훨씬 느리게 작동합니다. 이러한 이유로 swap을 사용하는 것은 성능 저하를 초래할 수 있습니다.

또한, swap을 사용하는 경우에는 메모리 부족 상황에서도 시스템이 계속해서 동작하지만, swap을 사용할 경우 시스템이 지연될 가능성이 높아집니다. 이는 메모리가 부족해지면 운영체제가 디스크로 데이터를 스왑하고 다시 메모리로 데이터를 읽어와야 하기 때문입니다.

따라서, 일반적으로 서버에서는 swap을 최소화하거나 비활성화하는 것이 좋습니다. 그러나, 메모리 사용이 매우 높은 경우에는 swap을 사용하여 시스템이 멈추지 않도록 할 수 있습니다.

스왑(swap) 영역 생성 및 관리

● mkswap

- **기능** : 스왑 파티션 또는 파일을 생성
- **형식** : **mkswap** [옵션] [스왑 파티션 경로] [크기]

실습은 진행하지 않습니다!

swap에 대한 개념과
명령어만 기억하세요

● swapon

- **기능** : 스왑 영역 활성화
- **형식** : **swapon** [옵션] [스왑 파티션 경로]

● swapoff

- **기능** : 활성화된 스왑 영역 중지
- **형식** : **swapoff** [옵션] [스왑 파티션 경로]

쿠버네티스 관련 서적에서 설치
과정을 다룰 때 **swapoff** 명령을
호출하는 경우가 있음

사용 예

mkswap /dev/sdb2 10240

: /dev/sdb2 파티션에 10MiB의 스왑 영역 생성

swapon /dev/sdb2

: /dev/sdb2를 활성화

swapon -s

: 활성화된 스왑 영역 상태를 출력

swapoff -a

: 활성화된 모든 스왑 영역 중지

의사 파일 시스템 - proc

proc 파일 시스템 : 보통 /proc에 마운트

시스템 관련 정보를 파일 형태로 제공

```
root@linux-test:~# ls /proc
```

1	15	1768	24	33	406	645	8	879	acpi	dma	kallsyms	mdstat	schedstat	thread-self
10	15104	18	25	343	407	646	80	882	buddyinfo	driver	kcore	meminfo	scsi	timer_list
1001	15239	19	26	345	409	647	81	897	bus	execdomains	keys	misc	self	tty
1016	153	1903	27	346	410	648	82	899	cgroups	fb	key-users	modules	slabinfo	uptime
1018	15371	199	28	36	424	649	83	9	cmdline	filesystems	kmsg	mounts	softirqs	version
11	15437	2	288	37	426	653	835	900	consoles	fs	kpagecgroup	mtrr	stat	version_signature
113	16	20	29	38	430	654	84	914	cpuinfo	interrupts	kpagecount	net	swaps	vmallocinfo
12	17	21	30	4	574	683	85	92	crypto	iomem	kpageflags	pagetypeinfo	sys	vmstat
13	1749	22	31	402	6	7	86	925	device	ioports	loadavg	partitions	sysrq-trigger	xen
14	1767	23	32	404	641	728	861	926	diskstats	irq	locks	sched_debug	sysvipc	zoneinfo

PID를 이름으로 갖는 디렉토리에
해당 프로세스 정보를 제공

/proc/cpuinfo
: 시스템의 CPU 정보 제공

```
root@linux-test:~# cat /proc/cpuinfo
```

```
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 85
model name    : Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz
stepping      : 4
microcode     : 0x2006e05
cpu MHz       : 2194.913
cache size    : 19712 KB
physical id   : 0
siblings      : 1
core id       : 0
cpu cores     : 1
apicid        : 0
initial apicid : 0
fpu           : yes
fpu_exception : yes
cpuid level   : 13
wp            : yes
```


의사 파일 시스템 - proc



/proc 디렉토리의 주요 파일에 대해 알려줘



`/proc` 디렉토리에는 다양한 파일이 존재하며, 각 파일은 시스템 정보에 대한 특정 부분을 제공합니다. 여기에는 일반적으로 `/proc` 디렉토리에서 자주 사용되는 주요 파일들이 포함됩니다.

cat 명령으로 출력되는 내용 확인 권장

1. `/proc/cpuinfo` : 현재 시스템에서 사용 중인 CPU의 정보를 제공합니다. CPU 모델, 속도, 코어 수 등과 같은 정보를 포함합니다.
2. `/proc/meminfo` : 메모리 사용량과 관련된 정보를 제공합니다. 전체 메모리 크기, 사용 가능한 메모리, 사용 중인 메모리 등의 정보를 포함합니다.
3. `/proc/loadavg` : 시스템의 로드 평균 정보를 제공합니다. 로드 평균은 시스템이 처리 중인 작업의 수를 나타내며, 시스템의 부하를 측정하는 데 사용됩니다.
4. `/proc/net/dev` : 네트워크 인터페이스에 대한 정보를 제공합니다. 각 인터페이스의 전송량, 수신량, 오류 수 등의 정보를 포함합니다.
5. `/proc/sys/kernel/hostname` : 시스템의 호스트 이름을 제공합니다.
6. `/proc/sys/kernel/osrelease` : 시스템의 커널 버전을 제공합니다.
7. `/proc/sys/kernel/panic` : 시스템 패닉을 발생시키는 데 필요한 임계값을 제공합니다.
8. `/proc/sys/fs/file-nr` : 현재 열린 파일 디스크립터의 수를 제공합니다.
9. `/proc/sys/vm/swappiness` : 가상 메모리의 사용 방식에 대한 설정 값을 제공합니다.

이 외에도 `/proc` 디렉토리에는 다양한 파일이 존재하며, 시스템의 다른 부분과 관련된 정보를 제공합니다. 파일 이름은 해당 정보를 식별하는 데 사용되므로, 필요한 정보를 찾아서 접근할 수 있습니다.

네이버 클라우드 추가 디스크 제거

반드시 아래의 순서에 따라 수행!!!

순서가 바뀌거나 빠지면 서버 부팅이 되지 않을 수 있으니 주의!!

```
root@linux-test:~# nano /etc/fstab
```

1) nano 편집기로 /etc/fstab 오픈

```

GNU nano 2.9.3 /etc/fstab

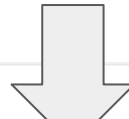
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/xvda1 during installation
UUID=8d992262-a389-4b90-b69a-8c010504a44a / ext4 errors=remount-ro 0 1
/dev/xvdb1 /mnt/disk1 ext4 defaults 1 2
/dev/xvdb2 /mnt/disk2 vfat defaults 0 0
  
```

2) /dev/xvdb1, /dev/xvdb2 설정
내용 삭제 후 저장

네이버 클라우드 추가 디스크 제거

3) **shutdown -f now** 명령으로
서버 Power-Off

```
root@linux-test:~# shutdown -f now
Connection to 106.10.56.244 closed by remote host.
Connection to 106.10.56.244 closed.
```



<input type="checkbox"/> 서버 이름	서버 이미지 이름	서버 구성	상태	비공인 IP	공인 IP	ZONE	모니터링	터링	Network 모니
<input type="checkbox"/> linux-test	ubuntu-18.04	[Compact] 1vCPU...	● 정지	10.41.185.135		KR-2	기본	해제	터링

4) 서버 상태 정지 확인

Server

Server

Bare Metal Server

Server Image

Server Image Builder

Storage

Snapshot

Public IP

스토리지 삭제

스냅샷 생성

서버에 연결

서버에 연결 해제

크기 변경

스토리지 이름

종류

크기

IOPS

생성 일시

상태

ZONE

연결정보

☐ 스토리지 이름

☒ hdd1

☐ linux-t

HDD

20 GB

2023-04-12 오후 3:40
(UTC+09:00)

● 사용중

KR-2

linux-test:/dev/xvdb

linux-t


2023-04-12 오후 3:22

● 사용중

KR-2

linux-test:/dev/xvda

네이버 클라우드 추가 디스크 제거



Server

Server

Bare Metal Server

Server Image

Server Image Builder

Storage

Snapshot

Public IP

시작 정지 재시작 반납 강제 정지 서버 접속 콘솔 모니터링 포트 포워딩 설정

서버 관리 및 설정

6) 서버 선택 후 [시작] 클릭

<input checked="" type="checkbox"/>	서버 이름	서버 이미지 이름	서버 구성	상태	비공인 IP
<input checked="" type="checkbox"/>	linux-test	ubuntu-18.04	[Compact] 1vCPU, 2GB ...	정지	10.41.185.135

```
Last login: Wed Apr 12 16:38:19 2023 from 223.131.46.118
root@linux-test:~# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda        202:0    0   50G  0 disk
└─xvda1     202:1    0   50G  0 part /
root@linux-test:~#
```

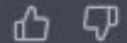
7) ssh로 서버에 접속 후 lsblk 실행
=> xvdb 항목 삭제 확인



GB와 GiB의 차이는?



GB와 GiB는 모두 데이터의 용량을 측정하는 단위입니다. 그러나 GB는 10의 거듭제곱으로 1,000,000,000 바이트를 나타내고, GiB는 2의 거듭제곱으로 1,073,741,824 바이트를 나타냅니다.



이러한 차이로 인해, 1GB는 약 0.93 GiB에 해당하며, 1GiB는 약 1.07 GB에 해당합니다. 따라서, 같은 데이터 용량이라면 GB와 GiB로 표기했을 때의 수치는 다를 수 있습니다.

기술적으로는, GB는 SI 단위계에서 사용되는 접두사 "기가"에 해당하고, GiB는 이진수 단위계에서 사용되는 접두사 "기비"에 해당합니다. 하지만 실제로는 두 용어가 혼용되어 사용되기도 합니다.

Thank You