

Linux & Git, Github

2020.01.09

학부생 인턴 이지현

리눅스란?

- 리눅스 토발즈에 의해 개발된 **유닉스(UNIX) 운영체제**를 기반으로 만들어진 컴퓨터 운영체제
- **다중 사용자, 다중 작업(Multitasking), 다중 스레드를 지원**하는 네트워크 운영체제(NOS)
- 현재 **상당수의 웹 서버와 모바일 장비(안드로이드 등)**를 구동하는 운영체제로도 많이 사용중

리눅스를 사용하는 이유

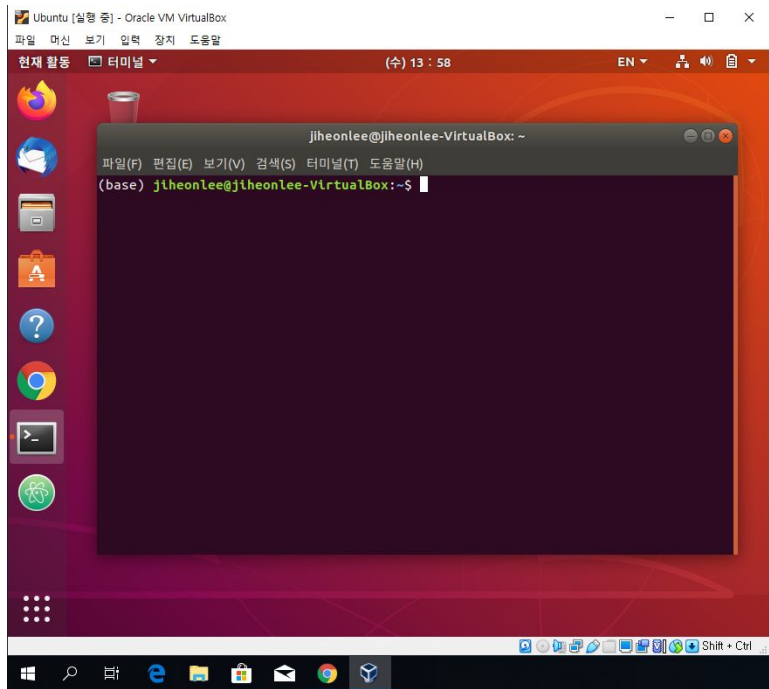
- 모든 소스가 오픈되어 있어 광범위하게 사용
- 다중 사용자, 다중 작업(**Multitasking**)을 지원으로
서버를 운영하기에 적합
- 리눅스는 무료이며 개발 환경이 풍부

리눅스 단점

- 윈도우와 다르게 어렵고, 적응하는데 오래 걸림으로 **사용자의 숙련된 기술 요구**
- 리눅스에서 사용되는 대부분의 응용 프로그램들이 **비상업적인 제품으로 기술 지원이 부족**
- 윈도우 사용자를 대상으로 만든 프로그램들은 **리눅스 환경에서 구동되지 않을 수 있음**

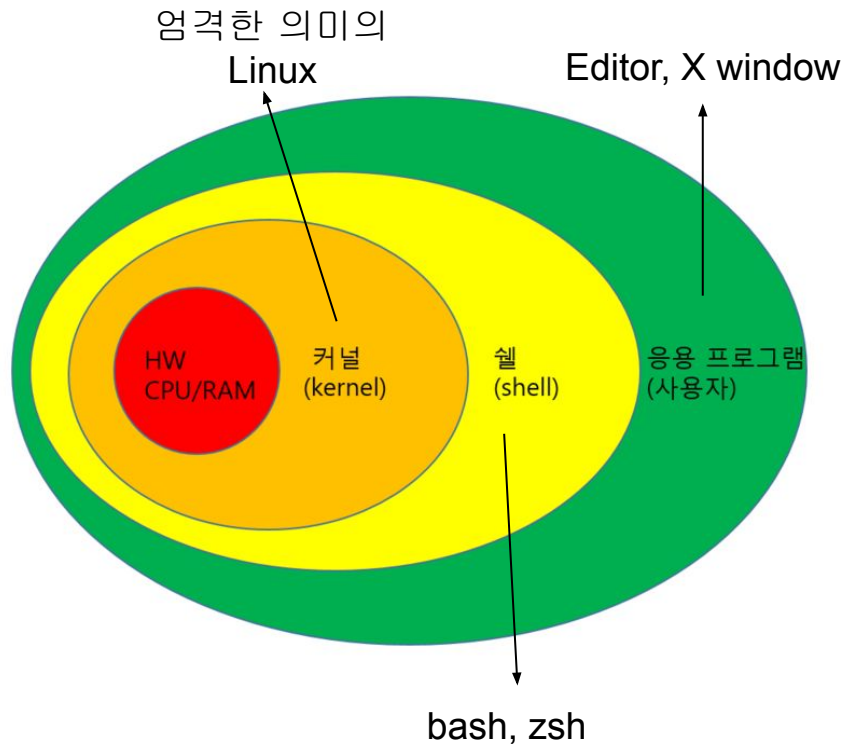
가상머신 (Virtual Box)

- 소프트웨어 기술을 이용하여 **가상의 하드웨어를 만들고** 그 위에 운영체제를 설치
- 윈도우가 설치된 컴퓨터에 또 다른 **윈도우나 리눅스, Mac OS를 동시에 실행**시켜서 마치 여러 대의 컴퓨터를 사용하는 것과 같은 효과

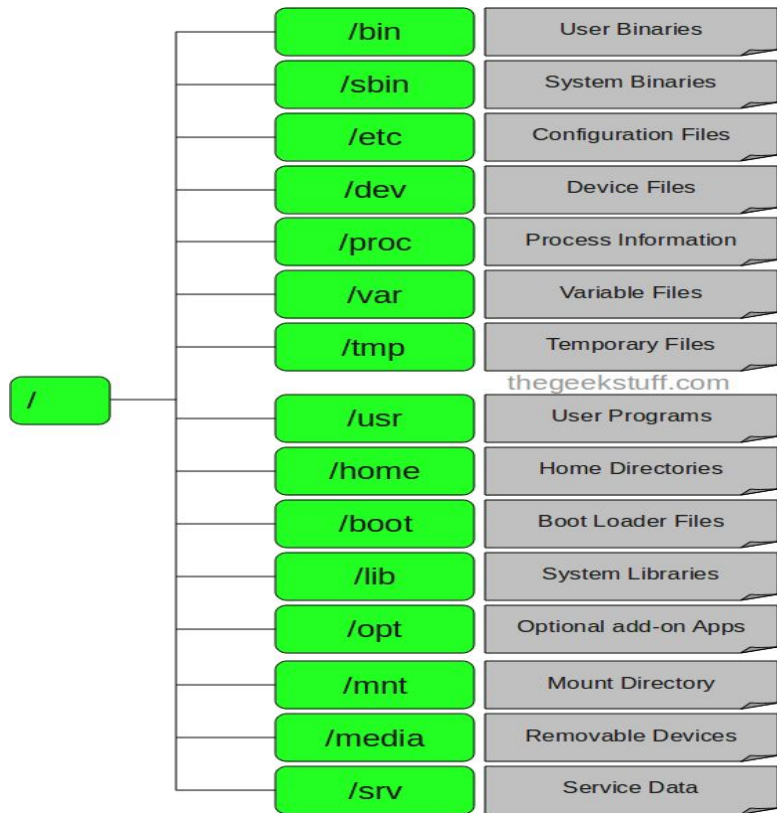


리눅스의 구조

- **커널(Kernel)** : 운영체제의 핵심,
운영체제의 수행에 필요한 여러
서비스를 제공하는 역할
- **셸(Shell)** : 명령어 해석기,
사용자가 입력한 명령어를
커널에게 전달하는 역할



디렉토리의 구조

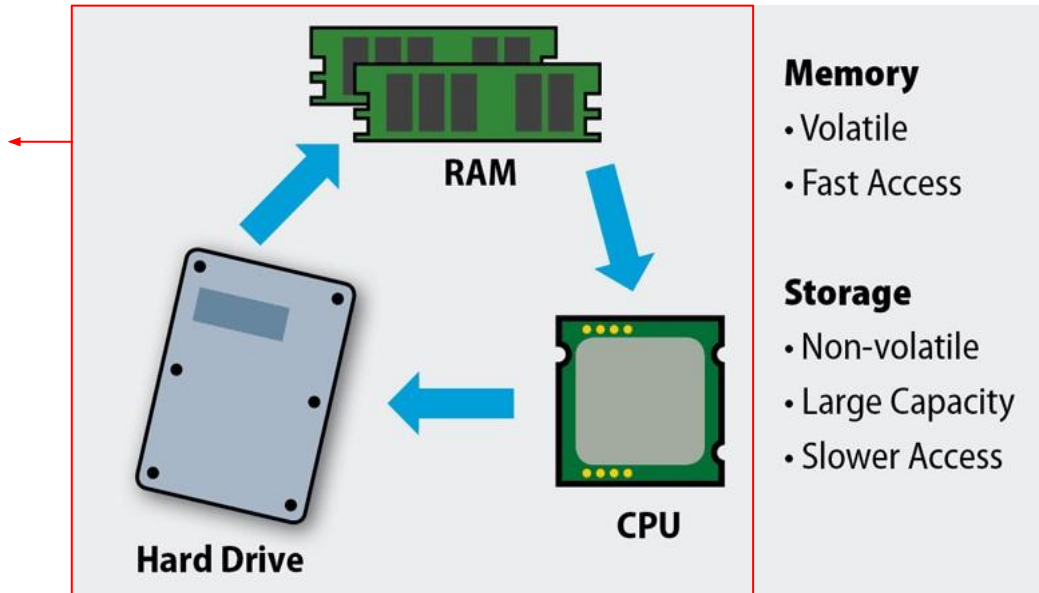


계층적 구조(=트리 구조)

위치	디렉토리명	설명
/	Root	최상위(루트) 디렉토리 절대 경로의 시작
/bin	User Binaries	기본 명령어들이 모여 있는 디렉토리 ex) mv, cp, rm
/etc	Configuration files	설정 파일을 두는 디렉토리
/var	Variable Files	시스템에서 사용되는 가변적인 파일들이 저장
/tmp	Temporary Files	임시 파일들을 위한 디렉토리
/usr	User Programs	일반 사용자들이 사용하는 디렉토리
/home	Home Directories	시스템 계정 사용자들의 홈 디렉토리

프로세스

- 하드디스크에 저장된 프로그램이 메모리에 적재되어 활성화 된 것
- 프로그램이 화면에 나타나지 않고 실행되고 있는 것을 백그라운드 프로세스라 함
- 리눅스의 프로세스 모니터링으로는 **ps, top, htop**



컴퓨터의 구조

데몬 (daemon)

- 백그라운드로 작동하는

서버 관련 프로세스

- 웹 서버 프로세스(**apache2**), 네임

서버 프로세스, DB 서버 프로세스 등

- daemon의 목적을 가지고 있는

프로그램들은 **/etc/init.d** 위치에 존재

```
jiheonlee@jiheonlee-VirtualBox:~$ cd /etc/init.d/
jiheonlee@jiheonlee-VirtualBox:/etc/init.d$ ls
acpid                cron                 keyboard-setup.sh   saned
alsa-utils           cups                kmod                 speech-dispatcher
anacron              cups-browsed        network-manager      spice-vdagent
apache-htcacheclean  dbus               networking           udev
apache2             dns-clean           plymouth             ufw
apparmor             gdm3               plymouth-log         unattended-upgrades
apport              grub-common         pppd-dns             uuid
avahi-daemon         hwclock.sh          procps               whoopsie
bluetooth           irqbalance          rsync                x11-common
console-setup.sh     kerneloops          rsyslog
jiheonlee@jiheonlee-VirtualBox:/etc/init.d$
```

◦ Daemon

데몬 실행 및 종료

```
$ sudo service {데몬이름} start # 데몬 실행
$ sudo service {데몬이름} stop # 데몬 중지
$ sudo service {데몬이름} restart # 데몬 재시작
```

권한 (Permission)

◦ CHMOD

- 파일의 권한을 변경
- 8진수 형태와 심볼릭 형태로 사용 가능
- 심볼릭이 기능적인 면으로 좋지만 조금 복잡
- 쉽게 쓸려면 8진수 형태

```
$ chmod {옵션} {8진수Permission} {파일명} - 8진수 형태
$ chmod {옵션} {대상}{+/-/=}{rwx} {파일명} - 심볼릭 형태
-R : 하위 디렉토리에 모든 권한 변경
-c : 권한 변경 파일내용을 출력
```

OwnerGroup Other

drwxr-xr-x	3	jiheonlee	jiheonlee	4096	1월	6 00:55	git_test
drwxr-xr-x	3	jiheonlee	jiheonlee	4096	1월	6 01:33	git_test1
-rw-r--r--	1	jiheonlee	jiheonlee	17	1월	7 15:51	test2.txt

Type Access Link Owner Group Capacity date Directory/File
Mode

8진수 0~7은 아래와 같이 2진수로 표현이 가능

0 : 000
1 : 001
2 : 010
3 : 011
4 : 100
5 : 101
6 : 110
7 : 111

위 2진수 세자리는 rwx 세자리와 일치하며 2진수가 1이면 해당 권한을 부여, 0이면 해당 권한을 제거

chmod 명령어 심볼릭

- 대상

u : user의 권한
g : group의 권한
o : other의 권한
a : 모든 사용자 권한

- +/-/=

+ : 해당 권한을 추가
- : 해당 권한을 제거
= : 해당 권한을 설정한데로 변경

- rwx

r : 읽기 권한
w : 쓰기 권한
x : 실행 권한

권한 (Permission)

○ CHOWN

chown(change the owner of a file) 파일의 소유권을 바꾸기 위해 사용

```
$ chown {옵션} {변경할유저이름:변경할그룹이름} {파일이름}
```

-R : 하위 디렉토리에도 모든 권한 변경

- chown {변경할유저이름} - 소유자만 변경
- chown {:변경할그룹이름} - 그룹만 변경
- chown {변경할유저이름:} - 소유자와 그룹 모두 동일한걸로 변경
- chown {변경할유저이름:변경할그룹이름} - 소유자와 그룹을 서로 다른걸로 변경

OwnerGroup Other

drwxr-xr-x	3	jiheonlee	jiheonlee	4096	1월	6 00:55	git_test
drwxr-xr-x	3	jiheonlee	jiheonlee	4096	1월	6 01:33	git_test1
-rw-r--r--	1	jiheonlee	jiheonlee	17	1월	7 15:51	test2.txt

↓
Access
Mode

↓
Owner

↓
Group

원격제어 (SSH:Secure Shell Protocol)

- 원격지에 있는 컴퓨터를 안전하게

제어하기 위한 프로토콜

- **SSH 클라이언트**와 **SSH서버**의

관계로 상호작용

- Unix 계열의 운영체제를 원격에서

제어하기 위한 방법



SSH Key?

- 서버에 접속 할 때 비밀번호 대신 **Key**를 제출하는 방식
- 공개키(Public Key)와 비공개키(Private Key)로 이루어짐
- 비밀번호 보다 높은 수준의 보안을 필요로 할 때 사용

SSH-Keygen

rsa라는 암호화 방식으로 키를 생성

```
$ ssh-keygen -t rsa
```

키를 확인

```
$ ls -al ~/.ssh/
```

파일	설명
id_rsa	private key, 절대로 타인에게 노출되면 안된다.
id_rsa.pub	public key, 접속하려는 리모트 머신의 authorized_keys에 입력한다.
authorized_keys	리모트 머신의 .ssh 디렉토리 아래에 위치하면서 id_rsa.pub 키의 값을 저장한다.



SSH Client

Private Key



SSH Server

Public Key

Git이란?

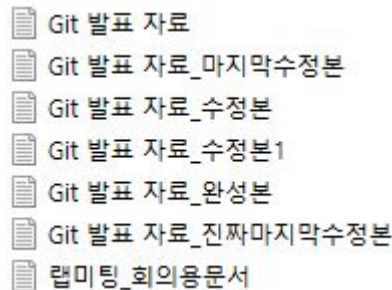
- 컴퓨터 파일의 변경사항을 추적하고 여러 사용자들 간에 작업을 조율하기 위한

분산 버전 관리 시스템

- 소스코드를 효율적으로 관리 할 수 있게 해주는 **형상 관리 도구**라고도 함
- 파일의 **이력을 관리**
- 리눅스 토발즈에 의해 개발되었으며, **무료이며 공개소프트웨어**

Git을 사용하는 이유

- 같은 파일을 여러 명이 동시에 작업하는 병렬 개발이 가능
- 협업이 필요할 때, 여러명과 함께 코드를 공유
- 버전 관리를 통해 체계적인 개발 가능
- 개발 환경에서 실수를 할 수 있기 때문 (백업, 복구)



Git 발표 자료
Git 발표 자료_마지막수정본
Git 발표 자료_수정본
Git 발표 자료_수정본1
Git 발표 자료_완성본
Git 발표 자료_진짜마지막수정본
랩미팅_회의용문서

ex) 특별한 규칙 없이 마음대로 이름을 붙여놓는 경우

Git repository

- 파일이나 폴더를 저장해 두는 곳
- 내용 일부 문구가 서로 다르면 다른

파일로 인식하여 **변경 사항 별로**

구분하여 저장 가능

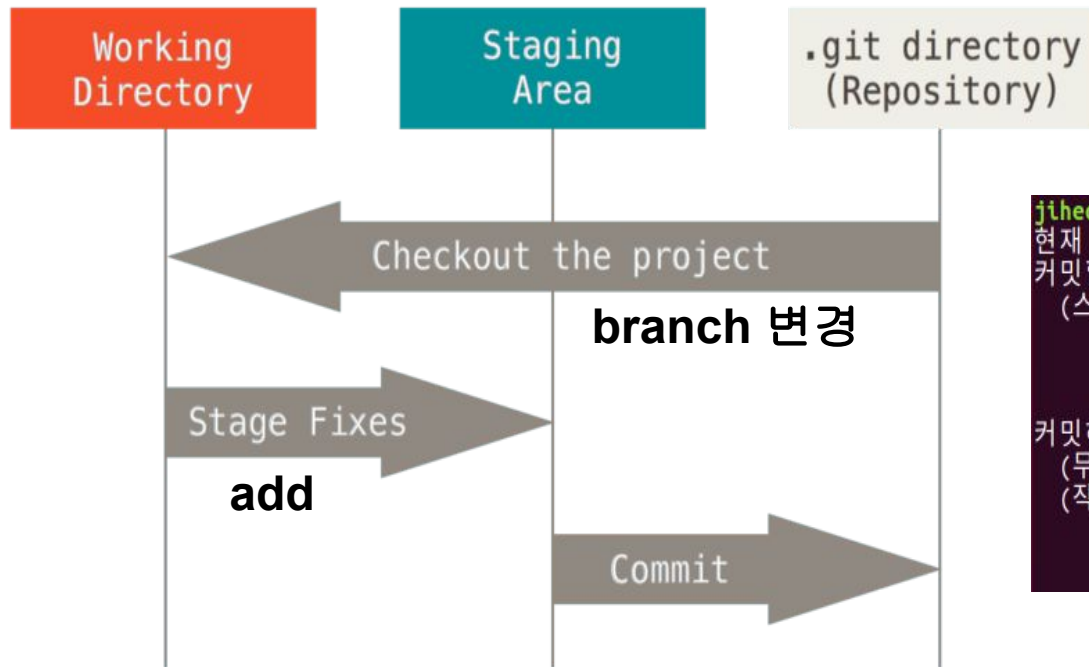
- 로컬 저장소(Local Repository)와
원격저장소(Remote Repository)로 나뉨

현재 디렉토리에 git 저장소를 생성

```
$ git init
```



Git의 구조



```
jiheonlee@jiheonlee-VirtualBox:~/workspace/git_test$ git status
현재 브랜치 master
커밋할 변경 사항:
(스테이지 해제하려면 "git reset HEAD <파일>..."을 사용하십시오)

수정함:      f1.txt

커밋하도록 정하지 않은 변경 사항:
(무엇을 커밋할지 바꾸려면 "git add <파일>..."을 사용하십시오)
(작업 폴더의 변경 사항을 버리려면 "git checkout -- <파일>..."을

수정함:      f2.txt
```

<두 파일 수정했지만 f1.txt
파일만 add한 상황>

Git add와 commit

파일은 수정했지만 아직 stage area에 올라가지 않은 파일들을 staging area에 올림

```
$ git add {파일이름}
```

staging area에 올라가 있는 파일들을 commit

```
$ git commit {옵션} {파일이름}
-a : 기존에 add를 하지 않아도 바로 staging area에 올려서 commit 함
-m : editor없이 commit의 메시지를 입력 # $ git commit -m test.txt "Version1"
```

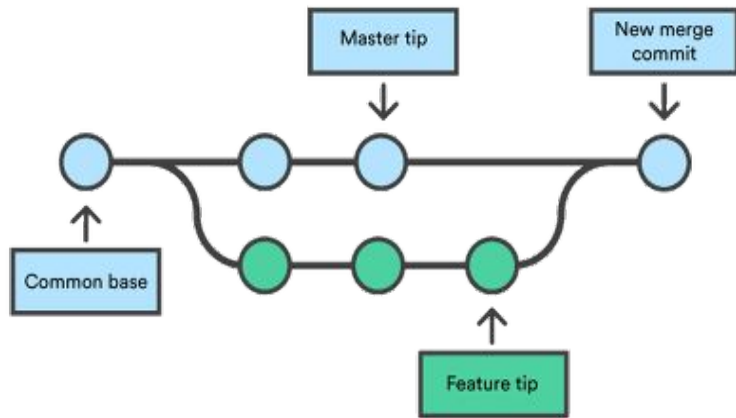
Git branch와 merge

- **branch**

어떠한 이슈가 발생하였을 때 그
이슈를 처리할 **새로운 작업공간**

- **merge**

두 개의 **branch**를 **병합**하는 것



브랜치 생성

```
$ git branch {branch명}  
- master가 기본 branch
```

다른 브랜치로 변경

```
$ git checkout
```

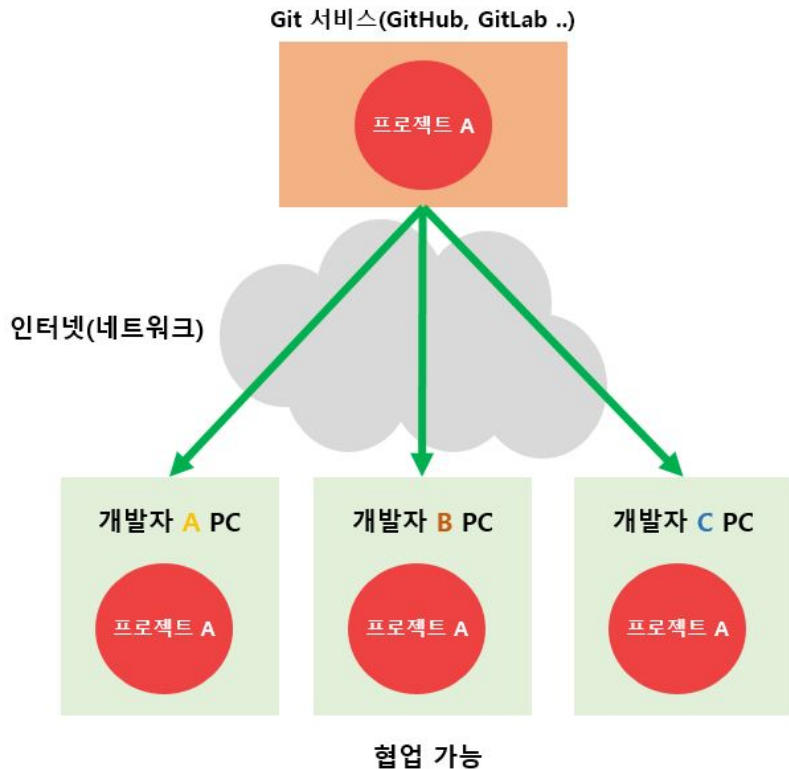
- **Merge**

현재 브랜치에서 입력한 브랜치와 merge(합치다)

```
$ git merger {branch명}
```

원격저장소 : remote repository

- 내 로컬PC 저장소가 아닌
네트워크상의 다른 위치에
존재하는 **Git** 저장소
- 백업과 여러 사람이 협업을 하기
위해 필요
- **Github**은 인터넷을 통해 원격
저장소 공간을 제공해주는 서비스



원격저장소 : remote, push

현재 저장소에 원격저장소를 add(연결)한다. 경로 앞에 origin이라 하면 경로는 origin을 가르킴, 원격저장소의 기본 이름은 origin

```
$ git remote add origin {경로}
```

추가한 원격저장소의 목록을 확인

```
$ git remote  
-v : 자세하게 보여줌
```

원격저장소를 제거

```
$ git remote remove origin
```

현재 브랜치를 연결시킨 지역저장소에서 원격저장소에 푸쉬, 즉 업로드

```
$ git push origin
```

원격저장소 : pull, clone

- **pull**

local repository와 비교하여 병합을 하고,
저장(add)까지 수행

- **clone**

Github의 모든 파일들을 가져오기만 함

원격저장소의 파일들을 지역저장소로 가져옴

```
$ git pull
```

원격저장소의 파일들을 디렉토리를 생성하여 안에 다운로드

```
$ git clone {원격저장소 주소} {디렉토리명}
```

감사합니다