<RO:BIT 리눅스 & git 보고서>

19기 지능형로봇팀 예비단원 최범수

1. Linux(리눅스)란

Linux는 1991년 Linus Torvals가 개발한 운영체제이다. Linux는 Unix 운영체제를 기반으로 만들어진 운영체제로 유닉스 클론 운영체제라 할 수 있다. Unix와 마찬가지로 다중 사용자, 다중작업 다중 스레드를 지원하는 네트워크 운영체제를 의미한다.

또한 Unix가 애초부터 통신 네트워크를 지향해 설계된 것처럼 Linux 역시 서버로 작동하는데 최적화 되어있다. 또한 Linux는 자유 소프트 라이센스로 누구나 소스코드를 활용, 수정 및 재 배포가 가능해서 지속적인 업그레이드가 이루어진다.

2. Linux의 구조

리눅스는 크게 커널, 셸, 디렉토리로 3가지로 구성되어 있다.

-커널(kernel)

커널은 운영체제의 핵심으로 메모리관리, 프로세스 관리, 장치 관리 등 컴퓨터의 모든 자원을 초기화하고 제어하는 기능을 수행한다.

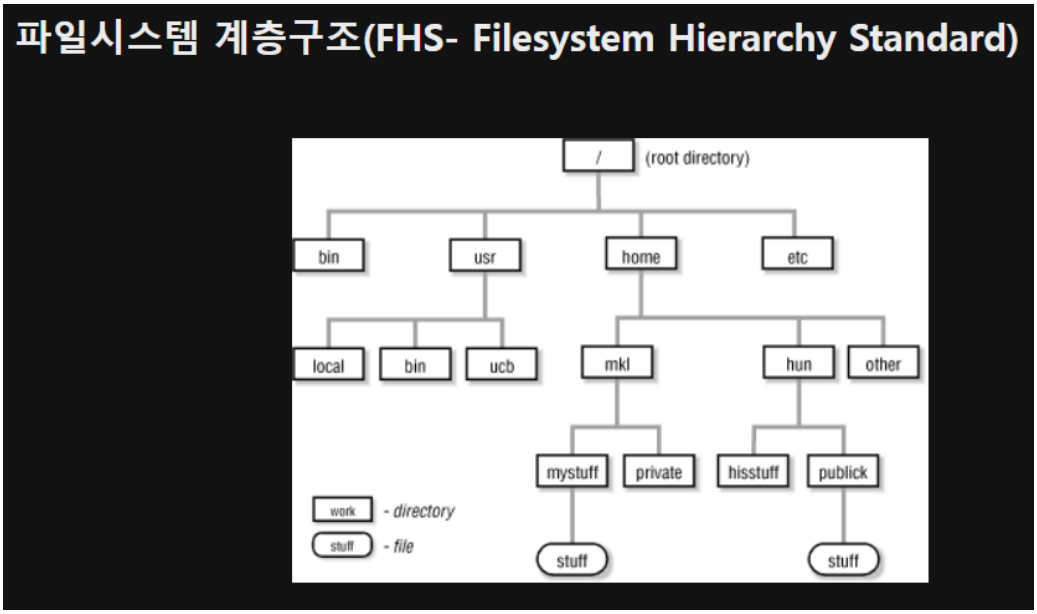
-셸(shell)

사용자가 입력한 문장을 읽어 요청을 실행하며 커널이 명령어를 해석해 결과를 수행한 후 결과를 다른 프로그램이나 커널로 전송한다.

즉 사용자와 커널의 중간다리 역할을 수행한다. 리눅스는 bash셸을 기본으로 사용한다.

-디렉토리

파일을 가지고 있거나 또 다른 디렉터리를 가지고 있는 그룹을 의미하며 파일 시스템에 의해 관리되고 있다.



디렉토리 구조는 전반적으로 Tree 구조, 명령어의 성격과 내용 및 사용권한에 따라 디렉토리로 구분

디렉토리 경로

-Absolute Path

- 이름 그대로 절대적인 경로, 완전한 경로를 의미
- Root 디렉토리부터 시작하는 경로를 뜻한다.
- 현재 나의 위치와 상관없이 항상 정확한 경로 전달

-Relative Path

- 이름 그대로 상대적인 경로를 의미
- 현재 내 위치를 기반으로 움직인다
- '.'과 '..' 이 두가지 심볼이 중요. '.'는 현재 디렉토리 '..'는 상위 디렉토리를 의미한다.

3. Linux 명령어 정리(102)

(1) ls (List)

ls : 현재 디렉토리의 모든 파일 및 폴더를 기본 형식으로 보여준다.

ls -l : 파일 및 폴더에 대한 자세한 정보와 함께 리스트업

ls -a : 숨겨진 파일을 포함해 모든 파일을 보여준다.

(2) cd (Change Directory)

cd : 디렉토리를 변경

cd Documents : 현재 디렉토리에서 'Documents'라는 이름의 폴더로 이동

cd .. : 현재 디렉토리의 상위 폴더로 이동(파일 탐색기에서 전 폴더)

(3) pwd (Print Working Directory)

pwd : 현재 작업 중인 디렉토리의 경로를 표시

(4) mkdir (Make Directory)

mkdir : 새로운 디렉토리(폴더) 생성

mkdir new_folder : 현재 디렉토리에 'new_folder'라는 이름의 새 디렉토리를 만든다.

mkdir -p folder1/folder2 : 'folder1'내에 'folder2'를 생성한다. (-p는 상위 디렉토리가 없다면 그 상위 디렉토리를 생성)

(5) rmdir (Remove Directory)

rmdir : 디렉토리 삭제

rmdir old_folder : old_folder라는 이름의 디렉토리 삭제

rmdir은 디렉토리가 비어있을 때만 작동. 내부에 파일이나 다른 디렉토리가 있으면 오류 발생

디렉토리 안의 파일과 함께 삭제하려면 rm -r 명령어 사용

(6) rm (Remove)

rm : 파일이나 디렉토리를 삭제

rm file.txt : 'file.txt'라는 파일을 삭제

rm -r folder : 'folder'라는 디렉토리와 그 안의 모든 내용을 삭제

삭제된 파일은 복구가 어려우므로 확인 항상 필요

(7) touch

touch : 새로운 빈 파일을 생성하거나, 기존 파일의 타임스탬프(날짜 및 시간 정보)를 현재 시간으로 갱신

touch new_file.txt : new_file.txt'라는 새 파일을 생성한다. 파일이 이미 존재한다면, 타임스탬프가 갱신.

간단하고 빠르게 파일 생성 가능, 스크립트나 로그 파일을 초기화할 때 유용하다.

(8) cp (Copy)

cp : 파일이나 디렉토리를 복사

cp source.txt destination.txt : 'source.txt'를 'destination.txt'로 복사한다.

'destination.txt'가 이미 존재하면 덮어쓴다.

cp -r source_dir destination_dir : 'source_dir'디렉토리와 그 내용을 'destination_dir'로 복사한다.

(9) mv (Move)

mv : 파일이나 디렉토리의 위치를 이동시키거나 이름을 변경한다.

mv old_name.txt new_name.txt : 'old_name.txt'의 이름을 'new_name.txt'로 변경한다.

mv file.txt /path/to/directory/ : 'file.txt'를 지정된 디렉토리로 이동한다.
mv 명령어는 파일을 이동시킬 때 복사 후 삭제하는 것이 아니라 파일의 위치정보만 변경하기 때문에 처리 속도가 빠르다.
이동하려는 대상 경로에 같은 이름의 파일이 이미 존재할 경우, 기존 파일은 덮어쓰여진다.

(10) cat (Concatenate)

cat : 텍스트 파일의 내용을 화면에 출력하거나, 여러 파일의 내용을 연결하여 출력한다.
cat file.txt : 'file.txt' 파일의 내용을 화면에 표시한다.
cat file1.txt file2.txt > combined.txt : 'file1.txt'와 'file2.txt'의 내용을 합쳐 'combined.txt'에 저장한다.

(11) chmod (Change Mode)

chmod : 파일이나 디렉토리의 권한을 변경
chmod 755 file.sh : 'file.sh'파일에 대해 소유자에게는 읽기, 쓰기, 실행 권한을 부여하고, 그룹과 기타 사용자에게는 읽기와 실행 권한만 부여한다.
chmod u+x file.sh : 'file.sh' 파일에 대해 현재 사용자에게 실행 권한을 추가한다.
권한 변경은 보안에 영향을 줄 수 있으므로 신중히 사용해야 한다.

(12) grep (Global Regular Expression Print)

grep : 파일 내용 중에서 지정된 패턴이나 문자열을 검색하여 그 결과를 출력한다.
grep "text" file.txt : 'file.txt'에서 "text"라는 문자열이 포함된 모든 줄을 표시한다.
grep -r "text" . : 현재 디렉토리와 하위 디렉토리에서 "text" 문자열을 재귀적으로 검색
정규 표현식을 사용하여 복잡한 검색 패턴을 지정할 수 있으며, 로그 파일 분석이나 특정 데이터 추출에 유용

(13) echo

주어진 문자열을 터미널에 출력. 환경 변수의 값을 표시하거나, 파일에 텍스트를 쓰는 데에도 사용된다.
echo "Hello World" : 터미널에 "Hello World"라는 문구를 출력한다.
echo $HOME : 'HOME' 환경 변수의 값을 출력한다.
echo "some text" > file.txt : "Some text"라는 문구를 'file.txt'파일에 저장한다.
스크립트 작성 시 변수의 값을 확인하거나, 파일에 내용을 빠르게 추가할 때 유용.

(14) man (Manual)

리눅스 명령어의 사용법, 옵션, 기능 등을 설명하는 매뉴얼 페이지를 제공한다.

man ls : 'ls 명령어에 대한 매뉴얼 페이지를 보여준다.

(15) sudo (SuperUser DO)

일반 사용자가 관리자 권한을 가지고 명령어를 실행할 수 있게 한다. 시스템 설정 변경, 중요한 파일 수정, 관리자 권한을 필요로 하는 소프트웨어 설치 시 사용된다.

sudo apt-get update : 패키지 리스트를 업데이트.

(16) find

파일이나 디렉토리를 검색한다.

find . -name "file.txt" : 현재 디렉토리에서 'file.txt'파일을 찾는다.

find / -type d -name "config" : 루트 디렉토리에서 'config'라는 이름의 디렉토리를 찾는다.

(17) less

less : 파일의 내용을 페이지 단위로 볼 수 있게 해준다.

(18) chown

chown : 파일이나 디렉토리의 소유권을 변경한다.

(19) du

du : 디렉토리의 디스크 사용량을 확인한다

(20) **df**

df : 파일시스템의 디스크 공간 사용량을 확인합니다.

예시: df -h

(21) **top**

top : 현재 실행 중인 프로세스의 정보를 실시간으로 보여줍니다.

예시: top

(22) **ps**

ps : 현재 실행 중인 프로세스를 출력합니다.

예시: ps -aux

(23) **kill**

kill : 프로세스를 종료합니다.

예시: kill -9 12345

(24) **tar**

tar : 파일을 압축하거나 압축을 해제합니다.

예시: tar -xvf archive.tar

(25) **gzip**

gzip : 파일을 압축합니다.

예시: gzip file.txt

(26) **gunzip**

gunzip : gzip으로 압축된 파일을 해제합니다.

예시: gunzip file.txt.gz

(27) **zip**

zip : 파일이나 디렉토리를 zip 형식으로 압축합니다.

예시: zip -r archive.zip folder/

(28) **unzip**

unzip : zip 파일을 해제합니다.

예시: unzip archive.zip

(29) **ssh**

ssh : SSH 프로토콜을 이용해 원격 호스트에 접속합니다.

예시: ssh

(30) **scp**

scp : 원격 호스트와 파일을 안전하게 복사합니다.

예시: scp file.txt :/path/

(31) **wget**

wget : 네트워크를 통해 파일을 다운로드합니다.

예시: wget http://example.com/file.txt

(32) **curl**

curl : 네트워크를 통해 데이터를 전송합니다.

예시: curl -O http://example.com/file.txt

(33) **apt-get**

apt-get : 패키지 관리자를 이용해 소프트웨어를 설치하거나 관리합니다.

예시: apt-get install nginx

(34) **yum**

yum : Red Hat 기반 시스템에서 소프트웨어 패키지를 관리합니다.

예시: yum install nginx

(35) **systemctl**

systemctl : systemd 시스템과 서비스 매니저를 관리합니다.

예시: systemctl start sshd

(36) **journalctl**

journalctl : systemd 로그를 확인합니다.

예시: journalctl -u nginx

(37) **crontab**

crontab : 예약된 작업(크론 작업)을 관리합니다.

예시: crontab -e

(38) **nano**

nano : 텍스트 에디터를 사용하여 파일을 편집합니다.

예시: nano file.txt

(39) **vi / vim**

vi / vim : 강력한 텍스트 에디터를 사용하여 파일을 편집합니다.

예시: vim file.txt

(40) **tail**

tail : 파일의 끝 부분을 출력합니다. 주로 로그 파일을 모니터링할 때 사용합니다.

예시: tail -f /var/log/syslog

(41) **head**

head : 파일의 시작 부분을 출력합니다.

예시: head file.txt

(42) **diff**

diff : 두 파일의 차이점을 비교합니다.

예시: diff file1.txt file2.txt

(43) **chmod**

chmod : 파일이나 디렉토리의 권한을 변경합니다.

예시: chmod +x script.sh

(44) **chgrp**

chgrp : 파일이나 디렉토리의 그룹 소유권을 변경합니다.

예시: chgrp newgroup file.txt

(45) **ln**

ln : 심볼릭 링크나 하드 링크를 생성합니다.

예시: ln -s source.txt link.txt

(46) **who**

who : 현재 시스템에 로그인한 사용자를 보여줍니다.

예시: who

(47) **w**

w : 현재 로그인한 사용자와 그들이 무엇을 하고 있는지 보여줍니다.

예시: w

(48) **history**

history : 사용자의 명령어 히스토리를 출력합니다.

예시: history

(49) **alias**

alias : 명령어에 별칭을 만듭니다.
예시: alias ll='ls -l'
(50) **unalias**
unalias : 별칭을 제거합니다.
예시: unalias ll
(51) **mount**
mount : 파일 시스템을 마운트합니다.
예시: mount /dev/sdb1 /mnt/usb
(52) **umount**
umount : 마운트된 파일 시스템을 언마운트합니다.
예시: umount /mnt/usb
(53) **fsck**
fsck : 파일 시스템의 무결성을 검사하고 수리합니다.
예시: fsck /dev/sda1
(54) **dd**
dd : 파일이나 장치 간에 낮은 단계의 데이터 복사를 수행합니다.
예시: dd if=/dev/zero of=/dev/sda1
(55) **fdisk**
fdisk : 디스크 파티션을 조작합니다.
예시: fdisk /dev/sda
(56) **parted**
parted : 파티션 수정에 사용합니다.
예시: parted -l
(57) **lsof**
lsof : 열려 있는 파일에 대한 정보를 출력합니다.
예시: lsof /var/log/syslog
(58) **netstat**
netstat : 네트워크 통계를 보여줍니다.
예시: netstat -tulnp
(59) **ss**
ss : 소켓 통계를 보여줍니다. netstat 대신 사용됩니다.
예시: ss -tuln
(60) **iptables**
iptables : 시스템의 방화벽 규칙을 설정합니다.
예시: iptables -L

(61) **chroot**

chroot : 루트 디렉토리를 변경합니다.

예시: chroot /mnt/newroot

(62) **useradd**

useradd : 새로운 사용자를 시스템에 추가합니다.

예시: useradd newuser

(63) **usermod**

usermod : 사용자 계정을 수정합니다.

예시: usermod -aG sudo newuser

(64) **userdel**

userdel : 사용자 계정을 삭제합니다.

예시: userdel olduser

(65) **groupadd**

groupadd : 새로운 그룹을 추가합니다.

예시: groupadd newgroup

(66) **groupdel**

groupdel : 그룹을 삭제합니다.

예시: groupdel oldgroup

(67) **passwd**

passwd : 사용자의 비밀번호를 변경합니다.

예시: passwd username

(68) **uptime**

uptime : 시스템이 얼마나 오랫동안 실행되고 있는지 보여줍니다.

예시: uptime

(69) **free**

free : 메모리의 사용량을 보여줍니다.

예시: free -h

(70) **watch**

watch : 주기적으로 프로그램을 실행하고 출력을 전체 화면에 보여줍니다.

예시: watch -n 5 'df -h'

(71) **whoami**

whoami : 현재 사용자의 사용자명을 출력합니다.

예시: whoami

(72) **hostname**

hostname : 시스템의 호스트 이름을 보여주거나 설정합니다.

예시: hostname

(73) **dig**

dig : DNS 조회를 위한 도구입니다.

예시: dig example.com

(74) **nslookup**

nslookup : 네트워크 관리, 서버 및 DNS 문제의 진단에 사용됩니다.

예시: nslookup example.com

(75) **traceroute**

traceroute : 패킷이 목적지까지 도달하기까지의 경로를 보여줍니다.

예시: traceroute example.com

(76) **ping**

ping : 다른 호스트로 ICMP ECHO_REQUEST를 보내 네트워크가 연결되어 있는지 테스트합니다.

예시: ping example.com

(77) **ifconfig**

ifconfig : 네트워크 인터페이스 구성을 보여주고 설정합니다.

예시: ifconfig

(78) **iwconfig**

iwconfig : 무선 네트워크 인터페이스를 구성합니다.

예시: iwconfig wlan0

(79) **netcat**

netcat : 네트워크 연결을 읽고 쓰기 위한 유틸리티입니다.

예시: netcat -l -p 1234

(80) **tcpdump**

tcpdump : 네트워크 트래픽을 캡처하고 표시합니다.

예시: tcpdump -i eth0

(81) **rsync**

rsync : 파일을 빠르고 변수적으로 복사 및 동기화합니다.

예시: rsync -av /src /dest

(82) **file**

file : 파일의 종류를 결정합니다.

예시: file image.jpg

(83) **stat**

stat : 파일이나 파일 시스템의 상태를 보여줍니다.

예시: stat file.txt

(84) **locate**

locate : 파일 위치를 빠르게 검색합니다. (updatedb를 통해 데이터베이스를 갱신해야 함)

예시: locate file.txt

(85) **whereis**

whereis : 바이너리, 소스, 매뉴얼 페이지 파일의 위치를 찾습니다.

예시: whereis ls

(86) **which**

which : 실행 파일의 전체 경로를 보여줍니다.

예시: which ls

(87) **type**

type : 명령어의 종류를 설명합니다.

예시: type cd

(88) **nohup**

nohup : 로그아웃 후에도 명령어가 계속 실행되게 합니다.

예시: nohup ./script.sh &

(89) **jobs**

jobs : 백그라운드 작업의 목록을 보여줍니다.

예시: jobs

(90) **bg**

bg : 작업을 백그라운드로 보냅니다.

예시: bg %1

(91) **fg**

fg : 작업을 포어그라운드로 가져옵니다.

예시: fg %1

(92) **disown**

disown : 쉘에서 작업을 분리합니다.

예시: disown %1

(93) **screen**

screen : 여러 쉘 세션을 관리할 수 있는 텍스트 기반의 윈도우 매니저입니다.

예시: screen -S session_name

(94) **tmux**

tmux : 터미널 멀티플렉서, 여러 터미널 세션을 사용하고 분리할 수 있습니다.

예시: tmux new -s session_name

(95) **script**

script : 터미널 세션의 활동을 기록합니다.
예시: script session.log
(96) **strace**
strace : 시스템 호출과 시그널을 추적합니다.
예시: strace -p 1234
(97) **ltrace**
ltrace : 라이브러리 호출을 추적합니다.
예시: ltrace -p 1234
(98) **htop**
htop : 'top'의 개선된 버전으로 시스템 프로세스를 인터랙티브하게 모니터링합니다.
예시: htop
(99) **iotop**
iotop : 디스크 I/O 사용량을 모니터링하는 도구입니다.
예시: iotop
(100) **nmap**
nmap : 네트워크 탐색 및 보안 감사를 위한 도구입니다.
예시: nmap -A example.com
(101) **sar**
sar : 시스템 활동을 보고하는 도구입니다.
예시: sar -u 1 3
(102) **vmstat**
vmstat : 시스템의 가상 메모리, 프로세스, CPU 활동 등을 보여줍니다.
예시: vmstat 1 5

4. GIT이란

Git이란 리누즈 토르발즈가 만든 리눅스 커널의 버전관리를 위해 만들었으며, 소스코드를 효과적으로 관리하기 위해 개발된 '분산 버전 관리 시스템'이다.

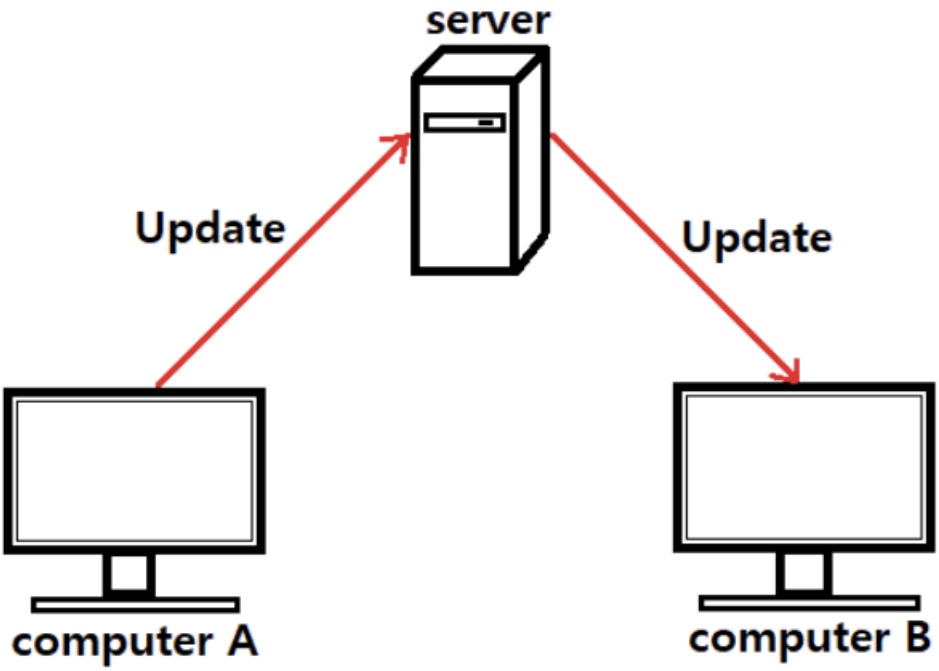
분산 버전 관리 시스템이란?

- 여러명의 개발자(분산)가 특정 프로젝트를 자신의 컴퓨터로 협업하여 개발하면서 버전을 관리할 수 있는 시스템이다.

git을 사용하는 이유

파일의 버전관리를 위해 미리 복사하는 것 대신 GIT을 사용하면 버전관리가 가능해지며 달라진 부분을 일일이 기록 가능하다. Git에서는 소스 코드가 변경된 이력을 쉽게 확인할 수 있고, 특정 시점에 저장된 버전과 비교하거나 특정 시점으로 되돌아갈 수도 있다.

GIT 버전 관리



- 컴퓨터 A에서 업데이트 한 것을 중앙 서버에 올리면 버전이 업데이트 되고, 컴퓨터 B는 이를 중앙 서버 컴퓨터로 부터 최신화 시키면서 버전과 파일을 모두 컴퓨터 A와 동일하게 유지할 수 있다.

Git의 저장소

원격 저장소(Remote repository)

- 파일이 원격 저장소 전용 서버에서 관리되며 여러 사람이 함께 공유

로컬 저장소(Local repository)

- 개인 PC에 파일이 저장되는 개인 저장소이다.

5. GIT 명령어 정리

1. **git init**

로컬 저장소를 초기화하는 명령어로, 새로운 Git 저장소를 생성할 때 사용한다.

2. **git clone**
원격 저장소를 로컬로 복제하는 명령어로, 주로 다른 사람의 저장소를 가져올 때 사용한다.

3. **git status**
현재 작업 디렉터리의 상태를 확인하는 명령어로, 수정된 파일이나 스테이징된 파일 등의 상태를 확인할 수 있다.

4. **git add**
변경된 파일을 스테이징 영역에 추가하는 명령어로, 커밋할 파일을 지정할 때 사용한다.

5. **git commit**
스테이징된 변경 사항을 로컬 저장소에 기록하는 명령어로, 변경 내역을 설명하는 메시지를 함께 작성한다.

6. **git push**
로컬 저장소에서 원격 저장소로 변경 사항을 업로드하는 명령어이다.

7. **git pull**
원격 저장소에서 최신 변경 사항을 가져와 로컬 저장소와 병합하는 명령어이다.

8. **git branch**
브랜치를 관리하는 명령어로, 현재 사용 중인 브랜치를 확인하거나 새로운 브랜치를 생성할 수 있다.

9. **git checkout**
다른 브랜치로 전환하거나 특정 커밋으로 이동할 때 사용하는 명령어이다. 브랜치를 생성하며 전환할 수도 있다.

10. **git merge**
현재 브랜치에 다른 브랜치의 변경 사항을 병합하는 명령어로, 충돌이 발생할 경우 수동으로 해결해야 한다.

11. **git log**
커밋 기록을 확인하는 명령어로, 각 커밋의 해시값, 작성자, 메시지 등을 확인할 수 있다.

12. **git reset**
특정 커밋으로 되돌리거나 스테이징 상태를 해제할 때 사용하는 명령어이다.

13. **git stash**
현재 작업 중인 변경 사항을 임시로 저장해두고 나중에 다시 적용할 수 있는 명령어이다.

14. **git remote**
원격 저장소를 관리하는 명령어로, 원격 저장소를 추가, 조회, 삭제할 수 있다.

15. **git fetch**
원격 저장소의 변경 사항을 로컬로 가져오지만 자동으로 병합하지 않으며, 병합은 수동으로 진행해야 한다.

16. git diff

두 커밋 간의 차이점을 비교하거나, 현재 변경된 파일의 차이점을 확인할 수 있다. 예를 들어 작업 디렉토리와 스테이징 영역의 차이점 또는 특정 커밋 간의 차이점을 볼 수 있다.

17. git cherry-pick

다른 브랜치에서 특정 커밋을 선택하여 현재 브랜치에 적용할 때 사용한다. 이는 전체 브랜치를 병합하는 대신, 필요한 커밋만 가져오고 싶을 때 유용하다.

18. git revert

특정 커밋을 되돌리지만, git reset과 달리 되돌린 기록을 새 커밋으로 남긴다. 프로젝트 히스토리를 유지하면서 되돌리고자 할 때 주로 사용한다.

19. git reflog

로컬 저장소의 모든 히스토리를 관리하며, 로컬에서 발생한 커밋, 체크아웃, 리셋 등의 모든 기록을 보여준다. 이를 통해 삭제된 커밋이나 변경 사항도 다시 찾을 수 있다.

20. git blame

특정 파일의 각 라인이 언제, 누가 변경했는지를 보여준다. 이를 통해 코드 작성자와 변경된 시점을 추적할 수 있다.

21. git tag

커밋에 태그를 추가하여 특정 버전을 명확하게 표시할 수 있다. 태그는 주로 릴리즈나 버전 관리에 사용되며, 태그를 통해 특정 시점으로 쉽게 돌아갈 수 있다.

22. git archive

Git 저장소의 특정 브랜치나 커밋을 압축 파일로 내보낼 수 있는 명령어이다. 저장소 전체를 복사하지 않고, 필요한 시점의 파일들만 압축본으로 만들 때 유용하다.

23. git gc (garbage collection)

Git 저장소에서 불필요한 파일이나 데이터를 정리하여 저장소의 크기를 줄이고, 성능을

최적화한다. 주기적으로 실행하면 저장소 관리에 도움이 된다.

24. git show

특정 커밋의 상세 정보를 보여준다. 해당 커밋에서 어떤 파일이 변경되었는지, 누가 작성했는지 등의 상세 내용을 확인할 수 있다.

25. git submodule

다른 Git 저장소를 현재 프로젝트 내에서 서브모듈로 추가할 때 사용하는 명령어이다. 서브모듈을 통해 외부 라이브러리나 종속된 프로젝트를 독립적으로 관리할 수 있다.

26. git clean

작업 디렉토리 내에서 추적되지 않은 파일이나 디렉토리를 삭제할 때 사용된다. 불필요한 파일을 제거하여 작업 환경을 깔끔하게 유지할 수 있다.

27. git bisect

이진 검색을 통해 코드에서 버그가 발생한 시점을 찾는 데 도움을 주는 명령어이다. 특정 커밋 사이에서 버그가 발생한 지점을 빠르게 찾아낼 수 있다.

6. Git의 분산형 협업 방식

Git은 분산 버전 관리 시스템(DVCS)으로, 각 개발자는 자신의 로컬 저장소를 이용해 작업을 진행한 후, 원격 저장소와 동기화하는 방식으로 협업을 할 수 있다. 이는 중앙 서버가 필요 없는 구조로, 중앙 서버에 문제가 생기더라도 각 개발자의 로컬 저장소에 모든 데이터가 존재하기 때문에 작업을 이어나갈 수 있다. 로컬 저장소에서의 작업이 끝나면 git push 명령어로 원격 저장소에 반영하고, 다른 개발자는 git pull로 해당 내용을 가져와 동기화할 수 있다.

7. Git의 Branch 전략

Git에서는 브랜치를 통해 개발을 분리할 수 있다. 이는 새로운 기능을 추가하거나 버그 수정을 하는 동안 기존 코드에 영향을 주지 않기 위해 사용된다. 주요 브랜치 전략으로는 Git Flow와 Github Flow가 있다.

- **Git Flow**: 안정적인 master 브랜치와 개발용 develop 브랜치를 구분하며, 기능별로 feature 브랜치를 만들어 작업 후 병합한다.
- **Github Flow**: master 브랜치만을 기준으로, 새로운 기능 추가 시 feature 브랜치

를 생성하고, 완료되면 바로 master 브랜치에 병합하는 방식이다.

8. Git의 충돌 해결

Git을 사용해 여러 개발자가 동시에 작업을 하다 보면, 같은 파일의 같은 부분을 수정하면서 충돌(conflict)이 발생할 수 있다. 충돌이 발생하면 Git은 자동으로 병합하지 못하고, 개발자가 수동으로 수정한 후 병합을 진행해야 한다. 충돌을 해결한 후에는 git add를 통해 변경 사항을 스테이징하고, git commit으로 병합을 완료할 수 있다.

9. Git Rebase와 Merge의 차이

Git에서 병합을 할 때, merge와 rebase 두 가지 방법이 있다.

- **git merge**는 두 브랜치의 변경 사항을 병합하면서 추가적인 커밋을 생성한다. 병합의 이력과 과정을 보존하기 때문에 협업 시 주로 사용된다.
- **git rebase**는 현재 브랜치를 다른 브랜치의 최신 커밋 위로 재배치하는 방식이다. 이 방법은 깔끔한 커밋 히스토리를 유지할 수 있지만, 충돌 발생 시 해결이 어려울 수 있다.

10. Git Tag

Git에서는 특정 커밋을 태그(tag)로 표시할 수 있다. 태그는 주로 버전 관리에 사용되며, 주요 릴리즈 시점에 커밋을 태그하여 관리한다. 태그는 git tag <태그명> 명령어로 생성할 수 있으며, 이를 이용해 나중에 특정 버전으로 쉽게 돌아갈 수 있다.

11. Git에서의 협업

Git을 사용한 협업에서는 주로 Pull Request(또는 Merge Request) 기능이 자주 사용된다. 이는 브랜치를 병합하기 전에 코드 리뷰를 통해 다른 개발자들이 코드를 확인하고 승인할 수 있는 절차를 제공한다. GitHub, GitLab 등 대부분의 Git 호스팅 플랫폼에서 지원하는 기능이다.

12. Git의 Gitignore 파일

.gitignore 파일은 Git에서 추적하지 않을 파일을 지정하는 데 사용된다. 빌드 결과물이나 개인 환경 설정 파일처럼 버전 관리에 포함되지 않아야 할 파일들을 이 파일에 추가하면, git add 명령어로도 해당 파일들이 Git에 추가되지 않는다.

13. Git 설정과 초기화

전역 사용자명/이메일 구성하기

git config - -global user.name “Your name”

git config - -global user.email “Your email address”

저장소별 사용자명/이메일 구성하기 (해당 저장소 디렉터리로 이동 후)

git config user.name “Your name”

git config user.email “Your email address”

전역 설정 정보 조회

git config - -global - -list

저장소별 설정 정보 조회

git config - -list

Git의 출력결과 색상 활성화하기

git config - -global color.ui “auto”

새로운 저장소 초기화하기

mkdir /path/newDir

cd /path/newDir

git init

저장소 복제하기

git clone <저장소 url>

새로운 원격 저장소 추가하기

git remote add <원격 저장소> <저장소 url>

사용법

새로운 파일을 추가하거나 존재하는 파일 스테이징하고 커밋하기

git add <파일>

git commit -m "<메시지>"

파일의 일부를 스테이징하기

git add -p [<파일> [<파일> [기타 파일들...]]]

add 명령에서 Git 대화 모드를 사용하여 파일 추가하기

git add -i

수정되고 추적되는 파일의 변경 사항 스테이징하기

git add -u [<경로> [<경로>]]

수정되고 추적되는 모든 파일의 변경 사항 커밋하기

git commit -m "<메시지>" -a

작업 트리의 변경 사항 돌려놓기

git checkout HEAD <파일> [<파일>]

커밋되지 않고 스테이징된 변경 사항 재설정하기

git reset HEAD <파일> [<파일>]

마지막 커밋 고치기

git commit -m “<메시지>” - -amend

이전 커밋을 수정하고 커밋 메시지를 재사용하기

git commit -C HEAD - -amend

3. 브랜치

지역 브랜치 목록 보기

git branch

원격 브랜치 목록 보기

git branch -r

지역과 원격을 포함한 모든 브랜치 목록 보기

git branch -a

현재 브랜치에서 새로운 브랜치 생성하기

git branch <새로운 브랜치>

다른 브랜치 체크아웃하기

git checkout <브랜치>

현재 브랜치에서 새로운 브랜치 생성하고 체크아웃하기

git checkout -b <새로운 브랜치>

다른 시작 지점에서 브랜치 생성하기

git branch <새로운 브랜치> <브랜치를 생성할 위치>

기존의 브랜치를 새로운 브랜치로 덮어쓰기

git branch -f <기존 브랜치> [<브랜치를 생성할 위치>]

브랜치를 옮기거나 브랜치명 변경하기

git checkout -m <기존 브랜치> <새로운 브랜치>

- <새로운 브랜치>가 존재하지 않을 경우

git checkout -M <기존 브랜치> <새로운 브랜치>

- 무조건 덮어쓰기

다른 브랜치를 현재 브랜치로 합치기

git merge <브랜치>

커밋하지 않고 합치기

git merge - -no-commit <브랜치>

선택하여 합치기

git cherry-pick <커밋명>

커밋하지 않고 선택하여 합치기

git cherry-pick -n <커밋명>

브랜치의 이력을 다른 브랜치에 합치기

git merge - -squash <브랜치>

브랜치 삭제하기

git branch -d <삭제할 브랜치>

- 삭제할 브랜치가 현재 브랜치에 합쳐졌을 경우에만

git branch -D <삭제할 브랜치>

- 삭제할 브랜치가 현재 브랜치에 합쳐지지 않았어도