**스터디 1주차 : git 명령어 및 역할**

**Git**

* 버전 변화를 기록하는 파일 관리 시스템
* 데이터를 파일 시스템 스냅샷의 연속으로 취급
* 중요한 순간을 저장하여 언제든 원래 내용으로 돌아갈 수 있게 함
* 거의 모든 명령을 로컬에서 실행(네트워크의 속도의 영향 x)
* 주요 개념
  + 저장소: 프로젝트 저장 및 관리. 로컬, 원격 존재
  + 커밋(Commit): 코드의 특정 상태를 저장하는 스냅샷
  + 브랜치(Branch): 코드 작업을 여러 갈래로 나누어 작업
  + 병합(Merge): 여러 갈래로 나뉜 작업을 하나로 합침
* Committed, Modified, Staged 세 가지 상태로 관리
  + Committed: 데이터가 로컬 저장소에 안전하게 저장 (Git directory)
  + Modified: 수정한 파일을 아직 로컬 저장소에 커밋하지 않음
  + Staged: 현재 수정한 파일을 곧 커밋할 것이라고 표시한 상태 (Staging Area)



* 워킹트리, Staging Area, Git directory
* Git directory: 프로젝트의 메타데이터와 객체 데이터베이스를 저장하는 곳. 다른 컴퓨터에 있는 저장소를 clone할 때 Git directory 만들어짐.
* 워킹트리: 프로젝트의 특정 버전을 Checkout 한 것. Git directory는 지금 작업하는 디스크에 있고 그 디렉토리 안에 압축된 데이터베이스에서 파일을 가져와 워킹 트리를 만들음.
* Staging Area(Index): Git directory에 존재. 단순한 파일로 곧 커밋할 파일에 대한 정보 저장
* Git에서의 작업 순서
  + 워킹 트리에서 파일 수정
  + Staging Area에 파일을 Stage 해서 커밋할 스냅샷 생성
  + Staging Area에 있는 파일들을 커밋해서 Git directory에 영구적인 스냅샷 저장

**Git hub**

* 원격 저장소를 제공하는 플랫폼
* 인터넷에 프로젝트를 업로드 하여 다른 사람과 협업, 공유 가능

**Git과 Github를 연결**

1. git으로 관리할 디렉토리에서 git init 실행하여 해당 디렉토리를 git 저장소로 설정
2. github에 원격 저장소 생성
3. git remote add <단축이름> <url>
4. git push [리모트이름] [브랜치명] 사용하여 로컬 저장소에서 원격 저장소로 파일 업로드.
5. 원격 저장소에서 로컬 저장소로 변경사항 가져오기
   1. git pull [리모트이름] [원격 브랜치명] 사용 (자동 병합)
   2. git fetch로 원격 변경 사항 가져오기 (수동 병합)

-> git merge [리모트이름]/[브랜치명] 으로 원격 브랜치 로컬에 병합

이때 확인할 점: 작업 중인 브랜치와 원격 브랜치의 연결 상태 git branch -vv

연결 되어 있지 않으면 git branch --set-upstream-to=origin/[브랜치명] [로컬브랜치명]

**많이 쓰는 리눅스 명령어**

* ls : 현재 디렉토리에 있는 파일, 폴더 출력
* mkdir 디렉토리 이름 : 디렉토리 이름을 가지는 폴더 생성
* cd 디렉토리 이름 : 디렉토리 위치로 이동
* pwd : 현재 경로 출력
* touch 파일명 : 파일명 파일 생성
* cp 파일1 파일2 : 파일1과 동일한 내용의 파일을 파일 2의 이름으로 생성
* rm 파일이름 : 해당 파일 삭제
* rmdir 디렉토리 이름 : 해당 폴더 삭제
* cat 파일명: 터미널 창에서 문서의 내용을 표시
* 명령어 앞 sudo: 관리자 권한으로 실행

**편의기능**

* tap : 존재하는 폴더 이름, 파일 이름 자동 완성
* 화살표 위, 아래: 이전에 썼던 명령어를 똑같이 사용할 때 사용

**설정**

* git help <verb>: 명령어에 대한 도움말 확인 (또는 명령어 뒤에 -h, --help옵션)
* git config : 환경설정 명령어 ( --global: 모든 사용자와 모든 저장소에 적용)
  + git config --global user.name “이름”: 사용자 이름
  + git config --global user.email 아이디@이메일.주소: 이메일 주소
  + git config –list: 설정 확인
  + git config <key>: 특정 key에 대해 어떤 값을 사용하는지 확인
* git init: 해당 디렉토리를 git 저장소로 설정
* git remote add <리모트 이름> <url>: 새 원격 저장소와 로컬 저장소를 연결
  + git remote: 현재 저장소 확인(origin이라는 리모트 이름 나옴)
  + git remote -v: 리모트 이름과 url 함께 나옴
* git clone <url>: 다른 프로젝트에 참여하거나 Git 저장소를 복사할 때 사용. 프로젝트 히스토리를 전부 받아옴

**소스 기록**

* git add: 작업 디렉토리 상의 변경 내용을 스테이징 영역에 추가

**소스 커밋**

* git status: 현재 작업의 상태를 볼 수 있는 명령어(HEAD, 워킹트리, 스테이징 영역 비교)
* git commit: 코드 변경 사항을 저장소에 영구적으로 기록(파일이나 폴더의 변경 사항을 로컬 저장소에 기록, 고유 ID, 변경 내역, 작성자 정보, 변경사항에 대한 설명 등)

**소스 업데이트**

* git pull [리모트 이름]: 원격 저장소에 있는 모든 내용을 가져와 로컬의 작업 내용과 합침(fetch, merge 하나로 합침)
* git fetch [리모트 이름]: 원격 저장소에서 최신 변경 사항을 로컬로 가져오기만 합치진 않음

**소스 복원**

* git reset: 커밋, 스테이징 영역, 작업 디렉토리 상태를 변경
  + --soft [커밋 해시]: 커밋 히스토리 되돌림, 변경 사항 스테이징 영역에 남김
  + --mixed(기본 옵션) [커밋 해시]: 커밋 히스토리 되돌림, 변경 사항 스테이징 영역에서 제거, 작업 디렉토리 유지
  + --hard [커밋 해시]: 커밋 히스토리 되돌림, 스테이징 영역과 디렉토리를 모두 해당 커밋 상태로 변경
* git restore: 파일이나 디렉토리의 변경 내용을 취소하거나 삭제된 파일 복구

**브랜치**

* git branch [브랜치명]: 새 브런치 생성 명령어
* git checkout/ git switch [브랜치명]: 지정한 브랜치로 이동
* git push [리모트 이름] [브랜치명]: 로컬 브랜치를 원격 저장소로 push 할 때 사용

**소스 병합**

* git merge [리모트이름]/[브랜치명]: 로컬 브랜치에 병합
* git rebase: 브랜치의 base commit를 재설정 (깔끔한 커밋 히스토리 유지)

**파일 삭제, 저장, 수정**

* git rm: git에서 파일 삭제 및 git의 추적 상태에서 제거
* git clean: 작업 트리에서 추적되지 않은 파일 제거
* git stash: 새로운 stash를 스택에 만들어 작업 임시 저장
* git revert: 이미 완료된 커밋을 취소하는 새로운 커밋을 생성하는 명령어
* git tag: 소스 버전을 표시할 때 사용. 조회/생성/삭제 등

**작업 이력**

* git reflog: git에서 HEAD가 변경된 모든 작업의 이력을 보여줌
* git log: 브랜치의 커밋 히스토리 확인
* git diff: 파일을 수정한 뒤 수정된 사항을 확인할 수 있게 함(작업 사항 add전)
* git show: 해당 커밋이 어떻게 변경되었는지
* git blame: 메소드의 마지막으로 언제, 누가 커밋했는지 기록 확인