Git & GitHub

Git

Git은 로컬 환경에서 구동되는 버전 관리 도구다.

버전 관리 도구를 사용하면 프로젝트 파일의 유지보수, 저장 용량, 코드 수정 이력(누가, 언제) 등 전반적인 면에서 장점이 있다.

Git을 사용하기 위해서는 https://git-scm.com/ 싸이트에 접속하여 운영체제에 맞는 Git을 다운로드하여 설치하면 된다.

(설치는 다운로드한 파일을 실행하여 기본값으로 설치를 진행하면 된다)

GitHub

Git으로 관리되는 폴더, 파일 등의 웹 호스팅 서비스다. 구글 드버이버와 같이 파일을 공유하는 싸이트라 보면 된다.

GitHub는 단순히 웹 상에 파일을 공유하는 저장소의 역할만 하는 것이 아니다.

팀 프로젝트 소스코드를 GitHub에 공유하면 프로젝트 관리에 필요한 커뮤니티, 이슈 생성, 코드 리뷰 등 킴 프로젝트 관리에 있어 다양한 기능을 제공한다.

GitHub 싸이트(https://github.com/)에 접속하여 회원가입 후 로그인을 통해 사용이 가능하다.

git을 실행하는 방법은 아래의 3가지 방식이 있다.

- 바탕화면 혹은 특정 폴더 안에서 마우스 우측 클릭 -> 'open Git Bash here' 선택
- vscode 터미널에서 Git 사용
- Intellij 터미널에서 Git 사용

위 3가지 중 상황에 따라 편한 방법을 사용하면 된다.

git 사용을 위해 먼저 사용자를 지정해야 한다(pc에서 최초 한 번만 실행)

- pc에 저장된 git 사용자의 이메일 확인

  git config --global user.email

- pc에 사용자의 이메일 추가 및 변경

  git config --global user.email 이메일주소

- pc에 저장된 git 사용자의 name 확인

  git config --global user.name

- pc에 사용자의 name 추가 및 변경

  git config --global user.name 사용자의 git username

git과 github를 아래의 순서를 따라하며 실습해보자.

1. git으로 버전 관리가 필요한 폴더 생성
2. 해당 폴더 안에서 git bash를 실행
3. 아래 명령어를 입력하여 현재 폴더를 git 지역저장소(local repository)로 만들기

   지역지장소가 되면 해당 폴더 안의 파일 및 폴더가 git에 의한 버전 관리 대상 후보가 됨

   git init

   위 명령어를 입력하면 .git이라는 숨김 폴더가 생성한다. .git 폴더가 생성된 폴더 안의 파일 및 폴더들은 git의 버전관리 대상이 된다.

   .git 폴더 안에는 버전 정보, 원격 저장소 주소 등이 저장된다.
4. 현재 폴더에 임의의 텍스트 파일을 2개 만들고 내용도 아무 내용이나 작성하고 저장한다.
5. 아래 명령어를 입력하여 지역저장소의 상태 확인하기

   git status

   위 명령어 입력 시 지역저장소에 추가한 2개의 텍스트 파일이 untracked 상태로 표시된다.

   untracked 상태란 파일이 버전관리 대상이 되는 폴더에는 존재하지만, 아직 버전관리를 시작하지 않은 상태를 의미한다.

6. 파일을 버전관리 대상 파일로 지정하기 위해서는 아래의 명령어를 사용한다.

   git add 파일명

   위 명령어를 하나의 텍스트 파일에 적용 후 다시 git status를 입력하여 지역저장소 상태를 확인하면 untracked와 commited 라는 상태로 각각의 파일 상태가 변경된 것을 확인 할 수 있다.

   commited 상태가 된 파일은 본격적으로 git의 버전관리 대상 파일이 된다.

   이렇게 git add 명령어를 통해 git의 버전 관리 대상 파일을 지정하는 것을 'staging' 이라 표현한다.

   .git 폴더가 존재하는 폴더는 대형마트, 해당 폴더에 존재하는 모든 파일 및 폴더는 대형마트에 진열된 상품들은 untracked,

   진열된 상품 중 구매를 위해 카트에 옮겨담은 상품은 staged로 생각하자.

   git add . 명령어로 모든 파일을 일괄 staging 할 수 있다.

7. 변경 사항 저장 명령 수행.

   git commit -m "커밋메세지"

   변경 사항을 저장하기 위해서는 commit 명령어를 사용한다. 현재의 변경 사항은 새로운 텍스트 파일의 추가이다.

   commit은 지금까지의 내용을 저장하는 명령어이면서 동시에 복원시점을 만드는 명령어이므로 자주 해주는 것이 좋다.

   커밋메세지는 상세히 작성하자. 커밋메세지로 어떤 내용을 수정했는지 가늠할 수 있어야 한다.

add 명령어 실행 시 간혹 LF... 으로 시작하는 경고 메세지가 뜰 수 있다. 운영체제마다 다른 문자 처리 방식 때문에 일어나는 경고로, 해당 경고가 뜨면 git config core.autocrlf true 명령어를 입력하면 된다. 그 후 commit을 진행하면 된다.

8. Github 싸이트에서 원격 저장소(remote repository) 생성.

   이 작업은 git으로 관리되는 폴더를 웹에 공유하기 위한 폴더를 만드는 것이라 생각하면 된다.

9. 지역 저장소와 원격 저장소 연결.

   git remote add origin 원격저장소url

   위 명령에서 origin은 원격 저장소에 이름을 부여한 것이다. 그렇기 때문에 origin 대신 단어를 입력해도 되지만, 거의 모든 사용자들이 origin이라 이름을 지정하기 때문에 우리도 항상 origin으로 이름을 부여할 것이다.

10. 원격 저장소와 지역 저장소의 브렌치(branch) 이름 통일 시키기.

    git branch -m main

11. 지역 저장소에서 작업한 내용을 원격 저장소에 올리기.

    git push origin main

    위 명령어를 입력하면 지금까지 지역 저장소에서 작업한 내용을 github 싸이트에 올릴 수 있다.

이후에는 파일 추가 및 삭제, 코드 변경 등의 작업 -> 6번 (add) -> 7번 (commit) -> 11번 (push) 순으로 반복하면 된다.

1. Github 싸이트에 올린 파일을 내려받을 폴더에서 git bash 시작
2. 아래 명령어를 입력하여 파일 받기

   git clone 원격저장소url

   위 명령어를 입력하면 원격저장소 이름으로 폴더가 생성되고, 해당 폴더에 원격저장소의 모든 파일을 내려 받는다.

   폴더 생성 후 해당 폴더 안에서 git clone 원격저장소url . 명령어를 입력하면 폴더 없이 원격저장소의 파일만 내려받을 수 있다.

   원격 저장소에서 내려받은 파일은 .git 폴더가 이미 존재하기 때문에 자동으로 git이 버전관리를 수행하는 폴더가 된다.

이후에는 Git, Github 사용해보기에서 학습한 순서대로

파일 추가 및 삭제, 코드 변경 등의 작업 -> 6번 (add) -> 7번 (commit) -> 11번 (push) 순으로 반복하면 된다.

개인 프로젝트나 학습한 내용을 git, github로 강의실과 집에서 사용할 때는 다음과 같이 진행할 수 있다.

아래는 강의실에서 학습한 코드를 최초 github에 올리는 방법이다.

1. github 싸이트에 학습한 내용을 올릴 원격 저장소 생성
2. 학습한 폴더에서 git bash 실행
3. 해당 폴더를 git이 관리하는 지역 저장소로 지정

   git init
4. 원격 저장소와 지역 저장소의 branch 이름 통일을 위해 지역 저장소의 branch명을 main으로 변경

   git branch -m main
5. 원격 저장소와 지역 저장소를 연결

   git remote add origin 원격저장소url
6. 작업한 모든 내용을 staging (주의!. 해당 폴더에 동영상 같은 파일 크기가 큰 파일이 있으면 안 됨!!!)

   git add .
7. staging된 모든 내용을 저장

   git commit –m “커밋메세지”
8. 저장된 내용을 원격 저장소에 반영

   git push origin main

다음은 강의실에서 올린 파일을 집에서 내려받는 방법이다.

1. 파일을 내려받을 폴더에서 git bash 실행
2. github 원격 저장소에 올린 원격저장소 복사하여 가져오기

   git clone 원격저장소url

   (만약, 원격저장소에 저장된 내용만 가져오고 싶다면 git clone 원격저장소url .)
3. 코드 추가, 수정 등 작업 진행
4. 작업한 모든 내용을 staging (주의!. 해당 폴더에 동영상 같은 파일 크기가 큰 파일이 있으면 안 됨!!!)

   git add .
5. staging된 모든 내용을 저장

   git commit –m “커밋메세지”
6. 저장된 내용을 원격 저장소에 반영

   git push origin main

이렇게 강의실과 집에서 작업할 내용을 git과 연결하면, 이후에는 아래의 내용을 반복한다.

1. 집 혹은 강의실에서 작업한 내용 내려받기

   git pull origin main

   (github에 올린 파일을 최초 내려받을 때는 clone을 사용하고, 그 이후에는 pull을 사용한다)

   pull 명령어를 사용하지 않으면 conflict(충돌)가 발생함으로, 작업 전 반드시 pull을 통해 최신 내용을 내려받아야 한다.
2. 코드 추가, 수정 등 작업 진행
3. 작업한 모든 내용을 staging (주의!. 해당 폴더에 동영상 같은 파일 크기가 큰 파일이 있으면 안 됨!!!)

   git add .
4. staging된 모든 내용을 저장

   git commit –m “커밋메세지”
5. 저장된 내용을 원격 저장소에 반영

   git push origin main

git reset과 git revert는 특정 시점의 커밋으로 돌아가고 싶을 때 사용하는 명령어이다.

- git reset

  특정 커밋 시점으로 돌아가며, 이후의 커밋들은 모두 없었던 일로 처리한다. 과거로 되돌아가는 것과 같다.

  reset을 통해 되돌아가면 그 후의 커밋 이력들도 모두 사라진다.

  reset은 팀 작업 시 push까지 진행한 상태라면 사용하면 안된다. 가급적 개인 파일에만 사용하도록 하자.

  reset 명령어를 사용할 때는 soft, mixed, hard의 3가지 옵션 중 하나를 선택해야 한다.

- git revert

  특정 커밋 시점의 내용만 취소하는 것으로 해당 커밋 이후의 내용은 남아있다.

  또한, revert를 사용해 특정 커밋을 취소한 것 또한 이력으로 남겨지기 때문에 팀원들이 revert한 이력을 확인할 수 있다.

  팀 작업 시 push까지 한 상태라면 revert를 사용해야 한다.

reset, revert 참고자료

https://www.devpools.kr/2017/01/31/%EA%B0%9C%EB%B0%9C%EB%B0%94%EB%B3%B4%EB%93%A4-1%ED%99%94-git-back-to-the-future/

다음은 git reset 명령어의 옵션과 명령어 사용 방법이다.

옵션	설명	사용 방법
--soft	커밋만 되돌리고, 변경 파일은 그대로 둠(staing 상태 유지) -> 수정 내용은 살려두고 저장하지 않은 상태로 돌아간다는 의미	git reset --soft commit hash
--mixed (default)	커밋을 되돌리고, 스테이징도 취소하지만 변경 파일은 그대로 둠(staing 상태 유지) -> 수정 내용은 살려두고 저장하지 않은 상태로 돌아간다는 의미. soft와의 차이점은 staging전이냐 후냐의 차이이다. 잘 모르겠으면 mixed 사용하면 됨.	git reset --mixed commit hash git reset commit hash
--hard	커밋을 되돌리고, 변경한 내용도 삭제 -> 되돌린다의 가장 기본적 의미. 변경 내용을 전부 없었던 일로 처리한다.	git reset --hard commit hash

commit hash는 git log 명령어로 알수 있다.

git bash에서는 복사/붙여넣기 단축키가 다르다. -> 복사 : Ctrl + Insert, 붙여넣기 : Shift + Insert

branch는 가지, 지사 등의 의미이다. branch를 사용하면 특정 시점에서의 복사본을 만들어 기존 내용에 영향을 주지 않고 새로운 작업을 진행 할 수 있다.

다음의 기본적인 branch 관련 명령어다.

명령어	설명
git branch	브랜치 목록 확인. 현재 브랜치명 앞에는 * 로 표시.
git branch 브랜치명	새로운 브랜치 생성 브랜치를 생성하면 자동으로 내용도 복사 됨
git checkout 브랜치명 git switch 브랜치명	특정 브랜치로 이동
git checkout -b 브랜치명 git switch -c 브랜치명	새로운 브랜치 생성 및 이동
git merge 병합할 브랜치명	현재 브랜치에서 다른 브랜치 병합
git branch -d 브랜치명	병합된 브랜치 삭제