Git 기초

CAT&DOG 기초 스터디 2차시

시작하기

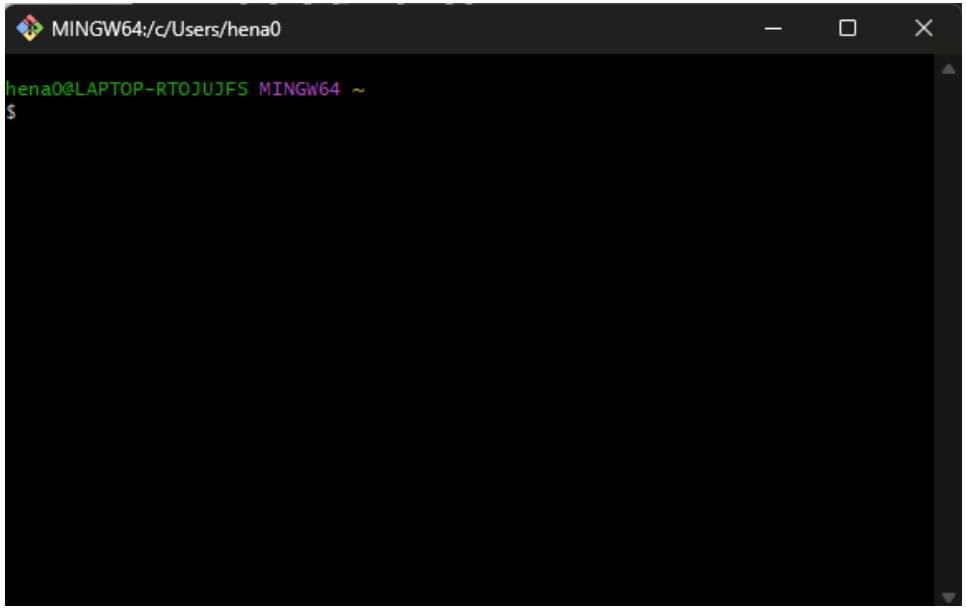
- Download Site: https://git-scm.com/
- 터미널에서 아래 명령어를 실행해서, git이 설치되었는지 확인

```
git --version
```

시작하기

Git bash

: 명령어(Git CLI)로 깃을 사용 가능하다.



시작하기

CLI vs GUI

- CLI(Command Line Interface): 명령어를 텍스트로 입력하는 방법
- GUI(Graphical User Interface): 시각적인 인터페이스를 사용해 명령어들을 사용하는 방법

Git을 직관적으로 이해하기 위해서는 GUI 툴을 사용할 줄 아는 것이 편하다.

CAT&DOG에서는 주로 GUI를 사용해 프로젝트를 진행하기 때문에 GUI를 중심으로 Git의 사용법을 익힐 예정이다.

대표적인 GUI 툴: Fork, GitKraken, SourceTree, GitHub Desktop 등

시작하기

GUI 툴 선택

자신에게 익숙한 툴을 사용하셔도 상관 없습니다.

본 수업에서는 Fork를 사용하도록 하겠습니다.

Fork Download Site: https://git-fork.com/

Git

프로젝트 진행 중 문제상황

- 작업물을 어떻게 공유할까?
- 이전 버전으로 돌아가고 싶을 때는?

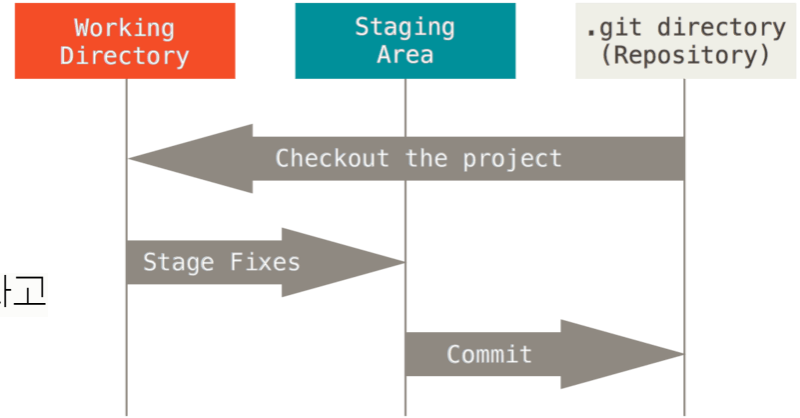
Git의 역할

- source code 관리: 버전 관리, 작업물 병합, 공유, 브랜치 관리
- Git이란, 버전 관리 시스템이다

Git

Git에서 파일의 세가지 상태: Committed, Modified, Staged

- Committed: 데이터가 로컬 데이터베이스에 안전하게 저장됐다는 것
- Modified: 수정한 파일을 아직 로컬 데이터베이스에 commit하지 않은 것
- Staged: 현재 수정한 파일을 곧 commit할 것이라고 표시한 상태



Git으로 하는 일

1. Working Tree에서 파일 수정
2. Staging Area에 파일을 Stage해서 commit할 스냅샷을 만들기
3. Staging Area에 있는 파일들을 commit해서 git directory에 영구적인 스냅샷으로 저장

GitHub

GitHub란?

- Git의 remote repo (원격 저장소)

Git과 GitHub의 차이

- GitHub: Git을 기반으로, 소스 코드를 저장하는 원격 저장소의 역할을 함
- 오픈 소스 프로젝트와 협업을 위해 사용

Init & Clone

Init

- 로컬에서 작업중인 폴더를 git repo로 만들거나, 새 repository를 만들 때 사용
- Git CLI: $ git init

Clone

- Github에 있는 프로젝트를 로컬에 복제함
- Git CLI: $ git clone https://github.com/KU-CATDOG/example.git

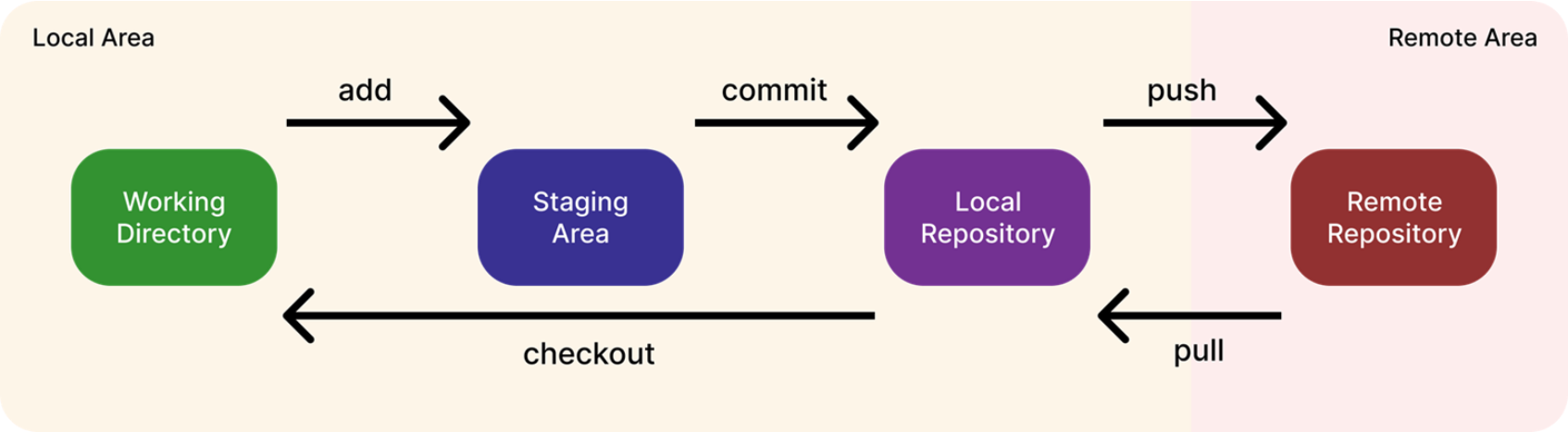
Clone

Remote

- 로컬 디렉토리와 원격 저장소를 연결함
- Git CLI: $ git remote add <원격 저장소명> <원격 저장소 URL>
- $ git remote -v: 연결된 원격 저장소를 확인할 수 있음

Add, Commit, Push

Git에서 변경 사항은, 아래와 같은 방식으로 저장된다.



Add, Commit, Push

Add

- working directory에서의 변경 사항을 staging area에 올림
- Git CLI:
  - $ git add 파일명
  - $ git add . // 모든 파일을 add (‘.’은 모든 파일 의미)

```
$ git add test.txt
```

Add, Commit, Push

Commit

- staging area에 있는 변경 사항을 local repo에 저장
- add해둔 파일들에 대한 변경 사항만 repo에 저장됨
- Git CLI: $ git commit -m “commit message”

Add, Commit, Push

Push

- Local repo를 remote repo(github)로 올려서 공유
- remote로 연결된 원격 저장소에 push 가능
- Github 웹에서 push한 파일을 확인할 수 있음
- Git CLI: $ git push origin targetbranch
    - origin이라는 remote repo의 targetbranch에 변경사항을 push

restore & reset

Restore

- commit하지 않은 실수를 되돌림
- Git CLI
  - $ git restore <되돌릴 파일 이름> // add하지 않은 변경 사항을 수정 전으로 되돌림
  - $ git restore -staged <되돌릴 파일 이름> // add한 변경사항을 취소

Reset

- commit한 실수를 되돌림
- Git CLI
  - $ git reset -hard <돌아갈 커밋 고유번호> // commit으로 돌아가고 working directory 초기화
  - $ git reset -mixed <돌아갈 커밋 고유번호> // commit으로 돌아가고 현재 working directory 유지
  - $ git reset -soft <돌아갈 커밋 고유번호> // commit으로 돌아가고 staging area의 변경 사항 유지

Pull

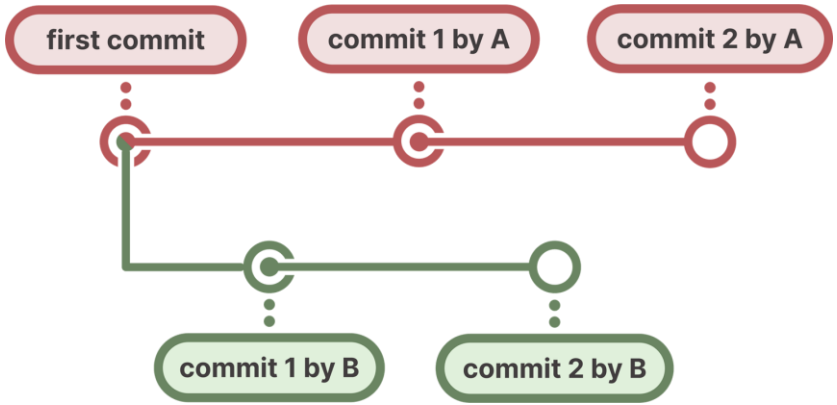
Pull

- 원격 저장소의 내용을 local repo에 저장
- Git CLI: $ git pull <원격 저장소명> <branch 이름>

Branch

Branch

- 특정 commit을 기준으로, 각자의 변경 사항을 따로 commit할 수 있도록 나눈 것
- 병렬적 작업을 위해 사용
- Git CLI
  - 브랜치 생성: $ git checkout -b “init”
  - 브랜치 전환: $ git switch “init”
  - 브랜치 목록 조회: $ git branch
  - 브랜치 삭제: $ git branch -d “init”



Merge & Conflict

Merge

- branch를 나눠 작업한 내용을 하나로 합치는 작업
- Git CLI: $ git merge <현재 branch와 병합할 branch 이름>

Merge Conflict

- 하나의 파일에 대한 변경 사항이 병합할 두 branch 모두에게 있어서, 이를 통합해야 한다는 뜻
- GUI 툴, vscode 등에서 merge conflict를 해결하기 위한 기능을 사용하면 됨
- CAT&DOG 협업 시: 되도록 건드리지 말고 '메인 프로그래머 호출하세요'

Pull Request

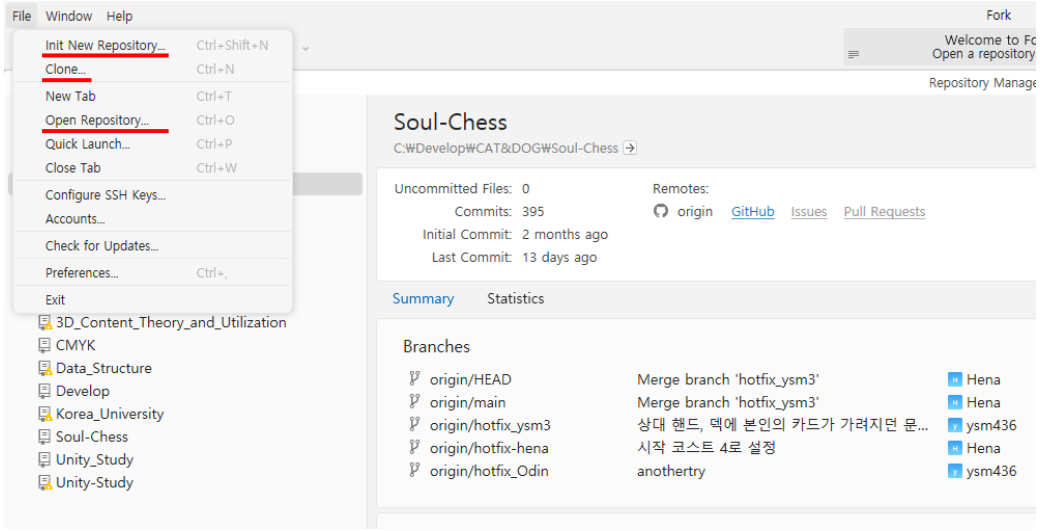
Pull Request

- push한 branch 내용을 merge하기 전에 pull해서 체크해달라는 의미
- 주로 fork한 상황에서 사용
- CAT&DOG에서는 코드를 merge하는 과정에서 서로 코드를 확인할 수 있고 어떤 부분을 수정했는지도 확인할 수 있기 때문에 사용

Git GUI

Init

- init New Repository 버튼
- Open Repository 버튼:
  기존에 존재하는 저장소를
  Fork를 통해 열 수 있음



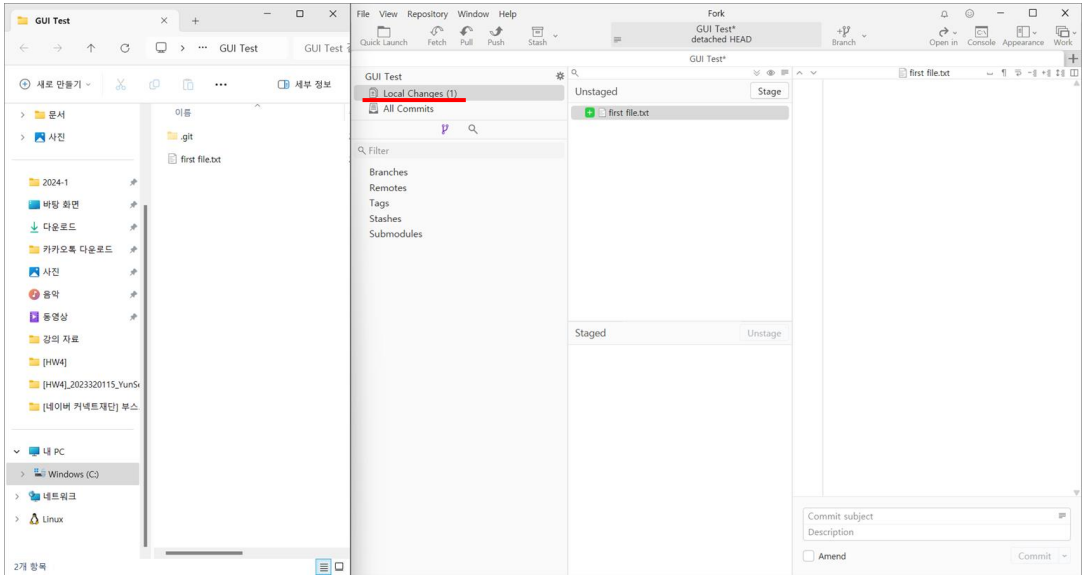
Clone

- Clone 버튼

Git GUI

Status

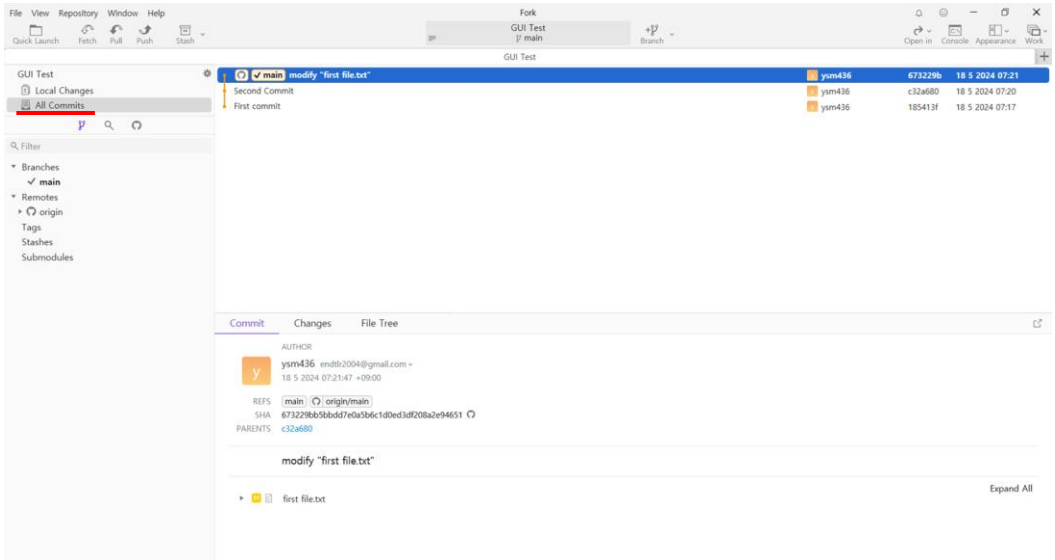
- Local Changes 화면: 파일에 변화를 주면 변경 사항의 상태를 알 수 있음
- 일일이 $ git status를 통해 변경 사항의 상태를 확인하지 않아도 됨



Git GUI

Commit Log

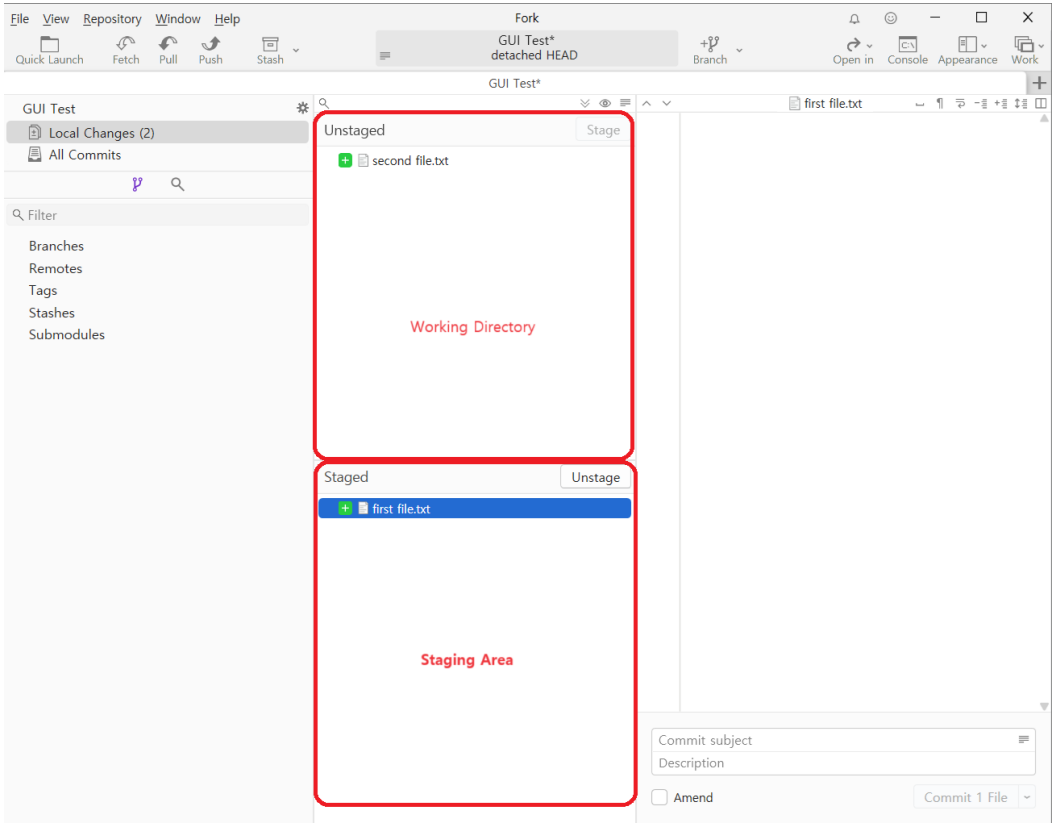
- All Commits 화면: commit log를 볼 수 있음
- 원하는 commit을 더블 클릭하면 바로 그 commit으로 checkout 가능



Git GUI

Stage, Unstage

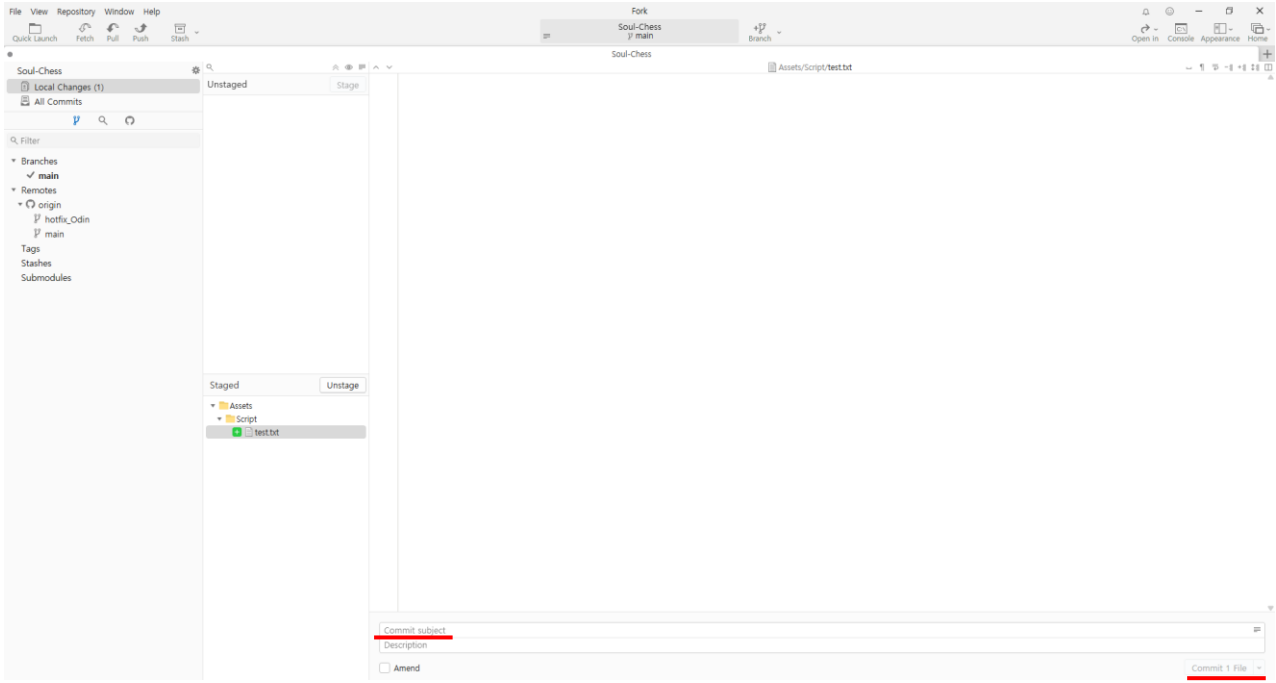
- Working Directory에 있는 파일과 Staging Area에 있는 파일들을 볼 수 있음
- Stage 버튼: git add
- Unstage 버튼: git restore —staged



Git GUI

Commit

- Commit message와 그에 대한 설명을 적고 commit 버튼을 누르면 됨



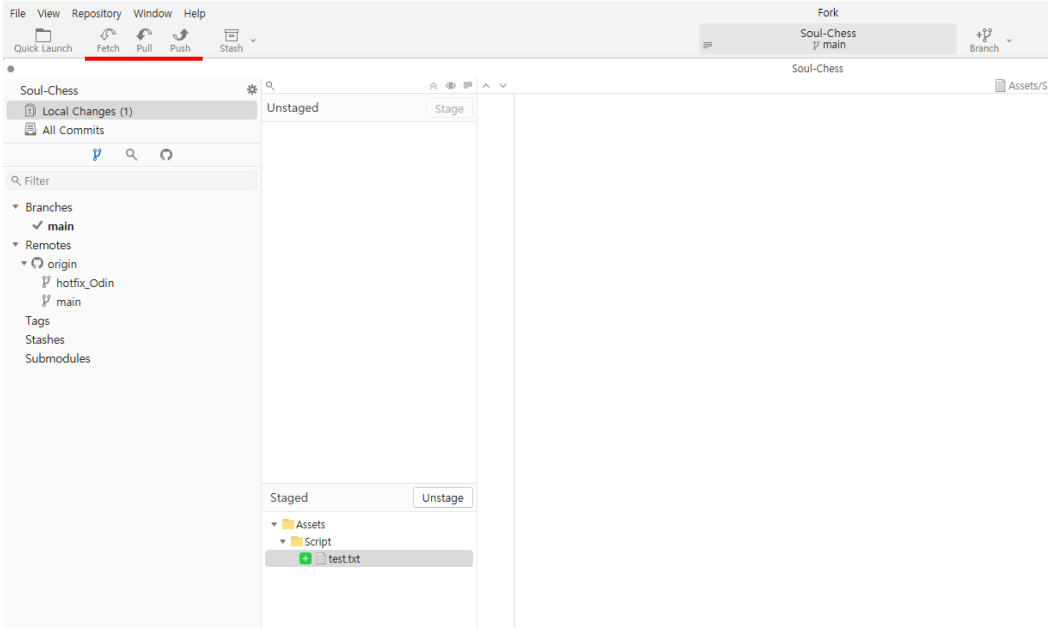
Git GUI

Push, Pull

- push: 좌상단에 있는 push 버튼
- pull: push 옆에 있는 pull 버튼

Fetch

- Remote repo의 변경 사항을 확인만 하고 local로 가져오지는 않음



실전 프로젝트

프로젝트 생성

1. Github에 repo 만들기
2. Local에서 clone 및 remote add 하기
3. gitignore 추가
4. 프로젝트 파일 생성 및 initial commit
5. branch 전략에 따라 프로젝트 시작하기

실전 프로젝트

1. Github에 repo 만들기

- github -> your repositories -> new
- 저장소 이름 설정 및 private / public 선택

실전 프로젝트

2. Local에서 clone 및 remote add 하기

- Git GUI
  - Clone 버튼
- Git CLI
  - $ git clone <레포지토리명>
  - $ git remote add origin <레포지토리명>

실전 프로젝트

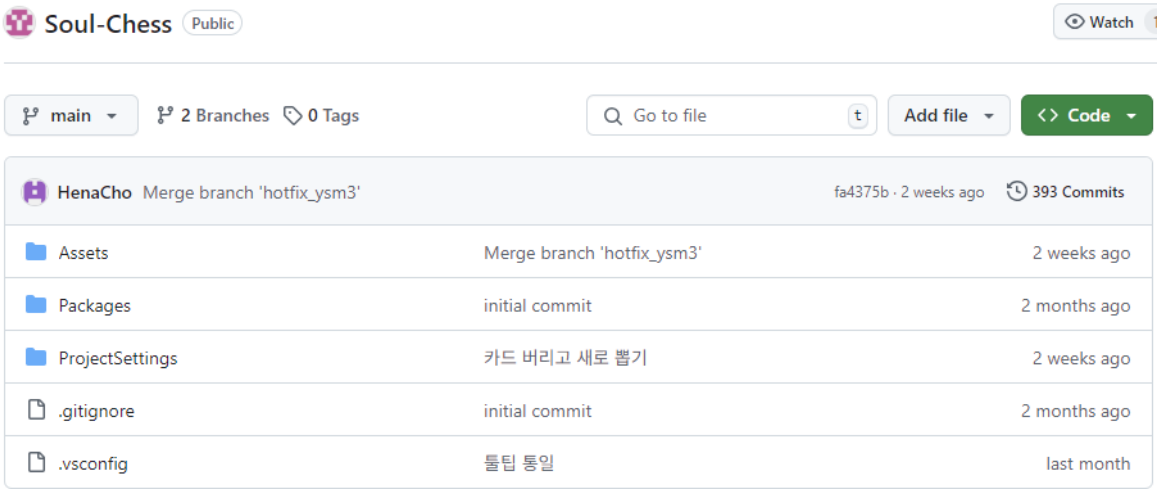
3. gitignore 추가

- .gitignore 파일: Git 버전 관리에서 제외할 파일 형식 목록을 지정하는 파일
- Github repo 생성 시 Add .gitignore > Unity를 선택하면, unity 프로젝트를 위한 gitignore가 추가됨
- .gitignore 파일 다운받기: https://github.com/github/gitignore
- 폴더의 루트에 두고 “ .gitignore ” 으로 이름 변경

실전 프로젝트

4. 프로젝트 파일 생성 및 initial commit

- git에서 clone한 폴더에 프로젝트 만들고 initial commit 생성
- 유니티, 언리얼 등



실전 프로젝트

5. branch 전략에 따라 프로젝트 시작하기

- branch 생성해서 프로젝트 시작하기
- branch 전략(Git Flow, Github Flow) 등을 따라도 됨

(자세한 건 3차시에서 다루겠습니다.)

과제 및 공지

2차시 과제는 없습니다.

다만 4차시부터는 과제를 git을 통해 제출하셔야 하기 때문에 2차시 내용을 반드시

숙지하셔야 합니다.

Git과 Fork 설치는 본격적인 과제 제출 시작 전에 해보시고 문제 발생 시 스터디

운영진에게 문의 바랍니다.

결석 구제 및 과제 제출에 대한 공지는 4차시 진행 시점에 정리해서 올려드리겠습니다.