



17기 DE 표선영

Contents

- VCS | Git | GitHub
- Three states & Git commands
- 실습 - git setup & GitHub repository 생성

VCS | Git | GitHub

VCS

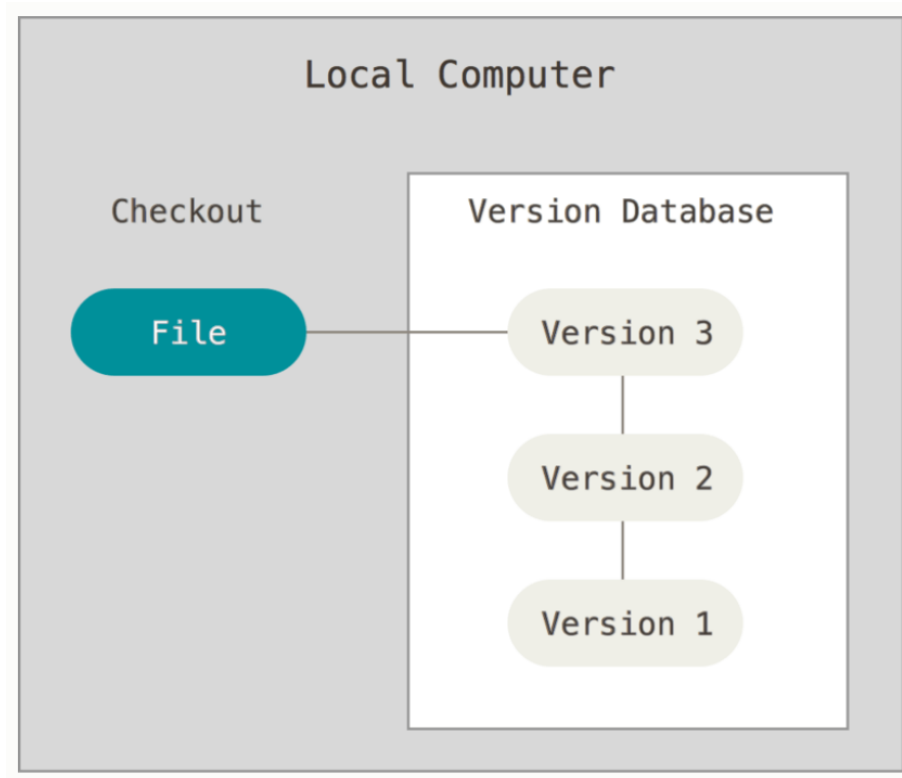
- Version Control System (버전관리시스템)
- 버전관리 = 파일/소스코드의 “Version”들을 기록하는 행위
- 사용 x시? 매 버전을 백업하거나, 협업 시 코드를 합치는 것이 복잡

문서집계표(최종).HWP
문서집계표(최종수정).HWP
문서집계표(최종수정컨펌).HWP
문서집계표(컨펌V1).HWP
문서집계표(컨펌V2).HWP
문서집계표(컨펌V3).HWP
문서집계표(진짜최종).HWP
문서집계표(진짜진짜최종).HWP
문서집계표(진짜진짜진짜최종).HWP
문서집계표(회장님).HWP
문서집계표(회장님지시수정).HWP
문서집계표(회장님수정.V1).HWP.....

← File System으로 관리 시 😊

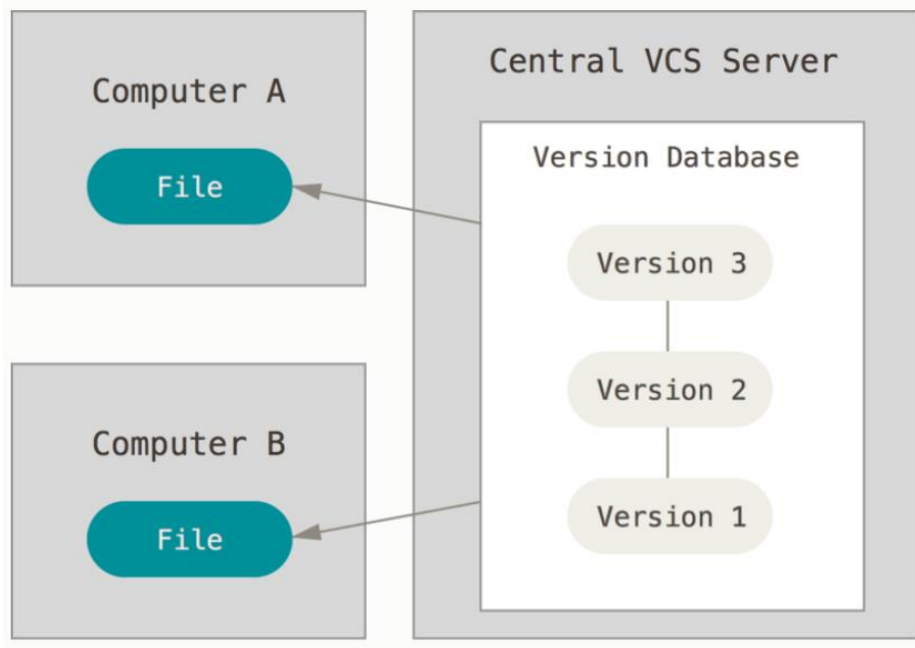
VCS를 이용한다면
버전 rollback, 변경사항 확인 쉬움

LVCS



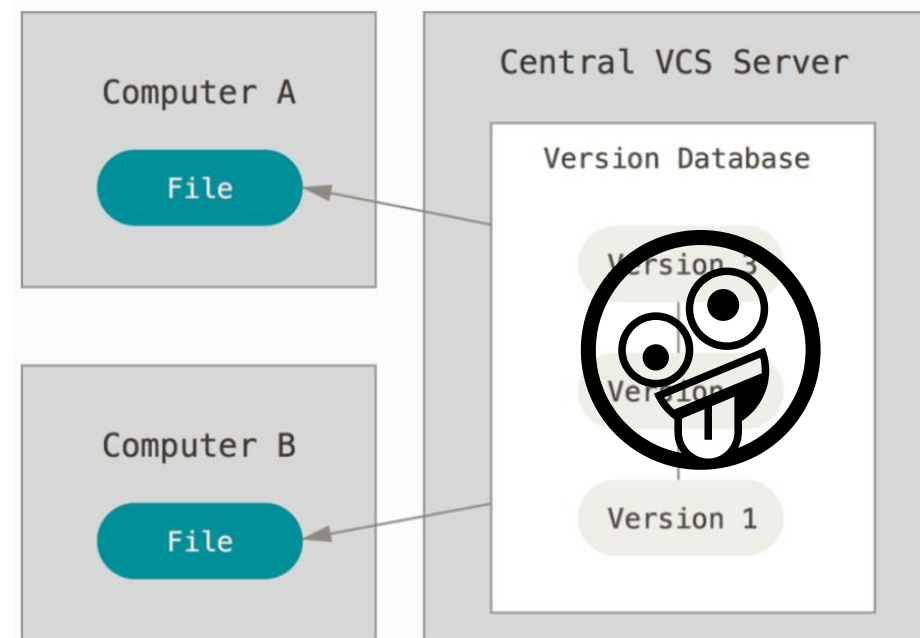
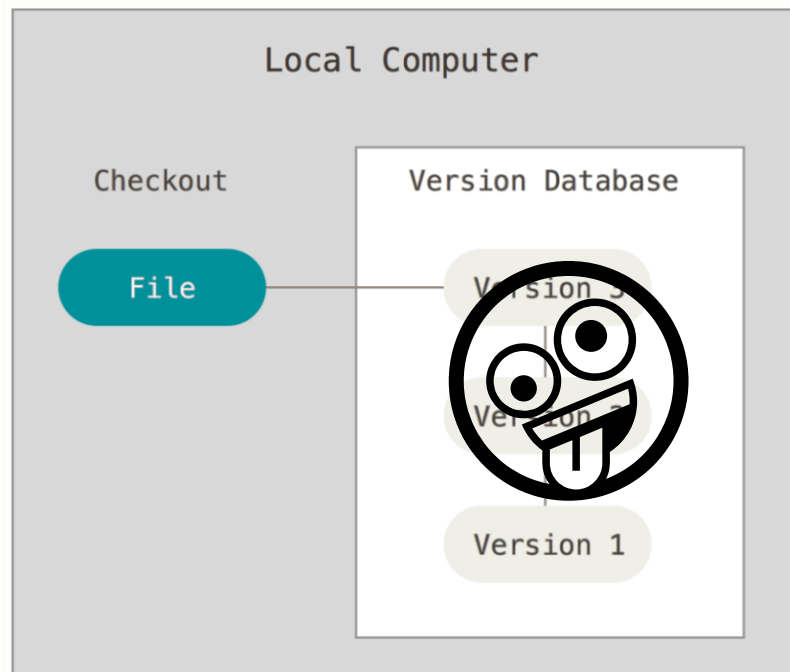
- Local Version Control System
- 로컬 컴퓨터에서 간단한 데이터베이스를 사용해 파일의 변경정보 관리

CVCS



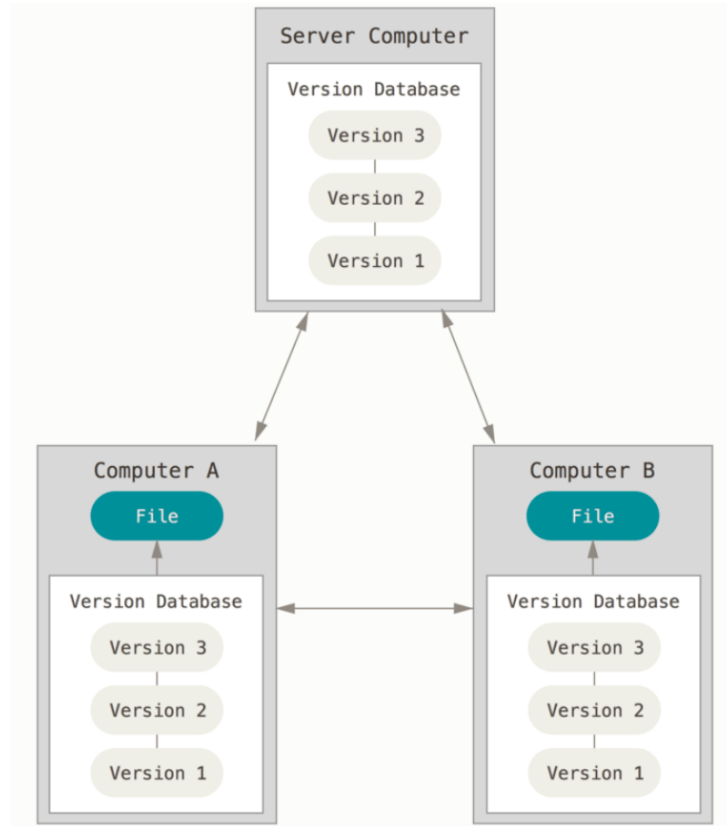
- Centralized Version Control System
 - 협업의 경우를 위해 개발됨
 - 별도의 서버가 파일을 관리
 - 클라이언트가 중앙 서버에서 파일 받아 사용, 스냅샷을 가짐
- (스냅샷 : 특정 시점에서 파일, 폴더 또는 워크 스페이스의 상태)

LVCS & CVCS



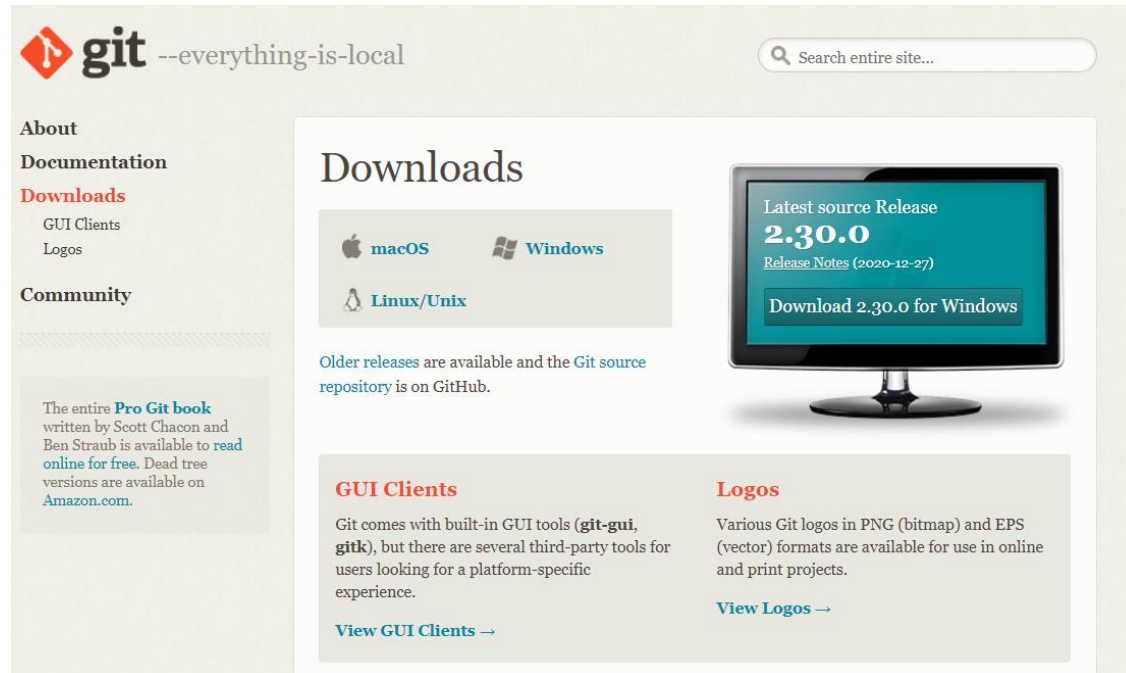
e.g. local database 문제 발생 / 중앙 서버 다운
=> 치명적 !!!! 모든 것을 잃을 수 있다

DVCS



- Distributed Version Control System
- 클라이언트가 단순히 파일 스냅샷만 가지는 것이 아니라, history까지 전부 복제해 서버와 같은 저장소를 갖게 됨
- > 서버 🤪?
- : 아무 클라이언트를 골라 서버 복원 가능
- Git은 이에 속함

GIT



- 전세계적으로 쓰이는 VCS
- Download link
- : <https://git-scm.com/downloads>

- 자료
- : <https://git-scm.com/book/ko/v2>
(한국어로 잘 나와있습니다)

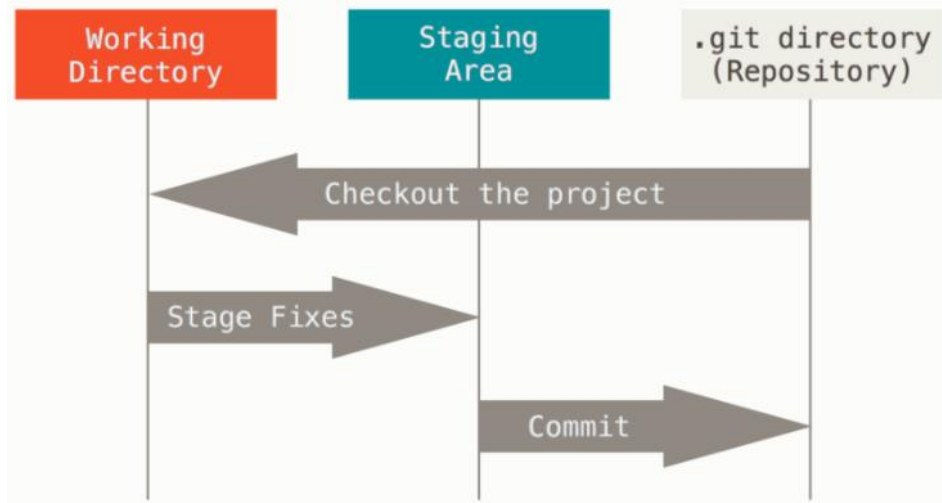
GitHub



- Git 프로젝트를 지원하는 웹호스팅 서비스
- GUI (graphic user interface)도 제공
 - 하지만 우리는 CLI 😊
- 이외에 Bitbucket, GitLab 등도 있음 !

Git Commands | Git Flow

Three States



- **Committed**
: 로컬 데이터베이스(=git repository)에 저장, commit해도 remote repository는 영향 X
- **Modified**
: 수정한 파일을 아직 로컬 데이터베이스에 커밋하지 않음
- **Staged**
: 수정한 파일을 곧 커밋할 것이라고 표시. 일종의 중간 저장 상태.

Git commands

- `git add [./* or 파일명]` -> 작업 파일을 staging area에 올림
.(dot)을 사용하면 : `.gitignore`에 있는 파일 제외 stage
- `git commit [-m “작업 내용”]`
: staging area 파일을 git local repository에 commit.
- `git push [리모트 저장소 이름] [브랜치 이름]`
: local repo -> remote repo
: e.g. `git push origin master` (master 브랜치를 origin server에 push)

Git commands

- 엄청 많아요!
- git branch
- git checkout
- git status
- git clone
- git revert
- git pull
- git fetch
- git merge
- git log
- git diff
- 등등..



GIT CHEAT SHEET

presented by Tower - the best Git client for Mac and Windows



CREATE Clone an existing repository <code>\$ git clone <url></code> Create a new local repository <code>\$ git init</code>	BRANCHES & TAGS List all existing branches <code>\$ git branch -av</code> Switch HEAD branch <code>\$ git checkout <branch></code> Create a new branch based on your current HEAD <code>\$ git branch <new-branch></code> Create a new tracking branch based on a remote branch <code>\$ git checkout -track <remote/branch></code> Delete a local branch <code>\$ git branch -d <branch></code> Mark the current commit with a tag <code>\$ git tag <tag-name></code>	MERGE & REBASE Merge <branch> into your current HEAD <code>\$ git merge <branch></code> Rebase your current HEAD onto <branch> <i>Don't rebase published commits!</i> <code>\$ git rebase <branch></code> Abort a rebase <code>\$ git rebase --abort</code> Continue a rebase after resolving conflicts <code>\$ git rebase --continue</code> Use your configured merge tool to solve conflicts <code>\$ git mergetool</code> Use your editor to manually solve conflicts and (after resolving) mark file as resolved <code>\$ git add <resolved-file></code> <code>\$ git rm <resolved-file></code>
LOCAL CHANGES Changed files in your working directory <code>\$ git status</code> Changes to tracked files <code>\$ git diff</code> Add all current changes to the next commit <code>\$ git add .</code> Add some changes in <file> to the next commit <code>\$ git add -p <file></code> Commit all local changes in tracked files <code>\$ git commit -a</code> Commit previously staged changes <code>\$ git commit</code> Change the last commit <i>Don't amend published commits!</i> <code>\$ git commit --amend</code>	UPDATE & PUBLISH List all currently configured remotes <code>\$ git remote -v</code> Show information about a remote <code>\$ git remote show <remote></code> Add new remote repository, named <remote> <code>\$ git remote add <shortname> <url></code> Download all changes from <remote>, but don't integrate into HEAD <code>\$ git fetch <remote></code> Download changes and directly merge/integrate into HEAD <code>\$ git pull <remote> <branch></code> Publish local changes on a remote <code>\$ git push <remote> <branch></code> Delete a branch on the remote <code>\$ git branch -dr <remote/branch></code> Publish your tags <code>\$ git push --tags</code>	UNDO Discard all local changes in your working directory <code>\$ git reset --hard HEAD</code> Discard local changes in a specific file <code>\$ git checkout HEAD <file></code> Revert a commit (by producing a new commit with contrary changes) <code>\$ git revert <commit></code> Reset your HEAD pointer to a previous commit ...and discard all changes since then <code>\$ git reset --hard <commit></code> ...and preserve all changes as unstaged changes <code>\$ git reset <commit></code> ...and preserve uncommitted local changes <code>\$ git reset --keep <commit></code>
COMMIT HISTORY Show all commits, starting with newest <code>\$ git log</code> Show changes over time for a specific file <code>\$ git log -p <file></code> Who changed what and when in <file> <code>\$ git blame <file></code>		

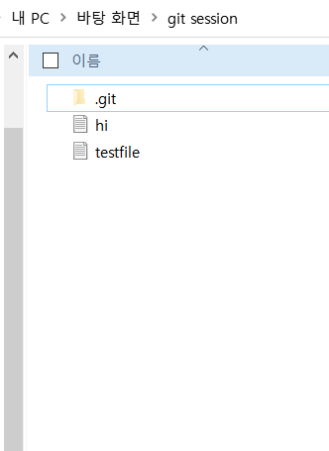
30-day free trial available at www.git-tower.com

TOWER
The best Git Client for Mac & Windows

Overall flow (Example)

현재 작업중인 장소

Workspace
(directory)



add

Staging Area
(= Index)

Tracked Files
• Staged 상태
• Modified
• Unmodified

Untracked Files
• Tracked 제외

commit

Local
Repository

Version 3
Version 2
Version 1

버전을
.git 안에 저장

Remote
add

clone

Remote
Repository

push

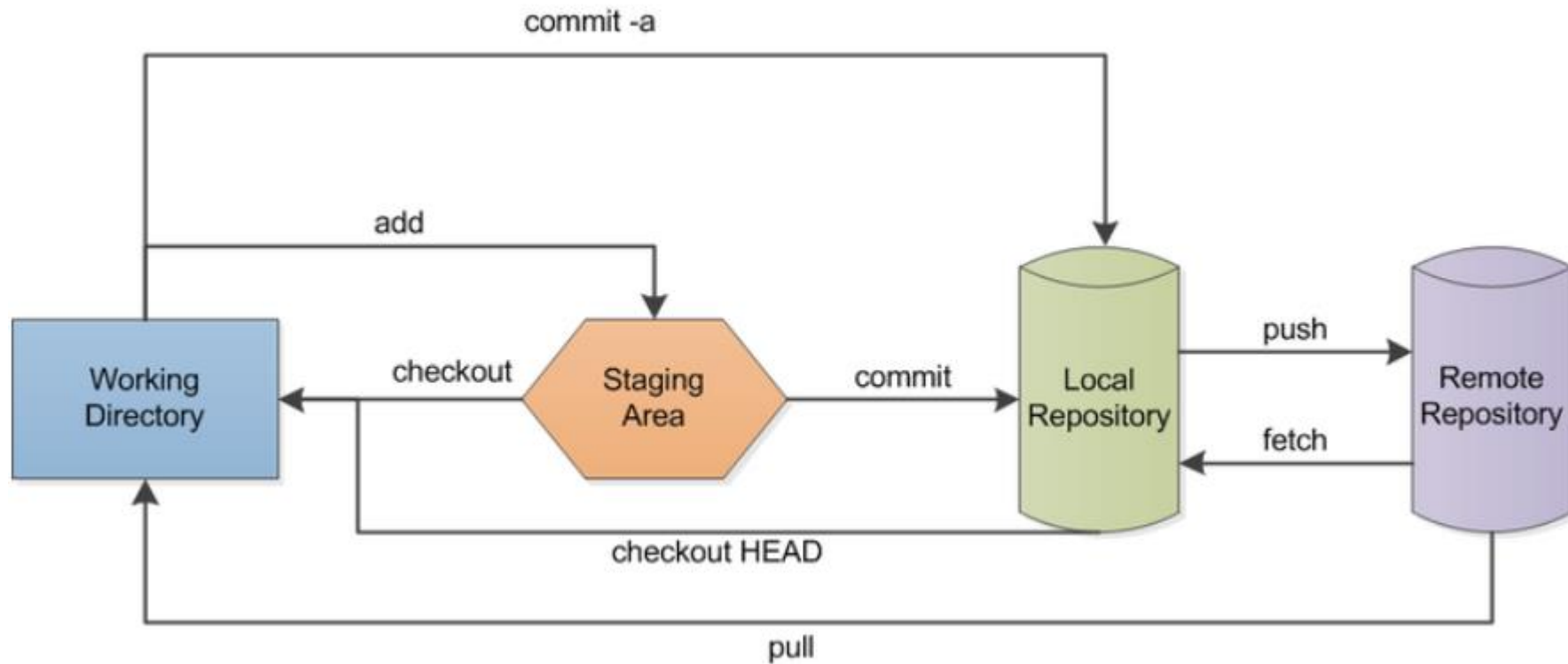
pull



GitLab

1. 로컬 저장소를 원격 저장소와 연결
2. 원격 저장소를 로컬 저장소로 가져오기

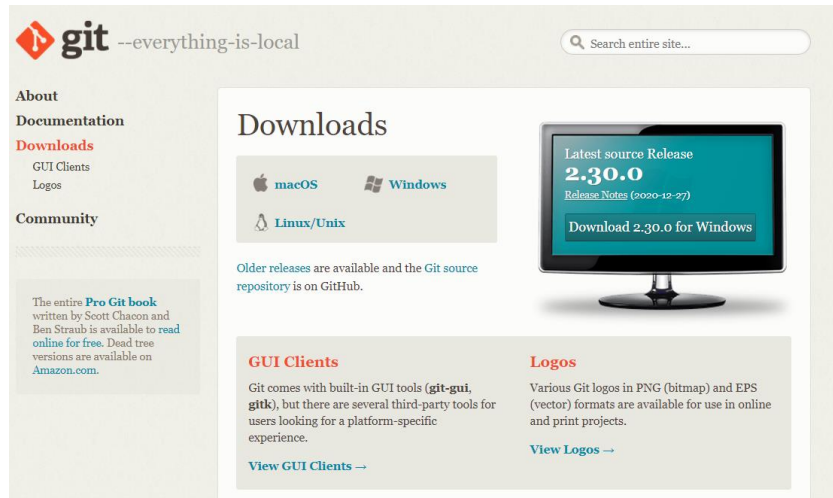
Overall flow



실습

| Git setup & GitHub Repo 생성

Git setup

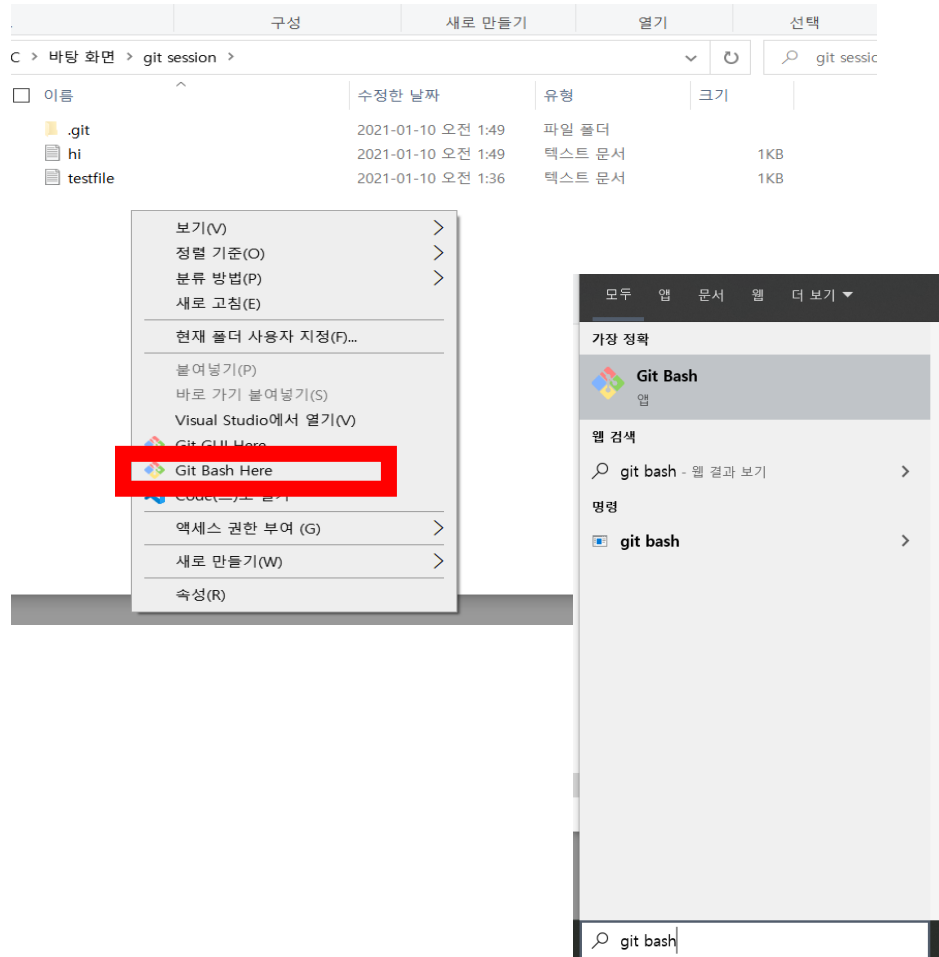


1. git 설치

<https://git-scm.com/downloads>

- 각 OS에 맞게 설치
- MacOS의 경우 `brew install -s git`으로 좀 더 편리하게 설치할 수 있음
- Windows : `git bash` / Mac : `terminal`

Git setup



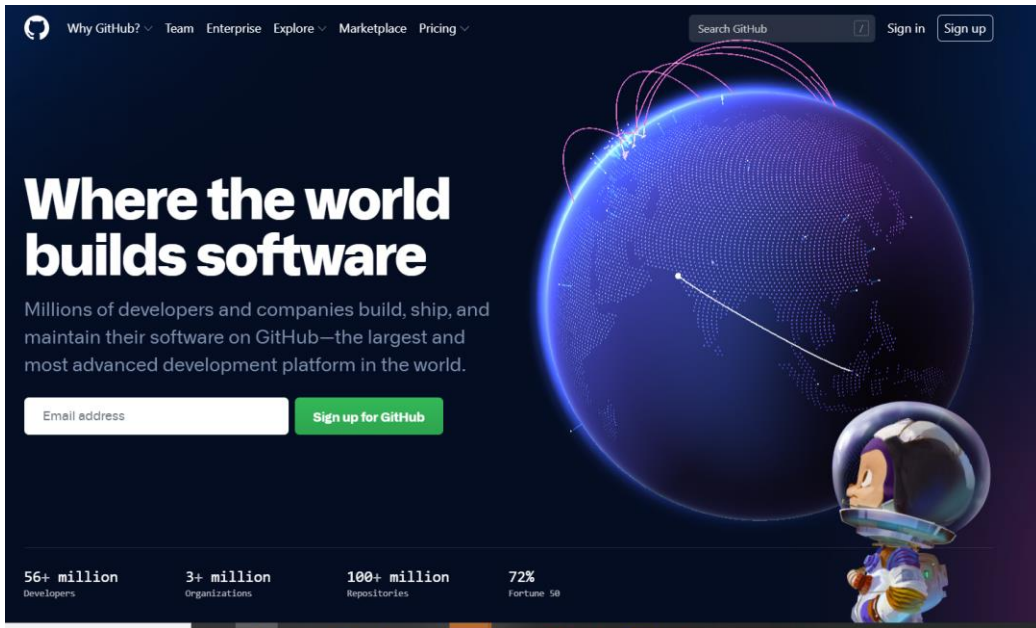
2. CLI 실행

- Git bash : 원하는 디렉토리(=git repository)에서 우클릭 or 검색창으로 실행한 다음 cd 명령어로 디렉토리 이동

```
pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop
$ git --version
git version 2.28.0.windows.1
```

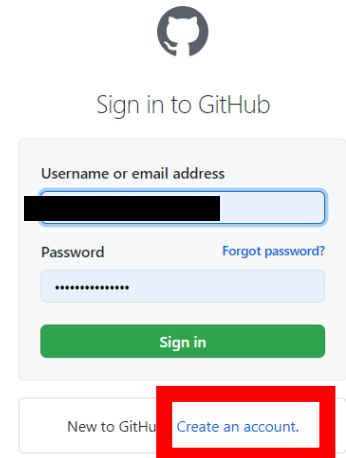
git --version으로 git이 잘 설치됐는지 확인

GitHub setup



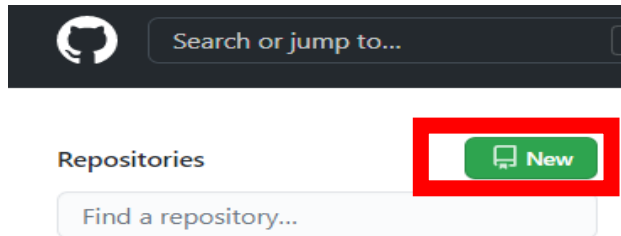
1. GitHub 가입

<https://github.com/>

A screenshot of the GitHub login and sign-up interface. At the top is the GitHub logo and the text 'Sign in to GitHub'. Below this are two input fields: 'Username or email address' and 'Password'. The password field has a 'Forgot password?' link. A green 'Sign in' button is positioned below the password field. At the bottom, there is a link 'New to GitHub?' followed by a button labeled 'Create an account.', which is highlighted with a red rectangular border.

- 회원가입 해주세요!
- 입력하라는 대로 하면 되고, 필요하신 분은 아래 링크 참고하시면 될 것 같습니다.
(<https://goddaehee.tistory.com/218>)

GitHub setup



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * / Repository name *

Great repository names are short and memorable. Need inspiration? How about [studious-journey?](#)

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

2. Repository 생성

- 로그인 후 좌측 상단에서 New 버튼 클릭 (or **Create New Repository** 버튼)
- 빈칸 적절하게 채우고 create

remote & local 연결

1. git repository 설정

- git init : 원하는 디렉토리에서

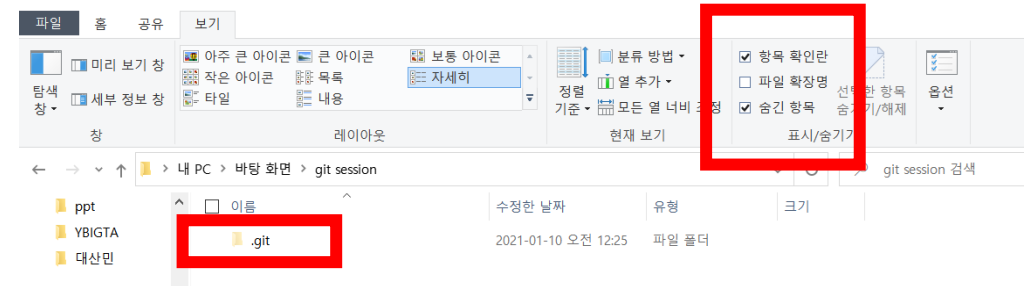
- 숨긴 파일 표시

-> .git이라는 폴더 생성된 것 확인 가능

- 해당 경로를 git이 관리하도록 초기화

git init후 (master) 표시 확인해주세요.
git init 전에는 표시 없습니다.

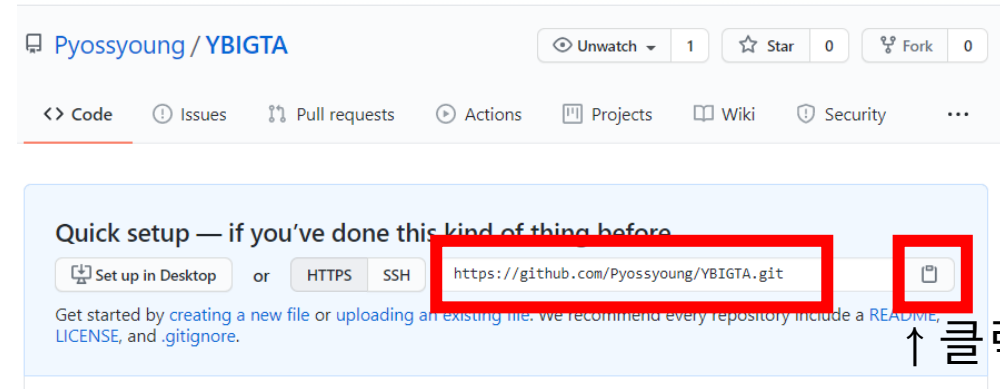
```
pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)  
$ git init  
Reinitialized existing Git repository in C:/Users/pyosyoung/Desktop/git session/.git/
```



remote & local 연결

2. remote repo와 연결

- GitHub에서 repo 경로 확인



- `git remote add origin [remote repository 경로]`
: local 레포와 remote repo를 연결해주는 명령어
: 확인한 GitHub repo 경로 사용

참고 : Windows에서는 GitBash 창에서 Ctrl+C/V로 복붙 안됨

-> Ctrl+Insert / Shift+Insert 사용

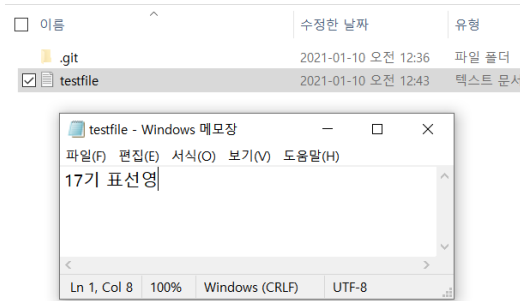
```
pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)
$ git remote add origin https://github.com/Pyosyoung/YBIGTA.git

pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)
$ |
```

add, commit, push

로컬 파일 수정, 저장 후

1. git add . (또는 git add [파일명] e.g. git add testfile.txt) : stage



```
pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)
$ git add .

pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   testfile.txt
```

✓ git status : 파일들의 상태를 확인하는 명령어

add, commit, push

로컬 파일 수정, 저장 후

2. git commit -m “커밋 메시지”

```
pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)
$ git commit -m "added my name to testfile.txt"
[master (root-commit) e53b2f0] added my name to testfile.txt
1 file changed, 1 insertion(+)
create mode 100644 testfile.txt

pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)
$ git status
On branch master
nothing to commit, working tree clean
```

c.f. git status

```
pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   testfile.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hi.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

- hi.txt : 새로 만들고 add 전
- testfile.txt : commit 이후에 파일 수정

add, commit, push

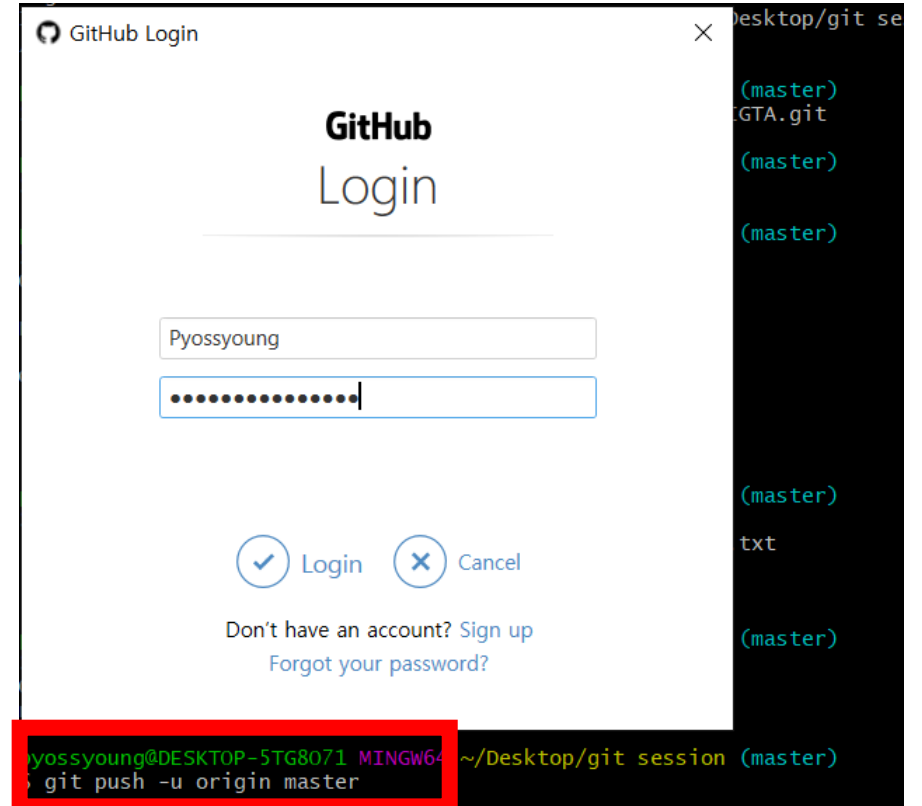
로컬 파일 수정, 저장 후

3. git push -u origin master

: 로컬 저장소를 원격 저장소 master branch에 반영

```
pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)
$ git push -u origin master
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 244 bytes | 81.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Pyosyoung/YBIGTA.git
[New branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'
```

↑ 로컬 master branch가
remote 서버 origin의 master branch를 추적중이다.

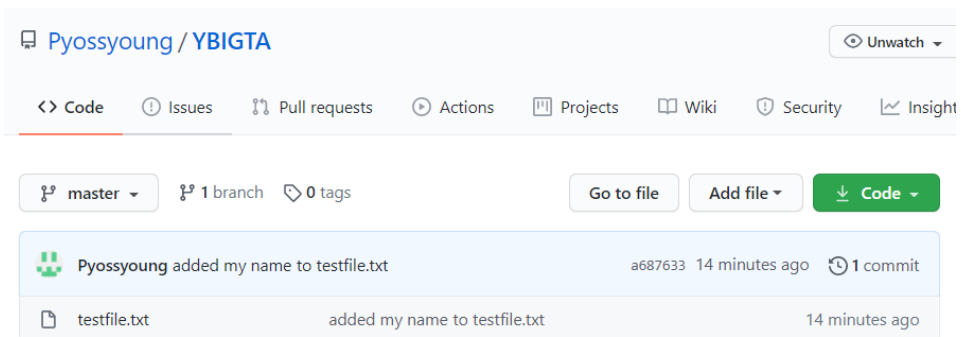


TIP

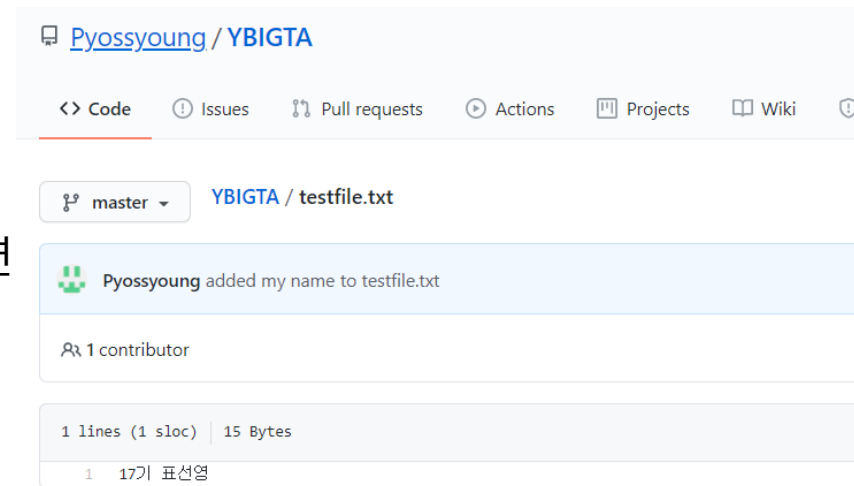
: git config로 사용자 설정해두면 커밋마다 로그인 정보 입력 필요 X

add, commit, push

GitHub repo를 새로 로드해 확인해보면 업로드된 것을 확인할 수 있음



→
파일 눌러서 확인하면
내용도 보임



과제

과제

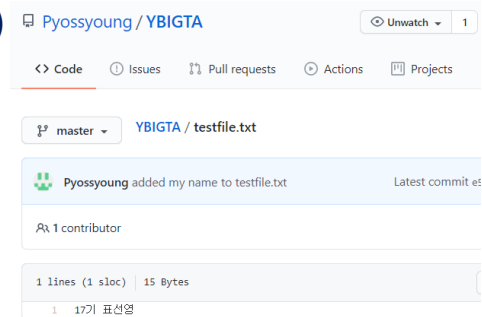
- Github 계정 생성 후 원격 저장소 생성, local repo와 연결
- 18기 [이름] 라고 적힌 txt파일 만들기 (파일명 무관)
- 깃헙 repo에 push하기 (add -> commit -> push)
- 1) git push 명령어 결과 캡처, 드라이브 업로드
(파일명 : push_이름)
- 2) git repo 캡처, 드라이브 업로드
(파일명 : repo_이름)
- 3) 깃헙 repo 구글 스프레드 시트에 올리기

- 앞으로의 과제는 구글 드라이브가 아닌, 각자 만드신 깃헙 레포에!
그러니 이번 과제 시 앞으로 과제를 올리실 레포를 만들어주세요 😊
- 이미지 파일들은 zip으로 묶지 마시고 이미지 각각 올려주세요!

1)

```
pyosyoung@DESKTOP-5TG8071 MINGW64 ~/Desktop/git session (master)
$ git push -u origin master
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 244 bytes | 81.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Pyosyoung/YBIGTA.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

2)





감사합니다.