Web Studio 2019
1. Git, Github

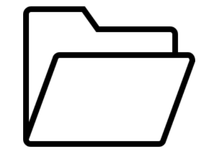
김상근 / sisobus1@gmail.com

Contents

버전관리의 필요성

숙제 할 때

1. 중간중간 수정사항이 생겨서 수정을 많이함
2. 갑자기 뻑남
3. 옛날 코드로 돌아가고 싶은데 …? ctrl+z로는 더 안돌아가네 …



HomeWork_5

HomeWork_1.py

HomeWork_2.py

버전관리의 필요성

파일 분리

1. 일단 파일을 분리해보자
2. 수정 -> 수정1 -> 완성 -> 진짜완성 -> 완성!!!
3. 제출은?

HomeWork_5

HomeWork_1.py

HomeWork_1_수정.py

HomeWork_1_수정1.py

HomeWork_1_완성.py

HomeWork_1_진짜완성ㅗ.py

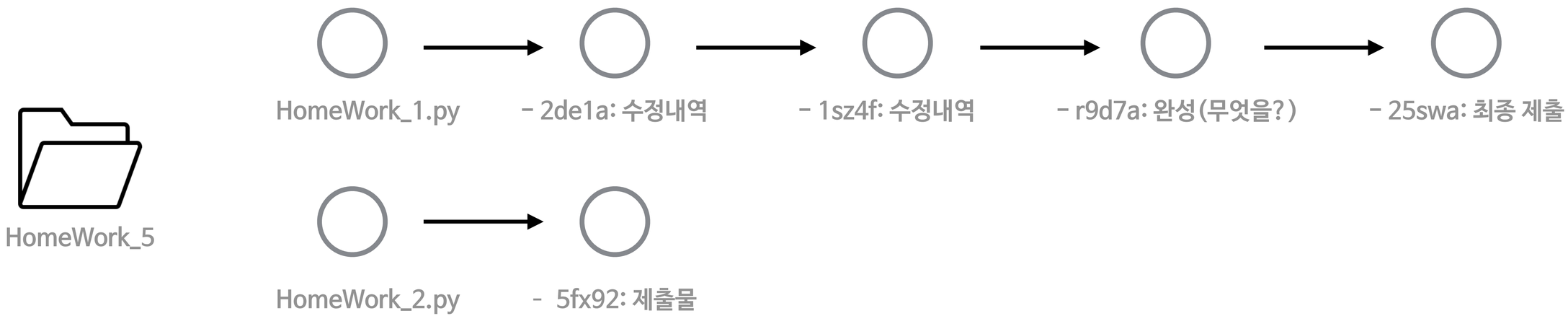
HomeWork_2.py

HomeWork_2_수정.py

Git

SCM (Source Control Management)

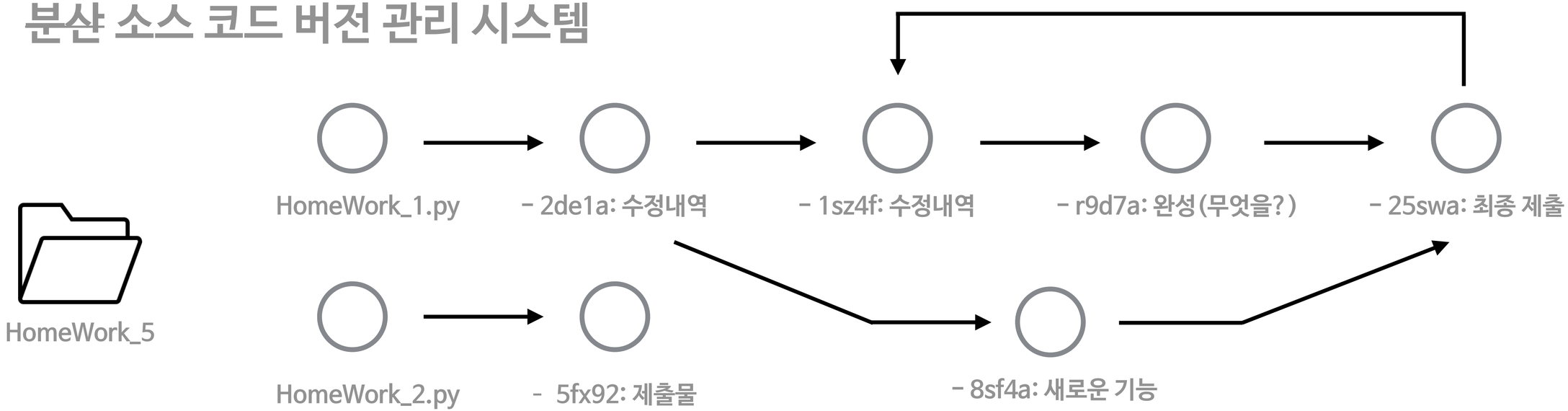
1. VCS (Version Control System)라고도 함
   - 파일 변화를 기록해두었다가 나중에 특정 기록을 불러올 수 있는 시스템
2. 분샨 소스 코드 버전 관리 시스템



Git

SCM (Source Control Management)

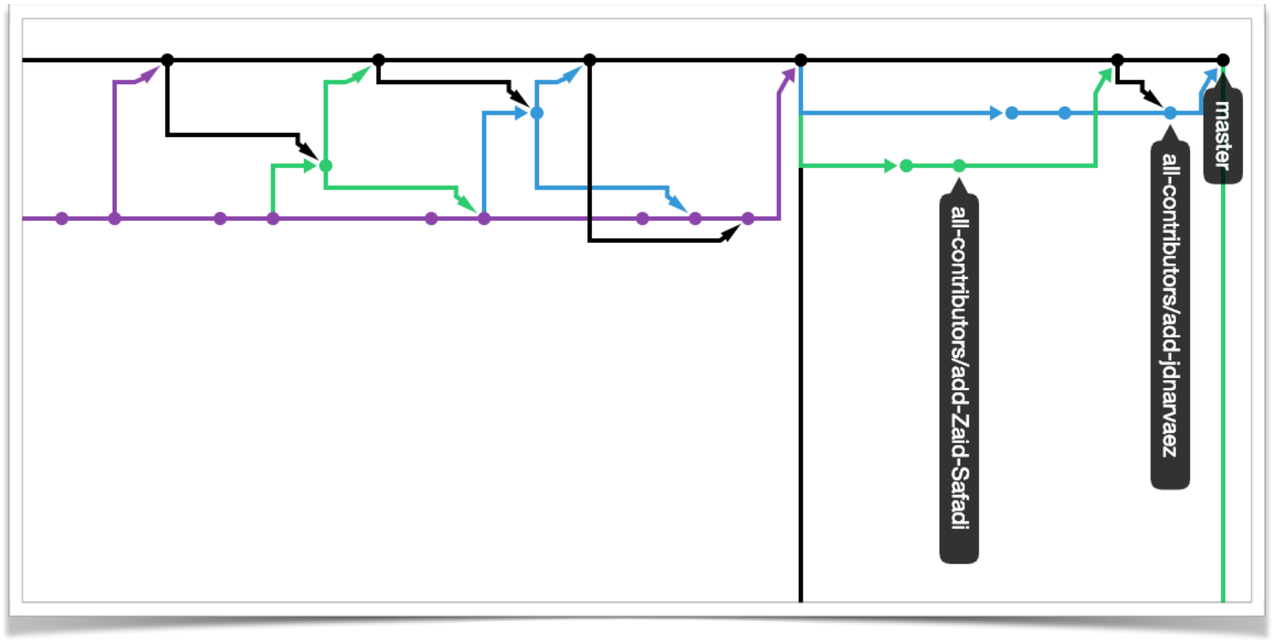
1. VCS (Version Control System)라고도 함
   - 파일 변화를 기록해두었다가 나중에 특정 기록을 불러올 수 있는 시스템
2. 분산 소스 코드 버전 관리 시스템



Git

좋은 버전관리의 예

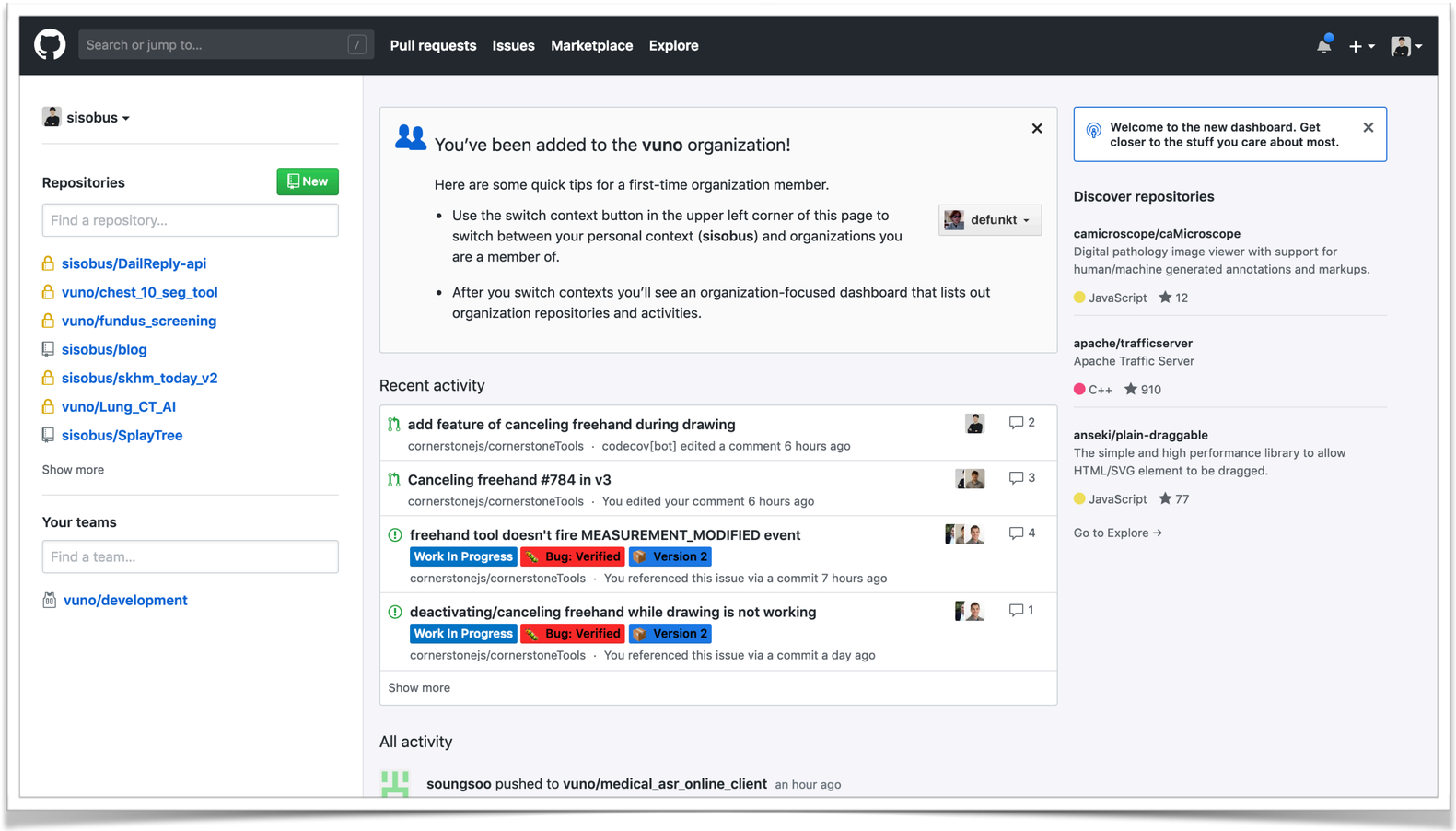
1. Feature마다 branch를 둔다.
2. Commit은 자주



Github

Git을 쉽게 이용하기 위한 웹 기반 Tool

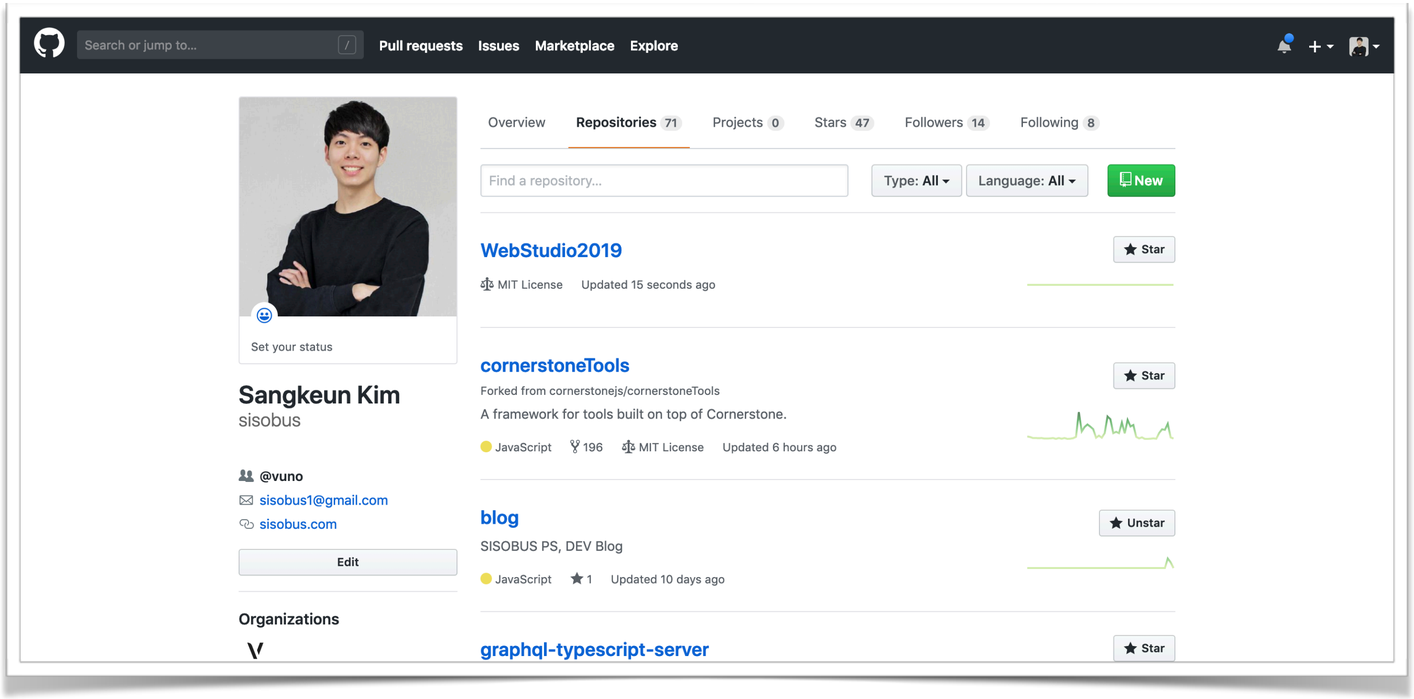
1. https://github.com
2. 저장소 관리를 쉽게 해줌
3. issue 관리 (Issues)



Github

Repository

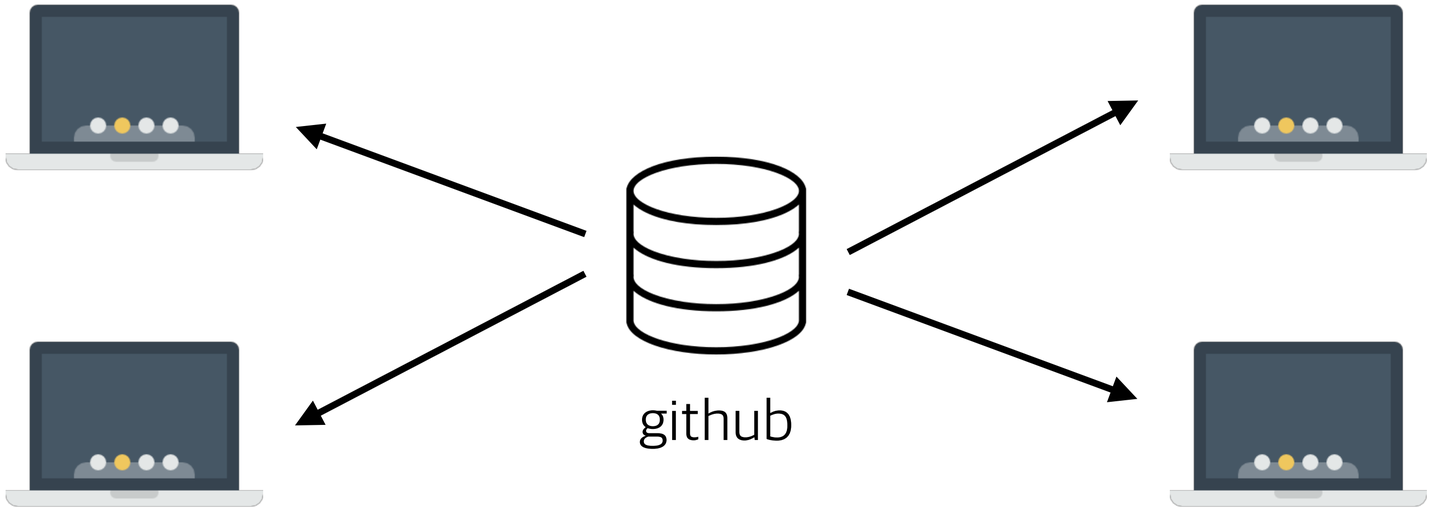
1. 버전 관리가 될 공간
2. Github라는 회사 서버의 한 공간을 빌리는 것(무려 free)
3. 이 저장소를 Clone하여 Local repository를 만들어야함



Github

Clone

1. Remote repository를 Local repository로 다운로드 받는 것
2. 실시간 동기화가 아님
3. 여러 곳에 clone이 가능(분산 버전관리 = git)




```
$ git clone https://github.com/sisobus/WebStudio2019.git
```

Github

Git Workflow

1. Add - git 작업을 할 파일을 고르는 작업
2. Commit - 변경 내용을 기록하는 작업
3. Push - remote repository에 commit 내용을 반영하는 작업

Github

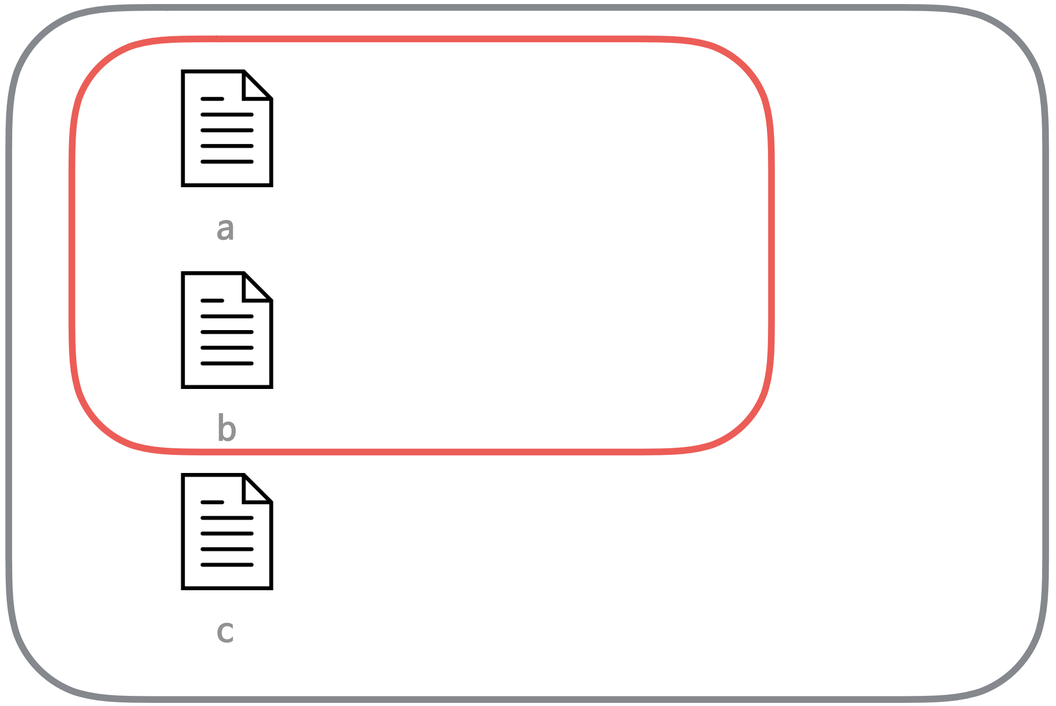
Add

1. 파일을 git 준비영역(staging area)에 등록함
2. 전부 추가할 때에는 $ git add .

```
$ mkdir test
$ cd test
$ touch a
$ touch b
$ touch c
$ git add a
$ git add b
$ git status
```

```
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   a
        new file:   b

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

        ../add.sh
        ../clone.sh
        c
```

test



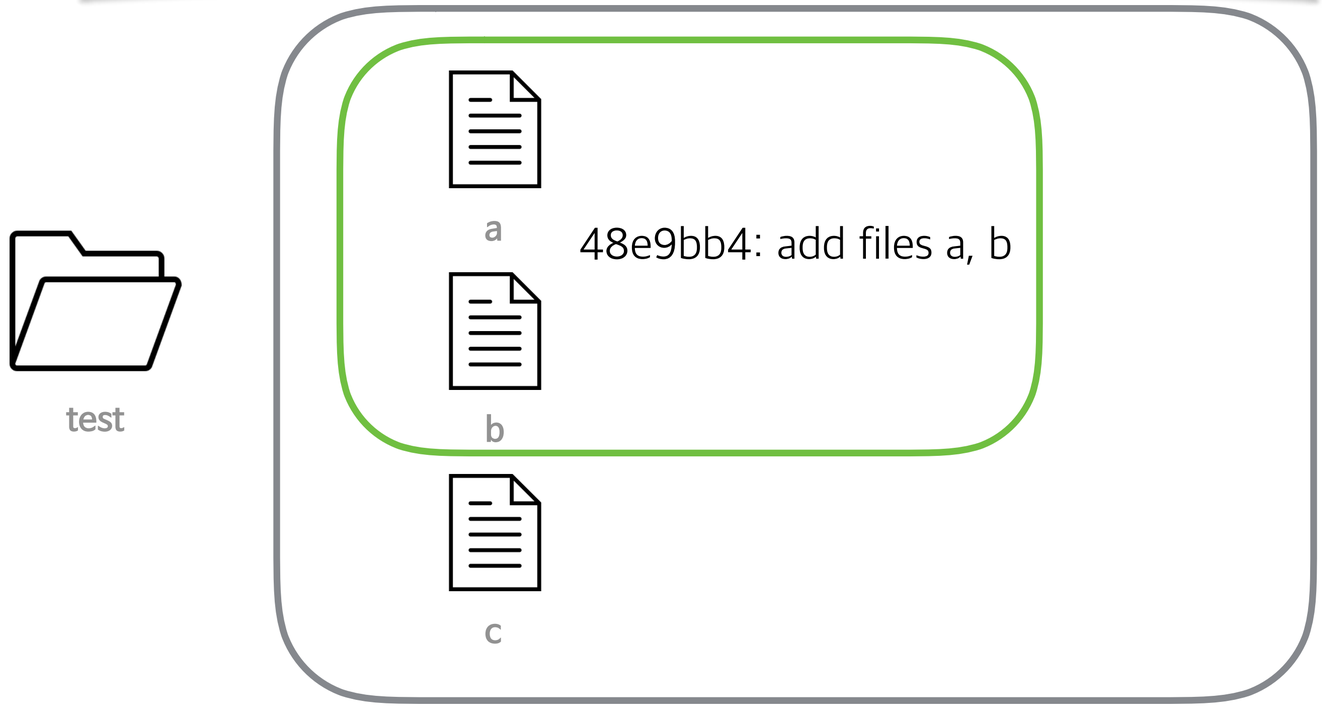
Github

Commit

1. 변경 사항을 기록함
2. 고유의 주소를 가짐 (48e9bb4)

```
$ git commit -m "add files a, b"
$ git status
```

```
[master 48e9bb4] add files a, b
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1_git/source/test/a
 create mode 100644 1_git/source/test/b
sisobus-ui-MacBook-Pro:source sisobus$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

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

        add.sh
        clone.sh
        commit.sh
        test/c

nothing added to commit but untracked files present (use "git add" to track)
```



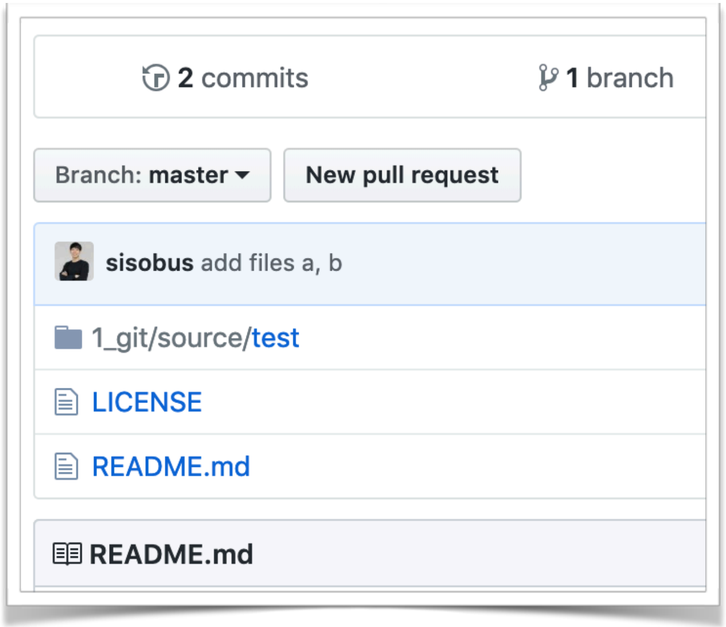
Github

Push

1. Remote repository (github)에 Local repository commit 내용을 반영함

```
$ git push origin master
$ git status
```

```
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 428 bytes | 428.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/sisobus/WebStudio2019.git
   aa5219e..48e9bb4  master -> master
```



Github

다시 한번

1. Remote repository (Github)에 이상한 것들 다 지워봅시다.

```
$ rm -rf test
$ git add .
$ git commit -m "clean all test files"
$ git push origin master
```

Github

Checkout

1. 과거 버전(commit)으로 돌아가는 기능
2. 먼저 Commit log를 봅시다. ($ git log)
3. 맨 처음 버전으로 돌아가볼까요? ($ git checkout aa521)
4. 다시 현재로 돌아옵시다. ($ git checkout master)

```
commit ac6c5b63ba281fdca68f3c0a6700b353c78688ff (HEAD -> master, origin/master, origin/HEAD)
Author: sisobus <sisobus1@gmail.com>
Date:   Wed Mar 6 18:38:51 2019 +0900

    clean all test files

commit 48e9bb455b6583712f7638afb756382a01366081
Author: sisobus <sisobus1@gmail.com>
Date:   Wed Mar 6 18:05:24 2019 +0900

    add files a, b

commit aa5219e9b890dffe45e2c625677bb958f40a8098
Author: Sangkeun Kim <sisobus1@gmail.com>
Date:   Wed Mar 6 17:35:48 2019 +0900

    Initial commit
```

Github

Pull

1. 쉽게 말하면 push의 반대라고 보면 됨
2. 여러 컴퓨터로 clone해서 작업하다 보면 당연히 싱크가 안맞음
3. Remote repository의 내용을 예전에 clone 뜬 local repository에 다운받는 방법
4. $ git pull origin master

Github

Fork

1. 남의 Repository를 다루기 위한 Github 기능
2. 남의 Repository를 그대로 가져와 본인의 Repository로 복제됨
3. 본인의 Repository를 clone해서 작업한 후, 본인의 repository로 push
4. 당연히 남의 Repository는 변함이 없음

Github

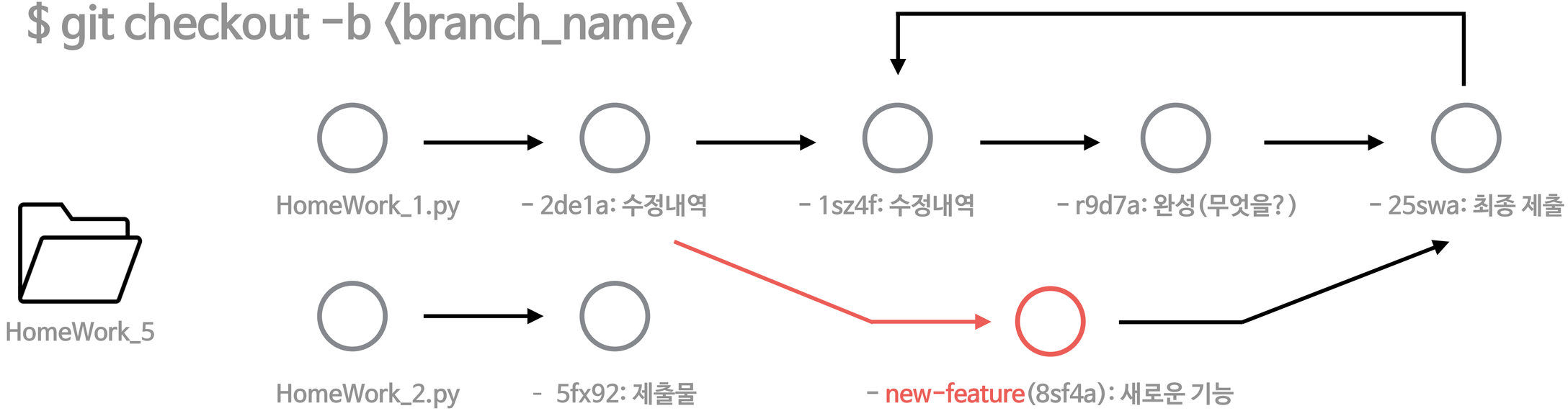
Pull Request

1. fork뜬 repository를 원본 repository에 반영해주세요 라고 제안하는 것
2. 실제로 반영하는 것은 원본 repository의 관리자가 하게 됨
3. 직접 해봐야 감이 옵니다. 실습 고고

Github

branch

1. 우리가 작업하고 있는 것은 master branch
2. 새로운걸 해보고 싶거나 할 때, 혹은 기능별로 branch를 파서 작업하는게 좋음
3. $ git checkout -b <branch_name>



실습

자기소개하는 글 작성하기

1. WebStudio2019 repository fork
2. 본인 repository clone
3. 새로운 브랜치 만들기 $ git checkout -b 1-practice-〈본인영어이름〉
4. mkdir 1_git/practice/〈본인영어이름〉
5. 1_git/practice/〈본인영어이름〉/README.md 를 작성
6. $ git add .
7. $ git commit -m “add 〈본인영어이름〉”
8. $ git push origin 1-practice-〈본인영어이름〉
9. 본인 repository를 새로고침하면 웬 버튼하나가 나와있음
10. 그걸 눌러서 pull request
11. 제가 승인

유용한 Git 명령어

Pull 할 때 Conflict가 납니다.

1. $ git stash
2. $ git pull origin master
3. $ git stash pop
4. 충돌난 파일에 가보면 어떤걸 쓸 것인지 아래와 같이 나옵니다.
5. 필요한 부분만 남기고 지웁니다.

```
=========HEAD
~~
>>>>>>>>>ad832dwq1
~~
=========
```

Reference

참고하세요

1. https://rogerdudler.github.io/git-guide/index.ko.html

과제

미정

1. git checkout -b 1-homework-〈본인영어이름〉
2. mkdir 1_git/homework/〈본인영어이름〉
3. 1_git/homework/〈본인영어이름〉에 아무거나해서 pull requests를 날려봅시다.
4. 제가 Accept하면 과제 끝 Reject면 실패

Q & A

Appendix

Bash 기본 명령어

1. ls : 현재 위치의 파일 리스트
   - ls -al : 숨겨진 파일까지 전부 보기
2. cd : 디렉토리 이동
   - .은 현재디렉토리, ..은 바로 전 디렉토리를 의미함
   - 예를들어 cd .. 이 명령어는 바로 전 디렉토리로 이동
3. rm : 파일 지우기
   - rm -rf : 디렉토리도 지울 수 있음
4. vi <filename> : vim 에디터를 이용한 파일 작성하기
5. mkdir <directoryname> : 디렉토리(폴더) 만들기
6. pwd : 현재 위치 출력하기

Appendix

Vim

1. 최고의 Text editor
2. 이것만 잘써도 코딩 생산성이 상당히 많이 올라감
3. 대부분의 IDE에는 Vim Plugin이 존재함
4. Vim 쓰세요
5. 두번쓰세요
6. 세번쓰세요
7. 평생쓰세요

Appendix

Vim

1. 명령어 모드와 에디터 모드로 나뉨
2. 맨처음 들어갈 때에는 명령어 모드임
3. 에디터 모드에서 명령어 모드로 바꾸는 것은 esc로 함
4. 명령어 모드에서 에디터 모드로 바꾸는 것은 i, o, a로 함 (i만 알아도 상관 없습니다.)

Appendix

Vim 명령어 모드

1. i는 현재 커서에서 에디터모드로 전환
2. yy는 현재 줄 복사
3. p는 복사한 것을 붙여넣기
4. dd는 현재줄 삭제
5. u는 undo
6. ctrl+r는 redo
7. :wq 는 저장후 종료를 의미함
8. /〈search string〉 으로 검색할 수 있음
9. :%s/〈stringA〉/〈stringB〉/g 는 stringA를 stringB로 대치함
   - :%s/^/₩/₩//g : 맨 앞에 //를 붙임
   - :3,12s/ha/hi/g : 3번줄부터 12번째줄의 ha를 hi로 바꿈