Learning Git

Git,

(분산)버전관리 시스템

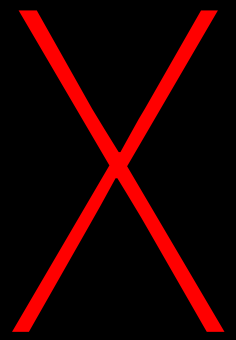
코드의 history를 관리하는 도구

개발된 과정과 역사를 볼 수 있고, 특정 시점으로 복구가 가능

OX퀴즈

git을 기반으로 한 서비스는
Github이 유일하다

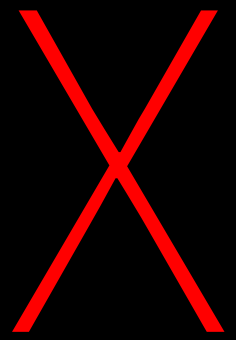
OX퀴즈



OX퀴즈

git == Github

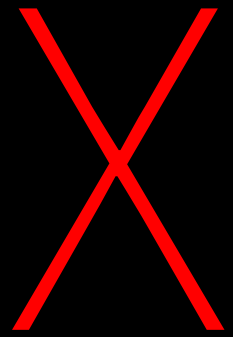
OX퀴즈



OX퀴즈

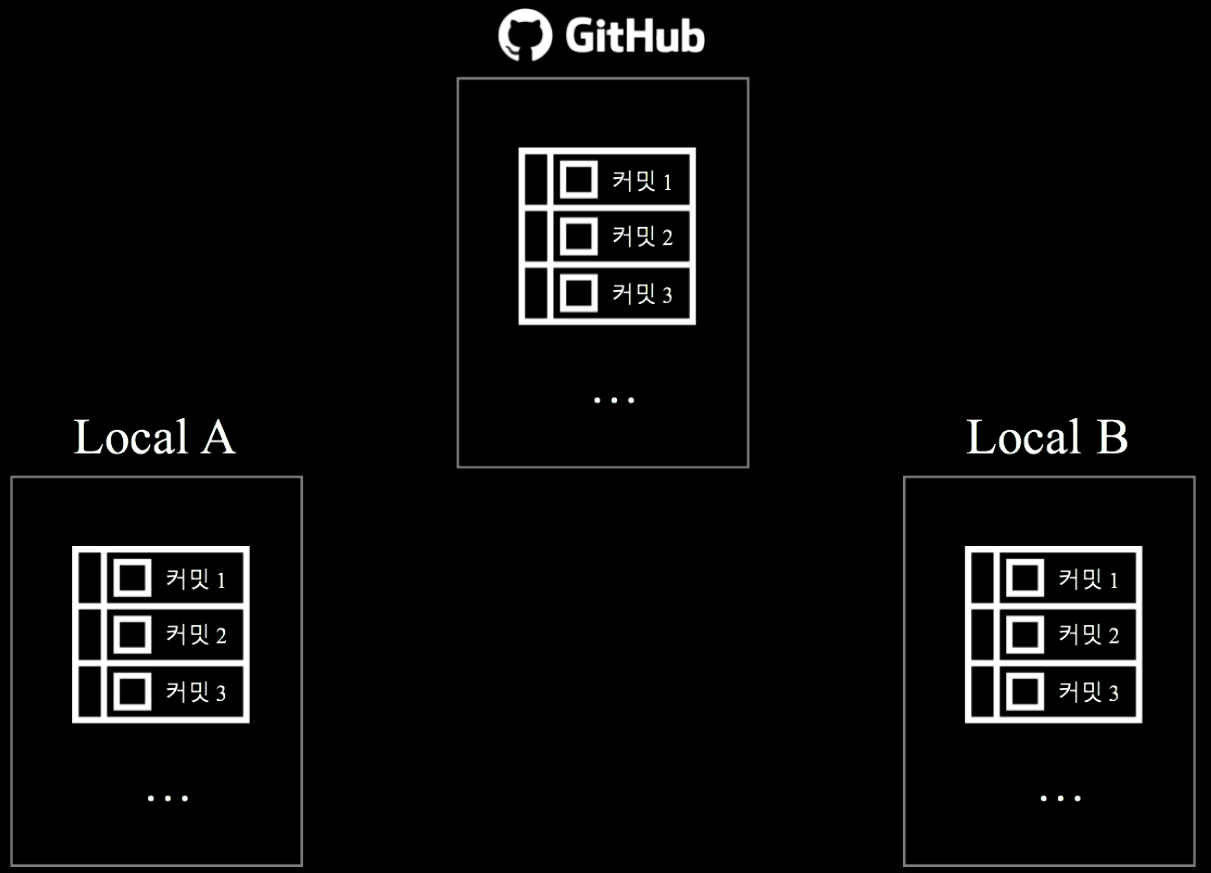
git 은 터미널에서 명령어만으로
작업할 수 있다.

OX퀴즈



But, we use git with CLI

Git, DVCS(Distributed Version Control System)



Git이 없었을 때의 우리의 작업환경

그 사이에 뭐가 바뀌었는지
차이(diff)를 알 수 없다

ctrl c, ctrl v의 반복

과제_v1_최종.zip
과제_v1_최종_수정.zip
과제_v1_최종_수정_2차.zip
과제_v2_파이널.zip
과제_v2_진짜_파이널.zip
…

Git이 있다면

차이(diff)가 무엇이고
수정 이유를 log로 남길 수 있다

뼈대 틀 구성
메인 기능 구현
로그인 기능 구현
채팅 기능 구현
디자인 입힘
…

현재 파일들은 안전한 상태로
과거 상태 그대로 복원 가능(반대도 마찬가지)

각 버전별로 차이점만 저장해서 사이즈 감소

git 기본

add 커밋할 목록에 추가
commit 커밋 (히스토리의 한 단위) 만들기
push 현재까지의 역사 (commits) 가 기록되어 있는 곳에 새로 생성한 커밋들 반영하기

git 기본

```
# git add README.md
```

git의 sub-command중 하나

arguments (1개)

```
# git config --global user.name “jiyun”
```

--로 시작하면 보통 long name 옵션

arguments (2개)

```
# git commit -s
```

-로 시작하면 보통 short name 옵션

git 기본

터미널 창 실행 후, 미리 설정되어있을지 모를 계정 정보 삭제 (처음 설치시 생략 가능)

```
# git config --global --unset credential.helper
```

```
# git config --system --unset credential.helper
```

나의 github 계정 이메일과 본인의 영문이름으로 계정 정보 등록

```
# git config --global user.email “내이메일@적어요”
```

```
# git config --global user.name “내이름 적어요”
```

git training

stage 0: initialize the local repository

git init 로컬 저장소 만들기

git training

stage 1: basic routine

git status	파일의 상태 확인
git show HEAD	가장 마지막 커밋을 조회
git log	저장소의 커밋 히스토리를 시간순으로 보여줌

git training
stage 2: check differences

git diff	파일의 변경사항을 보여줌
git log -p	각 커밋의 diff 결과를 보여줌
git log -p -숫자	가장 최근 숫자개의 커밋 diff를 보여줌

git training

stage 3: bring the repository from the remote

git clone *url*	로컬에 해당 리포의 파일들을 가져옴
git pull *origin master*	현재 로컬의 파일들을 최신상태로 업데이트함
git fetch *origin master*	현재 remote의 파일을 확인만 함 (FETCH_HEAD)

git training
stage 4: undo

git commit --amend	가장 최근의 커밋 메세지를 수정
git reset HEAD 파일명	가장 최근 커밋에서 해당 파일을 add 취소

git training
stage 5: versioning

git checkout HEAD~1	하나 직전 커밋으로 돌아감
git checkout 커밋해시코드6자리	해당 커밋으로 돌아감

git training - advanced

stage 6: stash

git stash	stash 하기(커밋하지 않고 임시보관)
git stash list	stash list 확인
git stash apply (--index)	stash list 불러오기 (-- staged 상태까지 복원)
git stash drop	stash list에서 삭제
git stash pop	stash 내용을 apply+drop

git training - advanced

stage 7: branch

git branch *branch*	*branch* 생성
git checkout *branch*	해당 *branch*로 이동
git checkout –b branch	*branch* 생성 및 이동
git checkout –d branch	*branch* 삭제

git training - advanced

stage 8: merge

git merge *branch*	해당 *branch*를 현재 브랜치로 병합
git checkout *branch*	해당 *branch*로 이동

git training - advanced

stage 9: rebase

git rebase *branch*	해당 *branch*로 rebase
git rebase *branch* --continue	conflict 발생시

git for advanced user

merge, fetch, rebase, stash, blame, cherry-pick, submodule …

https://git-scm.com/book/ko/v1/