

Git 로컬 저장소 생성하기

로컬 저장소란? (아직 터미널에 저장된 파일들이 있는곳)

버전을 관리하는 로컬 저장소 (작업파일에서 만들어진 버전 버전을 관리하는 저장소)

원격 저장소란? (github나 다른 현업 사이트에 등록된 파일들이 있는 곳)

내가 태영님께 fork했다면 태영님 저장소가 원격저장소가 된다.

git Workflow

`working directory`란

파일이 어느정도 준비가 되어 있다면 **staging area**로 옮겨놓음 (add 전 단계)

`staging area`란

어느정도 작업 하다가 버전 히스토리에 저장할 준비가 되어있는 파일들을 옮겨 놓는곳 (add로 후 단계)

`git directory`란

git directory는 이제 push를 이용해서 github에 올릴수있는 공간을 말함

commit이라는 명령어를 이용해서 staging area에 있는 파일들을 git directory에 저장

 git Workflow0

 git Workflow1

 git Workflow2

 git Workflow3

git init (git폴더로 사용)

폴더를 git폴더로 사용하겠다 라는 명령어

 git init

push = github에 올리기

push라는 명령어 github에 업로드 가능 (GitHub에 올리기(현업의시작))에서 개념 나눔

pull = github에 있는 파일 내컴퓨터에 저장하기

pull이라는 명령어를 이용해서 github에 있는 파일들을 다시 내 컴퓨터로 저장할수 있다.
(GitHub에 올리기(현업의시작))에서 개념 나눔

tracking files (Git에서 읽은 파일들)

git add를 해서 Git에서 파일을 읽어서 commit을 할수 있는 단계



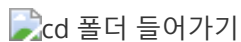
untracked files (Git에서 아직 읽지 못한 파일들)

git add를 하지 않아서 Git에서 파일을 읽지 못해 commit을 할수 없는 단계



cd (폴더 안으로 진입)

cd는 폴더 안으로 들어가는 명령어



echo : 생성할 내용과 함께 파일을 추가한다.

echo '저장할 내용' >파일명.파일타입 하면 새로 파일이 저장된다.
ex) echo hello world > a.txt



echo git 새로 추가내용>>a.txt

a.txt에 새로 내용을 추가 할수 있음
ex) a.txt에 원래 내용이 hello world만 있었는데
echo git study>>a.txt를 했기때문에
a.txt파일에는
hello world
git study으로 바뀌었다.

 echo 추가할내용 파일이미지

git status (git에 들어가있는 현재 파일들의 상태를 확인)

현재 파일의 상태를 확인 가능

빨간색글씨로 된 파일이 있으면 **untracked** 파일인것이다

 git status

git add (Untracking에서 Tracking으로 파일을 확인) or (working directory 에서 staging area로 파일을 보냄)

Git이 Tracking 할수 있도록 staging area에 옮길려면 **git add**라는 명령어를 이용하면 된다.


ex) `git add a.txt`

이름이 a인 txt형식의 파일을 untracking에서 tracking으로 만들수 있음
tracking된 파일은 초록색 글씨로 변함

 git add

git add *.txt (Untracking에있는 모든 txt파일을 전부 Tracking파일로 바꿈)

***.txt** = txt형식의 파일 전부를 tracking으로 바꾼다.

 git add all

git reset HEAD^ == git add 취소하기

`git reset HEAD^`
git add를 잘못했을때 사용

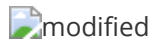
 git add취소하기

modified : 수정된 파일 (Tracking이된 파일을 수정해서 다시 Untracking으로 된 파일)

modified : c.txt 라고 되어있다면
c파일이 수정된 파일이라고 알려주는것

근데 여기서 Tracking이된 c.txt파일이 있고 (이건 기존에 Tracking되어 있는 c.txt)
Untracking이된 c.txt파일이 있는데 (이건 기존에 Tracking된 c.txt에 새로 수정된 c.txt파일)

그래서 다시 add로 c.txt파일을 tracking해주면 기존꺼랑 새로 수정된 파일이랑 겹쳐지게된다.



git rm --cached 빼낼 파일

`rm --cached`는 tracking된 파일을 다시 untracking된 파일로 빼낼 수 있음

ex) `git rm --cached a.txt`

=> tracking된 a.txt파일을 다시 untracking된 a.txt로 바꾼다



echo *.추가하고싶지 않은 파일형식>.gitignore

git에 포함하고 싶지 않을 때는

이렇게 Tracking하고 싶지 않은 파일들

깃과 깃허브에 올리고 싶지 않은 파일들은

`echo d.txt>.gitignore`

깃에 올라온 파일을 배제 시킬 수 있다.

폴더 안에는 존재하지만 깃에서는 없어진다.

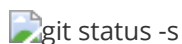
삭제하는 것이 아닌 일시적으로 배제시킬 수 있다.



git status -s

`git status -s`

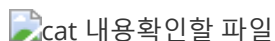
간단하게 깃의 상태를 표시해준다.



cat 내용확인할 파일

`cat c.txt`

c.txt파일에 있는 내용들을 확인할 수 있다.



git diff

두 저장소의 차이를 보여줌

로컬저장소 와 원격저장소의 차이를 보여줌

git diff는 로컬저장소와 원격저장소가 연결되었을 시 확인가능




git diff --staged = 원격이랑 차이를 보겠다.

내가 파일을 깃에 올렸을때 이전 버전하고 새로 업로드된 차이를 보겠다
--- 는 이전의 버전
+++ 은 이번의 버전에서 새로 생긴 파일이 뭐있는지 보여줌
+초록색 글씨는 새로 추가된 내용

 git diff --staged


git commit -m "메세지" : 메세지를 꼭 써야한다.

```
git commit -m "4월18일 저장"
```

 git commit

git commit -am "메세지"

```
git commit -am "메세지"
```

 git commit -am

git log (Git의 히스토리를 확인)

```
git log
```

언제 몇시에 파일이 저장되어있는지 확인 가능하다.

 git log

git reset(commit앞의 글자 6개 입력) --hard

```
git reset f7f67f --hard
```

git reset한 시점으로 돌아갈수있다 (대신 앞에 기록들은 삭제됨)

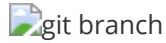
 git reset

git.reset 확인 (f7f67f으로 돌아와서 앞에 기록들은 히스토리는 없어짐)

 git.reset확인

git branch (평행우주)

```
git branch my-idea (my-idea라는 이름으로 평행우주 만듦)
```



git checkout (평행우주 이름) 이동

```
git checkout my-idea
master에서 => my-idea로 이동
master와는 별개로 여기서는 이것저것 막 시도해볼수있음
```



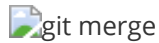
my-idea에서 파일 하나 생성



git merge (다른우주) 다른 우주에서 가져오기 (병합)

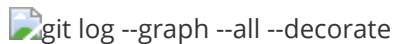
먼저 원래 우주로 돌아온 상태에서

```
git merge my-idea
```

를 입력하면
원래 우주에다가 my-idea우주의 파일들을 합칠수있다.

git log --graph --all --decorate (모든 우주의 기록 보기)

```
git log --graph --all --decorate
시각화된 다른우주에서의 작업내역을 볼수있다.
```



Rebase 다른 우주에서 가져오는 또다른 방법 (재배치)

깔끔하게 정리해서 가져올수있다 Tree구조로 이루어져있는데 이건 단일 트리로 가져올수있음
현재 이미 merge로 가져와서 가져올 데이터가 없어서 그런것
Current branch master is up to date == 현재 분기 마스터가 최신 상태입니다.



