버전 컨트롤

태그	
날짜	@2023년 5월 23일

GIT : 분산형 버전 관리 시스템

버전 관리가 가능하다. 다른 버전관리시스템과 차별화되는 점으로 협업이 가능하다.

로컬 머신에 있는 코드와 분산 서버에 있는 코드가 완전히 동일한 구조 로 되어 있으며 로컬 머신의 코드를 git push라는 명령어로 업로드 할 수 있다.

깃에는 세가지 영역이 있다.

작업을 하고 있는 작업 디렉토리, 즉 워킹 디렉토리라고도 한다.

깃에서 커밋을 하기 위해서 git add 명령어로 추가하면 올라가는 영역이 있는데 그곳을 스테이지 에어리어라고 한다.

git commit 하면 리포지터리가 있다 로컬머신에 리포지터리가 생기고 commit에서 push까지 하면 원격 서버에 commit한 파일들을 업로드 하게 된다.

파일 생성

git add test.txt

- git status 로 확인하게 되면 커밋할 변경 사항에 test.txt가 스테이징에리어로 올라간 것을 확인할 수 있다.

스테이지에서 commit을 하면

- git commit -m ‘test’ (-m 은 메시지의 약자)

리포지터리로 올라가게 된다. 그 후 버전관리가 가능해진다.

test를 고친 후 다시 commit을 하게 되면

브랜치가 origin/main으로 나온다. 이곳은 원격 서버를 뜻한다.

git log - -oneline 명령어를 입력하면 로그가 짧게 나온다.

해당 작업을 원격 서버로 올리려면 git push를 하면 된다.

그러면 다른사람은 해당 파일을 git pull을 사용하여 사용할 수 있게 된다.

github : 깃 기반의 호스팅 사이트

위에 변경한 파일들을 확인할 수 있다.

pull requests : 코드 관리자가 아닌 사람이 코드의 변경 요청을 할 때 사용하는 메뉴

github action : 코드 저장소에 어떤 이벤트가 일어날 때 특정 작업이 일어나게 하거나 주기적으로 어떤 작업들을 반복시켜서 실행하고 싶은 경우에 사용할 수 있다.

project : 프로젝트 관리 메뉴

wiki : 해당 프로젝트의 위키문서들을 저장할 수 있는 메뉴

security : 보안

insights : 이 저장소가 어떤 식으로 운영 되고 있는지 통계 수치들을 보여준다.

settings : 저장소의 대한 설정 변경 (변경, 삭제 등)

깃 허브를 사용하면 깃 기반으로 협업을 할 수 있고 주로 이슈와 풀리퀘스트를 통해 협업을 할 수 있다.