* **Github에 원격 저장소 만들기**

New repository 후 저장소를 하나 만든다. 그 후 원격 저장소에 저장할 디렉토리를 remote 시킨다. 또, 지역 저장소의 master와 원격 저장소의 master를 페어링 해야함.

$ git remote add origin <https://github.com/qweadzs/test.git>

$ git push -u origin master

* **같이 협업하기**

Settings 🡪 collaborators & teams 에 협업자 추가 가능. 추가하면 이메일로 메일이 감.

작업을 시작하기 전 pull이라는 명령어로 가장 최신 버전을 다운받아 작업을 해야한다. 안그럼 충돌이 발생하는데 같은 파일의 같은 부분을 수정할 때 충돌이 일어나는 것과 같은 원리.

* **Pull vs fetch**

원래 pull 🡪 commit 🡪 push 순인데

fetch 🡪 merge FETCH\_HEAD 🡪 commit 🡪 push 순으로 똑같이 할 수 있음.

즉 pull 은 fetch(리모트 브랜치를 가져옴) + merge FETCH\_HEAD

25f8b19 (HEAD -> master, origin/master) work 1

\* 4db48aa ver1

위에 origin이라는 이름의 저장소에 master 브랜치를 가리키고, 마지막으로 마스터 브랜치의 어떤 버전을 가져왔는지를 의미. 만약 오리진과 헤드가 마스터를 동시에 가리키지 않으면 push나 pull을 해야함.

* **오픈소스 참여하기 : Patch**

권한없이 오픈소스의 프로젝트에 참여하는 것. 마지막 작업 이후 만들어진 버전들을 주고 싶다면 git format-patch 마지막작업id 를 입력한다. 그러면 .patch 파일이 생성되는데 이걸 주면 된다. 언제 누가 어떤내용을 작업했는지 각 버전과의 차이점을 알 수 있다. 그런데 이렇게 patch로 받게되면 커밋 id가 다르다.

* **Pull request**

내가 작업한 내용을 가져가 달라고 요구 하는 것. 오픈 소스 개발방식에서 사용된다.

Fork = 원격 저장소를 복제해서 내 것으로 가져오는 것.

만약 깃허브 주소를 클론해서 작업을 하고 push를 한다면 그 주소에 해당하는 원격 저장소에 업로드가 된다.

Compare = 포크된 저장소와 포크한 저장소의 차이점을 비교해준다.

그 후 pull request를 요청하면 original에게 당신의 저장소를 기반으로 작업을 했는데 당신에게 필요한 작업을 했으니 가져가 달라고 요청된다.

머지를 하면 commit까지 완료되어 자동으로 합쳐진다.

\*\*명령어

Git remote add origin 깃허브주소 = 현재 디렉토리를 원격저장소에 연결

Git push -u origin master = 지역 저장소의 master와 원격 저장소의 master를 페어링

Git clone 깃허브 주소 B = B라는 디렉토리에 깃허브에 올라온 디렉토리를 복사한다.

git config --global alias.l 'log --all --graph --oneline' = git l 로 저 의미를 함축시킬 수 있음.

Git am -3 -I \*.patch = 3way merge를 이용, 패치 하나를 적용할 때마다 물어보는 패치를 적용.