**--------------------------github---------------------------------**

**jeonjunmin / Wnsals5348**

**>> git 시점 만들기**

**git status**

**git add . / git add -A**

**git commit -m "messsage"**

**>>현재 연결된 github리스트보기**

**git remote**

**>> git hub에 보내기 //github에 master브랜치가 없으면 새로 생김**

**git push origin master //master에서 master로 보내기 ,**

**git push -f origin master //master에서 강제로 master로 보내기**

**>>git hub에서 가져오기**

**git pull origin master**

**>>git 기록 보기, 소스 시점 바꾸기**

**git log //커밋시점로그**

**git reset 로그번호6자리 --hard //커밋시점이동**

**git revert 현재로그번호6자리 //바로전커밋시점이동**

**>>branch만들기**

**git branch 새로운branch명 //브랜치생성**

**git checkout 새로운branch명 //브랜치이동**

**>>branch목록보기**

**git branch //브랜치목록**

**git fetch //브랜치 현황 최신화**

**git branch -a //github에 연동된 브랜치 모두보기**

**>>merge 하기**

**git merge 가져올branch //현상태의 브랜치에서 가져올branch를 병합함**

**git log --graph --all --decorate //병합기록 보기**

**git rebase 가져올branch //브랜치 재정렬합치기. 재배치정리하기.**

**>>git hub에 있는거 로컬에 새로운 branch(mytest2)로 가져오기**

**git checkout -b mytest2 origin/mytest**

**>>원격 브랜치 지우기**

**git push -d origin 원격브랜치명**

**- pull request를 하는 이유**

* 자연스러운 코드 리뷰를 위해
* Push 권한이 없는 오픈 소스 프로젝트에 기여할 때
* 콜라보레이터에 소속되어있는 경우에는, 그 저장소에서 브런치를 따고 푸쉬하면 풀리퀘가 가능하다.  
  + 에서 branch가 보이는 경우는 그 저장소만의 기능이다
* Push로 협업했을때, 다른 사람의 commit을 볼 일이 많지 않고, master branch와 merge할 때서야 보게되는데, Pull Request는 당장 merge하지 않는다는 규칙이 Pull Request를 보고, 코드에 신경쓰게되고 어떤 작업이 언제 적용되었는지 알 수 있다. 오히려 당황스러운 코드충돌을 줄일 수 있다.

**- 방법**

**1. Fork**

* Upstream Repository를 자신의 저장소로 Fork(Origin Repository)한다.

**2. Clone, remote 설정**

* fork로 생성한 Repository에서 clone or download 버튼을 누르고 표시되는 URL을 복사한다.

$ git clone (복사한 URL)

* 내 컴퓨터에 생성된 로컬저장소에 원격저장소를 추가한다.
* Github에서 새로운 Repository를 생성
* 위와 같은 방법으로 clone or download 버튼을 누르고 표시되는 URL을 복사한다.
* Clone했던 원본프로젝트저장소(origin)를 원격저장소(github)로 추가

$ git remote add post(별명) (복사한 URL)

fork, clone한 프로젝트는 origin이라는 별명이 기본으로 추가되어있다. 때문에 따로 설정해주지 않아도 되고, 원격저장소로 추가할때에는 별명을 설정해주어야함.

* 원격저장소 설정 현황 확인하는 명령어

$ git remote -v

**3. branch 생성**

* 내 컴퓨터의 Clone프로젝트 저장소(origin)에서 코드를 수정하거나 추가하는 작업은 branch를 만들어서 진행한다.

$ git checkout -b develop

* develop branch 생성

$ git branch

\* develop

master

* 생성된 2개의 branch를 확인할 수 있다.

**4. 수정 작업 후 add, commit, push**

* editor를 통하여 코드를 수정한다.
* 작업이 완료되면 Github Repository(origin)에 add, commit, push한다.

$ git push origin develop

* push할때 develop 브랜치의 수정내역을 origin으로 푸시한다.

**5. Pull Request 생성**

* push 완료후 자신의 github 저장소에서 Compare & pull request버튼이 활성화 되어있는걸 확인할 수 있다. 버튼을 선택해 Pull Request를 생성한다.

**6. Merge Pull Request**

* PR을 받은 관리자는 코드 변경내역을 환인하고 Merge여부를 결정하게 된다.

**7. Merge 이후 동기화 및 branch 삭제**

* Merge가 완료되면 로컬 코드와 원본의 코드를 병합하고 최신의 상태를 유지하게 위해 동기화한다.

$ git pull post(remote 별명)

$ git branch -d develop

* 위 명령어를 통해 동기화하고, 브랜치를 삭제한다.
* 나중에 추가적으로 작업이 또 필요하다면, 동기화를 한 뒤 3-7번을 반복하면서 작업하면 된다!