Git Bash를 이용한 협업

Git 기본 명령

1. git init
   .git 숨김 폴더를 생성한다. .git 폴더가 있어야 git이 추적한다.

2. git add
   변경된 내용을 staging 영역으로 옮긴다.

3. git commit
   staging 영역의 내용을 local repository의 내용으로 확정한다.

4. git push
   local repository의 내용을 remote repository로 올린다.

5. git pull
   remote repository의 내용을 local repository로 내려 받는다.

Git 기본 명령

6. git remote add origin 깃허브주소
   깃허브주소 를 remote repository(원격 저장소)로 등록한다.

7. git clone 깃허브주소
   .git 디렉터리를 포함해서 remote repository의 내용을 모두 local repository에 복제한다.

8. git branch 브랜치명
   새로운 브랜치를 만든다.

9. git checkout 브랜치명
   다른 브랜치로 이동한다.

10. git merge 브랜치명
    다른 브랜치 내용을 현재 브랜치에 병합한다. 이 때 두 브랜치에서 같은 파일을 동시에 수정하면 충돌(conflict)이 발생하므로 주의한다.

git

GitHub

협업 시나리오

1. 3명이 한 조입니다.
2. 조원1이 기본 세팅을 마친 프로젝트를 Github에 올립니다.
3. 조원2,3은 Github에 올라간 프로젝트를 자신의 PC로 가져옵니다.
4. 조원1,2,3은 각자의 PC에 각자의 브랜치를 생성해서 작업을 수행합니다.
5. 조원1이 자신의 기능 구현을 마치면 자신의 브랜치를 main 브랜치에 병합한 뒤 Github에 올립니다.
6. 조원2,3은 조원1이 Github에 올린 최신 버전의 main 브랜치를 자신의 main 브랜치로 가지고 와서 작업을 계속 진행합니다.

조원1이 기본 세팅을 마친 프로젝트를 Github에 올립니다.

- 조원1은 프로젝트를 생성하고 pom.xml, web.xml 등 초기 세팅을 합니다.
- 조원1이 Github에 remote repository를 생성합니다. remote repository는 프로젝트와 같은 이름이면 됩니다. README.md와 .gitignore를 포함합니다.
- https://github.com/에서 Settings – Collaborators 메뉴를 이용해서 조원2,3을 collaborator로 추가합니다. 그래야만 조원2,3도 repository에 pull, push를 할 수 있습니다.

```
$ git init
$ git add .
$ git commit -m '커밋 메시지'
$ git remote add origin 깃허브주소
$ git pull origin main --allow-unrelated-histories
$ git push origin main
```

git

GitHub

조원2,3은 Github에 올라간 프로젝트(main 브랜치)를 자신의 PC로 가져옵니다.

- 조원2,3은 자신의 PC에 workspace를 준비하고 Github에 올라간 프로젝트를 복제합니다.(clone)
- 복제를 하면 자동으로 remote repository로 등록됩니다.
- 복제가 완료되면 모든 조원들은 본인의 PC에 동일한 main 브랜치를 가지게 됩니다.

```
$ git clone 깃허브주소
```

git

GitHub

조원1,2,3은 각자의 PC에 각자의 브랜치를 생성해서 작업을 수행합니다.

- 조원1,2,3은 각자의 PC에 자신이 작업할 브랜치를 생성합니다.
- 각 브랜치이름을 a,b,c라 칭하겠습니다.
- 조원1,2,3은 각 기능 구현이 끝나면 자신의 브랜치를 push합니다.
- 충돌(conflict)을 방지하려면 조원별로 작업할 범위를 잘 나누는 것이 중요합니다.

```
$ git branch a  // 브랜치 a 생성
$ git checkout a  // 브랜치 a로 이동
$ git add .
$ git commit -m '[조원1] 어떤 기능 구현 완료'
$ git push origin a
```

git

GitHub

조원1이 자신의 브랜치 a를 main 브랜치에 병합한 뒤 Github에 올립니다.

- 조원1은 기능 구현을 마쳤습니다.
- 조원1은 먼저 local repository의 main 브랜치에 a 브랜치를 병합합니다.
- local repository의 main 브랜치를 Github에 업로드(push)합니다.

```
$ git checkout main
$ git merge a
$ git push origin main
```

git

GitHub

조원2,3은 조원1이 올린 최신버전의 main 브랜치를 가져와서 작업을 계속 진행합니다.

- 조원2,3은 조원1이 Github에 올린 최신버전의 main 브랜치를 받아옵니다.
  (main 브랜치로 이동한 다음 pull 합니다.)
- 조원2,3은 최신 버전의 main 브랜치를 가져온 다음 자신의 작업 브랜치(b 또는 c)에
  최신버전을 병합합니다.

```
// 브랜치 b에서 작업 중인 조원2 예시
$ git add .
$ git commit -m '[조원2] 어떤 기능 구현 중'
$ git checkout main
$ git pull origin main
$ git checkout b
$ git merge main
```

git

GitHub

협업 시 주의사항

1. push 하기 전에 반드시 local repository는 remote repository의 최신 버전을 유지해야 한다.
2. pull은 remote repository의 내용과 local repository의 내용을 비교하고 병합(merge)한다.
3. 조원1이 index.jsp를 수정하고, 조원2도 index.jsp를 수정했다면 충돌이 발생하므로 주의한다.
4. 충돌이 나면 git이 자동으로 fast-forward(최근 커밋으로 변경)할 수 있다. 하지만 대부분은 사람이 직접 일일이 확인해서 수정하는 작업을 해야 한다.
5. 기능마다 커밋이 하나씩만 있는게 좋다. 한 기능에 커밋이 여러 개 있으면 디버깅이 불편하다. 가능하면 기능이 완료된 뒤 커밋을 하자.
6. 어느 시점으로 돌아가려면 git reset 또는 git revert를 이용한다.
   - git reset 커밋아이디
     1. 해당 커밋시점으로 이동하고 그 동안의 커밋 이력도 삭제
     2. 커밋 이력이 삭제되면 remote repository와 이력이 안 맞아서 push가 불가함
   - git revert 커밋아이디
     1. 해당 커밋시점으로 이동하지만 그 동안의 커밋 이력은 유지
     2. 이미 remote repository에 push를 한 상태에서 되돌아가려면 git revert 이용