<VSCODE에서의 Git 사용법>

1. 로컬 = 내 PC의 깃 저장소, 원격 = 깃허브(온라인) 저장소

2. git clone으로 깃허브의 파일들을 로컬로 옮길 수 있음

3. git push = 로컬에서 작업(중간중간 커밋) 후 원격저장소에 push

4. git pull = 협업 중에 다른 팀원이 원격저장소에 push해서 변경된 내용을 현재 내 로컬 환경에 반영하고 합쳐 개발하고자 하는 목적

5. 다른 팀원이 원격저장소의 내용을 수정했을 때, 내가 개발한 내용을 push하면 오류가 뜸. 왜냐하면 다른 팀원의 결과물에 내 것을 덮어 씌울 수 있기 때문. 따라서 pull을 push 하기 전에 하던가 fetch와 merge과정으로 나눠서 해결 가능함

6. fetch = 원격 저장소의 내용을 로컬로 가져옴, 이를 통해 로컬과 원격의 차이를 비교 가능. 이를 통해 충돌 여부 등을 확인 가능함

7. merge = fetch로 어떻게 하면 좋을지 확인한 후 merge로 두 브랜치를 병합

8. git init = 로컬에서 개발하고 git을 이용해 원격에 업로드 하는 코드

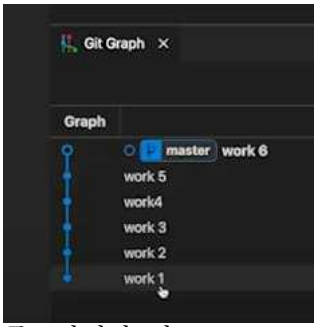
9. git add = commit을 할 때, 한 commit에 여러 파일의 수정사항을 기록해도 되지만 한 개의 파일의 수정사항만 기록할 수 있다면 다른 팀원들이 보기에 수월할 것임. 이때 활용함. git add는 git commit에 포함될 파일을 지정함. git add를 통해 하나의 파일을 지정하면 해당 파일을 stage에 올린다고 함(vscode 메뉴로도 있음, 더하기 표시)

10. stage = stage(플러스 표시)를 한 파일만 커밋이 됨

11. working directory = 사람이 작업하는 경로

12. git checkout = 프로젝트를 진행하던 중, 코드에 문제가 생겼을 때 git checkout을 이용해 기존 commit history에 있는 잘 작동하던 버전으로 저장소를 변경하고 오류를 수정해 다시 commit으로 버전을 업데이트 할 수 있음

13. head = vscode에서 git graph를 봤을 때,



동그라미가 있고 master가 떠있는 부분. 이 head를 work1으로 바꾸면 그 예전 버전에서 작업 가능. 이때 쓰는 명령이 checkout. git graph에서 우클릭 후 checkout. 바꾸면 그 버전 앞에 동그라미 생김

14. 시간 여행(버전 이동)을 끝낼 때는 master에 우클릭해서 checkout 할 것

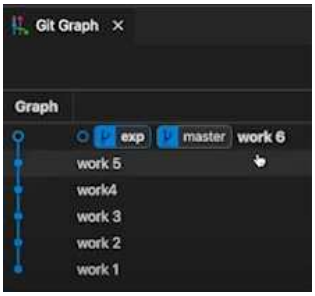
15. origin = 원격 저장소

16. add remote = 로컬에서 시작된 프로젝트를 새로운 원격저장소에 push 하기 위해 사용

17. branch = 하나의 프로젝트에서 master branch 말고 새로운 branch를 만들어 실험을 해볼 수도 있고 양식을 다르게 해서 기업들에게 줄 수도 있음. vscode를 기준으로 git graph에서 허공에 우클릭 후 create branch로 생성 가능.

위는 exp라는 branch가 생성된 모습

18. 실험을 하려면 head가 exp같은 새로만든 brach가 되어야 함
exp위에 우클릭하고 checkout branch 클릭



19. 새로운 버전이 만들어지면 head가 가리키는 brach가 새로운 버전이 만들어짐

20. 다 개발을 마치고 새로 만든 branch(이름은 예시로 exp)를 master에 병합하고 싶을 때는 master에 checkout 된 상태(왼쪽에 동그라미)에서 병합할 exp우클릭 - merge into current branch - yes merge

21. 20번에서 병합 후 exp branch를 지워도 master에 남아 있음

22. conflict(충돌)이 발생한 경우 vscode에서 지원하는 방식으로 해결할 수 있음

23. master branch에 push하기 전에 다만든거 검토가 필요할 때, 새로운 branch를 만든 다음에 개발 내역을 해당 branch에 적용하고 그 내용이 반영되기 전에 깃허브에서 Pull Request메뉴로 사람들에게 검토를 받을 수 있음

24.