6/30 git Seminar (Windows 기준)

- https://git-scm.com/book/en/v2/Getting-Started-Git-Basics
- git의 활용
  - e.g. 파란색을 입혀보고 빨강을 입혀본 것과 비교해보기 (이 때 파란색 가지와 빨간색 가지를 뻗어나가는 것은 branch 개념이 아님!! branch랑 혼돈하지 말 것!)*
  - 협업
- git (language) vs. github (platform)
- github의 세 가지 모드
  - working directory
    - 내가 일하고 있는 상태
  - staging area
    - add을 해도 com\mit이 되지는 않음; staging 단계로만 들어감.
    - add한 상태에서 또 변경사항이 있을 때 다시 add를 해줘야함.
  - git directory
    - 각 버전들의 (순간적인) 스냅샷을 간직하고 있는 트리 (각각의 node, 혹은 스냅샷을 commit point라고 부름)
- working directory와 git directory의 상태가 다를 때, 언제 commit을 하고 싶은지 정하는 것이 중요
  - add를 하다가 새로운 스냅샷을 찍을 때가 되었다 싶을 때 commit하면 됨.
  - commit할 때는 add한 것들만 변경이 됨. working directory의 모든 변경사항이 반영되는 것이 아님.
- branch란?
  - 일단 첫 commit에 master branch가 준비 됨.
  - commit을 하는 순간 거기로 master가 옮겨감.
  - 나무의 가지보다는 보드게임의 말이라고 보는 것이 맞음.
    - *e.g. 파란 가지를 따라가는 파란 말을 만들면 걔가 파란 branch, 빨간 가지 따라가는 빨간 말이 빨간 branch
    - 말을 얼마든지 자유롭게 이동 가능 (e.g. 뒤로 돌아가기, 빨간 말이 파란 가지로 가기)
    - branch가 없이도 새로운 commit point를 만들며 이동할 수 있음. 그러나 일반적으로 기억하고 작업하기 쉽도록 작업하고자 하는 기능에 해당되는 이름을 가진 branch를 새로 만들어서 직접 이동함.
    - 보통 기능에 따라 branch를 새로 만들어서 작업함. 하나의 기능이 온전하게 만들어졌다 싶으면 master branch로 이동해서 sub branch를 가져와서 merge함.
    - 보드게임이랑 동일하게 어느 위치로 말을 이동시키는지가 중요하고 (말의 위치 ~ branch의 commit point)
  - merge란?
    - 일단 branch가 두 개 있어야 함. (주로 master branch와 sub branch를 합침)

        - sub branch의 내용을 master branch로 합침. 그게 merge!
        - merge한 것도 하나의 commit point인 셈.
        - 충돌이 생긴다면?
            - github이 똑똑해져서 눈치로 자동 merge를 해주는 경우가 있음. (auto-merge)
            - auto merge하기에 애매한 경우는 conflict가 있다고 표시가 돼서 수동으로 수정을 하면 됨.
            - 일단은 충돌이 생기지 않도록 같은 파일을 건드리지 않는게 중요 (e.g. routes같이 잘 건드리게 되는 애들은 처음에 큰 틀을 능력자가 잡아놓고 시작하는게 좋음)
- 간단 실습
    - 기본
        - mkdir gittest
        - cd gittest
        - vi a.txt --> first line이라 입력
        - a.txt 나가서 git status 나가면 에러 뜸 (git project가 아니라는 뜻)
        - git init해서 git 폴더로 만들어줌.
        - ls -a 치면 .git 폴더가 만들어져있음. 모든 commit point가 이 곳에 저장이 됨.
        - 이제는 git status를 치면 Initial commit라고 뜸
        - git add a.txt
            - a.txt가 commit하고자 하는 변경사항으로 지정이된 셈. staging area로 감.
        - git commit -m "Create a.txt"
            - add하면서 지정된 변경사항들만 git directory로 commit이 됨.
            - commit 메세지 예쁘게 잘 씁시다
        - git status라고 치면 "nothing to commit, working directory clean"
        - a.txt.를 수정하고 b.txt를 새로 만들어봤음
            - git status:
                - modified: a.txt
                - untracked files: b.txt
            - 이 상태에서 git directoy: 수정 전의 a.txt, working directory: 수정된 a.txt와 b.txt
        - 수정된 파일은 하나하나 다 add를 해줘야 하는 건가요?
            - git add .도 가능 (수정된 모든 파일 추가)
            - 그리고 일부를 제외한 대부분을 commit하고 싶으면 .git 폴더에 있는 git ignore 파일에서 제외할 몇 가지를 지정할 수 있음.
        - 결국, git로 track하고자 하는 폴더로 이동 (working directory) --> git init --> 변경하고자 하는 애들을 add (staging area) --> commit (git directory)
            - 중간중간에 git status로 확인
    - 복수의 branch
        - git checkout -b mybranch (mybranch라는 새로운 브랜치를 만들었음)
        - git branch (어떤 branch가 있는지 확인)
        - git log --graph에서 각 branch가 어디있는지, 어떤 경로로 이동해 왔는지를 4개까지 확인할 수 있음.

        - git log --graph --all 하면 다 보여줌 (엔터 칠 때마다 하나씩 더 보여주고 :q하면 꺼짐)
    - git checkout master 치면 mybranch에서 master로 이동됨
        - git branch치면 *되어있는게 현재 있는 branch
    - merge할 때는 합친 결과물이 남아있을 branch로 이동해서 합침당할 아이를 merge하면 됨.
        - master branch에서 git merge mybranch --> 기괴한 창
        - Ctrl + X --> 기괴한 창 나가짐. (vim에서는 걍 저장하고 나가면 됨)
        - git log --graph --all에서 좌측 벽에 작은 동산이 솟아있는걸로 merge 됐는지 확인 가능
- complete
    - 서로 다른 branch에 있는 같은 파일을 동시에 수정한다면?!
        - auto-merge가 될 수도 있음
        - 안되면? <<<<<<<<<<,(이걸 고를까) ==========,(저걸 고를까) >>>>>>>>>를 기준으로 고르면 됨.
- mybranch에 있는 말을 master로 옮기고 싶으면 master의 변경사항들을 mybranch가 모두 반영하고 있으면 됨
    - mybranch로 이동해서 master를 merge하기!
- github
    - add. commit, checkout, status, log는 인터넷 연결 없이도 사용 가능 (로컬 작업) / push, pull은 인터넷 필요
    - 로컬의 내용을 github에서 반영하려면 git push
    - push할 경우 해당 branch만 반영이 됨
        - master에서 각기 다른 branch를 만들어서 push하면 master에서 여러 branch가 뻗어나감.
        - 각 branch를 pull할 수 있음.
        - 내꺼만 push하지 말고 master 자주 가져와서 다른 사람들이 작업한 내용이 update된 상태에서 작업합시다
- 오늘의 꿀팁
    - git add .; git commit -m "commit message" 방식으로 한 줄 쓰기 가능!
    - github --> insights --> network에서 log 확인 가능