git & gitHub

2021.02.09

git 이용한 문서 버전관리

- git이란 소스코드를 효과적으로 관리하기 위해 개발된 '분산형 버전 관리 시스템'입니다.
- 원래는 Linux 소스코드를 관리할 목적으로 개발 되었습니다.
- git에서는 소스 코드가 변경된 이력을 쉽게 확인할 수 있고, 특정 시점에 저장된 버전과 비교하거나 특정 시점으로 되돌아갈 수도 있습니다.
- 저장소(git repository)란 말 그대로 파일이나 폴더를 저장해 두는 곳입니다. 그런데 git 저장소가 제공하는 좋은 점 중 하나는 파일이 변경 이력 별로 구분되어 저장된다는 점입니다. 비슷한 파일이라도 실제 내용 일부 문구가 서로 다르면 다른 파일로 인식하기 때문에 파일을 변경 사항 별로 구분해 저장할 수 있습니다
  - 원격 저장소(Remote Repository): 파일이 원격 저장소 전용 서버에서 관리되며 여러 사람이 함께 공유하기 위한 저장소입니다.
  - 로컬 저장소(Local Repository): 내 PC에 파일이 저장되는 개인 전용 저장소입니다.

- ✓ git은 파일을 Committed, Modified, Staged 이렇게 세 가지 상태로 관리한다. Committed란 데이터가 로컬 데이터베이스에 안전하게 저장됐다는 것을 의미한다. Modified는 수정한 파일을 아직 로컬 데이터베이스에 커밋하지 않은 것을 말한다. Staged란 현재 수정한 파일을 곧 커밋할 것이라고 표시한 상태를 의미한다.

git 설치 및 업그레이드 & gitHub 계정 생성

// git 설치

https://git-scm.com/ 최신 버전 다운받아 설치 (windows용 64bit)

// 설치된 git 버전 확인

$ git --version

// git 업그레이드(Windows 용)

$ git update-git-for-windows

// 원격저장 클라우드 서비스 중에 하나인 gitHub에 계정 생성 및 repository 생성할 것

https://github.com/

git 환경 설정을 위해 해야 하는 것들 (I)

git를 사용하기 위해 필요한 설정

$ git --help <명령어>

git 명령 실행 시 필요한 도움말 보기

$ git init

원하는 폴더를 선정한 후 해당 폴더에서 상기 명령 실행하여 새로운 저장소 생성 : .git 폴더 생성.

main Branch 생성

기본으로 생성된 main Branch는 사회적인 의도로 master/slave 라는 용어 사용을 자제하기 위해 현재 main으로 변경되고 있는 추세로 변경하는 방법은 아래.

$ git config --global init.defaultBranch main **=> git init 을 통해 새로 생성할 Repository의 기본 Branch 명을 "main" 으로 변경**

$ git branch –m main **=> 이미 생성된 Repository의 기본 Branch 명을 "main" 으로 변경**

git 환경 설정을 위해 해야 하는 것들 (II)

저장소별 사용자명/이메일 구성하기

$ git config (--global) user.name "Your Name" // user.name 등록 : gitHub에 등록된 user name

$ git config (--global) –unset user.name "Your Name" // user.name 삭제

$ git config (--global) user.email "Your email address" // user.email 등록

$ git config (--global) --unset user.email "Your email address" // user.email 삭제

$ git config (--global) --list // user 설정 정보 조회

***** git config로 환경 설정시 범위는 3가지 : -- local, --global, --system**

$ git remote -v

연결된 원격저장소 리스트 보기

$ git remote add "원격저장소 이름" "원격저장소 url"

원격저장소 연결, <원격저장소 이름> : 로컬에 저장하는 원격저장소 이름으로 일반적으로 origin 사용.

$ git remote remove "원격저장소 이름"

git에서 원격 저장소 정보 삭제, 실제 원격 저장소가 삭제되는 것은 아님.

git status : 파일 상태 변화

Untracked File → Tracked(Staged File → Committed File)

$ git status --short

// ?? : Untracked, ! : ignored, 나머지는 Staged (A: Added, M: Modified, R : Renamed, C: Copied, D: Deleted, U: Updated but unmerged)

$ git status

Working Directory와 Index의 상태를 확인하기 위해 사용

Untracked files : Working Directory에 있는, 한 번도 git 저장소에 관리된 적이 없는 파일 목록

Tracked Files : 한 번 이라도 commit을 한 파일의 수정 여부를 계속 추적하는 상태

- **Changes to be committed:** Staging 되었으며, commit 될 수 있는 파일 목록
- **Changes not staged for commit:** 수정되었으나 아직 Staging 되지 않은 파일 목록

// 관련 명령어들

$ git rm "파일" // commit 된 파일을 삭제, 실제 폴더 상에서도 삭제함. 주의 !!!

$ git rm --cached -r * // Staged 된(add 된) 파일을 취소, 즉 git add 명령을 취소하는 명령 (-r 옵션은 서브 디렉토리까지 적용의 의미)

$ git mv "파일1" "파일2" // 다음과 동일 : $ mv "파일1" "파일2" 수행 후 이어서 $ git rm "파일1", $ git add "파일2"

git add

새로 파일을 추가하거나 또는 기존 파일을 편집하여 수정하여, 작업 디렉토리(working directory) 상의 변경 내용을 스테이징 영역(staging area, index)에 추가.

$ git add 파일명 : 기술된 파일이 Staging 된다. 여러 파일 가능(스페이스로 구분)

$ git add * : 현 위치 디렉토리 이하의 모든 변화를 Staging (.gitignore 포함 될 수도)

$ git add . : 현 위치 디렉토리 이하의 모든 변화를 Staging (.gitignore 제외)

$ git add -u : 이미 add된 파일을 업데이트 하는 경우, 끝에 path를 주지 않으면 전체

$ git add -A : git init를 실행한 저장소 이하 모든 곳의 모든 변화를 Staging

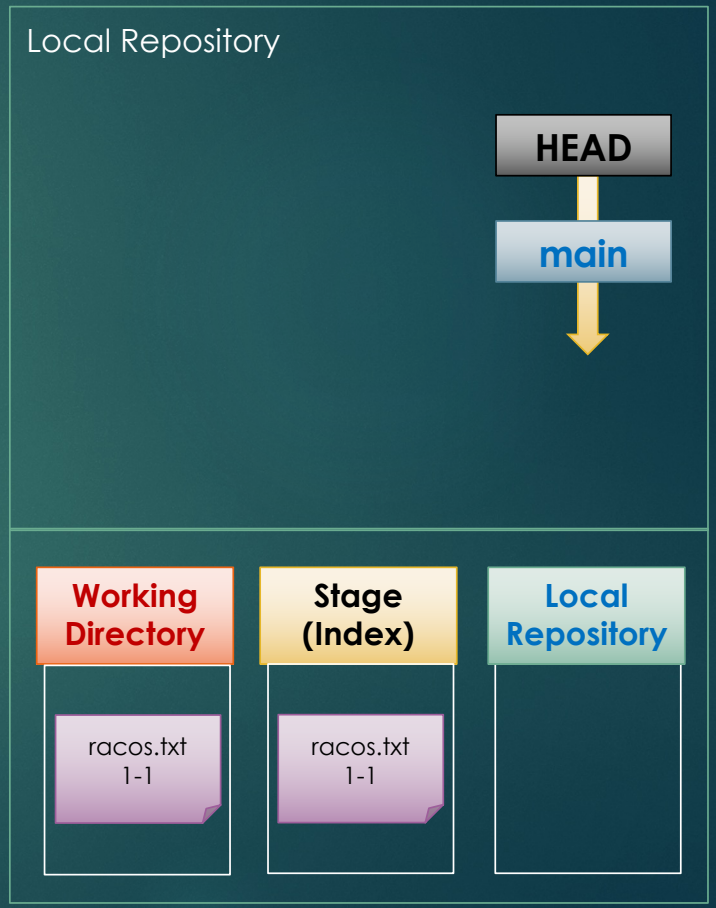
$ git add -p : -A 옵션과 동일하며 추가로 변경 내용 확인하며 Staging 가능

$ git add -i : 명령어를 선택해서 Staging 가능

$ git rm --cached -r *

Staging 된 파일을 Unstaging 상태로 변경,

-r : 하위 디렉토리까지 변경



git commit

새로 파일을 추가하거나 또는 기존 파일을 편집하여 수정하여, 작업 디렉토리(working directory) 상의 변경 내용을 스테이징 영역(staging area, index)에 추가.

$ git commit -m "메시지"

Staging된 파일들을 로컬 저장소(.git)에 저장. "메시지"를 통해 해당 commit이 어떤 변경 내용인지에 대해 반드시 메시지 남길 것. **$ git commit -am "메시지"** 를 통해 add와 commit 한 번에 수행할 수도 있음. **$ git commit --amend** 이미 commit된 내용에 수정하고자 하는 경우 사용

$ git log : commiit된 log 정보를 보여줌. ** 참고 : git log

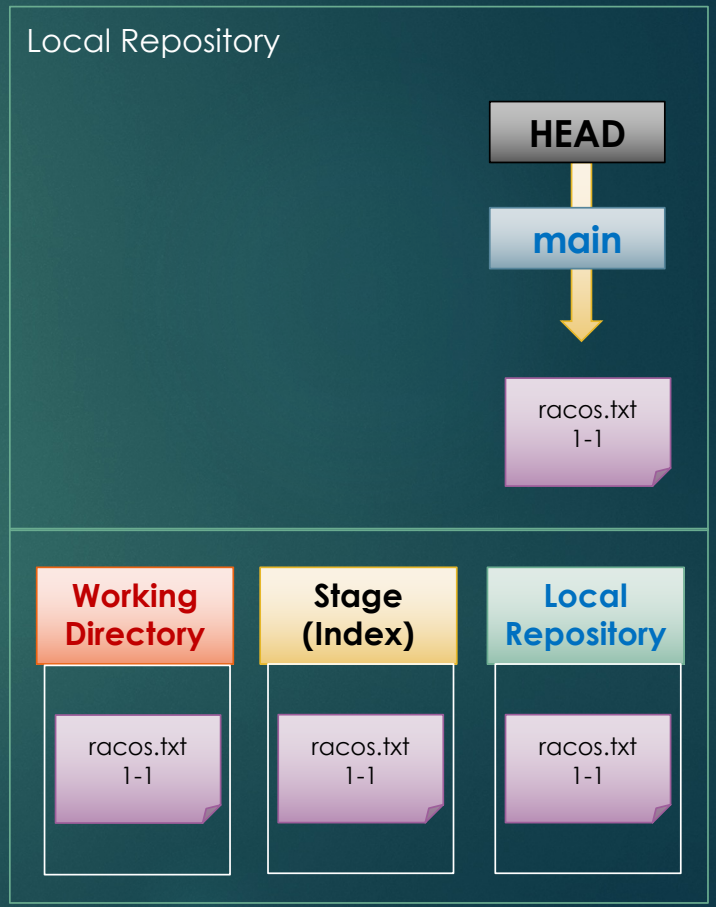
$ git reset --soft HEAD~ : Local Repository의 HEAD를 이전 commit으로 이동.

$ git reset HEAD~ == $ git reset --mixed HEAD~

: Staged File도 이전 commit으로 이동

$ git reset --hard HEAD~

*** HEAD^ HEAD^^ HEAD~ HEAD~2 HEAD~3 : 의미 이해 할 것.



git reset

여러 명이 원격저장소 사용하여 공유하는 경우에는 사용금지, 대신 git revert 사용