STEP

디지털 기초

Git 기본 퀵 가이드

2차시 : 기본 명령의 동작 원리

커밋 관련 명령

1. 커밋(Commit)

(1) 커밋이란?
- 커밋 명령으로 생성한 객체
- 작업 디렉토리의 스냅샷

(2) 커밋의 특징
- 한 번 생성한 커밋은 안전하게 저장
- 커밋 생성 시점으로 파일 되돌리기 가능

2. Git 사용하기 1단계: 사용자 정보 입력하기

(1) 소스트리에서 사용자 정보 입력하기

① 사용자 정보 입력: 소스트리의 [도구] – [옵션] – [일반] 탭
② 이름: 영문 입력
③ 이메일: GitHub 가입 계정 입력

(2) CLI로 사용자 정보 입력하기

① $ git config --global user.name “내 이름”→이름
② $ git config --global user.email “내 이메일 주소”
→ 이메일 주소
③ $ git config --global color.ui auto→깔끔한 화면 설정

커밋 관련 명령

3. Git 사용하기 2단계: Git 저장소 생성하기

(1) git init 활용

① git init 명령
② 현재 폴더 내 비어있는 Git 저장소 초기화
③ 저장소 생성(=로컬 저장소)
④ 생성된 커밋은 '.git' 폴더(로컬 저장소)에 저장

(2) git clone 활용

① git clone 활용 저장소 생성 시 특징
- 프로젝트 이름에 해당하는 디렉토리 자동 생성
- 로컬 저장소 및 필요한 파일 초기화

② git clone 사용조건
- 원격 저장소에 해당 프로젝트가 존재할 것

4. 커밋 생성 조건

(1) 작업 디렉토리에 신규 파일 생성하기

① 소스트리에서 클론한 프로젝트 선택
② 탐색기 버튼 클릭
③ 해당 저장소의 작업 디렉토리로 이동
④ 클론을 통해 만들어진 저장소에 신규 파일 생성 및 내용 입력 후 저장

(2) 커밋의 특징

① 스테이지에 업로드된 파일로 생성
② 스테이지에 작업 파일 미업로드 시 커밋에 작업 내용 미반영

커밋 관련 명령

5. 작업 파일 스테이지 업로드 방법

(1) CLI 활용

① 'git status' 명령: 현재 작업 디렉토리의 상태 확인하기
② 'git add [파일명]' 명령: 파일을 스테이지에 업로드 및 업로드 상태 확인하기
③ CLI 명령 사용 시 주의사항: 화면의 모든 메시지를 꼼꼼하게 확인하기

(2) 소스트리 활용

① CLI보다 훨씬 쉽게 Staging, Unstaging 가능

커밋 관련 명령

6. 첫 번째 커밋 생성하기

(1) 커밋 생성 명령

① $ git commit –m “[커밋 메시지]”
② 커밋은 반드시 스테이지 내용으로만 생성 가능

(2) 커밋 생성 과정

① 작업 디렉토리 내 파일 생성 및 변경
② 스테이지 업로드
③ 스테이지 내용 기반 커밋 생성

(3) git commit 명령 활용

① git commit: 커밋 생성 명령
② git status: 상태 확인
  - working tree: 작업 폴더
  - working tree clean: 작업 디렉토리에 커밋할 내용 없음
③ git log: 커밋 로그, 히스토리 확인
④ HEAD: 현재 작업 중인 브랜치
⑤ origin/main: 원격 저장소

커밋 관련 명령

7. 두 번째 커밋 생성하기

(1) github.py 신규 생성

(2) hello.py 내용 추가

(3) 소스트리 활용

① 커밋 버튼을 눌러 새로운 커밋 생성 가능
② 커밋 메시지 작성 중요

(4) 커밋 메시지 작성 규칙

① 첫 줄: 제목 작성
② 두 번째 줄: 비워 두기
③ 세 번째 줄: 본문 작성

(5) 커밋 메시지 작성 시 유의사항

① 메시지의 내용은 간결하고 명확하게 작성
② 커밋의 내용은 되도록이면 자세하게 작성
③ 커밋 생성 이유에 대한 부가 설명 추가

커밋 관련 명령

8. 작업 내용 원격 저장소 업로드

(1) git push [alias] [branch]→git push origin main

① [alias] : 원격 저장소의 별명 입력
- 첫 번째 원격 저장소 이름 : origin

② [branch] : 브랜치의 이름 입력
- 현재 브랜치의 이름 : main

9. 커밋 복구

(1) 하드 리셋 활용

① '$ git reset –hard HEAD~2' 명령 사용
② 작업 중인 브랜치의 커밋을 두 단계 이전으로 복구

(2) 소스트리 활용

① 두 단계 이전 커밋 선택 후 마우스 우클릭
② "이 커밋까지 현재 브랜치를 초기화" 선택 후
옵션에서 hard 선택

(3) git pull 활용

① git pull: git fetch+git merge
② git fetch : 원격 저장소, 로컬 저장소 동기화 명령
③ git merge : 브랜치의 내용 동기화 명령

커밋의 내부구조

1. 커밋에 포함된 정보

(1) 커밋 메시지

(2) SHA-1 체크섬 값
- 커밋 고유의 값으로 식별에 사용

(3) 파일 객체(BLOB, Binary Large Object)
- 스테이지에 있던 여러 파일들

(4) 트리 객체
- 여러 파일을 관리하는 데이터 구조

(5) Git 커밋은 일종의 불변 객체
- 새로운 커밋에 이전의 커밋을 포함시켜 연결 가능
- 정보 업데이트 불가

2. 커밋 객체의 내부구조

(1) SHA-1 체크섬

(2) 커밋 메시지

(3) 부모(이전) 커밋에 대한 참조

(4) 트리 객체

(5) BLOB 파일 객체

오늘의 키워드

커밋, Git, git init, git clone,
CLI, 소스트리, git commit, 커밋 메시지,
커밋의 내부구조, 불변 객체

생각할 거리

버전 관리를 위한 커밋 관련 명령어를 알아보고
각 명령어의 동작 원리와 내부 구조를 이해하자!