STEP

디지털 기초

Git 기본 퀵 가이드

1차시 : Git & GitHub 이해하기

Git의 특징, 등장 배경, 필요성

1. Git의 의미

(1) 분산 버전 관리 시스템
-버전(Version)=형상

(2) 버전
- 이전의 것 또는 비슷한 종류의 다른 것들과 약간 다른 형태

2. 수동 버전 관리법

(1) 직접 버전을 관리하는 방법

(2) 문제점

① 덮어쓰기 시 이전 버전으로 복구 불가
② 사용자마다 상이한 관리 방법으로 문제 야기
③ 파일 손상 시 복구 불가

3. 버전 관리 시스템(Version Control System)

(1) VCS 사용 시 장점

① 이전 버전으로 복구 가능
② 프로젝트 전체를 특정 시점으로 변경 가능
③ 시간에 따른 변경 이력 비교 가능
④ 문제 발생 이력 조회 가능

Git의 특징, 등장 배경, 필요성

4. Git의 등장 배경

(1) Linux 창시자 리누스 토발즈 개발팀에 의해 창시
- 용도: 복잡한 리눅스의 소스 관리

(2) Git 이전 소스 코드 관리 방법: Bit Keeper(~2000년대 중반)
- 상용 버전 관리 시스템(VCS)
- 빠른 성능 보유
- 대량의 소스 코드 관리에 적합

(3) Bit Keeper 창립자의 문제 제기
- Linux 개발팀원의 부적절한 Bit Keeper 사용 문제 제기
  →지원 중단→새로운 VCS Git 등장

(4) 의미
- 명령을 뜻하는 'Get'의 오타
- 멍청한 사람을 뜻하는 영국의 속어

Git의 특징, 등장 배경, 필요성

5. 버전 관리 방식의 차이에 따른 분류

(1) 로컬 VCS

① PC에 DB를 설치하여 버전 관리하는 방법으로
사용 및 설치 간단
② 대표적인 로컬 VCS: GNU RCS

(2) 중앙집중식 VCS

① 협업 작업 시 문제 해결에 대한 한계 존재
② 협업을 위해 등장한 중앙집중식 VCS
③ 대표적인 중앙집중식 VCS: Subversion
④ 문제점
-서버 다운 시 이용 불가
-서버의 하드디스크 문제 발생 시 치명적인 데이터 손실 발생

(3) 분산 버전 관리 시스템(DVCS)

① 중앙집중식 VCS의 문제 해결 대책
② 작업 파일을 내 PC에 저장하는 동시에 원격 서버 및 여러 PC에
복사본 저장
③ 대표적인 분산 VCS: Git
④ 장점
- 내 PC에 문제 발생 시 작업 파일의 빠른 복구 가능
- 원격 저장소 복수 관리 가능으로 서버 다운 시 개발, 협업
지속 가능

Git의 특징, 등장 배경, 필요성

6. GitHub

(1) Web을 통해 협업·버전 관리·프로젝트 관리 서비스를 제공하는 온라인 코드 저장소

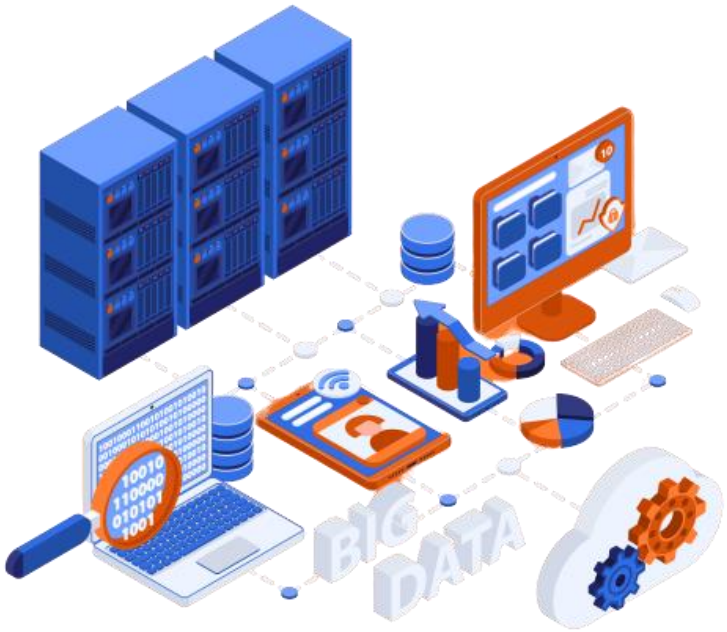
(2) 특징
- AI 기능 추가로 개발자의 필수 도구로 자리매김
- 이전 서비스와 비교 불가한 편리한 인터페이스 제공

(3) GitHub 사용 증가로 Git 대중화

7. Git의 장단점

(1) 장점: 빠른 성능, 높은 가용성

(2) 단점: 매우 높은 입문 난이도



Git & GitHub 사용을 위한 환경 구축

1. Git 관련 도구 사용 시 주의사항

(1) 소셜 로그인 연동 방식
- GitHub 홈페이지(https://github.com)에서 회원 가입 진행
- 인증 오류가 빈번히 발생: 패스워드를 입력하는 곳에 본인의 GitHub 패스워드를 입력해서 생기는 경우

(2) 패스워드 요구 시 토큰 입력
- Settings→Developer settings→Personal access tokens →Fine-grained tokens 또는 Tokens(Classic) 선택
- 입문자의 경우 Classic 토큰 생성을 추천

(3) GitHub 토큰 생성 시 주의사항
- 대부분의 권한 허용
- 클라우드 등 공개된 곳에 업로드 지양
- 분실 및 유출 시 재생성 또는 삭제 가능

(4) 2023. 9. 23 이후 MFA 인증 의무화
- GitHub MFA 인증 방법: Settings→Password and authentication 메뉴에서 MFA 설정

Git & GitHub 사용을 위한 환경 구축

2. Git 사용방법: GUI 도구

(1) GitHub Desktop

① GitHub에서 무료 다운로드 가능
② 간단한 사용법으로 입문자 사용 적합

(2) SourceTree

① 가장 대중적인 Git GUI 클라이언트
② sourcetreeapp.com에서 무료 다운로드 가능
③ 입문자를 위한 기본 명령부터
전문가를 위한 고급 명령까지 수행 가능
④ 복잡한 명령 수행 필요 시
SourceTree 터미널 기능 이용 가능

(3) TortoiseGit

① tortoisegit.org에서 무료 다운로드 가능
② 윈도우 탐색기와 통합되어 사용
③ 직관적인 메뉴로 빠른 작업이 가능해 선호도 높음

Git & GitHub 사용을 위한 환경 구축

3. Git 사용방법: 프로그래밍 에디터 자체 확장 기능

(1) 최신 에디터 사용

① Git, GitHub 지원 기능 제공
② 다양하고 강력한 기능 보유

(2) 에디터를 활용한 Git 사용의 장점

① 빠른 작업 속도
② 프로그램 전환 시간 절약 가능
③ 최근 에디터 사용 트렌드: 하나의 에디터로 개발 외 다양한 작업 처리

(3) VS Code

① 대표적인 에디터 중 하나로 Git 및 다양한 플러그인 지원
② VS Code에 Git Graph 플러그인 설치 시 작업 히스토리 조회 가능

(4) Git CLI 이용 방법

① SourceTree의 터미널 기능 사용
② Git 클라이언트에서 제공하는 명령 창 사용
③ 윈도우의 파워셸, 명령 프롬프트, 윈도우 터미널 사용
④ Git 클라이언트의 Git Bash 사용
(gitforwindows.org에서 다운로드 가능)

Git & GitHub 사용을 위한 환경 구축

4. GUI와 CLI 비교

(1) GUI

① 화면과 버튼을 통해 상호작용

(2) CLI

① 명령 & 명령에 대한 결과값 제시
② CLI가 어려운 이유
- 명령 시 명확한 의도 필요
- 명령에 대한 결과 해석 필요
- 추가 명령 결정 필요
- 오류 발생 시 원인 분석 및 해결책 마련 필요

5. 다양한 Git 사용 방법

(1) Git 도구 3가지를 꾸준히 사용하면 적합한 도구 활용 가능
- GUI 프로그램
- 코드 에디터 기본 기능
- Git Bash를 이용한 CLI

(2) Git 전문가가 되고 싶다면 원리를 파악
- 작업 히스토리에 문제 발생하는 등 복잡한 문제 해결을 위해 Git의 동작 원리 숙지

오늘의 키워드

Git, 버전 관리 시스템(VCS),
로컬 VCS, 중앙집중식 VCS, DVCS,
GUI 도구, GitHub, GUI, CLI

생각할 거리

Git의 특징, 등장 배경, 필요성을 이해하고
Git & GitHub 사용을 위한 환경을 구축해 보자!