GitHub 포트폴리오

1주차. Github 포트폴리오 개요

서울사이버대학교

강의소개

강의 개요 및 목표

강의 개요

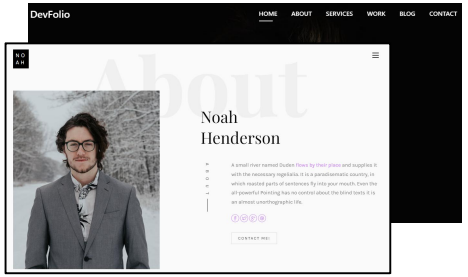
- 과목명 : Github 포트폴리오
- 강의시간 : 75 ~ 90분 (이론 10%, 실습 90%)
- 개요 : - Git을 이용한 프로그램 버전관리, Github를 이용한 포트폴리오 작성법 등 다양한 자원 관리 학습
  - 자기 주도 기반 학습

강의 목표

- Git & Github의 기본 사용법 이해 및 활용
- 프로젝트 관리와 협업 도구로서의 Github 활용 능력 함양
- 비슷한 환경이지만 미묘한 차이에서 발생하는 문제점 해결 역량 함양
- 자신만의 포트폴리오를 작성 및 공개 (평가/피평가)

강의소개

강의 핵심



"Open Certificate of Own Career"

Github 포트폴리오
(https://○○○.github.io)

"Git" knowledge	"Github" knowledge	"Web Development" knowledge	Basic Rules
이론 10% / 실습 90%	이론 20% / 실습 80%	이론 30% / 실습 70%	이론 10%

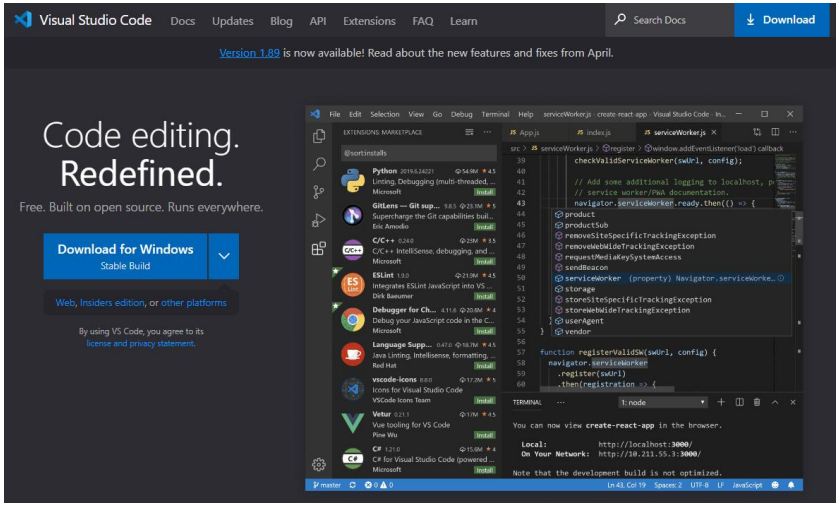
실습 환경

실습 가능한 환경

- 하드웨어 환경 : 데스크탑, 노트북 모두 가능
  (단, 모바일은 실습과 시험이 불가능함)
- 운영체제 환경 : Windows, 리눅스, Mac 등 모두 가능
- 강의에서의 실습 환경
  - Windows 11

개발 도구

포트폴리오 개발을 위한 도구
Visual Studio Code



강의소개

평가 방법

출석	비율	설명
출석	20%	
개인 포트폴리오 평가 (상세내용 필히 참고)	80%	• 학생 상호평가 : 20% • 교수 평가 : 60%

학교 LMS 또는 디스코드를 이용하여 질문을 하시면 학습에 도움이 되도록
답변을 꼭 남겨드리도록 할 예정임(디스코드 초대코드는 공지사항에 별도 안내)

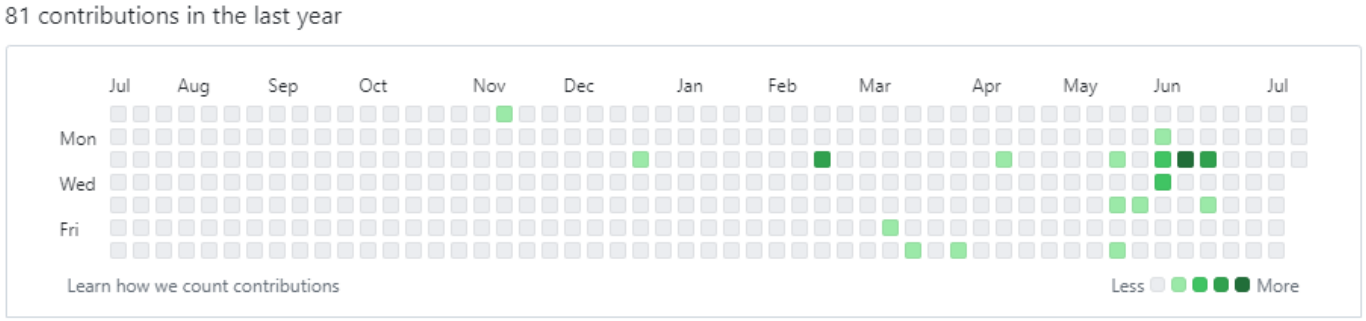
강의소개

GitHub

평가 방법

ⓘ 출석 : 20%

- 학교 LMS 접속 후 강의 시청
- Github 사이트 활동 정보 (필요시 별도 출석부 제출)



학교 LMS 출석 규정 반영

색상이 표시가 되지 않는 경우 : 결석처리

강의소개

평가 방법

ⓘ 학생 상호평가 : 20%(루브릭 평가)

- 평가대상
  - 공개된 학생들의 포트폴리오
  - 공개 방법 : 깃헙 URL, 캡쳐+설명이 포함된 PPT파일 또는 유튜브 등에 등록 후 URL을 제출 게시판에 등록
  - 공개 및 평가시기 : 9주차 ~ 15주차말까지
- 평가 방법
  - 제공되는 루브릭 평가표 작성 및 제출
  - 5명 이상 평가
    (본인 포함, 평가인원에 따라 가감)

기타 자세한 일정, 제출방법 등은 별도 공지 예정

평가 영역	평가 항목	우수 (5)	양호 (4)	보통 (3)	미흡 (2)	부족 (1)
기획력	프로젝트 계획	프로젝트 목적과 계획이 명확하며, 타임라인과 목표가 잘 정의됨.	프로젝트 목적과 계획이 명확하며, 타임라인과 목표가 대부분 정의됨.	프로젝트 목적과 계획이 명확하지만, 타임라인과 목표가 일부 정의되지 않음.	프로젝트 목적과 계획이 불명확하며, 타임라인과 목표가 대부분 정의되지 않음.	프로젝트 목적과 계획이 거의 없으며, 타임라인과 목표가 정의되지 않음.
기술 이해력	Github Pages 설정 및 배포	Github Pages가 완벽하게 설정되고, 문제 없이 배포됨. URL이 작동하며, 사이트가 기대한 대로 작동함.	Github Pages가 설정되었고, 대부분 문제 없이 배포됨. URL이 작동하며, 사이트가 거의 기대한 대로 작동함.	Github Pages가 설정되었으나, 일부 배포 문제나 작동 오류가 있음.	Github Pages 설정 및 배포 과정에서 여러 문제가 발생하여 사이트가 제대로 작동하지 않음.	Github Pages 설정 및 배포가 완료되지 않았거나, 사이트가 전혀 작동하지 않음.
	버전 관리	Git과 Github를 활용한 버전 관리가 매우 우수하며, 모든 커밋 메시지와 Pull Request가 명확함.	Git과 Github를 활용한 버전 관리가 우수하며, 대부분의 커밋 메시지와 Pull Request가 명확함.	Git과 Github를 활용한 버전 관리가 평균적이며, 일부 커밋 메시지와 Pull Request가 명확함.	Git과 Github를 활용한 버전 관리가 미흡하며, 커밋 메시지와 Pull Request가 대부분 명확하지 않음.	Git과 Github를 활용한 버전 관리가 매우 부족하며, 커밋 메시지와 Pull Request가 전혀 명확하지 않음.
표현 내용과 방법	콘텐츠의 질	포트폴리오 내용이 매우 명확하고, 잘 구성되어 있으며, 다양한 프로젝트와 코드 예제가 포함됨.	포트폴리오 내용이 명확하고, 잘 구성되어 있으며, 다양한 프로젝트와 코드 예제가 일부 포함됨.	포트폴리오 내용이 어느 정도 명확하며, 구성도 적절하나, 프로젝트와 코드 예제가 제한적임.	포트폴리오 내용이 불명확하며, 구성이 부족하고, 프로젝트와 코드 예제가 거의 없음.	포트폴리오 내용이 매우 부족하며, 구성도 부실하고, 프로젝트와 코드 예제가 전혀 없음.
	디자인 및 사용자 경험	사이트 디자인이 매우 우수하고, 사용자 경험이 훌륭함. 모든 요소가 일관되게 배치됨.	사이트 디자인이 우수하고, 사용자 경험이 좋음. 대부분의 요소가 일관되게 배치됨.	사이트 디자인이 평균적이며, 사용자 경험도 평균 수준. 일부 요소의 일관성이 부족함.	사이트 디자인이 미흡하며, 사용자 경험이 좋지 않음. 여러 요소의 일관성이 부족함.	사이트 디자인이 매우 부족하며, 사용자 경험이 나쁨. 대부분의 요소의 일관성이 없음.
발표력(PR)	발표 및 피드백	발표 내용이 매우 명확하고, 피드백을 잘 반영하여 개선함.	발표 내용이 명확하고, 피드백을 대부분 반영하여 개선함.	발표 내용이 어느 정도 명확하며, 피드백을 일부 반영함.	발표 내용이 불명확하며, 피드백을 거의 반영하지 않음.	발표 내용이 매우 부족하며, 피드백을 전혀 반영하지 않음.

강의소개

평가 방법

ⓘ 교수 평가 : 60%(루브릭 평가)

- 평가 대상 : 전체 학생
- 평가 방법 : 루브릭 평가표 작성
- 평가 주요 사항
  - 포트폴리오 기획 : 8주차말까지 제출
    * 기 제공되는 포트폴리오 테마, 자신이 발견한 테마, 개발한 테마 등에서 템플릿 선정
    * 선정된 템플릿 캡쳐본 (또는 드로잉) 및 선정사유 작성 제출
  - 포트폴리오 보완 : 9주차 ~ 14주차말까지 (1차 평가 및 피드백)
    * 1차 평가 루브릭의 피드백을 보고 보완
  - 포트폴리오 결과 : 15주차말까지 (2차 평가)

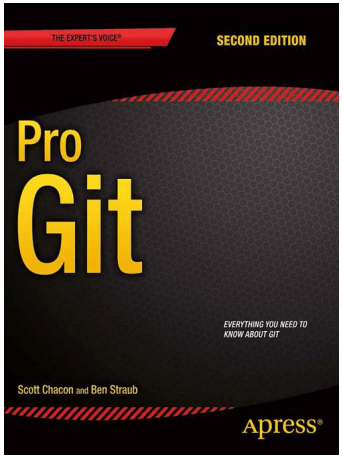
※ 일정이 준수되지 않아 기한이 초과되는 경우 감점 예정

강의소개

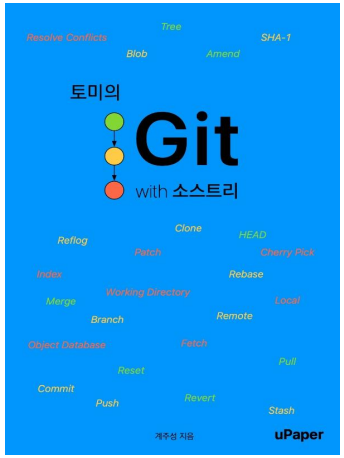
강의 교재

Lecture Note

참고 교재



무료 다운로드 가능
https://git-scm.com/book/ko/v2



토미의 Git with 소스트리

무료 eBook
https://www.yes24.com/Product/Goods/119420040
https://ebook-product.kyobobook.co.kr/dig/epd/ebook/E000005246582

강의에 앞서...

GitHub

실습 과목	반드시 PC, 노트북, 테블릿 등의 실습 가능한 환경 필요
공개용 S/W 학습 과목	강의 내용은 언제든 변경될 수 있음 (공지사항 필독)
Backend 개념의 이해가 필요한 과목	탄생 배경, 사용 이유, 그리고 공감대
결과물이 있는 과목	나만의 포트폴리오 제작 (필수)
지식 저장소를 만드는 과목	나의 지식을 공유하고, 다른 사람의 지식을 배움
Overlap 강의	각 주차별 강의 내용이 일부 겹치도록 구성하여 이해력 제고

학습개요

학습목표

01. 버전관리가 무엇인지 설명할 수 있다.

02. Git과 Github의 기본 개념 및 그 차이점을 설명할 수 있다.

03. Github 계정을 생성 및 기본 환경 설정할 수 있다.

학습내용

- 버전관리
- Git과 Github
- Github 준비 및 실습

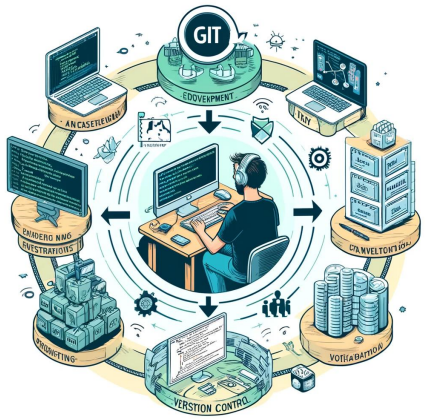
학습에 앞서…

버전관리 이해

Git과 Github 개념과 차이점

Github 계정 생성과 기본 설정

- 우리가 배우는 Git은 기본적으로 “프로그램 개발”이라는 간단하지만 쉽지 않은 과제를 해결하기 위하여 탄생하게 됨
  - 프로그램 개발 : 분석, 설계, 개발, 테스트, 다시 개발, 테스트 또는 다시 분석…
- 어떠한 이유나 환경에서든 한번에 끝나는 프로그램은 없으며, 지속적인 수정 개발과 테스트가 발생하는 것이 “프로그램 개발” 세계임



학습에 앞서...

GitHub

버전관리 이해

Git과 Github 개념과 차이점

Github 계정 생성과 기본 설정

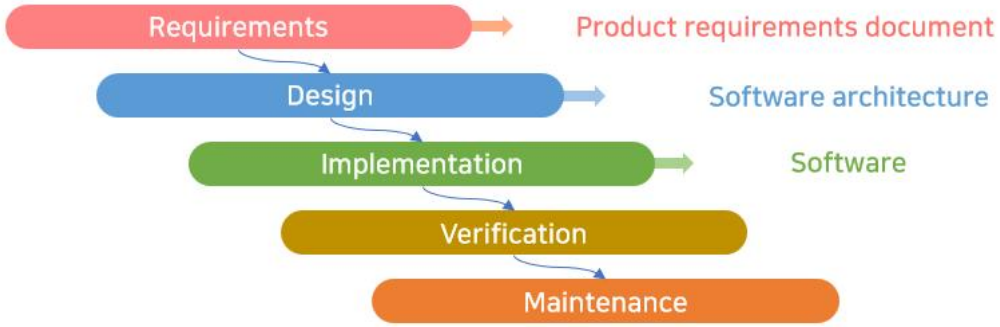
- Git은 이러한 프로그램 개발의 산출물인 “프로그램 소스”를 개발자의 상황에 맞춰 “버전”이라는 이름으로 보관하고 사용할 수 있도록 지원하는 도구임
- 그리고 Github는 이러한 Git 사용 개발자를 위해 만들어진 효율과 공유를 추구하는 플랫폼임

버전관리

UNIT

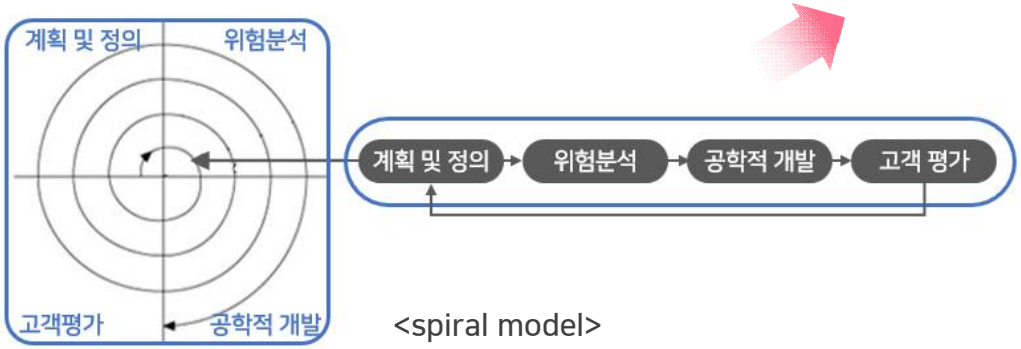
프로그램 개발과 버전

일반적인 프로그램 개발 과정(소프트웨어 개발 프로세스, SDP, SDLC)



<waterfall model (sashimi)>

여러 개의 프로그램 소스 생산



<spiral model>

프로그램 개발과 버전

GitHub

버전(Version) 이란 ?

버전(Version)	소프트웨어나 문서 등의 특정 상태를 구별하는 고유한 식별자

- 버전형식 : a.b.c
  - a : major 숫자로 큰 변화 발생시 부여
  - b : minor 숫자로 기능 추가나 개선 발생시 부여
  - c : patch 숫자로 버그 수정 발생시 부여

버전 부여는 S/W 개발자가 의미를 부여할 수 있는 자유도가 높은 정보임

버전을 관리한다 ?

> 위와 같은 버전형식으로 파일을 만들어 내기 위한 과정

> 만들어진 다양한 버전의 파일을 복구에 활용하거나 변경 내용을 비교하는데 활용

프로그램 개발과 버전

GitHub

버전 관리 시스템 (VCS, Version Control System)

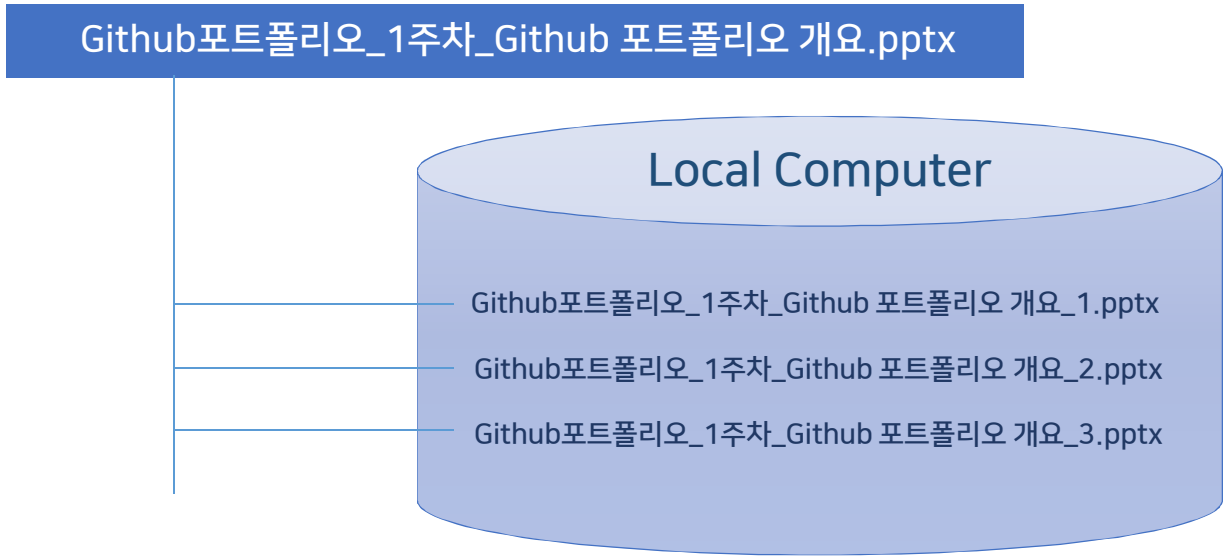
버전 관리 시스템 (VCS)	파일 변화를 시간에 따라 기록했다가 나중에 특정 시점의 버전을 다시 꺼내올 수 있는 시스템

- 컴퓨터를 이용한 거의 모든 작업에 부합하는 시스템
  (not necessary, but must have)
- 장점
  - 파일을 이전 상태로 되돌릴 수 있다.
  - 프로젝트를 통째로 이전 상태로 되돌릴 수 있다.
  - 시간에 따라 수정 내용을 비교해 볼 수 있다.
  - 누가 문제를 일으켰는지도 추적할 수 있다.
  - 누가 언제 만들어낸 이슈인지도 알 수 있다.
  - 파일을 잃어버리거나 잘못 고쳤을 때도 쉽게 복구할 수 있다.

고전적인 버전관리

로컬 버전 관리 – 복제

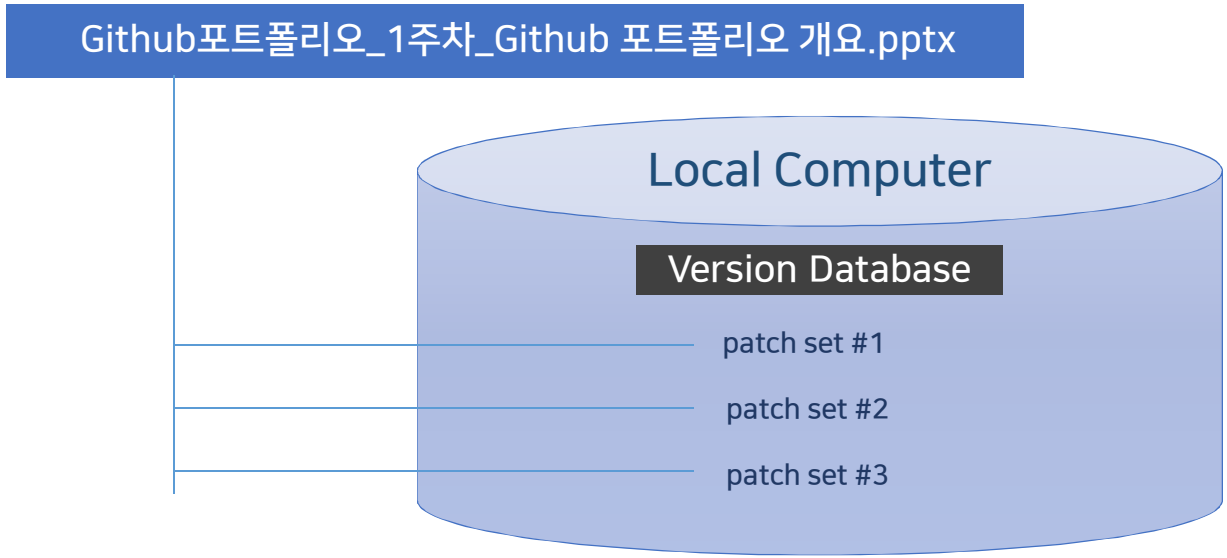
- 디렉토리 또는 파일을 복제 & 복제결과에 날짜를 기록



고전적인 버전관리

로컬 버전 관리 - RCS, Revision Control System

- 파일의 변경되는 부분(patch set)만 데이터베이스에 이용하여 기록



Git과 Github

UNIT

Git의 정의

GitHub

Git 이란 ?

Git

컴퓨터 파일의 변경사항을 저장, 추적하고 다수 사용자들 간에 해당 파일들의 작업을 조율하기 위한 스냅샷 스트림 기반의 분산 버전 관리 시스템이다

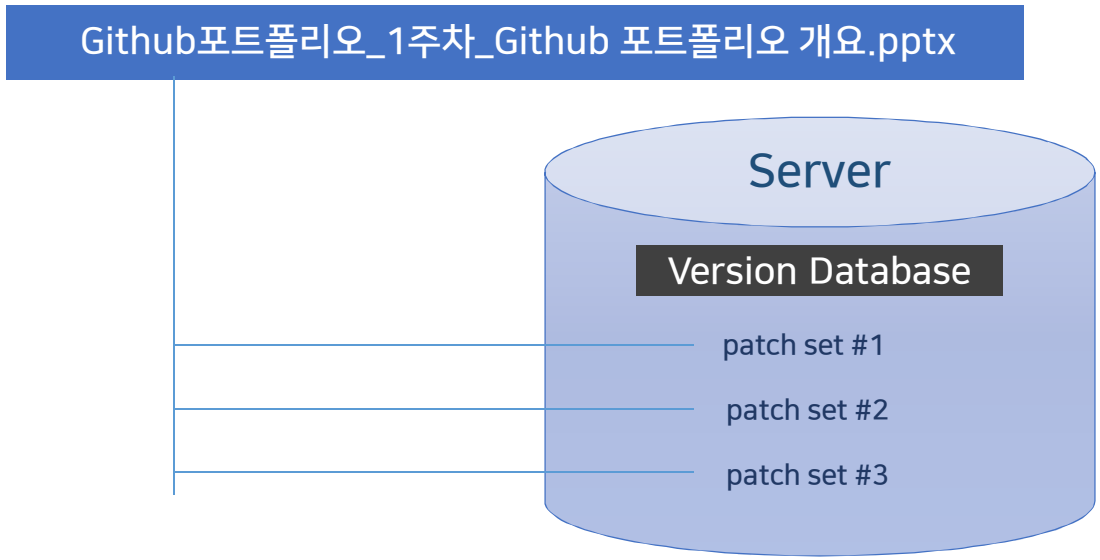
- 스냅샷 스트림
  - 연속된 특정 시점에서 파일, 폴더 또는 워크스페이스의 상태 정보
- 분산 버전 관리 시스템
  - 분산(Distributed) : 흩어진
  - 버전 관리 (Version Control) : 컴퓨터 파일의 변경 이력을 관리
  - 시스템 (System) : 체계적인

결국, Git은 컴퓨터 파일의 변경 이력을 흩어지게 관리하는 체계
(프로그램 등의 소스 코드 관리를 위한 분산 버전 관리 시스템)

Git의 정의

현대적인 버전 관리 (중앙 집중식 버전 관리 - CVCS, Central Version Control System)

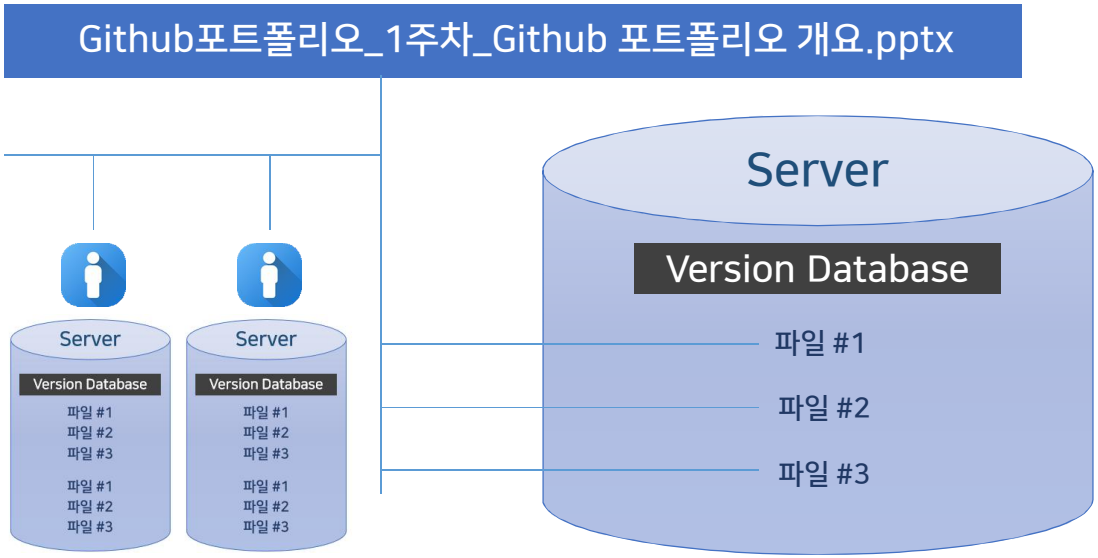
- 고전적인 버전관리 방법을 서버에 기록
- Checkin, Checkout 개념 등장



Git의 정의

현대적인 버전 관리 (분산 버전 관리 – DVCS, Distributed Version Control System)

- 고전적인 버전관리 방법을 로컬과 서버에 기록
- Init, Add, Commit, Push, Pull, Clone, Checkout, Merge 개념 등장
- 저장공간 문제 회피를 위해 마지막 커밋에서 스냅샷을 저장, 이전에는 스냅샷 차이를 저장



Git의 정의

GitHub

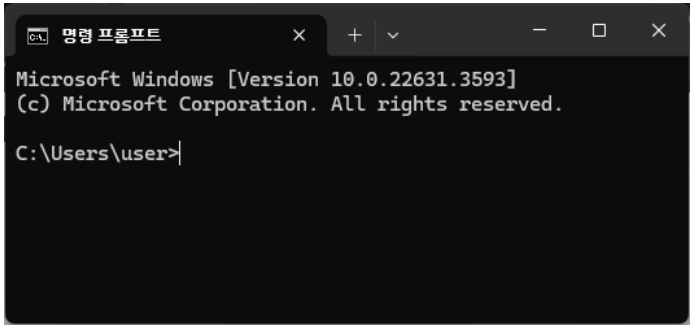
특징과 장점

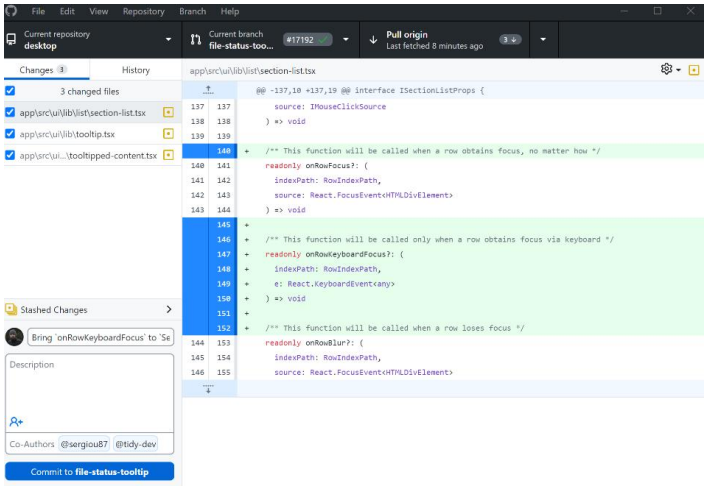
- 거의 모든 명령을 로컬에서 실행
  ➡ 네트워크 환경에 영향을 거의 받지 않음(오프라인 작업도 가능)
- 저장 전에 체크섬으로 데이터를 관리함(SHA-1 해시)
- Git 데이터베이스는 되돌리거나 삭제할 방법이 없음
- 세가지 상태 (★ 아주 중요)
  - Committed : 데이터가 로컬 데이터베이스에 안전하게 저장되었음
  - Modified : 수정한 파일을 아직 로컬 데이터베이스에 커밋하지 않았음
  - Staged : 현재 수정한 파일을 곧 커밋할 것이라고 표시했음
- 혼자 개발하겠다면 ? Git 불필요

Git의 정의

사용방법

- CLI vs GUI
- CLI : Command Line Interface
- GUI : Graphic User Interface





Github의 정의

Github 이란 ?

Github	Git 버전 관리 시스템을 사용하여 소스 코드 관리를 돕는 웹 호스팅 서비스 중 하나

Github 주요 기능

- repository : 프로젝트 소스 코드를 호스팅하고 버전 관리하는 공간
  - Issue tracker : 버그 추적, 기능 요청 등을 관리
  - pull request : 코드 변경 사항을 제안하고 코드 리뷰를 거쳐 병합
- collaboration : 여러 개발자가 동일한 프로젝트에서 작업하고 변경사항을 공유
- projects : 협업을 위한 공간 및 일정 등 다양한 프로젝트 관리

Git & Github 차이점

Github 이란 ?

- Git : 분산 버전 관리 시스템
- Github : Git 저장소를 호스팅하는 웹 서비스
- 차이점

구분	Git	Github
목적	로컬에서 버전 관리 도구로 사용	원격 저장소를 제공하고 협업을 지원
저장소 위치	개발자 로컬 시스템 (PC, 서버 등)	원격 웹 서버에 호스팅
협업	로컬 브랜치로 작업하므로 협업에 제한적	코드 공유, 이슈 트래킹, 코드 리뷰 등의 협업이 가능
기능	버전 관리에 초점을 맞춘 핵심 기능을 제공	프로젝트 관리, 버그 트래킹 등 부가 기능

Git은 버전 관리 도구이고, GitHub는 Git 저장소를 호스팅하는 서비스

Github 준비

UNIT

계정 생성과 기본 설정

계정 생성 방법

Git : 계정 불필요 & 다운로드 후 설치

- https://git-scm.com/downloads
- 지원 OS : macOS, Windows, Linux/Unix
- 과목 실습 환경 : Windows 11

Github : GitHub 서비스에 가입하여 계정 생성

- https://github.com/
- 가입 전 준비사항 : 이메일, 프로필 사진

계정 생성과 기본 설정

기본 설정

SSH 키 생성 및 등록 :
로컬 컴퓨터에서 SSH 키를 생성하고 Github 계정에 등록

- 보안 강화 : 사용자 계정/암호 ➡ SSH키를 사용
- 인증 프록세스 간소화 : 로그인할 때 마다 인증 ➡ 자동 인증
- 팀 프로젝트에서 SSH 키를 이용하여 개발자 액세스 제어 가능
- 여러 계정 관리 : 하나의 컴퓨터에서 서로 다른 SSH 키를 생성하여
  여러 개의 Github 계정 관리

단, 로컬 컴퓨터에서 git을 단독으로 사용한다면 SSH 키 사용은 선택적 필요

계정 생성과 기본 설정

GitHub

기본 설정

환경 설정 :
git config 명령어를 이용하여 사용자 정보 설정

- 사용자 이름 설정 : git config --global user.name <영문 이름>
- 사용자 이메일 주소 설정 : git config --global user.email <메일주소>

> "--global" 와 같이 전역설정하는 경우는 한번만 하면 됨
(프로젝트마다 다른 경우 제거)

> 설정 확인 : git config --list

Github 화면 메뉴

메인화면 (not dashboard)