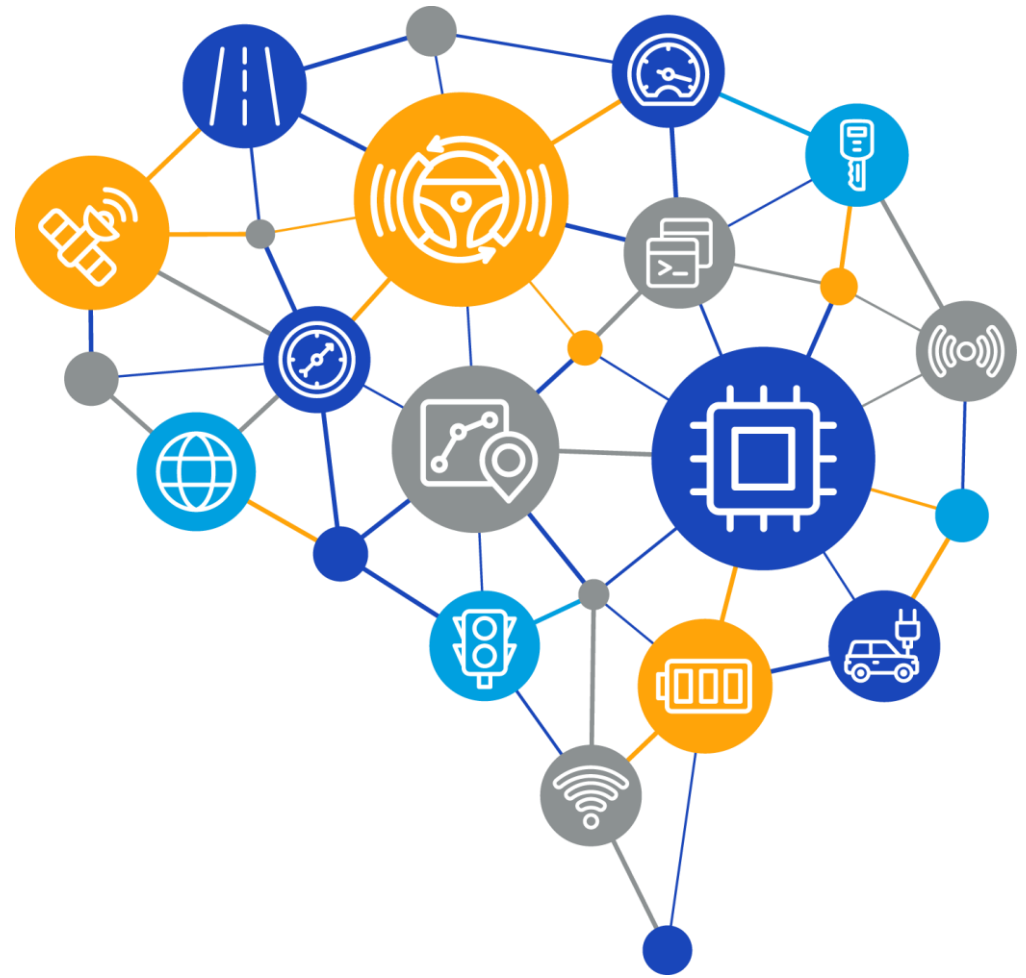


버전관리와 깃과 깃허브 개요

Version Control and
Git & Github basics

2022.03

강환수 교수



AI Experts Who Lead The Future

CONTENTS

01 | 버전관리 이해

02 | 깃 소개

03 | 깃허브 소개

04 | 깃허브 계정과 저장소 생성








AI Experts
Who Lead
The Future

01

버전관리 이해



- 이름으로 관리의 문제

Name
 120525_문서_업데이트.txt
 120604_문서.txt
 120605_문서_수정판.txt
 120605_문서_수정판2.txt
 120605_문서_최신 복사.txt
 120605_문서_최신.txt
 120605_문서.txt
 1200602_문서.txt
 문서_회의용.txt



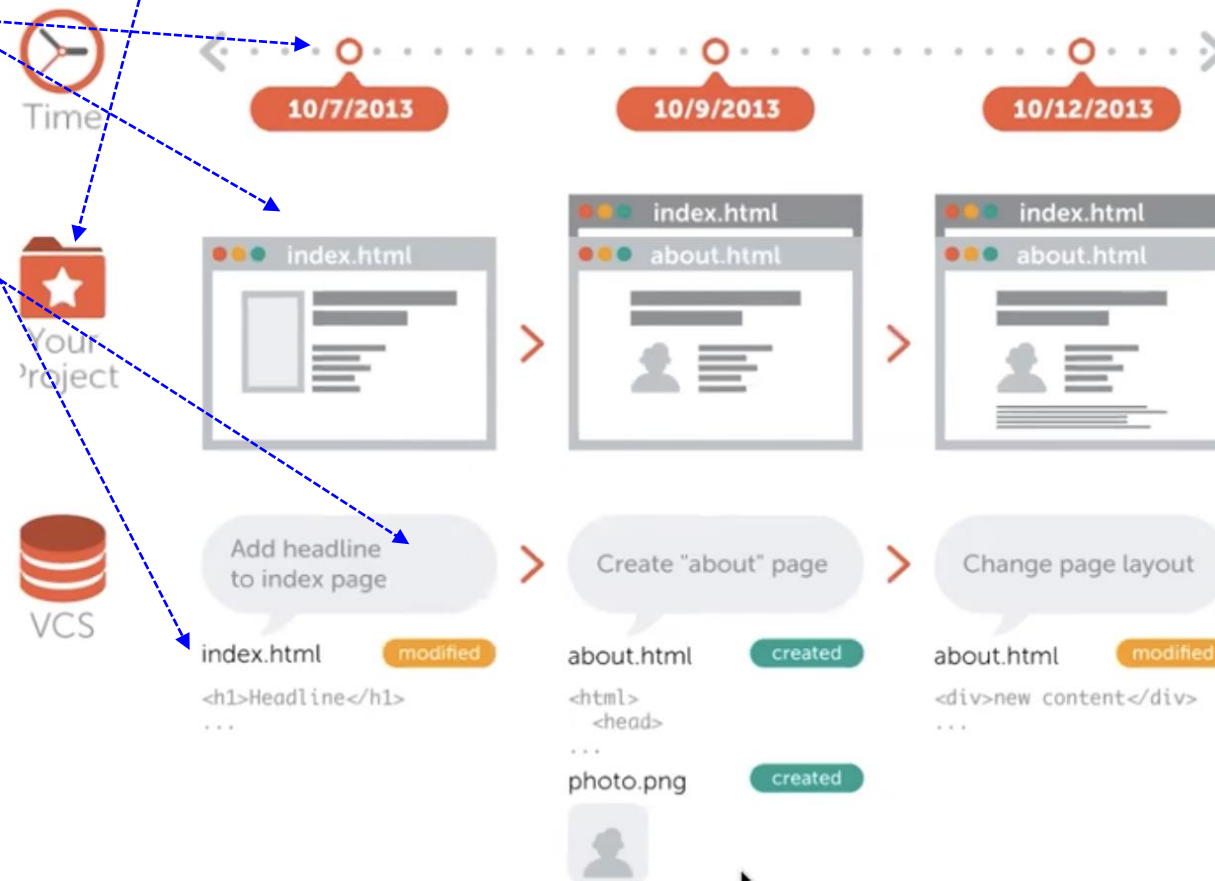
- 파일 버전 관리

- 파일의 수정 이력 관리
 - 언제, 무엇을

- 파일의 추가 및 수정 이력(추적) 관리

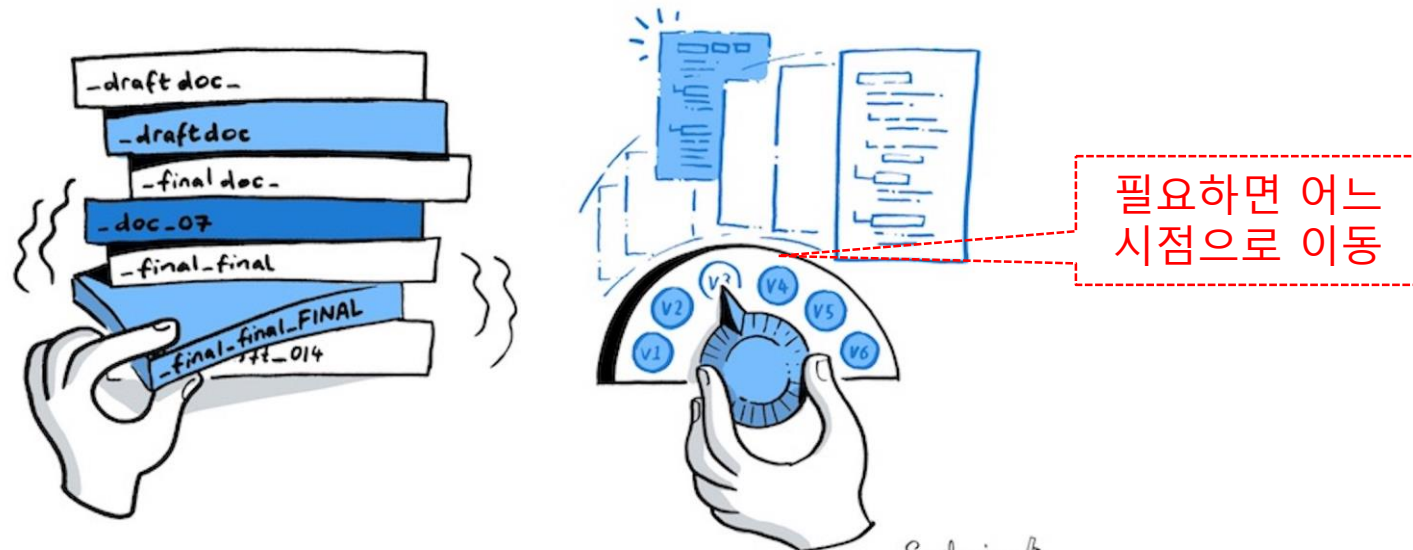
- 누가(다중 사용자)
- 저장소가 여러 개인 경우(어느 저장소에서)
- 어느 파일을
- 언제
- 추가
- 수정

- 내용



- 시간 흐름에 따라 파일 또는 파일 집합에 대한 변경 사항을 추적, 관리
 - the version control system tracks changes to a file or set of files over time
- 버전의 저장과 백업
 - 필요하면 이전의 버전으로 되돌림
 - 변경사항의 자세한 확인
- 여러 사용자에게 버전 이력 추적관리
 - 소스 내용의 충돌에 대한 처리도 필요
 - 어떤 파일이 언제 어떻게 삭제되고 추가 됐는지 확인이 가능

TRACK PROJECT HISTORY

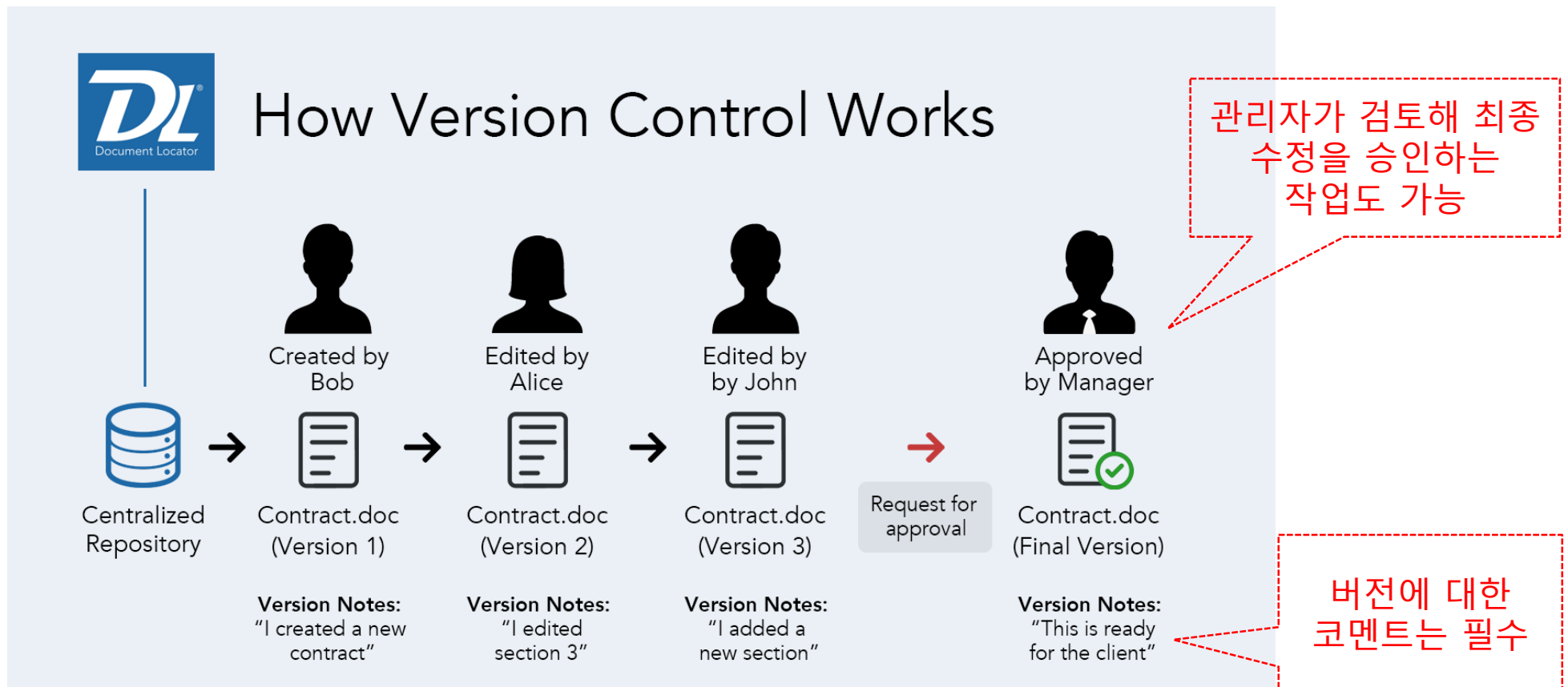


• 버전 관리

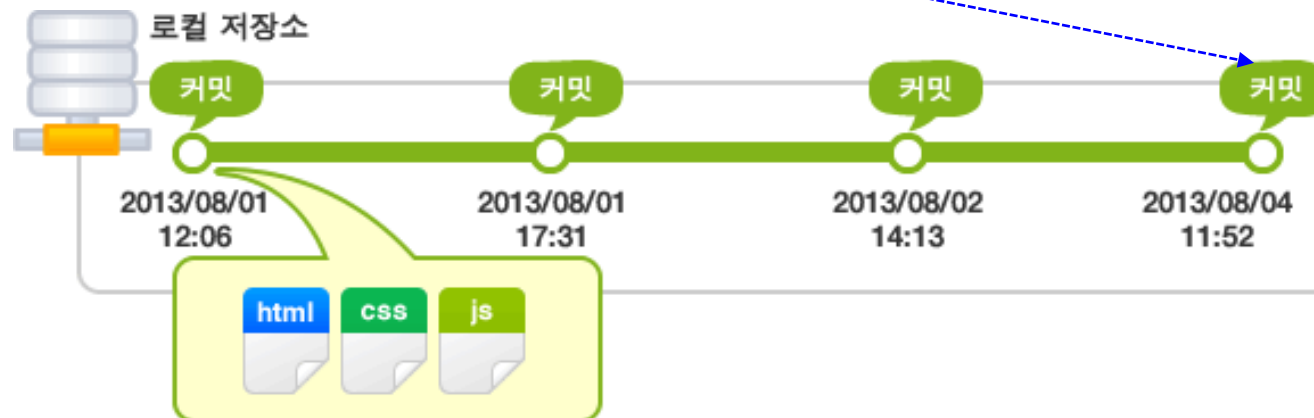
- 파일 또는 파일 집합에 시간 경과에 따른 변경사항을 기록하는 접근방식

- 사용자와 공동작업자가 기록을 추적하고
- 변경사항을 검토하고
- 이전 버전으로 되돌리거나 돌아갈 수 있도록

Version control is an approach to record changes made in a file or set of files over time so that you and your collaborators can track their history, review any changes, and revert or go back to earlier versions.



- 저장소의 현 상태를 담은 스냅샷 사진을 찍는 것에 비유
- 커밋(commit)
 - 저장소의 변경을 기록하는 작업
 - 어느 시점의 파일 및 폴더의 추가/변경 사항을 저장소에 기록
 - 이전 커밋 상태부터 현재 상태까지의 변경 이력이 기록된 커밋이 생성
 - 시간순으로 저장
 - 최근 커밋부터 거슬러 올라가면 과거 변경 이력과 내용을 알 수 있음



- 커밋
 - 프로젝트의 상태를 저장
- 저장소 관리
 - 스냅샷의 연속(연속된 커밋)으로 관리
 - 파일이 달라지지 않았으면 파일을 새로 저장하지 않음
 - 단지 이전 상태의 파일에 대한 링크만 저장



그림 5. 시간순으로 프로젝트의 스냅샷을 저장.

- 저장소(Git repository)

- 파일이나 폴더를 저장해 두는 곳
 - 파일이 변경 이력 별로 구분되어 저장
 - 비슷한 파일이라도 실제 내용 일부 문구가 서로 다르면 다른 파일로 인식하기 때문에 파일을 변경 사항 별로 구분해 저장

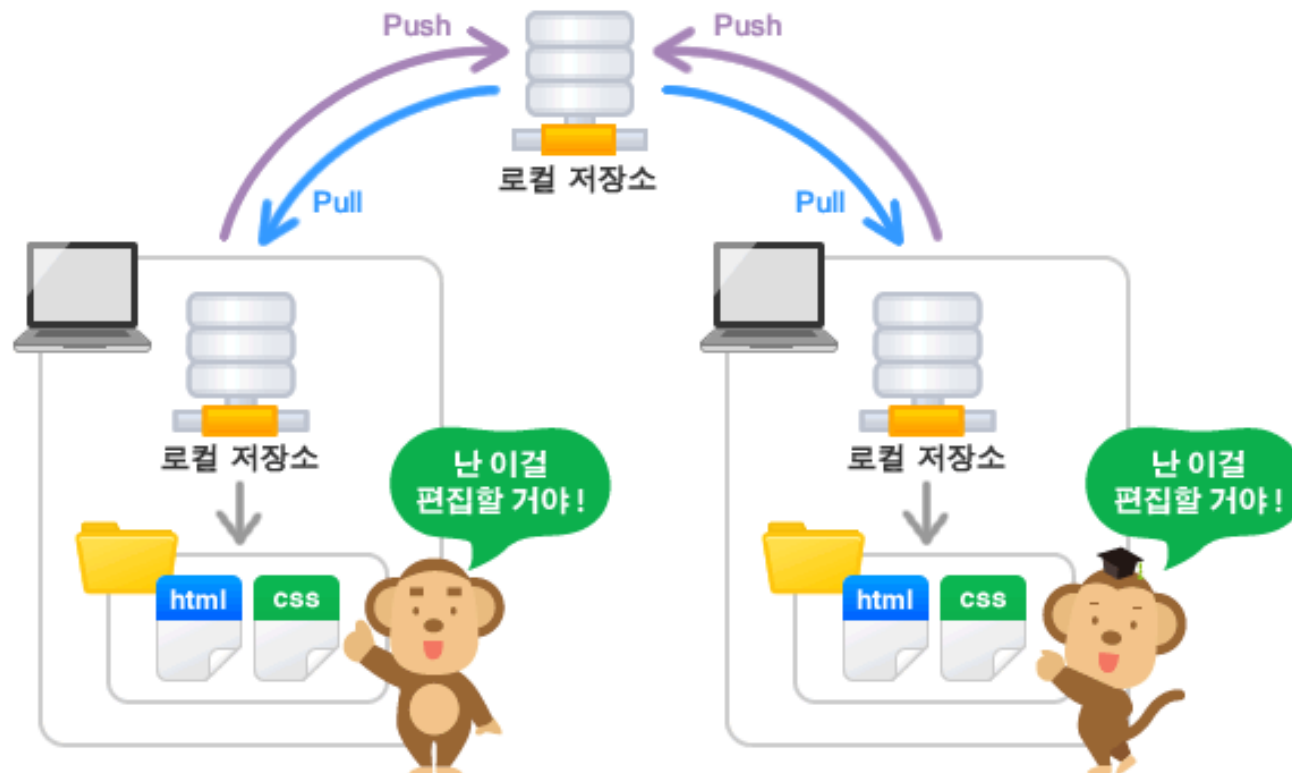


- 일반적으로 두 종류의 저장소를 제공

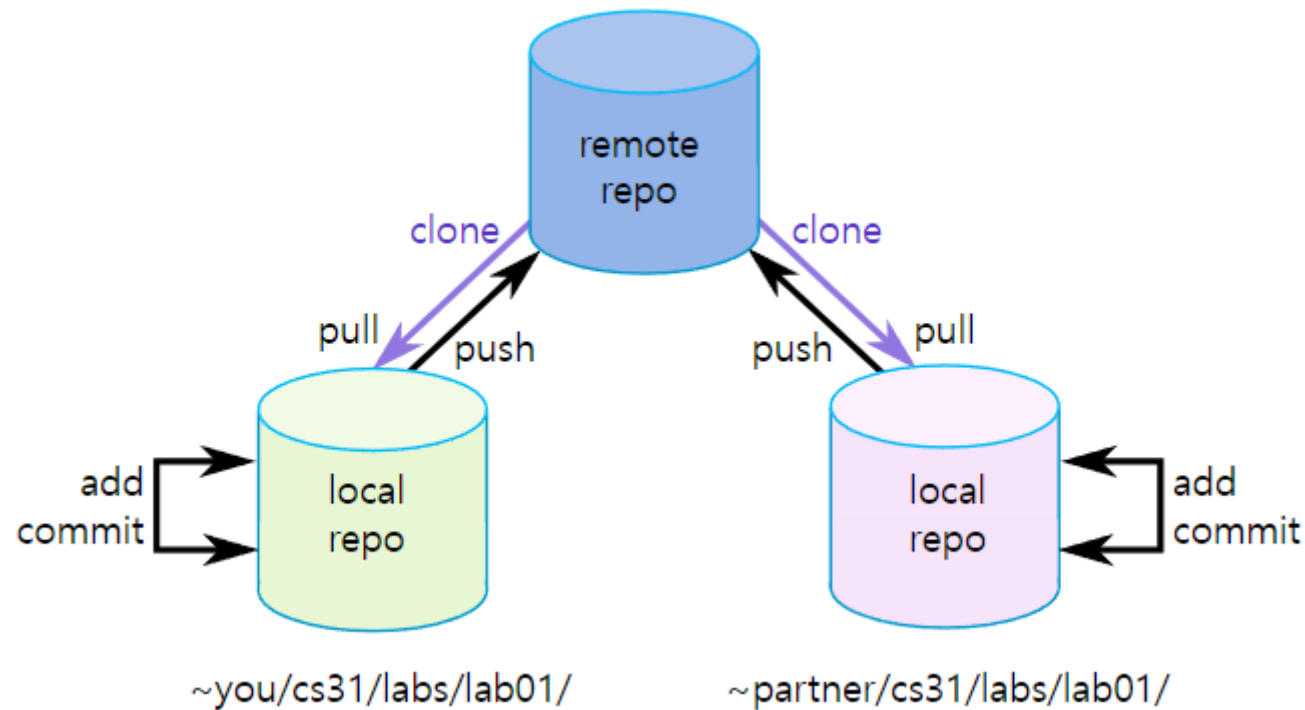
- 원격 저장소(Remote Repository)
 - 파일이 원격 저장소 전용 서버에서 관리되며 여러 사람이 함께 공유하기 위한 저장소
- 지역(로컬) 저장소(Local Repository)
 - 내 PC에 파일이 저장되는 개인 전용 저장소

- Push와 pull 활용

- Push: 원격저장소 올리기
 - 평소에는 내 PC의 로컬 저장소에서 작업하다가
 - 작업한 내용을 공개하고 싶을 때에 원격 저장소에 업로드(push)
- Pull: 원격 저장소에서 지역 저장소로 내리기
 - 물론 원격 저장소에서 다른 사람이 작업한 파일을
 - 로컬 저장소로 가져(pull)올 수도 있음

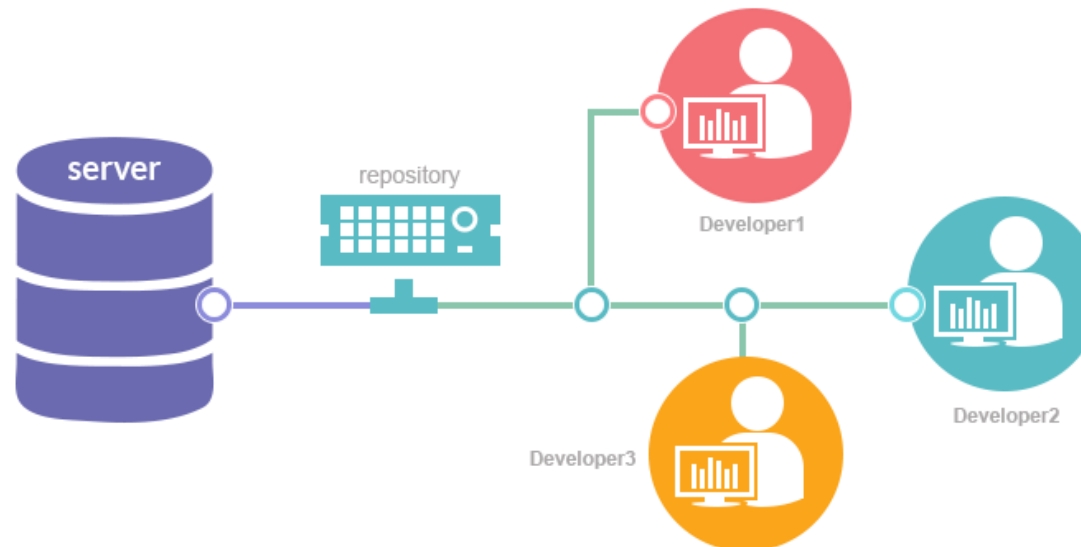


- **Push**
 - 서버로 올리기
- **Pull**
 - 지역 저장소에 내리기
- **파일을 저장소에 저장**
 - Add
 - commit



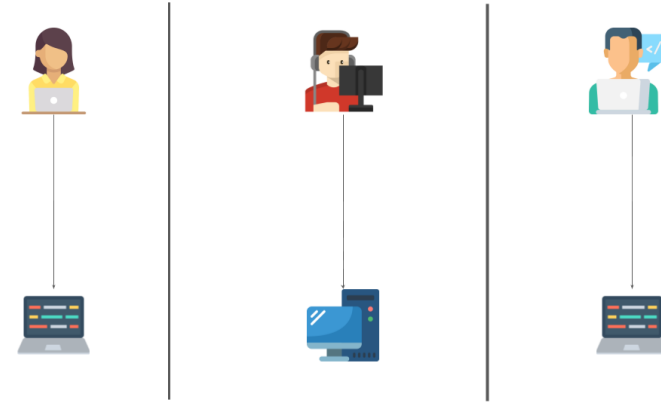
• 프로젝트 개발 시 버전 관리의 필요성

- 개발하는 동안 소스 코드의 변경 사항을 보존하기 위해
 - 버그 및 문제점이 발생했을 때 추적에 유용
 - 과거 특정 시점의 소스 파일 및 디렉토리의 내용을 손쉽게 확인 가능
 - 과거 특정 시점의 소스 파일로 손쉽게 되돌릴 수 있음
- 협동 작업을 가능하게 하기 위해
 - 대부분의 프로젝트는 팀 단위
 - 전체 팀원이 하나의 소스를 가지고 효율적으로 작업할 수 있는 도구가 필요
- 오픈 소스 소프트웨어(OSS) 프로젝트에 필수
 - 인터넷을 이용한 전세계 개발자들이 협업



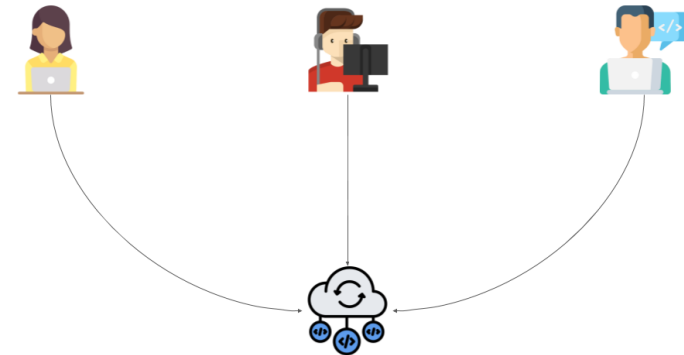
- 로컬 버전 관리

- 다중 사용자 불가
- 원격 서버가 없음



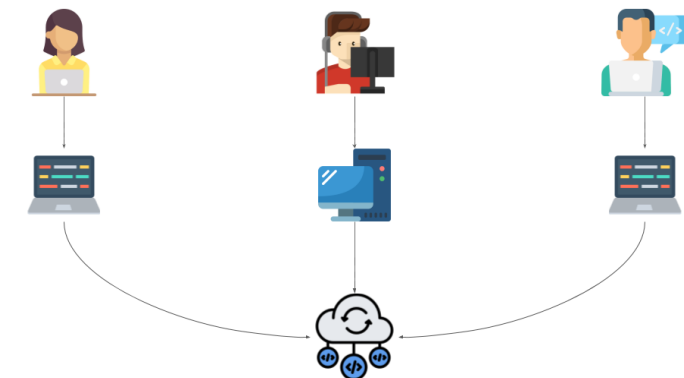
- 중앙 집중 버전 관리

- 다중 사용자 가능
- 원격 서버
 - 원격 서버 문제 시 복구 불가능

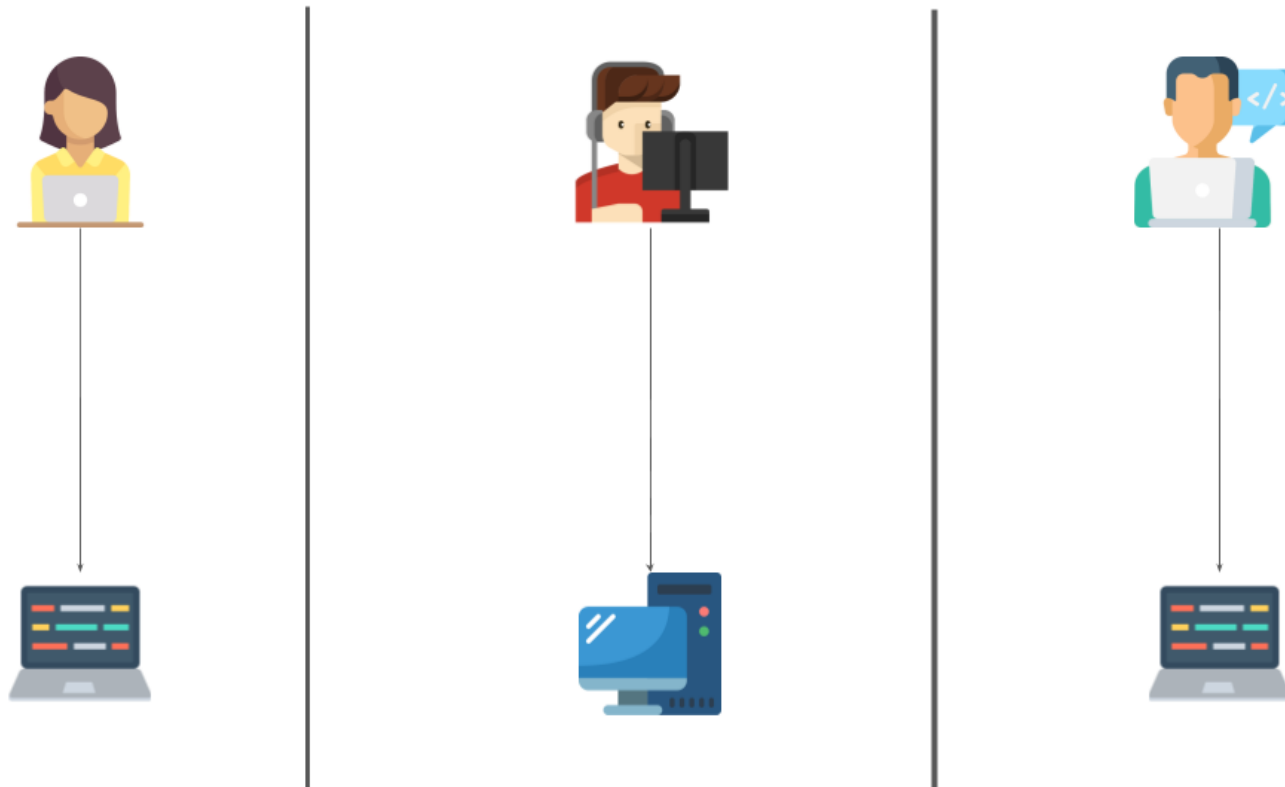


- 분산 버전 관리

- 다중 사용자 가능
- 원격 서버
 - 원격 서버 문제 시 복구 가능
- 주로 사용되는 유형



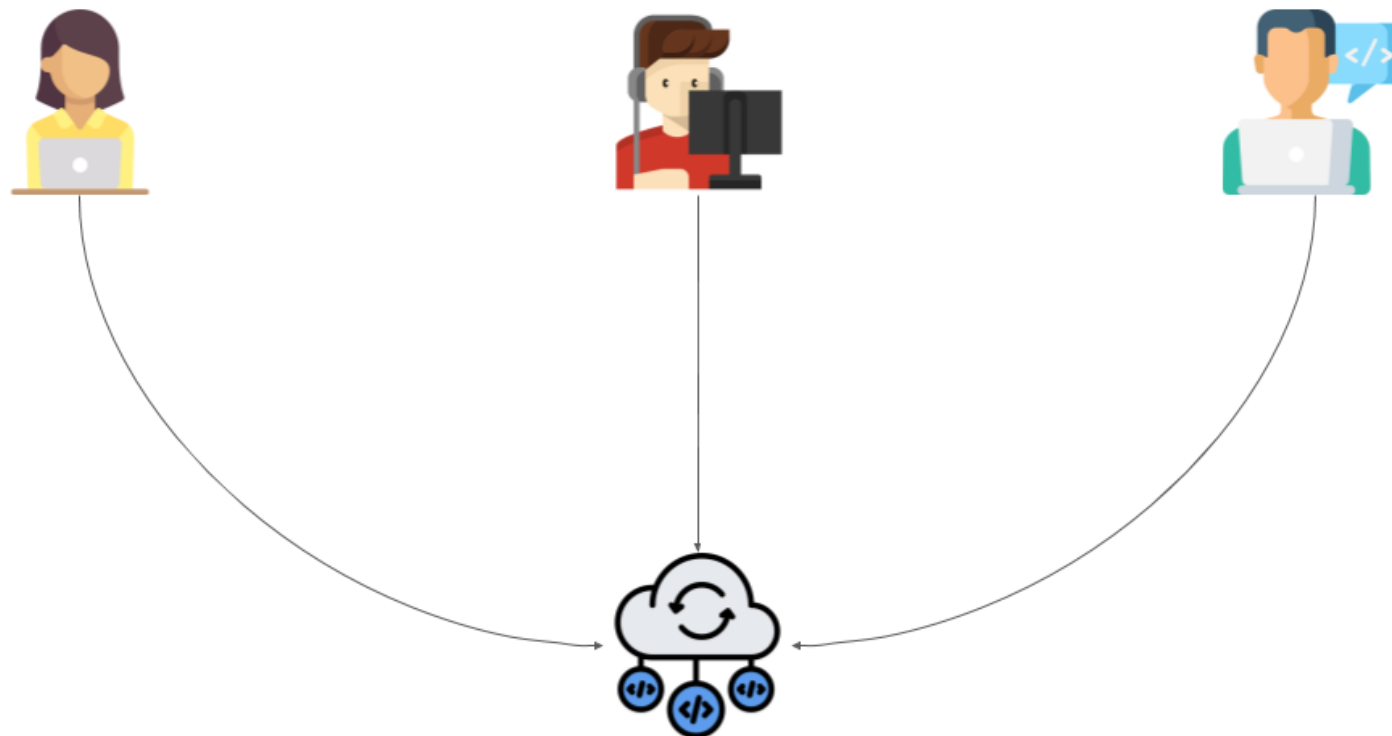
- 원격 저장소가 없는 버전관리
 - 로컬 시스템 내에서만 모든 파일을 관리하고 버전을 지정
 - 모든 변경 사항은 로컬 데이터베이스에 기록
 - 모든 개발자는 자신의 컴퓨터를 가지고 있으며 아무 것도 공유하지 않음
 - 원격 저장소에 따로 복사본 개념이 없음
 - 자신의 저장소의 문제에 복구 불가능



2. 중앙 집중 버전 관리 시스템

깃과 깃허브 Python language

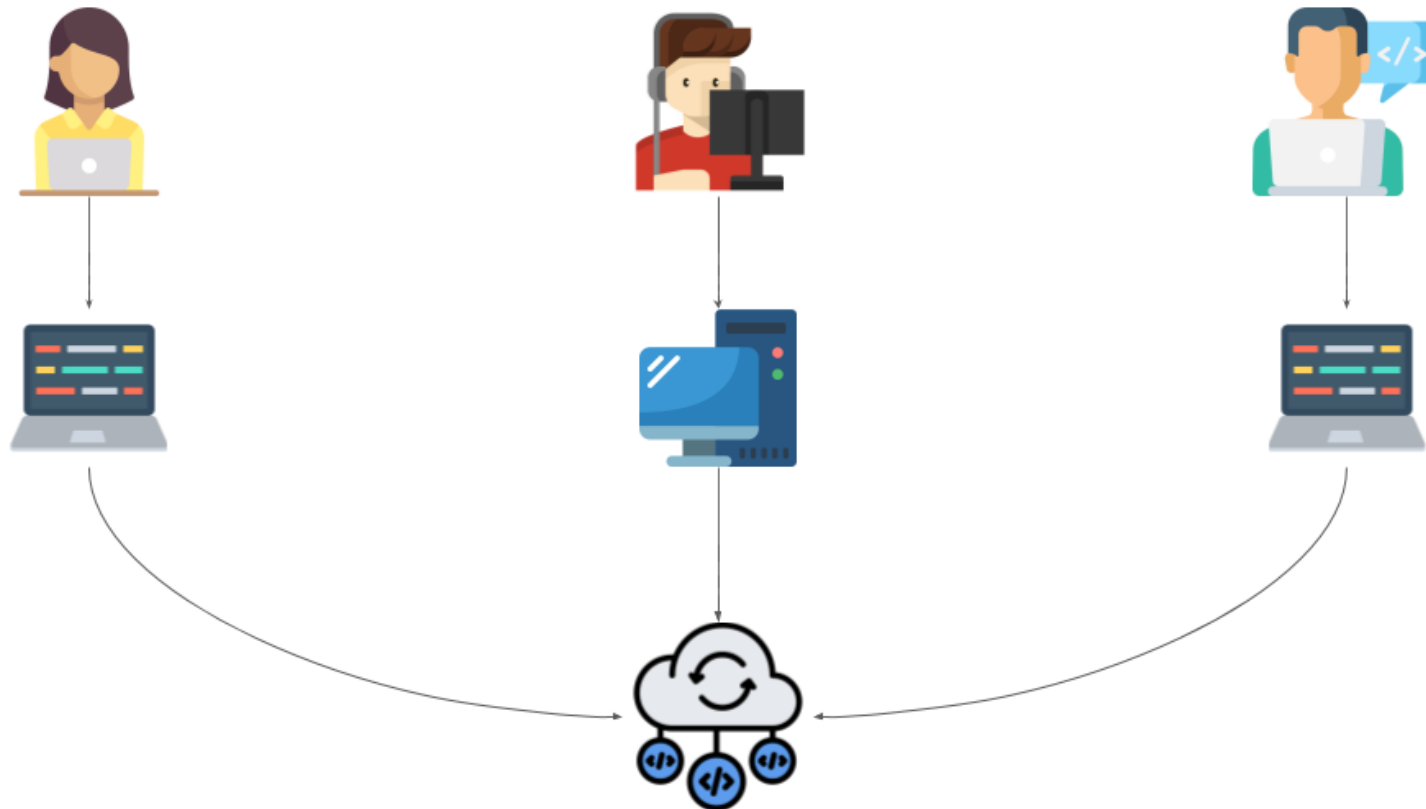
- 모든 개발자와 공유되는 중앙 저장소
 - 모든 사람이 자신의 작업 복사본
- 커밋할 때마다 변경 사항이 원격 저장소에 직접 반영
- 분산 시스템과 달리 개발자는 원격으로 직접 커밋
 - 파일에 고의로 또는 무의식적으로 영향을 줄 수 있음을 의미



3. 분산 버전 관리 시스템

깃과 깃허브 Python language

- 모든 개발자를 위한 각각의 로컬 복사본이 존재
 - 원격 저장소에 영향을 주지 않고 원하는 대로 변경하고 커밋 가능
 - 지역 저장소에서 커밋한 다음 변경 사항을 원격 저장소에 올림(푸시)
- 현재 대부분 사용되는 유형



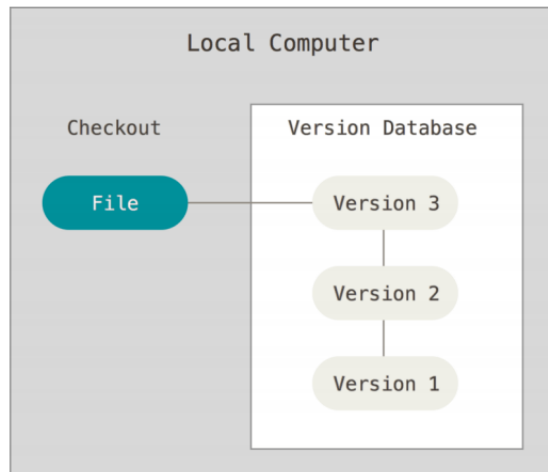


그림 1. 로컬 버전 관리.

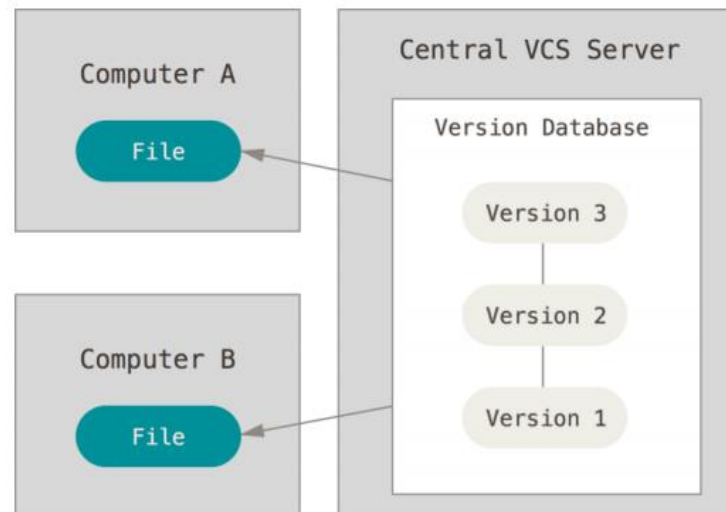


그림 2. 중앙집중식 버전 관리(CVCS).

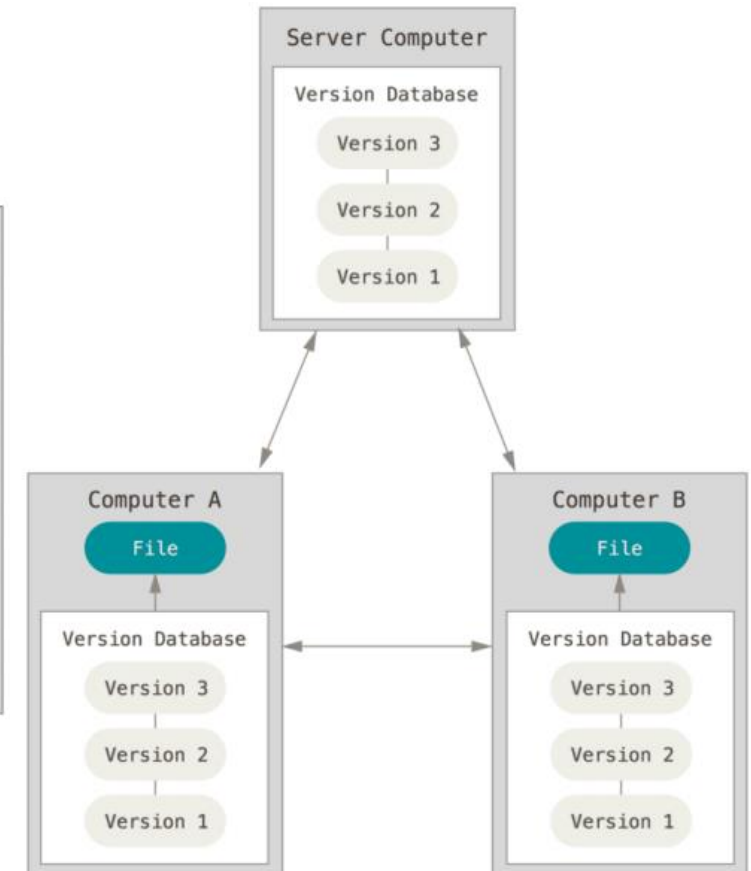


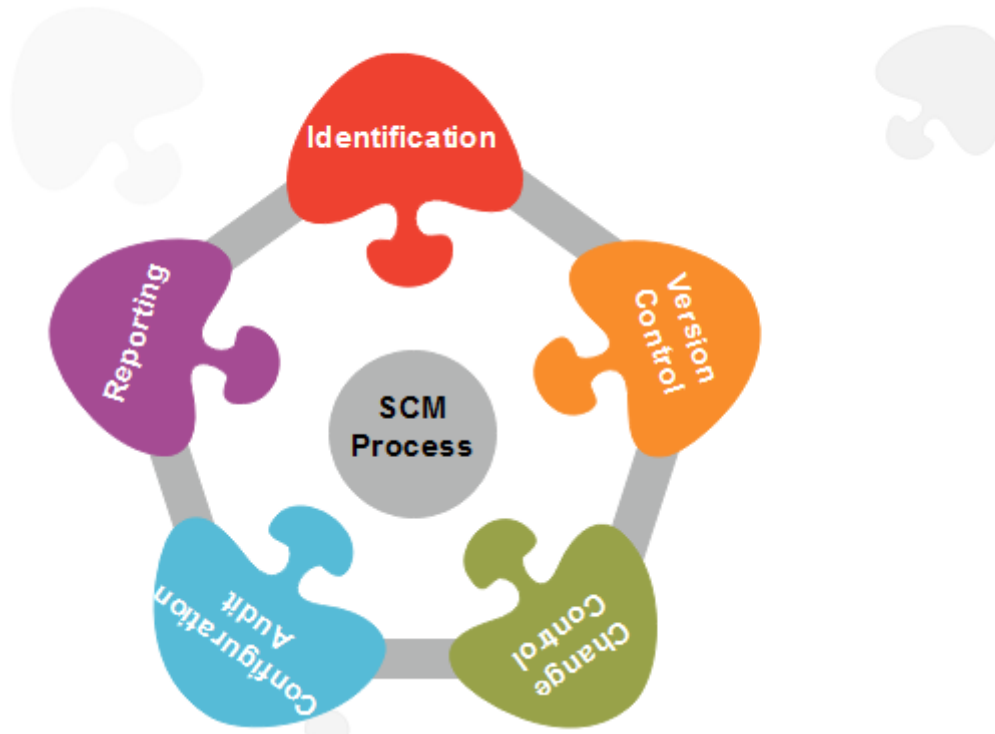
그림 3. 분산 버전 관리 시스템(DVCS).

- 소프트웨어 형상 관리(SCM: Software Configuration Management)

- SW개발 및 유지보수 과정에서 발생하는 소스코드, 문서, 인터페이스 등 각종 결과물에 대해 형상을 만들고, 이들 형상에 대한 변경을 체계적으로 관리, 제어하기 위한 활동
 - 프로젝트를 진행하면서 생성하는 소스코드를 CVS나 SVN, 또는 GIT와 같은 버전 관리 시스템을 이용하는 것
- 다수의 개발자가 프로젝트에서 동일한 기능을 동시에 개발
 - 작성된 소스 코드와 변경사항을 확인하고, 수정하는 협업을 도와주는 시스템

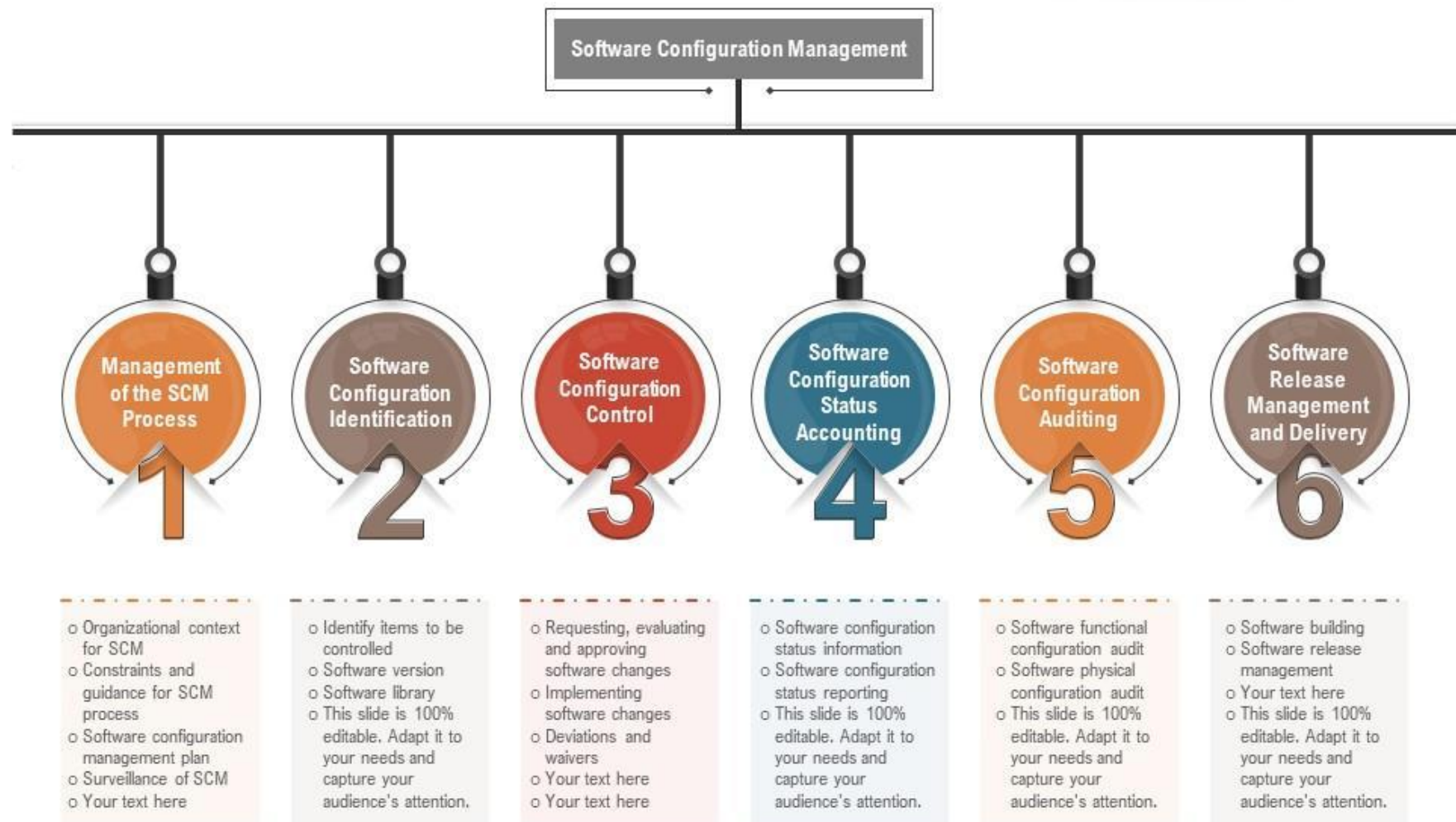
Software Configuration Management Process

형상관리 =
프로젝트 관리
+ 문서 관리
+ 버전 관리



Key Aspects of Software Configuration Management

This slide is 100% editable. Adapt it to your needs and capture your audience's attention.



- **Git**
 - 가장 많이 활용되는 분산 버전 관리 시스템

Top 5 Open Source Version Control Tools for System Admins



CVS



SVN



GIT



Mercurial



Bazaar

+1 (888) 571-6480

getfilecloud.com

AI Experts
Who Lead
The Future

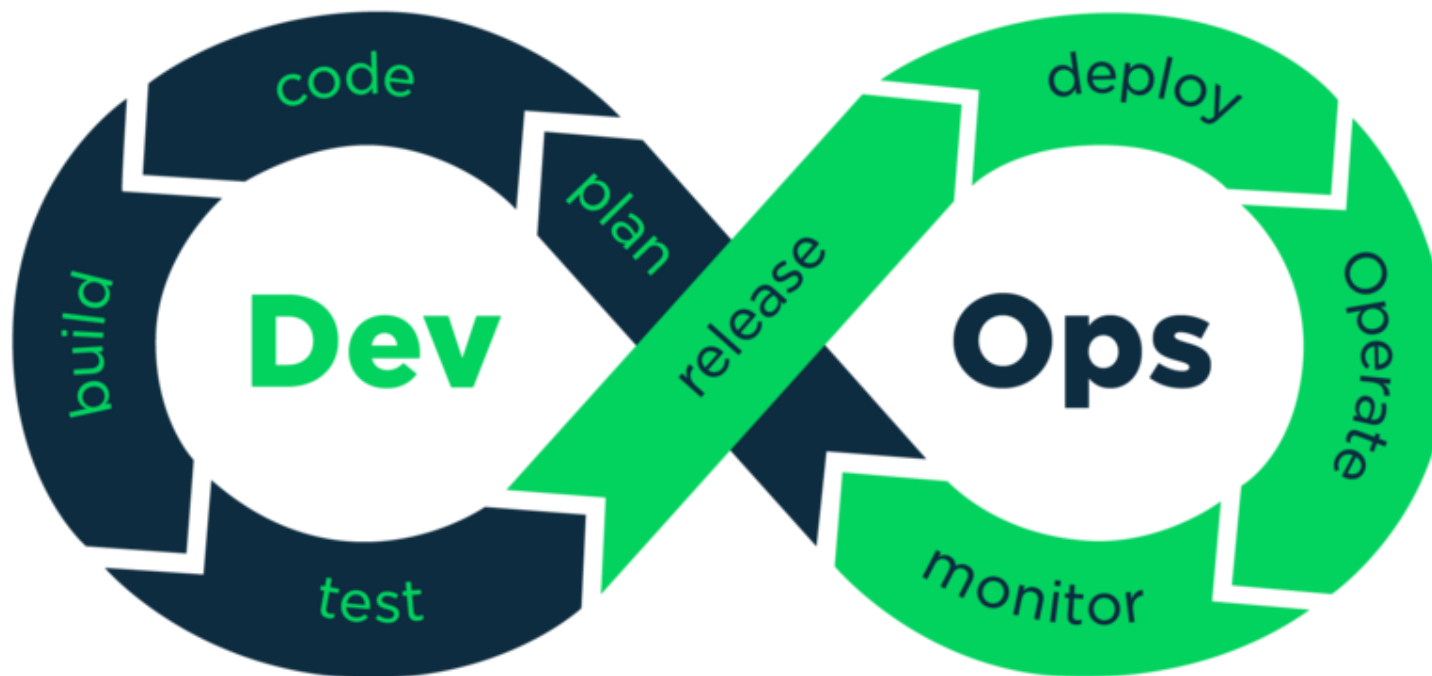
02

깃 소개

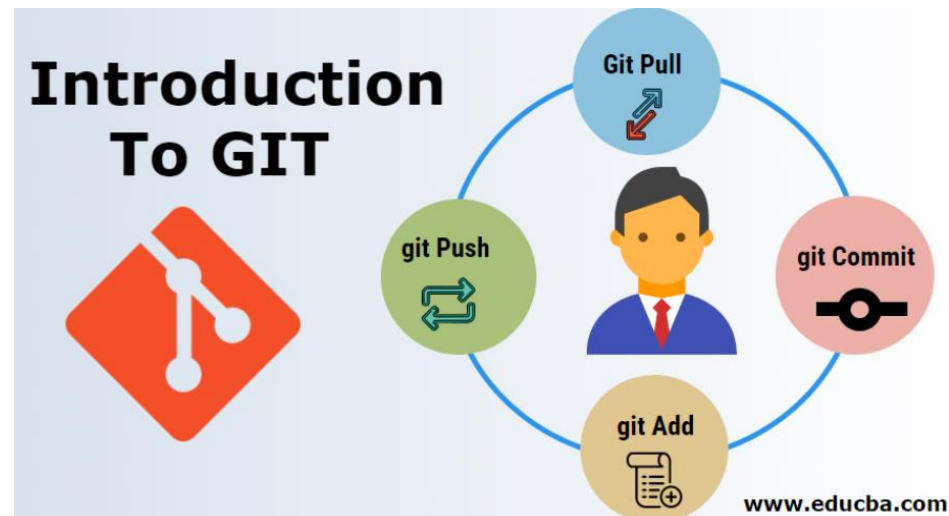


- 프로젝트 추진의 어려움
 - 다수 개발자 지원 필요
 - 소스 버전 관리
 - 협업 관리 기능
 - 소프트웨어 개발과 함께, 개발 이후의 프로젝트 운영과 관리가 함께 필요
 - 데브옵스 : DevOps == Development + Operation
 - 한번 작성한 코드가 잘 동작 되도록 상태를 유지 관리
 - 지속적인 기능 향상과 빠른 배포

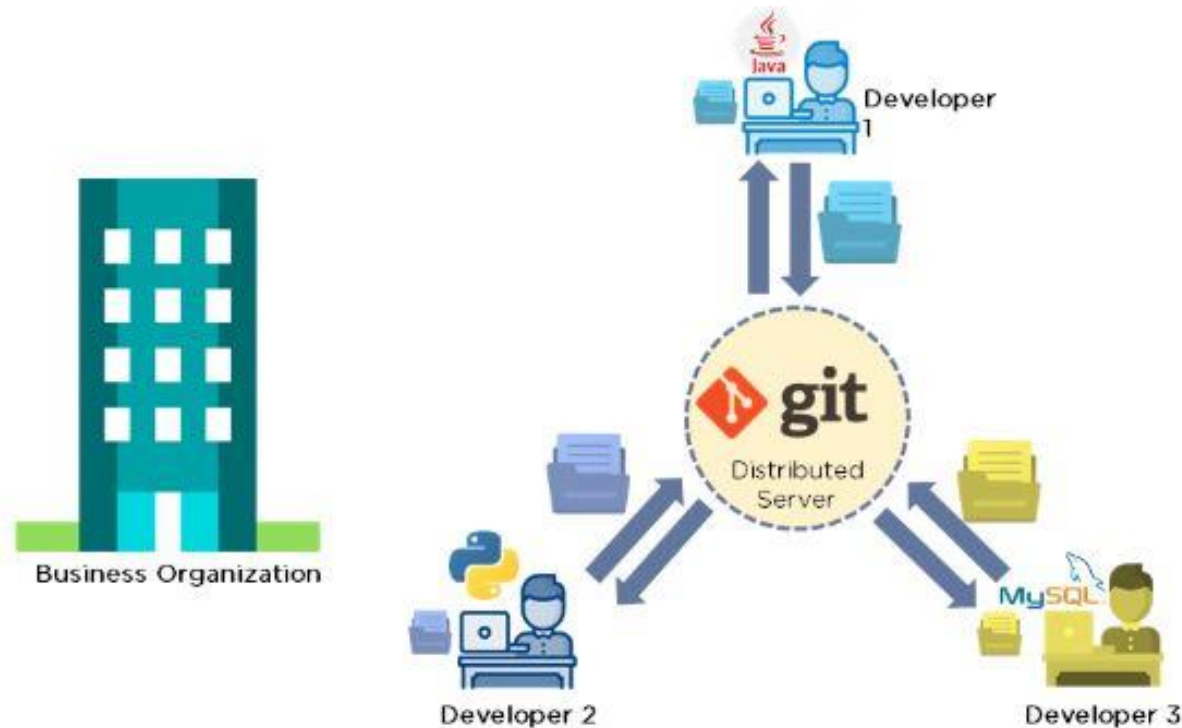
- **개발과 운영의 합성어**
 - 2009년 처음으로 등장하게 된 용어
 - 개발과 운영의 경계를 허물고 한 아이템으로서 소통, 협업 및 통합을 강조하는 개발환경이나 문화
- **데브옵스 적용이 가능한 둘을 묶어 하나의 체인 형식**
 - 모든 과정인 기획, 개발, 빌드, 테스트, 릴리즈, 배포, 운영, 모니터링을 묶어 연계



- 2005년, 리누스 토발즈가 개발
 - 주니오 하마노(Junio Hamano)가 소프트웨어의 유지보수
- 대표적인 오픈 소스 소프트웨어 프로젝트
 - <https://github.com/git>
 - <https://git.kernel.org/pub/scm/git/git.git>



- 모든 개발자는 지역 시스템에 코드의 전체 사본을 소유
- 소스 코드에 대한 모든 변경 사항은 다른 사용자가 추적 가능
- 개발자 간에 정기적으로 커뮤니케이션 가능



- 정의

- 컴퓨터 파일의 변경을 추적하는 데 사용되는 버전 관리 시스템

- 기능

- 소스 코드의 변경 사항을 추적하는 데 사용
- 소스 코드 관리에 분산 버전 제어 도구가 사용
- 여러 개발자가 함께 작업
- 여러 개의 평행 분기를 통해 비선형 개발을 지원

- 특징

- 기록 추적
- 백업 생성
- 협업 지원
- 분산 개발
- 비선형 개발 지원
 - 브랜치 지원
- 자유 및 오픈 소스에 적합

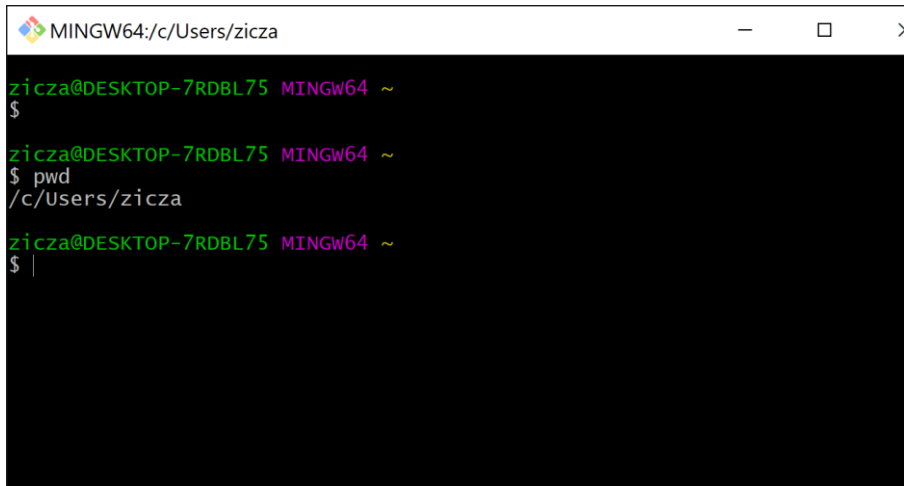


- **Git Bash: CLI**

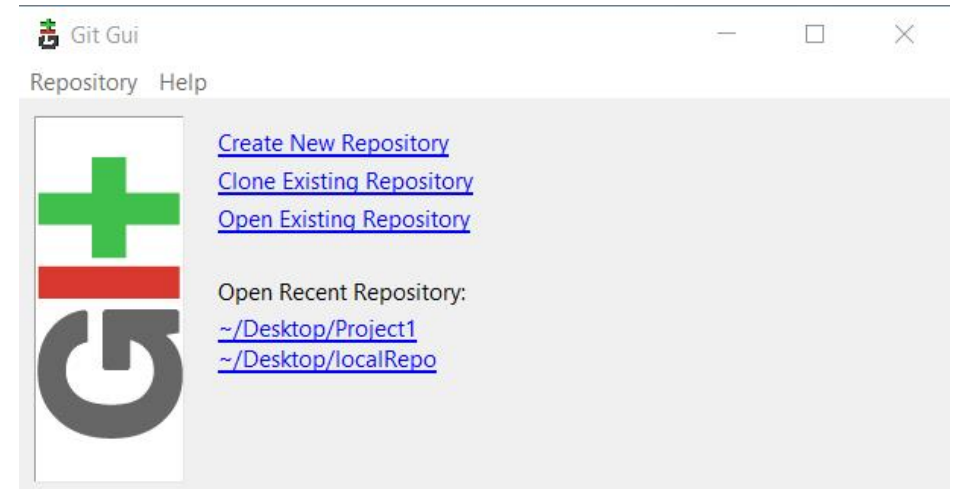
- 명령 행 인터페이스
 - 처음엔 어렵지만
 - CLI를 사용할 줄 알면 GUI도 사용할 수 있지만
 - 반대는 성립하지 않음
- Mac의 Terminal
- Windows의 CMD나 Powershell

- **Git GUI: GUI**

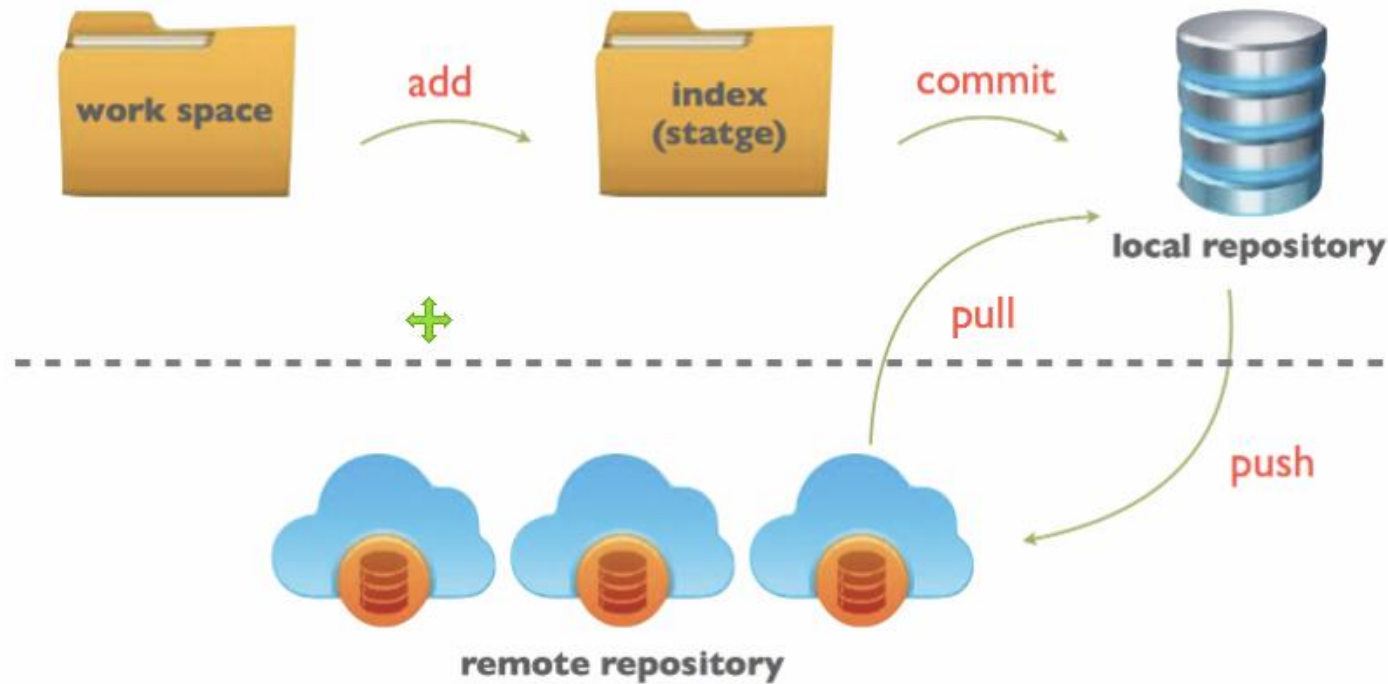
- GUI 프로그램의 대부분은 Git 기능 중 일부만 구현하기 때문에 비교적 단순



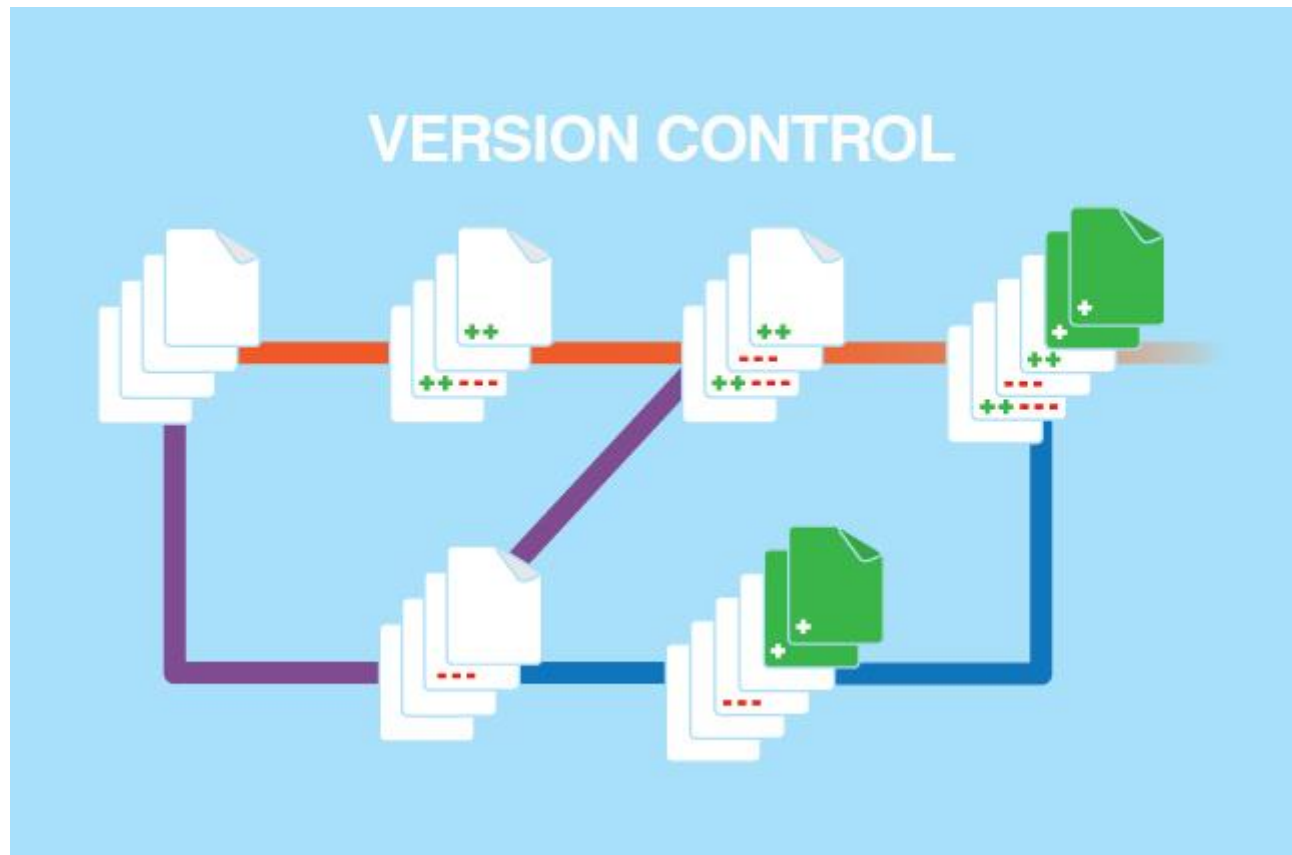
```
MINGW64:/c/Users/zicza
zicza@DESKTOP-7RDBL75 MINGW64 ~
$
zicza@DESKTOP-7RDBL75 MINGW64 ~
$ pwd
/c/Users/zicza
zicza@DESKTOP-7RDBL75 MINGW64 ~
$ |
```



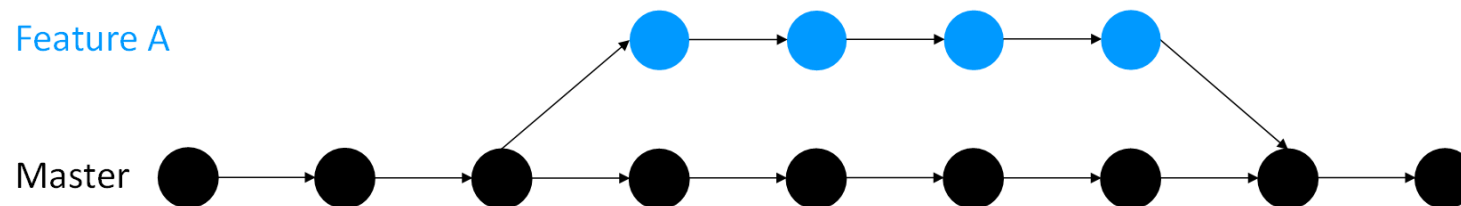
- 저장소에서의 3 상태
 - 작업 공간, 스테이징 영역, 저장소
 - 상태 간의 이동 명령
 - add
 - commit



- 분기, 가지(branch) 개념
 - 새로운 수정을 할 수 있는 또 다른 버전의 작업 흐름(workflow)



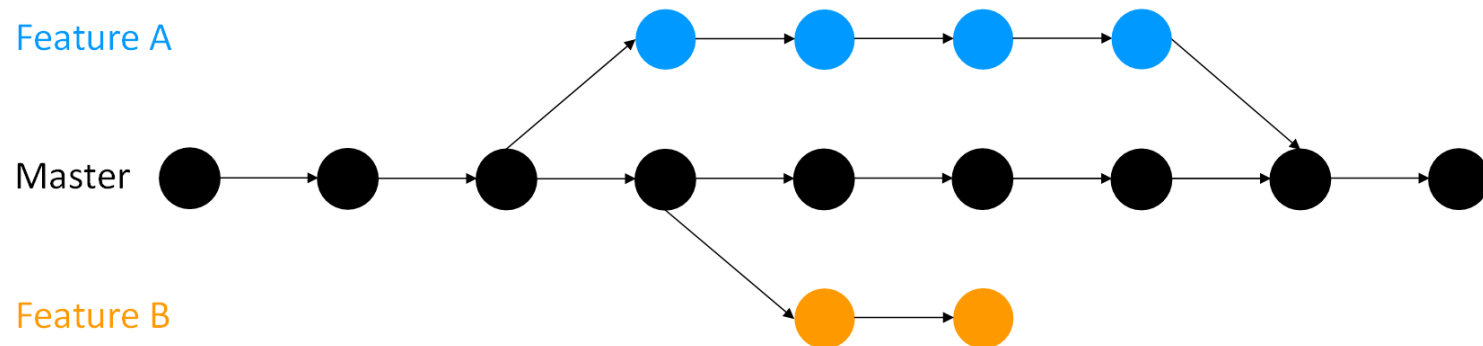
- **브랜치의 이용**
 - 기본의 저장소의 변경 사항을 반영하지 않고 새로운 버전으로
 - 무언가를 시도하고 싶다면
- **브랜치 생성**
 - 새로운 브랜치에서 하는 어떤 업무를 하든
 - 이전의 메인 브랜치에 반영되지 않으며, 안전하고 오류 없는 상태를 유지
 - 새로운 브랜치에서 아이디어를 테스트하고 문제를 해결 가능
- **브랜치 병합**
 - 새로운 브랜치의 개발이 완료되면
 - 메인 브랜치에 병합
 - 지역 브랜치의 독립된 개발 라인을 메인 브랜치로 통합



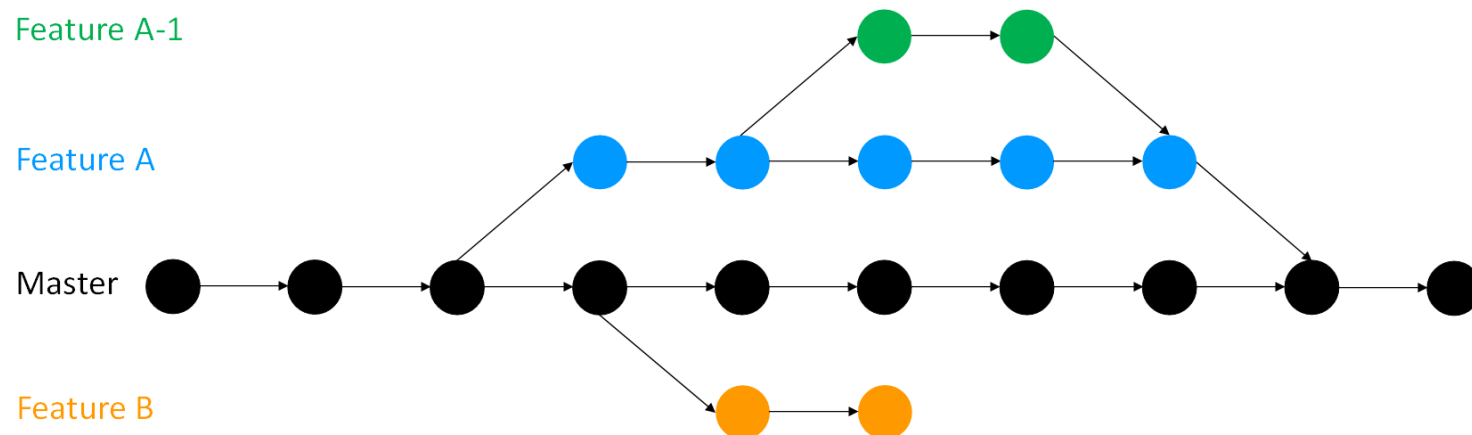
- 여러 분기도 가능

- 분기 중 하나가 작동하지 않으면

- 프로젝트의 주 분기에 영향을 주지 않고 해당 분기를 포기하거나 삭제 가능

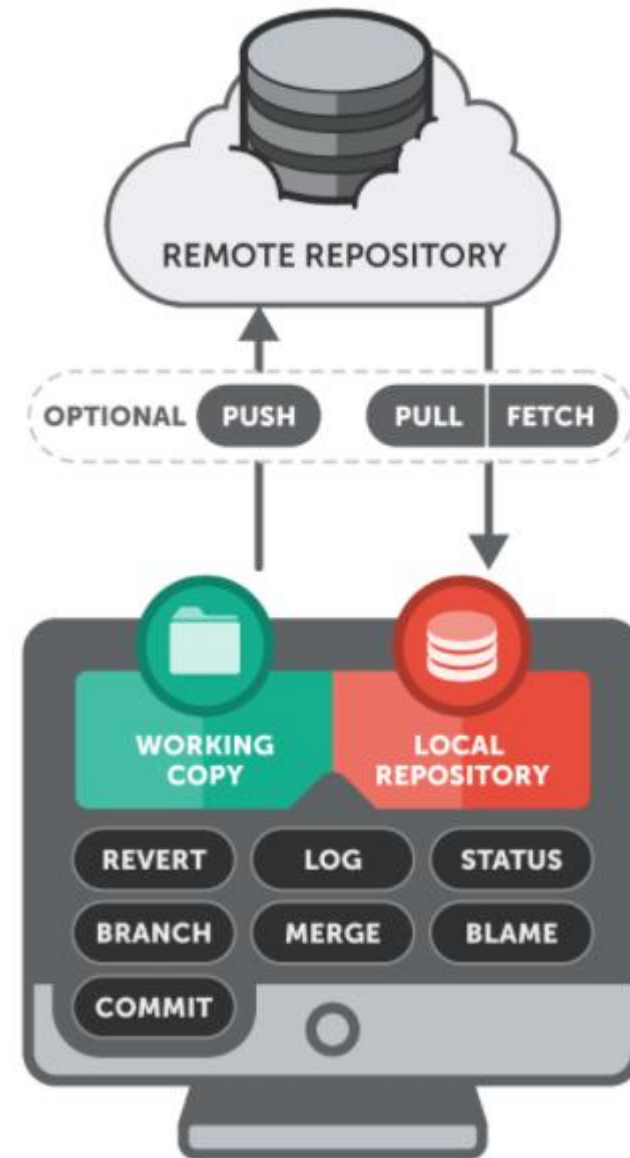


- 분기에 분기를 생성 가능

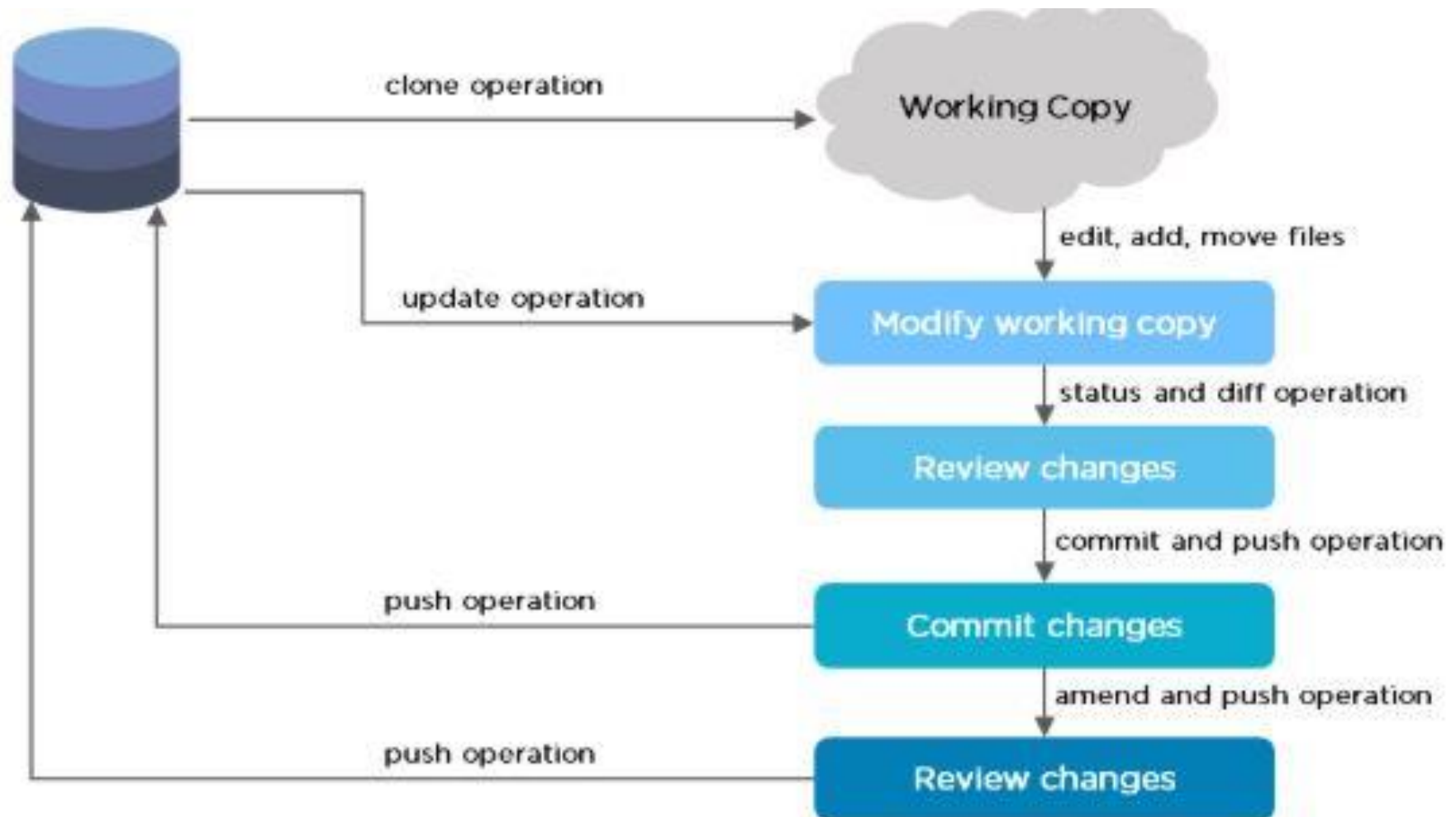


- 기능(features) 기반 워크플로우
 - 작업 중인 각 새 기능에 대해 새 분기를 만들어 해당 분기를 앞뒤로 원활하게 전환한 다음
 - 해당 기능이 주 라인에 병합되면 각 분기를 삭제
- 단순한 일회용 실험
 - 테스트할 분기를 만들고, 효과가 없을 것이라는 것을 깨닫고
 - 아무도 보지 않았으므로 그냥 지우면 OK!

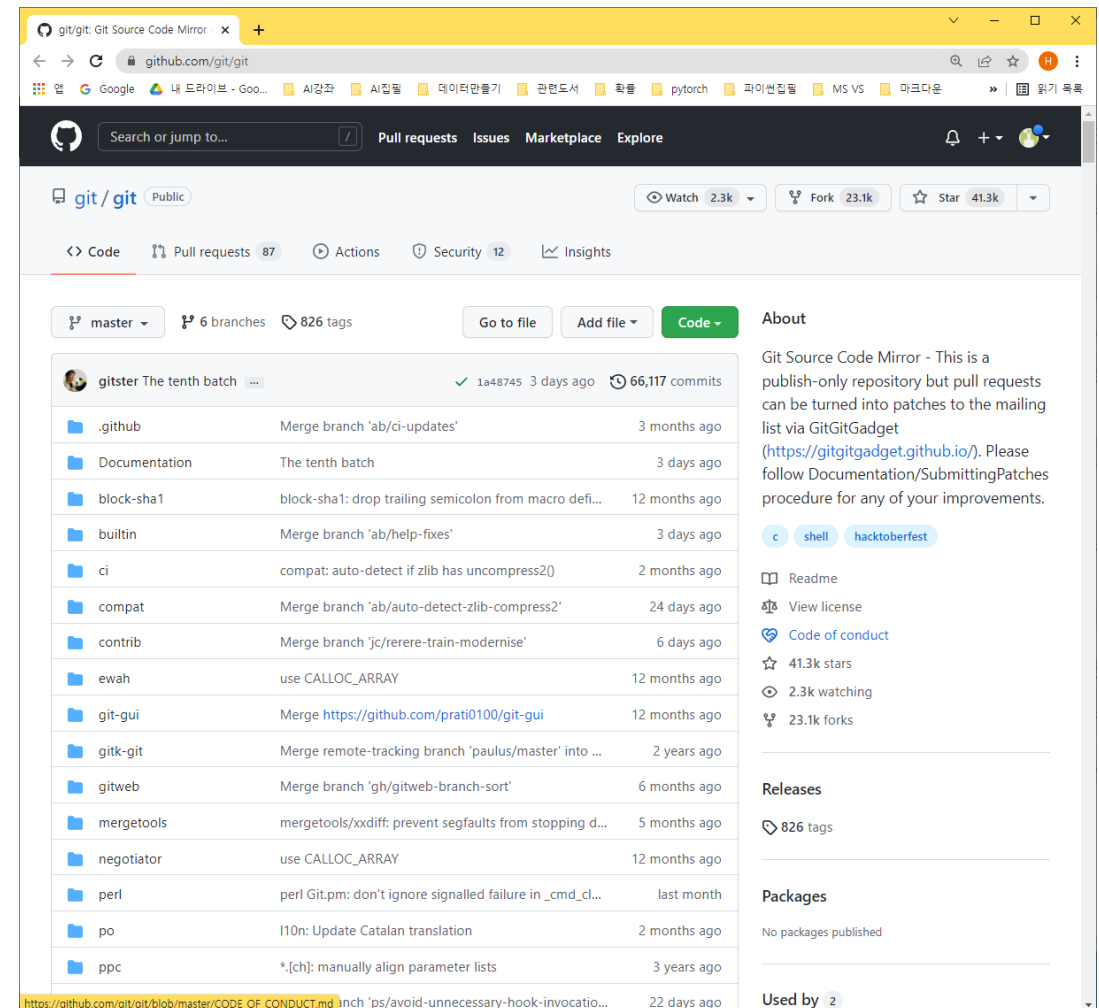
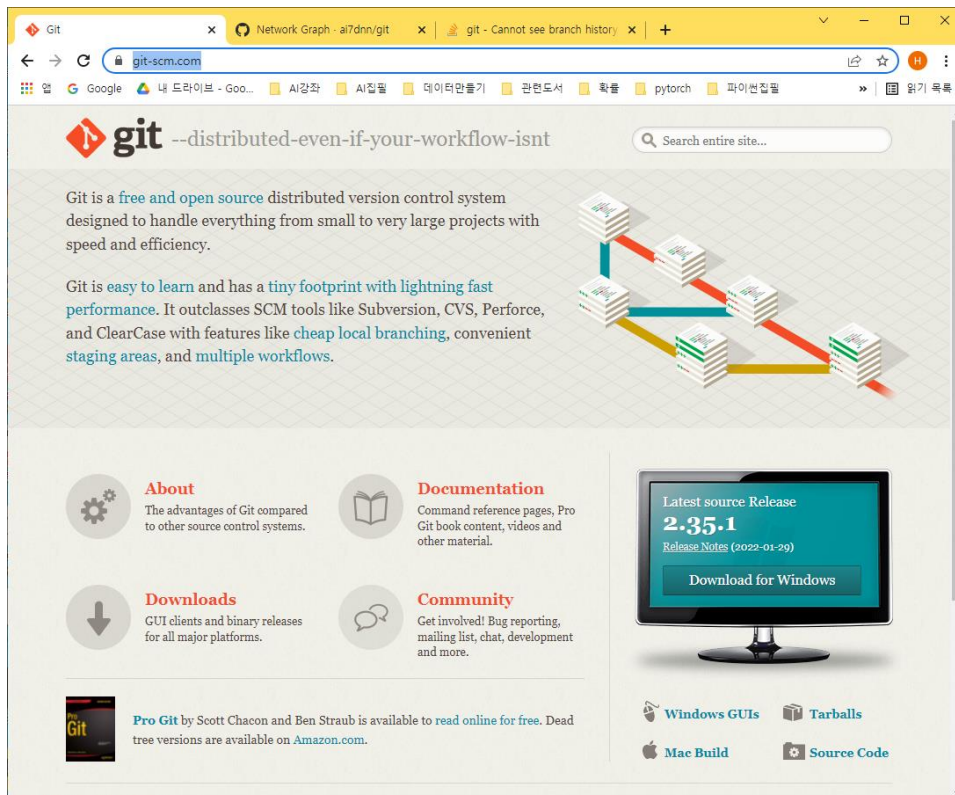




<https://www.git-tower.com/learn/git/ebook/en/command-line/appendix/from-subversion-to-git>



- 홈페이지
 - <https://git-scm.com/>
- 오픈소스소프트웨어 OSS 주소
 - <https://github.com/git/git>



- <https://git-scm.com/about>

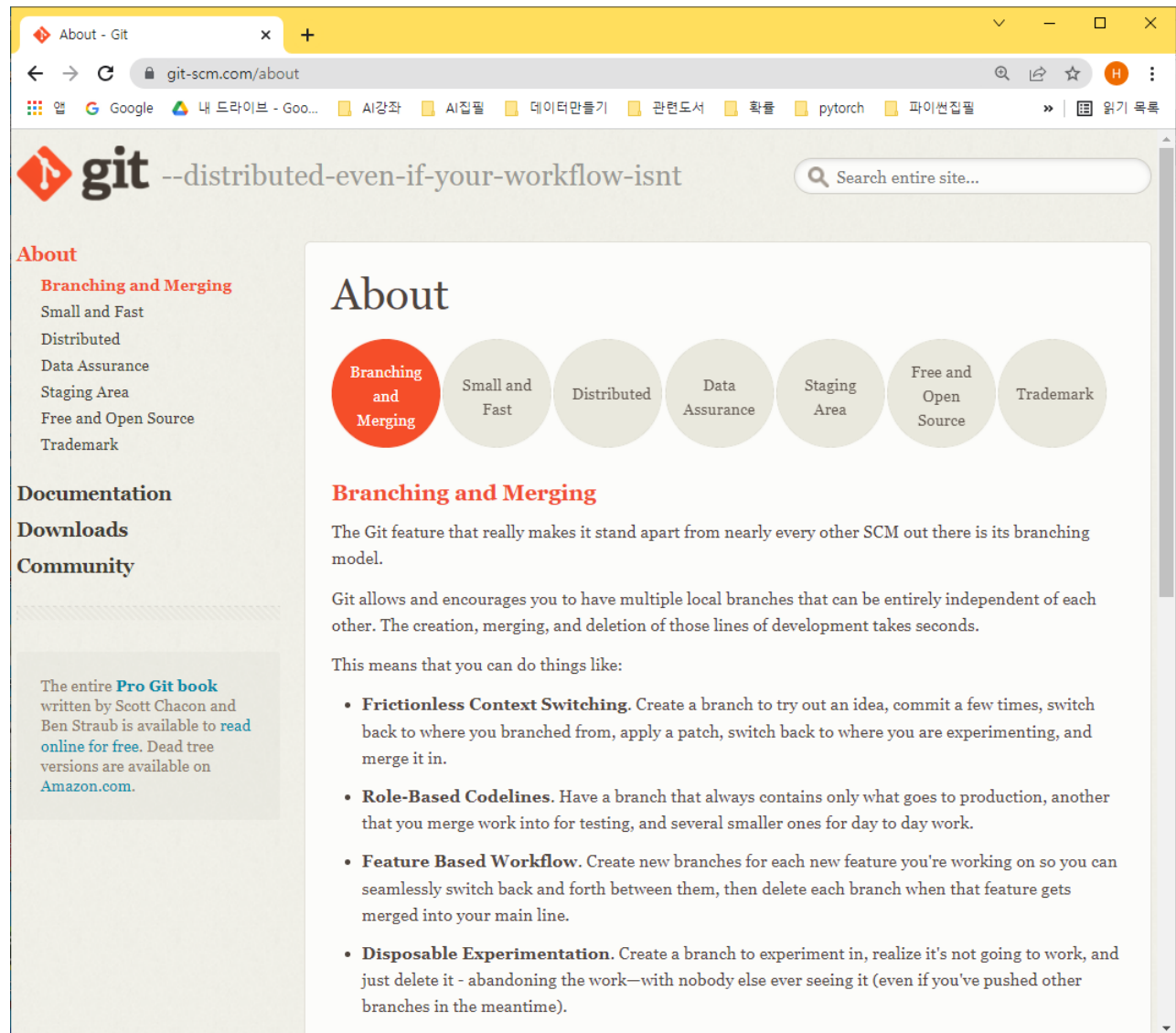
- **About**

- Branching and Merging
- Small and Fast
- Distributed
- Data Assurance
- Staging Area
- Free and Open Source
- Trademark

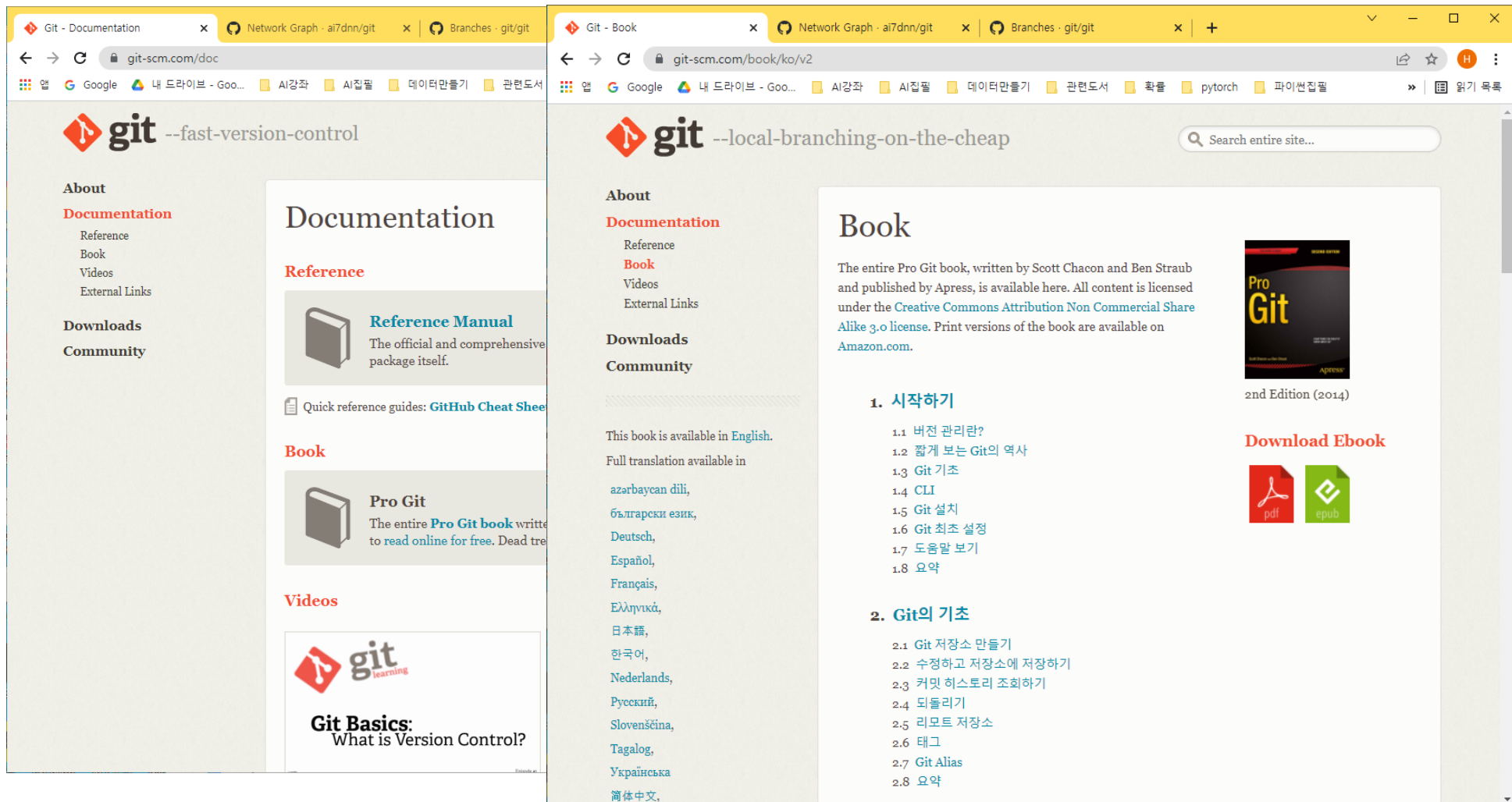
- **Documentation**

- **Downloads**

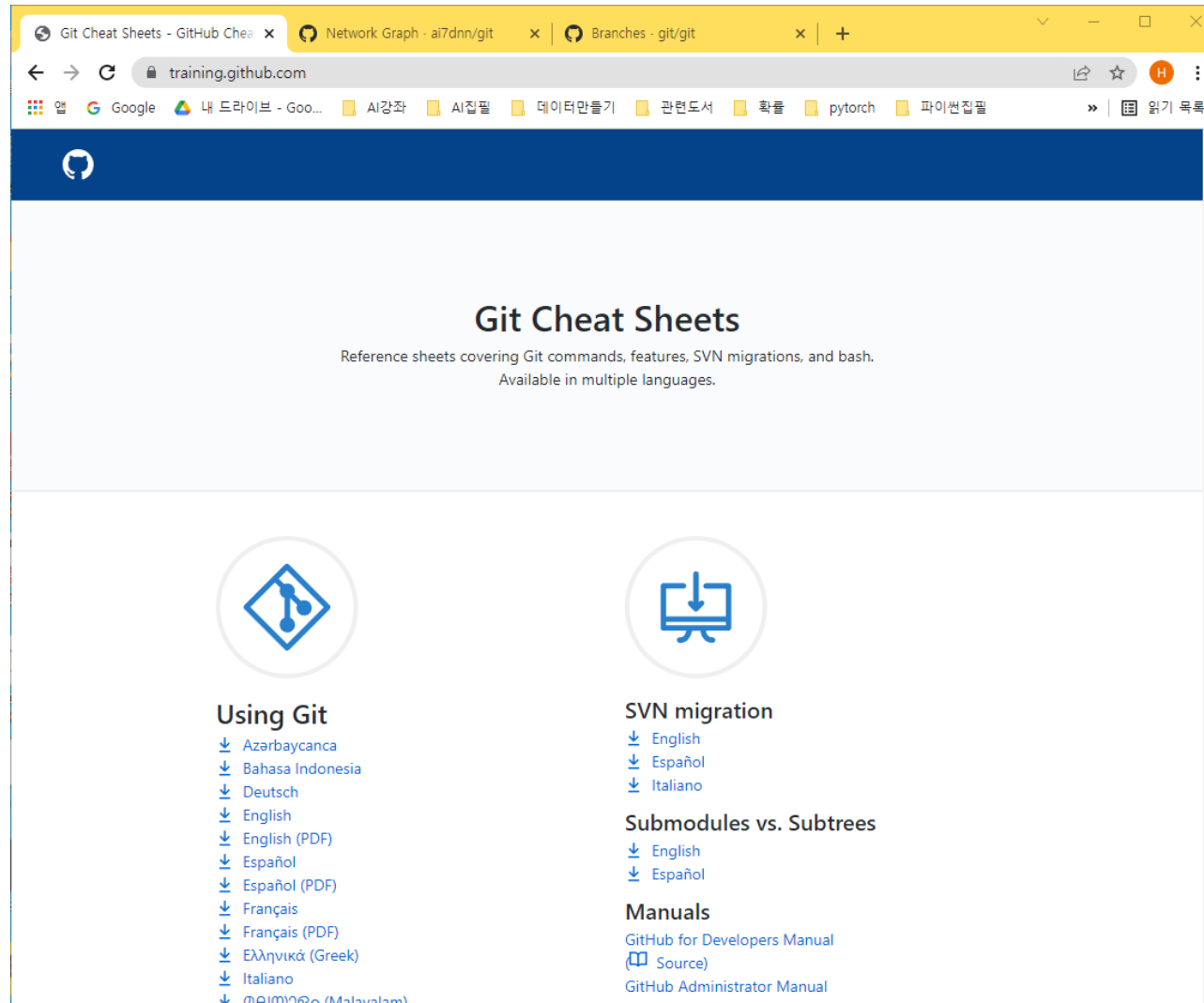
- **Community**



- <https://git-scm.com/book/ko/v2>



- <https://training.github.com/>



Windows GUIs



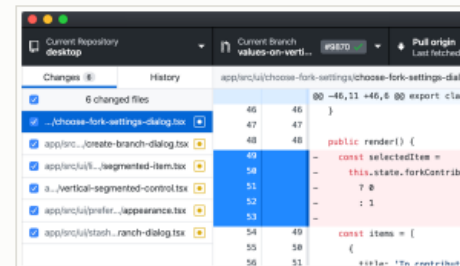
GUI Clients

Git comes with built-in GUI tools for committing ([git-gui](#)) and browsing ([gitk](#)), but there are several third-party tools for users looking for platform-specific experience.

If you want to add another GUI tool to this list, just [follow the instructions](#).

All Windows Mac Linux Android iOS

32 Windows GUIs are shown below ↓

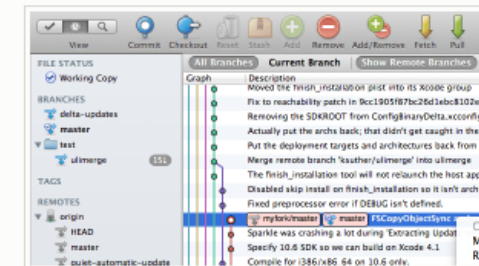


GitHub Desktop

Platforms: Mac, Windows

Price: Free

License: MIT

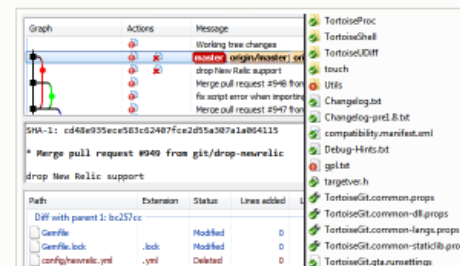


SourceTree

Platforms: Mac, Windows

Price: Free

License: Proprietary

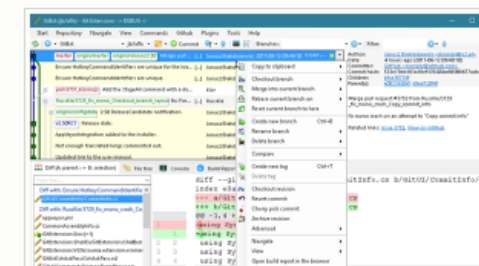


TortoiseGit

Platforms: Windows

Price: Free

License: GNU GPL



Git Extensions

Platforms: Linux, Mac, Windows

Price: Free

License: GNU GPL