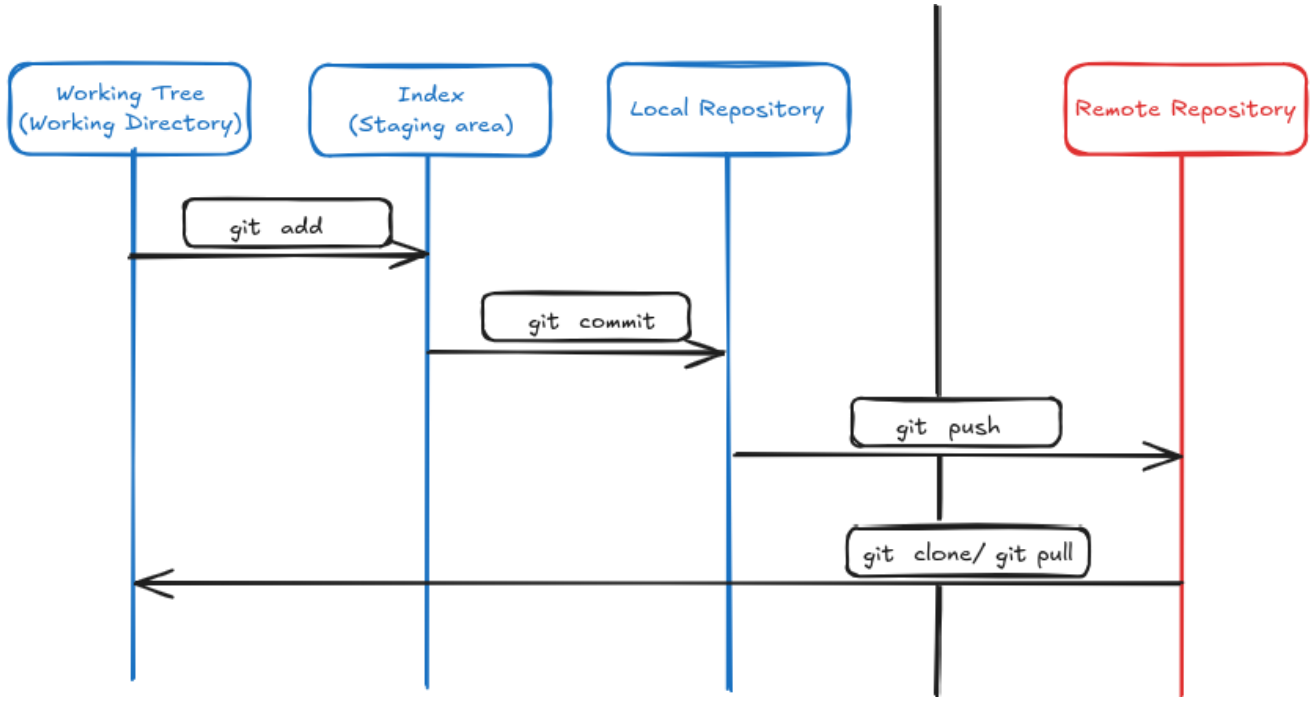
GIT

대표적인 형상 관리 도구

형상 관리 도구란

소프트웨어 버전 관리 도구(Version Control System-VCS)라고도 한다.
형상 관리(Configuration Management)는 소프트웨어 개발 시 **변경 사항을 체계적으로 관리하고 추적**하는 프로세스이다. 다시 말하면 프로그램을 만들 때 코드와 파일들의 변경된 이력(history)들을 관리하는 것을 말한다.
이를 통해 코드를 수정하거나 새로운 코드 파일을 추가하면 그 기록을 남겨서 누가 무엇을 바꿨는지 알 수 있도록 하고 문제가 생겼을 때 이전 상태로 돌아갈 수 있도록 한다.
형상 관리 도구에는 Git, SVN, Mercurial등이 있는데 Git이 가장 많이 사용된다.

GIT에서 형상 관리 상태



Working directory(Working Tree)

- Working directory, Working tree, 작업 디렉토리, 작업 폴더 등으로 불린다.
- 사용자의 작업 결과물들을 저장하는 디렉토리이다.
- 작업폴더에서 `.git` 디렉토리(Local 저장소)를 뺀 나머지 부분.

Index(Staging Area)

- Working directory의 코드 중 **변경된 파일들 중 local respoitory에 저장할 파일들을 임시로 저장하는 영역**이다.
    - Working directory와 git local repository사이에서 변경, 선택된 파일을 선택적으로 관리하고 준비할 수 있게 한다.
    - working directory의 모든 파일이 형상관리 대상이 아닐 수 있다.

Local Repository(로컬 저장소)

- `.git` 디렉토리에 관리 이력(history)이 저장된다.
  - `.git` 디렉토리를 저장소라고 하고 이것이 내 컴퓨터에 있으면 local 저장소, 서버에 있으면 원격 저장소라고 한다.
- `git init` 명령어로 생성된다.
- 커밋, 커밋 구성 객체, stage가 여기 저장된다.

Remote Repository (원격 저장소)

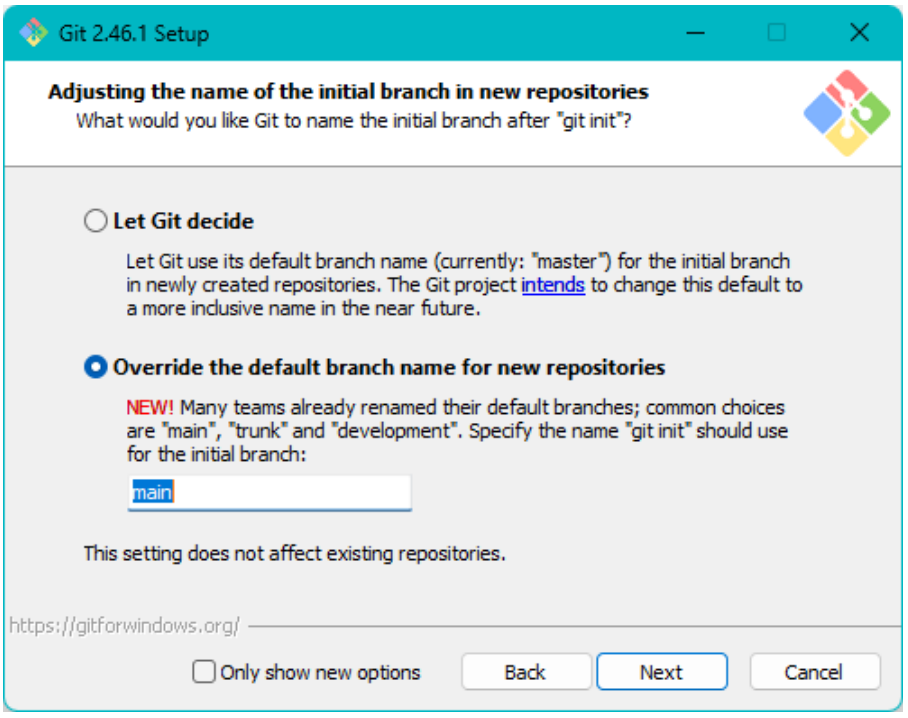
- 서버에 있는 Git 저장소
- ex) Github의 저장소

흐름

- working directory에 새로 작성한 파일이나 수정한 파일 **Modified** 상태라고 한다.
  - **modified**: 새로 생성 또는 commit 이후 수정된 파일들
- modified 된 파일 중에서 관리 대상 파일을 선택한다. 선택된 파일을 **Staged** 되었다고 한다.
  - **staged**: commit 대상으로 선택된 파일.
  - `git add 파일명`, `git add .`
- Staged 된 파일의 이력을 Local repository에 저장한다. 이력이 저장된 상태를 **commited** 라고 한다.
  - **commited**: Local Repository에 변경 이력이 저장된 파일들.
  - `git commit -m "메세지`
- Local Repository에 저장된 이력을 Remote Repository에 저장한다.
  - Local과 Remote의 상태를 일치 시킨다.
  - 주요 목적
    - 이력을 backup한다.
    - 여러 사람들과 협업
  - `git push`

설치

- https://git-scm.com/download
    - 운영체제에 맞춰 다운로드 및 설치
- 설치 시 **Adjusting the name of the initial branch in new repositories** 항목에서 **Override the default branch name for new repositoies**를 선택한다.



- 나머지 항목은 **Next** 선택한다.

기본 명령어

- **초기 설정**
  - `git config --global user.name "이름"`
    - 사용자 이름 등록
  - `git config --global user.email "이메일 주소"`
    - 사용자 이메일 주소 등록
  - `git config --global core.quotepath false`
    - 한글 처리
  - `git config --global core.autocrlf true`
    - Windows 에서 개행 문자(엔터) 변경 관련 처리 (Windows에서만 한다.)
  - `git config --global init.defaultBranch main`
    - 기본 branch 이름을 master에서 main으로 변경
    - 설치 할 때 설정한 경우 안 해도 된다.
- `git init`
  - 현재 폴더를 working directory + local repository로 만든다.
  - `.git` (git local repository) 디렉토리가 생성된다.
- `git clone 원격저장소경로`
  - 원격 저장소(Remote Repository)를 복사해서 Local 저장소를 만든다.
  - 협업 할 때 다른 사용자가 만든 저장소를 내 컴퓨터에 복사할 때 사용한다.
- `git status`
  - 현재 working directory와 Staging area의 상태를 확인
  - add 대상, commit 대상을 보여 준다.
- `git add 파일`
  - 지정한 파일을 staged 상태로 만들어 준다.
  - **staged 상태란** git에서 형상 관리를 대상 파일을 말한다.
  - 파일 지정
    - 개별 파일 경로지정
      - `git add 파일명`
      - ex) `git add abc.txt`
    - wild card 문자 이용
      - `git add .`: 현재 디렉토리와 하위 디렉토리의 변경사항을 모두 Index(Staging Area)에 추가
      - `git add *.txt`: 현재 디렉토리의 확장자가 txt인 모든 파일들의 변경사항을 Index에 추가.
- Staging(add) 취소 - `git reset HEAD [파일명]`
  - git add **취소** 하여 unstaged 상태로 만든다.
  - 파일명을 생략하면 add된 모든 파일을 unstaged 상태로 변경한다.

- `git commit`
  - 커밋(최종 확정) 처리
  - staged 상태의 파일들을 커밋한다.
  - Commit은 **하나의 작업 단위가 종료**되면 처리한다.
  - Commit 메세지를 작성하기 위해 editor 가 실행되며 메세지 작성 후 저장하면 commit이 처리 됨.
  - `git commit -m "커밋 메세지"` 로 editor 없이 메세지를 작성할 수 있다.
- Commit 취소 - `git reset [옵션] [되돌릴 commit hash]`
  - 되돌릴 commit hash는 앞에 일부만 넣어도 된다.
    - `git log`(커밋 이력 확인)로 commit hash 확인
  - 되돌릴 commit hash 대신
    - `HEAD^` : 최신 커밋 취소
    - `HEAD~정수` : 해당 정수만큼 최근 커밋부터 해당 숫자까지 커밋 취소.
  - **옵션**
    - `--soft`: commit 만 취소
    - `--mixed`(default): staging (add) 까지 취소
    - `--hard`: 수정한 내용까지 취소
      - Commit한 내용이 Error(문제)가 발생하여 커밋 전으로 돌아 가야 하는 경우 사용
- `git pull`
  - 원격저장소의 상태로 local 저장소로 다운로드
  - 작업을 시작 하기 전에 **가장 먼저 실행한다.**
- `git push 원격저장소 branch`
  - 원격저장소에 local 저장소의 커밋한 내용 올리기
  - ex) `git push origin main`
  - `git push --set-upstream origin main` 로 설정하면 `git push` 만 입력해도 자동으로 `origin main`으로 push한다.

gitignore파일

- Git을 이용한 관리 대상에서 제외할 파일들과 디렉토리들을 설정하는 파일.
- 파일명: `.gitignore`
- **Working Directory Root**에 위치 시킨다.
- 한 줄에 한개 파일 또는 한 개 디렉토리를 지정한다.
  - 명시적으로 파일명 또는 디렉토리 이름을 작성한다.
  - 디렉토리를 지정할 경우 뒤에 `/` 를 붙인다.
    - ex) `env/`: env 디렉토리를 관리대상에서 제외
    - 디렉토리를 지정하면 그 안의 파일들 모두 관리 대상에서 제외된다.
- 파일을 지정할 때 **wild card 문자**를 이용해 확장자 기반으로 설정할 수 있다.
  - `*.dat` : 확장자가 dat인 모든 파일들을 관리 대상에서 제외.
- 프로그래밍 언어 별로 .gitignore 파일을 생성해 주는 사이트
  - gitignore.io
  - Github에서 Repository를 생성할 때 생성할 수 있다.