<깃 설계 목적>

* 빠른 속도
* 단순한 디자인
* 비선형적 개발 지원(수천 개의 브랜치를 병행할 수 있음)
* 완전 분산형 시스템
* 리눅스와 같은 거대한 프로젝트도 속도 저하의 문제없이 관리할 수 있는 시스템

<Repository 만들기>

1. Git init | 비어 있는 레포지토리(프로젝트 디렉토리의 각 버전이 담기는 저장소)를 생성
2. ls -al | .git 🡪 레포지토리

<Commit 주의사항>

* 처음으로 커밋을 하기 전, 사용자 이름과 이메일 주소 설정
* 커밋 메시지 남기기(옵션 -m)
* 커밋할 파일을 git add로 지정

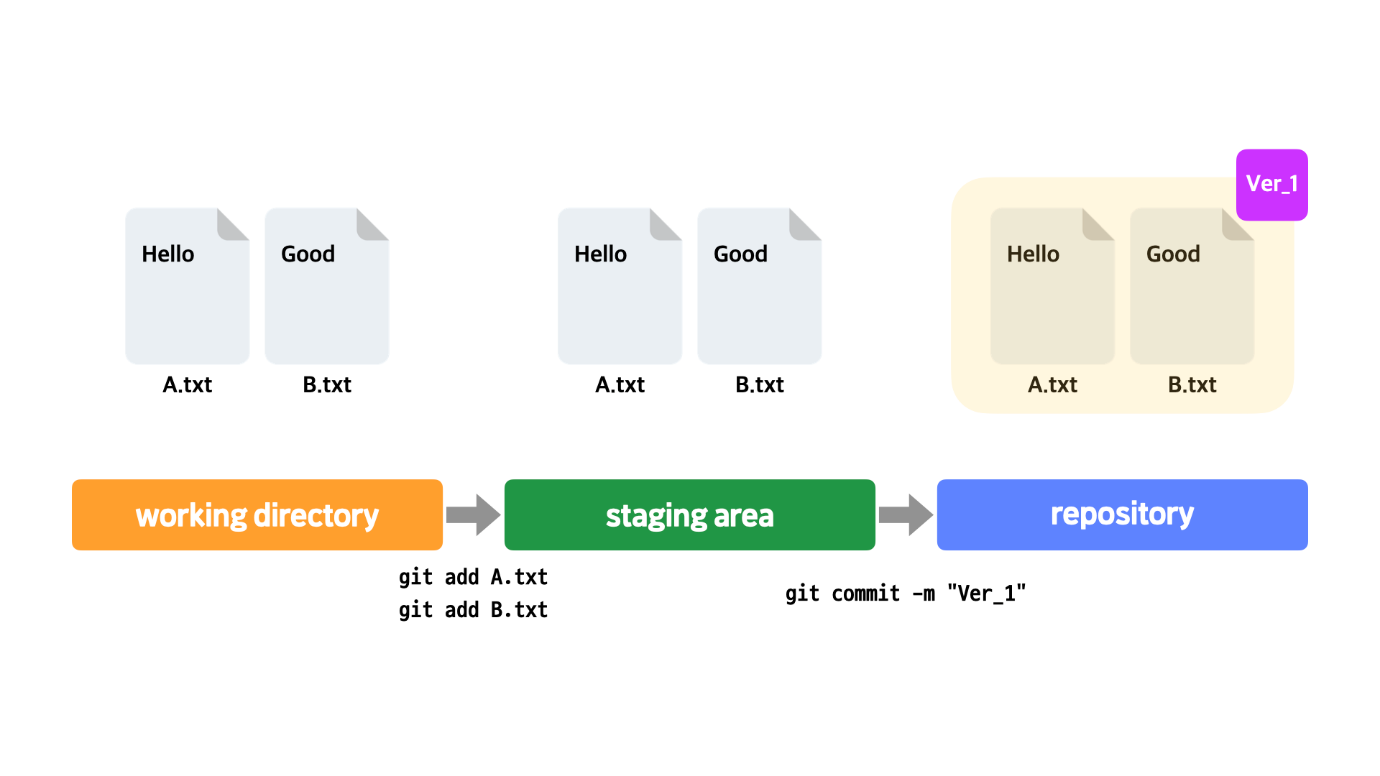
<Git 작업 영역>

1. working directory(working tree): 작업하는 프로젝트 디렉토리
   * ex. MathTool 디렉토리
2. staging area(index): git add를 한 파일들이 존재하는 영역. 커밋을 하게 되면 staging area에 있는 파일들만 커밋에 반영됨
3. repository: working directory의 변경 이력들이 저장되어 있는 영역. 즉, 커밋들이 저장되는 영역

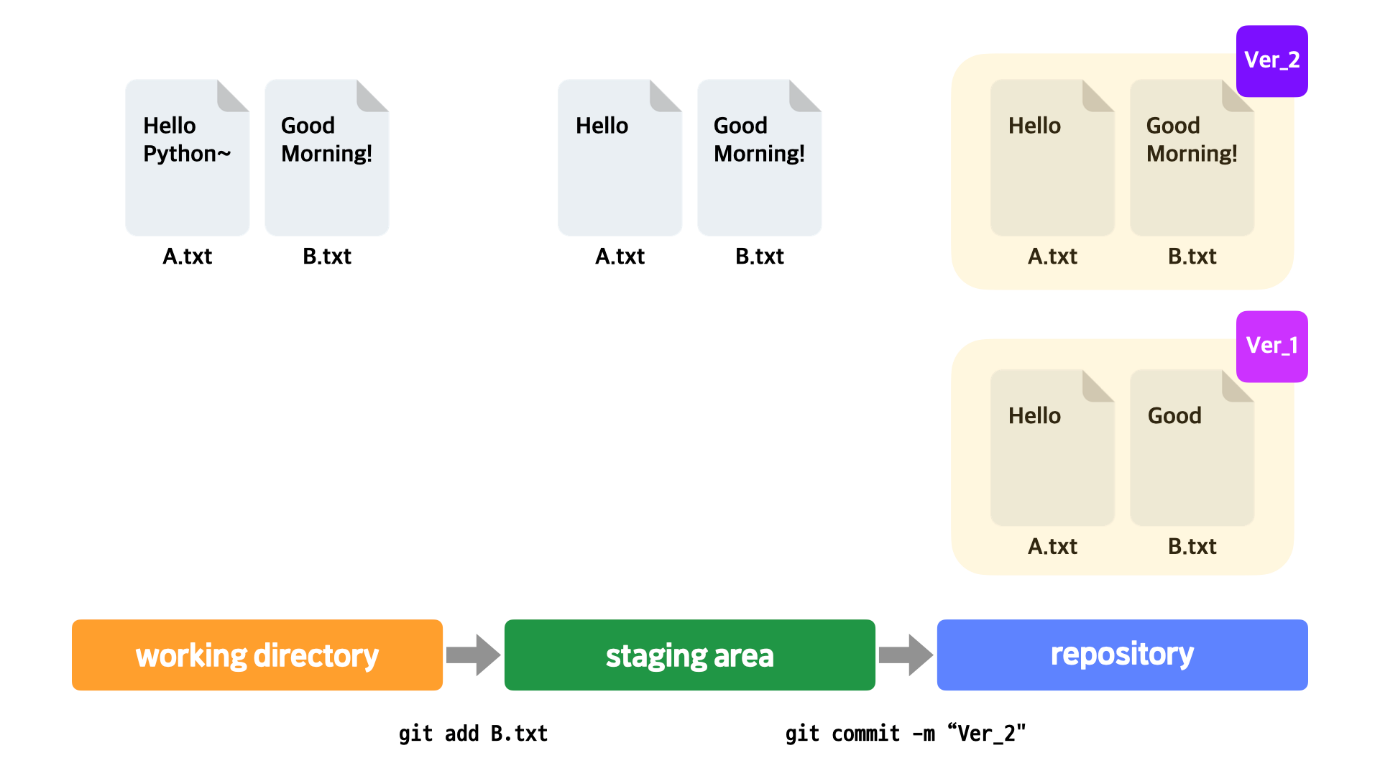
정리하면,

* working directory에서 뭔가 작업을 하고,
* 작업한 파일들을 git add 해주고,
* 커밋을 하면 staging area에 있던 파일들의 모습이 마치 영화의 한 장면, 스냅샷(snapshot)처럼 이 repository에 저장되는 것.

<Git 작업영역 예제>



1. working directory에서 A.txt 파일과 B.txt 파일을 작성하고
2. git add A.txt와 git add B.txt를 실행해서 A.txt, B.txt 둘 다 staging area에 올렸습니다.
3. 그 다음 git commit -m "Ver\_1"를 실행해서 staging area에 있는 파일들을 가져와 커밋으로 남겼습니다.



1. working directory에서 A.txt 파일 내용에 Python~이라는 단어를 추가, B.txt 파일 내용에 “Morning!”이라는 단어를 추가했습니다.
2. 그런데 이번에는 git add B.txt만 실행해서 B.txt 파일만 staging area에 올렸습니다.
3. 그 다음 git commit -m "Ver\_2"로 두 번째 커밋을 했습니다.

* 결론: working directory에서 수정했다는 사실은 동일하지만, staging area에 올렸는지 여부에 따라 그 최신 모습이 커밋에 반영되는지가 결정된다.

<Git 상태>

* Untracked 상태
* Tracked 상태
  + taged 상태
  + Unmodified 상태
  + Modified 상태

1. Untracked 상태: 파일이 Git에 의해서 그 변동사항이 전혀 추적되고 있지 않는 상태. 예를 들어, 파일을 새로 생성하고 그 파일을 한 번도 git add 해주지 않았다면 이 상태.

2. Tracked 상태: 파일이 Git에 의해 그 변동사항이 추적되고 있는 상태. 이 상태는 다시 그 특성에 따라 3가지 상태로 나뉨.

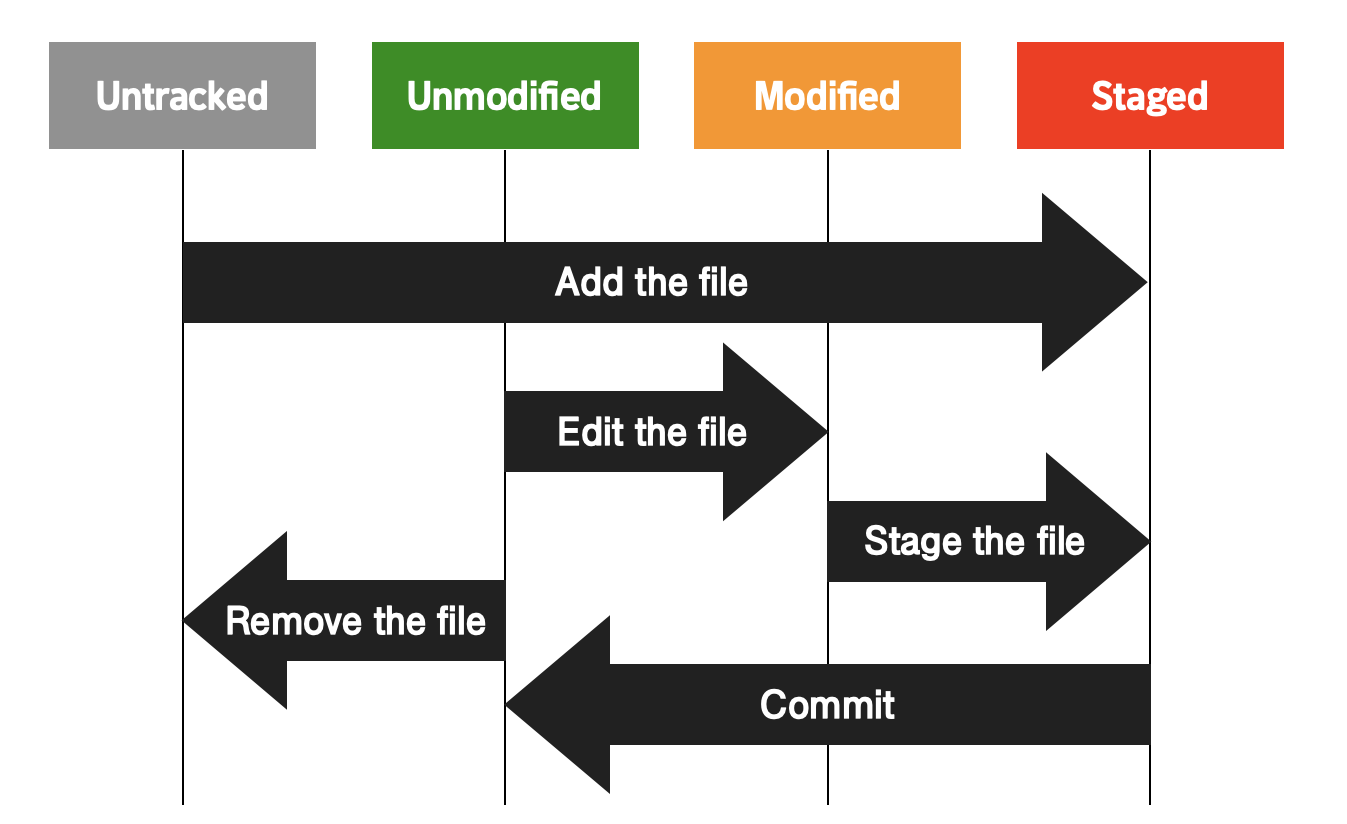
(1) Staged 상태: 파일의 내용이 수정되고나서, staging area에 올라와 있는 상태. 새로 생성한 파일에 내용을 쓰고 git add를 해주거나 한 번이라도 커밋에 포함됐었던 파일이라도 내용을 수정하고 git add를 해주면 이 상태.

(2) Unmodified 상태: 현재 파일의 내용이 최신 커밋의 모습과 비교했을 때 전혀 바뀐 게 없는 상태. 커밋을 하고 난 직후에는 working directory 안의 모든 파일들이 이 상태.

(3) Modified 상태: 최신 커밋의 모습과 비교했을 때 조금이라도 바뀐 내용이 있는 상태.

정리하면,

* Add the file: Untracked 상태의 파일을 처음으로 git add 해주면 Staged 상태가 됩니다.
* Edit the file: 최신 커밋과 비교했을 때 차이가 없는 Unmodified 상태의 파일의 내용을 수정하면 Modified 상태가 됩니다.
* Stage the file: Modified 상태의 파일을 git add 해주면 Staged 상태가 됩니다.
* Remove the file: 파일을 삭제하면 당연히 Git에서 더이상 인식하지 않겠죠?
* Commit: 커밋을 하면 staging area에 있던 파일들이 커밋에 반영되고, 이제 모든 파일들은 최신 커밋과 차이가 없게 되니까 Unmodified 상태가 됩니다.



<특정 Git 커맨드 사용법 확인>

* **git help**[알고 싶은 커맨드의 이름]
* **man git-**[알고 싶은 커맨드]
* 이 공식 메뉴얼 화면에서 나가고 싶으면 영어 단어 quit(나가다)의 줄임말인 **q**를 입력

<Chapter 2. 커맨드 정리노트>

* git init: 현재 디렉토리를 Git이 관리하는 프로젝트 디렉토리(=working directory)로 설정하고 그 안에 레포지토리(.git 디렉토리) 생성
* git config user.name 'codeit': 현재 사용자의 아이디를 'codeit'으로 설정(커밋할 때 필요한 정보)
* git config user.email 'teacher@codeit.kr': 현재 사용자의 이메일 주소를 'teacher@codeit.kr'로 설정(커밋할 때 필요한 정보)
* git add [파일 이름]: 수정사항이 있는 특정 파일을 staging area에 올리기
* git add [디렉토리명]: 해당 디렉토리 내에서 수정사항이 있는 모든 파일들을 staging area에 올리기
* git add . : working directory 내의 수정사항이 있는 모든 파일들을 staging area에 올리기
* git reset [파일 이름]: staging area에 올렸던 파일 다시 내리기
* git status: Git이 현재 인식하고 있는 프로젝트 관련 내용들 출력(문제 상황이 발생했을 때 현재 상태를 파악하기 위해 활용하면 좋음)
* git commit -m "커밋 메시지": 현재 staging area에 있는 것들 커밋으로 남기기
* git help [커맨드 이름]: 사용법이 궁금한 Git 커맨드의 공식 메뉴얼 내용 출력