02

Git Basic Concepts

Git

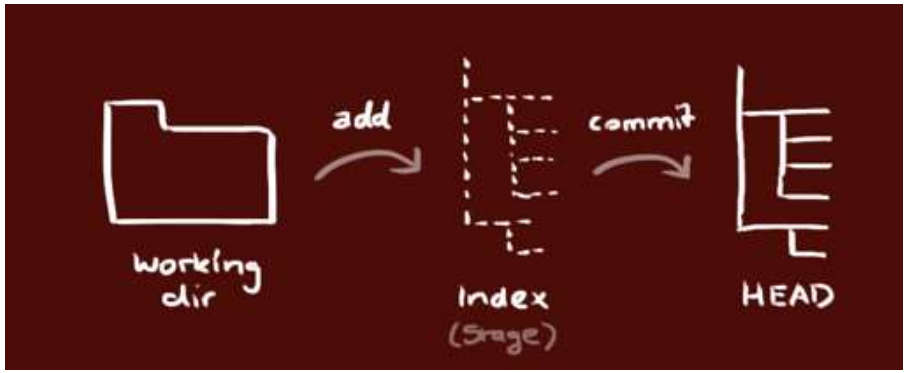
Three Areas

- **로컬 저장소는 3개의 영역으로 구성**
  - Working Directory : 작업 폴더
  - Staging Area (Index) : 준비영역
  - Head : 최종 확정본, Commit

- **Working Directory 는 개발자의 작업 영역, Git 과는 관련이 없다. 즉 Git 에 의해 관리(Tracking)되지 않는다.**
- **Staging Area 로 이동이 되어야 Git 에 의해 관리되기 시작한다.**
- **Head 에 추가되어야 버전관리가 된다.**

Three Areas

- **기본 명령어**
  - add 명령으로 Staging Area 로 이동
  - commit 명령으로 Head 영역으로 이동

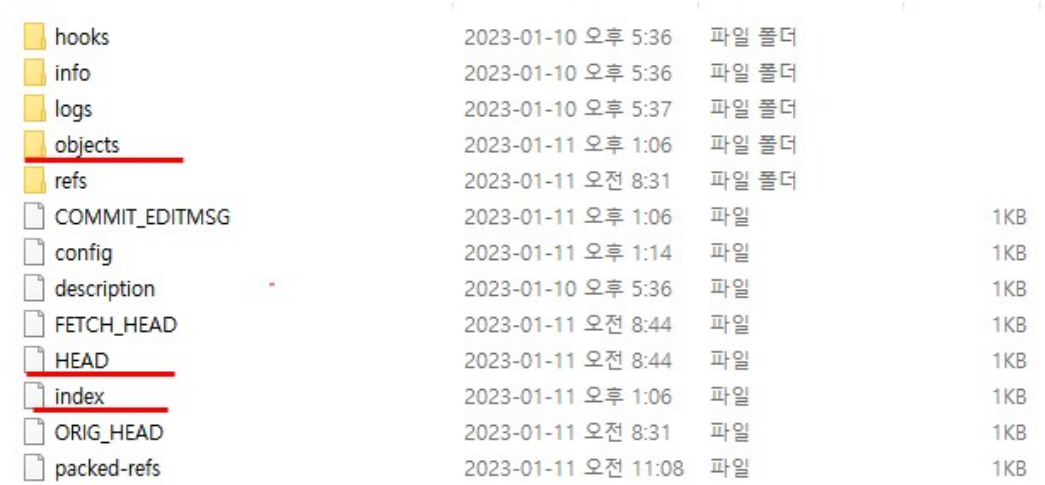


Three States

- **Git 파일은 committed, modified, staged 상태를 가진다.**
  - Committed : 데이터가 로컬 데이터베이스에 안정적으로 저장된 상태
  - Modified : 수정한 파일을 아직 로컬 데이터베이스에 커밋하지 않은 상태
  - Staged : 수정한 파일에 대한 snapshot 을 가지고 있으며 곧 커밋할 것이라고 표시한 상태

Git의 기본 용어

- **Repository :**
  - 파일들을 버전으로 저장해서 가지고 있는 곳.
  - Remote Repos 와 Local Repos



Git의 기본 용어

- **Commit**
  - 현재의 작업을 마치며 저장소에 저장하는 행위

- **Working Directory(Working Tree)**
  - 작업을 하는 디렉토리
  - 작업자의 현재 시점

- **Staging Area (Index, Tracking Area)**
  - 저장소에 커밋하기 전에 커밋을 준비하는 공간
  - Git 에 의해 파일이 관리되기 시작하는 위치

Git의 기본 용어

- **SnapShot**
  - 새로운 버전을 기록하기 위해 commit 하면 스냅샷이 저장
  - 특정 시점에 어떤 파일이 존재 했으며, 폴더 구조, 파일에 어떤 내용이 기록되어 있는지에 대한 모든 정보가 기록
  - Working Dir 에서 작성된 파일들을 Staging Area 에 등록, 이때 Snapshot 가 만들어 진다.
  - Staging Area 는 영구 저장소는 아니다. 그 순간의 파일의 상태에 대한 Snapshot 이 등록되는 곳이다.
  - Staging Area 의 Snapshot 을 영구 불변의 상태로 저장하는 것이 commit 이다.

- **Head**
  - 각 브랜치의 마지막 commit 을 의미
  - 즉 현재의 Staging 의 Snapshot 이 각 브랜치의 마지막 commit(head) 가 되었다는 의미이다.

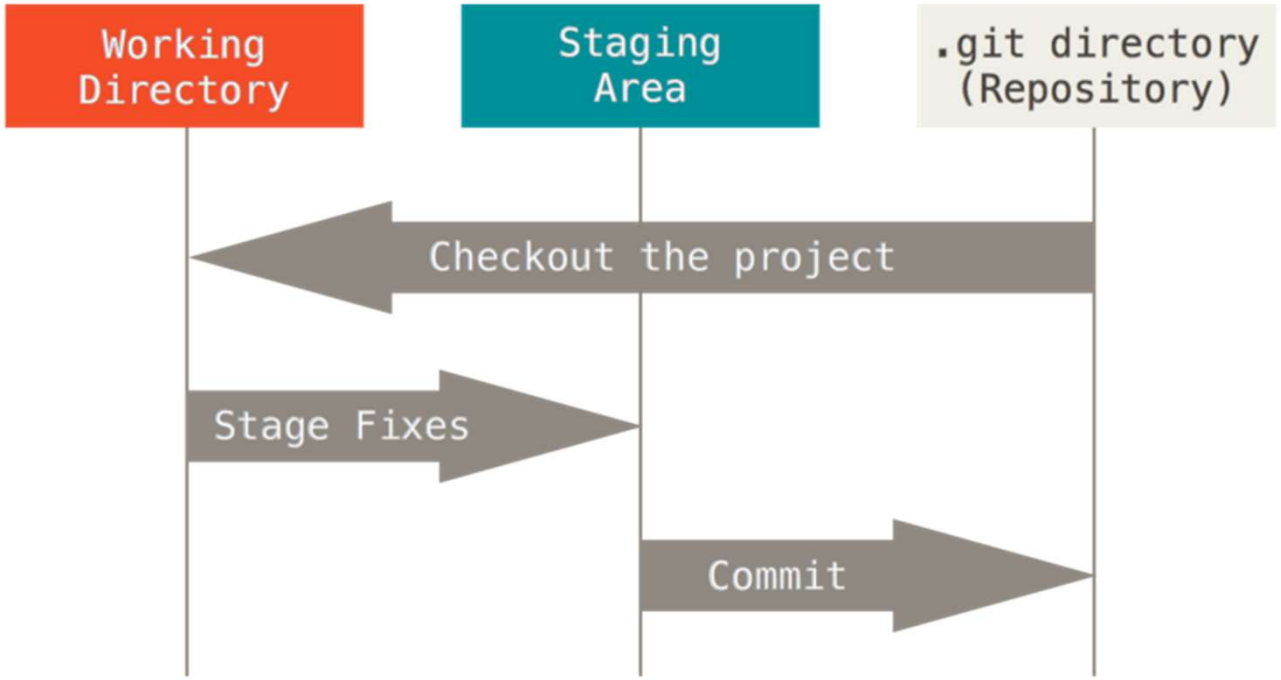
Git의 기본 용어

- **Checkout**
  - 이전 버전 작업을 불러오는 것

- **Branch**
  - 가지 도는 분기점
  - 새로운 작업 흐름
  - 각 브랜치별 Staging Area, head 가 따로 유지

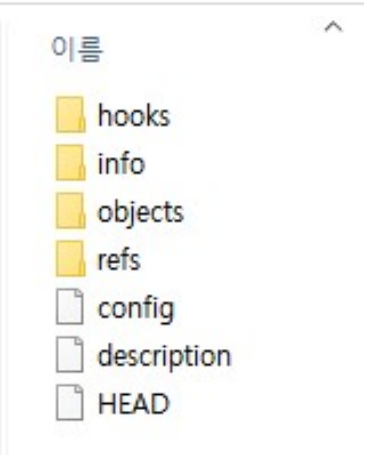
- **Merge**
  - 다른 Branch 의 내용을 현재 Branch로 가져와 합치는 작업

Git의 기본 용어


Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

기초 명령어 (init)

- **저장소**
  - 파일을 Git을 통해 저장하기 위한 임의 디렉토리
  - 작업 디렉토리 혹은 애플리케이션 프로젝트 디렉토리



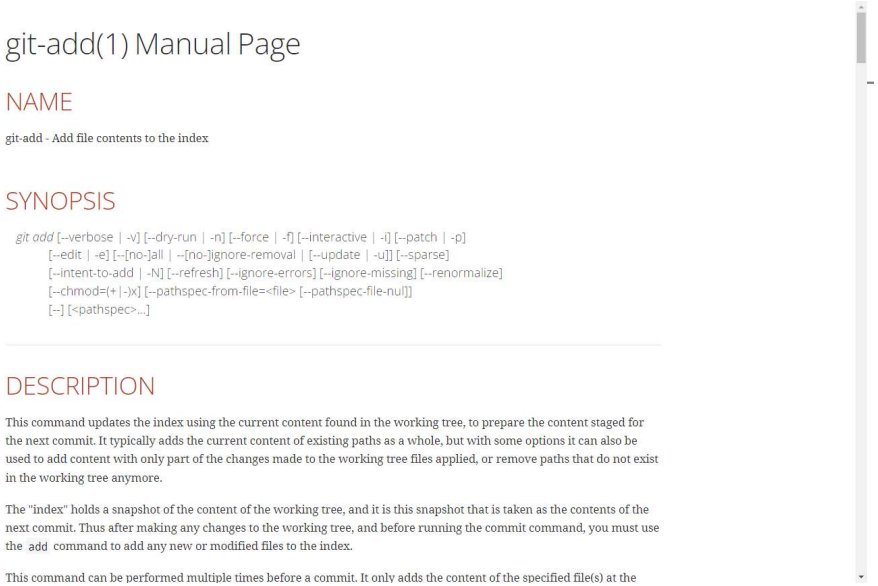
- **git init**
  - 새로운 저장소 만들기
  - .git : git repository 만들때 자동으로 만들어진 폴더. 이 폴더가 git 을 관리하는 툴

```
kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test1
$ git init
Initialized empty Git repository in C:/study-git/test1/.git/

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test1 (main)
$
```

기초 명령어 (help)

- 명령어 사용방법
- git help <명령어>
- git <명령어> --help
- git <명령어> -h



git-add(1) Manual Page

NAME

git-add - Add file contents to the index

SYNOPSIS

git add [--verbose | -v] [--dry-run | -n] [--force | -f] [--interactive | -i] [--patch | -p]
[--edit | -e] [--[no-]all | --[no-]ignore-removal | [--update | -u]] [--sparse]
[--intent-to-add | -N] [--refresh] [--ignore-errors] [--ignore-missing] [--renormalize]
[--chmod=(+|-)x] [--pathspec-from-file=<file> [--pathspec-file-nul]]
[--] [<pathspec>...]

DESCRIPTION

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

The "index" holds a snapshot of the content of the working tree, and it is this snapshot that is taken as the contents of the next commit. Thus after making any changes to the working tree, and before running the commit command, you must use the `add` command to add any new or modified files to the index.

This command can be performed multiple times before a commit. It only adds the content of the specified file(s) at the

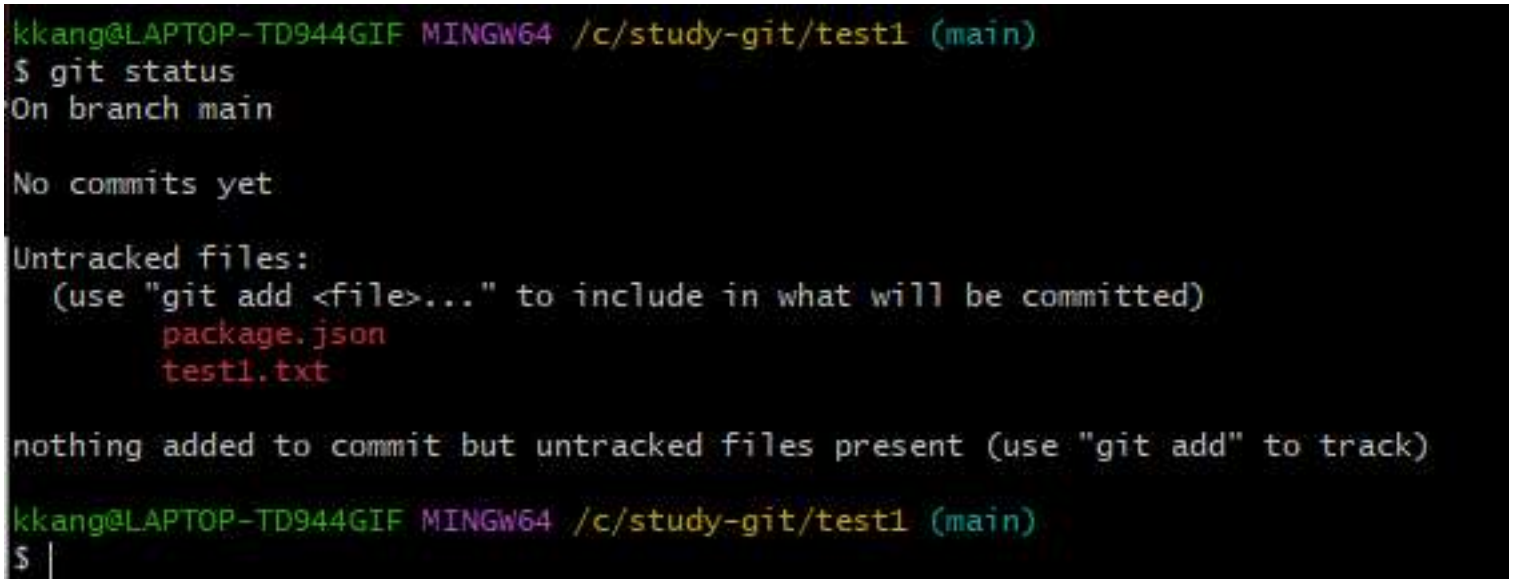
```
kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test18 (main)
$ git add -h
usage: git add [<options>] [--] <pathspec>...

    -n, --dry-run         dry run
    -v, --verbose         be verbose

    -i, --interactive     interactive picking
    -p, --patch           select hunks interactively
```

기초 명령어 (status)

- 상태 파악
- git status
- git status <file>



기초 명령어 (add)

- **Staging Area 에 등록**
  - git add <file>
  - git add *
  - git add .
  - git add <sub folder>/<file>
  - git add <sub folder>/

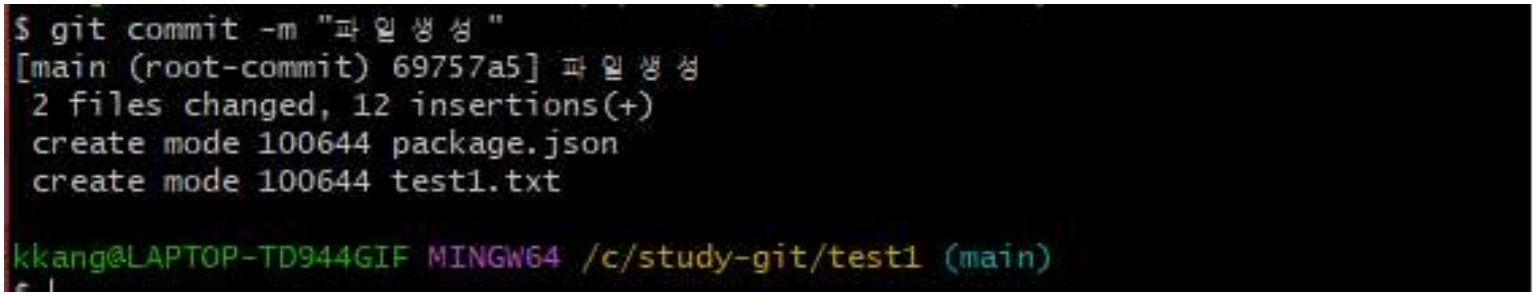
```
$ git add *
warning: in the working copy of 'package.json', LF will be replaced by CRLF the
next time Git touches it

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test1 (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   package.json
        new file:   sub/one/test5.txt
        new file:   sub/test2.txt
        new file:   sub/test3.txt
        new file:   sub/test4.txt
        modified:   test1.txt


kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test1 (main)
```

기초 명령어 (commit)

- SnapShot 을 HEAD 에 추가
- git commit –m “message”



기초 명령어 (log)

- **commit 정보 확인**
  - git log
  - git log --oneline

```
$ git log
commit 14f21467dfdc0a4be9e1957afc743ecd35bec739 (HEAD -> main)
Author: kkang <kkang104@gmail.com>
Date:   Mon Jan 9 16:14:25 2023 +0900

    두 번 째  변 경

commit d73d1b526d7d60548f1cf070b482d51e6ab30d02
Author: kkang <kkang104@gmail.com>
Date:   Mon Jan 9 14:57:10 2023 +0900

    두 번 째  커 밋

commit 69757a5a154cf75013d9786f23e61503838af4e5
Author: kkang <kkang104@gmail.com>
Date:   Mon Jan 9 14:38:42 2023 +0900

    파 일 생 성
$ git log --oneline
69757a5 (HEAD -> main) 파 일 생 성

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test1 (main)
```

기초 명령어 (show)

- **commit 정보 확인**
- **자세한 정보 확인**
  - git show
  - git show <commit hash>
  - git show HEAD
  - git show HEAD^
  - git show HEAD^^
  - git show HEAD~3

restore

- 파일을 이전 commit 내용으로 복구

- git restore <file>
- 파일을 이전 상태로 복구

- git restore --staged <file>
- Staging Area 에 등록된 파일을 untracking 상태로 변경

- git restore --source=<hash> <file>
- 특정 commit 상태로 파일을 복구

reset

- 파일을 이전 commit 내용으로 복구
- restore와 차이점은 HEAD 의 변경 여부

- restore 는 파일 내용을 이전 상태로 변경이 목적이며 commit HEAD 는 조정되지 않는다.
- reset 은 commit 을 삭제하고 HEAD 를 이전 commit 으로 바꾸는 것을 목적으로 한다.

reset

- git reset head^
- 바로 이전 상태로 조정
- 파일내용이 이전으로 복구되지는 않는다.

```
kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test2 (main)
$ git log --oneline
7045b6b (HEAD -> main) 2
66fbbc2 1

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test2 (main)
$ git reset head^
Unstaged changes after reset:
M       a.txt

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test2 (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   a.txt

no changes added to commit (use "git add" and/or "git commit -a")

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test2 (main)
$ git log --oneline
66fbbc2 (HEAD -> main) 1
```

reset

- git reset head^
- git reset head^^
- git reset head~2
- git reset <hash>

- git reset --mixed
- default, 이전 commit 으로 HEAD 가 이동하지만 파일이 복구되지는 않는다.

reset

- git reset --hard
- HEAD 를 조정하면서 파일 내용도 이전으로 복구

```
kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test2 (main)
$ git reset head^
Unstaged changes after reset:
M       a.txt

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test2 (main)
$ git reset head^ --hard
HEAD is now at 66fbbc2 1
```

reset

- git reset –soft
- --mixed 와 동일
- --mixed 는 Working Dir 의 파일이

  unStaged 상태이지만 –soft 는 Staged 상태

```
$ git reset head^
Unstaged changes after reset:
M       main.txt

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test24 (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.txt

no changes added to commit (use "git add" and/or "git commit -a")

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test24 (main)
$ git add .

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test24 (main)
$ git commit
[main 4181150] main commit 2
 1 file changed, 1 insertion(+)

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test24 (main)
$ git reset head^ --soft

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test24 (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.txt
```

amend

- 이전 commit 수정
- git commit --amend

```
kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test2 (main)
$ git log --oneline
66fbbc2 (HEAD -> main) 1

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test2 (main)
$ git commit --amend
[main f9b9718] 1->2
 Date: Tue Jan 31 13:15:32 2023 +0900
 1 file changed, 1 insertion(+)
 create mode 100644 a.txt

kkang@LAPTOP-TD944GIF MINGW64 /c/study-git/test2 (main)
$ git log --oneline
f9b9718 (HEAD -> main) 1->2
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare