

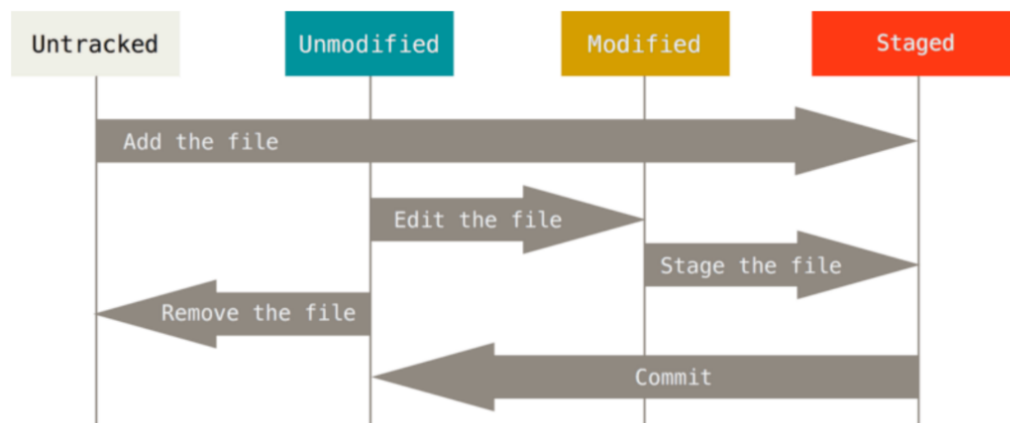
Git

Git 의 특징

- ▼ 버전 관리: 문서 수정 시 언제 수정했는지, 어떤 것을 변경했는지 편리하고 구체적으로 기록하기 위한 버전 관리 시스템
- ▼ 백업: 원격 저장소의 깃허브를 이용하여 깃 파일을 백업
- ▼ 협업: 깃허브를 이용하여 여러 사람이 함께 작업할 수 있도록 충돌을 해결하는 기능 제공

1. Git 버전 관리

- 파일 또는 문서를 관리하기 위한 다양한 방법들 중 한 가지



git → 커맨드 확인

git config --global user.name "???" -> 사용자 이름 설정

git config --global user.email "???@naver.com" -> 사용자의 Email 설정

ls -l → 현재 폴더 안에 무엇이 있는지 알려줌

mkdir ??? -> ??? 이란 이름으로 폴더 만들기

rm -r ??? -> ??? 이름의 폴더 삭제

git init -> 디렉토리 안에 파일을 버전관리 하려고 한다.

2. Git의 3가지 영역

Working Directory : Git이 추적 중인 파일들이 위치하는 영역입니다.

- git init을 통해서 git이 관리하도록 지정된 디렉토리입니다.
- 작업한 파일(생성, 수정한 파일)들이 저장되는 곳입니다.

Staging Area : commit 할 준비가 된 파일들이 위치하는 영역입니다.

- 해당 영역은 .git 디렉토리에 단순한 파일로 존재합니다.
- 작업한(수정된) 파일들 중 버전으로 만들고자(commit 하고자) 하는 파일을 저장합니다.
- git에서는 기술용어로서 "Index"라고 부르기도 합니다.

Git Directory(Repository) : 커밋되어 버전을 관리하는 파일들이 위치하는 영역입니다.


- Git이 프로젝트의 메타데이터와 객체 데이터베이스를 저장하는 곳입니다.
- 프로젝트의 버전 정보를 관리하기 필요로 한 모든 파일이 저장되어 있습니다.

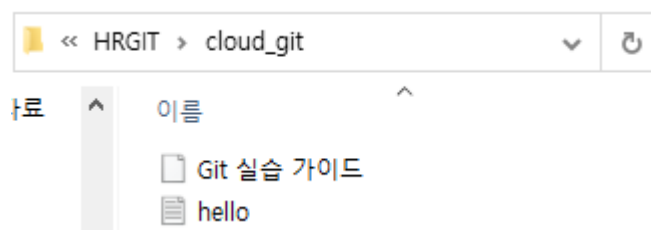
3. 실습 해보기

3-1.

notepad hello.txt → notepad 편집기 사용 hello.txt 생성

```
mzc@DESKTOP-TS7183K MINGW64 /d/HRGIT/cloud_git (harang)
$ notepad hello.txt
```

 hello - Windows 메모장



- 실제로 만들어진 모습 확인

3-2. Untracked : Working Directory에 존재는 하지만 git이 관리를 하지 않는 파일들의 상태입니다.

git status → 상태 체크

```
$ git status
On branch harang
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  hello.txt
```

- 상태를 체크 해본 결과 현재 빨간색으로 "hello.txt" 표시 빨간색으로 나온 이유는 현재 상황에서 내가 체크할 대상이 아님으로 표시 해준다.

3-3. Staged : commit 하고자 하는 파일의 상태입니다.

git add hello.txt → 관리 대상 영역에 "hello.txt" 를 넣어 준다

```
mzc@DESKTOP-TS7183K MINGW64 /d/HRGIT/cloud_git (harang)
$ git add hello.txt

mzc@DESKTOP-TS7183K MINGW64 /d/HRGIT/cloud_git (harang)
$ git status
On branch harang
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   hello.txt
```

- 관리 대상 영역에 hello.txt 를 넣어 주고 다시 상태 체크를 해보니 초록 불로 들어온 모습.

3-4 Modified : 수정을 한 파일의 상태입니다.

git commit -m "hi~~~ hi~~~" → commit 해주고 메시지를 남길수 있다

```
mzc@DESKTOP-TS7183K MINGW64 /d/HRGIT/cloud_git (harang)
$ git commit -m "hi~~~~hi~~~~"
[harang eb1ec52] hi~~~~hi~~~~
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 hello.txt

mzc@DESKTOP-TS7183K MINGW64 /d/HRGIT/cloud_git (harang)
$ git status
On branch harapang
nothing to commit, working tree clean
```

- commit 해주고 나는 현재 상황에서 내가 체크 해야할 파일은 없는 것으로 나온다

3-5

git log → git 에서 내가 작업한 목록 보여줌

```
$ git log
commit eb1ec52827d2f8f2ac971f0ca85cb0ac20c6353e (HEAD -> harapang)
Author: hr <qnfrhatleo@naver.com>
Date: Thu Jul 6 18:49:49 2023 +0900

    hi~~~~hi~~~~
```

- 앞서 내가 commit 한 파일에 메시지 hihi 로 남아 있는 것을 확인 할수 있다.

3-6

cat 명령어 사용의 만들어진 hello.txt 내용 확인 가능

```
mzc@DESKTOP-TS7183K MINGW64 /d/HRGIT/cloud_git (harang)
$ cat hello.txt
내일은 혁수님과 서버웨이 먹기로 할
```

3-7

git commit -am "ha !!! ha~" → 앞에서 한 내용들을 한번에 add 작업 과 commit 작업을 한번에 할 수 있다

```
$ git status
On branch harang
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:   hello.txt

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

mzc@DESKTOP-TS7183K MINGW64 /d/HRGIT/cloud_git (harang)
$ git commit -am "ha !!! ha~"
git commit -am "ha git status ! ha~"
[harang c6b36bf] ha git status ! ha~
1 file changed, 1 insertion(+)

mzc@DESKTOP-TS7183K MINGW64 /d/HRGIT/cloud_git (harang)
$ git status
On branch harang
nothing to commit, working tree clean

mzc@DESKTOP-TS7183K MINGW64 /d/HRGIT/cloud_git (harang)
$ |
```