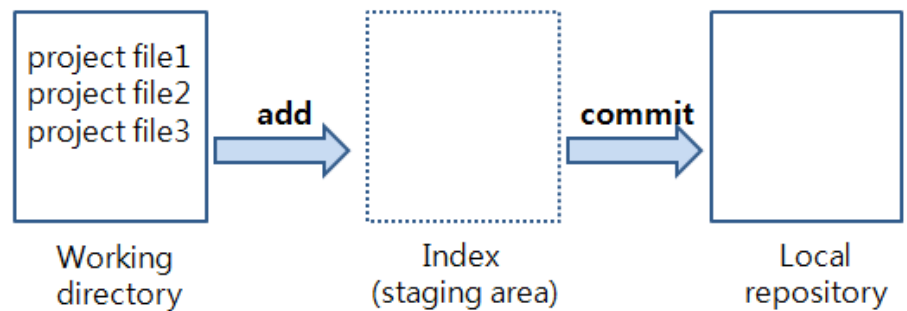


[Git] 명령어(1) – init, add, commit, status

이번 글에서는 git init, git add, git commit 명령어에 대해 알아보도록 하겠습니다.

위의 명령어를 이해하기 위해서는 Git의 구조를 알아야 합니다.

기본 흐름은 다음과 같습니다.



- **git init**
 - 여러 파일을 추적하는 **.git 폴더**가 생성됩니다.
- **git add**
 - Git이 추적하고 있는 수정된 파일이 working directory에서 staging area에 저장됩니다.
- **git commit**
 - staging area에 저장됐던 파일이 local repository로 확정됩니다.
 - commit은 "작업을 마무리 했다"라는 버전 등록을 의미합니다.

위의 구조적 개념을 바탕으로 git 명령어를 다뤄보도록 하겠습니다.

1. 명령어

- **git init**
- **git status**: 상태 확인
- **git add**
 - **--update**: 추적하고 있는 파일만 add
- **git rm**: 제거 관련 명령어
 - **--cached**: add된 파일 제거 (해당 파일을 working directory 상태로 되돌림)
- **git commit**
 - **-m**: 간단한 커밋 메시지 작성
 - **--amend**: 마지막 커밋 메시지 수정
- **git checkout**
 - **--{file}**: working directory에서 작업한 내용을 버림

2. 실습

1) git init

우선 아무 경로에 폴더를 생성하고, 커맨더 창에서 해당 폴더 경로로 이동합니다.

그리고 git이 버전 관리를 시작하도록 `git init` 명령어를 실행합니다.

```
# cd (폴더경로)
# git init
```

```
PS C:\Users\samsung\Desktop\gitTest> git init
Initialized empty Git repository in C:/Users/samsung/Desktop/gitTest/.git/
```

초기화된 **.git** 폴더가 생성되었다고 합니다. (폴더 속성에서 숨김 표시 해제)

.git 폴더에는 다음과 같은 파일/폴더들이 있습니다.

hooks	2018-08-31 오후...	파일 폴더	
info	2018-08-31 오후...	파일 폴더	
objects	2018-08-31 오후...	파일 폴더	
refs	2018-08-31 오후...	파일 폴더	
config	2018-08-31 오후...	파일	1KB
description	2018-08-31 오후...	파일	1KB
HEAD	2018-08-31 오후...	파일	1KB

- working directory
 - .git 폴더의 상위폴더를 말합니다. 예제에서는 gitTest 폴더가 되겠네요.
- .git 폴더
 - working directory 에 존재하는 파일들의 버전을 관리합니다.

2) git status

먼저 상태를 확인하는 `git status` 명령어를 실행해보겠습니다.

```
# git status
```

```
PS C:\Users\samsung\Desktop\gitTest> git status
On branch master

Initial commit



nothing to commit (create/copy files and use "git add" to track)
```

"commit 할 것이 아무 것도 없다"고 합니다.

`git init` 후, working directory에 아무런 작업을 안했기 때문에 당연합니다.

3) git add

다음으로 working directory에 텍스트 파일을 생성하고 아무 글이나 작성해보겠습니다.

 .git	2017-11-27 오전...	파일 폴더	
 victolee.txt	2017-11-27 오전...	텍스트 문서	1KB

그리고나서 상태를 다시 확인해보면,

```
# git status
```

```
PS C:\Users\samsung\Desktop\gitTest> git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        victolee.txt

nothing added to commit but untracked files present (use "git add" to track)
```

"추적되지 않은 파일(Untracked files) 목록"을 보여 보여줍니다.

최근에 등록(commit)된 버전과 비교하여 working directory에 새로운 파일이 추가되면, Git은 처음 보는 파일이므로 추적하고 있지 않은 파일이 됩니다. 따라서 해당 파일을 추적하지 않고 있으므로 git add 명령어를 실행해서 Git이 추적할 수 있도록 해야 합니다.

```
# git add victolee.txt
```

이처럼 git status를 확인하면 Git이 어떤 작업을 해야 좋을지에 대해 알려주기 때문에, 오류가 발생하거나 어떤 상황인지 파악이 안될 경우 status를 확인하는 것이 좋습니다.

또는 git add . 명령어를 실행해도 됩니다.

```
# git add .
```

점(.)은 모든 것(all), 즉 working directory 전체를 의미하는데, all의 사용은 권장하지 않습니다.

중요한 설정 파일과 IDE에서 제공하는 파일들까지 모두 add할 필요는 없기 때문입니다.

사실 이런 파일들은 .gitignore([링크](#))로 관리해야 하지만, 어쨌든 불필요한 파일들을 git이 추적하는 것은 좋지 않습니다.

그런데 일일이 수정된 파일들을 하나씩 입력하는 것은 번거로운데, *를 사용하면 이 불편함을 줄일 수 있습니다.

```
# git add *.txt
```

위 명령어를 실행하면, 모든 txt 파일을 add하겠다는 의미가 됩니다.

디렉토리에 대해서도 사용할 수 있습니다.

```
# git add project/app/*
```

또한 --update 옵션으로 현재 Git이 추적하고 있는 파일들만 add할 수도 있습니다.

```
# git add --update
```

4) git rm

add 명령을 실행한 후, 상태를 다시 확인해보겠습니다.

```
# git status
```

```
PS C:\Users\samsung\Desktop\gitTest> git add victolee.txt
PS C:\Users\samsung\Desktop\gitTest> git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   victolee.txt
```

초록색 문장을 보니, 새로운 파일이 추가되어 추적을 시작했다는 것을 확인할 수 있습니다.

현재 add 명령어만 실행한 상태이므로 staging area에 머물고 있는 상태입니다.

그런데 어떤 파일을 add 하면 안되는데 add한 경우, 파일을 add하기 전 상태인 **unstage** 상태로 만들 수 있습니다.

status에서 알려주는 위의 명령어를 입력한 후에, 다시 상태를 확인해보도록 하겠습니다.

```
# git rm --cached victolee.txt
# git status
```

```
PS C:\Users\samsung\Desktop\gitTest> git rm --cached victolee.txt
rm 'victolee.txt'
PS C:\Users\samsung\Desktop\gitTest> git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        victolee.txt

nothing added to commit but untracked files present (use "git add" to track)
```

staging area에 저장되었던 victolee.txt 파일이 삭제된 것을 확인할 수 있습니다.

즉, Git이 추적하지 않는 untracked file이 돼버렸네요.

이와 같이 add 했을 때, 되돌리고 싶다면 **git rm --cached {파일}** 명령어를 실행하면 됩니다.

git rm --cached 명령어는 Staging area에서만 제거하고 working directory는 유지하는 명령어입니다.

5) git commit

다음 과정을 진행하기 위해 다시 add 하도록 하겠습니다.

```
# git add victolee.txt
```

이제 **git commit** 명령어를 실행해서 버전을 확정짓도록 하겠습니다.

```
# git commit -m "첫 번째 파일 추가"
```

-m 옵션을 사용해서 commit 메시지를 간단하게 작성할 수 있습니다.

```
PS C:\Users\samsung\Desktop\gitTest> git commit -m "첫 번째 파일 추가"
[master (root-commit) 6bc1539] 첫 번째 파일 추가
1 file changed, 1 insertion(+)
create mode 100644 victolee.txt
```

commit을 하면 현재 staging area의 내용을 local repository에 저장하고, **하나의 버전으로** 등록됩니다.

만약 add를 한 상태에서 파일의 내용을 수정해도 add된 것만 버전으로 등록됩니다.

그런데 커밋 메시지를 잘못 작성했을 경우, --amend 옵션으로 다시 커밋하여 메시지를 수정할 수 있습니다.

```
# git commit --amend -m "수정된 메시지"
```

6) git checkout --

commit을 하고 나면 언제든지 특정 버전을 불러올 수 있습니다.

테스트를 위해 victolee.txt 파일을 삭제해보도록 하겠습니다.

그리고 나서 상태를 확인해보면 다음과 같습니다.

```
# git status
```

```
PS C:\Users\samsung\Desktop\gitTest> git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    victolee.txt

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

victolee.txt 파일이 삭제되었다고 합니다.

즉 working directory에 파일이 삭제되었다고 알리는 것이죠.

status에서 알려주는 내용들을 살펴보겠습니다.

현재 상황에서 **git add victolee.txt** 또는 **git rm victolee.txt** 명령어를 실행하면 "내 의도대로 삭제된 것이 맞으니 이대로 staging area에 올리겠다"는 의미입니다.

다음으로 **git checkout -- victolee.txt** 명령어를 실행하면 "그 파일은 실제로 삭제된 것이니 다시 되돌려 놓아라"는 의미입니다.

즉, victolee.txt 파일을 삭제했던 것을 없던일로 만들어 버리는 것으로 gitTest 폴더에 다시 victolee.txt 파일이 생깁니다.

git checkout -- 명령어는 working directory에 작업했던 것을 버리는 명령어입니다.

이상으로 기본이 되는 git init, git add, git commit, git status 명령어에 대해 알아보았습니다.

Git을 잘 다루기 위해서는 항상 status를 확인하는 습관이 중요합니다.

또한 Git의 구조를 이해하여야 git 명령어를 잘 사용할 수 있으니, 구조와 명령어의 연관 관계를 생각하시면 좋을 것 같습니다.