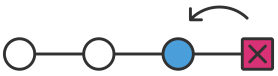
git clean



이 절에서는 git clean 명령에 대해 자세히 설명합니다. 이러한 다른 명령은 이전에 Git 추적 색인에 추가 된 파일에서 작동하지만 git clean 명령은 추적되지 않은 파일에서 작동합니다. 추적 할 수없는 파일은 repo의 작업 디렉토리에서 작성되었지만 아직 git add 명령을 사용하여 저장소의 추적 색인에 추가되지 않은 파일입니다. 추적 된 파일과 추적되지 않은 파일의 차이를보다 잘 보여주기 위해 다음 명령 줄 예제를 참고 하세요.

```
$ mkdir git_clean_test
$ cd git_clean_test/
$ git init .
Initialized empty Git repository in /Users/kev/code/git_clean_test/.git/
$ echo "tracked" > ./tracked_file
$ git add ./tracked_file
$ echo "untracked" > ./untracked_file
$ mkdir ./untracked_dir && touch ./untracked_dir/file
$ git status
On branch master

Initial commit

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

new file: tracked_file

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

이 예제는 git_clean_test 디렉토리에 새로운 Git 저장소를 생성합니다. 그런 다음, Git 색인에 추가 된 tracked_file을 작성하고, untracked_file이 작성되고 untracked_dir이 작성됩니다. 그런 다음 git status를 호출하여 Git의 추적 상태 및 추적되지 않은 변경 상태를 나타내는 출력을 표시합니다. 이 상태의 저장소를 사용하여 git clean 명령을 실행하여 의도 된 목적을 입증 할 수 있습니다.

```
$ git clean fatal: clean.requireForce defaults to true and neither -i, -n,
nor -f given; refusing to clean
```

이 시점에서 기본 git clean 명령을 실행하면 치명적인 오류가 발생할 수 있습니다. 위의 예는 이것이 어떻게 생겼는지를 보여줍니다. 기본적으로 Git은 시작하기 위해 git clean에 "force"옵션이 전달되도록 전역 적으로 구성됩니다. 이는 중요한 안전 장치입니다. 마지막으로 git clean을 실행하면 실행 취소 할 수 없습니다. 완전히 실행되면 git clean은 명령 줄 rm 유틸리티를 실행하는 것과 마찬가지로 하드 파일 시스템을 삭제합니다. untracked 파일을 실행하기 전에 정말로 삭제하고 싶은지 확인하십시오.

일반적인 옵션과 사용법

기본 git 정리 동작 및 경고에 대한 이전 설명이 주어지면 다음 내용은 다양한 git clean 사용 사례와 해당 작업에 필요한 명령 줄 옵션을 보여줍니다.

```
-n
```

-n 옵션은 git clean의 "dry run"을 수행합니다. 이렇게하면 실제로 파일을 제거하지 않고 어떤 파일을 제거 할 것인지 표시됩니다. 항상 제일 먼저 git clean을 수행하는 것이 가장 좋습니다. 이전에 만든 데모 저장소에서이 옵션을 시연 할 수 있습니다.

```
$ git clean -n
Would remove untracked_file
```

git clean 명령을 실행하면 untracked_file이 제거됩니다. untracked_dir은 여기서 출력되지 않습니다. 기본적으로 git clean은 디렉토리에서 재귀 적으로 작동하지 않습니다. 우발적 인 영구 삭제를 방지하는 또 다른 안전 메커니즘입니다.

```
-f or --force
```

force 옵션은 현재 디렉토리에서 untracked 파일의 실제 삭제를 시작합니다. clean.requireForce 구성 옵션을 false로 설정하지 않으면 force가 필요합니다. 이렇게하면 .gitignore에서 지정한 추적되지 않은 폴더 나 파일이 제거되지 않습니다. 이제 우리의 예제 저장소에서 깨끗한 git clean을 실행 해 보겠습니다.

```
$ git clean -f
Removing untracked_file
```

명령은 제거 된 파일을 출력합니다. untracked_file이 제거 된 것을 볼 수 있습니다. 이 시점에서 git status를 실행하거나 ls를 실행하면 untracked_file이 삭제되었고 아무 것도 발견되지 않는다는 것을 알 수 있습니다. 기본적으로 git clean -f는 현재 디렉토리가 아닌 모든 파일에 대해 작동합니다. 또한 특정 파일을 제거하는 -f 옵션과 함께 <path> 값을 전달할 수 있습니다.

```
git clean -f <path>
-d include directories
```

-d 옵션은 git clean에게 추적되지 않은 디렉토리를 지우고 싶다고 알려주며 기본적으로 디렉토리를 무시합니다. 앞의 예제에 -d 옵션을 추가 할 수 있습니다.

```
$ git clean -dn
Would remove untracked_dir/
$ git clean -df
Removing untracked_dir/
```

여기서 우리는 untracked_dir을 출력하는 -dn 조합을 사용하여 'dry run'을 실행했습니다. 그런 다음 강제 정리를 실행하고 untracked_dir이 제거 된 출력을 수신합니다.

```
-x force removal of ignored files
```

일반적인 소프트웨어 릴리스 패턴은 리포지토리 추적 색인에 커밋되지 않은 빌드 또는 배포 디렉토리를 갖는 것입니다. 빌드 디렉토리에는 커밋 된 소스 코드에서 생성 된 임시 빌드 아티팩트가 포함됩니다. 이 빌드 디렉토리는 대개 저장소 .gitignore 파일에 추가됩니다. 추적되지 않은 다른 파일로도이 디렉토리를 정리하는 것이 편리할 수 있습니다. -x 옵션은 git clean에 무시 된 파일을 포함하도록 지시합니다. 이전의 git clean 호출과 마찬가지로, 마지막 삭제 전에 먼저 'dry run'을 실행하는 것이 가장 좋습니다. -x 옵션은 프로젝트 빌드와 관련된 모든 파일에 적용됩니다. 이것은 ./.idea IDE 구성 파일과 같은 의도하지 않은 것일 수 있습니다.

```
git clean -xf
```

-d 옵션과 마찬가지로 -x는 다른 옵션과 함께 전달되고 작성 될 수 있습니다. 이 예제는 -f 옵션과 함께 현재 디렉토리에서 추적되지 않은 파일과 Git이 일반적으로 무시하는 파일을 제거하는 방법을 보여줍니다.

Interactive mode or git clean interactive

지금까지 살펴본 ad-hoc 명령 줄 실행 외에도 git clean에는 -i 옵션을 전달하여 시작할 수있는 "대화 형"모드가 있습니다. 이 문서가 소개 된 예제 repo를 다시 살펴 보겠습니다. 초기 상태에서는 대화식 클린 세션을 시작합니다.

```
$ git clean -di
Would remove the following items:
 untracked_dir/ untracked_file
*** Commands ***
 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6:
help
What now>
```

-d 옵션을 사용하여 대화식 세션을 시작 했으므로 untracked_dir에도 적용됩니다. 대화식 모드는 명령을 추적 할 수없는 파일에 적용 할 것을 요구하는 What now> 프롬프트를 표시합니다. 명령 자체는 상당히 자명합니다. 우리는 명령 6 : help로 시작하여 각각을 임의의 순서로 간략하게 살펴볼 것입니다. 명령 6을 선택하면 다른 명령이 더 설명됩니다.

```
What now> 6
clean - start cleaning
filter by pattern - exclude items from deletion
select by numbers - select items to be deleted by numbers
ask each - confirm each deletion (like "rm -i")
quit - stop cleaning
help - this screen
? - help for prompt selection
```

```
5: quit
```

똑바로 앞으로 대화 형 세션을 종료합니다.

```
1: clean
```

표시된 항목을 삭제합니다. 이 시점에서 1 : clean을 실행하면 untracked_dir / untracked_file이 제거됩니다.

```
4: ask each
```

untracked 각 파일을 반복하고 삭제를 묻는 Y / N 프롬프트를 표시합니다. 다음과 같이 보입니다.

```
*** Commands ***
 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6:
help
What now> 4
Remove untracked_dir/ [y/N]? N
Remove untracked_file [y/N]? N
```

```
2: filter by pattern
```

Will display an additional prompt that takes input used to filter the list of untracked files.

```
Would remove the following items:
 untracked_dir/ untracked_file
*** Commands ***
 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6:
help
What now> 2
 untracked_dir/ untracked_file
Input ignore patterns>> *_file
 untracked_dir/
```

여기서 우리는 * _file 와일드 카드 패턴을 입력하여 untracked 파일 목록을 untracked_dir로 제한합니다.

```
3: select by numbers
```

명령 2와 마찬가지로 명령 3은 추적되지 않은 파일 이름의 목록을 수정합니다. 대화식 세션에서는 추적 할 수없는 파일 이름에 해당하는 번호를 묻습니다.

```
Would remove the following items:
 untracked_dir/ untracked_file
*** Commands ***
 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6:
help
What now> 3
 1: untracked_dir/ 2: untracked_file
Select items to delete>> 2
 1: untracked_dir/ * 2: untracked_file
Select items to delete>>
Would remove the following item:
 untracked_file
*** Commands ***
 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6:
help
```