1.6 시작하기 - 최초 Git 설정

1. 모든 설정값 확인

```
$ git config --list
```

```
$ git config user.name
```

* 특정값 확인

2. 모든 설정값과 해당 위치 확인

```
$ git config --list --show-origin
```

3. 사용자 이름 및 이메일 설정

```
$ git config --global user.name "Yera Lee"
$ git config --global user.email
```

* global 옵션을 전달할 경우 한번만 수행해도 됨.

4. 설정값 지우기

```
$ git config --unset user.name
$ git config --unset user.email

$ git config --unset --global user.name
$ git config --unset --global user.email
```

* global 로 설정한 경우 global로 지워야 함

5. 편집기 설정

```
$ git config --global core.editor emacs
```

```
$ git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -
multiInst -notabbar -nosession -noPlugin"
```

* Windows 시스템에서 다른 텍스트 편집기를 사용하려면 실행 파일의 전체 경로를 지정해야 함.

6. 기본 Branch 설정

```
$ git config --global init.defaultBranch main
```

왜 안되는지 모르겠음

1.7

1. 모든 명령어 리스트 출력

```
git
```

2.1 Git 기본 - Git 리포지토리 가져오기

1. 로컬 디렉토리 -> GIt 리포지토리 (전환)

```
$ cd <dirpath>

$ git init
```

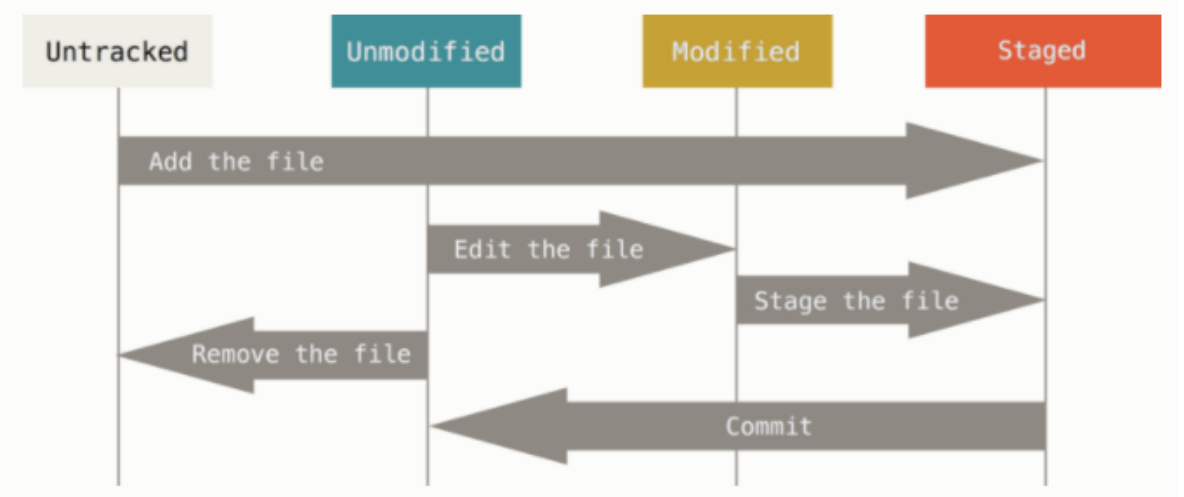
>> .git 디렉토리가 생성됨
이 디렉토리 안에 버전관리에 관련된 여러가지 정보가 저장됨

2. Git 리포지토리 복제

```
$ git clone <url>
$ git clone <url> <dirname>
```

2.2 Git 기본 - 변경사항 기록

모든 파일은 trackted 또는 untracked 의 두 가지 상태 중 하나
tracked 파일의 상태 : Unmodified, Modified, Staged



1. 스테이지에 올리기

```
git add [FILENAME]
```

2. 현재 상태 확인

```
git status
git status -s
```

* -s : 간단하게 보는 옵션

3. 수정사항 비교 (Staged & Modfied)

```
git diff
```

4. Staged 취소

```
git rm --cached [FILENAME]
```

5. 수정사항 비교 (Staged & Umodified)

```
git diff --staged
git diff --cached
```

* staged와 cached 는 동의어

6. Commit 하기 (→ Unmodified)

```
git commit
git commit -m "INLINE COMMIT MESSAGE"
git commit -a
```

-m : inline 커밋메세지 옵션
-a : stage를 건너뛰고 추적하는 파일 모두 commit

예제

1. testfile.txt 생성 후 상태 확인

```
$ git status -s
```

?? testfile.txt

2. 스테이지에 올린 후 (Untracked → Staged) 상태 확인

```
$ git add testfile.txt

$ git status -s
```

A testfile.txt

3. testfile.txt 수정 후 (Staged, Modified) 상태 확인

```
$ git status -s
```

AM testfile.txt

4. 수정된 내용 비교 (Staged & Modified)

```
$ git diff
```

diff --git a/testfile.txt b/testfile.txt
index 18084ee..453b147 100644
--- a/testfile.txt
+++ b/testfile.txt
@@ -1 +1,2 @@
-oh no
₩ No newline at end of file
+oh no
+oh yes
₩ No newline at end of file

5. 수정된 내용 비교 (Unmodified & Staged)

```
$ git diff --staged
```

diff --git a/testfile.txt b/testfile.txt
new file mode 100644
index 0000000..18084ee
--- /dev/null
+++ b/testfile.txt
@@ -0,0 +1 @@
+oh no
₩ No newline at end of file
>> Unmodified 상태의 파일이 없어서 표시 안됨

6. 스테이지에 올린 후 (Modified → Staged) 상태 확인

```
$ git add testfile.txt

$ git status -s
```

A testfile.txt

7. 스테이지에서 내린 후, 다시 올리기 (Staged → Untracked → Staged)

```
$ git rm --cached testfile.txt
rm 'testfile.txt'

$ git status -s
?? testfile.txt

$ git add testfile.txt

$ git status -s
A  testfile.txt
```

8. testfile 수정 후(Staged&Modified 상태), 강제로 stage에서 내리기

```
$ git status -s
AM testfile.txt

$ git rm --cached testfile.txt
error: the following file has staged content different from both the
file and the HEAD:
    testfile.txt
(use -f to force removal)

$ git rm --cached -f testfile.txt
rm 'testfile.txt'

$ git status -s
?? testfile.txt
```

* -f 옵션 : 변경사항을 커밋하지 않았을 경우에 강제로 삭제

9. testfile 스테이지에 올린 후 커밋

```
$ git add testfile.txt

$ git commit

$ git status -s
```

10. testfile 수정 후, 상태 확인, 그리고 Staged를 거치지 않고 바로 커밋 (→Unmodified)

```
$ git status -s
 M testfile.txt

$ git commit -a -m "바로 commit"
[main 76169d2] 바로 commit
 1 file changed, 2 insertions(+), 1 deletion(-)
```

```
$ git status -s

```

11. testfile 삭제 후, 커밋

```
$ git rm testfile.txt
rm 'testfile.txt'

$ git status -s
D  testfile.txt

$ git commit -a -m "파일 삭제"
[main 6ff800f] 파일 삭제
 1 file changed, 4 deletions(-)
 delete mode 100644 testfile.txt
```

2.2.1 Git 기본 - .gitignore

.gitignore : 자동으로 추적되지 않는 파일 패턴 목록

1. .gitignore 파일 생성
2. 추적하지 않을 파일 이름 패턴 추가 후 저장

패턴 규칙

- 빈 줄이나 #로 시작하는 줄은 무시
- 표준 glob 패턴이 작동, 전체 작업 트리에 재귀적으로 적용
- 슬래시(/)로 패턴을 시작할 경우, 재귀를 피함
- 슬래시(/)로 패턴을 종료하여 디렉토리를 지정
- 느낌표(!)로 시작하여 패턴을 부정

예

```
# ignore all .a files
*.a

# but do track lib.a, even though you're ignoring .a files above
```

```
!lib.a

# only ignore the TODO file in the current directory, not subdir/TODO
/TODO

# ignore all files in any directory named build
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory and any of its subdirectories
doc/**/*.pdf
```

2.3 Git 기본 - 커밋 기록 보기

https://git-scm.com/book/en/v2/Git-Basics-Viewing-the-Commit-History

```
git log
```

```
git log -p -{N}
```

원하는 갯수만큼 볼 수 있으면서 각 커밋에 도입된 차이점을 같이 볼 수 있음.

```
git log --stat
```

각 커밋에 대한 간략한 통계 볼 수 있음.

```
git log --pretty=oneline
```

각 커밋을 한 줄에 인쇄

```
git log --pretty=format:"%h - %an, %ar : %s"
```

아래처럼 원하는 형식으로 출력

```
ca82a6d - Scott Chacon, 6 years ago : Change version number
085bb3b - Scott Chacon, 6 years ago : Remove unnecessary test
a11bef0 - Scott Chacon, 6 years ago : Initial commit
```

```
git log --pretty=format:"%h %s" --graph
```

분기 및 병합을 보여주는 그래프 형식으로 출력

```
git log --since=2.weeks
```

시간제한 옵션

```
git log --pretty="%h - %s" --author='Junio C Hamano' --since="2008-10-01" \
   --before="2008-11-01" --no-merges -- t
```

여러 옵션

format 옵션 사용시 지정자

지정자	출력 설명
%H	커밋 해시
%h	축약된 커밋 해시
%T	트리 해시
%t	축약된 트리 해시
%P	상위 해시
%p	축약된 상위 해시
%an	저자 이름
%ae	작성자 이메일
%ad	작성자 날짜(형식은 --date=옵션을 따릅니다)
%ar	저자 날짜, 친척
%cn	커미터 이름
%ce	커미터 이메일
%cd	커미터 날짜
%cr	커미터 날짜, 상대
%s	주제

2.4 Git 기본 - 실행 취소

1. 어떤 파일 빼먹고 커밋했을 때

```
$ git add forgotten_file

$ git commit --amend
```

2.