.gitignore

- git 저장소에서 관리할 필요가 없는 파일이나 폴더 지정
- .gitignore sample :
  https://gist.github.com/octocat/9257657
- .gitignore 자동생성 사이트
  https://www.gitignore.io/
- 아래 내용으로 .gitignore 생성 -> 파일 생성 ->git status 확인!

```
# object file
*.o

# log file
*.log
```

git commands : cancel

- git add 취소 (unstage 로 변경)

```
git reset HEAD [file]
```

- 최근 Commit 메시지 변경

```
git commit --amend
```

- Add로 index에 넣은 파일 삭제

```
git rm --cached [file]
```

- Untracked 파일 삭제

```
git clean -n -f
git clean -f
git clean -f -d
git clean -f -d -x
```

git commands : cancel

- **Reset : commit 취소**

```
git reset [option] [commit id]
```

1. **mixed** : default, index 초기화, WD(Working Directory) 파일 보존

```
git reset --mixed a3bb3c
```

2. **hard** : 지정한 commit id 이후의 이력 및 파일 모두 지움, index 초기화

```
git reset --hard a3bb3c
```

3. **soft** : index 보존, WD 파일 보존, 모두 보존
   Commit하면 원래 상태로 복원 가능

```
git reset --soft a3bb3c
git reset HEAD~6
git reset HEAD^^
```

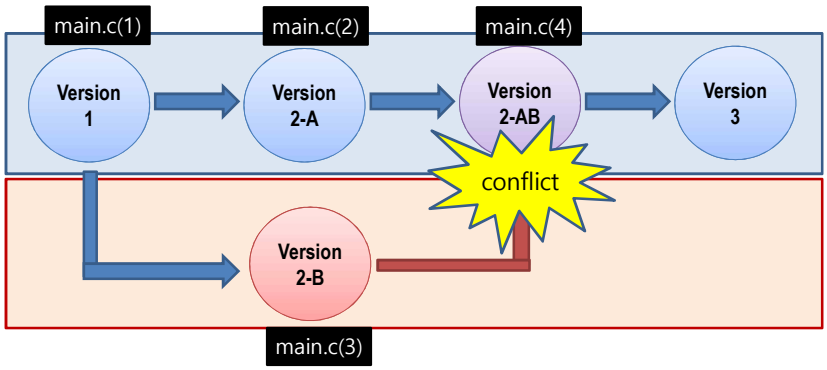
git commands : merge

- **Merge 브랜치 통합**

```
git merge [branch name]
```

```
jerry1004@peace:~/project/gitRepo:> git branch func_A
jerry1004@peace:~/project/gitRepo:> git checkout func_A
Switched to branch 'func_A'
jerry1004@peace:~/project/gitRepo:> touch func.c
jerry1004@peace:~/project/gitRepo:> git checkout -
Switched to branch 'master'
jerry1004@peace:~/project/gitRepo:> git branch
  func_A
* master
jerry1004@peace:~/project/gitRepo:> git merge func_A
```

Git merge conflicts



Git merge conflicts

Version 1

```
$ mkdir git_merge_test
$ cd git_mer*
$ git init
Initialized empty Git repository in /home/jerry1004/project
$ echo "V1 : This is some content for merging" > merge.txt
$ ls
merge.txt
$ git add merge.txt
$ git commit -m "version 1 : merge.txt "
[master (root-commit) 984dfc2] version 1 : merge.txt
 1 file changed, 1 insertion(+)
 create mode 100644 merge.txt
```

Git merge conflicts

Version 2-B

```
$ git checkout -b new_branch
M       merge.txt
Switched to a new branch 'new_branch'
$ echo "V2 : totally different content to merge later" > merge.txt
$ git status
On branch new_branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working dir

        modified:   merge.txt

no changes added to commit (use "git add" and/or "git commit -a")
$ git commit -a -m "version 2: different content(in new_branch)"
[new_branch ec20f98] version 2: different content(in new_branch)
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Git merge conflicts

Version 2-A

```
$ git checkout master
Switched to branch 'master'
$ echo "V2-A: content to append" >> merge.txt
$ cat merge.txt
V1 : This is some content for merging
V2-A: content to append
$ git commit -am "version2-A: appended content (in master)"
[master 561b227] version2-A: appended content (in master)
 1 file changed, 1 insertion(+)
```

Git merge conflicts

Version 2AB : merge (master <= new_branch)

conflict

```
$ git branch
* master
  new_branch
$ git merge new_branch
Auto-merging merge.txt
CONFLICT (content): Merge conflict in merge.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
$ cat merge.txt
<<<<<<< HEAD
V1 : This is some content for merging
V2-A: content to append
=======
V2 : totally different content to merge later
>>>>>>> new_branch
```

Git merge conflicts

Version 3 : edit the conflicted file

<<<<<<< HEAD
V1 : This is some content for merging
V2-A: content to append
=======
V2 : totally different content to merge later
>>>>>>> new_branch

V1 : This is some content for merging
V3: The most direct way to resolve a merge conflict is to edit the conflicted file

```
$ git commit -am "merged and resolved the conflict in merge.txt"
[master 6483d38] merged and resolved the conflict in merge.txt
$ git status
On branch master
nothing to commit, working directory clean
```