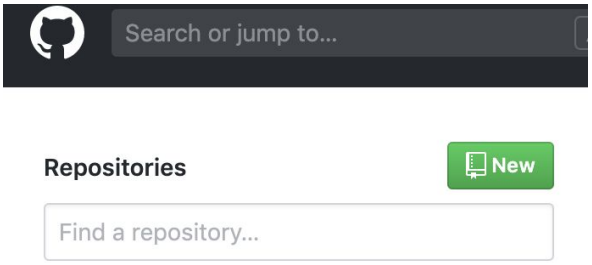
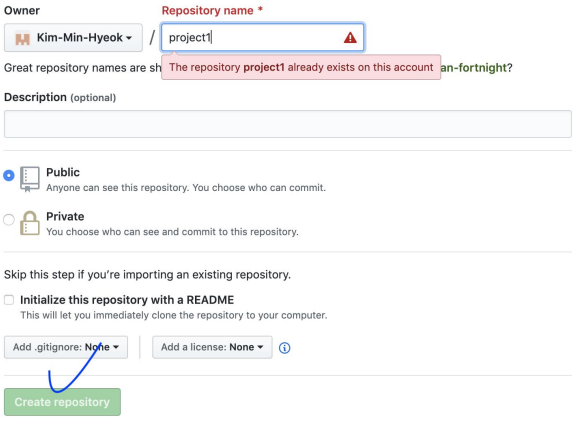
<6주차> #7주차는 밑에 이어져서 작성되어 있습니다.

git 명령어 정리 및 기본 연습 수행(레파지토리 생성과 첫번째 파일을 올리는 것은 맥에서, 수정본을 올리는 것은 putty를 이용함)

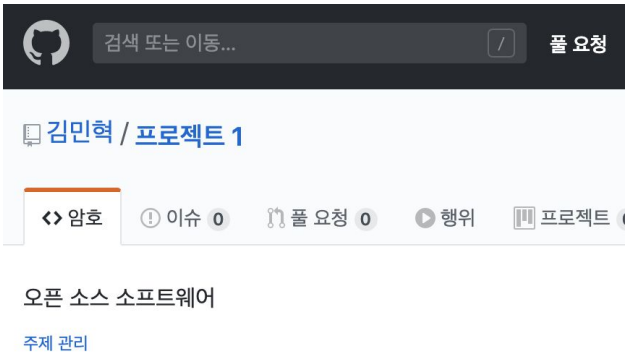
1. repository 생성



> github에 들어가서 다음과 같은 창의 New를 누른다.



> Repository의 이름을 적고 Create repository를 누른다.(이미 project1 을 생성해서 다음과 같이 오류가 난다)



> 다음과 같이 새 Repository가 생성된다.

> 다음과 같이 간단한 git 명령어도 제공하여 준다.

2. 현재 디렉토리에 .git 디렉토리 생성

```
[s21900102@peace:~/OSS2020/crud$ git init
 Initialized empty Git repository in /home/s21900102/OSS2020/crud/.git/
```

> 비어있는 깃 레파지토리가 현재 디렉토리(crud)에 생겼다.

3. 로컬 저장소에 소스파일 등록

```
[s21900102@peace:~/OSS2020/crud$ git add *c *h makefile *txt
```

> 임시 로컬 저장소에 현재 디렉토리에 있는 모든 .c파일과 .h파일, Makefile과 .txt파일을 저장하였다.

4. 로컬 저장소 등재(commit)

```
[s21900102@peace:~/OSS2020/crud$ git commit -m "first commit"
[master (root-commit) e5b3c86] first commit
 7 files changed, 670 insertions(+)
 create mode 100644 Criminal_List.txt
 create mode 100644 Criminal_Report.txt
 create mode 100644 Criminal_Wicked.txt
 create mode 100644 criminal.c
 create mode 100644 criminal.h
 create mode 100644 main.c
 create mode 100644 makefile
```

> commit을 이용하여 첫번째 커밋 커멘트를 달아서 일곱개의 파일을 등재하였다.

5. push할 repository 세팅

```
[s21900102@peace:~/OSS2020/crud$ git remote add origin http://github.com/Kim-Min-
Hyeok/project1.git
```

> push할 repository주소와, 올릴 이름을 origin으로 설정해주었다.

6. push

```
[s21900102@peace:~/OSS2020/crud$ git push -u origin master
[Username for 'https://github.com': git status
[Password for 'https://git status@github.com':
 remote: Invalid username or password.
 fatal: Authentication failed for 'https://github.com/Kim-Min-Hyeok/project1.git/
 '
```

> 커밋한 것을 push하였다.

*수정본 올리기

```
s21900102@peace:~/OSS2020/crud$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Criminal_List.txt
        modified:   Criminal_Report.txt
        modified:   criminal.c
        modified:   criminal.h
        modified:   main.c

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

        User_List.txt
        criminal.o
        main

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

> 현재 깃의 상태이다. 파일들을 이번 주 실습 미션에 따라 수정하고 레파지토리에 올리지 않고, add하지 않아서 stage area에 있지 않은 파일들이 빨간색으로 보인다. 또한 새롭게 만들어진 파일들이 밑에 나와있다.

```
s21900102@peace:~/OSS2020/crud$ git add *c *h *txt
```

> 위에서 권장한 것 처럼 git add를 이용하여 모든 .c, .h, .txt파일들을 stage area에 올려놓았다.

```
s21900102@peace:~/OSS2020/crud$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   Criminal_List.txt
        modified:   Criminal_Report.txt
        new file:   User_List.txt
        modified:   criminal.c
        modified:   criminal.h
        modified:   main.c

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

        criminal.o
        main
```

> 현재 깃의 상태이다. make 명령어를 실행하면서 생긴 .o파일과 main을 제외하고 모두 stage area에 올라간 것을 볼 수 있다.

```
s21900102@peace:~/OSS2020/crud$ git commit -m "Third version"
[master 735fe0e] Third version
 6 files changed, 334 insertions(+), 9 deletions(-)
 create mode 100644 User_List.txt
```

> 프로젝트 3번째 기능 구현이라 Third version이라는 커멘트로 깃허브에 커밋하였다.

```
s21900102@peace:~/OSS2020/crud$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        criminal.o
        main

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

> 현재 깃의 상태이다. stage area에 있던 파일들이 커밋되고 add가 되지 않은 파일들만 보이는 상태이다.

```
s21900102@peace:~/OSS2020/crud$ git log
commit 735fe0e9fdba33873fc395286e7d2443b2edcc98
Author: Kim-Min-Hyeok <21900102@handong.edu>
Date:   Sat Apr 11 23:58:23 2020 +0900

    Third version

commit e5b3c86f1db83658d742ca5e04113c34bbc460b2
Author: Kim-Min-Hyeok <21900102@handong.edu>
Date:   Tue Apr 7 13:09:56 2020 +0900

    first commit
```

> 깃의 기록을 출력한다.위와 같이 전에 올린 first commit과 방금 올린 Third version의 기록을 볼 수 있다.

```
s21900102@peace:~/OSS2020/crud$ git log --oneline
735fe0e Third version
e5b3c86 first commit
s21900102@peace:~/OSS2020/crud$
```

> -oneline이라는 옵션으로 각 기록마다 7자리의 코드를 받았다.

```
s21900102@peace:~/OSS2020/crud$ git diff 735fe0e:e5b3c86
```

> 위 두 코드를 이용하여 두 버전의 다른 점을 출력하였다.

```
diff --git a/Criminal_List.txt b/Criminal_List.txt
index 97e6635..e69de29 100644
--- a/Criminal_List.txt
+++ b/Criminal_List.txt
@@ -1,3 +0,0 @@
-kim 21 men Daegu killing x x x x 1
-park 99 men Busan sexual killing x x x 2
-bang 20 men Busan killing x x x x 1
diff --git a/Criminal_Report.txt b/Criminal_Report.txt
index 7e3d7b0..963812b 100644
--- a/Criminal_Report.txt
+++ b/Criminal_Report.txt
@@ -1 +1 @@
-20,30's: 2 30,40's: 0 older: 1 men: 3 women: 0
+20,30's: 0 30,40's: 0 older: 0 men: 0 women: 0
diff --git a/User_List.txt b/User_List.txt
deleted file mode 100644
index 13c514d..0000000
--- a/User_List.txt
+++ /dev/null
@@ -1 +0,0 @@
-min11203 18181818a
diff --git a/criminal.c b/criminal.c
:
```

> 다음은 출력결과이다.

*이외 log명령어 옵션들

1. --pretty=oneline

```
s21900102@peace:~/OSS2020/crud$ git log --pretty=oneline
735fe0e9fdba33873fc395286e7d2443b2edcc98 Third version
e5b3c86f1db83658d742ca5e04113c34bbc460b2 first commit
```

> 커밋 기록을 oneline형태로 출력한다.(=format)

2. --graph

```
s21900102@peace:~/OSS2020/crud$ git log --graph
* commit 735fe0e9fdba33873fc395286e7d2443b2edcc98
| Author: Kim-Min-Hyeok <21900102@handong.edu>
| Date:   Sat Apr 11 23:58:23 2020 +0900
|
|     Third version
|
* commit e5b3c86f1db83658d742ca5e04113c34bbc460b2
  Author: Kim-Min-Hyeok <21900102@handong.edu>
  Date:   Tue Apr 7 13:09:56 2020 +0900

      first commit
```

> 커밋 기록을 그래프형태로 출력한다.

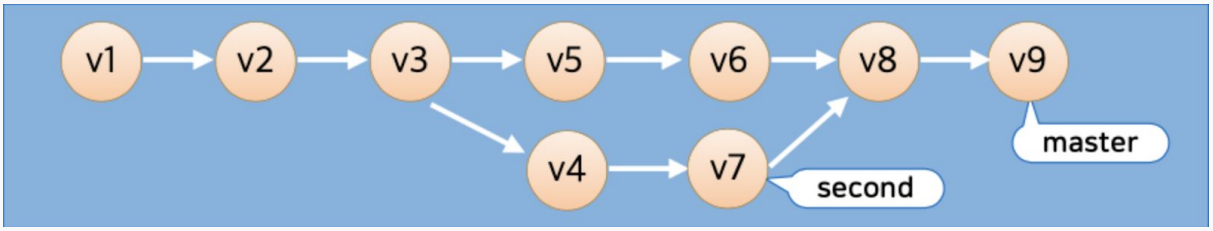
```
s21900102@peace:~/OSS2020/crud$ git log -p -1
```

```
commit 735fe0e9fdba33873fc395286e7d2443b2edcc98
Author: Kim-Min-Hyeok <21900102@handong.edu>
Date:   Sat Apr 11 23:58:23 2020 +0900

    Third version

diff --git a/Criminal_List.txt b/Criminal_List.txt
index e69de29..97e6635 100644
--- a/Criminal_List.txt
+++ b/Criminal_List.txt
@@ -0,0 +1,3 @@
+kim 21 men Daegu killing x x x x 1
+park 99 men Busan sexual killing x x x 2
+bang 20 men Busan killing x x x x 1
diff --git a/Criminal_Report.txt b/Criminal_Report.txt
index 963812b..7e3d7b0 100644
--- a/Criminal_Report.txt
+++ b/Criminal_Report.txt
@@ -1 +1 @@
-20,30's: 0 30,40's: 0 older: 0 men: 0 women: 0
+20,30's: 2 30,40's: 0 older: 1 men: 3 women: 0
diff --git a/User_List.txt b/User_List.txt
new file mode 100644
```

> 각 커밋의 diff결과를 보여준다.(-p)

> 최근 하나의 커밋의 정보를 보여준다.(-1)

<7주차>



1. v1(version1)

1) git 저장소 생성 >> git init(.git 생성)

```
[s21900102@peace:~/gitlab2$ git init
Initialized empty Git repository in /home/s21900102/gitlab2/.git/
```

```
[s21900102@peace:~/gitlab2$ ls -al
total 12
drwxrwxr-x 3 s21900102 s21900102 4096  4월  20 02:36 .
drwx------ 8 s21900102 s21900102 4096  4월  20 02:35 ..
drwxrwxr-x 7 s21900102 s21900102 4096  4월  20 02:36 .git
```

다음과 같이 git 저장소가 생성되었다.

2) my.txt(ver1)생성

```
[s21900102@peace:~/gitlab2$ cat > my.txt
[ver1
```

3) add & commit(+nano > vim)

```
add: command not found
[s21900102@peace:~/gitlab2$ git add my.txt
[s21900102@peace:~/gitlab2$ git status
On branch master

Initial commit

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

        new file:   my.txt
```

다음과 같이 git status로 현재 git 상태를 확인해 본 결과 my.txt가 로컬 저장소 임시공간에 등록되어 commit을 기다리고 있다.

```
  GNU nano 2.5.3    File: /home/s21900102/gitlab2/.git/COMMIT_EDITMSG

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# Explicit paths specified without -i or -o; assuming --only paths...
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   my.txt
#
```

다음과 같이 git commit 편집 에디터가 nano로 되어있어 먼저 vim으로 고쳐보도록 하겠다. (nano에디터는 컨트롤+x로 나간다)

```
[s21900102@peace:~/gitlab2$ git commit my.txt
 Aborting commit due to empty commit message.
[s21900102@peace:~/gitlab2$ git config --global core.editor "vim"
```

commit message를 설정하지 않아서 commit이 실패하였다. 다시 바뀐 에디터로 commit message를 설정해보도록 하겠다.

```
[s21900102@peace:~/gitlab2$ git commit my.txt
[master (root-commit) 8818a6f] -ver1
 1 file changed, 1 insertion(+)
 create mode 100644 my.txt
[s21900102@peace:~/gitlab2$ git log --oneline
8818a6f -ver1
```

다음과 같이 commit을 성공하였다. 실수로 편집 후 캡쳐를 안했으므로 다음 commit에서 캡쳐하여 보여주겠다.

2. v2(version2)

1) 수정

```
[s21900102@peace:~/gitlab2$ vim my.txt
```

```
ver2
~
```

ver1 > ver2 로 my.txt를 수정하였다.

2) add & commit

```
[s21900102@peace:~/gitlab2$ git add my.txt
[s21900102@peace:~/gitlab2$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   my.txt
```

다음과 같이 my.txt를 add하자 수정되었다고 뜬다.

```
- ver2
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#       modified:   my.txt
#
~
~
```

아까 바꾼 결과 nano에디터가 아닌 vim에디터가 뜨고 commit 메시지를 맨 첫줄에 - ver2로 편집하여 주었다.

```
[s21900102@peace:~/gitlab2$ git commit
[master 7e19929] - ver2
 1 file changed, 1 insertion(+), 1 deletion(-)
```

3. v3(version3)

2번과 마찬가지로 my.txt를 ver3으로 수정하여 add & commit한다.

```
[s21900102@peace:~/gitlab2$ vim my.txt
[s21900102@peace:~/gitlab2$ git add my.txt
[s21900102@peace:~/gitlab2$ git commit -m - ver3
error: pathspec 'ver3' did not match any file(s) known to git.
[s21900102@peace:~/gitlab2$ git commit -m "- ver3"
[master 71f4fb1] - ver3
 1 file changed, 1 insertion(+), 1 deletion(-)
```

-m 옵션 : commit 메시지를 “”를 이용하여 설정할 수 있다.

4. v4(version4)-branch

```
[s21900102@peace:~/gitlab2$ git branch second
```

second라는 브랜치를 git branch를 이용하여 생성하였다.

```
[s21900102@peace:~/gitlab2$ git branch
 * master
   second
[s21900102@peace:~/gitlab2$ git checkout second
 Switched to branch 'second'
[s21900102@peace:~/gitlab2$ git branch
   master
 * second
```

git branch를 이용하여 현재 branch위치를 확인하고 생성한 second로 git checkout을 이용하여 이동하였다.

```
[s21900102@peace:~/gitlab2$ touch second.txt
[s21900102@peace:~/gitlab2$ git add second.txt
[s21900102@peace:~/gitlab2$ git status
 On branch second
 Changes to be committed:
   (use "git reset HEAD <file>..." to unstage)

         new file:   second.txt

[s21900102@peace:~/gitlab2$ git commit -m "- ver4"
 [second 193b660] - ver4
  1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 second.txt
```

touch를 이용하여 second branch에 second.txt를 만들었다. 또한 add & commit을 하였다.(커밋 메시지 = - ver4)

5. v5(version5)

```
[s21900102@peace:~/gitlab2$ git branch
   master
 * second
[s21900102@peace:~/gitlab2$ git checkout master
 Switched to branch 'master'
[s21900102@peace:~/gitlab2$ git branch
 * master
   second
```

git branch를 통하여 현재 branch위치를 확인하였고, checkout을 통하여 master로 branch를 이용하였다.

```
[s21900102@peace:~/gitlab2$ touch master.txt
[s21900102@peace:~/gitlab2$ add master.txt
```

```
[s21900102@peace:~/gitlab2$ git add master.txt
[s21900102@peace:~/gitlab2$ git commit -m "- ver5"
[master ce0c34e] - ver5
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 master.txt
```

v4와 같이 이번에는 master.txt를 master branch에 만들고 add & commit하였다.(커밋 메시지=- ver5)

6. v6(version6)

```
[s21900102@peace:~/gitlab2$ cat > hello.txt
[hi
```

master에 hello.txt를 생성하여 hi라고 입력하였다.

```
[s21900102@peace:~/gitlab2$ git add hello.txt
[s21900102@peace:~/gitlab2$ git commit -m "- ver6"
[master 6ec7c1d] - ver6
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt
```

hello.txt를 add & commit하였다.

7. v7(version7)

```
[s21900102@peace:~/gitlab2$ git branch
* master
  second
[s21900102@peace:~/gitlab2$ git checkout second
Switched to branch 'second'
[s21900102@peace:~/gitlab2$ cat > hello.txt
[bye
```

위의 v6와 같이 second에 똑같이 hello.txt를 생성하였고, 이번에는 bye를 입력하였다.

```
[s21900102@peace:~/gitlab2$ git add hello.txt
[s21900102@peace:~/gitlab2$ git commit -m "- ver7"
[second e52837e] - ver7
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt
```

add와 commit을 하였다.(커밋 메시지=-ver7)

8. v8(version8)-merge

```
[s21900102@peace:~/gitlab2$ git branch
  master
* second
[s21900102@peace:~/gitlab2$ git checkout master
Switched to branch 'master'
[s21900102@peace:~/gitlab2$ git merge second
Auto-merging hello.txt
CONFLICT (add/add): Merge conflict in hello.txt
Automatic merge failed; fix conflicts and then commit the result.
[s21900102@peace:~/gitlab2$ ls
hello.txt  master.txt  my.txt  second.txt
```

다음과 같이 master에서 second를 merge할 것이기 때문에 checkout을 통하여 branch를 master로 이동하였고, git merge를 이용하여 second를 merge하였다. 밑의 파일들은 merge한 결과인 master의 파일들이다.

그러나 master와 second에 이름이 같은 파일인 hello.txt가 있으므로 충돌이 일어났다.(merge 실패)

```
[s21900102@peace:~/gitlab2$ cat hello.txt
hi-line from master
bye-line from second
```

그래서 vim 에디터를 이용하여 hello.txt를 다음과 같이 고쳤다.

```
<<<<<<< HEAD
hi
=======
bye
>>>>>>> second
~
```

원래 hello.txt파일은 다음과 같이 merge가 충돌되어 있었다.

```
[s21900102@peace:~/gitlab2$ git add hello.txt
[s21900102@peace:~/gitlab2$ git commit -m "- ver8"
[master 750d532] - ver8
[s21900102@peace:~/gitlab2$ git log --oneline --graph
*   750d532 - ver8
|\
| * e52837e - ver7
| * 193b660 - ver4
* | 6ec7c1d - ver6
* | ce0c34e - ver5
|/
* 71f4fb1 - ver3
* 7e19929 - ver2
* 8818a6f -ver1
```

hello.txt를 수정하고 난 후 다시 merge를 하여 merge를 성공시켰다. git log의 --grahp 옵션을 이용하여 살펴보면 다음과 같다.

```
[s21900102@peace:~/gitlab2$ git branch
* master
  second
[s21900102@peace:~/gitlab2$ cat > week7_mission.txt
[ver9
[s21900102@peace:~/gitlab2$ git add week7_mission.txt
[s21900102@peace:~/gitlab2$ git commit -m "- ver8"
[master aa639d3] - ver8
 1 file changed, 1 insertion(+)
 create mode 100644 week7_mission.txt
[s21900102@peace:~/gitlab2$ git log --oneline --graph
* aa639d3 - ver8
*   750d532 - ver8
|\
| * e52837e - ver7
| * 193b660 - ver4
* | 6ec7c1d - ver6
* | ce0c34e - ver5
|/
* 71f4fb1 - ver3
* 7e19929 - ver2
* 8818a6f -ver1
```

다음과 같이 week7_mission.txt를 만들어 ver9을 만들었고 commit 하였다.

7주차 2차시 수업내용

>> 그런데 실수로 커밋 메시지를 ver8으로 설정하였다.

```
[s21900102@peace:~/gitlab2$ git reset 750d532
Unstaged changes after reset:
M       hello.txt
[s21900102@peace:~/gitlab2$ ls
hello.txt  master.txt  my.txt  second.txt
[s21900102@peace:~/gitlab2$ git log --oneline --graph
*   750d532 - ver8
|\
| * e52837e - ver7
| * 193b660 - ver4
* | 6ec7c1d - ver6
* | ce0c34e - ver5
|/
* 71f4fb1 - ver3
* 7e19929 - ver2
* 8818a6f -ver1
```

`git reset`을 이용하여 바로 전 단계인 `ver8`까지 `commit`한 것으로 되돌렸다.

```
[s21900102@peace:~/gitlab2$ cat > week7_mission.txt
[ver9
[s21900102@peace:~/gitlab2$ git add week7_mission.txt
[s21900102@peace:~/gitlab2$ git commit -m "- ver9"
[detached HEAD 9522305] - ver9
 1 file changed, 1 insertion(+)
 create mode 100644 week7_mission.txt
[s21900102@peace:~/gitlab2$ git log --oneline --graph
* 9522305 - ver9
*   750d532 - ver8
|\
| * e52837e - ver7
| * 193b660 - ver4
* | 6ec7c1d - ver6
* | ce0c34e - ver5
|/
* 71f4fb1 - ver3
* 7e19929 - ver2
* 8818a6f -ver1
```

다음과 같이 다시 `add & commit` 하여 `ver9`을 생성하였다.
그런데 지금 보니 `v1`의 커밋 메시지가 띄어쓰기가 잘못된 듯 하다.

```
[s21900102@peace:~/gitlab2$ git rebase -i --root
```

```
reword 8818a6f -ver1
pick 7e19929 - ver2
pick 71f4fb1 - ver3
pick ce0c34e - ver5
pick 6ec7c1d - ver6
pick 193b660 - ver4
pick e52837e - ver7
pick aa639d3 - ver8
```

그래서 다음과 같이 rebase를 하여서 reword기능을 이용하고자 한다.
바꾸고자 하는 커밋 메시지의 버전의 pick을 reword로 수정한다.
그러면 v1의 commit 편집으로 넘어가게 된다.

```
- ver1

# Please enter the commit messa
# with '#' will be ignored, and
#
# Date:      Mon Apr 20 02:58:5
#
# interactive rebase in progres
# Last command done (1 command
#    reword 8818a6f -ver1
```

그래서 다음과 같이 맨 첫 줄에 있는 커밋 메시지를 수정하였다.

```
[s21900102@peace:~/gitlab2$ git log --oneline
05ece7f - ver4
799dd59 - ver6
9a26239 - ver5
1e469f9 - ver3
9ceb9cb - ver2
6fd9b65 - ver1
```

그래서 다음과 같이 ver1의 커밋 메시지가 알맞게 변경되었다.

*revert

```
s21900102@peace:~/gitlab$ git log --oneline --graph
*   712e121 merge hello.txt
|\
| * 532581d hello_ver2
* | 03eb662 hello_master
* |   8e02b29 Merge branch 'ver_2'
|\ \
| |/
| * 4a0be4f 시나리오2_ver_2
* | cdfdeaa 시나리오2_master
|/
* 04412e8 from ver2_시나리오1
* 06a6194 ver 2
* 96b6882 ver 1
```

이번에는 revert를 실험 해 보기 위하여 다른 디렉토리를 열었다.

```
s21900102@peace:~/gitlab$ cat my.txt
ver 2.0
#시나리오1
```

<원래 버전>

```
s21900102@peace:~/gitlab$ cat my.txt
ver 2.0
#revert text
```

<바뀐 버전>

```
s21900102@peace:~/gitlab$ git add my.txt
s21900102@peace:~/gitlab$ git commit -m "revert test"
[master d71d2d9] revert test
 1 file changed, 1 insertion(+), 1 deletion(-)
```

>> 그래서 다음과 같이 바뀐 버전을 - revert test라는 커밋 메시지로 커밋 하였다.

이번에는 git revert를 이용하여 이전 commit으로 돌아온 것 처럼 해보겠다.

다음장의 캡쳐를 보면 my.txt의 내용이 바뀐 것을 확인할 수 있고, commit 기록도 알 수 있다.

```
2ef10e6 Revert "revert test"
d71d2d9 revert test
712e121 merge hello.txt
532581d hello_ver2
03eb662 hello_master
8e02b29 Merge branch 'ver_2'
4a0be4f 시나리오2_ver_2
cdfdeaa 시나리오2_master
04412e8 from ver2_시나리오1
06a6194 ver 2
96b6882 ver 1
s21900102@peace:~/gitlab$ cat my.txt
ver 2.0
#시나리오1
```