CLI 환경에서의 tag, rebase, amend 등

태그

- tag 생성
  - git tag A
    - 현재 브랜치의 최신커밋에 A라는 이름의 태그를 담
    - ex) git tag v1.0.0
- tag 조회
  - git tag
    - 로컬저장소의 모든 태그 조회

태그

- tag push
  - git push origin A
    - 태그 A를 push함
- tag 삭제
  - git tag –d A
    - 로컬저장소의 태그 A를 삭제함
  - git tag --delete A
    - 동일함

태그 실습

- 아래와 같이 커밋을 한 후 push까지 진행함

파일내용	GitFile1.txt
Commit3	Contents3
Commit2	Contents2
Commit1	Contents1

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --oneline --graph
* 316d31d (HEAD -> master, origin/master) Commit3
* 963f4f9 Commit2
* a597443 Commit1
```

- Commit3 커밋에 v1.0.0라는 태그를 담
  - git tag v1.0.0

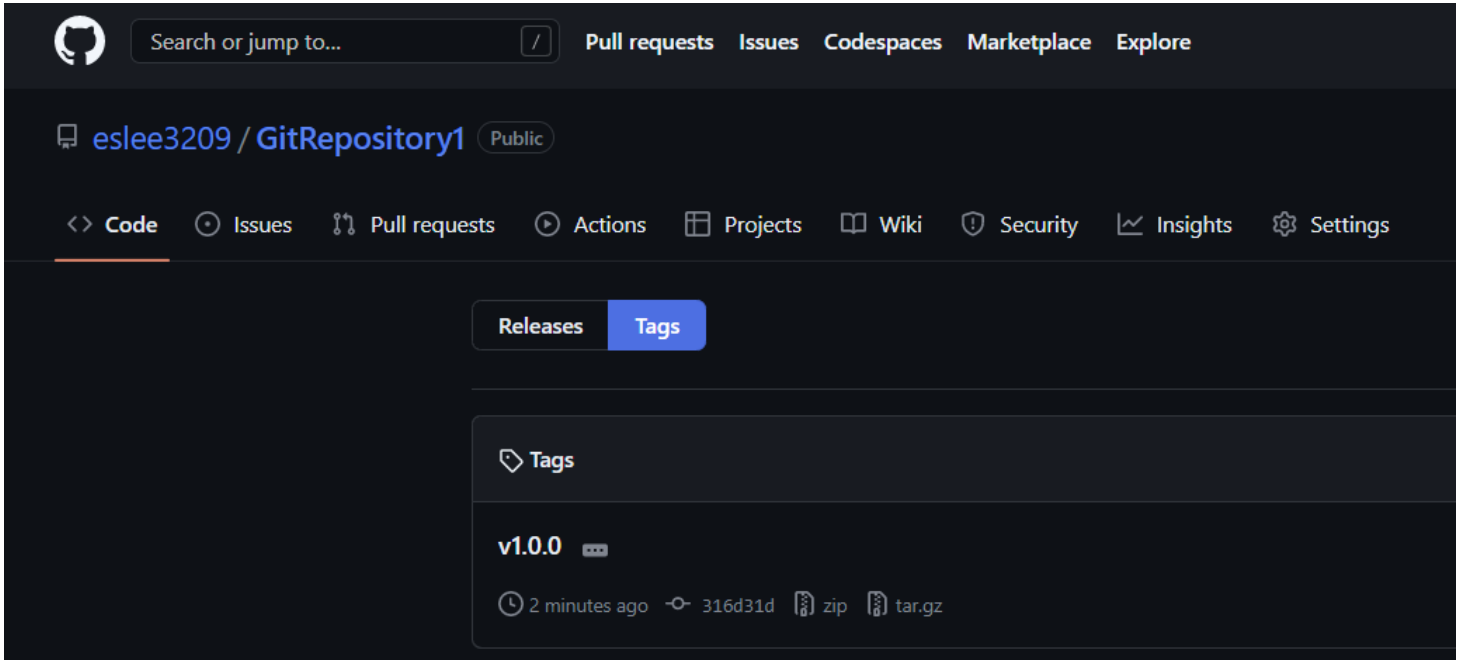
```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git tag v1.0.0

USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --oneline --graph
* 316d31d (HEAD -> master, tag: v1.0.0, origin/master) Commit3
* 963f4f9 Commit2
* a597443 Commit1
```

태그 실습

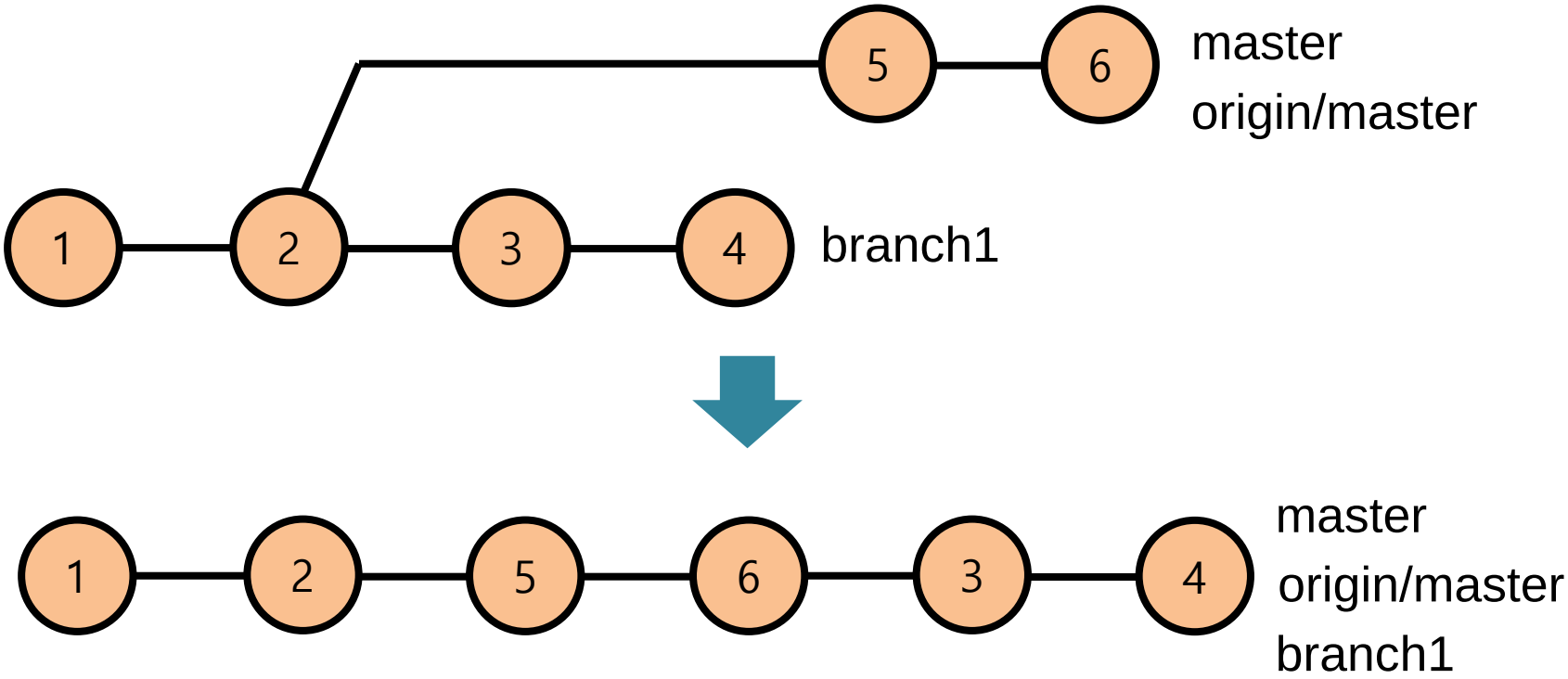
- 방금만든 tag v1.0.0을 원격저장소에 올림
  - git push origin v1.0.0

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git push origin v1.0.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/eslee3209/GitRepository1.git
 * [new tag]         v1.0.0 -> v1.0.0
```



rebase

- 브랜치 A를 베이스로 하여 브랜치 B를 rebase함
  - git checkout B
  - git rebase A



rebase 실습

- 아래와 같이 커밋을 한 후 push까지 진행함
  - master 브랜치가 Commit6를, branch1 브랜치가 Commit4를 가리킴

파일내용	GitFile1.txt	GitFile2.txt
Commit6	Contents2	Contents2
Commit5	Contents2	Contents1
Commit4	Contents4	X
Commit3	Contents3	X
Commit2	Contents2	X
Commit1	Contents1	X

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1)
$ git log --all --oneline --graph
* 842a700 (origin/master, master) Commit6
* 0ceaf65 Commit5
| * 0b0dd50 (HEAD -> branch1, origin/branch1) Commit4
| * 316d31d Commit3
|/
* 963f4f9 Commit2
* a597443 Commit1
```

rebase 실습

- master 브랜치를 base로 하여 branch1 브랜치를 rebase
  - git checkout branch1
  - git rebase master

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1)
$ git rebase master
Successfully rebased and updated refs/heads/branch1.

USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1)
$ git log --all --oneline --graph
* 374a5cf (HEAD -> branch1) Commit4
* beaa100 Commit3
* 842a700 (origin/master, master) Commit6
* 0ceaf65 Commit5
| * 0b0dd50 (origin/branch1) Commit4
| * 316d31d Commit3
|/
* 963f4f9 Commit2
* a597443 Commit1
```

rebase 실습

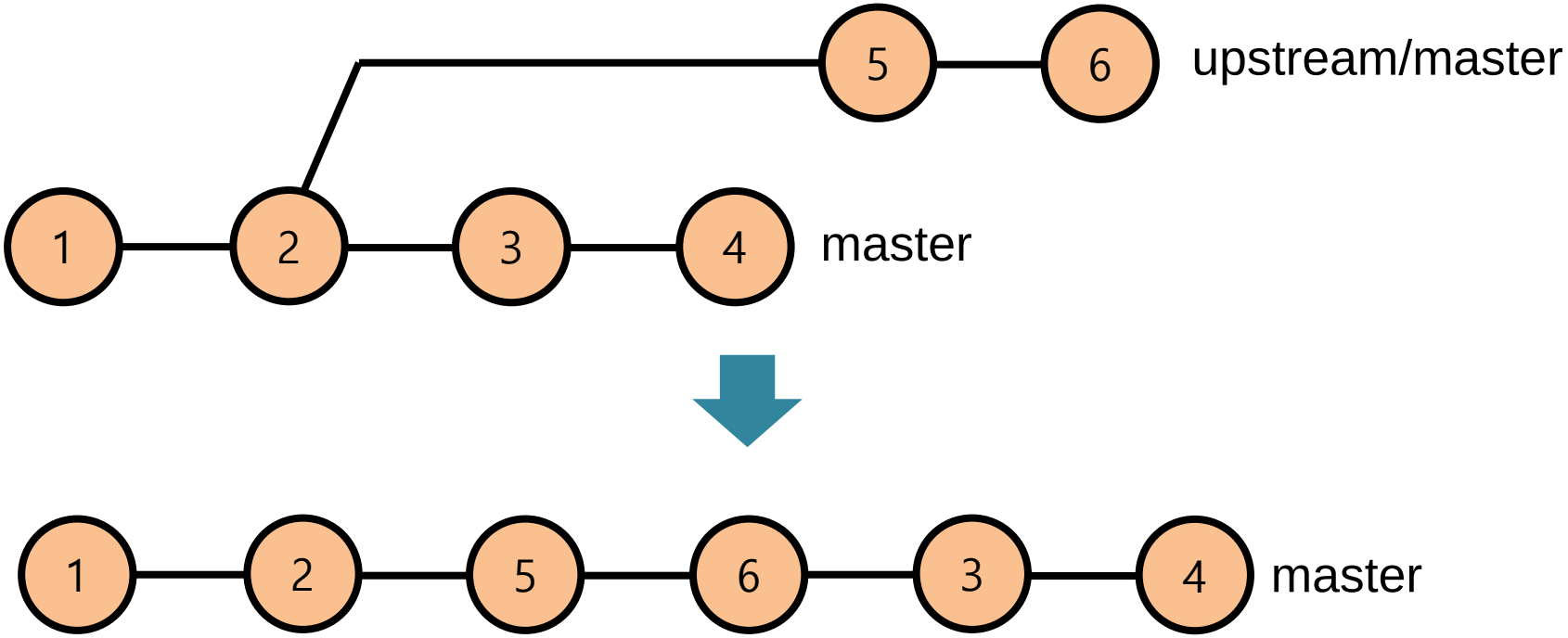
- branch1 브랜치를 강제 push함
  - git push -u origin branch1 --force

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1)
$ git push -u origin branch1 --force
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 523 bytes | 523.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/eslee3209/GitRepository1.git
 + 0b0dd50...374a5cf branch1 -> branch1 (forced update)
branch 'branch1' set up to track 'origin/branch1'.

USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1)
$ git log --all --oneline --graph
* 374a5cf (HEAD -> branch1, origin/branch1) Commit4
* beaa100 Commit3
* 842a700 (origin/master, master) Commit6
* 0ceaf65 Commit5
* 963f4f9 Commit2
* a597443 Commit1
```

rebase (fork)

- 원본저장소의 master 브랜치를 base로 하여 원격저장소의 master 브랜치를 rebase



rebase (fork)

- 상황
  - 주계정 원본저장소를 부계정으로 fork함
  - 부계정으로 fork한 로컬저장소에 들어갔을 때 다음 그래프가 된 상황
  - 아직, 원본저장소의 Commit5, Commit6이 그래프에 나타나지 않은 상황

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectoryFork1 (master)
$ git log --all --oneline --graph
* 4c5b820 (HEAD -> master, origin/master, origin/HEAD) Commit4
* 6f24503 Commit3
* 963f4f9 Commit2
* a597443 Commit1
```

rebase (fork)

- 절차
  - git remote add upstream [원본저장소 주소]
    - ex) git remote add upstream https://github.com/eslee3209/GitRepository1.git
  - git fetch upstream
    - 그러면 아래와 같이 원본저장소 최신 커밋을 가져옴

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectoryFork1 (master)
$ git log --all --oneline --graph
* 4c5b820 (HEAD -> master, origin/master, origin/HEAD) Commit4
* 6f24503 Commit3
| * 842a700 (upstream/master) Commit6
| * 0ceaf65 Commit5
|/
* 963f4f9 Commit2
* a597443 Commit1
```

rebase (fork)

- 절차
  - git rebase upstream/master

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectoryFork1 (master)
$ git log --all --oneline --graph
* f9e86cf (HEAD -> master) Commit4
* dea344b Commit3
* 842a700 (upstream/master) Commit6
* 0ceaf65 Commit5
| * 4c5b820 (origin/master, origin/HEAD) Commit4
| * 6f24503 Commit3
|/
* 963f4f9 Commit2
* a597443 Commit1
```

  - git push origin master --force

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectoryFork1 (master)
$ git log --all --oneline --graph
* f9e86cf (HEAD -> master, origin/master, origin/HEAD) Commit4
* dea344b Commit3
* 842a700 (upstream/master) Commit6
* 0ceaf65 Commit5
* 963f4f9 Commit2
* a597443 Commit1
```

rebase (fork)

- 절차
  - GitHub에서 부계정으로 fork한 원격저장소로 들어가 pull request
  - GitHub에서 주계정으로 원본저장소로 들어가 pull request를 승인함
  - fork 로컬저장소로 돌아옴
  - git fetch upstream

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectoryFork1 (master)
$ git log --all --oneline --graph
*   9481e56 (upstream/master) Merge pull request #5 from eslee4478/master
|\
| * f9e86cf (HEAD -> master, origin/master, origin/HEAD) Commit4
| * dea344b Commit3
|/
* 842a700 Commit6
* 0ceaf65 Commit5
* 963f4f9 Commit2
* a597443 Commit1
```

amend

- 로컬저장소에만 올린 최신커밋 A에 대해 파일을 수정하고 amend (커밋 이름 B)
  - VSCode로 파일 작업함
  - git add .
  - git commit --amend -m "B"
    - ex) git commit --amend -m "Commit2"
  - git push origin master --force
    - 현재 브랜치가 master 브랜치일 때, amend한 결과를 원격저장소에도 반영

amend 실습

- 상황
  - 아래와 같은 상황

파일내용	GitFile1.txt
Commit2	Contents2
Commit1	Contents1

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* 963f4f9 (HEAD -> master, origin/master) Commit2
* a597443 Commit1
```

amend 실습

- 절차
  - VSCode로 현재 GitFile1.txt의 내용을 Contents3로 수정
  - git add . (혹은 git add GitFile1.txt)
  - git commit --amend -m "Commit2"
  - git push origin master --force

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* 576009d (HEAD -> master) Commit2
| * 963f4f9 (origin/master) Commit2
|/
* a597443 Commit1
```

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* 576009d (HEAD -> master, origin/master) Commit2
* a597443 Commit1
```

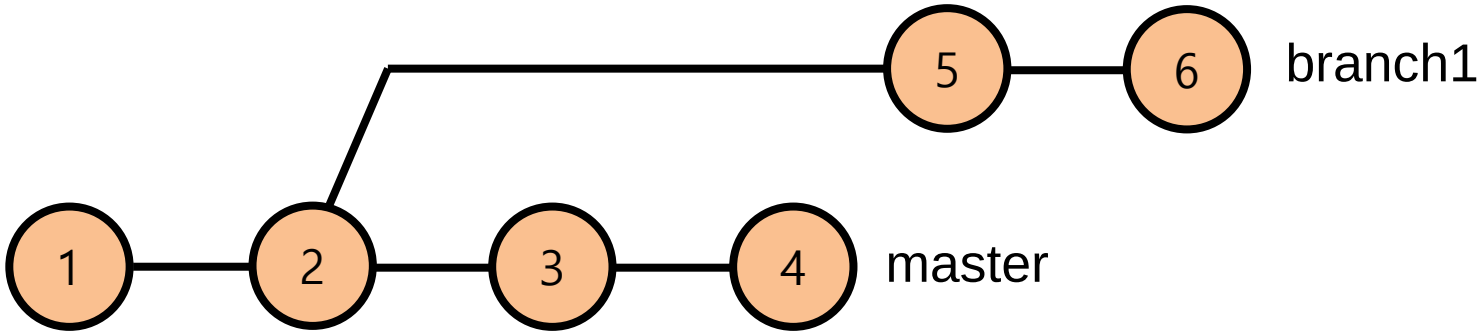
cherry-pick

- 다른 브랜치의 원하는 커밋A(체크섬)을 브랜치B에 최신 커밋으로 반영함
  - git checkout B
  - git cherry-pick A

cherry-pick 실습

- 아래와 같은 상태에서 시작함

파일내용	GitFile1.txt	GitFile2.txt
Commit6	Contents2	Contents2
Commit5	Contents2	Contents1
Commit4	Contents4	
Commit3	Contents3	
Commit2	Contents2	
Commit1	Contents1	




```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1)
$ git log --all --oneline --graph
* 0a93ffe (HEAD -> branch1, origin/branch1) Commit6
* 2723b48 Commit5
| * da50c85 (origin/master, master) Commit4
| * 4ff5d23 Commit3
|/
* f7d2f74 Commit2
* a597443 Commit1
```

cherry-pick 실습

- master 브랜치의 Commit3 커밋을 branch1 브랜치에 최신 커밋으로 반영함
  - git checkout branch1
  - git cherry-pick [커밋체크섬]
    - 아래 그림에 대해서는 git cherry-pick 4ff5d23

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1)
$ git log --all --oneline --graph
* 0a93ffe (HEAD -> branch1, origin/branch1) Commit6
* 2723b48 Commit5
| * da50c85 (origin/master, master) Commit4
| * 4ff5d23 Commit3
|/
* f7d2f74 Commit2
* a597443 Commit1
```

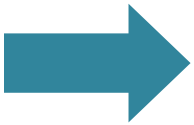
  - git push origin branch1

cherry-pick 실습

- 결과

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1|CHERRY-PICKING)
$ git cherry-pick 4ff5d23
[branch1 10cfaab] Commit3
 Date: Wed Apr 26 22:40:04 2023 +0900
 1 file changed, 1 insertion(+), 1 deletion(-)

USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1)
$ git log --all --oneline --graph
* 10cfaab (HEAD -> branch1) Commit3
* 0a93ffe (origin/branch1) Commit6
* 2723b48 Commit5
| * da50c85 (origin/master, master) Commit4
| * 4ff5d23 Commit3
|/
* f7d2f74 Commit2
* a597443 Commit1
```



```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (branch1)
$ git log --all --oneline --graph
* 10cfaab (HEAD -> branch1, origin/branch1) Commit3
* 0a93ffe Commit6
* 2723b48 Commit5
| * da50c85 (origin/master, master) Commit4
| * 4ff5d23 Commit3
|/
* f7d2f74 Commit2
* a597443 Commit1
```

reset (Hard모드)

- 브랜치A의 최근 커밋을 커밋B으로 Hard모드로 돌림
  - git checkout A
  - git reset --hard [커밋B체크섬]

reset (Hard모드) 실습

- 상황
  - 아래 상황에서 시작함

파일내용	GitFile1.txt
Commit3	Contents3
Commit2	Contents2
Commit1	Contents1

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* 4ff5d23 (HEAD -> master, origin/master) Commit3
* f7d2f74 Commit2
* a597443 Commit1
```

- 목표
  - Commit1으로 되돌리는 것

reset (Hard모드) 실습

- 절차
  - git reset --hard [커밋체크섬]
    - ex) 아래 상황에 대해서는 git reset --hard a597443

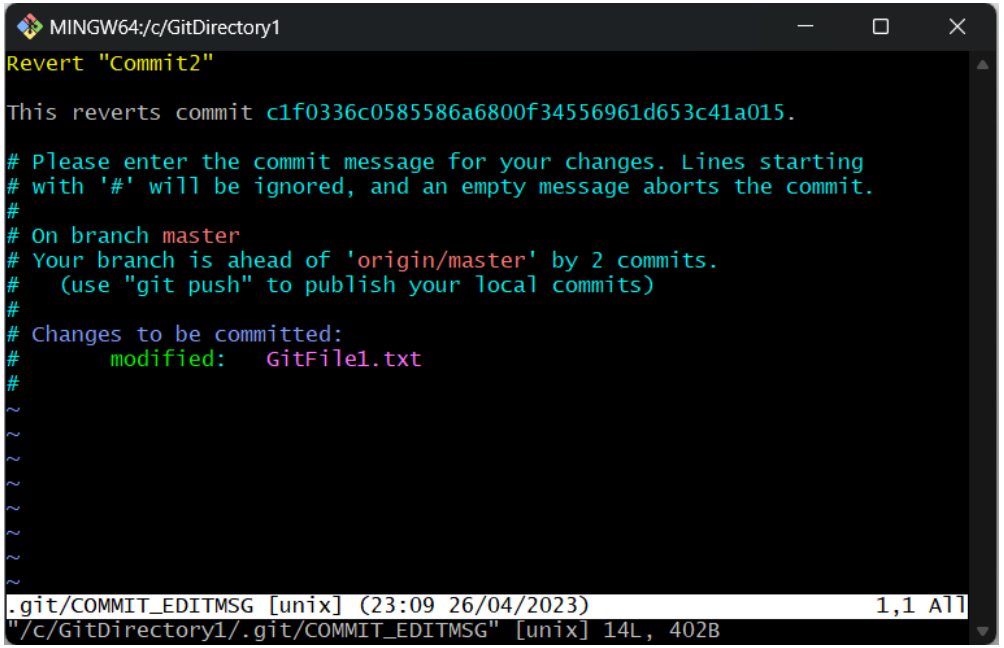
```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* 4ff5d23 (HEAD -> master, origin/master) Commit3
* f7d2f74 Commit2
* a597443 Commit1
```

  - git push origin master --force
- 결과

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* a597443 (HEAD -> master, origin/master) Commit1
```

revert

- 브랜치 A에서 브랜치 A내의 커밋B(체크섬)를 revert함
  - git checkout A
  - git revert b
    - 오른쪽 그림과 같은 창이 뜨면 그대로 저장해줌



- 브랜치 A에서 최신커밋을 revert함
  - git checkout A
  - git revert HEAD

revert 실습

- 상황
  - 아래 상황에서 시작함

파일내용	GitFile1.txt
Commit4	Contents4
Commit3	Contents3
Commit2	Contents2
Commit1	Contents1

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* 240fb39 (HEAD -> master, origin/master) Commit4
* 13637ac Commit3
* c1f0336 Commit2
* a597443 Commit1
```

revert 실습

- 목표
  - Commit4, Commit3, Commit2를 차례로 revert할 예정
- 절차
  - git revert c5b0d0a
  - git revert 13637ac
  - git revert c1f0336
  - git push origin master --force

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* c5b0d0a (HEAD -> master, origin/master) Commit4
* 13637ac Commit3
* c1f0336 Commit2
* a597443 Commit1
```

revert 실습

- 결과

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* e30e39f (HEAD -> master) Revert "Commit2"
* f217442 Revert "Commit3"
* 57b3d51 Revert "Commit4"
* c5b0d0a (origin/master) Commit4
* 13637ac Commit3
* c1f0336 Commit2
* a597443 Commit1
```

stash

- 현재 작업파일들을 A이름으로 stash 저장
  - git add .
    - 작업파일 중 untracked이 있는 경우 스테이지에 올려야함
  - git stash save A

stash 실습

- 상황
  - 아래 상황에서 시작함

파일내용	GitFile1.txt
Commit3	Contents3
Commit2	Contents2
Commit1	Contents1

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --all --oneline --graph
* 13637ac (HEAD -> master, origin/master) Commit3
* c1f0336 Commit2
* a597443 Commit1
```

  - VSCode에서 GitFile1.txt의 내용을 Contents4로 수정하고 GitFile2.txt를 새로 생성하여 Contents1으로 씀
  - 이 작업사항을 stash에 저장하고자 함

stash 실습

- 상황

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ 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 restore <file>..." to discard changes in working directory)
        modified:   GitFile1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        GitFile2.txt

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

stash 실습

- 절차
  - git add .
  - git stash save stash1

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git stash save stash1
Saved working directory and index state On master: stash1

USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

  - git stash list
    - stash 목록 보는 법

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git stash list
stash@{0}: On master: stash1
```

stash 실습

- 절차
  - git stash apply stash@{0}

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git stash apply stash@{0}
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   GitFile2.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   GitFile1.txt
```

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   GitFile2.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   GitFile1.txt
```