

LG전자 BS팀

『3일차』 : 오전

◆ 훈련과정명 : [BS] Git 형상관리에 대한 이해 및 실습

◆ 훈련기간 : 2023.06.07 ~ 2023.06.09

Copyright 2022. Daekyeong all rights reserved

- 1** 1교시 : Git 내부 동작 원리
- 2** 2교시 : Git 내부 동작 원리
- 3** 3교시 : Git 실무 사례 관련 이슈
- 4** 4교시 : Git 실무 사례 관련 이슈

『3과목』 Git, GitHub 추가 내용

1-2교시 :

Git 내부 동작 원리



학습목표

- 이 워크샵에서는 **git add, git commit, branch** 동작원리에 대해 알 수 있습니다.

눈높이 체크

- **git add, git commit, branch** 를 알고 계신가요?



1. git add 명령의 동작 원리

1. 새 로컬 저장소 생성

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces
```

```
$ pwd
```

```
/c/dev/gitworkspaces
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces
```

```
$ mkdir git_study_4th_project
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces
```

```
$ cd git_study_4th_project/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project
```

```
$ git init
```

```
Initialized empty Git repository in C:/DEV/gitworkspaces/git_study_4th_project/.git/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls -al
```

```
total 8
```

```
drwxr-xr-x 1 apro621 197121 0 5월 11 10:10 ./
```

```
drwxr-xr-x 1 apro621 197121 0 5월 11 10:10 ../
```

```
drwxr-xr-x 1 apro621 197121 0 5월 11 10:10 .git/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls -al .git/
```

```
total 11
```

```
drwxr-xr-x 1 apro621 197121 0 5월 11 10:10 ./
```

```
drwxr-xr-x 1 apro621 197121 0 5월 11 10:10 ../
```

```
-rw-r--r-- 1 apro621 197121 130 5월 11 10:10 config
```

```
-rw-r--r-- 1 apro621 197121 73 5월 11 10:10 description
```

```
-rw-r--r-- 1 apro621 197121 21 5월 11 10:10 HEAD
```

```
...
```



1. git add 명령의 동작 원리

2. git add와 git status

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ echo "git-add" > gitadd.txt
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git status
On branch main
```

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)
____gitadd.txt

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

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$
```

working	stage	local	remote
gitadd.txt			



1. git add 명령의 동작 원리

2. git add와 git status

● 파일의 체크섬 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git hash-object gitadd.txt
b337e7f9597a37f6bd96d99cd385066554696d6c
```

b337e

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$
```

● stage에 파일 추가

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git add gitadd.txt
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git status
On branch main
```

No commits yet

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

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```



1. git add 명령의 동작 원리

2. git add와 git status

● git add gitadd.txt 명령 수행 후 상태

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls -a .git
```

```
./ ../ config description HEAD hooks/ index info/ objects/ refs/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ file .git/index
```

```
.git/index: Git index, version 2, 1 entries
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git ls-files --stage
```

```
100644 b337e7f9597a37f6bd96d99cd385066554696d6c 0    gitadd.txt
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls -a .git/objects
```

```
./ ../ b3/ info/ pack/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls -a .git/objects/b3
```

```
./ ../ 37e7f9597a37f6bd96d99cd385066554696d6c
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git show b337e
```

```
git-add
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$
```

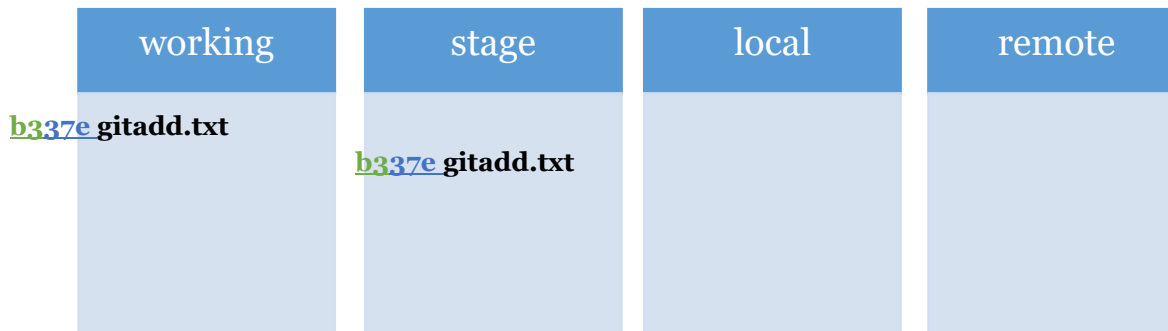
b337e



1. git add 명령의 동작 원리

2. git add와 git status

- git add 명령은 워킹트리에 존재하는 파일을 stage에 추가하는 명령이며, `./git/objects` 파일에 blob 객체가 생성되고 stage 내용은 `./git/index`에 기록됨을 알 수 있다.



```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git cat-file -t b337e
blob
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$
```



2. git commit 명령의 동작 원리

1. 커밋 수행

- git commit 명령후, clean 함은 워킹트리와 stage, HEAD 내용이 모두 같음

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git commit -m "커밋 확인용 커밋"
[main (root-commit) 52c0bea] 커밋 확인용 커밋
1 file changed, 1 insertion(+)
create mode 100644 gitadd.txt
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git log
commit 52c0bea309cbee6e3c8552dfdod5c3f1455d5203 (HEAD -> main)
Author: looker2zip <looker2zip@gmail.com>
Date: Thu May 11 11:44:19 2023 +0900
```

커밋 확인용 커밋

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git status
On branch main
nothing to commit, working tree clean
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$
```



2. git commit 명령의 동작 원리

1. 커밋 수행

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls -a .git/objects
```

```
./ ../ 1b/ 52/ b3/ info/ pack/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls -a .git/objects/52
```

```
./ ../ c0bea309cbee6e3c8552dfdod5c3f1455d5203
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git show 52c0b
```

```
commit 52c0bea309cbee6e3c8552dfdod5c3f1455d5203 (HEAD -> main)
```

```
Author: looker2zip <looker2zip@gmail.com>
```

```
Date: Thu May 11 11:44:19 2023 +0900
```

커밋 확인용 커밋

```
diff --git a/gitadd.txt b/gitadd.txt
```

```
new file mode 100644
```

```
index 0000000..b337e7f
```

```
--- /dev/null
```

```
+++ b/gitadd.txt
```

```
@@ -0,0 +1 @@
```

```
+git-add
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

2. git commit 명령의 동작 원리

1. 커밋 수행

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git ls-files --stage
```

```
100644 b337e7f9597a37f6bd96d99cd385066554696d6c o    gitadd.txt
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

앞 선 것처럼 stage에 100644 b337e7f9597a37f6bd96d99cd385066554696d6c o
gitadd.txt가 존재

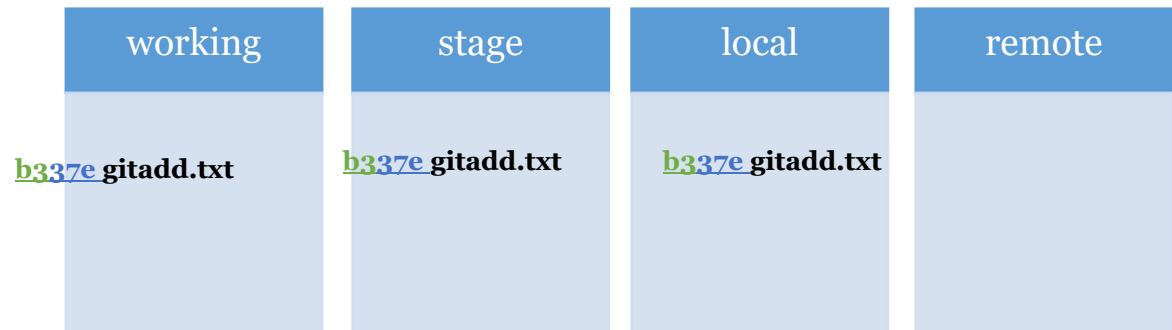
```
$ git status
```

```
On branch main
```

```
nothing to commit, working tree clean
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$
```



<HEAD>52c0bea



2. git commit 명령의 동작 원리

2. tree 객체

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ ls -a .git/objects

./ ../ 1b/ 52/ b3/ info/ pack/

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ ls -a .git/objects/1b

./ ../ 4eec31ec6fbb71d5c5d1495bab5fdeefe4eca0

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git show 1b4eec

tree 1b4eec

gitadd.txt

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

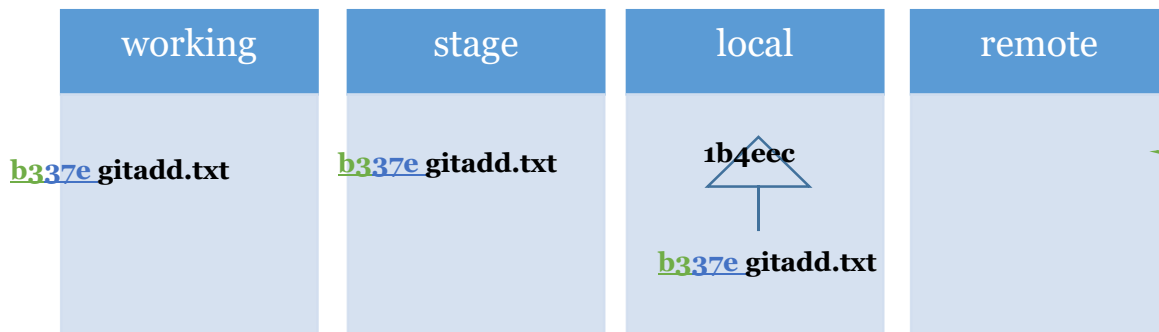
\$ git ls-tree 1b4eec

100644 blob b337e7f9597a37f6bd96d99cd385066554696d6c gitadd.txt

Object 폴더 내용 확인

tree 객체

tree 객체 내용



커밋 상태 최종 버전

<HEAD>52cobe

2. git commit 명령의 동작 원리

3. 커밋 객체 확인

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git ls-files --stage

100644 b337e7f9597a37f6bd96d99cd385066554696d6c 0 gitadd.txt

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git log --oneline -n1

52cobe a (HEAD -> main) 커밋 확인용 커밋

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git cat-file -t 52cobe a

commit

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git cat-file commit 52cobe a

tree 1b4eec31ec6fbb71d5c5d1495bab5fdeefe4ecao

author looker2zip <looker2zip@gmail.com> 1683773059 +0900

committer looker2zip <looker2zip@gmail.com> 1683773059 +0900

커밋 확인용 커밋

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$

stage 내용

커밋 체크섬 내용

커밋 객체 타입 확인

커밋 객체 내용 확인



2. git commit 명령의 동작 원리



4. 정리

- 커밋을 하면 스테이지의 객체로 트리가 만들어짐
- 커밋에는 커밋 메시지와 트리 객체가 포함

3. 커밋 좀 더 살펴보기

1. 파일 수정하고 추가 커밋하기

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls
```

```
gitadd.txt
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ cat gitadd.txt
```

```
git-add
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git hash-object gitadd.txt
```

```
b337e7f9597a37f6bd96d99cd385066554696d6c
```

체크섬 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ echo "Hello, Git-add" >> gitadd.txt
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ cat gitadd.txt
```

```
git-add
```

```
Hello, Git-add
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git hash-object gitadd.txt
```

```
049038981dde374c888d3477de42c504d5ba821
```

변경된 체크섬 확인

3. 커밋 좀 더 살펴보기

1. 파일 수정하고 추가 커밋하기

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git ls-files --stage

100644 b337e7f9597a37f6bd96d99cd385066554696d6c 0 gitadd.txt

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git ls-tree HEAD

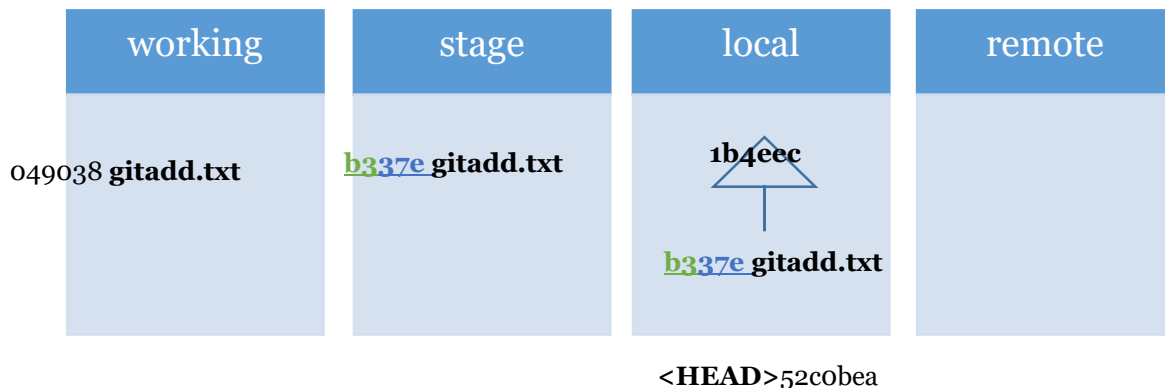
100644 blob b337e7f9597a37f6bd96d99cd385066554696d6c gitadd.txt

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$

스테이지 파일 확인

헤드 커밋 내용 확인



워킹트리 체크섬만 바뀜

3. 커밋 좀 더 살펴보기

2. 변경 내용 스테이지에 추가

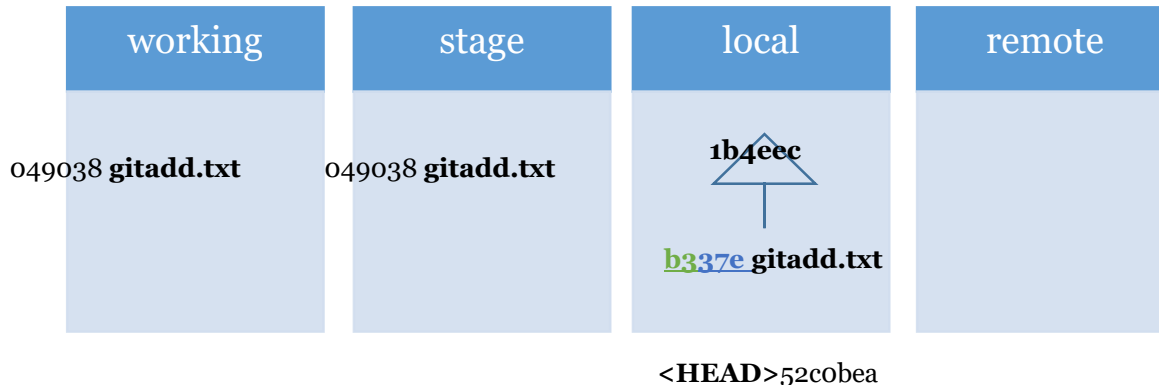
```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git add gitadd.txt
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$ git ls-files --stage
100644 049038981dde374c888d3477de42c504d5ba821 o    gitadd.txt
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
$
```

git add

헤드 커밋 내용 확인



git add 명령으로 스테이지 체크섬 변경

3. 커밋 좀 더 살펴보기

3. 트리로 커밋하기

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git write-tree
```

```
37bb523f87ddc518e63ccobddd574314d3d704ef
```

트리 생성

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git ls-tree 37bb5
```

```
100644 blob 049038981dde374c888d3477de42c504d5ba821 gitadd.txt
```

생성된 트리 객체 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ echo "트리로 커밋하기" | git commit-tree 37bb5 -p HEAD
```

```
37a081edb90a5b422ee42878c4df4775e10a13cc
```

트리로 커밋

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git cat-file commit 37a081
```

```
tree 37bb523f87ddc518e63ccobddd574314d3d704ef
```

```
parent 52c0bea309cbee6e3c8552dfd0d5c3f1455d5203
```

```
author looker2zip <looker2zip@gmail.com> 1683775938 +0900
```

```
committer looker2zip <looker2zip@gmail.com> 1683775938 +0900
```

생성된 커밋 확인

트리로 커밋하기

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git log --oneline
```

```
52c0bea (HEAD -> main) 커밋 확인용 커밋
```

커밋 로그 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$
```

3. 커밋 좀 더 살펴보기

4. HEAD 갱신하기

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls .git
```

```
COMMIT_EDITMSG config description HEAD hooks/ index info/ logs/ objects/ refs/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ cat .git/HEAD
```

```
ref: refs/heads/main
```

HEAD 파일 내용 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ cat .git/refs/heads/main
```

```
52c0bea309cbee6e3c8552dfd0d5c3f1455d5203
```

내용 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git update-ref refs/heads/main 37a081
```

직접 커밋한 객체로 업데이트

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ cat .git/refs/heads/main
```

```
37a081edb90a5b422ee42878c4df4775e10a13cc
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git log --oneline
```

```
37a081e (HEAD -> main) 트리로 커밋하기
```

```
52c0bea 커밋 확인용 커밋
```

업데이트와 로그 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$
```

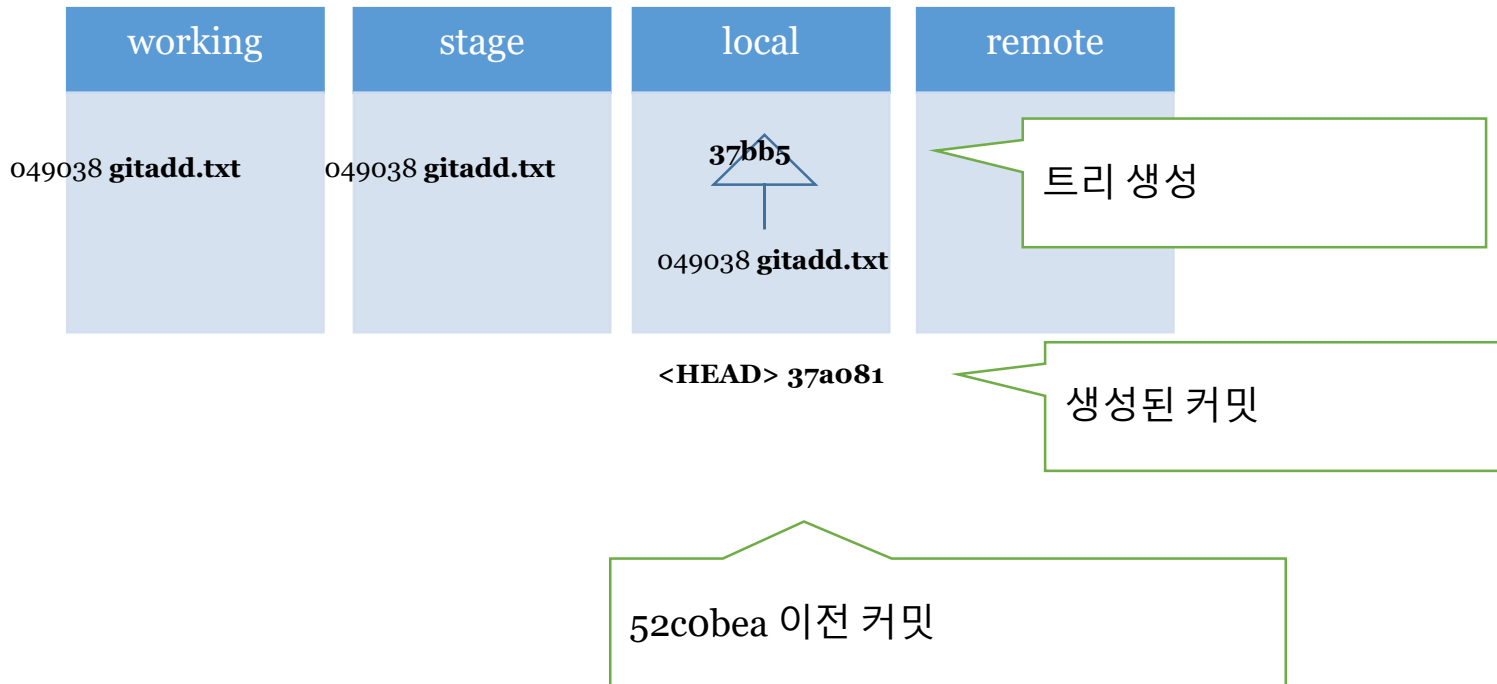
3. 커밋 좀 더 살펴보기

4. HEAD 갱신하기

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ ls .git

COMMIT_EDITMSG config description HEAD hooks/ index info/ logs/ objects/ refs/



4. 브랜치 작업

1. 브랜치 생성

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch
```

```
* main
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch branchtest
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch
```

```
branchtest
```

```
* main
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git log --oneline
```

```
37a081e (HEAD -> main, branchtest) 트리로 커밋하기
```

```
52c0bea 커밋 확인용 커밋
```

생성된 브랜치 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls .git/refs/heads/
```

```
branchtest main
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ cat .git/refs/heads/branchtest
```

```
37a081edb90a5b422ee42878c4df4775e10a13cc
```

파일 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$
```

4. 브랜치 작업

2. 브랜치 삭제 및 재생성

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch
```

```
branchtest
```

```
* main
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch -d branchtest
```

```
Deleted branch branchtest (was 37a081e).
```

브랜치 삭제

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch
```

```
* main
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls .git/refs/heads/
```

```
main
```

4. 브랜치 작업

2. 브랜치 삭제 및 재생성

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch branch-test
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch
```

```
branch-test
```

```
* main
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ ls .git/refs/heads
```

```
branch-test main
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git log --oneline -n1
```

```
37a081e (HEAD -> main, branch-test) 트리로 커밋하기
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ rm .git/refs/heads/branch-test
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git log --oneline -n1
```

```
37a081e (HEAD -> main) 트리로 커밋하기
```

로그 확인

4. 브랜치 작업

3. 브랜치 체크아웃

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch branch-test1 HEAD^
```

HEAD 부모 커밋으로부터
브랜치 생성

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git branch
```

```
branch-test1
```

```
* main
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git log --oneline -n2
```

```
37a081e (HEAD -> main) 트리로 커밋하기
```

```
52c0bea (branch-test1) 커밋 확인용 커밋
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ cat .git/HEAD
```

```
ref: refs/heads/main
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ git checkout branch-test1
```

```
Switched to branch 'branch-test1'
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (branch-test1)
```

```
$ cat .git/HEAD
```

```
ref: refs/heads/branch-test1
```

HEAD 파일 변경사항 확인

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (branch-test1)
```

```
$ git status
```

```
On branch branch-test1
```

```
nothing to commit, working tree clean
```

워킹트리 확인

4. 브랜치 작업

4. 수동 체크아웃

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (branch-test1)

\$ echo "ref: refs/heads/main" > .git/HEAD

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git log --oneline -n1

37a081e (HEAD -> main) 트리로 커밋하기

HEAD 파일 직접 수정

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git status

On branch main

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: gitadd.txt

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git reset --hard

HEAD is now at 37a081e 트리로 커밋하기

hard reset 수행

apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)

\$ git status

On branch main

nothing to commit, working tree clean



5. SSH 사용 해 저장소 클론

1. SSH란?

- SSH 프로토콜은 1995년에 개발되었는데 Unix나 Linux 같은 OS에 안전하게 접속하기 위해 만들어졌다.



5. SSH 사용 해 저장소 클론

2. SSH 키 생성

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_4th_project (main)
```

```
$ cd ~
```

```
apro621@DESKTOP-CL1JPPC MINGW64 ~
```

```
$ pwd
```

```
/c/Users/apro621
```

```
apro621@DESKTOP-CL1JPPC MINGW64 ~
```

```
$ ssh-keygen
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/c/Users/apro621/.ssh/id_rsa):
```

```
/c/Users/apro621/.ssh/id_rsa already exists.
```

```
...
```

```
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
```

```
Your identification has been saved in /c/Users/apro621/.ssh/id_rsa
```

```
Your public key has been saved in /c/Users/apro621/.ssh/id_rsa.pub
```

```
The key fingerprint is:
```

```
SHA256:+oI/xc2nsq979XfgMnwSY6kpp1snnr/cRCgbB5B6GV0 apro621@DESKTOP-CL1JPPC
```

```
The key's randomart image is:
```

```
+---[RSA 3072]-----+
```

```
| .. |  
| E. |  
| + o. |  
| o o .. |  
| oSoo o.. |  
| .o o=B.. |  
| ... +O.=.. |  
| . o.oo*==o+ o|  
| ..oB%+.+*...|
```

```
+----[SHA256]-----+
```

```
apro621@DESKTOP-CL1JPPC MINGW64 ~
```

```
$
```

엔터

엔터



5. SSH 사용 해 저장소 클론

2. SSH 키 생성

```
apro621@DESKTOP-CL1JPPC MINGW64 ~  
$ cd ~/.ssh/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 ~/.ssh  
$ pwd  
/c/Users/apro621/.ssh
```

```
apro621@DESKTOP-CL1JPPC MINGW64 ~/.ssh  
$ ls  
id_rsa id_rsa.pub
```

```
apro621@DESKTOP-CL1JPPC MINGW64 ~/.ssh  
$ cat id_rsa.pub  
ssh-rsa ...
```

```
R/4dZwPrJFpcp3ccFTI7QibTbktAVWL2N58J3r/dsIqeBrwCAZJEG2qIGWwPg/nzsZWUKfsTyDZtSEMibJokfqsBR8ajTSs/BOJKLd5evTtVns=  
apro621@DESKTOP-CL1JPPC
```

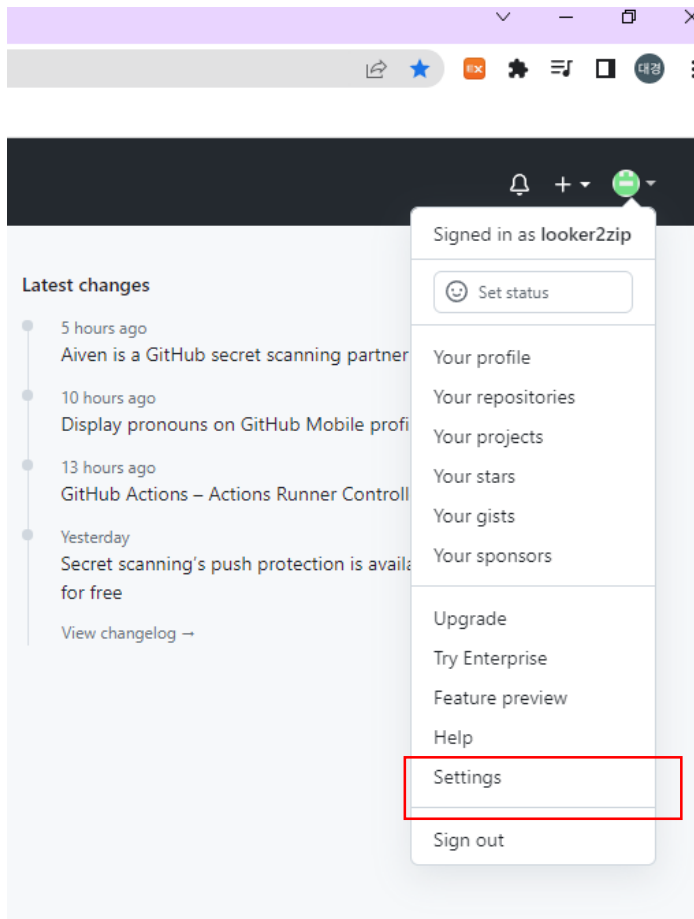
```
apro621@DESKTOP-CL1JPPC MINGW64 ~/.ssh  
$
```

공개키 확인

5. SSH 사용 해 저장소 클론

3. GitHub에 키 등록하기

- GitHub 우측 상단 > Settings 선택



5. SSH 사용 해 저장소 클론

3. GitHub에 키 등록하기

- SSH and GPG keys > SSH keys [New SSH key] 클릭

The screenshot shows the GitHub 'SSH and GPG keys' settings page for the user 'looker2zip'. The left sidebar contains navigation links: Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys (highlighted), Organizations, and Moderation. The main content area is divided into three sections: SSH keys, GPG keys, and Vigilant mode. In the SSH keys section, there is a message stating 'There are no SSH keys associated with your account.' and a link to 'generating SSH keys'. A red dashed circle highlights the 'New SSH key' button in the top right corner of this section. In the GPG keys section, there is a message stating 'There are no GPG keys associated with your account.' and a link to 'generate a GPG key'. The Vigilant mode section has a checkbox for 'Flag unsigned commits as unverified' which is currently unchecked, with a note explaining that this will include any commit attributed to the account but not signed with a GPG or S/MIME key. A link to 'Learn about vigilant mode.' is provided at the bottom of this section. A 'Go to your personal profile' button is located in the top right corner of the main content area.

looker2zip (looker2zip)
Your personal account

Public profile
Account
Appearance
Accessibility
Notifications

Access

Billing and plans
Emails
Password and authentication
Sessions

SSH and GPG keys

Organizations
Moderation

Code, planning, and automation

SSH keys

There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

New SSH key

GPG keys

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and add it to your account.

New GPG key

Vigilant mode

☐ Flag unsigned commits as unverified

This will include any commit attributed to your account but not signed with your GPG or S/MIME key.
Note that this will include your existing unsigned commits.


[Learn about vigilant mode.](#)

Go to your personal profile

5. SSH 사용 해 저장소 클론

3. GitHub에 키 등록하기

- SSH and GPG keys > SSH keys [New SSH key] 클릭

**looker2zip (looker2zip)**
Your personal account

[Go to your personal profile](#)

[Public profile](#)
[Account](#)
[Appearance](#)
[Accessibility](#)
[Notifications](#)

[Access](#)
[Billing and plans](#)
[Emails](#)
[Password and authentication](#)
[Sessions](#)
[SSH and GPG keys](#)
[Organizations](#)
[Moderation](#)

[Code, planning, and automation](#)
[Repositories](#)

SSH keys / Add new

Title
내 컴퓨터 공개키

Key type
Authentication Key


Key
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGCxx/AKO95qMBzpgKXfnuHeLwBfO0OF3OxpOTySUBKchBXxv9ieqoX4v8jHSJF7AWxldqREEGS7bx1SNQ2qAC523wW5BM3SXdfh8B31y5G4KRuT4Km6l2a+DvNv6+qQbJemSSgHToAI1uwVev5S+IXJGCqBdsm56D8zfWjr48FQnwdBc5uZiFLtSFPPCLFYF4uQ36WCAvTx3joikwlsLzENL6GFj4Af/vmcpeJkFX8i4VNwEaQxjltPi+s0cEgwr7kFfUzrT0Gc+d/sgkoRHGtV2//u2+GHwzc0uAKqft3I0hb9CNCC/LC5M2cQZmKZptLcd4BKvGCZnJJaoPo0VloE7j+yhOWaHualue1V6bCyHBxM8Q/d1nzNggOK+CAKvkFdd3Tav4VBxIACKJAaR/4dZwPrJFpcp3ccFTI7QibTbktAVWL2N58J3r/dslqeBrwCAZJEG2qlGWwPg/nzsZWUKfsTyDZtSEMIbJokfqsBR8ajTSs/BOJKLd5evTtVns= apr0621@DESKTOP-CL1JPPC

Add SSH key

5. SSH 사용 해 저장소 클론

3. GitHub에 키 등록하기

- SSH and GPG keys > SSH keys [New SSH key] 클릭

**looker2zip (looker2zip)**
Your personal account

[Go to your personal profile](#)

[Public profile](#)
[Account](#)
[Appearance](#)
[Accessibility](#)
[Notifications](#)

[Access](#)
[Billing and plans](#)
[Emails](#)
[Password and authentication](#)
[Sessions](#)
[SSH and GPG keys](#)
[Organizations](#)
[Moderation](#)


[Code, planning, and automation](#)

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

**내 컴퓨터 공개키**
SHA256: +oI/xc2nsq979XfgMnwSY6kpp1snnr/cRCgb8586GVo
Added on May 11, 2023
Never used — Read/write
[Delete](#)

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

GPG keys

[New GPG key](#)

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and add it to your account.

5. SSH 사용 해 저장소 클론

4. SSH를 이용해서 저장소 클론

`git@github.com:looker2zip/git_study_prex1.git`

The screenshot shows the GitHub interface for a repository named 'looker2zip / git_study_prex1'. The repository is public and forked from 'onlookertozip/git_study_prex1'. The main branch is 'main', which is 1 commit ahead of the upstream. The repository contains a README.md file. A 'Code' dropdown menu is open, showing options to clone the repository using Local, Codespaces, or various protocols (HTTPS, SSH, GitHub CLI). The SSH URL is displayed as 'git@github.com:looker2zip/git_study_prex1.git'. Other options include 'Open with GitHub Desktop', 'Open with Visual Studio', and 'Download ZIP'. The right sidebar shows repository statistics: 0 stars, 0 watching, and 1 fork. The footer contains the GitHub logo, copyright information (© 2023 GitHub, Inc.), and links to Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.



```
apro621@DESKTOP-CL1JPPC MINGW64 ~/.ssh
$ cd ~
```

```
apr0621@DESKTOP-CL1JPPC MINGW64 ~  
$ pwd  
/c:/Users/apr0621
```

```
apro621@DESKTOP-CL1JPPC MINGW64 ~
$ echo "Host github.com" >> ~/.ssh/config
```

```
apr0621@DESKTOP-CL1JPPC MINGW64 ~  
$ cat ~/.ssh/config  
Host github.com
```

```
ap0621@DESKTOP-CL1JPPC MINGW64 ~
$ vi ~/.ssh/config
```

```
Host github.com
  Hostname github.com
  IdentityFile ~/.ssh/id_rsa
```

2칸 들어쓰기

[illegible]

5. SSH 사용 해 저장소 클론

4. SSH를 이용해서 저장소 클론

```
apro621@DESKTOP-CL1JPPC MINGW64 ~  
$ cd c:\dev
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev  
$ cd gitworkspaces/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces  
$ ls  
git_example1/      git_study_2nd_project/ git_study_4th_project/  
git_study_1st_project/ git_study_3rd_project/ git_study_prex1/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces  
$ rm -rf git_study_prex1/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces  
$ git clone git@github.com:looker2zip/git_study_prex1.git  
Cloning into 'git_study_prex1'...  
The authenticity of host 'github.com (20.200.245.247)' can't be established.  
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDAozPMSvHdkr4UvCOqU.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.  
remote: Enumerating objects: 6, done.  
remote: Counting objects: 100% (6/6), done.  
remote: Compressing objects: 100% (2/2), done.  
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0  
Receiving objects: 100% (6/6), done.
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces  
$ cd git_study_prex1/
```

```
apro621@DESKTOP-CL1JPPC MINGW64 /c/dev/gitworkspaces/git_study_prex1 (main)  
$ git push  
Everything up-to-date
```

『4과목』 Git, GitHub 활용

3-4교시 :

Git 실무 사례 관련 이슈



학습목표

- 이 워크샵에서는 현장에서 유용하게 사용하는 기능에 대해 알 수 있습니다.

눈높이 체크

- amend, cherry-pick, reset, revert, stash를 알고 계신가요?

1. 실습을 위한 사전 준비

1. GitHub에 새로운 원격저장소 만들기

- Create a new repository
- git_playground

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * / Repository name *

git_playground is available.

Great repository names are short and memorable. Need inspiration? How about [fantastic-octo-umbrella?](#)

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template:

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License:

License

Filter...

None

Apache License 2.0

GNU General Public License v3.0

MIT License

BSD 2-Clause "Simplified" License

Create repository

1. 실습을 위한 사전 준비

1. GitHub에 새로운 원격저장소 만들기

- Create a new repository
- git_playground

The screenshot shows the GitHub interface for a newly created public repository named 'git_playground' by the user 'looker2zip'. The repository is currently empty, with a single initial commit (b0c693e) containing a LICENSE file and a README.md file. The README.md file contains the text 'git_playground'. The right sidebar shows the repository's metadata, including 0 stars, 1 watching, and 0 forks. The bottom of the page displays the GitHub footer with copyright information and various links.

looker2zip / git_playground (Public)

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file <> Code

looker2zip Initial commit b0c693e now 1 commit

File	Commit	Time
LICENSE	Initial commit	now
README.md	Initial commit	now

README.md

git_playground

About

No description, website, or topics provided.

Readme MIT license 0 stars 1 watching 0 forks

Releases

No releases published
[Create a new release](#)

Packages

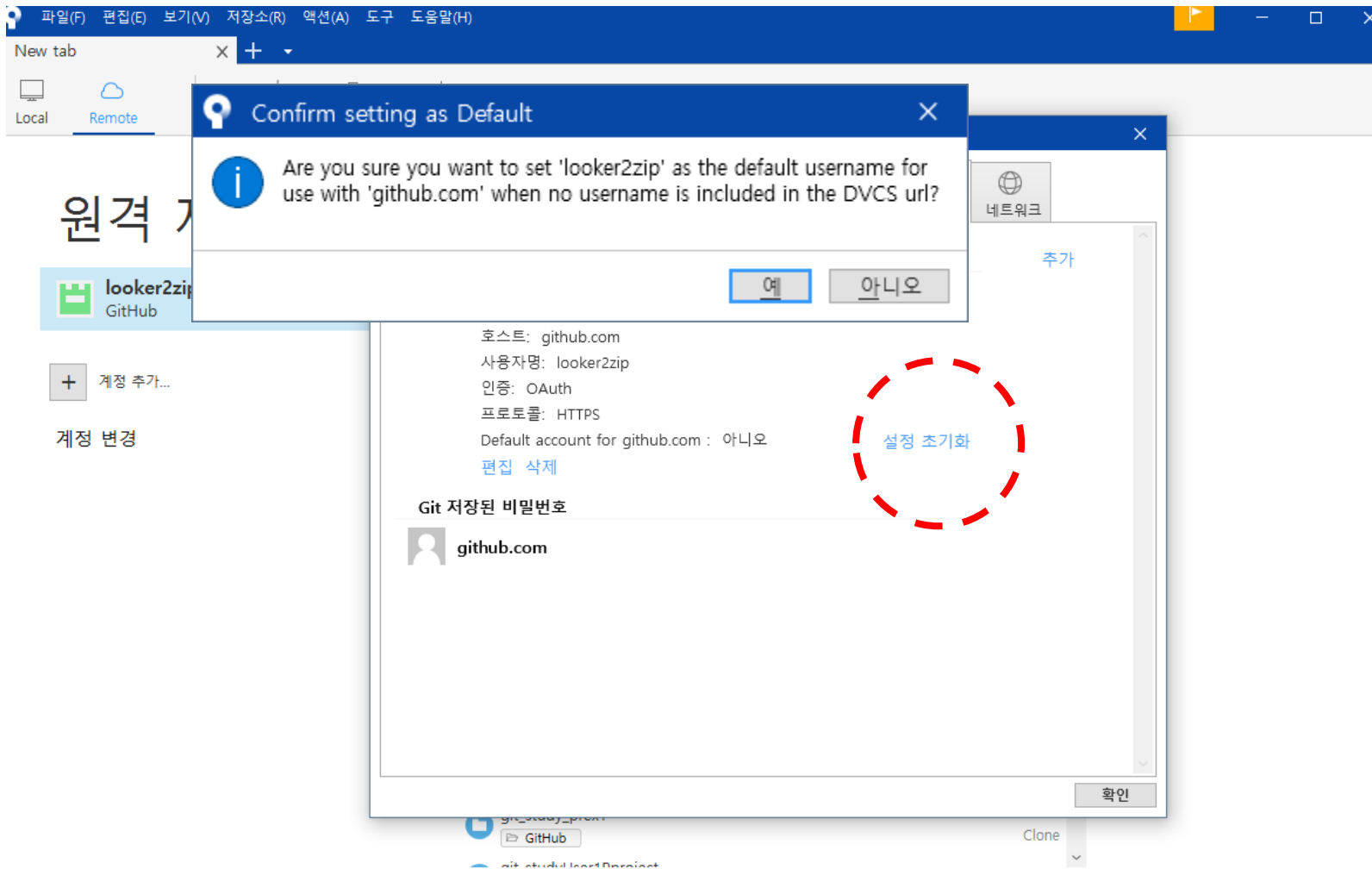
No packages published
[Publish your first package](#)

© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

1. 실습을 위한 사전 준비

2. 소스 트리 계정 설정 초기화

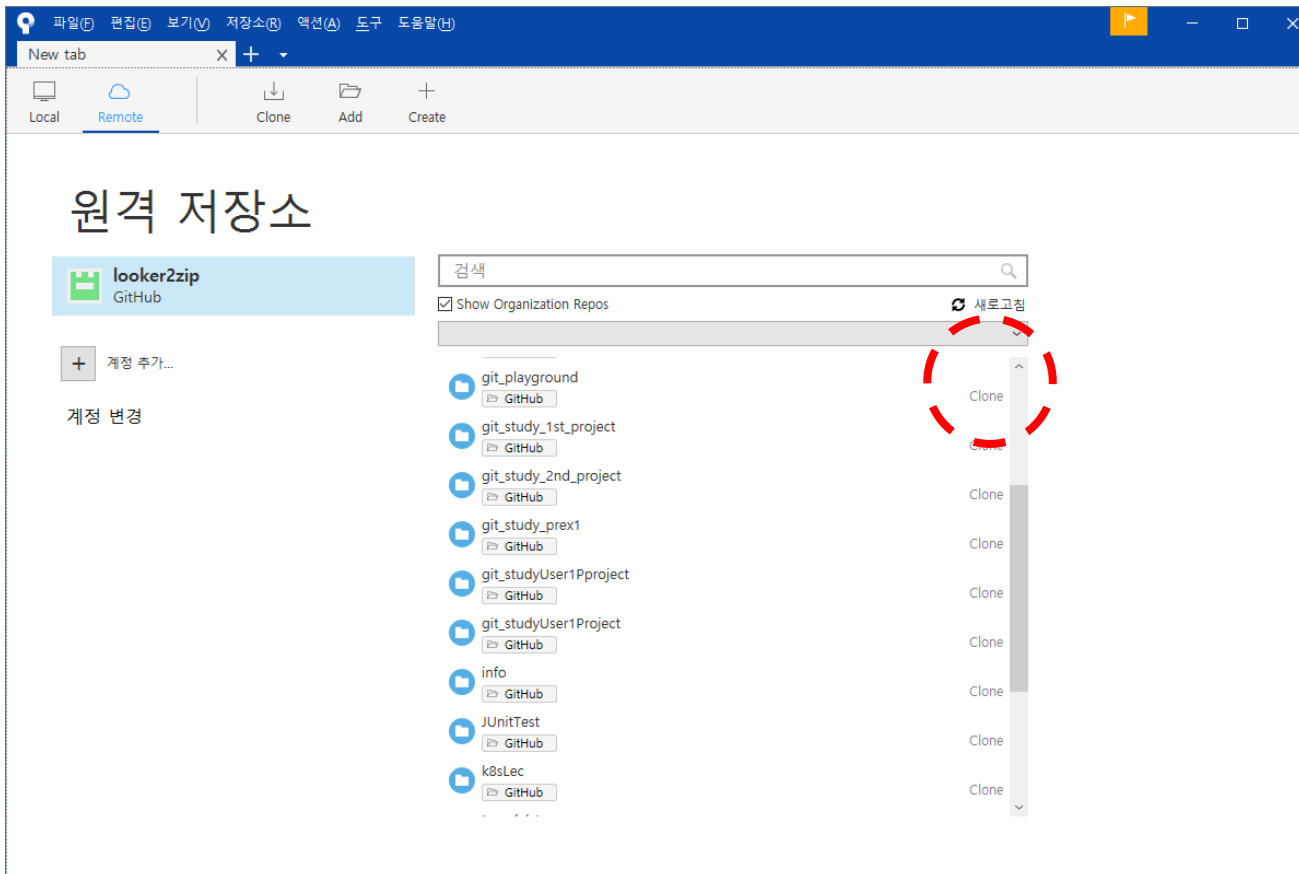
● 도구 > 옵션 > 인증



1. 실습을 위한 사전 준비

2. 소스 트리 계정 설정 초기화

- git_playground 원격 저장소 클론



1. 실습을 위한 사전 준비

2. 소스 트리 계정 설정 초기화

- **C:\DEV\gitworkspaces\git_playground**

Clone

Cloning is even easier if you set up a remote account

저장소 종류: Git 저장소입니다

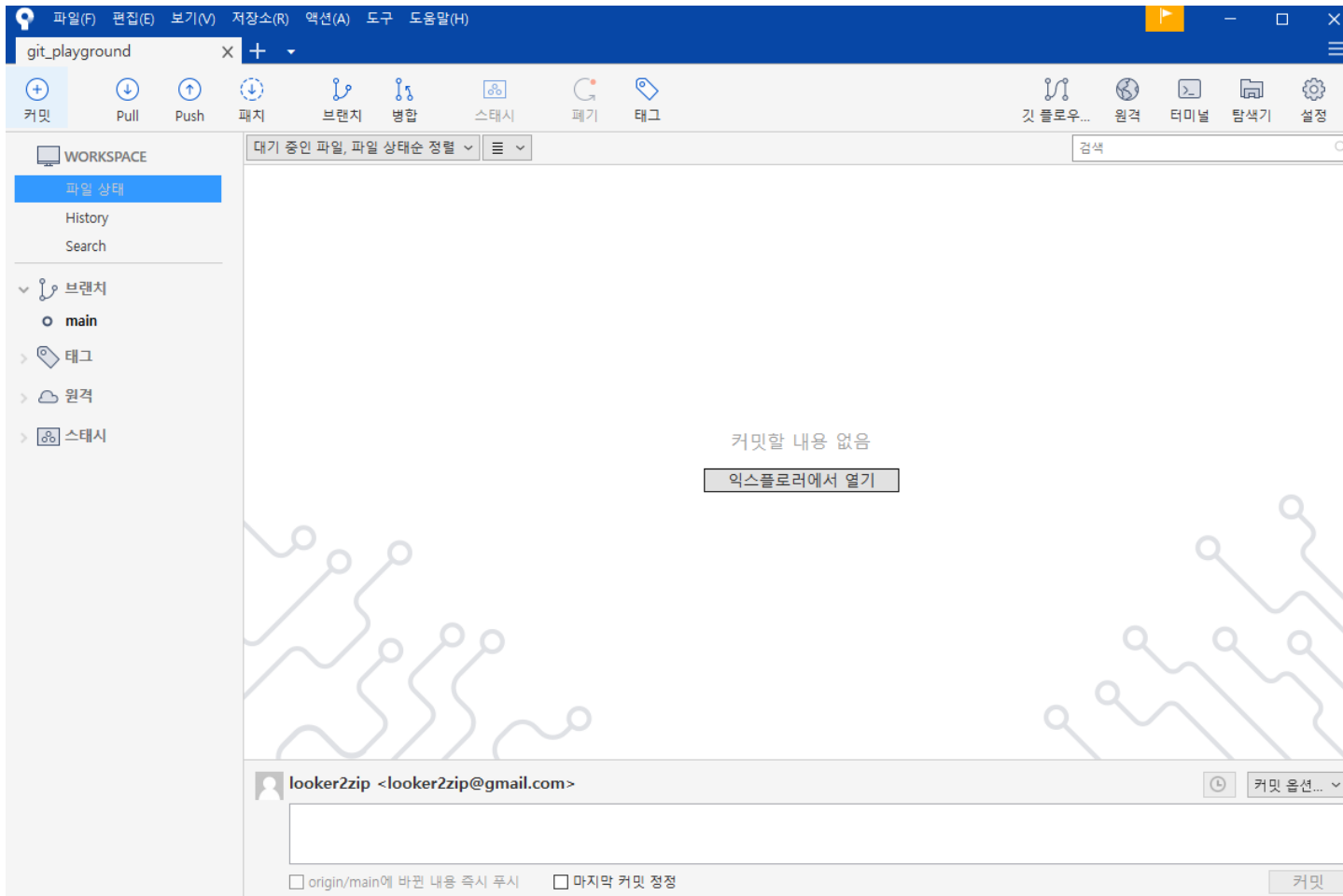
Local Folder:

> 고급 옵션

1. 실습을 위한 사전 준비

2. 소스 트리 계정 설정 초기화

- **C:\DEV\gitworkspaces\git_playground**

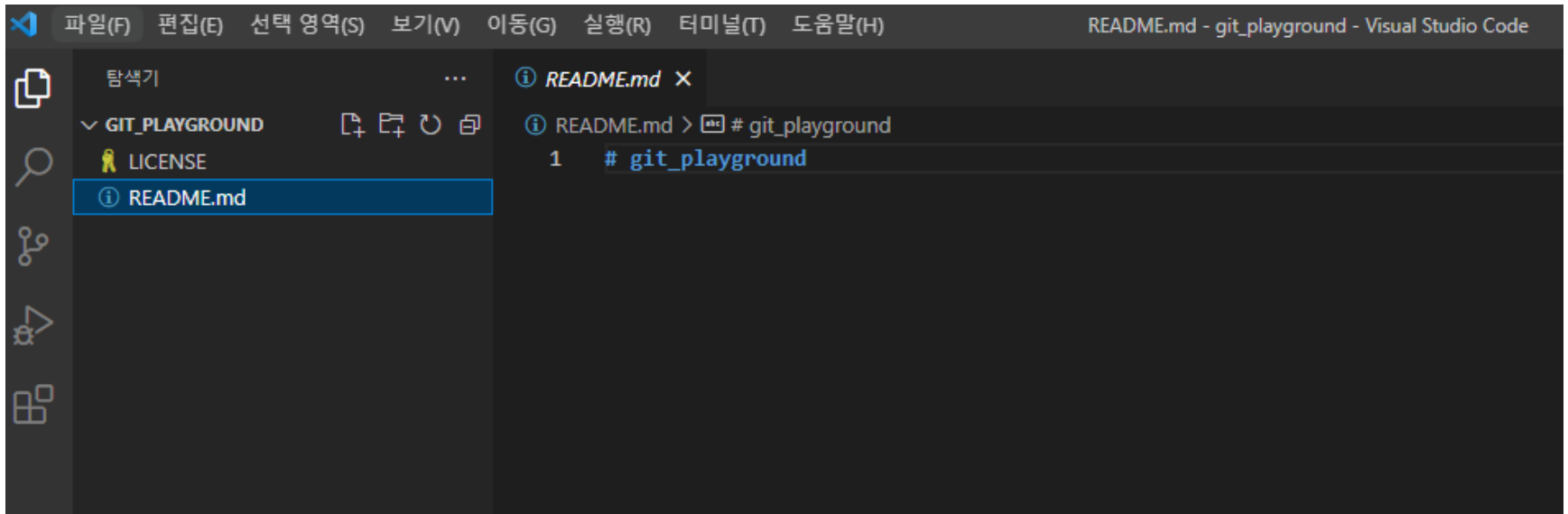




1. 실습을 위한 사전 준비

3. 비주얼 스튜디오 폴더 선택

- 파일 > 폴더 열기

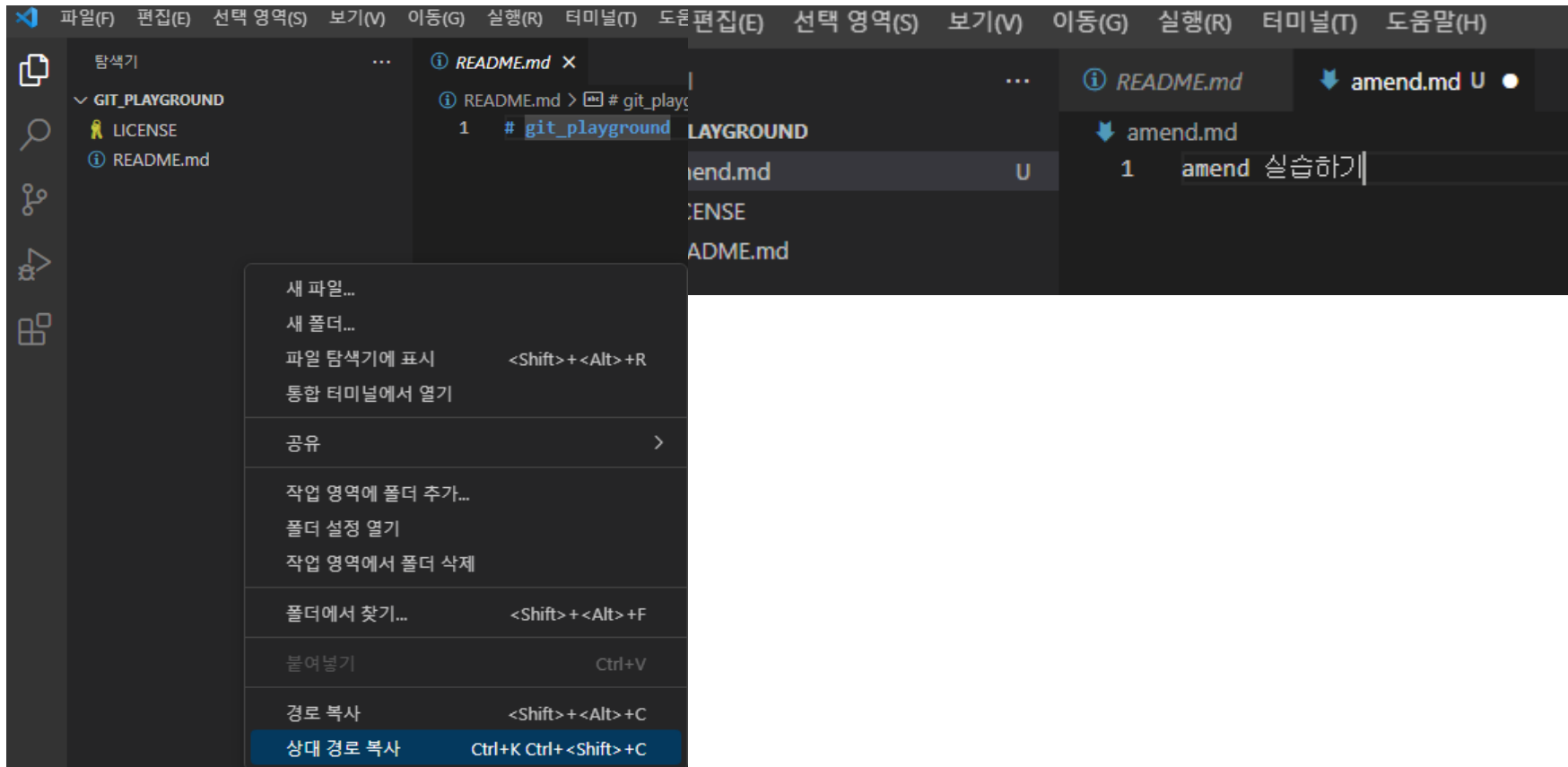




2. amend

1. amend로 마지막 커밋 수정

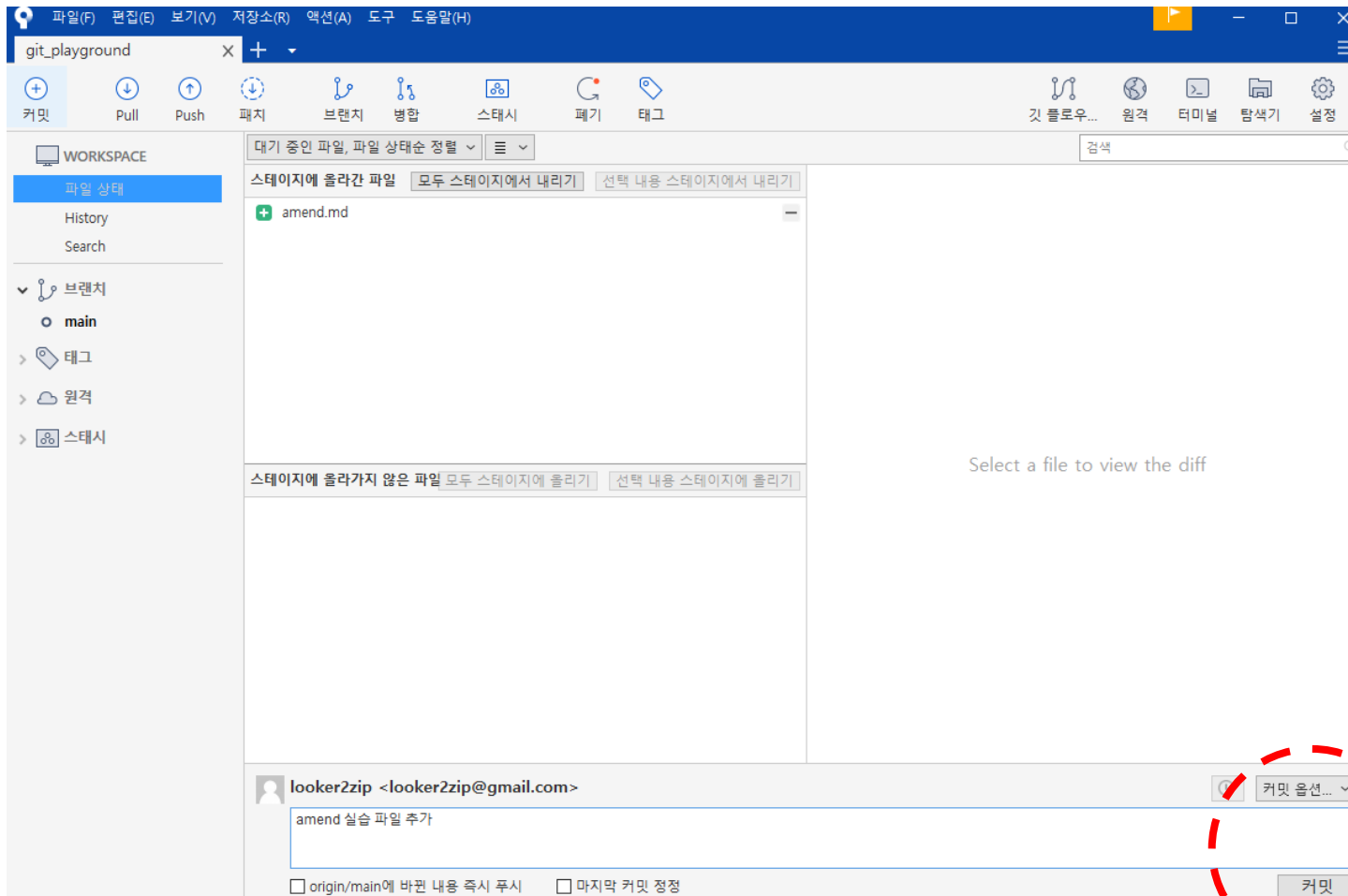
- 추가할 파일 존재 시, 커밋에 추가
- 새파일 추가 > amend.md



2. amend

1. amend로 마지막 커밋 수정

- 스테이지에 올리고 “amend 실습 파일 추가” 커밋



2. amend

1.amend로 마지막 커밋 수정

- 커밋 완료

The screenshot shows the git playground interface. The top bar includes a search icon and tabs for '파일(F)', '편집(E)', '보기(V)', '저장소(R)', '액션(A)', '도구', and '도움말(H)'. Below this is a toolbar with icons for '커밋', 'Pull', 'Push', '파지', '브랜치', '병합', '스테시', '폐기', and '태그'. The main workspace is divided into three sections: 'WORKSPACE' on the left, a commit history table in the center, and a file editor at the bottom.

WORKSPACE

- 파일 상태
- History
- Search
- 브랜치
 - main (1↑)
- 태그
- 원격
- 스테시

Commit History Table

그래프	설명	날짜	작성자	커밋
main 1	amend 실습 파일 추가	11 5 2023 21:55	looker2zip <looker2zip@gmail.com>	4b93c7d
origin/main		11 5 2023 17:54	looker2zip <80115650+looker2zip@users.noreply.github>	b0c693e

File Editor: amend.md

커밋: 4b93c7d620e9cc5e46154a0a1bcf9625618580a6 [4b93c7d]
상위 항목: b0c693e13a
작성자: looker2zip <looker2zip@gmail.com>
날짜: 2023년 5월 11일 목요일 오후 9:55:02
커밋한 사람: looker2zip

amend 실습 파일 추가

amend.md

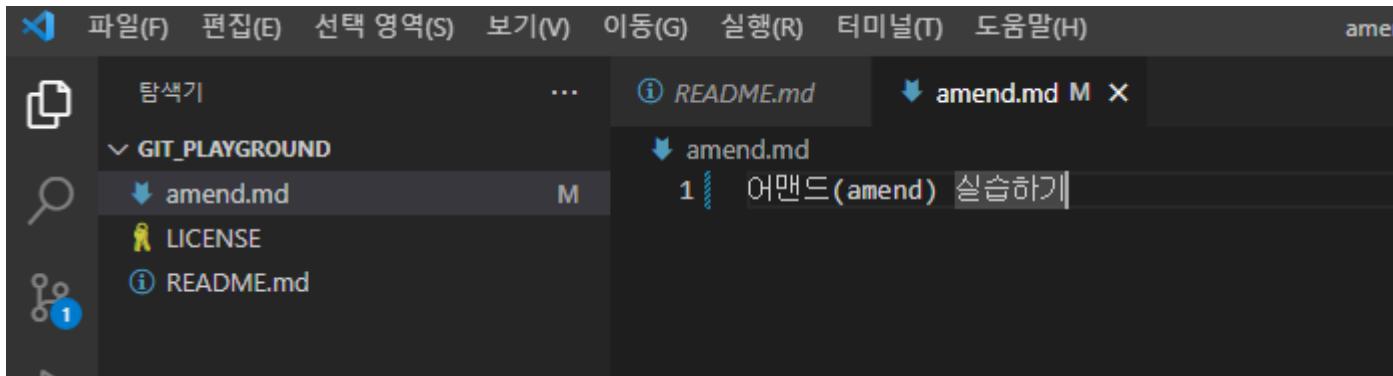
파일 내용

```
+ amend 실습하기  
...\No newline at end of file
```


2. amend

1.amend로 마지막 커밋 수정

- 비주얼 스튜디오에서 변경





2. amend

1.amend로 마지막 커밋 수정

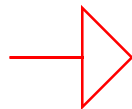
- 스테이지에 올림

대기 중인 파일, 파일 상태순 정렬 ▼ ≡ ▼

스테이지에 올라간 파일 모두 스테이지에서 내리기 선택 내용 스테이지에서 내리기

스테이지에 올라가지 않은 파일 모두 스테이지에 올리기 선택 내용 스테이지에 올리기

amend.md +



그래프	설명	날짜
	커밋하지 않은 변경사항	11 5 2023 21:57
main 1	amend 실습 파일 추가	11 5 2023 21:55
origin/main		11 5 2023 17:54
origin/HEAD		

대기 중인 파일, 파일 상태순 정렬 ▼ ≡ ▼

스테이지에 올라간 파일 모두 스테이지에서 내리기 선택 내용 스테이지에서 내리기

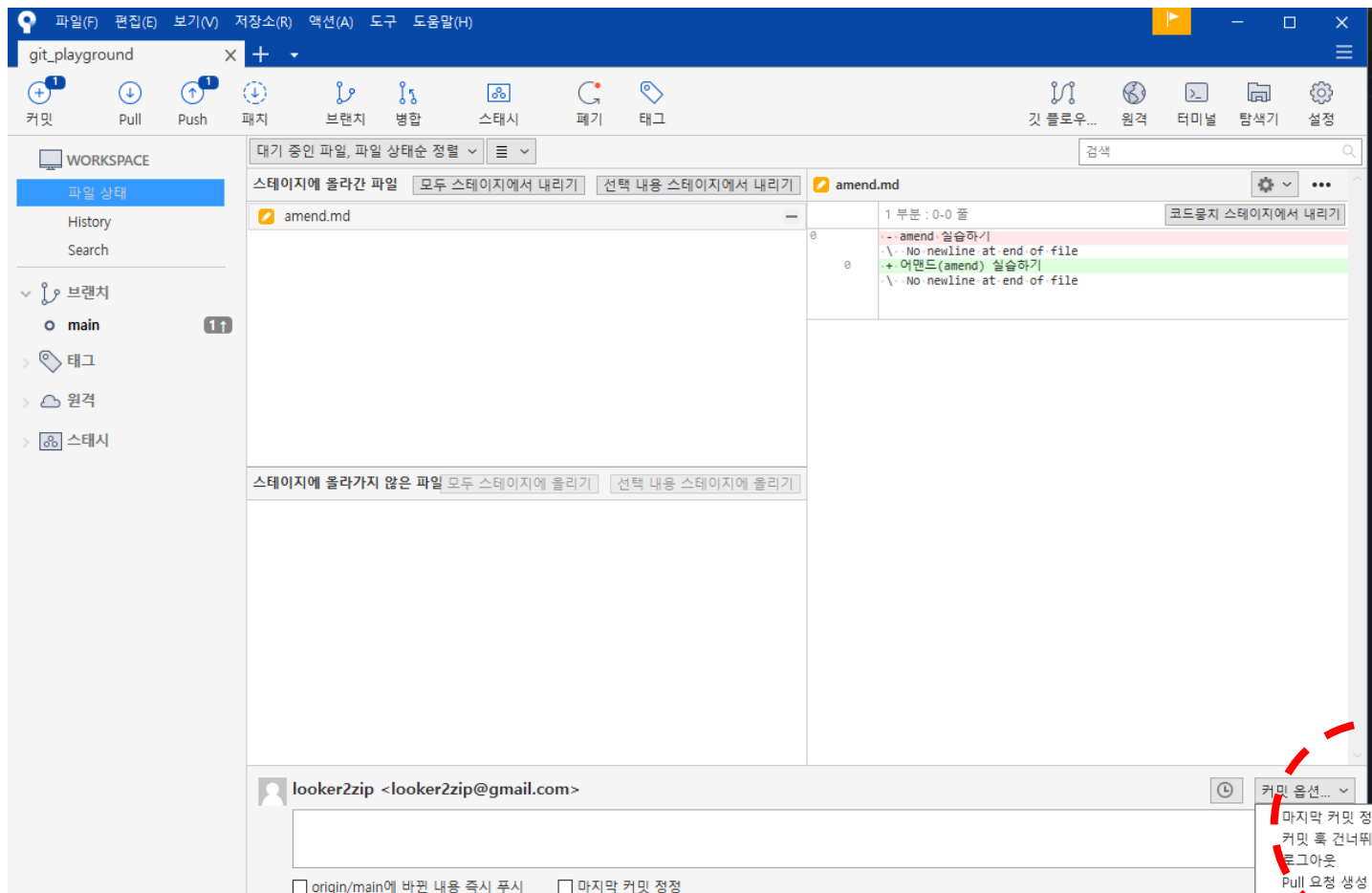
amend.md -

스테이지에 올라가지 않은 파일 모두 스테이지에 올리기 선택 내용 스테이지에 올리기

2. amend

1.amend로 마지막 커밋 수정

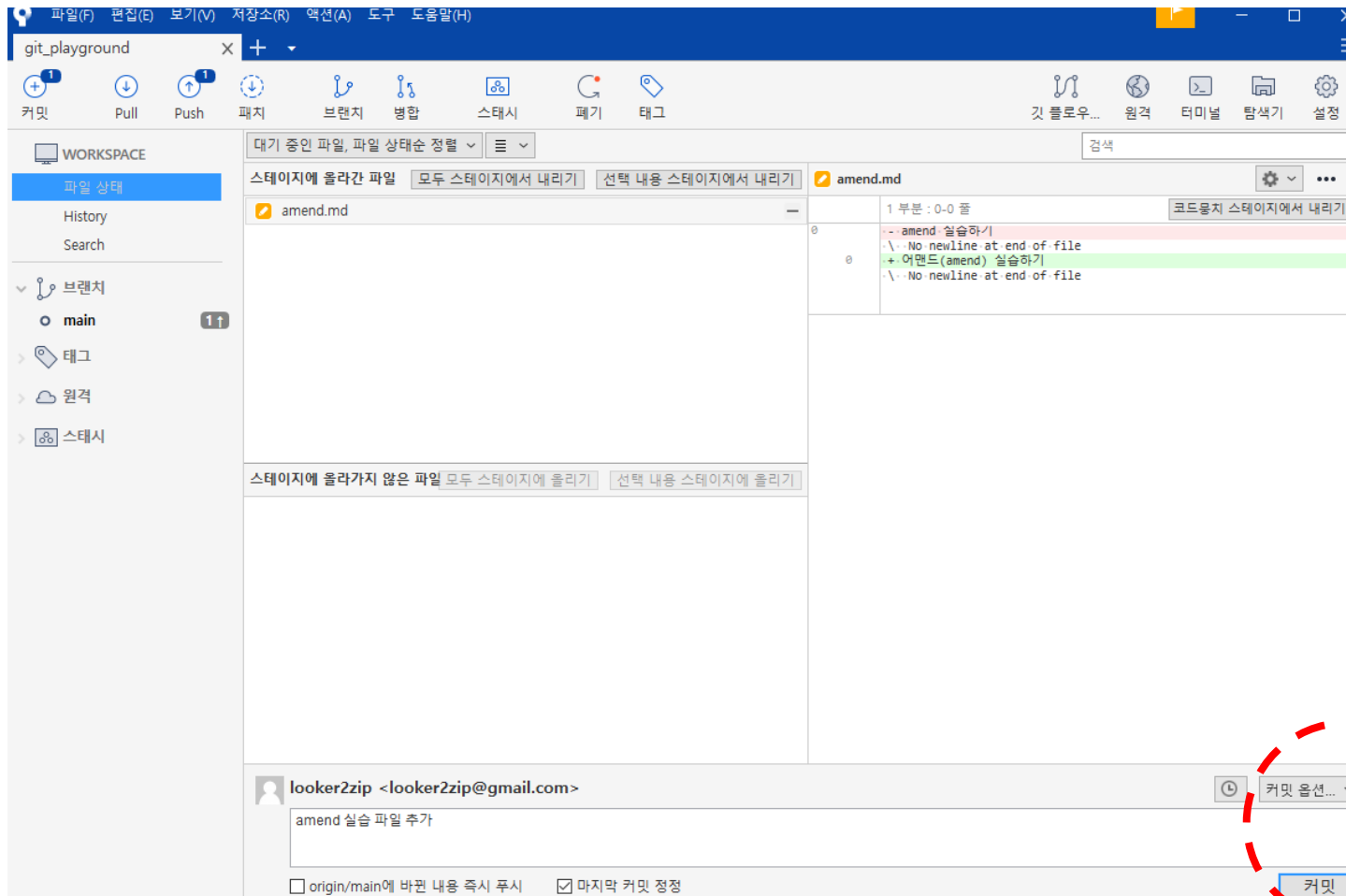
- 커밋 옵션 > 마지막 커밋 정정 버튼 클릭



2. amend

1.amend로 마지막 커밋 수정

- 커밋 옵션 > 마지막 커밋 정정 버튼 클릭



2. amend

1.amend로 마지막 커밋 수정

- History > 커밋 내용이 변경되어 있음

The screenshot shows the Git GUI interface for a repository named 'git_playground'. The left sidebar displays the 'WORKSPACE' section with '파일 상태' (File Status), 'History', and 'Search' options. The 'History' tab is selected, showing a list of commits. The main area displays the commit history table, and the bottom panel shows the details of the selected commit.

그래프	설명	날짜	작성자	커밋
main 1	amend 실습 파일 추가	11 5 2023 22:01	looker2zip <looker2zip@gmail.com>	4b0fa27
origin/main		11 5 2023 17:54	looker2zip <80115650+looker2zip@users.noreply.github>	b0c693e

Commit details for 4b0fa27f535f0871bedb1dc75ecfdb8728a35e9d [4b0fa27]:

- 상위 항목: b0c693e13a
- 작성자: looker2zip <looker2zip@gmail.com>
- 날짜: 2023년 5월 11일 목요일 오후 9:55:02
- 커밋한 사람: looker2zip
- 커밋 날짜: 2023년 5월 11일 목요일 오후 10:01:24

amend 실습 파일 추가

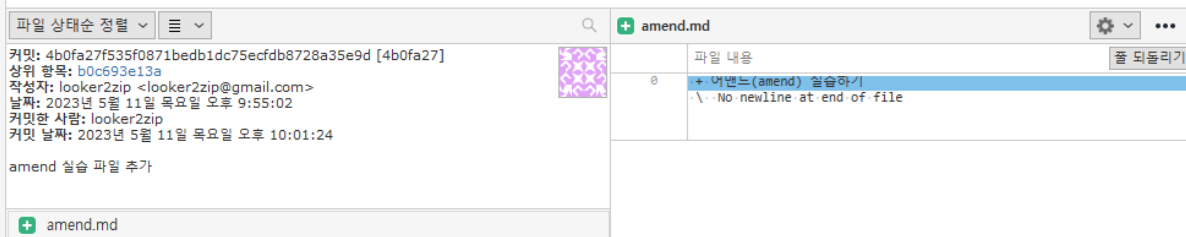
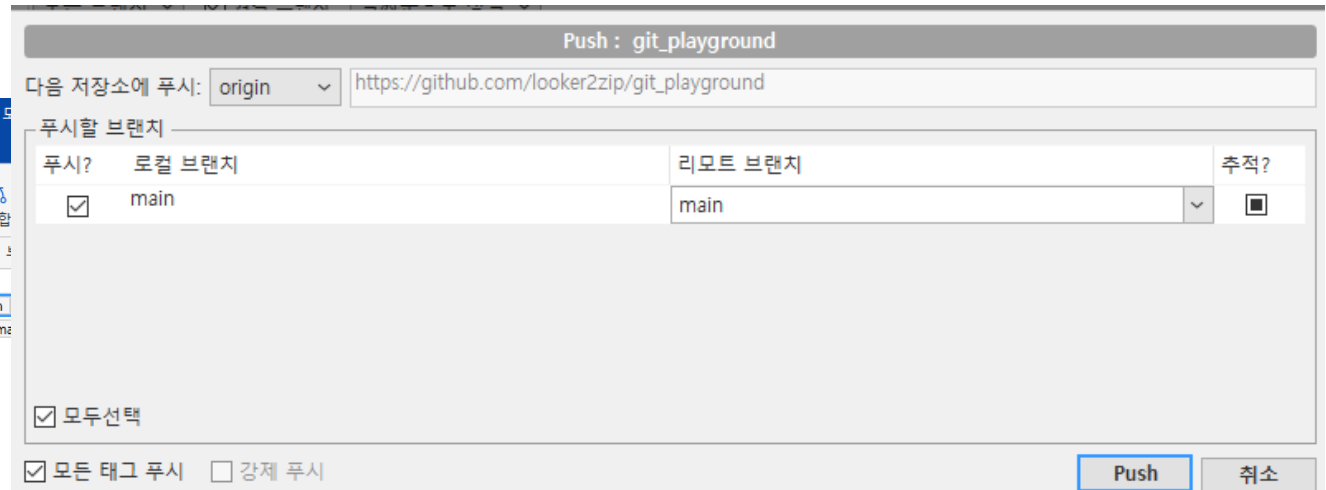
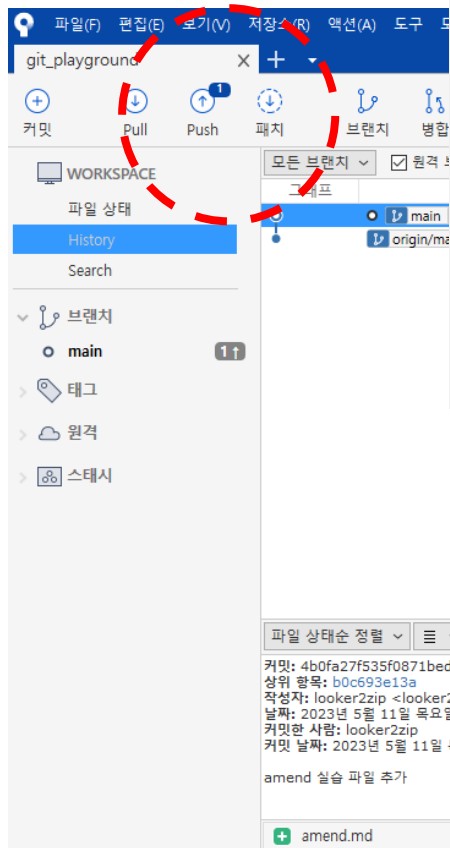
File details for amend.md:

- 파일 내용: + 어맨드(amend) 실습하기
- 내용: \..No newline at end of file

2. amend

2. amend로 커밋 수정 후 푸시하기

- Push 수행



2. amend

2. amend로 커밋 수정 후 푸시하기

- 커밋 메시지 수정
- 마지막 커밋 정정 선택

파일(F) 편집(E) 보기(V) 저장소(R) 액션(A) 도구 도움말(H)

git_playground

커밋 Pull Push 패치 브랜치 병합 스테시 페기 태그

WORKSPACE

파일 상태 History Search

브랜치 main 태그 원격 스테시

커밋 문구를 바꾸시겠습니까?

이전 커밋의 메시지를 위의 커밋 메시지로 바꾸시겠습니까?

예 아니오

커밋할 내용 없음

익스플로러에서 열기

looker2zip <looker2zip@gmail.com>

amend 실습 파일 추가 추가 커밋 메시지 수정

origin/main에 바뀐 내용 즉시 푸시 ☒ 마지막 커밋 정정

커밋 옵션...
✓ 마지막 커밋 커밋 후 건너뛰기
로그아웃
Pull 요청 상

2. amend

2. amend로 커밋 수정 후 푸시하기

- history 변경

The screenshot shows the Git GUI interface with the 'History' tab selected. The commit history table is as follows:

그래프	설명	날짜	작성자	커밋
o main 11↓	amend 실습 파일 추가 추가 커밋 메시지 수정	11 5 2023 22:16	looker2zip <looker2zip@gm	95092f6
o origin/main	amend 실습 파일 추가	11 5 2023 22:01	looker2zip <looker2zip@gmai	4b0fa27
o origin/HEAD	Initial commit	11 5 2023 17:54	looker2zip <80115650+looker	b0c693e

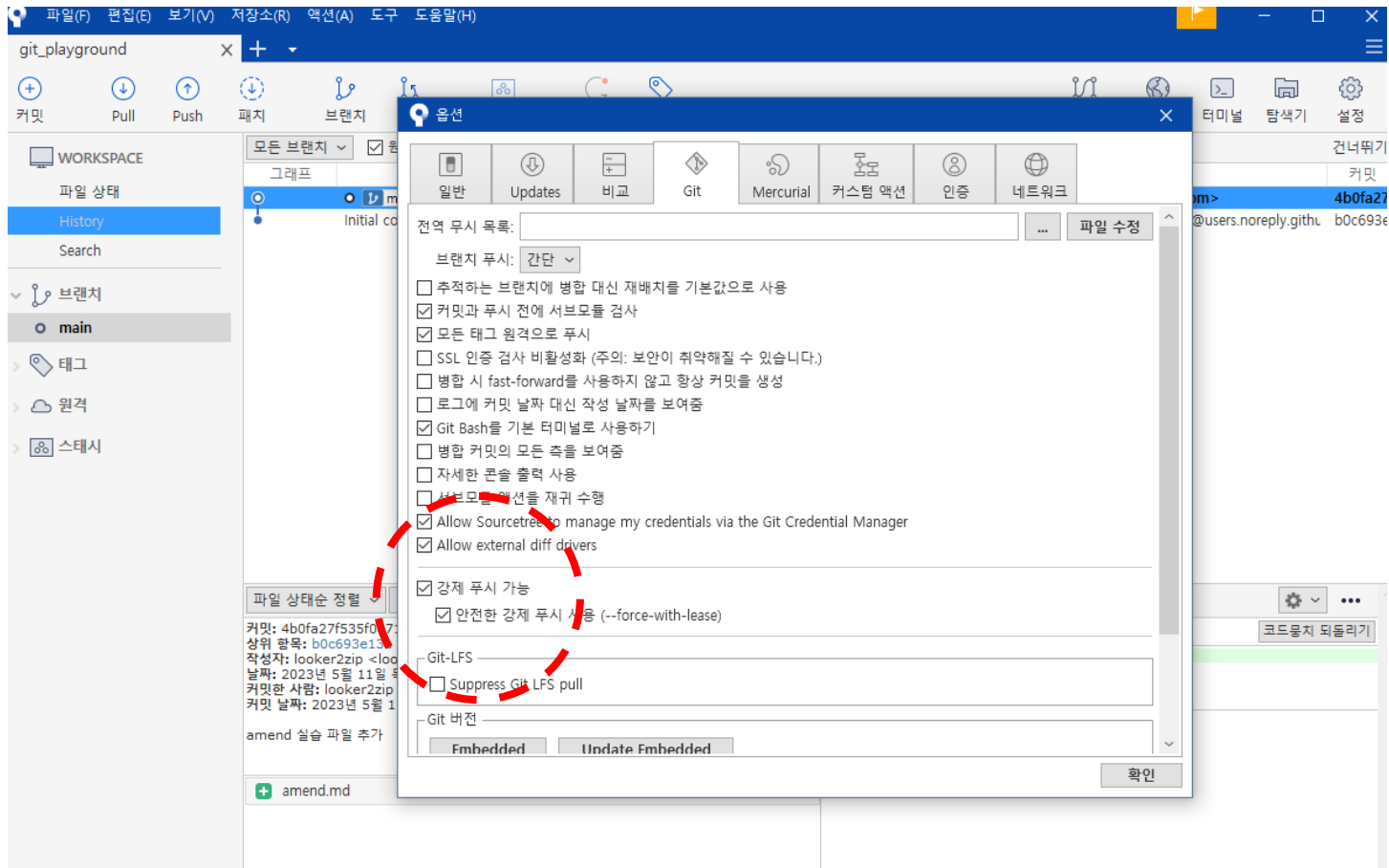
Below the history table, the 'Files' tab for the selected commit (4b0fa27) is shown, displaying the content of the 'amend.md' file:

```
+ 버전노(amend) 실습하기
\..No newline at end of file
```


2. amend

2. amend로 커밋 수정 후 푸시하기

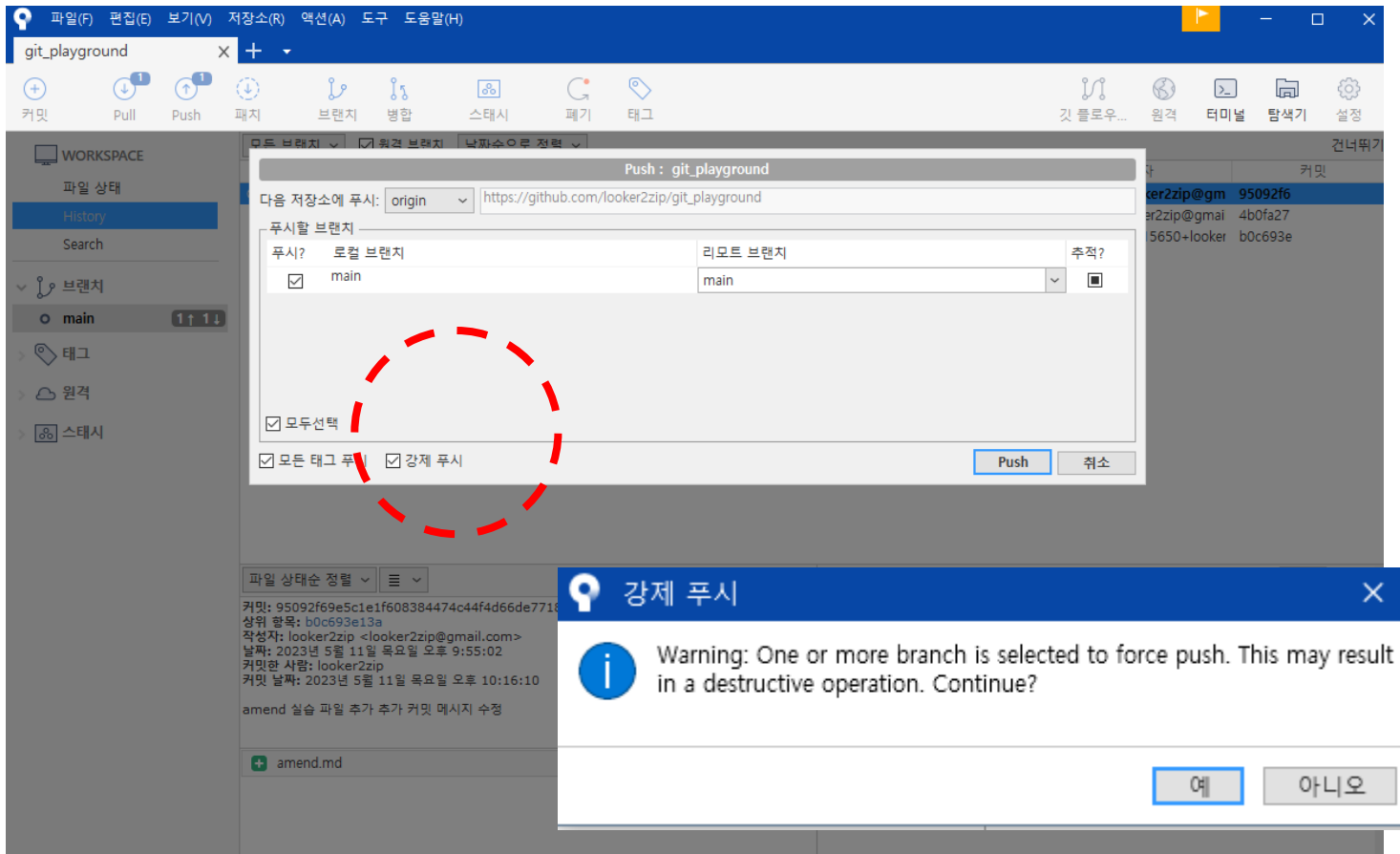
- 도구 > 옵션 > 강제 푸시 가능 > 안전한 강제 푸시 사용 체크



2. amend

2. amend로 커밋 수정 후 푸시하기

- Push 수행



2. amend

3. Push 후 history

- Push 수행 완료

The screenshot shows the VS Code Git Playground interface. The top toolbar includes buttons for Commit, Pull, Push, Patch, Branch, Merge, Stash, and Tag. The left sidebar shows the Workspace with options for File State, History, Search, Branches (main), Tags, Remotes, and Stashes. The main area displays the commit history for the 'main' branch. The current commit is 'amend 실습 파일 추가 추가 커밋 메시지 수정' with a commit hash of 95092f6. The commit message is 'amend 실습 파일 추가 추가 커밋 메시지 수정'. The commit details panel shows the commit hash, parent hash, author, date, and the commit message. The file 'amend.md' is shown with its content: '+ 커맨드 (amend) 실행하기' and '\. No newline at end of file'.

그래프	설명	날짜	작성자	커밋
Initial commit	amend 실습 파일 추가 추가 커밋 메시지 수정	11 5 2023 22:16	looker2zip <looker2zip>	95092f6
		11 5 2023 17:54	looker2zip <80115650+>	b0c693e

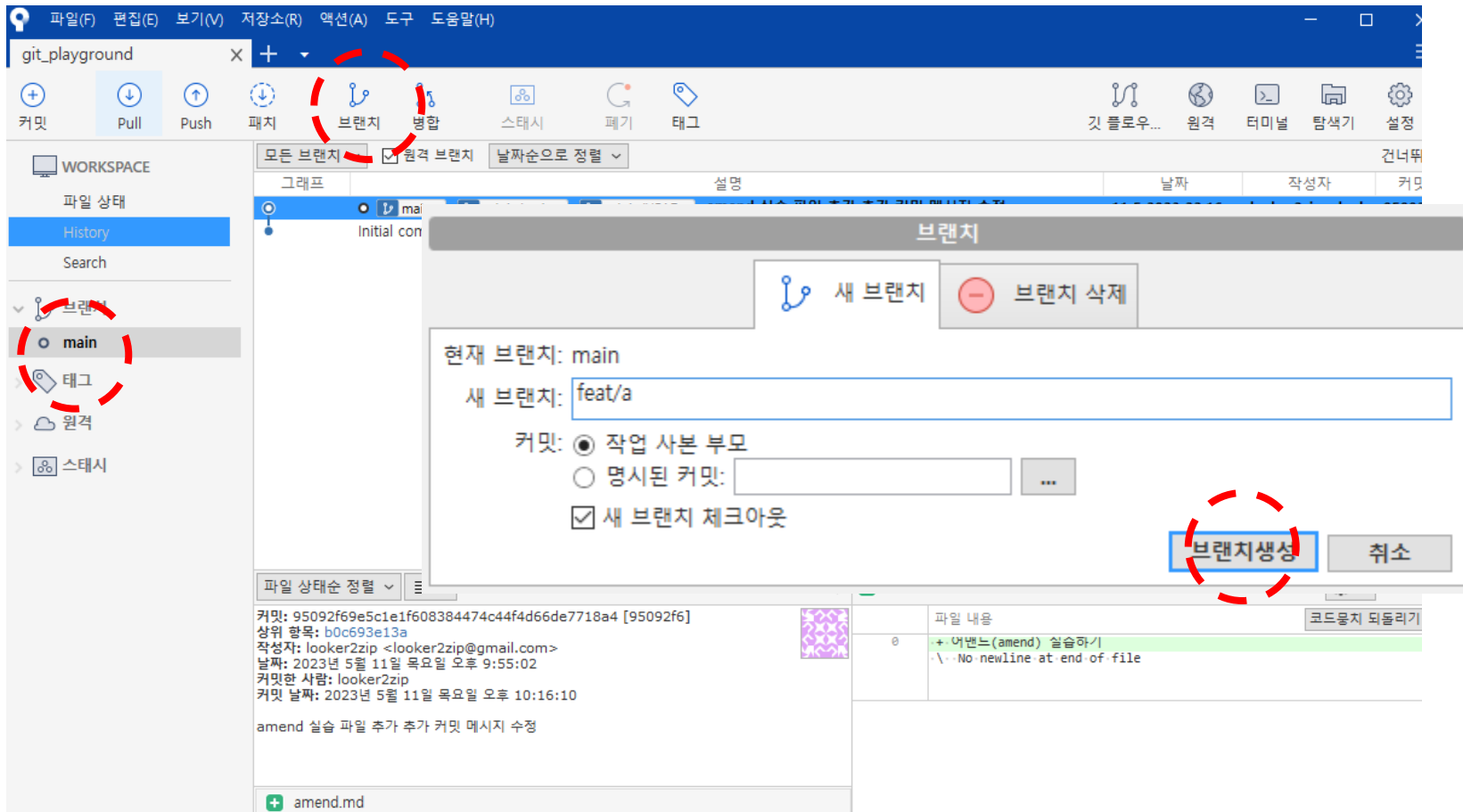
커밋: 95092f69e5c1e1f608384474c44f4d66de7718a4 [95092f6]
상위 항목: b0c693e13a
작성자: looker2zip <looker2zip@gmail.com>
날짜: 2023년 5월 11일 목요일 오후 9:55:02
커밋한 사람: looker2zip
커밋 날짜: 2023년 5월 11일 목요일 오후 10:16:10
amend 실습 파일 추가 추가 커밋 메시지 수정

파일 내용
+ 커맨드 (amend) 실행하기
\. No newline at end of file

3. cherry-pick

1. feat/a 브랜치 생성

- 원하는 체리(커밋)를 선택해, cherry-pick 한다.
- main 브랜치를 베이스로 feat/a 브랜치 생성



3. cherry-pick

1. feat/a 브랜치 생성

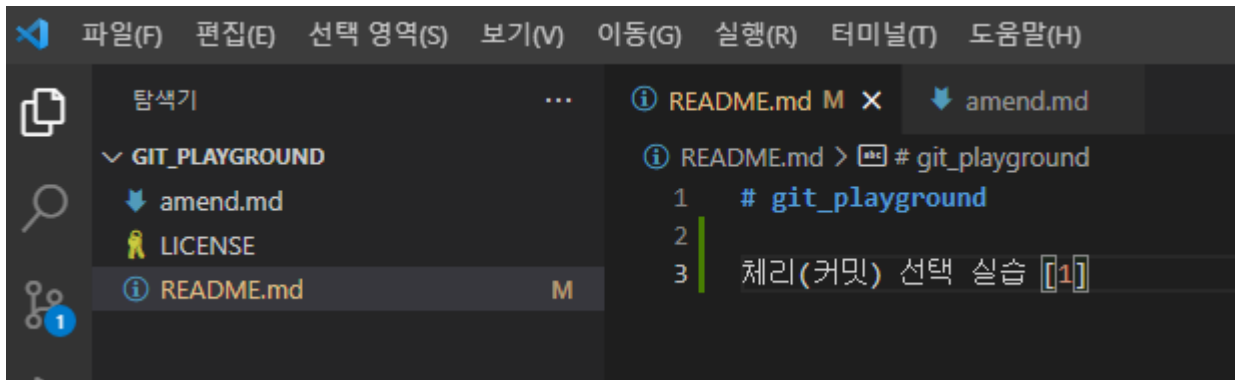
- 원하는 체리(커밋)를 선택해, cherry-pick 한다.
- main 브랜치를 베이스로 feat/a 브랜치 생성

The screenshot shows the Git GUI interface. On the left, the 'WORKSPACE' sidebar is visible. A red dashed circle highlights the '브랜치' (Branches) section, which shows a tree structure: 'feat' (expanded) containing 'a' (selected) and 'main'. The main area displays a commit history table with columns for '날짜' (Date), '작업자' (Author), and '커밋' (Commit). The selected commit is '95092f6' by 'looker2zip' with the message 'amend 실습 파일 추가 추가 커밋 메시지 수정'. Below the table, the commit details for '95092f6' are shown, including the author 'looker2zip <looker2zip@gmail.com>' and the commit message 'amend 실습 파일 추가 추가 커밋 메시지 수정'. The file 'amend.md' is listed at the bottom.

3. cherry-pick

1. feat/a 브랜치 생성

● README.md 편집



```
파일(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 실행(R) 터미널(T) 도움말(H)

검색기
GIT_PLAYGROUND
  amend.md
  LICENSE
  README.md M
  ...

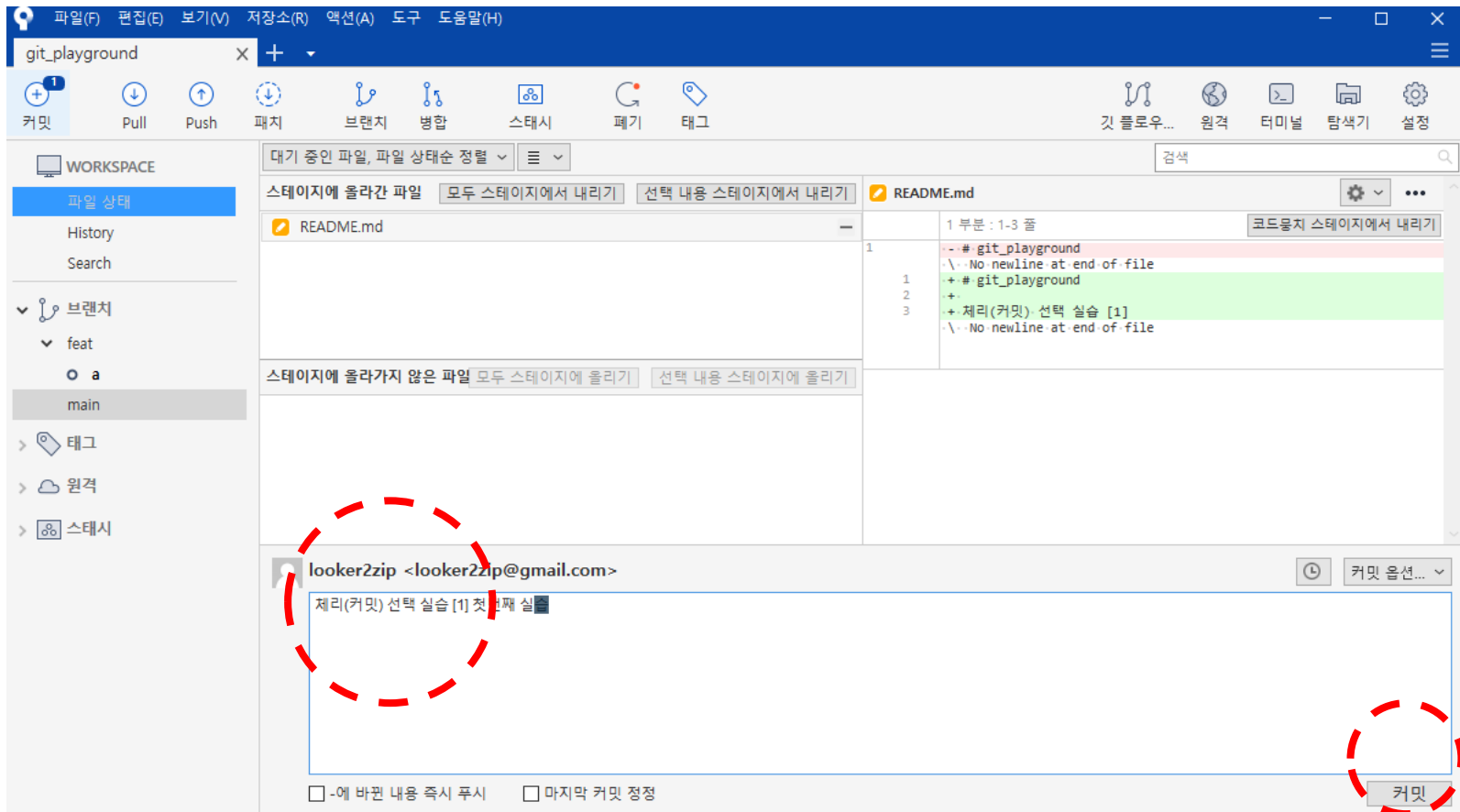
README.md M
  amend.md
  LICENSE
  README.md M
  ...

README.md > # git_playground
1 # git_playground
2
3 체리(커밋) 선택 실습 [1]
```

3. cherry-pick

1. feat/a 브랜치 생성

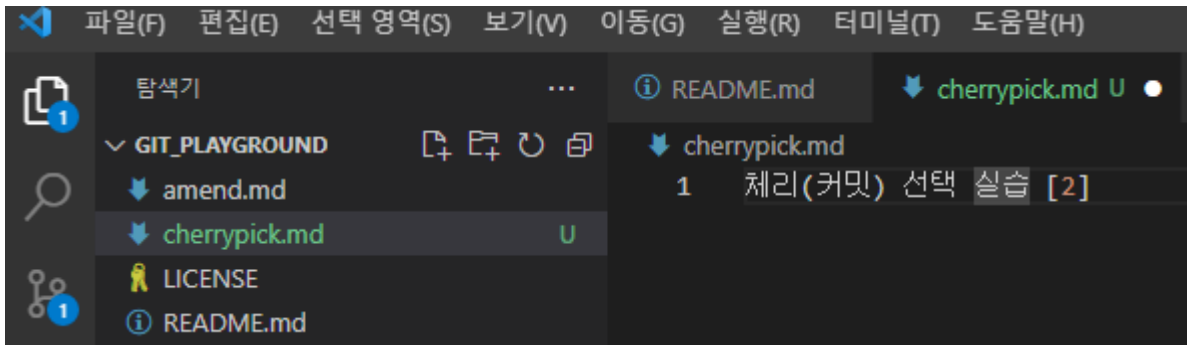
● README.md 편집



3. cherry-pick

2. cherrypick.md 파일 생성

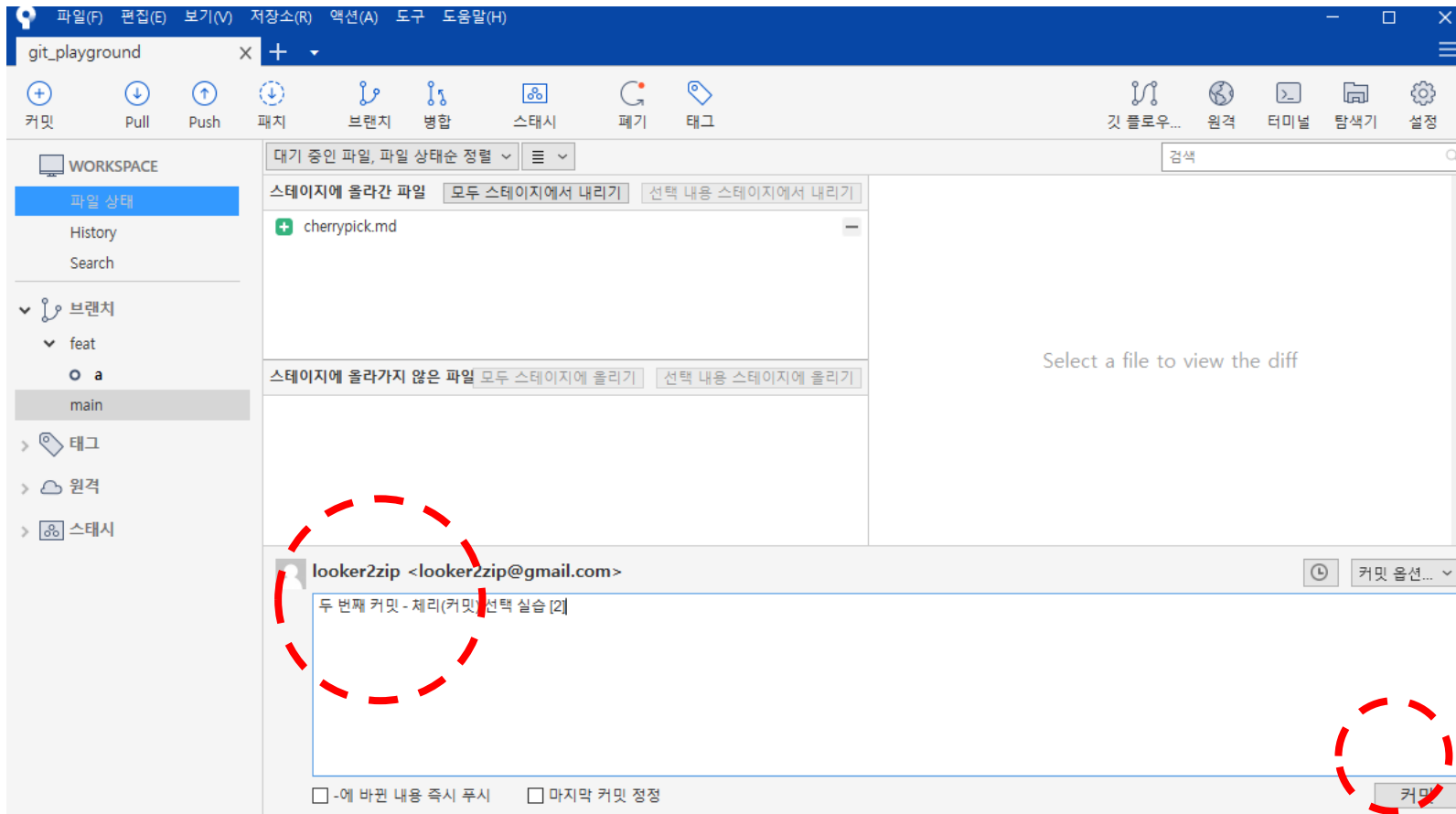
- cherrypick.md



3. cherry-pick

2. cherrypick.md 파일 생성

- cherrypick.md에 커밋하기



3. cherry-pick

2. cherrypick.md 파일 생성

● cherrypick.md에 커밋하기

The screenshot shows the Git GUI interface with the following components:

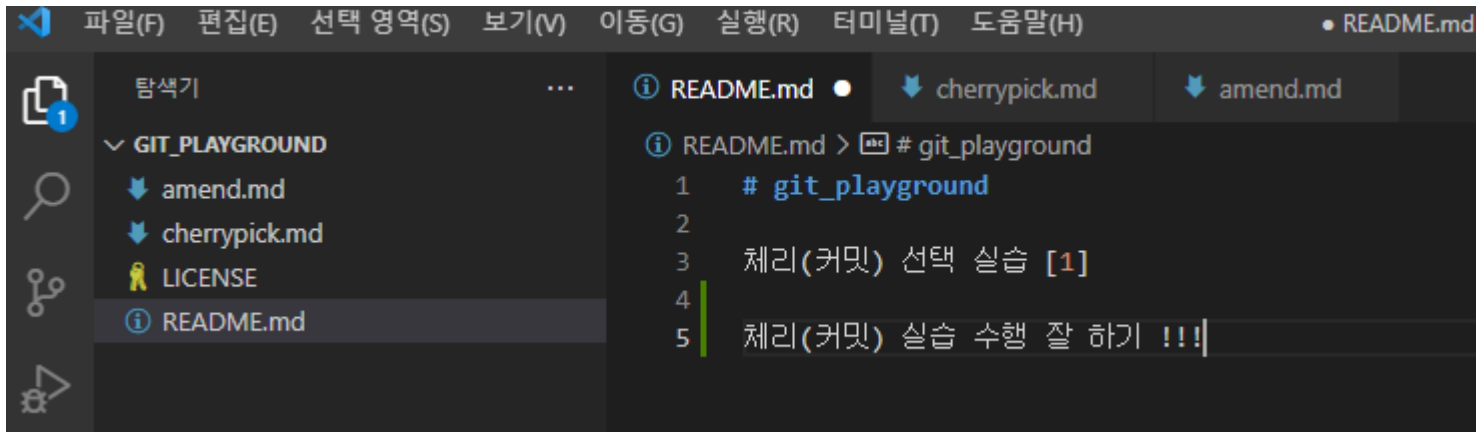
- Toolbar:** Includes buttons for Commit, Pull, Push, Fetch, Branch, Merge, Stash, Revert, and Tag.
- Workspace Panel:** Shows the file state, History, and Search options. The History panel is active, showing a list of commits.
- Commit History Table:**

Commit Hash	Message	Date	Author	Committer
d1bc977	두 번째 커밋 - 체리(커밋) 선택 실습 [2]	12 5 2023 10:46	looker2zip <looker2zip@gmail.com>	looker2zip
79c5f6	체리(커밋) 선택 실습 [1] 첫번째 실습	12 5 2023 10:36	looker2zip <looker2zip@gmail.com>	looker2zip
950921	amend 실습 파일 추가 추가 커밋 메시지 수정	11 5 2023 22:16	looker2zip <looker2zip@gmail.com>	looker2zip
b0c69	Initial commit	11 5 2023 17:54	looker2zip <looker2zip@gmail.com>	looker2zip
- File Panel:** Shows the file state of the selected commit. The file `cherrypick.md` is highlighted.
- File Content:** The content of `cherrypick.md` is displayed, showing a commit message and a file path.

3. cherry-pick

2. cherrypick.md 파일 생성

- README.md 다시 내용 편집



```
파일(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 실행(R) 터미널(T) 도움말(H) • README.md
```

탐색기

- ✓ GIT_PLAYGROUND
 - amend.md
 - cherrypick.md
 - LICENSE
 - README.md

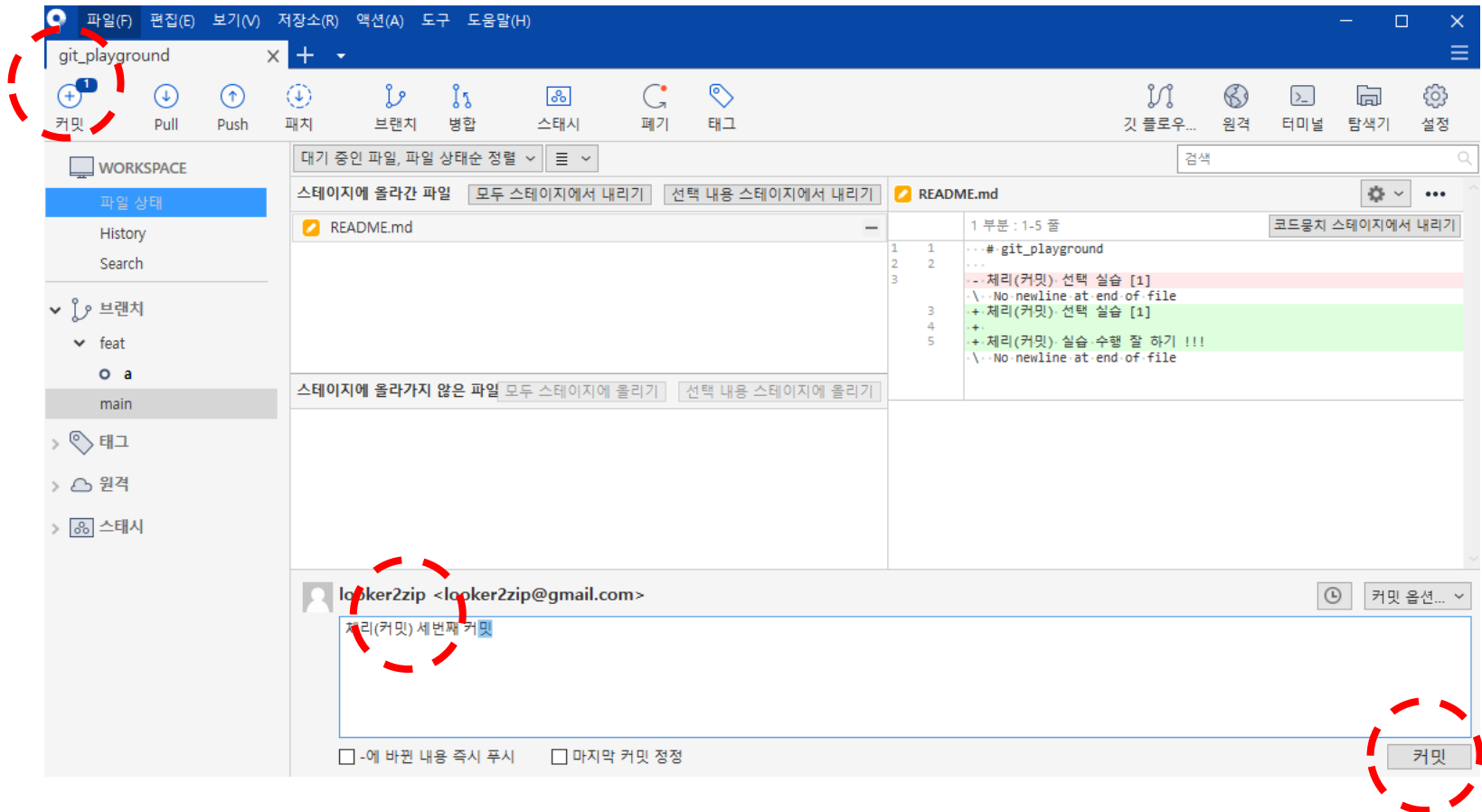
README.md

```
1 # git_playground
2
3 체리(커밋) 선택 실습 [1]
4
5 체리(커밋) 실습 수행 잘 하기 !!!
```

3. cherry-pick

2. cherrypick.md 파일 생성

● 커밋 다시 하기



3. cherry-pick

2. cherrypick.md 파일 생성

- 커밋 그래프 확인

The screenshot shows the git_playground application interface. The top bar includes menu items: 파일(F), 편집(E), 보기(V), 저장소(R), 액션(A), 도구, and 도움말(H). Below the bar is a toolbar with icons for 커밋, Pull, Push, 패치, 브랜치, 병합, 스테시, 페기, and 태그. The left sidebar shows the WORKSPACE with options for 파일 상태, History, Search, 브랜치 (feat, a, main), 태그, 원격, and 스테시. The main area displays the commit graph and the README.md file content.

Commit Graph:

그래프	설명	날짜	작성자	커밋
feat/a	체리(커밋) 세번째 커밋	12 5 2023 10:51	looker2zip <look	7c453d
	두 번째 커밋 - 체리(커밋) 선택 실습 [2]	12 5 2023 10:46	looker2zip <look	d1bc97
	체리(커밋) 선택 실습 [1] 첫번째 실습	12 5 2023 10:36	looker2zip <look	79c5f64
origin/main	amend 실습 파일 추가 추가 커밋 메시지 수정	11 5 2023 22:16	looker2zip <look	95092f
origin/HEAD	Initial commit	11 5 2023 17:54	looker2zip <8011	b0c693

README.md:

```
1 1 ...# git_playground
2 2 ...
3 3 ...
4 4 ...
5 5 ...
```

커밋: 7c453da66790f875d448c6621e59a9970567ce12 [7c453da]
상위 항목: d1bc97707a
작성자: looker2zip <looker2zip@gmail.com>
날짜: 2023년 5월 12일 금요일 오전 10:51:20
커밋한 사람: looker2zip

체리(커밋) 세번째 커밋

3. cherry-pick

3. feat/b 브랜치 생성

- main 브랜치 확인 후, feat/b 브랜치 생성

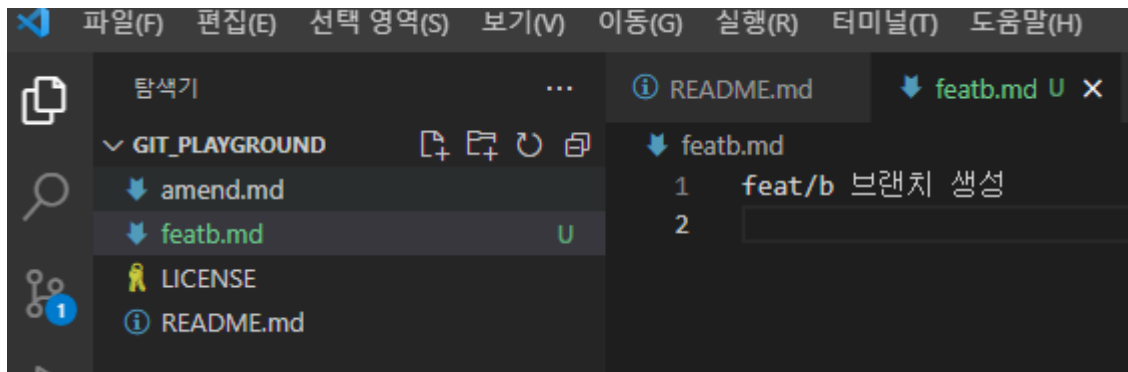
The screenshot shows the Git GUI interface with the following elements:

- Toolbar:** Includes icons for Commit, Pull, Push, Patch, Branch, Merge, Stash, Revert, and Tag. The 'Branch' icon is circled in red.
- Left Panel:** Shows the 'WORKSPACE' section with '파일 상태' (File Status), 'History', and 'Search'. Below it, the '브랜치' (Branch) section shows a tree with 'feat' and 'main' branches. The 'main' branch is selected and circled in red.
- Graph View:** Displays the commit history with 'feat/a' and 'main' branches. 'main' is the current branch.
- Branch Dialog:** A modal window titled '브랜치' (Branch) is open. It shows '현재 브랜치: main' (Current branch: main) and '새 브랜치: feat/b' (New branch: feat/b). The '커밋' (Commit) section has '작업 사본 부모' (Use parent of work copy) selected. The '새 브랜치 체크아웃' (Checkout new branch) checkbox is checked. The '브랜치 생성' (Create branch) button is circled in red.
- Bottom Panel:** Shows the 'amend' command being used to amend the last commit.

3. cherry-pick

3. feat/b 브랜치 생성

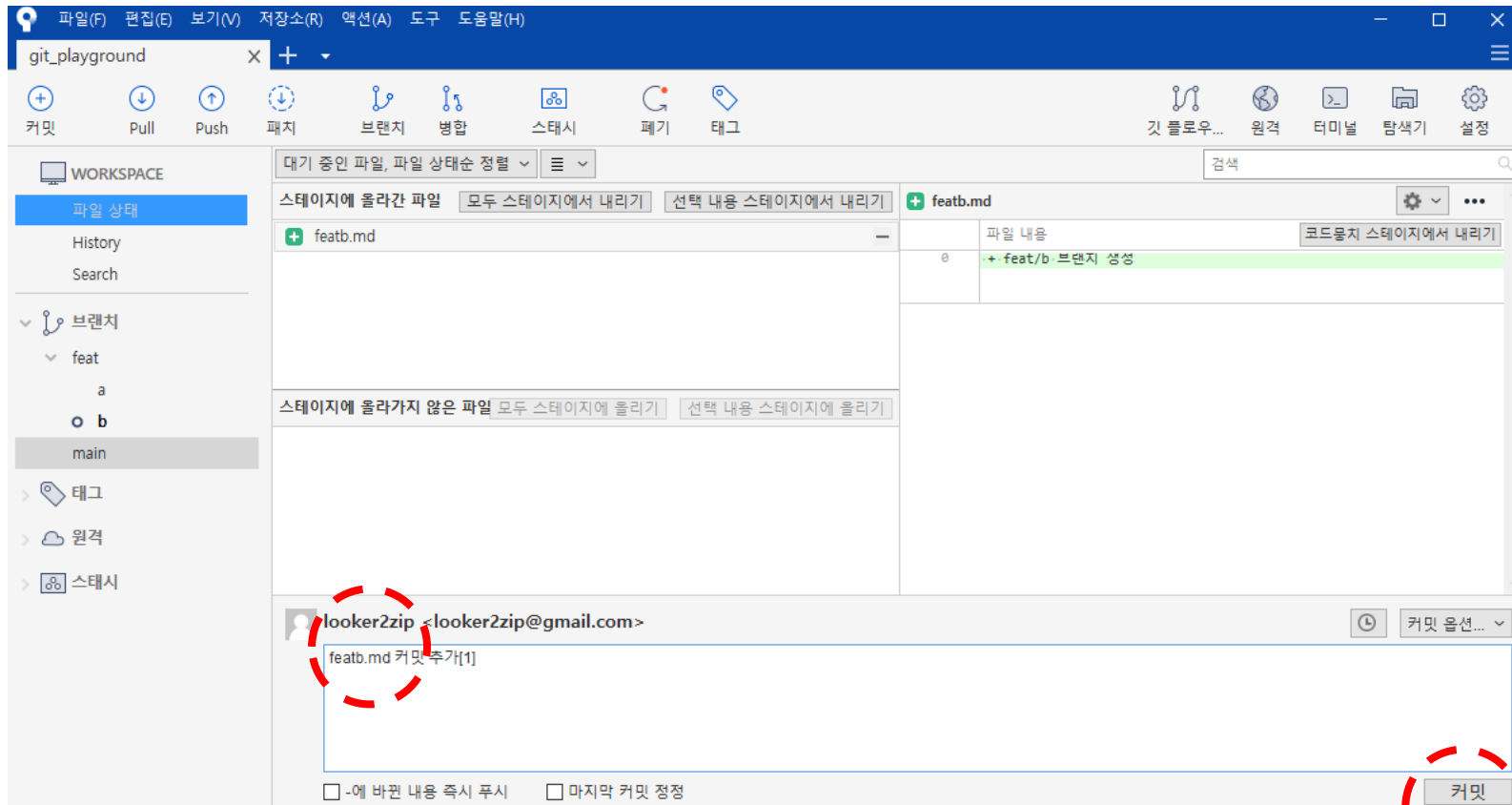
- featb.md 파일 생성



3. cherry-pick

3. feat/b 브랜치 생성

- featb.md 커밋 추가



3. cherry-pick

4. cherry-pick 하기

- feat/b 브랜치에 있는 상태에서 복제하길 원하는 “두 번째 커밋 - 체리(커밋) 선택 실습 [2]”위에서 우측 버튼을 클릭하고 [체리 픽]을 선택한다.

The screenshot illustrates the steps to perform a cherry-pick in Git GUI:

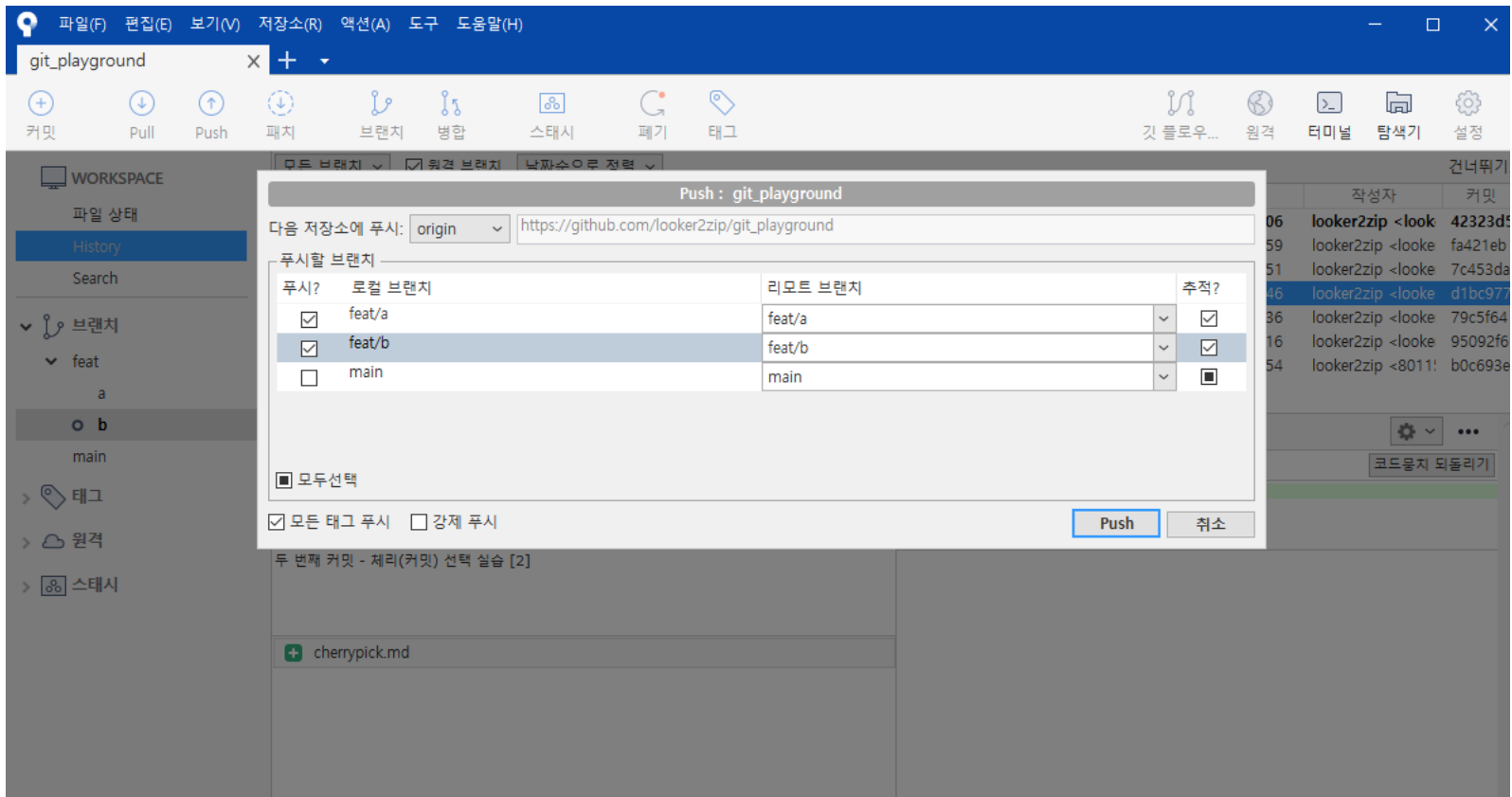
- The left sidebar shows the 'b' branch selected under the 'feat' folder.
- The commit history in the center shows the '두 번째 커밋 - 체리(커밋) 선택 실습 [2]' commit on the 'feat/b' branch.
- A right-click context menu is open over this commit, with the '체리 픽' (Cherry-pick) option selected.
- A dialog box titled '체리 픽' (Cherry-pick) is displayed, explaining that cherry-picking applies changes from a specific commit to the current branch. It includes options to 'Commit immediately upon successful merge' and 'Include commit IDs in the merge commit message', both of which are checked.
- The '확인' (Confirm) button in the dialog box is highlighted with a red dashed circle.



4. reset

1. feat/a, feat/b 브랜치 푸시

- Push



4. reset

2. 브랜치 되돌리기

- “featb.md 커밋 추가[1]” 커밋에서 우측 버튼을 클릭하고, > 이 커밋까지 현재 브랜치를 초기화 버튼을 클릭

The screenshot shows the Git Playground interface. On the left, the 'WORKSPACE' sidebar shows the 'feat/b' branch selected. The main area displays a commit history table. A context menu is open over the commit 'fa421eb' (hash: 95092f69e5), which is the commit 'featb.md 커밋 추가[1]'. The menu option '이 커밋까지 현재 브랜치를 초기화' (Reset this branch to this commit) is highlighted by a red dashed circle.

설명	날짜	작성자	커밋
featb.md 커밋 추가[1]	12 5 2023 11:06	looker2zip <look	42323d5
이 커밋까지 현재 브랜치를 초기화	12 5 2023 10:59	looker2zip <look	fa421eb
featb.md 커밋 추가[2]	12 5 2023 10:51	looker2zip <look	7c453da
featb.md 커밋 추가[3]	12 5 2023 10:46	looker2zip <look	d1bc977
featb.md 커밋 추가[4]	12 5 2023 10:36	looker2zip <look	79c5f64
featb.md 커밋 추가[5]	11 5 2023 22:16	looker2zip <look	95092f6
featb.md 커밋 추가[6]	11 5 2023 17:54	looker2zip <look	b0c693e

4. reset

2. 브랜치 되돌리기

- [Mixed]모드를 선택하고 확인

커밋 초기화...

브랜치 포인터를 옮기겠습니까?

브랜치 초기화: feat/b

커밋할 것: fa421eb: featb.md 커밋 추가[1]

사용 중인 모드: Mixed - 작업 상태는 그대로 두지만 인덱스는 리셋 ▾

- Soft - 모든 로컬 변경사항을 유지
- Mixed - 작업 상태는 그대로 두지만 인덱스는 리셋
- Hard - 모든 작업 상태 내 변경 사항을 버림

커밋 초기화...

브랜치 포인터를 옮기겠습니까?

브랜치 초기화: feat/b

커밋할 것: fa421eb: featb.md 커밋 추가[1]

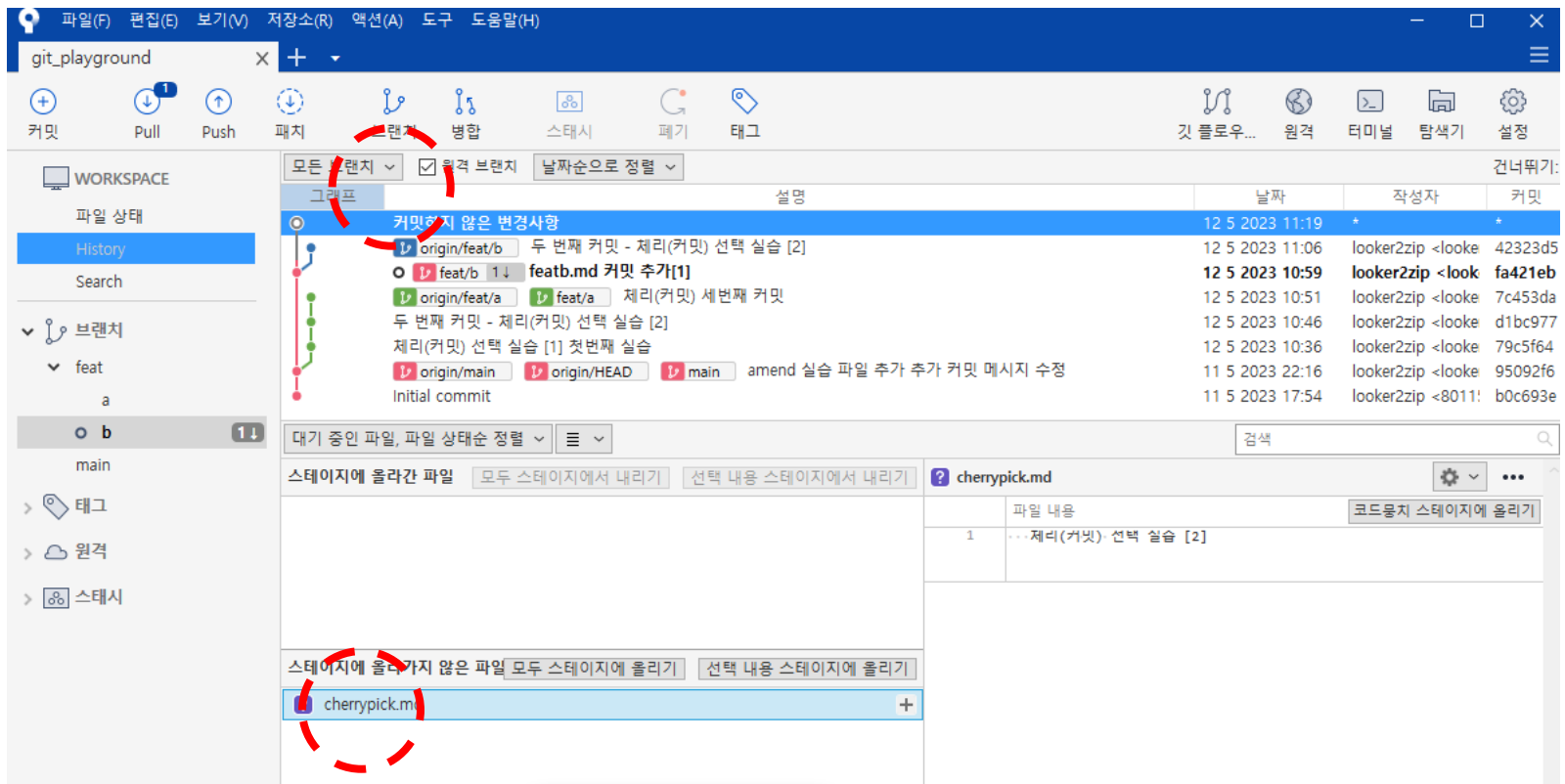
사용 중인 모드: Mixed - 작업 상태는 그대로 두지만 인덱스는 리셋 ▾

확인 취소

4. reset

2. 브랜치 되돌리기

- “featb.md 커밋 추가[1]” 커밋으로 되돌아 갔음. ‘커밋하지 않은 변경사항’이 생김



The screenshot shows the Git GUI interface with the following details:

- Toolbar:** Includes buttons for Commit, Pull, Push, Patch, Branch, Merge, Stash, Revert, and Tag.
- Workspace:** Shows the current state of the files. The 'cherrypick.m' file is highlighted.
- Commit History:** A table of commits is displayed. The commit 'featb.md 커밋 추가[1]' is highlighted.
- Commit Table:**

Commit Hash	Author	Message
42323d5	looker2zip <looker2zip@gmail.com>	두 번째 커밋 - 제리(커밋) 선택 실습 [2]
fa421eb	looker2zip <looker2zip@gmail.com>	featb.md 커밋 추가[1]
7c453da	looker2zip <looker2zip@gmail.com>	제리(커밋) 세 번째 커밋
d1bc977	looker2zip <looker2zip@gmail.com>	두 번째 커밋 - 제리(커밋) 선택 실습 [2]
79c5f64	looker2zip <looker2zip@gmail.com>	제리(커밋) 선택 실습 [1] 첫 번째 실습
95092f6	looker2zip <looker2zip@gmail.com>	amend 실습 파일 추가 추가 커밋 메시지 수정
b0c693e	looker2zip <looker2zip@gmail.com>	Initial commit

4. reset

3. Hard reset

- feat/b 브랜치에 있는 상태에서 복제하길 원하는 “두 번째 커밋 - 체리(커밋) 선택 실습 [2]”위에서 우측 버튼을 클릭하고, > 이 커밋까지 현재 브랜치를 초기화 버튼을 클릭

git_playground

WORKSPACE

파일 상태

History

Search

브랜치

feat

a

b

main

태그

원격

스태시

모든 브랜치

원격 브랜치

날짜순으로 정렬

그래프

커밋하지 않은 변경사항

origin/feat/b

feat/b

origin/feat/a

feat/a

두 번째 커밋 - 체리(커밋) 선택 실습 [2]

체리(커밋) 선택 실습 [1]

origin/main

origin/HEAD

main

amend

Initial commit

파일 상태순 정렬

커밋: 42323d5bc53cd9dd8f46d9470422b95ac5ea2621 [42323d5]

설명

날짜

작성자

커밋

12 5 2023 11:19

12 5 2023 11:06

12 5 2023 10:59

12 5 2023 10:51

12 5 2023 10:46

12 5 2023 10:36

11 5 2023 22:16

11 5 2023 17:54

looker2zip <looker2zip>

looker2zip <looker2zip>

looker2zip <looker2zip>

looker2zip <looker2zip>

looker2zip <looker2zip>

looker2zip <looker2zip>

looker2zip <looker2zip>

looker2zip <looker2zip>

42323d5

fa421eb

7c453da

d1bc977

79c5f64

95092f6

b0c693e

체크아웃...

병합...

재배치...

태그...

아카이브...

브랜치...

42323d5의 자식 커밋을 상방향 재배치...

이 커밋까지 현재 브랜치를 초기화

보드에 복사

커밋 초기화...

브랜치 포인터를 옮기겠습니까?

브랜치 초기화: feat/b

커밋할 것: 42323d5: 두 번째 커밋 - 체리(커밋) 선택 실습 [2]

사용 중인 모드: Hard - 모든 작업 상태 내 변경 사항을 버림

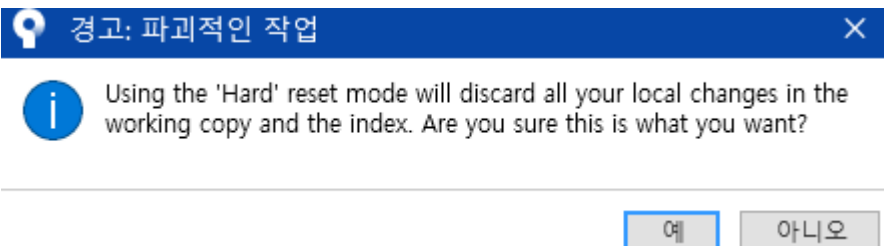
확인

취소

4. reset

3. Hard reset

- feat/b 브랜치에 있는 상태에서 복제하길 원하는 “두 번째 커밋 - 체리(커밋) 선택 실습 [2]”위에서 우측 버튼을 클릭하고, > 이 커밋까지 현재 브랜치를 초기화 버튼을 클릭



4. reset

3. Hard reset

- 되돌아 옴

The screenshot shows the Git GUI interface for a repository named 'git_playground'. The left sidebar displays the workspace structure with branches 'feat', 'a', and 'b' (selected), and 'main'. The main panel shows a commit history table with columns for commit hash, description, date, author, and committer. The selected commit is '42323d5' with the message '두 번째 커밋 - 제리(커밋) 선택 실습 [2]'. Below the table, the file 'cherrypick.md' is shown with a diff view highlighting the changes made in the selected commit.

커밋	날짜	작성자	커밋
42323d5	12 5 2023 11:06	looker2zip <looker2zip@gmail.com>	두 번째 커밋 - 제리(커밋) 선택 실습 [2]
fa421eb	12 5 2023 10:59	looker2zip <looker2zip@gmail.com>	featb.md 커밋 추가[1]
7c453da	12 5 2023 10:51	looker2zip <looker2zip@gmail.com>	제리(커밋) 세번째 커밋
d1bc977	12 5 2023 10:46	looker2zip <looker2zip@gmail.com>	두 번째 커밋 - 제리(커밋) 선택 실습 [2]
79c5f64	12 5 2023 10:36	looker2zip <looker2zip@gmail.com>	제리(커밋) 선택 실습 [1] 첫번째 실습
95092f6	11 5 2023 22:16	looker2zip <looker2zip@gmail.com>	amend 실습 파일 추가 추가 커밋 메시지 수정
b0c693e	11 5 2023 17:54	looker2zip <looker2zip@gmail.com>	Initial commit

커밋: 42323d5bc53cd9dd8f46d9470422b95ac5ea2621 [42323d5]
상위 항목: fa421eb404
작성자: looker2zip <looker2zip@gmail.com>
날짜: 2023년 5월 12일 금요일 오전 10:46:16
커밋한 사람: looker2zip
커밋 날짜: 2023년 5월 12일 금요일 오전 11:06:36

두 번째 커밋 - 제리(커밋) 선택 실습 [2]
(cherry picked from commit d1bc97707aeb204ba391e1eddf92580c303e109)

cherrypick.md

파일 내용

```
+ 제리(커밋) 선택 실습 [2]  
+ \.No newline at end of file
```


4. reset

3. Hard reset

- featb.md 커밋 추가[1]로 돌아가기

The screenshot shows the Git GUI interface with a commit history table. A context menu is open over the commit 'fa421eb'.

날짜	설명	날짜	작성자	커밋
12 5 2023 11:06	두 번째 커밋 - 체리(커밋) 선택 실습 [2]		looker2zip <look	42323d
12 5 2023 10:59	featb.md 커밋 추가[1]		looker2zip <look	fa421eb
12 5 2023 10:51	origin/feat/a		looker2zip <look	7c453d
12 5 2023 10:46	두 번째 커밋 - 체리(커밋) 선택 실습		looker2zip <look	d1bc97
12 5 2023 10:36	체리(커밋) 선택 실습		looker2zip <look	79c5f6
11 5 2023 22:16	커밋 메시지 수정		looker2zip <look	95092f
11 5 2023 17:54			looker2zip <8011	b0c693

Context menu options for commit fa421eb:

- 체크아웃...
- 병합...
- 재배치...
- 태그...
- 아카이브...
- 브랜치...
- fa421eb의 자식 커밋을 쌍방향 재배치...
- 이 커밋까지 현재 브랜치를 초기화
- 커밋 되돌리기...
- 패치 생성...
- 체리 픽
- SHA 값을 클립보드에 복사
- 커스텀 액션

4. reset

3. Hard reset

- featb.md 커밋 추가[1]로 돌아가기

커밋 초기화...

브랜치 포인터를 옮기겠습니까?
브랜치 초기화: feat/b
커밋할 것: fa421eb: featb.md 커밋 추가[1]

사용 중인 모드: Hard - 모든 작업 상태 내 변경 사항을 버림 ▼

확인 취소



경고: 파괴적인 작업



Using the 'Hard' reset mode will discard all your local changes in the working copy and the index. Are you sure this is what you want?

예

아니오

4. reset

3. Hard reset

- featb.md 커밋 추가[1]로 돌아간 화면

The screenshot shows the Git GUI interface with the following components:

- Toolbar:** Includes icons for commit, pull, push, patch, branch, merge, stash, reset, and tag.
- Workspace:** Shows the file state, history, and search options.
- Commit History:** A table of commits with columns for commit hash, description, date, author, and commit message. The current commit is highlighted in blue.
- Commit Details:** A section showing the commit hash, author, date, and message for the selected commit.
- File List:** A list of files in the current commit, including 'featb.md'.

Commit Hash	Description	Date	Author	Commit Message
42323d5	두 번째 커밋 - 체리(커밋) 선택 실습 [2]	12 5 2023 11:06	looker2zip <looker2zip@gmail.com>	
fa421eb	featb.md 커밋 추가[1]	12 5 2023 10:59	looker2zip <looker2zip@gmail.com>	
7c453da	체리(커밋) 세 번째 커밋	12 5 2023 10:51	looker2zip <looker2zip@gmail.com>	
d1bc977	두 번째 커밋 - 체리(커밋) 선택 실습 [2]	12 5 2023 10:46	looker2zip <looker2zip@gmail.com>	
79c5f64	체리(커밋) 선택 실습 [1] 첫 번째 실습	12 5 2023 10:36	looker2zip <looker2zip@gmail.com>	
95092f6	amend 실습 파일 추가 추가 커밋 메시지 수정	11 5 2023 22:16	looker2zip <looker2zip@gmail.com>	
b0c693e	Initial commit	11 5 2023 17:54	looker2zip <looker2zip@gmail.com>	

Commit Details for fa421eb:

- Commit Hash: fa421eb4040a1d9cc02502165fea7ed51b8cbfb4 [fa421eb]
- Author: looker2zip <looker2zip@gmail.com>
- Date: 2023년 5월 12일 금요일 오전 10:59:30
- Committed by: looker2zip

File List:

- featb.md

4. reset

4. Push

- 강제 푸시

The screenshot shows the Git GUI interface with a warning dialog and a push dialog. The warning dialog, titled "강제 푸시" (Force Push), contains the message: "Warning: One or more branch is selected to force push. This may result in a destructive operation. Continue?". It has "예" (Yes) and "아니오" (No) buttons. The push dialog, titled "Push : git_playground", shows the repository URL "https://github.com/looker2zip/git_playground" and a table of branches to be pushed. The "feat/b" branch is selected for pushing. The "모든 태그 푸시" (Push all tags) and "강제 푸시" (Force push) checkboxes are checked. The "Push" button is highlighted with a red dashed circle.

Warning: One or more branch is selected to force push. This may result in a destructive operation. Continue?

예 아니오

Push : git_playground

다음 저장소에 푸시: origin https://github.com/looker2zip/git_playground

푸시할 브랜치	로컬 브랜치	리모트 브랜치	추적?
<input type="checkbox"/>	feat/a	feat/a	<input type="checkbox"/>
<input checked="" type="checkbox"/>	feat/b	feat/b	<input checked="" type="checkbox"/>
<input type="checkbox"/>	main	main	<input checked="" type="checkbox"/>

☒ 모두선택

☒ 모든 태그 푸시 ☒ 강제 푸시

Push 취소

4. reset

4. Push

- feat/b로 돌아옴

The screenshot shows the Git GUI interface for a repository named 'git_playground'. The left sidebar displays the workspace structure with branches 'feat', 'a', and 'b' (selected), and tags and stashes. The main area shows a commit history table with columns for commit hash, description, date, author, and committer. The selected commit is 'fa421eb' with the message 'featb.md 커밋 추가[1]'. Below the table, the file 'featb.md' is shown with a diff view indicating a new branch creation.

커밋	날짜	작성자	커밋
fa421eb	12 5 2023 10:59	looker2zip <looker2zip@gmail.com>	featb.md 커밋 추가[1]
7c453da	12 5 2023 10:51	looker2zip <looker2zip@gmail.com>	제리(커밋) 세번째 커밋
d1bc977	12 5 2023 10:46	looker2zip <looker2zip@gmail.com>	두 번째 커밋 - 제리(커밋) 선택 실습 [2]
79c5f64	12 5 2023 10:36	looker2zip <looker2zip@gmail.com>	제리(커밋) 선택 실습 [1] 첫번째 실습
95092f6	11 5 2023 22:16	looker2zip <looker2zip@gmail.com>	amend 실습 파일 추가 추가 커밋 메시지 수정
b0c693e	11 5 2023 17:54	looker2zip <looker2zip@gmail.com>	Initial commit

파일 상태순 정렬

featb.md 커밋 추가[1]

커밋: fa421eb4040a1d9cc02502165fea7ed51b8cbfb4 [fa421eb]
상위 항목: 95092f69e5
작성자: looker2zip <looker2zip@gmail.com>
날짜: 2023년 5월 12일 금요일 오전 10:59:30
커밋한 사람: looker2zip

featb.md 커밋 추가[1]

featb.md

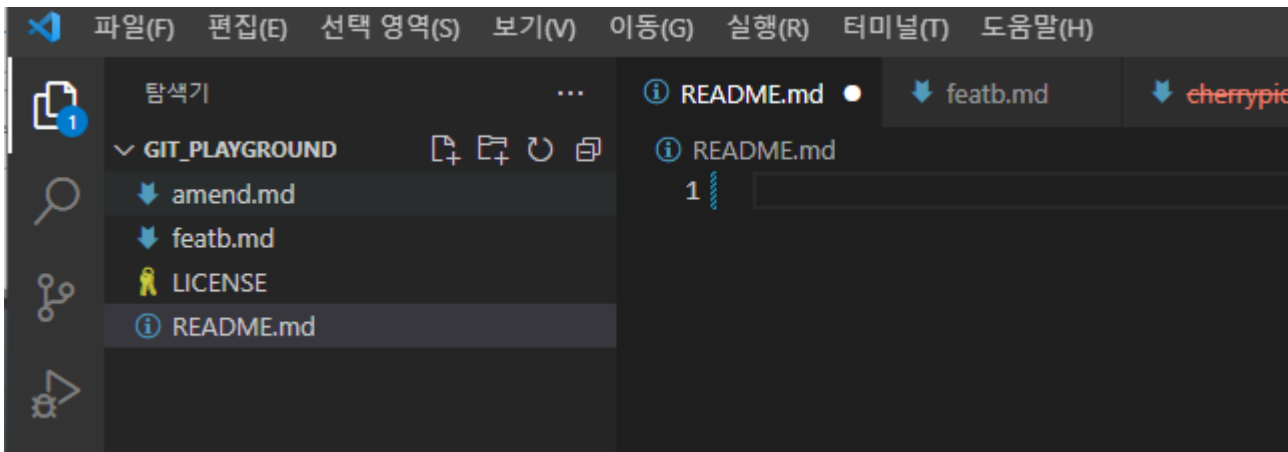
파일 내용

+ feat/b 브랜지 생성

5. revert

1. 새 커밋 생성하기

- README.md 파일을 열어서 “# git_playground” 삭제하고 저장

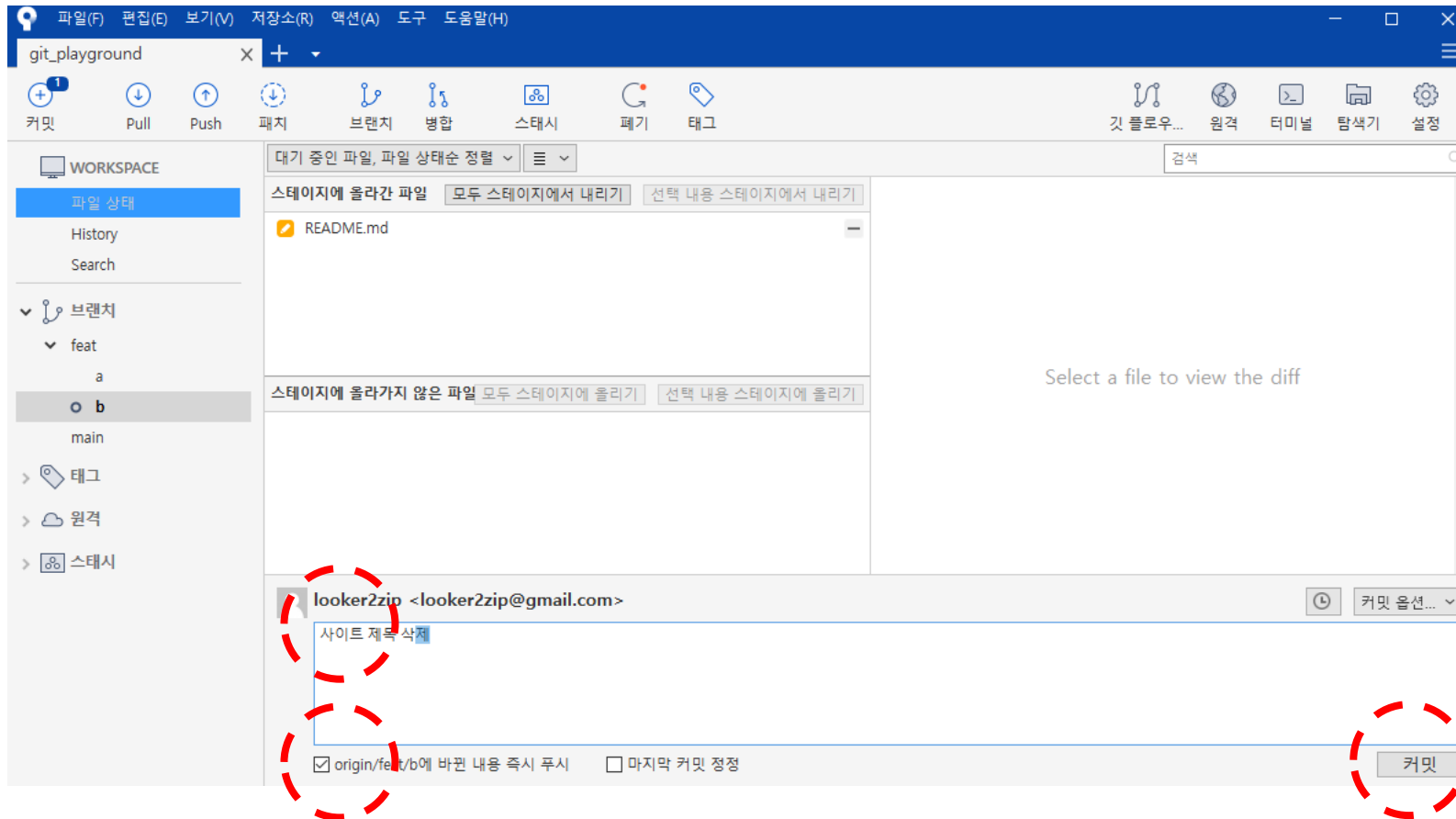




5. revert

2. 새 커밋 생성

- 커밋 생성



5. revert

3. 되돌리기

- “사이트 제목 삭제”에서 우측 버튼 클릭 후, 커밋 되돌리기

The screenshot shows the Git GUI interface. The left sidebar displays the workspace with branches 'a' and 'b'. The main area shows the commit history for 'feat/b', with the commit '사이트 제목 삭제' (Delete site title) selected. A context menu is open over this commit, with the option '커밋 되돌리기...' (Revert commit...) highlighted. Below the main area, a confirmation dialog asks: '정말 커밋을 되돌리시겠습니까?' (Are you sure you want to create a new commit reversing all the changes in the selected commit?). The dialog has two buttons: '예' (Yes) and '아니오' (No).

날짜	작성자	커밋
12 5 2023 11:41	looker2zip <looker2zip@gmail.com>	ed6dd41
12 5 2023 10:59	looker2zip <looker2zip@gmail.com>	fa421eb
12 5 2023 10:51	looker2zip <looker2zip@gmail.com>	7c453da
12 5 2023 10:46	looker2zip <looker2zip@gmail.com>	d1bc977
12 5 2023 10:36	looker2zip <looker2zip@gmail.com>	79c5f64
11 5 2023 22:16	looker2zip <looker2zip@gmail.com>	95092f6
11 5 2023 17:54	looker2zip <looker2zip@gmail.com>	b0c693e

5. revert

3. 되돌리기

- 새 커밋이 만들어졌다.

The screenshot shows the Git GUI interface with the following components:

- Toolbar:** Includes buttons for Commit, Pull, Push, Patch, Branch, Merge, Stash, Revert, and Tag.
- Workspace:** Shows the current branch 'b' and its parent 'a'.
- Commit History:** A list of commits with columns for commit hash, date, author, and message. The selected commit is 'feat/b 1 Revert "사이트 제목 삭제"'.
- File Diff:** Shows the changes in 'README.md' between the selected commit and its parent.

Commit Hash	Date	Author	Message
e35b46t	12 5 2023 11:42	looker2zip <look	Revert "사이트 제목 삭제"
ed6dd41	12 5 2023 11:41	looker2zip <look	사이트 제목 삭제
fa421eb	12 5 2023 10:59	looker2zip <look	featb.md 커밋 추가[1]
7c453da	12 5 2023 10:51	looker2zip <look	feat/a (커밋) 세 번째 커밋
d1bc977	12 5 2023 10:46	looker2zip <look	두 번째 커밋 - 체리(커밋) 선택 실습 [2]
79c5f64	12 5 2023 10:36	looker2zip <look	체리(커밋) 선택 실습 [1] 첫 번째 실습
95092f6	11 5 2023 22:16	looker2zip <look	amend 실습 파일 추가 추가 커밋 메시지 수정
b0c693e	11 5 2023 17:54	looker2zip <8011!	Initial commit

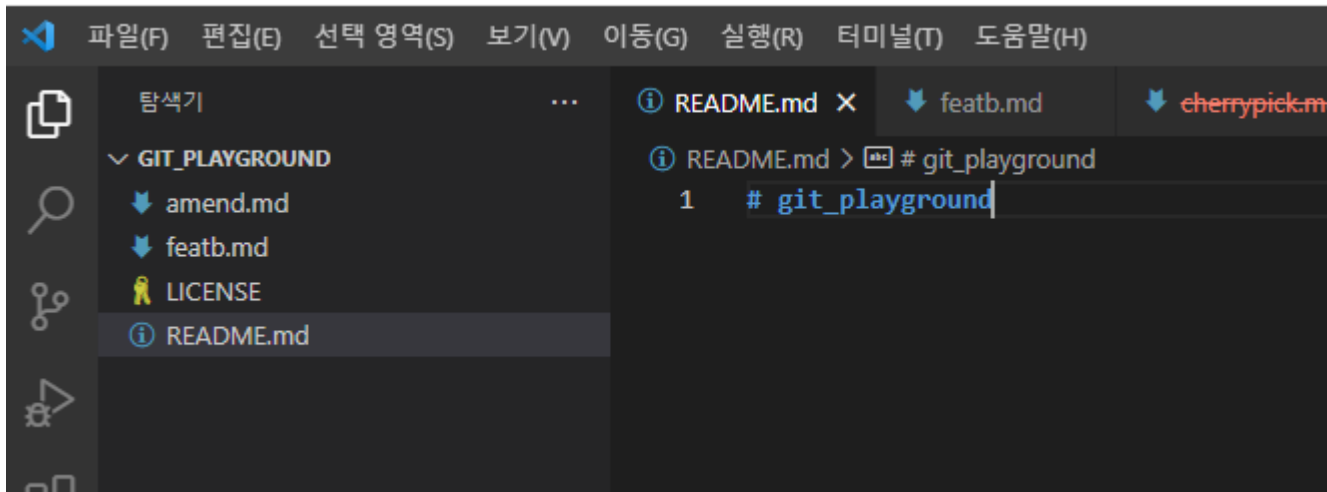
The diff for 'README.md' shows the following changes:

```
1 부분: 0-0 줄
-- # git_playground
...\No newline at end of file
```

5. revert

3. 되돌리기

- README.md 파일도 다시 되돌아감

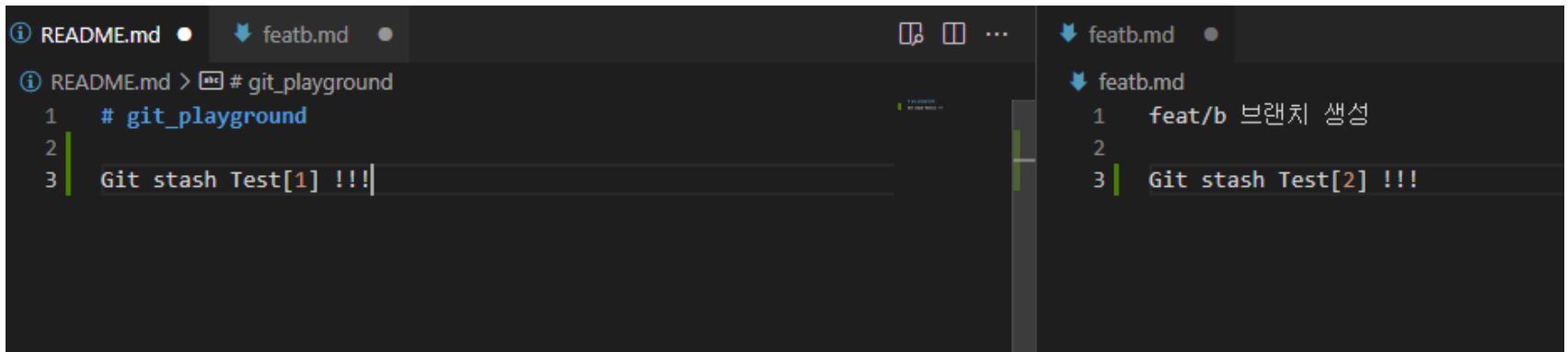


The screenshot shows a code editor interface with a dark theme. The left sidebar displays a file explorer for a project named 'GIT_PLAYGROUND'. The files listed are 'amend.md', 'featb.md', 'LICENSE', and 'README.md'. The 'README.md' file is selected and highlighted. The main editor area shows the content of 'README.md', which is a single line: '# git_playground'. The top of the editor has a menu bar with options: '파일(F)', '편집(E)', '선택 영역(S)', '보기(V)', '이동(G)', '실행(R)', '터미널(T)', and '도움말(H)'. The top of the editor also shows a tab bar with three tabs: 'README.md', 'featb.md', and 'cherrypick.m'.

6. stash

1. 변경사항 저장

- README.md 와 featb.md 내용 수정



The screenshot shows a code editor with two files open: README.md and featb.md. The README.md file contains the following text:

```
1 # git_playground
2
3 Git stash Test[1] !!!
```

The featb.md file contains the following text:

```
1 feat/b 브랜치 생성
2
3 Git stash Test[2] !!!
```

6. stash

2. stash

- 상단 메뉴 > 스테시 클릭

The screenshot shows the Git GUI interface. The 'stash' button in the top toolbar is circled in red. A dialog box is open, asking '변경점을 Stash하겠습니까?' (Do you want to stash the changes?). The dialog text says: '현재 변경 사항을 임시 저장하고 작업 중인 사항을 비웁니다. 이 스테시에 메시지를 입력하세요:' (Save current changes temporarily and clear working changes. Enter a message for this stash:). Below the text is a text input field containing '1'. There is an unchecked checkbox labeled '스테이지에 있는 변경사항 유지' (Keep changes in stage). At the bottom of the dialog are '확인' (OK) and '취소' (Cancel) buttons.

Below the dialog, the '스테이지에 올라가지 않은 파일 모두 스테이지에 올리기' (Add all uncommitted files to stage) button is visible. A list of files to be staged is shown:

- featb.md
- README.md

On the right side of the interface, a commit log table is visible:

날짜	작성자	커밋
12 5 2023 12:28	*	*
12 5 2023 11:42	looker2zip <look	e35b46t
12 5 2023 11:41	looker2zip <look	ed6dd41
12 5 2023 10:59	looker2zip <look	fa421eb
12 5 2023 10:51	looker2zip <look	7c453da
12 5 2023 10:46	looker2zip <look	d1bc977
12 5 2023 10:36	looker2zip <look	79c5f64
11 5 2023 22:16	looker2zip <look	95092f6
11 5 2023 17:54	looker2zip <look	b0c602a

Select a file to view the diff

6. stash

2. stash

- 상단 메뉴 > 스테시 클릭

The screenshot shows the Git GUI application window. The left sidebar contains a 'stash' icon at the bottom, which is circled in red. The main area displays the commit history and the current file state.

Commit History Table:

그래프	설명	날짜	작성자	커밋
feat/b	Revert "사이트 제목 삭제"	12 5 2023 11:42	looker2zip <look	e35b46b
origin/feat/b	사이트 제목 삭제	12 5 2023 11:41	looker2zip <look	ed6dd41
featb.md 커밋 추가[1]		12 5 2023 10:59	looker2zip <look	fa421eb
origin/feat/a	feat/a	12 5 2023 10:51	looker2zip <look	7c453da
feat/a	체리(커밋) 세번째 커밋	12 5 2023 10:46	looker2zip <look	d1bc977
	두 번째 커밋 - 체리(커밋) 선택 실습 [2]	12 5 2023 10:36	looker2zip <look	79c5f64
	체리(커밋) 선택 실습 [1] 첫번째 실습	11 5 2023 22:16	looker2zip <look	95092f6
origin/main	main	11 5 2023 17:54	looker2zip <8011!	b0c693e
origin/HEAD	amend 실습 파일 추가 추가 커밋 메시지 수정			
	Initial commit			

File State Section:

커밋: e35b46b091070fff331a60a32f152dd87faed563 [e35b46b]
상위 항목: ed6dd41f27
작성자: looker2zip <looker2zip@gmail.com>
날짜: 2023년 5월 12일 금요일 오전 11:42:54
커밋한 사람: looker2zip

Revert "사이트 제목 삭제"

This reverts commit ed6dd41f273fadca5787f51caf232645f2b9ccfc.

README.md File Content:

```
1 부분 : 0-0 줄  
0  
+ # git_playground  
+ \..No newline at end of file
```

6. stash

2. stash

- 스테이시 적용 후 삭제

The screenshot shows the Visual Studio Code interface with a Git repository named 'git_playground'. The left sidebar shows the 'Workspace' and 'Git Explorer' views. The 'Git Explorer' shows a branch named 'feat' with a commit 'a' and a stash entry '0:On feat/b: 1'. The main editor shows the 'README.md' file with a diff view. A dialog box is open in the center, asking '스테이시를 적용하시겠습니까?' (Do you want to apply the stash?). The dialog includes a warning icon and a message: '스테이시 '0:On feat/b: 1'을 현재 작업 디렉터리에 적용 하시겠습니까?' (Do you want to apply the stash '0:On feat/b: 1' to the current working directory?). There are two buttons: '확인' (Yes) and '취소' (No). A checkbox labeled '적용 후 삭제' (Delete after apply) is checked. A tooltip is visible over the '0:On feat/b: 1' stash entry, showing options: '0:On feat/b: 1' 스테이시 적용 (Apply) and '0:On feat/b: 1' 스테이시 삭제 (Delete).

6. stash

2. stash

- 스테시에 저장된 변경사항이 다시 작업 공간으로 나와 있는 것을 확인할 수 있다.

The screenshot shows the Git GUI interface. The left sidebar displays the workspace structure with branches 'feat', 'a', 'b', and 'main'. The main panel shows a commit history table with columns for commit hash, date, author, and message. The selected commit is 'Revert "사이트 제목 삭제"' with hash 'e35b46t'. Below the table, the '스테이지에 올라간 파일' (Staged files) section is visible, showing 'featb.md' and 'README.md'.

커밋	날짜	작성자	설명
e35b46t	12 5 2023 12:54	* <looker2zip	커밋하지 않은 변경사항
ed6dd41	12 5 2023 11:42	<looker2zip	Revert "사이트 제목 삭제"
fa421eb	12 5 2023 11:41	<looker2zip	사이트 제목 삭제
7c453da	12 5 2023 10:59	<looker2zip	featb.md 커밋 추가[1]
d1bc977	12 5 2023 10:51	<looker2zip	체리(커밋) 세번째 커밋
79c5f64	12 5 2023 10:46	<looker2zip	두 번째 커밋 - 체리(커밋) 선택 실습 [2]
95092f6	12 5 2023 10:36	<looker2zip	체리(커밋) 선택 실습 [1] 첫번째 실습
b0c502a	11 5 2023 22:16	<looker2zip	amend 실습 파일 추가 추가 커밋 메시지 수정

스테이지에 올라간 파일: featb.md, README.md

Select a file to view the diff