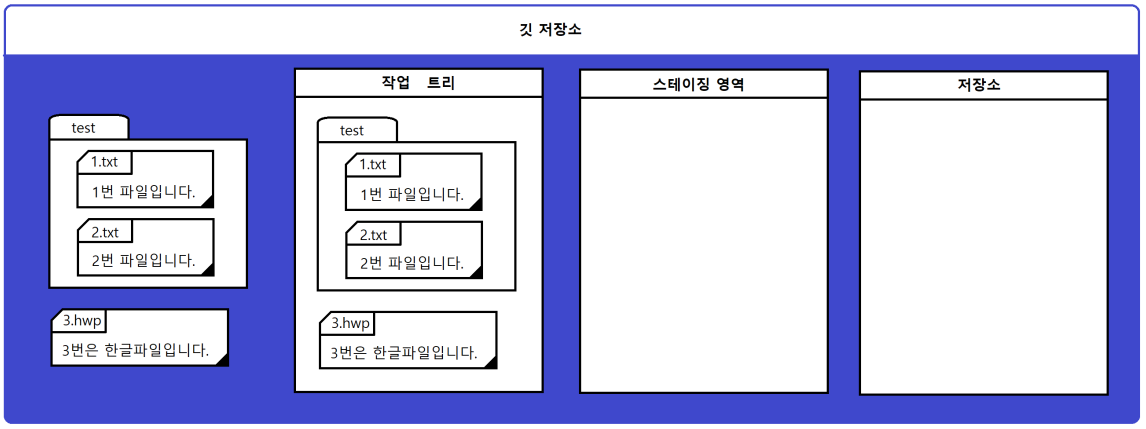
3-1. git 작업예시

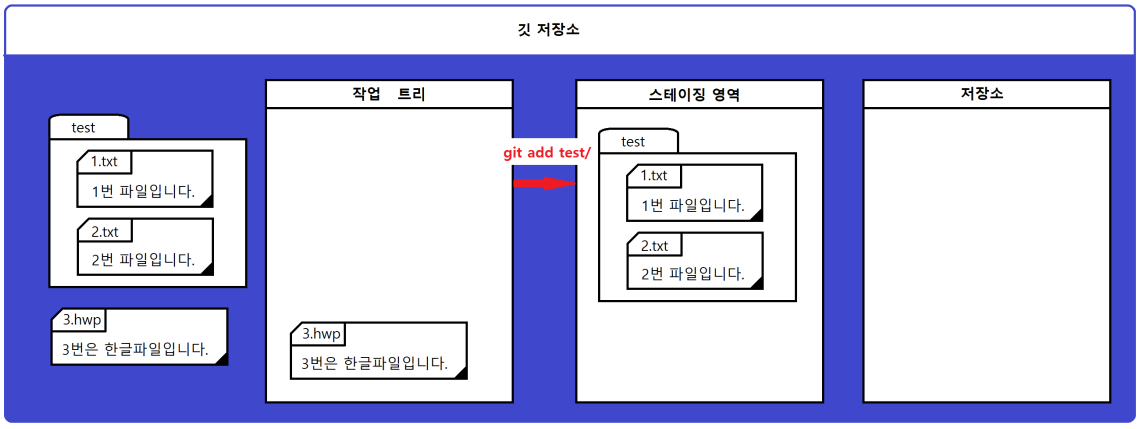
▼ 작업트리 영역에서 작업한 경우 (깃저장소에서 파일 생성만 한 경우)




```
>>> git 저장소 상태 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp
        test/

nothing added to commit but untracked files present (use "

----------------------------------------------------------

>>> git 저장소 커밋 이력 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git log
fatal: your current branch 'main' does not have any commit
```

```
----------------------------------------------------------------

>>> git 저장소 데이터 변경 내용 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git diff
***커밋이 한 번도 안되고 변경 이력이 없는 경우 아무런 내용도 나타나지 않
```

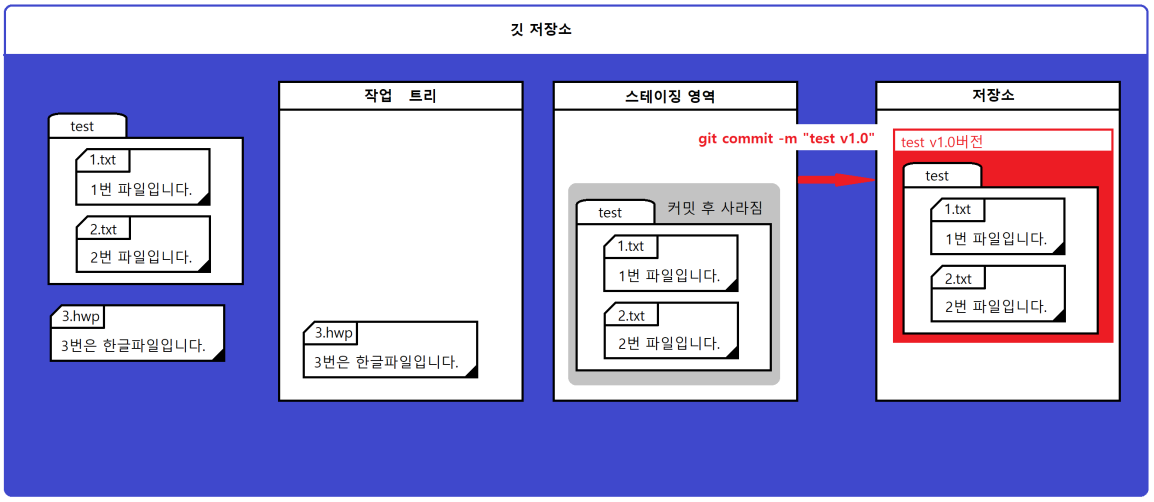
▼ git add test/ 작업 후

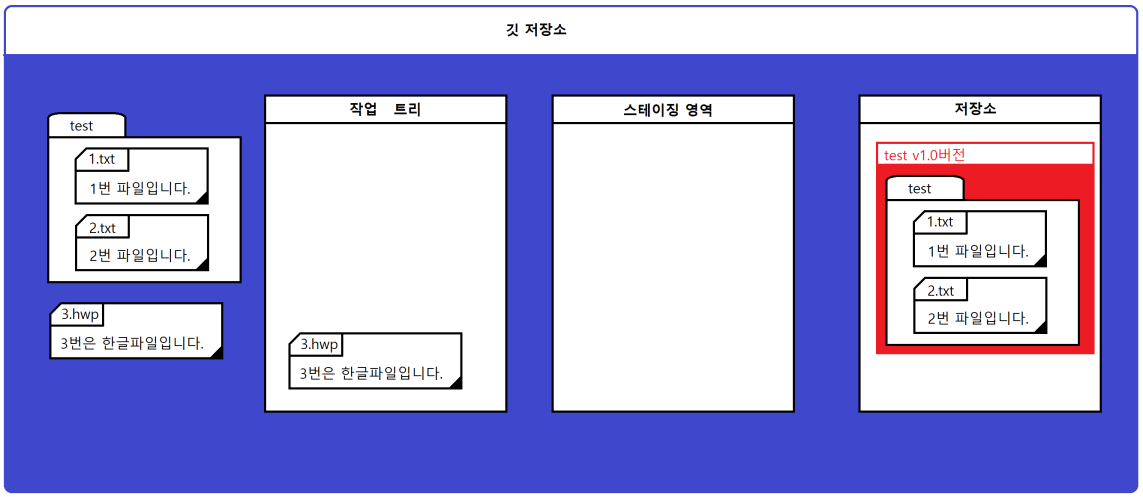



```
>>> git 저장소 상태 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test/1.txt
        new file:   test/2.txt

Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp
```

```
----------------------------------------------------------------

>>> git 저장소 커밋 이력 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git log
fatal: your current branch 'main' does not have any commit

----------------------------------------------------------------

>>> git 저장소 데이터 변경 내용 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git diff
***커밋이 한 번도 안되고 변경 이력이 없는 경우 아무런 내용도 나타나지 않
```

▼ git commit 작업 후

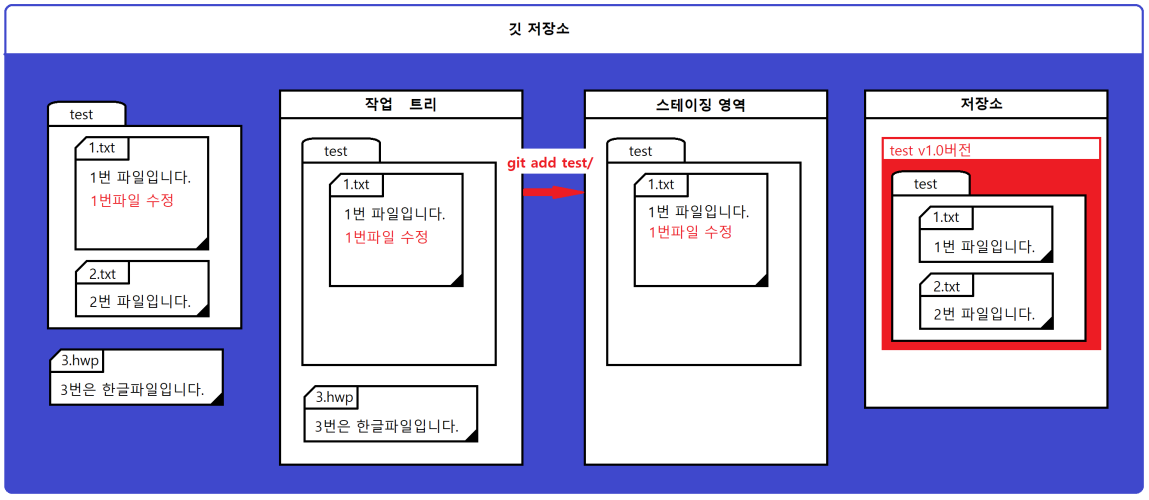





```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git commit -m "test v1.0"
[main (root-commit) 70d265a] test v1.0
 2 files changed, 3 insertions(+)
 create mode 100644 test/1.txt
 create mode 100644 test/2.txt



>>> git 저장소 상태 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp

nothing added to commit but untracked files present (use "

------------------------------------------------------------


>>> git 저장소 커밋 이력 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git log
commit 70d265a130766f9f88e7b6173dc402ec33fee477 (HEAD -> m
Author: YoungMi <ikarosala@gmail.com>
```

```
Date:   Thu Oct 24 13:24:35 2024 +0900

    test v1.0

--------------------------------------------------------------------

>>> git 저장소 데이터 변경 내용 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git diff
***커밋이 처리 후 변경 이력이 없는 경우 아무런 내용도 나타나지 않음***
```

▼ 커밋한 파일 중 test/1.txt파일만 수정하여 스테이징 영역으로 보내기

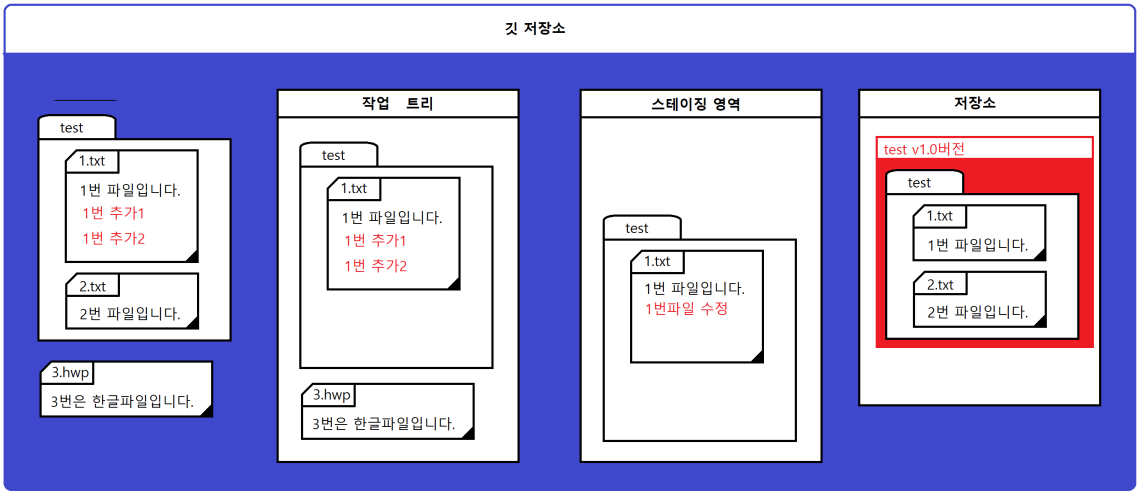



```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ vim test/1.txt

/* "1번파일 수정"이라는 내용 추가하기 */

--------------------------------------------------------------------

>>> 수정한 파일 내용 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ cat test/1.txt
```

```
1번 파일입니다.
1번파일 수정

------------------------------------------------------------

>>> git 저장소 상태 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committe
  (use "git restore <file>..." to discard changes in worki
        modified:   test/1.txt

Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp

no changes added to commit (use "git add" and/or "git comm

------------------------------------------------------------

>>> git 저장소 데이터 변경 내용 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git diff
diff --git a/test/1.txt b/test/1.txt
index a629700..660c7ca 100644
--- a/test/1.txt
+++ b/test/1.txt
@@ -1 +1,3 @@
 1번 파일입니다.     /* 기존과 변경없음 */
+1번파일 수정        /* [1번파일 수정] 이라는 문장이 추가되었다는 의미
+                    /* [ 한 줄 바꿈 ] 이 추가되었다는 의미  */


------------------------------------------------------------

>>> 수정한 파일 스테이징영역으로 보내기
```

```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git add test/

------------------------------------------------------------

>>> git 저장소 상태 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test/1.txt

Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp
```

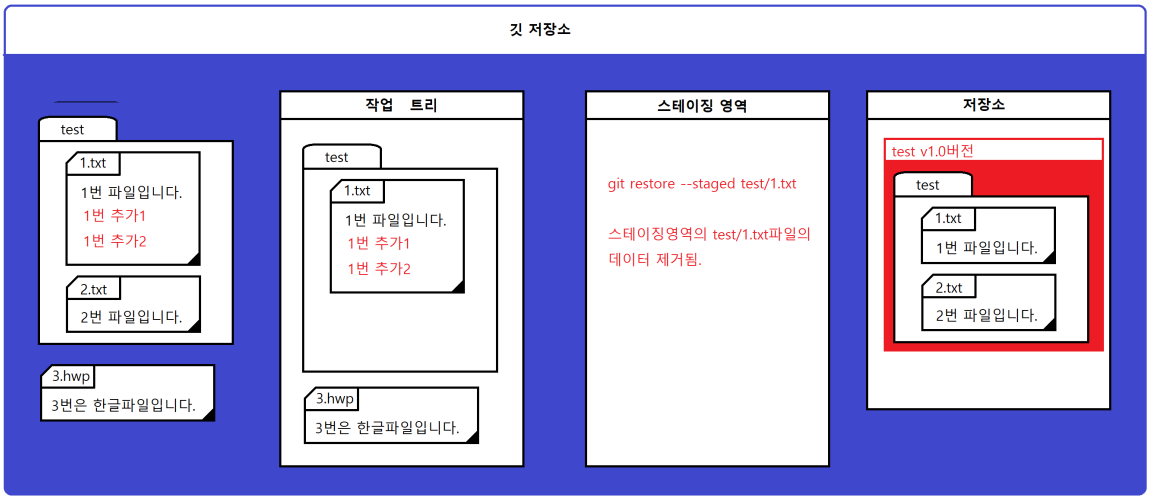
▼ 커밋한 파일 중 수정하여 스테이징 영역으로 보내놓은 test/1.txt파일만 다시 수정하기




```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ vim test/1.txt

/* "1번파일 수정"이라는 내용 삭제하기 */
```

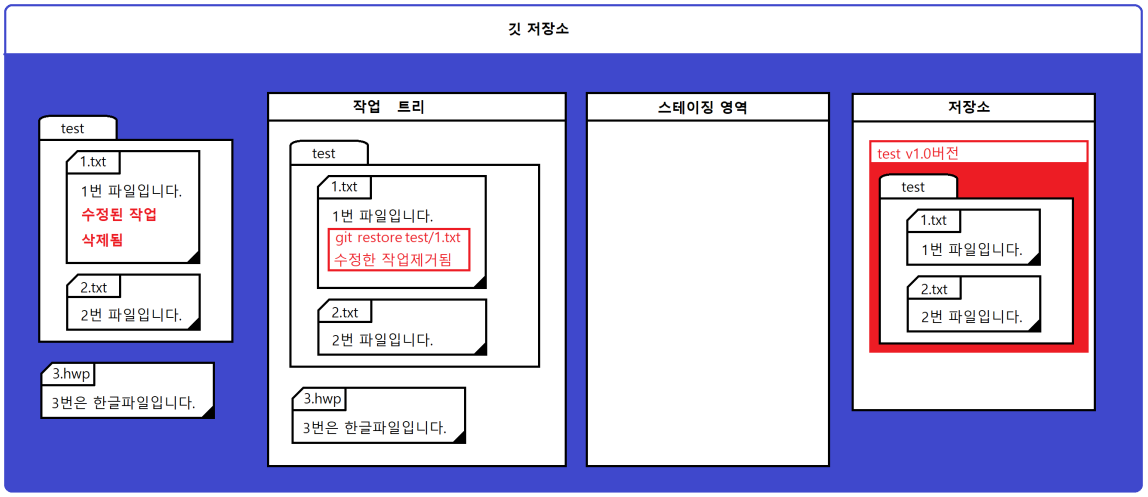
```
/* "1번 추가1"이라는 내용 추가하기 */
/* "1번 추가2"이라는 내용 추가하기 */

------------------------------------------------------------

>>> git 저장소 상태 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test/1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committe
  (use "git restore <file>..." to discard changes in worki
        modified:   test/1.txt

Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp

------------------------------------------------------------

>>> git 저장소 데이터 변경 내용 확인
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git diff
diff --git a/test/1.txt b/test/1.txt
index 660c7ca..c25e718 100644
--- a/test/1.txt
+++ b/test/1.txt
@@ -1,3 +1,4 @@
 1번 파일입니다.     /* 기존과 변경없음 */
-1번파일 수정        /* [1번파일 수정] 이라는 문장이 삭제되었다는 의미
+1번 추가1           /* [1번 추가1] 이라는 문장이 추가되었다는 의미
+1번 추가2           /* [1번 추가2] 이라는 문장이 추가되었다는 의미
```

▼ 스테이징 영역의 변경된 데이터를 제거하기




```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git restore --staged test/1.txt

ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committe
  (use "git restore <file>..." to discard changes in worki
        modified:   test/1.txt

Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp

no changes added to commit (use "git add" and/or "git comm
```

▼ 작업트리 영역에 수정된 데이터를 제거하기




```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git restore test/1.txt

ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp

nothing added to commit but untracked files present (use "

>>> test/1.txt파일 내용보기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ cat test/1.txt
1번 파일입니다.
```

```
>>> vim으로 test/1.txt파일 수정하기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ vim test/1.txt  /* 1번 파일 추가 수정  데이터 추가함 */

-----------------------------------------------------------------
>>> 3.hwp 파일과 test/1.txt 파일을 스테이지 영역으로 보내기
```

```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git add 3.hwp
warning: in the working copy of '3.hwp', LF will be replaced

ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git add test/1.txt


***스테이징 영역에 존재하는 모든 데이터 제거하기***
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git restore --staged .

***작업트리에 존재하는 모든 데이터 스테이징 영역으로 보내기***
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git add .

-----------------------------------------------------------------

>>> git 상태 확인하기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   3.hwp         /* 커밋이력이 없는 파일은 new 로
        modified:   test/1.txt    /* 커밋이력이 있는 파일은 modifi

-----------------------------------------------------------------

>>> 스테이징 역역의 데이터 모두 커밋하기(버전생성)
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git commit -m "all v1.0"
[main 7076286] all v1.0
 2 files changed, 4 insertions(+)
 create mode 100644 3.hwp

-----------------------------------------------------------------
```

```
>>> git 커밋(버전) 이력 확인하기
 ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git log
commit 70762868772383e357ac05534e1c8586d0ce85a8 (HEAD -> main
Author: YoungMi <ikarosala@gmail.com>
Date:   Thu Oct 24 14:40:31 2024 +0900

    all v1.0

commit 70d265a130766f9f88e7b6173dc402ec33fee477
Author: YoungMi <ikarosala@gmail.com>
Date:   Thu Oct 24 13:24:35 2024 +0900

    test v1.0
```

```
>>> 1) test/1.txt 파일만 수정(작업트리영역)
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ vim test/1.txt

>>> 2) 위(1번)에서 수정한 데이터를 스테이징영역으로 보내기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git add .

>>> git 상태 확인하기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test/1.txt

>>> 3) 2)의 스테이징 영역 있는 데이터 버전 생성 하기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git commit -m "test v1.1"
[main e641e00] test v1.1
```

```
 1 file changed, 1 insertion(+)

>>> git 커밋(버전) 이력 상세 확인하기  :     [--stat 옵션 ]
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git log --stat
commit e641e00d475d4efb1883127308c717465af67379 (HEAD -> main
Author: YoungMi <ikarosala@gmail.com>
Date:   Thu Oct 24 14:45:35 2024 +0900

    test v1.1

 test/1.txt | 1 +
 1 file changed, 1 insertion(+)

commit 70762868772383e357ac05534e1c8586d0ce85a8
Author: YoungMi <ikarosala@gmail.com>
Date:   Thu Oct 24 14:40:31 2024 +0900

    all v1.0

 3.hwp      | 2 ++
 test/1.txt | 2 ++
 2 files changed, 4 insertions(+)

commit 70d265a130766f9f88e7b6173dc402ec33fee477
Author: YoungMi <ikarosala@gmail.com>
Date:   Thu Oct 24 13:24:35 2024 +0900

    test v1.0

 test/1.txt | 1 +
 test/2.txt | 2 ++
 2 files changed, 3 insertions(+)
```

▼ .gitignore 파일로 버전 관리에서 제외하기

1. **vim을 이용해 .gitignore파일을 생성**
2. 파일 안에 버전관리에서 제외할 파일이나 폴더 등을 기술한다.

> 💡 예시 :
>
> **1.txt**
>
> **test/**
>
> ***.exe**

```
>>> .gitignore 파일 생성하기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ vim .gitignore

.gitignore 파일안에 작성할 내용
============================================================
*.txt
.gitignore
============================================================
```

```
>>> 작업트리 안의 데이터 스테이징으로 보내기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git add .

>>> git저장소 상태 확인하기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   3.hwp
        new file:   test/2.sql
```

```
***  .gitignore 파일안에 명시한 txt파일과 gitignore파일은 제외되고
```

```
>>> 커밋 완료 후 아무런 수정 작업도 없는 경우
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
nothing to commit, working tree clean
```

▼ 방금 커밋한(버전생성한) 데이터의 버전명 변경하기 (git commit —amend)

```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git log --stat
commit 7d8b6720eb6960555e9488b81582745dc330eaa3 (HEAD -> m
Author: youngmi <ikarosala@gmail.com>
Date:   Thu Oct 24 15:01:47 2024 +0900

    all v1.0   /* 현재 버전의 버전명 */

==========================================================

>>> 버전명 변경하기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git commit --amend    /* 명령 실행하면 vim편집창 나옴. */
[main c6177d8] all v0.1
 Date: Thu Oct 24 15:01:47 2024 +0900
 2 files changed, 4 insertions(+)
 create mode 100644 3.hwp
 create mode 100644 test/2.sql


----------------------------------------------------------
>>> vim 편집기로 편집가능하도록 내용이 나옴.(실행모드 상태로 편집하려면
----------------------------------------------------------
all v0.1      /*이 부분이 버전명으로 변경처리 한 후 ESC키(실행모드로
```

```
# Please enter the commit message for your changes. Lines
# with '#' will be ignored, and an empty message aborts th
#
# Date:      Thu Oct 24 15:01:47 2024 +0900
#
# On branch main
#
# Initial commit
#
# Changes to be committed:
#       new file:   3.hwp
#       new file:   test/2.sql
#
------------------------------------------------------------
============================================================
```

```
>>> git 커밋 이력 확인하기
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git log
commit c6177d82e6e5776599653c295d36f06efb80abaa (HEAD -> m
Author: youngmi <ikarosala@gmail.com>
Date:   Thu Oct 24 15:01:47 2024 +0900

    all v0.1
```