GIT CLI

GIT CLI

- 버전 만들기
  - ✓ git init
    - 로컬 저장소를 만드는 명령
    - 디렉토리를 생성하고 디렉토리로 이동한 후 git init

GIT CLI

- 버전 만들기
  - ✓ git status
    - 작업 디렉토리 상태를 확인하는 명령
    - 저장소에 A가 적힌 a.txt 파일을 생성
    - git status

      On branch main #현재 브랜치 이름

      No commits yet #커밋을 하지 않음

      nothing to commit (create/copy files and use "git add" to track) #아직 커밋한 파일이 없다는 의미

GIT CLI

- ❖ 버전 만들기
  - ✓ git add
    - 스테이지에 추가하는 명령으로 git add <스테이지에 추가할 대상>
    - git add a.txt
    - git status

      On branch main

      No commits yet

      Changes to be committed:
      (use "git rm --cached <file>..." to unstage)
      new file: a.txt
    - 현재 디렉토리의 모든 내용을 스테이지에 추가할 때는 git add .

GIT CLI

- ❖ 버전 만들기
  - ✓ git commit
    - ● 커밋을 할 때는 git commit --message(-m) "커밋 메시지"
    - ● git commit -m "first commit"

```
[main (root-commit) 0a71d36] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 a.txt
```

GIT CLI

- ❖ 버전 만들기
  - ✓ git log
    - 커밋 목록을 출력하는 명령으로 커밋 해시, 만든 사람, 커밋이 만들어진 날짜, 커밋 메시지가 출력됨
    - git log

```
commit 0a71d367a0d430a4f16d6ed7b515426d7b01bdab (HEAD -> main)
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:15:16 2026 +0900

    first commit
```

GIT CLI

- ❖ 버전 만들기
  - ✓ 스테이지에 올리는 것 과 커밋을 동시에 수행
    - git commit -am “커밋 메시지”
    - 스테이지에 이미 올라와 있거나 한 번이라도 커밋한 적이 있는 파일에만 사용 가능
    - a.txt 파일에 B를 추가
    - git status

      On branch main

      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: a.txt

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

      [main 6be5d90] second commit

       1 file changed, 2 insertions(+), 1 deletion(-)")

GIT CLI

- ❖ 버전 만들기
  - ✓ 스테이지에 올리는 것 과 커밋을 동시에 수행
    - git log

```
commit 6be5d909f17c09da7feb795e377c74e655cd2076 (HEAD -> main)
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:19:41 2026 +0900

    second commit

commit 0a71d367a0d430a4f16d6ed7b515426d7b01bdab
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:15:16 2026 +0900

    first commit
```

GIT CLI

- ❖ 버전 만들기
  - ✓ 커밋 제목과 메시지를 같이 입력
    - git commit 명령만 수행한 후 a 또는 i를 선택해서 입력
    - a.txt 파일에 C를 추가
    - git add .
    - git commit
      - ◆ i를 입력해서 INSERT 모드로 진입

        Third commit

        This is my third commit
      - ◆ ESC를 누르고 :wq!를 눌러서 저장

      [main b489bf1] third commit

      1 file changed, 2 insertions(+), 1 deletion(-)

GIT CLI

- ❖ 버전 만들기
  - ✓ 커밋 제목과 메시지를 같이 입력
    - git log

```
commit b489bf13ca96072f2c94839eeb1cb24161c9d1d0 (HEAD -> main)
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:24:39 2026 +0900
    third commit
    This is my third commit

commit 6be5d909f17c09da7feb795e377c74e655cd2076
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:19:41 2026 +0900
    second commit

commit 0a71d367a0d430a4f16d6ed7b515426d7b01bdab
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:15:16 2026 +0900
    first commit
```

GIT CLI

- ❖ 버전 만들기
  - ✓ 커밋 조회
    - ● 커밋 목록을 단순한 형태로 출력: git log --oneline

```
b489bf1 (HEAD -> main) third commit
6be5d90 second commit
0a71d36 first commit
```

GIT CLI

- ❖ 버전 만들기
  - ✓ 커밋 조회
    - ● 수정된 내용 확인: git log --patch

```
commit b489bf13ca96072f2c94839eeb1cb24161c9d1d0 (HEAD -> main)
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:24:39 2026 +0900

    third commit

    This is my third commit

diff --git a/a.txt b/a.txt
index 2fe4ab0..870951a 100644
--- a/a.txt
+++ b/a.txt
@@ -1,2 +1,3 @@
 A
-B
\ No newline at end of file
+B
+C
\ No newline at end of file
```

GIT CLI

- ❖ 버전 만들기
  - ✓ 커밋 조회
    - ● 커밋을 그래프의 형태로 출력: git log --graph

```
* commit b489bf13ca96072f2c94839eeb1cb24161c9d1d0 (HEAD -> main)
| Author: itggangpae <itstudy@kakao.com>
| Date:   Wed Mar 11 07:24:39 2026 +0900
|
|     third commit
|
|     This is my third commit
|
* commit 6be5d909f17c09da7feb795e377c74e655cd2076
| Author: itggangpae <itstudy@kakao.com>
| Date:   Wed Mar 11 07:19:41 2026 +0900
|
|     second commit
|
* commit 0a71d367a0d430a4f16d6ed7b515426d7b01bdab
  Author: itggangpae <itstudy@kakao.com>
  Date:   Wed Mar 11 07:15:16 2026 +0900

      first commit
```

GIT CLI

- ❖ 버전 만들기
  - ✓ 커밋 조회
    - ● 모든 브랜치의 커밋 목록 조회: git log --branches

```
commit b489bf13ca96072f2c94839eeb1cb24161c9d1d0 (HEAD -> main)
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:24:39 2026 +0900

    third commit

    This is my third commit

commit 6be5d909f17c09da7feb795e377c74e655cd2076
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:19:41 2026 +0900

    second commit

commit 0a71d367a0d430a4f16d6ed7b515426d7b01bdab
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:15:16 2026 +0900

    first commit
```

GIT CLI

- ❖ 태그 관리
  - ✓ 태그 추가
    - 현재 커밋에 태그 추가: git tag <태그>
    - 세번째 커밋에 v1.0.0 태그 추가: git tag v1.0.0
    - git log

```
commit b489bf13ca96072f2c94839eeb1cb24161c9d1d0 (HEAD -> main, tag: v1.0.0)
Author: itggangpae <itstudy@kakao.com>
Date:   Wed Mar 11 07:24:39 2026 +0900

    third commit

    This is my third commit
```

GIT CLI

- ❖ 태그 관리
  - ✓ 태그 추가
    - 특정 커밋에 태그 추가: git tag <태그> <커밋>
    - 커밋 해시 확인: git log --oneline

      b489bf1 (HEAD -> main, tag: v1.0.0) third commit

      6be5d90 second commit

      0a71d36 first commit
    - 두번째 커밋에 v0.0.1 태그 추가: git tag v0.0.1 6be5d90
    - 커밋 해시 확인: git log --oneline

      b489bf1 (HEAD -> main, tag: v1.0.0) third commit

      6be5d90 (tag: v0.0.1) second commit

      0a71d36 first commit

GIT CLI

- ❖ 태그 관리
  - ✓ 태그 조회
    - 태그 목록 조회: git tag --list
    - git tag --list

      v0.0.1

      v1.0.0
  - ✓ 태그 삭제
    - 태그 삭제: git tag --delete <태그>
    - git tag --delete v0.0.1
    - git tag --list

      v1.0.0

GIT CLI

- ❖ 버전 비교
  - ✓ git diff: 최근 커밋 과 작업 디렉토리 비교
    - a.txt 파일에 D라는 문자를 추가하고 저장
    - git diff

```
diff --git a/a.txt b/a.txt
index 870951a..1f30c63 100644
--- a/a.txt
+++ b/a.txt
@@ -1,3 +1,4 @@
 A
 B
C
₩ No newline at end of file
+D
₩ No newline at end of file
```

GIT CLI

- 버전 비교
  - ✓ git diff --staged: 최근 커밋 과 스테이지 비교
    - git add a.txt
    - git diff --staged

```
diff --git a/a.txt b/a.txt
index 870951a..1f30c63 100644
--- a/a.txt
+++ b/a.txt
@@ -1,3 +1,4 @@
 A
 B
 C
₩ No newline at end of file
+D
₩ No newline at end of file
```

GIT CLI

- ❖ 버전 비교
  - ✓ git diff <기준 커밋><변경된 커밋>: 커밋 간의 비교
    - git commit -m "fourth commit"
    - git log --oneline

      0bde5ef (HEAD -> main) fourth commit
      b489bf1 (tag: v1.0.0) third commit
      6be5d90 second commit
      0a71d36 first commit
    - 세번째 커밋을 기준으로 네번째 커밋이 달라진 점: git diff b489bf1 0bde5ef

      diff --git a/a.txt b/a.txt
      index 870951a..1f30c63 100644
      --- a/a.txt
      +++ b/a.txt
      @@ -1,3 +1,4 @@
       A
       B
       C
      ₩ No newline at end of file
      +D
      ₩ No newline at end of file

GIT CLI

- 버전 비교
  - ✓ git diff <기준 커밋><변경된 커밋>: 커밋 간의 비교
    - HEAD를 이용해서 커밋을 작성하면 편리한데 HEAD 이전 커밋은 HEAD^ 또는 HEAD~1 이고 두 개 이전 커밋은 HEAD^^ 또는 HEAD~2
    - 세번째 커밋을 기준으로 네번째 커밋이 달라진 점: git diff b489bf1 0bde5ef

```
diff --git a/a.txt b/a.txt
index 870951a..1f30c63 100644
--- a/a.txt
+++ b/a.txt
@@ -1,3 +1,4 @@
 A
 B
 C
￦ No newline at end of file
+D
￦ No newline at end of file
```

GIT CLI

- 버전 비교
  - ✓ git diff <기준 커밋><변경된 커밋>: 커밋 간의 비교
    - 세번째 커밋을 기준으로 네번째 커밋이 달라진 점: git diff HEAD~1 HEAD

```
diff --git a/a.txt b/a.txt
index 870951a..1f30c63 100644
--- a/a.txt
+++ b/a.txt
@@ -1,3 +1,4 @@
 A
 B
 C
￦ No newline at end of file
+D
￦ No newline at end of file
```

GIT CLI

- 버전 비교
  - ✓ git diff <브랜치><브랜치>: 브랜치 간의 비교

GIT CLI

- ❖ 작업 되돌리기
  - ✓ git reset <되돌아갈 커밋>: 예전 커밋으로 되돌리기
    - ● soft reset: 커밋만 되돌리기
      - ◆ git reset --soft <되돌아갈 커밋>
        - ▪ git log --oneline

          0bde5ef (HEAD -> main) fourth commit

          b489bf1 (tag: v1.0.0) third commit

          6be5d90 second commit

          0a71d36 first commit
        - ▪ git reset --soft b489bf1
        - ▪ git log --oneline: 네번째 커밋이 사라졌지만 D를 추가한 내용은 남아있음

          b489bf1 (tag: v1.0.0) third commit

          6be5d90 second commit

          0a71d36 first commit

GIT CLI

- 작업 되돌리기
  - ✓ git reset <되돌아갈 커밋>: 예전 커밋으로 되돌리기
    - soft reset: 커밋만 되돌리기
      - ◆ git reset --soft <되돌아갈 커밋>
        - git diff --staged

```
diff --git a/a.txt b/a.txt
index 870951a..1f30c63 100644
--- a/a.txt
+++ b/a.txt
@@ -1,3 +1,4 @@
 A
 B
C
₩ No newline at end of file
+D
₩ No newline at end of file
```

        - 네번째 커밋 생성: git commit -m "fourth commit"

GIT CLI

- ❖ 작업 되돌리기
  - ✓ git reset <되돌아갈 커밋>: 예전 커밋으로 되돌리기
    - ● mixed reset: 커밋 과 stage 되돌리기
      - ◆ git reset <되돌아갈 커밋>
        - ▪ git reset HEAD^
        - ▪ git log --oneline

          b489bf1 (HEAD -> main, tag: v1.0.0) third commit

          6be5d90 second commit

          0a71d36 first commit

GIT CLI

- ❖ 작업 되돌리기
  - ✓ git reset <되돌아갈 커밋>: 예전 커밋으로 되돌리기
    - ● mixed reset: 커밋 과 stage 되돌리기
      - ◆ git reset <되돌아갈 커밋>
        - ▪ git diff --staged
        - ▪ git status

          On branch main

          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: a.txt

          no changes added to commit (use "git add" and/or "git commit -a")
        - ▪ 네번째 커밋 생성: git commit -m "fourth commit"

GIT CLI

- 작업 되돌리기
  - ✓ git reset <되돌아갈 커밋>: 예전 커밋으로 되돌리기
    - hard reset: 커밋 과 stage 및 변경사항 되돌리기
      - git reset --hard <되돌아갈 커밋>
        - git reset --hard HEAD^
        - 소스 코드를 확인해보면 변경 내용도 사라짐

GIT CLI

- 작업 되돌리기
  - ✓ git revert <취소할 커밋>: 예전 커밋으로 되돌리기
    - reset이 특정 커밋으로 되돌아가는 방식이라면 revert는 해당 커밋을 취소한 새로운 커밋을 추가하는 방식
    - git revert < 취소할 커밋> 명령을 사용
    - git reset은 뒤에 되돌아갈 커밋을 명시하는 반면 git revert는 뒤에 취소할 커밋을 명
    - git reset < 두 번째 커밋>은 두 번째 커밋으로 돌아가는 명령이지만 git revert < 두 번째 커밋>은 두 번째 커밋을 취소한 새로운 커밋을 추가하는 명령
    - 현재 커밋 확인: git log --oneline

      3a0e52f (HEAD -> main) third commit

      6be5d90 second commit

      0a71d36 first commit
    - 세번째 커밋을 되돌린 새로운 커밋 생성: git revert 3a0e52f 명령 수행 후 커밋 메시지 작성

      Revert "third commit"

      This reverts commit 3a0e52f2ad75833d711d06a66b0443577201f278.

GIT CLI

- 작업 되돌리기
  - ✓ git revert <취소할 커밋>: 예전 커밋으로 되돌리기
    - reset이 특정 커밋으로 되돌아가는 방식이라면 revert는 해당 커밋을 취소한 새로운 커밋을 추가하는 방식
    - git revert < 취소할 커밋> 명령을 사용
    - git reset은 뒤에 되돌아갈 커밋을 명시하는 반면 git revert는 뒤에 취소할 커밋을 명
    - git reset < 두 번째 커밋>은 두 번째 커밋으로 돌아가는 명령이지만 git revert < 두 번째 커밋>은 두 번째 커밋을 취소한 새로운 커밋을 추가하는 명령
    - 현재 커밋 확인: git log --oneline

      3a0e52f (HEAD -> main) third commit

      6be5d90 second commit

      0a71d36 first commit
    - 세번째 커밋을 되돌린 새로운 커밋 생성: git revert 3a0e52f 명령 수행 후 커밋 메시지 작성

      Revert "third commit"

      This reverts commit 3a0e52f2ad75833d711d06a66b0443577201f278.

GIT CLI

- ❖ 작업 되돌리기
  - ✓ git revert <취소할 커밋>: 예전 커밋으로 되돌리기
    - ● 이전 내용과 비교: git diff HEAD^ HEAD

```
diff --git a/a.txt b/a.txt
index 870951a..2fe4ab0 100644
--- a/a.txt
+++ b/a.txt
@@ -1,3 +1,2 @@
 A
-B
-C
₩ No newline at end of file
+B
₩ No newline at end of file
```

GIT CLI

- ❖ 작업 임시 저장
  - ✓ git stash: 변경 사항 임시 저장
    - tracked 상태의 파일에 대해서만 사용 가능
    - git stash 명령을 입력해도 작업이 임시 저장되지만 git stash -m "메시지" 또는 git stash --message "〈메시지〉" 명령으로 간단한 메시지와 함께 임시 저장할 수 있음
    - git hard <첫번째 커밋> 명령을 이용해서 첫번째 커밋으로 이동
    - a.txt 파일에 B를 추가하고 저장한 후 git stash -m "add B"

      Saved working directory and index state On main: add B
    - a.txt 파일을 확인해보면 입력한 내용이 없어짐
    - a.txt 파일에 C를 추가하고 저장한 후 git stash -m "add C"

      Saved working directory and index state On main: add C
    - a.txt 파일을 확인해보면 입력한 내용이 없어짐

GIT CLI

- ❖ 작업 임시 저장
  - ✓ git stash list: 임시 저장된 작업 내역 조회
    - git stash list

      stash@{0}: On main: add C

      stash@{1}: On main: add B

GIT CLI

- ❖ 작업 임시 저장
  - ✓ git stash apply <스태시>: 임시 저장된 작업 적용
    - ● git stash apply stash@{0}

```
On branch main
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:   a.txt

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

GIT CLI

- ❖ 작업 임시 저장
  - ✓ git stash drop <스태시>: 임시 저장된 작업 삭제
    - ● git stash drop stash@{0}

      Dropped stash@{0} (a7c3e24f373467fbb6425d3311d9f35dd3822ec5)

GIT CLI

- ❖ 브랜치 관리
  - ✓ 실습 환경 구축
    - b.txt 파일을 만들고 B를 적은 뒤 커밋 메시지를 second commit 으로 커밋
    - c.txt 파일을 만들고 C를 적은 뒤 커밋 메시지를 third commit 으로 커밋
    - 브랜치 확인: git branch

    *main

GIT CLI

- ❖ 브랜치 관리
  - ✓ git branch <브랜치>: 브랜치 생성
    - foo 브랜치 생성: git branch foo
    - 브랜치 확인: git branch

      foo

      * Main

GIT CLI

- 브랜치 관리
  - ✓ git checkout <브랜치>: 체크아웃
    - foo 브랜치로 체크아웃: git checkout foo
    - 문자 D가 저장된 foo_d.txt 파일을 생성하고 fourth commit 메시지로 해서 commit
    - 문자 E가 저장된 foo_e.txt 파일을 생성하고 fifth commit 메시지로 해서 commit
    - main 브랜치로 체크아웃: git checkout main
    - main 과 foo 브랜치 비교: git diff main foo

```
diff --git a/a.txt b/a.txt
index 870951a..e0304e4 100644
--- a/a.txt
+++ b/a.txt
@@ -1,3 +1,5 @@
 A
 B
-C
₩ No newline at end of file
+C
+D
+E
₩ No newline at end of file
```

GIT CLI

- ❖ 브랜치 관리
  - ✓ git checkout <브랜치>: 체크아웃
    - ● 브랜치를 만듦과 동시에 체크아웃: git checkout –b 브랜치
    - ● bar라는 브랜치를 만듬과 동시에 체크아웃: git checkout -b bar

GIT CLI

- ❖ 브랜치 관리
  - ✓ git merge <브랜치>: 브랜치 병합
    - ● main 브랜치에 foo 브랜치 병합
      - ◆ main 브랜치로 체크아웃: git checkout main
      - ◆ 병합: git merge foo

```
Updating a797c80..7884754
Fast-forward
 a.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

GIT CLI

- ❖ 브랜치 관리
  - ✓ 충돌 해결
    - ● 충돌 상황 생성
      - ◆ main 브랜치에서 a.txt 파일을 main으로 수정하고 main commit 메시지를 이용해서 commit
      - ◆ foo 브랜치에서 a.txt 파일을 foo로 수정하고 foo commit 메시지를 이용해서 commit
      - ◆ main 브랜치로 체크아웃
      - ◆ foo 브랜치 병합

a.txt

현재 변경 사항 수락 | 수신 변경 사항 수락 | 두 변경 사항 모두 수락 | 변경 사항 비교

```
<<<<<<< HEAD (현재 변경 사항)
main
=======
foo
>>>>>>> foo (수신 변경 사항)
```

GIT CLI

- ❖ 브랜치 관리
  - ✓ 충돌 해결
    - ● 충돌 상황 해결
      - ◆ a.txt 파일의 내용을 main 으로 수정
      - ◆ a.txt 파일을 스테이지에 추가: git add a.txt
      - ◆ git commit 명령을 수행하고 메시지를 작성하고 저장

GIT CLI

- ❖ 브랜치 관리
  - ✓ 브랜치 삭제: git branch -d <브랜치>
    - foo 브랜치 삭제: git branch –d foo
  - ✓ 브랜치 재배치: git rebase <브랜치>
    - main 브랜치에서 foo 브랜치 생성: git branch foo
    - foo 브랜치로 체크아웃: git checkout foo
    - foo 브랜치에 커밋을 추가
      - ◆ foo_a.txt 파일을 만들고 A를 작성한 후 Commit
      - ◆ foo_b.txt 파일을 만들고 B를 작성한 후 Commit
    - 현재까지의 commit 확인: git log --oneline

      89404d4 (HEAD -> foo) add B

      9a0e4f4 add A

      a797c80 third commit

      7144b10 second commit

      0a71d36 first commit

GIT CLI

- ❖ 브랜치 관리
  - ✓ 브랜치 재배치: git rebase <브랜치>
    - main 브랜치로 체크아웃: git checkout main
    - main 브랜치에 커밋을 추가
      - ◆ d.txt 파일을 만들고 D를 작성한 후 Commit
      - ◆ e.txt 파일을 만들고 E를 작성한 후 Commit
    - 모든 브랜치의 commit 확인: git log --oneline --branches

```
d3356ad (HEAD -> main) add e
3751d4f add d
89404d4 (foo) add B
9a0e4f4 add A
5b5aebf (foo) Merge branch 'foo'
3b231c8 foo commit
ef3aac3 main commmit
7884754 fifth commit
b9e8b3b fourth commit
a797c80 third commit
7144b10 second commit
0a71d36 first commit
```

GIT CLI

- ❖ 브랜치 관리
  - ✓ 브랜치 재배치: git rebase <브랜치>
    - foo 브랜치로 체크아웃: git checkout foo
    - foo 브랜치를 main 브랜치로 재배치: git rebase main
    - 모든 브랜치의 commit 확인: git log --oneline --branches

```
0b59e8f (HEAD -> foo) add B
e2e0f57 add A
d3356ad (main) add e
3751d4f add d
5b5aebf (foo) Merge branch 'foo'
3b231c8 foo commit
ef3aac3 main commmit
7884754 fifth commit
b9e8b3b fourth commit
a797c80 third commit
7144b10 second commit
0a71d36 first commit
```