

2025년 서울 민간기업 맞춤형 매력일자리 사업

# GIT 활용법

- 2025.05.02 -

# GIT



# GIT 사용법

- 기본 -

# GIT정리

## 키 발급

키

```
$ git clone 'https://아이디:키체인@github.com/아이디/프로젝트.git'
```

```
$ git config --global user.name "본인이름"
```

```
$ git config --global user.email "본인전자메일"
```

```
$ git init
```

```
$ git remote add origin "경로"
```

```
$ git branch -M main
```

```
$ git pull origin main
```

## 저장소(Git repository) 파일이나 폴더를 저장하는 장소



# 저장소의 종류

## ① 원격저장소

파일이 원격 저장소 전용 서버에서 관리되며  
여러 사람이 함께 공유하기 위한 저장소

## ② 로컬저장소

내 PC에 파일이 저장되는 개인 전용 저장소

## SourceTree

<https://www.sourcetreeapp.com>

## Git-scm

<https://git-scm.com>

# GIT 설정

## 콘솔 설정(.gitconfig에 기록)

```
$ git clone 레파지토리 복사
```

```
$ git config --global user.name "사용자id"
```

```
$ git config --global user.email "사용자메일주소"
```

## Window에서 한글깨짐 해결

```
$ git config --global core.quotePath off
```



## GIT 로컬저장소 사용하기(내 컴퓨터)

## 콘솔

```
$ git init
```

Initialized empty Git repository in /Users/imac.ai/ai\_exam/flutter/git\_test/.git/

```
(base) imac.ai ~/ai_exam/flutter/git_test
● (base) imac.ai ~/ai_exam/flutter/git_test git init
  Initialized empty Git repository in /Users/imac.ai/ai_exam/flutter/git_test/.git/
● (base) imac.ai ~/ai_exam/flutter/git_test main git status
```

## Git 상태확인(git status)

Git의 관리하의 폴더의 작업트리 인덱스 상태확인

\$ git status

```
nothing to commit (create/copy files and use "git add" to track)
● (base) imac.ai ~/ai_exam/flutter/git_test [main] git status
On branch main

No commits yet

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

## Git 인덱스 등록(`git add .`)

Git에 작업된 모든 파일을 등록한다.

```
$ git add .
```

```
nothing added to commit but untracked files present (use "git add" to t
● (base) imac.ai ~/ai_exam/flutter/git_test □ main → git add .
● (base) imac.ai ~/ai_exam/flutter/git_test □ main + → git status
On branch main

No commits yet

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

○ (base) imac.ai ~/ai_exam/flutter/git_test □ main +
```

## Git 등록(git commit)

로컬 저장소에 저장(commit)한다

\$ git commit -m "2023-02-04 메인파일등록"

- (base) `imac.ai` `~/ai_exam/flutter/git_test` `main` + `git commit -m "2023-02-04 메인파일등록"`  
[main (root-commit) 897bdd1] 2023-02-04 메인파일등록  
1 file changed, 2 insertions(+)  
create mode 100644 sample.py
- (base) `imac.ai` `~/ai_exam/flutter/git_test` `main` `git status`  
On branch main  
nothing to commit, working tree clean
- (base) `imac.ai` `~/ai_exam/flutter/git_test` `main` `git log`
- (base) `imac.ai` `~/ai_exam/flutter/git_test` `main`

## Git 기록확인(git log)

기록된 정보를 확인한다.

\$ git log

```
commit 897bdd118e7eb7dc7c714a38089a1ac3726939f8 (HEAD -> main)
```

```
Author: matalcross <matalcross@naver.com>
```

```
Date: Sat Feb 4 03:19:17 2023 +0900
```

2023-02-04 메인 파일 등록

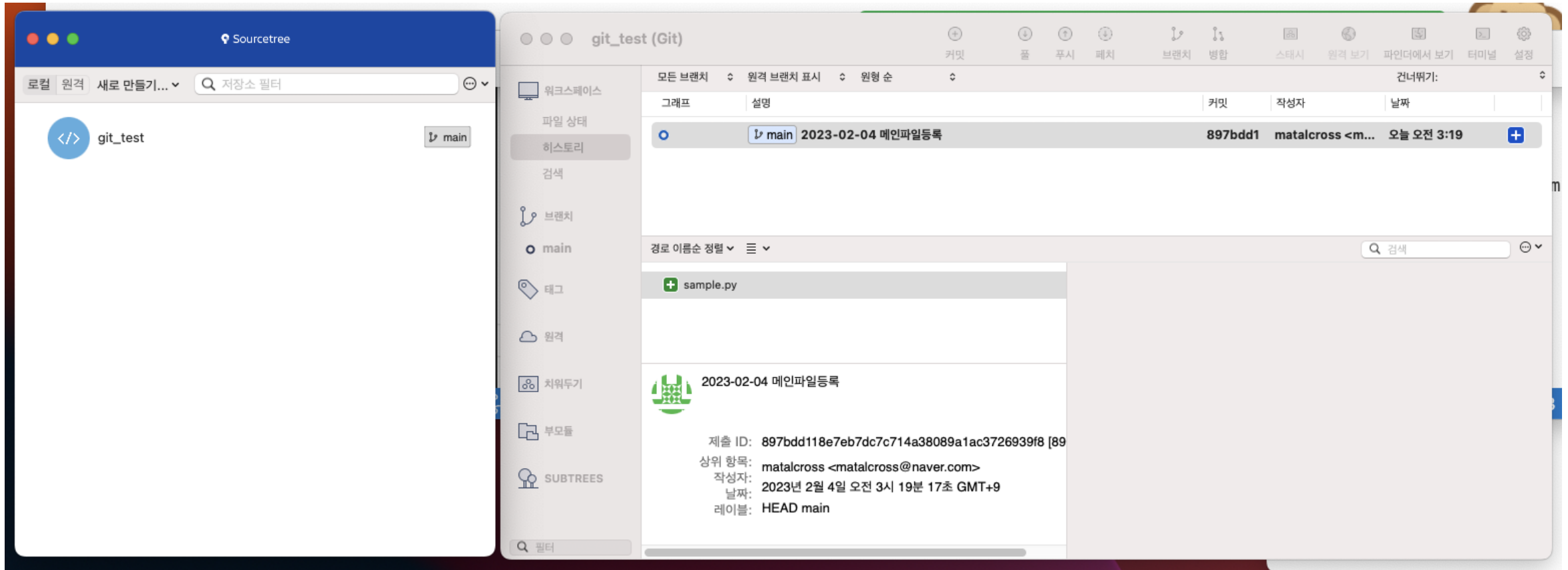
(END)

---



# GIT 새 저장소 만들기(로컬)

## SourceTree에서 확인



## GIT 원격저장소 사용하기(서버)

## 1 push

내 PC의 로컬 저장소에서 변경된 이력을 원격 저장소로 저장하는 업로드

## 2 clone


원격 저장소의 파일을 내 PC로 다운로드

## 3 pull

원격 저장소를 공유해 여러사람이 작업을 하면, 같은 저장소에서 푸시(Push)한다. 다른사람의 작업을 받을 경우 내 저장소에서 적용(Pull:다운로드)한다.

# GIT 원격저장소(서버)공유 - 생성

## Quick setup — if you've done this kind of thing before

 Set up in Desktop

or

HTTPS

SSH



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

## ...or create a new repository on the command line

```
echo "# git_test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/matacross/git_test.git
git push -u origin main
```



## ...or push an existing repository from the command line

```
git remote add origin https://github.com/matacross/git_test.git
git branch -M main
git push -u origin main
```



## ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

# GIT 원격저장소(서버)공유 - 푸시(push)

## 원격저장소 경로 저장

\$ git remote add origin git경로


## 원격저장소에 저장

\$ git push -u origin main


```
• (base) imac.ai ~/ai_exam/flutter/git_test [main] git branch -M main
• (base) imac.ai ~/ai_exam/flutter/git_test [main] git remote add origin https://github.com/matalcross/git_test.git
• (base) imac.ai ~/ai_exam/flutter/git_test [main] git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 289 bytes | 289.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/matalcross/git_test.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
○ (base) imac.ai ~/ai_exam/flutter/git_test [main]
```

# GIT 원격저장소(서버)공유 - 푸시(push)

main 1 branch 0 tags Go to file Add file <> Code

 **matalcross** 2023-02-04 메인파일등록

897bdd1 33 minutes ago 1 commit

 **sample.py**

2023-02-04 메인파일등록

33 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

## About

깃 사용법 테스트

0 stars

1 watching

0 forks

## Releases

No releases published

[Create a new release](#)

## Packages

No packages published

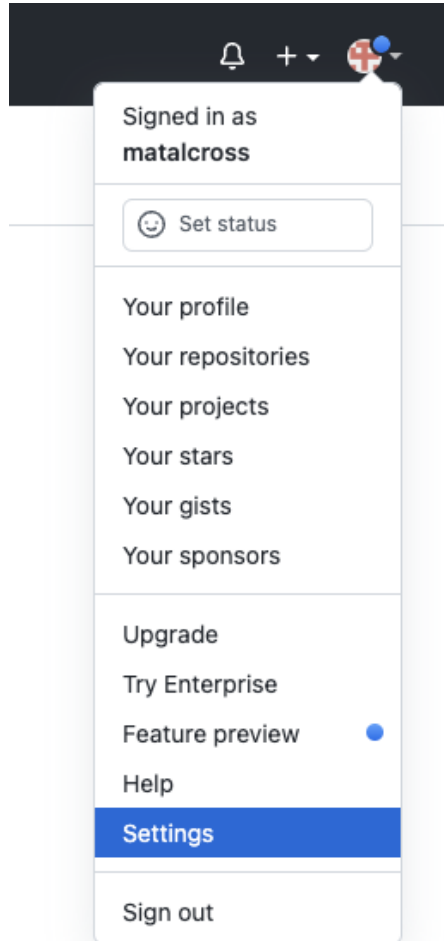
[Publish your first package](#)

## Languages

Python 100.0%



# GIT 원격저장소(서버)공유 - 키발급



Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Beta

Tokens (classic)

## Personal access tokens (classic)

Generate new token

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

## New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Note

개발자 토큰 정보

What's this token for?

### Expiration

90 days

The token will expire on Fri, May 5 2023

### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).


- |   |                                      |
|---|--------------------------------------|
| <input checked="" type="checkbox"/> repo            | Full control of private repositories |
| <input checked="" type="checkbox"/> repo:status     | Access commit status                 |
| <input checked="" type="checkbox"/> repo_deployment | Access deployment status             |
| <input checked="" type="checkbox"/> public_repo     | Access public repositories           |
| <input checked="" type="checkbox"/> repo:invite     | Access repository invitations        |
| <input checked="" type="checkbox"/> security_events | Read and write security events       |

# GIT 원격저장소(서버)공유 - 키발급

[Settings](#) / Developer settings

 GitHub Apps

 OAuth Apps

 **Personal access tokens** ^

Fine-grained tokens

Beta

Tokens (classic)

## Personal access tokens (classic)

Generate new token ▾

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp\_7VAFt6a3twugdQKqK1jst0aP2TT2qL3PXt6g 

Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).



© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

# GIT 원격저장소(서버)공유 - 받기(pull)

## 원격저장소에서 파일받기(pull)

\$ git pull origin main

```
PS Microsoft.PowerShell.Core\FileSystem::\\Mac\Home\Downloads\ai_exam\booktickets> git branch -M main
PS Microsoft.PowerShell.Core\FileSystem::\\Mac\Home\Downloads\ai_exam\booktickets> git remote add origin https://github.com/matalcross/git_test.git
PS Microsoft.PowerShell.Core\FileSystem::\\Mac\Home\Downloads\ai_exam\booktickets> git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 269 bytes | 10.00 KiB/s, done.
From https://github.com/matalcross/git_test
* branch          main          -> FETCH_HEAD
* [new branch]     main          -> origin/main
PS Microsoft.PowerShell.Core\FileSystem::\\Mac\Home\Downloads\ai_exam\booktickets> 
```

# GIT 원격저장소(서버)공유 - 병합(Merge)

## 원격저장소 변경이력 병합(Merge)

다른사람이 원격지에 파일을 수정한 곳에 내가 수정하고 싶은 경우

```
$git config pull.rebase false
```

```
$ git pull origin main
```

```
6
  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
7 <<<<<<< HEAD (Current Change)
8 # 파일 다시 수정
9 print("파일 다시 수정")
10
11 # 내가 수정한 부분
12 # matalcross
13 print("오류가 있네")
14 =====
15 # 파일 다시 수정 - 잘못되어서 제가 수정했습니다.
16 # 수정자 : matalcross.ai
17 print("파일 다시 수정 111")
18 >>>>>>> 688729359d4ccb25a6c0462a7c8d2a400ff4804d (Incoming Change)
19
```

[Resolve in Merge Editor](#)

# GIT 원격저장소(서버)공유 - 병합(Merge)

## 원격저장소 변경이력 병합(Merge)

다른사람이 원격지에 파일을 수정한 곳에 내가 수정하고 싶은 경우

```
$git config pull.rebase false
```

```
$ git pull origin main
```

```
6
  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
7 <<<<<<< HEAD (Current Change)
8 # 파일 다시 수정
9 print("파일 다시 수정")
10
11 # 내가 수정한 부분
12 # matalcross
13 print("오류가 있네")
14 =====
15 # 파일 다시 수정 - 잘못되어서 제가 수정했습니다.
16 # 수정자 : matalcross.ai
17 print("파일 다시 수정 111")
18 >>>>>>> 688729359d4ccb25a6c0462a7c8d2a400ff4804d (Incoming Change)
19
```

[Resolve in Merge Editor](#)

# GIT 원격저장소(서버)공유 - 로그(log)

## 로그 확인하기(그래프로)

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

```
* 51bc054 (HEAD -> main, origin/main) 2023-02-04 중복 오류가 있어서 병합함 (matalcross)
|\
| * 6887293 2023-02-04 박성주 수정하신 부분 오류가 있어서 수정함.
* | 25edfda 2023-02-04 오류가 있어서 수정하였음 (matalcross)
|/
* 58e0ab0 2023-02-04 박성주 3번째 수정 파일
* 5b4a4be 2023-02-04 박성주 -수정 파일
* 897bdd1 2023-02-04 메인파일 등록
(END)
```

---



# GIT 원격저장소(서버)공유 - 이전분기(reset)

## 이전 파일로 분기(reset)

\$ git reset --hard <커밋명>

2023-02-04 메인파일등록

```
PS Microsoft.PowerShell.Core\FileSystem::\\Mac\Home\Downloads\ai_exam\booktickets> git reset --hard 58e0ab0e54eb9bddca030649ad894dbe76d6c0bd
HEAD is now at 58e0ab0 2023-02-04 박성주 3번째 수정파일
PS Microsoft.PowerShell.Core\FileSystem::\\Mac\Home\Downloads\ai_exam\booktickets> █
```

원상복구(commit 취소)

\$ git reset --hard ORIG\_HEAD

<command> [<args>]

```
PS Microsoft.PowerShell.Core\FileSystem::\\Mac\Home\Downloads\ai_exam\booktickets> git reset --hard ORIG_HEAD
HEAD is now at 51bc054 2023-02-04 중복오류가 있어서 병합함(matalcross)
PS Microsoft.PowerShell.Core\FileSystem::\\Mac\Home\Downloads\ai_exam\booktickets> █
```

## GIT 브랜치(Branch) 사용하기

## 1 통합 브랜치(Integration Branch)

언제든지 배포할 수 있는 버전의 브랜치(안정버전)  
보통은 master를 통합 브랜치로 사용한다.

## 2 토픽 브랜치(Topic Branch)

기능 추가 버그 수정과 같은 단위 작업  
여러개의 작업을 진행할때 사용  
피쳐 브랜치(Feature branch)라고 함.

## 3 브랜치 전환(체크아웃 checkout)

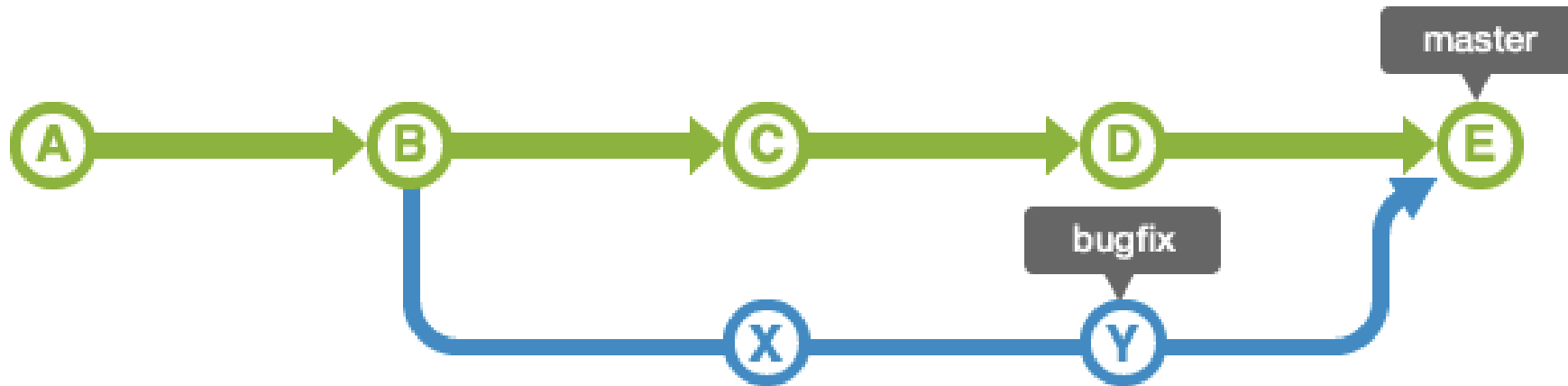
브랜치를 바꾸고 새로운 브랜치로 이동하기 위한 전환

## 4 Stash 임시저장

Branch에서 저장되지 않은 상태로 다른 브랜치로 이동하는 경우 임시저장하는 처리

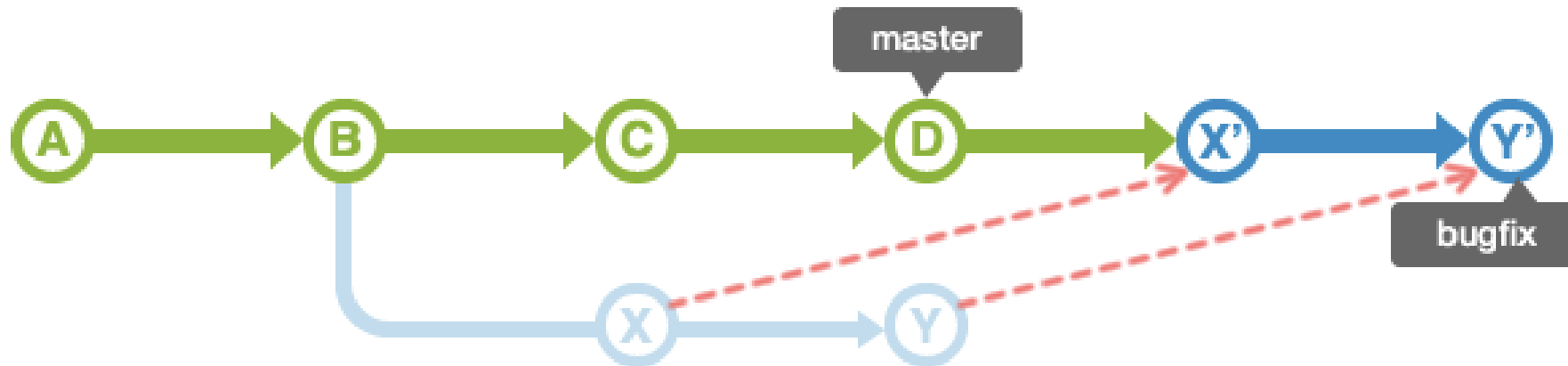
## 5 브랜치 통합하기(merge)

두개의 브랜치를 통합하는 방법  
변경 내용의 이력이 모두 남아있어, 복잡해짐



## ⑥ 브랜치 통합하기(rebase)

두개의 브랜치를 하나로 통합처리됨.  
이력이 단순해짐, 이력을 남길필요가 없을경우 사용





# GIT 브랜치(Branch) 변경하기

## 브랜치 만들기

```
$ git branch <브랜치 이름>  
$ git branch release
```

## 브랜치 확인

```
$ git branch
```

```
PS Microsoft.PowerSh  
* main  
  release  
PS Microsoft.PowerSh
```

# GIT 브랜치(Branch) 변경하기

## 브랜치 전환 체크아웃(checkout)

\$ git checkout <브랜치명>

\$ git checkout release

\$ git branch

```
Switched to branch 'release'
PS Microsoft.PowerShell> git checkout release
Switched to branch 'release'
main
* release
PS Microsoft.PowerShell>
```

# GIT 브랜치(Branch) 변경하기

## 브랜치 삭제하기

\$ git branch -d <브랜치이름>

```
Switched to branch  
PS Microsoft.Power  
main  
* release  
PS Microsoft.Power
```

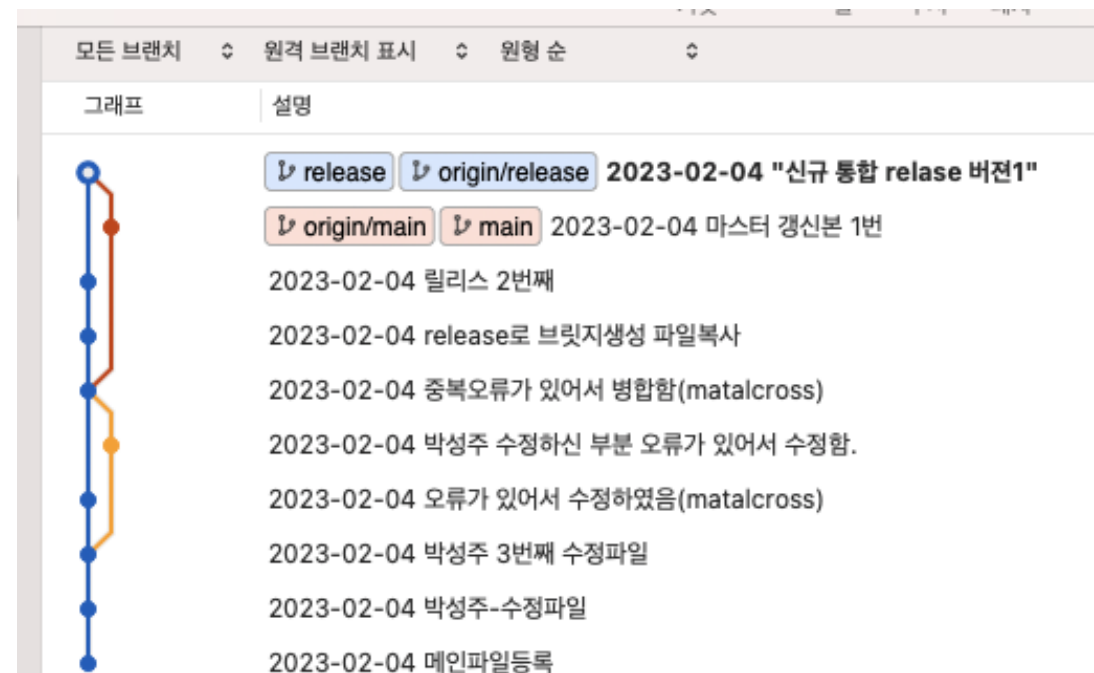
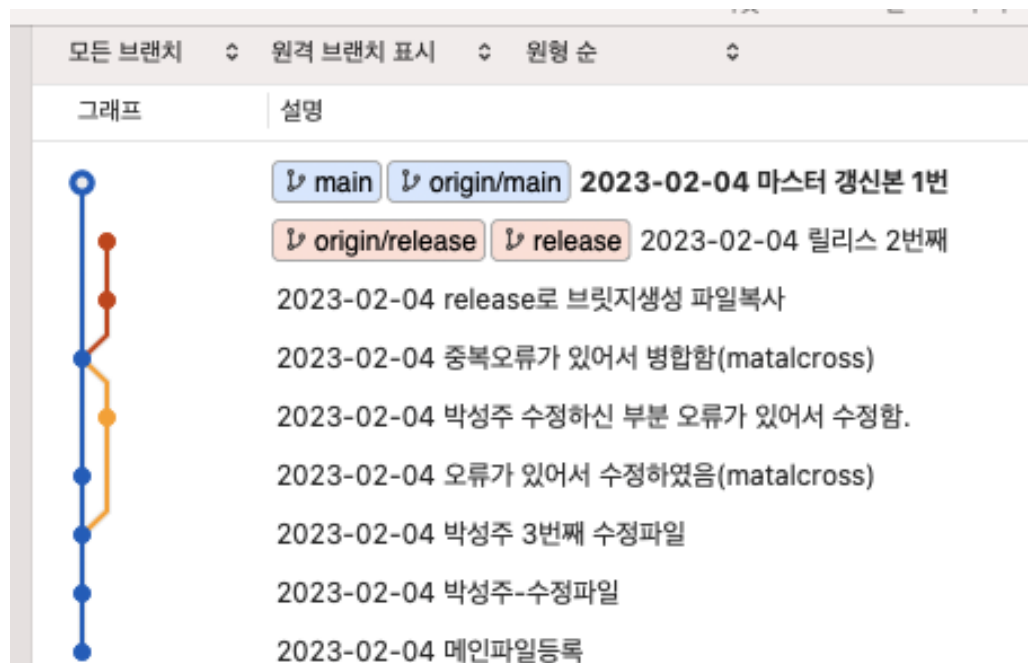
# GIT 브랜치(Branch) 변경하기

## 브랜치별 프로젝트 진행 병합

\$ git checkout <로그인된브랜치>

\$ git checkout release

\$ git merge main <= relase에 main을 병합

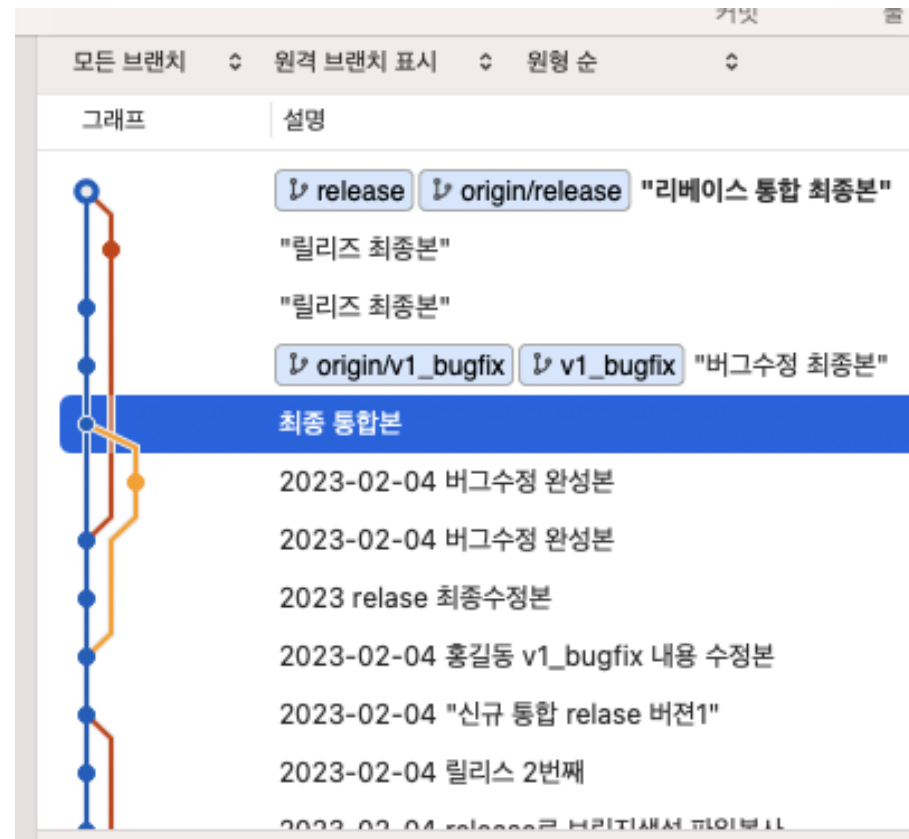
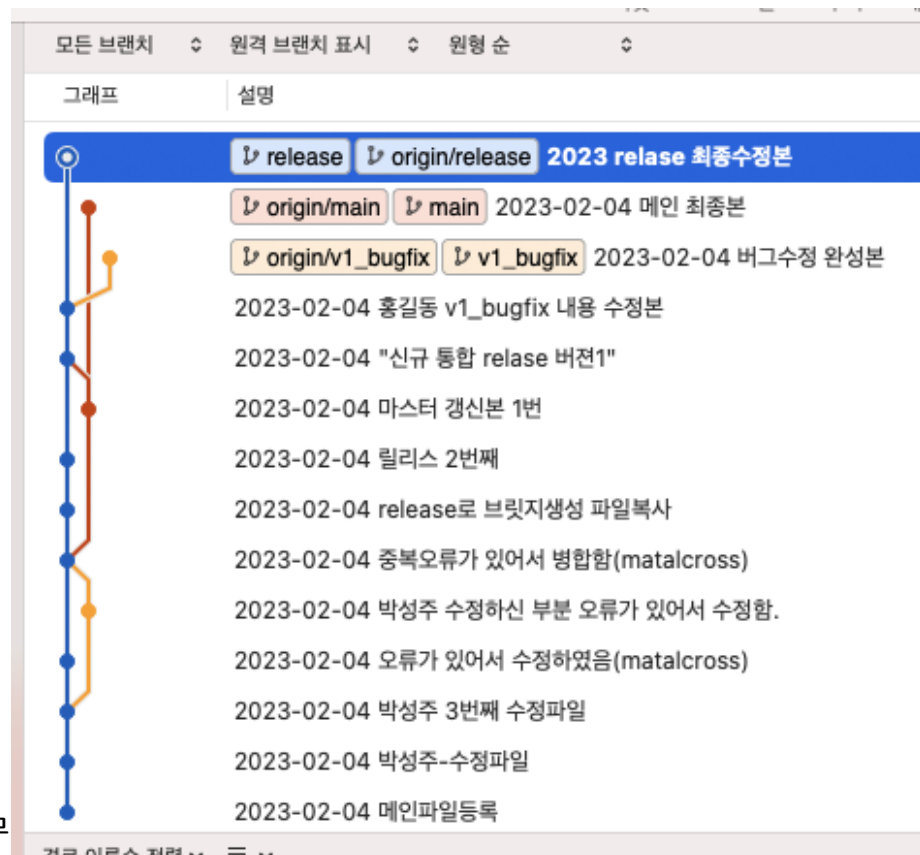


# GIT 브랜치(Branch) 변경하기

## 브랜치별 rebase로 병합

\$ git checkout release

\$ git rebase v1\_bugfix <- release 로 병합(v1\_bugfix 사라짐)

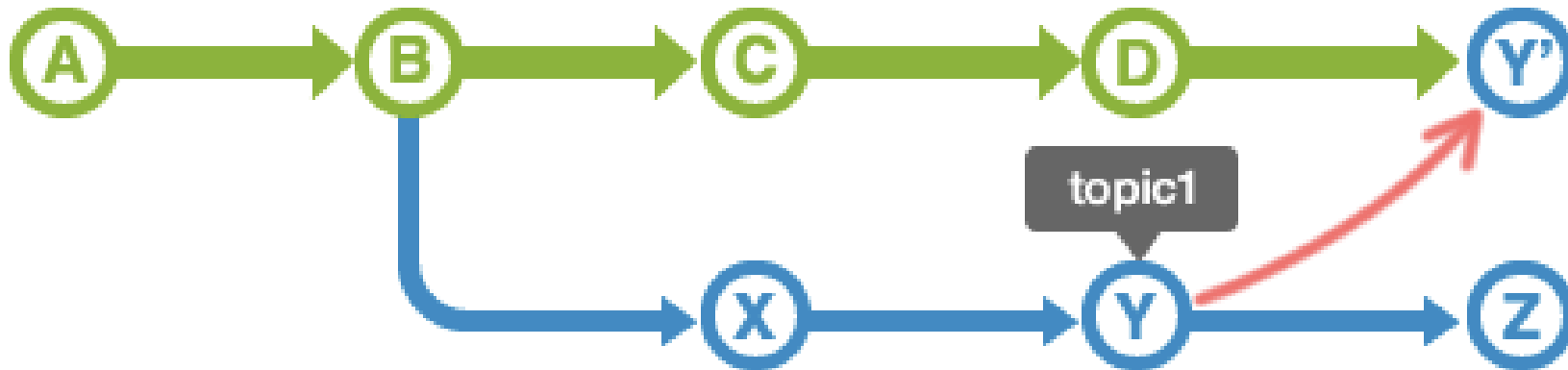


# GIT 브랜치(Branch) 변경하기

## 다른 브랜치로 특정 커밋 가져오기(cherry-pick)

\$ git checkout release

\$ git cherry-pick <커밋정보>





2025년 서울 민간기업 맞춤형 매력일자리 사업

# GIT 활용법

- 2025.05.02 -