

GIT 빠른 참조

브랜치, 머지, 리메이스, 스테시, 리모트

설정

사용자 설정

```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
git config --global core.editor vim
git config --list # show all settings
```

초기화 & 클론

```
git init # new repo in current dir
git clone <url> # copy remote repo
git clone <url> dir # clone into dir/
```

기본

상태 & 스테이징

```
git status # working tree status
git add file.txt # stage a file
git add . # stage all changes
git add -p # stage interactively
```

커밋

```
git commit -m "msg" # commit staged changes
git commit -am "msg" # stage tracked + commit
git commit --amend # edit last commit
```

Diff

```
git diff # unstaged changes
git diff --staged # staged changes
git diff HEAD~1 # vs previous commit
git diff branchA branchB
```

브랜치

브랜치 관리

```
git branch # list local branches
git branch -a # list all (incl. remote)
git branch feat # create branch
git branch -d feat # delete (safe)
git branch -D feat # delete (force)
```

브랜치 전환

```
git checkout feat # switch to branch
git checkout -b feat # create + switch
git switch feat # switch (modern)
git switch -c feat # create + switch (modern)
```

머지

```
git merge feat # merge feat into current
git merge --no-ff feat # always create merge commit
git merge --abort # cancel conflicted merge
```

리모트

리모트 관리

```
git remote -v # list remotes
git remote add origin <url> # add remote
git remote remove origin # remove remote
```

페치, 풀, 푸시

```
git fetch origin # download updates
git pull # fetch + merge
git pull --rebase # fetch + rebase
git push origin main # push to remote
git push -u origin feat # push + set upstream
```

로그

히스토리 보기

```
git log # full commit log
git log --oneline # compact one-line format
git log --oneline -10 # last 10 commits
git log --graph --oneline --all # branch graph
git log -p # show patches (diffs)
git log --stat # show file change stats
```

로그 필터링

```
git log --author="Alice"
git log --since="2024-01-01"
git log -- path/file # commits touching file
git log -S "keyword" # search content changes
```

스테시

```
git stash # save working changes
git stash push -m "wip" # save with message
git stash list # list all stashes
git stash pop # apply + remove top stash
git stash apply # apply but keep stash
git stash drop # remove top stash
git stash clear # remove all stashes
```

변경 취소

언스테이지 & 폐기

```
git restore file.txt # discard working changes
git restore --staged file.txt # unstage file
git checkout -- file.txt # discard (legacy)
```

리셋

```
git reset --soft HEAD~1 커밋 취소, 스테이지 유지
git reset --mixed HEAD~1 커밋 취소, 워킹 트리 유지 (기본값)
git reset --hard HEAD~1 커밋 취소, 모든 변경 폐기
```

리버트

```
git revert <commit> # new commit undoing changes
git revert HEAD # undo last commit (safe)
```

태그

```
git tag v1.0 # lightweight tag
git tag -a v1.0 -m "Release" # annotated tag
git tag # list tags
git push origin v1.0 # push single tag
git push origin --tags # push all tags
git tag -d v1.0 # delete local tag
```

.gitignore

패턴 예시

```
# .gitignore
*.log # all .log files
build/ # build directory
!important.log # exception (do track)
/TODO # only root TODO
doc/**/*.pdf # pdfs in doc/ subtree
```

일반 패턴

```
*.ext 해당 확장자의 모든 파일
dir/ 전체 디렉토리
!pattern 무시 제외 (다시 추적)
**/name 모든 하위 디렉토리에서 매칭
? 단일 문자 와일드카드
```