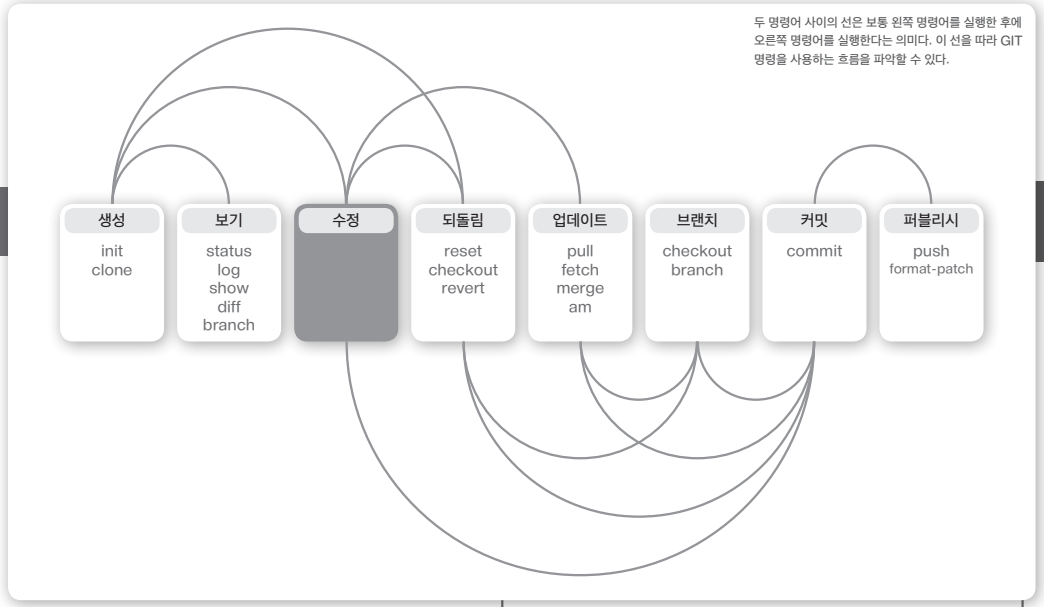
GIT 작업 흐름과 명령어

도움말: git "명령어" --help
Git의 글로벌 설정은 $HOME/.gitconfig에 저장 (git config --help)



생성

새 저장소 생성하기
```
cd ~/projects/myproject
git init
git add .
```

기존 저장소 Clone하기
```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone you@host.org/project.git
```

보기

워킹 디렉터리의 파일 상태 보기
```
git status
```

파일의 변경사항 보기
```
git diff
```

$ID1과 $ID2 사이의 변경사항 보기
```
git diff $id1 $id2
```

커밋 히스토리 보기
```
git log
```

특정 파일의 커밋 히스토리별 변경사항 보기
```
git log -p $file $dir/ec/tory/
```

특정 파일을 누가 언제 고쳤는지 보기
```
git blame $file
```

$ID 커밋 보기
```
git show id
```

$ID 버전의 파일 보기
```
git show $id:$file
```

로컬 브랜치들 보기
```
git branch
```
('*' 표시는 현재 브랜치를 나타냄)

범례

$id - 커밋 ID, 브랜치 이름, 태그 이름을 나타냄
$file - 파일 이름
$branch - 브랜치 이름

개념

Git 기본

master : 기본 브랜치
origin : 기본 리모트 저장소
HEAD : 현재 브랜치
HEAD^ : HEAD의 부모
HEAD~4 : HEAD의 부모의 부모의 부모의 부모

되돌림

마지막 커밋 시점으로 되돌리기
```
git reset --hard
```
⚠Hard Reset은 되돌릴 수 없음

마지막 커밋 내용을 되돌리고 커밋하기
```
git revert HEAD
```
새로운 커밋 생성

특정 커밋 내용을 되돌리고 커밋하기
```
git revert $id
```
새로운 커밋 생성

마지막 커밋 수정하기
```
git commit -a --amend
```
(잘못 커밋해서 수정하고 싶을 때)

$id 시점의 파일을 꺼내기
```
git checkout $id $file
```

브랜치

브랜치를 Checkout하기
```
git checkout $id
```

$branch1을 $branch2에 Merge하기
```
git checkout $branch2
git merge $branch1
```

새 브랜치 만들기
```
git branch $branch
```

$other와 같은 커밋을 가리키는 브랜치를 새로 만들고 바로 Checkout하기
```
git checkout -b $new_branch $other
```

$branch를 삭제하기
```
git branch -d $branch
```

업데이트

origin에서 최신 데이터를 가져오기
```
git fetch
```
(Merge하지는 않음)

origin에서 최신 데이터를 가져와 Merge하기
```
git pull
```
(Fetch하고 Merge까지 함)

누군가 보낸 패치를 Merge하기
```
git am -3 patch.mbox
```
(충돌이 발생하면 해결 후 git am --resolved)

퍼블리시

현재 모든 수정사항을 커밋하기
```
git commit -a
```

다른 개발자에게 보낼 패치를 작성하기
```
git format-patch origin
```

origin으로 업데이트를 Push하기
```
git push
```

버전이나 마일스톤을 생성하기
```
git tag v1.0
```

유용한 명령어

문제가 발생한 커밋 이진탐색하기
```
git bisect start
```
(이진탐색 시작)
```
git bisect good $id
```
($id를 문제 없는 상태로 표시)
```
git bisect bad $id
```
($id를 문제 있는 상태로 표시)

```
git bisect bad/good
```
(현재 상태가 문제가 있는지 없는지 설정)
```
git bisect visualize
```
(gitk를 실행하여 확인함)
```
git bisect reset
```
(시작했던 상태로 Checkout 함)

저장소의 무결성 검사 및 저장소 청소하기
```
git fsck
git gc --prune
```

워킹 디렉터리에서 'foo()'라는 문자열 검색하기
```
git grep "foo()"
```

Merge 충돌 해결

Merge 충돌 내용 보기
```
git diff
```
(충돌 내용 전체 보기)
```
git diff --base $file
```
(Merge Base를 기준으로)
```
git diff --ours $file
```
(현 브랜치를 기준으로)
```
git diff --theirs $file
```
(Merge할 브랜치를 기준으로)

충돌이 생기는 패치 버리기
```
git reset --hard
git rebase --skip
```

충돌 해결 후 Merge 진행하기
```
git add $conflicting_file
```
(충돌 해결한 파일)
```
git rebase --continue
```