??

- Git □□□ □□ □□ □□ : merge, rebase, cherry-pick

Git ??? ?? ?? ??: merge, rebase, cherry-pick

??

1. 기본___개념___요약__
2. 명령어별_____상세___설명__
3. 커밋___이력___비교__
4. 적합한____상황__
5. 주의사항____
6. 요약___표_
7. 시각적____예제__
8. 추가___명령어___: `revert`, `reset`, `stash`

---

1. ?? ?? ??

명령어	설명
`merge`	브랜치 간 변경사항을 병합 , 병합 커밋 생성
`rebase`	커밋 기반을 다른 브랜치 위로 옮겨 이력 재정렬
`cherry-pick`	특정 커밋만 골라 현재 브랜치에 복사

---

2. ???? ?? ??

`git merge <브랜치명>`

- 역할 : 다른 브랜치의 변경 내용을 현재 브랜치에 병합
- 병합 커밋 을 만들어 두 브랜치의 내용을 하나로 합침

```
git checkout main         # 병합 대상 브랜치
git merge feature/login   # 병합할 소스 브랜치
```

- □□ 즉 : `main ← feature/login` 병합됨

---

git rebase <브랜치명>

- 역할 : 현재 브랜치 커밋을 `<브랜치명>` 위로 옮김
- 커밋이 새로 복사되어 **SHA** 변경
- 병합 커밋 없이 깔끔한 직선형 이력 생성

```
git checkout feature/login
git rebase main
```

- □□ 즉 : `feature/login`을 `main` 기준으로 다시 쌓음

---

git cherry-pick <커밋해시>

- 역할 : 특정 커밋 하나를 현재 브랜치로 가져옴
- 전체 브랜치를 병합하지 않고 선택적 반영

```
git checkout main
git cherry-pick 7a1f3b2
```

---

3. ?? ?? ??

merge

```
A---B---C---M       (main)
     \     /
      D---E         (feature)
```

rebase

```
A---B---C---D'---E' (feature rebased onto main)
```

`cherry-pick`

```
main:    A---B---C---E'
feature:          D---E
```

---

4. ??? ??

상황	추천 명령어
팀 프로젝트에서 병합 이력 유지	`merge`
개인 작업 커밋 정리	`rebase`
특정 버그 수정만 따로 반영	`cherry-pick`

---

5. ????

- `merge`: 병합 커밋으로 이력이 복잡해질 수 있음
- `rebase`: 협업 중인 브랜치에 사용 금지 (이력 충돌)
- `cherry-pick`: 동일 커밋이 중복될 수 있음

---

6. ?? ?

항목	**merge**	**rebase**	**cherry-pick**
대상	브랜치 전체	브랜치 전체	특정 커밋
이력 변경 여부	□ (유지)	□ (SHA 변경됨)	□ (선택적 커밋 복사)
병합 커밋	생성됨	생성 안됨	없음
협업 안정성	□ □□	⚠ 위험	□ □□

---

7. ??? ??

```
# 예제 환경 구성
git init
```

```
echo "A" > f.txt && git add . && git commit -m "A"
echo "B" >> f.txt && git commit -am "B"
git checkout -b feature
echo "C" >> f.txt && git commit -am "C"
git checkout main
echo "D" >> f.txt && git commit -am "D"
```

merge

```
git merge feature
```

rebase

```
git checkout feature
git rebase main
```

cherry-pick

```
git log --oneline  # 커밋 해시 확인
git cherry-pick <커밋 해시>
```

---

8. ?? ???: `revert`, `reset`, `stash`

명령어	역할	주 사용 목적
`revert`	특정 커밋을 취소하는 새 커밋 생성	실수한 커밋을 안전하게 되돌림
`reset`	커밋 이력 /스테이징 /작업 폴더 상태를 되돌림	브랜치나 파일을 과거 상태로 복구
`stash`	현재 작업을 임시로 저장	급히 브랜치를 바꿔야 할 때

```
# revert: 안전한 취소
git revert <커밋 해시>

# reset: 이력 되돌리기
git reset --soft HEAD~1   # 커밋만 되돌림
git reset --hard HEAD~1   # 작업 내용까지 삭제
```

```
# stash: □□ □□ □□
git stash
git stash pop
```
