# 2주차

- revision selection: 7.1
- interactive staging: 7.2
- stash: 7.3
  - push
  - apply & pop
  - drop
- bisect: 7.5
- rebase deep dive: 7.6
  - commit 합치기
  - commit 분리하기
  - `-onto`
- reset & checkout는 어떻게 작동할까?: 7.7

## 7.1

### SHA-1 해시

입력 데이터를 고정된 길이의 고유한 40자리(16진수) 문자열로 변환하는 해시 알고리즘. Git은 이 해시 알고리즘을 사용해 커밋의 내용(변경된 파일 내용, 커밋 메시지, 시간 등)을 바탕으로 고유한 커밋 ID를 만든다.

=각 커밋 고유하게 식별하는 긴 문자열(고유ID)

### SHA-1 줄여 쓰기

해시가 저장소 내에서 유일하다면 앞부분 4글자만으로도 커밋을 나타낼 수 있음

`$ git log --abbrev-commit` : 짧고 중복되지 않는 해시 값을 보여줌

```
$ git log --abbrev-commit --pretty=oneline
ca82a6d changed the version number
```

```
085bb3b removed unnecessary test code
a11bef0 first commit
```

## 브랜치로 가리키기

어떤 커밋이 브랜치의 가장 최신 커밋이라면, 커밋을 SHA-1 값이 아닌 브랜치 이름으로 가리키기

`$ git show topic1` → topic1 브랜치의 최근 커밋 보는 것

## RefLog로 가리키기

Reflog=브랜치, HEAD가 지난 몇 달 동안에 가리켰었던 커밋의 기록(로그)

`git reflog` : Reflog 볼 수 있음

```
$ git reflog
734713b HEAD@{0}: commit: fixed refs handling, added gc aut
o, updated
d921970 HEAD@{1}: merge phedders/rdocs: Merge made by the
'recursive' strategy.
1c002dd HEAD@{2}: commit: added some blame and merge stuff
1c36188 HEAD@{3}: rebase -i (squash): updating HEAD
95df984 HEAD@{4}: commit: # This is a combination of two co
mmits.
1c36188 HEAD@{5}: rebase -i (squash): updating HEAD
7e05da5 HEAD@{6}: rebase -i (pick): updating HEAD
```

## 계통 관계로 가리키기

이름 끝에 `^` 기호를 붙임 → 해당 커밋의 부모 찾기

`HEAD^` = 바로 이전 커밋을 보여줌.

`$ git show HEAD~3` = `$ git show HEAD^^^`

## 범위로 커밋 가리키기

범위를 주고 여러 커밋을 한꺼번에 조회하기

- double dot

  어떤 커밋들이 한쪽에는 관련됐고 다른 쪽에는 관련되지 않았는지 확인

`master..experiment` : `master`에는 없고 `experiment`에만 있는 것 알려줌

`experiment..master` : `experiment` 에는 없고 `master` 에만 있는 것 알려줌

- triple dot

  양쪽에 있는 두 Refs 사이에서 공통으로 가지는 것을 제외하고 서로 다른 커밋만 보여줌

  `$ git log master...experiment` : `master` 와 `experiment` 의 공통부분은 빼고 다른 커밋만 보기

  `$ git log --left-right master...experiment` : 각 커밋이 어느 브랜치에 속하는지도 보여줌

# 7.2

## 대화형 모드로 들어가기

`$ git add -i`

## Staging Area에 파일 추가하고 추가 취소하기

`What now> 2` : Staging Area에 추가할 수 있는 파일을 전부 보여줌

`Update>> 1,2` : 1,2번 파일을 Stage에 추가

`Update>>` : 파일들 Staging Area로 추가

`Revert>> 1` : 1번 파일 Stage 취소

## 파일의 일부분만 Staging Area에 추가하기

대화형 프롬프트에서 `5`, `p` 를(patch) 입력 → 부분적으로 Staging Area에 추가할 파일이 있는지 물음

`git add -p` , `git add --patch` : 대화형 모드 없이도 파일의 일부만 Stage에 올리는 방법

# 7.3

## Stashing

`git stash save` = `git stash push` : 작업 중인 변경사항을 임시로 저장하는 기능.Stash 이후에는 워킹 디렉토리가 깨끗해짐

`git stash apply stash@{2}` : 선택한 Stash를 다시 적용

`git stash drop stash@{2}` : 해당 Stash를 제거(apply 옵션은 단순히 Stash를 적용하기에 Stash는 여전히 스택에 남아 있음)

`git stash pop` : Stash를 적용하고 나서 바로 스택에서 제거

# 7.5

## Git Grep

`git grep` : 워킹 디렉토리나 커밋 트리에서 특정 문자열이나 정규 표현식을 검색

## Git Log

`git log` : 커밋 히스토리 내에서 특정 내용이 추가되거나 삭제된 커밋을 찾음

# 7.6

## 마지막 커밋 수정

- 커밋 메세지만 수정

  `$ git commit --amend`

- 수정한 파일 마지막 커밋 안에 밀어넣기

  $ git add <수정된 파일>
  $ git commit --amend

- 커밋 메시지를 유지(편집기 실행X)

  `$ git commit --amend --no-edit`

## 커밋 합치기

"squash" 입력 → 해당 커밋과 바로 이전 커밋을 합침+커밋 메시지도 Merge

```
pick f7f3f6d changed my name a bit
squash 310154e updated README formatting and added blame
```

```
squash a5f4a0d added cat-file
```

## 커밋 분리하기

1. 기존의 커밋을 해제
2. Stage를 여러 개로 분리
3. 그것을 원하는 횟수만큼 다시 커밋

```
pick f7f3f6d changed my name a bit
edit 310154e updated README formatting and added blame
pick a5f4a0d added cat-file
```

```
$ git reset HEAD^ ->커밋 해제. 커밋된 변경 사항들을 Stage에서 Unst
aged 상태로 되돌림.
$ git add README
$ git commit -m 'updated README formatting' ->README 파일 변
경 사항을 Stage에 추가,

updated README formatting이라는 메시지로 커밋
$ git add lib/simplegit.rb
$ git commit -m 'added blame' ->lib/simplegit.rb 파일 변경 사
항을 Stage에 추가,

added blame이라는 메시지로 커밋
$ git rebase --continue
```

## onto

기존의 브랜치를 다른 커밋이나 브랜치 위로 이동시키고자 할 때 사용

`git rebase --onto <새로운 기준 커밋> <이동할 시작 커밋> <브랜치 이름>`

- feature 브랜치를 F커밋 위로 옮기기

  A - B - C - D - E - F  (main 브랜치)
  \
  X - Y - Z  (feature 브랜치)

  $ git rebase --onto F B feature

  A - B - C - D - E - F - X' - Y' - Z'

# 7.7

HEAD = 현재 브랜치를 가리키는 포인터(현재 브랜치의 마지막 커밋)

브랜치 = 브랜치에 담긴 커밋 중 가장 마지막 커밋

Index = 바로 다음에 커밋할 것들

워킹 디렉토리 = 실제 파일이 위치하는 작업 공간. 커밋 전 변경 사항을 자유롭게 수정할 수 있는 영역

## reset

1. `reset --soft`

- HEAD 브랜치를 이동시킴
- HEAD가 가리키는 커밋을 과거의 특정 커밋으로 변경
- Index와 워킹 디렉토리에는 아무런 영향X

2. `--mixed`

- Index를 현재 HEAD가 가리키는 스냅샷으로 업데이트
- 커밋과 Staging Area 모두 초기화

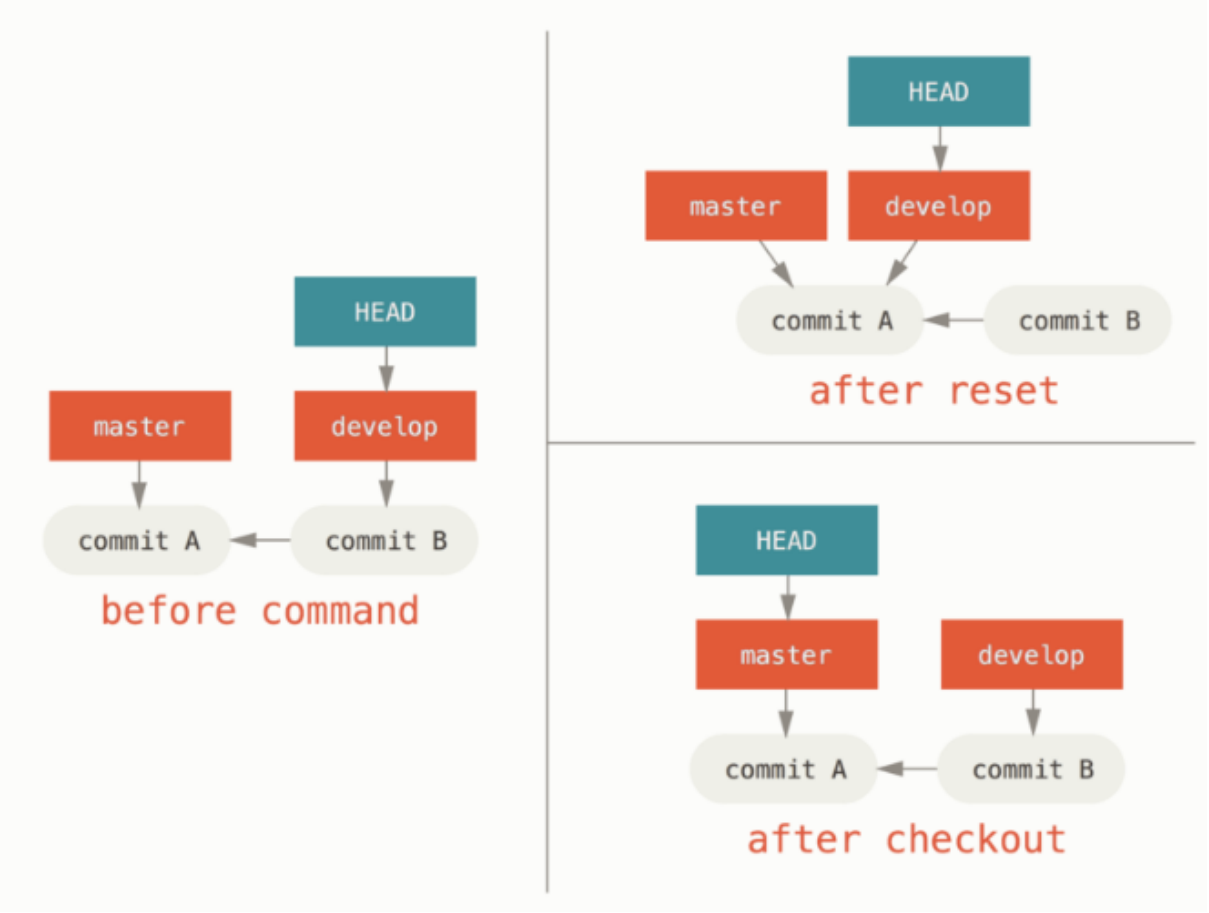
3. `--hard`

- 워킹 디렉토리까지 업데이트

## checkout

`git checkout [branch]`

- 워킹 디렉토리에서 Merge 작업을 한번 시도해보고 변경하지 않은 파일만 업데이트함
- HEAD 자체를 다른 브랜치로 옮김




# 7.10

## bisect

`git bisect` : 커밋 히스토리를 이진 탐색 방법으로 좁혀줌→ 이슈와 관련된 커밋을 최대한 빠르게 찾을 수 있음

```
$ git bisect start ->이진 탐색 시작
$ git bisect bad ->현재 커밋이 문제가 있는 상태라는 것을 표시
$ git bisect good v1.0 ->문제가 없는 마지막 커밋 표시
```