13~15

버전(version)

프로그램을 수정하거나 개선할 때마다 코드를 구분하려고 부여된 식별자를 의미

보통 두 자리 또는 세 자리 형태의 숫자(1.0 / 2.1.4) 를 사용하나 '년,월'을 사용하기도 함

SemVer(Semantic Versioning) 방식

major.minor.patch 세 자리 숫자 형태로 표기하는 버전

메이저 번호

- 첫 자리가 0으로 시작하면 아직 초기 개발중인 제품이라는 의미
- 정식 버전은 1부터 시작

마이너 번호

- 메이버 버전에서 기능을 추가하거나 변경 사항이 있을 때 바꿈

패치 번호

- 버그 수정 등 의미한 변화가 있을 때 바꿈

태그(tag) 기능

- 태그 : 특정 커밋(해시 값)에 버전 번호나 다른 이름을 부여하는 기능(ex. v 1.4.0)

[태그 2가지 종류]

- 주석 태그(Annotated tag) : 태그 이름+정보(태그 작성자 이메일, 태그 시각, 태그 메시지) 포함
- 일반(가벼운) 태그(Lightweight tag) : 태그 이름만 포함

주석 태그 생성

주석 태그 : 주석(누가,언제,태그 메시지 등의 정보)이 있는 태그 / 태그 버전 이름 중복 불가능

- $ git tag -a v1.0.0 -m 'first version' : 작성한 사람의 이메일,날짜,메시지 등 정보 포함

- $ git tag -a v1.0.0 : 기본 설정된 편집기로 메시지 편집 → 사용 어려움. $ git config - -global init.defaultBranch 'code - -wait' 설정 필요
- $ git tag -a v1.1.0 commitID : 특정 커밋에 태그를 붙임

일반 태그 생성

태그의 버전 정보만 관리

$ git tag v1.0.1 : 태그의 버전 정보만 관리 / 태그 버전 이름 중복, -a , -m 불가능

태그 목록 보기

$ git tag : 예전 태그부터 표시

$ git log : 최신 커밋부터 표시

$ git show v1.0.0 : Annotated 태그의 여러 정보 표시(태그 이름, 태그 작성자 이메일, 태그 시각, 태그 메시지)

태그 삭제

$ git tag -d v1.0.0 : 태그만 삭제

깃 브랜치

버전 관리를 수행하던 일련의 파일 집합을 통째로 복사해 독립적으로 다시 개발을 진행하는 개념

- 여러 개발자가 타인을 신경 쓰지 않고 동시에 다양한 작업을 할 수 있게 만들어 주는 기능
- 브랜치를 통해 하나의 프로젝트를 여러 갈래로 나누어서 관리

브랜치 병합(merge)

독립된 브랜치에서 마음대로 소스 코드를 변경하여 작업한 후 원래 버전과 합칠 수 있음

브랜치와 병합

- 브랜치 사용 장점 : 저장소에서 다른 브랜치에는 영향 없이 새로운 기능을 개발하거나 버그를 수정, 새로운 아이디어를 안전하게 실험 가능
- 브랜치 병합 : 브랜치와 브랜치를 합치는 수행

기본 브랜치

main : 저장소 생성 시 처음 만들어지는 브랜치(예전에는 master)

[이름 설정]

$ git config - -global init.defaultBranch main : 기본 설정으로 수정

$ git branch -M newBranch : 이미 생성된 저장소의 브랜치 이름 수정

브랜치와 HEAD

- 브랜치 : 커밋 사이를 가볍게 이동할 수 있는 포인터
- HEAD : 작업 중인 브랜치의 최신 커밋을 가리키는 포인터 / 필요에 따라 특정 커밋이나 브랜치를 가리키도록 헤드 이동 가능(checkout,switch 명령 사용)
- 결과 표시 (HEAD → main) : main은 마지막 커밋을 가리키고, HEAD는 현재 작업 브랜치인 main을 가리킨다는 의미

처음 커밋 → main 브랜치는 생성된 커밋을 가리킴 / 커밋이 계속 발생 → main 브랜치는 자동으로 가장 마지막 커밋을 가리킴

새로운 브랜치 생성

$ git branch bname : 단순히 생성하고 HEAD가 이동은 안함

$ git switch -c bname 또는 $ git checkout -b bname : 생성하고 새 브랜치로 HEAD 이동도 수행

브랜치 확인

$ git branch : 브랜치 목록 보기 / **녹색* 이면 현재 브랜치

$ git branch -v : 브랜치마다 마지막 커밋ID와 메시지도 함께 표시

생성된 새로운 브랜치로 이동

$ git switch [bname] 또는 $ git checkout [bname] : HEAD를 지정한 브랜치로 이동

$ git switch- 또는 $ git checkout- : HEAD를 이전 브랜치로 이동

분리된(detached) HEAD

HEAD가 현재 브랜치(마지막 커밋)가 아닌 그 이전 커밋을 가리키는 상태

$ git checkout HEAD~ : 현재 브랜치에서 마지막 커밋 이전 커밋으로 이동

브랜치 삭제

명령 branch에서 옵션 -D 또는 -d를 사용 / 아직 병합되지 않은 브랜치라면 강제로 삭제하기 위해 옵션 -D를 사용

- $ git branch [-d | —delete] [branchName] : 지정한 branchName(이미 병합된)을 삭제
- $ git branch -D [branchName] : 지정한 branchName(병합되지 않더라도)을 삭제

브랜치 병합 여부 보기

현재 작업 브랜치는 일반적으로 - -merged 브랜치

- $ git branch - -merged : 현재 작업 브랜치를 기준으로 병합된 브랜치 목록 표시
- $ git branch - -no-merged : 현재 작업 브랜치를 기준으로 아직 병합되지 않은 브랜치 목록 표시
- $ git branch - -merged branchName : 인자인 branchName 브랜치를 기준으로 병합된 브랜치 목록 표시
- $ git branch - -merged branchName : 인자인 branchName 브랜치를 기준으로 아직 병합되지 않은 브랜치 목록 표시

$ git branch -h : 도움말 보기

{15} - 내용정리