

# Version Control (Git)

ABC Program 5조

Mentor 정현준

2021/01/03



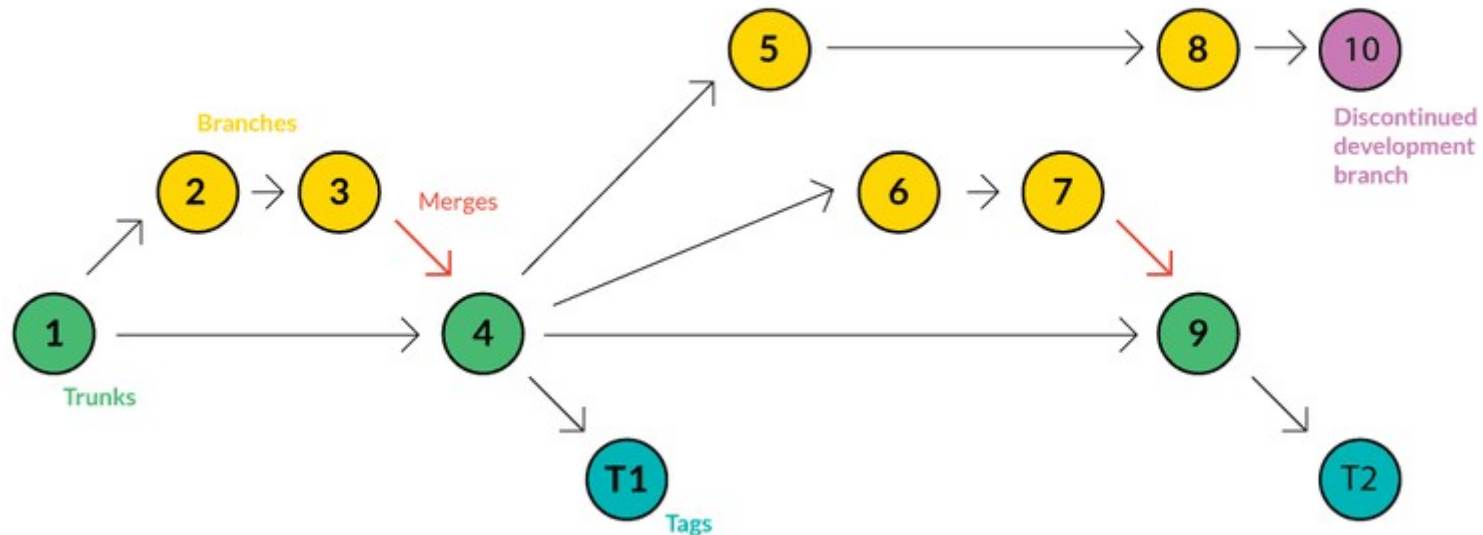
---



# Version Control

- Version Control System (VCS)

What is "version control"?



# 분산형 vs 중앙형

---

- 중앙 집중식 버전 관리 시스템 (CVCS)
  - 원격 저장소에서 최신 버전의 파일만 내려 받아 사용합니다.
  - 새로운 버전 추가를 위해서 무조건 원격 저장소에 추가해야 합니다.
  - 대표적으로 SVN이라는 소프트웨어가 있습니다.
- 분산형 버전 관리 시스템 (DVCS)
  - 최신 버전의 파일 뿐만 아니라 과거 이력을 포함한 저장소의 모든 데이터를 복제합니다.
  - 복제 후, 로컬에서 자유롭게 작업이 가능하고 원격 저장소에 문제가 생겨도 로컬을 통해 복구할 수 있습니다.
  - 대표적으로 Git, Mercurial이라는 소프트웨어가 있습니다.

# Git

---



대충 Git을 쓰고 있고, 어떻게 쓰는건지 알고 있지만,  
어떻게 Git이 작동하는건지 원리는 모른다는 얘기.

# Repository (저장소)

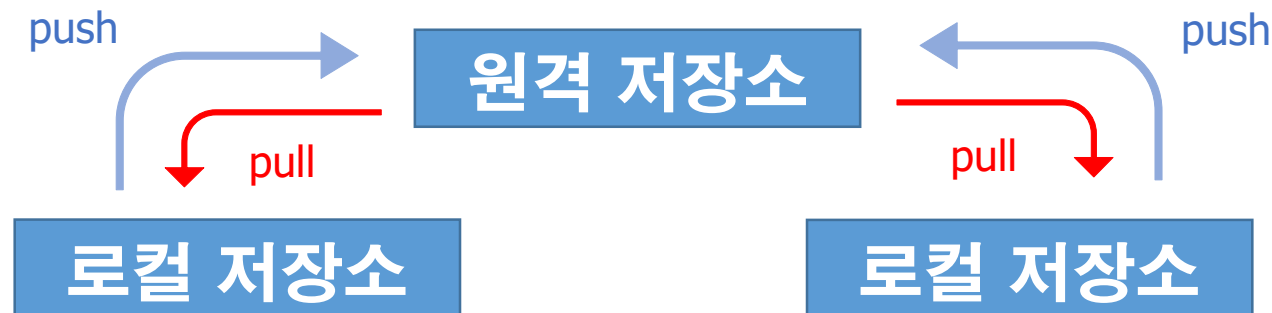
---

- 로컬 저장소 (Local repository)

코딩과 문서화가 일어나는 개인 전용 저장소. (보통 자신의 컴퓨터)

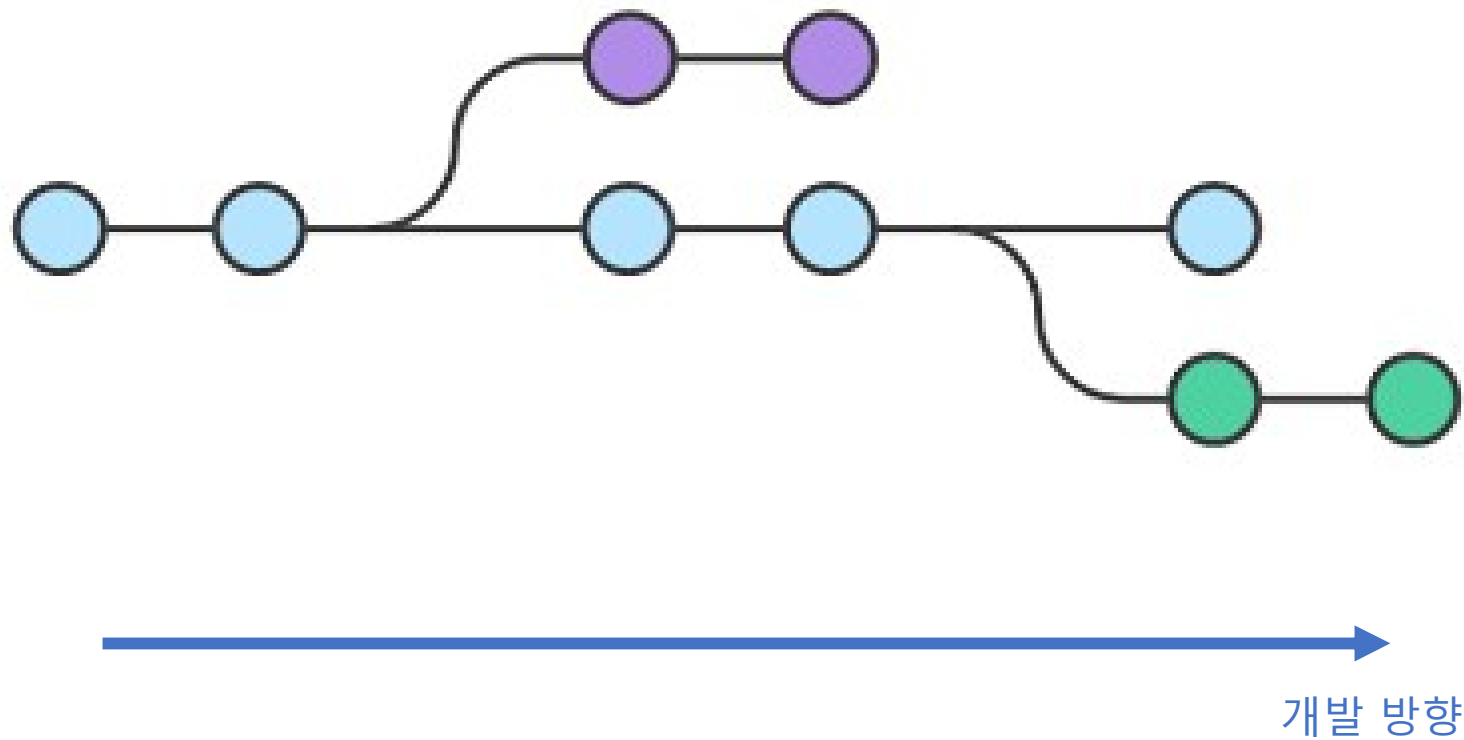
- 원격 저장소 (Remote repository)

여러 사람이 함께 공유하기 위해 전용 서버에서 관리되는 저장소.



# Branch

---



# Snapshot

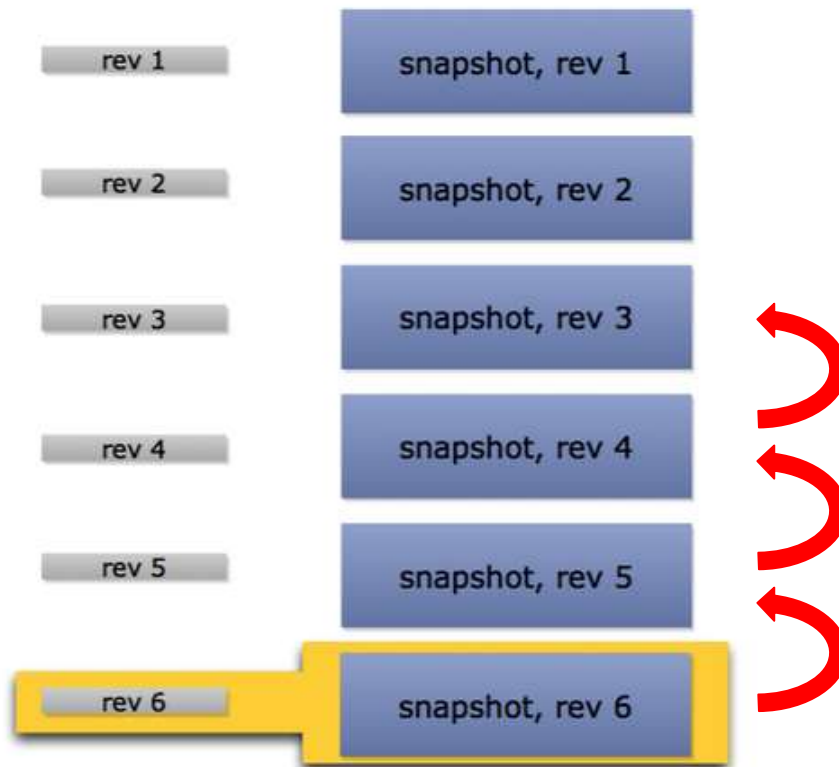
---

- Git의 데이터 저장 모델
- File = Blob, Directory = Tree
- Commit 당시의 파일들, 폴더들, 커밋들, 디렉토리 구조등을 그대로 저장한 것이 “snapshot”.
- 즉, 가장 상위의 추적이 가능한 Tree가 Snapshot.



# Snapshot

---

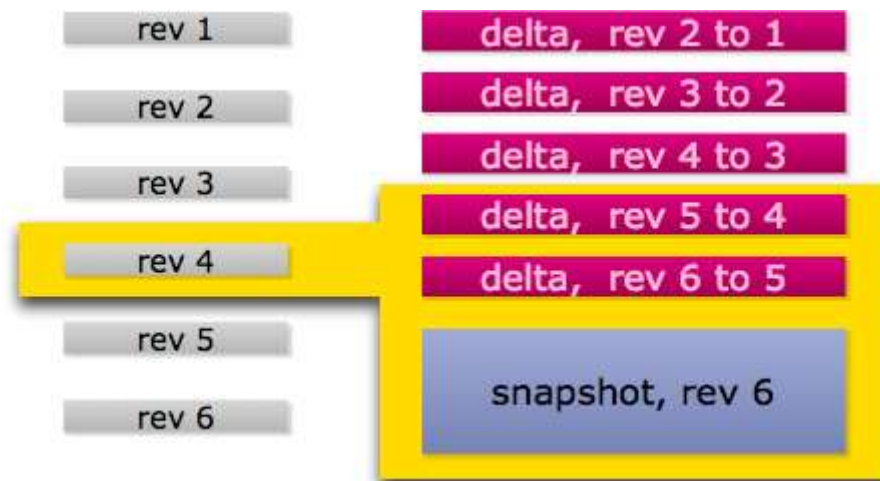


각 commit별 snapshot을 추적해  
불러오면 이전 버전에 commit  
했던 내용들이 복구 된다.

# Delta

---

- 모든 버전을 snapshot으로만 저장하게 되면 비효율적
- 두 snapshot 사이의 차이점(diff)을 Delta라고 합니다.
- 전체를 저장하는 snapshot과 달리 delta는 차이점만 저장합니다.



# SHA-1 Hash

---

- Hash란?

어떠한 알고리즘 (Hash function)을 통해 임의의 데이터를 고정된 길이의 데이터로 매핑하는 것

- Git은 모든 object(blob, tree, commit 등)를 SHA-1 Hash로 바꾸어 저장합니다.
- 다시 말해, 모든 Snapshot이 SHA-1 Hash로 바뀌어 .git 폴더에 저장됩니다.

```
// a file is a bunch of bytes
type blob = array<byte>

// a directory contains named files and directories
type tree = map<string, tree | blob>

// a commit has parents, metadata, and the top-level tree
type commit = struct {
  parent: array<commit>
  author: string
  message: string
  snapshot: tree
}
```

```
type object = blob | tree | commit
```

```
objects = map<string, object>
```

```
def store(object):
  id = sha1(object)
  objects[id] = object
```

```
def load(id):
  return objects[id]
```

# References (Refs)

---

- SHA-1 hash는 사람이 읽기 힘들기 때문에 reference를 사용합니다.
- Reference는 commit을 가리키는 포인터입니다.
- Object와는 다르게 reference는 mutable입니다.
- Branch는 어떤 작업 중 마지막 작업을 가리키는 Refs입니다.
- 예를 들어 master는 main branch의 마지막 commit을 가리키는 Refs.

# Git Commands - basic

---

- git help [다른 명령어]
- git init
- git status
- git add [파일 이름] / git add --all
- git commit / git commit -m "commit 내용"
- git log / git log --all --graph --decorate
- git diff [branch 이름] [비교하려는 branch 이름]
- git diff [commit SHA-1] [비교하려는 commit SHA-1]

# Git Commands - remote

---

- git clone [주소]
- git remote
- git remote add [이름] [주소]
- git push
- git pull
- git fetch

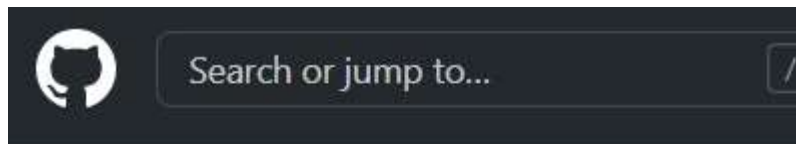
# Git Commands - branch

---

- git branch
  - git branch [생성하고자 하는 branch 이름]
  - git checkout [이동하고자 하는 branch 이름]
  - git checkout -b [생성하고자 하는 branch 이름]
  - git merge <병합할 commit 이름>
  - git merge <병합할 branch 이름>
- 
- git stash
  - git stash pop
  - .gitignore

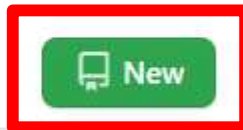
# Assignment

- 앞으로의 수업을 위해 Github에 Repository를 만들어 봅시다.



Repositories

Find a repository...



## Create a new repository

A repository contains all project files, including the revision history. [Import a repository.](#)

### Repository template

Start your repository with a template repository's contents.

No template ▾

Owner \*

with1015 ▾


Repository name \*


ABC\_Program ✓

Great repository names are short and memorable. Need inspiration? H



# Assignment

☒  **Public**  
Anyone on the internet can see your code.

☐  **Private**  
You choose who can see your code.

Initialize this repository with:  
Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description of your project.

☐ **Add .gitignore**  
Choose which files not to track from the repository.

☐ **Choose a license**  
A license tells others what they can and cannot do with your code.

[Create repository](#)

with1015 / ABC\_Program

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

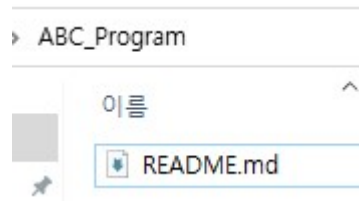
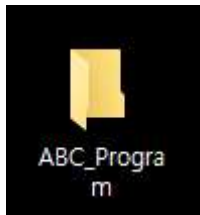
```
echo "# ABC_Program" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/with1015/ABC_Program.git
git push -u origin main
```

위 주소를 꼭 카카오톡을 통해 보내주세요!

# Assignment

---

- Github 서버에 만든 원격 저장소와 로컬 저장소를 연결하고 파일을 올려봅시다.



README.md 내용은 아무거나!

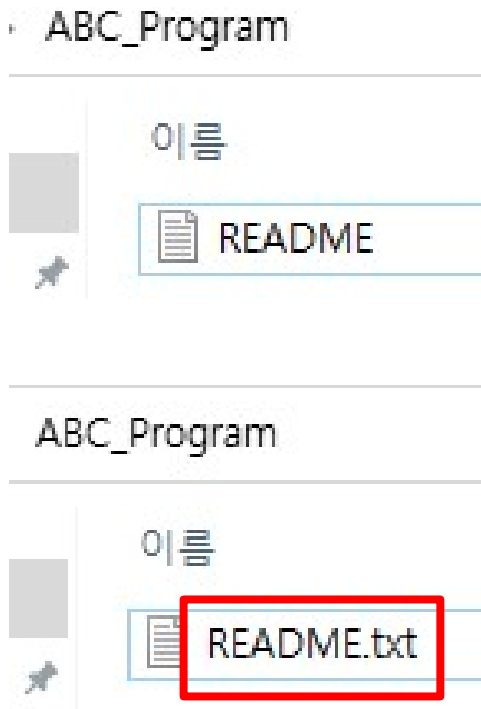
- Git bash를 열어서 아래 스크립트를 입력해줍니다.

```
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/with1015/ABC_Program.git
git push -u origin main
```

# Assignment

---

- 파일 확장자가 보이지 않는 경우 (Window 10 기준)



아래 블로그 참고

<https://jsix.tistory.com/992>

메모장에서 내용을 써 주시고 txt 확장자를 이름 바꾸기를 통해 md 확장자로 바꿔 주시면 됩니다.

# 참고 자료

---

Git에 대한 간단한 소개 (노마드 코더)

<https://youtu.be/YFNQwo7iTNc>

한국어로 된 Pro Git 책

<https://git-scm.com/book/ko/v2>

# Thank you

