



git



- Git 의 정의 및 개념
- Git Bash에서 명령어 사용법
 - add, commit, branch, switch, push, pull, fetch 등
- GitHub 활용법

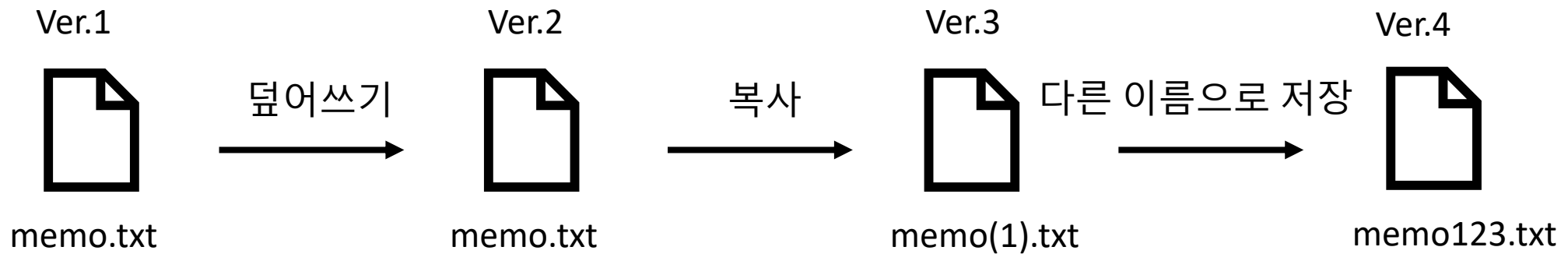


- 리눅스 개발자, 리누스 토발즈가 리눅스 버전 관리를 위해 개발
- 형상 관리 도구 (Configuration Management Tool)
- 분산형 관리 시스템
- 병렬 작업



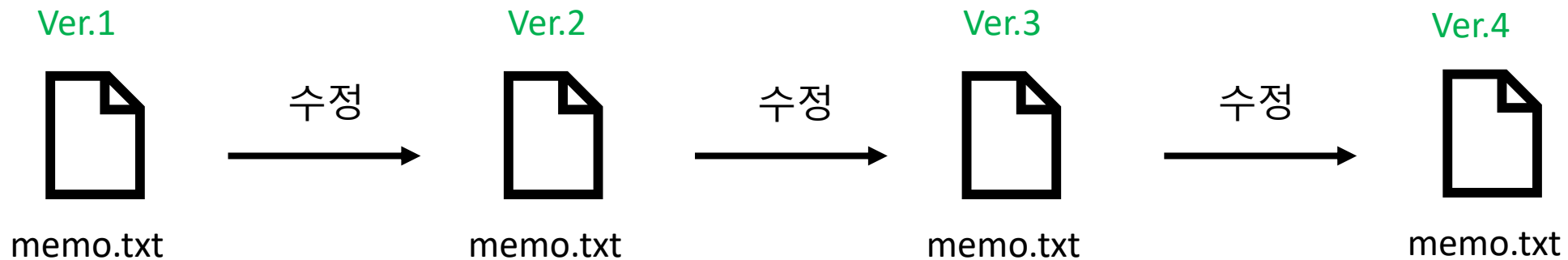
보통의 파일 관리

> 파일 복사, 덮어쓰기, 다른 이름으로 저장 등



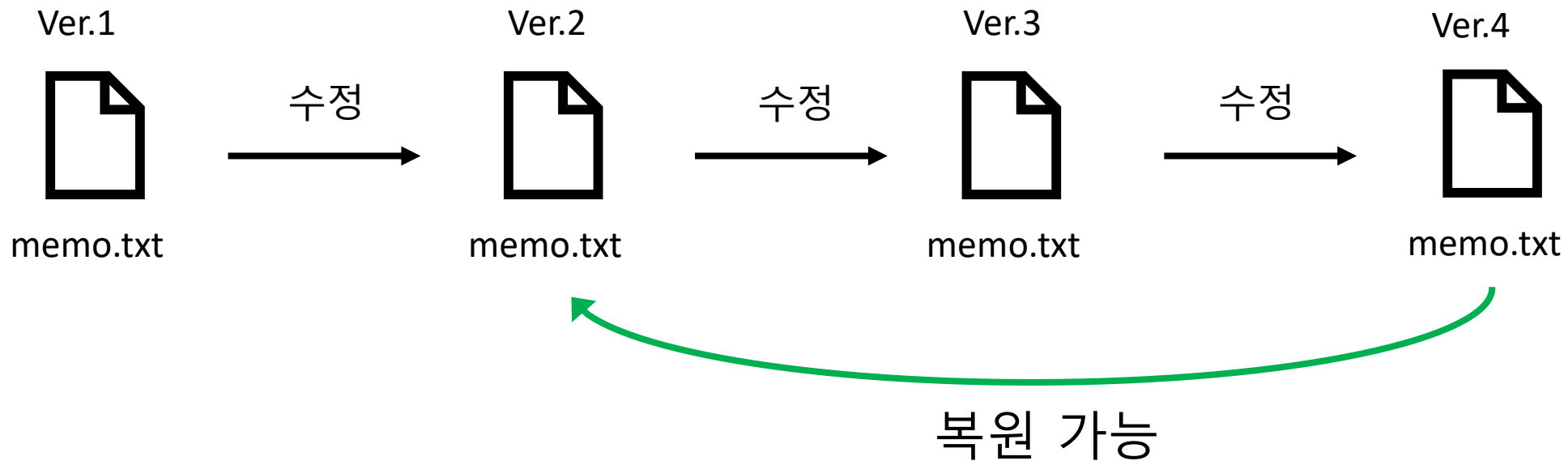
최종의..최종의..최종의.. 진짜 최종 제발 최종

형상 관리 도구 (Configuration Management Tool) ≈ 버전 관리 시스템 (Version Control System)

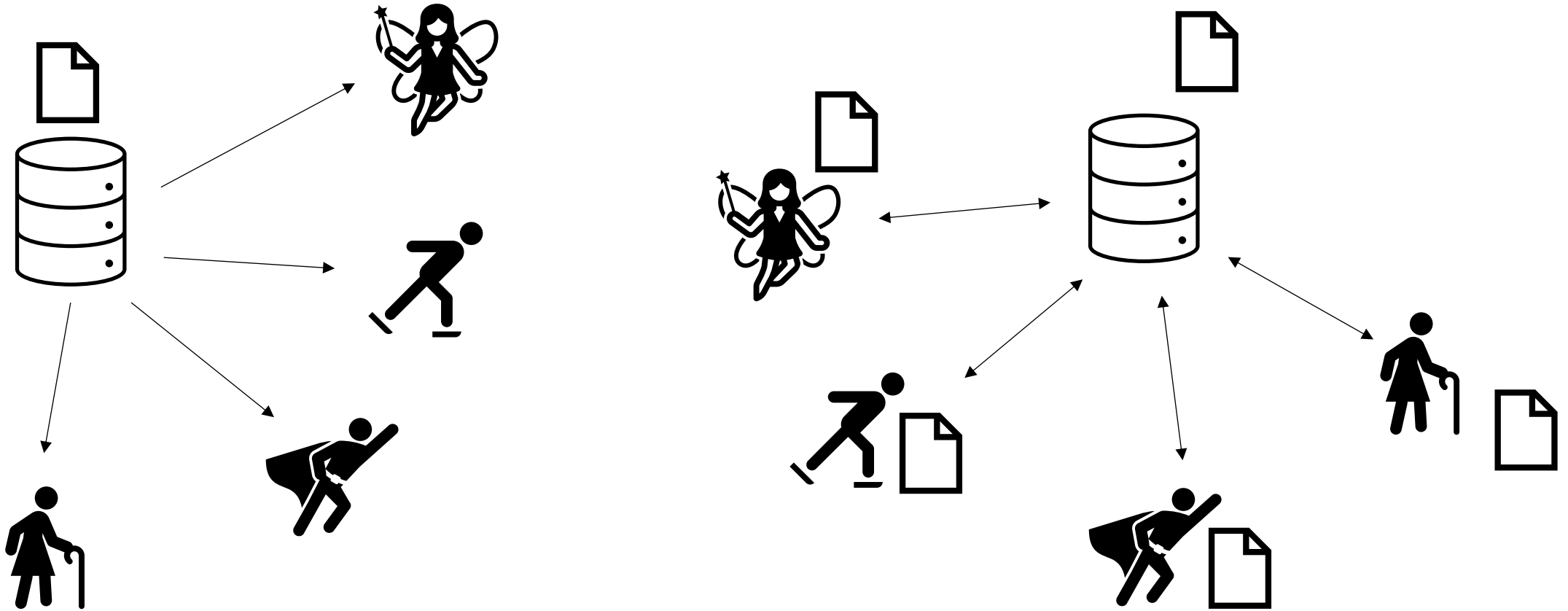


> 전체 파일을 복사하는 것이 아니라 수정된 내용을 기록

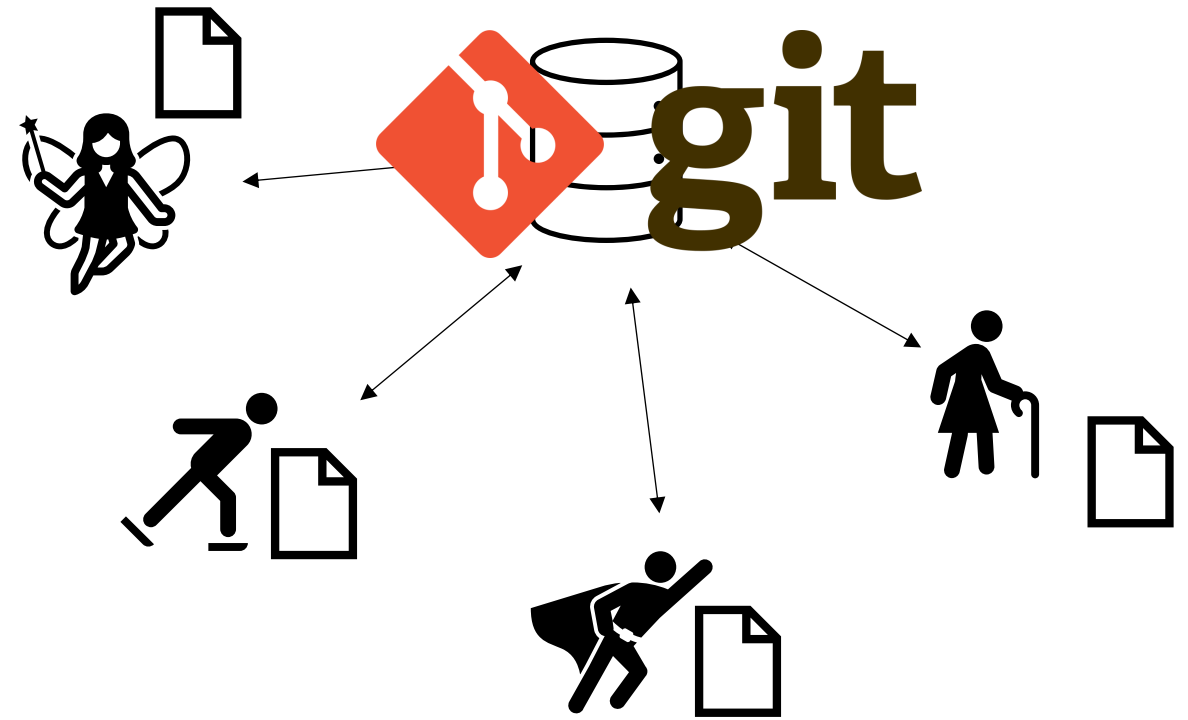
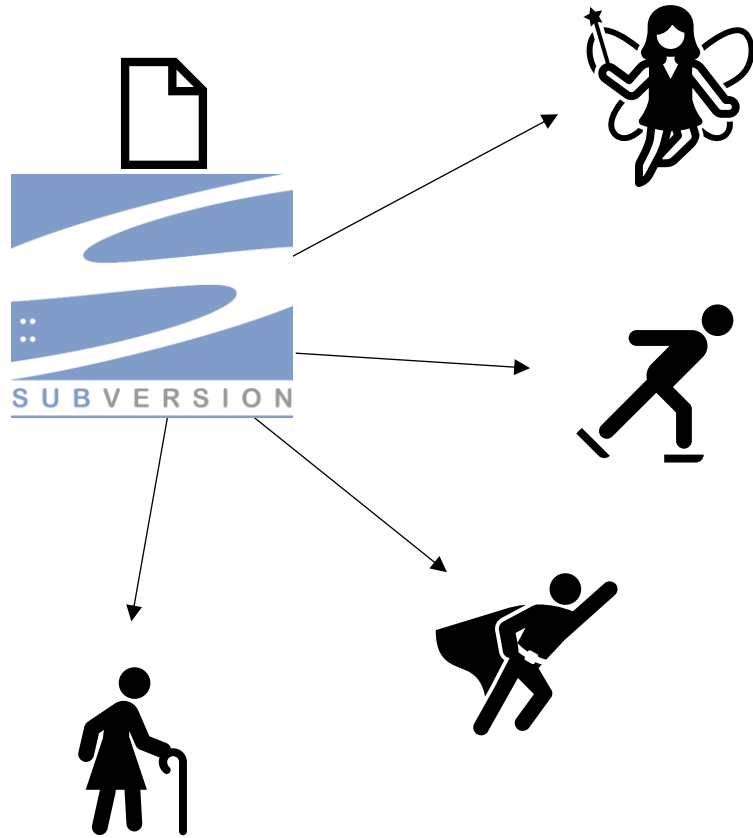
형상 관리 도구 (Configuration Management Tool) ≈ 버전 관리 시스템 (Version Control System)



분산형 관리 시스템 / 병렬 작업



분산형 관리 시스템 / 병렬 작업



Git 설치하기

Git 설치하기 (window)

<https://git-scm.com/>



Download for Windows

Click [here to download](#) the latest (2.46.0) 64-bit version of **Git for Windows**. This is the most recent [maintained build](#). It was released **22 days ago**, on 2024-07-29.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

Using winget tool

Install [winget tool](#) if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version **2.46.0**. If you want the newer version, you can build it from [the source code](#).

Git 설치하기 (mac)

<https://brew.sh/ko/>



터미널에서 실행하여 설치

Homebrew설치 후 명령어 `brew install git` 입력

Git 설치확인

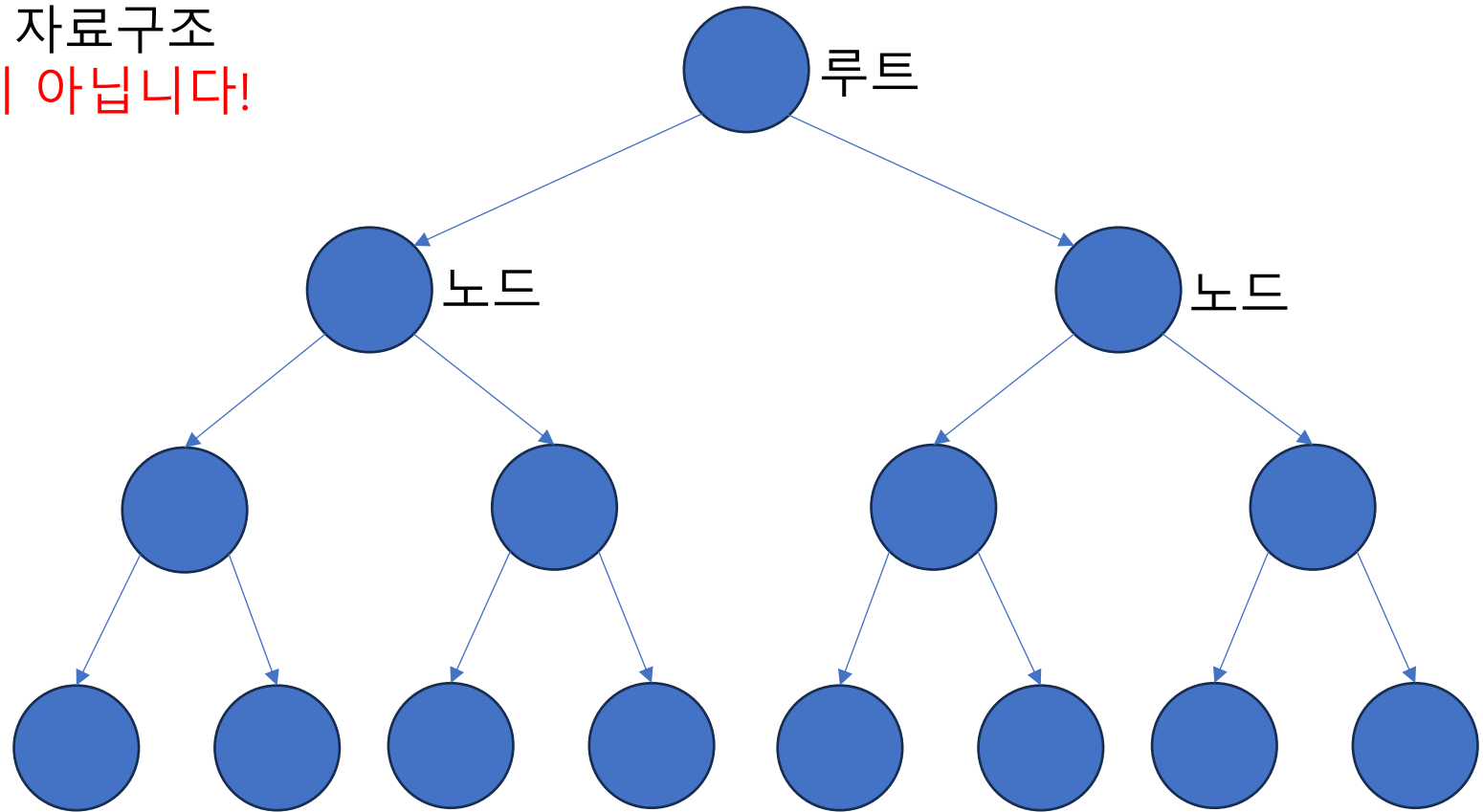
- 설치 확인
 - (window) cmd 실행
 - (mac) 터미널 실행
- git --version

```
C:\Users\shg02>git --version  
git version 2.46.0.windows.1
```

폴더 구조 이해

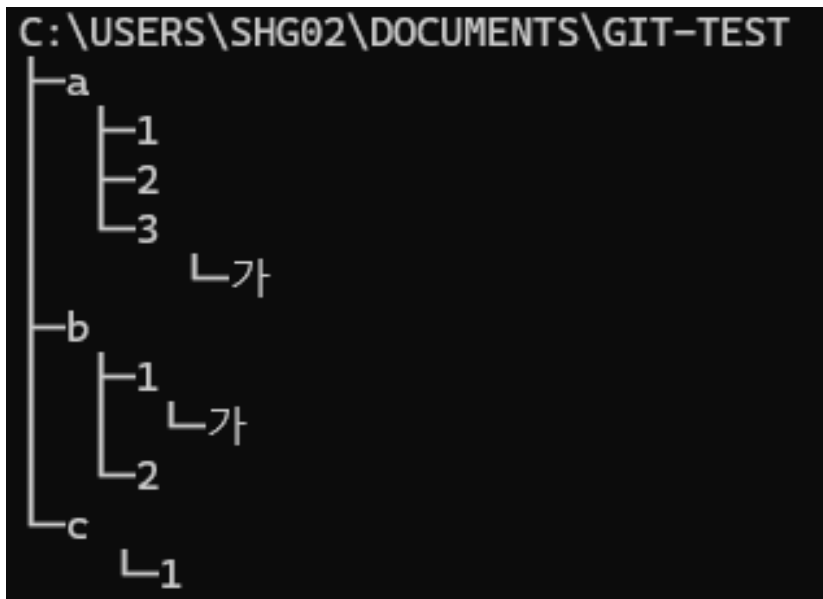
폴더 구조의 이해

- 트리구조
 - 자료구조에서 사용하는 용어로 계층적 관계를 나타내는데 사용하는 비선형 자료구조
 - => 우리는 자료구조를 하려는게 아닙니다!
- 앞으로 개발을 진행하면서 폴더 구조에 대한 이해를 돕기 위한 형태
- 루트가 프로젝트 소스코드를 담기 위한 최상위 폴더
- 루트 폴더 하나가 프로젝트 단위



폴더 구조의 이해

- 프로젝트에서 폴더 구조는 매우 중요
- 루트가 되는 폴더는 Git 저장소와 연결됨
- 현재 작업하는 파일의 폴더가 어디인지 꼭 확인하는 버릇이 필요



- 프로젝트 폴더 구조 예시
 - 루트 폴더: git-test
 - 하위 폴더: a, b, c
 - a 하위 폴더 : 1, 2, 3
 - b 하위 폴더 : 1, 2
 - c 하위 폴더 : 1
 - a > 3 하위 폴더 : 가
 - b > 1 하위 폴더 : 가


폴더 구조 확인하기

- (window) cmd 또는 git bash 실행
- (mac) 터미널 실행
- 폴더 구조 확인 명령어(구조를 볼 폴더의 부모 폴더에서)
 - tree 폴더명
- 명령어 실행이 안된다면?
 - (mac) brew install tree
 - (window) <https://gnuwin32.sourceforge.net/packages/tree.htm> 접속
 - 중간 Download의 Binaries에 zip 파일 다운
 - 압축해제하면 bin폴더 안에 tree.exe가 존재
 - C:\Program Files\Git\usr\bin에 tree.exe파일을 옮겨 넣어 줌

• Binaries

Zip

41328

 tree.exe

GUI & CLI

GUI

- GUI란?
- Graphical User Interface의 약자.
- 그래픽 사용자 인터페이스로, 사용자가 아이콘, 버튼, 메뉴, 창 등 시각적 요소를 마우스나 터치로 조작하여 컴퓨터와 상호작용하는 방식.
- Ex) 파일 복사
→ 마우스로 파일을 클릭 → 우클릭 → "복사" → 다른 폴더에 가서 "붙여넣기"
- 직관적이고 쉬움.

CLI

- CLI란?
- CLI는 Command Line Interface의 약자로 터미널 창에서 텍스트 기반으로 명령어를 입력하여 컴퓨터와 상호작용하는 방식을 말함.
- CLI를 사용하는 것이 처음에는 어려울 수 있음
- 이 어려운 것을 해결하기 위해 GUI(Graphic User Interface)형태의 프로그램도 존재
 - 예) 소스트리
- 하지만 작업속도와 Git의 모든 기능을 사용하기 위해서는 CLI를 사용하는 것이 좋음
- CLI에 먼저 익숙해 진 후 GUI를 사용해도 무관

CLI 명령어

- (window) git bash 실행
- (mac) 터미널 실행
- 코드를 저장할 루트 폴더를 생성(루트 폴더와 원격저장소가 연결)
- 자주 사용하는 명령어
 - pwd (print working directory) : 현재 나의 위치 출력
 - ls (list) : 현재 위치 폴더에 있는 모든 파일/폴더 검색
 - ls -l (long) : 상세 정보까지 보기
 - ls -a (all) : 숨김파일도 보기
 - clear : 터미널 화면을 깨끗하게 지움.
 - mkdir (make directory) : 새 폴더 만들기
 - touch : 새 파일 만들기.

- cd (change directory) : 폴더 위치 변경 (다른 폴더로 이동)
 - .
→ 현재 폴더를 의미. (자기 자신)
ex) ./파일명 → 현재 위치의 파일 실행
 - ..
→ 현재 폴더의 바로 위 폴더 (부모 디렉토리)
ex) cd .. → 한 단계 상위 폴더로 이동
 - ~ (틸드)
→ 현재 로그인한 사용자의 홈 디렉토리를 의미. (home)
- * 파일, 폴더, 이름 지을 때 **주의할 점.**
- 공백(space bar) 대신에 언더스코어(_) 혹은 하이픈(-)을 사용해서 단어 조합.
 - 한글 제발 쓰지 마세요! 영어, 대소문자를 사용하고 숫자도 상관없음.

Git Bash 리눅스 명령어 연습하기!

- Git Bash를 처음 켜었을 때 위치가 어디인지 찾아보기 (dir)
- 윈도우 상에서 바탕화면에 MyRepo 폴더 생성하기
- Git Bash에서 cd를 사용하여 MyRepo 폴더로 이동하기
 - Git Bash에서 복사 & 붙여넣기는 마우스 우클릭을 이용
 - 또는 Ctrl + Insert (복사), Shift + Insert (붙여넣기) 사용
- MyRepo 폴더 안에 file1.txt, file2.txt 라는 빈 파일 만들기.

원격저장소

레포지토리 (Repository, Repo.)

- Git에 의해 관찰되고 있는 폴더
- 즉, Git으로 버전 관리되고 있는 코드 저장소 전체를 의미.

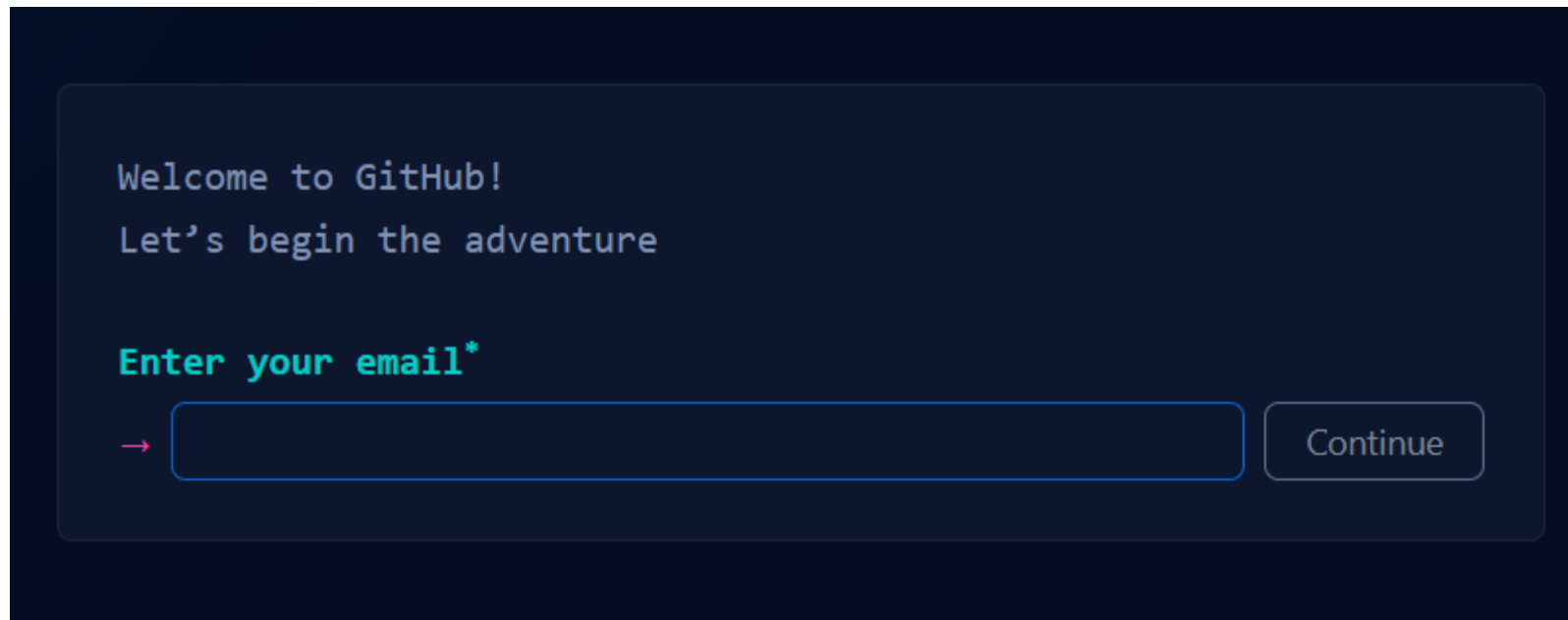
source > github >	
이름	수정한 날짜
.git	2023-09-21 오후 5:04
A.txt	2023-09-21 오후 4:02
README.md	2023-09-21 오후 5:04



source > github > .git >		.git 검색	
이름	수정한 날짜	유형	크기
hooks	2023-09-18 오후 4:42	파일 폴더	
info	2023-09-18 오후 4:42	파일 폴더	
logs	2023-09-18 오후 4:44	파일 폴더	
objects	2023-09-21 오후 6:00	파일 폴더	
refs	2023-09-18 오후 4:57	파일 폴더	
COMMIT_EDITMSG	2023-09-21 오후 5:04	파일	1KB
config	2023-09-21 오후 3:44	파일	1KB
description	2023-09-18 오후 4:42	파일	1KB
FETCH_HEAD	2023-09-21 오후 6:00	파일	1KB
HEAD	2023-09-21 오후 3:45	파일	1KB
index	2023-09-21 오후 5:04	파일	1KB
ORIG_HEAD	2023-09-21 오후 3:46	파일	1KB
packed-refs	2023-09-21 오후 3:44	파일	1KB
sourcetreeconfig.json	2023-09-21 오후 6:04	JSON 파일	1KB

그러므로! GitHub 회원가입!

- <https://github.com/>
- 회원가입에 사용한 이메일과 닉네임을 꼭 적어두기! 나중에 필요해요~! 🙄



Welcome to GitHub!
Let's begin the adventure

Enter your email*

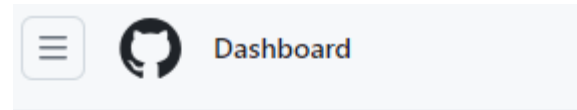
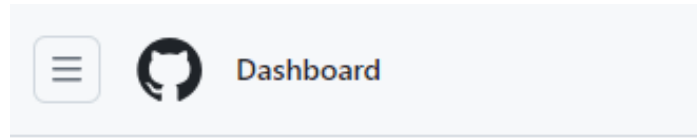
→

Continue

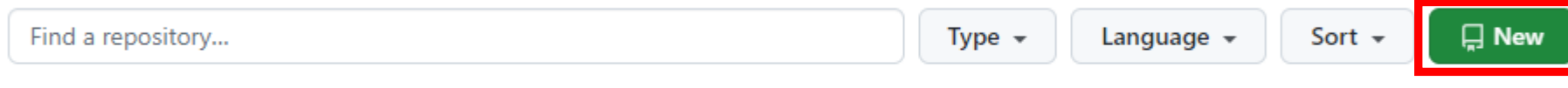
Git 설정

- (window) cmd 또는 git bash 실행
- (mac) 터미널 실행
- **설정확인** : `git config --global --list`
- **이름등록** : `git config --global user.name "프로필 이름"`
 - 예) `git config --global user.name "codingon"`
- **메일등록** : `git config --global user.email "이메일 주소"`
 - 예) `git config --global user.email "codingon@gmail.com"`
- 예외) default 브랜치가 main이 아니라면
 - `git config --global init.defaultBranch main`

레포지토리 생성



메인화면에서



나의 저장소 리스트 화면에서

레포지토리 생성

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

Repository name *

/ 저장소 영문명

Great repository names are short and memorable. Need inspiration? How about **legendary-octo-goggles** ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private


You choose who can see and commit to this repository.

Create repository

원하는 저장 방식을 선택

원격저장소와 내 폴더 연결

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

 Set up in Desktop

or

HTTPS

SSH

[https://github.com/\[username\]/git-test.git](https://github.com/[username]/git-test.git)



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 "# git-test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/[username]/git-test.git
git push -u origin main
```



ctrl + v 복사


원격저장소와 내 폴더 연결

- (window) git bash 실행
 - (mac) 터미널 실행
1. 위에서 만든 프로젝트 루트 폴더로 이동
 - 예) `cd MyRepo`
 2. Git 과 연결(명령어 차례대로 입력)
 - `git init`
 - `git remote add origin https://github.com/이름/저장소명.git`

```
MINGW64 ~/Documents/git-test
$ git init
Initialized empty Git repository in C:/Users/shg02/Documents/git-test/.git/

MINGW64 ~/Documents/git-test (main)
$ git remote add origin https://github.com/[redacted]/git-test.git
```

명령어 살펴보기

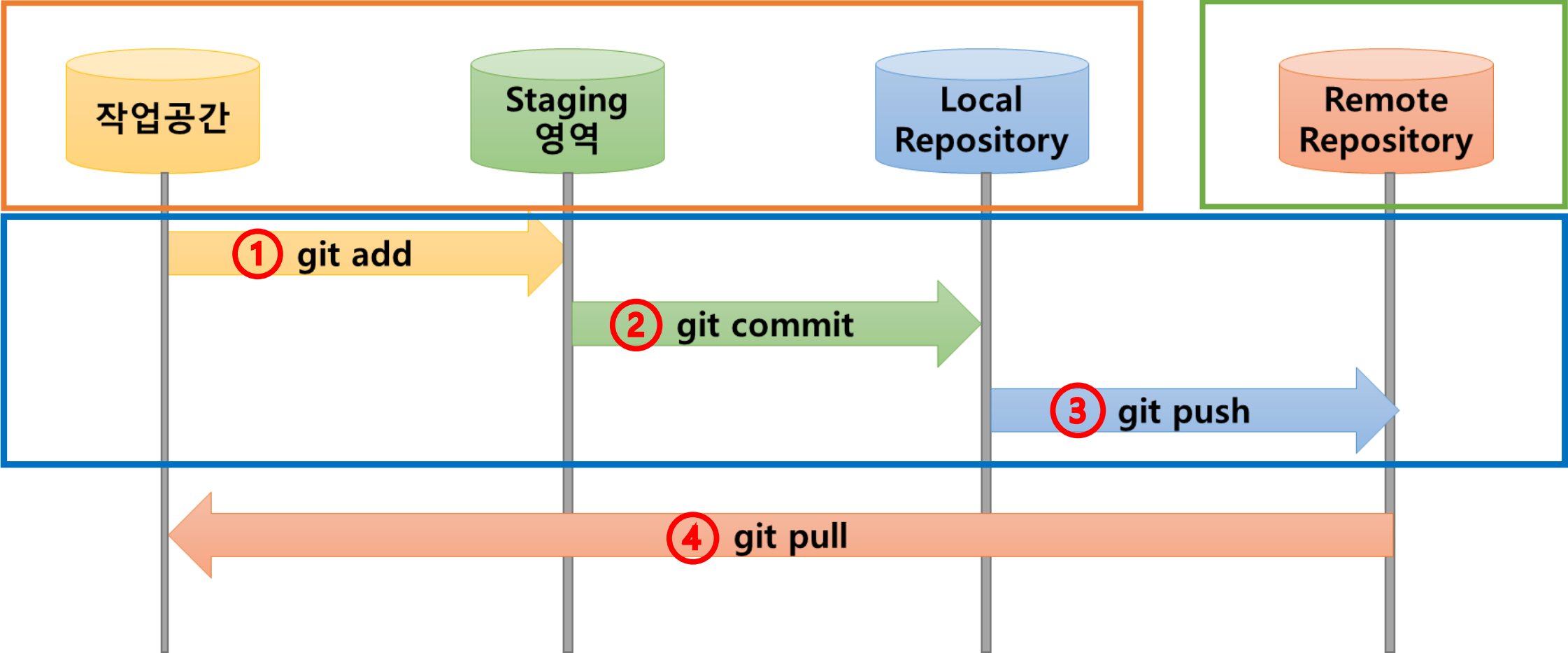
- git init
 - Git 로컬저장소가 생성됨
 - main 브랜치 생성
 - 해당 명령어 실행 후 폴더에 .git 폴더가 숨김폴더로  .git
 - 숨김 폴더 확인하기
 - (window) 탐색기에서 보기 -> 표시 -> 숨김항목 체크
 - (mac) Finder에서 Cmd + Shift + .
- git remote add origin <https://github.com/이름/저장소명.git>
 - 위 초기화된 .git폴더에 원격저장소 연결
- 원격저장소를 변경하고 싶으면 위 숨김폴더인 .git폴더 삭제 후 다시 git init과 git remote add origin 저장소 주소.git을 입력하면 됨
- 정상적으로 연결이 되었다면 폴더명 뒤에 main이 보여짐

```
~/Documents/git-test (main)
```

원격저장소에 올리기

PC 저장소 (Repository)

온라인(GitHub) 저장소



코드를 왜 올려?

- 코드 백업 : 작업한 코드를 안전하게 저장
- 협업 용이 : 팀원들과 쉽게 코드를 공유하고 함께 작업 가능
- 버전 관리 : 코드 변경 이력을 남겨 언제든지 이전 버전으로 되돌리기 가능
- 충돌 방지 : 팀원들과 작업할 때 변경 사항을 병합하여 코드가 중복되거나 누락되는 현상(충돌)을 방지 할 수 있음
- 문제 해결 용이 : 문제 발생시 코드 버전 이력을 확인하여 변경 사항을 추적

원격저장소에 파일 올리기

- Git 원격저장소로 올리기 위해서는 커밋을 해야함
- 커밋이란 버전을 뜻함. 한번 커밋하면 하나의 버전이 생성됨
- **커밋 순서(하나라도 빼놓고 진행하면 안됩니다)**
 - 커밋할 파일 추가
 - `git add 파일명` or `git add .` (. 은 한칸띄고! 폴더 전체라는 의미를 가짐)
 - ex) `git add my.txt`
 - 커밋 메시지 작성
 - `git commit -m "메시지 작성"`
 - ex) `git commit -m "first commit"`
 - 원격저장소의 브랜치로 파일 올리기
 - `git push origin 브랜치명`
 - ex) `git push origin main`

(추가) Git 주요 명령어

- 스테이징의 작업 상태 확인
 - git status
 - add가 필요한 상태 = **붉은색** 텍스트
 - add가 필요 없는 상태 = **초록색** 텍스트
- 커밋 확인(최신순으로 나옴)
 - git log 또는 git log --oneline --all --graph
 - --oneline : 로그 한줄로 보기
 - --all : 모든 브랜치 로그 보기, 안쓰면 현재 브랜치
 - --graph : 그래프 형태로 보기

원격저장소에 파일 올리기

- Git 원격저장소로 올리기 위해서는 커밋을 해야함
- 커밋이란 버전을 뜻함. 한번 커밋하면 하나의 버전이 생성됨
- 커밋 순서

```
MINGW64 ~/Documents/git-test (main)
$ git add .

MINGW64 ~/Documents/git-test (main)
$ git commit -m "first commit"
[main (root-commit) 6d63845] first commit
1 file changed, 3 insertions(+)
create mode 100644 my.txt

MINGW64 ~/Documents/git-test (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 286 bytes | 286.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/[redacted]/git-test.git
* [new branch]      main -> main
```

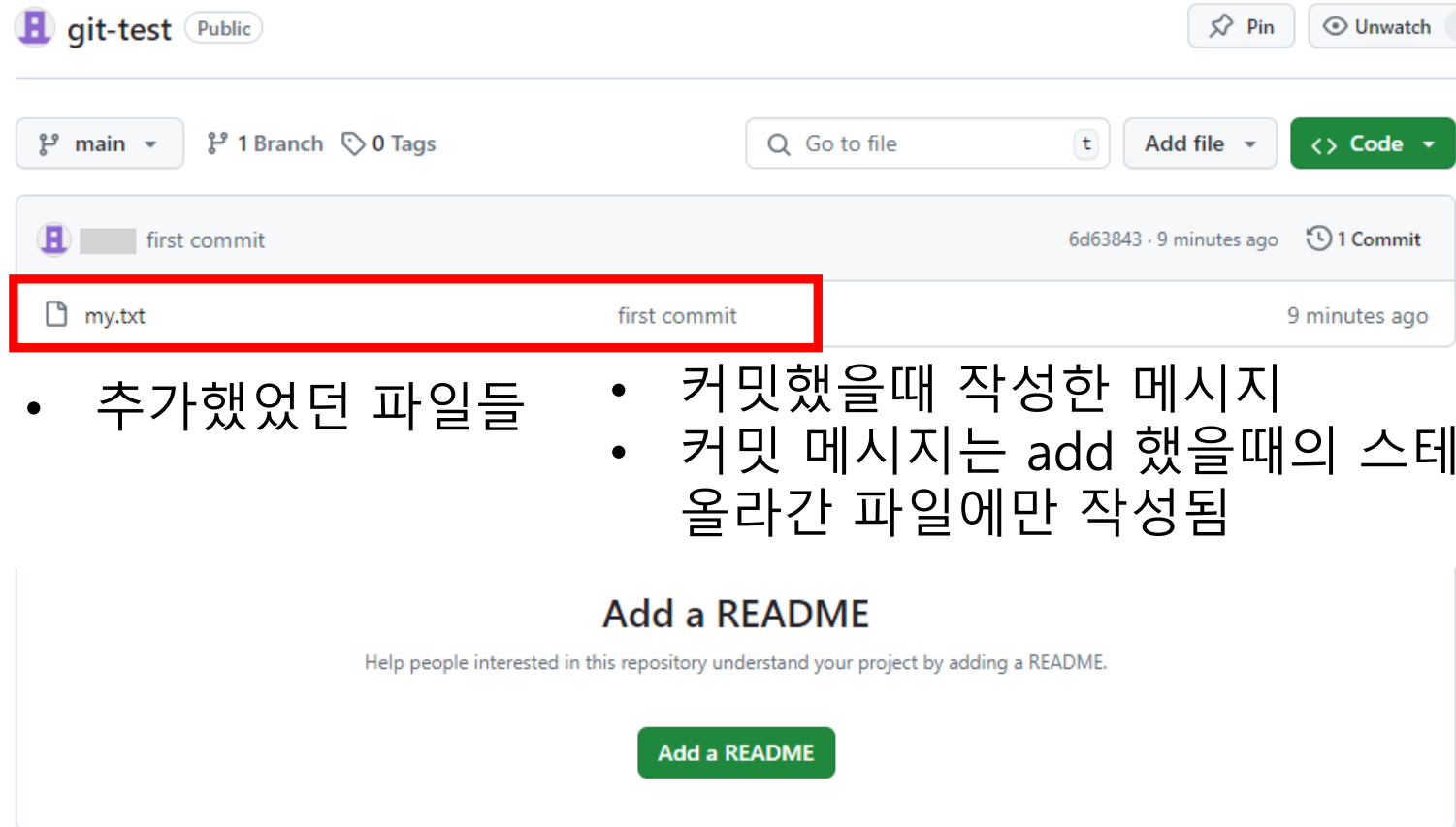
← 커밋할 파일 추가(스테이징에 등록)

← 커밋 메시지 작성(로컬저장소에 저장)

← 원격저장소의 브랜치로 파일 올리기

원격저장소에 파일 올리기

- 푸쉬 완료 후 github.com에 생성한 저장소로 이동



- 추가했었던 파일들
- 커밋했을때 작성한 메시지
- 커밋 메시지는 add 했을때의 스테이징에 올라간 파일에만 작성됨




실습. 원격저장소에 파일 올리기

1. 파일을 2개 더 생성하기

hello.txt, test.txt

2. 신규 생성한 파일에 텍스트로 글을 작성

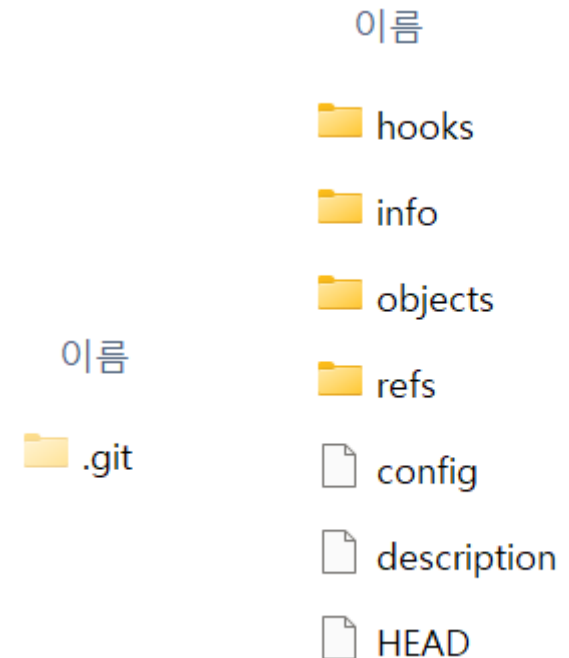
- 위 생성된 모든 파일을 원격저장소로 올려보세요

 hello.txt	실습. 원격저장소에 파일 올리기
 my.txt	first commit
 test.txt	실습. 원격저장소에 파일 올리기

실습 완료시 github에 올라간 화면 예시

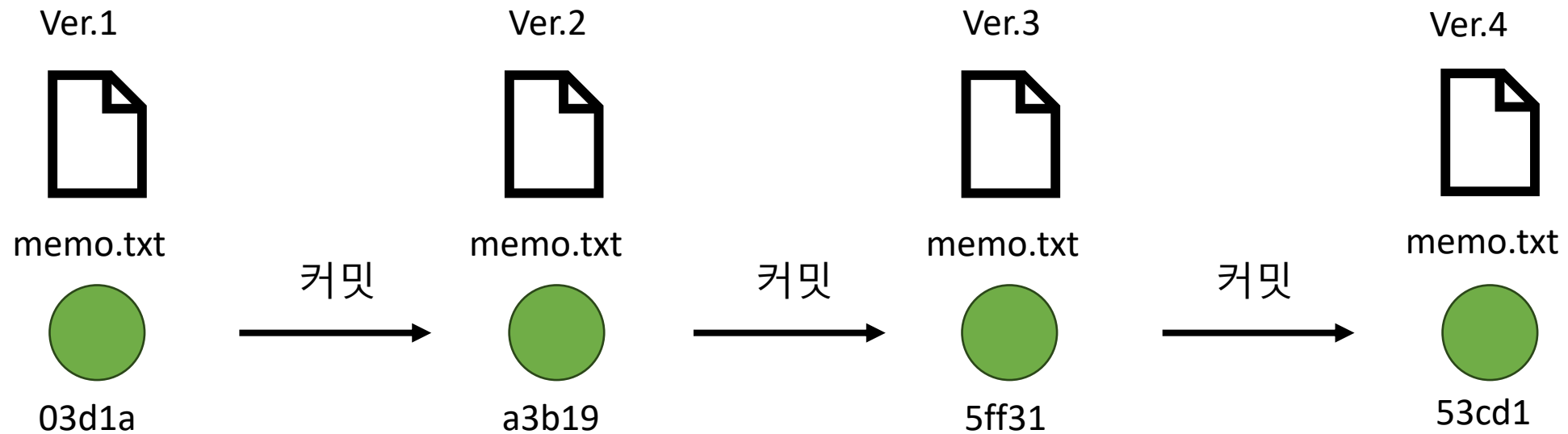
Git Repository (Local Repo)

- Git 이 관리하는 폴더
- 레포지토리에 있는 파일들에 대한 내용 변화를 Git이 즉시 감지
- Repository 생성
 - 비어있는 폴더에서 마우스 우 클릭
 - Open Git Bash here 실행
 - `git init` -> 엔터



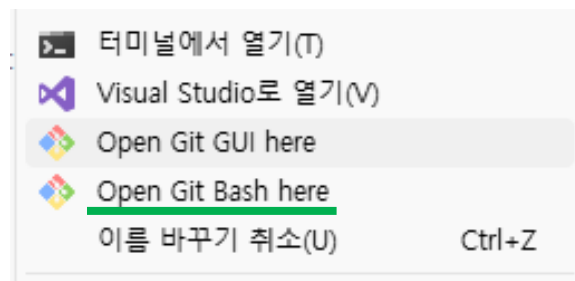
Commit Node

- Commit 을 수행하면 Node 가 생성됨
- 각 Commit Node는 고유한 ID를 가짐



실습. Git 테스트

1. 메모장에 “ABC”를 적고 Git 레포지토리에 txt 파일을 저장
2. 레포지토리에서 “우클릭 -> Open Git Bash here” 기능으로 Repo. 경로상에 GitBash 열기
3. 앞 페이지에서 학습한 명령어들을 사용하여 커밋 노드 생성
4. txt 파일을 다시 열어서 내용을 “DEF”로 바꾸고 또 커밋 노드 생성
5. 커밋 내역을 확인하는 명령어를 사용하여 2번의 커밋 내역 확인
6. 커밋 내역을 스크린샷으로 찍어서 슬랙에 댓글로

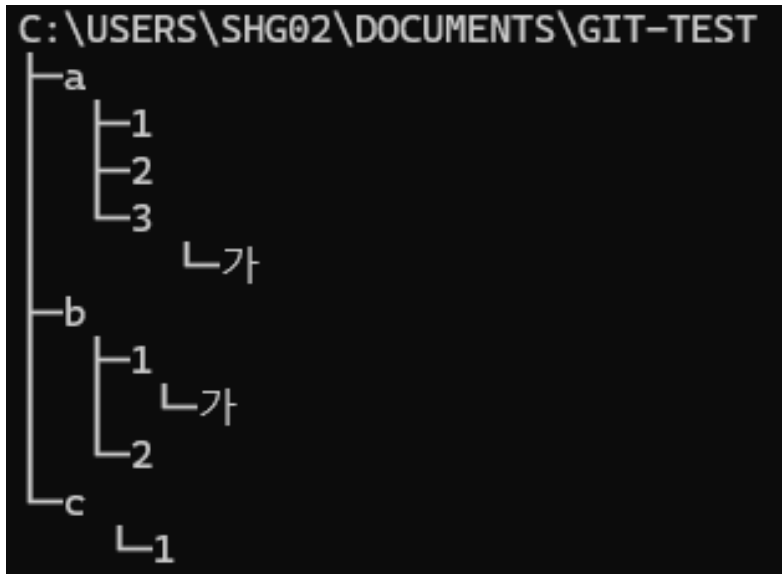


Q & A

- 파일을 신규로 제작하고 push하게 되면 원격저장소에 파일이 업로드됨
만약 기존 파일에서 코드만 수정만 한다면?
⇒ 다시 파일이 업로드 되는게 아닌 코드만 수정됨. 스냅샷 방식
- 폴더에 아무 파일도 존재하지 않는다면?
⇒ 해당 폴더는 원격저장소에 업로드 되지 않음. 적어도 하나의 파일이 존재해야함
⇒ 폴더가 올라가는게 아닌 파일이 올라가는 것이기 때문
- 중요한 정보가 담긴 파일도 Git 저장소에 올려도 되나요?
⇒ **아니요! 절대 안됨!!!!!!!!!!** .gitignore 파일을 만들어서 파일을 무시하게 해야함

실수 사례

- git init은 루트 폴더에서 해야하는데 폴더 구조를 생각하지 않아서 하위 폴더에서 또 git init을 하는 경우가 발생하여 루트폴더가 push가 안되는 사례
- 예)



git-test폴더(루트폴더)에서 git init 진행 후 하위 폴더가 많아지고 파일이 많아지게 되면서 하위 폴더인 a폴더에서 git init을 하여 또다른 원격저장소와 연결하는 경우가 있었음

⇒ 이 경우 git-test폴더에서 push를 하게 되면 오류가 뜸

⇒ git은 .git의 내용으로 원격저장소를 찾게 되는데 git-test폴더의 안에 .git이 두개가 생기게 되면서 git은 어디로 저장되어야하는지 알 수 없게됨

- 꼭 파일이 생성되는 폴더의 위치를 잘 확인하고 작업을 진행!!!

.gitignore

- 모든 파일이 전부 Git 원격저장소에 올릴 필요가 없음
- 특히 API key, DB 접속정보 등 외부에 노출하면 안되는 정보는

Git 원격저장소에 올리면 절대 안됨

- 올리면 안되는 파일들에 대한 정보를 담는 파일이 .gitignore 파일임

.gitignore

- 내용
- *.txt : 확장자가 txt로 끝나는 파일 모두 무시
- !text.txt : text.txt는 무시되지 않음
- test/ : 루트 폴더내 하위 폴더중 이름이 test폴더 파일들은 무시한다는 뜻(상대 경로)
- /test : 루트 폴더내 test폴더를 무시(절대 경로)
/ 가 프로젝트 루트 기준으로 경로 지정

```
.
├── .gitignore
├── test
│   └── b.exe
└── tmp
    └── test
        └── a.exe
```

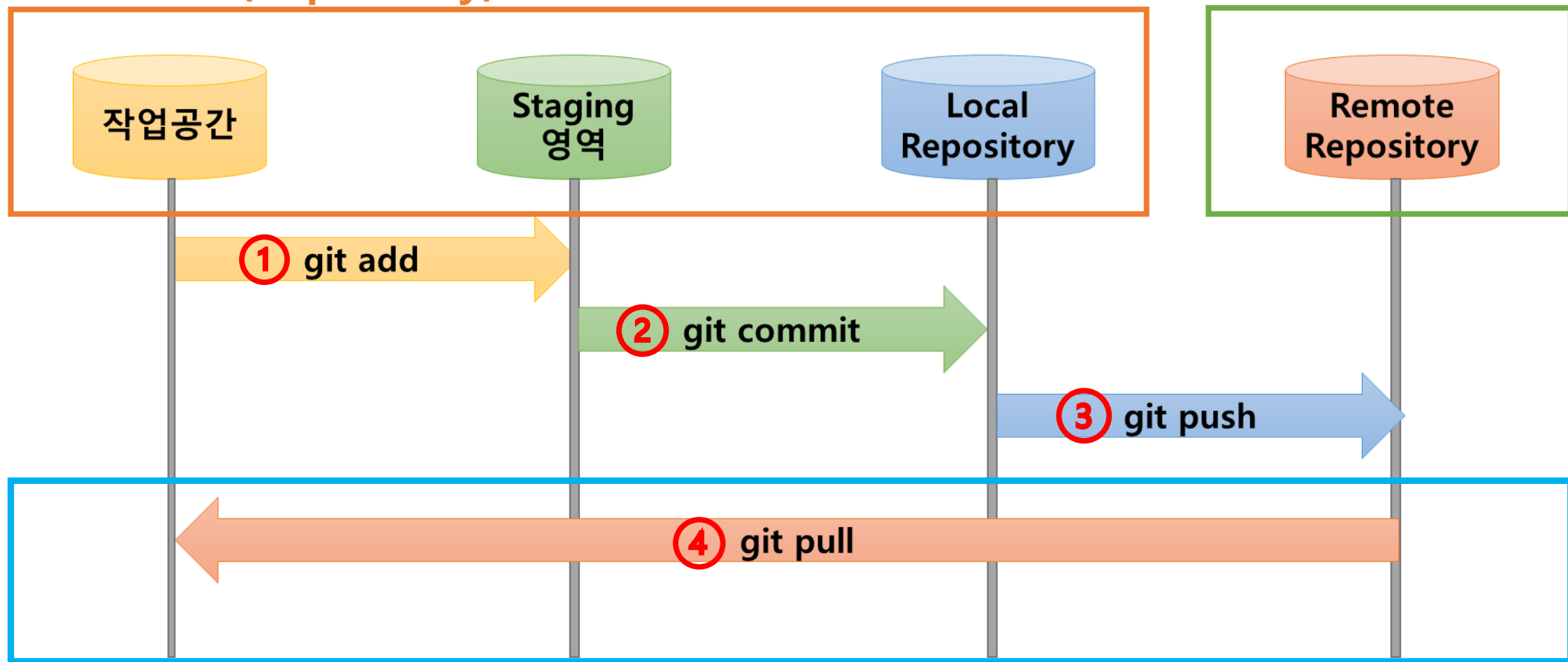
3 directories, 3 files

- 무시되는 파일 예시
- test/ : b.exe, a.exe
- /test : b.exe
a.exe 무시되지 **않음**.

원격저장소 내려받기

PC 저장소 (Repository)

온라인(GitHub) 저장소

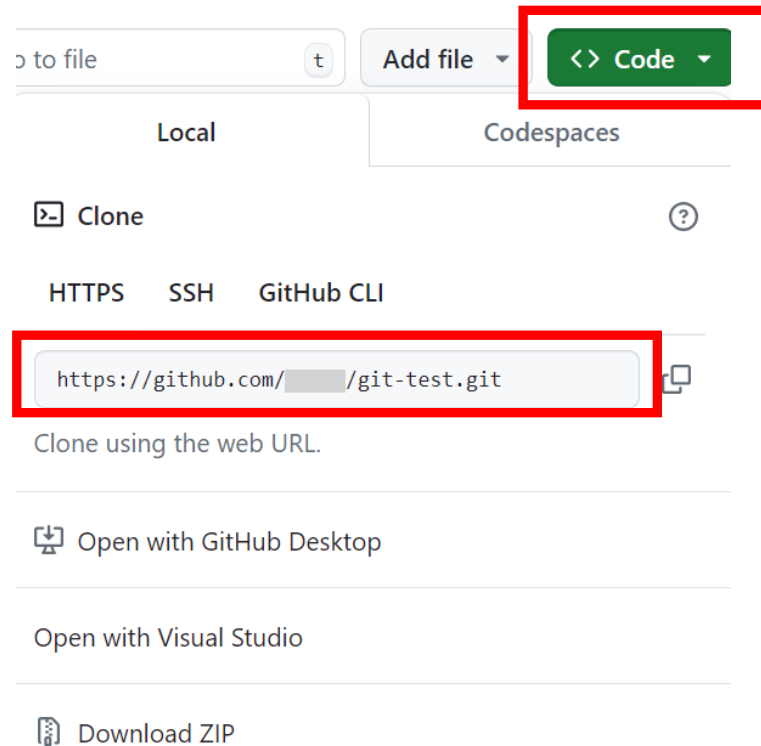


코드를 왜 받아?

- 코드 복사 : 원격저장소에 있는 코드를 내 컴퓨터에 백업 가능
- 협업 용이 : 팀원들이 작업중인 코드를 내 컴퓨터에서 작업 가능
- 최신 상태 유지: 항상 최신 버전의 코드를 가져와서 최신 상태로 작업 가능
- 빠른 시작 : 기존 프로젝트를 바로 내 컴퓨터에서 실행하고 테스트 가능
- 프로젝트 설정 : 프로젝트 처음 설정 시 필요한 파일과 폴더 구조를 한번에 가져 올 수 있음

내 컴퓨터에 로컬저장소 생성하기

- 원격저장소에 있는 프로젝트를 처음으로 내려받거나 다른 팀원의 저장소를 내 컴퓨터에 내려받을 때
- 저장소 주소 찾는 방법



내 컴퓨터에 로컬저장소 생성하기

방법1. 신규 폴더 생성 후 내려받기

- 원하는 폴더를 생성
 - mkdir 폴더명
 - ex) mkdir git-clone
- 원격저장소 복사하기(생성한 폴더로 이동 후)
 - **git clone 원격저장소 주소 .** (.은 한칸띄고! 이 점이 중요 포인트!)
 - ex) git clone https://github.com/원격저장소 주소.git .

```
MSYS ~/Documents/git-clone  
$ git clone https://github.com/[redacted]/git-test.git .
```

.git
hello.txt
my.txt
test.txt

clone이 완료되면 폴더안에 원격저장소에있던 파일들이 전부 받아져있는 것을 확인할 수 있습니다.

내 컴퓨터에 로컬저장소 생성하기

방법2. 원격저장소의 이름으로 내려받기(이름으로 폴더가 생성)

- 원격저장소 복사하기
 - `git clone` 원격저장소 주소
 - ex) `git clone https://github.com/원격저장소 주소.git`

```
MINGW64 ~/Documents/workspace  
$ git clone https://github.com/[redacted]/git-test.git
```

```
MSYS ~/Documents/workspace  
$ cd git-test/  
MSYS ~/Documents/workspace/git-test (main)  
$
```

원격저장소 생성시 작성했던 이름이 폴더명으로 생성된 것을 확인할 수 있습니다.

원격저장소 파일 내려받기

- 숨김폴더인 .git 폴더가 있는 폴더에서 명령어 실행(루트 폴더)
 - git pull origin 브랜치명
 - ex) git pull origin main

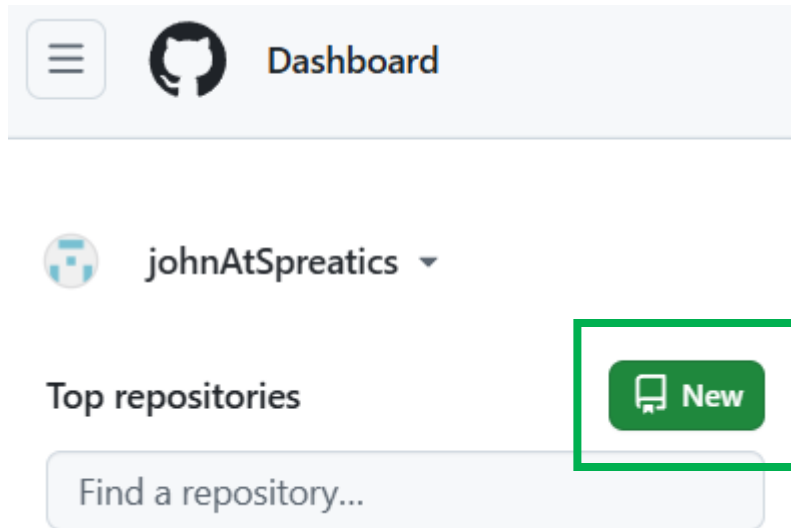
```
MSYS ~/Documents/git-test (main)  
$ git pull origin main
```

- 터미널에 git log 명령어 작성 후 HEAD 상태 확인

```
$ git log  
commit 0f30653547eb6a6f0e6150d0d5f1194c81de1a77 (HEAD -> main, origin/main, origin/HEAD)  
Author:   
Date: Tue Aug 27 09:29:00 2024 +0900  
  
pull test
```

로컬저장소와 원격저장소의 HEAD가 최신 커밋을 가리키고 있는 상태입니다.

Remote Repository 만들기




Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *

 johnAtSpreatics ▾

Repository name *

MyTestRepo

Great repository names are short and memorable. Need inspiration? How about **refactored-disco** ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

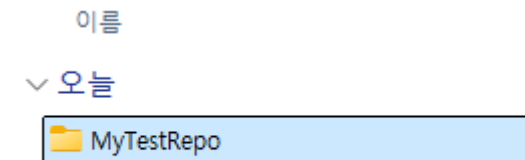
.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

clone 명령어

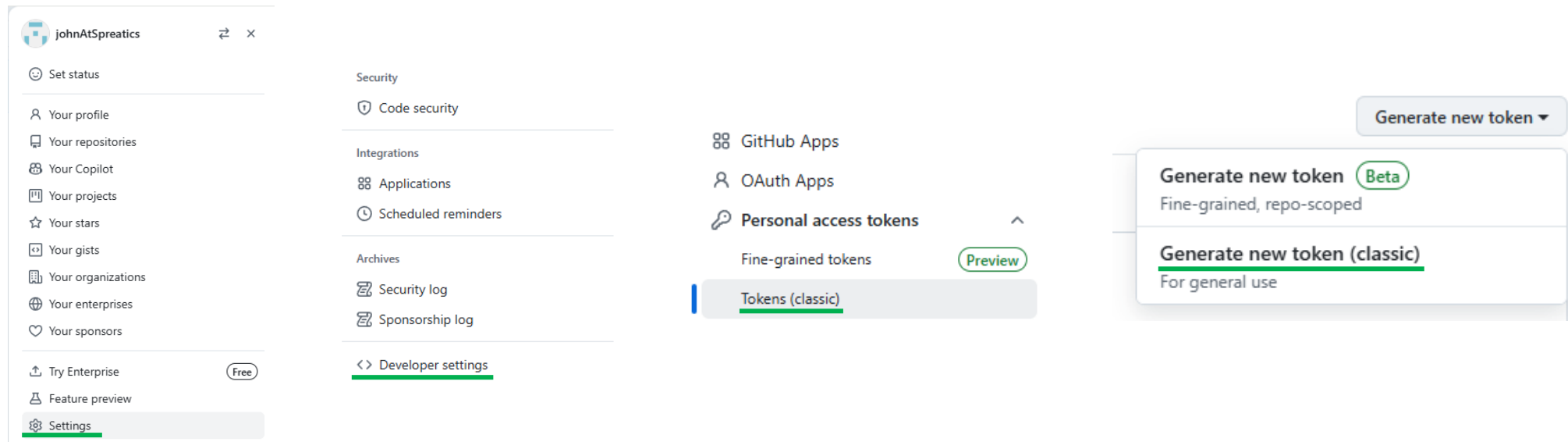
- 비어 있는 폴더에서 Git Bash를 실행
- git clone (Remote Repo.의 HTTPS 주소)
 - GitHub 로그인을 요청하는 경우도 있음
 - 단, 2차 보안이 걸려있을 경우 **Personal Access Token** 발급이 필요

```
$ git clone https://github.com/johnAtSpreatics/MyTestRepo.git
Cloning into 'MyTestRepo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```



Personal Access Token

- GitHub 접근을 위한 임시 비밀번호
- GitHub 계정 -> Settings -> Developer settings -> Personal access tokens -> Tokens (classic) -> Generate new token (classic) 으로 발급 가능



Personal Access Token

- 발급 받은 Token을 GitHub 로그인 창에서 비밀번호 대신 사용

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

office

What's this token for?

Expiration *

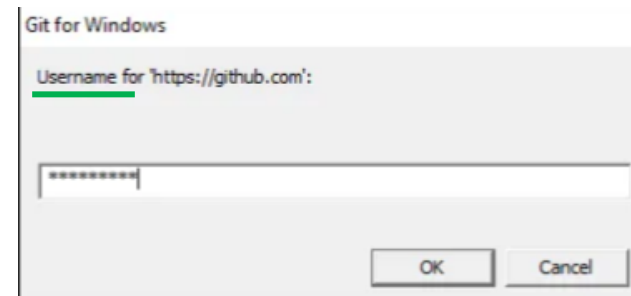
30 days ▼

The token will expire on Wed, Jan 8 2025

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events



실제 비밀번호 대신 발급 받은 Token을 사용

push 명령어

- Local Repo.의 변동 내역을 Remote Repo.에 적용
- 정확하게는 Local Repo.의 Branch 하나를 Remote Repo.의 Branch 하나와 Merge를 수행
- `git push -u origin main`
 - 현재 Local Branch를 Remote Repo. (별칭: origin)의 main Branch로 Merge 수행
 - `-u`: Remote Repo.의 별칭과 Branch를 기억하여 이후부터는 origin main을 생략하고 `git push` 만으로도 작동함
 - 단, 2차 보안(OTP 등)이 있을 경우 **Personal Access Token** 이 필요함

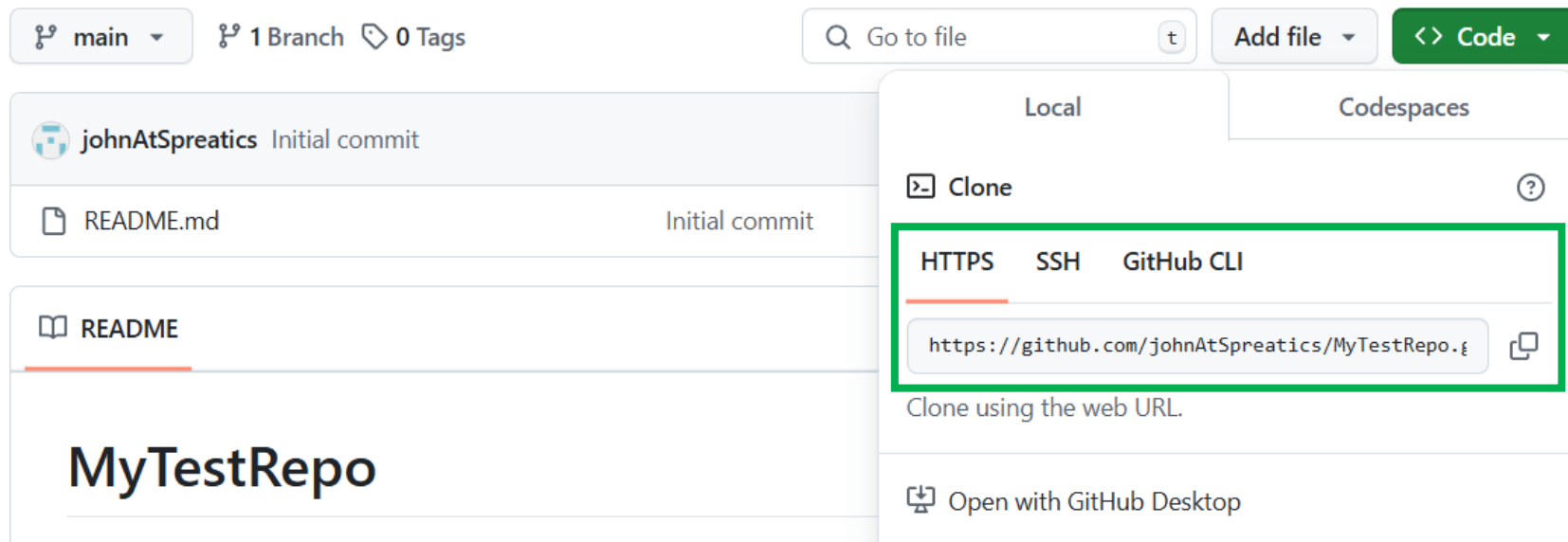
실습. clone 및 push

1. GitHub에 있는 Remote Repo. 인 MyTestRepo를 Local에 Clone
2. Clone 된 Local Repo.에 new_file.txt 파일 추가
3. Push 기능을 사용하여 new_file.txt를 Remote Repo.로 merge
4. GitHub 웹 사이트에서 MyTestRepo에 파일이 올라간 것을 확인
5. 파일이 올라간 화면을 캡처하여 슬랙 댓글로 제출

remote 명령어

- Local Repo. 에서 아래 명령어를 이용
- git **remote** add origin (GitHub HTTPS 주소)

```
git remote add origin https://github.com/johnAtSpreatics/MyTestRepo.git
```



remote 명령어

- Local Repo. 에서 GitBash로 아래 명령어를 이용
- `git remote add origin` (GitHub HTTPS 주소)

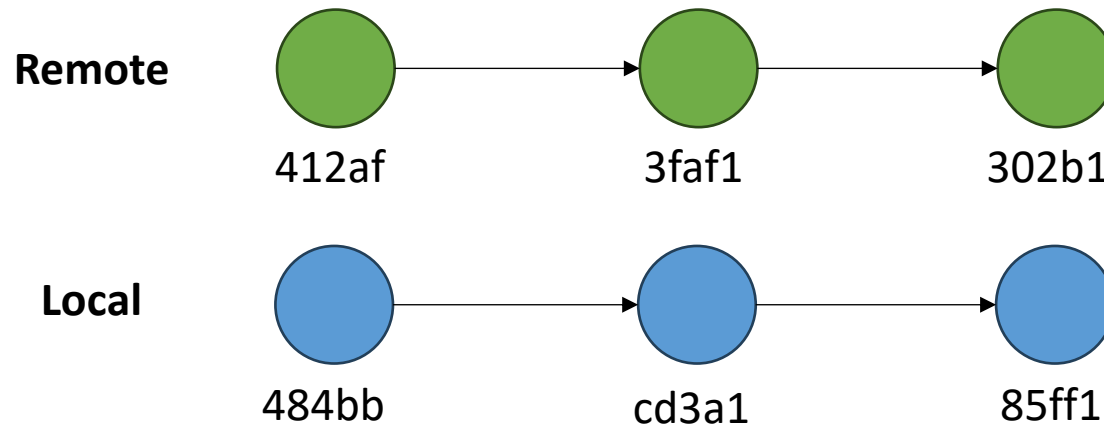
* origin 이란?

원격 저장소의 별칭으로 GitHub에서 생성한 Remote Repo.의 기본 별칭이 origin

`git remote rename origin raymond` 과 같이 별칭을 raymond 으로 변경 가능

remote 이후 push

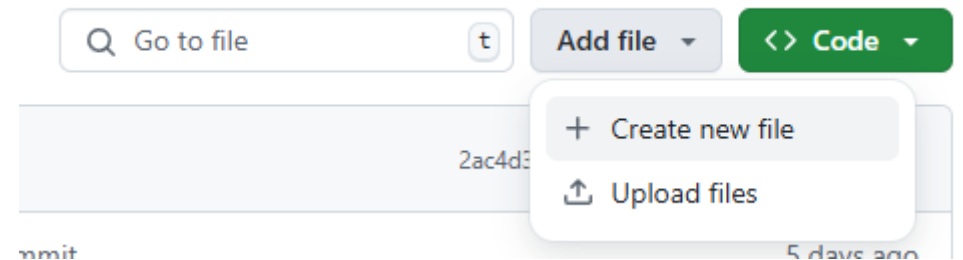
- Remote Repo.에 커밋 지점이 없다면 바로 push 가능
- readme.md 파일이 존재하거나 다른 커밋 내역이 있다면 공통된 커밋 지점이 없기 때문에 push가 불가능
- git push --force
 - Local Repo.의 상태를 강제로 Remote Repo.에 덮어쓰기



일치하는 커밋 노드가 전혀 없다면
push 및 pull 모두 불가능

실습. remote 및 pull

1. 새로운 Local Repo.를 생성
2. test.txt 파일을 추가하고 커밋
3. Remote Repo.인 MyTestRepo과 remote 연결
4. pull 시도 (당연히 오류 발생)
5. 강제로 push 실행
6. GitHub에서 Add file 기능으로 memo.txt 파일 추가
7. Local Repo.에서 fetch 및 pull 수행
8. git log를 캡처하여 슬랙 댓글로 제출



README.md

README

- Git 저장소에서 프로젝트의 개요와 사용법을 문서화
- Markdown 언어로 작성
- 프로젝트를 보는 사람들이 해당 프로젝트의 내용을 이해할 수 있도록 설명해주는 것이 일반적
- README를 작성하는 방법에 양식은 없음

Markdown 문법

- 제목 : # 1 개~ 6 개
- 기울임 : *기울임* or _기울임_
- 굵게 : **굵게** or __굵게__
- 기울임+굵게 : ***기울임과 굵게*** or ___기울임과 굵게___
- 취소선 : ~~취소선~~
- 이모지 : :smile:, :heart:, :rocket:
 - [이모지 이름 찾기](#)
- 줄바꿈은 엔터 두번을 눌러서 공백을 만들어 주기

Markdown 문법

- 인덱싱 : 순서 있는 인덱싱은 1번부터 번호를 입력해주면 됨
- +, *, - : 순서 없는 인덱싱 만들때 사용
 - 순서있던 없던간에 TAB을 사용하여 하위 인덱싱을 제작할 수 도 있음
- 하이퍼링크 : <url주소>
- 링크 : [링크이름](url주소, 옵션)
- 이미지 : ![이미지이름](이미지url주소)
- 인용 : > 인용내용, >> 중첩인용

Markdown 문법

- 코드 : `한줄코드`
- 여러줄 코드 : ```언어명```
- 수평선 : ---, ***, _
- 체크박스 : [] 빈체크, [x] 체크
- 표 :

헤더1	헤더2	헤더3
데이터1	데이터2	데이터3
데이터4	데이터5	데이터6