Git, github 스터디

2021-2 KCA

본 ppt의 자료는 git book, 한이음 ICT 멘토링을 참고하였음을 밝힙니다.

형상 관리

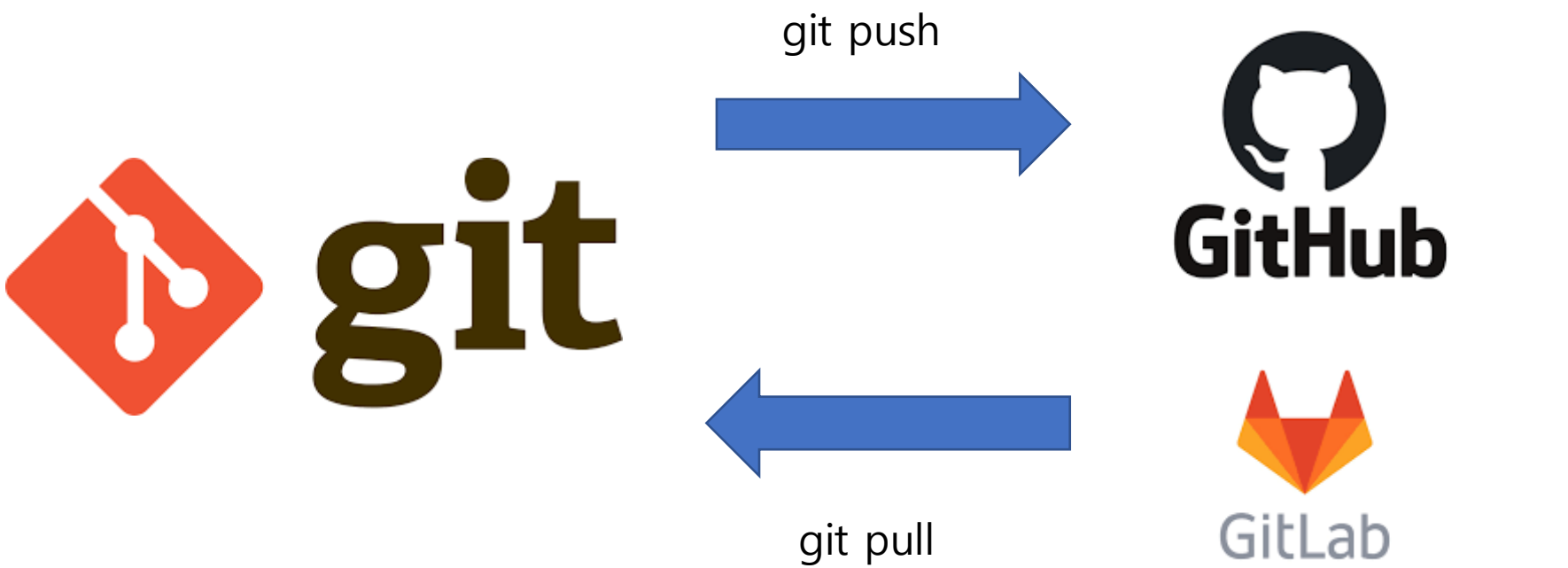
- 소프트웨어의 변경 사항을 체계적으로 추적하고 통제하는 것

- 변경 관리 < 버전 관리 < 형상 관리

형상 관리 도구 : CVS, SVN, Git, Perforce…

Git은 엄밀히 말하면 분산형 버전관리 시스템이다.

Git 저장소

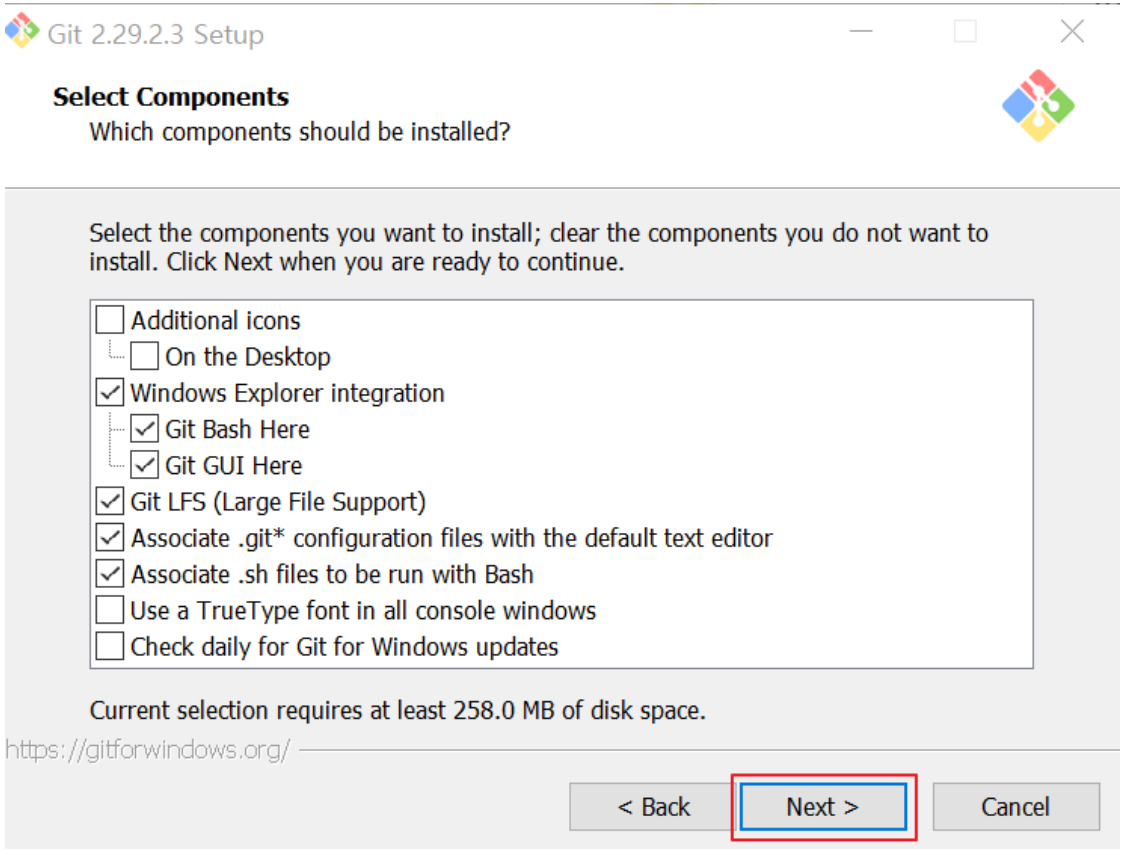


로컬 저장소

원격 저장소
== remote 저장소

Git 다운받기

https://git-scm.com/



Git Bash : git 명령어를 사용할 수 있는 Shell 창

Git Bash 실행시켜 보기!

Git 저장소 생성

1. 원격 저장소 것을 로컬 저장소에 clone
   git clone [원격 저장소 주소]

2. 로컬 저장소에 새로 만들기 init
   해당 폴더 위치에서 git init

Git 저장소라면? .git 폴더가 생긴다.

=> .git 폴더만 지우면 git history 다 삭제됨!

Git 사용자 설정

현재 Git 사용자의 이름과 이메일을 처음 설정해 주어야 한다.

git config user.name “{자신 이름}”
git config user.email “{이메일}”

git config --global : 항상 적용되는 설정 (global 없으면 이 저장소에만 적용)

* 현재 설정 확인 : git config -l 또는 git config --list

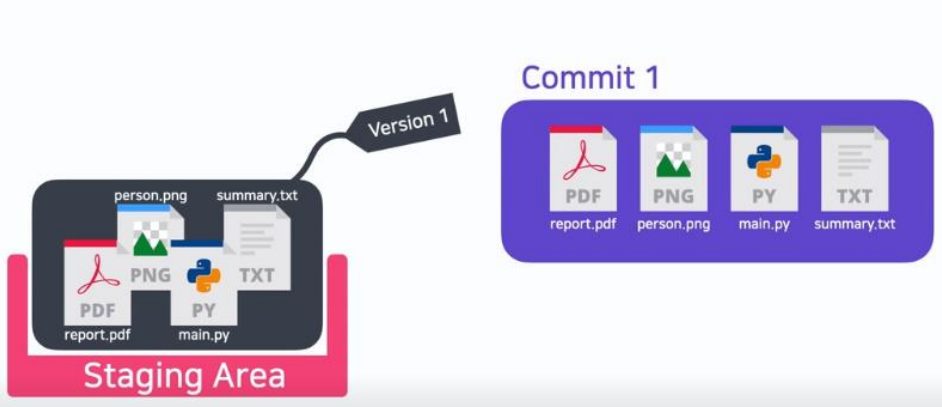
git commit

commit : git에서 특정 버전을 저장하는 것. (“스냅샷을 남긴다 “ 라 하기도 함)

1. 파일 생성 및 작성, 수정 & 저장
2. git add {파일 이름들} (git add . == 변경된 모든 파일)
3. git commit -m “커밋메세지”

(커밋 메시지는 명확하게 써 주어야 함.(무슨 기능을 추가했는가, 무슨 오류를 해결했는가 + issue 기반 개발)

git의 세 영역



1. Working Directory ==
작업 중인 폴더 (.git이 있는)

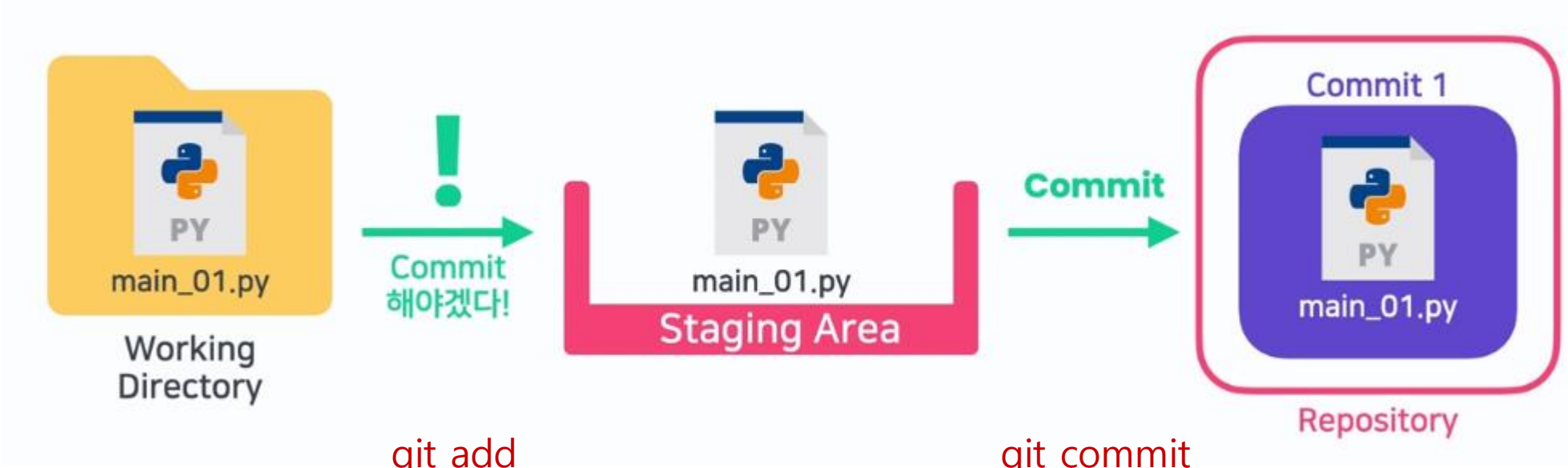
2. Staging Area ==
특정 버전(커밋)으로 관리하고 싶은 파일들

3. Repository ==
커밋들이 저장되는 영역

git add
staging area로 넘김

git commit
Repository로 넘김

commit (버전 관리) 과정



git add
staging area로 넘김

git commit
Repository로 넘김

git log :
Repository의 commit 기록을 보는 명령

git status :
Staging Area에 담긴 파일들을 보는 명령

실습!! + VSC gui

+ .gitignore

[실습] [참고] .gitignore 파일

git add . 을 해서 commit을 할 때,

개인정보(비밀번호 등)은 버전관리에 포함되면 안된다!!!!!!!! 매우 중요!!!!!

이때 .gitignore 파일을 만들고,

여기에 파일 / 폴더 명을 써 주면 Staging Area로 넘어가는 것을 막을 수 있다!

+ .env 파일

자세한 문법 & 설명 참고 : https://kyu9341.github.io/Git/2020/08/23/git_gitignore/

[주의점] git history에 대하여

- git은 history를 쌓아가는 방식으로 작동한다.

즉, 무슨 커밋을 삭제하면 삭제했다는 사실과 삭제한 것도 기록이 남는다. (commit histor랑 은 또 다름)
(git reflog 명령으로 확인 가능)

- 추가) 비밀번호 파일을 commit 했다가, 파일을 삭제한 후 gitignore에 등록하고 다시 commit 했을 때.

비밀번호가 commit history에 남아있다면, 복구가 가능하기 때문에 문제가 될 수 있다.

⇒ commit history에 없어도, git reflog history에 남았다면, 로컬에서는 복구 가능하다.

[실습] (참고) git commit

- 그냥 git commit 만 입력했을 때 이상한 입력 창이 뜬다??!

=> 리눅스의 vi 에디터 (git 만든 사람이 리누스 토르발스 라서..)

vi 에디터의 단축키를 따라야 함.

i 키 : 문자 삽입

insert 키 : 문자 삽입/수정

esc 키 : 삽입 상태에서 나오기

:wq : 저장하고 파일 닫기
:q! : 저장하지 않고 파일 닫기

: 주석 처리 (커밋 메시지로 기록 안됨.)

git config --local core.editor notepad
명령어로 메모장 에디터로 바꿀 수 있다고 한다!

commit 수정 | git commit --amend -m “수정할 메시지”

git commit --amend : vi 에디터로 이동

git commit --amend -m “수정할 메시지“ : 바로 수정

이 명령어는 commit message가 바뀐 새로운 commit을 생성한다.

새로운 파일을 add 한 후, --amend 하면 commit에 빠진 파일을 넣거나 오타를 수정할 수 있다.

[실습] git log

```
PS F:\건국대\_동아리\KCA\2021.2학기 스터디\git> git log
commit 8cde8921f9049baebe94e4e01dee3ca394501662 (HEAD -> master)
Author: Turtle-hwan <kjhwan0802@naver.com>
Date:   Thu Sep 9 14:45:17 2021 +0900

    first
```

노랑색 : commit 식별하는 ID
Author : commit 한 사람의 정보
Date : 날짜
commit message

[실습] git diff {commit_id} {commit_id}

- 두 commit 사이의 차이점

commit ID는 첫 4자리만 적어도 됨. (git log에서 보고 적자..!)

추가된 줄을 +로, 제거된 줄을 -로 나타내 준다.

- git log -p 도 동일 효과 (git log -p -2 : 최근 2개의 diff 옵션)

commit 간 이동 | git reset --{option} {commit_id}

- commit을 계속 해야 하는 이유?

⇒그 시점(버전)으로 이동할 수 있기 때문!!

```
PS F:\건국대\_동아리\KCA\2021.2학기 스터디\git> git log
commit 8cde8921f9049baebe94e4e01dee3ca394501662 (HEAD -> master)
Author: Turtle-hwan <kjhwan0802@naver.com>
Date:   Thu Sep 9 14:45:17 2021 +0900

    first
```

- 파랑 글씨 HEAD(포인터)가 가리키는 곳이 우리 눈에 소스 코드가 보이는 버전(커밋)

git reset --hard {해당 commit id} : 해당 commit으로 이동

정리

<commit 생성>

git add .

git commit -m “”

<commit 읽기>

git log

git diff

git reflog

<commit 수정>

git commit --amend -m “”

<commit 삭제(단, 기록은 남는다)>

git reset --{option} {commit_id}

여러분이 가져가야 할 것

이러 이러한 설정 / 명령어가 있었다.

다음에 git 저장소를 만들어야 할 때, github을 사용할 때 이런 게 있었다~ 정도면 충분히 검색해서 찾을 수 있음!!