

GitHub 포트폴리오

3주차. Git 기본 사용법 II

| | |
|----------------------|--|
| Git 저장소 및 설정 방법 | Windows 에서 Git 저장소 방법과 환경변수 설정 방법 |
| | 사용자 이름과 이메일을 설정하여 Git의 기본 설정을 완료하고 이를 검증하는 방법 |
| Git의 기본 명령어 | git init : 새로운 Git 저장소 초기화 |
| | git add : 파일을 스테이징 영역에 추가 |
| | git commit : 변경 사항 커밋 |
| 효과적인 로컬 저장소 관리 | 계정 불필요 & 다운로드 후 설치 |
| | Git 디렉토리 구조, 워킹 디렉토리, 스테이징 영역, 저장소의 관계 |
| | git status, git log : 파일 상태 변화를 추적하고 관리하는 방법 |

학습개요



학습목표

- 01.** Git 저장소의 개념과 로컬 저장소 관리 방법을 설명할 수 있다.
- 02.** Git 명령어로 원격 저장소와 로컬 저장소 간의 데이터 동기화를 수행할 수 있다.

학습내용

- Git 저장소의 개념
- 저장소 데이터 동기화

Git 저장소의 개념

저장소 데이터 동기화

- 이번 주차 학습은 Git의 저장소 개념과 로컬 저장소 관리, 그리고 원격 저장소와의 데이터 동기화 방법에 중점을 둠
- Git은 소프트웨어 개발에서 버전 관리를 위해 널리 사용되는 도구로, 기본 명령어인 push와 pull을 익히는 것은 협업 및 프로젝트 관리에 있어서 필수적인 능력임
- 이를 통해 실무에서 Git을 효율적으로 활용할 수 있는 기초를 다질 수 있음
- 또한, 실습을 통해 명령어 사용에 대한 실질적인 경험을 쌓아 문제 해결 능력을 향상시킬 수 있음

Git 저장소의 개념

UNIT

Git 저장소 개념

Git 저장소

파일의 변경 이력을 추적할 수 있는 버전 관리 시스템

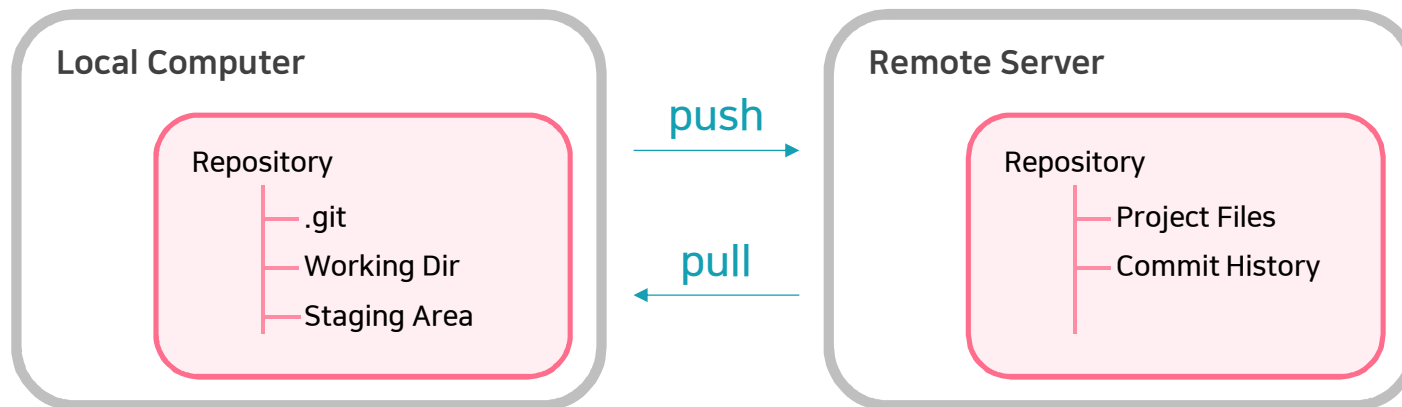
- 로컬 저장소 : 사용자의 로컬 컴퓨터에 위치
- 원격 저장소 : Github와 같은 서버에 위치

Git 저장소 개념

| 구분 | 로컬 저장소 | 원격 저장소 |
|---------------|---------------------------|---|
| 위치 | 개발자의 로컬 컴퓨터에 위치 | 인터넷이나 네트워크 상에 있는 중앙 서버(예: GitHub, GitLab 등)에 위치 |
| 접근방식 | 개발자가 직접 접근하여 파일을 수정하고 커밋 | 네트워크를 통해 접근하여 푸시(push)와 풀(pull) 작업을 수행 |
| 협업방식 | 단일 개발자의 작업 공간 역할 | 여러 개발자가 변경 사항을 공유하고 협업할 수 있는 중앙 저장소 역할 |
| 보안 및 백업 | 개인 컴퓨터에 있어 손상이나 손실 위험이 있음 | 서버에 저장되어 있어 데이터 백업과 보안 측면에서 상대적으로 안전 |
| 오프라인 작업 가능 여부 | 오프라인 상태에서도 Git 작업이 가능 | 인터넷 연결 필요 |

Git 저장소 작업 관계

- 로컬 저장소 : 수정 및 커밋
- 원격 저장소 : push 및 pull



.git 디렉토리 구조와 역할

- hooks: Git의 클라이언트 및 서버 후크 스크립트가 저장되는 디렉토리
- info: 원격 저장소에 대한 정보가 저장되는 디렉토리
- logs: Git 작업에 대한 로그 파일이 저장되는 디렉토리
- objects: Git 객체(blob, tree, commit)가 저장되는 디렉토리
- refs: 브랜치와 태그에 대한 참조(reference)가 저장되는 디렉토리
- config: Git 저장소의 구성 설정 파일이 저장되는 파일
- description: Git 저장소에 대한 설명이 저장되는 파일
- HEAD: 현재 체크아웃된 브랜치나 커밋을 가리키는 파일
- index: staging area(준비 영역)에 있는 파일에 대한 메타데이터가 저장되는 파일



> 직접 .git 디렉토리 내부를 수정하지 않고, Git 명령어를 통해 접근

로컬 저장소 관리



파일의 추가와 커밋(git add, git commit)

파일 생성 또는 수정

```
hello_git.txt
```

저장소로 사용할
디렉토리 생성 또는 수정

```
mkdir c:\my_git_dir
```

로컬 저장소 생성

```
move hello_git.txt my_git_dir  
cd c:\my_git_dir  
git init
```

Staging Area에 파일 추가

```
git add hello_git.txt
```

로컬 저장소 기록

```
git commit -m "첫번째 커밋"
```

커밋 확인

```
git log
```


저장소 데이터 동기화

UNIT

원격 저장소 설정



원격 저장소 생성

- Github 새 저장소 : Github 접속 & 로그인 후 저장소 생성(Repositories → New)

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 *
 trex99 / ◀ 저장소명(필수)

Great repository names are short and memorable. Need inspiration? How about [fluffy-guide](#) ?

Description (optional)

◀ 저장소 설명(옵션)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit. ◀ 저장소 공개여부(필수, 기본값은 공개)

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with: ◀ 저장소 처음 접속 시 설명내용을 담고 있는 파일(옵션)

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

Add .gitignore ◀ Git 관리 비대상 파일 목록 파일(옵션, 기본값은 비생성)

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

Choose a license ◀ 라이선스 설정 (옵션, 기본값은 라이선스 없음)

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

① You are creating a public repository in your personal account.

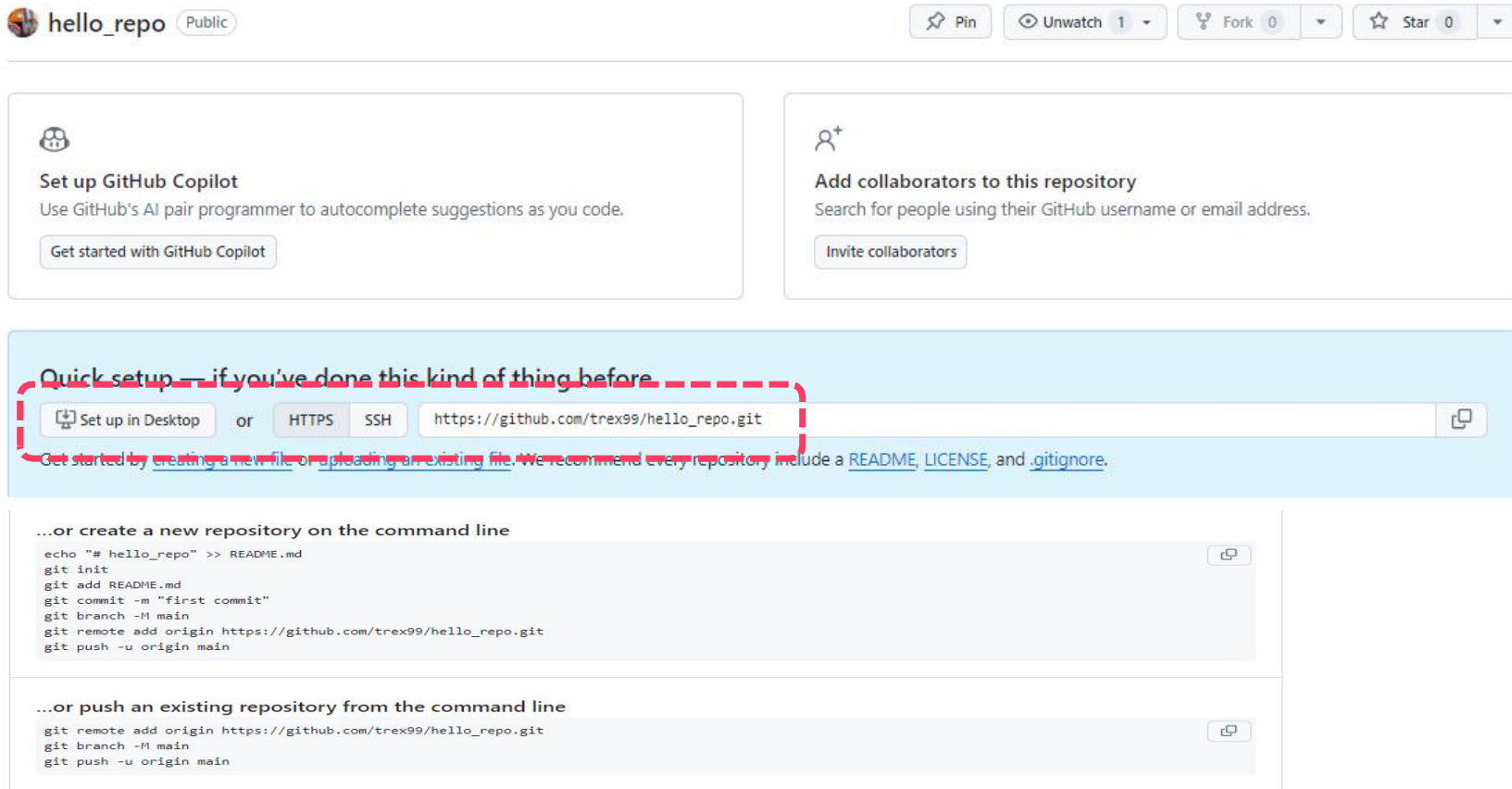
Create repository

원격 저장소 설정



원격 저장소 결과 예시

- https://github.com/trex99/hello_repo.git



The screenshot shows the GitHub repository page for 'hello_repo' (Public). At the top, there are buttons for Pin, Unwatch (1), Fork (0), and Star (0). Below the repository name, there are two main sections: 'Set up GitHub Copilot' and 'Add collaborators to this repository'. The 'Set up GitHub Copilot' section has a button 'Get started with GitHub Copilot'. The 'Add collaborators to this repository' section has a button 'Invite collaborators'. Below these sections is a blue box with the heading 'Quick setup — if you've done this kind of thing before'. Inside this box, there is a red dashed line highlighting the 'Set up in Desktop' button, the 'or' text, the 'HTTPS' button, and the URL 'https://github.com/trex99/hello_repo.git'. Below the highlighted elements, there is a note: 'Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).' Below the blue box, there are two sections for command-line setup. The first section is titled '...or create a new repository on the command line' and contains the following commands:

```
echo "# hello_repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/trex99/hello_repo.git
git push -u origin main
```

 The second section is titled '...or push an existing repository from the command line' and contains the following commands:

```
git remote add origin https://github.com/trex99/hello_repo.git
git branch -M main
git push -u origin main
```

원격 저장소 설정



원격 저장소 연결

- `git remote add origin https://github.com/trex99/hello_repo.git`
- `git branch -M main`

원격저장소 기본 별칭

원격 저장소 연결

- 전송
 - `git push origin main`
 - `git push -u origin main`
 - * `-u` : upstream을 설정하여 이후부터는 “`git push`” 만으로 원격 저장소에 push 함
- 수신
 - `git pull origin main`