GitHub 가이드

1. GitHub란?

GitHub는 **Git을 기반으로 하는 온라인 코드 저장소 플랫폼**으로, 개발자들이 코드를 저장하고 버전 관리하며 협업할 수 있도록 도와주는 도구입니다.

---

2. 주요 기능과 특징

- **버전 관리**: Git으로 프로젝트의 모든 변경 이력을 추적
- **저장소(Repository)**: 소스 코드 및 관련 파일을 저장하는 공간
- **협업 기능**: 브랜치, Pull Request로 안전하게 공동 작업 가능
- **이슈 관리**: 버그나 작업 항목을 기록하고 관리
- **자동화**: GitHub Actions로 테스트, 빌드, 배포 자동화 가능

---

3. GitHub 사용 방식: CLI vs UI

* CLI (Command Line Interface) 방식

터미널 또는 명령 프롬프트에서 **Git 명령어를 사용**해 GitHub와 소통하는 방식입니다. 개발자에게 가장 널리 쓰이며 자동화나 스크립트에 유리합니다.

CLI에서 자주 사용하는 Git 명령어 정리

목적	명령어	설명
초기 설정	`git config --global user.name "이름"`	사용자 이름 설정
	`git config --global user.email "이메일"`	이메일 설정
저장소 시작	`git init`	현재 폴더를 Git 저장소로 초기화
파일 추가	`git add .`	모든 변경 파일을 스테이징
	`git add 파일명`	특정 파일만 스테이징
커밋	`git commit -m "메시지"`	변경 사항을 커밋 (저장)
원격 연결	`git remote add origin [URL]`	GitHub 저장소와 연결
푸시	`git push -u origin master`	로컬 변경사항을 GitHub에 업로드
클론	`git clone [URL]`	GitHub 저장소를 내 컴퓨터에 복사

목적	명령어	설명
상태 확인	`git status`	현재 상태 확인 (변경 파일 등)
로그 보기	`git log`	커밋 히스토리 확인
브랜치 생성	`git branch 브랜치명`	새 브랜치 생성
브랜치 이동	`git checkout 브랜치명`	해당 브랜치로 전환
병합	`git merge 브랜치명`	브랜치를 현재 브랜치에 병합

* UI (Graphical User Interface) 방식

GitHub 웹사이트 또는 **GitHub Desktop** 같은 도구를 사용해 시각적으로 Git과 GitHub를 사용하는 방식입니다.

웹 UI 사용 방법 (github.com)

작업	방법
저장소 만들기	로그인 → "New repository" 클릭 → 이름과 옵션 설정 후 생성
파일 수정	저장소 > 파일 클릭 > 연필 아이콘으로 수정
커밋	변경 후 "Commit changes" 클릭
브랜치 관리	저장소 상단에서 브랜치 선택 또는 새로 만들기
Pull Request 만들기	브랜치에서 변경 → "Compare & pull request" 클릭
이슈 생성	저장소 > Issues 탭 → "New issue" 클릭

GitHub Desktop (GUI 프로그램)

- GitHub에서 제공하는 공식 앱으로, Git 명령어를 몰라도 버튼 클릭으로 대부분의 작업 수행 가능
- 클론, 커밋, 푸시, 브랜치 전환 등이 직관적으로 가능

---

4. GitHub 기본 사용 가이드

1단계: GitHub 회원가입

- https://github.com 에서 계정 생성

2단계: Git 설치 (CLI 사용 시 필수)

- https://git-scm.com 에서 Git 다운로드 및 설치

- 설치 후 사용자 정보 설정

```
git config --global user.name "이름"
git config --global user.email "이메일"
```

3단계: 저장소 생성 및 연결

웹에서 저장소 생성 후 CLI로 연결:

```
git init
git remote add origin https://github.com/사용자명/저장소이름.git
git add .
git commit -m "첫 커밋"
git push -u origin master
```

4단계: 로그인 정보가 다를 때 확인 및 조치

연결할 레포지토리 생성 후 수행

```
git remote -v
```

정상일 때 결과

```
origin  https://github.com/eunhye010615/portfolio.git (fetch)
origin  https://github.com/eunhye010615/portfolio.git (push)
```

다를 때 수행할 내용⇒ origin 주소와 저장소 존재 여부를 확인 필

```
git remote remove origin
git remote add origin https://github.com/eunhye010615/portfolio.git
```

---

5. GitHub의 장점과 단점

* 장점

항목	설명
버전 관리	변경 이력 추적 및 복구 가능
협업	여러 명이 동시에 작업해도 관리 가능
온라인 백업	로컬 문제 발생 시에도 코드 보호 가능
오픈소스 참여	다른 프로젝트에 기여 가능
자동화 기능	GitHub Actions 등으로 워크플로우 설정 가능
GUI와 CLI 모두 지원	사용자 편의에 따라 방식 선택 가능

* 단점

항목	설명
초기 진입장벽	Git 개념과 명령어에 익숙해지는 데 시간 필요
인터넷 필수	GitHub는 온라인 서비스이므로 연결 필요
용량 제한	대용량 파일 업로드에 제한 존재
보안 위험	실수로 민감 정보 업로드 시 보안 사고 가능

---

6. 마무리

GitHub는 초보자에게는 낯설 수 있지만, **코드를 안전하게 관리하고, 협업하며, 성장할 수 있는 매우 강력한 도구**입니다.

CLI와 UI 방식을 자유롭게 선택해 사용할 수 있으며, 익숙해질수록 개발 생산성이 크게 향상됩니다.

다른 컴퓨터에서 실행할 때