# GitHub 완전 정복 가이드

## 1. Git과 GitHub, 무엇이 다른가요?

### Git: 버전 관리 시스템

* **Git**은 컴퓨터 파일의 변경사항을 추적하고 여러 사용자 간에 해당 파일 작업을 조율하기 위한 \*\*'버전 관리 시스템'\*\*입니다. 쉽게 말해, 프로젝트의 특정 시점들을 사진 찍듯이 저장하고, 원할 때마다 과거의 모습으로 돌아갈 수 있게 해주는 마법 같은 도구입니다. 내 컴퓨터에 설치해서 사용합니다.

### GitHub: Git 저장소 호스팅 서비스

* **GitHub**는 Git으로 관리하는 프로젝트를 다른 사람들과 공유하고 함께 작업할 수 있는 \*\*'웹 기반 호스팅 서비스'\*\*입니다. Git이 프로젝트의 역사를 기록하는 일기장이라면, GitHub는 그 일기장을 친구들과 함께 쓰고 돌려볼 수 있는 온라인 공간(클라우드 + 소셜 네트워크)인 셈이죠.

## 2. GitHub 기본 사용법: 이것만 알면 시작할 수 있어요!

### Repository (저장소)

* 프로젝트의 모든 파일과 버전 기록이 담겨 있는 폴더입니다. 줄여서 'Repo'라고도 부릅니다. GitHub에서 New repository 버튼을 눌러 새로운 프로젝트 저장소를 만들 수 있습니다.

### Commit (커밋)

* 프로젝트의 변경사항을 저장하는 작업 단위를 의미합니다. "이 부분을 이렇게 수정했어"라고 기록을 남기는 행위입니다. 각 커밋에는 변경사항을 설명하는 메시지를 함께 기록하여 어떤 내용이 바뀌었는지 쉽게 파악할 수 있습니다.

### Branch (브랜치)

* 독립적으로 어떤 작업을 진행하기 위한 '가지'입니다. 메인 가지(보통 main 또는 master 브랜치)를 그대로 둔 채, 새로운 기능 개발이나 버그 수정을 위한 자신만의 작업 공간을 만드는 것입니다. 여러 사람이 동시에 다른 작업을 해도 서로 영향을 주지 않아 안전합니다.

### Pull Request (풀 리퀘스트)

* 내가 만든 브랜치에서의 변경사항을 메인 브랜치에 합치고(Merge) 싶을 때 보내는 '요청'입니다. "제가 이런 기능을 만들었는데, 확인해 보시고 괜찮으면 반영해주세요!"라는 의미입니다. 다른 개발자들이 내 코드를 검토(리뷰)하고 피드백을 줄 수 있어 협업의 핵심 기능으로 꼽힙니다.

## 3. GitHub 고급 활용: 협업의 달인이 되어보세요!

### Fork (포크)

* 다른 사람의 공개 저장소를 내 계정으로 그대로 복사해오는 기능입니다. 원본에 영향을 주지 않고 자유롭게 코드를 수정하고 실험해볼 수 있습니다. 마음에 드는 수정이 이뤄졌다면, 원본 저장소에 Pull Request를 보내 기여할 수도 있습니다.

### Issues (이슈)

* 프로젝트의 할 일, 버그, 개선점 등을 기록하고 논의하는 공간입니다. 특정 이슈를 특정 담당자에게 할당하거나, 관련된 Pull Request와 연결하여 작업 진행 상황을 체계적으로 관리할 수 있습니다.

### Projects (프로젝트)

* 칸반 보드 스타일로 이슈와 Pull Request를 시각적으로 관리할 수 있는 도구입니다. 'To do', 'In progress', 'Done' 같은 열을 만들어 작업의 흐름을 한눈에 파악하기 좋습니다.

### GitHub Actions (액션)

* 커밋, Pull Request 등 특정 이벤트가 발생했을 때, 코드 테스트, 빌드, 배포 등의 작업을 자동으로 실행해주는 워크플로우 자동화 도구입니다. 반복적인 작업을 줄여 개발 생산성을 크게 높여줍니다.

## 4. 실습 예시: GitHub와 첫걸음 떼기

이제 이론을 배웠으니 직접 체험해볼 시간입니다! 아래 순서대로 따라 해보세요.

### 첫째, 내 저장소 만들고 자기소개하기

1. GitHub에 로그인한 후, 오른쪽 위 '+' 아이콘을 클릭하고 New repository를 선택합니다.
2. Repository name에 hello-github와 같이 원하는 이름을 입력합니다.
3. Public으로 설정하고, Add a README file 옵션을 체크한 후 Create repository 버튼을 클릭합니다.
4. 생성된 저장소에서 README.md 파일을 찾아 연필 모양(Edit) 아이콘을 클릭합니다.
5. 마크다운 문법을 이용해 간단한 자기소개를 작성하고, 페이지 하단의 Commit changes 버튼을 눌러 저장합니다.

### 둘째, 새로운 브랜치 만들고 코드 작성하기

1. 저장소 페이지에서 main이라고 표시된 브랜치 버튼을 클릭합니다.
2. Find or create a branch 입력창에 feature/add-hello와 같이 새로운 브랜치 이름을 입력하고 Create branch를 선택합니다.
3. 새로 만든 브랜치로 이동된 것을 확인한 후, Add file > Create new file을 클릭합니다.
4. 파일 이름에 hello.py를 입력하고, 내용에는 아래 코드를 작성합니다.print("Hello GitHub!")
5. Commit new file 버튼을 눌러 파일을 저장합니다.

### 셋째, Pull Request로 main 브랜치에 합치기

1. hello.py 파일을 커밋하고 나면, 페이지 상단에 노란색 배경의 알림창이 나타납니다. Compare & pull request 버튼을 클릭하세요.
2. Pull Request의 제목과 설명을 작성하고 Create pull request 버튼을 누릅니다.
3. 이제 내가 요청한 코드 변경사항을 검토할 수 있는 페이지가 열립니다. 실무에서는 동료가 코드를 리뷰해주겠지만, 지금은 직접 합쳐보겠습니다.
4. 초록색 Merge pull request 버튼을 클릭하고, Confirm merge를 클릭하여 변경사항을 main 브랜치에 최종 반영합니다.
5. 이제 main 브랜치로 돌아가 보면, hello.py 파일이 성공적으로 추가된 것을 확인할 수 있습니다!

**축하합니다!** 여러분은 방금 GitHub 협업의 가장 기본적인 과정을 모두 체험했습니다.