## 🌳 Git 핵심 개념: 브랜치 기반 워크플로우

### 1. 🛡️ 메인 브랜치 보호하기

* **메인 브랜치 (main, master)**
  + **개념**: 실제 서비스에 배포되는, **100% 정상 작동하는 코드의 기준점**.
  + **규칙**: 절대로 직접 코드를 수정하지 않고, 항상 다른 브랜치의 수정사항을 \*\*병합(Merge)\*\*을 통해서만 반영합니다.

### 2. 👩‍💻 개발용 브랜치에서 작업하기

* **개발용 브랜치 (feat/..., fix/...)**
  + **개념**: 메인 브랜치에서 복사한 **독립적인 작업 공간**.
  + **규칙**: 여기서 마음껏 코드를 수정하고, 오류가 나더라도 메인 브랜치에 영향을 주지 않습니다.

### 3. 🚀 작업 배포 흐름: 안전 4단계 시스템

| **단계** | **역할** | **명령어** | **결과** |
| --- | --- | --- | --- |
| **1. 브랜치 생성 및 이동** | 메인 브랜치의 코드를 복사해서 작업 시작 | git checkout -b fix/perf | main은 안전하게 두고, fix/perf라는 새 작업 공간 생성 |
| **2. 코드 수정 및 기록** | 로컬 컴퓨터에서 코드 수정 후 변경사항 기록 | git add .  git commit -m "메시지" | 수정된 파일들이 내 로컬 fix/perf 브랜치에 커밋으로 저장 |
| **3. 테스트용 배포** | 원격 저장소에 작업 내용을 업로드 | git push origin fix/perf | Vercel이 **테스트용 URL**에 배포 (main에 영향 X) |
| **4. 최종 배포 준비** | 테스트 통과 후 메인 브랜치에 합치기 | GitHub/GitLab에서 **Pull Request 생성** 및 병합 | main 브랜치에 fix/perf의 코드가 반영됨 |

## 🛠️ 자주 쓰는 명령어 (자주 보는 순서)

### 1. 현재 상태 확인

* **git status**: 현재 작업 폴더의 상태를 보여줍니다. 어떤 파일을 수정했고, 어떤 브랜치에 있는지 확인할 수 있습니다.

### 2. 브랜치 이동

* **git checkout [브랜치 이름]**: 다른 브랜치로 이동합니다. (예: git checkout main)

### 3. 변경사항 저장

* **git add .**: 모든 수정 파일을 커밋 준비 상태로 만듭니다.
* **git commit -m "[메시지]"**: 준비된 파일을 '하나의 기록'으로 저장합니다.

### 4. 원격 저장소에 반영

* **git push origin [브랜치 이름]**: 내 로컬의 기록을 원격(GitHub/Vercel)으로 보냅니다.
* **git pull origin [브랜치 이름]**: 원격의 최신 기록을 내 로컬로 가져옵니다.

### 5. 긴급 상황 대처

* **git stash**: 커밋하지 않은 수정사항을 임시로 저장합니다.
* **git stash pop**: 임시 저장된 내용을 다시 불러옵니다.