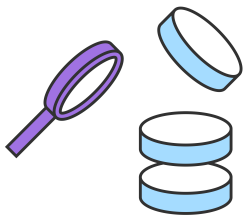
git status

- 사용법
  - Ignoring Files
- Example



git status 명령은 작업 디렉토리 및 스테이징 영역의 상태를 표시합니다. Git이 어떤 변경 내용을 준비했는지, 어떤 변경 사항이 추적되어 있지 않은지, 어떤 파일이 추적 중인지 확인할 수 있습니다. 상태 출력은 커밋 된 프로젝트 기록과 관련된 정보를 표시하지 않습니다. 이를 위해서는 git log를 사용해야합니다.

사용법

```
git status
```

Staged, unstaged, 그리고 untracked 파일 목록을 보여줍니다.

Ignoring Files

추적 할 수없는 파일은 일반적으로 두 가지 범주로 나뉩니다. 그것들은 방금 프로젝트에 추가되었고 아직 커밋되지 않은 파일이거나 .pyc, .obj, .exe 등의 컴파일 된 바이너리입니다. 이전에는 git에 포함하는 것이 유익했지만 상태 출력, 후자는 실제로 당신의 저장소에서 진행되고있는 것을보기 어렵게 만들 수 있습니다.

이러한 이유로 Git은 .gitignore라는 특수 파일에 경로를 넣어 파일을 완전히 무시할 수 있습니다. 무시하려는 파일은 별도의 줄에 포함해야하며 * 기호는 와일드 카드로 사용할 수 있습니다. 예를 들어 프로젝트 루트의 .gitignore 파일에 다음을 추가하면 컴파일 된 파이썬 모듈이 git 상태로 나타나지 않습니다.

Example

뜻하지 않은 것을 실수로 저 지르지 않도록 변경하기 전에 저장소의 상태를 확인하는 것이 좋습니다. 이 예에서는 스냅 샷을 준비하고 커밋하기 전과 후에 리포지토리 상태를 표시합니다.

```
# Edit hello.py
git status
# hello.py is listed under "Changes not staged for commit"
git add hello.py
git status
# hello.py is listed under "Changes to be committed"
git commit
git status
# nothing to commit (working directory clean)
```

첫 번째 상태 출력은 파일을 unstaged로 표시합니다. 자식 추가 작업은 두 번째 자식 상태에 반영되고 마지막 상태 출력은 커밋 할 것이 없다는 것을 알려주며 작업 디렉토리는 가장 최근의 커밋과 일치합니다. 일부 git 명령 (예 : git merge)은 우연히 변경 사항을 덮어 쓰지 않도록 작업 디렉토리가 깨끗해야합니다.

Git 가이드

1. Git 시작하기
   - Git 저장소
     - git init
     - git clone
     - git config
   - 변경 저장하기
     - git add
     - git commit
     - git diff
     - git stash
     - .gitignore
   - 저장소 점검하기
     - git status
     - git log
     - git tag
     - git blame
   - 변경 취소하기
     - git 실행 취소
     - git clean
     - git revert
     - git reset
     - git rm
   - Rewriting history
     - git commit --amend
     - git rebase
     - git reflog
2. Git 협업하기
   - 동기화하기
     - git remote
     - git fetch
     - git pull
     - git push
   - 브랜치 사용하기
     - git branch
     - git checkout
     - git merge
     - 병합 충돌 해결하기 (Merge conflicts)
     - 병합 전략 (Merge strategies)
   - Pull request 만들기