Git 보조자료

강의

- Git command
- **1**) **Setup**
  ↳ Initialize a new repository: **git init**
  ↳ Configure username and email:
  git config --global user.name <your-name>
  git config --global user.email <your-email>
  ↳ Clone a repository: **git clone <repository-url>**

  2) **Stage** & **Commit**
  ↳ Add a file: **git add <file>**
  ↳ Add all changes**: git add .**
  ↳ Check unstaged changes: git diff
  ↳ Commit changes: **git commit -m "Message"**
  ↳ Reset staging area**: git reset**

  3) **Status** & **History**
  ↳ Check repository state: **git status**
  ↳ View commit history: **git log**
  ↳ Show commit details: **git show <commit-hash>**

강의

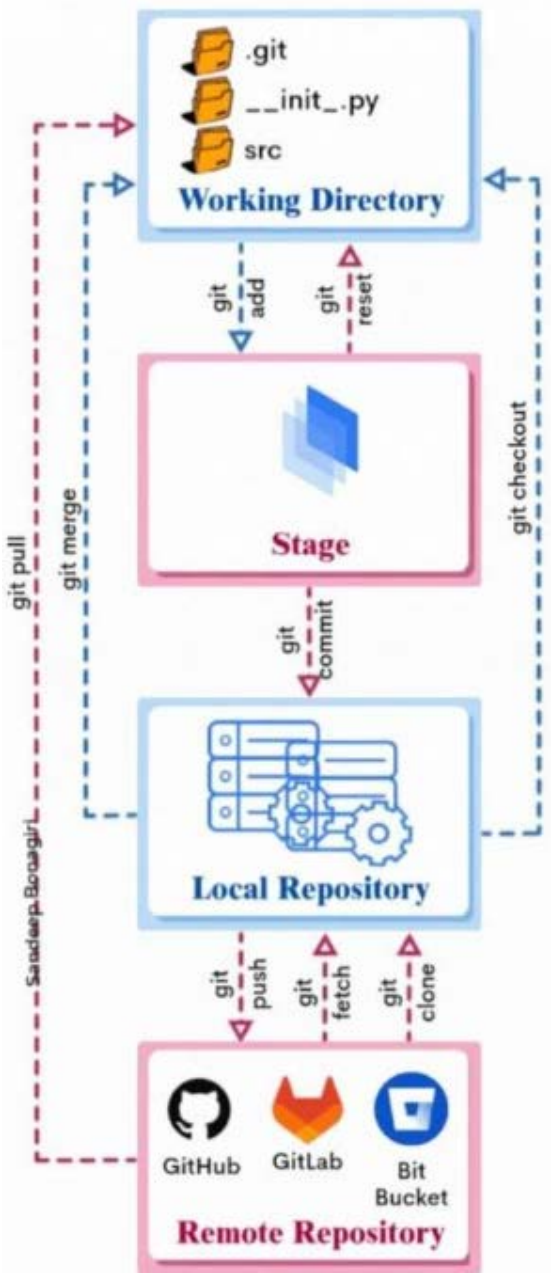
- **4) Branches**
  ↳ List branches: **git branch**
  ↳ Create a branch: **git branch <branch-name>**
  ↳ Rename current branch: **git branch -m <new-branch-name>**
  ↳ Delete a branch: **git branch -d <branch-name>**
  ↳ Switch branches: **git checkout <branch-name>**
  ↳ Merge a branch: **git merge <branch-name>**

  5) Remotes
  ↳ Add a remote: **git remote add <name> <repository-url>**
  ↳ Push commits: **git push <remote> <branch>**
  ↳ Pull changes: **git pull <remote>**

  6) Cleanup & Extras

  ↳ Optimize repository: **git gc**
  ↳ Stash changes: **git stash**
  ↳ Reapply stash: **git stash apply**

How Git Works



Key Concepts and Tools

- **Working Directory**: Local files you're currently working on.
- **Stage**: Area to prepare files for commit.
- **Local Repository**: Your personal copy of the repo.
- **Remote Repository**: Online version of the repository.
- **.git**: Metadata folder for version control.
- **init.py**: Initializes a Python package, not related to Git operation.
- **src**: Conventional directory for source code in many projects.
- **GitHub**: Host for code repositories.
- **GitLab**: Platform for Git repository management.
- **Bitbucket**: Service offering Git repository hosting.

Common Git Commands

- **git add**: Prepare files for a commit.
- **git commit**: Save changes to local history.
- **git push**: Upload commits to remote.
- **git fetch**: Get updates from remote.
- **git clone**: Copy a remote repository locally.
- **git pull**: Update local with remote changes.
- **git merge**: Combine changes from different branches.
- **git checkout**: Switches branches or restores working directory files.
- **git reset**: Unstages files or resets commit history; can be used to undo changes in the index or history.

강의

Git commands Cheat Sheet

Initialize a new git repository:
```
git init
```

Set configuration values for your username and email:
```
git config --global user.name <your-name>
git config --global user.email <your-email>
```

Clone a repository:
```
git clone <repository-url>
```

Add a file to the staging area:
```
git add <file>
```

Add all files changes to the staging area:
```
git add .
```

Check the unstaged changes:
```
git diff
```

Commit the staged changes:
```
git commit -m "Message"
```

Reset staging area to the last commit:
```
git reset
```

Check the state of the working directory and the staging area:
```
git status
```

Remove a file from the index and working directory:
```
git rm <file>
```

List the commit history:
```
git log
```

Check the metadata and content changes of the commit:
```
git show <commit-hash>
```

Lists all local branches:
```
git branch
```

Create a new branch:
```
git branch <branch-name>
```

Rename the current branch:
```
git branch -m <new-branch-name>
```

Delete a branch:
```
git branch -d <branch-name>
```

Switch to another branch:
```
git checkout <branch-name>
```

강의

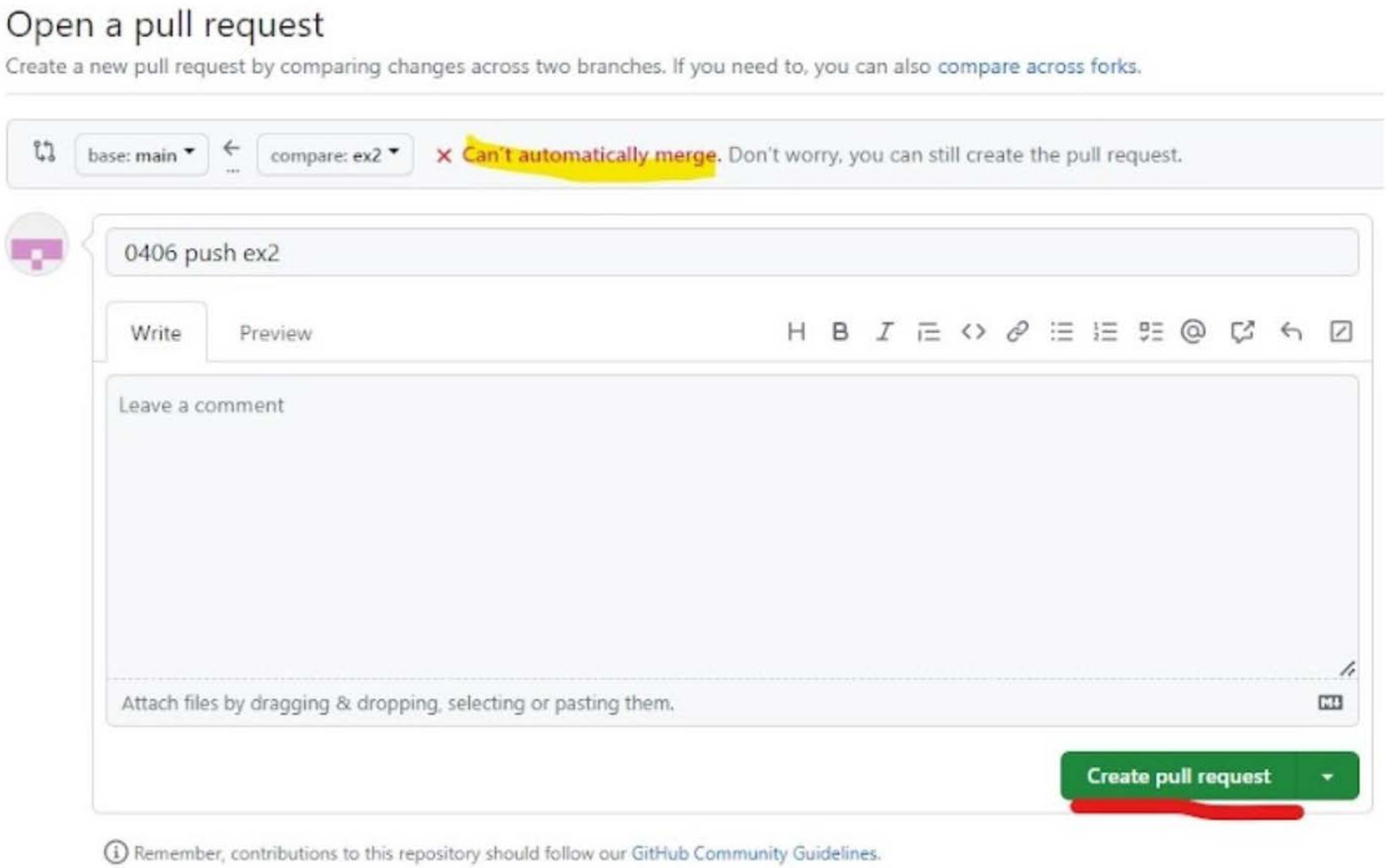
```
Merge specified branch into the current branch:
git merge <branch-name>

Create a new connection to a remote repository:
git remote add <name> <repository-url>

Push the committed changes to a remote repository:
git push <remote> <branch>

Download the content from a remote repository:
git pull <remote>

Cleanup unnecessary files and optimize the local repository:
git gc

Temporarily remove uncommitted changes and  save them for later use:
git stash

Reapply previously stashed changes
git stash apply
```

충돌시 해결방법

- 충돌의 원인
  - 충돌은 여러 사람이 동시에 작업할 때, 서로 다른 branch에서 같은 파일을 수정하고 merge할 때 발생할 수 있슴
  - 다음 예제는 서로 다른 위치에 repository를 clone 받아서 같은 파일의 같은 라인을 수정하고 merge함으로써 여러 개발자가 동시에 작업했을 때 충돌이 나는 상황을 가정한 것

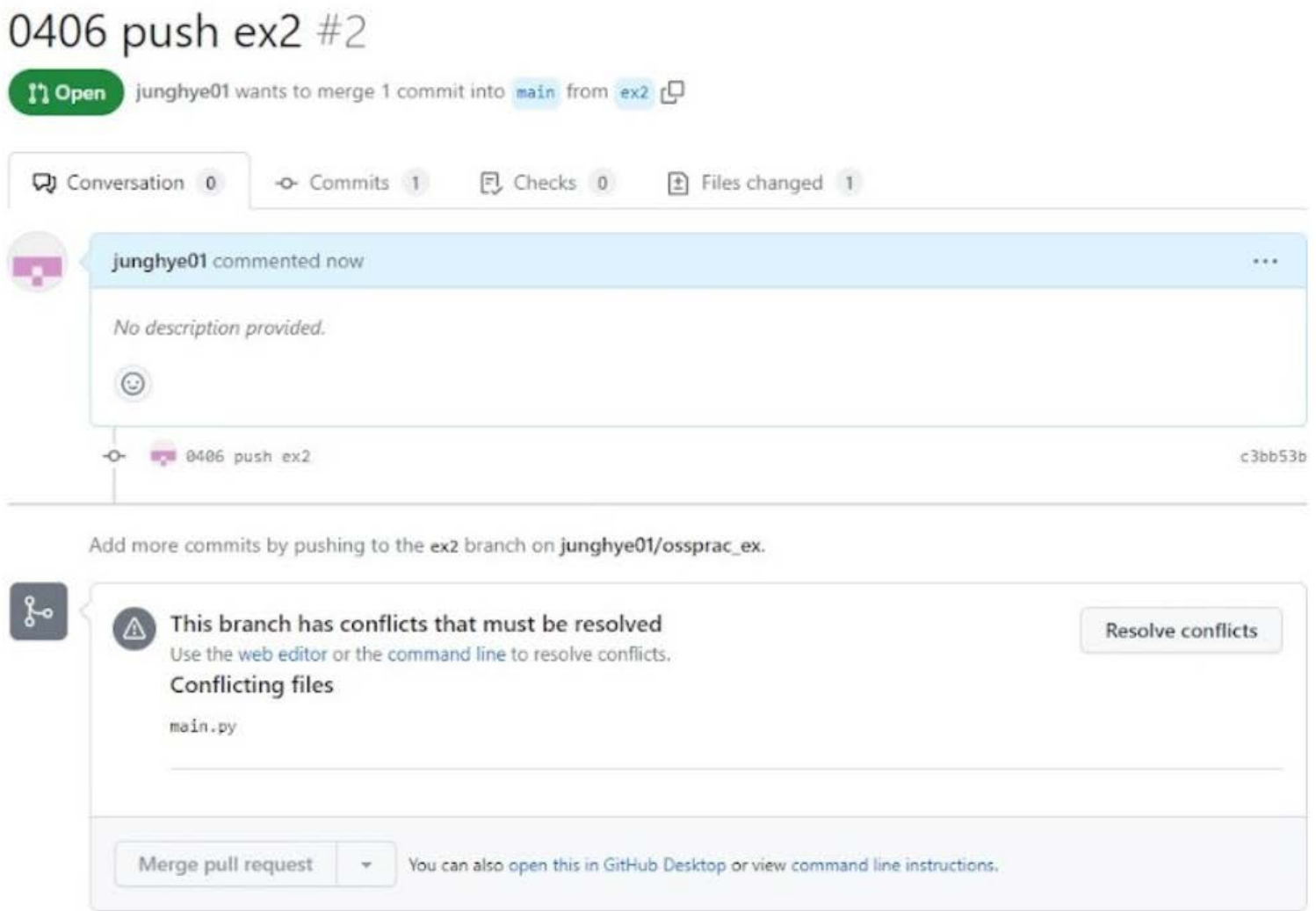
충돌시 해결방법

- 충돌의 해결 방식
  - Compare & pull request 버튼을 클릭



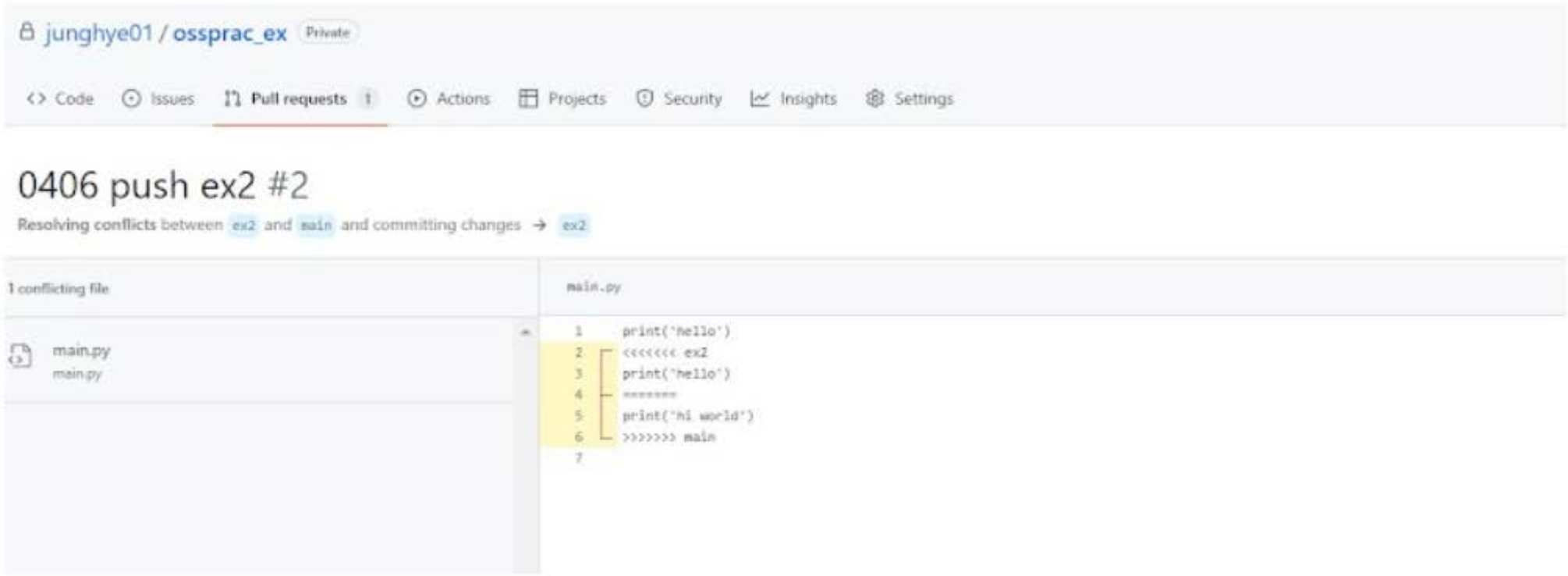
충돌시 해결방법

- 충돌의 해결 방식
  - Resolve conflict 버튼을 클릭



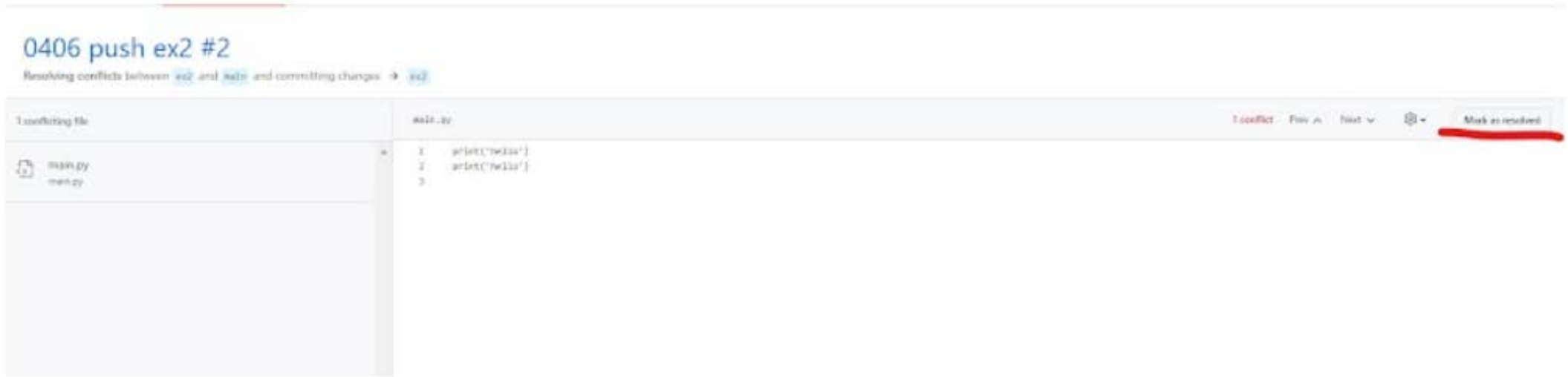
충돌시 해결방법

- 충돌의 해결 방식
  - conflict가 발생한 양쪽 파일의 수정사항 확인



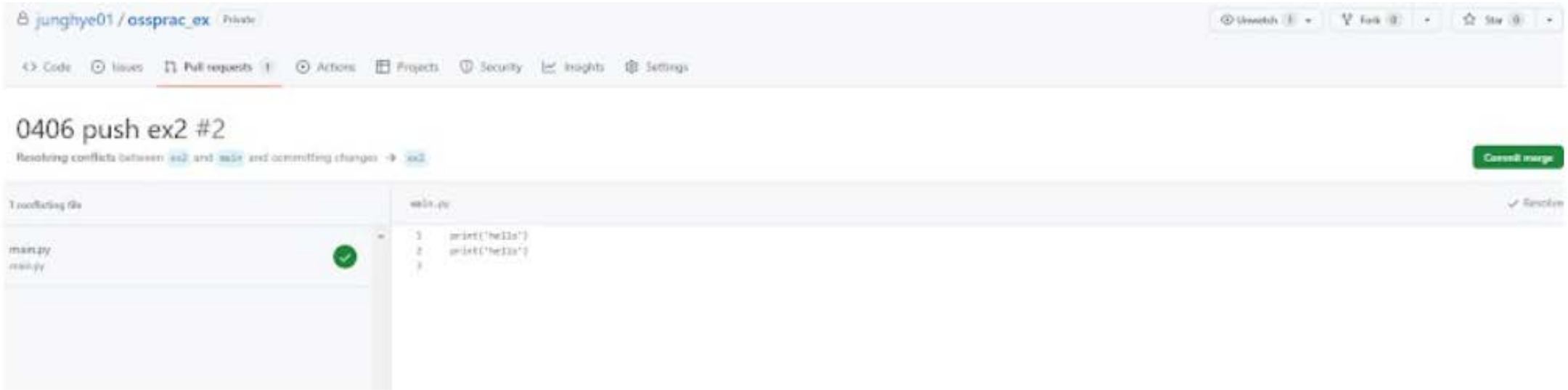
충돌시 해결방법

- 충돌의 해결 방식
  - 수동으로 수정 후 → ‘Mark as resolved’ 클릭



충돌시 해결방법

- 충돌의 해결 방식
  - 오른쪽 상단의 'Commit merge' 클릭



충돌시 해결방법

- 충돌의 해결 방식
  - Merge pull request