

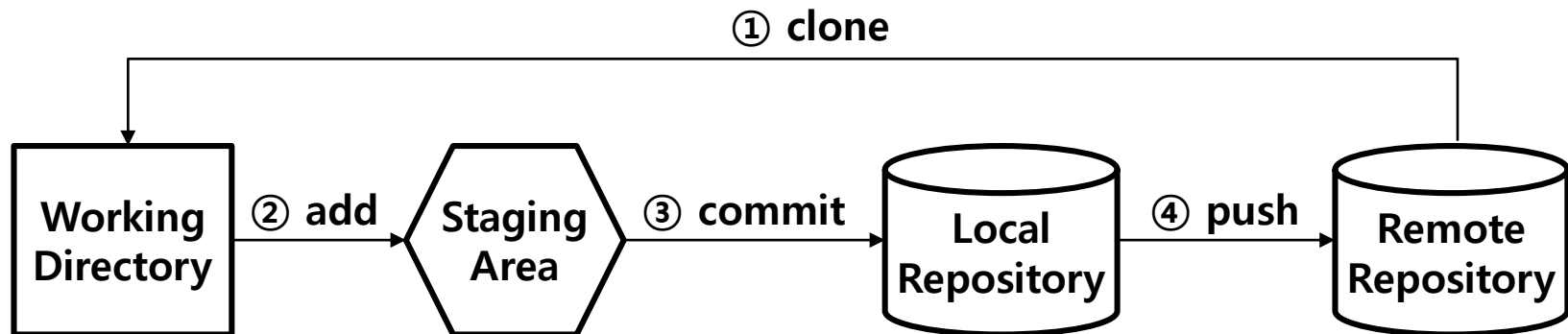
# Git 활용(2)

- 원격 저장소 우선 Workflow
- 로컬 저장소 우선 Workflow
- 팀 프로젝트 작업 Workflow
- 타 프로젝트 참여 Workflow
- [참고] Git 명령 요약

# 원격 저장소 우선 Workflow

## □ Workflow(1)

- 원격저장소(Github)에 Repository 먼저 생성 후 로컬 저장소로 복제
- 원격 저장소의 URL을 알고 있어야 작업 가능
- 이미 프로젝트가 진행되고 있는 소스 코드를 가져와서 작업할 때 사용
- 내려받은 소스 코드를 새로운 브랜치로 생성해서 작업



# 원격 저장소 우선 Workflow

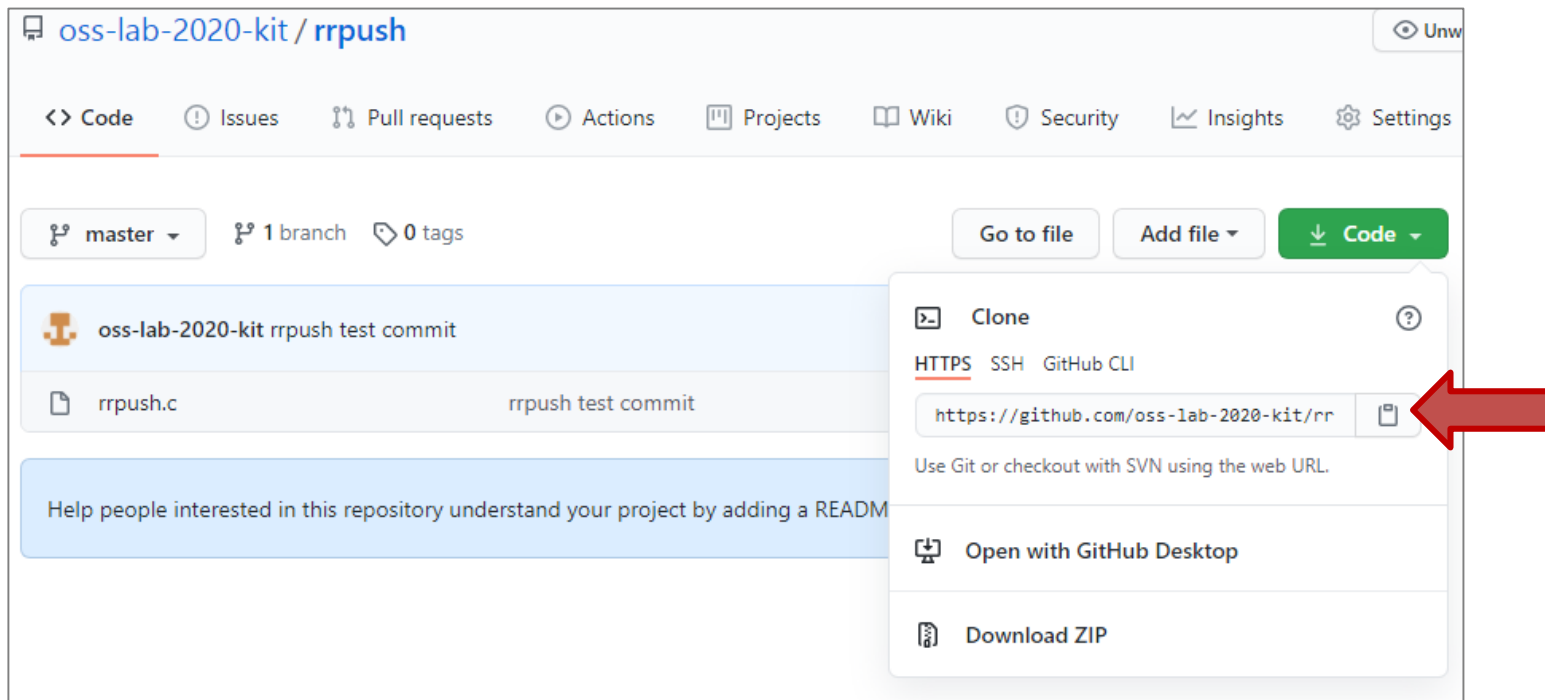
## □ [실습1] 원격 저장소 URL 획득

- 공개된 URL이나 검색을 통해 알게된 URL로 사이트에 접근

- "Code" 항목 클릭 후 Clone URL 복사

`https://github.com/oss-lab-2020-kit/rrpush`

`https://github.com/oss-lab-2020-kit/rrpush.git`



# 원격 저장소 우선 Workflow

## □ [실습2] 로컬 저장소에 복제 및 확인

- 현재 디렉터리 위치를 확인하고 복제할 것

```
linuxer@linuxer-PC:~$ cd
linuxer@linuxer-PC:~$ git clone https://github.com/oss-lab-2020-kit/rrpush
linuxer@linuxer-PC:~$ cd rrpsh
linuxer@linuxer-PC:~/rrpush$ ls -al
```

## □ [실습3] 로컬 작업 및 원격 저장소에 보관

```
linuxer@linuxer-PC:~/rrpush$ echo 'rrpush test' > main.c
linuxer@linuxer-PC:~/rrpush$ git add main.c
linuxer@linuxer-PC:~/rrpush$ git commit -m 'rrpush test'

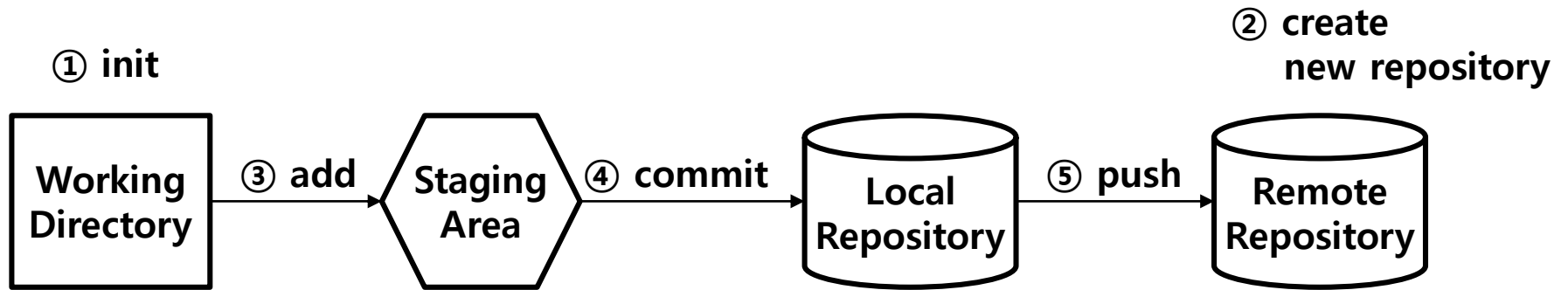
linuxer@linuxer-PC:~/rrpush$ git remote -v
origin  https://github.com/oss-lab-2020-kit/rrpush (fetch)
origin  https://github.com/oss-lab-2020-kit/rrpush (push)

linuxer@linuxer-PC:~/rrpush$ git push
```

# 로컬 저장소 우선 Workflow

## □ Workflow

- 로컬 저장소에서 작업하던 것을 원격 저장소(Github)로 보관
- 혼자 작업하던 것을 원격 저장소에 보관할 때
- 혼자 작업하던 것을 공개해서 다른 사람과 함께 작업해야 할 때



# 로컬 저장소 우선 Workflow

- [실습4] 로컬 저장소 생성 → 작업 파일 생성 → 원격 저장소에 push

```
linuxer@linuxer-PC:~$ mkdir lrpush
linuxer@linuxer-PC:~$ cd lrpush
linuxer@linuxer-PC:~/lrpush$ git init

linuxer@linuxer-PC:~/lrpush$ echo 'lrtest1' > main.c
linuxer@linuxer-PC:~/lrpush$ git add main.c
linuxer@linuxer-PC:~/lrpush$ git commit -m 'lrtest1'

linuxer@linuxer-PC:~/lrpush$ git remote add origin https://github.com/oss-
lab-2020-kit/lrpush.git
linuxer@linuxer-PC:~/lrpush$ git push --set-upstream origin master
Username for 'https://github.com': oss-lab-2020-kit
Password for 'https://oss-lab-2020-kit@github.com': ****
오브젝트 나열하는 중: 3, 완료.
오브젝트 개수 세는 중: 100% (3/3), 완료.
오브젝트 쓰는 중: 100% (3/3), 207 bytes | 207.00 KiB/s, 완료.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/oss-lab-2020-kit/lrpush.git
* [new branch]      master -> origin
```

# 로컬 저장소 우선 Workflow

- [실습5] 새로운 작업 파일 생성 → 원격 저장소에 push

```
linuxer@linuxer-PC:~/lrpush$ echo 'lrtest2' > lrtest.c
linuxer@linuxer-PC:~/lrpush$ git add lrtest.c
linuxer@linuxer-PC:~/lrpush$ git commit -m 'lrtest2'

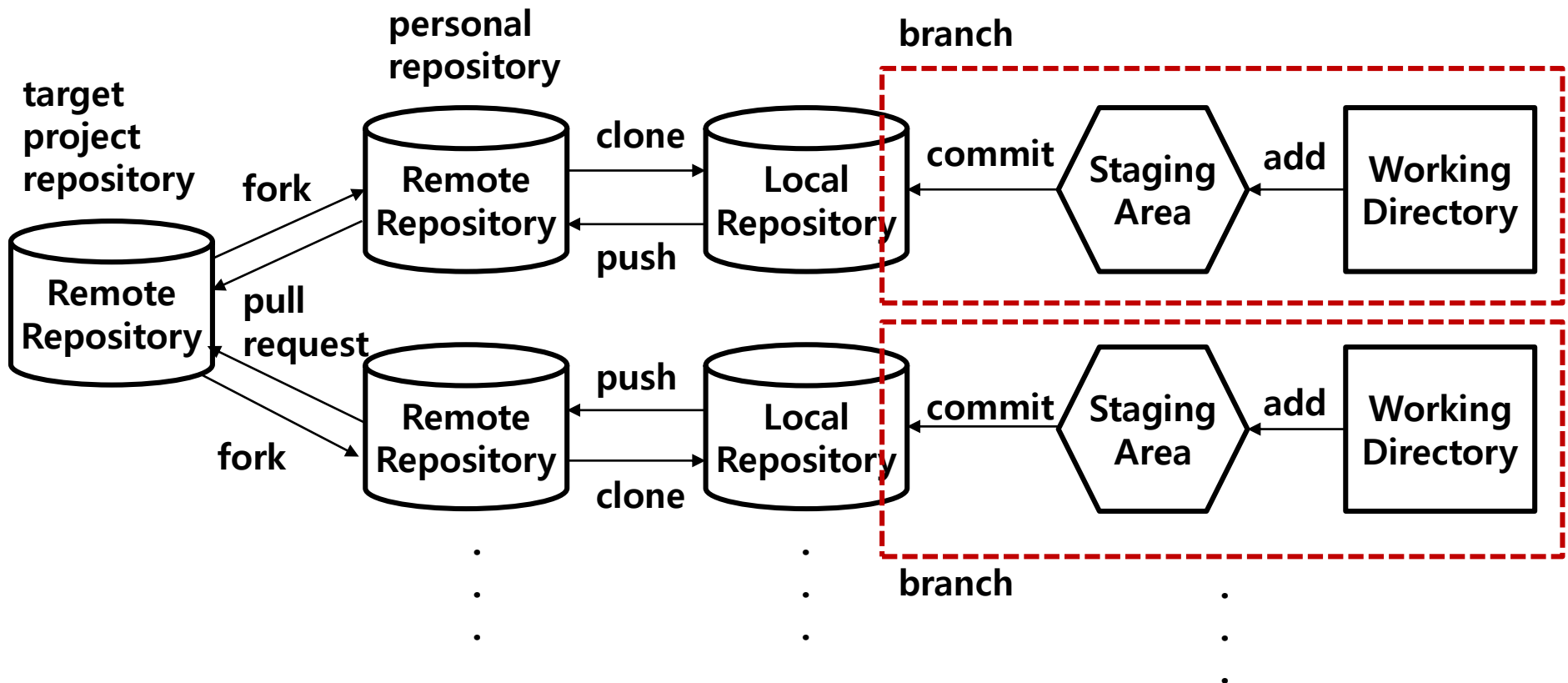
linuxer@linuxer-PC:~/lrpush$ git config --global push.default current

linuxer@linuxer-PC:~/lrpush$ git push
servername for 'https://github.com': oss-lab-2020-kit
Password for 'https://oss-lab-2020-kit@github.com':
오브젝트 나열하는 중: 4, 완료.
오브젝트 개수 세는 중: 100% (4/4), 완료.
Delta compression using up to 2 threads
오브젝트 압축하는 중: 100% (2/2), 완료.
오브젝트 쓰는 중: 100% (3/3), 265 bytes | 265.00 KiB/s, 완료.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/oss-lab-2020-kit/lrpush.git
72b3a3b..c49219f master -> origin
```

# 팀 프로젝트 작업 Workflow

## □ Workflow

- 타깃 프로젝트의 원격 저장소를 팀원 개인의 원격 저장소로 fork
- 팀원 개인의 원격 저장소를 clone 하여 로컬 저장소 생성
- 독립 작업을 위한 branch 생성 후 개발 진행

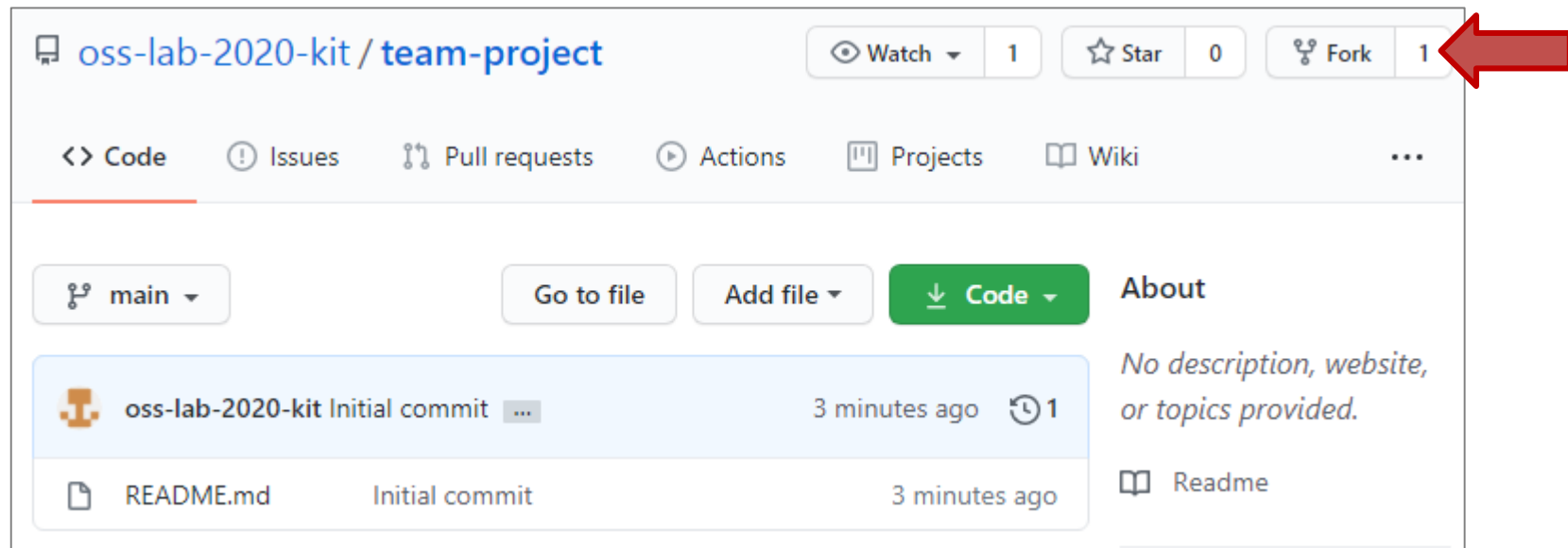




# 팀 프로젝트 작업 Workflow

## ❑ fork

- 팀원 개인 계정으로 github.com에 로그인
- 타깃 프로젝트 원격 저장소로 이동  
ex> `https://github.com/〈팀장계정〉/team-project`
- 오른쪽 상단의 "Fork" 버튼 클릭  
ex> `https://github.com/〈팀원계정〉/team-project`



# 팀 프로젝트 작업 Workflow

## ❑ clone

- fork 된 원격 저장소를 로컬 저장소로 복제

```
linuxer@linuxer-PC:~$ cd
linuxer@linuxer-PC:~$ git clone https://github.com/soyeum/team-project
linuxer@linuxer-PC:~$ cd team-project
linuxer@linuxer-PC:~/team-project$ ls
```

```
linuxer@linuxer-PC:~/team-project$ git remote -v
origin https://github.com/soyeum/team-project (fetch)
origin https://github.com/soyeum/team-project (push)
```

# 팀 프로젝트 작업 Workflow

## □ branch

- 팀원의 로컬 컴퓨터에서 코드를 추가하는 작업은 branch를 만들어서 진행
- 원래 코드와는 상관없이 독립적으로 개발을 진행할 수 있음

```
linuxer@linuxer-PC:~/team-project$ git checkout -b develop  
Switched to a new branch 'develop'
```

```
linuxer@linuxer-PC:~/team-project$ git branch  
* develop  
main
```

# 팀 프로젝트 작업 Workflow

## ❑ add, commit, push

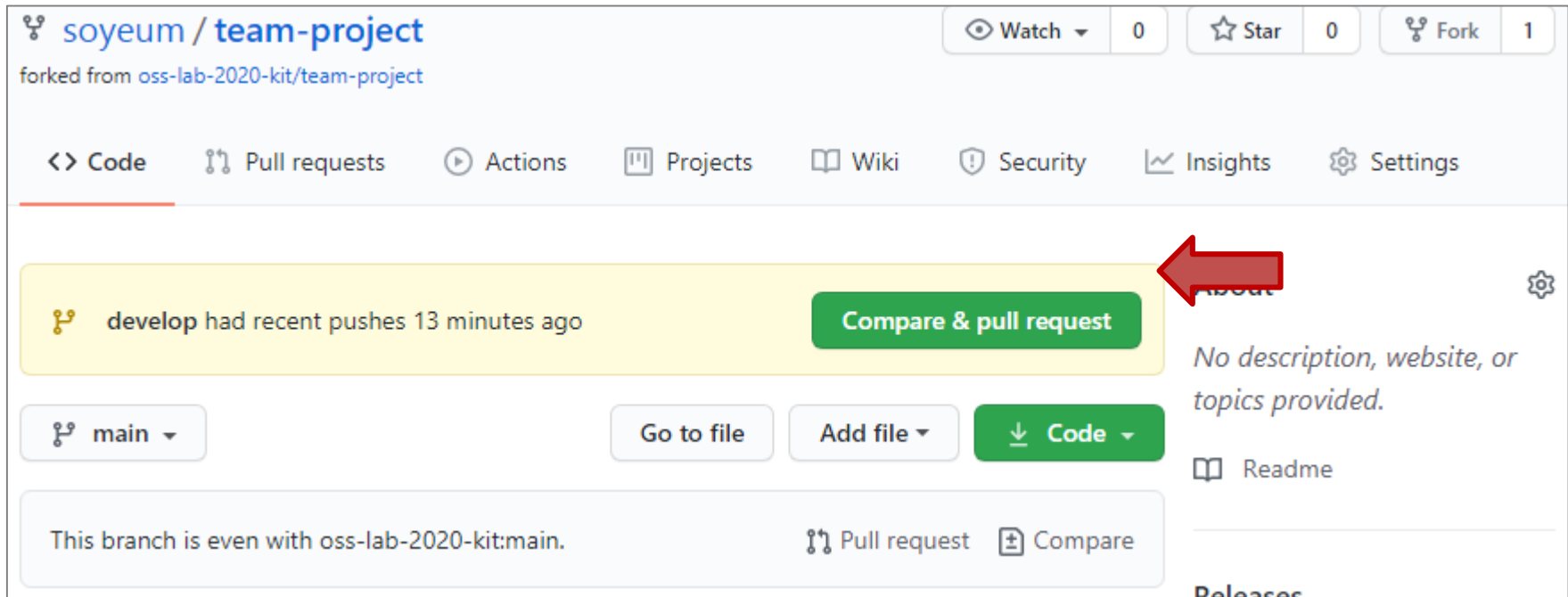
- 작업이 완료되면 로컬과 원격 저장소에 수정 사항을 반영

```
linuxer@linuxer-PC:~/team-project$ echo 'testing team project' > main.c
linuxer@linuxer-PC:~/team-project$ git add main.c
linuxer@linuxer-PC:~/team-project$ git commit -m 'main.c added'
linuxer@linuxer-PC:~/team-project$ git push origin develop
Username for 'https://github.com': soyeum
Password for 'https://soyeum@github.com': ****
오브젝트 나열하는 중: 4, 완료.
오브젝트 개수 세는 중: 100% (4/4), 완료.
Delta compression using up to 2 threads
오브젝트 압축하는 중: 100% (2/2), 완료.
오브젝트 쓰는 중: 100% (3/3), 288 bytes | 288.00 KiB/s, 완료.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/soyeum/team-project/pull/new/develop
remote:
To https://github.com/soyeum/team-project
* [new branch]      develop -> develop
```

# 팀 프로젝트 작업 Workflow

## □ pull request (PR 생성)

- 팀원 계정의 원격 저장소에 push된 것을 타깃 프로젝트 원격 저장소에 반영
- 변경된 내역 확인 없이 반영되면 기존 코드와의 충돌 문제 발생하기 때문에 팀장이 먼저 확인할 수 있도록 [compare & pull request] 진행
- 팀장 계정으로 들어가면 pull request 목록이 표시되고 승인 처리 가능



soyeum / team-project  
forked from oss-lab-2020-kit/team-project

Watch 0 Star 0 Fork 1

Code Pull requests Actions Projects Wiki Security Insights Settings

develop had recent pushes 13 minutes ago **Compare & pull request**

main Go to file Add file Code

This branch is even with oss-lab-2020-kit:main. Pull request Compare

# 팀 프로젝트 작업 Workflow

- ❑ merge 이후 동기화 및 branch 삭제
  - 승인 된다는 것은 merge 한다는 것임
  - 원본 저장소에 merge가 완료되면  
로컬 코드와 원본 저장소 코드를 동기화(필수)
  - 작업이 완료된 branch는 삭제(필수)

```
linuxer@linuxer-PC:~/team-project$ git branch
* develop
  main
linuxer@linuxer-PC:~/team-project$ git checkout main
linuxer@linuxer-PC:~/team-project$ git pull origin
linuxer@linuxer-PC:~/team-project$ git branch -d develop
```

# 타 프로젝트 참여 Workflow

## □ Workflow

- 진행 중인 오픈 소스 소프트웨어 개발 프로젝트에 참여하려 할 때는 각 프로젝트마다 가이드가 제공되고 있으니 숙지하고 접근할 필요있음
- 기본 동작은 [팀 프로젝트 작업 workflow]와 동일함
- pull request를 해도 곧바로 승인되지 않을 수 있음
- 기존 소스를 수정하는 것 보다,  
추가 기능을 구현해서 pull request 하는 것이 바람직함



# [참고] Git 명령 요약

## □ 자주 사용하는 명령어(1)

- **git init** : git 생성하기
- **git clone <git\_path>** : 코드가져오기
- **git checkout <branch\_name>** : 브랜치 선택하기
- **git checkout -t <remote\_path/branch\_name>** : 원격 브랜치 선택하기
- **git branch <branch\_name>** : 브랜치 생성하기
- **git branch -r** : 원격 브랜치 목록보기
- **git branch -a** : 로컬 브랜치 목록보기
- **git branch -m <branch\_name> <change\_branch\_name>**  
: 브랜치 이름 바꾸기





# [참고] Git 명령 요약

## □ 자주 사용하는 명령어(2)

- `git branch -d <branch_name>` : 브랜치 삭제하기
- `git add <file_path>` : 수정한 코드 선택하기 ( `git add *` )
- `git commit -m "commit_description"`  
: 선택한 코드 설명 적기 ( `git commit -m "내용"` )
- `git push <remote_name> <branch_name>`  
: add하고 commit한 코드 git server에 보내기(`git push origin master`)
- `git push <remote_name> --delete <branch_name>`  
: 원격 브랜치 삭제하기 ( ex. `git push origin --delete gh-pages` )
- `git pull` : git서버에서 최신 코드 받아와 merge 하기
- `git fetch` : git서버에서 최신 코드 받아오기



# [참고] Git 명령 요약

## □ 자주 사용하는 명령어(3)

- `git reset --hard HEAD^` : commit한 이전 코드 취소하기
- `git reset --soft HEAD^` : 코드는 살리고 commit만 취소하기
- `git reset --merge` : merge 취소하기
- `git reset --hard HEAD && git pull` : git 코드 강제로 모두 받아오기
- `git config --global user.name "user_name"` : git 계정명 변경하기
- `git config --global user.email "user_email"` : git 계정메일 변경하기
- `git stash / git stash save "description"`  
: 작업코드 임시저장하고 브랜치 바꾸기
- `git stash pop` : 마지막으로 임시저장한 작업코드 가져오기



# [복습] Git 활용

## □ Git command flow

