

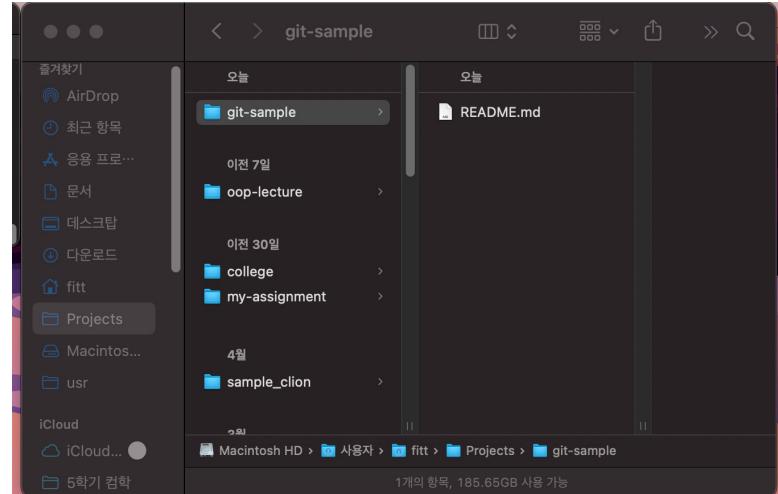
# 소프트웨어 공학 Git 과제

컴퓨터학부 20170517 이동재

GitHub Profile Url: <https://github.com/LouiMinister>  
GitHub Repository Url: <https://github.com/LouiMinister/git-sample>



## 1. 시나리오 시작



술실대에서 깃의 신이라고 불리는 깃신은 여름방학을 맞아 웹 계산기를 만드는 프로젝트를 진행하려 한다. 그래서 깃신은 다음과 같이 프로젝트를 만들고, 프로젝트에 대해서 설명을 하는 README.md 파일을 작성하였다.



## 2. 원격 레포지토리 생성

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

**Repository template**  
Start your repository with a template repository's contents.

No template ▾

**Owner** \*      **Repository name** \*

LouMinister / git-sample ✓

Great repository names are short and memorable. Need inspiration? How about [urban-broccoli](#)?

**Description** (optional)

**Public**  
Anyone on the internet can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**

Skip this step if you're importing an existing repository.

**Add a README file**  
This is where you can write a long description for your project. [Learn more](#).

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template: None ▾

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more](#).

License: None ▾

You are creating a public repository in your personal account.

**Create repository**

프로젝트를 만들고 나서, 깃신은 프로젝트의 버전관리를 위해 깃을 이용하기로 한다.

깃신은 자신의 프로젝트를 깃허브 원격저장소에 올리기 위해 깃허브 웹페이지에 접속하여 자신의 프로젝트 저장할 Repository를 생성한다.

프로젝트를 만들 때 깃신은 내 Repository를 다른 사람들도 볼 수 있도록 Public으로 설정을 하였고, 이미 readme 파일을 작성하였으므로 Add a README file 옵션은 선택하지 않았다.

깃신은 .gitignore파일을 이용하면 특정 파일을 Repository에 올릴 수 없도록 할 수 있음을 알았지만, 지금은 제외할 파일이 생각나지 않아서 .gitignore template를 None으로 선택하였다.

그리고 깃신은 모두가 저작권 문제에 엮이지 않고 접근할 수 있는 프로젝트를 만들려고 했기 때문에 라이센스도 None으로 선택하였다.

선택 옵션을 모두 고른 깃신은 Create repository 버튼을 클릭하였다.



### 3. 로컬 깃 저장소 생성

The screenshot shows a terminal window and a file browser side-by-side.

**Terminal Output:**

```
fitt@eastash-macpro:~/Projects/git-sample
fitt ~ cd Projects/git-sample
fitt ~/Projects/git-sample git init
한 트 : Using 'master' as the name for the initial branch. This default branch nam
e
한 트 : is subject to change. To configure the initial branch name to use in all
한 트 : of your new repositories, which will suppress this warning, call:
한 트 :
한 트 : git config --global init.defaultBranch <name>
한 트 :
한 트 : Names commonly chosen instead of 'master' are 'main', 'trunk' and
한 트 : 'development'. The just-created branch can be renamed via this command:
한 트 :
한 트 : git branch -m <name>
/Users/fitt/Projects/git-sample/.git/ 안의 빈 깃 저장소를 다시 초기화 했습니다
fitt ~/Projects/git-sample ✘ master ?
```

**File Browser:**

- Left sidebar: 즐겨찾기, AirDrop, 최근 항목, 응용 프로그램, 문서, 데스크탑, 다운로드, fitt, Projects, Macintosh HD, usr, iCloud, iCloud...
- Right sidebar: 오늘 (git-sample), 오늘 (README.md), 이전 7일 (oop-lecture), 이전 30일 (college, my-assignment), 4월 (sample\_clion), 2월 (Macintosh HD, 사용자, fitt, Projects, git-sample).
- Bottom status bar: 1개의 항목, 185.64GB 사용 가능

- 현재 프로젝트는 깃 저장소가 아니다.
- 깃신은 이 프로젝트를 깃 저장소로 만들고 싶다.
- 이 프로젝트를 깃 저장소로 만들기 위해 init 명령어를 사용하였다.
- init은 깃 저장소를 만들기 위해 사용하는 명령어이다.



## 4. 깃 저장소의 상태 확인하기

fitt@eastash-macpro:~/Projects/git-sample

```
fitt ~$ git status
master ? git status
현재 브랜치 master

아직 커밋이 없습니다

추적하지 않는 파일 :
  (커밋 할 사항에 포함하려면 "git add <파일>..."를 사용하십시오 )
    README.md

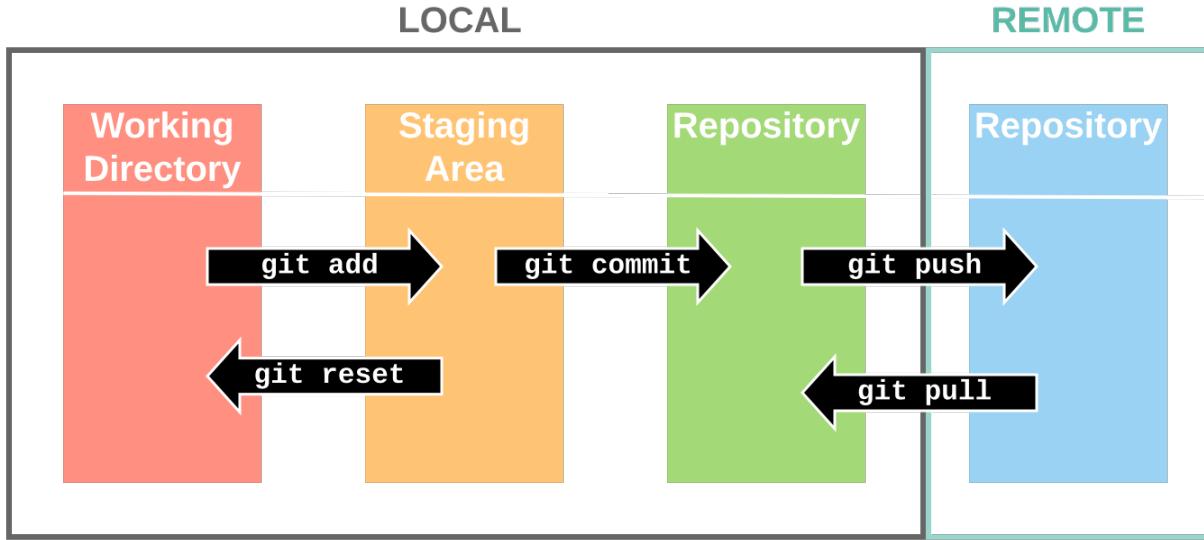
커밋 할 사항을 추가하지 않았지만 추적하지 않는 파일이 있습니다 (추적하려면 "git
add"를 사용하십시오 )
fitt ~$
```

- 현재 로컬 깃 저장소의 상태를 알 수 없다.
- 깃신은 현재 로컬 깃 저장소의 상태를 확인하고 싶다.
- 깃 저장소의 상태를 확인하기 위해 `status` 명령어를 실행하였다.
- `status`는 깃 저장소의 현재 상태(어떤 파일이 추가되어있는지, 어떤 커밋이 있는지 등)을 알 수 있는 명령어

README.md 파일이 git 저장소에 반영되지 않았음을 알 수 있다.



## Staging 상태



Staging 영역은 작업 디렉토리(로컬 디렉토리)와 Git 저장소 사이의 징검다리 역할을 한다.

깃 저장소에 저장될 파일들이 대기하는 장소이다.

Git add 명령어를 통해 작업 디렉토리에 있는 파일을 Staging 영역으로 이동시킬 수 있다.



## 5. 파일을 Staging 영역으로 이동시키기

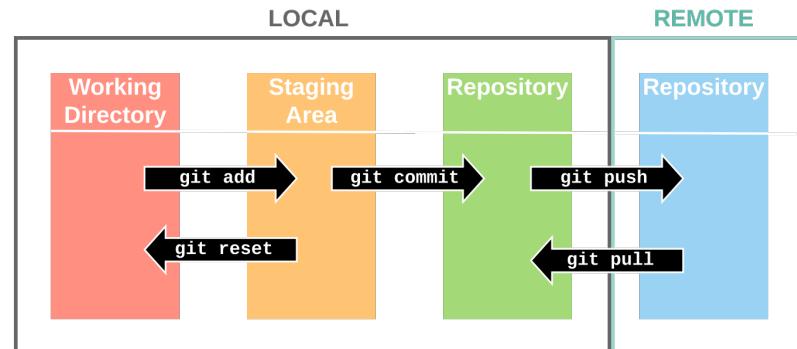
```
fitt@eastash-macpro:~/Projects/git-sample
13% 6.5 GB master * ~/Projects/git-sample
 README.md

커밋할 사항을 추가하지 않았지만 추적하지 않는 파일이 있습니다 (추적 하려면 "git add"를 사용하십시오)
fitt > ~/Projects/git-sample master ? git add README.md
fitt > ~/Projects/git-sample master + git status
현재 브랜치 master

아직 커밋이 없습니다

커밋할 변경 사항 :
(스테이지 해제하려면 "git rm --cached <파일>..."를 사용하십시오)
새 파일 : README.md

fitt > ~/Projects/git-sample master +
```

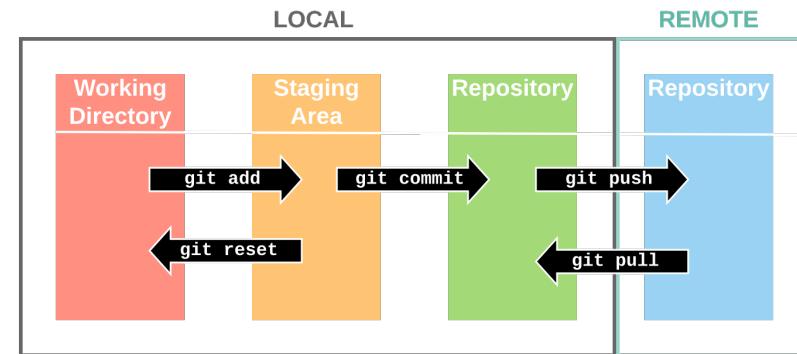


- 현재 README.md 파일은 깃 저장소에 반영되지 않았다.
- 깃신은 README.md 파일을 깃 저장소에 반영시키기 위해 파일을 Staging 영역으로 이동시키고 싶다..
- README.md 파일을 Staging 영역으로 이동시키기 위해 add 명령어를 사용하였다.
- add는 특정 파일을 Staging 영역으로 이동시킬 때 사용하는 명령어이다.
  - add . 으로 작업영역에 있는 모든 파일을 Staging Area로 옮길 수 있다.
  - add <파일명> 으로 특정 파일을 Staging Area로 옮길 수 있다.



## 6. 파일을 Repository 영역으로 이동시키기

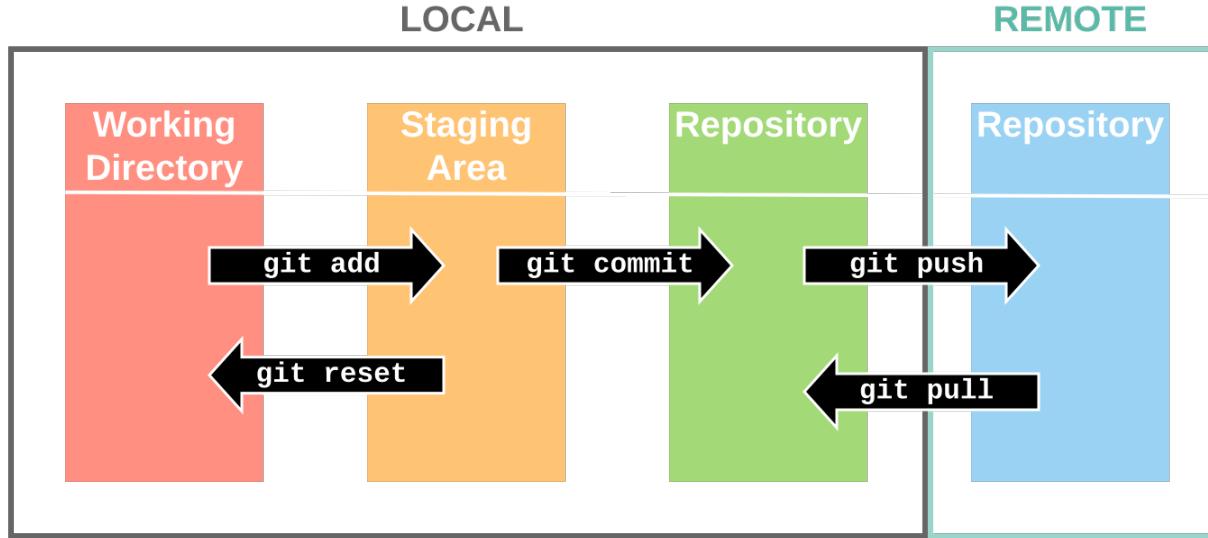
```
fitt@eastash-macpro:~/Projects/git-sample
[12%] 6.4 GB master ~/Projects/git-sample
fitt ~/Projects/git-sample master + git commit -m "first commit"
[master (최상위-커밋) 8b53136] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
fitt ~/Projects/git-sample master git status
현재 브랜치 master
커밋 할 사항 없음, 작업 풀더 깨끗함
fitt ~/Projects/git-sample master
```



- 현재 README.md 파일은 깃 저장소에 반영되지 않았다.
- 깃신은 README.md 파일을 깃 저장소에 반영시키기 위해 Staging Area 영역의 파일을 깃 저장소에 이동시키고 싶다.
- README.md 파일을 깃 저장소 영역으로 이동시키기 위해 commit 명령어를 사용하였다.
- commit은 Staging Area 영역의 파일들을 깃 저장소 영역으로 이동시킬 때 사용하는 명령어이다.
  - -m 옵션을 이용해 인자로 커밋 메시지를 전달할 수 있다.



## 깃 저장소에서 파일들의 영역



커밋을 하고나면 왜 깃에서 Staging Area라는 중간 단계를 만들었는지 알 수 있다.

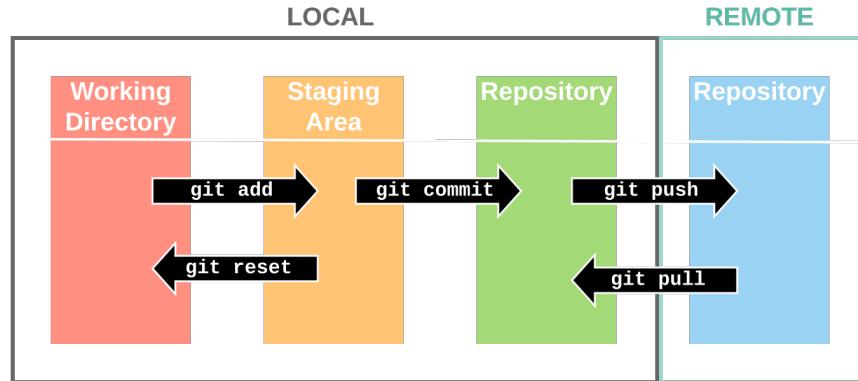
### Staging 영역의 존재 이유

- 저장소의 변경 이력을 남길 때 각각의 파일의 변경이 아니라 논리적인 변경 이력을 남길 수 있다.
- (파일 하나 수정 -> 추가 기능 구현 (여러파일수정))
- 따라서 버그를 추적하거나 변경이력을 뒤집하기 쉽다
- 변경 이력들을 확인하기 쉽다.



## 7. 원격 Repository에 반영하기

```
fitt > ~/Projects/git-sample > master > git config --global user.name  
LouiMinister  
fitt > ~/Projects/git-sample > master > git config --global user.email  
louiminister@gmail.com  
fitt > ~/Projects/git-sample > master > |
```



- 로컬 깃 저장소의 변경 사항은 원격 깃 저장소에 반영되어 있지 않다.
- 깃신은 로컬 깃 저장소의 변경 사항을 원격 깃 저장소에 반영하고 싶지만 서명이 되어있지 않아 원격 저장소에 접근하지 못한다. 따라서 서명을 하려한다.
- `git config global user.name`, `git config global user.email` 명령어를 사용하여 깃허브 이름과 이메일을 서명하였다.
- `git config` 명령을 사용하여 깃 설정 정보를 조회, 수정할 수 있고 이를 통해 계정정보를 서명할 수 있다.



## 8. 원격 Repository 연동하기

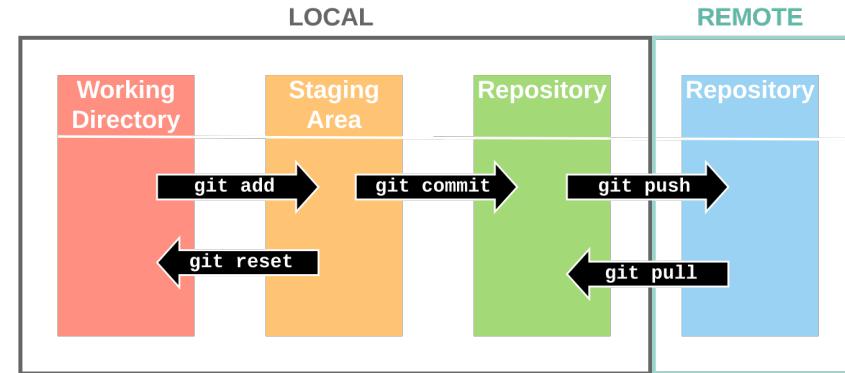
```
fitt@eastash-macpro:~/Projects/git-sample
fitt > ~/Projects/git-sample [master] git remote add origin https://github.com/LouiMinister/git-sample.git
fitt > ~/Projects/git-sample [master] git remote get-url origin
https://github.com/LouiMinister/git-sample.git
fitt > ~/Projects/git-sample [master]
```

- 로컬 깃 저장소의 변경 사항은 원격 깃 저장소에 반영되어 있지 않다.
- 깃신은 로컬 깃 저장소의 변경 사항을 원격 깃 저장소에 반영하고 싶지만 로컬 깃 저장소와 원격 깃 저장소가 연동되어 있지 않다.
- `git remote add origin` 명령을 사용하여 해당 원격 저장소를 연동하였다.
- `git remote add <원격저장소 이름> <원격 깃 저장소의 주소>`: 로컬 깃 저장소를 원격 깃 저장소에 연결하기 위해 사용하는 명령어
- `Git remote get-url <원격저장소 이름>`: <원격 저장소 이름>에 해당하는 원격 저장소의 url을 조회하는 명령어  
기본 원격 저장소의 이름은 보통 `origin`을 사용한다.



## 9. 원격 Repository에 반영하기

```
fitt@eastash-macpro:~/Projects/git-sample
[12%] 5.9 GB master ~/Projects/git-sample
fitt > ~/Projects/git-sample > master > git push origin master
오브젝트 나열하는 중 : 3, 완료 .
오브젝트 개수 세는 중 : 100% (3/3), 완료 .
오브젝트 쓰는 중 : 100% (3/3), 226 bytes | 226.00 KiB/s, 완료 .
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/LouiMinister/git-sample.git
 * [new branch] master -> master
fitt > ~/Projects/git-sample > master >
```



- 로컬 깃 저장소의 변경 사항은 원격 깃 저장소에 반영되어 있지 않다.
- 깃신은 로컬 깃 저장소의 변경 사항을 원격 깃 저장소에 반영하고 싶다.
- git push 명령어를 이용하여 로컬 저장소의 변경 이력(commit)을 원격 저장소에 반영하였다.
- git push <원격 저장소 이름> <브랜치 이름>: 로컬 저장소의 commit을 원격 저장소에 반영하도록 하는 명령어. 이전 설명에서는 원격 저장소의 이름을 origin으로 설정하였고, 기본 브랜치 이름이 master였기 때문에 git push origin master을 입력하였다.



## 9. 원격 Repository에 반영하기

The screenshot shows a GitHub repository page for 'LouiMinister/git-sample'. The repository is public and has 1 branch and 0 tags. The 'Code' tab is selected. A commit from 'LouiMinister' titled 'first commit' was made 36 minutes ago, with a commit ID of 8b53136. The commit message is 'first commit' and it was committed 1 hour ago. Below the commit, there is a link to 'DEARME.md'. The commit history shows a single commit on May 15, 2022, with the same details. Navigation buttons for 'Newer' and 'Older' commits are at the bottom.

- 원격 저장소를 확인해보니 정상적으로 반영됨을 확인할 수 있었음.



## 10. + 연산 기능 구현

The terminal window shows the following command sequence:

```
fitt@eastash-macpro:~/Projects/git-sample
fitt ~$ git add source.md
fitt ~$ git commit -m "+연산 기능 구현"
[master 9442b36] +연산 기능 구현
 1 file changed, 3 insertions(+)
 create mode 100644 source.md
fitt ~$ git push origin master
Branch 'master' up-to-date
  오브젝트 나열하는 중: 4, 완료.
  오브젝트 개수 세는 중: 100% (4/4), 완료.
  Delta compression using up to 10 threads
  오브젝트 압축하는 중: 100% (2/2), 완료.
  오브젝트 쓰는 중: 100% (3/3), 346 bytes | 346.00 KiB/s, 완료.
  Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
  To https://github.com/LouiMinister/git-sample.git
    8b53136..9442b36  master -> master
fitt ~$
```

The file browser shows the directory structure and the contents of the source.md file.

어제	오늘
<code>git-sample</code>	<code>README.md</code>
이전 7일	어제
<code>oop-lecture</code>	<code>source.md</code>

source.md content:

```
# Source
#
### + 연산 기능 구현
```

- 깃신은 본격적으로 프로젝트를 진행하기 위해 구현 파일(source.md)을 생성하고 계산기의 +연산 기능을 구현하였다.
- 이후 구현을 깃 원격 저장소에 반영하기 위해 add, commit, push 명령어를 순서대로 실행하였다.



## 11. 다른 작업자의 참여 - 깃 저장소 복제

```
fitt@eastash-macpro:~/Projects/git-sample-gitsun/git-sample
git clone https://github.com/LouiMinister/git-sample.git
'git-sample'에 복제합니다...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
오브젝트를 받는 중: 100% (6/6), 완료.
fitt@eastash-macpro:~/Projects/git-sample-gitsun/git-sample
fitt@eastash-macpro:~/Projects/git-sample-gitsun/git-sample
```

깃신이 계산기 프로젝트를 진행해 나가고 있을 때, 깃신의 여동생인 깃순이가 자신도 프로젝트에 참여하고 싶다는 의사를 밝혔다. 따라서 깃신은 깃순이에게 깃허브 원격 저장소(Repository) 주소를 알려주었고, 깃순이는 저장소를 받아오려고 한다.

- 깃순이가 깃 저장소를 받아오지 않은 상태
- 깃순이는 깃 저장소를 받아오려고 함
- 깃순이는 해당 레포지토리를 clone 명령어를 이용해 원격 깃 저장소를 자신의 로컬 환경으로 가져옴
- Git clone <repository URL>: 원격 깃 저장소를 현재 환경(로컬 환경)으로 복제(clone)함.



## 12. 깃 브랜치 생성, 브랜치 옮기기

The screenshot shows a terminal window with the following session:

```
fitt@eastash-macpro:~/Projects/git-sample-gitsun/git-sample
 11% 8.9 GB feature1 ~/P/g/git-sample
fitt > ~/Projects/git-sample-gitsun/git-sample ✘ master git branch feature1
fitt > ~/Projects/git-sample-gitsun/git-sample ✘ master git checkout featur
e1
'feature1' 브랜치로 전환합니다
fitt > ~/Projects/git-sample-gitsun/git-sample ✘ feature1
```

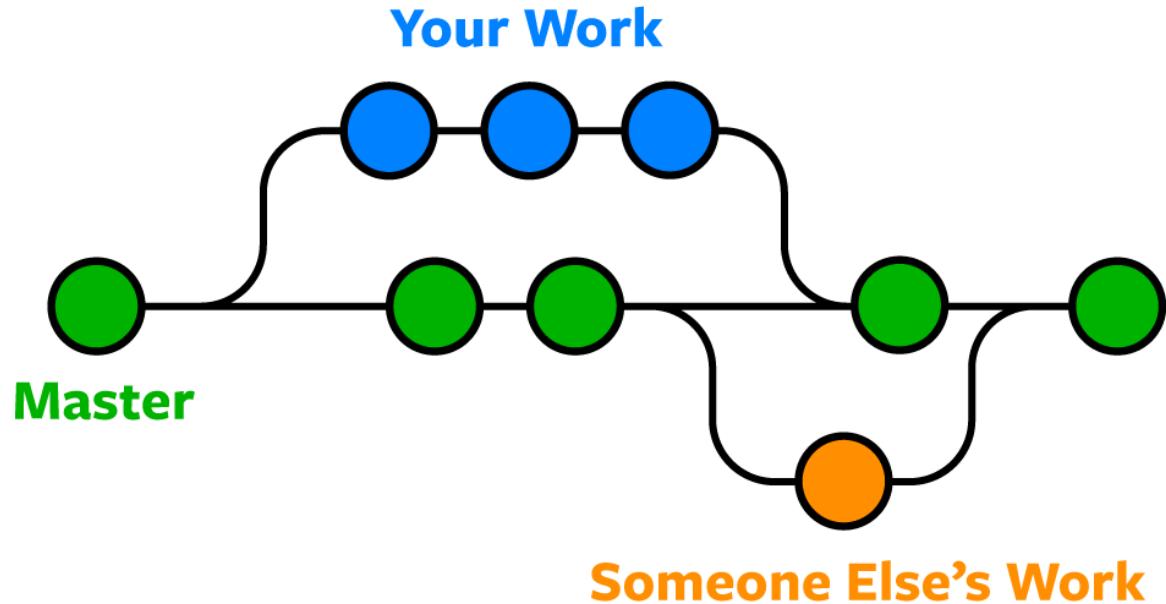
깃순이는 클론을 받은 후, 깃순이는 연산 기능 구현을 담당하기로 함. - 기능 구현 브랜치를 따로 관리하기 위해 feature1이라는 브랜치를 ‘branch’ 명령어를 이용하여 만들고 로컬 프로젝트의 브랜치를 feature1로 ‘checkout’ 명령어를 이용해 이동하였다.

- master 브랜치에 있는 상태
- - 기능 구현 브랜치 feature1을 따로 생성하고 그 곳에서 작업하고 싶음.
- git branch 명령을 통해 새로운 브랜치를 지금 브랜치 기준으로 생성하고 checkout 명령어를 통해 생성한 브랜치로 이동함.
- git branch <브랜치이름>: 새로운 브랜치를 작성
- git branch: 브랜치 목록 전체 확인
- git branch d <브랜치이름>: 해당 브랜치 삭제
- git checkout <브랜치이름>: 해당 브랜치로 이동

브랜치 개념에 대해서는 후술함.



## 깃 브랜치 개념



브랜치란 독립적으로 어떤 작업을 진행하기 위한 개념.

필요에 의해 만들어지는 각각의 브랜치는 다른 브랜치의 영향을 받지 않기 때문에, 여러 작업을 동시에 진행할 수 있음.

또한 이렇게 만들어진 브랜치는 다른 브랜치와 병합(Merge)함으로써, 작업한 내용을 다시 새로운 하나의 브랜치로 모을 수 있음.



## 13. 공용함수 util 구현

```
fitt ~/Projects/git-sample ✘ master • ? git add source.md
fitt ~/Projects/git-sample ✘ master + ? git commit
[master 9ca2ca2] 공용 함수 util() 구현
 1 file changed, 2 insertions(+)
fitt ~/Projects/git-sample ✘ master ? git push origin master
오브젝트 나열하는 중: 5, 완료.
오브젝트 개수 세는 중: 100% (5/5), 완료.
Delta compression using up to 10 threads
오브젝트 압축하는 중: 100% (3/3), 완료.
오브젝트 쓰는 중: 100% (3/3), 368 bytes | 368.00 KiB/s, 완료.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/LouiMinister/git-sample.git
 9442b36..9ca2ca2 master -> master
fitt ~/Projects/git-sample ✘ master ? []
```

```
git-sample >
source
source > 공용함수 util() 구현
# Source
#####
 3 ##### 공용함수 util() 구현
 4
 5 ##### + 연산 기능 구현
 6
```

깃순이가 브랜치를 만드는 동안, 깃신은 프로젝트 구현에 도움이 될 공통함수 `util()`을 구현하고 원격 저장소에 반영하였다. 깃신은 깃순이에게 이 커밋을 깃순이가 만든 브랜치에도 반영하고, 이 함수를 이용해 기능을 구현할 것을 요구하였다.



## 14. cherry-pick 으로 다른 브랜치 커밋 가져오기

The screenshot shows the GitHub interface for the 'master' branch. It displays two sets of commits:

- Commits on May 16, 2022:**
  - 공용함수 util() 구현 by LouiMinister committed 10 minutes ago (commit hash: 9ca2ca2)
  - +연산 기능 구현 by LouiMinister committed 41 minutes ago (commit hash: 9442b36)
- Commits on May 15, 2022:**
  - first commit by LouiMinister committed 2 days ago (commit hash: 8b53136)

The terminal session shows the following commands and their output:

```
fitt@eastash-macpro:~/Projects/git-sample-gitsun/git-sample % git remote update  
origin을 (를) 가져오는 중  
remote: Enumerating objects: 5, done.  
remote: Counting objects: 100% (5/5), done.  
remote: Compressing objects: 100% (3/3), done.  
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0  
오브젝트 루프 푸는 중 : 100% (3/3), 348 bytes | 116.00 KiB/s, 완료 .  
https://github.com/LouiMinister/git-sample URL에서  
9442b36..9ca2ca2 master -> origin/master  
fitt > ~/Projects/git-sample-gitsun/git-sample % git cherry-pick 9ca207c93219bc60b259930933d6e43c3a0d  
[feature1 4e4d3a] 공용 함수 util() 구현  
Date: Mon May 16 19:37:19 2022 +0900  
1 file changed, 2 insertions(+)  
fitt > ~/Projects/git-sample-gitsun/git-sample %
```

On the right side of the terminal, there is a code editor window showing a file named 'source'. The code contains the following:

```
1 # Source  
2  
3 ### 공용함수 util() 구현  
4  
5 ### + 연산 기능 구현  
6
```

- 깃인이 반영하라는 커밋은 master 브랜치에 있고 현재 브랜치는 feature1인 상태. 원격 저장소의 정보가 최신정보가 아님
- 원격 저장소에 대한 정보를 최신 정보로 업데이트하고, master 브랜치에 있는 한 커밋을 feature1 브랜치에 반영하려고 함
- git remote 명령어로 원격 저장소에 대한 정보를 최신 정보로 업데이트함. 그리고 git cherry-pick 명령어와 커밋의 해시값을 이용하여 해당 커밋을 현재 브랜치에 반영함
- git remote update: 모든 원격 브랜치들을 업데이트 하여 최신 상태로 갱신한다. (현재는 로컬에서 해시값에 해당하는 원격 저장소의 커밋을 찾기 위해 원격 브랜치를 최신 상태로 갱신함)
- git cherry-pick <commit의 hash값1> <commit의 hash값2> ...: 다른 브랜치에 있는 해시에 해당하는 커밋을 현재 브랜치에 반영. 여러 커밋 반영 가능
- continue 옵션을 사용하면 conflict 발생 시 conflict를 해결하고 cherry-pick을 진행함
- abort 옵션을 사용하면 conflict 발생 시 명령어 실행 이전으로 되돌아감



## 15. stash 로 현재 작업 영역 파일들 임시 저장하기

The screenshot shows a macOS desktop environment. On the left is a terminal window titled 'fitt@eastash-macpro:~/Projects/git-sample'. It displays the output of a 'git status' command, which shows a 'feature2' branch with changes staged for commit. The user then runs 'git stash' to save the changes. On the right is a Xcode project interface with two files visible: 'source' and '.DS\_Store'. The 'source' file contains code related to a 'util()' function.

```
fitt@eastash-macpro:~/Projects/git-sample
..un/git-sample (-zsh) ⌘1 ..ts/git-sample (-zsh) ⌘2 +
9.6 GB (feature2+*) ~ /Projects/git-sample
fitt ➤ ~/Projects/git-sample ⌘ feature2 • ? git status
현재 브랜치 feature2
커밋 하도록 정하지 않은 변경 사항 :
(무엇을 커밋할지 바꾸려면 "git add <파일>...."을 사용하십시오 )
(use "git restore <file>..." to discard changes in working directory)
수정함 : source.md
추적하지 않는 파일 :
(커밋 할 사항에 포함하려면 "git add <파일>...."을 사용하십시오 )
.DS_Store

커밋 할 변경 사항을 추가하지 않았습니다 ("git add" 및 / 또는 "git commit -a"를 사용하십시오 )
fitt ➤ ~/Projects/git-sample ⌘ feature2 • ? git stash
Saved working directory and index state WIP on feature2: 9ca2ca2 공용함수 util()
구현
fitt ➤ ~/Projects/git-sample ⌘ feature2 ? ⌘1
```

# Source  
## 공용함수 util() 구현  
## + 연산 기능 구현

- 그동안 깃신은 \*연산 기능을 구현하고 있었고, 아직 커밋하기 깔끄러운 상태에서 master에서 구현한 util()함수에 버그가 존재한다는 사실을 알아냄. 지금까지의 수정작업은 끝난 작업이 아니라 커밋하기는 깔끄러운 상태임.
- 작업 영역의 파일들을 커밋하지 않고 임시 저장해놓고 싶음.
- git stash 연산을 이용하여 지금까지 작업한 내용을 임시 스택에 저장함.
- git stash: 새로운 stash를 스택에 만들어 하던 작업을 임시로 저장한다. 변경사항을 commit을 사용하지 않고 임시 저장할 때 사용
- git stash pop: 스택에 임시 저장한 파일들을 pop 하는데 사용
- git stash list: stash stack에 저장된 파일들을 확인하는데 사용



## 16. util 함수 수정해서 재 반영

The terminal window shows the following command sequence:

```
fitt@eastash-macpro:~/Projects/git-sample
..un/git-sample (-zsh) %1 ..ts/git-sample (-zsh) %2 +
9% 9.3 GB master+ *
fitt ~/Projects/git-sample [master • ? @1] git add source.md
fitt ~/Projects/git-sample [master + ? @1] git commit -m "공용 함수 util"
버그수정"
[master acf62b5] 공용 함수 util 버그수정
1 file changed, 1 insertion(+), 1 deletion(-)
fitt ~/Projects/git-sample [master ? @1] git push origin master
오브젝트 나열하는 중: 5, 완료.
오브젝트 개수 세는 중: 100% (5/5), 완료.
Delta compression using up to 10 threads
오브젝트 압축하는 중: 100% (3/3), 완료.
오브젝트 쓰는 중: 100% (3/3), 396 bytes | 396.00 KiB/s, 완료.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/LouiMinister/git-sample.git
 9ca2ca2..acf62b5  master -> master
fitt ~/Projects/git-sample [master ? @1]
```

The code editor window shows a file named 'source' with the following content:

```
1 # Source
2
3 ### 공용함수 util() 구현, 버그수정 by 깃신
4
5 ### + 연산 기능 구현
6 |
```

그리고 깃신은 branch를 master로 바꾸고 util 함수를 수정해서 재 반영하였다.



## 17. stash pop 이용하여 저장한 임시 파일 꺼내기

The terminal window shows the following session:

```
fitt@eastash-macpro:~/Projects/git-sample
..un/git-sample (-zsh)    *1 | ..ts/git-sample (-zsh)    *2 +
7%                         9.2 GB                         ~ /Projects/git-sample
fitt > ~/Projects/git-sample > master ? @1 git checkout feature2
'feature2' 브랜치로 전환합니다
fitt > ~/Projects/git-sample > feature2 ? @1 git stash pop
현재 브랜치 feature2
커밋하도록 정하지 않은 변경 사항:
(무엇을 커밋할지 바꾸려면 "git add <파일>..."를 사용하십시오)
(use "git restore <file>..." to discard changes in working directory)
수정함: source.md

추적하지 않는 파일:
(커밋할 사항에 포함하려면 "git add <파일>..."를 사용하십시오)
.DS_Store

커밋할 변경 사항을 추가하지 않았습니다 ("git add" 및 / 또는 "git commit -a"를 사용하십시오)
Dropped refs/stash@{0} (b3dc68082477cd1f825c6a9bb71d500eb7ed5295)
```

A code editor window is visible on the right, showing a file named `source` with the following content:

```
1 # Source
2
3 ### 공용함수 util() 구현
4
5 ### + 연산 기능 구현
6
7 ### * 연산 기능 구현 중...
8
```

- 원래 작업하던 브랜치에 돌아온 상태
- stash로 저장한 파일들을 다시 꺼내고 싶음.
- git stash pop 명령어를 이용해 임시 저장한 파일들을 꺼냄
- git stash: 새로운 stash를 스택에 만들어 하던 작업을 임시로 저장한다. 변경사항을 commit을 사용하지 않고 임시 저장할 때 사용
- git stash pop: 스택에 임시 저장한 파일들을 pop 하는데 사용
- git stash list: stash stack에 저장된 파일들을 확인하는데 사용



## 18. git reset 으로 작업 디렉토리 영역 초기화하기

The terminal session shows the following steps:

```
fitt ~/Projects/git-sample > feature2 ? git reset --hard
fitt ~/Projects/git-sample > feature2 ? git add source.md
fitt ~/Projects/git-sample > feature2 + ? git commit
[feature2 c96dbf8] * 연산 기능 처음부터 다시 구현
  1 file changed, 2 insertions(+)
fitt ~/Projects/git-sample > feature2 ? git push origin feature2
오브젝트 나열하는 중: 5, 완료.
오브젝트 개수 세는 중: 100% (5/5), 완료.
Delta compression using up to 10 threads
오브젝트 압축하는 중: 100% (3/3), 완료.
오브젝트 쓰는 중: 100% (3/3), 417 bytes | 417.00 KiB/s, 완료.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/LouiMinister/git-sample.git
  9ca2ca2..c96dbf8  feature2 -> feature2
```

On the right, the source code for the 'source' file is shown:

```
1 # Source
2
3     ### 공용함수 util() 구현
4
5     ### + 연산 기능 구현
6
7     ### * 연산 기능 처음부터 다시 구현
8
```

Git stash pop을 이용해 원래 작업하던 소스를 받아온 깃신은 원래 수정하는 코드를 이해하는 것보다 처음부터 다시 만드는게 빠르다는 판단이 들었다.

- 작업 영역에 전에 작업하다 만 파일들이 있는 상태
- 작업 영역에 있는 파일들을 모두 없애서 가장 최근 커밋과 같은 상태로 만들고 싶음.
- git reset --hard 명령어를 이용해 작업 영역에 있는 파일을 모두 없앰
- git reset --hard HEAD: 워킹 디렉터리를 원격 저장소의 마지막 commit 상태로 되돌린다.
- git reset HEAD <파일이름>: 해당 파일의 git add를 취소한다. (Staging 상태에서 뺀다)
- git reset -soft HEAD^: commit을 취소하고 해당 파일들은 staged 상태로 워킹 디렉터리에 보존
- git reset mixed HEAD^: commit을 취소하고 해당 파일들은 unstaged 상태로 워킹 디렉터리에 보존 (디폴트)
- git reset hard HEAD^: commit을 취소하고 해당 파일들은 unstaged 상태로 워킹 디렉터리에서 삭제
- git reset HEAD~2 // 마지막 2개의 commit을 취소



## 18. git reset 으로 작업 디렉토리 영역 초기화하기

The terminal window shows the following sequence of commands:

```
fitt ~/Projects/git-sample > feature2 ? git reset --hard
fitt ~/Projects/git-sample > feature2 ? git add source.md
fitt ~/Projects/git-sample > feature2 + ? git commit
[feature2 c96dbf8] * 연산 기능 처음부터 다시 구현
1 file changed, 2 insertions(+)
fitt ~/Projects/git-sample > feature2 ? git push origin feature2
```

Output of the push command:

```
오브젝트 나열하는 중: 5, 완료.
오브젝트 개수 세는 중: 100% (5/5), 완료.
Delta compression using up to 10 threads
오브젝트 압축하는 중: 100% (3/3), 완료.
오브젝트 쓰는 중: 100% (3/3), 417 bytes | 417.00 KiB/s, 완료.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/LouiMinister/git-sample.git
  9ca2ca2..c96dbf8  feature2 -> feature2
```

On the right, the source code for 'source.md' is shown:

```
1 # Source
2
3     ### 공용함수 util() 구현
4
5     ### + 연산 기능 구현
6
7     ### * 연산 기능 처음부터 다시 구현
8
```

이후 \* 연산 기능을 처음부터 다시 만들고 feature2 브랜치에 반영함. 이후 원격 저장소에 반영



## 19. git rebase 으로 브랜치 병합 시 커밋 로그 정렬하기

The terminal shows the following sequence of commands and their results:

```
fitt ~/Projects/git-sample $ feature2 ? git branch feature3
fitt ~/Projects/git-sample $ feature2 ? git checkout feature3
'feature3' 브랜치로 전환합니다
fitt ~/Projects/git-sample $ feature3 ? git add source.md
fitt ~/Projects/git-sample $ feature3 +? git commit
[feature3 c460aa7] / 연산 기능 구현
1 file changed, 2 insertions(+)
fitt ~/Projects/git-sample $ feature3 ? git rebase feature2
Current branch feature3 is up to date.
fitt ~/Projects/git-sample $ feature3 ? git checkout feature2
'feature2' 브랜치로 전환합니다
fitt ~/Projects/git-sample $ feature2 ? git merge feature3
업데이트 중 c96dbf8..c460aa7
Fast-forward
 source.md | 2 ++
 1 file changed, 2 insertions(+)
fitt ~/Projects/git-sample $ feature2 ?
```

On the right, the resulting commit log is shown:

```
1 # Source
2
3 ### 공용함수 util() 구현
4
5 ### + 연산 기능 구현
6
7 ### * 연산 기능 처음부터 다시 구현
8
9 ### / 연산 기능 구현
10
```

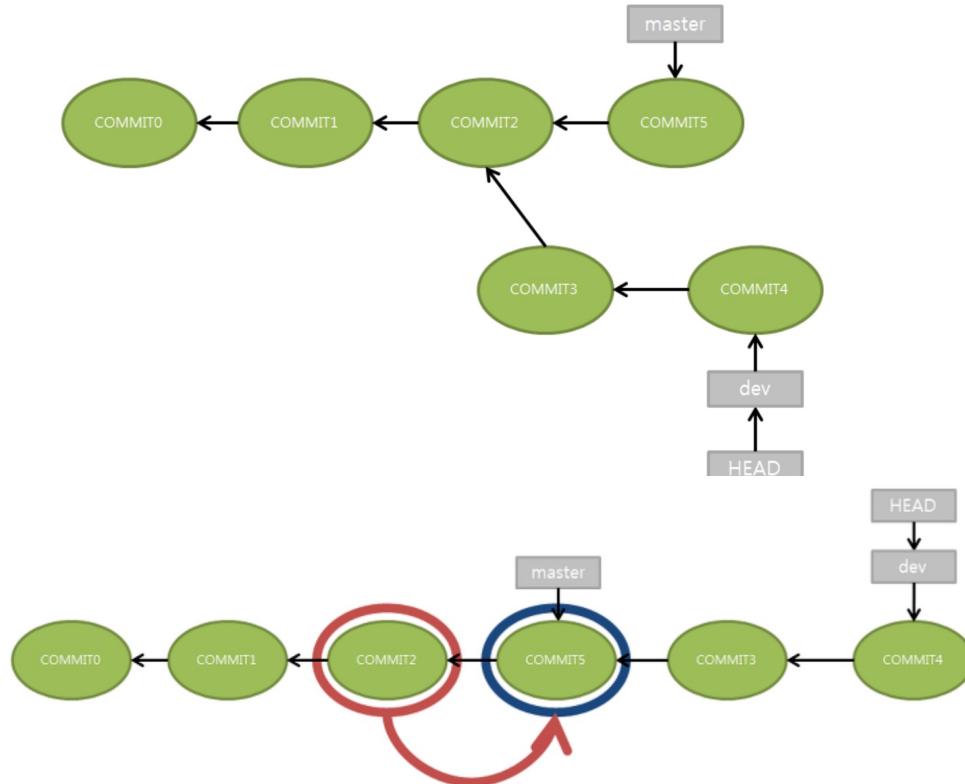
```
1 # Source
2
3 ### 공용함수 util() 구현
4
5 ### + 연산 기능 구현
6
7 ### * 연산 기능 처음부터 다시 구현
8
9 ### / 연산 기능 구현
10
```

깃신은 이제 / 연산 기능을 구현하기 위해 feature3 브랜치를 생성하였다, feature3 브랜치로 이동하여 /연산 기능을 구현한 후 커밋을 완료하였다. 이제 이 브랜치를 master 브랜치에 병합하려고 하였는데, 그것보다 feature2의 마지막 커밋을 기준으로 하여 새로 추가된 feature3의 커밋을 feature2 브랜치에 연결하는게 깃 히스토리가 더 깔끔해 질 것 같다는 생각이 들었다.

- feature2, feature3 브랜치에서 각각 작업을 완료한 상태
- feature2의 마지막 커밋을 기준으로 feature3의 커밋들이 오도록 정렬하고 병합하고싶음.
- git rebase 명령어를 이용하여 커밋들을 기준에 맞게 정렬하고 git merge 명령어를 이용하여 두 브랜치를 병합함.
- git merge <브랜치 이름>: 깃 브랜치를 병합할 때 사용. 현재 브랜치에 인자로 준 브랜치를 병합한다.
- git rebase <브랜치 이름>: 깃 브랜치의 base를 다시 설정함. 두 브랜치를 병합하게 되면 커밋들이 시간순으로 정렬되어 병합하게 되는데 rebase를 사용하면 병합 시 인자로 준 브랜치의 마지막 커밋 이후에 현재 브랜치의 커밋이 오도록 커밋 기록을 정렬함  
(뒷 장의 그림 참조)



## rebase 사용 시 커밋의 정렬 상황



rebase를 하게되면 두 브랜치의 커밋들의 시간과 상관없이  
base를 기준으로 커밋들이 정렬됨



## 20. feature2 -> master 병합 후 원격 저장소에 반영

```
fitt > ~/Projects/git-sample > feature2 ? git checkout master  
'master' 브랜치로 전환합니다.  
fitt > ~/Projects/git-sample > master ? git merge feature2  
자동 병합: source.md  
Merge made by the 'recursive' strategy.  
source.md | 4 +---  
1 file changed, 4 insertions(+)  
fitt > ~/Projects/git-sample > master ? git push origin master  
오브젝트 나열하는 중: 10, 완료.  
오브젝트 개수 세는 중: 100% (10/10), 완료.  
Delta compression using up to 10 threads  
오브젝트 압축하는 중: 100% (6/6), 완료.  
오브젝트 쓰는 중: 100% (6/6), 691 bytes | 691.00 KiB/s, 완료.  
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (2/2), completed with 1 local object.  
To https://github.com/LouiMinister/git-sample.git  
 acf62b5..26ea7a1 master -> master
```

```
1 # Source  
2  
3 ##### 공용함수 util() 구현, 버그수정 by 깃신  
4  
5 ##### + 연산 기능 구현  
6  
7 ##### * 연산 기능 처음부터 다시 구현  
8  
9 ##### / 연산 기능 구현  
10
```

이후 깃신은 feature2 브랜치를 master 브랜치에 merge 명령어를 이용해 병합한 후,  
git push origin master 명령어를 이용해 원격 저장소의 master 브랜치에 반영하였다.



## 21. git log 명령어로 커밋들의 기록 조회

```
commit 26ea7a14679a66e67e47a6893aab2a0449b47528 (HEAD -> master, origin/master)
Merge: acf62b5 c460aa7
Author: LouiMinister <louiminister@gmail.com>
Date:   Tue May 17 00:53:57 2022 +0900

    Merge branch 'feature2'

commit c460aa7f8736d921dc056739634e2b30dbd57ccf (feature3, feature2)
Author: LouiMinister <louiminister@gmail.com>
Date:   Tue May 17 00:29:55 2022 +0900

    / 연산 기능 구현

commit c96dbf8f9c0e66349797b69f7af5b2e4ba900bab (origin/feature2)
Author: LouiMinister <louiminister@gmail.com>
Date:   Tue May 17 00:25:26 2022 +0900

    * 연산 기능 처음부터 다시 구현

commit acf62b530493baa50bf78d156897bc4c83856336
Author: LouiMinister <louiminister@gmail.com>
Date:   Mon May 16 23:55:01 2022 +0900

    공용 함수 util 버그 수정
```

- 브랜치 rebase 후 merge를 한 상태
- 이후 깃신은 커밋들이 의도한 순서대로 저장되어있는지 조회하고 싶음
- git log 명령어를 이용해 커밋 기록 조회
- git log: 커밋들의 히스토리를 확인



## 22. 위험한 공용함수 util2 구현

The terminal window shows the following sequence of commands:

```
fitt ~$ cd ..un/git-sample (-zsh)          #> feature1 + | ~P/g/git-sample
fitt ~$ git add source.md
fitt ~$ git commit
힌트 : 편집기가 파일을 닫기를 기다리는 중입니다... error: There was a problem with the editor 'vi'.
메시지를 -m 또는 -F 옵션으로 입력하십시오.
fitt ~$ git push origin
feature1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature1' on GitHub by visiting:
remote:     https://github.com/LouiMinister/git-sample/pull/new/feature1
remote:
To https://github.com/LouiMinister/git-sample.git
 * [new branch]      feature1 -> feature1
fitt ~$
```

The code editor on the right shows a file named 'source' with the following content:

```
1 # Source
2
3 ### 공용함수 util() 구현
4
5 ### 위험한 공용함수 util2() 구현
6
7 ### + 연산 기능 구현
8
```

그동안 깃순이는 계산기의 연산을 구현하기 위해 feature1 브랜치에서 공용함수 util2()를 구현 후 add, commit, push 하여 원격 저장소에 반영하였다.



## 23. - 연산 기능 구현

```
fitt > ~/Projects/git-sample-gitsun/git-sample feature1 git add source.md  
fitt > ~/Projects/git-sample-gitsun/git-sample feature1 + git commit [feature1 93e8bd7] - 연산 기능 구현  
1 file changed, 4 insertions(+), 1 deletion(-)  
fitt > ~/Projects/git-sample-gitsun/git-sample feature1
```

1 # Source  
2  
3     ### 공용함수 util() 구현, 수정 by 깃순  
4  
5     ### 위험한 공용함수 util2() 구현  
6  
7     ### + 연산 기능 구현  
8  
9     /\* 필요 없는 주석\*/  
10    ### - 연산 기능 구현

그리고 깃순이는 util함수 수정, - 연산 기능 구현, 필요 없는 주석을 추가하고 git add, commit 을 통해 로컬 저장소에 코드를 반영하였다.



## 24. commit --amend를 이용한 커밋

```
fitt > ~/Projects/git-sample-gitsun/git-sample feature1 + git add source.  
md  
fitt > ~/Projects/git-sample-gitsun/git-sample feature1 + git commit --am  
end  
[feature1 eebf440] - 연산 기능 구현  
Date: Tue May 17 01:47:37 2022 +0900  
1 file changed, 3 insertions(+), 1 deletion(-)  
fitt > ~/Projects/git-sample-gitsun/git-sample feature1
```

```
1 # Source  
2  
3     ### 공용함수 util() 구현, 수정 by 깃순  
4  
5     ### 위험한 공용함수 util2() 구현  
6  
7     ### + 연산 기능 구현  
8  
9     ### - 연산 기능 구현
```

- 연산 기능을 구현하고 커밋을 넣은 상태. 쓸데 없는 주석이 남은 상태.
- 쓸데 없는 주석을 지우고 저장소에 반영하고 싶지만 커밋이 하나 더 늘어나는 것을 원하지 않음.
- git add 후 git commit --amend 명령어를 실행하여 이전 커밋에 현재 수정사항 반영
- git commit --amend: 이전 커밋을 수정하여 현재 staging 상태 파일들을 반영



## 25. commit --amend 정상 작동 확인

```
commit eebf44077fce89d1d0b5b106d5aaf56961aa1026 (HEAD -> feature1)
Author: louiminstery <louiminstery@gmail.com>
Date:   Tue May 17 01:47:37 2022 +0900

    - 연산 기능 구현

commit 0312eda4fc68d3d90d9fc81d11982836cb17a2c9
Author: louiminstery <louiminstery@gmail.com>
Date:   Tue May 17 01:46:01 2022 +0900

    위험한 공용 함수 util2 구현

commit 4e4d3a317b4f9893fc2305dfe378148675dfaef6 (origin/feature1)
Author: LouiMinister <louiminister@gmail.com>
Date:   Mon May 16 19:37:19 2022 +0900

    공용 함수 util() 구현

commit 9442b3663adc6778adf320374adcf3def164ade (master)
Author: LouiMinister <louiminister@gmail.com>
Date:   Mon May 16 19:06:48 2022 +0900

+연산 기능 구현
```

그리고 깃순이는 git log를 쳐서 깃 내역을 확인한 결과 새로운 커밋이 생기지 않은것을 보고 정상적으로 작동했음을 확인할 수 있었다.



## 26. revert를 이용해 원격 저장소에 올라간 커밋 원복하기

Commits on May 17, 2022

- 연산 기능 구현  
LouiMinistry committed 7 minutes ago
- 위험한 공용함수 util2 구현  
LouiMinistry committed 14 minutes ago

Copy the full SHA

```
fitt > ~/Projects/git-sample-gitsun/git-sample [feature1] git revert 0312eda  
a4fc68d3d90d9fcfd1d11982836cb17a2c9  
자동 병합 : source.md  
[feature1 6843aa1] Revert "위험한 공용함수 util2 구현"  
 1 file changed, 2 deletions(-)  
fitt > ~/Projects/git-sample-gitsun/git-sample [feature1] git push origin f  
eature1  
오브젝트 나열하는 중 : 5, 완료 .  
오브젝트 개수 세는 중 : 100% (5/5), 완료 .  
Delta compression using up to 10 threads  
오브젝트 압축하는 중 : 100% (3/3), 완료 .  
오브젝트 쓰는 중 : 100% (3/3), 389 bytes | 389.00 KiB/s, 완료 .  
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/LouiMinister/git-sample.git  
 eebf440..6843aa1 feature1 -> feature1
```

```
1 # Source  
2  
3 ### 공용함수 util() 구현, 수정 by 깃순  
4  
5 ### + 연산 기능 구현  
6  
7 ### - 연산 기능 구현  
8
```

깃순이는 작업을 마쳤으므로 이제 master branch에 병합을 하려고 하였으나, 자신이 짠 코드중에 위험한 공용함수 util2가 문제를 발생시킬 여지가 있다는 것을 알아차렸다.

- 원복하고 싶은 커밋이 이미 원격 저장소에 올라가 있는 상태.
- 원격 저장소에 올라간 커밋을 원복하고 싶음.
- git revert 와 커밋의 해시값을 이용하여 특정 커밋을 원복하는 커밋 생성
- git revert <commit hash> 원복하고 싶은 커밋의 해시값을 인자로 전달해서 명령어를 실행하면 해당 커밋을 원복하는 커밋을 만듬. 이후 push origin 명령어로 원격 저장소에 반영



## 27. Pull Request 발행

The screenshot shows the GitHub interface for creating a pull request. At the top, a message says "feature1 had recent pushes 5 minutes ago" and a green button says "Compare & pull request". Below that, a section titled "Open a pull request" says "Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks." A dropdown menu shows "base: master" and "compare: feature1". A warning message states "Can't automatically merge. Don't worry, you can still create the pull request." On the left, there's a "Feature1" branch card with "Write" and "Preview" tabs, and a comment input field. On the right, a modal window titled "Add more commits by pushing to the feature1 branch on LouiMinister/git-sample." contains a warning: "This branch has conflicts that must be resolved. Use the web editor or the command line to resolve conflicts." It lists "Conflicting files: source.md". A "Resolve conflicts" button is at the bottom right of the modal.

이제 정말 작업을 마친 깃순이는 master에 feature1 브랜치를 병합을 하려고 하였으나, 깃순이의 코드가 미덥지 못한 깃신이 코드를 확인하고 merge를 할 수 있도록 Pull Request를 발행하라고 얘기한 것이 생각났다.

- 원격 저장소에 병합할 브랜치가 올라가 있는 상태
- 두 브랜치를 병합하기 위한 Pull Request를 생성하고 싶음
- GitHub Repository로 들어가서 Compare & pull request, Create pull Request 클릭. 병합시 충돌이 있으므로 충돌을 해결하기 위해 Resolve conflicts 클릭
- GitHub에서 Pull Request를 생성하고 충돌을 해결할 수 있음.



## 28. Conflict Resolve

Feature1 #1

Resolving conflicts between `feature1` and `master` and committing changes → `feature1`

`Commit merge` `✓ Resolve`

1 conflicting file	source.md
<code>source.md</code>	<pre>1      # Source 2 3      &lt;&lt;&lt;&lt; feature1 4      ### 공용함수 util() 구현, 수정 by 깃순 5      ### 6      ### + 연산 기능 구현 7 8      ### - 연산 기능 구현 9      ==== 10     ### 공용함수 util() 구현, 버그수정 by 깃 11 12     ### + 연산 기능 구현 13 14     ### * 연산 기능 처음부터 다시 구현 15 16     ### / 연산 기능 구현 17 18      &gt;&gt;&gt;&gt; master</pre>

A screenshot of a GitHub pull request merge conflict resolution interface. The left sidebar shows a single conflicting file named 'source.md'. The main area displays two versions of the code side-by-side. A red bracket highlights the conflict in the first half of the file, and a blue arrow points from the right side of the conflict to the second half, indicating the flow of changes. The code is annotated with comments like '# Source', 'util()', and various modification markers ('+', '\*', '/', '-').

- Pull Request를 생성했는데 두 브랜치에서 충돌이 발생한 상태
  - 충돌된 부분을 수정하여 충돌을 해결하고 싶음
  - <<< 코드1 === 코드2 >>> 형태로 표현된 코드는 코드1, 코드2가 충돌된 상태임을 의미하므로 두 코드를 비교하고 그림과 같이 수정함
  - Conflict 발생 시 위와 같이 해결할 수 있음.
- 이후 깃신은 깃순이가 만든 Pull Request를 깃허브에서 확인하였고, 코드가 병합하기 적합함을 확인한 후에 Merge pull Request 버튼을 클릭하여 Merge를 완료하였다.



## 29. Commit에 Tag 붙이기

```
fitt > ~/Projects/git-sample > ⏎ master ? ⏎ git pull origin master
https://github.com/LouiMinister/git-sample URL에서
* branch           master      -> FETCH_HEAD
업데이트 중 26ea7a1..a54cbad
Fast-forward
 source.md | 4 +++
 1 file changed, 3 insertions(+), 1 deletion(-)
fitt > ~/Projects/git-sample > ⏎ master ? ⏎ git tag v1.0
commit a54cbadd28bb12583db45351e2ceb41aea221503 (HEAD -> master, tag: v1.0, origin/master)
Merge: 26ea7a1 4f3f7a1
Author: DongJae Lee <33686751+LouiMinister@users.noreply.github.com>
Date:   Tue May 17 02:12:10 2022 +0900

Merge pull request #1 from LouiMinister/feature1

Feature1
```

- 원격 저장소에서 master 브랜치의 병합이 완료되고 로컬 저장소에는 반영이 안된 상태.
- 원격 저장소의 변경을 로컬 저장소에 반영하고 싶음.
- git pull을 이용하여 원격 저장소의 브랜치를 로컬 저장소의 브랜치에 반영함
- git pull <remote 이름> <branch 이름> 해당 원격 저장소의 브랜치를 로컬 저장소에 반영함 (업데이트)



## 30. 원격 저장소에서 Pull 받기

```
fitt > ~/Projects/git-sample > ⏎ master ? ⓘ git pull origin master
https://github.com/LouiMinister/git-sample URL에서
* branch           master      -> FETCH_HEAD
업데이트 중 26ea7a1..a54cbad
Fast-forward
 source.md | 4 +++
 1 file changed, 3 insertions(+), 1 deletion(-)
fitt > ~/Projects/git-sample > ⏎ master ? ⓘ git tag v1.0
commit a54cbadd28bb12583db45351e2ceb41aea221503 (HEAD -> master, tag: v1.0, origin/master)
Merge: 26ea7a1 4f3f7a1
Author: DongJae Lee <33686751+LouiMinister@users.noreply.github.com>
Date:   Tue May 17 02:12:10 2022 +0900

    Merge pull request #1 from LouiMinister/feature1

    Feature1
```

기본적인 기능 구현이 끝났으므로 깃신은 마지막 커밋을 기준으로 릴리즈 버전을 관리하려고 한다.

- 마지막 커밋에 따로 설정한 정보가 없는 상태
- 커밋에 버전 태그를 달아서 편리하게 확인하고 관리하고 싶음
- git tag 명령어를 이용해 가장 최근 커밋에 태그를 붙힘
- git tag 명령어를 이용하면 커밋에 태그를 붙힐 수 있음.

git log 명령어를 사용하면 commit에 tag가 정상적으로 붙음을 확인할 수 있다.



## Index

add	7pg	pull	36pg
branch	16pg	push	12pg
checkout	16pg	rebase	25pg
cherry-pick	19pg	remote	11pg
clone	15pg	reset --hard	23pg
commit --amend	31pg	revert	33pg
commit -m	8pg	stash	20pg
config	10pg	status	5pg
init	4pg	tag	37pg
log	28pg		
merge	25pg		

목차번호가 아닌 페이지번호입니다.

명령어가 사용된 페이지는 한 명령어당 여러개이나, 주요한 페이지 하나만 명시하였습니다.