

The background of the slide is a composite image. It features a view of Earth from space in the upper left, a starry night sky, and a hand holding a tablet in the lower right. The tablet screen displays various futuristic digital elements, including a globe, a bar chart, and several hexagonal icons. Overlaid on the top half of the image is a black rectangular box containing the title text.

Git/GitHub 다가가기

(Hands-on.2)

JaeHong,Shin(SOSC)
innouo@gmail.com

SungkyunOpensourceSW Center

Contents

I
Git/GitHub
Intro

II
Hands-on
(Branch)

III
Hands-on
(Tag)

IV
Hands-on
(Commit)

Contents

I
Git/GitHub
Intro

II
Hands-on
(Branch)

III
Hands-on
(Tag)

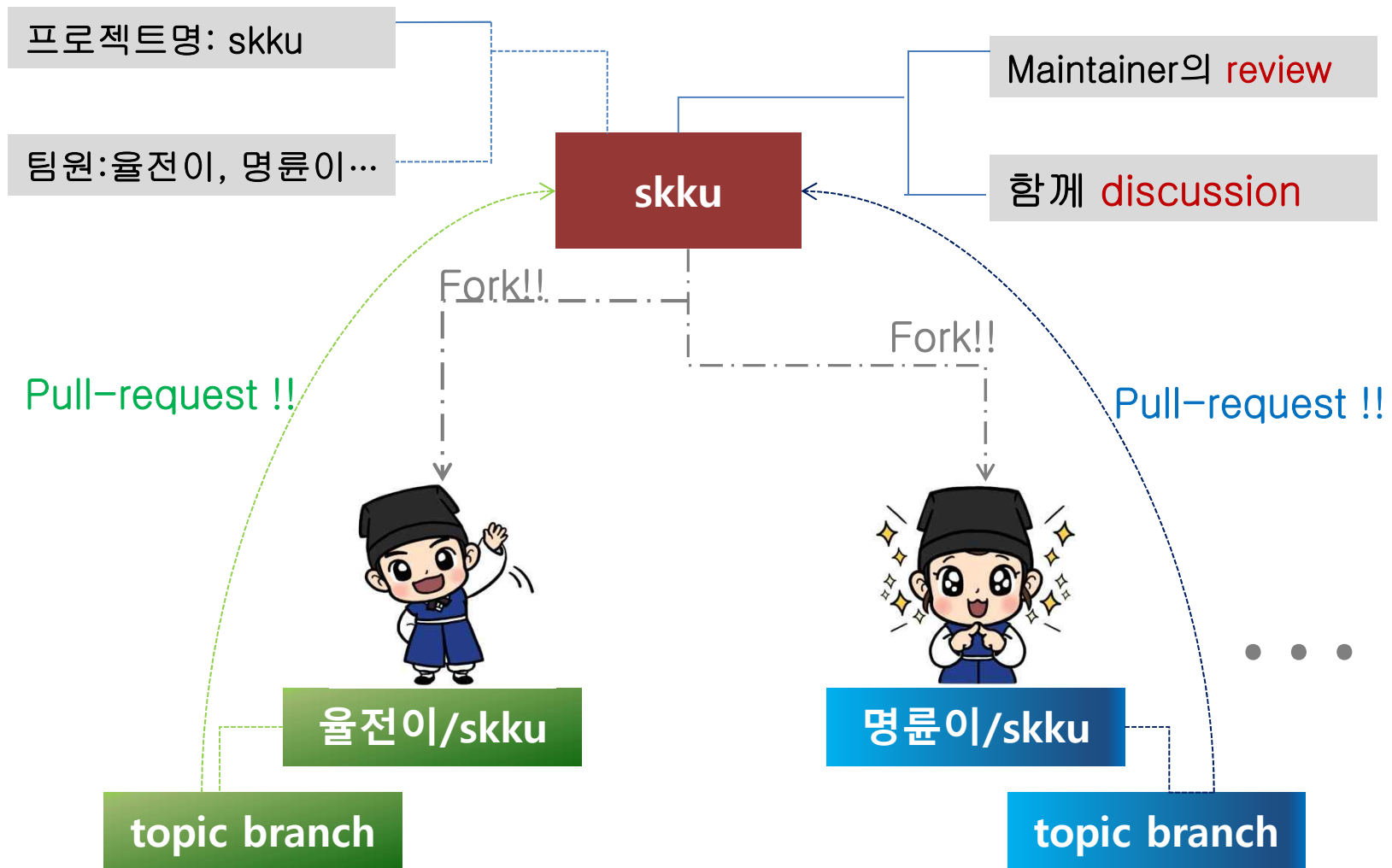
IV
Hands-on
(Commit)

Git – Overview



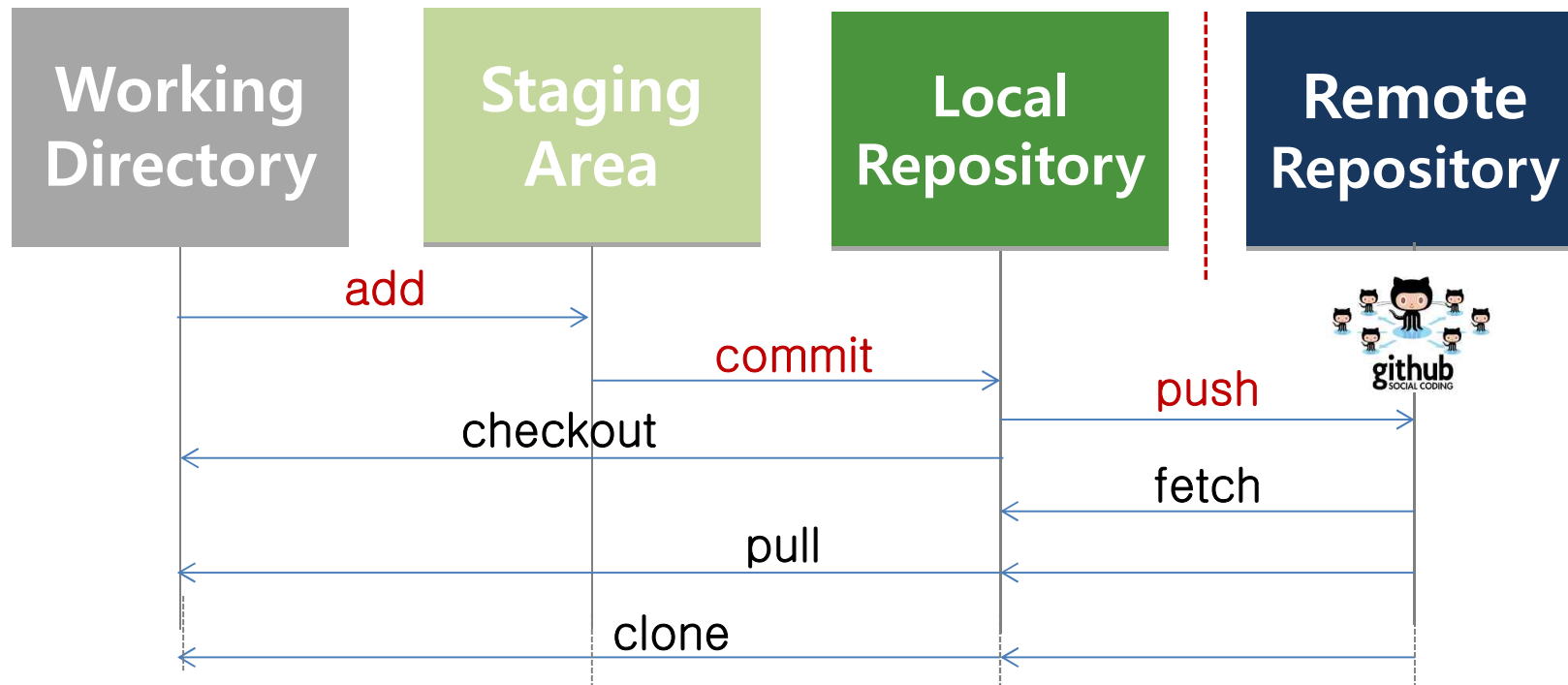
출처: <https://www.youtube.com/watch?v=o8NPllzkFhE>

우리 프로젝트와 Git 운용 전략



Git 필수 개념

– Git의 세 가지 상태



Git을 통한 작업 순서

- 워킹 디렉토리에서 파일을 수정
- 워킹 디렉토리에서 변경된 파일을 **스테이징 영역에 추가**(커밋할 스냅샷 생성)
- 스테이징 영역의 파일을 커밋하여 **Git 디렉토리에 영구적으로 저장**

Contents

I
Git/GitHub
Intro

II
Hands-on
(Branch)

III
Hands-on
(Tag)

IV
Hands-on
(Commit)

Branch



출처:<https://www.youtube.com/watch?v=H5GJfcp3p4Q>

Step-실습 준비

- Step 단계단계 해결 할때마다 "본인이름" 작성후 그래프에 색을 채우자

https://docs.google.com/spreadsheets/d/1OOIqbqN3oMITMxo3Ue2Lg3oF5BmaPEZRxaZ0ifsZ_mY/edit?usp=sharing

Hands-on2.(2017-04-11)												
파일 수정 보기 삽입 서식 데이터 도구 부가기능 도움말												
보기 전용												
	A	B	C	D	E	F	G	H	I	J	K	L
1	Git/GitHub - Hands-on.2 (2017-04-11)											
2			Step									
3	no	이름	1	2	3	4	5	6	7	8	9	10
4	ex	SKKU										
5	ex	SOSC										
6	1											
7	2											
8	3											
9	4											
10	5											

Step1—브랜치 실습 사전준비

- branch-test 폴더 만들고 myfile.txt 라는 파일 만들자

(1) 해당 폴더에 들어가서 git 초기화 하기

```
$ git cd branch-test; git init
```

- myfile.txt 열어서 (아래 간단한 글(만) 작성 해보자)

— 쉽게 배우는 Git 명령어

(2) 해당 파일 첫 커밋하기 (역사 한 단위 만들기)

```
$ git add myfile.txt; git commit -m "first commit"
```



Step2-브랜치 만들기

- **issue1** 이라는 이름으로 새로운 브랜치를 만들자
브랜치는 branch 란 명령어로 만들 수 있다.

(1) 새로운 브랜치 만들기

```
$ git branch issue1
```

- 옵션을 지정하지 않고 branch 명령어 실행하면 브랜치 목록 전체 확인 할 수 있다. 앞 부분에 *이 붙어있는 것이 현재 선택된 브랜치다.

(2) 브랜치 확인하기 (현재는 master)

```
$ git branch
```

```
issue1  
*master
```



Step2-브랜치 전환하기

- 앞에서 만든 **issue1** 이라는 이름의 브랜치를 사용하여 어떤 작업을 수행하려면 이 브랜치를 사용 하겠다고 명시적으로 지정해야함.
- 이 때 사용하는 명령어가 checkout이다. 체크아웃(checkout)이란 내가 사용할 브랜치를 지정하는 것을 의미한다.

```
$ git checkout issue1
```

* issue1
master



Step2-브랜치 전환하기

- checkout 명령에 -b 옵션을 넣으면 브랜치 작성과 체크아웃 한꺼번에 실행할 수 있다.

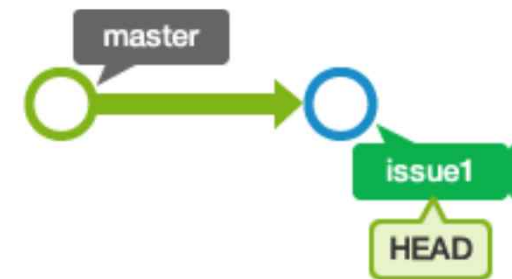
`$ git checkout -b <branch>` *Hands-on.1에서 다뤄본 내용

- `issue1` 브랜치를 체크아웃한 상태에서 커밋을 수행하면 `issue1` 브랜치에 그 이력이 기록된다.

- myfile.txt에 아래와 같은 문장을 추가한 후 커밋
- 쉽게 배우는 Git 명령어

add: 변경 사항을 만들어 인덱스 등록하기

```
$ git add myfile.txt; git commit -m "add message"
```



Step2-브랜치 병합하기(1)

- 이번에는 **issue1**의 브랜치 변경사항을 **master** 브랜치에 병합해보자.
- 브랜치 병합은 merge 명령어로 실행한다.
- 이 명령어에 병합할 커밋 이름을 넣어 실행하면 지정한 커밋 내용이 'HEAD'가 가리키고 있는 브랜치에 넣어진다.
'HEAD'는 현재 사용중인 브랜치에 위치함.
- master 브랜치에 **issue1**을 넣기 위해서는 우선 **master** 브랜치에 'HEAD'가 위치하게 만들어야함. 이 때 checkout 명령어 사용하여 현재 사용중인 브랜치를 **master**로 전환하자.

```
$ git checkout master
```

*병합 전 myfile.txt 파일을 열어 내용을 확인해보자.

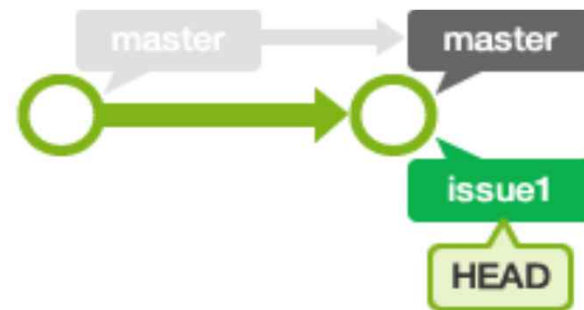
Step2-브랜치 병합하기(2)

- 이제 이번 실습에서의 파일 편집은 **issue1** 브랜치에서 실행했기 때문에 **master** 브랜치로 브랜치를 전환한 지금 myfile.txt 파일을 열었을때 그 내용이 **변경 되어 있지 않아야 한다.**

```
$ git merge issue1
```

- 이제 **master** 브랜치가 가리키는 커밋이 **issue1**과 같은 위치로 이동했다. 이런 방식의 병합을 '**fast-forward 병합**' 이라고 한다.

```
Updating 4b814a2..9b00c09
Fast-forward
 myfile.txt | 1 +
 1 file changed, 1 insertion(+)
```



Step2-브랜치 병합하기(3)

- myfile.txt 파일 열어 내용 확인해보자.
 - 쉽게 배우는 Git 명령어
add: 변경 사항을 만들어 인덱스 등록하기
- “add: 변경 사항을 만들어 인덱스 등록하기”
내용이 **추가 되어 있는 것**을 확인 할 수 있다.

Step2-브랜치 삭제하기

- `issue1` 브랜치의 내용이 모두 `master`에 통합되어 더 이상 `issue1` 브랜치는 필요가 없다.
- 브랜치 삭제하려면 `branch` 명령에 `-d` 옵션 지정하여 실행한다.

```
$ git branch -d <branchname>
```

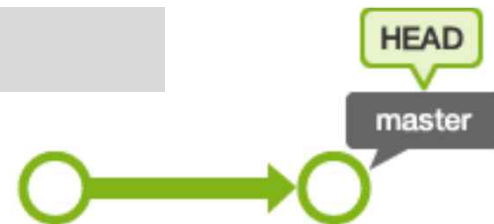
- `issue1` 브랜치를 삭제한다.

```
$ git branch -d issue1
```

- 제대로 삭제되었는지 확인해보자.

```
$ git branch
```

```
*master
```



Step3-동시에 여러 작업하기(1)

- 이번에는 2개의 브랜치 생성하여 동시에 여러 작업을 처리하는 상황을 만들어보자.

- **issue2** 와 **issue3** 브랜치 만들자

```
$ git branch issue2
```

```
$ git branch issue3
```

- **issue2** 브랜치로 전환해보자

```
$ git checkout issue2
```

```
$ git branch
```

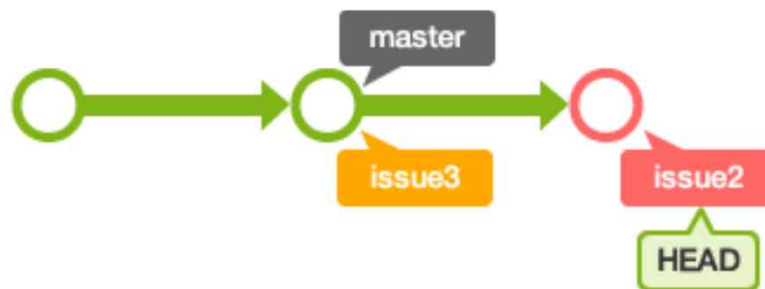
```
*issue2  
issue3  
master
```



Step3-동시에 여러 작업하기(2)

- **issue2** 브랜치의 myfile.txt에 아래와 같이 commit에 대한 설명을 추가
 - 쉽게 배우는 Git 명령어
 - add: 변경 사항을 만들어 인덱스 등록하기
 - commit: 인덱스 상태를 기록하기
- 문장을 추가했으면 커밋해보자.

```
$ git add myfile.txt; git commit -m "commit explain plus"
```



Step3-동시에 여러 작업하기(3)

- 이번에는 **issue3** 브랜치로 전환하자.

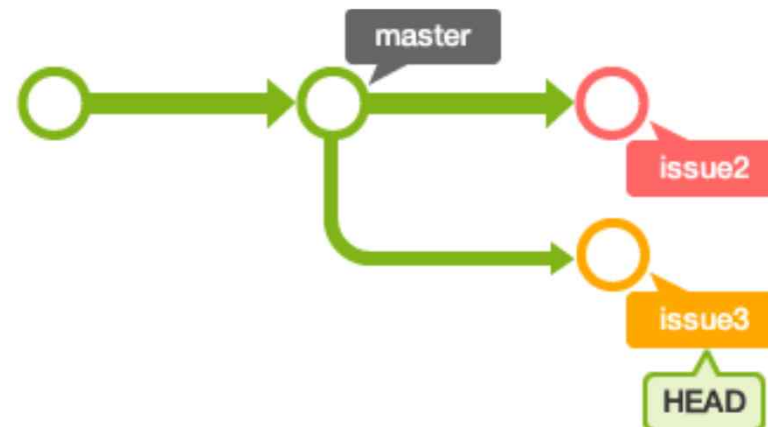
```
$ git checkout issue3
```

- myfile.txt 파일 열어 내용 확인해보자.
- commit 명령의 설명은 **issue2** 브랜치에서 추가했기 때문에 **issue3** 브랜치로 전환한 후의 myfile.txt 파일에는 add 명령의 설명밖에 없을 것이다.

Step3-동시에 여러 작업하기(4)

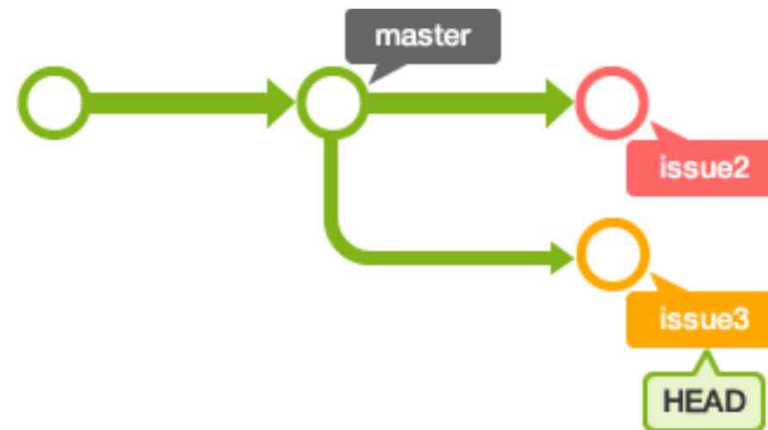
- 이번에는 pull 명령어의 설명을 추가하여 변경 사항을 커밋해 보자.
 - 쉽게 배우는 Git 명령어
 - add: 변경 사항을 만들어 인덱스 등록하기
 - pull: 원격 저장소의 내용을 가져오기
- 문장을 추가했으면 커밋해보자.

```
$ git add myfile.txt; git commit -m "pull explain plus"
```



Step3-동시에 여러 작업하기(5)

- 지금까지 **issue2** 브랜치에 commit에 대한 설명을
- **issue3** 브랜치에 pull에 대한 설명을 포함하여 커밋해 보았다.
- 이처럼 각각의 브랜치에서는 독립적으로 서로 다른 작업을 처리할 수 있다.



Step4-충돌 해결하기(1)

- **issue2**, **issue3** 각 브랜치에서 변경한 부분 모두 master 브랜치로 통합해보자.
- 먼저 master 브랜치를 체크아웃하자.

```
$ git checkout master
```

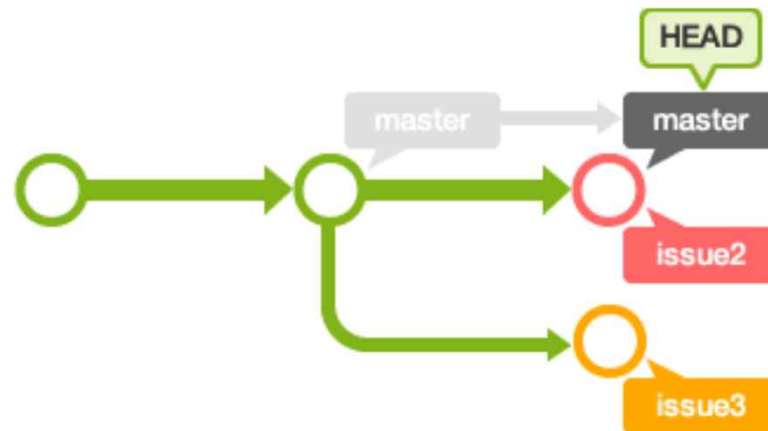
- **issue2** 브랜치를 병합한다.

```
$ git merge issue2
```

*위와 같이 하면 앞서 설명되었던 'fast-forward 병합'이 실행된다.

Step4-충돌 해결하기(2)

- 아래 그림을 보면 master 브랜치에 **issue2** 브랜치가 병합된 것을 확인할 수 있다.



- 이번에는 **issue3** 브랜치를 병합한다.

```
$ git merge issue3
```

Auto-merging myfile.txt

CONFLICT (content): Merge conflict in myfile.txt

Automatic merge failed; fix conflicts and then commit the result.

Step4-충돌 해결하기(3)

- conflict 발생에서 알 수 있듯이 자동 병합에 실패했다.
- 앞서 각각의 브랜치에서 변경한 내용이 myfile.txt의 **같은 행**에 포함되어 있기 때문이다.
- 실제로 myfile.txt의 내용을 확인해 보면 다음과 같이 변경되어 있다.

쉽게 배우는 Git 명령어

add: 변경 사항을 만들어서 인덱스에 등록해보기

<<<<<<HEAD

commit: 인덱스 상태를 기록하기

=====

pull: 원격 저장소의 내용을 가져오기

>>>>>>issue3

Step4-충돌 해결하기(4)

- 충돌이 있는 부분에 Git이 자동으로 이전 페이지와 같이 충돌 정보를 포함하여 파일 내용을 변경한다.
- 이 내용을 통해 어떤 브랜치에서 어떤 부분이 충돌 되었는지 확인할 수 있다.
- 충돌이 일어난 부분은 일일이 확인해서 수정해 줘야 한다.

- 쉽게 배우는 Git 명령어

add: 변경 사항을 만들어서 인덱스에 등록해보기

commit: 인덱스 상태를 기록하기

pull: 원격 저장소의 내용을 가져오기

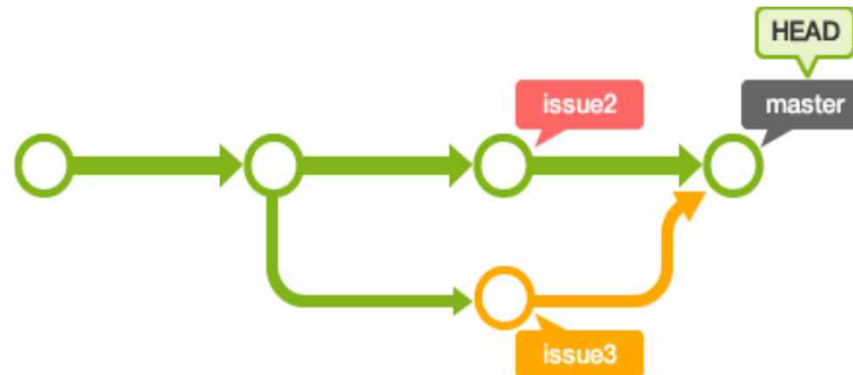
Step4-충돌 해결하기(5)

- 충돌 부분을 모두 수정했으므로 다시 커밋해보자.

```
$ git add myfile.txt
```

```
$ git commit -m "issue3 branch merge"
```

- 이 시점까지의 이력을 보면 그림과 같다. 이번 병합은 충돌 부분을 수정했기 때문에 그 변화를 기록하는 병합 커밋이 새로 생성되었다.
- 그리고 master 브랜치의 시작('HEAD')이 그 위치로 이동해 있는 것을 확인 할 수 있다. 이와 같은 방식을 'non fast-forward 병합'이라고 한다.



Step5—rebase 이해하기

Rebase와 Merge의 차이는?

둘다 두 branch의 차이점(commits)을 합치는것은 비슷하나
Rebase는 합치기전에 **되감기**(rewinding)를 하고 Merge는
안하고 합친다.

Step5—rebase는 언제 쓸까?

commit을 역사의 한단위 ‘블록’ 이라 하고 블록들의 모임을 ‘tree’라 할때

내가 쌓은 블록을 잠시 빼고

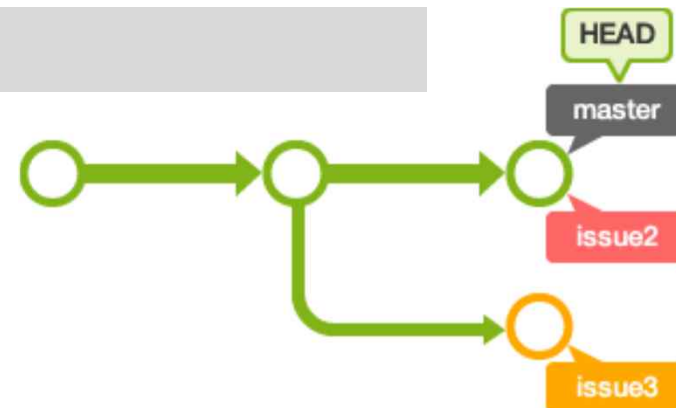
(뺀 나머지) 기준이 되는 tree를 최신 업데이트 한 후에

그 위에 다시 내 블록을 쌓아 올릴때 쓸 수 있다.

Step5—rebase로 병합하기(1)

- 앞서 두개의 브랜치를 master 브랜치로 모두 병합시켰다.
- 그로 인해 두개의 줄기로 브랜치가 분기 되었다가 다시 하나로 합쳐지는 것을 확인했다.
- **issue3** 브랜치를 병합할 때 rebase를 먼저 실행한 후 병합을 시도한다면 **그 이력을 하나의 줄기로 만들 수도 있다.**
- 이번에는 이와 같은 경우를 만들어보자.
- 앞의 실습에서 마지막으로 진행한 **병합 명령을 취소**하자.

```
$ git reset --hard HEAD~
```



Step5—rebase로 병합하기(2)

- 이제 **issue3** 브랜치로 전환하여 master 브랜치에 rebase를 실행한다.

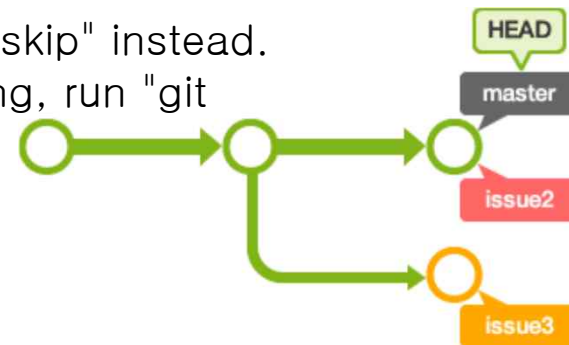
```
$ git checkout issue3
```

```
$ git rebase master
```

When you have resolved this problem, run "git rebase --continue".

If you prefer to skip this patch, run "git rebase --skip" instead.

To check out the original branch and stop rebasing, run "git rebase --abort".



Step5—rebase로 병합하기(3)

- merge 때와 마찬가지로 myfile.txt 파일 내용에 **충돌**이 있다.
- 그렇기 때문에 그 부분을 **이전 실습과 동일하게 처리**한다.
- 충돌난 파일 내용을 수정하자.
- rebase 경우 충돌 부분을 수정한 후에는 commit이 아니라 rebase 명령에 --continue 옵션을 지정하여 실행해야 한다.

```
$ git add myfile.txt
```

```
$ git rebase --continue
```

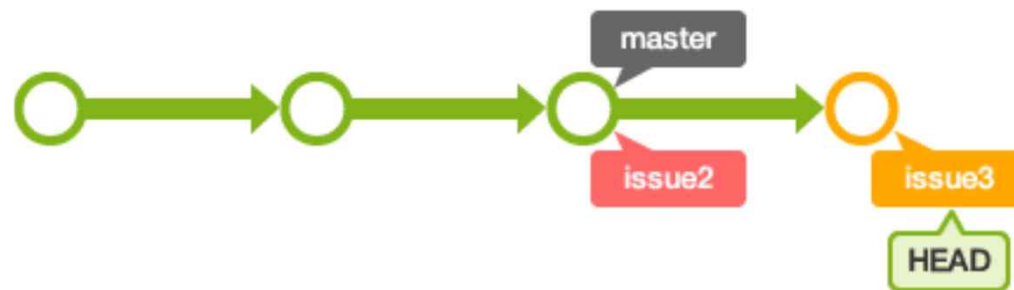
*만약 rebase 자체를 취소하려면 -abort 옵션을 실행하면 된다.

Step5—rebase로 병합하기(4)

- 앞선 실습에서 merge 명령어를 사용했을 때와 같이, rebase만 실행한 경우 아래 그림처럼 **issue3** 브랜치가 두 브랜치의 앞 쪽으로 **위치만 변경된 상태**이다.
- master브랜치는 아직 **issue3**의 변경사항이 적용되지 못한 상태임.
- master브랜치로 전환하여 **issue3** 브랜치의 변경 사항 모두 병합하자.

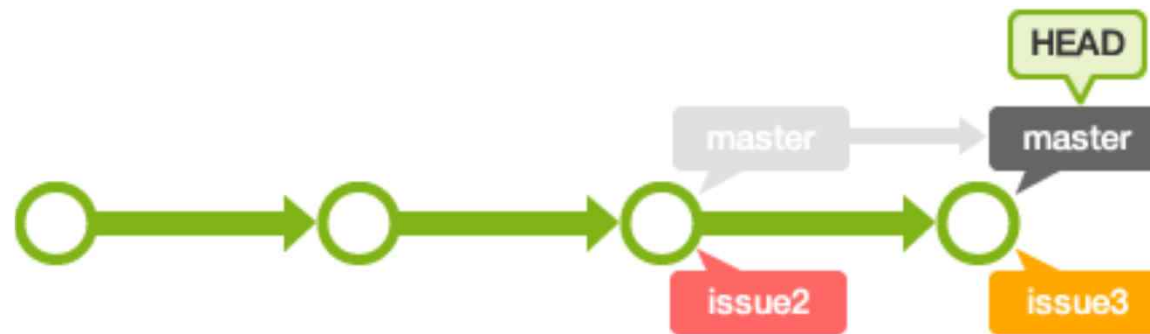
```
$ git checkout master
```

```
$ git merge issue3
```



Step5—rebase로 병합하기(5)

- myfile.txt 최종 내용은 merge 했을 경우와 동일.
- 이력은 아래 그림과 같이 달라진다.



Contents

I
Git/GitHub
Intro

II
Hands-on
(Branch)

III
Hands-on
(Tag)

IV
Hands-on
(Commit)

Tag

- 태그란 커밋을 참조하기 쉽게 **알기 쉬운 이름을 붙이는 것**을 말한다.
 - 한 번 붙인 태그는 브랜치 처럼 위치가 이동하지 않고 **고정**된다.
 - Git에서는 일반적으로 이름 정보만을 가지는 태그(Lightweight tag)
 - 보다 상세한 정보를 포함하는 주석 태그(Annotated tag)
두 가지 태그를 사용한다.
1. 일반 태그(Lightweight tag)
 - 이름만 붙일 수 있다.
 2. 주석 태그(Annotated tag)
 - 이름을 붙일 수 있다.
 - 태그에 대한 설명도 포함할 수 있다.
 - 저자 서명 삽입 가능
 - 태그를 만든 사람의 이름, 이메일과 태그를 만든 날짜 정보도 포함 가능

Tag

- 일반적으로
 - 릴리즈 브랜치 : 주석 태그 사용하여 설명이나 서명 넣은 상세 정보 포함
 - 토픽 브랜치 : 로컬에서 일시적으로 사용하므로 이름만 만들어 붙이는 태그를 사용함

- 태그 이름을 지정하여 체크아웃하거나 reset하여 과거의 특정 상태로 쉽게 되돌릴 수 있다.

Step6-사전준비

- branch-test 폴더에서 myfile2.txt라는 이름으로 파일을 만든다.
- myfile2.txt 열어서 (아래 간단한 글(만) 작성 해보자)
 - 쉽게 배우는 Git 명령어
- 작성한 다음 파일을 커밋해보자

```
$ git add myfile2.txt; git commit -m "myfile2.txt commit"
```

Step6-태그 추가하기

- 태그 추가하려면 tag 명령어를 사용한다.
- 현재 'HEAD'가 가리키고 있는 커밋에 'skku'라는 태그를 달아보자.

```
$ git tag skku
```

- 파라미터 없이 tag 실행하면 태그 목록을 볼 수 있다.

```
$ git tag
```

skku

Step6-태그 추가하기

- log 명령어에 --decorate 붙여 실행하면 태그 정보 포함한 이력 확인 가능

```
$ git log --decorate
```

```
commit e7978c94d2104e3e0e6e4a5b4a8467b1d2a2ba19(HEAD,  
tag: skku, master)  
Author: yourname yourname@yourmail.com  
Date: Tue Apr 11 19:40:27 2017 +0900
```


Step6-주석달린 태그 추가하기

- -m 옵션 지정하여 명령어를 실행할 때 바로 내용을 입력할 수도 있다.
- 현재 'HEAD'가 가리키고 있는 커밋에 edu라는 주석이 달린 태그를 달려면 다음과 같이 명령어를 실행한다.

```
$ git tag -m "difficult git" edu
```

- -n 옵션 지정하여 tag 명령어 실행하면 태그 목록과 주석 내용 확인 가능.

```
$ git tag -n
```

```
skku    myfile2.txt commit
edu     difficult git
```

Step6-주석달린 태그 추가하기

- 태그를 삭제하려면, tag 명령어에 -d 옵션을 지정하여 실행한다.

```
$ git tag -d <tagname>
```

Contents

I
Git/GitHub
Intro

II
Hands-on
(Branch)

III
Hands-on
(Tag)

IV
Hnads-on
(Commit)

Step7-이전에 작성한 커밋 수정하기(1)

- `--amend` 옵션 지정하여 커밋 수정하면, 같은 브랜치 상에서 이전에 커밋했던 내용에 새로운 내용 추가하거나 설명 수정할 수 있다.

※ 이럴 때 사용해 보세요:

- 누락된 파일 새로 추가하거나 기존의 파일 업데이트 해야 할 때
- 이전 커밋의 설명을 변경하고 싶을 때
- 이번 실습은 사전에 이력이 준비되어 있는 로컬 저장소 사용해보자.
- 실습 예제 파일 압축 풀고, `stepup-tutorial/tutorial1` 폴더로 이동하자.
- 이 저장소의 이력은 아래 그림과 같다.

```
$ cd tutorial1
```



Step7-이전에 작성한 커밋 수정하기(2)

- Log 명령어를 실행해 이력을 확인해 보자.

```
$ git log
```

```
commit 77e8c02f41f6c9aca2d8b7f8d51a2dd71a5ddee7  
Author: yourname <yourname@yourmail.com>  
Date: Mon Jul 16 23:17:56 2012 +0900
```

add의 설명을 추가

```
commit ffb2b9fc30a1649310bfe8d383e61fae39cdfea5  
Author: yourname <yourname@yourmail.com>  
Date: Mon Jul 16 23:16:14 2012 +0900
```

first commit

Step7-이전에 작성한 커밋 수정하기(3)

- Sample.txt 파일 열고 commit의 설명을 추가하자
 - 원숭이도 이해할 수 있는 Git 명령어
 - add : 변경 사항을 만들어서 인덱스에 등록해보기
 - commit: 인덱스의 상태를 기록하기
- --amend 옵션 이용하여 커밋해보자

```
$ git add sample.txt
```

```
$ git commit --amend
```

Step7-이전에 작성한 커밋 수정하기(4)

- 편집기에서 커밋한 내용이 포함되어 있을 것이다.
- 내용을 'add and commit explain'로 바꾸어 넣은 다음 저장하고 빠져 나오자.
- 커밋의 내용이 수정되었다. log 명령어에서 이력과 커밋 메시지를 확인해보자.

```
$ git log
```

```
commit c834a923bd9d027e8e08d3cca3665d6040ca49a7
Author: yourname <yourname@yourmail.com>
Date: Mon Jul 16 23:17:56 2012 +0900
```

```
    add and commit explain
```

```
commit ffb2b9fc30a1649310bfe8d383e61fae39cdfea5
Author: yourname <yourname@yourmail.com>
Date: Mon Jul 16 23:16:14 2012 +0900
```

```
    first commit
```

Vim 에디터

i	커서위치부터 입력
x	한글자 삭제
ESC	글자 입력 끝낼 때
:wq	저장하고 나가기

Step8-커밋 버리고 특정 버전으로 되돌아가기(1)

- reset 이용하여 master 브랜치 앞 두개의 커밋을 삭제해보자.
- stepup-tutorial/tutorial3 폴더로 이동한다.
- 이 저장소의 이력은 다음 그림과 같다.

```
$ cd ..
```

```
$ cd tutorial3
```

- log 명령어 실행하여 이력을 확인한 후 sample.txt 열어 내용을 확인.

```
$ git log
```



Step8-커밋 버리고 특정 버전으로 되돌아가기(2)

- reset 이용하여 커밋을 삭제하자.



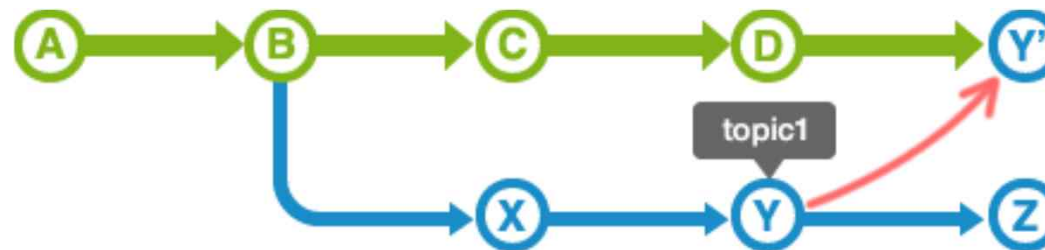
```
$ git reset --hard HEAD~~
```

- sampel.txt 열어 commit와 pull의 설명이 지워졌는지 확인후, log 명령어를 실행하여 이력을 확인하자.
- reset 전의 커밋은 'ORIG_HEAD'라는 이름으로 참조 가능하다.

```
$ git reset --hard ORIG_HEAD
```

Step9- 다른 브랜치로부터 특정 커밋 가져와서 내 브랜치에 넣기(1)

- Cherry-pick 이용하여 다른 브랜치에서 지정한 커밋을 복사하여 현재 브랜치로 가져올 수 있다.



◆ 이럴 때 사용해 보세요:

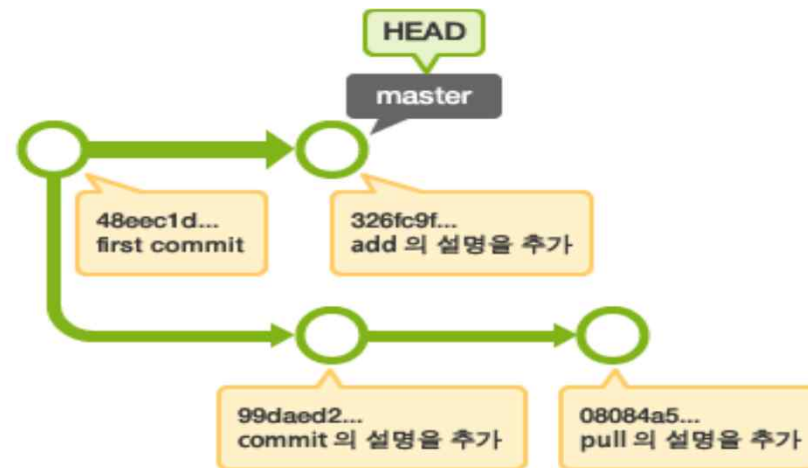
- 특정 브랜치에 잘못 추가한 커밋을 올바른 브랜치로 옮기려고 할 때
- 다른 브랜치의 커밋을 현재 브랜치에도 추가하고 싶을 때

Step9- 다른 브랜치로부터 특정 커밋 가져와서 내 브랜치에 넣기(2)

- 다운로드 한 stepup-tutorial/tutorial4 폴더로 이동한다.
- 이 저장소의 이력은 다음 그림과 같다.

```
$ cd ..
```

```
$ cd tutorial4
```



- 다른 브랜치에서 수행한 ‘commit의 설명 추가’ 커밋 내용을 master 브랜치로 가져와 보도록 하자.

Step9—다른 브랜치로부터 특정 커밋 가져와서 내 브랜치에 넣기(3)

- 문서 내의 커밋 “99daed2..”와 다운로드 한 저장소 내의 커밋은 다를 가능성이 있다.
- 다운로드한 저장소내 issue1브랜치에서 git log 실행하여 커밋을 확인하고 해당 **해시 값(SHA-1)** 복사하자.

```
$ git log
```

The screenshot shows a terminal window with the following content:

```
MINGW32/c/Users/USER/stepup-tutorial/stepup-tutorial/tutorial4
$ cd tutorial4
USER@ MINGW32 ~/stepup-tutorial/stepup-tutorial/tut
orial4 (issue1)
$ git log
commit 14ca078c3e5ca200c43f940dfb6705b2b3585e07
Author: yourname <yourname@yourmail.com>
Date: Mon Dec 15 15:13:54 2014 +0900

pull의 설명을 추가

commit e4ecbb81257ba33c75b918d9198228bd47b94df
Author: yourname <yourname@yourmail.com>
Date: Mon Dec 15 15:13:38 2014 +0900

commit의 설명 추가

commit 3492731cbf9f9389242a2fa4964461c5097a0b1
Author: yourname <yourname@yourmail.com>
Date: Mon Jul 16 23:16:14 2012 +0900

first commit
USER@ MINGW32 ~/stepup-tutorial/stepup-tutorial/tut
orial4 (issue1)
$
```

A context menu is overlaid on the terminal, showing options: Open, Copy (Ctrl+Ins), Paste (Shift+Ins), Select All, Search (Alt+F3), Reset (Alt+F8), Default Size (Alt+F10), Full Screen (Alt+F11), Flip Screen (Alt+F12), and Options...

Step9—다른 브랜치로부터 특정 커밋 가져와서 내 브랜치에 넣기(4)

- master 브랜치로 이동한 후, cherry-pick 사용하여 'commit의 설명 추가' 한 복사한 커밋의 해시 값(SHA-1) master에 추가해보자.

```
$ git checkout master
```

```
$ git cherry-pick e4ecbb81..
```

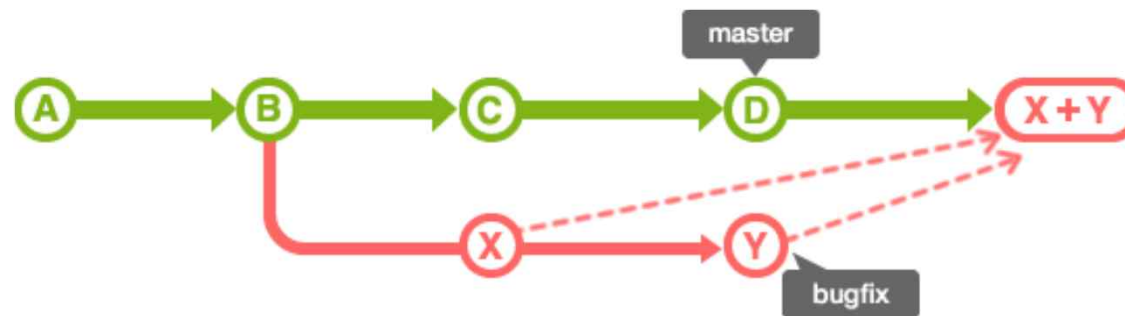
```
USER@ MINGW32 ~/stepup-tutorial/stepup-tutorial/tutorial4 (master)
$ git cherry-pick e4ecbb81257ba33c75b918d9198228bd47b94dfb
error: could not apply e4ecbb8... commit의 설명 추가
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'
```

- 충돌이 발생했다. Sample.txt 열고 충돌 부분을 수정한 후 커밋하자.

```
$ git add sample.txt; git commit
```

Step10—브랜치 상의 커밋 하나로 병합하기(1)

- 병합(merge)할 때 사용할 수 있는 특별한 옵션인 `-squash` 가 있다.
- 이 옵션 지정하여 브랜치 병합하면 해당 브랜치의 커밋 전체를 통합한 커밋이 추가 된다.



◆ 이럴 때 사용해 보세요:

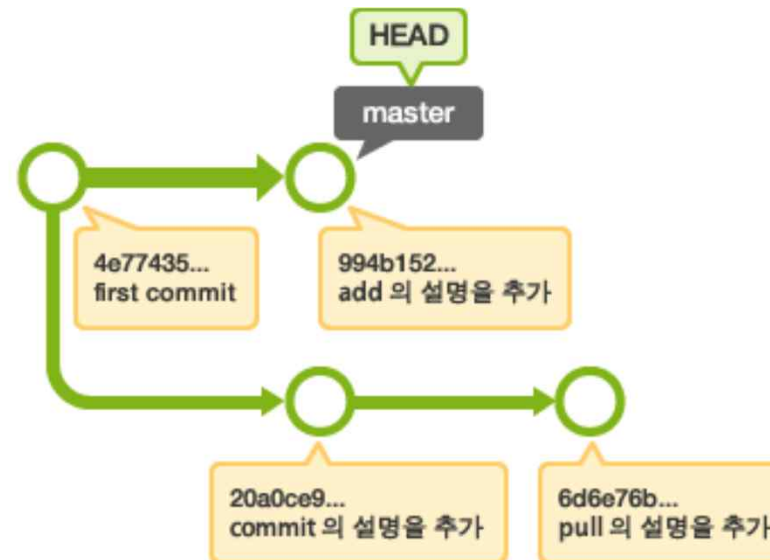
- 토픽 브랜치 안의 커밋을 **한꺼번에 모아서 통합 브랜치에 병합하고자 할 때** 사용한다.

Step10—브랜치 상의 커밋 하나로 병합하기(2)

- 다운로드 한 stepup-tutorial/tutorial7 폴더로 이동한다.
- 이 저장소의 이력은 다음 그림과 같다.

```
$ cd ..
```

```
$ cd tutorial7
```



Step10—브랜치 상의 커밋 하나로 병합하기(3)

- master 브랜치로 이동한 후 --squash 옵션 지정하여 merge 실행

```
$ git checkout master
```

```
$ git merge --squash issue1
```

- 충돌이 발생했으므로 sample.txt 열어 충돌 부분 수정한 후 커밋한다.

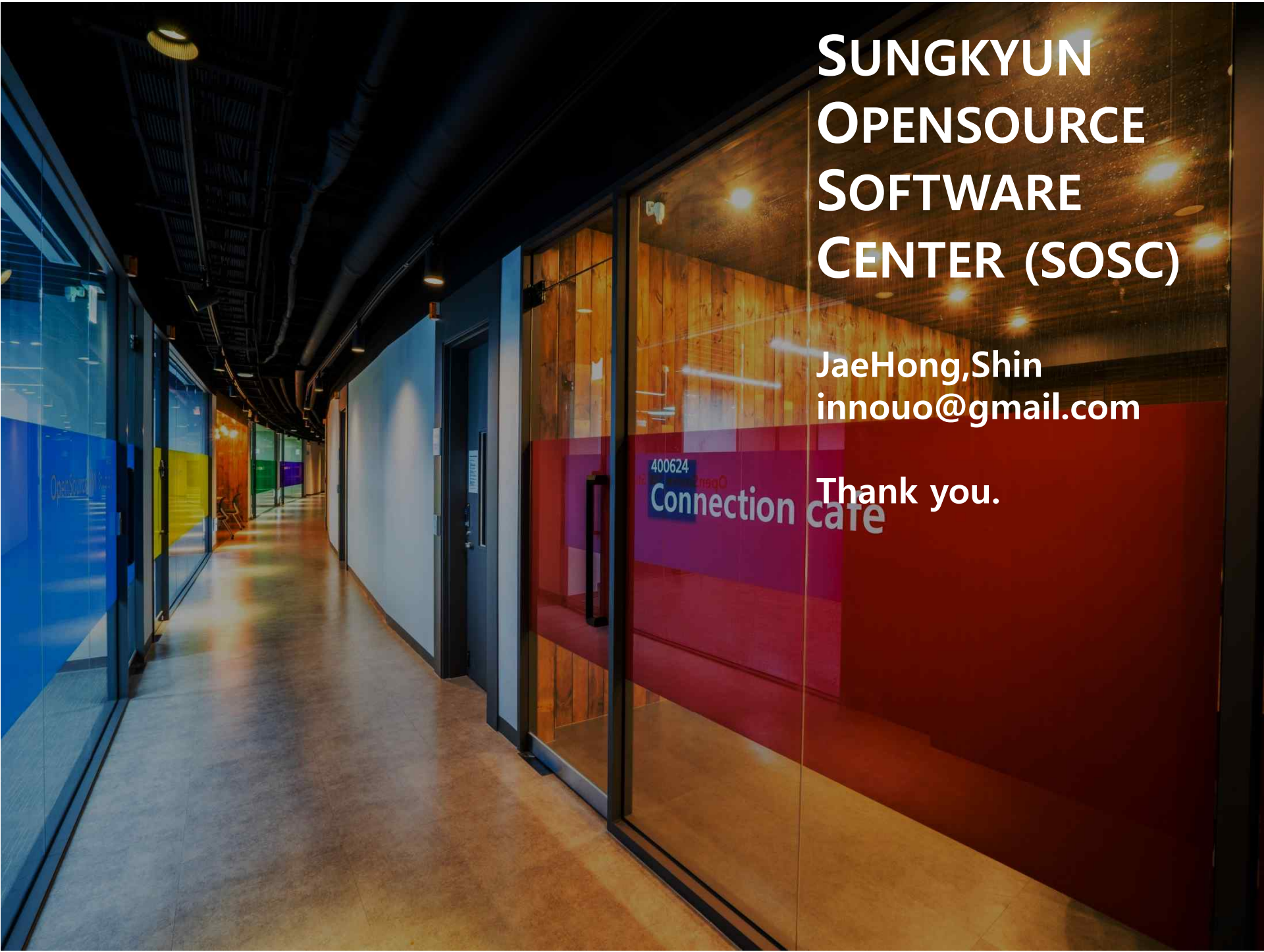
```
$ git add sample.txt
```

```
$ git commit
```


참여와 공유





A photograph of a modern office hallway. The hallway has a polished floor and a dark ceiling with exposed pipes and lights. On the left, there are glass-walled rooms with colorful partitions in blue, yellow, and green. On the right, there are glass-walled rooms with wood-paneled walls and red partitions. One of the red partitions has a sign that says "400624 Connection cafe".

SUNGKYUN OPENSOURCE SOFTWARE CENTER (SOSC)

JaeHong,Shin
innouo@gmail.com

Thank you.