

# Git\_lec\_2

2016025614 원준호



# Git\_lec\_2

2016025614 원준호

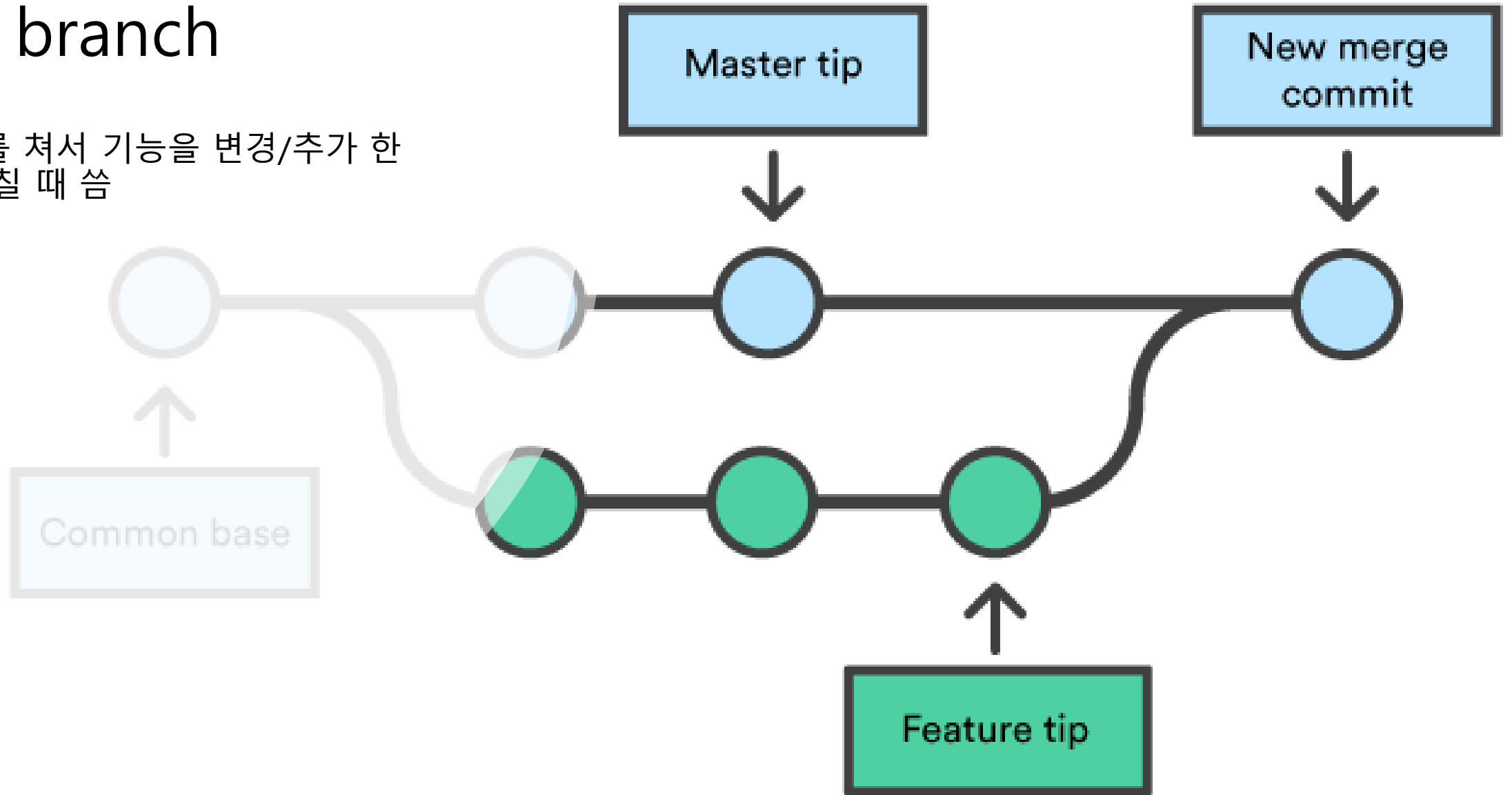


Branch란?



# git branch

- 가지를 쳐서 기능을 변경/추가 한 후 합칠 때 씬



branch 실습

# git branch -v git remote -v

- 브랜치의 상태
- 원격 저장소와의 상태
- (여기선 내 작업공간이 포크 뜬 저장소인지 확인용)

C:\Users\Joonho Wohn\Documents\Dev\GitPractice 디렉터리

```
2018-03-27 오전 10:48 <DIR> .
2018-03-27 오전 10:48 <DIR> ..
2018-03-27 오전 10:48 <DIR> GitLectureFile
2018-03-27 오전 10:50 <DIR> PushByNameHere
2018-03-27 오전 10:48      46 README.md
                        1개 파일      46 바이트
                        4개 디렉터리 217,197,383,680 바이트 남음
```

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git branch -v
* master 0ba11c5 junoflow
```

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git remote -v
origin  https://github.com/doomsheart/GitPractice.git (fetch)
origin  https://github.com/doomsheart/GitPractice.git (push)
```

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>
```

# 시나리오

- i) 자신이 원하는 기능을 추가하려 함. (ex. `printf("love uWn");`);)
- ii) 브랜치를 팜. (ex. `feat_rmtc`)
- iii) 그 브랜치에서 작업
- iiii) 완료 후 `master`에 병합

# 1. 브랜치 파기

- git branch 내가원하는브랜치이름
- git checkout 바꿀브랜치

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git branch feat_rmtc

C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git remote -v
origin  https://github.com/doomsheart/GitPractice.git (fetch)
origin  https://github.com/doomsheart/GitPractice.git (push)

C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git checkout feat_rmtc
Switched to branch 'feat_rmtc'

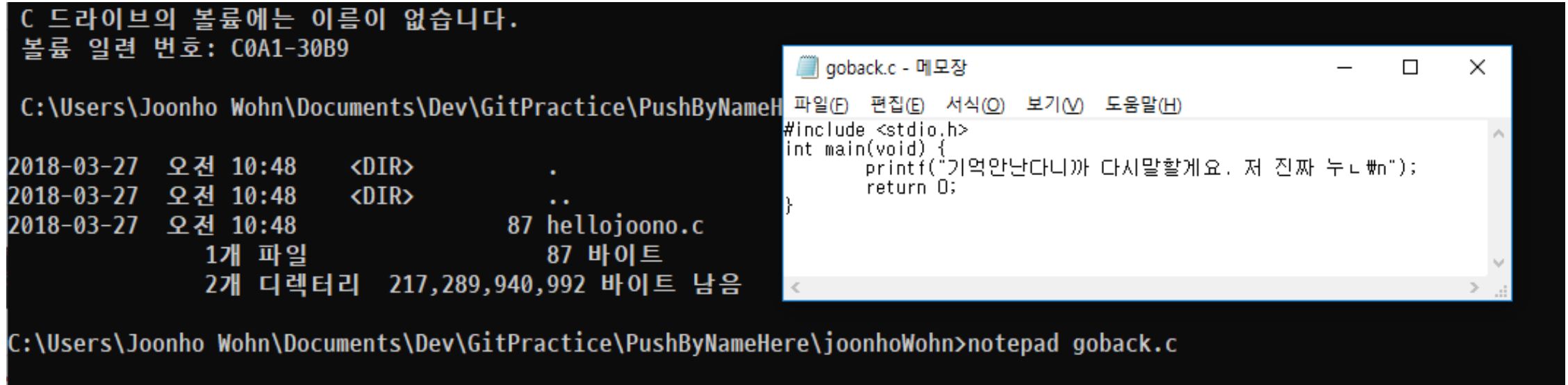
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git branch -v
* feat_rmtc 0ba11c5 junoflow
master      0ba11c5 junoflow

C:\Users\Joonho Wohn\Documents\Dev\GitPractice>
```



## 2. 수정하기

- Ex) 제 경우 자신의 이름 폴더에 goback.c 생성



The screenshot shows a Windows command prompt window with the following text:

```
C 드라이브의 볼륨에는 이름이 없습니다.  
볼륨 일련 번호: C0A1-30B9  
  
C:\Users\Joonho Wohn\Documents\Dev\GitPractice\PushByNameHere>ls  
2018-03-27 오전 10:48 <DIR> .  
2018-03-27 오전 10:48 <DIR> ..  
2018-03-27 오전 10:48      87 hellojoono.c  
                1개 파일      87 바이트  
                2개 디렉터리 217,289,940,992 바이트 남음
```

Below the command prompt, the command `notepad goback.c` is entered. To the right, a Notepad window titled "goback.c - 메모장" is open, displaying the following C code:

```
#include <stdio.h>  
int main(void) {  
    printf("기억안난다니까 다시말할게요. 저 진짜 누나\n");  
    return 0;  
}
```

# 여기서 좀 신기한거 (git의 유용함)

- 지금 우리는 새로 판 branch(제 경우 feat\_rmtc)에서 작업후
- **git add, git commit** 함
- 그 브랜치에서 파일 리스트를 보면 다음과 같음

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice\PushByNameHere\joonhoWohn>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: C0A1-30B9

C:\Users\Joonho Wohn\Documents\Dev\GitPractice\PushByNameHere\joonhoWohn 디렉터리

2018-03-28 오후 02:44 <DIR>          .
2018-03-28 오후 02:44 <DIR>          ..
2018-03-28 오후 02:48             108 goback.c
2018-03-27 오전 10:48             87 hellojoono.c
                2개 파일              195 바이트
                2개 디렉터리 217,291,956,224 바이트 남음
```

- 그럼 마스터(원래 있던데)는 어떻게 되어있을까?

# 마스터(원래 기본적으로 있던 곳)으로 돌아가봅시다

- git checkout master
- 그다음 파일 리스트를 찍어보면

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice\PushByNameHere\joonhoWohn>git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

C:\Users\Joonho Wohn\Documents\Dev\GitPractice\PushByNameHere\joonhoWohn>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: C0A1-30B9

C:\Users\Joonho Wohn\Documents\Dev\GitPractice\PushByNameHere\joonhoWohn 디렉터리

2018-03-29 오전 10:31 <DIR> .
2018-03-29 오전 10:31 <DIR> ..
2018-03-27 오전 10:48      87 hellojono.c
                1개 파일      87 바이트
                2개 디렉터리 215,759,753,216 바이트 남음
```

# 브랜치 푸시

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git push
fatal: The current branch feat_rmtc has no upstream branch.
To push the current branch and set the remote as upstream, use
```

```
git push --set-upstream origin feat_rmtc
```

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git push --set-upstream origin feat_rmtc
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 618 bytes | 309.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/doomsheart/GitPractice.git
 * [new branch]      feat_rmtc -> feat_rmtc
Branch 'feat_rmtc' set up to track remote branch 'feat_rmtc' from 'origin'.
```

# 브랜치 푸시후 깃허브를 들어가보면...

The screenshot shows the GitHub interface for the repository 'doomsheart / GitPractice', which is a fork of 'HyOsori/GitPractice'. The repository has 1 commit, 2 branches, 0 releases, and 1 contributor. The 'feat\_rmtc' branch is highlighted in the 'Your recently pushed branches' section. A 'Switch branches/tags' dialog box is open, showing a list of branches: 'feat\_rmtc' (selected) and 'master' (checked). The 'master' branch is the current active branch. The repository description is 'repository for practicing git'. The 'README.md' file is visible at the bottom of the page.

doomsheart / GitPractice  
forked from HyOsori/GitPractice

Unwatch 1 Star 0 Fork 5

Code Pull requests 0 Projects 0 Wiki Insights Settings

repository for practicing git Edit

Add topics

4 commits 2 branches 0 releases 1 contributor

Your recently pushed branches:

feat\_rmtc (less than a minute ago) Compare & pull request

Branch: master New pull request Create new file Upload files Find file Clone or download

Switch branches/tags

Find or create a branch...

Branches Tags

feat\_rmtc

✓ master

add lecture files 7 days ago

junoflow 2 days ago

Initial commit 7 days ago

README.md

## GitPractice

repository for practicing git



Fork하며 발생하는 문제

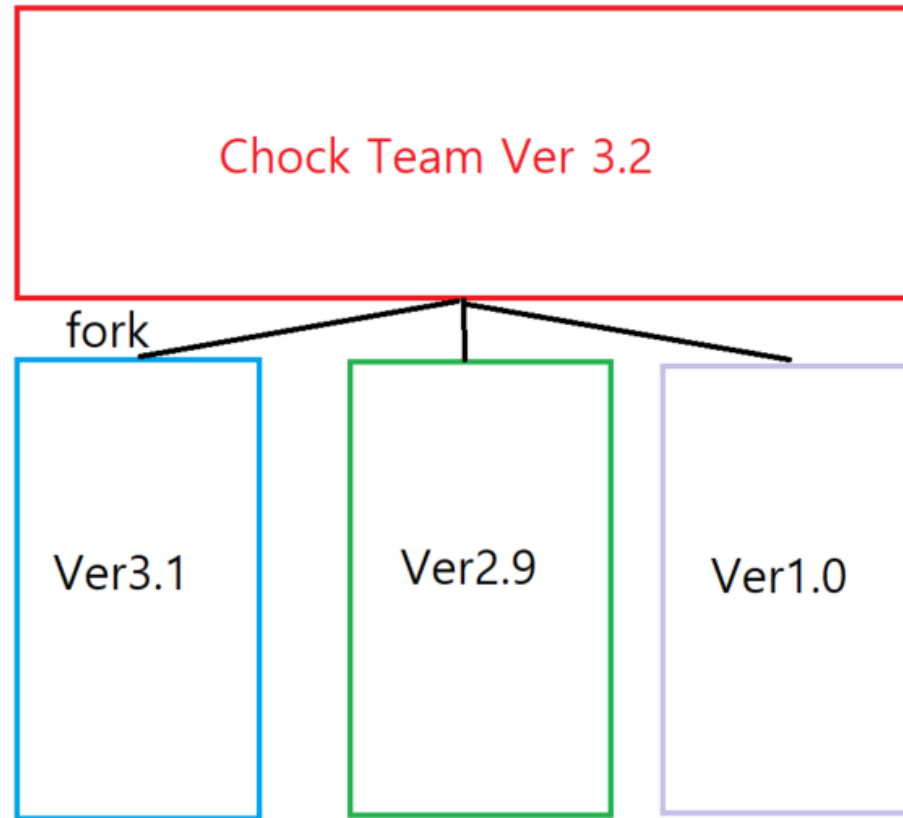
# 포크 동기화 (Syncing a fork)

- 시나리오

- i) 유비 관우 장비는 한날 한시에 프로젝트 시작
- ii) 유비 장비는 열심히 커밋, 관우는 술이 식기전에 돌아온다 했는데 안 돌아옴
- iii) 돌아와 보니 이미 유비 관우가 기능 많이 추가한 상태, 자신의 포크든 레포는 너무 옛날 버전
- iiiii) 관우의 선택은?
  - i) 포크 뜯거 삭제 후 다시 포크 아니면 그냥 삭제후 탈주
  - ii) 포크 동기화!



# 도식화





# 깃 피셜 링크 방법

## Syncing a fork

### The Setup

Before you can sync, you need to add a remote that points to the upstream repository. You may have done this when you originally forked.

*Tip: Syncing your fork only updates your local copy of the repository; it does not update your repository on GitHub.*

```
$ git remote -v
# List the current remotes
origin  https://github.com/user/repo.git (fetch)
origin  https://github.com/user/repo.git (push)

$ git remote add upstream https://github.com/otheruser/repo.git
# Set a new remote

$ git remote -v
# Verify new remote
origin  https://github.com/user/repo.git (fetch)
origin  https://github.com/user/repo.git (push)
upstream https://github.com/otheruser/repo.git (fetch)
upstream https://github.com/otheruser/repo.git (push)
```

## Syncing

There are two steps required to sync your repository with the upstream: first you must fetch from the remote, then you must merge the desired branch into your local branch.

### Fetching

Fetching from the remote repository will bring in its branches and their respective commits. These are stored in your local repository under special branches.

```
$ git fetch upstream
# Grab the upstream remote's branches
remote: Counting objects: 75, done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 62 (delta 27), reused 44 (delta 9)
Unpacking objects: 100% (62/62), done.
From https://github.com/otheruser/repo
* [new branch]      master    -> upstream/master
```

We now have the upstream's master branch stored in a local branch, upstream/master

```
$ git branch -va
# List all local and remote-tracking branches
* master                a422352 My local commit
remotes/origin/HEAD      -> origin/master
remotes/origin/master    a422352 My local commit
remotes/upstream/master  5fdff0f Some upstream commit
```

## Merging

Now that we have fetched the upstream repository, we want to merge its changes into our local branch. This will bring that branch into sync with the upstream, without losing our local changes.

```
$ git checkout master
# Check out our local master branch
Switched to branch 'master'

$ git merge upstream/master
# Merge upstream's master into our own
Updating a422352..5fdff0f
Fast-forward
 README                      |    9 -
 README.md                   |    7 +++
 2 files changed, 7 insertions(+), 9 deletions(-)
 delete mode 100644 README
 create mode 100644 README.md
```

If your local branch didn't have any unique commits, git will instead perform a "fast-forward":

```
$ git merge upstream/master
Updating 34e91da..16c56ad
Fast-forward
 README.md                   |    5 +++-
 1 file changed, 3 insertions(+), 2 deletions(-)
```

*Tip: If you want to update your repository on GitHub, follow the instructions [here](#)*

Fetch? Upstream? Merge?

# Fetch

- Pull 과는 조금 다른 개념
- 원격저장소의 최신버전을 **브랜치**에 당겨오는 명령어
- Pull  $\sim$  fetch + (마스터에) merge



# Merge

---

- 브랜치를 모으는 행위
- 특정 브랜치에서 작업후 merge할때 master 브랜치의 변경사항이 없다?  
->가꿀. 마스터를 주욱 끝까지 빨리감기 (fast forward) 하면됨
- 근데 그렇지 않다면?



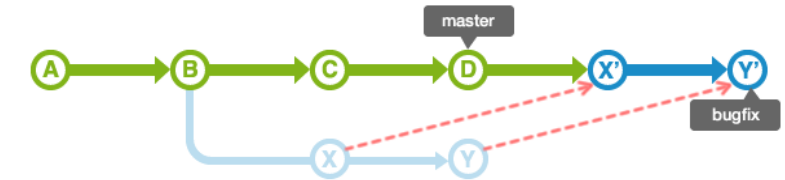
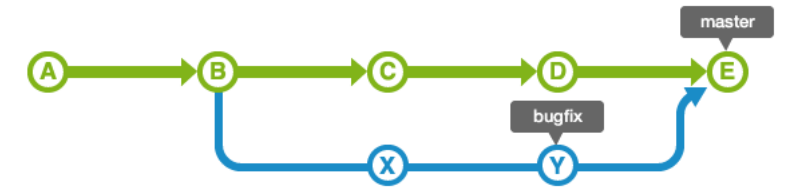
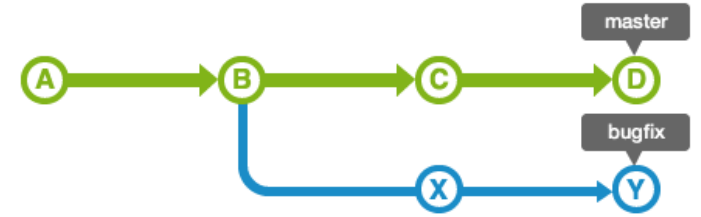
# 우선 출처를 밝히는 바랍니다..

- [https://backlog.com/git-tutorial/kr/stepup/stepup1\\_4.html](https://backlog.com/git-tutorial/kr/stepup/stepup1_4.html)
- 여기 자료 좀 많이 가져왔어요...

- 오른쪽과 같은 브랜치 진행중 마스터의 변경이 있을때!

- merge 를 하게되면 오른쪽과 같이 두 이력이 남음

- **rebase**라는걸 하게 되면 한 흐름으로 표현이 됨





- 넘모 좋아서 그대로 가져왔어요....

### Note

merge 와 rebase 는 통합 브랜치에 토픽 브랜치를 통합하고자 하는 목적은 같으나, 그 특징은 약간 다릅니다.



- **merge**

변경 내용의 이력이 모두 그대로 남아 있기 때문에 이력이 복잡해짐.

- **rebase**

이력은 단순해지지만, 원래의 커밋 이력이 변경됨. 정확한 이력을 남겨야 할 필요가 있을 경우에는 사용하면 안됨.

merge 와 rebase 는 팀 운용 방침에 따라 구별해 쓸 수 있습니다.

예를 들어 이력을 하나로 모두 모아서 처리하도록 운용한다고 치면 아래와 같이 구별해 사용할 수 있습니다.

- 토픽 브랜치에 통합 브랜치의 최신 코드를 적용할 경우에는 rebase 를 사용,
- 통합 브랜치에 토픽 브랜치를 불러올 경우에는 우선 rebase 를 한 후 merge

# 커밋 수정

- git commit --amend
  - **마지막** 커밋 메시지를 보여줌,  
수정 가능



```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git branch -v
* feat_rmtc c63ccb3 cause you don't rememeber, i'll say that again
master      0ba11c5 junoflow
```

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git commit --amend
```

```
[feat_rmtc c63ccb3] cause you don't rememeber, i'll say that again
```

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Thu Mar 29 10:31:03 2018 +0900
#
# On branch feat_rmtc
# Your branch is up to date with 'origin/feat_rmtc'.
#
# Changes to be committed:
#   new file:   PushByNameHere/joonhoWohn/goback.c
#
~
```

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git commit --amend
[feat_rmtc ef11788] past is past....
Date: Thu Mar 29 10:31:03 2018 +0900
1 file changed, 5 insertions(+)
create mode 100644 PushByNameHere/joonhoWohn/goback.c
```

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git push
To https://github.com/doomsheart/GitPractice.git
 ! [rejected]        feat_rmtc -> feat_rmtc (non-fast-forward)
error: failed to push some refs to 'https://github.com/doomsheart/GitPractice.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

오우 push 를 해볼까요

---

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git log
commit ef11788de14314e85b53e723af20cc0decf940b1 (HEAD -> feat_rmtc)
Author: Joonho Wohn <doomsheart@naver.com>
Date: Thu Mar 29 10:31:03 2018 +0900

    past is past....

commit 0ba11c541eb9802546e208a682414d48d0957fd1 (origin/master, origin/HEAD, master)
Author: Joonho Wohn <doomsheart@naver.com>
Date: Tue Mar 27 10:53:09 2018 +0900

    junoflow

commit a108bc340508bf8921a6dcbb947a33c25efee9a7
Author: Joonho Wohn <doomsheart@naver.com>
Date: Thu Mar 22 16:58:53 2018 +0900

    add lecture files
```

# 뭐야뭐야 로그를 찍어보쇼

---

# 결국 force push.... 지양하는 방법

```
C:\Users\Joonho Wohn\Documents\Dev\GitPractice>git push -f origin feat_rmtc
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 601 bytes | 300.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/doomsheart/GitPractice.git
+ c63ccb3...ef11788 feat_rmtc -> feat_rmtc (forced update)
```

# 왜 amend에서 문제?

- 이미 푸시한 커밋을 수정해서 그럼..
- 이미 푸시한 커밋 수정 방법? -> 푸시는 항상 주의하자....

# 그외의 방법들...

- git revert
- git reset
- git rebase



# 참고자료

- [https://backlog.com/git-tutorial/kr/stepup/stepup1\\_1.html](https://backlog.com/git-tutorial/kr/stepup/stepup1_1.html)
- 짱짱...