

65차

비개발자를 위한 Git 과 Github Page 블로그 만들기

안수빈(@subinium)

@subinium (#수비니움)

고려대학교 사이버국방학과 졸업 예정
SW 마에스트로 10기
카카오 엔터프라이즈 인턴 예정 (2020.01)



Overview Repositories 14 Projects 0 Packages 0 Stars 108 Followers 89 Following 8

Pinned

- subinium.github.io
- ProjectEuler
- Python-Visualization
- Pytorch-Practice
- awesome-deepfake-porn-detection
- swm-fp/Flower-Chrome-Extension

Customize your pins

Awesome.

Subin An
subinium

Edit profile

AI & Algorithm & Front-End World changing technology
Korea University
Seoul, Republic of Korea
subinium@gmail.com
subinium.github.io

Organizations

531 contributions in the last year

Contribution settings

Learn how we count contributions.

Contribution activity

December 2019

Created 2 commits in 2 repositories

subinium/interpretable-AI 1 commit
subinium/subinium.github.io 1 commit

Show more activity

Seeing something unexpected? Take a look at the [GitHub profile guide](#).

Hello Subinium!

안수빈의 블로그

오랫동안 꿈을 그리는 사람은 마침내 그 꿈을 닮아간다

Link

An Subin
Yeah, I am Subinium.

인공지능 페이지 : AI Lookbook
알고리즘 페이지 : 알고리즘에 고통받는 취준생을 위한 안내서
개발&디자인 페이지 : 삼질하는 디밸자와 개자이-나

Recent Posts

- [발표] 의미있는 PS를 하기 위해서..
- 개인이 생각하는 알고리즘(PS/CP) 공부 유형 및 보완법
- 원본은 Notion에 있지만, 여기도 올려봅니다.-)
- 2020 KAKAO BLIND RECRUITMENT Review
- 도막을 위한 작은 실태
- Part 2. Introduction to Ensemble Learning : Boosting
- 정말 오픈판에 돌아온 양상철 포스팅 2번짼
- Beginner Guide : Missing Data Handling
- Missing Data는 어떻게 관리할 수 있을까요? Handling을 해봅시다!!
- Beginner Guide : Feature Selection

오늘의 주제 : Git과 Github, 그리고 블로그

목차

Git 기초

Git이 필요한 이유

Git의 기본 Idea

Easy Git

Github 실습

Github과 Sourcetree

예시와 함께하는 기초 명령어

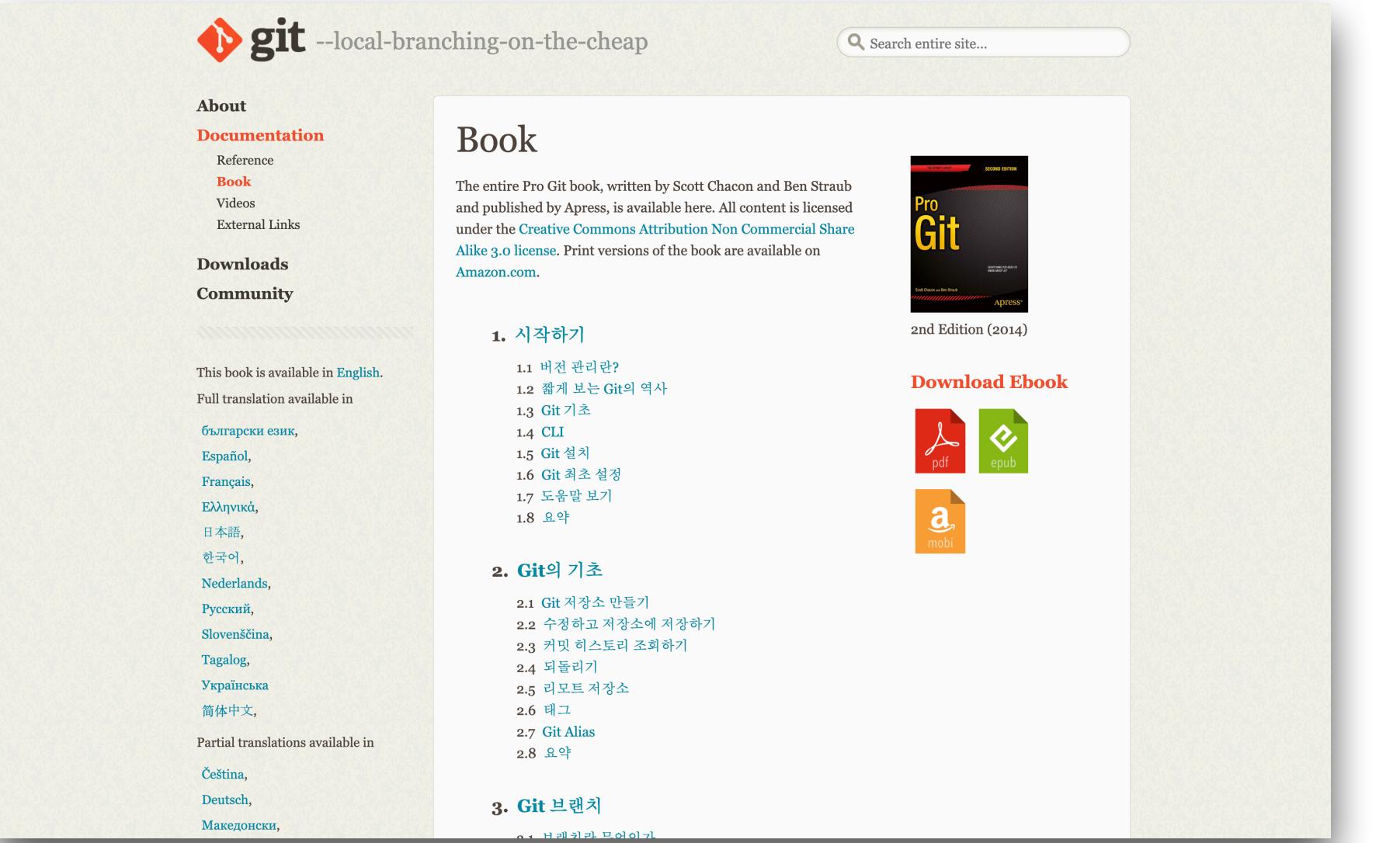
알아두면 좋은 심화 명령어

Github Page 활용하기

Markdown 문법

Github Page 구조 및 실습

앞으로 관리 방법



<https://git-scm.com/book/ko/v2>

오늘 배울 것은 모두 인터넷에 있어요

하지만 엄청난 양

오늘은 핵심적인 부분들을 배워봅시다.

Git 기초

Git이 무엇이고, 왜 써야할까?
초보 개발자에게 Git이란?

이런 상황 ...

1



2019_최종보고서

2



2019_최종보고서_수정

3



2019_최종보고서_수정2

4



2019_최종보고서_진짜최종

5



2019_최종보고서_리얼최종

6



2019_최종보고서_살려줘

*

: 파일의 버전

끊임없는 수정본의 문제점

1. **변경 사항 파악 어려움** : 무엇을 바꿨는가?
2. **되돌리기 어려움** : CMD / CTRL + Z 의 한계, 삭제 파일 추적 불가
3. **용량 차지** : N개가 있다면 N배
4. **협업이 어려움** : 누가 언제 무엇을 건드렸나



시간여행처럼 원하는 시점으로 움직이기만 하고 싶다!

VCS (Version Control System, 버전 관리 시스템)

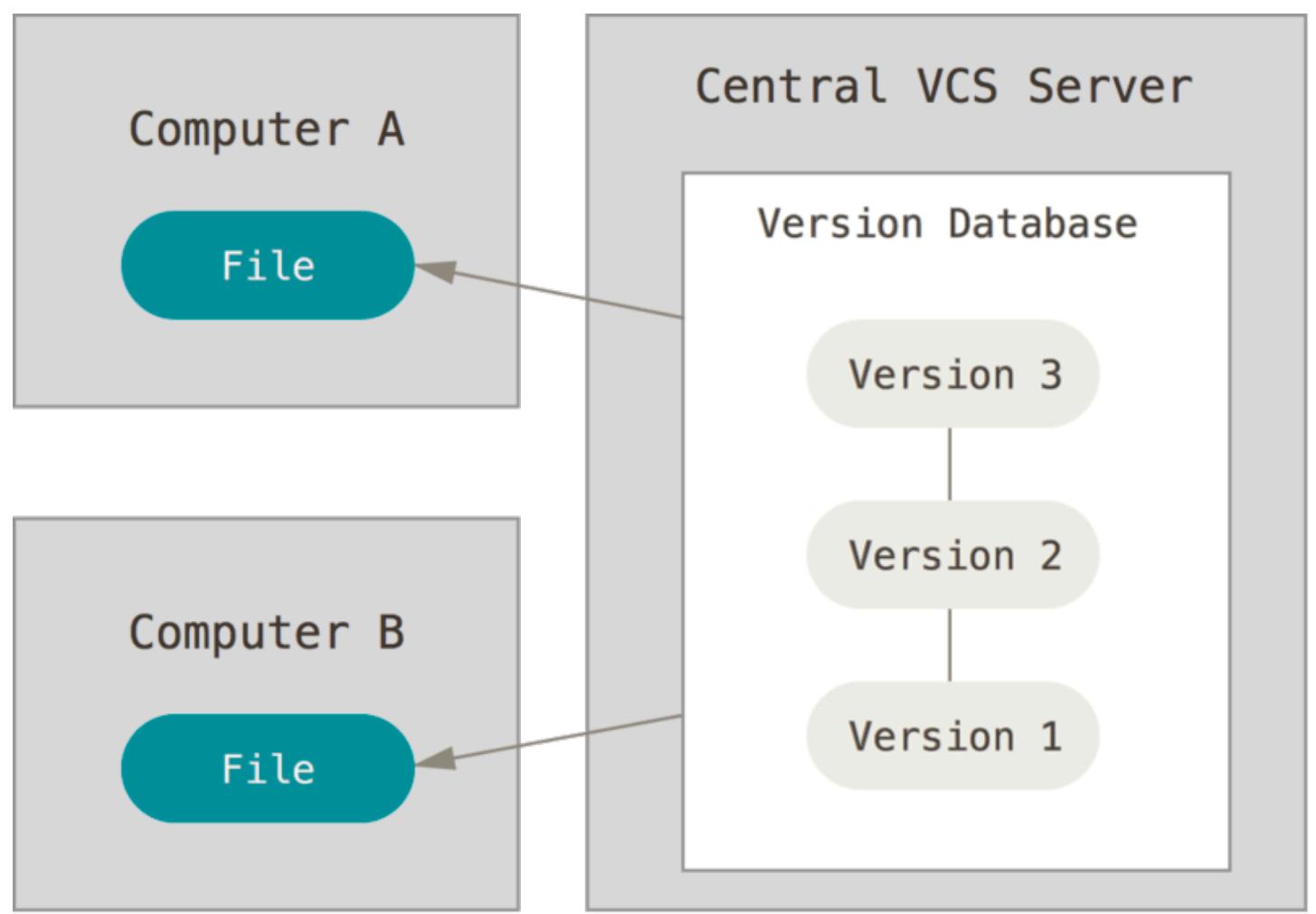
파일의 **변경 사항**을 저장하고, 원하는 시점의 버전을 다시 꺼내올 수 있는 시스템

Snapshot

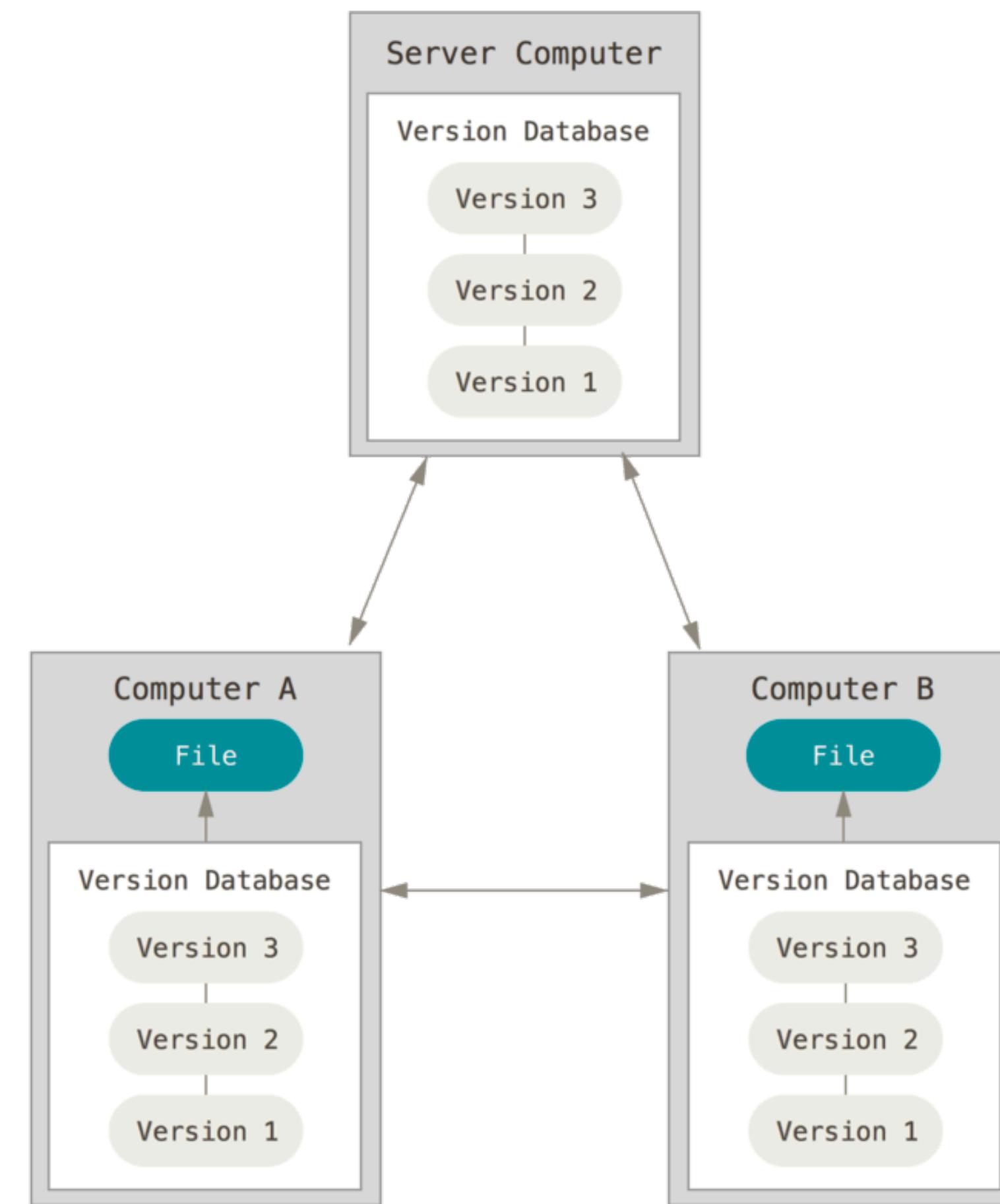
특정 시점에서 파일의 상태 (현재 상태의 모든 정보)

Delta

파일의 이전 상태와 비교한 변경사항

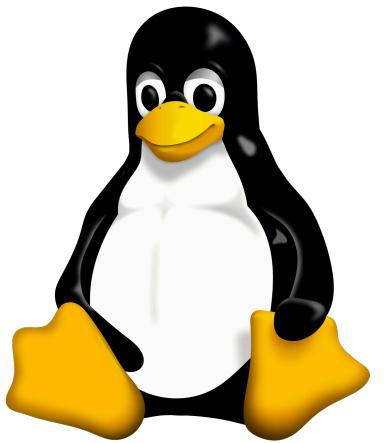


중앙집중식 버전 관리 시스템 (CVCS)



분산 버전 관리 시스템 (DVCS)

SIMPLE HISTORY OF GIT



리눅스(Linux)의
단순한 버전 관리



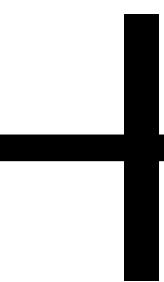
1991



2002



Linux 창시자 Linus Torvalds를 중심으로
Linux 개발 커뮤니티에서 자체 툴 개발



2005



BITKEEPER
Scalable Version Control

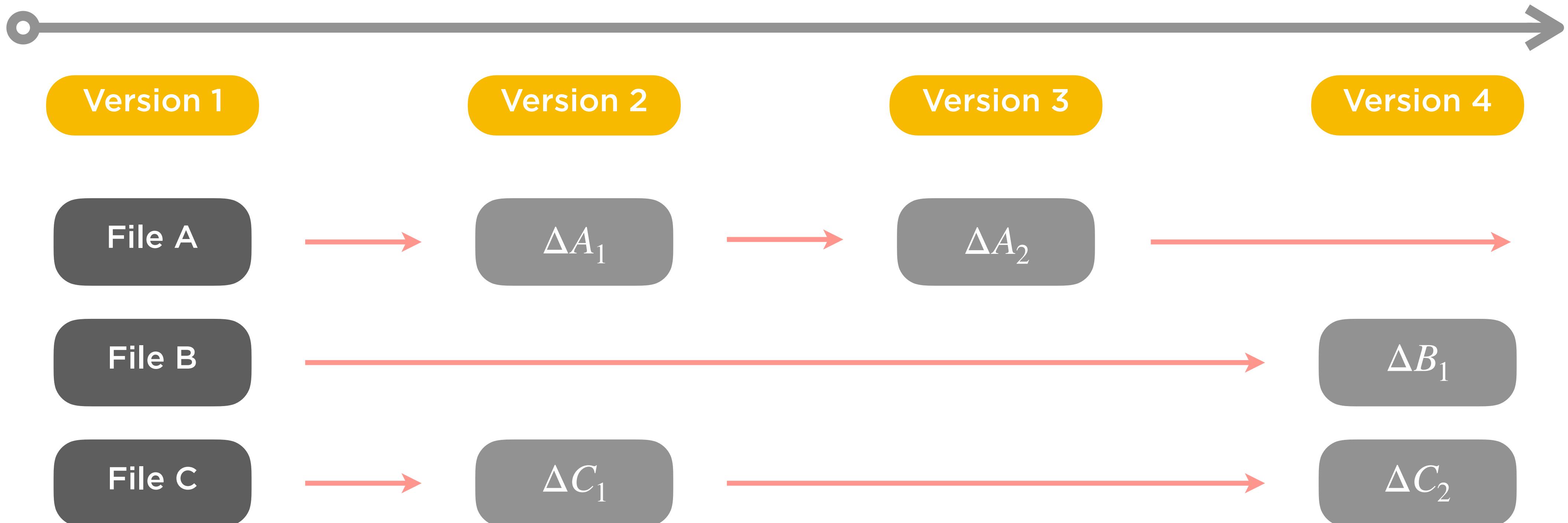
리눅스(Linux) 커널에서
DVCS인 BitKeeper 사용



- 빠른 속도
 - 단순한 구조
 - 비선형적인 개발
 - 완벽한 분산
 - Linux 커널 같은 대형 프로젝트에도 유용할 것
- 

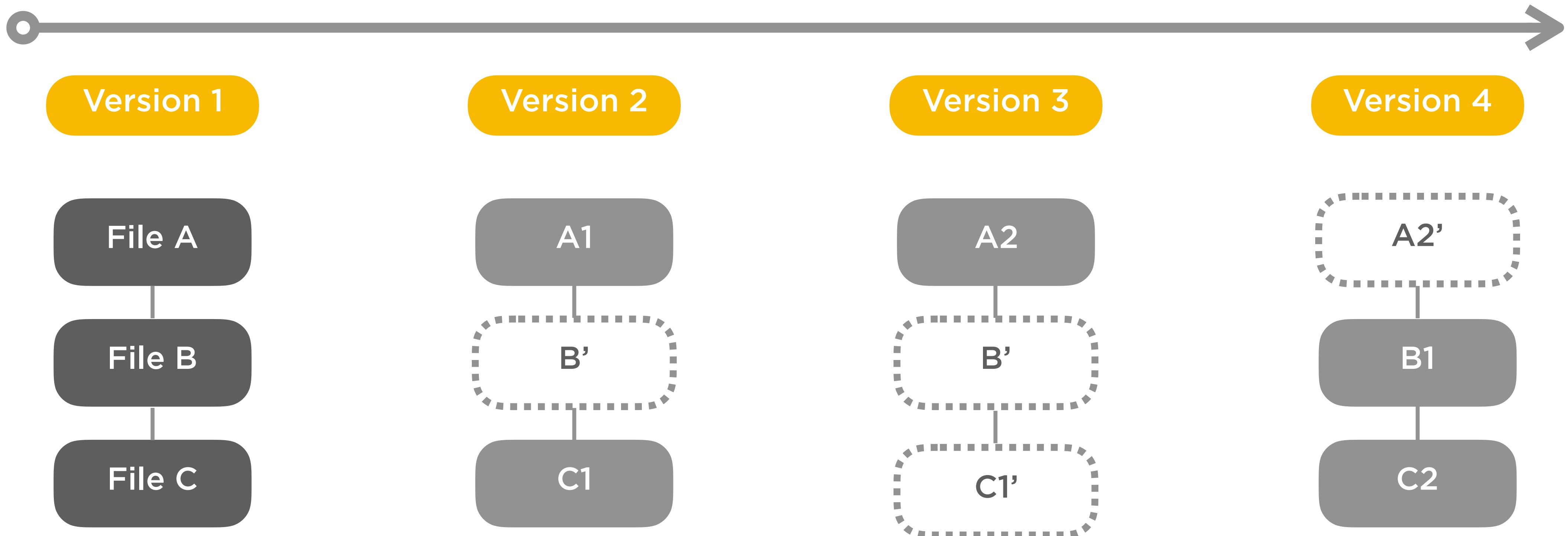
Subversion (SVN)

CVCS(Central VCS, 중앙집중식) 중 대표적인 시스템
파일의 모든 변경 사항을 저장하는 방법



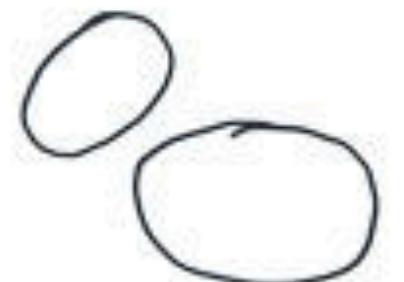
Git

DVCS, 저장소의 파일 시스템 전체를 스냅샷으로 취급
변경하지 않은 파일은 새로 저장하지 않고, 링크만 저장

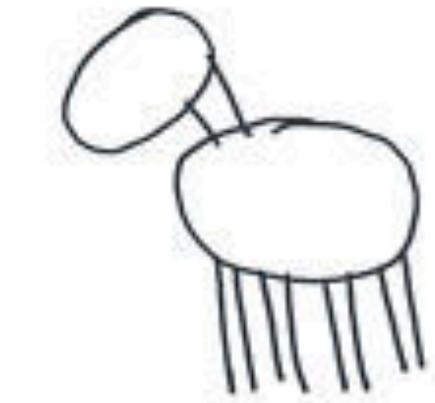


HOW TO: DRAW A HORSE

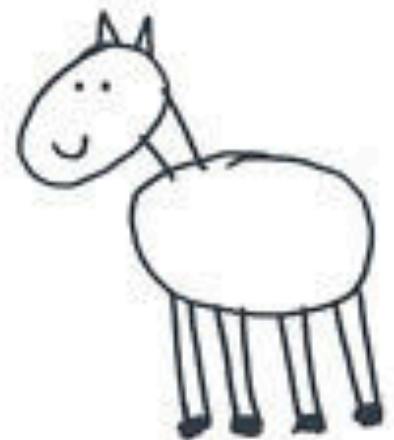
BY VAN OKTOP



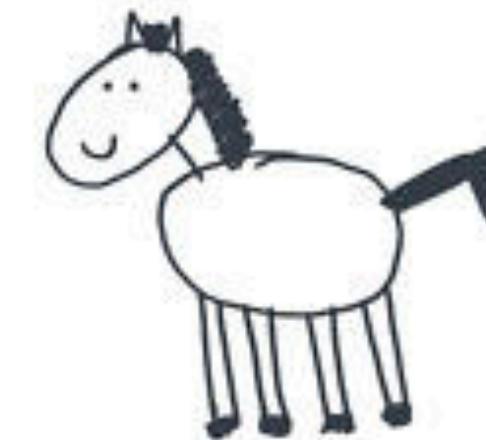
① DRAW 2 CIRCLES



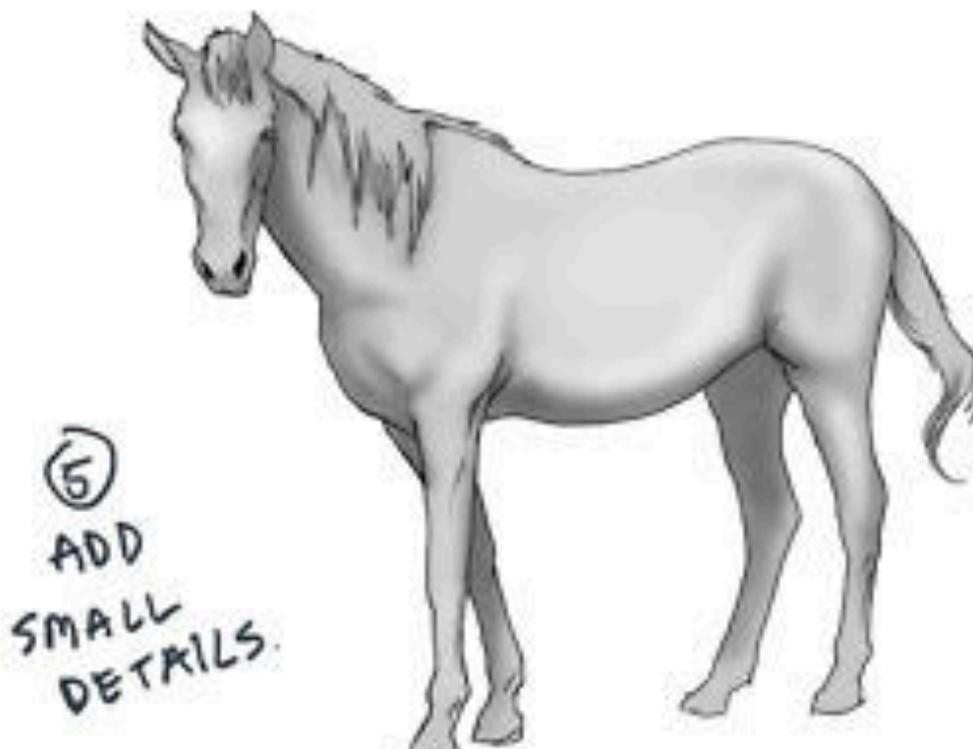
② DRAW THE LEGS



③ DRAW THE FACE



④ DRAW THE HAIR

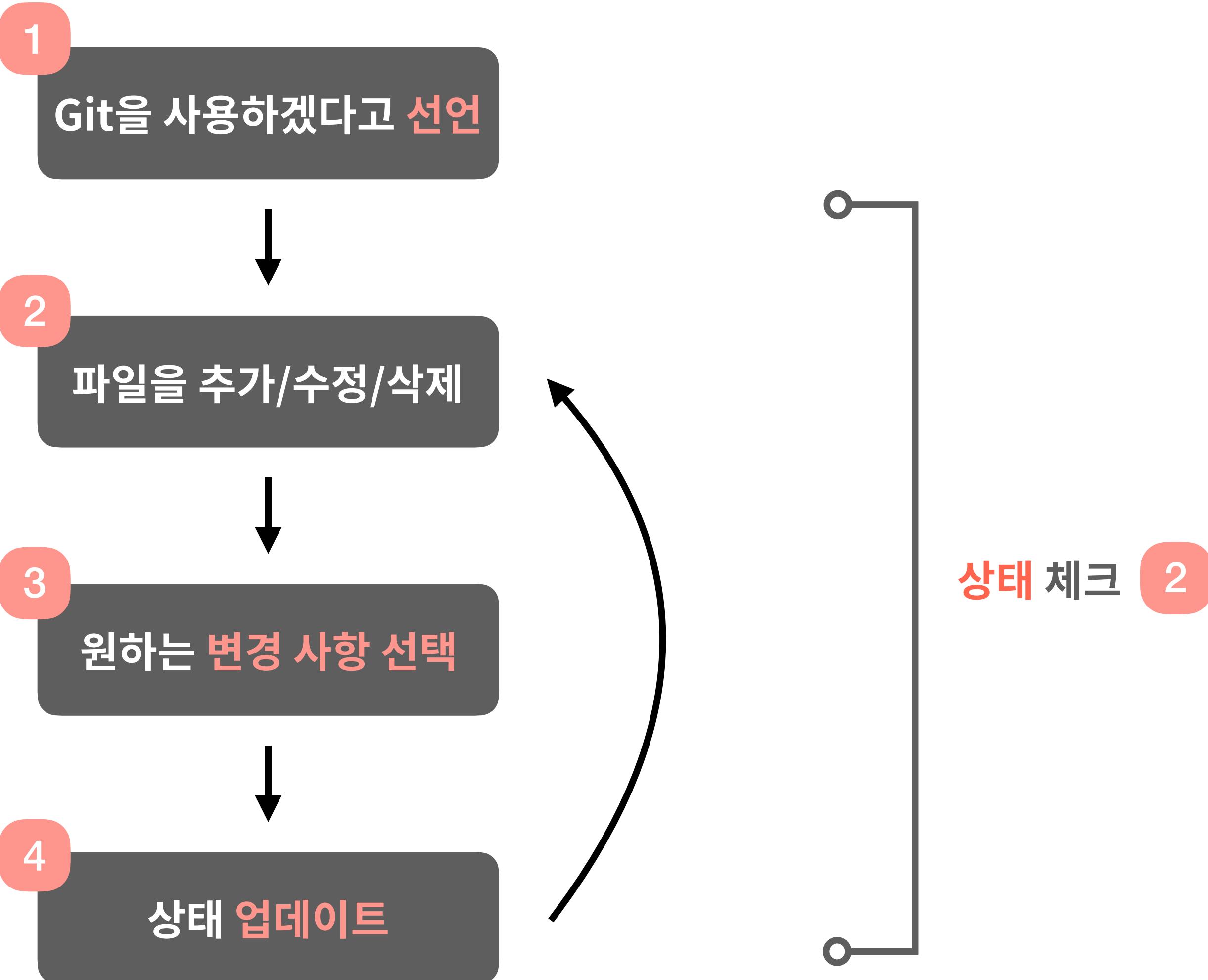


⑤
ADD
SMALL
DETAILS.

쉬운 Git workflow부터 살펴봅시다.

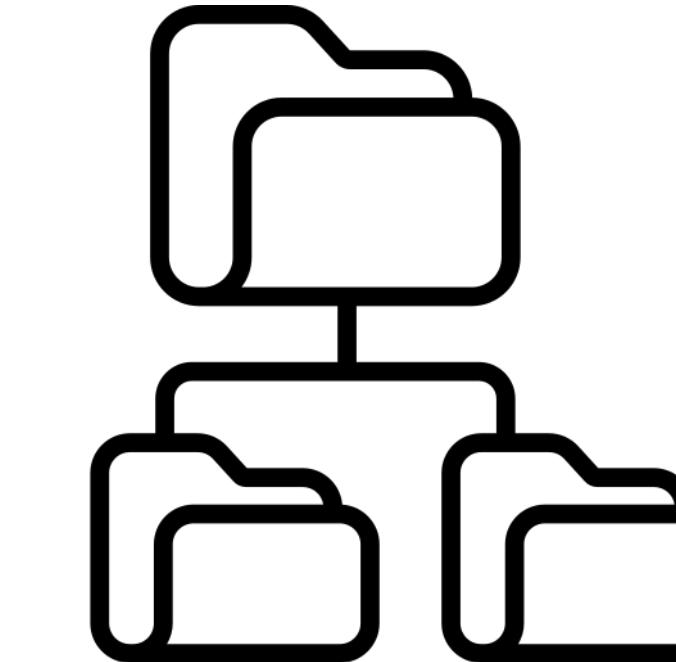
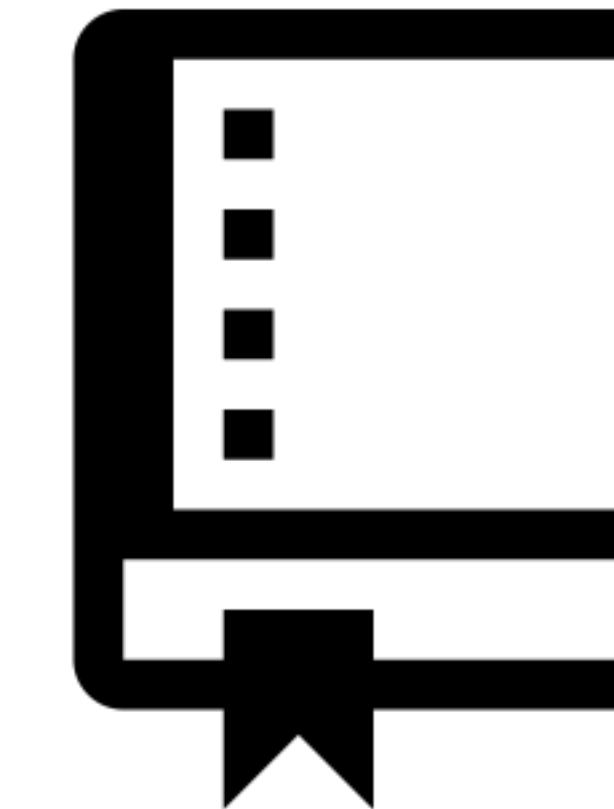
오늘은 3~4단계까지 :)

Easy Git



Git 사용 선언

선언과 저장소 (Initialization & Repository)



흔히 Repo(레포)라고 부름

사용자가 변경한 모든 내용을 추적하는 공간 (하나의 디렉토리로 볼 수 있음)

현재 상태, 변경 시점, 변경한 사용자, 설명 텍스트 등 저장



Git 사용 선언

Git은 이제 Local에서 가능한 상태

모든 것은 local에서 저장 및 버전 관리 가능 (서버 죽어도 무상관)

원격 서버에는 나중에 올릴 수 있음

즉, Wi-Fi 없는 환경에서도 작업 가능

Git은 데이터를 추가만 할 수 있다.

파일 삭제 == 삭제 기록 추가

=> -1 == + (-1)

데이터베이스에 저장한 순간부터는 삭제까지 추적

하지만 Git은 파일을 추적하지 않는다.

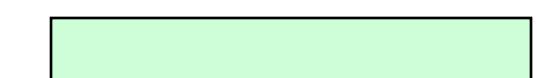
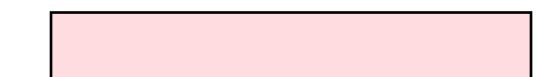
Git 사용 선언

Git은 신기하게 파일을 추적 X.

대신 파일의 내용 단위로 추적하고, 각 문자와 줄을 추적합니다.

빈 디렉토리는 추적 X

```
28 31 ## Education
29 32
30 33 - 포항제철중학교, 2010 ~ 2013
31 34 - 서울과학고등학교, 2013 ~ 2016
32 - 고려대학교 정보보호학부, 용합보안융합전공 2016 ~
33 + 고려대학교 정보보호학부 사이버국방학과, 용합보안융합전공 2016 ~
34 ## Awards
35
36 ### Algorithm : ICPC
37 - 2016 ACM-ICPC Asia Daejeon Regional Contest 12th place (AC_FROM_ZZAM)
38 - 2018 ACM-ICPC Asia Seoul Regional Contest 8th place (장려상, GoInMulDaeGoSipDa)
39 - 2018 ACM-ICPC Asia Hanoi Regional Contest 14th place (GoInMulDaeGoSipDa)
40 + 2016 ACM-ICPC Asia Daejeon Regional Contest **12th** place (AC_FROM_ZZAM)
41 + 2018 ACM-ICPC Asia Seoul Regional Contest **8th** place (장려상, GoInMulDaeGoSipDa)
42 + 2018 ACM-ICPC Asia Hanoi Regional Contest **14th** place (GoInMulDaeGoSipDa)
43
44 ### Algorithm : ELSE
45 - 2016 UCPC 전국 대학생 프로그래밍 대회 동아리 연합 여름 대회 26th place (WellMadeSubinium)
46 - 2018 UCPC 전국 대학생 프로그래밍 대회 동아리 연합 여름 대회 21th place (해킹의반대말은달린)
47 - 2018 SCPC(삼성 대학생 프로그래밍 경진대회) 본선 진출
48 + 2018 SCPC(삼성 대학생 프로그래밍 경진대회) 온사이트 본선 진출
49 + 2019 SCPC(삼성 대학생 프로그래밍 경진대회) 온사이트 본선 진출
50 - 2018 TCO winning T-shirt (Top 300)
```

 추가
 삭제

Git 사용 선언

어떤 파일이 상태에 따라 계속 바뀌고, 딱히 **저장할 필요가 없다면?**
저장하지 말고, 굳이 관리하고 싶지 않은 파일은 따로 처리하자.

파일 상태

Untracked

Tracked

Git 사용 선언

파일 상태

Unmodified

이전에 버전과 비교하여 **수정된 부분이 없는 상태**

Modified

이전에 버전과 비교하여 **수정된 부분이 있는 상태**

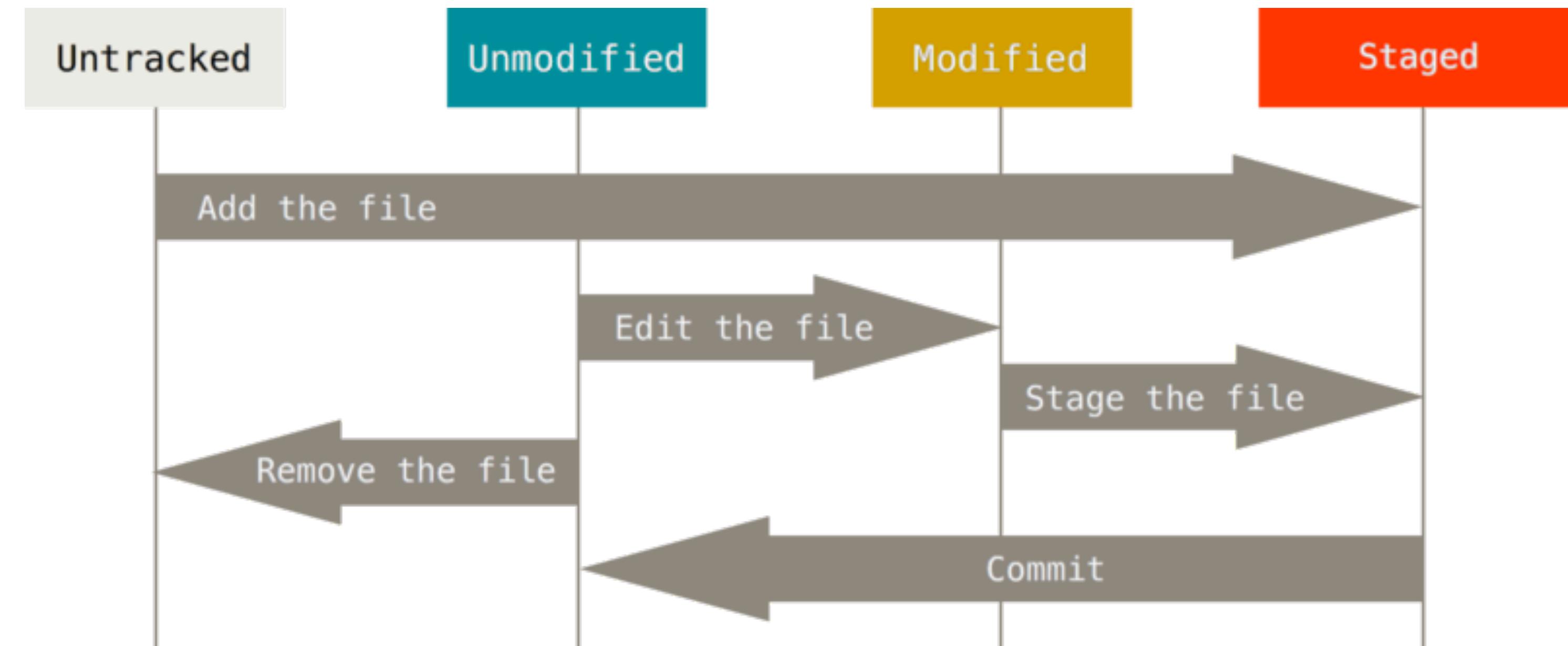
Staged

저장(커밋)을 위해 준비된 상태 (**스테이징, Staging**)

Git 사용 선언

스테이징(Staging)을 하면 커밋하고 싶은 파일 선택
커밋(Commit)을 하면 새로운 버전으로 업로드

파일 상태



변경 사항 선택

스테이징은 업로드를 2번하는과정으로 보이는데
왜 굳이 한 번 더 과정을 거치는걸까

커밋 전, 스테이징이 필요한 이유 (1)

Git 사용 선언

파일 상태

변경 사항 선택

여러 작업 중, 일부분만 커밋해야 할 때

작업을 하고 있는 수빈

갑작스럽게 팀원이 특정 기능(특정 파일)을 고친 부분까지 업로드 해달라고 부탁

하지만 수빈은 그 기능 뿐만 아니라 다른 부분도 수정중

스테이징 없이 바로 커밋한다면?

1. 지금 작성하고 있는 파일들 전부 커밋
2. 필요한 파일만 남기고 커밋한 다음 다시 가져와서 작업

커밋 전, 스테이징이 필요한 이유 (2)

Git 사용 선언

파일 상태

변경 사항 선택

커밋 전 상태를 수정 또는 체크할 때

커밋을 할 때는 수정 사항에 메세지를 남김

메세지에 오타를 내거나, 파일을 잘못 커밋하는 경우는 충분히 발생 가능

서버에서 직접 수정하는 방법은 위험

그럼 보다 안전하게 커밋하는 방법은?

업로드하고 싶은 파일과 하고 싶지 않은 파일을 커밋하기 전에

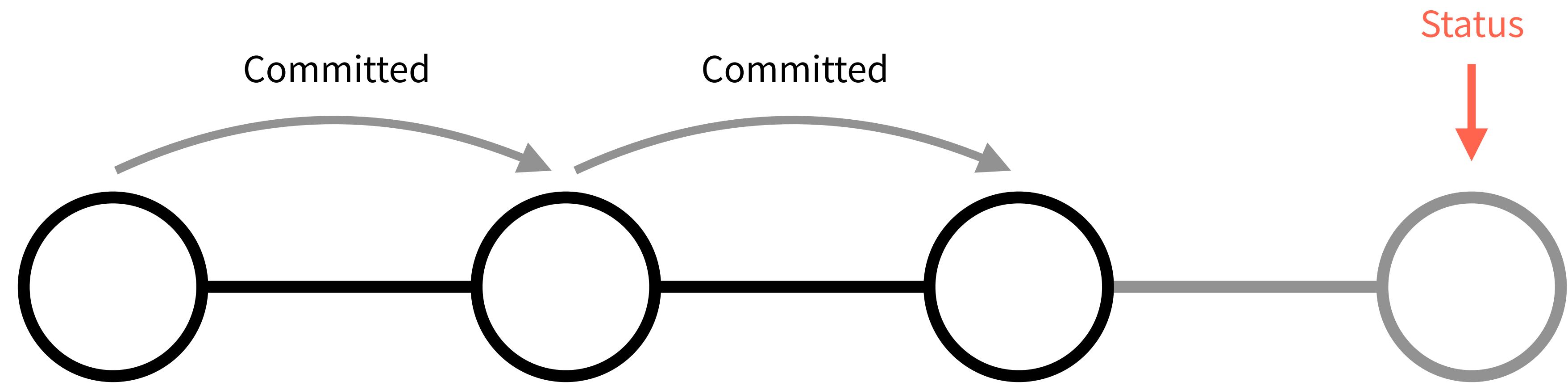
관리해줄 수 있는 공간이 Stage인 것이다.

Git 사용 선택

파일 상태

변경 사항 선택

커밋 완료



커밋을 하면 이상한 40자리 숫자 + 알파벳 조합이 생긴다?!

내용을 주소로 활용 (Content-addressable Key-Value Storage)

Git 사용 선언

파일 상태

변경 사항 선택

커밋 완료



상태를 찾기 위해서는 Key가 필요, 버전의 주소

내용(파일 구조)등을 Hash 값으로 만들고 상태를 나타냄

여기서는 SHA1 해시 값을 사용하여 40자리로 표현

* Hash : 임의의 데이터를 고정된 크기의 데이터로 바꾸는 과정

Git 사용 선언

파일 상태

변경 사항 선택

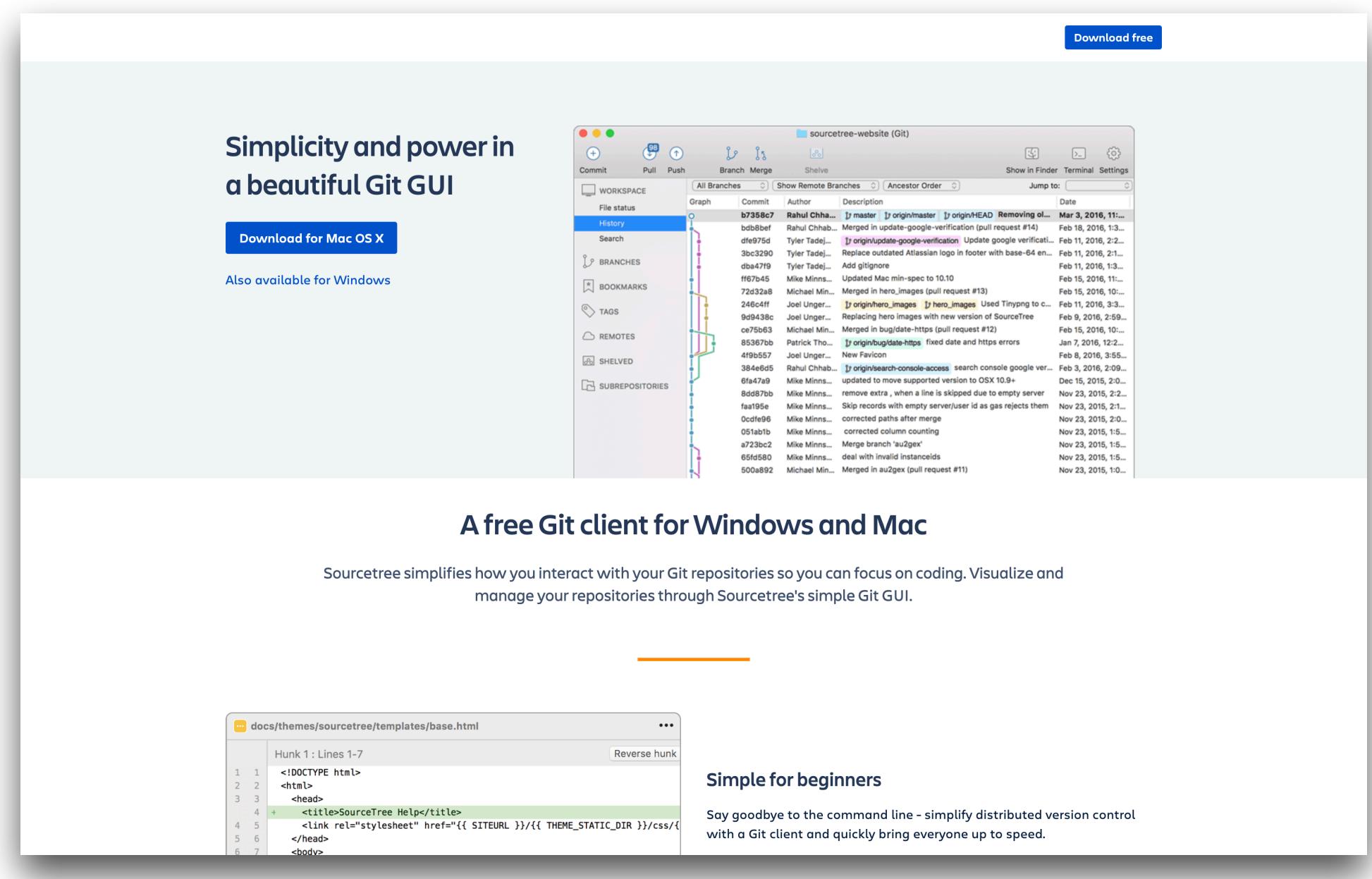
커밋 완료

Commit Hash 값으로 Checkout하면 버전 이동까지 완료!

기본적인 훌로 Git 사용법은 끝났습니다

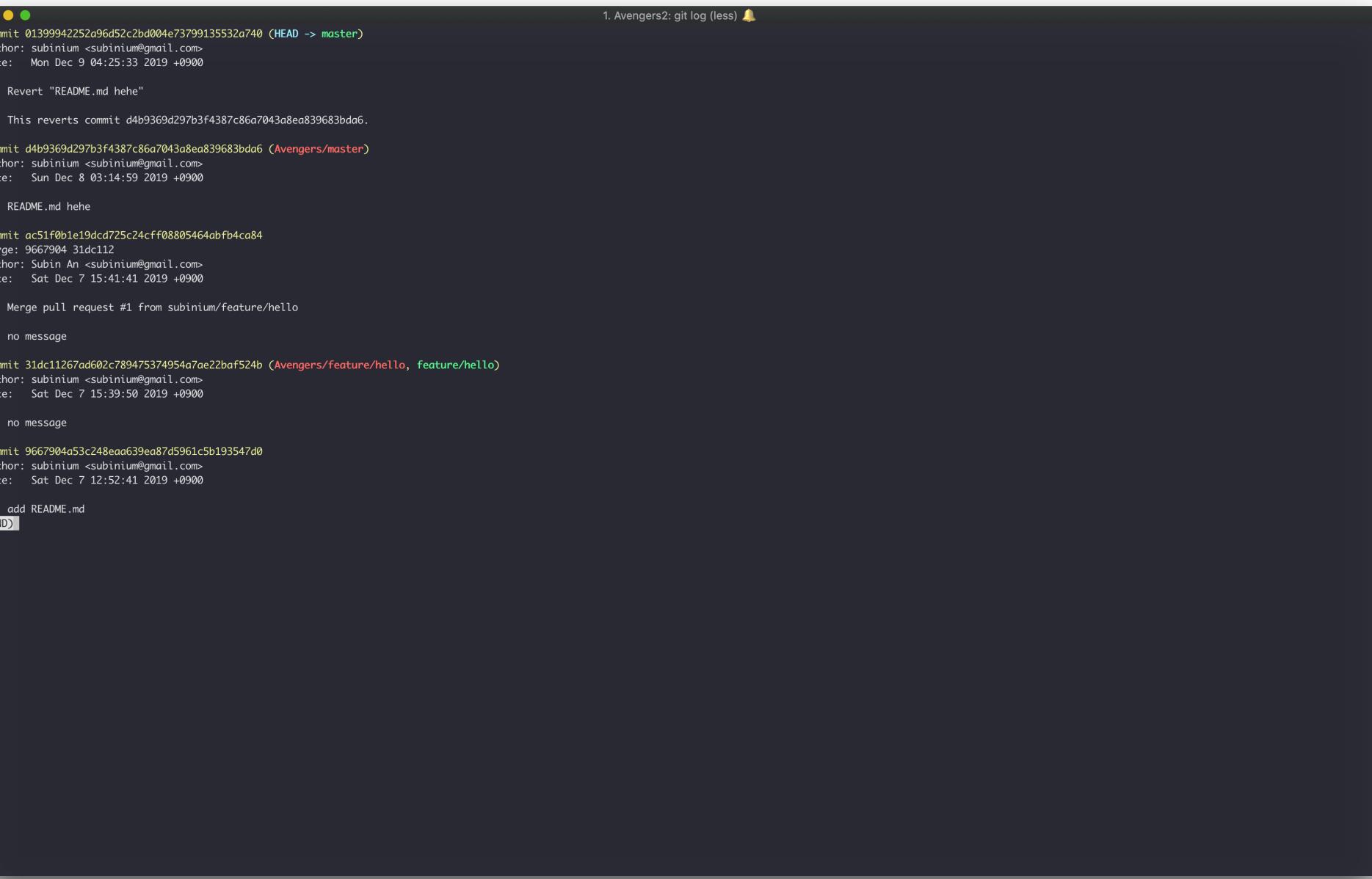
이것만 하더라도 버전 관리는 가능합니다

Git은 꼭 코드를 알아야하나요? NO!



<https://www.sourcetreeapp.com/>

하지만 코드도 조금 알면 편하다.



A screenshot of a terminal window titled "1. Avengers2: git log (less)". The window displays a list of git commits. The commits are as follows:

- commit 01399942252d96d52c2bd004e73799135532a740 (HEAD -> master)
Author: subinium <subinium@gmail.com>
Date: Mon Dec 9 04:25:33 2019 +0900
Revert "README.md hehe"
This reverts commit d4b9369d297b3f4387c86a7043a8ea839683bda6.
commit d4b9369d297b3f4387c86a7043a8ea839683bda6 (Avengers/master)
Author: subinium <subinium@gmail.com>
Date: Sun Dec 8 03:14:59 2019 +0900
README.md hehe
- commit a51f081d19acd725c24cff08805464dbfb4ca84
Merge: 9667304 31dc112
Author: Subin An <subinium@gmail.com>
Date: Sat Dec 7 15:41:41 2019 +0900
Merge pull request #1 from subinium/feature/hello
no message
- commit 31dc11267ad602c789475374954a7ae22ba524b (Avengers/feature/hello, feature/hello)
Author: subinium <subinium@gmail.com>
Date: Sat Dec 7 15:39:50 2019 +0900
no message
- commit 9667304d53c248ead639ea87d5961c5b193547d0
Author: subinium <subinium@gmail.com>
Date: Sat Dec 7 12:52:41 2019 +0900
add README.md
(END)

Terminal, git bash 등

<https://git-scm.com/downloads>

오늘은 2개 다 설치하고 연습하는 날 :)

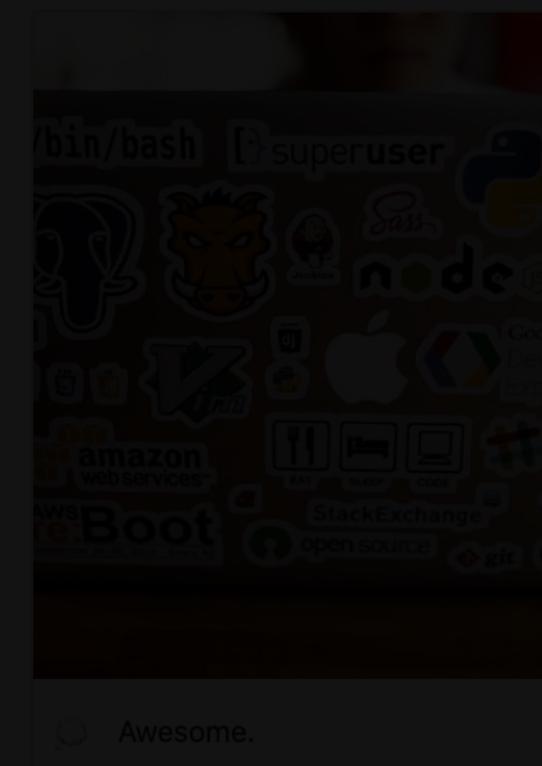
설치 그리고 계정과 연결

CLI 가벼운 명령어 연습



Search or jump to...

Pull requests Issues Marketplace Explore



Overview Repositories 14 Projects 0 Packages 0 Stars 108 Followers 59 Following 8

Pinned

Customize your pins

subinium.github.io
subinium's blog
HTML ★ 9 ⚡ 6

ProjectEuler
Project Euler source code
C++ ★ 3

Python-Visualization
different Python Visualization

Pytorch-Practice
PyTorch Practice

awesome-deepfake-porn-detection

papers, repos, datasets : deepfake and porn detection using deep learning

★ 34 ⚡ 5

swm-fp/Flower-Chrome-Extension

JavaScript ★ 3 ⚡ 5

Subin An

subinium

Edit profile

AI & Algorithm & Front-End World changing technology

Korea University

Seoul, Republic of Korea

subinium@gmail.com

subinium.github.io

Organizations



Github 실습하기

실습으로 익히는 Github

핵심 명령어부터 심화 명령어까지

Contribution activity

December 2019

2019

Created 2 commits in 2 repositories
subinium/Interpretable-AI 1 commit
subinium/subinium.github.io 1 commit

2018

2017

2016

2015

Show more activity

Seeing something unexpected? Take a look at the GitHub profile guide.

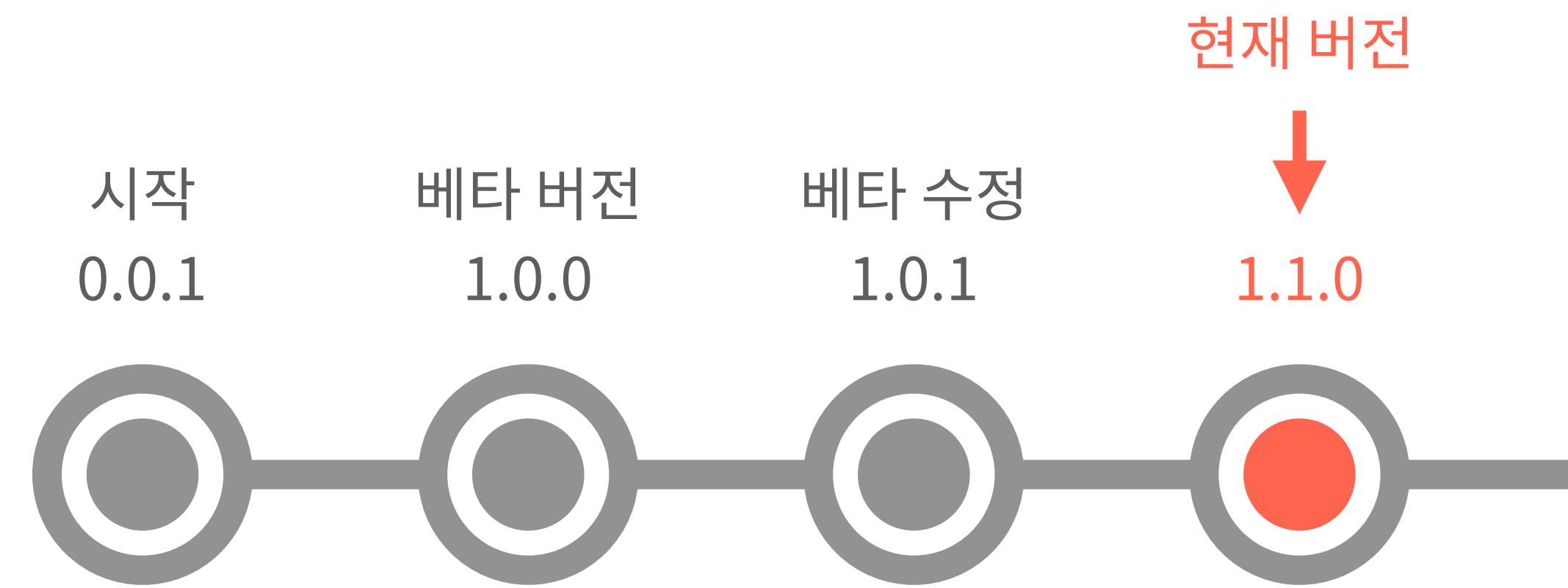
이제 협업과 분산 버전 관리 측면으로
좀 더 DEEP하게 살펴봅시다

여러 사람이 협업하기 위해서는



원하는 원격 호스팅 서비스를 쓰면 됩니다.
한 번 둘러봅시다.

우선 소프트웨어 버전은 다음과 같이 개발됩니다.



[A].[B].[C] 와 같이 보통 .으로 구분되는 3개의 숫자로 표현됩니다.
간단히 다음과 같이 생각하시면 됩니다.

- [A] : Major : 이전 버전과 호환이 안되는 아예 바뀐 상태
- [B] : Minor : 기능 추가, 변경
- [C] : Patch : 미미한 내부 에러 수정

하지만 프로젝트에서 여러 수정이나 기능 개발이

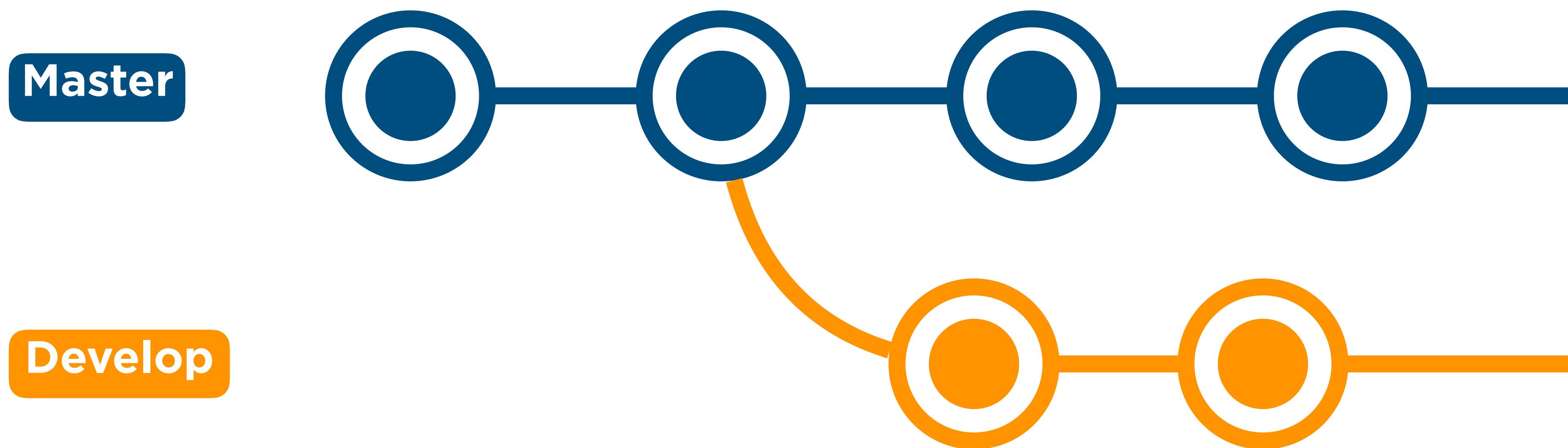
순차적으로 진행될까요?

대부분의 프로젝트는 NO!

작업을 병렬로 하기 위해서는

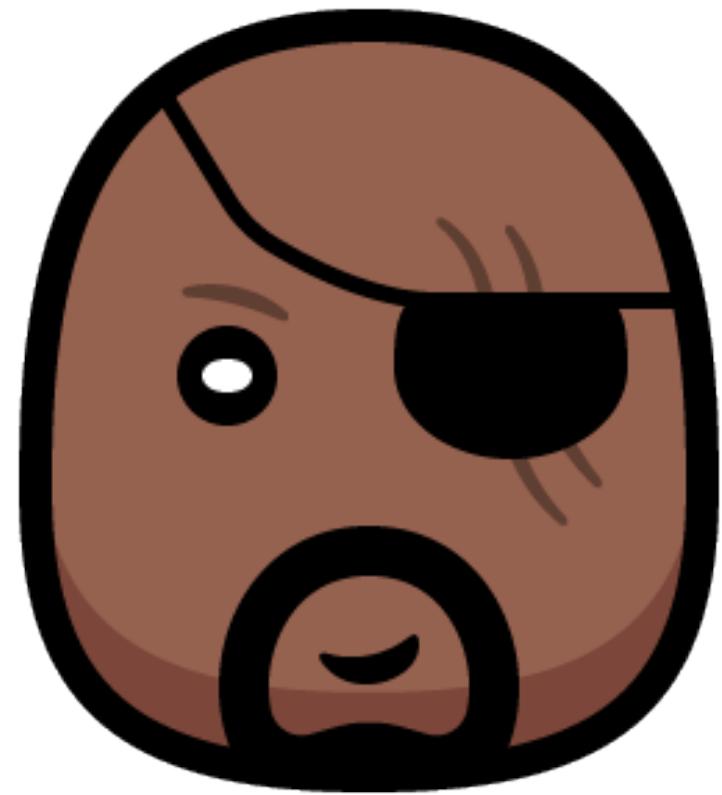
어떤 특별한 기능이 필요합니다.

그래서 나온 브랜치 (Branch)!



지구를 지키는
[어벤저스] 프로젝트를
통해 브랜치 등
Github에 대해
더 알아보겠습니다.





프로젝트 총 관리자 및 시작자

누 퓨리 시점

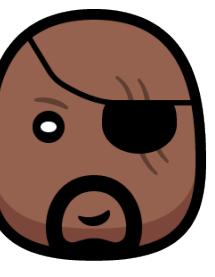


프로젝트 시작 선언

`git init` Git 초기화를 의미 로컬에서 진행



‘지구를 지키는 어벤저스 프로젝트를 기획하겠습니다.’
우선 시작 버전은 **master** branch에 기록될 것입니다.



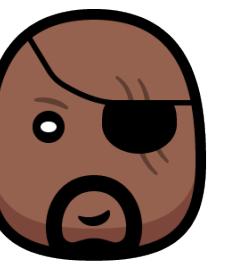
버전 저장은 .git



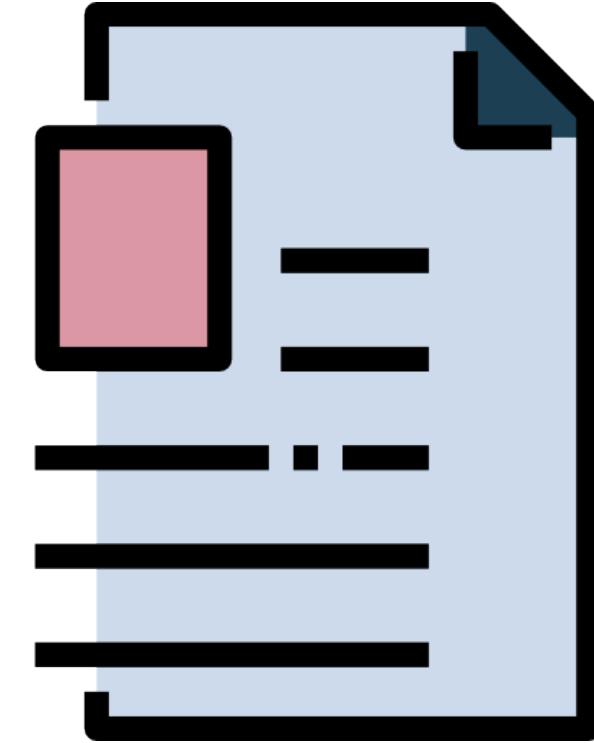
.git

버전 관리 정보는 이 폴더에 있습니다.

이 폴더를 지우면 모든 버전 관리 기록은 사라집니다.



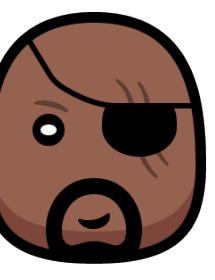
저장이 싫다면 .gitignore



.gitignore

추적을 무시하고 싶다면?

양식(정규표현식)을 맞춰서 .gitignore 파일에 작성하면 됩니다.



README.md : 설명 작성

프로젝트의 설명, 사용방법, LICENSE 등을 기술 + Repo의 Main Page 역할

README.md

지구를 지키는 어벤저스

필요한 사항

- 아이언맨
- 캡틴아메리카
- 등등

작성할 파일 명과 내용

README.md

PyTorch

PyTorch is a Python package that provides two high-level features:

- Tensor computation (like NumPy) with strong GPU acceleration
- Deep neural networks built on a tape-based autograd system

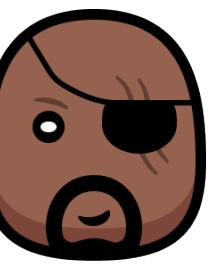
You can reuse your favorite Python packages such as NumPy, SciPy and Cython to extend PyTorch when needed.

- More about PyTorch
- Installation
 - Binaries
 - From Source
 - Docker Image
 - Building the Documentation
 - Previous Versions
- Getting Started
- Communication
- Releases and Contributing
- The Team

System	2.7	3.5	3.6
Linux CPU	build passing	build passing	—
Linux GPU	build passing	build passing	—
Windows CPU / GPU	—	build failing	—
Linux (ppc64le) CPU	build failing	—	build failing
Linux (ppc64le) GPU	build failing	—	build failing

See also the ci.pytorch.org HUD.

유명한 OpenSource인 PyTorch의 README



README.md : 작성 팁

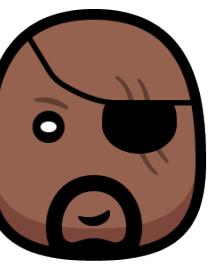
1. 프로젝트 내용 (이미지/로고)
2. 설치 방법
3. 코드 예제
4. 개발 환경 설정 방법
5. 기여방법
6. 로그 변경
7. 크레딧
8. 라이센스
9. 연락처

파일 스테이지로 올리기



git add [file] [file]을 스테이지로 올림, 풀더나 전체도 가능

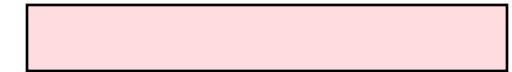




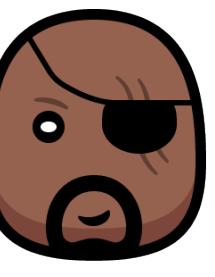
파일 상태 체크하기

git status

git diff



지금 어떤 파일들을 수정했고, 스테이징 했지?
어떤 파일이 얼마나 바뀐거지?

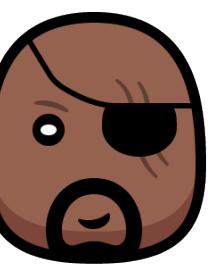


스테이지에 있는 내용 커밋

git commit -m “add README.md” 간단한 설명과 함께 commit



이제 내 버전에서는 확실하게 기록했다.
새로운 버전으로 재탄생 (1차 작업 완료!)



커밋 기록 살펴보기

git log 이전 commit 기록 살펴보기

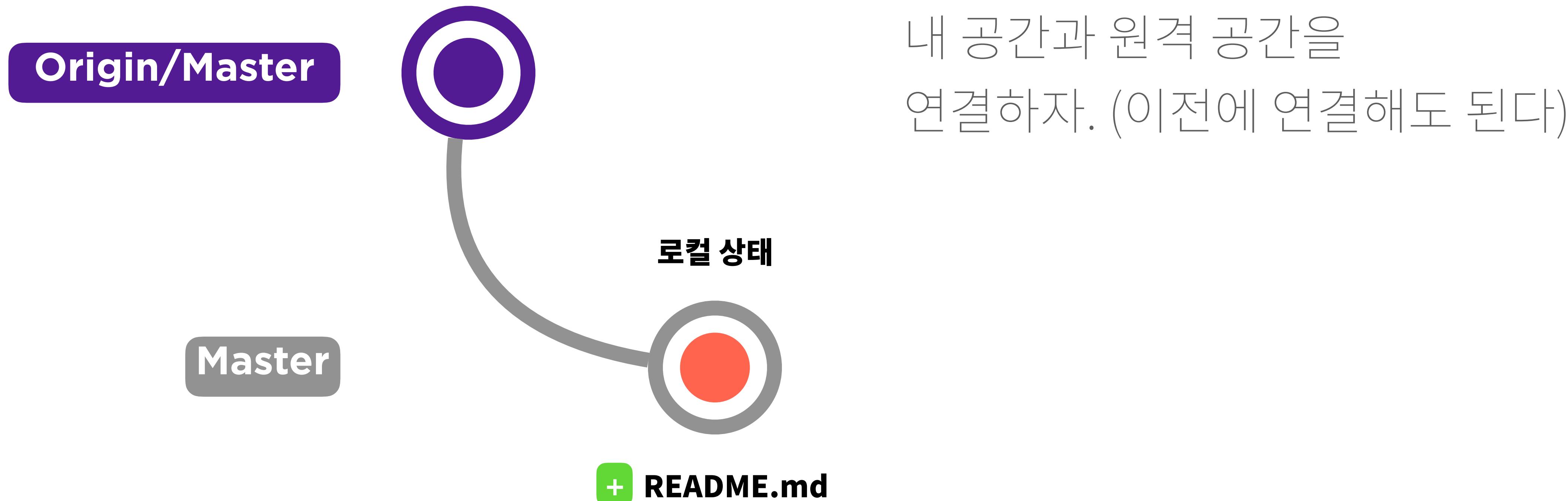


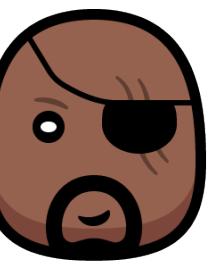
commit, Date, Author, Message 등을 확인하자



원격 저장소와 연결

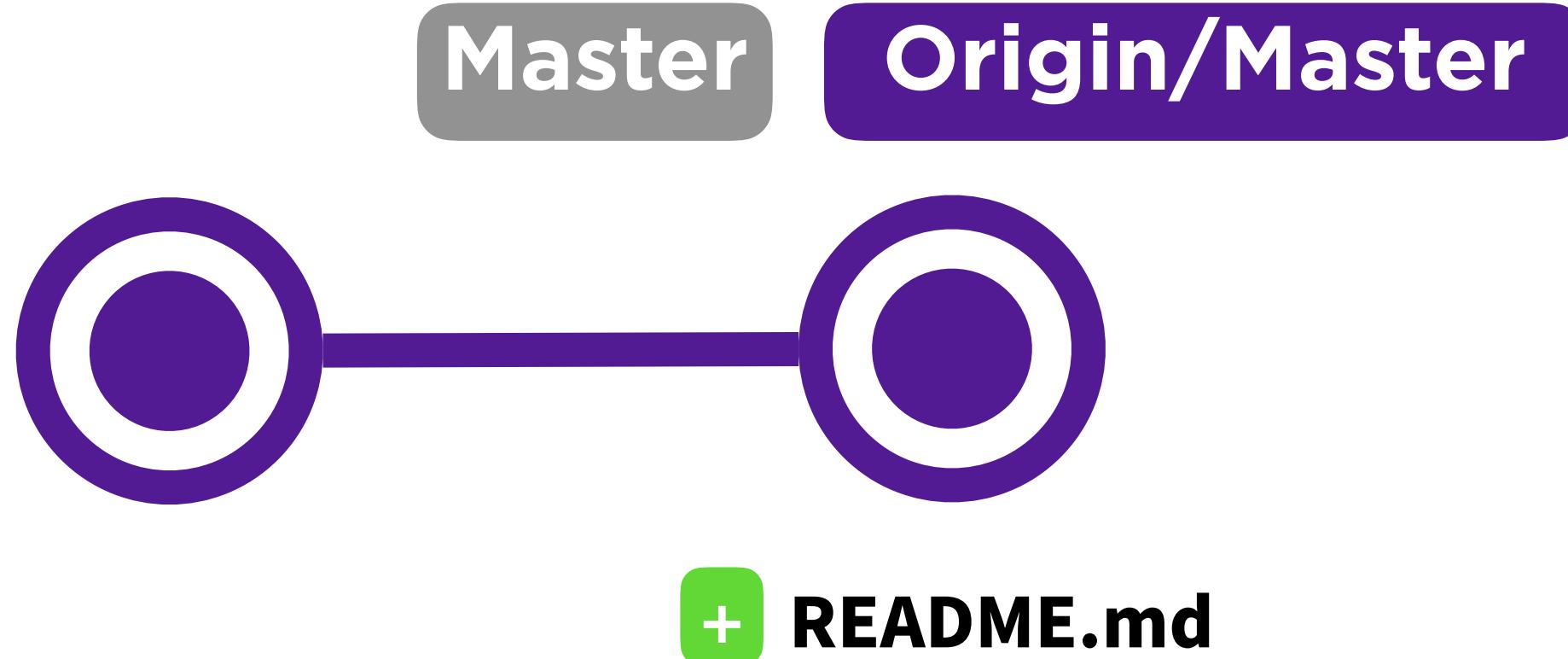
`git remote add origin [url]` origin이라는 이름으로 [url]과 연결





원격 저장소로 올리기

`git push origin master` 원격 저장소 master branch에 업데이트



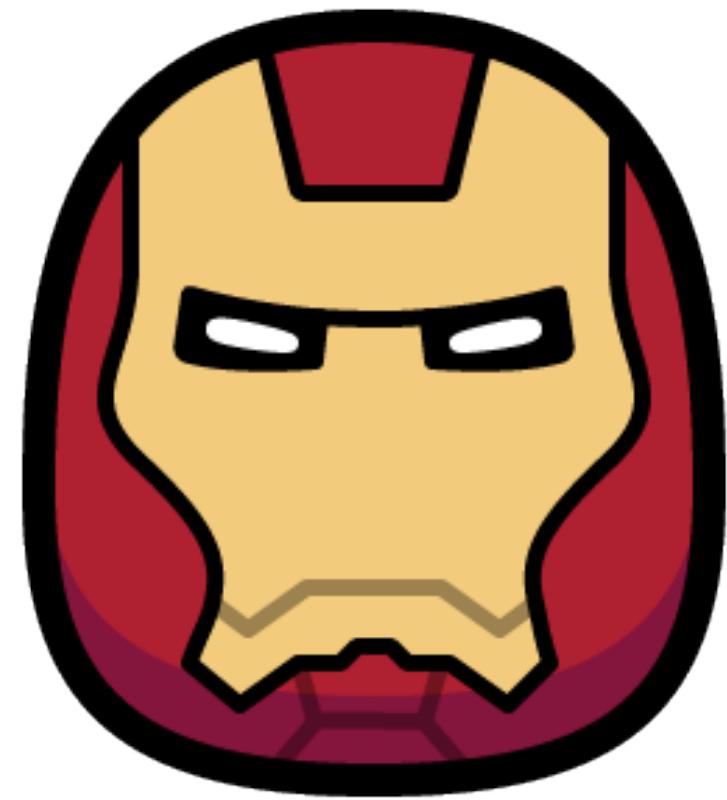
이제 내가 보는 버전과
모두가 보는 버전이 동기화 완료!

실습 TIME

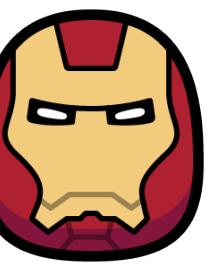
Terminal/Git Bash + SourceTree

Init, Add, Commit, Push

Diff, Status, Log

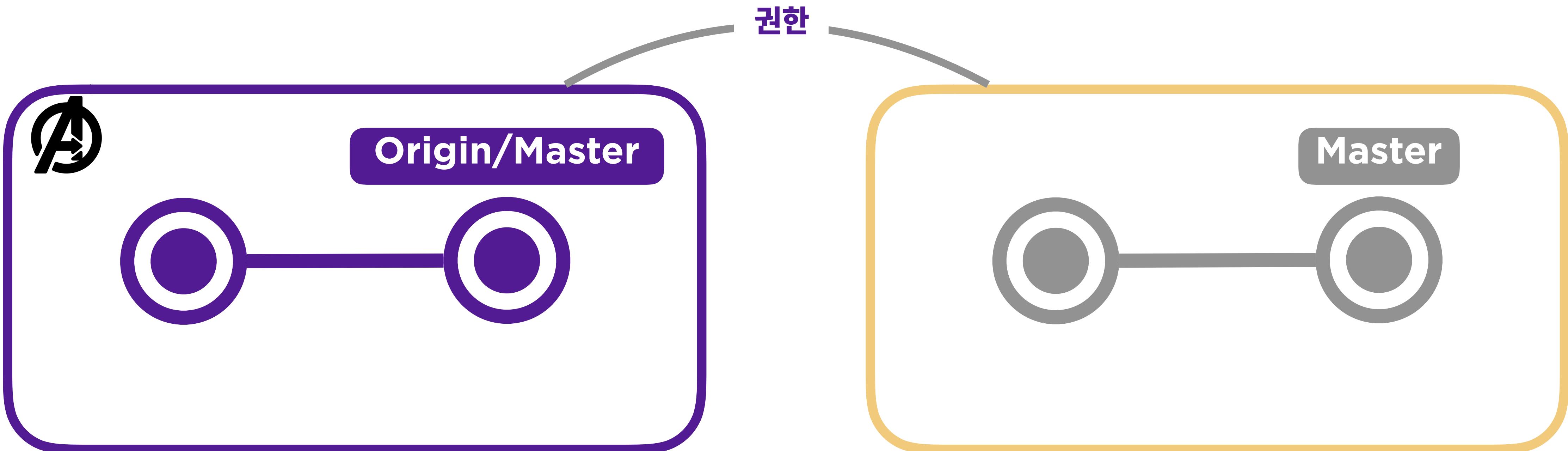


**슈퍼 개발자
아이언맨 시점**



원격 저장소 다운받기

`git clone [url]` 원격 저장소에서 다운로드



이제 저도 제 컴퓨터에서 따로 프로젝트 진행할게요.

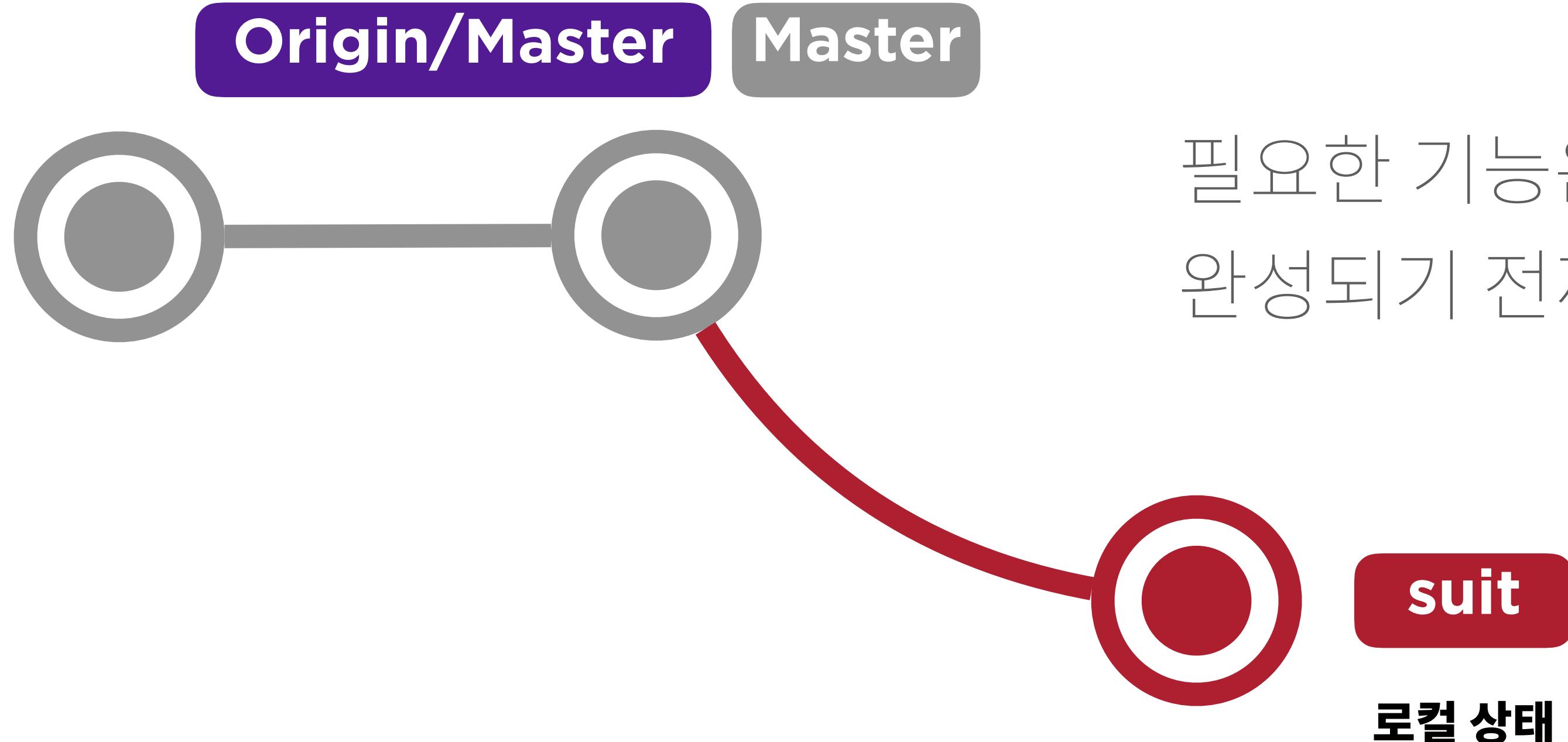
근데 **master**에 Commit을 바로 하는게 좋은걸까?

master는 배포용인데, 너무 갑작스럽지 않나?



기능별로 개발하기

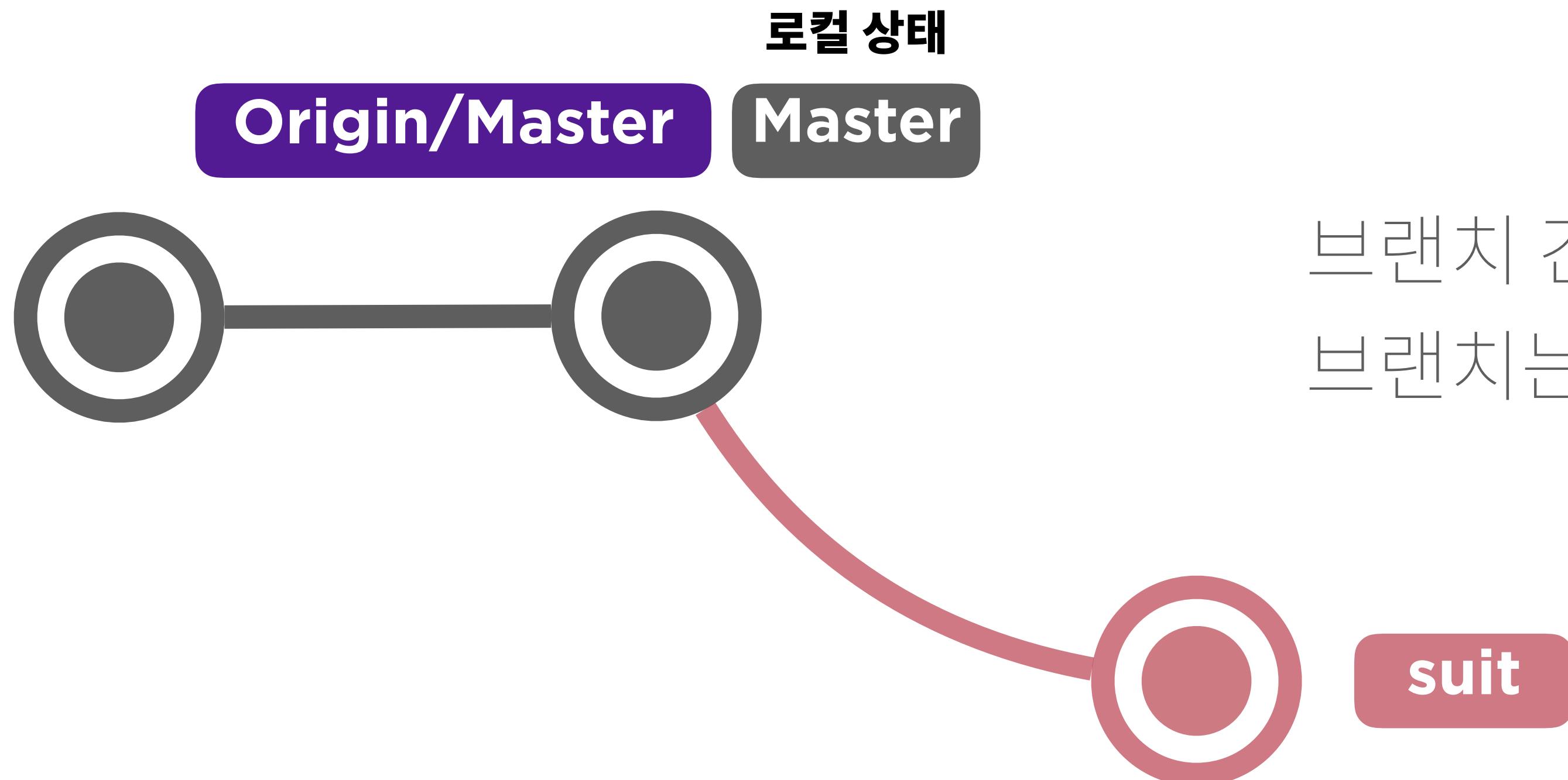
git branch [name] [name] branch 만들기



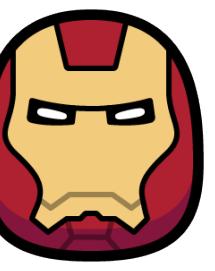


브랜치/버전 이동하기

git checkout [name] [name] branch로 이동하기

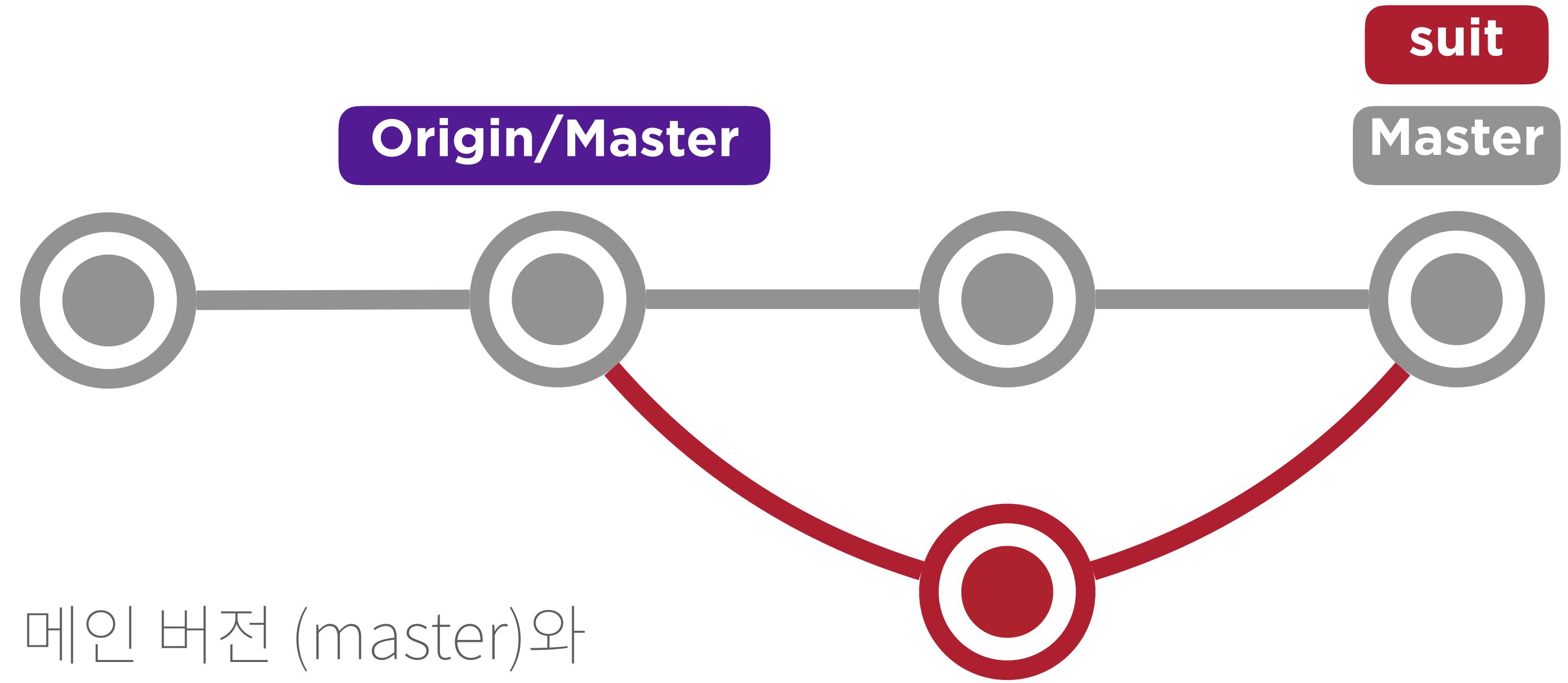


브랜치 간에도 자유롭게 이동할 수 있다.
브랜치는 결국 버전 관리를 위한 **도구**니까



브랜치 합치기

git merge [name] [name] branch를 현재 branch로 합친다

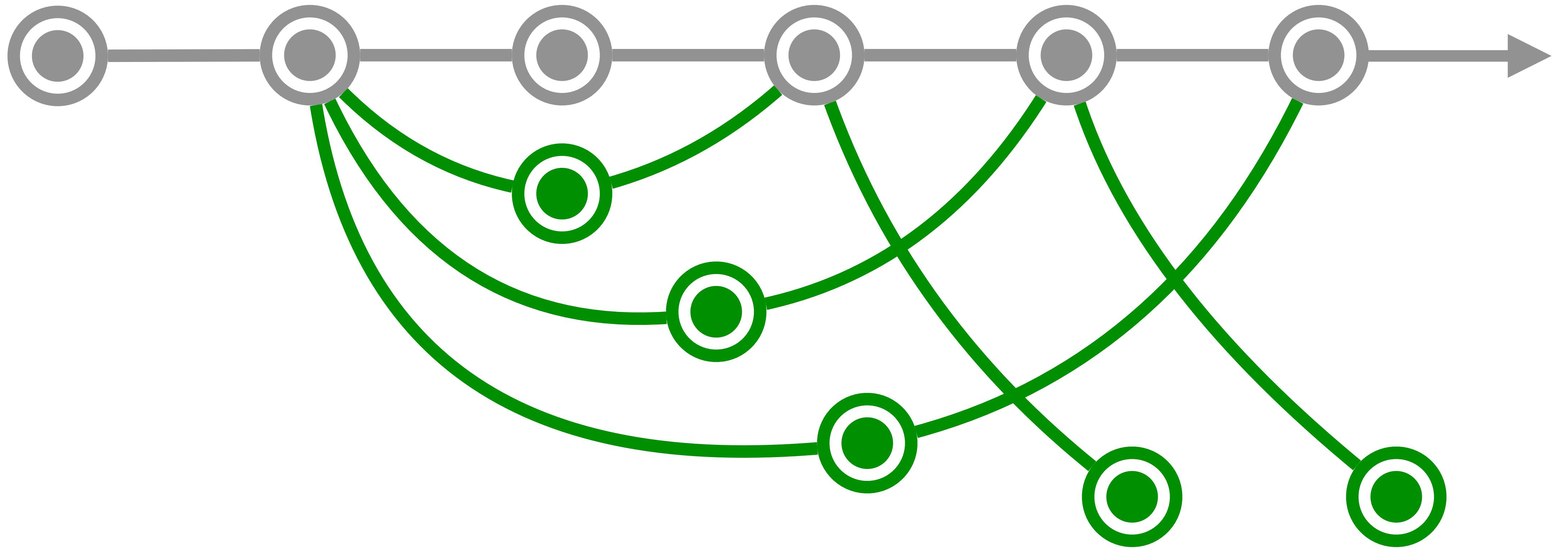


이제 가능 완성했으니 내 메인 버전 (master)와 합치고, 이후에 push하면 되겠다.



현실 Branch 마스터

닥터 스트레인지 시점

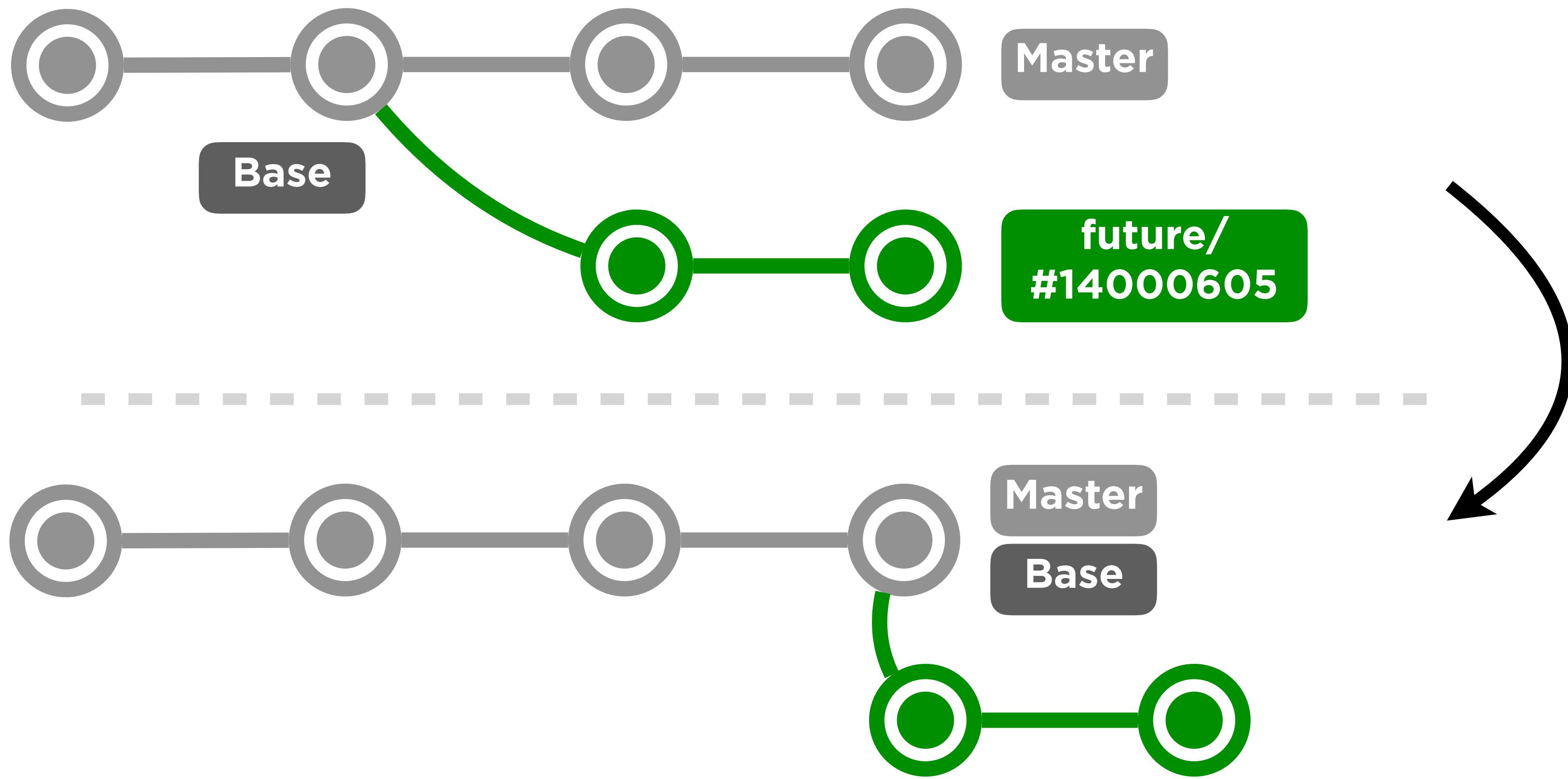


브랜치는 많아질수록 관리가 어렵다는 단점이 있네..
굳이 분기점(나눠지는) 부분이 필요할까?



브랜치 합치기 (2)

git rebase master base를 master로 re-base한다

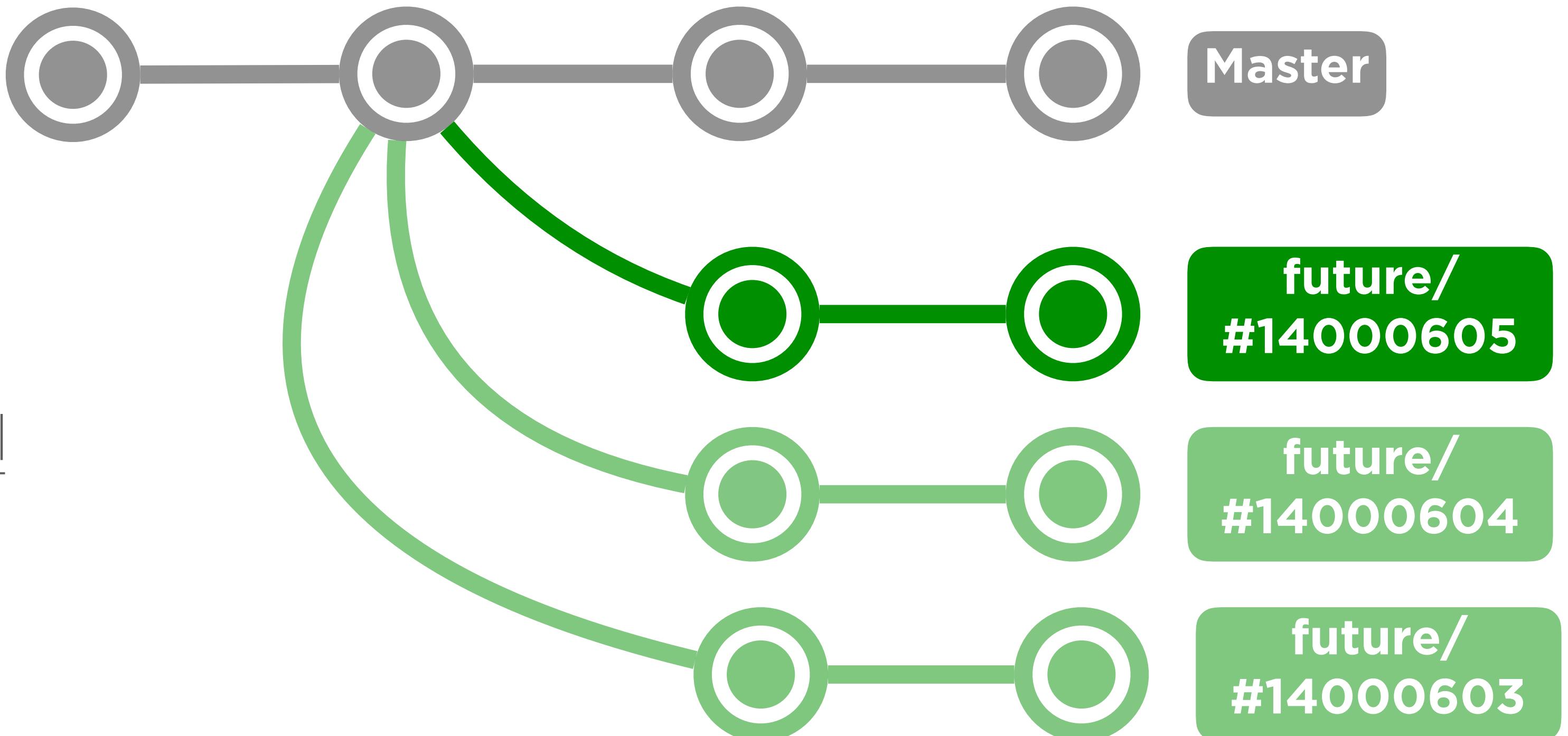


단 원격에 올린 커밋은 rebase 하지 말자



브랜치 지우기

`git branch -d [name]` 완료된 branch를 지웁니다.

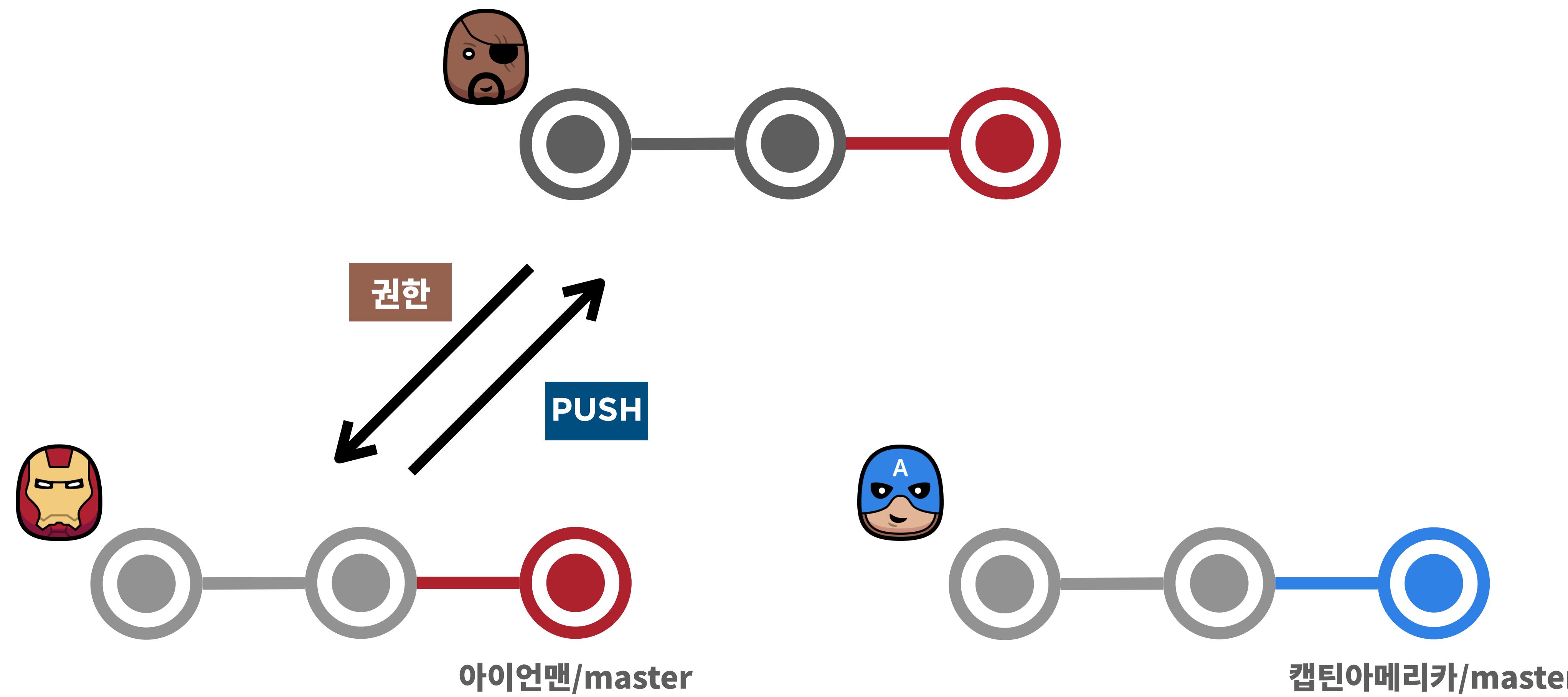


실습 TIME

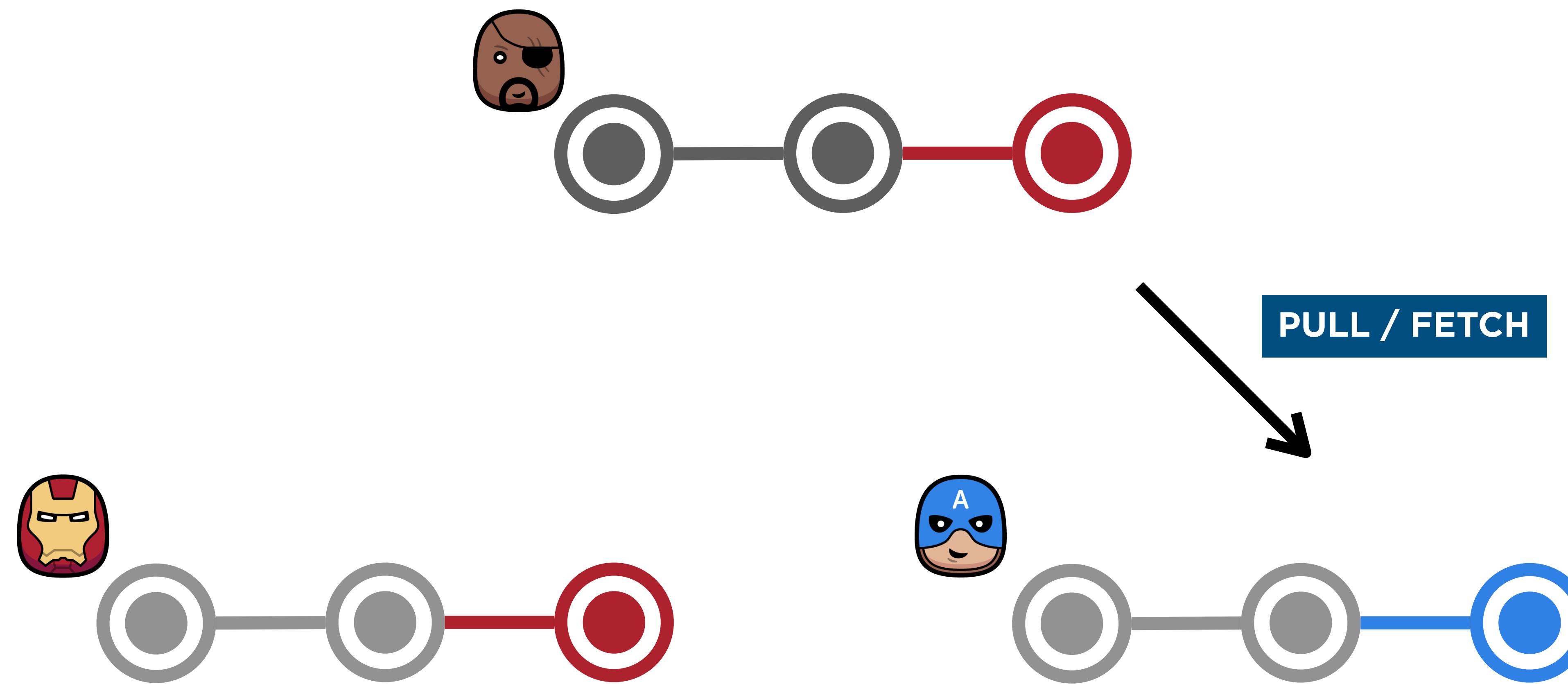
원격 저장소 다운로드 및 Branch 연습
충돌 일부러 발생시키고 해결하기



**프로젝트 리더
캡틴 아메리카 시점**



메인 버전이 **아이언맨**에 의해 업데이트 됐으니
나도 업데이트된 버전을 받아야겠다.

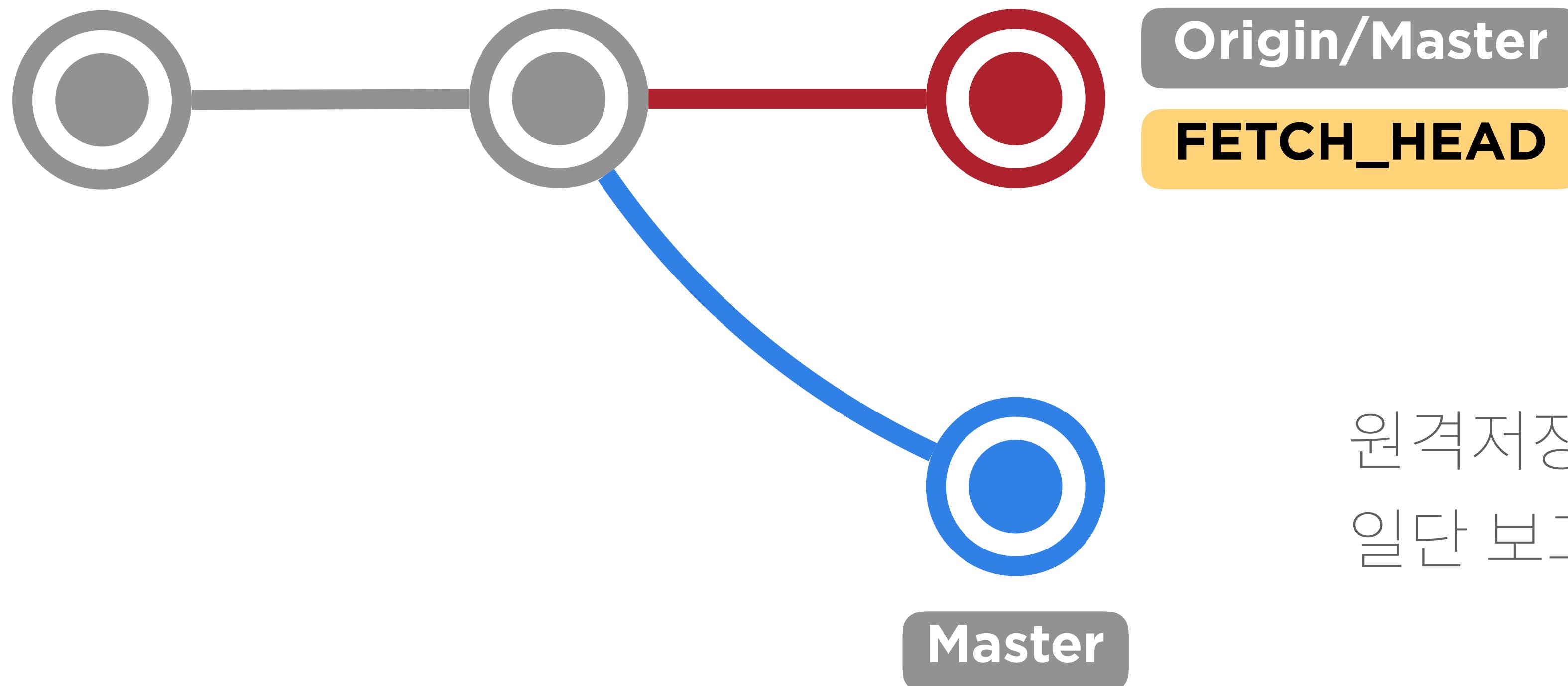


버전을 원격 저장소에서 받는 방법은 2가지입니다.
Pull 또는 Fetch입니다. 차이점을 알아봅시다.



원격에서 기록 가져오기 : Fetch

git fetch 원격 저장소와 동기화

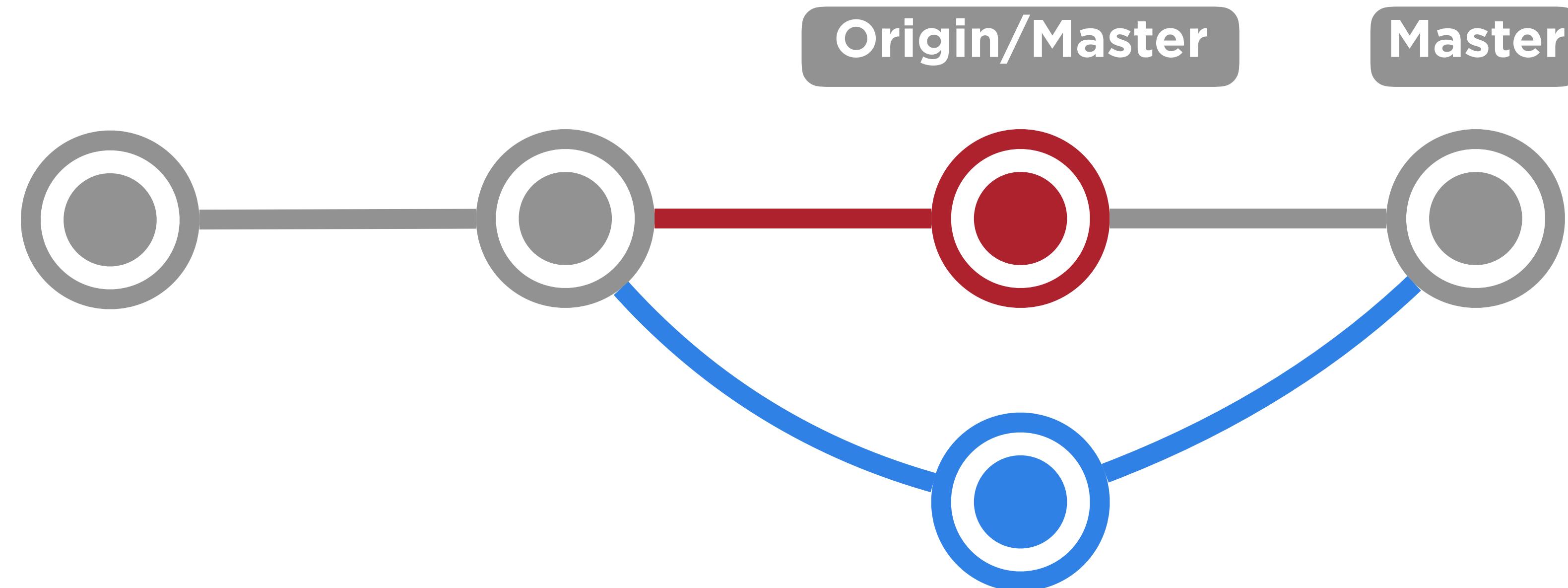


원격저장소 내용만 알고 싶다면?
일단 보고 또 생각해보자



원격에서 가져오고 합치기 : PULL

git pull 원격 저장소와 동기화하고 merge

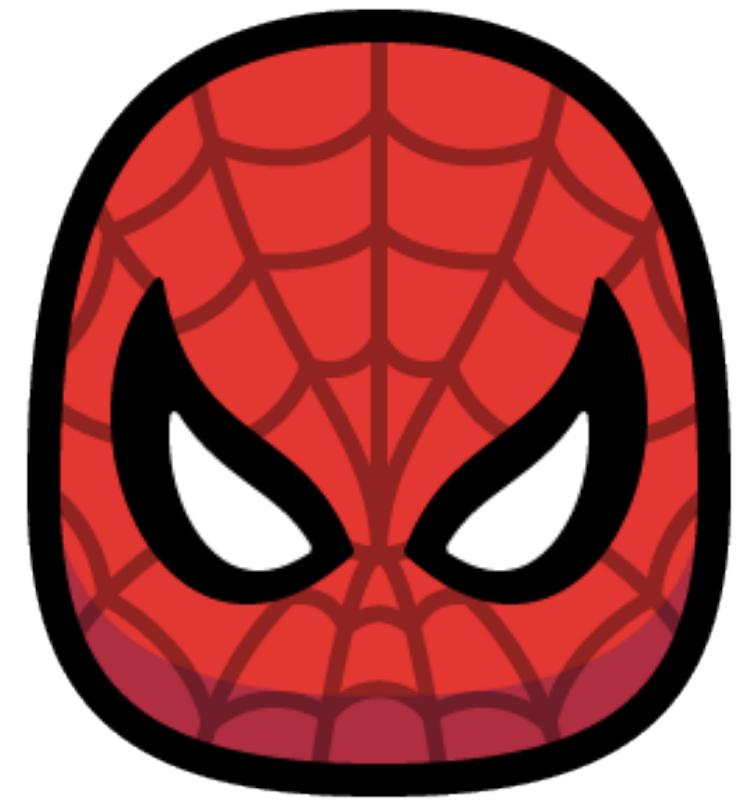


아이언맨이 한 작업은 믿고 합치자. (fetch + merge)

같은 파일의 같은 부분을 수정했다면
합칠 때…?

충돌(Conflict)

Commit 되돌리기(**reset**, **revert**), 직접 충돌 부분 수정하기
등등 다양한 해결법



**잘못하고 눈치보는
스파이더맨 시점**

실수를 하는 경우? 당연히 발생합니다.

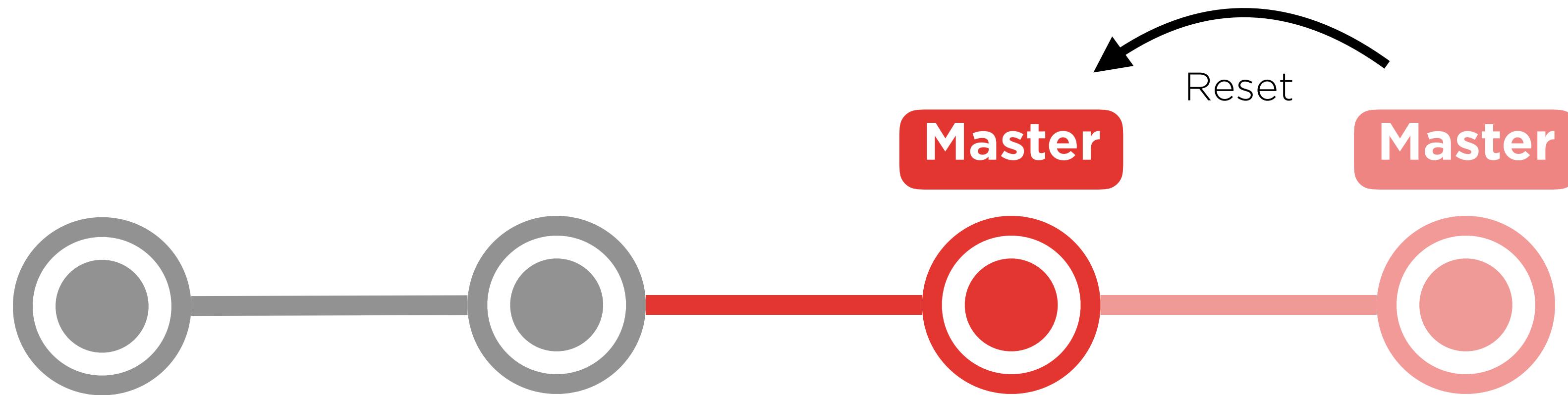
실수했다고 다 지우고, 다시 clone 하는 것보다

좋은 방법을 배워봅시다.



실수한 커밋을 RESET!

git reset [option] [branch] Branch 이후 기록을 없애자

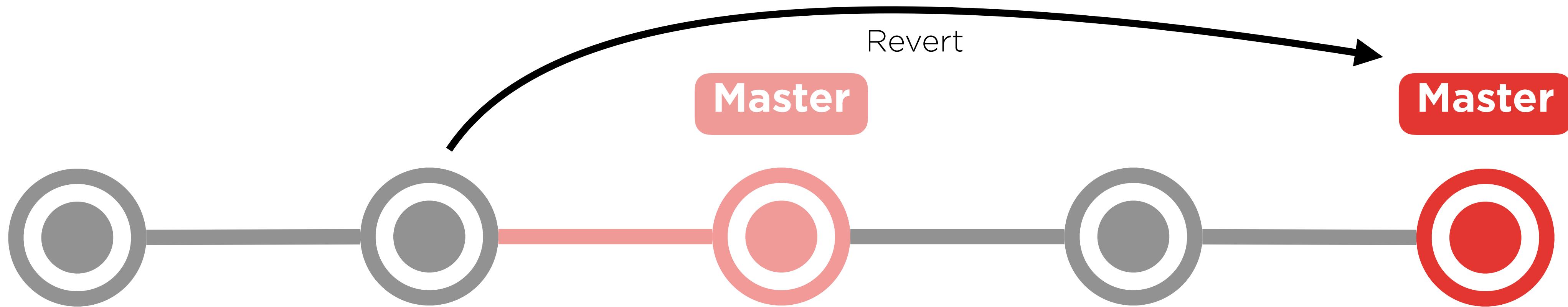


커밋으로 프로젝트가 망하면, 원하는 커밋으로 **reset**하자!
이 커밋들은 사실 다 필요 없었어! (Hard, Mixed, Soft)



실수한 커밋도 내 커밋이오.

git revert [branch] 수정한 기록도 남기자

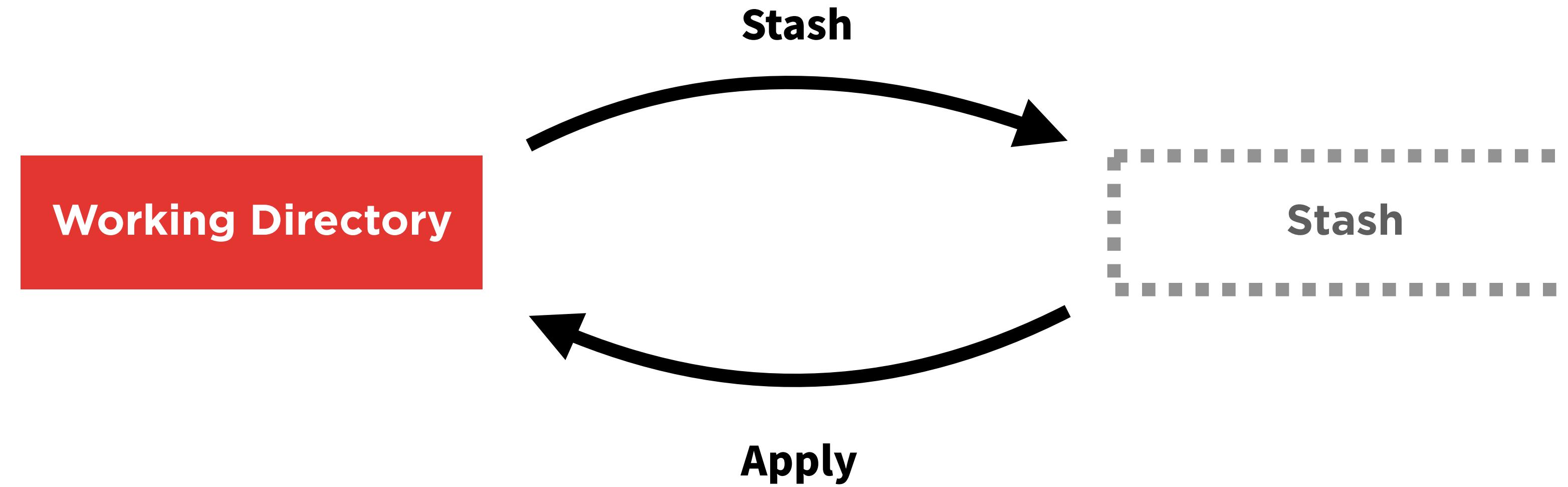


협업을 하는데, 커밋 로그를 함부로 지우면
서로 버전이 이상해질 수 있으니 revert로 수정 기록까지 남기자



Branch 바꿔야하는데 커밋이 싫다면?

git stash 현재 작업하고 있는 작업물을 따로 저장하기



아직 커밋하기는 부족한데, 빠르게 branch를 바꿔야하는 상황이라면?
따로 저장하자!

실습 TIME

저장소를 지우고 다시 받아오기

Commit 수정하기

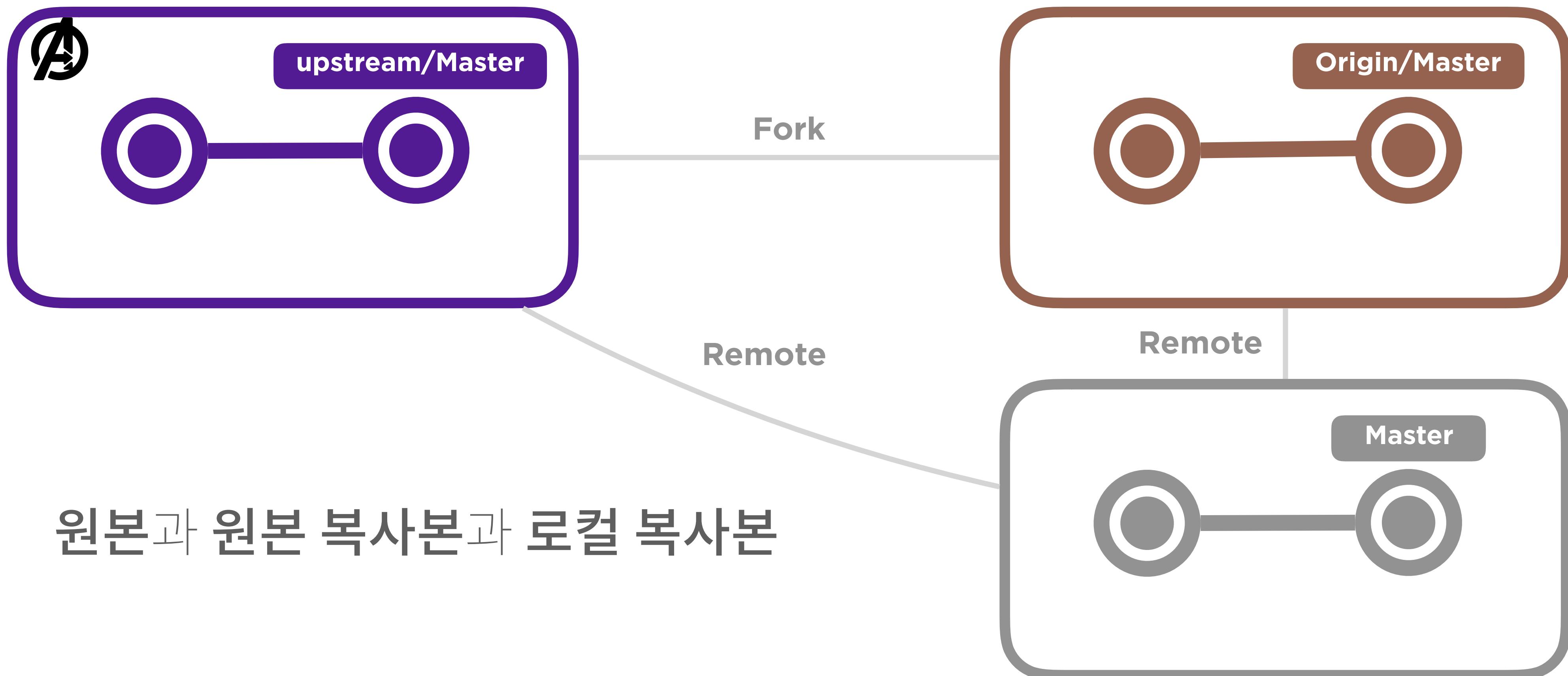


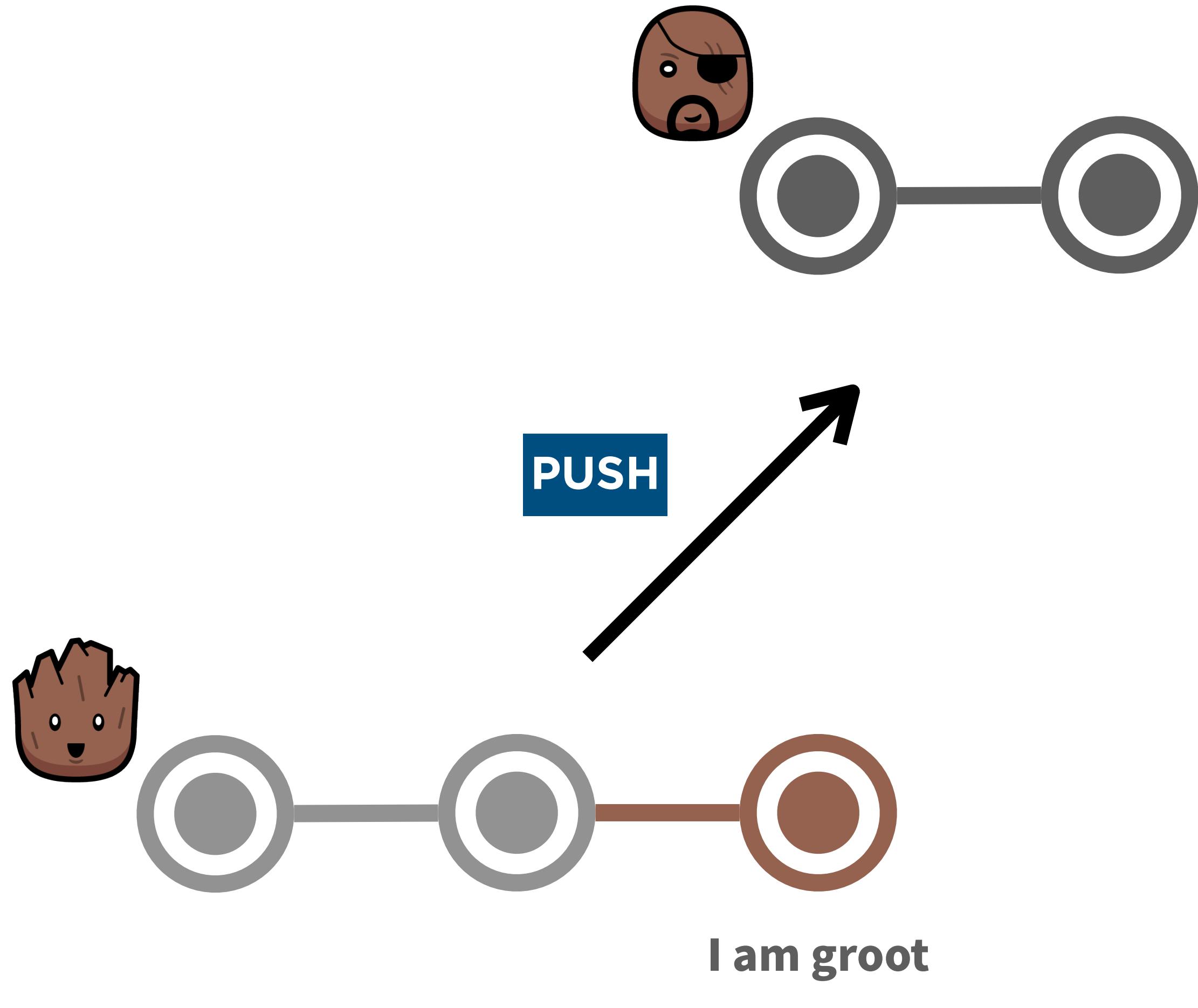
**기여하고 싶은
그루트 시점**



원격 저장소 fork 하기

따로 명령어 없음. 사이트에서 할 수 있음



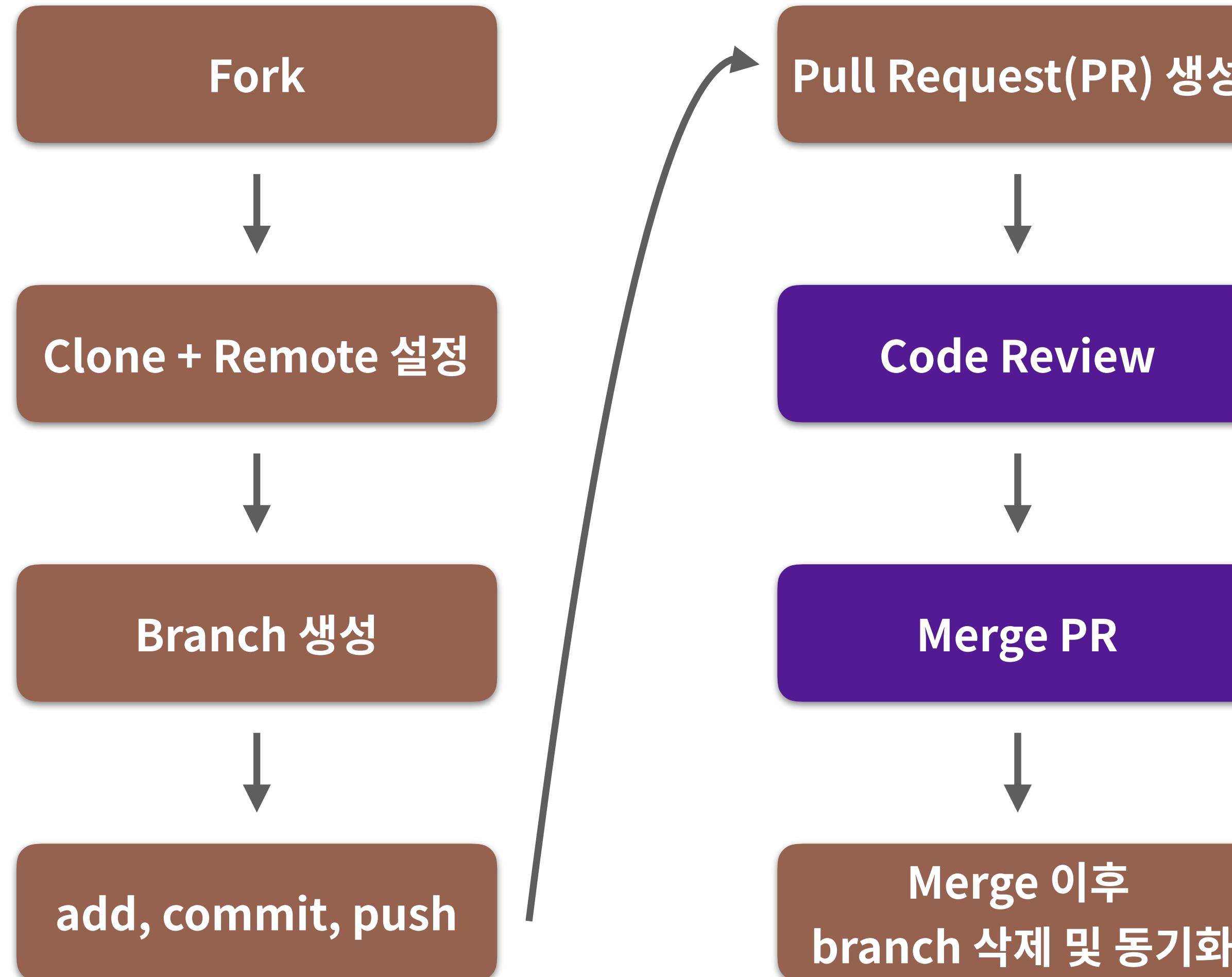


모든 사람이 쉽게 **PUSH**를 한다면?
프로젝트의 상태가 심히 걱정됩니다.

그루트를 믿을 수 있나요?



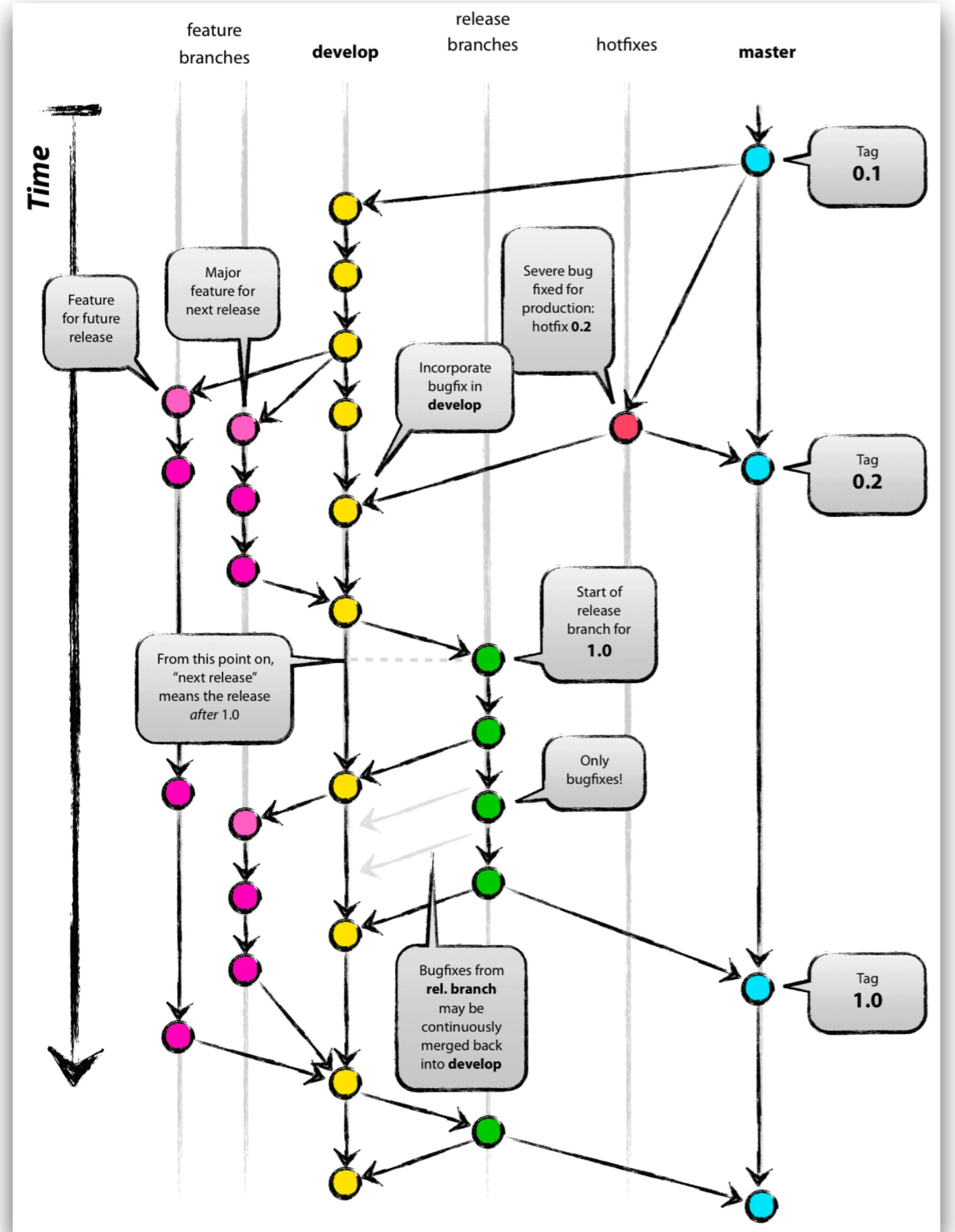
Pull Request 하는 법



Groot : I am Groot(합쳐주세요)

Avengers : 내용 확인해보고;;

Avengers : 괜찮다? 합친다 : 거절



Github Workflow

branch를 효율적으로 관리하기 위해서는?



안수빈의 블로그

오랫동안 꿈을 그리는 사람은 마침내 그 꿈을 닮아간다



An Subin

Yeah, I am Subinium.

Seoul, Korea

Email

Facebook

LinkedIn

Github

Steam

YouTube

CATEGORIES

인공지능 페이지 : AI Lookbook

알고리즘 페이지 : 알고리즘에 고통받는 최주생을 위한 안내서

개발&디자인 페이지 : 삽질하는 디발자와 개자이너

Recent Posts

[발표] 의미있는 PS를 하기 위해서..

Github Page + Jekyll로 개인 블로그를 만들고 개인이 생각하는 알고리즘을 정리하는 방법으로 앞으로 관리 방법까지!

2020 KAKAO BLIND RECRUITMENT Review

도약을 위한 작은 실패

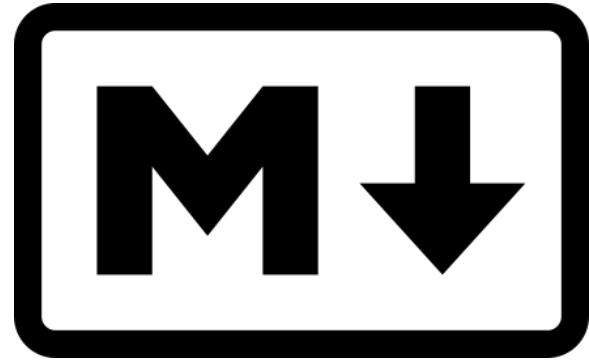
Part 2. Introduction to Ensemble Learning : Boosting

정말 오랜만에 돌아온 양상을 포스팅 2번째!

Beginner Guide : Missing Data Handling

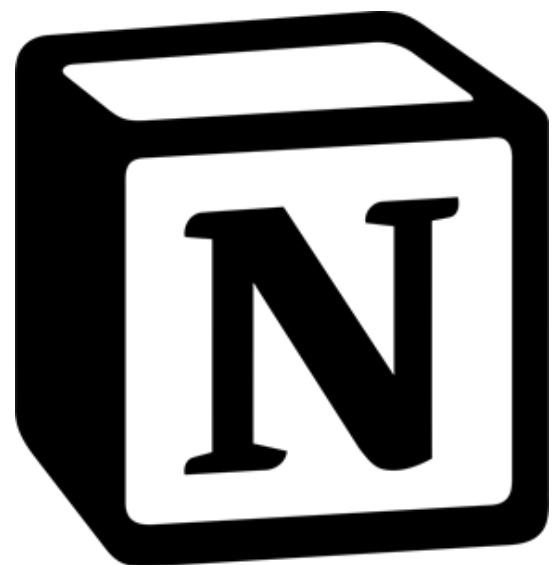
Missing Data는 어떻게 관리할 수 있을까요? Handling을 해봅시다!!

Beginner Guide : Feature Selection



Markdown 문법

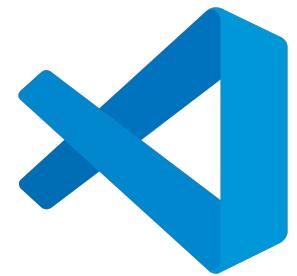
텍스트에 양식을 입히는 마크업 언어의 일종, HTML과 연동이 가능하다.



다양한 협업 툴에서 활용



다양한 Markdown용 Editor



Visual Studio Code

The screenshot shows the Visual Studio Code interface. On the left is the code editor with a file named 'git.md'. The content of the file is:

```
## 비전공생을 위한 Git과 Github Page 활용하기

컴퓨터로 협업을 하기 위해서는 git 사용이 필수적입니다.
하지만 이런 git 사용은 신규진입자에게 복잡하기만 합니다.

이번 세미나에서는 **Github Page로 블로그 만들기**라는 주제로 github에 대한 개념과 실습을 진행합니다.
필수적인 명령어, 그와 깃헙의 여러 기능을 실습해보며 본인의 루틴을 만들어봅시다.
최종적으로는 협업/개인 작업에서 어떤 방식으로 깃헙을 사용할 수 있는지 알아봅시다.

### 세부 일정
- git 기초
  - git을 사용하는 이유
  - git 개념과 github
  - github 둘러보기
  - sourcetree 사용과 연결하기
- github 실습
  - 필수 명령어
    - 알아두면 좋은 명령어
  - Github 페이지 활용하기
  - markdown 문법

### 필독 사항
- 효과적인 세미나 참여를 위해 별첨된 '세미나 가이드' 내 사전학습자료를 숙지해 주세요.
- 노트북은 개별 지참하셔야 하며, '세미나 가이드'에 따라 실습용 SW가 설치되어 있어야 합니다.

노트북 지참 및 source tree 설치
```

On the right is a preview window titled 'Preview git.md' showing the rendered content of the Markdown file.

Markdown-preview-enhanced

Title

bold, italic, Quote

List

Table

link, image

Code (Highlight)

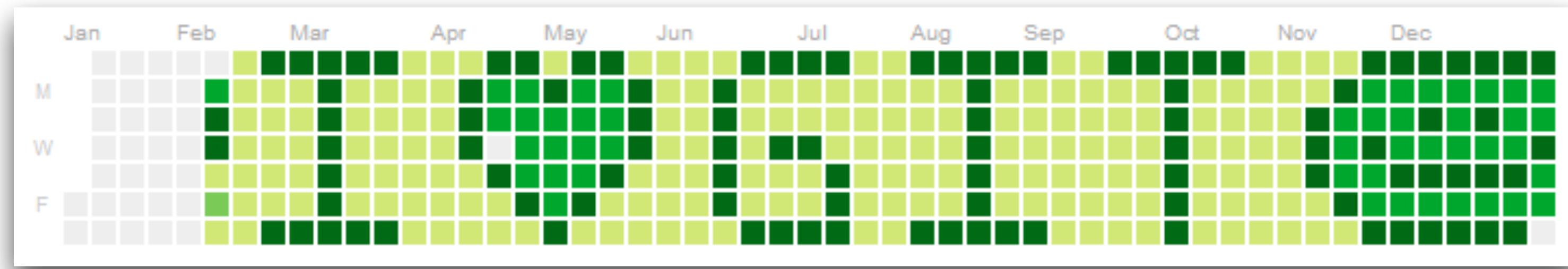
Formula (KaTeX, MathJax)

Diagram (Mermaid)

실습 TIME

Markdown의 모든 것

꾸준하게 Commit을 쌓는 방법

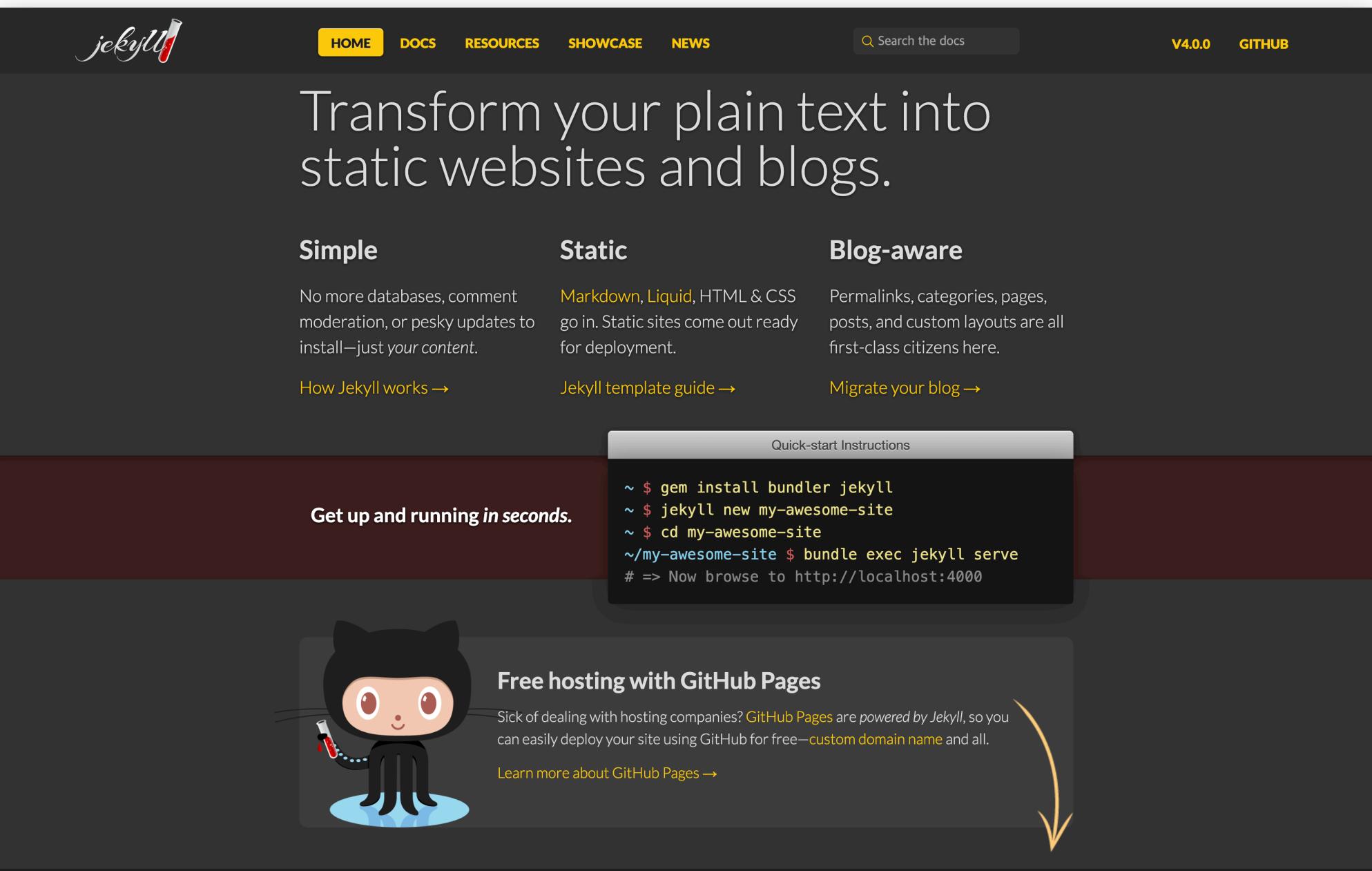


1. **TIL (Today I Learned)** : 오늘 배운 것
2. **Jekyll Blog** : 블로그를 깃헙으로 관리하기



Github 페이지에서 사용할 수 있는 사이트
Ruby 베이스지만 Ruby를 몰라도 괜찮아요

개인적으로 블로그 관리는 CLI를 추천



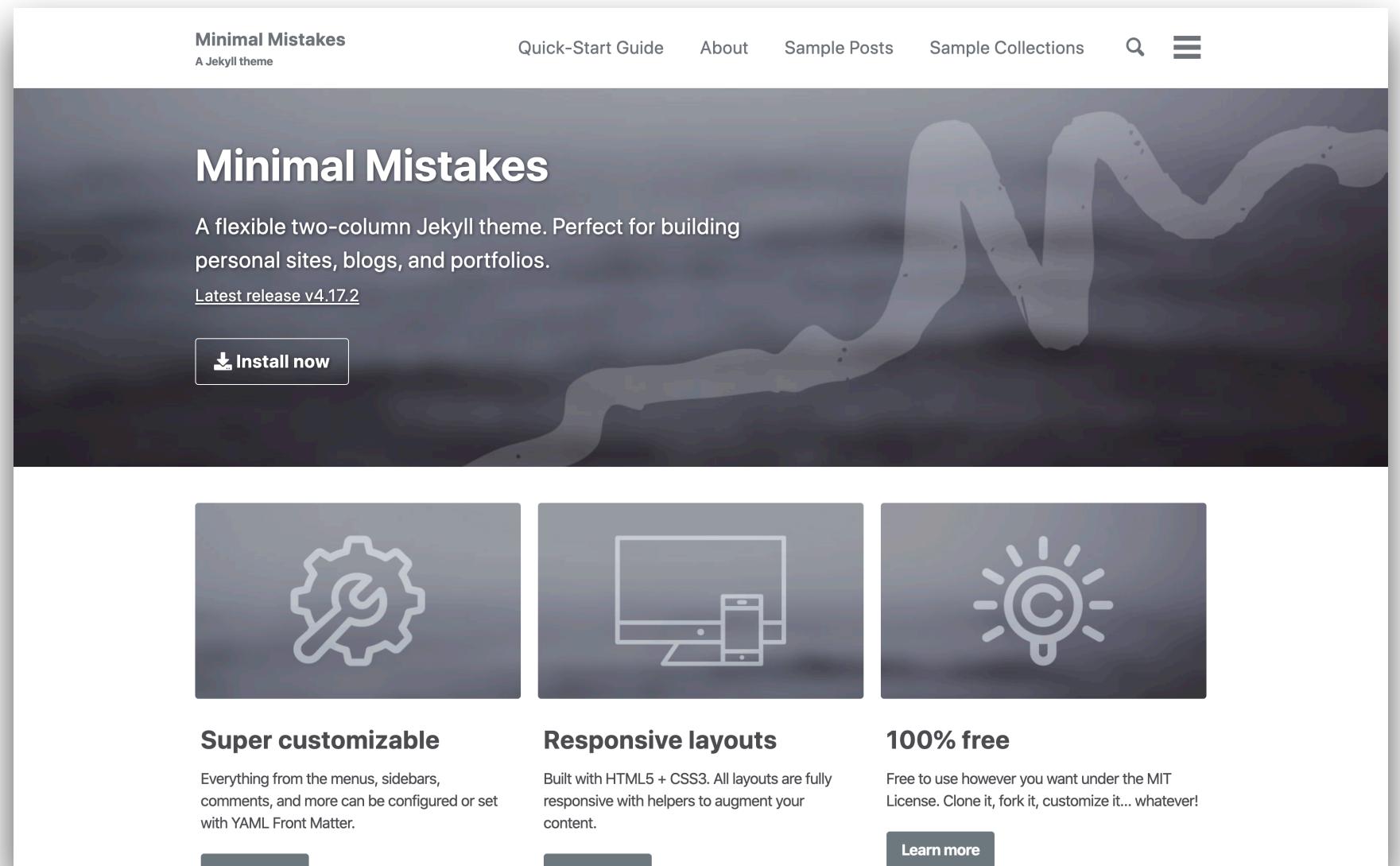
Mac, Ubuntu는 terminal, Windows는 Cmd with Ruby

<https://jekyllrb.com/docs/installation/windows/>

오늘의 블로깅

1. Theme Repo Clone / Fork
2. Jekyll 구조 살펴보기
3. 기본 setting : _config.yml
4. 첫 포스팅 연습
5. About 페이지 작성

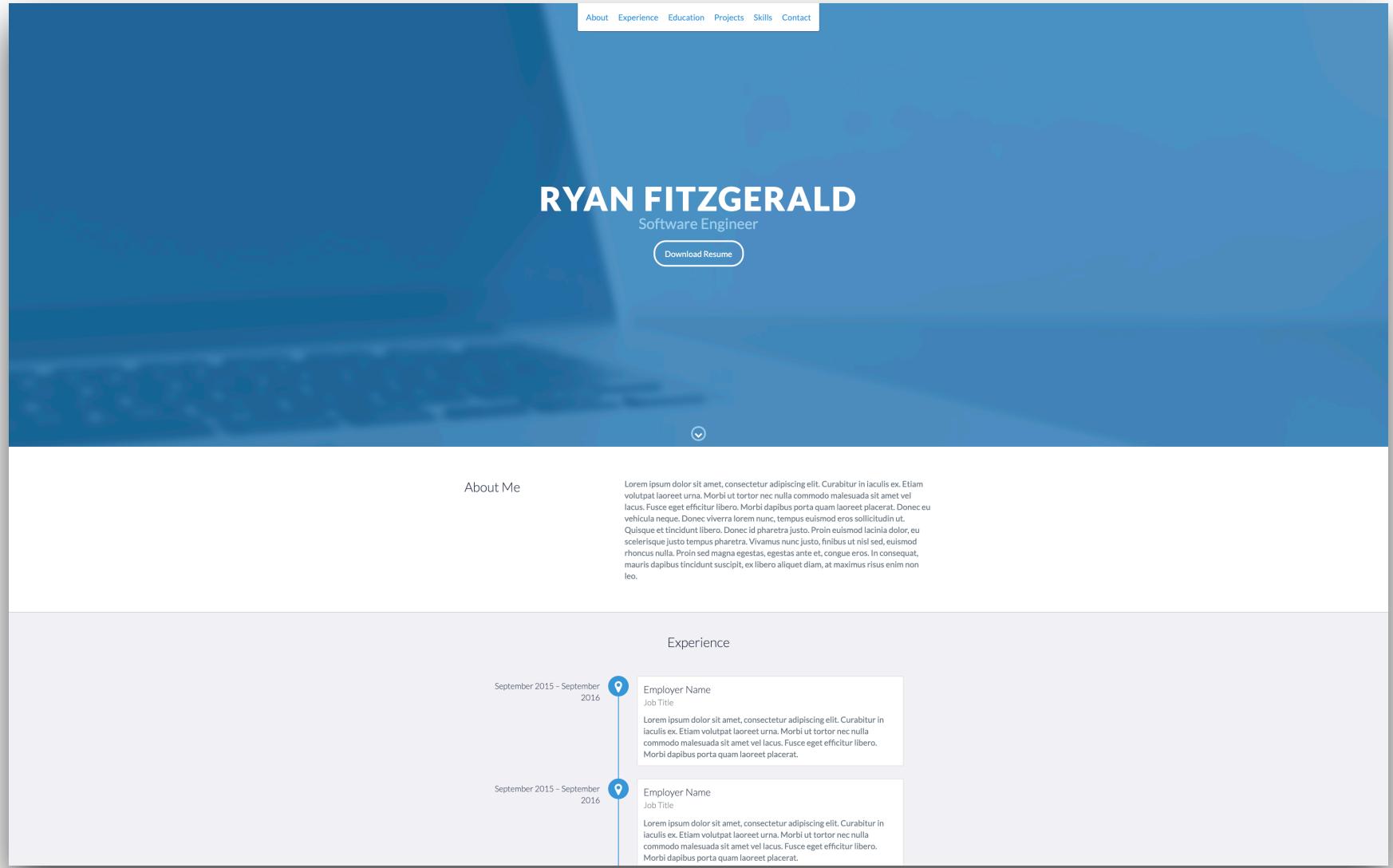




Github 블로그 인기 Theme 중 하나인
Minimal mistake

Github Page 구조 이해 및 연습하기 좋음

좋은 문서화 = Good for Custom



그 외에도 포트폴리오로 좋은 Theme

<https://github.com/RyanFitzgerald/devportfolio>

Jekyll 기본 파일들

_config.yml : 설정 파일

_posts : 블로그 포스팅

_pages : 개별 페이지

_includes : 글에 포함되는 개별 요소

_layouts : 글의 양식

assets : image, css 등

index.html : 표지

Liquid Tag

The screenshot shows the official Liquid documentation page. The left sidebar has a blue header "Liquid" and sections for "Basics", "Tags", and "Filters". The main content area starts with an "Introduction" section, which states that Liquid code can be categorized into **objects**, **tags**, and **filters**. It then moves to the "Objects" section, which explains that objects tell Liquid where to show content on a page. It shows an example of an input template with the code `{{ page.title }}` and its output being the word "Introduction". Below this, it says that in this case, Liquid is rendering the content of an object called `page.title`, and that object contains the text "Introduction". Finally, it moves to the "Tags" section, which explains that tags create logic and control flow. It shows an example of an input template with the code `{{ if user }} Hello {{ user.name }}! {{ endif }}` and its output being the text "Hello [user's name]!". The entire page has a clean, modern design with a white background and light blue accents.

<https://shopify.github.io/liquid/basics/introduction/>

Front End 개발자라면 익숙할 수도 있는 문법

MathJax를 적용하면서 가볍게 맛보기

Post Setting의 기본

Title

Category

Tag

Permalink

TOC (Table of Contents)

Sidebar

Overlay Image / Teaser

About Page (CV)

=

Portfolio, Resume

<https://sujinlee.me/professional-github/>

<http://woowabros.github.io/experience/2017/07/17/resume.html>

실습 TIME

블로그 이모저모

BLOGGING OVERVIEW

1. 왜 블야할까

쓰기 힘든 이유
그래도 써야하는 이유

3. 어떻게 블야할까

타입별 분석
세팅 : Editor + Image + Knowledge

5. 어디에 블야할까

사이트 별 간단 비교



2. 누가 쓰고, 누가 볼까

개인과 다수
타겟 설정

4. 무엇을 블야할까

시작시, 필수 내용
기술블로그 5가지 주제
피드백 분석

6. 언제 블야할까

꾸준하게 쓰는 방법

Think About

Why?

왜 글을 쓸까? Self Branding & Study



Think About

Who?

누구와 글을 쓸 것이고, 누가 글을 읽을 것인가?



Think About

How?

어떻게 쓸 것인가? 최대한 적은 노력



Think About

What?

무엇을 써야할까? 나의 성장을 위한 글



Think About

Where?

어디에 쓸 것인가? 효율적인 공유



Think About

When?

어느정도 주기로 쓸까? 우선순위



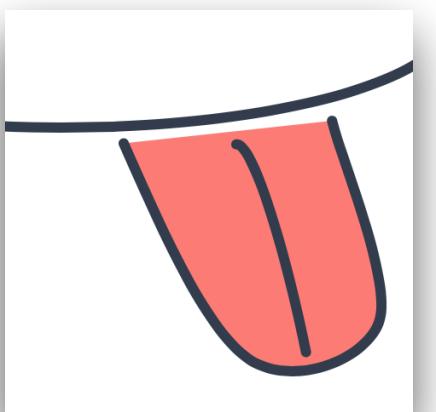
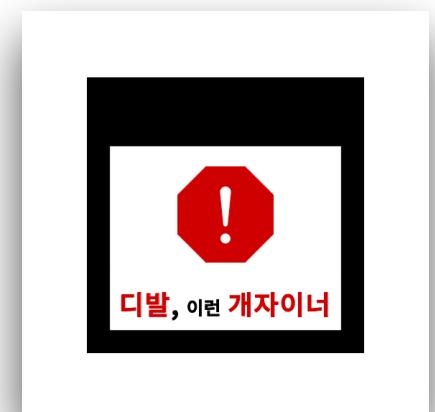
수비니움의 코딩일지 : <https://youtube.com/c/수비니움의코딩일지>

안수빈의 블로그 : <https://subinium.github.io>

A.I. Lookbook : <https://www.facebook.com/AI.Lookbook>

알고리즘으로 고통받는 취준생을 위한 안내서 : <https://www.facebook.com/algoguide/>

삽질하는 디발자와 개자이너 : <https://www.facebook.com/shovelingdesignoper/>



Thank You