Gitflow

The easy release management workflow

Git Flow란?

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

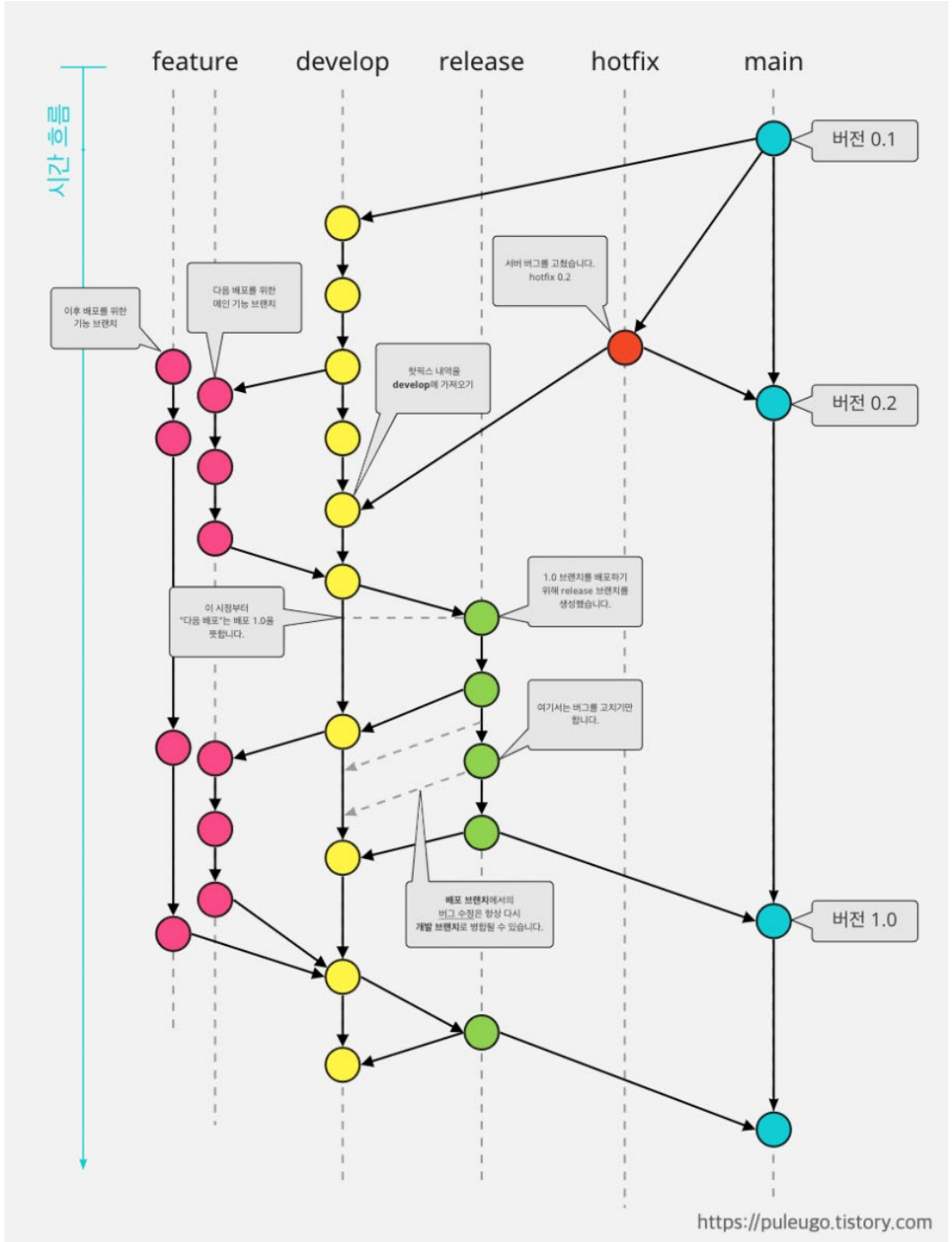
브랜치(branch)란? **브랜치**란 독립적으로 어떤 작업을 진행하기 위한 개념

브랜치 전략이란? **브랜치 전략**이란 여러 개발자가 1개의 저장소를 사용하는 환경에서 효과적으로 활용하기 위해 나온 개념

Git이 새롭게 활성화되기 시작하는 2010년에 **Vincent Driessen**이라는 사람의 블로그 글에 의해 널리 퍼지기 시작했고 현재는 **Git**으로 개발할 때 거의 표준과 같이 사용되는 **브랜치 관리 전략**

Git으로 개발 및 버전 관리할 때 여러 브랜치를 효율적으로 관리하기 위해 사용하는 **브랜치 관리 전략 (Branch management strategy)**

(간단히 말하면 '**이렇게 사용하면 브랜치 관리 하기 편하더라**'해서 사용하는 **브랜치 관리 방법론)**



Git Flow에서 사용하는 **브랜치**

master : **제품을 배포**하는 브랜치

develop : 개발자들이 **feature** 브랜치에서 개발한 수 많은 **단위 기능들이 병합되는 브랜치**

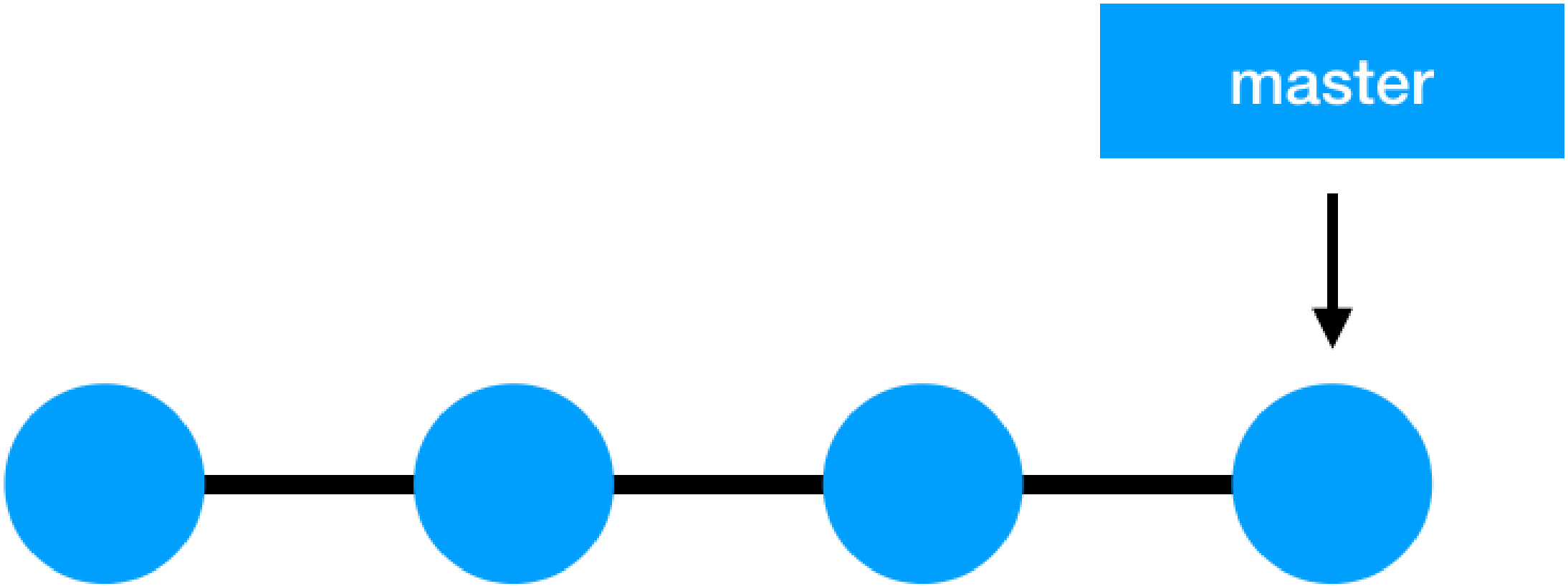
feature : **단위 기능을 개발하는 브랜치**
기능 구현이 완료되면 **develop** 브랜치에 **merge**

release : 제품 배포를 위해 **master 브랜치로 보내기 전**에 문제가 있는지 검사하고 **각종 버그를 수정**하는 브랜치
버그를 수정한 후, master 브랜치로 **merge**

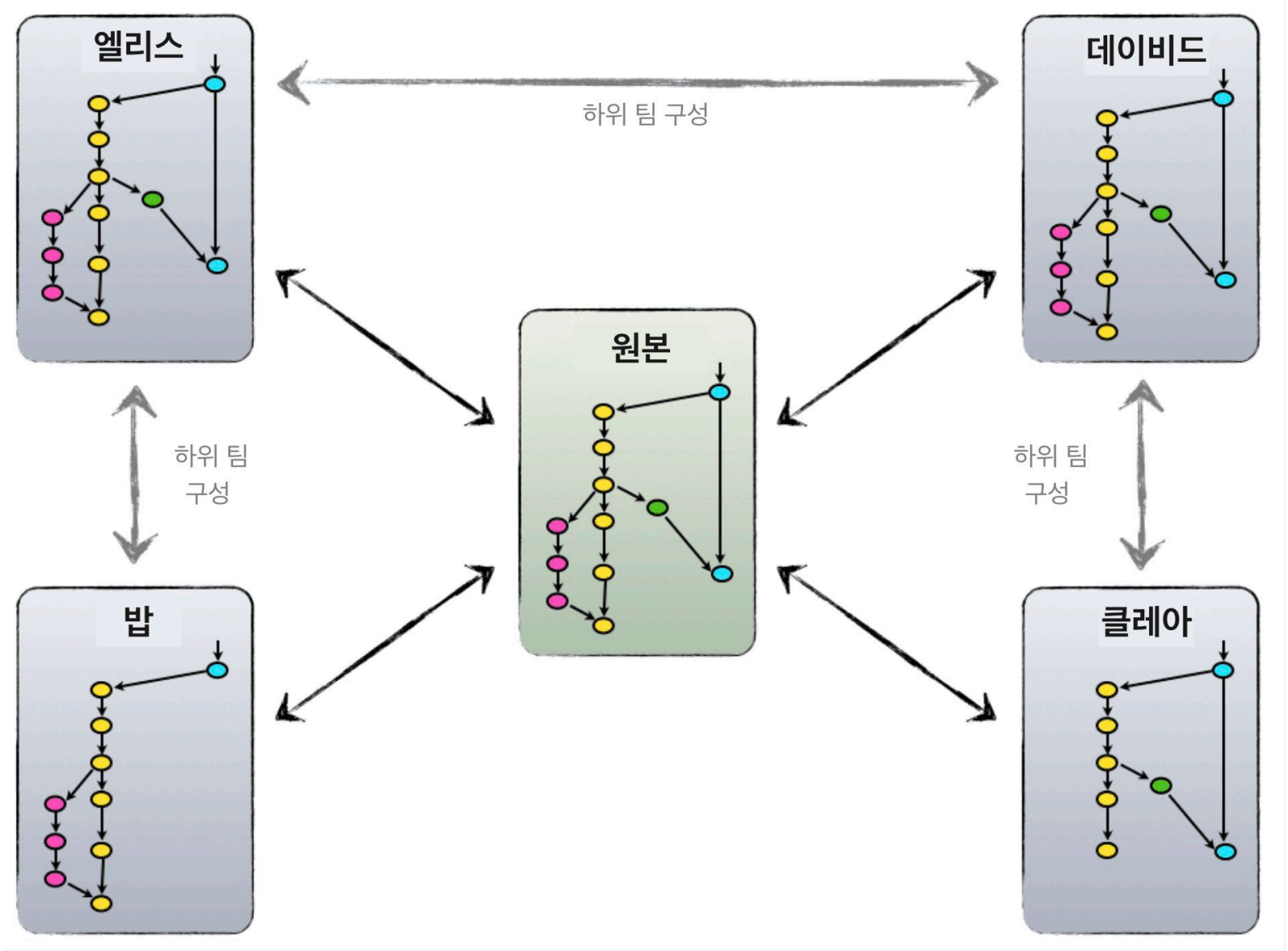
hotfix : release 브랜치에서 master 브랜치로 merge하고 **제품 배포 후**, 문제가 생겼을 때 **긴급 수정**을 위해서 사용하는 브랜치

- master, develop 브랜치는 메인 브랜치이므로 삭제되지 않고 삭제되면 안됨

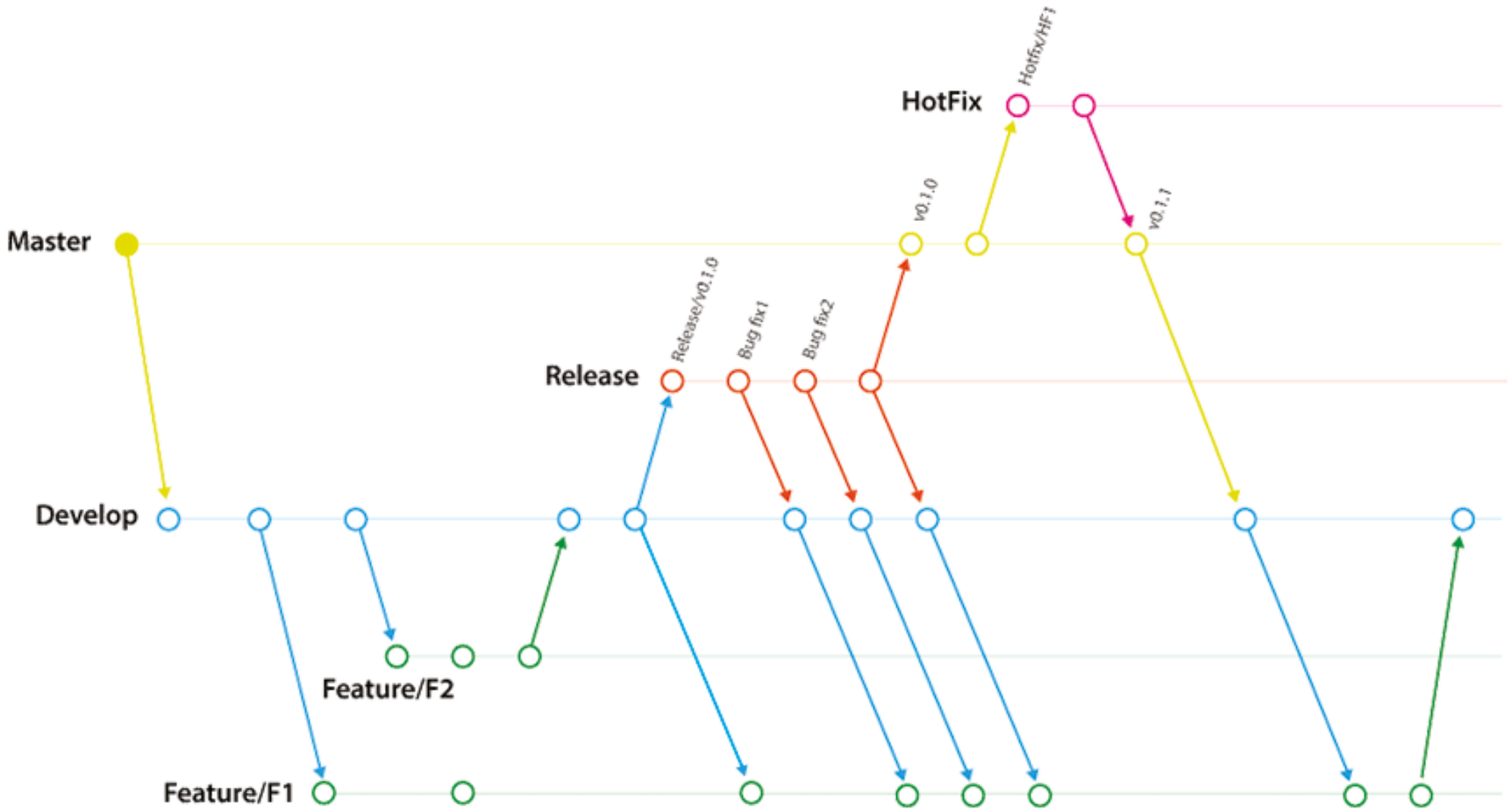
Git Flow를 쓰는 이유



Git Flow를 쓰는 이유



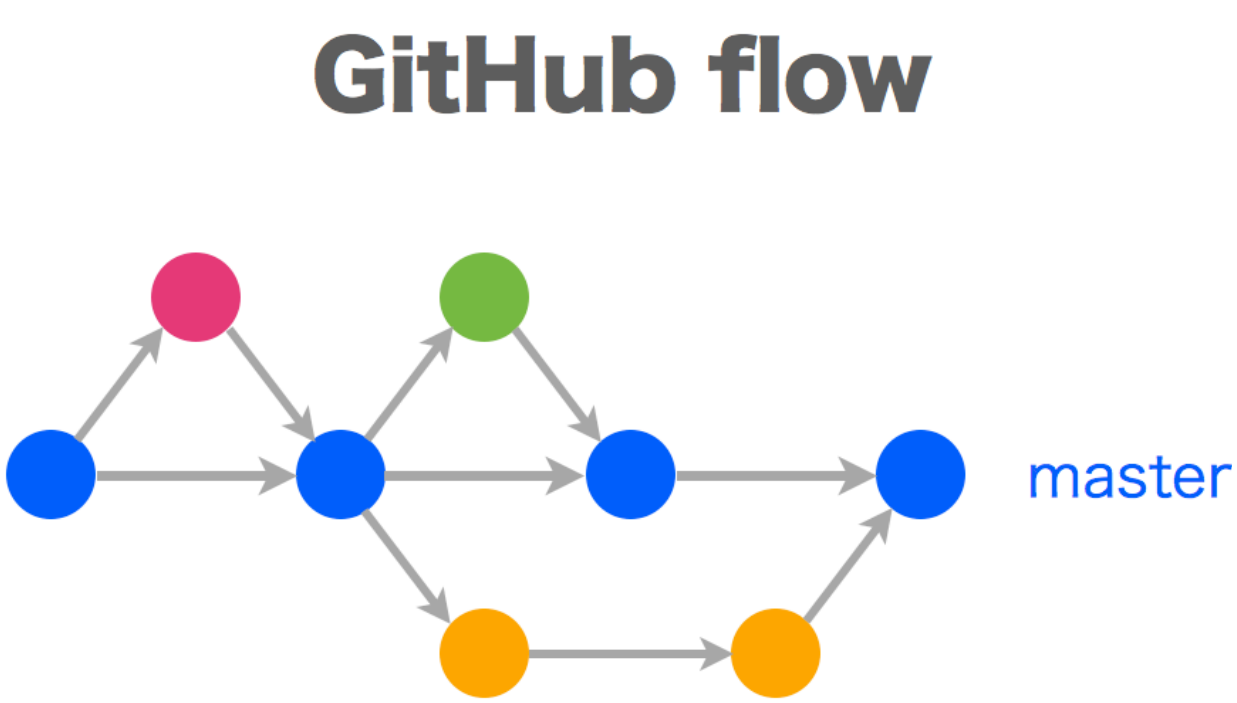
Git Flow 동작 방식

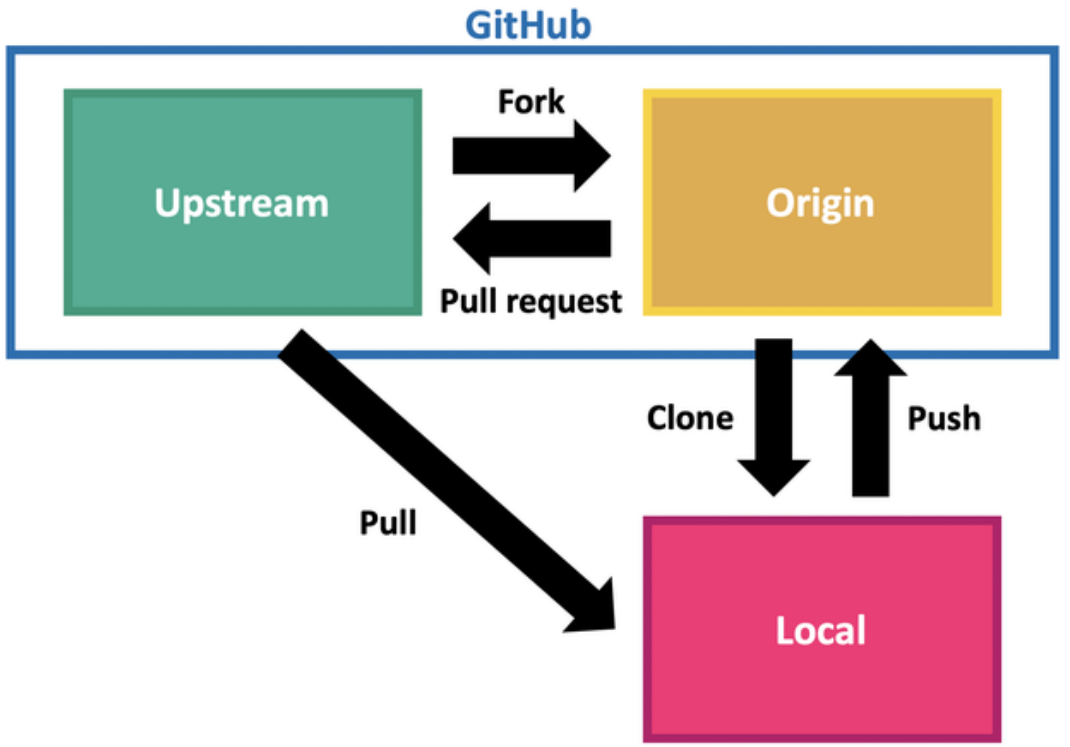


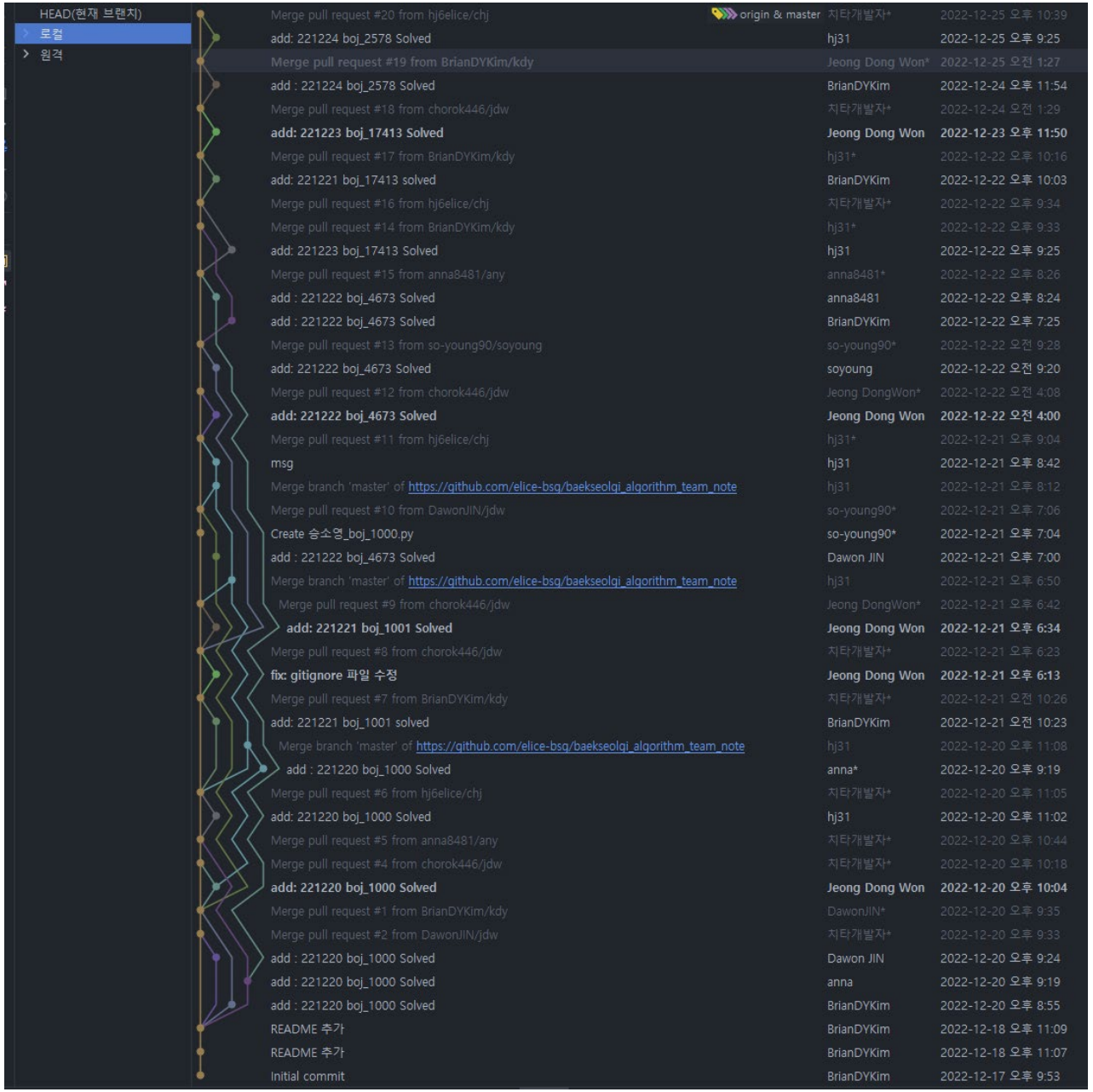
Github Flow란?

- 팀장의 저장소를 **Fork**해서 팀원마다 각자 저장소를 가지고 프로젝트를 진행하는 방식
- 팀원은 각자의 **저장소(Repository)**를 가지고 있기때문에 자유롭게 작업이 가능
- 팀원의 작업 내용은 **Pull request**를 통해 팀장의 확인 후 반영됨
- 팀장 저장소의 권한은 팀장만 가지고 있으면서 다른 사람의 커밋을 프로젝트에 적용이 가능
- 팀장이 코드를 확인하고 **merge**하기 때문에 안전하게 협업이 가능
- 오픈소스 프로젝트에서 주로 사용하는 방식

Github Flow 동작방식






HEAD(현재 브랜치)
로컬
원격
Merge pull request #20 from hj6elice/chj origin & master 치타개발자* 2022-12-25 오후 10:39
add: 221224 boj_2578 Solved hj31 2022-12-25 오후 9:25
Merge pull request #19 from BrianDYKim/kdy Jeong Dong Won* 2022-12-25 오전 1:27
add : 221224 boj_2578 Solved BrianDYKim 2022-12-24 오후 11:54
Merge pull request #18 from chorok446/jdw 치타개발자* 2022-12-24 오전 1:29
add: 221223 boj_17413 Solved Jeong Dong Won 2022-12-23 오후 11:50
Merge pull request #17 from BrianDYKim/kdy hj31* 2022-12-22 오후 10:16
add: 221221 boj_17413 solved BrianDYKim 2022-12-22 오후 10:03
Merge pull request #16 from hj6elice/chj 치타개발자* 2022-12-22 오후 9:34
Merge pull request #14 from BrianDYKim/kdy hj31* 2022-12-22 오후 9:33
add: 221223 boj_17413 Solved hj31 2022-12-22 오후 9:25
Merge pull request #15 from anna8481/any anna8481* 2022-12-22 오후 8:26
add : 221222 boj_4673 Solved anna8481 2022-12-22 오후 8:24
add : 221222 boj_4673 Solved BrianDYKim 2022-12-22 오후 7:25
Merge pull request #13 from so-young90/soyoung so-young90* 2022-12-22 오전 9:28
add: 221222 boj_4673 Solved soyoung 2022-12-22 오전 9:20
Merge pull request #12 from chorok446/jdw Jeong DongWon* 2022-12-22 오전 4:08
add: 221222 boj_4673 Solved Jeong Dong Won 2022-12-22 오전 4:00
Merge pull request #11 from hj6elice/chj hj31* 2022-12-21 오후 9:04
msg hj31 2022-12-21 오후 8:42
Merge branch 'master' of https://github.com/elice-bsq/baekseolgi_algorithm_team_note hj31 2022-12-21 오후 8:12
Merge pull request #10 from DawonJIN/jdw so-young90* 2022-12-21 오후 7:06
Create 승소영_boj_1000.py so-young90* 2022-12-21 오후 7:04
add : 221222 boj_4673 Solved Dawon JIN 2022-12-21 오후 7:00
Merge branch 'master' of https://github.com/elice-bsq/baekseolgi_algorithm_team_note hj31 2022-12-21 오후 6:50
Merge pull request #9 from chorok446/jdw Jeong DongWon* 2022-12-21 오후 6:42
add: 221221 boj_1001 Solved Jeong Dong Won 2022-12-21 오후 6:34
Merge pull request #8 from chorok446/jdw 치타개발자* 2022-12-21 오후 6:23
fix: gitignore 파일 수정 Jeong Dong Won 2022-12-21 오후 6:13
Merge pull request #7 from BrianDYKim/kdy 치타개발자* 2022-12-21 오전 10:26
add: 221221 boj_1001 solved BrianDYKim 2022-12-21 오전 10:23
Merge branch 'master' of https://github.com/elice-bsq/baekseolgi_algorithm_team_note hj31 2022-12-20 오후 11:08
add : 221220 boj_1000 Solved anna* 2022-12-20 오후 9:19
Merge pull request #6 from hj6elice/chj 치타개발자* 2022-12-20 오후 11:05
add: 221220 boj_1000 Solved hj31 2022-12-20 오후 11:02
Merge pull request #5 from anna8481/any 치타개발자* 2022-12-20 오후 10:44
Merge pull request #4 from chorok446/jdw 치타개발자* 2022-12-20 오후 10:18
add: 221220 boj_1000 Solved Jeong Dong Won 2022-12-20 오후 10:04
Merge pull request #1 from BrianDYKim/kdy DawonJIN* 2022-12-20 오후 9:35
Merge pull request #2 from DawonJIN/jdw 치타개발자* 2022-12-20 오후 9:33
add : 221220 boj_1000 Solved Dawon JIN 2022-12-20 오후 9:24
add : 221220 boj_1000 Solved anna 2022-12-20 오후 9:19
add : 221220 boj_1000 Solved BrianDYKim 2022-12-20 오후 8:55
README 추가 BrianDYKim 2022-12-18 오후 11:09
README 추가 BrianDYKim 2022-12-18 오후 11:07
Initial commit BrianDYKim 2022-12-17 오후 9:53

Github Flow 사용방법

1. **master** 브랜치는 어떤 때든 **배포(Depoloy)**가 가능한 상태를 유지해야함
2. **master** 에서 새로운 작업을 시작하기 위해 브랜치를 만든다면, 이름을 명확히 작성해야 함.
3. 원격지 브랜치로 수시로 **push**
4. 피드백이나 도움이 필요할 때, 그리고 **merge** 준비가 완료되었을 때는 **pull request**를 생성한다
5. 기능에 대한 리뷰와 논의가 끝난 후 **master**로 **merge**
6. **master**로 **merge**되고 **push** 되었을 때는, 즉시 **배포 (Depoloy)** 되어야 함

브랜치 관리전략에 정답은 없다

우아한형제들 기술 블로그

우린 Git-flow를 사용하고 있어요

Oct 30, 2017 • 나동호 

안녕하세요. 우아한형제들 배민프론트개발팀에서 안드로이드 앱 개발을 하고 있는 나동호입니다.
오늘은 저희 안드로이드 파트에서 사용하고 있는 Git 브랜치 전략을 소개하려고 합니다. '배달의민족 안드로이드 모바일 파트에서 이렇게 브랜치를 관리하고 있구나' 정도로 봐주시면 좋을 것 같습니다.

들어가며...

2016년 1월, Github로 소스코드를 이전하면서 Github-flow를 사용하기 시작했습니다. 그러다 2017년 6월부터 Git-flow로 브랜치 전략을 바꾸게 되었습니다. 오늘 할 이야기는 저희가 브랜치 전략을 왜 바꾸게 되었는지 그리고 어떻게 브랜치를 관리하고 있는지 이야기를 하려고 합니다.

Git-flow를 사용하게된 이유

기존의 배달의민족 앱의 개발 프로세스는 '기획-디자인-개발-QA-출시' 순서의 흐름으로 차근차근 흘러갔고 3주 주기마다 앱을 출시했습니다. 그 일을 하는 안드로이드 앱 개발자 인원은 보통 2~3명이였습니다.

회사에서는 지속적으로 개발자를 채용했고 어느새 배달의민족 안드로이드 앱 개발자가 5명으로 늘어났습니다. 기획, 디자인, 서버 등 많은 사람들과 흐름을 맞춰야하기 때문에 이 5명이 모두 이번 버전에 포함될 기능을 개발하는 것은 비효율적이였습니다. (무엇보다 iOS 개발자가 부족합니다..) 작업에 따라 개발 기간이 3주 이상이 필요한 작업들이 많아지기 시작한 이유도 있었습니다. 그래서 기존의 개발 프로세스에서 약간의 변화가 생겼습니다.

브랜치 관리 전략 선정기준

- 프로젝트에 맞게
- 팀원과의 협의후에

Reference

- https://ux.stories.pe.kr/183
- https://nvie.com/posts/a-successful-git-branching-model/
- https://groovypark.github.io/2018/01/02/git-flow-사용하기/
- https://overcome-the-limits.tistory.com/entry/협업-협업을-위한-Git-Flow-설정하기
- https://hellowoori.tistory.com/56
- https://velog.io/@hyowon_lee/Git-GitHub로-협업하기-Forking-Workflow
- https://backlog.com/git-tutorial/kr/stepup/stepup1_1.html
- https://puleugo.tistory.com/107