Git Flow

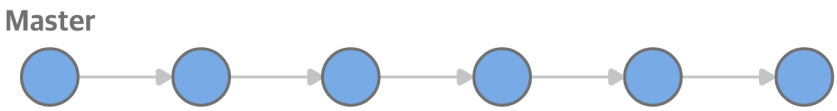
A successful Git branching model *by Vincent Driessen*

허가받지 않은 복제(복사), 전송, 수정 및 배포를 금합니다.

학습 조건

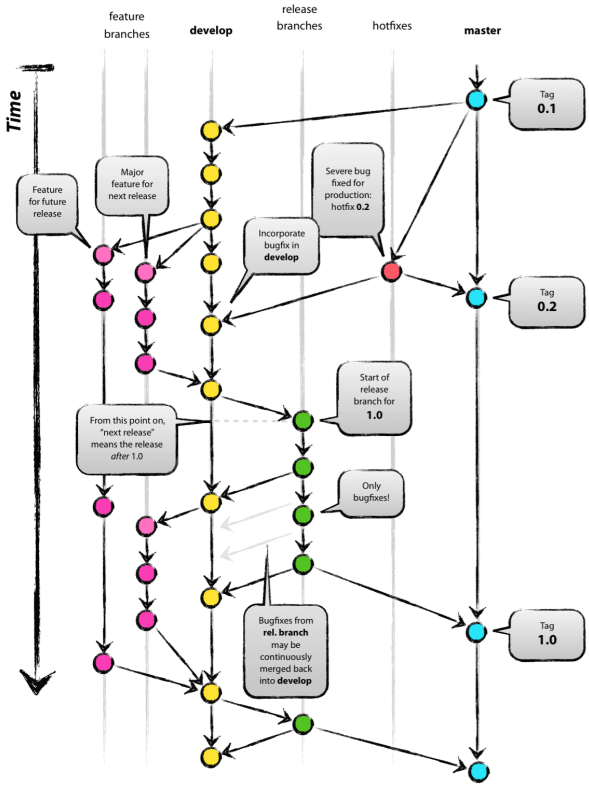
- Git 개념을 알고 있다
- Github 사용 경험이 있다
  - Push
  - Pull Request

No Flow



- 개인 프로젝트
- 소규모 프로젝트

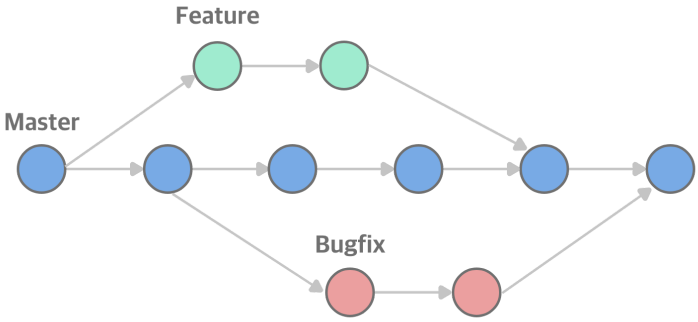
Git Flow



Github Flow

By Scott Chacon

Git Flow는 Github 에서 사용하기에는 복잡하다



특징

- **anything in the master branch is deployable**
  - 전제: 마스터 브랜치는 언제든 어떤 상황이든 배포 가능
- **create descriptive branches off of master**
  - `user-content-cache-key` `submodules-init-task` `redis2-transition`
- **push to named branches constantly**
  - 팀원의 작업 현황을 확인 할 수 있음
  - 백업 장점
- **open a pull request at any time**
  - 코드 리뷰 문화, 커뮤니케이션
- **merge only after pull request review**
  - 머지 후에 바로 배포

요약

- 매우 단순한 브랜치 구조
- 코드 리뷰 문화
- CI / CD 배포 자동화 필요
  - Github Actions 활용 추천

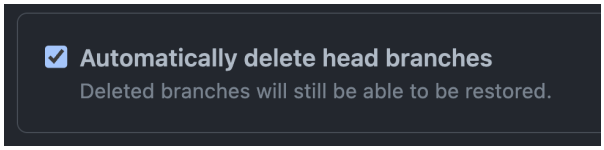
브랜치명 커스텀

- master 👉 main
- develop 👉 dev
- feature 👉 feat
  - 본인 닉네임 활용. `dev-it/post-list`
  - 기능 설명을 한글로도 활용. `feat/게시글-목록`
- release
- hotfix

Pull Request merge 후에 브랜치 삭제

Github 👉 Repository 👉 Settings 👉 General

> After pull requests are merged, you can have head branches deleted automatically.



해당 feature 나오기 전에는 Github App **Delete merged branch** 사용.

Github App (Zenhub, Jira, Slack, Sentry …)

로컬 Feature 브랜치 삭제

아래 명령어 활용

```
git fetch -p && for branch in $(git branch -vv | grep ': gone]' | awk '{print $1}'); do git branch -D $branch; done
```

> To conclude, always remember that panaceas don't exist.
> Consider your own context. Don't be hating. Decide for yourself.
>
> *by Vincent Driessen.*
> **March 5, 2020**

✉️ dev.it.nm@gmail.com