

Git-flow 도입의 회고

부제 : Git-flow 로 협업하고, 코드리뷰하기

김기현

회고 내용

설명하지 않을 내용

- Git의 기본기
 - Git의 30% 지식이면 충분
- 코드리뷰의 필요성

*회고에서 설명하지 않는 모든 것은 개인적으로 말씀해주시면 같이 알아가겠습니다.

[회고 스트립트 전문 \(링크\)](#)

Reference (link)

- ❑ [GitLab Documentation](#)
- ❑ [우아한형제들 기술블로그 : 우린 Git-flow를 사용하고 있어요](#)
- ❑ [당근마켓 팀블로그 : 매일 배포하는 팀이 되는 여정\(1\) — 브랜치 전략 개선하기](#)

설명할 내용

1.어떻게 일했는가

- Git-flow 이전 이후의 업무방식

2.Git-flow 소개

3.Git-flow로 일해보기

1.개발 협의 완료

2.Milestones 생성

3.Branch checkout

4.Issue 생성

5.Merge Request & Code Review

6.정리


4.회고

-장점 : 무엇이 좋았나

-단점 : 나의 미숙함?

1. 어떻게 일했는가

: Git-flow 이전 이후의 업무방식

no	항목	내용	Tool	Tool (이후)
1	준비	개발 내용 협의	Outlook, Teams, Powerpoint	Outlook, Teams, Powerpoint
2	개발	형상관리	SVN	
		일정관리	Trello, SpreadSheet	
		이슈공유	Teams, 컨플루언스	
		협업	협업이 미흡하고, 도움 요청 정도..? 메신저, 대화, 컨플루언스 등	
		코드리뷰	없음	
		테스트	웹 화면을 통한 테스트 (시연)	웹 화면을 통한 테스트 (시연)
3	배포		FTP, jenkins	FTP, jenkins

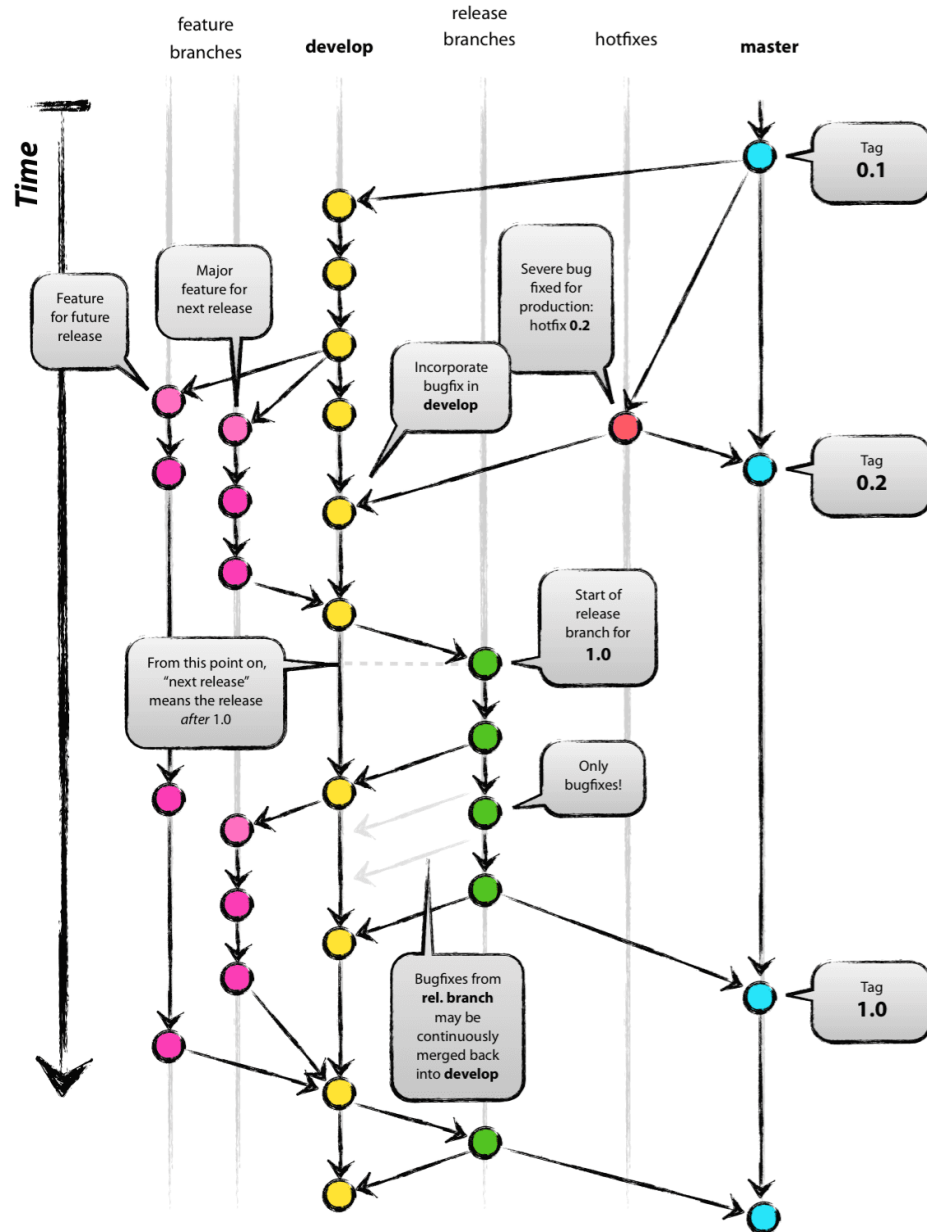
개발자의
주요 관심사

2. Git-flow 소개

- ❑ master : 마스터, 리포지터리의 기준, 배포 가능
- ❑ develop : 다음 릴리즈를 위해 개발 중
- ❑ feature : develop 에서 checkout 하여 기능 개발
- ❑ release : 배포하기 이전에 (master에 가기 전) 대기
- ❑ hotfix : 배포된 이후 긴급 버그를 수정



- ❑ **master*** : 마스터, 리포지터리의 기준, 배포 가능
- ❑ **develop*** : 다음 릴리즈를 위해 개발 중
- ❑ **feature*** : develop 에서 checkout 하여 기능 개발
- ~~❑ release : 배포하기 이전에 (master에 가기 전) 대기~~
- ~~❑ hotfix : 배포된 이후 긴급 버그를 수정~~



출처 : Vincent Driessen, "A successful Git branching model"
(<https://nvie.com/posts/a-successful-git-branching-model/>)

3. git-flow로 일해보기

: 개발 협의 완료 & Milestones 생성

1. 개발 협의 완료

이러쿵 저러쿵 사내 소통 채널 (ex. Outlook, Teams, Jandi, Slack, PowerPoint, SpreadSheet)을 통해 개발 요구사항이 정해졌다. 개발자들은 개발을 시작해보자.

2. GitLab Milestones 생성

마일스톤은 **이벤트**다.

이벤트는 Task 일수 있고, 작은 버그일수도 있고, 신규 프로젝트 일수도 있다.

Milestones?

Milestones in GitLab are a way to track issues and merge requests created to achieve a broader goal in a certain period of time.

Milestones allow you to organize issues and merge requests into a cohesive group, with an optional start date and an optional due date.

출처 : GitLab Docs <https://docs.gitlab.com/ee/user/project/milestones/>

3. git-flow로 일해보기

: GitLab Milestones

2. GitLab Milestones - 미리보기

- 마일스톤에 대한 설명
- 마일스톤 소속 각 이슈들의 상태
- 상태 정보
 - 진행 상황 (%)
 - 일정 : 시작일, 종료일
 - 이슈 개수, 해결 개수
 - MR 수

이슈

마일스톤 내용

상태

Open Milestone Apr 24, 2023–May 22, 2023

Edit Promote Close milestone Delete

진단평가 학원 BM 학제 개편

진단평가 리포트 학원 BM의 학제 관련 데이터 업데이트

오픈 : 5월 22일 오전 (예정)

Improve milestones with Burndown Charts.

Burndown Charts are visual representations of the progress of completing a milestone. At a glance, you see the current state for the completion a given milestone. Without them, you would have to organize the data from the milestone and plot it yourself to have the same sense of progress. [Read more](#)

Contact your Administrator to upgrade your license.

The tabs below will be removed in a future version

Learn more about [issue boards](#), to keep track of issues in multiple lists, using labels, assignees, and milestones. If you're missing something from issue boards, please create an issue on [GitLab's issue tracker](#).

Issues 11 Merge Requests 4 Participants 2 Labels 3

Unstarted Issues (open and unassigned) 0

Ongoing Issues (open and assigned) 2

Completed Issues (closed) 9

운영 DB 데이터 변경 사건 고찰

#26 일반

현재 학제 DB 구조 정리, 공유 (w. 상품연구팀)

#16 To Do

jenkins git 연동

#25 Doing

학원 학제 개편 작업 테이블 컬럼 update문 작성했습니다

#21 Doing

모바일 페이지 수정 여부 확인

#17 Doing

학제 테이블 데이터 엑셀 파일 요청 (to. 정현 팀장)

#13 To Do

개편 작업 일정 및 작수 회신 (todo. 박성준 팀장)

#12 To Do

개발 완료 후 develop 병합

#11 To Do

1차 구현

#10 Doing

학제개편 개발용 브랜치 생성 후 최초 커밋

#9 To Do

진단평가 학원 BM 학제 개편 개발범위 논의

#7

3. git-flow로 일해보기

: branch checkout & issue 생성

3.branch checkout

```
git checkout develop
## feature branch 생성 (작업 내용 : 레이아웃 수정)
git checkout -b feat/layout
```



feat/layout feat : 레이아웃 1차 수정

feat : 레이아웃 수정 필요

Git log

4. Issue 생성

개발이 진행되는 중 이슈가 발생하면 **GitLab** Issues 를 생성하여 공유하고, 코멘트한다.

- 마일스톤과 연동되며
- 이슈 별로 코멘트로 소통하고
- 이슈를 이슈별로 그룹화하여 업무 히스토리를 남길 수 있다.

3. git-flow로 일해보기

: issue 생성

이슈 내용

4. Issue 생성 > 활용도

- 다른 작업 (마일스톤, 이슈, 커밋 등)과 유기적으로 연동
- 이슈에 대해 다양한 포맷으로 소통
 - 그림, 이미지, 링크, 파일 등
- 이슈 별로 업무 히스토리 그룹화
- @ (멘션)을 통해 다른 사용자 호출
 - 다른 조직원 참여, 공유

이슈 소통

ClosedOpened 2 weeks ago by kghReopen issueNew issue

깃 커밋 메시지 양식의 통일 방법

커밋 메시지도 통일 된다면 나중에 작업 내역을 확인하거나, 히스토리를 볼때 편리합니다.

저는 아래 링크의 방법과 같이 작성하는데요.

다른 개발자들도 대부분이 사용하는 포맷입니다.

<https://velog.io/@archivonjang/Git-Commit-Message-Convention>

@kwanghyeon 씨도 이번에 아래와 같은 틀을 적용해서 커밋 메시지를 작성해나가면 좋겠습니다.

아래는 제 개인 깃헙의 특정 저장소 커밋 메시지 중 일부입니다.

Commits on Apr 13, 2023

feat : 4.1 사라진 SQLException

gihyeon6394 committed 2 weeks ago

feat : 4.1.4 Runtime 전환

gihyeon6394 committed 2 weeks ago

kgh @KimGiHyeon commented 2 weeks agoOwner👤🗑️⋮

추가로 .gitmessage.txt 를 작성하여 커밋 메시지를 템플릿화하는 방법도 있습니다.

<https://velog.io/@bky373/Git-%EC%B8%A4%EB%B0%8B-%EB%A9%94%EC%88%9C%EC%A7%80-%ED%85%9C%ED%94%8C%EB%A6%BF>

개발자 본인의 편리를 위해서 위 방법을 쓰기도 하니 참고 바랍니다

kwanghyeonbyeon @kwanghyeon commented 2 weeks agoDeveloper👤🗑️⋮

네!!! 참고해서 적용하도록 하겠습니다

👍 1👎

kgh @KimGiHyeon closed 2 weeks ago🔒

WritePreview

Write a comment or drag your files here...

3. git-flow로 일해보기

: Merge Request & Code Review

5. Merge Request & Code Review

- 내가 작성한 브랜치에 대한 병합 승인 요청
- MR은 Git 협업의 핵심
- MR 액션에 연관된 여러 옵션을 활용
 - 브랜치 보호 정책 (merge 인가 인원 배정)
 - MR 작성자는 요청 시 승인자 명시

MR 요청자
커밋 내역

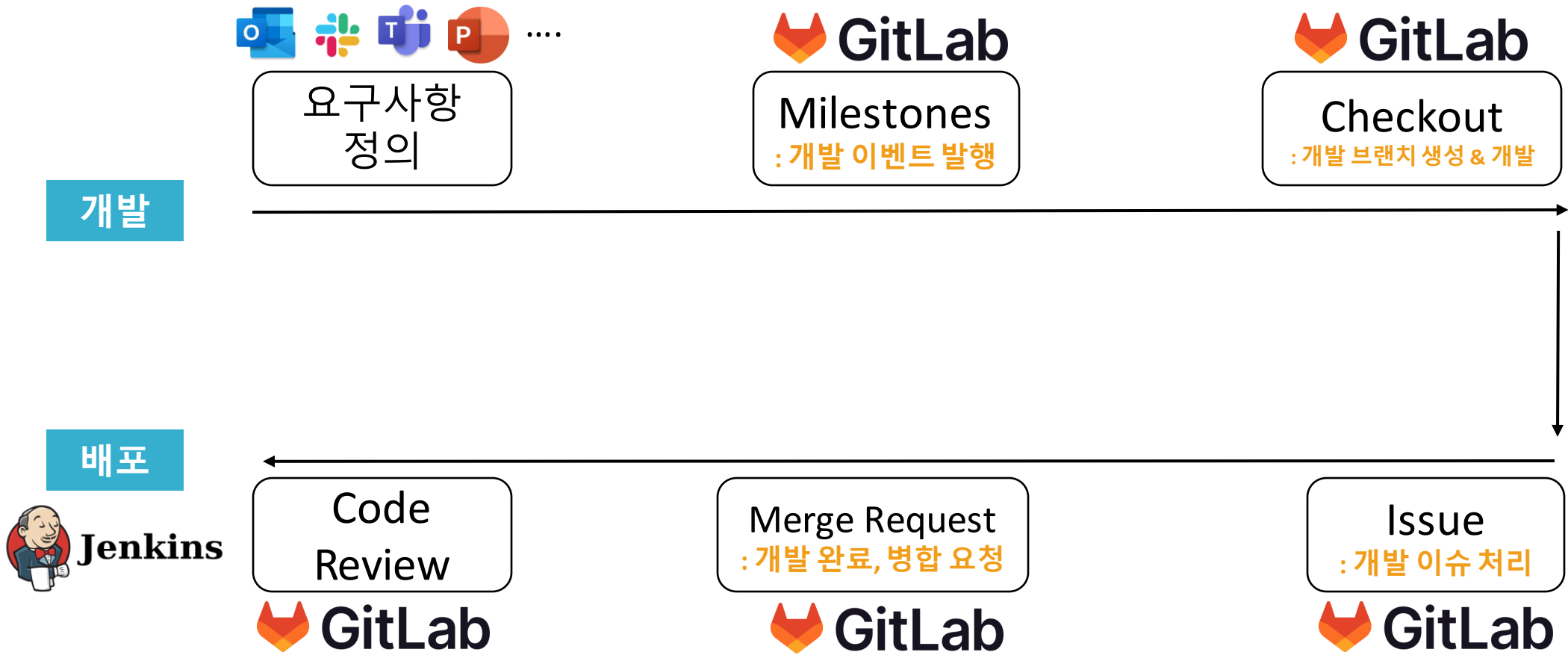
코드리뷰

The screenshot displays a GitHub Merge Request (MR) for the file `src/main/webapp/html/evaluationResult/evaluationResult_001.html`. The code diff shows changes to a table structure, with line numbers 170 through 178 visible. The table has columns for `data-curri-stg` and `data-curri-level`, with values like "Rookie", "RisingStar", "AllStar", and "MVP".

Below the code diff, there is a discussion thread. The first comment is from `kgh @KimGiHyeon`, stating "Started a discussion on the diff 2 weeks ago" and "Last updated by kwanghyeonbyeon 2 weeks ago". The second comment is from `@kwanghyeon`, asking "여기 `data-curri-level` 값들도 바꿔어야할 거예요" (These `data-curri-level` values also need to be changed, right?). The third comment is from `kgh @KimGiHyeon`, responding that the current data is from a static RDBMS and suggesting to use enums or JavaScript objects for better I/O performance. The fourth comment is from `kwanghyeonbyeon @kwanghyeon`, replying that the current data is from a static RDBMS and suggesting to use enums or JavaScript objects for better I/O performance.

3. git-flow로 일해보기

: 정리



4. 회고

: 장점 - 무엇이 좋았나

업무 효율 : 코드리뷰 용이

- 코드 리뷰 형식이 자유로움
- 리뷰 시점은 개인의 스케줄에 따라
- 리뷰 내용의 히스토리화
- 리뷰 결과가 소프트웨어에 반영

업무 내역 : 업무 히스토리 관리

1. 개발 외의 영역은 사내 tool 을 뒤져보고
2. 개발 단의 히스토리면 **GitLab** 에 들어와서
 - I. Milestone : Task 를 거시적으로 확인
 - II. issues : 세부 내용 확인
 - III. commit & MR history : 개발자의 상세 작업 내용 확인

업무 투명성 : 소스 코드 책임의 공유

- 명확한 업무 역할 : 소스 작성자, 소스 리뷰자
- 그러나 소스의 승인자는 MR에 참여한 모두
- 다음 릴리즈의 책임은 MR 요청자가 아니라, **MR 참여자 모두**
- 팀으로서 다음 릴리즈 내용을 투명하게 파악하고 공감

성장 : 업무 태도

- 소스는 내가 아닌 우리 팀의 품질
- 리뷰할 때는 감정이 아닌 코드로
- 근거, 설득력 없는 리뷰내용은 불필요
- Good or Bad 보다는 Better or Not
- [블로그 "깃 좀더 이해하기" 시리즈 포스팅 \(링크\)](#)

4. 회고

: 단점 - 나의 미숙함?

Git 은 툴이다

- 새로운 장치 (장애물)
- 충돌 상황의 해결
- Git을 모른다면 스테디
- 협업 정책 선정



- 우린 깃을 개발하는 개발자가 아님
- 툴에 매몰되지 말 것
- 깃에 매몰될 바에야 SVN 으로 롤백
- **Git-flow 의 필요조건**
 - 협업
 - 코드리뷰

업무 복잡도

- 귀찮음 : 개발단계에서 이루어지는 프로세스들
- 불편함 : 시니어는 머릿속에서 모두 가능한 일을 GitLab에 모두 명시, 공유해야함



- 브랜치 전략 간소화
- Git-flow 는 법이나 시스템 강제가 아니다
- 리뷰 인원 범위 축소 : 개발팀 전원일 필요 없다
- GitLab 컨텐츠 범위 축소
- SVN trunk-based

소감

- 산출물보다, 프로젝트 업무 히스토리 자체의 중요성
 - *"그때 왜 이렇게 했었지?"*
- 실현 가능성이 있느냐. 물음표다