



GIT 컨벤션

Develop Rule

made by 김현우

main branch에 PR이나 push 하지말 것!

기능별 브랜치 작성

1. Branch Naming Convention

Git flow

▼ 참고이미지

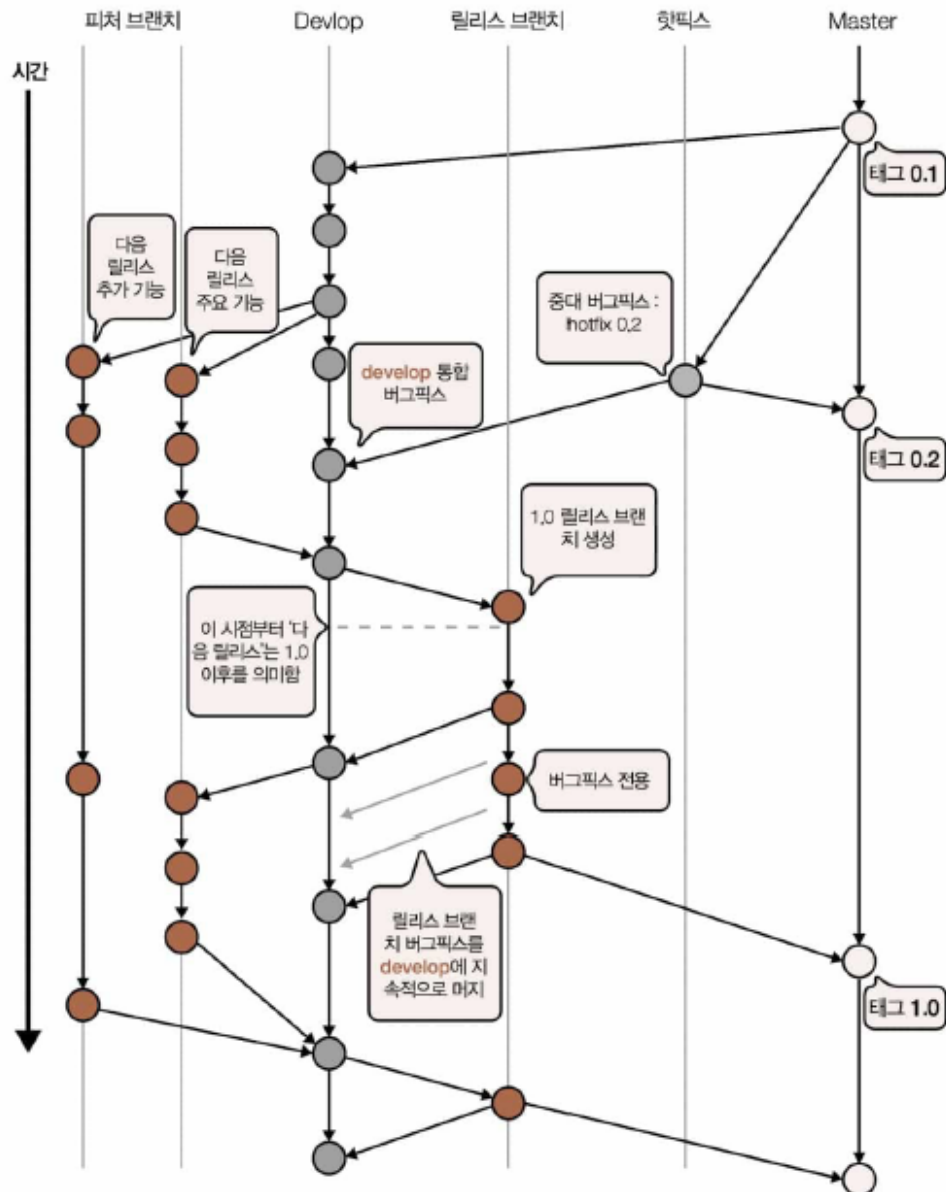


그림 2-14 깃폴로의 브랜치 관리와 통합¹⁴

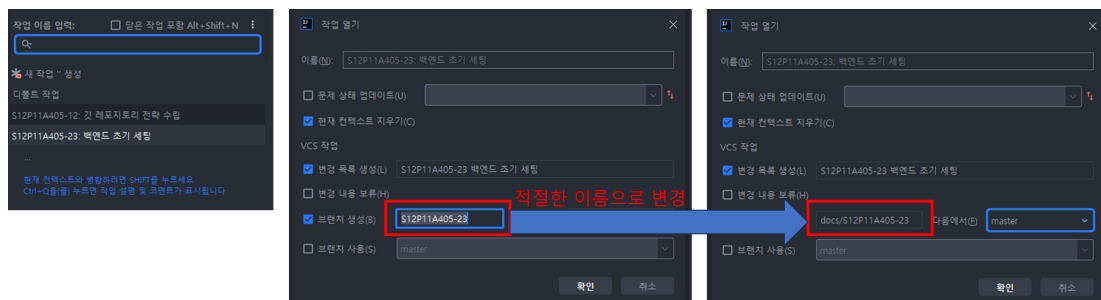
- **master** → 최종 릴리스용
- **develop** → 개발 기간 동안 배포 및 배포 테스트까지 develop에서 진행
- `{type}/{hyunwoo}/{이슈 키}/{이슈 요약}` → **Jira 이슈와 연결**해 작업별 브랜치 생성/작업 → develop에 PR
 - type: feat, fix, style,...
 - 이슈 요약: 일반적으로 목적을 나타내는 이름을 사용



▼ Jira에서 브랜치 생성 방법

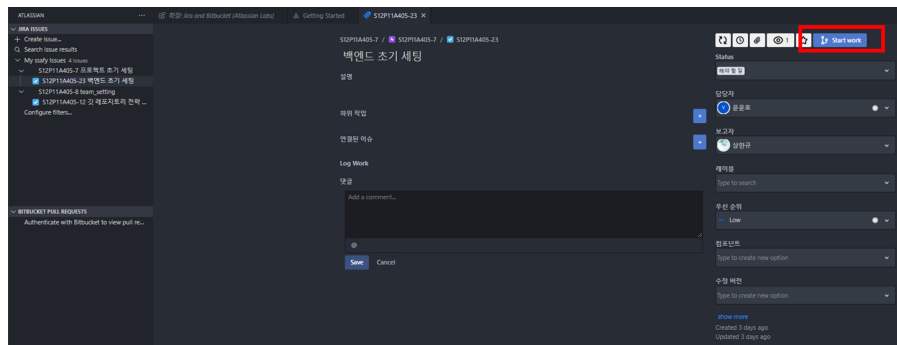
▼ IntelliJ에서 브랜치 생성 방법

- IntelliJ와 Jira 연결
 - 참고: <https://dealicious-inc.github.io/2022/02/15/web-pr.html>
 - Server URL: <https://ssafy.atlassian.net/>
- **Shift + Alt + N** → 나에게 할당된 작업 선택 → 브랜치 이름 변경/생성

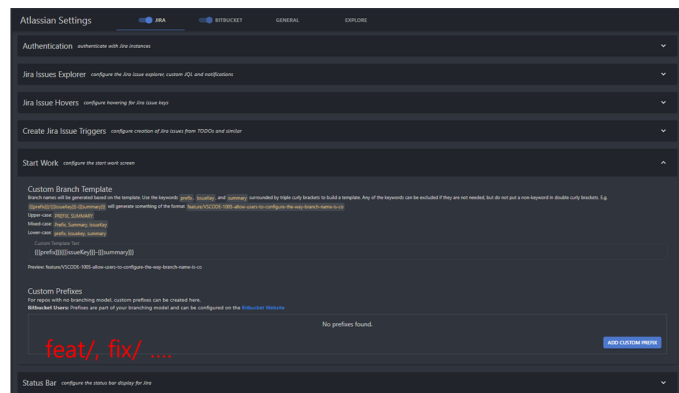
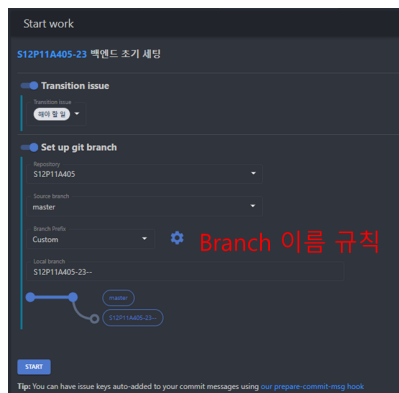


▼ VSCode에서 브랜치 생성 방법

- VSCode와 Jira 연결
 - 참고: <https://aisj.tistory.com/77>
- Extension에서 Jira 통해서 브랜치 이름 변경/생성



Start work



• Branch Name

- 일반적으로 목적을 나타내는 이름을 사용
- 성을 뺀 이름을 영어 소문자로 작성, 약자가 아닌 전체 이름을 작성 (kimhyunwoo → hyunwoo)
- 상태/이름/지라 이슈
 - 기능 개발: `feat/hyunwoo/S12P11A405-39`
 - 버그 수정: `fix/{name}/{feature name}`
 - 코드 수정: `refactor/hyunwoo/ocr`

2. Commit Message Structure

Commit message 작성



커밋 메시지 가이드

https://github.com/RomuloOliveira/commit-messages-guide/blob/master/README_ko-KR.md



Angular Commit convention을 존용

<https://github.com/angular/angular/blob/main/CONTRIBUTING.md#commit>

- .gitmessage.txt
기존 사용 방식 : git commit -m 'asdfasdf' → 사용 X
gitmessage 사용방식 : git commit → 사용 O , vim 에디터 처럼 템플릿에 title, message 작성
- gitmessage 사용 설정
 - git config --global commit.template ./gitmessage.txt 경로(절대경로 혹은 상대 경로 둘 다 가능) → 설정 방식

기본적인 commit message 구조

- 각 파트는 빈 줄로 구분

제목 (Type: Subject)

(공백)

본문 (Body)

(공백)

Footer

- **Subject(필수)**
 - 제목은 50자 이내
 - 마침표 및 특수기호 사용 금지
 - 영문인 경우 동사(원형)을 가장 앞에, 첫 글자는 대문자로 작성
- **Body(선택, 권장)**
 - 72자 이내로 작성
 - 최대한 상세히 작성
 - 무엇을, 왜 변경했는지 작성
- **Footer(선택)**

- issue tracker ID 명시하고 싶은 경우 작성
- 유형: #이슈 번호 형식으로 작성
- 이슈 트래커 유형
 - Fixes: 수정 중(아직 해결되지 않은 경우)
 - Resolves: 이슈를 해결
 - Ref: 참고할 이슈가 있을 때 사용
 - Related to: 관련된 이슈 번호

3. Commit Type

Tag Name	Description
feat	새로운 기능 추가
fix	버그 수정
docs	문서 수정
style	코드 포매팅, 코드 변경이 없는 경우
refactor	프로덕션 코드 리팩토링, 기능의 수정이 없는 경우
test	테스트 코드 추가
chore	소스코드, 테스트 파일을 제외한 수정사항(예: build.gradle)
remove	파일을 삭제한 경우
rename	파일 또는 디렉터리 명을 수정하거나 옮기는 작업만 있는 경우

예시





fix: 사용자 정보를 가져오는데 정보가 없는 경우 발생하는 NullPointerException 수정

사용자 정보가 없을 경우 빈 리스트 반환하도록 수정

feat: 모든 사용자 정보 조회 기능 추가

전체 사용자 정보를 반환하는 api 생성
List<MemberDto> members 반환

Git Commit Message Content

- 왜 변경 사항이 생겼는지 적어라. 
- 왜 이런 이슈가 발생했는지 적어라. 
- 이 commit을 보는 사람들이 이슈를 모두 알고 있다고 생각하지 마라. 
- 이 프로젝트의 코드를 혼자 쓰고 있다고 생각하지 마라. 
- commit message의 제목은 매우 중요하다. 