

챗GPT를 활용한 Git Flow 관리 자동화

개발자와 챗GPT의 협업으로 Git Flow 백포팅 자동화 스크립트를 만든 실제 경험을 소개합니다.



1 프로젝트 개요

오픈AI GPT-4와 협업해 git 관리 스크립트를 만들고 자동화한
실제 경험을 바탕으로 한 발표입니다.




IDE 밖으로 벗어나기 싫어하던 개발자가 챗GPT의 도움으로
Git Flow 관리 자동화를 이뤄낸 실제 경험담



2 챗GPT의 등장과 활용

2023년 젓브레인스의 개발자 에코시스템 현황에 따르면 개발자의 77%가 챗GPT를 사용하고 있습니다.

개인적으로도 다양한 분야에서 활용하고 있습니다.

 SQL  정규표현식  오픈소스 활용  새로운 언어 학습

그러나 업무에 즉시 적용할만한 프롬프트를 작성하는 것은 여전히 쉽지 않았습니다.

3 자동화 결심 계기

"때는 2023년 9월경, 팀 내 운영 업무 중 *git* 백포팅 업무가 진행되는 것을 보고 있었습니다."



집중력 저하

단순 반복 작업으로 인한 집중력 저하와 피로감 누적. 병합 작업에 실수 가능성 증가.



작업 병목

백포팅 작업 중에는 다른 개발자들의 병합 작업이 지연되는 병목 현상 발생.

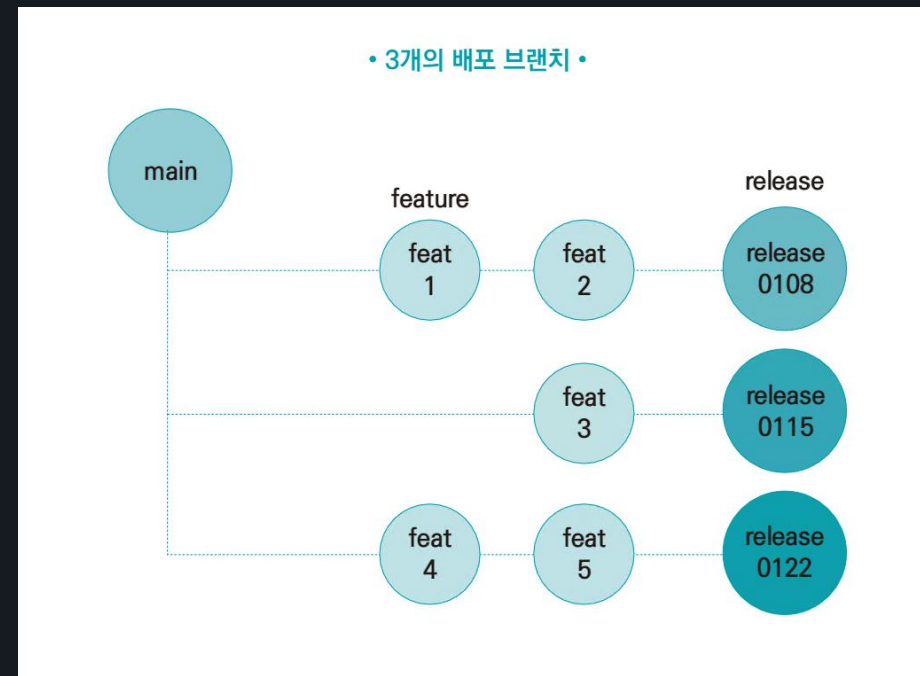


실수 리스크

forced-push 등 실수시 팀 전체에 영향을 주는 위험한 명령어 반복 사용.

4 팀의 Git Flow 전략

- 1 여러 배포 일자가 정해져 있고, 각 날짜에 해당하는 배포(release) 브랜치가 생성됨
- 2 각 기능(feature) 브랜치는 배포 브랜치로부터 생성됨
- 3 기능 구현이 완료되면 생성되었던 배포 브랜치에 병합(merge)됨
- 4 여러 기능이 배포 브랜치에 병합되고 나면 베타배포 및 운영 배포를 기다림



5 백포팅 전략(기차놀이)

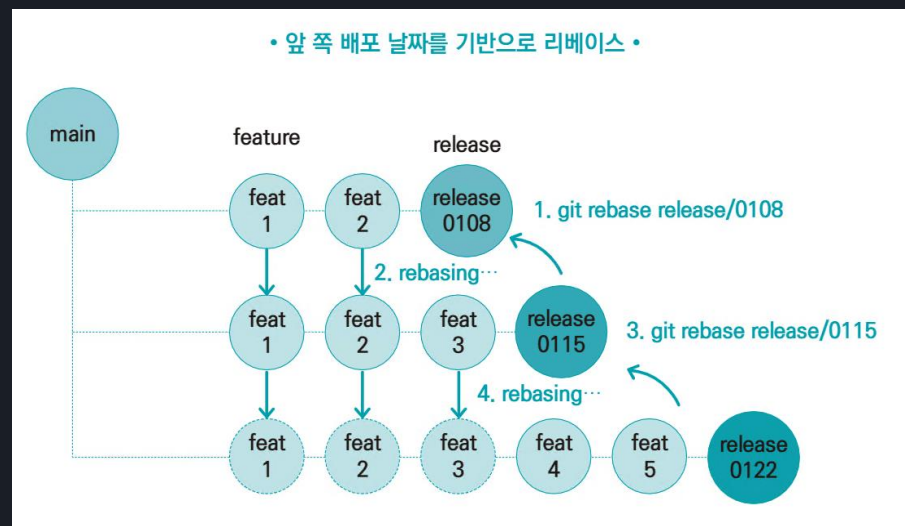
백포팅이란?

최신 소프트웨어의 기능이나 수정 사항을 이전 버전으로 적용하는 작업

팀의 백포팅 전략 - '기차놀이'

앞쪽 날짜 배포 브랜치의 변경사항을 뒤쪽 날짜 배포 브랜치에 리베이스

이 과정을 연쇄적으로 수행하여 가장 마지막 배포 브랜치에는 모든 기능이 포함되도록 함

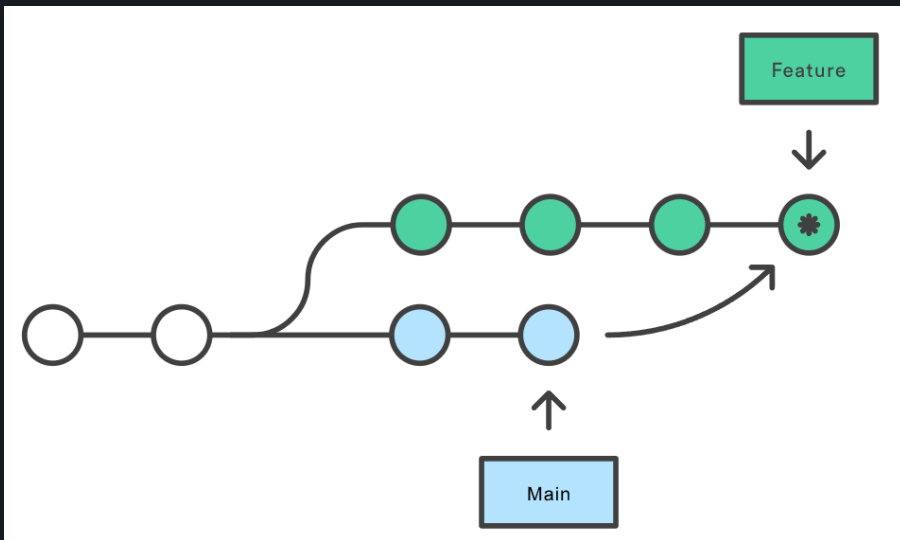


기차놀이란?

앞쪽 브랜치의 변경을 뒤쪽으로 연쇄적으로 전파, 단일 QA 환경에서 여러 배포 브랜치의 기능을 동시에 테스트 가능하게 하는 전략

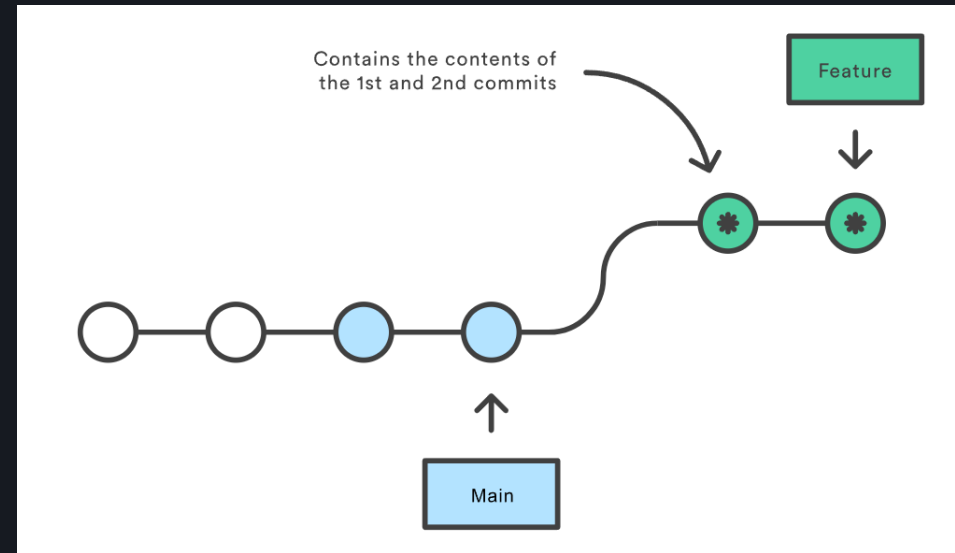
6 Merge vs Rebase: 시각적 비교

Merge 방식



- 🔗 분기 이력이 그대로 보존됨
- ⊕ 추가 병합 커밋이 생성됨
- 🕒 브랜치의 작업 맥락과 시간순서가 명확함

Rebase 방식



- ➔ 커밋들이 새 베이스 위로 재배치됨
- 🔗 선형적인 히스토리가 생성됨
- ⚠ 원래 커밋 ID가 변경됨

7 반복 작업과 그 문제점

- ⚠ 집중력 저하: 단순 반복 작업 중 실수 발생 가능성 증가
- ⌚ 시간 소모: 충돌 없어도 약 5분씩 소요, 여러 브랜치일수록 증가
- 🚫 팀원 간 협업 제약: 작업 중 다른 개발자들의 기능 브랜치 병합 차단
- 🎓 러닝 커브: 모든 작업자가 백포팅 전략을 숙지해야 하는 부담
- 💣 강제 푸시 위험성: `force push`로 인한 코드 유실 사고 가능성

```
# 반복해야 하는 Git 명령어들
git checkout release/20240115 # 앞 날짜
브랜치 리베이스
git rebase -r release/20240108 git push --
force-with-lease

git checkout release/20240122 # 앞 날짜
브랜치 리베이스
git rebase -r release/20240115 git push --
force-with-lease

# 브랜치마다 반복... 😞
```


8 자동화 도전과 챗GPT

🏠 IDE 은둔형 개발자로서의 고민

> 셀 스크립트 자동화에 대한 심리적 장벽

💡 챗GPT를 새로운 도구 탐색에 활용

"이거 자동화하면 모두 편해질 것 같은데요, 제가 스크립트 짜 볼게요!"

🤖 챗GPT에게 작업 자동화 목표 전달

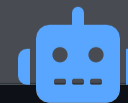
💬 백포팅 자동화를 위한 첫 프롬프트 작성

</> git rebase 기반 스크립트 설계 착수

😊 챗GPT의 코드로 자신감 상승



IDE 안의
편안함



챗GPT의
도움



자동화
도전

9 챗GPT와의 협업 과정

프롬프트 대화 예시

"안녕! 코드 관리를 git rebase -r 옵션을 통해서 하고 있어. 내가 하고 싶은 건 이런 작업을 자동화하는 거야.

1. origin의 release/yyyy-MM-dd 형식의 브랜치들이 있음
2. 시간 순서상 두 번째로 빠른 release 브랜치로부터, 바로 앞 시간 순서의 release 브랜치를 rebase -r 하고 origin에 forced push를 해
3. 시간 순서상 마지막 브랜치까지 완료할 때까지 2번 작업을 반복해"

"git rebase -r 옵션으로 작업을 하려면, 각 브랜치를 시간 순서대로 정렬하고 이를 반복적으로 rebase를 진행해야 합니다. 다음과 같은 셸 스크립트를 작성해 보겠습니다.."

반복 피드백

</> 스크립트 점진적 개선 과정

초기 스크립트 버전

```
#!/bin/bash
branches=$(git for-each-ref --sort=committerdate
refs/remotes/
origin/release * --format='%(refname:short)' | sed
's/origin\//')

```

```
if [ ${#branches[0]} -lt 2 ]; then
echo "Rebase할 브랜치가 충분하지 않습니다."
exit 1
fi

```

개선된 브랜치 필터링

```
branches=$(git ls-remote --heads origin | grep
'refs/heads/release/
[0-9]\{8\}\$' | sed 's .*refs/heads/ ?' | sort -t '/' -k 2)

```

약 30번의 대화 후 완성



3시간 소요

10 시행착오 및 문제점



프롬프트의 맥락 전달 한계

복잡한 팀 백포팅 작업의 모든 맥락을 한 번의 프롬프트로 전달하기 어려움
대화 방식의 점진적 맥락 전달이 효과적이었음



AI의 할루시네이션

원격 저장소 브랜치 초기화 과정에서 잘못된 명령어 제안

```
git reset --hard origin/$branch # 실제로는 작동하지 않음
```



실제 적용 시 상황 차이

스크립트 실행 시 예상하지 못한 브랜치 형식 처리 문제
오류 처리와 예외 상황 고려가 필요했음

11 최종 스크립트 및 효과

```
#!/bin/bash
echo "[기차놀이 차장]기차놀이를 시작합니다.rebase 과정에서 conflict
가 발생해 실패하는 경우는 수동으로 기차를 몰아주세요"
# 원격 저장소의 변경 사항 가져오기
git fetch origin
# 원격 저장소에서 브랜치 목록 가져오기
branches=$(git ls-remote --heads origin | grep 'refs/heads/release/[0-9]\\{8\\}\\$' | sed
's?\\.refs/heads/??' | sort -t '/' -k 2)
# main 브랜치 체크아웃 및 최신화
previous_branch="main"
git checkout $previous_branch git pull

# 변경 사항 확인
if ! git diff-index --quiet HEAD --; then
    echo "[기차놀이 차장]로컬 변경 사항이 발견되었습니다."exit 1
fi

for branch in $branches; do # 원격 브
랜치 백업
    git branch -D $branch-bak
```

3시간 협업의 결실

챗GPT와의 약 30여 번의 프롬프트 교환을 통해 완성
한 스크립트로 백포팅 자동화 구현

작업 효율성 향상

5분 소요되던 반복 작업을 명령어 한번으로 자동화
개발자 집중력 소모 감소

팀 협업 개선

다른 개발자들의 기능 브랜치 병합 지연 감소
QA 환경 효율 대폭 향상

12 협업에서 얻은 교훈 및 마무리



챗GPT는 요술 지팡이가 아니다 - '도서관의 모든 책을 읽은 선생님'과 같은 조언자로서 활용하는 것이 효과적



대화형 협업의 중요성 - 단번에 완벽한 프롬프트보다 맥락을 점진적으로 공유하는 대화 방식이 더 효과적



검증과 이해가 필수 - AI가 제시한 코드도 할루시네이션 가능성이 있으므로 원리를 이해하고 검증하는 과정이 중요

“혼자선 업무를 낼 수 없었던 작업을 챗GPT와의 협업으로 작업할 수 있었다는 경험 이번 작업의 가장 큰 성과입니다.
이런 경험을 통해 기술적 두려움을 극복하고 더 많은 개발자들이 새로운 도전을 할 수 있길 바랍니다.”