perplexity

현재 운영 중인 웹서비스 유지보수에 대한 개발-운영 서버 소스관리 분석과 개선 방안

현재 프로세스 분석

현재 운영하고 있는 개발 및 운영 반영 프로세스를 보면, GitHub를 통한 형상관리는 잘 구축되어 있지만 몇 가지 중요한 문제점들이 발견됩니다[1] [2] [3].

주요 이슈 및 문제점

1. 순차적 작업 프로세스 부재

문제 상황:

- 여러 요청사항이 동시에 처리되지 않고 한 개씩 순서대로만 진행
- 테스트 서버에 반영된 여러 요청 중 일부만 선택적으로 운영에 반영할 때 다른 개발 내용을 삭제해야 하는 상황

원인:
이는 **브랜치 전략의 부재** 때문입니다. 현재 방식으로는 각 기능별로 독립적인 브랜치를 만들지 않아서 발생하는 문제입니다[1] [4] [5].

개선 방안:

- **Git Flow 전략** 도입[1] [2]
  - `develop` 브랜치: 개발 통합용
  - `feature/*` 브랜치: 각 요청사항별 독립 브랜치
  - `release` 브랜치: 운영 배포 준비용
  - `main(master)` 브랜치: 운영 배포 완료 브랜치

2. 배포 도구 부재로 인한 수동 관리

문제 상황:

- 개발자가 직접 복사-붙여넣기로 소스 관리
- 배포 과정의 오류 가능성 높음
- 배포 이력 추적 어려움

개선 방안:

도구	특징	장점
Jenkins[6] [7]	가장 널리 사용되는 오픈소스 CI/CD 도구	무료, 플러그인 풍부, 브랜치별 자동 배포 가능
GitHub Actions	GitHub 내장 CI/CD	GitHub와 완벽 통합, 설정 간단
GitLab CI/CD	GitLab 내장 도구	GitLab 사용 시 추천

권장 개선 방안

1. GitHub 브랜치 전략 구축

```
main (운영서버 배포)
├── develop (개발 통합)
    ├── feature/user-request-1
    ├── feature/user-request-2
    └── feature/user-request-3
```

작업 흐름:

1. 사용자 요청별로 `feature` 브랜치 생성
2. 개발 완료 후 `develop` 브랜치로 병합
3. 테스트 완료된 기능만 선택하여 `release` 브랜치 생성
4. 운영 배포 승인 후 `main` 브랜치로 병합[3] [8]

2. CI/CD 파이프라인 구축

단계별 자동화:

단계	작업 내용	도구
개발	코드 푸시 → 자동 빌드 → 개발서버 배포	Jenkins/GitHub Actions
테스트	PR 생성 → 자동 테스트 → 테스트서버 배포	자동화 스크립트
운영	승인 후 → 자동 빌드 → 운영서버 배포	배포 스크립트

3. 환경별 설정 분리

현재 누락된 부분으로, 환경별로 다른 설정 파일을 관리해야 합니다[9] [10] [11]:

```
├── config/
    ├── development.properties    # 개발환경
    ├── test.properties           # 테스트환경
    └── production.properties     # 운영환경
```

4. 배포 자동화 스크립트 구성

```
# Jenkins 파이프라인 예시
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                // 소스 빌드
            }
        }
        stage('Test') {
            steps {
                // 자동 테스트
            }
        }
        stage('Deploy') {
            steps {
                // 환경별 자동 배포
            }
        }
    }
}
```

즉시 개선 가능한 사항

1. **브랜치 명명 규칙 도입**
   - `feature/YYYY-MM-DD-description` 형식 사용
   - 각 요청사항별 독립 브랜치 생성
2. **환경별 설정 파일 분리**[9] [12]
   - 개발/테스트/운영 환경의 DB 연결 정보 분리
   - 민감 정보는 환경변수로 관리
3. **간단한 배포 스크립트 작성**[13] [14]
   - 수동 복사-붙여넣기 대신 스크립트 사용
   - 배포 이력 로그 남기기

단계적 도입 계획

1단계 (즉시): 브랜치 전략 도입, 환경별 설정 분리
2단계 (1개월): Jenkins 같은 CI/CD 도구 도입
3단계 (2개월): 완전 자동화된 배포 파이프라인 구축

이러한 개선을 통해 여러 요청사항의 병렬 처리, 선택적 배포, 그리고 안정적인 운영서버 관리가 가능해질 거야. 특히 브랜치 전략 도입만으로도 현재 겪고 있는 대부분의 문제를 해결할 수 있을 것 같아!

⁂

1. https://adjh54.tistory.com/364

2. https://inpa.tistory.com/entry/GIT-⚡-github-flow-git-flow--브랜치-전략
3. https://sungjk.github.io/2023/02/20/branch-strategy.html
4. https://haon.blog/github/git-flow/
5. https://devocean.sk.com/blog/techBoardDetail.do?ID=165571&boardType=techBlog
6. https://blog.containerize.com/ko/top-5-open-source-deployment-tools-in-the-year-2021/
7. https://apidog.com/kr/blog/free-ci-cd-tools-2/
8. https://amaran-th.github.io/Github/[Github] Git 브랜치 전략/
9. https://maiplesyrup.tistory.com/88
10. https://luminousolding.tistory.com/111
11. https://velog.io/@sssungjin/React-.env-를-이용한-개발-환경-배포-환경-분리-전략
12. https://hstory0208.tistory.com/entry/Spring-배포-환경-별로-설정파일-분리하기-프로필
13. https://groups.google.com/g/ksug/c/JBtQJAMuCbE
14. https://hsb422.tistory.com/entry/문법-개발-운영-QA-스테이징-환경에-배포-방법