# 아래는 우리 회사의 실제 개발 및 배포 프로세스와 겪고 있는 문제 사례입니다.

[현재 프로세스]

- GitHub Desktop 프로그램을 사용해 소스 이력/협업을 관리하고 있습니다.
- 개발은 각자 feature 브랜치에서 진행, develop 브랜치로 병합(merge) 후 테스트 서버에 반영합니다.
- 테스트 서버에는 지금까지 개발된 모든 기능이 반영되어, 클라이언트가 테스트합니다.
- 운영 서버에는 클라이언트가 “운영 반영”을 요청한 기능만 따로 추려서, 필요 없는 소스는 삭제 후 수동으로 배포합니다.
- 자동화 배포, 서버 Git 연동, SSH/토큰 인증 등은 보안 정책상 모두 불가합니다.
- war 파일을 직접 빌드해서, 서버에 원격접속(VDI) 후 수동 업로드 방식만 사용 가능합니다.

[실제 겪은 문제/불편]

- 테스트 서버에 모든 개발 기능이 섞여 올라가, 운영에는 반영되지 않아야 할 소스(기능)까지 섞여 들어가 서비스 오작동 및 민원이 발생했습니다.
- 운영에 일부 기능만 반영하려다 소스 삭제/추리기 과정에서 실수 발생.
- 브랜치 병합(Merge), 롤백(복구), 협업 충돌 관리도 어렵습니다.

[보충설명]

1. 운영/테스트 서버의 구조
   운영 서버와 테스트 서버는 각각 어떤 방식으로 관리되나요? (예: 물리적으로 다른 서버인지, 같은 Tomcat 환경 내에서 포트만 다른지 등)
   => 같은 서버에 DEV 톰캣과 REAL 톰캣으로 따로 톰캣서버가 존재.해당 톰캣 webapp 폴더 밑에 war 파일을 붙여넣고 서비스. 아파치 웹서버와 연동. 각각의 도메인으로 접속.
2. 현재 배포 방식에서의 워크플로
   WAR 파일을 빌드해서 수동 업로드한다고 하셨는데, 이 때 정확히 어떤 디렉토리에 배포하나요?
   => 개발 서버는 DEV 톰캣 폴더 하위의 webapp 폴도, 운영 서버는 REAL 톰캣 폴더 하위 webapp 폴더에 배포

테스트 서버에는 어떤 기준으로 반영되며, 배포 시점은 누가/언제 결정하나요?
=> 개발자가 개발이 완료된 이후 클라이언트에게 테스트 요청하기 전에 개발 서버에서 war파일을 만들어 복사 붙여넣기함.

3. 현재 브랜치 전략 관련
   "운영에 반영해야 할 기능만 따로 추리는" 작업은 어떤 식으로 하고 있나요? 예: develop에서 작업

된 기능 중 일부만 main에 cherry-pick 하는 식인지, 아니면 수동 복붙?
=> 개발서버에서 개발된 소스만 관리. 운영 및 테스트에는 개발자기 직접 수동으로 복붙.

4. 협업 인원
   몇 명이 동시에 개발에 참여하고 있는지, GitHub 사용 경험은 모두 있는지 알고 싶어요.
   => 2명이 참여. GitHub에 대한 경험은 없음.
5. GitHub Desktop 사용 범위
   오직 GitHub Desktop만 사용하고 있는지, CLI나 VSCode Git 플러그인은 아예 사용하지 않는 건가요?
   => 오직 GitHub만 사용. STS4에서 작업한 다음 GitHub Desktop 프로그램을 실행하여 수정 및 추가한 소스를 push함.

**이런 상황에서,**

1. **GitHub Desktop 프로그램을 기준으로,**
   개발/테스트/운영 서버 소스 관리와 war 파일 배포 프로세스를
   **초보자도 이해할 수 있게, 한글로 단계별로 재정의**해 주세요.
2. 특히,
   - GitHub Desktop으로 브랜치/커밋/병합/푸시 등 실무에 바로 쓸 수 있는 상세 매뉴얼(설정/화면 예시 포함)
   - feature/develop/main 브랜치 관리 및 테스트/운영 서버 분리 배포 전략
   - 운영에 일부 기능만 반영 시 소스 혼입/오작동 방지법
   - 협업 중 실수, 충돌, 롤백(복구) 등 문제 해결 전략
   - 자동화/서버 Git/배포툴 없이 GitHub Desktop만으로 안전하게 협업/관리하는 방법
   - (개발환경: STS4.5, Zulu-8, Tomcat 9, Maven, MSSQL, MyBatis, JQGrid 등)
3. 그리고, 현재 프로세스의 **비효율/위험요소**와,
   **우리 환경에서 실현 가능한 구체적 개선/최적화 절차**도 같이 안내해 주세요.

**실제 문제 예시**까지 모두 반영해서,
**GitHub Desktop을 활용한 “실전 소스/배포 관리 프로세스 개선안”**을
초보자도 따라할 수 있게, 보다 자세하고 꼼꼼하게
**한글로, 단계별로, 예시/팁/가능하면 관련 이미지까지 포함해서** 설명해 주세요.
사용자가 우리 회사의 개발 및 배포 프로세스를 개선해달라고 요청했다. 현재 상황을 정리해보면: **현재 상황:** - GitHub Desktop 사용 중 - feature → develop → 테스트 서버 배포 흐름 - 운영 서버는 수동으로 소스를 추려서 배포 - 자동화 불가 (보안정책) - WAR 파일 수동 업로드만 가능 - 같은 서버에 DEV/REAL 톰캣 분리 - 개발자 2명, GitHub 경험 없음 - STS4, GitHub Desktop만 사용 **문제점:** - 테스트 서버에 모든 기능이 섞여서 운영 배포시 혼란 - 수동 소스 추리기 과정에서 실수 발생 - 브랜치 관리, 협업 충돌 관리 어려움 **요청사항:** 1. GitHub Desktop 기준으로 프로세스 재정의 2. 상세 매뉴얼 (설정/화면 예시 포함) 3. 브랜치 관리 전략 4. 소스 혼입 방지법 5. 협업/충돌/롤백 해결법 6. 자동화 없이 안전한 관리법 7. 비효율/위험요소 분석 및 개선안 언어는 한글로, 친구처럼 편하게, 단계별로 설명해야 함. 이 상황을 위해 다음 정보들을 조사해야 한다: 1. GitHub Desktop 사용법과 브랜치 관리 전략 2. Git Flow와 GitOps 없이도 안전한 배포 전략 3. 소규모 팀에서의 효과적인 협업 방법 4. WAR 파일 기반 배포에서의 베스트 프랙티스 5. 운영/개발 환경 분리 전략 정보를 수집한 후 실제 화면 이미지나 프로세스 다이어그램도 생성해야 할 것 같다.