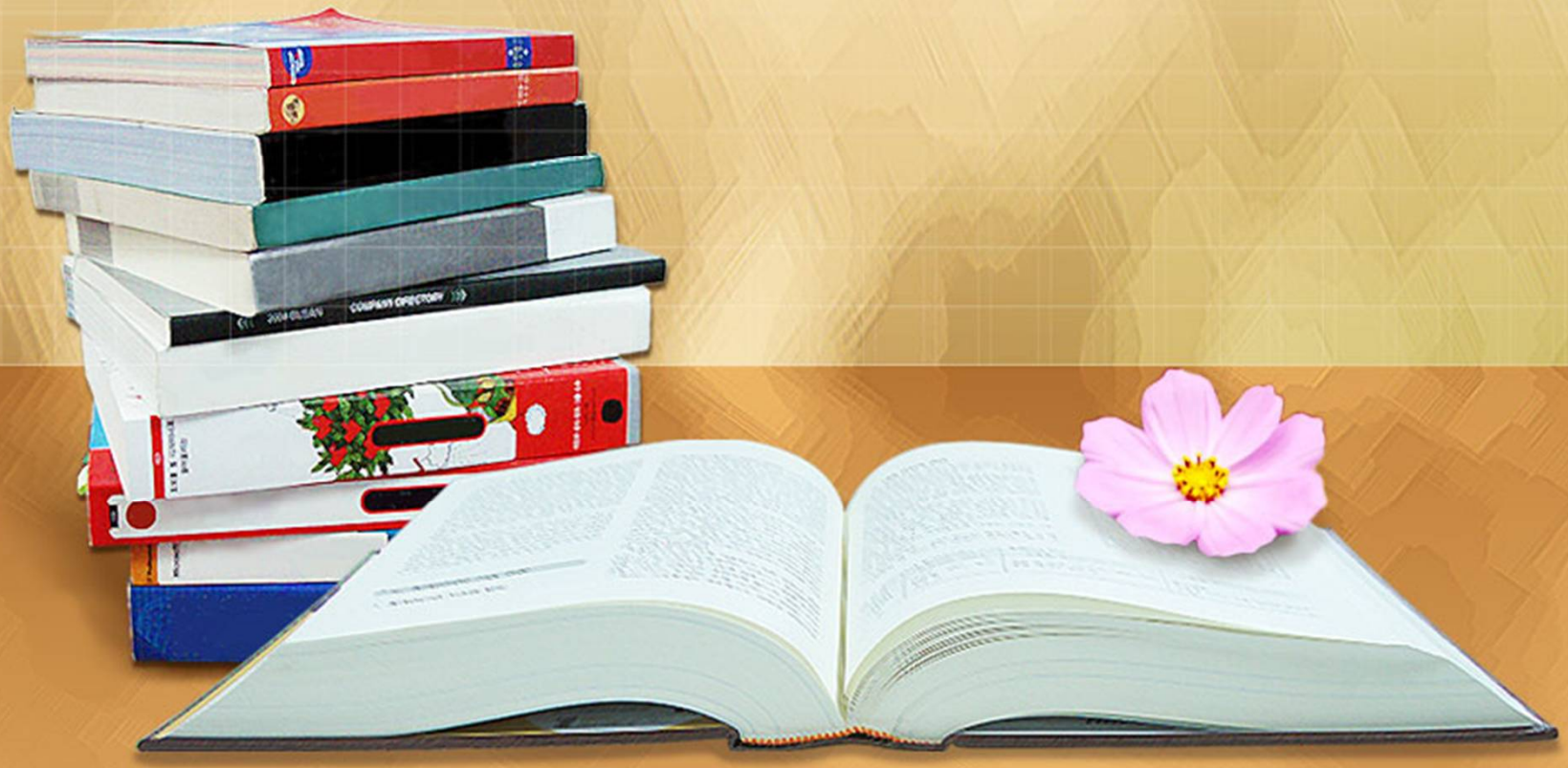
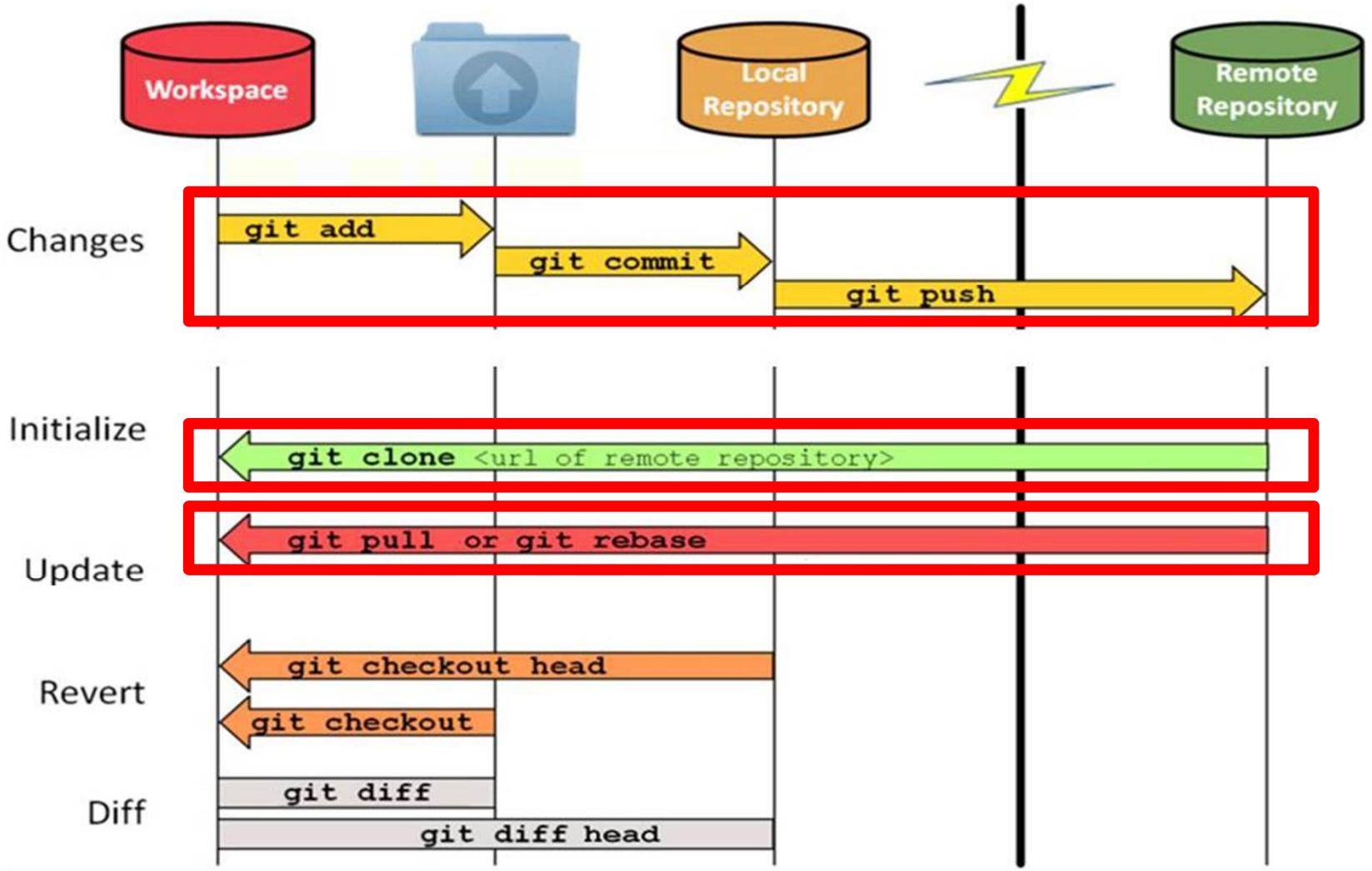
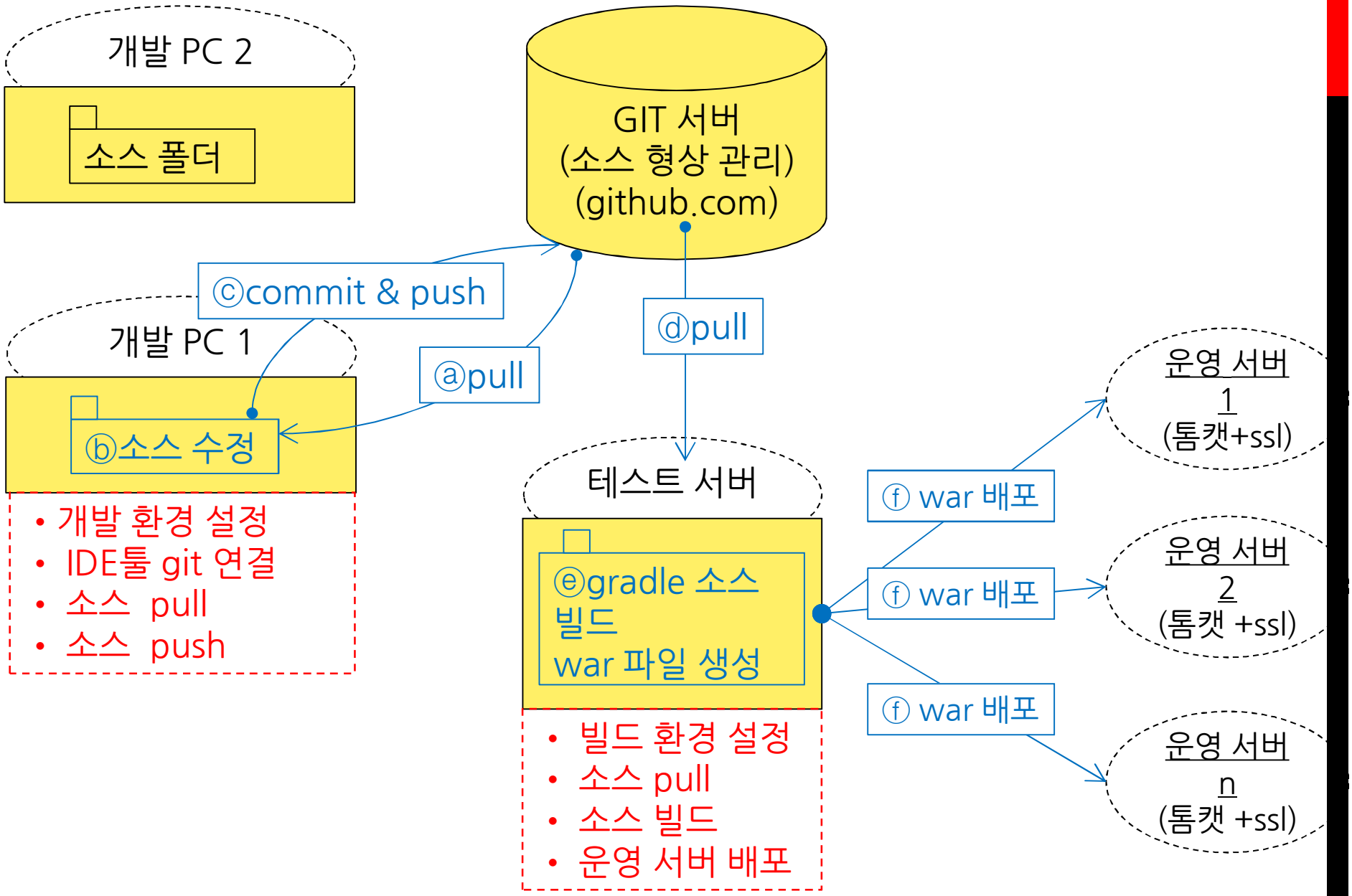
애플리케이션 배포
git & gradle



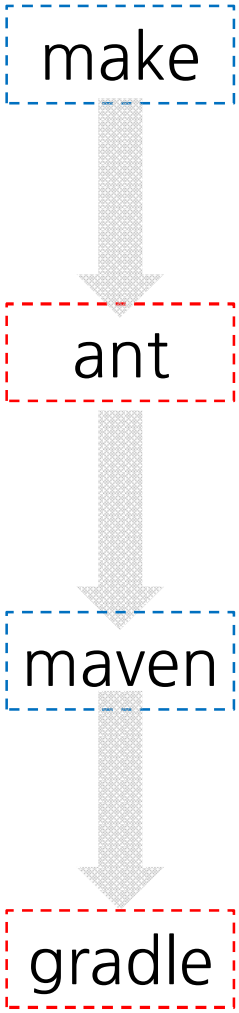
Git Repository



실습 과정


개발 PC 2
소스 폴더
GIT 서버
(소스 형상 관리)
(github.com)
ⓒcommit & push
ⓓpull
개발 PC 1
ⓐpull
ⓑ소스 수정
• 개발 환경 설정
• IDE툴 git 연결
• 소스 pull
• 소스 push
테스트 서버
ⓔgradle 소스 빌드
war 파일 생성
• 빌드 환경 설정
• 소스 pull
• 소스 빌드
• 운영 서버 배포
ⓕ war 배포
ⓕ war 배포
ⓕ war 배포
운영 서버 1 (톰캣+ssl)
운영 서버 2 (톰캣 +ssl)
운영 서버 n (톰캣 +ssl)

빌드 툴의 역서



소스 형상 관리 툴

CVS

라이선스	GNU GPL v2.0
적용언어	무관
OS	Windows/Linux Mac은 써드 파티 도구
실행환경	Comamand Line Interface
GUI	TortoiseCVS 등 써드 파티 도구

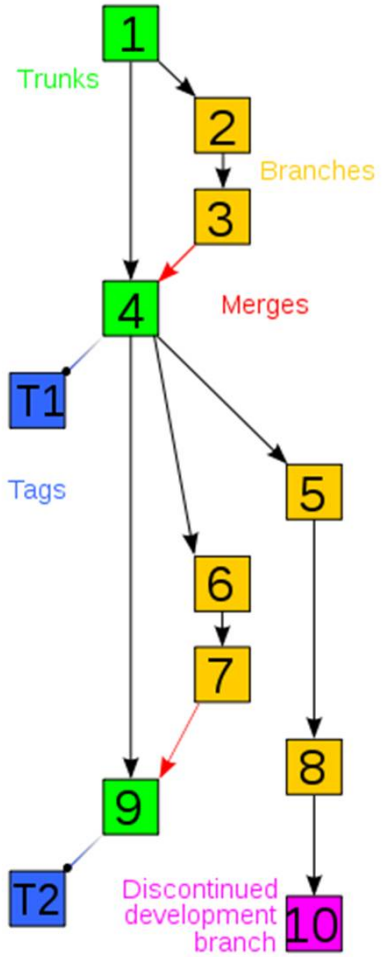
SUBVERSION

라이선스	Apache License v2.0
적용언어	무관
OS	Windows/Linux/Mac
실행환경	Comamand Line Interface
GUI	TortoiseSVN, WinSVN 등 써드 파티 도구

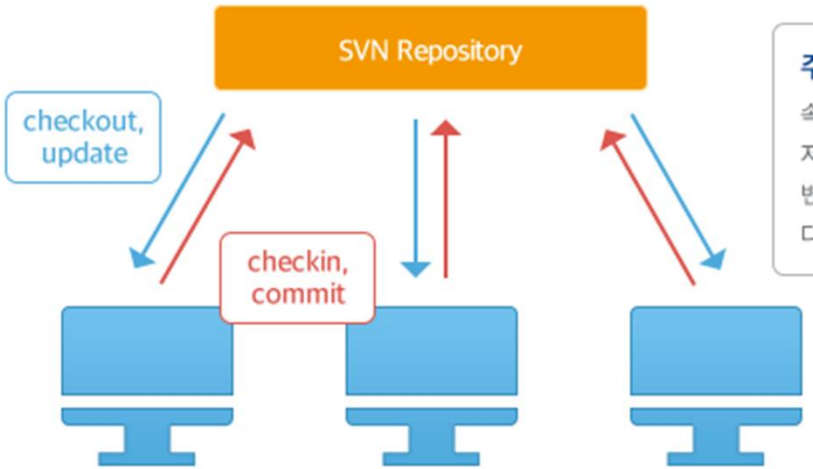
git

라이선스	GNU GPL v2.0
적용언어	무관
OS	Windows/Linux/Mac
실행환경	Comamand Line Interface
GUI	번들로 제공 SourceTree, GitEye, git-cola 등 다양한 써드 파티 도구

1990년	2000년	2005년
		리눅스 커널 개발에 사용
	중앙 버전 관리 시스템	분산 버전 관리 시스템



소스 형상 관리 툴 : SVN

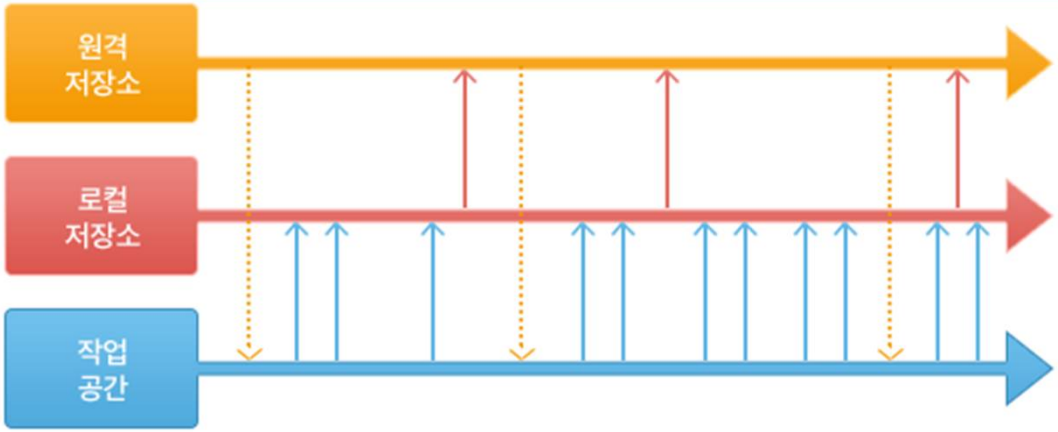


주요 개선 사항

속도 개선
저장 공간 절약
변경 관리 단위가 파일이 아닌 작업 모음 단위
디렉토리나 파일을 자유롭게 이동해도 이력 유지

주요기능	checkout/checkin, update/commit 등 서버 저장소와 클라이언트의 변경 사항 전송 diff를 통한 파일 내용 비교 바이너리 문서 형상 관리 작업 단위의 변경 사항 관리 atomic commit svn:ignore를 통한 형상 관리 배제 자원 지정 기능
장점	CVS 사용자가 쉽게 적응 가능 디렉토리나 파일을 자유롭게 이동해도 이력 유지 gzip을 통한 압축으로 저장 공간 절약 CVS에 비해 빠른 속도 atomic commit으로 커밋 실패 시 롤백 지원 다양한 써드 파티 GUI 도구 존재
단점	trunk, branch, tag가 모두 물리적인 저장 위치를 점유하므로 비효율적 Git에 비해 branch, tag 작업이 무거움

소스 형상 관리 툴 : GIT



주요기능	branch, checkout, commit, tag 등 로컬 환경에서의 형상 관리 기능 push, fetch, pull 등 원격 환경에서의 변경 사항 전송 기능 변경은 했지만 커밋에는 포함하지 않을 수 있는 staging 기능 SVN으로 관리되던 저장소를 Git로 전환해 주는 마이그레이션 기능 diff를 통한 파일 내용 비고 바이너리 문서 형상 관리 작업 단위의 변경 사항 관리 .gitignore를 통한 형상 관리 배제 자원 지정 기능
장점	branch 생성, 이동, 병합이 매우 가벼우므로 branch를 자주 사용하여 상황에 맞게 자주 분기하고 합칠 수 있어 코드 꼬임에 따른 위험 감소 각 로컬에 완전한 로컬 저장소가 있으므로 원격 저장소에 장애가 나더라도 쉽게 복구 가능 여러 번의 커밋을 로컬 저장소에 실행하고, 모아진 커밋을 원격 저장소에 반영할 수 있으므로 네트워킹 빈도는 줄고 속도는 향상 Pack 방식의 압축으로 SVN에 비해 저장 공간 절약 다양한 써드파티 GUI 도구 존재
단점	CVS, SVN과 기본 개념이 많이 다르므로 적응에 시간 필요 checkout, commit 등 텍스트는 같지만 의미나 동작이 SVN이나 CVS와 다른 명령어가 있어 혼란 야기 빈 디렉터리가 저장되지 않음

애플리케이션 구축 절차

[1단계]

개발 PC 설치

개발 환경 설정
- JDK
- IDE(이클립스)
- gradle
- git

IDE와 GIT 연결

[2단계]

개발 서버 설치

빌드 환경 설정
- JDK
- gradle
- git

gradle로 소스 빌드

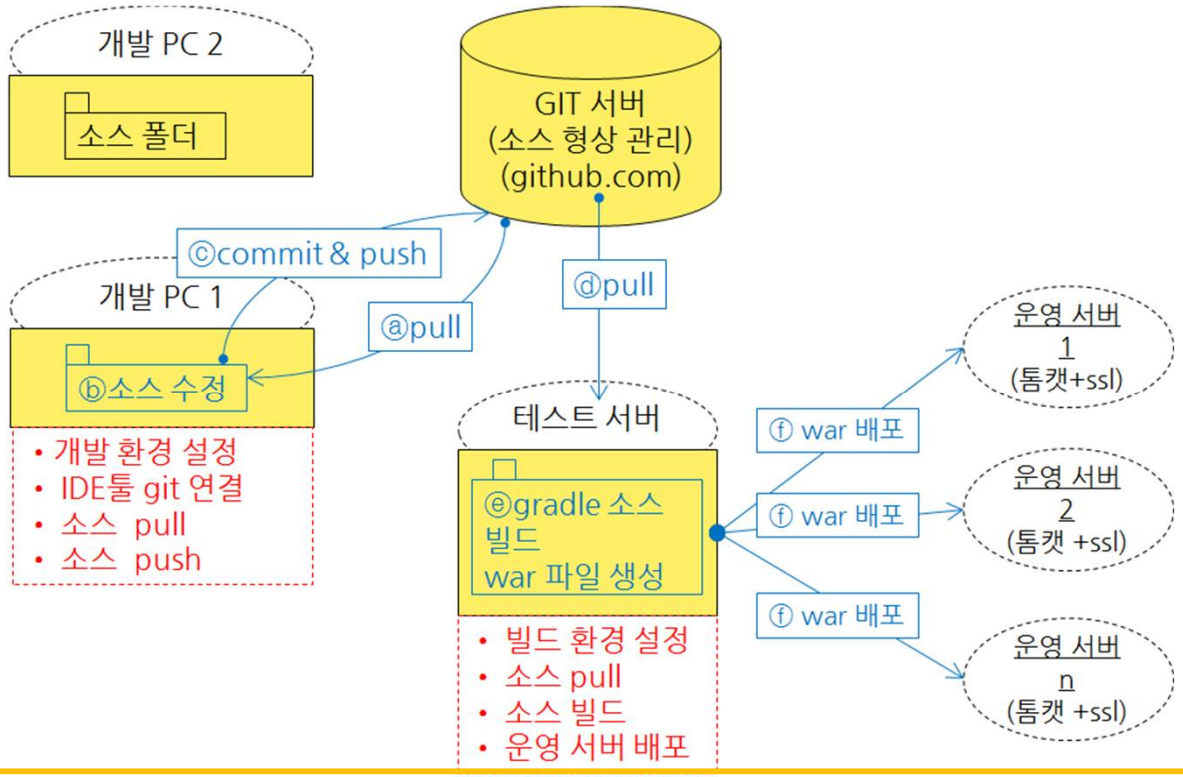
[3단계]

운영 서버 설치

톰캣 설치 & 설정
- ssl 설정

"애플리케이션 배포" 연습

- 실습: 테스트 서버 개발 환경 설정 : JDK, Tomcat , gradle 설치
- 실습: tomcat에 ssl 적용하기
- 실습: 테스트 서버에서 git로 clone, pull 하기
- 실습: 테스트 서버에서 gradle을 이용한 소스 빌드하기
- 실습: 테스트 서버에서 생성된 war 파일을 운영 서버 배포하기



"애플리케이션 배포" 연습

1. github.com 에서 저장소 생성
2. 개발 PC에서
    1. git clone
    2. git add 하기
    3. git commit 하기
    4. git push 하기
3. 테스트 서버에서
    1. git clone 하기
    2. git pull 하기
4. 테스트 서버에서 gradle로 빌드하기
    - **`gradle clean build test war`**
5. 테스트 서버에서 gradle 빌두 후 생성된 war 파일 운영 서버 톰캣에 배포하기
    - **`sudo cp ROOT-????.war /usr/local/tomcat8/webapps/ROOT.war`**
6. 톰캣 로그 실시간 모니터링
    - **`sudo tail -f /usr/local/tomcat8/logs/catalina.out`**
7. 톰캣 재시작
8. 웹 브라우저로 접속하여 배포 여부 확인

"애플리케이션 배포" 복습

1. 톰캣 삭제, 설치 & 설정
2. github.com 에 저장소 생성 후 소스 올리기
3. 테스트 서버에서 git를 이용하여 소스 내려받기
4. 테스트 서버에서 gradle을 이용하여 소스 빌드하기 <-- war 파일 생성.
5. 테스트 서버에서 빌드 후 생성된 war 파일을 운영 서버 배포하기