*** 버전 관리(Version Management)**

- 변경사항을 체계적으로 관리하는 것

*** 소프트웨어 구성(형상) 관리(Software Configuration Management)**

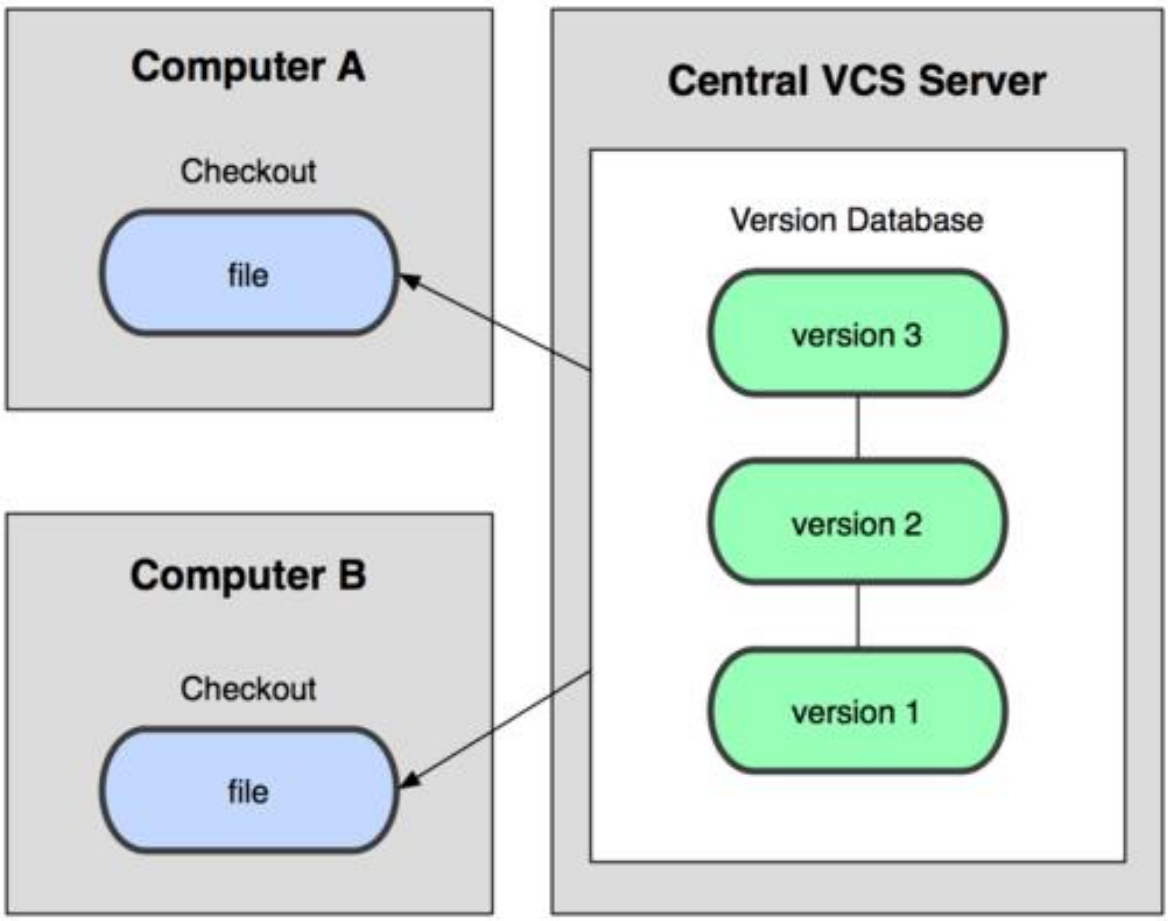
- 소프트웨어의 변경사항을 체계적으로 관리

*** 형상관리를 통해 얻을 수 있는 것**

1. 변경점 관리 : 누가 언제 어떤 내용을 변경했는지 확인 가능
2. 버전 관리 : 브랜치(Branch)등으로 동시에 여러 개발을 가능하게 해줌
3. 백업 & 복구 : 문제 발생 시 백업 및 복구
4. 협업 : 수정사항의 공유

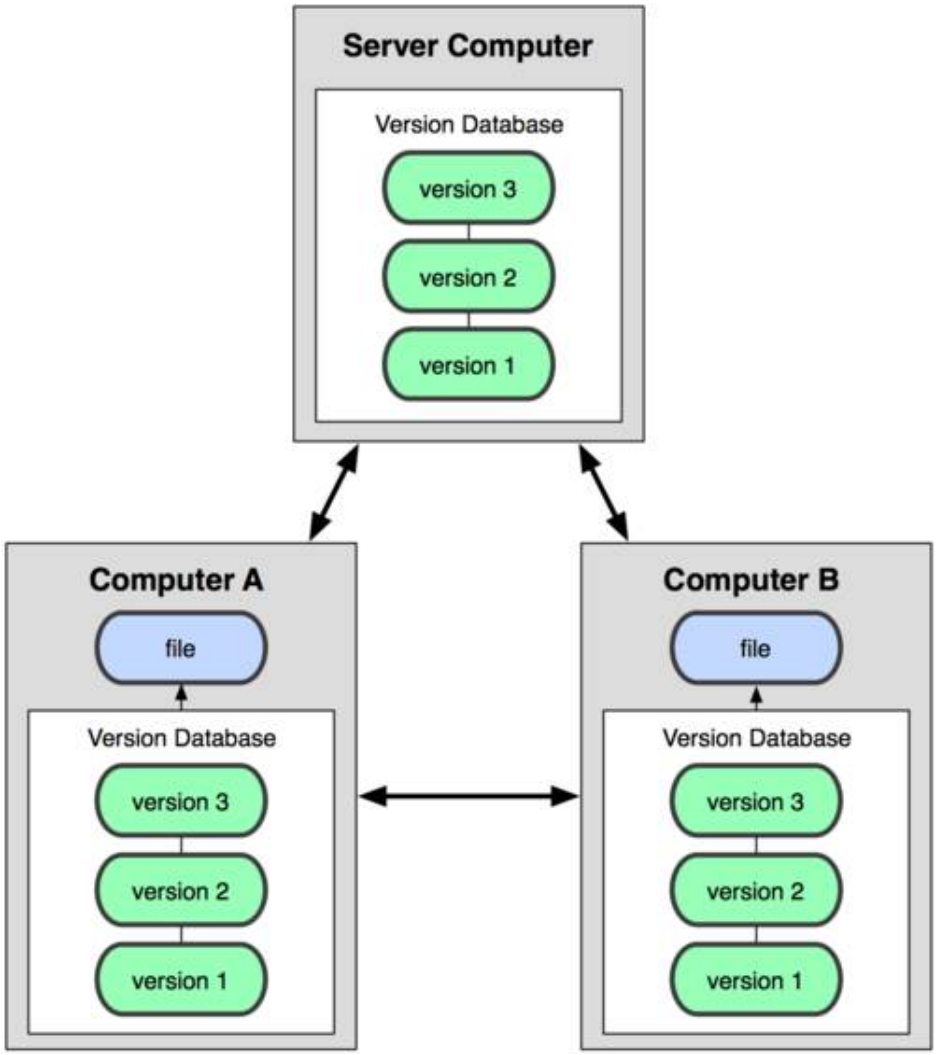
*** 형상관리의 종류**

1. 중앙집중식 버전관리시스템(Centeralized Version Control System: CVCS)



*** Subversion(SVN) : 생략**

2. 분산 버전 관리 시스템(Distributed Version Control System: DVCS)



*** Git**

- Local Repository & Remote Repository

Git은 별도의 버전 관리 서버가 없어도 로컬 환경에서 소스 버전을 관리할 수가 있다.
필요에 따라서 Remote repository에 연결 및 저장 할 수 있다.

- Git은 거의 모든 명령을 로컬에서 실행한다

SVN과 비교했을 때 월등한 속도 차를 보인다. 이는 모든 명령이 로컬 파일과 데이터만을 사용하기 때문이다. 로컬 환경에서도 소스를 커밋(Commit)하고 그 기록(History)들을 살펴볼 수 있다는 장점을 가지고 있다. 로컬에서 실행되는 환경 덕분에 어느 곳에서도 작업한 내용을 커밋할 수 있고, 네트워크가 연결되었을 때 Remote Repository에 소스를 푸시(Push) 할 수 있다.

*** GIT Guide**

- https://tonyne.jeju.onl/2016/06/07/why-use-github-on-small-company/

*** GIT에 대해 더 자세히 알아보기**

- https://git-scm.com/book/ko/v2

*** github.com & bitbucket.org (wiki - https://ko.wikipedia.org/wiki/%EA%B9%83%ED%97%88%EB%B8%8C)**

- https://bitbucket.org/dashboard/overview
- https://github.com/

*** 형상관리용어 정리**

- Repository : 프로그램 소스가 저장되는 저장소
- Checkout : 저장소에서 소스를 받아 오는 것
- Commit : 개발자가 수정, 추가, 삭제등의 갱신정보를 저장소에 반영하는 것
- Update : 최근에 수정된 상태를 로컬에 반영하는 것
- Tag : 꼬리표와 같은 기억하기 쉬운 이름을 부여함
- Trunk : 프로젝트에서 가장 중심이 되는 디렉토리
- Branch : Trunk에서 발생되어진 연관 프로젝트
- Merge : Branch의 수정분을 Trunk에 병합하는 것