Git :: GitHub

- **Git**
    - Git 은 **분산 버전 관리 시스템이다**.
        - Git 으로 프로젝트를 개발하는 사람은 모두 현재 상태의 파일뿐만 아니라 그 프로젝트의 전체 이력을 가지고 있게 된다는 뜻이다.
        - 버전관리 시스템 : 파일의 변경 내역을 계속 추적하도록 개발된 소프트웨어

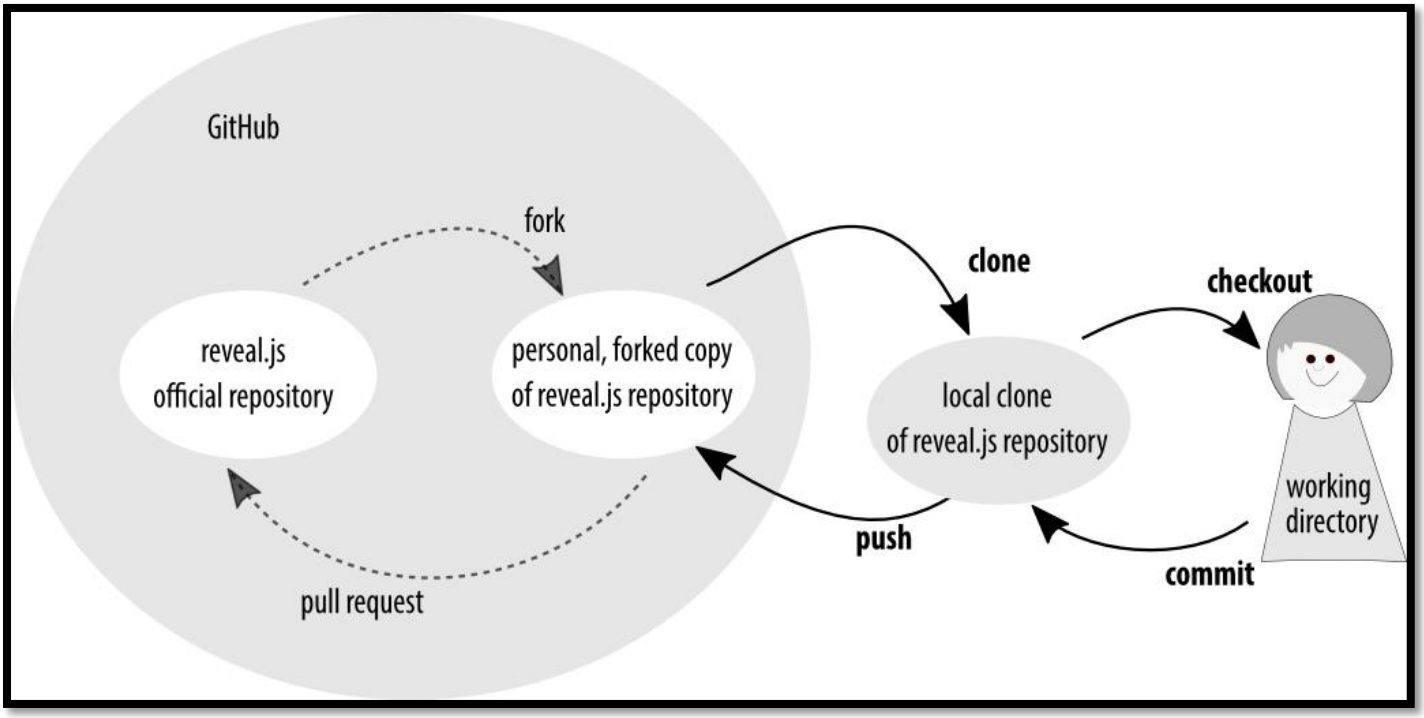
- **GitHub**
    - GitHub 는 Git 저장소(repository)를 업로드 할 수 있는 **호스팅 웹사이트**를 말한다.
    - 웹 상에 소스 코드를 올려서 여러 사람과 공유하는 원격저장소다.
    - GitHub 는 다른 사람들과의 협업을 매우 용이하게 해준다.
        - **리포지토리를 공유할 수 있는 원격저장소**
            - 웹 기반 인터페이스
            - forking
            - pull requests
            - issues
            - wikis
    - Github 는 위와 같은 기능을 제공하여 팀원들과 보다 효율적으로 변경안을 구체화하고 토론하며 검토할 수 있게 해준다.
    - 자신의 PC 에서 작업하는 공간을 Local Repository 라 하며, Github 에 있는 공간을 Remote Repository 라 한다

Git 의 장점

본질적으로 깃의 사용은 1 인 프로젝트를 진행하는 사람에게도 다양한 가치가 있다.

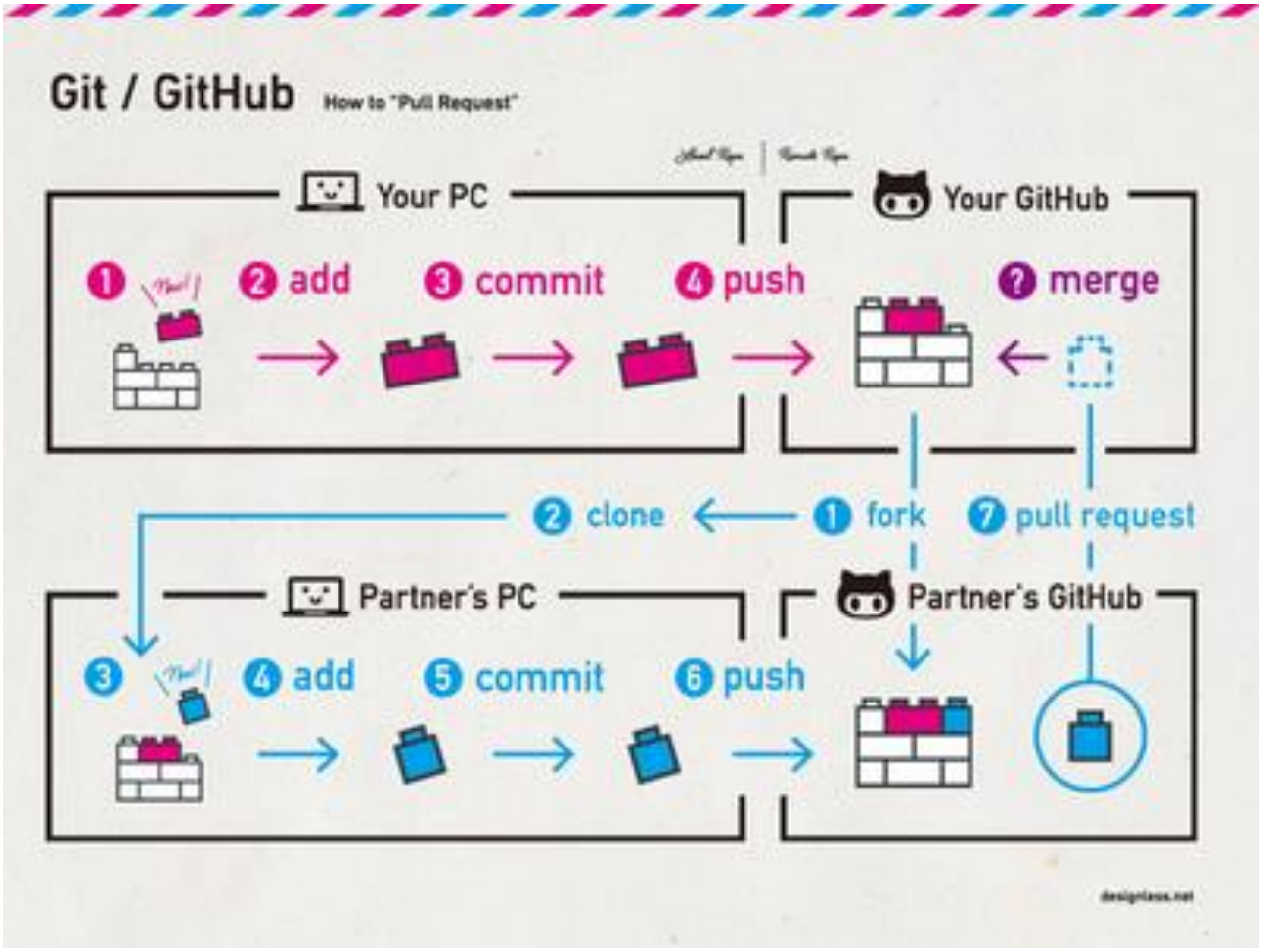
- **변경취소 가능** : 실수 했을 때 구 버전의 작업 파일로 돌아갈 수 있음.
- **모든 변경에 대한 완벽한 이력 History** : 짧게는 하루, 길게는 1 년 전에 프로젝트가 어떤 형태였는지도 알 수 있다.
- **변경한 이유를 기록** : 협업을 하다보면(때로는 내가 작성한 코드라할지라도) 왜 변경했는지 모르겠을 때가 많다. 이때 Git 의 커밋 메시지(**commit message**)를 이용하면 변경한 이유를 쉽게 기록할 수 있으며, 추후에 참조할 수 있다.
- **변경에 대한 확신** : 이전 버전으로의 복귀가 쉽기 때문에 자신을 원하는 대로 다 변경가능한다. 잘 안되면 언제든 이전 버전으로 복귀하면 된다.
- **여러 갈래의 히스토리(History)** : 콘텐츠의 변경 내용을 테스트해보거나 기능을 독립적으로 실험해보기 위해 별도의 브런치(**branch**)를 생성할 수 있다. 완료되면 변경 내용을 마스터 브랜치(**Master branch**)로 병합할 수 있고, 잘 작동하지 않을 경우 삭제 가능하다.

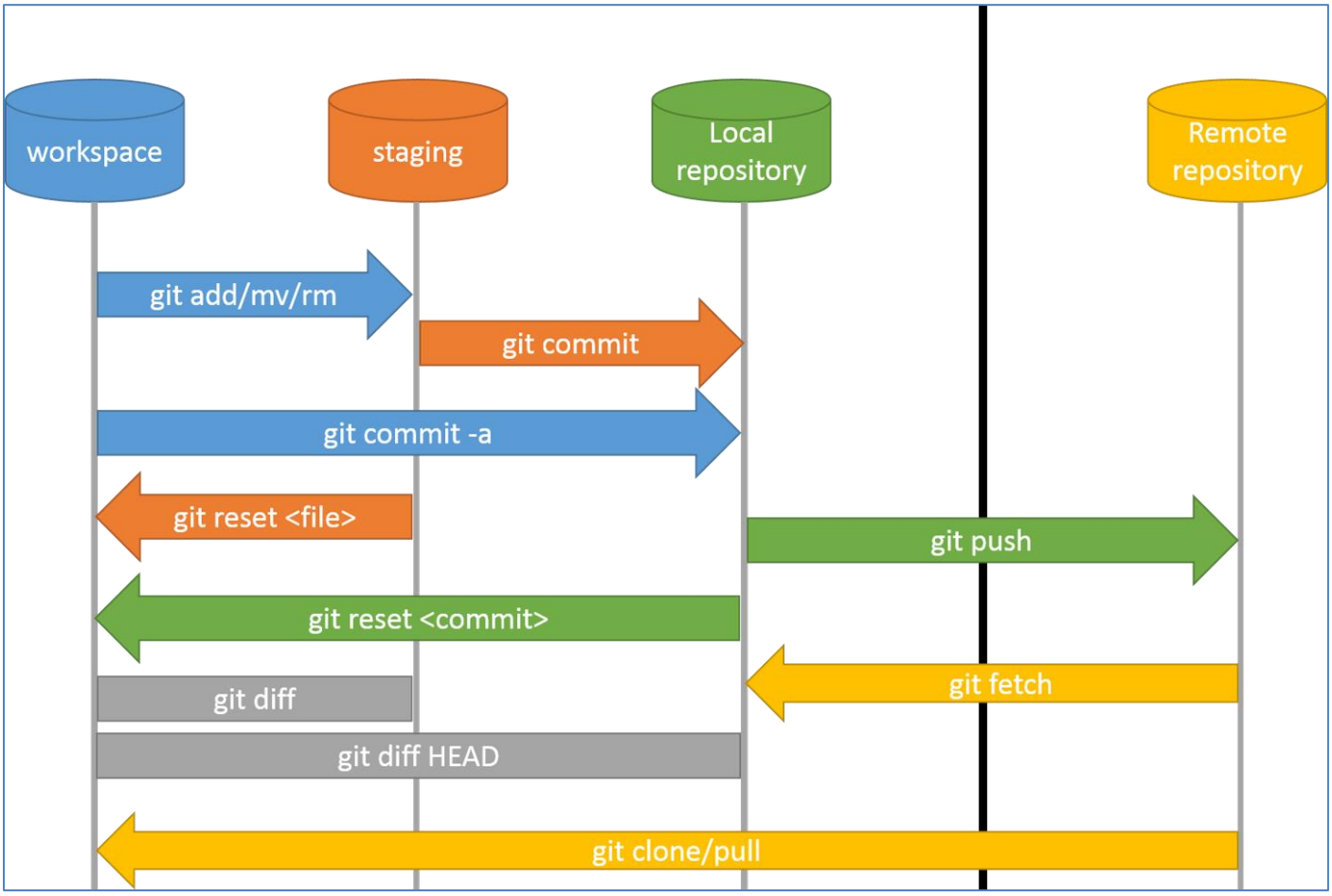
- **충돌 해결 능력** : Git 을 이용하면 여러사람이 동시에 같은 파일을 작업할 수 있다. Git 은 대개 자동으로 변경사항을 병합할 수 있다. 그렇지 못할 경우에 충돌이 무엇인지 알려주고 이를 해결하기 쉽게 해준다.
- **독립된 히스토리(History)** : 여러 사람들이 다른 브랜치(branch)에서 작업이 가능하다. 기능을 독립적으로 개발하고, 완료되었을 때 그 기능을 병합할 수 있다.

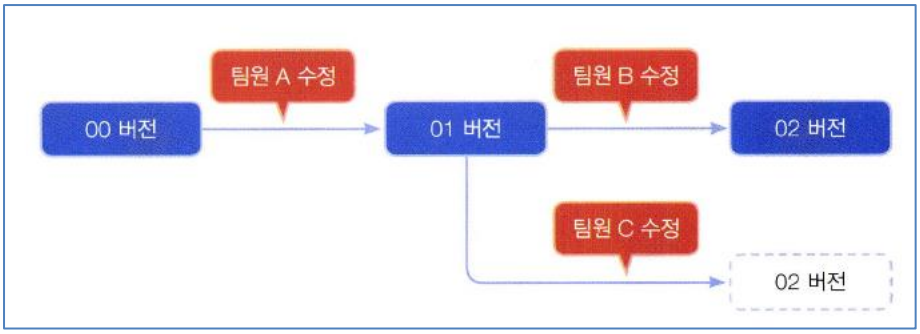


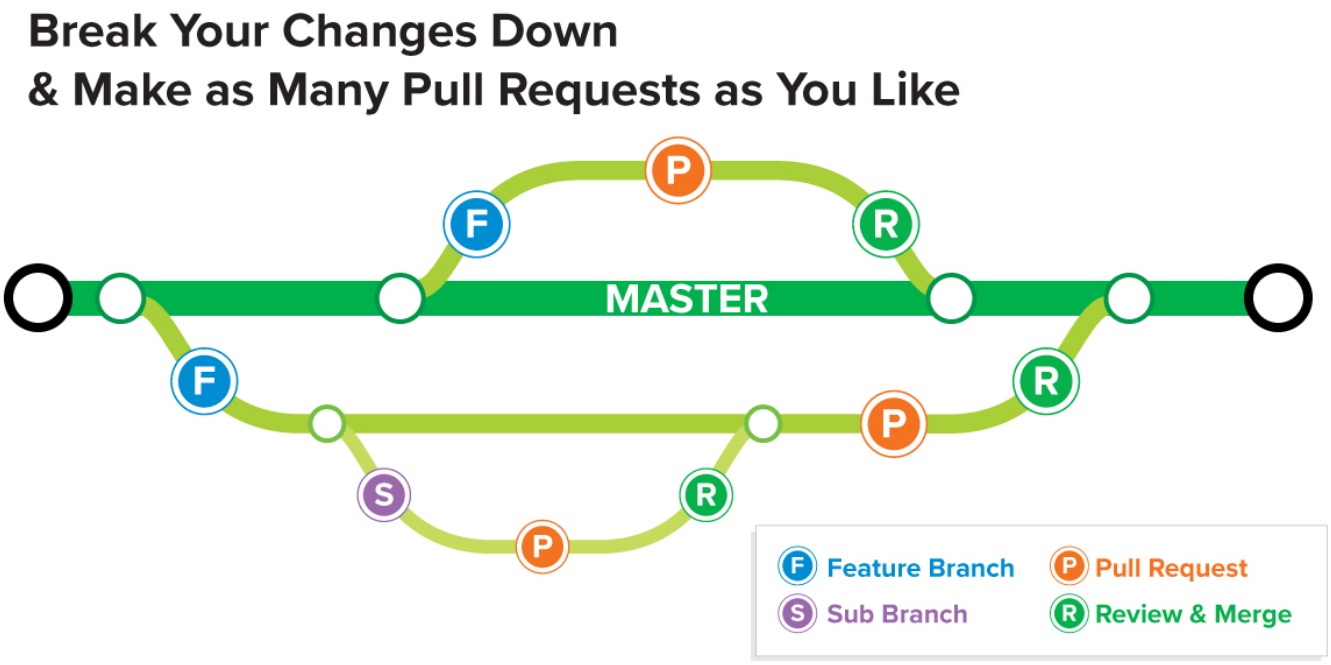
Git : 버전관리시스템

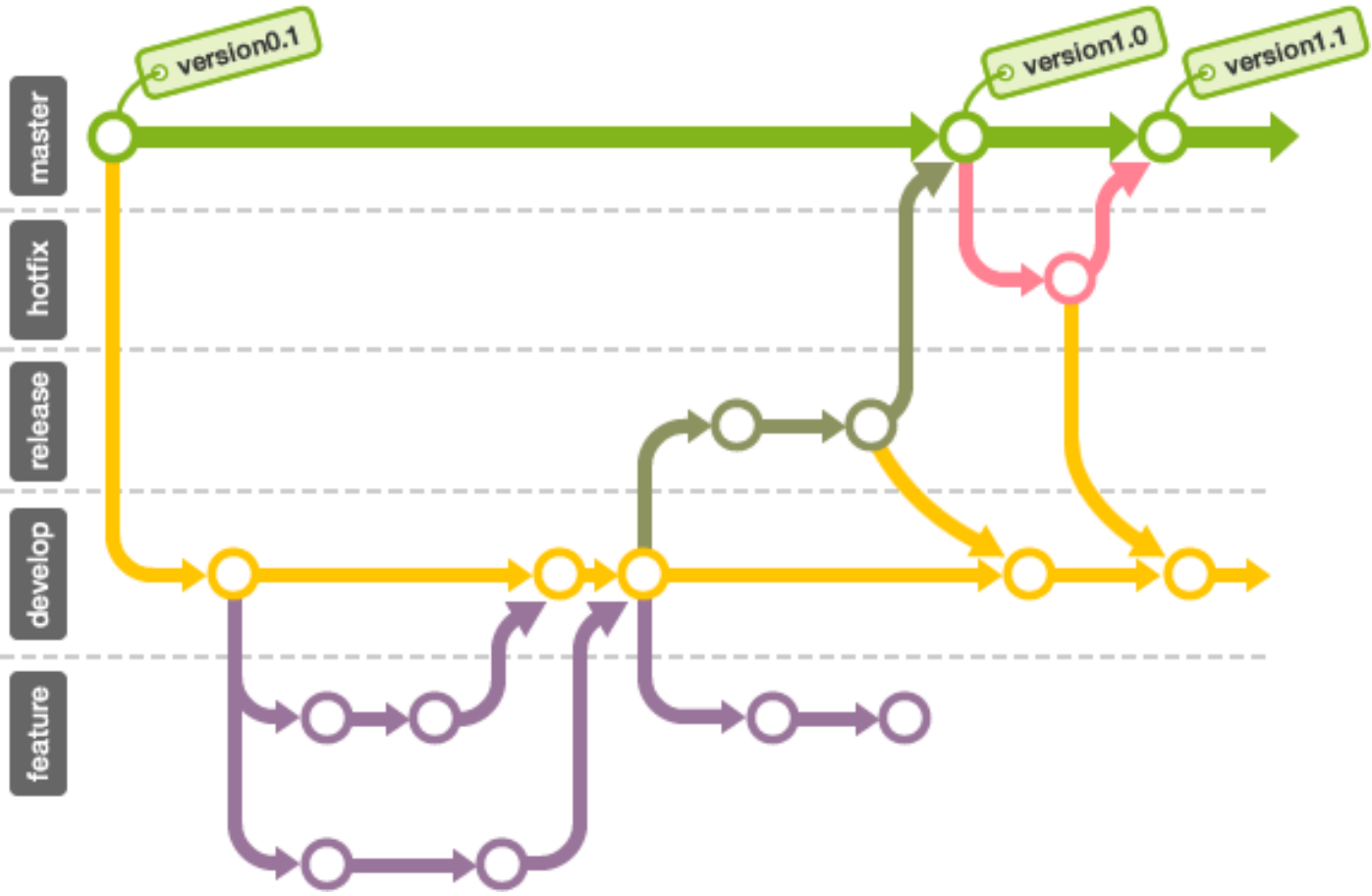
- GitHub : 깃허브라고 읽고, Git으로 관리하는 프로젝트를 올려둘 수 있는 사이트입니다.
- GUI : 그래픽 유저 인터페이스, 즉 마우스로 클릭해서 사용하는 방식입니다.
- CLI : 커맨드 라인 인터페이스, 즉 명령어를 하나씩 입력하는 방식입니다.
- Git Bash : CLI 방식으로 Git을 사용할 수 있는 환경입니다.
- **커밋** : 버전 관리를 통해 생성된 파일, 혹은 그 행위를 의미합니다.
- **log 명령어** : 지금까지 만든 커밋을 모두 확인합니다.
- **체크아웃한다** : checkout으로 원하는 지점으로 파일을 되돌릴 수 있습니다. 타임머신과 같죠.
- **로컬저장소** : Git으로 버전 관리하는 내 컴퓨터 안의 폴더를 의미합니다.
- **원격저장소** : GitHub에서 협업하는 공간(폴더)를 의미합니다.
- **레포지토리** : 원격저장소를 의미합니다. (좀 더 멋있어 보이는 용어입니다.)
- **푸시** : 로컬저장소의 커밋(버전 관리한 파일)을 원격저장소에 올리는 것입니다. upload
- **풀** : 원격저장소의 커밋을 로컬저장소에 내려받는 것입니다. download


Git / GitHub
How to "Pull Request"
Your PC
Your GitHub
1
2 add
3 commit
4 push
7 merge
2 clone
1 fork
7 pull request
Partner's PC
Partner's GitHub
3
4 add
5 commit
6 push
designlass.net


workspace
staging
Local repository
Remote repository
git add/mv/rm
git commit
git commit -a
git reset <file>
git push
git reset <commit>
git fetch
git diff
git diff HEAD
git clone/pull


팀원 A 수정
팀원 B 수정
00 버전
01 버전
02 버전
팀원 C 수정
02 버전

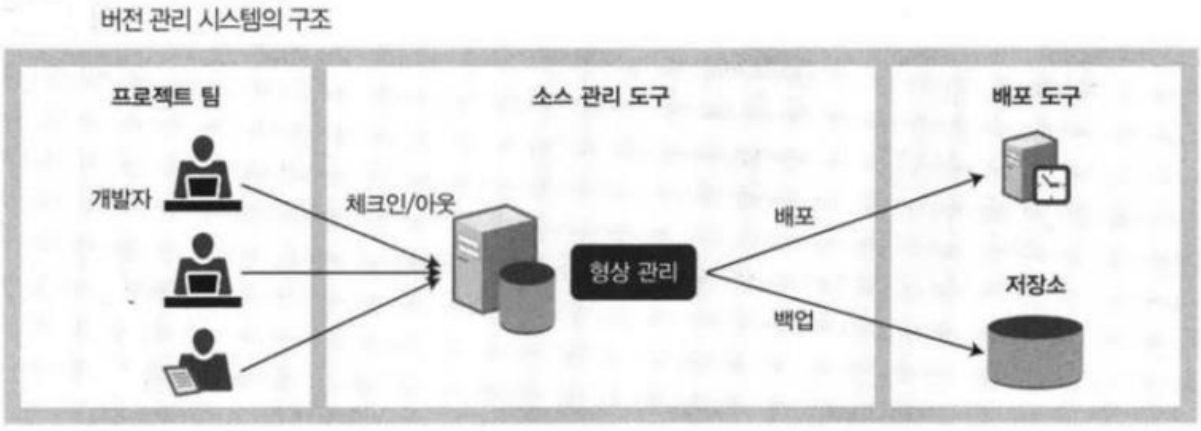

Break Your Changes Down
& Make as Many Pull Requests as You Like
MASTER
F
P
R
S
Feature Branch
Pull Request
Sub Branch
Review & Merge

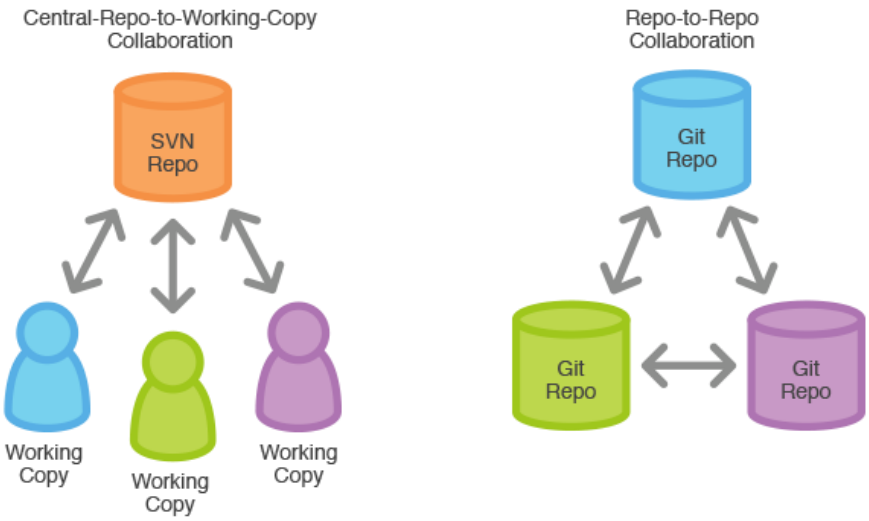


▶소스 형상관리의 정의

소프트웨어 형상관리는 Software Configuration Management, 줄여서 SCM 라는 단어를 쓰기도 하는데, SW 개발 및 유지보수 과정에서 발생하는 소스코드, 문서, 인터페이스 등 각종 결과물에 대해 형상을 만들고, 이들 형상에 대한 변경을 체계적으로 관리, 제어하기 위한 활동이다. 단순히 말하자면 프로젝트를 진행하면서 생성하는 소스코드를 CVS 나 SVN, 또는 GIT 와 같은 버전 관리 시스템을 이용하는 것을 말한다. 다수의 개발자가 프로젝트에서 동일한 기능을 동시에 개발한다고 할 때, 작성된 소스 코드와 변경사항을 확인하고, 수정하는 협업을 도와주는 시스템이라고 할 수 있다.

형상관리는 일반적으로 버전 관리 (version control, revision control), 소스 관리 (source control), 소스 코드 관리 (source code management, SCM)와 동일한 의미로 사용된다.





▶대표적인 소스 형상 관리 툴

1. GitHub

2. SVN

SVN 은 SubVersion 의 줄임말로 형상관리/소스관리 툴의 일종으로 Open Source 버전관리 시스템으로 2000 년도에 CSV 를 대체하기 위해 개발되었습니다. 파일 및 디렉토리의 버전관리 기능을 제공하며, 버전의 파일트리는 한 곳에 집중된 레파지토리에서 관리된다. Subversion 은 효율적인 Branch 및 Merge 기능과 작업의 무결성을 보장하고 네트워크 기능 지원(http) 및 크로스 플랫폼 (Windows, Mac, Linux) 을 지원한다. GIT 과 함께 굉장히 많이 쓰이는 소스형상관리 툴이다.

3. SourceSafe

마이크로소프트에서 개발한 프로그램으로 풀네임(Microsoft Visual SourceSafe, VSS)으로 현재는 개발이 중단된 소스 관리 프로그램으로 조그마한 소프트웨어 개발 프로젝트를 대상으로 한다. 대부분의 소스 제어 시스템들처럼 소스세이프는 컴퓨터 파일의 가상 라이브러리를 만든다.