**Git & GitHub**

1. 버전 관리 시스템이란

조별 과제로 리포트를 작성할 때 ‘~Final.dox’, ‘~진짜 최종.dox’, ‘~최종의 최종.dox’등과 같이  저장하거나 프로그램 코딩을 할 때 오류가 날 때를 대비하여 소스 사본을 백업하는데  하나하나의 저장파일이나 백업을 개별 버전으로 간주하고 그걸 관리하는 방법이다. 즉, 버전 관리 시스템은 사본 생성, 보존, 복원을 한 번에 해 줄 수 있는 도구이다.

1. 버전 관리 시스템의 종류
   1. 클라이언트-서버 모델

하나의 중앙 저장소를 공유한 후 각각의 클라이언트(개발자)는 저장소의 일부분만 갖는 형태이다. 즉 자신이 작업하는 부분만 로컬에 임시로 저장한 후 작업하는 형태.

* 1. 분산 모델

프로젝트에 참여하는 모든 클라이언트가 전체 저장소에 대한 개별적인 로컬 저장소를 갖고 작업하는 형태이다. ‘클라이언트-서버’모델과 달리, ‘분산’모델의 클라이언트는 각자가 온전한 전체 저장소의 사본으로 로컬에 가지게 된다.

* 1. CVS

Concurrent Versions System’을 의마하는 말로 클라이언트-서버 방식의 버전관리 시스템이다.

공식 웹 사이트 : [**https://savannah.nongnu.org/projects/cvs**](https://savannah.nongnu.org/projects/cvs)

* 1. 서브버전

CVS의 여러 단점을 개선한 ‘클라이언트-서버’모델의 자유 소프트웨어 버전 관리 시스템이다.

공식 웹 사이트 : [**https://subversion.apache.org/**](https://subversion.apache.org/)

* 1. 머큐리얼

현재 구글에서 이 시스템의 관리를 지원하고 있는 분산 모델의 버전 관리 시스템이다.  우리가 사용할 Git은 필요한 기능을 골라서 사용하지만 머큐리얼은 버전 관리 시스템에 필요한 모든 기능을 한 번에 통합해서 제공한다. 파이썬으로 개발되었다는 특징이 있다.

1. Git
   1. Git의 장점

* 전 세계의 수많은 사용자가 사용 중
* git을 사용한 저장소를 공유 사이트인 GitHub 웹 사이트의 존재
* 사용자 수에서 나오는 어마어마한 숫자의 튜토리얼과 프로젝트가 존재
  1. Git의 특징
* 로컬 및 원격 저장소 생성
* 로컬 저장소에 파일 생성 및 추가
* 수정 내역을 로컬 저장소에 제출
* 파일 수정 내역 추적
* 원격 저장소에 제출된 수정 내역을 로컬 저장소에 적용
* master에 영향을 끼치지 않는 브랜치 생성
* 브랜치 사이의 병합(Merge)
* 브랜치를 병합하는 도중의 충돌 감지

**Chapter 1. Git과 Github 설명**

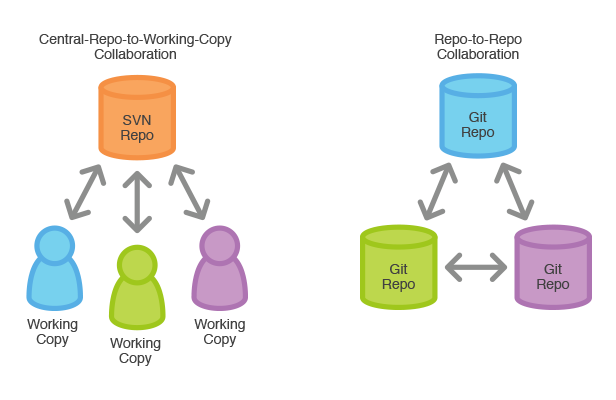
* **Git**
  + Git은 **분산 버전 관리 시스템이다**.
    - Git으로 프로젝트를 개발하는 사람은 모두 현재 상태의 파일뿐만 아니라 그 프로젝트의 전체 이력을 가지고 있게 된다는 뜻이다.
    - 버전관리 시스템 : 파일의 변경 내역을 계속 추적하도록 개발된 소프트웨어
* **GitHub**
  + GitHub는 Git 저장소(repository)를 업로드 할 수 있는 **호스팅** **웹사이트**를 말한다.
  + GitHub는 다른 사람들과의 협업을 매우 용이하게 해준다.
    - **리포지토리를 공유할 수 있는 원격저장소**
      * 웹 기반 인터페이스
      * forking
      * pull requests
      * issues
      * wikis
  + Github는 위와 같은 기능을 제공하여 팀원들과 보다 효율적으로 변경안을 구체화하고 토론하며 검토할 수 있게 해준다.

**Git의 장점**

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

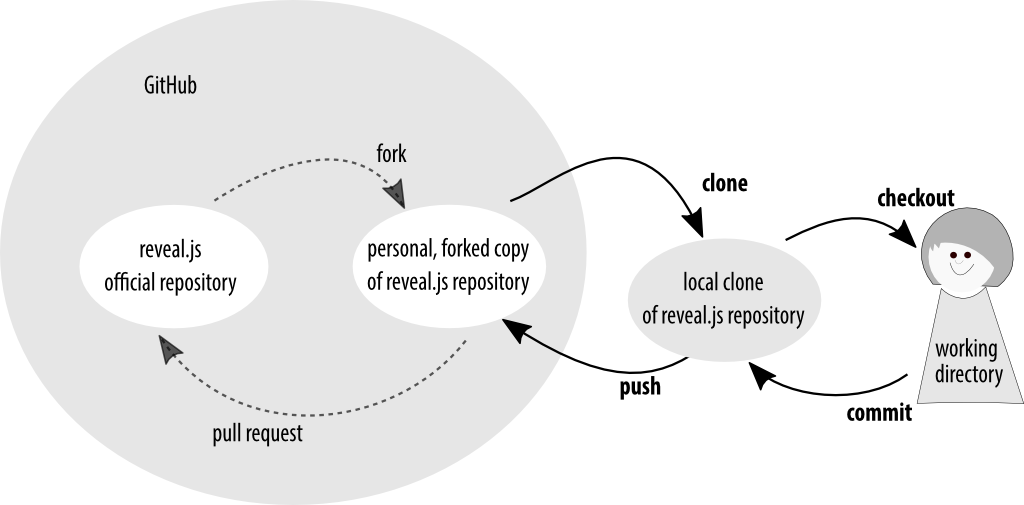
* **변경취소 가능** : 실수 했을 때 구 버전의 작업 파일로 돌아갈 수 있음.
* **모든 변경에 대한 완벽한 이력 History** : 짧게는 하루, 길게는 1년 전에 프로젝트가 어떤 형태였는지도 알 수 있다.
* **변경한 이유를 기록** : 협업을 하다보면(때로는 내가 작성한 코드라할지라도) 왜 변경했는지 모르겠을 때가 많다. 이때 Git의 커밋 메시지(**commit message**)를 이용하면 변경한 이유를 쉽게 기록할 수 있으며, 추후에 참조할 수 있다.
* **변경에 대한 확신** : 이전 버전으로의 복귀가 쉽기 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** : Git을 이용하면 여러사람이 동시에 같은 파일을 작업할 수 있다. Git은 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.

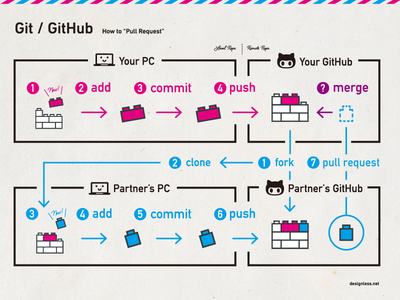
|  |
| --- |
| **▶소스 형상관리의 정의**  소프트웨어 형상관리는 Software Configuration Management, 줄여서 SCM라는 단어를 쓰기도 하는데, SW개발 및 유지보수 과정에서 발생하는 소스코드, 문서, 인터페이스 등 각종 결과물에 대해 형상을 만들고, 이들 형상에 대한 변경을 체계적으로 관리, 제어하기 위한 활동입니다. 단순히 말하자면 프로젝트를 진행하면서 생성하는 소스코드를 CVS나 SVN, 또는 GIT와 같은 버전 관리 시스템을 이용하는 것을 말합니다. 다수의 개발자가 프로젝트에서 동일한 기능을 동시에 개발한다고 할 때, 작성된 소스 코드와 변경사항을 확인하고, 수정하는 협업을 도와주는 시스템이라고 할 수 있습니다.  형상관리는 일반적으로 버전 관리 (version control, revision control), 소스 관리 (source control), 소스 코드 관리 (source code management, SCM)와 동일한 의미로 사용됩니다. 즉, 동일한 정보(프로그램)에 대한 여러 버전을 관리하는 것으로, 소프트웨어 공학에서는 팀 단위로 개발 중인 소스 코드나 청사진(설계도) 등 디지털 문서의 작업 단계별 버전을 관리하는 작업으로 정의됩니다. 소스 형상관리 툴의 대략적인 구조는 아래와 같습니다.  버전관리시스템  **▶대표적인 소스형상관리 툴**  깃  **1. GitHub**  [깃허브 홈페이지](https://github.com/)  '깃(Git)'은 2005년 리눅스를 만든 리누스 토발즈와 주니오 하마노가 개발한 분산형 버전관리 시스템입니다. 어떤 코드를 누가 수정했는지 기록, 추적을 할 수 있는 오픈 소스 소프트웨어로 수천명의 사람들이 이용해도 안정적이고, 속도가 빠르다는 장점이 있습니다. 깃허브가 나오기전에는 깃이라는 프로그램을 사용하였는데요. '깃'은 명령어를 입력하면서 이용해야되는 불편함이 있었습니다. 이 불편함을 더욱 편리하게 이용할 수 있게 만든게 바로 '깃허브'입니다. 깃허브는 개발자들끼리 소스를 공유할 수 있는 오픈소스 페이지(개발자 스스로 본인의 소스를 다른사람들에게 공개하여 필요한 정보를 공유할 수 있는 자유로운 공간) , SNS기능, 블로그 기능까지 추가하는 등 소스형상관리기능에서 개발 플랫폼으로 점점 진화하고 있습니다. 개발자들이 굉장히 많이 사용하고 친숙한 형상관리툴입니다. GitHub의 상징 옥토캣 캐릭터도 인기가 굉장히 많죠. 단점이라면 다른 툴에 비해 다소 사용하기 무겁고, 어렵다는 단점이 있습니다. (익숙해지면 쉽습니다.)  svn  **2. SVN**  SVN은 SubVersion의 줄임말로 형상관리/소스관리 툴의 일종입니다. Open Source 버전관리 시스템으로 2000년도에 CSV를 대체하기 위해 개발되었습니다. 파일 및 디렉토리의 버전관리 기능을 제공하며, 버전의 파일트리는 한 곳에 집중된 레파지토리에서 관리됩니다.. Subversion은 효율적인 Branch및 Merge기능과 작업의 무결성을 보장합니다. 네트워크 기능 지원(http) 및 크로스 플랫폼 (Windows, Mac, Linux) 을 지원합니다. GIT과 함께 굉장히 많이 쓰이는 소스형상관리 툴입니다. Git하고 SVN이 거의 머 양대산맥이죠 ^^; Git보다 사용하기 가벼우며 핵심적인 기능만 딱딱 제공합니다. 초보자들이 사용하기에도 Git보다는 간편할듯 하군요.  sourcesafe  **3. SourceSafe**  마이크로소프트에서 개발한 프로그램으로 풀네임은(Microsoft Visual SourceSafe, VSS)입니다. 현재는 개발이 중단된 소스 관리 프로그램으로 조그마한 소프트웨어 개발 프로젝트를 대상으로 합니다. 대부분의 소스 제어 시스템들처럼 소스세이프는 컴퓨터 파일의 가상 라이브러리를 만듭니다. 소스 코드에 가장 흔히 쓰이는 소스세이프는 데이터베이스내에서 어떠한 종류의 파일도 처리할 수 있지만 구 버전의 경우 이미지와 같은 텍스트가 아닌 많은 양의 데이터나 컴파일된 실행 파일을 저장할 때 불안정한 것으로 알려져 있습니다. SourceSafe는 마이크로소프트에서 만들었기에 비쥬얼 스튜디오와 호환이 좋습니다. C언어 계열 소스 형상관리를 할때 굉장히 많이쓰입니다.  **▶소스 형상관리는 왜 해야할까?**  방대한 양의 개발을 혼자서 하기에는 무척이나 힘이듭니다. 고로 개발자들간의 협업은 매우 중요하죠. 공동으로 개발작업을 하다보면 서로가 수정한 소스를 최신 버전으로 공유하여 개발을 진행하여야 합니다. 간단한 프로그램이라면 그냥 카카오톡이나 메일로 소스코드를 주고받으면서 할수도 있겠지만 이것은 소스의 양이 많아지고 페이지가 많아질수록 비효율적이게 됩니다. 또한 형상관리툴을 사용하므로써 얻을수있는 많은 이점들을 얻을 수 없습니다. 그래서 형상관리툴을 활용하여 프로그램 소스라면 특정 저장소(레파지토리)에 저장해놓고 본인이 수정한 최신 소스를 업로드하면서 다른 사람의 최신버전을 내려받아 개발을 진행하는 것이 필요한데, 이를 형상관리라고 합니다. 소스의 형상(모양)을 관리한다는 것 말 그대로 전체적인 틀을 유지한다는 것입니다.  **▶소스 형상관리 툴 사용시 얻는 장점**  **1.** 소스 코드를 프로젝트 팀원 및 관계자들과 공유할 수 있다.  **2.**소스 코드의 변경 이력을 관리할 수 있다.  **3.**서버나 클라이언트에 배포할 때에도 유용하게 사용된다.  **4.** 여러 사람이 동일한 소스 코드를 공유해서 개발할 수 있으며 소스 코드를 공유할 때 생기는 버전 충돌 문제를 해결할 수 있다.  **5.**장애 혹은 기능상 필요할 때 이전 버전으로 소프트웨어를 원상복구할 수 있다.  **6. 동일한 소프트웨어를 여러 개의 버전으로 분기해서 개발할 필요가 잇는 경우에 유용하게 사용된다.** |



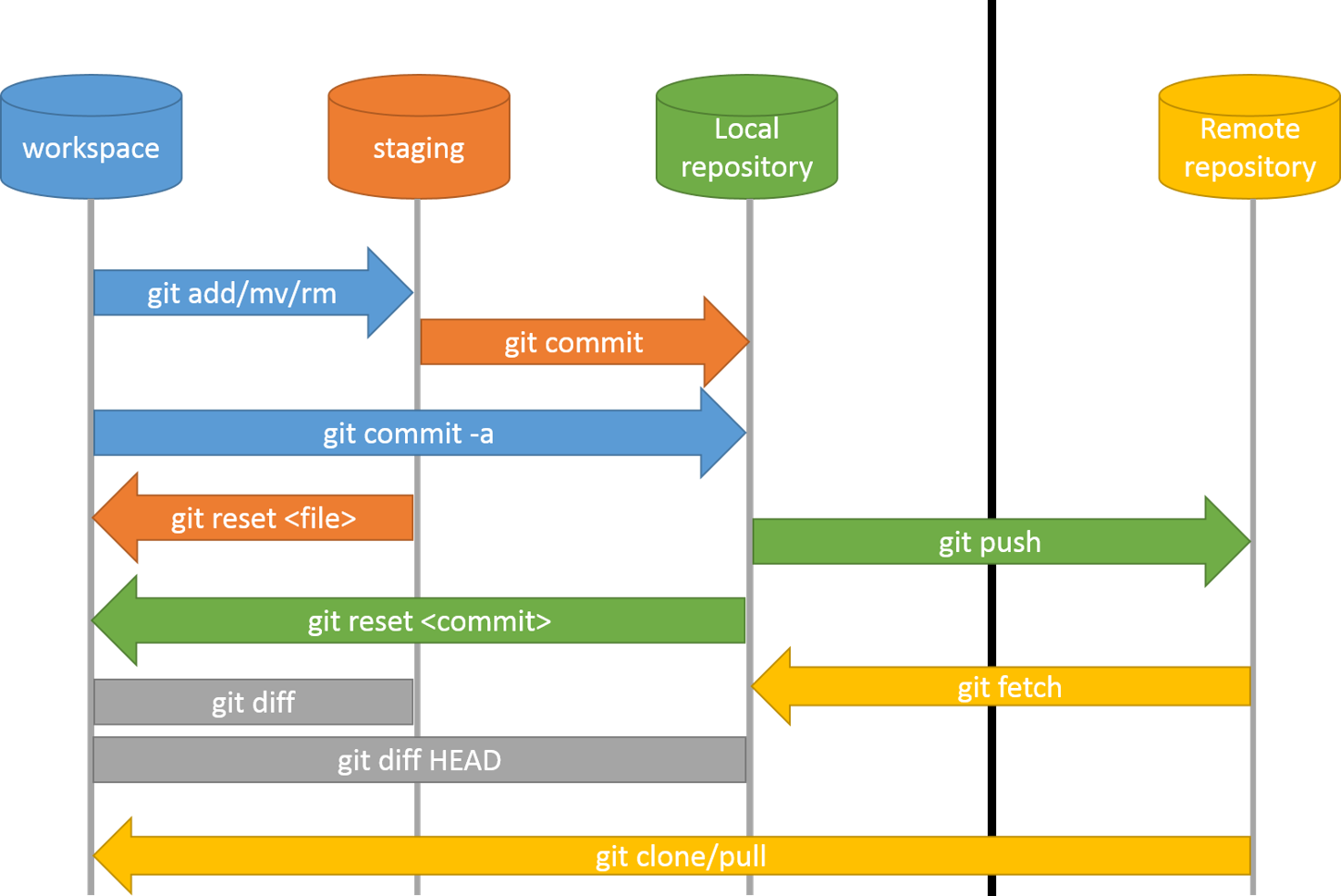
**Git :: GitHub 기본개념**

|  |
| --- |
| Git은 버전 관리 툴이며, Github은 웹 상에 소스 코드를 올려서 여러 사람과 공유하는 원격저장소다.  자신의 PC에서 작업하는 공간을 Local Repository라 하며, Github에 있는 공간을 Remote Repository라 한다 |







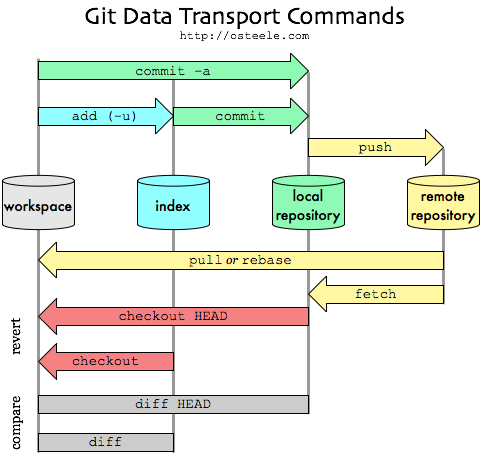


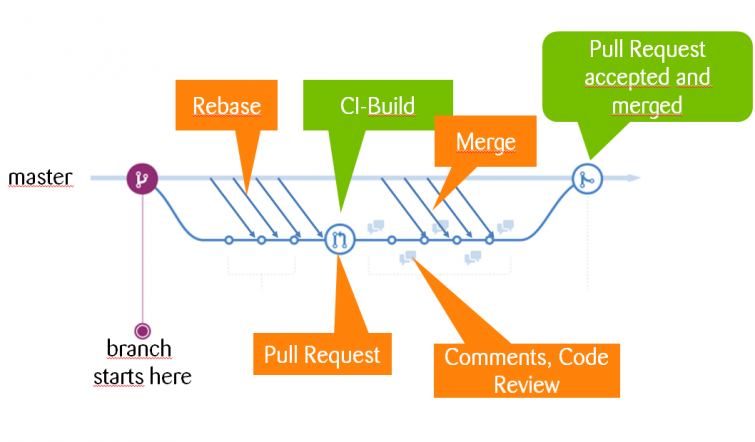


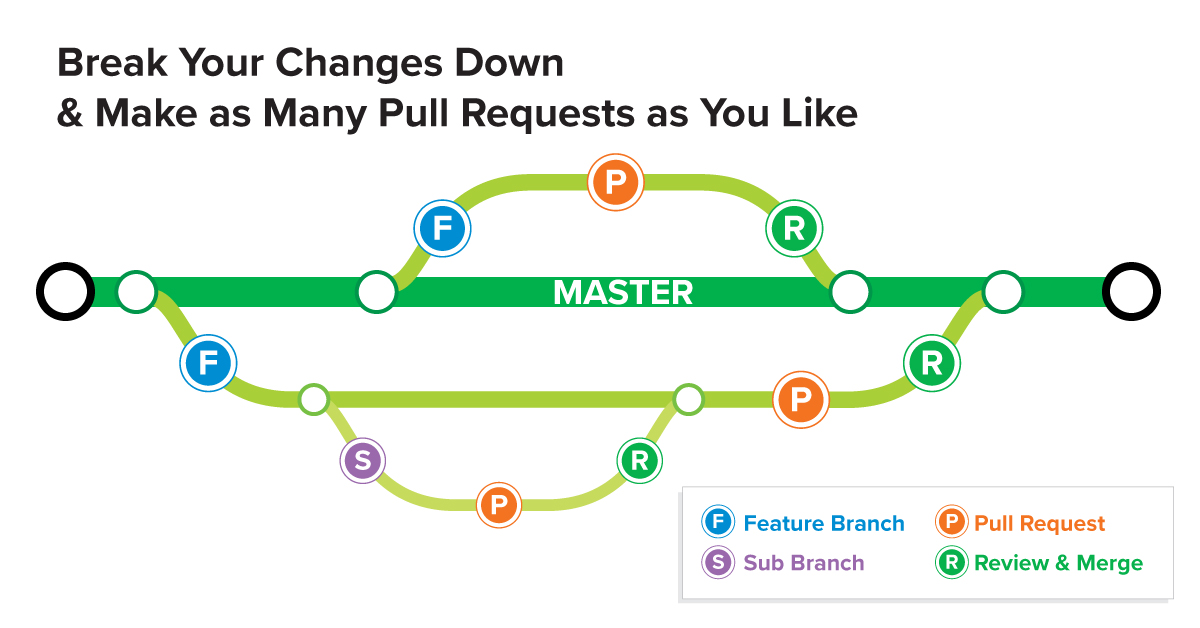
.

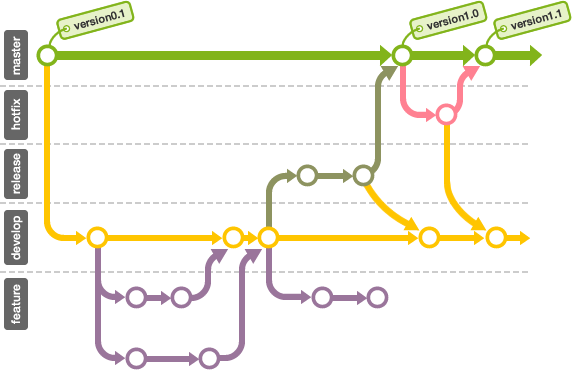
|  |
| --- |
| 로컬 저장소는 git이 관리하는 세 그루의 나무로 구성되어 있다.  첫번째 나무인 작업 디렉토리(Working directory)는 로컬(실제) 파일들로 이루어져있고, 두번째 나무인 인덱스(Index)는 준비 영역(**staging area)**의 역할을 하며, 마지막 나무인 HEAD는 최종 확정본(commit)을 나타낸다.  https://t1.daumcdn.net/cfile/tistory/2171D43E56ADC9F018 |

..



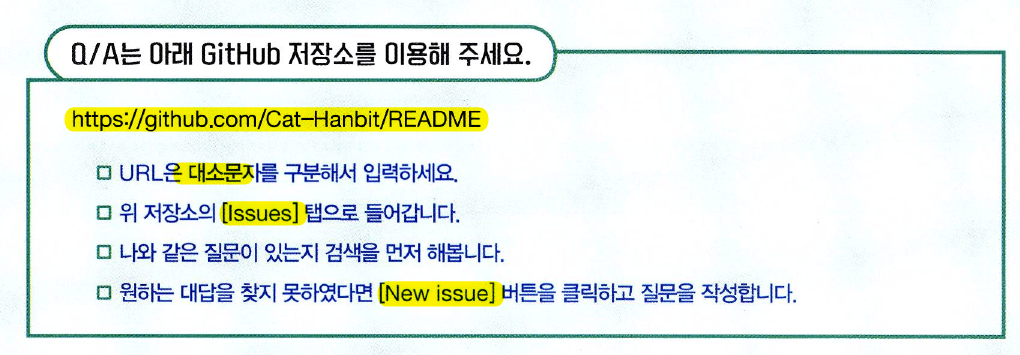
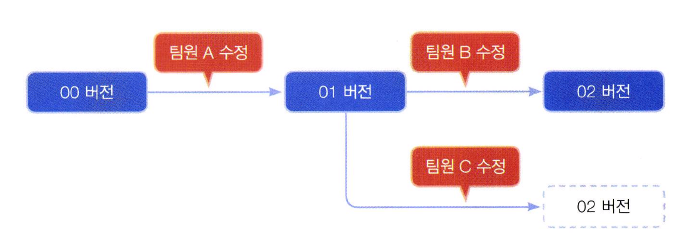




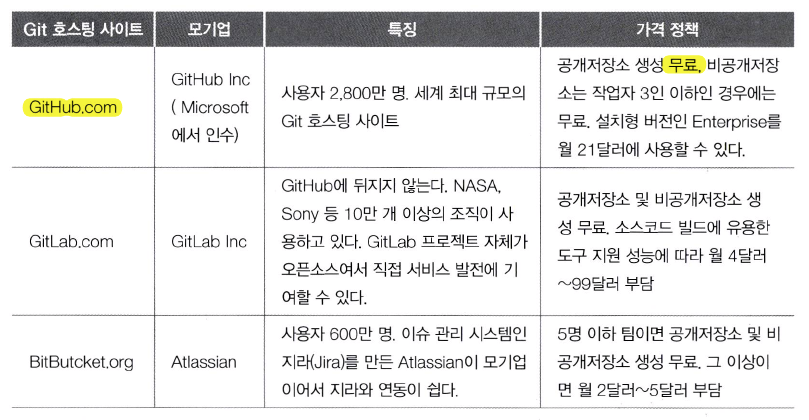


-버전관리?

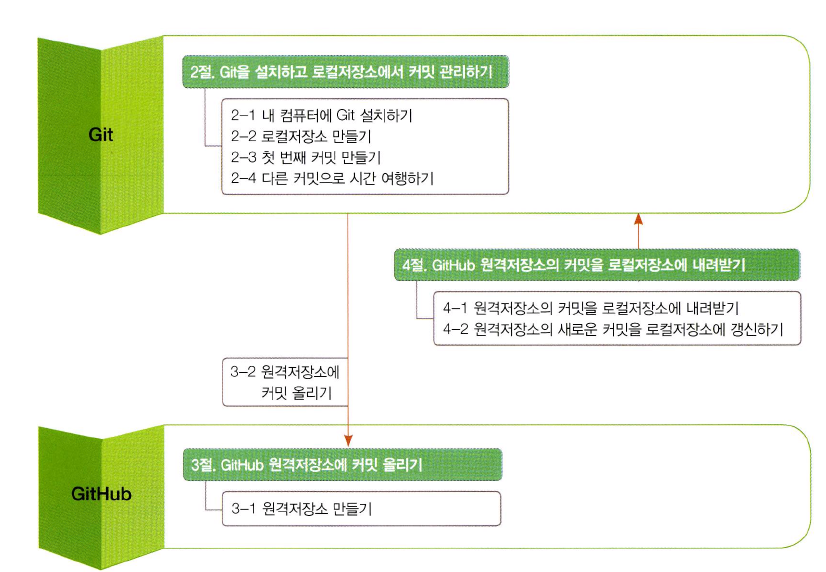
<https://docs.github.com/en>



-Git(소스버전 관리 시스템), GitHub(Git으로 관리하는 프로젝트를 올려둘수 있는 Git호스팅 사이트)



-GitHub가입하기



31페이지

