git & GitHub

Let's build from here, together

WHY ?

나 : 최종의 진짜 라스트 완벽한 수정본의 최종의 정말 진짜 마지막...

??? : 아 석민님 저번에 보여주신거 그거 있죠? 그걸로 다시 수정해 주세요.

나 였던 것 : 예??


git


GitHub

git & GitHub

- `Git`은 소스 코드 버전 관리 시스템을 뜻합니다. Git은 소스 코드 버전 사이를 오가는 시간 여행 이상의 기능을 제공합니다.
- `Git`은 데이터를 저장할 공간만 있다면 어디서나 사용할 수 있습니다. 로컬 저장소인 개인 컴퓨터, USB에 저장한 프로젝트도 마찬가지입니다. 만약 드롭박스, 구글 드라이브와 같은 클라우드 서버에 올려둔다면 어떨까요? 팀 프로젝트를 진행하는 다른 팀원과 함께 인터넷을 통해 ***버전 관리*** 를 할 수 있을 겁니다.
- 이렇게 Git으로 관리하는 프로젝트를 올려둘 수 있는 대표적인 Git 호스팅 사이트 중 하나가 바로 `GitHub` 입니다. 블로그를 만들 수 있는 곳이 네이버, 다음, 워드프레스 등 다양한 것처럼 Git으로 관리하는 프로젝트를 올릴 수 있는 사이트도 GitHub뿐 아니라 GitLab, BitBucket 등 다양합니다. 우리는 그중 GitHub에 대하여 알아 보겠습니다.

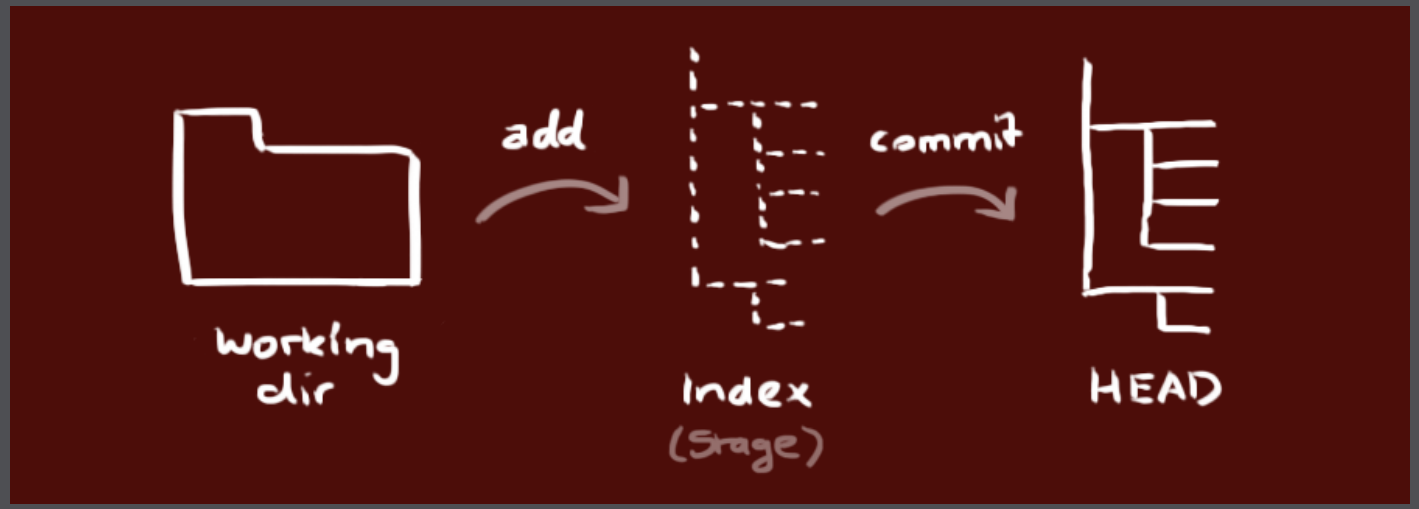
버전 관리

- 버전관리 시스템은 파일변화를 시간에 따라 기록했다가 이후 특정 시점의 버전을 다시 꺼내올 수 있는 시스템입니다.

  - 각 파일을 이전 상태로 되돌릴 수 있다.
  - 프로젝트를 통째로 이전 상태로 되돌릴 수 있다.
  - 시간에 따라 수정 내용을 비교해 볼 수 있다.
  - 누가 문제를 일으켰는지도 추적할 수 있다.
  - 파일을 잃어버리거나 잘못 고쳤을 때도 쉽게 복구할 수 있다.

Repository

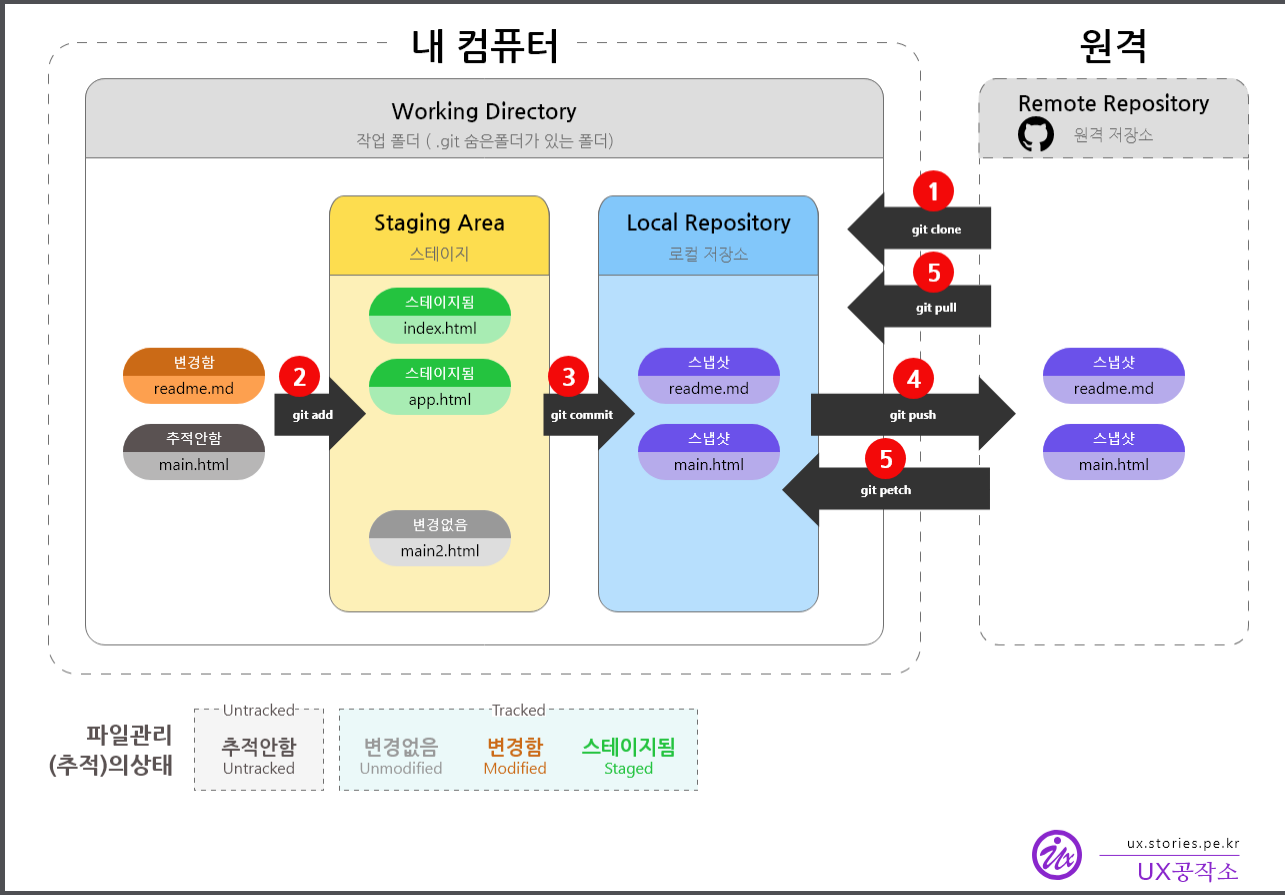
- Git의 Repository 구조는 크게 세가지로 구성되어 있습니다.
- 작업폴더(Working directory) > 인덱스(Staging Area) > 저장소(Head - Repository)
- 우리가 작업하는 폴더를 작업트리(Working directory) 라고 부르며 commit을 실행하기 전에 작업트리와 저장소 사이에 존재하는 가상의 준비 영역(Staging Area)을 인덱스(Index)라고 합니다.



GitHub

- Repository : 저장소
  - local : 로컬 저장소
  - remote : 원격 저장소

- Working Directory : 작업이 이루어지고 있는 공간
- Staging Area : 깃에 추가한 파일이 기록되는 공간 (SnapShot to index)



Git flow

1. git clone
   서버에서 파일을 Clone 하게 되면 내 컴퓨터의 지정된 폴더에 `.git` 이라는 숨겨진 폴더가 생성되고 이 .git폴더를 가지고 있는 폴더가 작업 폴더(Working Directory)가 됩니다. 이 작업 폴더는 자동으로 서버와 링크가 맺어지게 됩니다.

2. git add
   작업 폴더에 처음 파일을 생성한다면 이것은 아직 관리대상이 아닙니다. 관리대상이 아닐 경우는 아무런 히스토리 정보를 가지고 있지 않습니다. 이제 이 파일을 관리대상으로 삼기 위해서는 `git add` 명령어를 실행해 줘야 합니다. git add 명령어를 실행하면 이 파일이 스테이지(Staging Area) 로 올라가게 됩니다. 이제부터 이 파일의 수정, 삭제 등의 모든 정보는 Git에 기록이 되어집니다.

   파일의 상태는 추적안함(Untracked), 수정함(Modified)이 스테이지(Staged)로 변경됩니다.

Git flow

3. git commit
   이제 스테이징된 파일들을 로컬 저장소(Local Repository)로 등록을 해야 합니다. 그러기 위해서 `git commit`을 합니다. git commit을 하면 현재 스태이징된 파일들을 그대로 스냅샷으로 찍어서 로컬 저장소(Local Repository)에 보관하게 됩니다. 이 스냅샷이 하나의 히스토리 기록이 되는 것입니다. 이 기록을 기준으로 과거로 되돌아갈 수도 있고 미래로 갈 수도 있고 다른 브랜치로도 이동할 수 있는 기준점이 됩니다.

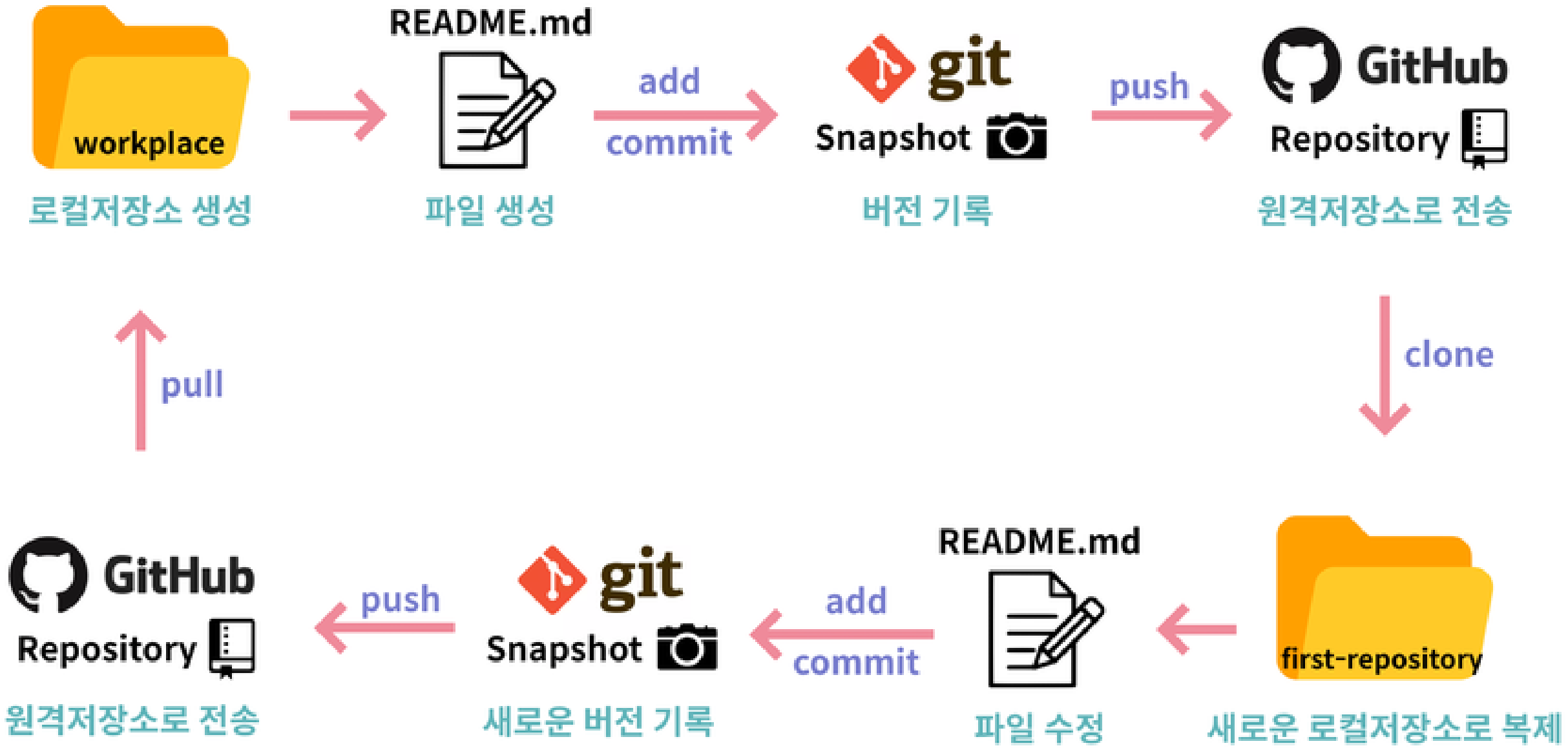
   파일의 상태는 스테이지(Staged)가 수정없음(Unmodified)으로 변경됩니다.

Git flow

4. git push
   git commit으로 로컬 저장소(Local Repository)에 스냅샷을 찍어 이용하는 것은 내 컴퓨터에서만 가능한 상태입니다. 이것을 서버에 올려서 다른 사람들과 공유를 하기 위해서는 원격 저장소(Remote Repository)에 업로드를 해야 합니다.

   그 명령어가 `git push` 입니다. git push를 하면 로컬 저장소의 커밋(git commit)된 모든 내용이 그대로 원격 저장소로 올라갑니다.

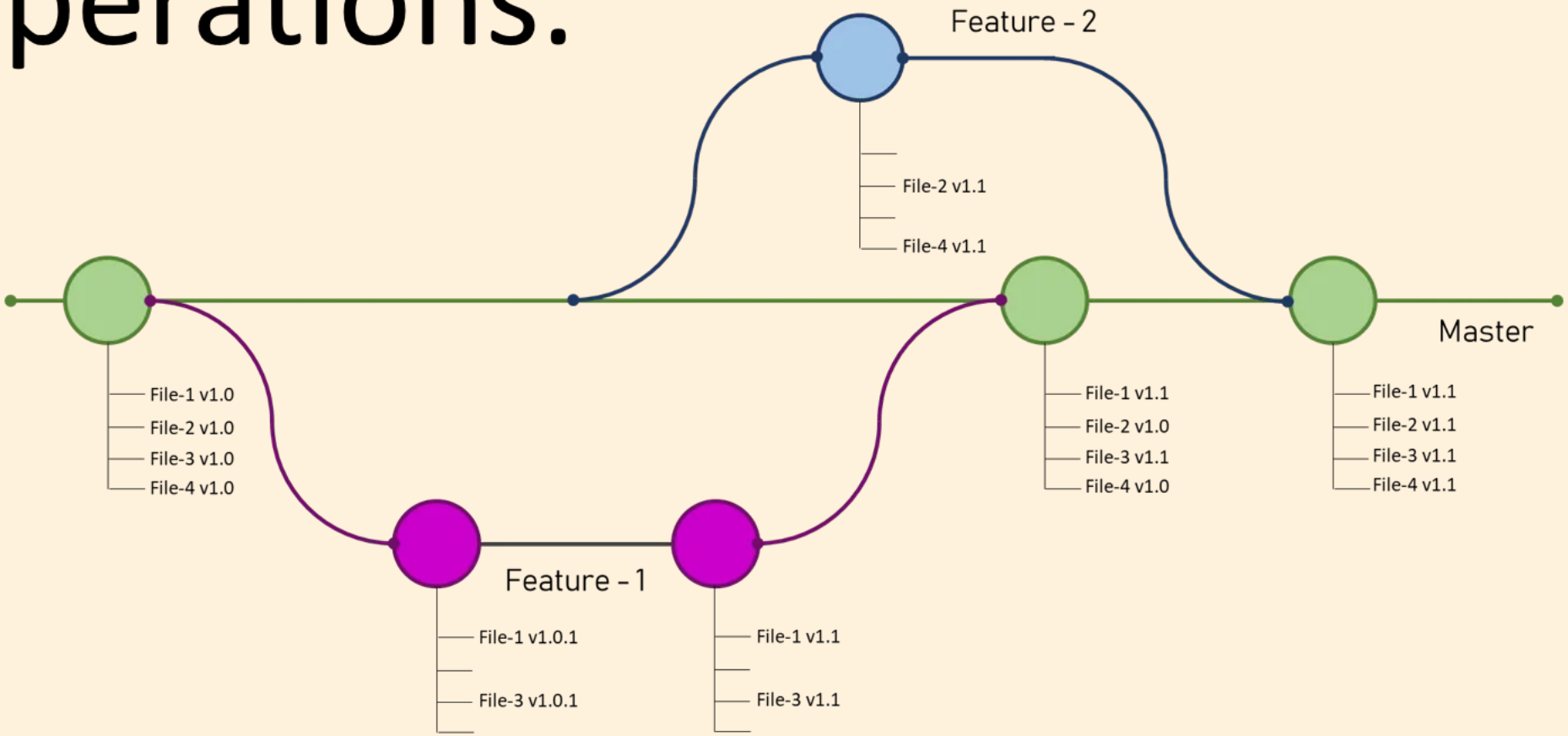
5. git pull
   원격 저장소에 올라온 수정된 최신 파일을 내 로컬 저장소로 업데이트를 해야 할 필요가 있습니다. 그때 필요한 명령어가 `git pull` 입니다. 일단 한번은 서버와 링크가 맺어있어야 실행이 됩니다. 또는 git fetch 명령어를 사용할 수도 있습니다.


workplace
로컬저장소 생성
README.md
파일 생성
add
commit
git
Snapshot
버전 기록
push
GitHub
Repository
원격저장소로 전송
clone
first-repository
새로운 로컬저장소로 복제
README.md
파일 수정
add
commit
git
Snapshot
새로운 버전 기록
push
GitHub
Repository
원격저장소로 전송
pull

Branch

- Head
  현재 작업하고 있는 브랜치의 마지막 커밋을 가리키는 포인터
- Branch
  분기점을 의미하며, 작업을 할 때 현재 상태를 복사하여 Branch에서 작업을 한 후에 완전하다 싶을 때 Merge 하여 작업합니다.
- Merge
  다른 Branch의 내용을 현재 Branch로 가져와 합치는 작업을 의미합니다.
- rebase
  head를 최신으로 이동시켜 그 뒤로 작업 내용(commit)을 이어붙입니다. rebase 한다고 해서 main과 바로 동기화 되지 않기 때문에 rebase 해서 붙은 포인트는 끝에 있지만, main branch는 아직 rebase 전 상태이기 때문에 merge를 통해서 끝 포인트로 이동시키는 것을 말합니다.

GIT Branch and its Operations.



GUI

- git을 사용하려 cmd 혹은 터미널로 모든 것을 제어하기엔 아직 우리는 부족 합니다.
  - 숙련된 사용자가 아니라면 레포지토리에 직접 접근하여 모든 구문을 코드로만 작성해야하는 불편함이 있습니다.
  - 이론적으로 충분히 숙달 된 이후에 머리속으로 모든 시그널 플로우를 정립했다면 추가로 프로그램을 사용하지 않고 기본 명령창에서 충분히 사용 가능합니다.
- 따라서 초보자가 git을 이용하여 협업하고 작업을 이해하기 쉽도록 GUI 프로그램을 사용한다면 팀 프로젝트에서도 좋은 시너지를 낼 수 있다고 생각합니다.

link

gitHub : https://github.com/

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

sourcetree : https://www.sourcetreeapp.com/