[Git 개념] Git 주요 용어 및 구성

1. Git

Git은 버전관리를 위한 프로그램(Version Control System)입니다.

버전 관리란 예를 들면 매일 회의를 한다고 가정했을 때,
엊그제 회의한 내용을 회의록-2019.04.18.txt 파일에 저장하고
어제 회의한 내용을 회의록-2019.04.19.txt 파일에 저장하고
오늘 회의한 내용을 회의록-2019.04.20.txt 파일에 저장해야 합니다.
이 때, 각 날 마다 회의한 내용을 버전이라 하며, Git은 이 버전들을 효율적으로 관리할 수 있게 도와줍니다.

즉, Git을 사용하면 여러 파일을 생성할 필요없이 회의록.txt 하나의 파일에 버전으로 남기고 싶은 내용을 commit하면 됩니다.
그러면 필요에 따라 버전을 바꿔서 특정 버전으로 이동해서 그 날의 회의 내용을 볼 수 있죠.

뿐만 아니라 Git을 활용하면 **협업이 가능**합니다.
Git을 사용하면 깃헙같은 Remote repository에 소스 코드를 올려서,
본인원 소스 코드와 동료의 소스 코드를 하나의 소스 코드로 합칠 수 있으며, 각각 독립적으로 버전을 관리할 수도 있습니다.

Git을 사용하다 보면, 자연스럽게 이해가 될 내용이니 지금 당장 이해가 안되도 상관없습니다.
일단은 "**Git은 버전 관리, 협업을 가능할 수 있게 해준다!**" 정도로 기억하면 좋을것 같습니다.

2. Git 주요 용어 및 구성

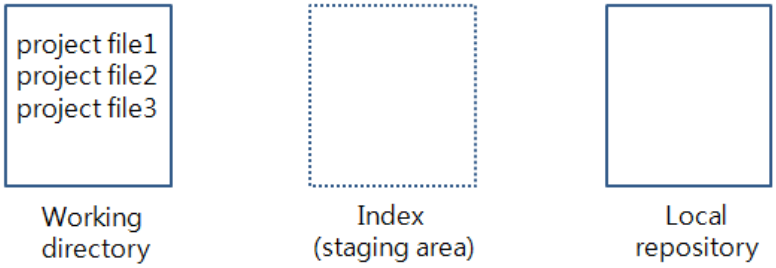
Git의 주요 용어들과 Git이 어떻게 구성되어 있는지 Flow를 그림으로 살펴보도록 하겠습니다.

결국 Git을 사용하려면 명령어가 핵심이지만, 영역과 Flow를 알아야 명령어를 실행했을 때 동작 과정을 상상해볼 수 있습니다.

이 용어들은 이후의 글들에서 다룰 목차이기도 합니다.

1) 영역

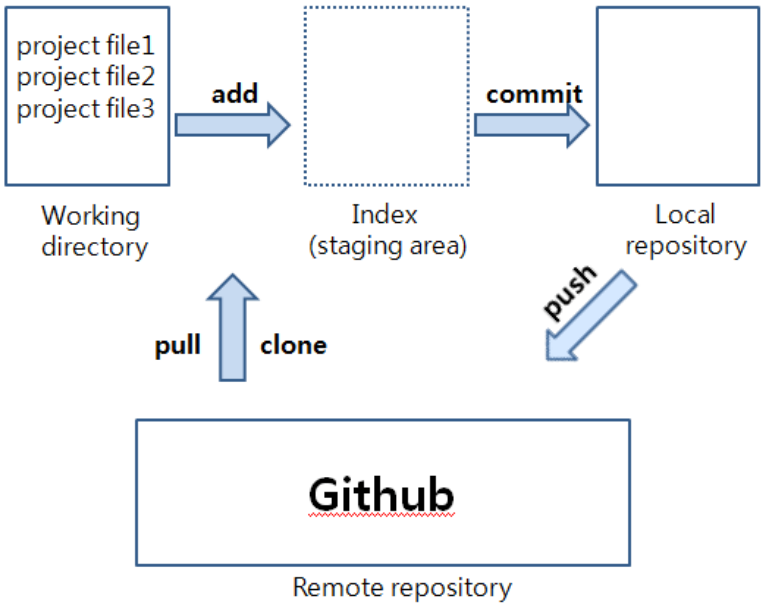
- **working directory**
  - 현재 작업하고 있는 공간을 말합니다.
  - Git이 관리하고 있지만, 아직 추적(track)하고 있지 않은 상태입니다.
- **index**
  - stage 또는 staging area라고 하며, 준비 공간을 말합니다.
  - Git이 추적하고 있으며, 버전으로 등록되기 전 상태입니다.
- **repository**
  - 저장소를 의미합니다.
    - 본인 PC에 존재하는 저장소인 **local repository**
    - Github, Gitlab 같은 원격 저장소인 **remote repository**가 있습니다.



2) Flow

- **git init**
  - **.git** 폴더를 생성합니다.
  - .git 폴더가 있어야 파일을 추적할 수 있으며, Git과 관련된 작업을 할 수 있습니다.
- **git add**
  - working directory의 변경된 작업 파일을 staging area로 추가시킵니다.
- **git commit**
  - staging area의 내용을 local repository에 확정 짓습니다.
- **git push**
  - local repository의 내용을 remote repository로 업로드 합니다.
- **git pull**
  - local repository의 내용을 remote repository에서 가져옵니다.
- **git clone**
  - .git을 포함한 remote repository의 파일들을 local repository에 복사합니다.
  - 깃헙에서 zip 파일로 받으면 .git 폴더가 없다는 것이 명령어와의 차이점입니다.

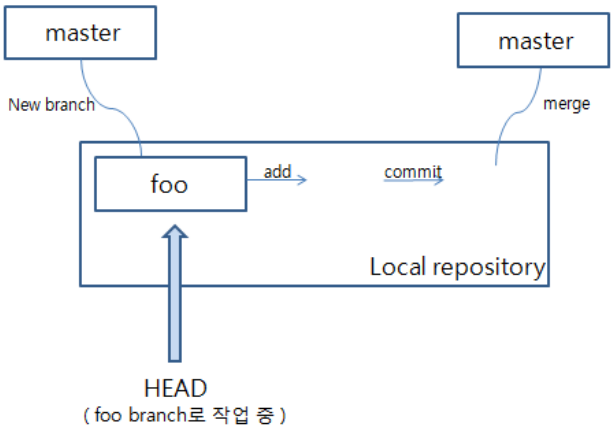
위에서 살펴본 영역과 명령어를 통해 flow를 그려보면 다음과 같습니다.



3) 협업 - 병합

- **git branch**
  - 독립된 working directory를 의미합니다.
  - 브랜치를 통해 프로젝트 참여자마다 브랜치를 가져서 독립된 작업 공간을 갖습니다.
    - 어떻게 브랜치를 관리할 것인지에 대한 **브랜치 전략**이라는 것도 있습니다.
  - 테스트 및 백업 등의 용도로 사용할 수도 있습니다.
- **head**
  - 포인터를 의미하며, 지금 작업하고 있는 branch를 가르킵니다.
- **merge**
  - 2개의 branch에서 작업한 다른 내용을 하나로 합치는 것을 말하며, 현재 브랜치를 기준으로 병합됩니다.
  - 만약 두 branch가 같은 파일의 같은 곳을 수정했다면, 충돌(**merge conflict**)이 발생해서 이를 해결해야 합니다.
    - 해당 이슈 관계자들이 상의하여 수동으로 충돌을 해결해줘야 합니다.
    - 따라서 충돌 이슈가 발생하지 않으려면 작업 내용이 겹치지 않도록 분리시키는 것이 좋겠죠.

아래는 협업과 관련된 Flow를 표현해 봤습니다.
기존에는 master 브랜치만 있는데 branch를 생성해서 새로운 working directory가 생성되었고, commit을 한 후 master로 병합(merge)하는 것을 보여줍니다.



이상으로 Git의 주요 용어와 Flow에 대해 알아보았습니다.
다음 글부터 위의 용어, 명령어들을 하나씩 살펴보도록 하겠습니다.