Git

2020.01.09

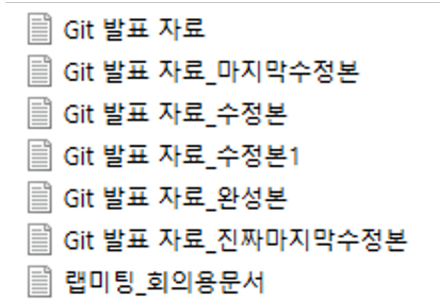
학부생 인턴 이지헌

Git이란?

- 컴퓨터 파일의 변경사항을 추적하고 여러 사용자들 간에 작업을 조율하기 위한 **분산 버전 관리 시스템**
- 소스코드를 효율적으로 관리 할 수 있게 해주는 **형상 관리 도구**라고도 함
- 파일의 **이력을 관리**
- 리누스 토발즈에 의해 개발되었으며, **무료**이며 **공개소프트웨어**

Git을 사용하는 이유

- 같은 파일을 **여러 명이 동시에 작업**하는 병렬 개발이 가능
- **협업**이 필요할 때, 여러명과 함께 **코드를 공유**
- 버전 관리를 통해 **체계적인 개발** 가능
- 개발 환경에서 **실수**를 할 수 있기 때문 (백업, 복구)



ex) 특별한 규칙 없이 마음대로 름을 붙여놓는 경우

Git repository

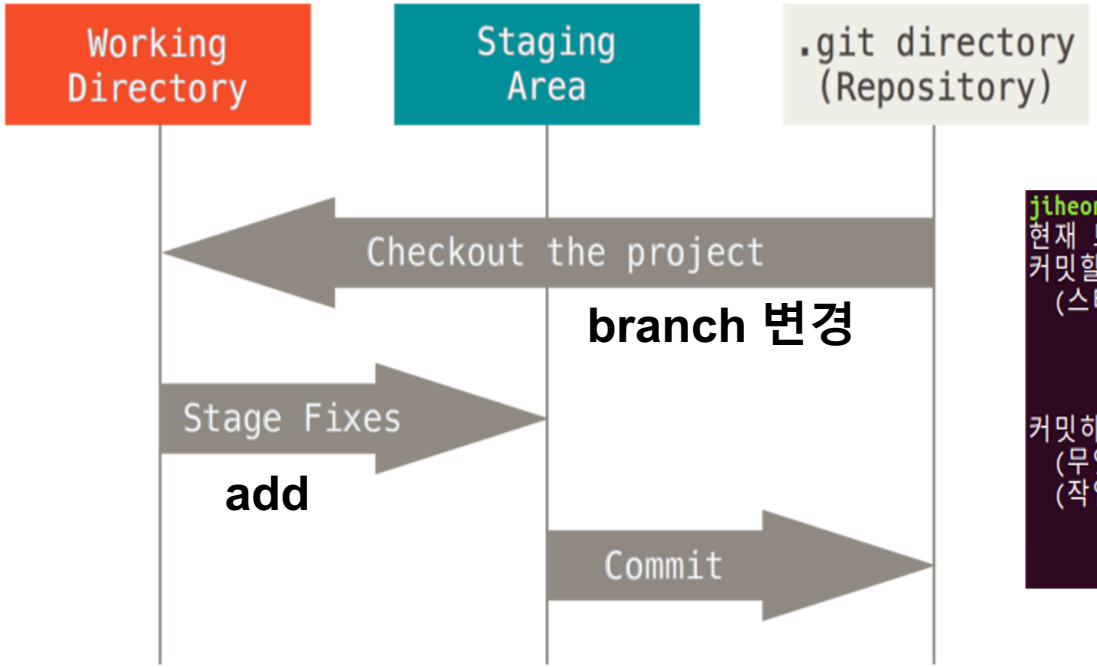
- 파일이나 폴더를 저장해 두는 곳
- 내용 일부 문구가 서로 다르면 다른 파일로 인식하여 **변경 사항 별로 구분**하여 저장 가능
- **로컬 저장소**(Local Repository)와 **원격저장소**(Remote Repository)로 나뉨

현재 디렉토리에 git 저장소를 생성

```
$ git init
```

출처: https://backlog.com/git-tutorial/kr/intro/intro1_2.html

Git의 구조




```
jiheonlee@jiheonlee-VirtualBox:~/workspace/git_test$ git status
현재 브랜치 master
커밋할 변경 사항:
  (스테이지 해제하려면 "git reset HEAD <파일>..."을 사용하십시오)

        수정함:         f1.txt

커밋하도록 정하지 않은 변경 사항:
  (무엇을 커밋할지 바꾸려면 "git add <파일>..."을 사용하십시오)
  (작업 폴더의 변경 사항을 버리려면 "git checkout -- <파일>..."을

        수정함:         f2.txt
```

<두 파일 수정했지만 f1.txt 파일만 add한 상황>

출처: https://backlog.com/git-tutorial/kr/intro/intro1_2.html

Git add와 commit

파일은 수정했지만 아직 stage area에 올라가지 않은 파일들을 staging area에 올림

```
$ git add {파일이름}
```

staging area에 올라가 있는 파일들을 commit

```
$ git commit {옵션} {파일이름}
       -a : 기존에 add를 하지 않아도 바로 staging area에 올려서 commit 함
       -m : editor없이 commit의 메시지를 입력 # $ git commit -m test.txt "Version1"
```

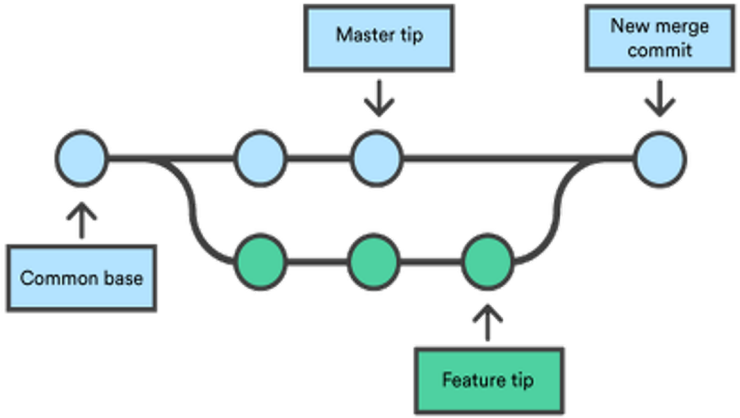
Git branch와 merge

- **branch**

  어떠한 이슈가 발생하였을 때 그 이슈를 처리할 **새로운 작업공간**

- **merge**

  두 개의 branch를 **병합**하는 것



브랜치 생성

```
$ git branch {branch명}
        - master가 기본 branch
```

다른 브랜치로 변경

```
$ git checkout
```

- Merge

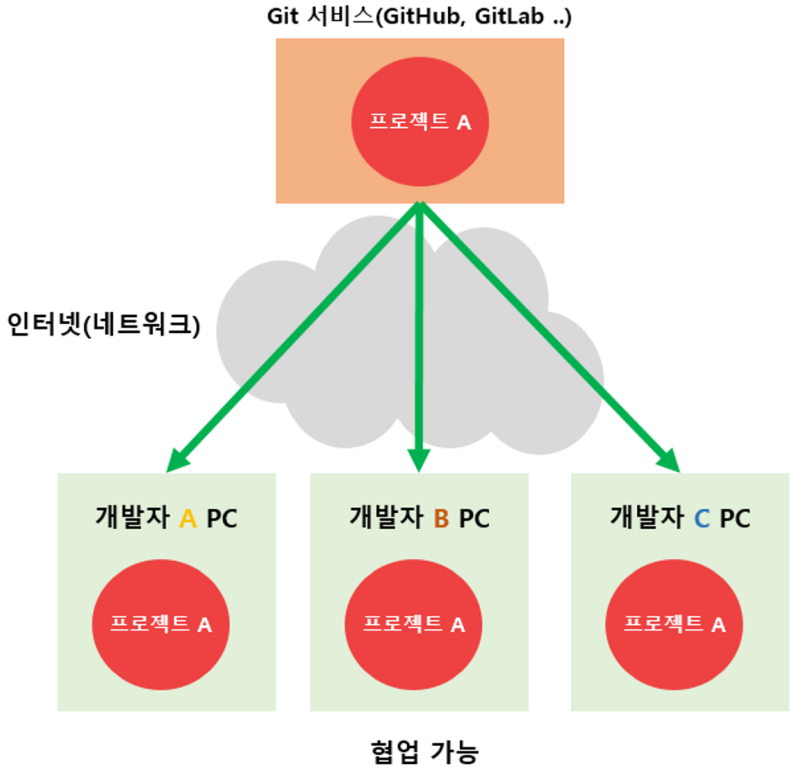
  현재 브랜치에서 입력한 브랜치와 merge(합치다)

```
$ git merger {branch명}
```

출처: https://yongyi1587.tistory.com/30

원격저장소 : remote repository

- 내 로컬PC 저장소가 아닌 네트워크상의 **다른 위치에 존재하는 Git 저장소**
- **백업**과 여러 사람이 **협업**을 하기 위해 필요
- **Github**은 인터넷을 통해 원격 저장소 공간을 제공해주는 서비스



출처: https://dololak.tistory.com/346

원격저장소 : remote, push

현재 저장소에 원격저장소를 add(연결)한다. 경로 앞에 origin이라 하면 경로는 origin을 가르킴, 원격저장소의 기본 이름은 origin

```
$ git remote add origin {경로}
```

추가한 원격저장소의 목록을 확인

```
$ git remote
       -v : 자세하게 보여줌
```

원격저장소를 제거

```
$ git remote remove origin
```

현재 브랜치를 연결시킨 지역저장소에서 원격저장소에 푸쉬, 즉 업로드

```
$ git push origin
```

원격저장소 : pull, clone

- **pull**
  local repository와 비교하여 병합을 하고,
  저장(add)까지 수행
- **clone**
  Github의 모든 파일들을 가져오기만 함

원격저장소의 파일들을 지역저장소로 가져옴

```
$ git pull
```

원격저장소의 파일들을 디렉토리를 생성하여 안에 다운로드

```
$ git clone {원격저장소 주소} {디렉토리명}
```

감사합니다