Why Git?

상황 1

13기 와이빅타 과제로 코딩을 하고 있는 와빅이
와빅이는 과제에서 구현하라고 하는 기능 5가지 중 4가지를 끝냈다.
그렇지만 마지막 기능을 추가하는 과정에서 이유 모를 오류가 발생해서 프로그램이 아예 동작하지 않는다.

> 어쩔 수 없지.. 4개만 하고 내야겠다!

엥! 근데 이게 무슨 일?
마지막 기능을 구현하면서 코드를 너무 많이 고쳐서 4개가 되던 `version`이 기억나지 않는다!
와빅이는 다 지우고 처음부터 해야하는 것일까?

와빅이의 선택

다행히 와빅이는 바보가 아니었다! 4개가 될 때의 `version`을 복사해서 가지고 있었다.

결국 와빅이의 폴더는...

> assi_1.py
> assi_1_last.py
> assi_2.py
> assi_last.py
> assi_real_last.py
> assi_real_real_last.py

상황 2

이번에는 와이빅타 프로젝트를 하기로 한 와빅이
그렇지만 코딩은 업무분담을 어떻게 나눠야 할지 모르겠다.
결국 각자 해보고 결과가 잘나온 것으로 하기로 한 와빅과 친구들

그렇게 며칠 후, 서로 어떻게 하고 있는지 공유하기로 했다.

> 좋아 그럼 카톡으로 python 파일 보내줄게!

엥! 근데 이게 무슨 일?
파이썬 파일은 카톡으로 보낼 수 없는 확장자라고 한다.
그럼 귀찮게 이메일로 다른 3명의 조원에게 보내야 하나?

와빅이의 선택

와빅이는 바보가 아니었다! 확장자를 `zip`으로 압축해서 보내면 카톡에서도 보내지는 것이었다.

결국 와빅이의 카톡방은....

> 와빅이_결과.zip
> 태오_중간_결과.zip
> 와빅_태오것_수정.zip
> 현우_최종.zip
> 우리팀_최종.zip
> 우리팀_제출.zip

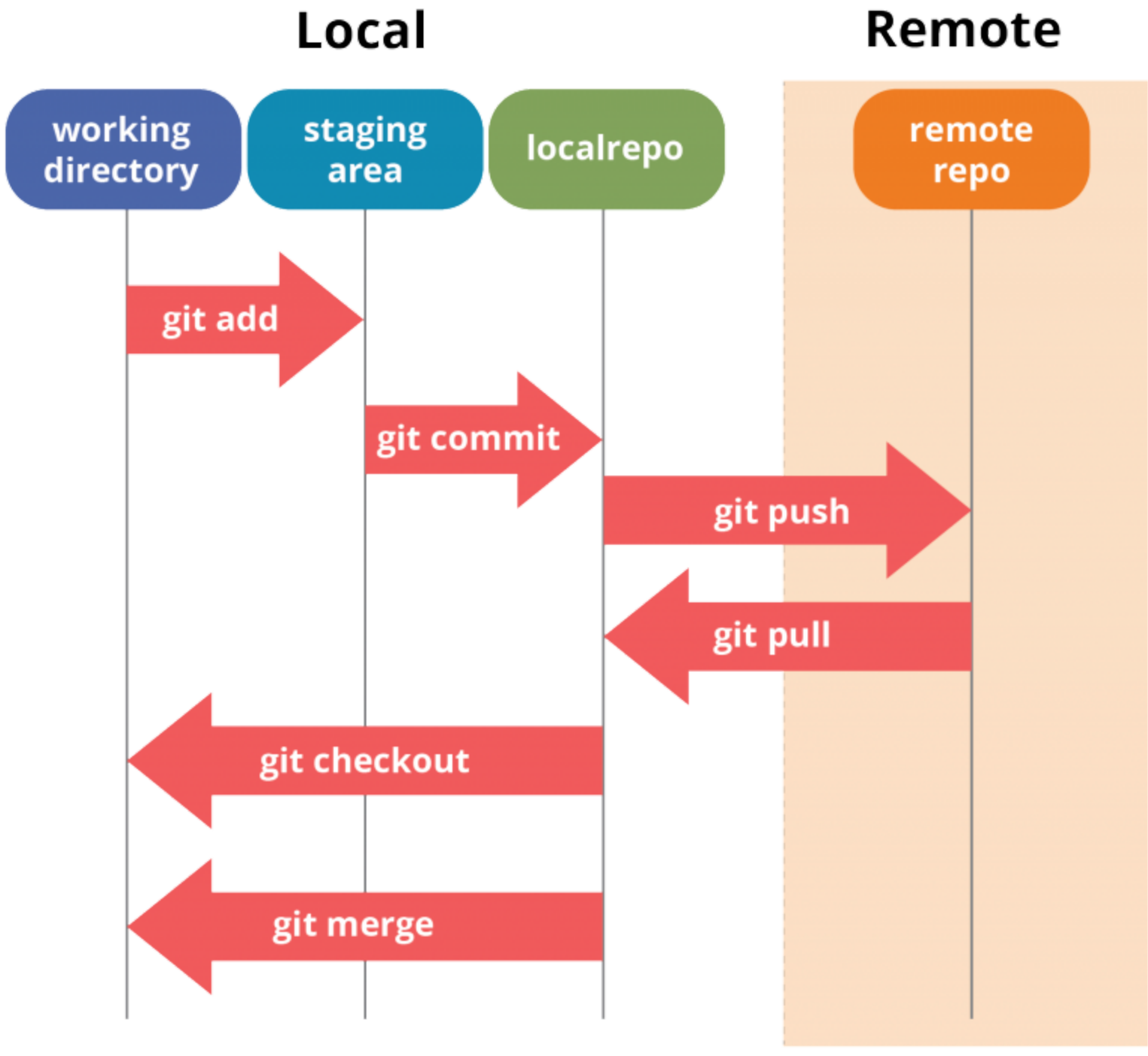
이렇듯 코딩할 때에도 `버전관리`가 필요하다.

→ Git은 프로젝트의 `버전관리`를 위해 탄생한 VCS(Version Control System)이다.

Git 기본 개념

Git이란?

Git은 폴더(프로젝트)에 들어있는 파일들의 변경사항을 기록합니다. 또한 Branch를 사용하여, 여러가지 다양한 버전으로 나누어 기록할 수 있게 해줍니다.


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout
git merge

- Working directory
  : Git 저장소로 사용되고 있는 프로젝트 폴더
- Staging Area
  : Working directory 내에서도 실제로 기록하려고 하는 파일들의 index를 저장
  - `git add assign.py`
    : assign.py 라는 파일을 staging area에 기록한다.
- Local repo
  : 실제 변경사항을 저장하는 저장소
  - `git commit -m 'first init'`
    : `first init` 이라는 메세지로 현재 staging area를 local repo로 저장한다.

예시

```
kimta@Taeoh MINGW64 ~/ybigta/example
$ git init
Initialized empty Git repository in C:/Users/kimta/ybigta/example/.git/

kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ ls
codealike.json  func1.py  func2.py

kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git add func1.py

kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git add func2.py
```

- `git init`
  : example 폴더에 git 저장소를 생성합니다.
- `git add func1.py` , `git add func2.py`
  : func1.py, func2.py 파일을 기록하기위해 add 한다.

```
kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   func1.py
        new file:   func2.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        codealike.json
```

- `git status`
  : 현재 git 상태를 확인합니다.
  `codealike.json` 파일은 add 하지 않았지 때문에 git에 의해 추적되지 않음을 확인할 수 있습니다.

```
kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git commit -m 'first init'
[master (root-commit) 6a8bf1f] first init
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 func1.py
 create mode 100644 func2.py

kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        codealike.json

nothing added to commit but untracked files present (use "git add" to track)
```

- commit 으로 현재 staging area의 있는 file (func1.py, func2.py) 의 현재 상태를 기록합니다.

```
kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git add codealike.json

kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git commit -m 'add 2'
[master b766d56] add 2
 1 file changed, 1 insertion(+)
 create mode 100644 codealike.json

kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git log
commit b766d56c724e3cc0a7cd288c0f2b2dea66234f7f (HEAD -> master)
Author: Taeoh <kimtaeoh95@gmail.com>
Date:   Tue Jul 3 03:19:46 2018 +0900

    add 2

commit 6a8bf1f3a9765f57d32a556988c50db7775c699e
Author: Taeoh <kimtaeoh95@gmail.com>
Date:   Tue Jul 3 03:07:16 2018 +0900

    first init
```

- `git add codealike.json`, `git commit -m 'add 2'`
  : 새롭게 변경된 것을 commit 함

Git Branch

- Git은 여러 Branch를 운영하며 안정적인 개발을 할 수 있게 해줍니다.

```
kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git checkout -b dev
Switched to a new branch 'dev'

kimta@Taeoh MINGW64 ~/ybigta/example (dev)
```

```
kimta@Taeoh MINGW64 ~/ybigta/example (dev)
$ git status
On branch dev
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   func2.py

no changes added to commit (use "git add" and/or "git commit -a")
```

- `git checkout -b dev`
  : dev라는 이름의 새로운 branch를 만듭니다. 그 후 func2,py에 새로운 기능을 추가합니다.

```
kimta@Taeoh MINGW64 ~/ybigta/example (dev)
$ git add .

kimta@Taeoh MINGW64 ~/ybigta/example (dev)
$ git commit -m 'new feature'
[dev 2b8a837] new feature
 1 file changed, 2 insertions(+)

kimta@Taeoh MINGW64 ~/ybigta/example (dev)
$ git checkout master
Switched to branch 'master'

kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git merge dev
Updating b766d56..2b8a837
Fast-forward
 func2.py | 2 ++
 1 file changed, 2 insertions(+)
```

- `git add .` & `git commit -m 'new feature'`
  : 변경사항을 dev branch에 commit 한다.
- `git checkout master`
  : master (기본 브랜치)로 이동한다.
- `git merge dev`
  : dev branch를 master branch와 합친다.

원격저장소

Github, Gitlab, Bitbucket

Git이 `내 컴퓨터`에만 존재하는 저장소라면,

원격저장소는 Git 저장소를 `업로드` 할 수 있는 코딩계의 `구글 드라이브`

원격 저장소 사이트들

Github - 가장 유명함. 하지만 Privite 저장소는 유로

Gitlab - 무제한의 Privite 저장소를 제공

Bitbucket - Privite한 저장소를 공유하는 사람 수가 무료에서는 한정되어있다. 하지만 Git GUI 툴과 잘 사용됨.

예시는 Github로 진행합니다.

Github ← Click!

push - 원격저장소에 git 저장소를 `업로드` 하는 것

pull - 원격저장소의 변경사항을 git 저장소에 `다운로드` 하는 것

clone - 원격저장소를 `내 컴퓨터` 에 다운로드 하는 것

fork - 다른 사람의 원격저장소 프로젝트를 내 원격저장소에 복사하는 것

```
kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git remote add origin https://gitlab.com/xodhx4/example.git

kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git push -u origin master
git: 'credential-cache' is not a git command. See 'git --help'.
Counting objects: 9, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (7/7), done.
Writing objects: 100% (9/9), 770 bytes | 110.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0)
To https://gitlab.com/xodhx4/example.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

kimta@Taeoh MINGW64 ~/ybigta/example (master)
$ git branch -a
  dev
* master
  remotes/origin/master
```

- `git remote add origin https://gitlab.com/xodhx4/example.git`
  : gitlab에 존재하는 원격저장소를 origin이라는 이름으로 추가한다.
- `git push -u origin master`
  : origin 원격 저장소에 master branch를 업로드 한다. 이 후에는 `git push`만 해도 자동으로 master branch에서 origin branch로 향한다.

실제 어떻게 할까?

혼자 개발할 때

1. 새 폴더를 만든다
2. `git init`으로 git 저장소를 만든다.
3. `remote add`로 원격저장소를 추가한다. (저는 보통 privite.. 창피해서)
4. dev branch를 만든다.
5. 일부분을 완성하고 test 한다.
6. test에 이상이 없을 시 master branch와 merge한다.
7. 4 - 6 반복

여럿이서 개발할 때

- Collaborator를 활용하는 방법
- pull request를 활용하는 방법

Collaborator를 활용하는 방법

github repo에 settings 에서 collaborator를 추가한다.

- 장점 : 모든 collaborator가 commit 권한을 가지고 있기 때문에, 쉽게 이용할 수 있다.
- 단점 : 관리자가 없기 때문에 누군가 실수할 시 대참사가 발생할 수 있다.

Pull request를 활용하는 방법

1. 대표 관리자가 원격 저장소를 만든다.
2. 다른 팀원은 fork로 원격저장소를 자신의 원격저장소에 복사한다.
3. 팀원은 fork한 저장소에 commit 한다.
4. 그 후 대표 관리자의 원격 저장소에 pull request
   > 저의 원격저장소의 변화를 받아주십쇼!
5. 대표 관리자가 변경사항을 확인 후 merge할지 아닐지 결정한다.

참고자료

git - 간편 안내서

누구나 쉽게 이해할 수 있는 git 입문

Pro git

Let's `code` !