GIT & GITHUB
사용 방법 공유 세미나

2018.02.22

김 진 형

Install (설치)

install https://git-scm.com/

OS에 맞게 download
다음, 다음 누르면서 설치 완료 후
Command 창(window버튼 클릭 후 'cmd' enter)에서 git을 쳐서 설치 확인

git

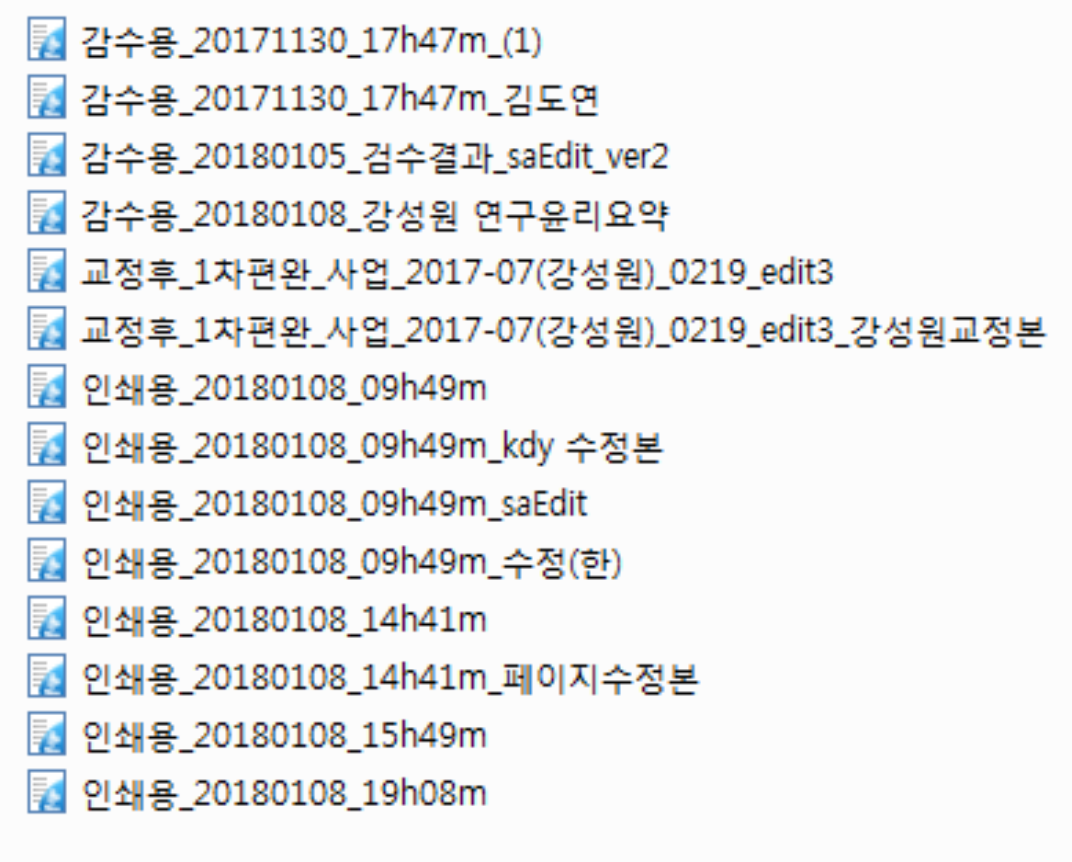
GitHub

Version Control

Collaboration & Backup

Version Control

Version Control (버전 관리)

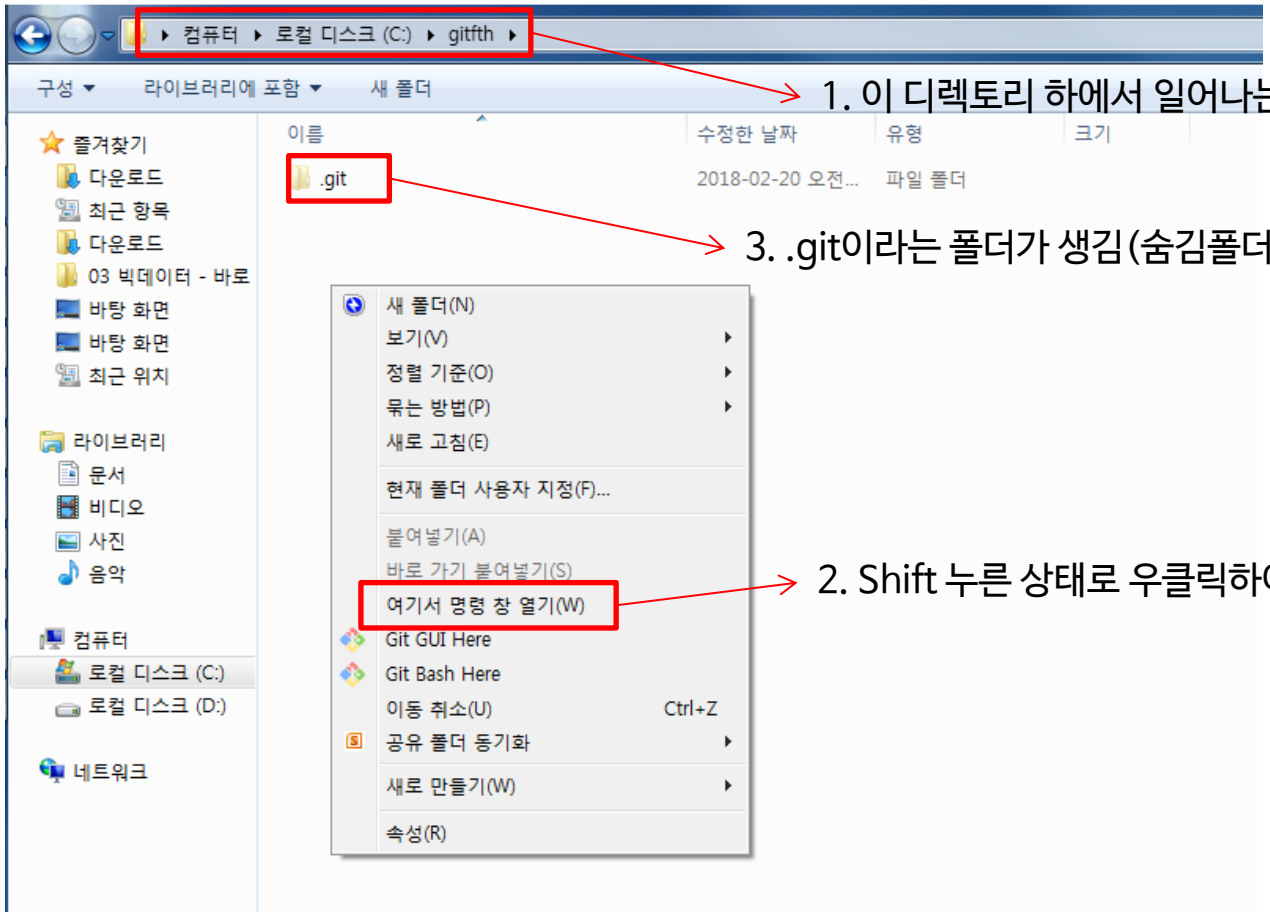


DropBox, Google Drive, Naver Drive

...

GIT으로 버전 관리 시작하기: git init

$ git init : git 저장소(Repository) 생성, 작업하는 환경의 가장 상위폴더에서 실행



1. 이 디렉토리 하에서 일어나는 작업을 버전관리할 예정

3. .git이라는 폴더가 생김(숨김폴더임), 지우거나 수정 하면 안됨!

2. Shift 누른 상태로 우클릭하여 선택하고 git init 입력

GIT으로 버전 관리할 파일 알려주기: git add

$ git add : 버전 관리할 파일을 등록, 새로운 버전에 선택적으로 파일(변경내용)을 포함시킴

$ git status : 파일의 상태를 알 수 있음

```
C:₩gitfth>git status
On branch master

Initial commit

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

        f1.txt

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

```
C:₩gitfth>git add f1.txt

C:₩gitfth>git status
On branch master

Initial commit

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

        new file:   f1.txt
```

GIT으로 버전 만들기: git commit

$ git commit : 버전을 만듦

※ 버전은 **의미 있는** 변화를 의미 함!!

$ git config --global user.name "자신의 닉네임"
$ git config --global user.email "자신의 이메일"

※ 한 번만 하면 됨

```
# Please enter the commit message for your changes. Lines
# with '#' will be ignored, and an empty message aborts t
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   f1.txt
#
~
~
~
~
~
~
~
~
~
~
~
~
~
~
/c/gitfth/.git/COMMIT_EDITMSG [unix] (16:31 20/02/2018)
"C:/gitfth/.git/COMMIT_EDITMSG" [converted][unix] 10L, 226C
```

※ commit message 입력
- 이 버전을 왜 만들었는지
- 이 버전이 어떤 변화를 담고 있는지

※ vim editor
- "i" 입력하면 insert mode
- 메시지 입력하고 esc
- :wq
- git config --global core.editor "vim"

```
C:₩gitfth>git commit
[master (root-commit) 2708c74] 1
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt
```

GIT으로 버전 만들기: git commit

$ git log : commit log 봄

```
C:₩gitfth>git log
commit 2708c74546a6fc771924d63b4530e513a0e12827
Author: jinhyeong <jinhyeongkim@kei.re.kr>
Date:   Tue Feb 20 16:31:05 2018 +0900

    1
```

※ Commit 복습

```
C:₩gitfth>git status
On branch master
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:   f1.txt

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

```
C:₩gitfth>git add f1.txt

C:₩gitfth>git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   f1.txt
```

```
C:₩gitfth>git commit
[master 5f5c995] 2
 1 file changed, 1 insertion(+), 1 deletion(-)

C:₩gitfth>git log
commit 5f5c995fc09ff8ad07d6cdb8163fd08a426f42dc
Author: jinhyeong <jinhyeongkim@kei.re.kr>
Date:   Tue Feb 20 17:16:03 2018 +0900

    2

commit 2708c74546a6fc771924d63b4530e513a0e12827
Author: jinhyeong <jinhyeongkim@kei.re.kr>
Date:   Tue Feb 20 16:31:05 2018 +0900

    1
```

GIT으로 버전 만들기: git commit

```
C:₩gitfth>git status
On branch master
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:   f1.txt
        modified:   f2.txt

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

C:₩gitfth>git add f1.txt

C:₩gitfth>git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   f1.txt

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:   f2.txt
```

※ Stage Area
Commit 대기 상태

※ Repository
Commit된 결과 저장되는 곳

GIT으로 버전 관리할 때 좋은 점: git log, git diff

GIT으로 버전 관리할 때 좋은 점: git reset, git revert

※ GIT으로 버전 관리할 때 좋은 점?
1. 과거와 현재의 차이점을 알 수 있고 과거의 내용을 알 수 있음
2. 과거로 돌아갈 수 있음

$ git log -p : 각 버전 사이의 차이점을 알 수 있음

$ git diff "버전id"..."버전id" : 각 버전 사이의 차이점을 알 수 있음

$ git diff: 파일 add 전 이전 버전과의 차이점을 알 수 있음

$ git reset "file명": add취소

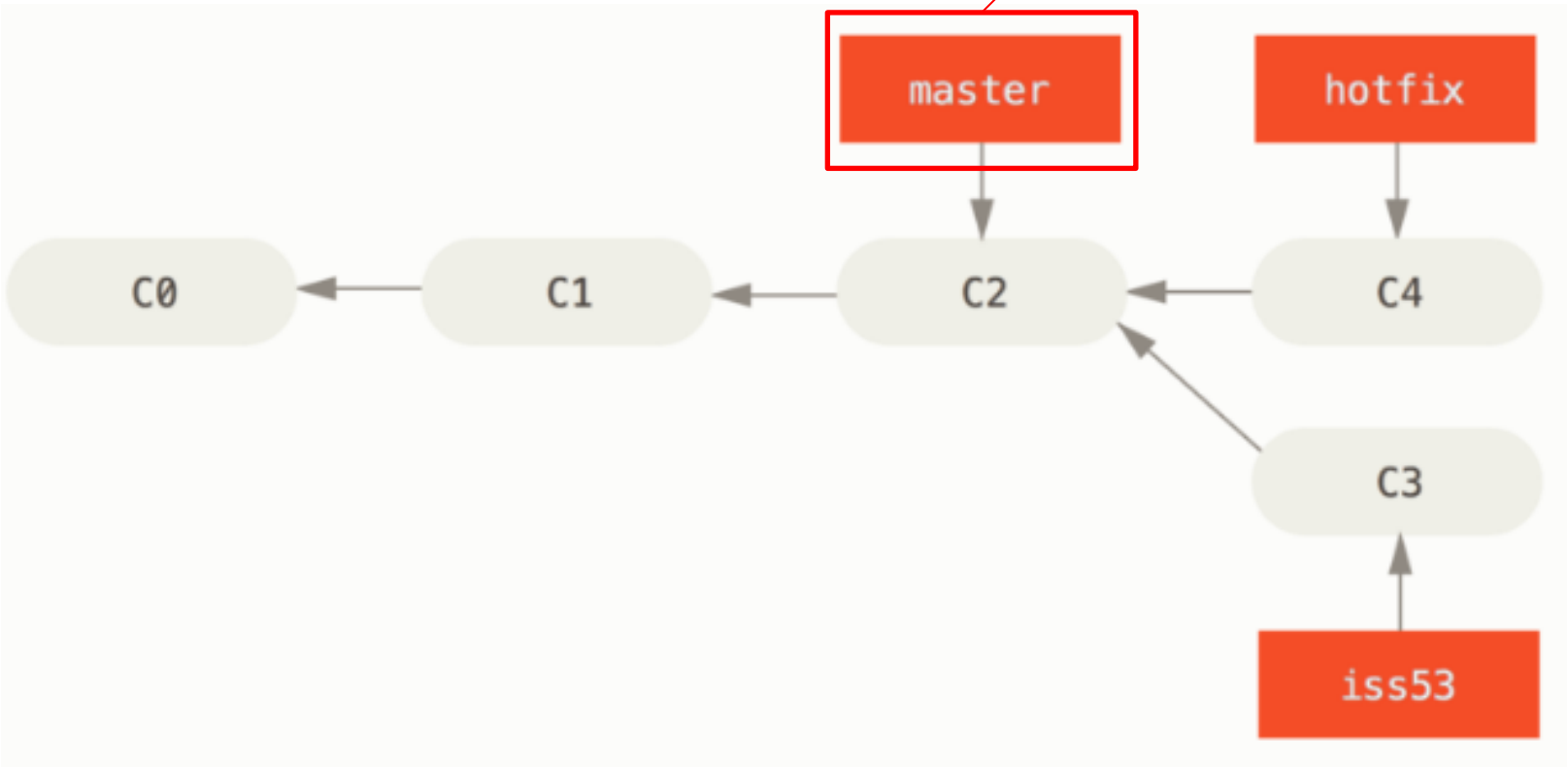
$ git reset --hard "버전id": commit 취소, 권장X!!

$ git revert "버전id": commit 취소, 권장X!

GIT에서 버전 관리하기: git branch

※ branch는 왜 쓰는가?
코드에 분기를 만들어내고 싶을 때

Git 기본 브랜치



GIT에서 버전 관리하기: git branch

$ git branch: 현재 브랜치를 보여줌
$ git branch "브랜치명": 브랜치 만들기
$ git checkout "브랜치명": 브랜치 이동
$ git branch –d "브랜치명": 브랜치 삭제

$ git log --branches --decorate -- graph --oneline
$ stree : 소스트리

$ git log "브랜치명1"..."브랜치명2": 브랜치1에는 있고 브랜치 2에는 없는 commit

$ git log –p "브랜치명1"..."브랜치명2": 브랜치1에는 있고 브랜치 2에는 없는 commit의 코드
$ git diff "브랜치명1"..."브랜치명2": 브랜치1에는 있고 브랜치 2에는 없는 코드

GIT에서 버전 관리하기: git branch

$ git merge "브랜치1": 현재 브랜치에 "브랜치1"을 커밋함

$ git status : 충돌이 발생한 파일 알 수 있음

$ git add :수동 수정 후 add

$ git commit :commit

GitHub

Collaboration & Backup

Remote Repository (원격 저장소)

※ 원격 저장소의 의미

1. 백업
2. 협업

$ git clone "주소" : 복사해옴

$ git checkout "버전id": 해당 버전으로 이동

원격 저장소에 올리기: git push

https://github.com/
가입 후, “New Repository” 생성

Quick setup — if you’ve done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/keibigdata/GCPStudy.git

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# GCPStudy" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/keibigdata/GCPStudy.git
git push -u origin master
```

원격 저장소 추가

주소의 별명: main remote repository, 기본 원격 저장소

...or push an existing repository from the command line

```
git remote add origin https://github.com/keibigdata/GCPStudy.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

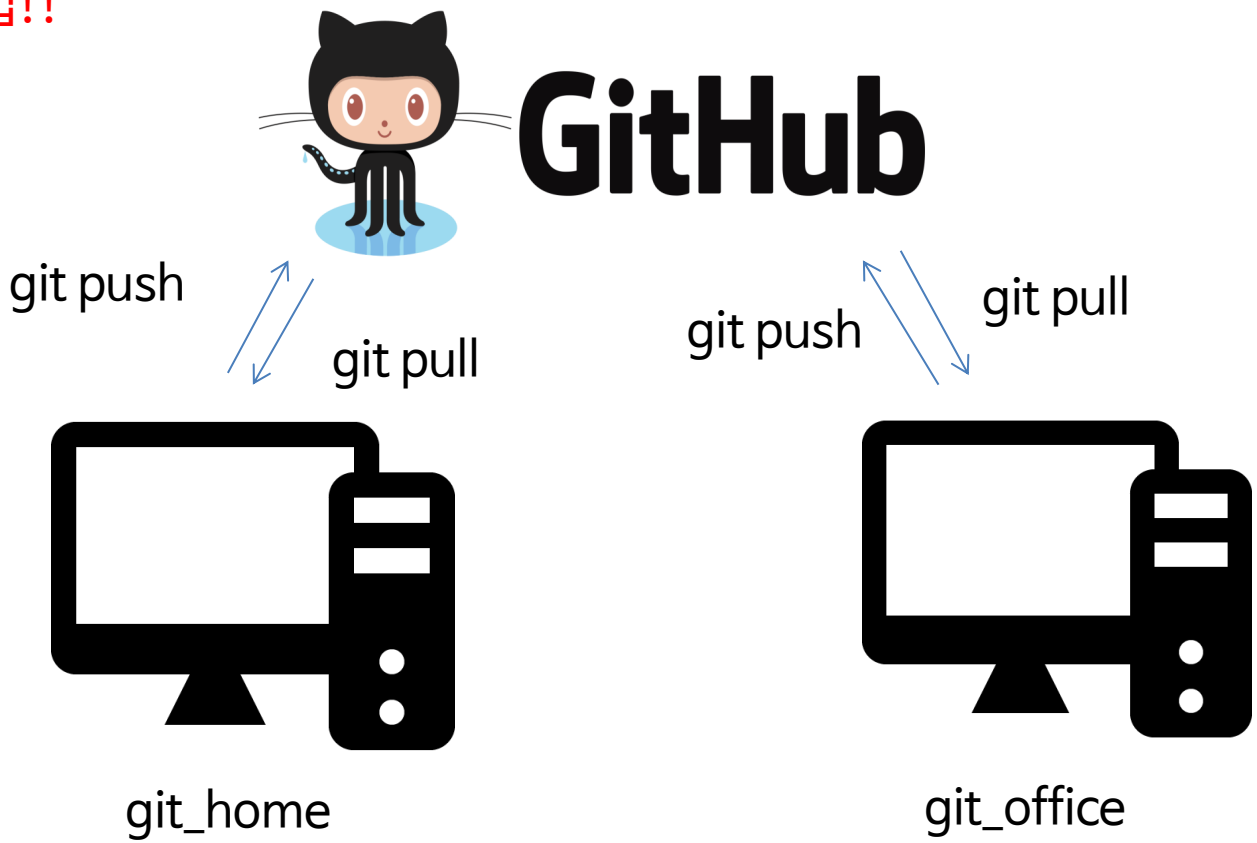
업로드

원격 저장소에서 가져오기: git pull

$ git clone "주소" git_home

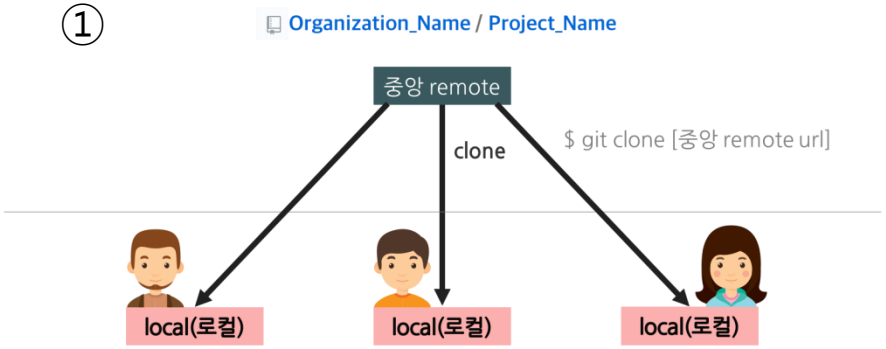
$ git clone "주소" git_office

※ 완벽한 백업!!



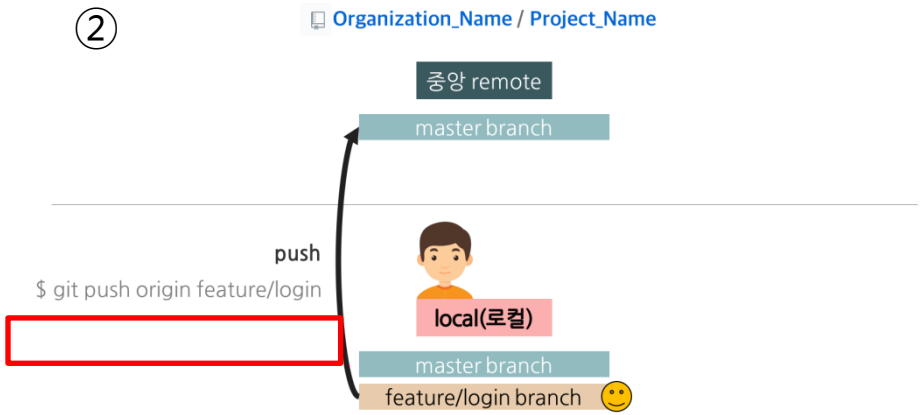
Github에서 협업하기: 1개의 원격저장소 사용

①



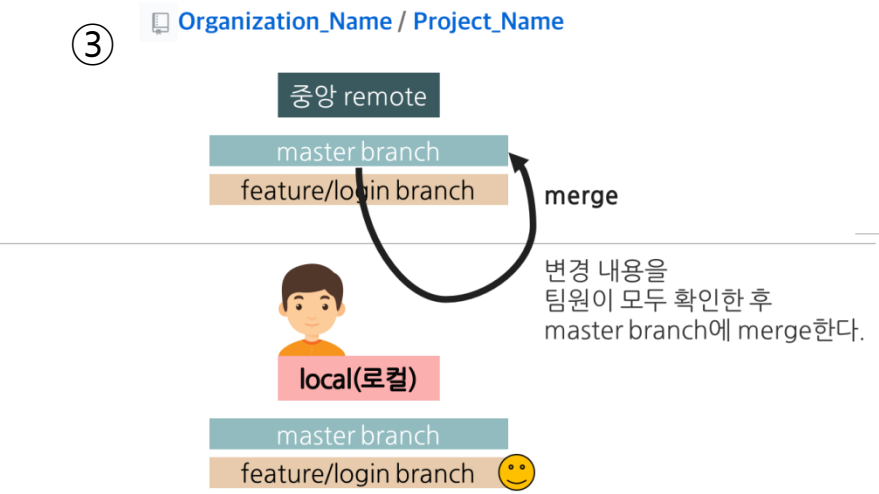
Clone 명령어를 통해 (아래의 명령어를 포함한 작업이 수행된다.)
1. 디렉토리를 만들고
2. 디렉토리로 들어가고 나서 git init 명령으로 빈 Git 저장소를 만든다.
3. 입력한 URL을 origin이라는(기본값) 이름의 리모트로 추가하고(git remote add)
4. git fetch 명령으로 리모트 저장소에서 데이터를 가져온다.

②

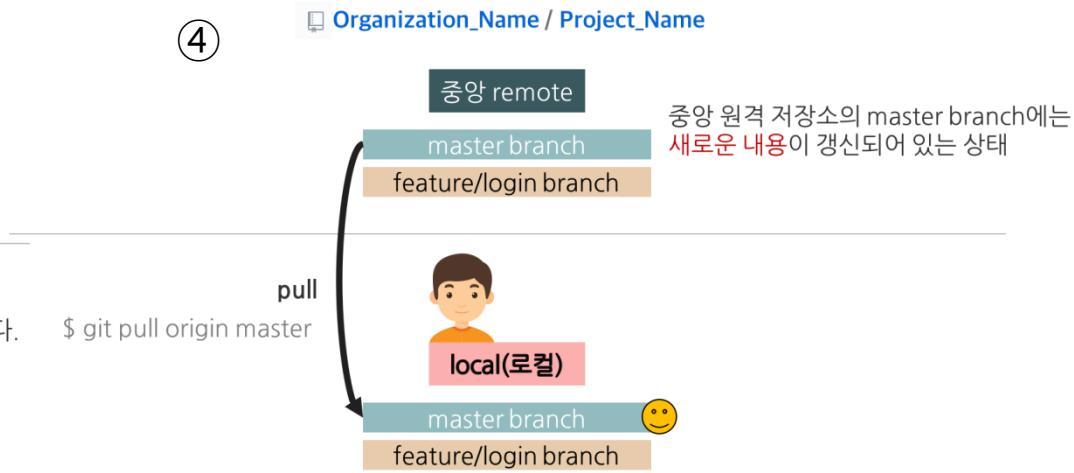


나의 작업이 끝난 경우(해당 기능을 모두 구현한 경우)
feature/login branch 를 origin remote로 push한다.
origin remote에도 똑같은 feature/login branch가 생긴다.

③



④



참고문헌

생활코딩(지옥에서 온 Git), https://opentutorials.org/course/2708

https://gmlwjd9405.github.io/2017/10/27/how-to-collaborate-on-GitHub-1.html

https://gmlwjd9405.github.io/2017/10/28/how-to-collaborate-on-GitHub-2.html

Thank you

Version control, Backup and Collaboration are essential!