정규 교육 세미나

ToBig's 11기 권혜민

Git & Github

개념 및 실습

Contents

Unit 01 | Git이란?

- Version Control System 중 하나
  ->Backup & Recovery
- Collaboration
- Distribution & Portfolio

버전 관리란?

Unit 01 | Git이란?

버전 관리란?

model.py

model_최종.py

model_진짜최종.py

model_15일
오후9시.py

Unit 01 | Git이란?

버전 관리란?

학교를 다니며 누구나 한 번 쯤은 해봤을 **버전관리!**

파일의 이름은 유지하되,
컴퓨터가 **버전을 관리** 하도록 하자!

model.py

model_최종.py

model_진짜최종.py

model_15일 오후9시.py

Unit 01 | Git이란?

- Version Control System 중 하나
  ->Backup & Recovery
- Collaboration
- Distribution & Portfolio

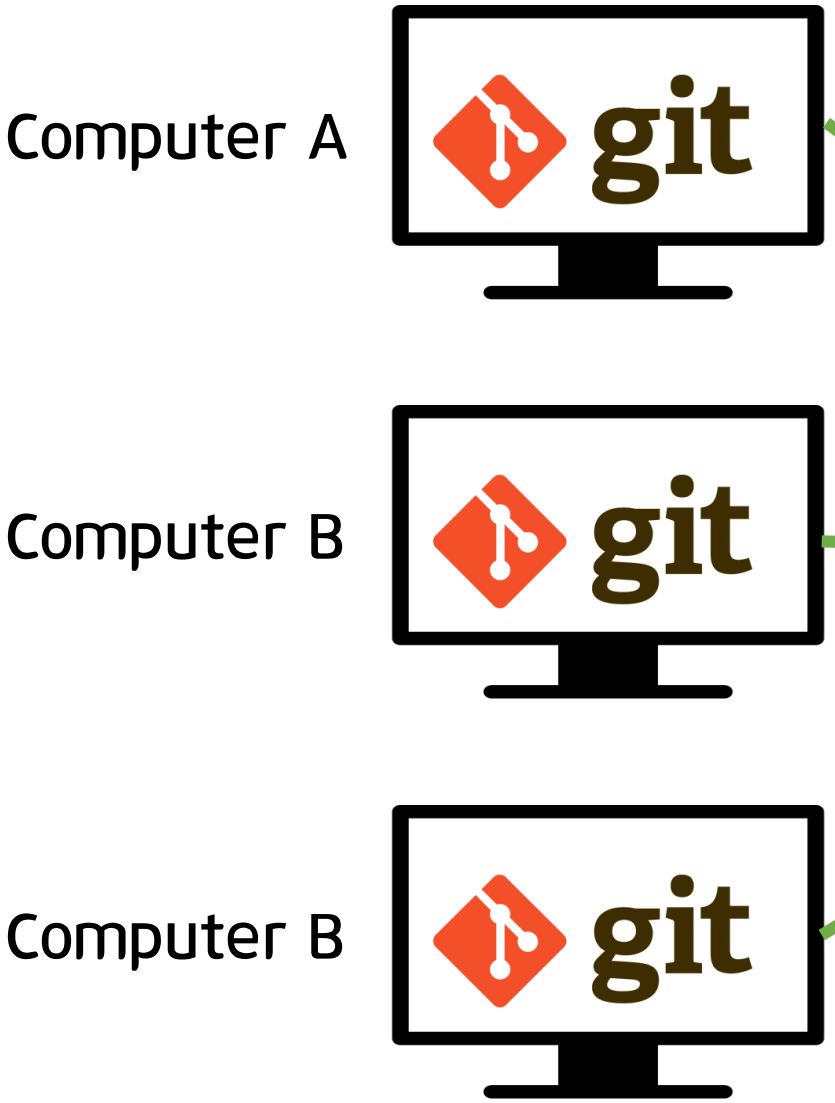
Unit 01 | Github이란?

My Computer

Remote Repository

- 소스코드 관리를 위한 분산 버전 관리 시스템
- 저장소가 본인의 컴퓨터 안에!

Unit 01 | Github이란?

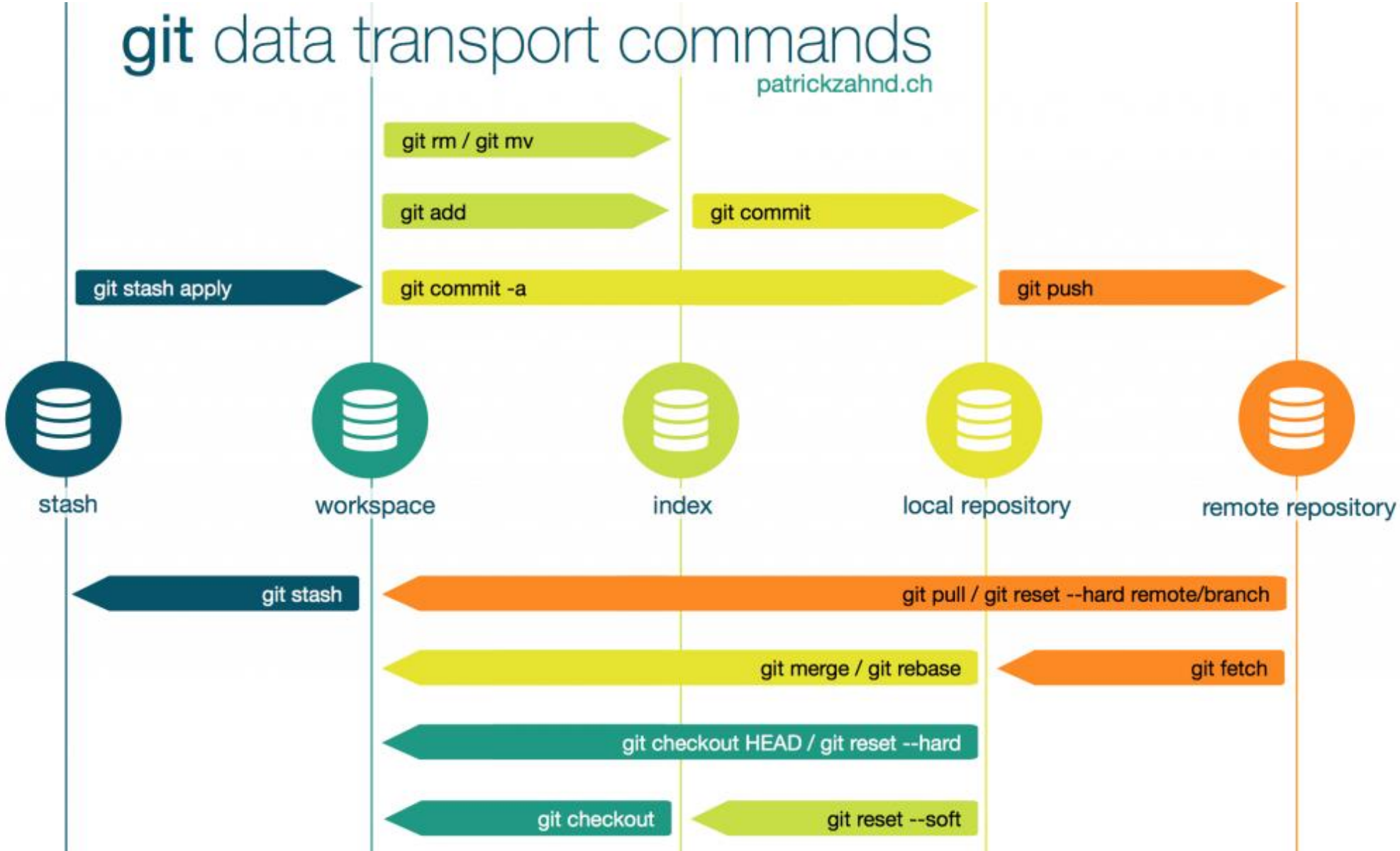


Remote Repository

GitHub

- Git을 지원하는 웹 호스팅 서비스 중 하나
- **원격 저장소** 관리
- 다양한 오픈소스
- 협업에 도움

Unit 02 | Git & Github 개념



Local Repository

개인 PC에 파일이
저장되는
개인 전용 저장소

Remote Repository

파일이 원격 저장소 전용
서버에서 관리되며
여러 사람이 함께
공유하기 위한 저장소

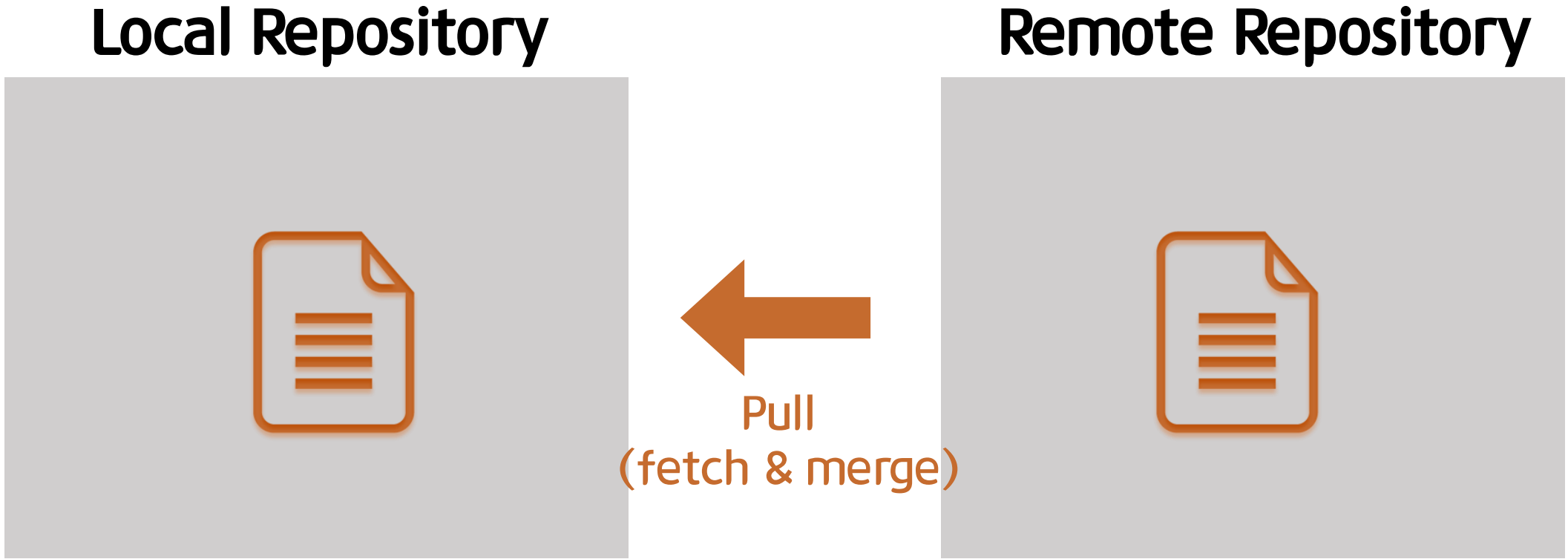
Unit 02 | Git & Github 개념

Local Repository

Push

Remote Repository

Unit 02 | Git & Github 개념



<Before Local Repository>

Working Directory

내 PC안의
작업공간들 중
Git을 사용하는
작업공간

Index
(stage 라고도 불림)

임시 버전들이
올라가는 공간

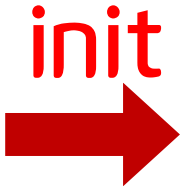
Local Repository

최종 확정본이
올라가는 공간

Unit 02 | Git & Github 개념

Working Directory

init



.git/

Index
(stage 라고도 불림)

임시 버전들이
올라가는 공간

Local Repository

최종 확정본이
올라가는 공간

Unit 02 | Git & Github 개념

init

Working Directory

.git/

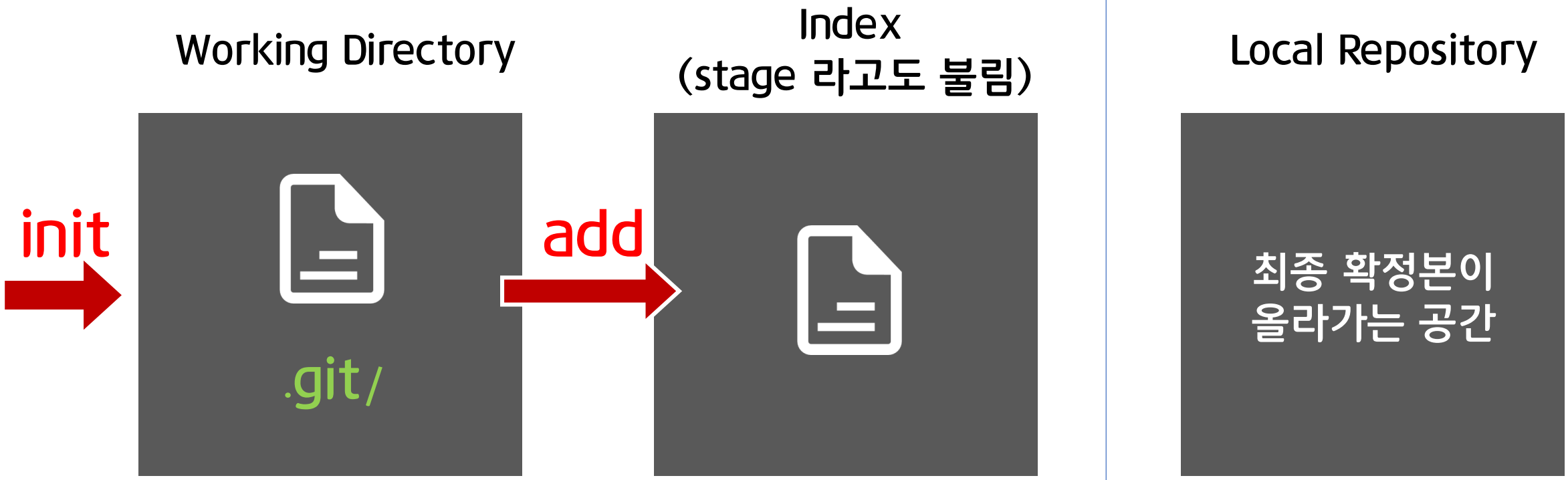
Index
(stage 라고도 불림)

임시 버전들이
올라가는 공간

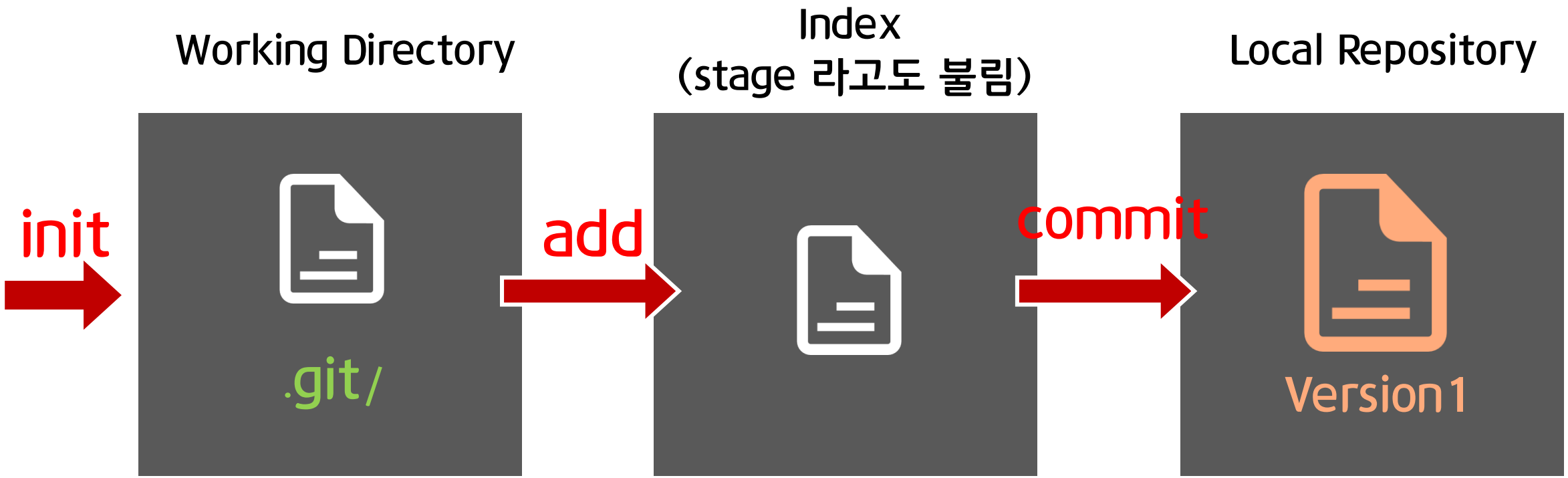
Local Repository

최종 확정본이
올라가는 공간

Unit 02 | Git & Github 개념



Unit 02 | Git & Github 개념



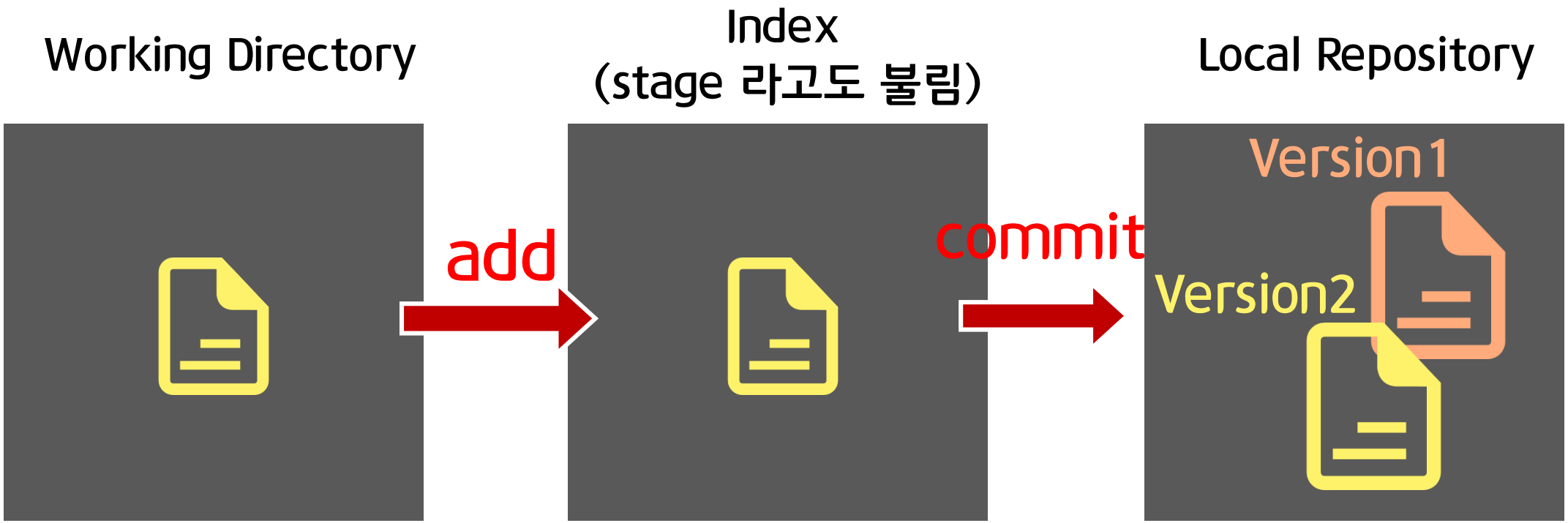
Unit 02 | Git & Github 개념

Working Directory

Index
(stage 라고도 불림)

Local Repository

Unit 02 | Git & Github 개념



Git 실습

Unit 03 | Git 실습

참고) 간단한 bash 명령어 정리

mkdir : 디렉토리를 만듦

ls : 현재 디렉토리의 list

- ls : 파일명만 보여줌
- ls –a : 디렉토리 내 모든 것 (ex. 폴더, ..)
- ls –al : 디렉토리 내 모든 것 + 접근권한 + 생성날짜 등

pwd : 현재 위치

cd : 디렉토리 변경

- cd : 최상위 폴더로 이동
- cd dir_name : dir 로 이동

vim file_name : 파일 생성, 편집

- file_name 이 현재 dir에 없는 경우 생성
- file_name 이 현재 dir에 있는 경우 편집
- (편집 방법들
  - i - 삽입, :w - 저장,
  - :q - 나가기, :wq – 저장 후 나가기)

cat file_name : 파일의 내용 출력

Unit 03 | Git 실습

0. 환경설정

전역 사용자명/메일 설정

```
git config --global user.name “username”
```

```
git config --global user.email “useremail@address”
```

저장소별 사용자명/메일 설정

```
git config user.name “username”
```

```
git config user.email “useremail@address”
```

전역 설정 정보 조회

```
git config --global --list
```

저장소별 설정 정보 조회

```
git config --list
```

Unit 03 | Git 실습

1. Working Directory 만들기

방법 1. 저장소를 새로 만들기

1) 디렉토리 생성
2) 그 디렉토리 안으로 이동
3) git init실행

```
USER@DESKTOP-B1HHQDT MINGW64 ~
$ mkdir tobig12

USER@DESKTOP-B1HHQDT MINGW64 ~
$ cd tobig12

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12
$ git init
Initialized empty Git repository in C:/Users/USER/tobig12/.git/
```

Unit 03 | Git 실습

1. Working Directory 만들기

방법 1. 저장소를 새로 만들기

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ ls

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ ls -al
total 24
drwxr-xr-x 1 USER 197121 0 7월   16 21:29 ./
drwxr-xr-x 1 USER 197121 0 7월   16 21:29 ../
drwxr-xr-x 1 USER 197121 0 7월   16 21:29 .git/
```

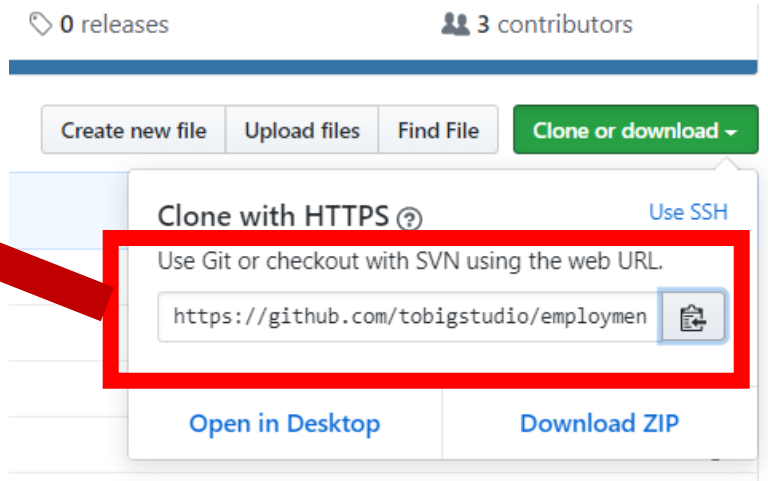
Unit 03 | Git 실습

1. Working Directory 만들기

방법 2. 이미 만들어져 있는 원격 저장소를 로컬 저장소로 복사해오기

1) 원격저장소의 주소를 복사하여
2) git clone http://github.com/이름/repo이름 을 실행

```
stKwon@stKwon-PC MINGW64 ~
$ git clone http://github.com/tobigstudio/employment_photo_generator.git
Cloning into 'employment_photo_generator'...
warning: redirecting to https://github.com/tobigstudio/employment_photo_ger
r.git/
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 33 (delta 10), reused 25 (delta 4), pack-reused 0
Unpacking objects: 100% (33/33), done.

stKwon@stKwon-PC MINGW64 ~
$ cd employment_photo_generator

stKwon@stKwon-PC MINGW64 ~/employment_photo_generator (master)
$ ls
classification_code.py  Git_simple_guide.md  README.md  Suit.py
gender_classifier.py    model.h5             src/
```



- dir만들기, dir로 이동, git init, 입력 URL을 origin이름의 remote로 추가(git remote add), git fetch(remote repository에서 file들 가져옴) 를 모두 다 한다!
- 즉, git clone <url>에 git remote add origin <url>이 포함.

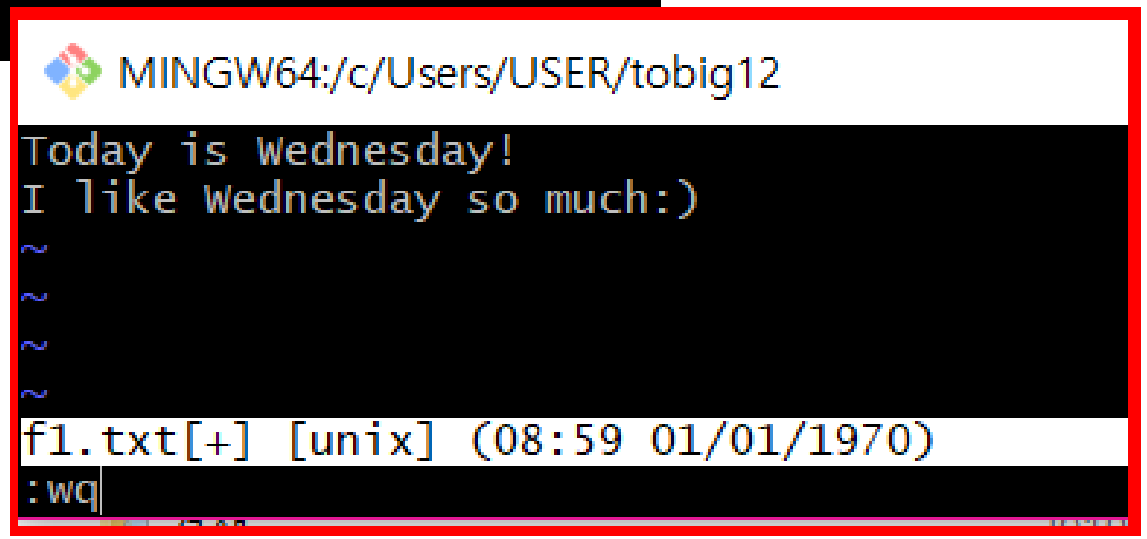
Unit 03 | Git 실습

2. staging하기

Vim editor이용하여 파일 수정

1) vim f1.txt 입력하여vim editor이용해 문서 작성/편집하기
2) i를 입력 -> 내용 작성 -> esc : wq
3) git status를 이용해 현재 상태를 확인

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ vim f1.txt
```



Unit 03 | Git 실습

2. staging하기

중간 중간 상황을 확인하는 명령어

1) git status commit전 파일들의 상태를 확인
2) git log commit된 내역을 확인

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git status
On branch master

No commits yet

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)
```

Unit 03 | Git 실습

2. staging하기

git add <파일이름> 입력하여 파일의 생성/수정을 알린다

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git add f1.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git status
On branch master

No commits yet

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

        new file:   f1.txt
```

Unit 03 | Git 실습

3. Local repository에 올리기

add된 문서들을 모두 commit

☑ git commit –m “설명”

add한 파일 중 일부만 commit하기

git commit <파일이름>

변경내용 확정을 위해 git commit

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git commit -m "version1_wednesday"
[master (root-commit) 74a2f3f] version1_wednesday
 1 file changed, 2 insertions(+)
 create mode 100644 f1.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git status
On branch master
nothing to commit, working tree clean

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git log
commit 89eea0b42fc17126f4b571cba9e9046d8c01561b (HEAD -> master
Author: HyeMinKwon <hmkwon1025@naver.com>
Date:   Tue Jul 16 21:45:29 2019 +0900

    version1_wednesday
```

Unit 03 | Git 실습

4. f1.txt 파일 수정하기

f1.txt파일 수정 후 다시
add / commit

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ vim f1.txt
```

MINGW64:/c/Users/USER/tobig12

```
Today is Wednesday!
I hate Wednesday :(
I'm sleepy
~
~
f1.txt[+] [unix] (21:35 16/07/2019)
:wq
```

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git add f1.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   f1.txt


USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git commit -m "version2_wednesday"
[master 75a230f] version2_wednesday
 1 file changed, 2 insertions(+), 1 deletion(-)

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git log
commit 75a230f0a179424b3efb7e34668396348dc4a9d1 (HEAD -> master)
Author: HyeMinKwon <hmkwon1025@naver.com>
Date:   Tue Jul 16 22:26:01 2019 +0900

    version2_wednesday

commit 89eea0b42fc17126f4b571cba9e9046d8c01561b
Author: HyeMinKwon <hmkwon1025@naver.com>
Date:   Tue Jul 16 21:45:29 2019 +0900

    version1_wednesday
```

Unit 03 | Git 실습

5. 이전 commit으로 되돌리기

현재 f1.txt

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ cat f1.txt
Today is Wednesday!
I hate Wednesday :(
I'm sleepy
```

Unit 03 | Git 실습

5. 이전 commit으로 되돌리기

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git log
commit 75a230f0a179424b3efb7e34668396348dc4a9d1 (HEAD -> master)
Author: HyeMinKwon <hmkwon1025@naver.com>
Date:   Tue Jul 16 22:26:01 2019 +0900

    version2_wednesday

commit 89eea0b42fc17126f4b571cba9e9046d8c01561b
Author: HyeMinKwon <hmkwon1025@naver.com>
Date:   Tue Jul 16 21:45:29 2019 +0900

    version1_wednesday
```

1) git log 입력 후 되돌리고 싶은 버전의 commit id를 copy

2) git reset --hard <commit id>

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git reset --hard  89eea0b42fc17126f4b571cba9e9046d8c01561b
HEAD is now at 89eea0b version1_wednesday

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ cat f1.txt
Today is Wednesday!
I like Wednesday so much:)
```

<보너스>
Reset을 취소하고 싶다!

3) git reset --hard ORIG_HEAD

Unit 03 | Git 실습

6. 삭제된 파일 backup하기

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ rm f1.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ ls
```

1) 파일을 삭제

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git checkout -- f1.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ ls
f1.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ cat f1.txt
Today is Wednesday!
I like Wednesday so much:)
```

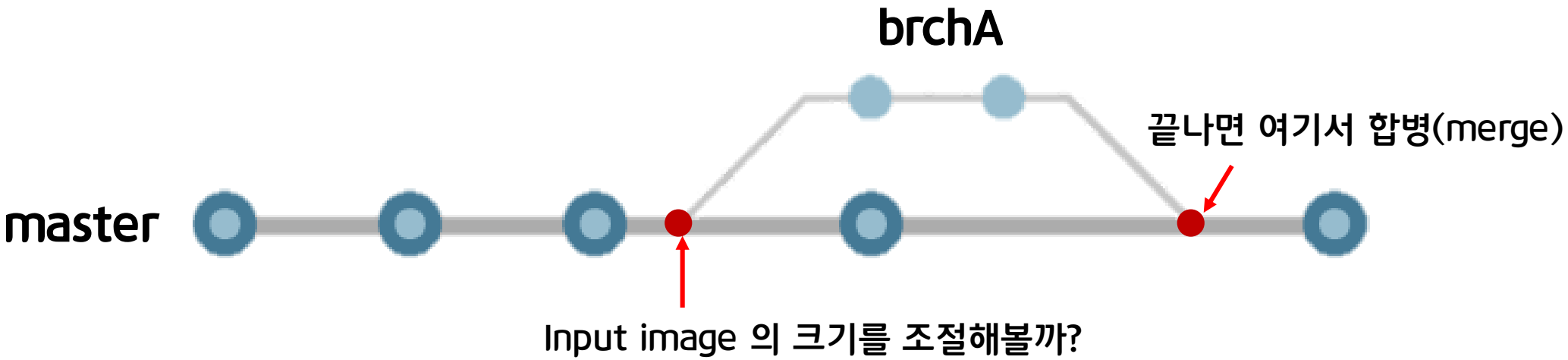
2) git checkout -- <파일이름>

로컬 저장소-> working directory 파일 복원

7. branch 사용하기

branch란? : 독립적인 작업 공간

- 테스트가 필요하거나, 다양한 버전을 구현하고 싶을 때 사용
- Local/ remote 둘 다에 branch 생성 가능
- 각자 / 기능별로 구현 후 통합
- Master에는 팀원들과 협의 후 완성된 것들만!



7. branch 사용하기

Branch사용하기

- 브랜치 확인
  git branch
- Remote repository의 remote 브랜치 확인
  git branch –r
- 브랜치 생성
  git branch <브랜치명>
- 브랜치 생성 후 이동
  git checkout –b <브랜치명>
- Remote repository에 remote branch생성
  git push origin <브랜치명>
- Master 브랜치와 합병(merge)할 때
  git merge <브랜치명>

Unit 03 | Git 실습

7. branch 사용하기

1) git branch <브랜치 명> - branch생성

2) git branch - local branch확인

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git branch Test

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git branch -a
  Test
* master
```

현재 branch

* git branch –a : 전체(local & remote) branch확인

* git branch –r : remote branch확인

Unit 03 | Git 실습

7. branch 사용하기

원하는 branch에서 작업하기 위해

git checkout <브랜치 명> - 해당 브랜치로 이동

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git checkout Test
Switched to branch 'Test'

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git branch
* Test
  master
```

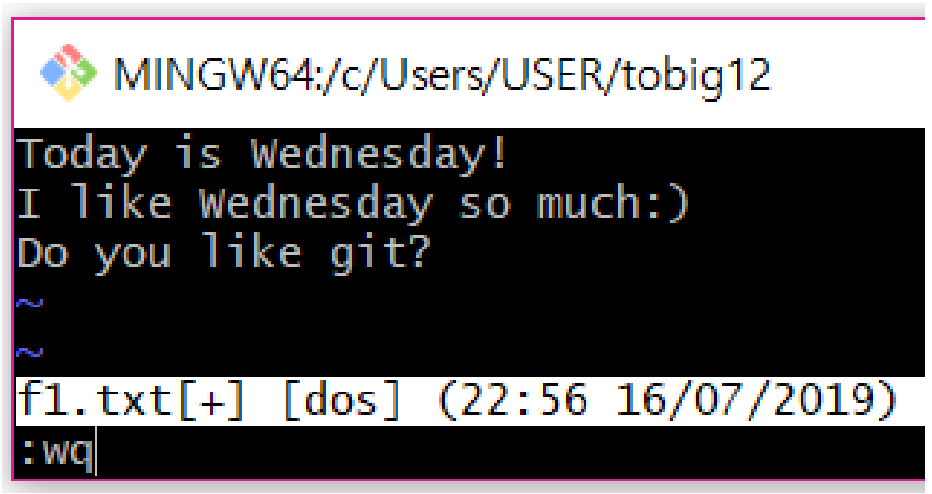
현재 branch

Unit 03 | Git 실습

7. branch 사용하기

Test 브랜치에서 f1.txt를 수정/add/status

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ vim f1.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git add f1.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git status
On branch Test
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   f1.txt
```



Unit 03 | Git 실습

7. branch 사용하기

Test 브랜치에서 a.py를 vim 에디터로 생성/status

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ vim a.py

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git status
On branch Test
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   f1.txt

Untracked files:
  (use "git add <file>..." to include in what wil

        a.py
```

MINGW64:/c/Users/USER/tobig12

```
x=10
y=20
print(x+y)
~
~
~
a.py[+] [unix] (08:59 01/01/1970)
:wq
```

Unit 03 | Git 실습

7. branch 사용하기

1) Test 브랜치에서 a.py를 git add a.py

2) status/commit

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git add .

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git status
On branch Test
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   a.py
        modified:   f1.txt


USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git commit -m "edit f1.txt and make a.py"
[Test 74542ee] edit f1.txt and make a.py
 2 files changed, 4 insertions(+)
 create mode 100644 a.py
```

3) git log --branches --decorate --graph --oneline

Log에 모든 브랜치를 표시, 그래프로 표현, 한줄로 표시

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git log --branches --decorate --graph --oneline
* 74542ee (HEAD -> Test) edit f1.txt and make a.py
* 89eea0b (master) version1_wednesday
```

Unit 03 | Git 실습

7. branch 사용하기

1) git checkout master 하여 master브랜치로 다시 이동

2) ls -> a.py가 없다?

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git checkout master
Switched to branch 'master'

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ ls
f1.txt
```

3) git merge <브랜치명> 으로

master에 Test의 변경사항을 병합(merge)

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git merge Test
Updating 89eea0b..74542ee
Fast-forward
 a.py   | 3 +++
 f1.txt | 1 +
 2 files changed, 4 insertions(+)
 create mode 100644 a.py

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ ls
a.py  f1.txt
```

4) cat f1.txt

Unit 03 | Git 실습

7. branch 사용하기

단, 항상 merge가 잘 되지는 않는다!!

MINGW64:/c/Users/USER/tobig12

```
def a(x){
    if x%2 ==0:
        return 1
    else:
        return 0
}
~
~
conf.py[+] [unix] (08:59 01/01/1970)
:wq
```

Master branch

```
def a(x){
    if x%2 ==0:
        return "I love you"
    else:
        return 0
}
```

Test branch

```
def a(x){
    if x%2 ==0:
        return 'even'
    else:
        return 'odd'
}
```

참고 자료 : https://jeong-pro.tistory.com/106

Unit 03 | Git 실습

7. branch 사용하기

단, 항상 merge가 잘 되지는 않는다!!

Master branch

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git merge Test
Auto-merging conf.py
CONFLICT (content): Merge conflict in conf.py
Automatic merge failed; fix conflicts and then commit the result.
```

Test branch

```
def a(x){
    if x%2 ==0:
<<<<<<< HEAD
        return "I love you"
=======
        return 'even'
>>>>>>> Test
    else:
        return 'odd'
}
```

master

Test

가끔 이렇게 충돌(conflicts)가 발생

But! Git이 충돌발생부분을 알려줌

그 부분을 직접 수정 후 다시 add/commit

참고 자료 : https://jeong-pro.tistory.com/106

Unit 03 | Git 실습

7. branch 사용하기

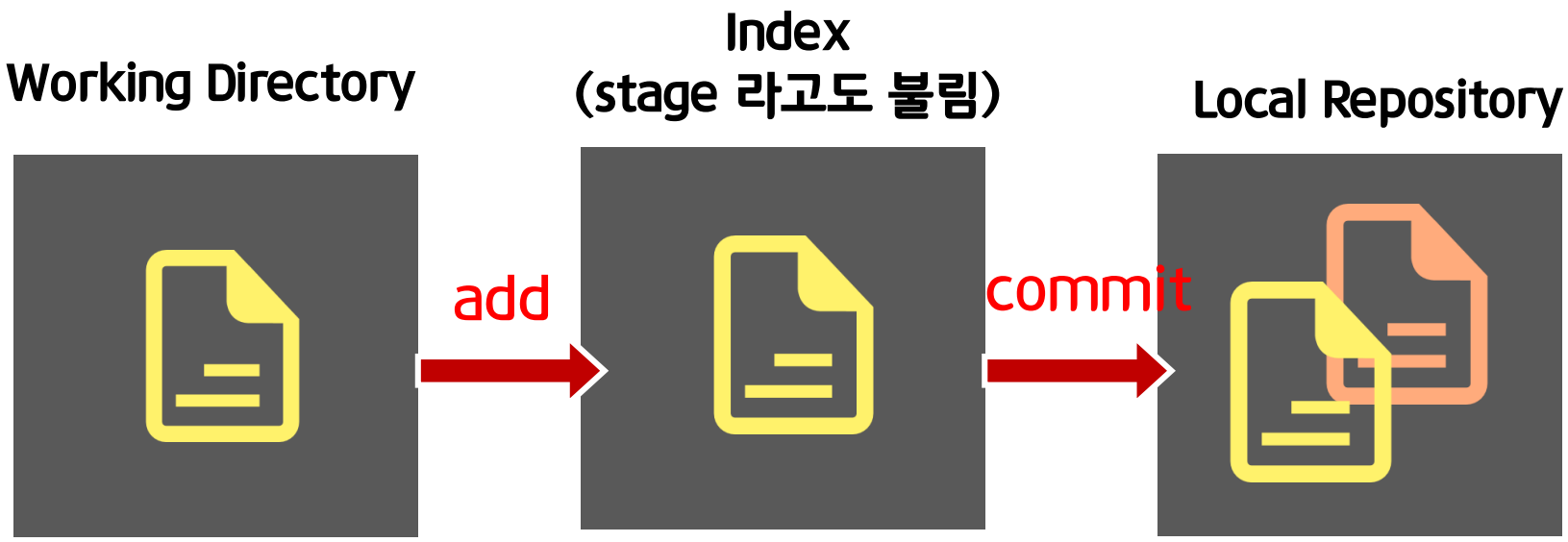
수명을 다한 branch…

1) git checkout master하여 master로 돌아옴
2) git branch –d <브랜치명>으로 branch제거
3) git branch 로 제거 확인

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test2)
$ git checkout master
Switched to branch 'master'

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git branch -d Test2
Deleted branch Test2 (was b511487).
```

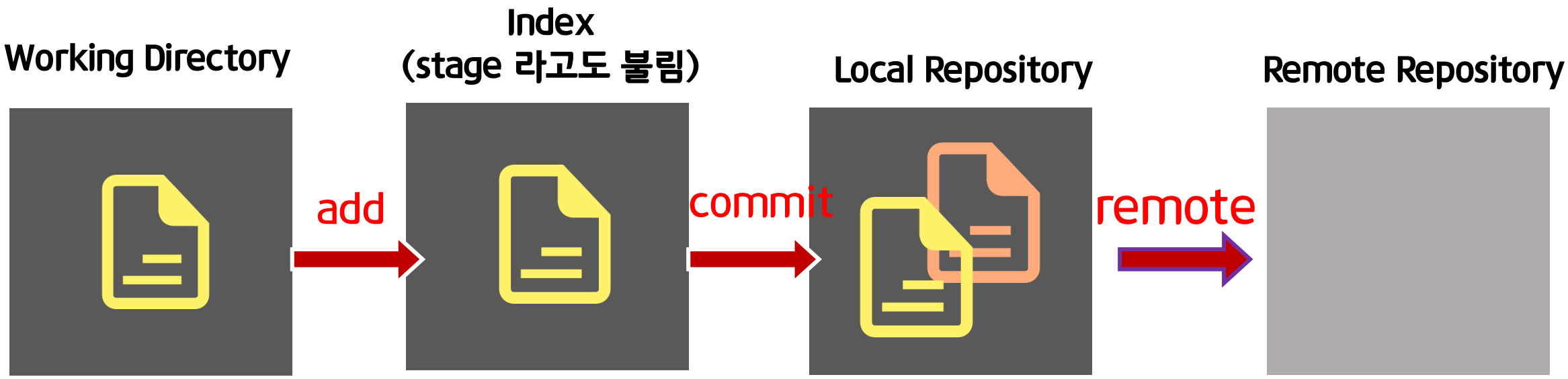
Github 실습

Unit 04 | Github 실습



Remote Repository

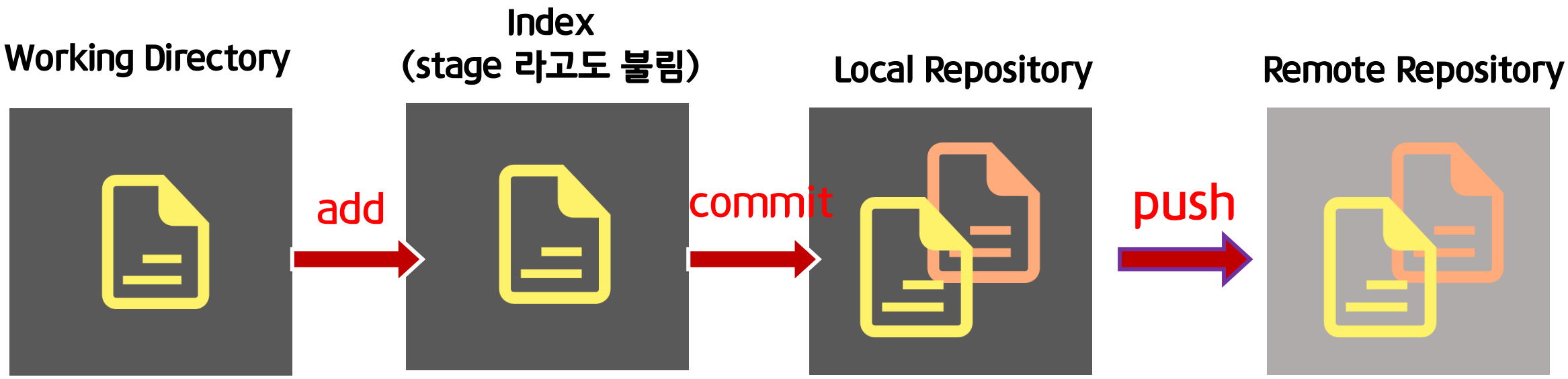
Unit 04 | Github 실습



git

GitHub

Unit 04 | Github 실습

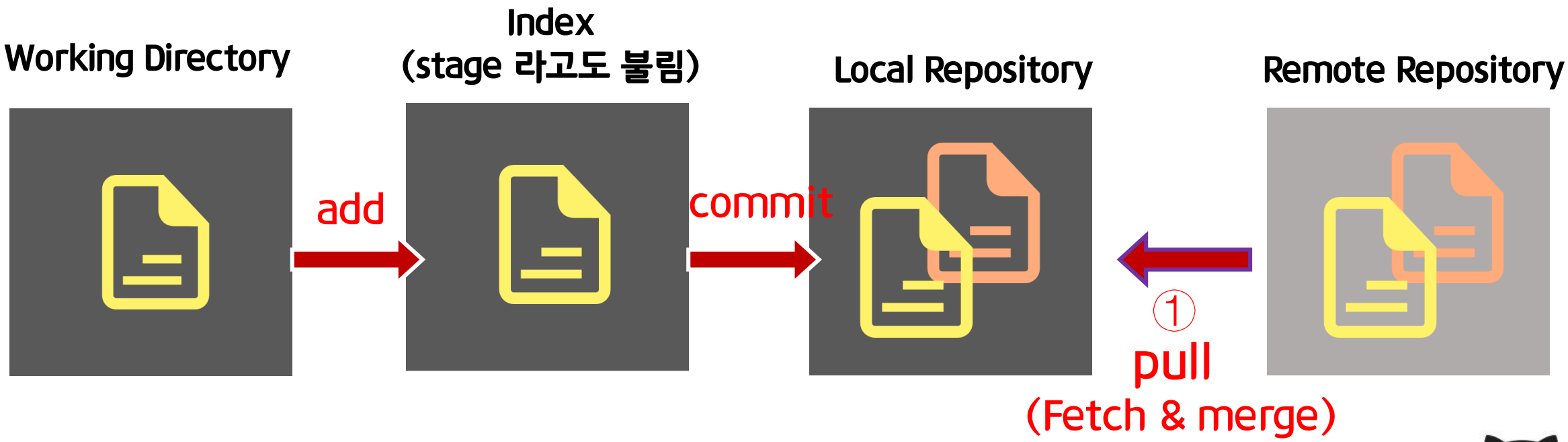


git

GitHub

Unit 04 | Github 실습

Pull : 다른 사람의 update를
fetch & merge

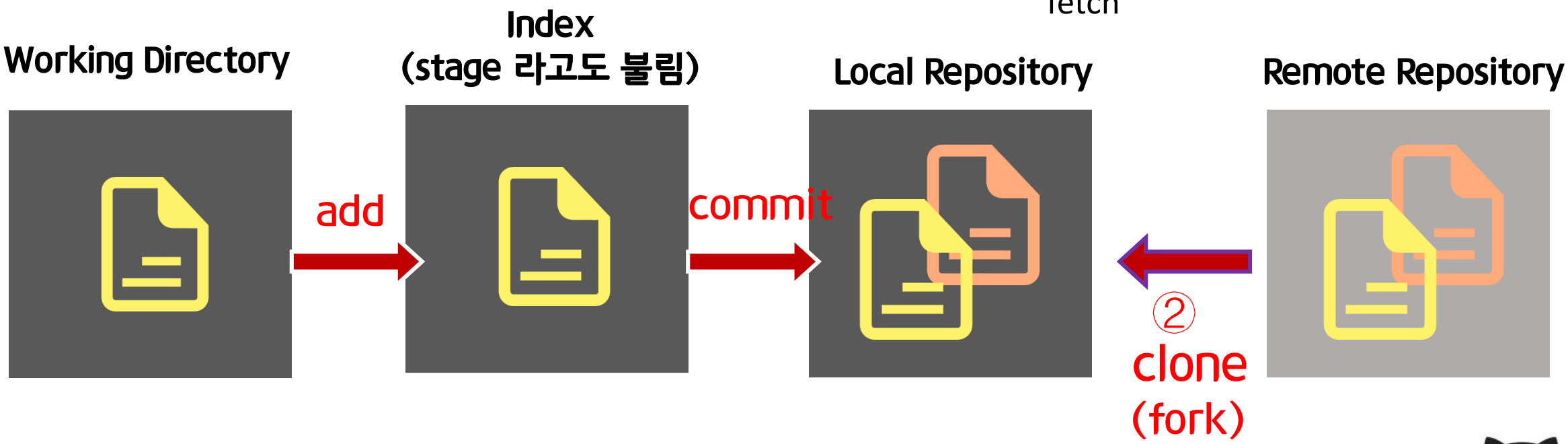


git

GitHub

Unit 04 | Github 실습

Clone : local에 아무것도 없을 때 remote의 프로젝트를 fetch



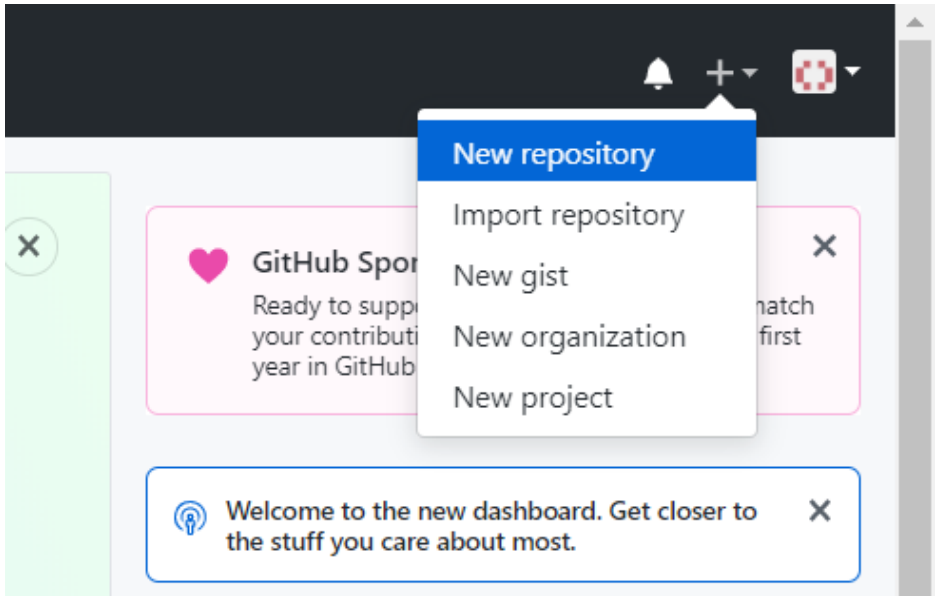
git

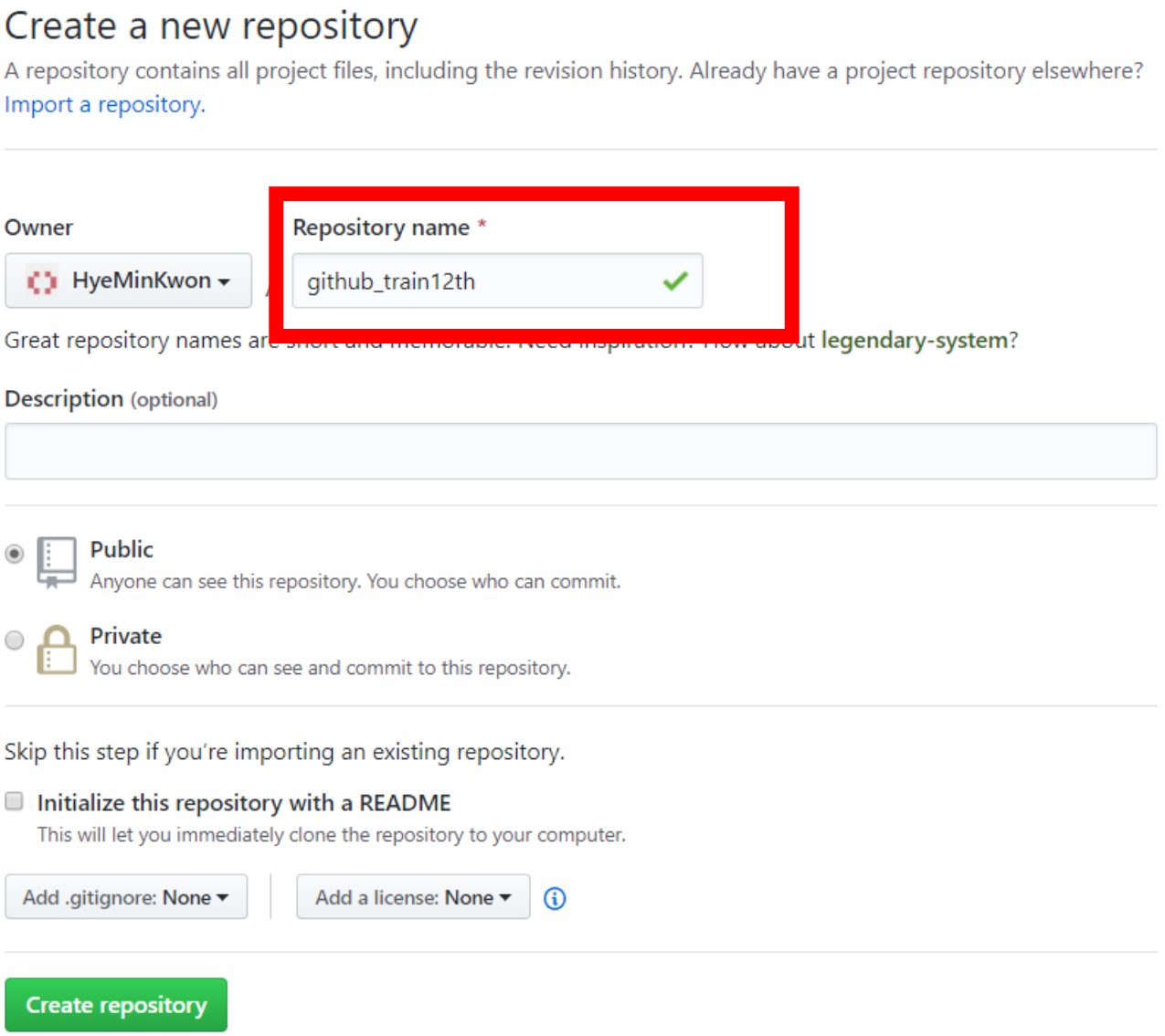
GitHub

Unit 04 | Github 실습

New Repository 생성

https://github.com/ 로 이동~





Unit 04 | Github 실습

Remote 저장소와 연결

☑ 1) git remote add origin <github http주소> 명령어 이용하여 원격 저장소와 연결

꼭 origin이라고 안해도 됨!
그저 이 git에서는 원격 저장소 주소를 origin이라고 부를 뿐!(그냥 별칭)

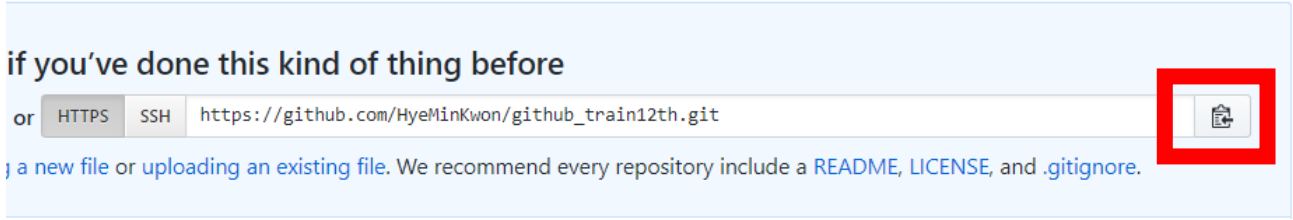
2) 또는 git clone <github 주소>
-> git directory생성과 동시에, remote까지

** ssh키를 사용하고 싶다면…

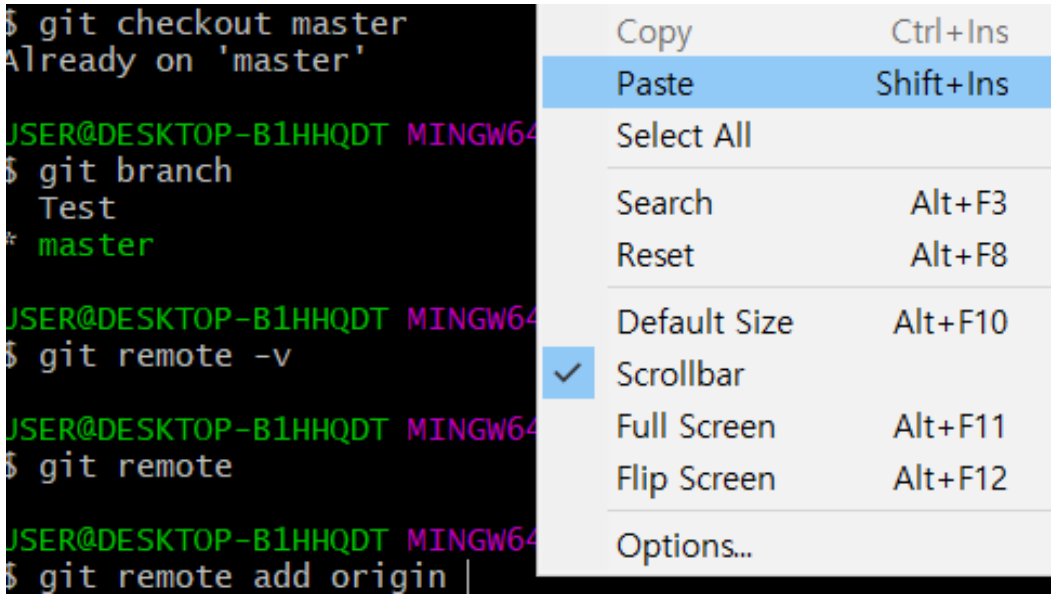
https://git-scm.com/book/ko/v1/Git-%EC%84%9C%EB%B2%84-SSH-%EA%B3%B5%EA%B0%9C%ED%82%A4-%EB%A7%8C%EB%93%A4%EA%B8%B0

Unit 04 | Github 실습

1) http 주소 복사



2) git remote add origin <http주소 붙여넣기>



3) git remote 명령어로 원격저장소 확인

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git remote add origin https://github.com/HyeMinKwon/semina_12th.git

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git remote
origin
```

Unit 04 | Github 실습

로컬저장소 -> 원격저장소로 push~!

git push –u origin master 으로
원격 저장소에 내 local안의 파일들을 push

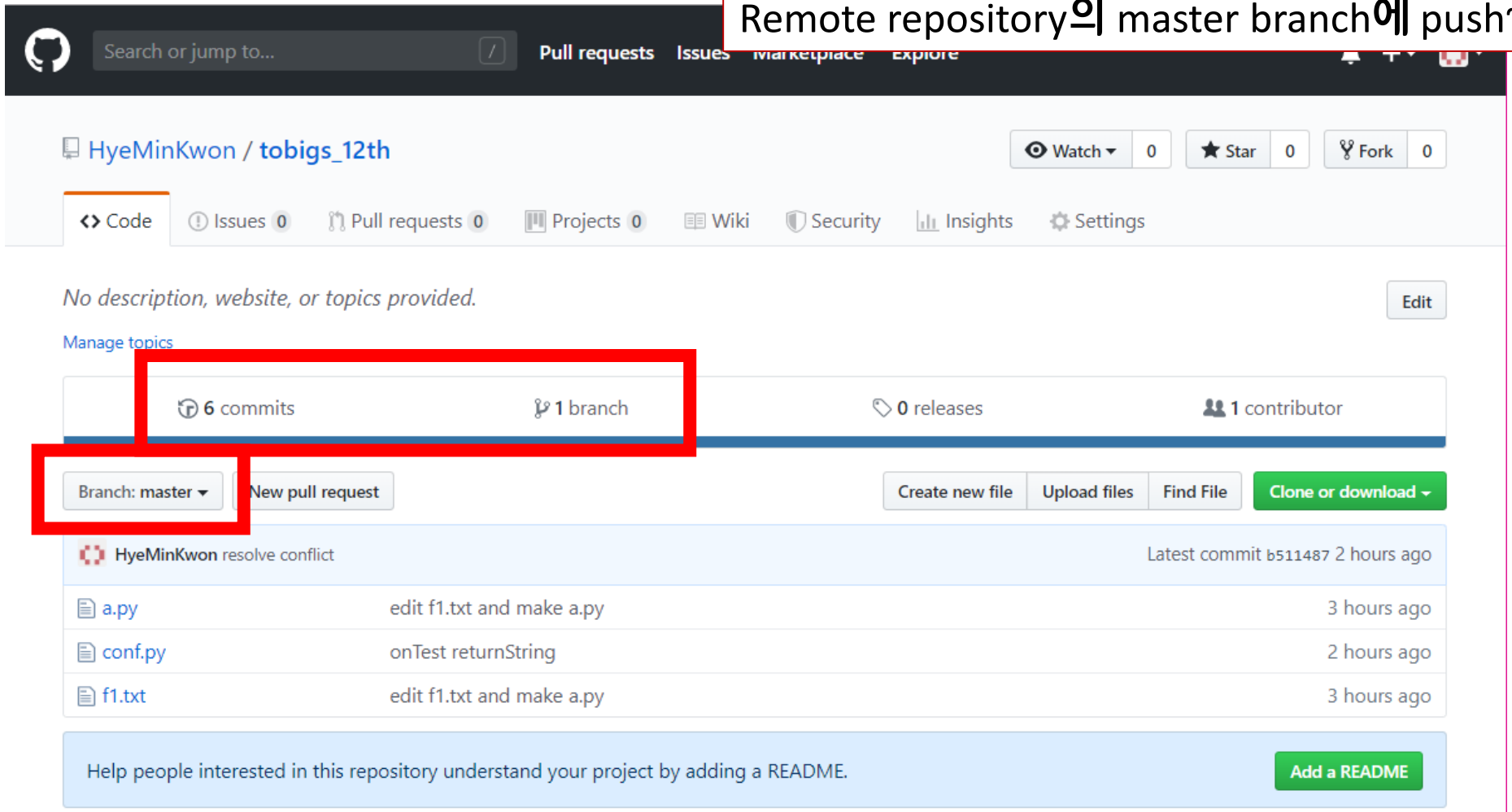
```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git push origin  master
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (17/17), 1.59 KiB | 181.00 KiB/s, done.
Total 17 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/HyeMinKwon/tobigs_12th.git
 * [new branch]      master -> master
```

-u 옵션 : 입력 인자를 기억하는 옵션. 이거 주면 그 다음부터는 그냥 git push만 하면 된다

Unit 04 | Github 실습

로컬저장소 -> 원격저장소로 push~!

Remote repository의 master branch에 push~!



Unit 04 | Github 실습

로컬저장소 -> 원격저장소로 push~!

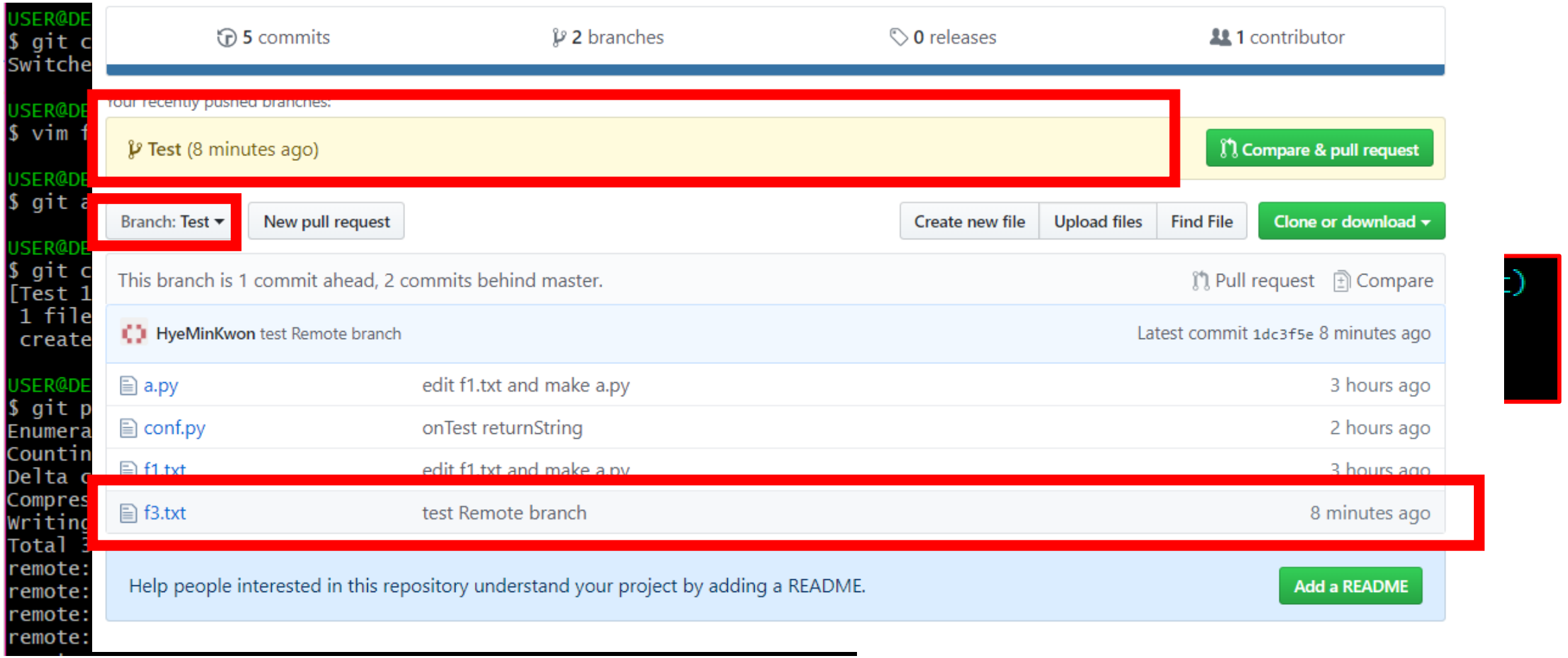
```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (master)
$ git checkout Test
Switched to branch 'Test'

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ vim f3.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git add f3.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git commit -m "test Remote branch"
[Test 1dc3f5e] test Remote branch
 1 file changed, 1 insertion(+)
 create mode 100644 f3.txt

USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git push origin  Test
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 290 bytes | 290.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'Test' on GitHub by visiting:
remote:      https://github.com/HyeMinKwon/tobigs_12th/pull/new/Test
```

이번엔 Test branch에서 push

```
USER@DESKTOP-B1HHQDT MINGW64 ~/tobig12 (Test)
$ git branch -r
  origin/Test
  origin/master
```

Unit 04 | Github 실습

로컬저장소 -> 원격저장소로 push~!



Unit 04 | Github 실습

로컬저장소 -> 원격저장소로 push~!

```
USER@DE
$ git c
Switche

USER@DE
$ vim f

USER@DE
$ git a

USER@DE
$ git c
[Test 1
 1 file
 create

USER@DE
$ git p
Enumera
Countin
Delta c
Compres
Writing
Total 3
remote:
remote:
remote:
remote:
```

5 commits | 2 branches | 0 releases | 1 contributor

Your recently pushed branches:

Test (8 minutes ago) — Compare & pull request

Branch: Test | New pull request | Create new file | Upload files | Find File | Clone or download

This branch is 1 commit ahead, 2 commits behind master. — Pull request | Compare

HyeMinKwon test Remote branch — Latest commit 1dc3f5e 8 minutes ago

a.py	edit f1.txt and make a.py	3 hours ago
conf.py	onTest returnString	2 hours ago
f1.txt	edit f1.txt and make a.py	3 hours ago
f3.txt	test Remote branch	8 minutes ago

Help people interested in this repository understand your project by adding a README. — Add a README

원격저장소 -> 로컬저장소로 pull~!

변경된 원격 저장소를 로컬 저장소에 동기화(다른 사람들이 수정한 것을 내 local에 반영하기 위해) 하려면
git pull origin master 명령어 사용

-> origin의 내용이 master로 복사된다.

Cf) http프로토콜 사용할 때 – remote repo접근 시 github의 user name과 password를 입력

매번 비밀번호를 치기 귀찮으면

git config --global credential.helper ‘store --file 경로’
-> 해당 경로에 비밀번호가 저장된 파일이 생성(단, 파일로 저장되는 만큼 보안에 취약)

git config --global credential.helper 'cache --timeout=864000’
-> 특정 시간 동안 비밀번호를 물어보지 않는다

참고 자료 : https://git-scm.com/book/ko/v2/Git-%EB%8F%84%EA%B5%AC-Credential-%EC%A0%80%EC%9E%A5%EC%86%8C

< 참고>

원격저장소와 관련된 개념들

- github Fork : 타인의 github repository를 내 github repository에 그대로 복제 (원본 github repo와 연결되어 있다)
- pull request : fork한 파일을 수정하여 original github repository에 적용하고 싶을 때 original repo의 관리자에게 승인해달라고 보내는 요청
- clone : 특정 repository를 내 local repository에 복사하여 새로운 저장소를 만드는 것 (원본 repository를 origin[remote저장소의 별명]으로 저장)
- fetch : 원격 저장소의 내용을 그저 가져오기만 하기
- pull : 원격 저장소에서 fetch + merge.

Unit 05 | 과제

< 보충 자료 >

- Github io

  https://dreamgonfly.github.io/2018/01/27/jekyll-remote-theme.html

- Git stash
  - 아까 10페이지에서 설명하지 않은 부분!
  - 현재 branch에서 작업이 끝나지 않았는데 다른 branch로 checkout해야 할 때 이용 (branch에서 작업한 내용을 숨기는 작업)

  https://git-scm.com/book/ko/v1/Git-%EB%8F%84%EA%B5%AC-Stashing

- Git의 다양한 명령어

  https://medium.com/@joongwon/git-git-%EB%AA%85%EB%A0%B9%EC%96%B4-%EC%A0%95%EB%A6%AC-c25b421ecdbd

- ssh키 설정

  https://git-scm.com/book/ko/v1/Git-%EC%84%9C%EB%B2%84-SSH-%EA%B3%B5%EA%B0%9C%ED%82%A4-%EB%A7%8C%EB%93%A4%EA%B8%B0

- 협업을 하고 싶어요

  https://victorydntmd.tistory.com/91

Unit 05 | 과제

< 필수 과제 >

1. 팀장이 github에 새로운 remote repository를 만들고 collaborator에 팀원들을 추가 해주세요.
2. 하나의 파일을 생성해 주세요.
3. 팀장, 팀원 모두 각자의 이름 branch를 생성해 그 branch에서 하나의 파일을 팀원 모두 한 번 이상 수정 후 master에서 merge해 주세요.

branch예시) 대웅 ->Daewoong (O). 혜민 -> Grace (X)

< 제출 방법>

팀원 중 한 명이 각 조 이름과 작업한 github주소를 올리기

Ex) 1조. http://github.com/ABC/Tobig_test

Unit 05 | 과제

< 필수 과제 >

팀 배정(팀장. 팀원. 팀원)

1조 – 강민성. 김수빈. 김수아

2조 – 김주호. 김탁영. 김태욱

3조 – 김태한. 김효은. 박재민

4조 – 박진혁. 배유나. 신윤종

5조 – 윤기오. 이도연. 이세윤. 이승현

6조 – 이유진. 이홍정. 조민호. 최승호

Unit 05 | 과제

< 선택 과제 >

1. 자신의 github에 새로운 repository를 만든다.
2. 10주 정규 세미나 동안 하게 되는 과제들을 자신의 github에 계속 업로드한다.
3. Commit 단위를 잘 고려하면서 과제를 수행한다

Unit 05 | 과제

< Reference >

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

https://www.youtube.com/watch?v=eVo2lmkXaDc&list=PLuHgQVnccGMA8iwZwrGyNXCGy2LAAsTXk&index=11

http://www.datamarket.kr/xe/board_jPWY12/48448

http://www.datamarket.kr/xe/index.php?mid=board_jPWY12&page=2&document_srl=50526

https://victorydntmd.tistory.com/78?category=682764

https://docs.microsoft.com/en-us/azure/devops/learn/git/git-share-code

들어주셔서 감사합니다.