

GitHub

EunAh Lee, PhD



Contents

- **Introduction on VCS (Version control system)**
- **Git & Github**
- **Handling Repository, Directory, & Files**
- **Fork & Pull Request**
- **Branching**

Introduction on VCS (Version control system)

버전 관리 시스템, 형상 관리 도구

VCS (Version Control System)

- 의미 있는 변화들 (기능개선, 버그수정, 요구에 따른 수정사항 등)에 대한 변화들을 관리하는 도구
- 소스코드의 중요한 변화를 기록하는 행위
- 문제나 코드의 변경사항을 저장해서 과거의 상태를 열람&복원 가능
 - 어떤 문제가 발생했을 때 문제의 맥락을 파악할 수 있도록 도와주고, 변화에 실패했을 때 과거의 상태로 쉽게 돌아갈 수 있음
 - 실패에 대한 부담이 줄어들면서 작업이 원활하고 효율 증가됨
 - 백업, 협업과 같은 중대한 장점을 제공
 - 충돌 방지(?) → 충돌 관리 가능
- SVN, GIT이 오늘날 많이 사용되는 버전 관리 시스템의 두 축

SVN과 GIT의 차이점

SVN과 GIT은 모두 소스코드의 효율적 관리를 위한 버전관리시스템으로 비슷하게 작동하는듯 하나, 용어와 개념에서 다르게 동작하는 부분들이 있어서 주의 필요

- SVN과 GIT의 가장 큰 차이는 '분산'
- SVN은 중앙집중식 소스코드 관리방식/GIT은 분산 소스코드 관리방식
(GIT을 사용하는 경우 중앙 저장소가 폭파되더라도 분산되어 있는 로컬 저장소를 이용해 중앙 저장소를 복원할 수 있음)
- 그 외, 명령어가 반대의 개념을 포함하고 있어서 반대의 작용을 하는 경우가 있음

Git & Github

GIT & GITHUB

- **GIT**
 - 버전관리 시스템 제품 중 하나
 - 분산형 버전관리 시스템으로 분류
- **Github**
 - 버전관리 시스템인 Git을 이용하는 프로젝트들을 위한 원격 저장소를 제공하는 서비스
 - 오픈소스는 무료, 비공개 프로는 유료 정책으로 시작
 - 현재는 비공개도 무료
 - 저장소 크기의 제한이 없으나, 용량 한계는 있음 (파일 당 50MB 미만)
 - '누가' 개발했는지 위주로 저장되므로 사람중심 서비스 구성 - ID/PW로 저장소에 접근
 - 공개 SW에서 많이 사용됨
(cf. Gitlab은 폐쇄성이 있고 기업체 인트라넷에서 많이 사용)

GIT 설치 & 초기화 설정

www.git-scm.com

- 설치 진행은 default로 진행
- 설치가 모두 끝나고 나서 cmd 창에 git을 입력하면 설치여부 확인 가능
- Git bash 생성됨
- cmd 창에서도 git 사용 가능

GUI 기반 툴

- <http://www.sourcetreeapp.com>

GIT Guide

- Progit (<http://www.git-scm.com>)
- 한국어판: <http://www.git-scm.com/book/ko/v1> (by Insub Lee)

GIT의 작동 방식

- 소스코드 주고받기가 필요 없음
- 같은 파일을 여러 명이 동시에 작업 - 병렬개발 가능
- 버전관리 용이 - 생산성 증가 (그룹 작업 뿐만 아니라 개인 개발에서도 유용)
- 소스코드의 수정내용을 commit 단위로 관리, 패치 형식으로 배포 가능
- 프로그램의 변동과정을 체계적으로 관리 가능
- 언제든지 지난 시점으로 되돌아갈 수 있음
- 분산 관리 방식이므로 인터넷이 연결되지 않은 곳에서도 local repository를 통해 개발을 진행할 수 있음
- 큰 변화가 필요한 수정사항 등의 실험적 개발을 추가하는 경우, branch를 통해 충분히 실험을 한 뒤 본 프로그램에 merge 하는 방식으로 개발을 진행할 수 있음
- “작업한 내용을 stage에 올려서 local repository에 commit 하고 이를 push 해서 remote repository로 올린다”

GIT 관련 주요 용어

- Repository
- Checkout
- Stage
- Commit
- Tag
- Push
- Pull
- Branch
- Merge

Repository

- Source code가 저장되어 있는 여러 개의 branch 들이 모여 있는 디스크 상의 물리적 공간
- Local repository와 remote repository로 나뉨
- 작업 시작 시 remote repository에서 local repository로 source code를 clone 해 가져오고 이후 source code를 변경한 다음 commit 수행 → 수행한 commit은 local repository에만 저장되고 push를 하기 전에는 remote repository에 반영되지 않음

이전에는 오픈소스 코드를 배포할 때 버전 별 압축파일 형태로 배포했으나, git이 등장한 이후 저장소를 통해 배포하는 경우가 많아짐

- 오픈소스 코드를 구하려고 사이트에 방문하면 git이나 http 프로토콜로 시작하는 저장소 주소 한줄만 쓰여있는 경우를 자주 접할 수 있음

공개/비공개

- 공개된 repository가 있는 반면, 인증된 사용자만 접근할 수 있는 비공개 repository도 있음
- 공개 repository일지라도 저장소의 download만 가능하고 수정된 코드를 저장소에 반영하기 위해서 저장소관리자의 허가가 필요한 경우가 대부분

Checkout

특정 시점이나 브랜치의 소스코드로 이동하는 것을 의미

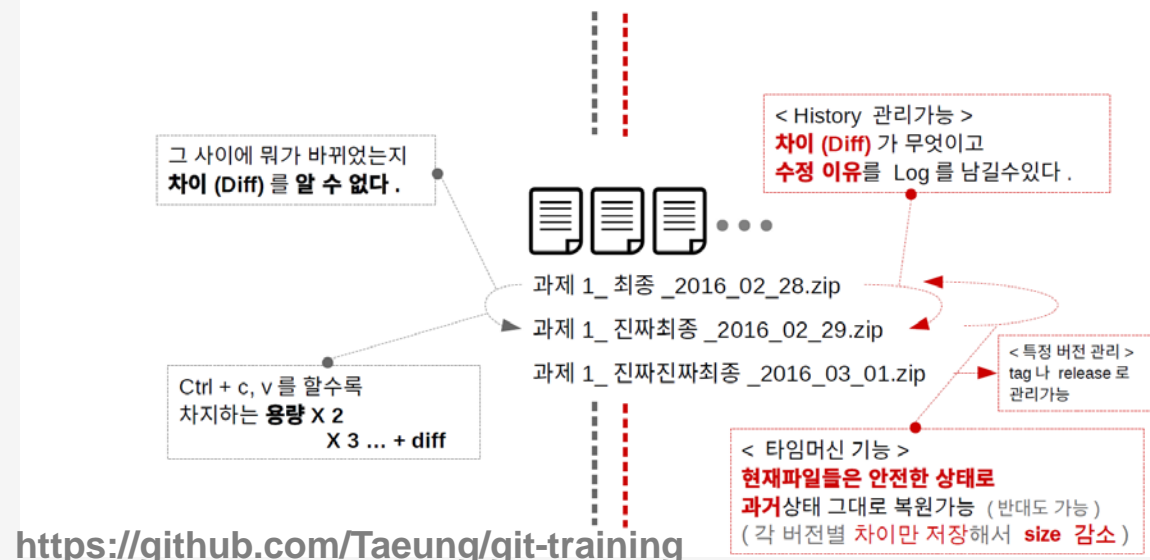
Checkout 대상은 branch, commit, 그리고 tag

Checkout을 통해 과거 여러 시점의 소스코드로 이동 가능

- c.f. - SVN에서의 checkout은 원격저장소의 파일을 작업하기 위해 로컬로 가져오면서 동시에 다른 사람이 수정할 수 없도록 하는 것임
 Source code management tool checkout과는 전혀 다른 의미임

Ctrl+c, v 나 Alzip 압축파일 관리법

Source code management tool

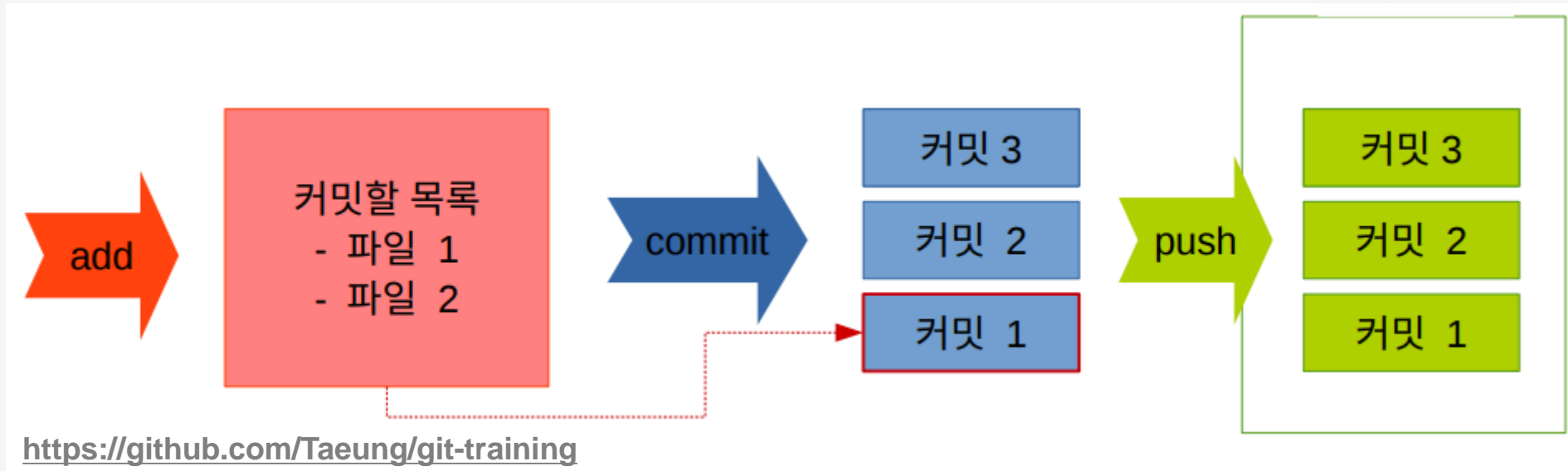


<https://github.com/Taeung/git-training>

Stage

작업한 내용이 올라가는 임시 저장 영역

이 영역을 이용하여 작업한 내용 중 commit에 반영할 파일만 선별하여 commit 수행 가능



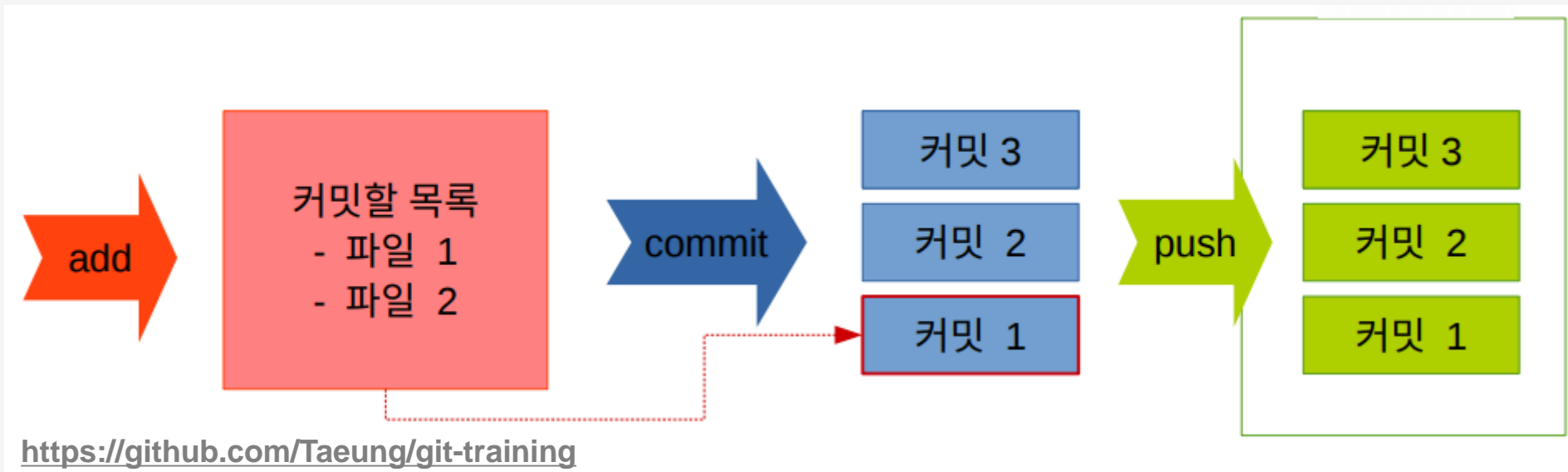
Commit

작업한 내용을 로컬 저장소에 저장하는 과정

의미 있는 변경의 단위를 이름

변경에 대한 설명을 commit 로그로 추가 가능 (ex. Bug_fix, Signin_func_added)

- 원격 저장소와 연동된 자동화 시스템을 사용하고 있는 경우, 이 자동화시스템이 인식할 수 있도록 엄격한 형식에 맞춰 커밋 로그를 작성해야할 수도 있음



태그 (Tag)

- 커밋의 임의 위치에 쉽게 찾아갈 수 있도록 붙여 놓은 이정표
- 태그가 붙여진 커밋은 commit ID 대신 태그명을 입력하여 쉽게 checkout 할 수 있음

Push

- 로컬 저장소(local repository)의 내용 중 원격 저장소(remote repository)에 반영되지 않은 commit을 보내는 과정

Pull

- Push와 반대로 remote repository에 있는 내용 중 local repository에 반영되지 않은 내용을 가져와서 저장하는 과정
 - 다른 팀원이 변경하고 push한 내용을 내 local repository에 반영하는 과정
 - Push 과정에서 충돌이 일어나서 push가 거절되는 경우, 우선 pull을 통해 원격 저장소의 변경 내용을 반영한 뒤 다시 push를 시도해야 함

Ctrl+c, v 나 Alzip 압축파일 관리법



Git 배우는데 **시간소비**하느니
Code **한줄이라도** 더 개발 ..)

Source code management **tool**

좋은건 알겠는데 ..

Git 을 쓸 **이유가 부족** ..

Wants 는 맞지만 **Needs 는 아니야**



과제 1_ 최종 _2016_02_28.zip

과제 1_ 진짜최종 _2016_02_29.zip

과제 1_ 진짜진짜최종 _2016_03_01.zip

Git 실습 시작 - Git bash에서

미리 저장되어 있을 지 모를 계정정보를 삭제 (처음 설치 시 생략 가능)

```
#git config --global --unset credential.helper  
#git config --system --unset credential.helper
```

나의 Github 계정 정보입력

```
#git config --global user.email "본인 이메일"  
#git config --global user.name "본인 이름"
```

Home 경로로 이동

```
#cd ~ (엔터)
```

파일 관리는 리눅스 명령어 사용

- 간단한 리눅스 명령어 참고

<https://github.com/Violet-Bora-Lee/linux-survival-for-korean>

Git 상태확인 명령어
(중간중간 치면서 수시로 확인하자)

```
# git show  
# git log  
# git shortlog  
# git diff  
# git status
```

Handling Repository, Directory, & Files

Repository 생성 & 연결

Git remote

- **#git remote** 현재 local 저장소에 등록된 remote repository 확인
- **#git remote add [URL]** URL의 remote repository를 local 저장소에 등록
- **#git remote add [name] [URL]** URL의 remote repository를 [name]이라는 이름으로 등록 (이렇게 하면 이후로는 name으로 handling 가능)
- **#git remote rename [current_name] [new_name]** 현재의 이름을 [new_name]으로 변경
- **#git remote rm [repository_name]** repository를 삭제

Git fetch

- **#git fetch [name]** local에는 없지만 remote에 있는 data를 모두 가져오는 과정
- 자동으로 merge 하지는 않음. Local에서 하던 작업을 정리하고 나서 수동으로 merge해야함

Git pull remote repository에서 data를 모두 가져와서 자동으로 local branch와 merge

- Clone한 서버에서 data를 가져오고 그 data를 자동으로 현재 작업 code와 merge 시킴

Git clone [URL]

- **#git clone [URL]** 자동으로 remote repository를 origin 이라는 이름으로 추가함
- **#git clone [URL] [name]** remote repository를 [name]이라는 이름으로 추가함
- 자동으로 local의 master 브랜치가 remote repository의 master branch를 추적하도록 만듦

Summary - Remote repository와 연결하는 방법

#git remote add [name] [URL]

- The repository was just connected & nothing happens

#git fetch [name]

- 자동으로 merge 되지는 않음

#git pull

- Fetch + merge
- 변경분만 가져와서 merge

#git clone [URL]

- Remote add + pull
- 프로젝트 파일 전체를 가져와서 merge

#git clone [URL] [name]

- URL 저장소를 name 이라는 이름으로 가져옴.
- 이후로는 name 으로 handling 가능

Handling files

#git add [file name]

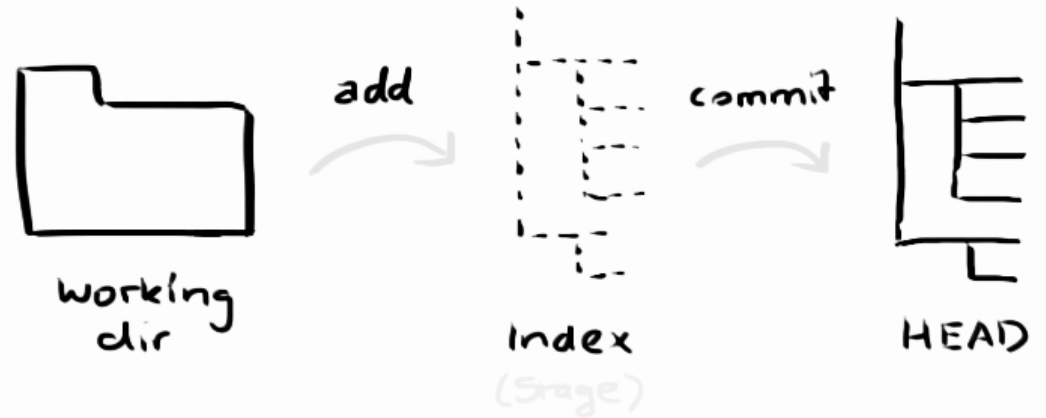
#git rm [file name]

- 다음과 같은 식으로 regular expression 사용 가능

#git rm *~ 이 명령어는 ~로 끝나는 모든 파일을 삭제함

#git commit -m '설명'

- Commit수행하면 history로 남아 되돌아갈 수 있음
- 한번도 commit한 적 없는 data는 git으로 복구 불가



<https://damienfremont.com/2015/09/01/git-the-simple-guide/>

Regular Expression

- *는 문자가 하나도 없거나 하나 이상을 의미
- [abc]는 중괄호 안에 있는 문자 중 하나를 의미
- ?는 문자 하나
- [0-9] 처럼 중괄호 안의 캐릭터 사이에 하이픈을 사용하면 그 캐릭터 사이에 있는 문자 하나를 의미

Commit 수정

Commit 되돌리기

한번 되돌리면 없었던 일이 되고 commit history에 기록되지 않아 복구가 불가능

#git reset HEAD 이렇게 하면 수정 이전의 파일로 덮어썼기 때문에 수정했던 내용은 전부 사라짐 수정한 내용을 완전히 삭제해야 할 때에만 사용

Commit 수정하기

너무 일찍 커밋 했거나 간단한 한가지를 빠뜨렸을 때 다시 commit 하고 싶으면 - amend 옵션을 사용

<예시>

#git commit -m 'initial commit'

#git add forgotten_file'

#git commit -amend 편집기 실행되고 여기서 수정하고 마치면 3줄의 명령어가 하나의 commit으로 저장됨

Remote Repository와 연동

Git push origin master

- `#git push -u origin master`

- u 옵션은 원격저장소로부터 업데이트를 받은 후 push를 한다는 의미 (습관적 사용 권장)

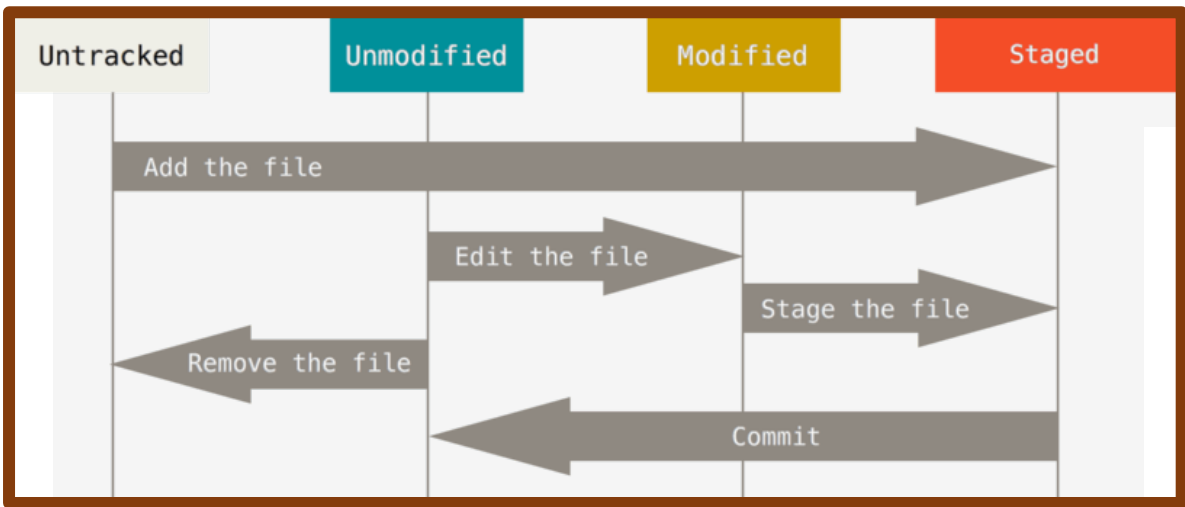
- Clone 한 저장소에 쓰기 권한이 있는 경우만 실행 가능

- Clone 한 후 아무도 remote 저장소에 push하지 않았을 경우에만 충돌문제가 안 생김

- 여러 명이 개발하는 repository인 경우 push 하기 전에 pull (또는 fetch 후 merge)해서 내 local 저장소가 remote repository와 동일한 상태로 만든 후에 push를 해야함

git push - u origin master

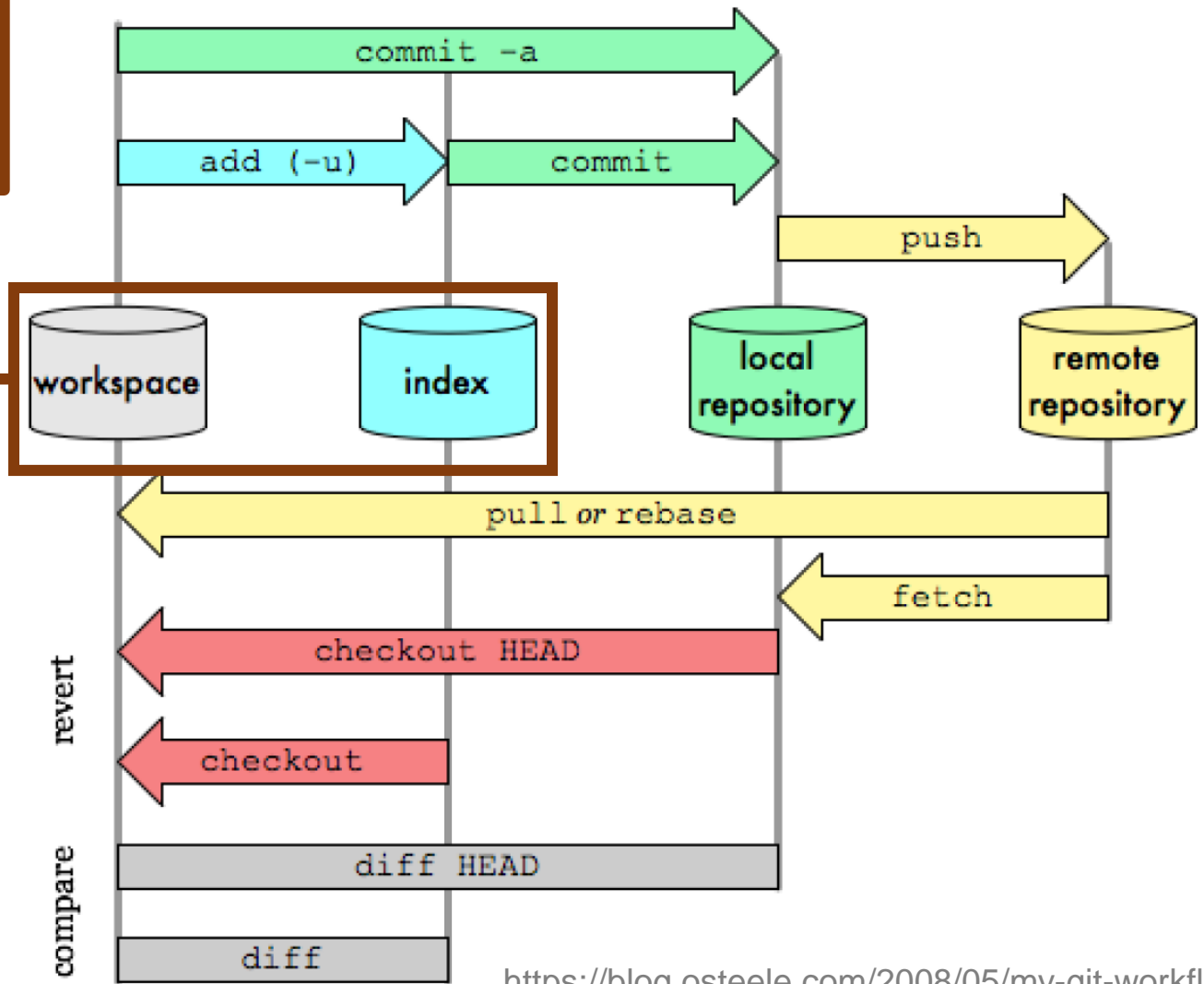
- origin master 저장소로 연결되며, 다음엔 git push/pull 만 해도 연결됨



git-scm.com

Git Data Transport Commands

<http://osteele.com>



<https://blog.osteele.com/2008/05/my-git-workflow/>

Branching

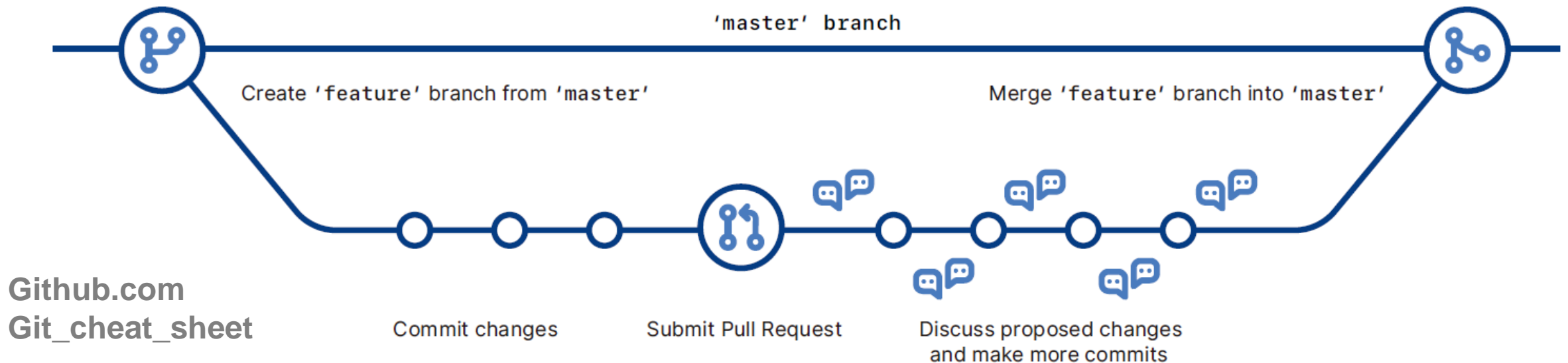
Branch의 개념

Branch 모델이 git의 최고의 장점

Git의 브랜치는 매우 가볍고 쉽게 만들고 쉽게 이동할 수 있음

- Git의 branch는 특정 commit에 대한 reference
- Branch를 많이 만들어도 메모리나 디스크 공간에 부담이 되지 않음

Branch는 하나의 커밋과 그 부모 커밋들을 포함하는 작업 내역과 같음



Merge

하나의 branch를 다른 branch와 합치는 과정

- 두 branch를 합쳐 한 branch로 만드는 3-way merge

Merge의 대상이 되는 두 branch는 주종관계가 성립

- 'A branch를 B branch에 병합' 하는 작업과 'B branch에 A branch를 병합' 하는 작업은 서로 다른 작업

Merge도 commit의 한 종류

- 일반적인 commit은 조상 commit이 하나인데 반해, merge는 조상 commit이 둘 이상인 경우
- 3-way merge는 서로 다른 두 commit으로부터 하나의 새로운 commit을 생성하는 작업
- Merge 과정에서 한 파일의 같은 부분을 서로 다르게 수정한 경우 충돌이 발생하며, merge가 일시 정지됨 - 이 경우 충돌이 발생한 부분을 직접 수정하거나 merge tool등을 활용하여 충돌을 해결한 후 merge를 진행

일반적인 Merge 작업 수행 방법

- Merge 작업은 각 팀원이 수행하기 보다는 project manager가 일괄적으로 수행하는 것이 일반적 (한 branch의 작업이 끝나면 PM에게 Merge Request를 보내고, PM은 병합하기 전 해당 branch를 작업한 개발자와 함께 code review를 진행한 뒤 이상이 없으면 mater branch에 해당 branch를 병합하는 작업을 수행)

#git diff [원래 branch] [작업한 branch]

- 변경내용을 병합하기 전에 어떻게 바뀌었는지 비교해보는 방법

Branch

Commit을 단위로 구분된 source code의 타임라인에서 분기해서 새로운 commit을 쌓을 수 있는 가지를 만드는 것 혹은 그 가지를 branch라고 함

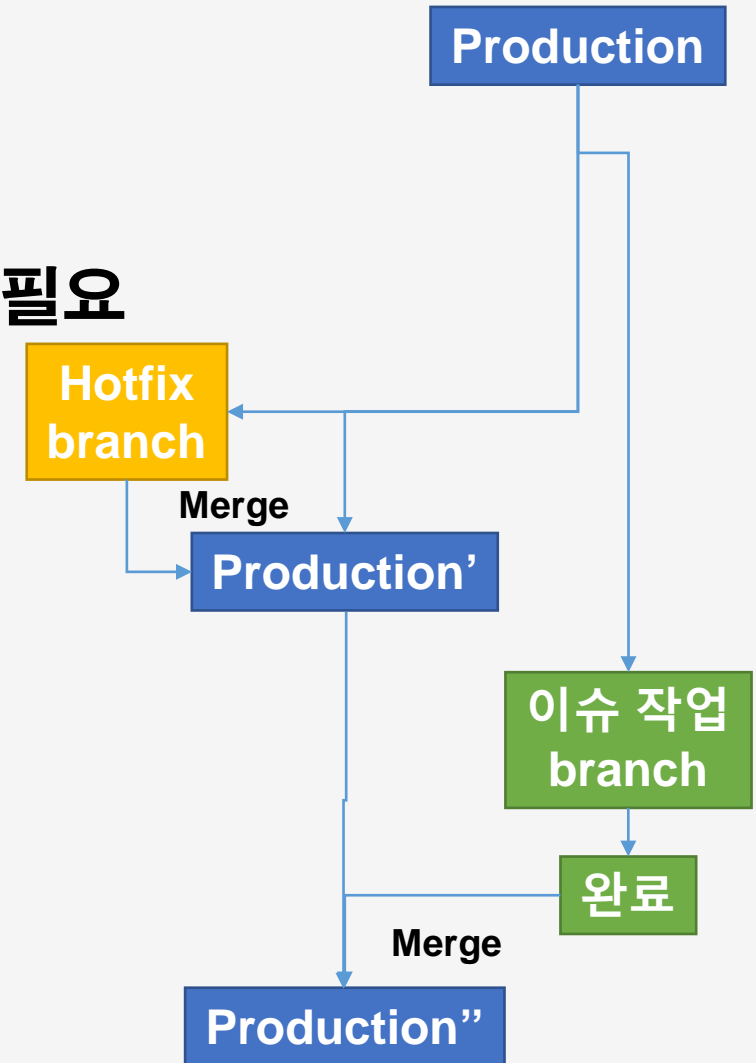
- Master branch - 개발의 주축이 되는 branch. 모든 branch는 master branch에서 분기되어 최종적으로 다시 master branch에 merge되어 개발이 진행됨

대개 프로젝트에서 branch는 프로그램 전체를 몇 개의 단위로 나눈 topic 단위로 생성함

- 각 topic branch에서 개발자 별로 새로운 branch를 생성하여 작업을 함
- 개발자 별 작업한 내용은 topic branch에 병합되며 최종적으로 master branch에 병합되어 하나의 토픽이 종료됨

Branch를 활용하는 실제 개발 과정의 예

1. 작업중인 web site가 있음
2. 이슈를 처리할 새 branch를 하나 생성
3. 새로 만든 branch에서 작업 중
4. 이때 중요한 급한 문제가 생겨서 그걸 해결하는 hotfix 필요
 - I. 새로운 이슈를 처리하기 이전의 branch(Production)로 이동
 - II. Hotfix branch를 새로 하나 생성
 - III. 수정한 hotfix 테스트를 마치고 운영 브랜치로 merge
5. 다시 작업하던 브랜치로 옮겨 가서 하던 일 진행



Github 란



Git 이라는 도구를 응용한 사이트

각종 Remote repository (원격저장소) 들의 집합소

Branch 관련 명령어

#git branch local branch 확인

#git branch -r 서버 branch 확인

#git checkout [브랜치 명] 브랜치로 이동

#git checkout -b [브랜치명] 브랜치를 만들고 바로 이동

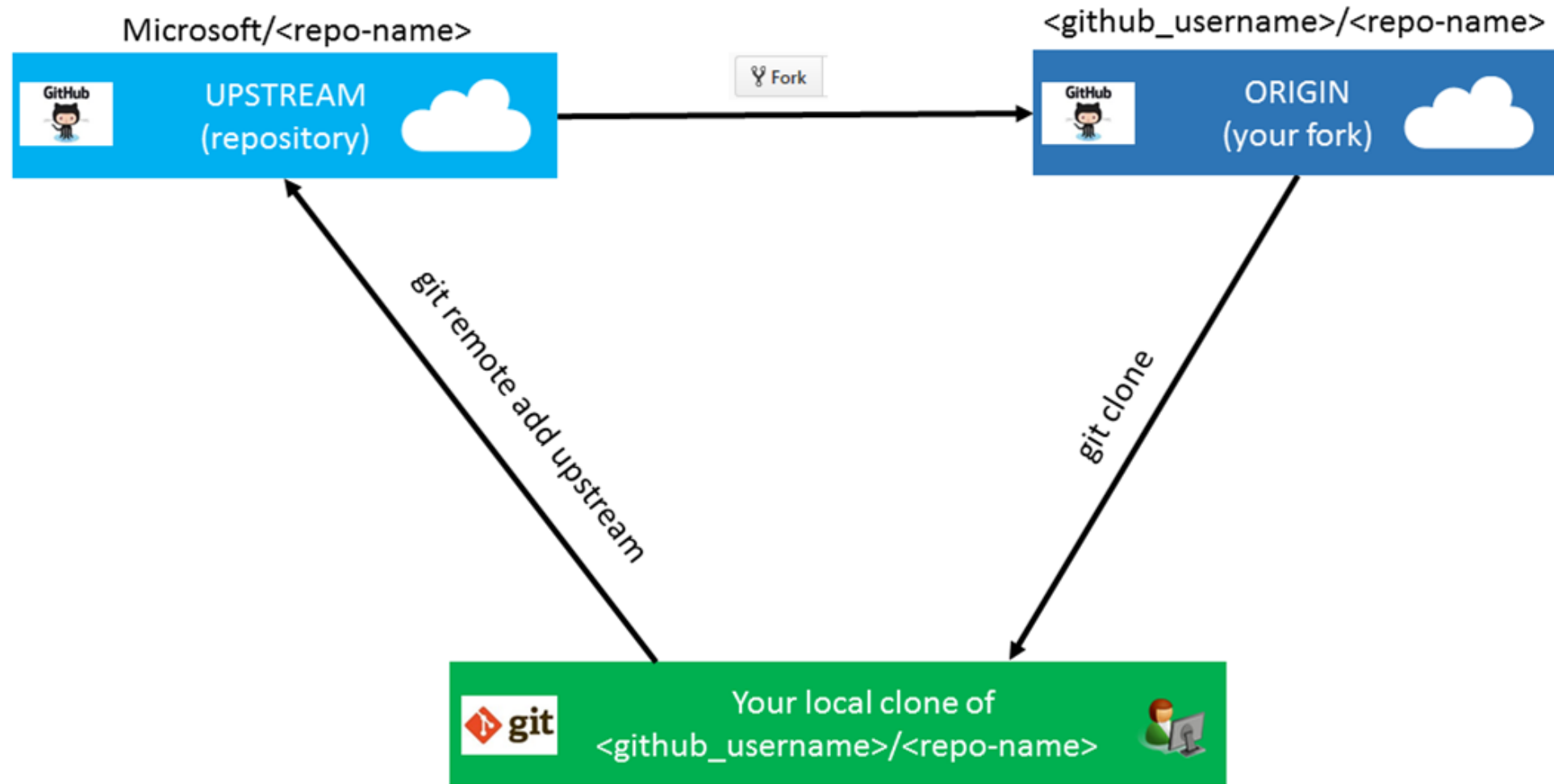
#git branch -d test test라는 이름의 branch 를 삭제

Fork & Pull Request

Fork

- Git에서 fork는 리눅스에서의 fork와 유사한 개념을 가짐
 - 리눅스에서 새로운 프로세스를 실행하는 방법 중 하나인 fork는 이미 동작중인 process를 복제해서 새로운 process를 만드는 것을 의미
- Git의 fork는 저장소를 복제해서 새로운 독립된 저장소를 만드는 작업
- 오픈소스 프로젝트 일지라도 소스코드의 메인스트림에 commit을 반영하는 commiter의 권한을 아무에게나 주지않고 대개 통과해야 하는 기준이 있음 (대규모 오픈소스 프로젝트일수록 이 기준이 더 까다로움)
- 공개된 오픈소스 프로젝트를 자신이 프로젝트 매니저가 되어 입맛대로 수정하고 싶을 때 사용하는 기능이 fork
- 원래의 오픈소스 repository(upstream)를 자신의 remote repository (origin)로 복제하여 수정작업 후 원래의 원격 저장소에 push 하여 변화분의 반영을 시도하도록 도입한 기능임

Repository first time set up



Fork a repository (Upstream)

The screenshot shows the GitHub interface for the repository 'dhrim / hongik_2020'. The top navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The repository name is 'dhrim / hongik_2020'. The 'Fork' button is highlighted with a red box. The repository has 1 Watch, 3 Stars, and 3 Forks. The repository content shows a README.md file with the following text:

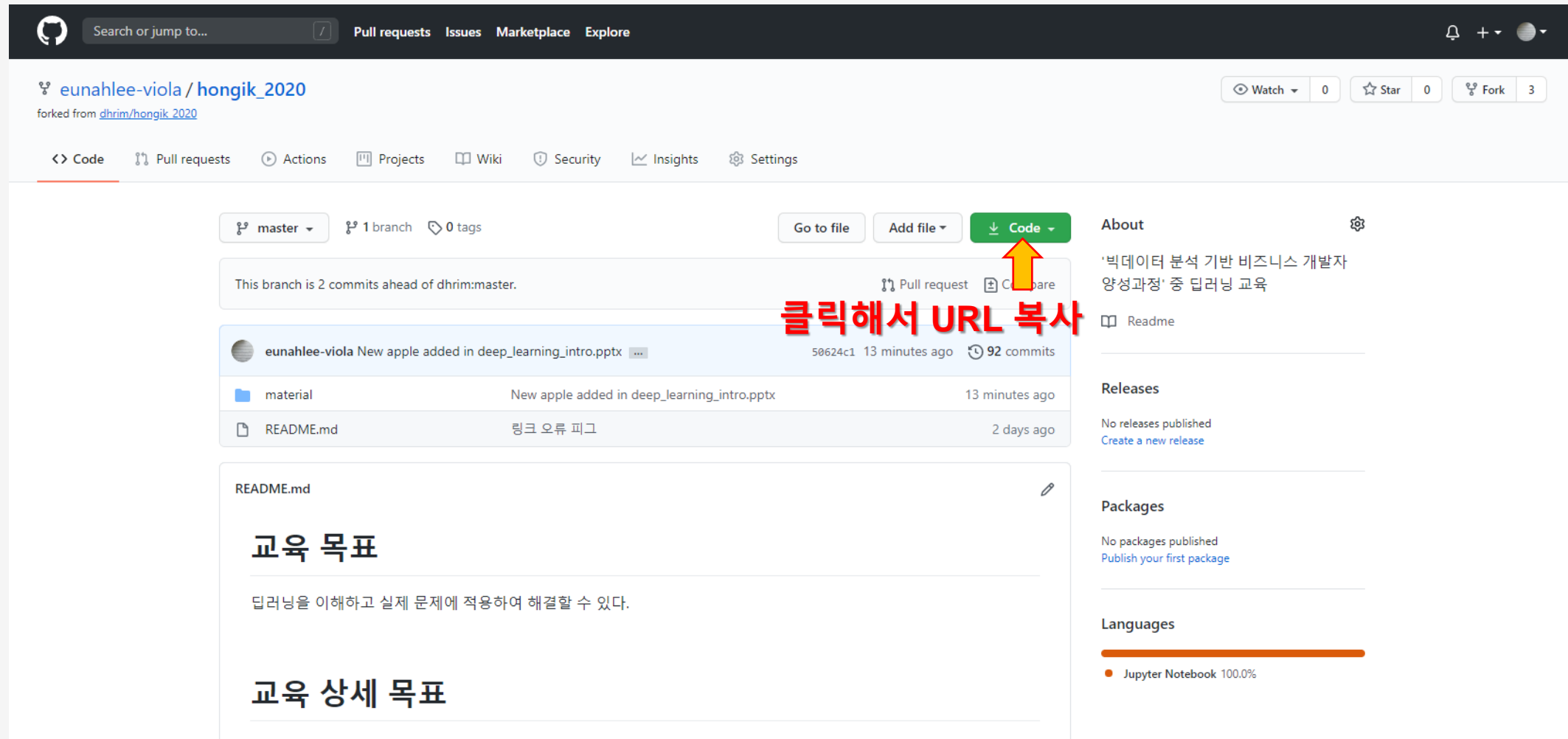
교육 목표

딥러닝을 이해하고 실제 문제에 적용하여 해결할 수 있다.

교육 상세 목표

- 딥러닝을 이해한다.
- 딥러닝을 적용할 수 있는 문제를 이해한다.
- 딥러닝을 실 문제에 적용하는 방법을 이해한다.

Forked to my remote repository (Origin)



Search or jump to... Pull requests Issues Marketplace Explore

eunahlee-viola / hongik_2020
forked from dhrim/hongik_2020

Watch 0 Star 0 Fork 3

<> Code Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

This branch is 2 commits ahead of dhrim:master.

Go to file Add file Code

클릭해서 URL 복사

eunahlee-viola New apple added in deep_learning_intro.pptx 50624c1 13 minutes ago 92 commits

material	New apple added in deep_learning_intro.pptx	13 minutes ago
README.md	링크 오류 피그	2 days ago

README.md

교육 목표

딥러닝을 이해하고 실제 문제에 적용하여 해결할 수 있다.

교육 상세 목표

교육 상세 목표

About

'빅데이터 분석 기반 비즈니스 개발자
양성과정' 중 딥러닝 교육

Readme

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages

Jupyter Notebook 100.0%

Clone, edit, & add

#git clone [URL]

[브랜치 만들어서 작업 가능]

[수정할 파일에 작업 후 저장]

#git status

#git add [directory/파일명]

```
MINGW64:/c/Users/user/hongik_2020/material/deep_learning
user@DESKTOP-BAJQSVJ MINGW64 ~ (master)
$ cd ~

user@DESKTOP-BAJQSVJ MINGW64 ~ (master)
$ git clone https://github.com/eunahlee-viola/hongik_2020.git
Cloning into 'hongik_2020'...
remote: Enumerating objects: 83, done.
remote: Counting objects: 100% (83/83), done.
remote: Compressing objects: 100% (74/74), done.
remote: Total 491 (delta 35), reused 19 (delta 9), pack-reused 408
Receiving objects: 100% (491/491), 398.86 MiB | 13.88 MiB/s, done.
Resolving deltas: 100% (222/222), done.
Updating files: 100% (93/93), done.
```

```
MINGW64:/c/Users/user/hongik_2020/material/deep_learning
user@DESKTOP-BAJQSVJ MINGW64 ~/hongik_2020 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   material/deep_learning/deep_learning_intro.pptx

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

user@DESKTOP-BAJQSVJ MINGW64 ~/hongik_2020 (master)
$ git add material/deep_learning/deep_learning_intro.pptx

user@DESKTOP-BAJQSVJ MINGW64 ~/hongik_2020 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   material/deep_learning/deep_learning_intro.pptx
```

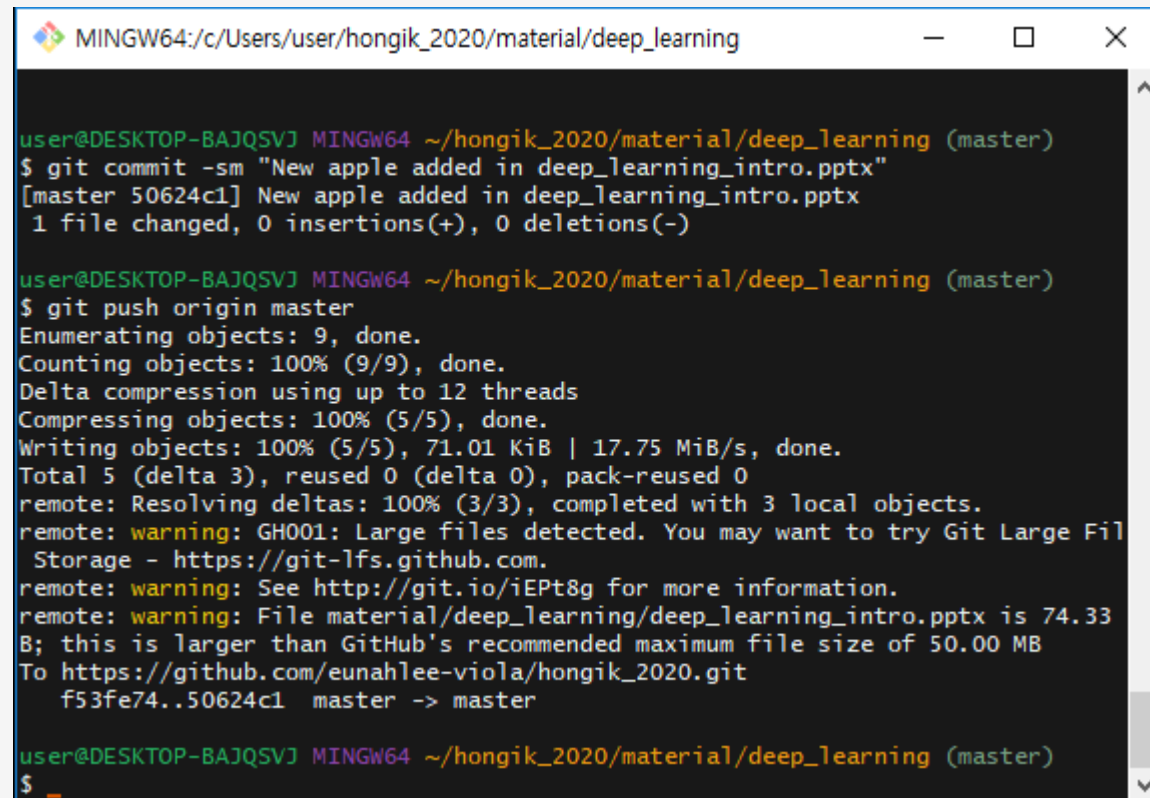
Commit & Push

#git commit -sm “작업이름”

[commit으로 변화분 확정]

#git push origin master

[이 과정은 파일을 upstream이 아닌 origin에 올리는 과정임]

A screenshot of a terminal window titled 'MINGW64: c:/Users/user/hongik_2020/material/deep_learning'. The terminal shows the execution of two git commands. The first command is 'git commit -sm "New apple added in deep_learning_intro.pptx"', which results in a commit with hash 50624c1. The second command is 'git push origin master', which successfully pushes the commit to the remote repository. The output of the push command includes progress bars for enumerating, counting, compressing, and writing objects, as well as a warning about a large file (74.33 MB) exceeding GitHub's recommended maximum file size of 50.00 MB. The terminal ends with a prompt '\$ _'.

Pull request - 내 github repository에서 하는 과정임!

Search or jump to... Pull requests Issues Marketplace Explore

eunahlee-viola / hongik_2020
forked from dhrim/hongik_2020

Watch 0 Star 0 Fork 3

<> Code **Pull requests** Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

Go to file Add file Code

This branch is 2 commits ahead of dhrim:master. Pull request Compare

eunahlee-viola New apple added in deep_learning_intro.pptx 50624c1 30 minutes ago 92 commits

material	New apple added in deep_learning_intro.pptx	30 minutes ago
README.md	링크 오류 피그	2 days ago

README.md

교육 목표

딥러닝을 이해하고 실제 문제에 적용하여 해결할 수 있다.

교육 상세 목표

- 딥러닝을 이해한다.

About

'빅데이터 분석 기반 비즈니스 개발자 양성과정' 중 딥러닝 교육

Readme

Releases


No releases published
[Create a new release](#)


Packages

No packages published
[Publish your first package](#)

Languages

Jupyter Notebook 100.0%


 Search or jump to... / Pull requests Issues Marketplace Explore

 eunahlee-viola / hongik_2020
forked from dhrim/hongik_2020

Watch 0 Star 0 Fork 3


<> Code Pull requests Actions Projects Wiki Security Insights Settings


Filters is:pr is:open Labels 9 Milestones 0 **New pull request**



Welcome to pull requests!

Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should [create a pull request](#).

 **ProTip!** `no:milestone` will show everything without a milestone.

© 2020 GitHub, Inc. Terms Privacy Security Status Help  Contact GitHub Pricing API Training Blog About

Pull request 확인 - Upstream repository에서 하는 과정!

여기에서 내가 pull request한 내용이 반영되어 있는지 확인!

dhrim / hongik_2020

<> Code Issues Pull requests 1 Actions Projects Wiki Security Insights

master 1 branch 0 tags

Go to file Add file Code

dhrim 링크 오류 피그 e6201eb 2 days ago 90 commits

material	Add files via upload	16 days ago
README.md	링크 오류 피그	2 days ago

README.md

교육 목표

딥러닝을 이해하고 실제 문제에 적용하여 해결할 수 있다.

교육 상세 목표

- 딥러닝을 이해한다.
- 딥러닝을 적용할 수 있는 문제를 이해한다.
- 딥러닝을 실 문제에 적용하는 방법을 이해한다.


About
'빅데이터 분석 기반 비즈니스 개발자 양성과정' 중 딥러닝 교육


Readme

Releases
No releases published

Packages
No packages published

Languages
Jupyter Notebook 100.0%

 Search or jump to... / Pull requests Issues Marketplace Explore

 dhrim / hongik_2020

Watch 1

Star 3

Fork 3

<> Code ⓘ Issues **🔗 Pull requests 1** ▶ Actions 📁 Projects 📖 Wiki ⓘ Security 📄 Insights

Filters ▾ 🔍 is:pr is:open

🏷️ Labels 9 ➡ Milestones 0

New pull request


🔗 1 Open ✓ 0 Closed

Author ▾ Label ▾ Projects ▾ Milestones ▾ Reviews ▾ Assignee ▾ Sort ▾

🔗 Real Apple Images Added

#1 opened 28 minutes ago by eunahlee-viola

💡 ProTip! Type `g p` on any issue or pull request to go back to the pull request listing page.

© 2020 GitHub, Inc. Terms Privacy Security Status Help  Contact GitHub Pricing API Training Blog About



Search or jump to...



Pull requests

Issues

Marketplace

Explore



dhrim / hongik_2020

Watch 1

Star 3

Fork 3

<> Code

! Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

master

Commits on Sep 25, 2020

Merge pull request #1 from eunahlee-viola/master

Verified



fa4c5be



dhrim committed 17 minutes ago

New apple added in deep_learning_intro.pptx



50624c1



eunahlee-viola committed 5 hours ago

New apple in deep_learning_intro_PPT



f53fe74



eunahlee-viola committed 5 hours ago

Commits on Sep 23, 2020

링크 오류 피그

Verified



e6201eb



dhrim committed 3 days ago

Commits on Sep 9, 2020

Add files via upload

Verified



1dc647f



dhrim committed 16 days ago

Commits on Sep 8, 2020

segmentation 관련 예제 업데이트



a1f55e7



dhrim committed 17 days ago



Create a Repository

From scratch -- Create a new local repository

```
$ git init [project name]
```

Download from an existing repository

```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit]:[file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git branch new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branch_a into branch_b

```
$ git checkout branch_b
```

```
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$ git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

```
$ git pull --rebase
```

Push local changes to the origin

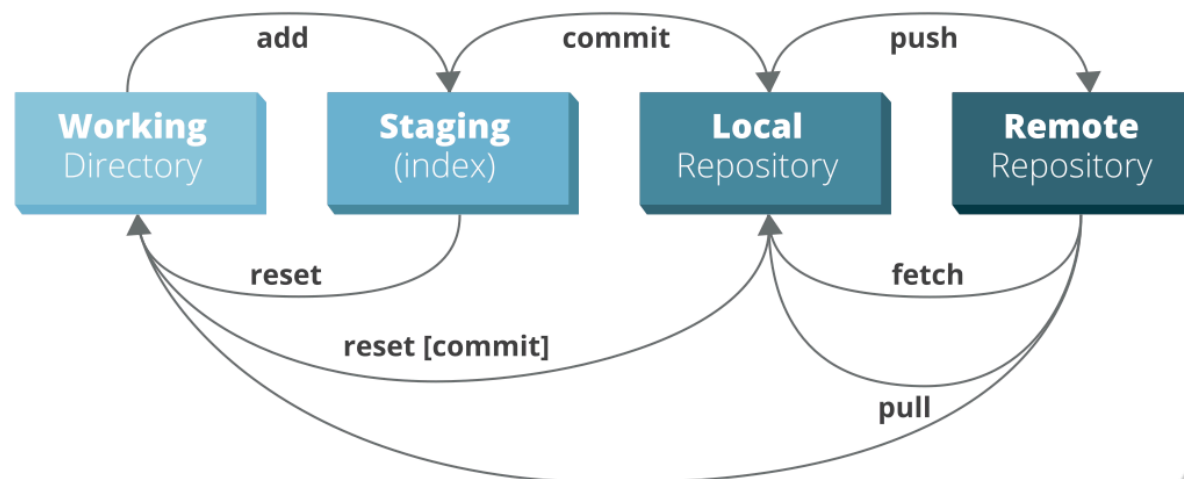
```
$ git push
```

Finally!

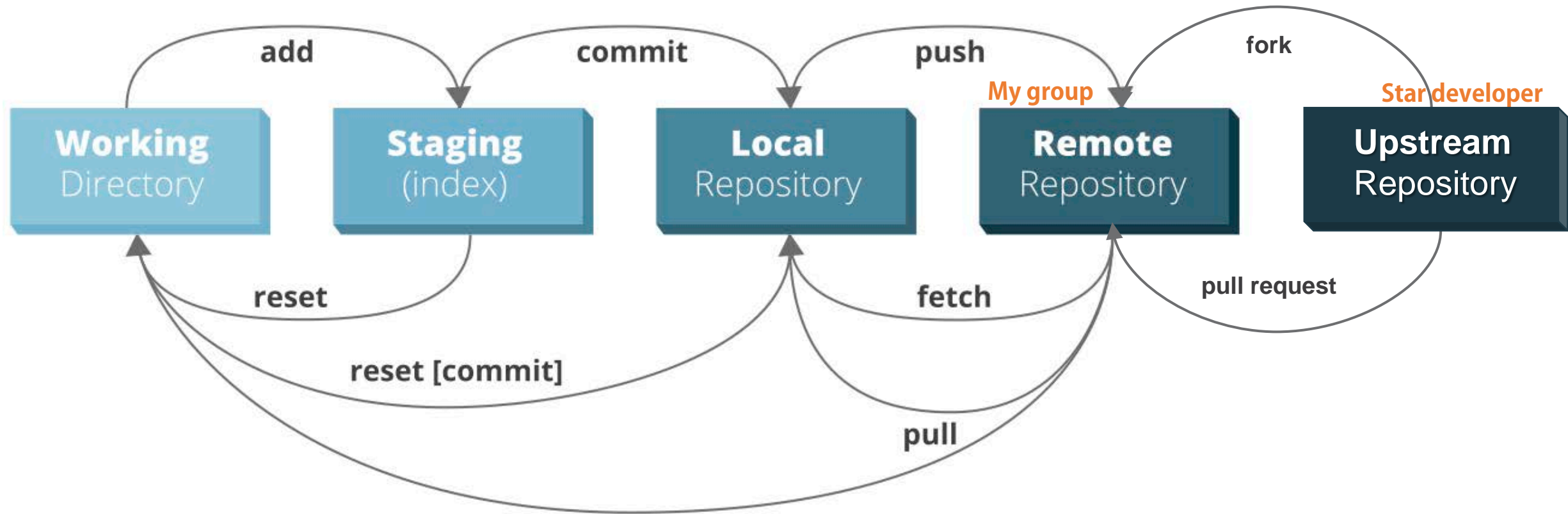
When in doubt, use git help

```
$ git command --help
```

Or visit <https://training.github.com/> for official GitHub training.



“작업한 내용을 stage에 올려서 local repository에 commit 하고
이를 push 해서 remote repository로 올린다”



Markdown 작성방법

<https://gist.github.com/ihoneymon/652be052a0727ad59601#file-gistfile1-md>

README.md

CurrLifeSci

2020 Fall Semester - KHU BME Graduate School Class

Notice

1. Quiz test at the beginning of every class ? 10 Pt
 - Quiz test becomes take home exam in case you are late or absent.
 - Take home exam score will be multiplied by 0.95
2. Discussion topics will be given (Tissue Reconstruction topics) ? 50 Pt
 - Short Briefing followed by discussion
3. Final Term Exam ? 100 Pt
4. Attendance ? 50 Pt

List of Topics

1. Fundamentals in Basic Biochemistry & Cell Biology
2. Introduction to Tissue Engineering & Regenerative Medicine
3. Developmental Tissue Reconstruction
4. Wound Healing & Regeneration
5. Natural Tissue Composition & Cell-ECM Interaction
6. Stem Cells & Cell-Based Therapy
7. Biomaterials
8. Mid-Term Exam
9. Mechano-transduction & Bioreactors
10. Discussions on Tissue Reconstruction
11. Regulation & Ethics
12. AI in Current Life Science
13. Machine Learning & Github
14. Deep Neural Network
15. Convolutional Neural Network
16. Final Exam

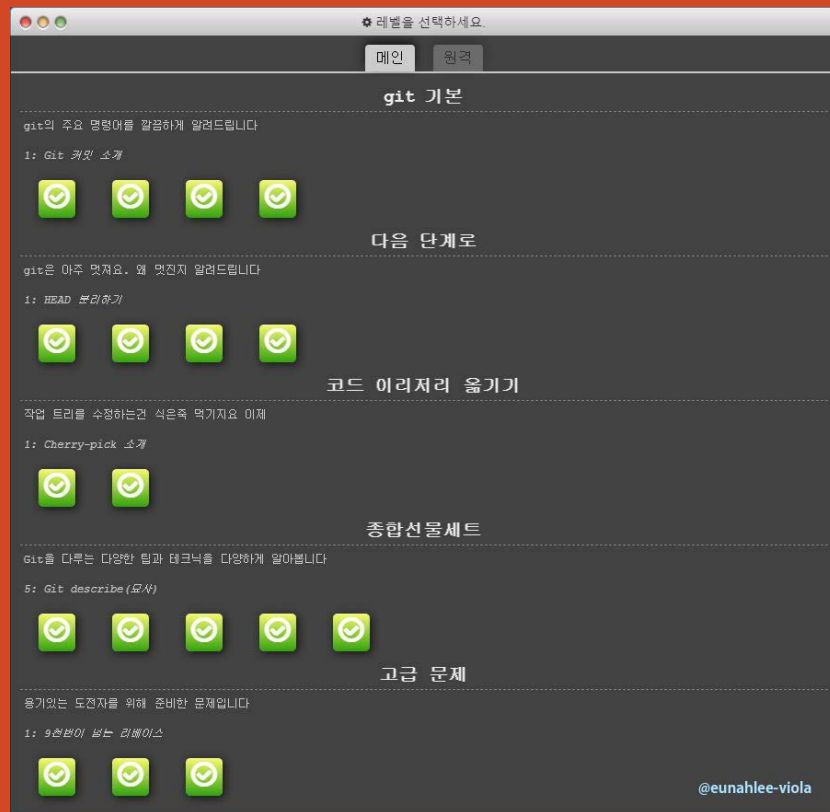
```

1  # CurrLifeSci
2  ### 2020 Fall Semester - KHU BME Graduate School Class
3
4  ## Notice
5
6      1. Quiz test at the beginning of every class ? 10 Pt
7          - Quiz test becomes take home exam in case you are late or absent.
8          - Take home exam score will be multiplied by 0.95
9      2. Discussion topics will be given (Tissue Reconstruction topics) ? 50 Pt
10         - Short Briefing followed by discussion
11      3. Final Term Exam ? 100 Pt
12      4. Attendance ? 50 Pt
13
14
15  ## List of Topics
16
17      1. Fundamentals in Basic Biochemistry & Cell Biology
18      2. Introduction to Tissue Engineering & Regenerative Medicine
19      3. Developmental Tissue Reconstruction
20      4. Wound Healing & Regeneration
21      5. Natural Tissue Composition & Cell-ECM Interaction
22      6. Stem Cells & Cell-Based Therapy
23      7. Biomaterials
24      8. Mid-Term Exam
25      9. Mechano-transduction & Bioreactors
26      10. Discussions on Tissue Reconstruction
27      11. Regulation & Ethics
28      12. AI in Current Life Science
29      13. Machine Learning & Github
30      14. Deep Neural Network
31      15. Convolutional Neural Network
32      16. Final Exam
33
34
35
```

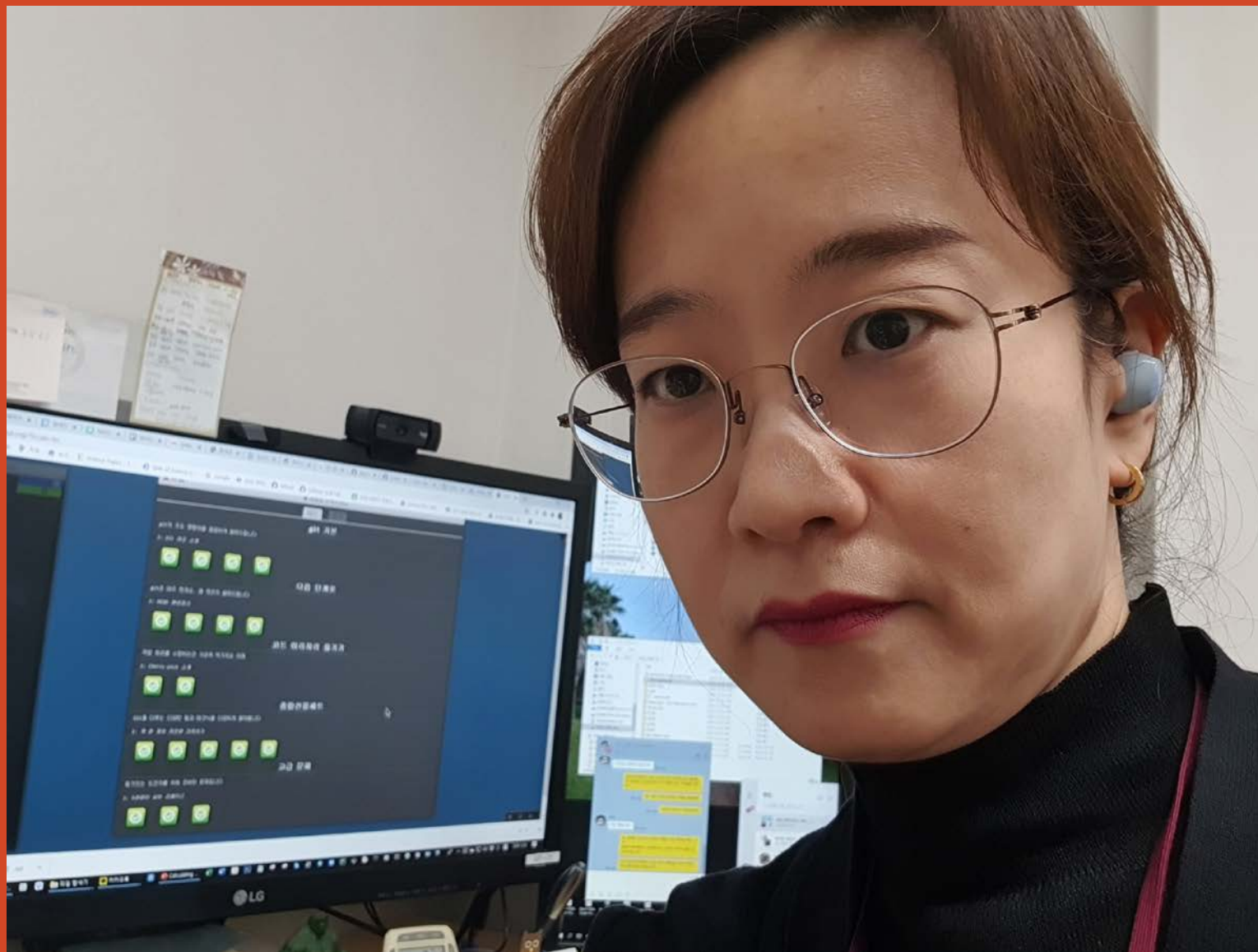
Assignment (기말고사 전까지)

<https://learngitbranching.js.org/?locale=ko>

위 학습 사이트의 모든 단계를 **본인의 컴퓨터로** clear 하고, 다음과 같이 모두 clear된 레벨의 상태를 화면에 띄워놓고 셀피를 찍어 제출 할것



제출할 셀피 예시 - 메인과 원격에 대해 각각 1장씩 찍어 제출할 것



References

1. Progit_v2.1.1
<https://git-scm.com/book/ko/v2>
2. 예제로 배우는 파이썬 프로그래밍
<http://pythonstudy.xyz/>
3. Git 교과서
<https://git.jiny.dev/>
4. Medium post - Git 기본 (이보라 개발자 post)
<https://medium.com/@violetboralee/git-%EA%B8%B0%EB%B3%B8-4ebc1e0f4f10>
5. Git/Github tutorial (송태웅, KOSSLAB)
<https://github.com/Taeung/git-training>
6. <https://trevorburnham.com/presentations/year-of-git/#/>

EOD