

# Introduction to Git & Github

조형기

<https://airlab.jbnu.ac.kr>

전북대학교 전자공학부

2024-1

# What is Git & Github?

---

# Version

흔한 대학생의 삶..

```
HW_조형기_v1.cpp  
HW_조형기_v2.cpp  
HW_조형기_v2(최종).cpp  
HW_조형기_v2(진짜최종).cpp  
...  
HW_조형기_final.cpp  
HW_조형기_final_final.cpp
```



## Version



- 내가 원하는 시점, 원하는 버전에 자유롭게 이동 가능
- 디자인, 웹 개발 등등 어떤 프로젝트든지 가능

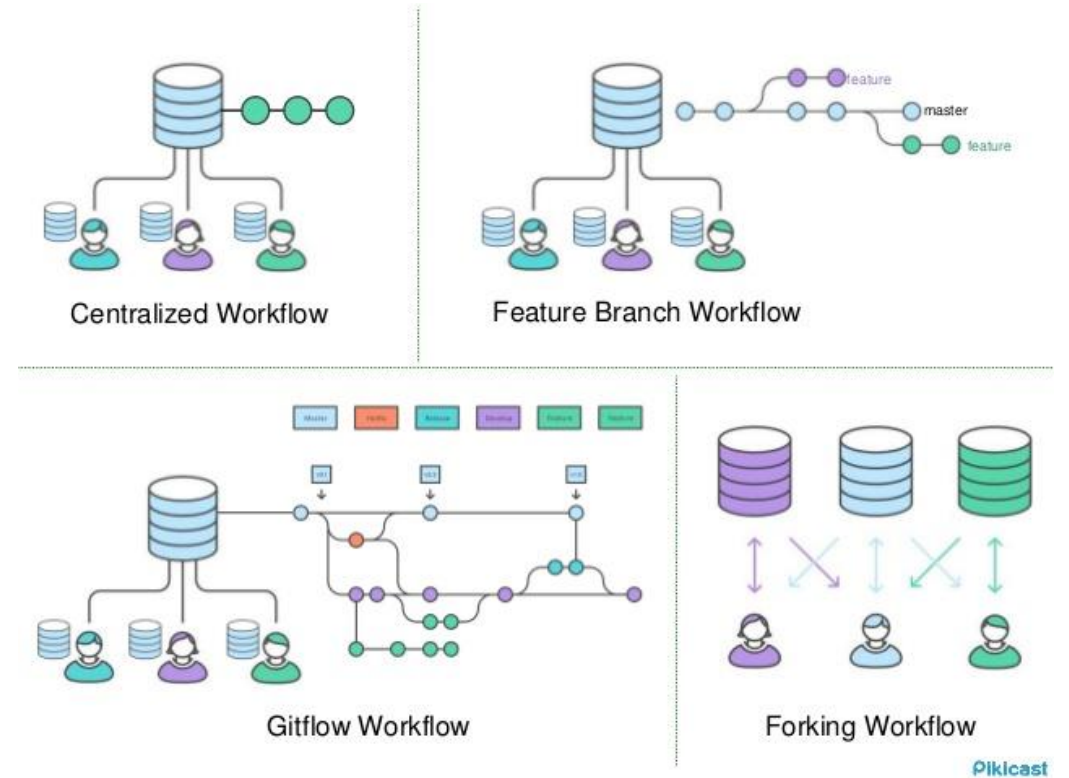
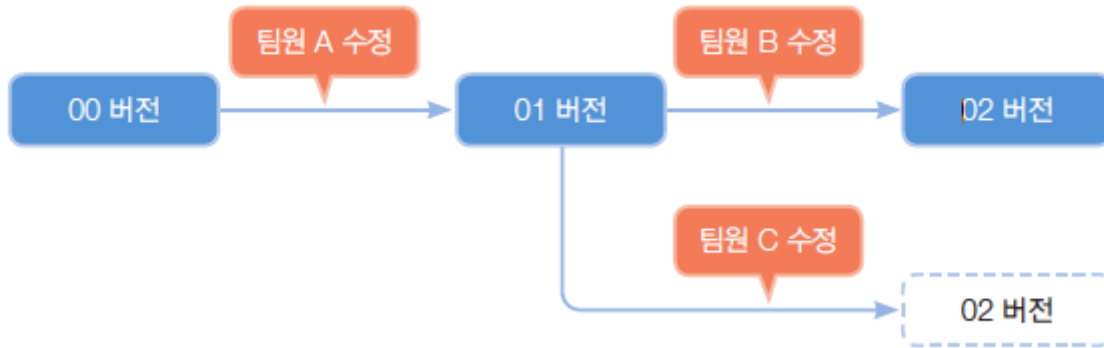
## Version

여러 명이 같이 하는 Project라면..?

```
Project_조형기_v1.cpp  
Project_조형기_v2.cpp  
Project_조형기_v2(최종).cpp  
Project_조형기_v2(진짜최종).cpp  
Project_김복대_v2(최종).cpp  
Project_팀1_합본.cpp  
...  
Project_조형기_final.cpp  
Project_조형기_final_final.cpp  
Project_팀1_합본_최종.cpp
```



# Version



- 여럿이 함께 작업하는 협업 프로젝트에서 더욱 강력함
- 따로 조금씩 작업하다가 원할 때 합치기 + 백업도 가능
- 기존 버전과 현재 버전을 비교하여 바뀌어 있는 부분 확인 가능

# Powerful Git

1.

코딩할 때 (코딩에만 한정되어 있지 않음)  
단순히 ctrl+z를 눌러 이전상태로 되돌리는 것이 아니라,  
**원하는 시점마다 깃발을 꽂고**(버전을 만들고)  
이들 간에 **자유롭게 돌아다닐 수** 있다.

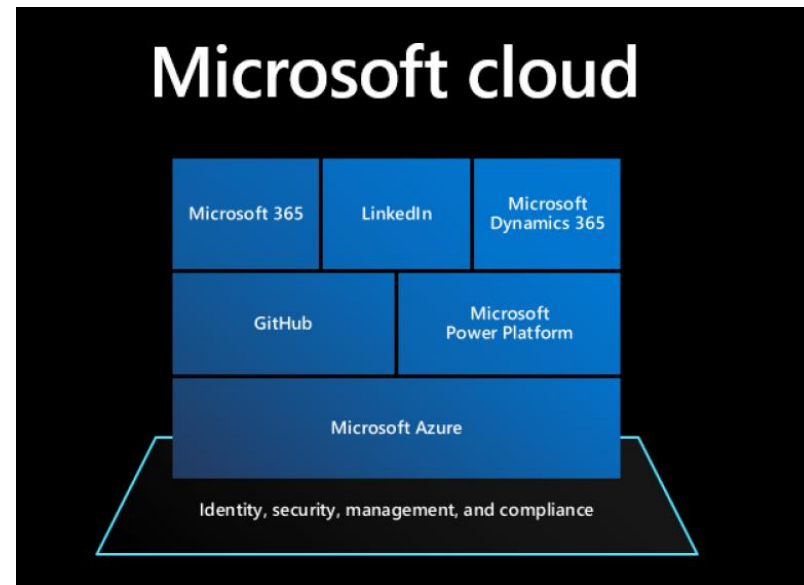
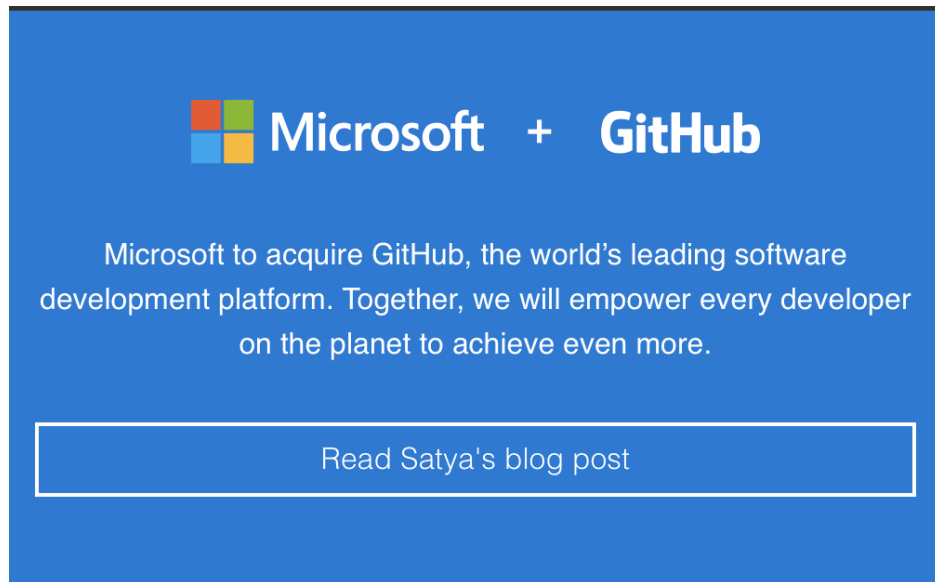
2.

내가 만든 버전 뿐만 아니라  
**동료가 만든 버전으로 이동할 수** 있고,  
동료와 내 버전을 **비교**해서  
**최신본으로 코드를 업데이트** 할 수 있다.

# Git

➤ Git(버전이 관리되는 작업공간)은 어디에나 저장 가능함

- USB
- PC
- 특정 서버
- 클라우드
- 웹 (**GitHub**, GitLab, BitButcket)



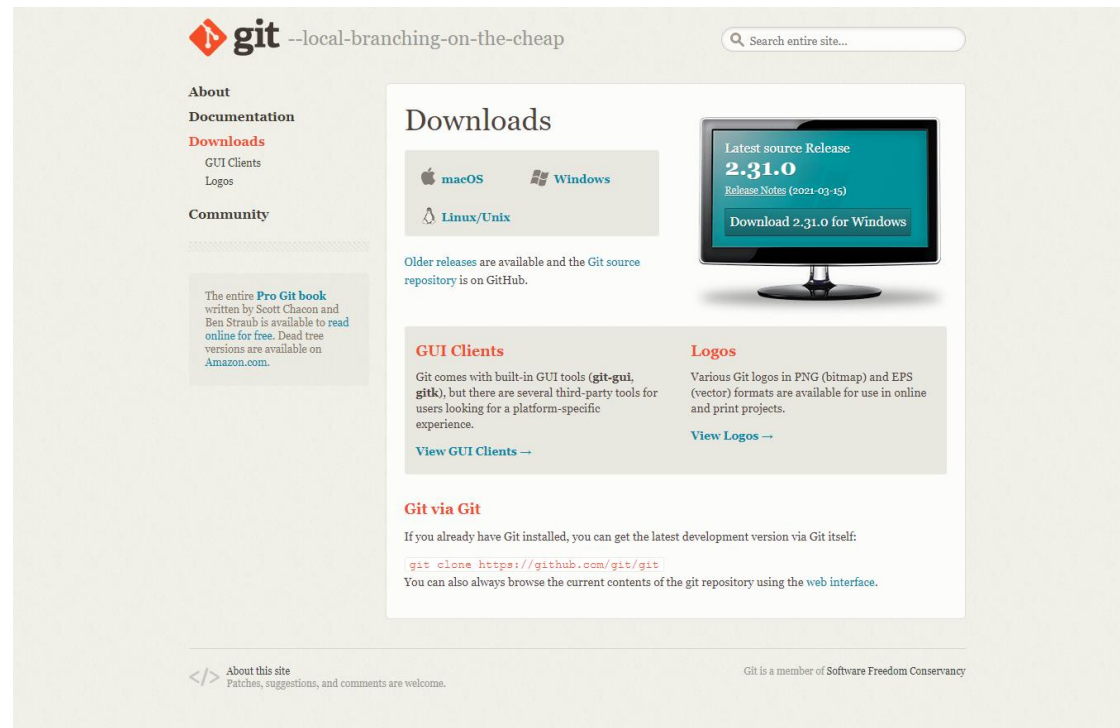


# 로컬에서 Git으로 관리하기

---

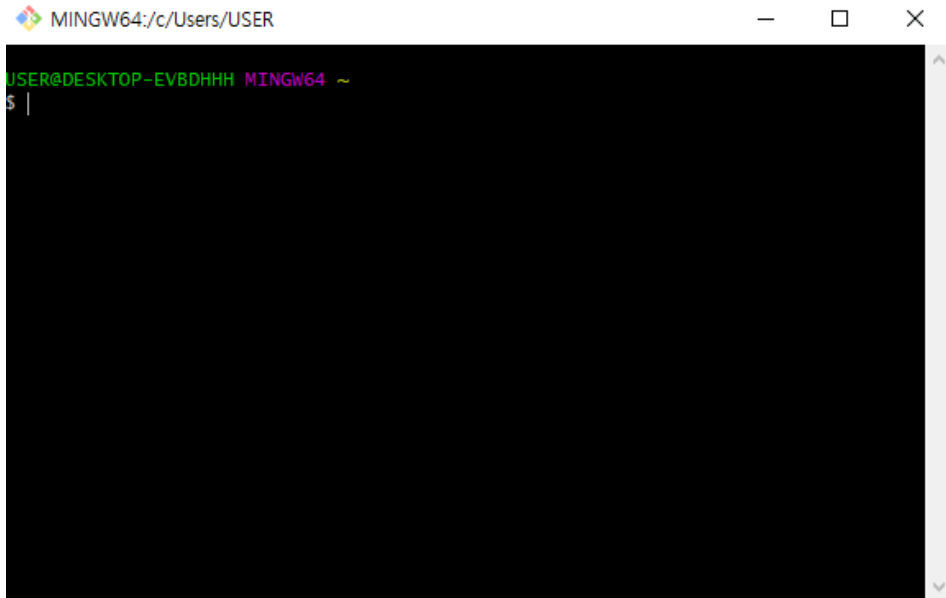
# Install

1. Git 다운로드 (윈도우 버전) <https://git-scm.com/downloads>
2. Git 설치하기
  - 설정 바꾸지 않고 Next, Install 진행

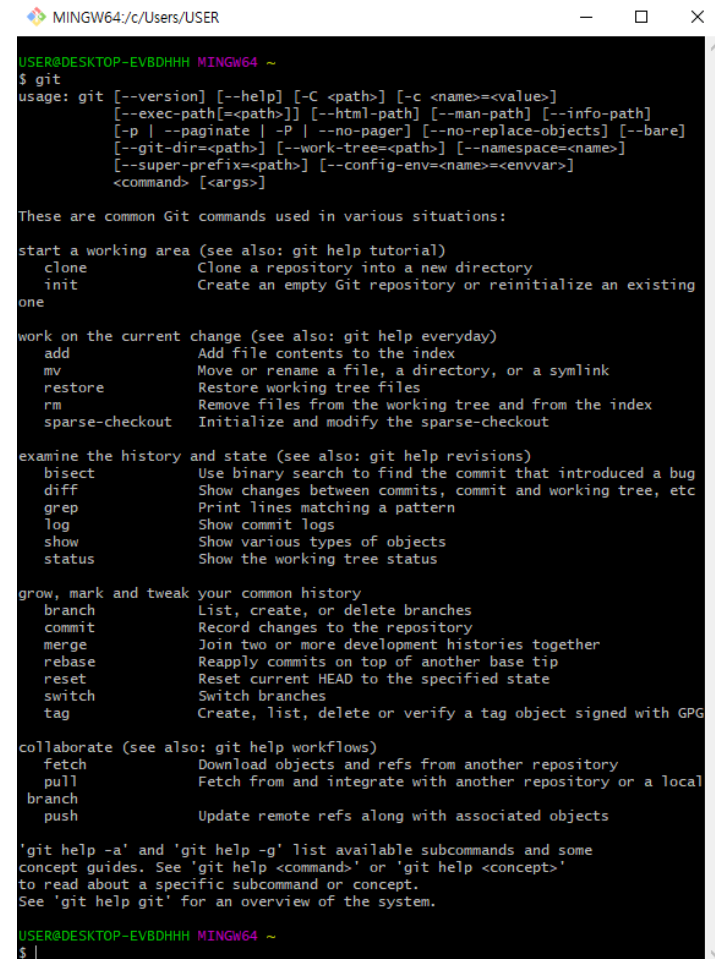


# Install

1. 윈도우 시작 버튼 옆 돋보기 버튼
2. "Git Bash" 입력
3. \$ git



A screenshot of a Windows Start menu search interface. The search bar at the top contains the text "Git Bash". Below the search bar, a list of search results is displayed, including "Git Bash" and "Git Bash (x86)". The "Git Bash" result is highlighted.



A screenshot of a terminal window titled "MINGW64/c/Users/USER". The terminal shows the output of the command "git --help". The output lists various Git commands and their descriptions, organized into sections: "start a working area", "work on the current change", "examine the history and state", "grow, mark and tweak your common history", and "collaborate".

```
USER@DESKTOP-EVBDHHH MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing
            one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index
  sparse-checkout  Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

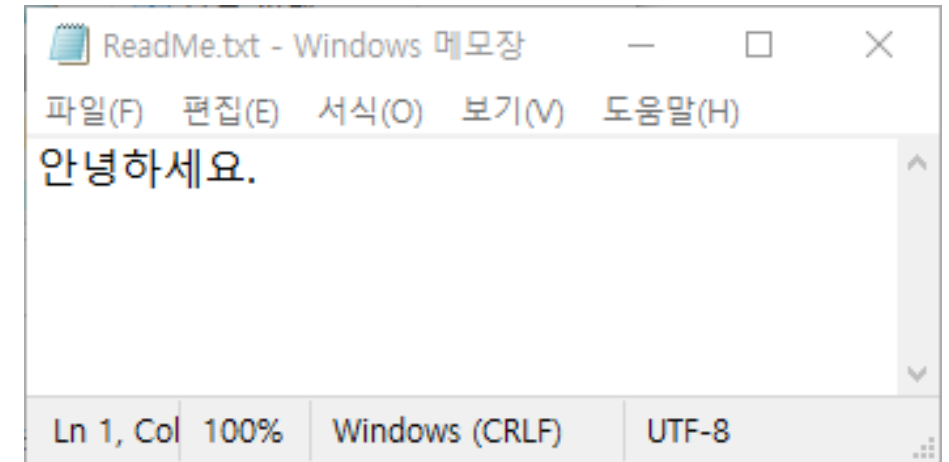
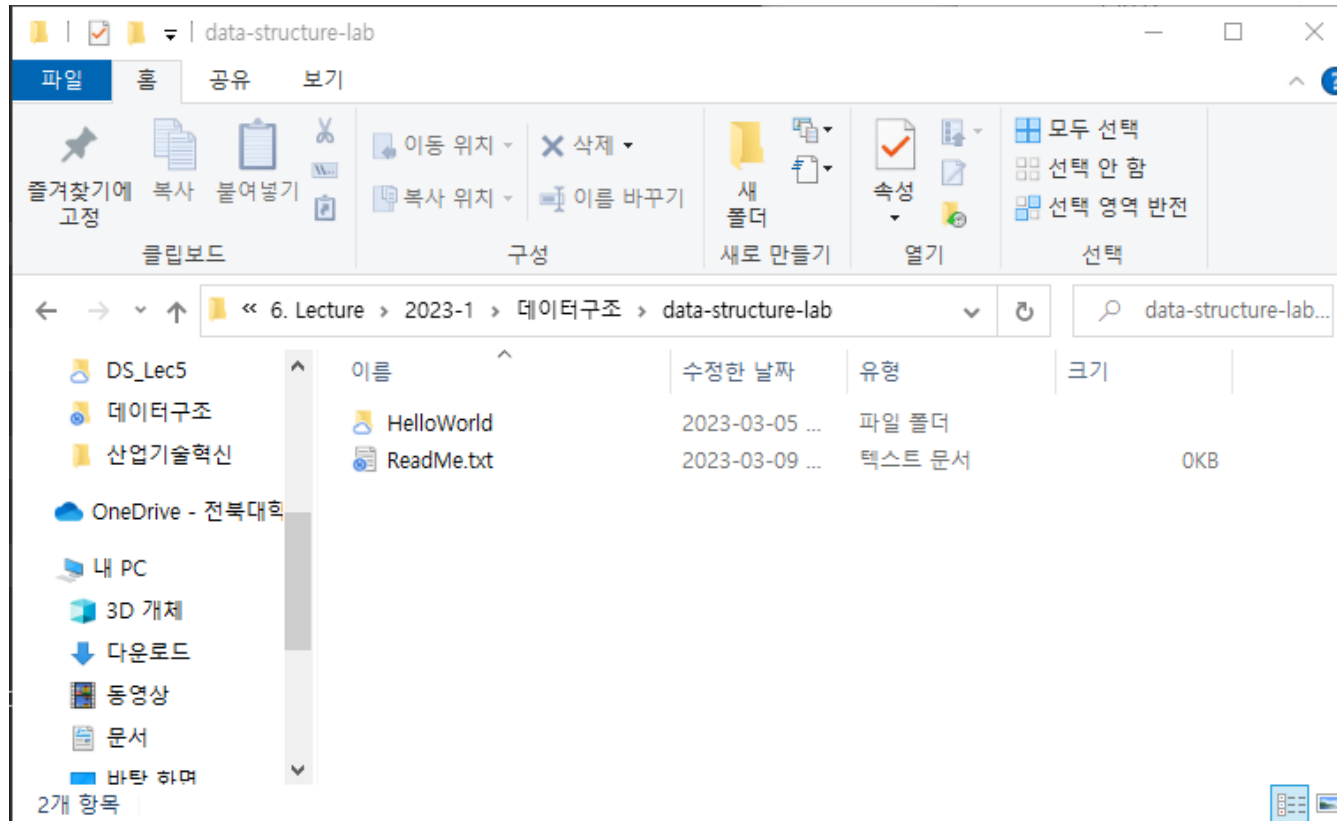
collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local
            branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

USER@DESKTOP-EVBDHHH MINGW64 ~
$
```

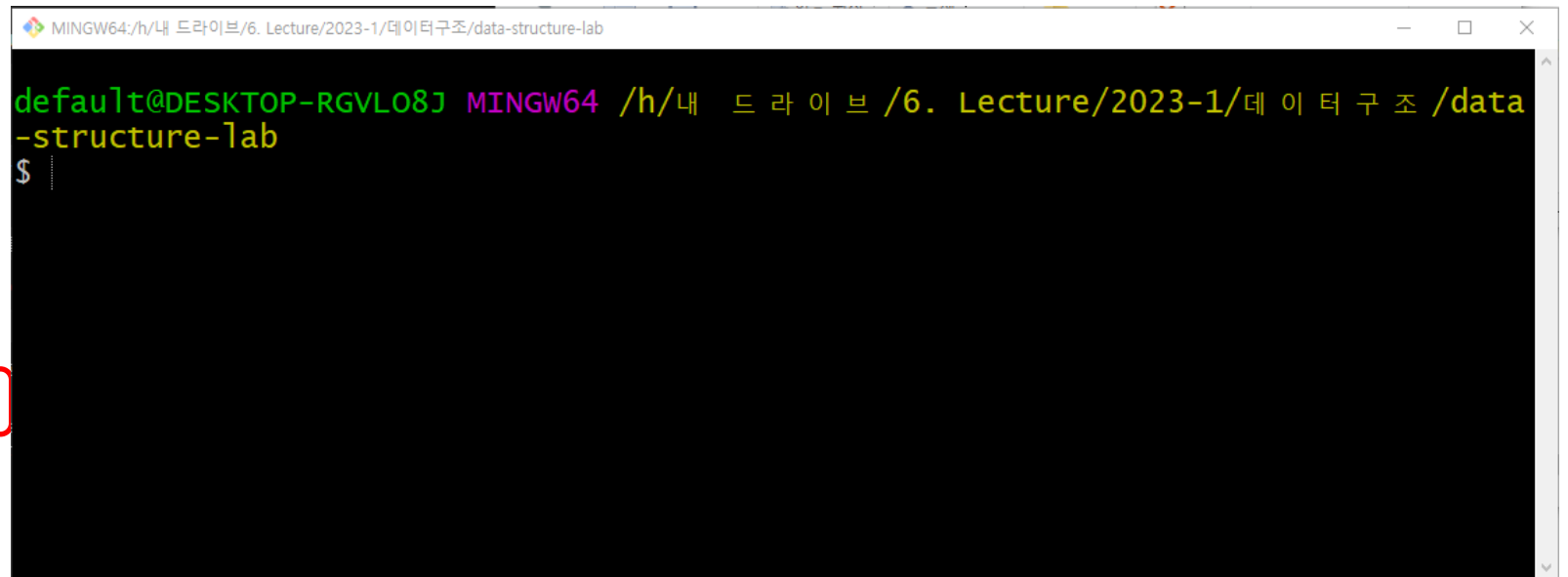
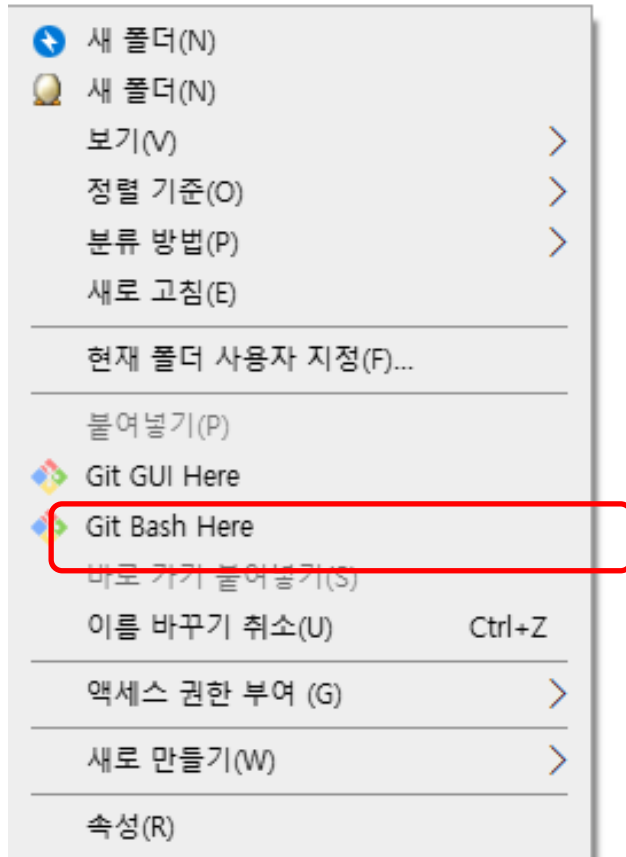
# Sample Project - Init

1. 작업 폴더 생성 (data-structure-lab)
2. 메모장으로 ReadMe.txt 파일 생성 (아무내용이나)



## Sample Project - Init

3. 작업 폴더 안에서 오른쪽 클릭
4. 해당 폴더 경로의 터미널이 열림



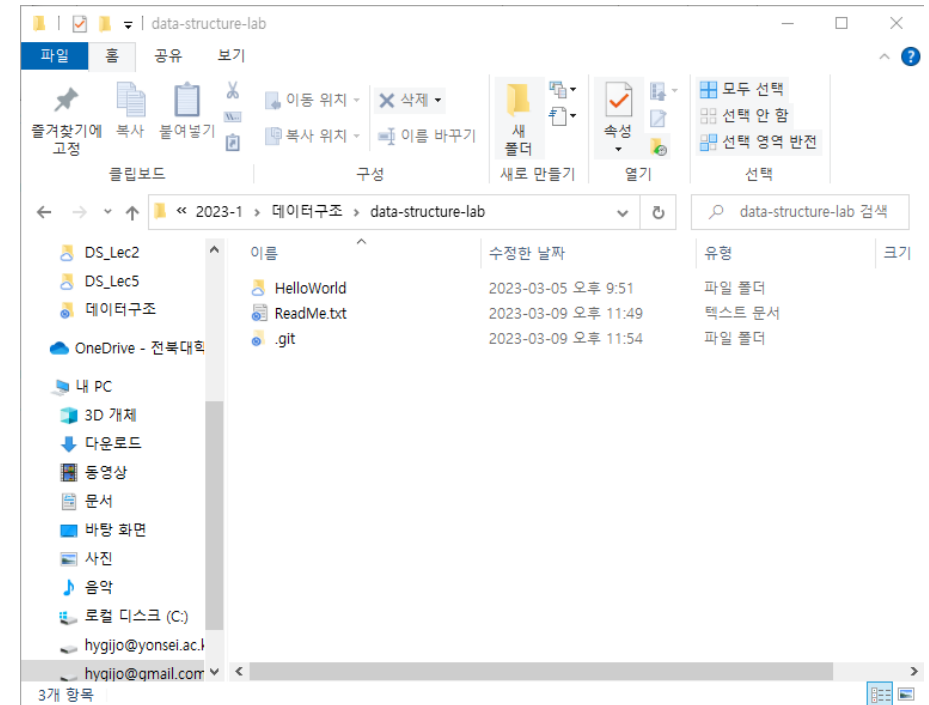
# Sample Project - Init

## 5. Git 초기화

\$ git init

Git Repository 생성됨 (.git 폴더가 자동 생성)

```
MINGW64: h:\내 드라이브\6. Lecture\2023-1\데이터구조\data-structure-lab
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브/6. Lecture/2023-1/데이터구조/data-structure-lab
$ git init
Initialized empty Git repository in H:/내 드라이브/6. Lecture/2023-1/데이터구조/data-structure-lab/.git/
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브/6. Lecture/2023-1/데이터구조/data-structure-lab (master)
$
```



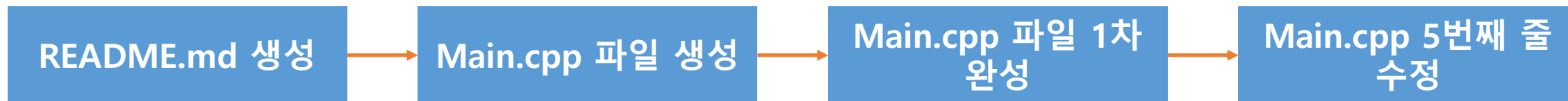
# Commit 커밋

## ➤ 커밋이란?

- 하나의 변경 사항 단위
- Ex) xxx\_yyy.cpp 5번째 줄 수정
- Ex) README.md 파일 생성

## ➤ 꾸준히 쌓이는 커밋

- 어떤 의미있는 변동사항이 있을 때마다 커밋을 하면 된다.
- `$ git commit -m "어떤 변동사항이 있었는지 메시지 간단하게"`



# Sample Project

## 6. 커밋 Commit

- ReadMe.txt 파일을 하나의 버전으로 저장
- RPG 게임에서 중간 저장이라고 생각

### 6-1. 계정 등록

- Email address, Username

```
$ git config --global user.email "your_email@gmail.com"
```

```
$ git config --global user.name "username"
```

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab (master)
$ git config --global user.email "hygijo@gmail.com"

default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab (master)
$ git config --global user.name "johg15"
```



# Sample Project

## 6-2. 커밋할 파일 선택

- 특정 파일 선택 또는 전체를 지정할 수 있음

\$ git add ReadMe.txt → ReadMe.txt만 커밋

(또는) \$ git add . → 전체 커밋

## 6-3. 커밋하기

- 어떤 커밋인지 설명 메시지 간략하게 넣어서 커밋하기

\$ git commit -m "readme 파일 생성"

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab (master)
$ git add ReadMe.txt

default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab (master)
$ git commit -m "readme 파일 생성"
[master (root-commit) b9b804d] readme 파일 생성
1 file changed, 1 insertion(+)
create mode 100644 ReadMe.txt
```





# Sample Project

## 6-4. 두 번째 커밋하기

- 새로운 파일 추가하고 커밋해보기

```
$ git add Manual.docx
```

```
$ git commit -m "second file added"
```

이름	수정한 날짜	유형
 HelloWorld	2023-03-05 ...	파일 폴더
 ReadMe.txt	2023-03-09 ...	텍스트 문서
 .git	2023-03-09 ...	파일 폴더
 Manual.docx	2023-03-10 ...	Microsoft Word ...

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab (master)
$ git add Manual.docx

default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab (master)
$ git commit -m "second file added"
[master e4f208e] second file added
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Manual.docx
```

# Sample Project

## 6-5. 시간 여행

- 지금까지 커밋 확인
- 첫 번째 커밋 ID 앞 7자리 확인

\$ git log

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab (master)
$ git log
commit e4f208e098e87bc343da414aa700b2e09b544827 (HEAD -> master)
Author: johg15 <hygijo@gmail.com>
Date:   Fri Mar 10 00:15:50 2023 +0900

    second file added

commit b9b804da7977925b62cedfebbe81288d61d45515
Author: johg15 <hygijo@gmail.com>
Date:   Fri Mar 10 00:03:13 2023 +0900




    readme 파일 생성
```

# Sample Project

## 6-6. 시간 여행

- 첫 번째 커밋으로 돌아가기

\$ git checkout 003de720

이름	수정한 날짜	유형
 HelloWorld	2023-03-05 ...	파일 폴더
 ReadMe.txt	2023-03-09 ...	텍스트 문서
 .git	2023-03-09 ...	파일 폴더

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab (master)
$ git checkout b9b804d
Note: switching to 'b9b804d'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-c` with the switch command. Example:

```
git switch -c <new-branch-name>
```

or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable `advice.detachedHead` to false

HEAD is now at b9b804d readme 파일 생성

# Sample Project

## 6-6. 시간 여행

- 다시 master로 되돌아오기

\$ git checkout master

이름	수정한 날짜	유형
.git	2023-03-10 ...	파일 폴더
HelloWorld	2023-03-05 ...	파일 폴더
Manual.docx	2023-03-10 ...	Microsoft Word ...
ReadMe.txt	2023-03-09 ...	텍스트 문서

```
default@DESKTOP-RGVLO81 MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab ((b9b804d...))
$ git checkout master
Previous HEAD position was b9b804d readme 파일 생성
Switched to branch 'master'
```

```
default@DESKTOP-RGVLO81 MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab (master)
$
```

# GitHub 원격저장소

---

# GitHub

➤ Git(버전이 관리되는 작업공간)은 어디에나 저장 가능함

- USB
- PC
- 특정 서버
- 클라우드
- 웹 (**GitHub**, GitLab, BitButcket)

Google a\* algorithm python github

동영상 이미지 쇼핑 도서 뉴스 지도 금융

검색결과 약 127,000,000개 (0.35초)

**GitHub**  
https://github.com > TheAlgorithms > Python  
**TheAlgorithms/Python: All Algorithms implemented in Python**  
All algorithms implemented in Python - for education. Implementations are for learning purposes only. They may be less efficient than the implementations in the ...  
[Contributing guidelines](#) · [DIRECTORY.md](#) · [Issues 25](#) · [Pull requests 209](#)

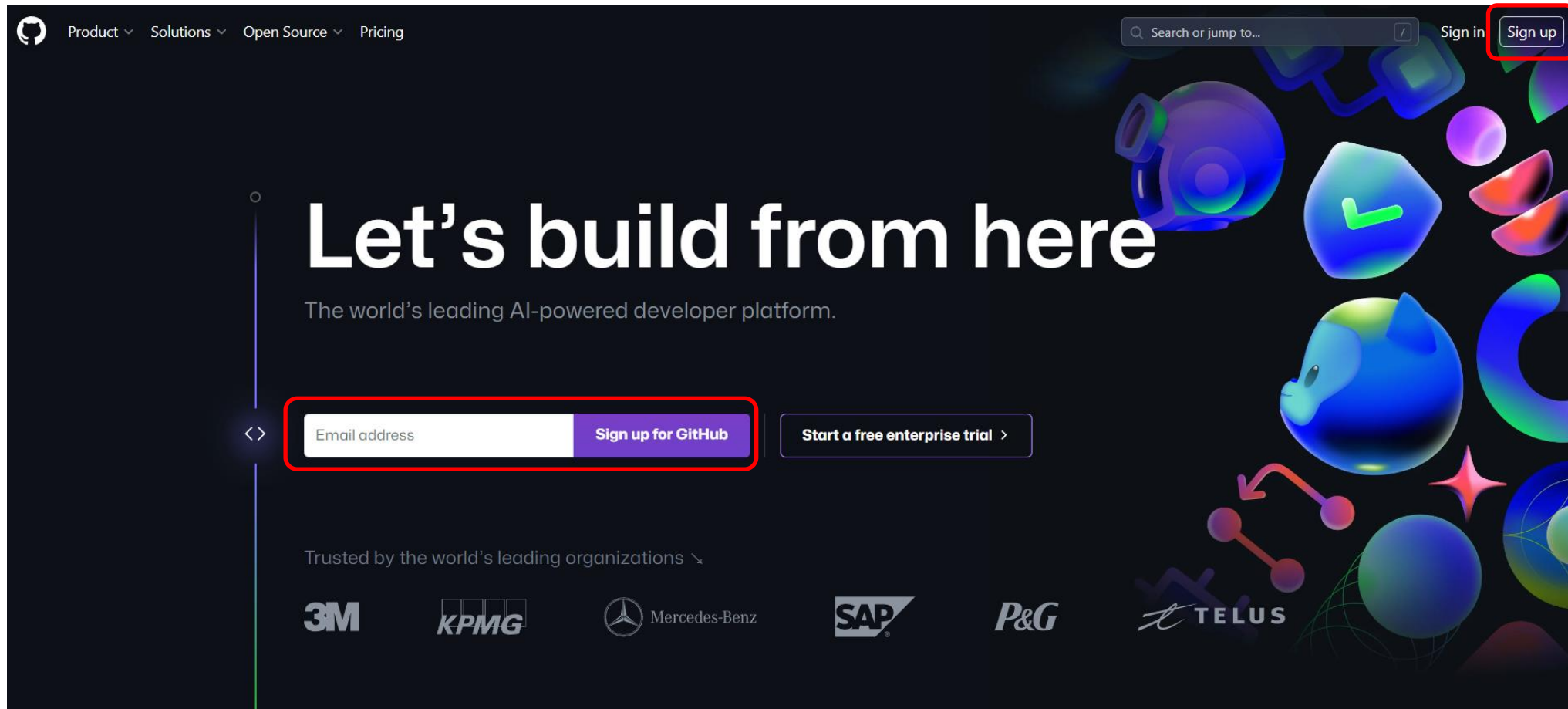
**GitHub**  
https://gist.github.com > jamlees2  
**A\* Algorithm implementation in python.**  
The aim was to update the G value of node in openset, in case the node was already present in the openset. For this update process, we should get to point the ...

**GitHub**  
https://github.com > Chrisbelefantis > A-Star-Algorithm  
**Implementation of the A Star algorithm using Python**  
In this project I use tKinter package in order to create an implementation of the A Star path search algorithm. - Chrisbelefantis/A-Star-Algorithm.

**GitHub**  
https://github.com > topics > astar-search-algorithm > l=p...  
**astar-search-algorithm**  
A python script that implements a generic planner to solve a series of minigames using heuristic algorithms to generate the best possible moves to reach the ...

# Sign up (GitHub)

1. GitHub 사이트 가입하기 <https://github.com>

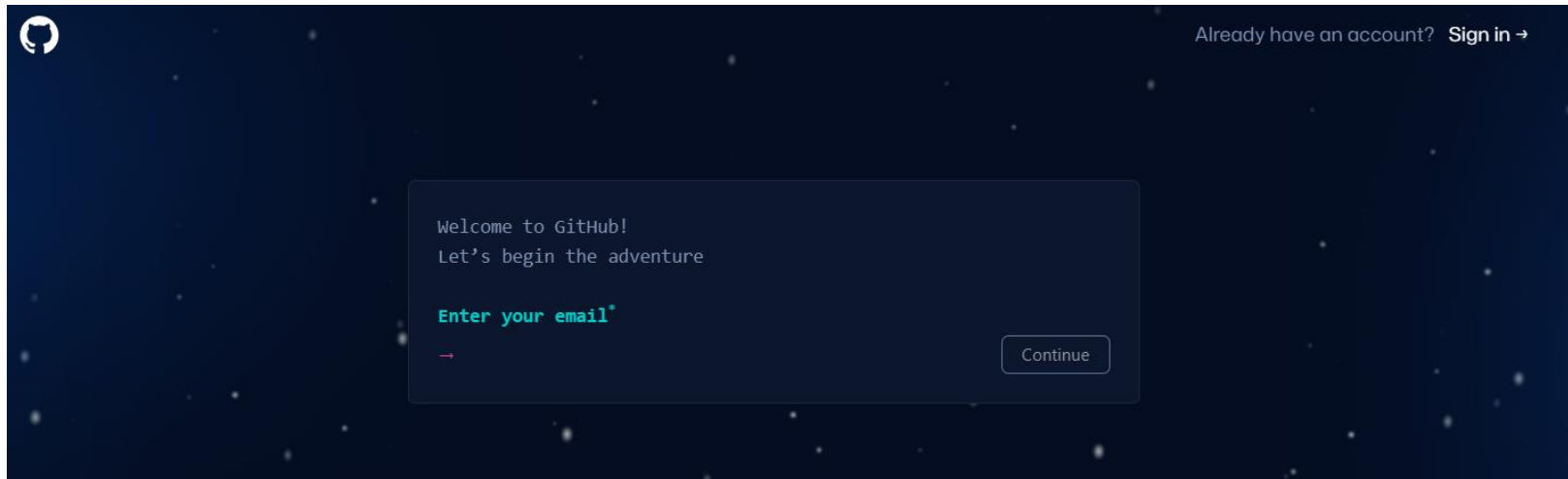




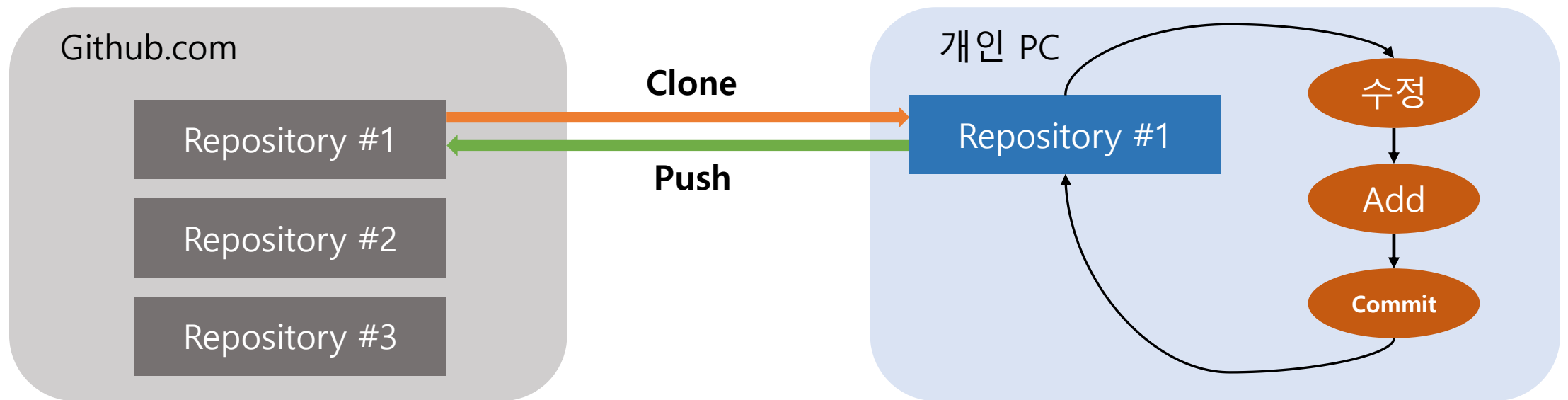
# Sign up (GitHub)

## 2. Account 생성

- **Email address**
- **Username** – 본명(영문)으로
- Email Verification 완료하기

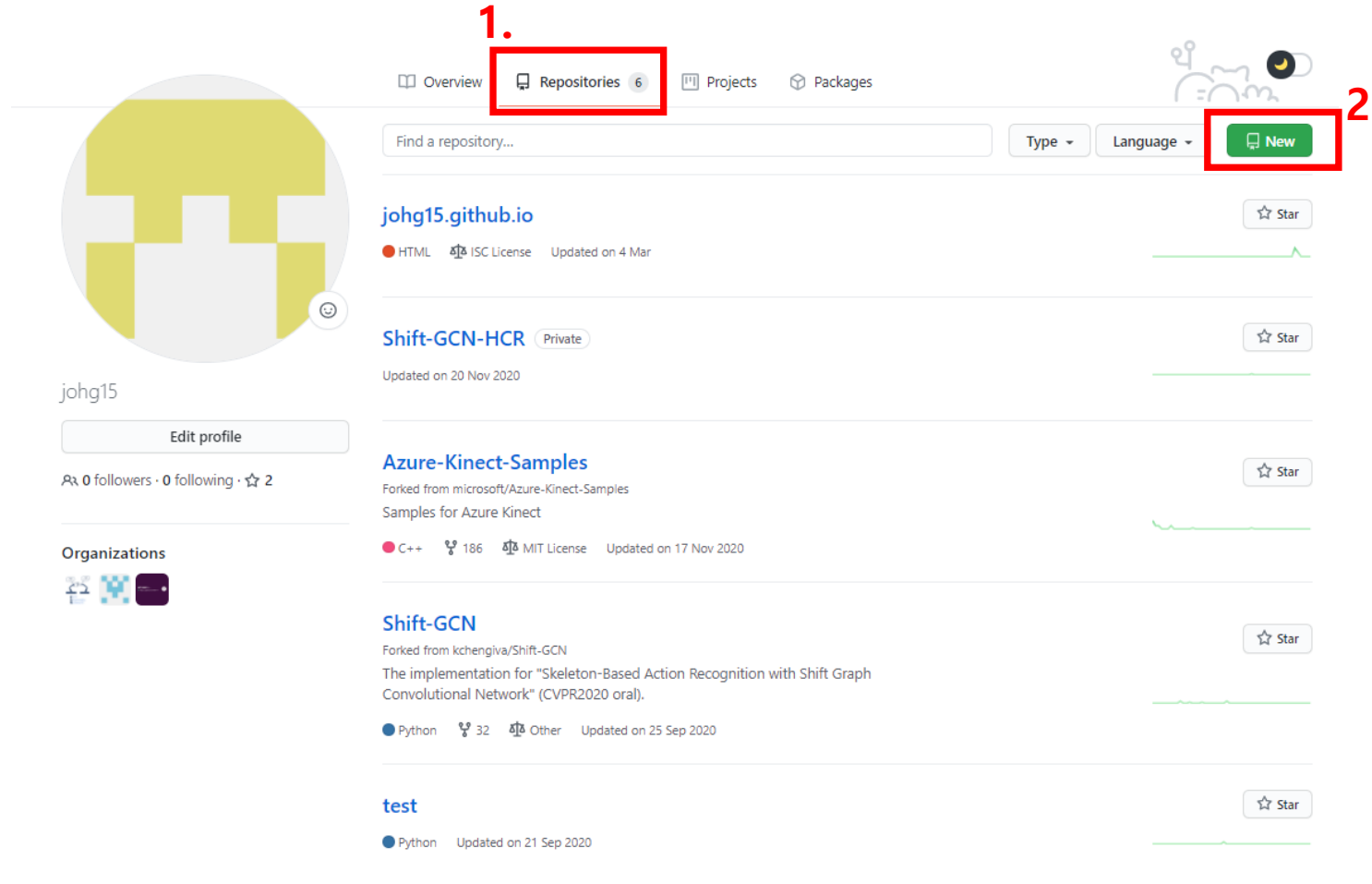


# Git, GitHub



# GitHub 원격저장소

## ➤ Repository 생성



The screenshot shows the GitHub profile of user 'johg15'. The 'Repositories' tab is selected and highlighted with a red box and the number '1.'. In the top right corner, the 'New' button is highlighted with a red box and the number '2.'. The profile includes a yellow and grey avatar, the username 'johg15', an 'Edit profile' button, and statistics: 0 followers, 0 following, and 2 stars. Below the profile are 'Organizations' and a list of repositories:

- HTML**: ISC License, Updated on 4 Mar
- Shift-GCN-HCR**: Private, Updated on 20 Nov 2020
- Azure-Kinect-Samples**: Forked from microsoft/Azure-Kinect-Samples, Samples for Azure Kinect, C++ 186, MIT License, Updated on 17 Nov 2020
- Shift-GCN**: Forked from kchengiva/Shift-GCN, The implementation for "Skeleton-Based Action Recognition with Shift Graph Convolutional Network" (CVPR2020 oral), Python 32, Other, Updated on 25 Sep 2020
- test**: Python, Updated on 21 Sep 2020

# GitHub 원격저장소

## ➤ Repository 이름

- data-structure-lab

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

#### Repository template

Start your repository with a template repository's contents.

No template ▾

Owner \*

johg15 ▾

Repository name

data-structure-lab ✓

Great repository names are short and memorable. Need inspiration? How about [verbose-rotary-phone?](#)

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

#### Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

#### Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

#### Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾


You are creating a public repository in your personal account.

Create repository

# GitHub 원격저장소

## ➤ Quick Setup

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


 Set up in Desktop

 or 

HTTPS

SSH

https://github.com/johg15/data-structure-lab.git




Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

1.

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


```
echo "# data-structure-lab" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/johg15/data-structure-lab.git
git push -u origin main
```



2.

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

```
git remote add origin https://github.com/johg15/data-structure-lab.git
git branch -M main
git push -u origin main
```



...or import code from another repository

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

Import code

# GitHub 원격저장소

## ➤ Quick Setup

- 로컬과 원격저장소 연결
- \$ git remote add origin https://github.com/(your\_username)/(Repository\_name).git

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab ((b9b804d...))  
$ git remote add origin https://github.com/johg15/data-structure-lab.git
```

- 잘 연결되었는지 확인
- \$ git remote -v

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab ((b9b804d...))  
$ git remote -v  
origin https://github.com/johg15/data-structure-lab.git (fetch)  
origin https://github.com/johg15/data-structure-lab.git (push)
```

# GitHub 원격저장소

## ➤Quick Setup

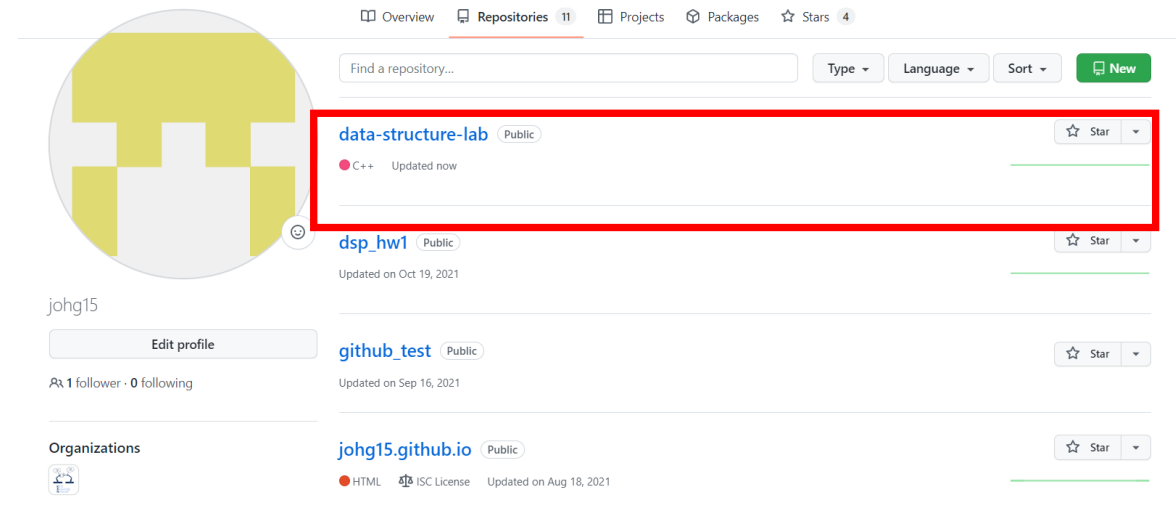
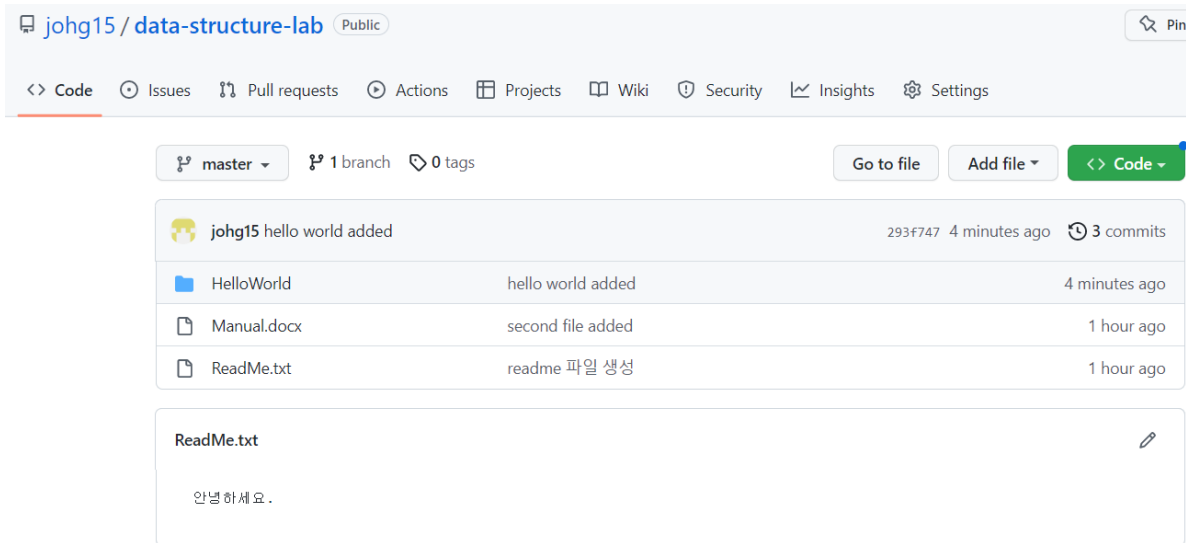
- 로컬 저장소의 커밋을 push 명령어로 원격저장소에 올리기
- \$ git push -u origin master

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/
데 이 터 구조 /data-structure-lab (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
writing objects: 100% (2/2), 260 bytes | 65.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/johg15/data-structure-lab.git
   293f747..5c072d5  master -> master
```

# GitHub 원격저장소

## ➤ Quick Setup

- 새로그침해서 GitHub에서 잘 올라왔는지 확인

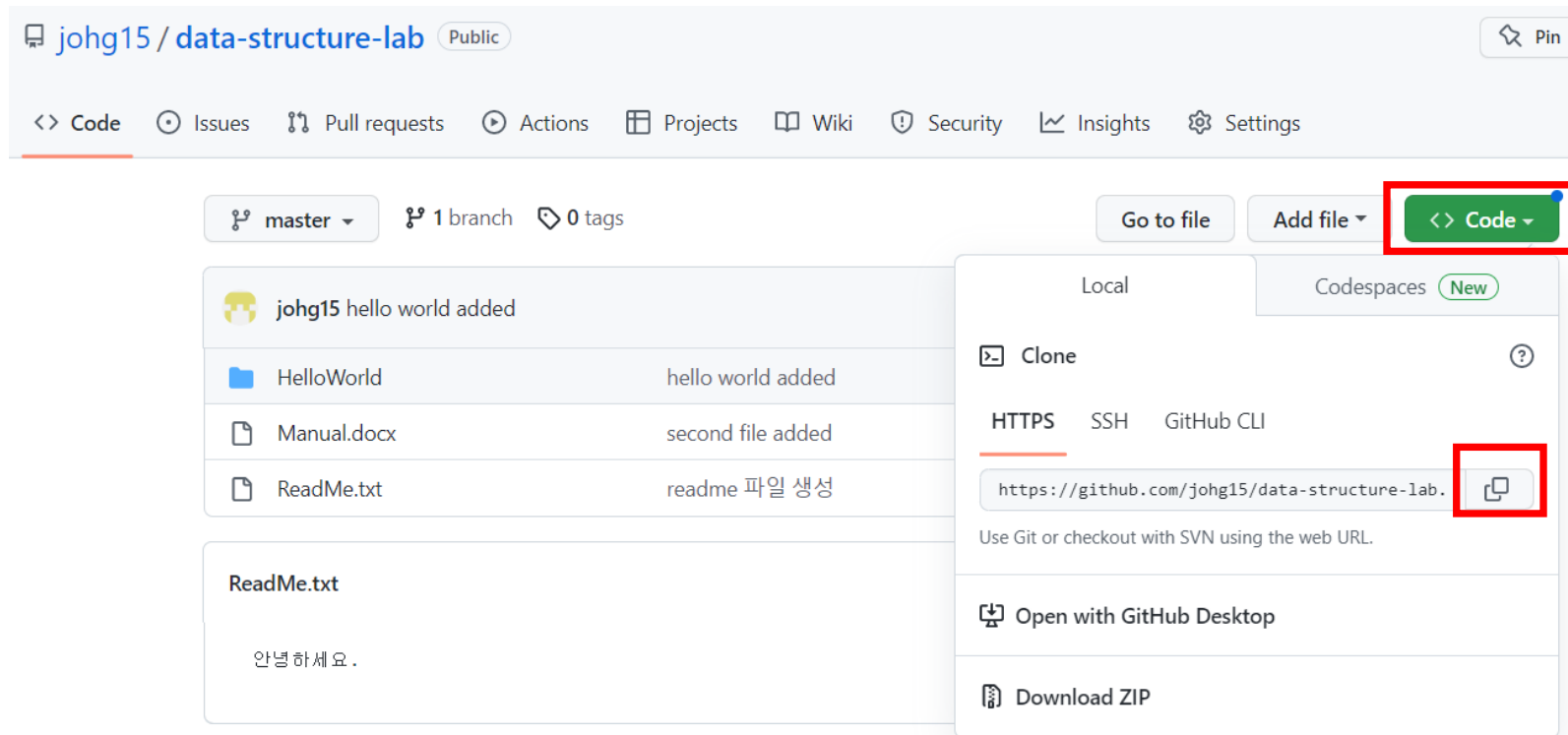




# GitHub 내려받기

## ➤ 원격저장소에서 내려받기 (Clone)

- 다른 새로운 폴더 생성 (data-structure-lab2) (or 본인의 컴퓨터에서) → Git Bash Here



# GitHub 내려받기

## ➤ 원격저장소에서 내려받기 (Clone)

- \$git clone https://github.com/johg15/data-structure-lab.git .
- 내려받기 확인

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab2
$ git clone https://github.com/johg15/data-structure-lab.git .
Cloning into '.'...
remote: Enumerating objects: 53, done.
remote: Counting objects: 100% (53/53), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 53 (delta 9), reused 53 (delta 9), pack-reused 0
Receiving objects: 100% (53/53), 1003.34 KiB | 7.66 MiB/s, done.
Resolving deltas: 100% (9/9), done.

default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab2 (master)
$
```

# GitHub 올리기

## ➤ 로컬저장소에서 수정 후 올리기

- \$git add .
- \$git commit -m "hi"
- \$git push origin master

```
MINGW64/h/내 드라이브/6. Lecture/2023-1/데이터구조/data-structure-lab

default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브/6. Lecture/2023-1/데이터구조/data-structure-lab (master)
$ git add .

default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브/6. Lecture/2023-1/데이터구조/data-structure-lab (master)
$ git commit -m "test"
[master d49341a] test
1 file changed, 2 insertions(+), 1 deletion(-)

default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브/6. Lecture/2023-1/데이터구조/data-structure-lab (master)
$ git push -u origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 291 bytes | 72.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/johg15/data-structure-lab.git
  5c072d5..d49341a master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

# GitHub 올리기

## ➤로컬저장소에서 수정 후 올리기

johg15 / data-structure-lab Public

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

master 1 branch 0 tags Go to file Add file <> Code

johg15 test	d49341a 9 minutes ago	🕒 5 commits
📁 HelloWorld	hello world added	22 minutes ago
📄 ReadMe.txt	test	9 minutes ago

ReadMe.txt

안녕하세요.  
**Test**

# GitHub 다시 내려받기

- 수정된 사항 다시 내려받기
  - 다른 작업 폴더에서 다시 내려받기
  - \$ git pull origin master

```
default@DESKTOP-RGVLO8J MINGW64 /h/내 드라이브 /6. Lecture/2023-1/데이터 구조 /data-structure-lab2 (master)
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 276 bytes | 5.00 KiB/s, done.
From https://github.com/johg15/data-structure-lab
* branch                master      -> FETCH_HEAD
   d49341a..12a860a      master      -> origin/master
Updating d49341a..12a860a
Fast-forward
 README.txt | 3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)
```

---

## Summary

1. (init한 후) 코드를 수정/추가 했을 때 업뎃 방법

```
$ git add .
```

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

```
$ git push origin master
```

3줄만 입력하면 됨

2. Github에 있는 것들을 local 컴퓨터에 가져오고 싶을 때

```
$ git clone "주소" . (→처음 복사)
```

```
$ git pull origin master
```

---

## Example

1. Chap01-1 Project 생성