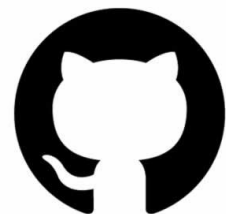




# GIT, GITHUB 사용법



git



GitHub

20183159 최보열

init, clone, remote, add, commit, push, pull, branch, merge,  
pull request, contributor, repository...

2022.01.21

# 1. 로컬 저장소, 원격저장소



로컬 저장소와 원격 저장소의 개념, git  
과 github 설명 및 사용하는 이유





## Git, Github



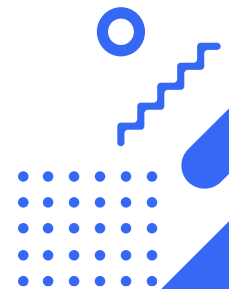
### Git

컴퓨터 파일의 **변경사항**을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 **조율**하기 위한 분산 **버전 관리 시스템**.  
(VCS: Version Control System)



### GitHub

분산 버전 관리 툴인 깃 **저장소 호스팅**을 지원하는 웹 서비스이다.  
(구글 드라이브를 떠올리면 좋다)





## Repository(저장소)

버전 관리 시스템에서, 저장소 또는 리포지터리(repository)는 파일이나 디렉터리 구조의 집합을 위해 **메타데이터를 저장하는 자료 구조**이다.

사용 중인 버전 관리 시스템이 **분산 방식**(예: Git 또는 머큐리얼)인지, 아니면 중앙 집중 방식(예: (서브버전 또는 퍼포스)인지에 따라 저장소 내 정보 전반이 모든 사용자 시스템에 복제되거나 단일 서버 상에서 관리될 수 있다. 저장소에 들어가는 **메타데이터**에는 다음과 같은 것들이 포함된다:

- 저장소의 역사적 변경 기록.
- 커밋 객체의 집합.
- 헤드(head)라는 이름의 커밋 객체의 참조 집합.





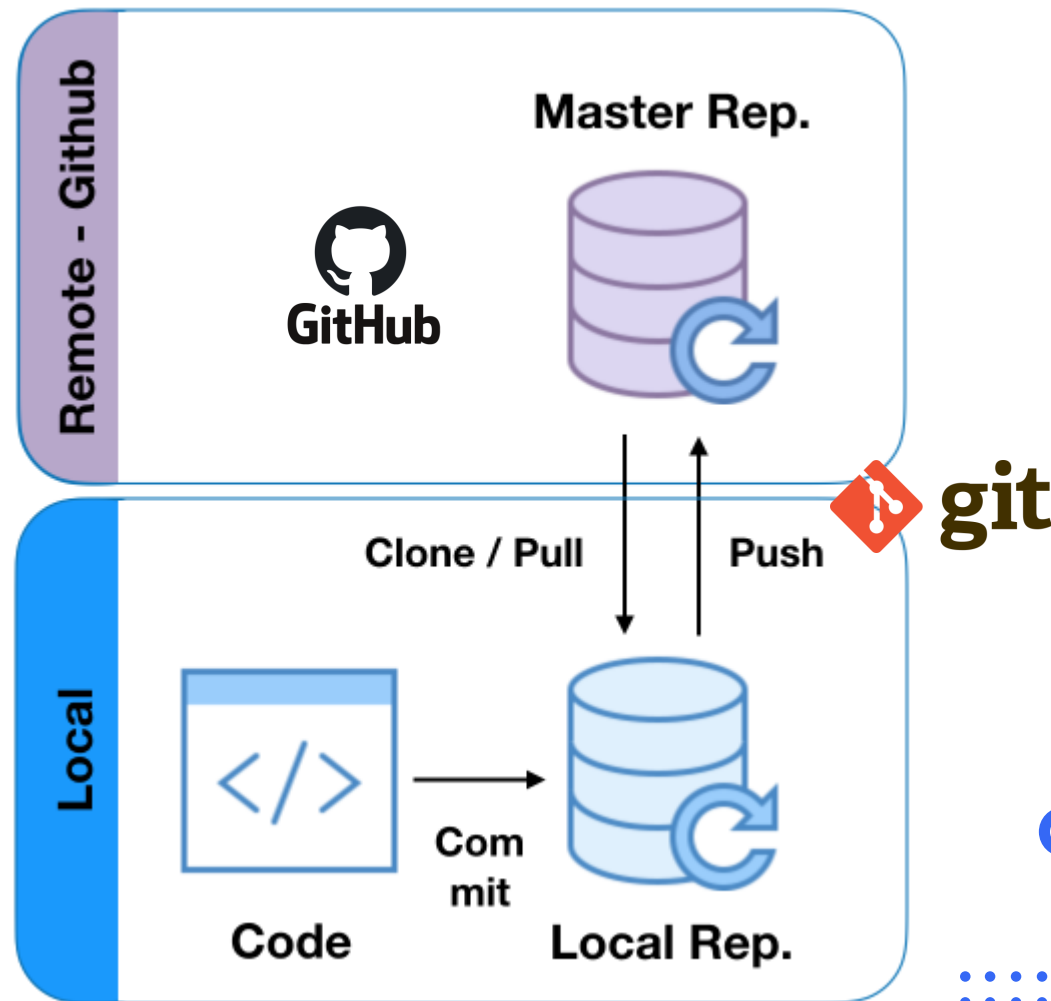
## 로컬 저장소, 원격저장소

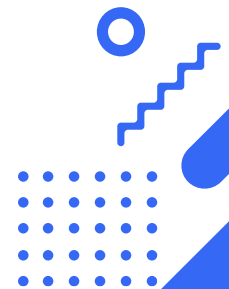
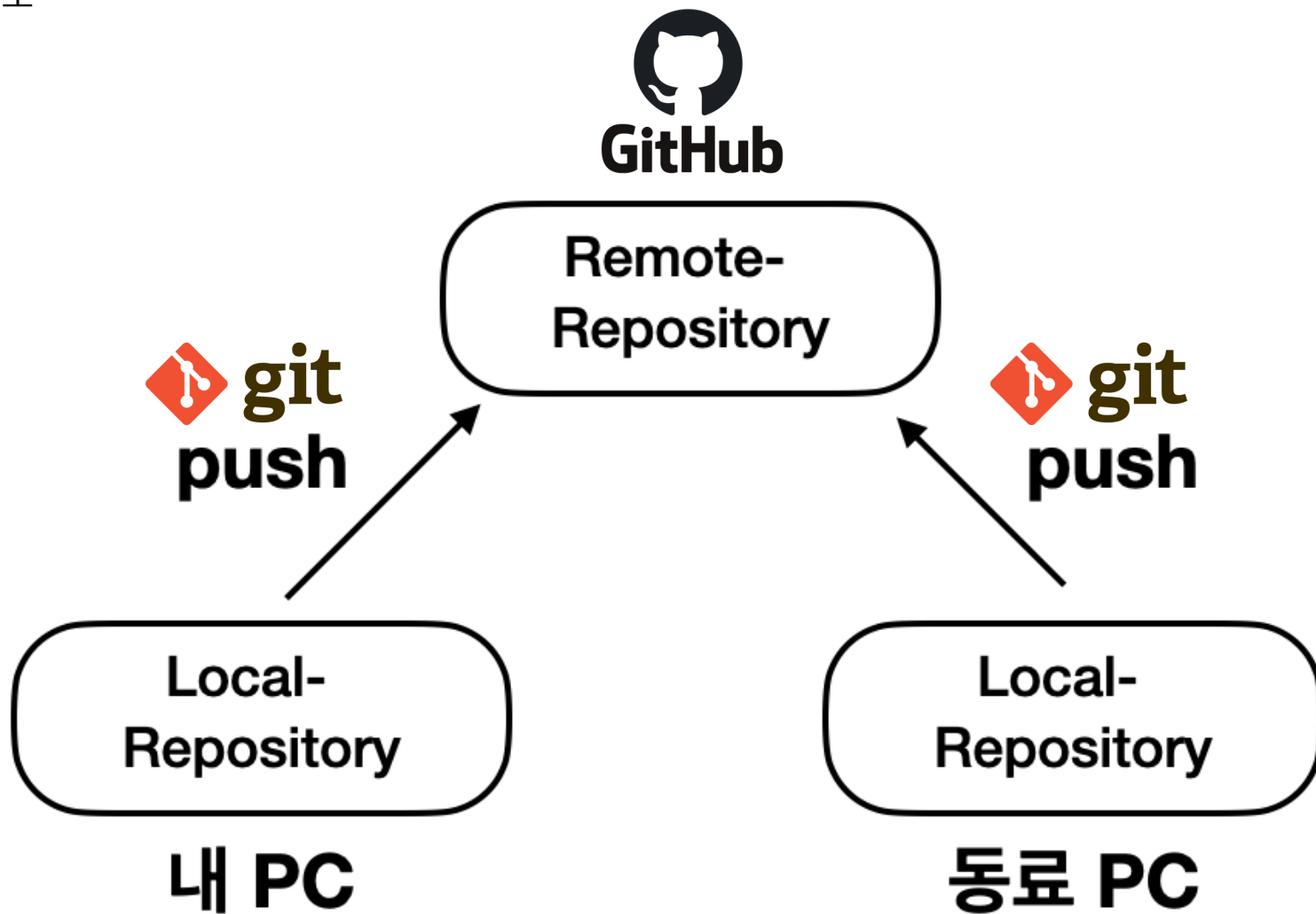
### 로컬 저장소 (Local Repository)

현재 내가 사용하고 있는 내 디바이스(PC)에 저장되는 저장소이다. 내 PC의 폴더 라고 생각하면 된다. 원격저장소의 파일, 혹은 폴더를 로컬 저장소로 Pull 할 수 있다.

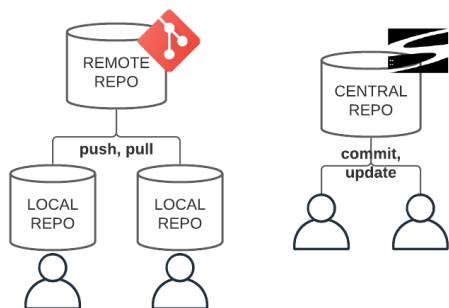
### 원격 저장소 (Remote Repository)

Git의 주 목적은 파일 공유에 있다. 원격 저장소는 원격 서버에 저장되고 관리되는 저장소이며, 로컬 저장소의 파일, 혹은 폴더를 원격 저장소에 Push 할 수 있다.





## 용어설명



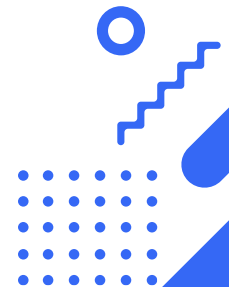
### 분산 버전 관리 시스템

모든 cilient가 저장소가 될 수 있는 버전관리 시스템. 중앙집중형 버전관리는 최신 코드만 받아오는 반면, 분산 버전 관리 시스템은 server에서 저장소를 통채로 받아온다.



### 메타 데이터

메타데이터(metadata)는 데이터(data)에 대한 데이터이다. 이렇게 흔히들 간단히 정의하지만 엄격하게는, 캐런 코일(Karen Coyle)에 의하면 '어떤 목적을 가지고 만들어진 데이터(Constructed data with a purpose)'라고도 정의한다.

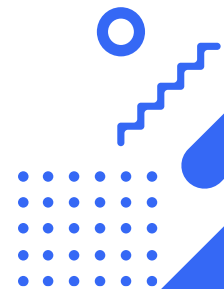
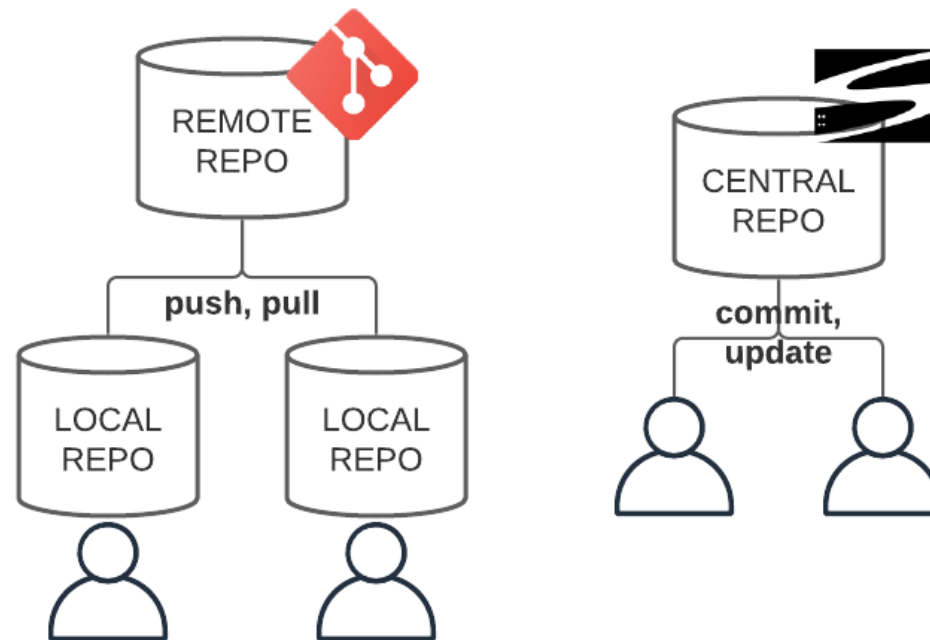




## 분산 버전 관리 시스템 Decentralized Version Control Systems(DVCS)

모든 cilient가 저장소가 될 수 있는 버전관리 시스템. 중앙 집중형 버전관리는 최신 코드만 받아오는 반면, 분산 버전 관리 시스템은 server에서 저장소를 통채로 받아 온다.

분산 버전 관리 시스템은 serve에서 .git도 받아와서 해당 로컬에서 git log를 포함해서 모든 변경내역들을 가져오고 control할 수 있게 된다.





# git이 관리하고 있는 경우 숨김파일로 .git이 존재한다.

```
gju06051@LAPTOP-ER5F0VF4:/mnt/c/Users/gju06/Desktop/AIDC$ ls -al
total 0
drwxrwxrwx 1 gju06051 gju06051 512 Jan 21 19:27 .
drwxrwxrwx 1 gju06051 gju06051 512 Jan 22 14:37 ..
drwxrwxrwx 1 gju06051 gju06051 512 Jan 21 19:28 .git
-rwxrwxrwx 1 gju06051 gju06051 252 Jan 20 22:02 common.sh
drwxrwxrwx 1 gju06051 gju06051 512 Jan 20 22:02 dale
drwxrwxrwx 1 gju06051 gju06051 512 Jan 20 22:02 rtl
drwxrwxrwx 1 gju06051 gju06051 512 Jan 21 19:29 sim
drwxrwxrwx 1 gju06051 gju06051 512 Jan 20 22:02 syn
drwxrwxrwx 1 gju06051 gju06051 512 Jan 20 22:02 verification
gju06051@LAPTOP-ER5F0VF4:/mnt/c/Users/gju06/Desktop/AIDC$ git log
commit 123d2298fc87f9eebd03a4f4bf3754b0a51aad22 (HEAD -> master, origin/master, origin/HEAD)
Author: Scalable Architecture Lab <ScalableArchiLab@taurus.(none)>
Date:   Fri Jan 21 19:06:31 2022 +0900

    test environment in gju06051 file

commit c5804605d5ae0801848a3b131d02ce35c89ccd60
Merge: 60b400c 498e92d
Author: dale40 <dale40@gmail.com>
Date:   Thu Jan 20 12:47:50 2022 +0900

    Merge pull request #6 from kiheon1116/dale_AXI_interface

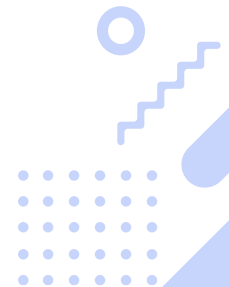
    Changes the design to use interfaces

commit 498e92d40156d87662eb240630f96879bf07df20 (origin/dale_AXI_interface)
Author: Scalable Architecture Lab <ScalableArchiLab@taurus.(none)>
Date:   Thu Jan 20 12:46:22 2022 +0900

    Working

commit 5478880968577a729a7e529e6da8ac42d124c9c5
Author: Scalable Architecture Lab <ScalableArchiLab@taurus.(none)>
Date:   Thu Jan 20 12:35:32 2022 +0900

    Changed interfaces
```





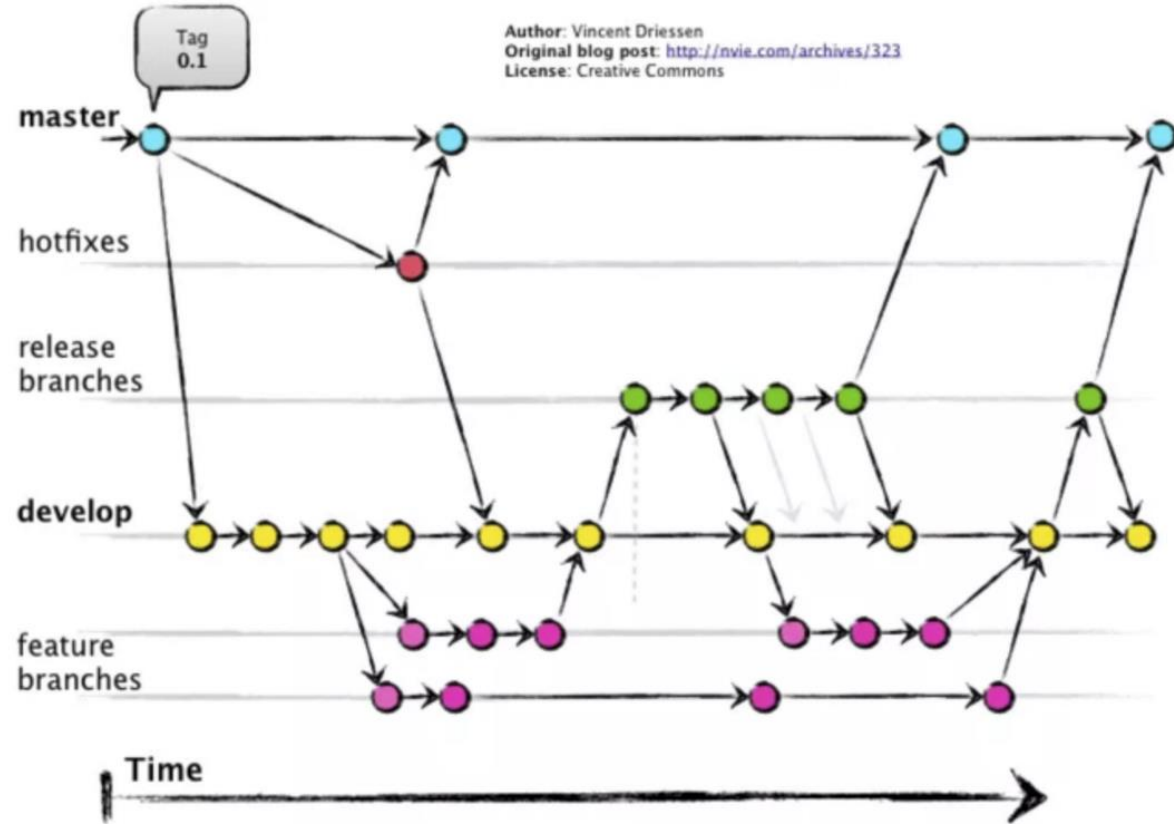
## Git을 사용하는 이유?

Version

Branch

Merge

Commit



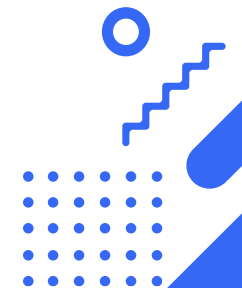
## 2. GIT, GITHUB 기본설정

Local repository, Remote repository  
생성 및 연결





## repository 생성 과정



## 명령어

### git init

git init는 새로운 Git 저장소(repository)를 생성할 때 사용하는 Git 명령어 입니다

### git remote

현재 프로젝트에 저장된 리모트 저장소 확인, 옵션으로 원격 repo 조작 가능

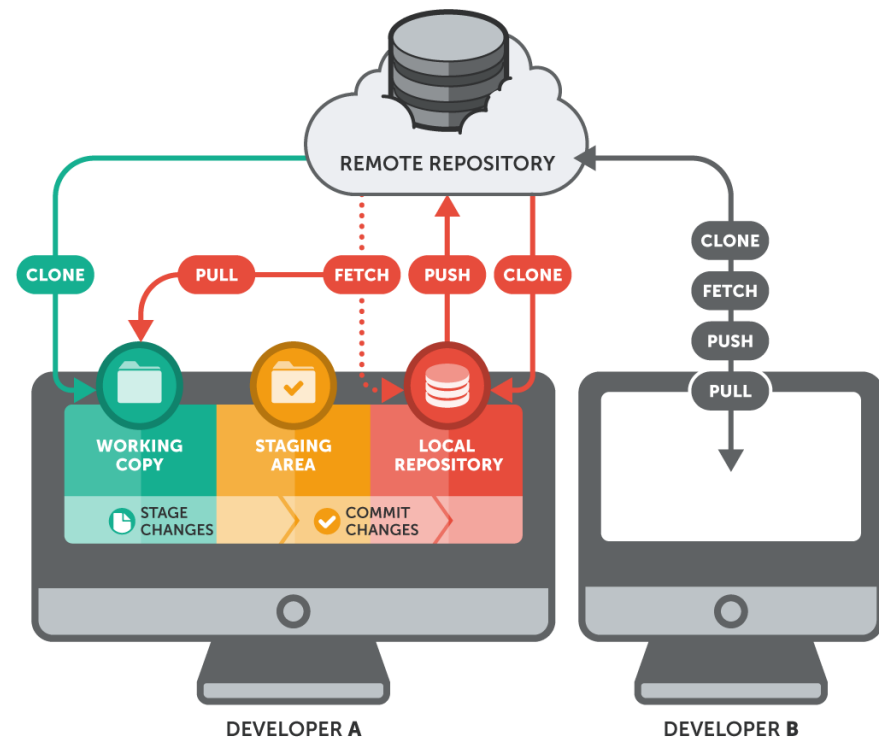
-v : remote 상태보기

add <name> <url> : url주소로 name이라는 remote주소 설정

### git clone <url>

clone은 git clone <리모트 저장소 주소>를 이용하여 사용할 수 있습니다. 이 명령어는 원격 저장소에 있는 프로젝트를 가져오는 역할을 합니다.

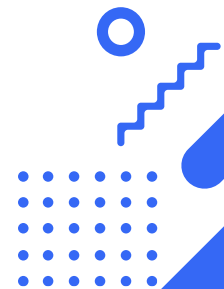
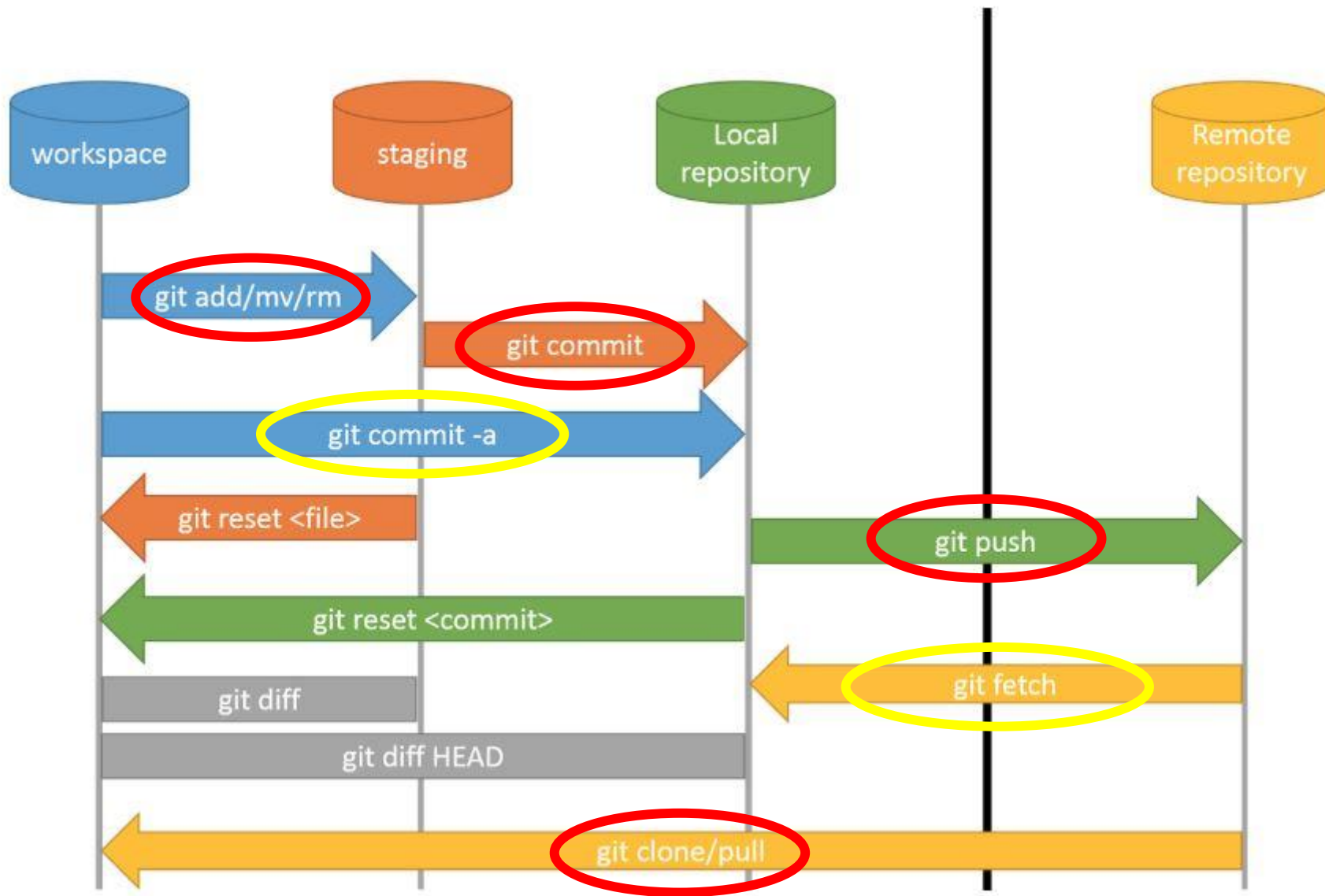
master 브랜치를 자동으로 가져오며 origin으로 remote도 add해 줍니다. git init 명령어로 git 프로젝트가 아닌 곳에서도 사용할 수 있는 명령어입니다.



### 3. GIT 기본 명령어

수정 작업 이후에 git을 이용해서 작업내용을 관리하고 공유하는데 사용되는 기본 명령어인 add, commit, push, pull 명령어 설명







## 명령어

### git add <filename>

git add는 변경사항을 stage영역에 올리는데 사용됩니다. 파일명 대신 \* 기호를 사용하면 모든 변경내역을 stage에 업로드합니다.

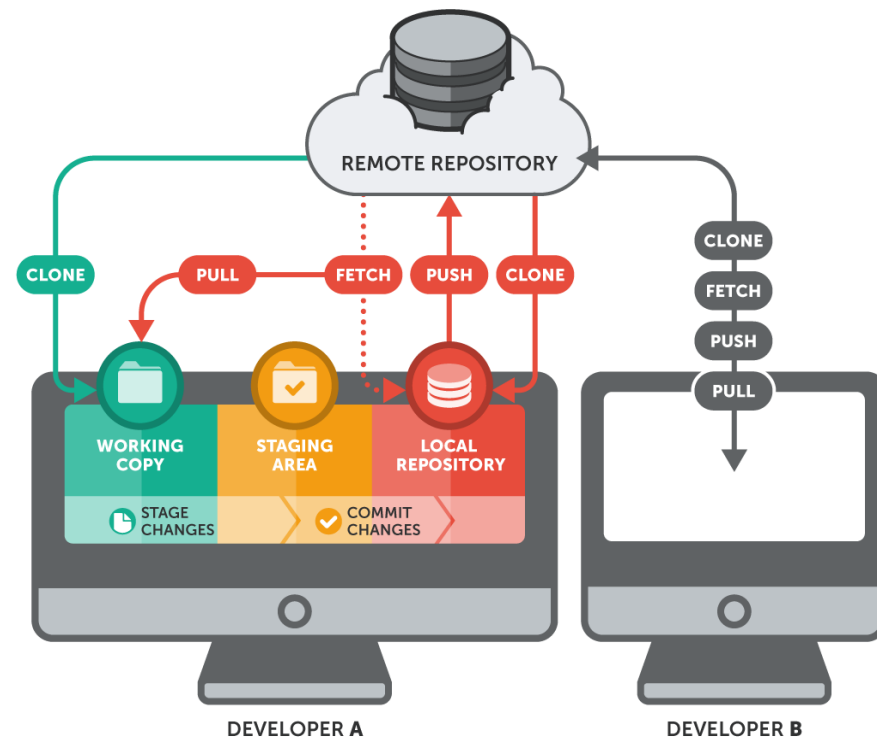
### git status

git status는 현재 프로젝트의 변경사항을 추적합니다. 트래킹되지 않은 변경사항은 붉은색으로, 트래킹된 변경사항은 녹색으로 출력됩니다. add하게되면 해당 변경사항은 tracking되고, 이후에 commit 작업으로 변경사항을 local repo에 반영할 수 있습니다.

### git commit

stage 영역에 있는 변경사항을 .git(local repo)에 반영합니다. 보통 -m 옵션을 주고 뒤에 "commit message"를 붙여서 사용합니다.

ex) git commit -m "chage the axi\_interface with stream interface"





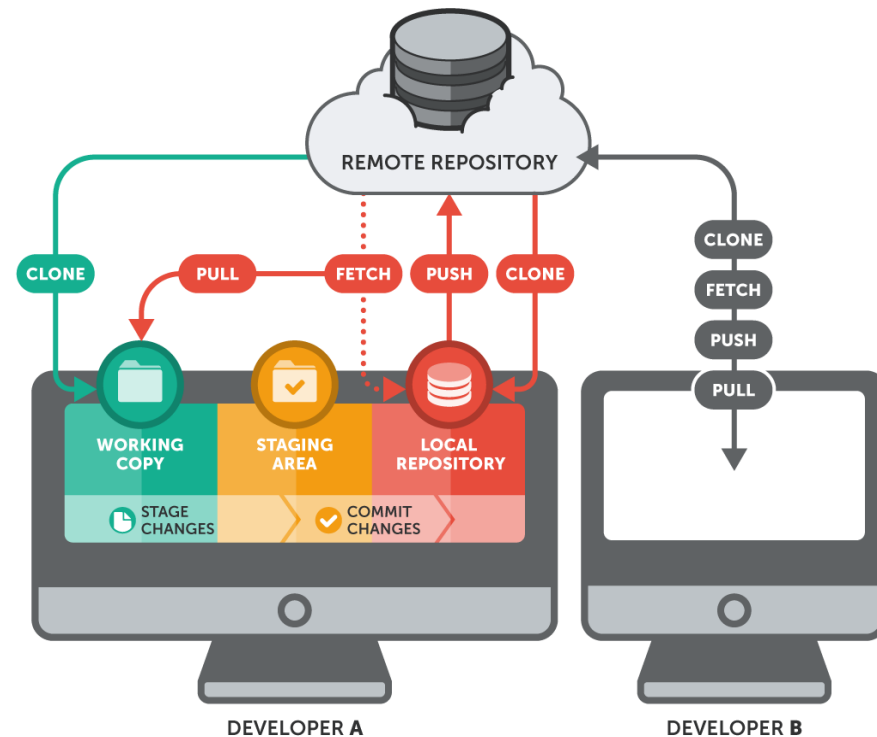
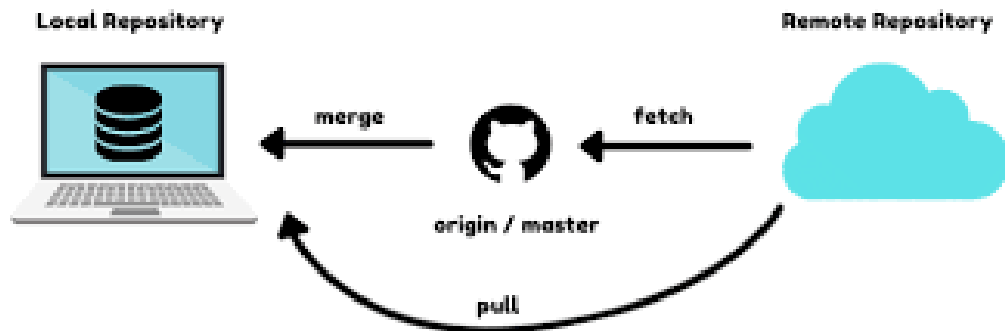
## 명령어

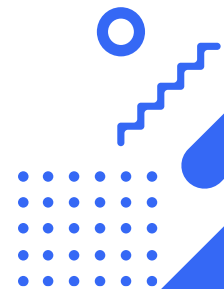
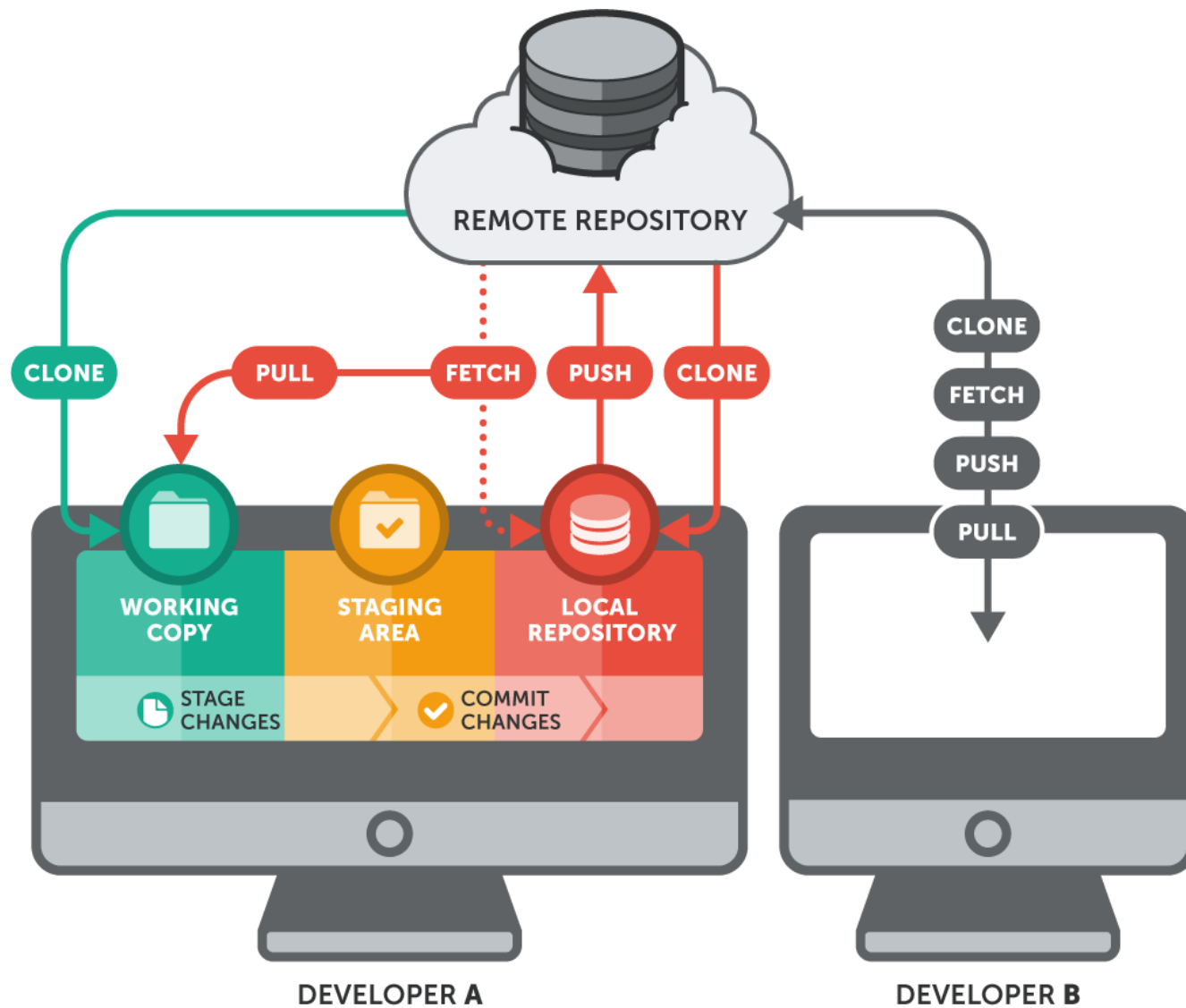
### git push <remotename> <branch name>

git push는 local영역에 있는 변경사항을 remote영역으로 업로드 할 때 사용됩니다.  
default일 경우에는 연결되어있는 origin repo의 master branch로 push합니다.

### git pull

remote 영역의 repo의 변경사항을 local영역으로 가져오고(fetch)  
+ 기존의 local 영역의 내용과 합치는 작업까지 함께 수행합니다.(merge)  
fetch와 merge를 동시에 진행하는 작업으로 충돌이 있을 수 있습니다.



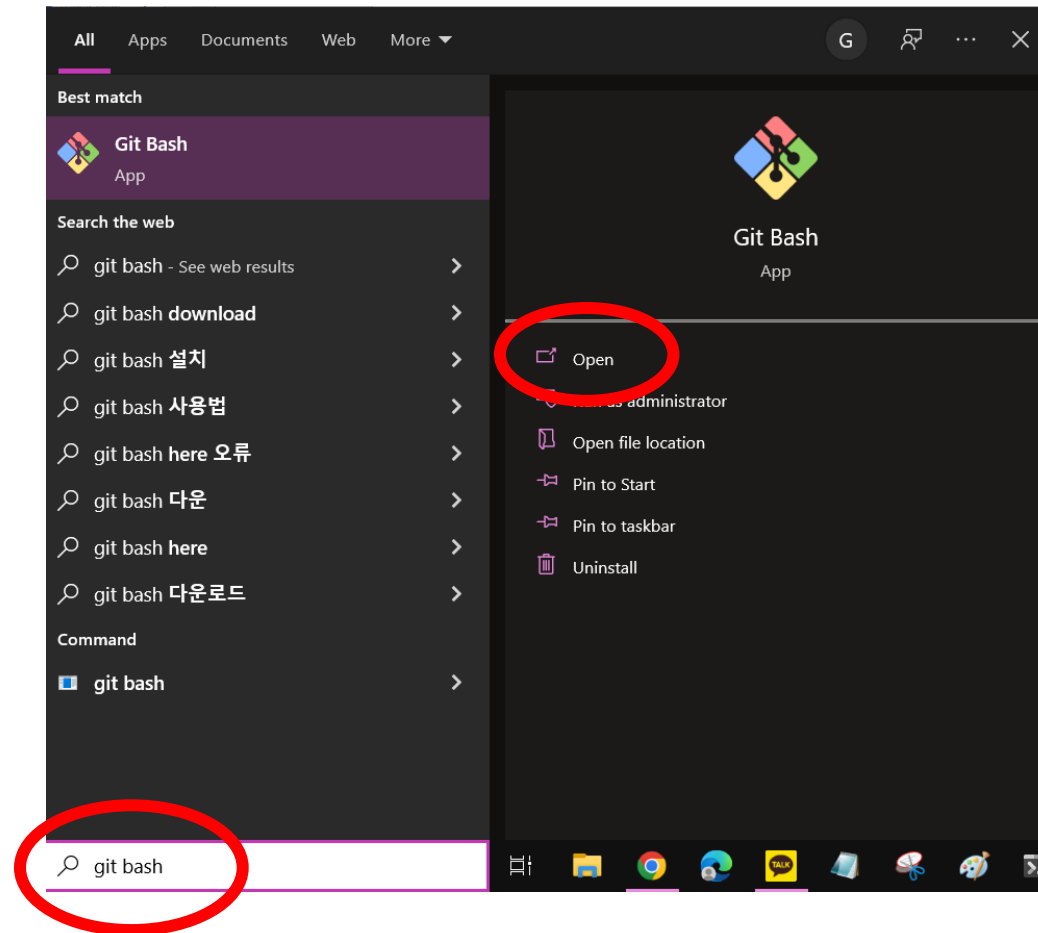


## 4. WORK FLOW





## 검색기에 git bash 검색해서 열기





## 리눅스 기본 CLI(Command Line Interface)

현재 본인들이 사용하고 계신 마우스 기능, 바탕화면의 폴더들, 더블클릭이나 오른쪽 클릭과 같은 활동으로 할 수 있는 활동들 모두 GUI(Graphic User Interface) 환경을 이용하는 것 입니다. Git bash와 같은 CLI환경에서는 명령어로 이러한 활동을 합니다.

폴더 = 디렉토리

pwd : 현재 디렉토리 경로를 보여줌

cd + (하위 디렉토리 이름) : 하위 디렉토리로 이동

상위 디렉토리를 이동하려면 cd + ".."(마침표 2개 입니다) 입력하면됨  
리눅스에서는 "." 마침표 1개가 현재 디렉토리, ".." 마침표 2개가 상위 디렉토리를 의미함.  
하위로 갈때는 파일이름

ls : 현재 디렉토리의 파일들 리스트로 출력, -a or -al 옵션으로 숨김 파일들도 확인가능  
(.git파일을 보려면 이걸로 보면됨)

CLI는 Tab을 자동완성으로 지원하며, cd와 같은 명령어에서 tab을 누르면 하위 디렉토리 이름중에 해당하는 것들을 자동완성합니다. 자동완성을 할 파일이 여러개 일 경우 아무것도 뜨지 않고, tab을 한번 더 누르면 겹치는 것들을 리스트로 보여줍니다.





Bash에서 cd명령어를 사용해서 Desktop의 SKKU\_TermProject로 이동  
(경로는 각자 다를 수 있습니다.)

```
MINGW64:/c/Users/gju06

gju06@LAPTOP-ER5F0VF4 MINGW64 ~
$ pwd
/c/Users/gju06

gju06@LAPTOP-ER5F0VF4 MINGW64 ~
$ cd Desktop/
```

```
MINGW64:/c/Users/gju06/Desktop/workspace/SKKU_TermProject

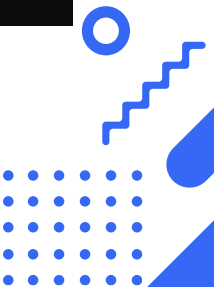
gju06@LAPTOP-ER5F0VF4 MINGW64 ~
$ pwd
/c/Users/gju06

gju06@LAPTOP-ER5F0VF4 MINGW64 ~
$ cd Desktop/

gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop
$ cd workspace/

gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace
$ cd SKKU_TermProject/

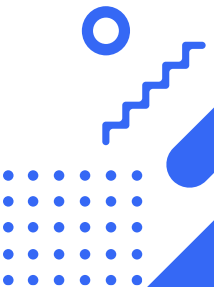
gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject (main)
$
```





ls -al을 이용해서 해당 파일에 목록들을 확인해보세요.  
“.git” 이 있다는 것은 GIT이 해당 파일을 버전관리 하고있다는 것입니다.

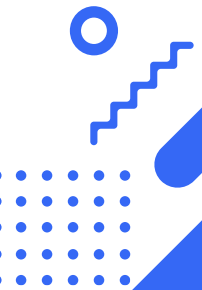
```
gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject (main)
$ ls -al
total 21
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:25 ./
drwxr-xr-x 1 gju06 197609 0 Mar 25 11:10 ../
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:49 .git/
drwxr-xr-x 1 gju06 197609 0 Apr 10 16:45 B00/
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:25 Hello_git/
-rw-r--r-- 1 gju06 197609 18 Mar 25 11:10 README.md
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:46 RTL/
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:47 SIM/
drwxr-xr-x 1 gju06 197609 0 Apr 10 16:46 VITIS_IF/
drwxr-xr-x 1 gju06 197609 0 Apr 10 16:45 VIVADO_PROJ/
```





ls -al을 이용해서 해당 파일에 목록들을 확인해보세요.  
“.git” 이 있다는 것은 GIT이 해당 파일을 버전관리 하고있다는 것입니다.

```
gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject (main)
$ ls -al
total 21
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:25 ./
drwxr-xr-x 1 gju06 197609 0 Mar 25 11:10 ../
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:49 .git/
drwxr-xr-x 1 gju06 197609 0 Apr 10 16:45 B00/
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:25 Hello_git/
-rw-r--r-- 1 gju06 197609 18 Mar 25 11:10 README.md
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:46 RTL/
drwxr-xr-x 1 gju06 197609 0 Apr 22 00:47 SIM/
drwxr-xr-x 1 gju06 197609 0 Apr 10 16:46 VITIS_IF/
drwxr-xr-x 1 gju06 197609 0 Apr 10 16:45 VIVADO_PROJ/
```







## git 업데이트 하기

git은 하나의 큰 저장공간을 repository라고 부릅니다. 여기선 SKKU\_TermProject가 가장 큰 단위인 repository입니다.

git status : 현재 해당 repo의 상태를 보여줌.

git pull origin main : 현재 repo를 업데이트하는 명령어. 각각의 단어는 아래와 같은 의미입니다.

git(git 명령어)

pull(업데이트 내용 가져오기)

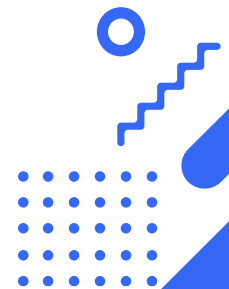
origin(github)

main(branch이름)

```
git@SGSLI70P-EP5F0YF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

git@SGSLI70P-EP5F0YF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject (main)
$ git pull origin main
From https://github.com/sg05060/SKKU_TermProject
* branch          main          -> FETCH_HEAD
Already up to date.
```





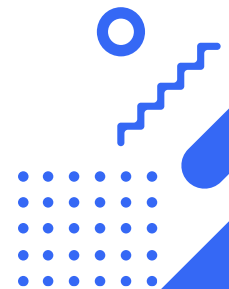
## 작업하기

```
gju06@LAPTOP-ER5F0YF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ ls -al
total 7
drwxr-xr-x 1 gju06 197609  0 Apr 22 00:25 ./
drwxr-xr-x 1 gju06 197609  0 Apr 22 00:25 ../
-rw-r--r-- 1 gju06 197609 20 Apr 22 00:25 박상현.txt
-rw-r--r-- 1 gju06 197609 20 Apr 22 00:25 장재성.txt
-rw-r--r-- 1 gju06 197609  9 Apr 22 00:25 홍나경.txt

gju06@LAPTOP-ER5F0YF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ vim 최보열.txt

gju06@LAPTOP-ER5F0YF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ ls -al
total 12
drwxr-xr-x 1 gju06 197609  0 Apr 22 01:23 ./
drwxr-xr-x 1 gju06 197609  0 Apr 22 00:25 ../
-rw-r--r-- 1 gju06 197609 20 Apr 22 00:25 박상현.txt
-rw-r--r-- 1 gju06 197609 20 Apr 22 00:25 장재성.txt
-rw-r--r-- 1 gju06 197609 24 Apr 22 01:23 최보열.txt
-rw-r--r-- 1 gju06 197609  9 Apr 22 00:25 홍나경.txt

gju06@LAPTOP-ER5F0YF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$
```





## 변경사항 확인

파일을 추가했으니 git status를 사용해서 확인해보면 untrack파일이 보이는 것을 확인가능합니다.  
(한글과 같은 경우에는 unicode로 쪼개져서 숫자로 보입니다)

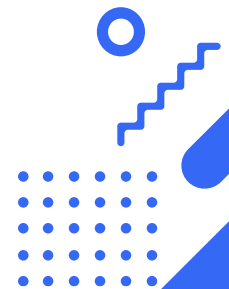
따라서 해당 파일을 먼저 stage영역에 올려주어야 합니다.

```
gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        "#354#265#234#353#263#264#354#227#264.txt"

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

gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$
```





## Stage에 파일 변경내역 업로드

변경 내역을 stage 영역에 업로드 하기 위해서 git add 명령어를 사용합니다.

add뒤에 수정, 추가, 삭제 한 파일이름을 적어야 되지만, 모든 변경내역을 stage에 올리겠다는 명령어로 뒤에 -A라는 옵션을 줄 수 있습니다.

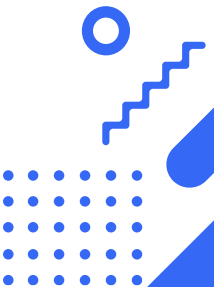
git add -A

```
gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ git add -A

gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   "₩354₩265₩234₩353₩263₩264₩354₩227₩264.txt"

gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$
```





## Stage의 변경사항을 local에 저장하기

지금의 변경내역은 stage라는 가상의 공간에 있습니다. 이를 local(지금은 노트북이겠죠)에 저장시키기 위해서 commit을 해줘야 합니다. commit은 이 변경내역을 확정짓겠다는 의미로 생각하시면 됩니다.

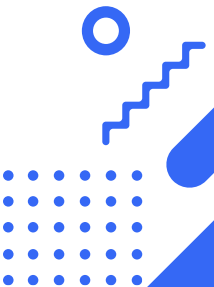
`git commit -m "commit 내용 여기에 적어주세요 "`

```
gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ git commit -m "add test txt file"
[main 00afa0c] add test txt file
1 file changed, 1 insertion(+)
create mode 100644 "Hello_git/₩354₩265₩234₩353₩263₩264₩354₩227₩264.txt"

gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$
```





## Git log보기

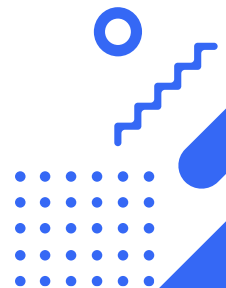
물론 github에서도 볼 수 있지만 해당 화면에서도 볼 수 있습니다.

git log 명령어를 사용하시면 되고, 해당 log화면에서 나가려면 키보드 'q'를 눌러주세요.

git log

```
~/Desktop/workspace/SKKU_TermProject/Hello_git (main)  
$ git log  
commit 68ef30e301569a87a6c74aa96779dd08c52116d0 (HEAD -> main)  
Author: gju06051 <gju06051@gmail.com>  
Date: Fri Apr 22 01:30:02 2022 +0900  
  
    add test txt file  
  
commit 175be00b57347efc3e71497546ec17c8911a7b2 (origin/main, origin/HEAD)  
Author: gju06051 <gju06051@gmail.com>  
Date: Fri Apr 22 00:49:33 2022 +0900  
  
    Making new directory for Counter in RTL& SIM. Upload Done_Counter design spec  
  
commit d89aa9cb8a7e53552c5f226c4fe861f7b8c22b9d  
Author: Nakyung <nakyung0305@gmail.com>  
Date: Thu Apr 21 20:23:37 2022 +0900  
  
    실습  
  
commit 7e1e583a852a7c15fddd29cf4c0925ec98f7a477  
Author: JJS <louis12057874@gmail.com>  
Date: Thu Apr 21 20:14:59 2022 +0900  
  
    실 습  
  
commit 6d5e741cdcdedc6910ce7d6e4685bb3f2583b60f  
Author: sg05060 <psh2018314072@gmail.com>  
Date: Thu Apr 21 20:06:49 2022 +0900  
  
    git 실습
```

The terminal output shows the history of commits. A red circle highlights the first commit (68ef30e301569a87a6c74aa96779dd08c52116d0) and its details. A red circle highlights the second commit (175be00b57347efc3e71497546ec17c8911a7b2) and its details. A red circle highlights the third commit (d89aa9cb8a7e53552c5f226c4fe861f7b8c22b9d) and its details. A red circle highlights the fourth commit (7e1e583a852a7c15fddd29cf4c0925ec98f7a477) and its details. A red circle highlights the fifth commit (6d5e741cdcdedc6910ce7d6e4685bb3f2583b60f) and its details.





## Git push로 github에 업로드하기

이제 모든 과정이 끝났고 해당 commi되어있는 변경사항을 github에 업로드하기 위해서 push 명령어를 사용해줍니다.

git push origin main

해당 단어들의 의미는 다음과 같습니다

git : git 명령어

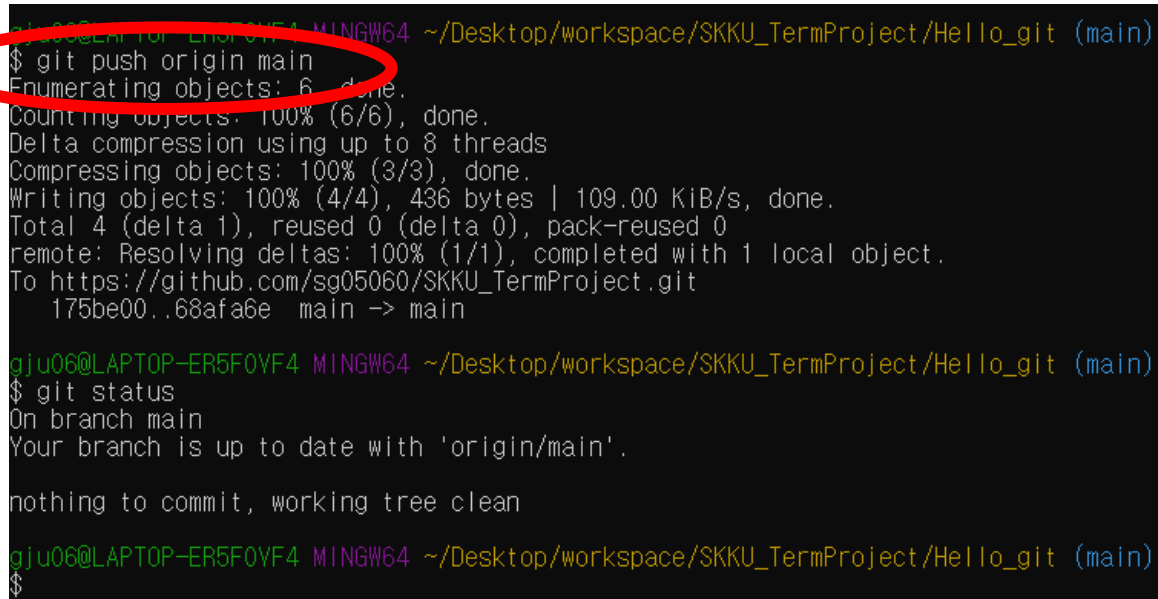
push : commit 변경사항을 원격저장소에 보내기

origin : 여기선 github의 SKKU\_TermProject사이트를 의미

main : branch이름

이 과정이 끝나면 하나의 작업이 완료됩니다.

pull -> 작업 -> add -> commit -> push  
가 전체적인 과정입니다.



```
gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 436 bytes | 109.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/sg05060/SKKU_TermProject.git
  175be00..68afa6e  main -> main

gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

gju06@LAPTOP-ER5F0VF4 MINGW64 ~/Desktop/workspace/SKKU_TermProject/Hello_git (main)
$
```



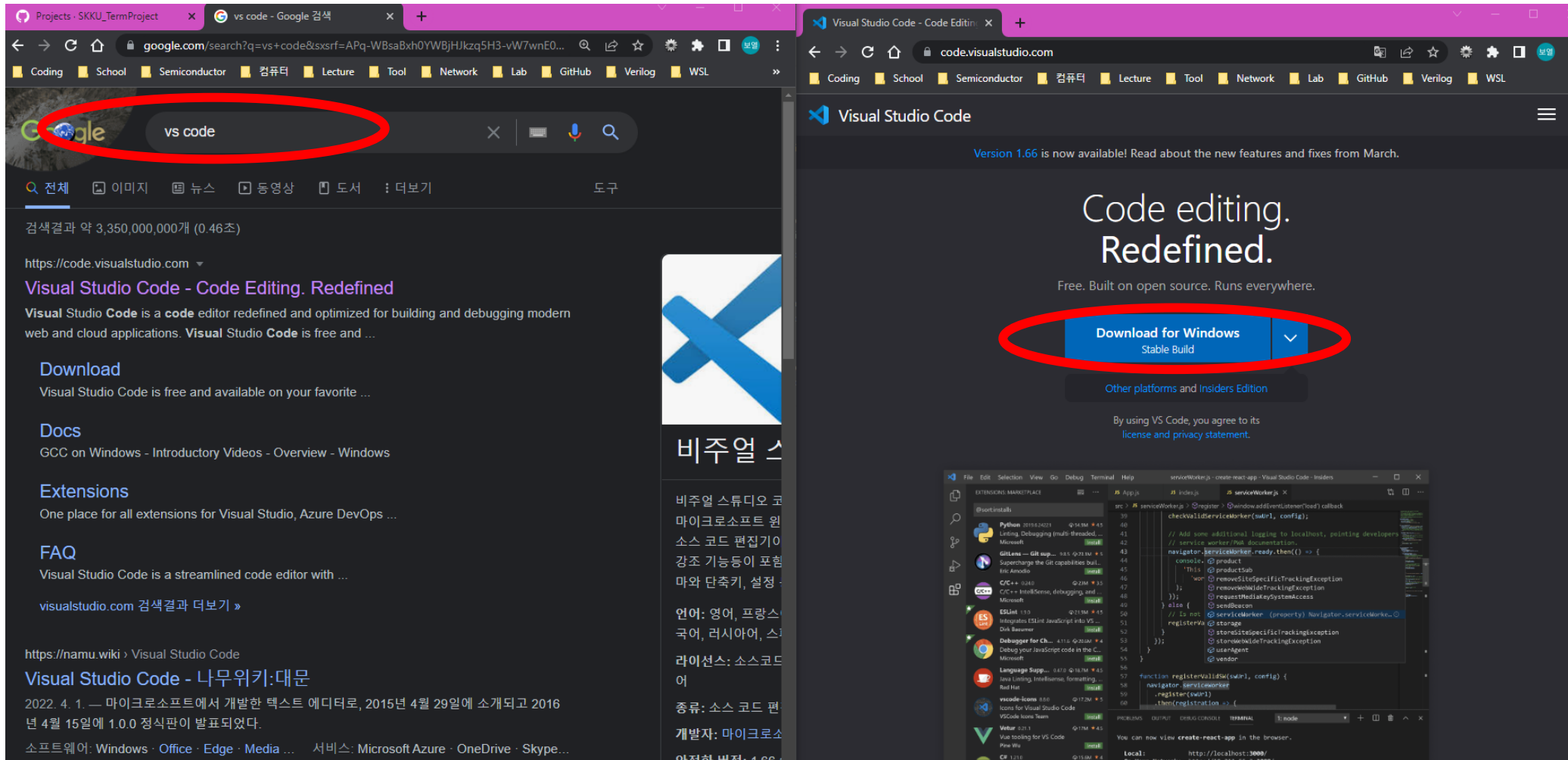
## 5. EDITOR 설정

VS CODE를 쓰면 자동완성 기능으로 실수도 줄어들고 가독성도 좋습니다.





## VS code 사이트 가기

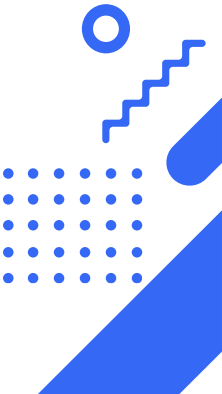




## 다운로드 받기

<https://penguigoon.tistory.com/185>

해당 게시물 보시면 됩니다. 사실 뭐 그냥 yes만 계속 하시면되요.





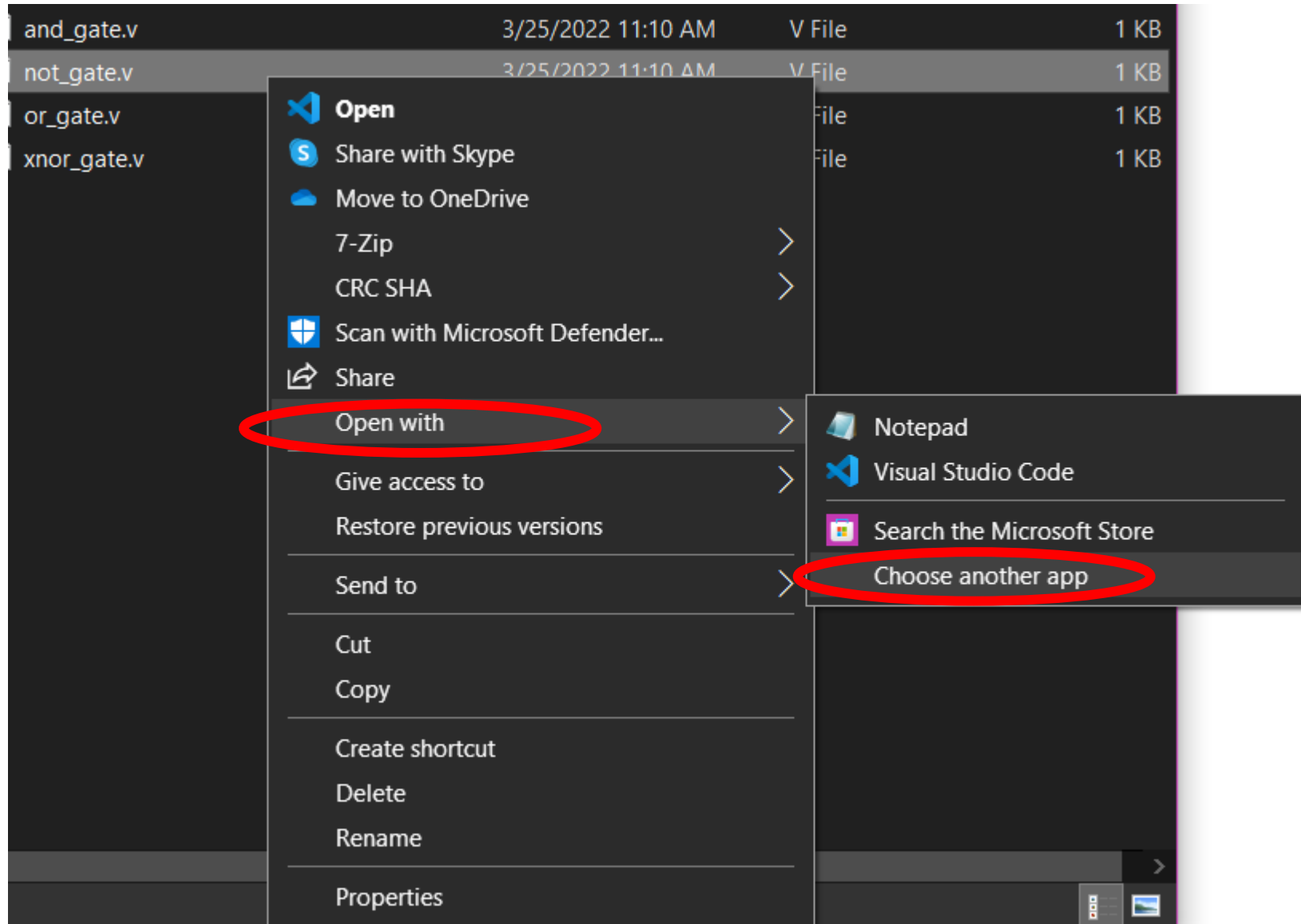
## Verilog 열 때 기본설정

다음으로

Verilog 파일을 열 때 vs code 에디터로 열리게 설정해봅시다.

먼저 아무 Verilog 파일이나 오른쪽 클릭합니다.

open with가 보일 겁니다.  
(한글로는 다른 경로 열기? 뭐 이렇게 적혀있을 겁니다)

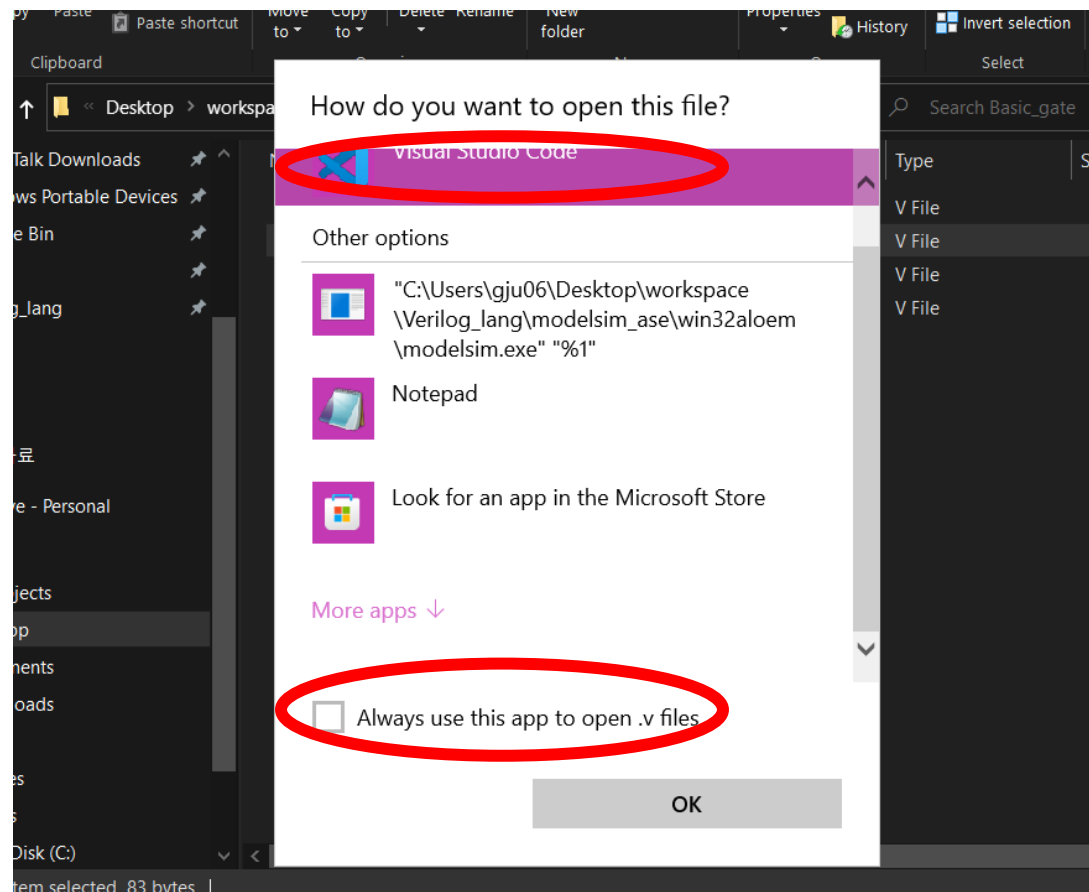




## Verilog 열 때 기본설정

이제 vscode를 선택하시고  
체크박스를 체크 하셔서 항상  
vs code로 열도록 설정해주  
면 준비는 끝납니다.

이렇게 설정하면 modelsim  
을 켜서 project를 만들고,  
Verilog 파일을 수정하려고  
마우스 더블클릭 or open을  
하면 vscode 열리게 됩니다!





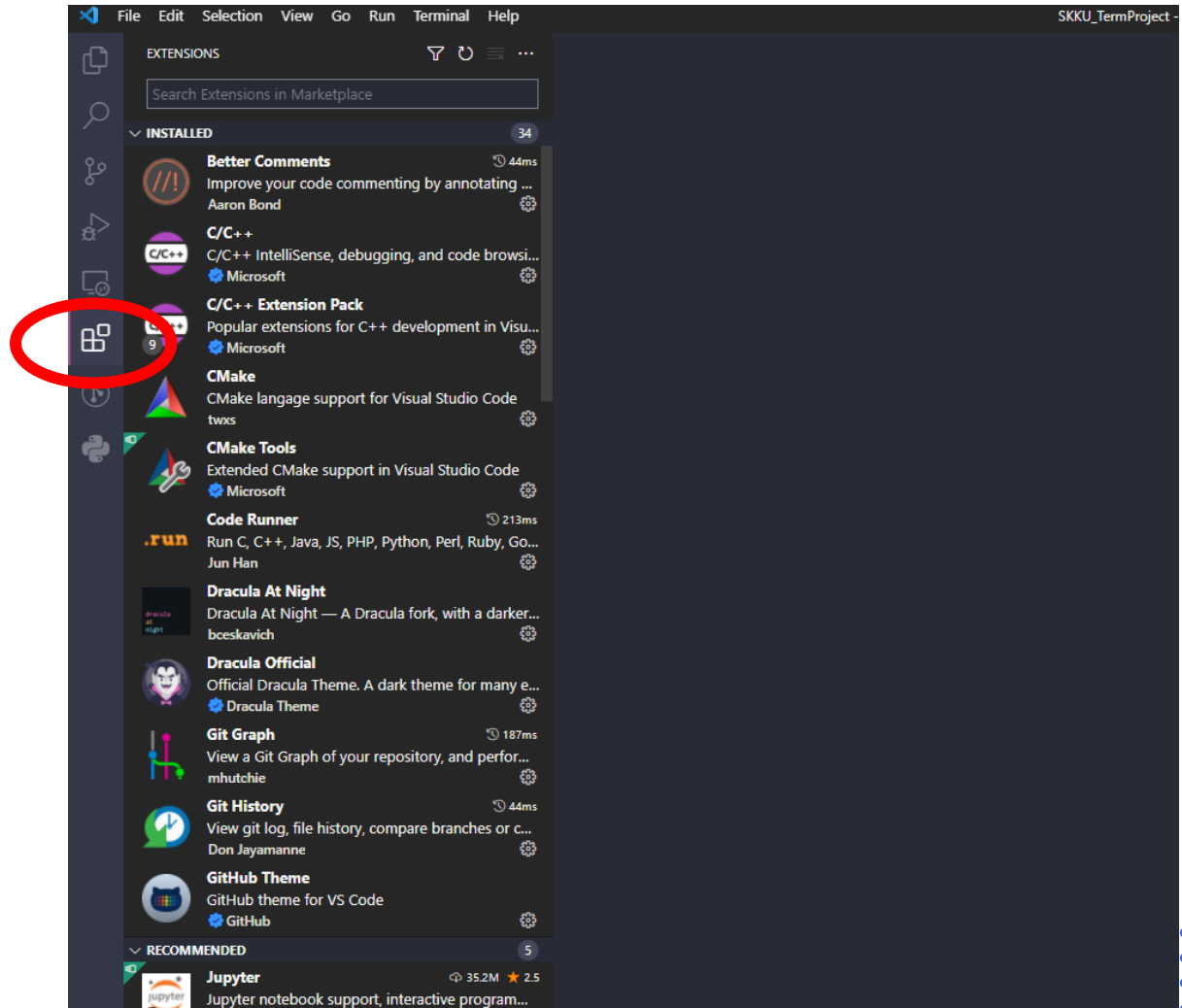
## Extension 설치

vs code를 사용하는 목적이자 vs code가 사람들이 많이 쓰는 이유입니다.

extension이라고 추가적인 기능들을 다운로드 받아서 사용하는데 정말 많은 기능들이 있습니다. 대부분 무료이고요.

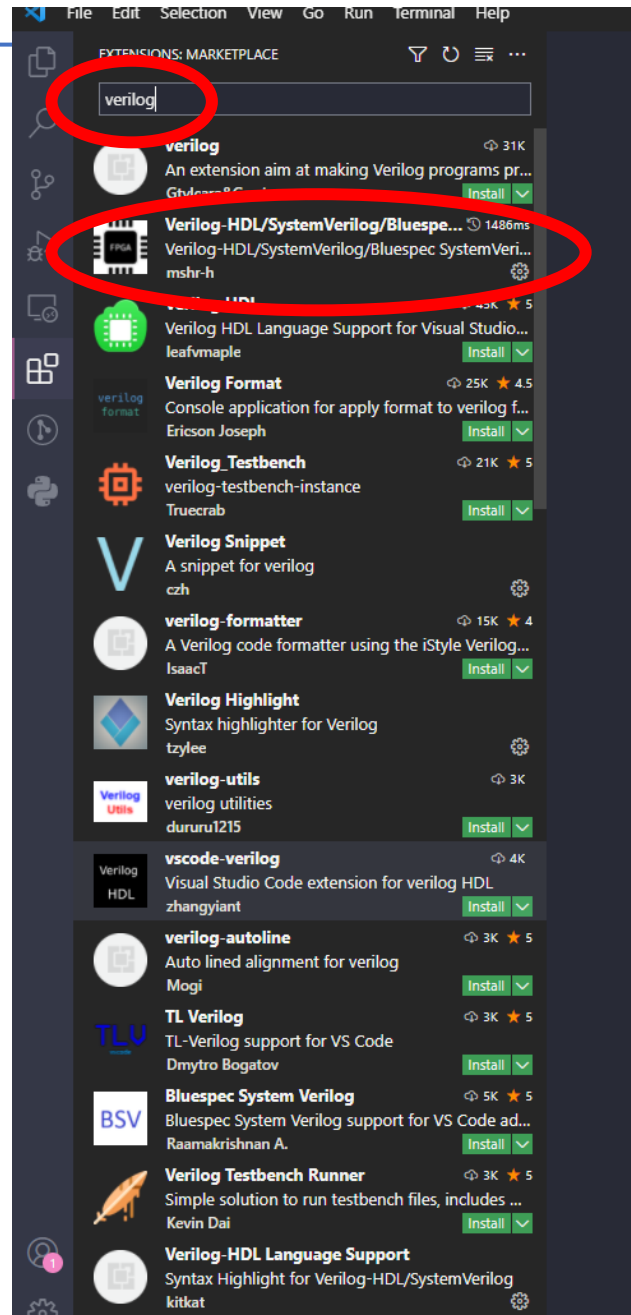
저희는 Verilog 문법 자동완성을 도와주는 것과 몇 가지 시각적으로 도움되는 것들을 다운로드 할 것입니다.

왼쪽에 extension을 열어주세요.



## Extension 설치

위에 검색어 창에 Verilog 를 입력하시고 2번째 있는 것을 다운로드 하면됩니다. 해당 아이콘을 눌러주세요.

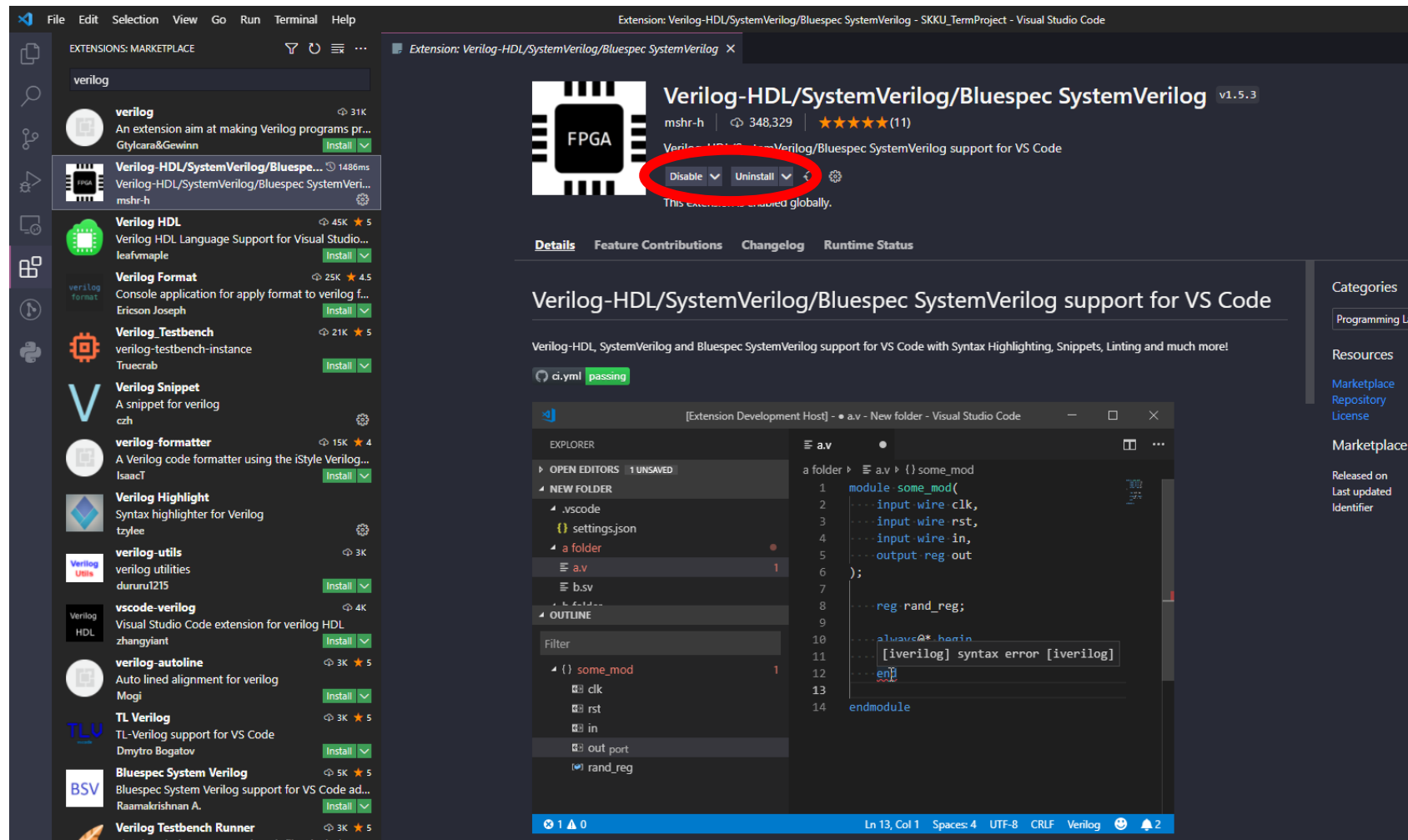




## Extension 설치

누르면 옆에 다음과 같은 창이 뜨고 install을 해주시면 됩니다.

extension의 장점이 이렇게 install만 해주면 알아서 내부에서 설정 및 실행을 해줘서 따로 뭔가 설정할 필요가 없습니다.





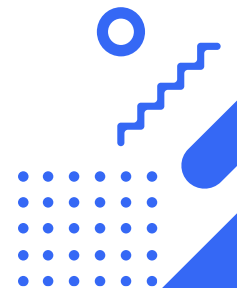
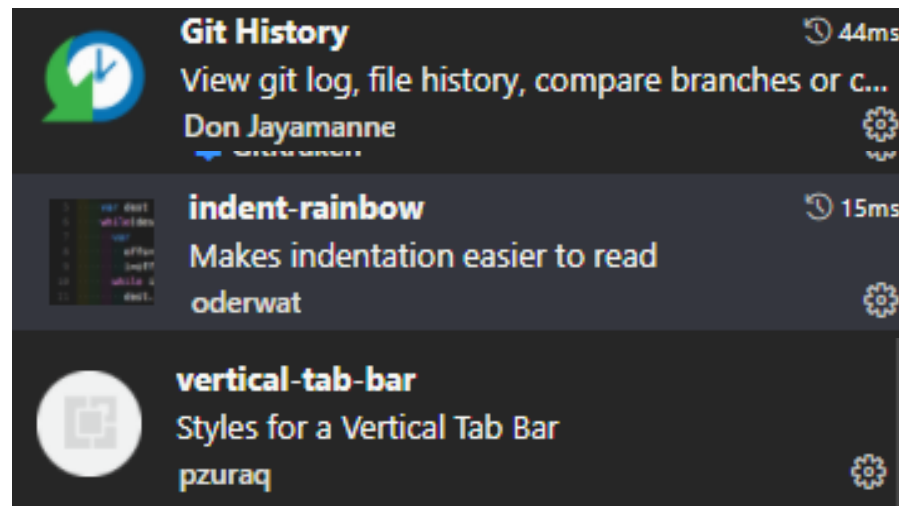
## Extension 목록

일단 기본적으로 저거 1가지 (Verilog~~)면 되는데 추가적으로 설치하면 좋은 것들을 알려드립니다.

방금 설치한 것처럼 이름을 검색해서 install하시면 됩니다.

git history는 git관리하는 폴더 변경사항을 좀 더 직관적으로 확인할 수 있게 도와주고,

indent rainbow와 tab bar는 코드 tab칸(space4칸)을 맞춰줘서 코드 짤 때 깔끔하게 만들어줍니다.



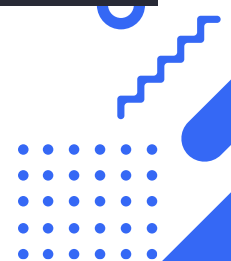
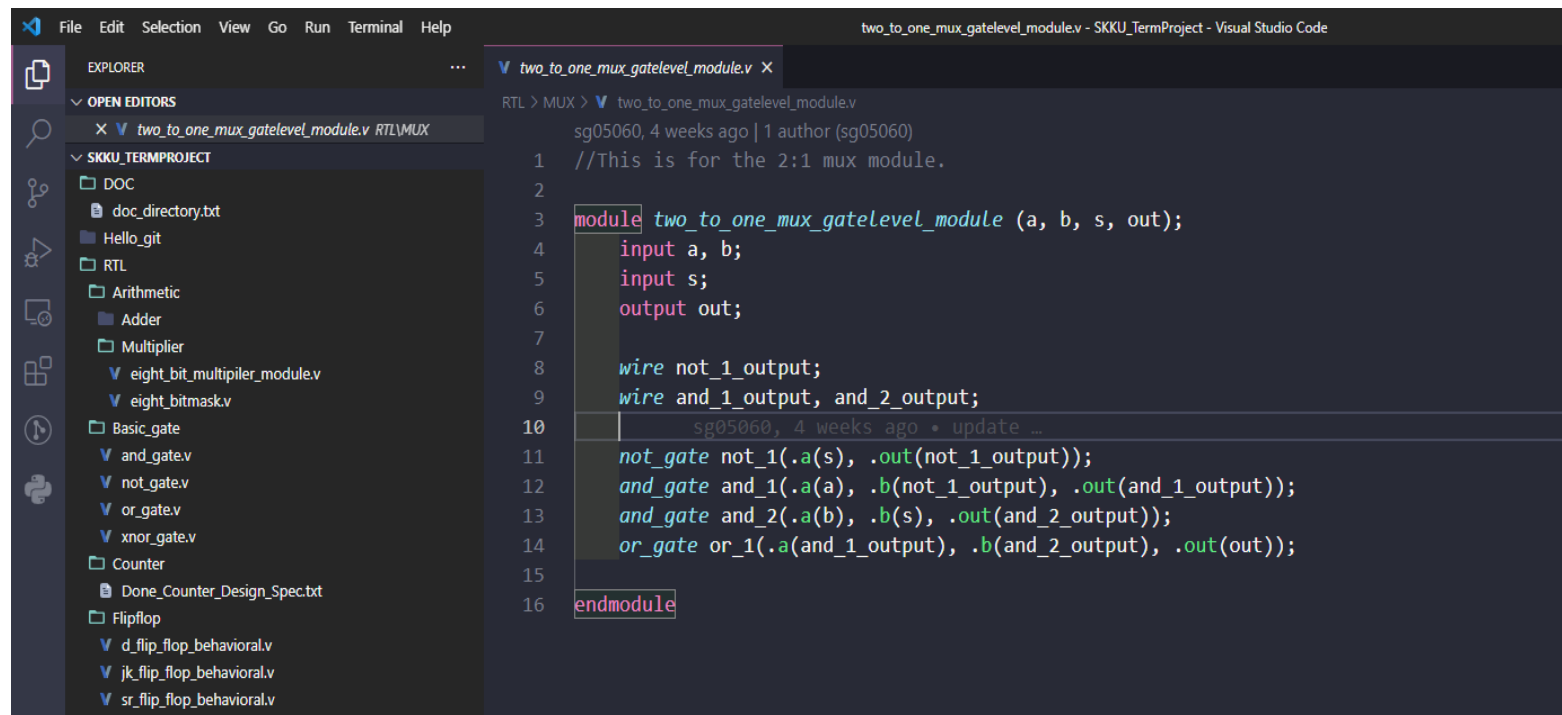
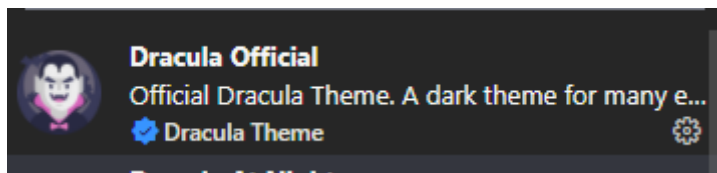





## Extension 목록

추가로 좀 더 간지나고 싶으면 theme를 install 해주면 됩니다.

종류가 엄청 많은데 저는 Dracula 사용하고 있습니다.





## 6. GITHUB에서 사용되는 유용한 기능들

Issue, project, milestone에 대한 설명





## Issue

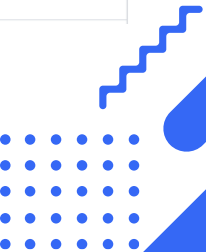
프로젝트에서 발생하는 모든 문제들을 관리하기 위한 수단으로 issue 생성이 허용된 모든 repo에서 생성할 수 있다.

issue의 종류마다 라벨링을 다르게 하여서 issue를 주제별로 구분할 수 있다.

assign기능으로 책임자를 선정할 수 있다.

The screenshot shows the GitHub interface for the repository 'khyunjee / Github\_IssueTracking'. The 'Issues' tab is selected, showing 2 issues. Below the navigation bar, there are tabs for 'Labels' and 'Milestones', a search bar for labels, and a 'New label' button. A table lists 11 labels with their names, descriptions, and edit/delete options.

11 labels		Sort ▾
bug	Something isn't working	Edit Delete
documentation	Improvements or additions to documentation	Edit Delete
duplicate	This issue or pull request already exists	Edit Delete
enhancement	New feature or request	Edit Delete
fix	bug fix	1 open issue or pull request Edit Delete
good first issue	Good for newcomers	Edit Delete





## Milestone

khyunjiee / Github\_IssueTracking

Unwatch 1 Star 0 Fork 0

Code Issues 2 Pull requests 1 Actions Projects 0 Wiki Security 0 Insights Settings

Labels Milestones New milestone

2 Open 0 Closed Sort

**new**  
Due by April 25, 2020 Last updated less than a minute ago  
new milestone  
0% complete 0 open 0 closed  
Edit Close Delete

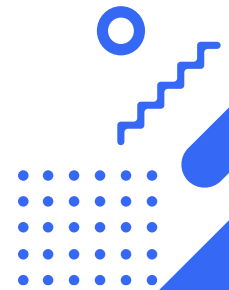
**main**  
Due by April 22, 2020 Last updated about 15 hours ago  
milestone for main page  
50% complete 1 open 1 closed  
Edit Close Delete

issue에 labeling과 assignesss를 부여해도  
기능별 유사한 issue가 존재하며,

이를 찾거나 구현정도를 파악하기 위해서는 일  
일이 추적해야되는 필요성이 있다.

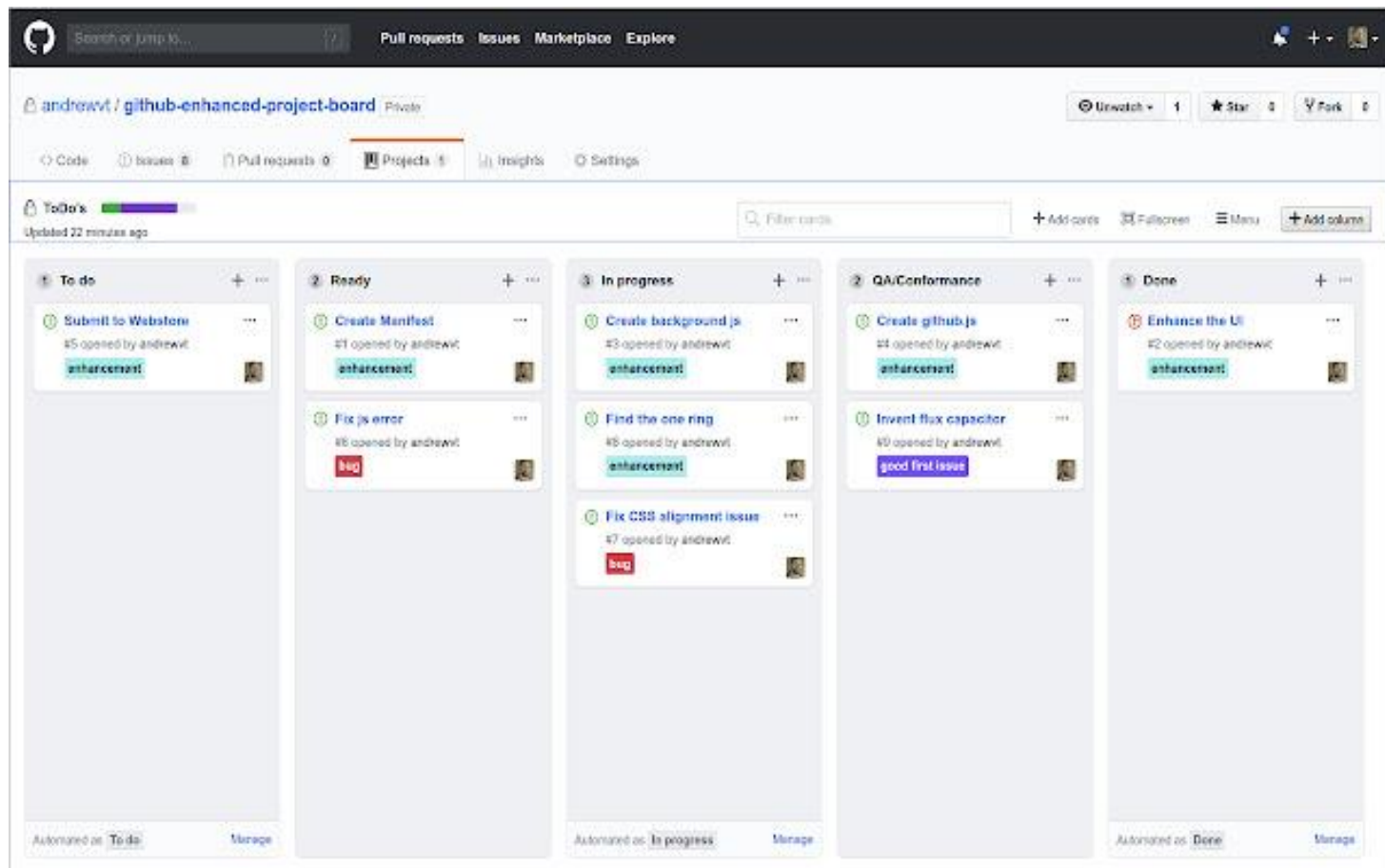
이를 편리하게 해주기 위해서 도입된 것이  
Milestone(이정표)이다.

이슈를 그룹화하고 해당 이슈들의 해결 진척도  
를 퍼센트로 환산해서 보여준다.



## 5. Github에서 사용되는 유용한 기능들

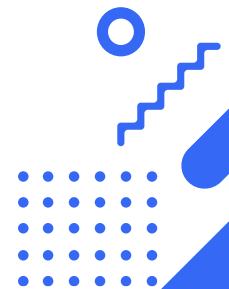
## Projects



To do list나 issue등을 작성하고 이에 대한 진척도를 정리해놓을 수 있는 보드양식이다.

그림처럼 todo, ready, progress, QA, done 등 자유롭게 issue 및 작업들을 분류하고,

GUI형식으로 이동시키거나 control 할 수 있다.





# THANKS !

2018312959 최보열



Scalable Architecture Lab