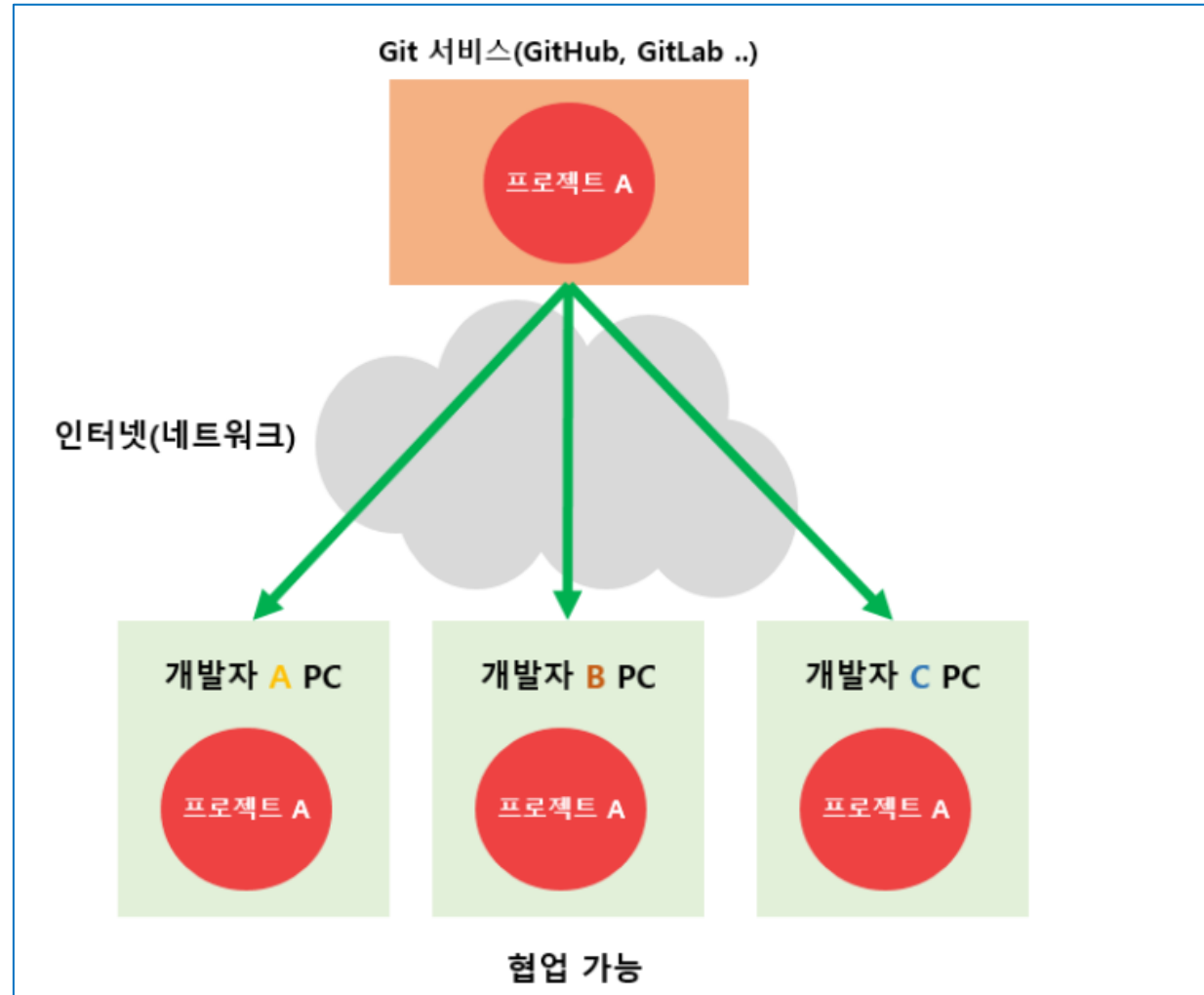
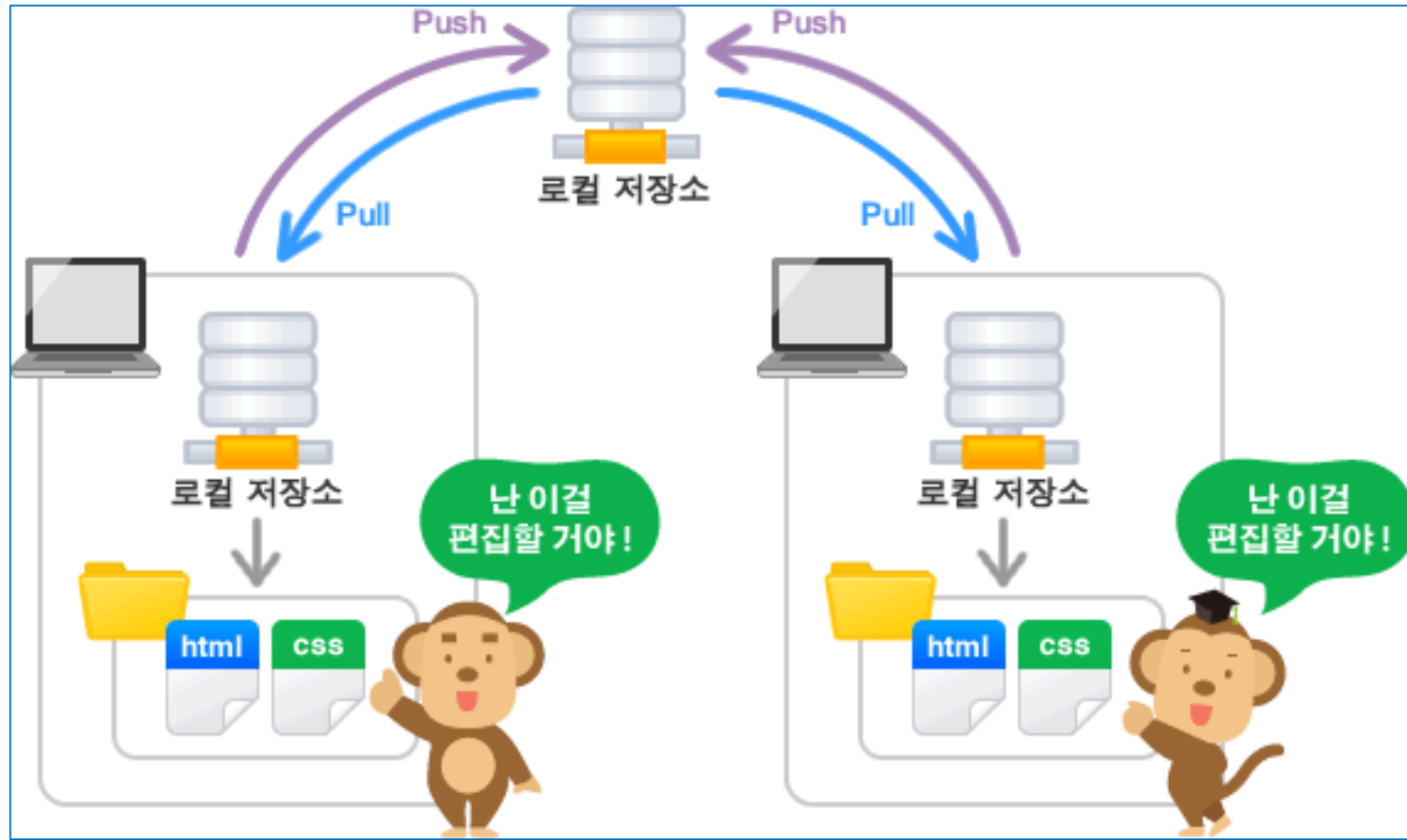


깃 사용법

git 서비스



git hub란



git이란

- **Git**이란 **버전 관리 시스템**(VCS, Version Control System)의 한 종류이다.
- 그렇다면 여기서 말하는 버전 관리란 무엇이고, 왜 필요할까?
- 버전 관리란 이름 그대로 여러 파일을 하나의 버전으로 묶어 관리하는 것이다. 버전을 관리할 예를 들어 말해보자.
- 예를 들어, 우리가 제출해야 하는 레포트가 하나 있다. 처음에 'report.txt'를 만들었다가 내용이 바뀔 경우 'report_최종.txt'으로 바꿨다가 다시 수정을 거치면서 'report_진짜최종.txt' 이런 식으로 만든 경험이 있을 것이다.
- 우리는 여러 파일들을 복사, 저장, 백업 등을 하였고 이것을 버전 관리라고 부른다.

버전 관리 시스템

- **클라이언트-서버 모델**

- 하나의 중앙 서버가 존재하며, 여러 클라이언트들은 중앙 서버에서 각자 맡은 파트만 가져와서 작업하고, 다시 중앙으로 통합하는 것을 말한다.
- 대표적 시스템으로 CVS, Subversion 등이 있다.

- **분산 모델**

- 하나의 중앙 서버가 존재하지만, 여러 클라이언트들은 **각자의 컴퓨터 저장소에** 중앙 서버의 전체 사본을 가지고 작업을 하는 것을 의미한다.
- 대표적 시스템으로 Git이 있다.

Git의 필요성

- 나와 나의 동료가 동시에 같은 웹사이트에서 페이지를 업데이트하고 있다고 하자.
- 나는 무언가를 변경하고 저장한 다음 웹사이트에 그것을 업로드한다. 그런데 이때 문제는 동료가 동시에 같은 페이지에서 작업할 때이다. 확인하지 않고 동시에 작업을 한다면 누군가의 작업은 겹쳐쓰여질 것이고 지워질 것이다.
- 깃과 같은 버전관리 시스템은 그런 일을 방지해준다.
- 나와 동료는 같은 페이지에 각자의 수정사항을 각각 업로드할 수 있고, 깃은 두 개의 복사본을 저장한다.
- 그런 후 우리는 어떤 작업도 잃어버리지 않고 변경사항들을 병합(Merge)할 수 있다. 깃은 이전에 만들어진 모든 변경사항의 "스냅샷"을 저장하기 때문에 이전 시점의 어떤 버전으로 되돌릴 수도 있다.
- 이렇기에 프로젝트를 진행할 때 Git은 아주 유용하며 필수적으로 많이 사용한다. 프로젝트에서 내가 작업한 부분을 올리고 다른 사람이 작업한 부분을 올려 스냅샷이 저장되고 어떠한 코드도 손실되지 않고 나눠서 작업한 파일을 병합할 수 있기 때문이다.

Git 기본용어

- **Repository** : 저장소를 의미하며, 저장소는 히스토리, 태그, 소스의 가지치기 혹은 branch에 따라 버전을 저장한다. 저장소를 통해 작업자가 변경한 모든 히스토리를 확인 할 수 있다.
- **Working Tree** : 저장소를 어느 한 시점을 바라보는 작업자의 현재 시점이다.
- **Staging Area** : 저장소에 커밋하기 전에 커밋을 준비하는 위치이다.(index)
- **Commit** : 현재 변경된 작업 상태를 점검을 마치면 확정하고 저장소에 저장하는 작업이다.
- **Head** : 현재 작업중인 Branch를 가리키는 것이다.
- **Branch** : 가지 또는 분기점을 의미하며, 작업을 할때에 현재 상태를 복사하여 Branch에서 작업을 한 후에 완전하다 싶을때 Merge를 하여 작업을 한다.
- **Merge** : 다른 Branch의 내용을 현재 Branch로 가져와 합치는 작업을 의미한다.

Git 주요 명령어 1

- **git init** : 깃 저장소를 초기화한다. 저장소나 디렉토리 안에서 이 명령을 실행하기 전까지는 그냥 일반 폴더이다. 이것을 입력한 후에야 추가적인 깃 명령어들을 줄 수 있다.
- **git help** : 명령어를 잊어버렸다면 커맨드 라인에 "git help"를 쳐보자. 그럼 21개의 가장 많이 사용하는 깃 명령어들이 나타난다. 좀 더 자세하게 "git help init"이나 다른 용어를 타이핑하여 특정 깃 명령어를 사용하고 설정하는 법을 이해할 수도 있다.
- **git status** : 저장소 상태를 체크한다. 어떤 파일이 저장소 안에 있는지, 커밋이 필요한 변경사항이 있는지, 현재 저장소의 어떤 브랜치에서 작업하고 있는지 등을 볼 수 있다.
- **git clone** : 원격 저장소의 저장소를 내 local에서 이용할 수 있게 그대로 복사해 가져온다.
- **git add** : 이 명령이 저장소에 새 파일들을 추가하진 않는다. 대신, 깃이 파일들을 지켜보게 한다. 파일을 추가하면, 깃의 저장소 "스냅샷"에 포함된다.

Git 주요 명령어 2

- **git commit** : 깃의 가장 중요한 명령어이다. 파일을 수정한 후, 저장소의 "스냅샷"을 찍기 위해 사용하는 명령어이다. 보통 "git commit -m "Message hear." 형식으로 사용한다. -m은 명령어의 다음 부분을 메시지로 남긴다는 뜻이다.
- **git push** : 로컬 컴퓨터에서 작업하고 당신의 커밋을 깃허브에서 온라인으로도 볼 수 있기를 원한다면, 이 명령어로 깃허브에 변경사항을 "push"한다.
- **git pull** : 로컬 컴퓨터에서 작업할 때, 작업하고 있는 저장소의 최신 버전을 원하면, "git pull"을 통해 깃허브로부터 변경사항을 다운로드할 수 있다.
- **git log** : 커밋 내역을 확인해보고 싶을 때 사용하는 명령어이다.
- **git branch** : 여러 협업자와 작업하고 자신만의 변경을 원한다면 이 명령어로 새로운 브랜치를 만들고, 자신만의 변경사항과 파일 추가 등의 커밋 타임라인을 만든다. 새 브랜치를 "hello"로 지정하고 싶다면 "git branch hello"라고 쓸 수 있다.

Git 주요 명령어 3

- **git checkout** : 현재 위치하고 있지 않은 저장소를 "체크아웃"할 수 있다. 이것은 체크하길 원하는 저장소로 옮겨가게 해주는 탐색 명령이다. 만약 master 브랜치를 들여다 보고 싶으면, `git checkout master`를 사용할 수 있다.
- **git merge** : 브랜치에서 작업을 끝내고, 모든 협업자가 볼 수 있는 master 브랜치로 병합할 수 있다. "`git merge hello`"라고 입력한다면 hello브랜치에서 만든 모든 변경사항을 master로 추가한다.

Git File 상태

Committed : 데이터가 로컬 데이터베이스에 안전하게 저장됐다는 것을 의미한다.

Modified : 수정한 파일을 아직 로컬 데이터베이스에 커밋하지 않은 것을 말한다.

Staged : 현재 수정한 파일을 곧 커밋할 것이라고 표시한 상태를 의미한다.

github


- GitHub는 Git 원격 저장소 서비스입니다.
- Git과 GitHub는 별개이며 Git을 통해 프로젝트를 진행하면서 저장소를 여러 사람이 공유하고 협업할 수 있는데, 이때 협업을 위해 프로젝트를 공유할 수 있는 네트워크상의 저장공간이 필요하고 이러한 네트워크상의(인터넷) 저장공간을 제공해 주는 서비스 중 하나가 GitHub인 것입니다.
- 원격 저장소는 인터넷이 아니더라도 사내 사설 네트워크 안에 Git 원격 저장소 서버를 두고 관리할 수도 있으며, GitHub 이외에도 다양한 인터넷 Git 원격 저장소 서비스들이 존재합니다.

기본명령어

- 화면 초기화 : Ctrl + L
- 한 행의 처음과 끝 : Ctrl + A, Ctrl + E
- 목록 보기 : ls 또는 dir
- 파일의 내용 보기 : cat
- 특정 문자를 검색 : grep
- 디렉터리로 이동 : cd
- 디렉터리 생성 : mkdir
- 파일 삭제 : rm
- 파일 생성 : touch

git hub 로그인

<https://github.com/login>



Sign in to GitHub

Username or email address

Password [Forgot password?](#)

[Sign in](#)

New to GitHub? [Create an account.](#)

인증



Device verification

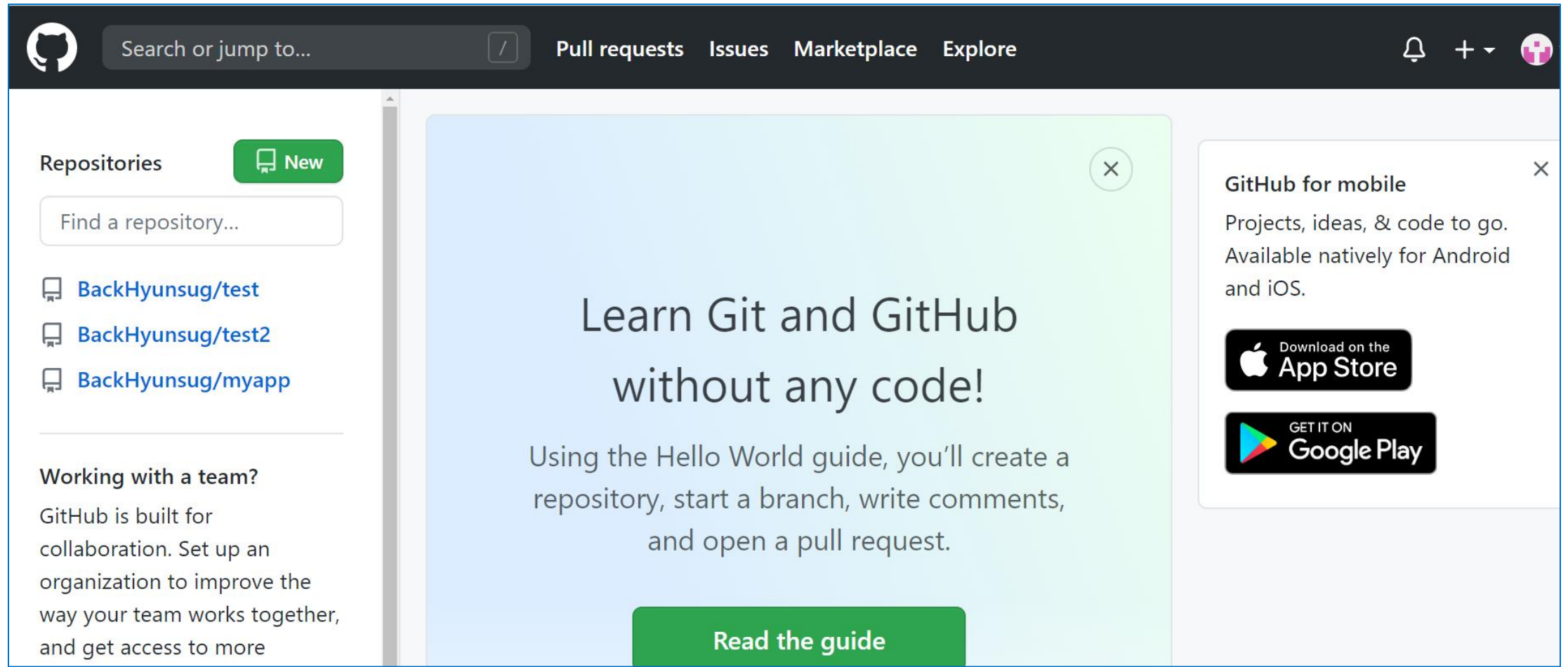
Device verification code

6-digit code

Verify

✉ We just sent your authentication code via email to l*****@hanmail.net. The code will expire at 12:49AM KST.




git hub tutorial



The screenshot shows the GitHub homepage with a dark header bar. The header contains the GitHub logo, a search bar with the text "Search or jump to...", and navigation links for "Pull requests", "Issues", "Marketplace", and "Explore". On the right side of the header are icons for notifications, a dropdown menu, and a user profile. The main content area features a large, light blue and green tutorial overlay in the center. To the left of the overlay is a sidebar with the "Repositories" section, a "New" button, a search bar, and a list of repositories. Below the repositories is a section titled "Working with a team?". To the right of the overlay is a sidebar with a "GitHub for mobile" section, including text about the app and buttons to download it from the App Store and Google Play.

Repositories New

Find a repository...

-  [BackHyunsug/test](#)
-  [BackHyunsug/test2](#)
-  [BackHyunsug/myapp](#)

Working with a team?

GitHub is built for collaboration. Set up an organization to improve the way your team works together, and get access to more


Learn Git and GitHub without any code!


Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#)

GitHub for mobile

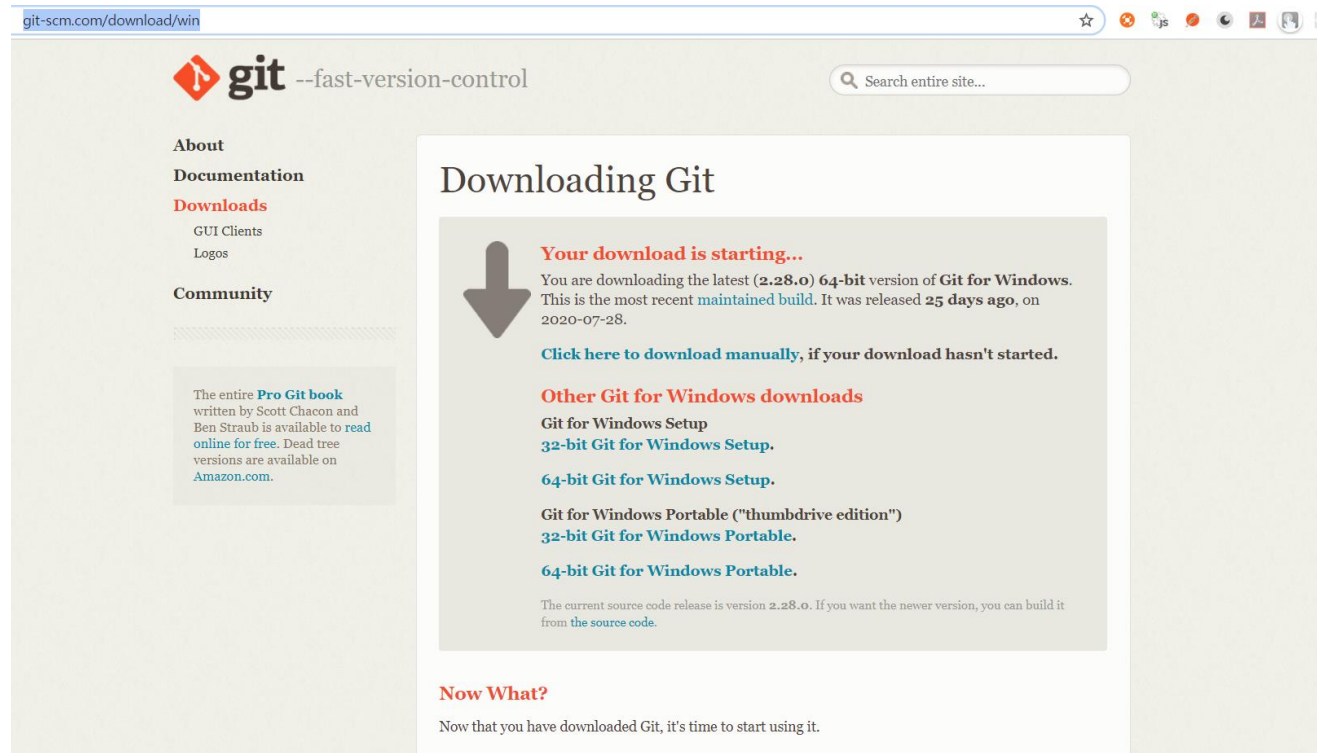
Projects, ideas, & code to go. Available natively for Android and iOS.

 Download on the **App Store**

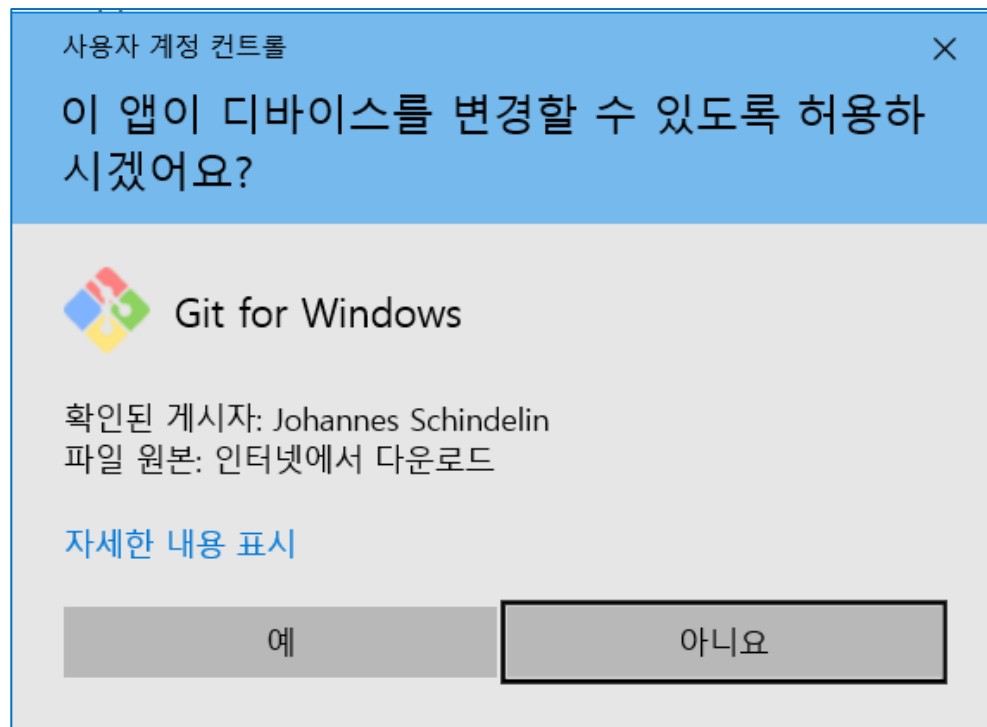
 GET IT ON **Google Play**

git 설치

- <https://git-scm.com/>
- <https://git-scm.com/download/win>



설치하기



git 환경 설정

- Git을 설치하고 나면 Git의 사용 환경을 적절하게 설정해 주어야 한다.
- 환경 설정은 한 컴퓨터에서 한 번만 하면 된다.
- 설정한 내용은 Git을 업그레이드해도 유지된다. 언제든지 다시 바꿀 수 있는 명령어도 있다.
- git config
- **C:\Users\Wuser\W.gitconfig** 파일이 생성된다. (환경설정)

git 환경 설정

- `git config --global user.name "your_name"` // git commit에 사용될 email
- `git config --global user.email "your_email@example.com"`
- `git config --list`
- Git을 설치하고 나서 가장 먼저 해야 하는 것은 사용자이름과 이메일 주소를 설정하는 것이다.
- Git은 커밋할 때마다 이 정보를 사용한다. 한 번 커밋한 후에는 정보를 변경할 수 없다.
- 각 프로젝트마다 다른 정보를 사용하고 싶을 경우에 global 옵션을 생략한다

Git 환경설정

- `git config user.name`
- `Git config user.email`

git 적용 순서

1. Git 설치

2. Git 적용하기

1. `git status`

2. `.gitignore`

3. `git add`

4. `git commit`

5. `git diff`

6. `git restore`

7. `git commit -a`

8. `git log`

Git 저장소 만들기

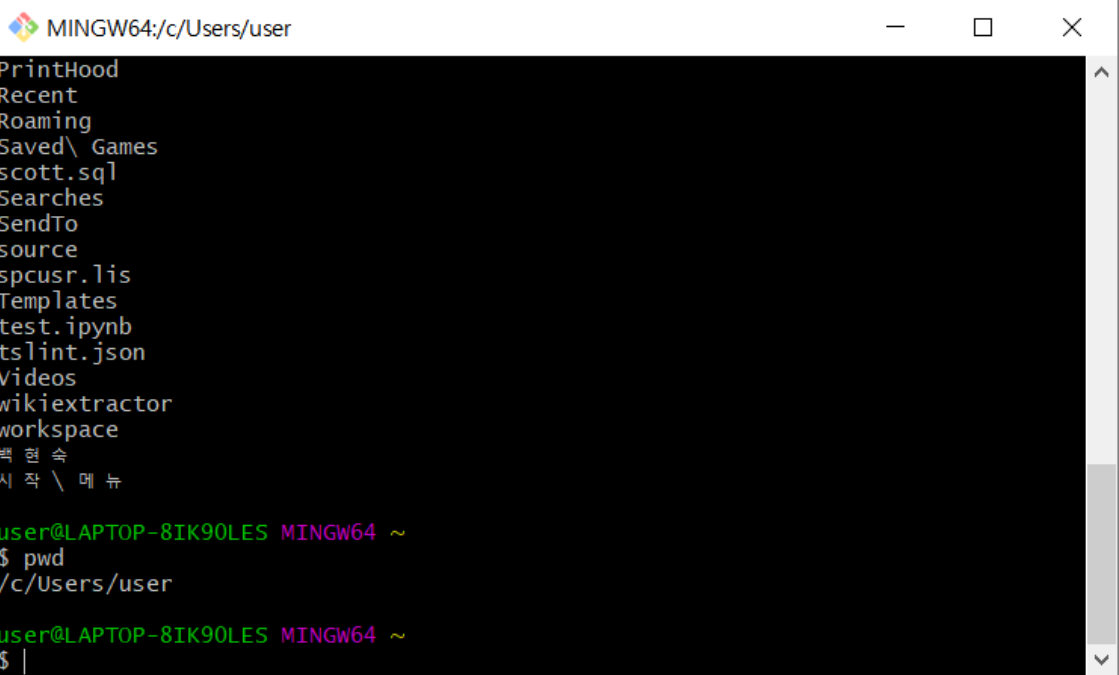
- **c:/test** 폴더를 저장소로 만들어 보자
- **cmd**
- **cd W**
- **mkdir test**
- **cd test**
- **git init** -- 폴더 초기화, git 저장소 설정
- **dir /a** 숨김폴더까지

```
C:\test 디렉터리
2020-08-22 오후 04:04 <DIR> .
2020-08-22 오후 04:04 <DIR> ..
2020-08-22 오후 04:04 <DIR> .git
                0개 파일                0 바이트
                3개 디렉터리 20,642,742,272 바이트 남음

C:\test>
```

git 저장소 만들기(linux모양)

- gitbash 실행(gui임)
- pwd 명령어(현재 위치확인명령)
- c:/users/user 아래가 기본폴더임
- cd /c/users/user
- cd /c/test
- touch test.txt (파일만들기)
- Ls



```
MINGW64:/c/Users/user
PrintHood
Recent
Roaming
Saved\ Games
scott.sql
Searches
SendTo
source
spcusr.lis
Templates
test.ipynb
tslint.json
Videos
wikiextractor
workspace
백현숙
시작 \ 메뉴

user@LAPTOP-8IK90LES MINGW64 ~
$ pwd
/c/Users/user

user@LAPTOP-8IK90LES MINGW64 ~
$ |
```


test.txt파일 추가하기

c:/test/test.txt 파일을 추가합시다

c:/test>git status

```
C:\test>git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
C:\test>
```

git add 저장소에 파일 추가

- `git add --all .`

```
nothing added to commit but untracked files present (use "git add" to track)
C:\test>git add --all .
C:\test>git status
On branch master

No commits yet

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

git commit

- **git commit -m "test"**

```
C:\#test>git commit -m "test"
[master (root-commit) 210c3f2] test
1 file changed, 1 insertion(+)
create mode 100644 test.txt

C:\#test>git status
On branch master
nothing to commit, working tree clean
```

Commit 전의 내용 취소하기

- 커밋전의 삭제
- `Git rm -cached` 파일명
- **첫커밋후의 삭제**
- `git reset HEAD` : Index에 반영된 모든 내용이 취소
- `Git reset HEAD [files]`

파일의 내용을 변경 후 저장하자

- 파일의 내용을 메모장에서 열어서 변경후 저장하자
- **Git status**
- `git add --all .`
- **`git commit -m "test"`**
-

git hub

- 깃(Git)을 이용하여 로컬 저장소를 생성하고 소스코드를 관리할 수 있게 되었다. 하지만 개인 프로젝트가 아니라면 여러명이서 함께 프로젝트를 진행해야 하므로 원격 저장소는 반드시 필요하다. 또 개인 프로젝트라고 하더라도 소스코드의 유실을 방지하기 위해서 원격 저장소는 필수이다.
- 깃허브는 Git을 지원하는 마이크로소프트(Microsoft)사에서 운영하는 호스팅 서비스이다. 현재 4,000만명 이상의 사용자와 4,400만 개 이상의 신규 저장소가 있다고 하니 그 인기가 어마어마하다. Github는 오픈소스 지원정책에 의해 무료로 사용하더라도 공개방식을 취하면 사용상에 전혀 제약이 없다.

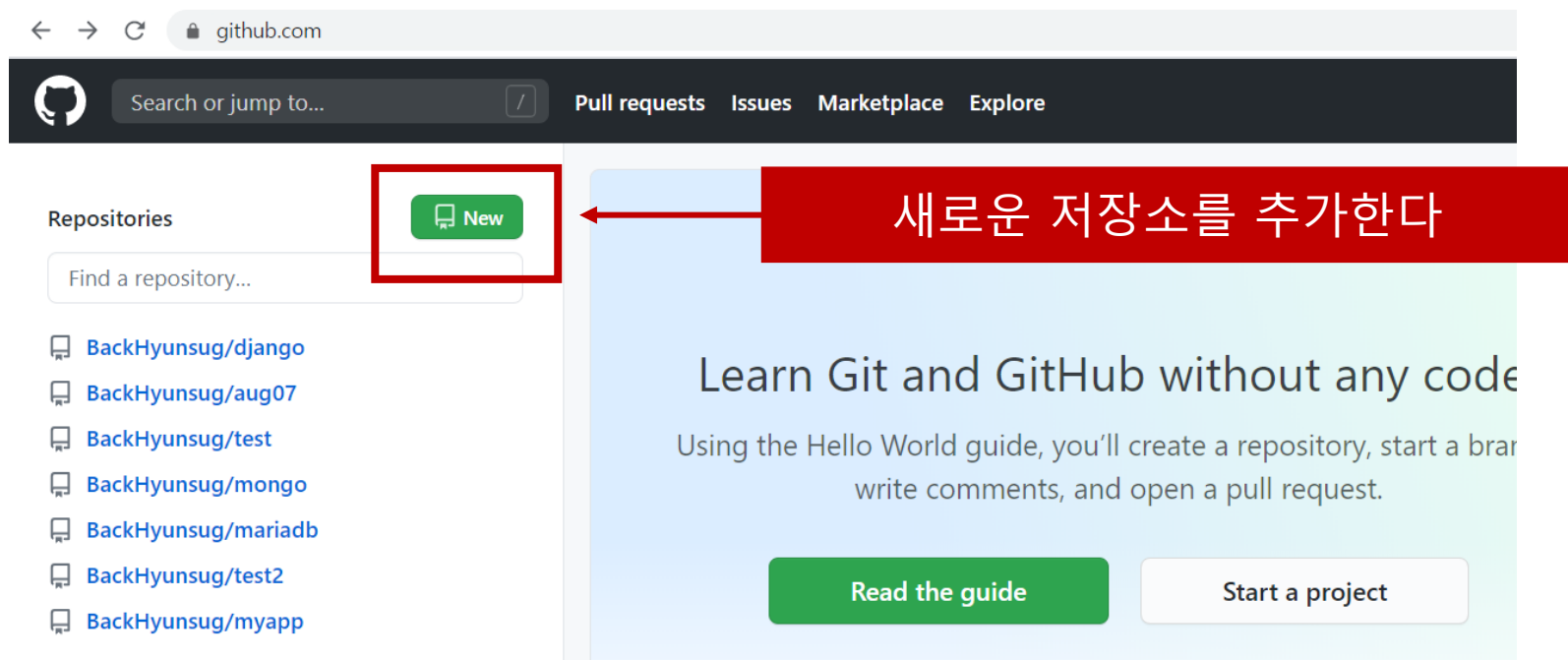
- **github 가입**

<https://github.com/>

- **github 저장소 생성**
- **github 연결**

github 저장소 연결

- Github에 로그인한 후에 Repositories의 "New"버튼을 클릭한다.



github 저장소 연결

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner *



BackHyunsug ▾

Repository name *

/ mysite



Great repository names are short and memorable. Need inspiration? How about **super-duper-goggles?**

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.

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

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

github 저장소 연결

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



Set up in Desktop

or

HTTPS

SSH

`https://github.com/BackHyunsug/mysite.git`

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

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

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

github 저장소 연결

- <https://github.com/BackHyunsug/mysite>
- `echo "# mysite" >> README.md`
- `git init`
- `git add README.md`
- `git commit -m "first commit"`
- `git remote add origin https://github.com/BackHyunsug/mysite.git` 원격저장소와 내 로컬 저장소 연결하기
- `git branch -m master`
- `git push -u origin master` (master라는 branch가 자동으로 만들어졌었음, 안되면 직접만들자) 새버전은 main으로 바뀌었음

github 저장소 연결

- ...or push an existing repository from the command line
- `git remote add origin https://github.com/BackHyunsug/mysite.git`
- `git push -u origin master`

장고사이트

- C:\project\mysite>git init ← 로컬 저장소 만들기
- .gitignore 파일작성

```
.idea  
db.sqlite3  
*.pyc  
__pycache__
```

장고사이트

- **git add -all .**
- `git config --global user.email "littleconan@hanmail.net"`
- `git config --global user.name "Backhyunsug"`
- `git commit -m " 장고 프로젝트 최초 커밋"` 커밋해야 push 가능함
- `git remote add origin https://github.com/BackHyunsug/mysite.git`
- `git push -u origin master (littleconan@hanmail.net)`

error 시

- 새로운 branch를 만들자(git 폴더 삭제 후 다시하기)
- git init
- git add .
- git commit -m "first"
- git branch -M main
- git remote add origin https://github.com/BackHyunsug/myproject1.git
- git push -u origin main 오류발생시 **git push -u origin +main**

수정 후

- `git add .`
- `git commit -m "새로운문구"`
- `git push -u origin 브랜치명`

복원하기

- `git log`
- `git log --oneline`

선택 명령 프롬프트

```
C:\hello>git log --oneline
41a49dd (HEAD -> hello, origin/hello) test
43adf52 first
C:\hello>
```

- `git revert 43adf52`

브랜치 이동하기

- `git branch`
- `git checkout hello`

친구초청

The screenshot displays the GitHub repository settings interface. At the top, a navigation bar includes links for Code, Issues (0), Pull requests (0), Actions, Projects (0), Security (0), Insights, and Settings. The Settings menu is highlighted with a red box. On the left sidebar, the 'Manage access' option is also highlighted with a red box. The main content area is titled 'Who has access' and contains two panels: 'PRIVATE REPOSITORY' (locked icon) and 'DIRECT ACCESS' (people icon). The 'DIRECT ACCESS' panel shows '0 collaborators have access to this repository. Only you can contribute to this repository.' Below this, the 'Manage access' section features a lock icon and the text 'You haven't invited any collaborators yet'. Underneath, the Korean text '유저 초대하기' (Invite user) is followed by a green button labeled 'Invite a collaborator', which is highlighted with a red box. Red arrows indicate the navigation path from the top Settings menu to the left sidebar, and then to the 'Invite a collaborator' button.

<> Code ! Issues 0 Pull requests 0 Actions Projects 0 Security 0 Insights **Settings**

Options
Manage access
Webhooks
Notifications
Integrations
Deploy keys
Secrets
Actions

Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

[Manage](#)

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet

유저 초대하기 **Invite a collaborator**

친구초청 절차

1. 로그인 후 저장소 페이지로 갑니다.
2. Settings > Collaborators > Manage access를 선택.
3. invite a collaborator 클릭 후 초대할 사람의 Username을 빈칸에 입력
4. 검색 결과에서 초대할 사람을 선택
5. Select a collaborator above를 누른 후 수락을 기다립니다.

초청받은쪽에서 당기기

- `git clone https://github.com/BackHyunsug/myproject1.git`
- 안되면 토큰 받기
- 수정완료후
- `git add .`
- `git commit -m "test"`
- `git push -u`

참고자료

- <https://www.lainyzine.com/ko/article/how-to-cancle-git-add/>
- <https://git-scm.com/book/ko/v2/%EC%8B%9C%EC%9E%91%ED%95%98%EA%B8%B0-%EB%B2%84%EC%A0%84-%EA%B4%80%EB%A6%AC%EB%9E%80%3F>
- <http://www.devpools.kr/2017/02/05/%EC%B4%88%EB%B3%B4%EC%9A%A9-git-%EB%90%98%EB%8F%8C%EB%A6%AC%EA%B8%B0-reset-revert/>
- [**https://www.lainyzine.com/ko/article/git-revert-reverting-commit-in-git-repository/**](https://www.lainyzine.com/ko/article/git-revert-reverting-commit-in-git-repository/)