

버전관리 도구 Git



버전관리 도구 - Git

- 0 버전관리
- 1 GitHub
- 2 Local Git 설치
- 3 Remote 저장소 연동하기

0. 버전관리

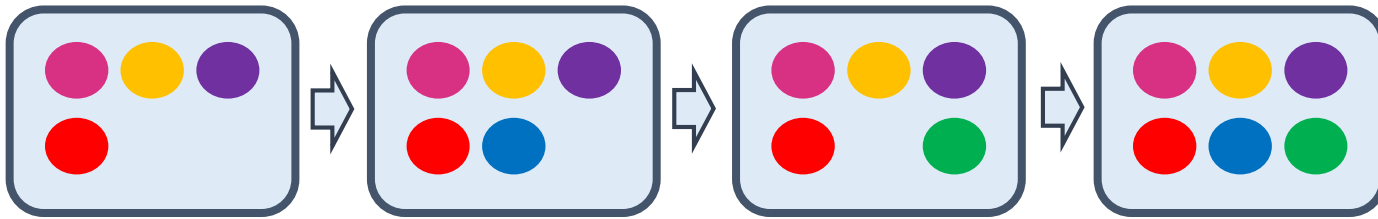
버전관리

• 버전 관리

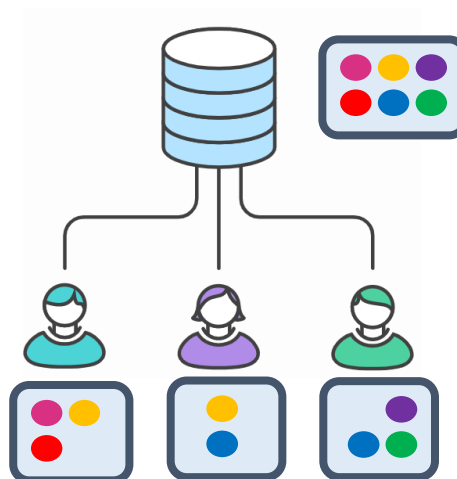
- ✓ 소프트웨어 개발시 매 버전마다 형상을 관리하는 것

• 버전관리의 필요성

- ✓ 코드의 변경사항의 추적과 관리



- ✓ 팀원들 간의 효율적인 협업
 - 내가 고친거 누가 덮어 썼어?
 - 길동씨. 어제 고친거 메일로 보내줘요



과제제출_최종
과제제출_최종_1
과제제출_최종_2
과제제출_최종_3
과제제출_최종_진짜최종
과제제출_최종_진짜최종_마지막

처음꺼가 좋았던 것 같아요.
다시 처음으로 갑시다!



누구인가?
누가 버그를 심었는가?



버전관리

- 여러 개발자가 함께 개발시 source code 수정, 추가, 삭제 관리
 - ✓ 서로 같은 code를 고치면 문제 발생
 - ✓ 버그가 발생시 어느 버전에서 버그 삽입이 되었는지 확인 및 그 이전버전으로 변경 필요
 - ✓ 누가 어떤 bug를 만들어 냈는지 확인 및 수정 가능
- 여러 버전 관리 도구
 - ✓ CVS, Subversion, Git 등

Q. 여러 명이 함께 작업할 때만 필요한 것 아닌가요? 혼자 할 때 굳이 필요해요?

➔ A. 스스로의 체계를 갖추면 필요 없음. 효율적인 관리를 위한 것임.

작업하는 곳이 한곳 이상이라면 필요 (회사 컴퓨터, 집 컴퓨터)

소프트웨어 규모가 커질수록 체계적인 버전 관리는 필수



Git 기술을 활용한 github

- **Git Cloud 서비스**

- ✓ Code 저장소를 github에서 제공 (무료, 웹 기반)
- ✓ Git 기술이 구현되어 있어 버전관리 용이
- ✓ <https://github.com/>

<Github을 활용한 예>

- **Opensource 프로젝트**

- ✓ 전세계 여러 개발자와 함께 코드 작성

- **여러 사람과 협업 프로젝트**

- ✓ 권한을 가진 사람들만이 code를 공유하고, 수정할 수 있다.

- **개인 프로젝트 외부 공유**

- ✓ 본인 작업의 포트폴리오로 활용
- ➔ 지금부터 꾸준히 외부 저장소 github에 저장 하기



Github 활용

- 목표 – 학생 개인별 Github 운영 –웹 포트폴리오

- ✓ 효과

- 학생의 포트폴리오 관리 (코드 및 홈페이지)
- Git 활용 역량 입증
- 작업 history 관리
- 개발자로서의 첫걸음, 취업시 유리

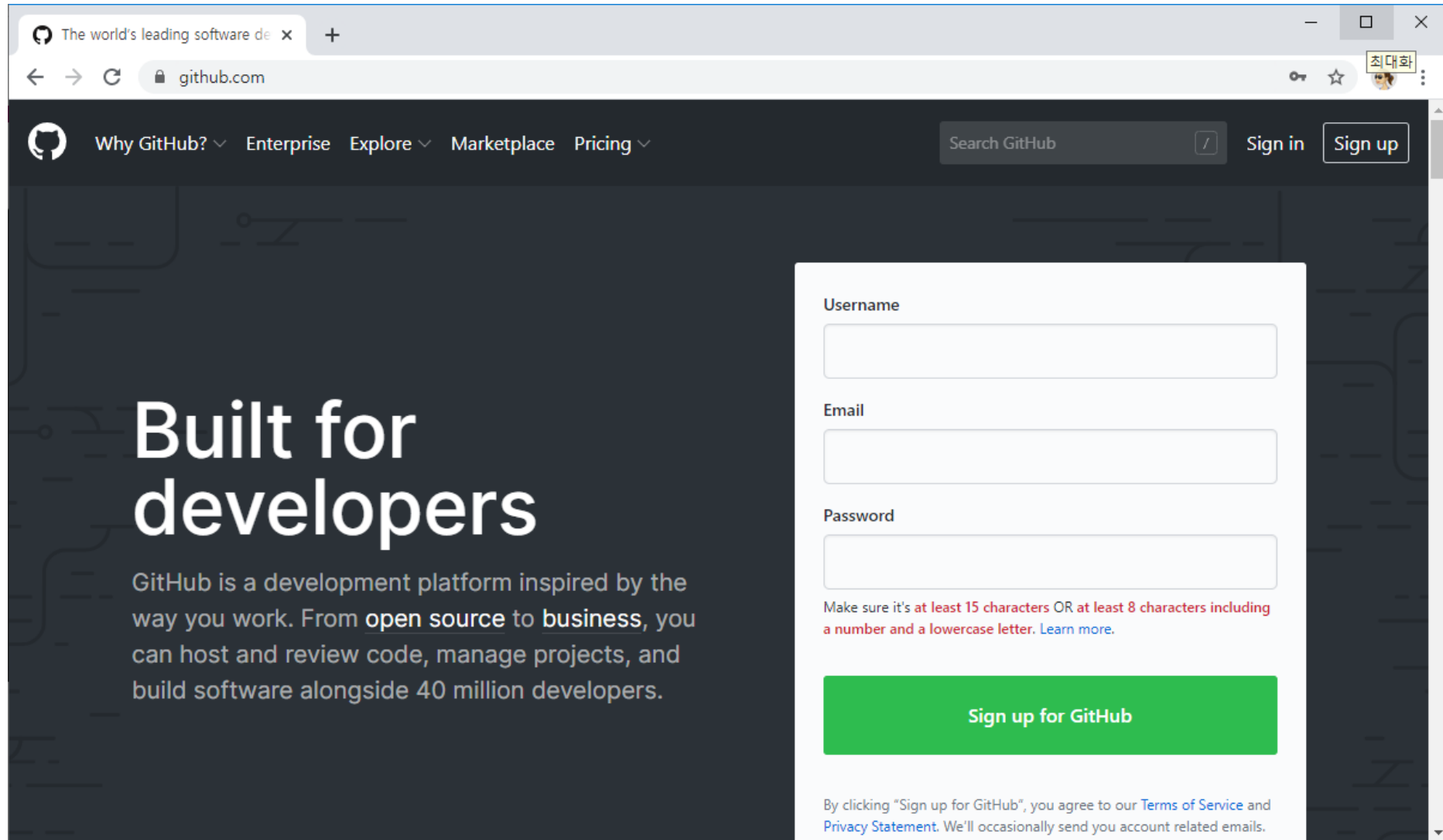
- ✓ 본인이 직접 관리

목표

- Github 가입 및 기초 사용
- Local에서 Git console 사용
- Local – remote (github) 연결
 - ✓ Console로 연동
 - ✓ pyCharm 설치 및 helloworld
 - ✓ Github와 pyCharm을 연동

I. GitHub

I.1 Github 가입하기



The screenshot shows the GitHub homepage in a web browser. The browser's address bar displays 'github.com'. The page header includes the GitHub logo, navigation links for 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing', a search bar, and 'Sign in' and 'Sign up' buttons. The main content area features the headline 'Built for developers' and a paragraph describing GitHub as a development platform. A white sign-up form is overlaid on the right side of the page, containing input fields for 'Username', 'Email', and 'Password', along with a green 'Sign up for GitHub' button. Below the button, there is a disclaimer about the terms of service and privacy statement.

The world's leading software de x +

github.com

Why GitHub? Enterprise Explore Marketplace Pricing

Search GitHub / Sign in Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 40 million developers.

Username

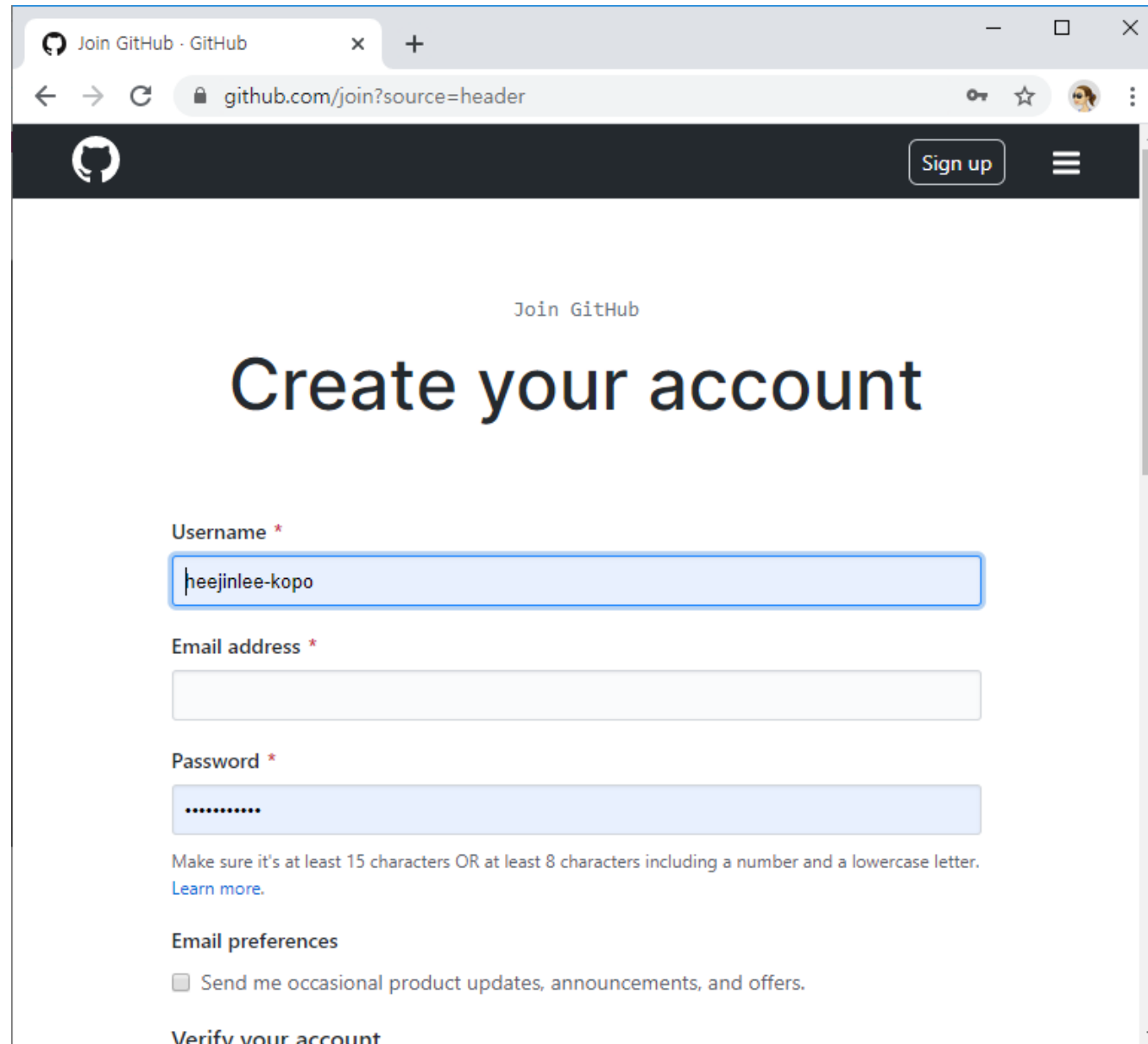
Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.



The screenshot shows a web browser window with the GitHub sign-up page. The browser's address bar shows the URL `github.com/join?source=header`. The page has a dark header with the GitHub logo and a 'Sign up' button. The main content area is white and features the heading 'Join GitHub' followed by 'Create your account'. Below this, there are three input fields: 'Username' with the value 'heejinlee-kopo', 'Email address', and 'Password' (masked with dots). A note below the password field states: 'Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)'. At the bottom, there is an 'Email preferences' section with a checkbox labeled 'Send me occasional product updates, announcements, and offers.' and a link to 'Verify your account'.

Join GitHub · GitHub

github.com/join?source=header

Sign up

Join GitHub

Create your account

Username *

heejinlee-kopo

Email address *

Password *

.....

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.
[Learn more.](#)



Email preferences

☐ Send me occasional product updates, announcements, and offers.

[Verify your account](#)

Choose a plan

- Individual – free 선택

Individual Pick the plan that's right for you, personally.	Team Choose a plan to help your team grow and collaborate.
 Free \$0 USD	 Pro \$7 USD Per month
The basics of GitHub for every developer	Pro tools for developers with advanced requirements
Choose Free	Choose Pro

Create a new repository

- Repository “hello-world”

Create a new repository

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

Owner



Repository name *

hello-world



Great repository names are short and memorable. Need inspiration? How about [cuddly-funicular?](#)

Description (optional)

test repository



☒ Public

Anyone 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

Learn Git and GitHub without any code!

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

[Read the guide](#)

hjkopo / **hello-world** Unwatch 1 Star 0 Fork 0

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

test repository [Manage topics](#) [Edit](#)

1 commit 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file [Clone or download](#)

hjkopo Initial commit Latest commit ee74717 16 seconds ago

README.md Initial commit 16 seconds ago

README.md

testrepository

test repository

<https://guides.github.com/activities/hello-world/>

Create a new branch

- Repository에 가서, new branch를 생성

- ✓ Branch "master" – 전체 프로젝트의 완성된 기능을 가지는 branch
- ✓ Branch "feature1" – 하나의 기능인 feature1을 구현하는 branch
- ✓ Branch는 기능별로 상황별로 만들 수 있다.

Just another repository — Edit

1 commit

1 branch



branch: master

hello-world / +

Initial commit



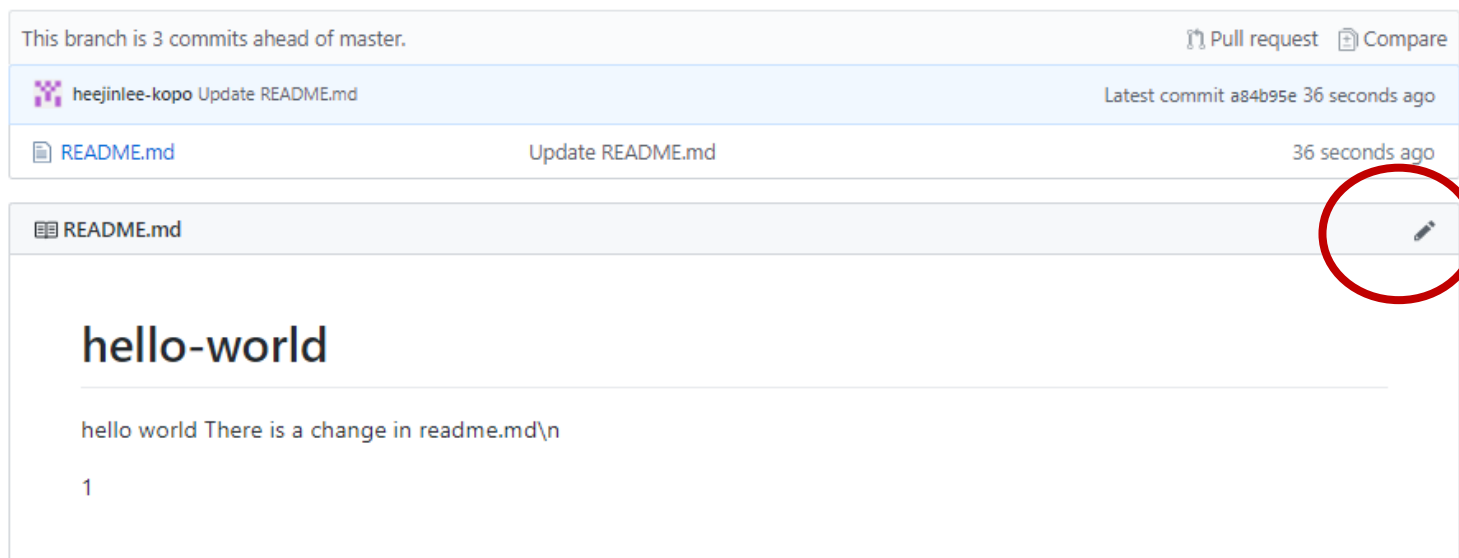
hubot authored just now

README.md

Initial

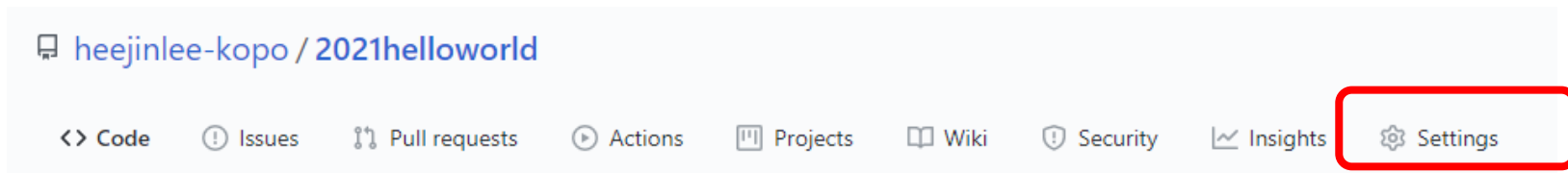
README.md

- Commit – 저장소에 수정사항을 저장하는 것
- Commit 방법
 - ✓ Git client에서 하는 방법(사용자 pc)
 - ✓ 약식으로 해보자
- Branch “master”에서 commit 실습
 - ✓ Readme.md 수정
 - ✓ Commit
 - ✓ master branch에서만 Commit log 확인
 - ✓ 다른 branch에는 반영이 안되어 있음.



- Repository별로 홈페이지를 생성

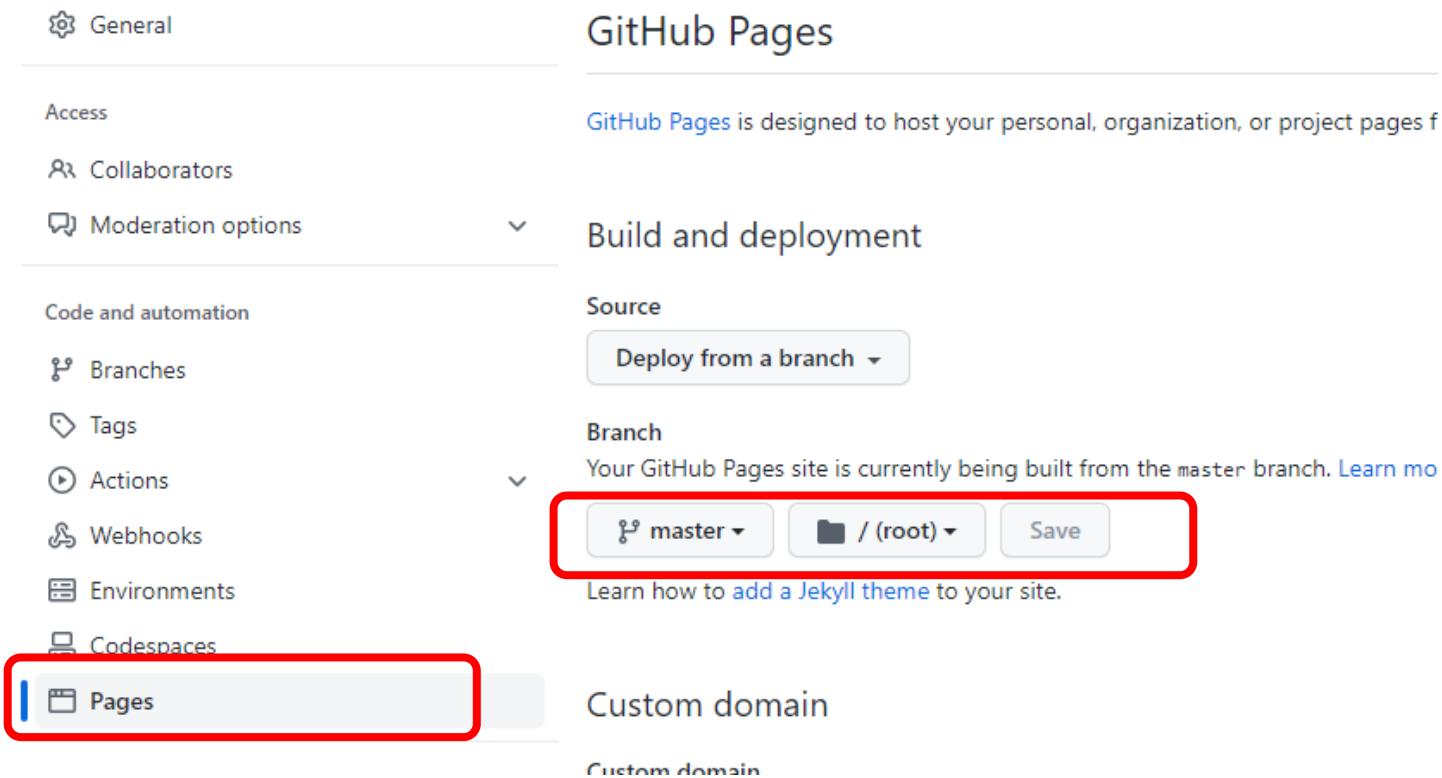
- ✓ `https://{github-id}.github.io/{repository-name}` 에서 확인
- ✓ Repository에서 Settings 선택



- ✓ 하단의 GitHub Pages

- Source

- Homepage로 사용할 branch 선택
 - 원하는 branch 선택
 - master



홈페이지 만들기 (2/3)

Github

- 홈페이지 주소를 간단히 해보자.

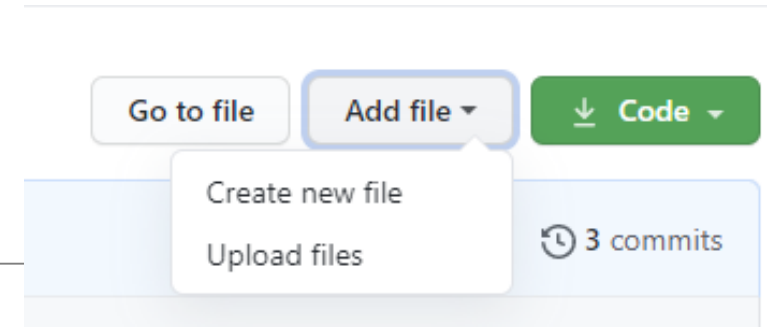
- ✓ <https://{github-id}.github.io>
- ✓ <https://{github-id}.github.io/{repository-name}>

- Repository “{github-id}.github.io” 생성

- ✓ Upload files – “[architecture.png](#)”, “[project.pptx](#)”
- ✓ Readme.md Commit

```
# 홍길동의 github 홈페이지
## IT 어린이입니다.
파이썬 프로그래밍 프로젝트입니다.
## 전체 설계도
 <br>
[발표자료](/project.pptx)<br>
## 발표 동영상
발표 동영상입니다.
<iframe id="ytplayer" type="text/html" width="640"
height="360" src="https://www.youtube.com/embed/jsRqtnGOavk" frameborder="0"> </iframe>
## Reference
[검색엔진](https://naver.com)
[Repository 1](https://{github-id}.github.io/{repository-name})
```

 [dongmisw](#) / [dongmisw.github.io](#)



홈페이지 만들기 (3/3)

- 더 멋진 남의 포트폴리오 가져오자
- Github portfolio 검색
- Repository fork
- 수정
- 마크다운언어 - naver에 검색

Ⅱ. local Git 설치

Ⅱ.1 Git 설치

Ⅱ.2 Git 기본기능

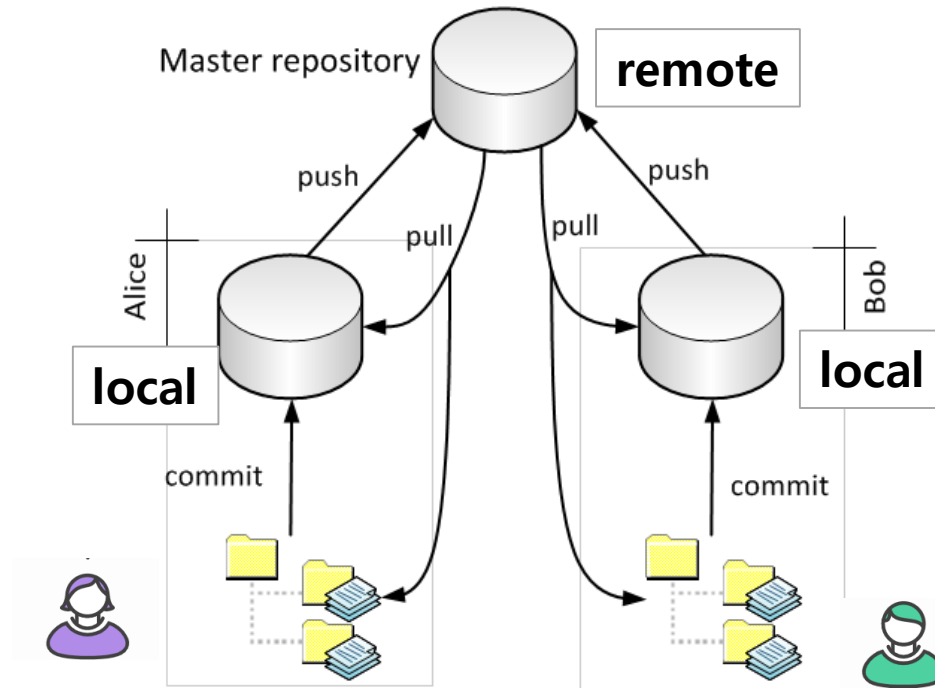
Ⅱ.3 Branch

- 내가 작업하는 컴퓨터에 설치 (local)

- ✓ Console 사용 (설치 필요)
- ✓ GUI console 사용 (설치 필요)
- ✓ IDE를 사용

- 공용 저장소에 설치 (remote)

- ✓ On premise – 서버실에 있는 서버에 설치 (private한 용도, 보안 중요한 경우)
- ✓ Cloud – 외부의 저장소를 두고 제공하고 있는 서비스를 사용(SaaS) *



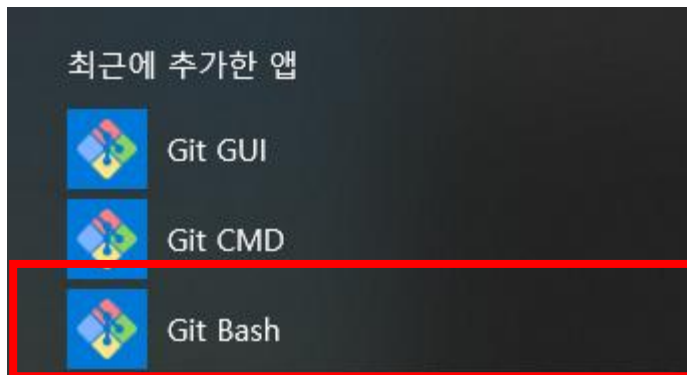
1. Git 설치

- <https://git-scm.com/>
- 다운로드 및 설치 - 각자 os에 맞는 설치파일 설치
- Windows - 2.26.0 - 64bit

2. Git-bash.exe 실행 (windows 기준)

- C:\Program Files\Git\Git-bash.exe

3. Console에서 사용



<기본 linux 명령어>

- 디렉토리 생성

```
>> mkdir MyGitRepository
```

- 하위 디렉토리 MyGitRepository 들어가기 (상위 디렉토리로 가기)

```
>> cd MyGitRepository  
>> cd ..
```

- 현재 디렉토리 확인

```
>> pwd
```

- 디렉토리 내용 확인 - 윈도우의 경우

```
>> dir
```

- vi editor로 들어갔을 때 빠져나오기

```
:q! (enter)
```

• Git 저장소로 만들기

- ✓ MyGitRepository 디렉토리 안에 관리하고자 하는 code들을 넣고 git으로 관리
- ✓ .git이라는 숨김 파일이 생김.

```
>> git init
```

이름

 .git

```
USER@Heejin MINGW64 /c/workspace/hello-world
$ git init
Initialized empty Git repository in C:/workspace/hello-world/.git/

USER@Heejin MINGW64 /c/workspace/hello-world (master)
$ |
```

- 저장소 상태를 확인

```
>> git status
```

- Git이 변경사항을 반영할 파일을 지정

- ✓ git add를 하지 않으면 commit시에 git에 반영 안됨. 매번 지정해줘야함

```
>> git add 파일명
>> git add .           //해당 directory안의 모든 파일을 지정
예) git add a.txt
```

- 변경사항 최종 저장

- ✓ Add한 파일을 commit

```
>> git commit -m '설명'
예) git commit -m [feature]로그인 기능 추가
```

Git 기본 사용 방법 (4/6)

- 기본 사용자 setting

```
>> git config --global user.name "홍길동 "  
>> git config --global user.email "you@example.com"
```

```
USER@Heejin MINGW64 /c/workspace/hello-world (master)  
$ git commit -m "a.txt add first commit"  
  
*** Please tell me who you are.  
  
Run  
  
    git config --global user.email "you@example.com"  
    git config --global user.name "Your Name"  
  
to set your account's default identity.  
Omit --global to set the identity only in this repository.  
  
fatal: unable to auto-detect email address (got 'USER@Heejin.(none)')
```

- 변경사항 파악

- ✓ Commit전에 마지막 버전 대비 변경된 내용을 최종 확인 기능

```
>> git diff
```

```
이희진 @DESKTOP-A9DQD3M MINGW64 /d/workspace/gittest (master)
$ git diff
warning: LF will be replaced by CRLF in index.html.
The file will have its original line endings in your working directory
diff --git a/index.html b/index.html
index 1f92ce1..36746bc 100644
--- a/index.html
+++ b/index.html
@@ -3,6 +3,7 @@
     <body>
         Hello Git!
         Hello Again
+        New sentence added
     </body>
 </head>
```


Git 기본 사용 방법 (6/6)

• 전체 변경 이력 조회

```
>> git log           //변경 이력 조회
>> git log -p        //전체 변경 이력 조회
```

```
MINGW64:/d/workspace/gittest
1 file changed, 1 insertion(+)

이희진@DESKTOP-A9DQD3M MINGW64 /d/workspace/gittest (master)
$ git log
commit 8a937492b0d6c0363abf35f414d5a949562f966 (HEAD -> master)
Author: 이희진 <heejinlee@kopo.ac.kr>
Date: Thu Feb 27 12:54:21 2020 +0900

    3rd

commit b5286db33cf0f40e5f79e5a3f95ad8a02676a96c
Author: 이희진 <heejinlee@kopo.ac.kr>
Date: Thu Feb 27 12:43:59 2020 +0900

    2nd description

commit 2ee0b80ef5ee7df1fa2c1d684953ea378b65c48b
Author: 이희진 <heejinlee@kopo.ac.kr>
Date: Thu Feb 27 12:41:54 2020 +0900

    git test 첫 번째 commit 설명

이희진@DESKTOP-A9DQD3M MINGW64 /d/workspace/gittest (master)
$ git |
```

```
MINGW64:/d/workspace/gittest

Hello Again
New sentence added
</body>
</head>

commit b5286db33cf0f40e5f79e5a3f95ad8a02676a96c
Author: 이희진 <heejinlee@kopo.ac.kr>
Date: Thu Feb 27 12:43:59 2020 +0900

    2nd description

diff --git a/index.html b/index.html
index 7cd50d9..1f92ce1 100644
--- a/index.html
+++ b/index.html
@@ -2,6 +2,7 @@

    <body>
        Hello Git!
        Hello Again
    </body>
</head>

commit 2ee0b80ef5ee7df1fa2c1d684953ea378b65c48b
Author: 이희진 <heejinlee@kopo.ac.kr>
Date: Thu Feb 27 12:41:54 2020 +0900

    git test 첫 번째 commit 설명

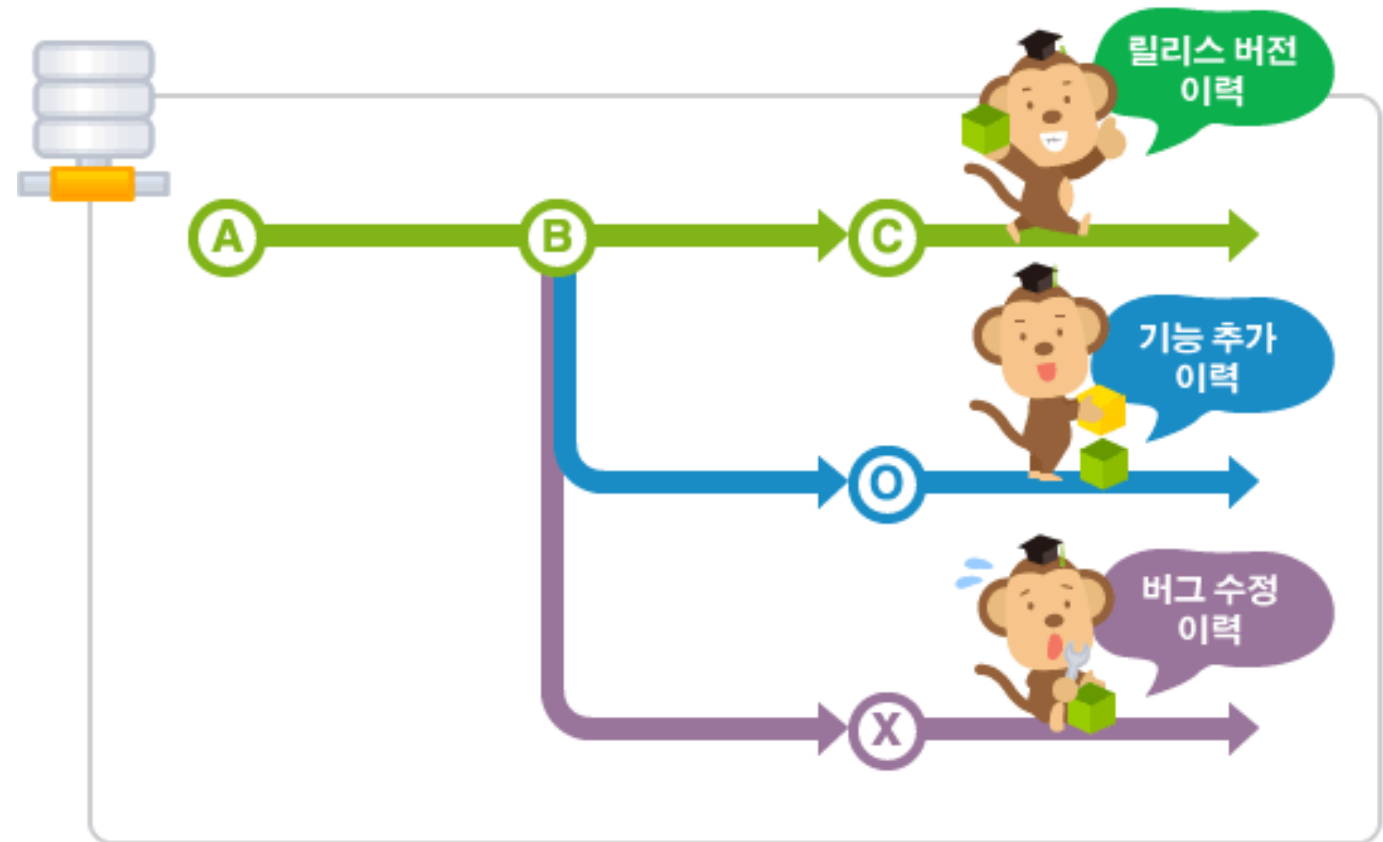
diff --git a/index.html b/index.html
new file mode 100644
index 0000000..7cd50d9
--- /dev/null
+++ b/index.html
@@ -0,0 +1,7 @@
+<head>
+
+    <body>
+        Hello Git!
+    </body>
+</head>
+
(END)
```

• Branch 정의

- ✓ 독립적인 작업을 진행하기 위한 다른 장소
- ✓ 서로 영향을 받지 않고, 여러 작업이 동시에 진행 가능
- ✓ 다른 브랜치와 병합(Merge)하여, 하나의 브랜치로 모을 수 있다.

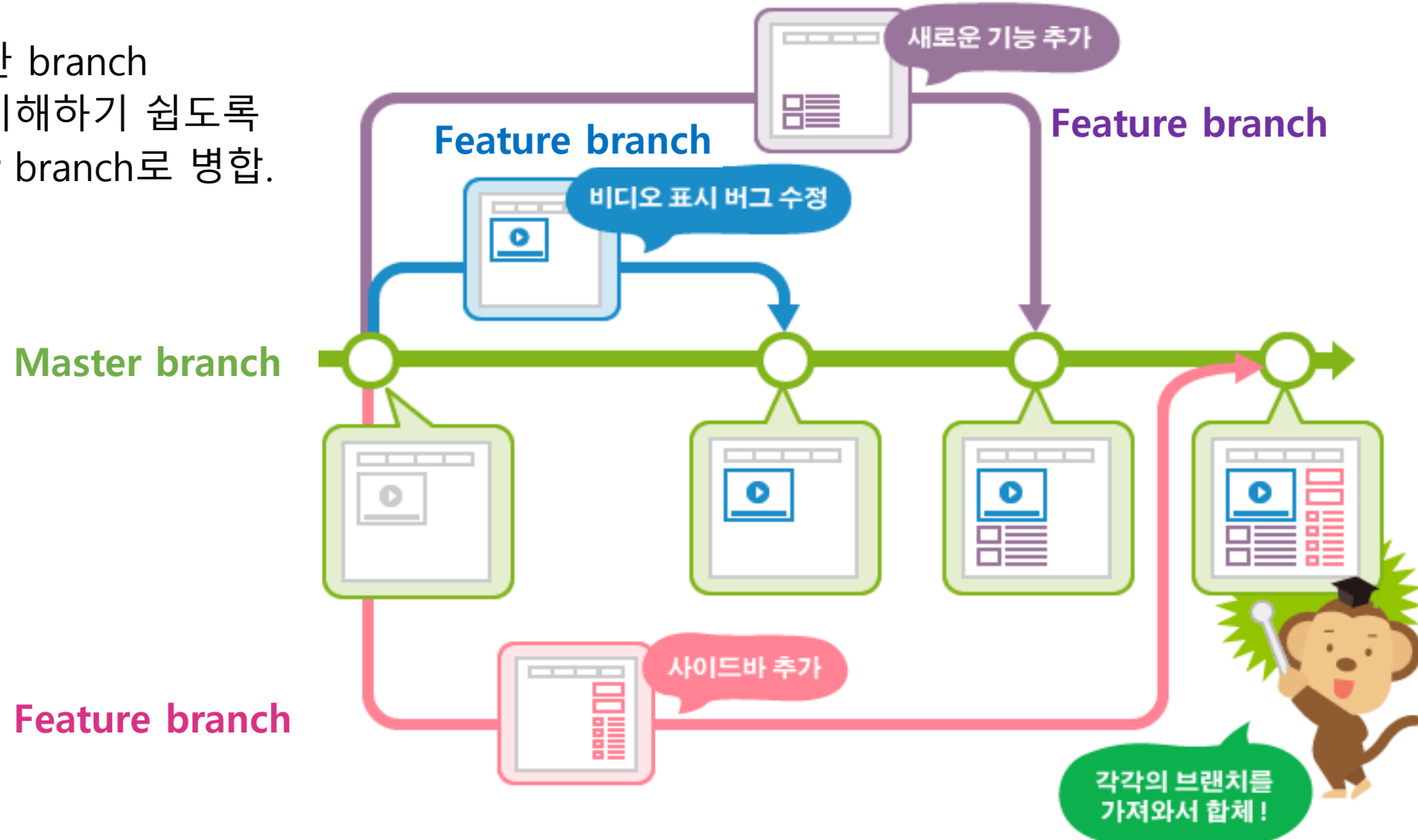
• Branch 필요한 경우 예시

- ✓ 안정적인 버전 외에 새로운 기능을 추가시
- ✓ 개발과 운영을 따로 할 때
- ✓ 고객사 별로 새로운 기능을 제공해야 할 때



• 기본 규칙

- ✓ Master branch - 안정화 branch
- ✓ Feature branch
 - 새로운 기능 구현을 위한 branch
 - 이름을 다른 사람들도 이해하기 쉽도록
- ✓ Feature branch에서 master branch로 병합.



1. 현재 branch 확인

```
>> git branch  
master // 현재 branch 이름
```

2. Branch 생성

```
>> git branch 새로운_branch_명  
>> git branch new_branch
```

3. Branch 변경

```
>> git checkout 변경하고자_하는_branch_명  
>> git checkout new_branch
```

4. Merge branch

<현재 branch(new_branch)에 test.txt commit> (1)

```
>> git branch
```

```
new_branch
```

```
>> git add test.txt
```

```
>> git commit -m "New file test.txt"
```

<반영할 branch(master)로 이동> (2)

```
>> git checkout 반영할_branch_명
```

```
>> git checkout master
```

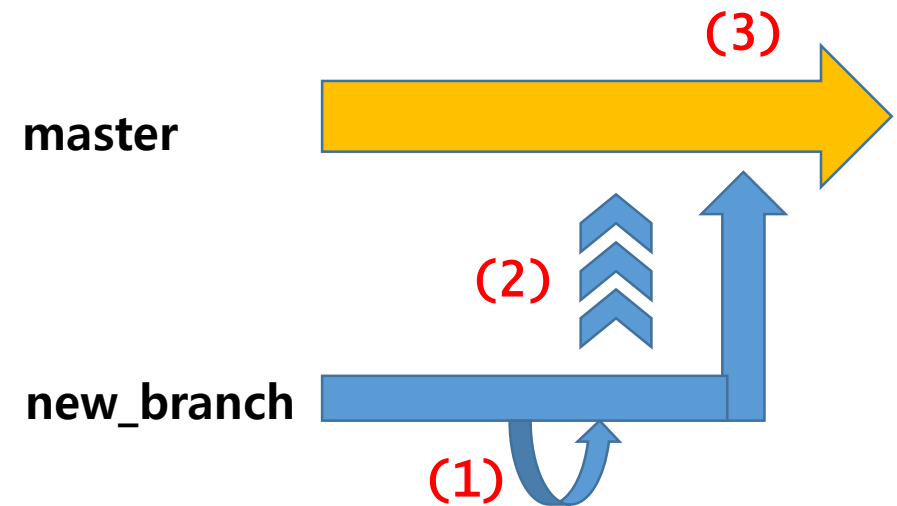
```
>> git branch
```

```
master
```

<merge - new_branch를 master에 merge> (3)

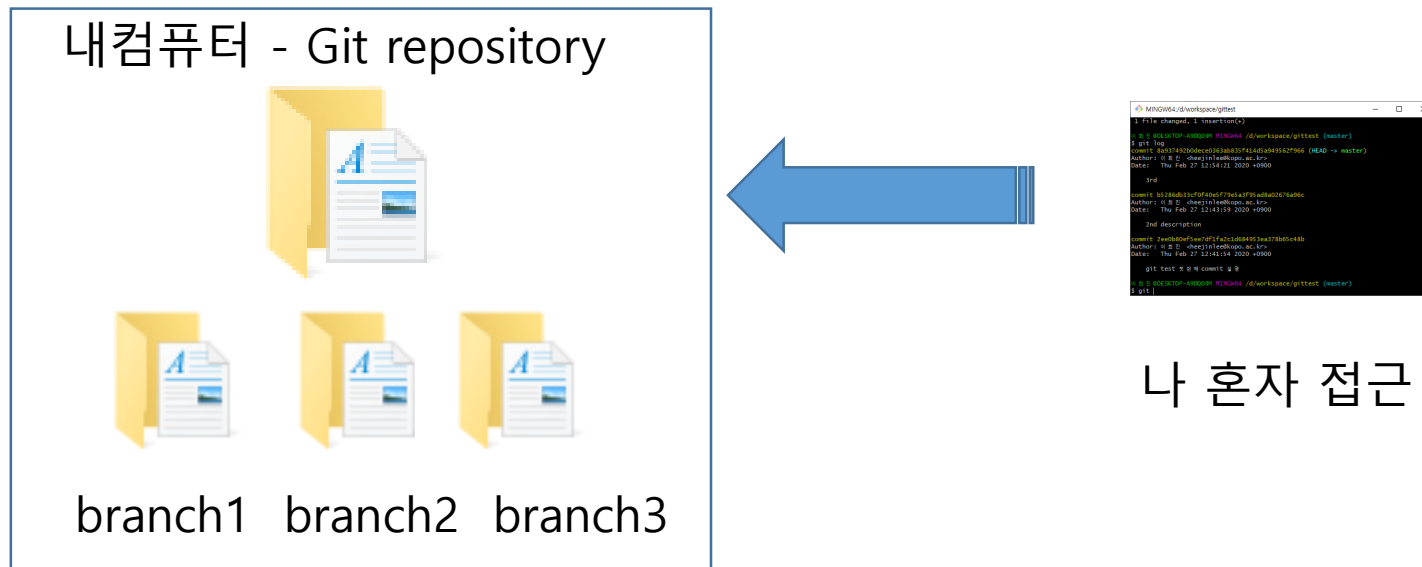
```
>> git merge 병합할_branch_명
```

```
>> git merge new_branch
```



Summary

- Github에 가입하고, branch 생성, file commit 해보았다.
- Git을 내 컴퓨터에서 쓸 수 있다.
 - ✓ 내 컴퓨터에 저장공간을 놓고, 소프트웨어 형상관리를 할 수 있다.
 - ✓ 물리적으로 혼자 쓰는 공간이지만, 버전관리를 위하여 사용



- ➔ 공용 저장소에 올려서 여러명이 다 같이 접근해서 사용하자. 내 컴퓨터와 외부 공용 저장소를 연결하자

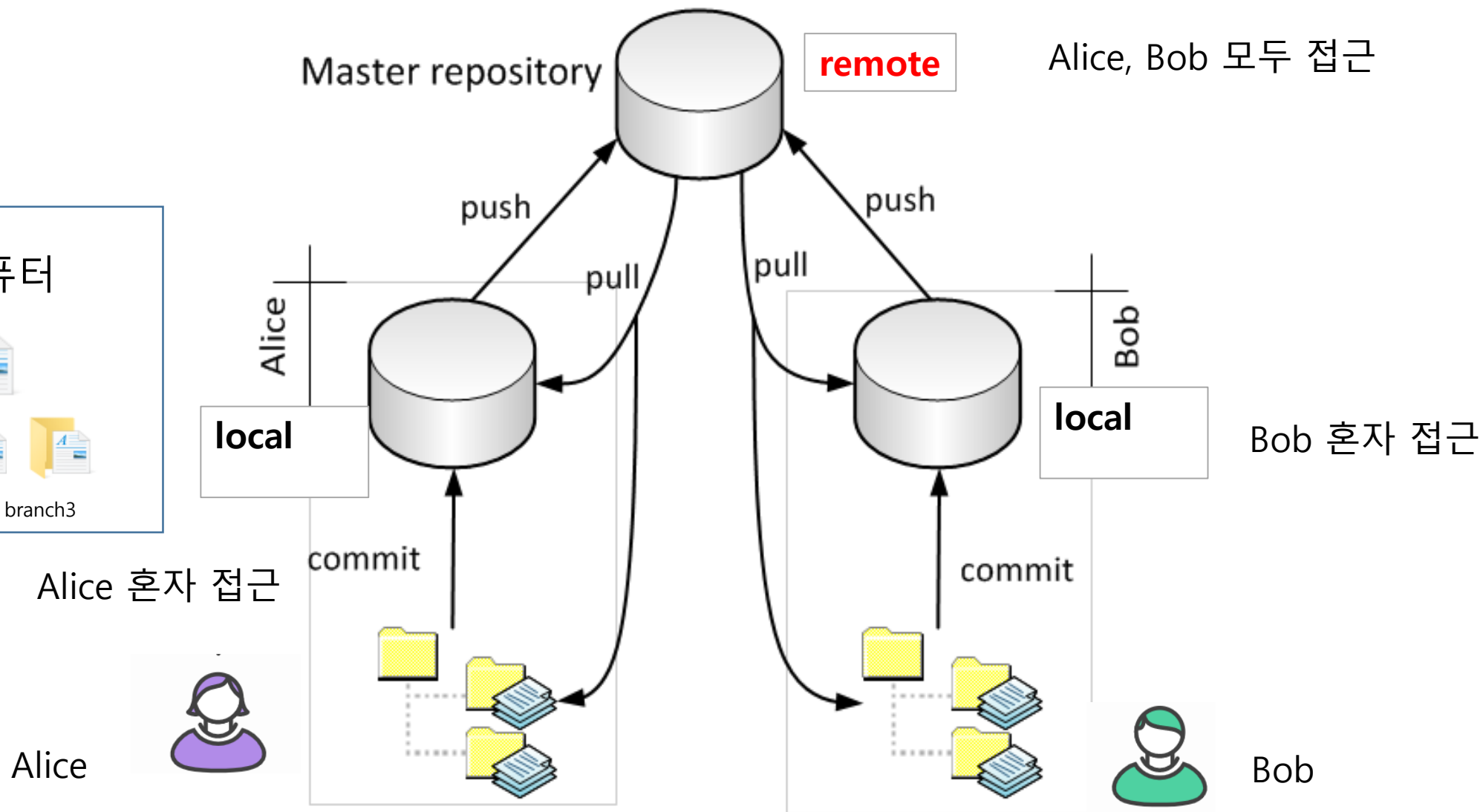
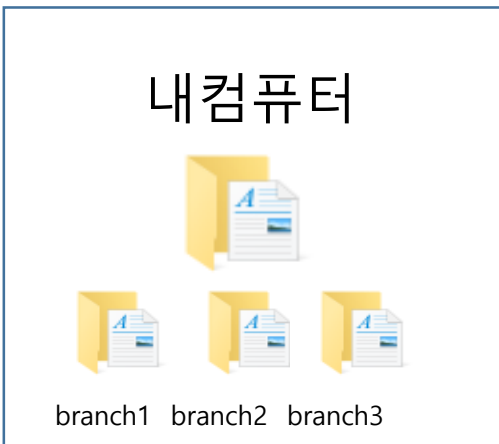
Ⅲ. Remote 저장소 연동하기

Ⅲ.1 remote 저장소 추가 및 관리

Ⅲ.2 Git Clone

Ⅲ.3 Git Push

Ⅲ.4 Git Pull

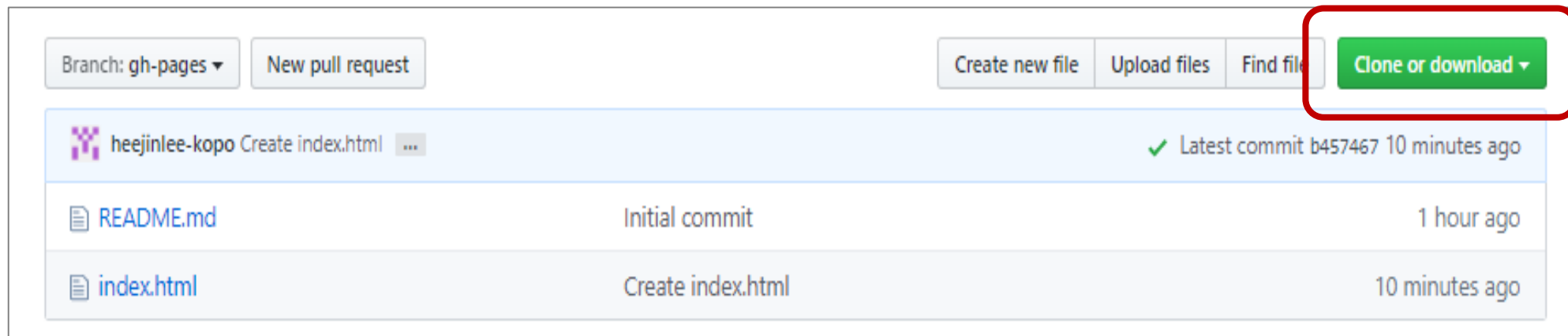


- 로컬저장소와 Remote 저장소 (Github/외부 서버) 연결

```
>> git remote add 저장소이름 원격저장소_주소 //원격저장소 저장
>> git remote add origin http://github.com/xxx/yyy.git

>> git remote                               //원격저장소 목록 - 이름
origin

>> git remote -v                             //원격저장소 목록 - 이름, 주소
origin http://github.com/xxx/yyy.git
```



Remote 저장소에서 가져오기 - Git Clone

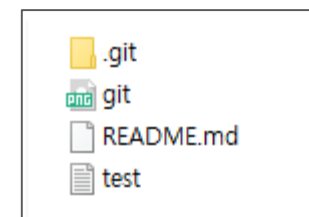
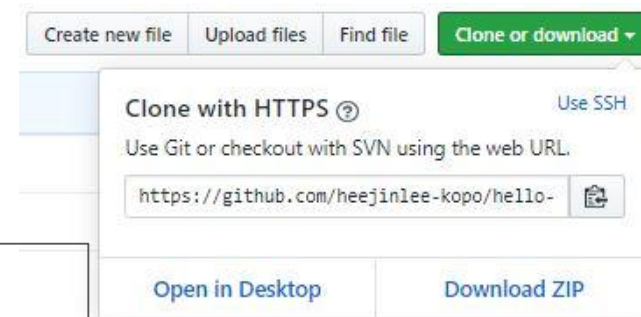
remote 저장소 연동

- **Git Clone - Remote 저장소의 branch를 그대로 로컬 저장소로 가져오기**

- ✓ 외부 repository 주소 복사
 - `https://github.com/dongmisw/hello-world.git`
- ✓ 해당 디렉토리에서 `git clone` 명령

```
>> git clone 외부_repository_주소  
>> git clone https://github.com/user\_id/repository\_name.git
```

- ✓ 해당 디렉토리에 폴더 생성 및 `.git` 폴더 생성(숨겨진 폴더)



- Q1. clone을 할 때 어떤 branch가 clone되는가?
- Q2. 다른 branch를 clone할 방법이 있는가?

- Q2. 다른 branch를 clone할 방법이 있는가?

```
>> git clone -b 특정_branch_이름 외부_repository_주소  
>> git clone -b mybranch https://github.com/user\_id/repository\_name.git
```

Remote 저장소에서 전체 가져오기 – Git Clone

- 이미 default branch를 clone했는데, 다른 branch도 clone하고 싶다면?
- Remote 저장소 branch 확인

```
>> git branch -r  
>> git branch -a
```

//Remote 저장소 branch 리스트
//Local, Remote 저장소 branch 리스트

```
heejinlee@LAPTOP-RMM0FDLH MINGW64 ~/git/2021helloworld2/helloworld2 (feature)  
$ git branch -r  
origin/HEAD -> origin/main  
origin/feature  
origin/feature2  
origin/main  
  
heejinlee@LAPTOP-RMM0FDLH MINGW64 ~/git/2021helloworld2/helloworld2 (feature)  
$ git branch -a  
* feature  
main  
remotes/origin/HEAD -> origin/main  
remotes/origin/feature  
remotes/origin/feature2  
remotes/origin/main
```

- Remote 저장소 branch 가져오기

```
>> git checkout -t 외부특정_branch_이름  
>> git checkout -t origin/feature2
```

- **Push**

- ✓ Remote 저장소에 정보 저장하는 것
- ✓ Local 저장소에 commit 한 것은 remote 저장소에 자동 저장되지 않으므로, 명시적으로 작업 필요
- ✓ Default branch에 반영

```
>> git push
```

- **실습**

1. Remote 저장소에서 repository를 Clone
2. 새로운 file을 추가하여 local 저장소에 commit
3. Remote 저장소로 push

Remote 저장소에 변경사항 업데이트 받기 – Git Pull

remote 저장소 연동

• Pull

- ✓ Remote 저장소의 내용을 로컬 저장소로 가져오는 것
- ✓ 다른 사람이 작업한 것을 업데이트
- ✓ Fetch + Merge
 - fetch – remote 저장소에서 데이터 가져오는 것
 - merge – 다른 부분을 합치는 것

```
>> git pull  
  
>> git fetch  
>> git merge
```

• 실습 1

1. Remote 저장소 repository에 새로운 file을 추가 (remote 저장소에 변화 생김)
2. Pull 명령어를 통하여 변화를 local로 반영

• 실습 2

1. Remote 저장소 repository에 새로운 file을 추가 (remote 저장소에 변화 생김)
2. Fetch 명령어
3. Local 저장소에 반영되었는지 확인
4. Merge 명령어
5. Local 저장소에 반영되었는지 확인



Q & A



Thank you