

# GIT

천재교육 서비스개발팀  
이수한

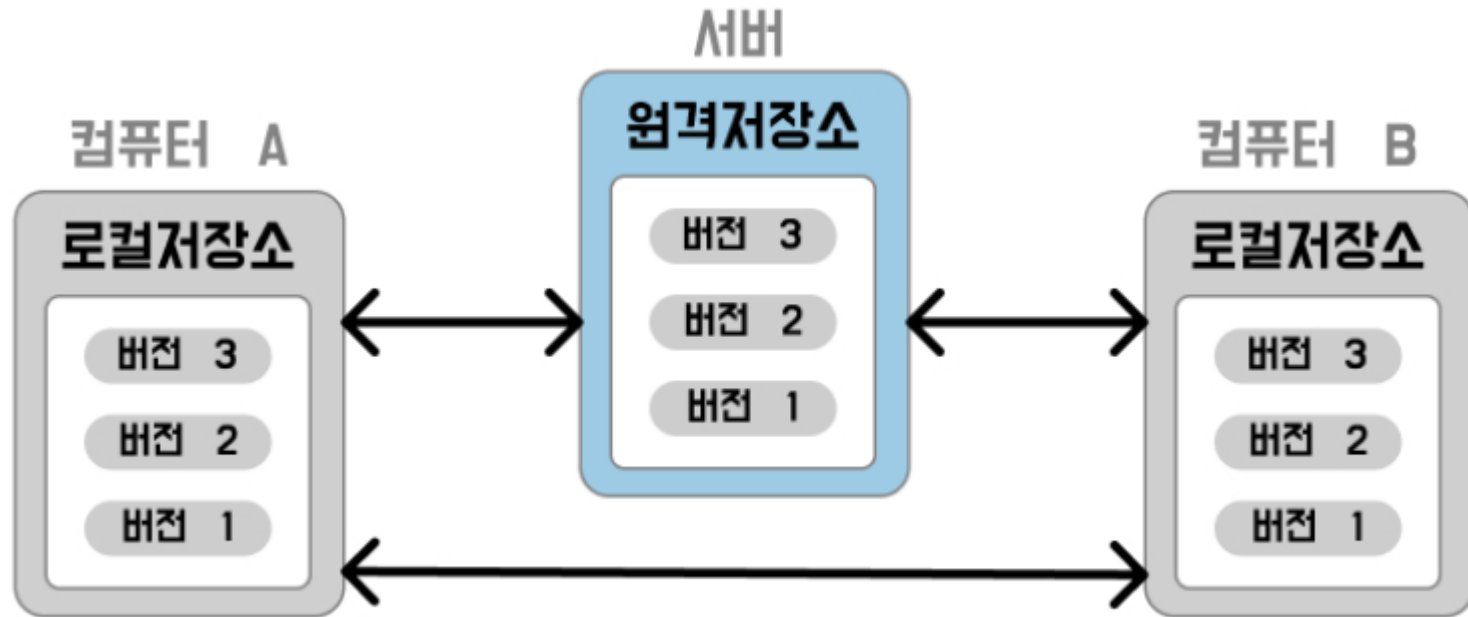
# GIT의 개념과 이해

# 버전 관리 시스템(VCS)

- LVCS(Local VCS) – 로컬 버전 관리
  - 하나의 로컬 컴퓨터에서 파일 관리.
  - EX) RCS(Revision Control System)
- CVCS(CVCS-Central VCS)- 중앙집중식 버전관리
  - 파일을 관리하는 서버가 별도로 존재하고 클라이언트가 중앙서버에서 파일을 받아 사용 (checkout)함.
  - EX) CVS, SVN, Perforce
- DVCS(Distributed VCS)- 분산버전관리
  - 클라이언트가 단순히 파일의 마지막 스냅샷만 사용하지않음.
  - 서버에 문제가 생기면 복제물로 다시 작업 시작. 클라이언트 중 하나를 골라 서버에 복원 .
  - 리모트 저장소가 존재해 다양그룹과 다양한 방법으로 협업가능.
  - EX) **Git**, Mercurial, Darcs

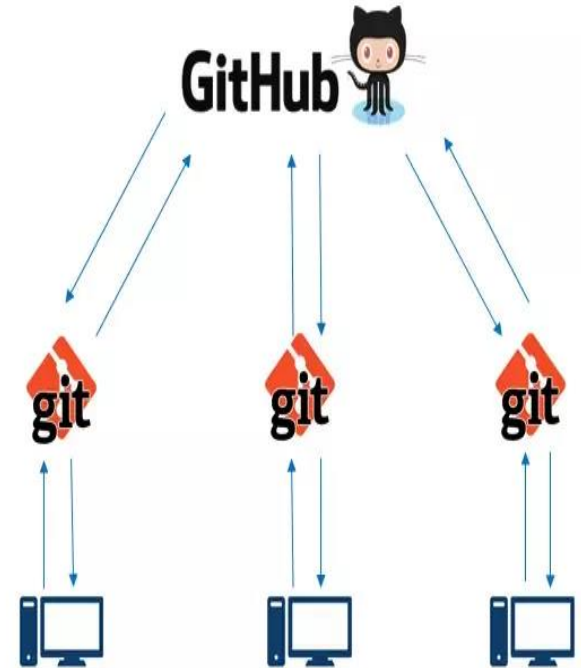
# Git 이란

- 컴퓨터 파일의 변경사항을 추적(문서를 수정할 때마다 언제 무엇을 왜 변경 했는지에 대한 일력 관리.)
- 여러 명의 사용자와 협업을 위한 스냅샷 스트림 기반의 분산버전 관리시스템.
- 깃 -> 1.버전관리 2.백업하기 3.협업하기



# Git과 GitHub

- Git 은 버전관리 시스템이고 GitHub는 Git으로 관리하는 프로젝트를 올려 놓을 수 있는 사이트
- Git의 repository 를 업로드 할 수 있는 웹사이트
- GitHub는 다른 사람들과의 협력을 용이.
- 리포지토리를 공유할 수 있는 중앙저장소, 웹 기반 인터페이스.
- 팀원들과 보다 효율적으로 변경 안을 구체화 하고 토론하여 검토 가능



Repository: 조직내의 흩어져 있는 각종 정보나 개발에 관련된 정보를 모아놓고 서로 공유 할 수 있게 한 정보의 저장소

# Git을 사용하는 이유

- **변경취소**
  - 실수를 했을 경우 구 버전의 작업파일을 복구하여 이전단계로 돌아 갈 수 있다.
- **모든 변경에 대한 완벽한 이력(history)**
  - 구 버전의 프로젝트를 확인해 그 당시 파일의 상태를 정확히 볼 수 있다.
- **변경한 이유 기록**
  - 왜 변경했는지 기억할 수 없을때 Git의 commit message를 이용하여 변경한 이유를 추후에도 참조할 수 있다.
- **변경에 대한 확신**
  - 잘 되지 않으면 언제든지 이전 버전으로 복귀하면 된다.
- **여러 갈래의 히스토리(history)**
  - 다른 기능을 독립적으로 실험해 보기 위해 별도의 branch를 생성할 수 있다. 성공하면 변경내용을 master branch로 병합하고 잘 동작하지 않으면 삭제할 수 있다.
- **충돌 해결 능력**
  - Git을 이용하며 여러 사람들이 동시에 같은 파일을 작업할 수 있다. 대개 자동으로 변경사항을 병합할 수 있다. 그렇지 못할 경우 충돌이 무엇인지 알려주고 이를 해결하기 쉽게 해준다.
- **독립된 히스토리(history)**
  - 여러 사람들이 다른 branch에서 작업이 가능하다. 기능을 독립적으로 개발하고 완료되었을 때 기 기능을 병합할 수 있다.

# Git을 사용하는 이유

- 기록요구
  - Issues를 사용해 버그를 기록하거나 개발하고 싶은 새로운 기능을 구체화 할 수 있다.
- 독립된 히스토리(history)에 대한 협력
  - Branch와 pull request를 이용해 다른 branch 또는 기능에 협력할 수 있다.
- 진행중인 작업 검토
  - Pull request 목록을 통해 현재 무슨 작업이 진행되고 있는지 모두 볼 수 있다. 그리고 특정 pull request를 클릭하여 최근의 변경 내용과 변경에 관한 모든 논의 내용을 볼 수 있다.
- 팀의 작업 진척 상황 확인
  - Pulse를 훑어보거나 commit history를 살펴보면 팀의 진척상황을 알 수 있다.

# 용어

- **Commit**
  - 하나 또는 다수의 파일에 대한 변경 내용을 저장할 때마다 새로운 commit을 생성한다.
  - (이 변경내용을 commit하고 GitHub에 push 합니다.)
- **Commit message**
  - commit을 생성할 때 마다 왜 변경했는지에 대한 이유를 설명하는 메시지를 제공해야 한다. 이 commit 메시지는 변경한 이유가 추후에 유용하다.
  - (새로운 member controller에 대한 박진수에 대한 의견을 commit 메시지에 꼭 넣으세요)
- **Branch**
  - 테스트를 해 보거나 새로운 기능을 개발하기 위해 사용할 수 있는 따로 떨어진 독립적인 commit들을 말한다.
  - (새로운 검색 기능을 구현하기 위해 branch를 생성합니다)
- **Master branch**
  - 새로운 Git프로젝트를 만들때 마다 master라는 기본 branch가 생성된다. 배포할 준비가 되면 작업이 최종적으로 마무리되는 branch이다.
  - (master로 바로 commit하면 안 된다는 것을 기억하세요)
- **Feature(혹은 topic) branch**
  - 새로운 기능을 개발할 때 마다 작업할 branch를 만드는데 이를 feature branch라고 한다.
  - (feature branch가 너무 많습니다. 이들 중 두개를 완료해서 출시하는데 주력합니다)
- **Release branch**
  - 직접 QA(품질보증) 작업을 하거나 고객의 요구로 구 버전의 소프트웨어를 지원해야 한다면 모든 수정이나 업데이트를 위한 장소로 release branch가 필요하다. feacher branch와 release branch는 기술적인 차이는 없지만 팀원들과 프로젝트에 대해 이야기 할 때 구별할 수 있어 유용하다.
  - (release branch 전체에 대한 보안 버그를 수정해야 합니다.)
- **Merge(병합)**
  - Merge는 한 branch에서 완성된 작업을 가져와 다른 branch에 포함하는 방법이다. 흔히 feacher branch를 master branch로 병합한다.
  - ("로그인" 기능이 훌륭합니다. 배포할 수 있게 master로 병합해 줄 수 있겠습니까?)



# 용어

- **Tag**
  - 특정 이력이 있는 commit에 대한 참조, 어떤 버전의 코드가 언제 배포(release)되었는지 정확히 알 수 있도록 제품 배포 기록에 가장 자주 사용된다.
  - (이번 배포판에 태그를 달고 출시 합니다.)
- **Check out**
  - 프로젝트 history의 다른 버전으로 이동해 해당 시점의 파일을 보기 위해 check out 한다. 가장 일반적으로 branch에서 완료된 작업을 모두 보기위해 check out 하지만, commit도 check out 할 수 있다.
  - (최종 릴리즈 태그를 check out 해주시겠어요? 제품에 버그가 있어서 검증하고 수정해야 합니다.)
- **Pull request**
  - Branch에서 완료된 작업을 다른 사람이 리뷰하고 master로 병합하도록 요청하기 위해 사용. 요즘은 개발 가능한 기능에 대한 논의를 시작하는 초기 작업 단계에서 자주 사용한다
  - (다른 팀원들이 어떻게 생각하는지 알 수 있게 새로운 투표 기능에 대한 pull request를 만드십시오)
- **Issue**
  - GitHub는 기스에 대해 논의하거나 버그를 추적하거나 혹은 두가지 경우 모두 사용될 수 있는 Issue라는 기능을 갖고 있다.
  - (당신이 맞아요. 아이폰에서는 로그인 되지 않네요. 버그를 검증하기 위한 단계를 기록하면서 GitHub에 Issue를 생성해 주시겠습니까?)
- **Wiki**
  - Ward Cunningham이 최초로 개발. Wiki는 링크들 간을 연결해 간단하게 웹 페이지를 만드는 방법이다. GitHub 프로젝트는 문서작성에 Wiki를 자주 사용한다.
  - (복수의 서버에서 작동하게끔 프로젝트 환경 설정 방법을 설명하는 페이지를 추가해 주시겠습니까?)
- **Clone(복제)**
  - 종종 로컬로 작업하기 위해 프로젝트 복사본을 GitHub에서 다운로드한다. Repository를 사용자의 컴퓨터로 복사하는 과정을 cloning(복제) 이라고 한다.
  - 리포지토리를 복제하고 버그를 수정한 다음 오늘밤 수정본을 다시 GitHub로 push 해주시겠습니까?)
- **Fork**
  - 때로는 프로젝트를 직접 변경하는데 필요한 권한을 보유하지 못할때가 있다. 잘 알지 못하는 사람이 작성한 오픈소스 프로젝트 이거나, 회사에서 작업을 같이 많이 하지 않는 다른 그룹이 작성한 프로젝트일 수도 있다. 그러한 프로젝트를 변경하고 싶다면 GitHub의 사용자 계정에 프로젝트 복사본을 만들어야 한다. 그 과정을 리포지토리를 fork 한다고 한다. 그런 다음 복제(clone)하고 변경하며, pull request를 이용해 원본 프로젝트에 변경 내용을 반영할 수 있다.
  - (홈페이지 마케팅 원고를 당신이 어떻게 재작성 했는지 보고 싶습니다. 리포지토리를 fork하고 수정안과 함께 pull request를 제출해 주십시오.)

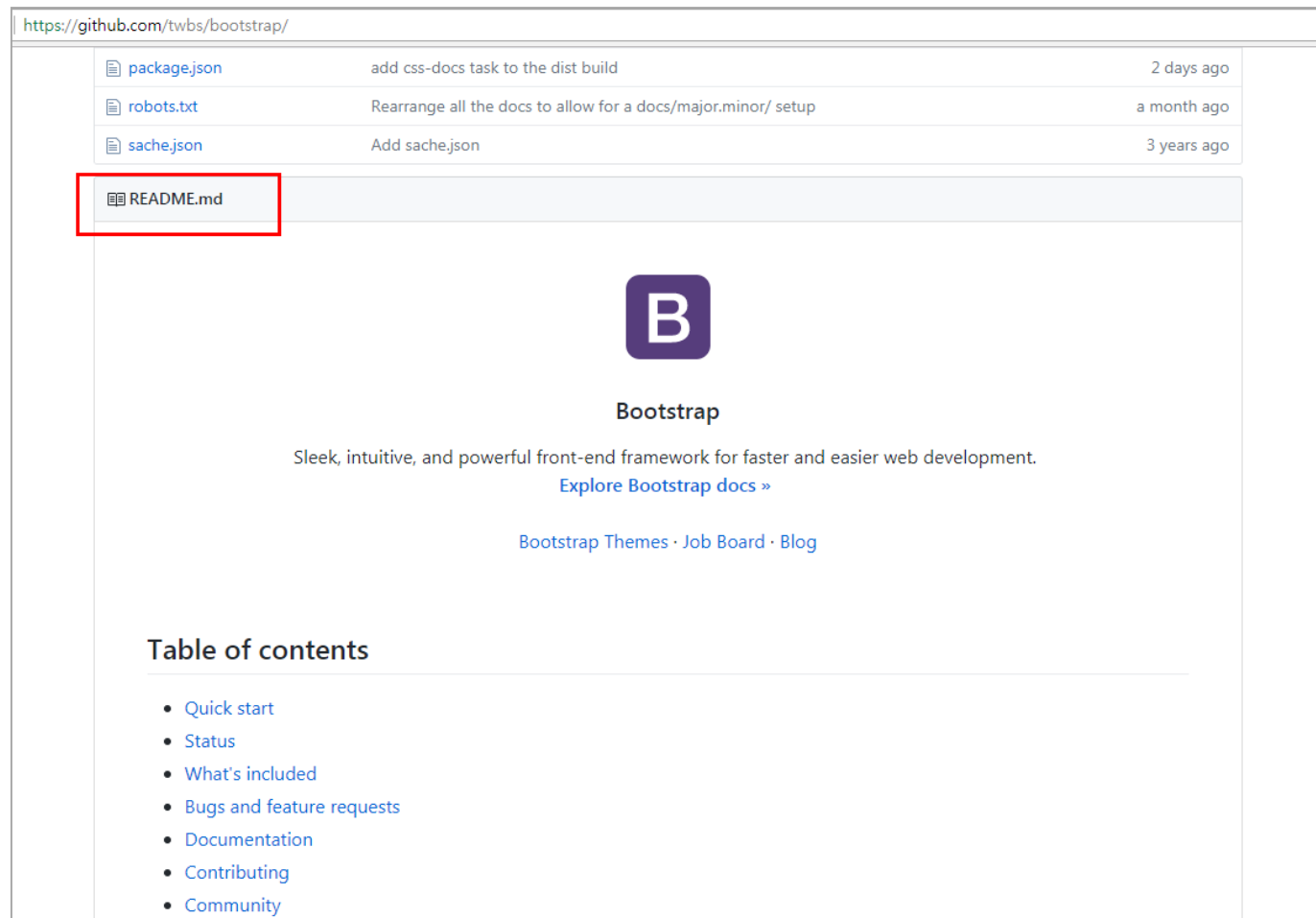
# 프로젝트 상태 확인

- <https://github.com/twbs/bootstrap>

The screenshot displays the GitHub repository page for `twbs / bootstrap`. The repository is described as "The most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web." with a link to <http://getbootstrap.com>. The page includes navigation links for Features, Business, Explore, Marketplace, and Pricing. The repository statistics are highlighted with a red box: 6,982 Watchers, 112,178 Stars, and 51,898 Forks. The repository has 16,440 commits, 25 branches, 41 releases, and 865 contributors. The MIT license is also shown. The page includes a "Clone or download" button and a list of recent commits.

Commit	Message	Time
mdo fixes #22968	Rearrange all the docs to allow for a docs/major.minor/ setup	a month ago
_data	fix some busted links	10 hours ago
_includes	fix some busted links	10 hours ago

# README.md



The screenshot shows the GitHub repository page for Bootstrap. At the top, there's a list of recent commits with files like package.json, robots.txt, and sache.json. Below this, the README.md file is highlighted with a red box. The main content of the README features the Bootstrap logo (a purple square with a white 'B'), the word 'Bootstrap' in bold, and a description: 'Sleek, intuitive, and powerful front-end framework for faster and easier web development.' It includes a link to 'Explore Bootstrap docs »' and another set of links for 'Bootstrap Themes · Job Board · Blog'. At the bottom, there's a 'Table of contents' section with a list of links: Quick start, Status, What's included, Bugs and feature requests, Documentation, Contributing, and Community.

https://github.com/twbs/bootstrap/

package.json	add css-docs task to the dist build	2 days ago
robots.txt	Rearrange all the docs to allow for a docs/major.minor/ setup	a month ago
sache.json	Add sache.json	3 years ago

README.md

**B**

**Bootstrap**

Sleek, intuitive, and powerful front-end framework for faster and easier web development.

[Explore Bootstrap docs »](#)

[Bootstrap Themes](#) · [Job Board](#) · [Blog](#)

### Table of contents

- [Quick start](#)
- [Status](#)
- [What's included](#)
- [Bugs and feature requests](#)
- [Documentation](#)
- [Contributing](#)
- [Community](#)

프로젝트에 대한 소개와 협력자들에게 유용한 추가 정보(소프트웨어를 어떻게 설치하는지, 자동화된 테스트를 어떻게 실행하는지, 코드를 어떻게 사용하는지, 프로젝트에 어떻게 기여할 수 있는지)를 제공한다.

# Commit History 보기: 가장 최근 작업이 무엇인지 알아볼 수 있다.

The screenshot shows the GitHub repository page for `twbs/bootstrap`. The repository has 6,982 watches, 112,178 stars, and 51,898 forks. The commit history is displayed, showing the latest commit `6cf1a10` by `mdo` with the message `fixes #22968`. A red arrow points from the `16,440 commits` link in the repository header to the commit list.

Repository: `twbs / bootstrap`

Stats: 6,982 Watch, 112,178 Star, 51,898 Fork

Branch: `v4-dev` | New pull request | Find file | Clone or download

Commits on Jul 3, 2017

- `fixes #22968` (6cf1a10) - mdo committed 7 hours ago ✓
- `fix some busted links` (67245a4) - mdo committed 10 hours ago ✓
- `Merge branch 'v4-dev' of https://github.com/twbs/bootstrap into v4-dev` (ad15fc0) - mdo committed 11 hours ago ✓
- `npm build` (93b2682) - mdo committed 11 hours ago


Commits on Jul 2, 2017

- `follow up to 4e067f7` (3b08beb) - mdo committed 22 hours ago
- `Removed Japanese translation` (ac3e4b7) - infectedbean committed with mdo 11 days ago ✗
- `Merge branch 'v4-dev' of https://github.com/twbs/bootstrap into v4-dev` (7bce864) - mdo committed 22 hours ago ✗
- `remove unused $list-group-color variable; it should naturally inherit...` (4e067f7) - mdo committed 22 hours ago 2
- `update comment` (f282a88) - mdo committed with mdo 23 hours ago ✓
- `put button active bg and border in the mixin params for more customiz...` (4b6d2c0) - mdo committed 23 hours ago

# Commit History 보기

fixes #22968

v4-dev (#66)

 mdo committed 8 hours ago

1 parent 67245a4    commit 6cf1a106d4fcd043f47d3eab4ffef3eaf868ce38

Browse files

Showing 1 changed file with 1 addition and 1 deletion.

Unified Split

2 docs/4.0/migration.md

@@ -266,6 +266,6 @@ Our responsive utility classes have largely been removed in favor of explicit `d`

266 266 - Old names: `.visible-print-block`, `.visible-print-inline`, `.visible-print-inline-block`, `.hidden-print`

267 267 - New classes: `.d-print-block`, `.d-print-inline`, `.d-print-inline-block`, `.d-print-none`

268 268

269 -Rather than using explicit `.visible-\*` classes, you make an element visible by simply not hiding it at that screen size. You can combine one `.d-\*-none` class with one `.d-\*-block` class to show an element only on a given interval of screen sizes (e.g. `.d-none.d-md-block.d-lg-none` shows the element only on medium and large devices).

269 +Rather than using explicit `.visible-\*` classes, you make an element visible by simply not hiding it at that screen size. You can combine one `.d-\*-none` class with one `.d-\*-block` class to show an element only on a given interval of screen sizes (e.g. `.d-none.d-md-block.d-xl-none` shows the element only on medium and large devices).

270 270

271 271 Note that the changes to the grid breakpoints in v4 means that you'll need to go one breakpoint larger to achieve the same results. The new responsive utility classes don't attempt to accommodate less common cases where an element's visibility can't be expressed as a single contiguous range of viewport sizes; you will instead need to use custom CSS in such cases.

0 comments on commit 6cf1a10

Please [sign in](#) to comment.

커밋 메세지

빨간색 : 삭제된 콘텐츠

녹색 : 추가된 콘텐츠

# Pull request 보기

- 현재 진행중인 작업이 무엇인지 알 수 있다.
- Pull request중 하나를 선택하면 그 pull request를 설명하는 제목, 변경안을 포함하는 하나이상의 commit이 있으며 그 변경안에 대해 논의하고 있는 사람들이 달아 놓은 댓글이 많을 수도 있다.
- Pull request를 살펴보면 사람들이 현재 무슨 작업을 하고 있으며, 버그 수정을 하든 기능 개발을 하든 각각의 변경 사항에 대해 어떤 역할을 하고 있는지 알 수 있다.

The screenshot shows the GitHub interface for the `twbs/bootstrap` repository. The `Pull requests` tab is selected and highlighted with a red circle, showing 63 pull requests. The page includes a search bar with the query `is:pr is:open`, a `New pull request` button, and a list of open pull requests. Each pull request entry includes a title, a status check, labels, and the author's name.

Author	Labels	Projects	Milestones	Reviews	Assignee	Sort
#22965 opened a day ago by mdo	css, feature, v4					
#22964 opened a day ago by mdo	css, v4					
#22963 opened a day ago by mdo	css, v4					
#22960 opened a day ago by mdo	css, feature, v4					
#22958 opened a day ago by mdo	css, feature, v4					
#22955 opened a day ago by mdo	docs, js					
	css, v4					

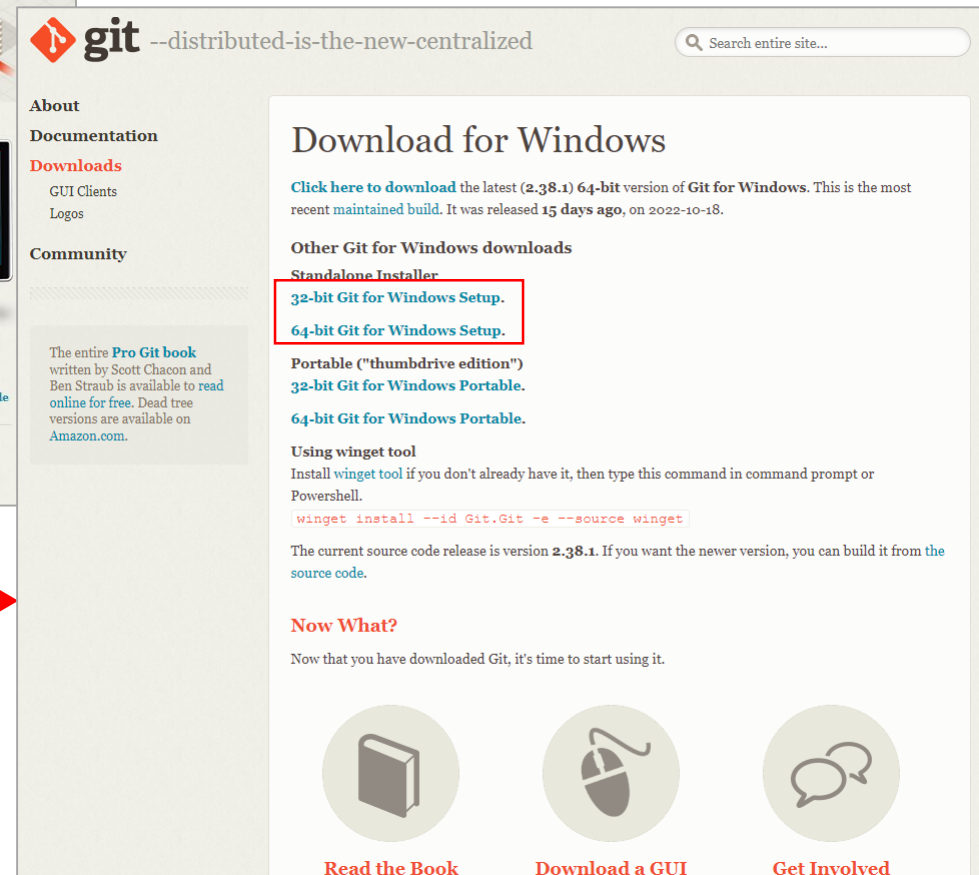
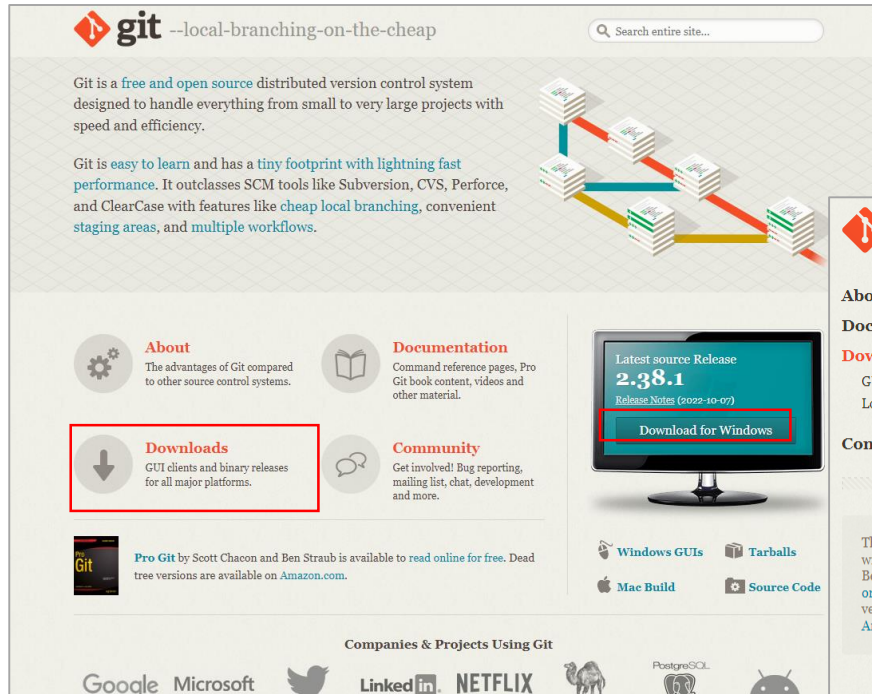
# Fork를 이용한 기여

- 프로젝트에 직접 참여하려면 그 프로젝트를 **소유하거나 협력자로 포함**되어야 한다.
- 그렇지 않으면 GitHub의 사용자 계정에 프로젝트의 복사본을 만든다. 이 과정을 forking 이라 한다.
- 프로젝트를 fork하면 복사본을 원하는 대로 변경할 수 있고 **pull request**를 이용해 원본 프로젝트에 변경 내용을 반영할 수 있다.

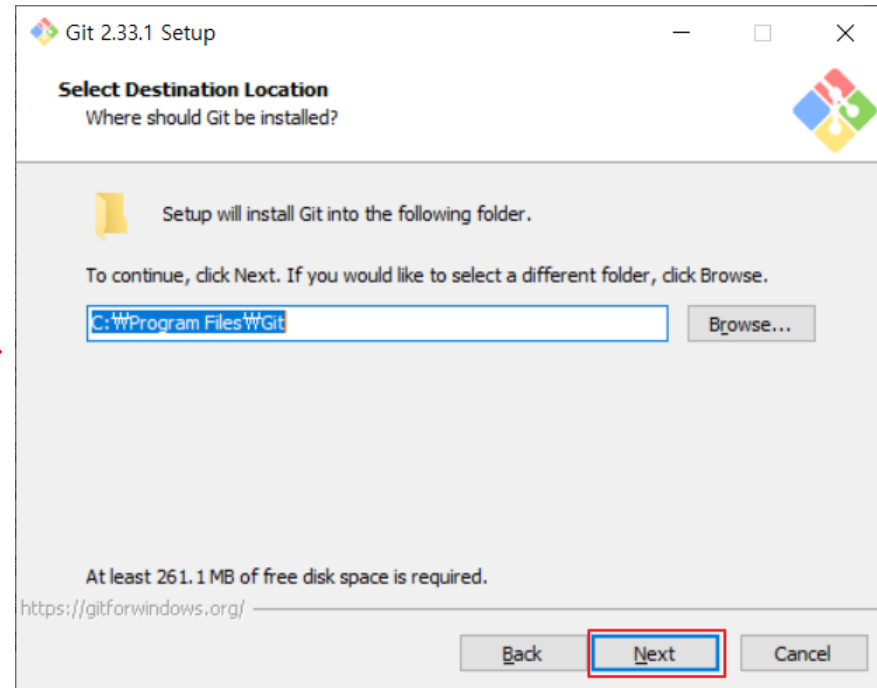
# Git 설치



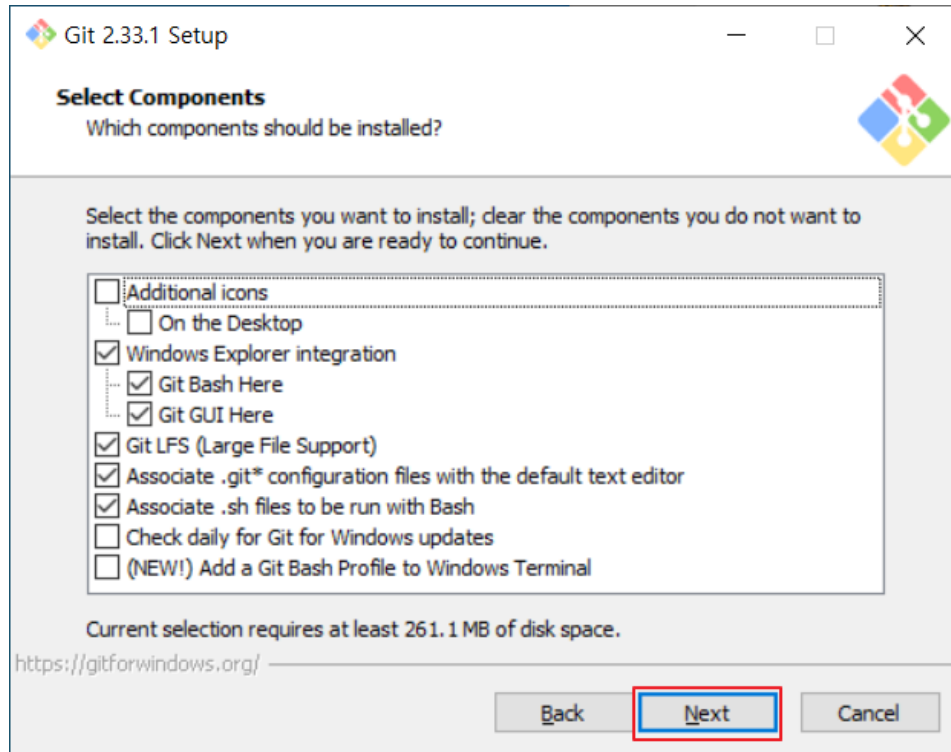
# <https://git-scm.com> 다운로드



# 프로그램 파일 위치 지정



# 설치 옵션 설정



## Additional icons

- On the Desktop : 바탕화면에 바로가기를 생성한다.

## ✓ Windows Explorer integration

- ✓ Git Bash Here : 폴더 오른쪽 클릭 메뉴에 이 폴더 위치에 Git CMD 화면을 띄우는 기능을 추가한다.

- ✓ Git GUI Here : 폴더 오른쪽 클릭 메뉴에 Git GUI 화면을 띄우는 기능을 추가한다.

## ✓ Git LFS (Large File Support) :

대용량 파일을 지원할지 안 할지 체크한다.

## ✓ Associate .git\* configuration files with the default text editor :

기본 텍스트 에디터에 .git 확장자를 연결한다.

## ✓ Associate .sh files to be run with Bash :

Bash에 .sh 확장자를 연결한다. (Linux)

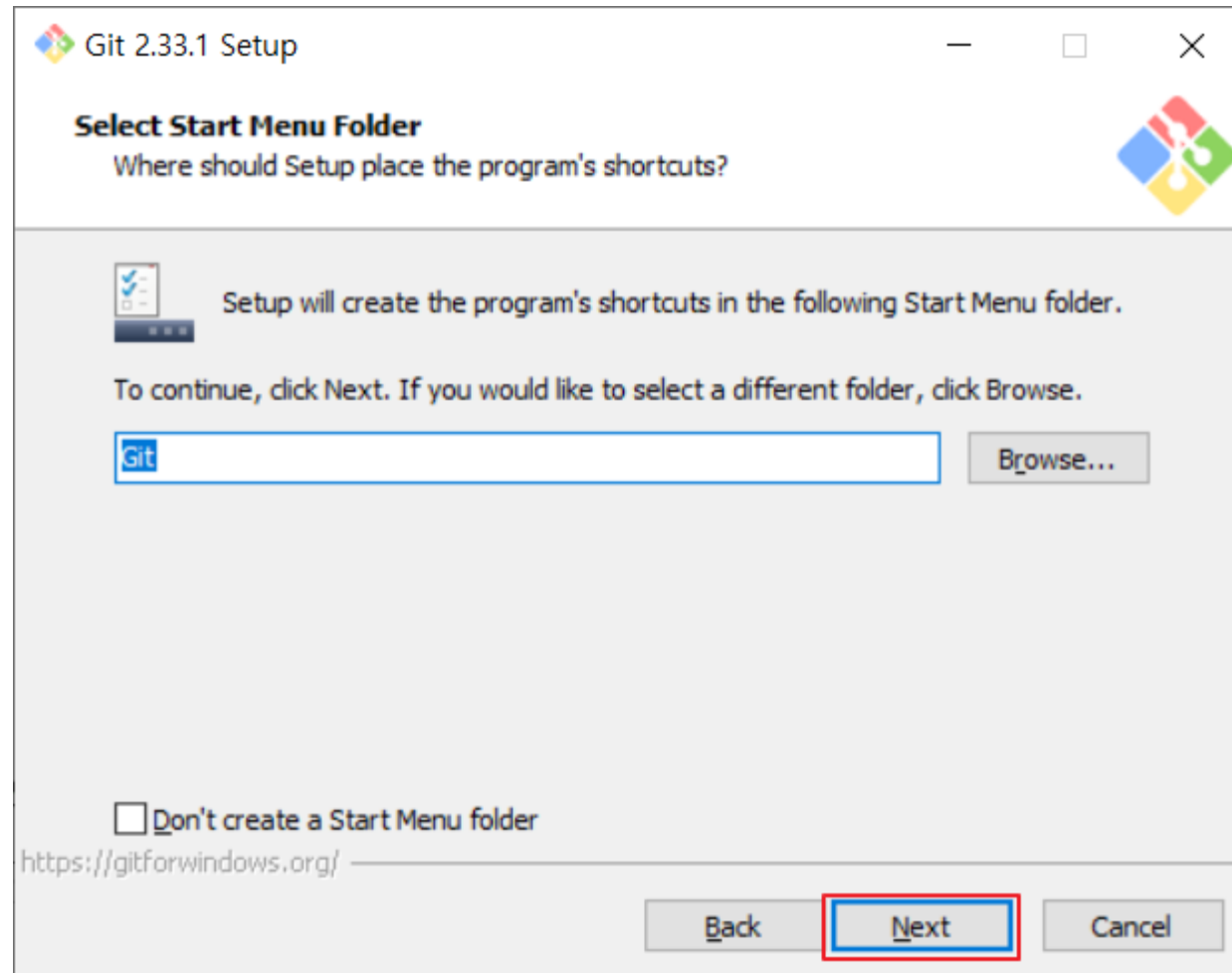
## Check daily for git for Windows updates :

(Windows 만) 매일 새로운 업데이트가 있는지 확인한다.

## Add a Git Bash Profile to Windows Terminal :

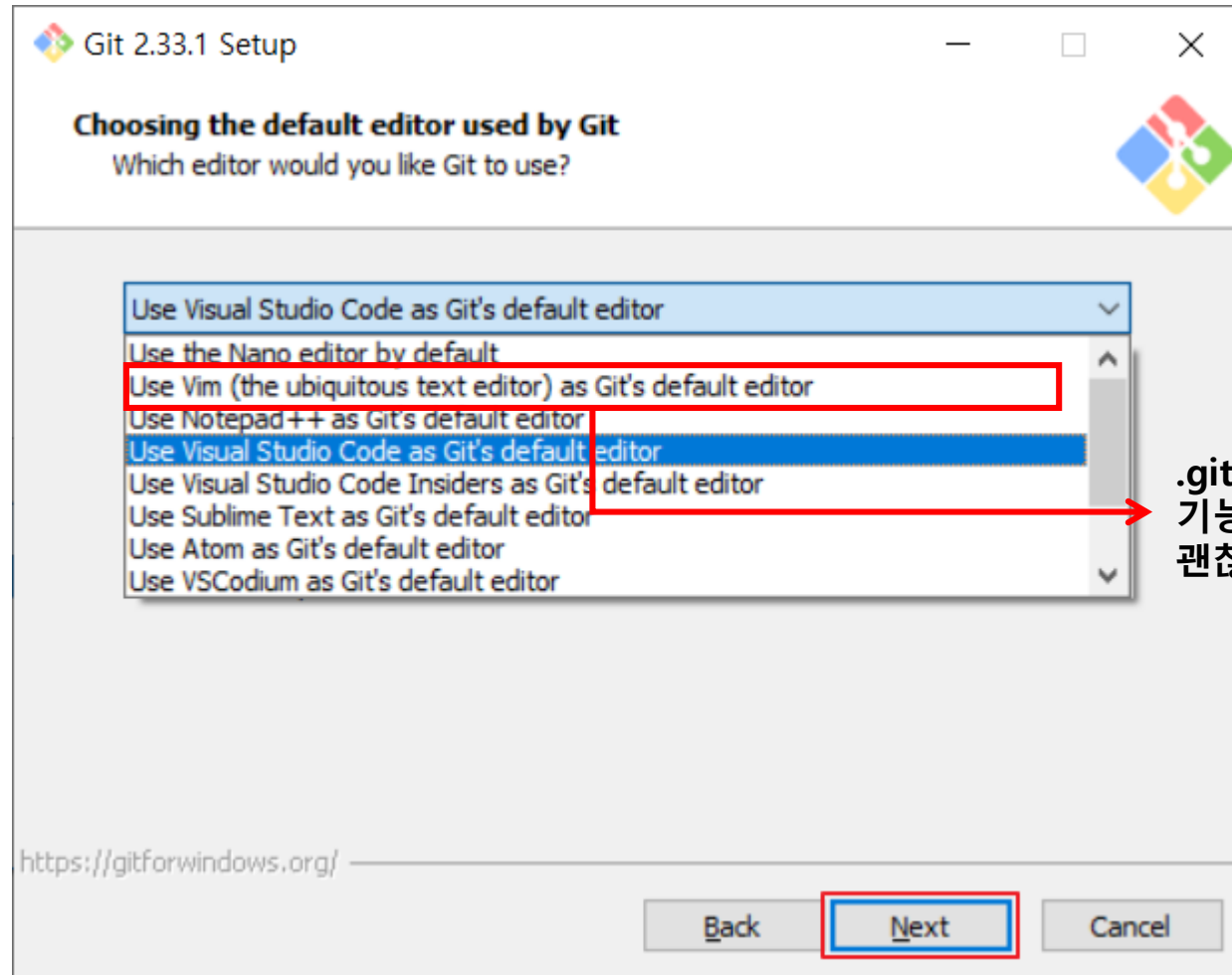
윈도우 기본 터미널에 Git Bash 프로파일을 추가한다.

# 바로가기 폴더 지정



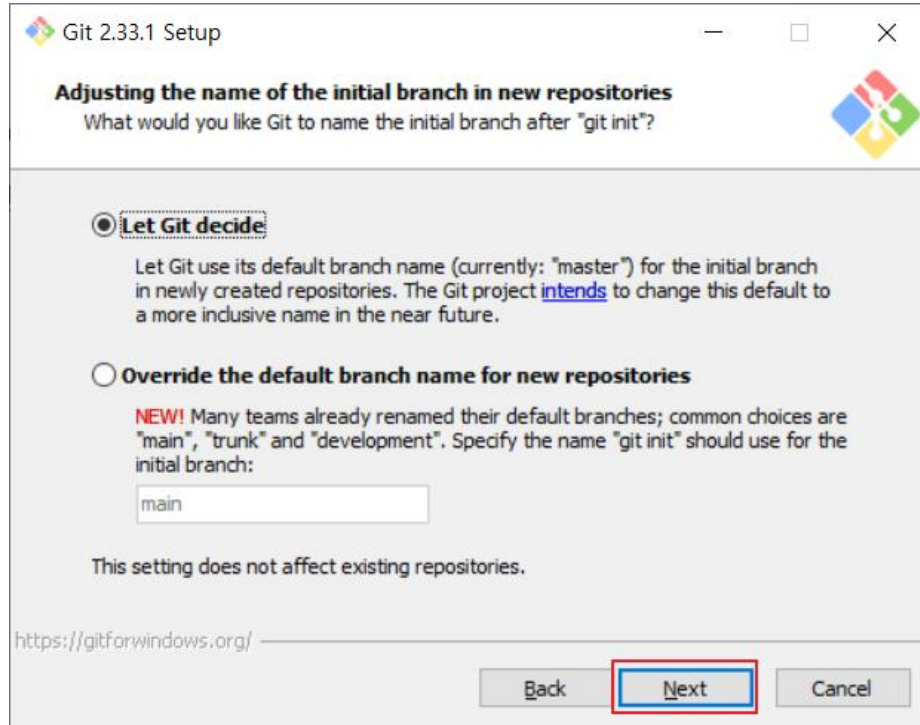
# 디폴트 에디터 지정

## - vim으로 설정



.git 확장자 파일 에디터를 위한  
기능으로 어떤 것이든 선택해도  
괜찮다. (Vim 사용)

# Repository 마스터 이름 지정



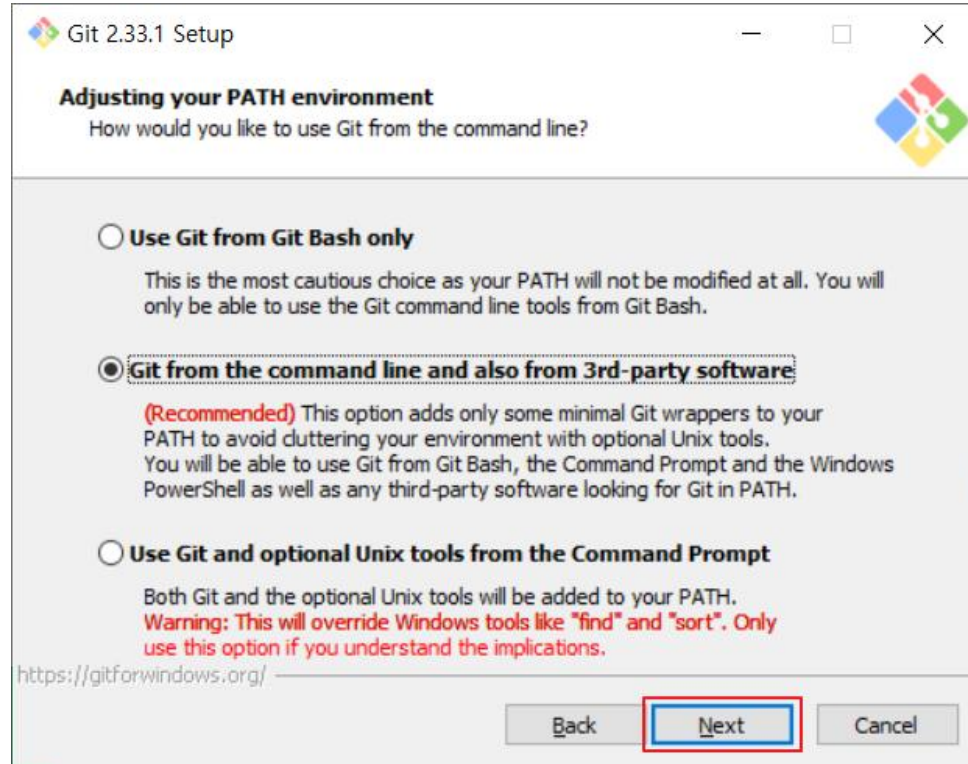
## ✓ Let Git decide :

새로운 Repository가 생성될 때 기본적으로 "master" 브랜치를 생성한다.

## Override the default branch name for new repositories :

새로운 Repository 생성될 때 사용자가 지정한 이름의 브랜치를 생성한다. 팀에서 관용적으로 쓰는 명칭이 있을 때 사용한다.

# 깃 커맨드 사용 설정



## Use Git from Git Bash only :

Git Bash(깃 전용 터미널) 에서만 Git 명령어를 수행한다.

## ✓ Git from the command line and also from 3rd-party software :

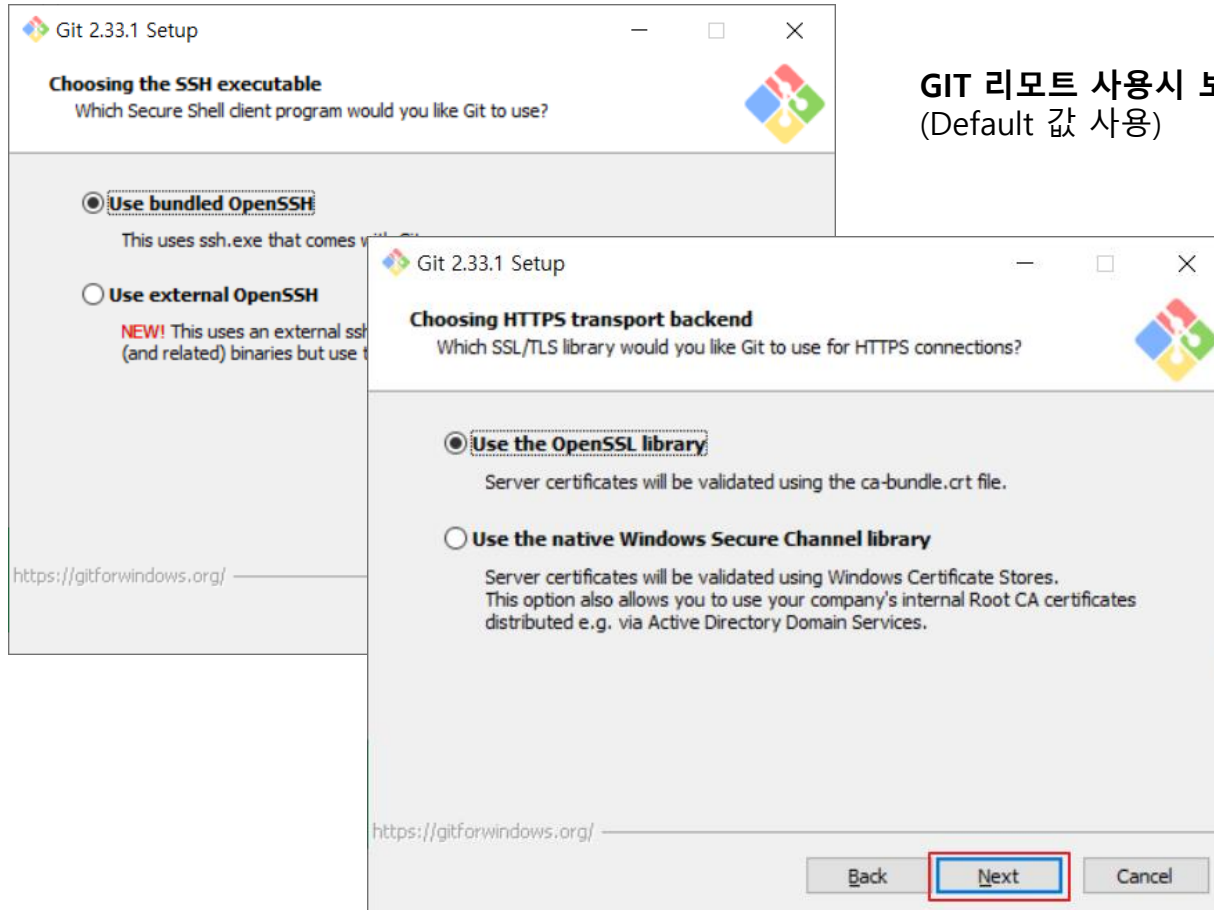
Git을 환경변수(PATH)에 추가하여 윈도우 기본 명령 프롬프트(CMD) 등에서도 Git 명령어가 수행 되도록 한다.

## Use Git and optional Unix tools from the Command Prompt :

Git과 Unix 도구를 모두 환경변수(PATH)에 추가한다. 이 경우 몇 가지 Windows 기본 기능들이 새로운 기능으로 덮어씌워진다. (CMD 명령어)

▶ 이러한 위험을 충분히 숙지하고 있는 경우에만 이 옵션을 사용하기를 권장합니다.

# SSH 설정

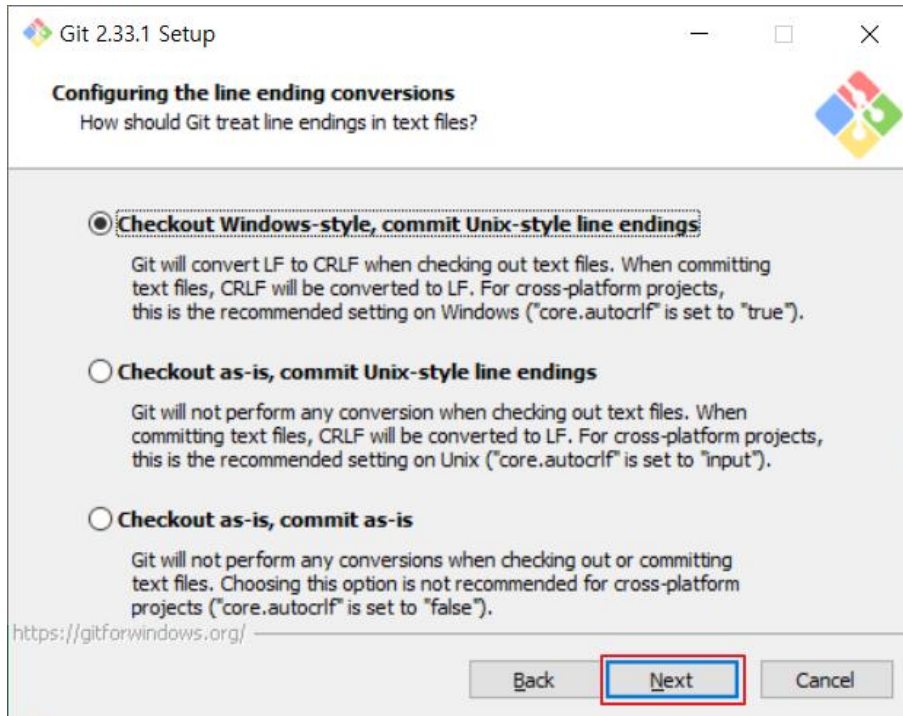


GIT 리모트 사용시 보안 연결(SSH) 설정  
(Default 값 사용)

- use bundled openSSH: 깃에 포함되어 있는 openssh를 사용.
- Use external openSSH: 외부 SSH 사용
- **Use the OpenSSL library** : OpenSSL 라이브러리 사용, 서버인증서는 ca-bundle.crt 파일을 사용하여 유효성 검사.
- **Use the native Windows Secure Channel library** : Windows 인증서 저장소를 사용하여 서버 인증서의 유효성 검사, Active Directory 도메인 서비스를 통해서 회사 내부 로트 CA인증서도 사용할 수 있습니다.



# GIT 저장소 줄바꿈 옵션



## ✓ Checkout Windows-style, commit Unix-style line endings :

체크아웃은 윈도우 스타일, 커밋은 유닉스 스타일로 자동 변경되도록 설정한다.

## Checkout as-is, commit Unix-style line endings :

체크아웃은 변경 없이, 커밋은 유닉스 스타일로 설정한다.

## Checkout as-is, commit as-is :

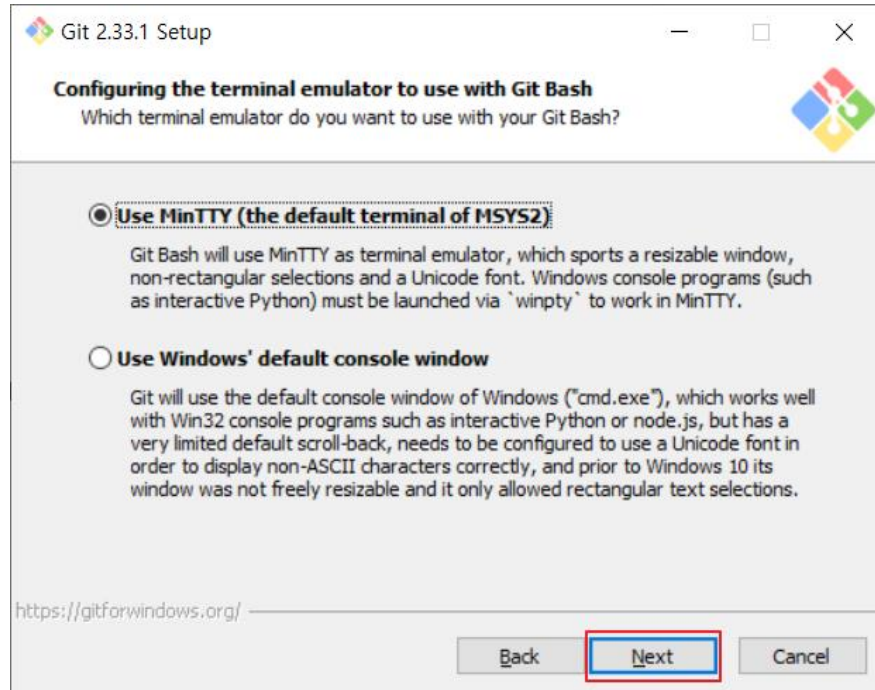
체크아웃, 커밋 모두 스타일 변경 없이 진행한다.

## ! 이 옵션은 무엇을 의미하나요?

윈도우와 유닉스의 개행(줄 바꿈) 표기가 서로 다릅니다. (윈도우: `\r\n` 유닉스: `\n`)

따라서 여러 운영체제에서 작업할 경우, 개행 표기가 달라져 수정 사항이 없음에도 수정된 것으로 인식할 가능성이 있습니다. 이 문제를 해결하기 위해 설정하는 옵션입니다.

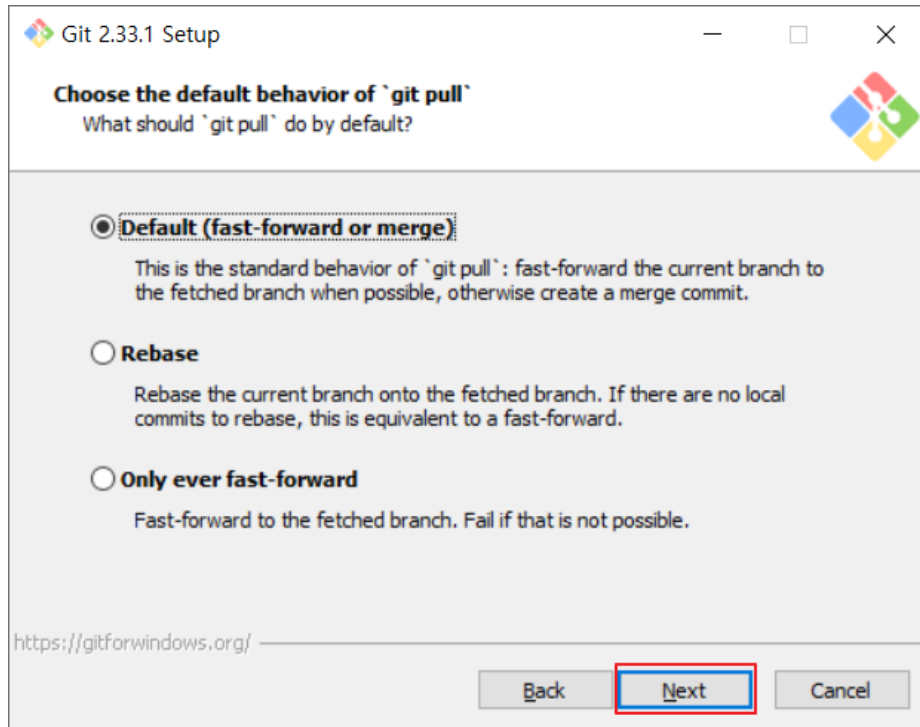
# GIT Bash 터미널 선택



✓ **Use MinTTY (the default terminal of MSYS2) :**  
Git Bash 기본 터미널 에뮬레이터(MinTTY)를 사용한다.

**Use Windows' default console window :**  
윈도우 기본 콘솔(CMD)을 사용한다.

# GIT PULL 수행작업 선택

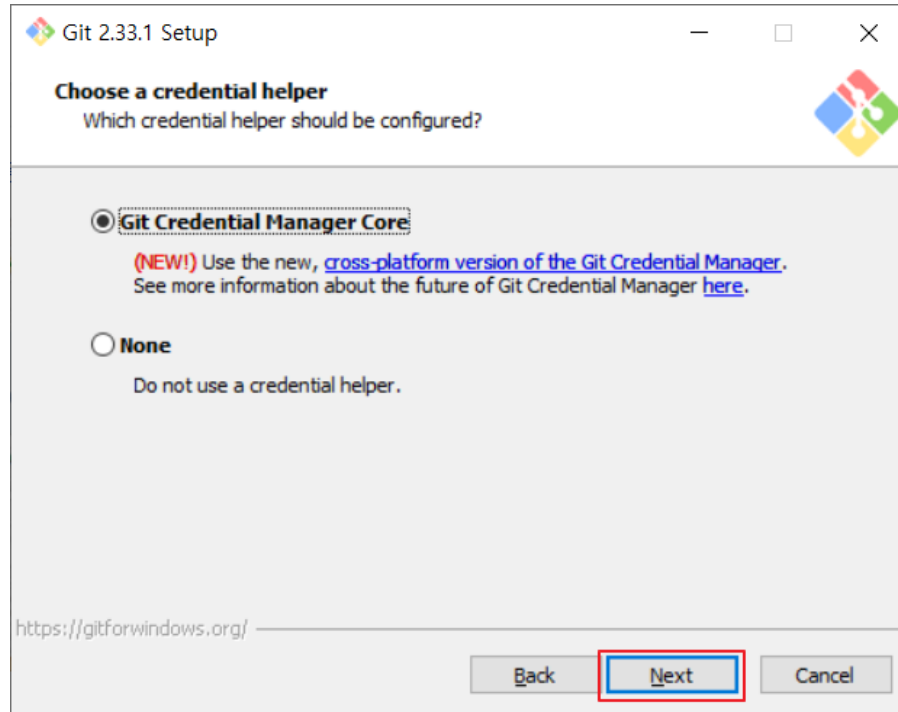


✓ **Default (fast-forward or merge) :**  
'git pull' 작업을 내부적으로 수행한다. (기본값)

**Rebase :**  
'git pull'의 수행 시 현재 브랜치를 불러온 브랜치에 새로 배치한다.

**Only ever fast-forward :**  
'git pull'의 수행 동작으로 불러온 분기로 빠르게 넘어간다. 명령어 수행에 실패할 가능성이 있다.

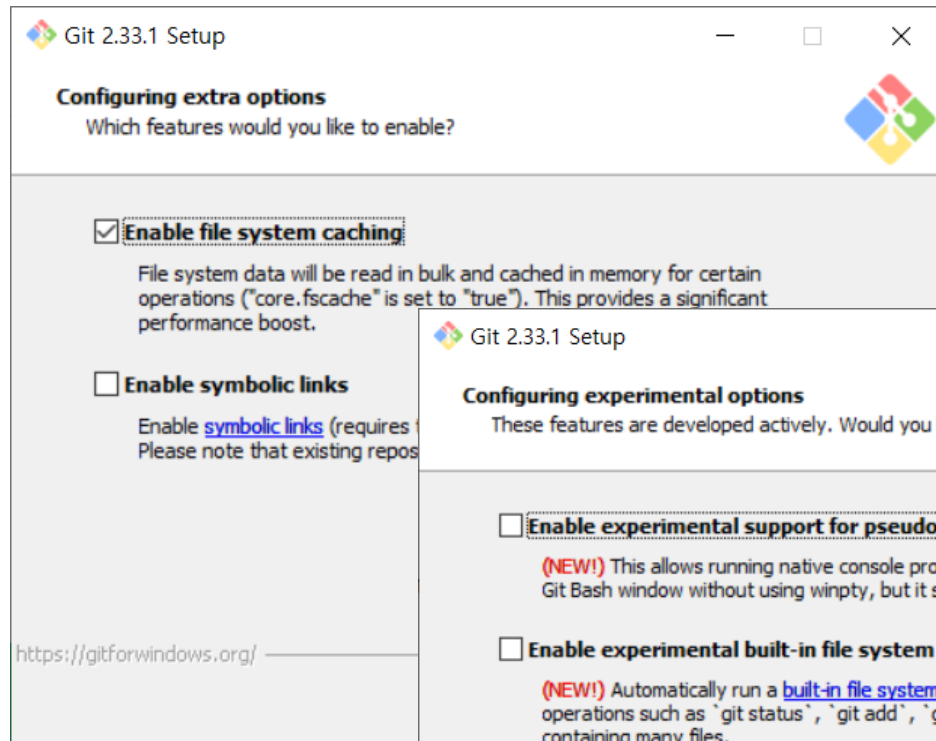
# GIT 보안 도우미 선택



✓ **Git Credential Manager Core :**  
Git에서 기본으로 제공하는 자격 증명 도우미를 사용한다.

**None :**  
자격 증명 도우미를 사용하지 않는다.

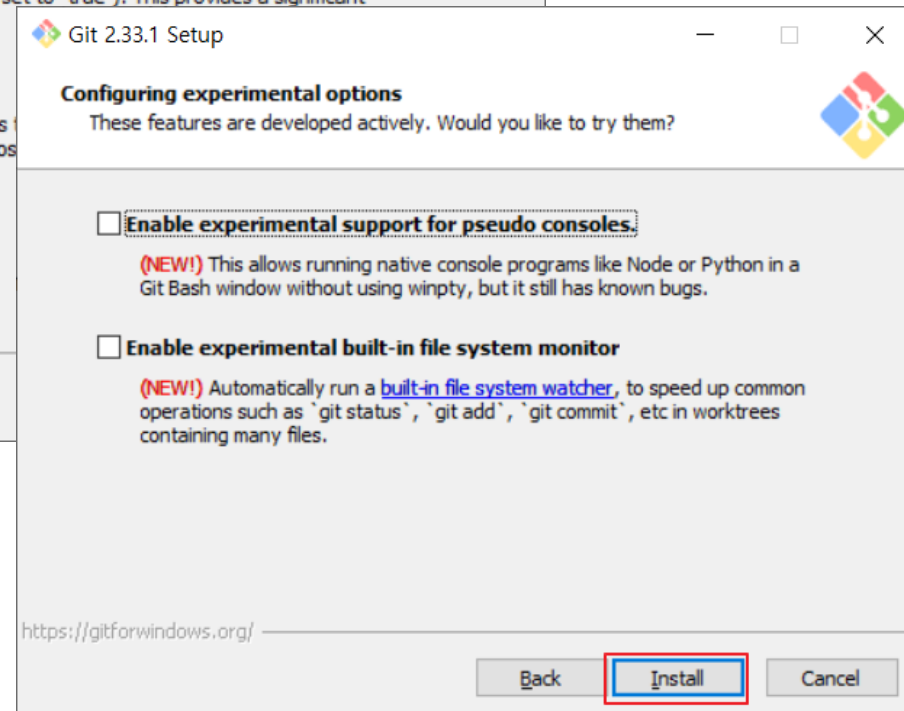
# 기타 옵션



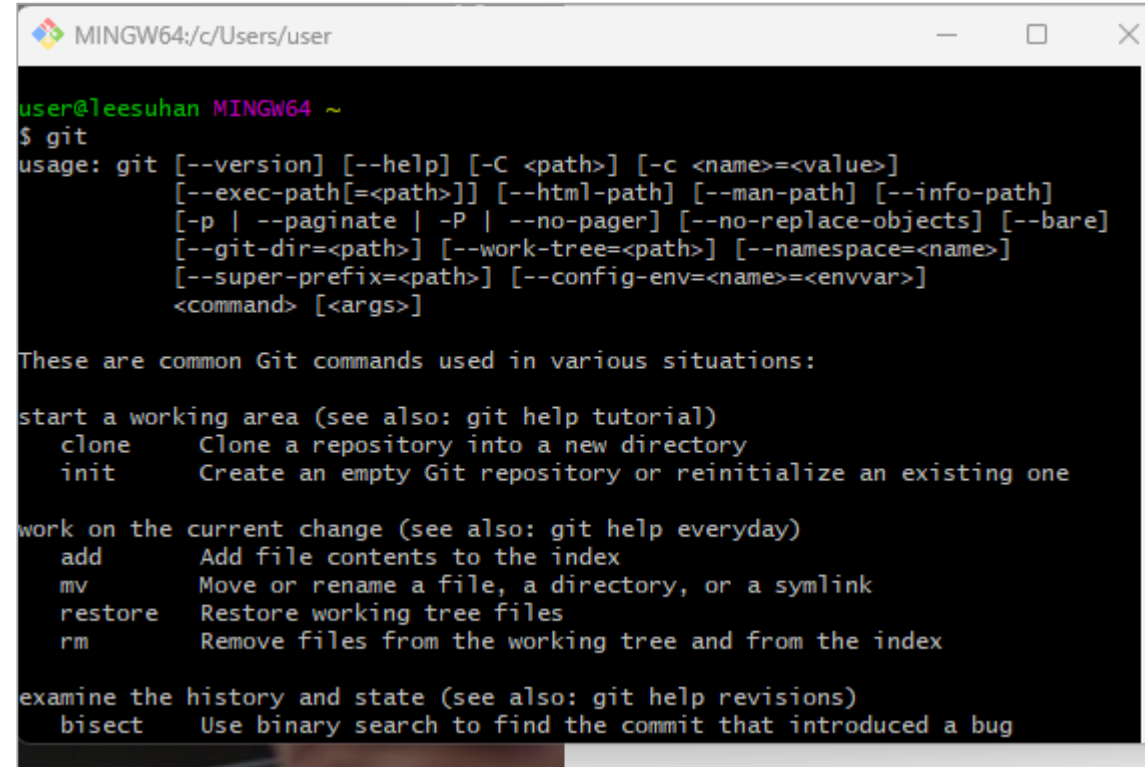
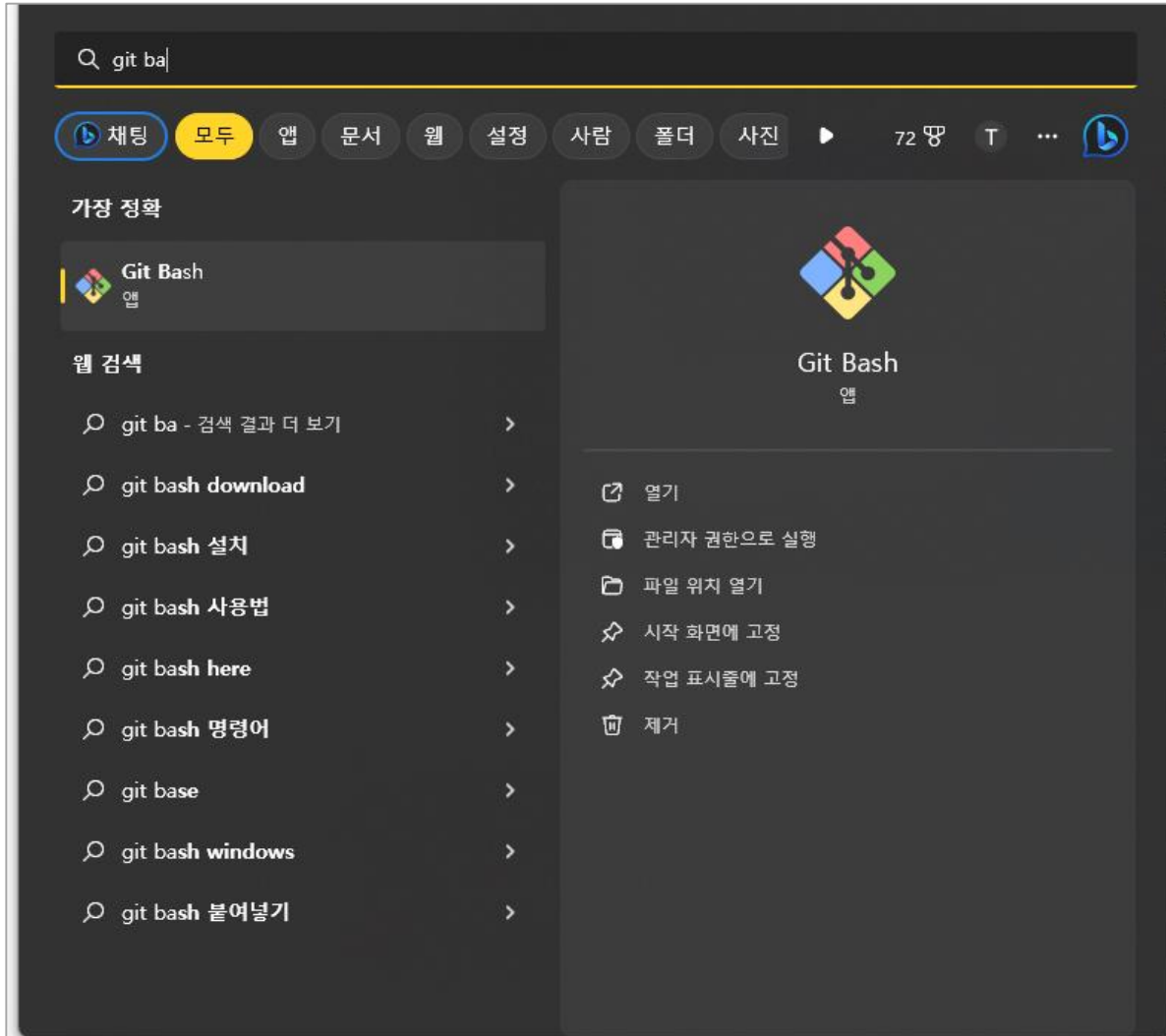
## ✔ Enable file system caching :

파일 시스템 캐싱을 활성화하여 상당한 성능 향상을 제공한다.

(그 외에는 실험적인 기능으로 선택하지 않는다)



# 윈도우에서 깃 실행하기



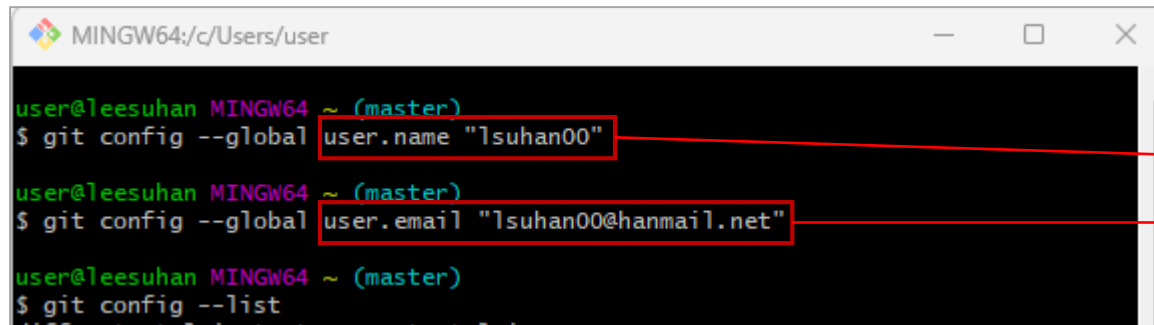
1. 윈도우 작업 표시줄의 검색창에 git bash 검색
2. Git bash창이 열리면 "git" 이라고 입력한 후 엔터
3. 깃 명령에서 사용할 수 있는 옵션이 표시되면 깃 설치 완료.

# 깃 환경설정 하기(깃허브와 연동)

1. <https://github.com>에서 계정 생성

2. git bash에서 사용자 정보 입력.

-> 깃은 협업을 전제로 하기 때문에 작업자에 대해 기록 하는 것이 중요하다. 깃을 사용 한다면 설치 후 미리 설정 권장.



```
MINGW64/c/Users/user
user@leesuhan MINGW64 ~ (master)
$ git config --global user.name "lsuhan00"
user@leesuhan MINGW64 ~ (master)
$ git config --global user.email "lsuhan00@hanmail.net"
user@leesuhan MINGW64 ~ (master)
$ git config --list
```

✓ 사용자 추가

1. Github 계정으로 사용자를 추가한다.

→ 1-1. user.name "깃허브 이름"

→ 1-2. user.email "깃허브에 등록한 메일"

2. git config --list 를 통해 사용자 정보 추가됐는지 확인

윈도우에서 깃 사용해보기



# VIM 편집기를 사용한 문서 작성

```
MINGW64:/d/git_test
user@leesuhan MINGW64 /d
$ cd /d

user@leesuhan MINGW64 /d
$ mkdir git_test

user@leesuhan MINGW64 /d
$ cd git_test

user@leesuhan MINGW64 /d/git_test
$ vim test.txt

user@leesuhan MINGW64 /d/git_test
$ cat test.txt
오늘 날씨가 좋습니다.
```

1. 원하는 경로에 git\_test 폴더를 만든다.
2. git\_test 폴더에 들어가 test.txt 를 만든다.
3. 편집기에서 i 혹은 a 를 누른 후 텍스트를 입력한다. (입력 모드로 전환)
4. ESC 를 누른다. (명령 모드로 전환)
5. :wq 명령어를 통해 편집 문서를 저장 한 후 종료한다. (w: 저장, q: 나가기)
6. cat 명령어를 이용해 텍스트 문서를 확인한다.

## ! VIM 이란

터미널에서 사용 할 수 있는 편집기.(리눅스의 기본편집기)  
폴더에 text문서를 만들어 사용해도 무관.

### <<리눅스 명령어>>

- cd: 디렉터리 이동하기
- mkdir: 폴더 생성하기
- vim: vim 편집기 사용하기
- cat: 파일 미리보기

# 깃 저장소 만들기

```
MINGW64:/d/git_test
user@leesuhan MINGW64 /d/git_test
$ ls
test.txt

user@leesuhan MINGW64 /d/git_test
$ git init
Initialized empty Git repository in D:/git_test/.git/

user@leesuhan MINGW64 /d/git_test (master)
$ ls -la
total 9
drwxr-xr-x 1 user 197121 0 Oct 4 12:47 ./
drwxr-xr-x 1 user 197121 0 Oct 4 12:39 ../
drwxr-xr-x 1 user 197121 0 Oct 4 12:47 .git/
-rw-r--r-- 1 user 197121 31 Oct 4 12:40 test.txt
```

1. git init 명령어를 사용해 현재 디렉터리에서 깃을 사용 할 수 있도록 초기화 (.git 파일 생성)
2. ls -la 명령어를 사용해 안에 있는 디렉토리 확인 -l -a 는 숨김파일 까지 확인가능

# 저장소에 text 파일 올리기.

```
MINGW64:/d/git_test

user@leesuhan MINGW64 /d/git_test (master)
$ 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)

user@leesuhan MINGW64 /d/git_test (master)
$ git add test.txt
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory

user@leesuhan MINGW64 /d/git_test (master)
$ git status
On branch master

No commits yet

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

```
user@leesuhan MINGW64 /d/git_test (master)
$ git commit -m "커밋 할꺼야"
[master (root-commit) 2f76725] 커밋 할꺼야
1 file changed, 1 insertion(+)
create mode 100644 test.txt

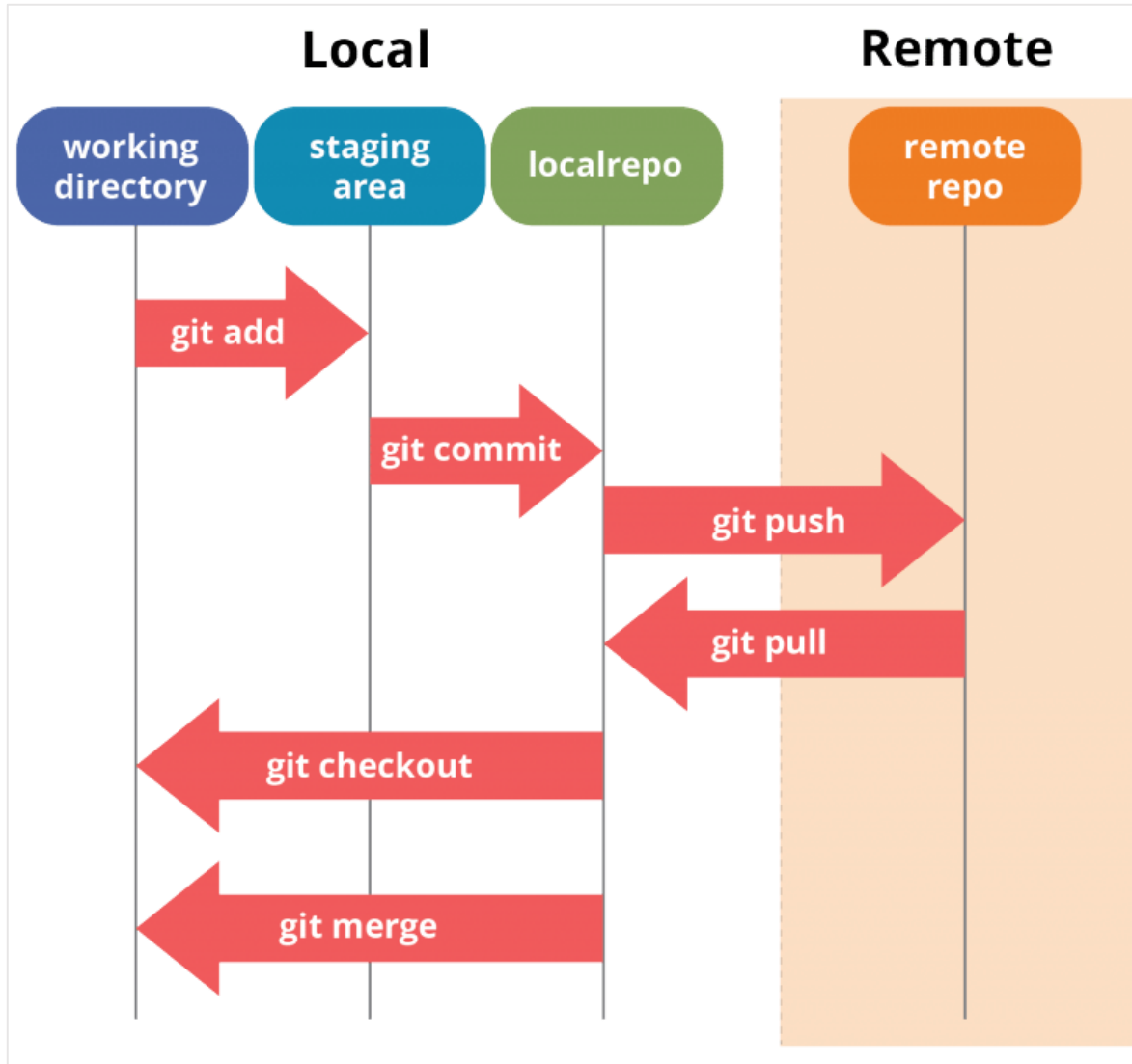
user@leesuhan MINGW64 /d/git_test (master)
$ git status
On branch master
nothing to commit, working tree clean

user@leesuhan MINGW64 /d/git_test (master)
$ git log
commit 2f767255b9f349ed8284f7cdb9f14c81e9afa0ba (HEAD -> master)
Author: lsuhan00 <lsuhan00@hanmail.net>
Date:   Wed Oct 4 12:58:16 2023 +0900

    커밋 할꺼야
```

1. git add를 통해 test.txt 파일 스테이징 영역에 올리기
2. 스테이지 영역에 있는 test.txt 파일을 commit 하기.  
(저장소에 올리기)

# Git 커밋 흐름



1. **Working directory(작업트리)** : 파일 수정 및 저장 작업을 하는 디렉터리. (우리가 눈으로 볼 수 있는 디렉터리. Ex)git\_test 폴더)
2. **Staging area(스테이지)** : 버전으로 만들 파일이 대기 하는 곳.  
ex) add 명령어를 통해 staging 영역에 들어감
1. **저장소**: 버전으로 만들어서 저장하는 곳.
2. **Remote repository** : 깃 허브에 있는 저장소. (여러 사람의 공동작업 저장소.)

Remote repository

# Github 저장소 생성하기.

The screenshot displays the GitHub homepage. At the top, there is a search bar with the placeholder text "Type / to search". Below this, a navigation bar contains a search input field labeled "Find a repository...", followed by filters for "Type", "Language", and "Sort". A green "New" button with a repository icon is highlighted with a red box and a circled "2". The main content area shows a list of repositories, including "TE11" (Public, updated 2 hours ago), "butter\_man" (Private, JavaScript, updated last week), and "sdjdjdj" (Public, updated 3 weeks ago). On the right side, a user profile sidebar for "Isuhan LeeSuHan" is visible. It includes options like "Edit status", "Your profile", "Your repositories" (highlighted with a red box and a circled "1"), "Your projects", "Your codespaces", "Your organizations", "Your enterprises", "Your stars", "Your sponsors", "Your gists", "Upgrade", and "Try Enterprise".

# 저장소 생성하기


## Create a new repository

A repository contains all the files for your project, including the revision history.

---

Owner

Repository name


 kimeunyoung01 ▾

 /


Great repository names are short and memorable. Need inspiration? How about **sturdy-enigma**.

Description (optional)

---

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**


You choose who can see and commit to this repository.

---

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

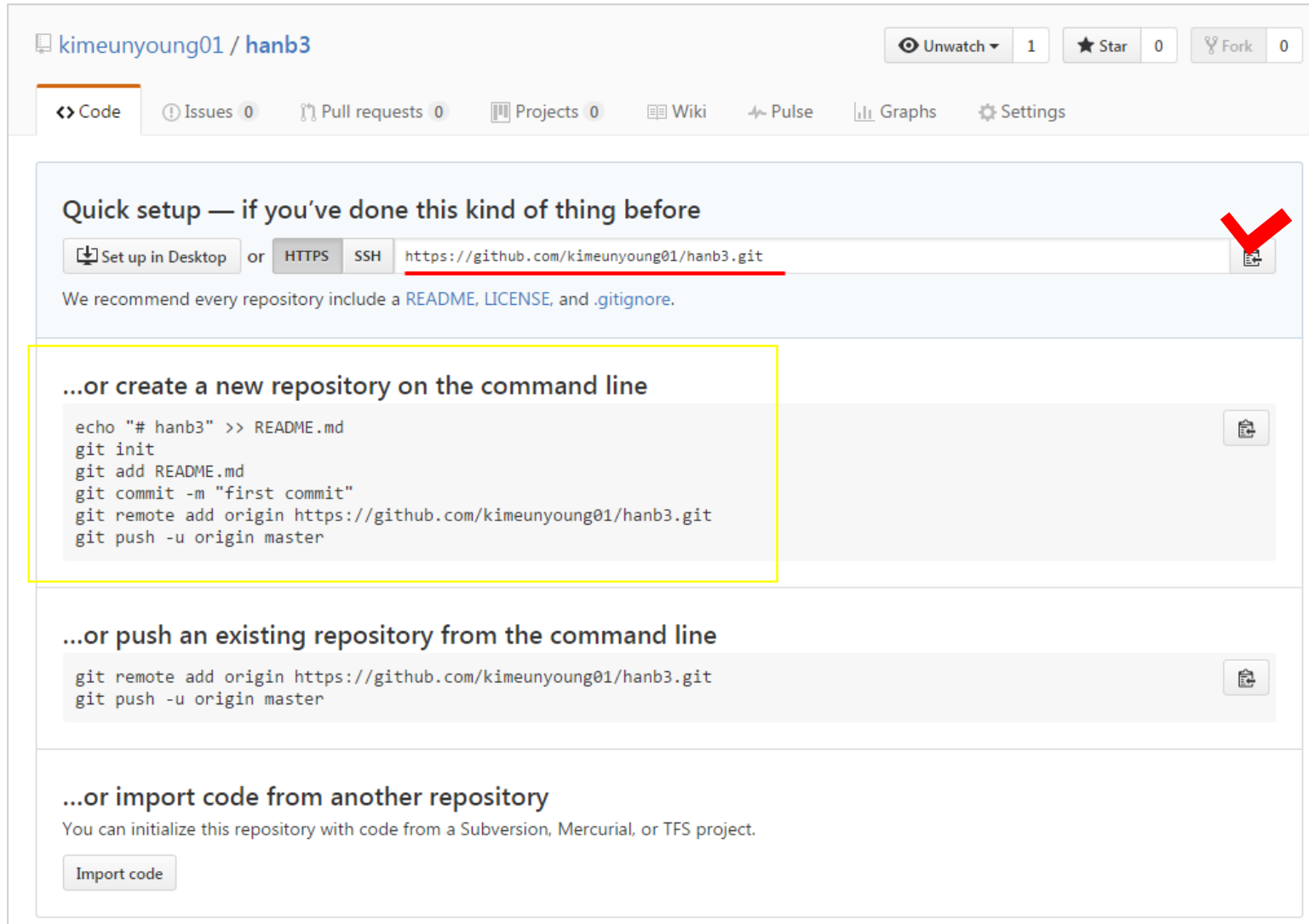
Add .gitignore: **None** ▾

Add a license: **None** ▾ 

---

**Create repository**

# Git URL 가져오기



The screenshot shows the GitHub interface for the repository `kimeunyoung01 / hanb3`. At the top, there are buttons for `Unwatch` (1), `Star` (0), and `Fork` (0). Below the repository name, there are tabs for `Code`, `Issues` (0), `Pull requests` (0), `Projects` (0), `Wiki`, `Pulse`, `Graphs`, and `Settings`.

The main content area is titled **Quick setup — if you've done this kind of thing before**. It features a `Set up in Desktop` button, an `or` separator, and buttons for `HTTPS` and `SSH`. The `HTTPS` button is selected, and the URL `https://github.com/kimeunyoung01/hanb3.git` is displayed in a text box. A red checkmark is placed to the right of the text box. Below the text box, there is a note: "We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#)."

Below the quick setup section, there is a section titled **...or create a new repository on the command line**, which is highlighted with a yellow border. It contains a code block with the following commands:

```
echo "# hanb3" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/kimeunyoung01/hanb3.git
git push -u origin master
```

Below this section, there is a section titled **...or push an existing repository from the command line**. It contains a code block with the following commands:

```
git remote add origin https://github.com/kimeunyoung01/hanb3.git
git push -u origin master
```

At the bottom, there is a section titled **...or import code from another repository**. It contains a note: "You can initialize this repository with code from a Subversion, Mercurial, or TFS project." and an `Import code` button.



# 원격 저장소와 로컬 저장소 연결 및 push

```
MINGW64:/d/git_test

user@leesuhan MINGW64 /d/git_test (master)
$ git remote add origin https://github.com/lsuhan/TE11.git

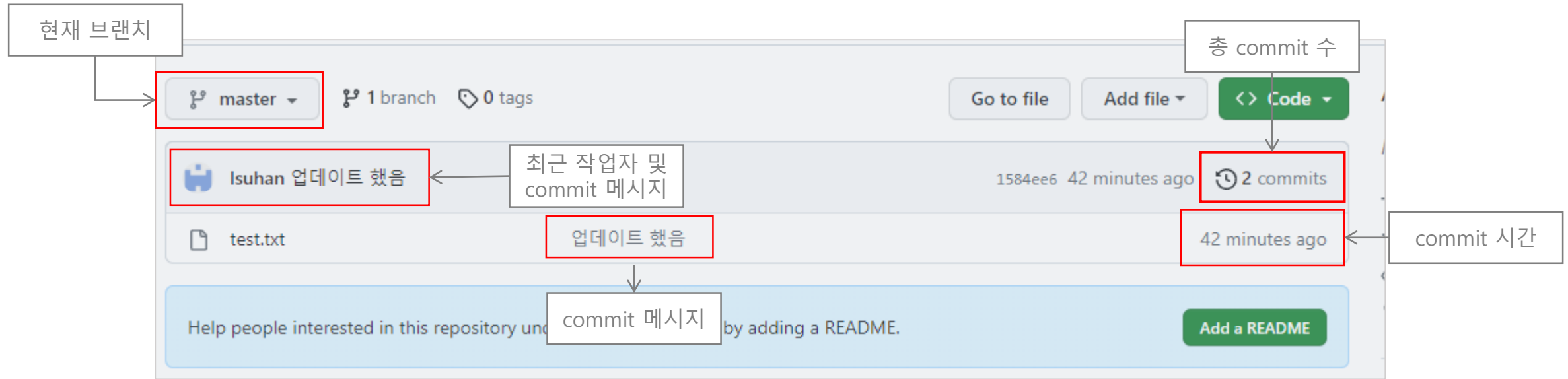
user@leesuhan MINGW64 /d/git_test (master)
$ git remote -v
origin https://github.com/lsuhan/TE11.git (fetch)
origin https://github.com/lsuhan/TE11.git (push)

user@leesuhan MINGW64 /d/git_test (master)
$ git status
On branch master
nothing to commit, working tree clean

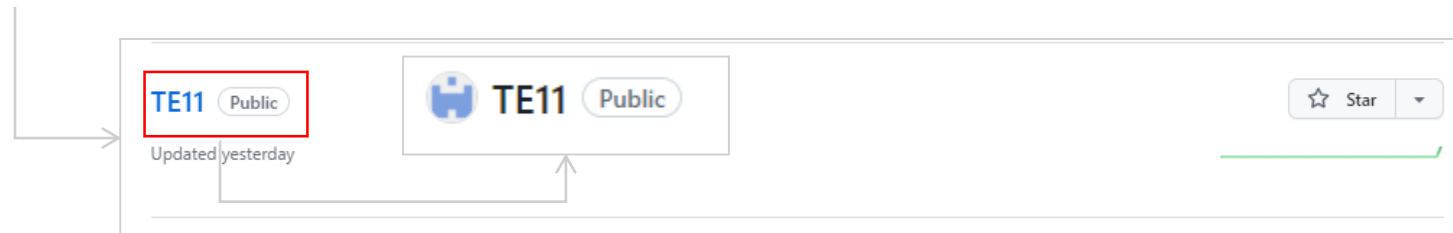
user@leesuhan MINGW64 /d/git_test (master)
$ git push origin master
git: 'credential-aws' is not a git command. See 'git --help'.
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 539 bytes | 539.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/lsuhan/TE11.git
 * [new branch]      master -> master
```

1. **git remote add origin [repository 주소]**: 원격 저장소를 추가
2. **git remote -v**: 로컬 저장소와 원격 저장소 연결 확인
3. **git push origin master**: master 브랜치에 commit 했던 내용들을 원격저장소에 올리기

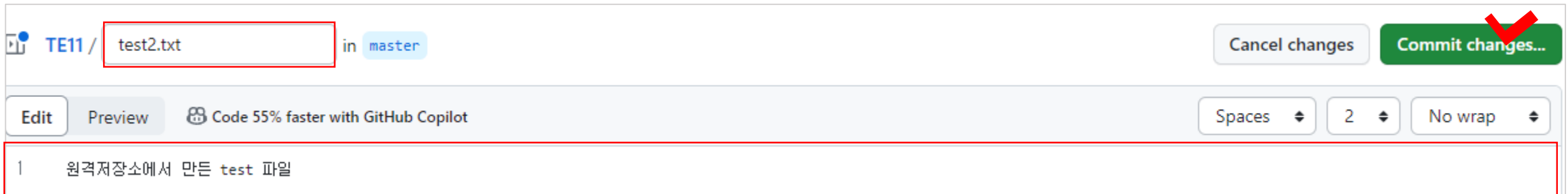
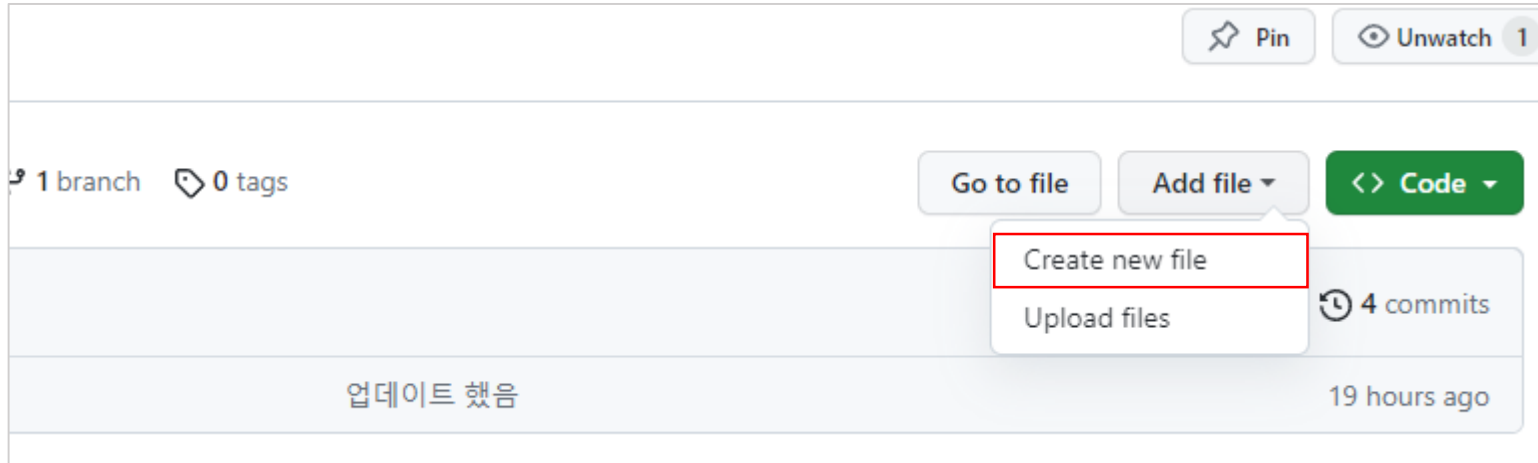
# Github에서 올린내용 확인하기



- 깃허브에 지금까지 로컬에서 작업한 test.txt 파일이 올라간 걸 확인 할 수 있다.
- 리포지토리가 public 으로 되어있을 경우 다른 사람들도 이 파일을 확인 할 수 있음.



# 원격저장소에서 작업트리로 pull 받기



# 원격저장소에서 작업트리로 pull 받기

Isuhan Merge branch 'master' of <a href="https://github.com/lsuhan/TE11">https://github.com/lsuhan/TE11</a> ... fb59002 2 minutes ago 7 commits		
test.txt	수정	4 minutes ago
test2.txt	Create test2.txt	16 minutes ago

```
user@leesuhan MINGW64 /d/git_test (master)
$ git pull origin master
git: credential-aws is not a git command. See 'git --help'.
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 633 bytes | 633.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/lsuhan/TE11.git
c4c44ee..fb59002 master -> master
```

## <내 디렉터리>

test.txt	2023-10-05 오전 9:01	텍스트 문서	1KB
test2.txt	2023-10-05 오전 9:03	텍스트 문서	1KB

- **git pull origin master**: 원격저장소에 있는 문서를 내 디렉터리로 가져오기

# 주요 Git 명령어

## → **-h -help**

- 깃 명령어 리스트를 출력한다.

## → **clone [git\_url]**

- 현재 파일 위치에 원격 깃 프로젝트를 복사한다.

## → **init**

- 현재 파일 위치에 로컬 저장소를 만든다.

## → **add [파일 위치]**

- 주어진 파일, 폴더들을 깃 변경사항 인덱스로 적재한다.

## → **status**

- 현재 깃 상태를 보여준다 (branch, 파일 상태 등...)

## → **add [파일 위치]**

- 주어진 파일, 폴더들을 깃 변경사항 인덱스로 적재한다.

## → **branch [브랜치 이름]**

- 현재 브랜치에서 새로운 브랜치를 생성한다.

## → **commit [-m 메시지]**

- add로 등록한 깃 데이터를 커밋한다.

## → **checkout [-b 브랜치 이름]**

- 저장된 브랜치로 이동한다. (파일 구조가 해당 브랜치로 변경된다.)

## → **merge [-b 브랜치 이름]**

- 현재 파일 위치에 원격 깃 프로젝트를 복사한다.

## → **init**

- 현재 파일 위치에 로컬 저장소를 만든다.

## → **fetch**

- 다른 브랜치, 저장소에 있는 정보를 다운로드 받는다.

## → **pull [저장소이름] [브랜치이름]**

- 다른 브랜치에 있는 내용을 로컬로 다운로드 받는다.

## → **push [저장소이름] [브랜치이름]**

- 커밋 내용을 해당 브랜치에 적용한다.

## → **reset**

- 파일 상태를 현재 브랜치 기준으로 완전히 바꾼다. (파일 유실에 주의한다)

# 실습

1. 드라이버에 git\_test2를 만들어 깃 저장소를 만든다.  
(git\_test2 폴더 생성 -> 깃 저장소에 연결)
2. work.txt 를 commit 해본다.
3. 원격 저장소를 만들어 work.txt를 push 해본다.

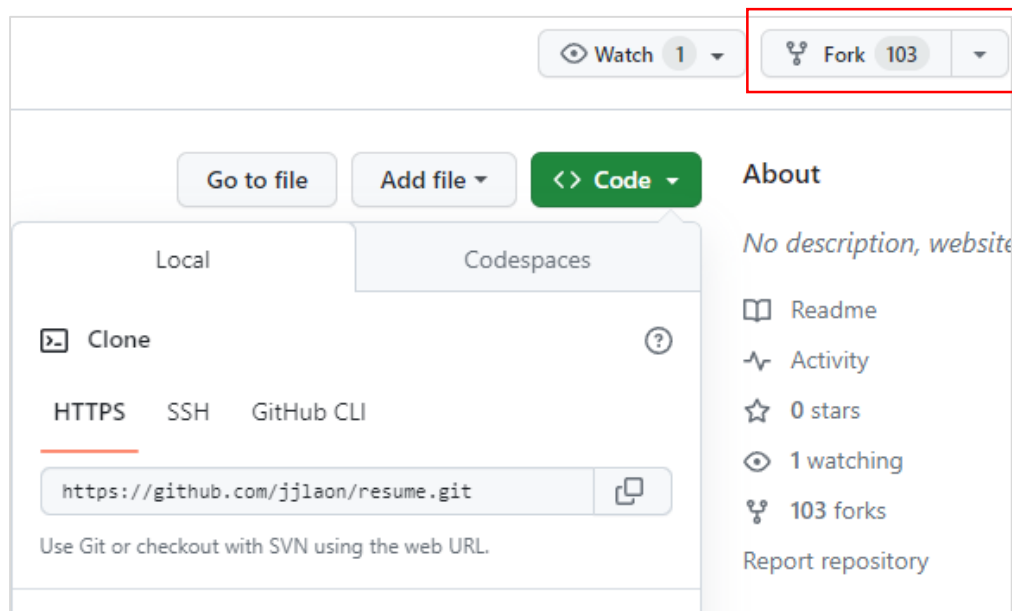
# 코드편집기에서의 깃 사용 (VS CODE)

[https://code.visualstudio.com/d  
ownload](https://code.visualstudio.com/download)

# 예제 포트폴리오 Fork

<https://github.com/jjlaon/resume>

✓우측 상단에 Fork (프로젝트 퍼오기)



### Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (\*).

Owner \*

Repository name \*

Choose an owner ▾

/ my\_portfolio

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

나의 포트폴리오

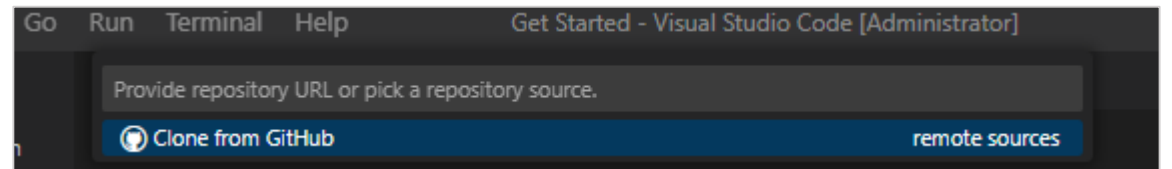
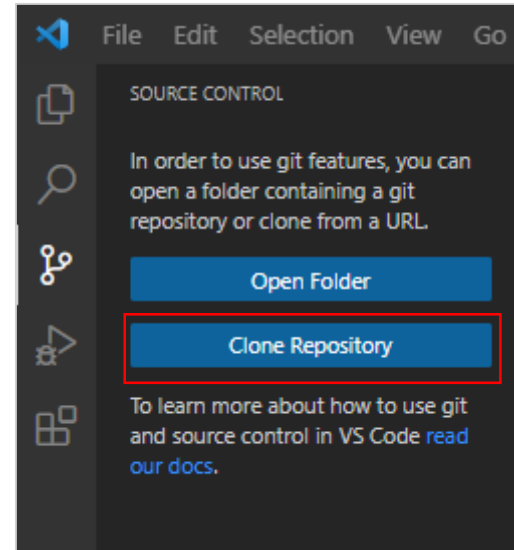
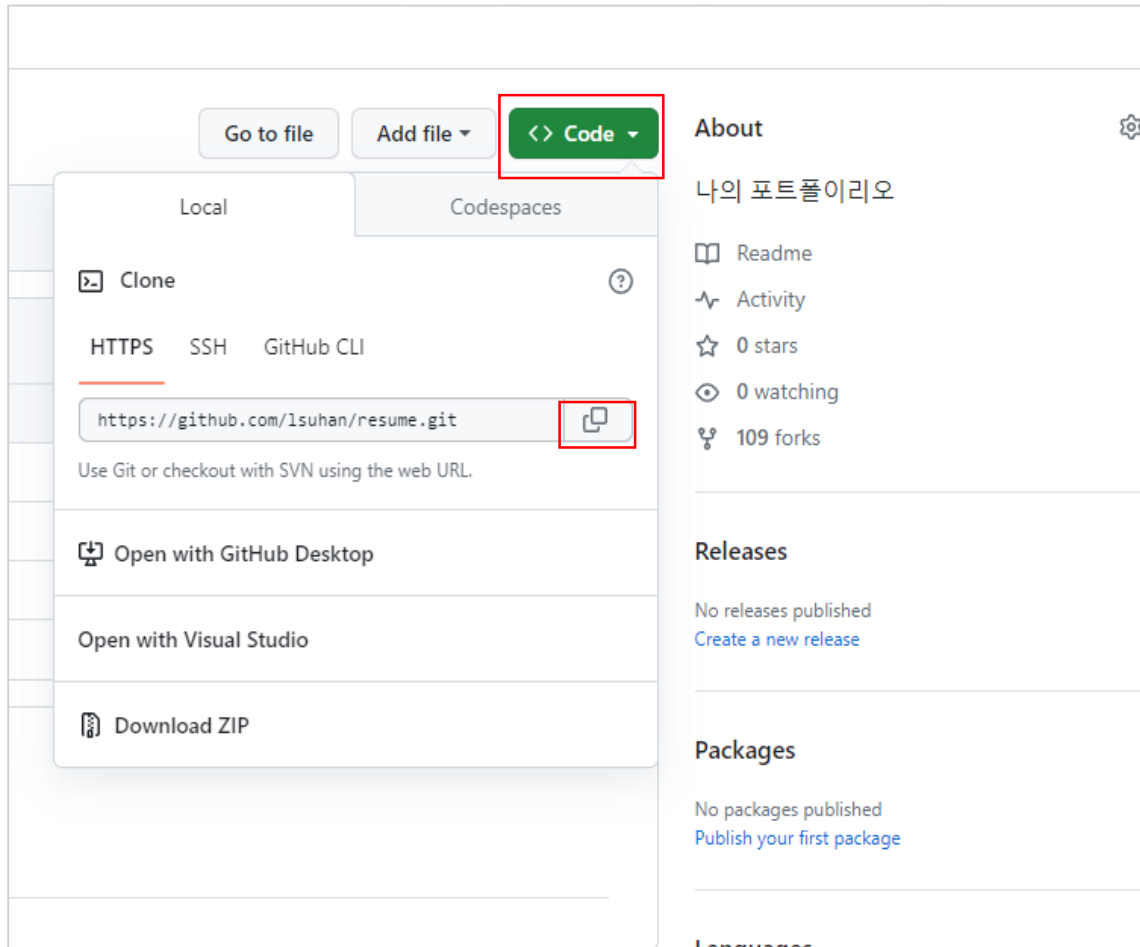
☒ Copy the `main` branch only

Contribute back to jjlaon/resume by adding your own branch. [Learn more.](#)

Create fork



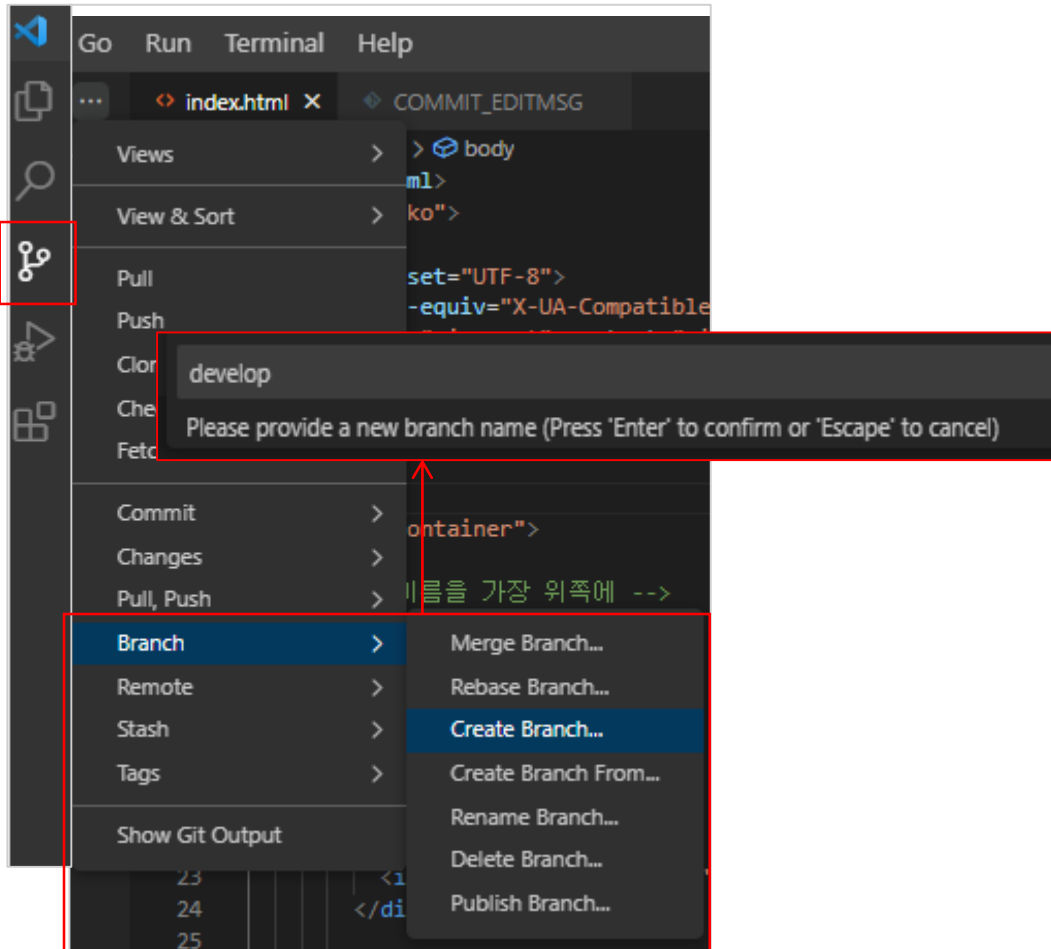
# VS CODE 에 Clone



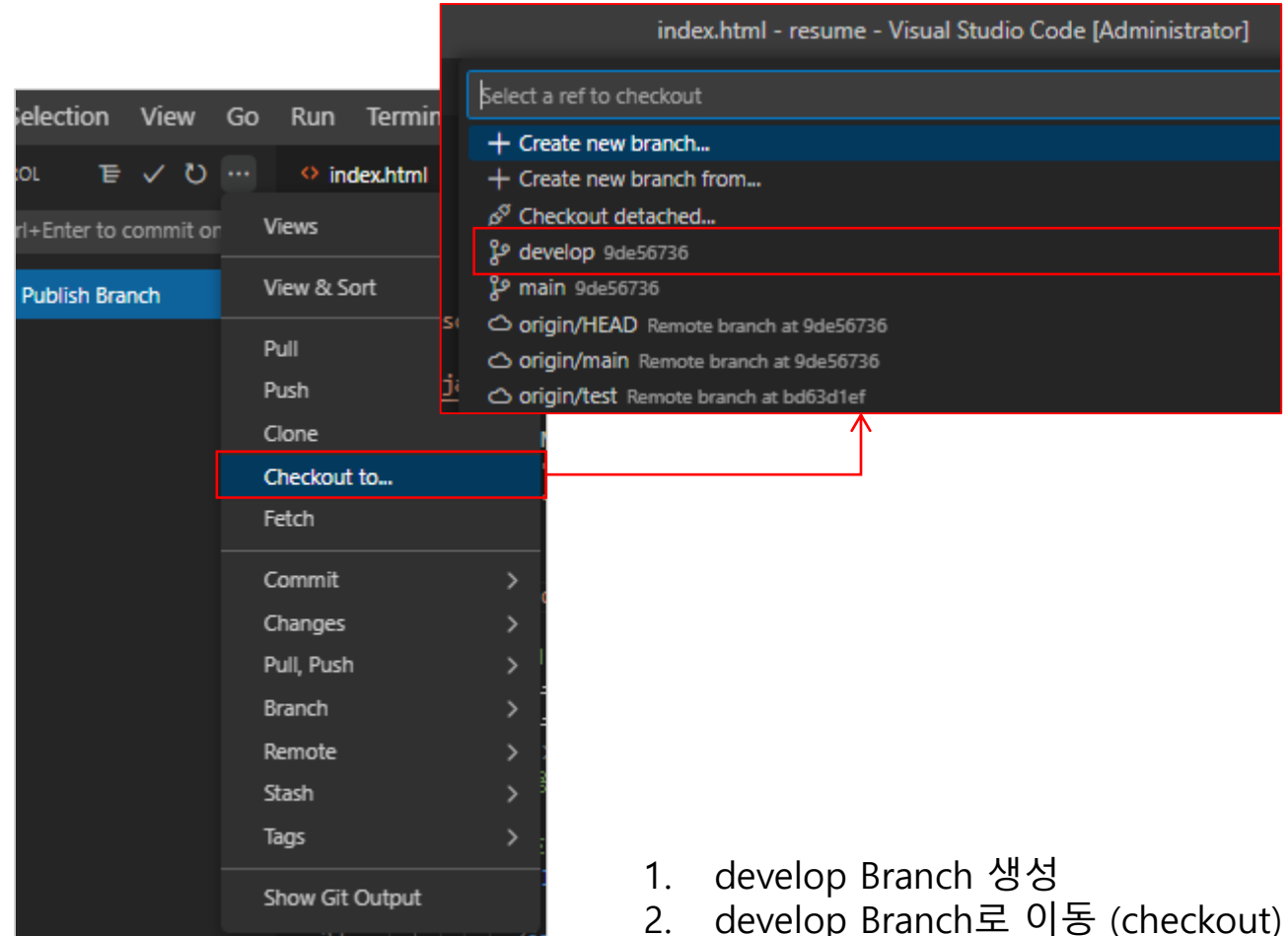
깃허브 리파지토리에 있는 소스를 내 개인 디렉토리로 복사

# Branch 생성 및 이동하기

## 1. Branch 생성하기

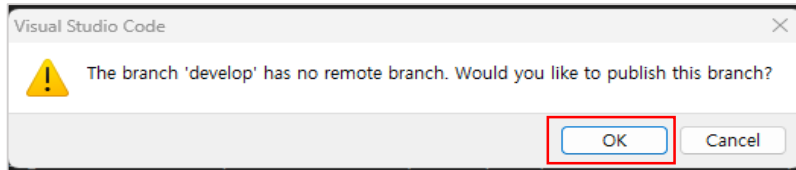


## 2. Branch 이동하기

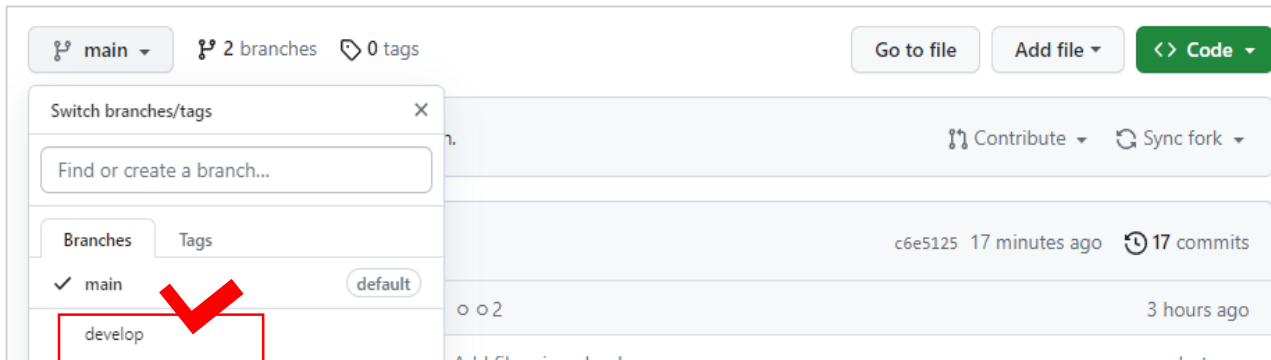


1. develop Branch 생성
2. develop Branch로 이동 (checkout)
3. index.html 파일 수정 후 commit, push

# 생성한 Branch github에서 확인하기



1. develop 라는 Branch를 원격 저장소와 연결.

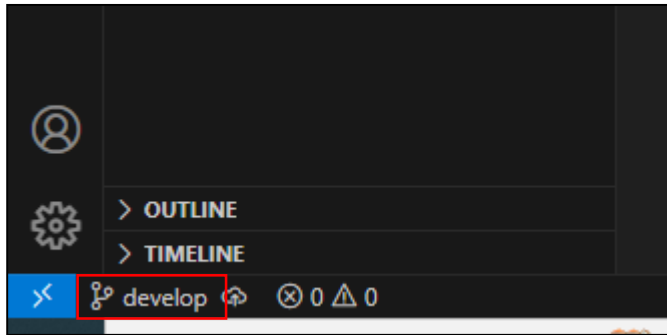


2. github에서 develop branch 생성 확인  
3. develop branch로 checkout 후 변경된 내용 확인

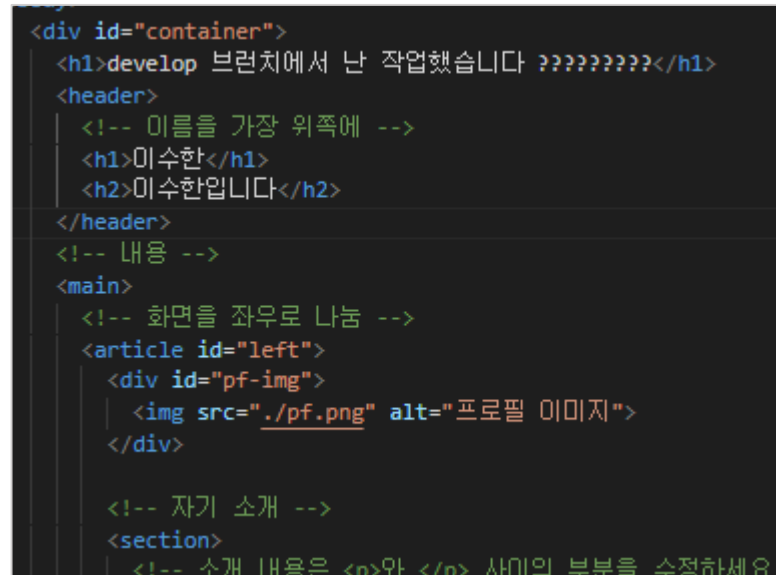
+ main branch랑 비교해보기

# develop branch에서 작업해보기

## 1. 현재 branch 확인하기

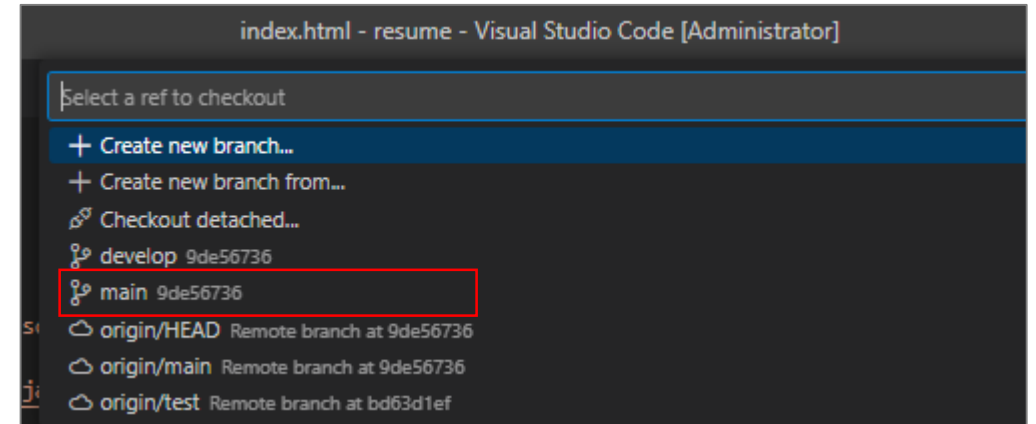
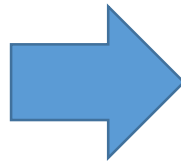
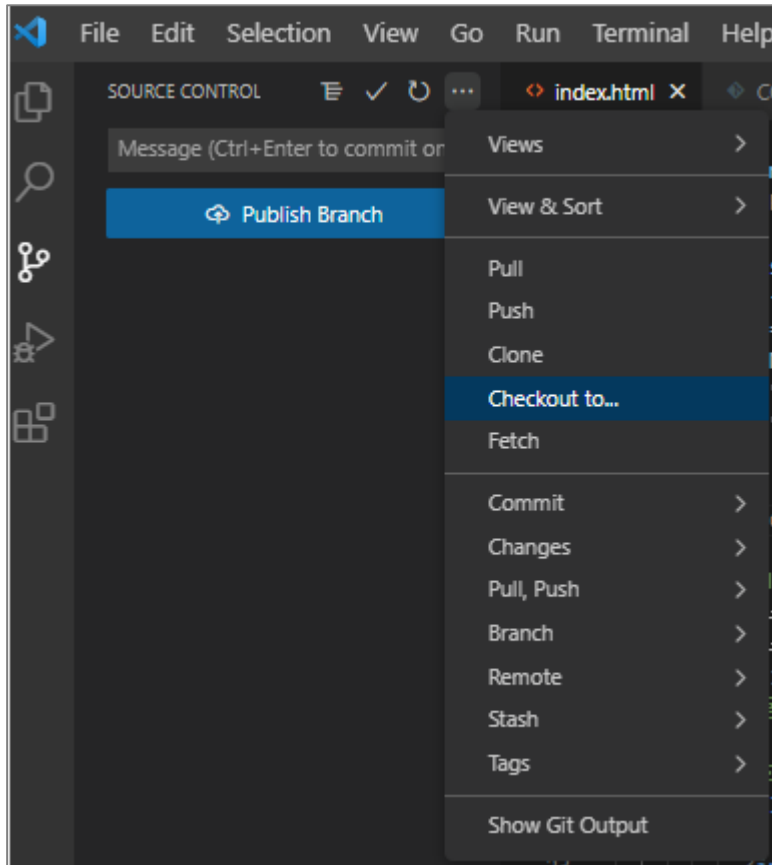


## 2. 수정하기



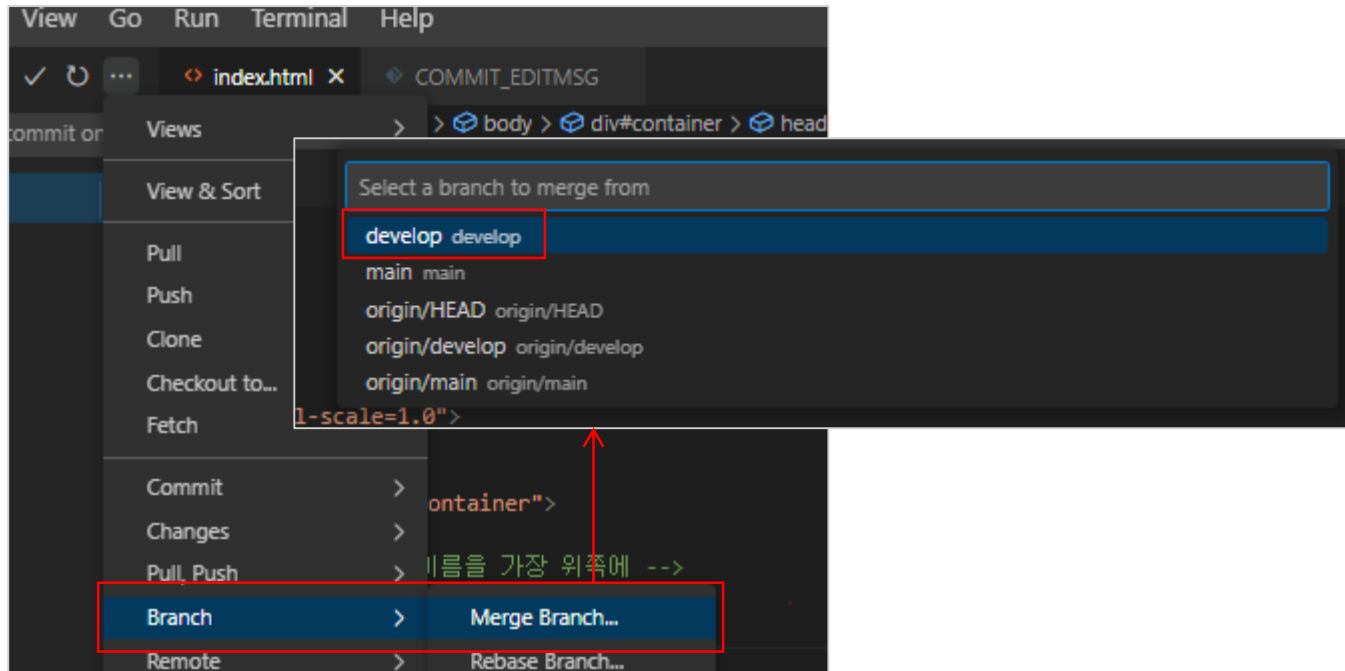
# main branch에서 develop 작업내역 병합(1)

1. Main branch로 이동하기 (checkout)



# main branch에서 develop 작업내역 병합(2)

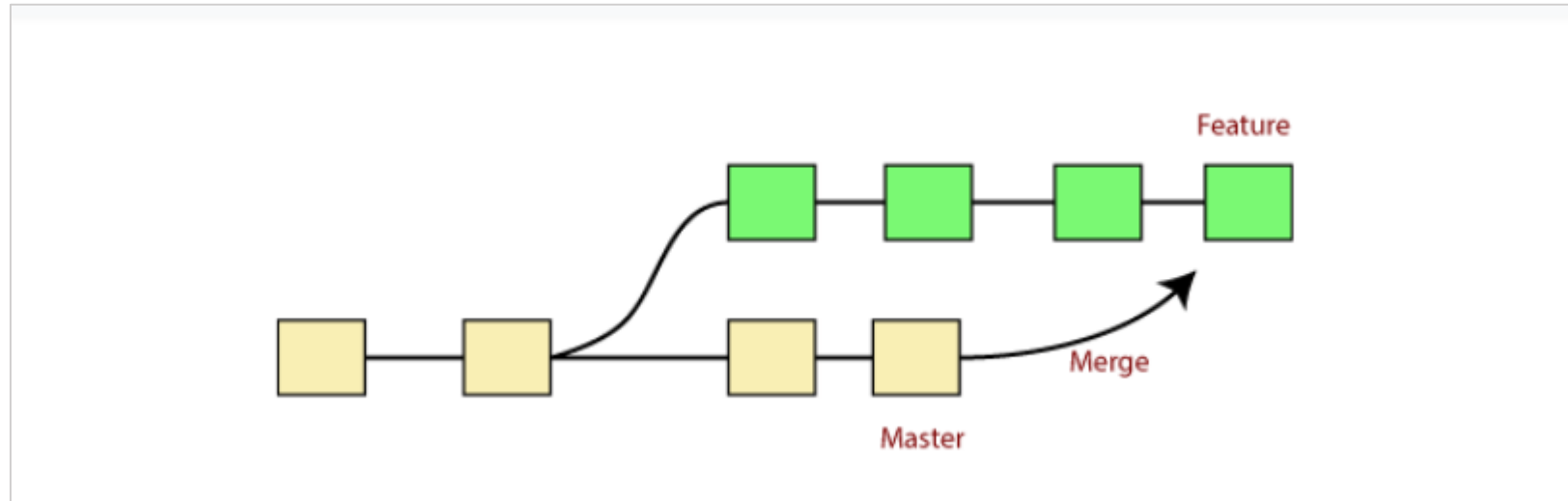
## 2. develop branch 병합하기 (develop에서 작업한 내용을 main branch에 반영하기)



```
<div id="container">
  <h1>develop 브랜치에서 난 작업했습니다 ?????????</h1>
  <header>
    <!-- 이름을 가장 위쪽에 -->
    <h1>이수한</h1>
    <h2>이수한입니다</h2>
  </header>
  <!-- 내용 -->
  <main>
    <!-- 화면을 좌우로 나눔 -->
    <article id="left">
      <div id="pf-img">
        
      </div>
    </article>
    <!-- 자기 소개 -->
    <section>
      <!-- 소개 내용은 <p>와 </p> 사이의 부분을 수정하세요
```

## 3. develop branch에서 작업했던 내용들이 main branch에 병합되었는지 확인하기

# MERGE



- Feature 브랜치로 생성해 작업한 내용들을 다른 브랜치에 병합 할 수 있다.
- 각 줄기로 뻗어져 있는 feature 브랜치들을 하나의 브랜치에서 통합하여 소스를 관리 할 수 있다.

# Git 브랜치 전략

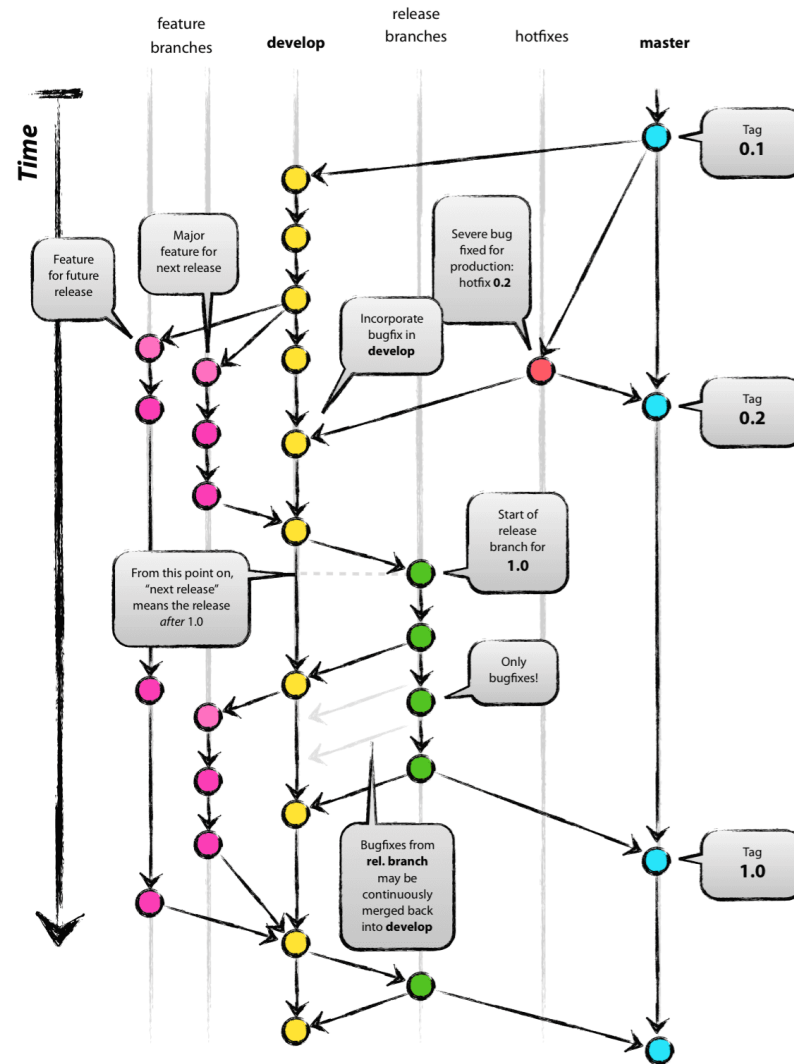


# Git 브랜치 전략

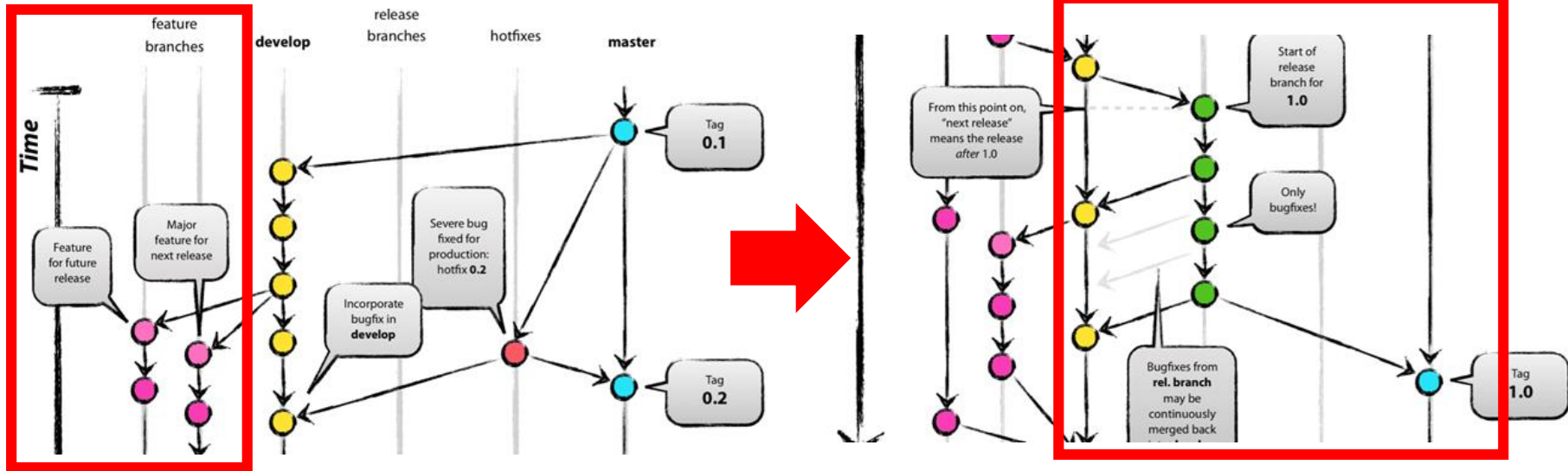
- **브랜치 전략**
  - 브랜치 전략이란 여러 개발자가 하나의 저장소를 사용하는 환경에서 저장소를 효과적으로 활용하기 위한 **work-flow**다.
  - 브랜치의 생성, 삭제, 병합 등 git의 유연한 구조를 활용해서, 각 개발자들의 혼란을 최대한 줄이며 다양한 방식으로 소스를 관리하는 역할을 한다.
  - 즉, **브랜치 생성에 규칙을 만들어서 협업을 유연하게 하는 방법론**을 말한다.
  - 정해진 것은 없으나, 개발자들이 내부적으로 합의한 것이다.
- **만약 브랜치 전략이 없다면**
  - 작업자가 서로 다른 이름의 브랜치를 무수히 생성함으로써 현재 작업중인 버전에 대한 혼란이 온다.
  - 중간에 투입되는 작업자가 브랜치를 보고 식별이 불가능해진다.
  - 배포 버전 선정에 어려움이 온다.
  - 실질적인 이력 관리가 불가능해진다.
- **브랜치 전략에는 git-flow, Github-flow 전략이 있다.**

# Git 플로우 전략

- 기본적인 브랜치의 이름은 다음과 같이 정의한다.
  - feature > develop > release > hotfix > master
- **feature**
  - 추가 기능 개발 브랜치로 Develop 브랜치에 들어간다
- **develop**
  - 다음 출시 버전을 대비하여 개발하는 브랜치로 feature들을 모으는 역할을 한다.
- **release**
  - 다음 버전 출시를 대비하는 브랜치로 develop 브랜치를 release 브랜치로 옮긴 후 QA, 테스트를 진행한다. 테스트 서버에 올라가는 버전이 들어간다. (develop과 통합하여 사용하기도 한다.)
- **master / main**
  - 라이브 서버에 실제 제품 / 서비스로서 출시되는 브랜치를 말한다.
- **hotfix**
  - master 브랜치에서 발생한 버그를 수정하여 반영하는 브랜치를 의미한다.



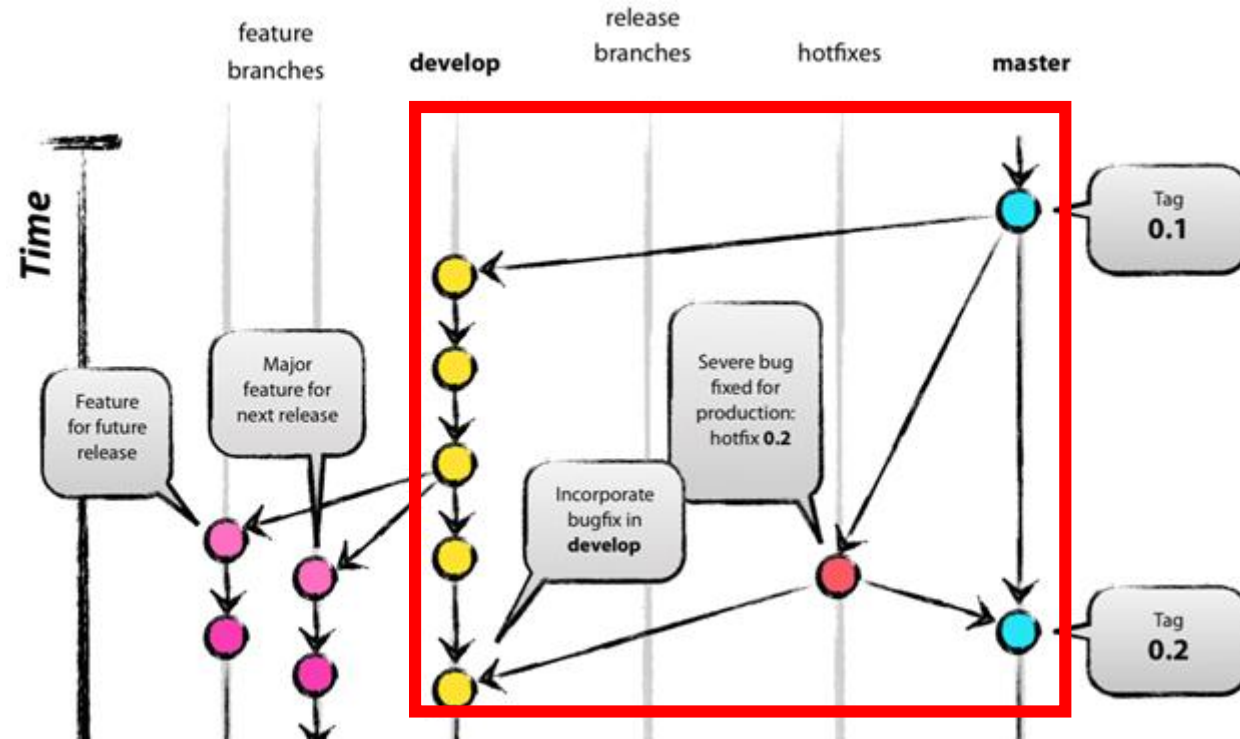
# 신규 기능 개발 시



- 개발자는 develop 브랜치로 부터 신규 개발을 위한 feature 브랜치를 생성한다.
- feature 브랜치에서 기능을 완성하면 develop 브랜치에 merge를 진행한다.

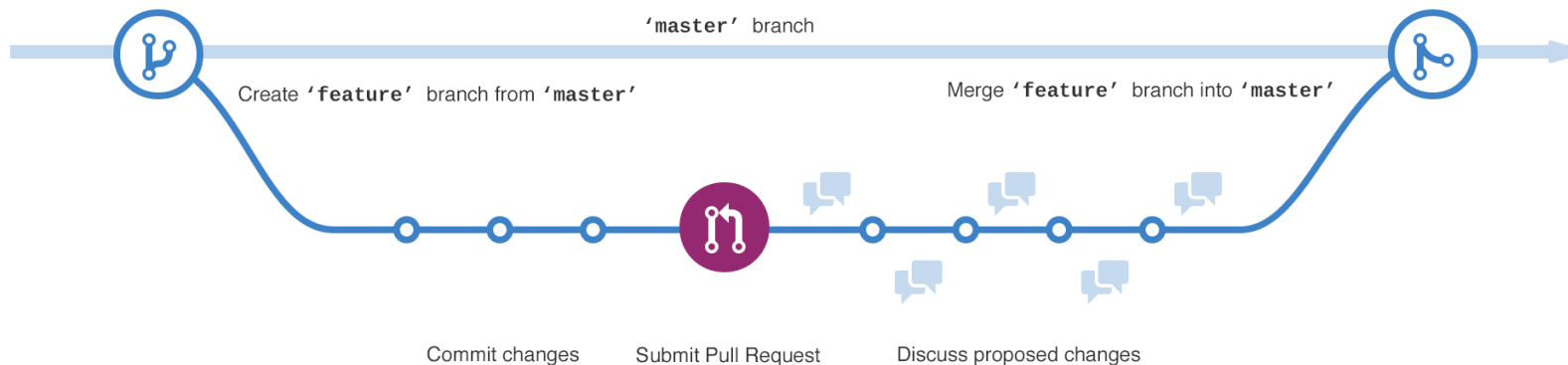
- develop으로 부터 release 브랜치를 생성한다.
  - 개발 서버는 release 브랜치 코드를 기반으로 구축한다.
- release 내부에서 발생한 오류는 release에서 수정한다.
- 테스트 종료 시 Release 브랜치를 Master 브랜치로 Merge 한다.

# 배포 이후 관리



- 배포된 라이브 서버(master)에서 버그가 발생하면, hotfix 브랜치를 생성하여 버그 픽스를 진행한다.
- 버그픽스가 종료되면 master와 develop 양 쪽에 데이터를 Merge하여 동기화한다.

# GitHub-flow 전략



- github-flow는 git flow의 브랜치 생성이 많고 복잡하며, 적용이 어려움에 따라 생겨난 브랜치 전략을 말한다.
- github-flow는 master 브랜치 하나만을 가지고 진행되는 방식을 말한다.
- master 브랜치에 모든 기능과 오류 수정을 merge하여 항상 master가 업데이트 되는 상태를 유지한다.
- GitHub-flow 전략 과정
  - master 브랜치에서 개발을 시작한다.
  - 신기능, 버그 발생 시 issue를 작성한다.
  - 팀원들이 issue를 해결하기 위해 feature 브랜치를 작성(단 하나!)해서 작업을 진행한다.
  - 팀원들이 feature에 commit 시 pull request를 날리고 이 과정에서 서로 코드를 확인하며 피드백, 버그 수정을 진행한다.
  - 모든 리뷰와 테스트가 종료되면 master 브랜치에 머징을 진행한다.

Github pages io 이용하기

# 포트폴리오 홈페이지 만들기

The screenshot shows the GitHub repository settings for 'Isuhan / resume'. The 'Settings' tab is selected in the top navigation bar. In the left sidebar, the 'Pages' option is highlighted. The main content area is titled 'GitHub Pages' and contains the following sections:

- General**
  - Access**
  - Collaborators**
  - Moderation options**
  - Code and automation**
    - Branches
    - Tags
    - Rules
    - Actions
    - Webhooks
    - Environments
    - Codespaces
    - Pages** (highlighted)
  - Security**
    - Code security and analysis
    - Deploy keys
    - Secrets and variables
  - Integrations**
    - GitHub Apps
    - Email notifications
- GitHub Pages**

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

**Build and deployment**

**Source**

Deploy from a branch

**Branch**

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more about configuring the publishing source for your site.](#)

None Save

**Visibility** **GITHUB ENTERPRISE**

With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. A privately published site can only be accessed by people with read access to the repository the site is published from. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise.

[Try GitHub Enterprise risk-free for 30 days](#) [Learn more about the visibility of your GitHub Pages site](#)

# 포트폴리오 홈페이지 만들기

## GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

### Build and deployment

Source

Deploy from a branch ▾

Branch

GitHub Pages is currently disabled. Select a source [configuring the publishing source for your site](#).

None ▾

Save

None ▾

Save

Select branch

Select branch

main

✓ None

main ▾

/ (root) ▾

Save

### Visibility

GITHUB ENTERPRISE

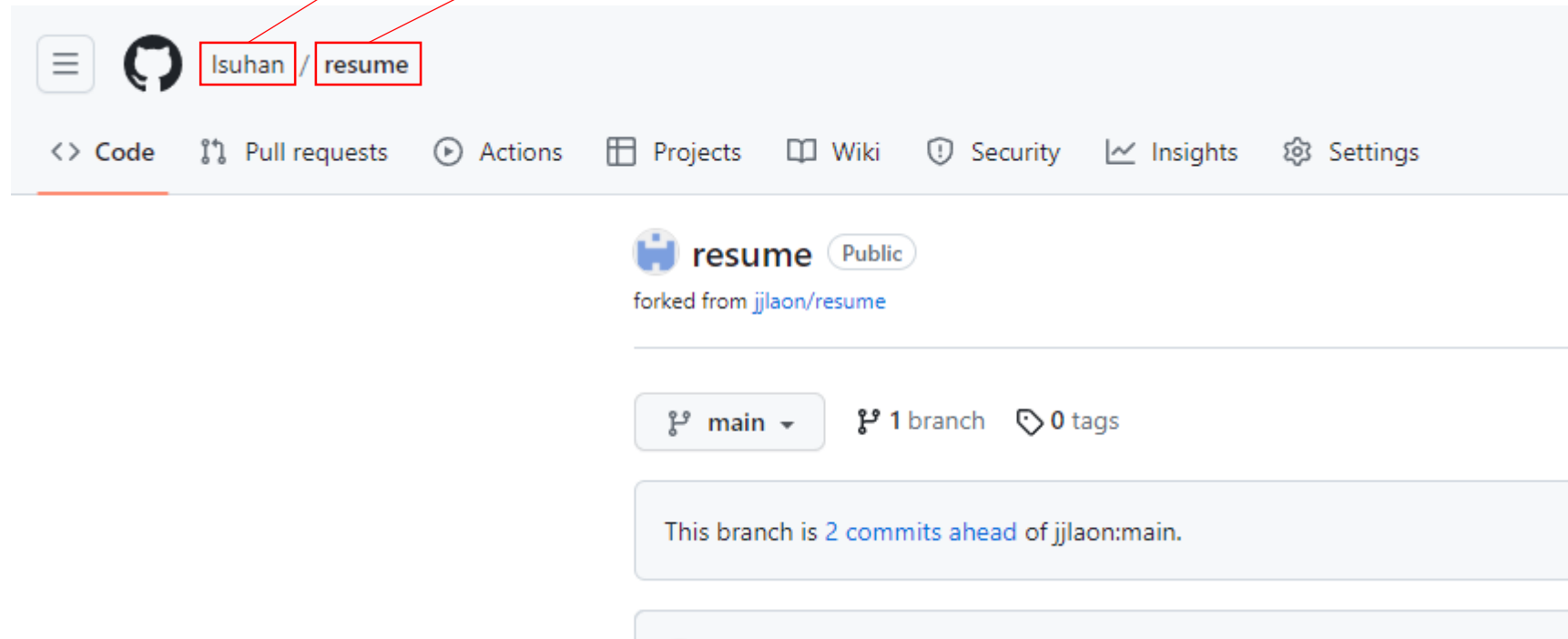
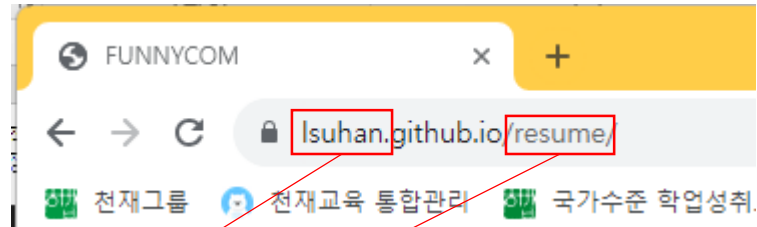
With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. A privately published site can only be accessed by people with read access to the repository the site is published from. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise.

Try GitHub Enterprise risk-free for 30 days

[Learn more about the visibility of your GitHub Pages site](#)




# 포트폴리오 홈페이지 만들기



SourceTree

# <https://www.sourcetreeapp.com/>



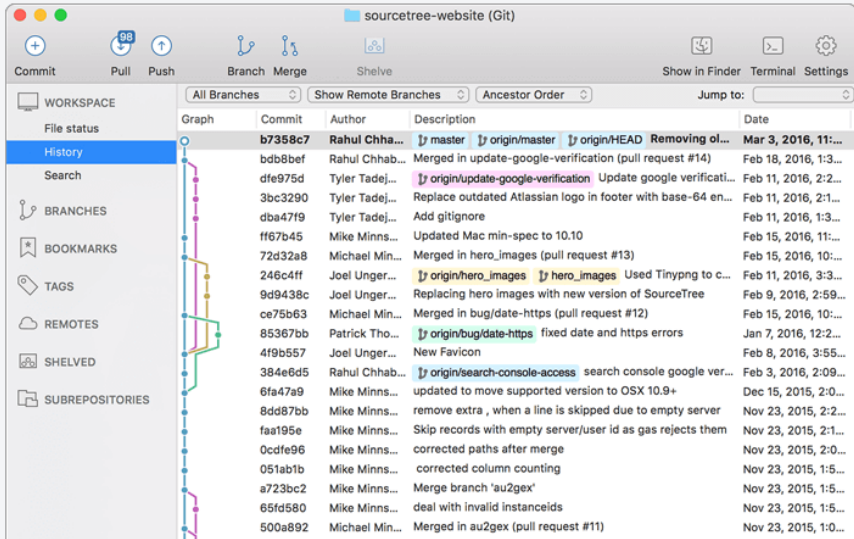
Download free

## Simplicity and power in a beautiful Git GUI

Download for Windows

Also available for Mac OS X

Latest release notes: [Mac OS X](#) & [Windows](#)

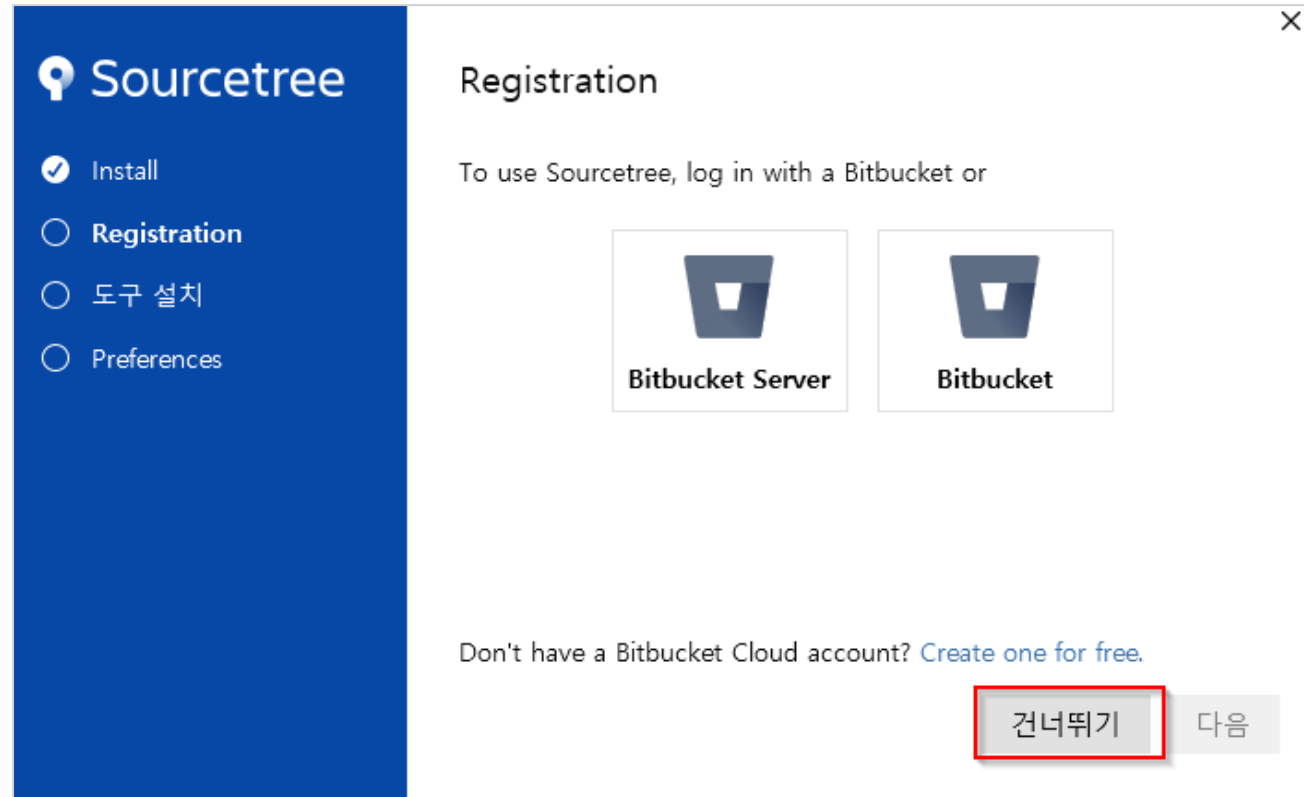


Commit	Author	Description	Date
b7358c7	Rahul Chha...	origin/master origin/HEAD Removing ol...	Mar 3, 2016, 11:...
bdb8bef	Rahul Chhab...	Merged in update-google-verification (pull request #14)	Feb 18, 2016, 1:3...
dfe975d	Tyler Tadej...	origin/update-google-verification Update google verificati...	Feb 11, 2016, 2:2...
3bc3290	Tyler Tadej...	Replace outdated Atlassian logo in footer with base-64 en...	Feb 11, 2016, 2:1...
dba47f9	Tyler Tadej...	Add glitgnore	Feb 11, 2016, 1:3...
ff67b45	Mike Minns...	Updated Mac min-spec to 10.10	Feb 15, 2016, 11:...
72d32a8	Michael Min...	Merged in hero_images (pull request #13)	Feb 15, 2016, 10:...
246c4ff	Joel Unger...	origin/hero_images hero_images Used Tinypng to c...	Feb 11, 2016, 3:3...
9d9438c	Joel Unger...	Replacing hero images with new version of SourceTree	Feb 9, 2016, 2:59...
ce75b63	Michael Min...	Merged in bug/date-https (pull request #12)	Feb 15, 2016, 10:...
85367bb	Patrick Tho...	origin/bug/date-https fixed date and https errors	Jan 7, 2016, 12:2...
4f9b557	Joel Unger...	New Favicon	Feb 8, 2016, 3:55...
384e6d5	Rahul Chhab...	origin/search-console-access search console google ver...	Feb 3, 2016, 2:09...
6fa47a9	Mike Minns...	updated to move supported version to OSX 10.9+	Dec 15, 2015, 2:0...
8dd87bb	Mike Minns...	remove extra , when a line is skipped due to empty server	Nov 23, 2015, 2:2...
faa195e	Mike Minns...	Skip records with empty server/user id as gas rejects them	Nov 23, 2015, 2:1...
0cdf96	Mike Minns...	corrected paths after merge	Nov 23, 2015, 2:0...
051ab1b	Mike Minns...	corrected column counting	Nov 23, 2015, 1:5...
a723bc2	Mike Minns...	Merge branch 'au2gex'	Nov 23, 2015, 1:5...
65fd580	Mike Minns...	deal with invalid instanceids	Nov 23, 2015, 1:5...
500a892	Michael Min...	Merged in au2gex (pull request #11)	Nov 23, 2015, 1:0...

## A free Git client for Windows and Mac

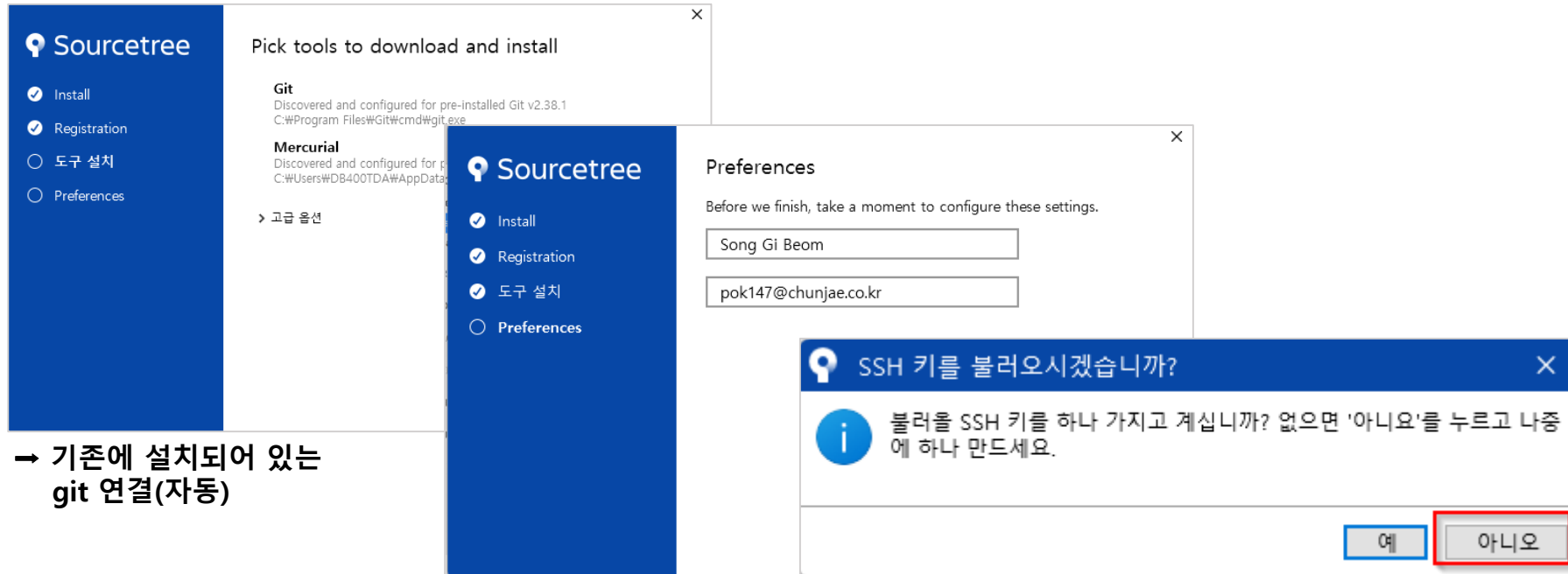
Sourcetree simplifies how you interact with your Git repositories so you can focus on coding. Visualize and manage your repositories through Sourcetree's simple Git GUI.

# 소스트리 설치



→ BitBucket 계정이 없으므로 건너뛰다

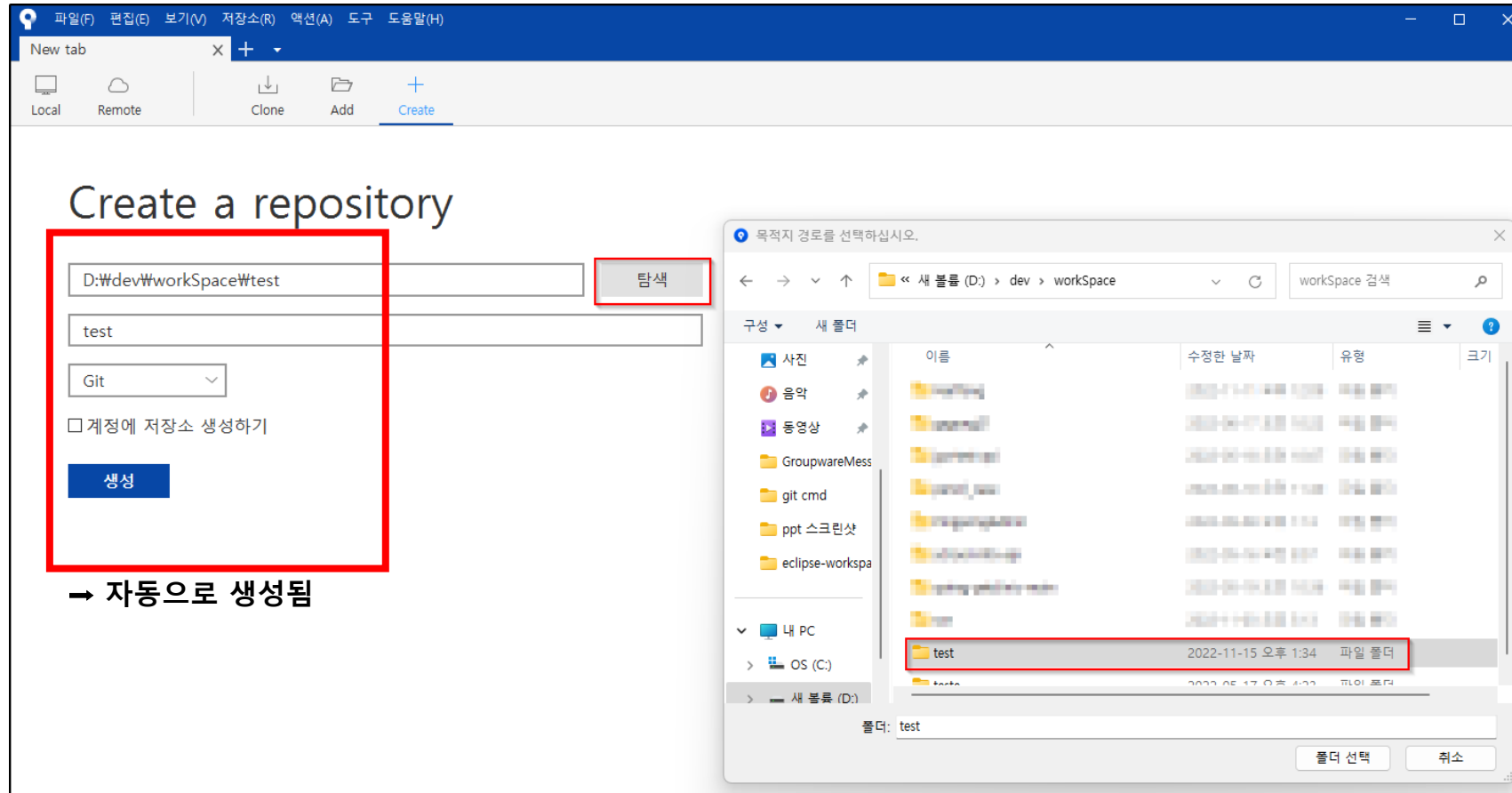
# 소스트리 설치



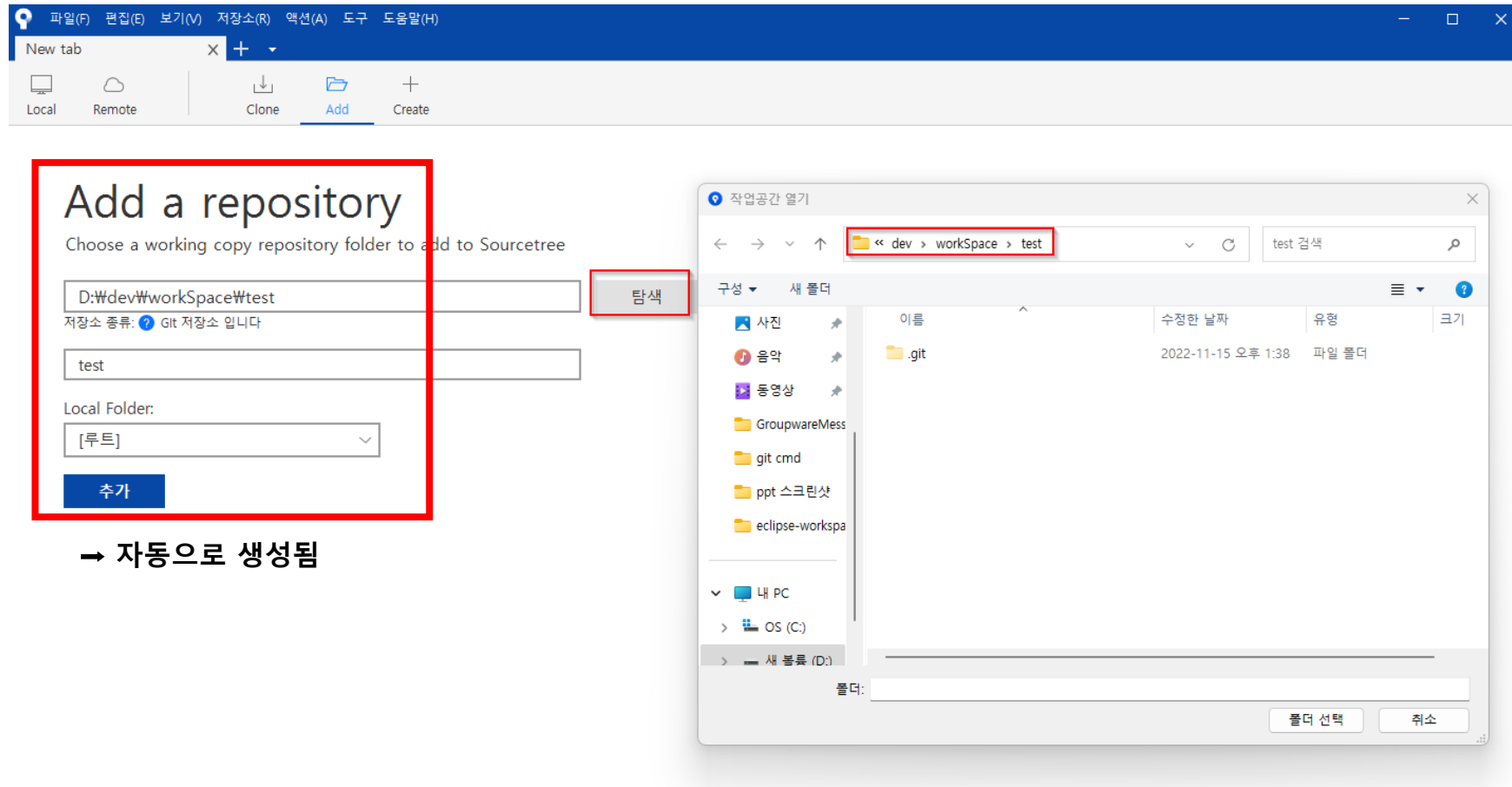
→ 기존에 설치되어 있는  
git 연결(자동)

→ 깃 환경에 있는 이름 / 이메일 등록  
- 초기에 깃 연동 시 나왔던 이름 / 메일이 등록되어 있음

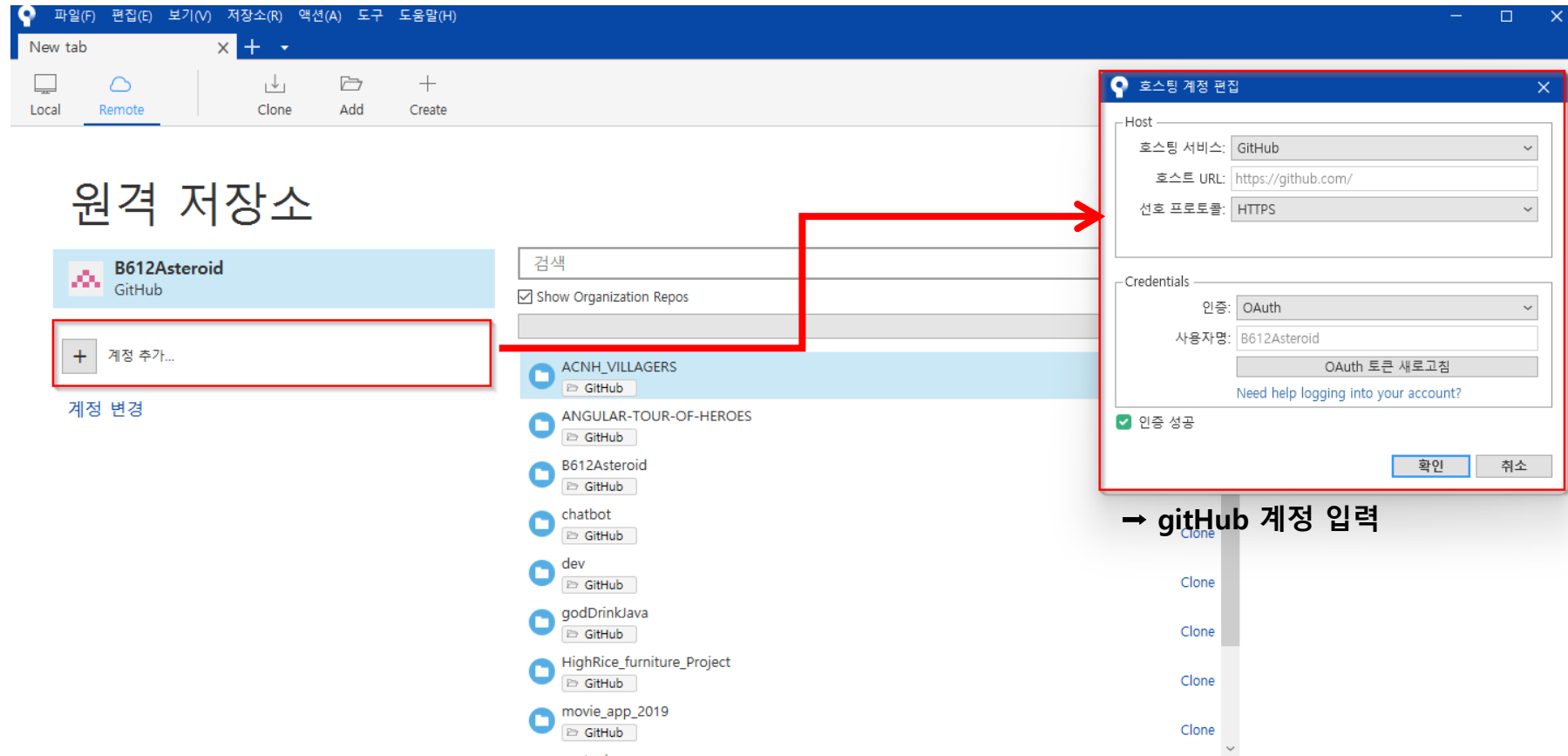
# 새로운 Repository 만들기



# 기존 Local 저장소 연결



# gitHub 저장소 연결

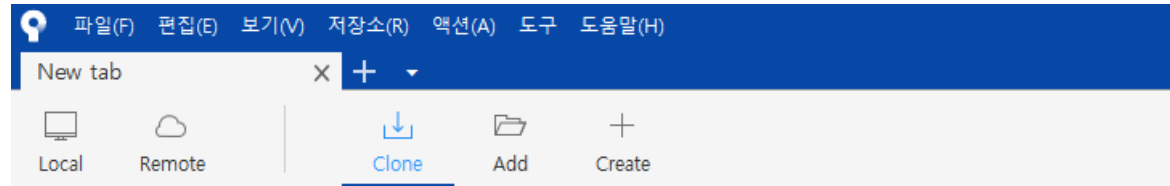


→ gitHub 계정 입력

→ 가지고 있는 gitHub Repository 중 선택하여 Clone



# gitHub 저장소 연결



## Clone

Cloning is even easier if you set up a [remote account](#)

→ Remote에서 왔을 경우 자동 생성

ANGULAR-TOUR-OF-HEROES

→ 저장할 로컬 폴더 지정

Local Folder:

[루트]

> 고급 옵션

클론