

Git & GitHub

◆ 깃허브로 백업하기

정수아

Contents

01 원격 저장소

02 깃허브란?

03 지역 저장소를 원격 저장소에 연결하기



01

원격 저장소

원격 저장소

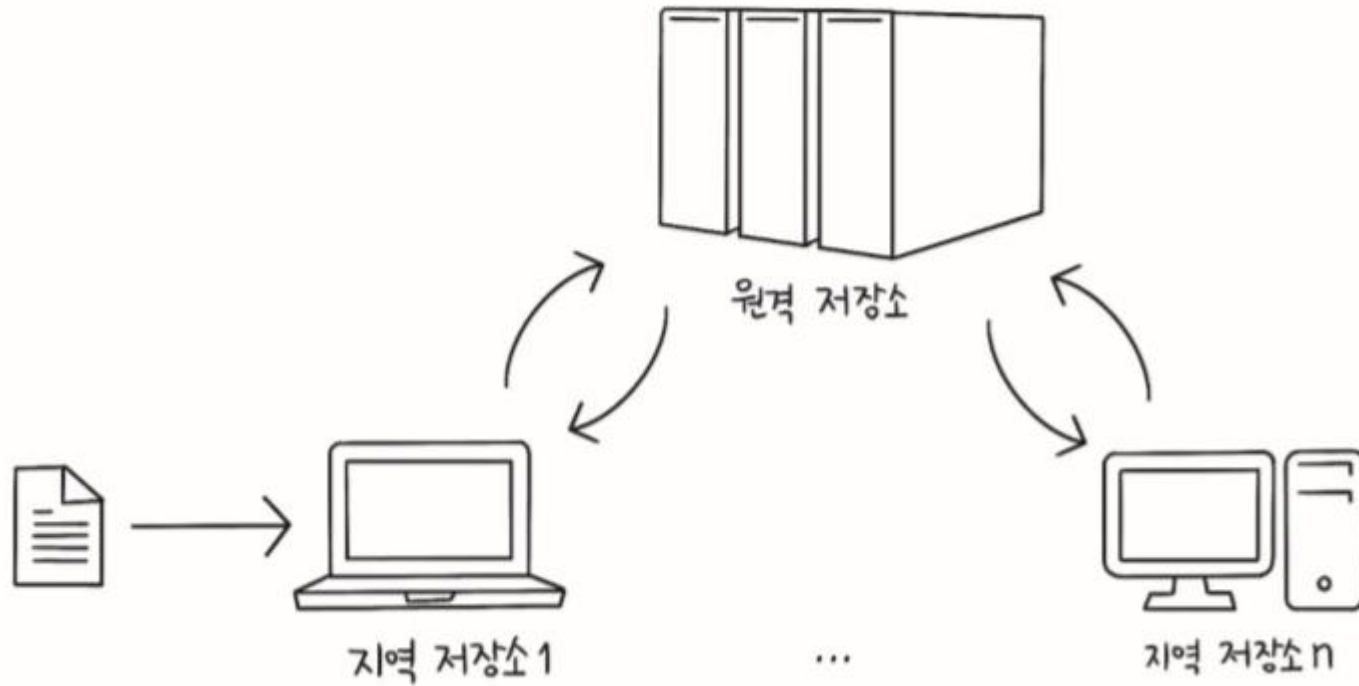
❖ 지역 저장소(local repository)

- 작업을 수행한 후 커밋을 저장한 컴퓨터

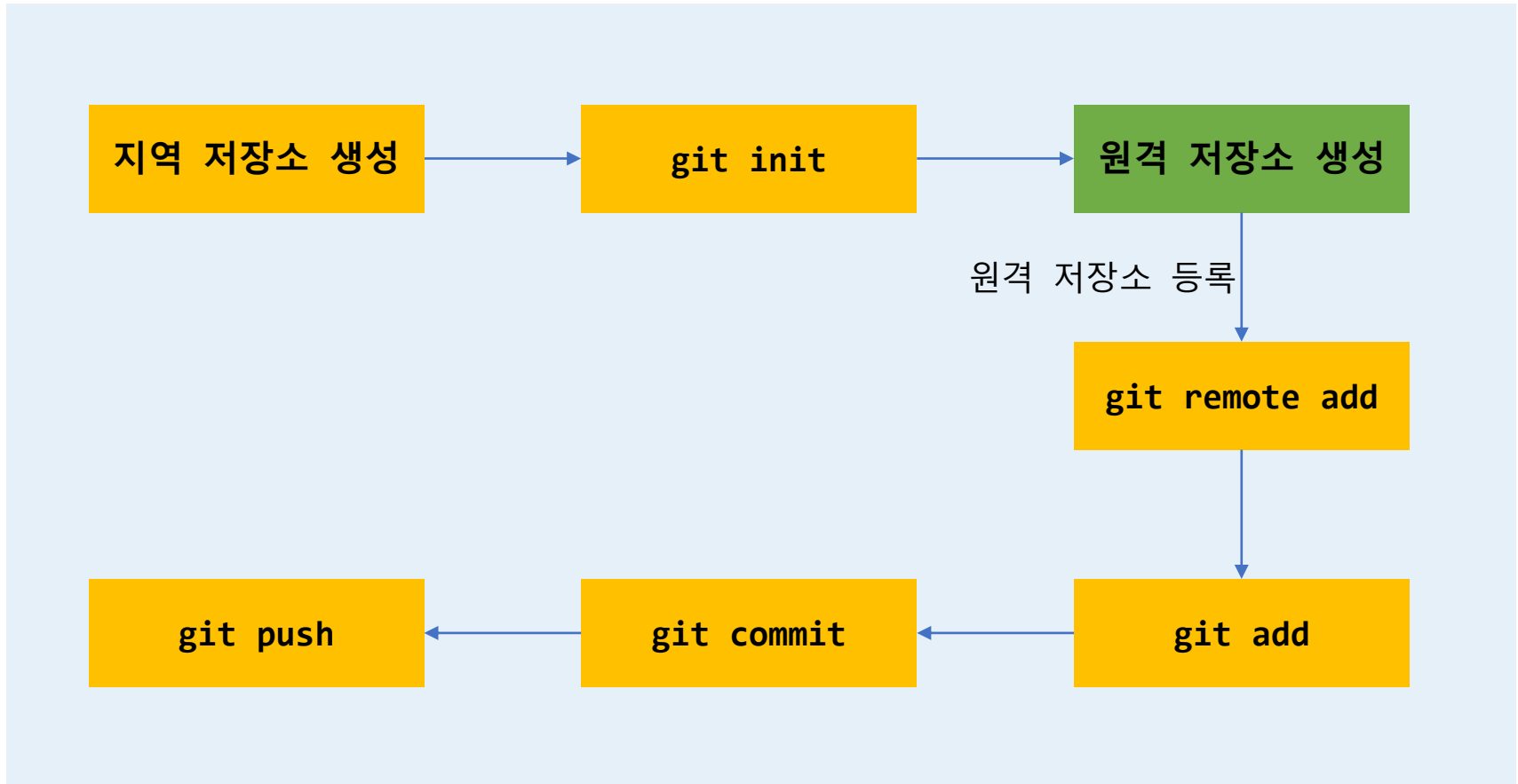
❖ 원격 저장소(remote repository)

- 지역 저장소가 아닌 컴퓨터나 서버에 만든 저장소
- 지역 저장소와 연결되어 있으면서 백업 및 협업을 가능하게 함
- 인터넷에서 원격 저장소를 제공하는 서비스를 주로 사용
 - 깃허브

원격 저장소



원격 저장소





02

깃허브란?

깃허브란?

❖ 깃허브(GitHub)

- 깃과 관련해서 가장 많이 사용되는 원격 저장소 제공 서비스

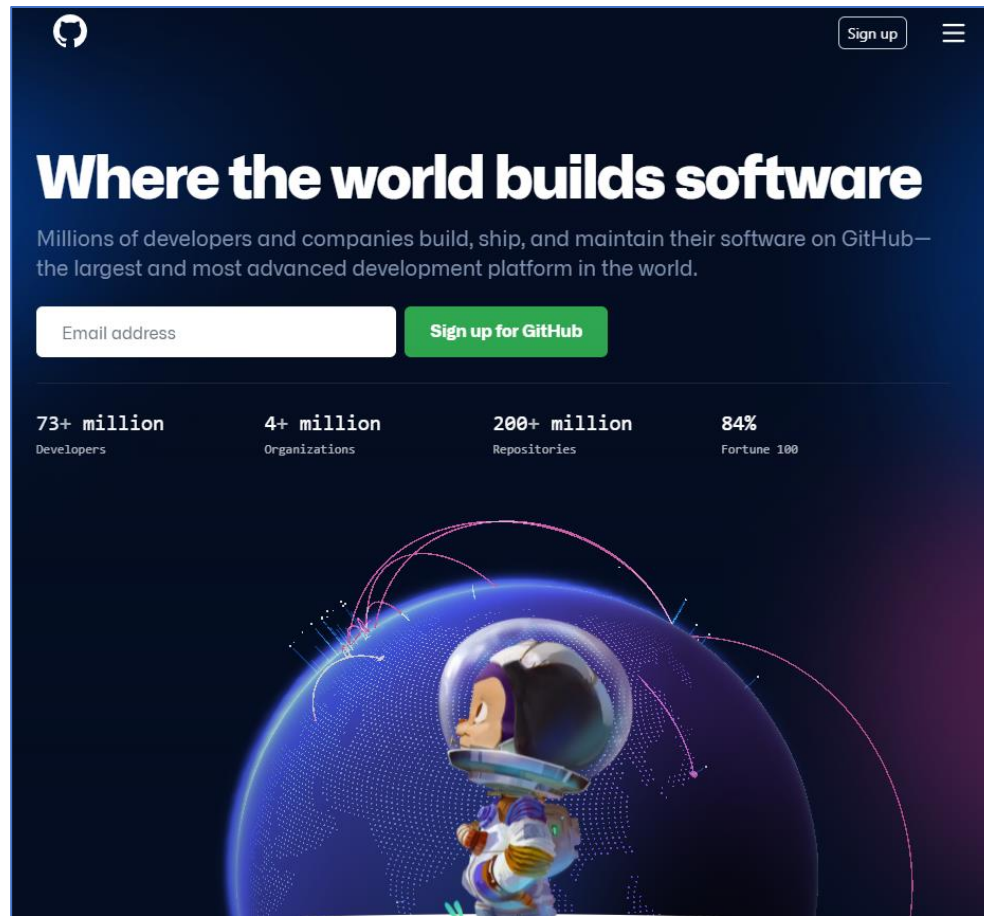
❖ 특징

- 무료 및 유료 서비스
- 다양한 오픈 소스 제공
- Git을 설치하지 않고도 온라인상에서 버전 관리 기능 사용 가능
- 지역 저장소를 온라인상에 백업 가능
- 협업 프로젝트에 사용 가능
- 개발 이력 기록 가능

깃허브 시작하기

❖ 깃허브 가입

- www.github.com



깃허브 시작하기

❖ 정보 입력

Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ artklab@naver.com






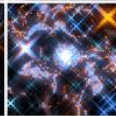
Create a password
✓



Enter a username
✓ ARTKLAB

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
✓ n

Verify your account


나선은하를 고르십시오

깃허브 시작하기

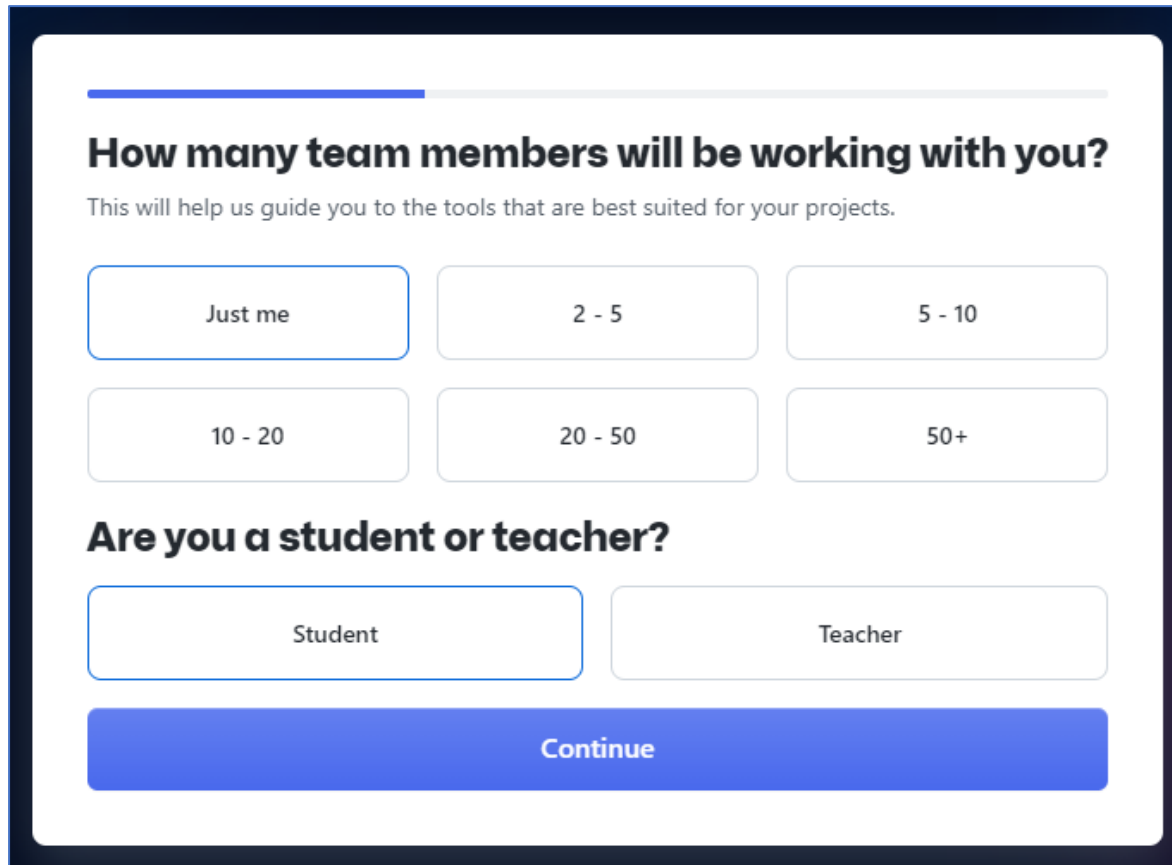
❖ 이메일 인증

A screenshot of a GitHub email verification interface. It features a dark blue background with white text. The text reads: "You're almost done!" followed by "We sent a launch code to artklab@naver.com" where the email address is in green. Below this, there is a prompt "→ Enter code" in pink. At the bottom, there are eight empty square input boxes for the launch code.

You're almost done!
We sent a launch code to `artklab@naver.com`
→ Enter code

깃허브 시작하기

❖ 팀 설정



A screenshot of the GitHub team setup form. The form is titled 'How many team members will be working with you?' and includes a progress bar at the top. Below the title is a subtitle: 'This will help us guide you to the tools that are best suited for your projects.' There are six radio button options for team size: 'Just me', '2 - 5', '5 - 10', '10 - 20', '20 - 50', and '50+'. The 'Just me' option is selected. Below the team size options is another question: 'Are you a student or teacher?' with two radio button options: 'Student' and 'Teacher'. The 'Student' option is selected. At the bottom of the form is a large blue 'Continue' button.

How many team members will be working with you?

This will help us guide you to the tools that are best suited for your projects.

☒ Just me ☐ 2 - 5 ☐ 5 - 10

☐ 10 - 20 ☐ 20 - 50 ☐ 50+

Are you a student or teacher?

☒ Student ☐ Teacher








Continue

깃허브 시작하기

❖ 관심사 선택

What specific features are you interested in using?

Select all that apply so we can point you to the right GitHub plan.

- ☐  Collaborative coding
Codespaces, Pull requests, Notifications, Code review, Code review assignments, Code owners, Draft pull requests, Protected branches, and more.
- ☐  Automation and CI/CD
Actions, Packages, APIs, GitHub Pages, GitHub Marketplace, Webhooks, Hosted runners, Self-hosted runners, Secrets management, and more.
- ☐  Security
Private repos, 2FA, Required reviews, Required status checks, Code scanning, Secret scanning, Dependency graph, Dependabot alerts, and more.
- ☐  Client Apps
GitHub Mobile, GitHub CLI, and GitHub Desktop.
- ☐  Project Management
Projects, Labels, Milestones, Issues, Unified Contribution Graph, Org activity graph, Org dependency insights, Repo insights, Wikis, and GitHub Insights.
- ☐  Team Administration
Organizations, Invitations, Team sync, Custom roles, Domain verification, Audit Log API, Repo creation restriction, and Notification restriction.
- ☐  Community
GitHub Marketplace, GitHub Sponsors, GitHub Learning Lab, Electron, and Atom.

Continue

깃허브 시작하기

❖ 무료 및 유료 선택

The screenshot displays the GitHub account creation interface. On the left, the 'Free' plan is detailed with the following features:

- ① Unlimited public/private repositories
- ① 2,000 Actions minutes/month
Free for public repositories
- ① 500MB of Packages storage
Free for public repositories
- ① Community support

At the bottom of the 'Free' section, a button labeled 'Continue for free' is highlighted with a red rectangular border.

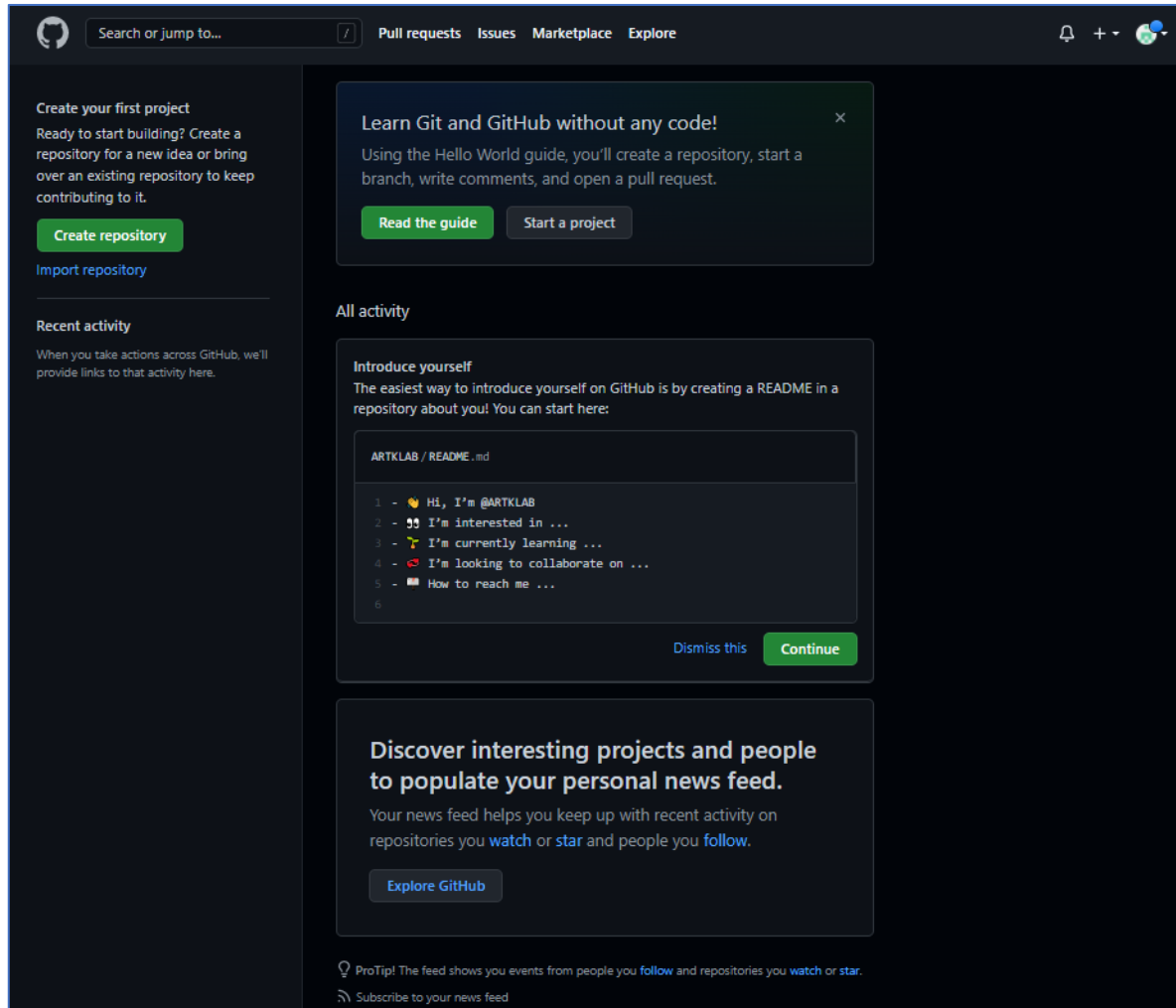
On the right, the 'Get additional student benefits' section is shown, which includes:

- GitHub Pro**
 - ① Protect your branches
Ensure that collaborators on your repository cannot make irrevocable changes to branches.
 - ① Draft pull requests
 - ① Pages and Wikis
 - ① 3,000 CI/CD minutes/month
Free for public repositories
 - ① 2GB of Packages storage
Free for public repositories
 - ① Web-based support
- GitHub Student Developer Pack**
 - ① Free access to the industry's best developer tools
Hundreds of offers, including Digital Ocean, Microsoft Azure, Heroku, MongoDB, DataDog, Twilio, and Stripe.
- GitHub Campus Expert training**
 - ① Enrich your college technical community
Learn the skills to build diverse tech communities on campus with training, mentorship, and support from GitHub.

At the bottom of the right section, a blue button labeled 'Apply for your GitHub student benefits' is visible.

깃허브 시작하기

❖ 초기 화면





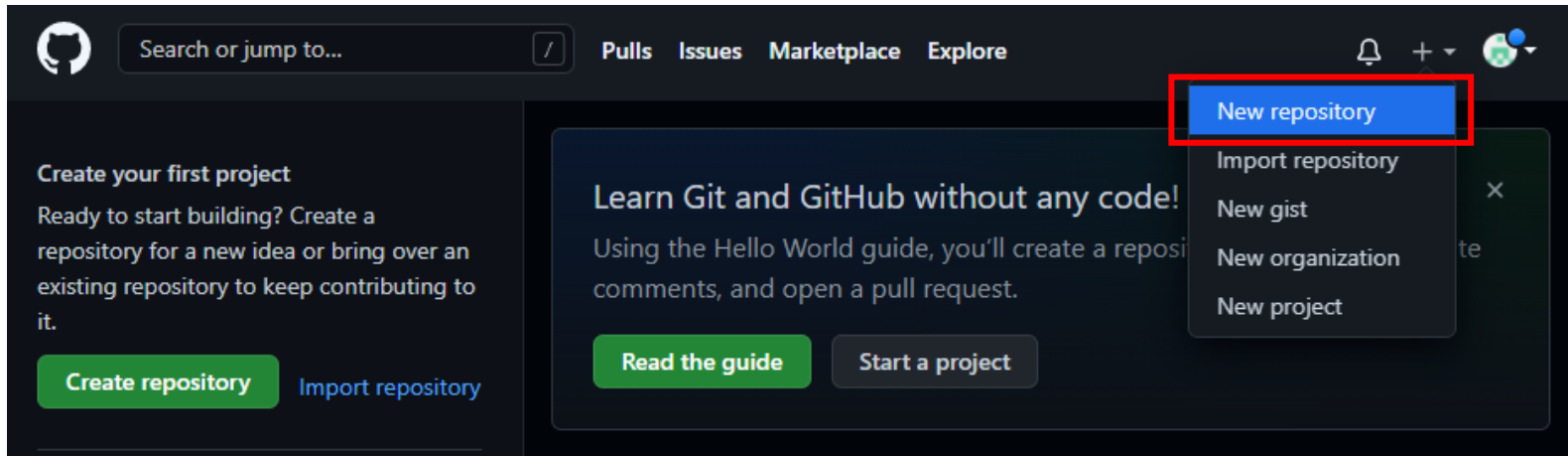
03

지역 저장소를
원격 저장소에
연결하기

원격 저장소 생성

❖ 저장소(repository) 생성

- 오른쪽 상단 + 버튼 클릭 → New repository 선택



원격 저장소 생성

Create a new repository

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

Owner *

HelloCoding22 ▾


Repository name *


저장소 이름

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

Description (optional)

저장소 설명(옵션)

☒  **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.

공개 여부 설정

Initialize this repository with:

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

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

Add .gitignore

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

.gitignore template: None ▾

Choose a license

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

License: None ▾

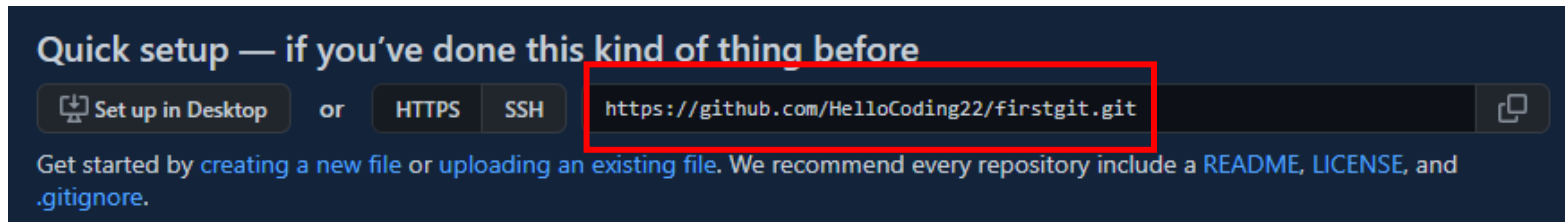
① You are creating a public repository in your personal account.

Create repository

원격 저장소 생성

❖ 원격 저장소 주소

```
https://github.com/아이디/저장소이름.git
```



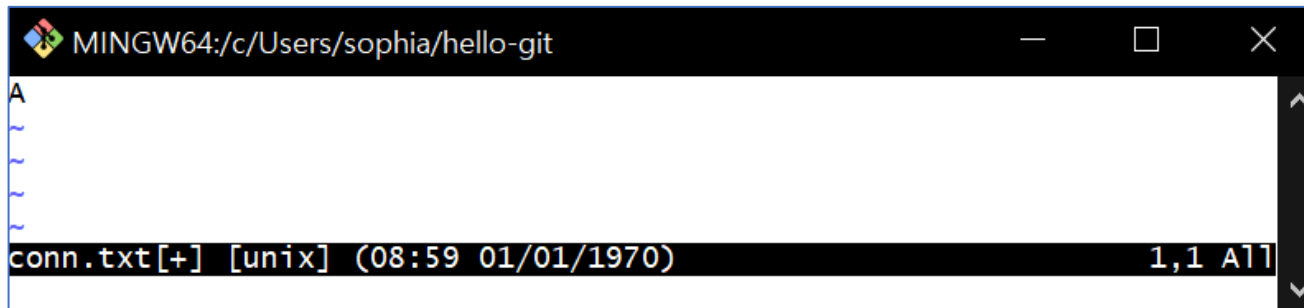
- 원격 저장소 주소만 있으면 어디서든 지역 저장소를 백업하거나 다른 사람과 협업이 가능함

지역 저장소와 원격 저장소 연결

❖ 지역 저장소(hello-git)를 원격 저장소에 연결

- 새로운 파일(conn.txt) 생성 후, 'A' 입력

```
$ vim conn.txt
```



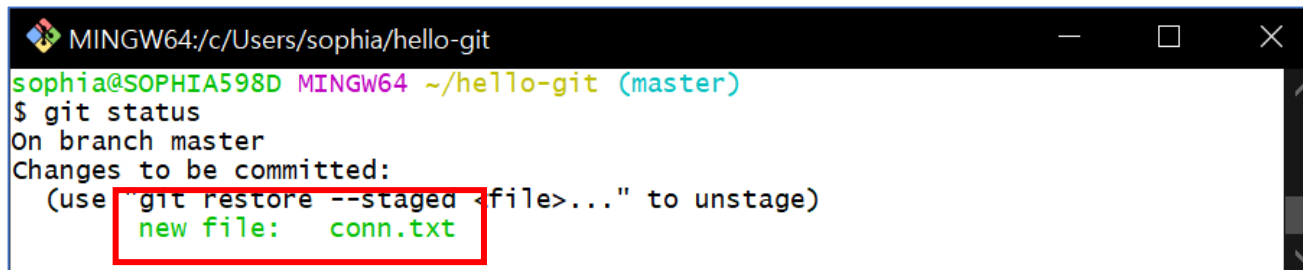
지역 저장소와 원격 저장소 연결

❖ 스테이지에 파일 추가

```
$ git add conn.txt
```

❖ 깃 상태 확인

```
$ git status
```

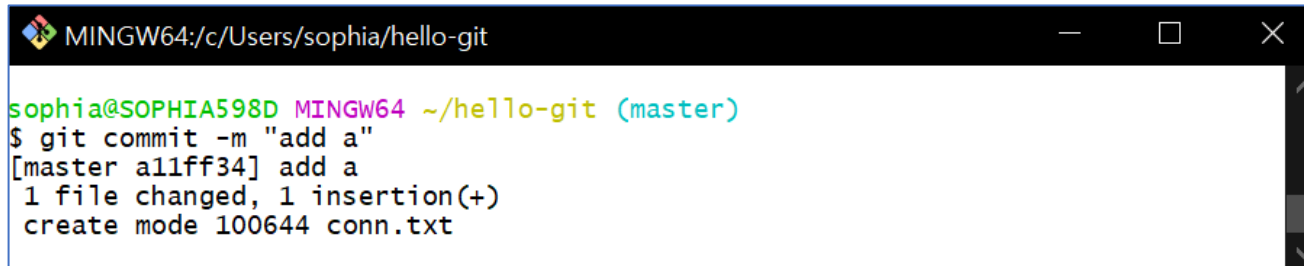


```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   conn.txt
```

지역 저장소와 원격 저장소 연결

❖ 커밋 실행

```
$ git commit -m "add a"
```

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/sophia/hello-git'. The terminal shows the execution of the 'git commit -m "add a"' command. The output indicates that the commit was successful on the 'master' branch with hash 'a11ff34', adding a file named 'conn.txt'.

```
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git commit -m "add a"
[master a11ff34] add a
1 file changed, 1 insertion(+)
create mode 100644 conn.txt
```

지역 저장소와 원격 저장소 연결

❖ 브랜치 변경

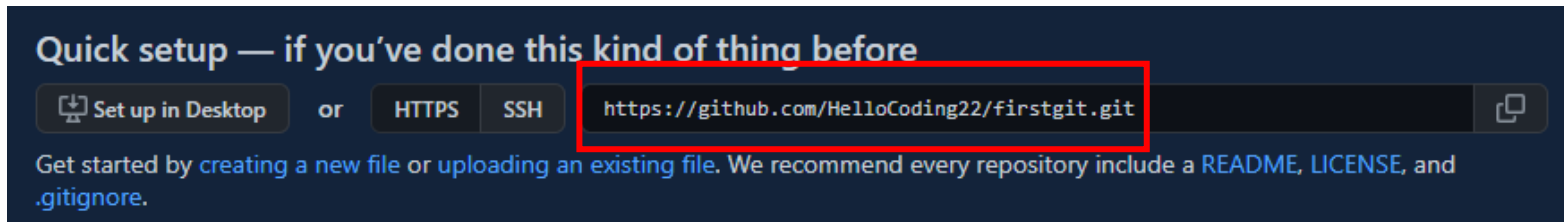
- 기존 master 브랜치에서 main 브랜치로 변경

```
$ git branch -M main
```

지역 저장소와 원격 저장소 연결

❖ 원격 저장소에 연결

- 깃허브 저장소 주소 복사



- 터미널에 명령어 입력

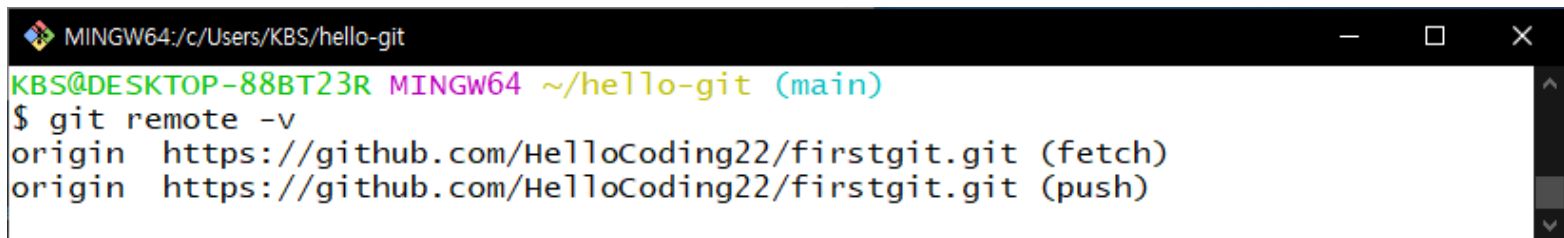
```
$ git remote add origin 복사한_주소
```

- remote : 원격저장소
- origin : 깃허브 저장소 주소를 등록할 식별자

지역 저장소와 원격 저장소 연결

❖ 원격 저장소에 제대로 연결되었는지 확인

```
$ git remote -v
```

A screenshot of a Windows terminal window with a black title bar and standard window controls. The title bar text is 'MINGW64:/c/Users/KBS/hello-git'. The terminal content shows the command '\$ git remote -v' and its output: 'origin https://github.com/HelloCoding22/firstgit.git (fetch)' and 'origin https://github.com/HelloCoding22/firstgit.git (push)'. The prompt character is '\$'. The terminal has a vertical scrollbar on the right side.

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git remote -v
origin https://github.com/HelloCoding22/firstgit.git (fetch)
origin https://github.com/HelloCoding22/firstgit.git (push)
```

원격 저장소에 파일 올리기

❖ 원격저장소에 파일 올리기(push)

```
$ git push -u origin main
```

- -u
 - 지역저장소의 브랜치를 원격저장소에 연결
 - 최초 한 번만 실행하면 됨
- origin
 - 원격저장소 주소
- main
 - 원격저장소의 브랜치 이름

(에러 대응) 지역저장소와 원격저장소 연결

❖ 지역저장소와 원격저장소의 브랜치가 다른 경우

- 파일 업로드(push) 실행 시, 에러 발생
- 원격저장소 브랜치 : main
- 지역저장소 브랜치 : master



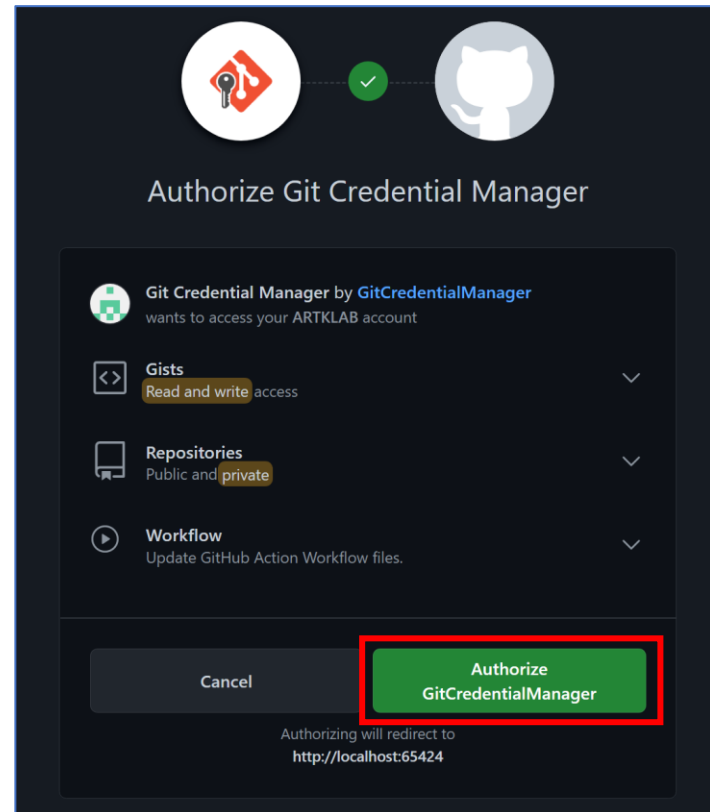
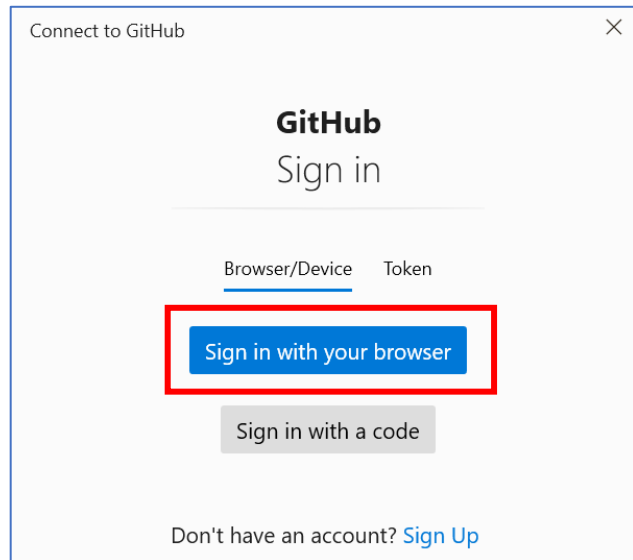
A screenshot of a Windows terminal window titled "MINGW64:/c/Users/KBS/hello-git". The prompt is "KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)". The user has entered the command "\$ git push -u origin main". The terminal output shows an error: "error: src refspec main does not match any" and "error: failed to push some refs to 'https://github.com/HelloCoding22/firstgit.git'". A red box highlights the "(master)" in the prompt, and a red arrow points from the text "지역저장소 브랜치" (Local Repository Branch) to this box.

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/HelloCoding22/firstgit.git'
```

원격 저장소에 파일 올리기

❖ 사용자 인증 과정

- windows
 - ID, PW로 인증



원격 저장소에 파일 올리기

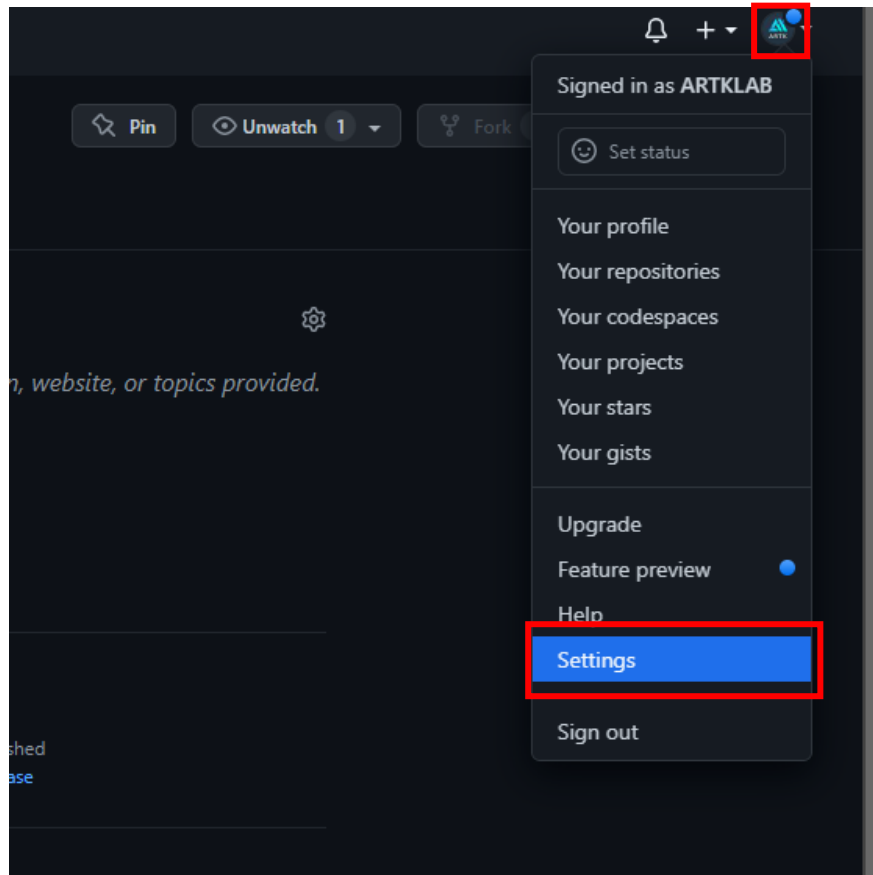
❖ 사용자 인증 과정

- mac
 - 주로 토큰 방식의 인증을 사용
 - Github 사이트에서 토큰 발급 과정 필요

[macOS] 토큰 인증 로그인

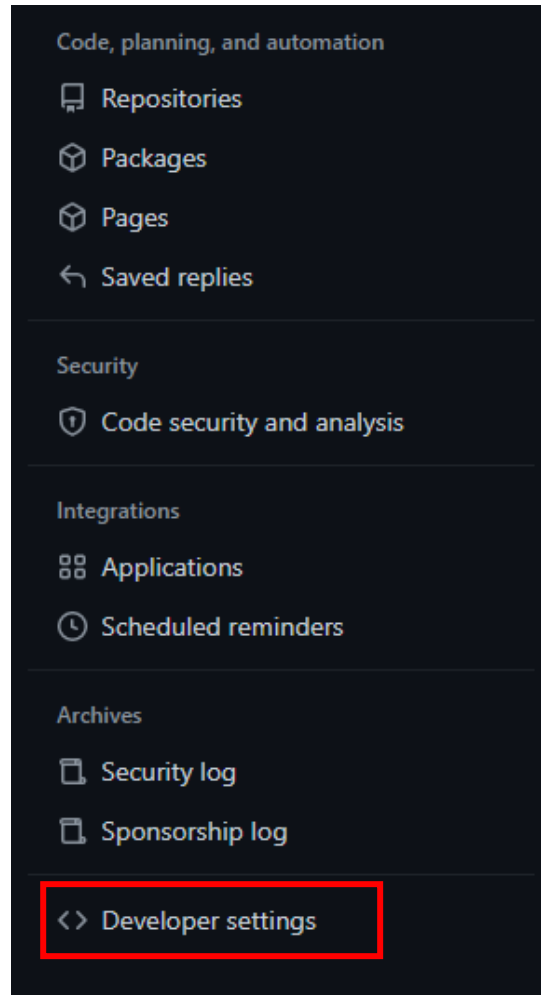
❖ Personal Access Token 생성

- 깃허브 로그인 → [Settings] 선택



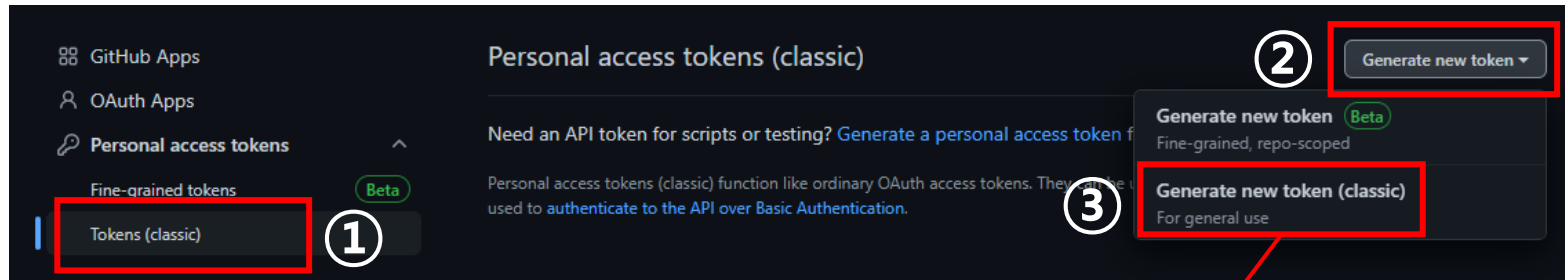
[macOS] 토큰 인증 로그인

❖ 왼쪽 메뉴 하단 → [Developer settings] 선택

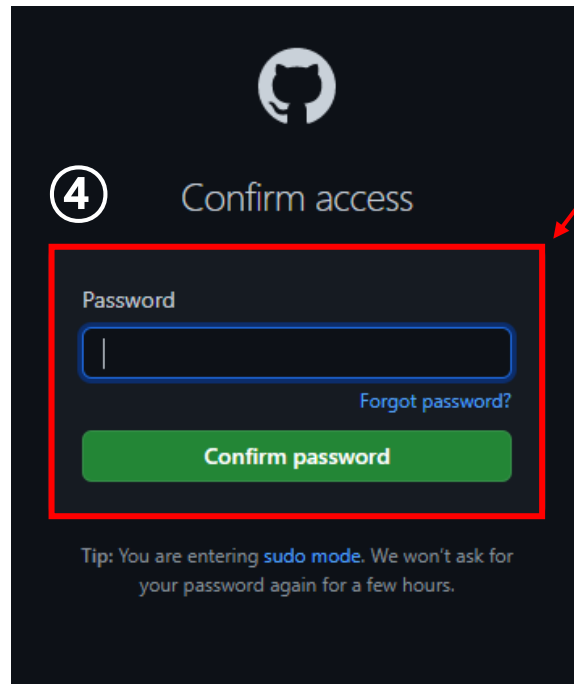


[macOS] 토큰 인증 로그인

❖ [Personal access tokens] 선택



❖ 비밀번호 입력



[macOS] 토큰 인증 로그인

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

What's this token for?

Expiration *

30 days

The token will expire on Fri, Feb 25 2022

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<div>V</div> <div>repo</div> <div><input type="checkbox"/> repo:status</div> <div><input type="checkbox"/> repo_deployment</div> <div><input type="checkbox"/> public_repo</div> <div><input type="checkbox"/> repo:invite</div> <div><input type="checkbox"/> security_events</div>	<div>Full control of private repositories</div> <div>Access commit status</div> <div>Access deployment status</div> <div>Access public repositories</div> <div>Access repository invitations</div> <div>Read and write security events</div>
<div><input type="checkbox"/> workflow</div>	<div>Update GitHub Action workflows</div>
<div><input type="checkbox"/> write:packages</div> <div><input type="checkbox"/> read:packages</div>	<div>Upload packages to GitHub Package Registry</div> <div>Download packages from GitHub Package Registry</div>
<div><input type="checkbox"/> delete:packages</div>	<div>Delete packages from GitHub Package Registry</div>
<div><input type="checkbox"/> admin:org</div> <div><input type="checkbox"/> write:org</div> <div><input type="checkbox"/> read:org</div>	<div>Full control of orgs and teams, read and write org projects</div> <div>Read and write org and team membership, read and write org projects</div> <div>Read org and team membership, read org projects</div>

Generate token

Cancel

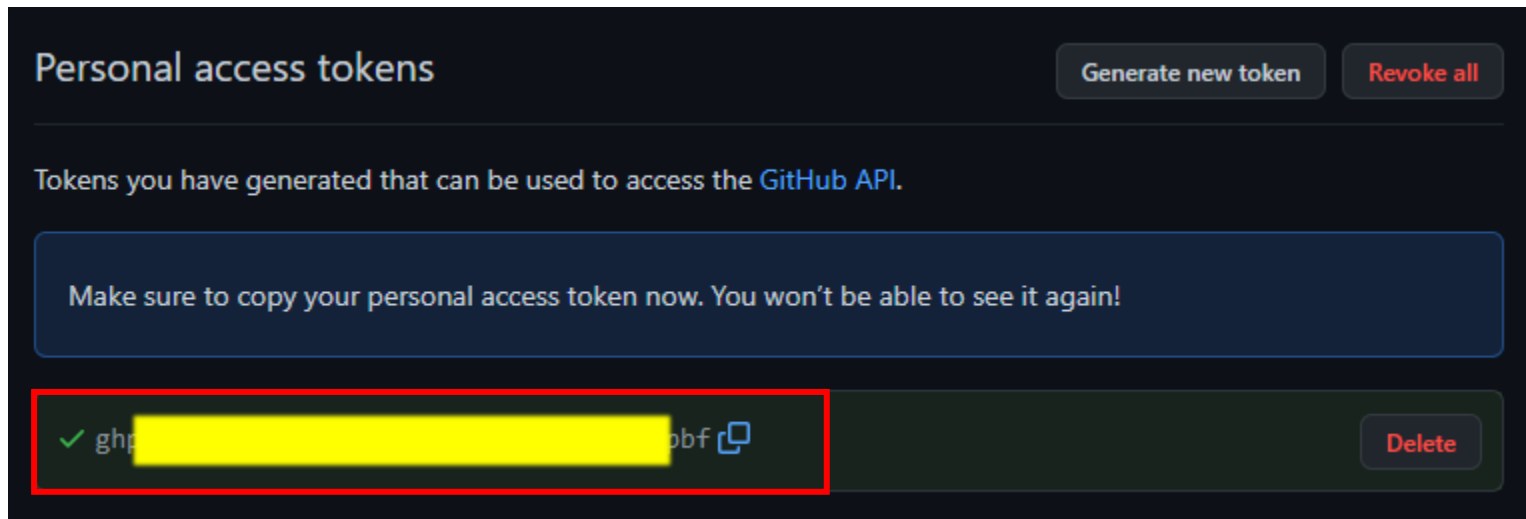
토큰 이름 입력

토큰 유효기간

토큰 권한 선택

[macOS] 토큰 인증 로그인

❖ 토큰 복사

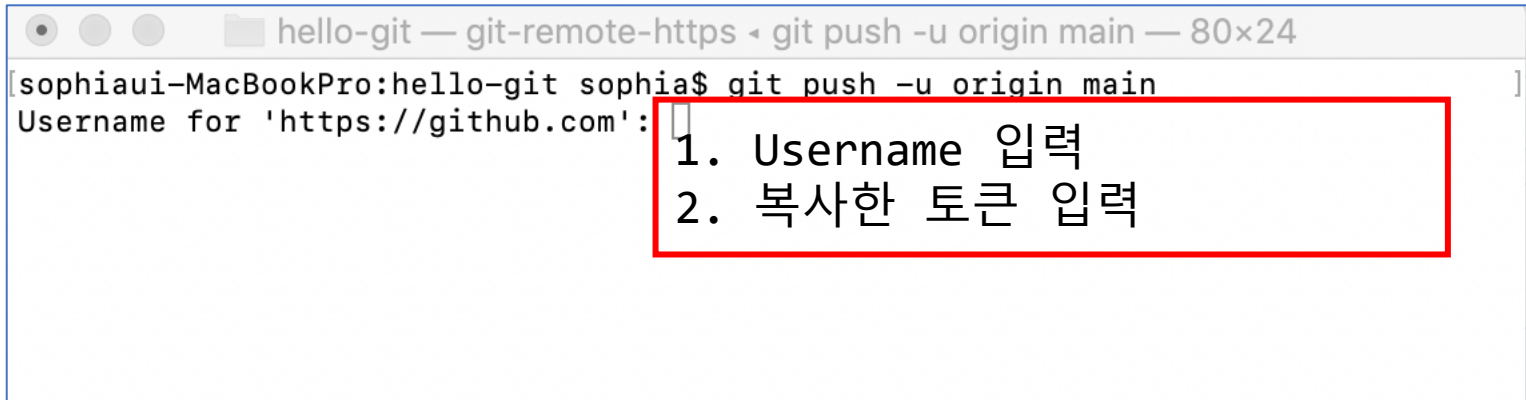


[macOS] 토큰 인증 로그인

❖ 원격 저장소에 파일 올리기(push)

```
$ git push -u origin main
```

❖ 토큰 인증



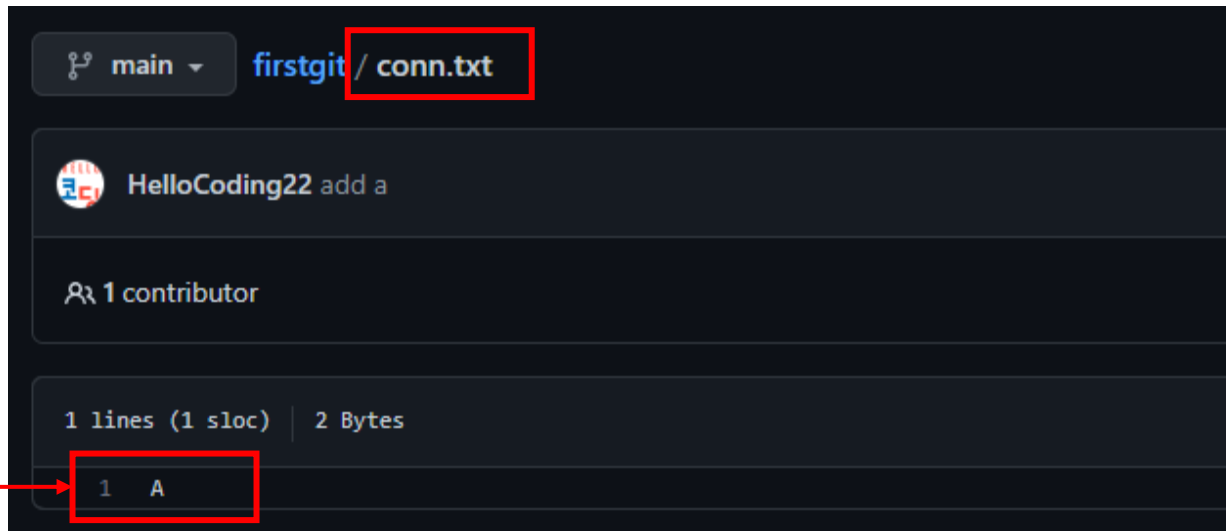
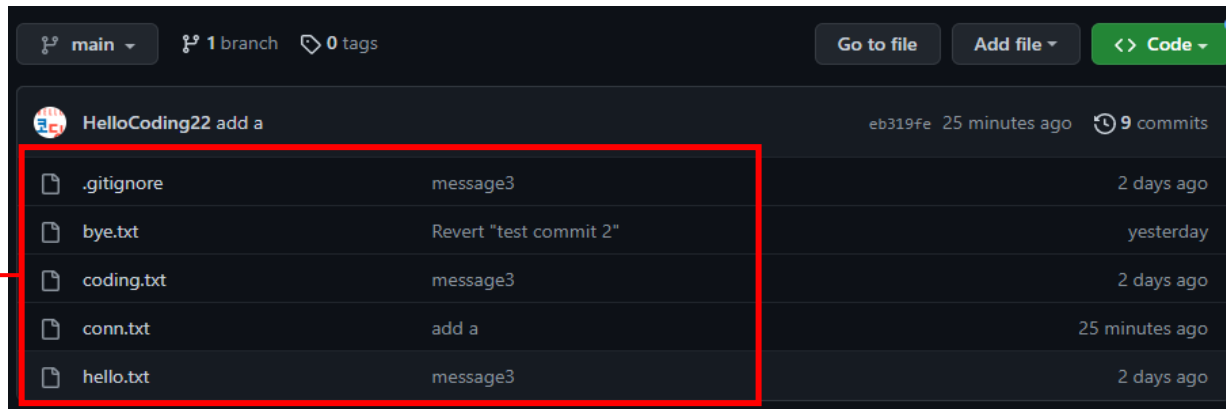
원격 저장소에 파일 올리기

❖ 원격 저장소에 파일 올리기 성공

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 254 bytes | 254.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/HelloCoding22/firstgit.git
    eb319fe..17af6d2  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

원격 저장소에 파일 올리기

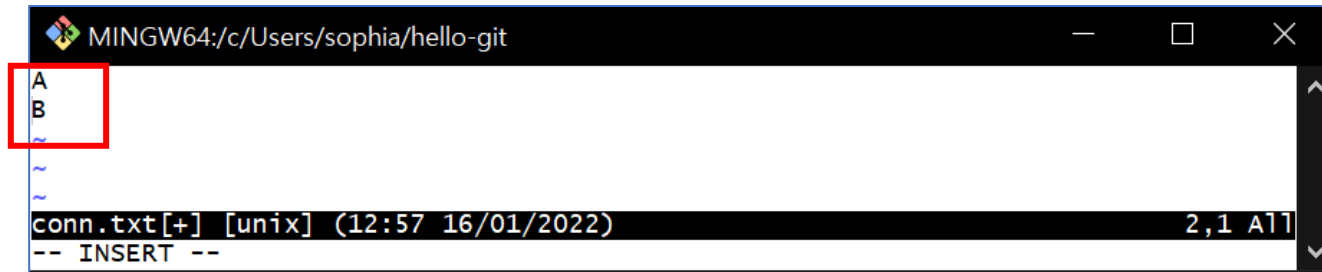
❖ 깃허브에서 확인하기



원격 저장소에 파일 올리기

❖ 파일 수정 후 원격 저장소에 올리기

```
$ vim conn.txt
```



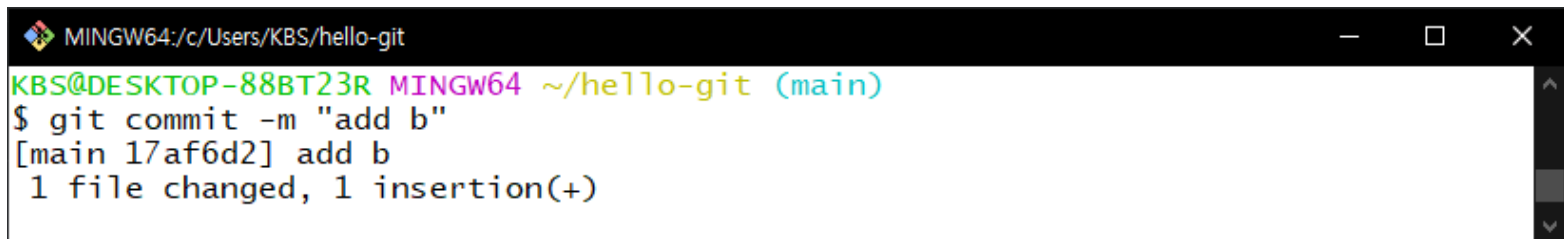
원격 저장소에 파일 올리기

❖ 스테이지에 파일 추가

```
$ git add conn.txt
```

❖ 커밋 실행

```
$ git commit -m "add b"
```

A screenshot of a Windows terminal window with a black background and white text. The title bar shows 'MINGW64: c:/Users/KBS/hello-git'. The terminal content shows the user 'KBS@DESKTOP-88BT23R' in a 'MINGW64' environment at the directory '~/hello-git' on the 'main' branch. The command '\$ git commit -m "add b"' has been executed, resulting in the output '[main 17af6d2] add b' and '1 file changed, 1 insertion(+)'.

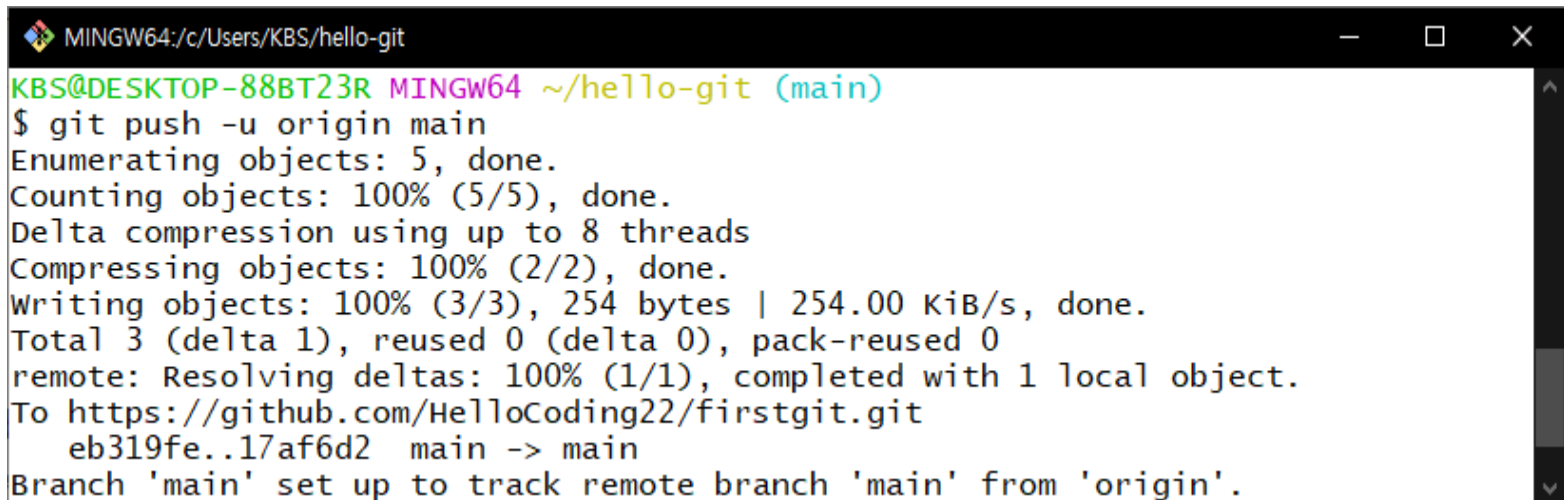
```
MINGW64: c:/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git commit -m "add b"
[main 17af6d2] add b
1 file changed, 1 insertion(+)
```

원격 저장소에 파일 올리기

❖ 원격 저장소에 파일 올리기(push)

- 한 번 이상 원격 저장소에 업로드한 경우, 다음 명령어로 push 가능

```
$ git push
```



```
MINGW64:/c:/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 254 bytes | 254.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/HelloCoding22/firstgit.git
   eb319fe..17af6d2  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```


원격 저장소에 파일 올리기

❖ 깃허브에서 확인하기

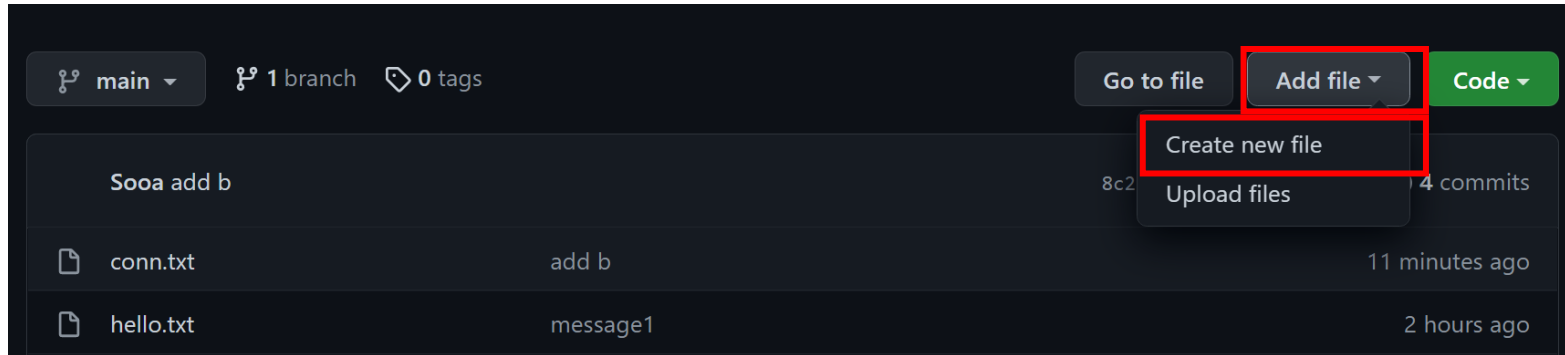
The image shows a GitHub repository interface. At the top, there are buttons for 'Go to file', 'Add file', and 'Code'. Below this, the repository name 'HelloCoding22' is shown with a commit message 'add b' and a timestamp '6 minutes ago'. A list of files is displayed, with 'conn.txt' highlighted by a red box. A red arrow points from this box to the file's content view below. The content view shows the file 'conn.txt' with 2 lines of code: '1 A' and '2 B', which are also highlighted by a red box.

File	Commit Message	Time
.gitignore	message3	2 days ago
bye.txt	Revert "test commit 2"	yesterday
coding.txt	message3	2 days ago
conn.txt	add b	6 minutes ago
hello.txt	message3	2 days ago

Line	Content
1	A
2	B

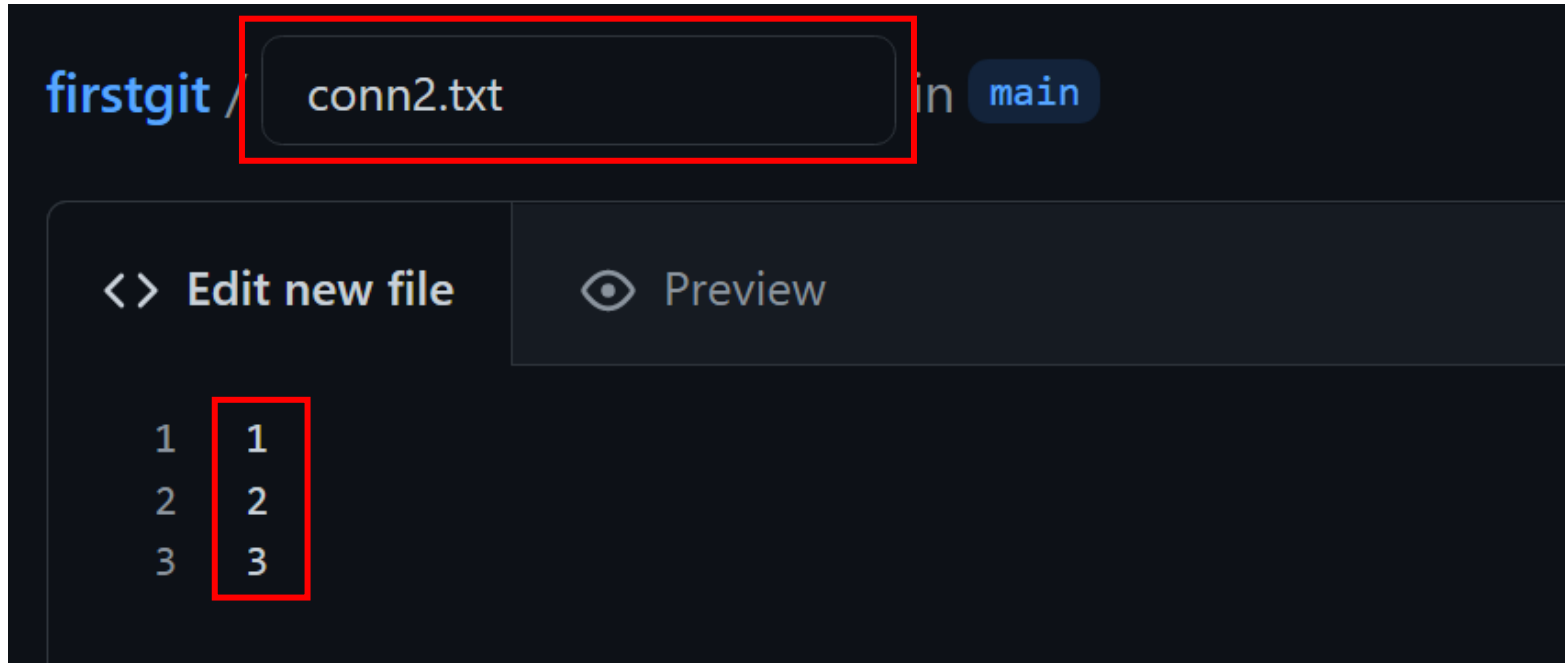
깃허브 사이트에서 직접 커밋하기

- ❖ 지역 저장소가 있는 컴퓨터를 사용하지 못하는 경우
 - 깃허브 사이트에서 직접 커밋이 가능



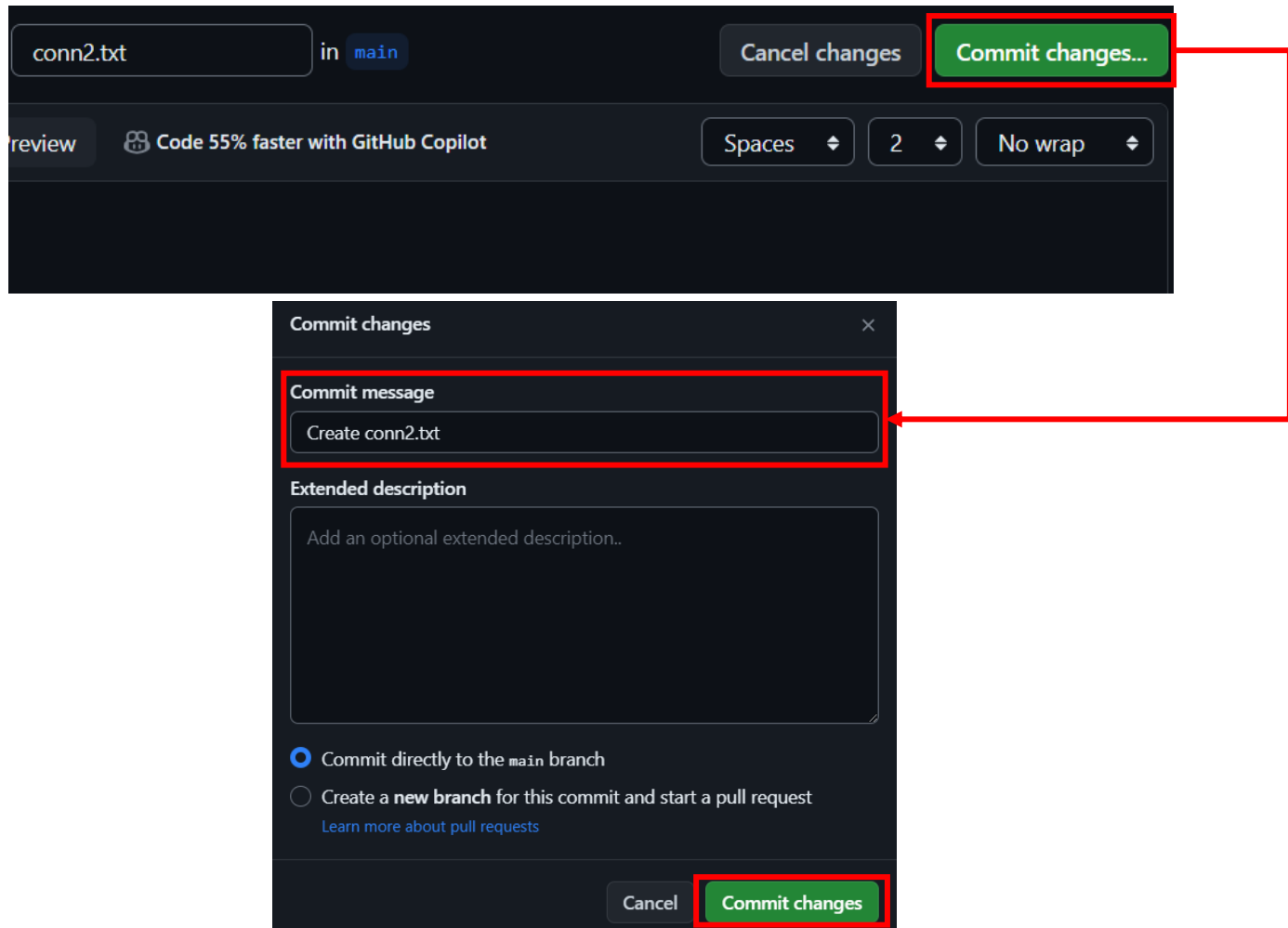
깃허브 사이트에서 직접 커밋하기

❖ 파일명과 내용 입력



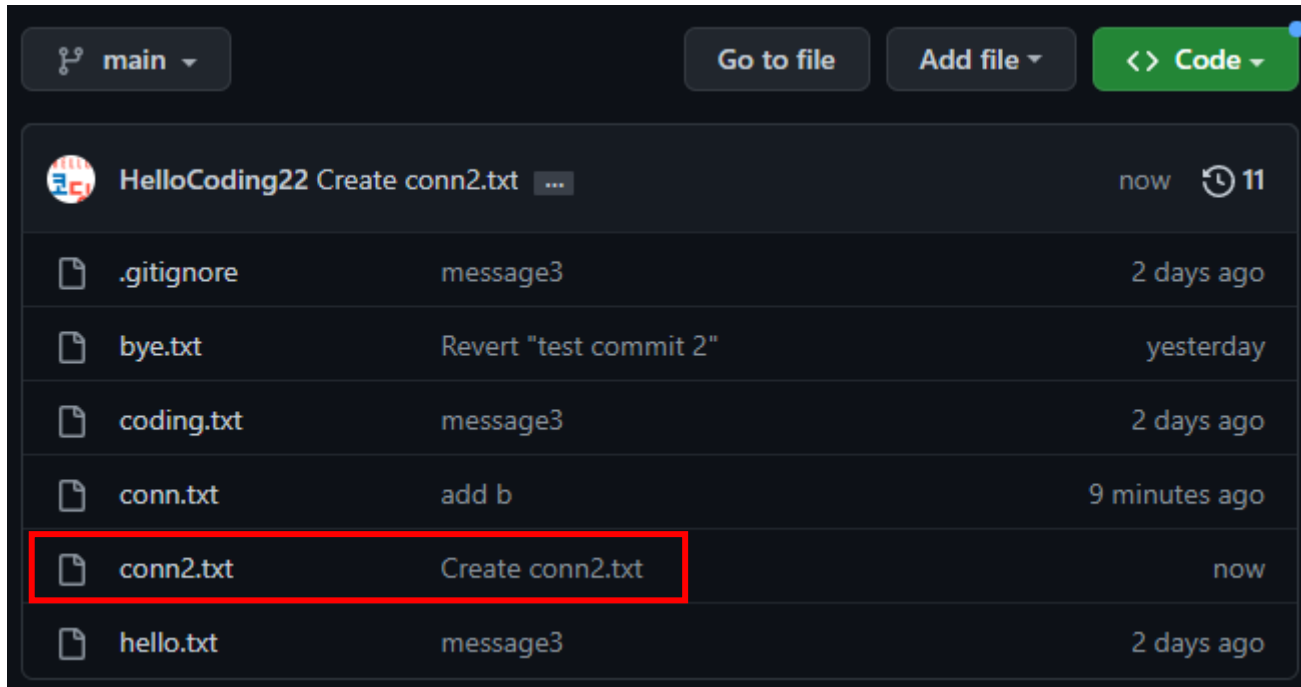
깃허브 사이트에서 직접 커밋하기

❖ 커밋 메시지 입력 후 커밋



깃허브 사이트에서 직접 커밋하기

❖ 새 파일(conn2.txt) 추가



원격 저장소에서 파일 다운로드

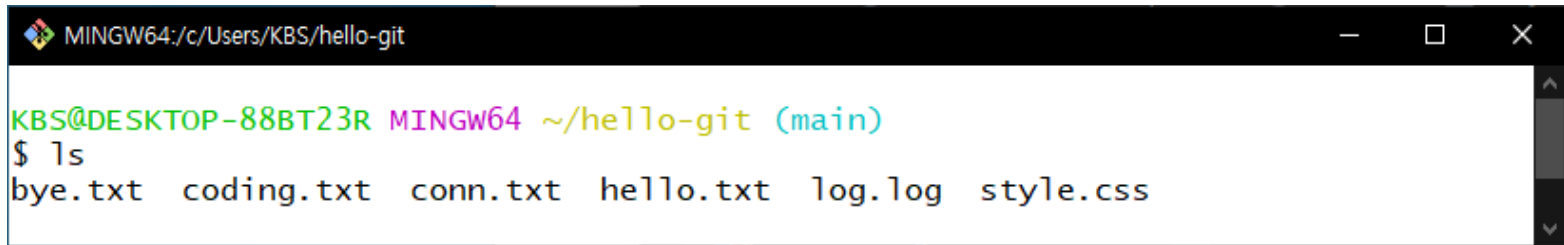
❖ 원격 저장소에서 파일 다운로드(pull)

- 원격 저장소와 지역 저장소의 상태가 다른 경우
 - 원격 저장소에 있는 파일을 다른 사용자가 수정
 - 깃허브 사이트에서 직접 커밋
- 원격 저장소와 지역 저장소의 상태를 같게 만들어야 함
 - 원격 저장소의 파일을 지역 저장소로 가져옴
 - 풀(pull)

원격 저장소에서 파일 내려받기

❖ 지역 저장소 파일 확인

```
$ ls
```



A terminal window titled 'MINGW64:/c/Users/KBS/hello-git' showing the output of the 'ls' command. The prompt is 'KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)'. The command '\$ ls' has been entered, and the output is 'bye.txt coding.txt conn.txt hello.txt log.log style.css'. The file 'conn2.txt' is not listed.

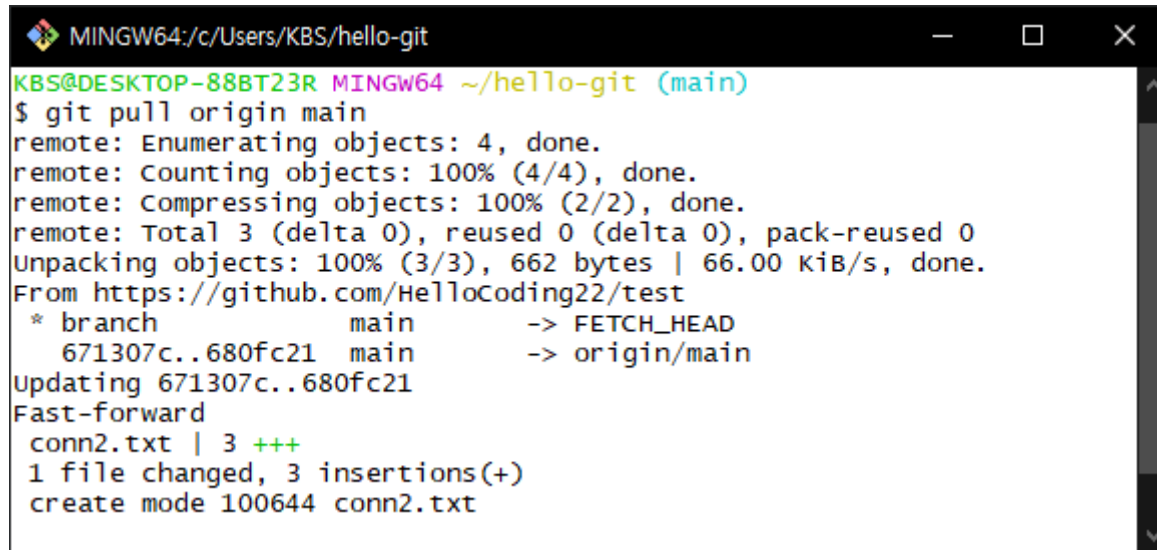
```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ ls
bye.txt  coding.txt  conn.txt  hello.txt  log.log  style.css
```

- conn2.txt 파일이 존재하지 않음

원격 저장소에서 파일 내려받기

❖ 원격 저장소의 내용을 가져오기

```
$ git pull origin main
```

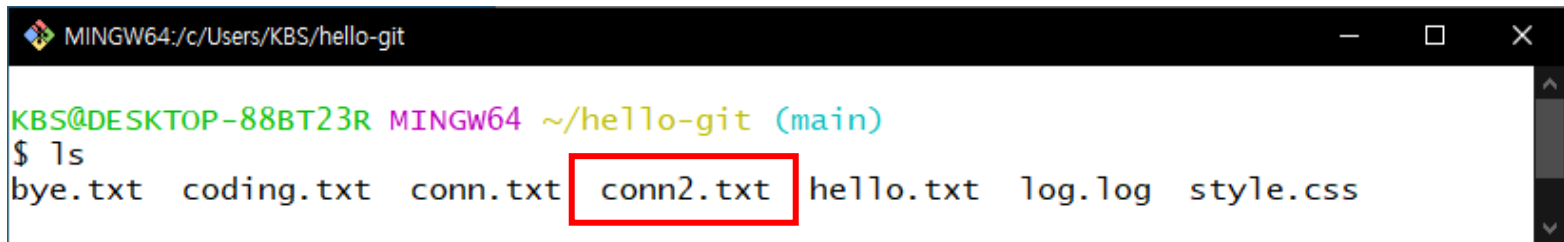
A terminal window titled 'MINGW64:/c/Users/KBS/hello-git' showing the output of a 'git pull' command. The output indicates that 4 objects were enumerated, 4/4 counted, and 2/2 compressed. It shows a fast-forward update from commit 671307c to 680fc21, pulling in 3 new lines to 'conn2.txt'.

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 662 bytes | 66.00 KiB/s, done.
From https://github.com/HelloCoding22/test
 * branch                main                -> FETCH_HEAD
    671307c..680fc21      main                -> origin/main
Updating 671307c..680fc21
Fast-forward
 conn2.txt | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 conn2.txt
```


원격 저장소에서 파일 내려받기

❖ 지역 저장소 파일 확인

```
$ ls
```



A screenshot of a terminal window titled 'MINGW64: c:/Users/KBS/hello-git'. The prompt is 'KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)'. The command '\$ ls' has been executed, and the output is 'bye.txt coding.txt conn.txt conn2.txt hello.txt log.log style.css'. The file 'conn2.txt' is highlighted with a red rectangular box.

- conn2.txt 파일 생성됨

정리하기

<code>git remote add origin 원격_저장소_주소</code>	원격 저장소에 연결하기
<code>git remote -v</code>	원격 저장소에 연결되었는지 확인하기
<code>git push -u origin main</code>	지역 저장소의 커밋을 원격 저장소에 올리기
<code>git push</code>	두번째 커밋을 원격 저장소에 올리기
<code>git pull</code>	원격 저장소의 커밋을 지역 저장소로 가져오기

THANK 😊 YOU