

Github 연동 처리

pykwon

GitHub 용도

Git 원격 저장소를 제공하는 대표적인 서비스가 GitHub이다. 단순히 원격 저장소만을 제공하는 것이 아니라 여러가지 프로젝트 진행을 원활하게 하는 도구를 함께 제공한다.

GitHub 장점

전 세계에서 진행되는 오픈 소스 프로젝트가 많이 모여 있어서 이에 참여하고 오픈소스에 기여할 수 있는 기회가 제공된다.

개발자는 GitHub를 이용해서 자신이 작성했던 코드 그 자체를 곧바로 제공할 수 있다.

IT 개발과 관련된 많은 디자이너 및 기획자 역시 자신이 준비했던 문서 및 포트폴리오를 공개할 수 있다.

가장 큰 장점 중 하나로 개발 시 협업이 가능하다.

원격 저장소 관련 기본 명령어

명령어	기 능
git clone	원격 저장소의 모든 내용을 로컬 저장소로 복사한다.
git remote	로컬 저장소를 특정 원격 저장소와 연결한다.
git push	로컬 저장소의 변경사항을 원격 저장소로 보낸다.
git fetch	로컬 저장소와 원격 저장소의 커밋 버전이 다를 때, 원격 저장소의 커밋 내역을 로컬로 가져온다. 필요시 git merge 로 나중에 병합할 수 있다.
git pull	원격 저장소의 커밋 내역을 로컬로 가져와서 자동으로 병합한다.

2. GitHub 가입

GitHub 가입 및 로그인
<https://github.com/>

Welcome to GitHub!
Let's begin the adventure

Enter your email*

✓ pyk202056@gmail.com

Create a password*

✓

Enter a username*

✓ pyk202056

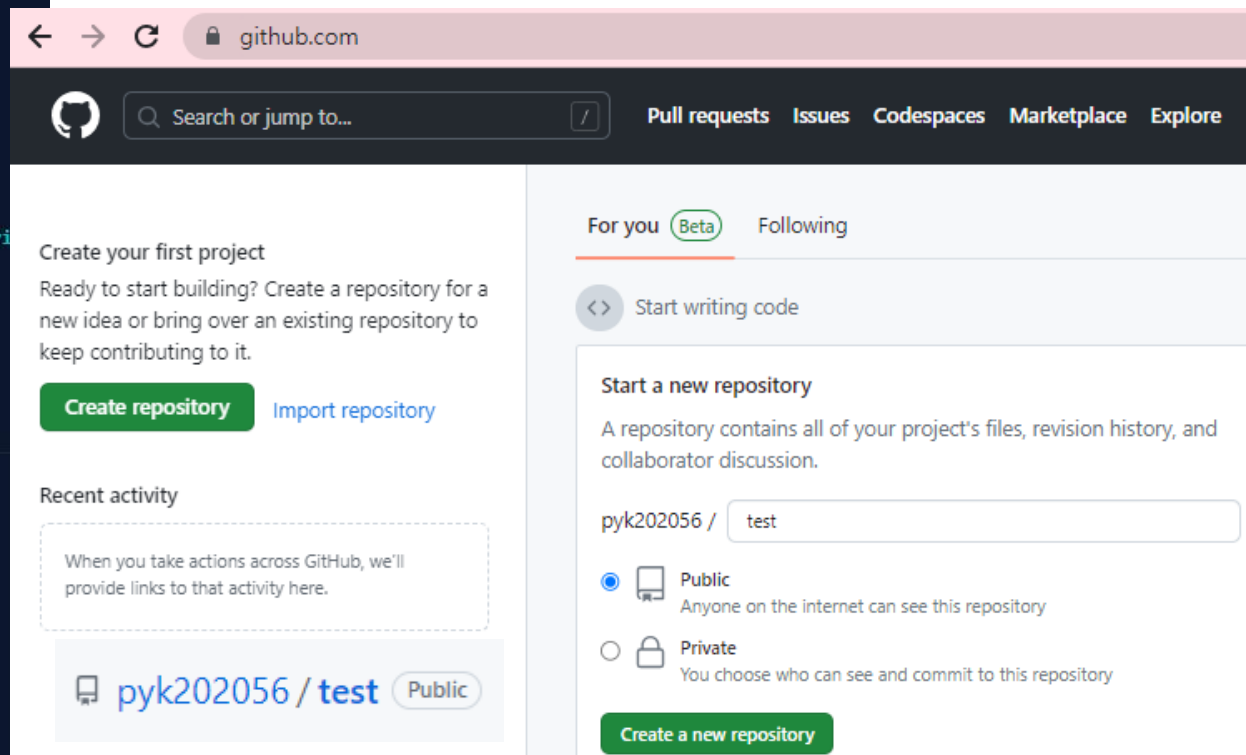
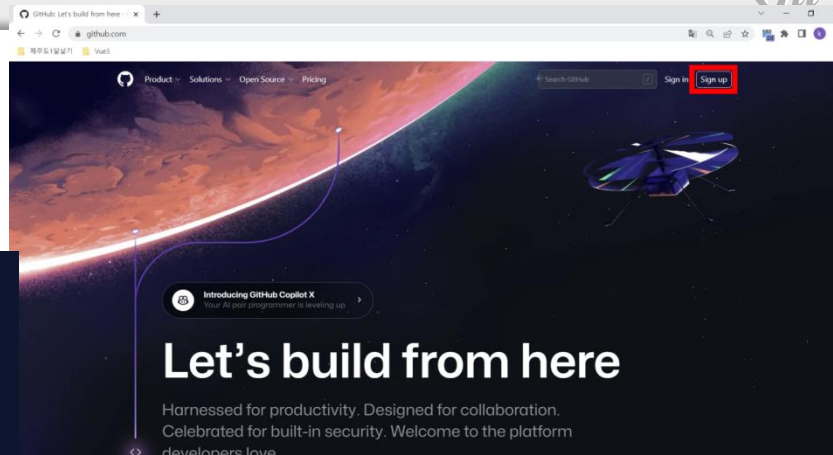
Would you like to receive product updates and announcements via email?

Type "y" for yes or "n" for no

✓ y

Verify your account

두 개의 같은 물체를 표시하는 하나의 사각형을 선택하십시오.



3. 원격 저장소 기본 관리

1) 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.](#)

Required fields are marked with an asterisk (*).

Owner *

Repository name *

pyk202056

/ study

study is available.

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

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.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

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

Choose a license

License: None

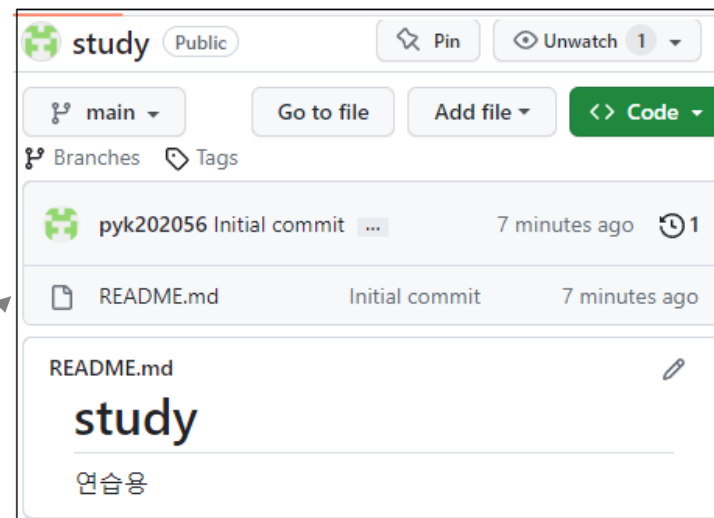
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

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

Create repository

선택한 경우

선택 안한 경우



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

Set up in Desktop

 or

HTTPS

SSH

https://github.com/pyk202056/study.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository has a README file.

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

```
echo "# study" >> README.md
https://github.com/pyk202056/study.git
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/pyk202056/study.git
git branch -M main
git push -u origin main
```

3. 원격 저장소 기본 관리

2) 로컬 저장소와 초기 원격 저장소 연결

git remote add 원격 저장소 별칭 https://~

로컬 저장소 생성

```
$ pwd
$ mkdir github_tutorial
$ cd github_tutorial/
$ git init
Initialized empty Git repository in C:/work/github_tutorial/.git/
tom@DESKTOP-DCAF7UT MINGW64 /c/work/github_tutorial (master)
```

로컬 저장소와 원격 저장소 연결해 파일 업로드

```
$ echo "# study" >> README.md      ← 파일 생성
$ ls
README.md
$ git add README.md
$ git commit -m "first commit"
[master (root-commit) 27eb22e] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

tom@DESKTOP-DCAF7UT MINGW64 /c/work/github_tutorial (master)
$ git branch -M main                ← 브랜치명 변경
tom@DESKTOP-DCAF7UT MINGW64 /c/work/github_tutorial (main)
$ git remote add origin https://github.com/pyk202056/study.git
```

GitHub로 파일 업로드

```
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 210 bytes | 210.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/pyk202056/study.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

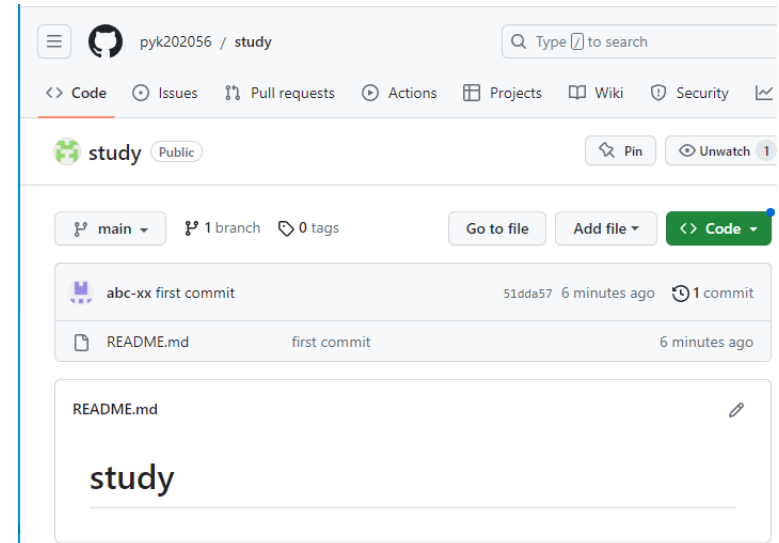
연결 상태 확인

```
$ git remote -v
origin https://github.com/pyk202056/study.git (fetch)
origin https://github.com/pyk202056/study.git (push)
```

GitHub와 연결 종료 / 연결 재개

```
$ git remote rm origin
```

```
$ git remote add origin https://github.com/pyk202056/study.git
```



3. 원격 저장소 기본 관리

3) 로컬 작업 내역을 원격 저장소에 올리기

```
git push 원격브랜치별칭 로컬브랜치명
git push 원격브랜치별칭 --all
```

로컬 저장소 변경

```
$ vi hello.txt
$ cat hello.txt
hello
$ git add hello.txt
$ git commit -m 'first commit'
[main (root-commit) e8596ba] first commit
$ git status
On branch main
nothing to commit, working tree clean
```

원격 저장소에 올리기

```
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 210 bytes | 210.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
...
To https://github.com/pyk202056/study.git
* [new branch]      main -> main
```

참고 : **git the requested URL returned error : 403 해결 방법**

처음에 git remote add origin git_repository주소.git 을 통해 origin 명칭을 만들었지만 해당 주소에 대한 권한이 없어서 push를 할 때 에러가 난 거예요.

Git 인증을 해주면 된다.

```
$ git remote set-url origin https://pyk202056@github.com/pyk202056/study.git
```


3. 원격 저장소 기본 관리

Git 활용한 버전관리

GitHub에서 확인

The screenshot shows the GitHub interface for a repository named 'study' owned by 'pyk202056'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows two commits: 'abc-xx first commit' (b26c4d7, 4 minutes ago) and 'first commit' (16 minutes ago). The file list shows 'README.md' (first commit, 16 minutes ago) and 'hello.txt' (first commit, 4 minutes ago). The README content is 'study'.

pyk202056 / study

Type / to search

<> Code Issues Pull requests Actions Projects Wiki Security

study Public Pin Unwatch 1

main 1 branch 0 tags Go to file Add file <> Code

abc-xx first commit b26c4d7 4 minutes ago 2 commits

README.md	first commit	16 minutes ago
hello.txt	first commit	4 minutes ago

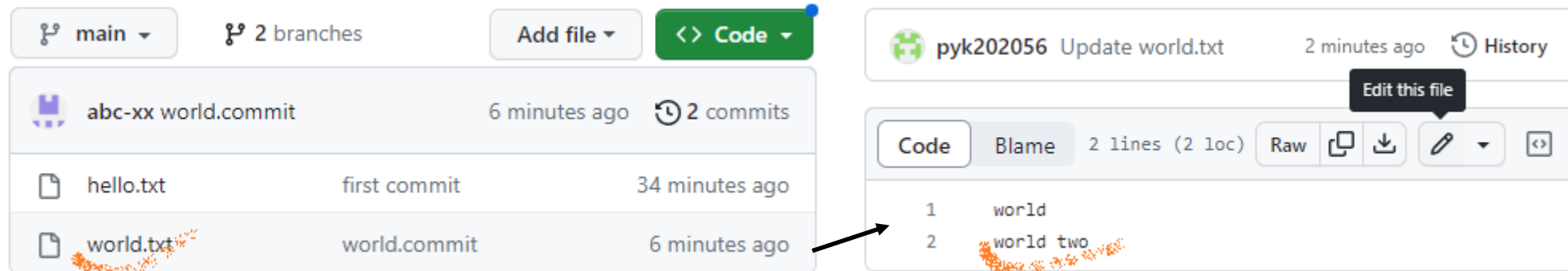
README.md

study

4. 원격 저장소에서 로컬 저장소로 가져오기

원격 저장소 변경

```
$ vi world.txt    $ git add world.txt    $ git commit -m 'first commit'    $ git push origin main
```



원격 저장소의 파일(world.txt)을 변경한 후 로컬 영역에서 해당 파일을 수정 후 commit 한 다음 push 명령을 주면 [rejected] 에러가 발생한다. 이유는 로컬 저장소와 원격 저장소 간에 커밋 버전이 일치하지 않기 때문이다.

```
$ git push origin main
```

To https://github.com/pyk202056/study.git

! [rejected] main -> main (fetch first)

error: failed to push some refs to 'https://github.com/pyk202056/study.git'

hint: Updates were rejected because the remote contains work that you do

hint: not have locally. This is usually caused by another repository pushing

hint: to the same ref. You may want to first integrate the remote changes

hint: (e.g., 'git pull ...') before pushing again. hint: See the 'Note about fast-forwards' in 'git push --help'

for details.

4. 원격 저장소에서 로컬 저장소로 가져오기

1) 원격 저장소 커밋 내역을 로컬 저장소로 가져오기

`git fetch` `git fetch` 명령어는 원격 저장소의 커밋 내역만 가져온다.

단, local repository와 합쳐지진 않는다. 즉, fetch는 원격 저장소에 변경사항이 있는지 확인만 하고, 변경된 데이터를 로컬 저장소에 실제로 가져오지는 않는다. 즉, 원격 저장소의 최신 이력을 확인하는 것이 목적이다.

\$ `git fetch`

remote: Enumerating objects: 5, done.

remote: Counting objects: 100% (5/5), done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), 643 bytes | 107.00 KiB/s, done.

From https://github.com/pyk202056/study

10fb5cd..75b071e main -> origin/main

로컬 저장소의 모든 브랜치 정보 보기

\$ `git branch -a`

* main

remotes/origin/main

로컬 저장소와 원격 저장소의 변경 내역 비교

\$ `git log main..origin/main`

commit 75b071e72da883f12a541188797009a91654d986 (origin/main)

Author: pyk202056 <136792896+pyk202056@users.noreply.github.com>

Update world.txt

4. 원격 저장소에서 로컬 저장소로 가져오기

로컬 저장소와 원격 저장소의 병합

```
$ git merge origin/main
```

```
Updating 10fb5cd..75b071e
```

```
Fast-forward
```

```
world.txt | 1 +
```

```
1 file changed, 1 insertion(+)
```

```
$ git commit -a
```

```
On branch main
```

```
nothing to commit, working tree clean
```

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

```
75b071e (HEAD -> main, origin/main) Update world.txt
```

```
10fb5cd first commit
```

```
b26c4d7 first commit
```

```
51dda57 first commit
```

```
$ git status
```

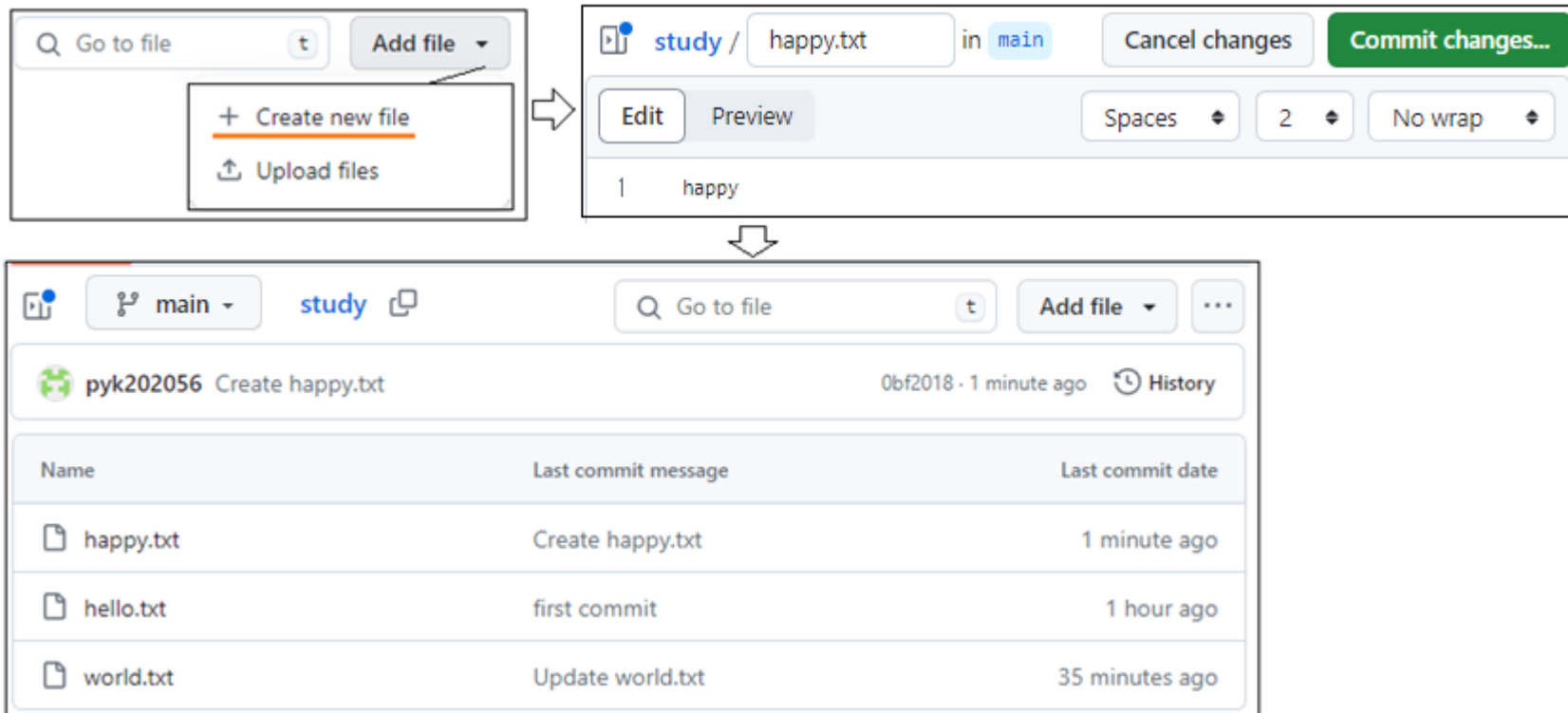
```
On branch main
```

```
nothing to commit, working tree clean
```

4. 원격 저장소에서 로컬 저장소로 가져오기

2) 원격 저장소 커밋 내역을 로컬 저장소로 가져오기 `git pull origin main`

원격 저장소
변경



이 경우 git push 하면 [rejected] 에러가 발생된다. 이유는 로컬저장소와 원격 저장소 간에 커밋 버전이 불일치하기 때문

\$ git push origin main

To <https://github.com/pyk202056/study.git>

! [rejected] main -> main (fetch first)

error: failed to push some refs to 'https://github.com/pyk202056/study.git'

...

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

4. 원격 저장소에서 로컬 저장소로 가져오기

`git pull origin main` `git pull` 명령어는 원격 저장소의 커밋 내역을 로컬로 가져와 자동 병합됨.

```
$ 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), 686 bytes | 76.00 KiB/s, done.
From https://github.com/pyk202056/study
* branch          main          -> FETCH_HEAD
   cbbe302..0bf2018 main        -> origin/main
Merge made by the 'ort' strategy.
 happy.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 happy.txt
```

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

```
789b164 (HEAD -> main, origin/main) Create happy.txt
75b071e Update world.txt
10fb5cd first commit
b26c4d7 first commit
51dda57 first commit
```

```
$ ls
```

```
README.md  happy.txt  hello.txt  world.txt
```

pull은 원격 저장소에서 변경된 메타데이터 정보를 확인할 뿐만 아니라 최신 데이터를 복사하여 로컬 저장소에 가져온다. `git fetch` 후 `git merge`를 해주는 격이다.

`git fetch`는 마지막 pull 이후 원격 저장소 또는 브랜치에 적용된 변경 사항을 확인할 수 있다. 만일 원격 저장소에 변경 사항이 존재하는 상황에서 pull을 바로 실행하면 현재 브랜치와 작업 복사본의 파일이 변경되는 동시에 새로 작업한 내용이 손실되는 일이 생길 수 있다.

따라서 `fetch`로 변경 사항을 먼저 확인한 후 pull을 실행하는 방법이 보다 안전하다.

`git pull = git fetch + merge`

```
$ git fetch
```

```
$ git diff ...origin
```

5. 원격 저장소를 로컬 저장소에 복사

git clone

github의 원격 저장소를 로컬 저장소로 복사하는 작업을 의미한다.

git clone = git init + git remote origin origin + git pull

github 저장소에서 local 저장소로 새로 파일을 받아와 프로젝트 작업을 시작할 때 사용한다.

```
$ pwd
```

```
/c/work
```

```
$ mkdir git_clone
```

```
$ cd git_clone/
```

```
$ git clone https://github.com/pyk202056/study.git
```

```
Cloning into 'study'...
```

```
remote: Enumerating objects: 3, done.
```

```
remote: Counting objects: 100% (3/3), done.
```

```
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
```

```
Receiving objects: 100% (3/3), done.
```

```
$ ls
```

```
study/
```

```
$ cd study/
```

```
tom@DESKTOP-DCAF7UT MINGW64 /c/work/git_clone/study (main)
```

```
$ ls
```

```
README.md happy.txt hello.txt world.txt
```

```
$ git status
```

```
On branch main
```

```
Your branch is up to date with 'origin/main'.
```

```
nothing to commit, working tree clean
```

HTTPS

SSH

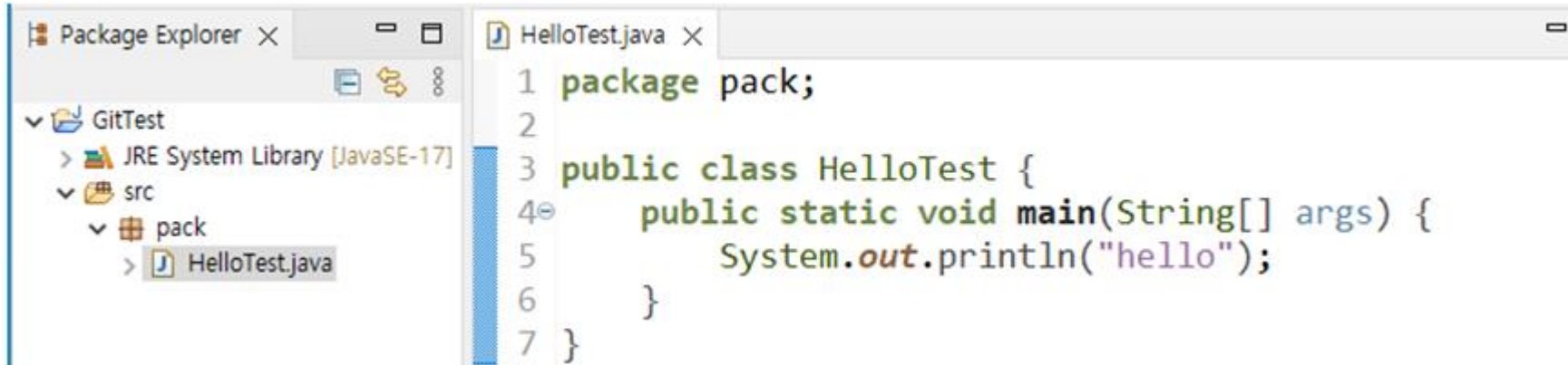
GitHub CLI

<https://github.com/pyk202056/study.git>



Eclipse에서 Git/Github 사용

1) 프로젝트 생성 후 HelloTest.java 작성

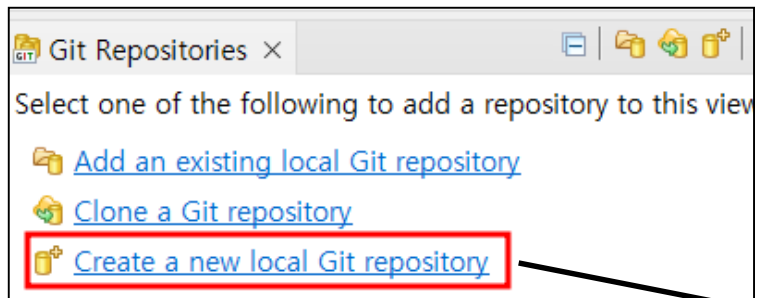
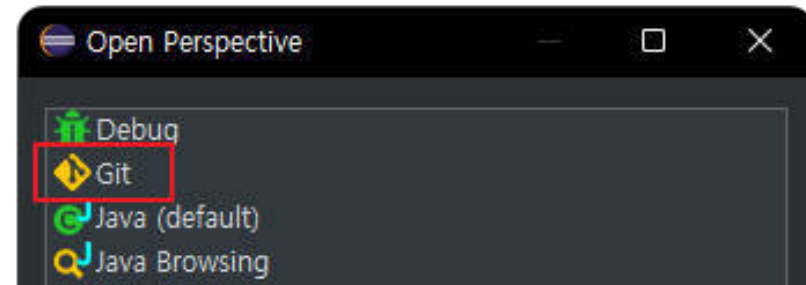
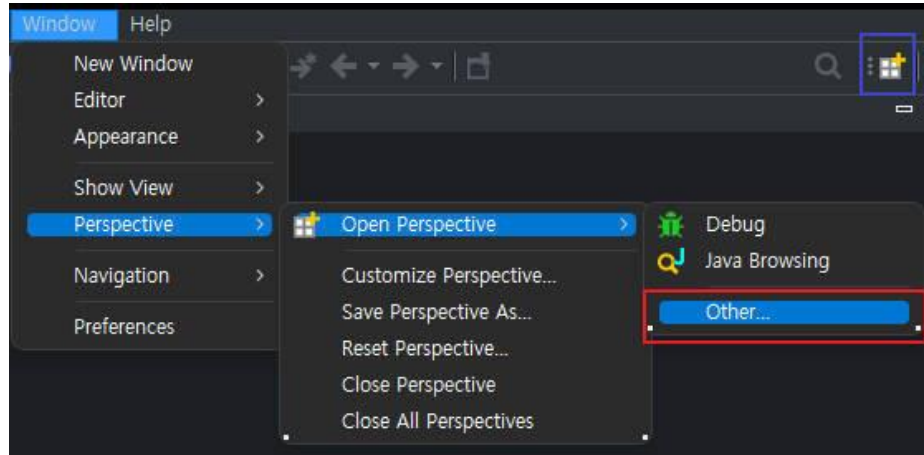


The screenshot shows an IDE interface. On the left, the Package Explorer displays a project named 'GitTest' with a source folder 'src' containing a package 'pack' and a file 'HelloTest.java'. On the right, the 'HelloTest.java' file is open, showing the following code:

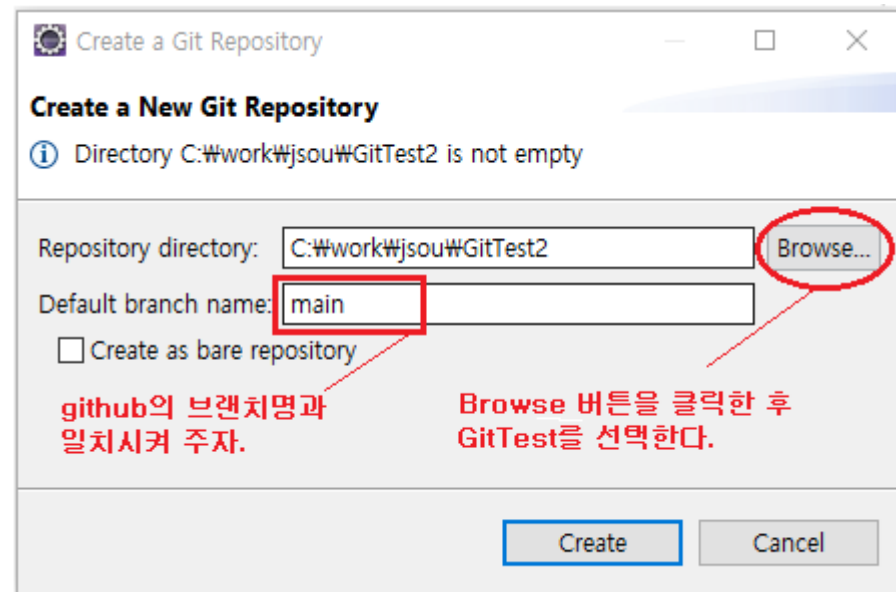
```
1 package pack;
2
3 public class HelloTest {
4     public static void main(String[] args) {
5         System.out.println("hello");
6     }
7 }
```

1. 로컬 저장소 생성 및 초기화

2) Git 작업 영역을 열고, 로컬 저장소 생성 및 초기화



github에 repository를 이미 만들어 두었으면 Clone a Git repository를 선택한다.
github에 Repository를 생성하지 않았다면 Create a new local Git repository를 통해 생성할 수 있다.



1. 로컬 저장소 생성 및 초기화

The screenshot displays the Eclipse IDE interface with several views open to illustrate the initial setup of a local Git repository.

- Git Repositories View (Top Left):** Shows the local repository structure. The 'Local' branch is highlighted with a blue circle. The 'Working Tree' is also visible, showing the current state of the files. A yellow circle highlights the '.git', '.settings', 'bin', 'src', '.classpath', '.gitignore', and '.project' files in the 'Working Tree'.
- Package Explorer (Bottom Left):** Shows the project structure. The 'src' folder is expanded, showing the 'pack' package and the 'HelloTest.java' file. A yellow circle highlights the 'pack' package and 'HelloTest.java' file.
- Git Staging View (Bottom Center):** Shows the 'Unstaged Changes (7)' and 'Staged Changes (0)'. The 'Unstaged Changes' list includes '.classpath', '.gitignore', '.project', 'HelloTest.java - src/pack', and 'module-info.java - src'. The 'Staged Changes' list is empty.
- Commit Message View (Bottom Right):** Shows the 'Commit Message' field with the text 'Unborn branch: this commit will create the branch 'master''. Below the message field are fields for 'Author' and 'Committer', both set to 'pykwon <abc@abc.com>'. At the bottom are buttons for 'Commit and Push...' and 'Commit'.

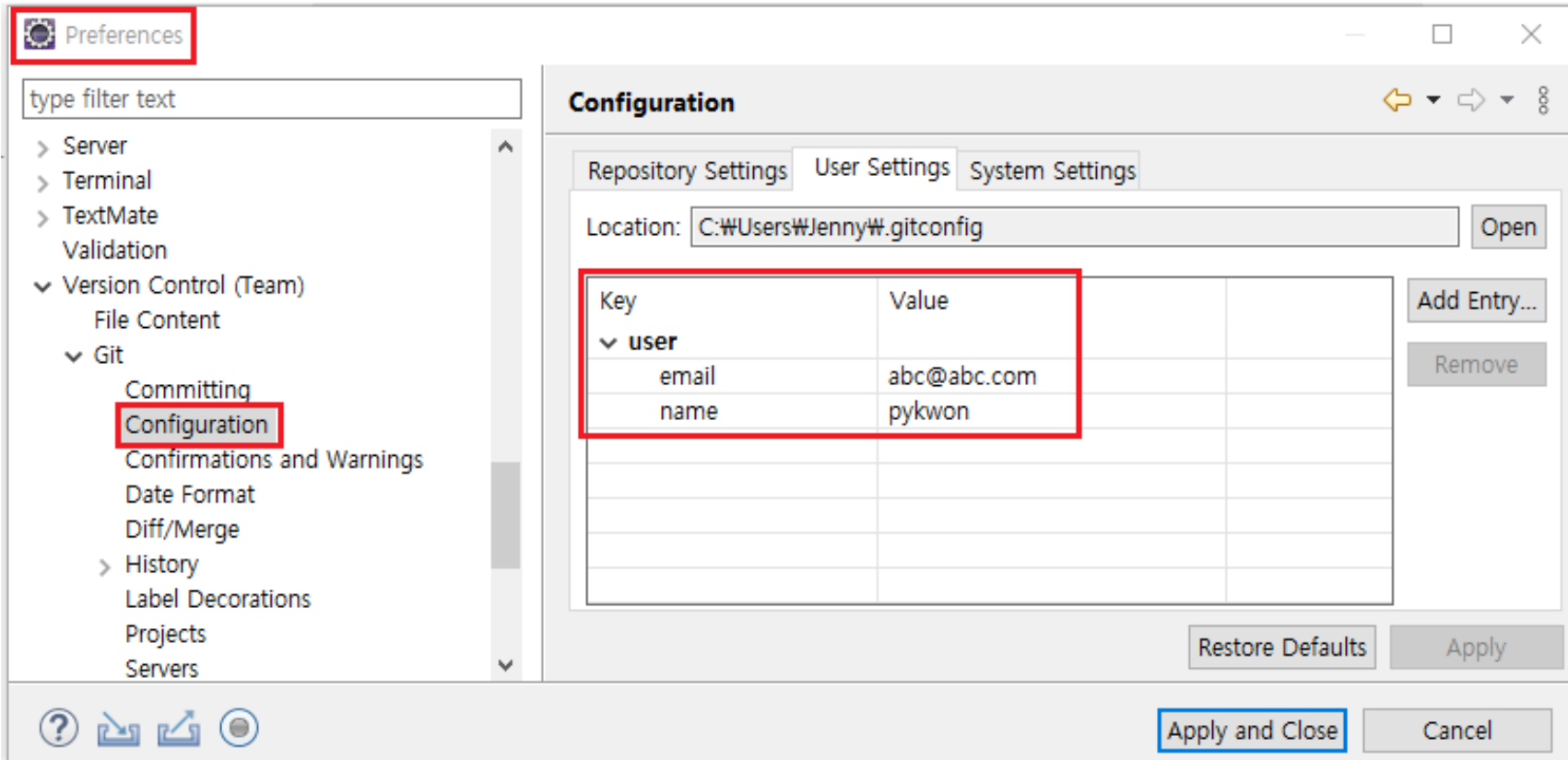
The main editor window shows the code for 'HelloTest.java' in the 'pack' package:

```
1 package pack;
2
3 public class HelloTest {
4     public static void main(String[] args) {
5         System.out.println("hello");
6     }
7 }
8
9
```

2. Git 환경 설정

1) 사용자 정보

버전을 저장할 때마다 그 버전을 만든 사용자 정보도 함께 저장된다.



```
$ git config user.name  
pykwon  
$ git config user.email  
abc@abc.com
```

2. Git 환경 설정

2) 불필요한 파일이나 디렉토리는 git 처리에서 제외하기(.gitignore)

저장 및 추적할 필요가 없는 부수적인 파일들 목록을 만들어서 제외 시키는 방법

<https://www.gitignore.io>

gitignore.io

자신의 프로젝트에 꼭 맞는 .gitignore 파일을 만드세요

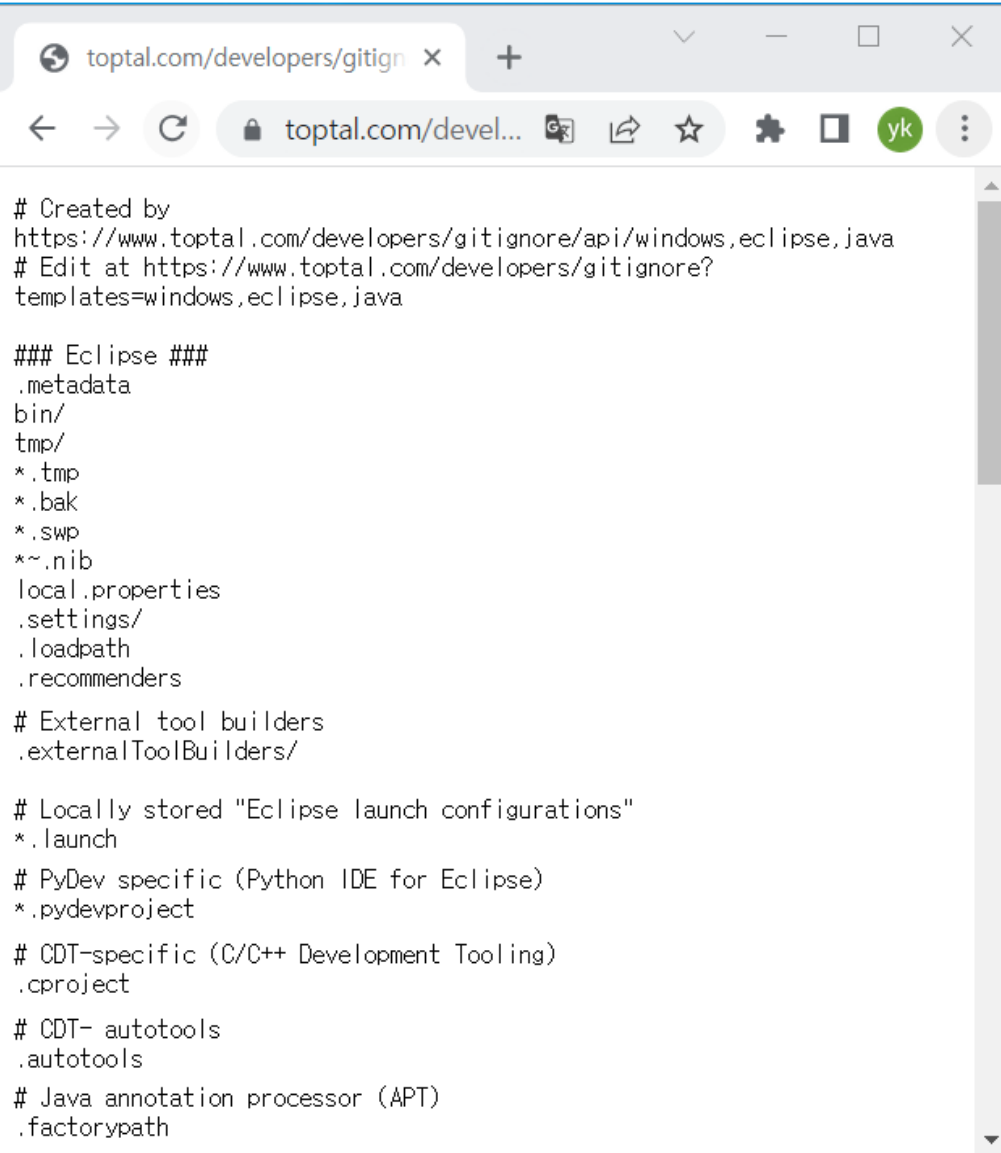
Windows ×

Eclipse ×

Java ×

생성

[소스 코드](#) | [커맨드라인 문서](#)



```
# Created by
https://www.toptal.com/developers/gitignore/api/windows,eclipse,java
# Edit at https://www.toptal.com/developers/gitignore?
templates=windows,eclipse,java

### Eclipse ###
.metadata
bin/
tmp/
*.tmp
*.bak
*.swp
*~.nib
local.properties
.settings/
.loadpath
.recommenders

# External tool builders
.externalToolBuilders/

# Locally stored "Eclipse launch configurations"
*.launch

# PyDev specific (Python IDE for Eclipse)
*.pydevproject

# CDT-specific (C/C++ Development Tooling)
.project

# CDT- autotools
.autotools

# Java annotation processor (APT)
.factorypath
```

2. Git 환경 설정

gitignore.io에서 만들어진 내용을 .gitignore 파일에 복사한다. 추가적으로 제외할 것이 있다면 적어 준다.

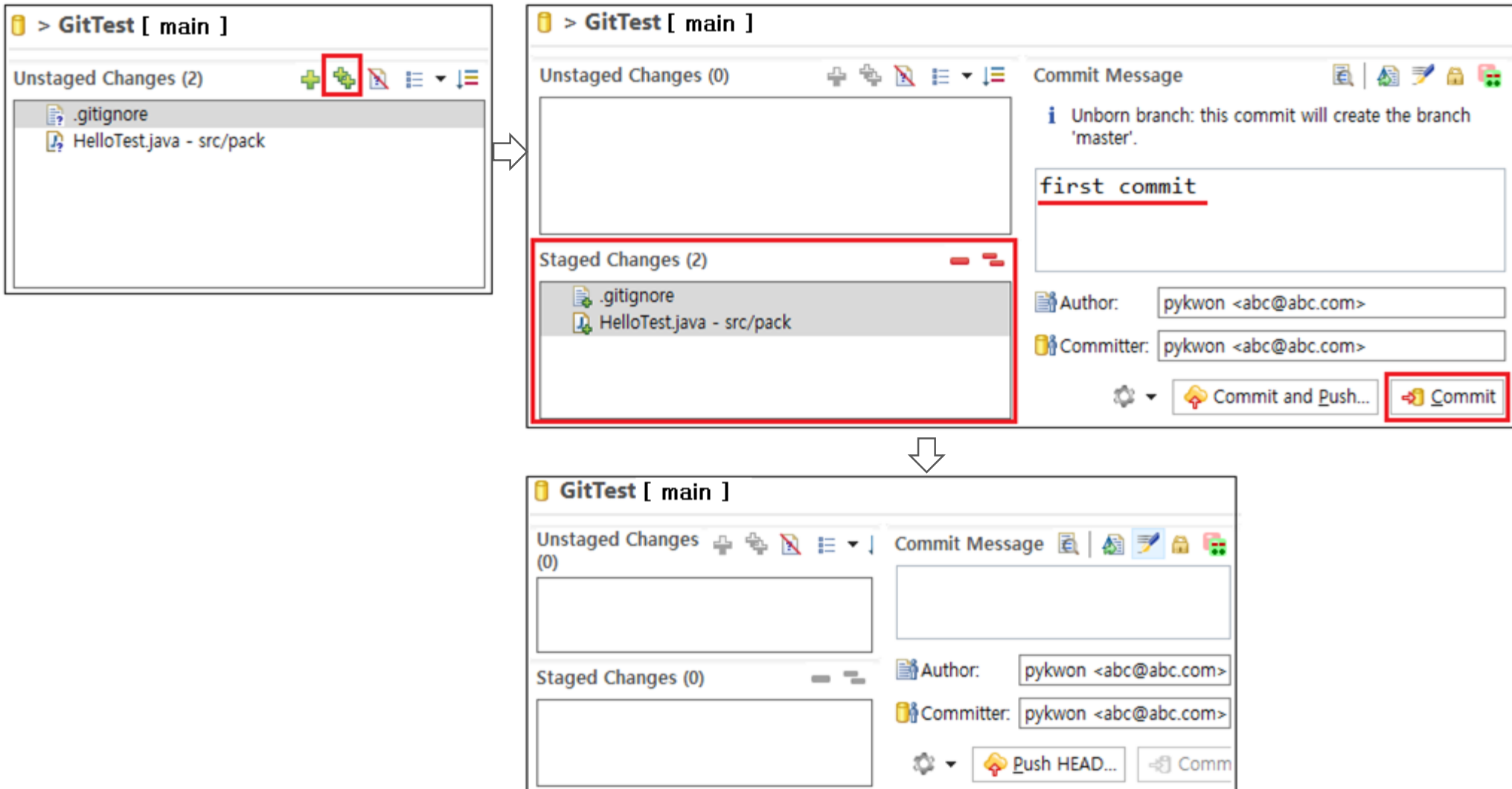
The screenshot illustrates the steps to configure a .gitignore file in an IDE:

- Left Panel (Git Repositories):** Shows the project structure. The `.gitignore` file is highlighted in the `Working Tree`.
- Top Editor:** Displays the content of the `.gitignore` file. The first three lines are highlighted with a red box:

```
1 /bin/  
2 .classpath  
3 .project
```

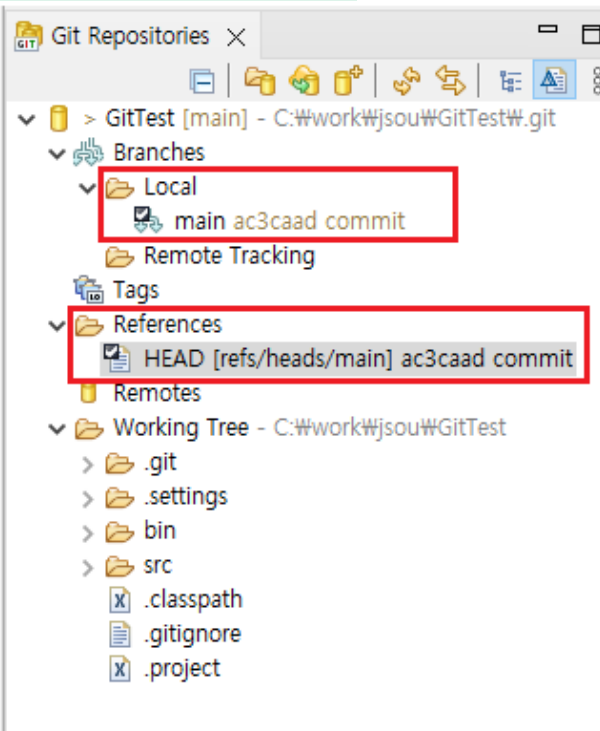
<== 이 내용을 추가한다.
- Bottom Editor (Git Staging):** Shows the `Unstaged Changes (2)` section. The `.gitignore` file is listed, and a red box highlights it. A red text box states: `.gitignore` 파일을 저장하면 언급된 내용은 작업에서 제외된다.
- Right Panel (Commit Message):** Shows the commit message field and the `Commit and Push...` button.

3. 저장소에 추적 파일 등록 및 제출

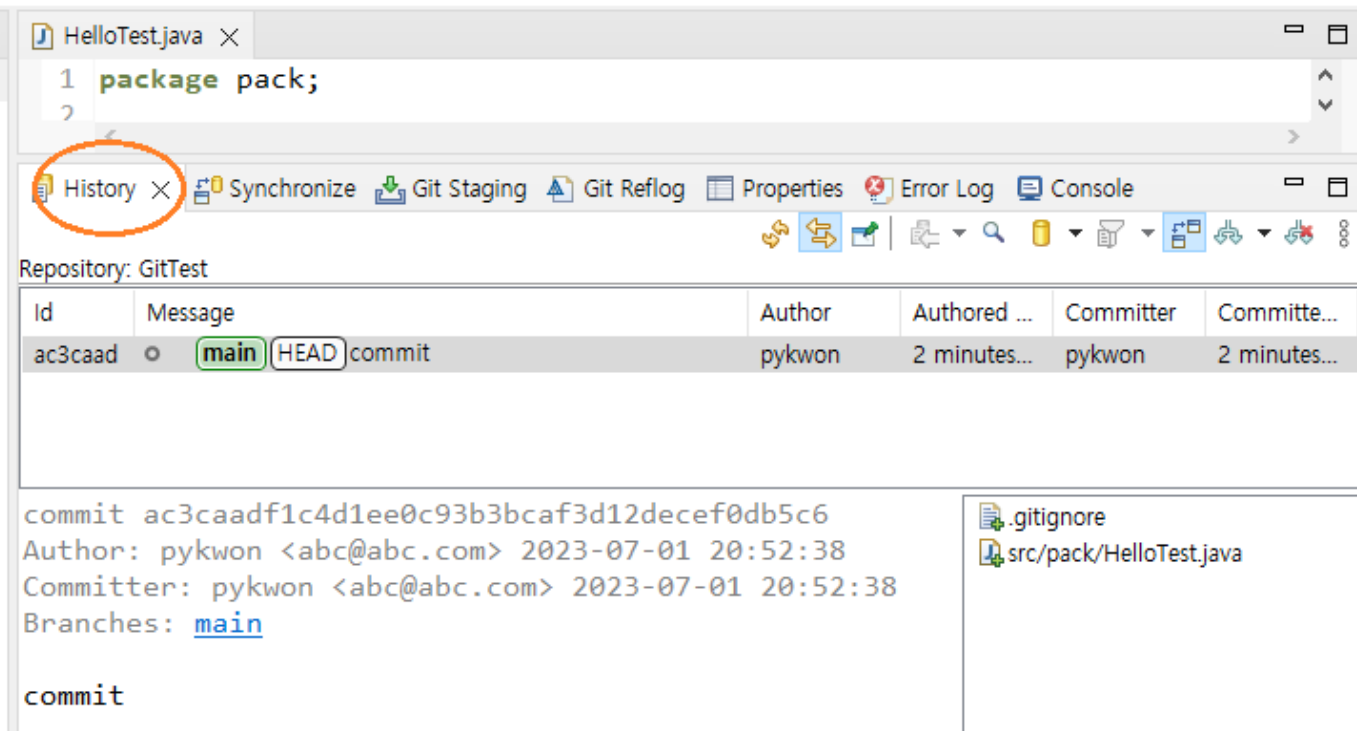


4. 등록된 버전 확인

Git Repositories



History



5. 여러 버전 생성

HelloTest.java 변경

The screenshot shows the Eclipse IDE interface. The top editor displays the `HelloTest.java` file with the following code:

```
1 package pack;  
2  
3 public class HelloTest {  
4     public static void main(String[] args) {  
5         System.out.println("hello");  
6         System.out.println("hello");  
7     }  
8 }  
9
```

The `System.out.println("hello");` line on line 6 is highlighted with a red rectangle. Below the editor, the **Git Staging** tab is selected and circled in orange. It shows the following structure:

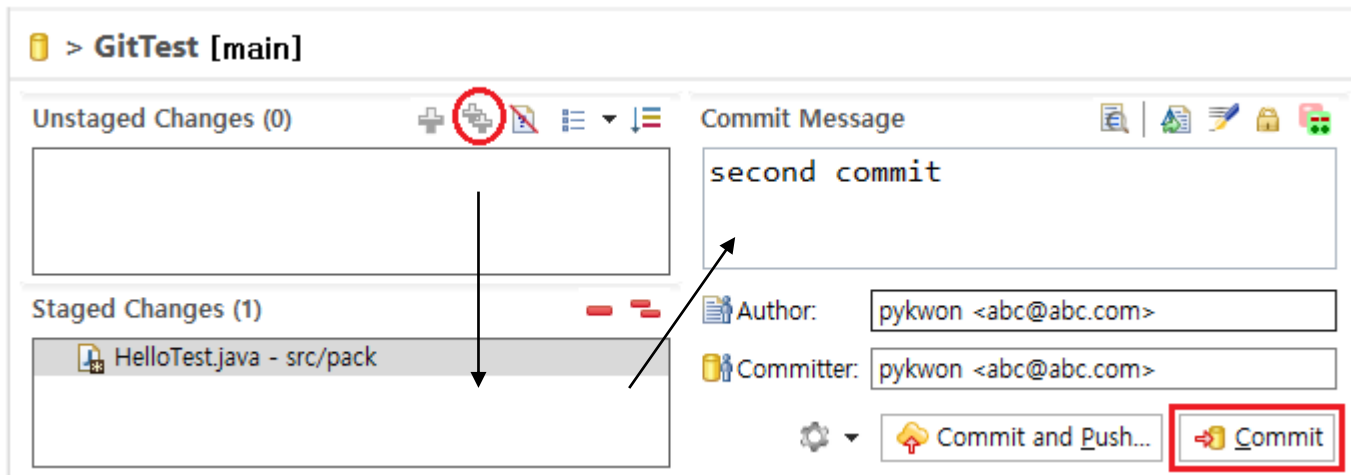
- Unstaged Changes (1)**:
 - `> HelloTest.java - src/pack`
- Staged Changes (0)**: (Empty list)

On the right side of the **Git Staging** tab, the **Commit Message** field is empty. Below it, the **Author** and **Committer** fields are both set to `pykwon <abc@abc.com>`. At the bottom right, there are buttons for **Push HEAD...** and **Commit**.

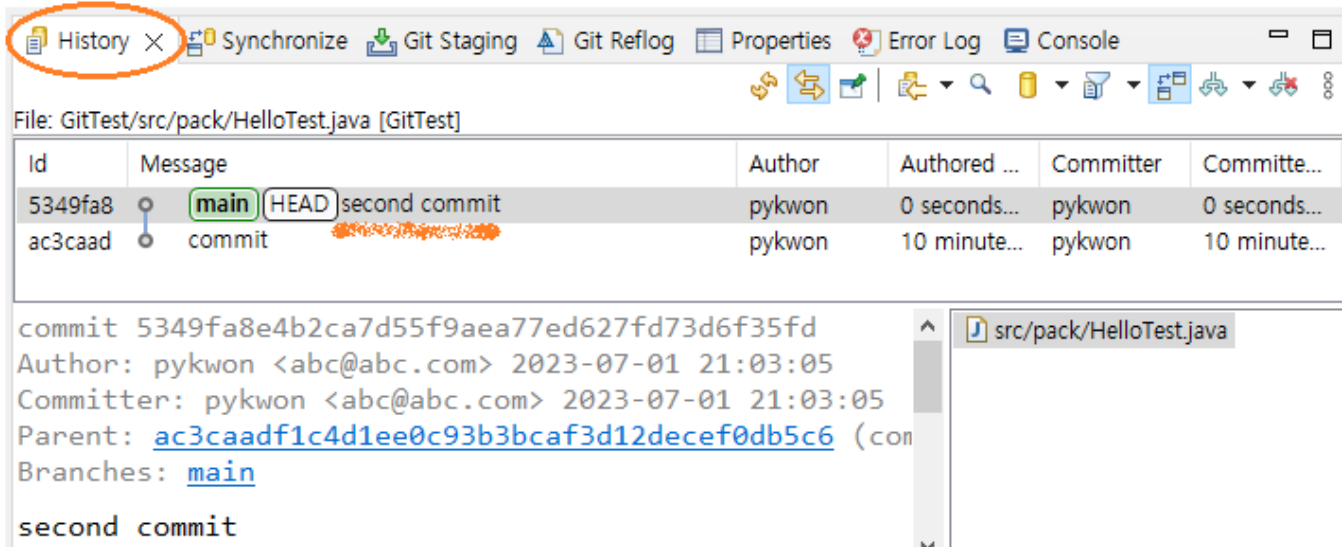
코드가 변경되면 Git Staging 탭에 수정된 파일이 보여진다.

5. 여러 버전 생성

이후에는 계속적으로 commit 까지 반복 작업한다.



History 탭



5. 여러 버전 생성

HelloTest.java 변경내용 비교

Select a Commit
2 commits in repository GitTest
Please select a commit from the list

Id	Message	Author	Authored ...	Committer	Committe...
5349fa8	main HEAD second commit	pykwon	3 minutes...	pykwon	3 minutes...
ac3caad	commit	pykwon	13 minute...	pykwon	13 minute...

Compare HelloTest.java Current and 99c4dff

```
Local: HelloTest.java
1 package pack;
2
3 public class HelloTest {
4     public static void main(String[] args) {
5         System.out.println("hello");
6         System.out.println("hello");
7     }
8 }
```

HelloTest.java 99c4dff (pykwon)

```
1 package pack;
2
3 public class HelloTest {
4     public static void main(String[] args) {
5         System.out.println("hello");
6     }
7 }
8 }
```

Git Reflog

Id	Message	Author	Authored Date	Committer	Committed Date
bebb35c	main HEAD second commit	pykwon	15 minutes ago	pykwon	15 minutes ago
99c4dff	first commit	pykwon	46 minutes ago	pykwon	46 minutes ago

commit bebb35cb8838b858fd0ef1064873edd730e23a76
Author: pykwon <abc@abc.com> 2023-06-29 15:01:18
Committer: pykwon <abc@abc.com> 2023-06-29 15:01:18
Parent: 99c4dff5e2dc0f1352c58c04932dbfc094d52cae (first commit)
Branches: master

6. 작업 되돌리기

특정 commit 으로 되돌리기

File: GitTest/src/pack/HelloTest.java

Id	Message	Author	Authored Date	Committer	Committed Date
bebb35c	master HEAD	pykwon	29 minutes...	pykwon	29 minutes...
99c4dff	first commit	pykwon	60 minutes...	pykwon	60 minutes...

오른쪽 버튼 <== 되돌리고 싶은 버전을 선택!

File: GitTest/src/pack/HelloTest.java [GitTest]

Id	Message	Author	Authored Date	Committer	Committed Date
ac3caad	main HEAD commit	pykwon	24 minutes ...	pykwon	24 minutes ...

Git Repositories x

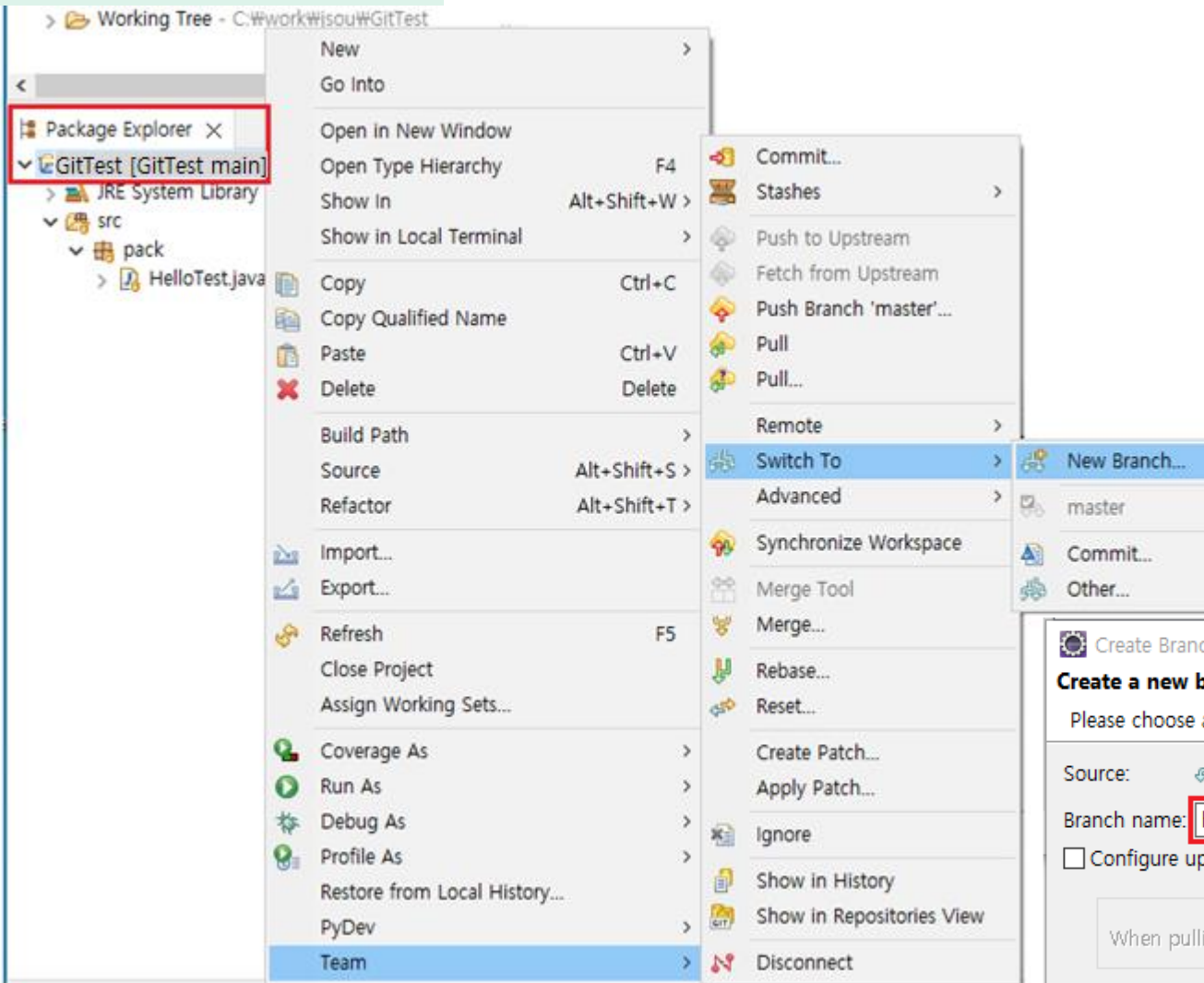
- GitTest [main] - C:\work\jsou\GitTest\#.git
 - Branches
 - Tags
 - References
 - HEAD [refs/heads/main] ac3caad commit
 - ORIG_HEAD e04f03c second commit
 - Remotes
 - Working Tree - C:\work\jsou\GitTest

HelloTest.java x

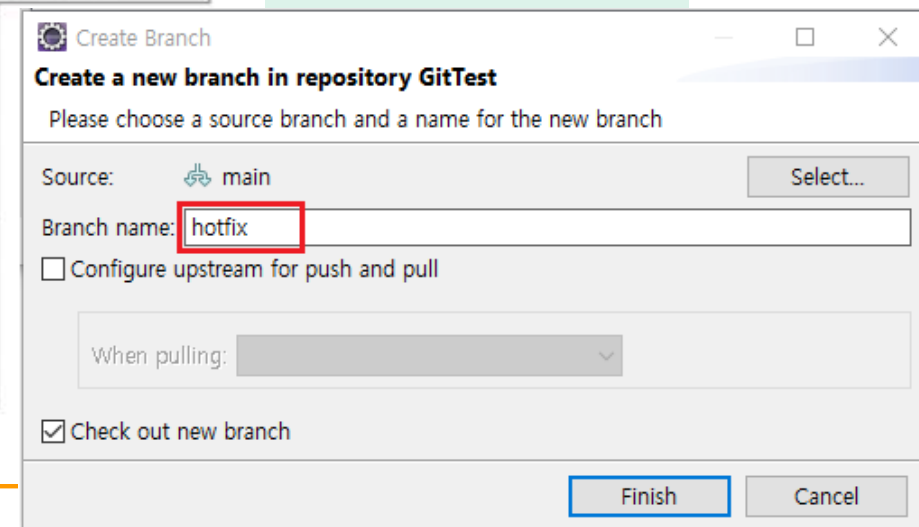
```
1 package pack;
2
3 public class HelloTest {
4     public static void main(String[] args) {
5         System.out.println("hello");
6     }
7 }
8
```

7. 브랜치 생성

New Branch ... 선택



hotfix 브랜치 생성



7. 브랜치 생성

The screenshot shows the Eclipse IDE interface with the following components:

- Git Repositories:** Shows the 'GitTest' repository. Under 'Local', the 'hotfix' branch is highlighted with a red box, indicating it is the current branch. The 'main' branch is also visible.
- Package Explorer:** Shows the project structure. The 'GitTest [GitTest hotfix]' project is selected. The 'src' folder contains a 'pack' folder, which contains the 'HelloTest.java' file.
- History:** Shows the commit history for the 'GitTest' repository. The 'hotfix' branch is highlighted with a green box. The commit details show the commit message 'hotfix main HEAD commit' and the commit hash 'ac3caad'.

```
1 package pack;
2
3 public class HelloTest {
4     public static void main(String[] args) {
5         System.out.println("hello");
6     }
7 }
8
```

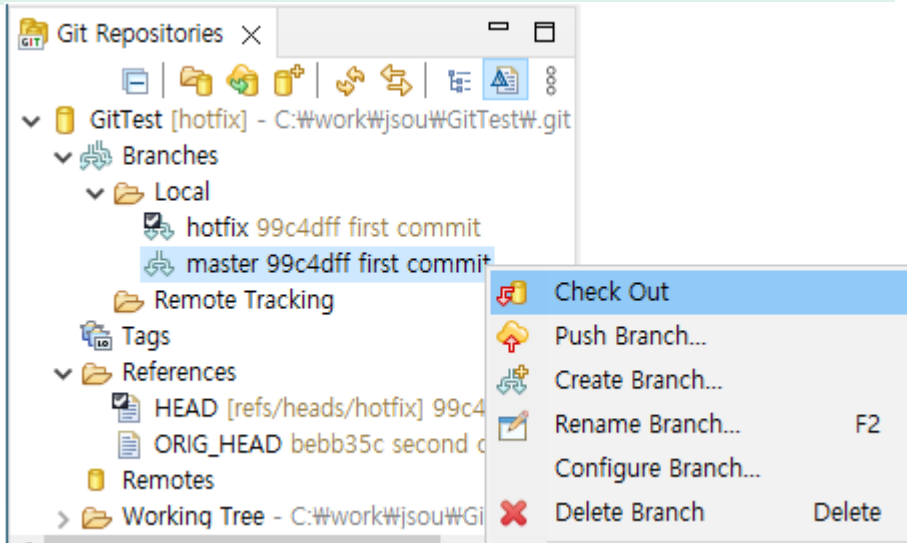
Id	Message	Author	Authored ...	Committer	Committed...
ac3caad	hotfix main HEAD commit	pykwon	37 minute...	pykwon	37 minute...

commit ac3caadf1c4d1ee0c93b3bc3d12decef0db5c6
Author: pykwon <abc@abc.com> 2023-07-01 20:52:38
Committer: pykwon <abc@abc.com> 2023-07-01 20:52:38
Branches: hotfix, main
commit

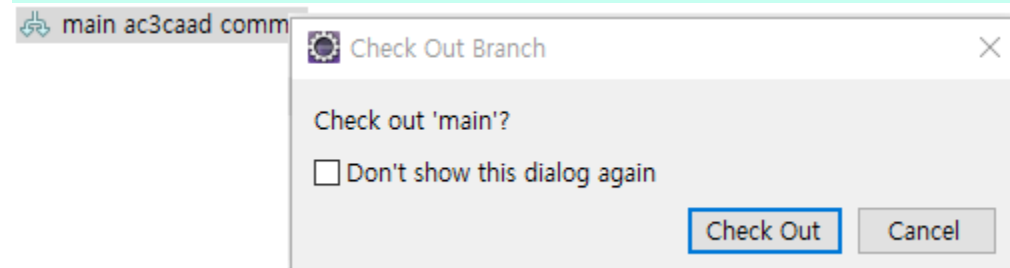
src/pack/HelloTest.java

8. 브랜치 이동

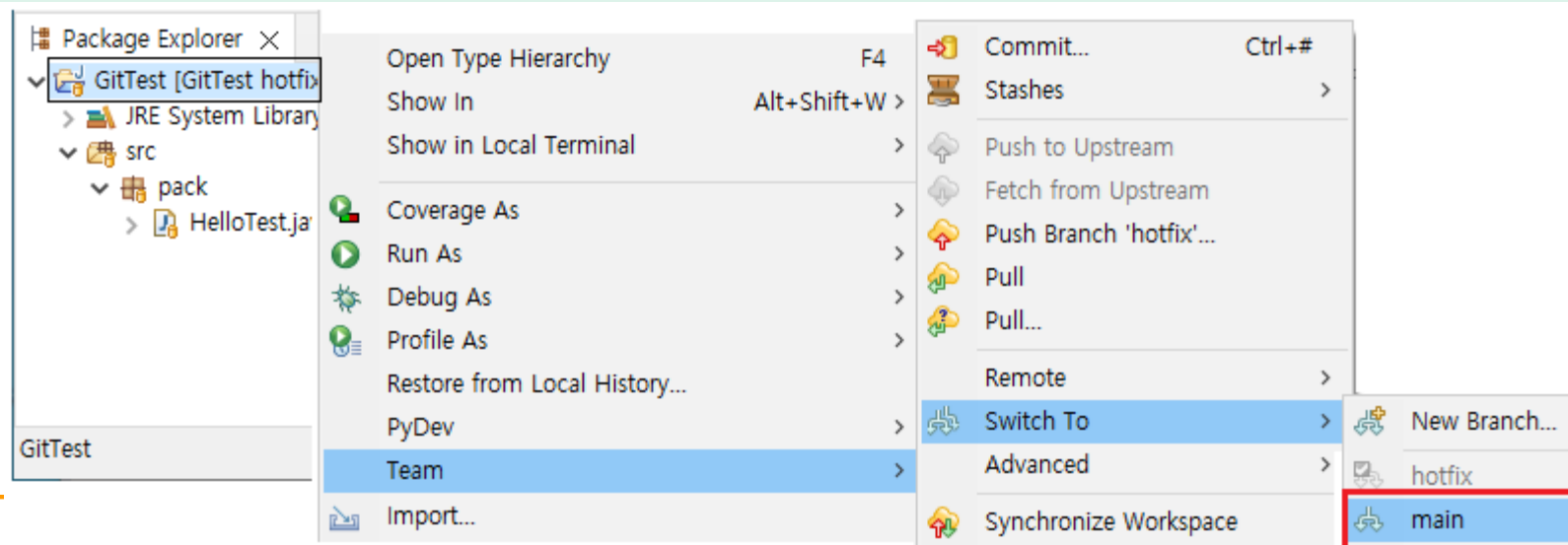
방법1 : git checkout 명령으로 브랜치 이동



특정 브랜치명에서 더블클릭해도 브랜치 이동 가능



방법2 : 해당 프로젝트에서 오른쪽버튼 -> Team -> Switch To 메뉴를 이용한 브랜치 이동



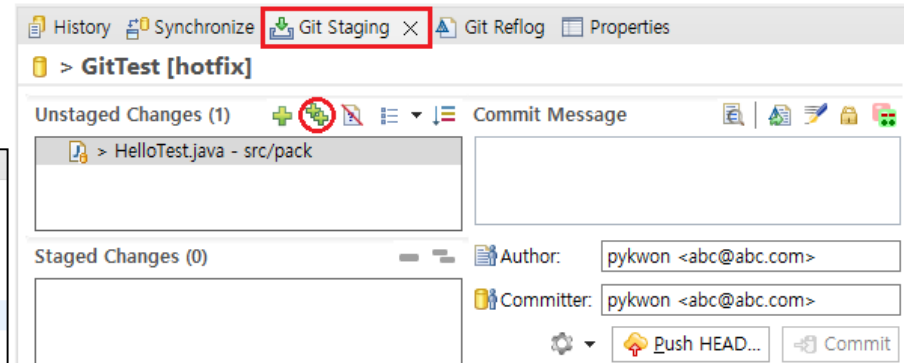
9. 브랜치 병합

main 브랜치 최종 commit 내용

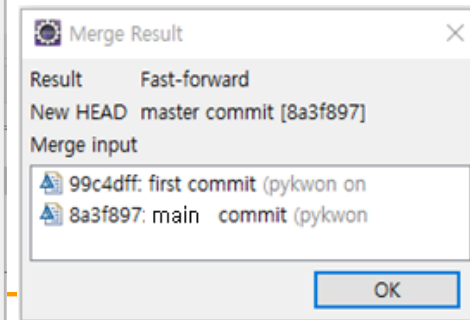
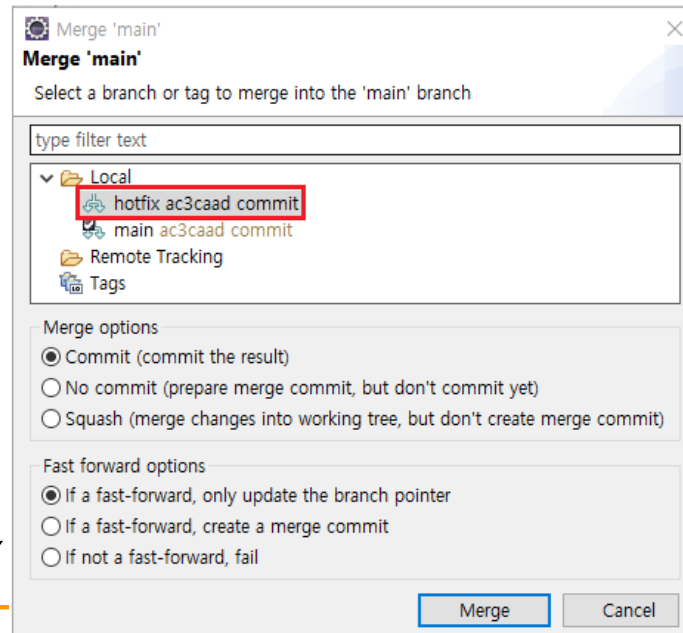
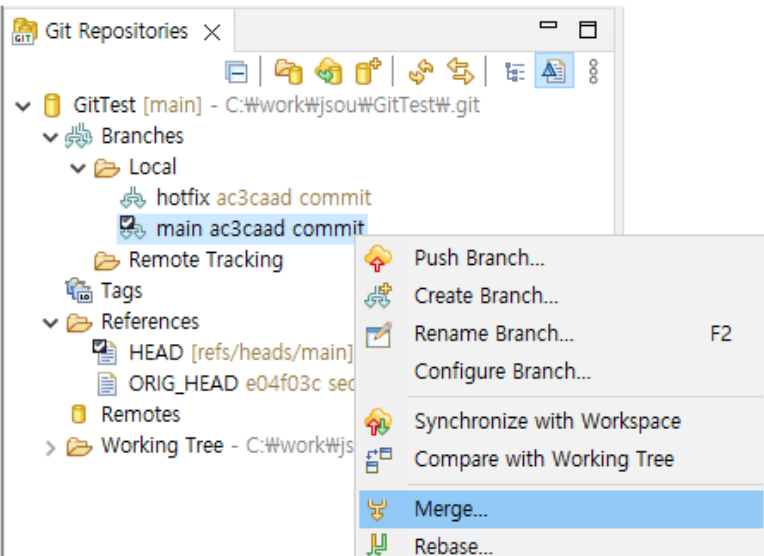
```
HelloTest.java ×
1
2 public class HelloTest {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5     }
6 }
```

```
HelloTest.java ×
1
2 public class HelloTest {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5         System.out.println("hotfix 브랜치 내용변경");
6     }
7 }
```

hotfix 브랜치 최종 commit 내용



1) main 에서 병합



9. 브랜치 병합

2) 병합 후 main 내용

main 브랜치 최종 commit 내용

Git Repositories X

GitTest [master] - C:\work\wjsou\GitTest\#

Branches

Local

hotfix 8a3f897 master commit

main 8a3f897 master commit

Remote Tracking

Tags

References

HelloTest.java X

```
1 package pack;
2
3 public class HelloTest {
4     public static void main(String[] args) {
5         System.out.println("hello");
6         System.out.println("hotfix 브랜치에서 내용 변경");
7     }
8 }
9
```

History X Synchronize Git Staging Git Reflog Properties

File: GitTest/src/pack/HelloTest.java [GitTest]

Id	Message	Author	Authored Date	Committer	Committed Date
8a3f897	hotfix master HEAD master commit	pykwon	17 minutes ago	pykwon	17 minutes ago
aec2274	hotfix 브랜치에서 내용 변경 커밋	pykwon	25 minutes ago	pykwon	25 minutes ago
99c4dff	first commit	pykwon	3 hours ago	pykwon	3 hours ago

10. 브랜치 병합시 충돌

main 브랜치 최종 commit 내용

Git Repositories

- GitTest [master] - C:\work\jsou\GitTest
- Branches
 - Local
 - hotfix 8a3f897 master commit
 - main 8a3f897 master commit**
 - Remote Tracking
- Tags
- References
 - HEAD [refs/heads/master] 8a3f897
 - ORIG_HEAD bebb35c second commit
- Remotes
- Working Tree - C:\work\jsou\GitTest

Package Explorer

- GitTest [GitTest master]
- JRE System Library [JavaSE-17]
- src
 - pack
 - HelloTest.java

History Synchronize Git Staging Git Reflog Properties Filter files

> GitTest [main]

Unstaged Changes (0)

Staged Changes (1)

- HelloTest.java - src/pack

Commit Message

master 브랜치에서 내용 추가 커밋

Author: pykwon <abc@abc.com>

Committer: pykwon <abc@abc.com>

Commit and Push... **Commit**

hotfix 브랜치에서도 파일 내용 변경 후 위와 같이 최종 commit

Git Repositories

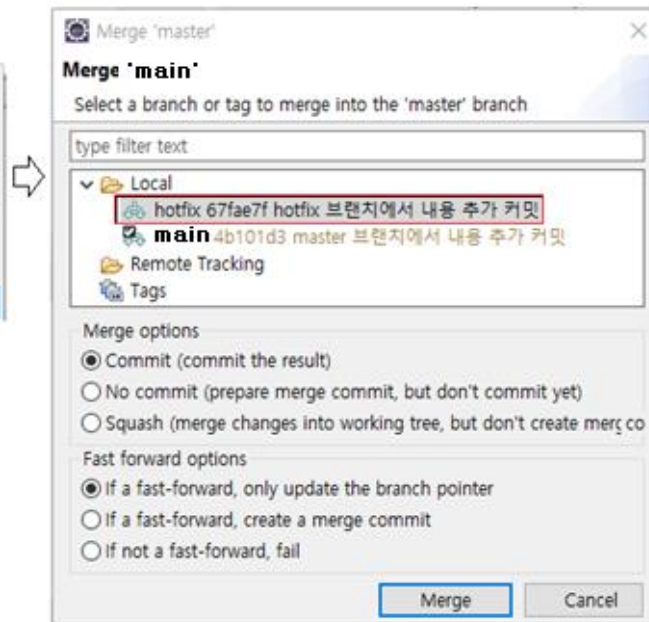
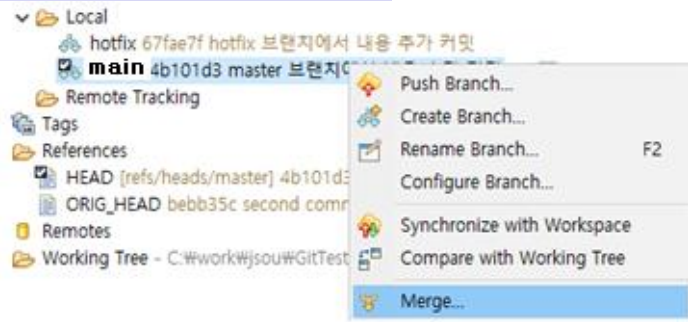
- GitTest [hotfix] - C:\work\jsou\GitTest\git
- Branches
 - Local
 - hotfix 67fae7f hotfix 브랜치에서 내용 추가 커밋
 - master 4b101d3 master 브랜치에서 내용 추가 커밋
 - Remote Tracking
- Tags

GitTest.java

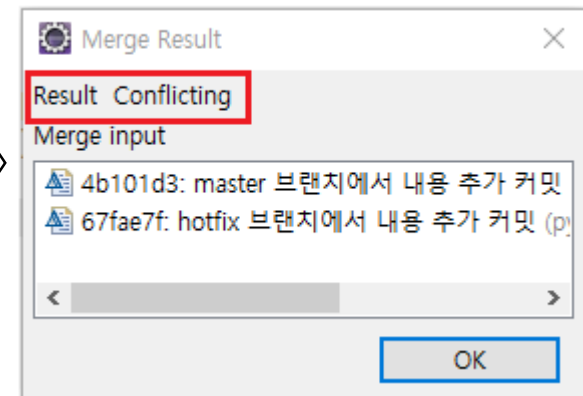
```
1 package pack;
2
3 public class HelloTest {
4     public static void main(String[] args) {
5         System.out.println("hello");
6         System.out.println("hotfix 브랜치에서 내용 변경");
7         System.out.println("hotfix 브랜치에서 내용 추가");
8     }
9 }
```

10. 브랜치 병합시 충돌

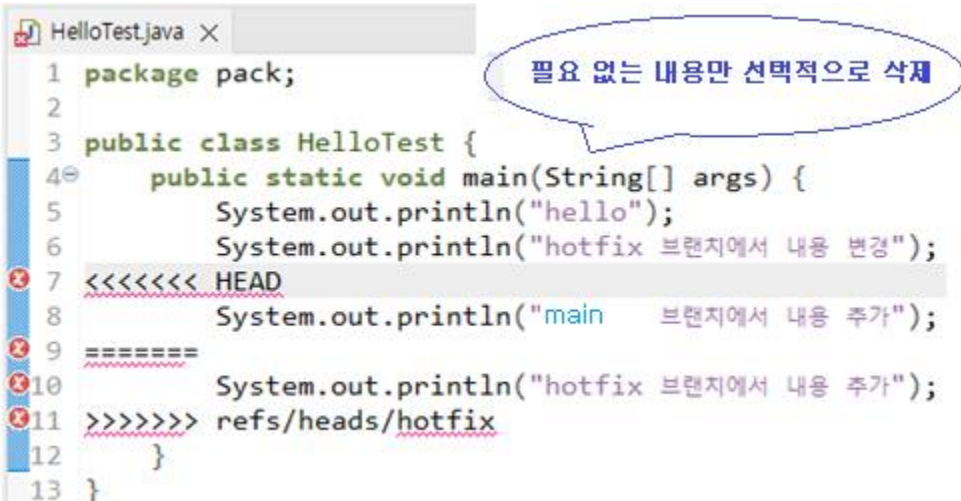
1) main 에서 병합



병합시 result conflict 문제 발생



main 브랜치 병합 결과

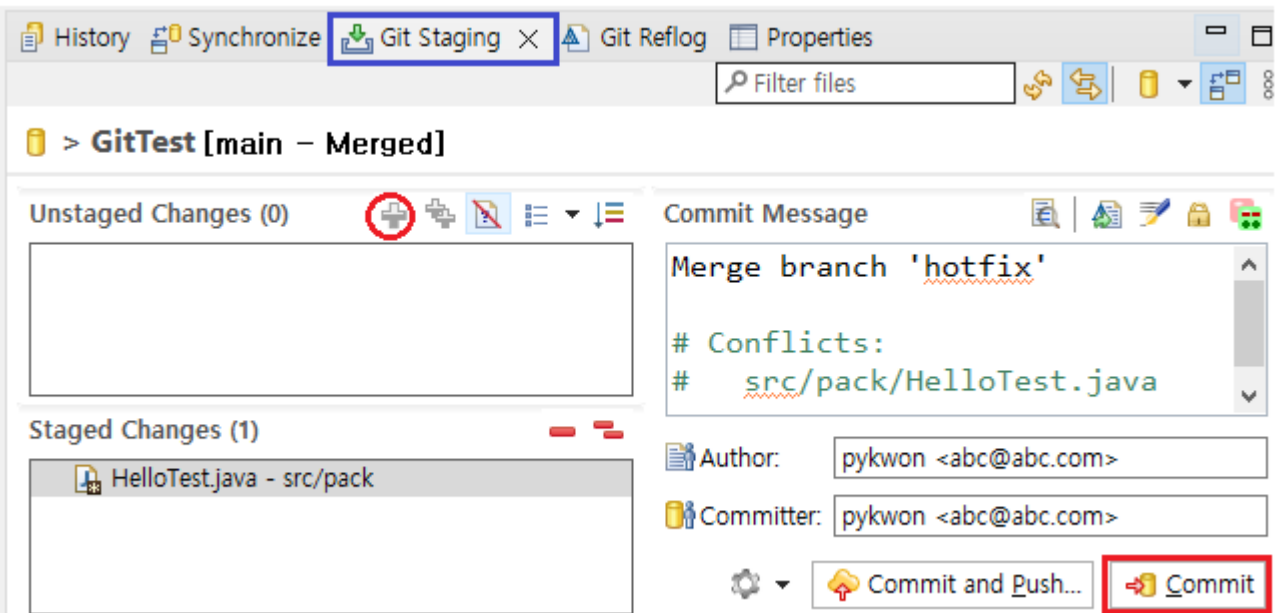


2) 충돌 해결

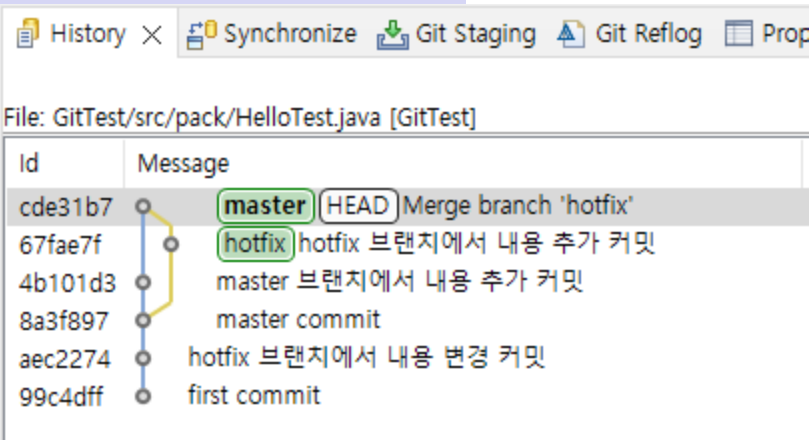


11. 브랜치 삭제

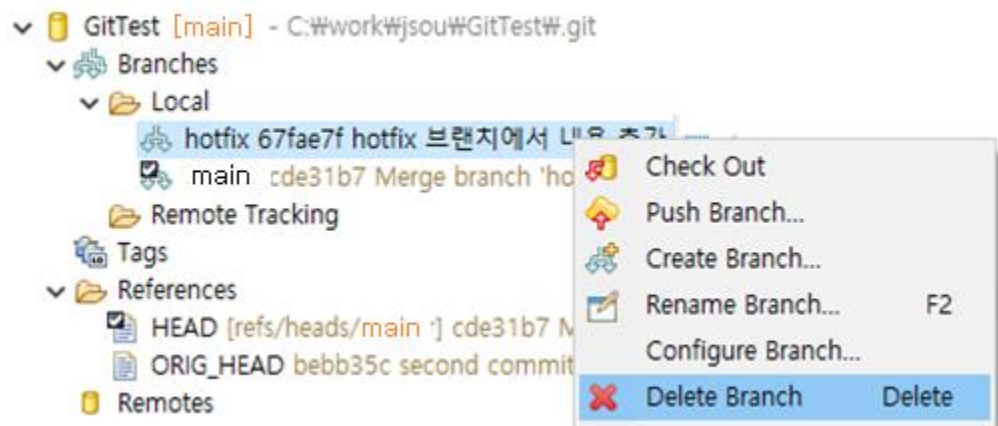
3) 수정 후 커밋을 해 준다



History 탭으로 결과 보기



hotfix 브랜치 삭제



원격 저장소 관리

1. 원격 저장소 생성

1) 원격 저장소 생성 : samplepro


Create a new repository

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

Required fields are marked with an asterisk ().*


Owner *

Repository name *

 pyk202056


/

samplepro


 samplepro is available.

Great repository names are short and memorable. Need inspiration? How about [congenial-octo-dollop](#) ?

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.

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: None

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

Choose a license

License: None

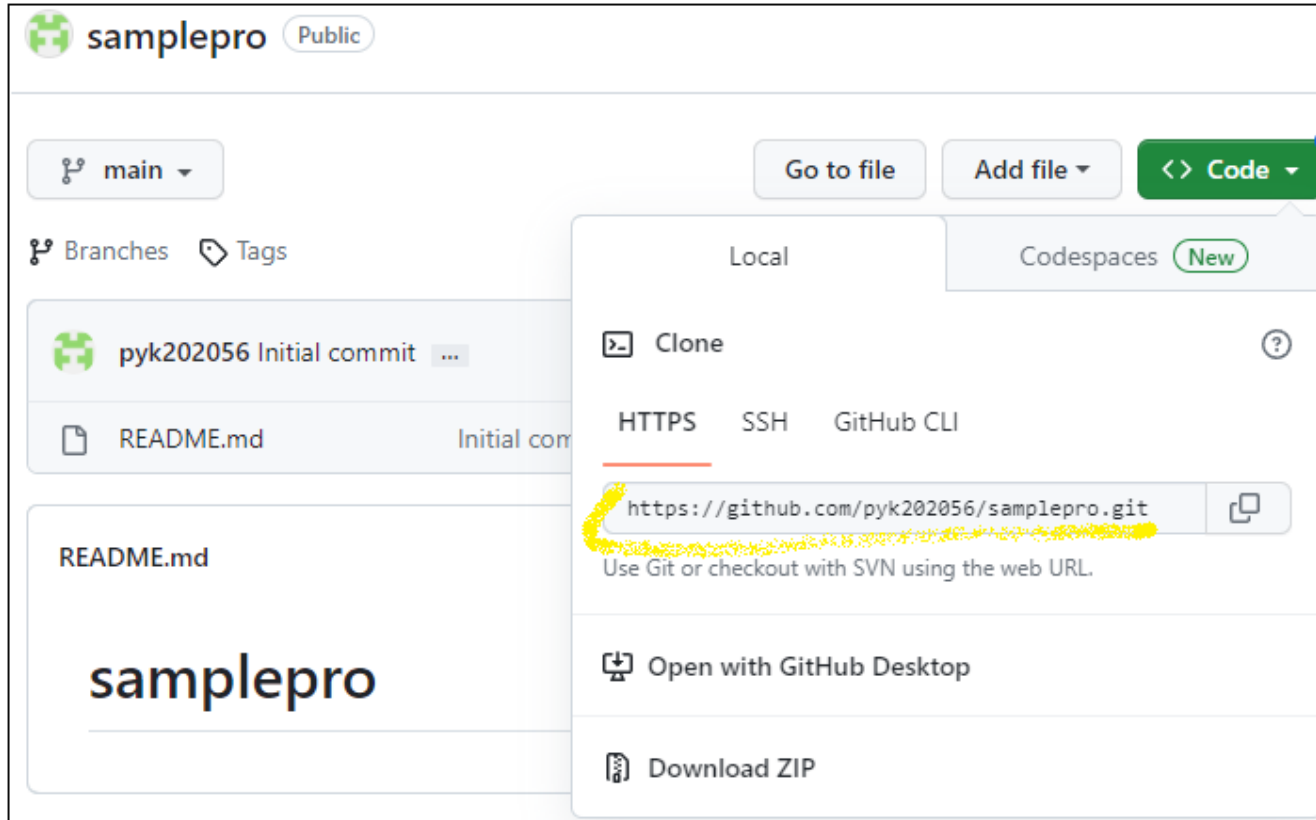
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

Create repository

1. 원격 저장소 생성

2) 원격 저장소 주소 복사



<https://github.com/pyk202056/samplepro.git>

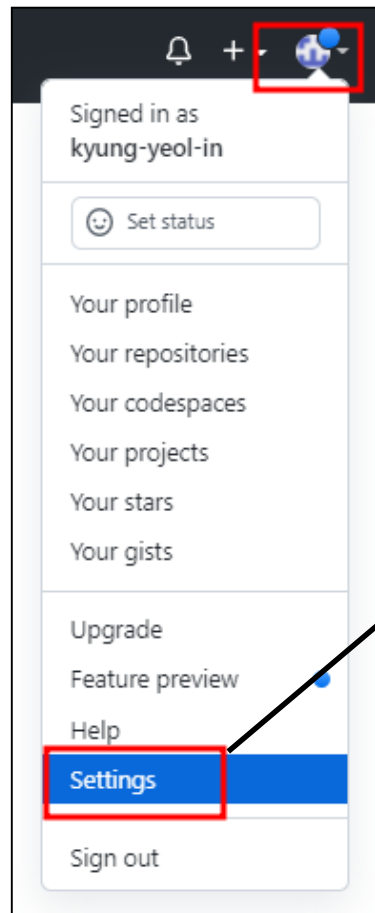
참고 : 원격 저장소를 이용한 작업 전에 원격 접속 token 을 준비해야 한다.

1. 원격 저장소 생성

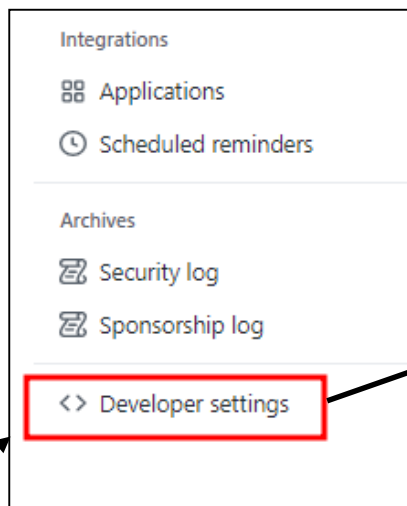
3) 원격 접속 token 얻기

“깃허브 github personal access token 발급하는 방법”

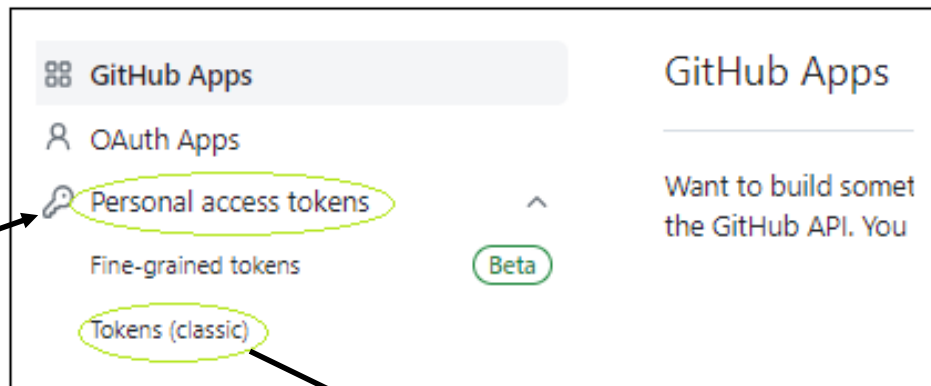
보안 강화 차원에서 비밀번호 대신 토큰을 사용한다. 개인 액세스 토큰(클래식)은 일반 OAuth 액세스 토큰처럼 작동한다. HTTPS를 통해 Git의 암호 대신 사용하거나 기본 인증을 통해 API를 인증하는 데 사용할 수 있다.



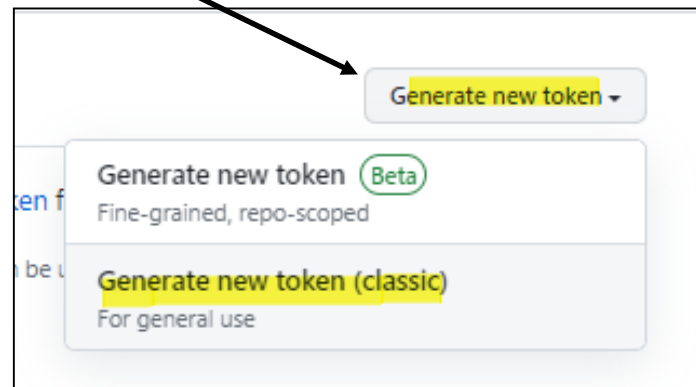
깃 허브의 프로필
아이콘을 클릭



Developer settings 화면에서 좌측 메뉴에
[Personal access tokens > Tokens(classic)]을 클릭



[Generate new token] 박스를 선택 하면
그림과 같이 선택 화면이 나오는데
[Generate new token(classic)]을 선택한다.
비밀번호 입력 창이 나오면 비밀번호를 입력한다.



1. 원격 저장소 생성

New personal access token (classic)

Personal access tokens (classic) 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

samplepro

What's this token for?

Expiration *

30 days

The token will expire on Sat, Jul 29 2023

Select scopes

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

<input checked="" type="checkbox"/> repo	Full control of private repositories	
<input checked="" type="checkbox"/> repo:status	Access commit status	
<input checked="" type="checkbox"/> repo_deployment	Access deployment status	
<input checked="" type="checkbox"/> public_repo	Access public repositories	
<input checked="" type="checkbox"/> repo:invite	Access repository invitations	
<input checked="" type="checkbox"/> security_events	Read and write security events	
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows	
<input checked="" type="checkbox"/> admin:ssh_signing_key	Full control of public user SSH signing keys	
<input checked="" type="checkbox"/> write:ssh_signing_key	Write public user SSH signing keys	
<input checked="" type="checkbox"/> read:ssh_signing_key	Read public user SSH signing keys	

Generate token

Cancel

GitHub는 정보를 안전하게 유지하는 데 도움이 되도록 토큰의 만료 날짜를 설정할 것을 강력히 권장!

항목은 필요한 것만 선택하면 된다.

Note란에는 적절하게 토큰명을 입력하고, Select scopes 체크란에는 해당 토큰에 대한 접근 범위에 대해서 체크하는 영역인데 자신의 상황에 맞게 접근 범위에 대해서 선택하면 된다. 만료일(Expiration)은 기본 30일이다.

토큰이 발급되었으면, 토큰을 복사해서 자신이 사용하고자 하는 애플리케이션에 인증하여 사용하면 된다.



Personal access tokens (classic)

Generate new token

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp_sWnMyHZNaCAfrnZAwac5Z4471w8xWa4fSZRA

Delete

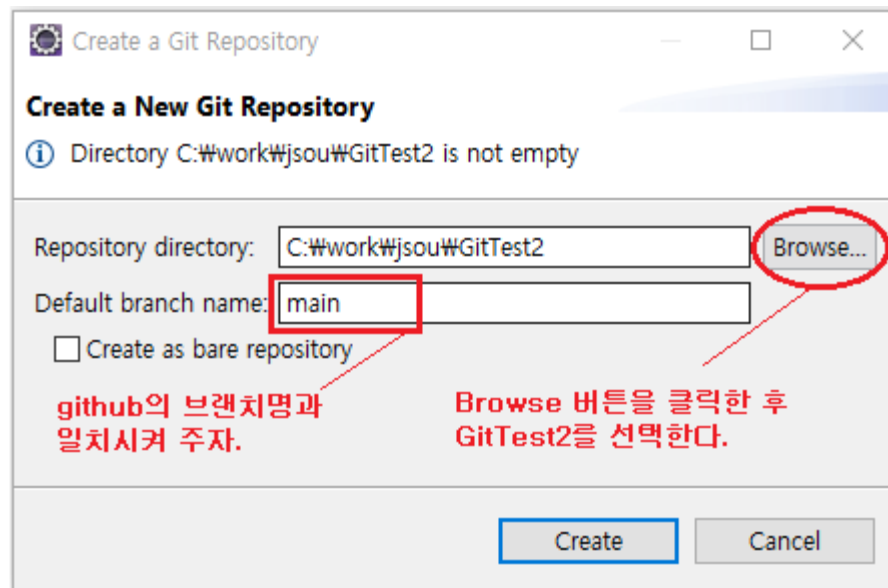
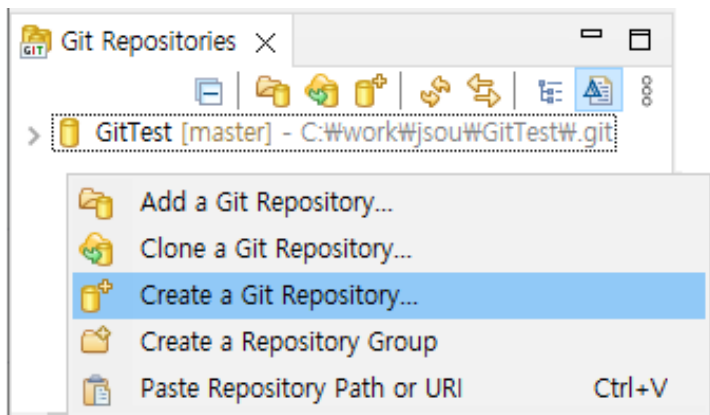
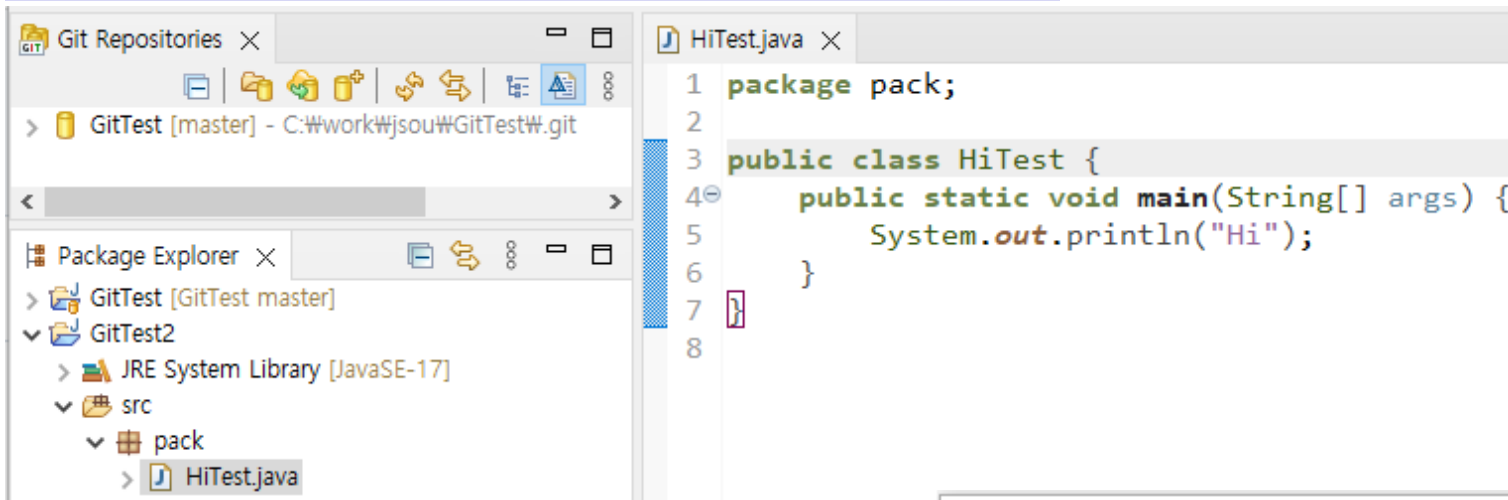
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

personal access token(classic) 토큰 발급 화면

- 생성된 토큰은 확인하고 안전한 곳에 복사해 둘 것
ghp_Jz2c2quHffyFJiykGglOcw1czbqJIP0tGXZD

2. 원격 저장소에 저장하기

1) GitTest2 자바 프로젝트 생성 및 로컬 저장소 생성



2. 원격 저장소에 저장하기

The screenshot shows an IDE with three main panels. The top-left panel, 'Git Repositories', displays a tree view of the project structure. A blue box highlights the 'Working Tree' section, and a red box around the '.gitignore' file is accompanied by the text: '이전에 작성한 .gitignore 파일의 내용을 복사한다.' (Copy the contents of the .gitignore file created previously). The top-right panel shows the 'HiTest.java' file with the following code:

```
1 package pack;
2
3 public class HiTest {
4     public static void main(String[] args) {
5         System.out.println("Hi");
6     }
7 }
8
```

The bottom-left panel, 'Package Explorer', shows the project structure with 'HiTest.java' selected in the 'src' directory of 'GitTest2'. The bottom-right panel, 'Git Test2 [main]', shows the 'Git Staging' tab. The 'Staged Changes (2)' list includes '.gitignore' and 'HiTest.java - src/pack'. The 'Commit Message' field contains 'commit'. The 'Author' and 'Committer' fields are both set to 'pykwon <abc@abc.com>'. The 'Commit and Push...' button is highlighted with a red box.

2. 원격 저장소에 저장하기

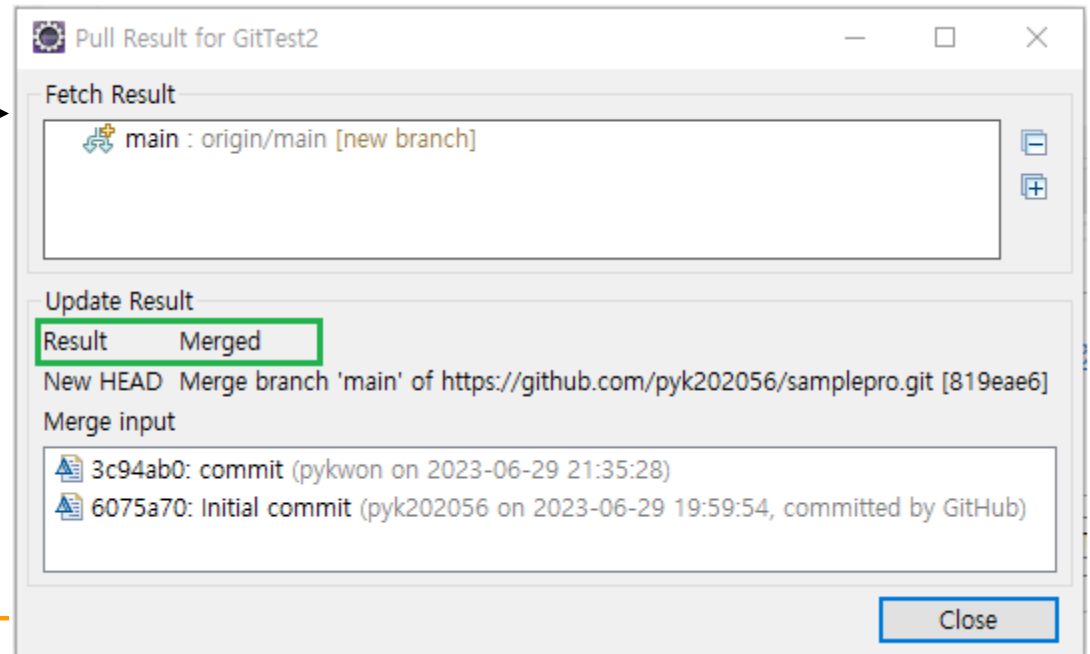
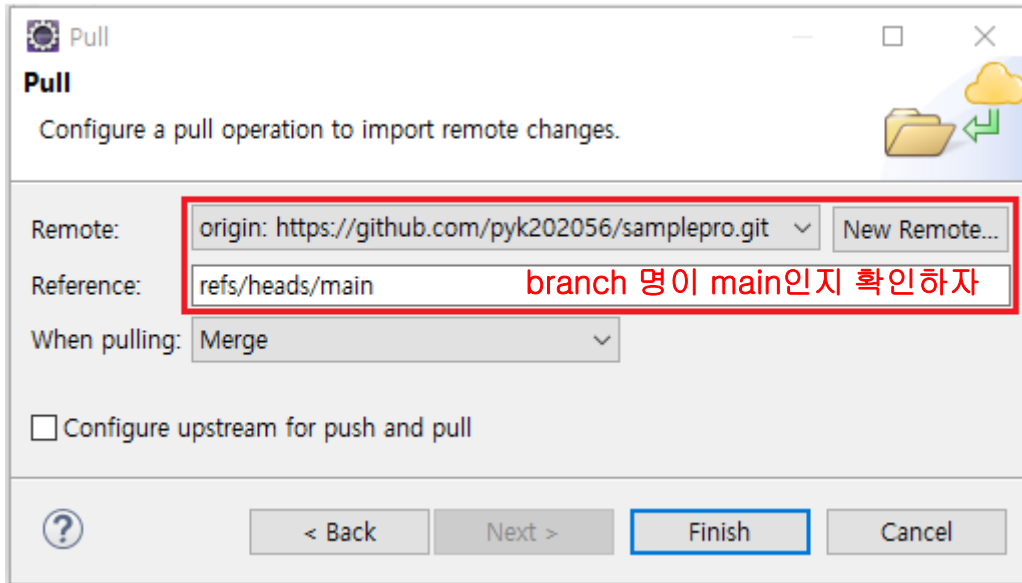
2) 원격 저장소에 저장하기 전 반드시 pull 먼저 하기

The screenshot illustrates the process of pulling code from a remote repository in Eclipse. On the left, the Package Explorer shows a project named 'GitTest2' with a red box around it. A right-click context menu is open, and the 'Pull...' option is highlighted. An arrow points from this option to the 'Pull' dialog box on the right. The dialog box is titled 'Pull' and contains the following fields:

- Destination Git Repository**: Enter the location of the destination repository.
- Remote name**: origin
- Location**:
 - URI**: https://github.com/pyk202056/samplepro.git (highlighted with a red box)
 - Host**: github.com
 - Repository path**: /pyk202056/samplepro.git
- Connection**:
 - Protocol**: https
 - Port**: (empty)
- Authentication**:
 - User**: pykwon
 - Password**: (masked with dots)
 - ☒ Store in Secure Store

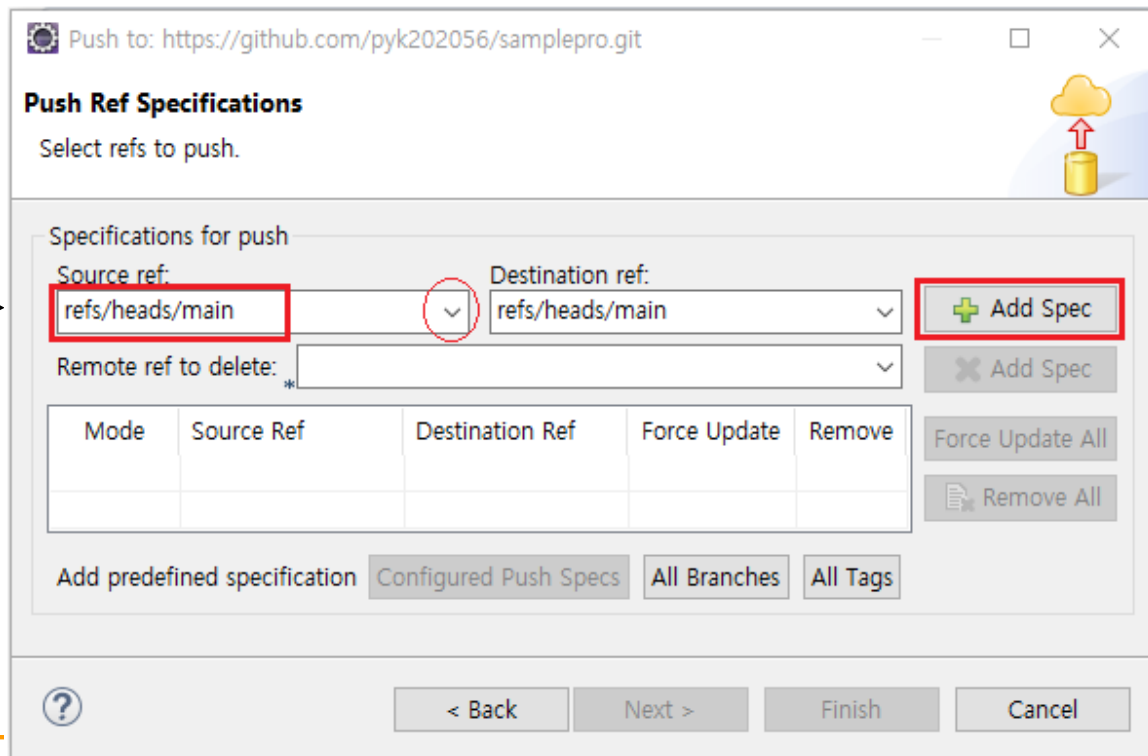
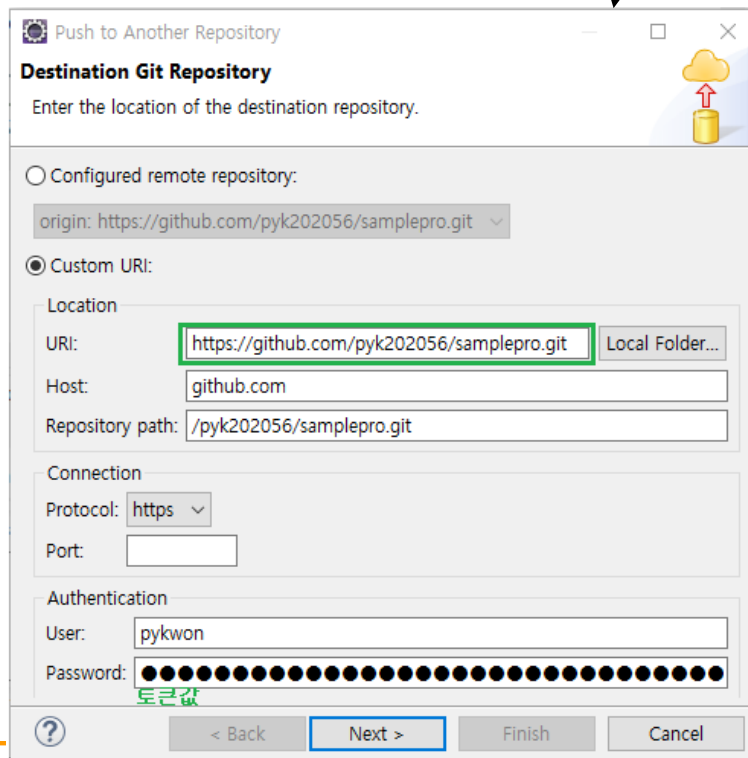
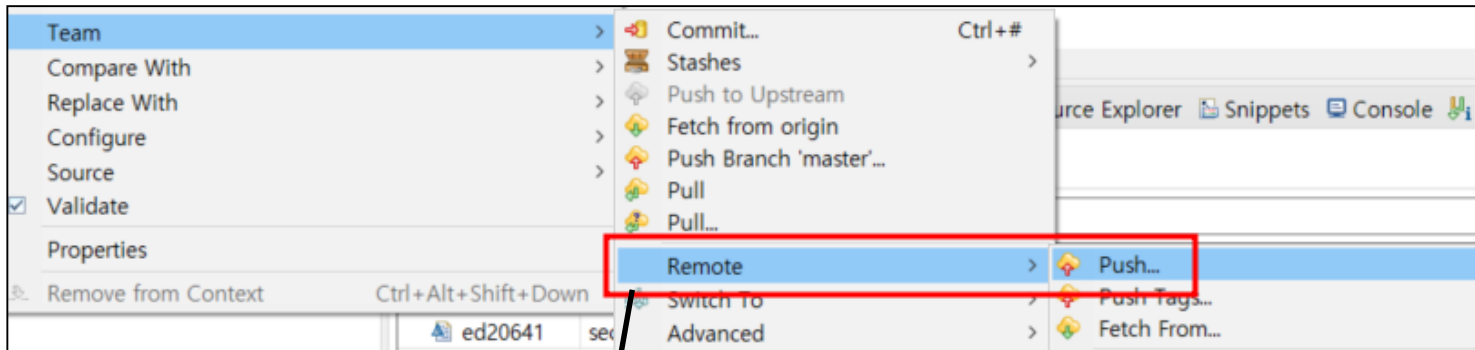
At the bottom of the dialog, there are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted. A red text label '토큰값 입력' (Enter token value) is positioned below the password field.

2. 원격 저장소에 저장하기



2. 원격 저장소에 저장하기

5) 원격 저장소에 저장하기 (push)



2. 원격 저장소에 저장하기

Push to: <https://github.com/pyk202056/samplepro.git>

Push Ref Specifications

Select refs to push.

Specifications for push

Source ref: Destination ref:

Remote ref to delete:

Mode	Source Ref	Destination Ref	Force Update	Remove
Update	refs/heads/main	refs/heads/main	<input checked="" type="checkbox"/>	

Push to: <https://github.com/pyk202056/samplepro.git>

Push Confirmation

Confirm following expected push result.

main → main [5b562c5..195ce19] (2)

- 5b562c5: Merge branch 'main' of <https://github.com/pyk202056/samplepro.git> into main (pyk202056)
- 9b00d9a: commit (pykwon on 2023-06-30 09:52:06)

Message Details

Repository <https://github.com/pyk202056/samplepro.git>

☐ Cancel push if result would be different than above because of changes on remote

☐ Show dialog with result only when it is different from the confirmed result above

Push Results: <https://github.com/pyk202056/samplepro.git>

Pushed to <https://github.com/pyk202056/samplepro.git>

main → main [5b562c5..195ce19] (2)

- 5b562c5: Merge branch 'main' of <https://github.com/pyk202056/samplepro.git> into main (pyk202056)
- 9b00d9a: commit (pykwon on 2023-06-30 09:52:06)

Message Details

Repository <https://github.com/pyk202056/samplepro.git>

Resolving deltas: 0% (0/1)

2. 원격 저장소에 저장하기

원격 저장소 최종 내용

The screenshot shows the GitHub interface for a repository named 'samplepro'. The 'main' branch is selected and circled in red. The file list includes 'abc-xx', 'src/pack', '.gitignore', and 'README.md'. Below the file list, the 'GitTest2 [main]' window shows the 'HiTest.java' file in the 'src/pack' directory, which is also circled in red. The file content is displayed in the editor, showing a Java class with a 'main' method that prints 'Hi' and 'Hi2'.

이후부터는 원격 저장소에 저장하기 전에 **pull** 먼저 하고 나중에 **push** 한다.

The screenshot shows the Git GUI window with the 'Commit and Push' button circled in red. The commit message is 'HiTest2 내용 추가' and the author is 'pykwon <abc@abc.com>'.

The screenshot shows the GitHub repository page for 'samplepro' with the 'main' branch selected. The file 'src/pack/HiTest.java' is selected, and the 'Code' tab is active, displaying the file content. The content is a Java class with a 'main' method that prints 'Hi' and 'Hi2'.

git clone VS git pull

git clone : remote 설정을 자동으로 해주는 초기 다운로드 때 사용

git pull : remote 설정이 이미 되어 있을 때 업데이트 사항 등을 다운로드 할 때 사용

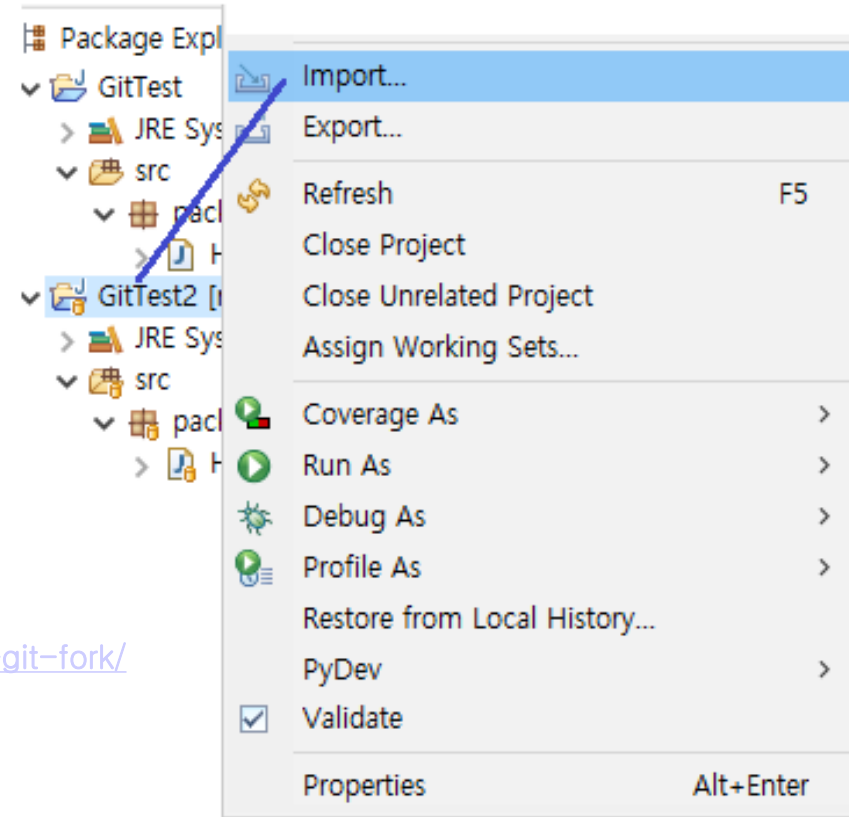
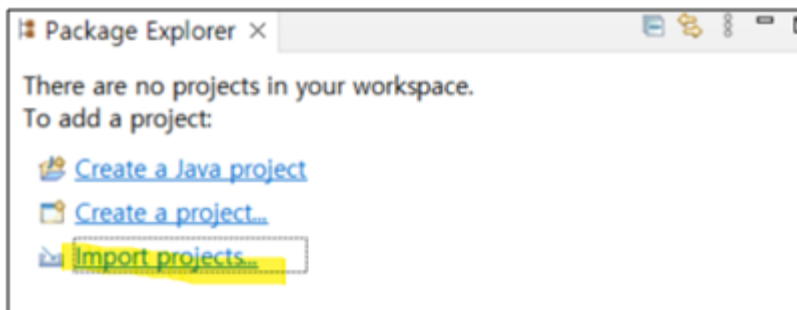
3. Eclipse로 clone 하기

Github 저장소 복제

이미 초기화된 원격저장소를 복제해와서 개발하는 경우, 복제 작업을 Git에서는 Clone이라고 한다. 협업을 하는 경우나 인터넷의 소스 코드를 공개 하는 경우 GitHub에 저장소를 만들고 이 저장소를 클론해서 작업하는 방식을 주로 사용한다.

GitHub.com에 리포지토리를 만들면 원격 리포지토리가 존재한다. 리포지토리를 복제하여 컴퓨터에 로컬 복사본을 만들고 두 위치 간에 동기화할 수 있다.

현재 eclipse 프로젝트 구조 화면



Git Clone과 Git Fork의 차이점

<https://dejavuhyo.github.io/posts/difference-between-git-clone-and-git-fork/>

깃허브(github) 프로젝트 import 하기

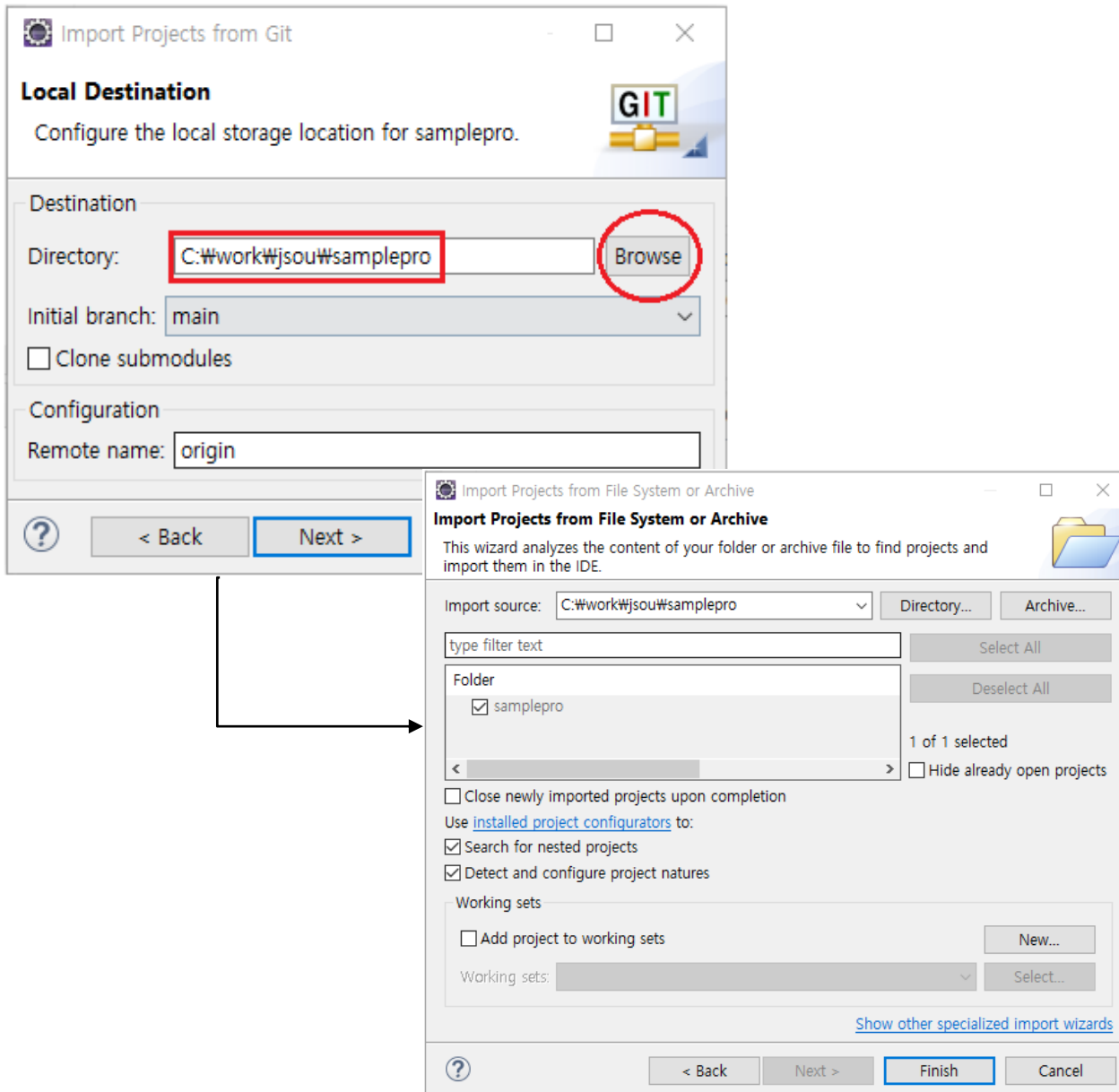
3. Eclipse로 clone 하기

1) 원격 저장소를 로컬 저장소로 clone 하기

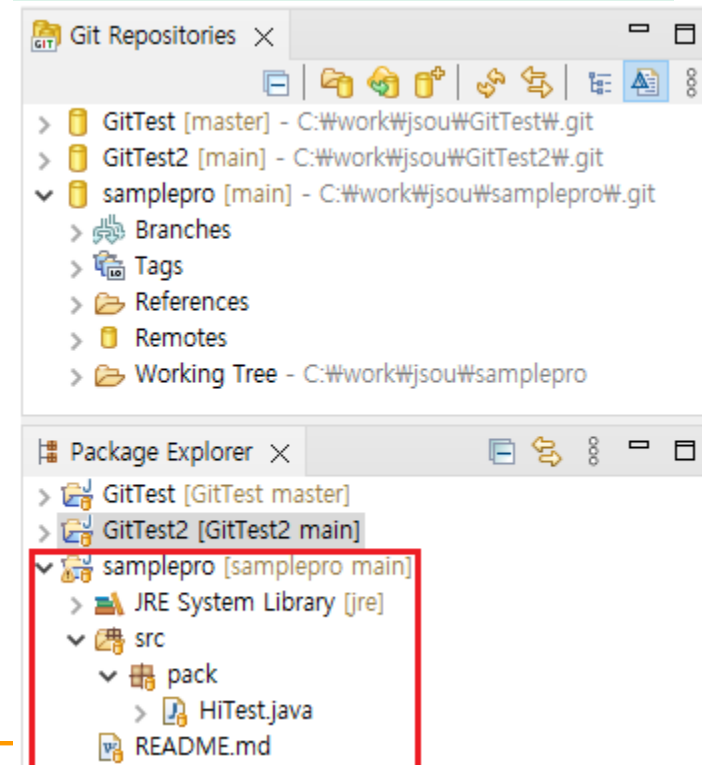
The process of cloning a Git repository in Eclipse involves the following steps:

- Import Dialog:** Select **Projects from Git (with smart import)** from the list of import wizards.
- Select Repository Source:** Choose **Clone URI** as the repository source.
- Source Git Repository:** Enter the repository details:
 - URI: `https://github.com/pyk202056/samplepro.git`
 - Host: `github.com`
 - Repository path: `/pyk202056/samplepro.git`
 - Connection: Protocol `https`
 - Authentication: User `pykwon`, Password (masked), and ☒ **Store in Secure Store**
- Branch Selection:** Select the **main** branch to clone from the remote repository.

3. Eclipse로 clone 하기



clone 후 eclipse 프로젝트 구조 화면



협업 처리

Github를 이용한 팀 프로젝트 협업 1

Github를 이용한 팀 프로젝트 협업 2

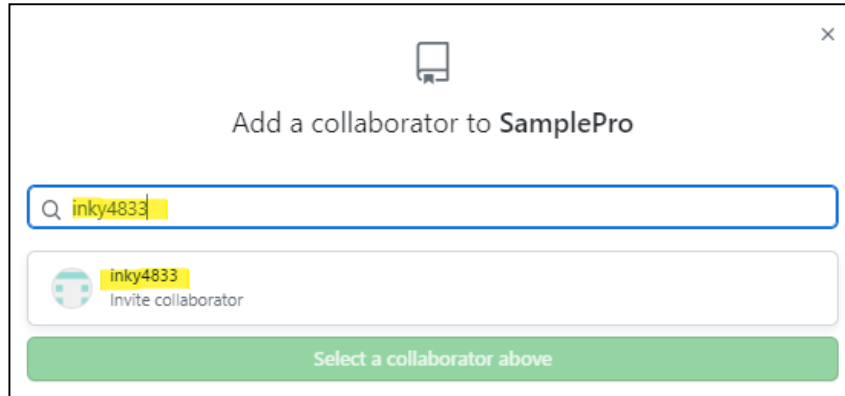
1. Collaborators 추가

1) 조장(pyk202056)이 로그인 후 특정 repository 에서 Collaborators 링크를 선택한다

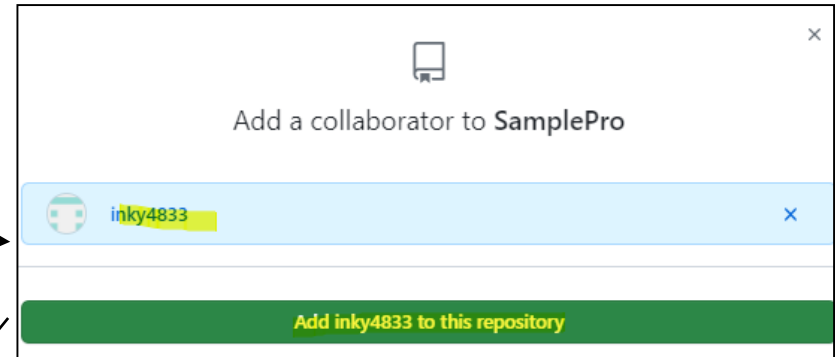
The screenshot shows the GitHub interface for a repository named 'SamplePro' by user 'pyk202056'. The 'Settings' tab is active in the top navigation bar. In the left sidebar, the 'Collaborators' link is highlighted. The main content area displays 'Who has access' with two sections: 'PUBLIC REPOSITORY' (This repository is public and visible to anyone. Manage) and 'DIRECT ACCESS' (0 collaborators have access to this repository. Only you can contribute to this repository.). Below this is the 'Manage access' section, which shows a message 'You haven't invited any collaborators yet' and an 'Add people' button. A modal dialog is open, titled 'Add a collaborator to SamplePro', with a search bar and a green 'Add people' button.

1. Collaborators 추가

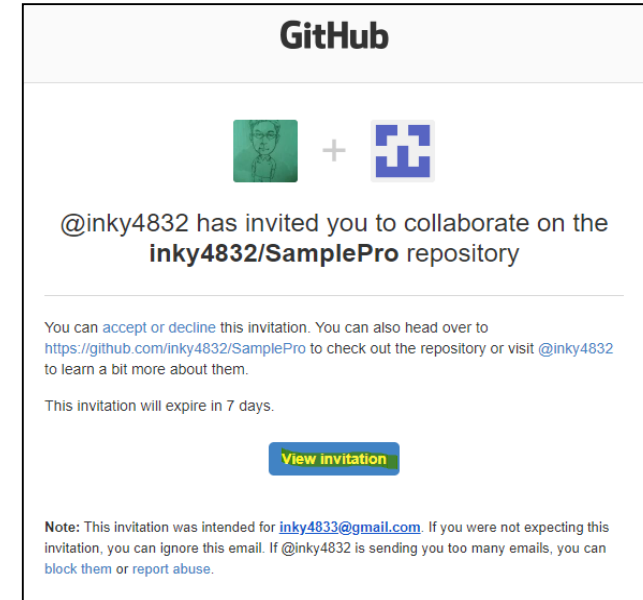
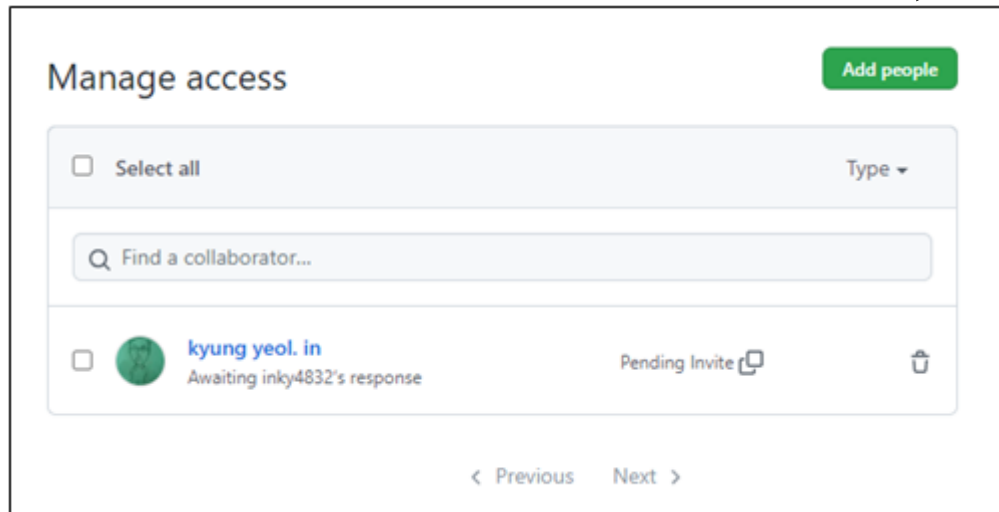
2) 조원(inky4833@gmail.com)으로 등록



이메일로 검색할 것

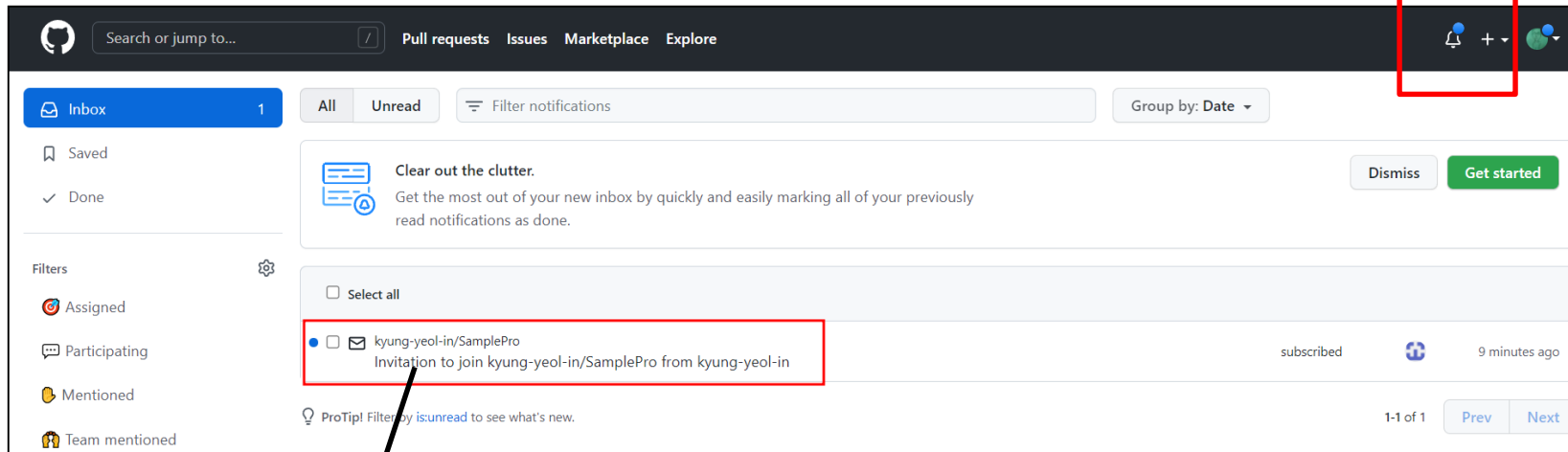


이메일 전송됨

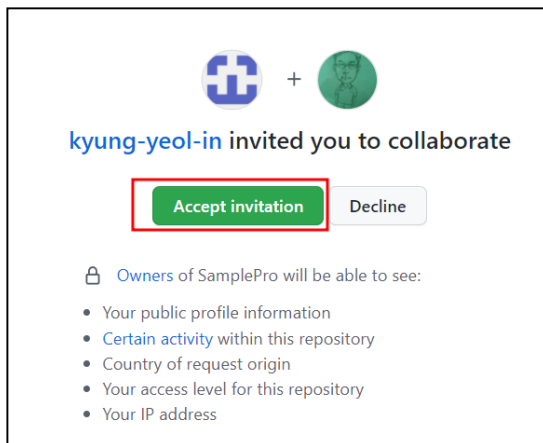


1. Collaborators 추가

3) inky4833 로그인 후 알림 선택

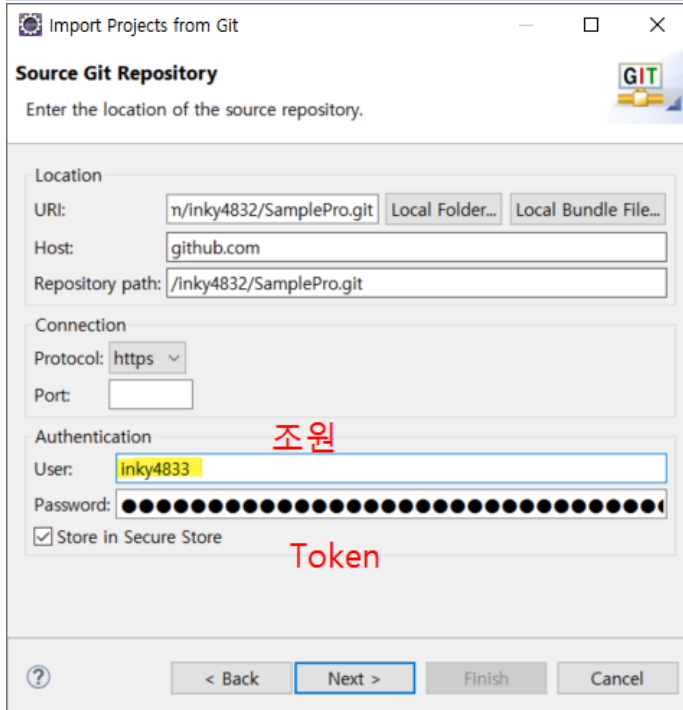


4) 조원으로 inky4833 등록됨



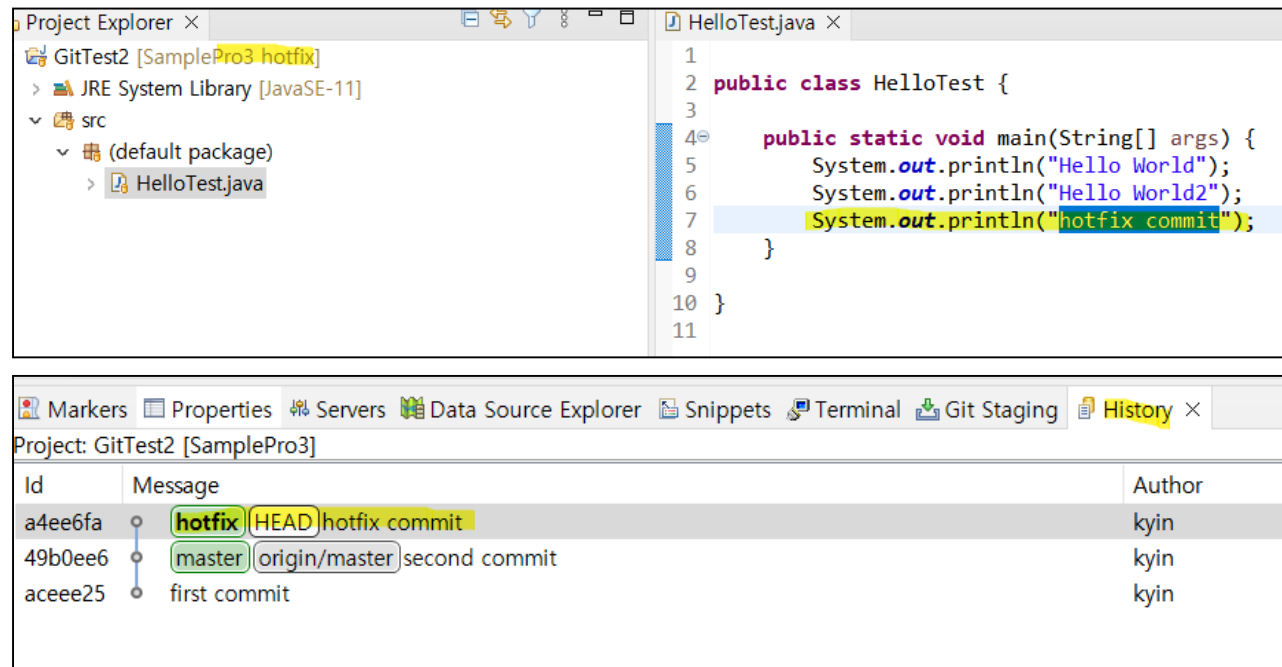
2. git clone

5) 조원들은 조장이 GitHub에 Push한 리소스를 조원 Eclipse로 clone한다.



The dialog box 'Import Projects from Git' is shown. It has a 'Source Git Repository' section with a 'Location' group containing 'URI', 'Host', and 'Repository path' fields. The 'URI' field contains 'n/inky4832/SamplePro.git', 'Host' contains 'github.com', and 'Repository path' contains '/inky4832/SamplePro.git'. There are buttons for 'Local Folder...' and 'Local Bundle File...'. Below is a 'Connection' section with 'Protocol' set to 'https' and an empty 'Port' field. The 'Authentication' section has 'User' set to 'inky4833', a masked 'Password' field, and a checked 'Store in Secure Store' option. Red text labels '조원' (Member) and 'Token' are placed over the 'User' and 'Password' fields respectively. At the bottom are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

6) hotfix 브랜치 생성 및 추가 작업 후 commit



The Eclipse IDE interface is shown. The 'Project Explorer' on the left shows a project 'GitTest2 [SamplePro3-hotfix]' with a 'src' folder containing 'HelloTest.java'. The 'HelloTest.java' editor on the right shows the following code:

```
1 public class HelloTest {
2
3
4     public static void main(String[] args) {
5         System.out.println("Hello World");
6         System.out.println("Hello World2");
7         System.out.println("hotfix commit");
8     }
9
10 }
11
```

The 'Git Staging' view at the bottom shows the commit history for 'Project: GitTest2 [SamplePro3]':

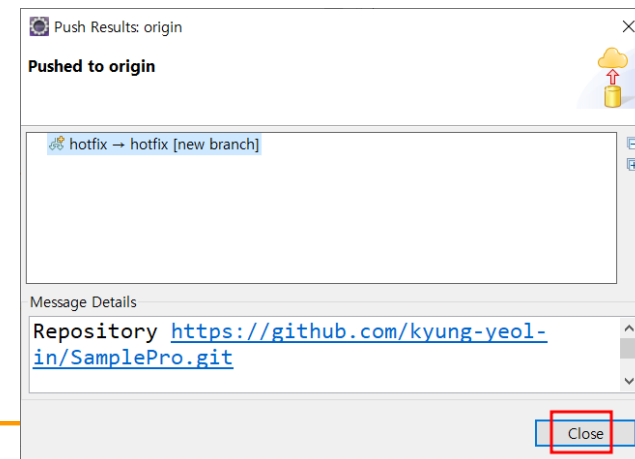
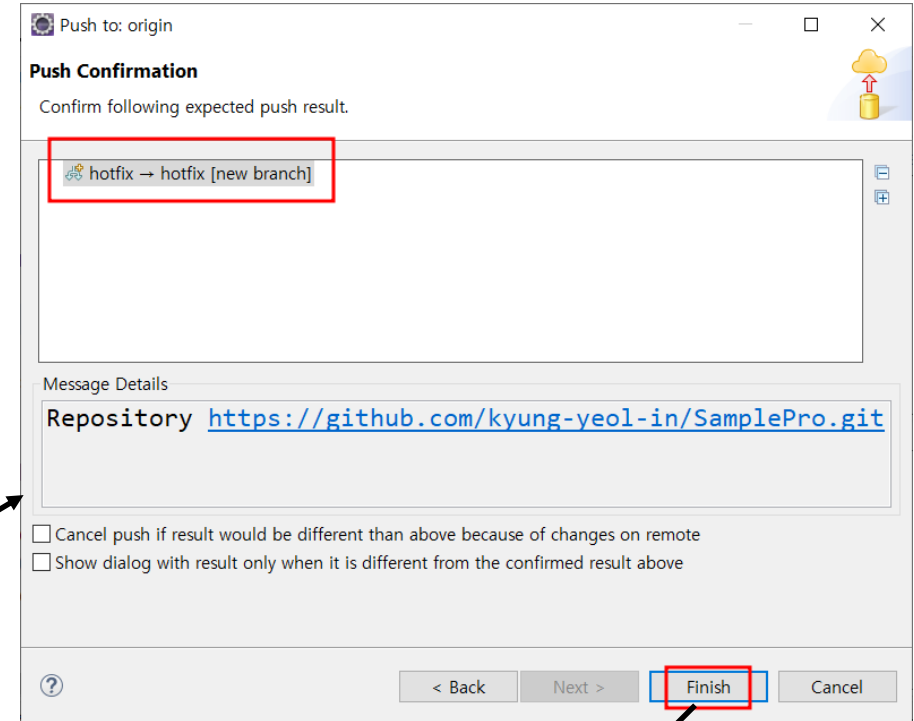
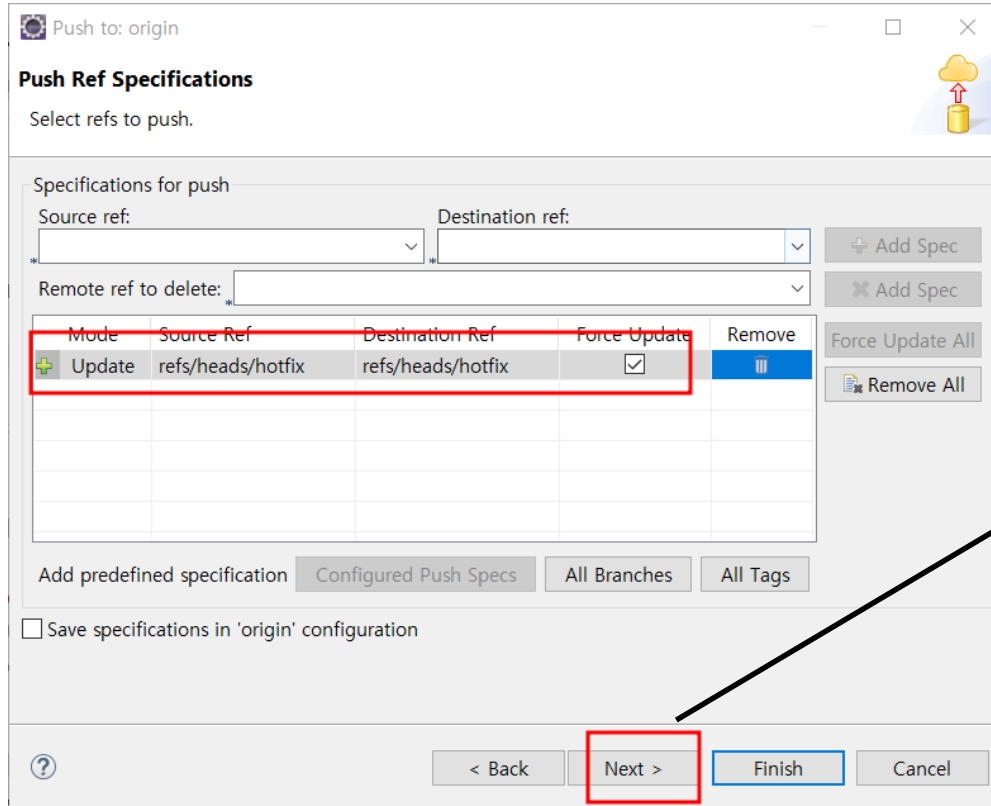
Id	Message	Author
a4ee6fa	hotfix HEAD hotfix commit	kyin
49b0ee6	master origin/master second commit	kyin
aceee25	first commit	kyin

4. hotfix 브랜치 push

7) hotfix 브랜치를 원격 저장소에 push

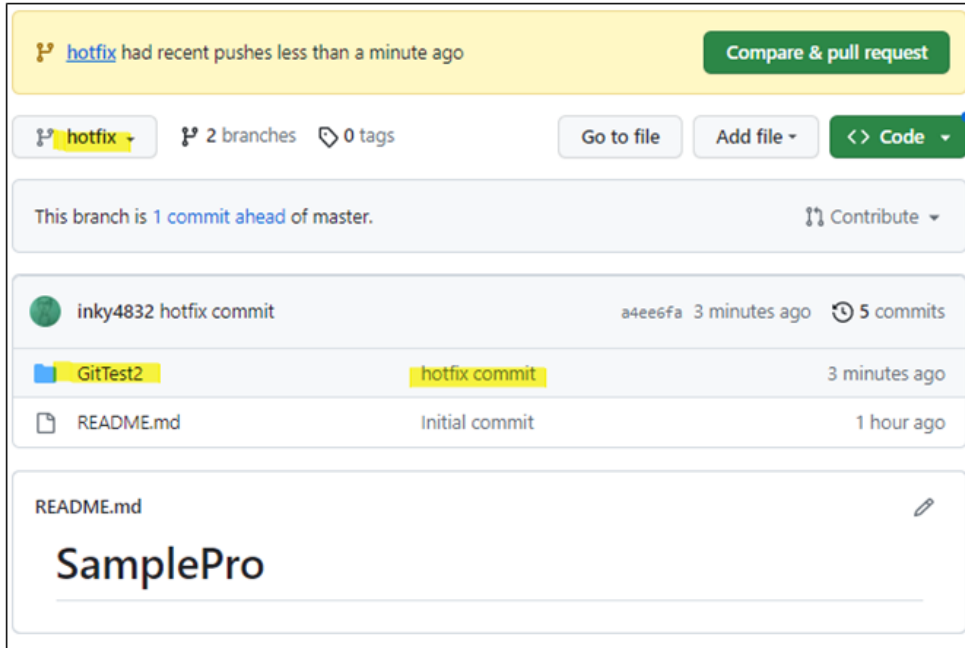
The screenshot illustrates the process of pushing a hotfix branch to a remote repository. On the left, the IntelliJ IDEA menu is open, with the 'Remote' option highlighted. The 'Push...' option is also highlighted. On the right, the 'Push to Another Repository' dialog is shown. The 'Destination Git Repository' section is active, and the 'Custom URI' option is selected. The 'URI' field contains the repository address: `https://github.com/inky4832/SamplePro.git`. The 'Host' is set to `github.com`, and the 'Repository path' is `/inky4832/SamplePro.git`. The 'Connection' section shows the 'Protocol' as `https`. The 'Authentication' section shows the 'User' as `inky4833` and the 'Password' field is filled with dots. The 'Store in Secure Store' checkbox is checked. The 'Next >' button is highlighted.

4. hotfix 브랜치 push

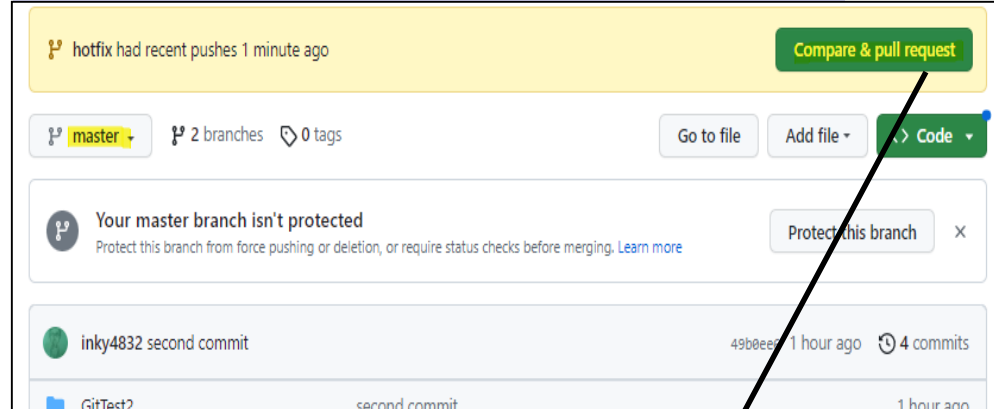


4. hotfix 브랜치 push

hotfix push 후 원격 저장소 구조 화면

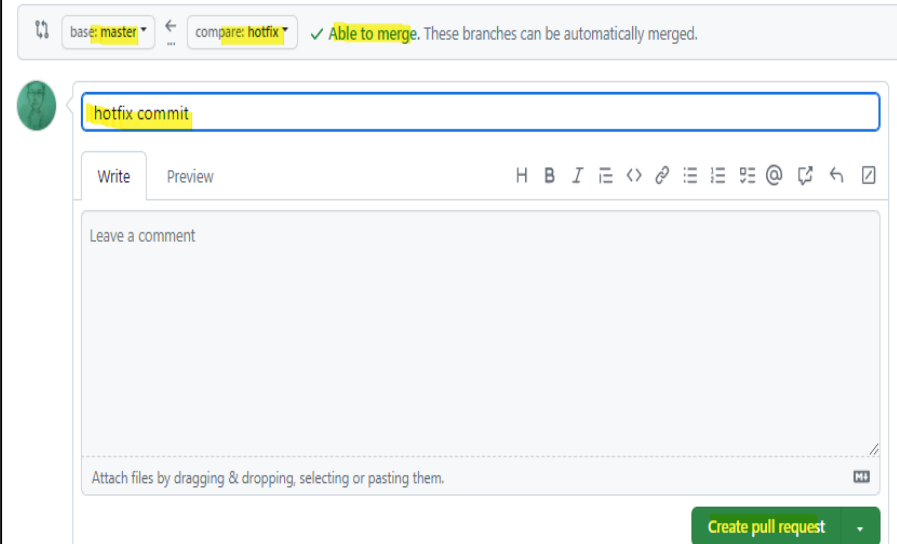


8) hotfix 브랜치를 master 브랜치에 병합



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



5. master 브랜치에 hotfix 병합

hotfix commit #1

[Open](#) inky4832 wants to merge 1 commit into `master` from `hotfix`

Conversation 0 Commits 1 Checks 0 Files changed 1

inky4832 commented now

No description provided.

hotfix commit a4ee6fa

Add more commits by pushing to the `hotfix` branch on inky4832/SamplePro.

- Require approval from specific reviewers before merging
Branch protection rules ensure specific people approve pull requests before they're merged. [Add rule](#)
- Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.
- ✓ This branch has no conflicts with the base branch
Merging can be performed automatically.

[Merge pull request](#) You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Add more commits by pushing to the `hotfix` branch on inky4832/SamplePro.

Merge pull request #1 from inky4832/hotfix

hotfix commit

This commit will be authored by inky4832@daum.net

[Confirm merge](#) [Cancel](#)

inky4832 merged commit e363e42 into `master` now

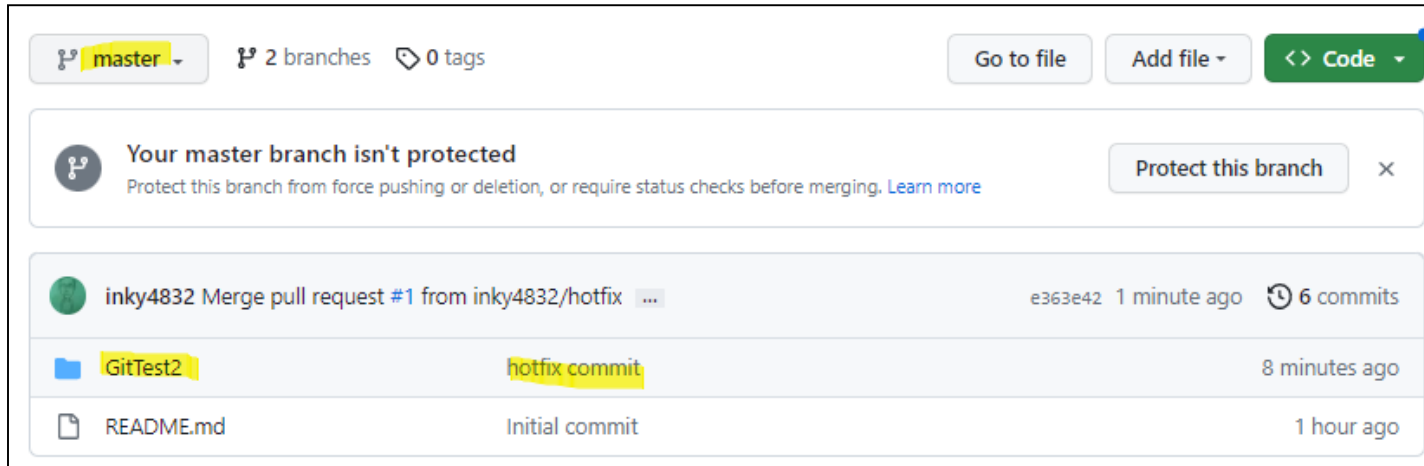
Pull request successfully merged and closed
You're all set—the `hotfix` branch can be safely deleted. [Delete branch](#)

Write Preview

H B I

5. master 브랜치에 hotfix 병합

병합 후 원격 저장소 master 브랜치 화면



이후부터는 다른 조원들이 병합된 master 브랜치를 pull 한다.

11 - E
