

Github

inky4832@daum.net

원격 저장소 생성 및 관리 (GitHub)

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.

개발시 협업이 가능하다.

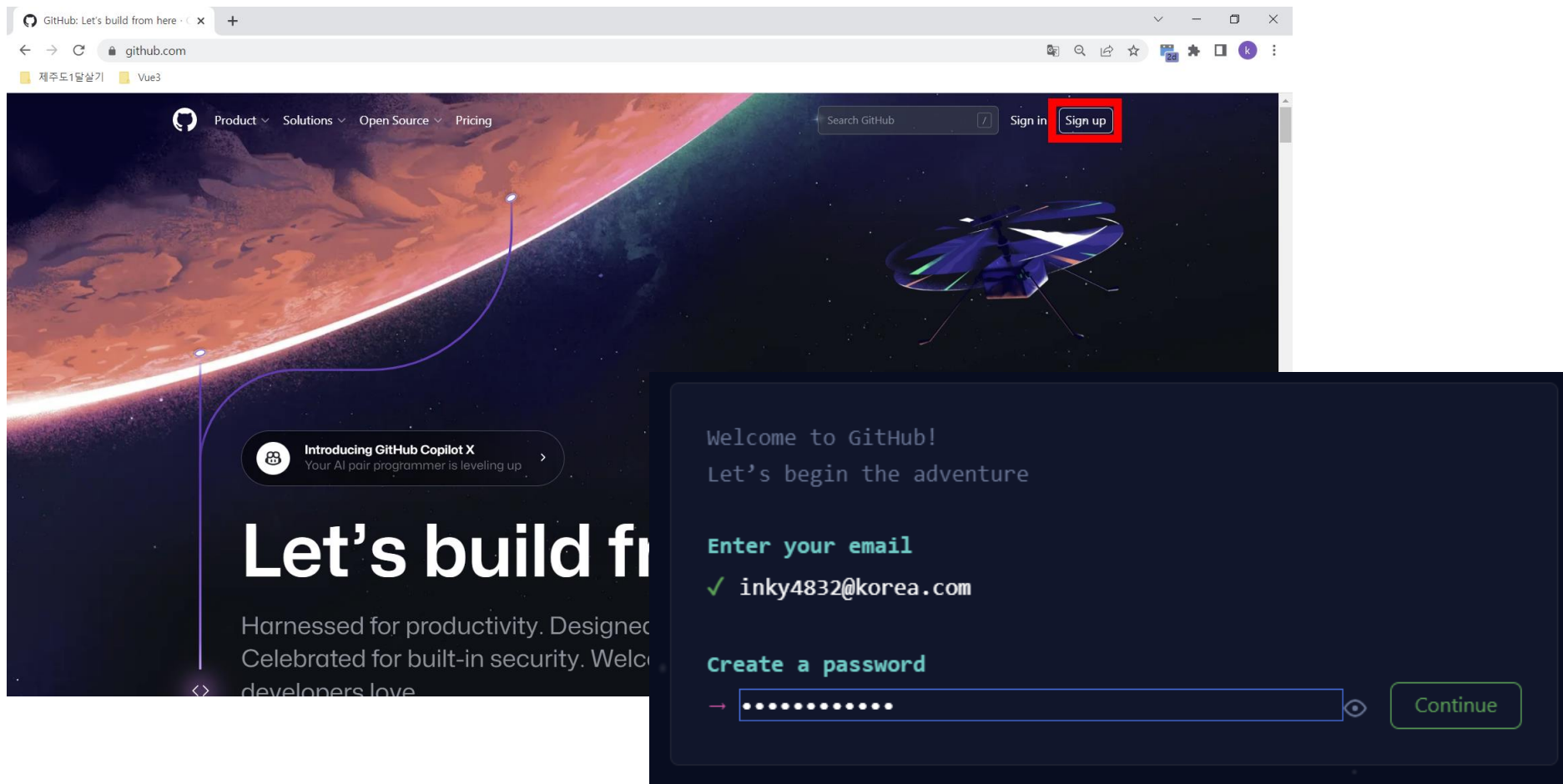
원격 저장소 관련 기본 명령어

명령어	기능
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/>



3. 원격 저장소 기본 관리

1) 빈 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 *

Repository name *

inky4832

/ study

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

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:

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



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

Set up in Desktop

 or

HTTPS

SSH

<https://github.com/inky4832/study.git>

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

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

```
echo "# study" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/inky4832/study.git
git push -u origin main
```

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

```
git remote add origin https://github.com/inky4832/study.git
git branch -M main
git push -u origin main
```

3. 원격 저장소 기본 관리

2) 로컬 저장소와 빈 원격 저장소 연결

```
git remote add 원격저장소별칭 https://~
```

로컬 저장소 생성

```
$ pwd  
$ mkdir github_tutorial  
$ cd github_tutorial/  
$ git init
```

로컬 저장소와 원격 저장소 연결

```
$ git remote add origin https://github.com/inky4832/study.git
```

연결 상태 확인

```
$ git remote -v  
origin https://github.com/inky4832/study.git (fetch)  
origin https://github.com/inky4832/study.git (push)
```

연결 종료

```
$ git remote rm origin
```

3. 원격 저장소 기본 관리

3) 로컬 작업 내역을 원격 저장소에 올리기

git push 원격브랜치별칭 로컬브랜치명

git push 원격브랜치별칭 --all

로컬 저장소 변경

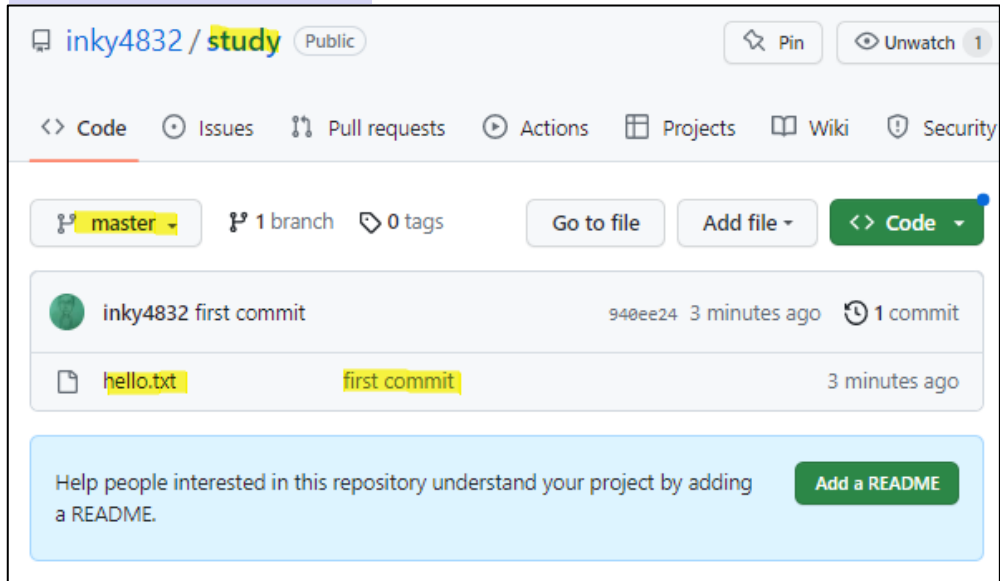
```
$ vim hello.txt
$ git add hello.txt
$ git commit -m 'first commit'
$ cat hello.txt
hello
```

원격 저장소에 올리기

```
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 211 bytes | 211.00 kiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/inky4832/study.git
 * [new branch]      master -> master
```


3. 원격 저장소 기본 관리

github에서 확인



4. 원격 저장소에서 로컬 저장소로 가져오기

1) 원격 저장소 커밋 내역을 로컬 저장소로 가져오기

git fetch

원격 저장소 변경



이 경우에 git push 하면 [rejected] 에러가 발생된다. 이유는 로컬저장소와 원격 저장소간에 커밋 버전이 일치하지 않기 때문이다.

```
$ git push origin master
To https://github.com/inky4832/study.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/inky4832/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

git fetch

git fetch 명령어는 원격 저장소의 커밋 내역만 가져온다.

```
$ git fetch
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), 642 bytes | 91.00 KiB/s, done.
From https://github.com/inky4832/study
 940ee24..55d3685  master    -> origin/master
```

로컬 저장소의 모든 브랜치 정보 보기

```
inky4@DESKTOP-410SFPS MINGW64 ~/github_tutorial (master)
$ git branch -a
* master
remotes/origin/master
```

로컬 저장소와 원격 저장소의 변경 내역 비교

```
inky4@DESKTOP-410SFPS MINGW64 ~/github_tutorial (master)
$ git log master..origin/master
commit 55d3685e7f0ba15382cc641eac1b280fa47a928f (origin/master)
Author: kyung yeol. in <inky4832@daum.net>
Date: Mon Apr 3 21:44:05 2023 +0900

    world commit
```

4. 원격 저장소에서 로컬 저장소로 가져오기

로컬 저장소와 원격 저장소의 병합

```
inky4@DESKTOP-410SFPS MINGW64 ~/github_tutorial (master)
$ git merge origin/master
Updating 940ee24..55d3685
Fast-forward
 world.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 world.txt
```

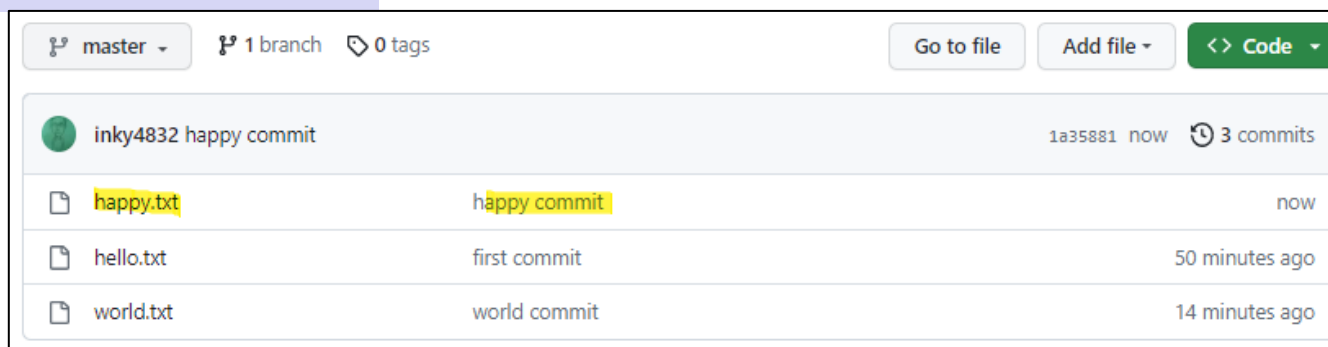
```
inky4@DESKTOP-410SFPS MINGW64 ~/github_tutorial (master)
$ git log --oneline
55d3685 (HEAD -> master, origin/master) world commit
940ee24 first commit
```

4. 원격 저장소에서 로컬 저장소로 가져오기

2) 원격 저장소 커밋 내역을 로컬 저장소로 가져오기

```
git pull origin master
```

원격 저장소 변경



The screenshot shows the GitHub web interface for a repository. At the top, it indicates the current branch is 'master', there is 1 branch, and 0 tags. Below this, a commit history table is displayed. The first commit is by 'inky4832' with the message 'happy commit', hash '1a35881', and is 'now'. Below this, three files are listed: 'happy.txt' (linked to 'happy commit'), 'hello.txt' (linked to 'first commit'), and 'world.txt' (linked to 'world commit').

File	Commit	Time
happy.txt	happy commit	now
hello.txt	first commit	50 minutes ago
world.txt	world commit	14 minutes ago

이 경우에 git push 하면 [rejected] 에러가 발생된다. 이유는 로컬저장소와 원격 저장소간에 커밋 버전이 일치하지 않기 때문이다.

```
$ git push origin master
To https://github.com/inky4832/study.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/inky4832/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

```
git pull origin master
```

git pull 명령어는 원격 저장소의 커밋 내역을 로컬로 가져와서 자동으로 병합한다.

```
inky4@DESKTOP-410SFPS MINGW64 ~/github_tutorial (master)
$ git pull origin master
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), 679 bytes | 97.00 kiB/s, done.
From https://github.com/inky4832/study
 * branch                master      -> FETCH_HEAD
    55d3685..1a35881      master      -> origin/master
Updating 55d3685..1a35881
Fast-forward
 happy.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 happy.txt
```

```
inky4@DESKTOP-410SFPS MINGW64 ~/github_tutorial (master)
$ git log --oneline
1a35881 (HEAD -> master, origin/master) happy commit
55d3685 world commit
940ee24 first commit
```

5. 원격 저장소를 로컬 저장소에 복사

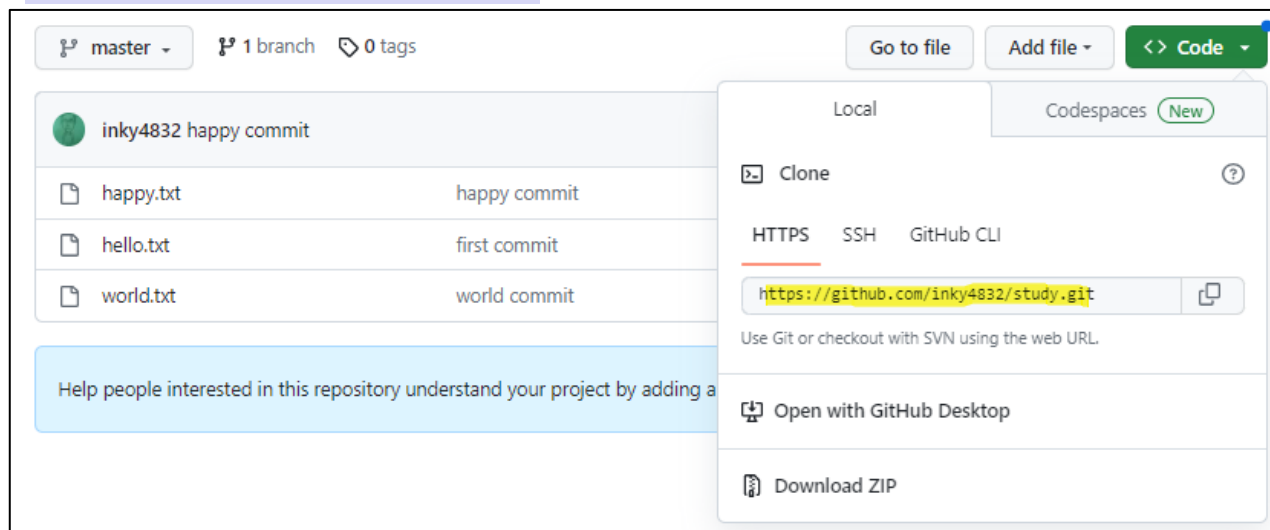
git clone

github의 원격 저장소를 로컬 저장소로 복사하는 작업을 의미한다.

로컬에 디렉터리 생성

```
$ pwd  
/c/Users/inky4  
$ mkdir local_clone  
$ cd local_clone/
```

원격 저장소에 주소 복사



5. 원격 저장소를 로컬 저장소에 복사

git clone

```
inky4@DESKTOP-410SFPS MINGW64 ~/local_clone
$ git clone https://github.com/inky4832/study.git
Cloning into 'study'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), done.
```

```
inky4@DESKTOP-410SFPS MINGW64 ~/local_clone
$ cd study/

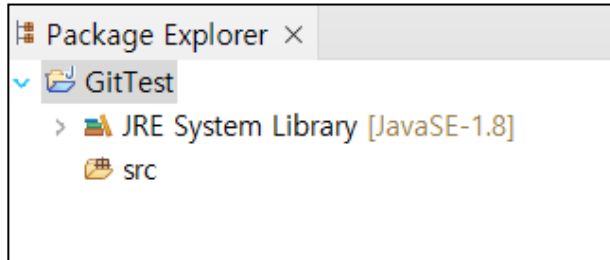
inky4@DESKTOP-410SFPS MINGW64 ~/local_clone/study (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

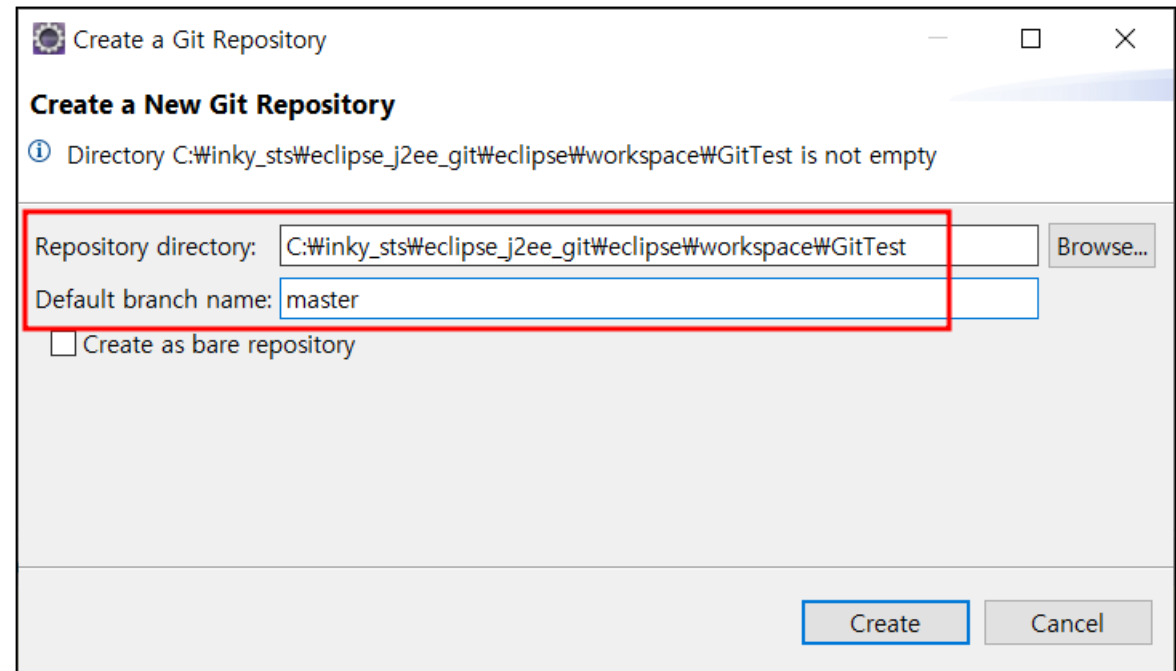
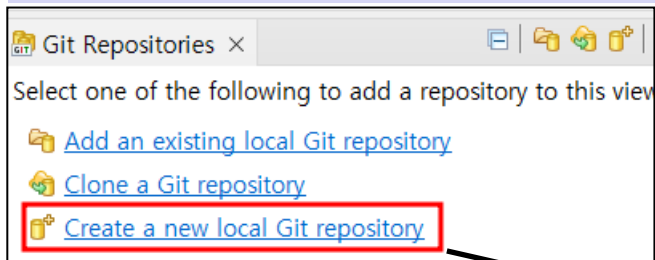

eclipse에서 Git/Github 사용

1. 로컬 저장소 생성 및 초기화

1) 프로젝트 생성

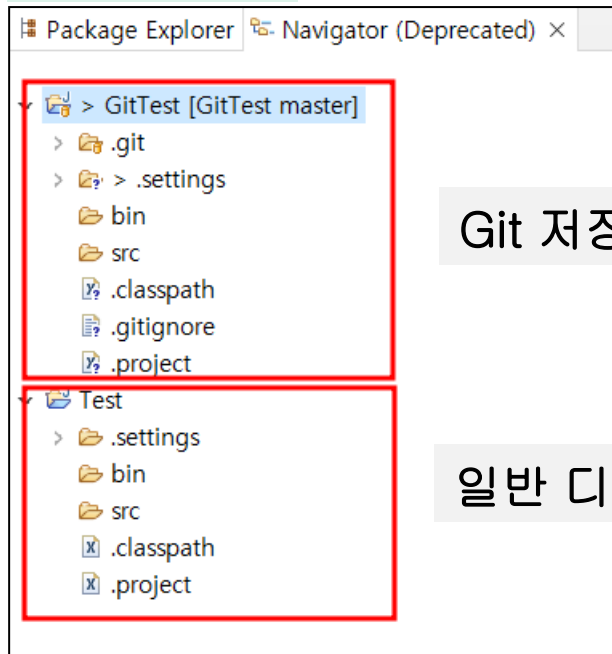


2) 로컬저장소 생성 및 초기화



1. 로컬 저장소 생성 및 초기화

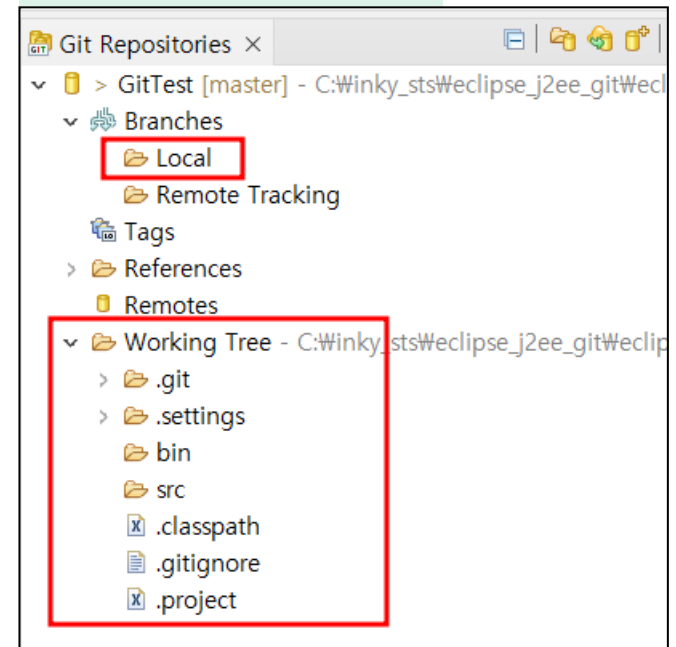
Navigator 탭



Git 저장소 구조

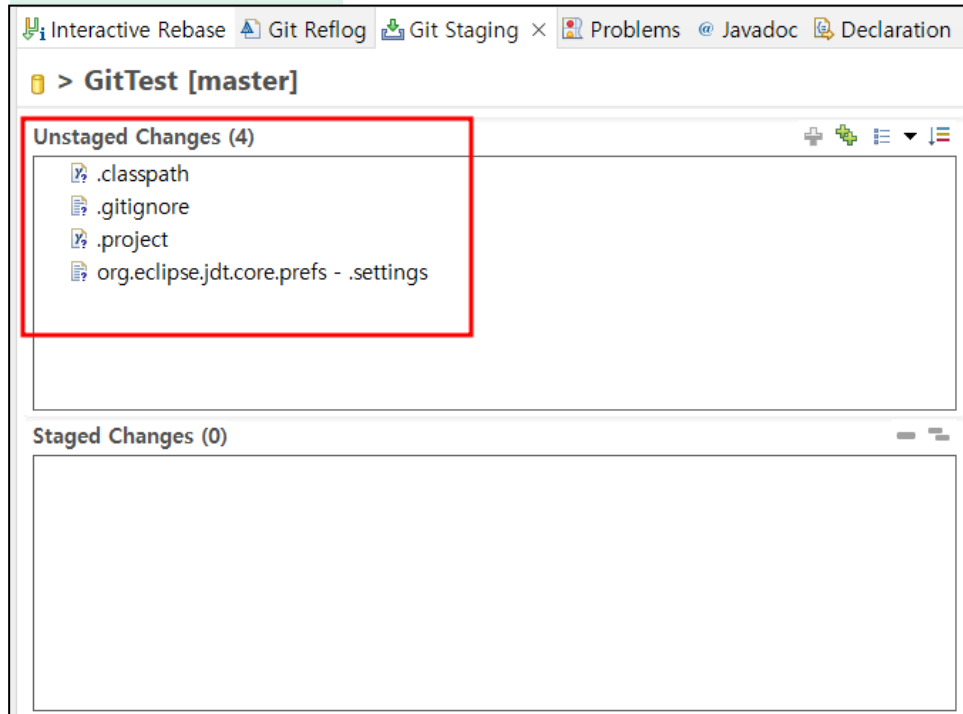
일반 디렉토리 구조

Git Repositories 탭



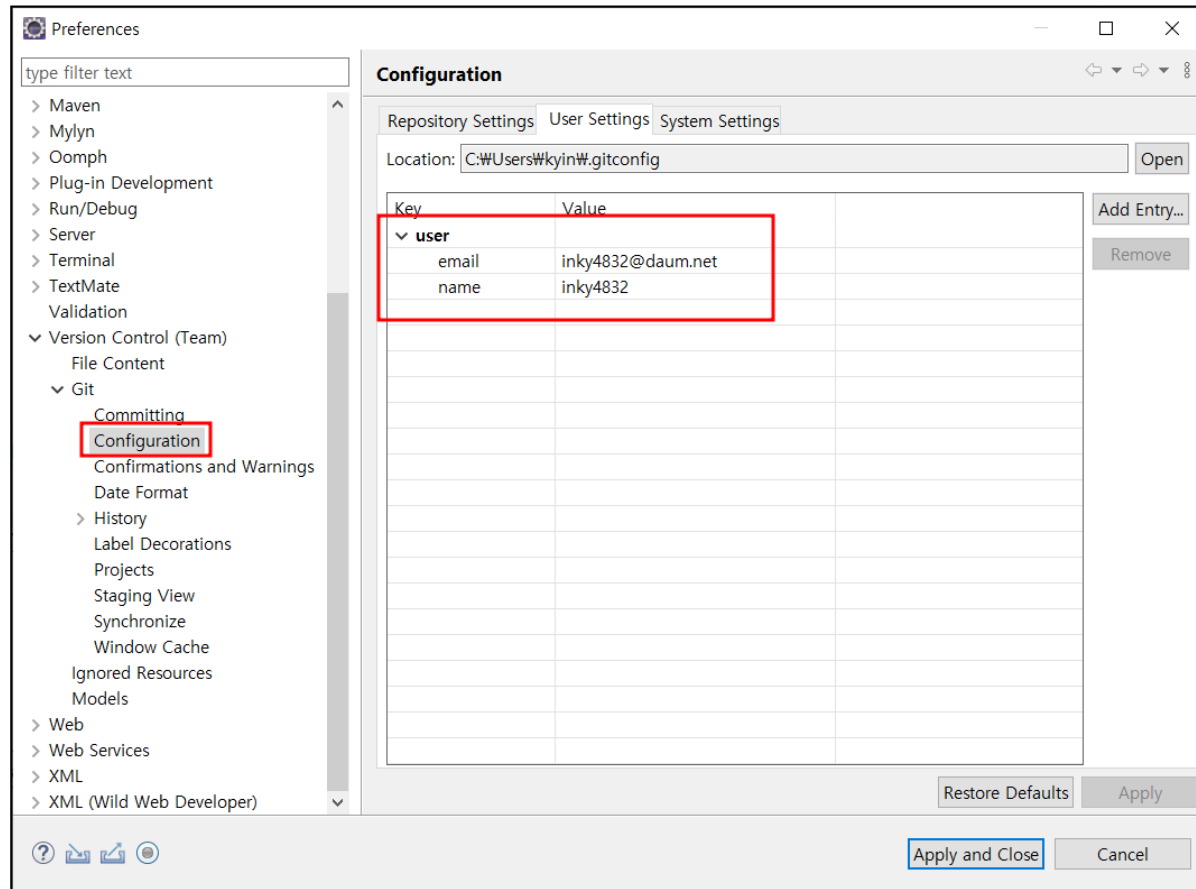
1. 저장소 생성 및 초기화

Git Staging 탭



1) 사용자 정보

버전을 저장할 때마다 그 버전을 만든 사용자 정보도 함께 저장된다.



```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

2. Git 환경 설정

2) 불필요한 파일 무시(.gitignore)

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

<https://www.gitignore.io>



```
*.log
# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
*.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

# virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*
replay_pid*

### Windows ###
# Windows thumbnail cache files
Thumbs.db
Thumbs.db:encryptable
ehthumbs.db
ehthumbs_vista.db

# Dump file
*.stackdump

# Folder config file
[Dd]esktop.ini

# Recycle Bin used on file shares
$RECYCLE.BIN/

# Windows Installer files
*.cab
*.msi
*.msix
*.msm
*.msp

# Windows shortcuts
*.lnk

# End of https://www.toptal.com/developers/gitignore/api/windows,java,eclipse
```

2. Git 환경 설정

HelloTest.java 추가

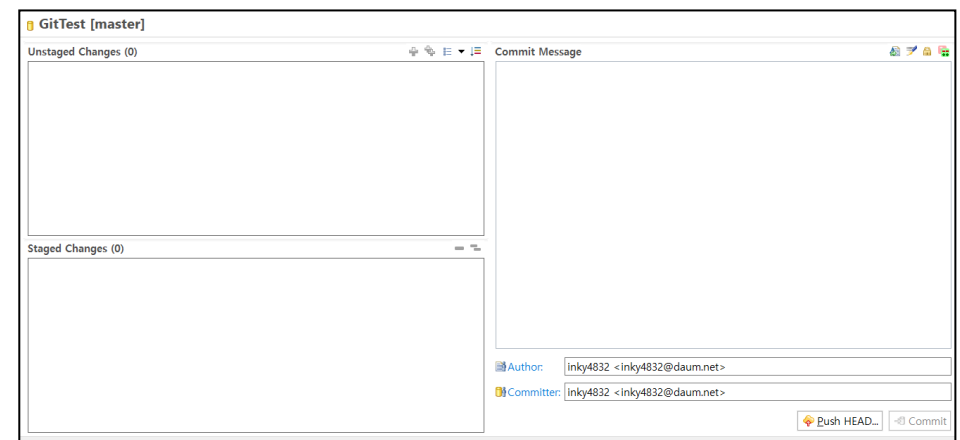
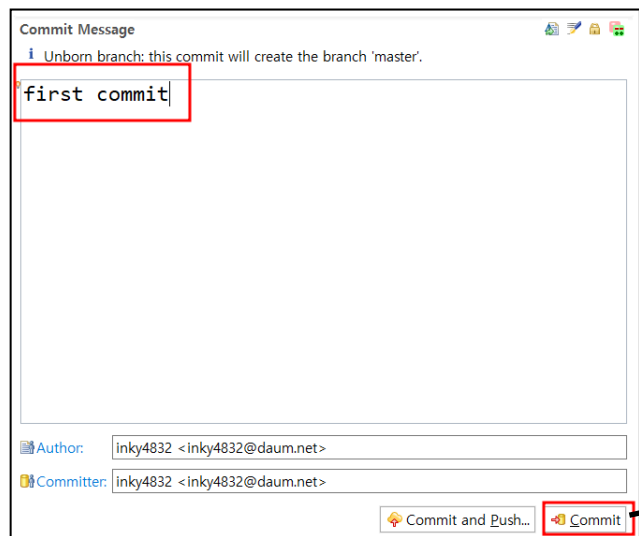
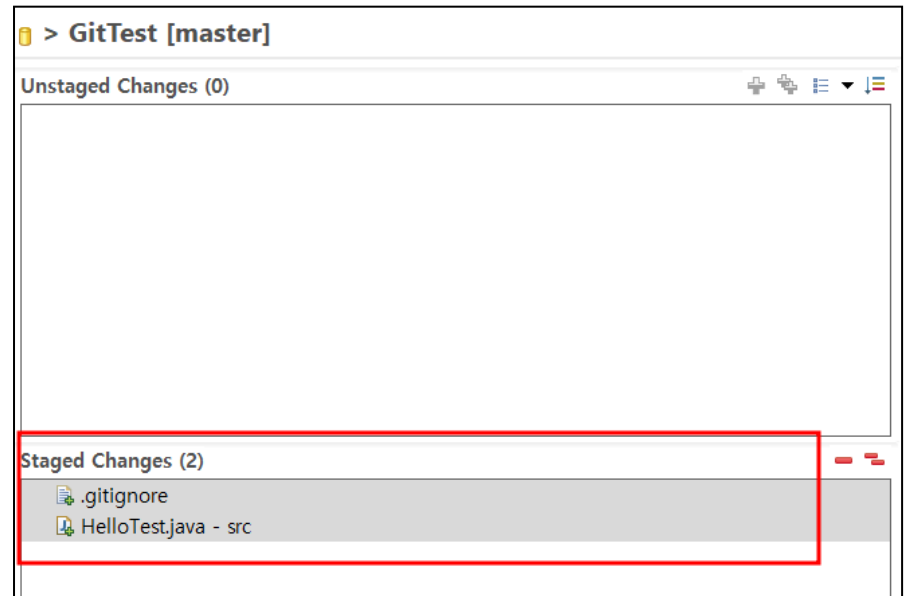
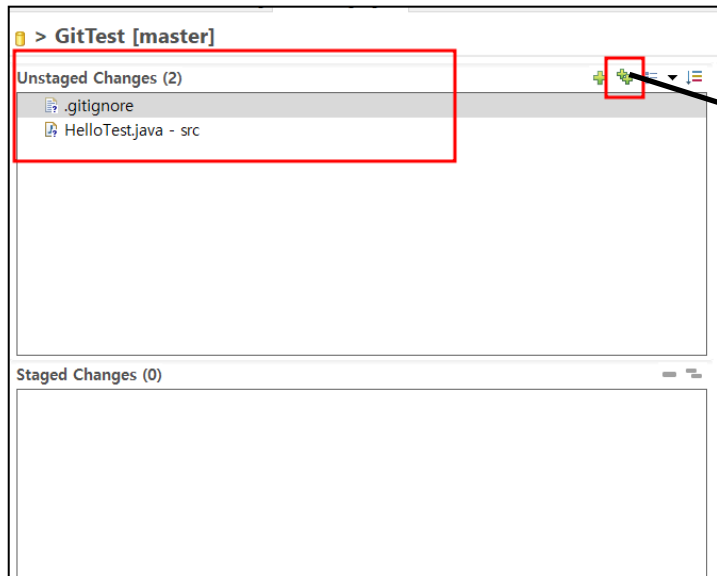
```
HelloTest.java ×  
1  
2 public class HelloTest {  
3     public static void main(String[] args) {  
4         System.out.println("Hello World");  
5     }  
6 }  
7
```

.gitignore 파일에 복사

The screenshot shows an IDE with the `.gitignore` file open. Lines 125 and 126, which contain `.classpath` and `.project`, are highlighted with a red box. Below the editor, the Git GUI shows the current branch as `GitTest [master]`. Under 'Unstaged Changes (2)', the files `.gitignore` and `HelloTest.java - src` are listed, also highlighted with a red box.

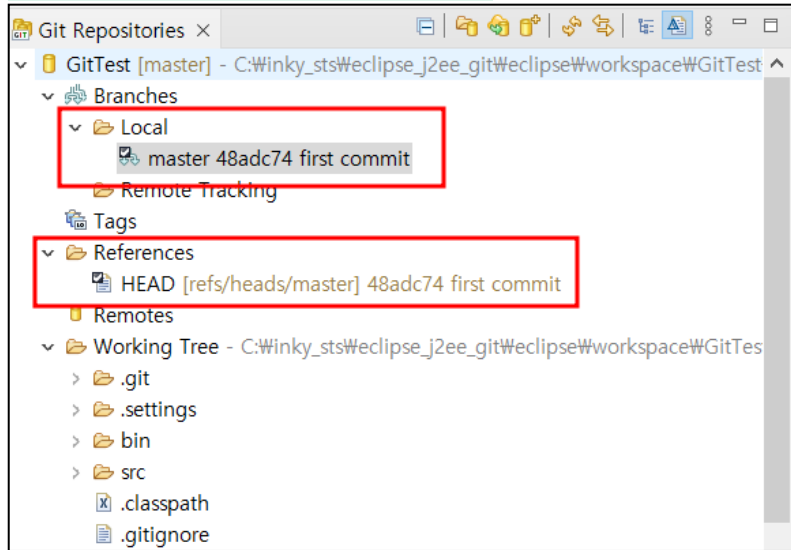
The screenshot shows the Git commit dialog. The 'Author' and 'Committer' fields both contain the text `inky4832 <inky4832@daum.net>`, which is highlighted with a red box. At the bottom, there are two buttons: 'Commit and Push...' and 'Commit'.

3. 저장소에 추적 파일 등록 및 제출

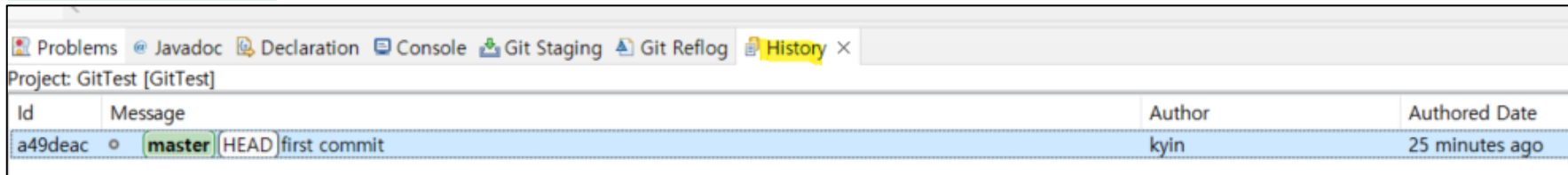


4. 등록된 버전 확인

Git Repositories 탭



History 탭



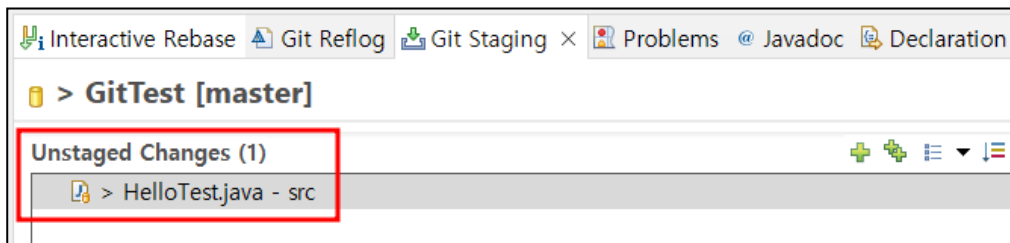
5. 여러 버전 생성

HelloTest.java 변경

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

Git Staging 탭

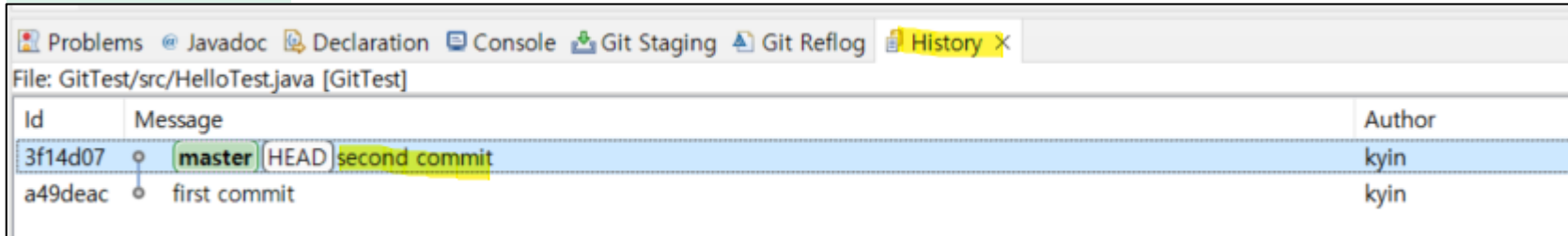
코드가 변경되면 Git Staging 탭에 수정된 파일이 보여진다.



이후에는 계속적으로 commit까지 반복 작업한다.

5. 여러 버전 생성

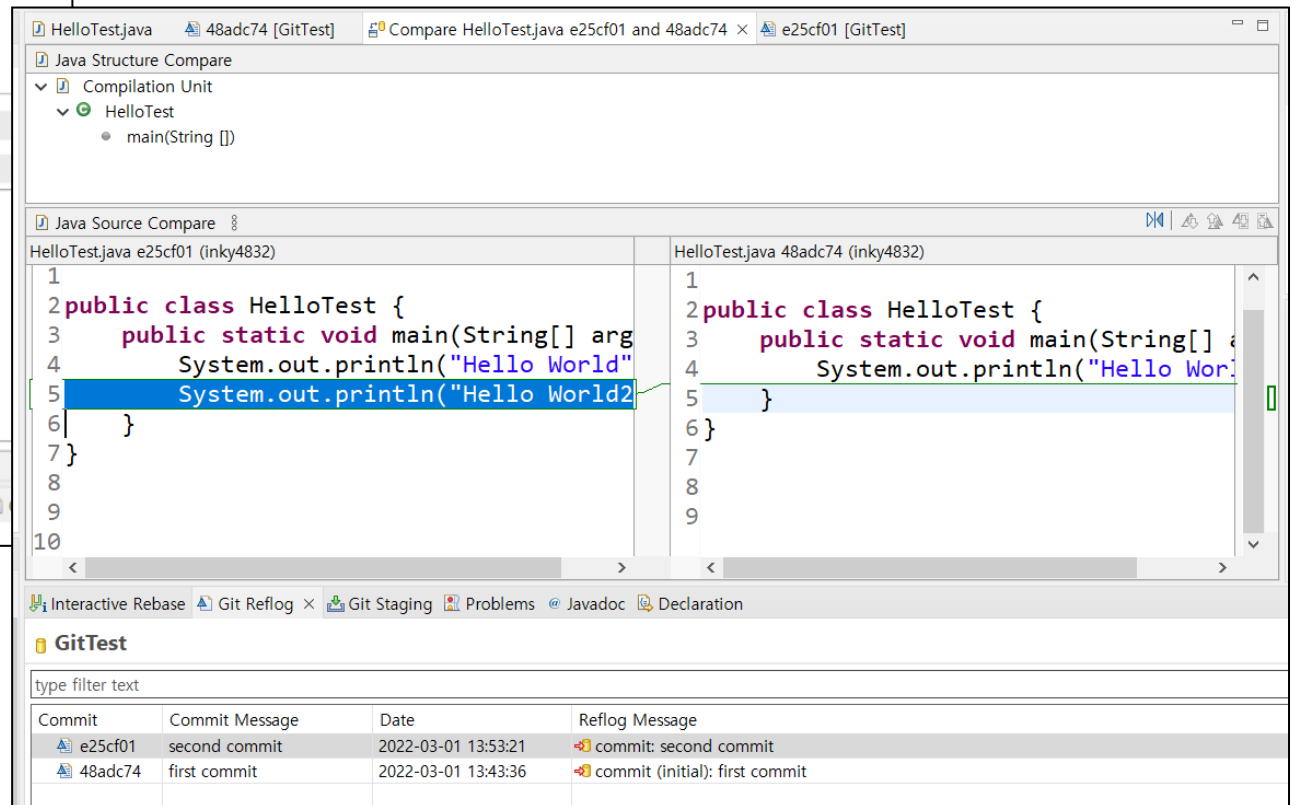
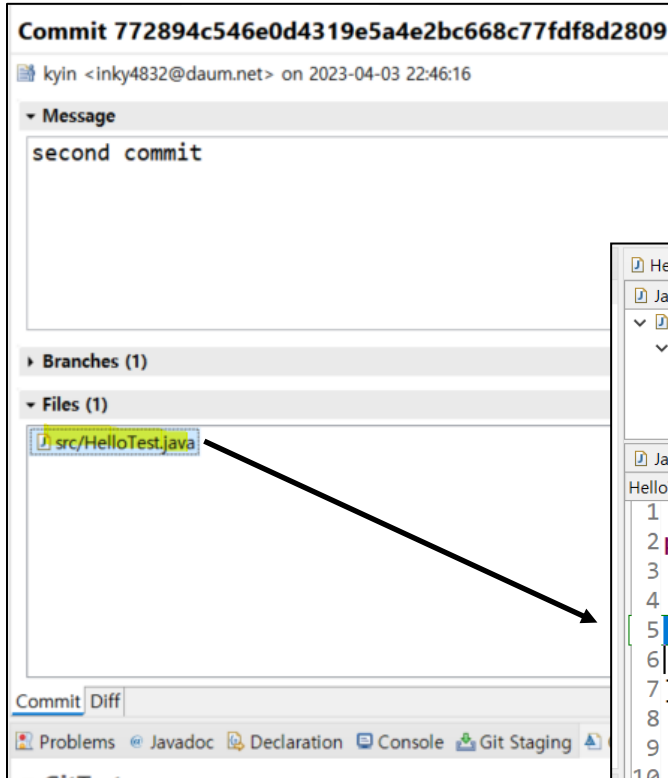
History 탭



Id	Message	Author
3f14d07	second commit	kyin
a49deac	first commit	kyin

5. 여러 버전 생성

HelloTest.java 변경내용 비교



6. 작업 되돌리기

특정 commit 으로 되돌리기

History 탭

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

Id	Message
3f14d07	second commit
a49deac	first commit

Context Menu Options:

- Interactive Rebase
- Reset (selected)
- Revert Commit
- Quick Diff
- Modify
- Import Changed Projects
- Copy Commit Id (Ctrl+C)

Reset Sub-menu Options:

- Soft (HEAD Only)
- Mixed (HEAD and Index)
- Hard (HEAD, Index, and Working Tree) (selected)

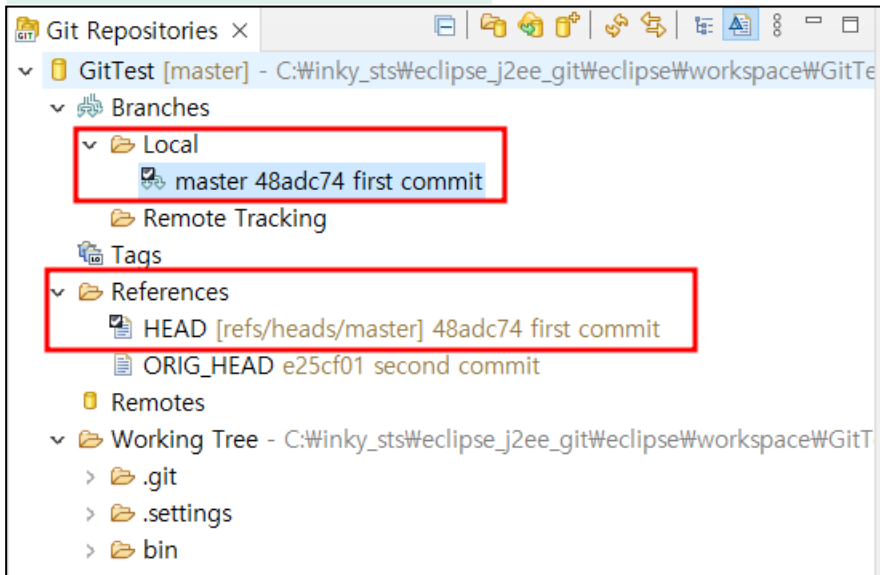


File: GitTest/src/HelloTest.java [GitTest]

Id	Message	Author
a49deac	first commit	kyin

6. 작업 되돌리기

Git Repositories 탭

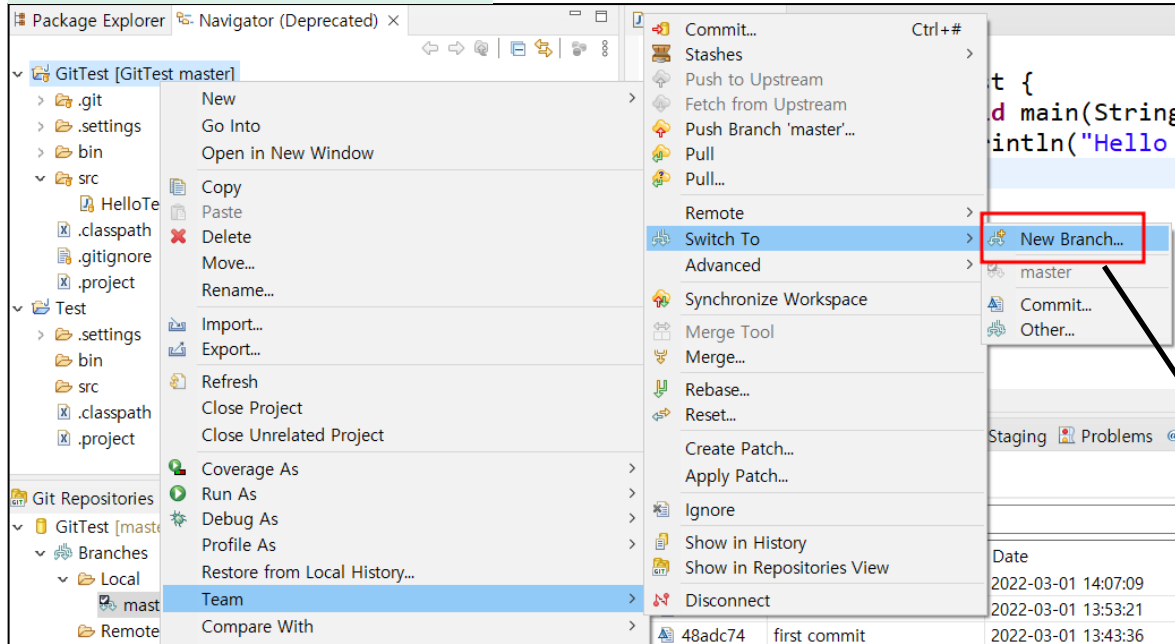


HelloTest.java

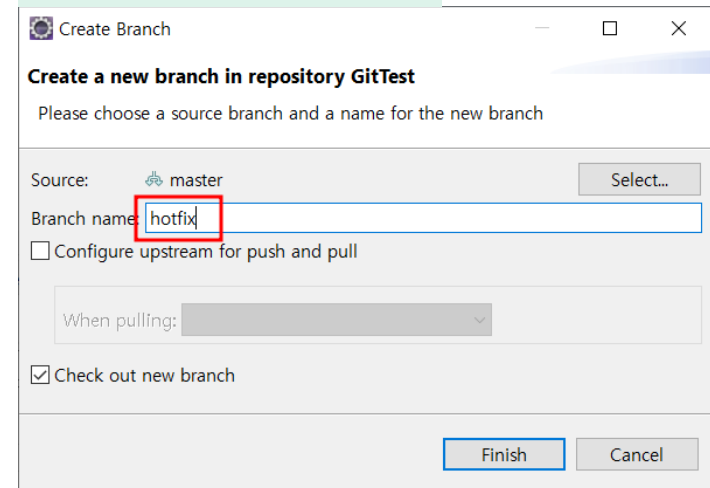
```
HelloTest.java ×
1
2 public class HelloTest {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5     }
6 }
7
```

7. 브랜치 생성

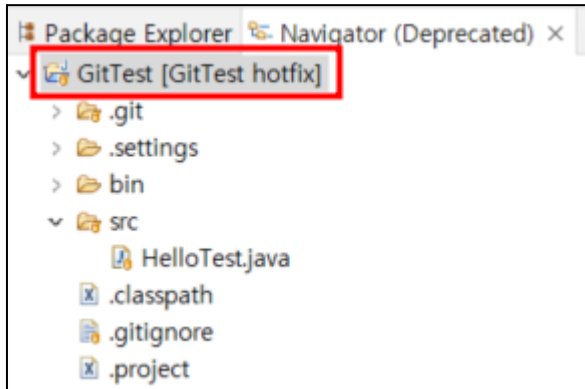
New Branch ... 선택



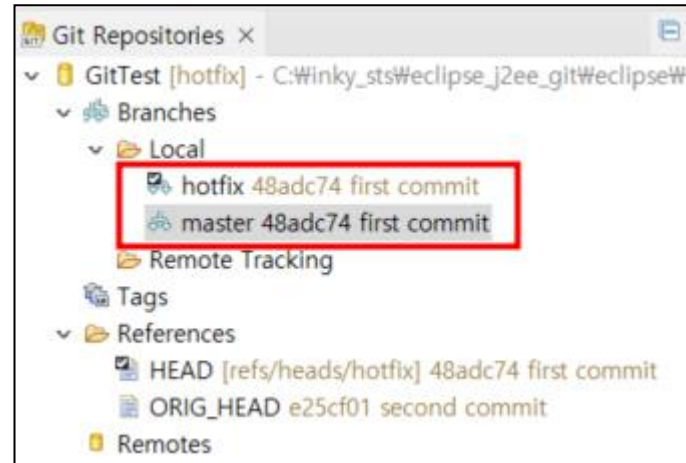
hotfix 브랜치 생성



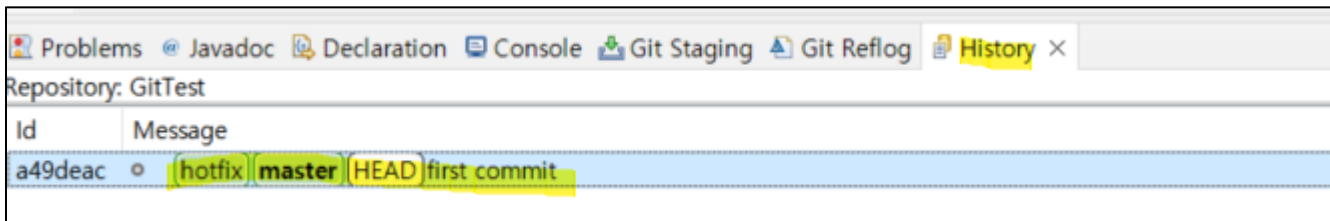
7. 브랜치 생성



Git Repositories 탭

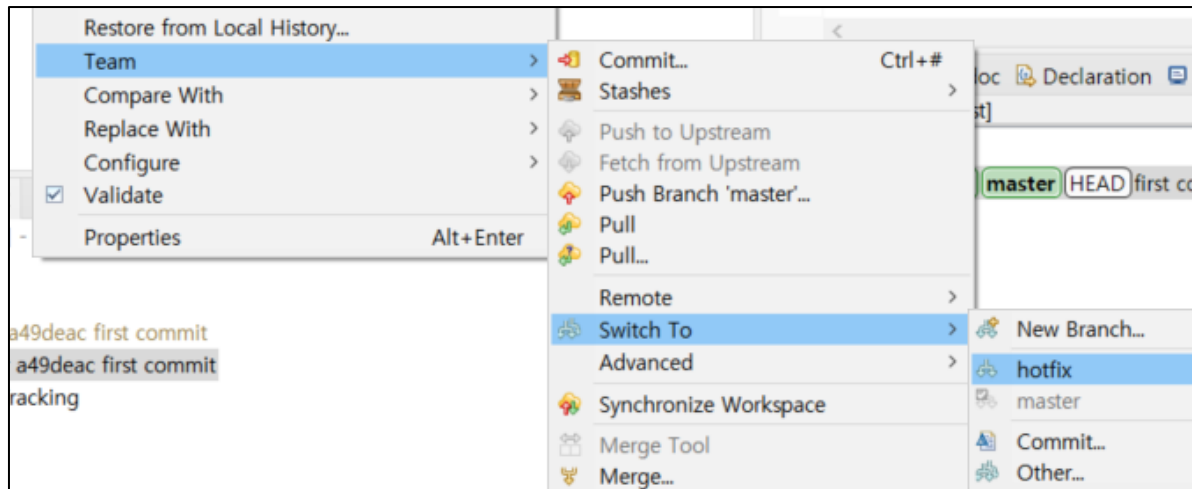
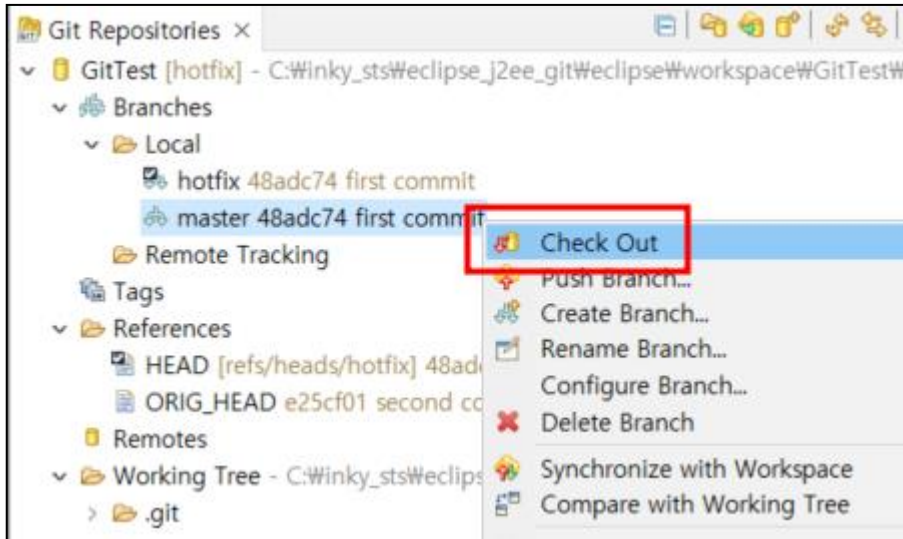


History 탭



8. 브랜치 이동

git checkout



9. 브랜치 병합

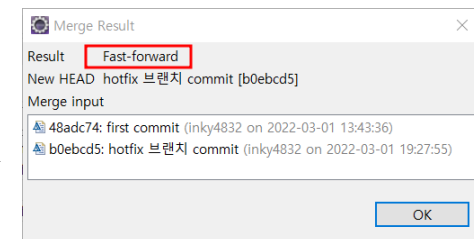
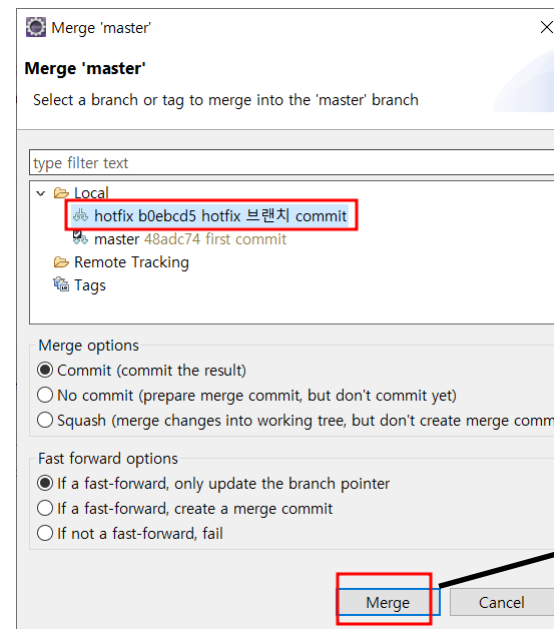
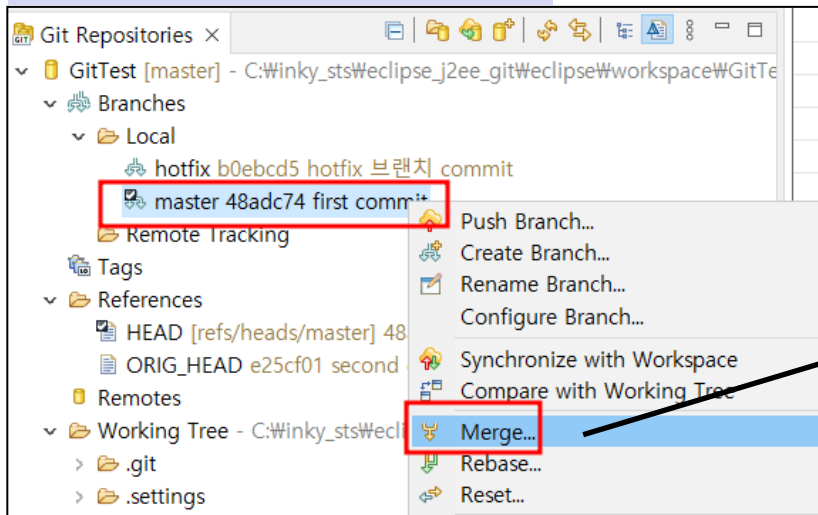
master 브랜치 최종 commit 내용

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

hotfix 브랜치 최종 commit 내용

```
HelloTest.java x
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 }
```

1) master 에서 병합



2) 병합후 master 내용

master 브랜치 최종 commit 내용

```
HelloTest.java x
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 }
```

History 탭

Id	Message
b4b12d1	hotfix commit
a49deac	first commit

10. 브랜치 병합시 충돌

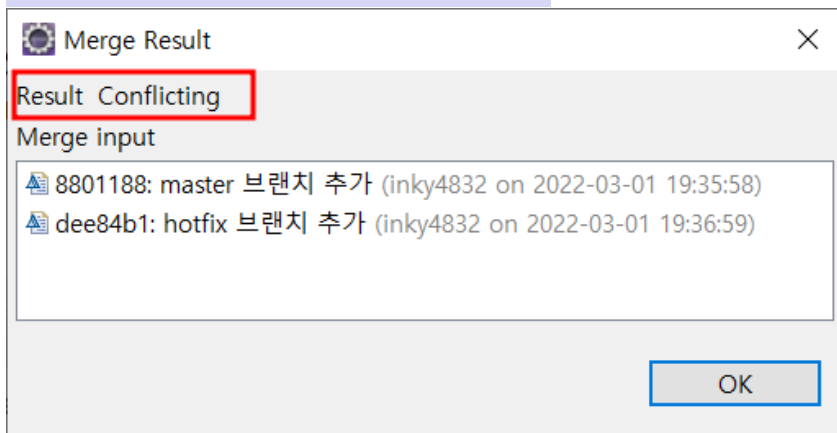
master 브랜치 최종 commit 내용

```
HelloTest.java x
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         System.out.println("master 브랜치 추가");
7     }
8 }
9 }
```

hotfix 브랜치 최종 commit 내용

```
HelloTest.java x
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         System.out.println("hotfix 브랜치 추가");
7     }
8 }
```

1) master 에서 병합



master 브랜치 병합 결과

```
HelloTest.java x
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 <<<<<<< HEAD
7         System.out.println("master 브랜치 추가");
8
9 =====
10         System.out.println("hotfix 브랜치 추가");
11 >>>>>>> refs/heads/hotfix
12     }
13 }
```

2) 충돌 해결

```
HelloTest.java x
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 <<<<<<< HEAD
7         System.out.println("master 브랜치 추가");
8
9 =====
10        System.out.println("hotfix 브랜치 추가");
11 >>>>>> refs/heads/hotfix
12    }
13 }
```

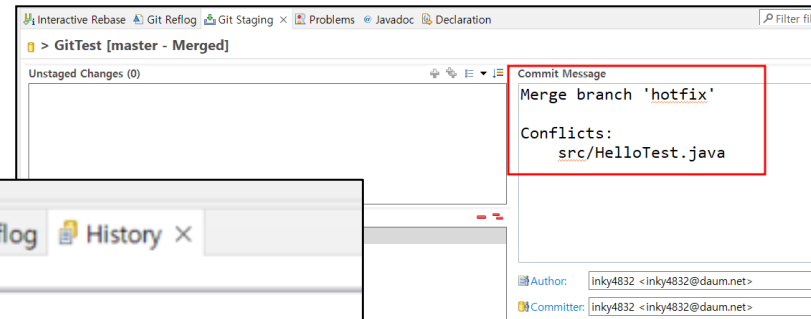


```
HelloTest.java x
1
2 public class HelloTest {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5         System.out.println("hotfix 브랜치 내용변경");
6
7         System.out.println("master 브랜치 추가");
8         System.out.println("hotfix 브랜치 추가");
9
10    }
11 }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

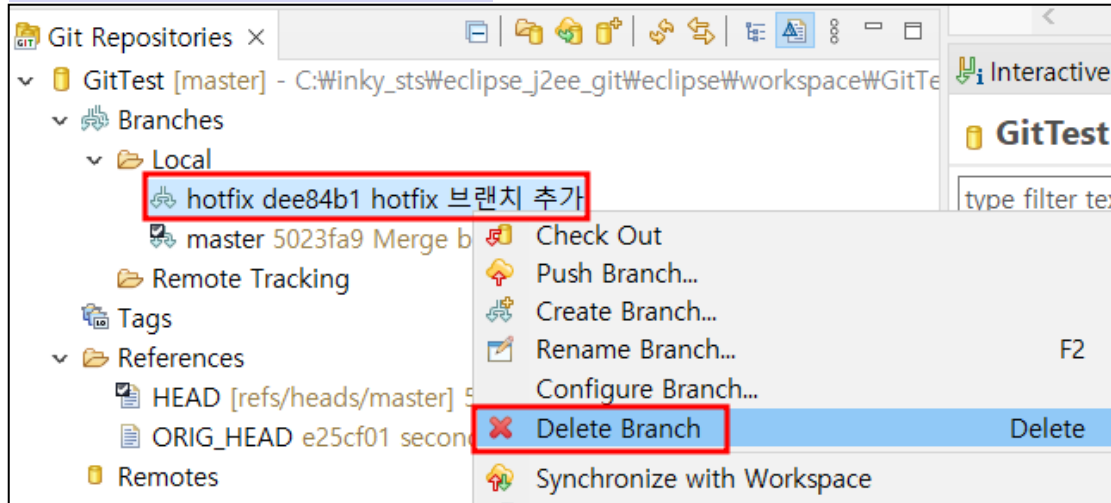
History 탭

Repository: GitTest

Id	Message
2211063	master HEAD Merge branch 'hotfix'
d61a302	hotfix 브랜치 추가
8a92ebf	master 브랜치 추가
b4b12d1	hotfix commit
a49deac	first commit



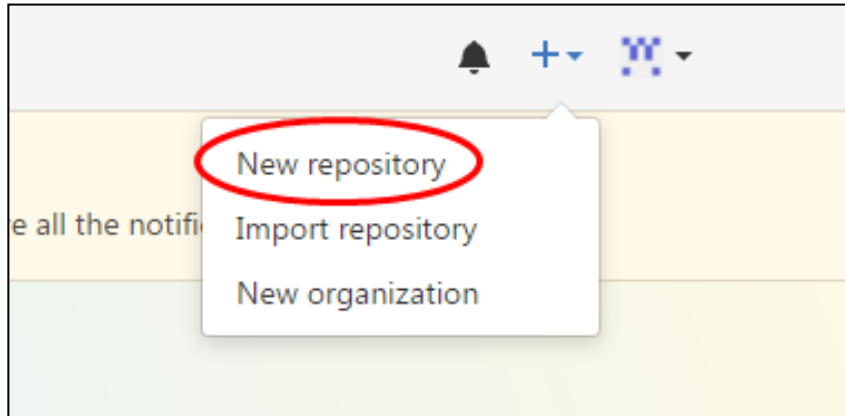
hotfix 브랜치 삭제



원격 저장소 관리

1. 원격 저장소 생성

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.](#)

Owner * kyung-yeol-in / Repository name * SamplePro ✓

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

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:

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.](#)

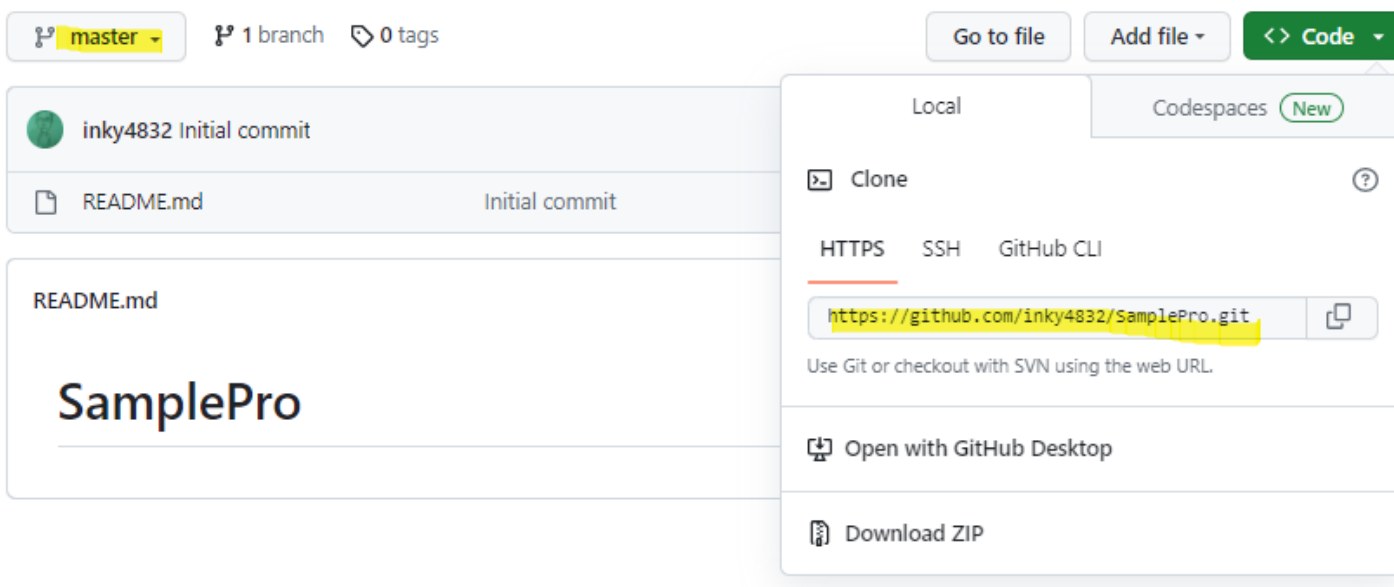
☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

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

Create repository

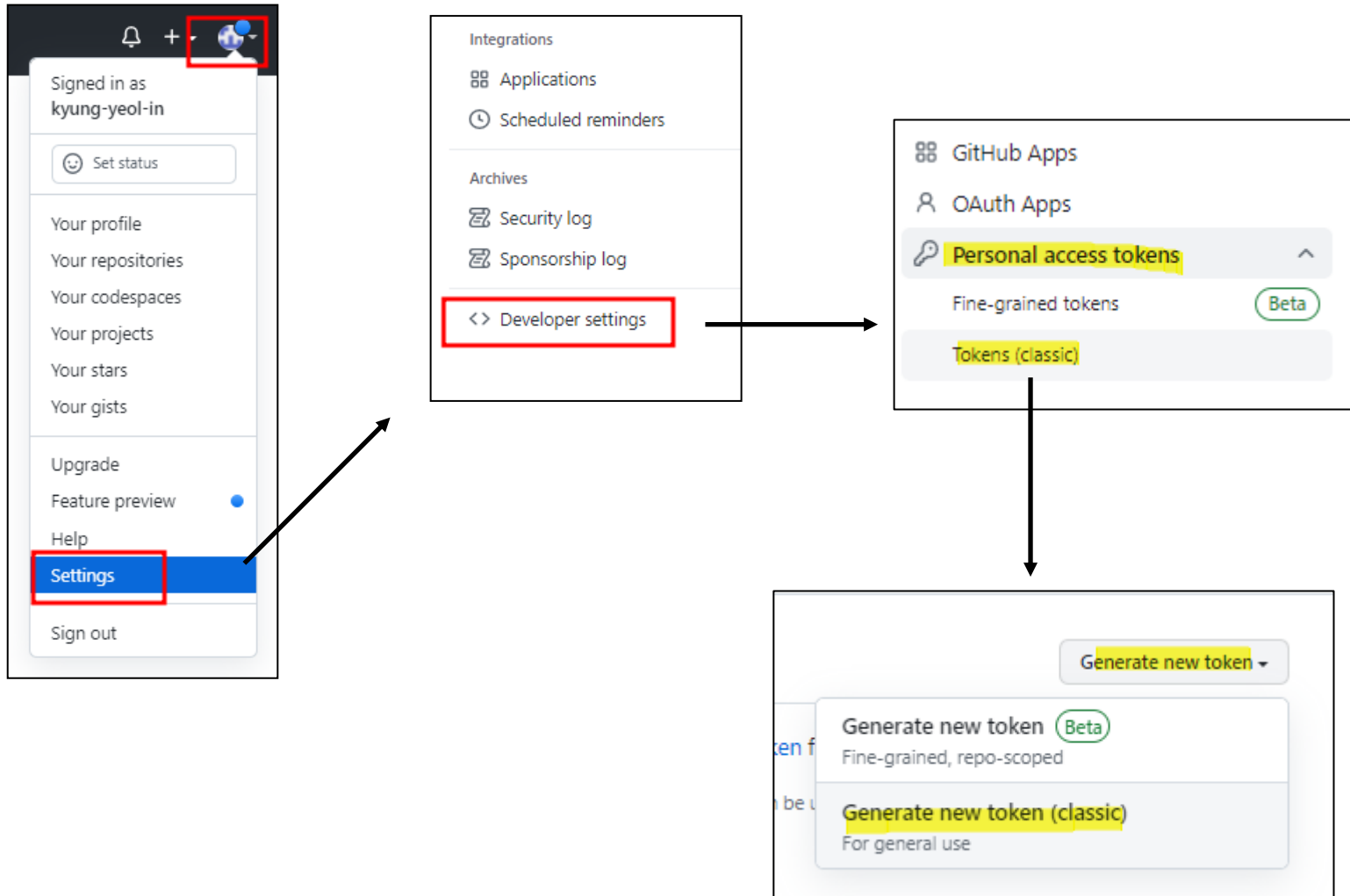
1. 원격 저장소 생성

2) 원격 저장소 주소 복사 (main → master 이름 변경)



<https://github.com/inky4832/SamplePro.git>

3) 원격 접속 token 얻기



1. 원격 저장소 생성

NOTE

SamplePro

What's this token for?

Expiration *

30 days The token will expire on Fri, Apr 1 2022

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 type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input checked="" type="checkbox"/> delete_repo	Delete repositories
<input type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> admin:enterprise	Full control of enterprises

– 생성된 토큰은 확인하고 안전한 곳에 복사해 둘 것

ghp_ZQDu9vnCLqBob5SXqe1dLkciFK
EGIN3MXb4z



Personal access tokens (classic)

Generate new token ▼ Revoke all

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

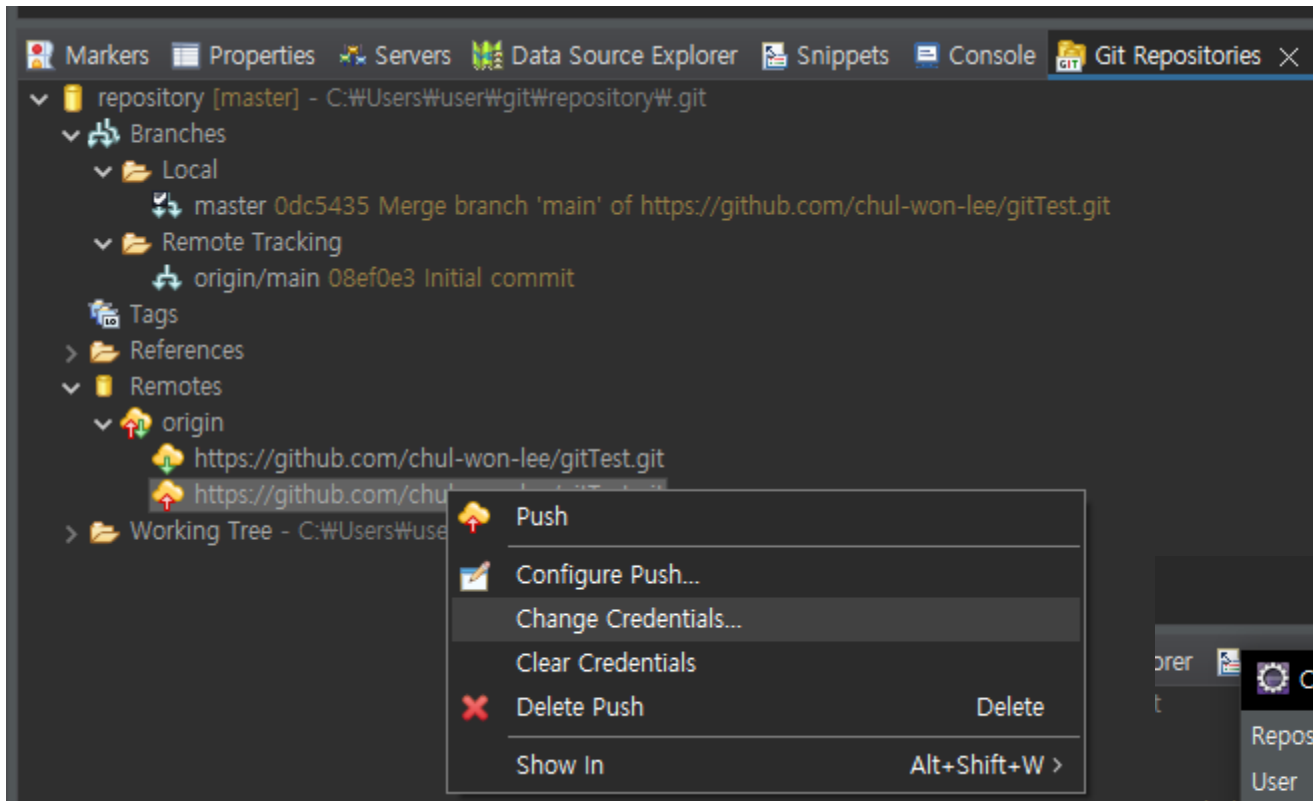
Make sure to copy your personal access token now. You won't be able to see it again!

✓ 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

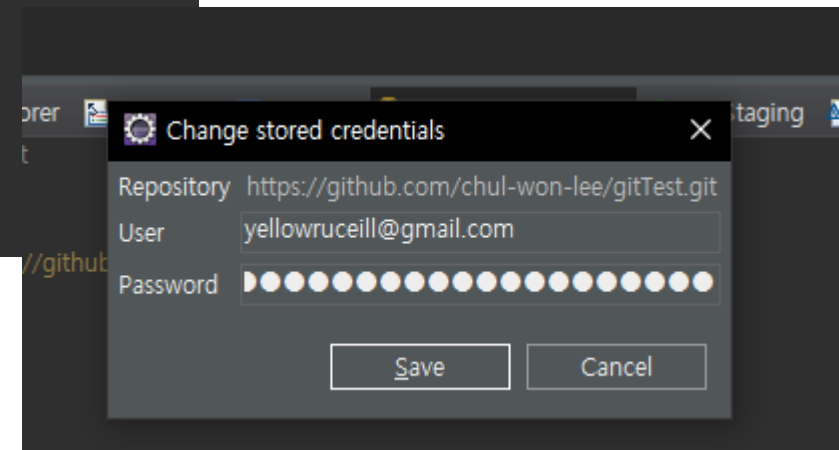
Delete

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](#).

토큰이 수정된 경우 다시 generate한 후 git repository에서 remote-push부분에 change credintials 에서 재설정함

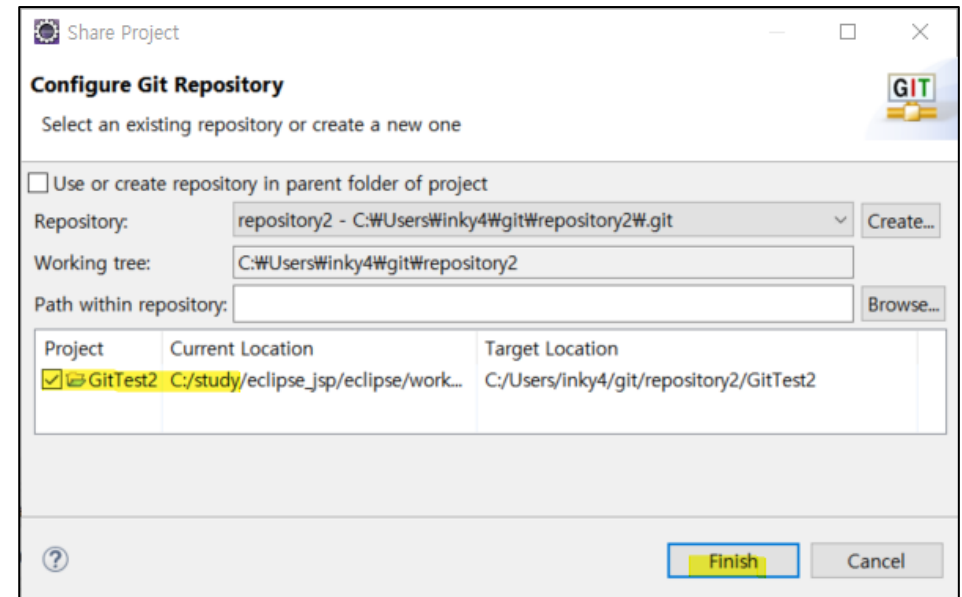
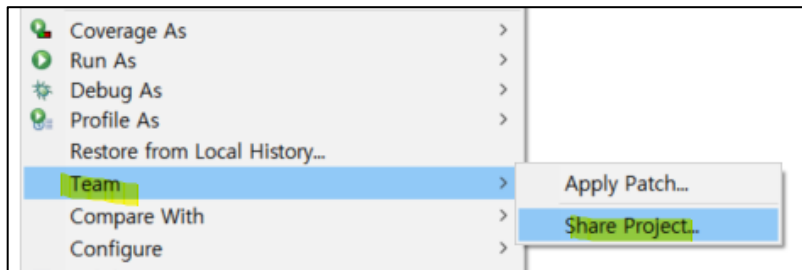
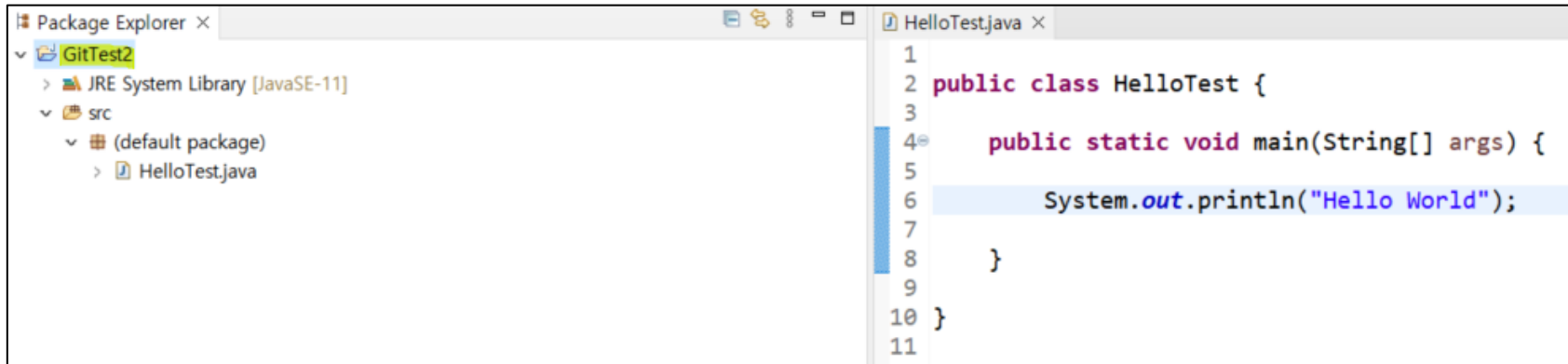


-계정/토큰 입력



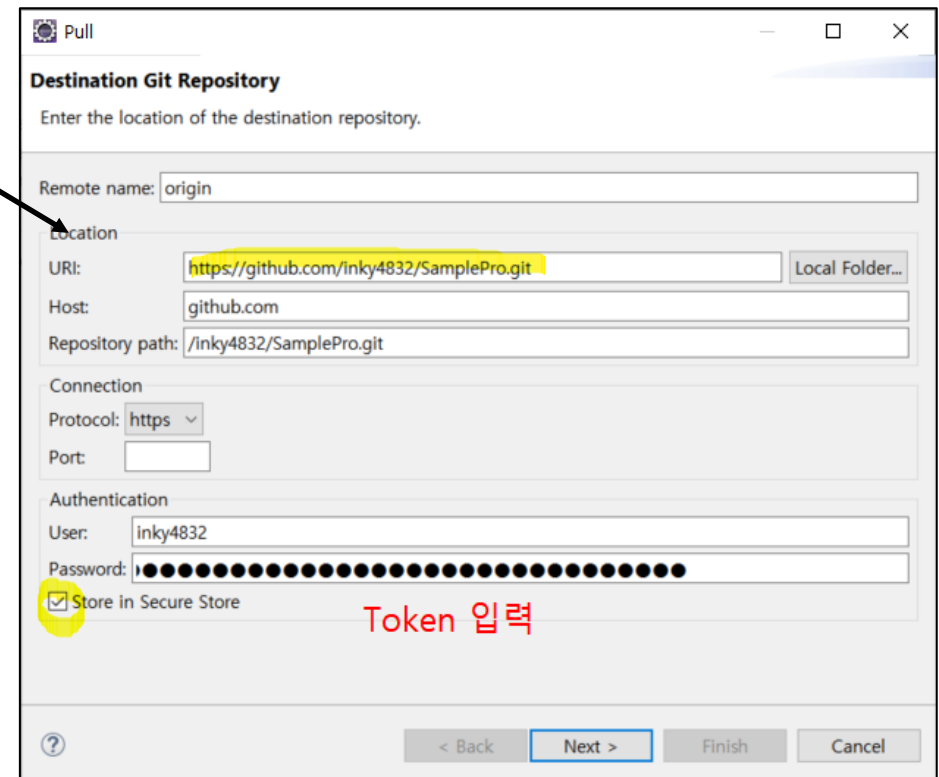
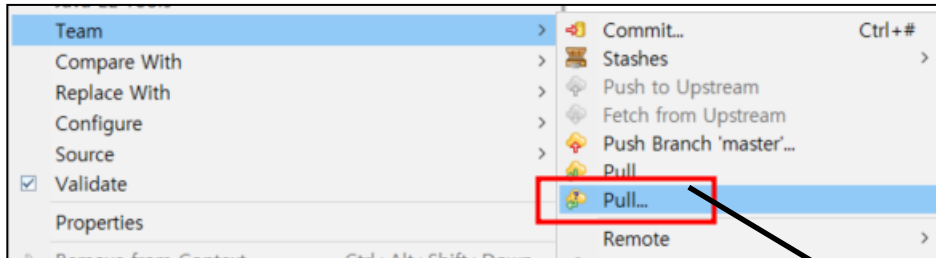
2. 원격 저장소에 저장하기

1) GitTest2 프로젝트 생성 및 로컬저장소 생성

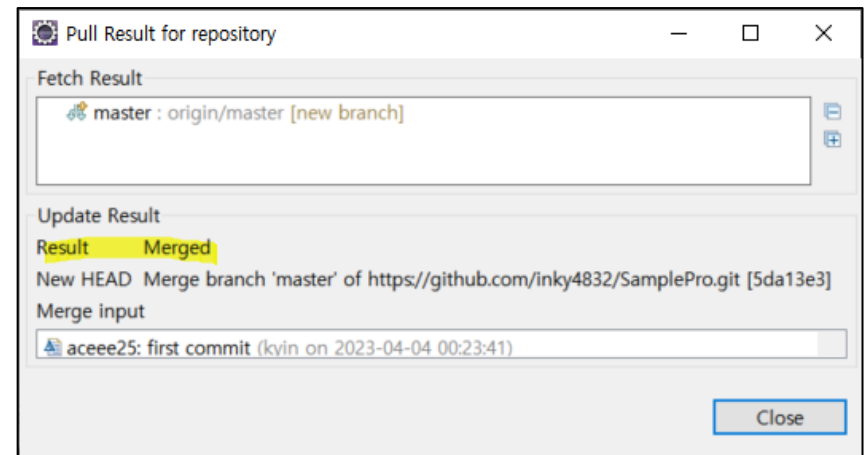
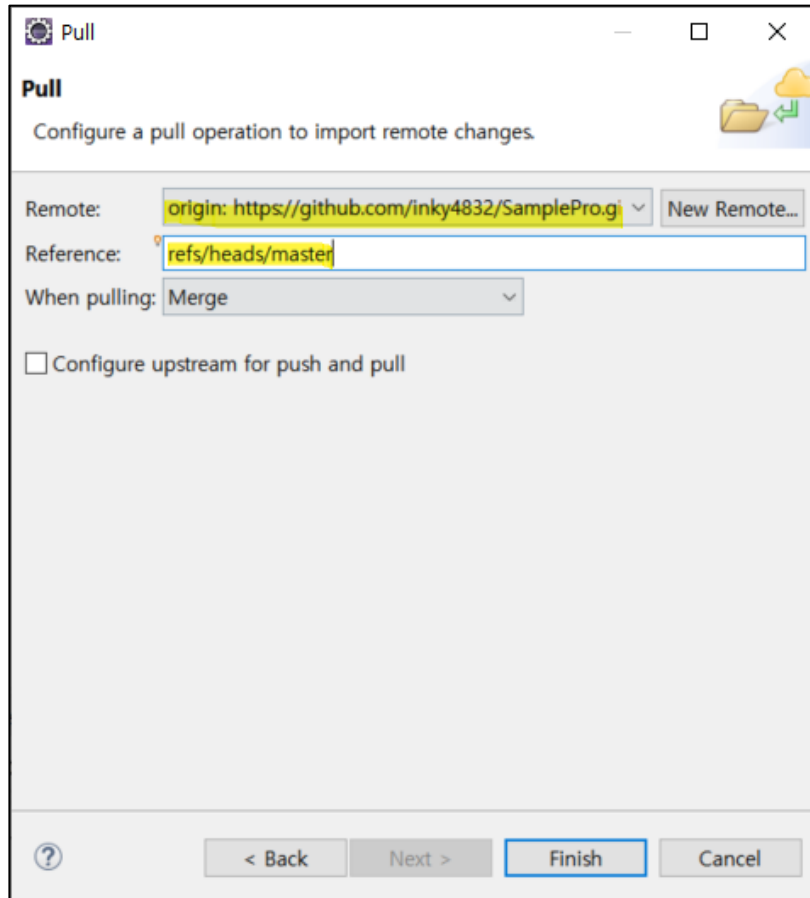


2. 원격 저장소에 저장하기

2) 원격 저장소에 저장하기 전 반드시 pull 먼저 하기

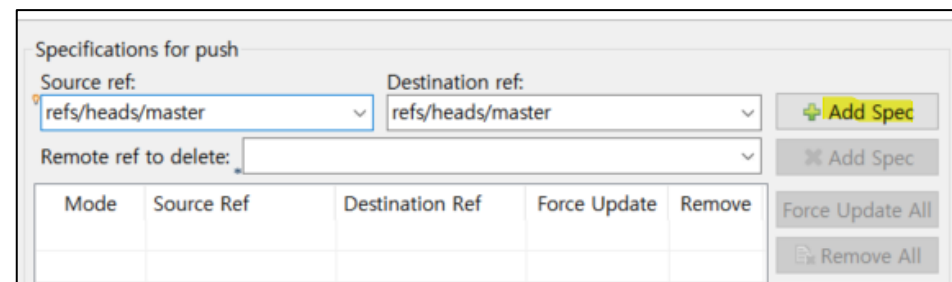
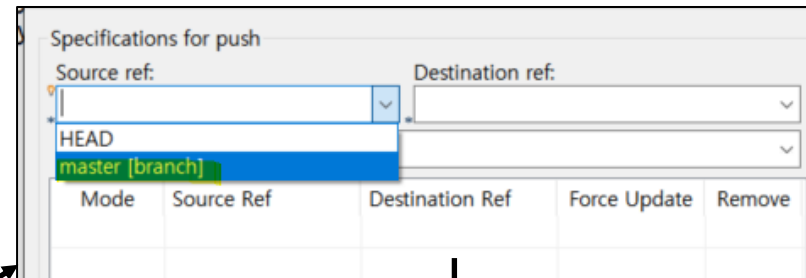
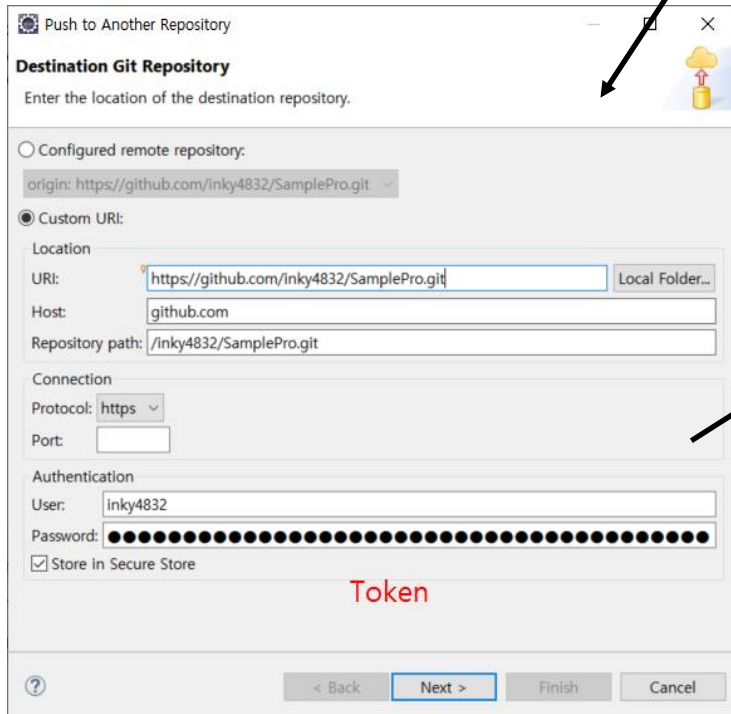
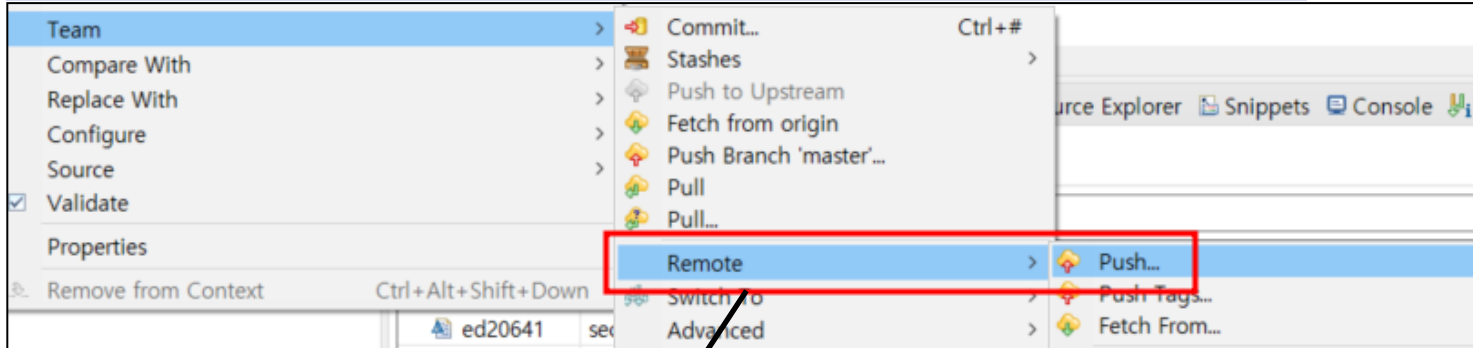


2. 원격 저장소에 저장하기

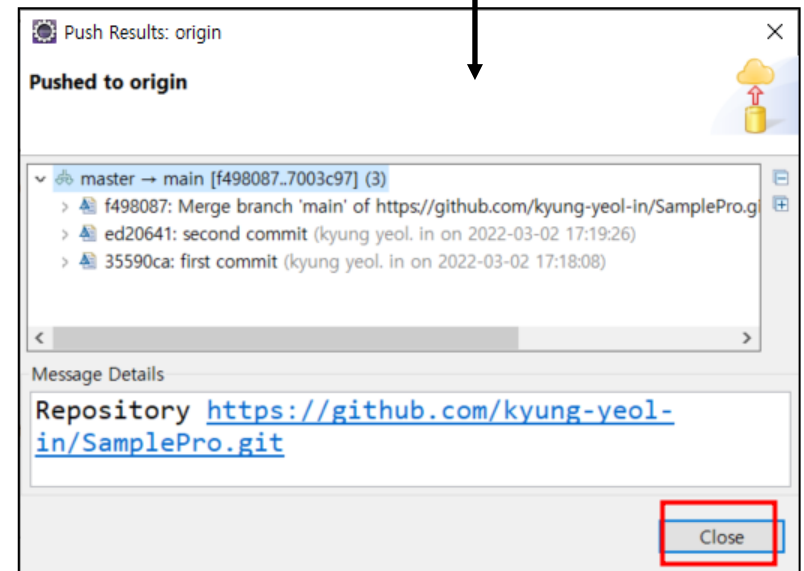
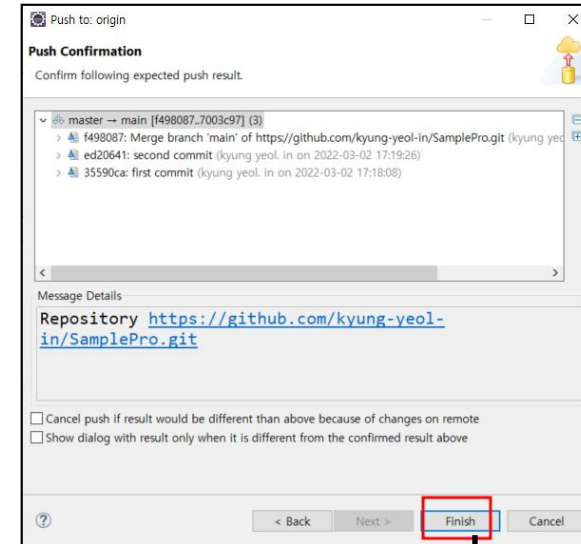
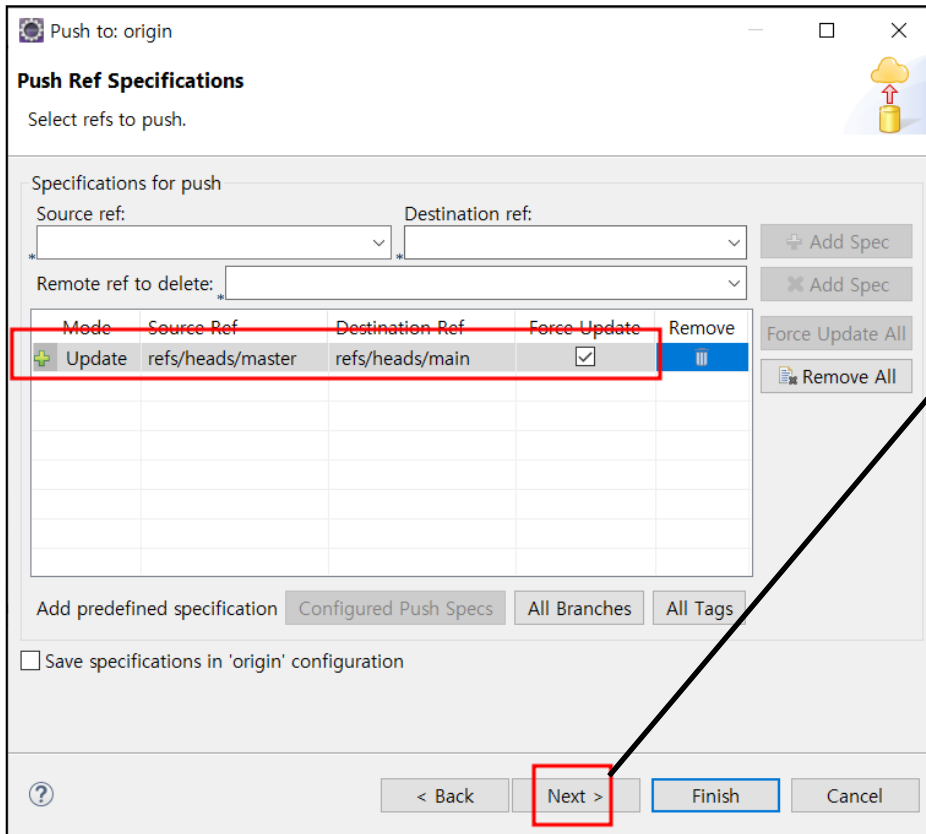


2. 원격 저장소에 저장하기

5) 원격 저장소에 저장하기 (push)

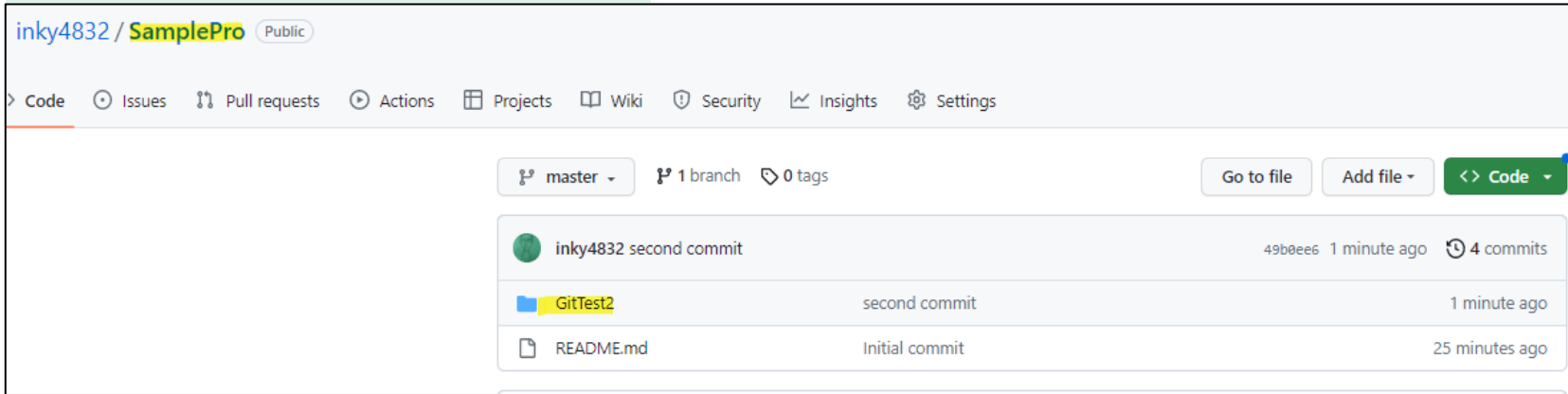


2. 원격 저장소에 저장하기



2. 원격 저장소에 저장하기

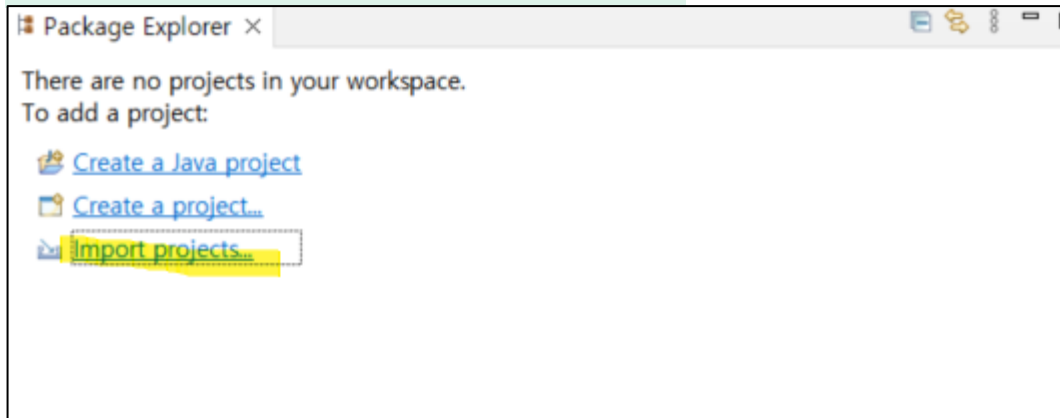
원격 저장소 최종 내용



이후부터는 원격 저장소에 저장하기 전 반드시 pull 먼저하고 나중에 push 한다.

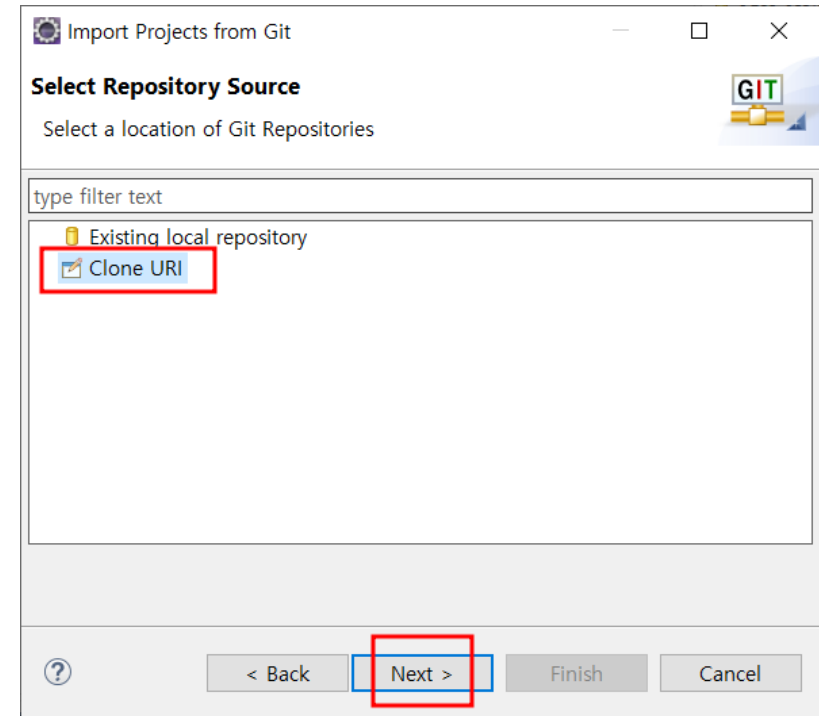
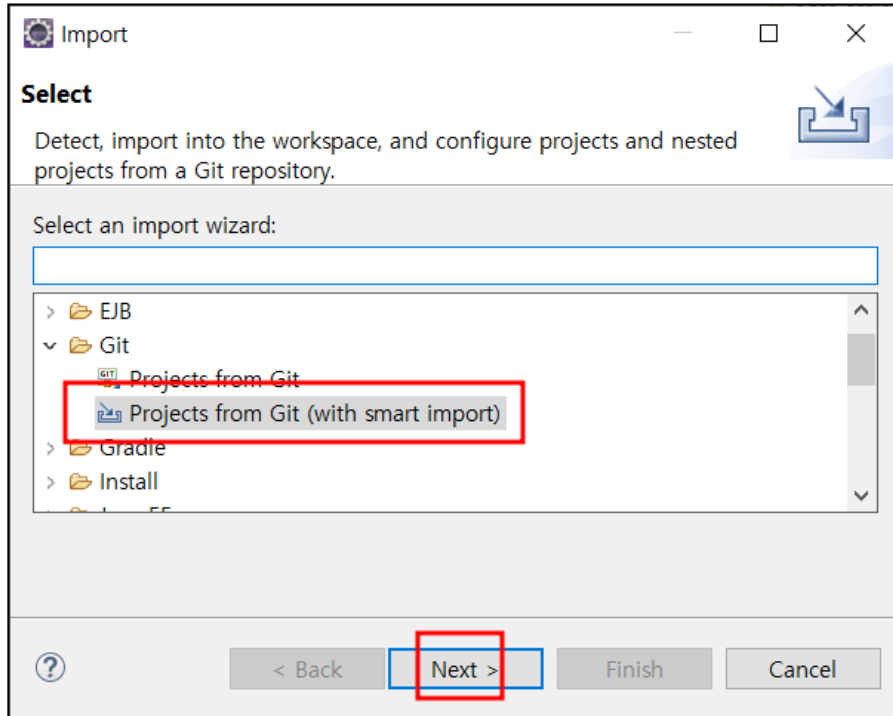
3. Eclipse로 clone 하기

현재 eclipse 프로젝트 구조 화면

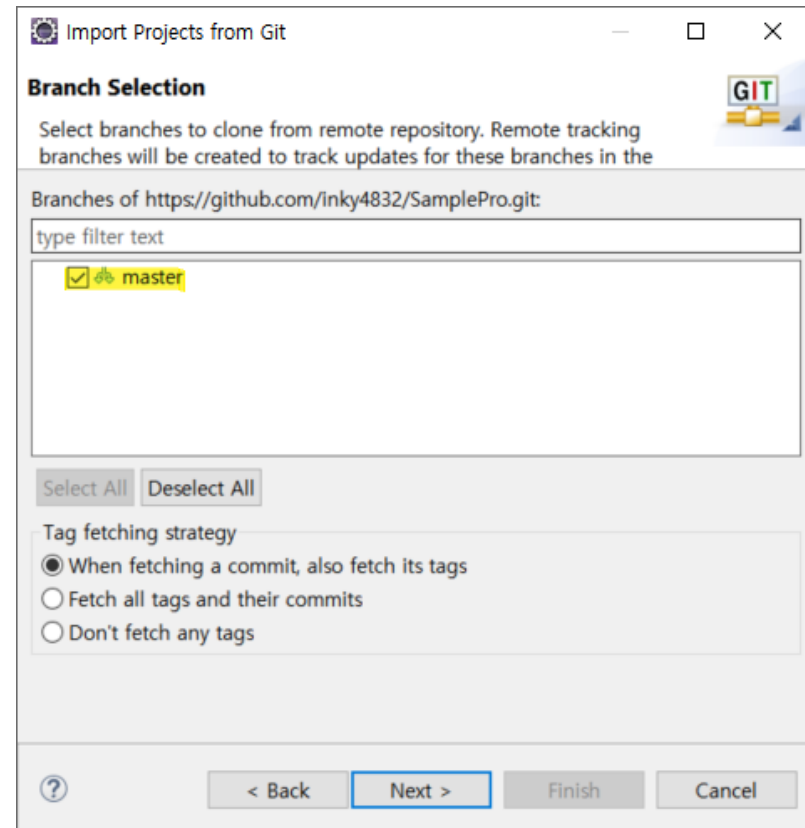
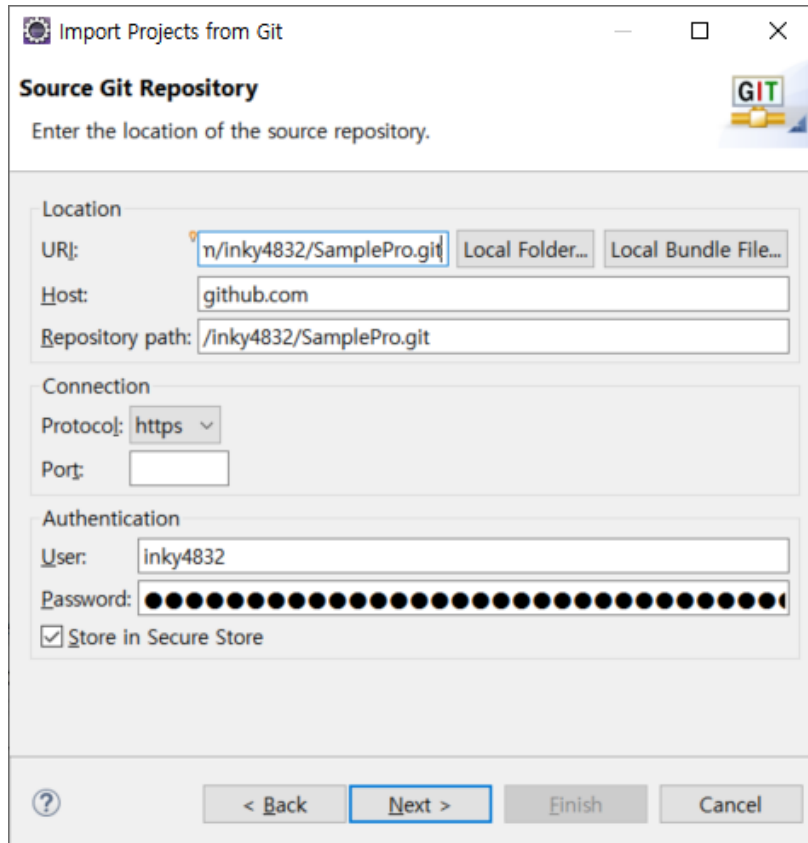


3. Eclipse로 clone 하기

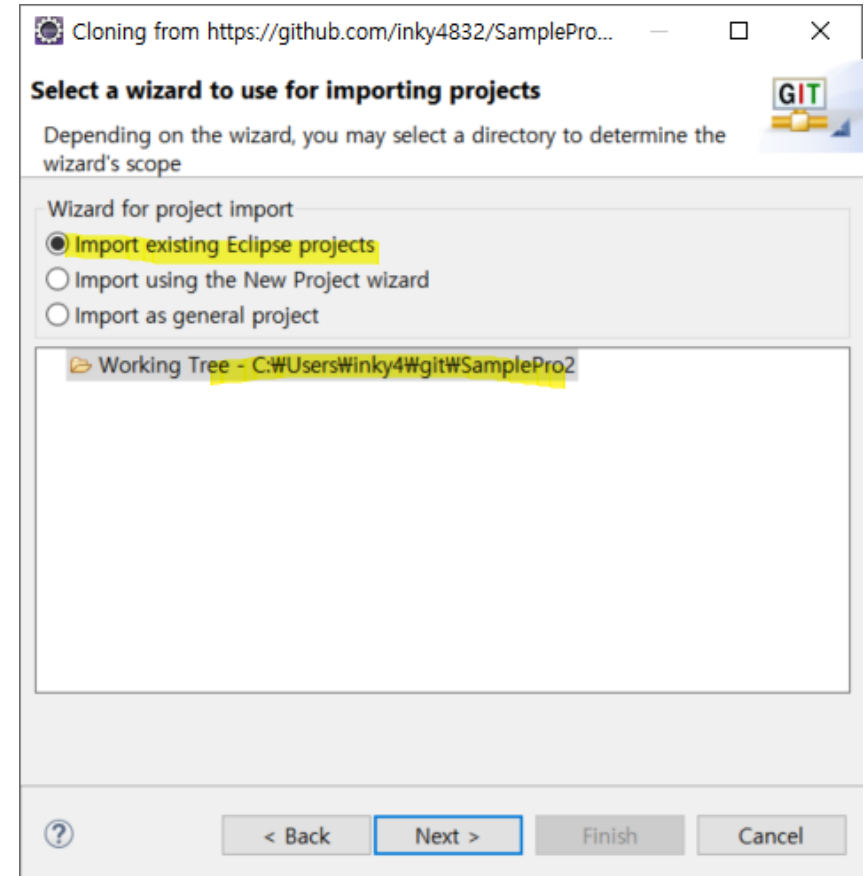
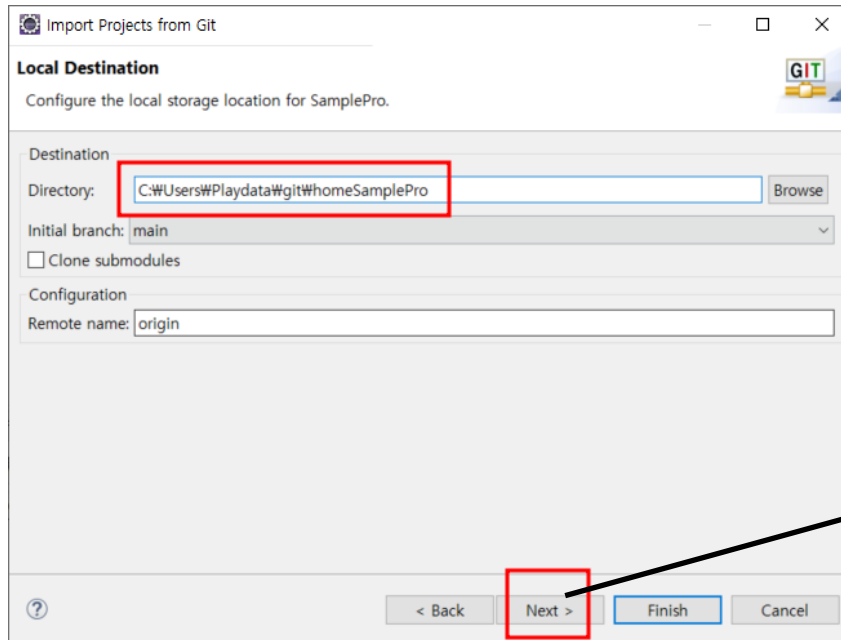
1) 원격 저장소를 로컬 저장소로 clone 하기



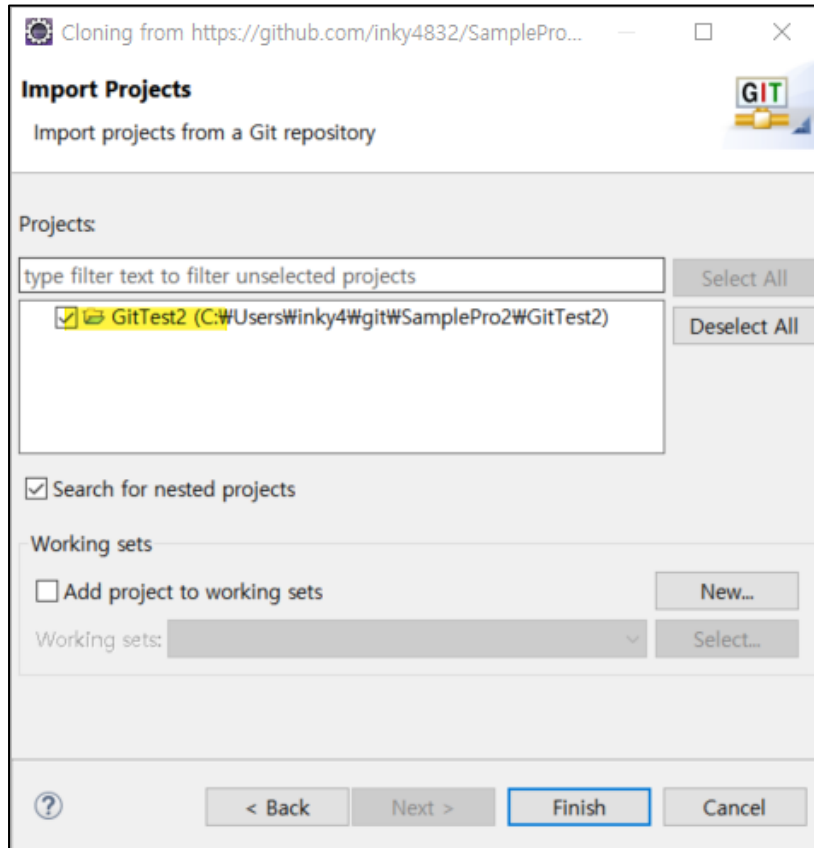
3. Eclipse로 clone 하기



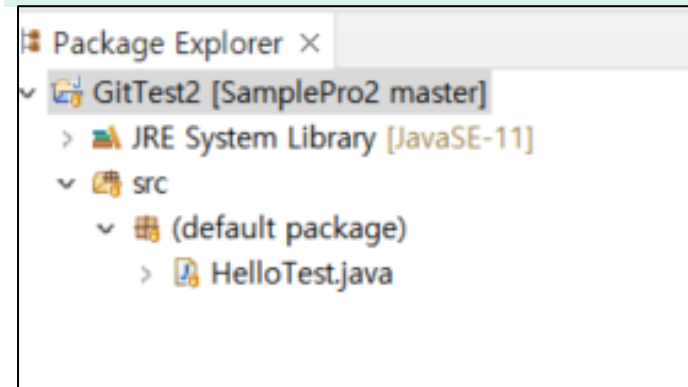
3. Eclipse로 clone 하기



3. Eclipse로 clone 하기



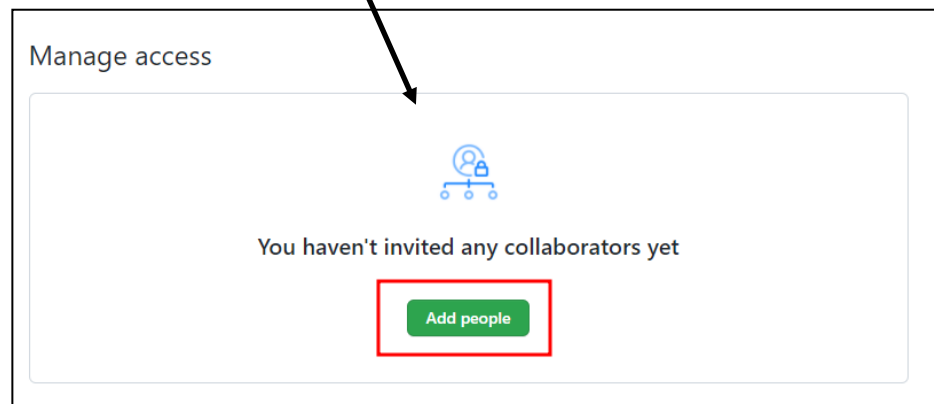
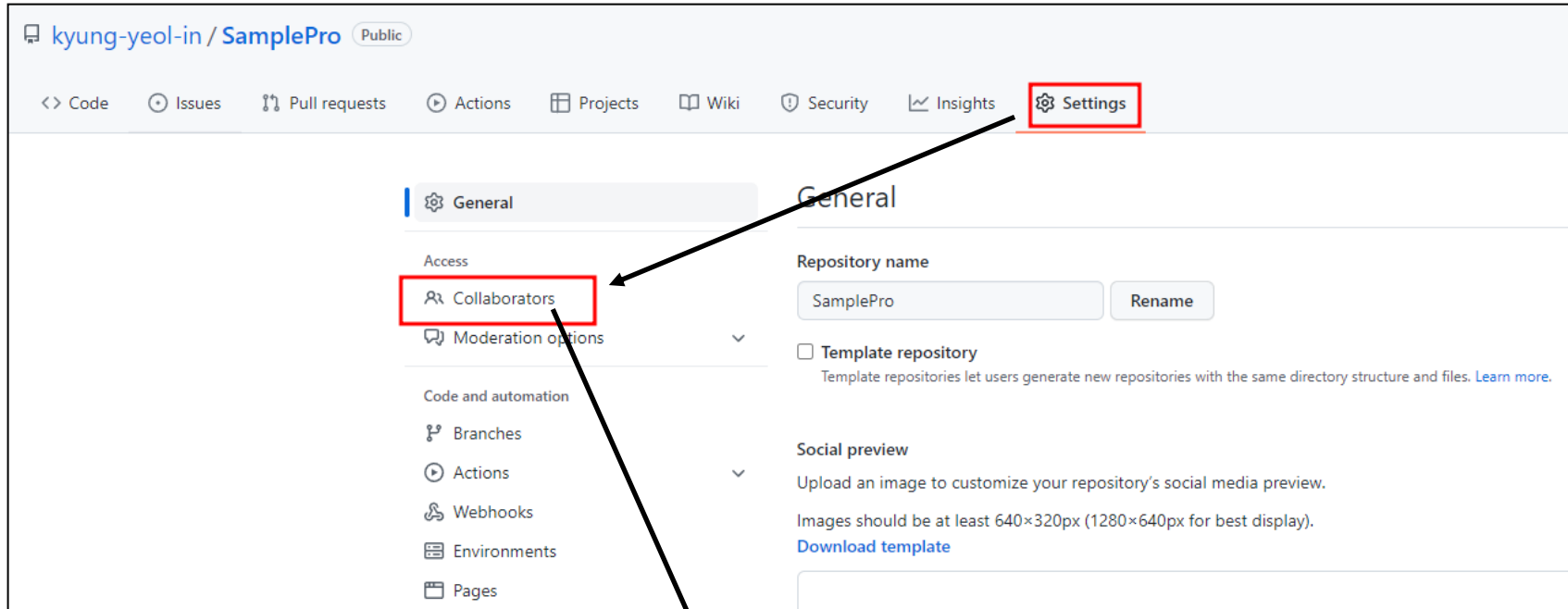
clone 후 eclipse 프로젝트 구조 화면



협업 처리

1. Collaborators 추가

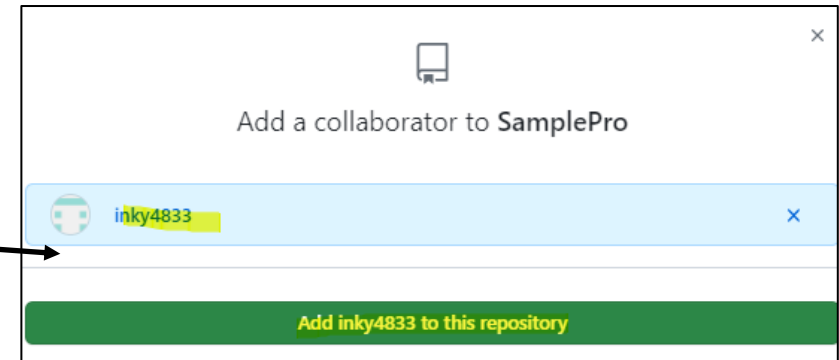
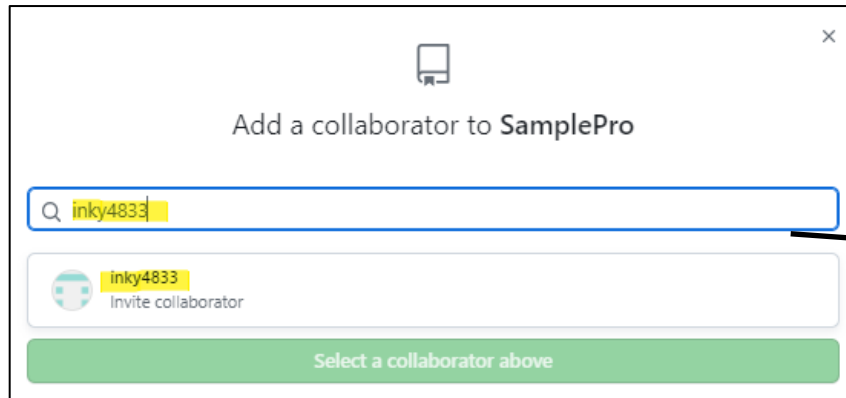
1) 조장(inky4832)이 로그인후 Collaborators 링크를 선택한다



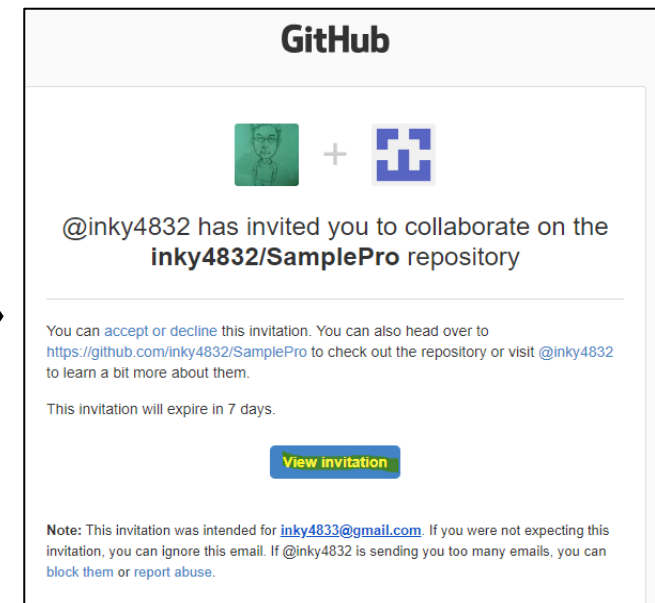
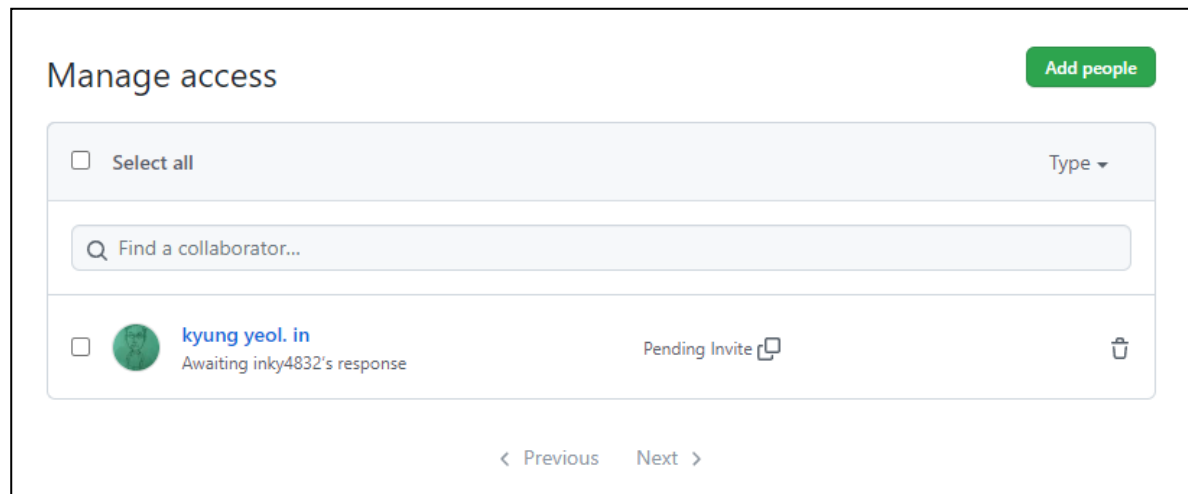
1. Collaborators 추가

2) 조원 (inky4833@gmail.com)으로 등록

이메일로 검색할 것

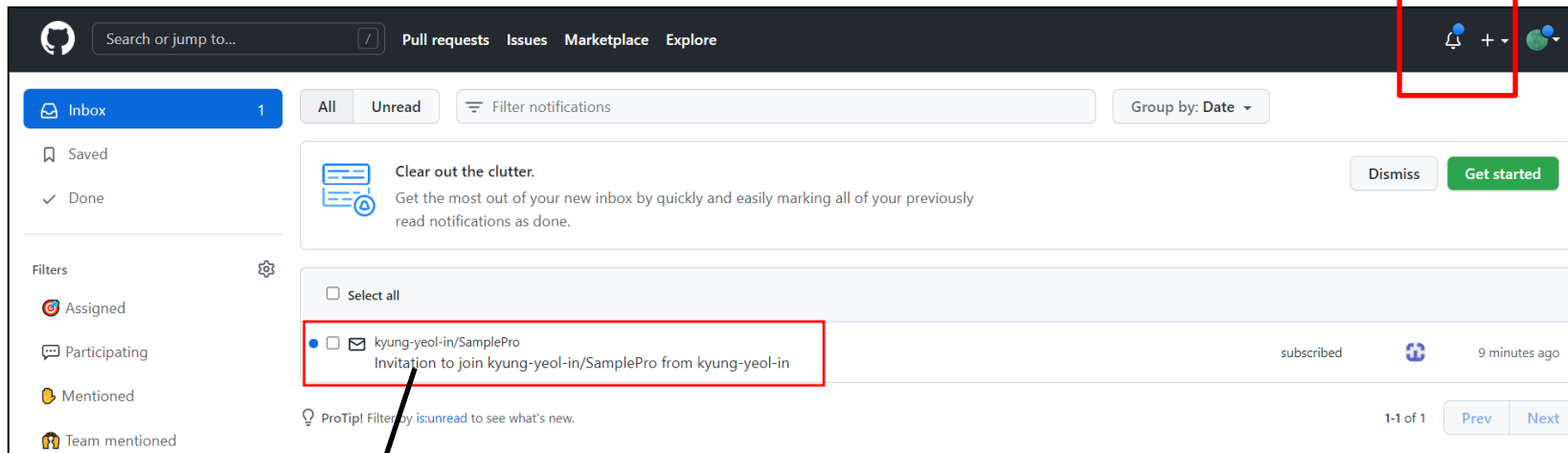


이메일 전송됨

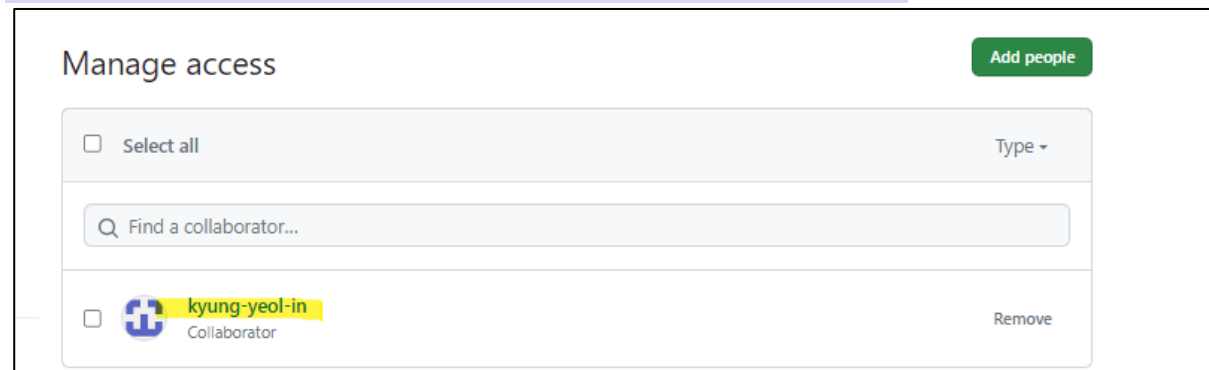
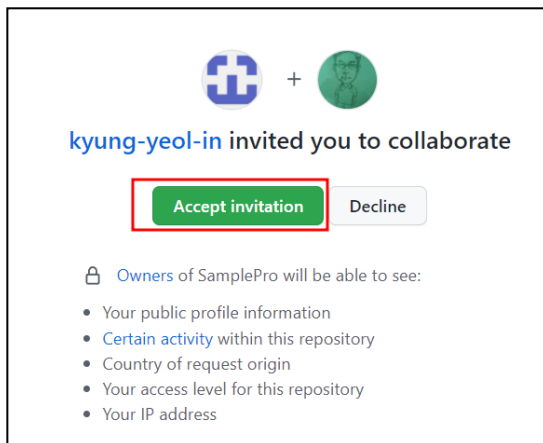


1. Collaborators 추가

3) inky4833 로그인 후 알림 선택



4) 조원으로 inky4833 등록됨



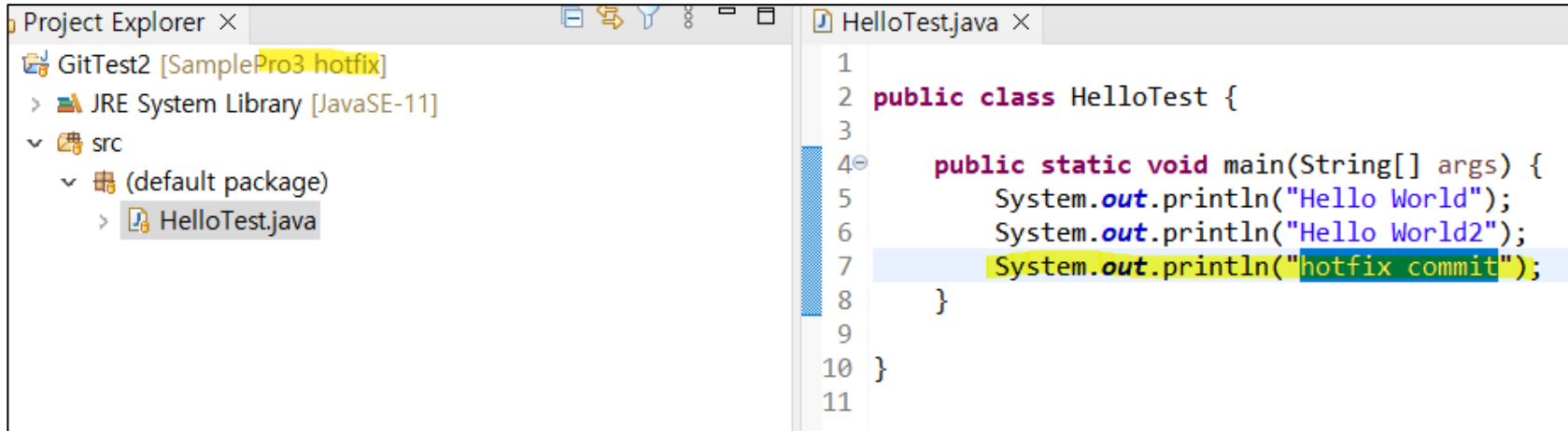
Git 활용한 버전관리

5) 조원들은 조장이 GitHub에 Push한 리소스를 조원 Eclipse로 clone한다.

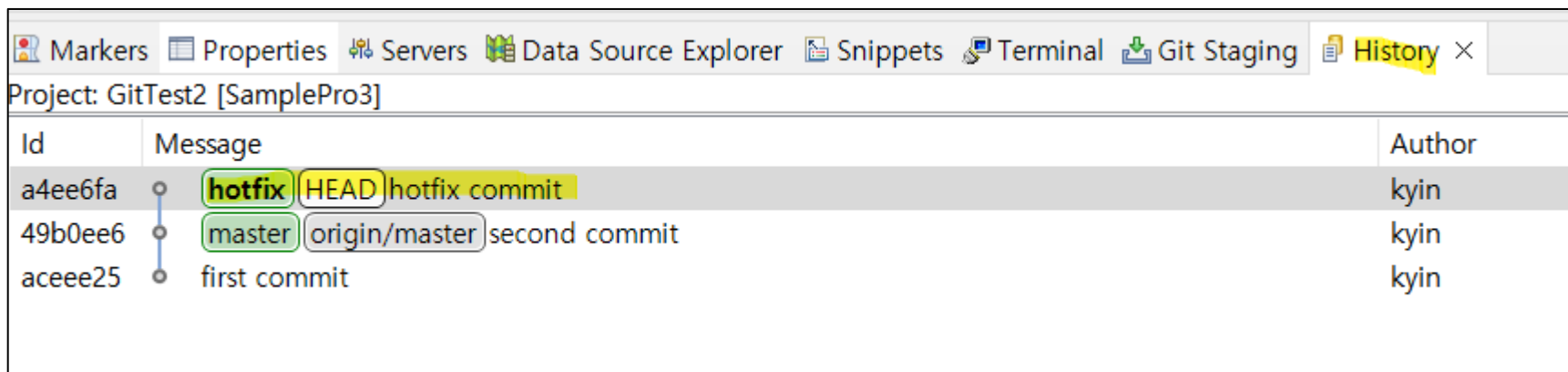
[illegible]

3. hotfix 브랜치 생성

6) hotfix 브랜치 생성 및 추가작업후 commit



```
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
```



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 interface shows the 'Remote' menu open, with 'Push...' highlighted. A red box highlights the 'Remote' menu and the 'Push...' option. A large white arrow points from the 'Push...' option to the 'Push to Another Repository' dialog box on the right.

The 'Push to Another Repository' dialog box is titled 'Destination Git Repository' and prompts the user to 'Enter the location of the destination repository.' It features two radio buttons: 'Configured remote repository:' and 'Custom URI:'. The 'Custom URI:' option is selected. The 'Location' section contains the following fields:

- URI: `https://github.com/inky4832/SamplePro.git`
- Host: `github.com`
- Repository path: `/inky4832/SamplePro.git`

The 'Connection' section shows the 'Protocol' set to 'https' and the 'Port' field empty. The 'Authentication' section shows the 'User' field with the value 'inky4832' and the 'Password' field masked with dots. The 'Store in Secure Store' checkbox is checked. At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

4. hotfix 브랜치 push

Push to: origin

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/hotfix	refs/heads/hotfix	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Add predefined specification Configured Push Specs All Branches All Tags

☐ Save specifications in 'origin' configuration

Push to: origin

Push Confirmation

Confirm following expected push result.

hotfix → hotfix [new branch]

Message Details

Repository <https://github.com/kyung-yeol-in/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: origin

Pushed to origin

hotfix → hotfix [new branch]

Message Details

Repository <https://github.com/kyung-yeol-in/SamplePro.git>

4. hotfix 브랜치 push

hotfix push후 원격 저장소 구조 화면

The screenshot displays a GitHub repository interface for a branch named 'hotfix'. At the top, a yellow banner indicates that 'hotfix' has recent pushes less than a minute ago, with a 'Compare & pull request' button. Below this, the repository name 'hotfix' is shown with a dropdown arrow, along with '2 branches' and '0 tags'. Action buttons include 'Go to file', 'Add file', and a green 'Code' button. A status bar states 'This branch is 1 commit ahead of master.' with a 'Contribute' dropdown. The commit history shows three entries: a commit by 'inky4832' labeled 'hotfix commit' (hash a4ee6fa, 3 minutes ago, 5 commits), a commit labeled 'GitTest2' (3 minutes ago), and an 'Initial commit' for 'README.md' (1 hour ago). The file list shows 'README.md'. The content of the README.md file is visible, displaying 'SamplePro'.

5. master 브랜치에 hotfix 병합

8) hotfix 브랜치를 master 브랜치에 병합

The screenshot shows the GitHub repository interface for a project. At the top, a yellow banner indicates 'hotfix had recent pushes 1 minute ago' with a green button labeled 'Compare & pull request'. Below this, the repository status shows 'master' as the current branch, with '2 branches' and '0 tags'. A message states 'Your master branch isn't protected' with a 'Protect this branch' button. The commit history shows a commit by 'inky4832' titled 'second commit' from 1 hour ago. An arrow points from the 'Compare & pull request' button to the 'Open a pull request' section below.

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](#).

base: master ← compare: hotfix ✓ Able to merge. These branches can be automatically merged.

hotfix commit

Write Preview H B I IE <> 🔗 ⌨️ ⌨️ ⌨️ @ 📎 ↩️ 🗑️

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

5. master 브랜치에 hotfix 병합

The image illustrates the process of merging a hotfix into the master branch on GitHub, shown in three sequential screenshots.

Top Screenshot: Pull Request View

- Title: **hotfix commit #1**
- Status: **Open** (inky4832 wants to merge 1 commit into master from hotfix)
- Conversation: 0, Commits: 1, Checks: 0, Files changed: 1
- Comment by inky4832: "No description provided."
- Commit: hotfix commit (a4ee6fa)
- Message: "Add more commits by pushing to the hotfix branch on inky4832/SamplePro."
- Checklist:
 - Require approval from specific reviewers before merging (Branch protection rules ensure specific people approve pull requests before they're merged)
 - Continuous integration has not been set up (GitHub Actions and several other apps can be used to automatically catch bugs and errors)
 - ✓ This branch has no conflicts with the base branch** (Merging can be performed automatically)
- Buttons: **Merge pull request** (highlighted with a green box and an arrow pointing to the next screenshot), "You can also open this in GitHub Desktop or view command line instructions"

Middle Screenshot: Merge Confirmation Dialog

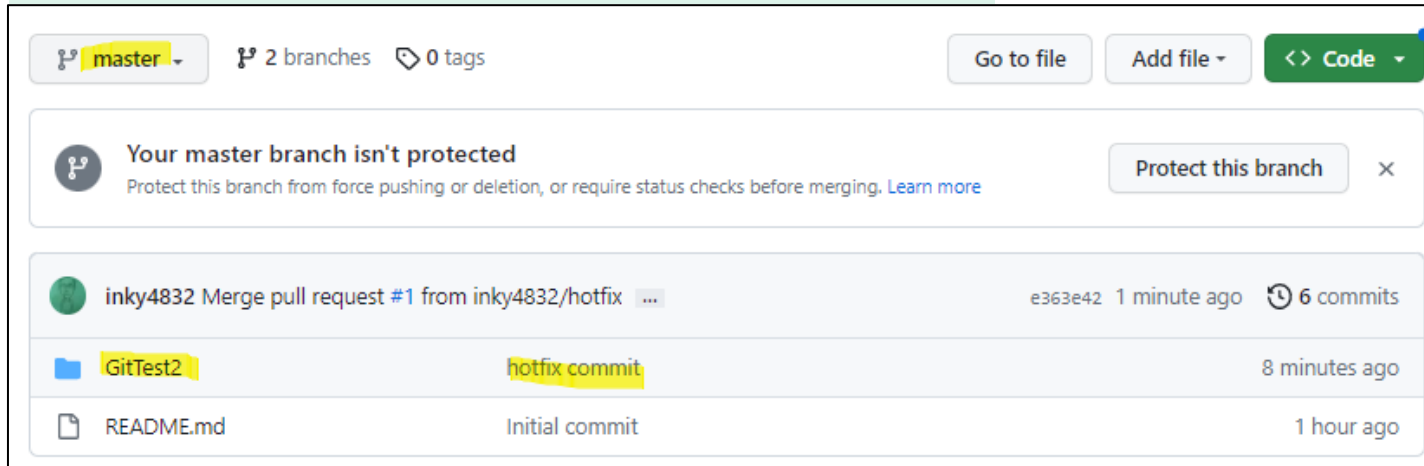
- Message: "Add more commits by pushing to the hotfix branch on inky4832/SamplePro."
- Text: "Merge pull request #1 from inky4832/hotfix"
- Text: "hotfix commit"
- Text: "This commit will be authored by inky4832@daum.net"
- Buttons: **Confirm merge** (highlighted with a green box), Cancel

Bottom Screenshot: Pull Request Closed

- Status: **inky4832 merged commit e363e42 into master now** (with a **Revert** button)
- Message: **Pull request successfully merged and closed** (highlighted with a yellow box). "You're all set—the hotfix branch can be safely deleted." (with a **Delete branch** button)
- Editor: Write, Preview, and rich text formatting options (H, B, I, etc.)

5. master 브랜치에 hotfix 병합

병합후 원격 저장소 master 브랜치 화면



이후부터는 다른 조원들이 병합된 master 브랜치를 pull 한다.



Thank you
