


Boot Project Team share

1. 팀장이 Github에 먼저 프로젝트를 생성성한다.

Required fields are marked with an asterisk (*).

Owner *

 kdgfox

Repository name *

miniproject

miniproject is available.

Great repository names are short and memorable. Need inspiration? How about **probable-octo-palm-tree** ?

Description (optional)

backend miniproject

☐ 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.](#)

2. sts에서 Spring start project를 이용해서 String boot project를 생성한다. (spring boot 환경설정.pdf 참조)
 - 2.1 application.properties 에 DB 설정을 해야 한다. (project 생성시 mybatis와 mysql를 생성했기 때문에)

```
# DataBase Settings : hikariCP : https://github.com/brettwooldridge/HikariCP
spring.datasource.hikari.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.jdbc-url=jdbc:mysql://localhost:3306/uereka?serverTimezone=UTC
spring.datasource.hikari.username=uereka
spring.datasource.hikari.password=uereka
spring.datasource.hikari.pool-name=hikari-pool

# hikariCP property setting
spring.datasource.hikari.maximum-pool-size=50
spring.datasource.hikari.minimum-idle=50
spring.datasource.hikari.connection-timeout=5000
spring.datasource.hikari.connection-init-sql=SELECT 1
spring.datasource.hikari.idle-timeout=600000
spring.datasource.hikari.max-lifetime=1800000
spring.datasource.hikari.auto-commit=true
```

- 2.2 config에 패키지 추가한 후 DataBaseConfiguration, WdbMvcConfiguration 파일을 넣어준다.

```
▼ ? > com.uplus.eureka.config
    > ? DataBaseConfiguration.java
    > ? WebMvcConfiguration.java
```

2.3 swagger 설정하는 경우 참조'

2.3.1 pom.xml 에 의존성 추가

```
<!-- swagger를 위한 의존성 추가. springdoc로 검색 : SpringDoc OpenAPI Starter Web  
<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui</-->  
<dependency>  
    <groupId>org.springdoc</groupId>  
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>  
    <version>2.6.0</version>  
</dependency>
```

2.3.2 application.properties 에 swagger 설정 추가 하기

```
# swagger v2.6 setting  
springdoc.packages-to-scan=com.uplus.eureka.book.controller,com.uplus.eureka.member.controller  
springdoc.paths-to-match=/**  
springdoc.default-consumes-media-type=application/json;charset=UTF-8  
springdoc.default-produces-media-type=application/json;charset=UTF-8  
springdoc.swagger-ui.enabled=true  
springdoc.swagger-ui.path=/swagger-ui.html  
springdoc.swagger-ui.tags-sorter=alpha  
springdoc.swagger-ui.operations-sorter=alpha  
springdoc.api-docs.path=/v3/api-docs  
springdoc.api-docs.groups.enabled=true  
springdoc.cache.disabled=true
```

2.3.3 config에 Swagger config 추가하기

```
▼ 📁 com.uplus.eureka.config  
    > 📄 DataBaseConfiguration.java  
    > 📄 SwaggerConfiguration.java  
    > 📄 WebMvcConfiguration.java
```

2.4 lombok을 사용하는 경우 참조

pom.xml에 의존성 추가

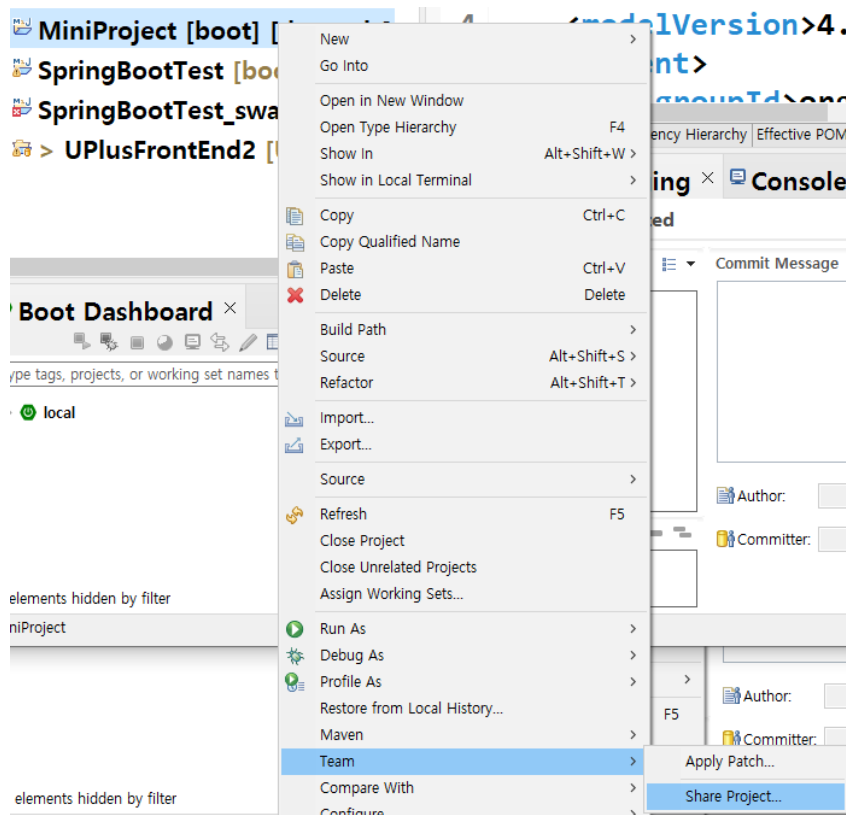
```
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->  
<dependency>  
    <groupId>org.projectlombok</groupId>  
    <artifactId>lombok</artifactId>  
    <version>1.18.36</version>  
    <scope>provided</scope>  
</dependency>
```

3. 프로젝트안에 .github 디렉토리와 .gitmessage.txt 파일을 추가한다.

디스크 (C:) > uplus > workspace > MiniProject >

이름	수정된 날짜	유형	크기
.github	2025-03-13 오전 9:33	파일 폴더	
.mvn	2025-03-13 오전 9:28	파일 폴더	
.settings	2025-03-13 오전 9:28	파일 폴더	
src	2025-03-13 오전 9:28	파일 폴더	
target	2025-03-13 오전 9:28	파일 폴더	
.classpath	2025-03-13 오전 9:28	CLASSPATH 파일	3KB
.gitattributes	2025-03-13 오전 9:28	텍스트 문서	1KB
.gitignore	2025-03-13 오전 9:35	Git Ignore 원본 파...	1KB
.gitmessage.txt	2025-03-11 오전 10:33	텍스트 문서	1KB
.project	2025-03-13 오전 9:28	PROJECT 파일	1KB
HELP.md	2025-03-13 오전 9:28	Markdown 원본 ...	2KB
mvnw	2025-03-13 오전 9:28	파일	11KB
mvnw.cmd	2025-03-13 오전 9:28	Windows 명령어 ...	7KB
pom.xml	2025-03-13 오전 9:28	Microsoft Edge H...	3KB

4. 프로젝트를 우 클릭 후 team>share project를 선택한다.



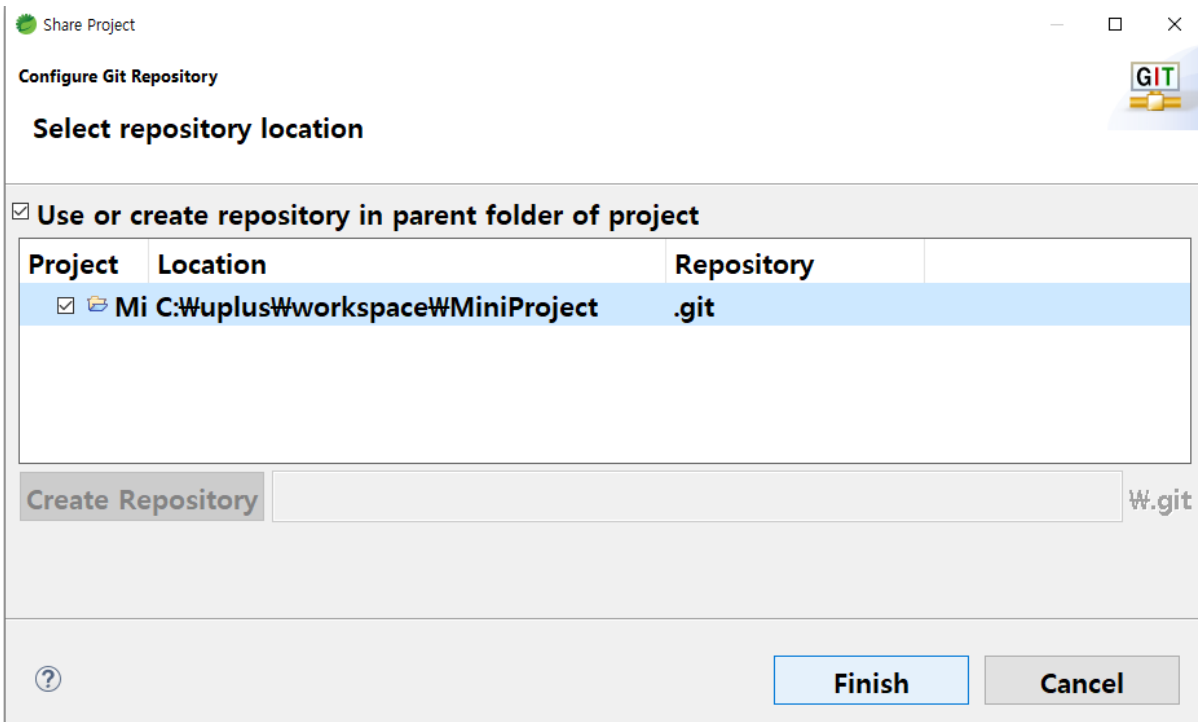
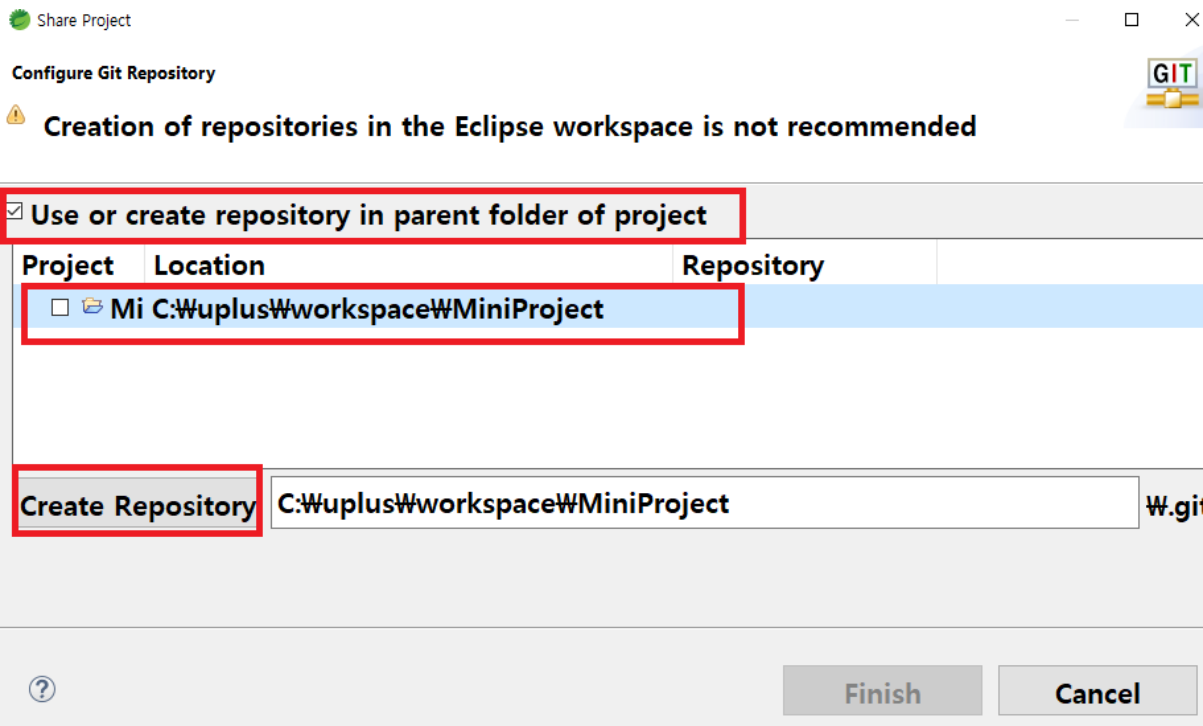
5. Repository 설정하기

5.1 Use or create repository~ 를 클릭한다.

5.2 프로젝트를 선택하면 Create Repository를 클릭할 수 있게 활성화 된다.

5.3 활성화된 Create Repository를 클릭하면 finish 버튼이 활성화된다.

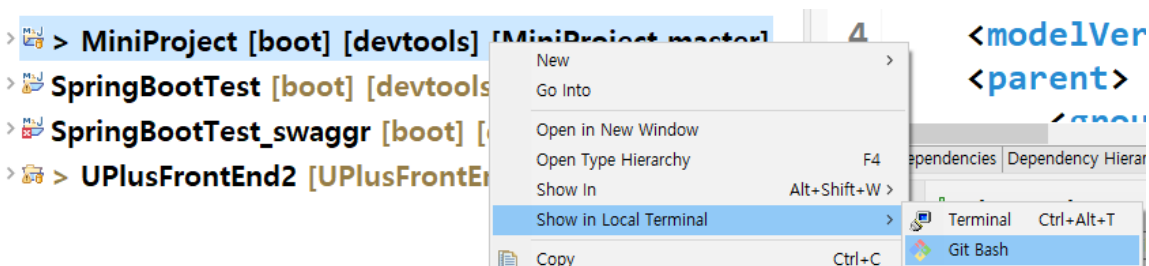
5.4 finish 버튼을 클릭한다.



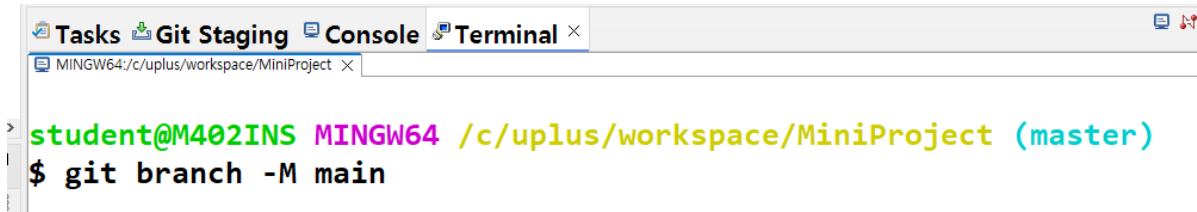
프로젝트에 Repository가 생성되었다.

> MiniProject [boot] [devtools] [MiniProject master]

6. 프로젝트를 우 클릭 > show in Local Terminal > Git bash를 선택한다.

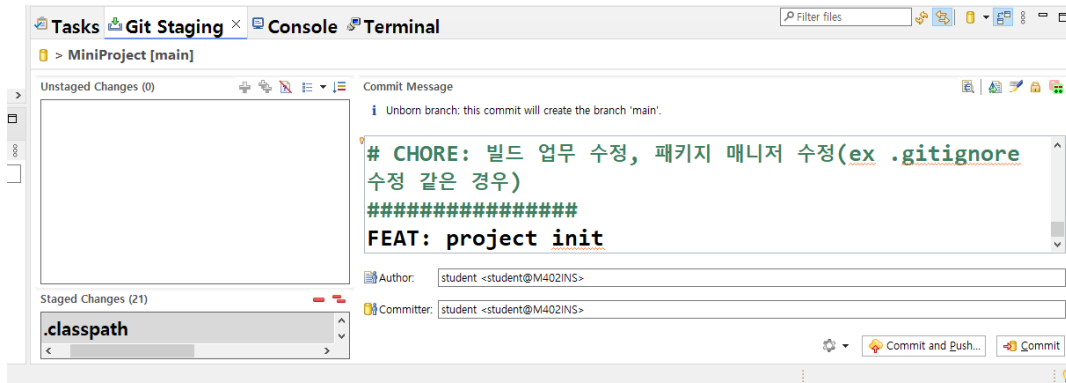


Git branch – M main

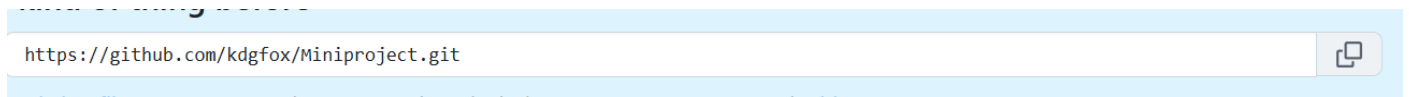


```
student@M402INS MINGW64 /c/uplus/workspace/MiniProject (master)
$ git branch -M main
```

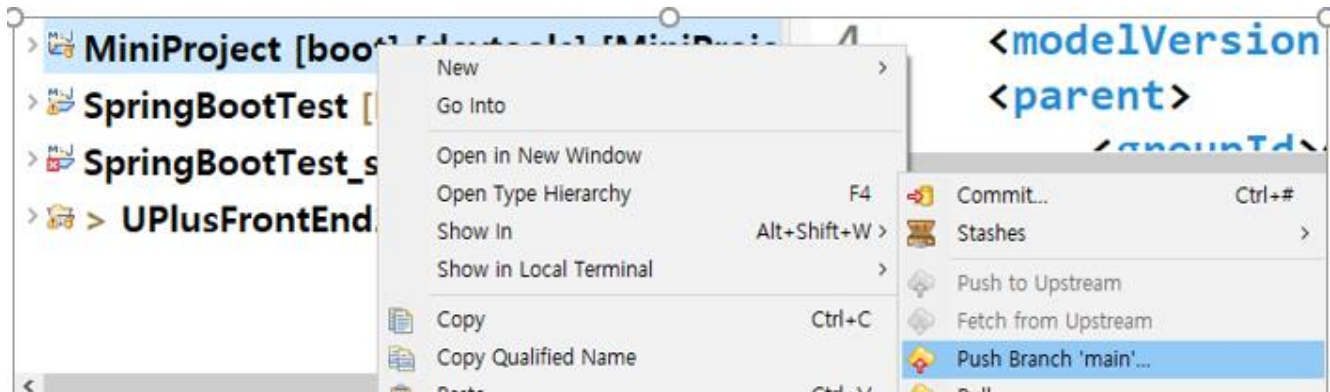
sts에서 team>commit을 선택해 ++ 로 모두 staging 하고 commit 메시지 입력 후 commit 버튼 클릭



Github 의 원격 주소를 복사한다.



Temp>push Branch main을 선택한다.



계정명과 token을 입력후 previw> push 한다.

Push Branch master

Destination Git Repository

Enter the location of the destination repository.

Remote name: origin

Location

URI: /kdgfox/miniproject.git Local Folder...

Host: github.com

Repository path: /kdgfox/miniproject.git

Connection

Protocol: https

Port:

Authentication

User: kdgfox

Password:

☒ Store in Secure Store

?

< Back

Preview >

Push

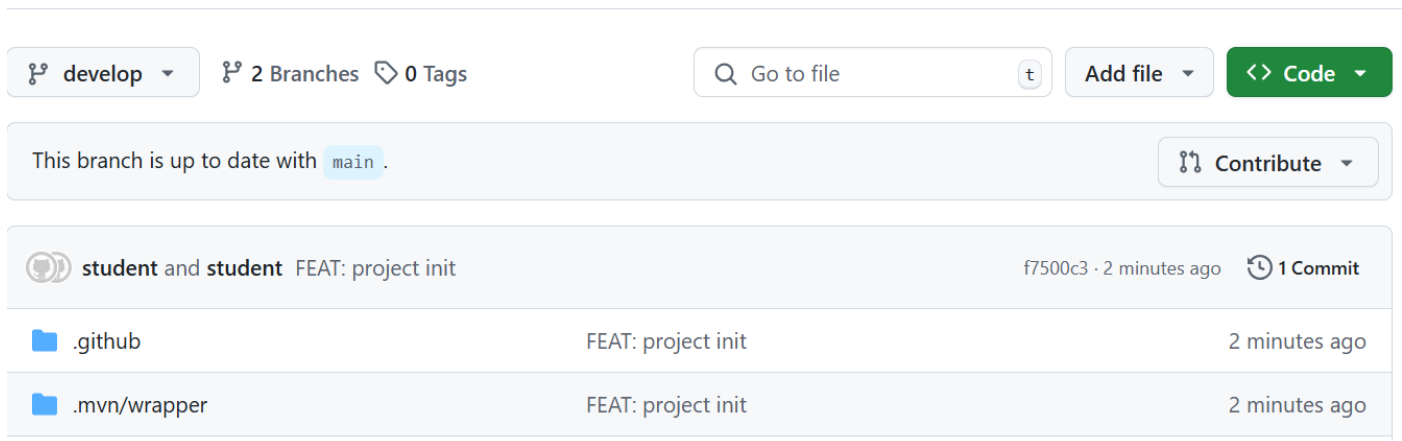
Cancel

원격 레포지토리를 새로고침하면 push된 것을 확인할 수 있다.

7. Sts에 열어온 git bash에서 develop branch 생성해서 push 한다.

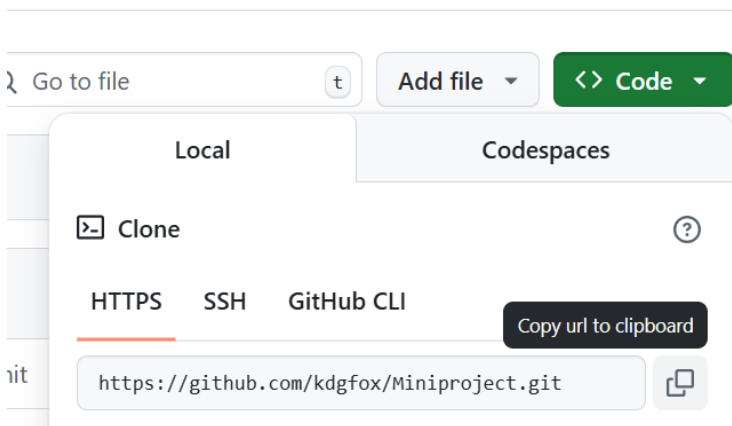
```
student@M402INS MINGW64 /c/uplus/workspace/MiniProject (main)
$ git branch -M develop

student@M402INS MINGW64 /c/uplus/workspace/MiniProject (develop)
$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
```

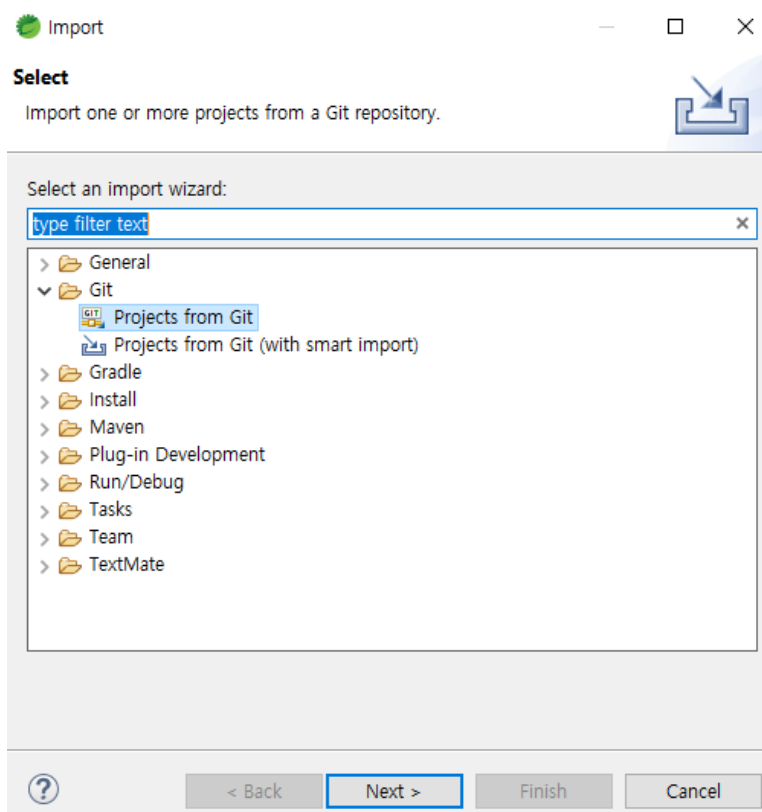


팀원이 공유 프로젝트 sts에 import하기

1. 공유 프로젝트의 주소를 복사한다.



2. Import project를 선택 후 Projects from Git을 선택하고 next 버튼을 클릭한다.



Import Projects from Git

Source Git Repository

Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

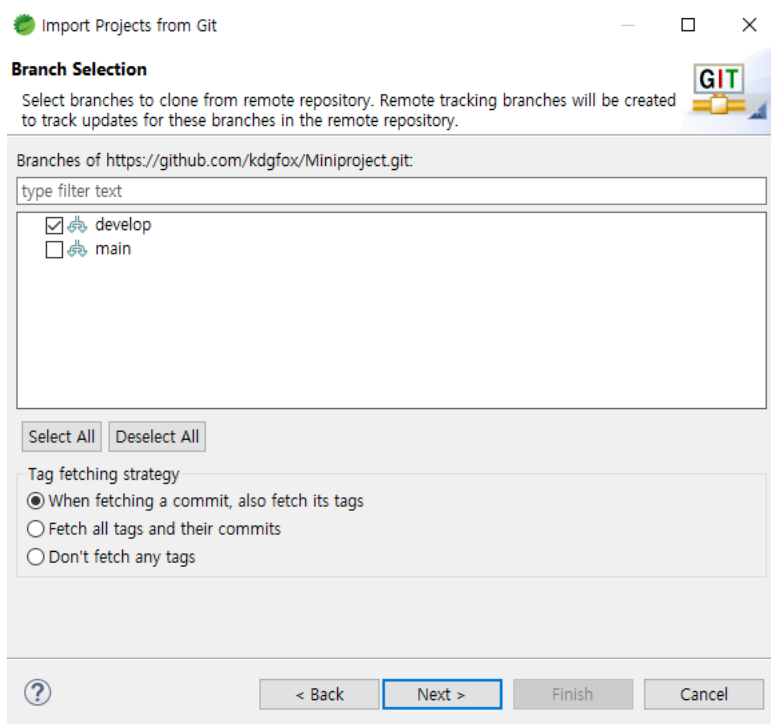
Authentication

User:

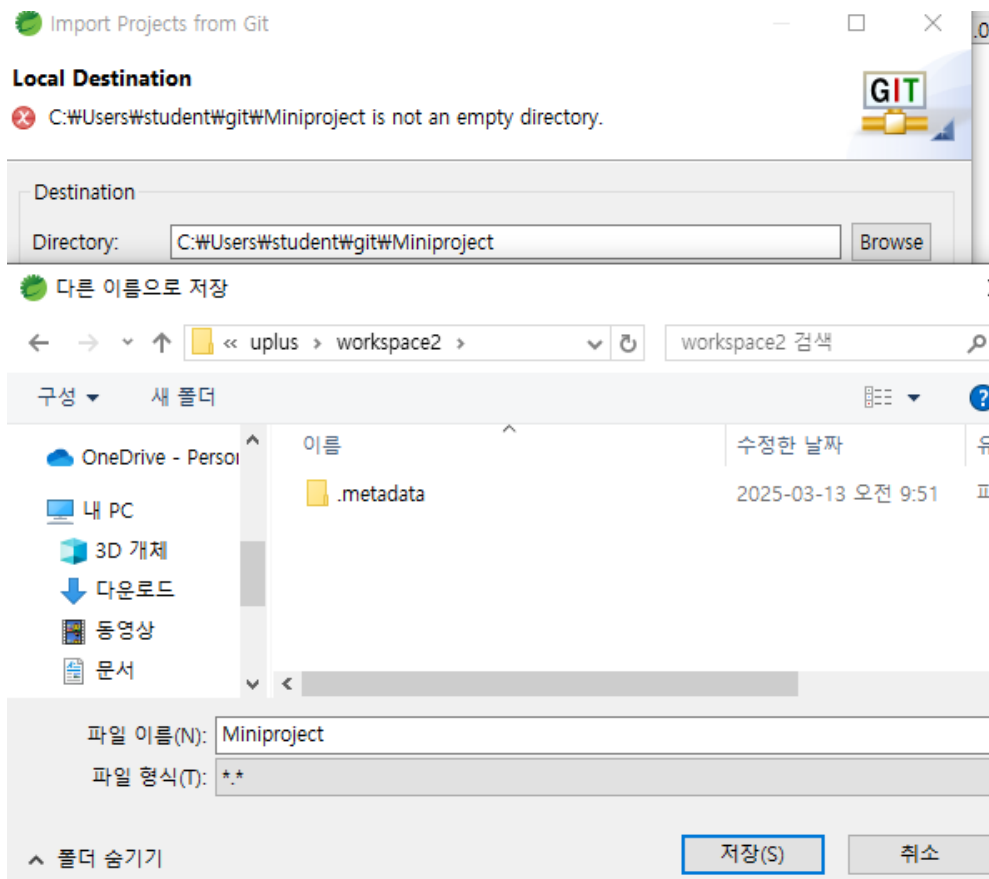
Password:

☒ Store in Secure Store

팀원은 Develop만 선택하고 next한다.



다운 받을 경로를 설정한다.



Import Projects from Git

Local Destination
Configure the local storage location for Miniproject.

Destination

Directory:

Initial branch:

☐ Clone submodules

Configuration

Remote name:

Next 선택한다.

Import existing Eclipse projects를 선택후 next

Cloning from <https://github.com/kdgfox/Miniproject.git>

Select a wizard to use for importing projects
Depending on the wizard, you may select a directory to determine the wizard's scope

Wizard for project import

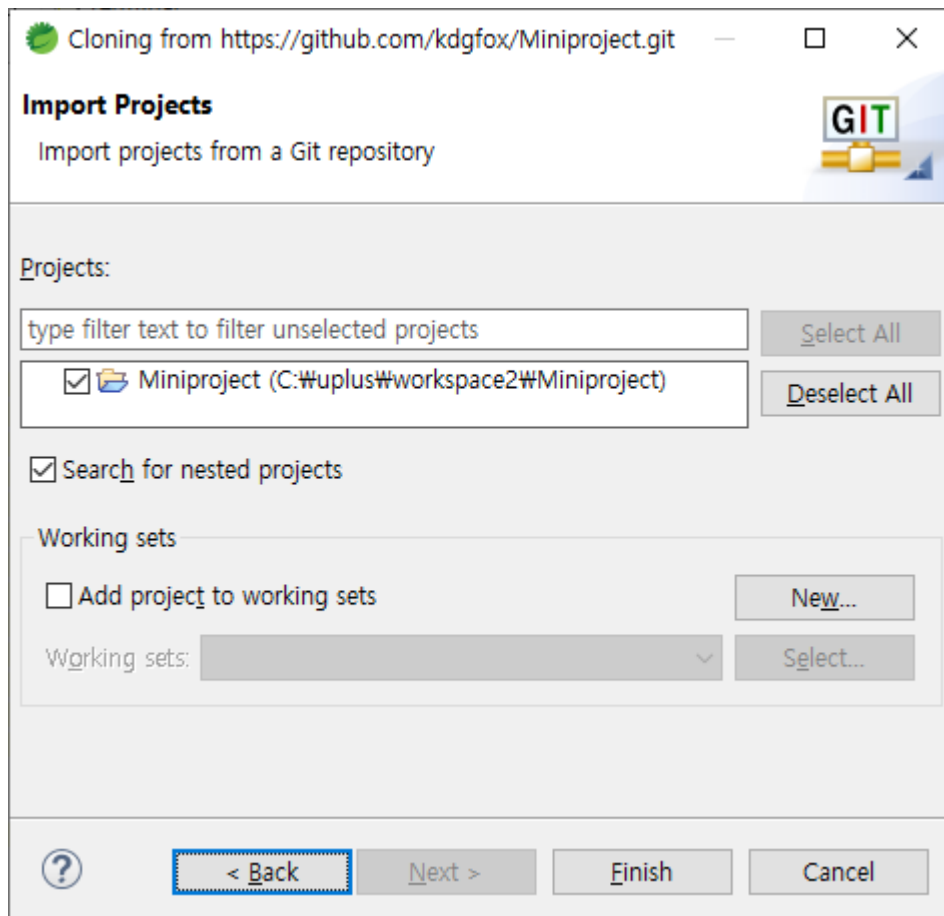
☒ Import existing Eclipse projects

☐ Import using the New Project wizard

☐ Import as general project

Working Tree - C:\uplus\workspace2\Miniproject

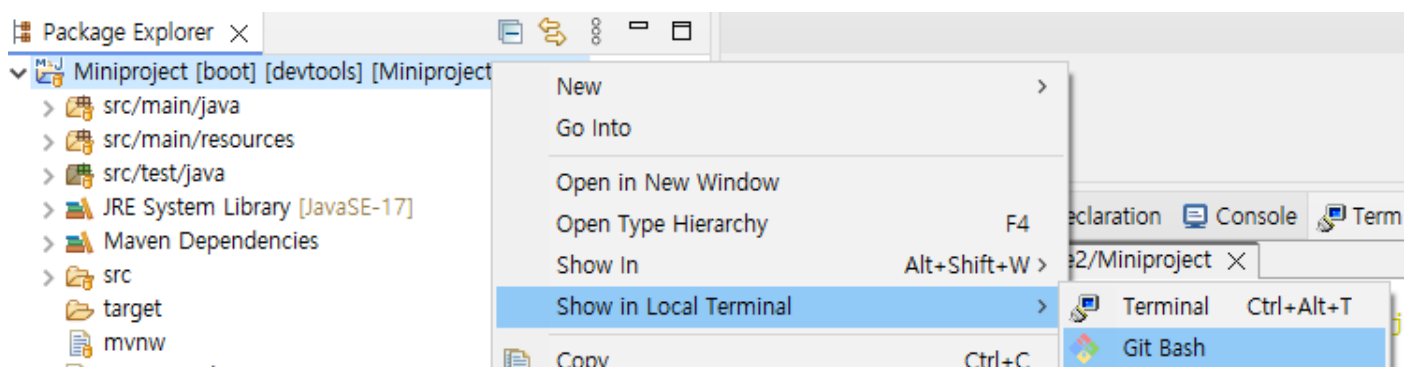
Finish 버튼을 클릭한다.



프로젝트가 import됐다.

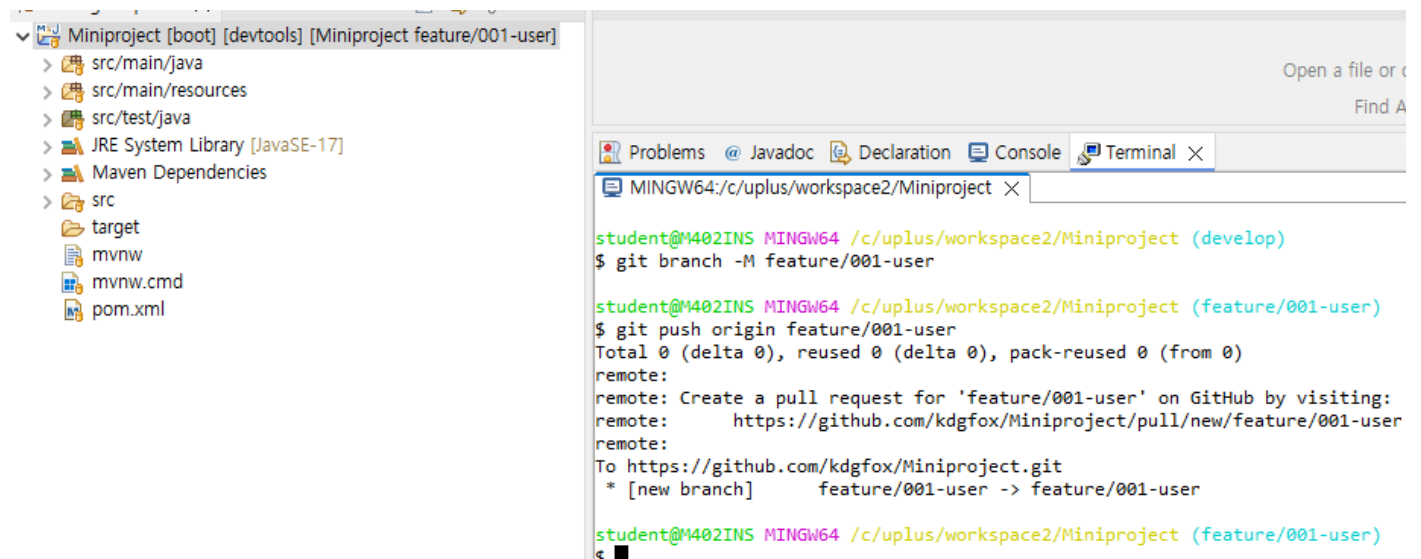


Git bash를 연다



feature/~ branch를 생성해서 push한다. Push 해야 프로젝트가 feature branch로 변경된다.

변경된 feature branch에서 작업한다.



```
student@M402INS MINGW64 /c/uplus/workspace2/Miniproject (develop)
$ git branch -M feature/001-user

student@M402INS MINGW64 /c/uplus/workspace2/Miniproject (feature/001-user)
$ git push origin feature/001-user
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature/001-user' on GitHub by visiting:
remote:   https://github.com/kdgfox/Miniproject/pull/new/feature/001-user
remote:
To https://github.com/kdgfox/Miniproject.git
 * [new branch]      feature/001-user -> feature/001-user

student@M402INS MINGW64 /c/uplus/workspace2/Miniproject (feature/001-user)
< █
```