

1 Git 을 이용한 버전 관리

1 Git이란

트렁크, 브랜치 개념을 지닌 SVN은 중앙 집중형 형상 관리 기법을 사용하는 소프트웨어인 반면, Git은 로컬 저장소(Local Repository)와 원격 저장소(Remote Repository)로 나뉘어 별도 브랜치를 생성 후 merge하는 분산 소스형 형상 관리 기법을 통해 버전 관리를 수행하는 소프트웨어이다.

또한 원본 소스를 저장할 공유 Server가 필요한 SVN과는 다르게 Github, Bitbucket이라는 소스를 저장할 수 있는 원격 저장소를 제공하는 웹 사이트가 있어, 팀 프로젝트 수행 시 비용이 적게 드는 장점이 있다.

Git에서는 원격저장소의 master를 최대한 건드리지 않고, **각자 브랜치를 생성하여 작업** 후 원본(master / origin)에 merge하는 방식을 쓰기 때문에, SVN 방식과 헷갈리지 않도록 하자.

2 Git 설치하기

- Git 직접 설치

공식 사이트인 <https://git-scm.com/> 에서 오른쪽 화면에 보이는 'Downloads for Windows' 을 클릭하여 다운로드 페이지로 이동한다.



- ※ 현재 접속중인 컴퓨터의 운영체제에 따라 자동으로 <Downloads for Linux>, <Downloads for Mac> 등으로 달라진다.
- ※ Git 에 대해 더 알고 싶다면 <https://git-scm.com/book/ko/v2> 사이트를 참조하면 된다.

아래쪽에 자동으로 자신의 버전과 맞는 최신 버전이 뜬다. 다른 버전을 다운받고

싶으면 취소를 누르고 원하는 버전을 직접 다운로드 받으면 된다.



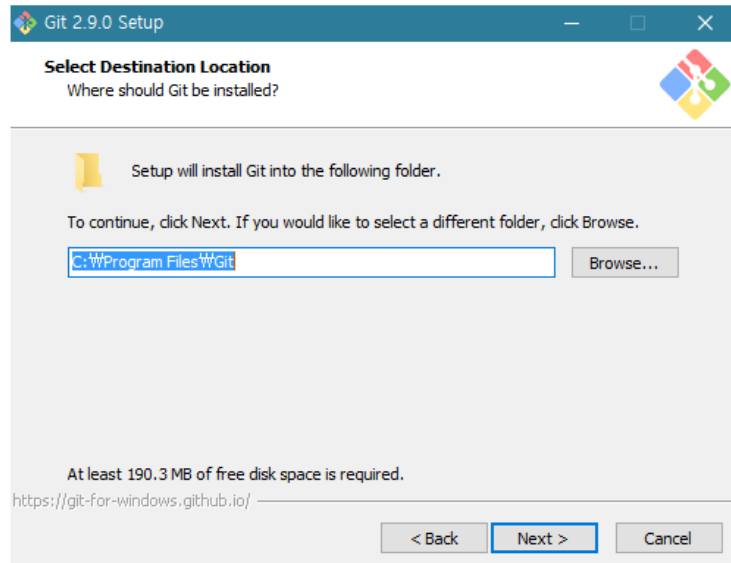
※ Git은 안정성이 검증되지 않았기 때문에 실제 다운로드 페이지에서 표기된 최신 버전보다 더 옛날 버전을 사용한다.

1. 윈도우에서 설치

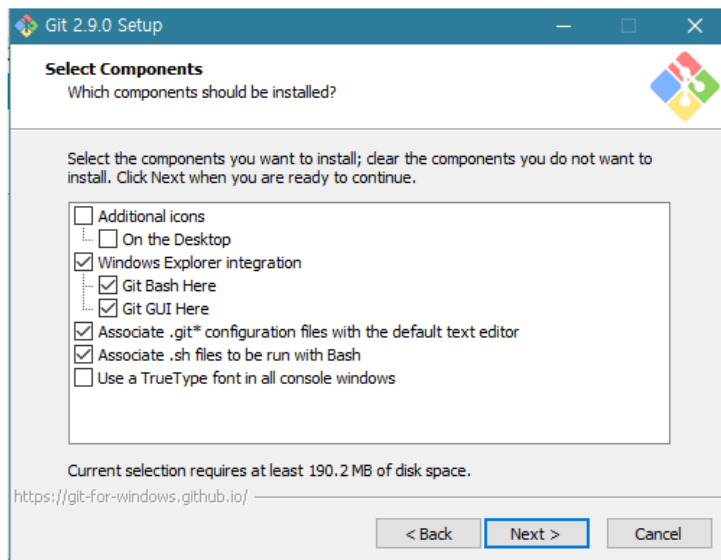
- <Next> 클릭



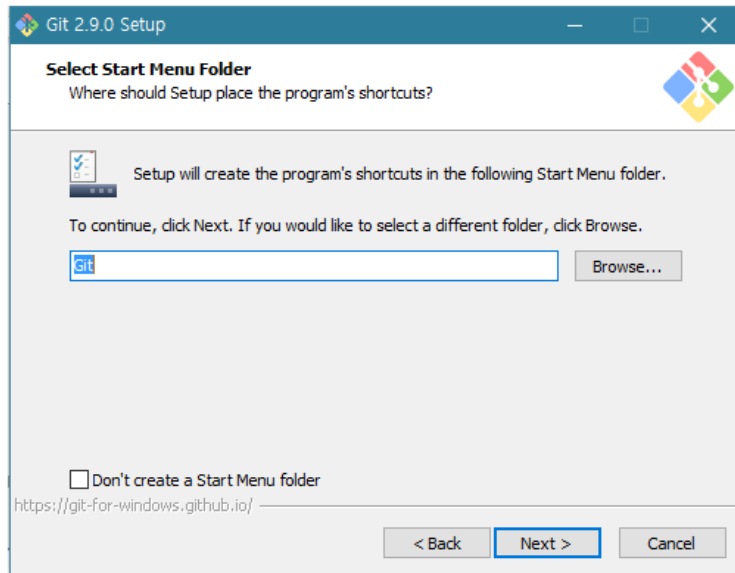
- <Next> 클릭



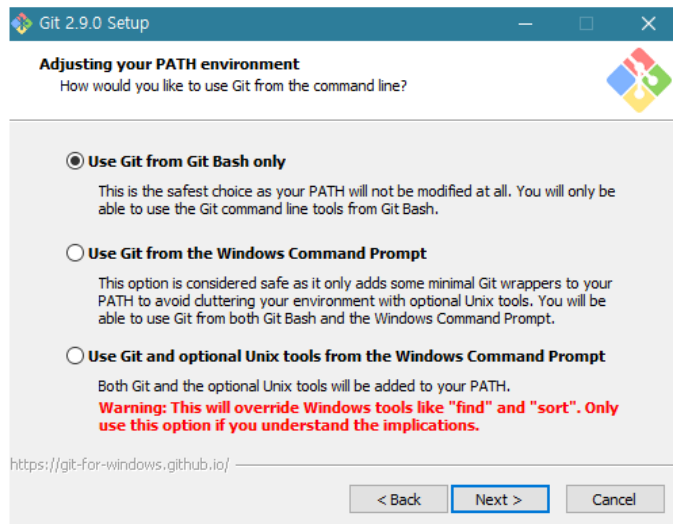
- 추가로 추가할 옵션이 있다면 선택하고 Next 클릭



- <Next> 클릭



- 환경 변수를 맨 위로 체크하고 <Next> 클릭

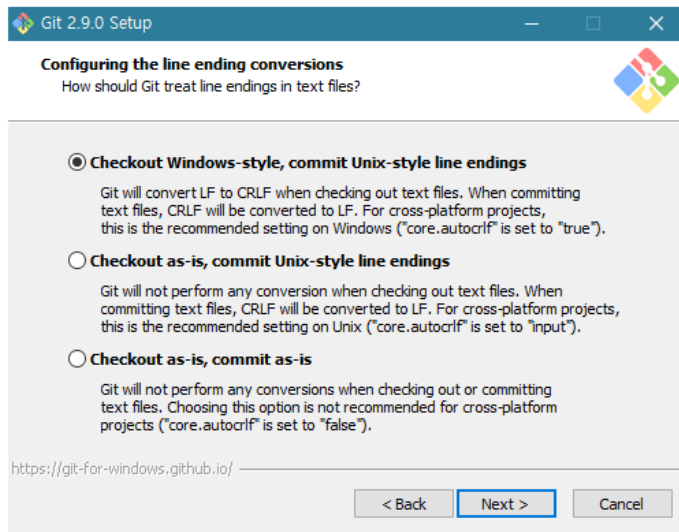


※ Use Git from Git Bash only : 환경 변수를 변경하지 않고 Git bash 커맨드 라인 도구에서만 실행한다.

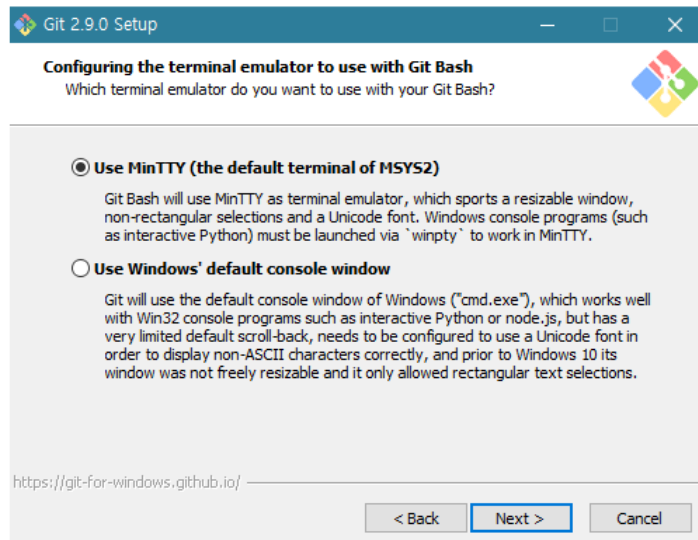
※ Use Git from Windows Command Prompt : 윈도우 커맨드 프롬프트에서 Git을 실행할 수 있는 최소한의 내용을 환경 변수에 추가해 설치한다.

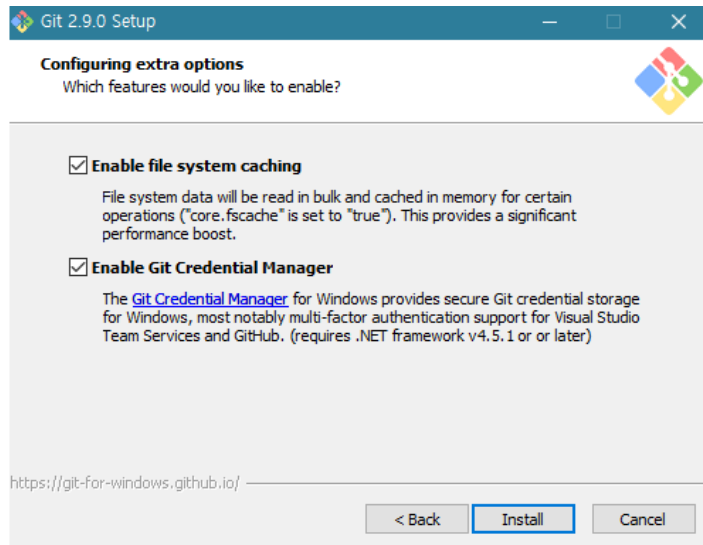
※ Use Git from optional Unix tools from the Windows Command Prompt : Git과 부수적인 UNIX 도구들을 모두 윈도우 환경 변수에 추가한다.

- Git에서 커밋할 때 작성하는 라인 끝을 어떻게 처리할 지 결정한다. 기본으로 선택된 첫 번째 옵션으로 두고 <Next> 클릭

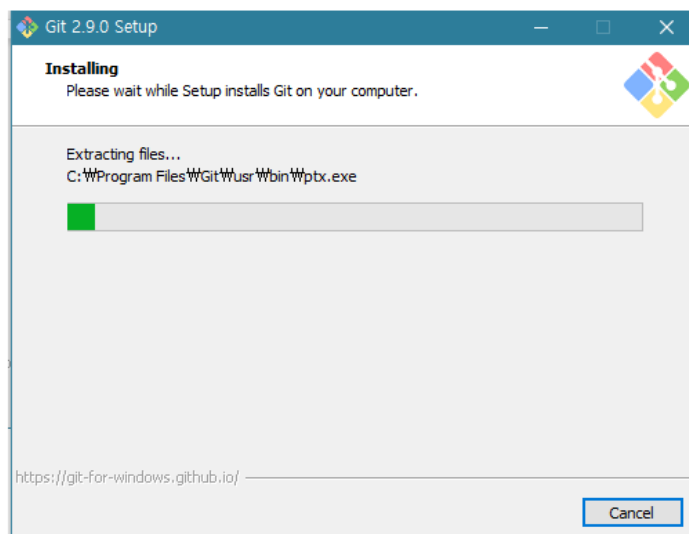


- <Next> 클릭



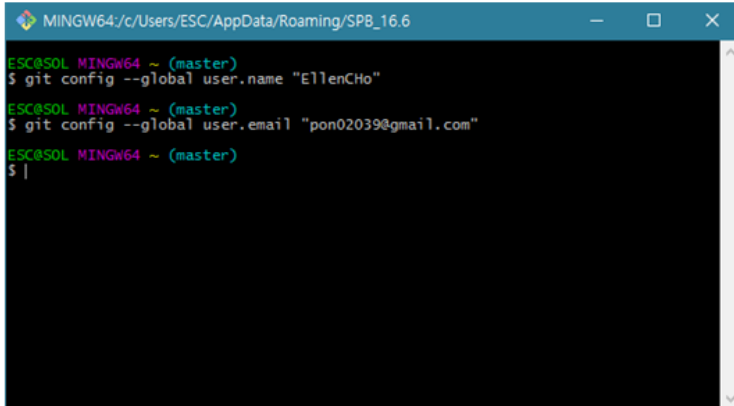


- 설치



- 시작메뉴 -> Git -> Git Bash 를 실행하여 사용자 이름과 이메일을 설정한다.

```
git config --global user.name "사용자 이름"
git config --global user.email "이메일"
```



```
ESC@SOL MINGW64 ~ (master)
$ git config --global user.name "EllenCho"
ESC@SOL MINGW64 ~ (master)
$ git config --global user.email "pon02039@gmail.com"
ESC@SOL MINGW64 ~ (master)
$ |
```

- 설정 완료

- Github 사용하여 Git 저장소 만들기

GitHub는 원격 저장소를 제공하며 여러가지 프로젝트 진행을 원활하게 하는 도구를 함께 제공한다.

1. 회원 가입

- 공식 사이트인 <https://github.com/> 에 접속해서 오른쪽에 사용자 아이디, 이메일, 비밀번호를 치고 <Sign up for GitHub>를 클릭한다.



- 지불하는 비용에 따라 비공개 저장소를 사용할 수 있다. 무료로 GitHub를 이용한다면 모든 저장소는 공개 저장소가 된다. 맨 아래 기본으로 선택된 'Free'항목으로 그대로 두고 <Finish sign up> 버튼을 누른다.

※ 'Help me set up an organization next' 체크 박스는 많은 사람이 협업할 때 팀을 만들어서 활동하도록 설정하겠다는 의미이다.

- 회원 가입할 때 입력했던 이메일을 확인해서 GitHub 가입을 인증한다.

- 회원 가입 완료

2. 원격 저장소 생성

- 포크 : 다른 사람의 저장소를 내 저장소로 복사하는 기능
- 풀 리퀘스트 : 포크한 저장소를 수정해 다시 원본 저장소에 병합해달라는 요청을 보내 사용자 사이의 상호작용을 일으키는 기능
- 이슈 : 저장소 안에서 사용자들 사이의 문제를 논의하는 기능
- 위키 : 저장소와 관련된 체계적인 기록을 남기는 기능

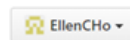
- 오른쪽 아래에 있는 <New repository>를 클릭해서 새 원격 저장소를 생성한다.

- 아래 항목을 참고해서 빈칸을 채운다.
- **Owner** : 사용자의 아이디가 표시된다. 협업 환경에서는 다른 사용자의 아이디를 지정 할 수 있다.
- **Repository name** : 새로 생성할 원격 저장소의 이름을 입력한다. 가능하면 로컬 환경에서 작업할 Git 프로젝트 디렉터리 이름과 같게 하는 것이 좋다.
- **Description** : 꼭 작성할 필요는 없는 항목이지만 생성한 원격 저장소가 어떤 역할을 하는 지를 간단하게 적어두면 원격 저장소가 많아졌을 때 구분하기 쉽다.
- **Public/Private** : 원격 저장소의 공개 여부를 선택하는 옵션이다. 무료 사용자는 **Public**만 사용 가능하다.
- **Initialize this repository with a README** : 기본적으로는 체크 표시를 해준다. 체크해주면 GitHub에서 생성한 원격 저장소를 바로 로컬 저장소에 복사해서 가져올 수 있다. 또한 '저장소 이름'과 'Description' 항목의 내용을 담은 README.md 파일을 생성한다.
- **Add .gitignore** : 원격 저장소에 포함하지 않을 파일들의 목록을 만들 때 사용한다. 지금 당장은 사용할 필요가 없으니 'none' 상태로 둔다.
- **Add a license** : 원격 저장소에 저장할 프로젝트가 어떤 라이선스에 속할지를 선택한다. 지금 당장은 사용할 필요가 없으니 'none' 상태로 둔다.

Create a new repository

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

Owner



Repository name

Great repository names are short and memorable. Need inspiration? How about **turbo-umbrella**.

Description (optional)



Public

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



Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

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

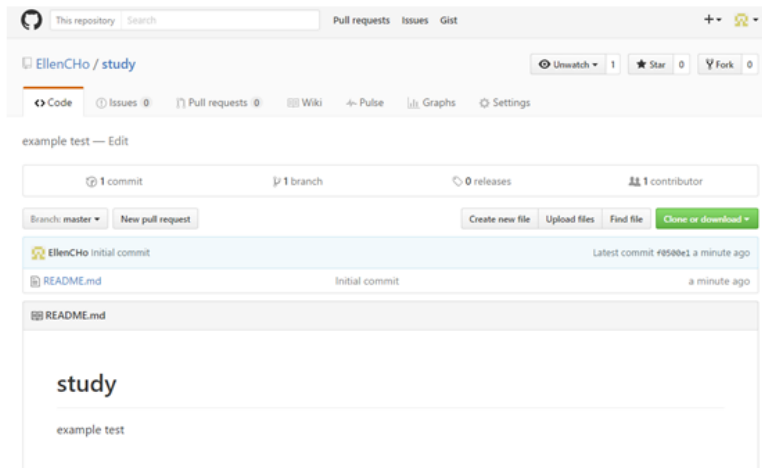
Add .gitignore: **None**

Add a license: **None**



Create repository

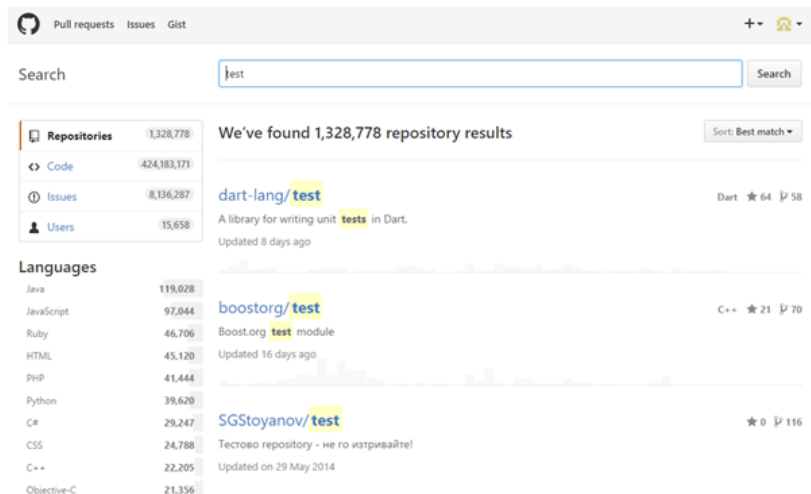
- <Create repository>를 클릭하면 아래와 같은 원격저장소가 생성된다.



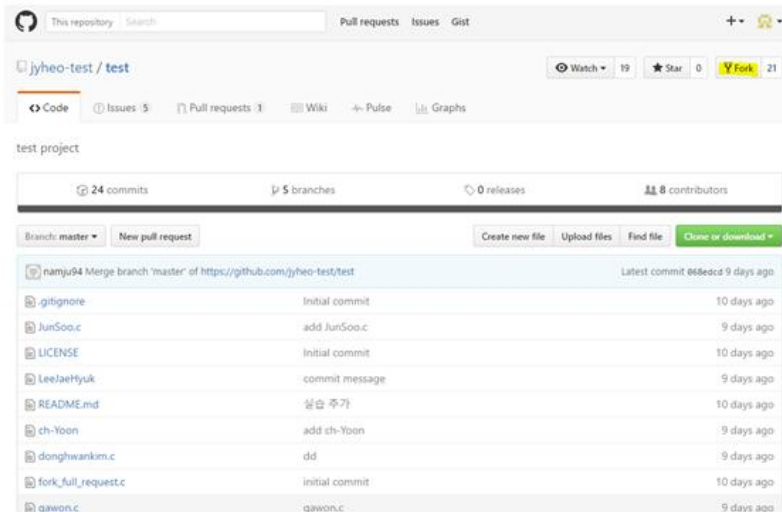
3. 포크

다른 사람의 원격 저장소를 내 계정으로 복사하는 방법을 포크라고 한다. 만약 포크하지 않는다면 내 것이 아닌 저장소 다 시말하면 쓰기 권한이 없는 원격 저장소를 사용하는 것이므로 자유롭게 파일을 생성하거나 수정하여 원격 저장소에 반영 하는 것이 불가능하다.

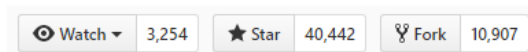
- 원격 저장소 검색(ex. test)



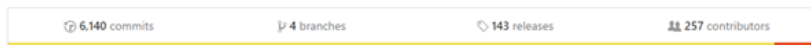
- 오른쪽에 있는 <Fork>를 클릭하면 내 원격 저장소에 저장됨



4. GitHub 원격 저장소의 구조

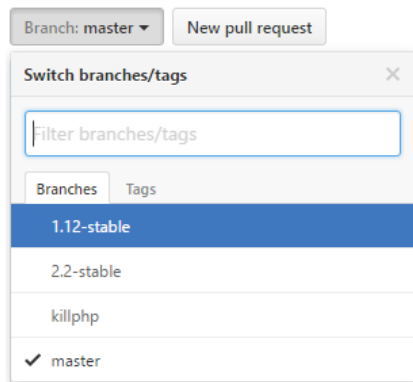


- **Watch** : 해당 버튼을 클릭하면 원격 저장소의 활동 내역을 사용자에게 알려준다. 댓글이나 이슈 등에서 언급될 때만 알려주는 **Not Watching**, 모든 활동 내역을 알려주는 **Watching**, 모든 알림을 무시하는 **Ignoring**을 선택할 수 있다. 오른쪽 숫자는 현재 활동 내역을 보고 있는 사람의 수이다.
- **Star** : 해당 원격 저장소에 관심이 있을 때 클릭하면 된다. 오른쪽 숫자는 관심이 있는 사람의 수를 나타낸다.
- **Fork** : 해당 버튼을 클릭하면 원격 저장소를 포크한다. 오른쪽은 포크한 사람의 수를 나타낸다.

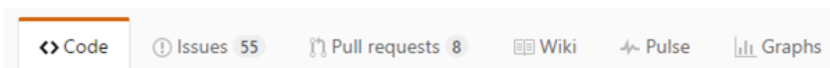


- **Description** : 원격 저장소를 설명하는 메시지가 나타난다.
- **Commits** : 원격 저장소의 총 커밋 수를 나타낸다.
- **Branches** : 원격 저장소의 브랜치 수를 나타낸다.
- **releases** : 원격 저장소의 태그 수를 나타낸다. 주로 특정 버전에 표식을 주고 싶을 때 사용한다. 이 표식을 통해서 특정 버전을 다운로드할 수 있다.
- **Contributor** : 원격 저장소에 커밋 혹은 풀 리퀘스트가 받아들여진 사용자 수이다. 이 저장소가 오픈 소스라면 이 오픈 소스에 공헌한 사람 수라고 생각해도 된다.

※ 커밋은 뒤에 나오지만 저장할 때마다 끼워넣는 책갈피라고 생각하면 쉽다. 지금 저장한 버전이 무엇을 수정했는지, 무슨 내용을 담고 있는지 설명해주는 지표이다.



- **Current branch** : 원하는 브랜치를 선택하는 기능
- **Current branch** : 원하는 브랜치를 선택하는 기능

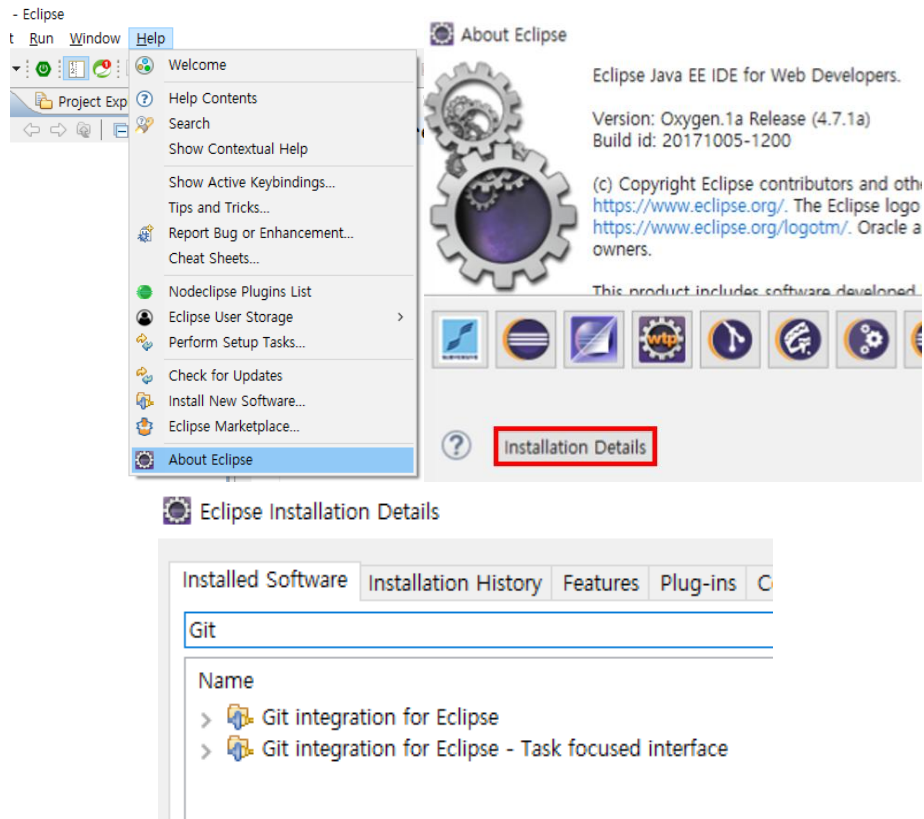


- **Code** : 해당 원격 저장소의 루트 디렉토리로 이동한다. 어떤 경로에 있더라도 루트 디렉토리로 이동한다.
- **Issues** : 해당 원격 저장소의 주요 이슈 사항을 기재한 후 관리합니다. 게시판 형태이며 댓글 형태로 토론이 이뤄지기도 한다. 보통은 문제점이나 개선점을 얘기한다.
- **Pull Requests** : 풀 리퀘스트 전체 목록을 모아서 보여준다. Issues와 마찬가지로 목록마다 댓글 형태로 토론할 수 있다. 보통 오른쪽에 있는 숫자는 현재 요청이 온 풀 리퀘스트를 받아들일 것인지에 대한 논의가 몇개인지 알려주는 기능이다.
- **Wiki** : 공유할 정보나 개발 문서, 참고 자료 등을 작성하기 위한 기능입니다. 마크 다운과 위키위키 문법을 사용한다.
- **Pulse** : 해당 원격 저장소의 최근 변경 내역을 확인할 수 있다. 최대 한 달까지의 변경 내역을 확인할 수 있으며 풀 리퀘스트 몇 개중에 몇 개가 받아들여졌나, 이슈는 몇 개 있고 몇 개가 해결되었나, 해당 풀 리퀘스트와 이슈에 관련된 활동을 볼 수 있다.
- **Graphs** : 공헌자의 공헌 내역, 커밋 수 등 해당 저장소의 활동 내역을 그래프화해서 보여준다.

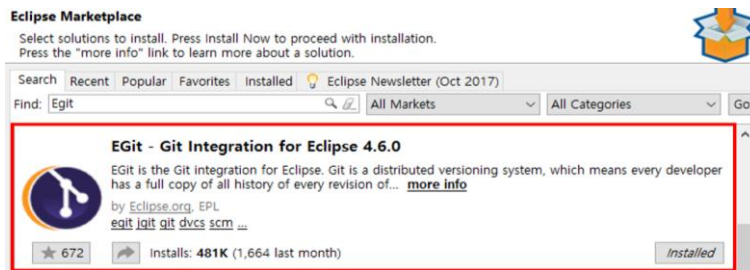
3 프로젝트에 Git 적용하기

사용의 편의에 따라 본 문서에서는 Github를 사용하는 예시를 주로 다루도록 하겠다.
먼저 이클립스에서 git과 관련된 플러그인이 설치되어 있는지 확인하자.

이클립스 상단의 Help > About Eclipse > Installation Details 에서 Git으로 검색 후
하단의 항목이 뜨면 별도의 git 플러그인 설치 필요 없다.

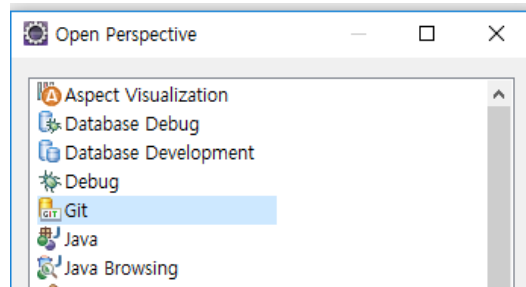
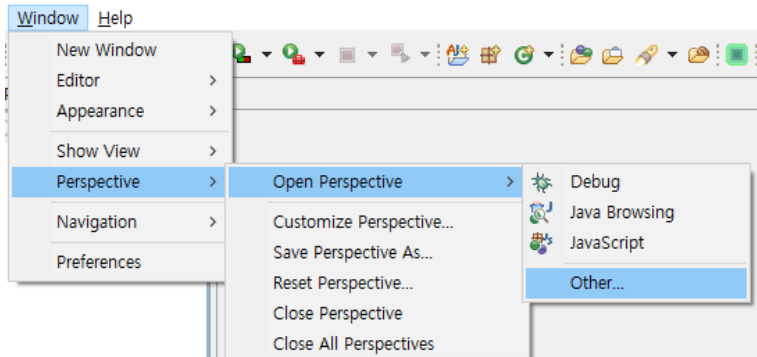


만약 상단의 Git 플러그인 설치가 안되어 있을 경우, Help > Eclipse Marketplace를 통해 EGit을 받도록 하자.



이제 Github를 통해 생성한 저장소를 연동하도록 하겠다. 여기서는 임시로 test라는 저장소를 생성했다고 가정한다. 만약 **ignore할 설정이 필요하다면, 저장소 생성 시 설정해야 함**을 참고하도록 하자.

먼저 상단의 Windows > perspective > Open Perspective > Other 로 접속하여 Github와 연동 설정을 추가한다.

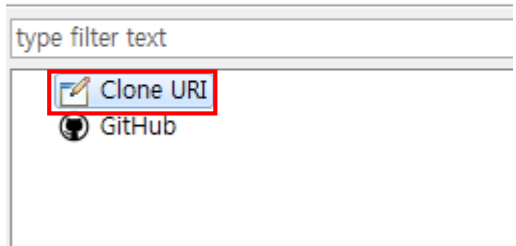


Select Repository Source

Select a location of Git Repositories

Select one of the following to add a repository to t

-  [Add an existing local Git repository](#)
-  [Clone a Git repository](#)
-  [Create a new local Git repository](#)



해당 창이 팝업되면 github에 만들어 둔 저장소의 주소를 복사해서 붙여 넣고, ID 와 Password 설정 후 Next를 누른다.

Source Git Repository

Enter the location of the source repository.



Location

URL: Local File...

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

☒ Store in Secure Store

Master 브랜치를 확인하고 next를 클릭한다.

Clone Git Repository

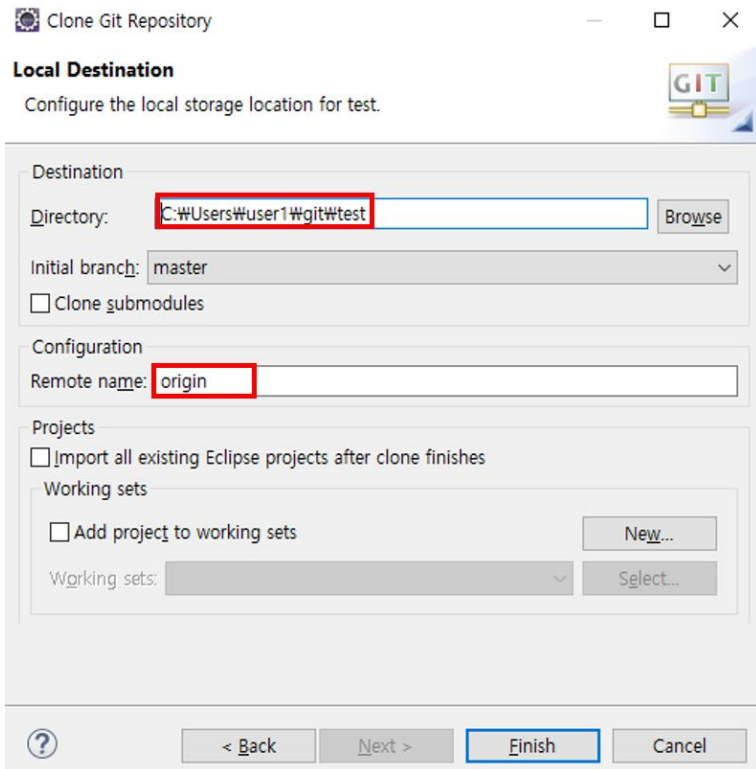
Branch Selection

Select branches to clone from remote repository. Remote tracking branches will be created to track updates for these branches in the remote repository.

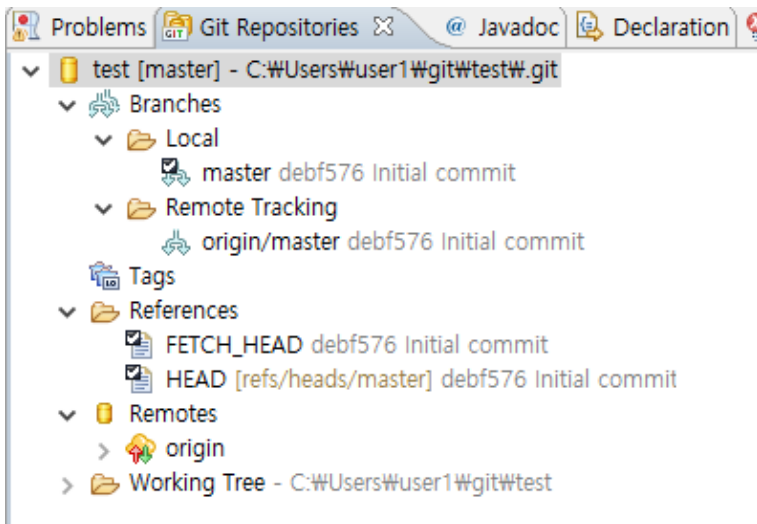
Branches of https://github.com/username/test:

type filter text

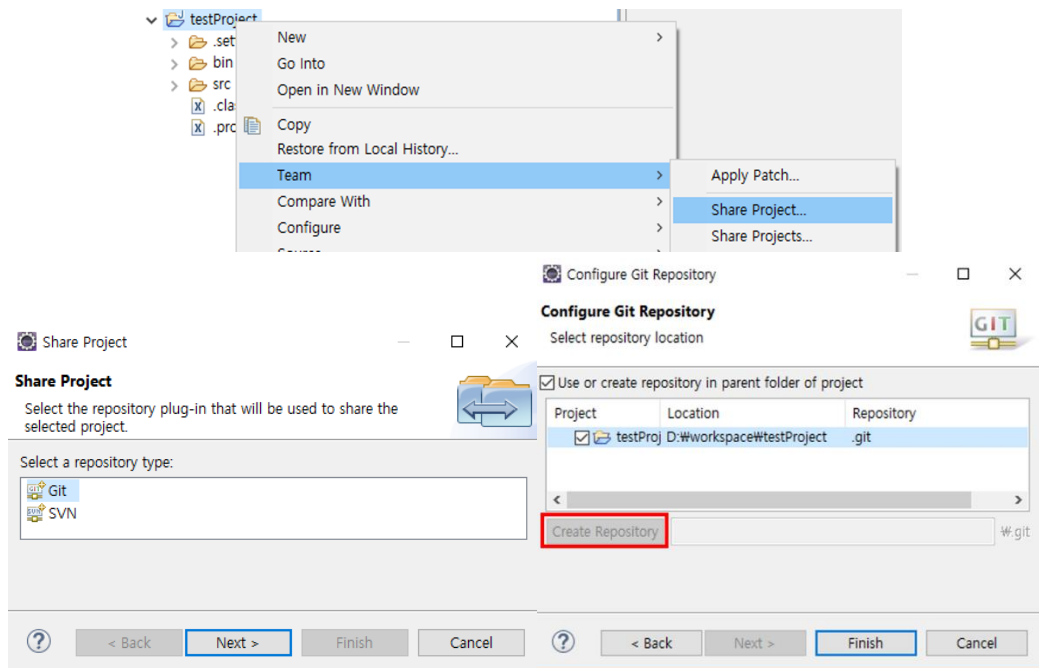
☒ master



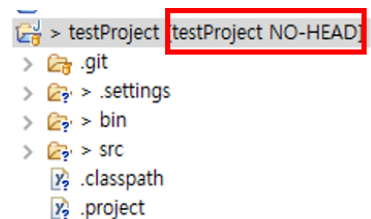
다음으로 로컬 저장소(본인 컴퓨터의 저장 경로)의 경로를 설정하고 원격 저장소의 호칭을 정하고 Finish를 누르면 Git Repositories 라는 views에 하단의 이미지와 같이 경로가 연결됨을 확인할 수 있다.

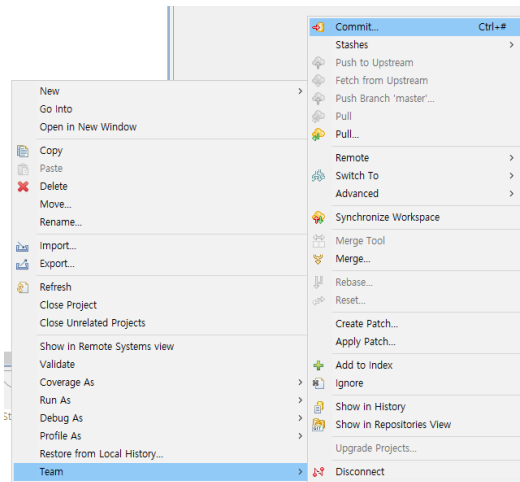


이제 해당 저장소에 프로젝트 파일을 업로드 해보자. 먼저 작업한 프로젝트 상단에 마우스 우클릭으로 Team > Share Project 에 접근, Git을 선택한 뒤 git 별도의 설정파일(.git)을 생성해야 하므로 git 용도의 별도 저장소를 Create Repository를 클릭한다.

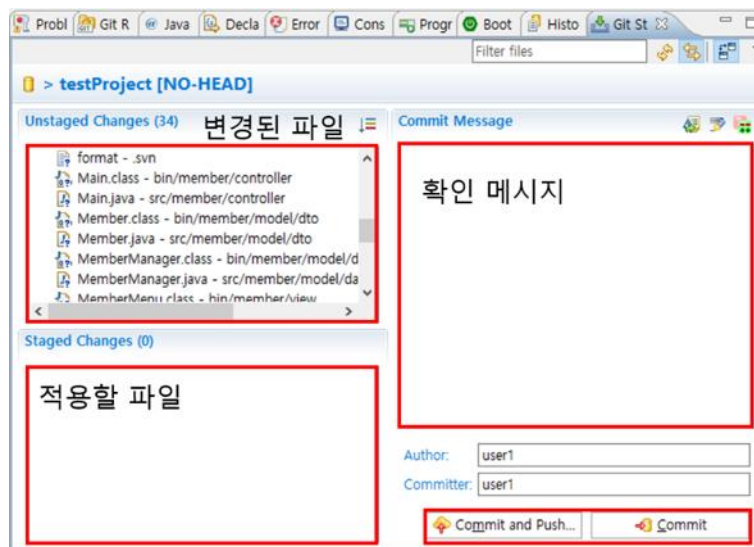


생성이 완료되면 좌측 프로젝트 명 옆에 tag가 붙으며 No-HEAD라고 표시되는데, 이는 아직 최초 commit을 하지 않아 발생한 것이므로, 우 클릭 후 Team > commit으로 소스를 저장해주도록 하자.





그럼 상단과 같은 창이 하나 발생하는데, 좌측 상단은 변경된 파일, 하단은 저장소에 적용할 파일, 우측 상단은 SVN과 동일한 commit 메시지를 적는 부분, 하단은 각각 저장 후 원격지의 저장소에 반영할 것인지, 단순 저장만 할 것인지를 나타내는 버튼들이 있다.



버전에 따라 체크박스 형식으로 파일 선택 창이 나타날 수 있는데, 선택하여 적용하는 방식은 동일하다. 이어서 저장 및 원격지 반영을 하기 위해 Commit and Push 버튼을 클릭하도록 하자. 그럼 목적지 저장소에 대해 설정하는 부분이 나오는데, 이 부분은 처음 github 연동한 것과 동일하게 uri 와 ID, Password를 적어주도록 한다.

Push Branch master

Destination Git Repository
Enter the location of the destination repository.

Remote name:

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

☒ Store in Secure Store

Next를 누르면 우리가 입력한 Commit 메시지와 함께, 반영할 브랜치가 표시되는데, 차 후 팀 프로젝트 작업 시 각자의 브랜치를 생성하여 그 이름을 해당 란에 적으면 된다.

Push Branch master

Push to branch in remote
Select a remote and the name the branch should have in the remote.

Source:

Destination:

Remote:

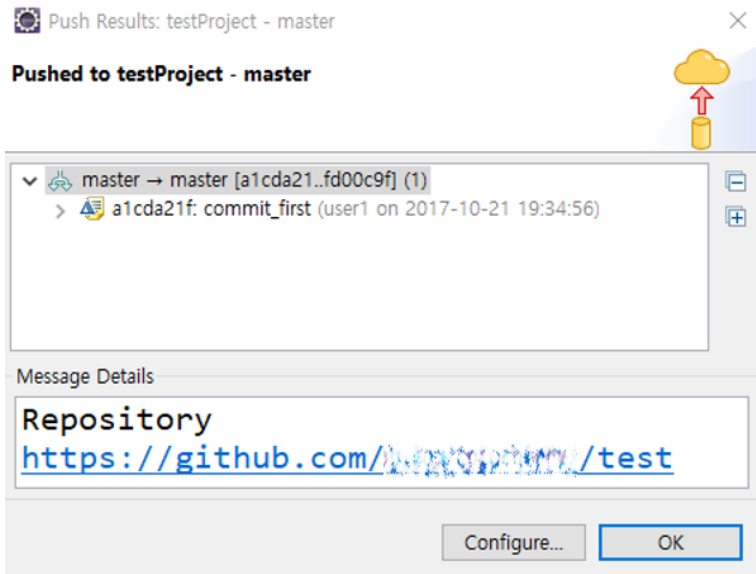
Branch:

☒ Configure upstream for push and pull

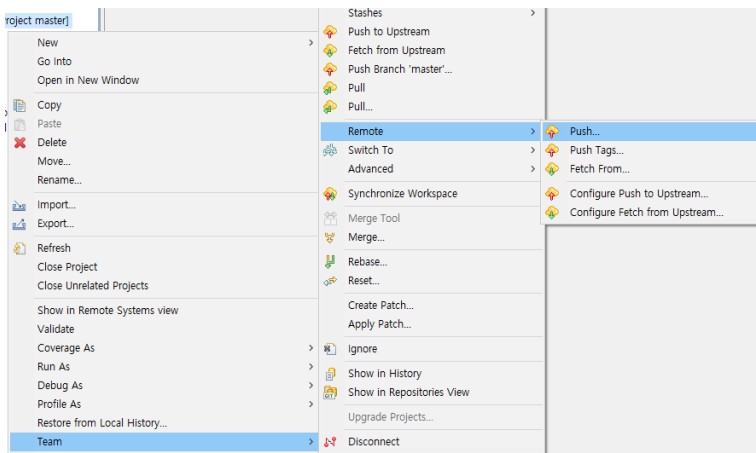
When pulling:

☐ Force overwrite branch in remote if it exists and has diverged

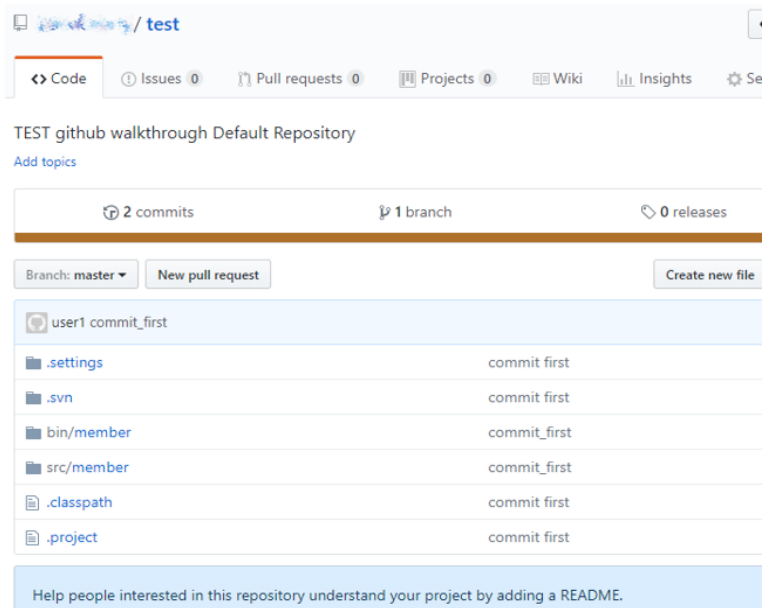
[Show advanced push dialog](#)



본 이미지와 같이 떴다면, 원격 저장소에 Push가 성공한 것이다. 만약 Reject non-forwarding 에러가 발생했다면, 연동과정에 에러가 발생하여 연동이 제대로 되지 않은 것이므로, Team > Remote > Push 를 통해 진행해보도록 하자. 절차는 commit 과정과 동일하다.

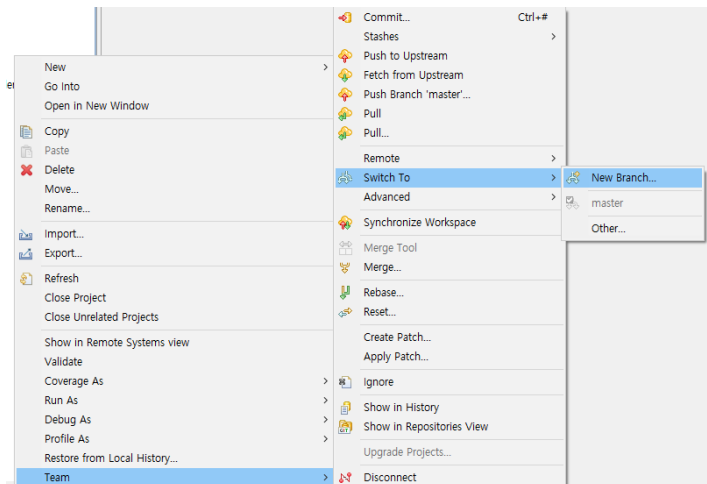


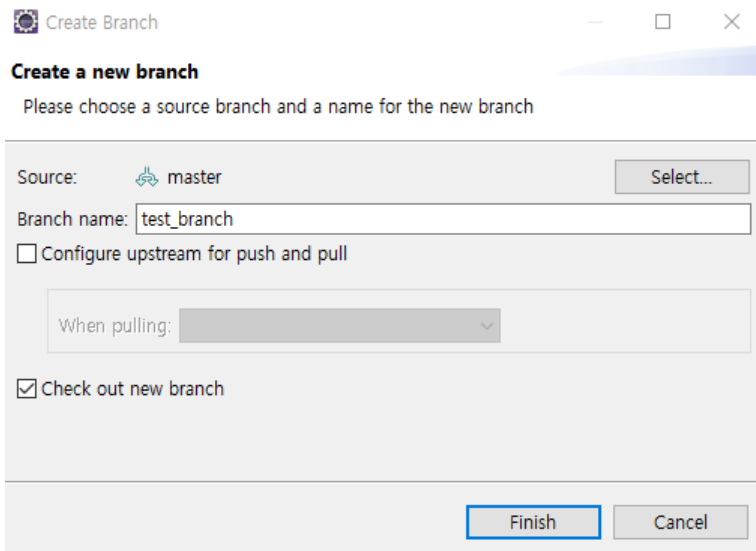
원격 저장소에 반영이 완료되면, github에서도 해당 커밋에 대한 정보를 확인할 수 있다.



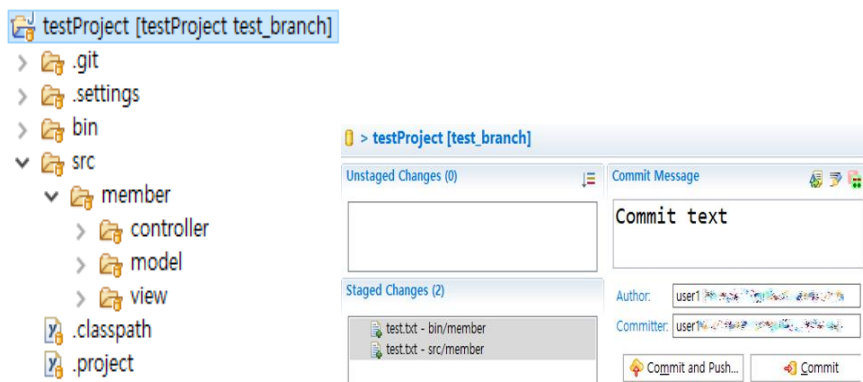
4 Git 소스코드 수정 및 Merge

보통 Git에서는 master에 직접 접근하여 commit 처리를 하지 않고, 별도의 브랜치를 생성하여 작업 후 synchronize 및 merge를 수행한다. 그럼 여기서도 새로 브랜치를 생성하여 작업 내용을 반영해보도록 하겠다. 먼저 Team > Switch To > New branch로 새로운 브랜치를 생성한다.





생성을 완료하면 프로젝트 명 우측의 tag로 해당 프로젝트의 포커스가 새로 생성한 브랜치로 잡히는 것을 알 수 있다. 이제 test.txt 파일을 작성하여 커밋&푸시를 수행해보자.



그럼 푸시 대상이 다음과 같이 새로 생성한 브랜치로 잡히는 것을 알 수 있다. 확인 후 Next, Finish를 누른다.

Push to branch in remote

Select a remote and the name the branch should have in the remote.



Source:
test_branch 90dcb6e Commit text

Destination:
Remote: origin: https://github.com/username/test ▼ New Remote...
Branch: test_branch

☒ Configure upstream for push and pull

When pulling: Merge ▼

☐ Force overwrite branch in remote if it exists and has diverged
[Show advanced push dialog](#)

? < Back Next > Finish Cancel

Push Confirmation

Confirm following expected push result.



test_branch → test_branch [new branch]

Message Details

Repository
<https://github.com/username/test>

☐ 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

? < Back Next > Finish Cancel

New branch로 생성이 완료되면, 원격 저장소인 Github에서도 확인이 가능하다

2 commits 2 branches

Your recently pushed branches:

test_branch (less than a minute ago)

Branch: master New pull request

Switch branches/tags

Find or create a branch...

Branches Tags

✓ master

test_branch

.classpath

.project

commit first

commit first

commit_first

commit_first

commit first

commit first

Help people interested in this repository understand your project by adding a README.

그럼 이제 master에 해당 내용을 반영해보자, git에서는 이 과정을 Pull 또는 Pull Request라고 한다. 다음은 GitHub에서 Pull Request를 수행하는 과정이다. 브랜치를 작업하던 브랜치로 변경한 후, 우측 상단의 new pull Request를 누르면,

Your recently pushed branches:

test_branch (1 minute ago)

Branch: test_branch New pull request

This branch is 1 commit ahead of master.


user1 Commit text


.settings	commit first
.svn	commit first
bin/member	Commit text
src/member	Commit text
.classpath	commit first
.project	commit first

Help people interested in this repository understand your project by adding a README.

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: **test_branch** ✓ Able to merge. These branches can be automatically merged.



Commit text


Write

Preview

AA B i " < > ☰ ☷ ☹ ↶ @

Leave a comment


Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

 Styling with Markdown is supported

Create pull request

다음과 같은 창이 뜨면서 pull을 요청하는 사유를 적을 수 있으며 상단에는 해당 코드가 충돌이 나지 않는지 검사를 수행하여 가능할 경우 Able to merge라고 표시해준다. 하단에는 변경 사항을 출력해서 보여준다. 만약 충돌이 날 경우 하단에 충돌 나는 사유가 발생하므로 해당 소스를 주석처리 하거나 수정하여 머지를 수행하면 되겠다.

 Commits on Oct 21, 2017

  user1 Commit text

 Showing **2 changed files** with **2 additions** and **0 deletions**.

1 ■■■■■ bin/member/test.txt

... .. @@ -0,0 +1 @@

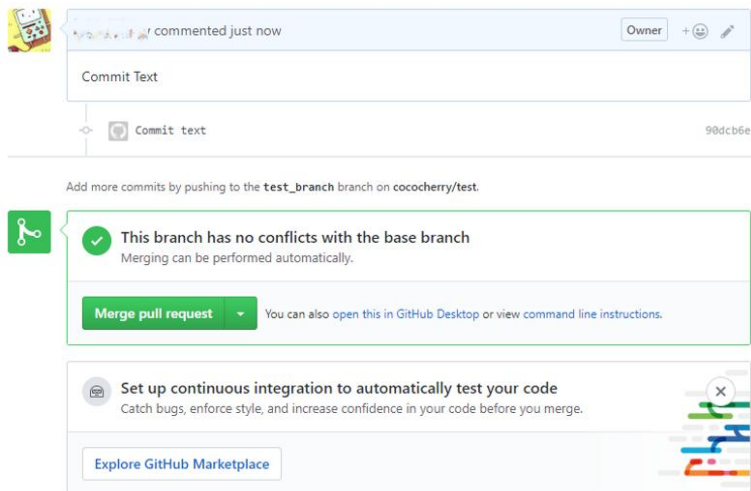
1 +test.txt

1 ■■■■■ src/member/test.txt

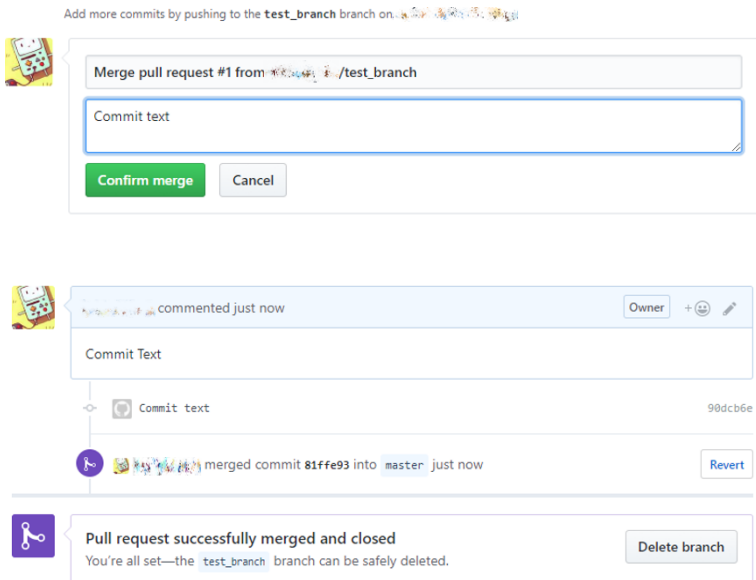
... .. @@ -0,0 +1 @@

1 +test.txt

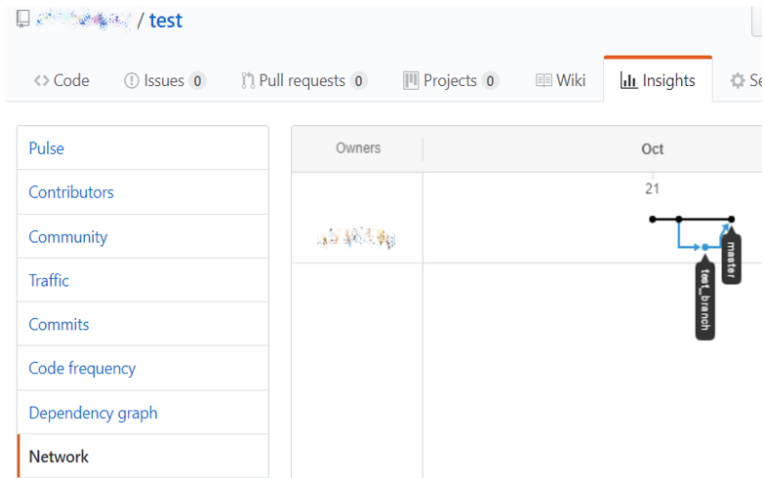
Request를 정상적으로 수행하면 이제 Owner 입장에서는 Merge pull request로 요청받은 머지를 수락할 수 있다.



수락하면 확인 메시지를 적을 수 있다.

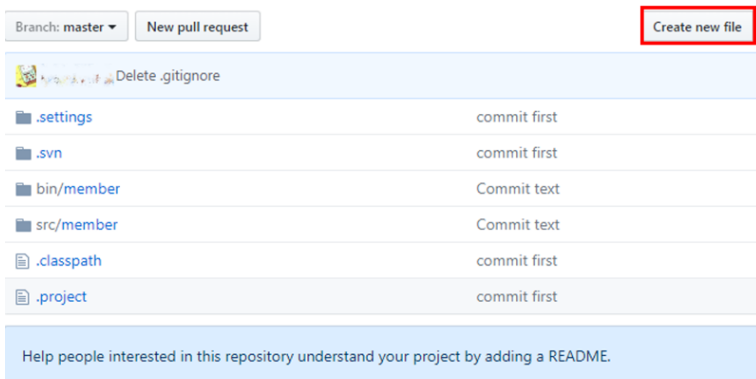


머지가 완료되면, 위와 같은 메시지가 나오고 해당 브랜치를 삭제할 수 있으며, Insights > Network 메뉴에서 다음과 같이 확인할 수 있다.

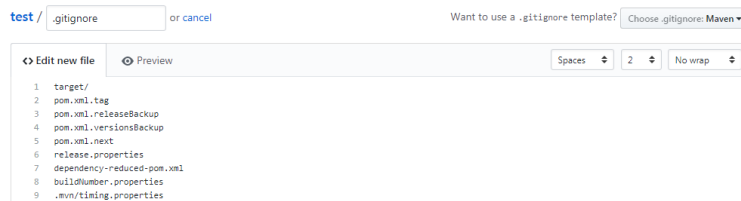


5 Git ignore 설정 활용

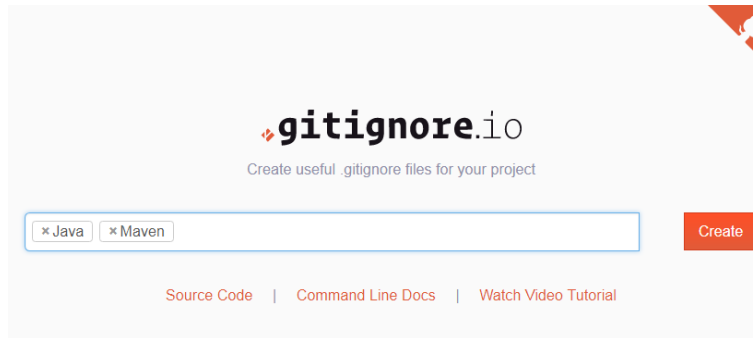
Git Ignore의 통합 설정은 저장소를 생성 시 적용해야 하는데, 그 이유는 중간에 생성하게 되어도 해당 파일이 저장소에 이미 올라와 있을 경우, git의 형상 관리를 받기 때문이다. 따라서 저장소 생성 시 무시하길 원하는 소스에 대한 설정을 하여 ignore 설정을 잡아 주도록 하자. ignore 파일 생성은 프로젝트 최 상단에서 하도록 한다. 먼저 github에서 우측 상단에 Create new file로 새로운 파일을 하나 생성한다.



파일 이름은 .gitignore 로 짓고, 내용 란에, 무시하길 원하는 파일을 적어 주면 된다. 참고로 github에서 제공하는 우측의 양식을 이용해도 된다.



만약 어떤 설정을 잡아줘야 할지 모르겠다면 <https://www.gitignore.io/> 사이트를 참조하여 설정을 작성하도록 하자.



gitignore.io는 원하는 키워드를 적은 뒤, Create를 누르게 되면, 밑의 이미지와 같이 필요한 설정에 대한 소스 코드를 작성해준다.

```
# Created by https://www.gitignore.io/api/java,maven

### Java ###
# Compiled class file
*.class

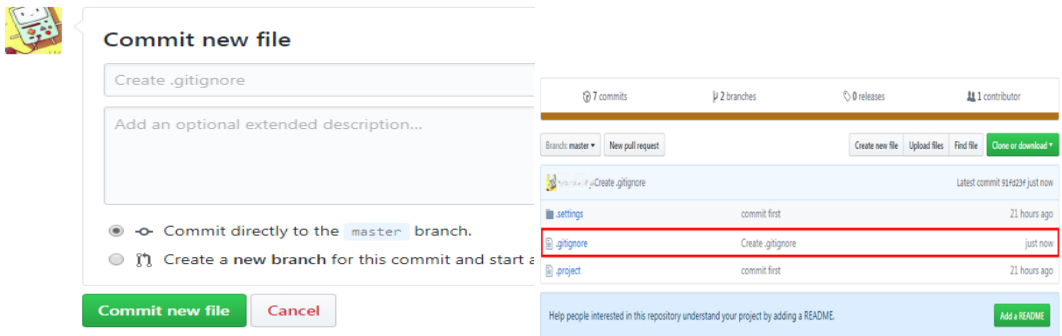
# Log file
*.log

# BlueJ files
*.ctxt




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

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

작성이 끝나면 커밋 처리를 하도록 하자. 그 후 화면과 같이 ignore 파일이 생성되어 있으면 성공!



이후 프로젝트를 Check Out으로 불러와서, ignore된 설정이 정상적으로 반영이 되어 있다면 commit 시에 무시할 파일은 목록에 뜨지 않을 것이다.

 test.ctxt 현재 test2.txt와 test.ctxt 두 개를 작성 후 commit 처리하려고 하였을 때,
 test.txt text.ctxt는 정상적으로 무시되어 목록에 없는 것을 확인 할 수 있다.
 test2.txt

```
# Created by https://www.gitignore.io/api/java,maven

### Java ###
# Compiled class file
*.class

# Log file
*.log

# BlueJ files
*.ctxt

> test [test1]
```

Unstaged Changes (1)

test2.txt - src/member

Staged Changes (0)

Git ignore 설정은 저장소를 생성 시 함께 작성할 수 있는데,
github에서 저장소를 생성 시, 하위 항목을 설정하여 진행하면 된다.

Owner / Repository name

test

Great repository names are short and memorable. Need inspiration? How about ub

Description (optional)

test

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

☐ Private
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're ir

Add .gitignore: None

Add a license: None

Create repository