

git 를 사용한 공동 작업 및 버전관리

ver 2.5.1
(2020.06/it여행자)

프로그램을 개발하다보면 하나 이상의 프로그램이 버전업이 되거나 버전업은 아니지만 프로세스의 변경등과 같이 기존 프로그램이 수정되는 경우가 다반사입니다. 이때 파일명에 버전표시를 추가하거나, 아예 폴더를 추가하여 수정된 프로그램을 저장하여 이전 프로그램과 구분하여 소스등을 관리하곤 합니다.

그러나 자칫 잘못하면 기존 소스가 그대로 수정되어 난감한 상황이 발생하기도 하죠. 이럴 때 소스의 버전관리를 자동화 할수 없을까 하여 방법을 찾곤합니다.

이럴 때 사용되는 툴이 형상관리 툴입니다. 그중에 하나가 git인데 이 git이 현재 개발자들 사이에서 가장 많이 사용되는 프로그램입니다. git 이외에도 유명한 몇가지를 간단히 소개하자면 아래와 같습니다.

종 류	특 징
CVS	<ul style="list-style-type: none">- Concurrent Version System- 1986년 개발되었으며, 서버-클라이언트 개념으로 사용됨.- 파일전체가 아니라 변경된 부분만 저장됨.- 안정적이지만 파일명 변경이나 폴더 관리가 다소 어려움.
SVN	<ul style="list-style-type: none">- CVS의 단점을 보완하여 2000년에 만들어짐.- 중앙 관리만 지원
Git	<ul style="list-style-type: none">- 리누즈 토발즈에 의해 2005년 개발됨.- 로컬, 원격 저장 모두 지원.- GUI툴 빈약.- 배우기 다소 어려움.

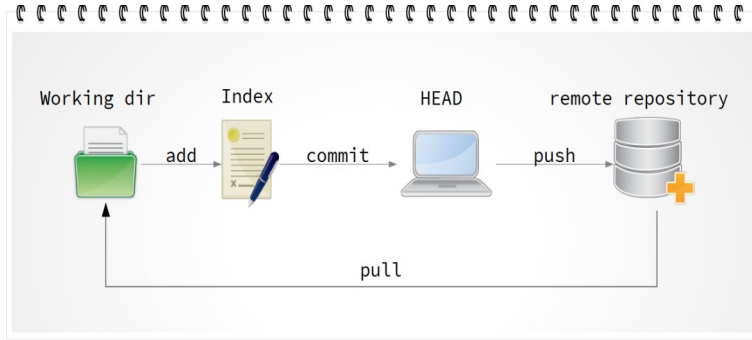
1.1. Git 사용하기

1.1.1. 개요

- git은 리누즈 토발즈에 의해 리눅스 커널의 형상 관리를 위해 만듦.
- github는 git의 최대 저장 공간이다.

1.1.2. git의 데이터 흐름

git을 사용할 때 많은 용어들이 필요하지만 대표적인 용어는 아래와 같습니다.



- 1) 작업 폴더(Working dir)에서 파일이 수정되면 Index 요소에 수정된 정보가 추가됩니다.
- 2) 작업자가 commit을 수행하면 Index에 있던 정보들이 Head에 생성됩니다.
- 3) push를 실행하면 Head에 있던 정보들이 원격 저장소(remote repository)에 저장됩니다.
- 4) pull을 하게되면 원격 저장소에 있던 정보들이 로컬 작업 폴더에 다운로드 됩니다.

1.1.3. 중요 용어

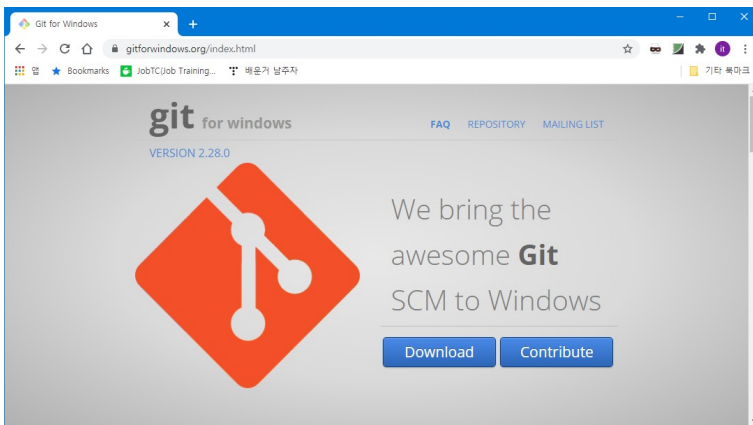
용 어	설 명
working dir	작업 코드가 있는 폴더
index	stage 영역(준비영역)을 의미
HEAD	commit에 의해 최종 확정본을 의미
commit	파일의 변경 상황을 저장소에 저장하는 것.
push	파일의 변경 상황을 원격 저장소에 업로드 하는 것.
pull	원격 저장소에 있는 정보를 로컬로 가져오는 것.
branch	master로 진행된 형상 관리 이력을 다른 가지(branch)들로 묶어 다른 부분에 영향을 최소화 하면서 저장하는 것. 추후 master가 병합(marge)할 수 있다.
revert	commit 된 정보를 이전 버전으로 되돌리는 것.

2. GUI Git 프로그램 설치

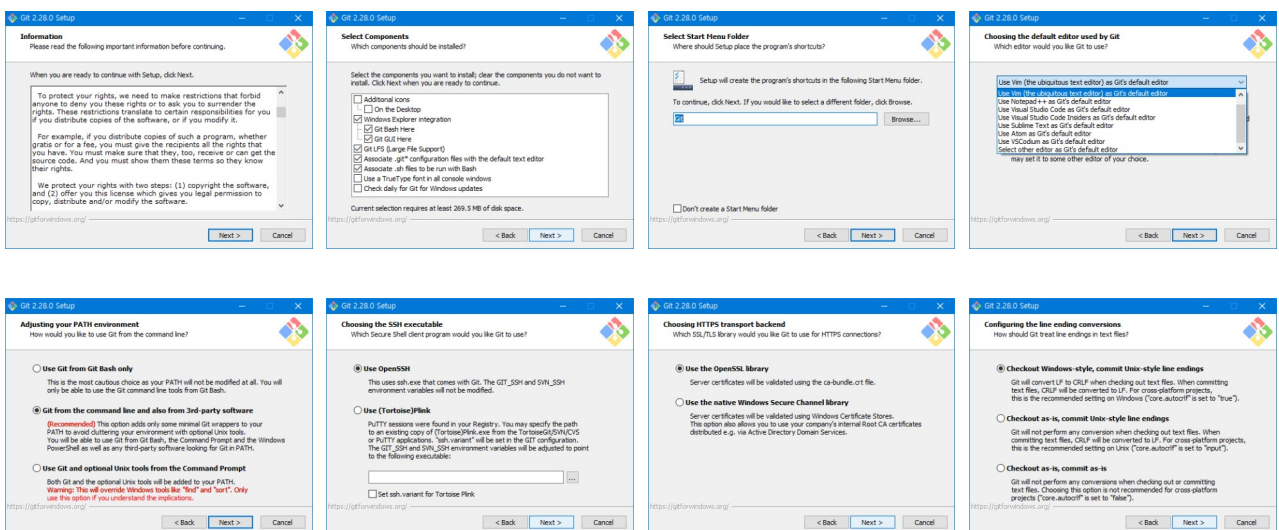
도스창과 같은 콘솔모드에서 git명령을 통해 사용할 수도 있지만 처음 접근하는 개발자들에게는 쉬운 접근법이 아니여서, git의 모든 기능을 사용할 수는 없지만 보다 쉽고 편하게 사용해 보도록 GUI형태의 프로그램을 사용하여 설명하도록 하겠습니다.

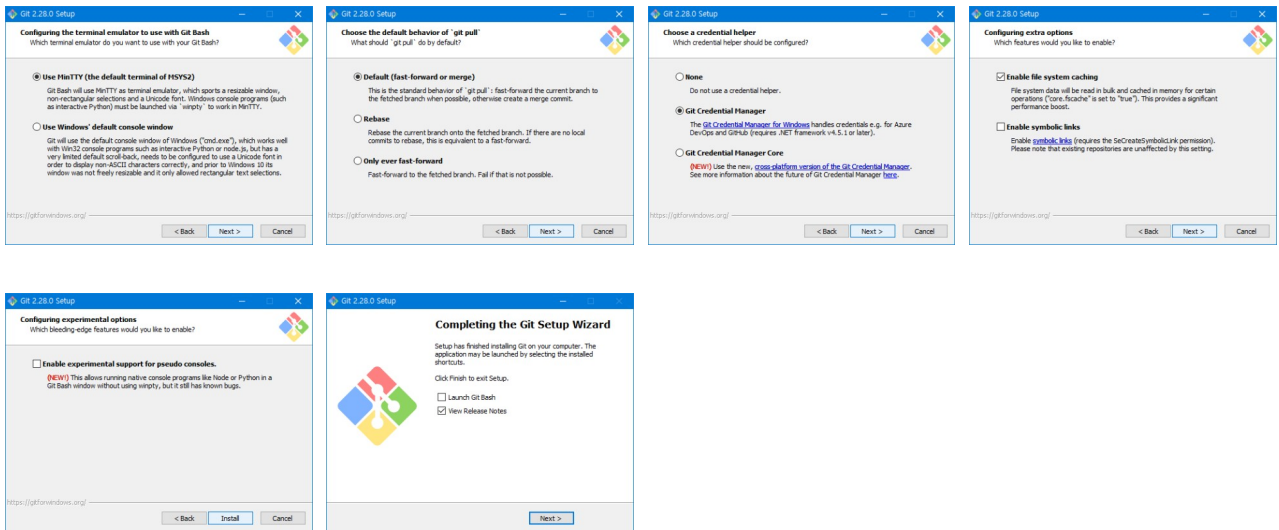
2.1. git GUI 툴 download and install

사이트 : <https://gitforwindows.org/index.html>



다운로드 받은 실행 파일을 실행하여 프로그램을 설치하게 되면 아래와 같은 각종 설정 화면들이 나옵니다. 웬만하면 그냥 초기값으로 계속 진행하셔도 무방합니다.

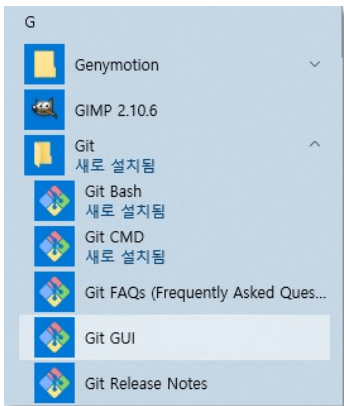




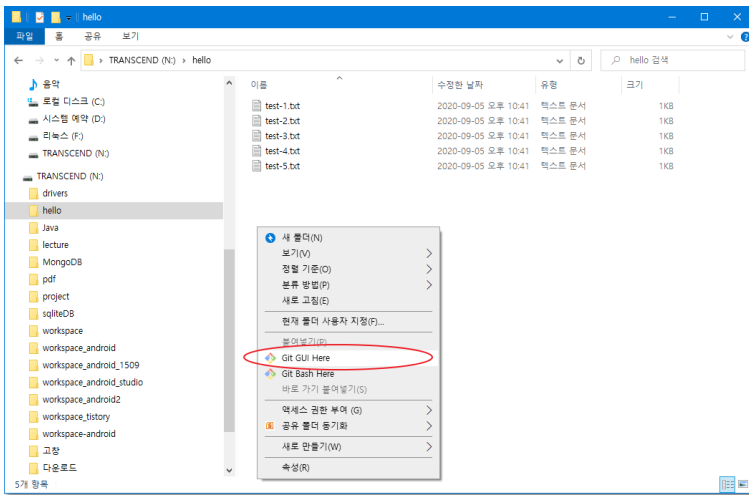
2.2. Git GUI 실행하기

[실행 방법 2 가지]

1) 윈도우 시작 메뉴를 사용하는 방법 : 윈도우 시작메뉴 > Git > Git GUI



2) 탐색기를 사용하는 방법 : 해당 폴더를 열고, 마우스 우클릭



윈도우 메뉴를 선택하여 Git GUI를 열면 무조건 아래의 창이 뜨고, 탐색기를 통해 Git GUI창을 열었을 때는 Repository 정보의 유무에 따라 서로 다른 창이 뜹니다.

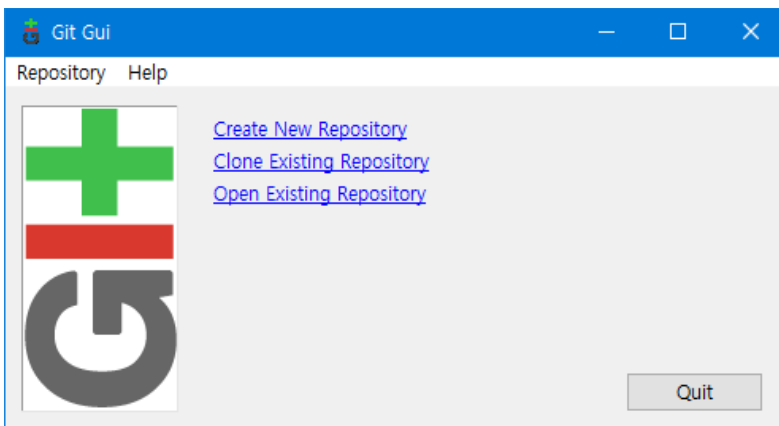


그림 1: repository 정보가 없는 경우

git의 사용 이력이 있는 경우에는 아래와 같이 목록이 함께 출력됩니다.

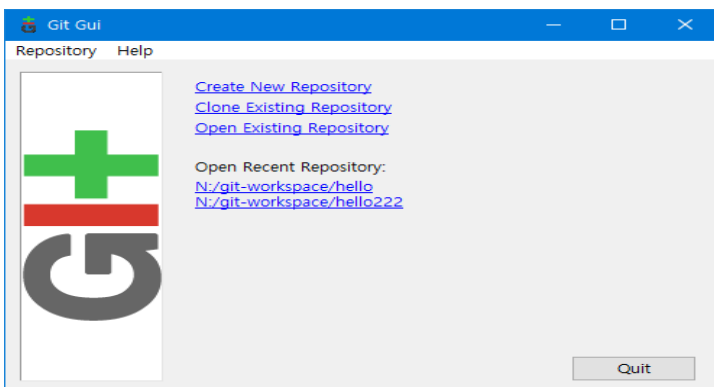


그림 2: git 사용 이력이 있는 경우

2.3. 새로운 Local Repository 생성하기

Create New Repository를 선택하여 탐색기가 브라우징되면 새로운 폴더를 생성한 후 선택합니다. 이때 폴더 내부에 어떤 내용들이 들어 있으면 안됩니다.

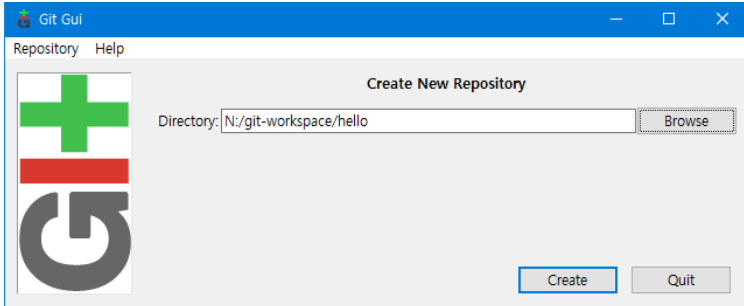


그림 3: local repository 생성시

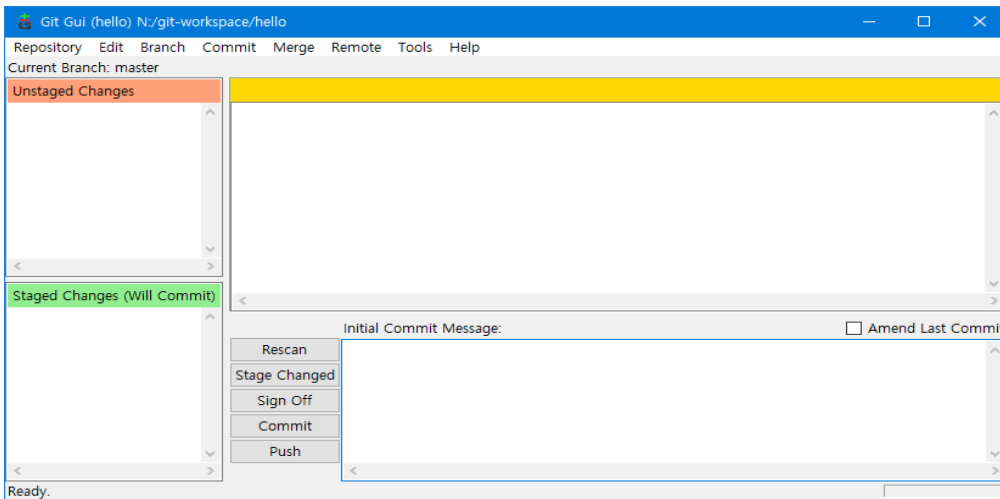
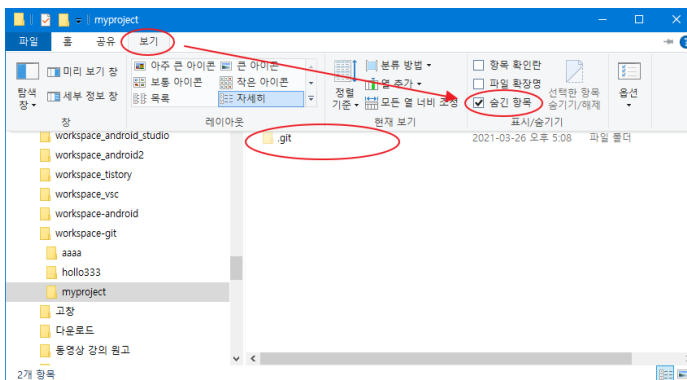


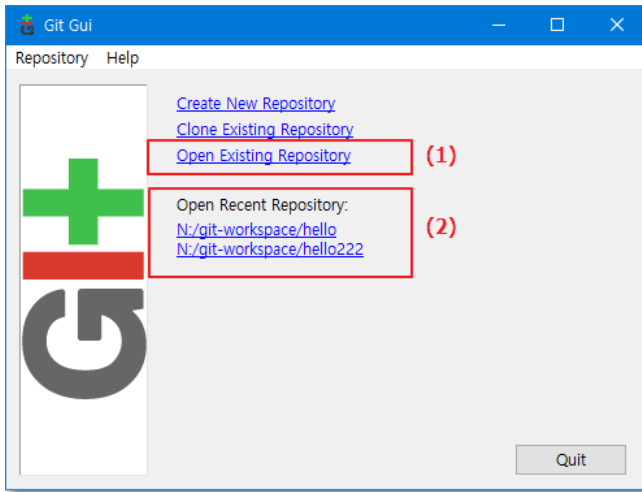
그림 4: 그림 4: 새로운 local repository 생성후

repository를 생성하게 되면 실제 폴더에 .git 폴더가 추가되면서 git의 상태 정보들을 저장하게 됩니다.



탐색기의 기본 설정값은 숨긴항목을 보여주지 않기 때문에 .git 폴더가 보이지 않습니다. 이 때 탐색기의 보기 메뉴를 선택하여 숨긴항목을 보이게 하거나 감출 수 있습니다.

2.4. 기존 Repository 가져오기



(1) 탐색기를 열어 기존에 존재하던 repository를 선택하도록 합니다.

(2) 이미 사용했던 Repository가 있다면 하단에 목록이 표시되는데, 이 때 목록에 나열되어 있는 항목을 클릭하면 됩니다.

★ Clone Existing Repository 는 원격 Repository 사용법에서 사용하게 됨.

3. 로컬 Repository 사용하기

Git을 처음 사용할 것이라 가정하고 진행하기로 합니다. 먼저 탐색기등을 통해 Git을 사용할 폴더를 생성한 후 Create New Repository 메뉴를 선택합니다.

3.1. 로컬에 새 Repository 생성하기

step 1.

Git에서 사용하게 될 폴더 생성

```
/workspace-git/myproject
```

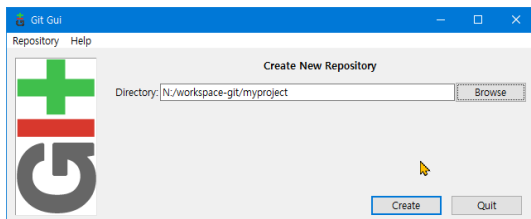
step 2.

Git GUI를 실행하여 Create New Repository 선택한 후 step 1에서 생성한 폴더 지정

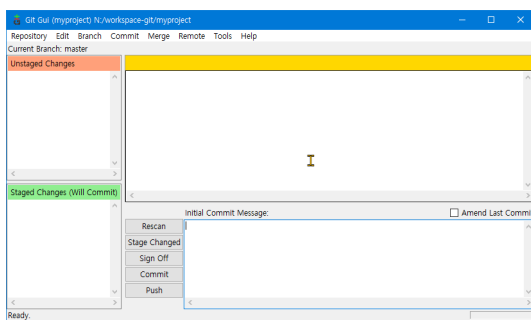


Open Recent Repository는 처음 실행시 보이지 않을 것입니다.

Create New Repository를 선택합니다.



step 1에서 생성한 폴더를 선택한 후 Create 버튼을 클릭합니다.



빈공간의 화면이 실행됩니다.

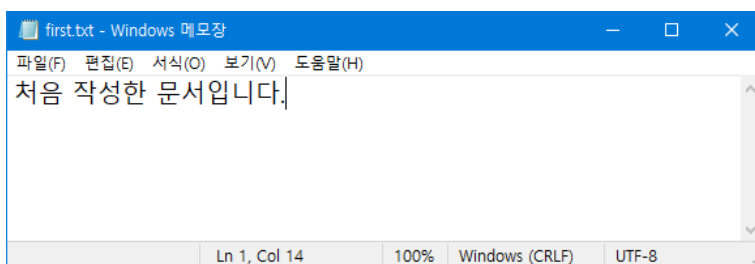
3.2. Commit 하기

수정된 파일들을 Git이 관리하고 있는 저장 공간에 저장하는 것을 commit이라고 합니다.

step 1.

Working Dir에서 메모장을 열어 텍스트 문서를 하나 추가하여 내용을 편집합니다.

```
/workspace-git/myproject/first.txt
```



선택된 로컬 Repository 내에서 파일을 편집하게 되면 수정된 상태를 Git이 인식하게 됩니다. 그러나 Git GUI가 자동으로 인식된 결과를 표시해 주지 않기 때문에 Rescan 버튼을 클릭하여 수정된 파일들의 목록을 확인해야 하는데 이는 Unstaged Changes에 표시됩니다.

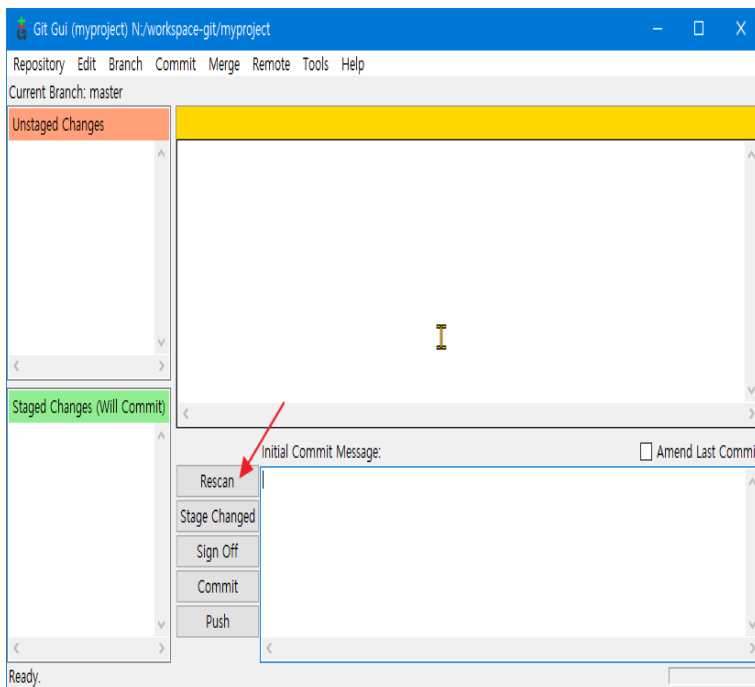


그림 5: Rescan 이전

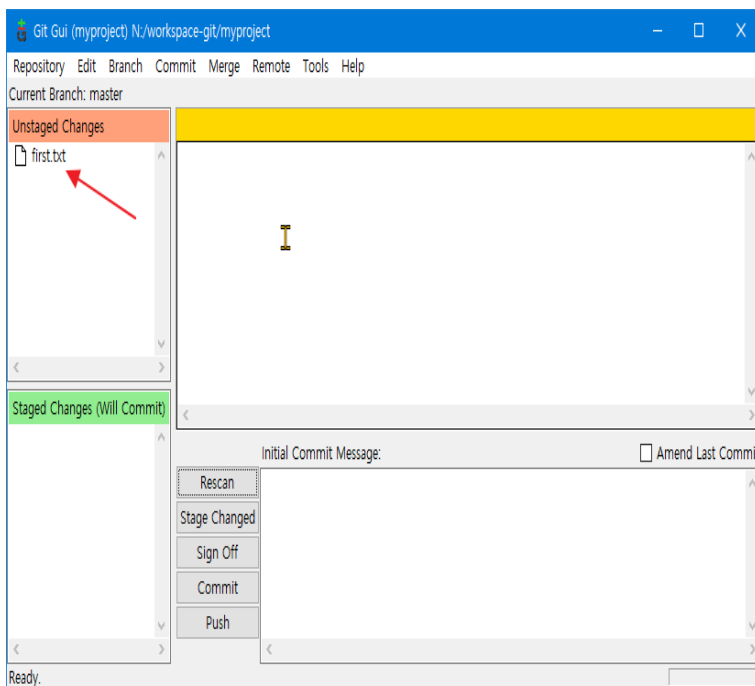


그림 6: Rescan 이후

변경된 파일이 있다면 Git은 가장 먼저 Git이 관리하는 Index 영역에 관련된 내용을 저장합니다. 이를 Rescan을 통해 표시하고 있는 것입니다.

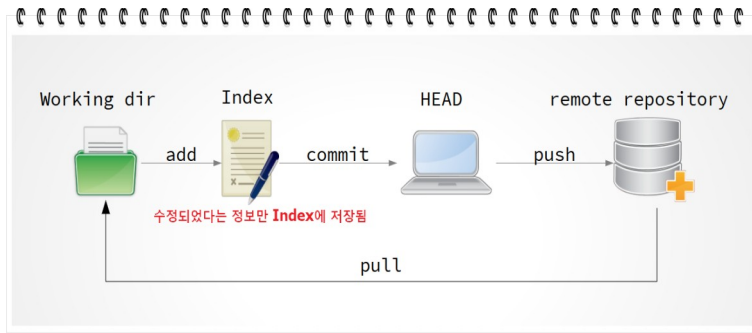


그림 7: 수정 정보가 Index에 저장

한글로되어 있는 문서라면 아마도 한글이 정상적으로 표시되지 않을 수 있습니다. 이때는 edit>option에 들어가서 한글 인코딩 방식을 utf-8를 변경하면 됩니다.

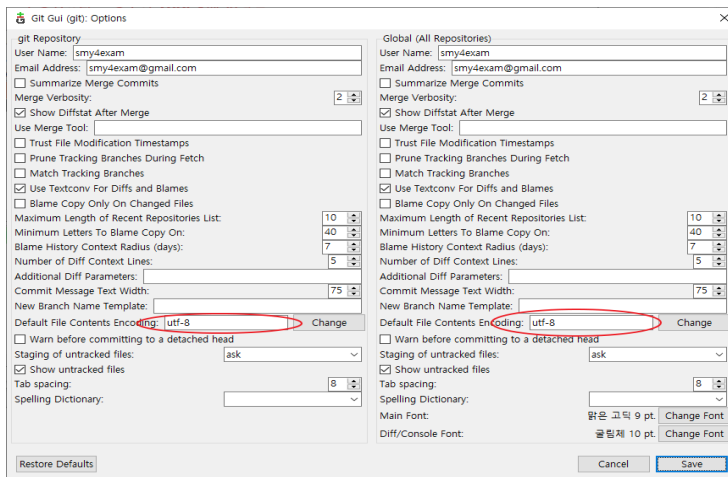


그림 8: 한글 인코딩

step 2.

Repository내에서 파일이 수정되면 Index 영역에 관련된 내용이 추가됩니다. 그러나 실제로 수정된 내용이 Repository에 저장된 것은 아닙니다. 단순히 수정된 이력만이 Index에 추가될 뿐이지요. Rescan 후 수정된 파일의 목록이 Unstaged Changes에 표시되었다면 Stage Changed 버튼을 클릭하여 Commit 대상 파일로 변경해야 합니다. 파일명 앞에 있는 **아이콘을 클릭하여** 한개씩 stage로 옮길 수도 있습니다.

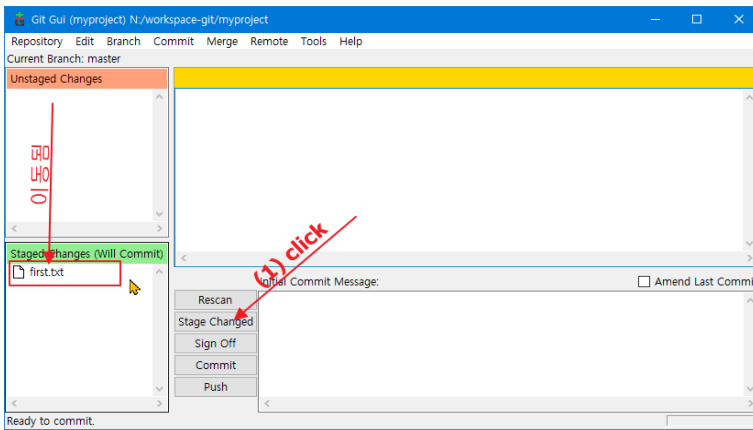


그림 9: Stage로 파일 이동

만약 stage 상태로 되어 있는 파일을 다시 Unstaged Changes 항목으로 이동시키려면 파일명 앞에 있는 **아이콘**을 **클릭**하면 됩니다.

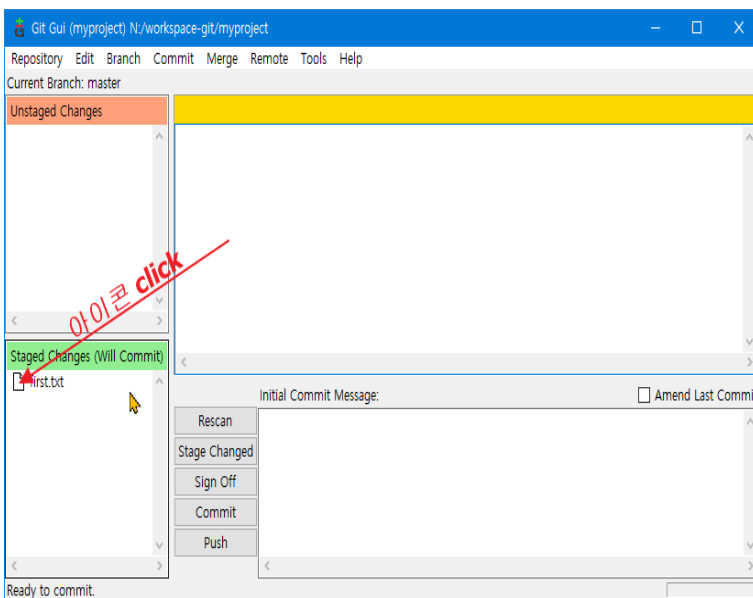


그림 10: Unstaged로 다시 옮길 때

하단의 파란색 원에 있는 내용은 한글이 깨져 보일 때 utf-8로 설정하면 됩니다. 설정해 주는 것을 권장 합니다.

step 4.

환경이 수정되었으니, 수정된 파일과 파일의 내용을 최종적으로 Repository에 저장해 봅시다.

이렇게 Rescan된 정보를 실제로 저장하려면 Commit 버튼을 클릭하면 됩니다. Commit 할때는 반드시 Git GUI 프로그램의 오른쪽 하단에 있는 창에 Commit Message를 작성해야 합니다. 나중에 어떤 정보가 수정되었는지 알아 보기 좋게 자세히 작성하는 것이 좋습니다.

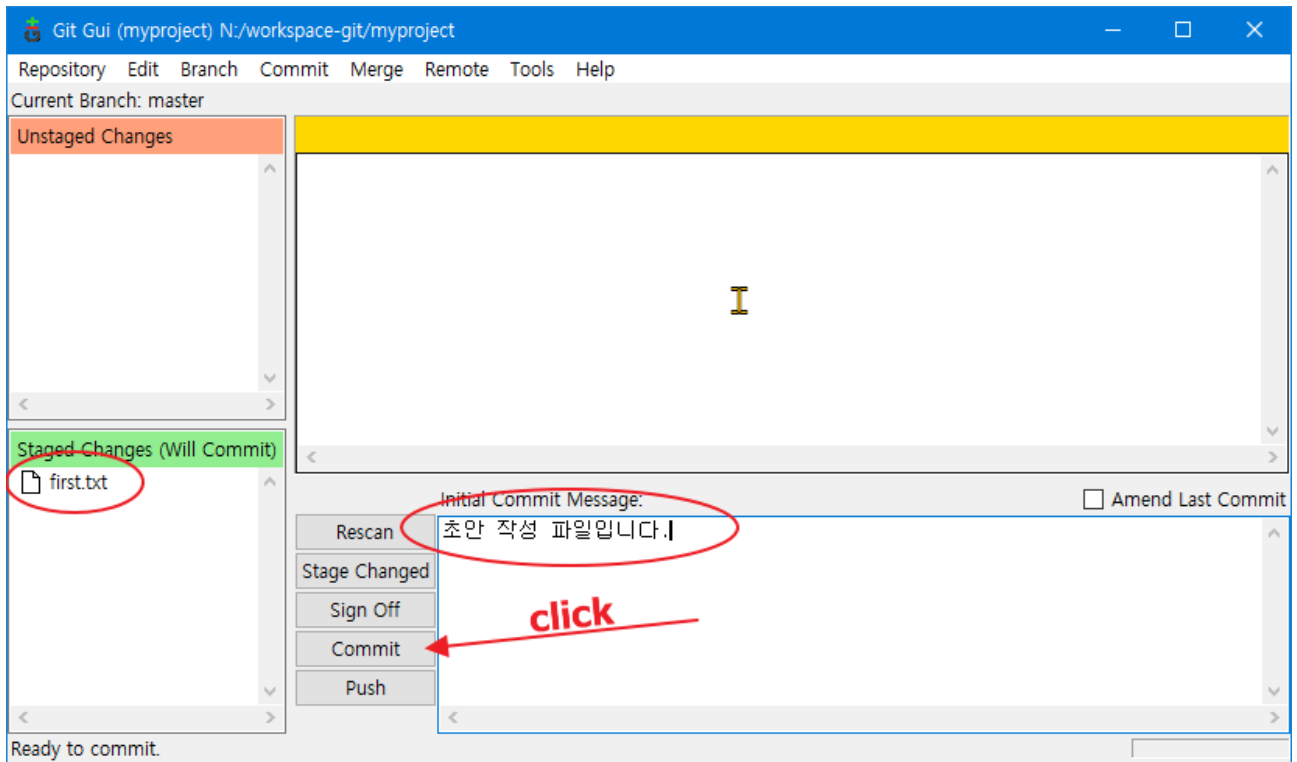


그림 11: commit 하기

commit 버튼을 클릭하면 수정 정보가 최종적으로 local repository에 저장되고 모든 내용은 삭제된 빈 화면이 다시 나타납니다.

파일을 추가하거나 기존 파일을 수정한 후 다시

Rescan -> Stage Changed -> 메시지 작성 -> Commit

과정을 반복 하시면 됩니다.

3.3. Commit 목록 확인

Git GUI 툴을 사용하여 Commit 된 이력들을 확인해 볼 수 있습니다.

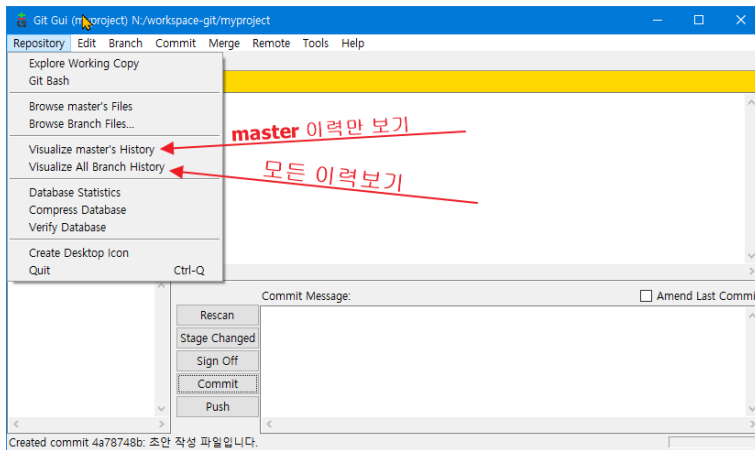


그림 12: commit 이력 보기

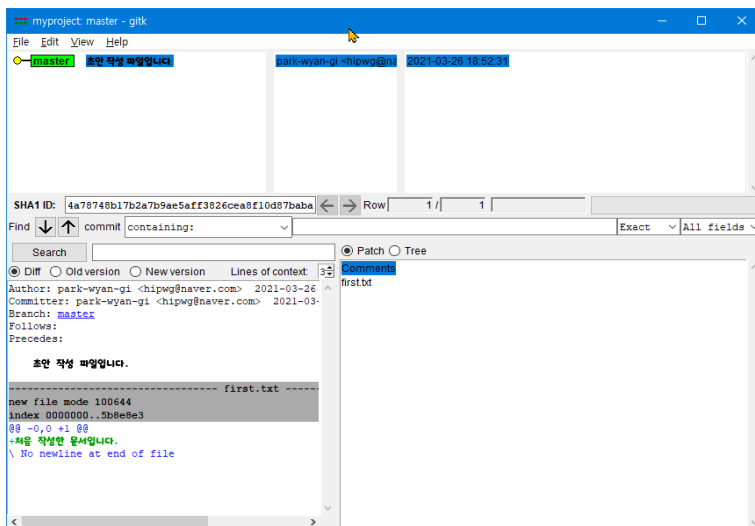


그림 13: master 이력

현재로서는 다른 branch가 없기 때문에 master's history나 All branch history나 차이가 없습니다. 일단, master's의 history목록을 보면서 진행하도록 하겠습니다.

step 1. 기존 파일 수정

commit 목록을 확인하기 위해 앞서서 작성했던 first.txt 문서에 아래와 같이 내용을 수정한 후

Rescan -> Stage Changed -> 메시지 작성 -> Commit

과정을 한번 더 진행합니다.

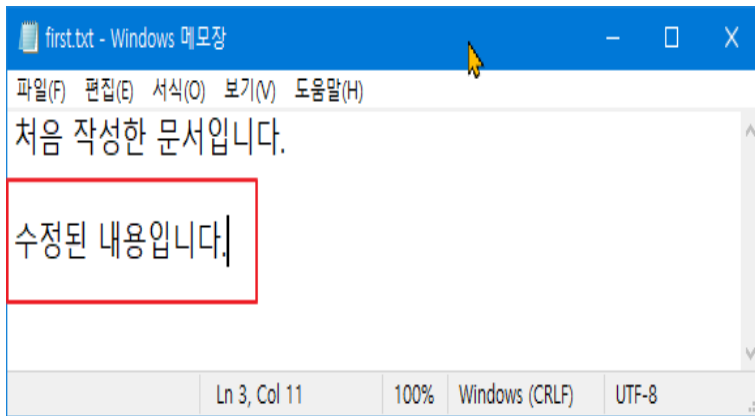
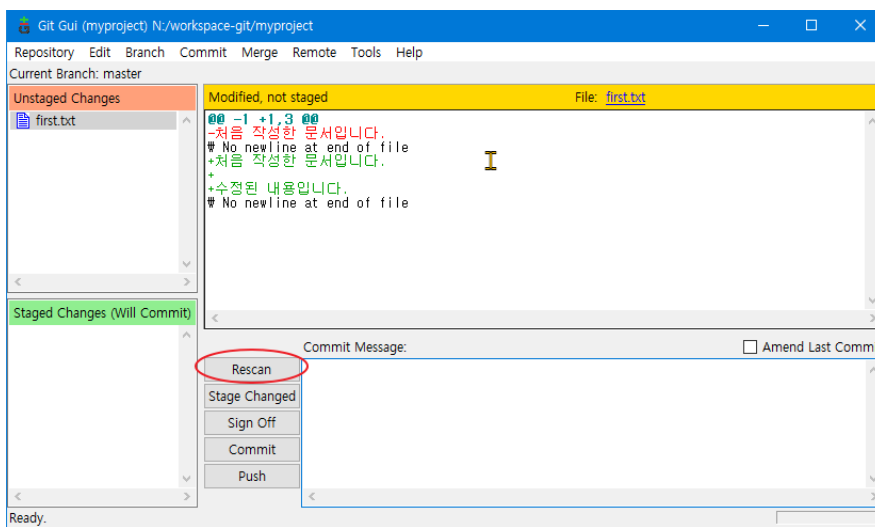
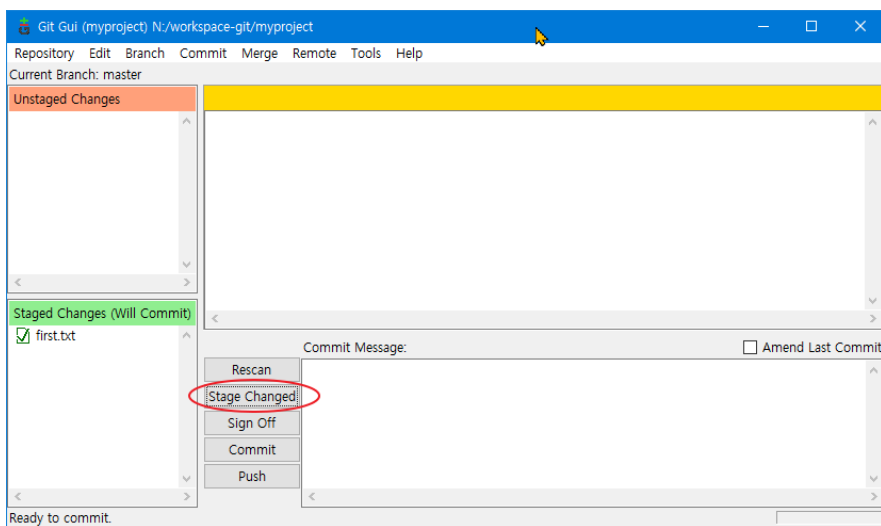


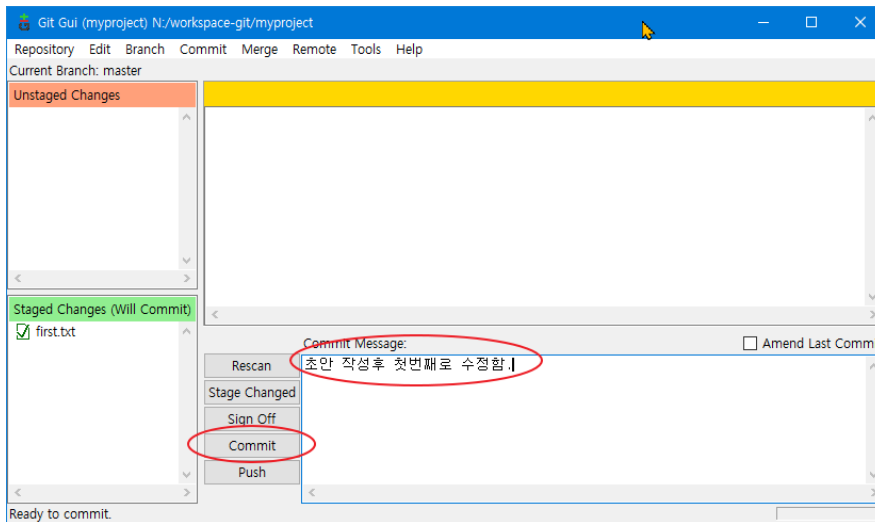
그림 14: 기존 파일 내용 수정



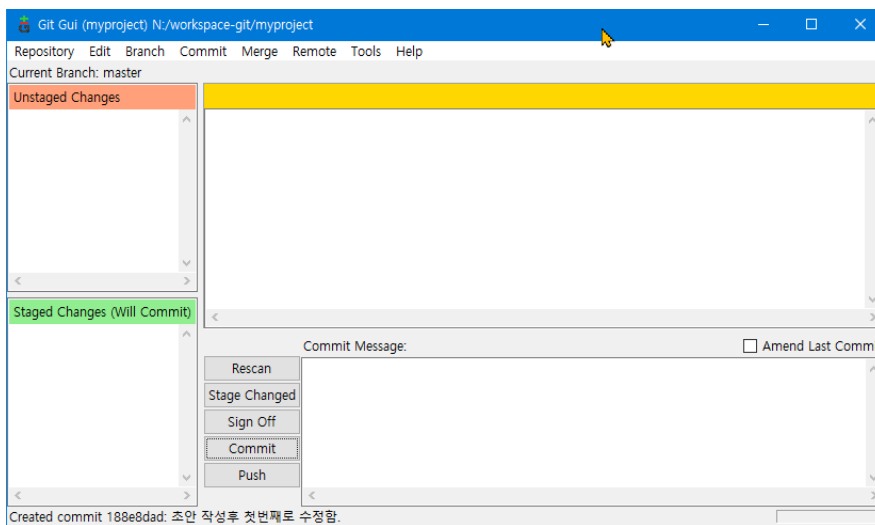
first.txt 문서를 저장한 후
Rescan 버튼을 클릭한 이후.



Stage Changed 버튼을 클릭한
이후.

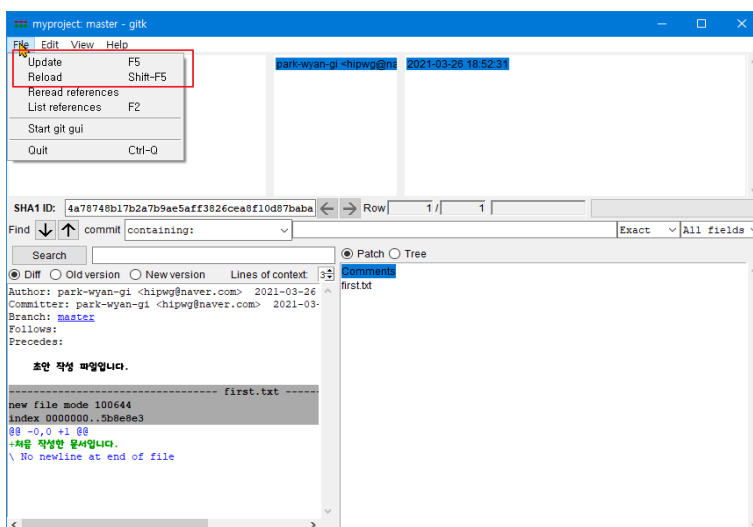


Commit Message 작성후 commit 버튼 클릭.



commit 된 이후.

자, 그럼, master의 commit 이력을 보기 위해 Master's History 창에서 Update 메뉴를 선택합니다.



Update 메뉴 선택

그림 15: commit 정보 새로 고침 이전

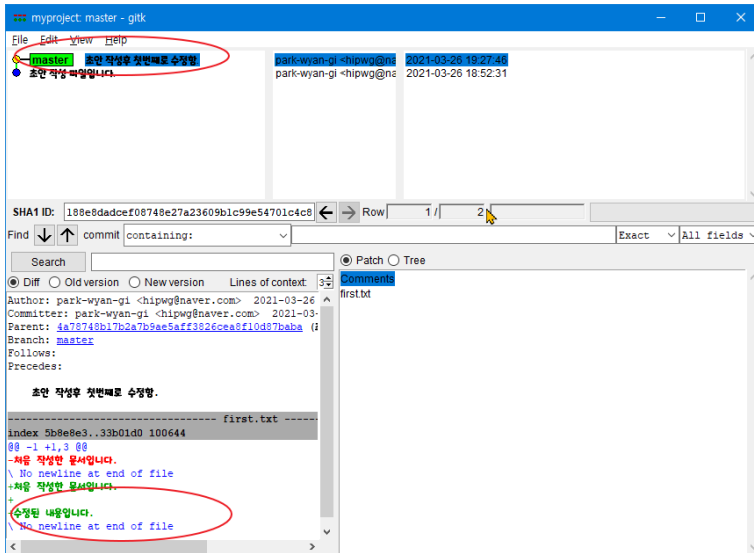


그림 16: Update 새로 고침 이후

왼쪽 그림처럼 여러 가지 정보들이 새로 나타나는것을 알 수 있습니다.

commit을 할 때 마다 히스토리 목록은 계속해서 추가될 것입니다. 맨 위쪽의 정보가 최신 정보입니다.

3.4. Revert Changes

Revert Changes는 commit된 내용을 이전 내용으로 되돌리는 기능을 말합니다.

여러단계의 Revert를 테스트하기 위해 first.txt 문서에 내용을 한번 더 수정한 후 commit한 후 진행하겠습니다.

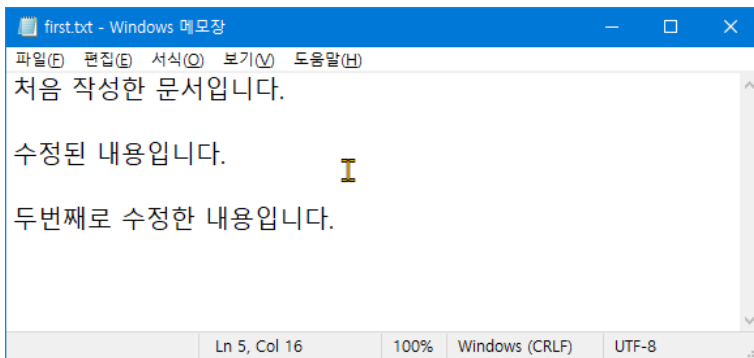
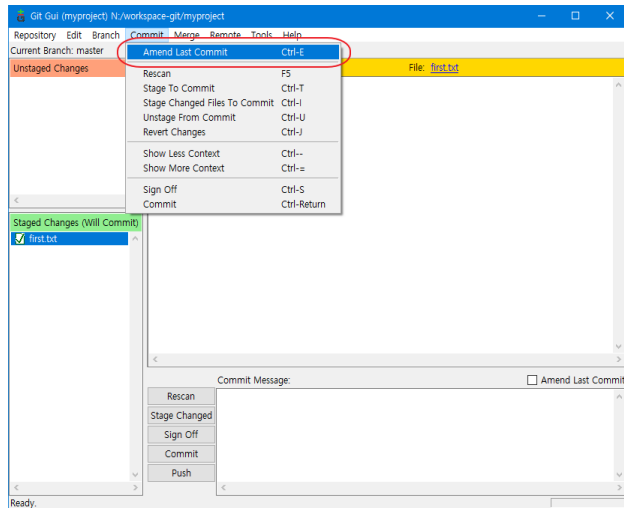


그림 17: revert를 위해 한번 더 수정한 문서 내용

내용 수정.

3.5. Revert 취소

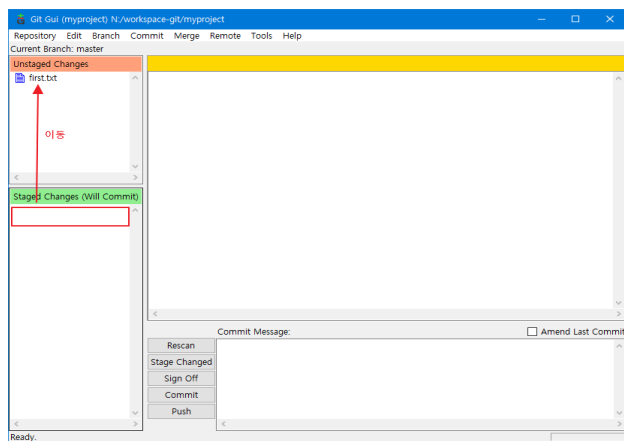
commit 했던 작업을 취소하는 명령이 Revert인데, 만약 잘못 Revert되었다면 대단히 곤란한 상황이 발생할 수도 있습니다. 이 때 잘못 Revert된 것을 취소할 수도 있습니다.



step 1.

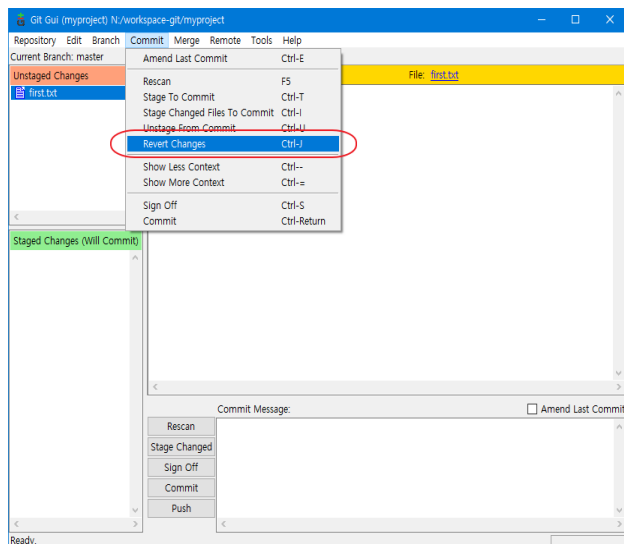
Commit>Amend Last Commit

메뉴를 선택합니다.



step 2.

Amend Last Commit 메뉴가 선택되면 마지막으로 commit되었던 파일 목록이 stage에 표시되는데 이때 필요한 파일의 아이콘을 선택하면 Unstage에 파일이 이동됩니다.



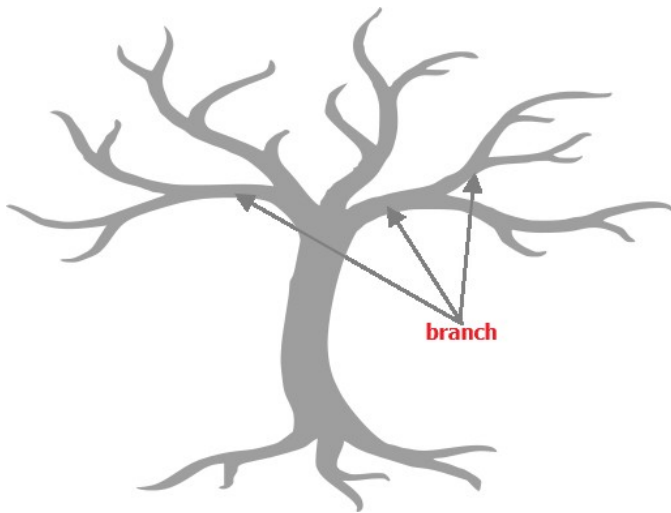
step 3.

Commit>Revert Changes

메뉴를 클릭하면 확인창이 뜨고 OK 버튼을 클릭하면 commit 이전 파일의 내용으로 변경된것을 확인할 수 있습니다.

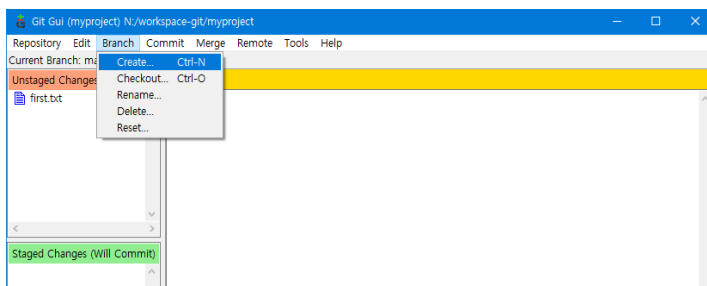
3.6. Branch

git을 로컬에서 혼자 사용할 때 보다는 여러명이 협업을 할 때 많이 사용하는 개념입니다. 직역 하자면 '가지'라는 의미인데, 편집 방향을 여러 형태로 나누어 편집하여 저장한 후 다른 유형과 비교한 후 가장 적절한 것을 선택하여 적용하려 할 때 주로 사용됩니다.



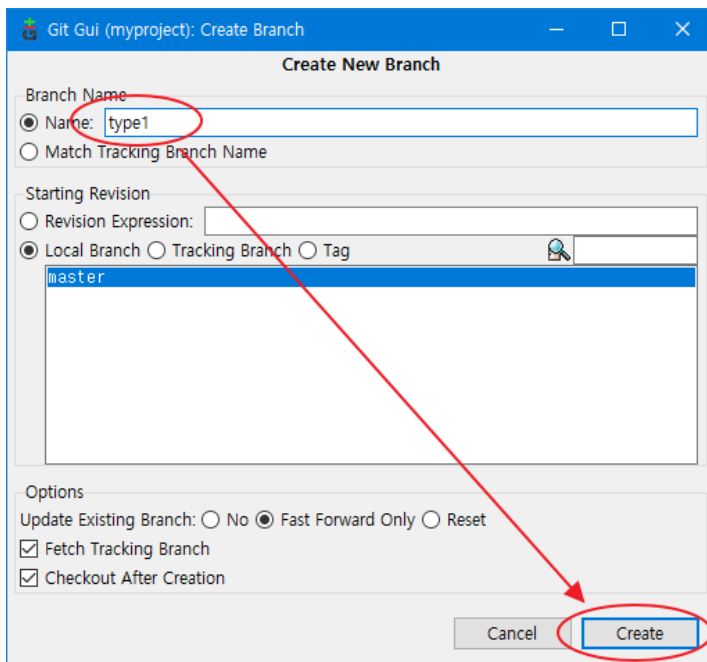
3.6.1. branch 생성하기

[branch 생성 절차]



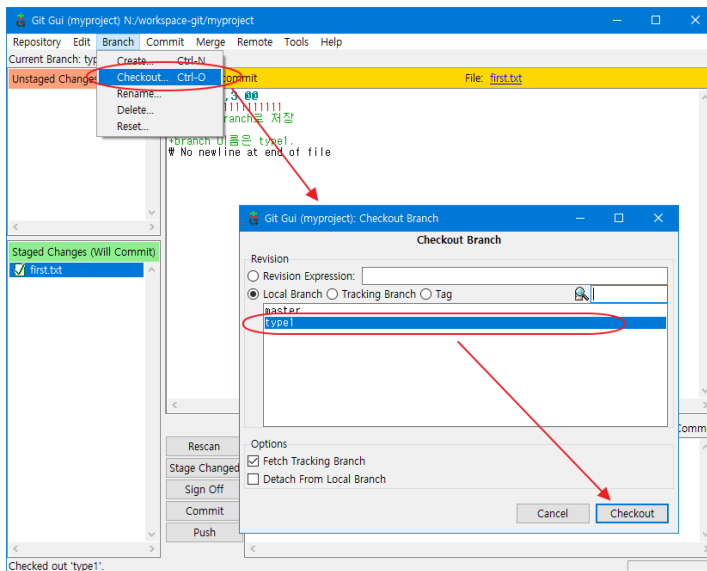
step 1.

Git GUI 툴에서 메뉴
Branch>Create...
를 선택한다.



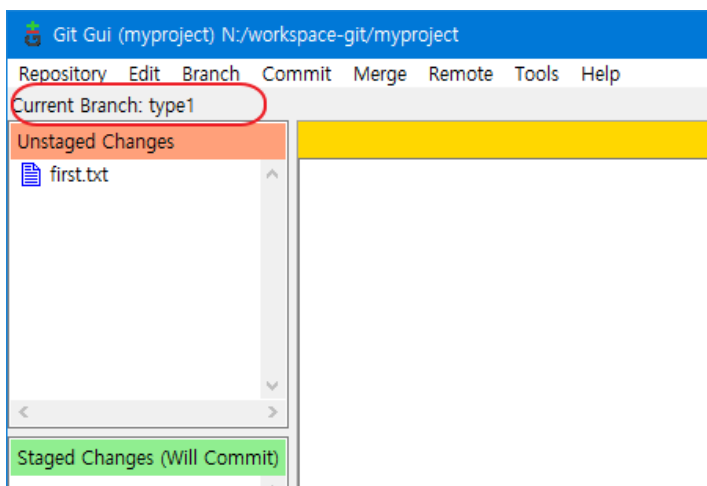
step 2.

Name항목에 branch로 사용할 적당한 이름을 넣고 Create 버튼을 클릭한다.



step 3.

Branch메뉴의 Checkout 메뉴를 선택하면 새로운 다이얼로그가 보입니다. 이 때 새로 생성한 branch이름을 선택한 후 Checkout 버튼을 클릭합니다.

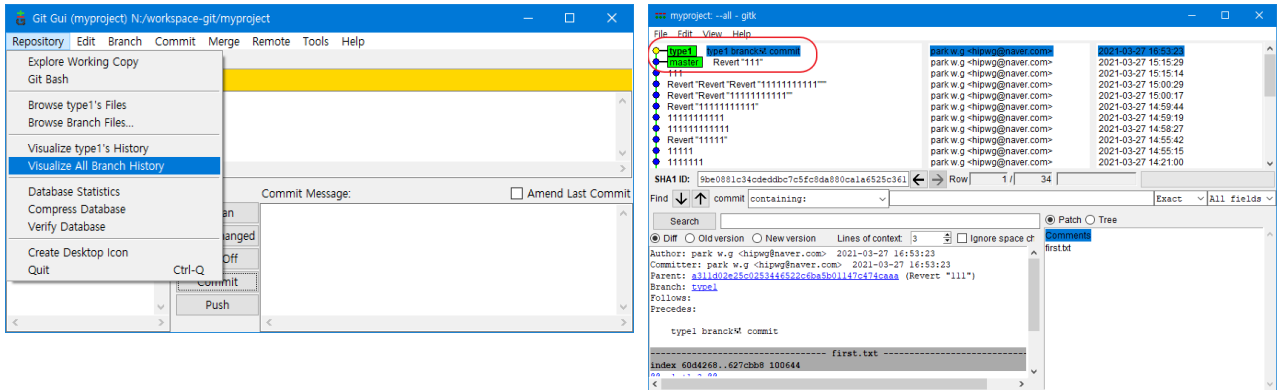


step 4.

Curent Branch 이름이 master에서 type1로 변경되었음을 알 수 있습니다.

3.6.2. commit 된 branch 확인

branch로 다른 것으로 변경된 경우 commit된 정보는 master용 Visualize로 볼 수 없습니다. 아래의 그림 처럼, type1 용 History나 All Brach History 메뉴를 선택해야 볼 수 있습니다. 아래의 그림은 All Branch History 메뉴를 사용하여 본 그림입니다.



아래의 그림은 type2 branch를 새로 만든후 master, type1, type2 branch 이름으로 계속해서 commit된 결과입니다.

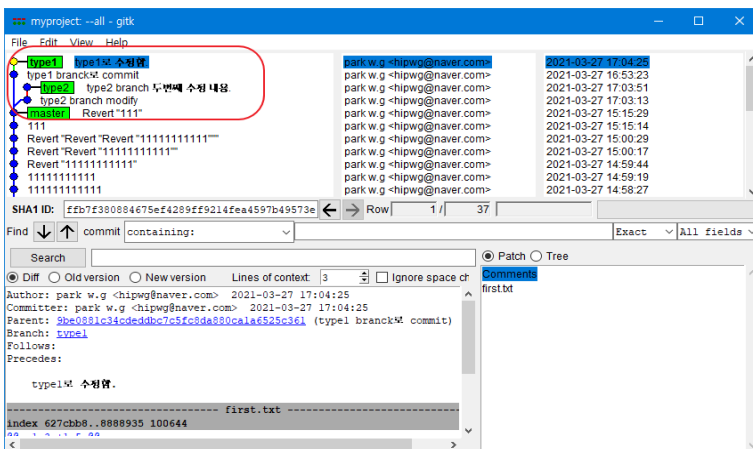
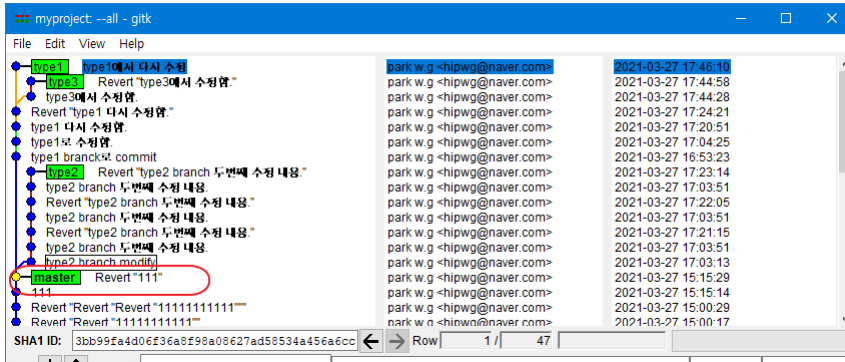


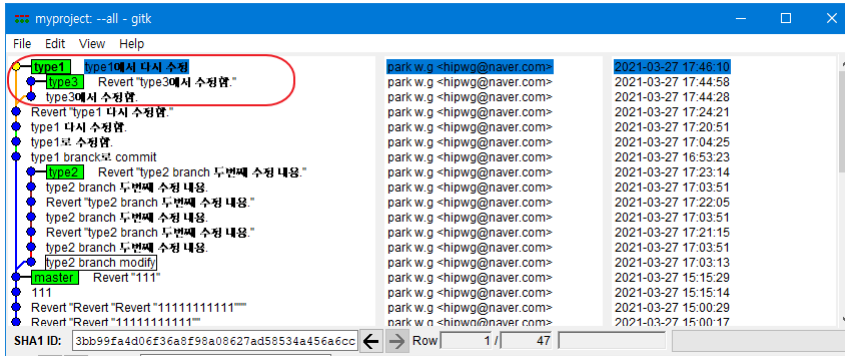
그림 20: 여러 branch로 commit 한 결과

3.6.3. check out

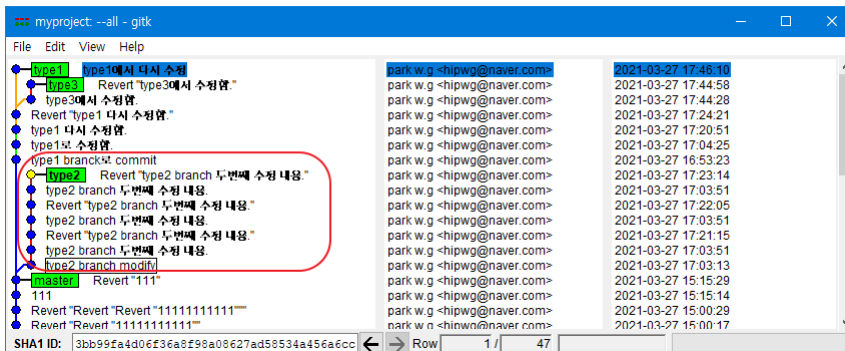
branch를 check out하고 다른 branch를 선택하면 기존 코드가 해당 branch에서 작성된 내용으로 자동 변경됩니다.



master로 checkout한 경우



type1로 checkout한 경우



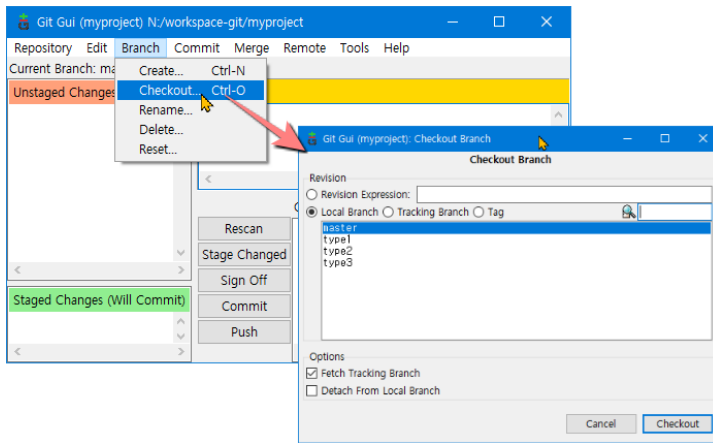
type2로 checkout한 경우

Working Dir에 가서 실제 코드들도 확인해 보시기 바랍니다.

3.7. merge

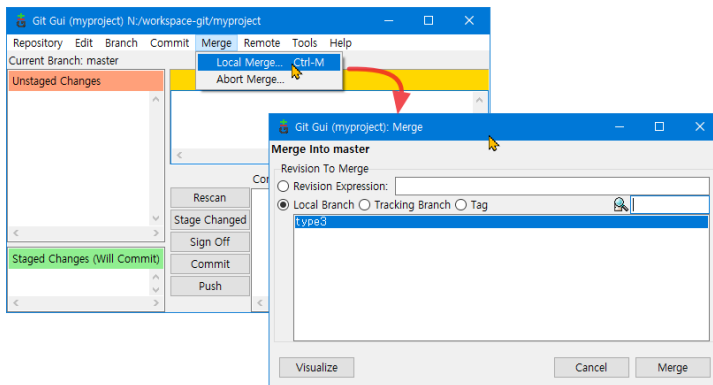
여러가지 특성상 다양한 branch를 나누어 작업하다가 필요한 내용들을 합쳐야 될 경우가 발생합니다. 이 때 merge를 사용하여 필요한 부분들을 병합할 수 있습니다. 이 때 중요한 체크 사항은 현재 작업하는 있는 작업들을 모두 commit 한 뒤 작업하는 것이 안전합니다. 나중에 commint되지 않은 정보 때문에 merge가 되지 않을 수 있기 때문이다.

merge 하는 방법은 아주 간단합니다.



step 1.

merge할 주체가 되는 branch로 checkout 한다.



step 2.

Merger>Local Merge 메뉴를 선택하면 현재 상태에서 merge가 가능한 branch들이 나타난다. 이 때 원하는 branch를 선택하면 현재 branch와 병합된다.

4. 원격 Repository 사용하기

원격 Repository를 사용하려면 git 계정을 등록할때 사용할 이메일을 등록해야 합니다. Edit>options에서 작업해 둡니다.

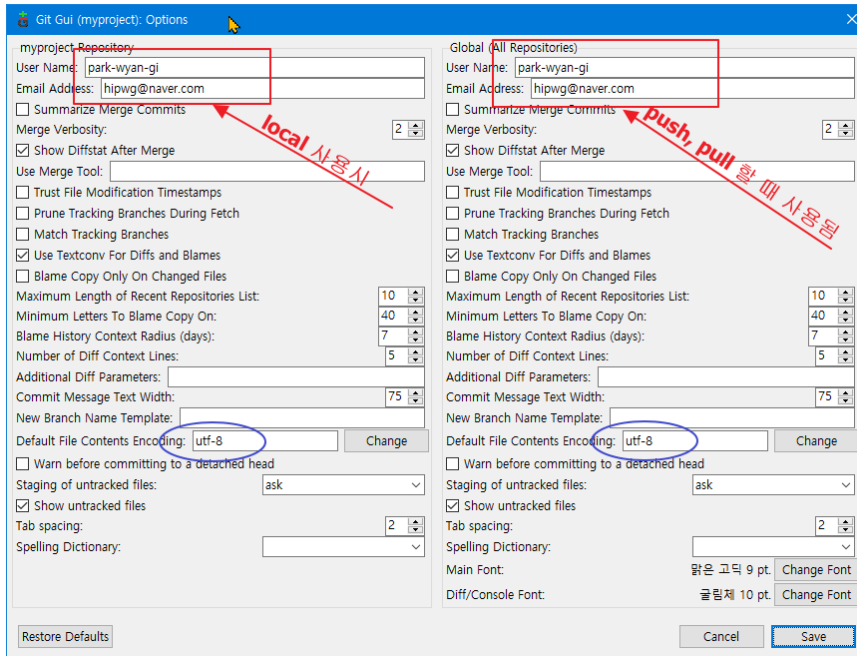


그림 21: 원격 repository 사용을 위한 이메일 등록

4.1. github 에 계정 트기

step 1.

<http://www.github.com> 사이트를 방문하여 사용자명, 이메일, 암호만을 입력하면 간단히 저장 공간을 할당 받을 수 있다. 물론 무료로 사용되는 공간은 저장 내용을 공개해야 하지만 약간의 금액을 지불하면 공개되지 않는 공간을 만들 수도 있다.

회원 가입 화면은 사이트가 리뉴얼된다면 위와 같은 화면이 다른 화면으로 변경되겠지만 현재(2020.09)까지는 위와 같은 화면이다.

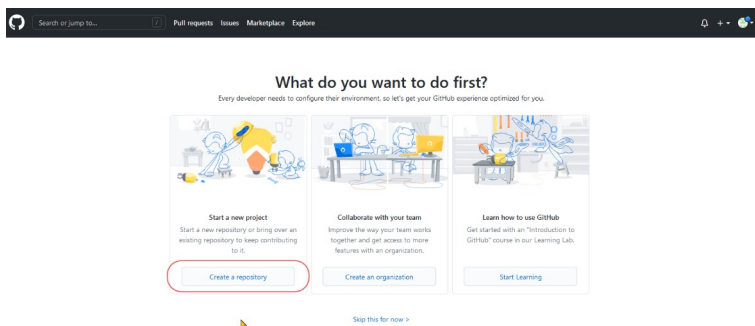
- Username : 영문자와 하이픈만 입력됨. 문자열 앞뒤의 하이픈은 안됨.

4.2. repository(저장소) 만들기

github에 소스를 저장하려면 가장 먼저 해야 할일이 repository를 생성하는 것이다.

step 2.

Start a project 버튼을 클릭하면 repository를 만들 수 있는 화면으로 이동하게 된다.






step 3.

적당한 Repository name(저장소 이름)을 정하고 [Create repository] 버튼을 클릭한다.


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 ^{*} 

Great repository names are  Your new repository will be created as `git-test`. [Learn about laughing-lamp?](#)

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.](#)

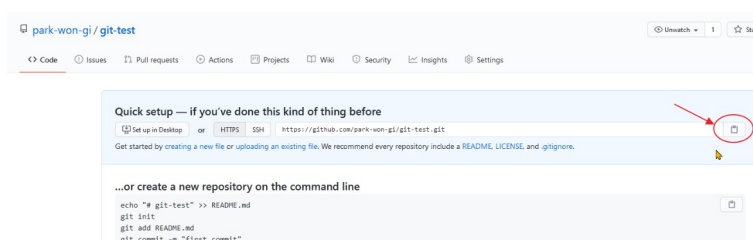
☐ Add .gitignore
 Choose which files not to track from a list of templates. [Learn more.](#)

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

[Create repository](#)

step 4.

다음 단계인 이클립스에서 github의 repository를 가져올 때 필요한 정보를 클립보드에 복사해 둔다. 아래의 그림처럼 아이콘을 클릭하면 쉽게 클립보드에 복사해 둘 수 있다

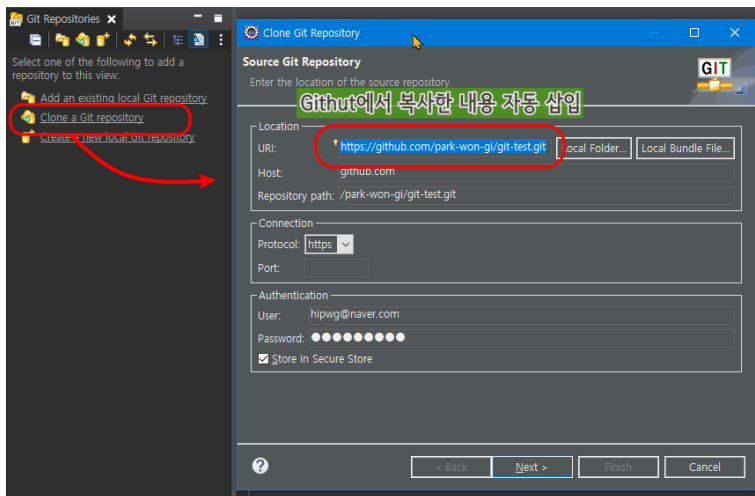
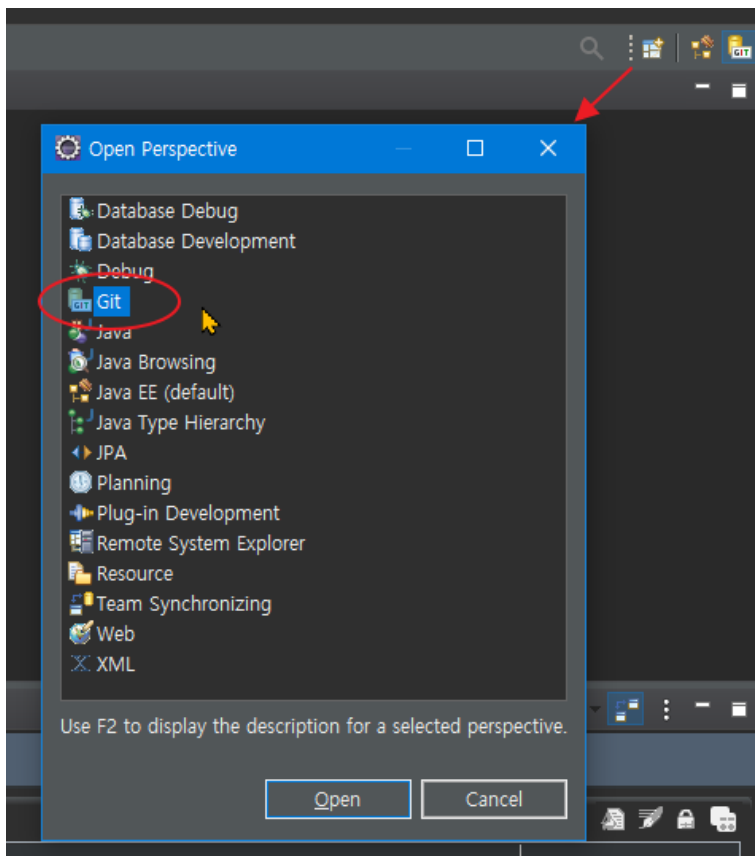


4.3. repository 정보를 이클립스로 가져오기(등록된 소스가 없는 경우)

eclipse EE버전인 경우 기본적으로 github API가 설치되어 있으므로 별도로 API를 설치할 필요는 없을 것이다.

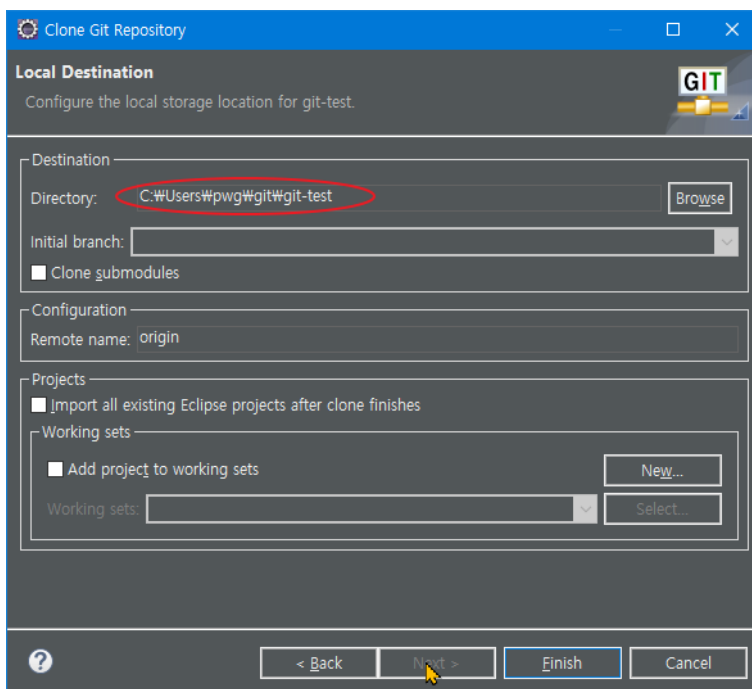
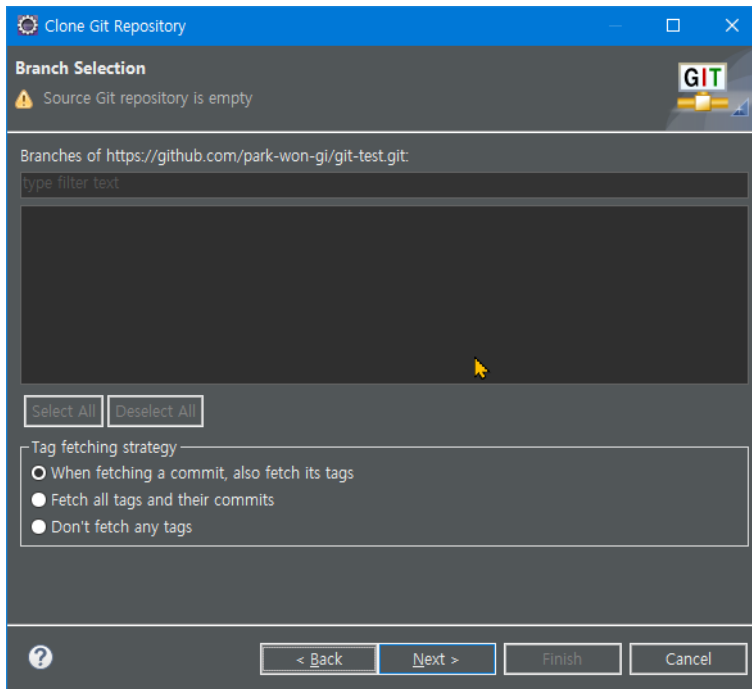
step 5.

Perspective 종류를 git로 변경 한 뒤 "Clone a Git repository" 링크를 클릭한다.

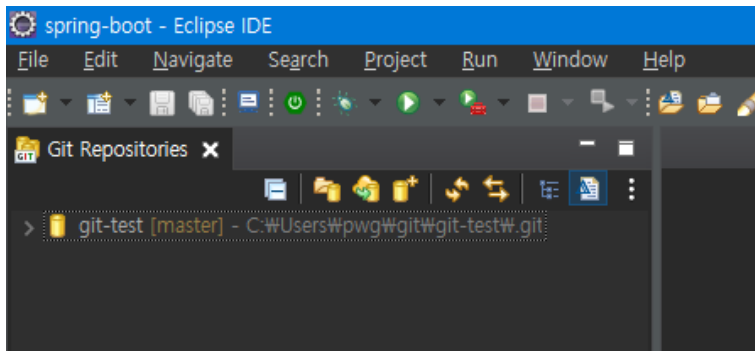


step 6.

repository에 파일들이 없기 때문에 처음엔 repository가 비어있다고 알려주고 다음으로 진행하면 github와 연동될 디렉토리를 지정하게 된다.

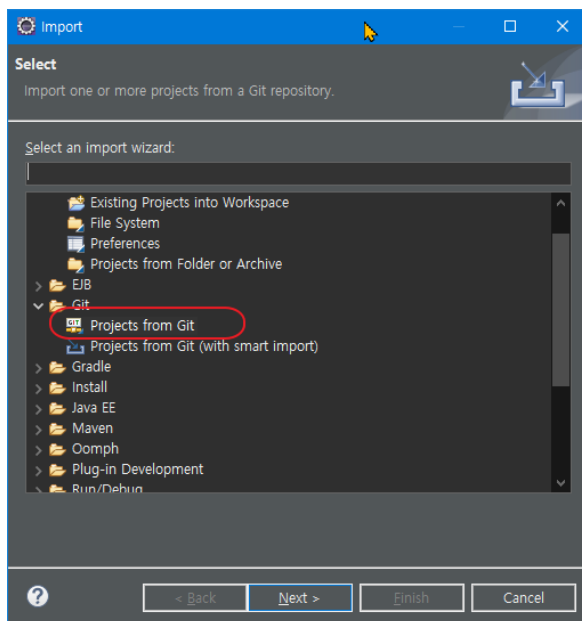


이클립스와 github의 repository가 연동된 화면이다.



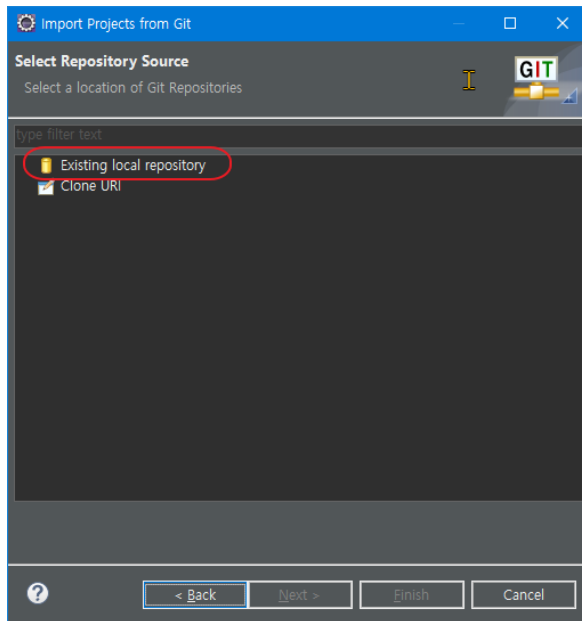
4.4. Git Project import 하기

이클립스와 Github와 연결 되었다고해서 소스를 편집할 수 있는 것은 아니다. 현재 Git Perspective에서는 Commit, Push, Pull 등 Git과 관련된 작업을 하는 곳이다. 다시, Java Perspective나 Java EE Perspective로 변경한 후 아래의 작업을 이어가야 한다.



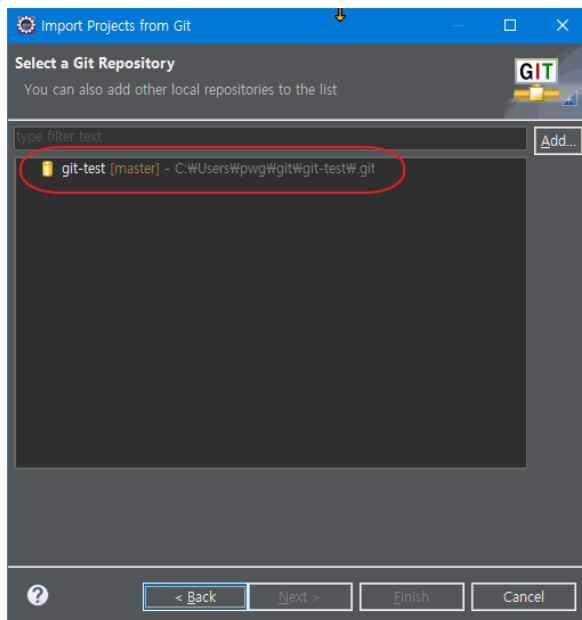
step 1.

이클립스의 Package Explorer에서 import를 실행한 후 Git>Projects from Git을 선택한다.



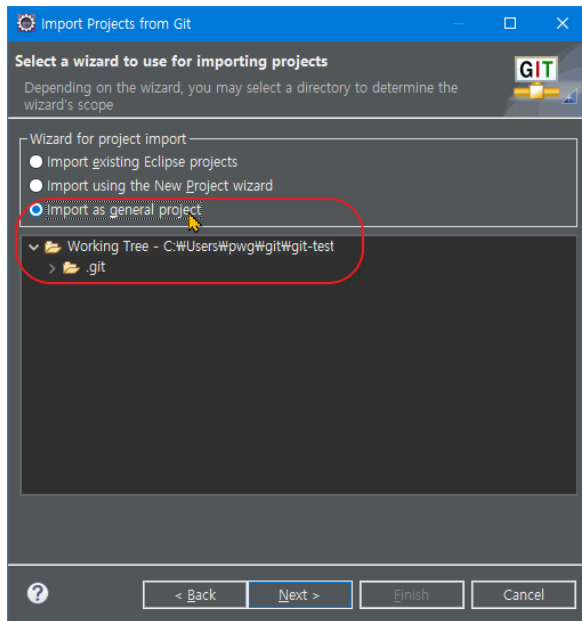
step 2.

Git Perspective에서 추가한 repository가 있기 때문에 첫번째 항목을 선택한다. 만약 추가한 repository가 없다면 Clone URI를 선택하여 추가할 수도 있다.



step 3.

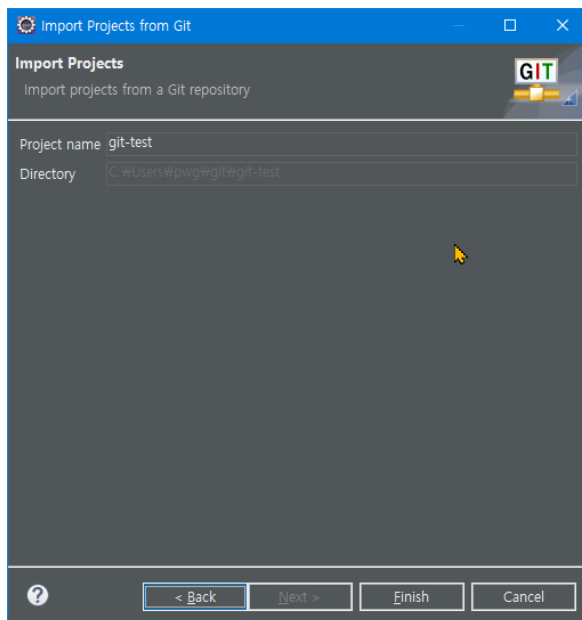
Git Perspective에서 추가한 항목을 선택한다.



step 4.

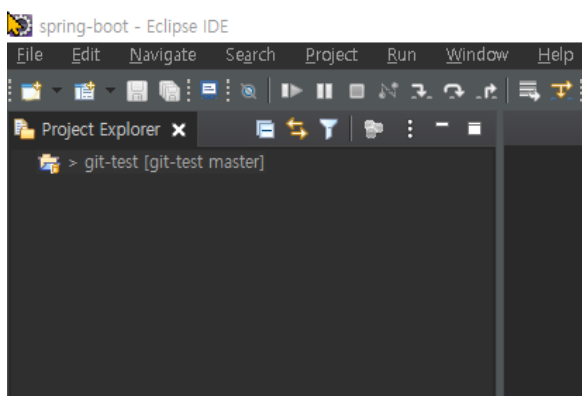
Git repository에 올려진 project의 종류에 따라 선택항목이 달라질 수 있다.

프로젝트의 유형이 특별히 없으므로 import as general project를 선택한다.



step 5.

최종 선택한 프로젝트의 내용을 보여준다. Finish 버튼을 클릭한다.

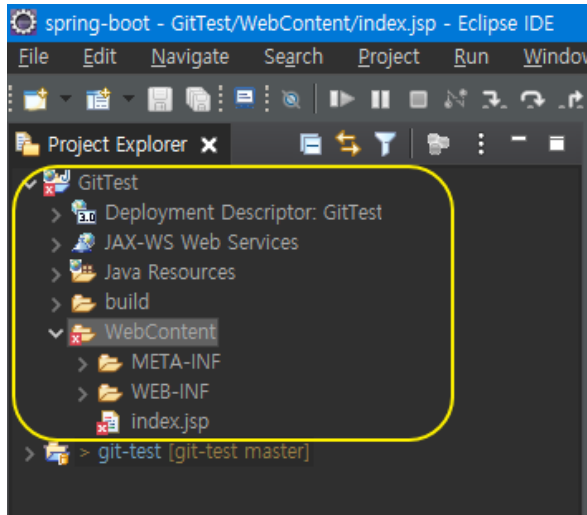


step 6.

Git Perspective에서 봤던 모양과 비슷한 프로젝트가 보인다. 다른 프로젝트와 다른점은 프로젝트명 앞에 '>' 표시가 붙어있는 것이다.

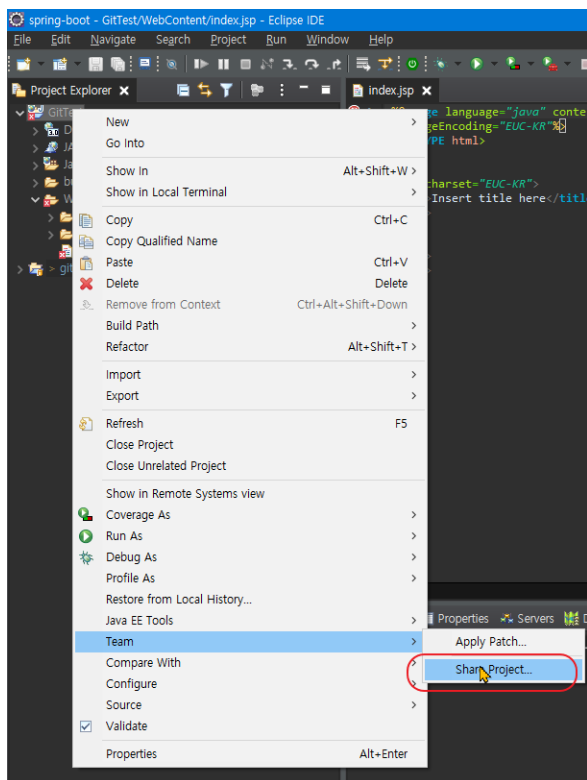
5. 이클립스 프로젝트를 repository에 생성하기

이번에는 반대로 이클립스에서 생성된 프로젝트를 github의 repository에 생성해 보도록 하자.



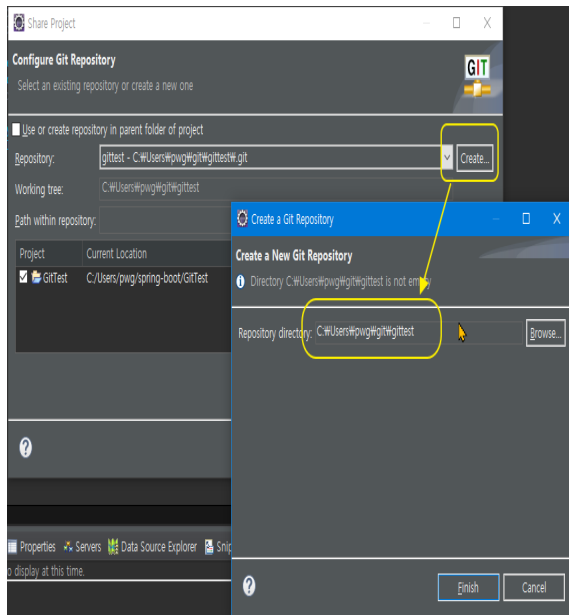
step 1.

이클립스에서 Dynamic Web Project 형식으로 GitTest 프로젝트를 생성한 뒤 index.jsp 페이지를 생성해 둔다.



step 2.

생성된 프로젝트명에서 마우스 우클릭하여 [Team] 메뉴안에 [Share project...]를 선택한다.



step 3.

로컬에 있는 프로젝트이기 때문에 아직 git repository가 로컬에 만들어지기 이전 상태이다. 따라서 Create 버튼을 클릭한 후 로컬영역에 git repository영역을 만들어주어야 한다.

step 3.

로컬공간과 github의 repository 공간과 데이터를 동기화 시키는 과정이라 볼 수 있다. [Create...] 버튼을 클릭하면 [step 6]에서 등록한 정보가 표시되며 해당 항목을 선택하면 된다.

작업이 정상적으로 마무리되면 이클립스 프로젝트 정보가 아래의 그림처럼 ">" 기호가 붙어 있어 프로젝트의 현재 상태를 알려 준다.

step 9.

공유가 되었으니 이제 프로젝트 현재 상태를 repository에 전송해 보자.

프로젝트 우클릭 → Team → commit 를 선택한다. 아래의 그림과 같이 "Commit Message"란에 적당한 내용을 입력하고 전송할 파일들을 선택한 후 [Commit and Push] 버튼을 클릭하면 된다.

설정된 기본값으로 계속 진행하면 된다.

step 10.

작업이 종료되면 github를 방문하여 "myproject" repository를 보면 아래와 같이 프로젝트가

등록되어 있음을 확인할 수 있다.

5.1. 프로젝트 commit 하기

이클립스에서 소스를 수정한 후 repository에 저장해 보자.

step 11.

index.jsp의 내용을 아래와 같이 간단히 수정 한 후 저장한다.

step 12.

프로젝트 마우스 우클릭 → Team → Commit을 선택한후 진행한다.

5.2. repository에 저장된 프로젝트 가져오기

github의 repository에 저장된 프로젝트를 가져와 보자.

step 13.

github 사이트를 방문하여 repository uri를 복사해 온다.

step 14.

이클립스의 git perspective에서 아래의 그림 처럼 작업한다.

다음으로 "master"를 선택한 후 진행한다.

step 15.

로컬 컴퓨터에 github내용을 저장할 디렉토리를 생성한 후 해당 디렉토리를 지정한다. 이때 디렉토리는 비어있어야 한다.

다음으로 기본 프로젝트를 import하는 것과 동일하지만 git로 부터 프로젝트를 import하면 된다.

5.3. 프로젝트 pull하기

다른 작업자에 의해서 수정된 정보를 github로 부터 가져 오기 위한 작업이다.

step 16.

프로젝트명 마우스 우클릭 → Team → Pull

5.4. 공동 작업자 추가하기

github에 등록된 유저들의 이메일을 등록함으로써 공동작업을 할 수 있다.

step 17.

github를 방문하여 공동 작업자를 추가할 repository를 선택한 후 [Settings]를 선택한다.

step 18.

[Collaborations] 를 선택하여 이메일을 입력하여 추가 한다.

5.5. 프로젝트 참여하기

8번에서 이메일로 참여자를 추가하게 되면 해당 이메일로 관련 정보가 전송된다. 이 때 해당 프로젝트를 승인하면 내 repository에 해당 프로젝트가 추가된다.

6. GUI Mode에서 pull 과 push

6.1. pull(bash console)

- remote repository에서 local repository에 파일들을 다운로드 하는 것.
- remote repository에 있는 저장소 이름과 동일한 폴더가 있으면 안됨.
- `git clone github 주소`

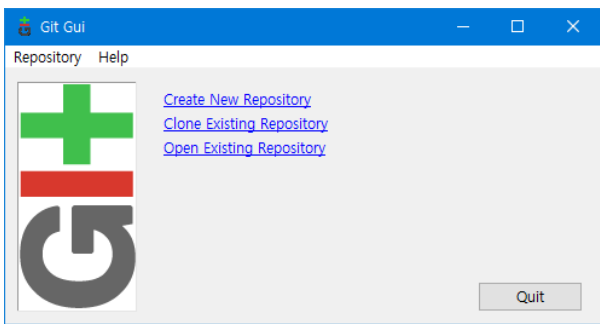
ex) hello 저장소에 있는 소스 다운로드

```
$ git clone https://github.com/park-wyan-gi/hello.git
```

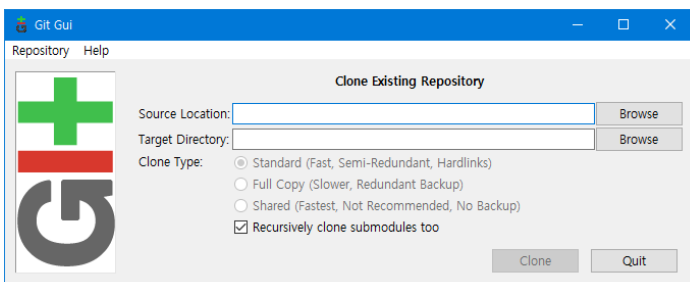
6.2. remote repository 가져오기

step 1.

Git GUI 창을 연 후 'Clone Existing Repository' 링크를 선택한다.



step 2.

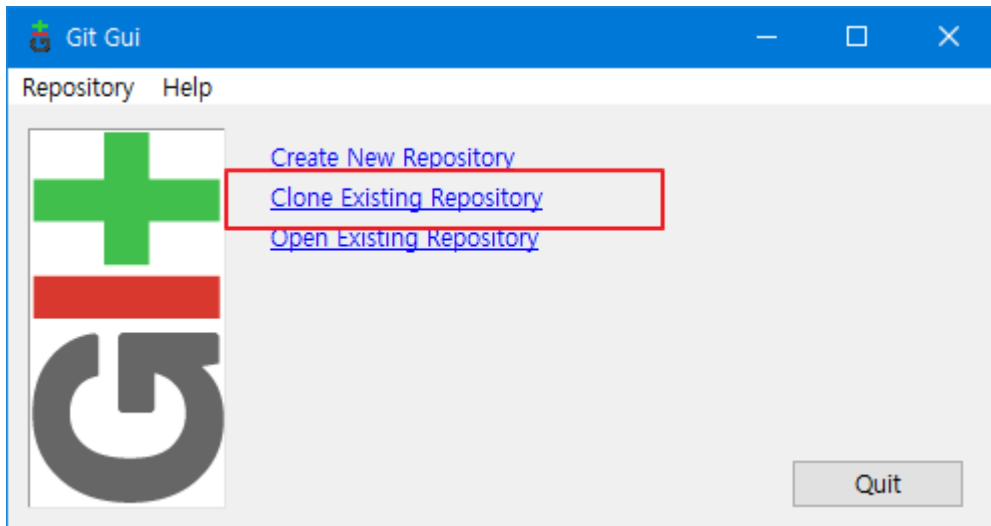


- Source Location 영역에 github의 주소를 입력한다.(ex: <https://github.com/park-wyan-gi/hello.git>)

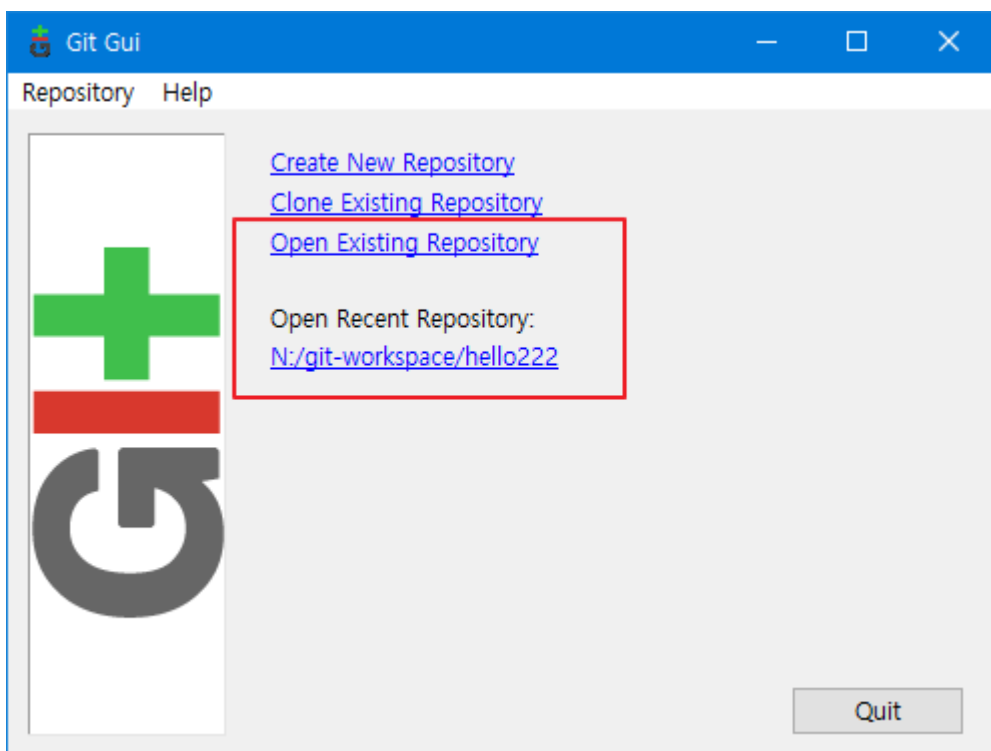
- Target Directory : github에 있던 내용이 담겨질 디렉토리이다. 이 때 선택된 Directory 명이 있으면 안된다. 따라서 최종 경로는 수작업으로 입력해 준다.(ex: [n:/git-workspace/hello](#) → hello는 수작업 입력. 디렉토리명은 어떤이름이든 상관없다.)

6.3. Git GUI사용 pull 하기

local repository가 없는 경우



local repository가 있는 경우



6.4. pull

github의 계정이 연결되어 있어야 함.

