

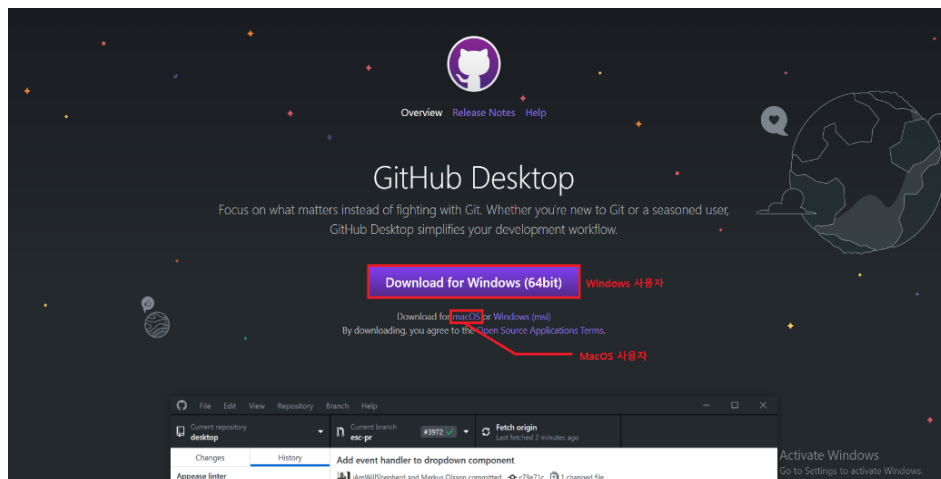
Getting Started with GitHub

GitHub Step1. GitHub Repository

- 1. GitHub Desktop 설치**
- 2. GitHub Repository 생성**
- 3. GitHub Repository 복제(Clone)**
- 4. GitHub Repository Commit**

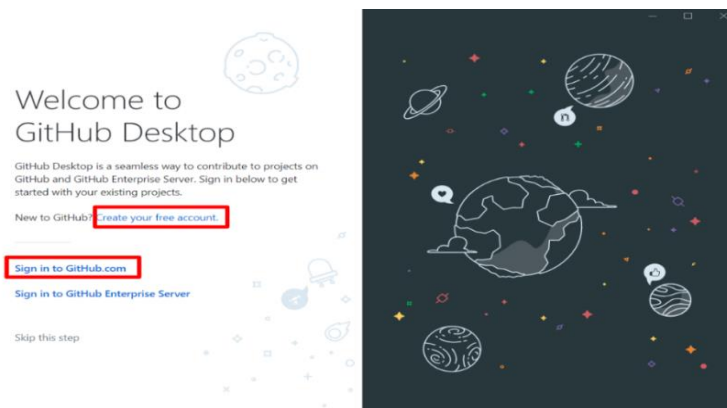
1. GitHub Desktop 설치

- **GitHub**는 버전 관리(**Version Control**)를 위한 코딩 호스팅 플랫폼입니다. 이때에 버전 관리란 동일한 코드 또는 정보에 대한 여러 버전을 관리하는 것입니다. 소프트웨어 개발에서 팀 단위로 개발 중인 소스 코드나 프로젝트를 관리하는 데에 사용됩니다. GitHub를 통해 세계의 여러 프로그래머들의 코드를 확인 및 자신의 소스로써 사용할 수 있습니다. 또한, 다른 프로그래머들의 코드를 편집할 수 있으며, 여러 프로그래머들이 협업을 하여 다양한 크기의 프로젝트를 진행할 수도 있습니다. **GitHub Desktop**은 GitHub을 자신의 개인 PC Desktop에서 쉽고 빠르게 상호 작용을 할 수 있도록 도와주는 무료 툴입니다. GitHub Desktop을 통해 GitHub을 이용한 작업을 간편하게 처리할 수 있습니다.
- 가장 먼저 <https://desktop.github.com>에 접속하여 자신의 PC(Windows 또는 MacOS)에 맞는 GitHub Desktop을 설치해줍니다.



(자신의 PC에 맞게 중앙의 설치 버튼이 변경됨)

- GitHub Desktop을 성공적으로 설치 완료하여 실행하면 시작 화면을 확인할 수 있습니다.



(GitHub Desktop 시작 화면)

- **“Create your free account”**을 클릭하면 GitHub의 회원가입 페이지가 나타납니다. 회원가입 페이지에서 자신의 GitHub 계정을 생성합니다. 만약에 GitHub에 계정이 미리 있다면 회원가입을 하지 않아도 됩니다. 자신의 계정을 통해 **“Sign in to GitHub.com”**을 클릭하여 GitHub에 로그인을 해줍니다.
- **(선택)** GitHub에서는 학생을 위한 무료 소프트웨어와 서비스 패키지를 제공하고 있습니다. 만약에 GitHub에 필요한 다양한 소프트웨어와 서비스 패키지를 무료로 받고 싶다면 <https://education.github.com/pack>에 접속하여 패키지를 받아줍니다.



[Home](#) [Students](#) [Student Developer Pack](#)

Learn to ship software like a pro.


There's no substitute for hands-on experience, but for most students, real world tools can be cost prohibitive. That's why we created the GitHub Student Developer Pack with some of our partners and friends: to give students free access to the best developer tools in one place so they can learn by doing.

Get your Pack

클릭



2. GitHub Repository 생성

- **Repository**는 하나의 프로젝트 및 코딩이 진행되는 데이터 저장소와 같습니다. Repository는 보통 폴더, 파일, 사진, 영상, 스프레드시트, 데이터 세트 등과 같이 프로젝트 진행에 필요한 정보를 하나의 저장소로써 보관하는 역할을 합니다.
- GitHub에서 Repository 생성 방법은 아래와 같습니다.
 1. 먼저 GitHub 페이지 상단의 **+** 버튼 클릭 후에 **"New repository"**를 선택해줍니다. 혹은 상단 Repositories 탭을 선택 후에 New버튼을 클릭해도 가능합니다. 방법은 다양하니 편한 방법으로 만들어 주시면 됩니다.
 2. **"Owner"**의 이름은 자신의 GitHub 계정의 이름으로 자동 설정되며, **"Repository name"**은 **"joyaix"**로 지정해줍니다. (모두를 위한 인공지능의 활용에서의 모든 과제는 해당 repository 안에서 진행되어야 합니다. 모인할 repository 이름은 "joyai"이며, "joyaix"는 joyai의 확장판(Extended), 즉 학생 개인의 repository를 의미합니다.)
 3. **Visibility**를 **"Private"**으로 지정하여 생성할 repository가 다른 사용자들에게 보이지 않도록 설정합니다. 자신을 포함하여 자신이 허용한 사용자들에게만 확인 가능하도록 할 때에는 **Private**을, 전세계의 사용자가 모두 확인할 수 있도록 설정할 때에는 **Public**을 선택 해주면 됩니다.
 4. **"Add a README file"**을 체크(선택)해주어 repository 생성과 동시에 README 파일도 자동으로 생성해줍니다. README 파일은 보통 해당 repository의 프로젝트를 설명하는 용도로 쓰입니다.
 5. .gitignore 파일에는 Git이 자동적으로 해당 repository에 추가 또는 업데이트를 진행하지 않을 파일들을 기록해줍니다. 사용자가 repository에 추가 또는 업데이트를 원하지 않는 유형의 파일들을 .gitignore 파일에 기록해줍니다.
 6. **[Create repository]** 버튼을 클릭하여 자신이 설정한 옵션에 맞게 repository를 생성해줍니다.

 Search or jump to...

1

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

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 *

returnb2b /

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

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



© 2021 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) 

[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

 Search or jump to...

1

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

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 *

returnb2b / joyaix ✓

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

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.](#)

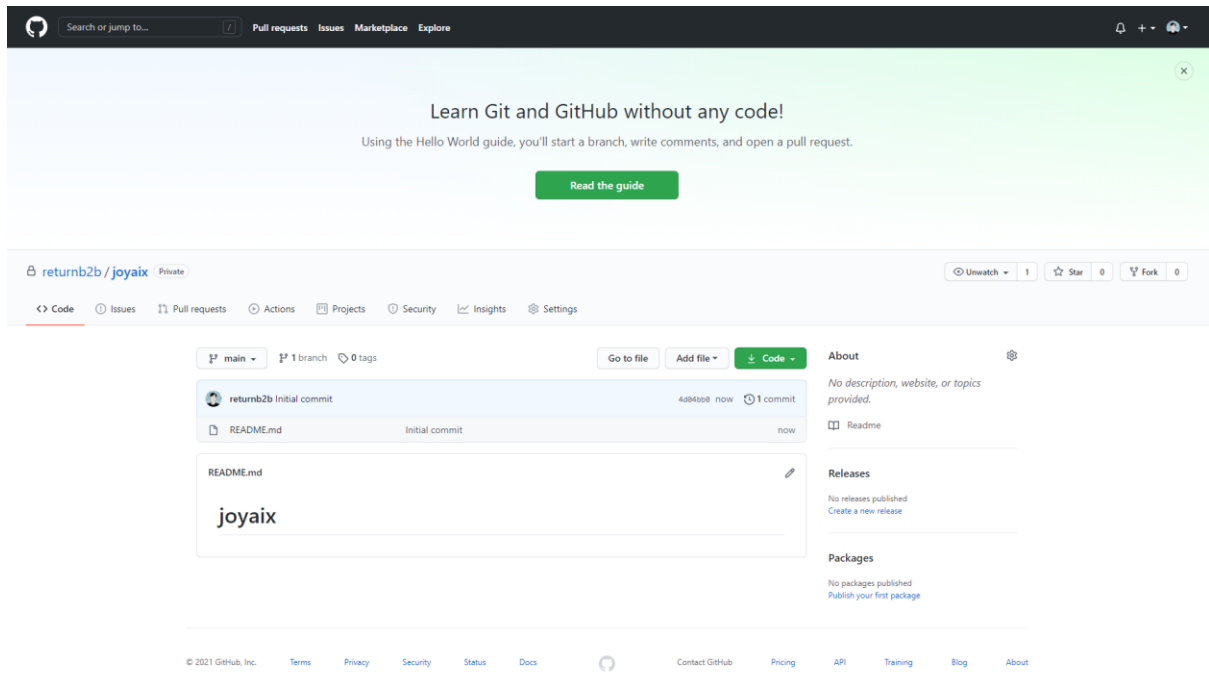
This will set [main](#) as the default branch. Change the default name in your [settings](#).

Create repository

© 2021 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) 

[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

- Repository 생성을 성공적으로 완료했다면 repository의 첫 페이지를 확인할 수 있습니다.



(생성한 repository의 첫 페이지 화면)

- GitHub의 repository 구성은 아래와 같습니다.
1. 클라우드(github.com)에서 사용자의 **GitHub 계정에 사용된 이름과 생성한 repository 이름이 생성되었음을** 확인할 수 있습니다. 해당 URL을 통해 자신의 repository로 바로 이동할 수도 있습니다.
 2. 해당 repository에 위치하고 있다는 표시입니다. **Repository 안의 다른 폴더로 이동할 때** 마다 /(일반 슬래시)가 추가되면서 해당 폴더의 Path를 나타냅니다. Path 안의 폴더들을 클릭하여 해당 폴더로 바로 이동할 수 있습니다.
 3. 현재의 repository에 존재하는 폴더와 파일 목록을 나타냅니다. 폴더 및 파일들의 추가 또는 업데이트된 시간을 확인할 수 있습니다. 현재는 README 파일만이 존재합니다. README 파일은 repository 페이지에서 한 눈에 확인할 수 있습니다.
 4. README 우측의 “연필” 모양 버튼을 클릭하여 **README 파일을 수정**할 수 있습니다. **README 파일은 마크다운(Markdown) 유형의 파일로써 사용 방법이 다른 일반 텍스트와는 다릅니다.** <https://gist.github.com/ihoneymon/652be052a0727ad59601>에 접속하여 마크다운에 대한 설명과 사용 방법을 확인할 수 있습니다. 수정 완료하였다면 하단의 **[Commit changes]** 버튼을 클릭하여 수정된 내용을 업데이트 해줍니다.

3. GitHub Repository 복제(Clone)

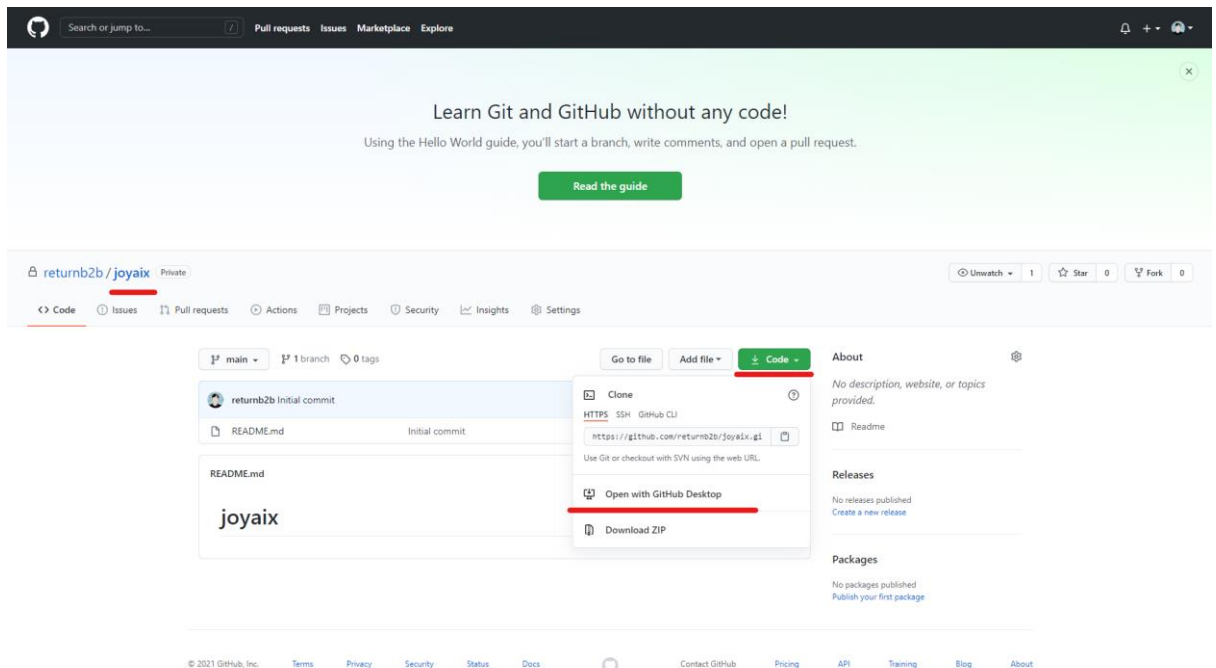
- 자신이 생성한 repository를 자신의 PC(Local Computer)에 복제하여 저장하는 기능은 GitHub에서 제공하는 유용한 기능 중 하나입니다. 자신이 생성한 joyaix 폴더(작업 폴더)를 개인 PC에 복제하여 저장하는 방법을 알아보겠습니다.

- GitHub에서 생성한 repository를 개인 PC에 복제하여 저장하는 방법은 아래와 같습니다.

1. GitHub에서 복제를 원하는 repository로 이동합니다.
2. 복제를 원하는 repository로 이동을 했다면 우측의 **[Code]** 버튼을 눌러준 후, **[Open with GitHub Desktop]**을 클릭해줍니다.

[Download ZIP] 버튼을 통해 repository에 있는 모든 파일들을 zip 유형의 파일로 다운로드 받아도 괜찮습니다. 다만, zip 유형의 파일로 다운로드 받게 된다면 repository 안의 파일들이 변경될 때마다 다시 다운로드를 받아주어야 하기에 번거롭습니다. 반대로 repository 자체를 그대로 복제하여 사용한다면 **GitHub Desktop** 툴을 사용해 업데이트가 훨씬 간편하고 쉬워집니다. 그래서 **[Open with GitHub Desktop]**을 사용하는 걸 권해드립니다.

이를 실행하기 위해서는 반드시 **GitHub Desktop** 설치를 해주어야 하니 주의해주시기 바랍니다.



3. 팝업 메시지가 등장하여 “GitHubDesktop.exe을(를) 여시겠습니까”라 묻는다면 **[GitHubDesktop.exe 열기]** 버튼을 클릭하여 응답해줍니다.

GitHubDesktop.exe을(를) 여시겠습니까?

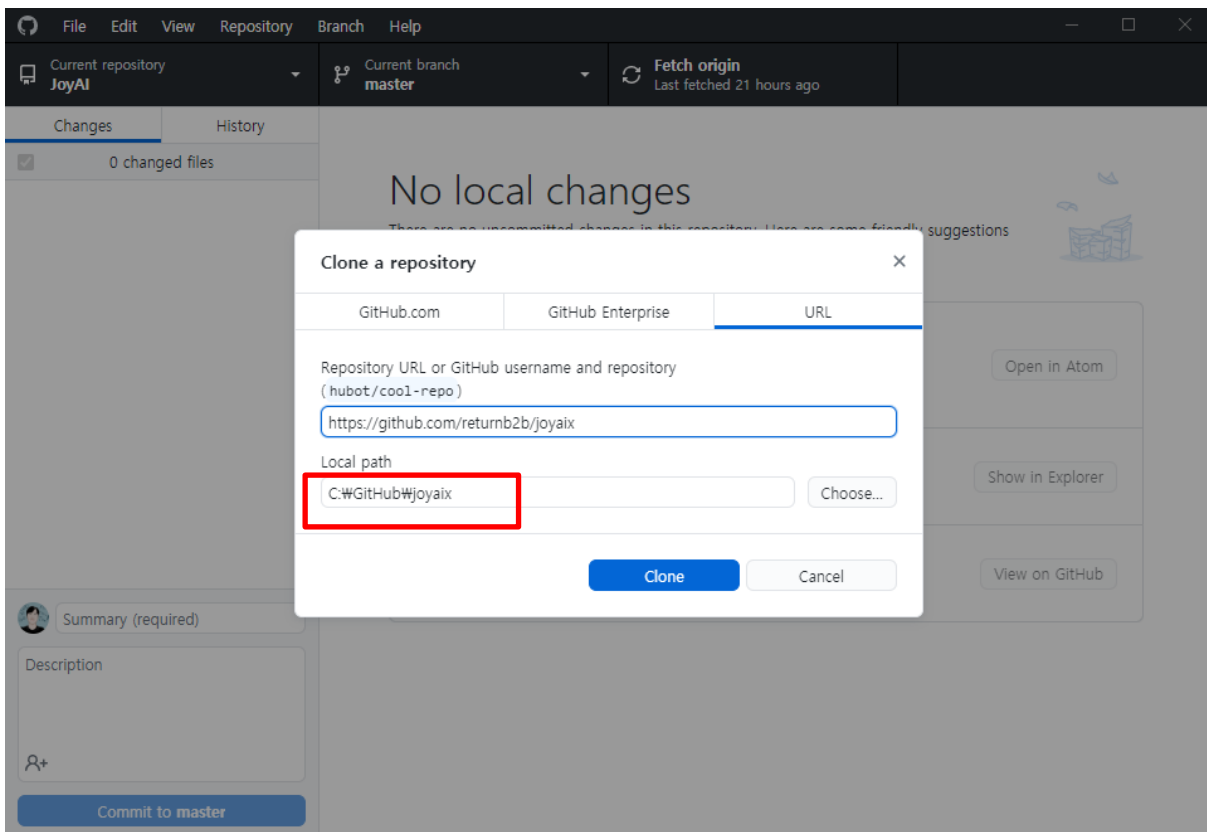
<https://github.com>에서 이 애플리케이션을 열려고 합니다.

☐ 항상 github.com에서 연결된 앱에 있는 이 유형의 링크를 열도록 허용

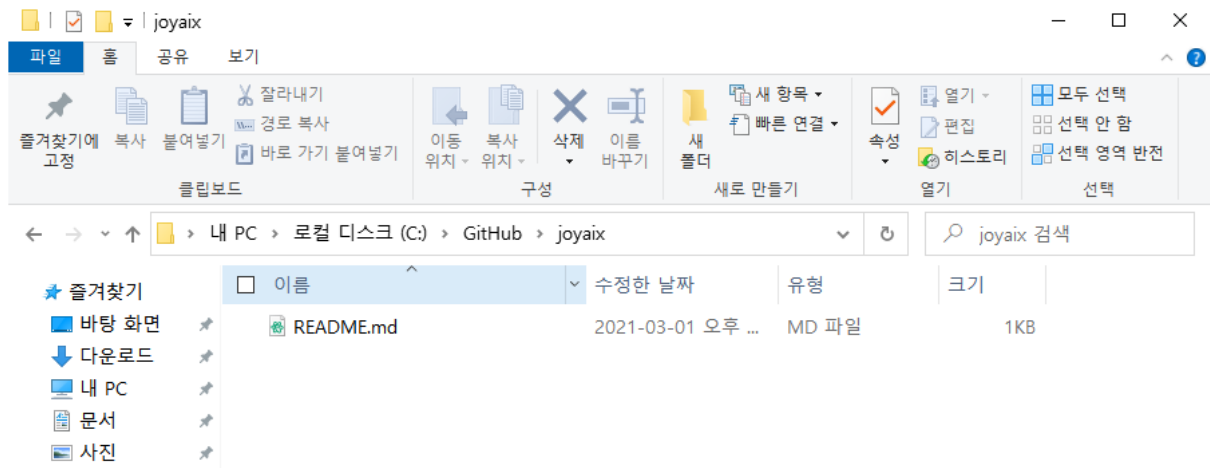
GitHubDesktop.exe 열기

취소

4. GitHub Desktop이 자동으로 열려 복제를 위한 설정 창이 나타납니다. 이때에 **“Repository URL or GitHub username and repository”**가 자신이 복제를 원하는 repository의 URL과 일치하는지 확인해주어야 합니다. **“Local path”**는 자신이 복제한 repository의 저장할 원하는 Path이며, **“C:\GitHubWjoyaix”**을 Path로 지정해줍니다. **C 드라이브 루트 폴더**에 저장하면 작업이 간편해지기에 **“C:\GitHubWjoyaix”**에 저장하는 것이 가장 좋습니다. 설정이 모두 완료되었다면 **[Clone]** 버튼을 눌러줍니다.



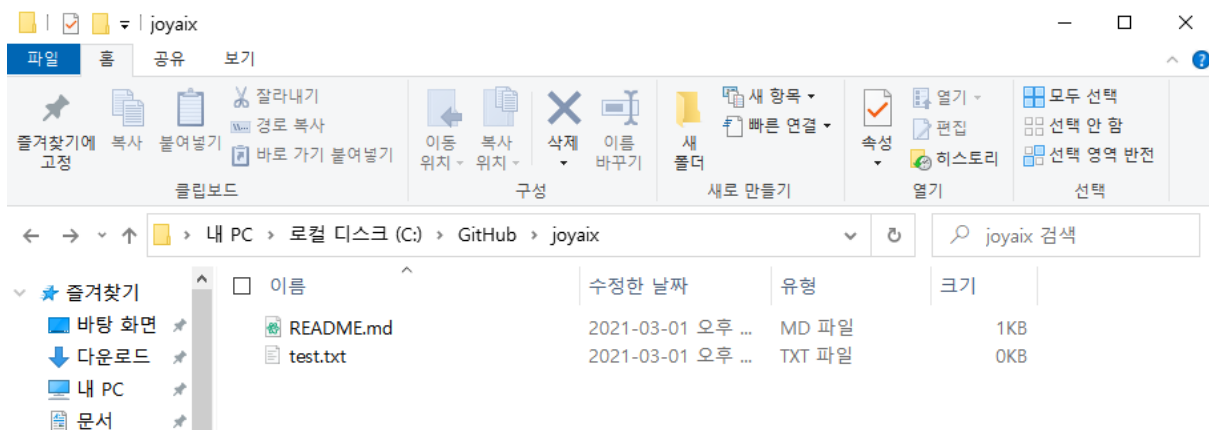
5. Repository를 저장한 로컬 Path로 이동하여 복제가 성공적으로 실행되었는지를 확인해줍니다. 본인의 github에 있는 파일 로컬 폴더에 다 있으면 성공입니다. 여기서는 README.md파일만 생성하였기 때문에 README 파일만 있으면 잘한 것입니다.



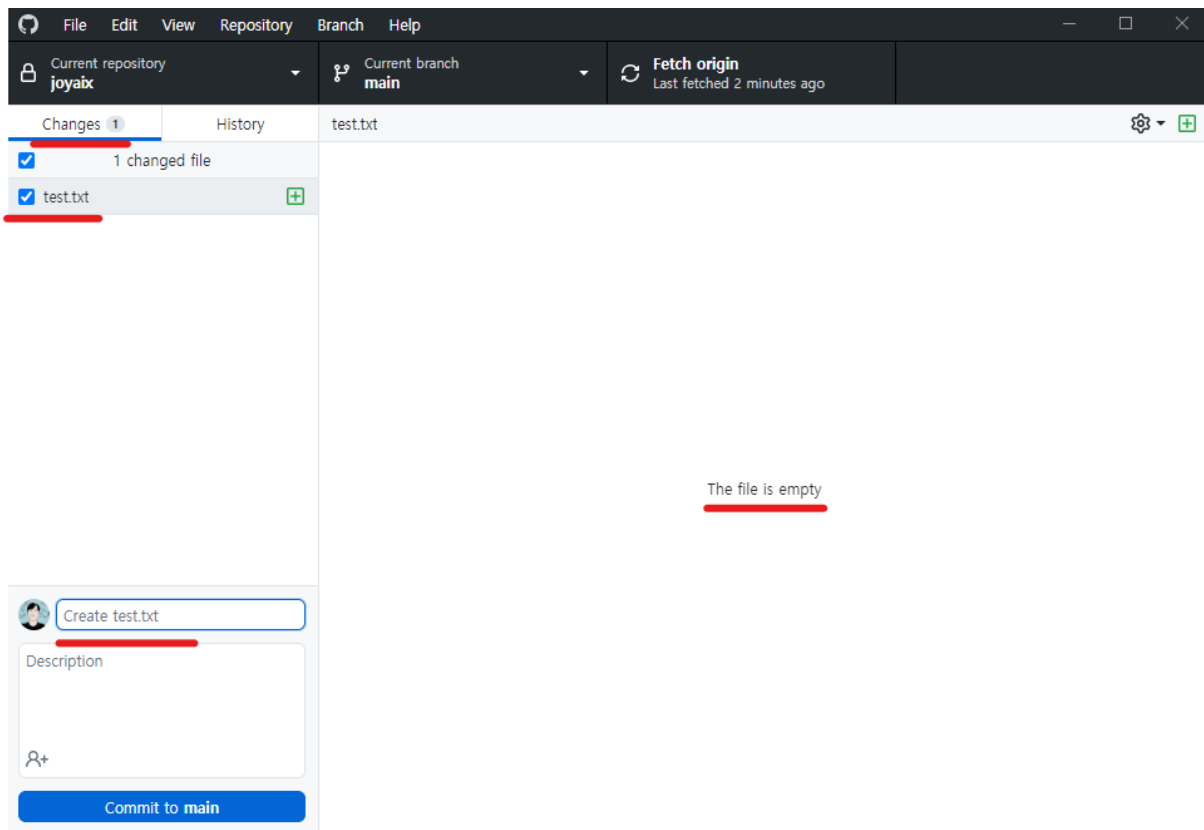
4. GitHub Repository Commit

- “Commit”이란 GitHub의 유용한 기능 중 하나로써 자신이 개인 PC에서 작업한 파일을 GitHub repository에 업로드하는 기능을 뜻합니다. 설치한 **GitHub Desktop**을 이용하여 쉽고 빠르고 **commit**을 실행해볼 수 있습니다.
- GitHub repository의 commit 방법은 아래와 같습니다.

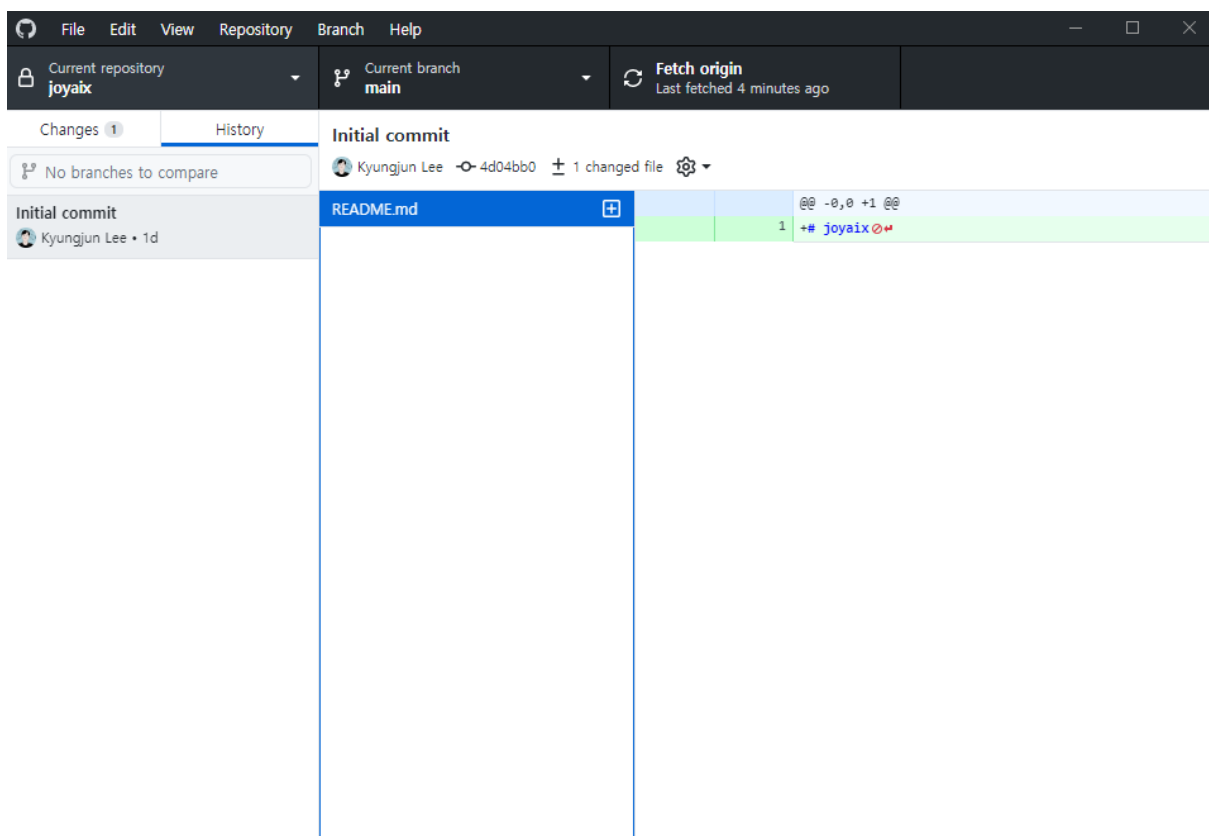
1. 먼저 **joyaix** 폴더로 이동하여 test.txt 파일을 하나 생성해 줍니다.



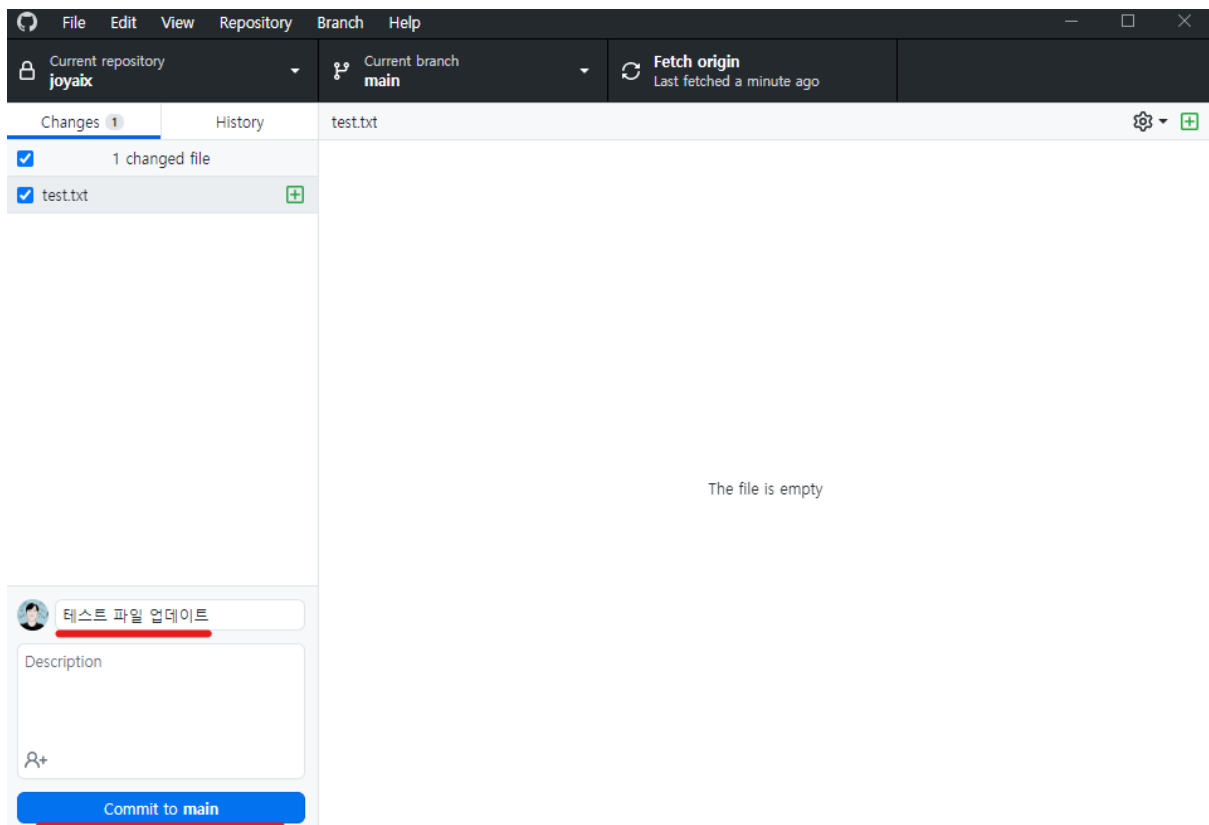
2. GitHub Desktop을 실행시켜줍니다. “**Changes**” 옆에 숫자①가 생겼으며, 생성한 “**test.txt**” 파일이 추가되었음을 확인할 수 있습니다. (체크되어 있어야 합니다.). 현재는 파일 안에 아무것도 입력하지 않아서 The file is empty라고 나오지만, 내용이 들어있으면 내용을 간략하게 보여줍니다.



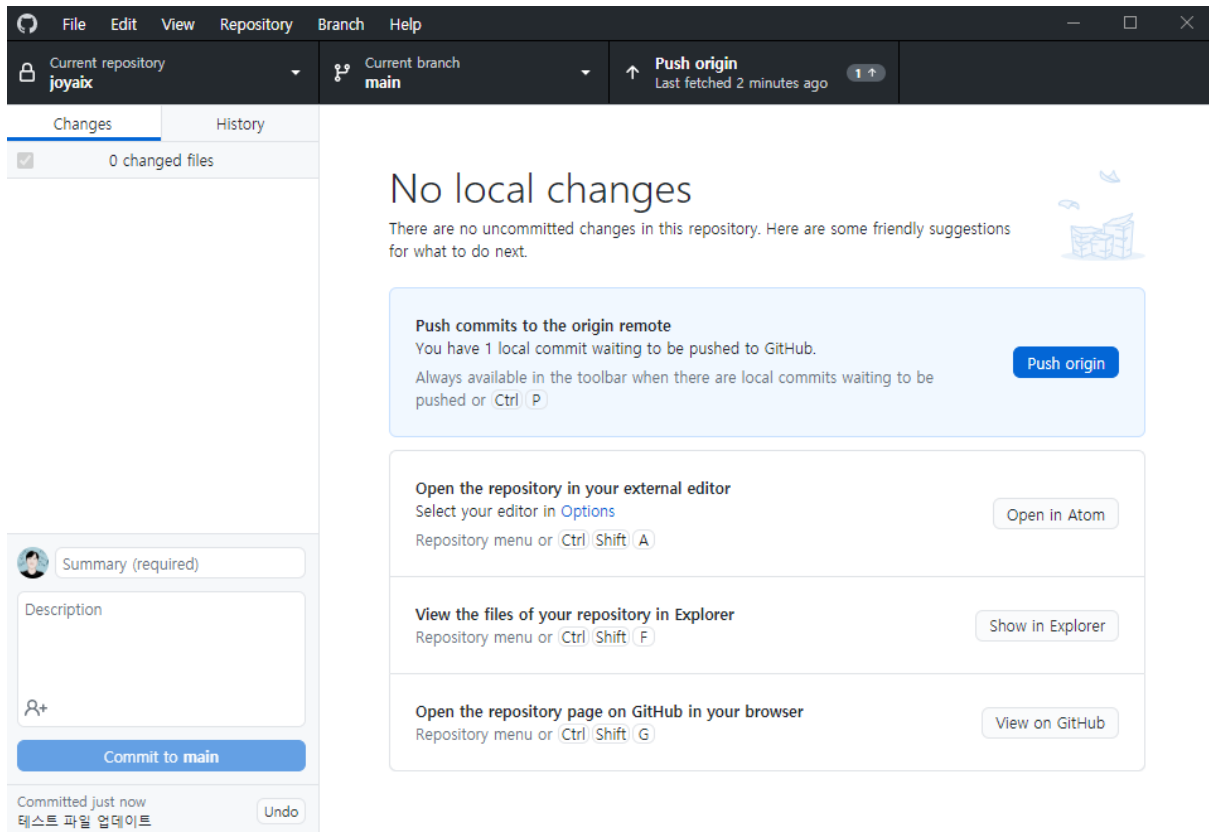
3. “Changes” 옆의 “History”를 선택하여 자신이 commit한 파일 이름, 메시지, 시간 등의 정보를 확인할 수 있습니다.



4. 다시 Changes 탭으로 이동하여 Commit 메시지를 입력한 후에 **[Commit to main]** 버튼을 눌러줍니다. Commit 메시지는 자신의 GitHub repository에 업데이트된 파일에 대한 간략한 소개를 나타냅니다. Commit 메시지는 새로 업데이트된 파일을 확인하기 위해 중요한 역할을 하며, 작성 법은 <https://djeeh.github.io/articles/How-to-write-a-git-commit-message-kor>에 접속하여 확인할 수 있습니다. 보통 업데이트 하는 파일이 어떤 파일인지 혹은 어떤 업데이트를 했는지 등의 정보를 적습니다.



5. Push origin 버튼을 눌러 GitHub에 Commit 해줍니다. (업로드 느낌)



6. 자신의 GitHub repository에서 새로 업데이트된 파일을 확인해줍니다.

