

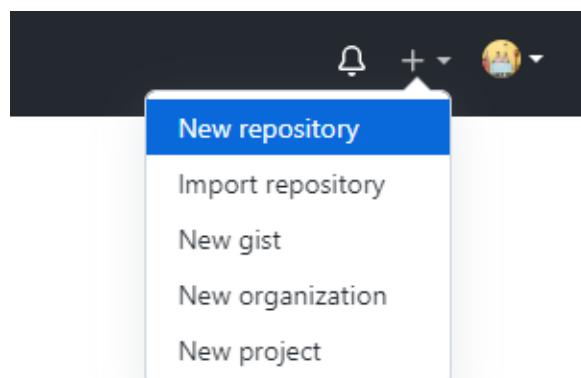
github

1. 웹에서 git repository 생성
 - 1-1. git repository create(생성)
2. 홈페이지에서 git repository에 파일 올리기
 - 2-1. github 홈페이지에 'Add file' 버튼 활용
3. Tool(fork)을 이용한 github
 - 3-1. Tool을 사용하는 이유
 - 3-2. Fork 설치
 - 3-3. 생성한 개인프로젝트 or 팀 프로젝트를 clone 하기.
 - 3-4. branch를 만드는 이유 & 간단한 용어
 - 3-5. develop 브런치 만들기 & 개인 브런치 만들기.
 - 3-6. commit → push → develop에 merge 까지
4. git bash로 github 다루기
 - 4-1. git upload
 - 4-2. git download
 - 4-3. git pull(변경사항 받기)
 - 4-4. git branch create
 - 4-5. git branch 확인하기
 - 4-6. git 특정 branch로 이동
 - 4-7. git branch merge
 - 4-8. git 특정 branch 생성하며 이동하기
5. git의 개념

1. 웹에서 git repository 생성

1-1. git repository create(생성)

1. git 로그인
2. 우상단 '+' 클릭 → New repository 클릭



3. 주소, 설명, 공개 설정(원하는대로) 후 Create repository 클릭

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 *

skybaer0804 / web4 ✓

Great repository names are short and memorable. Need inspiration? How about sturdy-broccoli?

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

2. 홈페이지에서 git repository에 파일 올리기

2-1. github 홈페이지에 'Add file' 버튼 활용

- 간단하지만, 단순한 파일 추가이므로 교재 자료 같은것을 올릴때 사용한다.

master 1 branch 0 tags

Go to file Add file Code

skybaer0804 Add files via upload

day01 commit2 1 hour ago

day01_1.txt Add files via upload 1 hour ago

day01_2.txt Add files via upload 1 hour ago

Create new file Upload files 3 commits

About web3_study

Releases No releases published [Create a new release](#)

Packages No packages published [Publish your first package](#)

Add a README

web3 /

Drag files here to add them to your repository
Or choose your files

Commit changes

Add files via upload

Add an optional extended description...

Commit directly to the master branch.

Create a new branch for this commit and start a pull request. [Learn more about pull requests](#).

Commit changes Cancel

3. Tool(fork)을 이용한 github

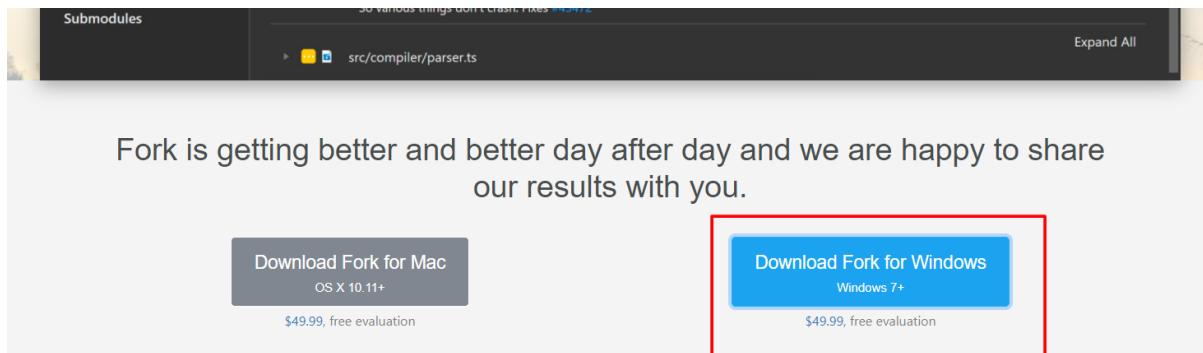
3-1. Tool을 사용하는 이유

- 실무에서는 Tool을 사용한다.
- 사용이 간편하다. 직관적이다.
- 초보자도 바로 사용한다.
- git을 잘 몰라도 필요한 최소한의 지식만으로 사용 가능하다.
- 그러나 이해도를 높이기 위해서는 git, 버전관리, git bash 사용법을 익히길 권장한다.
- github 가입이 기본이다.

3-2. Fork 설치

Fork는 git의 Tool중 하나이다. 무료버전으로도 충분히 사용가능하다.

<https://www.fork.dev/>로 들어가서 OS에 맞는 것을 다운받는다.

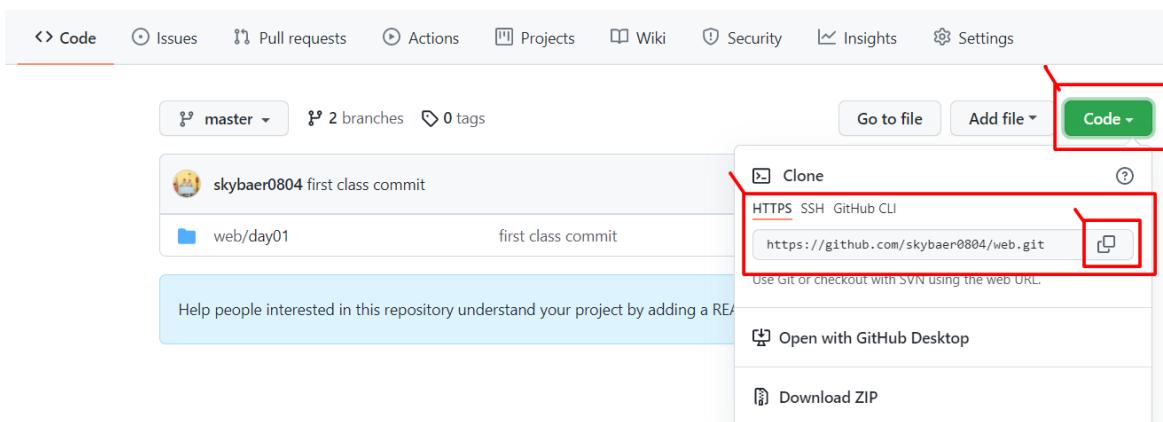


저는 window OS입니다.

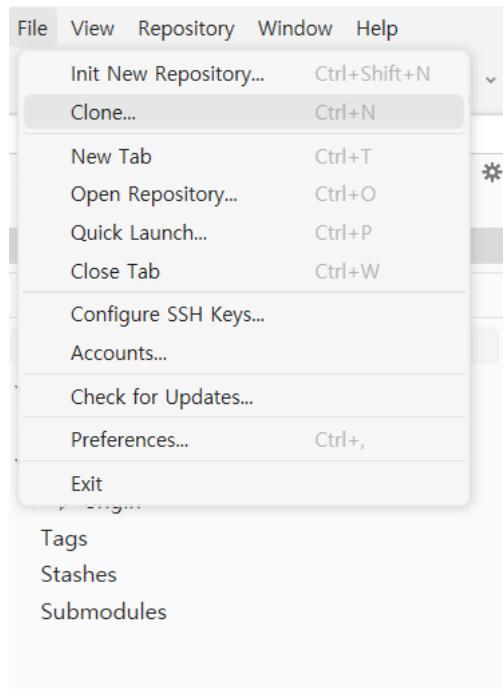
다운을 받은 후 설치 - 로그인한다.

3-3. 생성한 개인프로젝트 or 팀 프로젝트를 clone 하기.

1. github 웹에서 repository를 들어간다.
2. code → 우측 code 를 누르면 https, ssh 주소를 확인할 수 있다.
3. 우리는 https 주소를 복사해서 clone 할것이다.



4. Fork 프로그램으로 돌아와서 File → Clone 선택한다. 단축키 Ctrl + N



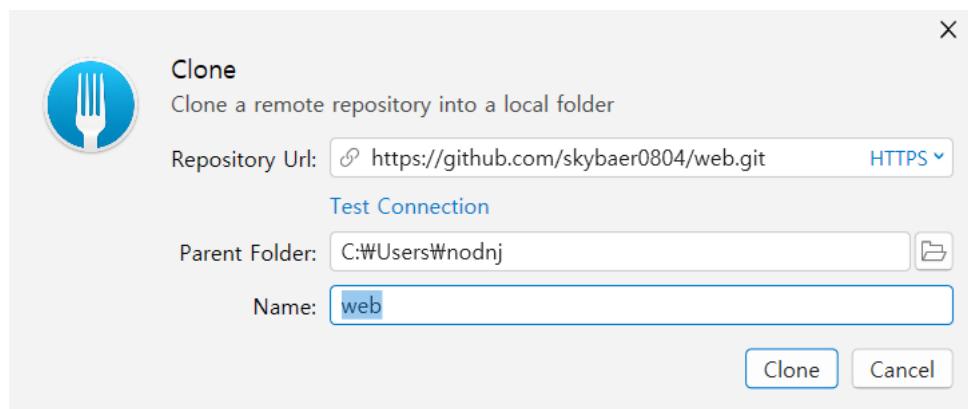
5. git에서 https를 '복사한 상태' 이면 아래 이미지와 같이 자동으로 정보가 기록된다.

만약 https에서 복사를 안한 상태라면 직접 입력해 주어야한다.

Parent Folder 주소 안에 git에서 받아온 파일들이 저장된다.

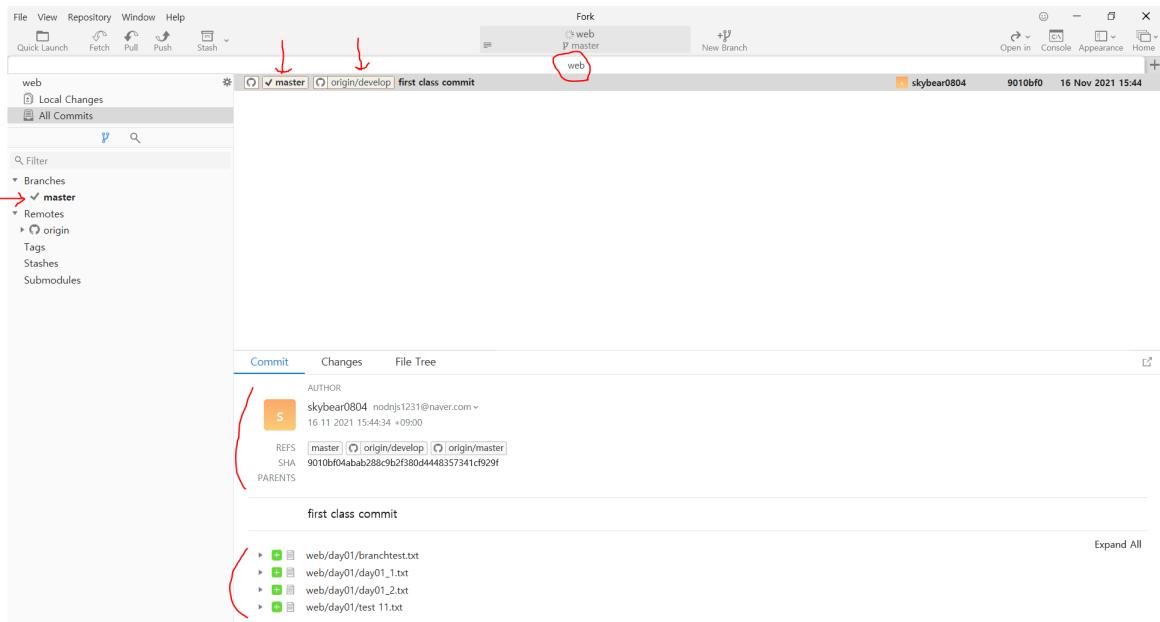
따로 폴더를 관리한다면 그 주소로 지정해주자.

Clone 을 누른다.



6. 내 PC로 프로젝트 파일이 잘 받아졌다!

화면 구성을 대략 살펴보자!



왼쪽에 Branch를 목록화 하여 보여주고, 중앙은 git branch 하나가 보인다.

하단은 선택한 branch의 파일 목록을 보여준다.

앞으로 협업을 하게되면 new branch를 따서 작업하고 merge하고 긴급수정(hotfix) 하고 merge하고 를 반복하게되고 나무의 가지같은 모습을 띄게 된다.

3-4. branch를 만드는 이유 & 간단한 용어

1. branch를 만드는 이유

버전 관리와 협업을 하기 위해서이다.

new branch는 새로운 branch를 생성한다는 뜻으로 해당 파일을 복제한다.

2. 버전 관리

만약 서비스를 하고 있는 프로그램(master; 운영파일)에 바로 수정을 한다고 생각해보자.

코드를 수정하는 동안은 프로그램은 제대로 작동 하지 않게 되고 서비스는 중단 된다.

잘못된 코드를 넣게 될 수도 있다. 그렇기 때문에

git의 branch를 통해 복사한 파일에 코드를 수정하고 테스트 해보고 잘된다면 master(원본파일 혹은 서비스가 운영하는 파일)에 합친다.

회사 혹은 팀마다 다르겠으나 git 버전관리를 어떻게 할 것인가에 대한 룰이 있다.

대표적으로 하나의 이슈(수정 및 개발사항)에 대해서 하나의 commit 혹은 push 한다.

→ 직관적으로 수정 사항, 업데이트 사항에 대해서 파악할 수 있다.

→ 만약 하나의 이슈에 대해 A라는 사람은 12번 PUSH, B라는 사람은 6번의 PUSH를 했다면 git의 가지는 branch는 꼬이고 꼬여서 누가 언제 무슨 내용으로 업데이트 했는지 찾기가 매우 어려울 것이다.

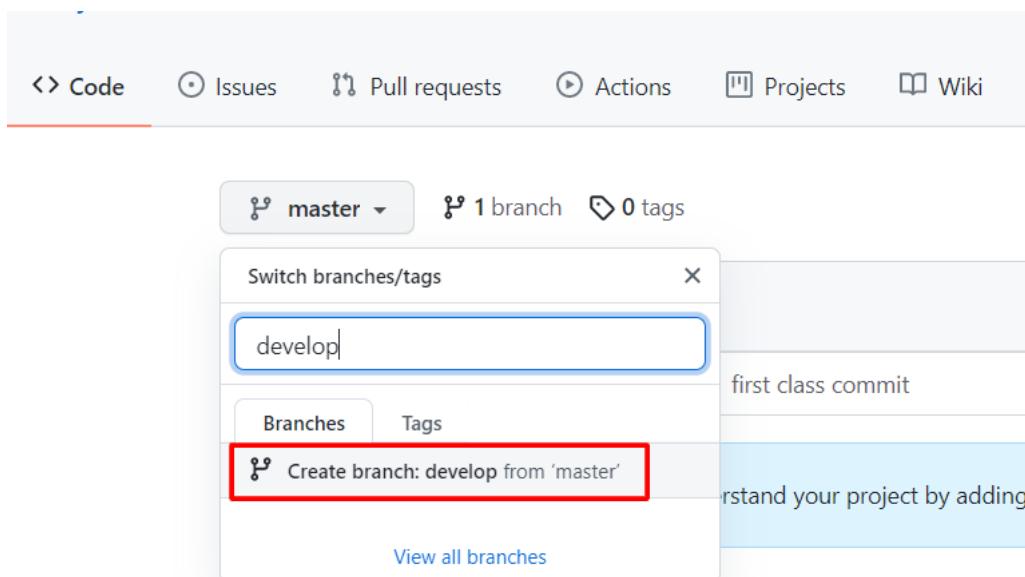
3. 간단 용어 설명

- branch : 원본에서 복제한 프로젝트 파일. 복사한 원본과 똑같은 파일을 갖는다.
- master : 최초 원본 파일(운영 파일을 의미)
- develop : 최초 원본 파일을 복사한 파일. (다음 버전 파일)
 - master에서 바로 new branch 하지 않는다.
 - hotfix(긴급한 수정) 일 경우에만 master에서 바로 new branch로 작업한다.
 - master에서 develop branch를 생성한 다음. develop에서 new branch하여 작업하는 것이 안전하고 관리가 용이하다.
- commit : 로컬 저장소(local repository)에 코드 변경 이력을 남긴다.
- local repository : clone 을 통해서 내 컴퓨터에 복제해둔 원격 저장소(remote repository)의 복사본을 의미한다.

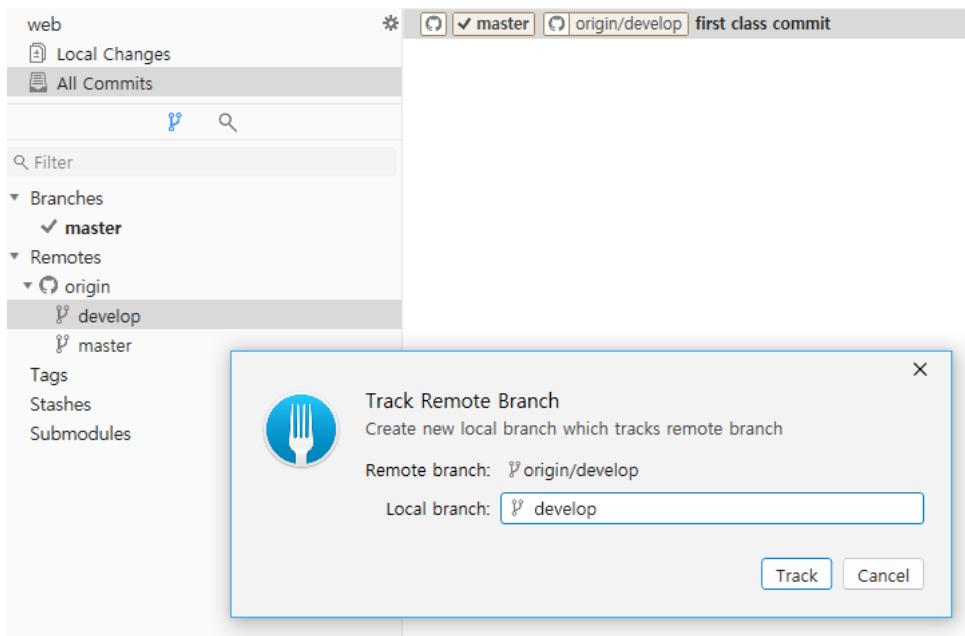
- push : commit 한 코드(그 동안 로컬 저장소에서 남겨놓은 코드 변경 이력들)를 원격 저장소로 전송한다.
 - push를 하게 되면 같은 팀 내의 사람들이 수정된 코드들을 볼 수 있게 된다.
 - 관리자가 내 코드를 리뷰해 줄 수 있다.
 - 단, push를 원본에 하면 안된다. 그것은 merge와 동일한데. 관리자의 리뷰를 생략하게 된다. git 관리 룰에 따라 다를 수 있다.
 - 순서는 commit → push → merge
- merge : branch를 병합한다. 즉, 합친다. 리뷰를 마친 후 최종적으로 merge를 함으로써 수정된 코드를 반영한다.
 - develop에서 new branch 하여 코드를 수정하고 리뷰를 마친 후 develop에 merge한다.
- pull : 변경된 것을 받는다. 최신화 한다.
 - master, develop은 각 개인의 merge를 통해 계속해서 코드가 수정된다.
 - 주의할 점은 내가 수정한 코드를 push하기 전 반드시 pull로 변경 사항을 반영한 후에 해야 한다.
그렇지 않으면 merge하는 순간 다른 사람의 수정한 부분과 내가 수정한 부분이 '충돌'하게 된다.
 - 이곳에서 설명하지 않은 (사실은 못한) 지식들은 구글링을 하자

3-5. develop 브런치 만들기 & 개인 브런치 만들기.

1. github 웹에서 develop branch를 만든다.

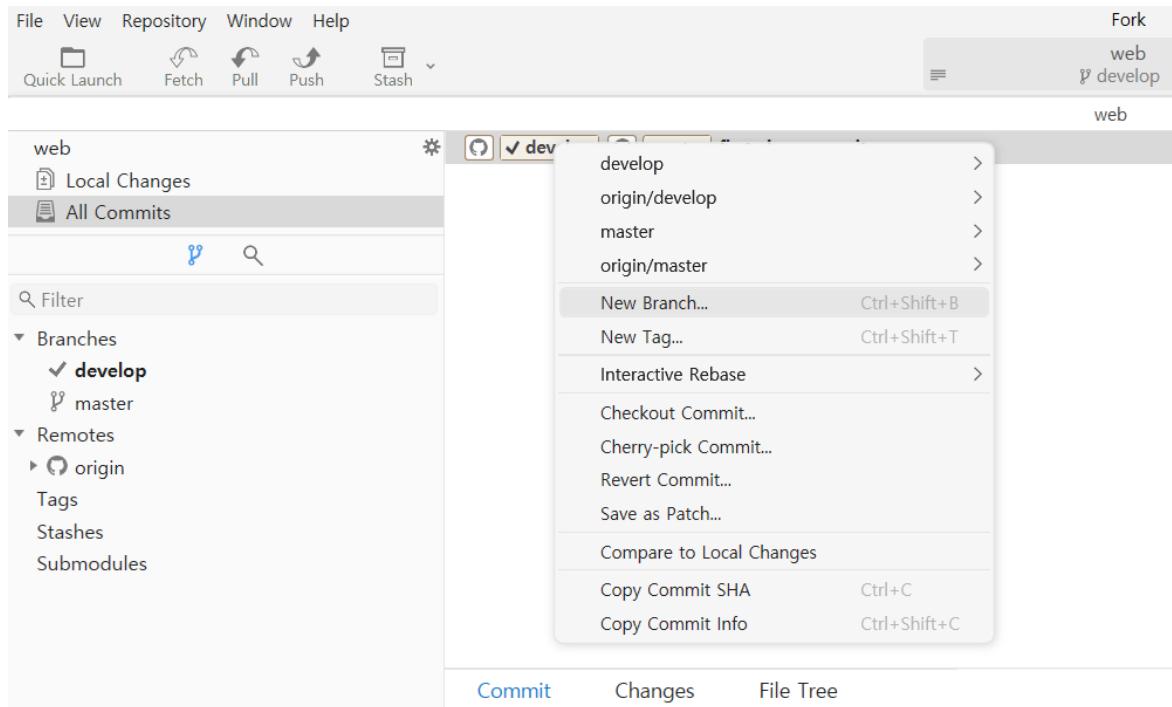


2. Remotes에 있는 origin/develop를 Local Branch로 가져옵니다.
왼쪽 브런치 목록에서 develop를 더블클릭 해서 Track버튼을 누릅니다.
Local branch 이름은 수정하지 않습니다. origin 이름과 동일하게 합니다.



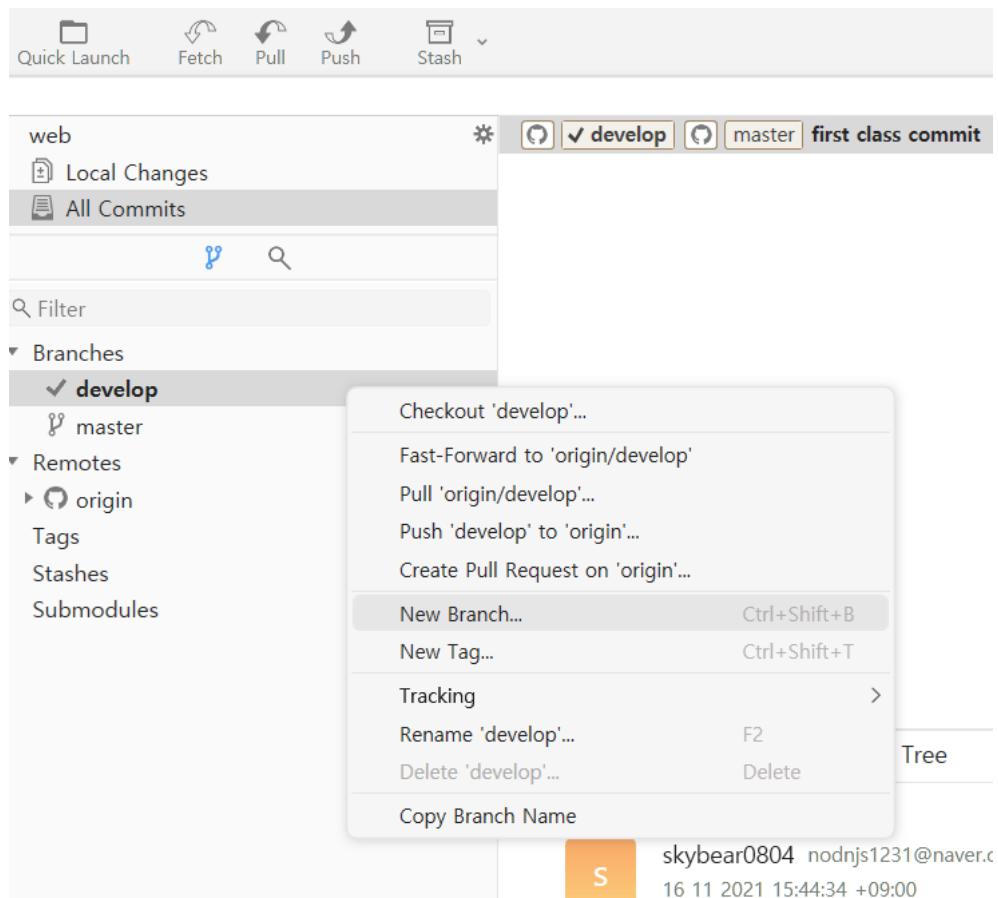
3. Fork에서 develop branch에서 new branch를 만든다.

하단 이미지 위치에서 우클릭으로 new branch 합니다.

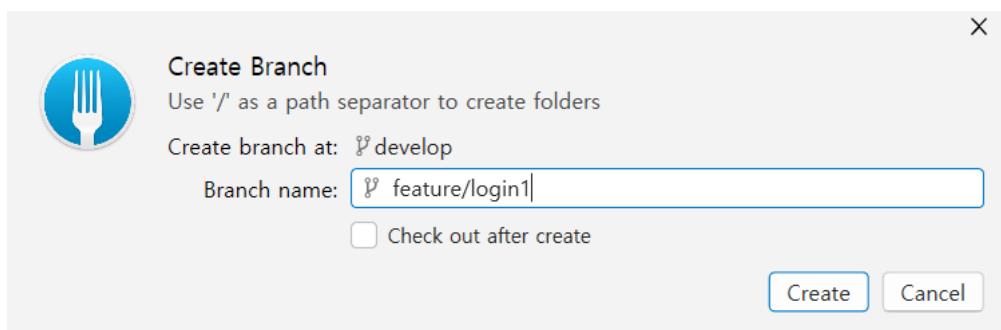


목록에 있는 브런치에서 우클릭해도 됩니다.

branch 이름 옆에 있는 'v' 표시는 현재 해당 Branch에 접속해있다. 참조한다 이런 의미입니다.



목록에서 우클릭 해서 new branch 해도 된다.



featrue 는 기능이란 단순 의미, login1 은 login 기능을 만드는 branch라는 뜻이다.

이렇듯 branch name도 직관적이고 또 팀에서 통일되게 지어야 한다.

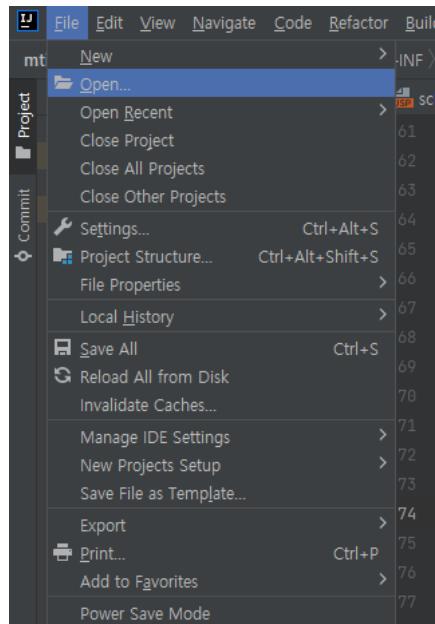
Create 버튼을 누르면 feature 폴더가 생성되고 그 안에 login1이라는 브랜치가 생성된다.

4. IntelliJ , Eclipse에서 branch 열기

IntelliJ 를 캡쳐했습니다. 다만 이클립스도 똑같은 개념입니다.

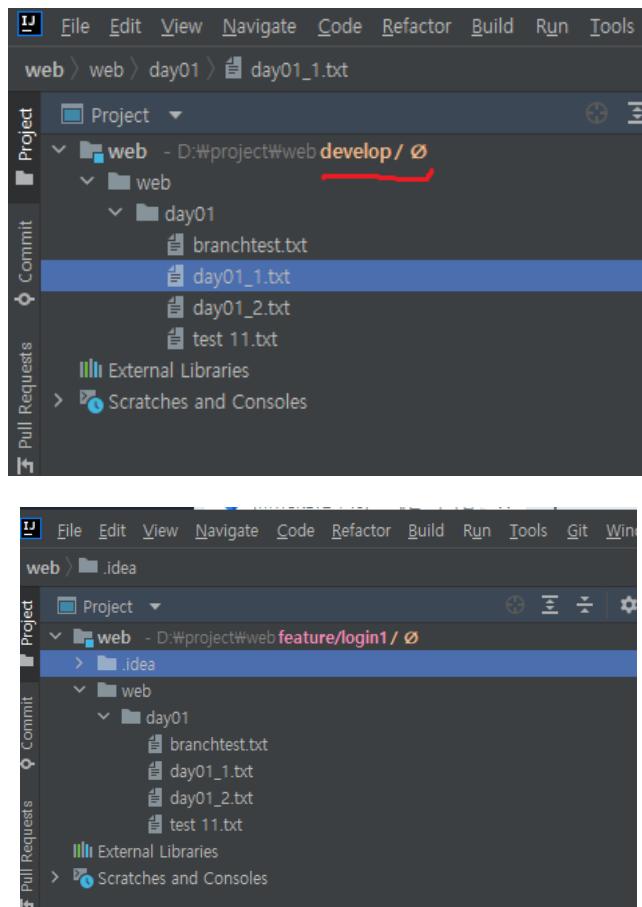
IntelliJ 혹은 eclipse 를 실행 → File → open → 프로젝트 폴더 선택

프로젝트는 clone 할때 Parent Folder 주소 안에 있습니다.



5. 내가 보고 있는 branch 확인하기

Fork에서 branch를 선택하면 IDE는 자동으로 해당 branch 코드를 가져옵니다.



3-6. commit → push → develop에 merge 까지

1. 코드 작성하거나 파일을 생성하는 등 '수정 작업'을 합니다.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>mergetest</title>
6  </head>
7  <body>
8
9      HTML파일을 생성해 내용을 적었습니다.
10     commit, push, develop에 merge 한 다음 master에 develop를 merge 해보겠습니다.
11
12 </body>
13 </html>

```

2. 변경 된 내용은 FORK 와 자동으로 연동됩니다.

수정 해보았습니다.

Local Changes 가 Local Changes (1) 로 바뀝니다.

한개의 수정 사항이 발생했다는 의미이고 눌러보면

Unstage에 수정한 파일명이 보입니다.

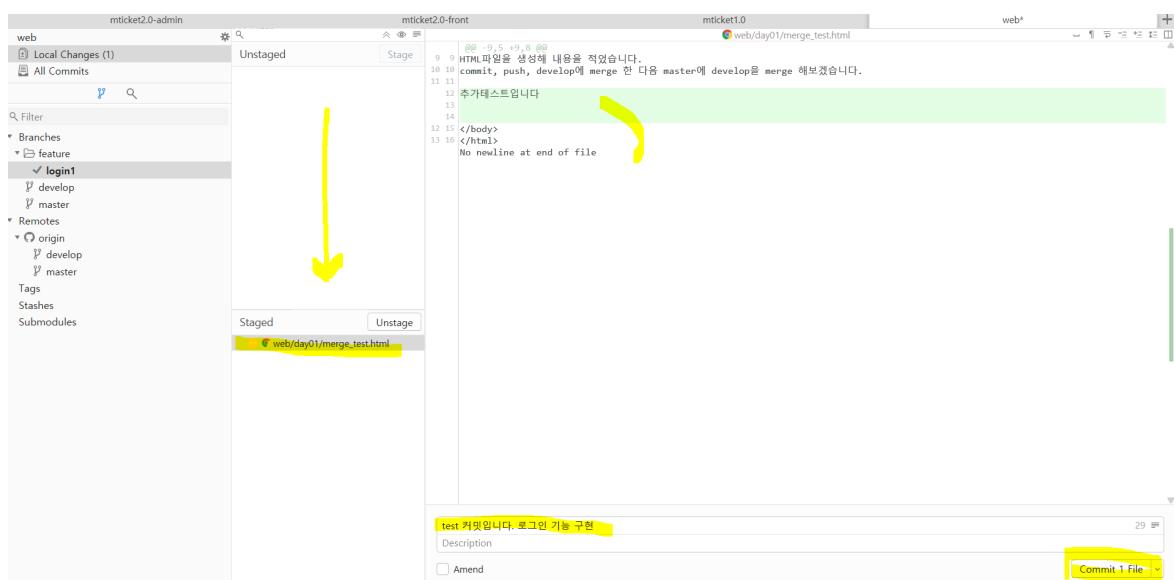
우측에는 수정한 파일의 상세 수정 코드들이 보입니다.



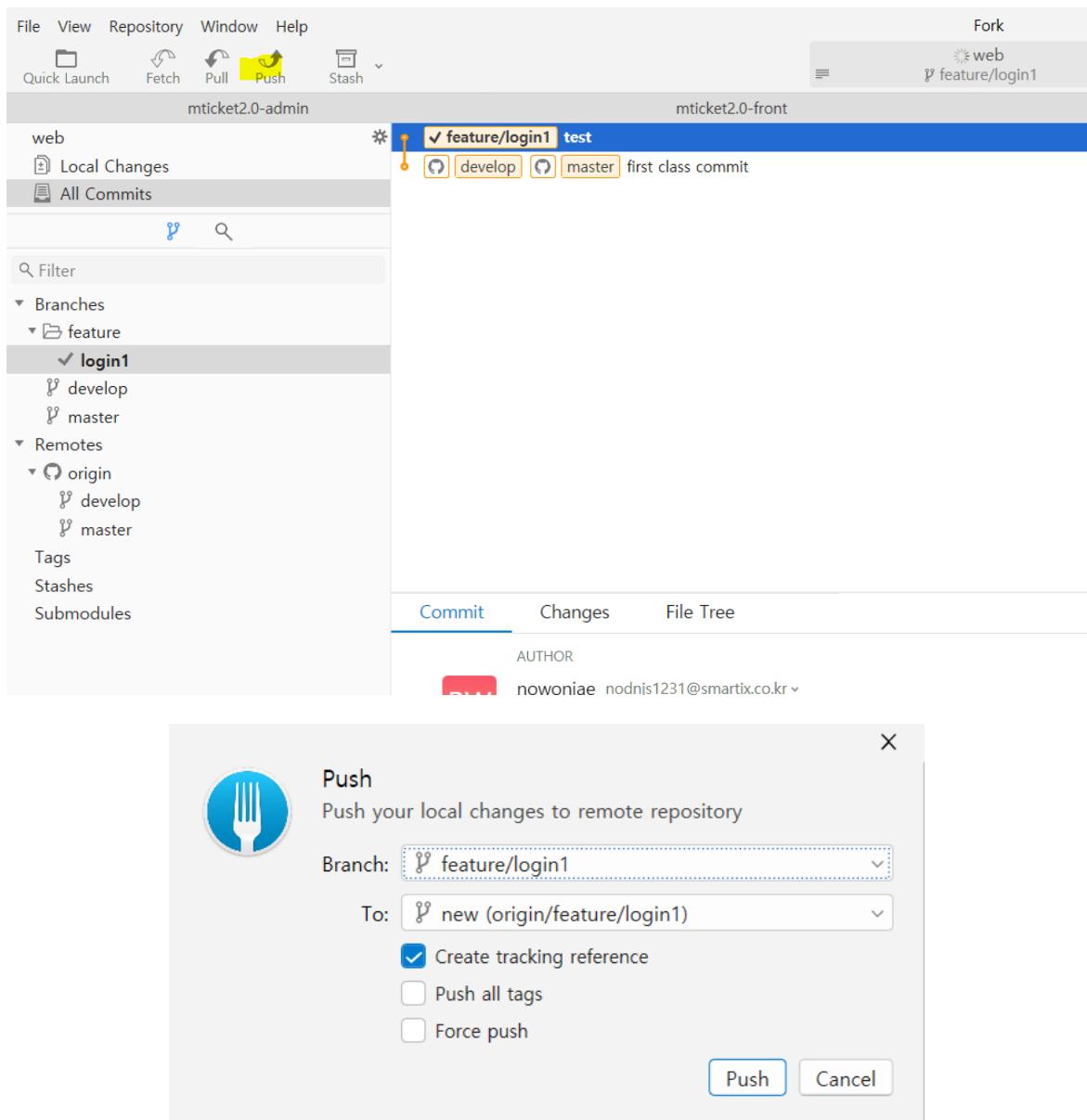
Unstage에서 Staged로 수정한 내용을 옮겨줍니다.

commit할 내용을 선택하는 작업입니다.

코멘트를 달아준 후 commit 버튼을 누릅니다!



3. 수정을 다했다면 FORK에서 PUSH를 누릅니다. 아래 이미지는 푸쉬 완료 된 상태입니다.



Branch : feature/login1 에서 To 로 PUSH를 하는데

기본 값 new (origin / feature / login1) 그대로 푸쉬 합니다.

다른 값을 누를 필요 없습니다.

푸쉬까지 됐다면 다른 사람이 내가 만든 브런치를 볼 수 있습니다.

내가 한게 맞는지 확인 받고 싶다면 팀원들에게 리뷰어를 받으세요!

- 변경 사항 받기는 Push 전에 해도 되고 후에 해도 상관은 없습니다.
- 다만 Push 전에 변경을 받고 하는게 좋습니다.
보기에도 깔끔하고 다른 팀원이 편하게 보게 하기 위함.
- 단 develop에 merge 하기 전에는 변경사항을 받고 머지 해야합니다.

4. Local Changes에서 내가 수정한 코드를 바로 삭제할 수 있습니다.

그럴 경우 원래 코드로 돌아옵니다.

```

@@ -9,5 +9,8 @@
9 9 HTML파일을 생성해 내용을 적었습니다.
10 10 commit, push, develop에 merge 한 다음 master에 develop을 merge 해보겠습니다.
11 11
12 12 추가테스트입니다.
13 13
14 14
15 15
16 16

```

No newline at end of file

5. push한 브랜치를 develop에 머지합니다.

github 홈페이지에 들어가 해당 repository에 Pull requests를 들어가면 push한 브랜치가 보인다. 우측에 Compare & pull request를 누른다.
안보이면 push를 안 누른 경우가 많다. 다시 Fork에서 push를 누르고 오자.

Label issues and pull requests for new contributors
Now, GitHub will help potential first-time contributors discover issues labeled with [good first issue](#)

[Compare & pull request](#)

Filters ▾ Labels 9 Milestones 0 New pull request

0 Open 4 Closed

There aren't any open pull requests.

해당 브랜치를 develop를 선택하고 우 하단에 Create pull request를 눌른다.

[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: develop ▾ ← compare: feature/test3 ▾ ✓ Able to merge. These branches can be automatically merged.

 test3

Write Preview H B I ≡ <> ⌂ ≡ ≡ ☑ @ ⌂ ↵

Leave a comment

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

Create pull request ▾

 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

 **test3** #6
skybaer0804 wants to merge 1 commit into `develop` from `feature/test3` 

 **skybaer0804** commented now   

No description provided.

 **test3** 

Add more commits by pushing to the `feature/test3` branch on [skybaer0804/web](#).

Merge pull request

This branch has no conflicts with the base branch

Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Leave a comment

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

test3 #6
skybaer0804 wants to merge 1 commit into `develop` from `feature/test3`

test3

8ec1509

Add more commits by pushing to the `feature/test3` branch on `skybaer0804/web`.

Merge pull request #6 from skybaer0804/feature/test3

test3

Confirm merge Cancel

Write Preview

H B I E ↵ ↶ ↷ ↸ ↹ ↻

Leave a comment

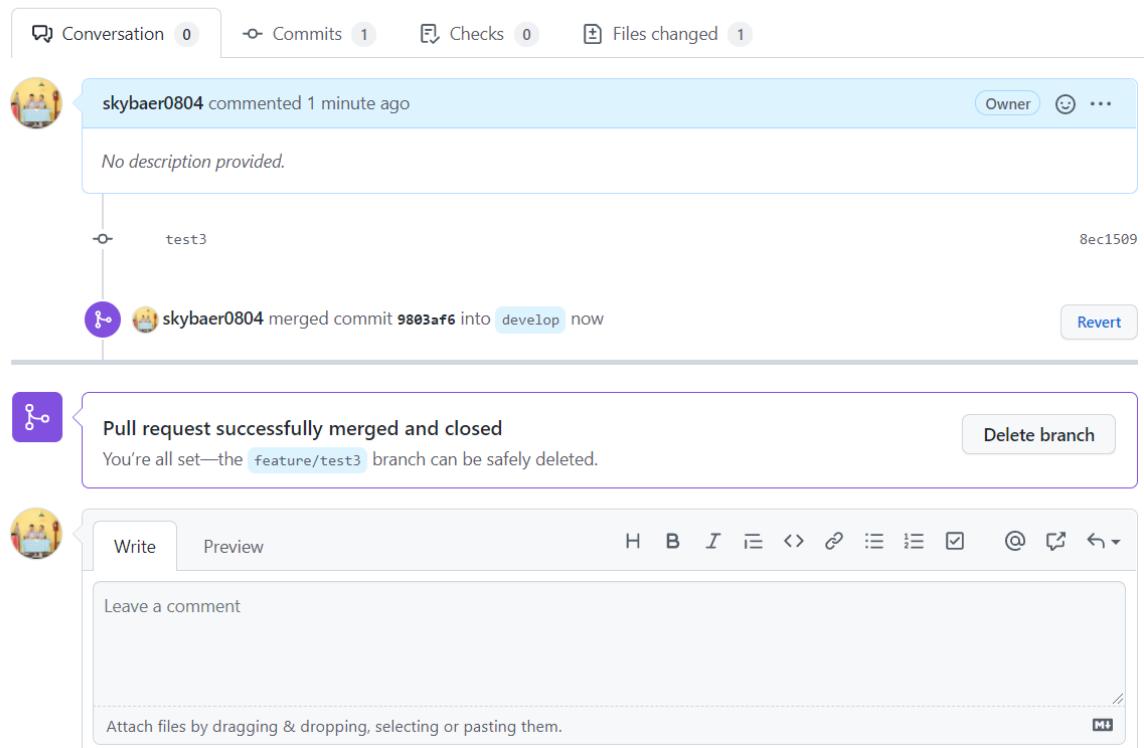
Attach files by dragging & dropping, selecting or pasting them.

Close pull request **Comment**

아래는 완료 페이지

test3 #6

Merged skybaer0804 merged 1 commit into develop from feature/test3 now



Fork 프로그램에서 Fetch를 누르거나 develop 브런치를 선택한 후 pull을 받으면 변경된 브런치가 보인다!

The screenshot shows a GitHub fork interface with the following details:

- File View Repository Window Help**
- Quick Launch Fetch Pull Push Stash**
- Fork** button
- New Branch** button
- Open in Console Appearance Home**
- Branches**: **test3** (selected)
- Commits**:
 - origin/develop**: Merge pull request #6 from skybaer0804/feature/test3
 - feature/test3**: test3
 - develop**: Merge pull request #5 from skybaer0804/feature/test2
 - feature/test2**: test2
 - feature/test1**: test1
 - master**: Merge pull request #3 from skybaer0804/develop
 - feature/kkkk**: 네번째 커밋
 - feature/twotwo**: dddd
 - feature/login1**: 두번째 커밋
 - test**: first class commit
- Remotes**: **origin**
- Tags**
- Stashes**
- Submodules**

Commit tab selected.

COMMITTER	DATE
GitHub noreply@github.com	6 Dec 2021 09:54:42 +09:00
skybaer0804	9803af6 6 Dec 2021 09:54:42 +09:00
nowonjae	a335e63 6 Dec 2021 09:49
skybaer0804	3451f18 6 Dec 2021 09:48
nowonjae	3dbd675 6 Dec 2021 09:47
skybaer0804	2d80f68 6 Dec 2021 09:46
nowonjae	37a1e40 6 Dec 2021 09:42
skybaer0804	83af678 6 Dec 2021 09:37
nowonjae	4db599c 6 Dec 2021 09:35
nowonjae	5ac9b4d 6 Dec 2021 09:34
nowonjae	4f51ae1 6 Dec 2021 09:32
nowonjae	1557724 1 Dec 2021 16:25
skybear0804	9010bf0 16 Nov 2021 15:44



여기까지 fork 사용법이였습니다.

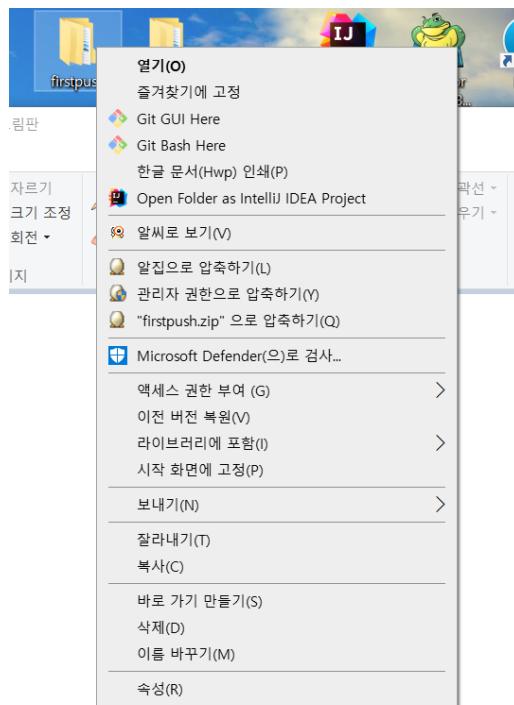
그 외 기능이나 git bash, 다른 털 들을 사용해도 됩니다.

아래는 git bash를 요약한 내용이므로 안보셔도 무방합니다.

4. git bash로 github 다루기

4-1. git upload

- 업로드할 파일 마우스 우클릭 → git bash 클릭



- \$ git init 입력

```
nodnj@DESKTOP-HMKE6KP MINGW64 ~/Desktop/firstpush (master)
$ git init
Reinitialized existing Git repository in C:/Users/nodnj/Desktop/firstpush/.git/
```

- \$ git status
- \$ git add . or \$ git add "파일명 or 폴더명"
- \$ git commit -m "Push 메시지명"
- git remote add origin repository 주소
- git remote -v
- git push origin master
- \$ git config --global user.name "username"
\$ git config --global user.email "email@email.com"
- \$ Username for https://github.com :
\$ Password for https://github.com/hyun98 :

4-2. git download

- \$ git clone repository주소

4-3. git pull(변경사항 받기)

- \$ git pull origin master

4-4. git branch create

1. \$ git branch 브랜치 이름

4-5. git branch 확인하기

1. \$ git log --decorate
HEAD는 현재 위치를 가르킵니다.

4-6. git 특정 branch로 이동

1. git checkout 브랜치이름

4-7. git branch merge

1. \$ git merge 브랜치이름

4-8. git 특정 branch 생성하며 이동하기

1. \$ git checkout -b 브랜치이름

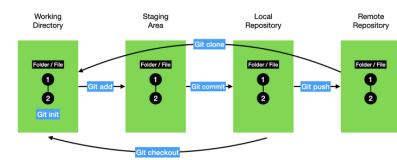
5. git의 개념

Reference

[Git] 깃 사용법 정리(1) - 깃으로 버전 관리하기

그 전까지 깃과 깃허브의 차이도 잘 몰랐고, 깃을 사용하는 방법도 야매로 배워서 레포지터리에 그저 공부한 흔적을 남기곤 했었는데 오늘에서야 비로소! 깃 사용법을 깨우쳤다! (>>> 아래놓고 막상 또 협업하면 어버버할지도...) 생각보다 그렇게 복잡하지 않더라구!! 그러니 겁먹지 말자! 여하튼 깃은!! 협업을 하려면 필수적으로 알아야

• <https://roniruny.tistory.com/112?category=907116>



Git 설치와 사용법(Git Bash)

Git을 이용하려면 두 가지 방법이 있다. SourceTree를 이용하면 GUI 툴이기에 접근하기에는 편하지만, 리눅스는 지원이 안되고 좀 더 디테일한 명령어를 다룰 수 없다. 따라서 보기엔 불편하지만 Git Bash를 이용하는 것이 낫다고 생각한다. (그리고 애가 좀 더 간지난다) 지금부터 두번째 방법에 대해서 알아보자. 현재 디렉토리를 로컬저장

• <https://gbsb.tistory.com/10>

```
rec@rec-PC MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [-chtml-path] [-man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:
start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an exist
```

<https://lee-mandu.tistory.com/298?category=715432>

git token create

[Mac] GitHub push token 오류 해결

7.29일 새벽 갑자기 git push가 안 되는 현상을 겪었다. 오류의 첫 줄이 무슨 말이냐면 Password 인증방식이 일시적으로 brownout(shutdown?)되었다. Password 대신에 personal access token을 사용해주세요. 깃허브에서 일시적으로 Password 인증 방식에 오류가 생긴 듯하다. 잠시 기다리면 해결될 수도 있겠지만 나는 바로 push를

<https://hyeo-noo.tistory.com/184>

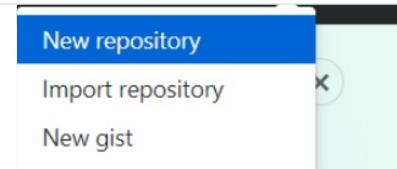


git push

[Git] Git 레파지토리 생성 & 소스 올리기 (Git Bash 활용)

GitHub에서 원격저장소를 만든 뒤 Git Bash를 활용하여 Push하는 방법입니다. 2. repository 정보를 입력하고 Create repository버튼을 눌러줍니다. 1. 원하는 소스폴더에서 git bash를 실행시킵니다. 깃을 init 시켜줍니다. (로컬저장소 만들기) 올라가 파일이 있는지 현재 폴더에 있는 파일들을 확인합니다. 지금은 파일들이 빨간색으로

 <https://coding-factory.tistory.com/244>



추가로 login시 토큰을 써야함.

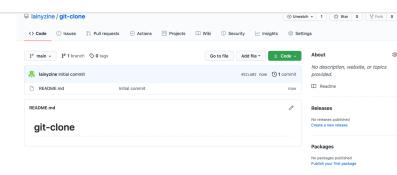
git HTTPS 프로토콜로 Git 저장소 clone

```
$ git clone https://github.com/lainyzine/git-clone.git
```

git clone 사용법: 원격 Git 저장소 복제

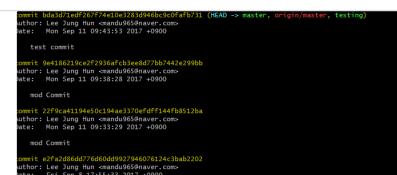
Git은 분산형 버전 관리 도구로, Linux 커널 소스코드를 관리하기 위해서 Linus Torvalds가 개발했습니다. Git으로 소스 코드를 관리하고 개발을 하려면 먼저 저장소를 초기화하거나 원격 저장소를 클론해와야 합니다. 저장소를 초기화하는 방법은 아래 글에서 소개합니다. 이미 초기화된 원격의 저장소를 복제해와서 개발하는 경우, 복제 작업

 <https://www.lainyzine.com/ko/article/git-clone-command/>



드디어 브랜치를 실습할 시간이 왔습니다. 이전에 학습했던 내용 첨부하겠습니다. [개발/개발도구] - git update 빙기(pull, fetch, merge) [개발/개발도구] - git 저장소 만들고 커밋(commit) 하기 [개발/개발도구] - [git 설치] github.com 처음 사용기 현재 필자는 svn을 사용중에 있습니다. svn 사용중에도 commit 이력을 확인 할 수 있습니다. 이에 저

<https://lee-mandu.tistory.com/313?category=715432>



Git - Branch와 Merge

여기서 소개하는 명령어만 알면 Branch를 사용하고 Merge 하는 일은 능히 할 수 있다. git branch 명령은 브랜치를 관리하는 도구다. 이 명령은 브랜치를 모두 보여주고 브랜치를 새로 만들고 브랜치를 삭제하고 브랜치 이름을 변경한다. git branch -u 명령으로 트래킹 브랜치를 만드는 것을 브래치 초기에서 보여준다. git checkout 명령은 브랜치를 선택하는 명령이다.

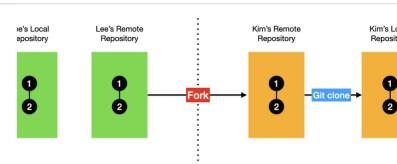
◆ <https://git-scm.com/book/ko/v2/%EB%B6%80%EB%A1%9D-C%3A-Git-%EB%AA%85%EB%A0%EB%9EC%96%EB%84-Branch%EC%99%80-Merge>



[Git] 깃 사용법 정리(2) - Fork와 Pull Request

Fork는 아래와 그림과 같이 Kim이라는 사람이 Lee의 프로젝트에 기여하기 위한 첫 시작이다. 기본적으로 Kim이 Lee의 프로젝트의 공동 협력자가 아니라면!! Kim은 자기 마음대로 Lee의 프로젝트를 수정할 권한이 없다. 따라서, 먼저!! 자신의 저장소로 Lee의 프로젝트를 통째로 복사한 후 자신의 로컬 저장소에 클론해 수정을 해야 한다.

⇒ <https://roniruny.tistory.com/113>



GitHub 접속 용 SSH 키 만드는 방법

GitHub를 사용하려면 SSH 키를 등록해야 합니다. Git과 GitHub를 처음 사용하면 SSH 키를 왜 만들어야 하는지, 어떻게 만들어야 하는지, 그리고 따라서 키를 만들어봤는데 잘 만들어진 건지 확인하는 데 어려움을 느낄 수 있습니다. 아래 내용을 따라온다면 어렵지 않게 SSH 키를 만들고, GitHub를 사용해볼 수 있습니다.

 <https://www.lainyzine.com/ko/article/creating-ssh-key-for-github/>

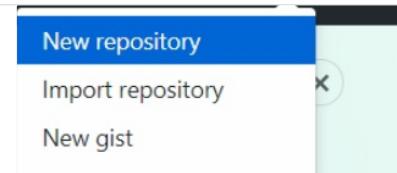
Title

Key
Begin with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', or 'ssh-ed25519'

[Git] Git 레파지토리 생성 & 소스 올리기 (Git Bash 활용)

GitHub에서 원격저장소를 만든 뒤 Git Bash를 활용하여 Push하는 방법입니다. 2. repository 정보를 입력하고 Create repository 버튼을 눌러줍니다. 1. 원하는 소스폴더에서 git bash를 실행시킵니다. 깃을 init 시켜줍니다. (로컬저장소 만들기) 올라가 파일이 있는지 현재 폴더에 있는 파일들을 확인합니다. 지금은 파일들이 빨간색으로

↳ <https://coding-factory.tistory.com/244>



GitHub 토큰 인증 로그인 하기 - [오류 해결]: remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token.

Creating a personal access token - GitHub Docs Personal access tokens (PATs) are an alternative to using passwords for authentication to GitHub when using the command line.

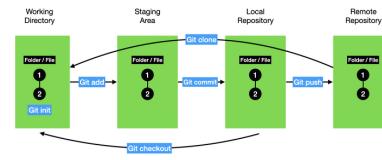
↳ <https://miracleground.tistory.com/entry/GitHub-%ED%86%A0%ED%81%B0-%EC%9D%B8%EC%A6%9D-%EB%A1%9C%EA%B7%BB%EC%9D%BB-%ED%99%A4%EB%A5%98-%ED%95%B4%EA%B2%BD-remote-Support-for-password-authentication-was-removed-on-August-13-2021-Please-use-a-personal-access-token>

1. git branch 전략

[Git] 깃 사용법 정리(1) - 깃으로 버전 관리하기

그 전까지 깃과 깃허브의 차이도 잘 몰랐고, 깃을 사용하는 방법도 애매로 배워서 레포지토리에 그저 공부한 흔적을 남기곤 했었는데 오늘에서야 비로소! 깃 사용법을 깨우쳤다! (>>> 아래놓고 막상 또 협업하면 어버버 할지도...) 생각보다 그렇게 복잡하지 않더라구!! 그러니 겁먹지 말자! 여하튼 깃은!! 협업을 하려면 필수적으

↳ <https://roniruny.tistory.com/112?category=907116>

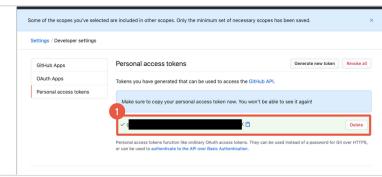


2. git token 인증 로그인

GitHub 토큰 인증 로그인: Personal Access Token 생성 및 사용 방법

GitHub에서 ID/PW기반의 Basic Authentication 인증을 금지하고, ID/Personal Access Token 방식의 Token Authentication 인증을 요구하고 있다. 앞으로는 소스코드를 push/clone하려고 하면, 아래와 같은 문제가 뜨면서 잘 되지 않는 상황이 발생할 수 있다. Password authentication is temporarily disabled as part

↳ <https://curryyou.tistory.com/344>



3. git bash

Git 설치와 사용법(Git Bash)

Git을 이용하려면 두가지 방법이 있다. SourceTree를 이용하면 GUI 틀이기에 접근하기에는 편하지만, 리눅스는 지원이 안되고 좀 더 디테일한 명령어를 다룰 수 없다. 따라서 보기엔 불편하지만 Git Bash를 이용하는 것이 낫다고 생각한다. (그리고 애가 좀 더 간지난다) 지금부터 두번째 방법에 대해서 알아보자. 현재 디렉토

↳ <https://gbsb.tistory.com/10>

```
xc@Beom-PC MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [-e exec-path=<path>] [-E html-path] [-m man-path] [-i info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [-g git-dir=<path>] [-w work-tree=<path>] [-n namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:
start a working area (see also: git help tutorial)
  clone   Clone a repository into a new directory
  init    Create an empty Git repository or reinitialize an exist
```

4. fork tool

Fork - a fast and friendly git client for Mac and Windows

Fork - a fast and friendly git client for Mac and Windows

↳ <https://www.fork.dev/>

