



# GIT & Github

Kim Hye Kyung

# | 개요

Kim Hye Kyung

# Git & Github



소스 코드의 변경 사항 내역을 관리하는 분산  
버전 관리 시스템

코드 변경 추적

변경자 추적

코딩 협업

형상 관리 Tool

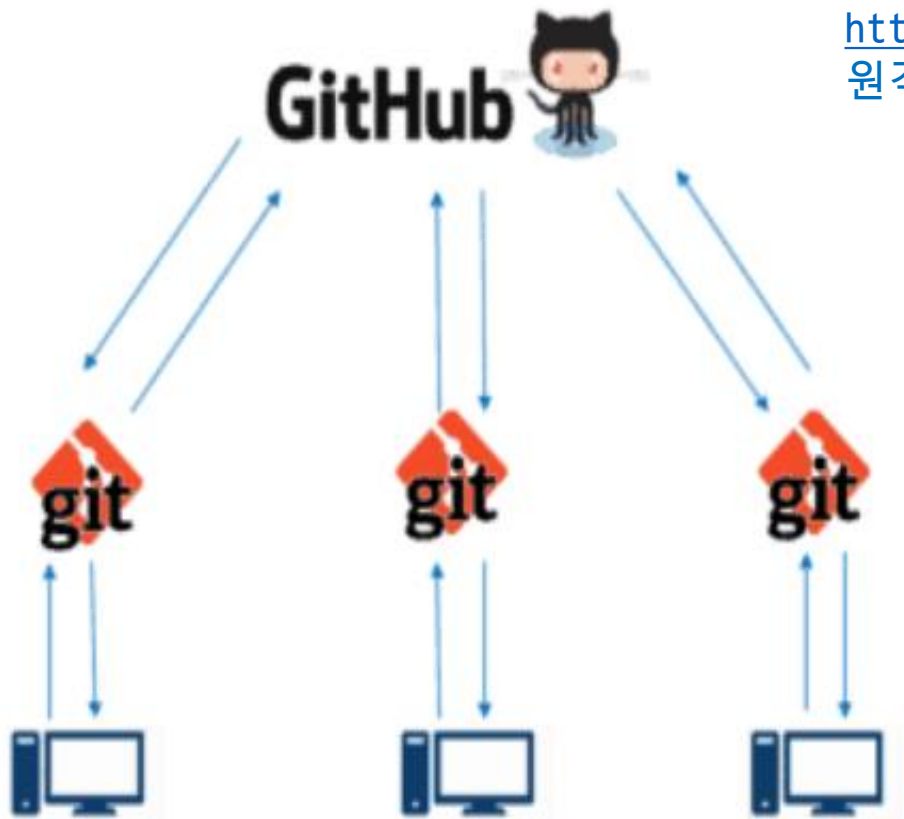


원격 중앙 저장소

개발자간의 협업을 위해 중앙 서버 역할을 하는  
서비스

협업을 위한 코드 review, document 생성 및  
관리 등 개발 프로젝트 운영에 필요한 여러가지  
기능을 제공

# Git & Github



<https://github.com/>  
원격 저장소



Github인 원격 저장소와 Local Repository는 동기화가 되어 동일한 내용 유지

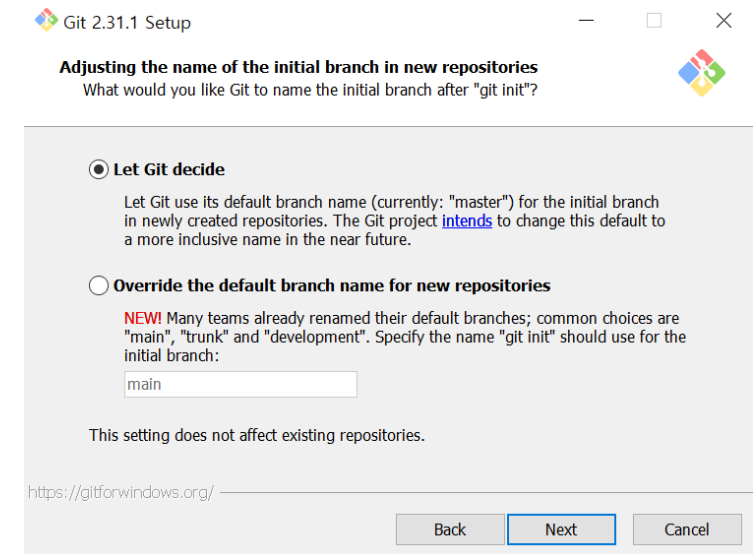
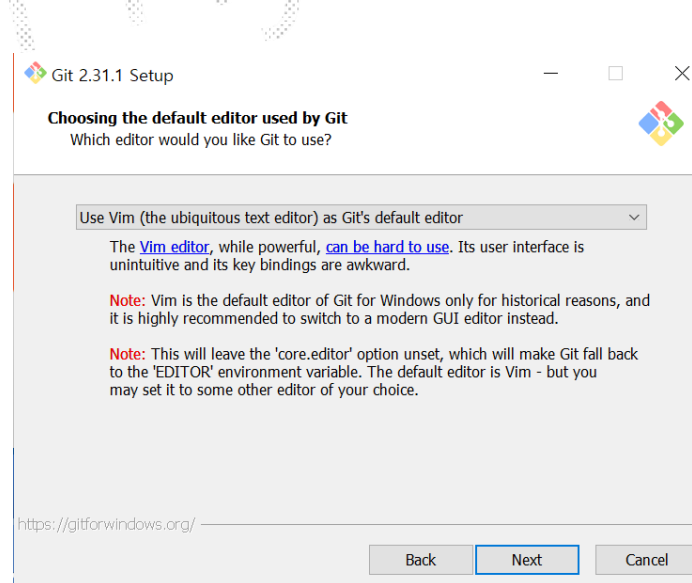
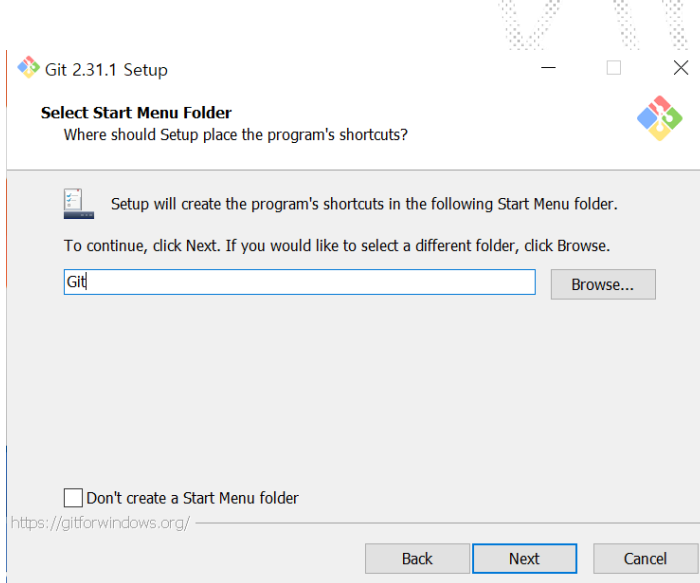
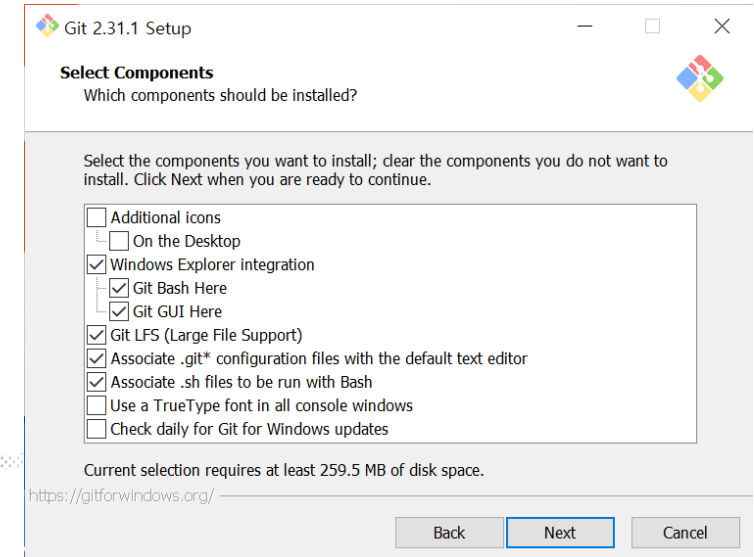
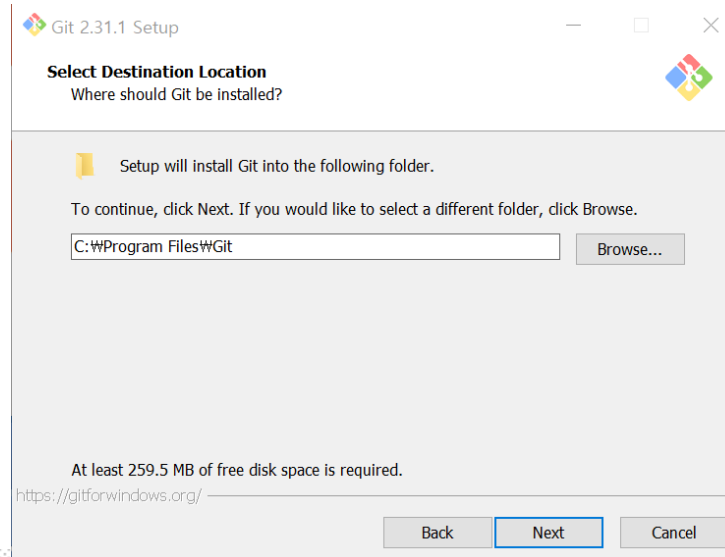
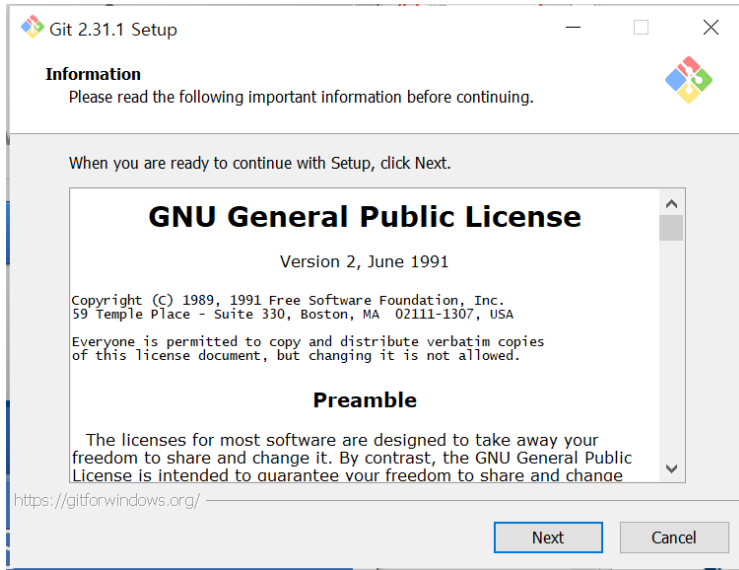
gitbash 설치 / 로컬 저장소

Git for Windows는 명령줄에서 Git을 실행

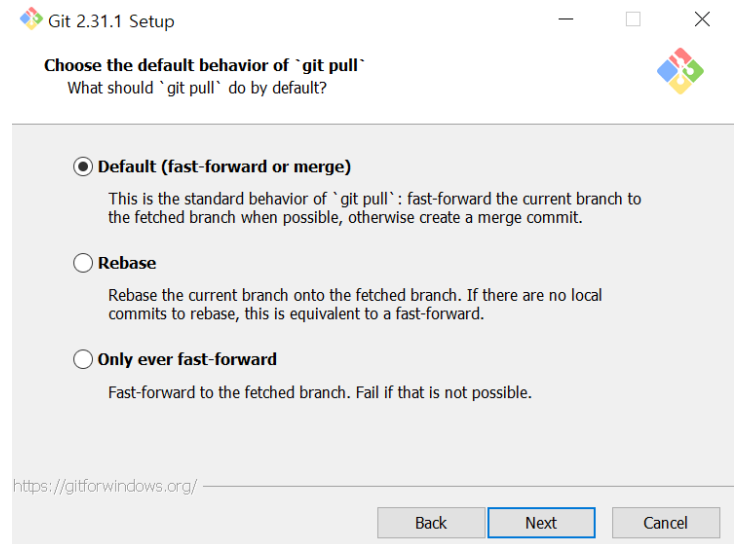
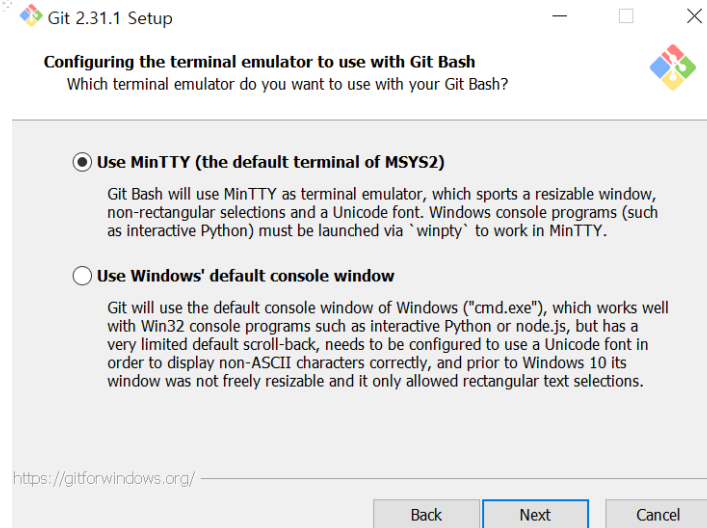
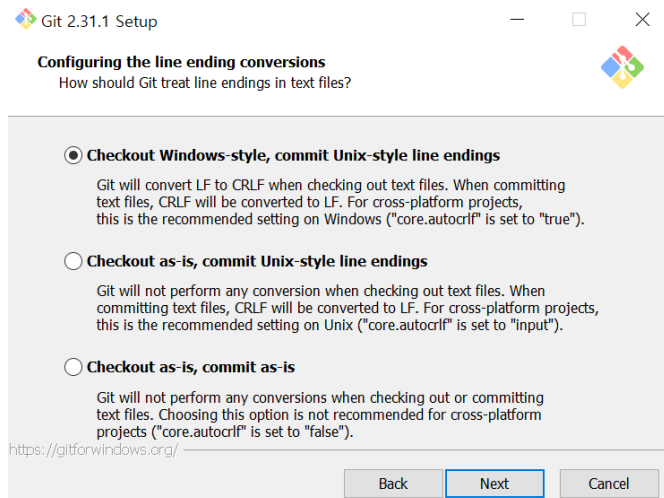
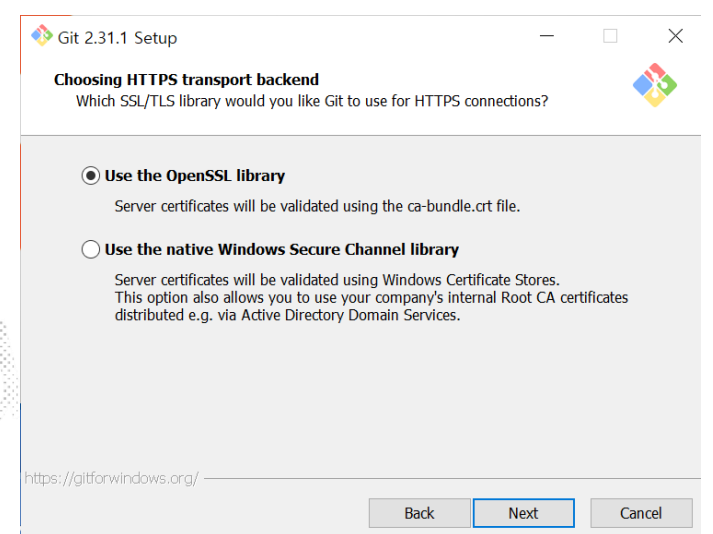
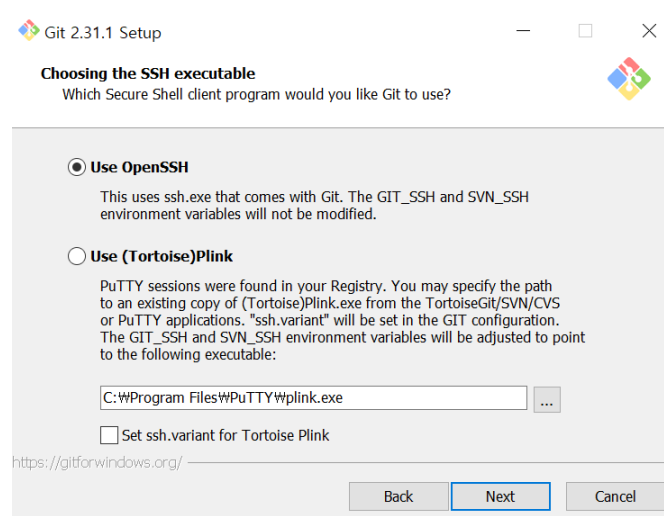
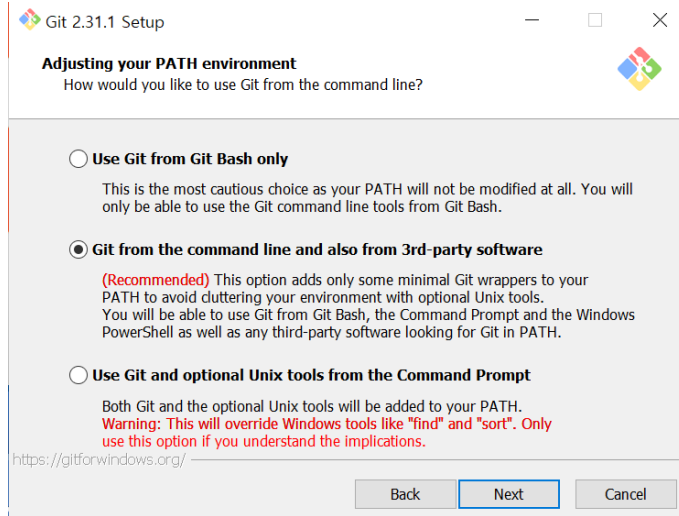
<https://git-scm.com/>



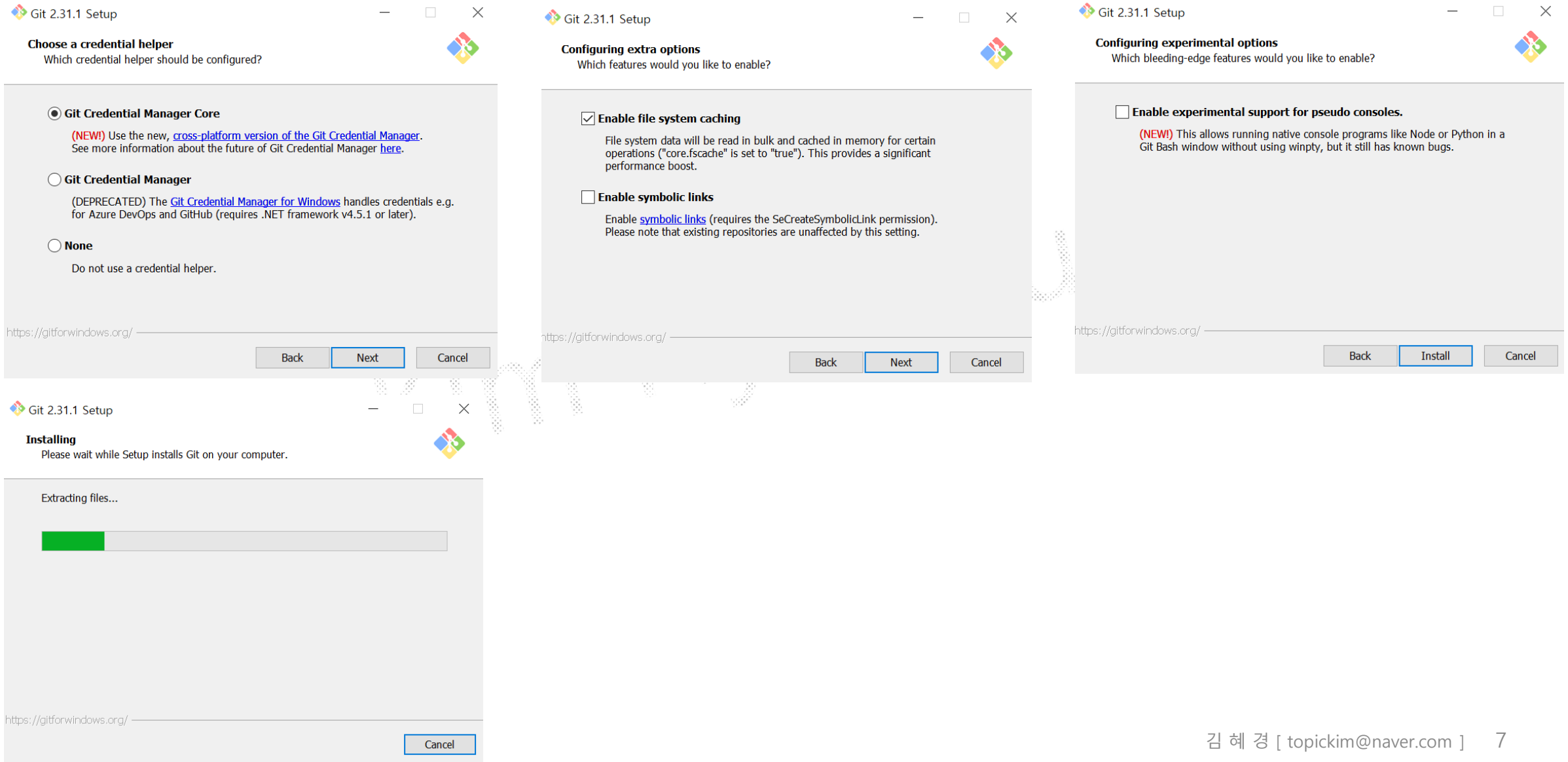
# Git & Github 사용을 위한 gitbash 설치 1/3



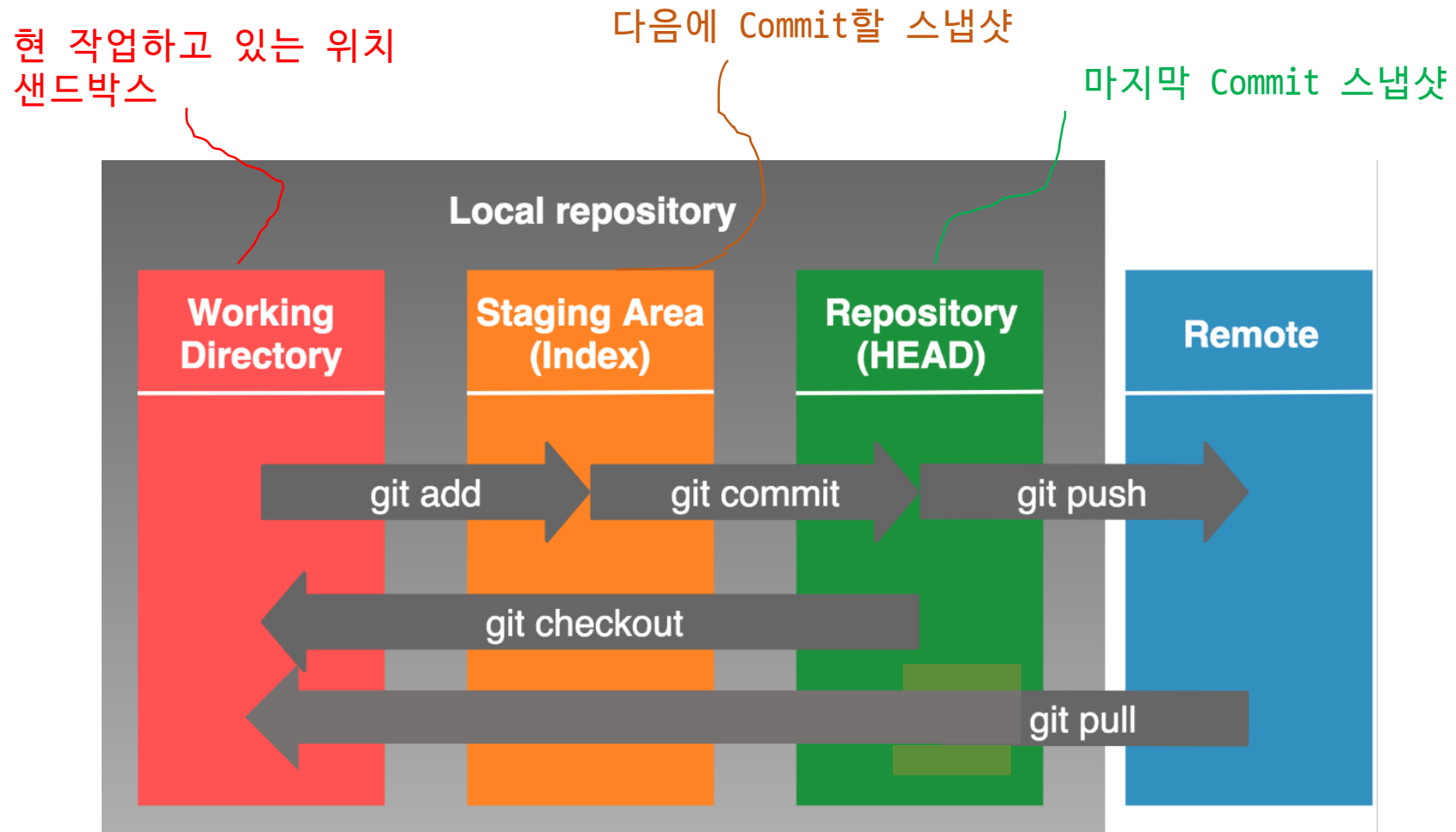
# Git & Github 사용을 위한 gitbash 설치 2/3



# Git & Github 사용을 위한 gitbash 설치 3/3

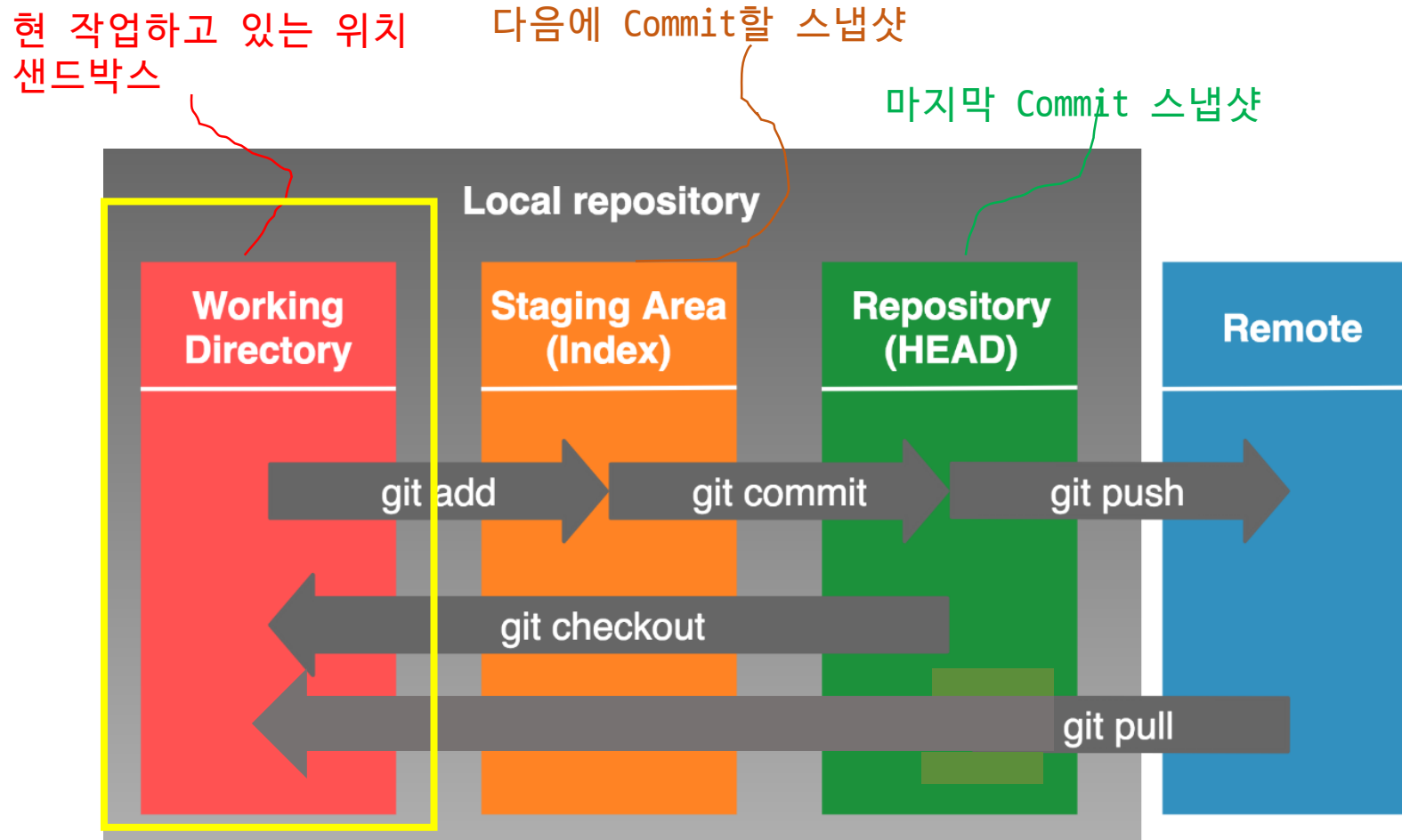


# Git 시스템 이해하기 1/3





# Git 시스템 이해하기 2/3



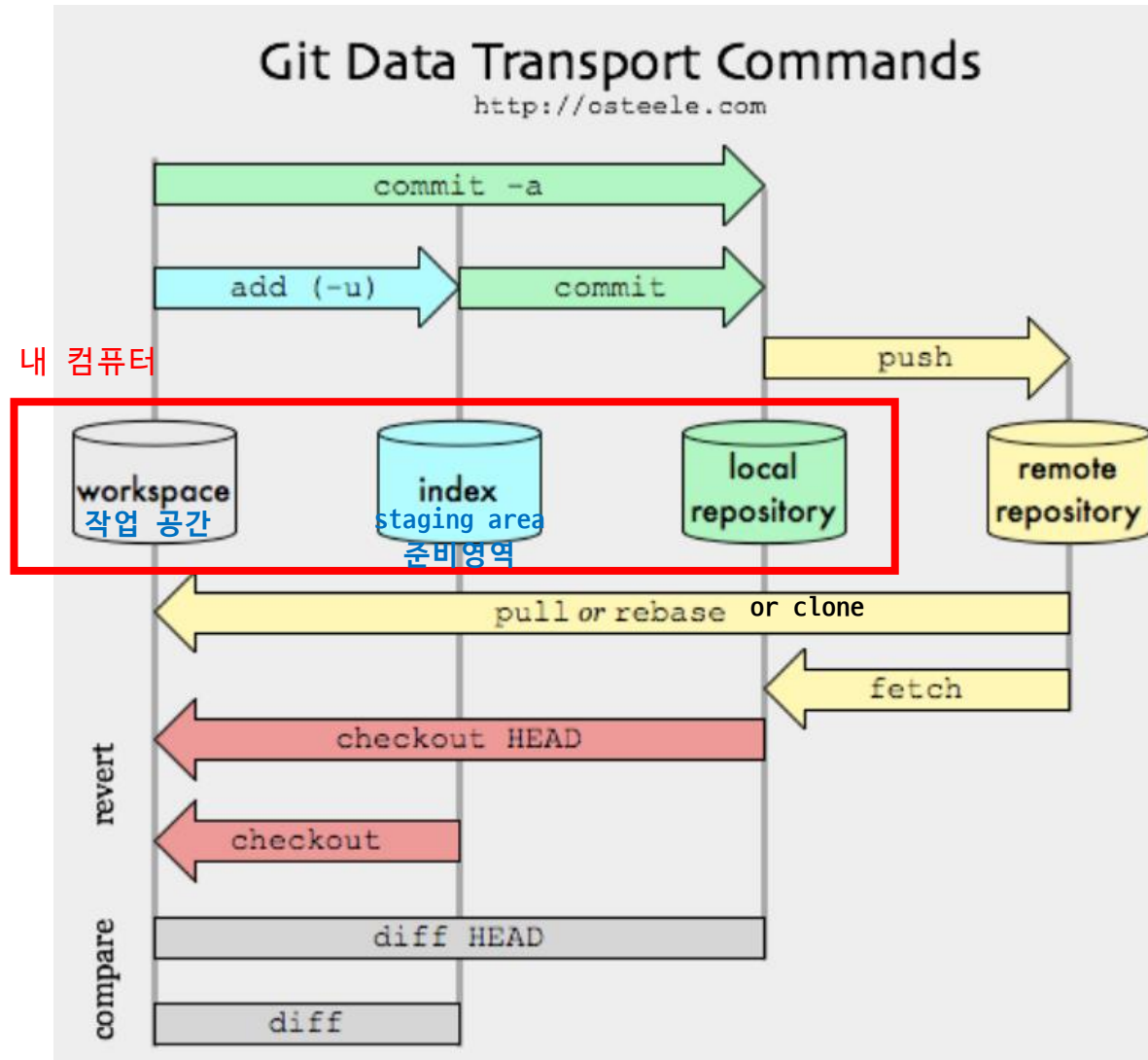
## Working Directory 구성

작업폴더는 관리(추적)가 되는 파일과 관리(추적)되지 않는 파일로 나누어져 있음  
관리(추적)가 된다는 것 - 파일의 생성, 수정, 삭제 등의 히스토리 정보를 모두 가지고 있다는 뜻

## File 관리 상태

- 추적안함 (**Untracked**) : 관리대상이 아님
- 추적함(**Tracked**)
- 수정없음 (**Unmodified**) : 변경이 없는 파일
- 수정함 (**Modified**) : 변경된 파일
- 스테이지됨 (**Staged**) : 스테이지에 올라간 파일

# Git 시스템 이해하기 3/3



**add** = 작업을 하고 Stage공간에 올리는 작업

**Commit** = staging 파일들을 로컬에 저장하는 것

**Fetch** = 원격 저장소 내용을 가져와서 확인만 하는 용도, 병합은 하지 않음

**Pull (fetch+merge)** = 원격 저장소의 코드를 가져와서 내 로컬 코드와 병합

**Push** = 로컬에서 작업한 내용을 원격저장소에 저장

**Merge** = 여러개의 Commit을 하나의 Commit으로 병합하는 것

**Discard** = Commit을 하지 않은 자료를 최종 버전의 Commit으로 되돌리는 것

**Revert** = Commit 이력 그대로 두고, 되돌릴 Commit의 코드만 원복함  
(push했다면 revert)

**Reset** = Commit 이력까지 삭제하고 특정 Commit 지점으로 돌아감

참고 : 원격 저장소는 Origin이라 명시

| 실습

Kim Hye Kyung

# Trouble Shooting 이란?

---

**문제가 발생했을 때 원인을 규명하고 해결하는 작업 의미**

# Trouble Shooting 1/2

---

- 발생 가능한 이슈
  - 문제
    - 원격 저장소와 연동시 github 사이트 연동 불가인 이슈가 발생할 경우
    - "Logon failed, use ctrl+c to cancel basic credential prompt."
    - gitbash 버전이 낮을 경우에 발생됨
  - 해결책
    - git update 명령어로 업데이트 하기
    - >git update-git-for-windows

# Trouble Shooting 2/2

```
Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ ls
README.md  index.html

Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 625 bytes | 312.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ITkim/hello-world.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'
.

Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ git push -u origin master
Everything up-to-date
Branch 'master' set up to track remote branch 'master' from 'origin'
.

Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ git push -u origin master
To https://github.com/ITkim/hello-world.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/ITkim/hello-world.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

## 문제

github 사이트에서 수정 후 로컬로 받아오지 않은 상태에서 로컬의 파일을 push 시도시 에러 발생

## 해결책

로컬로 pull 먼저 수행 후에 작업

# 실습을 위한 단계

- 소스 개발
- gitbash 실행
- 로컬 저장소 생성
- stage 영역으로 소스 저장
- commit 메시지와 함께 로컬 저장소에 저장
- 로그 기록 보기
- github 사이트에 repository 생성
- 로컬 저장소의 소스들을 github 사이트에 push
- github 사이트에서 소스 편집해 보기
- 로컬 저장소로 pull 해 오기
- ...
- 원격 저장소에 저장되는 파일 list 관리하기
- 원격 저장소와 연결 끊기
- ...

# 실습

add -> commit -> push 의 단계로 명령어를 입력

- |                                 |                                        |
|---------------------------------|----------------------------------------|
| 1. >git config                  | git commit에 사용될 username / email 사용    |
| 2. >git init                    | 현재 디렉토리를 로컬저장소로 설정                     |
| 3. >git status                  | 로컬 저장소의 현재 상태 확인                       |
| 4. >git add 파일명                 | 파일을 준비영역(Staging Area)영역으로 옮김          |
| 5. >git commit -m 'push 메세지'    | 준비영역의 파일을 로컬 저장소에 저장                   |
| 6. >git log                     | commit한 히스토리 확인                        |
| 7. >git remote add origin 원격URL | 로컬저장소와 원격저장소를 연결                       |
| 8. >git remote -v               | 연결된 원격저장소 확인                           |
| 9. >git push origin master      | 원격 저장소에 저장                             |
| 10. >git pull                   | 원격 저장소에 존재하는 최신 수정본 파일을 내 로컬 저장소로 업데이트 |
| 11. ...                         |                                        |



# git diff 실습 예시

- 로컬에서 파일 수정
- >git status
- >git diff
  - 갱신된 내용 검색됨
- >git add file명
- >git status

```
Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ git diff
diff --git a/index.html b/index.html
index 8c588dd..1c9e5aa 100644
--- a/index.html
+++ b/index.html
@@ -10,5 +10,7 @@
     <p>github 사이트에서 추가한 내용 </p>
     <p>local에서 추가한 내용 </p>
+
+     local에서 수정
+
   </body>
 </html>

Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ ls
README.md  index.html

Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ git add index.html

Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ git diff

Kimhyekyung@Hyekyung MINGW64 /d/98.tool/myproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html
```

# gitignore 이란?

- 원격 저장소에서 관리하지 말아야 되는 파일들을 제어하는 방식
  - Git이 무시할 파일들, 즉 추적하지 않을 파일들을 정의하는 파일
  - 적용 방법
    - 파일명 : .gitignore
    - 위치 : 프로젝트 최상위에 존재
    - 방식 : 패턴을 활용하여 git이 untracked할 파일 또는 디렉토리 등을 정의하여 파일로 생성
  - 작성 패턴 규칙
    1. '#'로 시작하는 라인은 무시
    2. **표준 Glob 패턴**을 사용
    3. 슬래시(/)로 시작하면 하위 디렉터리에 적용되지(recursivity) 않음
    4. 디렉터리는 슬래시(/)를 끝에 사용하는 것으로 표현
    5. 느낌표(!)로 시작하는 패턴의 파일은 무시하지 않음
- Glob 패턴은 와일드카드 문자를 사용해서 일정한 패턴을 가진 파일 이름들을 지정하기 위한 패턴

# .gitignore 파일 생성 도우미 site

- <https://www.toptal.com/developers/gitignore>



**gitignore.io**

자신의 프로젝트에 꼭 맞는 .gitignore 파일을 만드세요

운영체제, 개발 환경(IDE), 프로그래밍 언어 검색

생성

[소스 코드](#) | [커맨드라인 문서](#)

**gitignore.io**

자신의 프로젝트에 꼭 맞는 .gitignore 파일을 만드세요

Python x

생성

```
# Created by https://www.toptal.com/developers/gitignore/api/python
# Edit at https://www.toptal.com/developers/gitignore?templates=python
```

```
### Python ###
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class
```

```
# C extensions
*.so
```

```
# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
parts/
sdist/
var/
wheels/
pip-wheel-metadata/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST
```

```
# PyInstaller
# Usually these files are written by a python script from a template
# before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec
```

```
# Installer logs
pip-log.txt
pip-delete-this-directory.txt
```

```
# Unit test / coverage reports
htmlcov/
.tox/
.nox/
coverage
```

# .gitignore 실습

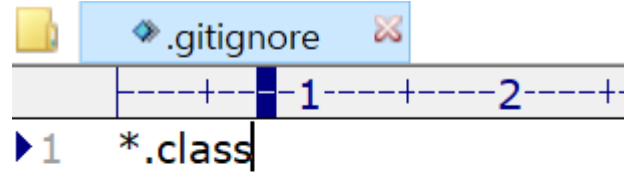
## 1. 디렉토리

이름

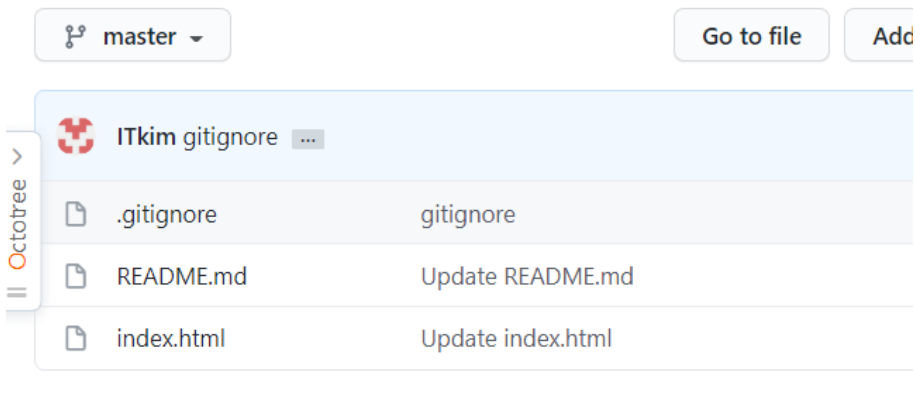
- .gitignore
- Beneficiary.class
- Donator.class
- index.html
- README.md
- TalentDonationProject.class
- TalentDonationType.class

## 2. file 설정

– class 확장자 파일제외



## 3. push 후 class가 제외된 원격 저장소



```
Kimhyekyung@HyeKyung MINGW64 /d/98.tool/myproject (master)
$ ls -a
./      .gitignore      README.md      index.html
../     Beneficiary.class TalentDonationProject.class
.git/   Donator.class   TalentDonationType.class
```

```
Kimhyekyung@HyeKyung MINGW64 /d/98.tool/myproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  .gitignore
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

```
Kimhyekyung@HyeKyung MINGW64 /d/98.tool/myproject (master)
$ git add .gitignore
```

```
Kimhyekyung@HyeKyung MINGW64 /d/98.tool/myproject (master)
$ git commit -m 'gitignore'
[master 6b2bfaf] gitignore
1 file changed, 1 insertion(+)
create mode 100644 .gitignore
```

```
Kimhyekyung@HyeKyung MINGW64 /d/98.tool/myproject (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 309 bytes | 309.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ITkim/hello-world.git
9463703..6b2bfaf master -> master
```

```
Kimhyekyung@HyeKyung MINGW64 /d/98.tool/myproject (master)
$ ls
Beneficiary.class  TalentDonationProject.class
Donator.class      TalentDonationType.class
README.md          index.html
```

# 원격 저장소와 연결 끊기

- git remote remove origin

```
Kimhyekyung@HyeKyung MINGW64 /d/98.tool/myproject (master)
$ git remote remove origin
```

```
Kimhyekyung@HyeKyung MINGW64 /d/98.tool/myproject (master)
$ git status
On branch master
nothing to commit, working tree clean
```

```
Kimhyekyung@HyeKyung MINGW64 /d/98.tool/myproject (master)
$ git pull origin master
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

# GitHub의 이슈 관리 기능

- GitHub에 이슈 등록하기

- 코드를 작성하기에 앞서 요구 사항이나 해결해야 하는 문제를 명확하게 정의하는 것 권장
- 활용하면 협업하는 동료에게 쉽게 공유될 뿐만 아니라 기록이 남아 짧게는 며칠에서 길게는 1년 후에도 참고할 수 있는 좋음

## [권장하는 이슈 template]

### ## Overview

- [x] Features
- [ ] QA/Bug Reports

### ## Current behavior

### ## Expected behavior

- [ ] A

### ## :memo: To Reproduce

..

The image shows two screenshots of the GitHub issue template editor. The top screenshot is the 'Write' tab, showing the template text with placeholders like '## Overview', '- [x] Features', '- [ ] QA/Bug Reports', '## Current behavior', '## Expected behavior', '- [ ] A', '## :memo: To Reproduce', and '..'. The bottom screenshot is the 'Preview' tab, showing the rendered HTML with checkboxes for 'Features' and 'QA/Bug Reports', and a section for 'To Reproduce' with a placeholder image icon.

# 참고 사이트

- <https://git-scm.com/book/ko/v2>
- <https://velog.io/@jcinsh/Git-Github-%EA%B8%B0%EC%B4%88>
- glob 패턴
  - <https://velog.io/@k7120792/Glob-%ED%8C%A8%ED%84%B4%EA%B3%BC-%EC%A0%95%EA%B7%9C%ED%91%9C%ED%98%84%EC%8B%9D>
  - test 사이트
    - <https://www.digitalocean.com/community/tools/glob?comments=true&glob=%2F%2A%2A%2F%2A.js&matches=f%2F%2F%20This%20will%20match%20as%20it%20ends%20with%20%27.js%27&tests=%2Fhello%2Fworld.js&tests=%2F%2F%20This%20won%27t%20match%21&tests=%2Ftest%2Fsome%2Fglobs>

