

형상관리도구

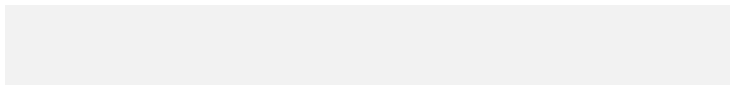


목차

1. GIT
2. GITHUB
3. GIT과 GITHUB의 주요 개념
4. GITHUB 설정하기
5. 기타설정

GITHUB

1. GIT



- 깃(Git)의 개념

- 프로그램 등의 소스 코드 관리를 위한 분산 버전 관리 시스템.
- 깃의 작업 폴더는 모두 기록하고 있어서 추적이 가능하고, 완전한 형태의 저장소임.
- 원격 저장소를 만들고 사용하면 협업이 가능.
- 인터넷을 경유할 필요가 없기 때문에 빠르고, 커밋(Commit)에 부담이 없는 장점이 있음.

※ 커밋(Commit)

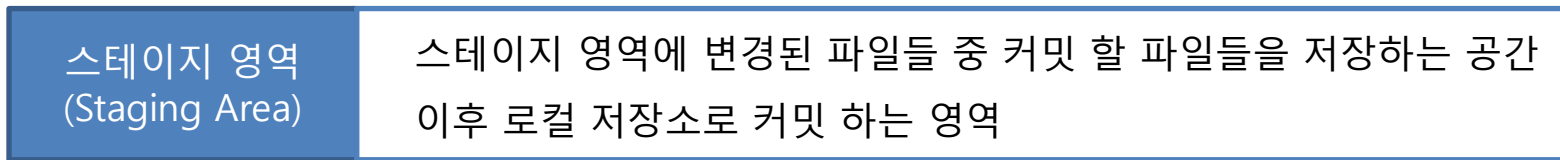
분산 트랜잭션 처리(DTP)에서, 하나의 트랜잭션에 포함되는 조작이 모두 실행되고 그에 따른 데이터베이스의 갱신 내용이 작업 영역(기억 장치)에 기록되어 트랜잭션의 적용이 완료되었다고 판단되는 시점에서 그 종료를 요구하는 동작

※ 트랜잭션

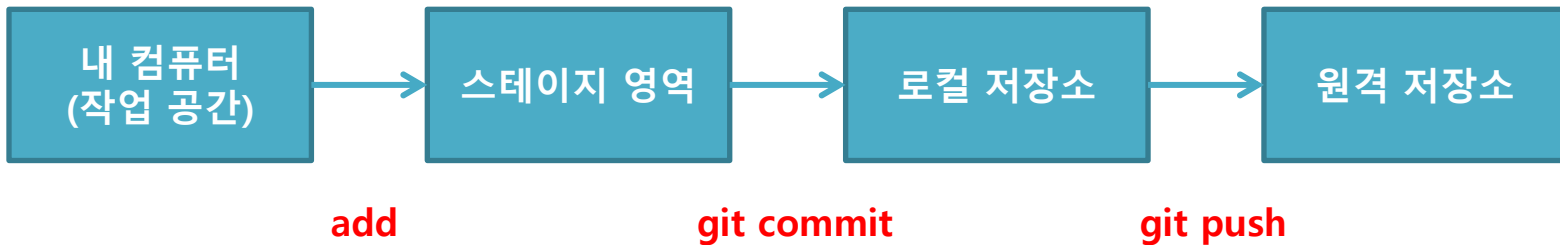
특정 목적을 수행하는 일련의 작업들의 집합

- 깃(Git)의 개념

- Git에는 로컬 저장소에 커밋을 하기 전, 스테이지 영역(인덱스) 단계가 있음

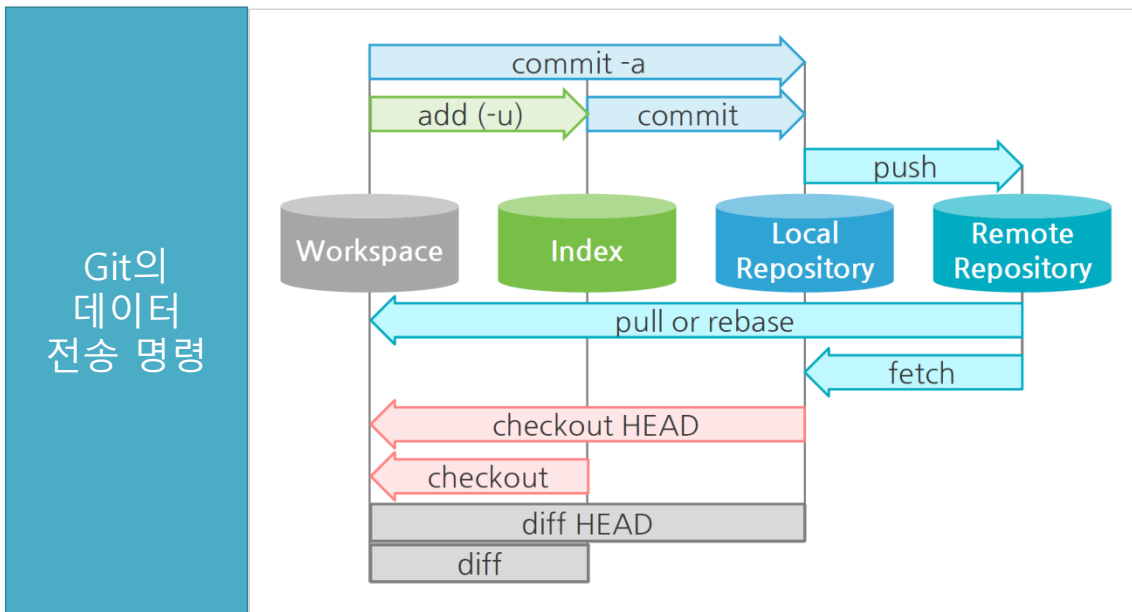


- Git의 저장소 구조



- 깃(Git)의 개념

- Commit은 로컬 저장소에서 이루어 지고 Push라는 동작으로 원격 저장소에 반영됨(SVN의 차이점)
 - 로컬 저장소에서 작업이 이루어져 매우 빠른 응답을 받을 수 있음
 - Pull 또는 Fetch로 서버에서 변경된 내역을 받아 올 수 있음



- 깃(Git)의 특징
 - Distributed Development
 - 전체 개발 이력을 각 개발자의 로컬로 복사본을 제공하고 변경된 이력을 다시 하나의 저장소로 복사함
→ 이러한 변경은 추가개발 지점을 가져와, 로컬개발 지점과 **동일하게 병합(Merge)**할 수 있음
 - 저장소는 Git Protocol 및 HTTP로 특별한 웹 서버 구성없이 **쉽고 효율적으로 접근이 가능함**
 - Strong Support for Non-linear Development
 - 신속하고 편리한 Branch 및 Merge 지원
 - 비선형(여러 갈래) 개발 이력을 시각화하고 탐색할 수 있는 강력한 도구 제공
 - Efficient Handling of Large Projects
 - Git은 매우 빠르고, 대형 프로젝트나 이력이 많은 작업에 매우 합리적임
 - Git은 대부분의 다른 버전관리 시스템보다 빠르게 요청하며, 일부 작업에서는 더 빠르게 진행함

- 깃(Git)의 특징
 - Cryptographic Authentication of History
 - Git의 이력은 성공한 개발 이력의 Commit에 의해 개정명으로 저장됨
 - 일단 저장된 이력이 배포되면, 예전버전으로 변경하는 것은 불가능함
 - 이력은 암호화가 가능함
 - Toolkit Design
 - UNIX의 전통에 따라 Git은 C로 작성된 많은 소규모 도구모음이며, 많은 스크립트들이 기능 보강을 제공함
 - Git은 새로운 기발한 작업을 위한 손쉬운 사용과 쉬운 스크립팅을 위한 도구를 제공함

- 깃허브(GitHub)의 개념
 - Git을 호스팅 해주는 웹 서비스
 - Git 저장소 서버를 대신 유지 및 관리해주는 서비스
 - 오픈 소스 프로젝트, Private 프로젝트 모두 무료이며, 다른 유저들과 함께 온라인으로 하나의 프로그램을 제작하는 것도 가능함
 - 많은 오픈 소스 프로그램들이 GitHub를 통해 전세계 유저들에 의해 제작되고 있음

- 깃허브(GitHub)의 장점

1. 기록요구

- 이슈(Issue)를 사용해 버그를 기록하거나 개발하고 싶은 새로운 기능을 구체화할 수 있음

2. 독립된 히스토리(History)에 대한 협력

- Branch와 Pull Requests를 이용하여 다른 Branch 또는 기능에 협력할 수 있음

3. 진행 중인 작업 검토

- Pull Requests 목록을 통해 현재 무슨 작업이 진행되고 있는지 모두 볼 수 있음 특정 Pull Request를 클릭하여 최근의 변경 내용과 변경에 관한 모든 논의 내용을 볼 수 있음

4. 팀의 작업 진척 상황 확인

- 펄스(Pulse)를 훑어보거나 Commit History를 살펴보면 팀의 작업 진척 상황을 알 수 있음

- **Commit**
 - 하나 또는 다수의 파일에 변경 내용을 저장할 때마다 새로운 Commit이 생성됨
예) “이 변경 내용을 Commit하고 이를 GitHub로 Push 합시다.”
- **Commit Message**
 - Commit을 생성할 때마다 왜 변경을 했는지에 대한 이유를 설명하는 메시지를 제공해야 함
변경을 한 이유를 추후에 이해할 때 유용함
예) “새로운 SEC 가이드라인에 대한 OO의 의견을 Commit Message에 꼭 넣으세요.”
- **Branch**
 - 테스트를 해보거나 새로운 기능을 개발하기 위해 사용할 수 있는 따로 떨어진 독립적인 Commit들
예) “새로운 검색 기능을 구현하기 위해 Branch를 생성합시다.”

- Master Branch

- 새로운 Git 프로젝트를 만들 때마다 'Master'라고 불리는 기본 Branch가 생성됨 배포할 준비가 되면 작업이 최종적으로 마무리되는 Branch

예) "Master로 바로 Commit 하면 안 된다는 것을 기억하세요."

- Feature(Topic) Branch

- 기능을 개발할 때마다 만드는 작업할 Branch

예) "Feature Branch가 너무 많습니다. 이들 중 하나나 두 개를 완료해서 출시하는데 주력합니다."

- Release Branch

- 직접 QA(품질보증) 작업을 하거나 고객의 요구로 구 버전의 소프트웨어를 지원해야 한다면 모든 수정이나 업데이트를 위한 장소로 Release Branch가 필요함
- Feature Branch와 Release Branch는 기능적 차이는 없지만, 팀원들과 프로젝트에 대해 이야기할 때 확연히 구별할 수 있는 용어임

예) "Release Branch 전체에 대한 보안 버그를 수정해야 합니다."

- Merge(병합)

- 한 Branch에서 완성된 작업을 가져와 다른 Branch에 포함하는 방법 흔히 feature Branch를 Master Branch로 Merge함
예) “‘내 계정’기능이 훌륭합니다. 배포할 수 있게 Master로 Merge 해줄 수 있습니까?”

- Tag

- 특정 이력이 있는 Commit에 대한 참조 어떤 버전의 코드가 언제 배포(Release)되었는지 정확히 알 수 있도록 제품 배포 기록에 가장 자주 사용됨
예) “이번 배포판에 tag를 달고 출시합니다.”

- Checkout

- 프로젝트 History의 다른 버전으로 이동하여 해당 시점의 파일을 보기 위해 ‘Checkout’을 함 가장 일반적으로 Branch에서 완료된 작업을 모두 보기 위해 Branch를 Checkout함 Commit도 Checkout할 수 있음
예) “최종 릴리즈 태그(Release Tag)를 Checkout 해주시겠어요? 제품에 버그가 있어서 검증하고 수정해야 합니다.”

- Pull Request

- 본래는 Branch에서 완료된 작업을 다른 사람이 리뷰하고 Master로 Merge요청을 하기 위해 사용되었음 요즘은 개발 가능한 기능에 대한 논의를 시작하는 초기 단계에서 자주 사용함

예) “다른 팀원들이 어떻게 생각하는지 알 수 있게 새로운 투표 기능에 대한 Pull Request를 만드십시오.”

- Issue

- 기능에 대해 논의하거나 버그를 추적하거나 혹은 두 가지 경우 모두 사용될 수 있는 GitHub의 기능

예) “아이폰에서는 로그인되지 않네요. 버그를 검증하기 위한 단계를 기록하면서 GitHub에 Issue를 생성해 주시겠습니까?”

- Wiki

- Ward Cunningham이 최초로 개발함
- Wiki는 링크들 간을 연결해 간단하게 웹 페이지를 만드는 방법
- GitHub 프로젝트는 문서 작성에 자주 Wiki를 사용함

예) “복수의 서버에서 작동하게끔 프로젝트 환경 설정 방법을 설명하는 페이지를 추가해 주시겠습니까?”

- Fork
 - 프로젝트를 직접 변경하는 데 필요한 권한을 보유하지 못한 경우(오픈 소스 프로젝트이거나 다른 그룹이 작성한 프로젝트인 경우) 프로젝트를 변경하고 싶다면 먼저 GitHub의 사용자 계정에 프로젝트 복사본을 만들어야 함
→ 이러한 과정을 Repository를 Fork한다고 함
 - Repository를 Fork한 후 복제(Clone)하고, 변경하고, Pull Request를 이용해 원본 프로젝트에 변경 내용을 반영할 수 있음
예) "홈페이지 마케팅 원고를 당신이 어떻게 재 작성 했는지 보고 싶습니다. Repository를 Fork하고 수정안과 함께 Pull Request를 제출해주세요."

- 설정 절차
 - Git 설치 및 GitHub 가입
 - Git 다운로드 경로
 - Windows용 : <https://git-scm.com/download/win>
 - OS X용 : <https://git-scm.com/download/mac>
 - GitHub 회원가입 : <https://github.com>
 - 저장소 생성 및 저장하기
 1. 원격 저장소 생성하기
 2. 로컬 저장소 생성하기
 3. 로컬 저장소에 저장하기
 4. 원격 저장소에 저장하기

- 설정 절차

- Git 설치

- git 홈페이지 → [Downloads] → OS 버전에 맞는 설치 파일 다운로드 → 다운로드 받은 설치 파일 클릭 → [NEXT]로 계속 진행 → 바탕화면에 생성된 [Git Bash] 클릭하면 Shell이 뜸 현재 디렉토리의 파일 목록 확인

Downloading Git



You are downloading the latest (**2.31.1**) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **about 2 months ago**, on 2021-03-27.

[Click here to download manually](#)

Other Git for Windows downloads

Git for Windows Setup

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

Git for Windows Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

The current source code release is version 2.31.1. If you want the newer version, you can build it from [the source code](#).



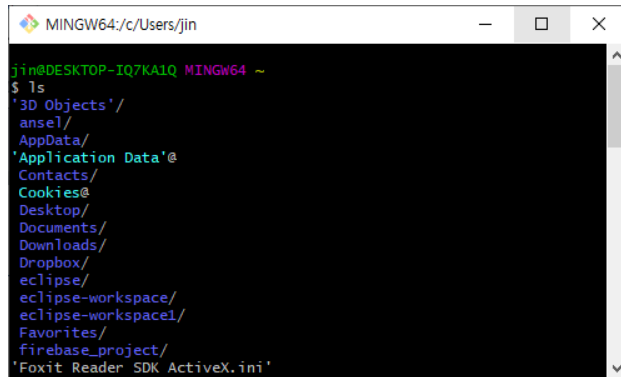
Git Bash

- 설정 절차

1. Git 설치

- 현재 디렉토리의 파일 목록 확인

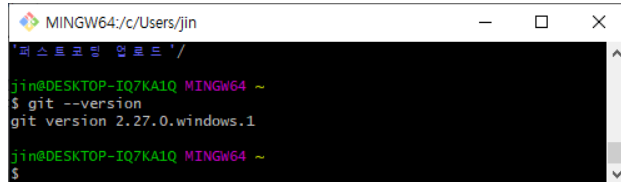
```
$ls
```



```
MINGW64:/c/Users/jin
jin@DESKTOP-IQ7KA1Q MINGW64 ~
$ ls
'3D Objects'/
ansei/
AppData/
'Application Data'@
Contacts/
Cookies@
Desktop/
Documents/
Downloads/
Dropbox/
eclipse/
eclipse-workspace/
eclipse-workspace1/
Favorites/
firebase_project/
'Foxit Reader SDK ActiveX.ini'
```

- git 버전 확인

```
$git --version
```



```
MINGW64:/c/Users/jin
'프로젝트명 업로드 '/'
jin@DESKTOP-IQ7KA1Q MINGW64 ~
$ git --version
git version 2.27.0.windows.1
jin@DESKTOP-IQ7KA1Q MINGW64 ~
$
```

- 설정 절차

1. Git 설치 : Git gui Clients

About
Documentation
Downloads
GUI Clients
Logos
Community



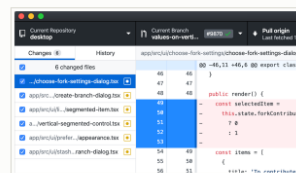
GUI Clients

Git comes with built-in GUI tools for committing ([git-gui](#)) and browsing ([gitk](#)), but there are several third-party tools for users looking for platform-specific experience.

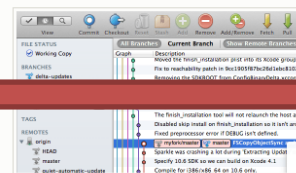
If you want to add another GUI tool to this list, just [follow the instructions](#).

All Windows Mac Linux Android iOS

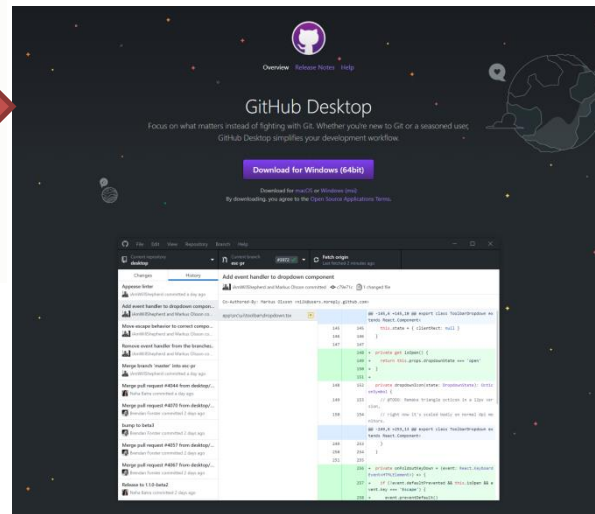
31 Windows GUIs are shown below ↓



GitHub Desktop
Platforms: Mac, Windows
Price: Free
License: MIT



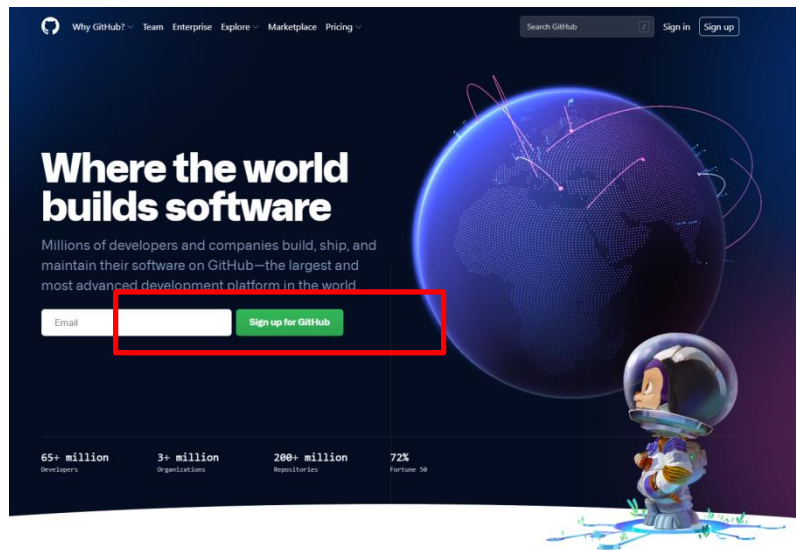
SourceTree
Platforms: Mac, Windows
Price: Free
License: Proprietary



• 설정 절차

2. GitHub 가입

- GitHub 홈페이지 → [Sign up for GitHub] 클릭 → 이름, 이메일, 패스워드 입력 → [Create an account] → 공개 버전 또는 개인 저장소 선택 → [Continue] → 설문 답변 또는 [skip this step] → [Start a Project] → 이메일 인증 안내 화면 → 메일의 [Verify email address] 클릭 → [Start a Project]




GitHub

[GitHub] Please verify your email address.

Almost done, @bitcampjava205!

To complete your GitHub sign up, we just need to verify your email address: [bitcamp.java205@gmail.com](https://github.com/users/bitcampjava205/emails/161245470/confirm_verification/acb45e7d7de60485e3576dbfcbf7611136abb282).

[Verify email address](#)

Once verified, you can start using all of GitHub's features to explore, build, and share projects.

Button not working? Paste the following link into your browser:
https://github.com/users/bitcampjava205/emails/161245470/confirm_verification/acb45e7d7de60485e3576dbfcbf7611136abb282

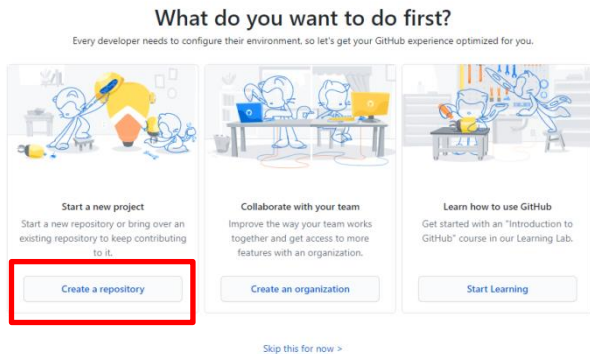
[Email preferences](#) · [Terms](#) · [Privacy](#) · [Sign in to GitHub](#)

You're receiving this email because you recently created a new GitHub account or added a new email address. If this wasn't you, please ignore this email.

GitHub, Inc. · 88 Colin P Kelly Jr Street · San Francisco, CA 94107

- 설정 절차

- 3. 원격 저장소 생성하기



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner *

bitcampjava205

Repository name *

Great repository names are short and memorable. Need inspiration? How about [miniature-octo-garbanzo](#)?

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

- Repository name : 저장소 이름 입력
- Description : 설명 입력(선택)
- Public, Private : 공개 또는 개인저장소 선택 → [Create repository]

- 설정 절차

- 3. 원격 저장소 생성하기

The screenshot shows the GitHub interface for a newly created repository named 'firstrepository' under the user 'bitcampjava205'. The repository is currently empty, with the 'main' branch selected. The file list shows two initial commits: '.gitignore' and 'README.md'. The README file is open, displaying the repository name 'firstrepository' and the description '처음만드는 원격저장소' (First remote storage). The right sidebar contains sections for 'About', 'Releases', and 'Packages', each with a brief description and a link to create a new item.

bitcampjava205 / firstrepository

Unwatch 1 Star 0 Fork 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file Code

bitcampjava205 Initial commit f565c58 now 1 commit

File	Commit	Time
.gitignore	Initial commit	now
README.md	Initial commit	now

README.md

firstrepository

처음만드는 원격저장소

About

처음만드는 원격저장소

Readme

Releases

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

Packages

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

- 설정 절차

- 4. 로컬 저장소 생성하기

- 디렉토리에 폴더 및 파일 생성
 - 생성한 디렉토리 확인

```
$ls
```

- 디렉토리로 이동

```
$cd 디렉토리 명
```

- Git 저장소 생성

```
$git init
```



```
MINGW64:/c/Users/jin/Documents/github/testdir
$ ls
aij202003/  classproject/  javabook/  testdir/
bitcampjin201904/  firstcoding/  kite/
jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/github
$ cd testdir
jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/github/testdir
$
```



```
MINGW64:/c/Users/jin/Documents/github/testdir
$ cd testdir
jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/github/testdir
$ git init
Initialized empty Git repository in C:/Users/jin/Documents/Github/testdir/.git/
jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/github/testdir (master)
$ |
```

- 설정 절차

- 6. 원격 저장소에 저장하기

- 저장소 등록

```
$git remote add orijin 복사한 원격 저장소의 주소
```

- Push : 로컬 저장소 내용을 원격 저장소로 보내기

```
$git push -u origin master Username for "https://github.com" : Username입력
```

- Password 입력 → 새로 고침(F5)로 저장완료 확인

- 설정 절차

- 7. 원격 저장소에 저장하기

- 테스트 문서에서 마우스 오른쪽 클릭 → Notepad → [인코딩]에서 [UTF-8로 변환] → 파일명 수정
 - 새로운 파일 생성 및 원격 저장소에 저장하기

```
$git add 파일명  
$git commit -m "메시지 내용"  
$git push -u origin master
```

- 새로운 문서가 저장된 것 확인

- 설정 절차

- 8. 원격 저장소 파일 가져오기

- [Clone or Download] → URL 복사
 - 새로운 저장소 생성 및 원격 저장소에 등록하기

```
$git init  
$git remote add orijin 복사한 원격 저장소의 주소
```

- 파일 가져오기

```
$ git pull origin master
```

- Git .gitignore File 적용하기

- .gitignore이란?

Project에 원하지 않는 Backup File이나 Log File , 혹은 컴파일 된 파일들을 Git에서 제외시킬 수 있는 설정 File이다.

- .gitignore 파일 만들기

항상 최상위 Directory에 존재해야 한다.

ex)

```
# Compiled class file
*.class

# Log file
*.log

# BlueJ files
*.ctxt

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

- Git .gitignore File 적용하기

- 문법

```
# : comments
```

```
# .a 파일들을 무시한다.
```

```
*.a
```

```
# .a 파일들을 무시하지만 lib.a 파일은 예외로 한다.
```

```
!lib.a
```

```
# subdir/TODO 파일이 아닌 현재 폴더 안의 TODO만 무시한다.
```

```
/TODO
```

```
# build/ 폴더의 모든 파일을 무시한다.
```

```
build/
```

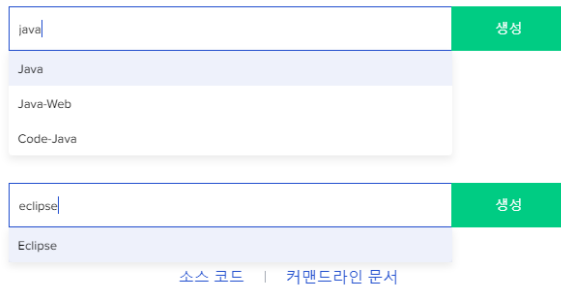
```
# doc/ 폴더 안의 모든 *.txt 파일은 무시하지만 doc/ 폴더 안의 서브 폴더 안의 *.txt 파일은 무시하지 않는다.
```

```
doc/*.txt
```

```
# doc/ 폴더 안과 서브 폴더 안의 모든 *.pdf 파일을 무시한다.
```

```
doc/**/*.pdf
```

- Git .gitignore File 적용하기
 - gitignore.io 사이트를 이용한 .ignore 내용 정의 (<https://www.toptal.com/developers/gitignore>)



<https://www.toptal.com/developers/gitignore/api/eclipse.java>