

깃(Git)이란 무엇인가?

# 깃(Git)은 무엇인가?

- 쉽게 말하면, SW 개발에서 소스 코드 관리에 주로 사용되는 분산 버전 관리 시스템
- 핵심 기능은 Version Control, Backup, Collaboration
- 혼자서 사용할 경우엔 클라우드 서비스라고 생각해도 됨

# 깃(Git)은 무엇인가?



초안



수정



최종



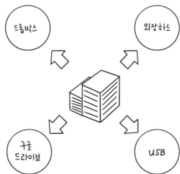
진짜 최종

...

## 버전 관리

- 평소에 문서를 작성하고 수정할 때 저장하는 파일 이름들
- 어떤 것을 수정했는지 기억할 수 없음
- 깃을 사용하면 언제 수정했는지, 어떤 것이 변경되었는지 기록

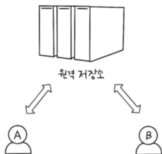
# 깃(Git)은 무엇인가?



## 백업하기

- 드롭박스, 구글 드라이브, 클라우드 서비스 등과 같은 역할
- 깃허브(GitHub) <https://github.com/>

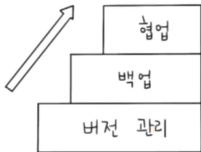
# 깃(Git)은 무엇인가?



## 협업하기(Collaboration)

- 팀 프로젝트를 할 때 유용하게 사용
- A가 작업을 하고 원격 저장소에 올린 것을 B가 내려받아 작업하고 다시 원격 저장소에 올림

# 깃(Git)은 무엇인가?

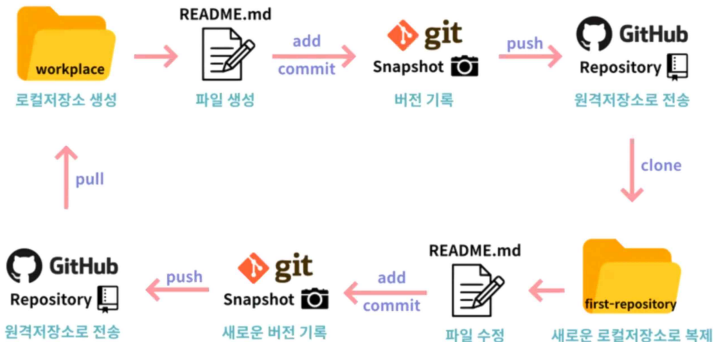


- 개발자가 아니라면 버전 관리, 백업만 사용해도 충분
- 개발자라면 협업까지 알아두면 좋음

# 깃(Git) 종류

- 깃허브 데스크톱 (<https://github.com/>)
- Git(<https://git-scm.com/>)
- CLI (Command Line Interface)
- 토터스 깃(TortoiseGit) – window 전용 프로그램
- 소스트리(SourceTree) – 깃의 기본 기능부터 고급 기능까지 사용할 수 있는 프로그램

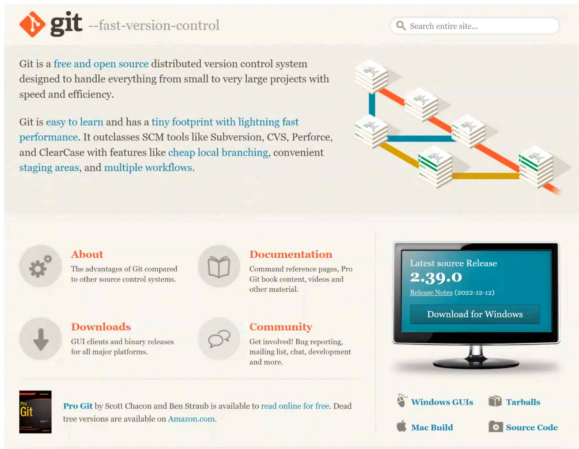
# 깃과 깃허브 간단한 사용법





# Git 설치

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



The screenshot shows the Git website homepage. At the top, the Git logo is followed by the tagline "--fast-version-control". A search bar is located in the top right corner. The main content area features a description of Git as a free and open source distributed version control system, highlighting its speed and efficiency. It also mentions that Git is easy to learn and has a tiny footprint with lightning fast performance, outclassing other SCM tools like Subversion, CVS, Perforce, and ClearCase. To the right of the text is a diagram illustrating Git's distributed nature with multiple stacks of code blocks connected by lines. Below the main text, there are four sections: "About" (advantages compared to other systems), "Documentation" (command reference, book content, videos), "Downloads" (GUI clients, binary releases), and "Community" (bug reporting, mailing list, chat). On the right side, there is a section for the "Latest source Release 2.39.0" with a "Download for Windows" button. At the bottom, there is a section for "Pro Git" by Scott Chacon and Ben Straub, available for free on Amazon.com. The footer includes links for "Windows GUIs", "Tarballs", "Mac Build", and "Source Code".

**git** --fast-version-control

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient staging areas, and **multiple workflows**.

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Downloads**  
GUI clients and binary releases for all major platforms.

**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

**Latest source Release**  
**2.39.0**  
Release Notes (2022-12-12)  
[Download for Windows](#)

**Pro Git** by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on [Amazon.com](#).

[Windows GUIs](#) [Tarballs](#)  
[Mac Build](#) [Source Code](#)

# Git 설치

## 윈도우에 깃 설치하기

이제에서는 리눅스 명령을 사용하기 때문에 윈도우에 깃을 설치하면 리눅스 명령을 사용할 수 있도록 깃 배시(Git Bash)라는 프로그램이 함께 설치됩니다.

1. 웹 브라우저에서 <https://git-scm.com/> 사이트로 접속하면 운영체제에 따라 프로그램을 내려받을 수 있는 화면이 나타납니다. 화면 오른쪽의 [Download 2.2x.x for Windows]를 누르면 다음 화면으로 이동하면서 자동으로 파일을 내려받기 시작합니다. 내려받기가 끝나면 파일을 실행하세요.



크롬 브라우저에서 내려받을 경우 '컴퓨터를 손상시킬 수도 있는 파일'이라는 경고 메시지가 표시될 수 있습니다. [계속]을 눌러 진행하면 됩니다.

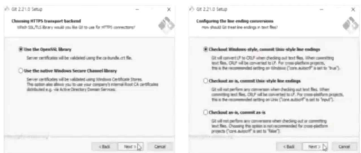
2. 첫 화면에서는 라이선스 정보를 확인합니다. 그다음 [Next]를 눌러 깃에서 설치할 구성요소를 선택합니다. 여기에서는 기본 값 그대로 진행하겠습니다. [Next]를 누르세요.



3. 깃에서 사용할 기본 편집기를 선택합니다. 기본 값으로 뱀(Vim)이 선택되어 있을 겁니다. 그대로 [Next]를 눌러 다음 화면으로 넘어간 다음 커맨드 라인에서 어떤 방법으로 깃을 사용할지 선택합니다. 기본 값 'Get from the command line and also from 3rd-party software'가 선택된 상태 그대로 [Next]를 누르세요.



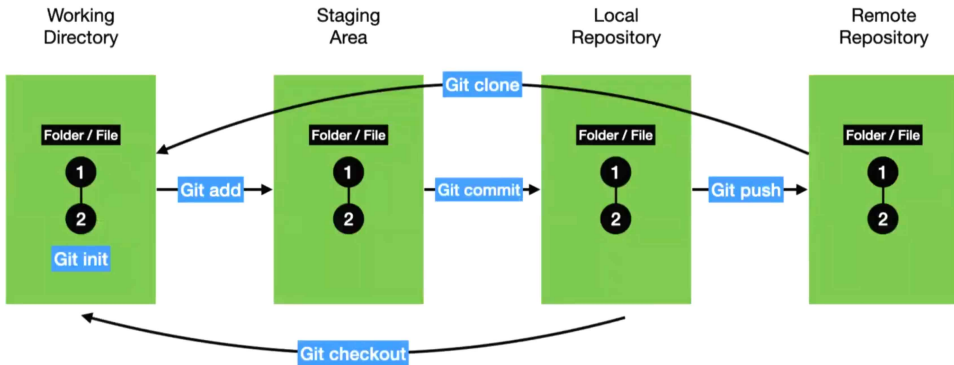
4. 보안 서버에 접속하기 위한 방법을 선택합니다. 기본 값 'Use the OpenSSL library'가 선택된 상태로 [Next]를 누르세요. 다음 화면에서는 텍스트 파일에서 줄 끝부분을 어떻게 처리할 것인지 선택합니다. 기본 값 'Checkout Windows-style, commit Unix-style line endings'가 선택된 상태로 [Next]를 누르세요.



5. 터미널 에뮬레이터를 선택합니다. 기본 값 'User Window's default console window'는 윈도우의 명령 프롬프트 창을 사용한다는 뜻입니다. 그대로 [Next]를 누르세요. 다음으로 기타 옵션을 선택하는 화면이 나타나면 기본 값 그대로 두고 [Install]을 눌러 설치를 시작합니다.



# 깃 사용하기



# 깃 저장소 환경 설정하기

git config : 깃에서 사용자 정보 설정

--global : 현재 컴퓨터에 있는 모든 저장소에서 같은 사용자 정보를 사용하도록 설정

- \$ git config --global user.name "사용자 명"
- \$ git config --global user.email "사용자 이메일"

```
jhjun@DESKTOP-TQ4E75V MINGW64 ~/Desktop/hello-git (master)
$ git config --global user.name "jhwannabe"

jhjun@DESKTOP-TQ4E75V MINGW64 ~/Desktop/hello-git (master)
$ git config --global user.email "jhjun0328@naver.com"
```

# 깃 버전 사용하기



- 작업 트리 : 파일 수정, 저장 등의 작업을 하는 디렉토리
- 스테이지 : 버전으로 만들 파일이 대기하는 곳 (staging area)
- 저장소 : 스테이지에서 대기하고 있던 파일들을 버전으로 만들어 저장하는 곳

# 깃 저장소 만들기

## 1. git init : 깃 초기화하기

- \$ mkdir hello-git
- \$ cd hello-git
- \$ git init

```
jhhjun@DESKTOP-TQ4E75V MINGW64 ~/Desktop (master)
$ mkdir hello-git

jhhjun@DESKTOP-TQ4E75V MINGW64 ~/Desktop (master)
$ cd hello-git/

jhhjun@DESKTOP-TQ4E75V MINGW64 ~/Desktop/hello-git (master)
$ ls -al
total 4
drwxr-xr-x 1 jhhjun 197609 0 Dec 19 16:15 ./
drwxr-xr-x 1 jhhjun 197609 0 Dec 19 16:15 ../

jhhjun@DESKTOP-TQ4E75V MINGW64 ~/Desktop/hello-git (master)
$ git init
Initialized empty Git repository in C:/Users/jhhjun/Desktop/hello-git/.git/

jhhjun@DESKTOP-TQ4E75V MINGW64 ~/Desktop/hello-git (master)
$ ls -al
total 8
drwxr-xr-x 1 jhhjun 197609 0 Dec 19 16:16 ./
drwxr-xr-x 1 jhhjun 197609 0 Dec 19 16:15 ../
drwxr-xr-x 1 jhhjun 197609 0 Dec 19 16:16 .git/
```

# 깃 버전 사용하기



- `$ git status`
- `$ git add hello.txt`
- `$ git commit -m “커밋 메세지”`
- 동시에 하려면 `$ git commit -am “커밋 메세지”`
- `$ git log` : 저장소에 저장된 버전을 확인 (history)

# 커밋 내용 확인

- \$ git log : 커밋 기록 자세히 살펴보기

```
MINGW64/c/Users/funco/hello-git
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ git log
commit d1eb0361d7592f0195fe16e41d4ccbe6b291c5d (HEAD -- master)
Author: Kyunghee <jump2dev@gmail.com>
Date: Fri Jul 26 23:22:45 2019 +0900
    message2
commit ac5b4fa5e2813c1e2a7934f964fcf27dc90f1f88
Author: Kyunghee <jump2dev@gmail.com>
Date: Fri Jul 26 16:22:32 2019 +0900
    message1
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ |
```

- \$ git diff : 변경사항 확인하기

```
MINGW64/c/Users/funco/hello-git
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ git diff
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
diff --git a/hello.txt b/hello.txt
index 1191247..0b66db0 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,2 +1,2 @@
1
c2
c two
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ |
```



# 커밋 내용 확인

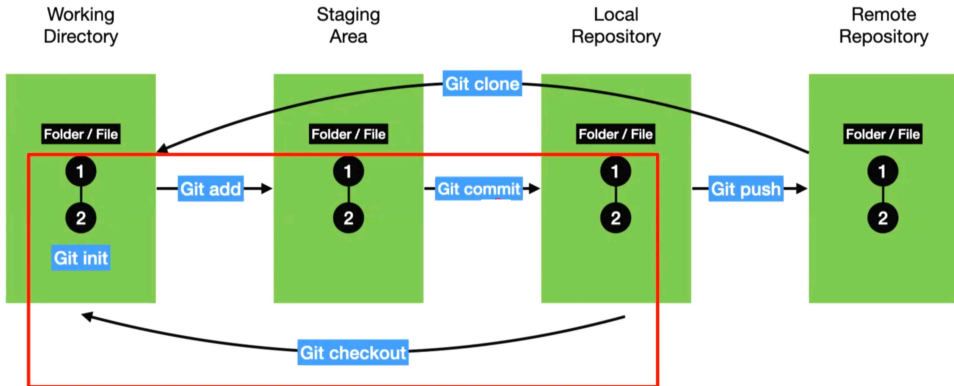
- \$ git checkout 파일명 : 변경하기 이전으로 되돌림

```
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ git status
on branch master
changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ |
```

# 깃 사용하기



# 깃허브 레포지토리에 Push

깃허브 계정 생성 → 새 레포지토리 생성 → 해당 레포지토리 주소 복사 (github.com/계정/레포지토리.git)

\$ git remote add origin 레포지토리 주소

\$ git push (-u origin master) → ()는 처음 한번만 사용

- origin : 원격 저장소 이름

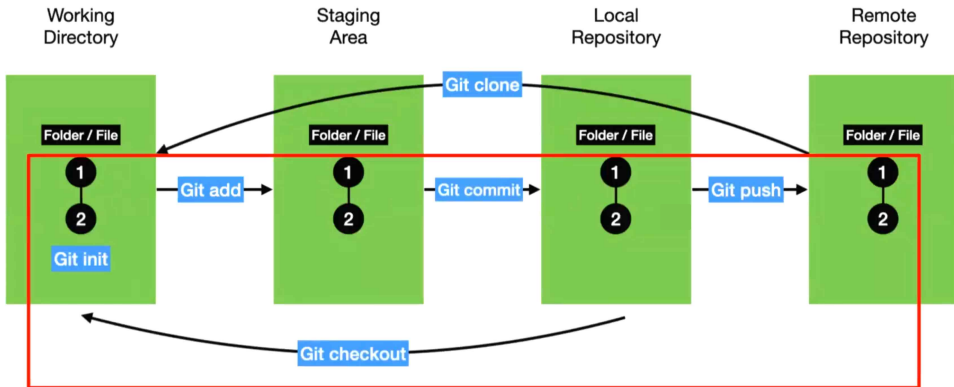
- master : 브랜치 이름 = default/main branch

## 깃허브 레포지토리에 Pull

```
$ git pull origin master
```

origin(원격 저장소)의 내용을 master 브랜치로 가져옴

# 깃 사용하기



# Git Clone

\$ git clone 레포지토리주소

현재 위치한 디렉토리에 clone한 소스가 복사됨.

\$ git clone 레포지토리주소 디렉토리명

설정된 디렉토리에 clone한 소스가 복사됨. (없으면 자동 생성)

# Git Clone 이후 다시 Push / Pull

Push

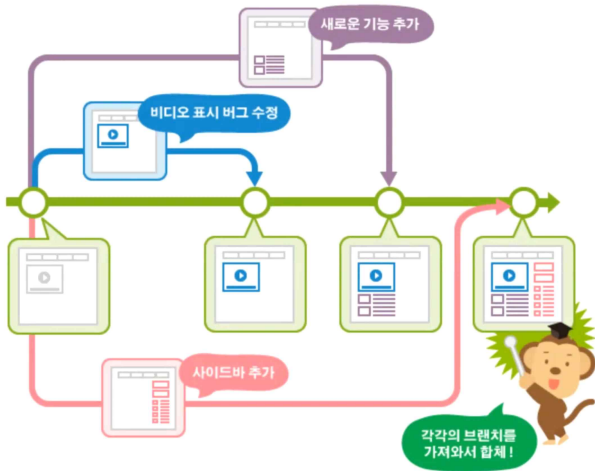
```
$ git commit -am "메세지"
```

```
$ git push
```

Pull

```
$ git pull
```

# Git Branch





# Git Branch

\$ git branch : 브랜치를 만들거나 확인

\$ git branch 브랜치이름



```
MINGW64/c/Users/funco/manual
funco@DESKTOP-CKTPGVC MINGW64 ~/manual (master)
$ git branch apple

funco@DESKTOP-CKTPGVC MINGW64 ~/manual (master)
$ git branch
  apple
* master

funco@DESKTOP-CKTPGVC MINGW64 ~/manual (master)
$ |
```

\$ git branch -d 브랜치 이름 : 브랜치 삭제

# Git Branch

\$ git checkout 브랜치이름 : 브랜치 이동하기

\$ git log A..B : A에는 없고 B에만 있는 커밋을 보여줌

# Git Branch

\$ git branch : 브랜치를 만들거나 확인

\$ git branch 브랜치이름



```
MINGW64/c/Users/funco/manual
funco@DESKTOP-CKTPGVC MINGW64 ~/manual (master)
$ git branch apple

funco@DESKTOP-CKTPGVC MINGW64 ~/manual (master)
$ git branch
  apple
* master

funco@DESKTOP-CKTPGVC MINGW64 ~/manual (master)
$ |
```

\$ git branch -d 브랜치 이름 : 브랜치 삭제

# Git Branch

\$ git merge 브랜치이름 : 브랜치 병합

--no-edit : 브랜치를 병합할 때 자동으로 편집기가 실행되면서 커밋 메시지를 추가 작성할 수 있는데, 편집기 창을 열지 않고 깃에서 지정하는 커밋 메시지를 그대로 사용하는 옵션