

Git Tutorial

lamCoder 0x24

Git이란 무엇인가?

Git Tutorial



위키백과: Git은 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 **분산 버전 관리 시스템**이다

Git이란 무엇인가?

Git Tutorial

Git과 Github를 헷갈리면 안된다!

Git은 버전 관리 프로그램이고

Github는 Git을 지원하는 원격 저장소일 뿐이다.

Git은 gui를 지원한다. 다만 cui가 편할 뿐이다.



Github Desktop

Git을 왜 쓰는가?

Git Tutorial

편해서, Github가 있기 때문에

Git 설치하기

Git Tutorial

terminal에 다음을 입력한다.

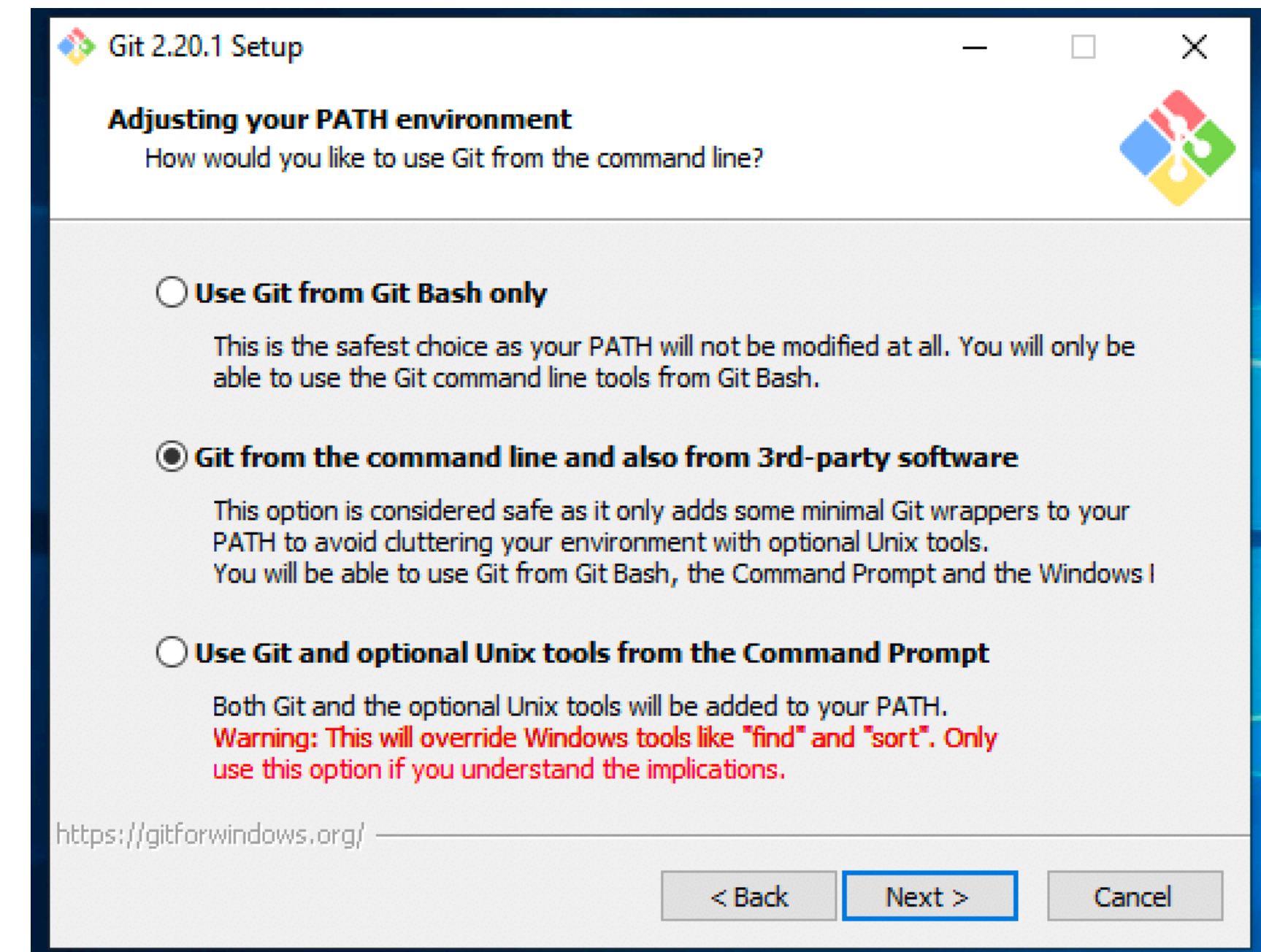
```
$ sudo apt-get install git
```

```
retr0@ubuntu:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
 lib32ncurses5 lib32tinfo5 lib32z1 lib32z1-dev linux-headers-4.15.0-45
 linux-headers-4.15.0-45-generic linux-headers-4.15.0-46
 linux-headers-4.15.0-46-generic linux-headers-4.15.0-47
 linux-headers-4.15.0-47-generic linux-headers-4.15.0-50
 linux-headers-4.15.0-50-generic linux-headers-4.15.0-51
 linux-headers-4.15.0-51-generic linux-headers-4.8.0-36
 linux-headers-4.8.0-36-generic linux-image-4.15.0-45-generic
 linux-image-4.15.0-46-generic linux-image-4.15.0-47-generic
 linux-image-4.15.0-50-generic linux-image-4.15.0-51-generic
 linux-image-4.8.0-36-generic linux-image-extra-4.8.0-36-generic
 linux-modules-4.15.0-45-generic linux-modules-4.15.0-46-generic
 linux-modules-4.15.0-47-generic linux-modules-4.15.0-50-generic
 linux-modules-4.15.0-51-generic linux-modules-extra-4.15.0-45-generic
 linux-modules-extra-4.15.0-46-generic linux-modules-extra-4.15.0-47-generic
 linux-modules-extra-4.15.0-50-generic linux-modules-extra-4.15.0-51-generic
Use 'sudo apt autoremove' to remove them.
Suggested packages:
 git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
 gitweb git-arch git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
 git
0 upgraded, 1 newly installed, 0 to remove and 103 not upgraded.
Need to get 3,176 kB of archives.
After this operation, 24.1 MB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 git amd64 1:2.7.4-0ubuntu1.6 [3,176 kB]
Fetched 3,176 kB in 3s (854 kB/s)
Selecting previously unselected package git.
(Reading database ... 528459 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.7.4-0ubuntu1.6_amd64.deb ...
Unpacking git (1:2.7.4-0ubuntu1.6) ...
Setting up git (1:2.7.4-0ubuntu1.6) ...
```

Linux

Windows

Git 홈페이지에서 설치한다.



이 부분만 세팅을 잘하면 된다.

*Mac은 이미 설치되어 있다.

Git 사용해보자 - 새로운 저장소 만들기

Git Tutorial

```
$ git init
```

위 명령어는 현재의 디렉토리를 버전 관리를 할 수 있도록 초기화를 해준다.

구체적으로는 .git이라는 폴더를 만들고 그 안에 여러 파일을 만든다.

Git 사용해보자 - 다른 저장소에서 받아오기

Git Tutorial

```
$ git clone /path/to/clone
```

```
$ git clone <url-to-clone>
```

위 명령어는 다른 저장소에 저장되어 있는 다른 파일들을 모두 받아오는 커맨드이다.

Github에서 받아오려면 그 url을 복사해서 넣으면 된다.

Git 사용해보자 - 파일 만들기 및 추가하기

Git Tutorial

```
$ touch README.md           ← Linux 커맨드이다  
$ git add <file-name>  
$ git add .
```

첫번째 명령어는 README.md라는 파일을 만든다. 다른 방법으로 파일을 만들어도 된다.

그 다음 버전 관리 시스템에 추가하고 싶다면 두번째 명령어를 쓰게 되면 그 파일만 추가된다.

모든 파일을 추가하고 싶다면 마지막 명령어를 쓰면 된다.

Git 사용해보자 - 파일을 확정하기

Git Tutorial

```
$ git commit -m "message to commit"
```

전 commit 후 추가된 파일, 전 파일에서의 변경사항 등을 저장한다.

Github에 올라갈 경우 파일 이름 옆에 뜨게 된다.

메세지를 쓰는 방법은 다음을 참고하면 된다.

<https://meetup.toast.com/posts/106>

Git 사용해보자 - Git의 구조

Git Tutorial

Working Directory



`git add`

Staging Area



`git commit`

Repository


Git 사용해보자 - 나뭇가지

Git Tutorial

```
$ git branch
```

만약 두명 이상의 사람들이 개발을 같이 하려고 할 때 등
원래의 코드를 건들고 싶지 않을 때 사용하는 것이 branch이다.
말 그대로 나뭇가지를 새로 만들어서 수정을 하면 따로 저장되는 것이다.
처음 branch는 항상 master로 지정되어 있다.
위 명령어는 현재 속한 branch를 알려준다.

Git Tutorial



각 화살표는
앞에서부터 변화한
과정을 담고 있다.

Git 사용해보자 - 나뭇가지 만들고 이동하기

Git Tutorial

```
$ git branch <new-name>
```

```
$ git checkout <new-name>
```

```
$ git checkout -b <new-name>
```

먼저 첫번째 명령어를 이용해서 새로운 branch를 만든다.

그 뒤 새로운 branch로 이동을 해야 하는데, 이는 두번째 명령어를 이용해서 가능하다.

위 과정을 한번에 하고 싶으면 마지막 명령어를 사용해서도 가능하다.

다른 branch로 이동할때 그곳의 내용이 다르면 파일이 알아서 바뀌게 된다.

Git 사용해보자 - 나뉘어있는 브랜치 합치거나 삭제하기

Git Tutorial

```
$ git merge <other-branch>
```

```
$ git branch -d <branch-name>
```

Git은 첫번째 명령어를 사용했을 때 자동으로 현재 branch와 주어진 branch를 합치려고 한다.

만약 실패하면 직접 수정하고 파일을 다시 추가(add)해주면 된다.

두번째 명령어를 사용하게 되면 branch가 삭제된다.

Git 사용해보자 - 외부 저장소 사용하기

Git Tutorial

```
$ git remote add origin url.to/push/to  
$ git remote rm origin
```

첫번째 명령어는 외부저장소를 연결하는 이름을 `origin`으로 지정한 것이다.

그리고 두번째 명령어는 origin이 담고 있는 내용을 지운 것이다.

Git 사용해보자 - 파일을 보내기

Git Tutorial

```
$ git push origin master
```

이는 master branch에 있는 커밋 내용을 origin 저장소로 보낸다는 의미를 갖고 있다.

저장소에서 원래 갖고 있던 내용과 내가 갖고 있는 내용이 다르다면 에러가 난다.

만약 다 지우고 강제로 푸시를 하고 싶다면 인자로 -f 를 붙이면 된다.

Git 사용해보자 - .gitignore

Git Tutorial

```
# : comments
# no .a files
*.a
# but do track lib.a, even though you're
# ignoring .a files above
!lib.a
# only ignore the TODO file in the current
# directory, not subdir/TODO
/TODO
# ignore all files in the build/ directory
build/
# ignore doc/notes.txt, but not doc/server/
# arch.txt
doc/*.txt
# ignore all .pdf files in the doc/ directory
doc/**/*.pdf
```

.gitignore은 git에 추가하지 않을 파일들을 정하는 것으로, 디렉토리 안에 같이 만들어 두면 된다. 다음과 같이 파일을 만들면 된다.

```
$ touch .gitignore
```

그러면 왼쪽의 문법을 이용해서 작성을 해주면 된다. 보통 빌드한 파일, ide 파일 등들을 적어주면 된다.