|  |  |
| --- | --- |
| **Program / Project Name:** | Git |
| **Checklist / Template Completed by:** | TBD |
| **Date Completed:** | Click here to enter a date. |

**Tips for Git**

Revision: 0.1

**Detailed Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Editor(s)** | **Description of change** |
| 0.1 | 20-MAY-2019 | Sang-Gu Kang | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1 Install 4](#_Toc21592078)

[1.1 Git 4](#_Toc21592079)

[1.2 TortoiseGit 4](#_Toc21592080)

[1.2.1 Download 4](#_Toc21592081)

[1.2.2 Install 4](#_Toc21592082)

[2 사용법 10](#_Toc21592083)

[2.1 Git Bash 10](#_Toc21592084)

[2.1.1 Command 10](#_Toc21592085)

[3 Useful Tip 15](#_Toc21592086)

[3.1 Overlay Icon 보이게 하기 15](#_Toc21592087)

# Install

## Git

Download 후 그냥 next, next.

## TortoiseGit

### Download



TortoiseGit 설치파일과 언어팩을 다운로드한다.





### Install











언어팩 설치



한글 설정을 놓쳤다면 settings에서 설정해 준다.





# 사용법

## Git Bash

### Command

git < > [ ] …

#### add

git add <file> or <directory path>

파일을 새로 추적할 수 있다(untracted -> tracted). git add 명령은 파일을 새로 추적할 때(새로 만든 파일을 add하여 tracted상태로 만들고자 할 때)도 사용하고 수정한 파일(이미 tracted된 파일이지만 modified된 상태의 파일)을 Staged 상태로 만들 때도 사용한다. Merge할 때 충돌난 상태의 파일을 Resolve 상태로 만들때도 사용한다.

#### branch

새 브랜치 생성.

#### checkout

다른 브랜치로 이동할 때 사용한다.

[-b]: 브랜치를 만들면서 checkout까지 한번에 한다.  
git checkout -b iss53

#### clone

[url]



#### commit

[--amend]:

#### config

[--global] 사용자 이름과 이메일을 설정한다.

[--list] 설정한 모든 것을 보여준다.



#### diff

단순히 파일이 변경됐다는 사실이 아니라 어떤 내용이 변경됐는지 살펴보려면 git status 대신 git diff 명령을 사용해야 한다. Patch처럼 어떤 라인을 추가했고 삭제했는지가 궁금할 때 사용한다.

이 명령은 working directory에 있는 것과 Staging Area에 있는 것을 비교한다. 그래서 수정하고 아직 Stage하지 않은 것을 보여준다.

만약 commit하려고 Staging Area에 넣은 파일의 변경부분을 보고 싶으면 git diff –staged 옵션을 사용한다. 이 명령은 저장소에 commit한 것과 Staging Area에 있는 것을 비교한다.



Commit 하기 전 마지막으로 실수한 것이 없는지 확인할 수 있는 기회를 제공한다.

#### fetch

리모트 저장소의 데이터를 모두 로컬로 가져오지만, 자동으로 Merge하지는 않는다(git pull 명령 사용은 merge시킬 수 있다).

#### init

.git이라는 하위 디렉토리를 만든다. .git 디렉토리에는 저장소에 필요한 뼈대 파일(Skeleton)이 들어 있다.

#### push

#### log

저장소의 commit 히스토리를 시간 순으로 보여줌.

[-p]: 각 commit의 diff 결과를 보여준다.

[-2]: 최근 두 개의 결과만 보여준다.

[--stat]: 각 commit의 어떤 파일이 수정됐는지, 얼마나 많은 파일이 변경됐는지, 또 얼마나 많은 라인을 추가하거나 삭제했는지의 통계 정보를 보여준다.

[--pretty=oneline]: 각 commit을 한 라인으로 보여준다.

[--decorate]: 브랜치가 어떤 커밋을 가리키는지 알 수 있다. git log –oneline --decorate

기타 다른 많은 옵션들은 pro git 등을 참조한다.



#### mv

#### remote

remote 저장소 확인.

[-v] 단축 이름과 URL을 함께 볼 수 있다.

[show <리모트 저장소 이름>] 예)git remote show origin

[rename]

[remove]

#### rm

staged된 파일을 unstaged하게 할 때 사용한다.

  
  


#### show

태그 정보와 커밋 정보를 모두 확인할 수 있다.

#### status

파일의 상태를 확인한다.

[-s] or [--short] 현재 변경한 상태를 짤막하게 보여준다.

#### tag

[-a] annotated tag 만들기.

## 상황별 용도

### 되돌리기

* 파일 상태를 Unstaged로 변경  
  git reset HEAD <file name>
* Modified 파일 되돌리기(수정된 파일 되돌리기)  
  git checkout -- <file name>  
  매우 위험한 명령. 원래 파일로 덮어쓰기 때문에 수정한 내용은 전부 사라짐. 변경한 내용을 쉽게 버릴 수는 없고 하지만 당장은 되돌려야만 하는 상황이라면 Stash와 Branch를 사용한다.

### 리모트 저장소

* 리모트 저장소 확인하기  
  git remote -v
* 리모트 저장소 추가하기  
  ???
* 리모트 저장소를 Pull 하거나 Fetch하기  
  git fetch [remote-name]
* 리모트 저장소에 push하기  
  git push origin master
* 리모트 저장소 살펴보기  
  git remote show [remote 저장소 이름] 예)git remote show origin
* 리모트 저장소 이름을 바꾸거나 리모트 저장소를 삭제하기  
  git remote rename pb paul  
  git remote remove paul

### 태그

* 태그 조회하기  
  git tag
* 태그 붙이기  
  ?
* 나중에 태그하기  
  ?
* 태그 공유하기  
  ?
* 태그를 checkout하기  
  ?

### 브랜치

* 새 브랜치 생성하기  
  git branch <branch-name>  
  git branch testing
* 브랜치 이동하기  
  git checkout <branch-name>  
  git checkout testing

# Useful Tip

## Overlay Icon 보이게 하기

레지스트리의 우선순위를 변경해주면 된다. 15개 까지만 표시해 준다고 한다.

-. regedit.exe 실행



-. 아래 위치로 이동 HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers



-. Tortoise 항목들의 이름을 바꾸어 상위로 이동시킨다.(앞에 공백을 추가하면 됨)

-. 재부팅 하면 Overlay Icon이 보인다.

-. 만약 보이지 않는다면, 백업 하기 위해 내보내기를 한 후, reg 파일이 생성되면 더블 클릭하고 재부팅한다.

