|  |
| --- |
| Git and Git hub Guide |

18. 5. 20 last edited

Kiseok

- Contents

|  |
| --- |
| - The purpose of Git : Source control  - What is git?  - Git terminology  - Installation and setup |

- The purpose of Git : Source control

🡪 why source control

- Backup / Archive

- Versioning / History

- Undo Changes

- Isolation of changes

- Collaboration

- Review code

- Share code / Open source

- Source control options : Centralized(Subversion, CVS) / Distributed(Git) 🡪 network에 connect 안되어 있더라도 충분히 할 수 있다.

- What is git?

- Distributed source control system

- Massively scales

- most operations are local 🡪 can work without the connection to the network

Git terminology

- Repository contains files, history, config

- Three states of Git

- Working directory

- Staging area - pre-commit holding area

- Commit - Git repository(history) 🡺 repository(hidden .git folder)

- Remote repository (GitHub) : 4th state

- master branch

- Installation and setup

- git-for-windows.github.io

- sign up : KiseokLee lkipp@snu.ac.kr

- Master main commands : quick start

|  |
| --- |
| Use gitbash  리눅스랑 똑같네  - pwd : present working directory  - cd ~~  - mkdir project  🡪 user name 바로 밑에 project directory를 만들면 편하다.  ls  ls -al 🡪 .git repository folder도 보인다.  clear  exit  git status  rm -rf directory\_you\_wish\_to\_remove  ~ ## home directory  # 이름 바꾸기 mv ~~~ ~~~~  $ unzip ~/Downloads/initializr-verekia-4.0.zip  $ mv initializr/ web-project ## rename  # user information 바꿀때  np ~/.gitconfig  ## track file을 보는 법  git ls-files  $ cd ../../..  - user name이랑 이메일을 따로 입력해주는 것이 좋다.  - 복사는 control insert, 붙여넣기는 shift insert  $ git config --global user.name 'Kiseok Lee'  $ git config --global user.email 'lkipp@snu.ac.kr'  $ git config --global --list  - Cloning a repository  : github에서 repository를 만든 다음에 local하게 연결시키고 싶은 것이다.    - clone option을 들어간다.  link를 복사해서  $ git clone URL~~~  - cd github-demo  - git statue를 해보자.  $ git status  On branch master  Your branch is up to date with 'origin/master'.  nothing to commit, working tree clean  - 새로운 파일을 만들어 보자.  $ echo "Test Git Quick Start demo" >> start.txt  🡪 이렇게 하면 start.txt file에 생긴다. echo 뒤에 거는 file안에 들어갈 text file이다.  - 그 파일에 들어가 보자.  $ cat start.txt  - 이 때 이 파일의 git status를 보게 되면  Untracked file이라고 뜬다. 🡪 working directory에는 있는데  $ git add start.txt ## 이제는 staging area에 있게 된다.    - 이제 repository에 commit을 해보자.  $ git commit -m ‘Adding start text file’  다시  git status하면 working tree clean이라고 나온다.    🡪 이제 local repository에 있게 된다.  - 이거를 Github에 넣고 싶다면 Push를 하면 된다.  $ git push origin master  \* origin refers to the Github copy of our repository  \* master refers to our default and only branch in the repository  🡪 아이디 비번을 입력하면 완료!    🡪 github에 들어가보면 들어가 있다. |



- Text editor to work seamlessly with git.

- configurate Notepad++

- Notepad++가 있는 file directory 위치를 알아낸다. 복사한다.

- 내 컴퓨터 오른쪽 클릭 🡪 속성 🡪 고급시스템설정 🡪 환경변수 🡪 path 새로만들기.

- 이제 bash에서 $ notepad++하면 뜬다.

|  |
| --- |
| np만 하면 notepad++가 나오게끔 해보기!!  step 1) git bash에서  notepad++ .bash\_profile 🡪 yes 클릭  step 2) notepad++에서  alias np=’notepad++.exe -multiInst -nosession’  alt w  alt f4  🡪 이제 np만 하면 notepad++를 안 써도 된다. |

- notepad++를 default editor로 하기 위한 방법

$ cat ~/.gitconfig

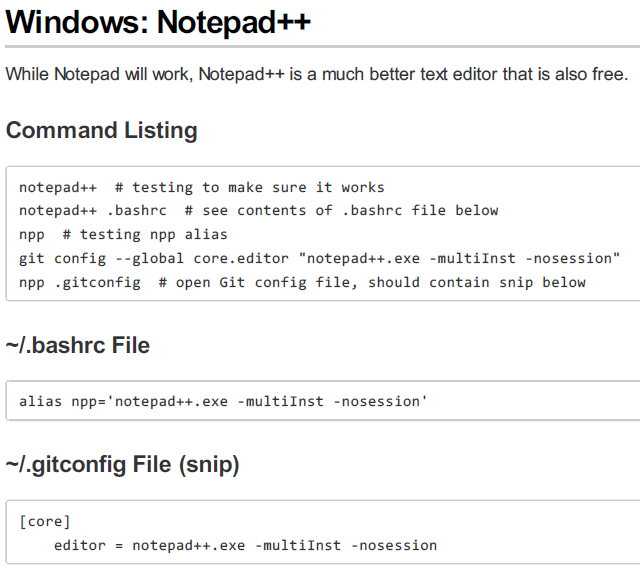
$ git config --global core.editor "notepad++.exe -multiInst -nosession"

🡪 잘 됐는지 확인하려면

$ git config --global --list

$ git config --global -e ## default editor로 열게 된다.

- 요약 :



- Basic command

순서

1. Start a Project :

1) Fresh?

2) Existing source

3) GitHub project(Fork and clone)

2. Basic Workflow (add, commit, push & pull)

3. Working with File (rename, move & delete)

4. History and Aliases

5. Ignore unwanted files

1. Start a new project

1) Fresh start

$ git init fresh-project

hipsum.co라는 사이트에 가서 text를 복사한다.(beer me)

np hipster.txt ## notepad에 연결되며 파일이 만들어진다.

그 다음에 복붙한다.

- add to staging area

git add hipster.txt

- commit하려면

git commit

🡺 notepad에서 commit message를 위에 치면 된다.

- 다시 열려면

np hipster.txt

- 만약에 지우려면

cd ..

pwd

ls

rm -rf fresh-project/

2) Adding Git to an existing project

사이트 : <http://www.initializr.com/>

$ unzip ~/Downloads/initializr-verekia-4.0.zip

$ mv initializr/ web-project ## rename

- 여기서 working directory를 열려면

$ git init (굉장히 중요한 step)

- staging area로 add하자

$ git add . ## .는 모든 파일을 add한다.

- commit

$ git commit -m "My first commit, inline"

- git control을 안하고 싶으면

$ rm -rf .git

- project를 지우고 싶으면

$ rm -rf web-project

3) Start from existing GitHub project

사이트 : <https://github.com/scm-ninja/starter-web>

- fork하자

fork을 클릭하면 된다!! that easy!!

🡪 we have our own personal copy of that repository

- clone하자

$ git clone https://github.com/KiseokLee/starter-web.git

쉽네!!

2. Basic workflow

dummy text

meettheipsums.com

hipsum.co

git add ~~

git commit -m “message”

(중요!!!)

Always do a pull, before doing a push

git pull origin master # origin 🡪 remote depository, # branch : master

🡪 remote repository에 있는 걸 다시 가져온다. push를 하기 전에 하는 게 좋다. old version으로 덮어씌울 수 있으니까. 🡺 team이 있으면!!!

- add and commit process를 shortcut하고 싶으면??

$ git commit -am "Adding more ipsum text"

\* a는 at the stage

\* m은 message

- 이것은 track file만 가능하다. (track이란 staging area에 있는 것을 의미한다.)

## track file을 보는 법

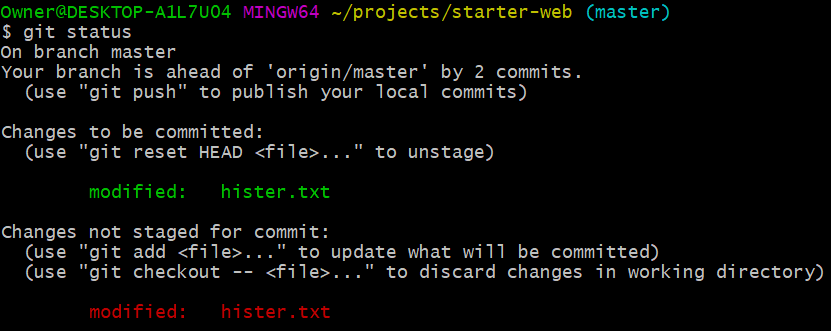
git ls-files

- newfile.txt를 만들면 track file에 안 뜬다.

- git add newfile.txt하면 git ls-files 에 뜬다.

3. Working with files

3.1. Editing file



- staging area에 있고 나서 다음에 hister.txt를 edit했을 때, git status를 하면 위와 같이 뜬다.

그러면 다시 add해야하는데, 이것은 바뀐 내용을 업데이트하는 것(adding the chages to the git staging area)이라고 생각하면 된다.

3.2. Recursive add

# 만약에 폴더에 들어가면서 계속 add를 하고 싶으면

$ git add .

3.3. Back out changes : I didn’t mean to do that

stage에 있을 때,

$ git reset HEAD level1-file.txt

Working directory에서도 그 change를 없애고 싶을 때,

$ git checkout -- level1-file.txt

🡪 수정했던 내용이 사라졌다!!

3.3. rename

git mv level3-file.txt level3.txt

🡪 stage했다.

commit하자.

$ git commit -m 'renaming level3 file'

- 만약에 bash command사용하면

$ mv level2-file.txt level2.txt

$ git add -A ## rename, move, delete file update해준다. staging area에서

다음에

git commit하자

- 만약에 이름을 바꿨는데 취소하고 싶으면

$ git mv 2.txt level2.txt

## reset을 할 필요가 없다.

3.3. 파일을 다른 directory로 옮기는 방법

$ git mv level2.txt level3

다음에

git commit

- bash command로!!

$ mv level2.txt ..

다음에 update

git add -A

git commit

- 만약에 파일 탐색기에서 했으면

git add level1.txt

git add -u

git commit

## 왜냐하면 system level operation file을 -A로 다 넣고 싶지는 않기 때문이다.

3.4. removing file

- tracking files 🡪 git rm doomed.txt

$ git commit -m "deleting new file" ## commit까지 해야 한다.

- remove한 거를 취소하고 싶으면

$ git reset HEAD hister.txt

$ git checkout -- hister.txt

- untracked files 🡪 rm doomed.txt

git add -A 하면 stage되고 finalize하려면

git commit

- 폴더

rm -rf level1

git add -A # stage된다.

git commit

4. git history

$ git help log

$ git log

$ git log --abbrev-commit

$ git log --oneline --graph --decorate

- log 번호로 찾기(앞의 7개 번호만)

$ git log a5363c...44d44b0

- 시간으로 찾기

$ git log --since='3 days ago'

- 1개 file 만

$ git log -- hister.txt

- rename history

$ git log --follow -- level1/level2/level2.txt

- git show ~~commitID~~

4.2. Git alias

$ git log --all --graph --decorate --oneline

이거를 하나의 간단한 command로 바꾸자!!

$ git config --global alias.hist "log --all --graph --decorate --oneline"

- 만약에 alias를 바꾸고 싶으면

$ np ~/.gitconfig

5. Ignoring unwanted files and folders

만약에 file이 accidentally add하지 않도록 하기 위해서는

.gitignore file이 필요하다.



안에다가 ignore하고 싶은 filename을 집어넣으면 된다.

git add .ignore

git commit

- folder 전체를 할 수도 있다.

- Comparing

- conquer branching and merging

- GitHub introduction

- Tools Setup(Text Editor, Compar/Merge Tool)