

이화이언 기술설계팀

2 차 수습과제

2 차

[생활코딩- 지옥에서 온 Git]

37 기 수습운영진 김윤서

Git = 버전 관리 시스템 (Version Control System)

- 필요성

Report.xls -> report_final.xls -> ... 이름 다 다르게 해야하는 불편함

버전 관리 시스템 (ex Git) -> 이름 같게 하면서 버전 따로 관리해줌

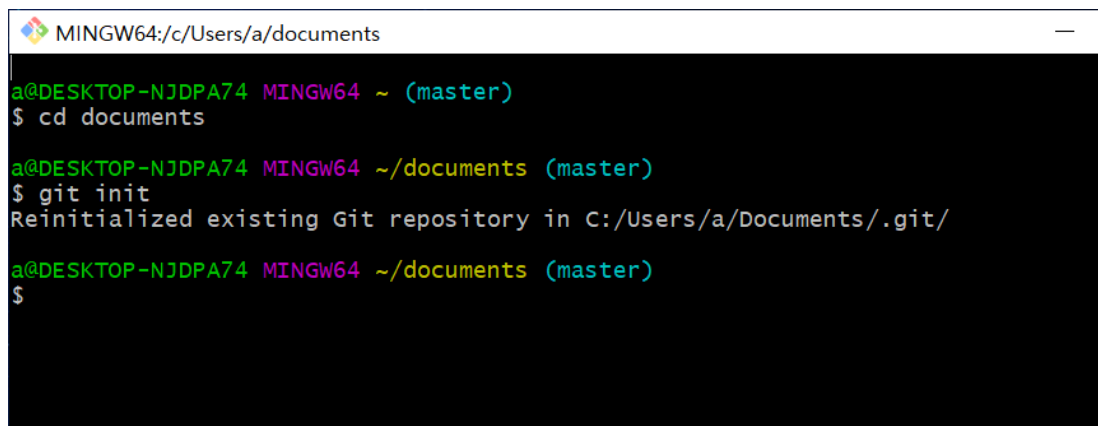
Back up/ Recovery/ Collaboration 등등을 버전 관리 시스템이 담당.

종류 : CVS, SVN, GIT – 공통점? " 변경 사항을 관리한다! "

Tip : CodeOnWeb 이라는 사이트 이용 -여러 프로그램을 깔지 않아도 돼서 편리.

Cd documents //documents 폴더로 들어가라

mkdir gitfth // gitfth 라는 디렉토리 만들어라

A screenshot of a terminal window with a black background and white text. The window title is 'MINGW64:/c/Users/a/documents'. The prompt is 'a@DESKTOP-NJDPA74 MINGW64 ~ (master)'. The user enters '\$ cd documents', and the prompt changes to 'a@DESKTOP-NJDPA74 MINGW64 ~/documents (master)'. Then, the user enters '\$ git init', and the terminal shows 'Reinitialized existing Git repository in C:/Users/a/Documents/.git/'. Finally, the user enters '\$' and the prompt returns to 'a@DESKTOP-NJDPA74 MINGW64 ~/documents (master)'.

```
MINGW64:/c/Users/a/documents
a@DESKTOP-NJDPA74 MINGW64 ~ (master)
$ cd documents

a@DESKTOP-NJDPA74 MINGW64 ~/documents (master)
$ git init
Reinitialized existing Git repository in C:/Users/a/Documents/.git/

a@DESKTOP-NJDPA74 MINGW64 ~/documents (master)
$
```

Cd gitfth // gitfth 로 들어가라 그래야 ls -al 했을 때 불필요한 정보없이 내가 쓴 파일만 보임

- 처음 시작 시(이어서 켤 때) 할 일

```
$ cd documents
$ cd gitfth

$ git init

$ ls-al
//이렇게 하면 내가 작업한 파일들 보임
```

6강. Git 명령

```
a@DESKTOP-NJDP74 MINGW64 ~/documents/gitfth
$ git init
Initialized empty Git repository in C:/Users/a/Documents/gitfth/.git/

a@DESKTOP-NJDP74 MINGW64 ~/documents/gitfth (master)
$ ls -a
bash: ls-a: command not found

a@DESKTOP-NJDP74 MINGW64 ~/documents/gitfth (master)
$ ls -al
total 8
drwxr-xr-x 1 a 197121 0  4월 12 02:07 ./
drwxr-xr-x 1 a 197121 0  4월 12 02:05 ../
drwxr-xr-x 1 a 197121 0  4월 12 02:07 .git/

a@DESKTOP-NJDP74 MINGW64 ~/documents/gitfth (master)
$
```

\$ git init 명령어 - "현재 디렉토리()에 작업을 시작하겠다 선언"

- .git 에 정보를 저장하겠다.
- ls -al 명령어 쳐보면 .git 뜸

7강. 파일을 생성해서 관리해보자.

vim f1.txt

입력 -> i 입력 -> <insert> 모드로 변경됨 (입력 모드 종료하려면 Esc)

i 모드에서 내용 입력하고, :wq 저장+파일 나가기

파일 다시 보는 방법은 Cat f1.txt -> 입력한 내용 표시됨

```
a@DESKTOP-NJDP74 MINGW64 ~/documents/gitfth (master)
$ vim f1.txt
bash: vim.f1.txt: command not found

a@DESKTOP-NJDP74 MINGW64 ~/documents/gitfth (master)
$ vim f1.txt

a@DESKTOP-NJDP74 MINGW64 ~/documents/gitfth (master)
$ cat f1.txt
iiiiisnjk
i

source : 1
wq
:wq
i
:wq

a@DESKTOP-NJDP74 MINGW64 ~/documents/gitfth (master)
$ |
```

버전 관리 알아보자.

```
Git add f1.txt
```

New file 로 인식함

새로운 파일은 add 명령을 이용해야 함

이유는? 프로젝트의 핵심적인 파일이 있고, 임시적인 파일이 있는데, 이는 버전관리 하면 x -> 이들을 배제하고 핵심만 포함하기 위해서 명확히 해줘야 함

8강. 지금까지 작업한 내용을 버전으로 저장해보자

코딩하다가 완결되지 않고 잠시 임시저장 해둔 상태 => 버전이 아님

버전은 완성된 상태 를 말함.

지금 버전을 확인하는 방법 `git status`

일단 내 이름 설정해야됨 (초기 단 한 번)

```
git config --global user.name yoonseo
```

```
git config --global user.email dotsi@naver.com
```

파일 열기 `git commit`

I (insert 모드) - 1 입력 - :wq 로 닫으면 버전 생성 완료

```
a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$ git config --global user.name yoonseo

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$ git config --global user.email dotsi@naver.com

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$ git commit
[master (root-commit) d92bc2f] 1 :wq :wq :wq
1 file changed, 15 insertions(+)
create mode 100644 f1.txt

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$
```

-완료!

버전이 잘 만들어졌는가 확인? `Git log`

```
a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$ git log
commit d92bc2f3a5a2b54b60efb7495c913538f01a4dde (HEAD -> master)
Author: yoonseo <dotsi@naver.com>
Date:   Sun Apr 12 02:33:57 2020 +0900

    1
    :wq
    :wq
    :wq

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$
```

- 버전 정보 (생성자 아이디, 이메일, 날짜 내용 등등 표시됨)

-2번째 버전 생성해보자 (버전 바꾸기 = 내용 바꿔주기)-

`Vim f1.txt`

파일 열리면

1을 2로 바꾸어주고 (그냥 내용 변경)

Esc - `:wq` 로 빠져나옴

이후 `git status` // (치면 modified: f1.txt 라는 경고메시지 뜨는데, 버전 add 안해줬기 때문임)

버전 add 해주기 - `git add f1.txt`

9강. f2.txt 생성해보기

```
ls -al //기록 보기
cp f1.txt f2.txt //copy
ls -al // 다시 하면 f2.txt 생성된 걸 볼 수 있음
git status //로 파일 생성된 걸 확인 (add 해줘야 함)
git add f2.txt
git commit
git log
vim f2.txt // i - 내용 바꾸어주기 - Esc - :wq
cat f2.txt // 바뀐 내용 확인
```

- Add 하는 이유?

Commit 타이밍 놓쳤을 때, 거대한 파일을 한 번에 변경해야 할 때..

Commit 되지 않은 파일 확인 가능

Add를 한 파일만이 commit 되기 때문에, 선택적으로 파일을 commit할 수 있다

이미 작업한 내용 중에서 커밋할 내용만 선택해서 커밋할 수 있다 = 혁신, git 의 장점

Add 함으로써 커밋 대기 상태에 들어가는 것 = stage area

Stage(커밋 대기하는 파일들이 가는 곳) - **repository** (커밋된 파일들의 저장소)

10강. 변경 사항 확인하기.

Git log

1. 차이점 알 수 있고 과거의 어느 곳을 확인 가능
2. 과거로 돌아갈 수 있다

Git log -p

```
--- /dev/null  
+++ b/f3.txt
```

이거에 대한 거 너무 어려워서 패스함...

```
--- 그 이전을 나타냄  
+++ 현재를 나타냄
```

```
commit d2ff38b4cd49fb530057c2db90f7603509277cb3  
Author: yoonseo <dotso@naver.com>  
Date:   Sun Apr 12 02:44:47 2020 +0900  
  
2
```

커밋 메시지 = 주소값: 그 과거로 돌아가는 매우 중요한내용

Git diff d2ff38b4cd49fb530057c2db90f7603509277cb3

// 이 주소값과 이전 텍스트의 차이점을 보여줌!

Git diff // 커밋하기 전에 그 이전과 현재의 차이점을 보여주는 것

11강. 과거로 돌아가기 =커밋을 취소하는 명령 (어려운 내용, 여기선 교양 수준)

reset VS revert

```
git reset d2ff38b4cd49fb530057c2db90f7603509277cb3 --hard
```

//협업 시, 공유한 이후에는 절대로 reset 하면 안 됨. 내 컴퓨터에 있을 때만!

하드는 조금 위험, soft는 나중에 배우겠음...

12강. 어떤 명령어가 많이 사용될까? -스스로 공부하는 법

Git commit -a //우리가 수정하거나 삭제한 파일은 자동으로 add하고 바로 commit할 수 있도록

틈틈이 매뉴얼을 보는 연습을 하자!

13강.

- 지금까지 배운 내용 - 버전 생성, 버전 수정하고 과거로 돌아가기 등

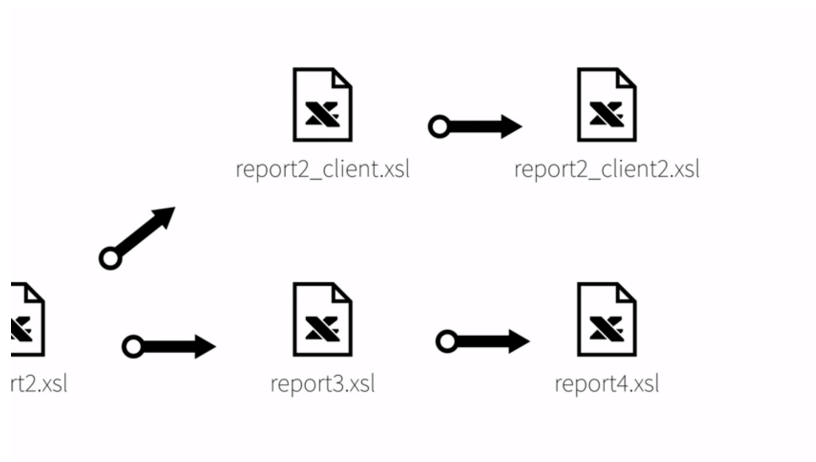
버전 관리 시스템이 가져온 혁신을

백업!!! 중요 - 이 버전관리 시스템을 백업 시스템(ex 구글 드라이브 등)에 업로드하세요

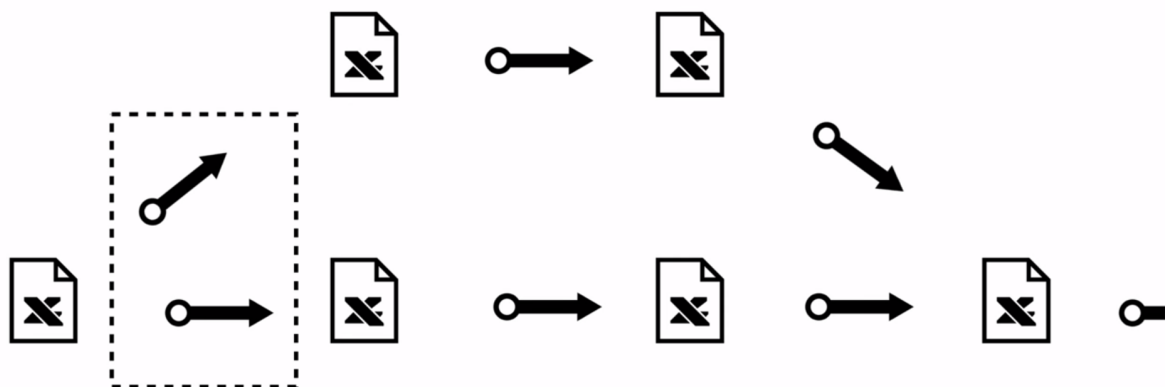
14~19강. 원리 소개 강의 – 넘어감, 나중에 필요하면 듣는 걸로..

20강. Branch 소개

Branch 나무의 가지



- 이런 경우 복잡해지기 때문에 브랜치가 필요.



작업이 분기되는 시점 = “브랜치를 만든다”

21강. 브랜치 만들기

git commit -a 하면 자동으로 add 됨

git commit -m "1" // 1추가

git commit -am "2" //2추가하고 add (?) 맞나

```
a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$ git branch
* master

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$ git branch exp

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$ git branch
exp
* master

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (master)
$ git checkout exp
Switched to branch 'exp'

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$
```

브랜치는 내가 만든 프로그램의 속성을 그대로 보관함

f1 이라는 파일이 어떤 브랜치에 속해있느냐(master / exp)에 따라 내용이 완전히 달라진다-

```
a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$ vim f2.txt

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$ git add f2.txt
warning: LF will be replaced by CRLF in f2.txt.
The file will have its original line endings in your working directory

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$ git commit -m "
> "
> "
> "
[exp 405df21] 2
1 file changed, 1 insertion(+), 3 deletions(-)

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$ git log
commit 405df21c8fddc06c3d8cd382a620c3e7717f21ad (HEAD -> exp)
Author: yoonseo <dotsi@naver.com>
Date: Sun Apr 12 05:06:49 2020 +0900

commit 51511bcf4ca947aa175069c9bfb95e3c4e5e1103
Author: yoonseo <dotsi@naver.com>
Date: Sun Apr 12 05:03:54 2020 +0900

commit 944497fbae8263428cd4967d569e7b8b9e52140c (master)
Author: yoonseo <dotsi@naver.com>
Date: Sun Apr 12 04:25:51 2020 +0900

commit cf01a6f131a50ea5c9229ed8c330b254697af424
Author: yoonseo <dotsi@naver.com>
Date: Sun Apr 12 04:24:05 2020 +0900

commit d79320968525f0101527c43d5e965f07d3f1112a
Author: yoonseo <dotsi@naver.com>
Date: Sun Apr 12 04:10:04 2020 +0900

10101010101010101010
```

exp 에 (branch)

"2" 추가

exp 와 완전히 다른 f2.txt

Branch 는 저장소를 쓰지 않고, 같은 파일의 버전을 관리하는 편리한 시스템

Git checkout master – master 브랜치를 헤드로 정함

Git checkout exp - exp 를 헤드로

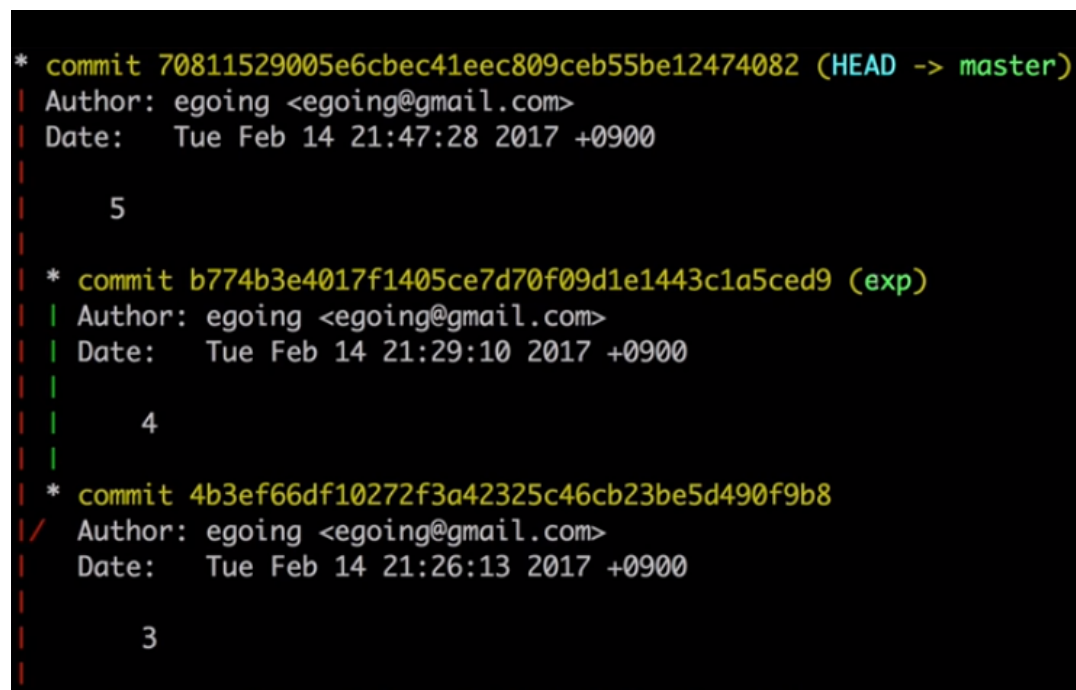
22 강. 브랜치 정보확인 (비교)

```
$ git log --branches --decorate
```

// 각 브랜치의 정보 확인 가능, 각 브랜치의 가장 최신 정보, 헤드 브랜치 확인 가능

```
$ git log --branches --decorate --graph
```

// 그래프 사용- 각 브랜치가 다른 커밋을 사용할 때 아주 유용.



```
* commit 70811529005e6cbec41eec809ceb55be12474082 (HEAD -> master)
| Author: egoing <egoing@gmail.com>
| Date: Tue Feb 14 21:47:28 2017 +0900
|
| 5
|
| * commit b774b3e4017f1405ce7d70f09d1e1443c1a5ced9 (exp)
| | Author: egoing <egoing@gmail.com>
| | Date: Tue Feb 14 21:29:10 2017 +0900
| |
| | 4
| |
| | * commit 4b3ef66df10272f3a42325c46cb23be5d490f9b8
| | / Author: egoing <egoing@gmail.com>
| | Date: Tue Feb 14 21:26:13 2017 +0900
| |
| | 3
|
```

Git log master..exp // master에만 있고 exp에는 없는 것을 보여줌

Git log exp..master // exp에만 있는 거 보여줌!

Git diff // 각각의 브랜치의 현재 상태를 비교

23 강. 브랜치 병합하기

Git log -branches -graph -decorate -

- Exp 를 master 로 가져올거면, master 를 체크아웃하고, 그 상태에서 git merge exp ,

병합 완료 후, (exp -> master 로 병합 완료하면)

```
$ git branch -d exp // exp branch 지우기
```

** 병합 부분 조금 더 보충하기

25 강. Stash

Branch 가 다 끝나지 않았을 때,

내가 작업한 내용을 어딘가에 숨겨두자

```
$ git checkout -b exp // exp라는 브랜치를 만들고 checkout 까지 하는 코드
```

필요한 상황 : exp 브랜치 수정하던 중 master 브랜치로 체크아웃 해야할 때, save 안하면 다른 브랜치랑 구분 x

```
a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$ vim f1.txt

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$ git checkout master
error: Your local changes to the following files would be overwritten by checkout:
    f1.txt
Please commit your changes or stash them before you switch branches.
Aborting

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$ git status
On branch exp
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   f1.txt

no changes added to commit (use "git add" and/or "git commit -a")

a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$ git checkout exp
Already on 'exp'
M       f1.txt
```

- exp 수정하다가 master 로 체크아웃하면, exp 에서 수정하던 내용이 master 에도 영향을 끼치는 일 발생

```
$ git stash save (OR, $ git stash 만 써도 ok)
```

```
a@DESKTOP-NJDPA74 MINGW64 ~/documents/gitfth (exp)
$ git stash save
Saved working directory and index state WIP on exp: 8614492 1
```

```
$ git reset --hard HEAD
```

// 초기화.

```
$ git stash apply // 복원, 적용이 됨, 가장 최신 stash 가 적용된다
```

```
Git stash drop //
```

33강. 원격 저장소 소개

Repository

내 컴퓨터가 아닌 다른 저장소를 원격 저장소라고 함

1. 버전을 백업한다
2. 협업을 한다

프로젝트가 커지는 과정에서 굉장히 중요

34강. 원격 저장소 생성

자기가 커밋하는 저장소 = **지역 저장소**

지역저장소와 동기화되는 저장소 = **원격 저장소** : 다른 컴퓨터에서 진행됨

- Remote 라는 원격저장소에 local이라는 지역저장소를 연결해보자

```
git init local // local 이라는 저장소가 생김
```

```
Git remote add
```

```
Pwd // 현재 디렉토리 주소가 나옴 -복사
```

```
Git remote add origin /home/git/git/remote // origin이 경로의 별명이 됨
```

```
Git remote -v
```

(Git remote remove origin //하면 지워짐)

```
Git push
```

// simple 방식을 쓰는 게 좋다!

```
Git config --global push.default simple // simple로 바꾼다
```

```
Git push origin master // origin의 master로 연결하겠다
```

-- local이 remote로 푸시되어서 동기화시킨 방법을 알아보았다

**** 원격저장소 만들 때 , git init --bare 라는 옵션**

**** 지역저장소의 내용을 원격저장소로 업로드 할 때 = push 할 때 , push라는 명령어 사용한다!**

35강. github 소개 - 원격저장소를 제공, 오픈소스,

github.com/git/git -- 원격저장소 볼 수 있음

Fork 버튼 누르면, 그 소스코드가 복제되어 내 것이 됨. 내가 마음대로 수정할 수 있음

`git clone github.com/git/git(원격저장소 홈페이지주소) gitsrc` // init 대신 clone 복제를 사용하자

`git log --reverse`

`git checkout +commit ID` //커밋아이디로

정리) 어딘가의 원격저장소의 내용을 로컬저장소로 가져오고 싶다면?

git clone +주소 +이 저장소를 저장하고 싶은 디렉토리

ex) `git clone github.com/git/git gitsrc`

36강. 내 컴퓨터의 작업 내용을 원격저장소에 올리는 방법?

일단 파일 생성해보자

```
a@DESKTOP-NJDPA74 MINGW64 ~ (master)
$ mkdir gitfth

a@DESKTOP-NJDPA74 MINGW64 ~ (master)
$ cd git
bash: cd: git: No such file or directory

a@DESKTOP-NJDPA74 MINGW64 ~ (master)
$ cd gitfth

a@DESKTOP-NJDPA74 MINGW64 ~/gitfth (master)
$ git init .
Initialized empty Git repository in C:/Users/a/gitfth/.git/

a@DESKTOP-NJDPA74 MINGW64 ~/gitfth (master)
$ vim f1.txt

a@DESKTOP-NJDPA74 MINGW64 ~/gitfth (master)
$ git add f1.txt
warning: LF will be replaced by CRLF in f1.txt.
The file will have its original line endings in your working directory

a@DESKTOP-NJDPA74 MINGW64 ~/gitfth (master)
$ git commit -m '1'
[master (root-commit) 73e2e54] 1
1 file changed, 1 insertion(+)
create mode 100644 f1.txt

a@DESKTOP-NJDPA74 MINGW64 ~/gitfth (master)
$ |
```

git commit -m '1' // 1이라는 커밋 생성?

git remote add origin <https://~~> //

git remote -v //