3. git명령어1

▼ 나의 디렉토리를 깃 로컬저장소로 만들기 및 초기화

```
ikaro@chocosoo MINGW64 /c
$ git init gitsave
Initialized empty Git repository in C:/gitsave/.git/

또는

ikaro@chocosoo MINGW64 /c/gitsave
$ git init
Initialized empty Git repository in C:/gitsave/.git/

-- 설치 시 생성되는 브랜치명을 main으로 설정해주었기에 대표 브랜치가 m
*** 깃허브의 원격저장소의 브랜치명이 main으로 생성되므로 여기에 맞춰서
*** 깃저장소가 되면 해당 디렉토리내에 [.git]이라는 디렉토리가 생성된다
ikaro@chocosoo MINGW64 /c/gitsave (main)
$ ls -la
...
drwxr-xr-x 1 ikaro 197609  0 10월 22 17:09 .git/
...
```

▼ git 환경설정하기 : 버전을 변경한 사람의 정보를 같이 전송해야하므로 버전을 수정한 사용자의 정보 설정을 해야만 깃을 사용할 수 있다.

```
ikaro@chocosoo MINGW64 /
$ git config --global user.name "설정할이름-영어로"

ikaro@chocosoo MINGW64 /
$ git config --global user.email "되도록 github연결된 이메일로"
```

[user]
name = YoungMi

email =
ikarosala@gmail.com

만약 —global옵션이 없이 설정하면 설정한 git저장소 내에서만 해당 사용자 정보를 사용하도록 설정할 수 있다.

```
ikaro@chocosoo MINGW64 /c/gitsave (main)
$ git config user.name "Youngmi"

ikaro@chocosoo MINGW64 /c/gitsave (main)
$ git config user.email "ikarosala@gmail.com"
```

```
ikaro@chocosoo MINGW64 /c/gitsave/.git (GIT_DIR!)
$ cat config

[core]
repositoryformatversion = 0
filemode = false
bare = false
logallrefupdates = true
symlinks = false
ignorecase = true

[user]
name = Youngmi
email =
ikarosala@gmail.com
```

▼ git 설정 삭제

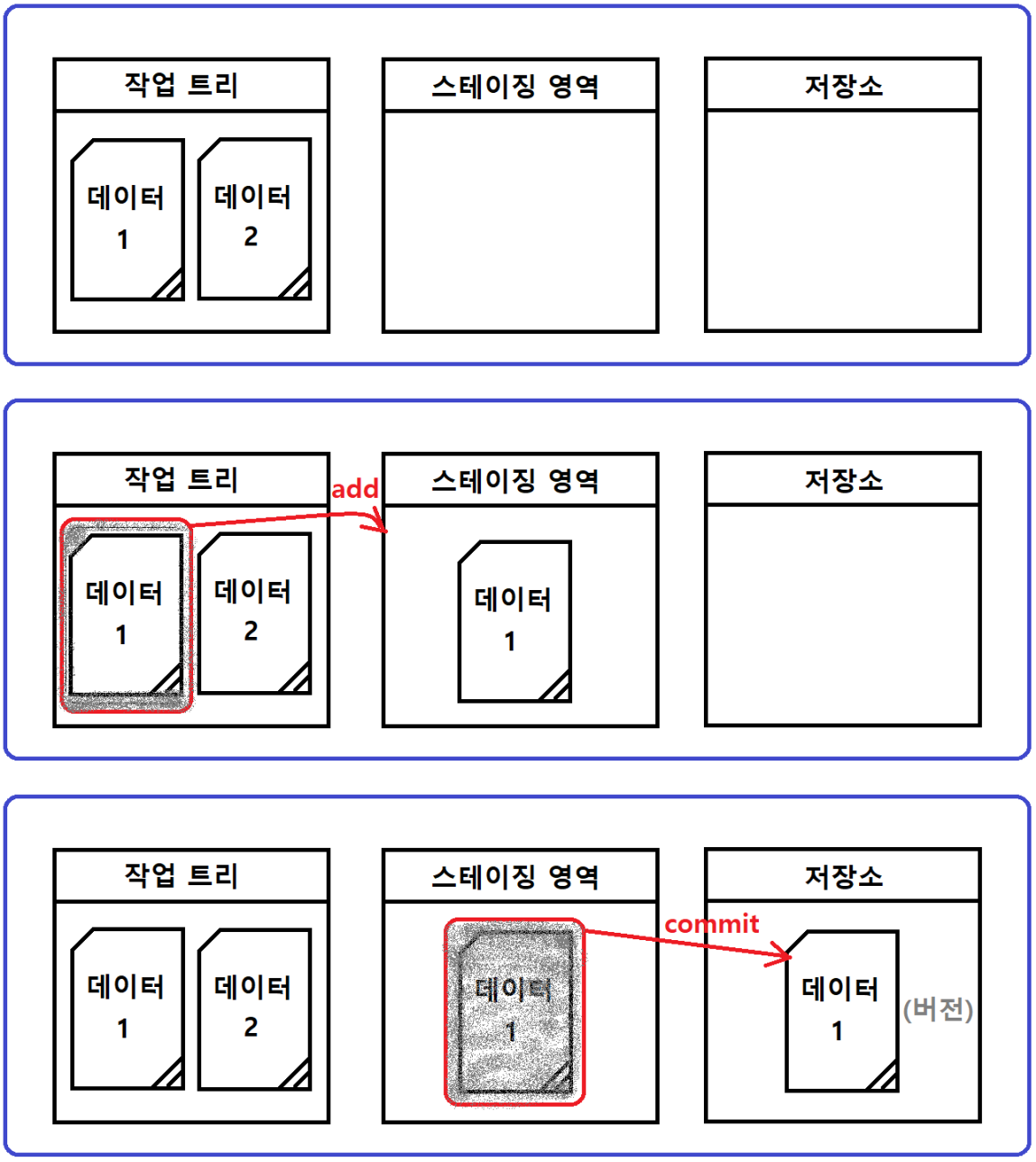
```
ikaro@chocosoo MINGW64 /
$ git config --global --unset-all user.name

ikaro@chocosoo MINGW64 /
$ git config --global --unset-all user.email
```

▼ git 버전 만들기

- 버전이란 : 문서를 수정하고 저장할때마다(변경할때마다) 생기는 데이터
- 작업트리(Working Tree) : 로컬깃저장소 디렉토리(폴더)를 작업트리라고 함. 직접 확인가능
- 스테이지(Stage) : 작업트리에서 생성된 데이터를 버전으로 만들기 전 대기하는 영역. 스테이징 영역이라고도 함. 보여지지 않음
- 저장소(Repository) : 스테이징 영역에 존재하던 데이터를 버전을 만들어 보관하는 곳. (실제 원격저장소(깃허브)에 올라갈 수 있는 데이터) . 보여지지 않음

▼ git에서 데이터의 버전을 생성하는 단계



** commit작업이 완료되어 저장소에 저장된 데이터를 버전이라고 함

▼ tracked 와 untracked 파일의 차이

> 💡 tracked : 커밋이력이 있는 파일들로 추적 대상이 되는 파일
>
> untracked : 커밋이력이 없는 파일들로 추적 대상이 되지 않는 파일

▼ 작업순서

1. 데이터를 작업트리에 넣기

- 깃 저장소로 만든 폴더 안에 작업한 파일을 옮겨두거나 깃 저장소로 만든 폴더 안에서 데이터 작업을 하면 해당 데이터는 `작업트리 영역에 존재`하게 된다.

2. **작업트리의 데이터를 스테이징 영역으로 이동하기(깃 저장소까지 작업디렉토리 이동 후 명령어 사용할 것) :** `git add`

- 명령어: git add 파일명.확장자명 또는 파일패턴 또는 폴더명

```
예시:
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git add test/
warning: in the working copy of 'test/1.txt', LF will b
warning: in the working copy of 'test/2.txt', LF will b
```

****LF will be replaced by CRLF the next time Git touches it ：Linux나 MacOS 계열의 운영체제의 줄바꿈(LF, CR) 기능을 윈도우 계열의 운영체제의 줄바꿈(CRLF)기능으로 대체한다는 문구**

****참고사이트 : https://dabo-dev.tistory.com/13**

3. **스테이징 영역의 데이터를 저장소까지 이동하기 :** `git commit`

- 명령어 : git commit -m “버전명”

**** -m 옵션 : message옵션임. 반드시 버전명을 기입해주어야 처리 가능함.**

```
예시:
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git commit -m "test v1.0"
[main (root-commit) 4da0ace] test v1.0
 2 files changed, 5 insertions(+)
 create mode 100644 test/1.txt
 create mode 100644 test/2.txt
```

▼ git저장소내의 파일 상태 확인하기 : `git status`

```
--명령어 : git status
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main     /*메인 브랜치라는 의미*/

No commits yet   /*아직 커밋한 내용이 없음을 의미*/

Untracked files:  /*commit을 위해 staging할 데이터가 아래와 같이
  (use "git add <file>..." to include in what will be comm
        3.hwp
        test/

nothing added to commit but untracked files present (use "
/*커밋할 데이터가 현재 없음을 의미. 커밋을 위해서는 git add 명령(스테
```

```
--git add를 하고 난 후의 status확인

ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main

No commits yet

Changes to be committed: /*스테이징 영역에서 커밋으로 변경하기 위한
  (use "git rm --cached <file>..." to unstage) /*스테이징에서
        new file:   test/1.txt
        new file:   test/2.txt

Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp
```

```
--git commit을 하고 난 후의 status확인
```

```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Untracked files:/*스테이징 역역으로 보내기위한 파일이 작업트리에 존재
  (use "git add <file>..." to include in what will be comm
        3.hwp

nothing added to commit but untracked files present (use "
/*커밋할 데이터가 현재 없음을 의미. 커밋을 위해서는 git add 명령(스테



------------------------------------------------------------
--git commit을 하고 난 후의 파일을 수정한 경우는 스테이징 작업과 커밋
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git commit -am "test v1.1"   /*-a옵션 : add의미 : 반드시 한번
warning: in the working copy of 'test/1.txt', LF will be r
[main cd2a467] test v1.1
 1 file changed, 1 insertion(+)
```

▼ **커밋한 이력 확인하기 :** `git log`

```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git log
commit 4da0ace6eb0fbda596fe1395d20bce140dfee089 (HEAD -> ma
/*HEAD : 가장 최신 커밋버전을 의미  (최신버전의 커밋이면서 main브랜치
/*4da0ace6eb0fbda596fe1395d20bce140dfee089 : 커밋해시값*/
Author: YoungMi <ikarosala@gmail.com>  /*버전을 만든 사용자의
Date:   Thu Oct 24 11:27:00 2024 +0900   /*버전을 만든 시간 정

    test v1.0  /*버전명*/

------------------------------------------------------------
--git commit을 하고 난 후의 로그내용

ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git log
commit cd2a467e9fc9159c1296f888a978a879bef2402b (HEAD -> ma
```

```
Author: YoungMi <ikarosala@gmail.com>
Date:   Thu Oct 24 11:56:58 2024 +0900

    test v1.1

commit 4da0ace6eb0fbda596fe1395d20bce140dfee089
Author: YoungMi <ikarosala@gmail.com>
Date:   Thu Oct 24 11:27:00 2024 +0900

    test v1.0
```

▼ 버전관리된 파일의 변경 상세 내용 확인하기 : git diff

```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git diff
warning: in the working copy of 'test/1.txt', LF will be r
diff --git a/test/1.txt b/test/1.txt
index 7e251a3..96f2231 100644
--- a/test/1.txt
+++ b/test/1.txt
@@ -1,3 +1,4 @@
 1번 파일입니다.  /*변경 이력이 없이 기존과 동일함*/
-1번파일 수정합니다.   /* -가 붙은 내용은 기존파일에서 제거되었다는 의
+1번 파일 내용추가합니다. /* +가 붙은 내용은 기존파일 내용에서 추가되었
+수정완료합니다./* +가 붙은 내용은 기존파일 내용에서 추가되었다는 의미

------------------------------------------------------------
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git diff
warning: in the working copy of 'test/1.txt', LF will be r
diff --git a/test/1.txt b/test/1.txt
index 96f2231..dabf2c2 100644
--- a/test/1.txt
+++ b/test/1.txt
@@ -1,4 +1,5 @@
 1번 파일입니다.
 1번 파일 내용추가합니다.
```

```
-수정완료합니다.
+1번 추가1
+1번 추가2
```

▼ 스테이징 영역의 데이터 제거하기 (반드시 커밋전이어야 함)

```
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test/1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committe
  (use "git restore <file>..." to discard changes in worki
        modified:   test/1.txt

Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp

------------------------------------------------------------
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git restore --staged test/1.txt    /*스테이징영역의 test/1.

ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committe
  (use "git restore <file>..." to discard changes in worki
        modified:   test/1.txt

Untracked files:
  (use "git add <file>..." to include in what will be comm
```

```
        3.hwp

no changes added to commit (use "git add" and/or "git comm

------------------------------------------------------------
ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git restore test/1.txt  /*작업트리의 수정내역 되돌리기*/


ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git restore test/1.txt

ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be comm
        3.hwp

nothing added to commit but untracked files present (use "

ikaro@chocosoo MINGW64 /c/ikarosalaLocal (main)
$ cat test/1.txt
1번 파일입니다.
1번파일 수정합니다.
```

git 작업예시

git 작업 시 에러케이스