1/11 git

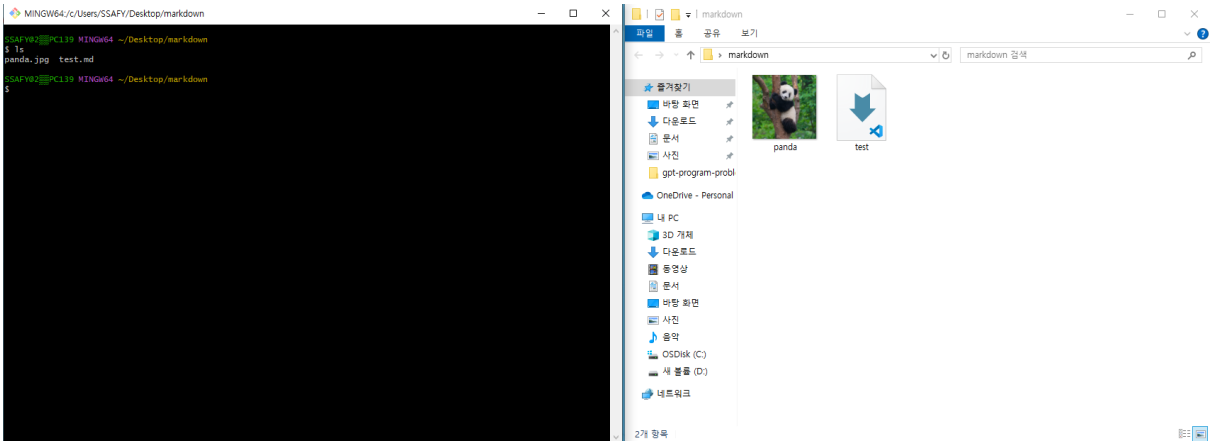
CLI GUI

CLI command line interface > 명령어를 통해

GUI graphic user interface > 그래픽을 통해

GUI는 CLI에 비해 사용하기 쉽지만 단계가 많고 성능을 상대적으로 많이 소모

수많은 서버/개발 시스템이 CLI를 통한 조작 환경을 제공함



두 화면은 같은 곳임

윈도우 바탕 화면의 절대 경로 예시

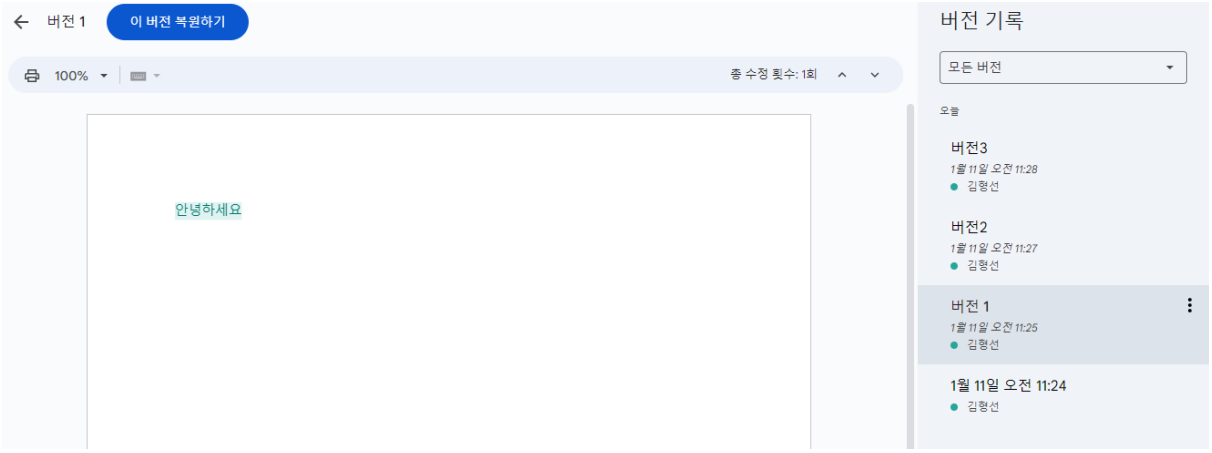
C:/Users/ssafy/Desktop

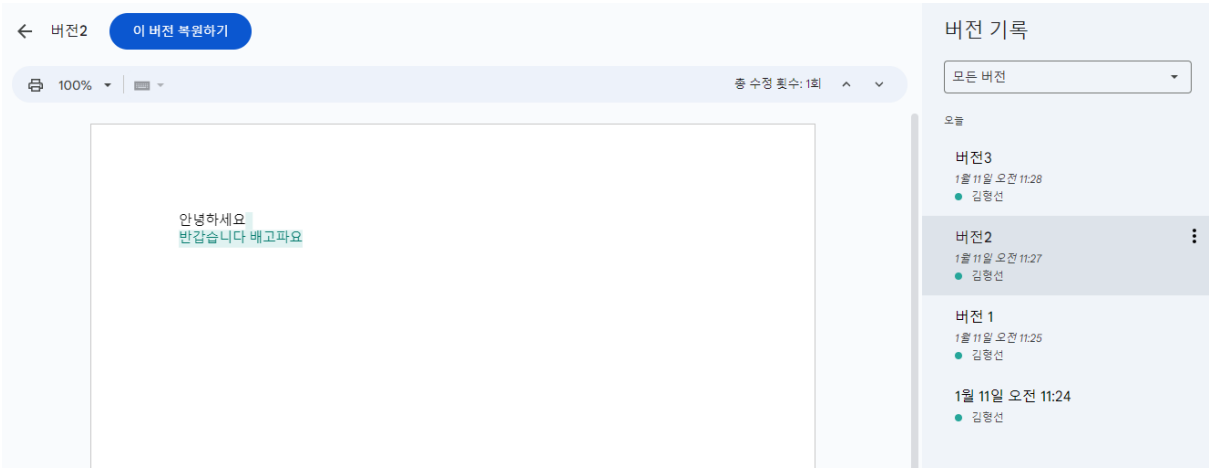
Git

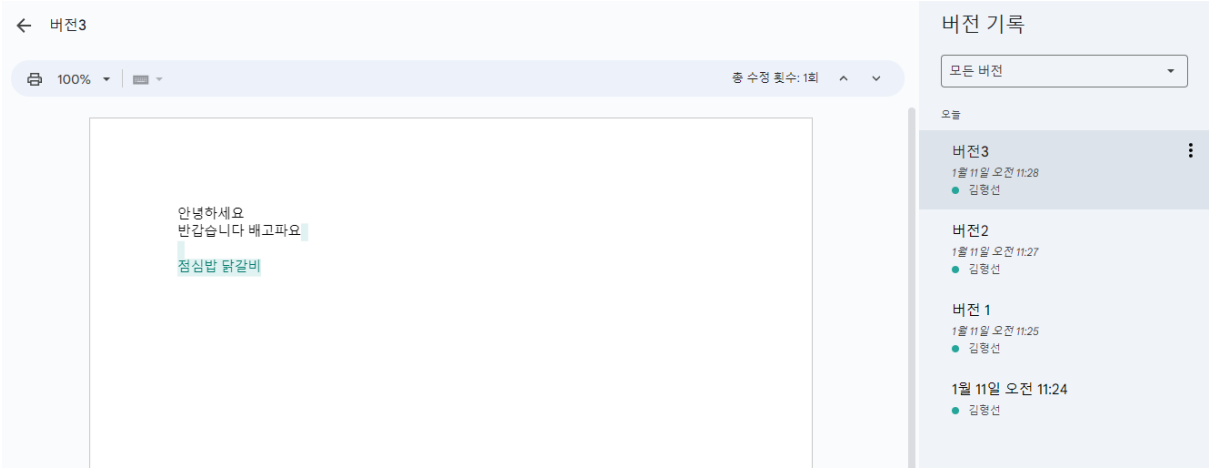
분산 버전 관리 시스템

버전 관리 : **변화**를 기록하고 추적하는 것

Google docs에서 버전 3개를 만들고 상황을 봐보겠다







버전 1,2,3을 보면 변화(변경 사항)만 기록을 한다(전체를 기록해버리면 너무 비효율적임)

추가된 각 줄들을 코드로 생각하면 된다

각 버전은 변경 사항에 대한 것들만 기록되고, 최종본만 전체에 대한 것이 모두 기록되어 있다고 생각하면 된다

git은 분산 버전 관리 시스템이다(중앙 집중식x)

git > 코드의 변경 이력을 기록하고 협업을 원활하게 하는 도구

git 저장소 안에 또 다른 git저장소는 존재할 수 없다

git init을 진행한 상위 디렉토리의 하위 디렉토리 그 어디에서도 git init을 진행하면 안 된다

git의 영역

1. working directory
- 실제 작업 중인 파일들이 위치하는 영역
2. Staging Area

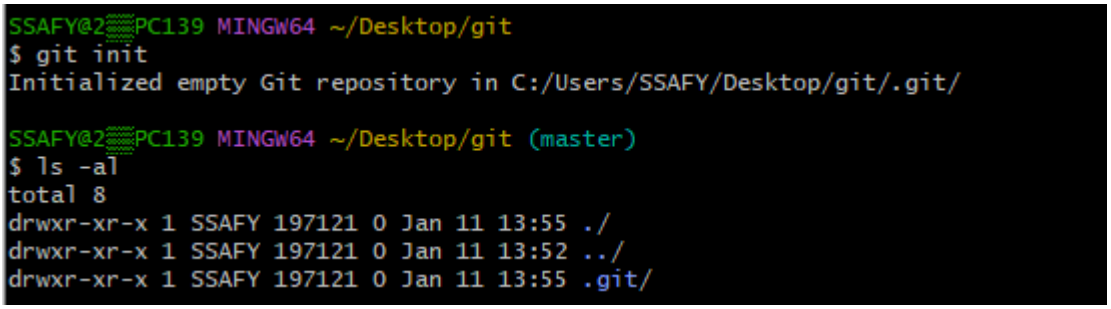
- Working Directory에서 변경된 파일 중 다음 버전을 포함할 파일들을 선택적으로 추가하거나 제외할 수 있는 중간 준비 영역(a.txt와 c.txt에서 각각 변경이 일어났을 때, a.txt만 stage로 add할 수 있음)
- 코드를 추가했다면 그런 변경 사항들 git status 명령어로 확인 가능

3. Repository

- 버전 이력과 파일들이 영구적으로 저장되는 영역
- 모든 버전과 변경 이력이 기록됨
- commit > 변경된 파일들을 저장하는 행위, snapshot이라고도 함

Git 명령어(git init, git add, git commit)

git init > 초기화, 로컬 저장소 설정, git 버전 관리를 시작할 디렉토리에서 진행



~/Desktop/git < 이 경로는 git의 영역이라는 표시로 (master)가 뜬다

commit하기 전에 **git add** 명령어로 변경사항이 있는 파일을 staging area에 추가

git commit

staging area에 있는 파일들을 저장소에 기록 > 해당 시점의 버전을 생성하고 변경 이력을 남기는 것

---

vscode를 키고 sample.txt파일을 만들겠다

git status 명령어로 변경 사항을 감지할 수 있다

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        sample.txt

nothing added to commit but untracked files present (use "git add" to track)
```

commit을 하기 위해서는 staging area에 올려야 하며 그러려면 git add를 해야한다

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git add sample.txt

SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   sample.txt
```

git add로 staging area에 올렸다

이제 commit을 해보겠다.

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git commit -m "first commit"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'SSAFY@2ÃþPC139.(none)')
```

commit을 하려는데 내가 누구인지를 확인해야 한다는 것이다.

git config로 configuration(설정) 관련 정보를 넘겨야 한다.

```
git config --global user.email "yumisgood@gmail.com"
git config --global user.name "JAMIEKIMHS"

--global 옵션(폴더 단위가 아닌 컴퓨터 전체)을 사용하면 한 번만 설정해주면 다음부터는 안 해도 된다
이 설정은 나중에 바꿀 수 있는 부분이다
```

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git config --global user.email "yumisgood@gmail.com"

SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git config --global user.name "JAMIEKIMHS"

SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git config --global --list
user.email=yumisgood@gmail.com
user.name=JAMIEKIMHS
```

```
git config --global --list
위 명령어로 설정 관련 정보를 확인할 수 있다
```

그리고 다시 git status로 어떤 상태인지를 보겠다

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   sample.txt
```

여전히 staging area에서 대기 상태인 것을 확인할 수 있다

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git commit -m "first commit"
[master (root-commit) 2a09aca] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 sample.txt
```

1 file changed라는 메시지가 보인다.

변경된 파일 안에 line들이 변한 것이 없다는 것도 뒤의 insertions와 deletions를 통해 확인할 수 있다

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master
nothing to commit, working tree clean
```

이제 다시 상태를 확인해 보면 더 이상 스냅샷을 찍을 것이 없는 것을 확인할 수 있다

commit내용을 확인하려면 **git log** 명령어 사용하면 된다

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git log
commit 2a09aca0b7c2a5a499caa672c9857de754ed04f5 (HEAD -> master)
Author: JAMIEKIMHS <yumisgood@gmail.com>
Date:   Thu Jan 11 14:14:50 2024 +0900

    first commit
```

이제 두 번째 버전을 만들어보겠다. 변경 사항을 하나 만들겠다.

sample.txt에 text 내용을 추가하고 git status로 확인해보겠다.

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master
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:   sample.txt

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

처음에 파일을 생성했을 때는 빨간 메시지가 untracked라고 나왔는데, 지금은 modified라고 나와있다.

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git add sample.txt

SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   sample.txt
```

git add 후에 다시 상태를 확인해보면 staging area로 잘 올라간 것을 확인할 수 있다

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git commit -m "second commit"
[master 3720ec8] second commit
 1 file changed, 1 insertion(+)

SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master
nothing to commit, working tree clean
```

commit이 잘 되었고 working tree도 clean한 상태인 것을 확인할 수 있다

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git log
commit 3720ec8d89fd84550caea0b254af6fa3cae39f5c (HEAD -> master)
Author: JAMIEKIMHS <yumisgood@gmail.com>
Date:   Thu Jan 11 14:21:00 2024 +0900

    second commit

commit 2a09aca0b7c2a5a499caa672c9857de754ed04f5
Author: JAMIEKIMHS <yumisgood@gmail.com>
Date:   Thu Jan 11 14:14:50 2024 +0900

    first commit
```

git log 명령어를 통해 commit이 잘 올라간 것까지 확인했다.

이제 추가적인 변경 사항을 더 만들어보겠다

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        sample2.txt
        sample3.txt

nothing added to commit but untracked files present (use "git add" to track)
```

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git add .

SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   sample2.txt
        new file:   sample3.txt
```

```
git add .
위 명령어로 한 번에 staging area로 올린다
```

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git commit -m "third commit"
[master b7bcfdf] third commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 sample2.txt
 create mode 100644 sample3.txt

SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git status
On branch master
nothing to commit, working tree clean
```

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git log
commit b7bcfdf5e207c591121564a7e5216adc0b27b13b (HEAD -> master)
Author: JAMIEKIMHS <yumisgood@gmail.com>
Date:   Thu Jan 11 14:43:01 2024 +0900

    third commit

commit 3720ec8d89fd84550caea0b254af6fa3cae39f5c
Author: JAMIEKIMHS <yumisgood@gmail.com>
Date:   Thu Jan 11 14:21:00 2024 +0900

    second commit

commit 2a09aca0b7c2a5a499caa672c9857de754ed04f5
Author: JAMIEKIMHS <yumisgood@gmail.com>
Date:   Thu Jan 11 14:14:50 2024 +0900

    first commit
```

---

git init을 하는 순간 .git디렉토리가 생성이 된다

.git 디렉토리를 삭제하면 git init을 잘못했을 때 해결할 수 있다

rm -r .git

git 저장소 안에 git 저장소가 있을 경우 가장 바깥 쪽의 git저장소가 안쪽의 git 저장소의 변경 사항을 추적할 수 없기 때문

```
git log --oneline : commit 목록 한 줄로 보기
```

```
SSAFY@2□□PC139 MINGW64 ~/Desktop/git (master)
$ git log --oneline
b7bcfdf (HEAD -> master) third commit
3720ec8 second commit
2a09aca first commit
```

로컬

- local(지역)
- 반대는 global(전역), online(remote)
- 현재 사용자가 직접 접속하고 있는 기기 또는 시스템
- 개인 컴퓨터, 노트북, 태블릿 등 사용자가 직접 조작하는 환경

로컬 저장소 하나 당 원격 저장소 하나만 대응이 된다