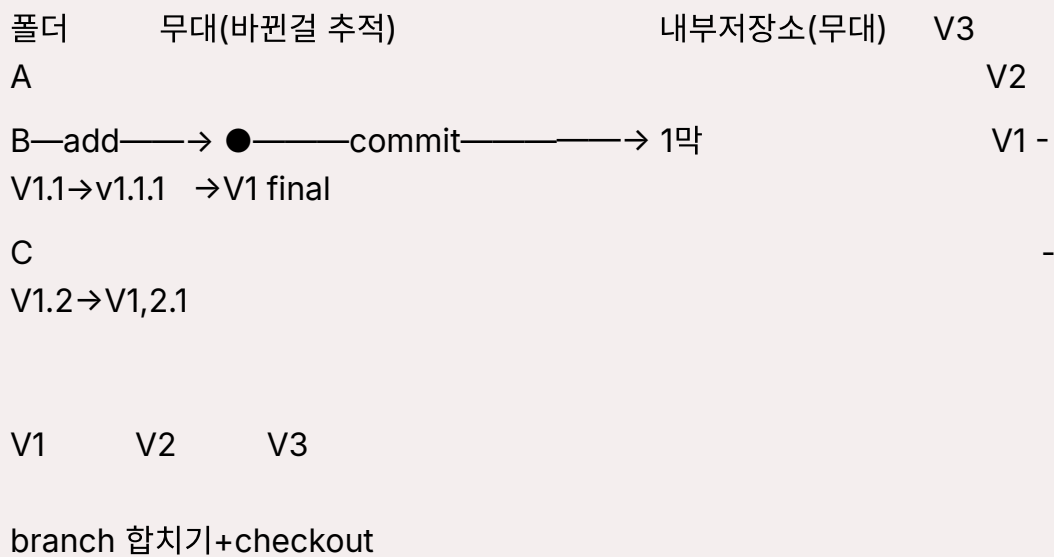


git hub

☐ git hub

☐ docker



무대는 동선을 추적하는 역할을 함.

merge(버전 병합

merge할때 승인 받아야함.

pull 버전 branch down

ChatGPT의 말:

ChatGPT

Git과 브랜치(branch)에 대해 처음 배우면 혼란스러울 수 있지만, 천천히 이해하면 코드 관리를 효과적으로 할 수 있는 강력한 도구입니다. Git과 브랜치에 대해 쉽게 설명할게요.

1. Git이란?

- **Git**은 분산 버전 관리 시스템(DVCS)입니다. 즉, 여러 사람이 협력해서 코드를 관리할 수 있게 해주는 도구입니다.
- **버전 관리**: 코드의 변화를 기록하고, 특정 시점의 코드 상태로 돌아갈 수 있게 해줍니다. 마치 문서의 '이전 버전으로 돌아가기' 기능처럼, Git을 사용하면 코드의 과거 상태를 쉽게 확인하고 복원할 수 있습니다.
- **저장소(Repository)**: Git에서 프로젝트 파일들이 저장되고, 그 파일의 모든 변경 기록이 담기는 공간입니다. 이 저장소는 로컬(내 컴퓨터)이나 원격(클라우드, 서버)에 있을 수 있습니다.

2. Branch(브랜치)란?

- ***브랜치(Branch)***는 독립적인 작업 공간을 의미합니다. 기본적으로 Git의 모든 저장소에는 `main` 혹은 `master` 라는 기본 브랜치가 있습니다.
- **브랜치의 역할**: 여러 사람이 동시에 하나의 프로젝트를 작업할 때 서로의 작업이 충돌하지 않도록, 각자 독립적인 공간(브랜치)에서 작업한 뒤, 모두가 동의하는 시점에서 그 결과를 합칩니다(머지).
- **예시**: 게임을 만들 때 "캐릭터 추가" 작업을 하는 사람과 "배경 디자인"을 변경하는 사람이 있을 수 있습니다. 이 두 사람이 동시에 하나의 코드 파일에서 작업하면 충돌이 일어날 수 있습니다. 그래서 각자 다른 브랜치에서 작업한 뒤, 작업이 끝나면 이를 병합(merge)하여 최종 결과물을 만듭니다.

3. Git의 기본 명령어

- **git init**: 새로운 Git 저장소(repository)를 초기화합니다. 프로젝트 폴더에서 처음 사용할 때 씁니다.
- **git add <파일이름>**: 수정된 파일을 Git의 스테이징(staging) 영역에 추가합니다. 즉, 커밋을 위한 준비 단계입니다.

- **git commit -m "메시지"**: 스테이징 영역에 있는 변경 사항을 기록합니다. "메시지" 부분에는 변경 내용을 요약한 메시지를 적습니다.
- **git status**: 현재 저장소의 상태(어떤 파일이 수정되었고, 스테이징 되었는지 등)를 보여줍니다.
- **git log**: 커밋 히스토리를 확인할 수 있습니다.

4. 브랜치 관련 명령어

- **git branch**: 현재 어떤 브랜치들이 있는지 확인합니다.
- **git branch <브랜치이름>**: 새로운 브랜치를 만듭니다. `main` 브랜치에서 새로운 브랜치를 만들면 `main`의 모든 내용이 복사된 새 작업 공간이 생성됩니다.
- **git checkout <브랜치이름>**: 특정 브랜치로 이동합니다. 이 명령어를 사용하면 그 브랜치에서 작업을 할 수 있습니다.
- **git merge <브랜치이름>**: 현재 작업 중인 브랜치에 다른 브랜치의 내용을 합칩니다(병합).

5. 브랜치를 쓰는 이유

- **독립적인 작업**: 브랜치를 사용하면 메인 프로젝트에 영향을 주지 않고 실험적인 기능을 개발하거나 새로운 시도를 할 수 있습니다.
- **충돌 방지**: 여러 사람이 동시에 작업할 때 충돌을 방지하기 위해 각자 브랜치를 만들어 작업하고 나중에 하나로 병합합니다.
- **안전한 코드 유지**: 메인 브랜치(`main`)는 항상 안정적인 코드만 유지할 수 있습니다. 새로운 기능은 다른 브랜치에서 작업 후, 문제가 없을 때만 메인 브랜치에 병합할 수 있습니다.

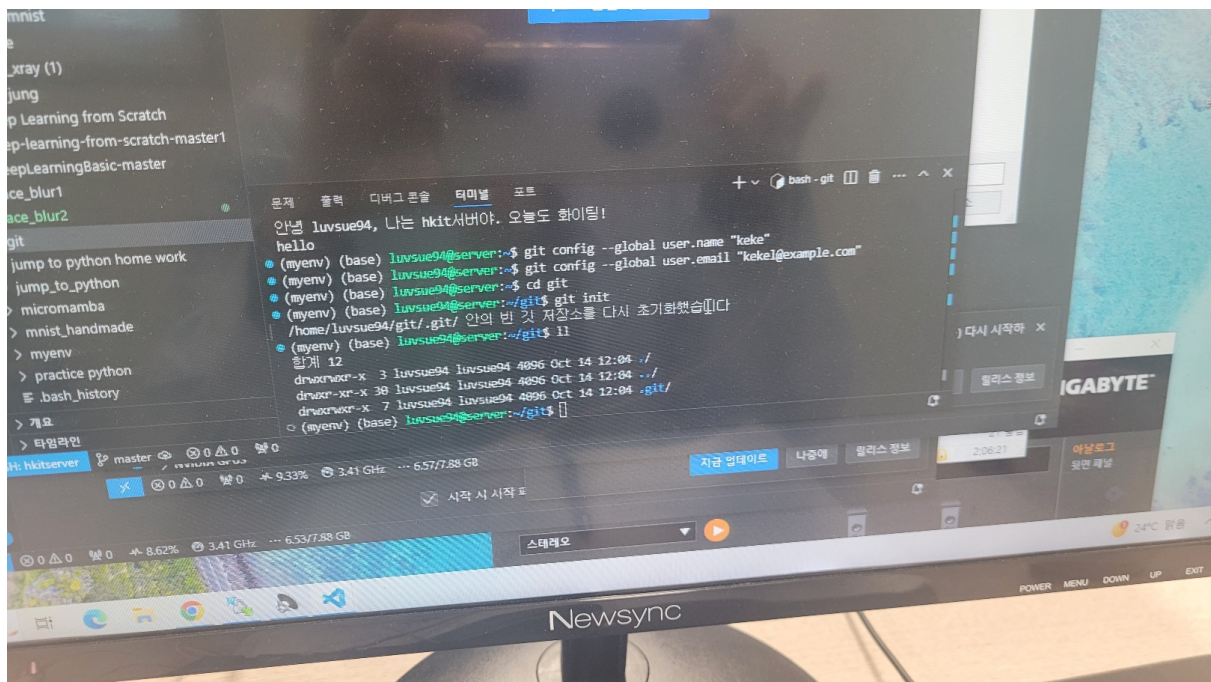
6. 브랜치 병합(Merge)

- 다른 브랜치에서 완료된 작업을 메인 브랜치에 합치고 싶을 때 `git merge`를 사용합니다. 병합 시 충돌(conflict)이 발생할 수 있는데, 이 경우 Git은 충돌을 해결하도록 도와줍니다.
- **충돌이란?** 두 명의 개발자가 같은 파일을 동시에 수정했을 때 발생합니다. 이 경우, 어느 부분이 최종 수정본인지 선택하는 과정을 거칩니다.

7. 실제 예시

- 예를 들어, 웹사이트를 개발 중인데 새 기능을 추가하고 싶다면 `git branch new-feature` 로 새로운 브랜치를 만들고 그 브랜치에서 작업을 합니다. 기능이 완성되면 `main` 브랜치로 돌아와(`git checkout main`), `git merge new-feature` 로 작업을 병합하면 됩니다.

git init-워킹 디렉토리를 나타내주는 파일(이니셜라이즈 초기화 1번만 하면 됨)
 drwxrwxr-x 7 luvsue94 luvsue94 4096 Oct 14 12:04 **.git/**



```

안녕 luvsue94, 나는 hkit서버야. 오늘도 화이팅!
hello
(myenv) (base) luvsue94@server:~$ git config --global user.name "keke"
(myenv) (base) luvsue94@server:~$ git config --global user.email "keke1@example.com"
(myenv) (base) luvsue94@server:~$ cd git
(myenv) (base) luvsue94@server:~/git$ git init
/home/luvsue94/git/.git/ 안의 빈 깃 저장소를 다시 초기화했습니다
(myenv) (base) luvsue94@server:~/git$ ll
합계 12
drwxrwxr-x  3 luvsue94 luvsue94 4096 Oct 14 12:04 ./
  
```

```
drwxr-xr-x 30 luvsue94 luvsue94 4096 Oct 14 12:04 ../
drwxrwxr-x  7 luvsue94 luvsue94 4096 Oct 14 12:04 .git/
(myenv) (base) luvsue94@server:~/git$ git status
현재 브랜치 master
```

아직 커밋이 없습니다

추적하지 않는 파일:

(커밋할 사항에 포함하려면 "git add <파일>..."을 사용하십시오)
test.py

커밋할 사항을 추가하지 않았지만 추적하지 않는 파일이 있습니다 (추적하려면 "git add"를 사용하십시오)

```
(myenv) (base) luvsue94@server:~/git$ git add test.py
(myenv) (base) luvsue94@server:~/git$ git status
현재 브랜치 master
```

아직 커밋이 없습니다

커밋할 변경 사항:

(스테이지 해제하려면 "git rm --cached <파일>..."을 사용하십시오)
새 파일: test.py

```
(myenv) (base) luvsue94@server:~/git$ git commit -m
error: switch `m'은(는) 값을 필요로 합니다
(myenv) (base) luvsue94@server:~/git$ git comiit-m "메령"
git: 'comiit-m'은(는) 깃 명령이 아닙니다. 'git --help'를 참고하십시오
(myenv) (base) luvsue94@server:~/git$ git commit -m "메롱"
[master (최상위-커밋) d217dfe] 메롱
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.py
(myenv) (base) luvsue94@server:~/git$ git push
fatal: 푸시 대상을 설정하지 않았습니다.
명령행에서 URL을 지정하거나 다음을 사용해 리모트 저장소를 설정하십시오
```

```
git remote add <이름> <URL>
```

그리고 이 리모트 이름을 사용해 푸시하십시오

```
git push <name>
```

```
(myenv) (base) luvsue94@server:~/git$ git branch
```

```
* master
```

```
(myenv) (base) luvsue94@server:~/git$ git branch 대구
```

```
(myenv) (base) luvsue94@server:~/git$ git branch
```

```
* master
```

```
대구
```

```
(myenv) (base) luvsue94@server:~/git$ git check out -b 대구
```

```
git: 'check'은(는) 깃 명령이 아닙니다. 'git --help'를 참고하십시오.
```

가장 비슷한 명령은

```
checkout
```

```
(myenv) (base) luvsue94@server:~/git$ git check out 대구
```

```
git: 'check'은(는) 깃 명령이 아닙니다. 'git --help'를 참고하십시오.
```

가장 비슷한 명령은

```
checkout
```

```
(myenv) (base) luvsue94@server:~/git$ git checkout 대구
```

```
'대구' 브랜치로 전환합니다
```

```
(myenv) (base) luvsue94@server:~/git$ git branch
```

```
master
```

```
* 대구
```

```
(myenv) (base) luvsue94@server:~/git$ git status
```

```
현재 브랜치 대구
```

```
커밋할 사항 없음, 작업 폴더 깨끗함
```

```
(myenv) (base) luvsue94@server:~/git$ git status
```

```
현재 브랜치 대구
```

```
추적하지 않는 파일:
```

```
(커밋할 사항에 포함하려면 "git add <파일>..."을 사용하십시오)
```

```
test2.py
```

커밋할 사항을 추가하지 않았지만 추적하지 않는 파일이 있습니다 (추적하려면 "git add"를 사용하십시오)

```
(myenv) (base) luvsue94@server:~/git$ git add test2.py
```

```
(myenv) (base) luvsue94@server:~/git$ git status
```

```
현재 브랜치 대구
```

커밋할 변경 사항:

(스테이지에서 제외하려면 "git restore --staged <file>..."을 사용하여
새 파일: test2.py

```
(myenv) (base) luvsue94@server:~/git$ git commit
```

커밋을 중지합니다. 커밋 메시지가 비어 있습니다.

```
(myenv) (base) luvsue94@server:~/git$ git commit -m "동구"
```

[대구 b02aaed] 동구

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 test2.py

```
(myenv) (base) luvsue94@server:~/git$ git checkout master
```

'master' 브랜치로 전환합니다

```
(myenv) (base) luvsue94@server:~/git$ git merge 대구
```

업데이트 중 d217dfe..b02aaed

Fast-forward

test2.py | 0

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 test2.py

```
(myenv) (base) luvsue94@server:~/git$ git branch
```

* master

대구

```
(myenv) (base) luvsue94@server:~/git$ git status
```

현재 브랜치 master

커밋할 사항 없음, 작업 폴더 깨끗함

```
(myenv) (base) luvsue94@server:~/git$ git status
```

현재 브랜치 master

커밋하도록 정하지 않은 변경 사항:

(무엇을 커밋할지 바꾸려면 "git add <파일>..."을 사용하십시오)

(작업 디렉토리의 변경을 무시하려면 "git restore <file>..."을 사용하

수정함: test.py

수정함: test2.py

커밋할 변경 사항을 추가하지 않았습니다 ("git add" 및/또는 "git commit
사용하십시오)

```
(myenv) (base) luvsue94@server:~/git$ git push
```

fatal: 푸시 대상을 설정하지 않았습니다.

명령행에서 URL을 지정하거나 다음을 사용해 리모트 저장소를 설정하십시오

```
git remote add <이름> <URL>
```

그리고 이 리모트 이름을 사용해 푸시하십시오

```
git push <name>
```

```
(myenv) (base) luvsue94@server:~/git$ git remote add origin h
(myenv) (base) luvsue94@server:~/git$ git remote -v
origin https://github.com/user/repo.git (fetch)
origin https://github.com/user/repo.git (push)
(myenv) (base) luvsue94@server:~/git$ git remote rename 대구 L
fatal: 해당 리모트가 존재하지 않습니다: '대구'
(myenv) (base) luvsue94@server:~/git$ git remote rename <기존O
bash: 여기치 않은 토큰 '<' 근처에서 문법 오류
(myenv) (base) luvsue94@server:~/git$ git remote rename origi
(myenv) (base) luvsue94@server:~/git$ git remote remove origi
fatal: 해당 리모트가 존재하지 않습니다: 'origin'
(myenv) (base) luvsue94@server:~/git$ git remote rename origi
fatal: 해당 리모트가 존재하지 않습니다: 'origin'
(myenv) (base) luvsue94@server:~/git$ git status
현재 브랜치 master
```

커밋하도록 정하지 않은 변경 사항:

(무엇을 커밋할지 바꾸려면 "git add <파일>..."을 사용하십시오)

(작업 디렉토리의 변경을 무시하려면 "git restore <file>..."을 사용하

```
수정함:      test.py
수정함:      test2.py
```

커밋할 변경 사항을 추가하지 않았습니다 ("git add" 및/또는 "git commit"을 사용하십시오)

```
(myenv) (base) luvsue94@server:~/git$ git add test.py
(myenv) (base) luvsue94@server:~/git$ git add test2.py
(myenv) (base) luvsue94@server:~/git$ git status
```

현재 브랜치 master

커밋할 변경 사항:

(스테이지에서 제외하려면 "git restore --staged <file>..."을 사용하

```
수정함:      test.py
수정함:      test2.py
```



```
(myenv) (base) luvsue94@server:~/git$ git remote add origin h
(myenv) (base) luvsue94@server:~/git$ git remote -v
main-repo      https://github.com/user/repo.git (fetch)
main-repo      https://github.com/user/repo.git (push)
origin https://github.com/luvminn2/git_test.git (fetch)
origin https://github.com/luvminn2/git_test.git (push)
(myenv) (base) luvsue94@server:~/git$ git push origin master
오브젝트 나열하는 중: 5, 완료.
오브젝트 개수 세는 중: 100% (5/5), 완료.
Delta compression using up to 28 threads
오브젝트 압축하는 중: 100% (3/3), 완료.
오브젝트 쓰는 중: 100% (5/5), 411 바이트 | 411.00 KiB/s, 완료.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/luvminn2/git_test.git
 * [new branch]      master -> master
(myenv) (base) luvsue94@server:~/git$
```