Git 설치

1. 공식 홈페이지- https://git-scm.com/ 접속
2. Download for Windows 버튼 클릭
3. 파일 다운로드 후 설치 프로그램 실행
4. 여러 옵션을 선택하는데, 기본값으로 Next 선택 (여러 번 물어봄)
5. Git 실행을 위해 바탕화면에서 오른쪽 버튼을 누르고 Git Bash Here 선택
6. 설치 확인
    - git version
    - git version 2.39.0.windows.2 - 버전정보가 뜨면 정상설치완료!

Git 환경 설정

1. Your Name을 변경해 주세요

- git config --global user.name "Your Name"

2. you@your-email.com을 변경해 주세요

- git config --global user.email "you@your-email.com"

3. 한글 출력 오류 방지

- git config --global core.quotepath false

- 기본명령어 정리 #은 약어로 zsh 설치가 되어있을 때 사용가능

git init - 저장소 만들기

작업

1. sample 디렉토리 생성
2. red, orange 파일 추가
3. sample 디렉토리를 로컬 저장소로 설정

- mkdir, cd, touch, echo 명령어
- mkdir: 디렉토리 생성
- cd: 디렉토리로 이동
- touch: 빈 파일 생성
- echo "[글자]" >> [파일]: 파일에 글자 추가

실습 예제

```
hhjeo@hyunho MINGW64 ~
- $ mkdir sample

hhjeo@hyunho MINGW64 ~
- $ cd sample
```

```
hhjeo@hyunho MINGW64 ~/sample
- $ touch red orange

hhjeo@hyunho MINGW64 ~/sample
- $ echo "빨강" >> red

hhjeo@hyunho MINGW64 ~/sample
- $ echo "주황" >> orange

hhjeo@hyunho MINGW64 ~/sample
- $ git init
```

결과

```
Initialized empty Git repository in C:/Users/hhjeo/sample/.git/
```

sample 디렉토리에 Git 저장소 생성

- 디렉토리 하위에 .git 디렉토리 생성 - Git과 관련된 정보 저장 git bash가 ➜ sample에서 ➜ sample git: (main) ✗로 변경

---

git status - 현재 상태 확인

작업

1. 상태 확인

- git status # gst

gst

- alias로 oh-my-zsh을 설치하면 사용할 수 있는 별칭.
- git status대신 gst만 입력해도 동일하게 동작.

결과

```
On branch master
No commits yet
Untracked files:

  (use "git add <file>..." to include in what will be committed)

        git & github
        orange
        red
```

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

- 현재 브랜치(main)와 커밋 상태, 작업 중인 파일의 상태 확인
- untracked files(추적하지 않는 파일)이 존재하는 것을 확인

---

git add - 현재 상태 추적

작업

1. -A 옵션을 이용하여 전체 파일(orange, red)을 인덱스에 추가
2. 상태 확인

실습

```
- git add -A # gaa
- git status # gst
```

결과

```
On branch master

No commits yet

Changes to be committed:

  (use "git rm --cached <file>..." to unstage)

        new file:   git & github
        new file:   orange
        new file:   red
```

- untracked files에 있던 git&github, orange, red의 상태가 변경된 것을 확인

---

git commit - 현재 상태 저장

구조

- git commit [-m]

작업

- -m 옵션을 이용하여 첫 번째 이력에 대한 메시지 작성

실습

```
- git commit -m "v1 commit" # gc -m "v1 commit"
```

결과

```
[master (root-commit) 25b3885] v1 commit

 3 files changed, 57 insertions(+)

    create mode 100644 git & github
    create mode 100644 orange
    create mode 100644 red
```

- v1 commit 커밋 생성완료.

새 파일 추가

작업

1. yellow 파일을 만들기.
2. 상태 확인.

실습

```
touch yellow
echo "노랑" >> yellow
git status # gst
```

결과

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        yellow
no changes added to commit (use "git add" and/or "git commit -a")
```

새 파일 커밋

작업

1. 직전 커밋 이후 변경된 전체 파일(yellow)을 인덱스에 추가

2. 두 번째 이력 커밋

실습

```
git add -A # gaa
git commit -m "v2 commit" # gc -m "v2 commit"
```

결과

```
[master ed66e82] v2 commit
2 files changed, 85 insertions(+)
create mode 100644 yellow
```

다양한 변화

추가/수정/삭제를 이용한 세 번째 이력 만들기

작업

1. red 삭제
2. orange에 내용 추가
3. green 파일 추가
4. 상태 확인
5. 전체 파일 인덱스에 추가
6. 세 번째 이력 커밋

실습

```
rm red
echo "오렌지" >> orange
touch green
git status # gst
git add -A # gaa
git commit -m "v3 commit" # gc -m "v3 commit"
```

결과

```
On branch master
Changes not staged for commit:

  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
```

```
      modified:   orange
      deleted:    red

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

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

- orange를 수정했고 red는 삭제, green은 새로 만들어진 것을 확인

---

git log - 이력 확인

구조

git log [] [] [[--] ...] // git log는 다양한 옵션을 조합하여 원하는 형태의 로그를 출력할 수 있는 강력한 기능

작업

- 전체 로그 확인

실습

```
git log
```

결과

```
commit 2c9f173011b896f37ed0112d5e9a7b2d0b2d3c8a (HEAD -> master)
Author: Hyun Ho <hhjeong@gmail.com>
Date:   Mon Jan 2 16:47:28 2023 +0900
    v3 commit

commit ed66e82ab9dee5ffa10cd26a5da7c289e8ddf13b

Author: Hyun Ho <hhjeong@gmail.com>
Date:   Mon Jan 2 16:43:00 2023 +0900

    v2 commit

commit 25b3885f720f42d3d9452b30bbf0cb9dbb6c57fd
Author: Hyun Ho <hhjeong@gmail.com>
Date:   Mon Jan 2 16:30:08 2023 +0900

    v1 commit
```

- 전체 커밋 로그 확인

- 확인 후 Q로 나가기기

---

git reset - 이전 상태로 (이력 제거)

구조

git reset [] [--soft | --mixed [-N] | --hard | --merge | --keep]

- 특정 커밋까지 이력을 초기화.
- 바로 전, 또는 n번 전까지 작업했던 내용을 취소가능.
- git reset은 다양한 옵션이 존재하는데 여기선 --hard 옵션을 사용

작업

1. git log로 2번 커밋 ID 조회
2. 2번 커밋까지 이력 초기화

실습

```
git log
git reset {v2 커밋 아이디} --hard # 커밋 아이디 예) 27a00b7 (앞에 7자 정도 복사)
```

결과

```
HEAD is now at 27a00b7 v2 commit
```

- 2번 커밋까지 이력 초기화 -> 결론적으로 3번 이력 삭제 확인
- 지웠던 red가 되살아나고 orange 내용이 수정되고 green 파일이 사라진 것을 확인

---

git revert - 이전 상태로 (이력 유지)

구조

- git revert ...

- 특정 커밋을 취소하는 새로운 커밋 생성
- 3번 커밋을 취소하는 새로운 커밋을 생성하여 2번 커밋 상태로 돌아간 것 같지만 기존 이력을 유지하는 모습 확인.

작업

1. git reset 명령어로 3번 커밋이 지워졌다면 이전 실습을 통해 다시 커밋 추가
2. git log로 3번 커밋 ID 조회
3. 3번 커밋 취소

실습

```
git log
git revert {v3 커밋 아이디} # 커밋 아이디 예) 306b947 (앞에 7자 정도 복사)
git log
```

결과

```
Removing green
[master 2c9f173011b8] Revert "v3 commit"
3 files changed, 1 insertion(+), 1 deletion(-)
delete mode 100644 green
create mode 100644 red
```

- 지웠던 red가 되살아나고 orange 내용이 수정되고 green 파일이 사라진 것을 확인

---

git switch -c - 브랜치 생성

구조

- git switch (-c|-C)

작업

1. 실습한 4개의 커밋이 있는 저장소에서 -c 옵션으로 add-color 브랜치를 생성하면서 이동

실습

git checkout -b add-color # 예전 Git 버전 명령어

```
git switch -c add-color # gsw -c add-color
```

-c 옵션

- git switch의 -c 옵션은 브랜치 생성과 브랜치 이동을 한번에 수행
- 명령어 동일

```
git branch add-color
git swtich add-color
```

결과

```
Switched to a new branch 'add-color'
```

- 기존 main 브랜치의 작업 공간과 동일한 add-color 브랜치를 생성하면서 이동 <<<<<<< HEAD
- git bash가 ➜ sample git:(master)에서 ➜ sample git:(add-color)로 변경

---

add-color 브랜치에서 작업

작업

1. green, blue 파일 추가
2. 전체 변경사항을 인덱스에 추가
3. 커밋 작성

실습

```
touch green blue
echo "녹색" >> green
echo "파랑" >> blue
git add -A # gaa
git commit -m "add green, blue" # gc -m "add green, blue"
```

결과

```
[add-color d71a952] add green, blue
 3 files changed, 30 insertions(+), 1 deletion(-)
 create mode 100644 blue
 create mode 100644 green
```

---

git switch - 브랜치 변경

작업

master 브랜치로 이동

실습

git checkout main # 예전 Git 버전 명령어

```
git switch master # gsw master, gsw 한칸 띄고 <tab>을 눌러보세요
```

결과

```
Switched to branch 'master'
```

- git bash가 ➜ sample git:(add-color)에서 ➜ sample git:(master)로 변경
- add-color 브랜치에서 추가한 green, blue 파일이 사라지고 이전 상태로 돌아온 것을 확인

---

update-red 브랜치 추가

작업

1. update-red 브랜치 생성 후 이동
2. red 파일 내용 변경
3. 전체 변경사항을 인덱스에 추가
4. 커밋 작성

실습

git checkout -b update-red # 예전 Git 버전 명령어

```
git switch -c update-red # gsw -c update-red
echo "붉은색" > red
git add -A # gaa
git commit -m "update red" # gc -m "update red"
```

결과

```
Switched to a new branch 'update-red'
[update-red 044058e] update red
1 file changed, 1 insertion(+), 1 deletion(-)
```

현재 브랜치 상황

- master: red, orange, yellow
- add-color: red, orange, yellow, green, blue
- update-red: red(변경), orange, yellow

---

git merge - 브랜치 합치기

구조

git merge [...]

- add-color에서 작업한 내용을 master에 합치기.

작업

1. master 브랜치로 이동
2. add-color 브랜치의 수정사항을 master 브랜치로 머지
3. 전체 커밋 메시지 확인

실습

```
git switch master # gsw master
git merge add-color # gm add-color
git log
```

결과

```
Merge made by the 'recursive' strategy.
blue  | 1 +
green | 1 +
2 files changed, 2 insertions(+)
create mode 100644 blue
create mode 100644 green
```

> add-color에서 작업한 내용(파일 추가)이 master 브랜치로 머지됨
>
> > master 브랜치에 green, blue 파일이 추가된 것을 확인
> >
> > > add-color에서 작성한 커밋 로그가 master 브랜치에도 추가된 것 확인

---

conflict - 충돌 해결

- update-red 브랜치에서 수정한 red 파일을 main 브랜치에서도 수정하여 충돌 상황을 만들기

작업

1. master 브랜치에서 red 파일 수정
2. 전체 변경사항을 인덱스에 추가
3. 커밋 작성
4. update-red 브랜치를 master로 머지

실습

```
echo "뻘건색" > red
git add -A # gaa
git commit -m "update red color" # gc -m "update red color"
git merge update-red # gm update-red
```

결과

```
Auto-merging red
CONFLICT (content): Merge conflict in reds
Automatic merge failed; fix conflicts and then commit the reult.
```

> CONFLICT라는 메시지와 함께 실패(failed)
>
> > 충돌을 해결하고 커밋을 하거나, 머지 작업을 취소(git merge --abort)할 수 있다.

red 파일내용

<<<<<<< HEAD 뻘건색

- =====

붉은색

- >>>>>> update-red
- 충돌이 발생하면 양쪽 브랜치에서 동시에 변경된 사항을 표시
- <<<<<<<, ========, >>>>>>> 이 내용이 충돌이 발생한 지점을 의미
- 붉은색만 남기고 다른 줄을 삭제합니다. <<<<<<<, 뻘건색, ========, >>>>>>>를 모두 제거
- 수정 후 git add -A # gaa >> git commit # gc
- 머지 성공.