

# 오픈소스 SW 입문

## 3w git



2023-0504, 강의실(과학관213)

강사 : 박노현

<https://github.com/nparkcourage/2023-kau-0504>



# 한국항공대학교

## 목차

### 7교시

15:00

#### ◆ git 개요

- 소스 코드 버전 관리와 git
- git CLI
- 저장소 만들기, 환경 설정

### 8교시

16:00

#### ◆ git CLI 사용법

- 소스 관리(add, commit, 복구)
- branch

### 9교시

17:00

#### ◆ github

- 보안 설정과 ssh키 저장
- 저장소 만들기과 clone
- markdown 편집, push

## 첫째시간

7교시

15:00

### ◆ git 개요

- 소스 코드 버전 관리와 git
- git CLI
- 저장소 만들기, 환경 설정

8교시

16:00

### ◆ git CLI 사용법

- 소스 관리(add, commit, 복구)
- branch

9교시

17:00

### ◆ github

- 보안 설정과 ssh키 저장
- 저장소 만들기과 clone
- markdown 편집, push

## 소스 코드 버전 관리와 git

### 소스 코드 버전 관리

- VCS(Version Control System) : 버전 관리
- SCM(Source Code Management) : 소스 코드 관리

### 분산 버전 관리 시스템

- DVCS(Distributed Version Control System) : 저장소가 분산되어 서버의 저장소에 의존하지 않는 버전 관리 시스템

### git

- 분산 버전 관리 시스템
- 2005년 리누스 토발즈가 개발
- 라이선스 : GPL V2.0
- 개발 배경
  - 2002년부터 리눅스 버전 관리시스템으로 BitKeeper(BitMover사의 상용 SW, 2016년 오픈소스, 개발 중단) 사용
  - BitMover사가 리눅스 개발자들이 무료로 사용할 수 있도록 하였으나, 앤드루 트리젤이 비슷한 소스 풀러를 만들면서 리눅스 개발자들에 대한 무료 사용을 철회
  - 리누스 토발즈가 다른 오픈소스 버전관리 시스템을 사용하려 하였으나 당시 원하는 기능을 만족하는 SW가 없어서 직접 만들게 됨
  - 그 후 리눅스 버전 관리 시스템으로 사용되면 현재 가장 많이 사용되는 버전 관리 시스템이 됨

## 소스 코드 버전 관리와 git

### git의 목표

- 빠른 속도
- 단순한 구조
- 비선형적인 개발(수천 개의 동시 다발적인 브랜치)
- 완벽한 분산
- Linux 커널 같은 대형 프로젝트에도 유용할 것(속도나 데이터 크기 면에서)

## 소스 코드 버전 관리와 git

### git 공식 홈페이지

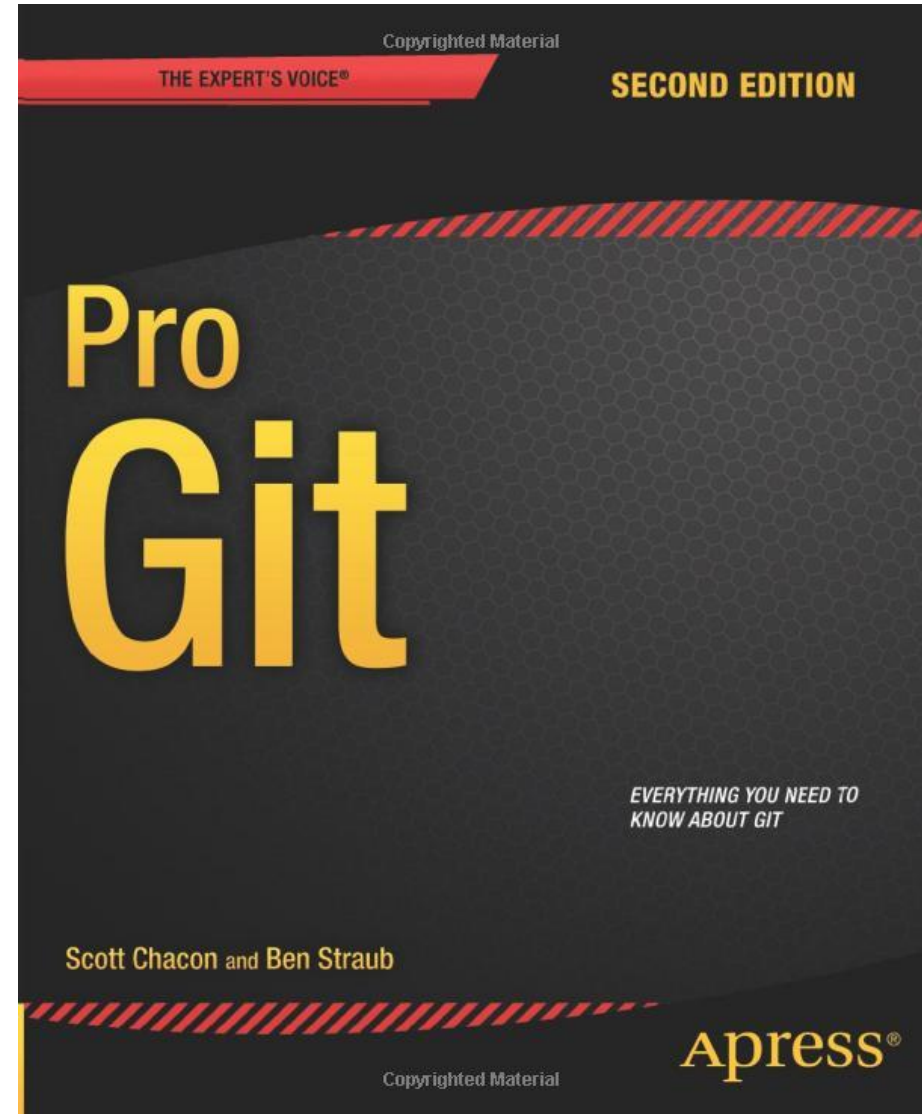
- <https://git-scm.com>

### 공식 문서 및 영상 자료

- <https://git-scm.com/doc>

### Pro Git

- <https://git-scm.com/book/en/v2>
- 한국어 버전
  - <https://git-scm.com/book/ko/v2>
- 무료 온라인 문서
- 무료 전자책 다운로드



## git CLI

### git을 사용하는 방법

- git CLI
- 다양한 GUI 지원 SW들 존재
- git의 모든 기능을 지원하는 SW는 git CLI뿐

### git 설치

- 리눅스에는 기본적으로 설치되어 있음
- 없으면 다음 명령으로 설치
- `sudo apt install git`

### 작업 디렉토리 만들기

- 임의의 디렉토리를 만들면 된다.  
(이번 강의에서는 이 문서의 디렉토리 구조를 따른다.)
- `mkdir ~/projects/w3/git-study`
- `cd ~/projects/w3/git-study` : 작업 디렉토리, 워킹 트리

### git 저장소 만들기

- `git init` : 작업 디렉토리를 git 저장소로 초기화
  - `.git` 디렉토리 생성
- `git branch` : 브랜치 보기, 기본 브랜치는 main
- 기본 브랜치가 master인 경우 버전 확인(2.28이전)
  - `git --version`

### git 업그레이드 하기

```
sudo add-apt-repository ppa:git-core/ppa -y
sudo apt update
sudo apt install git -y
```

### git 저장소 다시 만들기

- git 저장소 삭제
  - `.git` 디렉토리 삭제
- `git init` 실행
- 브랜치 확인(main)
  - `git branch`

## git 환경 설정

### git 환경 설정 파일

- /etc/gitconfig
  - 시스템의 모든 사용자와 모든 저장소에 적용되는 설정
  - git config --system <설정>
- ~/.gitconfig
- ~/.config/git/config
  - 특정 사용자(즉 현재 사용자)에게만 적용되는 설정
  - git config --global <설정>
- .git/config
  - 작업 디렉토리에 위치
  - 현재 작업 중인 프로젝트에만 적용
  - git config <설정>
  - git config --local <설정>
- 적용 순위
  - .git/config > ~/.gitconfig > /etc/gitconfig

### git 환경 설정

- git config --global user.name "N Park"
- git config --global user.email [npark.courage@gmail.com](mailto:npark.courage@gmail.com)

### git 환경 설정 보기

- git config --list
  - 종료 : q
- git config <key> : <key>에 대한 설정값 확인

### git 기본 편집기

- 확인
  - git config core.editor
- 설정
  - git config --global core.editor nano
- 디폴트(설정되지 않을 때)
  - EDITOR 환경 변수
  - 시스템의 기본 편집기 : select-editor



## 둘째시간

7교시

15:00

### ◆ git 개요

- 소스 코드 버전 관리와 git
- git CLI
- 저장소 만들기, 환경 설정

8교시

16:00

### ◆ git CLI 사용법

- 소스 관리(add, commit, 복구)
- branch

9교시

17:00

### ◆ github

- 보안 설정과 ssh키 저장
- 저장소 만들기과 clone
- markdown 편집, push

## 소스 관리

### 소스 상태

- 관리 대상 여부
  - tracked(추적대상)
    - 관리 대상
  - untracked(추적대상 아님)
    - 관리 대상 아님
- 확인
  - git status

### 관리 대상 상태

- committed : 버전관리에 안전하게 저장됨, snapshot
- staged : 커밋할 대상이라고 표시된 상태
- modified : 파일이 수정되었지만 staged 되지 않음
- tracked인 파일은 변경을 감지함
- untracked인 파일은 변경을 감지하지 않음
- untracked인 파일이 새로 생긴 것은 감지함

### 관리 대상으로 만들기

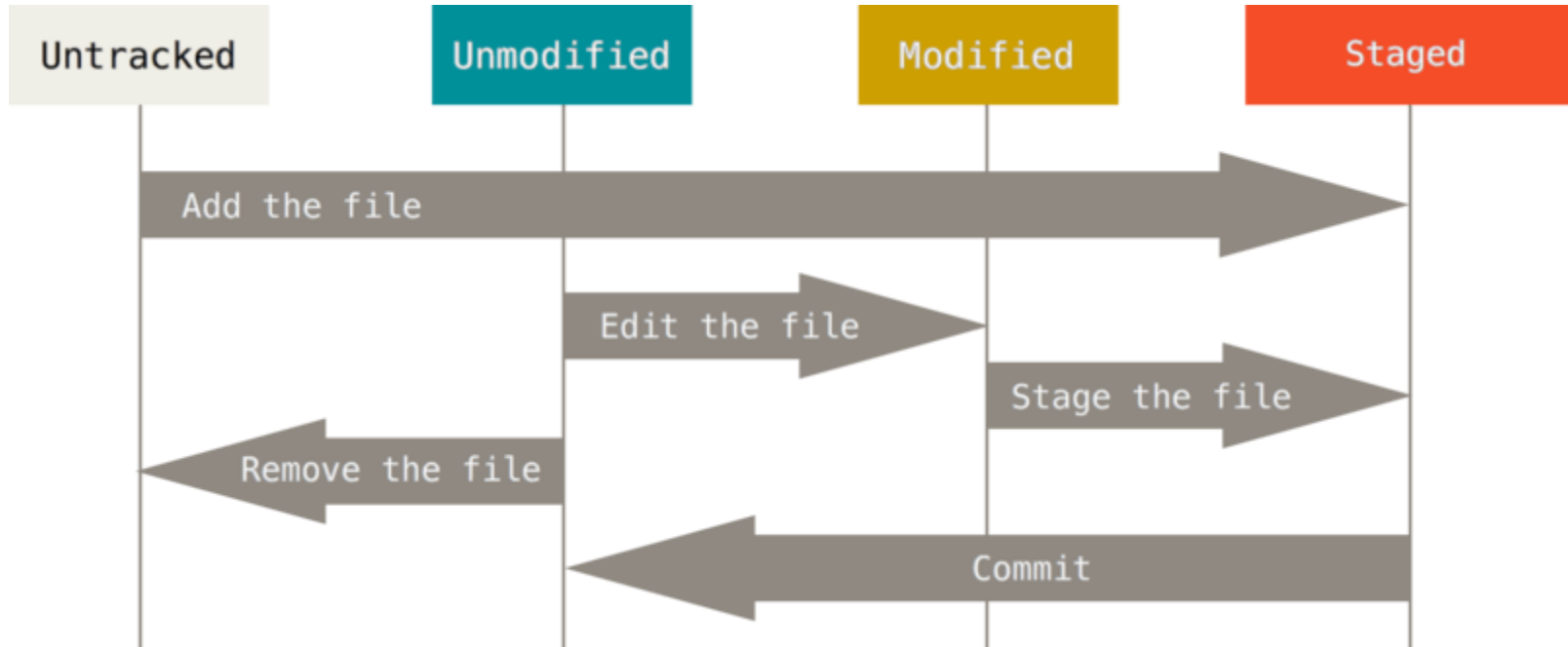
- git add <file 또는 디렉토리>
  - untracked 파일을 tracked로 변경
  - 동시에 staged로 변경
  - 이미 tracked이고 modified인 경우 staged로 변경
  - 디렉토리를 지정한 경우 하위의 모든 파일까지 변경

### Commit하기

- git commit
  - staged된 파일만 커밋할 수 있음
- 커밋 메시지 입력(필수)
  - git commit -m "의미있는 메시지"
  - git commit 후 편집기에서 입력
  - git commit -a는 staged로 변경하고 커밋

# 3W git : git CLI 사용법

## 소스 관리



## 소스 관리

### 소스 만들기

- 작업 디렉토리에 다음 파일 만들기
  - hello.py

```
#!/usr/bin/env python
print("hello world")
```
- 확인
  - git status

### 관리 대상으로 만들기

- git add hello.py
- git status

### Commit하기

- git commit -m "hello.py 생성"
- git status
- git status --short 또는 git status -s

### 수정

- hello.py 수정

```
#!/usr/bin/env python
print("hello world")
print("hello git")
```
- git status
- git commit -m "hello git added"
- git status
- git add hello.py
- git status
- git commit hello.py
- 메시지 편집
- git status

### 추가

- test.txt 만들기

```
test text
```
- git status
- git add test.txt

## 소스 관리

### 소스 관리에서 제외 하기

- git add dir : dir과 하위 디렉토리에 있는 모든 파일들이 stage됨, .gitignore파일을 만들고 무시할 패턴을 적으면 관리되지 않음
- .gitignore 파일의 패턴 사용법
  - 아무것도 없는 라인이나, # 로 시작하는 라인은 무시
  - 디렉토리는 슬래시(/)를 끝에 사용하는 것으로 표현
  - 슬래시(/)로 시작하면 하위 디렉토리에 적용되지(Recursivity) 않음
  - 느낌표(!)로 시작하는 패턴의 파일은 무시하지 않음
  - 표준 Glob 패턴을 사용
  - 예)
    - \*.swp
  - Glob 패턴(참고 : [위키백과](#))
    - \* : 1이상의 어떤 문자
    - [abc] : a, b, c 중 하나
    - ? : 하나의 어떤 문자
    - [0-9] : 0부터 9까지의 수 중 하나
    - \*\* : 디렉토리 하위의 디렉토리까지 지정

## 소스 관리

### .gitignore 파일 예

```
# 확장자가 .a인 파일 무시
*.a

# 윗 라인에서 확장자가 .a인 파일은 무시하게 했지만 lib.a는 무시하지 않음
!lib.a

# 현재 디렉토리에 있는 TODO파일은 무시하고 subdir/TODO처럼 하위디렉토리에 있는 파일은
무시하지 않음
/TODO

# build/ 디렉토리에 있는 모든 파일은 무시
build/

# doc/notes.txt 파일은 무시하고 doc/server/arch.txt 파일은 무시하지 않음
doc/*.txt

# doc 디렉토리 아래의 모든 .pdf 파일을 무시
doc/**/*.pdf
```

## 소스 관리

### 비교

- git diff
  - 수정했지만 아직 stage하지 않은 파일들을 비교
  - 커밋한 파일과 아직 stage되지 않은 파일 비교
  - stage된 파일과 unstage된 파일 비교
- git diff --staged
  - git diff --cached 동일한 기능
  - 커밋한 것과 staging area에 있는 파일들을 비교

### 커밋 히스토리

- git log
  - 커밋한 이력을 보여줌
  - 최근 커밋이 위에 표시
- git log --patch
  - 각 커밋의 diff 결과
  - git log -p와 동일
- git log --patch -2
  - 최근 2개의 커밋만 비교
- git log --pretty=oneline
- git log --oneline -n 3
- 커밋은 해시값으로 구분
  - 해시값 : 40자의 16진수
  - 주로 앞의 7자리로 사용

## 소스 관리

### 파일 삭제

- git rm : tracked 상태의 파일을 삭제
  - 실제(작업 디렉토리의) 파일도 삭제
- 그리고 커밋해야 함

### 파일 이름 변경

- git mv file\_from file\_to
- 다음 명령과 동일
  - mv file\_from file\_to
  - git rm file\_from
  - git add file\_to

### 실습

- test2.txt 만들기  
test2
- test3.txt 만들기  
test3
- git rm test2.txt
- ls
- git status
- git commit -m "test2 remove"
- git mv test3.txt test4.txt
- ls
- git status
- git commit -m "mv test3 to test4"
- git status



## 소스 관리

### 되돌리기

- 버전 관리 시스템이므로 과거의 스냅샷으로 되돌릴 수 있음, 즉 잘 못 수정된 것을 얼마든지 되돌릴 수 있음
- 하지만 잘 못 되돌린 것은 다시 되돌릴 수 없으므로 되돌리는 작업은 각별한 주의를 요함

### 완료한 커밋 수정

- git commit --amend : 커밋 재작성
- 수정을 한 후 하나의 커밋으로 기록됨

### 파일 상태를 unstage로 변경

- git reset HEAD <파일>

### stage되지 않은 파일을 되돌리기

- git checkout -- <file>

### 실습

- git commit -amend
  - test5.txt, test6.txt 만들기(touch)
  - git add test5.txt
  - git commit -m "test5 test6 added"
  - git status
  - git log
  - git add -m "hello git added"
  - git commit --amend
  - git status
  - git log
- 파일 상태를 unstage로 변경
  - test7.txt, test8.txt 만들기(touch)
  - git add \*
  - git status
  - git reset HEAD test8.txt
  - git status

## 소스 관리

### Modified 파일을 되돌리기

- `git checkout -- <file>`
- 수정한 내용이 모두 사라지므로 주의 해서 사용
- 브랜치를 사용해야 한다.

### 이전 커밋으로 되돌리기

- 마지막 커밋을 취소
  - `git revert <되돌리고 싶은 커밋 이름>`
- 특정 커밋으로 이동
  - `git reset --hard <이동하려는 이전 커밋 이름>`
- 협업을 할 경우 되돌리기를 주의해야 하며, 자신만의 브랜치를 만들어서 하는 것이 좋다.

### 실습

- Modified 파일 되돌리기
  - test7.txt 파일 수정
  - `git status`
  - `git checkout -- test7.txt`
  - 편집기로 test7.txt 열기
  - `git status`
- 이전 커밋으로 되돌리기
  - `git rm test7.txt`
  - `git commit -m "test7 deleted"`
  - `git revert 24fb94a8`
  - `git status`
  - ls로 파일 확인

## branch

### branch란?

- 코드를 통째로 복사해서 독립적으로 개발
- 새로운 작업은 브랜치를 만들어서 하고 완성되면 합병
- 어떻게 브랜치를 만들고 합병할 지 전략 필요
  - 브랜치 워크플로

### branch 명령

- git branch : 브랜치 보기
- git branch <새 브랜치 이름> : 브랜치 만들기
- git checkout <브랜치 이름> : 브랜치 이동
- git checkout -b <새 브랜치 이름>  
: 브랜치를 만들고 이동
- git merge : 브랜치 합병
- git log --oneline --decorate --graph --all  
: 로그에 브랜치가 잘 나타나게 표시
- git branch -d <브랜치 이름> : 브랜치 삭제

### 실습

- 브랜치 확인
  - git branch
- 브랜치 만들기
  - git branch testing
  - git checkout testing
  - git branch
- 파일 수정
  - test8.txt 수정
  - git commit -am "test8 수정"
- 확인
  - git checkout main
  - test8.txt 내용 확인
- 합병
  - git merge testing
- 로그 보기
  - git log --oneline --decorate --graph --all

## 세째시간

7교시

15:00

### ◆ git 개요

- 소스 코드 버전 관리와 git
- git CLI
- 저장소 만들기, 환경 설정

8교시

16:00

### ◆ git CLI 사용법

- 소스 관리(add, commit, 복구)
- branch

9교시

17:00

### ◆ github

- 보안 설정과 ssh키 저장
- 저장소 만들기과 clone
- markdown 편집, push

## 보안 설정과 ssh키 저장

### github란?

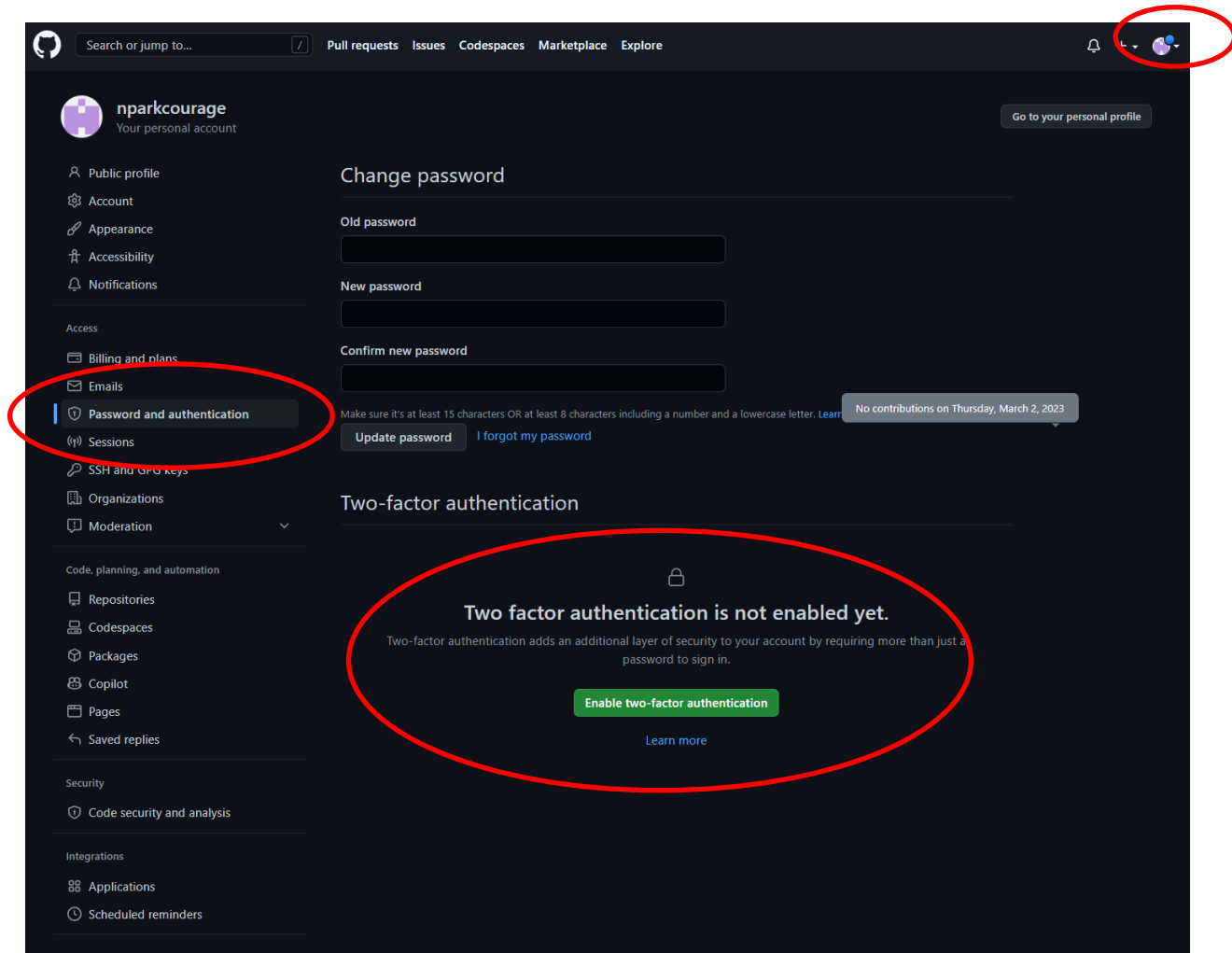
- git을 기반으로 한 원격 저장소 호스팅 서비스
- <https://github.com>
- 계정을 만들고 사용
- 무료, 유상 서비스
- 문서 : <https://docs.github.com/ko>
- 스킬 : <https://skills.github.com/>

### 보안 설정

- "2FA"(투팩터 인증)을 설정
  - <https://docs.github.com/ko/authentication/securing-your-account-with-two-factor-authentication-2fa/about-two-factor-authentication>
  - 보안 강화(권장됨)
- ssh key 생성과 계정에 추가
  - github는 패스워드 입력을 이용한 저장소 사용이 금지됨, ssh key를 사용해야 함
  - <https://docs.github.com/ko/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>
  - <https://docs.github.com/ko/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>

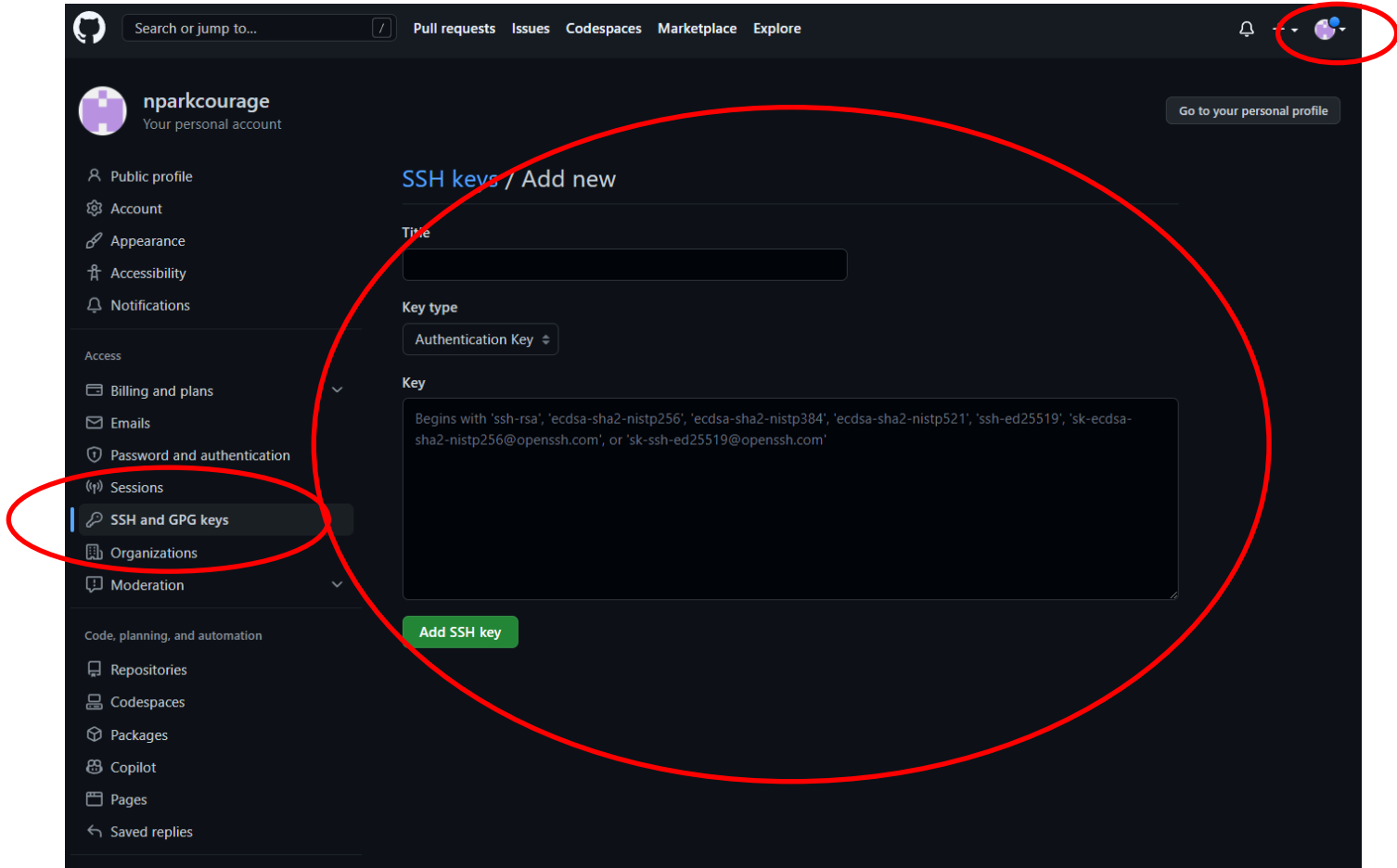
## 보안 설정과 ssh키 저장

“2FA”(투팩터 인증)을 설정



## 보안 설정과 ssh키 저장

### SSH Key 추가



```
$ssh-keygen -t ed25519 -C "your_email@example.com"
```

## 저장소 만들기과 clone

### 저장소 만들기

The screenshot shows the GitHub profile page for user 'nparkcourage'. The user's profile picture is a purple and white pixelated design. The page includes a search bar, navigation links (Pull requests, Issues, Codespaces, Marketplace, Explore), and a dropdown menu with the 'New repository' option highlighted. The 'Popular repositories' section shows a repository named '2023-kau-0504' with 2 HTML files and 2 stars. The '16 contributions in the last year' section displays a calendar grid with green squares indicating contributions. The 'Contribution activity' section shows a bar chart for March 2023, indicating 7 commits in 1 repository. The footer contains the GitHub logo, copyright information, and various links (Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, About).



## 저장소 만들기와 clone

### 저장소 만들기

- Repository name : 2023\_OSS
- Public : 체크(공개 저장소)
- Add a README file : 체크
- Add .gitignore : Python
- Choose a license : MIT License

**Create a new repository**  
A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \* nparkcourage ▾ / **Repository name \*** 2023\_OSS ✓

Great repository names are short and memorable. Need inspiration? How about [animated-dollop?](#)

Description (optional)  
2023 OSS 수업

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

**.gitignore template:** Python ▾

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

**License:** MIT License ▾

This will set `main` as the default branch. Change the default name in your [settings](#).

① You are creating a private repository in your personal account.

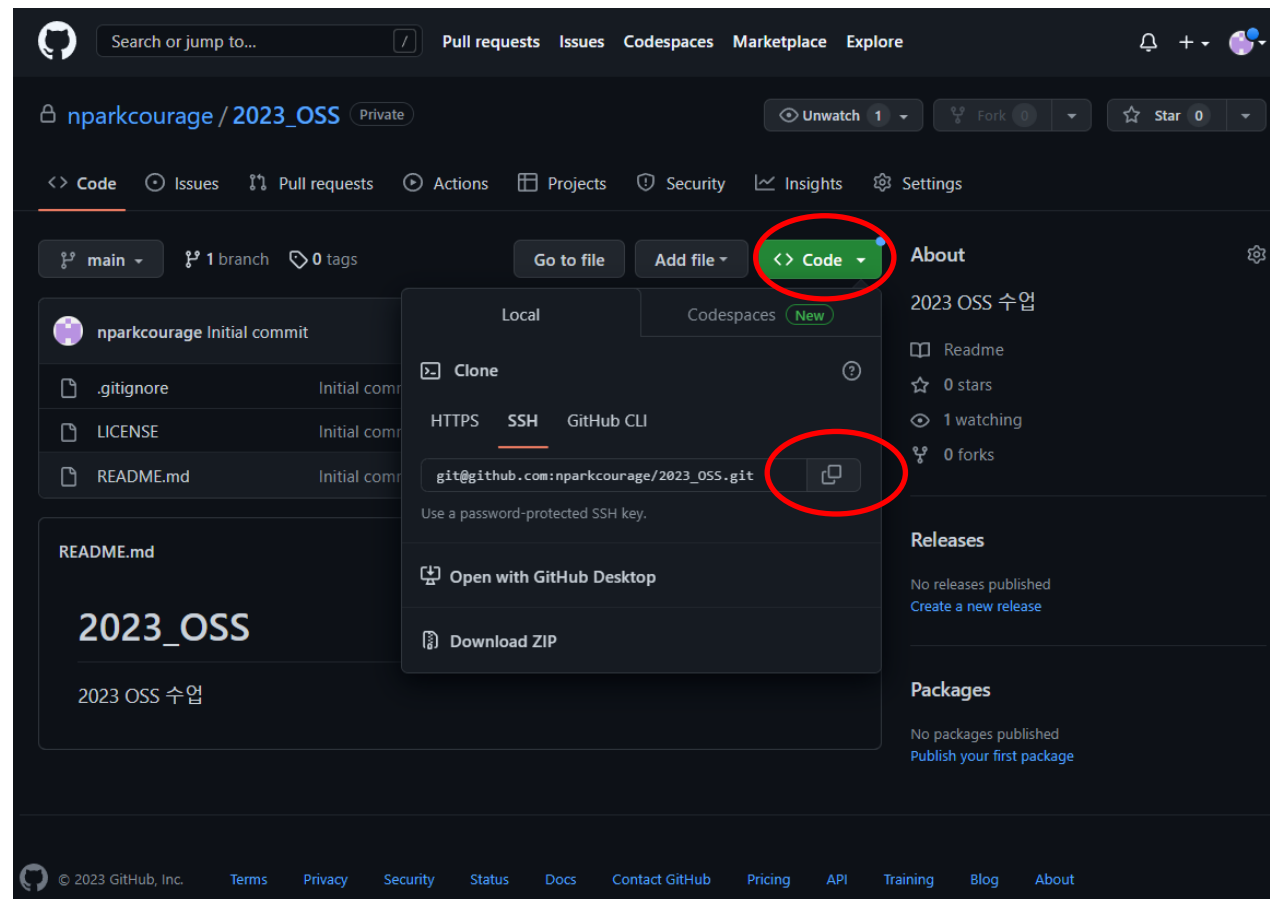
**Create repository**

## 저장소 만들기과 clone

### 원격 저장소 클론

- Clone URL 복사
- 내 PC(WSL)에서 작업
  - `cd ~/projects`
  - `git clone <url>`
  - 확인 :  
`cd 2023_OSS`  
`ll or tree`
- 원격저장소 확인 :  
`git remote -v`

```
git clone git@github.com:nparkcourage/2023_OSS.git
```



## markdown 편집과 push

### markdown

- 텍스트 기반의 마크업 언어로 2004년 존 그루버에 의해 만들어졌으며 쉽게 쓰고 읽을 수 있으며 HTML로 변환이 가능
- github의 README.md에서 사용
- <https://gist.github.com/ihoneymon/652be052a0727ad59601>
- github의 README.md : github에서 기본적으로 보이는 문서
- 장점
  1. 간결하다.
  2. 별도의 도구없이 작성가능하다.
  3. 다양한 형태로 변환이 가능하다.
  4. 텍스트(Text)로 저장되기 때문에 용량이 적어 보관이 용이하다.
  5. 텍스트파일이기 때문에 버전관리시스템을 이용하여 변경이력을 관리할 수 있다.
  6. 지원하는 프로그램과 플랫폼이 다양하다.
- 단점
  1. 표준이 없다.
  2. 표준이 없기 때문에 도구에 따라서 변환방식이나 생성물이 다르다.
  3. 모든 HTML 마크업을 대신하지 못한다.

## markdown 편집과 push

### markdown 사용법

- 줄바꿈 : 3칸 이상 띄어쓰기를 하면 줄이 바뀐다.

- Header :

- 글머리: 1~6까지만 지원

- 목록 :

- 번호 있는 목록
- 번호 없는 목록\*, -, +

- 코드블럭 : ``bash(언어지정, 옵션) 틱(키보드 외쪽 위) ``

- 링크 : [Title](link)

- 수평선 : ---- 또는 <hr/> 등등

- 강조 :

\*single asterisks\*

\_single underscores\_

\*\*double asterisks\*\*

\_\_double underscores\_\_

~~cancelline~~

- 이미지 : ![Alt text](/path/to/img.jpg "Optional title")

- 글머리: 1~6까지만 지원

```
# This is a H1
## This is a H2
### This is a H3
#### This is a H4
##### This is a H5
##### This is a H6
```

## This is a H1

## This is a H2

### This is a H3

#### This is a H4

##### This is a H5

###### This is a H6

현재까지는 어떤 번호를 입력해도 순서는 내림차순으로 정의된다.

1. 첫번째
3. 세번째
2. 두번째

1. 첫번째
2. 세번째
3. 두번째

## markdown 편집과 push

### README.md 파일 편집

- 저장소 루트의 README.md
- w3 디렉토리를 만들고 이 디렉토리에 README.md 파일을 새로 만들고 편집
- wsl에서 직접 편집해도 되고 윈도우에서 편집해서 복사해도 됨

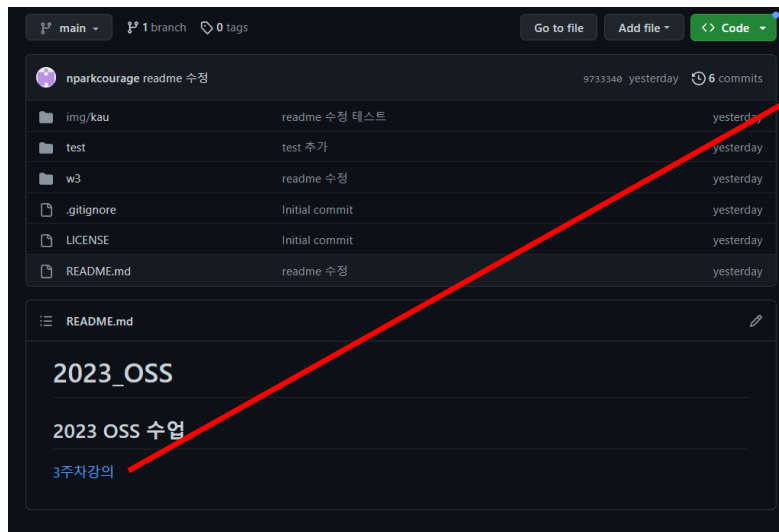
### 소스 저장소로 푸시

- github에서 clone한 소스를 커밋한 후 원격 저장소로 푸시
  - `cd ~/projects/2023_OSS`
  - `git add .` : 모든 파일들을 stage로 보내기
  - `git commit -m "readme 편집"`
  - `git push`

## 과제

### github에 README.md 수정해서 올리기

- 2023\_OSS/README.md 파일 수정
  - w3하위 디렉토리의 README.md 링크 포함
- 2023\_OSS/w3/README.md 만들고 아래 내용 포함
  - 이미지 링크
  - 링크(LMS, ProGit)
  - git 명령어 설명
  - 2주차 숙제를 코드 블록으로
  - 마크다운 사용법 일부



내부이미지

내부링크



## 과제

### 2주차 숙제

```
#!/usr/bin/env bash
echo "-----"
echo "name :"
echo "박노현"
echo

echo "-----"
echo "student id :"
echo "202321018"

file_path=`find /home/kau2/ -name w2_homework.txt 2> /dev/null`
echo "-----"
echo
echo "file path :"
echo $file_path
echo

line_num=`wc -l $file_path | cut -c 1 -`
echo "-----"
echo "line number :"
echo $line_num
echo

echo "-----"
echo "lask line :"
tail -n 1 $file_path
```

### 마크다운

#### 목록

번호 있는 목록 : 내림차순 정렬

1. 첫번째
2. 세번째
3. 두번째

번호 없는 목록 : \*, -, +

- 첫번째
- 세번째
- 두번째

- 빨강
  - 녹색
    - 파랑

#### 강조

*single asterisks*

*single underscores*

**double asterisks**

**double underscores**

cancelline

## 과제

### 제출

- 다음 주 월요일(27) 자정(24시)까지(KST;한국 표준시)
- 자신의 github에 만들고 LMS에 보고서 제출
- 보고서에 과제 저장소의 github URL 표시
- LMS 제출은 LMS의 공지 확인하고 공지내용에 맞게 제출(조교에게 문의)