

# Git 버전 관리 실습

nodejs v14.17.0  
git version 2.18.0.windows.1  
Windows 10

먼저 사용 하는 PC에 Git을 설치하고  
Github 웹 사이트에도 계정을 만들어야 합니다.

다음은 CLI 환경에서 작업 합니다. (Window는 명령프롬프트(CMD)에서 작업)  
(실행 하려는 위치에서 Shift + 오른쪽 클릭 > 명령프롬프트, 파워셸, 터미널 등)  
또는 실행 > cmd 열기 합니다.



```
C:\WINDOWS\system32\cmd
Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\beomj>cd C:\Users\beomj\OneDrive\OneDrive문서\mulcam_work2024\server20240527
C:\Users\beomj\OneDrive\OneDrive문서\mulcam_work2024\server20240527>
```

깃 버전 확인 명령

> git --version

깃 업데이트

> git update-git-for-windows

1. git 사용자 이름과 이메일 등록

- > git config --global user.name "사용자 이름"
- > git config --global user.email "사용자 이메일"

2. git 사용자 이름과 이메일 확인

- > git config user.name
- > git config user.email
- > git config --global --list

### 3. nodejs 프로젝트 생성하기

> npm init -y

### 4. 지역 저장소 git 초기화

> git init

### 5. gitignore.io를 이용한 .gitignore 생성

주소창에 [gitignore.io](https://www.toptal.com/developers/gitignore)를 치면 아래 경로로 이동한다.

<https://www.toptal.com/developers/gitignore>

생성 창에 Node Windows를 입력 하고 생성 클릭

생성된 내용을 프로젝트 루트에 .gitignore 파일로 저장한다.

.gitignore 파일에 기록된 파일들은 버전 관리 추적 대상에서 제외된다.

### 6. 프로젝트 파일들을 지역 저장소에 저장

워킹디렉토리	> add	스테이지 영역	> commit	지역 저장소
--------	-------	---------	----------	--------

커밋 상태 확인

> git log

git add --all 후 파일 상태 확인

> git status

워킹디렉토리의 모든 작업 파일을 스테이지 영역에 추가

> git add --all

스테이지 영역의 파일을 다시 워킹 디렉토리로 돌려 보내기

> git rm --cached <파일이름>

프로젝트 파일을 지역 저장소에 저장

> git commit -m "커밋 메세지"

### 7. 새로운 브랜치를 만들어서 파일을 수정하고 지역 저장소에 저장하기

새 브랜치 만들고 checkout

> git branch 새브랜치 (가장 최근의 커밋을 가리킨다)

> git checkout 새브랜치 (HEAD를 master에서 새브랜치로 변경)

파일 내용을 수정하고 새로 저장소에 commit

> git add --all

> git commit -m "커밋내용"

### 8. 파일 내용 수정 전으로 돌아가기 (master 브랜치로 돌아가기)

> git checkout master

(브랜치는 완벽한 독립공간이다.)

-- 새 브랜치로 checkout하면 파일 수정했던 내용이 살아난다.

-- master브랜치로 checkout하고 git log 하면 새 커밋이 나타나지 않을 수도 있다.

로컬 브랜치 목록 조회

> git branch

-- 새로운 수정에 대한 분기를 commit하지 않으면 checkout 되지 않음.

#### 9. 브랜치 목록 조회

> git branch (로컬 브랜치 목록 조회)

> git branch -r (원격 브랜치 목록 조회)

> git branch -a (모든 브랜치 목록 조회)

#### 10. 파일 상태 구분

tracked	unmodified	커밋되고 난 후에 아무 것도 수정되지 않은 상태
	modified	커밋되고 난 후에 수정 된 상태
	staged	add를 통해 스테이징 영역에 추가된 상태
untracked	한번도 스테이징 영역에 추가 되지 않은 파일의 상태	

#### 11. 방금 수행했던 커밋 변경하기

커밋한 지점에서 작업 파일을 새로 변경 한 후 기존의 commit에 추가한다.

> git commit --amend

-- vi 편집기가 실행 되면 저장 메시지와 작업 파일 목록을 편집 할 수 있다.

-- i키를 누르면 vi 편집 모드로 들어간다.

-- 편집이 끝나면 vi편집기 저장 명령으로 내용을 저장한다.

-- esc키를 누그로 :wq 명령으로 저장 후 vi 종료.

> git log

-- 커밋 로그를 확인 하면 현재 커밋의 내용이 변경된 것을 확인 할 수 있다.

> git checkout master

-- 수정 전의 내용으로 돌아간다.

> git checkout newbranch

-- 수정 후의 내용으로 다시 돌아온다.

#### 12. 이전의 버전관리 도구인 Subversion과 같은 도구는 변경된 부분만을 저장하는 반면 Git은 스냅샷을 저장한다. Git의 각각 커밋은 40자리 문자열인 SHA-1 해시코드로 표시된다.

-- 일반적으로 master 브랜치에서 새로운 작업을 하진 않고 토픽 브랜치(topic branch)를 만들어 작업한 후에 master 브랜치와 병합(merge)한다. 그것이 더 안전하기 때문이다.

#### 13. 브랜치 병합(merge)

작업한 브랜치를 특정 브랜치와 병합(merge)한다.

(Fast-Forward 병합 - 앞으로 빨리 가기)

병합전에 마스터 브랜치로 전환한다.

> git checkout master

마스터 브랜치로 전환 후 작업 브랜치와 병합한다.

> git merge 작업브랜치 (master브랜치가 작업브랜치로 옮겨지면서 병합 됨)

병합한 후에는 작업했던 브랜치는 삭제 해 준다.

> git branch -D 작업브랜치

Fast-Forward 병합이 아닌 일반 병합은 새로운 커밋을 생성한다.

병합 시 유의 사항

- 1) 병합은 브랜치 레벨에서만 병합이 가능하다.
- 2) 현재 브랜치에 없는 커밋만 병합 가능
- 3) 변경 대상 브랜치로 전환 후 병합

(주의) 병합할 두 브랜치가 동일 파일의 동일한 곳을 수정 했다면 충돌이 발생해서 병합이 거부된다. 이럴 때는 병합을 먼저 해결하고 커밋을 재 수정 해야 한다.

#### 14. stash에 작업 내용을 임시 저장

stash : 임시 기억 클립보드 (HEAD가 가리키는 어떤 브랜치도 이용 가능)

새 작업 브랜치 생성

```
> git branch new_work
> git checkout new_work
-- 새로운 작업을 한다.
```

작업 브랜치에서 stash로 저장

```
> git stash (작업 하던 내용이 stash에 저장되고 마스터로 전환)
```

stash에 임시 저장된 내용 반영하기

```
> git stash pop (적용 후 임시 저장소에서 제거)
> git stash apply (적용)
```

#### 15. 두 브랜치에서 같은 내용을 수정 했을 때 커밋 후 병합하기

- 두 브랜치에서 같은 위치를 수정하고 모두 커밋 후 병합을 하면 병합은 됨
- 병합 오류가 발생(현재 수정내용과 병합할 커밋의 수정 내용이 모두 보임)
- VS code에서는 "Accept Current Change | Accept Incoming Change | ..." 보임
- 변경된 커밋을 적용 하려면 Accept Incoming Change를 선택하면 바로 수정 됨.

```
5 // aaa브랜치에서 추가함 - stash에 저장
6 function add(a, b) {
  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
7 <<<<<< HEAD (Current Change)
8 // master에서 새로 만들...
9 =====
10 // 오페라! 오페라!
11 // edit aaa2
12 >>>>>> aaa2 (Incoming Change)
13 let result = a + b;
14 return a + b;
15 }
```

-> 병합 시 충돌되는 부분 수정 후 다시 commit 또는 commit -a 해야 한다.

## GitHub 사용하기

#### 1. github 사이트에 회원 가입

<https://github.com/>

- 가입메일로 인증 확인 메일이 날아옵니다. 인증 확인 해야 guthub 사용 가능.
- 인증 완료되면 Settings 메뉴에서 본인 정보 입력.
- 사용할 프로젝트의 레파지토리 생성. (New Repository)

## 2. local저장소에 github remote

- (명령어는 깃허브에 새 레파지토리 생성 하면 보여진다.)

> git remote add <저장소별칭> <원격저장소 url>

## 3. 지역 저장소 데이터를 원격저장소에 push

> git push -u origin <지역저장소 브랜치>

## 4. 지역 저장소의 모든 브랜치 push

> git push -u origin --all

## 5. 개발자가 원격 저장소의 내용을 pull 한 후에 원격 저장소의 내용이 변경되면 Push 거절 된다.

Push 거절 된 경우에는 원격 저장소의 커밋 이력을 가져와 개발자의 커밋 이력을 반영하기 위한 병합을 수행 해야 함.

> git fetch origin

--- 변경된 원격 저장소의 최신 커밋 이력을 가져 온다.

수동으로 지역 저장소와 병합 하기

원격 브랜치 : origin/master의 형식으로 지역 저장소에서 참조.

> git checkout master

> git merge origin/master

> git push origin master

## 6. Github 사이트에서 내용을 직접 수정 한 후 원격 저장소 내용 Push 해보기

1) 원격 저장소의 소스 수정 (Commit change)

2) 지역 저장소에서 파일 수정 후 커밋

3) 지역 저장소 원격 저장소에 Push

> git push origin master (Push 거부 됨)

4) 변경이력 가져오기

> git fetch origin

5) 원격 브랜치와 병합

> git checkout master

> git merge origin/master

6) 병합 충돌 후 원격 저장소에 Push

> git add --all

> git commit -m "커밋 메세지"

> git push origin master

## 7. 클론(Clone)

원격 저장소의 내용을 지역 저장소로 복제하는 것.

> git clone 원격저장소url

## 8. 원격 저장소 변경

git remote set-url origin <변경할 원격 저장소 주소>

## 9. 자격증명 확인 및 업데이트

git config --global credential.helper cache

git config --global credential.helper 'cache --timeout=3600'

## 10. 캐시 삭제 및 비활성화

# 자격 증명 비활성화

```
git config --global credential.helper
```

```
git config --global --unset credential.helper
```

# 매번 물어보도록 빈 문자열 설정하기

```
git config --global credential.helper ""
```

# 자격 증명 삭제

```
git credential-cache exit
```

## 11. 설정 된 remote 확인 및 재 설정

```
git config - -global user.name "사용자"
```

```
git config - -global user.emmail "사용자이메일"
```

```
git remote -v
```

```
git remote set-url origin https://github.com/comstudyschool/sw202407day05.git
```

## 12. Github에서 토큰 생성

"Developer settings"

-> "Personal access tokens"

-> "Tokens (classic)"으로 이동

-> "Generate new token" 버튼을 클릭.

## 13. 다른 계정의 레파지토리 가져오기 명령

```
git remote add teacher https://github.com/comstudyschool/sw20240729js.git
```

# 확인

```
git remote -v
```

```
git fetch teacher/main
```

# remote repository 에서 필요한 파일만 가져온다.. ( ex) [root]/./day10/images )

```
git checkout teacher/main -- [가져올 remote repository path]
```