

졸업작품(종합설계)

개인 포트폴리오



ICE BREAKER

JiYe Hyun GitHub Address : <https://github.com/i-Jea-i>

컴퓨터정보공학과
2학년 A반
현지예

CONTENTS

01.

졸업작품

- 작품 소개
- 배경 및 필요성
- 작품 구성
- 말은 파트
- API와 API사용 방법

02.

Git이란?

- Git
- Git의 사용



01

졸업작품

For You

For You

- For You는 항상 사용자들을 위해~!
- 집에서도 누구나 스스로 건강 관리를 할 수 있는
안드로이드 어플리케이션
- 여러가지 IoT 기기를 통한 신체 건강 관리 가능
- 약 및 영양제 복용에 대한 관리 가능

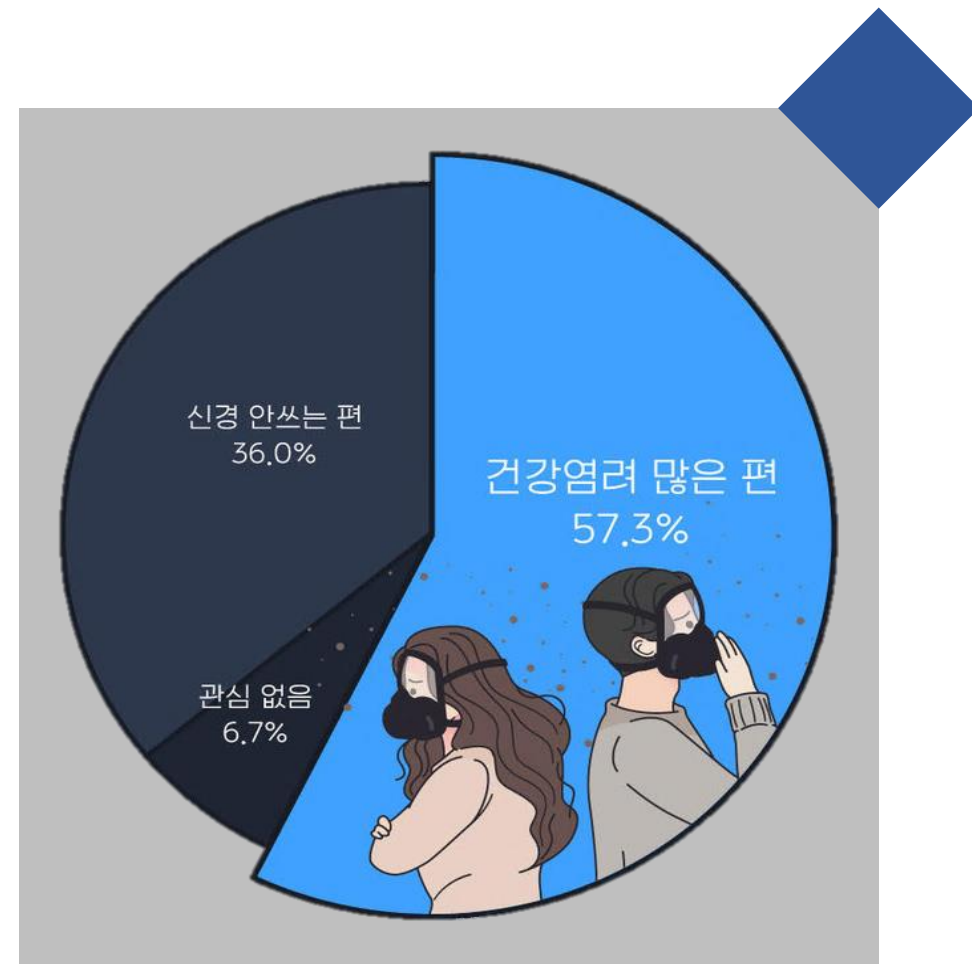


프로젝트 배경

- 코로나19로 인하여 집에 머무는 시간이 늘어남
- 코로나19가 장기화 되면서 많은 사람들이 건강 분야에 관심을 가짐
- 스스로 건강 관리를 하려는 사람들이 늘어남

For You의 필요성

- 'For You'는 이런 언택트 시대에 적합한 헬스케어 앱임
- 언제든지 앱과 IoT 기기를 활용하여 헬스케어 가능
- 심박수 측정과 약 및 영양제 복용 등의 기능이 대표적
- 이 외로 병원 위치와 간단한 치료 영상을 제공, 커뮤니티 기능도 사용 가능



직장인 2,420 설문조사 결과

출처 : 잡코리아, 알바몬

작품 구성



신체 건강 체크



병원 위치 및 영상 제공



안드로이드 앱 (For You)

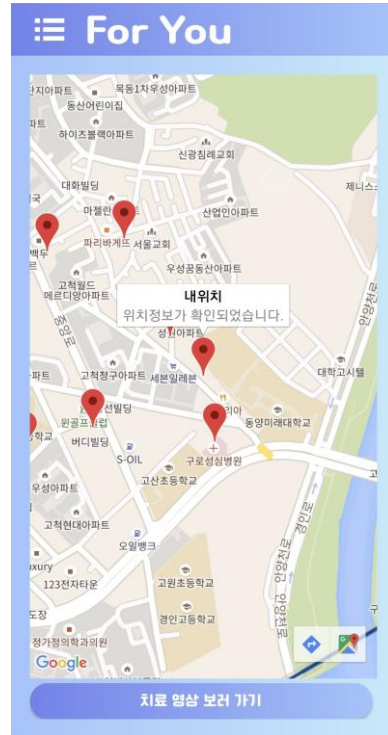


스마트 약통 & 복용 관리



커뮤니티

말은 파트



병원 위치 제공

- 나의 현위치를 알 수 있음
- 현위치를 기준으로 하여 주변에 있는 병원의 위치를 띄워서 보여줌



간단한 치료 영상 제공

- 요가, 스트레칭, 마사지 등 혼자서도 따라할 수 있는 쉽고 간단한 치료 영상을 주제별로 나누어 제공
- 유튜브에 올라온 영상을 사용

API란?

- API
 - : Application Programming Interface (응용 프로그래밍 인터페이스)
- 다른 사람이 작성해놓은 코드인 라이브러리를 사용할 수 있게 도와주는 설명서, 설계도 그리고 코드 그 자체
- 사용하는 이유
 - : 이미 만들어진 코드 중 자주 쓰이는 것을 쉽게 이용하기 위함
- 반복적인 코드 작성을 없애기 위해 언제든지 필요한 곳에 호출하여 class나 funtion으로 만들어짐
- 사용 방법
 - : API마다 사용 방법은 천차만별이므로, 각 API 문서를 읽어보며 사용하면 됨. Android의 경우, Gradle의 dependency에 사용하려는 라이브러리를 추가해주는 식으로 사용 가능하다.
(+ Gradle이란, 빌드 자동화 도구이다)

Google Map API 사용 방법

- Google Developers Console 사이트 (<https://console.developers.google.com/apis/dashboard>)에 접속 후 '프로젝트 만들기'를 클릭하여 구글 지도를 사용할 프로젝트 생성
- '+API 및 서비스 사용 설정'을 클릭하여, 'Maps SDK for Android'를 검색한 후 선택하고 '사용'을 클릭하면 API 활성화 완료
- 메뉴에서 'API 및 서비스' > '사용자 인증 정보' > '사용자 인증 정보 만들기' 클릭
- '사용자 인증 정보 만들기'에서 'API 키'를 클릭 후 '키 제한' 클릭
- 앞에서 생성된 API 키에 애플리케이션 제한사항, API 제한사항 등 API 사용에 대한 사용 제한을 설정
- 지금까지 설정한 내용을 안드로이드 스튜디오에도 적용
 - + Android Studio에서 설정할 내용
 - : AndroidManifest.xml에서 <application> 태그 하위요소로 <meta-data> 태그를 사용하여 앞에서 생성한 API 키를 입력, 'Google Play services' 라이브러리 패키지 설치, 모듈 app의 build.gradle 파일에 Google Play services 라이브러리 추가
- xml 및 java 파일에 코딩하여 Google Map API 사용

YoutubeAndroidPlayer API 사용 방법

- 앞에서 설명한 Google Map API 발급 과정과 동일하게 Youtube Data API를 발급
 - YoutubeAndroidPlayer API 라이브러리 다운로드
다운로드 주소 : developers.google.com/youtube/android/player/downloads
 - 다운로드 받은 라이브러리를 압축을 풀어, 'Project – app – libs'에 붙여넣음
 - API를 사용하기 위해 jar 파일을 등록
'File – Project Structure – Dependencies – (+) – Jar Dependency'로 들어가 jar 파일을 저장한 경로를 복사하여 붙여넣고, 'OK'를 클릭 (자동으로 Sync가 진행)
 - AndroidManifest.xml에 인터넷 권한 추가
 - xml 및 java 파일에 코딩하여 YoutubeAndroidPlayer API 사용
- 주의해야 할 점 : java 파일에서 AppCompatActivity가 아닌 YoutubeBaseActivity를 extends해야 함
- 가상 디바이스 AVD는 오류가 잦아, 실제 기기 사용을 권장



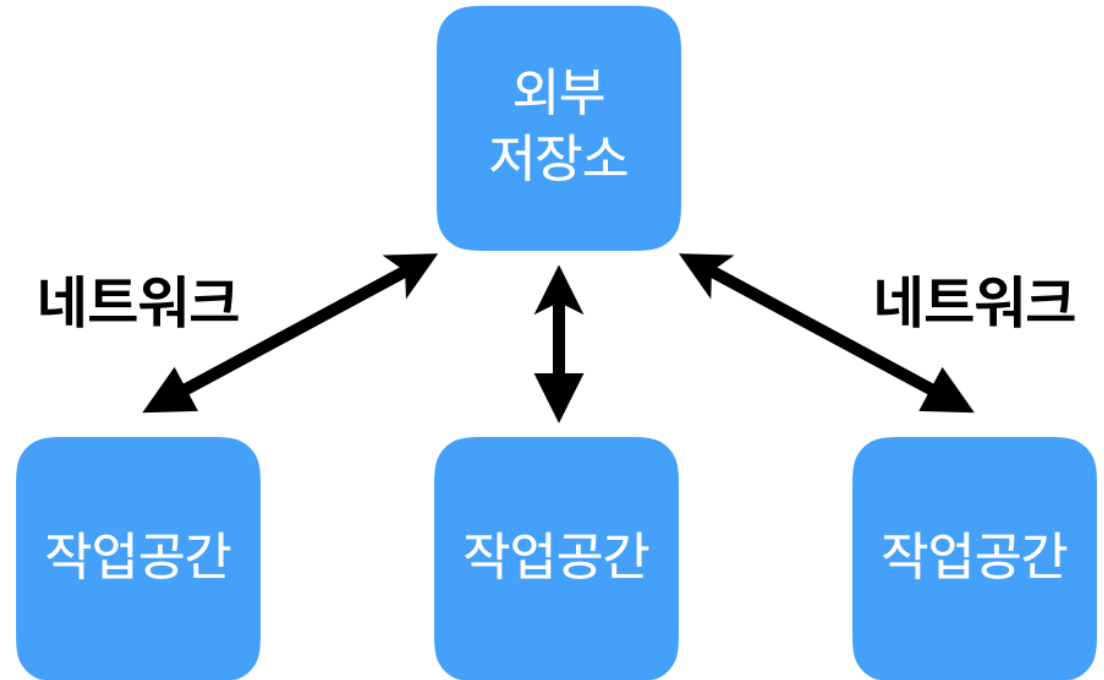
02

Git이란?

Git&Github

버전 관리 시스템이란?

- 프로젝트 파일의 변경사항을 추적할 수 있도록 도와주는 시스템
- 사용자가 변경한 모든 내용을 버전별로 관리하고, 이것이 저장되는 곳을 'repository'라고 부르는 것
- 사용자가 repository에 변경한 내용을 저장하는 것을 'commit'이라고 함
- 일반 버전 관리 시스템은 하나의 중앙 저장소에 사용자들이 변경사항을 저장



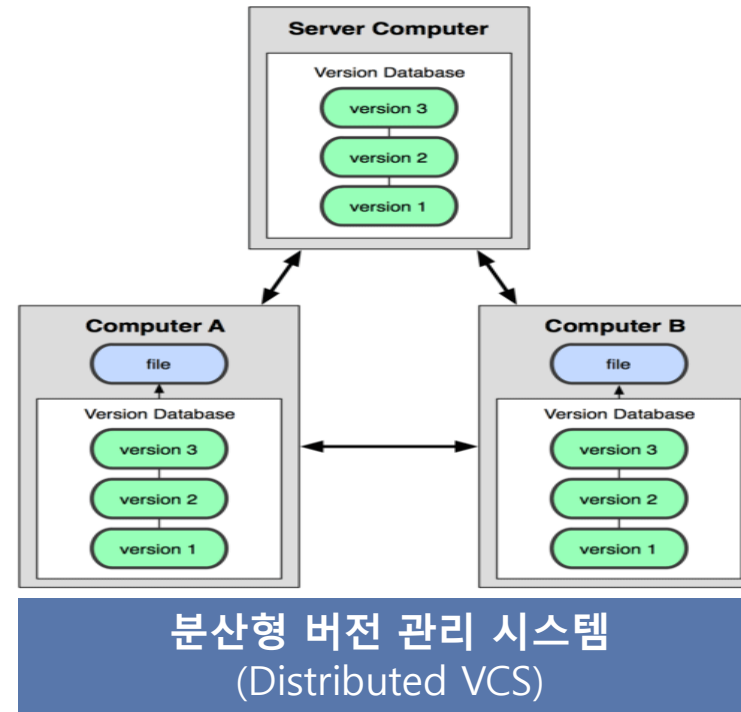
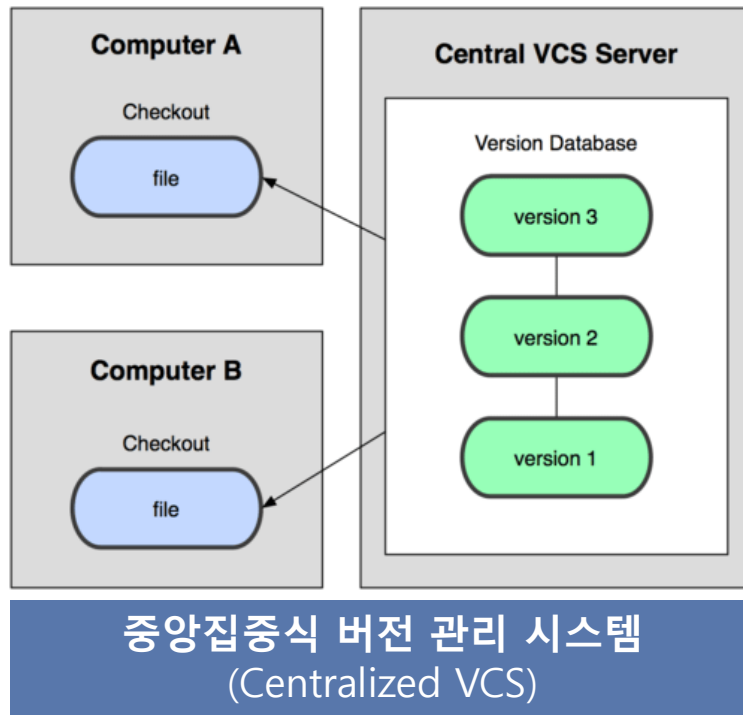
분산 버전 관리 시스템이란?

- 버전 관리 시스템과 같은 구조를 가지고 있지만, 개인 저장소를 갖고 있다는 차이점이 존재
- '작업공간 <-> 외부 저장소'라는 일반적 패턴에서
 '작업공간 <-> 개인 저장소 <-> 외부 저장소'라는 패턴으로 변경
- 네트워크에 연결이 되어있지 않은 환경에서도 작업 가능
- 상황에 따라 공개를 원하지 않으면, 개인 저장소에서 개인적으로 작업 가능

Git이란?

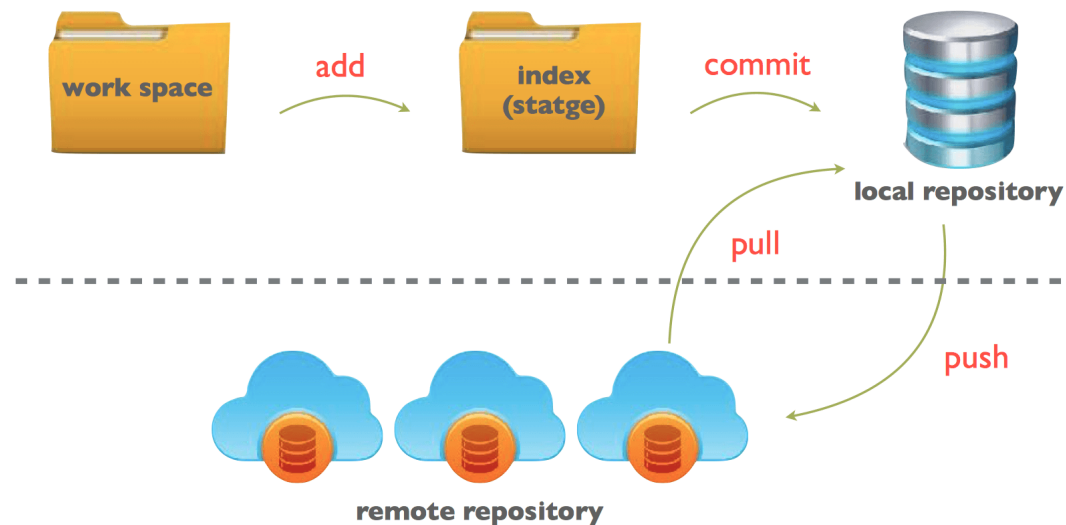
- 분산 버전 관리 시스템의 한 종류
- 빠른 속도와 성능을 바탕으로, 분산 작업이 가능
- Staging area가 존재해 index라는 추가적인 과정을 거침
- Git의 가장 대표적인 특징 : branch 모델이 존재한다는 점이 있음

중앙집중식 버전 관리 시스템 & 분산형 버전 관리 시스템



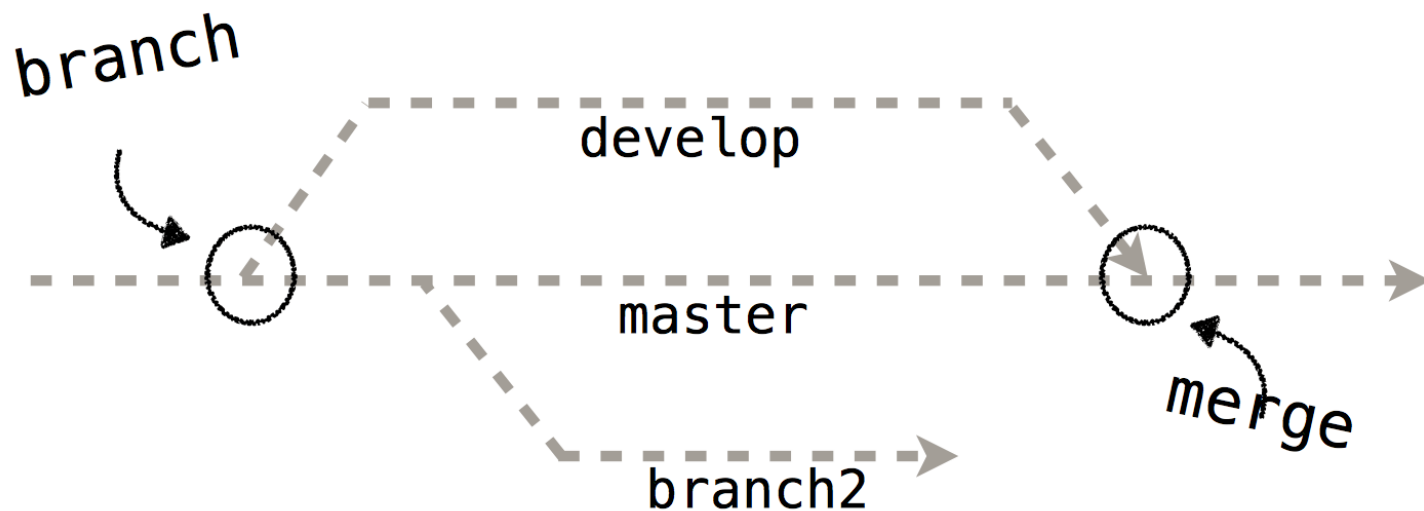
- 중앙집중형 : 저장소가 서버에 존재, 서버와 연결이 끊어지면 기존 소스 수정 외의 작업 불가
- 분산형 : 각 개발자마다의 저장소가 존재, 작업 후 동시에 서버로 반영 가능, 오프라인에서도 commit하며 작업 가능
- 복구가 불가능한 중앙집중형과 다르게, 분산형은 복구가 가능

Git의 구조



- Git은 아래의 순서를 거쳐 소스코드 버전을 관리
- Work Space > Index > Internal Storage > External Storage
- Work Space : 현재 프로젝트의 작업이 이뤄지는 곳
- Index : Work Space의 내용이 Internal Storage에 저장되기 전 올려지는 중간 단계 공간
- Internal Storage : Git이 설치된 컴퓨터 저장공간
- External Storage : 외부에 저장되는 중앙 저장소 (Github가 이에 해당하는 대표적인 서비스)

Branch 모델



- Git에는 'Branch 모델'이라는 대표적인 기능이 존재
- 그림처럼 가지치기(Branch)를 통해 따로 개발했다가, 나중에 합치는(Merge) 기능
- Branch 모델은 Commit을 통해 Head를 생성하다 Branch를 통해 새로운 가지를 치고, Checkout으로 작업 환경을 변경한 후 Merge로 최종적으로 합치는 방법



Github란?

- Git이라는 시스템의 일부를 서비스하는 기업의 이름 즉, 중앙 저장소 서비스를 제공하는 기업
- 저장소에 올린 소스코드를 공개한다는 조건하에 무료로 서비스를 제공
- 소스코드를 비공개하고 싶다면, 유료로 이용 가능
- Private Repository가 일부 무료가 됨
- 오픈된 소스코드가 많아지면서 Github는 크게 성장할 수 있었음

Git

- 2005년, 리누스 토발즈가 개발
주니오 하마노가 소프트웨어 유지보수
- 현재는 오픈 프로젝트
 - + <https://github.com/git>
 - + <https://git.kernel.org/pub/scm/git/git.git>

CLI

- CLI vs GUI
 - GUI 프로그램의 대부분은 Git 기능 중 일부만 구현 (비교적 단순)
 - CLI를 사용할 줄 알면 GUI도 사용 가능 (반대의 경우는 불가능)
 - GUI는 GUI만 사용하고 싶어도 CLI가 기본으로 설치되는 도구임

Git의 설치

- Git 설치 주소 : <https://git-scm.com/>
- 참고 : <https://goddaehee.tistory.com/216>
- 위 주소들을 참고하여 에디터를 설정하고, Path 설정을 맞춰서 설치 가능
 - + Path 설정
- Use Git from Git Bash only : Git bash에서만 Git Command 수행
- Use Git from the Windows Command Prompt
 - : Git을 환경변수에 등록하고, 윈도우 cmd 등에서도 Git 사용 가능
- Use Git and optional Unix tools from the Windows Command Prompt
 - : 윈도우 cmd에서 Git과 유닉스 도구를 사용할 경우 환경변수에 추가

CLI git bash

- CLI : Command Line Interface

- 명령어의 구조

: 명령어의 맨 앞에는 항상 'git'을 써줌

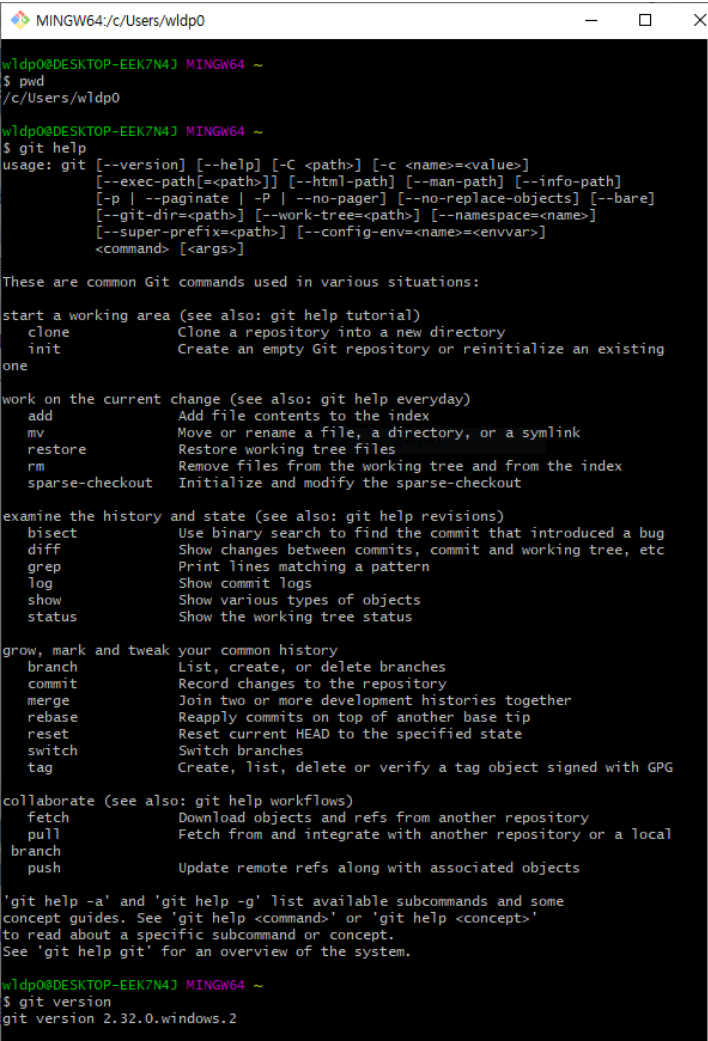
+ working directory 상의 변경 내용을
staging area에 추가 : `$ git add`

+ 의미있는 변화를 기록 : `$ git commit`

+ 현재 폴더 확인 : `$ pwd`

+ 도움말 보기 : `$ git help`

+ 버전 보기 : `$ git version`



```
MINGW64: c:/Users/wldp0
wldp0@DESKTOP-EEK7N4J MINGW64 ~
$ pwd
/c/Users/wldp0

wldp0@DESKTOP-EEK7N4J MINGW64 ~
$ git help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing
            one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index
  sparse-checkout  Initialize and modify the sparse-checkout


examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status


grow, mark and tweak your common history
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG


collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local
            branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

wldp0@DESKTOP-EEK7N4J MINGW64 ~
$ git version
git version 2.32.0.windows.2
```

Git Bash

Git의 주요 명령어

- git status

- + 현재 commit이 필요한 가지 등 여러 상태 표시
- + 빨간 글씨일 경우, 깃이 아직 관리를 시작하지 않은 (Untracked) 상태임을 의미
- + .idea/나 .vscode/와 같은 폴더들은 작성 프로그램에 따라 다르며, 앞에 ' .'이 붙는다면 평소에 숨기는 파일임을 의미
- + 보안에 신경을 써야하는 중요한 파일들은 git을 통해 관리하지 않는 것을 권장

- git status

- + 명령시 commit 대기 상태로 관리할 파일을 전환
- + Untracked에서 Staged로 올라갔다는 것을 의미
- + commit 대기된 파일을 되돌리고자 한다면 'rm --cached' 파일명을 이용
- + 이 때, '--cached'를 붙이지 않으면 원본 파일이 삭제되는 경우가 발생할 수도 있기 때문에 붙여주는 것을 권장

Git의 주요 명령어

- git commit

- + 작업 진행 중 문제가 발생할 경우, 돌아올 수 있는 지점을 저장하는 명령어
- + 백업과 비슷한 개념
- + 변경사항을 add 후 commit할 경우, 많은 과정을 거치며 효율이 떨어짐
- + 'git commit -a' 명령어는 add와 commit을 한 번에 실행시켜주며, commit이 필요한 모든 파일들을 한 번에 commi해줌
- + commit을 할 때는 어떤 부분을 수정했는지에 대해 상세하게 메시지를 작성하는 것이 좋음

- git remote

- + git remote add 리모트명 주소 명령을 통해 원하는 리모트 이름으로 등록하고자 하는 주소를 깃에게 기억하도록 하는 명령어
- + 메일에서의 주소록과 비슷한 개념
- + get-url 옵션을 사용한다면 각 리모트에 대해 어떤 주소가 등록되어 있는지 확인 가능
- + 적어도 하나의 리모트를 등록하고 난 다음, git remote만 입력하게 되면 여태까지 등록해놓은 모든 리모트들을 리스트 형태로 확인 가능

Git의 주요 명령어

- git push

- + git push 리모트명 브랜치명 명령을 사용하게 되면 remote를 통해 등록한 주소로 여태까지 commit 했던 내역들이 보여지게 됨
- + 아래 사진에서는 git push origin master 명령을 통해 깃허브로 커밋 내용을 보냄

```
C:\Users\LEE\Documents\GitHub>git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 654 bytes | 327.00 KiB/s, done
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/Sy2tema/PersonalPractice.git
c6b11cf..5a1be64 master -> master
```

- git pull

- + git pull은 기존 깃허브 등 서버에 커밋되어있는 상태와 현재 개인 컴퓨터와의 동기화를 실행
- + 이 때 두 사용자(깃허브와 개인)간 log history가 다른 상태라면 'git pull 리모트명 브랜치명 --allow-unrelated-histories'라는 명령어를 사용하여 병합
- + git pull 명령은 git merge와 git fetch 명령을 통합해 사용하는 명령임

Git의 주요 명령어

- git reset

- + 기존의 add된 Staged 상태인 커밋을 원래대로 Modified, 나아가 Unmodified로 만들고자 할 때 주로 쓰이는 명령어
- + 보통 옵션과 reset 사이에는 HEAD를 넣어주며 이는 제일 위 커밋부터 몇번째 커밋까지 되돌릴지 알려주기 위한 옵션임
- + 기본적으로 mixed가 되며, 이는 커밋을 Modified 상태로 만들게 됨
- + soft 옵션 : Unstaged 상태로 돌아감
- + hard 옵션 : Unmodified 상태로 돌아감

- git revert

- + commit된 사실을 되돌리도록 하는 명령어
- + reset과는 다르게 되돌리는 과정 자체를 또 다른 commit으로 만들어줌
- + reset은 log에 남지 않지만, revert는 log에 revert 되었다는 사실 자체도 기록하는 것
- + 협업 시에는 revert만 사용하는 것이 팀원들 간 커밋이 꼬이지 않도록 도와줘 유용함
- + 커밋을 제거할 수 있는 명령어도 있지만 의도치 않은 오류가 발생할 수 있으니, revert 같은 명령을 통해 되돌리는 것을 권장

Git의 주요 명령어

- git branch

- + Git이 SVN이나 Mercurial보다 더 강력한 성능을 보유하게 만들어주는 시스템
- + 주로 기존의 작업이 있어 커밋을 한 후 새로운 방법을 시도하고 싶을 때 사용함
- + 방법의 가짓수를 늘려주는 역할
- + 각 가지들은 독자적으로 운용할 수 있으며, 그와 동시에 master 가지에 다시 합치는 것도 가능
- + branch 명령은 수행하다보면 각 commit들 간의 충돌이 발생할 가능성이 항상 존재하니 버전 관리에 주의가 필요

- git merge

- + 가지로 나뉜 새로운 작업이 master 또는 받는 쪽에서 해당 가지를 다시 흡수하는 방식
- + merge를 하는 과정에서는 conflicts 등 오류가 자주 일어날 수 있기 때문에 버전을 최신 상태로 유지하는 것이 매우 중요
- + 서로 다른 가지를 뺀어나가 한 가지를 선택하기 곤란한 경우, git merge -abort 명령어를 사용하여 병합 작업을 포기하고 서로 별개의 가지 상태로 둘 수 있음



THANK YOU

GitHub Address : <https://github.com/i-Jea-i>

