2022학년도 1학기
학생 교과목 포트폴리오

학과 : 컴퓨터정보공학과
과목명 : 시스템분석설계
학번 : 20212173
이름 : 히다야

목차

1. 강의계획서
2. 소프트웨어 공학 이해
3. 깃과 깃허브

1. 강의계획서

강의계획서

교과목 기본정보

교과목명	시스템분석설계(종합설계)	학년/학기(분반)	2학년 / 1학기 (9분반)
책임교수	이동규	담당교수	강환수
과목구분	국정	이수구분	전공선택
역량	응용SW개발	운영연도	2022
이수시간(실습)	4시	학점	4

NCS정보 및 교과목표

역량	하위역량/책무(Duty)	하위역량코드	하위구성요소
응용SW엔지니어링	요구사항 확인	2001020201_14v2	현행 시스템 분석하기 (2001020201_14v2.1)
응용SW엔지니어링	요구사항 확인	2001020201_14v2	요구사항 확인하기 (2001020201_14v2.2)
교과목표	- 학과교과과정을통해습득한기술을활용한개발프로젝트를수행하여기술활용능력배양.프로젝트수행실무능력향상응용소프트웨어졸업작품개발을위한각종설계서,보고서및프로토타입을개발한다. - 최종적으로해당년생리는한간학부학생대회에개발된작품을출품하는것을목표로한다.		

교수/학습방법

A.이론강의	B.실습	C.발표	D.토론	E.팀프로젝트	F.캡스톤디자인	G.포트폴리오	H.기타
	✓	✓	✓	✓		✓	

교육장소

일반강의실		전용실습실		컴퓨터실습실		학부 교육시설		기타
유실	실명	유실	실명	유실	실명	유실	실명	

교육장소

일반강의실		전용실습실		컴퓨터실습실		외부 교육시설		기타
호실	설명	호실	설명	호실	설명	호실	설명	
3호관114호								

교재 (학습모듈)

학습모듈		교수학습지침서	
주교재		자체개발교재 사용	
부교재			
참고 교재			

평가방법

A.포트폴리오	B.문제해결시나리오	C.서술형시험	D.논술형시험	E.사례연구	F.평가자 질문	G.평가자 체크리스트
✓				✓	✓	
H.피평가자 체크리스트	I.일지/저널	J.역할연기	K.구두발표	L.직업장평가	M.기타	
			✓			

교육정보

하위구성요소 및 수행준거

순번	하위구성요소 (하위구성요소코드)	수준	수행준거

교육정보

하위구성요소 및 수행준거

순번	하위구성요소 (하위구성요소코드)	수준	수행준거
1	현행 시스템 분석하기 (2001020201_14v2.1)	4	1.1 개발하고자 하는 응용소프트웨어에 대한 이해를 높이기 위해, 현행 시스템의 적용현황을 파악함으로서 개발범위와 향후 개발될 시스템으로의 이행방향성을 분석할 수 있다. 1.2 개발하고자 하는 응용소프트웨어와 관련된 운영체제, 데이터베이스관리시스템, 미들웨어 등의 요구사항을 식별할 수 있다. 1.3 현행 시스템을 분석하여, 개발하고자 하는 응용소프트웨어가 이후 적용될 목표시스템을 명확하고 구체적으로 기술할 수 있다.
2	요구사항 확인하기 (2001020201_14v2.2)	4	2.1 소프트웨어 공학기술의 요구사항 분석 기법을 활용하여 업무 분석가가 정의한 응용소프트웨어의 요구사항을 확인할 수 있다. 2.2 업무 분석가가 분석한 요구사항에 대해 정의된 검증기준과 절차에 따라서 요구사항을 확인할 수 있다. 2.3 업무 분석가가 수집하고 분석한 요구사항이 개발하고자 하는 응용소프트웨어에 미칠 영향에 대해서 검토하고 확인할 수 있다.

지식 / 기술 / 태도

순번	하위구성요소 (하위구성요소코드)	수준	지식/기술/태도
1	현행 시스템 분석하기 (2001020201_14v2.1)	4	[지 식] 해당 산업 분야에 대한 지식 해당 플랫폼에 대한 지식 프로젝트 환경 및 특수성 플랫폼에 따른 기능 및 성능 특성 가상화 관련 지식 클라우드 컴퓨팅 관련 지식 [기 술] 내부 및 외부 환경 분석 기술 운영체제 구성 및 관리 능력 저장장치 구성 및 관리 능력 네트워크 구성 및 관리 능력 DBMS 구성 및 관리 기술 가상화 관련 기술 [태 도] 기술 관련 각종 정보 수집에 대한 적극성 용어정의 정확성과 완전성을 기하고자 하는 의지 분류 및 정리 태도

2	요구사항 확인하기 (2001020201_14v2.2)	4	[지 식] 해당 산업 분야에 대한 지식 업무 특성에 대한 이해 프로젝트 환경 및 특수성 요구공학 방법론 요구분석기법 소프트웨어 개발 방법론 타당성 분석기법 통계학 [기 술] 유즈케이스 작성 능력 UML 작성 기술 분석 자동화도구 도구 사용 능력 요구사항 관리 도구 사용 기술 리뷰 진행 기술 [태 도] 요구사항의 정확성과 완전성을 확보하려는 자세 정확한 유스케이스를 이해하고 분석하려는 자세 검증을 항목 분석을 위한 치밀한 태도 책임감 및 검증에 대한 완벽함을 추구하는 태도

주차별 학습 방법(15주차)

주차	하위구성요소	수업내용	활용자료 및 비고
1	현행 시스템 분석하기	1.1 개발하고자 하는 응용소프트웨어에 대한 이해를 높이기 위해, 현행 시스템의 적용현황을 파악함으로서 개발범위와 향후 개발될 시스템으로의 이행방향성을 분석할 수 있다.	
2	현행 시스템 분석하기	1.1 개발하고자 하는 응용소프트웨어에 대한 이해를 높이기 위해, 현행 시스템의 적용현황을 파악함으로서 개발범위와 향후 개발될 시스템으로의 이행방향성을 분석할 수 있다.	
3	현행 시스템 분석하기	1.2 개발하고자 하는 응용소프트웨어와 관련된 운영체제, 데이터베이스관리시스템, 미들웨어 등의 요구사항을 식별할 수 있다.	직무수행능력평가1차
4	현행 시스템 분석하기	1.2 개발하고자 하는 응용소프트웨어와 관련된 운영체제, 데이터베이스관리시스템, 미들웨어 등의 요구사항을 식별할 수 있다.	
5	현행 시스템 분석하기	1.3 현행 시스템을 분석하여, 개발하고자 하는 응용소프트웨어가 이후 적용될 목표시스템을 명확하고 구체적으로 기술할 수 있다.	
6	현행 시스템 분석하기	1.3 현행 시스템을 분석하여, 개발하고자 하는 응용소프트웨어가 이후 적용될 목표시스템을 명확하고 구체적으로 기술할 수 있다.	

6	현행 시스템 분석하기	1.3 현행 시스템을 분석하여, 개발하고자 하는 응용소프트웨어가 이후 적용될 목표시스템을 명확하고 구체적으로 기술할 수 있다.	
7	현행 시스템 분석하기	1.3 현행 시스템을 분석하여, 개발하고자 하는 응용소프트웨어가 이후 적용될 목표시스템을 명확하고 구체적으로 기술할 수 있다.	
8	현행 시스템 분석하기	1.1 개발하고자 하는 응용소프트웨어에 대한 이해를 높이기 위해, 현행 시스템의 적용현황을 파악함으로써 개발범위와 향후 개발될 시스템으로의 이행방향성을 분석할 수 있다. 1.2 개발하고자 하는 응용소프트웨어와 관련된 운영체제, 데이터베이스관리시스템, 미들웨어 등의 요구사항을 식별할 수 있다. 1.3 현행 시스템을 분석하여, 개발하고자 하는 응용소프트웨어가 이후 적용될 목표시스템을 명확하고 구체적으로 기술할 수 있다.	직무수행능력평가2차
9	요구사항 확인하기	2.1 소프트웨어 공학기술의 요구사항 분석 기법을 활용하여 업무 분석가가 정의한 응용소프트웨어의 요구사항을 확인할 수 있다.	
10	요구사항 확인하기	2.1 소프트웨어 공학기술의 요구사항 분석 기법을 활용하여 업무 분석가가 정의한 응용소프트웨어의 요구사항을 확인할 수 있다.	
11	요구사항 확인하기	2.2 업무 분석가가 분석한 요구사항에 대해 정의된 검증기준과 절차에 따라서 요구사항을 확인할 수 있다.	
12	요구사항 확인하기	2.2 업무 분석가가 분석한 요구사항에 대해 정의된 검증기준과 절차에 따라서 요구사항을 확인할 수 있다.	
13	요구사항 확인하기	2.3 업무 분석가가 수집하고 분석한 요구사항이 개발하고자 하는 응용소프트웨어에 미칠 영향에 대해서 검토하고 확인할 수 있다.	
14	요구사항 확인하기	2.1 소프트웨어 공학기술의 요구사항 분석 기법을 활용하여 업무 분석가가 정의한 응용소프트웨어의 요구사항을 확인할 수 있다. 2.2 업무 분석가가 분석한 요구사항에 대해 정의된 검증기준과 절차에 따라서 요구사항을 확인할 수 있다. 2.3 업무 분석가가 수집하고 분석한 요구사항이 개발하고자 하는 응용소프트웨어에 미칠 영향에 대해서 검토하고 확인할 수 있다.	항상/심화 고육
15	요구사항 확인하기	2.1 소프트웨어 공학기술의 요구사항 분석 기법을 활용하여 업무 분석가가 정의한 응용소프트웨어의 요구사항을 확인할 수 있다. 2.2 업무 분석가가 분석한 요구사항에 대해 정의된 검증기준과 절차에 따라서 요구사항을 확인할 수 있다. 2.3 업무 분석가가 수집하고 분석한 요구사항이 개발하고자 하는 응용소프트웨어에 미칠 영향에 대해서 검토하고 확인할 수 있다.	직무수행능력평가3차

교과목 구성

하위역량(수준)	하위구성요소	교과목명
요구사항 확인(4)	현행 시스템 분석하기	시스템분석설계(종합설계)
	요구사항 확인하기	

2. 소프트웨어 공학 이해

프로그램이란?

- 컴퓨터를 실행시키기 위해 차례대로 작성된 명령어 모음
  - 원시코드 source code (C언어, Java, Python)

소프트웨어란? (software)

- 컴퓨터를 작동하고 특정 작업을 실행하기 위해 사용되는 명령, 데이터 또는 프로그래밍의 집합
- SW 안에 :
  - 원시코드
  - 산출물 (자료구조, DB구조 등)
  - 각 단계마다 생산되는 문서
  - 사용자 매뉴얼

소프트웨어의 분류

관리 소프트웨어 (management)

- 정보를 받아들여 처리한 후 제공함
- 예)
  - 인터넷 뱅킹 시스템
  - 대학의 종합정보 시스템

제어 소프트웨어 (control)

- 센서를 이용하거나 기기들의 동작을 제어함
- 예)
  - 교통신호 제어
  - 의료기기 제어

임베디드 소프트웨어 (embedded)

- 기기에 내장된 형태의 소프트웨어
- 일상생활에서 사용하는 가전제품 (난방)

소프트웨어의 특징

제조가 아닌 개발

- 개발 : 새로운 아이디어, 시스템 등 좀 더 상세하게 디자인하고 발전시켜 만들어낸다.
- 개인 능력에 따라 큰 결과물 차이가 볼 수 있다.

소모가 아닌 품질 저하

- HW : 오래 사용하면 부품이 닳고, 고장 자주 발생하고, 기능도 떨어짐
- SW : 많이 사용해도 닳지 않고, 큰 고장 없고, 기능 동일 함

하드웨어 실패 곡선(욕조 곡선)의 특징

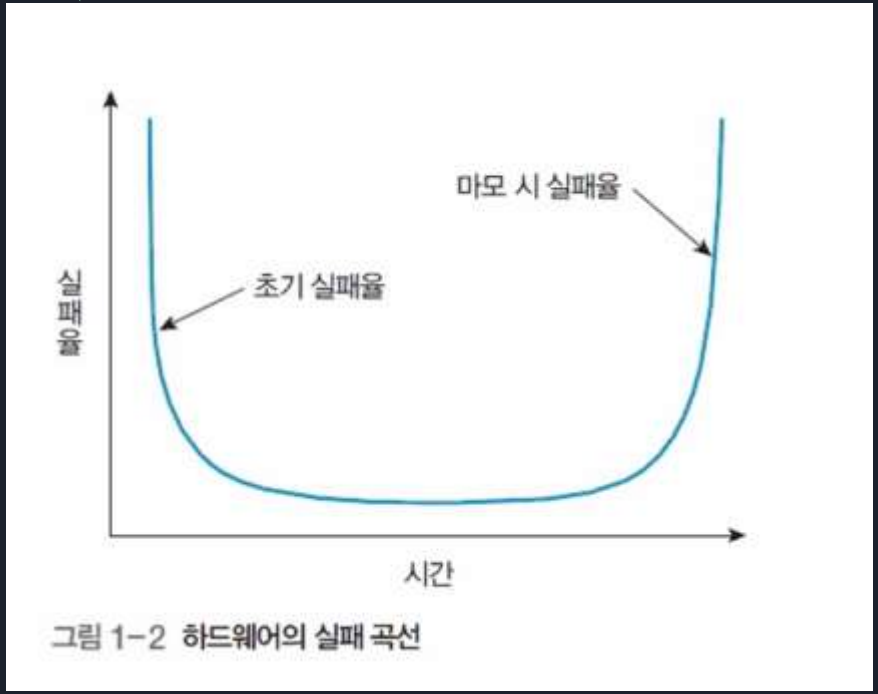
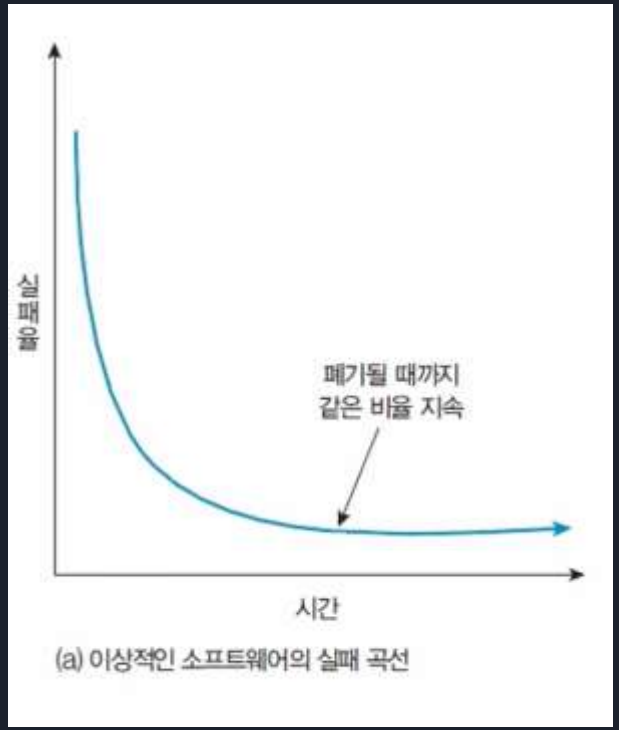


그림 1-2 하드웨어의 실패 곡선

- 초기에 실패율이 높음
  - 이것저것 문제가 많음
- 오류를 해결하고 오랜 기간 동안 사용될 수 있다
- 안정화 된다
- 주변 환경 문제가 발생 (먼지, 진동)
- 다시 고장 나고 실패율 증가

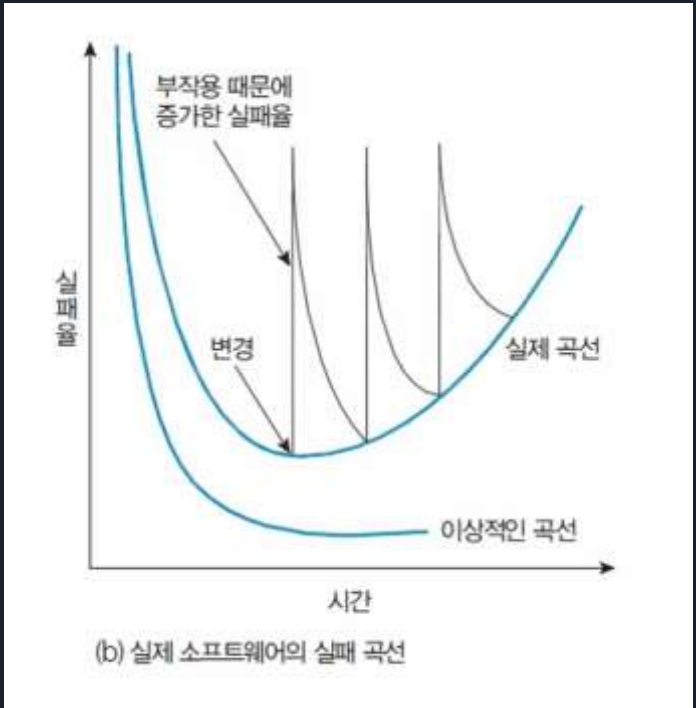
이상적인 소프트웨어 실패 곡선



(a) 이상적인 소프트웨어의 실패 곡선

- 이상적인 상황 특징
  - 개발 완료 후 변경 상황, 환경 변화 없어야 함
- 초기에 실패율이 높음
- 오류 해결하고 오랜 기간 사용할 수 있음
- 안정화 된다

실제 소프트웨어 실패 곡선



(b) 실제 소프트웨어의 실패 곡선

- 초기 실패율 높음
- 오류 해결 후 다시 실패율 낮춘다
- 변경 발생 (기능 추가 및 수정)
- 기능들이 떨어뜨린다
- 실패율 급격하게 증가

소프트웨어 개발의 어려움

- 개발 과정이 복잡하다
  - 문제가 많이 발생할 수 있다
  - 복잡함을 줄이기 위한 방법과 기술을 제시한다
- 참여 인력이 많다
  - 의사소통 경로가 많아져 의사 결정 과정도 복잡할 것이다
  - 중간에 변화도 많이 발생한다
  - 참여하는 팀을 구성, 관리하는 효율적인 방법 필요하다
- 개발 기간이 길다
  - 프로젝트 진행 상황을 파악하기 어렵다
  - 개발 비용 산정도 어렵다

소프트웨어 개발단계
SDLC (Software Development Life Cycle)

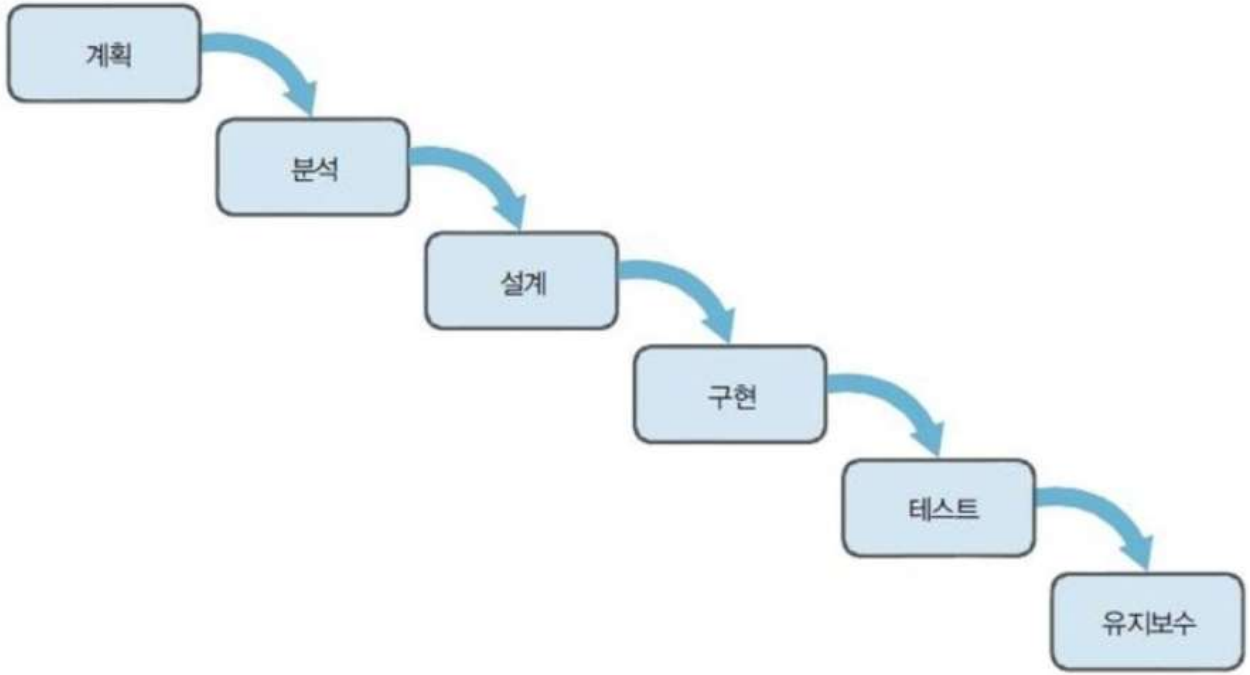


그림 1-8 **소프트웨어 개발 생명주기** SDLC: Software Development Life Cycle

SDLC

1. **계획**
2. **분석**
3. **설계**
4. 구현
5. 테스트
6. 유지보수

1 단계 : 계획 (Plan)

- 사용될 자원과 비용을 측정한다
- 인력, 장비, 시간의 소요량 등

2 단계 : 분석 (Analysis)

- 사용자의 요구사항을 구체적으로 파악한다
- 고객이 원하는 핵심요구를 짚어낸다

3 단계 : 설계(Design)

- 소프트웨어의 구조, 알고리즘, 자료 구조 등을 작성한다
- 프로그램 설계용 언어나 자연 언어 또는 다이그램으로 표현

SDLC

1. 계획
2. 분석
3. 설계
4. **구현**
5. **테스트**
6. **유지보수**

4 단계 : 구현(Implementation)

- 정해진 설계 내용을 프로그래밍 언어로 표현한다
- 조직 및 프로그래밍 도구에 의해 정의된 코딩 지침

5 단계 : 테스트(Testing)

- 실행 프로그램의 오류를 발견하고 수정한다
- 품질 표준에 도달할 때까지 오류를 추적, 수정 및 재시험한다

6 단계 : 유지보수(Maintenance)

- 시스템을 유지하고 보수하게 된다
- 테스트에서 발견하지 못한 오류를 수정
- 새로운 요구사항에 맞춰 시스템을 수정

3. 깃과 깃허브
(Git & GitHub)

버전관리란?
(Version Control)

- 파일의 수정을 시간에 따라 기록했다가 나중에 특정 시점의 버전을 다시 꺼내올 수 있는 시스템
- 개발자는 안전하게 파일을 수정 할 수 있다
- 여러 수정된 파일들을 묶고 관리한다

깃(Git)이란?

- 분산 버전 관리시스템 (Distributed Version Control System, DVCS)
- 로컬 소프트웨어
  - 파일이 컴퓨터에 저장된다
- 파일의 변경 사항을 추적한다
- 여러 사용자의 변경 사항을 모두 하나의 소스로 병합

깃허브(GitHub)란?

- 깃을 사용하는 프로젝트를 지원하는 웹 호스팅 서비스
- 깃을 위한 웹 저장소
- 저장소 = Repository
  - 원격 저장소
  - 로컬 저장소

깃 기본 용어

1. Repository
2. Branch
3. Merge
4. Commit
5. Push
6. Pull

1. **Repository**
2. **Branch**
3. **Merge**
4. **Commit**
5. Push
6. Pull

1. Repository (저장소)

- 모든 프로젝트 파일, 히스토리, 소스코드가 저장되어 있는 공간
- 분류 :
  - 로컬 저장소
  - 원격 저장소

2. Branch (브랜치)

- 독립적으로 어떤 작업을 진행할 수 있다
- 각각의 브랜치는 다른 브랜치의 영향을 받지 않다
- Master 브랜치 :
  - 저장소를 처음 만들 때, 깃은 ‘master’ 만들어 둔다

3. Merge

- 다른 브랜치의 내용을 현재 브랜치로 합친다

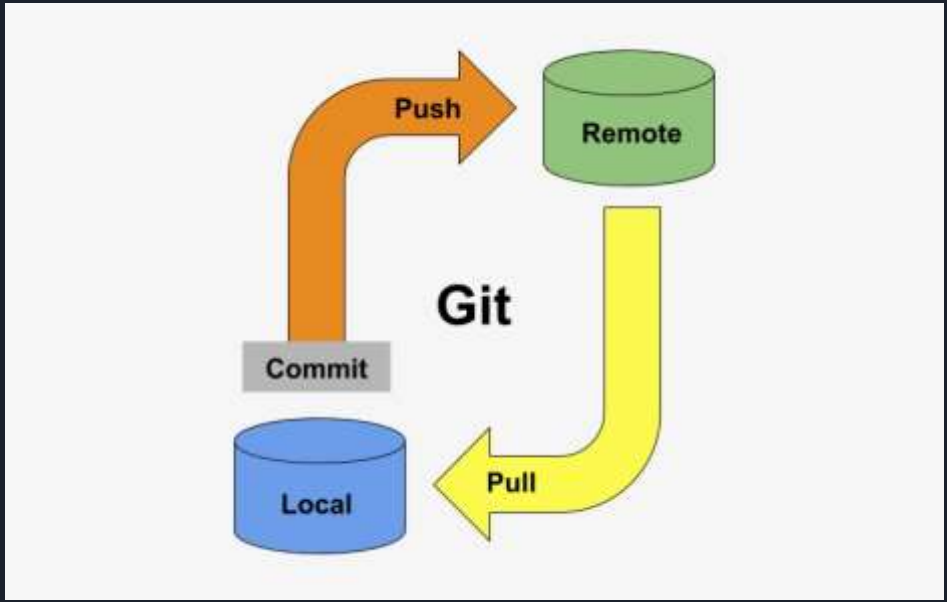
4. Commit

- 수정된 작업 상태를 잘 확인한 후 저장소에 저장한다

깃 Pull & Push

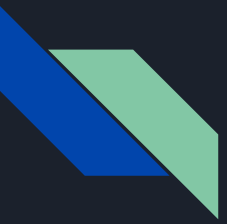
5. Push

로컬 저장소에서 원격 저장소에 업로드



6. Pull

원격 저장소에서 로컬 저장소로 가져온다



발표가 여기까지 입니다

감사합니다!