



# LG Almers

## Github 사용법과 팀규칙

VARIOUS ANALYSIS TEMPLATES FOR SALES STRATEGY



ver. 01



연세대학교 소프트웨어학부  
황선준

## GitHub의 정의 및 역할

- GitHub는 소스 코드 관리를 위한 협업 플랫폼으로, 개발자들이 효과적으로 버전 관리를 수행하고 팀 내에서 협업을 원활히 할 수 있도록 돕습니다.
- GitHub는 Git을 기반으로 만들어졌으며, 코드 공유와 협업을 위한 다양한 도구와 기능을 제공합니다.
- GitHub를 사용하면 다양한 개발 도구와 통합할 수 있어 개발 환경을 보다 체계적으로 관리할 수 있습니다.
- 원격 저장소와 로컬 저장소 간의 동기화를 쉽게 수행할 수 있으며, 팀 전체가 동일한 코드 기반에서 작업할 수 있도록 지원합니다.

## Git과의 차이점

- Git은 분산 버전 관리 도구로, 로컬 저장소에서 코드의 변경 사항을 추적하고 버전을 관리합니다.
- GitHub는 Git을 호스팅하며, 이를 통해 전 세계 어디서나 코드에 접근하고 협업할 수 있는 클라우드 기반 플랫폼을 제공합니다.
- GitHub는 웹 인터페이스, 통합 개발 도구, 그리고 자동화 기능을 제공하여 Git의 사용성을 극대화합니다.
- 또한 GitHub는 팀 구성원 간의 작업 흐름을 체계적으로 관리할 수 있는 다양한 기능을 포함하고 있습니다

## Organization이란?

- Organization은 팀 또는 회사 단위의 프로젝트 관리 기능으로, 팀 멤버 관리와 프로젝트 관리를 한 곳에서 체계적으로 수행할 수 있습니다
- 팀 멤버 간의 역할과 권한을 설정하고, 여러 프로젝트를 동시에 관리할 수 있는 환경을 제공합니다.
- 대규모 협업을 지원하며, 보안 및 접근 제어 기능을 강화한 환경을 제공합니다.
- Organization을 통해 리소스 관리와 팀 생산성을 극대화할 수 있습니다.

## Repository 관리

- 중앙 집중형 리포지토리로 팀원 간 코드 공유 및 협업을 효율적으로 관리합니다.
- Private 리포지토리와 Public 리포지토리를 선택하여 보안 요구에 따라 설정 가능합니다.
- 리포지토리별로 브랜치와 태그를 설정하여 코드의 안정성과 버전 관리를 동시에 수행합니다.
- 조직 내 다양한 프로젝트를 체계적으로 관리하고, 팀원 간의 파일 및 코드 공유를 원활하게 할 수 있습니다.

## 팀 관리

- 팀 단위로 권한과 역할(Role)을 설정하여 작업을 체계적으로 분배할 수 있습니다.
- 관리자(Admin), 유지 관리자(Maintainer), 기여자(Contributor) 등으로 역할을 분리하여 책임을 명확히 할 수 있습니다.
- 팀원 간의 작업 범위를 명확히 정의하고, 팀원별 기여도를 추적할 수 있는 기능을 제공합니다.

## 프로젝트 보드

- 칸반 스타일의 프로젝트 보드를 통해 작업 흐름을 시각적으로 관리합니다.
- 작업 상태를 "To-Do", "In-Progress", "Done"으로 나누어 팀원 간 진행 상황을 공유할 수 있습니다.
- 프로젝트 보드는 Issue와 Pull Request를 연동하여 작업과 진행 상황을 체계적으로 추적 합니다.
- 팀 간 협업이 필요한 작업이나 우선순위가 높은 작업을 손쉽게 확인할 수 있도록 도와줍니다.

## Actions

- CI/CD 파이프라인을 설정하여 코드 테스트 및 배포를 자동화합니다.
- 다양한 Actions 템플릿을 활용해 워크플로우를 간편하게 구성할 수 있습니다.
- 작업 효율성을 높이기 위해 커스텀 스크립트 작성도 지원합니다.
- 반복적인 작업을 자동화하여 개발 속도를 가속화할 수 있습니다.

## 보안 및 설정

- 브랜치 보호 규칙을 설정하여 중요한 브랜치의 변경 사항을 제한할 수 있습니다.
- 코드 스캔 및 종속성 분석 기능을 통해 보안 취약점을 사전에 탐지할 수 있습니다.
- 접근 제한과 2단계 인증(2FA)을 설정하여 계정 및 데이터 보안을 강화합니다.
- 보안 로그를 확인하고, 팀 구성원의 활동을 모니터링하여 잠재적인 위협을 미리 방지합니다.

## GitHub 기본 사용법

- a. 필수 용어 정리
  - Repository: 프로젝트의 파일과 이력(history)을 저장하는 공간.
  - Commit: 파일 변경 사항을 기록하는 저장 단위.
  - Push: 로컬 저장소에서 변경 사항을 원격 저장소로 업로드하는 작업.
  - Pull Request: 코드 변경 사항을 병합하기 위한 요청.
  - Merge: 브랜치 간 변경 사항을 통합.
  - 브랜치(Branch): 작업 흐름을 분리하여 독립적으로 개발하거나 버그를 수정할 수 있는 경로.
  - Clone: 원격 저장소를 복제하여 로컬 환경에서 작업을 시작할 수 있게 함.
  - Fork: 다른 사용자의 리포지토리를 복사하여 개인 작업 환경에서 사용.

## GitHub 기본 사용법

- b. 협업 기본 흐름
  - Fork & Clone: 리포지토리를 복제하여 개인 작업 환경을 설정.
  - Branch 생성: 새로운 기능 개발 또는 버그 수정을 위해 브랜치를 생성.
  - 작업 및 Commit: 변경 사항을 작업 단위로 저장.
  - Push & Pull Request(PR): 원격 저장소로 변경 사항을 업로드하고 병합 요청 생성.
  - Code Review: 팀원들과 변경 사항을 검토하여 품질을 보장.
  - Merge: 검토 후 Main 브랜치 또는 Develop 브랜치로 변경 사항을 통합.

## 팀규칙

- a. 브랜치 관리 규칙
  - Main Branch: 항상 배포 가능한 안정된 상태를 유지.
  - Develop Branch: 개발 중인 통합 브랜치.
  - Feature Branch: 특정 기능 개발을 위한 브랜치.
  - 브랜치 네이밍 컨벤션: feature/, bugfix/, hotfix/ 등으로 브랜치 목적을 명확히 구분.
  - 브랜치 생성 시, 관련 Issue 번호를 포함해 작업을 추적 가능하도록 함.

## 팀규칙

- b. Commit Message 규칙
  - 형식: [태그] 간결한 제목 (50자 이내)
    - 예시: [Feature] 회원가입 기능 추가
  - 주요 태그:
    - Fix: 버그 수정.
    - Feature: 새로운 기능 추가.
    - Refactor: 코드 리팩토링.
    - Chore: 사소한 변경 사항.
    - Docs: 문서 작업.
  - 커밋 메시지 본문에 작업 내용을 상세히 기술.

## 팀규칙

- c. PR & Code Review 규칙
  - 모든 Pull Request는 최소 1명 이상의 승인 필요.
  - PR 작성 시:
    - 변경 사항에 대한 간단한 설명 첨부.
    - 관련 Issue와 연결.
  - 코드 리뷰 시:
    - 명확하고 구체적인 피드백 제공.
    - 논의 후 필요하면 변경 사항 수정.
  - 팀 내 합의된 코딩 컨벤션에 따라 코드 작성.

## 팀규칙

- d. 작업 관리 규칙
  - GitHub Issues를 통해 작업 생성 및 할당.
  - Label: 우선순위 및 작업 유형 구분.
  - Milestone: 작업 기한 및 목표 설정.
  - 완료된 작업은 PR과 Issue를 연결하여 투명성 확보.
  - 주간 회의를 통해 작업 현황 점검.

## Best Practices

- 정기적인 Sync
  - 브랜치 동기화를 정기적으로 수행하여 충돌 방지.
  - 팀 내 동기화 규칙을 정하고, 개발 환경을 일치시킴.
- 자동화 활용
  - Actions를 활용하여 테스트 및 배포를 자동화하고 개발 주기를 단축.
  - 린트(lint) 및 코드 포맷팅 검사를 자동화하여 코드 품질 유지.
- 팀 커뮤니케이션
  - PR 템플릿을 작성하여 팀원 간의 소통 강화.
  - 코드 리뷰 가이드라인을 마련하여 일관된 코드 품질 유지.
  - Slack이나 MS Teams와 같은 도구를 GitHub와 연동하여 알림을 자동화.
- 보안 강화
  - 2단계 인증을 활성화하여 계정 보안 강화.
  - 브랜치 보호 규칙을 설정하여 중요한 브랜치의 무단 변경을 방지.
  - 코드 스캔과 종속성 분석을 통해 보안 취약점을 예방. 보안 키 관리 도구를 사용하여 민감한 정보 보호.

# 감사합니다

따로 github와 관련한 문의가 있으시면  
(이메일: sunjun7559012@yonsei.ac.kr)으로 연락 부탁드립니다.

연세대학교 소프트웨어학부  
황선준