

GitHub Actions 실습

■ Git

- 소프트웨어 개발 과정에서 코드의 변화를 추적하고 관리하는 버전 관리 시스템
- 기본적으로 커맨드라인 인터페이스(CLI)를 통해 사용하는 명령어 기반 도구

■ GitHub

- 소프트웨어 개발자들이 코드를 저장하고 관리하며, 다른 개발자들과 협업할 수 있도록 도와주는 웹 기반 플랫폼
- Git 기능을 웹 인터페이스를 통해 제공하는 서비스

■ GitHub Actions

- GitHub에서 제공하는 강력한 CI/CD(Continuous Integration/Continuous Delivery) 플랫폼
- 코드를 변경하고 저장소에 push하면 자동으로 빌드, 테스트, 배포 등의 과정을 수행해주는 자동화 도구

GitHub Actions 실습

■ node.js 프로젝트를 로컬에 만들기

- - PC에 git, node.js, vscode 설치
- github에 회원 가입
- - 테스트를 위한 폴더 만들기(예: test.githubAction)
- - 이 폴더를 vscode로 열기
- vscode 터미널에서 다음 명령 실행
 - npm init 실행
- package.json 편집: 배포된 예제 파일 내용 복사
- src 폴더 만들고, src/index.js 만들기: 배포된 예제 파일 복사할 것
- test 폴더 만들고, test/index.test.js 만들기: 배포된 예제 파일 복사할 것
 - 자바스크립트 테스팅 프레임워크인 jest를 이용한 코드임

GitHub Actions 실습

■ vscode에서 테스트하기

- 터미널에서 아래 명령어 실행하기
 - npm install
 - npm test
- 테스트 성공 메시지 확인하기

GitHub Actions 실습

■ 깃허브에 퍼블리시하기

- vscode 터미널에서 아래 명령어 실행
 - git config --global user.name "Your Name"
 - git config --global user.email you@example.com
- .gitignore 파일 만들기: 배포된 예제 파일을 프로젝트 폴더에 복사할 것
 - github에 올릴 때 무시해야 하는 파일을 설정함
- vscode 터미널에서 아래 명령어 실행
 - git init
- vscode 왼쪽 아이콘 중 'source code control' 아이콘 클릭
- commit 및 publish 실행
 - Message 칸에 "first commit" 입력 > "Commit" 버튼 클릭 > "Publish Branch" 버튼 클릭
 - "publish to public repository" 액션 클릭
- 깃허브에 접속해서 push 가 잘 되었는지 확인할 것

GitHub Actions 실습

■ 깃허브 액션 파일

- 코드 변경 시 자동으로 특정 작업을 수행하도록 설정할 때 작성하는 파일(YAML 형식으로 작성함)
- 위치: 깃허브리포지토리/.github/workflows/
- 구성 요소
 - 워크플로(Workflow)
 - 특정 이벤트가 발생할 때 실행되는 자동화된 프로세스. 하나 이상의 잡으로 구성됨
 - 이벤트(Event)
 - 워크플로를 트리거하는 특정 활동(push, pull_request 등)
 - 잡(Job)
 - 워크플로 내에서 실행되는 하나의 작업 단위(병렬로 실행될 수 있음)
 - 하나 이상의 스텝으로 구성됨
 - 스텝(Step)
 - 잡 내에서 실행되는 가장 작은 단위의 작업
 - 특정 명령을 실행하거나 액션을 호출함
 - 액션(Action)
 - 복잡한 작업을 간단하게 수행할 수 있는 재사용 가능한 작업 단위
 - 깃허브 마켓플레이스에서 제공하거나 커스텀으로 만들 수 있음

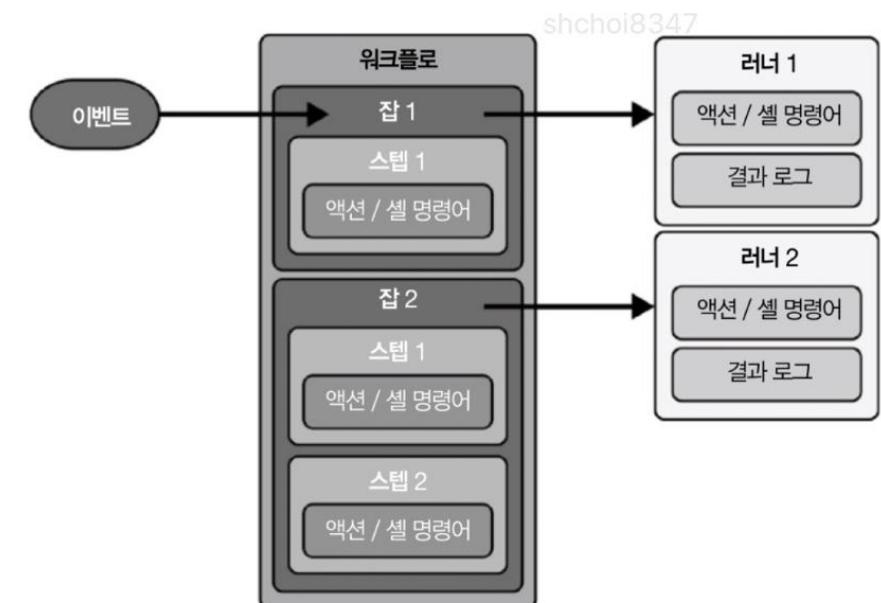


그림 2-1 깃허브 액션 구성 요소의 관계

GitHub Actions 실습

■ 깃허브 액션 만들기

- 깃허브 로그인 > 해당 리포지토리("test_githubAction") 접속 > "Actions" 메뉴 클릭
- Continuous Integration 항목 > Node.js 항목 > "Configure" 버튼 클릭
- 파일 이름 변경: ci.yml
- 파일 내용 작성: 배포된 ci.yml 내용을 복사할 것
- 우측 상단 'Commit changes...' 버튼 클릭
- 커밋 메시지("ci.yml committed") 작성 > 'Commit changes' 버튼 클릭
- 리포지토리/.github/workflows 폴더에 ci.yml 이 생기는 것을 확인
- 상단 "Actions" 메뉴 클릭 > ci.yml 항목을 클릭해서 workflow 동작 확인하기

GitHub Actions 실습

■ ci.yml 내용

- 워크플로우 기본 정보

항목	값	의미
name	Node.js CI	워크플로우의 이름입니다. GitHub Actions UI에 이 이름으로 표시됩니다.
on	push, pull_request	이 워크플로우가 언제 실행될지를 정의합니다.
branches	master	코드가 master 브랜치에 푸시되거나, master 브랜치를 대상으로 풀 리퀘스트가 들어올 때만 실행됩니다.
jobs	build	이 워크플로우 내에서 실행될 작업(Job)의 ID입니다.

GitHub Actions 실습

■ ci.yml 내용

▪ build job 상세 내역

코드 라인	역할	설명
runs-on: ubuntu-latest	실행 환경 설정	작업을 실행할 가상 머신(Virtual Machine, VM)을 지정합니다. 최신 버전의 Ubuntu Linux 환경에서 실행됩니다.
steps:	순차적 단계	build 작업을 구성하는 일련의 명령들을 순서대로 나열합니다.
uses: actions/checkout@v4	코드 체크아웃	Github Actions의 기본 액션으로, 저장소의 코드를 VM으로 복제합니다. (버전 4 사용)
name: Use Node.js 18.x	Node.js 환경 설정	Node.js 환경을 설정하는 액션입니다.
uses: actions/setup-node@v4	Node.js 설치	지정된 버전의 Node.js를 VM에 설치합니다.
node-version: 18.x	버전 지정	Node.js 버전 18.x를 사용합니다.
cache: 'npm'	종속성 캐싱	npm 패키지 캐싱을 활성화하여, 매번 새로 설치하지 않고 이전에 다운로드한 패키지를 재사용하여 빌드 속도를 높입니다.
run: npm ci	종속성 설치	npm ci (clean install) 명령을 실행하여 package-lock.json 파일에 따라 종속성을 깨끗하게 설치합니다. 일반 npm install보다 빠르고 일관적입니다.
run: npm test	테스트 실행	프로젝트의 테스트 스크립트를 실행하여 모든 테스트가 통과하는지 검증합니다.

GitHub Actions 실습

■ vscode에서 깃허브로 push 하기

- index.js 수정 및 저장하기
 - if(n < 0) { } 블록을 커멘트 처리할 것
- 터미널에서 npm test 실행 => test 두 개 중 하나 실패하는 것 확인
- 커밋 후 푸시(sync changes)
 - 좌측 "Source Control" 아이콘 선택 > 커밋 메시지 입력 > "commit" 버튼 클릭 > "Sync Changes" 버튼 클릭
- 깃 허브 > 해당 리포지토리 접속 > Actions 메뉴에서 액션 중 run npm test 부분이 실패하는 것 확인
- vscode > index.js 파일을 원래대로 수정 및 저장
 - if(n < 0) { } 블록의 커멘트를 없앨 것
- 터미널에서 npm test 실행 => test 성공 확인
- 커밋(메시지: "3rd commit") 후 푸시 > Actions에서 자동으로 test 실행 및 성공 확인