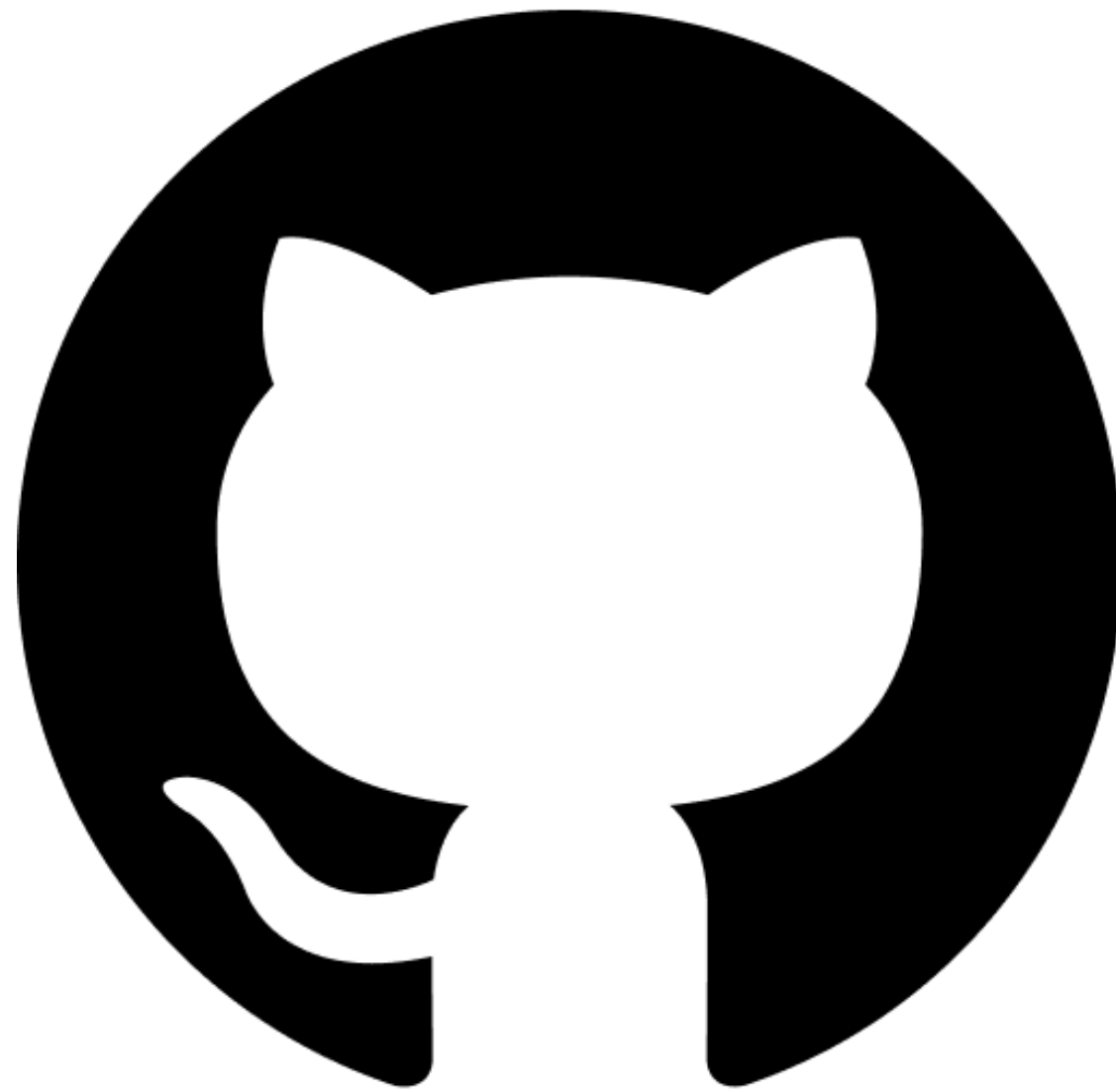
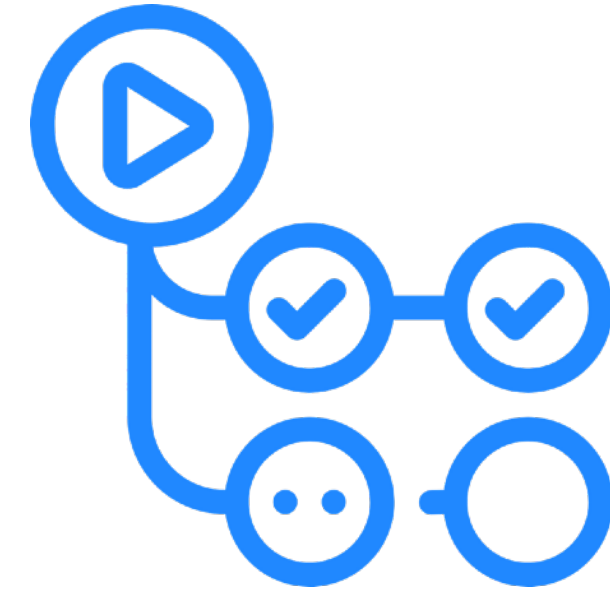
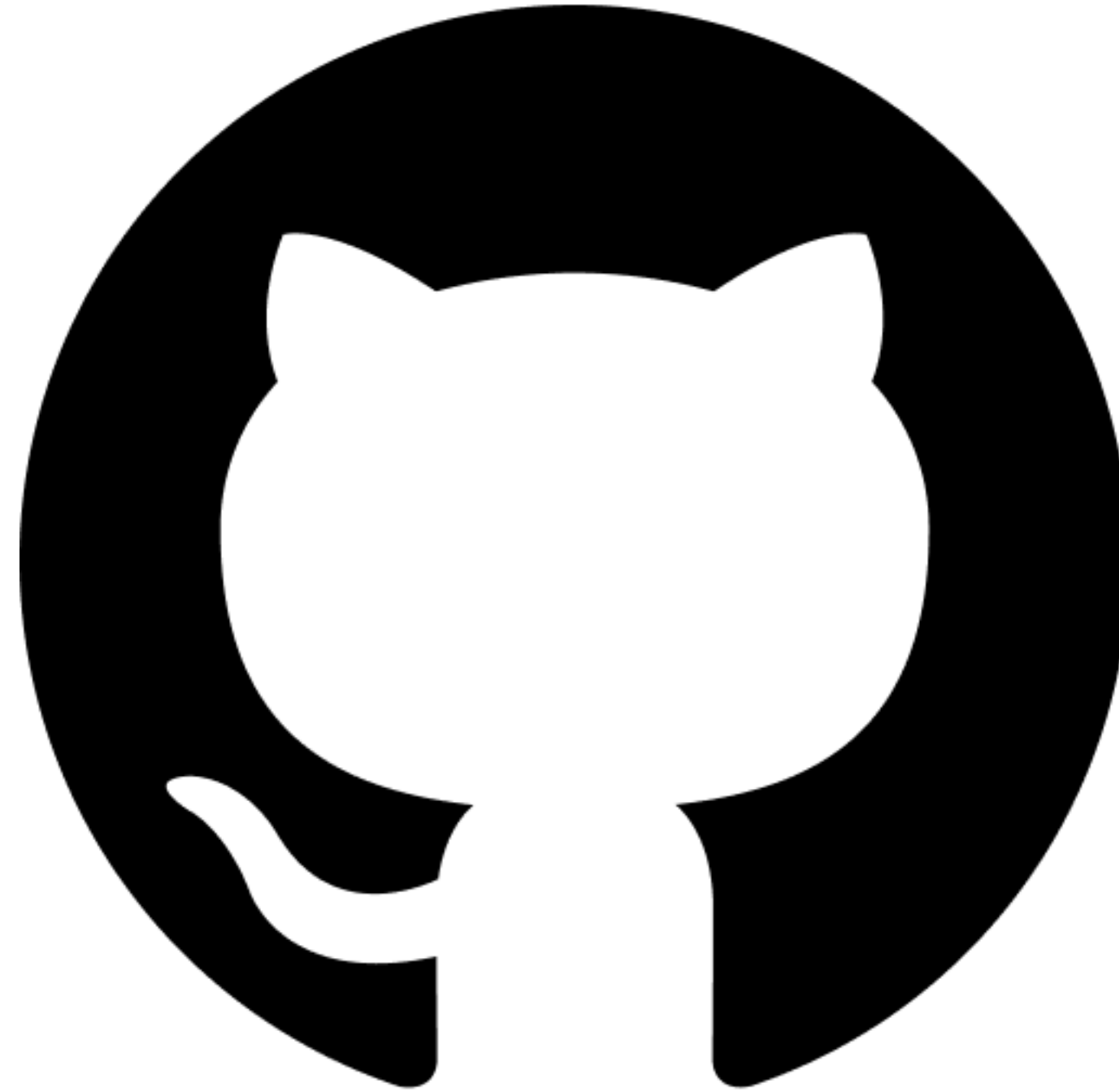


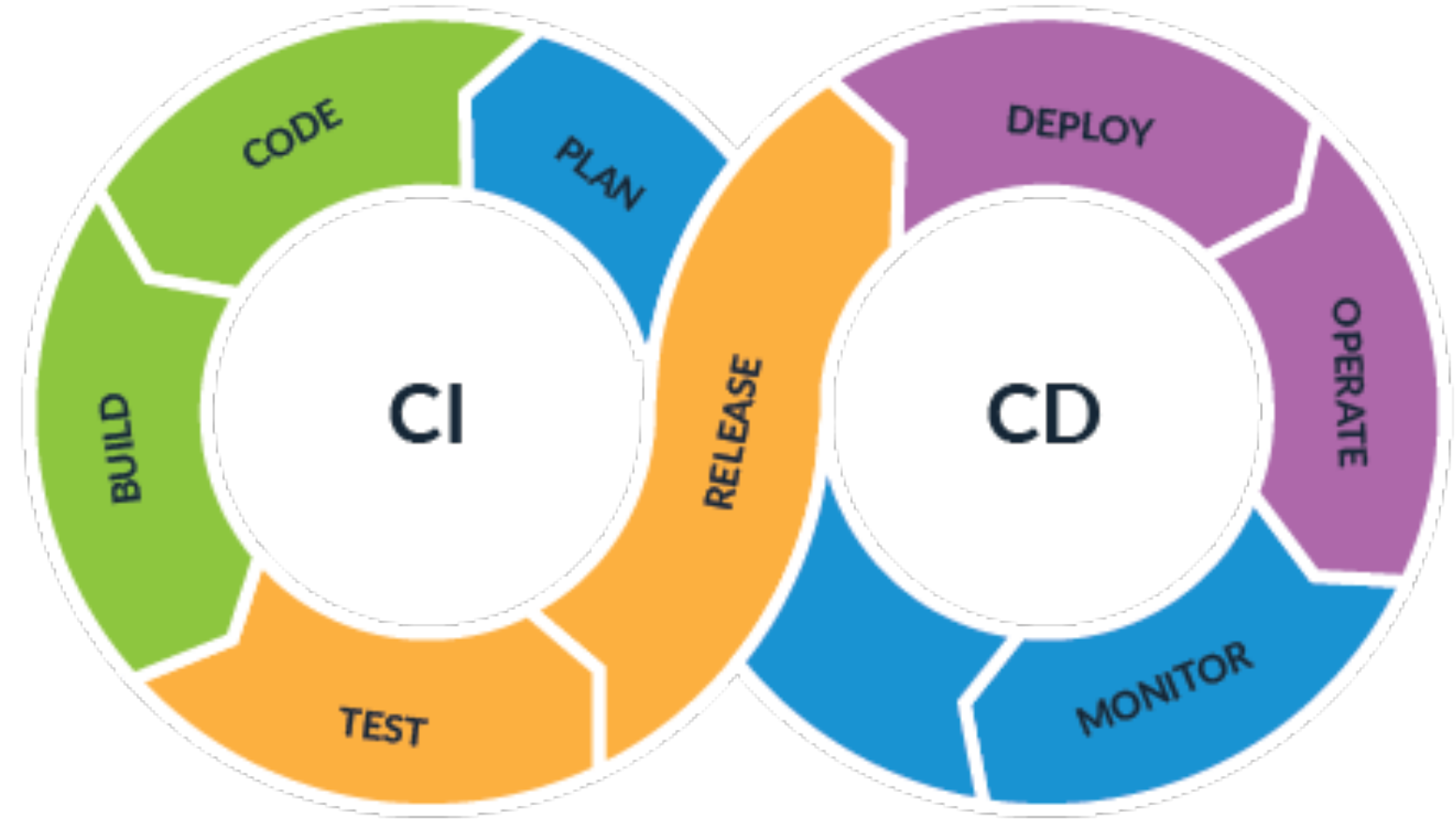
Github 100% 활용하기





Github Actions

↳ Github가 제공하는 CI / CD 



Github Actions의 기본 개념

1. runner: 깃허브 액션이 실행되는 서버
2. action: 러너에서 실행되는 작업 단위
3. event: 깃허브 액션의 실행을 발생시키는 이벤트(pr, issue, push...)
4. steps: 액션 내부의 작업 단위
5. jobs: 여러 스텝의 묶음 단위, 병렬로 실행

steps => jobs => action => runner

```
1  name: Deploy static content to Pages 액션의 이름 (optional)
2
3  on: 액션의 트리거 (mandatory)
4    push:
5      branches: ['main']
6    workflow_dispatch:
7
8  permissions: 권한 설정
9    contents: read
10   pages: write
11   id-token: write
12
13  concurrency: 병렬 실행 방지
14    group: 'pages'
15    cancel-in-progress: false
```

```
17 jobs: step의 묶음
18   build-and-deploy: job의 이름
19     environment:
20       name: github-pages
21       url: ${ steps.deployment.outputs.page_url }
22   runs-on: ubuntu-latest
23   steps: step들
24     - name: Checkout
25       uses: actions/checkout@v4 코드 체크아웃
26
27     - name: Set up Node.js
28       uses: actions/setup-node@v3 node.js 18 버전을 사용
29       with:
30         node-version: '18'
31
32     - name: Install dependencies
33       run: npm ci 의존성 설치
34
35     - name: Build
36       env:
37         VITE_GA_MEASUREMENT_ID: ${ secrets.VITE_GA_MEASUREMENT_ID } 빌드 및 환경변수 사용
38       run: npm run build
39
40     - name: Setup Pages
41       uses: actions/configure-pages@v5 Github Pages 설정
42
43     - name: Upload artifact
44       uses: actions/upload-pages-artifact@v3 dist 폴더의 결과물을 업로드
45       with:
46         path: './dist'
47
48     - name: Deploy to GitHub Pages
49       id: deployment Github Pages 배포
50       uses: actions/deploy-pages@v4
```


Github Actions Marketplaces


Actions

Automate your workflow from idea to production

Filter: All

By: All creators

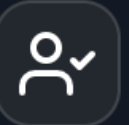
Sort: Popularity



TruffleHog OSS

Scan Github Actions with TruffleHog


Action



Metrics embed

An infographics generator with 40+ plugins and 300+ options to display stats about your GitHub account


Action



yq - portable yaml processor

create, read, update, delete, merge, validate and do more with yaml


Action



Super-Linter

Super-linter is a ready-to-run collection of linters and code analyzers, to help validate your source code


Action



Gosec Security Checker

Runs the gosec security checker


Action



Rebuild Armbian and Kernel

Support Amlogic, Rockchip and Allwinner boxes


Action



OpenCommit — improve commits with ...

Replaces lame commit messages with meaningful AI-generated messages when you push to remote


Action



Checkout

Checkout a Git repository at a particular version


Action



SSH Remote Commands

Executing remote ssh commands

Action



GitHub Pages action

GitHub Actions for GitHub Pages Deploy static files and publish your site easily. Static-Site-Generators-friendly

Action

Github Dependabot

↳ Github가 제공하는 의존성 관리



Github Dependabot

[siojeong.github.io / package.json](https://siojeong.github.io/package.json)

```
"dependencies": {  
  "buffer": "^6.0.3",  
  "gray-matter": "^4.0.3",  
  "react": "19.0.0",  
  "react-dom": "19.0.0",  
  "react-markdown": "^9.0.1",  
  "react-router-dom": "^6.26.1",  
  "react-syntax-highlighter": "^15.5.0",  
  "rehype-raw": "^7.0.0",  
  "remark-gfm": "^4.0.0"  
},
```

Version의 주. 부. 수?

ver 16.0.1

1. 주(16): 기존 버전과 호환되지 않게 변경되면 => 17.0.0
2. 부(0): 기존 버전과 호환되면서 새로운 기능이 추가되면 => 16.1.0
3. 수(1): 기존 버전과 호환되면서 버그를 수정했다면 => 16.0.2

의존성

1. dependencies: 프로젝트를 실행하는 데 꼭 필요한 패키지
=> npm install <module-name> (--save)
2. devDependencies: 개발 단계에서 필요한 패키지
=> npm install <module-name> --save-dev
=> npm install <module-name> -d
3. peerDependencies: 패키지나 라이브러리 개발 단계에서 필요한 패키지

peerDependencies?

```
{  
  "name": "my-library",  
  "version": "1.0.0",  
  "peerDependencies": {  
    "react": "^17.0.0"  
  }  
}
```

peerDependencies는 프로젝트 사용자에게 **특정 버전 설치를 강제**

package.json의 최상위 레벨에 한 번만 선언

dependencies, devDependencies에 중복 선언하면 안됨

의존성 충돌

```
{  
  "name": "my-library",  
  "version": "1.0.0",  
  "peerDependencies": {  
    "react": "^17.0.0"  
  }  
}
```

```
{  
  "name": "user-project",  
  "version": "1.0.0",  
  "dependencies": {  
    "react": "^18.0.0",  
    "my-library": "1.0.0"  
  }  
}
```

my-library는 react v 17.0.0을 peerDependencies로 강제
user-project 실행환경에서는 dependencies에
react v 18.0.0으로 설치되어 있음

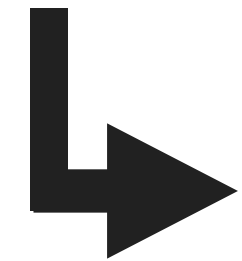
의존성 충돌 해결 방법

```
npm ERR! Could not resolve dependency:  
npm ERR! peer react@"^17.0.0" from my-library@1.0.0  
npm ERR! Conflicting peer dependency: react@18.0.0
```

```
{  
  "peerDependencies": {  
    "react": "^17.0.0 || ^18.0.0"  
  }  
}
```

1. 개발자가 peerDependencies인 17.0.0 버전에 맞추거나
2. 라이브러리의 peerDependencies에서 v17과 v18이 모두 호환되도록 수정

Github Pages



- Github가 제공하는 무료 호스팅 서비스
- Github 무료 사용자는 public 레포지토리만 사용 가능
- Github Pro 사용자는 private 레포지토리도 사용 가능
- github.io 도메인을 제공 (커스텀 도메인 사용 가능)