

버전관리 Git, Github 이해와 활용

6. 실습(2) Node.js 패키지 배포 및 활용

유재명(wesleyok@jnu.ac.kr)

1. Git 원리
2. 개발환경 만들기
3. 로컬에서 Git 사용하기
4. 로컬저장소, 원격저장소 연계 및 Github 활용
5. 실습(1) Python 패키지 배포 및 활용
6. 실습(2) Node.js 패키지 배포 및 활용
 - 1) Node.js 패키지 원리 소개
 - 2) NPM 프로젝트 생성
 - 3) 모듈 작성 및 테스트
 - 4) Github에 배포 및 활용
7. 브랜치 이해 및 활용

★ NPM(Node.js Package Manager, 이하 NPM) 개요

- NPM은 Node.js로 만들어진 패키지를 관리해 주는 프로그램으로 개발 시 필요한 라이브러리나 패키지를 쉽게 설치 및 제거하도록 도와주는 도구
 - Node.js 관련 오픈 소스 패키지는 기본적으로 NPM 저장소에서 다운로드하여 설치
 - 기본 npm저장소 URL: [https://npmjs.com/package/\[패키지명\]](https://npmjs.com/package/[패키지명])
 - 또한, 자신이 만든 패키지를 NPM레지스트리에 게시하여 다른 개발자들과 공유 가능
 - 이 경우, (<https://npmjs.com>) npm계정이 있어야 한다.
 - 우리 계획은 이러한 NPM저장소를 Github 원격저장소로 대체하여 사용하고자 함
 - Github저장소 URL: <https://npm.pkg.github.com>
 - NPM 패키지 만들고자 할 때 'package.json'파일을 생성하여 관련 정보를 기록
- 선수조건: Node.js, NPM 설치
 - Node.js 다운로드(<https://nodejs.org>)하여 설치하면, NPM도 자동 설치됨
 - Node.js 버전 확인: `node -v`
 - NPM 버전 확인: `npm -v`
- 활용도구:
 - Node.js 기반의 TypeScript, React/Vue, Javascript 등에서 활용 가능

★ NPM 패키지 배포 기본 절차

■ Npm저장소(<https://npmjs.com>) 활용시

① `npm init -y --scope=<조직/개인>`

- `-y` 옵션이 있으면 'package.json' 파일을 기본 옵션으로 자동 생성, 없으면 몇 가지 질문(상호작용)으로 'package.json' 파일 생성

② 패키지 모듈 작성/테스트

③ 패키지 배포

- www.npmjs.com 사이트 계정이 있어야 한다. (`npm login`)
- `npm publish`

■ Github 원격저장소를 활용한 npm 패키지 배포 절차

① `npm init --scope=<조직/개인>` ◀ 원격저장소생성 후 로컬저장소로 복제한 공간에서

② 패키지 작성/로컬에서 테스트

③ 로컬저장소에서 Github저장소에 패키지 배포

④ 배포한 모듈 활용

⑤ 로컬저장소와 원격저장소 동기화

2) NPM 프로젝트 생성(1)

■ Github 원격저장소를 활용한 npm 패키지 배포

① `npm init --scope=<조직/개인>` ◀ 원격저장소생성 후 로컬저장소로 복제

- Github 원격저장소 생성 : npmProject1

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

사용자 ID / npmProejct1

npmProejct1 is available.

Great repository names are short and memorable. Need inspiration? How about [fuzzy-train](#) ?

Description (optional)

NPM 패키지 배포(버전1)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Create repository

```
F:\Projects>git clone https://github.com/사용자 ID /npmProejct1.git
Cloning into 'npmProejct1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 3, done.
git clone https://github.com/UserID/Project.git
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

F:\Projects>dir npmProejct1
F 드라이브의 볼륨: Data
볼륨 일련 번호: F032-03F9

F:\Projects\npmProejct1 디렉터리

2024-08-27 오전 11:05 <DIR> .
2024-08-27 오전 11:05 <DIR> ..
2024-08-27 오전 11:05      46 README.md
                        1개 파일                46 바이트
                        2개 디렉터리 3,876,630,765,568 바이트 남음
```

■ Github 원격저장소를 활용한 npm 패키지 배포

① `npm init --scope=<조직/개인>` ◀ 로컬저장소

- `npm init --scope=<UserID>` 명령창에서 상호작용하면서 기본값 및 정보 등록

```
F:\Projects\npmProejct1>npm init --scope= 사용자 ID
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (@ 사용자 ID /npmproejct1)
version: (1.0.0)
description: nodejs utils for everyone 모듈 설명
entry point: (index.js)
test command:
git repository: https://github.com/ 사용자 ID /npmProject1.git
keywords: add,sum,hi
author: 사용자 이름
license: (ISC)
```

2) NPM 프로젝트 생성(3)

■ Github 원격저장소를 활용한 npm 패키지 배포

① `npm init --scope=<조직/개인>` ◀ 로컬저장소

- `npm init --scope=<UserID>` 기본값 및 정보 등록 결과

```
About to write to F:\Projects\npmProejct1\package.json:
{
  "name": "@사용자 ID /npmproejct1",
  "version": "1.0.0",
  "description": "nodejs utils for everyone",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/사용자 ID /npmProject1.git"
  },
  "keywords": [
    "add",
    "sum",
    "hi"
  ],
  "author": "Yoo Jae Myeong",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/사용자 ID /npmProject1/issues"
  },
  "homepage": "https://github.com/사용자 ID /npmProject1#readme"
}
```

3) 모듈 작성 및 테스트(1)

■ Github 원격저장소를 활용한 npm 패키지 배포

② 패키지 모듈 **작성**/로컬에서 테스트

```
F:\Projects\npmProejct1>notepad index.js
```

```
function add(a,b){  
  return a+b;  
}  
function sum(numbers){  
  let result = 0;  
  for (const number of numbers){  
    result += number;  
  }  
  return result;  
}  
function hi(){  
  console.log("Hi, Everyone !!!");  
}  
  
module.exports = {add, sum, hi};
```

모듈작성

```
F:\Projects\npmProejct1>dir
```

F 드라이브의 볼륨 : Data

볼륨 일련 번호 : F032-03F9

F:\Projects\npmProejct1 디렉터리

2024-08-27	오후 12:55	<DIR>	.
2024-08-27	오전 11:05	<DIR>	..
2024-08-27	오후 12:58		253 index.js
2024-08-27	오후 12:54		568 package.json
2024-08-27	오전 11:05		46 README.md
			3개 파일 867 바이트
			2개 디렉터리 3,876,632,178,688 바이트

3) 모듈 작성 및 테스트(2)

■ Github 원격저장소를 활용한 npm 패키지 배포

② 패키지 모듈 **작성**/로컬에서 테스트

```
F:\Projects\npmProejct1>npm link  
  
added 1 package, and audited 3 packages in 604ms  
  
found 0 vulnerabilities
```

npm link

※ Node.js 개발환경에서 **로컬 패키지**를 **전역적으로 연결** 또는 전역 패키지를 로컬로 연결할 때 사용

```
F:\Projects\npmProejct1>npm ls -g --depth=0  
C:\Users\master\AppData\Roaming\npm  
+-- @시용시ID/nmpmproejct1@1.0.0 -> .\F:\Projects\npmProejct1  
+-- ijavascript@5.2.1  
+-- node-red@3.1.9  
+-- npm@10.2.0
```

npm ls -g --depth=0

* npm ls : 설치된 패키지들의 트리 출력

- -g : global의 약어로, 전역적으로 설치된 패키지들을 대상으로
- --depth=0 : 출력할 패키지의 의존성 깊이 제어, '0'은 최상위만

3) 모듈 작성 및 테스트(3)

■ Github 원격저장소를 활용한 npm 패키지 배포

② 패키지 모듈 작성/로컬에서 테스트(node test.js)

```
F:\Projects>cd playground 테스트를 위해 다른 장소로 이동

F:\Projects\playground>npm ls -g --depth=0 전역적 모듈 목록 확인
C:\Users\master\AppData\Roaming\npm
+-- @사용자ID /npmproejct1@1.0.0 -> .\F:\Projects\npmProejct1
+-- ijavascript@5.2.1
+-- node-red@3.1.9
+-- npm@10.2.0
F:\Projects\playground>npm link @사용자ID /npmproejct1
added 1 package in 333ms 전역 패키지를 로컬로 연결

F:\Projects\playground>notepad test.js
F:\Projects\playground>node test.js
7
28
Hi, Everyone !!!
```

테스트파일 작성

테스트

```
const { add, sum, hi } = require("@사용자ID/npmproejct1");
console.log(add(3,4));
console.log(sum([1,2,3,4,5,6,7]));
hi();
```

4) Github에 배포 및 활용(1)

■ Github 원격저장소를 활용한 npm 패키지 배포

③ 로컬저장소에서 Github저장소에 패키지 배포

- `cd` <로컬저장소> `F:\Projects\playground>cd ..\npmProejct1`
- `npm login --scope=@사용자ID --registry=https://npm.pkg.github.com`

여기서, 저장해 놓은 Github ID/토큰 입력

```
F:\Projects\npmProejct1>npm login --scope=@사용자ID --registry=https://npm.pkg.github.com
npm notice Log in on https://npm.pkg.github.com/
Username: -mytoken
Password: Logged in to scope @사용자ID on https://npm.pkg.github.com/.
```

- `npm publish --scope=@사용자ID` #배포 `npm publish --scope=@사용자ID`
- `npm logout --scope=@사용자ID` #로그아웃 `npm logout --scope=@사용자ID`

4) Github에 배포 및 활용(2)

■ Github 원격저장소를 활용한 npm 패키지 배포

④ 배포한 모듈 활용

- 내PC의 홈디렉토리(C:\Users\[로그인계정명])에 .npmrc 파일 생성
 - .npmrc : Github 패키지 저장소에서 패키지를 가져올 인증정보 저장(자동화)

```
//npm.pkg.github.com/:_authToken=YOUR_PERSONAL_ACCESS_TOKEN  
@your-github-username:registry=https://npm.pkg.github.com
```

- 사용할 장소로 이동하여 패키지 설치

```
F:\Projects>mkdir playground2  
F:\Projects>cd playground2  
F:\Projects\playground2>npm install @사용자ID/npmproejct1  
added 1 package in 776ms  
F:\Projects\playground2>npm list | findstr npmpro  
`-- @사용자ID/npmproejct1@1.0.0
```

```
npm install @your-github-username/package-name
```

4) Github에 배포 및 활용(3)

■ Github 원격저장소를 활용한 npm 패키지 배포

④ 배포한 모듈 사용

- test2.js 파일을 만들어 모듈 테스트

```
// GitHub Packages에서 설치한 패키지를 불러옵니다.  
const {add, sum, hi } = require('@사용자ID/npmproject1');  
  
// 패키지에서 제공하는 기능 사용하기  
console.log('3+4='+add(3,4));  
console.log('1+2+3+4+5='+sum([1,2,3,4,5]));  
hi();
```

```
F:\Projects\playground2>node test2.js  
3+4=7  
1+2+3+4+5=15  
Hi, Everyone !!!
```

4) Github에 배포 및 활용(4)

■ Github 원격저장소를 활용한 npm 패키지 배포

⑤ 로컬저장소와 원격저장소 동기화 진행

- 로컬저장소로 이동
- Npm 패키지에 관련있는 파일 목록 정리
 - package.json
 - index.js
 - playground\test.js
 - playground\test2.js
- Staging Area에 추가
 - git add
- 로컬저장소에 추가
 - git commit -m '메시지'
- 동기화
 - git push origin main

4) Github에 배포 및 활용(5)

```
F:\Projects>cd npmProejct1
```

원격저장소와 동기화를 위해 로컬저장소로 이동

```
F:\Projects\npmProejct1>git status
```

작업 상태 확인

```
On branch main
```

```
Your branch is up to date with 'origin/main'.
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
index.js
```

```
package.json
```

```
playground/
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

```
F:\Projects\npmProejct1>git add .
```

관련 파일 추가

```
F:\Projects\npmProejct1>git status
```

```
On branch main
```

```
Your branch is up to date with 'origin/main'.
```

```
Changes to be committed:
```

```
(use "git restore --staged <file>..." to unstage)
```

```
new file: index.js
```

```
new file: package.json
```

```
new file: playground/test.js
```

```
new file: playground/test2.js
```

4) Github에 배포 및 활용(6)

```
F:\Projects\npmProejct1>git commit -m "NPM패키지 버전1.0.0"
[main 572ec47] NPM패키지 버전1.0.0
4 files changed, 50 insertions(+)
create mode 100644 index.js
create mode 100644 package.json
create mode 100644 playground/test.js
create mode 100644 playground/test2.js
```

로컬저장소에 저장

```
F:\Projects\npmProejct1>git log --oneline --all
572ec47 (HEAD -> main) NPM패키지 버전1.0.0
a52f824 (origin/main, origin/HEAD) Initial commit
```

```
F:\Projects\npmProejct1>git remote -v
origin https://github.com/[사용자 ID]/npmProejct1.git (fetch)
origin https://github.com/[사용자 ID]/npmProejct1.git (push)
```

```
F:\Projects\npmProejct1>git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 24 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.23 KiB | 1.23 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/[사용자 ID]/npmProejct1.git
a52f824..572ec47 main -> main
```

동기화

■ Github 원격저장소를 활용한 npm 패키지 배포 정리하면

- 지속적인 관리를 위해 Github 원격저장소를 생성한 후
- 버전 테스트 등 원활한 작업을 위해 로컬저장소를 복제
- 버전관리할 작업 내용의 절차정의 및 작성
 - Python 패키지 모듈
 - Node.js 패키지 모듈
- 작업공간(Working Area)의 관리 대상 파일 및 폴더를 Staging Area로 추가
 - `git add 파일1, 파일2, 파일3`
 - 또는 한번에 모든 파일 추가: `git add .`
- 로컬저장소에 저장
 - `git commit -m "메시지"`
- 로컬저장소와 원격저장소 동기화
 - `git push origin main`
- 위의 단계를 반복하면서 지속적으로 버전 관리를 수행

- Github Actions 워크플로우를 이용하는 방법이 있다.
- 이 방법은 CI/CD(Continuous Integration/Continuous Deployment) 앱 배포 방식으로
 - Push 이벤트가 발생할 때 자동으로 배포하는 방식
 - 간단한 절차를 정의하면
 1. package.json 파일에서 끝부분에 publishConfig 필드 추가
 - "publishConfig":{
 - "registry": <https://npm.pkg.github.com>
 - }
 2. Github Personal access tokens 획득(**packages read, write, repo 권한**)
 3. 원격저장소 Settings > Secrets and variables > Actions 에서
 - 'GITHUB_TOKEN' 변수에 token 입력하여 Github Secrets에 추가
 4. Github Actions 워크플로우 파일 생성(**자동화 절차 정의**)
 - **.github/workflows/** 폴더 생성하고, 워크플로우 YAML파일 생성
 - publish-to-github-packages.yml 파일의 주요 부분: **이벤트 트리거, 빌드, 배포** 포함
- Github Actions을 활용하면 패키지 빌드, 테스트, 배포 등 전 과정을 자동화함으로써
 - 실수를 줄이고,
 - 배포 효율성을 높일 수 있다.

감사합니다.