



9 배포

세션 시작 전 디자인파트장님의 피그마 실속 강의 ⚡

피그마 보는 법(웹 파트)

피그마 꿀팁. 개발자가 꼭 알아야할 피그마 기능. etc..

사다리타기 目 2명 발표

GitHub - yhyem/mju-likelion-session9
Contribute to yhyem/mju-likelion-session9 development by creating an account on GitHub.


<https://github.com/yhyem/mju-likelion-session9>

1 Contributor 0 Issues 0 Stars 0 Forks

빌드

`yarn build`

컴파일 된 코드를 실행할 수 있는 상태로 만드는 일

소스 코드 및 기타 리소스를 컴퓨터가 이해할 수 있는 형태로 변환하는 과정을 말합니다.

소스 코드를 실행 가능한 실행 파일, 라이브러리, 웹 애플리케이션 등으로 변환하여 애플리케이션을 구축하고 준비하는 작업이에요!

1. 개발자가 작성한 코드를 컴파일
2. 정적 파일을 번들링
3. 애플리케이션의 의존성을 관리
4. 필요한 리소스를 병합

5. 최종 실행 파일 또는 배포 가능한 패키지를 생성

이렇게 빌드의 과정을 거치면 실행할 수 있는 하나의 완성된 파일을 가지게 됩니다

배포

빌드가 완성된 실행 가능한 파일을 사용자가 접근할 수 있는 환경에 배치시키는 일

애플리케이션을 실제 환경에 전달하여 실행되도록 하는 과정입니다.

1. 빌드: 애플리케이션을 실행하기 위한 필요한 파일 및 구성 요소들을 준비하는 작업
2. 설정: 애플리케이션이 올바르게 동작하기 위해 필요한 환경을 설정하는 작업 (ex. 환경 변수, 데이터베이스 연결정보, 서버 설정등)
3. 배포 환경 준비: 서버 인스턴스를 프로비저닝(서버를 설정하고 구성하여 사용할 수 있는 상태로 만드는 과정)하거나 호스팅 환경을 설정하는 작업입니다. (ex. 애플리케이션 실행할 서버 환경 준비)
4. 애플리케이션 전달: 애플리케이션을 실행할 서버에 파일을 복사하거나, 호스팅 환경에 업로드하는 작업 (빌드한 파일이나 패키지 전달)
5. 실행 및 테스트: 애플리케이션이 올바르게 동작하는지 확인하는 작업
6. 서비스 제공: 애플리케이션을 외부에 노출하고, 도메인이나 IP 주소에 연결하는 작업

CI/CD란?

- CI란?

빌드/테스트 자동화 과정을 의미하는 용어로 개발자를 위한 자동화 프로세스

쉽게 GitHub에 특정 브랜치(master)에 새로운 커밋이 될 때마다, 해당 코드를 바탕으로 빌드하고 사용자가 미리 만들어둔 테스트 코드를 실행하여 문제가 있는지 없는지를 체크하는 과정을 자동화 한 것입니다.

- CD란?

배포 자동화 과정을 의미하는 용어로 지속적 서비스 제공(Continuous Delivery) 또는 지속적 배포(Continuous Deployment)를 의미

기존에는 빌드 후 문제가 없다고 판단되면 실제 서버든, 클라우드 환경의 서버 환경에 합쳐진 코드(빌드 된 상태)를 올리는 과정을 하며 이를 배포한다라고 합니다. 위에서 설명한 CI의 과정이 되어 있다면, 우리는 배포를 CI가 완료되는 시점에 자동으로 실행하면 됩니다. 이를 CD라고 합니다.

CI/CD 적용 전

- 1) 코드를 수정/추가 등을 하며 개발을 진행
- 2) 각자의 코딩 컨벤션에 따라 브랜치에 push 한다.
- 3) 해당 브랜치 코드에 문제가 없다고 판단되면 main branch에 병합한다.
- 4) 에러가 발생했다면 1)~3) 과정으로 에러를 고치고, 에러가 발생하지 않았다면 직접 배포를 진행한다.

CI/CD 적용 후

- 1) CI/CD 적용 전과 동일
- 2) push된 코드를 CI가 알아서 Build, Test, Lint(포맷팅)을 실행하고 결과를 알려준다.
- 3) 개발자들은 결과를 보고 에러가 난 부분을 수정한 후 main branch에 병합한다.
- 4) main branch를 감지하고 있던 CD 과정이 알아서 배포를 수행한다.

아래에서 실습할 Netlify, Vercel과 같은 경우에는 CI/CD가 자동으로 구축되어 있어서 이제 배포하고 난 뒤에 새롭게 코드가 **push** 될 경우 알아서 새롭게 빌드하고 배포가 완료됩니다!

다들 들어봤을 AWS와 같은 경우에는 기본적인 인프라 서비스만 사용하는 경우에는 CI/CD가 기본적으로 구축되어 있지 않아요!

AWS는 CI/CD를 구축하기 위한 다양한 서비스를 제공하지만, 이러한 서비스를 조합하여 자신의 프로젝트에 맞는 CI/CD 파이프라인을 구축해야합니다.

ex) AWS CodePipeline, AWS CodeDeploy, AWS CodeCommit, AWS CodeBuild, AWS CodeStar

실습해 볼 Netlify, Vercel 같은 경우에는 정적 웹 사이트 및 싱글 페이지 애플리케이션을 위한 서버 리스 배포 플랫폼으로, 정적 파일의 호스팅, 빌드 자동화, CDN 제공 등을 지원합니다.

▼ 서버리스 배포 플랫폼?

개발자가 애플리케이션 코드를 서버에 직접 배포하거나 관리하는 대신, 인프라 구성 및 서버 관리를 추상화하여 개발자가 애플리케이션 코드에 집중할 수 있도록 도와주는 플랫폼
코드를 실행하기 위해 필요한 인프라 자원 (예: 서버, 네트워크, 스토리지 등)을 자동으로 프로비저닝하고 관리

여기서 정적 웹사이트 및 싱글 페이지를 위한 애플리케이션이라고 설명하고 있지만 서버리스 배포 플랫폼을 통해 API 간의 통신을 가능하게 하여 동적인 기능을 추가할 수 있게 합니다.

이러한 플랫폼은 개발자에게 더 간편한 배포 경험을 제공하고, 전 세계적으로 분산된 CDN을 통해 빠른 속도와 성능을 제공하고 있습니다!

Netlify

서버 없이(정확히는 서버는 API랑 통신하고) 프론트엔드 스택으로만 구성된 정적 애플리케이션을 배포하는 용도로 최적화된 서비스

Develop and deploy websites and apps in record time | Netlify

Accelerate the time to deploy your websites and apps. Bring your integrations and APIs together on one powerful serverless platform. Get started for free!

 <https://www.netlify.com/>



1. Netlify와 Github Repository 연동

- Sites > Add new site > Import an existing project 클릭

Import an existing project from a Git repository

From zero to hero, three easy steps to get your site on Netlify.

1. Connect to Git provider 2. Pick a repository 3. Site settings, and deploy!

Connect to Git provider

Choose the Git provider where your site's source code is hosted. When you push to Git, we run your build tool of choice on our servers and deploy the result.

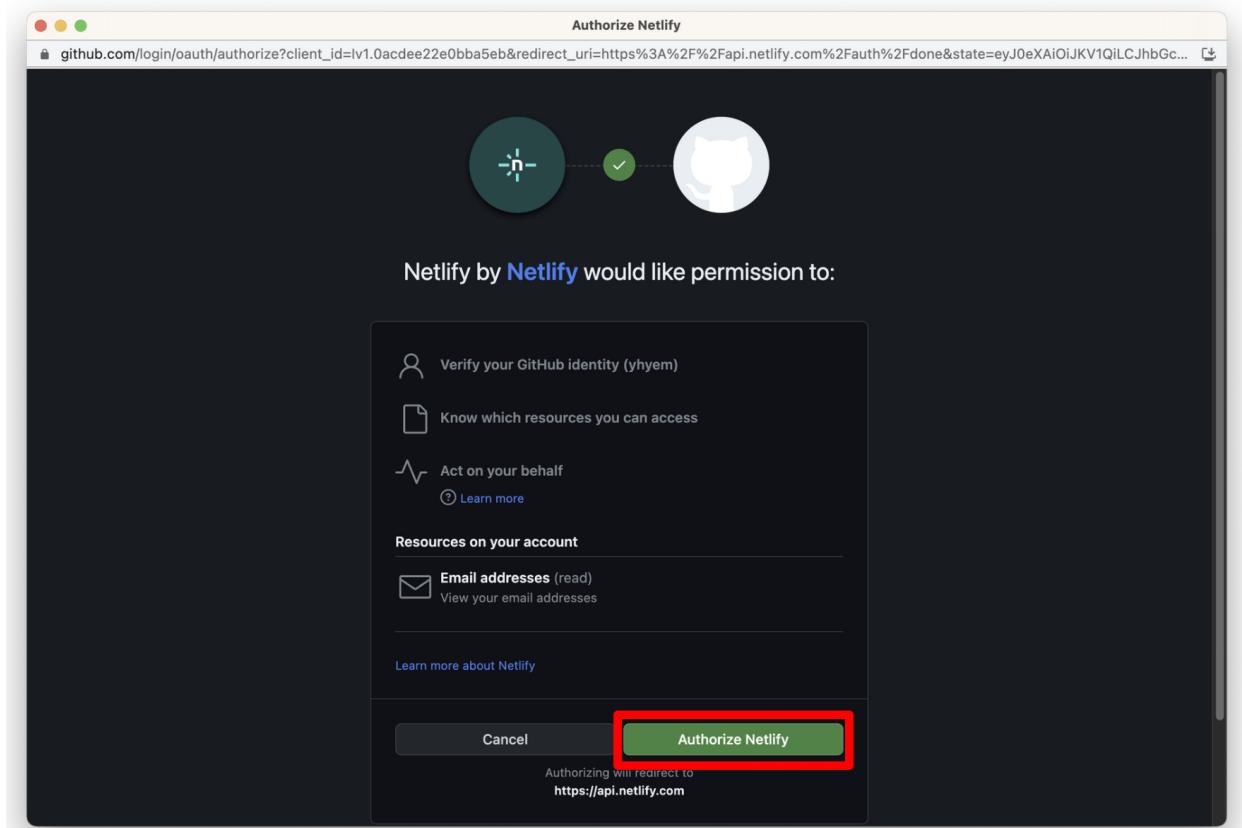
You can unlock options for self-hosted GitHub/GitLab by upgrading to the Business plan.

[GitHub](#) [GitLab](#) [Bitbucket](#) [Azure DevOps](#)

Don't have a project yet? Start from a template instead.

- GitHub 클릭

2. Github과 연동을 위해 권한을 요청을 한다.



- Authorize Netlify 클릭

3. Authorize 완료 후 배포하고자하는 레파지토리 선택

Pick a repository from GitHub

Choose the repository you want to link to your site on Netlify. When you push to Git, we run your build tool of choice on our servers and deploy the result.

The screenshot shows a list of GitHub repositories under the user 'yhyem'. The repositories are:

- yhyem/.github
- yhyem/AppExample
- yhyem/baekjoon (Private)
- yhyem/cgv-clone
- yhyem/CODELION
- yhyem/database

4. 빌드를 위한 기본 설정

Site settings for yhyem/watchapedia-clone

Get more control over how Netlify builds and deploys your site with these settings.

Owner

sb8956's team

Branch to deploy

dev

Basic build settings

If you're using a static site generator or build tool, we'll need these settings to build your site.

[Learn more in the docs ↗](#)

Base directory

Build command

yarn build

Publish directory

build

- 배포할 브랜치 선택 (일반적으로 master or main 브랜치)
- Build command : 빌드 커맨드(Build Command)가 있다면 설정
→ yarn 사용 : yarn build (빌드 명령어)

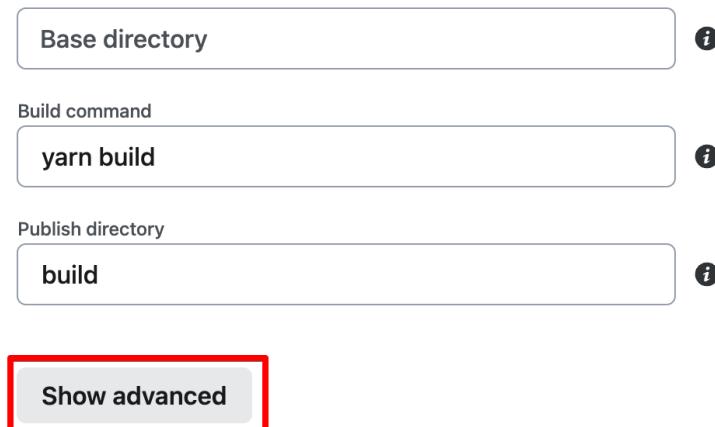
★ build를 해야지 배포가 가능한 파일로 만들 수 있는데, 이를 구현하기 위한 코드를 어떻게 할 것인지를 작성해줘야한다.

- Publish directory : 정적 디렉토리(Public Directory)를 설정
- yarn build 실행 후 생성되는 build 폴더를 publish directory로 설정

★ 배포할 파일을 정하라는 것이다. build를 하면 build 폴더가 생기기 마련인데, 그 폴더의 이름을 알려줘야한다.

CRA에서는 build라는 이름으로 폴더가 생성

5. 환경 변수 설정 위한 Show advanced 버튼 클릭



Environment variables

Define environment variables for more control and flexibility over your build.

New variable

Environment variables

Define environment variables for more control and flexibility over your build.

Key	Value	⋮	✖
REACT_APP_API_KEY	⋮	✖

- 환경 변수 설정 후 배포

if) 배포 실패 오류 뜰 경우

Production: dev@HEAD Failed

6:47 PM: Failed during stage 'building site': Build script returned ...

▼ 공식문서

경우에 따라 로컬로 실행할 때 빌드 실패를 유발하지 않는 경고 메시지로 인해 빌드가 실패할 수 있습니다. 이는 일부 라이브러리 및 빌드 도구가 CI(Continuous Integration) 환경에서 실행되고 있음을 감지할 때 경고를 다르게 처리하기 때문입니다.

다른 많은 CI 툴 및 플랫폼과 마찬가지로 Netlify는 빌드 환경 변수 CI=true를 자동화된 환경에서 실행 중임을 나타내는 규칙으로 설정합니다. 대부분의 라이브러리는 CI 변수의 존재를 사용하여 진행률 스피너 애

니메이션 또는 사용자 프롬프트 제거와 같은 동작 변경을 트리거합니다.
경우에 따라 라이브러리가 경고 메시지를 오류로 처리하여 빌드에 실패할
수도 있습니다.

일반적으로 경고에 실패하도록 선택한 라이브러리는 사용자가 경고를 유
발하는 문제를 해결하려고 할 것으로 가정합니다. 사용 사례에 적합하지
않은 경우 `site build` 명령의 시작 부분에 `CI=''`를 추가하여 CI 변수를 재정
의할 수 있습니다.

The screenshot shows the Netlify site settings interface. On the left, there's a sidebar with various options like Site overview, Deployments, Functions, Edge Functions, Integrations, Forms, Large Media, Split Testing, Analytics, and Site settings. The Site settings option is selected. In the main area, the 'General' tab is active, but the 'Continuous deployment' tab is also visible. Under 'Build settings', the 'Build command' field is set to 'CI=false yarn build'. This field is highlighted with a red box. Other settings shown include Runtime (Not set), Base directory (Not set), Publish directory (build), Deploy log visibility (Logs are public), and Build status (Active).

Build command 에 `CI=false` 추가해서 수정

6. 도메인 설정



도메인? 사이트에 액세스하는 데 사용되는 전체 이름
ex) yoursitename.netlify.app, www.yourcustomdomain.com

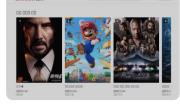
Settings for symphonious-kataifi-7bd665

[symphonious-kataifi-7bd665.netlify.app](#)

Deploys from [GitHub](#).

Owned by [sb8956's team](#).

Last update on May 31 (19 hours ago)



General

Build & deploy

Environment variables

Domain management

Domains (selected)

HTTPS

Analytics

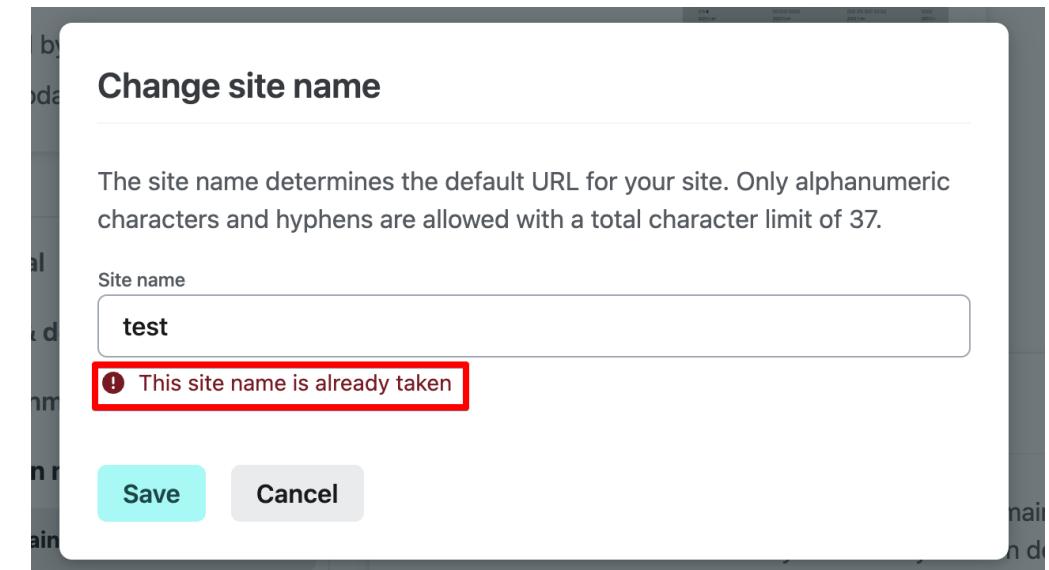
Domains

Use your own domain for your Netlify site for free

Production domains

Your site is always accessible at a [netlify.app](#) subdomain based on the site name. Custom domains allow visitors to access your site at your own domains.

[symphonious-kataifi-7bd665.netlify.app](#) Options ▾



Change site name

The site name determines the default URL for your site. Only alphanumeric characters and hyphens are allowed with a total character limit of 37.

Site name

! This site name is already taken

Save **Cancel**

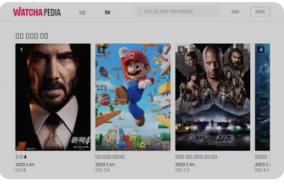
Settings for likelion-9th-session

[likelion-9th-session.netlify.app](#)

Deploys from [GitHub](#).

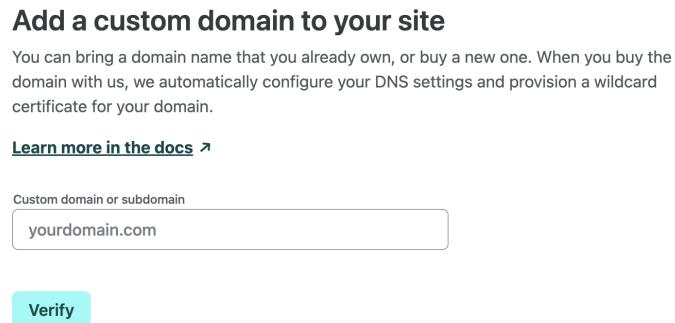
Owned by [sb8956's team](#).

Last update at 2:09 PM (a few seconds ago)



7. 개인 도메인을 설정하고 싶다면?

netlify로 배포하게 되면 기본적으로 .netlify.app 붙은채로 배포됩니다.



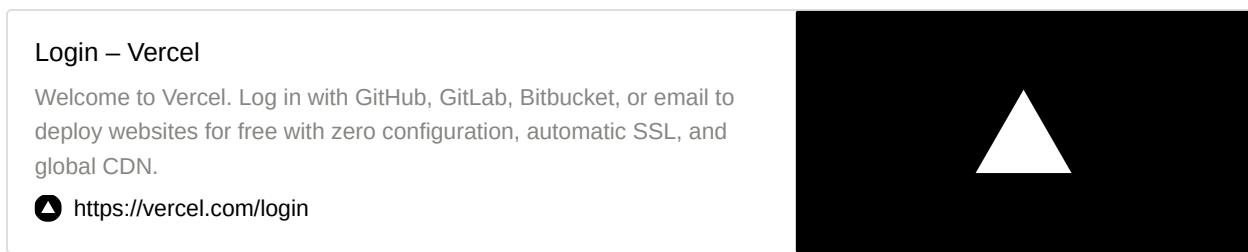
이미 소유한 도메인 이름을 가져오거나 새 도메인 이름을 구입할 수 있습니다. 도메인을 구입하면 자동으로 DNS 설정을 구성하고 도메인에 대한 와일드카드 인증서를 프로비저닝합니다.

▼ 공식문서

1. 팀의 **도메인/도메인 추가 또는 등록** 페이지로 이동하고 선택합니다 .
2. 등록하려는 도메인을 입력하고 **확인**을 선택합니다 .
3. 도메인에 이미 소유자가 있는 경우 새 도메인을 선택하고 2단계를 반복합니다.
4. 도메인을 사용할 수 있는 경우 1년 등록 가격이 표시됩니다. 팀을 위해 등록된 결제 방법을 사용하거나 결제 방법을 추가하거나 업데이트할 수 있습니다.
5. 결제를 승인하고 새 도메인을 등록하려면 **지금 도메인 등록**을 선택합니다 .

Git에서 해당 프로젝트에 push를 하게되면 Netlify에도 자동으로 프로젝트가 빌드 및 배포 가능

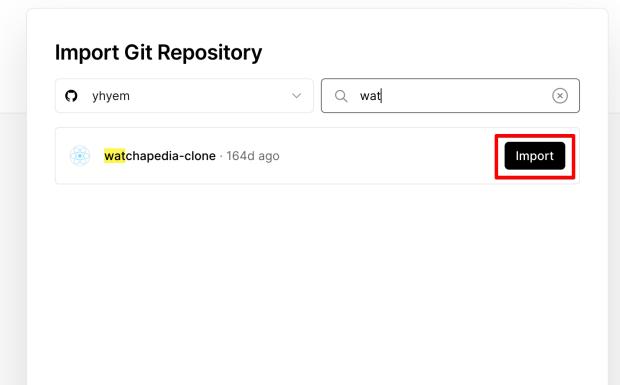
Vercel



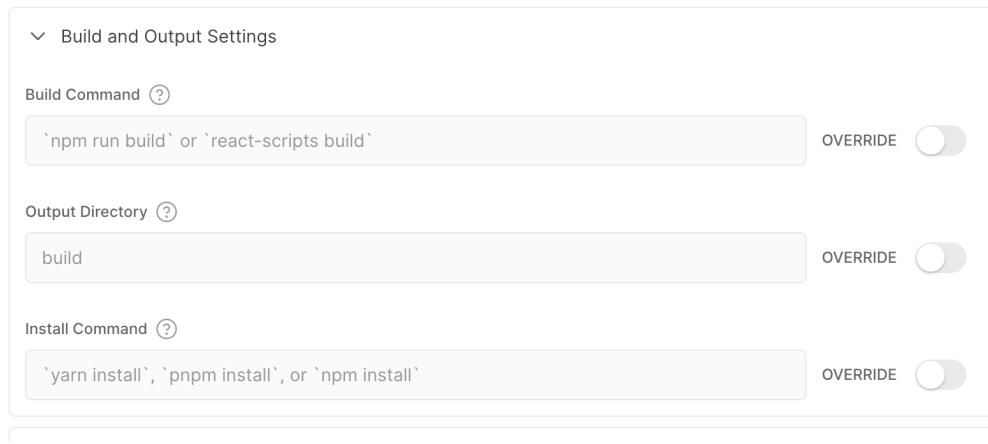
1. 본인의 **github** 계정 인증 후, 배포를 원하는 레포지토리를 선택하고 [import]

Let's build something new.

To deploy a new Project, import an existing Git Repository or get started with one of our Templates.



2. 프로젝트명, 빌드를 위한 명령어를 설정



3. 환경변수 설정

Build and Output Settings

Build Command (?)
react-scripts build

Output Directory (?)
build

Install Command (?)
yarn install

Environment Variables

Name	Value (Will Be Encrypted)	Add
EXAMPLE_NAME	I9JU23NF394R6HH	
NAME	VALUE	
REACT_APP_API_KEY	[REDACTED]	⊖

Deploy

4. Deploy

Congratulations!

You just deployed a new Project to Vercel.

[Continue to Dashboard](#)

The screenshot shows the Vercel dashboard after deployment. On the left, there's a sidebar with links like 'Watcha Pedia' and 'Recent'. The main area displays deployment statistics: '지금까지 875,296,504 개의 캐시가 불렸어요.' and a link to 'View Cache'. Below this, it says '서비스 이름과 개인정보 보호법에 따라 고객님의 개인 정보는 보호됩니다.' and provides a contact number. To the right, under 'Next Steps', there are three sections: 'Instant Previews' (Push a new branch to preview changes instantly), 'Add Domain' (Add a custom domain to your project), and 'Enable Speed Insights' (Track how users experience your site over time).

- vercel.app 이 도메인에 붙는 것을 볼 수 있음
- 바꾸고 싶으면 도메인을 추가로 구입해서 등록해야함!

Domains

These domains are assigned to your Production Deployments. Optionally, a different [Git branch](#) or a [redirection](#) to another domain can be configured for each one.

mywebsite.com Add

watchapedia-clone.vercel.app ↗ Production

Valid Configuration Assigned to master

Refresh Edit

Domain: watchapedia-clone.vercel.app

Redirect to: No Redirect

Git Branch: master

Remove Cancel Save

Git에서 해당 프로젝트에 push를 하게되면 Vercel에도 자동으로 프로젝트가 빌드 및 배포 가능

⭐️ 크로스 오버 전 필요 지식 ⭐️

1. Postman



API 개발 및 테스트를 위한 협업 도구입니다.

개발자들은 Postman을 사용하여 RESTful API를 만들고 테스트하고 문서화할 수 있습니다.

Postman은 사용자가 HTTP 요청을 보내고 응답을 검사하고, 매개 변수를 조작하고, 요청을 테스트하고 디버깅할 수 있는 다양한 기능을 제공합니다.

로컬 개발 서버는 일반적으로 인터넷을 통해 웹 브라우저에서 직접 접근할 수 없으므로, Postman과 같은 도구를 사용하여 로컬 개발 서버에 API 요청을 보내고 응답을 확인하는 방법이 필요합니다.

이를 위해서는 Postman 앱을 설치하고 실행해야 합니다. Postman 앱은 로컬 컴퓨터에 설치되어 API 요청을 만들고 테스트할 수 있는 인터페이스를 제공합니다.

Postman API Platform | Sign Up for Free

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

<https://www.postman.com/>



- 사용자가 API 요청을 만들고 테스트하기 위한 직관적인 사용자 인터페이스를 제공
- 요청 URL, 헤더, 매개 변수, 본문 데이터 등을 구성하고, 다양한 요청 메서드(GET, POST, PUT, DELETE 등)를 선택
- 요청을 전송하고 서버로부터 받은 응답을 확인하는 기능

1. Postman 설치

The Postman app

Download the app to get started with the Postman API Platform.

 Mac Intel Chip

 Mac Apple Chip

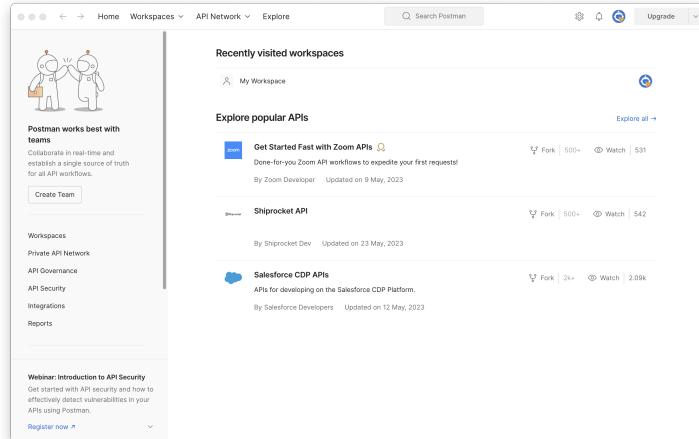
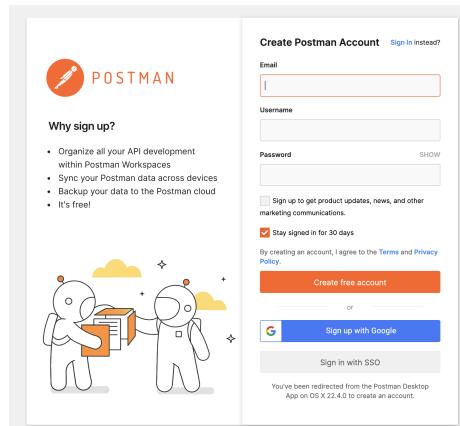
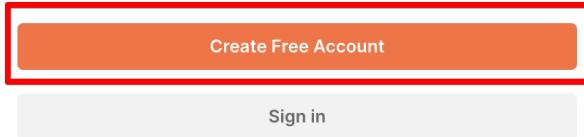
By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

[Release Notes](#) · [Product Roadmap](#)

Not your OS? Download for Windows ([x64](#)) or Linux ([x64, arm64](#))

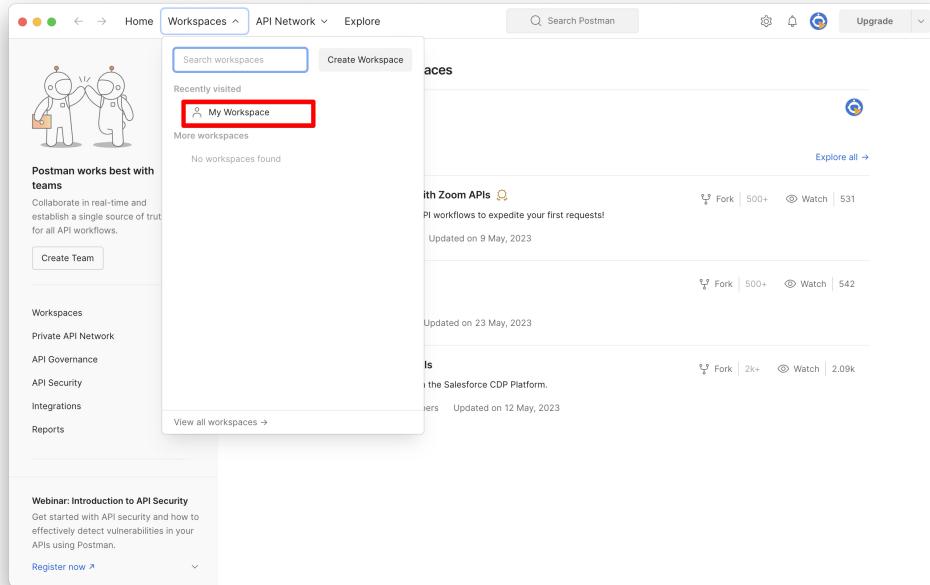
2. 회원가입

Create an account or sign in



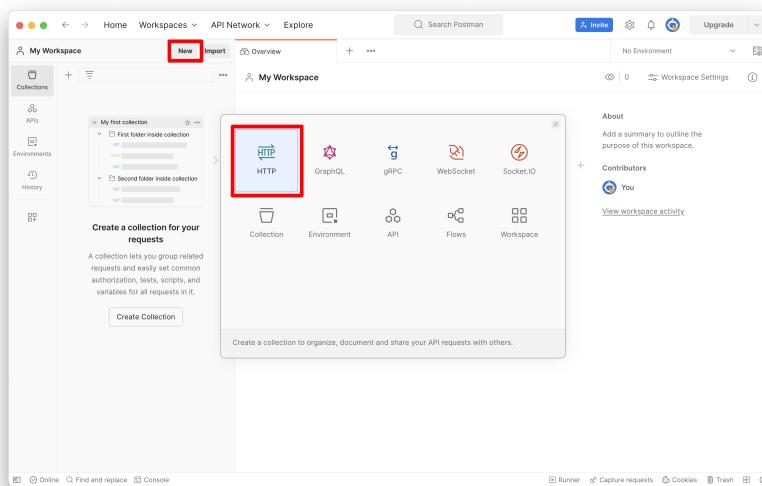
3. 포스트맨 요청 테스트

- **Workspaces 생성 → 새로 워크스페이스를 생성하거나 기본적으로 만들어져있는 My Workspace에 진입한다.**
 - Workspaces는 포스트맨의 모든 기능을 담고 있는 뷰
 - 프로젝트 용도에 따라 Personal workspace와 Team workspace로 구분
 - workspace의 단위는 목적에 따라 자유롭게 지정



• Collection 생성

- Collection은 request를 그룹화한 단위
- 모든 HTTP 요청은 Collection에 포함되어 있다.
- Collection에 속한 모든 요청에 대해 통합 테스트를 실행할 수 있다.
- 여러 request를 모아서 관리하고 공유함으로써 API 사용 설명서 역할을 하는 것

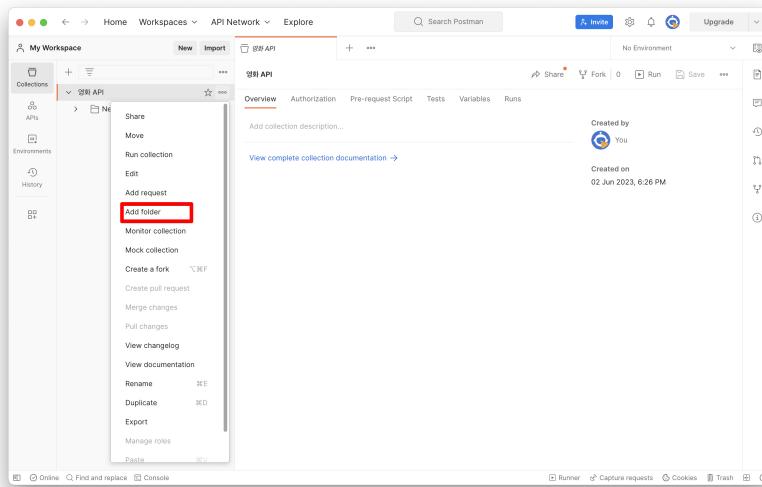


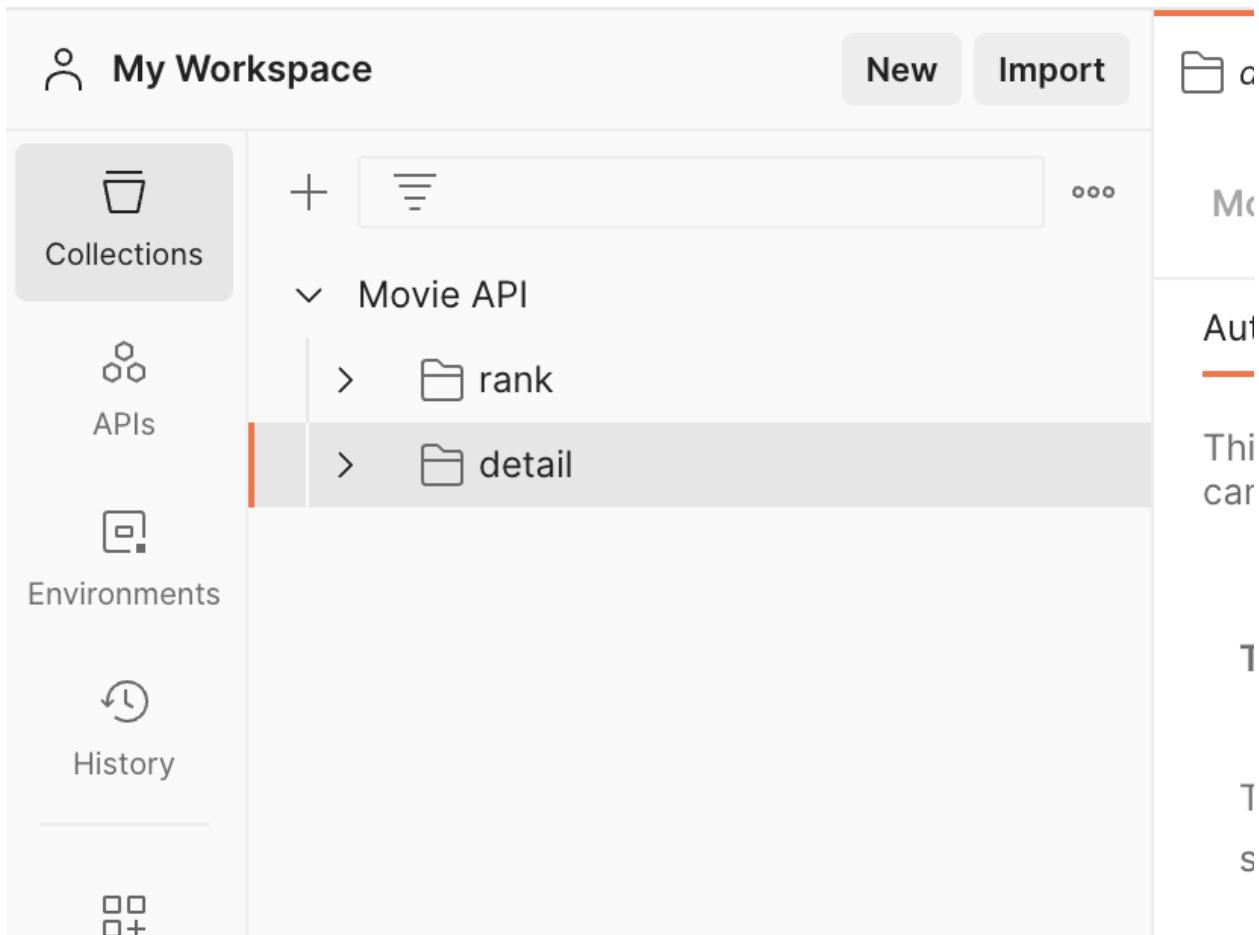
HTTP → 간단하게 요청 하나 테스트

Collection → 요청들을 체계적으로 그룹화하여 관리

- **folder** 생성

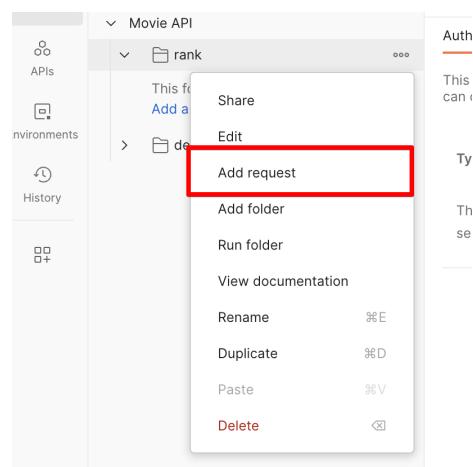
- 체계적으로 관리하기 위해 폴더별로 관리





- **Request 생성**

- 실제로 API에 대한 테스트를 진행하기 위해서는 Request를 생성



- API 문서를 보고 API 테스트 진행

Now Playing

Get a list of movies that are currently in theatres.

<https://developer.themoviedb.org/reference/movie-now-playing-list>

Popular

GET <https://api.themoviedb.org/3/movie/popular>

Get a list of movies ordered by popularity.

YOUR REQUEST HISTORY

TIME	STATUS	PATH	USER AGENT
2023. 6. 2. 오후 0...	200	/3/movie/popular	API Explorer
2023. 6. 2. 오후 0...	200	/3/movie/popular	API Explorer
2023. 6. 2. 오후 0...	200	/3/movie/popular	API Explorer

Note

This call is really just a discover call behind the scenes. If you would like to tweak any of the default filters head over and read about [discover](#).

Equivalent Discover Call

```
curl --request GET \
--url 'https://api.themoviedb.org/3/discover/movie?include_adult=false&language=en-US&page=1&region=US' \
--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJ1ZjAxNTQ2MiJ9.eyJqdGkiOiIxMjM0NTY3ODkwIiwianRpIjoiMTMzNDUwNzg4OTBhZDQyMjEwMjIwMjIwMjIwMjIwIiwidmVyc2lvbiI6IjIwMDAifQ.' \
--header 'accept: application/json'
```

LANGUAGE

- Shell
- Node
- Ruby
- PHP
- JavaScript

AUTHENTICATION

Header: eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJ1ZjAxNTQ2MiJ9.eyJqdGkiOiIxMjM0NTY3ODkwIiwianRpIjoiMTMzNDUwNzg4OTBhZDQyMjEwMjIwMjIwMjIwMjIwIiwidmVyc2lvbiI6IjIwMDAifQ.

REQUEST

```
const options = {
  method: 'GET',
  headers: {
    accept: 'application/json',
    Authorization: 'Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJ1ZjAxNTQ2MiJ9.eyJqdGkiOiIxMjM0NTY3ODkwIiwianRpIjoiMTMzNDUwNzg4OTBhZDQyMjEwMjIwMjIwMjIwMjIwIiwidmVyc2lvbiI6IjIwMDAifQ.'
  }
}

fetch('https://api.themoviedb.org/3/movie/popular?language=en-US')
  .then(response => response.json())
  .then(response => console.log(response))
  .catch(error => console.error(error))
```

Try It!

RESPONSE

Click **Try It!** to start a request and see the response here!
Or choose an example:

1. GET/POST/PUT/DELETE → HTTP 형식 확인

2. 필수 Params 확인

QUERY PARAMS

language string	en-US
page int32	1
region string	
ISO-3166-1 code	

HTTP Movie API / rank / New Request

Save Send

GET https://api.themoviedb.org/3/discover/movie

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Need help authorizing with The Movie DB API?

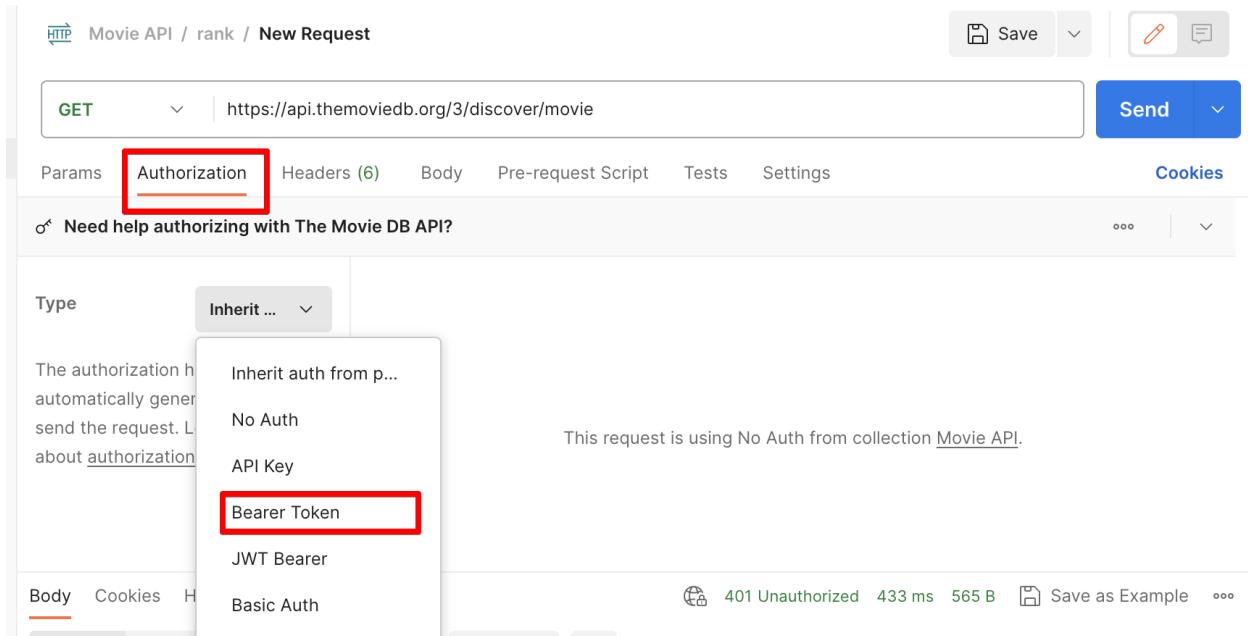
Type Inherit ...

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

Inherit auth from p...
No Auth
API Key
Bearer Token
JWT Bearer
Basic Auth

This request is using No Auth from collection [Movie API](#).

401 Unauthorized 433 ms 565 B Save as Example



3. 헤더가 필요한 경우 추가

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

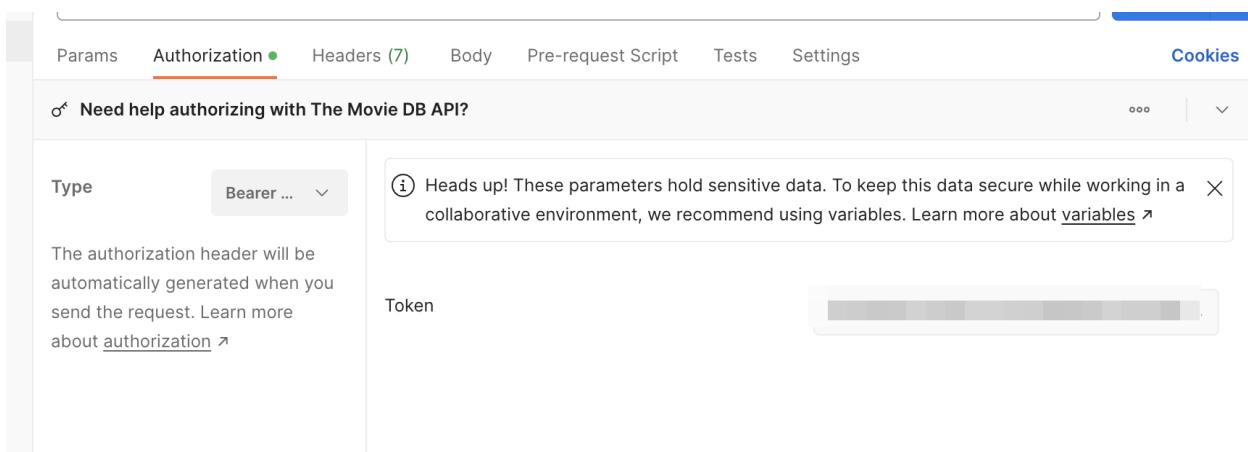
Need help authorizing with The Movie DB API?

Type Bearer ...

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

(i) Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)

Token



4. 결과 확인 (200 OK)

```

1 {
2   "page": 1,
3   "results": [
4     {
5       "adult": false,
6       "backdrop_path": "/h8gHn00zBoaefsYseUByqsmEDMY.jpg",
7       "genre_ids": [
8         28,
9         53,
10        80
11      ],
12      "id": 603602

```

이제 이 결과창에서

400 대 에러의 경우 웹쪽 잘못

500 대 에러의 경우 서버쪽 잘못

2. Swagger

현재 예비오빠한테 요청해둠 (x-rated)

3. infinite scroll



정보를 한꺼번에 가져와서 보여주기엔 정보량이 많거나 무거워서 api fetch로 받는 결과가 느릴 때, 스크롤을 통해 아주 작은 일부분만 가져와 추가로 보여주면서 사용자 경험을 높이는 기술이자 인터페이스

```

import { useState, useEffect } from "react";
import axios from "axios";
import Content from "./Content";

const InfiniteScroll = () => {
  const [data, setData] = useState([]); // 불러온 영화 데이터를 저장하는 배열
  const [isLoading, setIsLoading] = useState(false); // 데이터 로딩 중인지 여부를 나타내는 상태 변수입니다.
  const [page, setPage] = useState(1); // 현재 페이지 번호를 나타내는 상태 변수입니다.

  const fetchData = async () => {
    // 함수는 영화 데이터를 불러오는 역할
    setIsLoading(true); // 데이터 로딩
    try {
      const response = await axios.get(
        `https://api.themoviedb.org/3/movie/popular?language=ko-KR&page=${page}`,
        {
          method: "GET", //method 지정을 하지 않을 경우에 default 값은 GET이다. 사실상 필요 없는 코드

```

```

        headers: {
          accept: "application/json",
          Authorization: "Bearer " + process.env.REACT_APP_API_KEY,
        },
      }
    );
    const newData = response.data.results;

    setData((prevData) => [...prevData, ...newData]); // 새로운 데이터를 기존 데이터 배열에 추가
    setPage((prevPage) => prevPage + 1); // 페이지 번호를 업데이트
  } catch (error) {
    console.log(error);
  } finally {
    setIsLoading(false); // 로딩이 끝난 후 false
  }
};

useEffect(() => {
  fetchData(); // 초기 데이터 로딩
}, []); // 컴포넌트가 마운트되었을 때, 한 번만 실행

useEffect(() => {
  const handleScroll = () => {
    // 스크롤 이벤트를 처리하는 함수
    if (
      // 로딩 중이 아니고, 스크롤이 페이지의 맨 아래에 도달했을 때만 데이터를 다시 로드
      !isLoading &&
      window.innerHeight + window.scrollY >=
        document.documentElement.scrollHeight
    ) {
      fetchData();
      console.log("데이터 새로 로딩");
    }
  };

  window.addEventListener("scroll", handleScroll);
  return () => window.removeEventListener("scroll", handleScroll);
}, [isLoading, page]);

return (
  <div>
    {data.map((item, index) => (
      <Content key={item.id} content={item}>
        {item.title}
      </Content>
    )));
    {isLoading && <div>Loading...</div>}
  </div>
);
};

export default InfiniteScroll;

```

4. react-hook-form / yup

해당 라이브러리 같은 경우에는 로그인/회원가입과 같은 폼을 구현하여 validation을 검사해야하는 경우에 유용하게 쓸수 있는 라이브러리입니다.



React Hook Form : React를 기반으로 한 폼 관리 라이브러리



yup : 객체 스키마를 기반으로 한 JavaScript 유효성 검사 라이브러리

React Hook Form과 함께 사용하면 폼 필드의 유효성을 간단하게 검사할 수 있습니다.

```
yarn add react-hook-form  
yarn add yup  
yarn add @hookform/resolvers yup
```

각 필드에 대해 `register` 함수를 사용하여 등록하고 있습니다.

`handleSubmit` 함수를 통해 폼 제출 시 `onSubmit` 함수가 호출되며, 입력된 데 이터는 콘솔에 출력됩니다.

에러 메시지는 `errors` 객체를 통해 표시되며, 필드가 비어있을 경우 해당 에러 메시지가 렌더링됩니다.

```
//Hook.js  
import React from "react";  
import { useForm } from "react-hook-form";  
  
const Form = () => {  
  const {  
    register,  
    handleSubmit,  
    formState: { errors },  
  } = useForm();  
  
  const onSubmit = (data) => {  
    console.log(data);  
  };  
  
  return (  
    <div>
```

```

<form onSubmit={handleSubmit(onSubmit)}>
  <div>
    <label>아이디</label>
    <input
      type="text"
      {...register("id", { required: "아이디를 입력해주세요." })}
    />
    {errors.id && <h3>{errors.id.message}</h3>}
  </div>

  <div>
    <label>비밀번호</label>
    <input
      type="password"
      {...register("password", {
        min: { value: 3, message: "3이상 값을 입력해주세요." },
        required: "값을 입력해주세요.",
      })}
    />
    {errors.password && <h3>{errors.password.message}</h3>}
  </div>

  <div>
    <button type="submit">확인</button>
  </div>
</form>
</div>
);

};

export default Form;

```

각각의 폼 필드는 `register` 함수를 사용하여 등록되며, 이를 통해 폼 필드의 값을 추적하고 유효성 검사를 수행할 수 있습니다.

`register` 함수는 각 필드의 이름과 필요에 따라 유효성 검사 규칙을 설정하는 옵션을 받습니다.

▼ react-hook-form 유효성 검사 제공 기능

1. `required`: 필수 필드 여부를 검사합니다.
2. `maxLength` 및 `minLength`: 최대 길이와 최소 길이를 검사합니다.
 - a. ex) { maxLength: 10, minLength: 5 }
3. `pattern`: 정규식 패턴을 사용하여 필드 값을 검사합니다.
 - a. { pattern: /^[A-Za-z]+\$/i }
4. `validate`: 사용자 정의 유효성 검사 함수를 사용하여 필드를 검사합니다.

```

const validateAge = (value) => {
  const age = parseInt(value);
  if (isNaN(age)) {
    return "유효한 나이를 입력하세요.";
  }
  if (age < 18) {
    return "미성년자는 등록할 수 없습니다.";
  }
  return true;
};

<input {...register("age", { validate: validateAge })} />

```

5. **validate** 객체: 여러 개의 유효성 검사 함수를 객체 형태로 정의할 수 있습니다.

```

const validateRules = {
  required: "필수 필드입니다.",
  minLength: {
    value: 5,
    message: "최소 5자 이상 입력해주세요.",
  },
};

<input {...register("fieldName", { validate: validateRules })} />

```

이외에도 다양한 유효성 검사 기능이 제공

자세한 내용은 [React Hook Form 공식 문서](#) 참조

1. **yup** 라이브러리를 사용하여 유효성 검사 스키마를 정의합니다. 아이디 필드는 반드시 입력되어야 하며, 비밀번호 필드는 최소 길이를 3으로 설정합니다.
2. **useForm** 혹은 사용하여 React Hook Form의 필수 구성 요소를 가져옵니다. **yupResolver** 를 사용하여 유효성 검사 규칙을 적용합니다.
3. 폼 필드들을 생성하고 **register** 함수를 사용하여 각 필드를 등록합니다. **...register("이름")** 구문을 사용하여 필드를 등록하고 필요한 속성을 설정합니다.
4. **{...register("이름")}** 구문을 사용하면 해당 필드를 등록하고, 필요한 유

효성 검사 규칙을 적용할 수 있습니다. 에러 메시지는 `errors` 객체를 통해 표시됩니다.

```
//YupHook.js
import { useForm } from "react-hook-form";
import * as yup from "yup";
import { yupResolver } from "@hookform/resolvers/yup";

const YupForm = () => {
  const schema = yup.object().shape({
    id: yup.string().required("아이디를 입력해주세요."), // required 설정
    password: yup
      .min(3, "3이상 값을 입력해주세요.")
      .typeError("값을 입력해주세요."), // 최솟값, 최댓값 설정
  });

  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm({
    resolver: yupResolver(schema),
  });

  const check = (data) => {
    console.log(data);
  };

  return (
    <div>
      <div>
        <label>아이디</label>
        <input type="text" {...register("id")} />
        {errors.id && <h3>{errors.id.message}</h3>}
      </div>
      <div>
        <label>비밀번호</label>
        <input type="password" {...register("password")} />
        {errors.password && <h3>{errors.password.message}</h3>}
      </div>

      <div>
        <button onClick={handleSubmit(check)}>확인</button>
      </div>
    </div>
  );
};

export default YupForm;
```

위의 코드처럼 yup 스키마를 정의하고 폼 필드에 적용해야 합니다.

React Hook Form의 `register` 함수를 사용하여 필드를 등록할 때, `validate` 속성을 사용하여 해당 필드에 대한 yup 스키마를 적용할 수 있습니다.

이렇게 설정된 유효성 검사는 폼을 제출하기 전에 자동으로 실행되며, 유효하지 않은 경우 에러 메시지가 표시됩니다.

▼ yup 유효성 검사 메서드

- `string()` : 문자열 필드의 유효성을 검사합니다.
 - `required(message)` : 필수 필드임을 검사합니다.
 - `min(length, message)` : 최소 길이를 검사합니다.
 - `max(length, message)` : 최대 길이를 검사합니다.
 - `matches(regex, message)` : 정규식 패턴과 일치 여부를 검사합니다.
- `number()` : 숫자 필드의 유효성을 검사합니다.
 - `required(message)` : 필수 필드임을 검사합니다.
 - `min(value, message)` : 최소값을 검사합니다.
 - `max(value, message)` : 최대값을 검사합니다.
 - `positive(message)` : 양수 여부를 검사합니다.
 - `negative(message)` : 음수 여부를 검사합니다.
- `date()` : 날짜 필드의 유효성을 검사합니다.
 - `required(message)` : 필수 필드임을 검사합니다.
 - `min(date, message)` : 최소 날짜를 검사합니다.
 - `max(date, message)` : 최대 날짜를 검사합니다.
- `boolean()` : 불리언 필드의 유효성을 검사합니다.
 - `required(message)` : 필수 필드임을 검사합니다.
- `array()` : 배열 필드의 유효성을 검사합니다.
 - `required(message)` : 필수 필드임을 검사합니다.
 - `min(length, message)` : 최소 길이를 검사합니다.
 - `max(length, message)` : 최대 길이를 검사합니다.

- `object()`: 객체 필드의 유효성을 검사합니다.
 - `required(message)`: 필수 필드임을 검사합니다.
 - `shape(fields)`: 객체 내부의 필드 구조를 검사합니다.

이외에도 다양한 유효성 검사 메서드가 있음

`yup` 공식 문서(<https://github.com/jquense/yup>)를 참조

5. 로그인 인증 방식 - 토큰 방식(JWT) / 쿠키 방식

토큰을 이용한 로그인 인증 방식

토큰 기반 인증 방식은 API 서버와 클라이언트 애플리케이션 간의 통신에서 주로 사용됩니다.

이 방식은 RESTful API와 같이 상태를 유지하지 않는(stateless) 서비스에 적합하며, 클라이언트가 토큰을 보관하고 인증 헤더에 포함시켜 요청을 보내는 방식입니다.

토큰은 주로 JSON Web Token (JWT)과 같은 형식으로 사용됩니다.

▼ 과정

1. 사용자가 로그인 페이지에 필요한 정보를 제출합니다.
2. 서버는 사용자가 제출한 정보를 확인하고, 유효한 사용자인지 검증합니다.
3. 유효한 사용자인 경우, 서버는 인증 토큰과 리프레시 토큰을 발급합니다. 인증 토큰은 일정 시간 동안 유효하며, 리프레시 토큰은 보다 긴 유효 기간을 가지고 있습니다.
4. 클라이언트는 받은 인증 토큰을 헤더에 포함시켜 서버에 요청을 보냅니다.
5. 인증 토큰이 만료되면, 클라이언트는 리프레시 토큰을 사용하여 서버에 새로운 인증 토큰을 요청합니다.
6. 서버는 받은 리프레시 토큰의 유효성을 검사하고, 유효한 경우 새로운 인증 토큰을 발급하여 응답합니다.
7. 클라이언트는 새로 발급된 인증 토큰을 사용하여 이후의 요청을 보냅니다.

```
import { useState } from 'react';
import axios from 'axios';

function Login() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = async () => {
```

```

try {
  // 로그인 요청을 보내고 응답을 받습니다.
  const response = await axios.post('/api/login', { username, password });

  // 응답에서 토큰을 추출합니다.
  const token = response.data.token;

  // 추출한 토큰을 저장합니다.
  localStorage.setItem('token', token);

  // 로그인 성공 메시지를 출력합니다.
  console.log('로그인에 성공했습니다.');
} catch (error) {
  // 로그인 실패 메시지를 출력합니다.
  console.error('로그인에 실패했습니다.', error);
}

return (
  <div>
    <input
      type="text"
      placeholder="사용자 이름"
      value={username}
      onChange={(e) => setUsername(e.target.value)}
    />
    <input
      type="password"
      placeholder="비밀번호"
      value={password}
      onChange={(e) => setPassword(e.target.value)}
    />
    <button onClick={handleLogin}>로그인</button>
  </div>
);
}

export default Login;

```

쿠키를 이용한 로그인 인증 방식

주로 웹 애플리케이션에서 사용되며, 쿠키는 브라우저에서 관리됩니다.

서버는 클라이언트에게 쿠키를 설정하여 클라이언트가 요청을 보낼 때마다 쿠키를 함께 보내도록 유도합니다.

서버는 쿠키를 통해 세션을 유지하고 인증 상태를 확인합니다.

이 방식은 세션 관리와 보안 측면에서 서버 측에서 더 많은 관리가 필요합니다.

▼ 과정

1. 사용자가 로그인 페이지에 사용자 이름과 비밀번호를 입력하고 제출합니다.
2. 서버는 사용자가 제출한 정보를 검증하여 유효한 사용자인지 확인합니다.
3. 유효한 사용자인 경우, 서버는 로그인 세션을 생성하고 고유한 세션 ID를 생성합니다.
4. 서버는 쿠키에 세션 ID를 저장하고, 응답의 헤더에 "Set-Cookie" 헤더 필드를 추가하여 클라이언트에게 쿠키를 전송합니다. 예를 들어, "Set-Cookie: sessionID=abcdef1234567890"과 같이 전송될 수 있습니다.
5. 클라이언트는 받은 쿠키를 저장합니다. 일반적으로 웹 브라우저는 쿠키를 자동으로 저장하고 관리합니다.
6. 이후, 클라이언트가 추가 요청을 보낼 때마다 저장된 쿠키를 요청의 헤더에 포함시켜 전송합니다. 예를 들어, "Cookie: sessionID=abcdef1234567890"과 같이 요청 헤더에 포함됩니다.
7. 서버는 클라이언트가 보낸 쿠키를 확인하여 세션 ID를 추출하고, 해당 세션 ID와 연결된 사용자를 인증합니다. 이를 통해 로그인 상태를 유지하고 사용자의 요청을 처리합니다.

```
yarn add react-cookie
```

```
// index.js

import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import { BrowserRouter } from 'react-router-dom';
import { CookiesProvider } from 'react-cookie';

ReactDOM.render(
  <React.StrictMode>
    <CookiesProvider> // 쿠키 provider
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </CookiesProvider> // 쿠키 provider
  </React.StrictMode>,
  document.getElementById('root')
);
```

```
import { useState } from "react";
import axios from "axios";
import { useCookies } from "react-cookie"; // useCookies import

function Login() {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [cookies, setCookie] = useCookies(["id"]); // 쿠키 흐
```

```

const handleLogin = async () => {
  try {
    // 로그인 요청을 보내고 응답을 받습니다.
    const response = await axios.post("/api/login", { username, password });

    // 응답에서 쿠키를 추출합니다.
    const cookies = response.data.token;
    // 쿠키에 토큰 저장
    setCookie("id", cookies);

    // 로그인 성공 메시지를 출력합니다.
    console.log("로그인에 성공했습니다.");
  } catch (error) {
    // 로그인 실패 메시지를 출력합니다.
    console.error("로그인에 실패했습니다.", error);
  }
};

return (
  <div>
    <input
      type="text"
      placeholder="사용자 이름"
      value={username}
      onChange={(e) => setUsername(e.target.value)}
    />
    <input
      type="password"
      placeholder="비밀번호"
      value={password}
      onChange={(e) => setPassword(e.target.value)}
    />
    <button onClick={handleLogin}>로그인</button>
  </div>
);
}

export default Login;

```

백엔드가 애플리케이션의 요구사항과 보안 상의 고려사항을 고려 선택.

일반적으로 토큰 방식이 더 유연하고 확장성이 높아서 API 서버와 클라이언트 애플리케이션 간의 통신에 주로 사용됩니다.

→ 섭팟장님과 논의 해본 결과 저희는 크로스오버 세션에서 토큰 인증 방식으로 구현할 예정입니다



크로스 오버 조 구성

1조	경규혁 김정민
2조	김동영 김효리
3조	김아영 오현의
4조	안지유 이진혁

[양조장] 프로젝트 개발

▼ 초기세팅은 어떤걸 해야할까?

- repository Naming
- Naming Rule
- Git Convetion
- ESlint, Prettier
- Collaborator 등록
- develop 브랜치 생성 → default 변경
- 필수 라이브러리 설치 (Tech Stack)
 - React
 - styled-components
 - styled-reset
 - react-router-dom
 -
- eslint, prettier 설정
- global style 설정 (font/color)

▼ 9주차 과제

배포한 netlify, vercel 링크 제출

크로스오버 팀 조 레파지토리 생성 후 같은 팀원 콜라보레이터 설정해서 초기 세팅 후 레파지토리 링크 제출

