

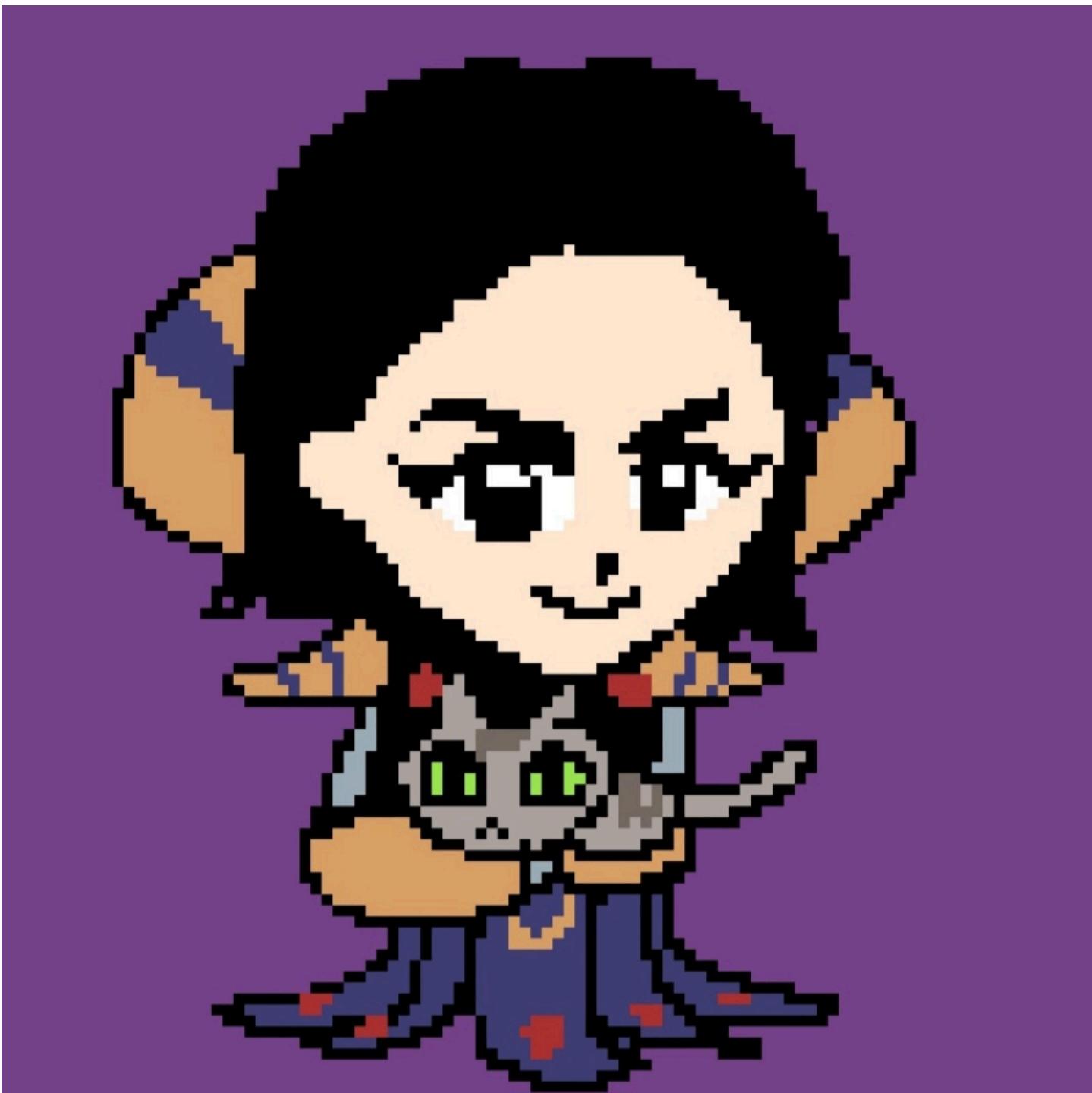
Next-Auth With Cognito

#AWS_한국_사용자_모임
#프론트엔드



2024.01.03
GitHub @lmjourney

소개



최 지연

기획자 시절, 클라이언트의 '팬시 하게, 구현해 줘'에 지쳐 과감하게 퇴사. 폐관 수련 후, 개발자로 전직

⭐멋진 회사 활발하게 찾는 중 (많은부..)⭐

발표 목적 - 샷건..아니 눈물 흘렸던 순간들 공유 및 혹시 적용하게 되면 덜 고생 하시라는 마음에서 용기를 내 발표

<h1>목차</h1>

레파지토리 링크

핵심 기술 스택 소개

핵심 구현

트러블 슈팅



2024.01.03

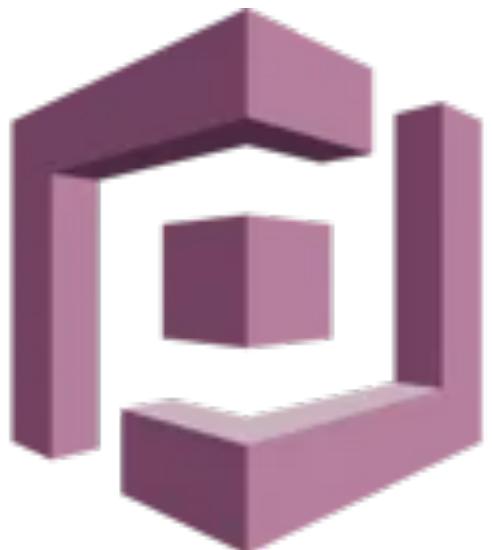
GitHub @lmjourney

<h2>레파지토리 링크</h2>



<h3>핵심 기술 스택 소개</h3>

Cognito



AWS Cognito

웹 및 모바일 앱에 대한 **인증과 권한 부여 그리고 사용자 관리**를 제공
기존의 아이디, 패스워드 방식 이외에도 Facebook, Amazon, Google, Apple과 같은
여러 회사의 소셜 로그인 기능을 제공하는 서비스

사용자 풀(user pool)과 **자격 증명 풀(identity pool)**으로 구성
이 둘을 조합하거나 또는 각각 별개의 형태로 사용

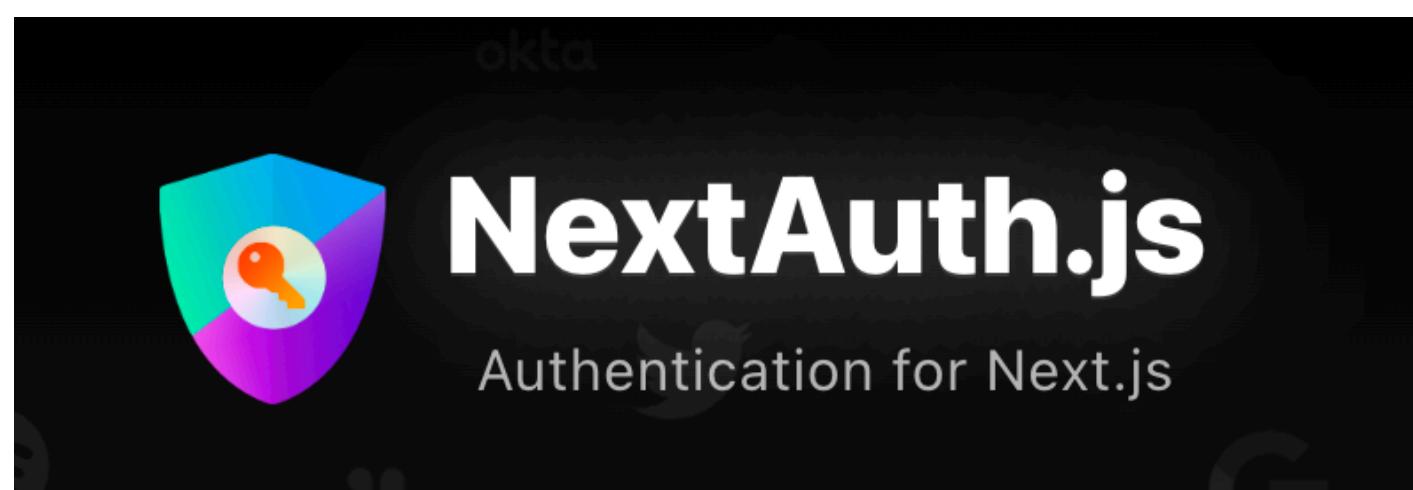
- 사용자 풀 - 사용자의 가입과 로그인을 제공하는 사용자 저장소
- 자격 증명 풀 - 사용자 풀에 저장된 정보를 바탕으로 로그인 또는 회원가입에 성공한 사용자에게
AWS 인프라의 여러 서비스에 대한 권한을 부여할 수 있는 서비스

[설명 출처\(링크\)](#)

<h3>핵심 기술 스택 소개</h3>

Next-auth & amazon-cognito-identity-js

Next-auth



next.js로 구현되어 있는 페이지에서 로그인을 쉽게 구현할 수 있도록
관련 기능을 제공하는 3rd Party 라이브러리

Oauth Provider 제공

Oauth 인증 방식의 로그인 서비스를 보다 쉽게 구현 가능(쪼렙인 저도 합니다.. 😊)

callback 함수를 이용한 jwt, session 조작 가능

client에서도 토큰 정보 일부를 노출시킬 수 있고,
useSEssion과 같은 툥을 통해 참조 가능 (서버 세션에서 가져오는 것도 가능)

amazon-cognito-identity-js

amazon-archives/
amazon-cognito-identit...



Amazon Cognito Identity SDK for JavaScript

56 Contributors 42 Used by 984 Stars 512 Forks

Amazon Cognito를 사용하여 사용자 인증 및 관리를 위한 클라이언트 측 라이브러리
웹 애플리케이션에서 사용자의 인증, 사용자 풀 관리,
그룹 및 권한 관리 등을 쉽게 처리할 수 있도록 도와줍니다.

<h4>핵심 구현</h4>

구현 UI 소개

Next-Auth with Cognito

Login

email

이메일을 입력해주세요.

password

비밀번호를 입력해주세요.

Login

Next-Auth with Cognito

Login

email

jurneyx2@gmail.com

이메일을 확인해주세요

password

.....

비밀번호를 확인해주세요

Login

안녕하세요! **최지연**님



Logout

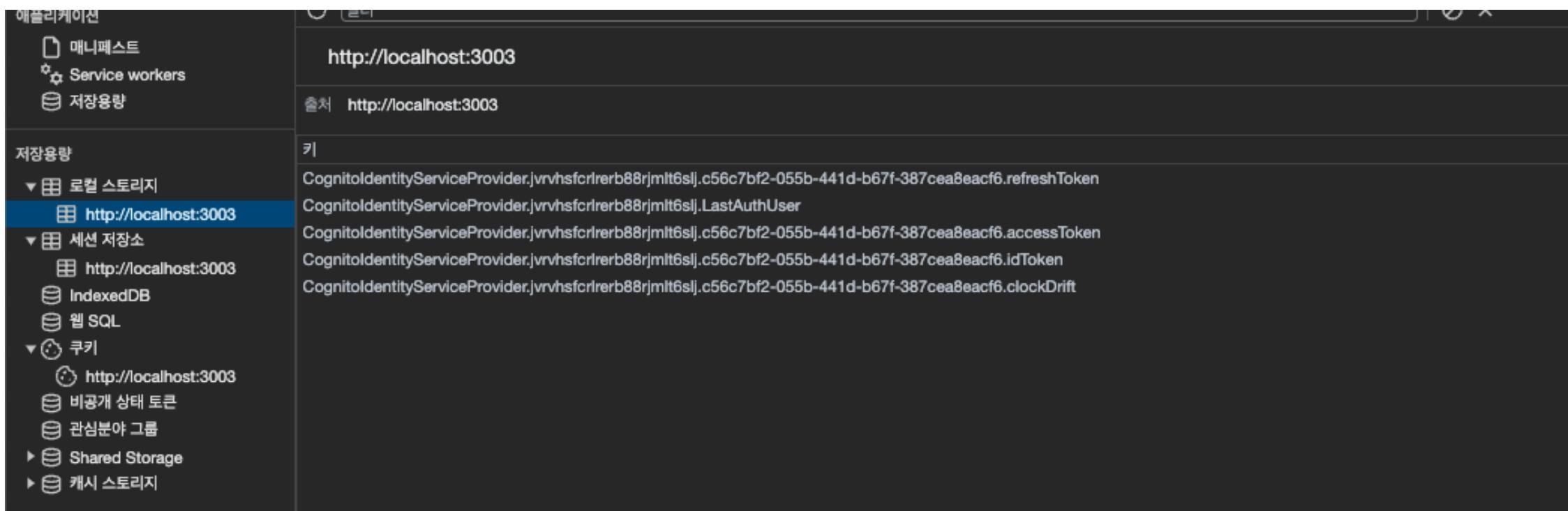
로그인 화면

로그인 실패

로그인 성공

<h4>핵심 구현</h4>

기존 - amazon-cognito-identity-js



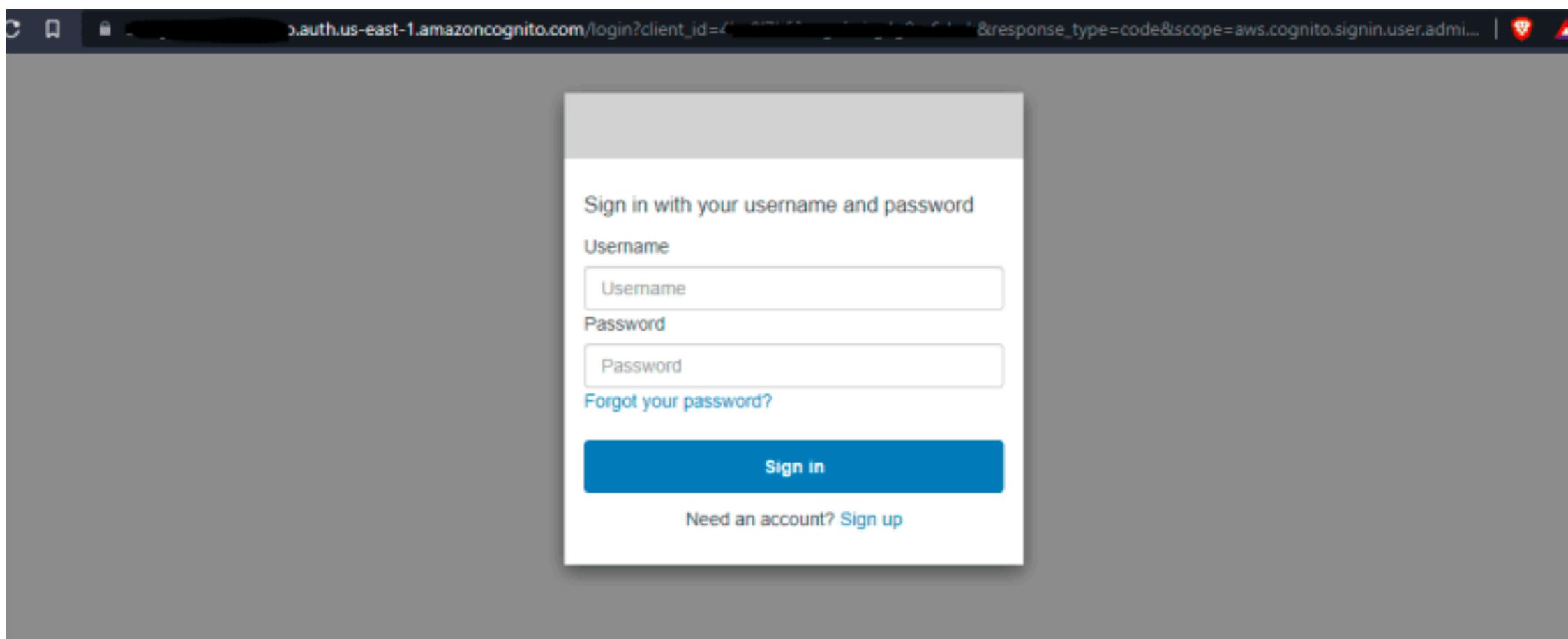
LocalStorage에 5개의 데이터들이 쌓임
Client의 상태 생명 주기에 의존

[Amazon-cognito-identity-js\(링크\)](#)

<h5>핵심 구현</h5>

기존 - Next-auth Cognito Provider

```
import CognitoProvider from "next-auth/providers/cognito";
...
providers: [
  CognitoProvider({
    clientId: process.env.COGNITO_CLIENT_ID,
    clientSecret: process.env.COGNITO_CLIENT_SECRET,
    issuer: process.env.COGNITO_ISSUER,
  })
]
...
...
```

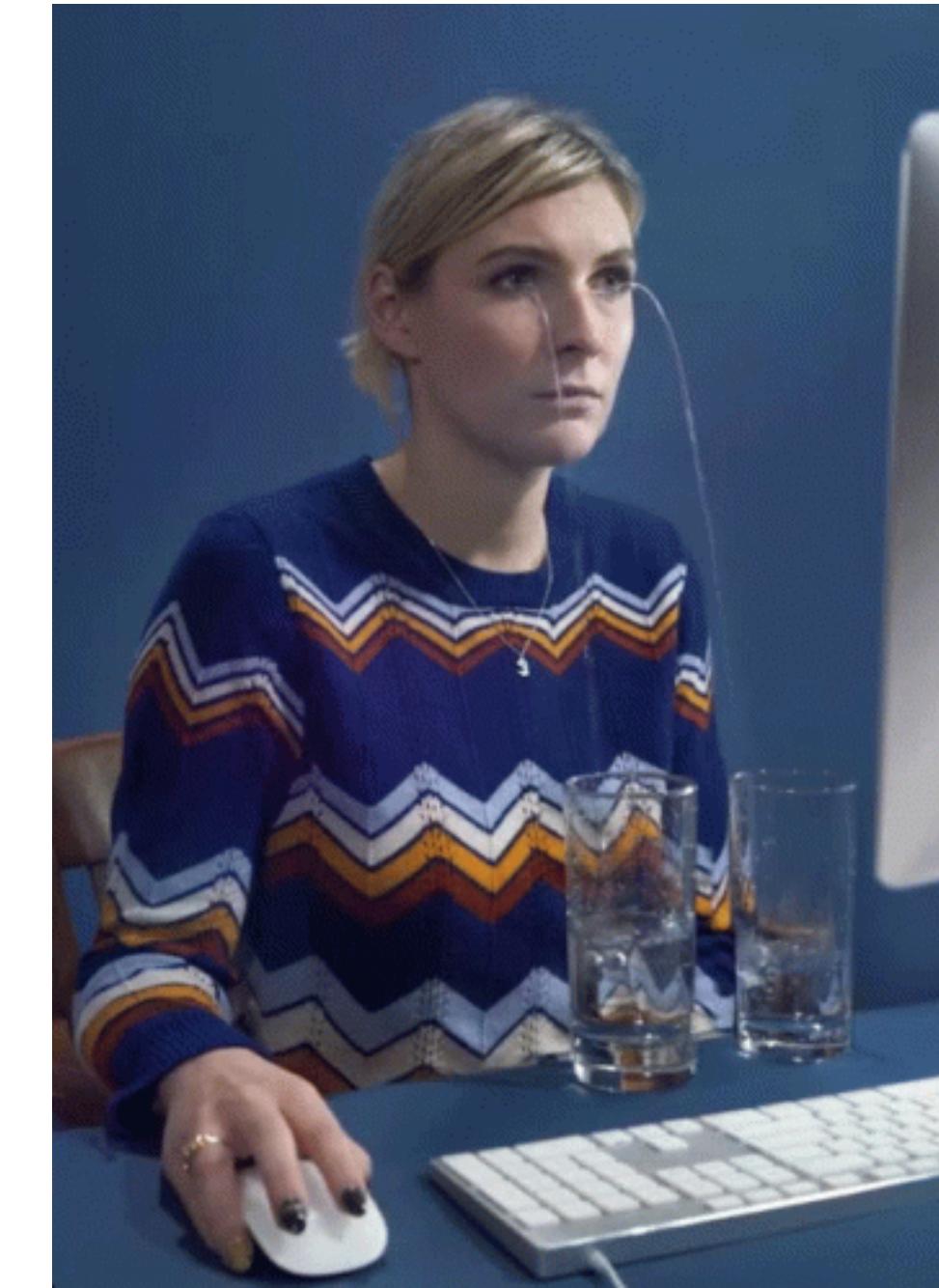


[Next-auth cognito 공식문서 \(링크\)](#)

Next-auth 내 존재하는 cognito provider으로 사용할 수 있으나
타 소셜 로그인처럼 외부 url로 이동 (feat. 멋스러운 레트로 UI)

<h5>핵심 구현</h5>

기준 - 내 심정



자료가 절망적으로 없음..(stackoverflow도 전멸 수준.. 왜 도전하신 인도인 분들도 없는 거죠)

유연한 UI를 쉽게 구축할 수 있다면서 방법은 안 알려줌

<h5>핵심 구현</h5>

적용 - Next-auth Credentials 이용

사용자가 직접 임의의 credentials(로그인에 필요한 정보들)을 구성하여 로그인을 구현할 수 있도록 만들어 주는 provider

[next-auth 공식문서 \(링크\)](#)

The screenshot illustrates the setup for a user pool in the AWS Cognito console and its implementation in a Next.js application.

사용자 풀 세부 정보 정보
사용자 풀 ID와 앱 클라이언트 ID를 선택하세요.

사용자 풀 ID: [REDACTED]
액 클라이언트 ID: [REDACTED]

.env

```
1 BASE_URL="http://localhost:3003"
2
3 NEXT_PUBLIC_COGNITO_CLIENT_ID=
4 NEXT_PUBLIC_COGNITO_USER_POOL_ID=
5 NEXTAUTH_URL=$BASE_URL
6 NEXTAUTH_SECRET=
```

폴더 구조

환경 변수

<h5>핵심 구현</h5>

적용 - Next-auth Credentials 이용

사용자가 직접 임의의 credentials(로그인에 필요한 정보들)을 구성하여 로그인을 구현할 수 있도록 만들어 주는 provider

[next-auth 공식문서 \(링크\)](#)

Credentials

credentials 프로퍼티를 통해 어떤 입력 정보를 받을지 지정

패당 프로퍼티값을 가진 객체가 authorize 콜백 함수의 첫 번째 인수로 전달

Authorize

credentials 값을 통해 해당 사용자의 로그인 가능 여부를 판단하여

로그인을 제어할 수 있는 함수 (여기서 Cognito-identity-js API를 사용)

```
export const authOptions: NextAuthOptions = {
  providers: [
    CredentialsProvider({
      name: 'next-auth-with-cognito-login',
      credentials: {
        email: { label: 'Email', type: 'text' },
        password: { label: 'Password', type: 'password' },
      },
    }),
  ],
}
```

```
authorize(credentials) {
  const userPoolId = process.env.NEXT_PUBLIC_COGNITO_USER_POOL_ID
  const clientId = process.env.NEXT_PUBLIC_COGNITO_CLIENT_ID

  const poolData = [
    UserPoolId: userPoolId as string,
    ClientId: clientId as string,
  ]

  const userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData)

  const authenticationData = {
    Username: credentials?.email as string,
    Password: credentials?.password,
  }

  const authenticationDetails =
    new AmazonCognitoIdentity.AuthenticationDetails(authenticationData)

  const userData = {
    Username: credentials?.email as string,
    Pool: userPool,
  }
  const cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData)
}
```

<h5>핵심 구현</h5>

적용 - Next-auth Credentials 이용

사용자가 직접 임의의 credentials(로그인에 필요한 정보들)을 구성하여 로그인을 구현할 수 있도록 만들어 주는 provider

[next-auth 공식문서 \(링크\)](#)

Authorize

authenticateUser 함수를 호출한 후,
반환된 결과에 따라 Promise를 사용해 비동기적으로 작업을 처리

- **onSuccess:** 사용자가 성공적으로 인증되면 호출되며, 여기서는 세션 정보에서 사용자 정보를 추출해 userInfo 객체를 만들고 resolve를 호출하여 해당 정보를 반환
- **onFailure:** 인증에 실패한 경우 호출되며, 여기서는 오류가 있다면 해당 오류를 reject를 통해 반환

```
const cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData)

return new Promise((resolve, reject) => {
  cognitoUser.authenticateUser(authenticationDetails, {
    onSuccess: (session) => {
      if (session instanceof AmazonCognitoIdentity.CognitoUserSession) {
        const userInfo: DefaultUser = {
          id: session.getIdToken().payload.sub,
          email: session.getIdToken().payload.email,
          name: session.getIdToken().payload.name,
        }
        resolve(userInfo)
      }
    },
    onFailure: (err) => {
      if (err) reject(err)
    },
  })
})
```

<h5>핵심 구현</h5>

적용 - Next-auth Credentials 이용

사용자가 직접 임의의 credentials(로그인에 필요한 정보들)을 구성하여 로그인을 구현할 수 있도록 만들어 주는 provider

[next-auth 공식문서 \(링크\)](#)

Callback

JWT 토큰과 세션을 관리하는 데 사용

+) Handler

해당 코드가 Next.js API 라우트의 핸들러 역할을 수행하도록 설정

```
callbacks: {  
  async jwt({ token, user }) {  
    return { ...token, ...user }  
  },  
  async session({ session, token }) {  
    session.user = token as any  
    return session  
  },  
},
```

```
const handler = NextAuth(authOptions)  
  
export { handler as GET, handler as POST }
```

<h5>핵심 구현</h5>

결과 - Next-auth Credentials 이용

기존 클라이언트에서 호출

The screenshot shows the Network tab of the browser developer tools. On the left, there's a tree view of network requests and resources. On the right, a table lists various cookies and tokens. Key entries include:

- Cookie: CognitoidentityServiceProvider.jrvhsfcrlerb88rjmlt6slj.c56c7bf2-055b-441d-b67f-387cea8eacf6.refreshToken
- Cookie: CognitoidentityServiceProvider.jrvhsfcrlerb88rjmlt6slj.LastAuthUser
- Cookie: CognitoidentityServiceProvider.jrvhsfcrlerb88rjmlt6slj.c56c7bf2-055b-441d-b67f-387cea8eacf6.accessToken
- Cookie: CognitoidentityServiceProvider.jrvhsfcrlerb88rjmlt6slj.c56c7bf2-055b-441d-b67f-387cea8eacf6.idToken
- Cookie: CognitoidentityServiceProvider.jrvhsfcrlerb88rjmlt6slj.c56c7bf2-055b-441d-b67f-387cea8eacf6.clockDrift

Next-auth를 이용한 Server-side 호출

The screenshot shows the Network tab of the browser developer tools. On the left, there's a tree view of network requests and resources. On the right, a table lists various cookies. Key entries include:

이름	값	Do...	Path	Expires / Max-Age	크기	HttpOnly
next-auth.session-token	eyJhbGciOiJkaXIIJCJlbmMiOiJBMIU2R0NNIn0..TC8lHeZ7-IU85Db4.0vS7UGZfy0SrL-E_5-Fl...	loca...	/	2024-02-01T18:39:15.141Z	394	✓
next-auth.callback-url	http%3A%2F%2Flocalhost%3A3003%2F%3Femail%3D%26password%3D	loca...	/	세션	79	✓
next-auth.csrf-token	bd2ce10e273f70ea93e9260f246cd7d6db3dfca746c013b5901e9c4ecb853b7e%7C8d57d1a...	loca...	/	세션	151	✓

<h6>트러블 슈팅</h6>

이라 말하고 유의 사항

client secret을 설정하면 접근이 안됨



client secret을 crypto를 이용해 secretHash를 만들어
페이로드에 보냄 ➡ 어림 없음 (공식 문서엔 왜.. 된다구..)

토큰 엔드 포인트에 차근 차근 append해서 Post 요청을 보냄
➡ 어림 없음

온갖 도전 시작



Configuration

The Amazon Cognito Identity SDK for JavaScript requires two configuration values from your AWS Account in order to access your Cognito User Pool:

- The User Pool Id, e.g. us-east-1_aB12cDe34
- A User Pool App Client Id, e.g. 7ghr5379orhbo88d52vphda6s9
 - When creating the App, the generate client secret box must be **unchecked** because the JavaScript SDK doesn't support apps that have a client secret.

The [AWS Console for Cognito User Pools](#) can be used to get or create these values.

If you will be using Cognito Federated Identity to provide access to your AWS resources or Cognito Sync you will also need the Id of a Cognito Identity Pool that will accept logins from the above Cognito User Pool and App, i.e. us-east-1:85156295-afa8-482c-8933-1371f8b3b145.

Note that the various errors returned by the service are valid JSON so one can access the different exception types (err.code) and status codes (err.statusCode).



JS SDK에서 되지 않으니 꼭 client secret을 해제!

Thank you for watching!

#AWS_한국_사용자_모임
#프론트엔드



2024.01.03
GitHub @lmjourney