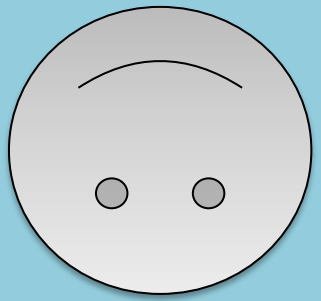
OAuth 2.0

Mash-up 10기 노드팀 조준형

OAuth = ~~Authentication~~ + Authorization

인터넷 사용자들이 **비밀번호를 제공하지 않고 다른 웹사이트 상의 자신들의 정보**에 대해 **웹사이트나 애플리케이션의 접근 권한을 부여**할 수 있는 공통적인 수단으로서 사용되는, 접근 위임을 위한 개방형 표준이다.

[https://ko.wikipedia.org/wiki/OAuth]



고객

우리 서비스는 구글에 있는 너의 정보가 필요하니까 구글 아이디랑 비밀번호 좀 알려줘..

아이디 비밀번호

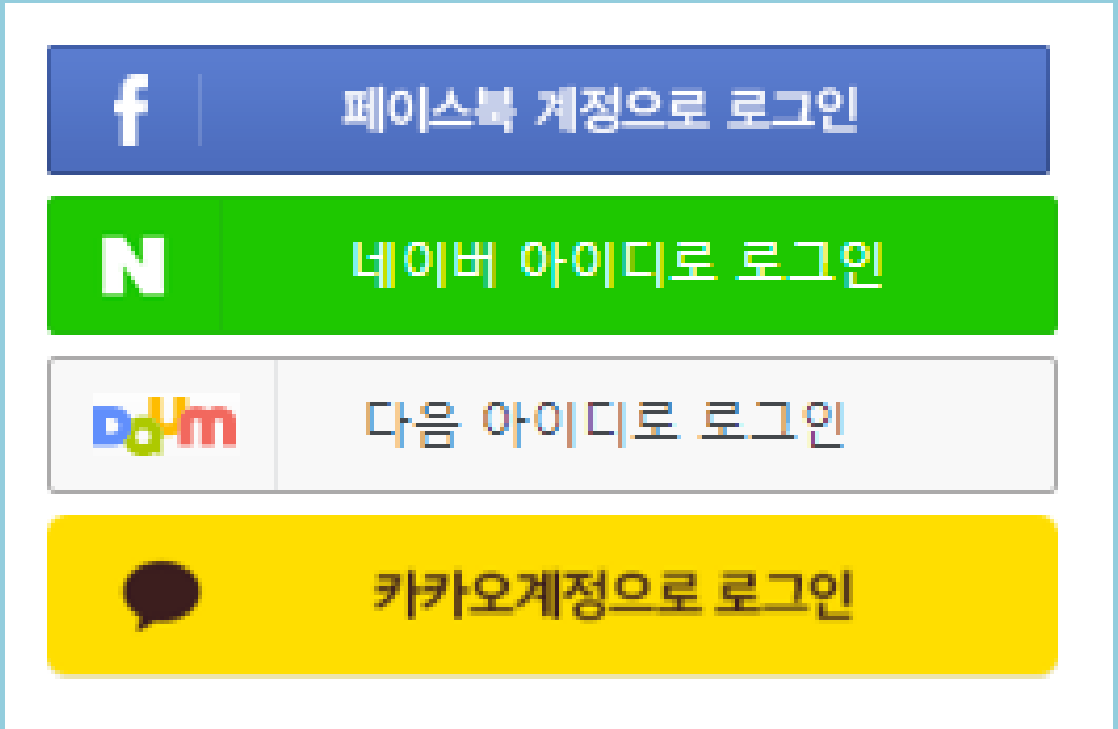
우리가 만든 서비스

거대 서비스
ex) 네이버, 카카오, 구글, 깃헙

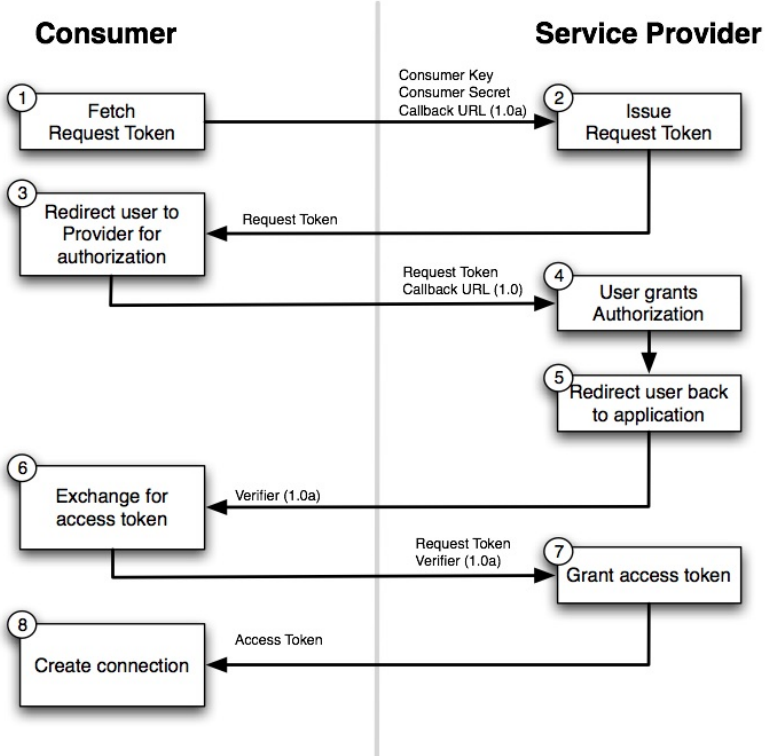
고객 정보

이름, 나이, 전화번호, 이메일, 성별, 주소 등

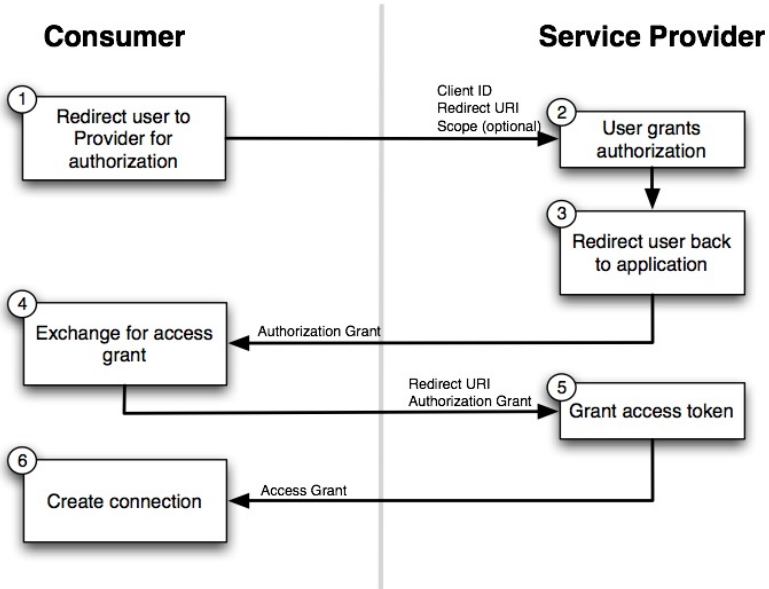
사용자의 로그인 정보 없이 어떻게
다른 서비스에 있는 사용자의
정보를 얻을 수는 없을까?


페이스북 계정으로 로그인
네이버 아이디로 로그인
다음 아이디로 로그인
카카오계정으로 로그인

OAuth 1.0



OAuth 2.0



OAuth2.0의 4가지 인증 방법

Authorization Code Grant Type : 권한 부여 코드 승인

- 클라이언트가 다른 사용자 대신 특정 리소스에 접근을 요청할 때 사용
- 리스소 접근을 위한 사용자 명과 비밀번호, 권한 서버에 요청해서 받은 권한 코드를 함께 활용하여 리소스에 대한 엑세스 토큰을 받는 방식

Implicit Grant Type : 암시적 승인

권한 부여 코드 승인 타입과 다르게 권한 코드 교환 단계 없이 엑세스 토큰을 즉시 반환받아 이를 인증에 이용하는 방식입니다.

Resource Owner Password Credentials Grant Type : 리소스 소유자 암호 자격 증명

클라이언트가 암호를 사용하여 엑세스 토큰에 대한 사용자의 자격 증명을 교환

Client Credentials Grant Type : 클라이언트 자격 증명

클라이언트가 컨텍스트 외부에서 액세스 토큰을 얻어 특정 리소스에 접근을 요청할 때 사용하는 방식입니다.

출처 : https://cheese10yun.github.io/oauth2/

거대
서비스

인증서버

리소스
서버

1. Client ID
2. Client Secret
3. Authorized redirect URL

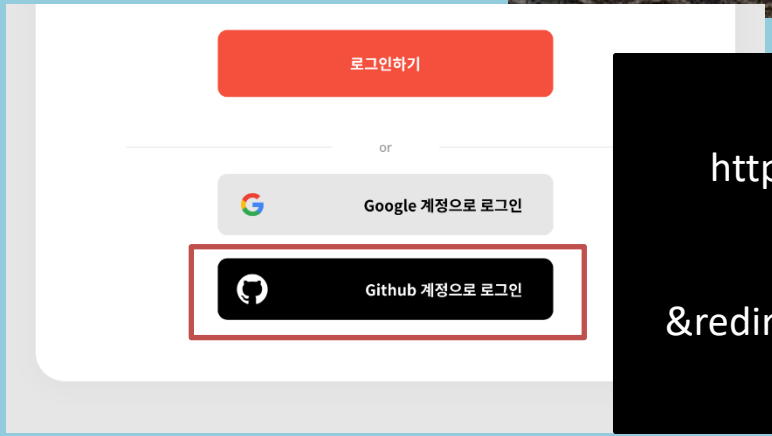
우리가
만든
서비스

사용자

거대
서비스

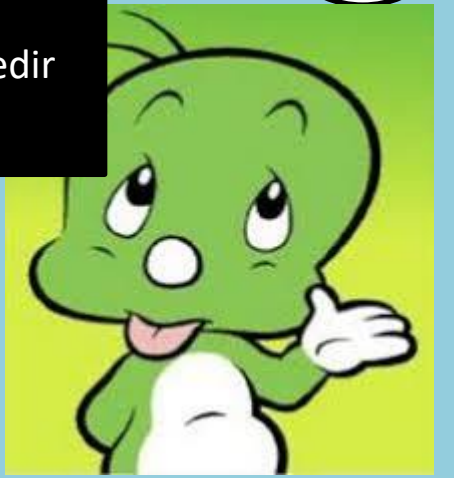
인증서버

리소스
서버



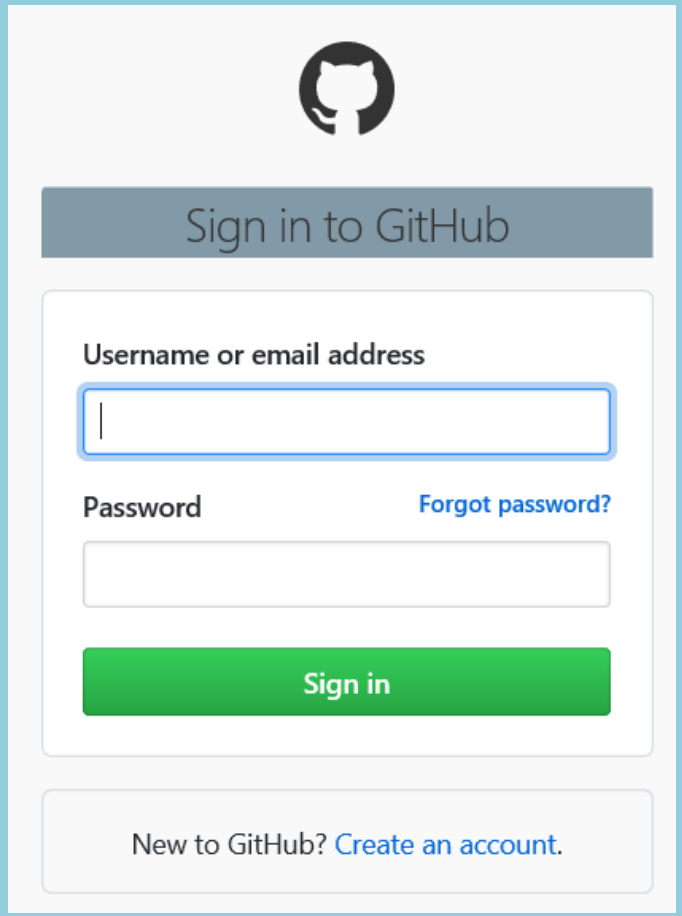
https://github.com/login/oauth/authorize?
client_id=CLIENT_ID
&scop=user:name
&redirect_url=http://localhost:3000/oauth-redir

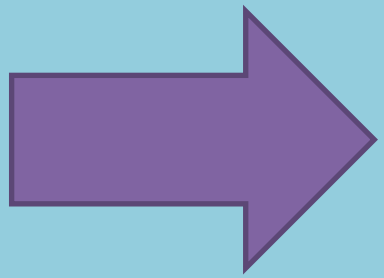
우리가
만든
서비스



사용자

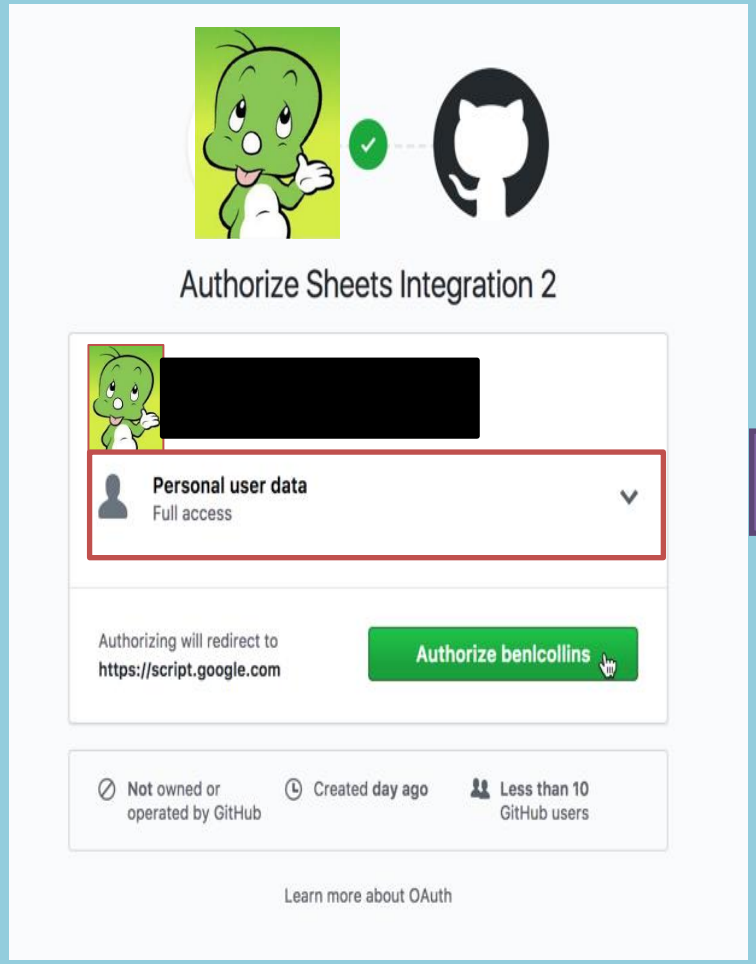
페이지





거대
서비스

Cilient_id, redirect_uri값 확인



User가 허용한 사실 정보를 저장

거대
서비스

Authorization code를 발급

Location :
http://localhost:3000/oauth-redir?code=3

사용자

나는 아무것도
모른다..

사용자

[Redirect]
http://localhost:3000/oauth-redir?code=3

우리가
만든
서비스

Code가 3을 받음

거대
서비스

https://github.com/login/oauth/access_token?
client_id=CLIENT _ID
&client_secret=CLIENT_SECRET
&code=3
&redirect_uri= http://localhost:3000/oauth-redir

우리가
만든
서비스

CLIENT _ID와 CLIENT_SECRET, 그리고 그 외 모든 정보를 확인하고 Access_token 발급

거대
서비스

https://github.com/login/oauth/access_token?
client_id=CLIENT _ID
&client_secret=CLIENT_SECRET
&code=3
&redirect_uri= http://localhost:3000/oauth-redir

우리가
만든
서비스

CLIENT _ID와 CLIENT_SECRET, 그 외 모든 정보를 확인
=> code 값을 삭제
=> 해당 유저의 정보에 접근할 수 있는
Access_token 발급

거대
서비스

accessToken=4

우리가
만든
서비스

accesToken=4를 저장

거대
서비스

accessToken=4

우리가
만든
서비스

accesToken=4를 저장

거대
서비스

Authorization : Access_Token
https://api.github.com/user

요청

응답

우리가
만든
서비스

Access_Token값을 확인하고, User Data를 응답

```
const githubLogin = (req, res) => {
  const state = rs.generate()
  const url = 'https://github.com/login/oauth/authorize?'
  const query = qs.stringify({
    client_id: process.env.CLIENT_ID,
    redirect_uri:
      process.env.NODE_ENV === 'development'
        ? process.env.BACKEND_HOST + '/api/users/githublogin'
        : process.env.PRODUCTION_HOST + '/api/users/githublogin',
    state: state,
    scope: 'read:user',
  })
  const githubAuthUrl = url + query
  res.send(githubAuthUrl)
}
```

```
const handleGithubCallback = async (req, res) => {
  const code = req.query.code
  const returnedState = req.query.state
  const githubUrl = 'https://github.com/login/oauth/access_token?'
  const query = qs.stringify({
    client_id: process.env.CLIENT_ID,
    client_secret: process.env.CLIENT_SECRET,
    code: code,
    redirect_uri:
      process.env.NODE_ENV === 'development'
        ? process.env.BACKEND_HOST + '/api/users/githublogin'
        : process.env.PRODUCTION_HOST + '/api/users/githublogin',
    state: returnedState,
  })
  const authUrl = githubUrl + query

  try {
    const loginData = await axios.post(authUrl)
    const config = {
      headers: {
        Authorization: 'token ' + qs.parse(loginData.data).access_token,
        'User-Agent': 'Login-App',
      },
    }
    const { data } = await axios.get('https://api.github.com/user', config)
```

질문있으신가요..?