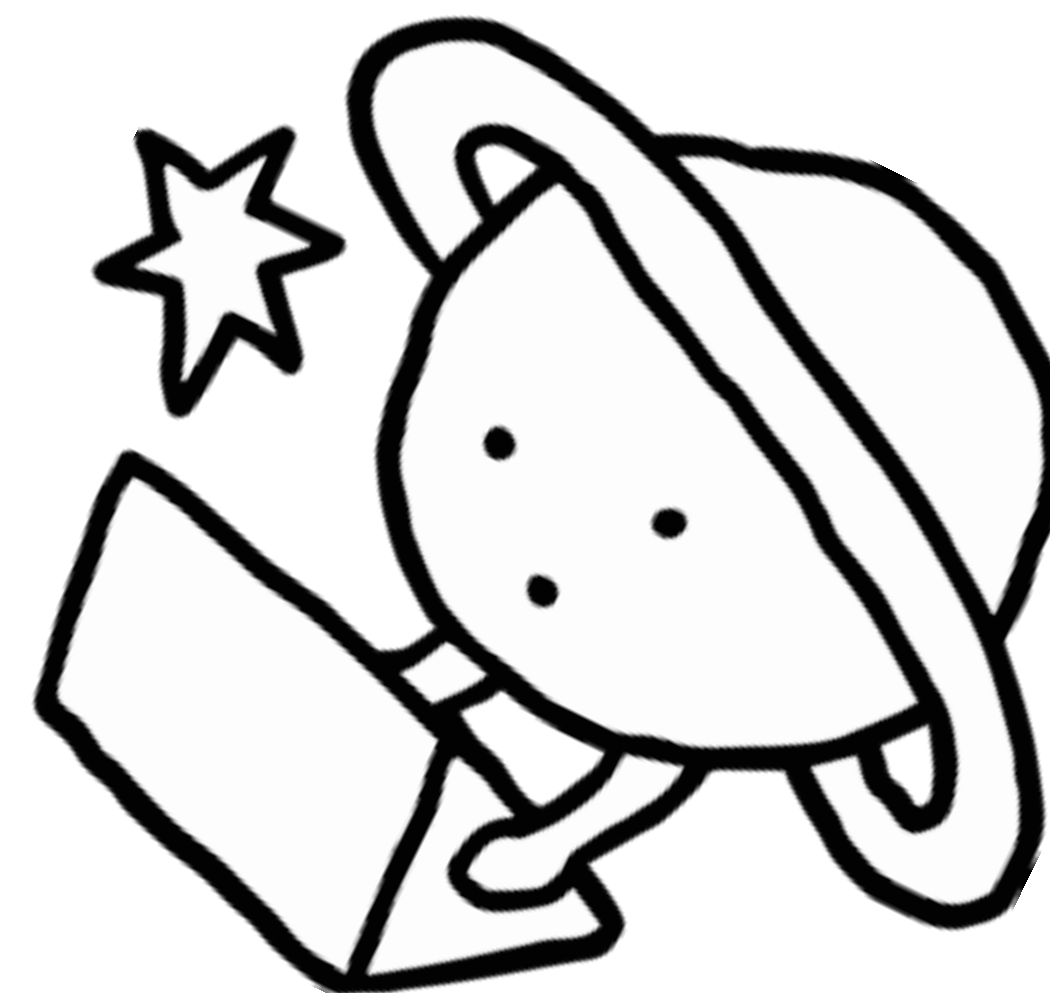
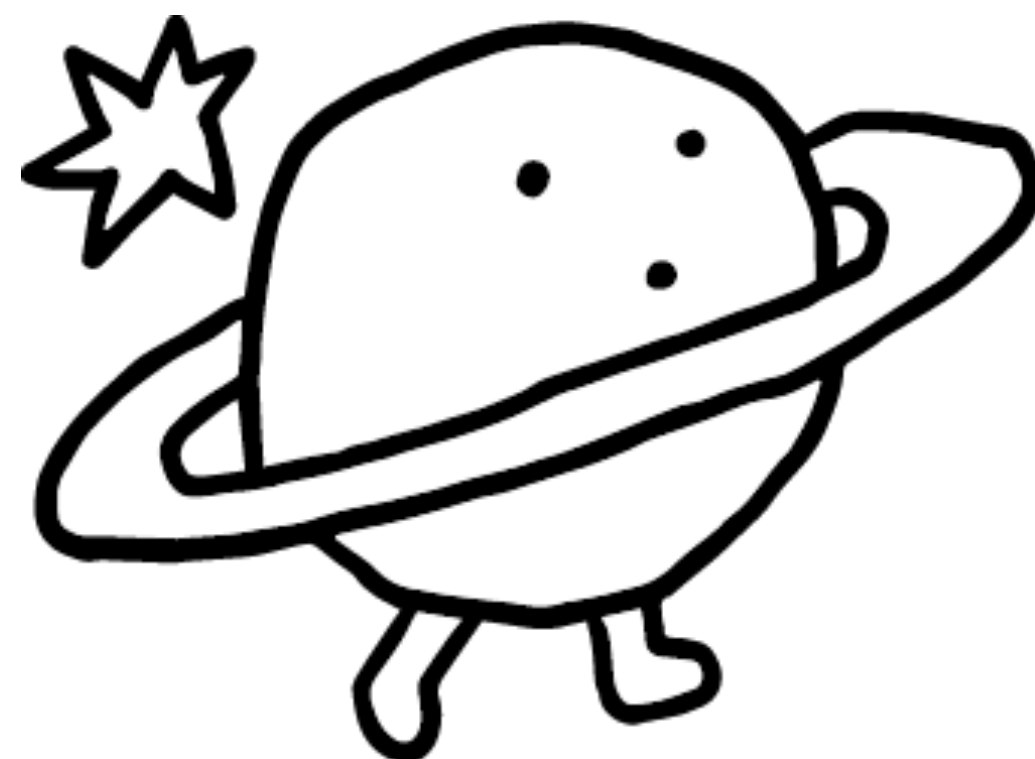


# OIDC 기반 소셜로그인 연동

jwt없이 자체 인증 체계 구축하기



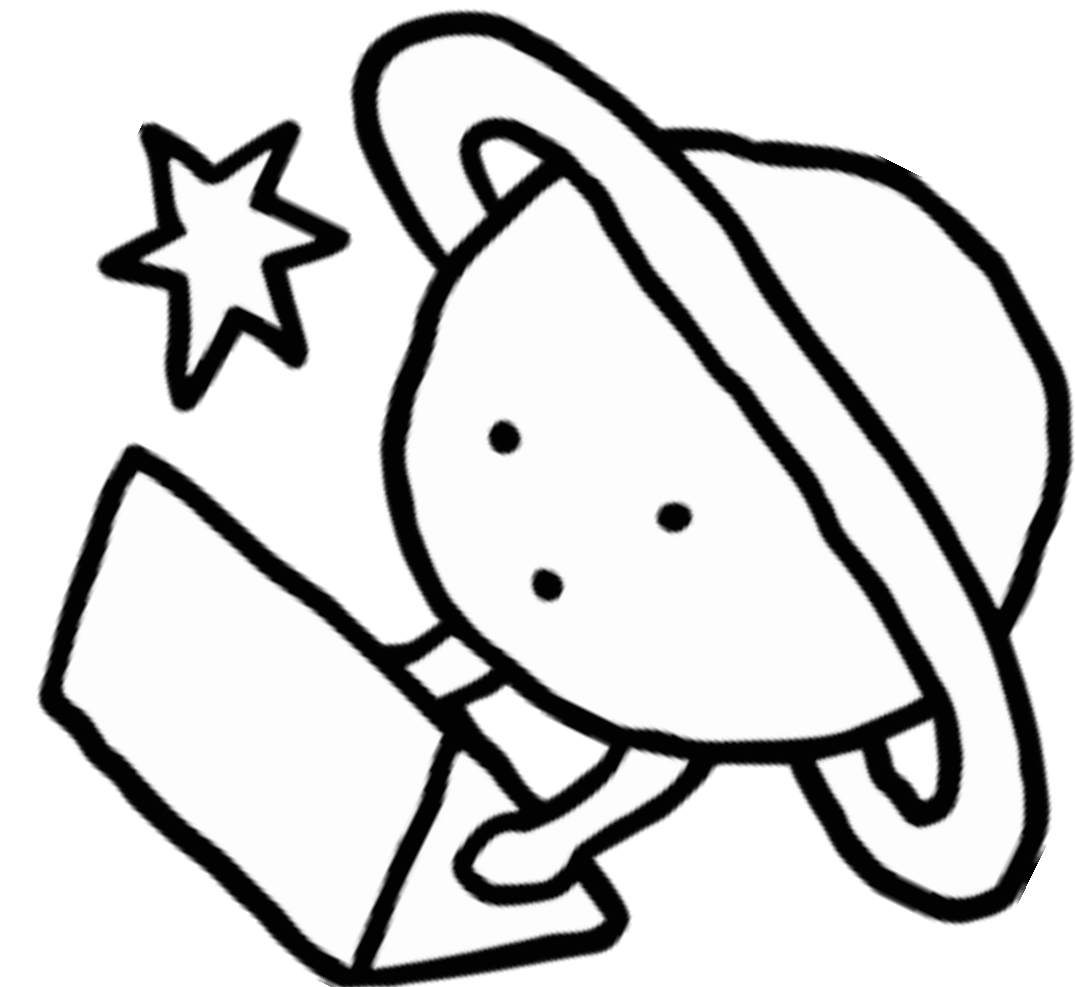
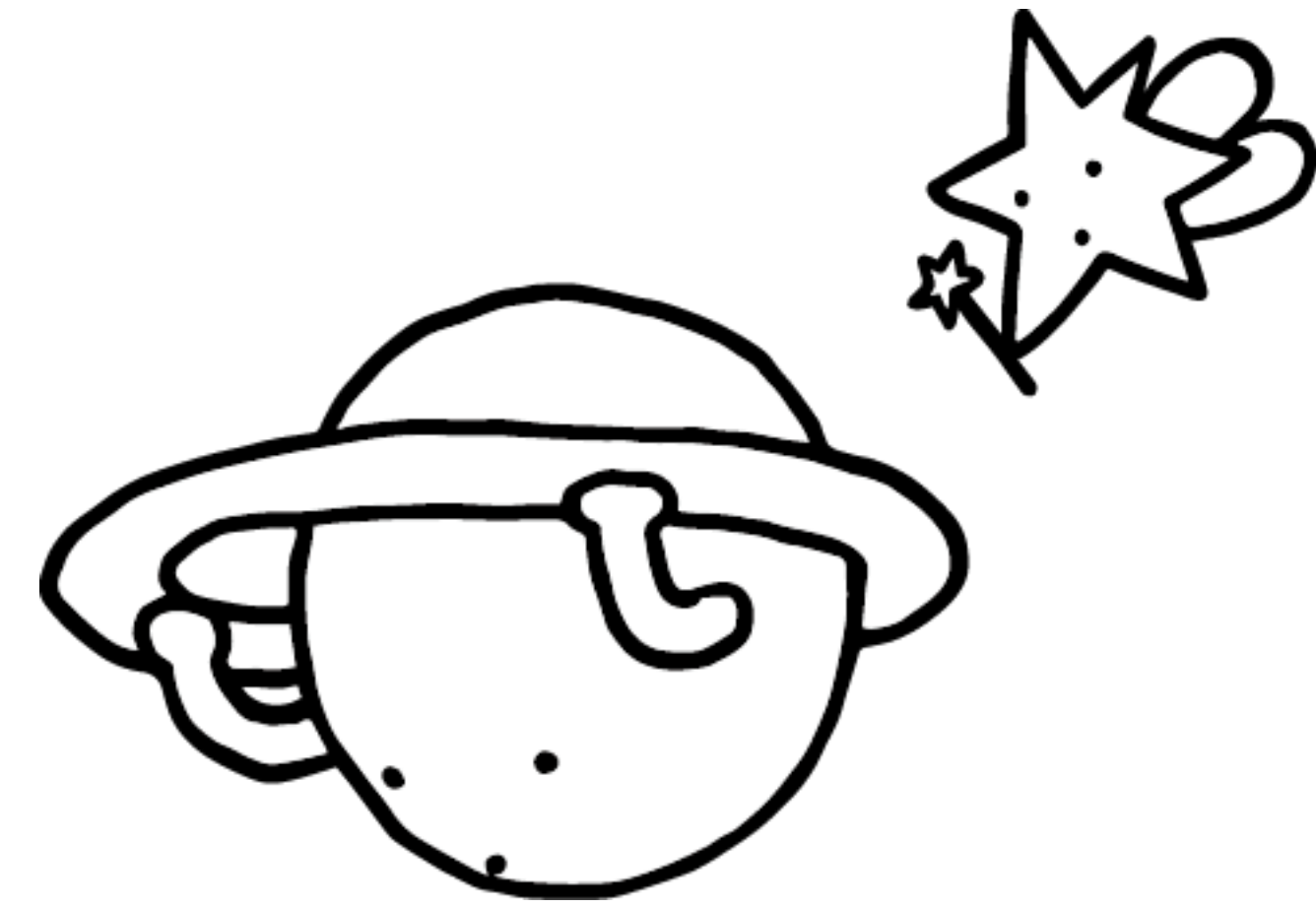
우아한테크코스 7기 BE 모코 🌱

# 목차

1. OAuth 2.0

2. OIDC

3. 자체 인증 체계



# OAuth 2.0

# OAuth 2.0이란?

**OAuth2.0(Open Authorization 2.0)**은 웹 및 애플리케이션 인증 및 권한 부여를 위한 개방형 표준 프로토콜입니다. 이 프로토콜에서는 **third-party 애플리케이션**이 **사용자의 리소스에 접근하기 위한 절차를 정의하고 서비스 제공자의 API를 사용할 수 있는 권한을 부여합니다.** 대표적으로 네이버 로그인, 구글 로그인과 같은 소셜 미디어 간편 로그인이 있습니다. OAuth2.0을 사용해 **third-party 애플리케이션**이 **사용자의 소셜미디어 프로필 정보에 접근할 수 있도록 합니다.**

# OAuth 2.0이란?

OAuth2.0(Open Authorization 2.0)은 웹 및 애플리케이션 인증 및 권한 부여를 위한 개방형 표준 프로토콜입니다. 이 프로토콜에서는 third-party 애플리케이션이 사용자의 리소스에 접근하기 위한 절차를 정의하고 서비스 제공자의 API를 사용할 수 있는 권한을 부여합니다. 대표적으로 네이버 로그인, 구글 로그인과 같은 소셜 미디어 간편 로그인이 있습니다. OAuth2.0을 사용해 third-party 애플리케이션이 사용자의 소셜미디어 프로필 정보에 접근할 수 있도록 합니다.

# OAuth 2.0의 역할

**리소스 소유자(Resource Owner):** 사용자. 클라이언트를 인증하는 역할을 수행한다.

**클라이언트(Client):** 리소스에 접근하려는 third-party 애플리케이션. 우리.

**권한 서버(Authorization Server):** 클라이언트가 리소스 소유자의 권한을 얻을 수 있도록 도와주는 서버입니다. 사용자 인증, 권한 부여 및 토큰 발급을 관리한다.

**리소스 서버(Resource Server):** 리소스를 호스팅하는 서버. 액세스를 허용하거나 거부하며 실제 데이터를 제공한다.

# OAuth 2.0의 역할

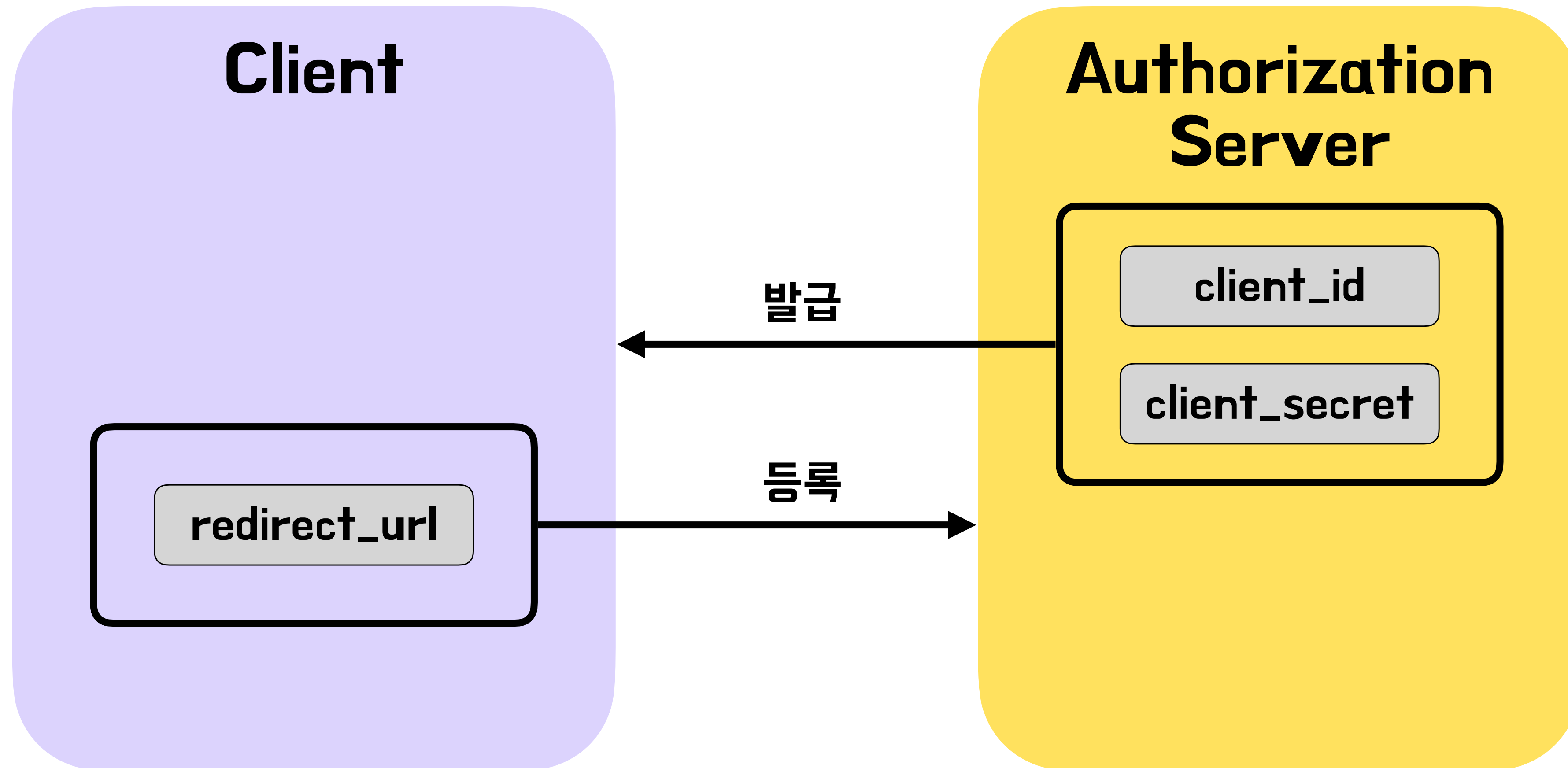
**리소스 소유자(Resource Owner):** 사용자. 클라이언트를 인증하는 역할을 수행한다.

**클라이언트(Client):** 리소스에 접근하려는 third-party 애플리케이션. 우리.

**권한 서버(Authorization Server):** 클라이언트가 리소스 소유자의 권한을 얻을 수 있도록 도와주는 서버입니다. 사용자 인증, 권한 부여 및 토큰 발급을 관리한다.

**리소스 서버(Resource Server):** 리소스를 호스팅하는 서버. 액세스를 허용하거나 거부하며 실제 데이터를 제공한다.

# OAuth 2.0 구성 요소





# OAuth 2.0 구성 요소

## Client

client\_id

client\_secret

redirect\_url

## Authorization Server

client\_id

client\_secret

redirect\_url

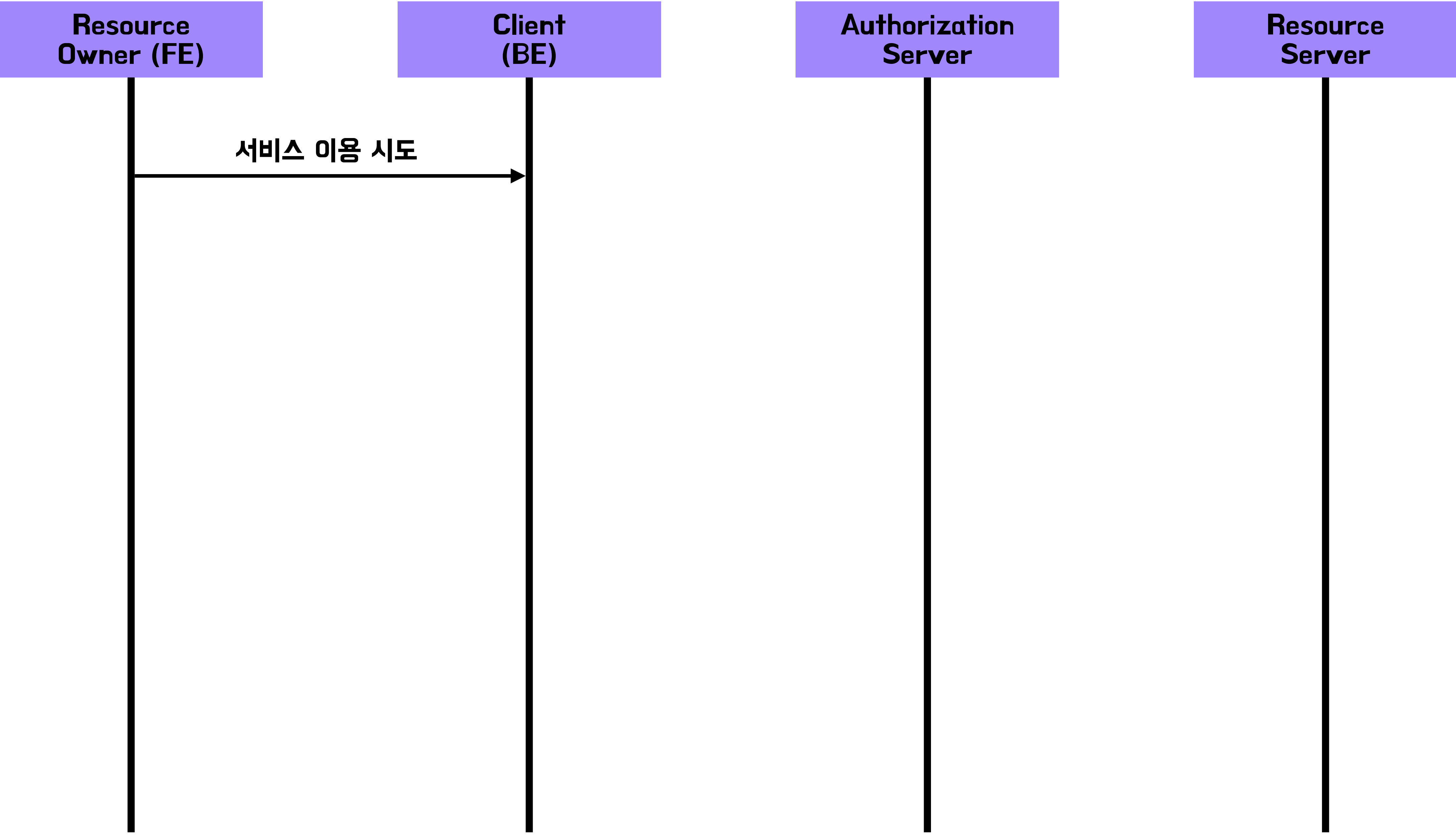
# OAuth 2.0 인증 과정

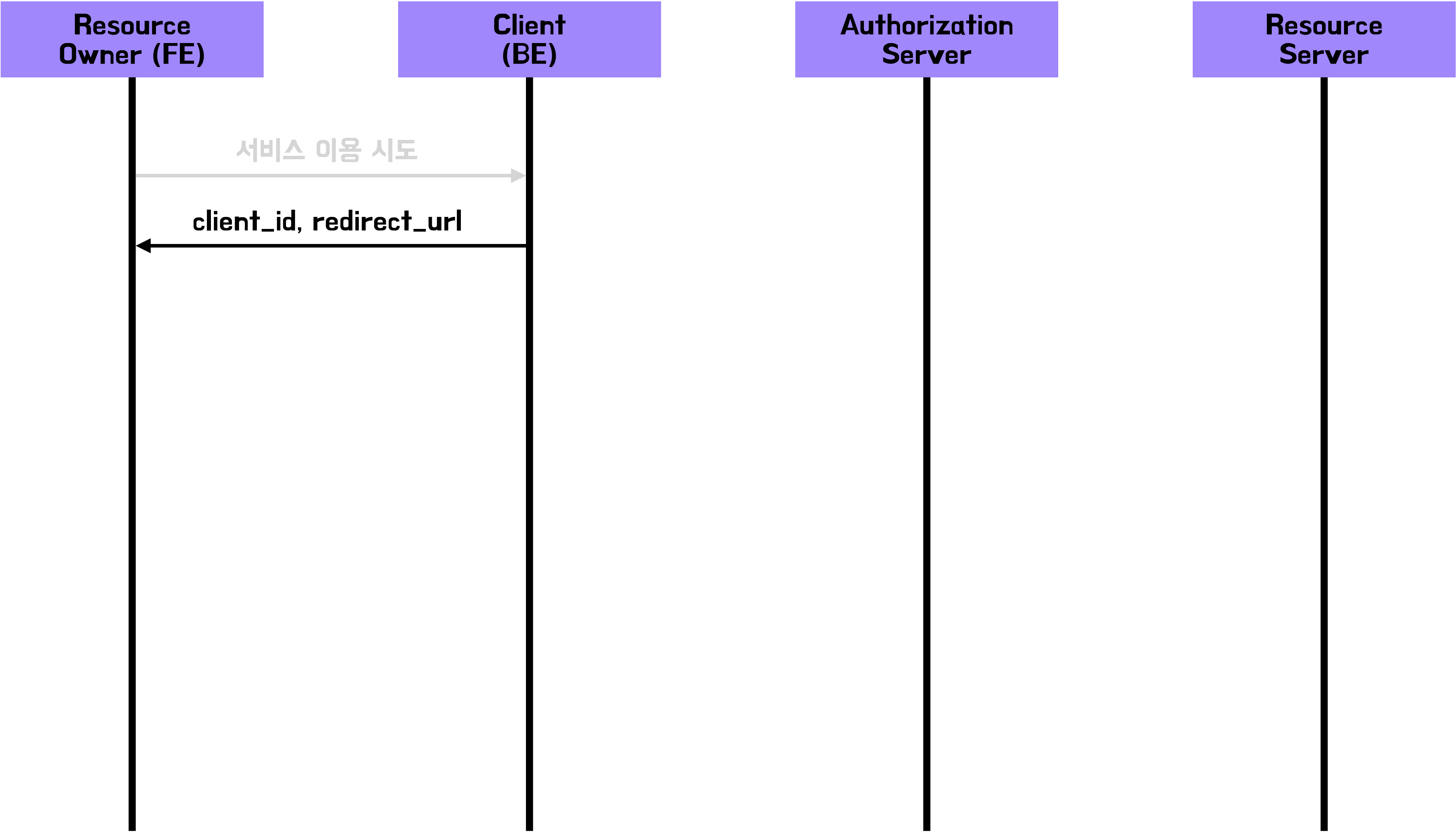
**Resource  
Owner (FE)**

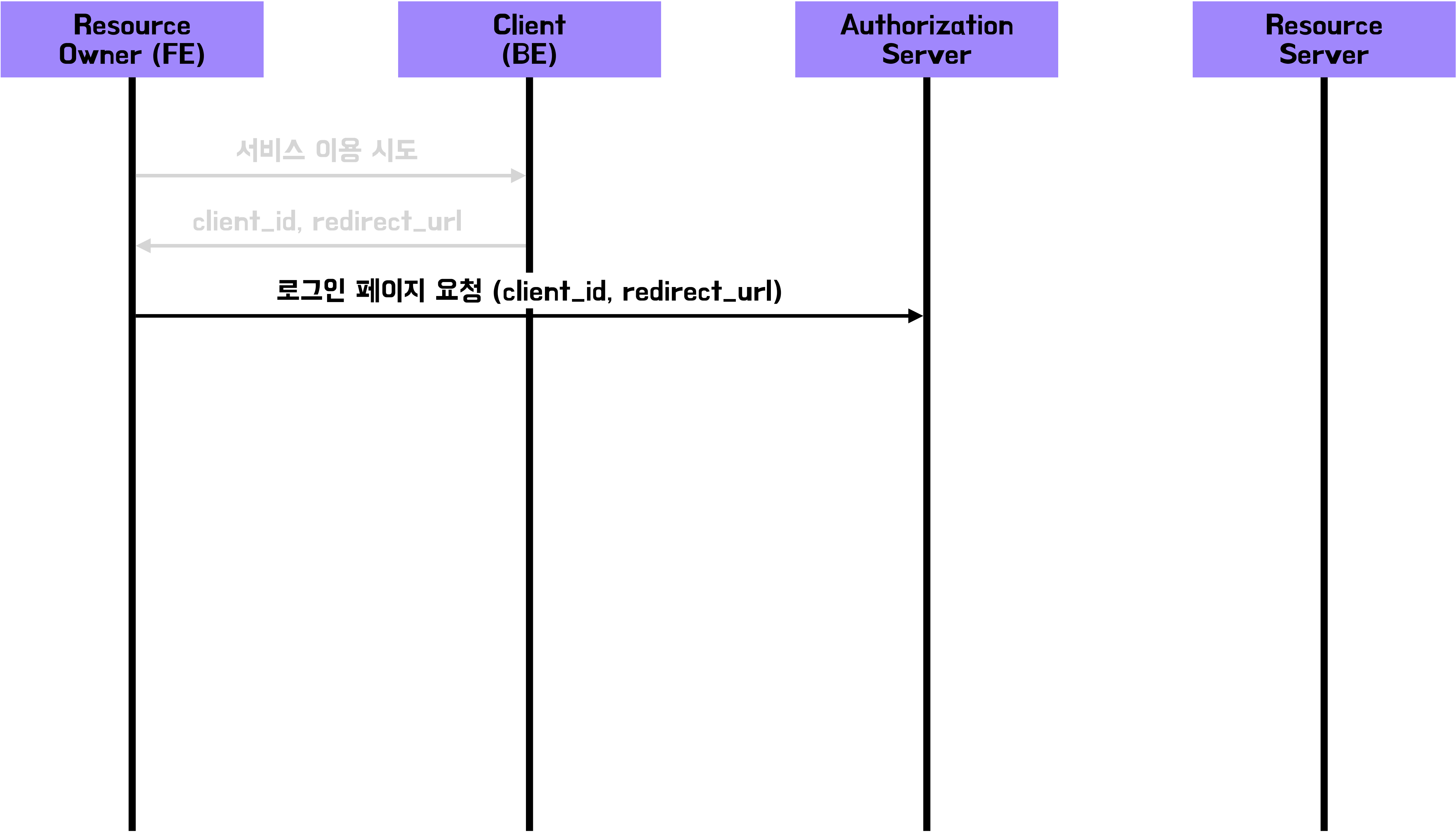
**Client  
(BE)**

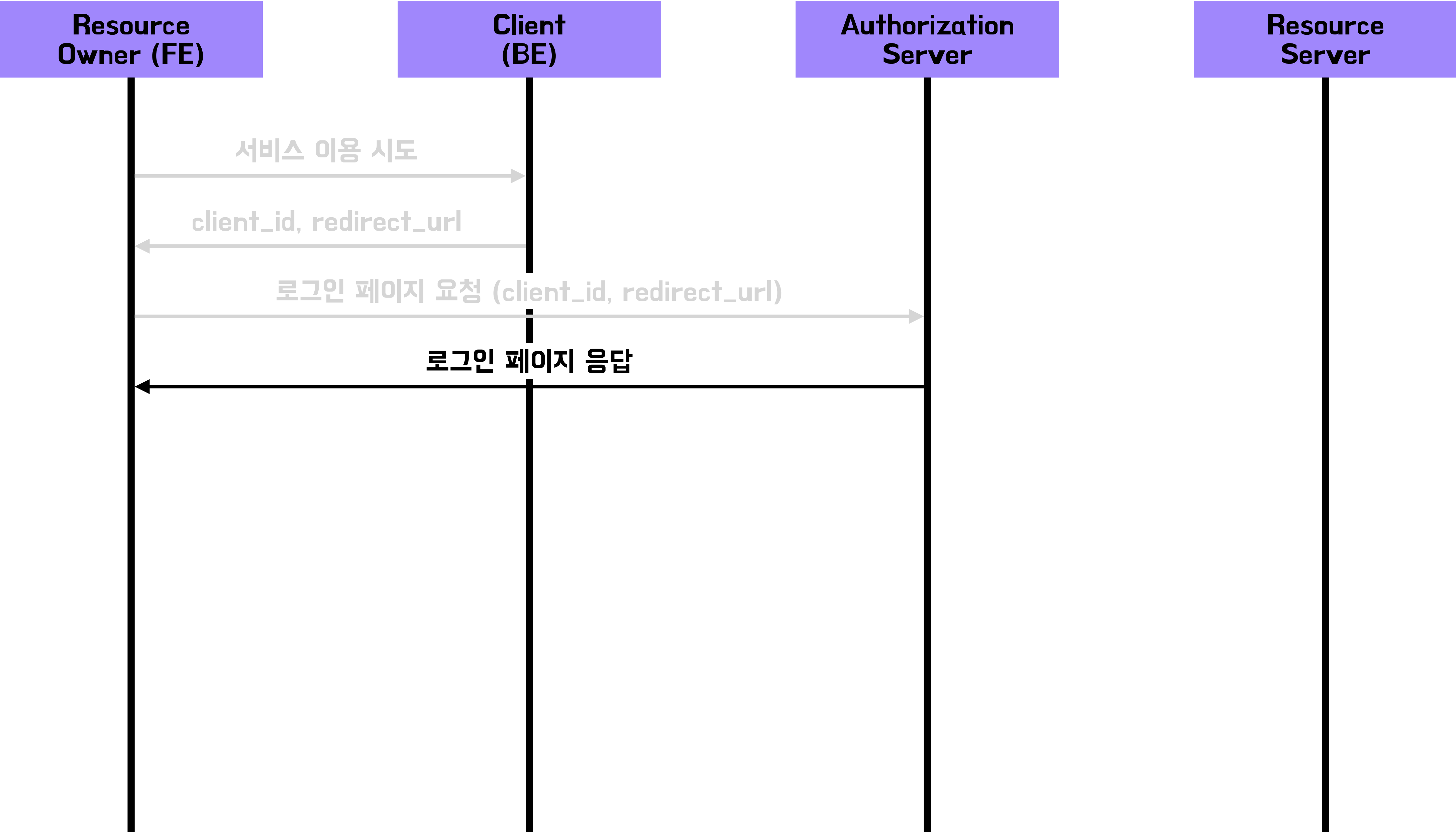
**Authorization  
Server**

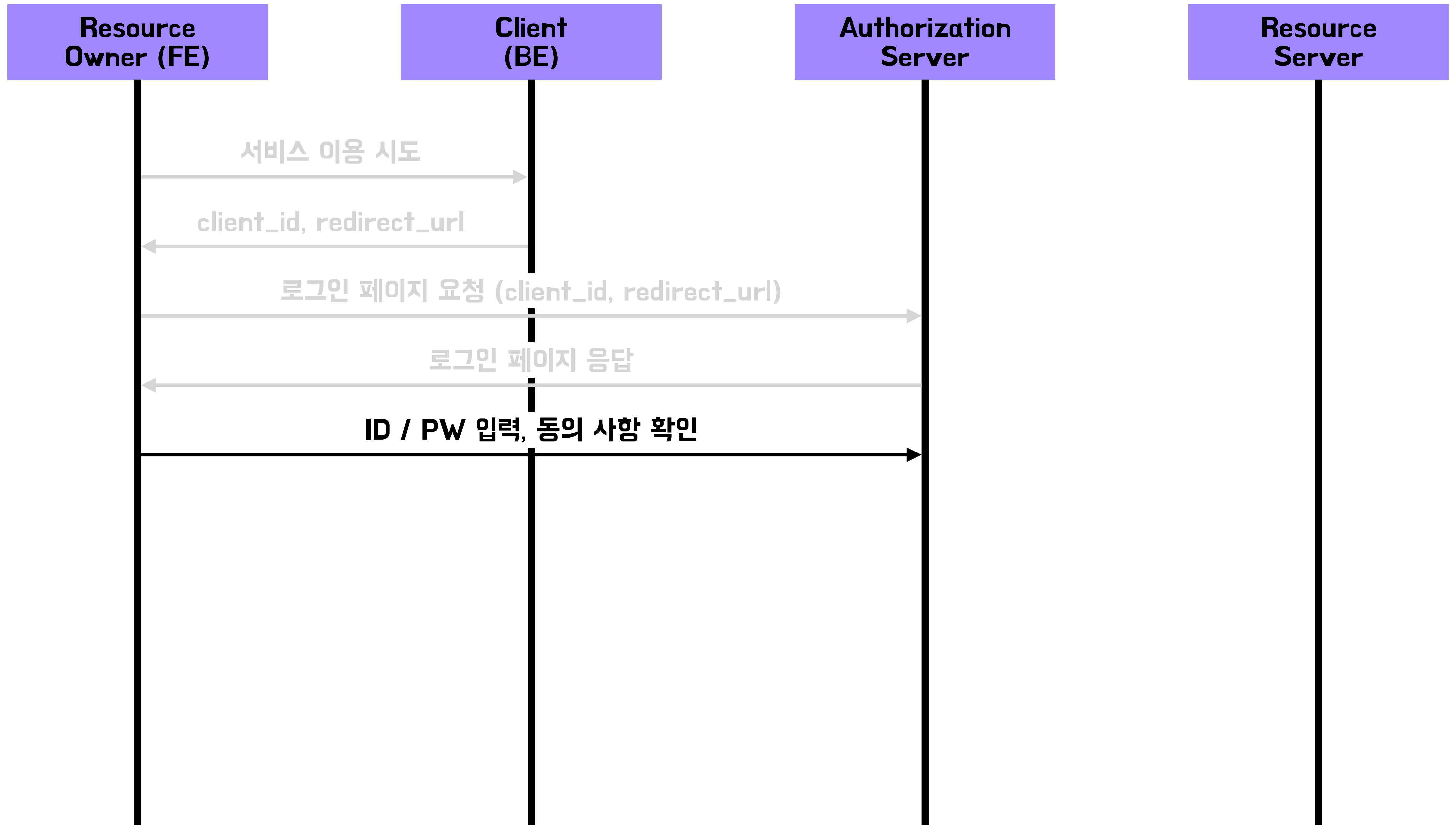
**Resource  
Server**



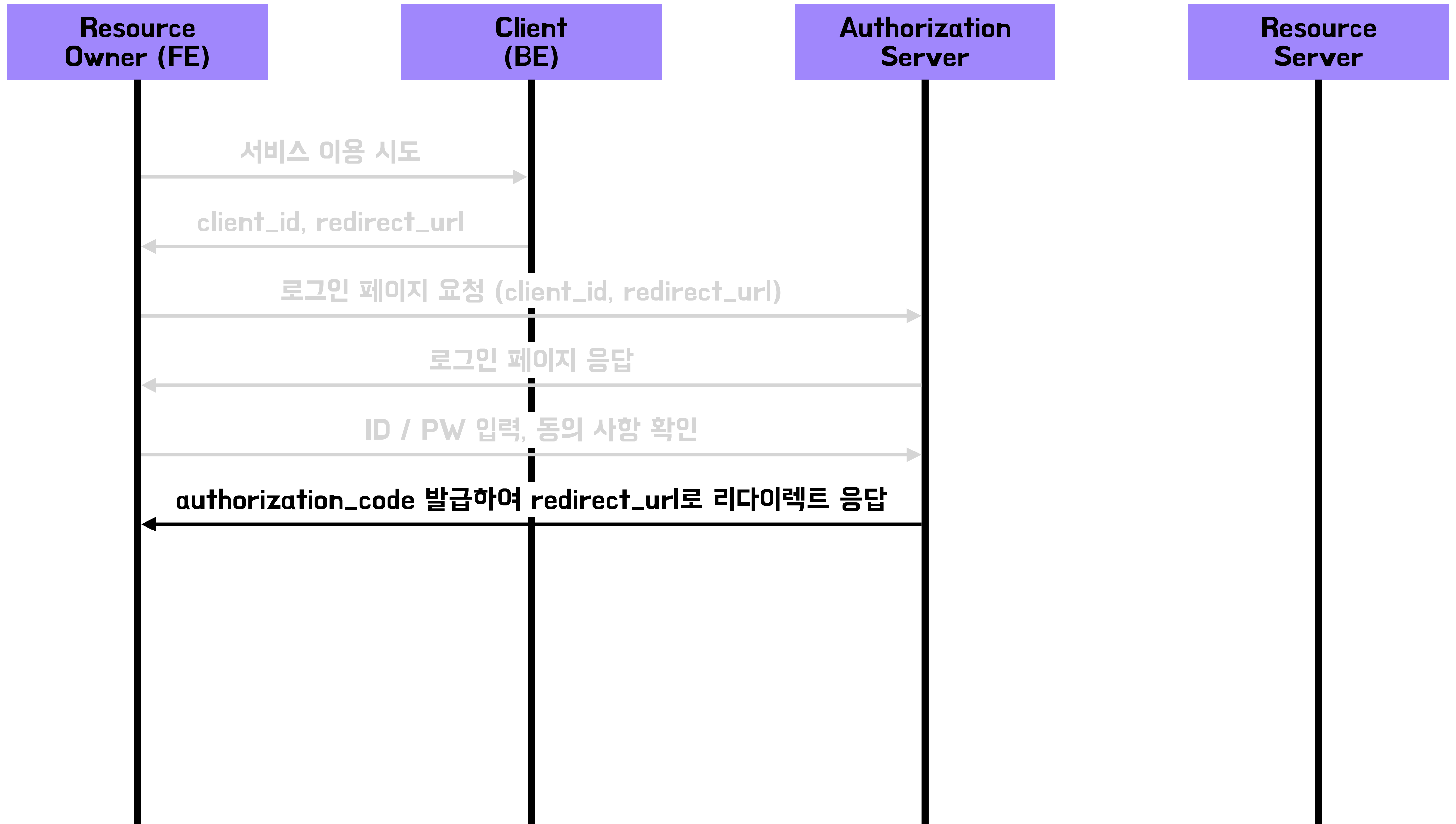


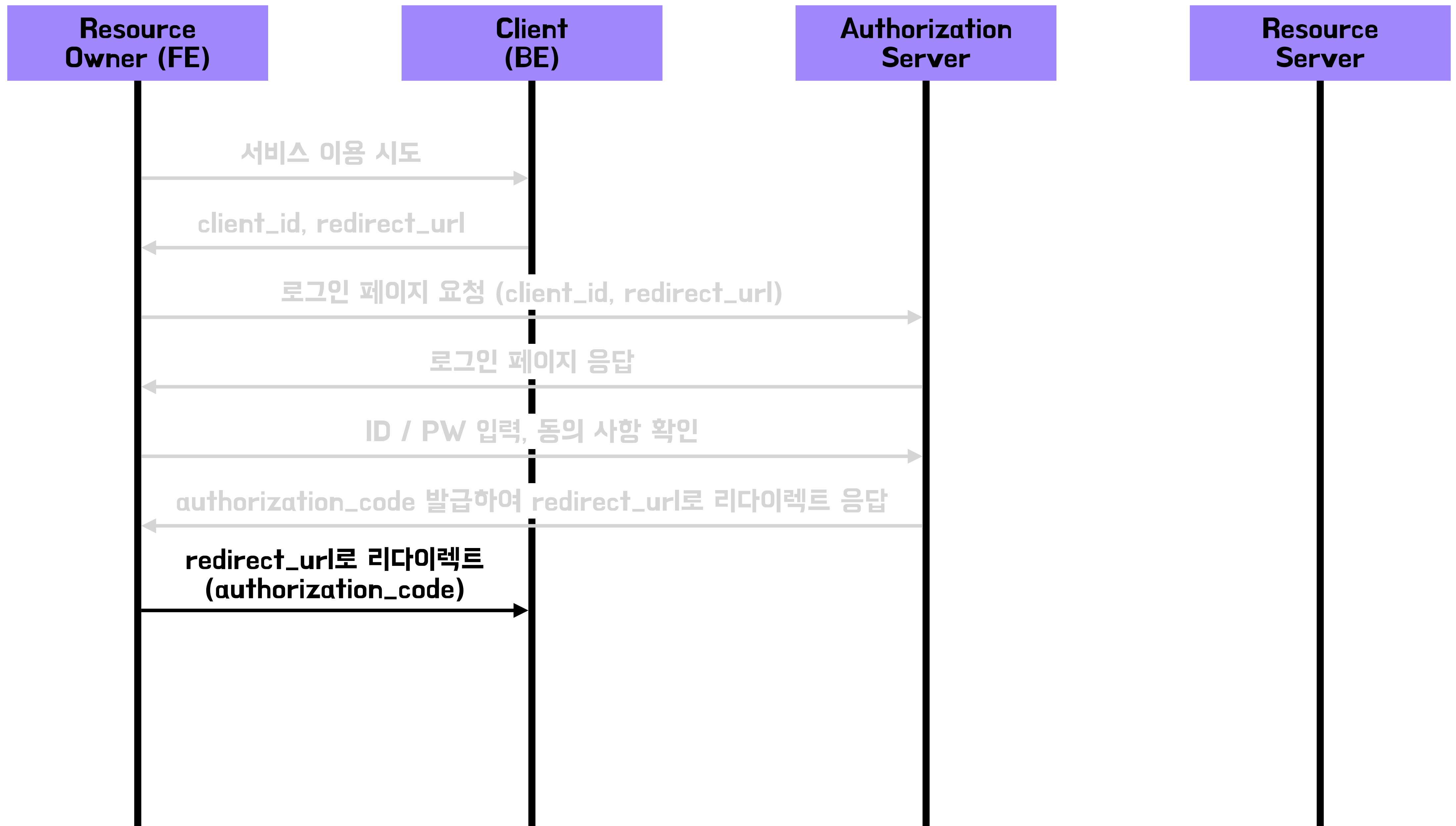


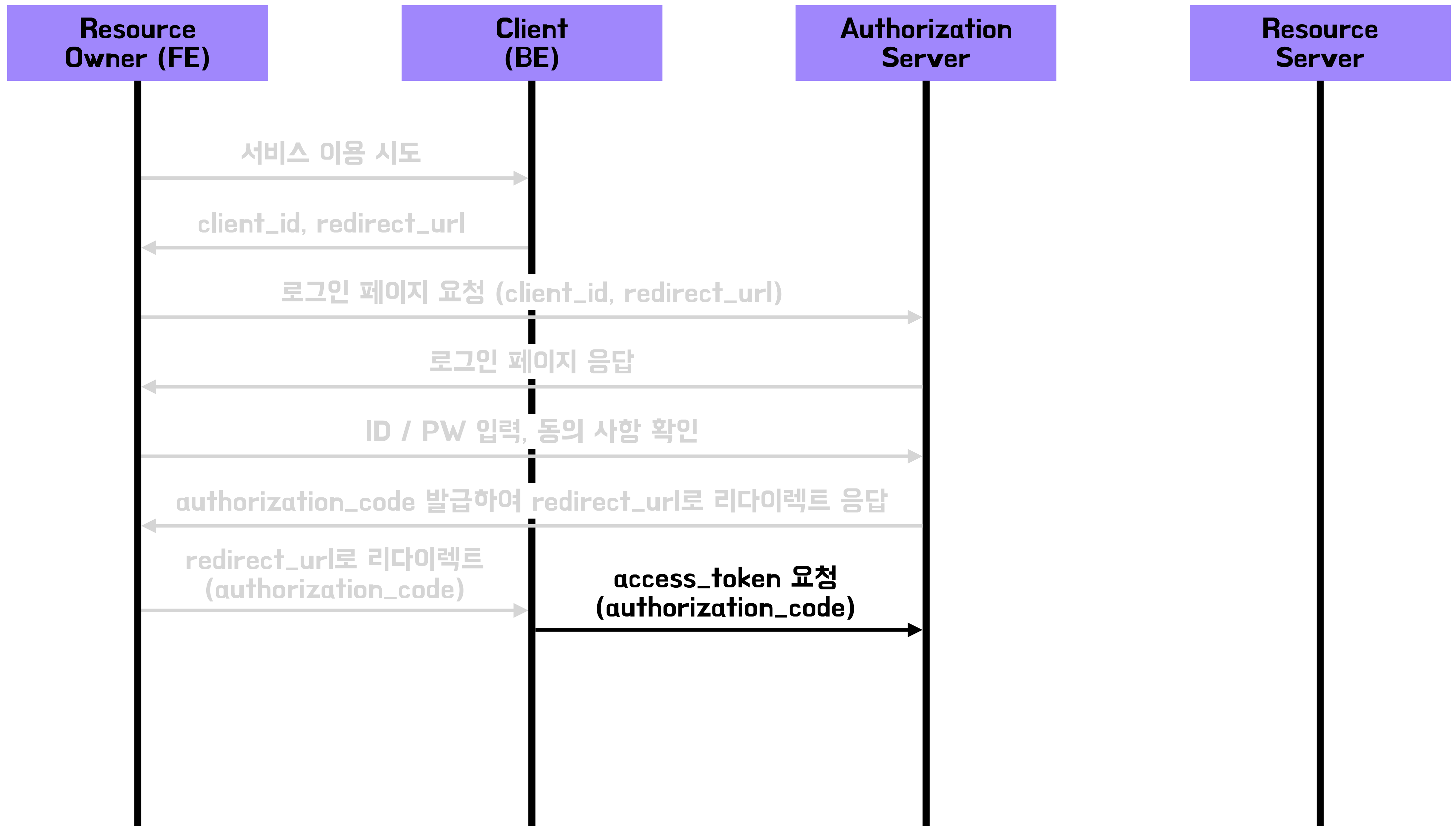


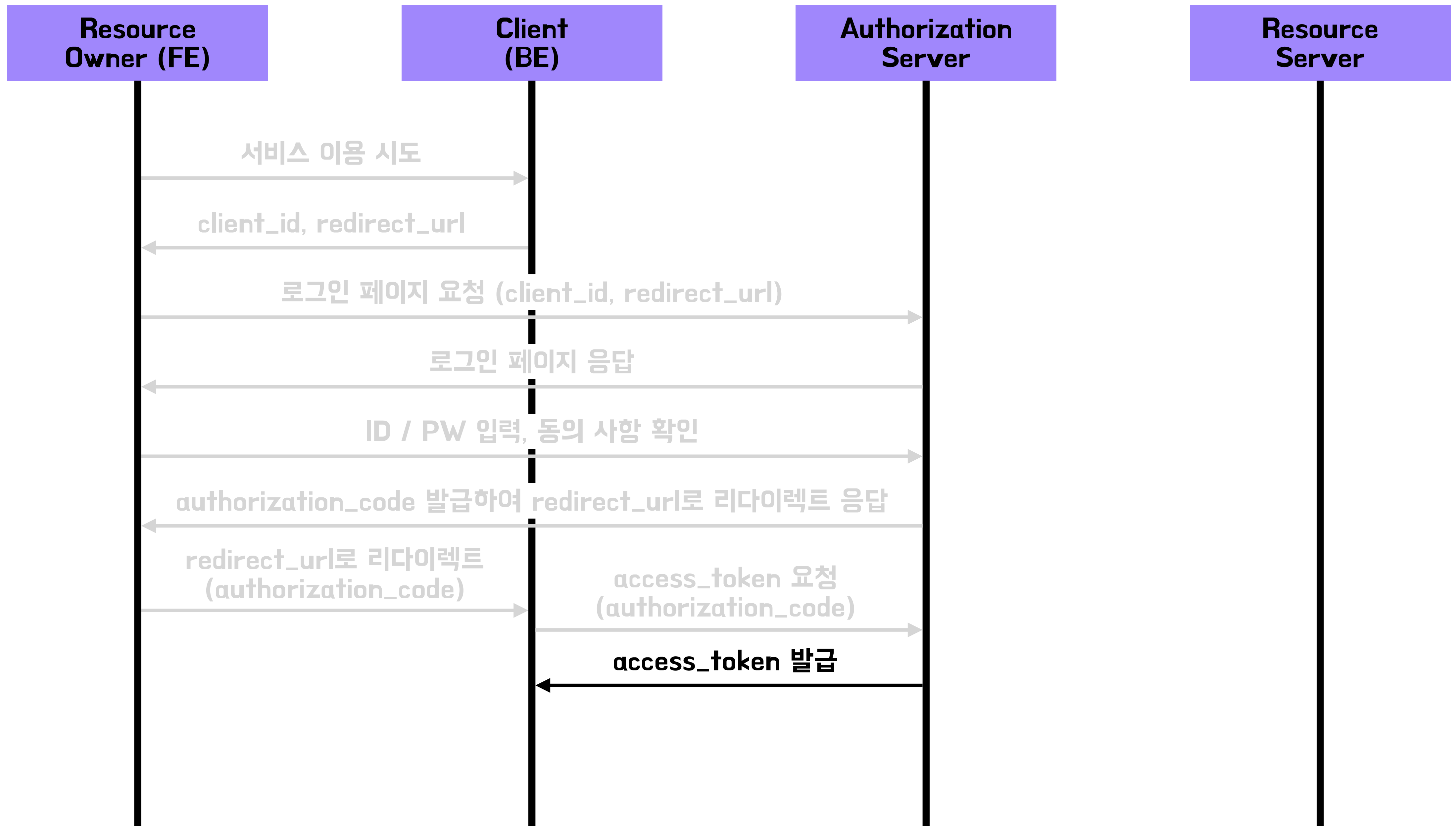


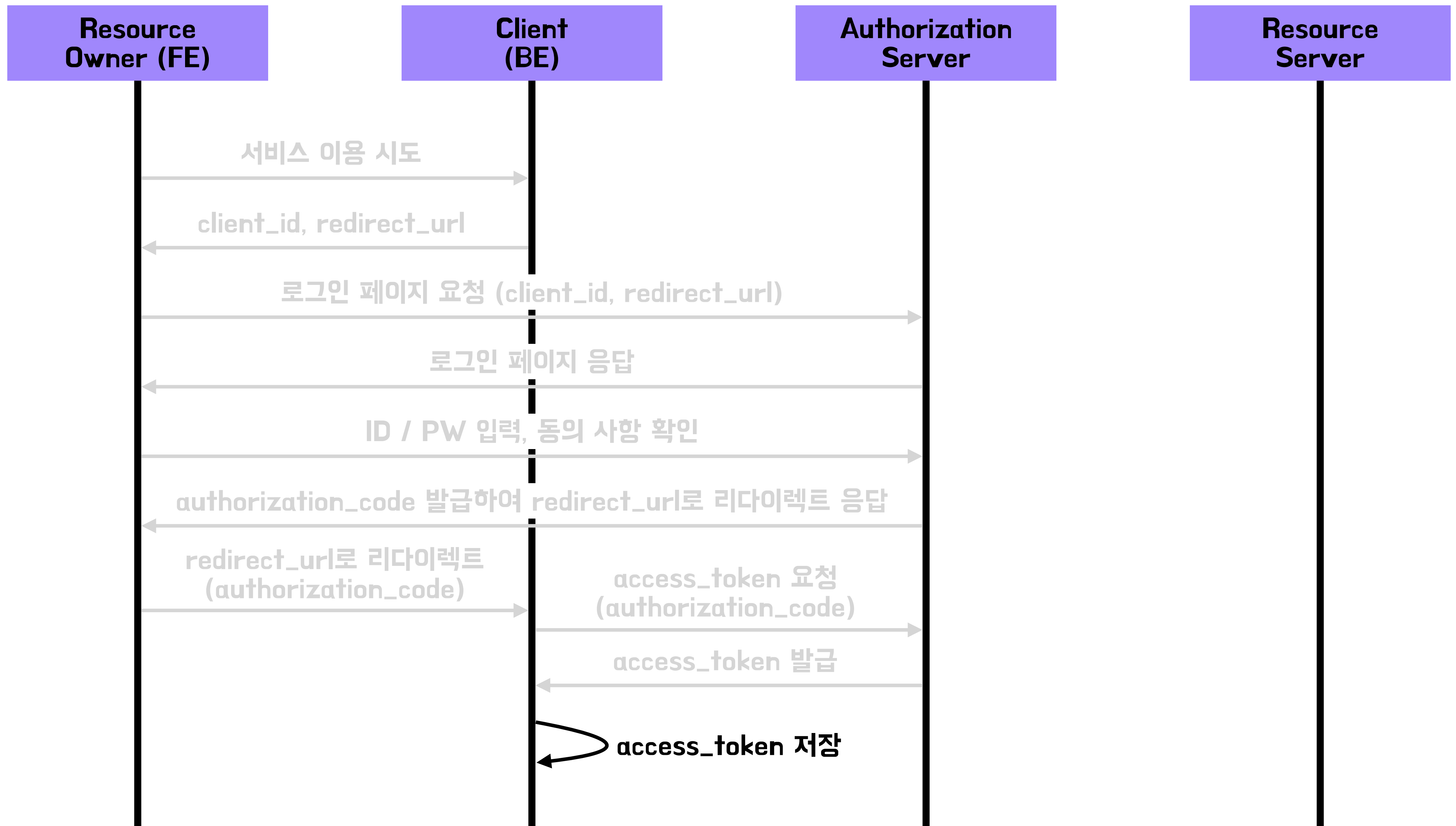


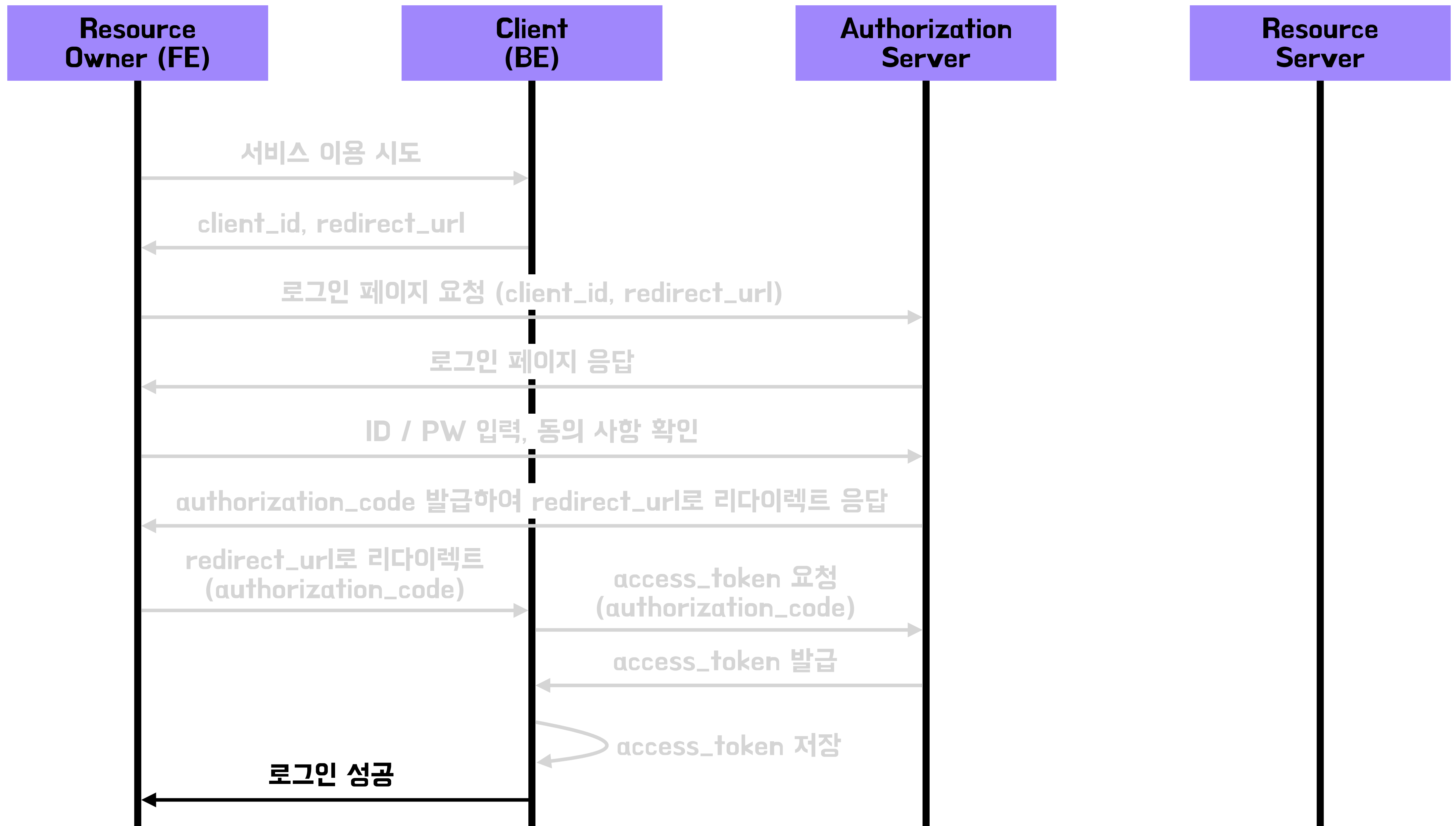












# OAuth 2.0 로그인 이후의 정보 조회

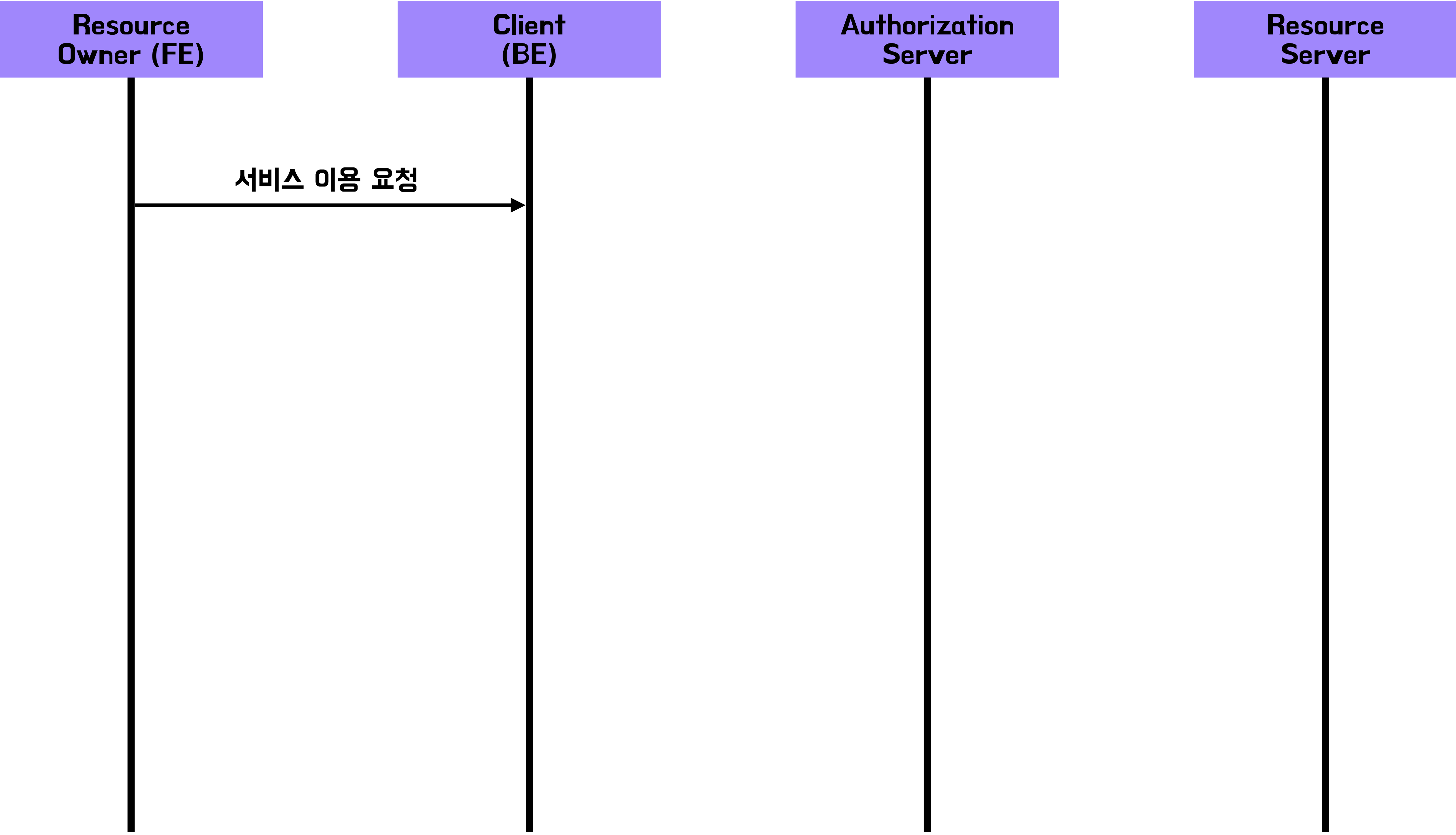
**Resource  
Owner (FE)**

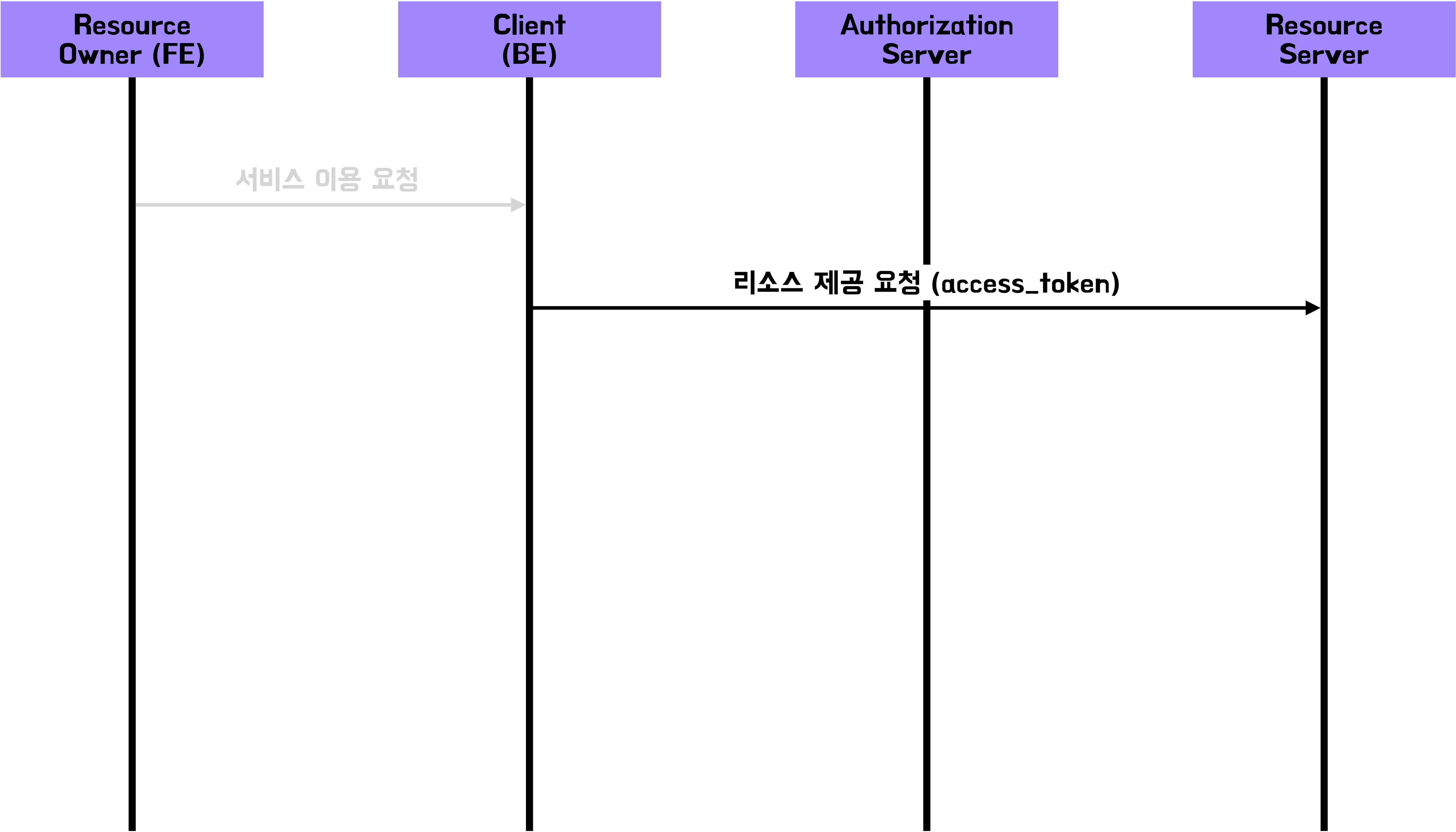
**Client  
(BE)**

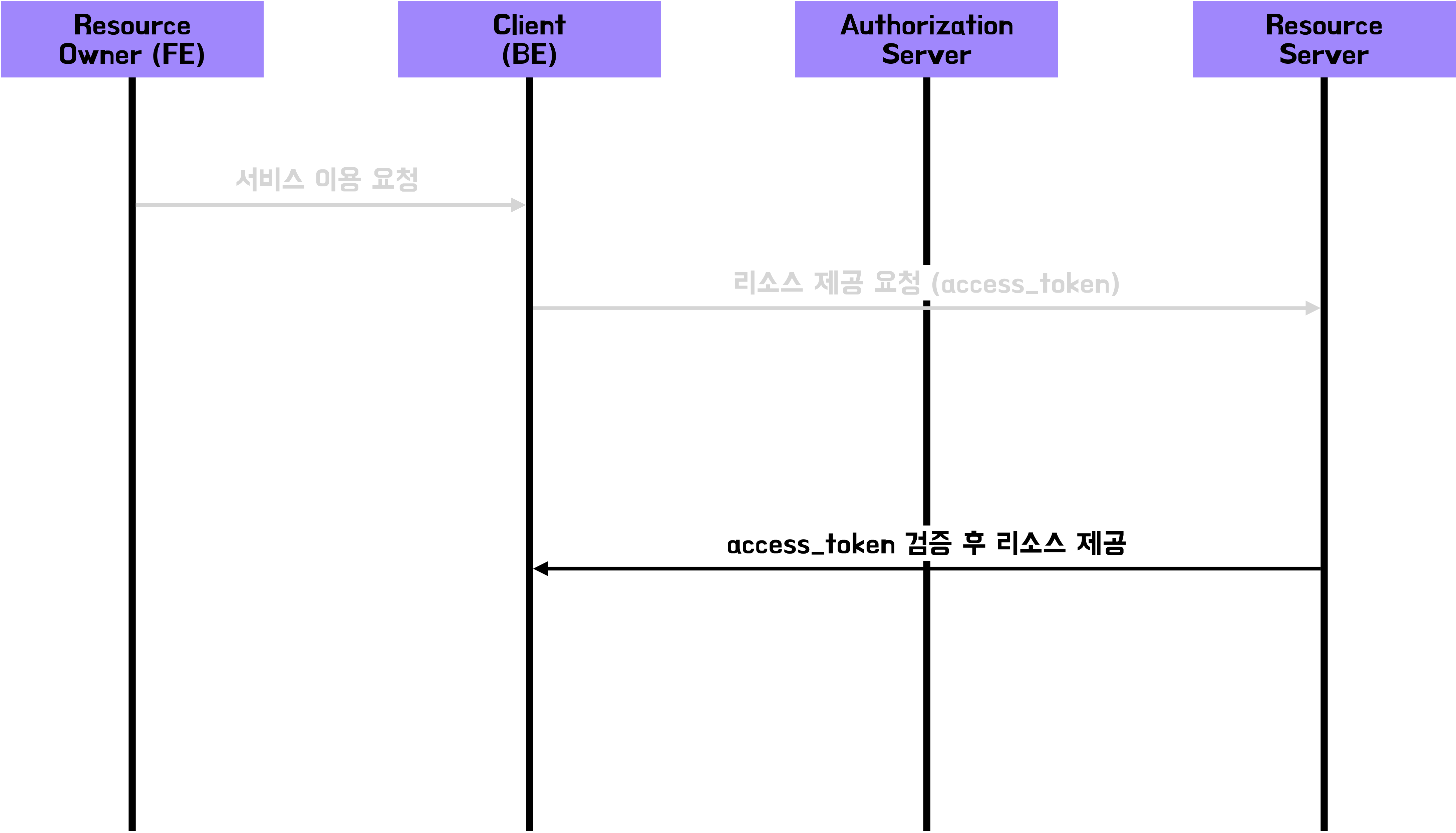
**Authorization  
Server**

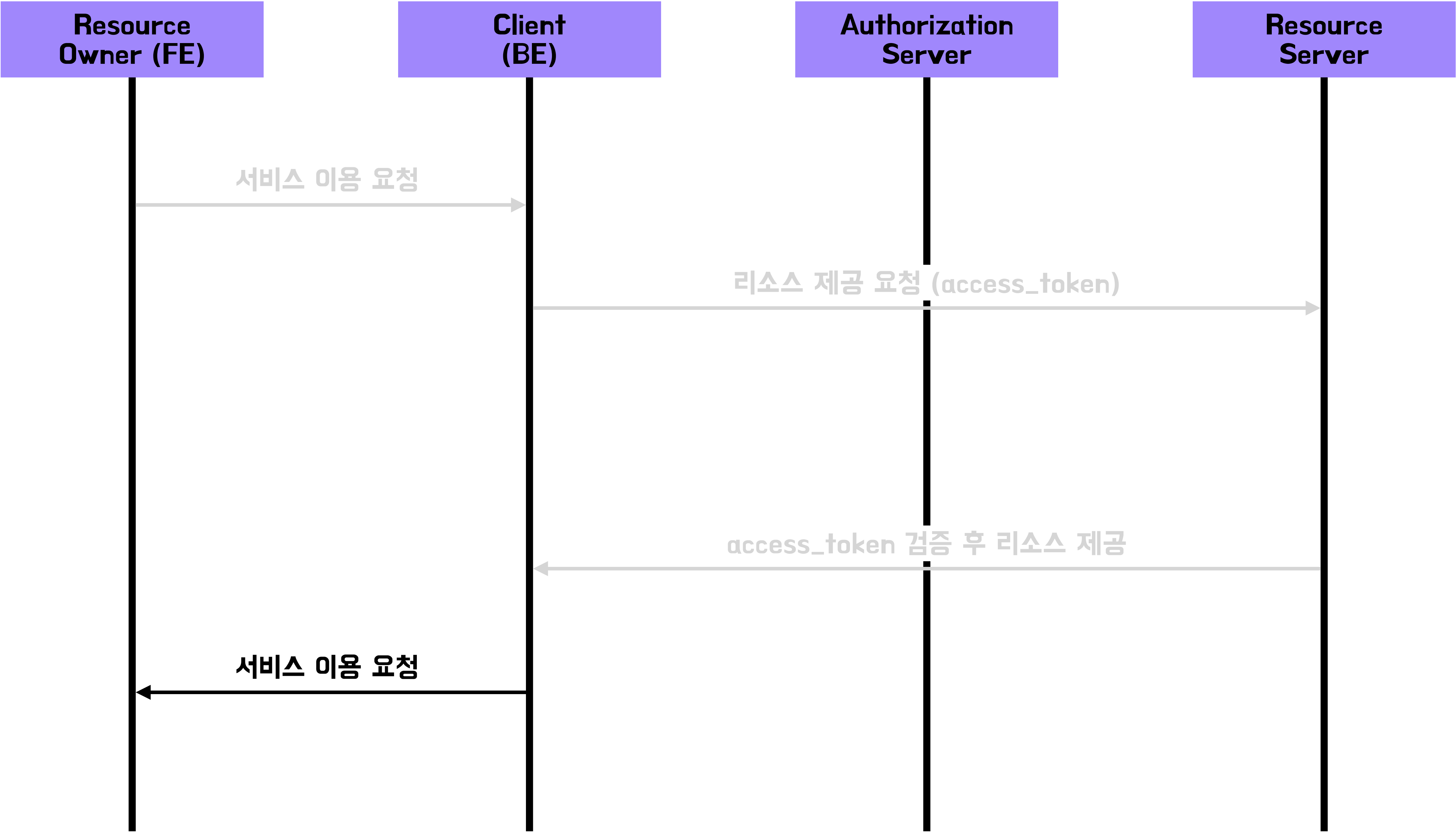
**Resource  
Server**











# OIDC

# OpenID Connect

OpenID Connect를 활성화하면 카카오 로그인 시 액세스 토큰과 ID 토큰을 함께 발급받을 수 있습니다. [앱 관리 페이지](#)의 [카카오 로그인] > [OpenID Connect]에서 아래 순서로 OpenID Connect를 사용하도록 설정할 수 있습니다.

## 사용 설정

[문서 확인](#)

카카오 로그인의 사용 여부를 선택할 수 있습니다.

상태 ☒ ON

**i** 가입한 회원의 개인정보 관리를 위해 웹훅(계정 상태 변경 웹훅(User Unlinked) 또는 연결 끊기 웹훅)을 설정해야 합니다. 미설정 시, 서비스 외부에서의 회원 탈퇴 정보를 전달받을 수 없어 개인정보 처리가 누락될 수 있습니다. [웹훅 바로가기](#)

## OpenID Connect

[문서 확인](#)

사용자 인증 정보가 담긴 ID 토큰을 추가로 발급받기 위한 OpenID Connect를 사용하도록 설정합니다.

상태 ☒ ON

# OpenID Connect

OpenID Connect를 활성화하면 카카오 로그인 시 액세스 토큰과 ID 토큰을 함께 발급받을 수 있습니다. [앱 관리 페이지](#)의 [카카오 로그인] > [OpenID Connect]에서 아래 순서로 OpenID Connect를 사용하도록 설정할 수 있습니다.

## 사용 설정

[문서 확인](#)

카카오 로그인 사용 여부를 선택할 수 있습니다.

상태 ☒ ON

**i** 가입한 회원의 개인정보 관리를 위해 웹훅(계정 상태 변경 웹훅(User Unlinked) 또는 연결 끊기 웹훅)을 설정해야 합니다. 미설정 시, 서비스 외부에서의 회원 탈퇴 정보를 전달받을 수 없어 개인정보 처리가 누락될 수 있습니다. [웹훅 바로가기](#)

## OpenID Connect

[문서 확인](#)

사용자 인증 정보가 담긴 ID 토큰을 추가로 발급받기 위한 OpenID Connect를 사용하도록 설정합니다.

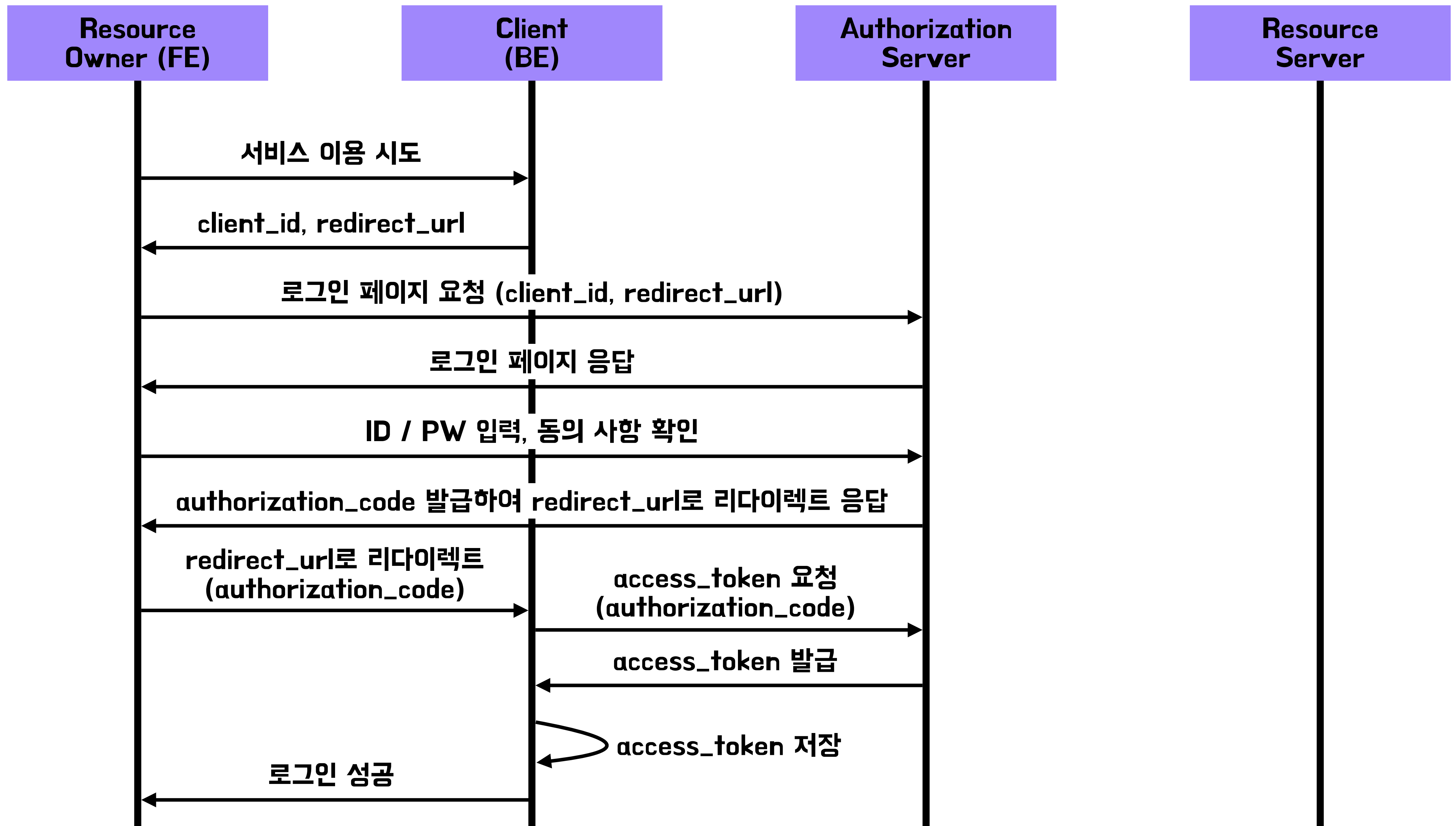
상태 ☒ ON

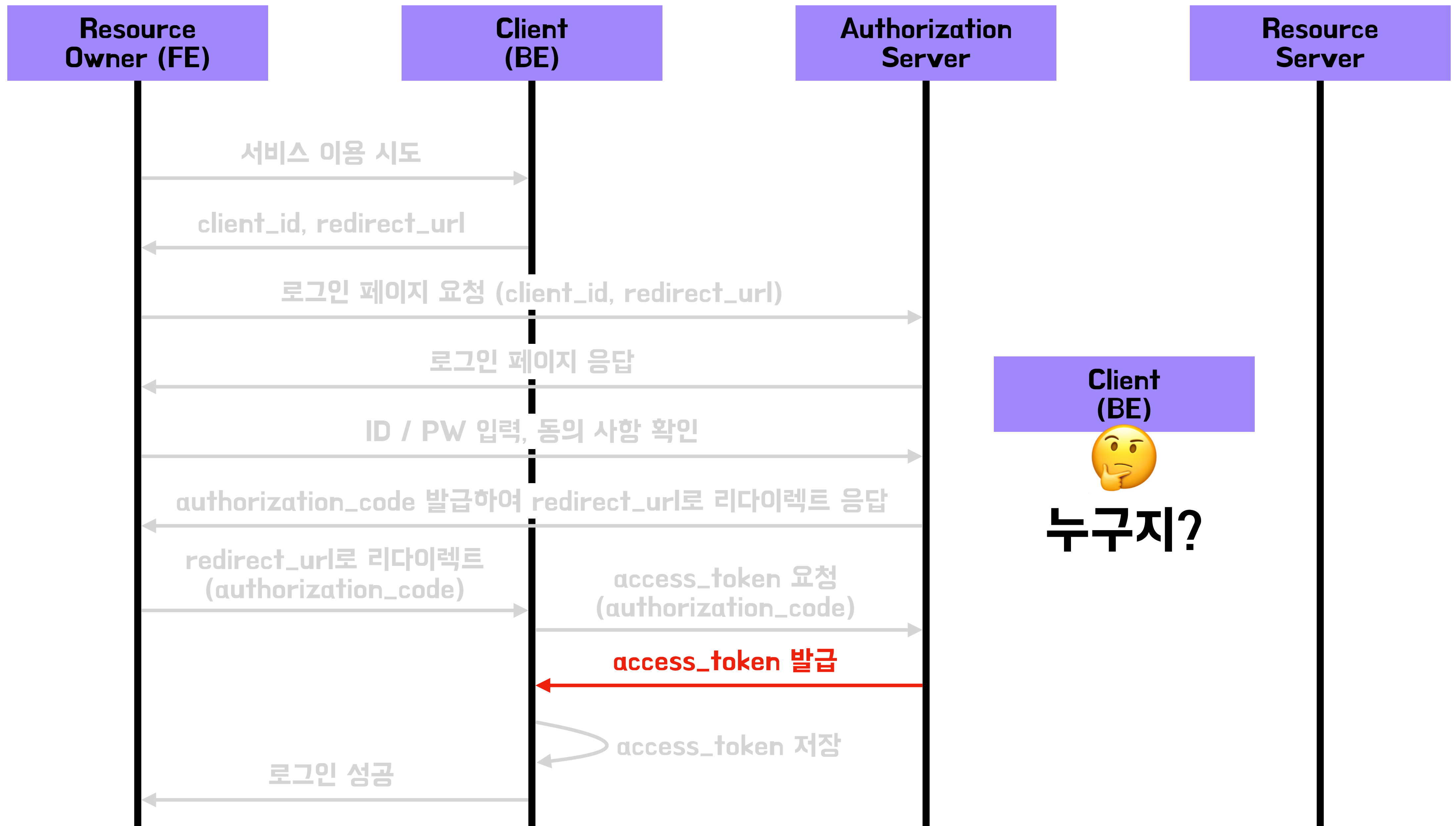
**로그인한 사용자가 누구인지 식별할 수 있다!**

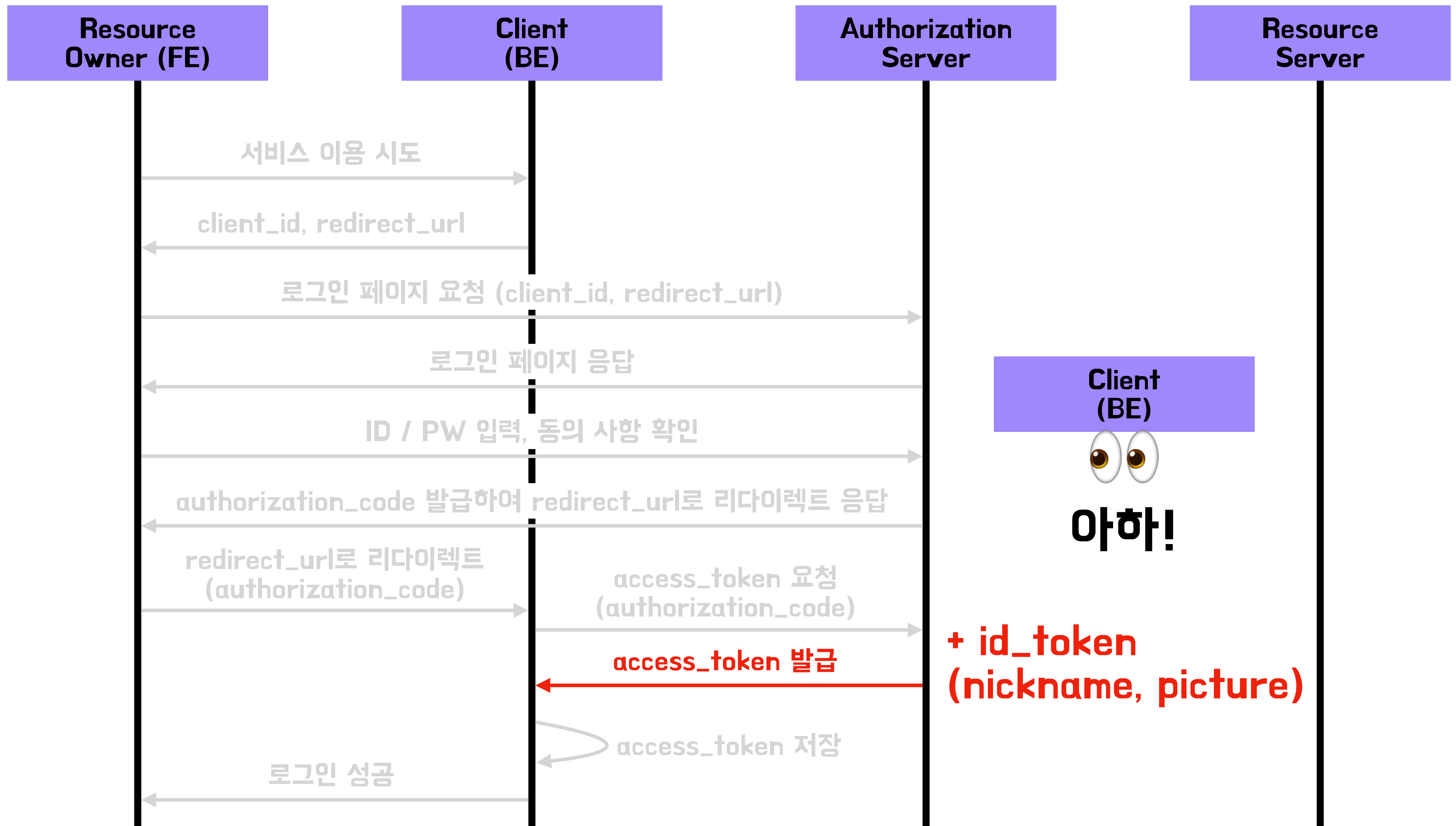


로그인한 사용자가 누구인지 식별할 수 있다!

**OAuth 2.0만으로는 사용자를 식별할 수 없다.**







```
{
  "access_token": "btZRMWhMuZqaaEsiYYUVzBQmz_qDX_BDAAAAAQoXFmIAAAGYf0Cf0yn2EFsnJsRZ",
  "token_type": "bearer",
  "refresh_token": "grlld4PhKacFvGv5zFibQ6h9jLAY7ufIAAAAAgoXFmIAAAGYf0CfNSn2EFsnJsRZ",
  "expires_in": 21599,
  "scope": "profile_image profile_nickname",
  "refresh_token_expires_in": 5183999
}
```

```
{
  "access_token": "btZRMWhMuZqaaEsiYYUVzBQmz_qDX_BDAAAAAQoXFmIAAAGYf0Cf0yn2EFsnJsRZ",
  "token_type": "bearer",
  "refresh_token": "grlld4PhKacFvGv5zFibQ6h9jLAY7ufIAAAAgoXFmIAAAGYf0CfNSn2EFsnJsRZ",

  "expires_in": 21599,
  "scope": "profile_image profile_nickname",
  "refresh_token_expires_in": 5183999
}
```

```
{
  "access_token": "btZRMWhMuZqaaEsiYYUVzBQmz_qDX_BDAAAAAQoXFmIAAAGYf0Cf0yn2EFsnJsRZ",
  "token_type": "bearer",
  "refresh_token": "grlld4PhKacFvGv5zFibQ6h9jLAY7ufIAAAAAgoXFmIAAAGYf0CfNSn2EFsnJsRZ",
  "id_token": "eyJraWQiOiI5ZjI1MmRhZGQ1ZjIzM2Y5M2QyZmE1MjhkMTJmZWElLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiI2M2EyMjYjYjU3MGY3YzU0YzUwYWRLNDRIZjVlZGQxOCIsInN1YiI6IjQzNzQ4MTYxMDkiLCJhdXRoX3RpbWUiOiE3NTQ0ODE1OTgsImIzcyI6Imh0dHBzOi8va2F1dGgua2FrYW8uY29tIiwibmIja25hbWUiOiLshqHshKDqtowiLCJleHAiOiE3NTQ1MDMxOTgsImIhdCI6MTc1NDQ4MTU5OCwicGljdHVyZSI6Imh0dHA6Ly9rLmtha2FvY2RuLm5ldC9kbi9ocFkzWi9idHNPQlA5N1d4RC91b3ZybzQ1V0I1Y25RdzltMkVDQlkwL2ltZ18xMTB4MTEwLmpwZyJ9\n.ahALrpm1IBBgIbEULoPM3WZXLXUZeh_sRKWdbw0jK1YV89guH0CeuzY1ghvabJyFeH\n-Ess_tjiQHyBcNgu01_140bKfCpazkdaBy00SIB0p5aPhgWfLY3nCsNex90C_qSjHcLJUW0_WVGNZJY9tjwZir\npotnkCLDDRktjar8f0FLTnJ3f9WjgUOTg2q981CqYNggIp26fGui6j70Rjhbi5oDb0fSbbYF0eZN0lv0qD7214\nPuBMbn5PVaLPBJaliW6s0BBEGONg\n-1ph6LwG_RUbfBzq67UBHWLUTj5pA_69yG4ExwQjV7a9yN9YejPgi8bgytPli2ceC5iIc0Upk30w",
  "expires_in": 21599,
  "scope": "profile_image profile_nickname",
  "refresh_token_expires_in": 5183999
}
```



JWT Decoder    JWT Encoder

Paste a JWT below that you'd like to decode, validate, and verify.

ENCODED VALUE

☐ Enable auto-focus

JSON WEB TOKEN (JWT) COPY CLEAR

Valid JWT

Signature Verified

eyJraWQiOiI1ZjI1MmRhZGQ1ZjIzM2Y5M2QyZmE1MjhkMTJmZWEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiI2M2EyMjJkYjU3MGY3YzU0YzUwYWwRNDRIZjVlZGQxOClIsInN1YiI6IjQzNzQ4MTYxMDkiLCJhdXRoX3RpbWUiOiJlZ3NTQ00DE10TgsImZcyI6Imh0dHBzOi8va2F1dGgua2FrYW8uY29tIiwibmIja25hbWUiOiLshqHshKDQtowiLCJleHAiOiJE3NTQ1MDMxOTgsImIhdCI6MTc1NDQ4MTU5OCwicGljdHVyZSI6Imh0dHA6Ly9rLmtha2FvY2RuLm5ldC9kbi9ocFkzWi9idHNPQlA5N1d4RC91b3Zyb3Q1V0I1Y25RdzltMkVdQlkwL2ltZ18xMTB4MTEwLmpwZyJ9.ahALrpm1IBBgIbEULoPM3WZXLXUzeh\_sRKWdbw0jK1YV89guH0CeuZY1ghvabJyFeH-Ess\_tjiQHyBcNgu01\_140bKfCpqzkdaBy00SIB0p5aPhgWfLY3nCsNex90C\_qSjHcLJUW0\_WVGNZJY9tjwZirpotnkCLDDRktjar8f0FLTnJ3f9WjgU0Tg2q981CqYNggIp26fGui6j70Rjhbi5oDb0fSbbYF0eZN0lv0qD7214PuBMbn5PVaLPBJaliW6s0BBEG0Ng-1ph6LwG\_RUBfBzq67UBHWLUTj5pA\_69yG4ExwQjV7a9yN9YejPgi8bgytPli2ceC5iIc0Upk30w

Generate example

DECODED HEADER

JSON CLAIMS TABLE COPY ↗

{  
 "kid": "9f252dadd5f233f93d2fa528d12fea",  
 "typ": "JWT",  
 "alg": "RS256"  
}

DECODED PAYLOAD

JSON CLAIMS TABLE COPY ↗

{  
 "aud": "63a222db570f7c54c50ade44bf5edd18",  
 "sub": "4374816109",  
 "auth\_time": 1754481598,  
 "iss": "https://kauth.kakao.com",  
 "nickname": "송선권",  
 "exp": 1754503198,  
 "iat": 1754481598,  
 "picture": "http://k.kakaocdn.net/dn/hpY3Z/bts0BP97WxD/uovro45WB5cnQw9m2ECBY0/img\_110x110.jpg"  
}

JWT SIGNATURE VERIFICATION (OPTIONAL)

Enter the public key used to sign the JWT below:

PUBLIC KEY COPY CLEAR

Valid public key

{  
 "e": "AQAB",  
 "ktv": "RSA".  
}

Public Key Format JWK ▼



## DECODED PAYLOAD

JSON

CLAIMS TABLE

COPY



```
{
  "aud": "63a222db570f7c54c50ade44bf5edd18",
  "sub": "4374816109",
  "auth_time": 1754481598,
  "iss": "https://kauth.kakao.com",
  "nickname": "송선권",
  "exp": 1754503198,
  "iat": 1754481598,
  "picture": "http://k.kakaocdn.net/dn/hpY3Z/bts0BP97WxD/uovro45WB5cnQw9m2ECBY0/img_110x110.jpg"
}
```

사용자 식별자

이름

프로필 사진

## DECODED PAYLOAD

JSON

CLAIMS TABLE

COPY



```
{
  "aud": "63a222db570f7c54c50ade44bf5edd18",
  "sub": "4374816109",
  "auth_time": 1754481598,
  "iss": "https://kauth.kakao.com",
  "nickname": "송선권",
  "exp": 1754503198,
  "iat": 1754481598,
  "picture": "http://k.kakaocdn.net/dn/hpY3Z/bts0BP97WxD/uovro45WB5cnQw9m2ECBY0/img_110x110.jpg"
}
```

# 자체 인증 체계

# 서비스 인증 체계

```
{
  "access_token": "btZRMWhMuZqaaEsiYYUVzBQmz_qDX_BDAAAAAQoXFmIAAAGYf0Cf0yn2EFsnJsRZ",
  "token_type": "bearer",
  "refresh_token": "grlld4PhKacFvGv5zFibQ6h9jLAY7ufIAAAAAgoXFmIAAAGYf0CfNSn2EFsnJsRZ",
  "id_token": "eyJraWQiOiI5ZjI1MmRhZGQ1ZjIzM2Y5M2QyZmE1MjhkMTJmZWElLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiI2M2EyMjYyU0YzUwYWw1NDRIZjVlZGQxOCIsInN1YiI6IjQzNzQ4MTYxMDkiLCJhdXRoX3RpbWUiOiJlZ3NTQ0ODE1TgsImIzcyI6Imh0dHBzOi8va2F1dGgua2FrYW8uY29tIiwibmIja25hbWUiOiLshqHshKDqtowiLCJleHAiOiJlZ3NTQ1MDMxOTgsImIhdCI6MTc1NDQ4MTU5OCwicGljdHVyZSI6Imh0dHA6Ly9rLmtha2FvY2RuLm5ldC9kbi9ocFkzWi9idHNPQlA5N1d4RC91b3ZybzQ1V0I1Y25RdzltMkVDQlkwL2ltZ18xMTB4MTEwLmpwZyJ9\n.ahALrpm1IBBg1bEULoPM3WZXLXUZeh_sRKWdbw0jK1YV89guH0CeuZY1ghvabJyFeH\n-Ess_tjiQHyBcNgu01_140bKfCpqzkdaBy00SIB0p5aPhgWfLY3nCsNex90C_qSjHcLJUW0_WVGNZJY9tjwZir\npotnkCLDDRktjar8f0FLTnJ3f9WjgUOTg2q981CqYNggIp26fGui6j70Rjhbi5oDb0fSbbYF0eZN0lv0qD7214\nPuBMbn5PVaLPBJaliW6s0BBEGONg\n-1ph6LwG_RUbfBzq67UBHWLUTj5pA_69yG4ExwQjV7a9yN9YejPgi8bgytPli2ceC5iIc0Upk30w",
  "expires_in": 21599,
  "scope": "profile_image profile_nickname",
  "refresh_token_expires_in": 5183999
}
```

**액세스 토큰과 리프레시 토큰?  
그냥 이걸로 할까..?**



# 서비스 인증 체계

```
{
  "access_token": "btZRMWhM...YUVzBQmz_qDX_BDAAAA...f0yn2EFsnJsRZ",
  "token_type": "bearer",
  "refresh_token": "grlld4...bQ6h9jLAY7ufT...Yf0CfNSn2EFsnJsRZ",
  "id_token": "eyJraWQiOiI5...jIzM2Y5M2...EiLCJ0eXAiOiJKV1QiLCJhbGc...
    iOiJSUzI1NiJ9.eyJhdWQiOi...BMGY3Y...ZjVlZGQxOCIsInN1YiI6IjQzNzQ4
    MTYxMDkiLCJhdXRoX3RpbWUiOi...sIm...i8va2F1dGgua2FrYW8uY29tIiwibm
    lja25hbWUiOiLshqHshKDqtow...CI6MTc1NDQ4MTU5OCwicGljdHVyZSI6
    Imh0dHA6Ly9rLmtha2FvY2RuLm...Id4RC91b3ZybzQ1V0I1Y25RdzltMkVDQl
    kWL2ltZ18xMTB4MTEwLmpwZyJ9
    .ahALrpm1IBBg1bEULoPM3WZXLXU...h0CeuzY1ghvabJyFeH
    -Ess_tjiQHyBcNgu01_140bKfCpqzk...LY3nCsNex90C_qSjHcLJUW0_WVGNZJY9tjwZir
    potnkCLDDRktjar8f0FLTnJ3f9Wjgl...fGui6j70Rjhbi5oDb0fSbbYF0eZN0lv0qD7214
    PuBMbn5PVaLPBJaliW6s0BBEGONa...
    -1ph6LwG_RUbfBzq67UBHWLUTj...gi8bgytPli2ceC5iIc0Upk30w",
  "expires_in": 21599,
  "scope": "profile_image",
  "refresh_token_expires_in": 21599
}
```

이 토큰과 리프레시 토큰?  
그냥 이걸로 알까..?

# 카카오의 토큰을 사용할 수 없는 이유

## 1. 만료된 액세스 토큰

1. 우리 서비스는 최초 카카오 로그인 이후 사용자 정보를 수집하지 않는다.
2. 카카오에 요청을 보내지 않는다.
3. 발급받은 액세스 토큰이 만료되었는지 여부를 알 수 없다.

## 2. 카카오의 토큰

1. 우리 서비스는 카카오의 액세스/리프레시 토큰에 의존해야 한다.
2. 토큰이 노출되어도 우리가 할 수 있는 일이 없다.

# 카카오의 토큰을 사용할 수 없는 이유

## 1. 만료된 액세스 토큰

1. 우리 서비스는 최초 카카오 로그인 이후 사용자 정보를 수집하지 않는다.
2. 카카오에 요청을 보내지 않는다.
3. 발급받은 액세스 토큰이 만료되었는지 여부를 알 수 없다.

## 2. 카카오의 토큰

1. 우리 서비스는 카카오의 액세스/리프레시 토큰에 의존해야 한다.
2. 토큰이 노출되어도 우리가 할 수 있는 일이 없다.

우리 서비스는 **만료된 카카오의** 토큰을 사용해야 한다.



**세션으로도 충분하지 않을까?**  
**아 JWT는 쓰기 싫은데... (홍대병)**



# JWT를 거부한 이유

JWT는 **오버 엔지니어링** 아닐까!? (절대 귀찮은 거 맞음)

# JWT를 거부한 이유

JWT는 **오버 엔지니어링** 아닐까!? (절대 귀찮은 거 맞음)

1. JWT 써도 리프레시 토큰은 서버에 저장해야 한다면.. 세션과 다를 바 있나?
2. 어차피 단일 서버만 운영함이 명백하다면.. 세션과 다를 바 있나?
3. JWT는 의존성과 복잡한 검증 로직이 필요한데.. 세션은 편하잖아?
4. 서버에서 정보를 관리하니 토큰이 조작될 우려도 없고.. 좋은데?


# 까짓 거 그냥 세션으로 해버리자

ㅋㅋ

# 스프링 세션 지속시간

◆ AI 개요

 +11 

스프링 세션의 기본 지속 시간은 **30분**입니다. 이 시간은 **사용자가 아무런 동작을 하지 않아도 세션이 유지되는 시간**을 의미하며, 사용자가 서비스를 이용하는 동안에는 세션 만료 시간이 계속 갱신됩니다. 즉, **사용자가 30분 이내에 사이트에서 활동을 하면 세션이 만료되지 않고 계속 유지됩니다.** 

세션 지속 시간 설정 방법:

[application.properties](#) 설정:

- `server.servlet.session.timeout` 속성을 사용하여 세션 지속 시간을 설정할 수 있습니다. 예를 들어, `server.servlet.session.timeout=600` 은 600초(10분)으로 설정하는 것입니다. 

- 단위는 초이며, 60초보다 작은 값은 설정할 수 없습니다. 

더보기 

# **그럼 30분마다 카카오 재로그인을 해야하나?**

**아... 사실상 재접속 할때마다 카카오 로그인을 다시 시켜야 한다고? 오반데...**

# 그럼 30분마다 카카오 재로그인을 해야하나?

아... 사실상 재접속 할때마다 카카오 로그인을 다시 시켜야 한다고? 오반데...

**리프레시 토큰**을 도입하자!

# 그럼 30분마다 카카오 재로그인을 해야하나?

아... 사실상 재접속 할때마다 카카오 로그인을 다시 시켜야 한다고? 오반데...

**리프레시 토큰**을 도입하자!

어떻게? 이걸 세션에 못넣는데? (세션이 30분마다 만료되니)

# 그럼 30분마다 카카오 재로그인을 해야하나?

아... 사실상 재접속 할때마다 카카오 로그인을 다시 시켜야 한다고? 오반데...

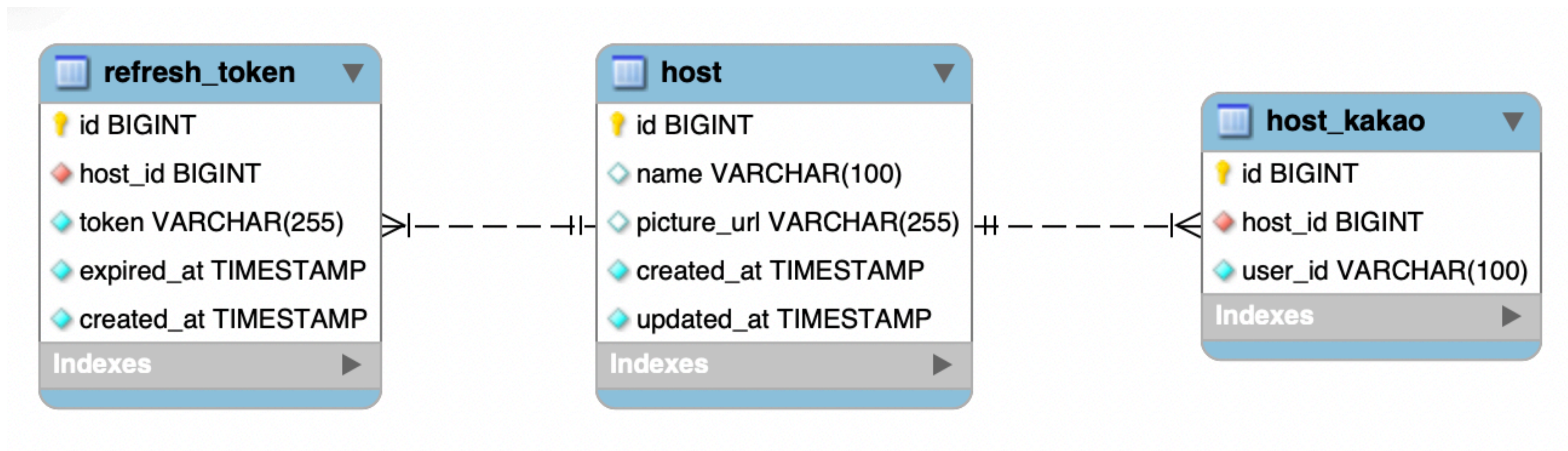
**리프레시 토큰**을 도입하자!

어떻게? 이건 세션에 못넣는데? (세션이 30분마다 만료되니)

그럼 그냥 DB에 넣지뿔~ (영속화 필요, 자주 접근 x)



# 테이블 구조



```

@GetMapping("/login/kakao/callback") 송선권
@Operation(summary = "Kakao 로그인 콜백",
description = "Kakao 로그인 후 리다이렉트되는 URL에서 authorization code를 받아 로그인 처리를 합니다.")
public ResponseEntity<Void> kakaoLoginCallback(
    @RequestParam(name = "code") String authorizationCode,
    HttpServletResponse httpServletResponse,
    HttpSession session
){
    var response = authService.requestKakaoLoginToken(authorizationCode);
    session.setAttribute("host_id", response.hostId());
    ResponseCookie refreshToken = createRefreshCookie(response.refreshToken(), response.expirationDays());
    httpServletResponse.addHeader(HttpHeaders.SET_COOKIE, refreshToken.toString());
    return ResponseEntity.ok().build();
}

private ResponseCookie createRefreshCookie(String refreshTokenValue, long expirationDays) { 2개 사용 위치
    ResponseCookie refreshToken = ResponseCookie.from("refresh_token", refreshTokenValue)
        .httpOnly(true)
        .secure(true)
        .sameSite("Strict")
        .path("/auth")
        .maxAge(maxAgeSeconds: 60 * 60 * 24 * expirationDays)
        .build();

    return refreshToken;
}

```

**세션에 host\_id 설정**

**쿠키로 리프레시 토큰 제공  
(여러 옵션으로 강건성 UP!)**

# 실제 응답 확인

×	헤더	페이로드	미리보기	응답	시작점	타이밍	쿠키		
쿠키 요청 <input type="checkbox"/> 필터링을 통해 제외된 요청 쿠키 표시									
이름	값	Domain	Path	Expires / Max-...	크기	HttpOnly	Secure	SameSite	Pa
JSESSIONID	F929D1067EC...	api.dev.forgath...	/	세션	42	✓			
응답 쿠키									
이름	값	Domain	Path	Expires / Max-...	크기	HttpOnly	Secure	SameSite	Pa
JSESSIONID	4F653252720...	api.dev.forgath...	/	세션	62	✓			
refresh_token	b11044f8e913...	api.dev.forgath...	/auth	90 일	181	✓	✓	Strict	

**JSESSIONID:** host\_id가 담긴 세션 ID  
**refresh\_token:** 리프레시 토큰 쿠키

# 리프레시 토큰 삭제

**DB에서 관리하다보니 TTL 지정 불가능**



# 리프레시 토큰 삭제

매일 새벽마다  
만료된 토큰 일괄 파기

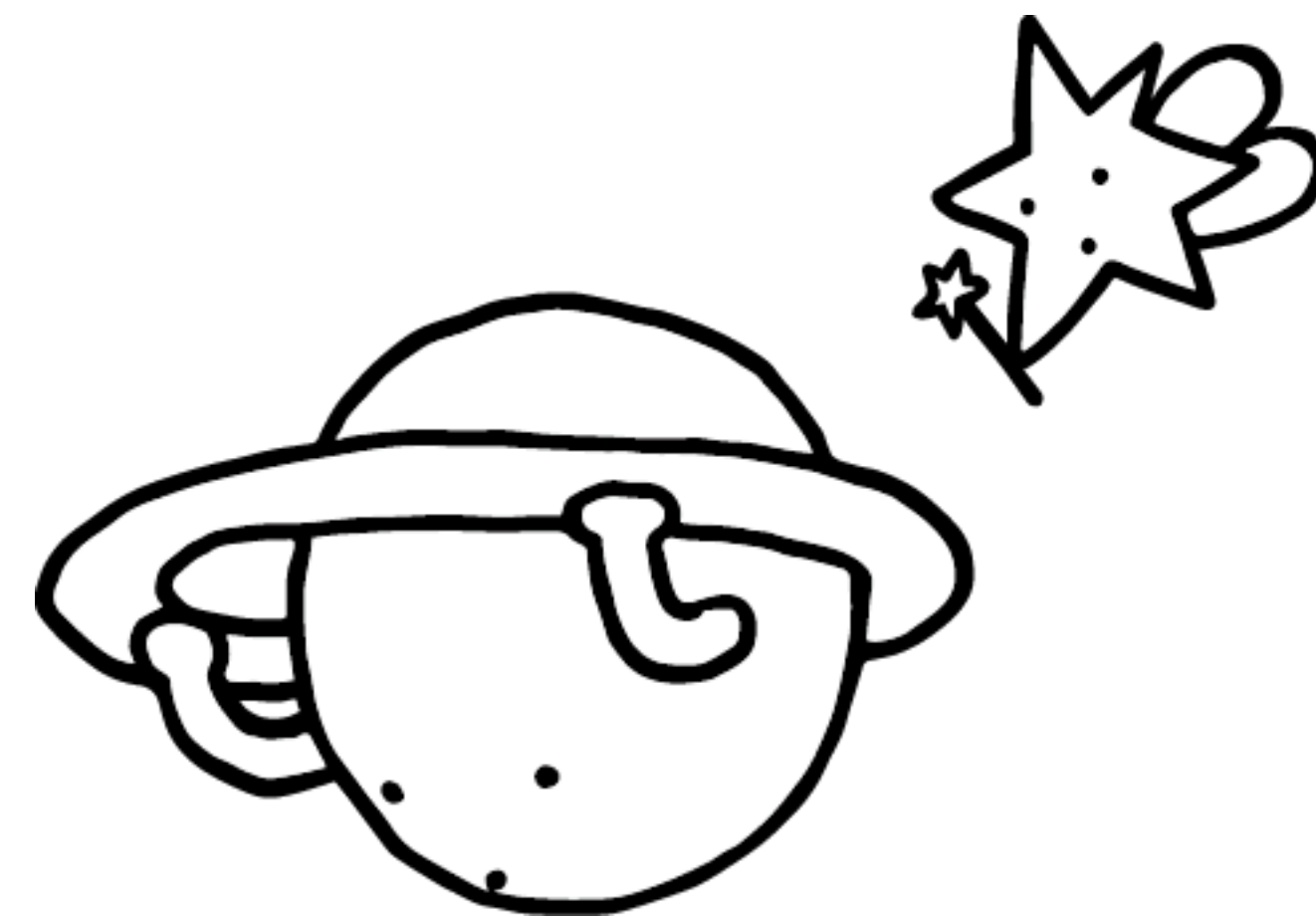
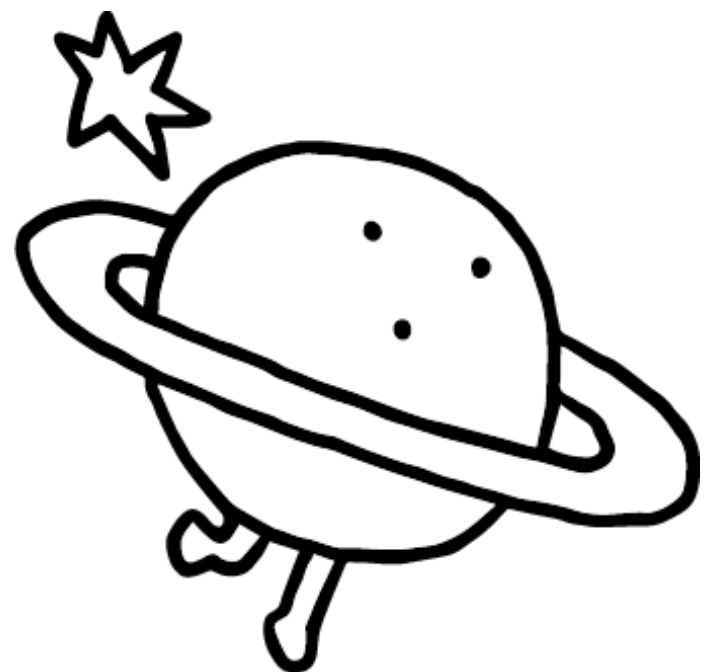
로그아웃 시 토큰 제거

```
@Component ㄹ 송선권
@RequiredArgsConstructor
public class RefreshTokenRemoveScheduler {

    private final AuthService authService;

    @Scheduled(cron = "0 0 3 * * *") // 매일 새벽 3시 실행 ㄹ 송선권
    public void removeExpiredRefreshTokens() {
        authService.removeExpiredRefreshTokens();
    }
}
```

```
@Transactional 1개 사용 위치 ㄹ 송선권
public void logout(String refreshToken) {
    refreshTokenRepository.findByToken(refreshToken)
        .ifPresent(refreshTokenRepository::delete);
}
```



감사함둥

