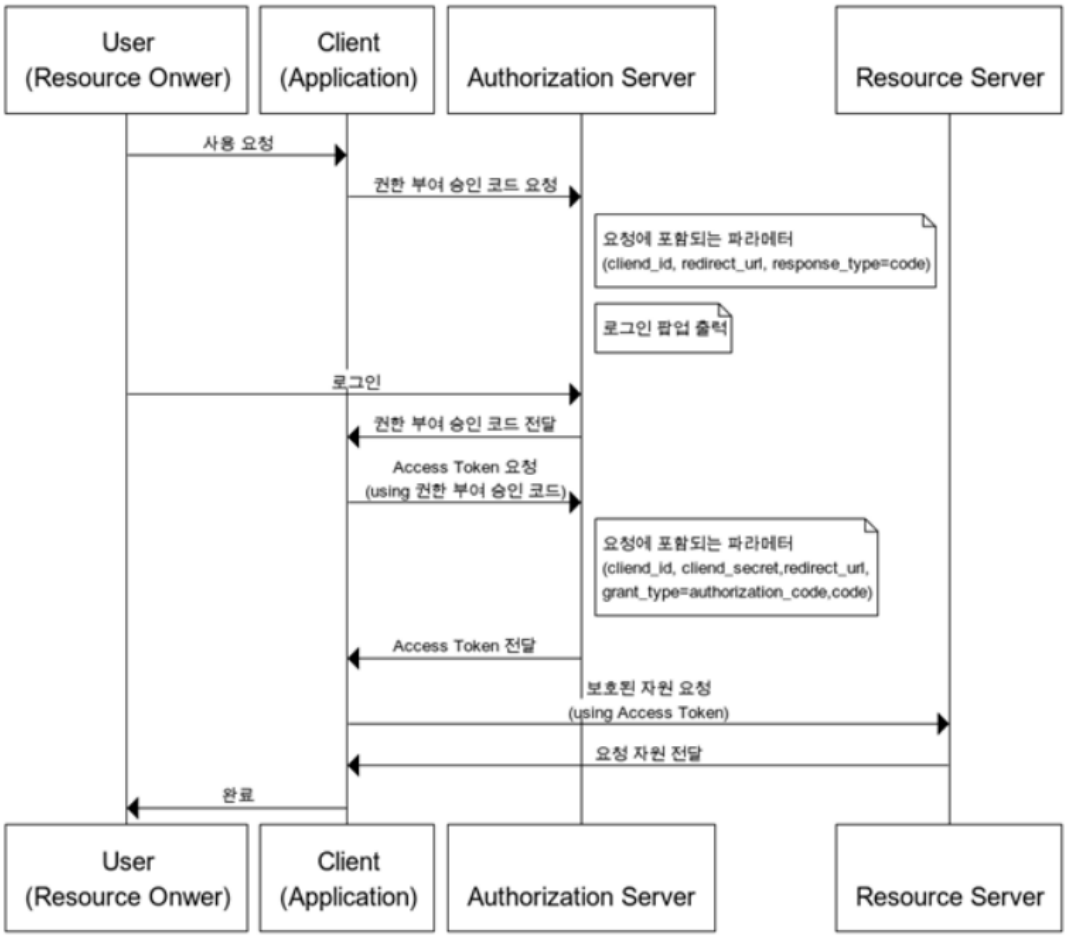
OAuth 2.0

1. 용어 정리
    - Resource Owner
        - 리소스 소유자
        - 본인의 정보에 접근할 수 있는 자격을 승인하는 주체
        - 클라이언트를 인증(Authroize)하는 역할을 수행
        - 인증이 완료되면 동의를 통해 권한 획득 자격을 클라이언트에게 부여
        - **사용자**
    - Client
        - Resource Owner의 리소스를 사용하고자 접근 요청을 하는 어플리케이션
    - Resource Server
        - Resource Owner의 정보가 저장되어 있는 서버
        - **구글, 페이스북**
    - Authorization Server
        - 권한 서버 = 인증/인가 를 수행하는 서버
        - 클라이언트의 접근 자격을 확인하고 access token을 발급하여 권한을 부여하는 역할 수행
    - Authentication(인증)
        - 인증, 접근 자격이 있는지 검증하는 단계
    - Authorization(인가)
        - 자원에 접근할 권한을 부여하고 리소스 접근 권한이 담긴 access token을 제공
    - Access Token
        - 리소스 서버에게서 리소스 소유자의 정보를 획들할 때 사용되는 만료 기간이 있는 token
    - Refresh Token
        - access token 만료시 이를 재발급 받기 위한 용도로 사용하는 token

2. 권한 부여 방식
    a. Authorization Code Grant (권한 부여 승인 코드 방식)
        - 권한 부여 승인을 위해 자체 생성한 authorization code를 전달하는 방식 (많이 쓰이는 방식)
        - 간편 로그인 기능에서 사용되는 방식으로 클라이언트가 사용자를 대신하여 특정 자원에 접근을 요청할 때 사용되는 방식
        - 보통 타사의 클라이언트에게 보호된 자원을 제공하기 위한 인증에 사용
        - Refresh token의 사용이 가능한 방식



    b. Implicit Grant (암묵적 승인 방식)
    c. Resource Owner Password Credentials Grant (자원 소유자 자격증명 승인 방식)
    d. Client Credentials Grant (클라이언트 자격증명 승인 방식)

3. OAuth 2.0의 흐름
    a. OAuth 등록
        - 클라이언트가 리소스 서버를 이용하기 위해선 사전에 승인을 받아야 함 → 등록이 필요
        - 요구사항
            - Client ID : 애플리케이션을 식별하는 식별자
            - Client Secret : 식별 비밀번호
            - Authorized redirect URIs : 사용자가 인증을 마치고 리다이렉션시킬 위치
        - 리소스 서버와 클라이언트는 요구사항을 공유(클라이언트가 보내줬기 때문에)
    b. Resource Owner의 승인
        - 리소스 서버는 요구사항(Cli ID, Cli secret, URLs)을 받은 상태
        - 리소스 서버는 클라이언트가 요청하는 기능만큼 인증을 요구
        - 리소스 오너는 cli 접속 시 로그인 버튼을 눌러 Cli ID, scope(요청 기능), 리다이렉트 주소를 받음
        - 리소스 서버는 리소스 오너가 가지고 있는 Cli ID, scope, URL을 자신의 정보와 매칭 시킴 → 매칭 실패시 작업 종료
        - 매칭이 같을 경우 리소스 오너에게 범위에 해당하는 권한을 Cli에게 부여할 것인지 확인하는 메세지 전송
        - 허용할 경우 리소스 오너는 허용 정보(유저 아이디, scope 정보)를 리소스 서버에 전송
    c. Resource Server의 승인
        - 리소스 서버는 autorization code를 생성한 뒤 리소스 오너에게 코드가 담긴 url로 이동하라고 명령
        - 아주 짧은 시간에 리소스 오너가 다시 이동 및 Cli에게 authorization code 정보를 전송
        - Cli는 authorization code를 받음
        - cli는 자신의 정보(cli id, cli secret, redir_url, grant_type) + 받은 정보(authorization code)를 통해 리소스 서버에 직접 이동
        - 리소스 서버는 정보를 종합해 맞는지 매핑 → access token 발급

d. 액세스 토큰 발급

- client, 리소스 서버는 둘 다 auth code를 삭제
- 리소스 서버는 access token 생성 및 client에 발급
- cli가 access token과 cli id를 리소스 서버에 던질 경우 리소스 서버는 매핑하여 해당 범위만큼 기능을 허용

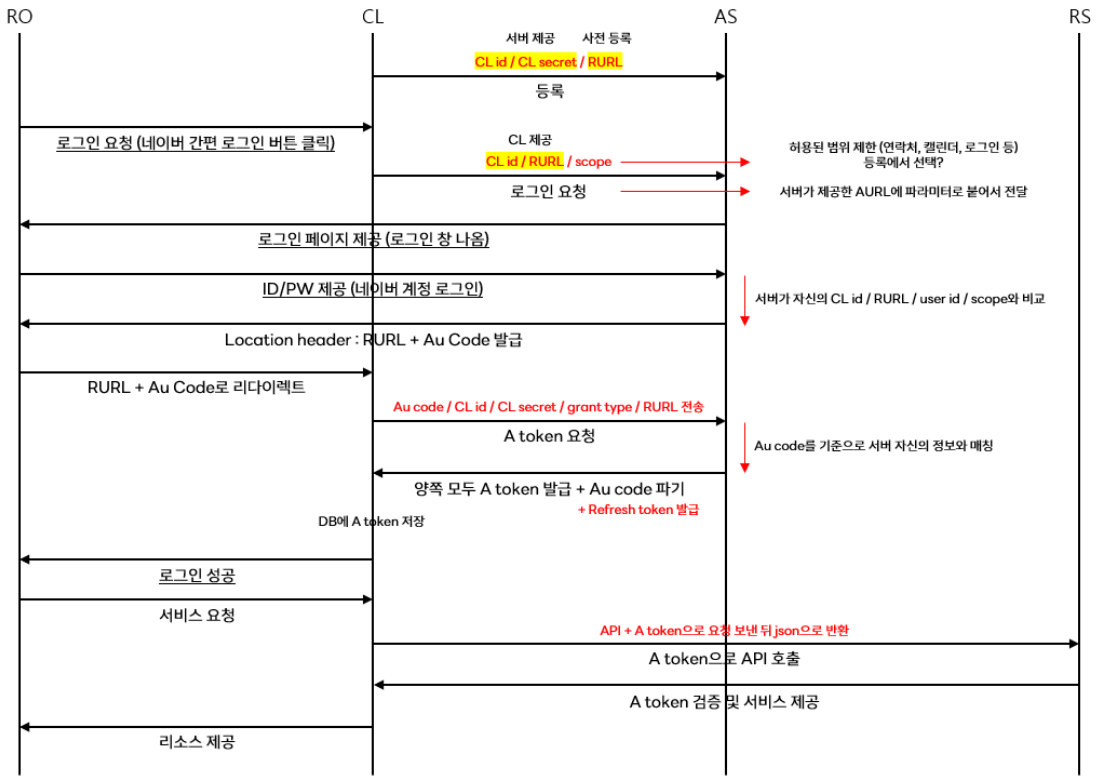
e. API 호출

- 리소스 서버가 cli에게 정보를 제공하는 방식 → API
- url에 api + access token을 입력할 경우 원하는 데이터가 json 방식으로 출력

f. Refresh Token

- access token은 대부분 수명이 짧음. 수명이 다할 경우 계속 재발급 하기 번거로움
- access token은 stateless로, 서버에서 제어할 수 없음. access token을 제3자에게 탈취당할 경우 보안에 큰 문제
- cli에서 refresh token을 인증 서버에 보낼 경우 access token과 조건적인 refresh token을 발급받음

4. OAuth 2.0 도식화


RO
CL
AS
RS
서버 제공
사전 등록
CL id / CL secret / RURL
등록
로그인 요청 (네이버 간편 로그인 버튼 클릭)
CL 제공
CL id / RURL / scope
허용된 범위 제한 (연락처, 캘린더, 로그인 등)
등록에서 선택?
로그인 요청
서버가 제공한 AURL에 파라미터로 붙어서 전달
로그인 페이지 제공 (로그인 창 나옴)
ID/PW 제공 (네이버 계정 로그인)
서버가 자신의 CL id / RURL / user id / scope와 비교
Location header : RURL + Au Code 발급
RURL + Au Code로 리다이렉트
Au code / CL id / CL secret / grant type / RURL 전송
A token 요청
Au code를 기준으로 서버 자신의 정보와 매칭
양쪽 모두 A token 발급 + Au code 파기
+ Refresh token 발급
DB에 A token 저장
로그인 성공
서비스 요청
API + A token으로 요청 보낸 뒤 json으로 반환
A token으로 API 호출
A token 검증 및 서비스 제공
리소스 제공