OAuth 2.0 공부 (2020. 10. 26)

**1. 정의**

- 획득한 AccessToken을 발급하여 나의 서비스에서 Resource Server에 접근해서 데이터를 가져오고 CRUD등의 기능을 사용할 수 있게 해주는 기술. Federated Identity의 기반이 되는 기술.

**2. 역할**

Resource Owner : 유저

Client : 나의 서비스

Resource Server : 데이터를 가지고 있는 서버 (ex. Google, Facebook, Twitter)

Authorization Server : 인증과 관련된 처리를 전담하는 서버

🡪 앞으로의 설명에서는 편의를 위해 Resource Server와 Authorization Server를 하나로 볼 것임.

**3. 등록**

- Client가 Resource Server를 이용하기 위해서 서버에 승인을 사전에 받아 놓아야 하는 행위

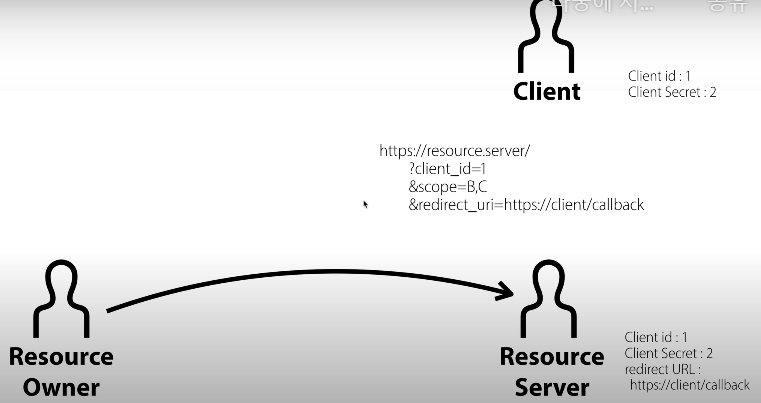
등록 후 Resource Server와 Client가 알게 되는 것 3가지

* Clinet ID
* Clinet Secret
* Authorized redirect URIs

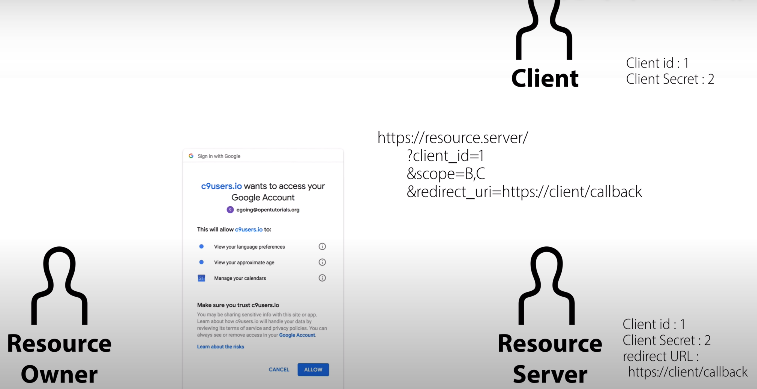
<https://developers.facebook.com/>

<https://cloud.google.com/> : Google Cloud Platform

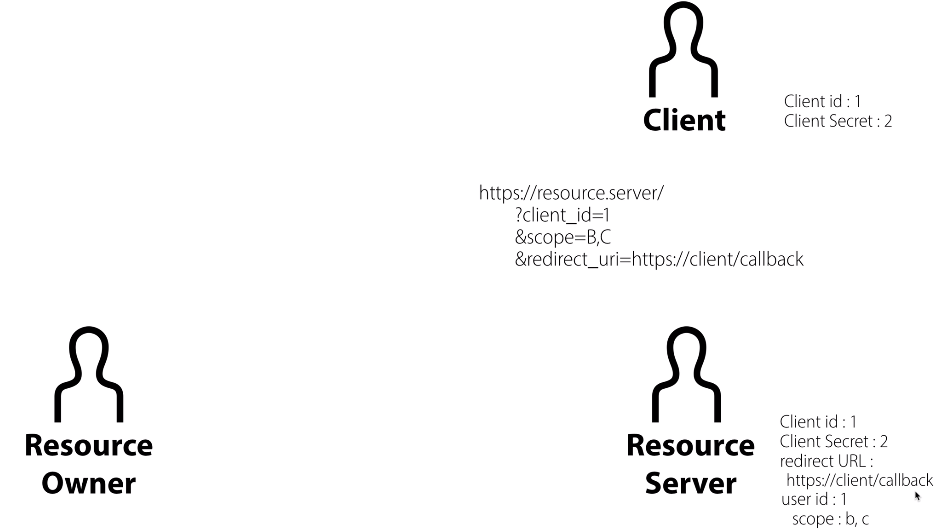
**4. Resource Owner의 승인**



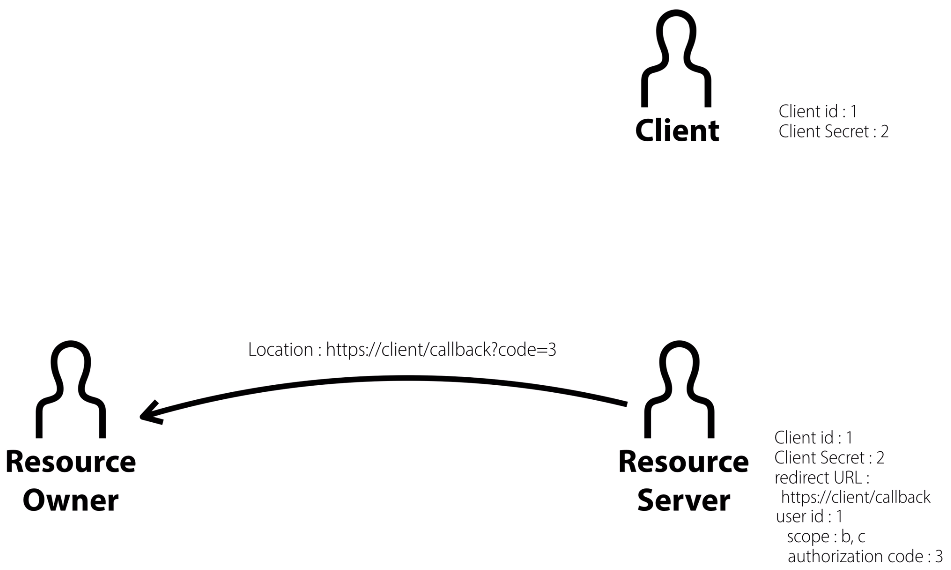
Resource Owner가 요청한 redirect\_url 값이 Resource Server의 redirect URL과 같은지 확인 후 다르면 작업 중단. 같다면 scope에 해당하는 권한을 Client에게 부여할 것인지에 대한 메시지를 아래 그림과 같이 전송.



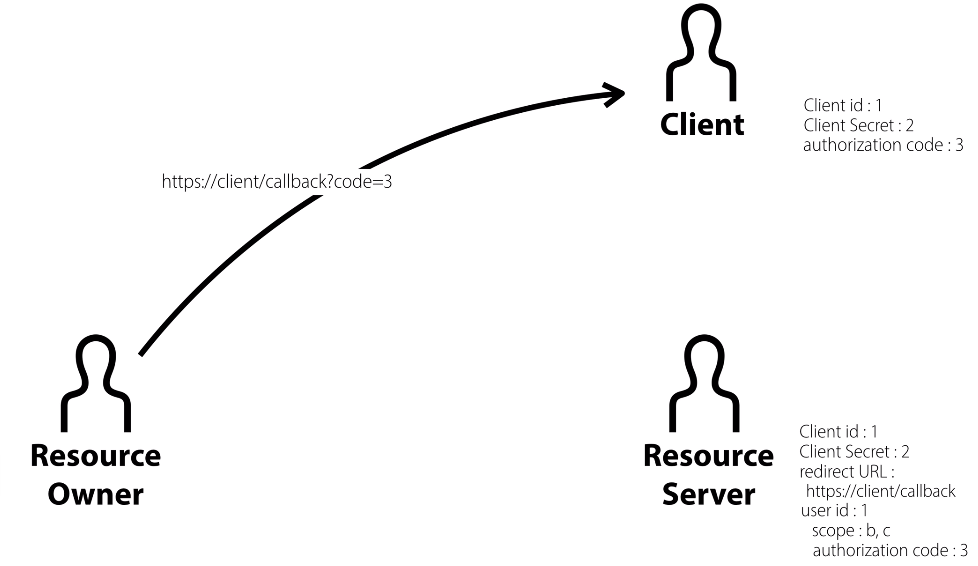
Resource Owner가 승인버튼을 누르면 Resource Server는 아래 그림과 같이 Resource Owner의 user id, scope를 획득. User id 1은 scope B, C작업을 허용하는 것에 동의하였다는 의미.



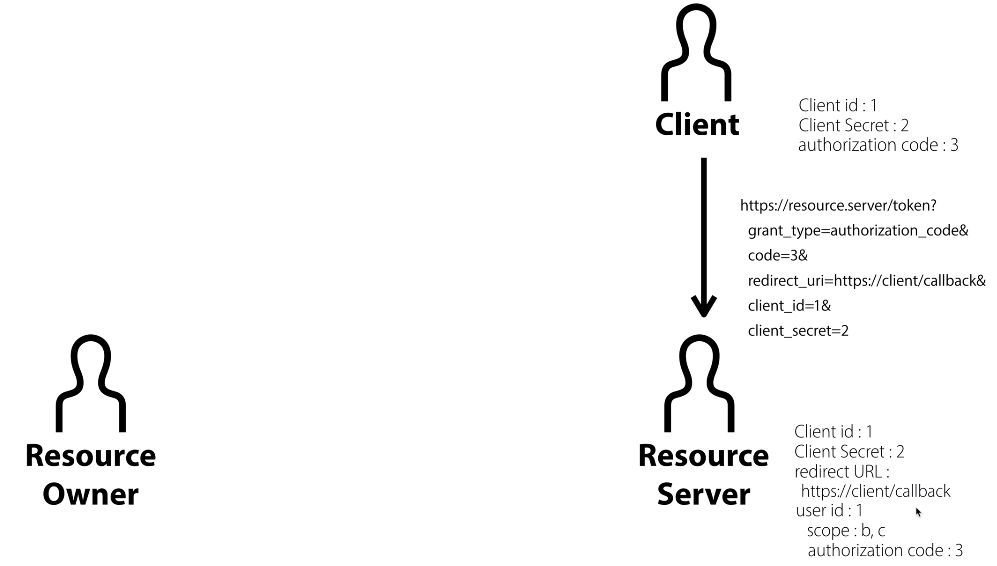
**5. Resource Server의 승인**



삼자간의 인증을 위한 임시 비밀번호 authorization code를 Resource Server는 Resource Owner에게 전송. Resource Server가 Resource Owner의 웹 브라우저에게 redirection 주소를 전달.



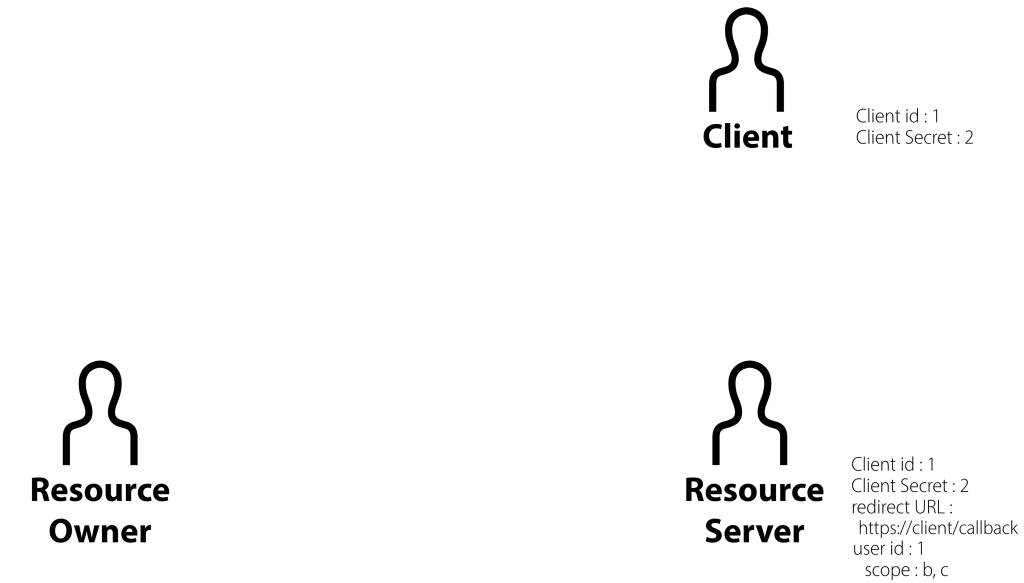
Resource Owner가 인식하지 못하게 은밀하게 redirection 주소로 이동. 그러면 code=3에 의해 Client는 authorization code : 3 값을 갖게 된다.



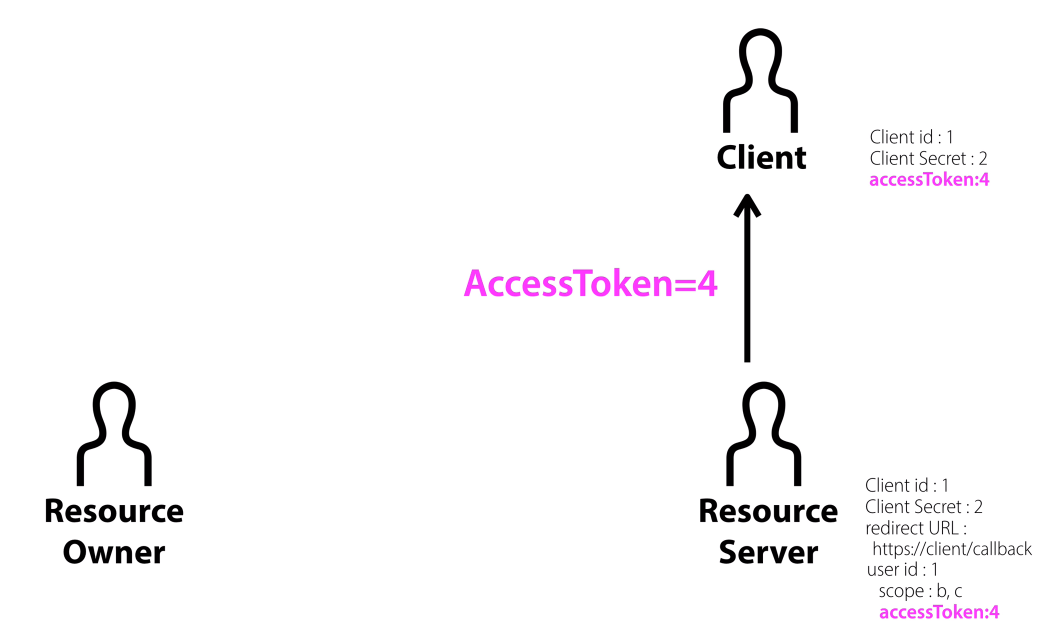
Client는 Resource Owner를 통하지 않고 Resource Server에 위와 같은 주소로 직접 접속. Client\_id와 함께 Authorization code와 client\_secret이라는 두 개의 비밀 정보를 결합하여 Resource Server에 전송.

Resource Server는 Client가 전송한 Client id, secret, authorzation code, redirect\_url이 완전히 일치하는 지 확인하고 모두 일치한다면 AccessToken 발급 단계로 진행.

**6. Access Token**



Authorization code로 인증이 완료되면, Client, Resource Server에서 코드 값을 삭제한다.



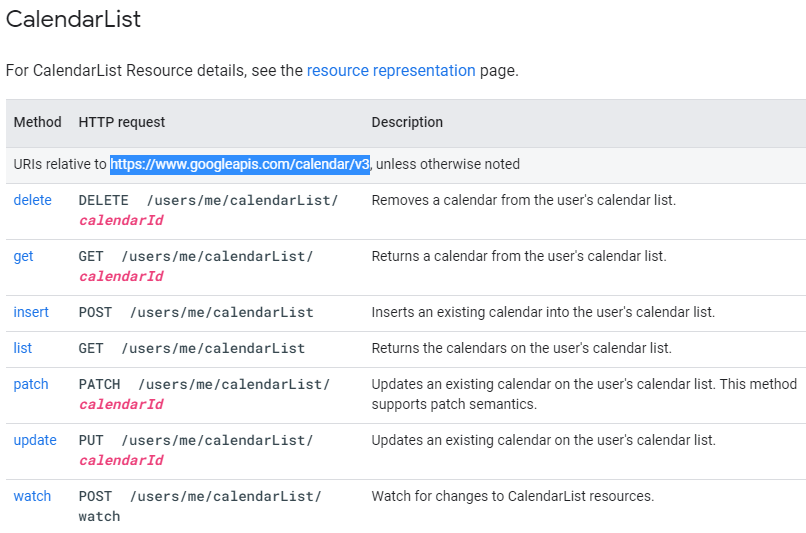
이제 Resource Server가 accessToken을 발급한다. 이후 accessToken을 client에게 응답해준다. Client는 accessToken 을 내부적으로 DB, 파일 등에 저장한다.

client가 accessToken 4로 접근할 때 resource server는 accesstoken 4를 보고 이 토큰은 user id 1에 해당하는 사용자의 유효한 기능 b, c 에 대해 권한이 열려있는 access key 이니깐,

b, c, user id : 1에 해당하는 사용자의 정보에 대해서 accessToken 4를 가진 사람에게 허용을 해준다.

**7. API 호출**

Ex. Google calendar api



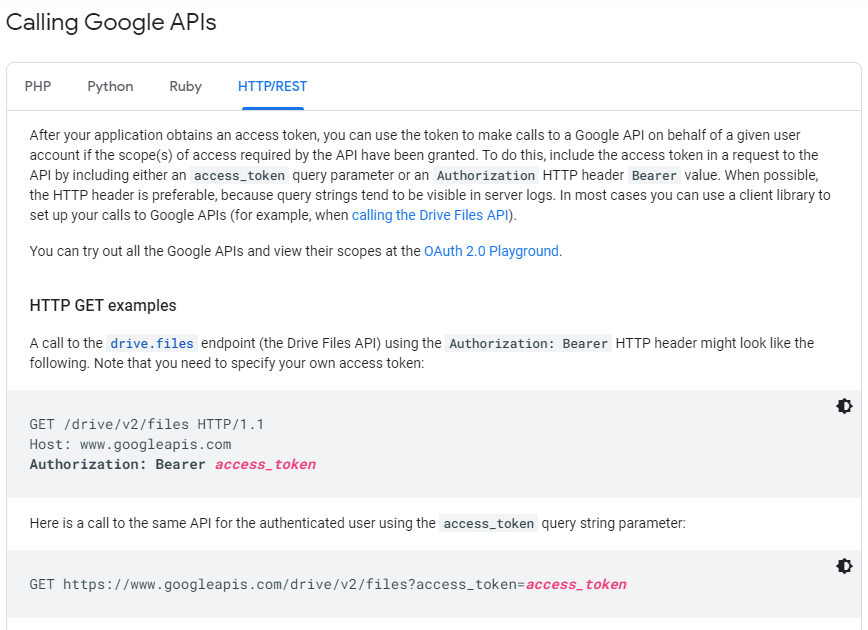
<https://www.googleapis.com/calendar/v3/users/me/calendarList>

🡪 Login required Error 출력

🡪 OAuth(Access Token) 를 통해서 데이터를 가져올 수 있다.

Access Token을 구했다 치고 어떻게 할 수 있는 지 살펴보자

Ex. <https://developers.google.com/identity/protocols/oauth2/web-server> (Google api access token oauth)

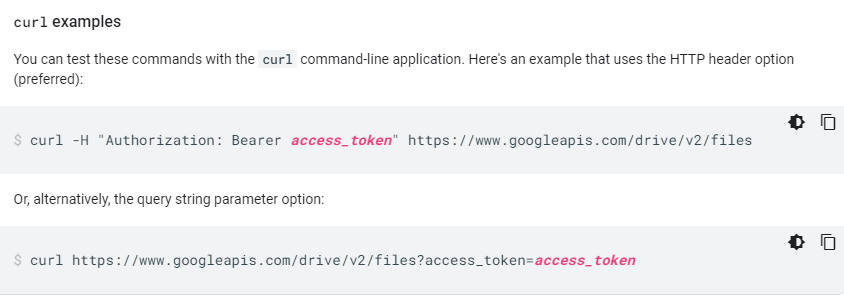


<https://www.googleapis.com/calendar/v3/users/me/calendarList>?access\_token=*access\_token*

이런식으로 접속하면 가지고 있는 calendar 목록이 나온다.

더 안전하고 선호되는 방식은 다음과 같다.

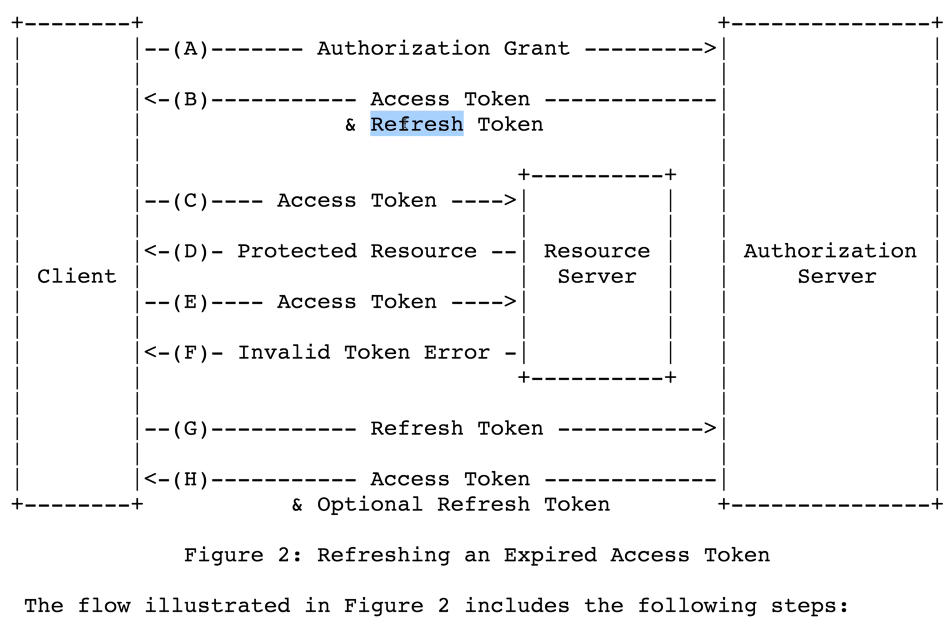
Bearer : OAuth를 위해 고안된 인증방법. Curl 프로그램 실행(POST man etc…)

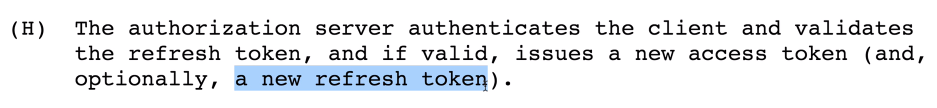


**8. Refresh token**

- Access token이 만료되었을 때 손쉽게 다시 발급받을 수 있는 방법.

<https://tools.ietf.org/html/rfc6749> : OAuth 표준 문서



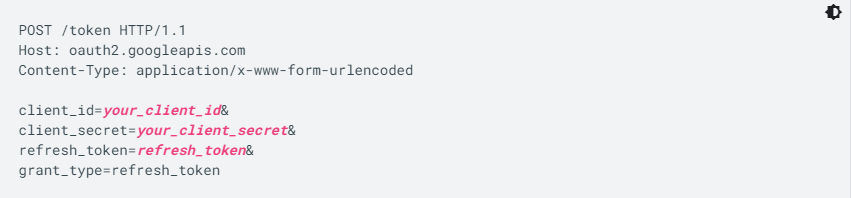


Refresh 토큰도 새로 갱신되는 경우도 있고, accessToken만 계속 갱신되는 방식도 있다.

Access token이 만료되었을 때 Refreshing 예제

<https://developers.google.com/identity/protocols/oauth2/web-server>

Refreshing an access token



application/x-www-form-urlencoded = form에서 POST 방식으로 전송하라.

client\_id, client\_secret, refresh\_token, grant\_type 값을 전송하면 구글에서는 아래와 같은 JSON 포맷으로 데이터를 리턴



새롭게 발급된 access\_token과, 유효기간들을 담아서 보내준다.