oAuth

One service to authorize another service on behalf of the user.

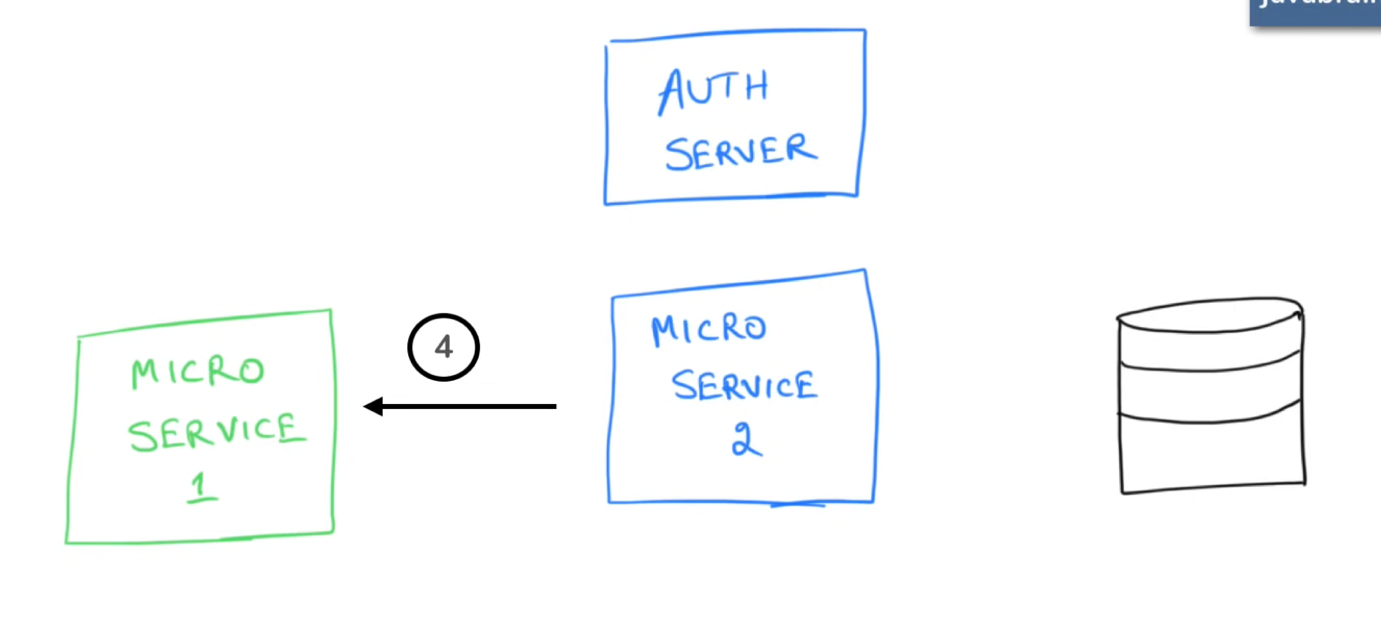
More relevant in Microservice Architecture for services to talk to each other in a secure manner

Resource Owner (User), Client (App), Authorization Server (oAuth Server), Resource Server (For example, gmail)

Basic Workflow: Client requests the AS – AS replies that you can have access if the user authenticates – AS redirects to user (pop up browser where user signs in the AS) – AS provides Access Token to client – client sends requests to the RS with the Access token – RS calls the AS to validate the token – if the AS validates the RS sends data to client

Workflow with OpenIdConnect: Same as before but the AS responds with Access Token and Id token that contains information about the user – Client uses the ID token to create its own session – ID token is typically JWT

Client Credentials Flow for oAuth: Used when clients is trustworthy. MS1 calls AS1 and provides its identity (key) - AS provides Access token for kind of information that MS1 needs – MS1 calls Apis in MS2 with Access token to fetch the relevant resources



https://<ouathEndpoint>/client\_id=<>&client\_secret=<>&grant\_type=client\_credentials&scope=<scope>\* => Access Token (JWT) which will have aud and roles

aud will be same endpoint as the WebRtc setupCallAPI and roles could be

\*write\* i.e. setupCall. We should define roles ourselves and is an implementation detail

Client\_id and client\_secret => service account (GCS Helpcenter) and generate a client\_secret

How to share secrets => Email => keymaker

**POST** /oauth2/token

**Host**: webrtc.paypal.com

**Header:**

*Content Type:* application/x-www-form-urlencoded

**Body:**

*grant\_type*: client\_credentials

*client\_id*: set to the identifier of the client application

client\_secret: set to the client secret.

**Sample Request:**

POST https://webrtc.paypal.com/oauth2/token >

Content-Type='application/x-www-form-urlencoded'&

client\_id=Basic aSdxd892iujendek523uedj&

client\_secret= 2YotnFZFEjr1zCsicMWpAA

grant\_type=client\_credentials

**Sample Response (Positive):**

HTTP/1.1 200 OK

Content-Type: application/json

{

"access\_token":"eyJz9sdfsdfsdfsd",

"token\_type":"Bearer",

"expires\_in":3600

}

**Sample Response (Negative):**

HTTP/1.1 400 Bad Request

Content-Type: application/json;charset=UTF-8

{

"error":"invalid\_request|invalid\_client|invalid\_grant|unauthorized\_client|unsupported\_grant\_type|"

}

**Description of error codes in accordance with RFC6749:**

*invalid\_request*: Missing a required paramemter (for example, grant\_type)

*invalid\_client*: The client has failed authentication.

*invalid\_grant*: The client doesn’t have access to the scope to create calls.

*unauthorized\_client*: The client is not authorized for client\_credentials grant type.

*unauthorized\_grant\_type*: Returned if the client uses any other grant\_type than client\_credentials.