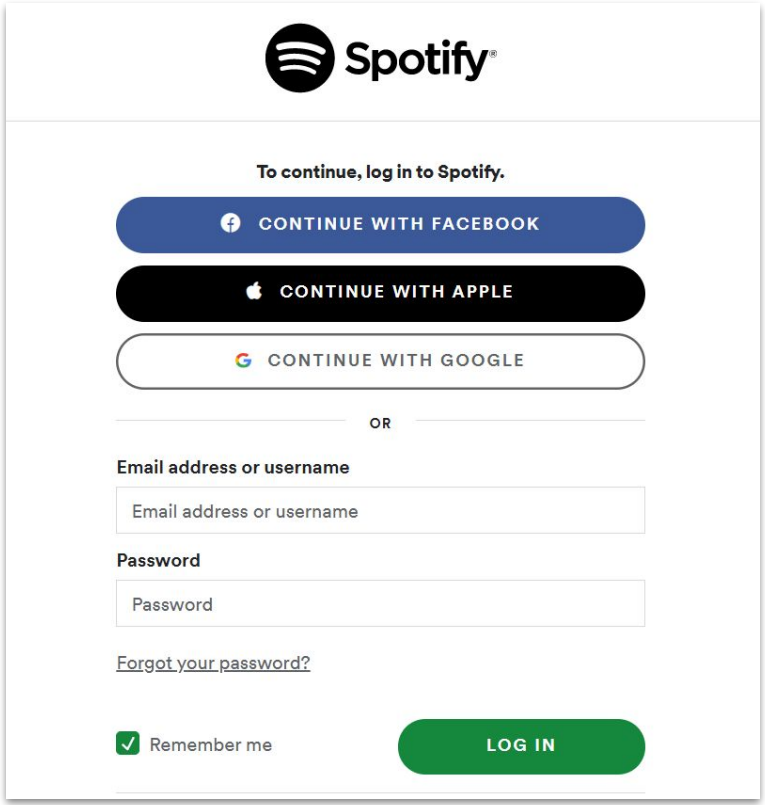
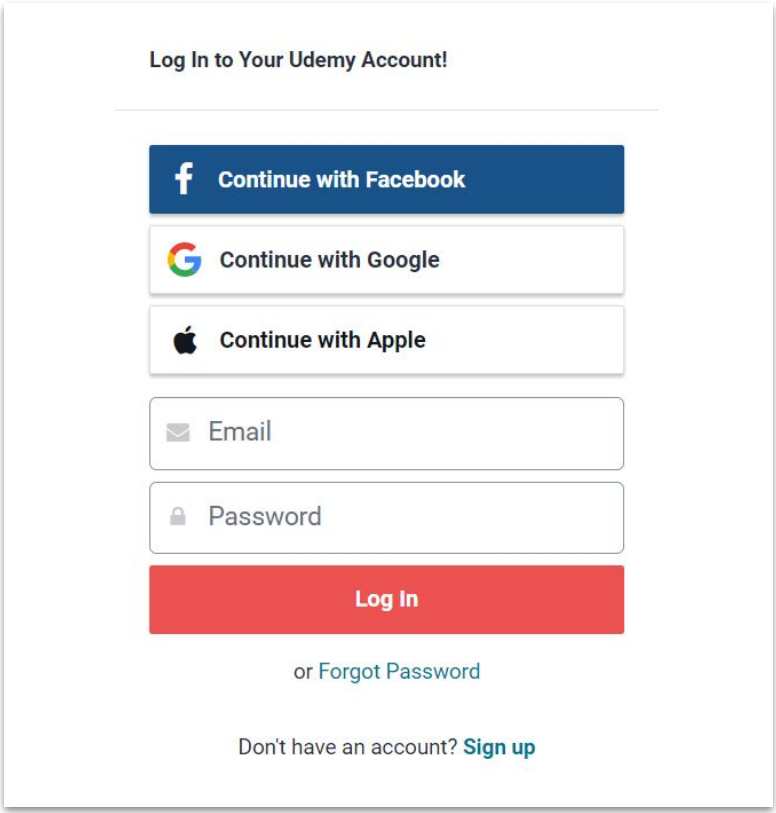
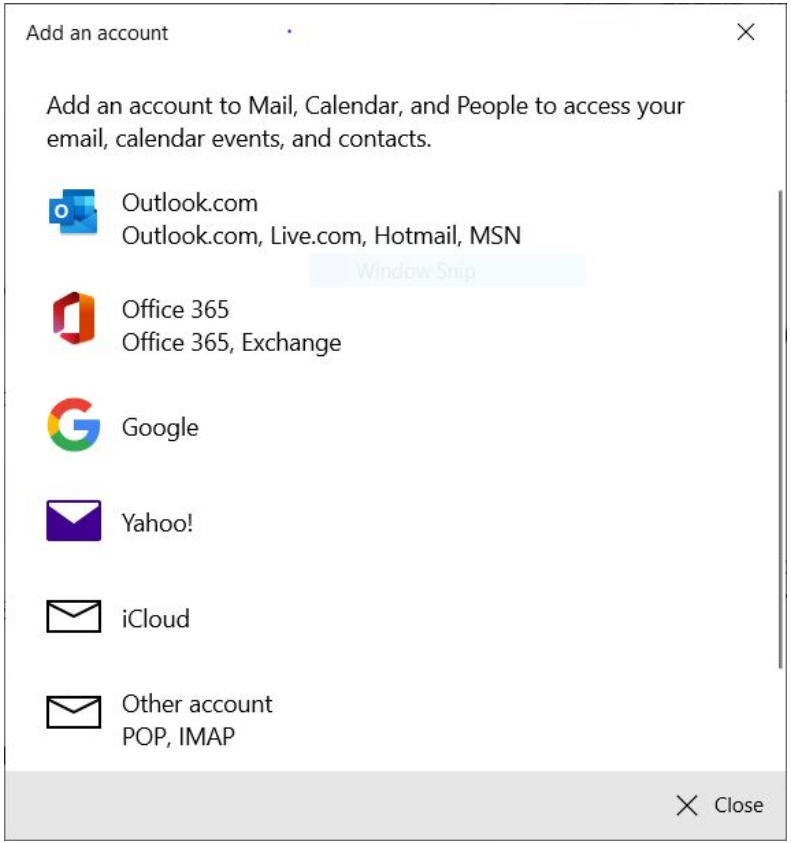
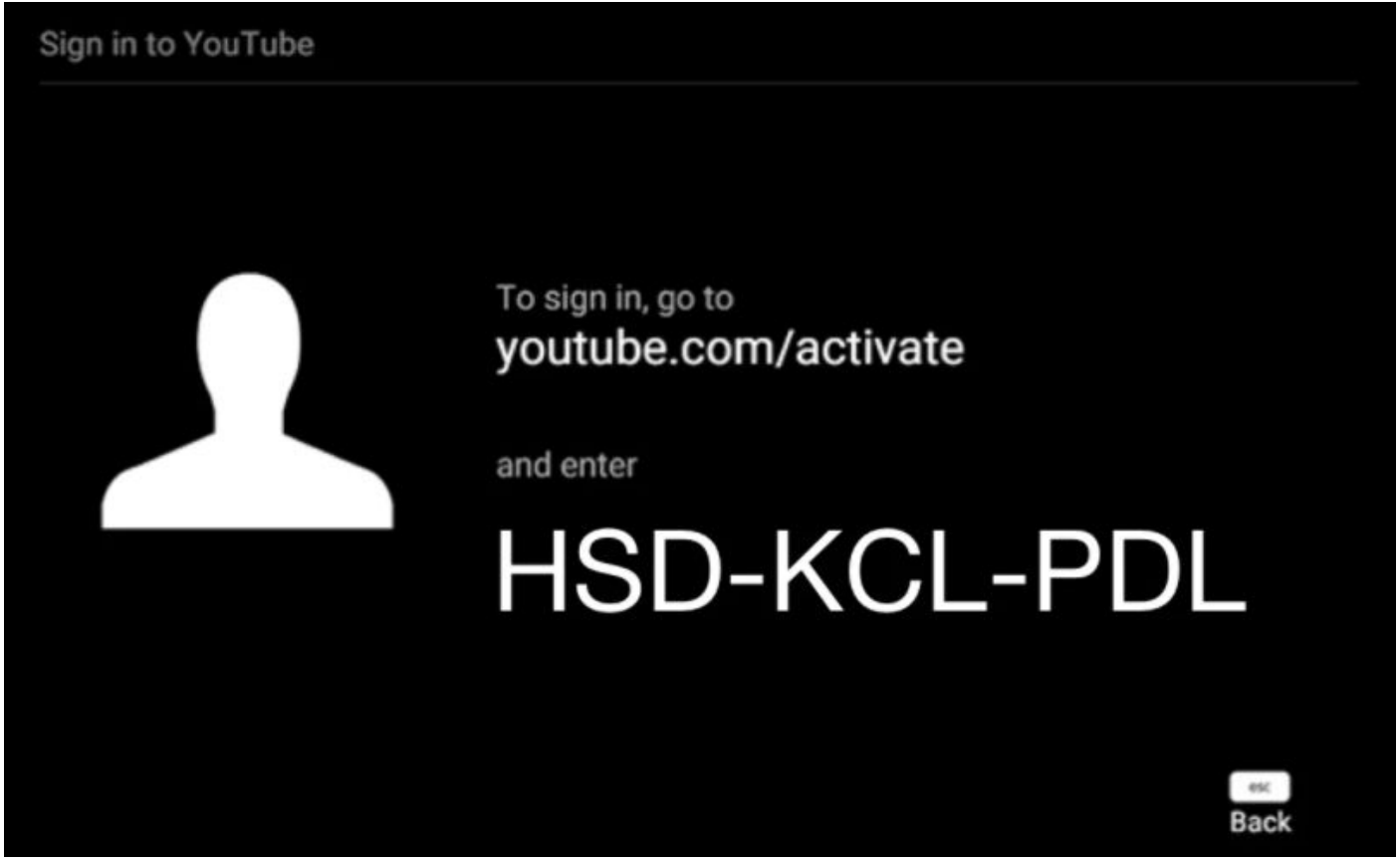
# Why learn OAuth?

# Mobile Applications



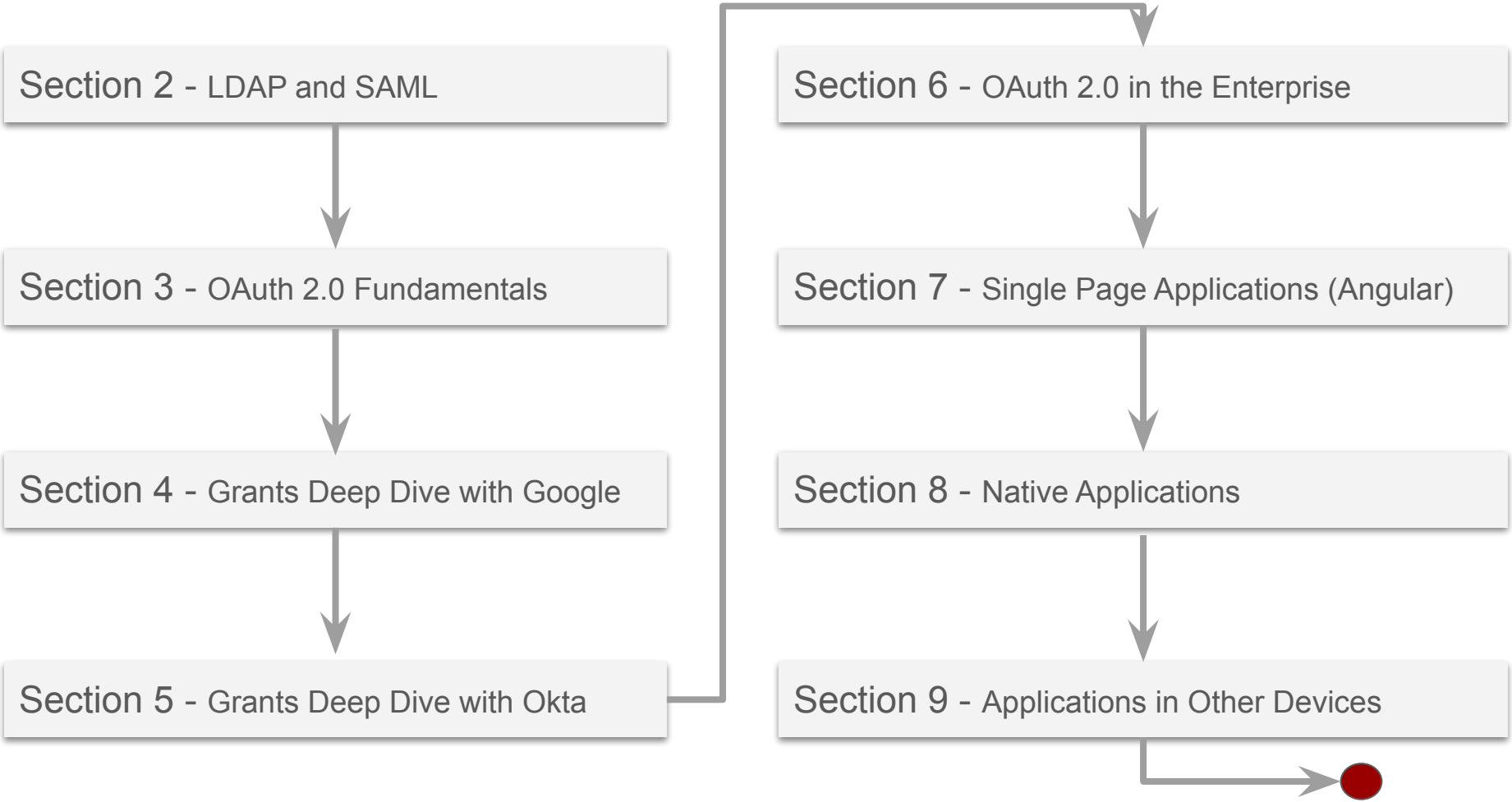







# Why learn OAuth?

# Course Contents

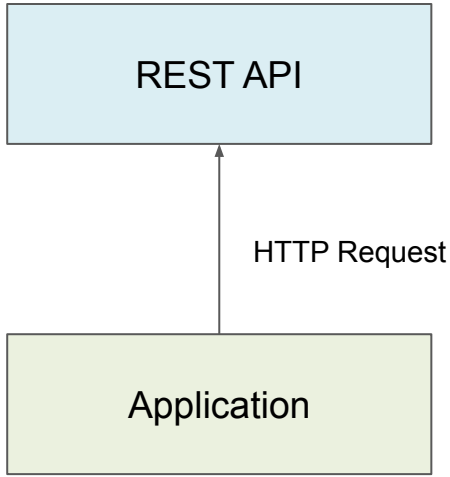


Process

Application

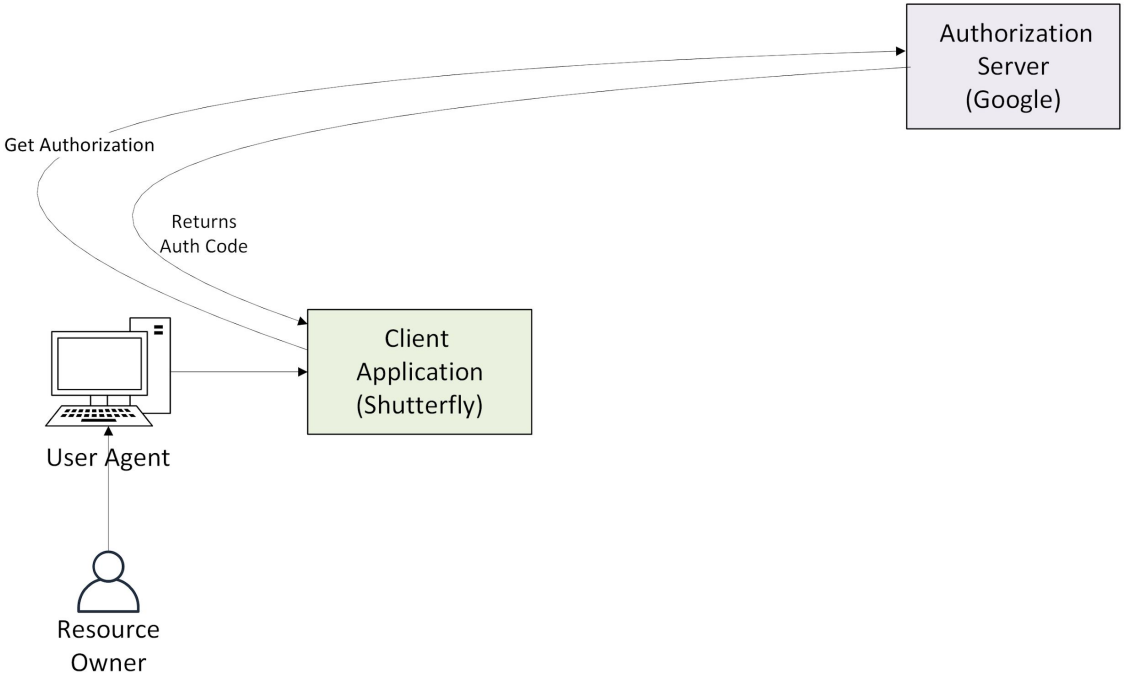
REST API

Authorization Server



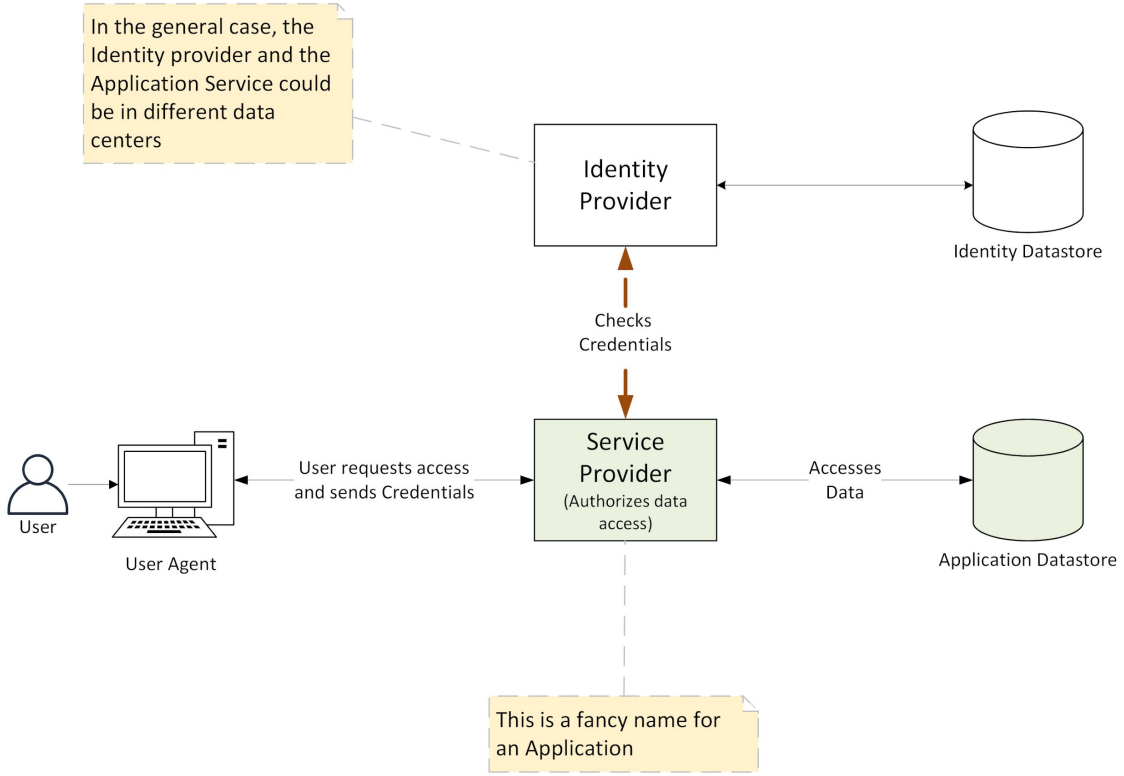
Scheduled Cron Jobs can call REST API across network boundaries but how can they be secured ? What user would they run as ?

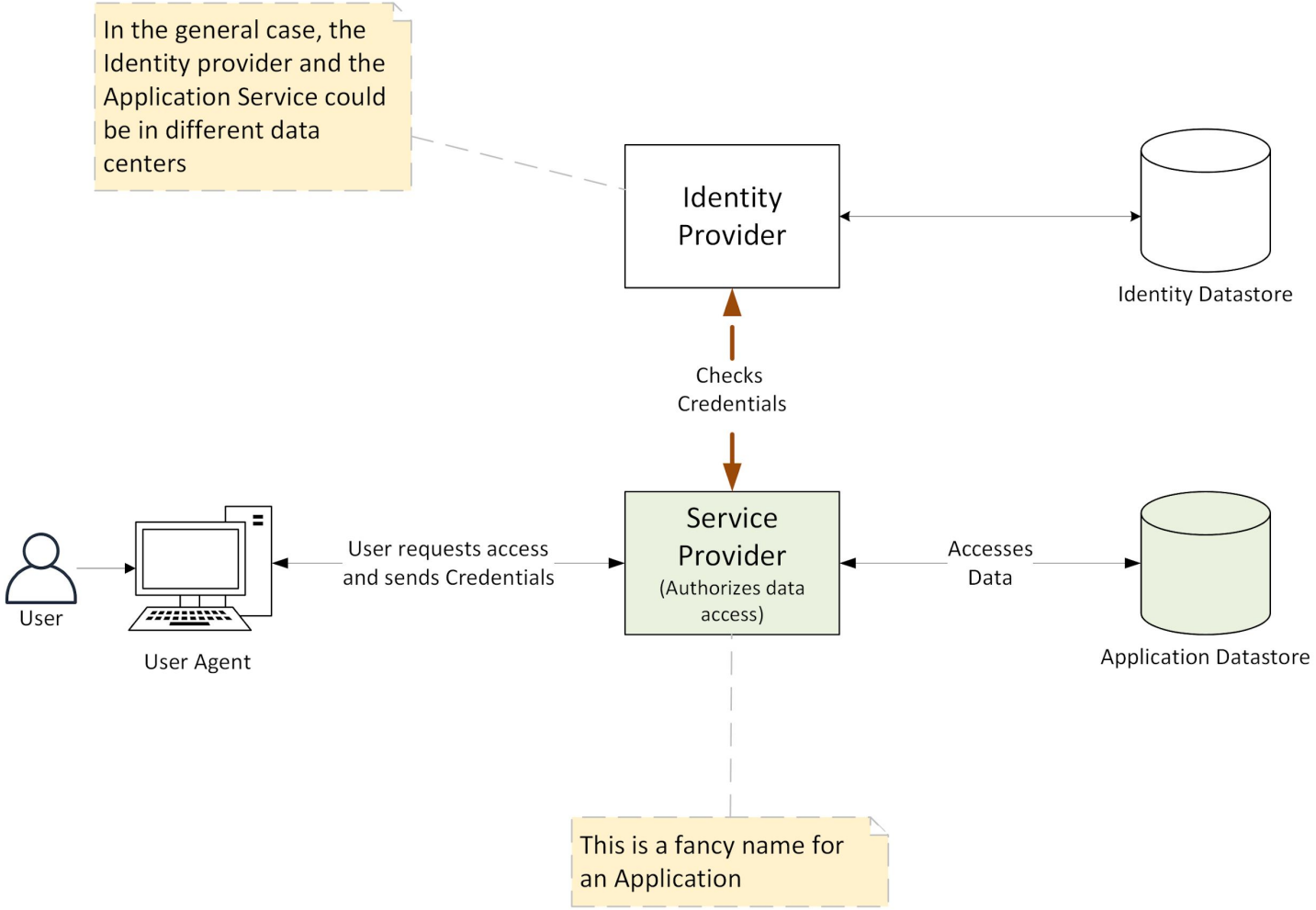


# Security Basics

# Security Basics - Providers

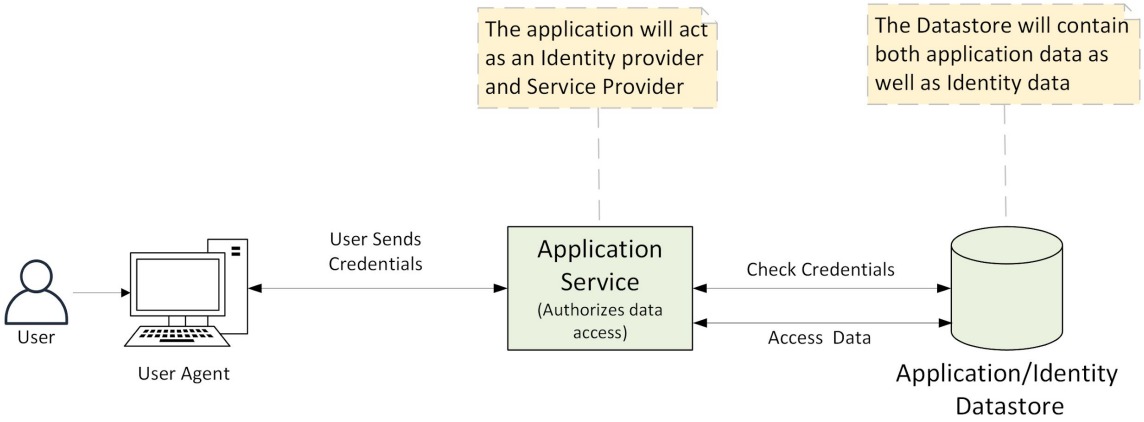
- User, Identity
  - Where are the user credentials stored?
- Authentication
  - Who does this ? idP
  - How does it happen?
- Authorization
  - Application does this
  - Uses information from IdP





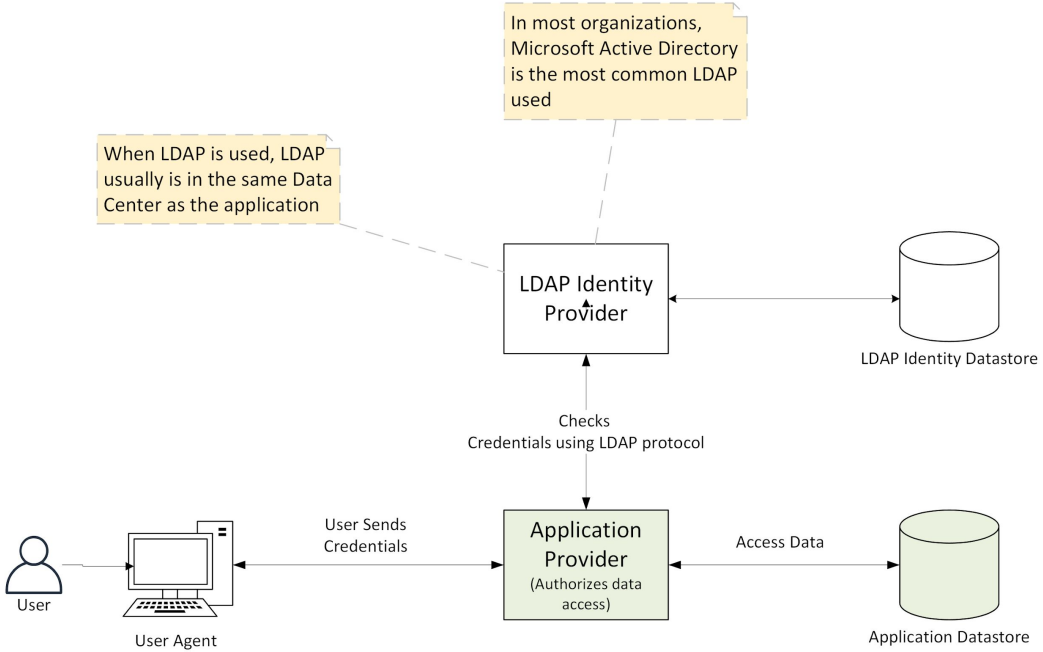
# Security Basics - Homegrown

- Identity
  - Stored in application database
- Authentication
  - Done by Application
- Authorization
  - Done by Application
- Not recommended

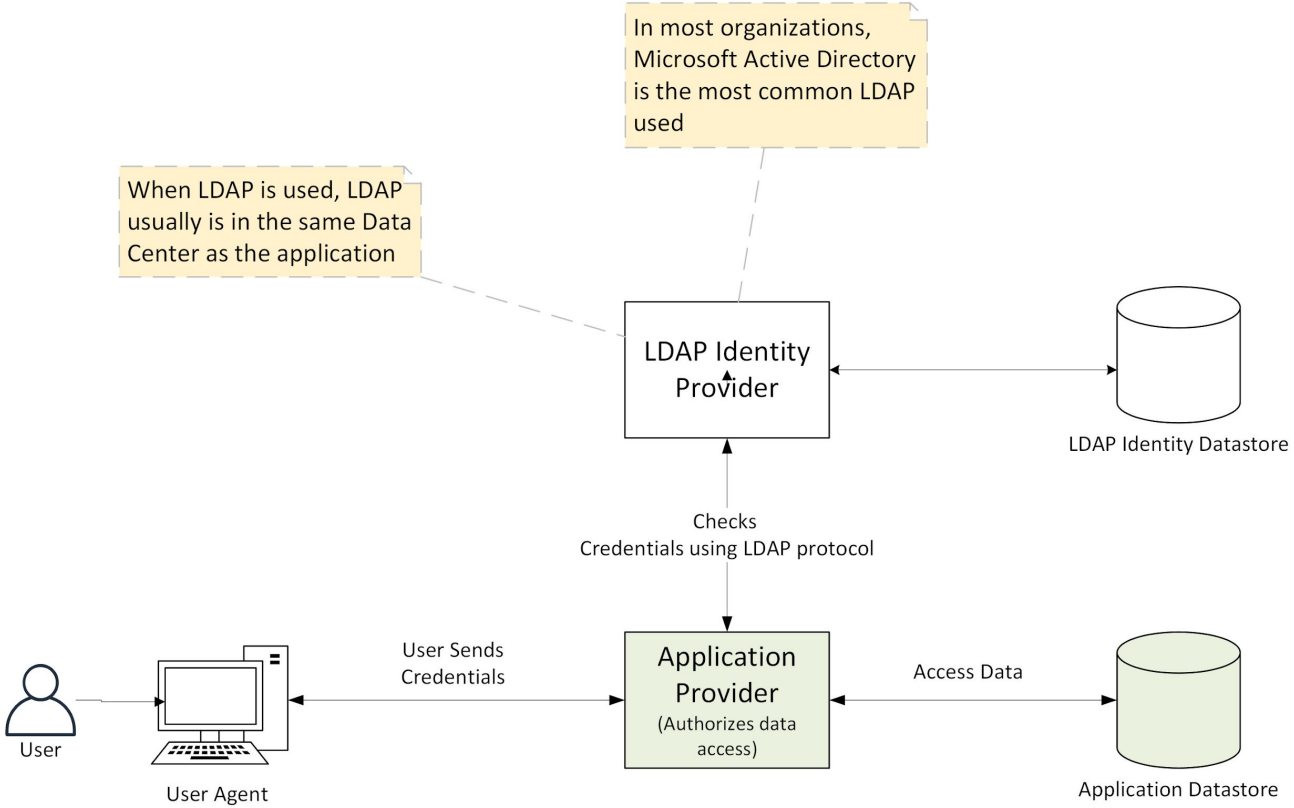


# Security Basics - LDAP

- Identity
  - Stored in LDAP database
  - Microsoft Active Directory
- Authentication
  - Done by LDAP (AD)
- Authorization
  - Done by Application
- LDAP and Application in same data center



# Security Basics - LDAP

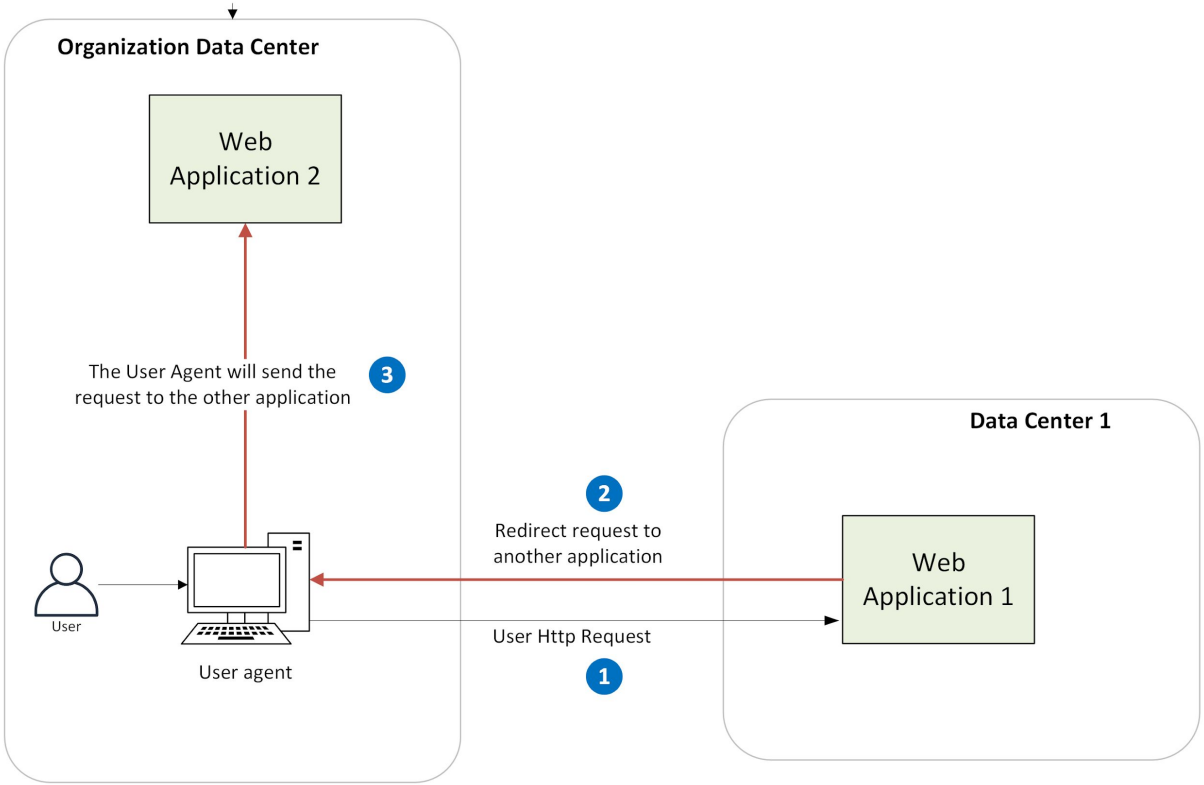


# What problem does SAML solve?

## (Security Assertion Markup Language)

# Security Basics - SAML

- Communicating across data centers (use HTTP Redirect)

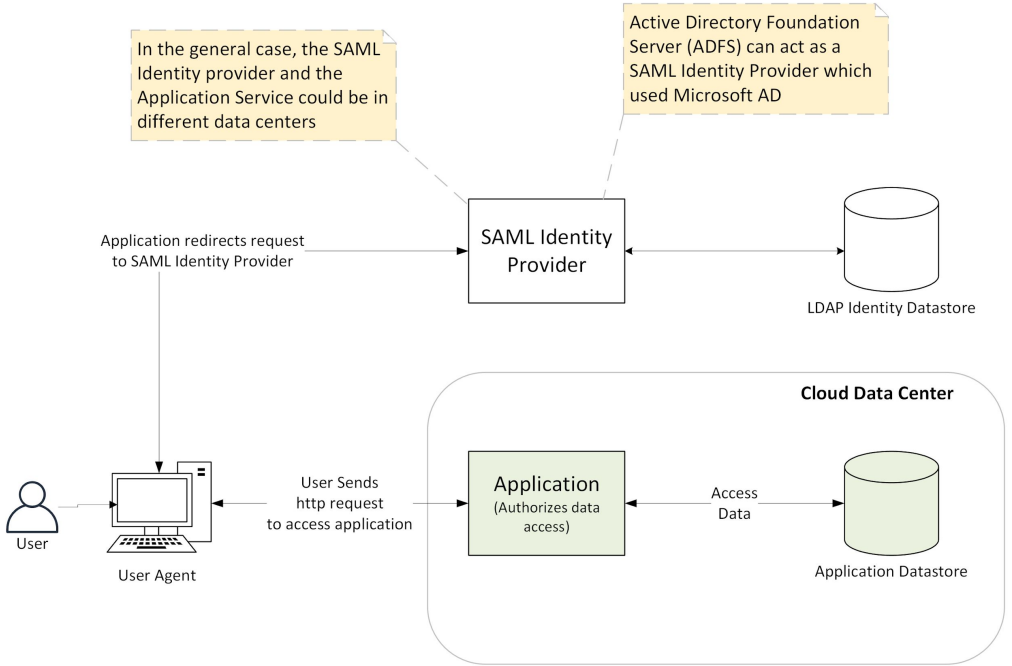


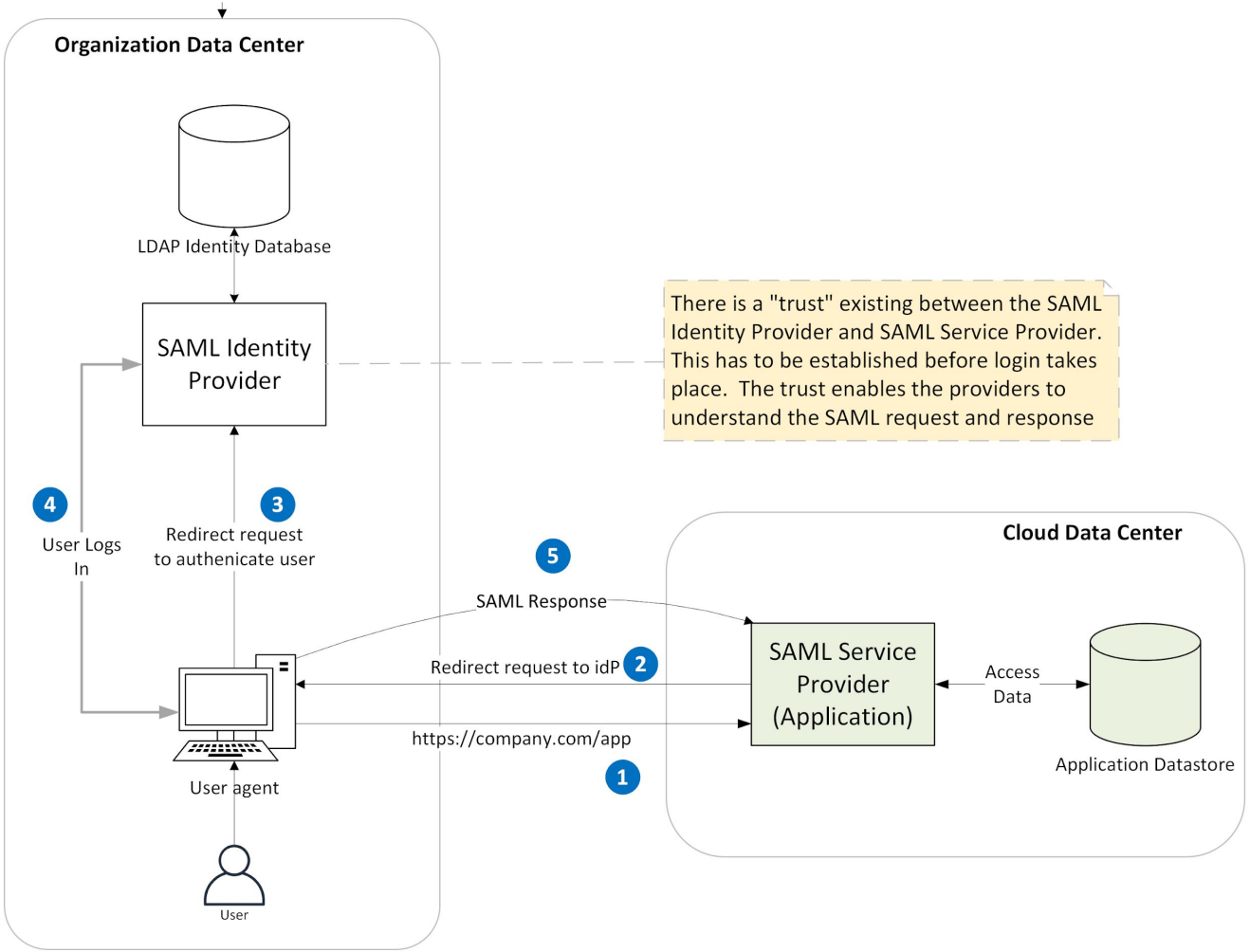
# Security Basics - SSO

- Avoid entering the user credentials (use SSO)
- The network user is already a part of AD
- Single Sign On (SSO)
  - Enterprise SSO

# Security Basics - SAML

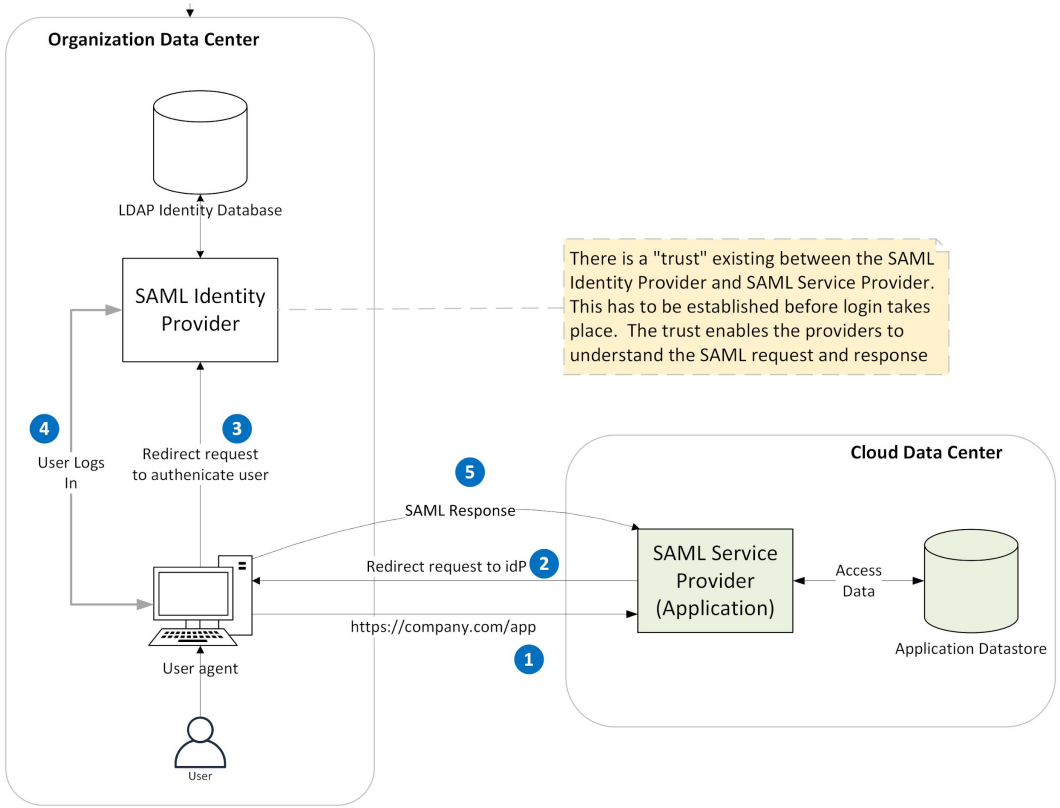
- Identity
  - Stored in LDAP database
  - Microsoft Active Directory
- Authentication
  - SAML Identity Provider does the authentication
  - ADFS
- Authorization
  - Application controls it
  - Can use LDAP groups





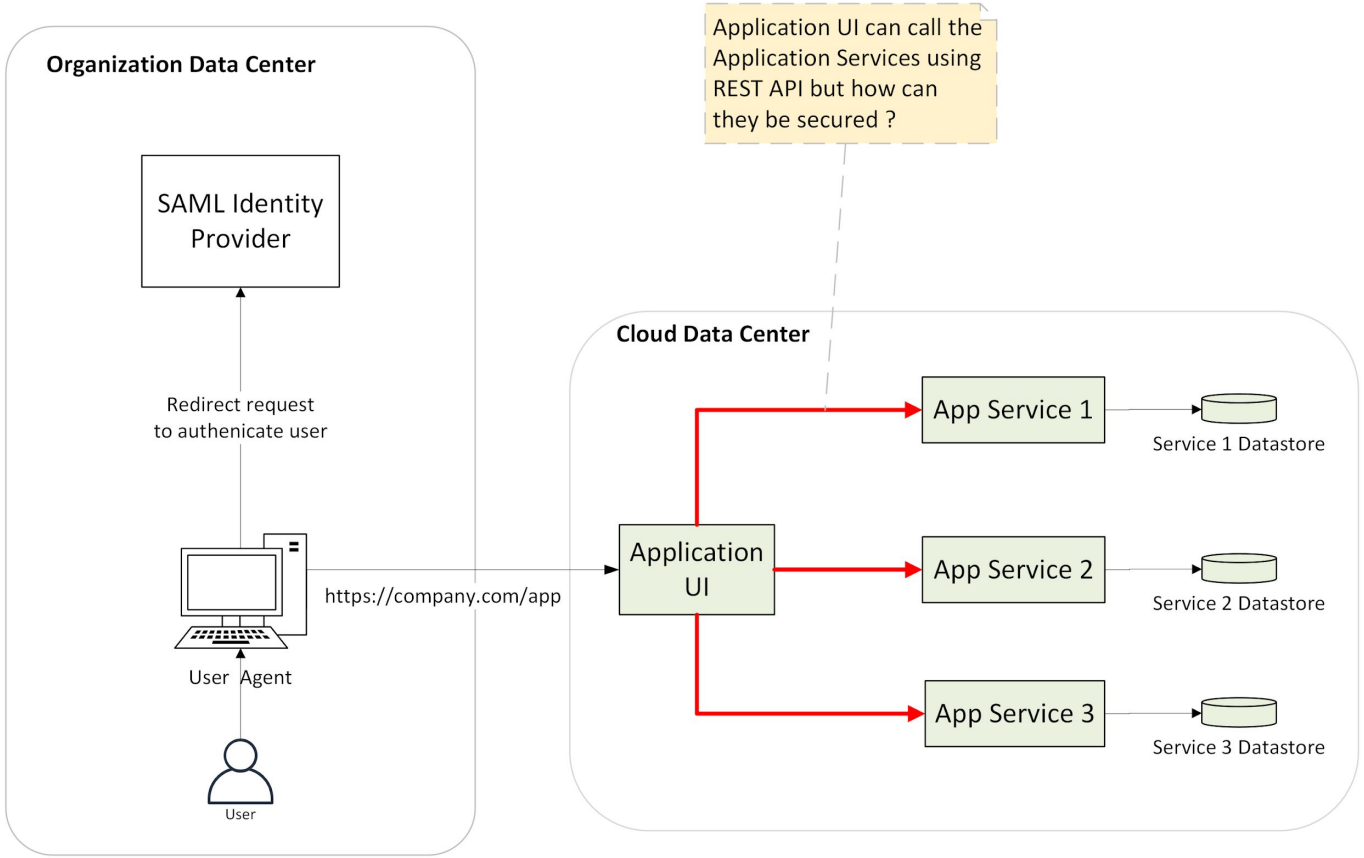
# Security Basics - SAML

- SAML Metadata File
- Trust between
  - SAML Identity Provider
  - SAML Service Provider
- SAML Response
  - Contains SAML Token
  - Token contains claims
- Federated User
- **Single Sign On**
- **Redirect importance**



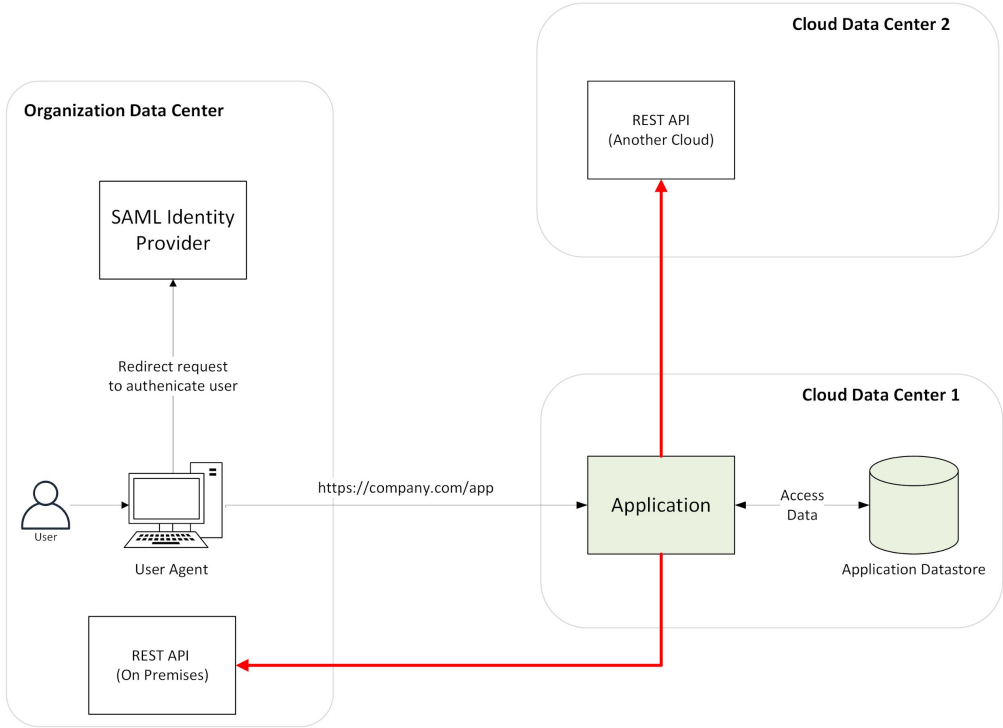
# Enterprise Problem use cases

# Enterprise problem 1 - Microservices



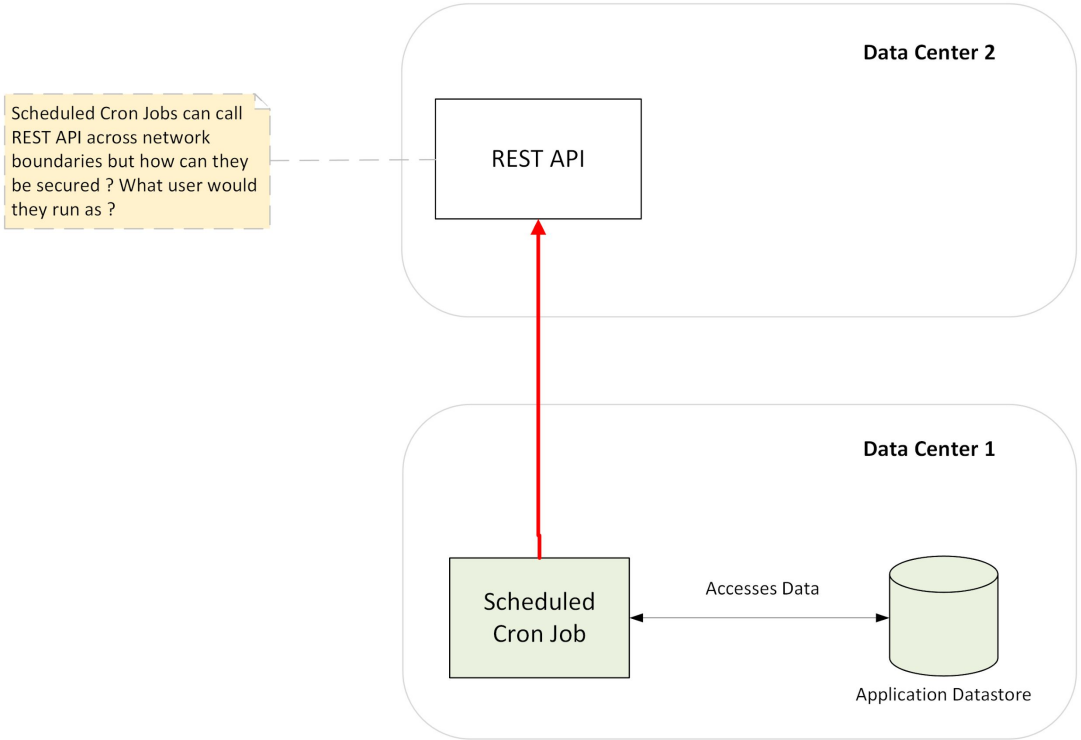
# Enterprise problem 2 - Cloud apps

- How does REST calls across network boundaries get secured ?



# Enterprise problem 3 - Machine to Machine

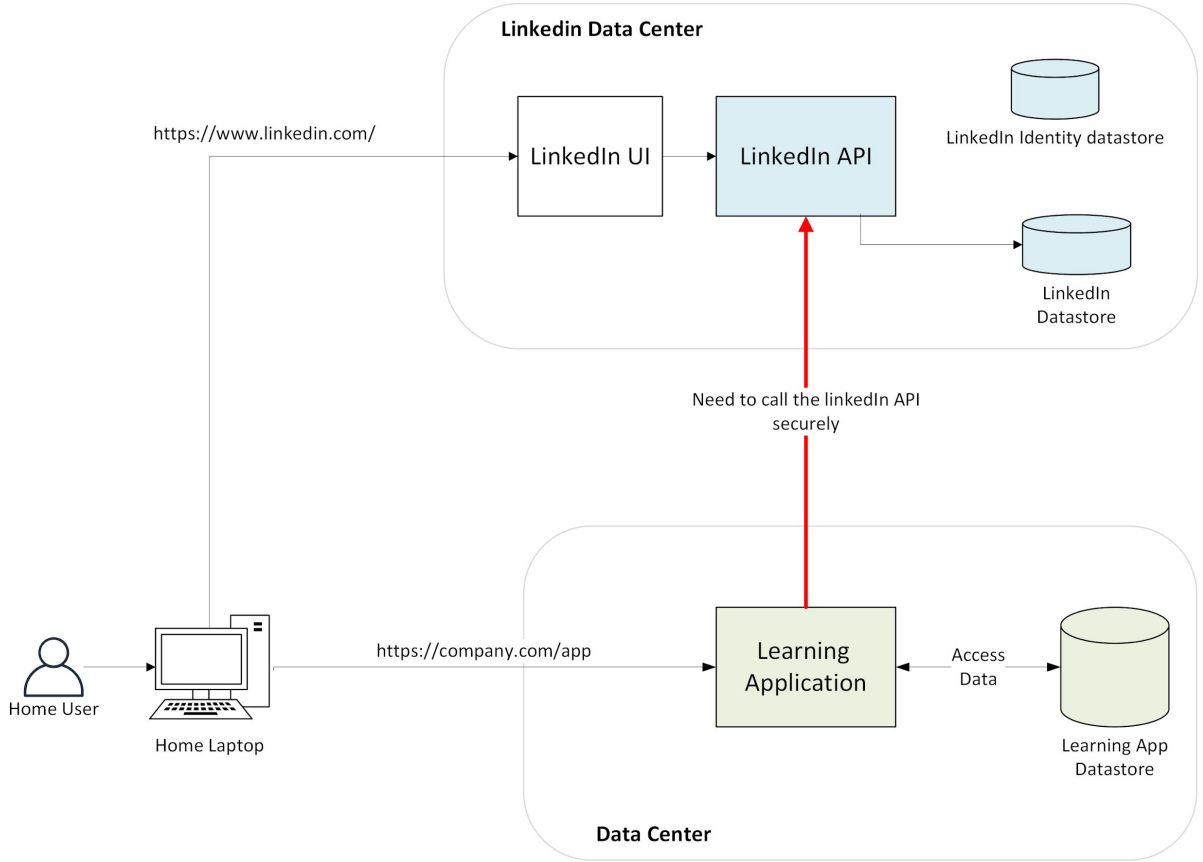
- Scheduled Tasks, Daemons sometimes need to call REST APIs. How are they secured ?
- No user involved



# Social Media Platform

- Social Media Sites
  - Facebook
  - LinkedIn
  - Google
  - Twitter
  - GitHub
  - Yahoo
- A user usually has Multiple Identities
  - Many Identity Providers
- What if a third party application wants to access or publish to these sites on behalf of its user ?
  - User/password would be a bad idea.

# Social Media Applications - Problem



# The Answer ..

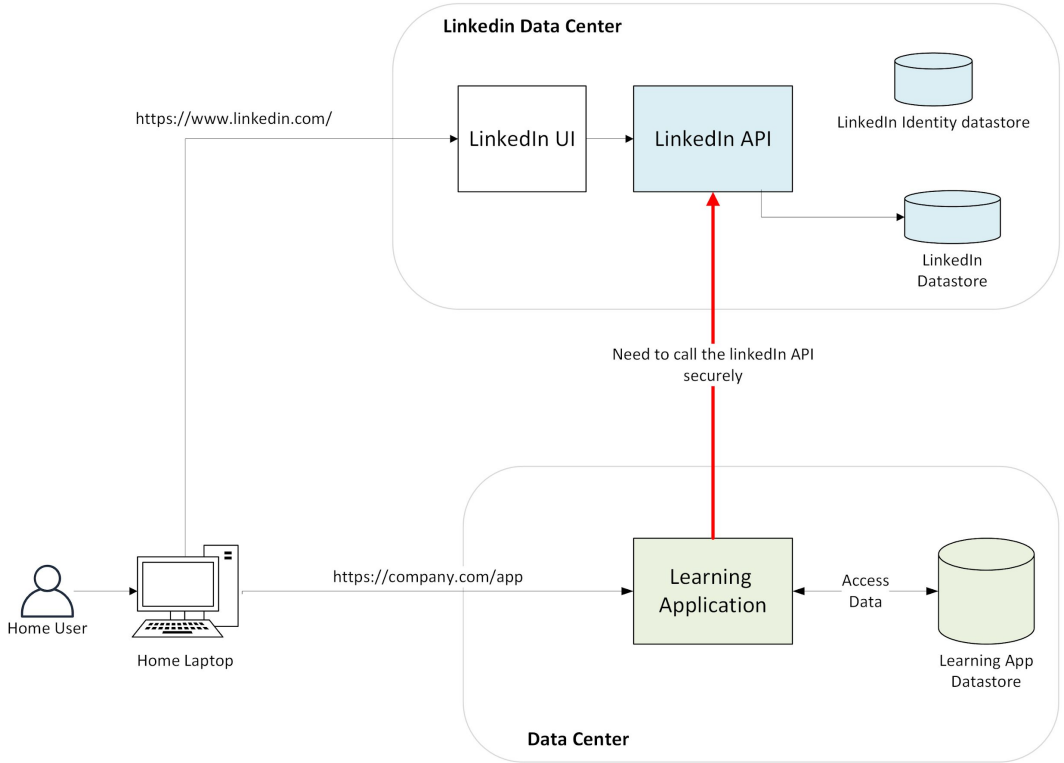
# OAuth 2.0 Authorization Server

# OAuth 2.0 Authorization Framework (RFC 6749)

The OAuth 2.0 authorization framework enables a **third-party application** to obtain **limited** access to an **HTTP service**, either **on behalf of a resource owner** by orchestrating an **approval interaction** between the resource owner and the HTTP service, or by allowing the third-party application to obtain **access on its own behalf**.

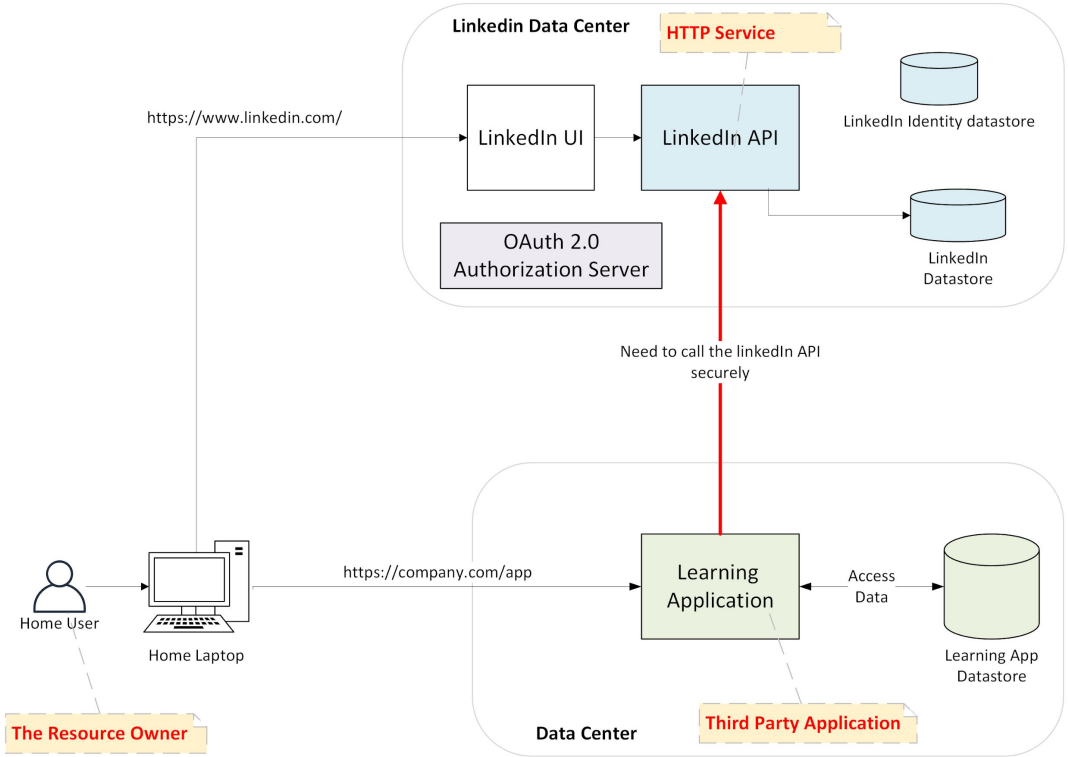
# OAuth 2.0 Authorization Framework

The OAuth 2.0 authorization framework enables a **third-party application** to obtain **limited** access to an **HTTP service**, either **on behalf of a resource owner** by orchestrating an **approval interaction** between the resource owner and the HTTP service, or by allowing the third-party application to obtain **access on its own behalf**.



# OAuth 2.0 Authorization Framework

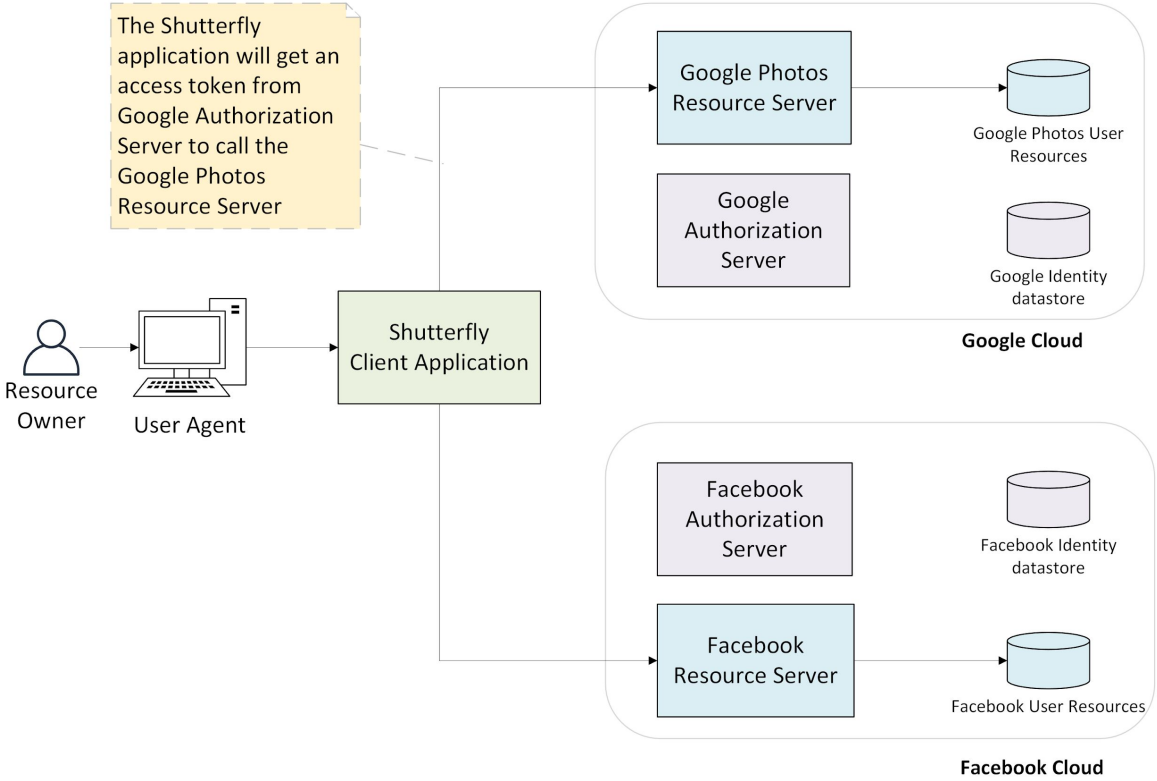
The OAuth 2.0 authorization framework enables a **third-party application** to obtain **limited** access to an **HTTP service**, either **on behalf of a resource owner** by orchestrating an **approval interaction** between the resource owner and the HTTP service, or by allowing the third-party application to obtain **access on its own behalf**.



# Demonstration

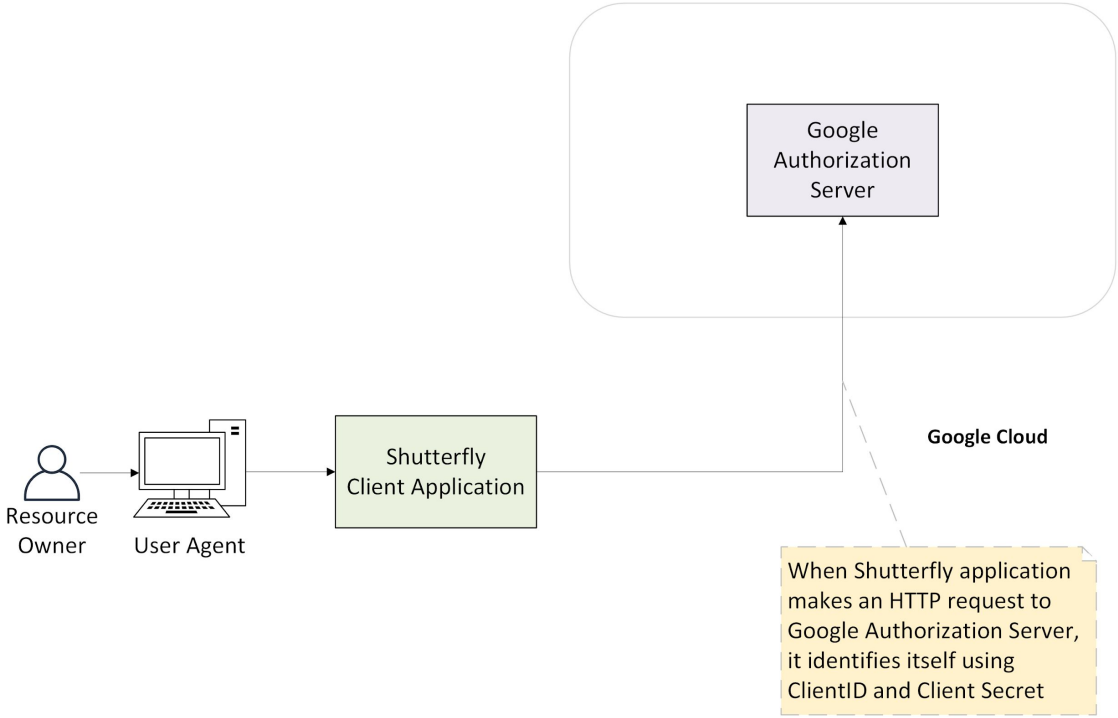
# OAuth 2.0 - Roles

- Resource Owner
  - The User
  - User Agent
- Resource Server
  - REST API which protects resource
- Client
  - Application that needs access
- Authorization Server
  - Authorizes the client
  - Gives out access tokens
  - Many on the internet
  - OAuth Endpoints



# OAuth 2.0 - Client Registration

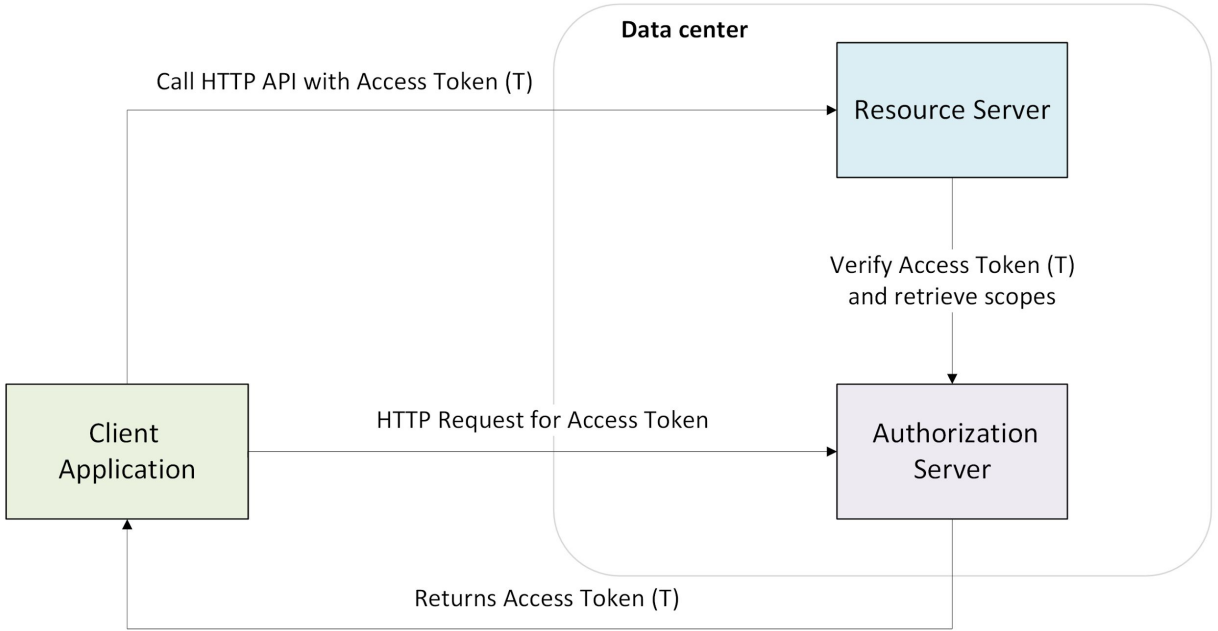
- Administration utility to register a Client
- Redirect URIs
- Client ID, Client Secret
- When Client sends request to Authorization Server, it will send Client ID, Client Secret as well



# OAuth 2.0 - Opaque Token

Access Token is sent in an HTTP Header

**Authorization : Bearer <token>**



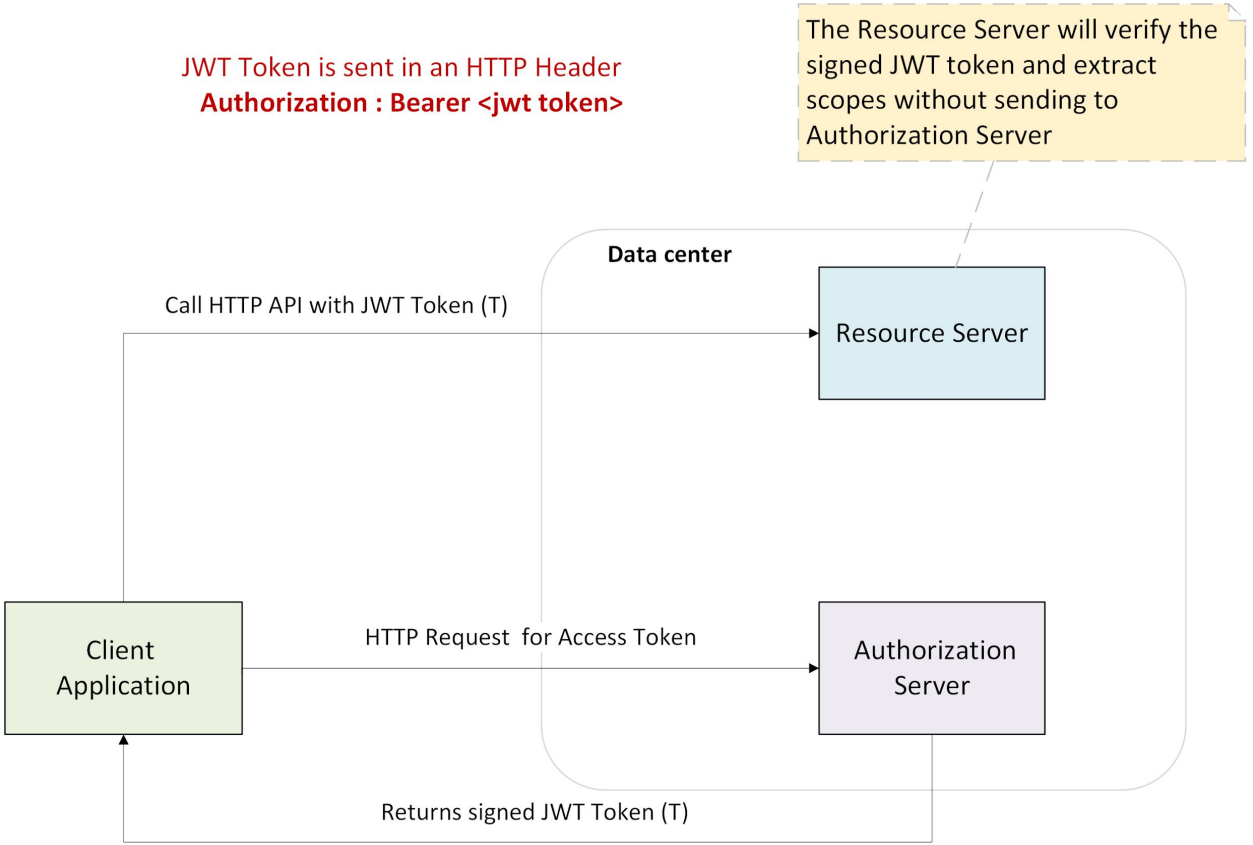
# OAuth 2.0 - Opaque Token Example

1d52703551c84012a7b0af0930092ea6

Authorization: Bearer 1d52703551c84012a7b0af0930092ea6

# OAuth 2.0 - JWT Token

JWT Token is sent in an HTTP Header
**Authorization : Bearer <jwt token>**

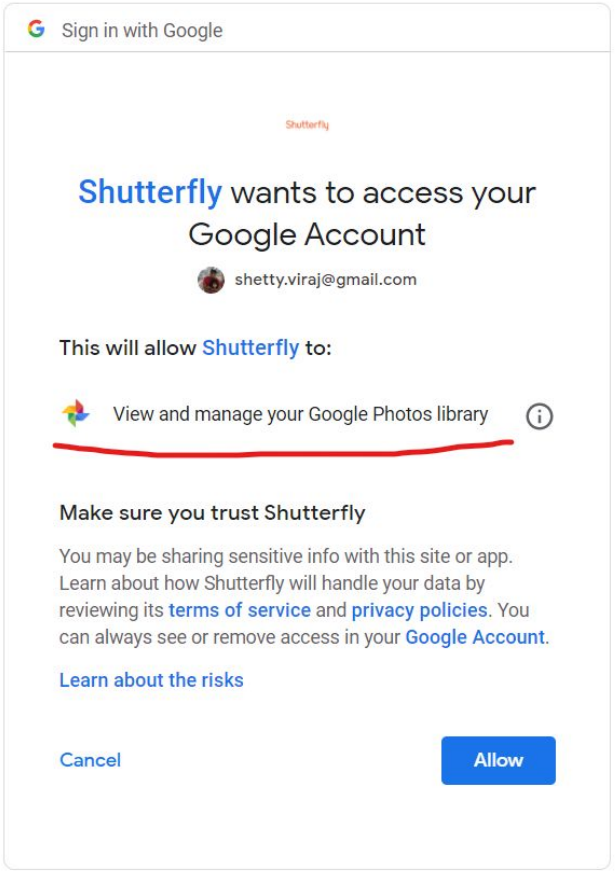


# OAuth 2.0 - JWT Token Example

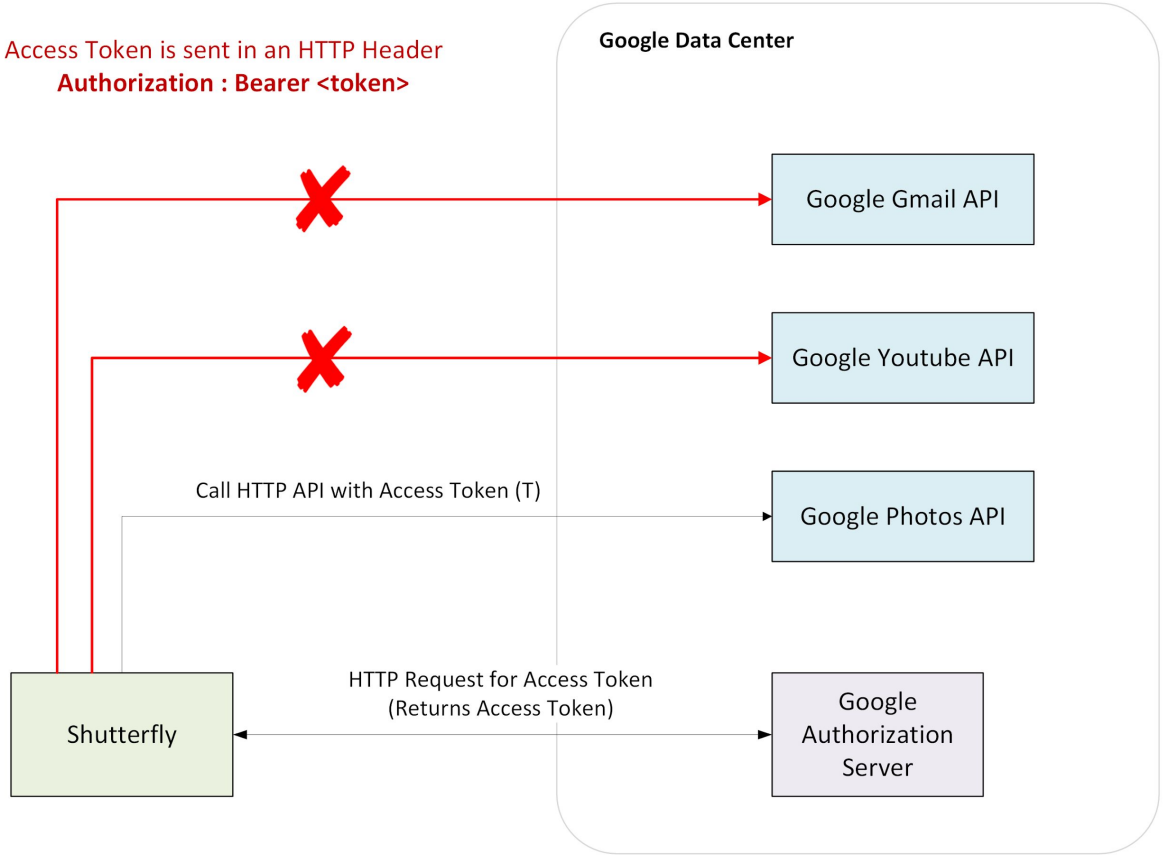
eyJraWQiOiJoWDBaVXliU3pzRjVLVVA4azltdGtDTkEy
X2l6WC03XzZuU1JTVFphLVRjIiwiYWxnIjoiUlMyNTYif
Q.eyJ2ZXIiOjEsImp0aSI6IkFULjY5dUl4V2w4Qi1KcTV
ueXg5SzgyclVQVENhbHdkLXJ4S3dZUmlEZ1RWZFk
ub2FyMWl3eXp5UWFrU2VyT2Y1ZDYiLCJpc3MiOiJo
dHRwczovL2Rldi0yMTQ4MjczLm9rdGEuY29tL29hdX
RoMi9kZWZhdWx0IiwiYXVkIjoiYXBpOi8vZGVmYXVs
dCIsImlhdCI6MTYxOTk4NjQ5MiwiZXhwIjoxNjE5OTg2
NzkyLCJjaWQiOiIwb2FtMmR2YnBQbDJJTUt1UTVkNi
IsInVpZCI6IjAwdTM2b3k4ZkN4eHhQV2hmNWQ2Iiwic
2NwIjpbIm9mZmxpbmVfYWNjZXNzIiwib3BlbmlkIiwiZ
mFrZWJvb2thcGkucmVhZCIsInByb2ZpbGUiLCJmYWt
lYm9va2FwaS5hZG1pbiIsImVtYWlsIl0sInN1YiI6InNoZ
XR0eS52aXJhakBnbWFpbC5jb20ifQ.FoDNU5IW48Jz
dpFwSDQmiA-c_IF23c-oEBjQuJ3X298RAW8ETbL8-5
hhcGV8n4g2kaH33xSaSHa5CZpZfVkfK4hmSb__vTe
DPPedWFfeypTu5kjTVPgsefdIxpUrkyKEd0ci9Q2sonf
R5G_7NW-e2jKXVMLDbWwAUmouKB1207VQQfmIb
VNPway9ZG-yg_j_XR42KMX1RcYTys53yQ71qweoQ
9e2xPynC2DNHGIL8fa060RtVTRQ-dUTn4EUdLjdvpv
EiioMnrzcZjfoxG7SUr5hwGLo_kygf0O4-PDL_I3-sU_o
zEnb7m6HZwxC2tPtA8w7qcv7harY61_4Ygmbvw


```
{
   "ver": 1,
   "jti": "AT.69uB8WI8B-Jq5nyx9K82rUPTCalwd-rxKwYRiDgTVdY.oar1iwyzyQakSerOf5d6",
   "iss": "https://dev-2148273.okta.com/oauth2/default",
   "aud": "api://default",
   "iat": 1619986492,
   "exp": 1619986792,
   "cid": "0oam2dvbpPl2IMKuQ5d6",
   "uid": "00u36oy8fCxxxPWhf5d6",
   "scp": [
      "offline_access",
      "openid",
      "fakebookapi.read",
      "profile",
      "fakebookapi.admin",
      "email"
   ],
   "sub": "shetty.viraj@gmail.com"
}
```

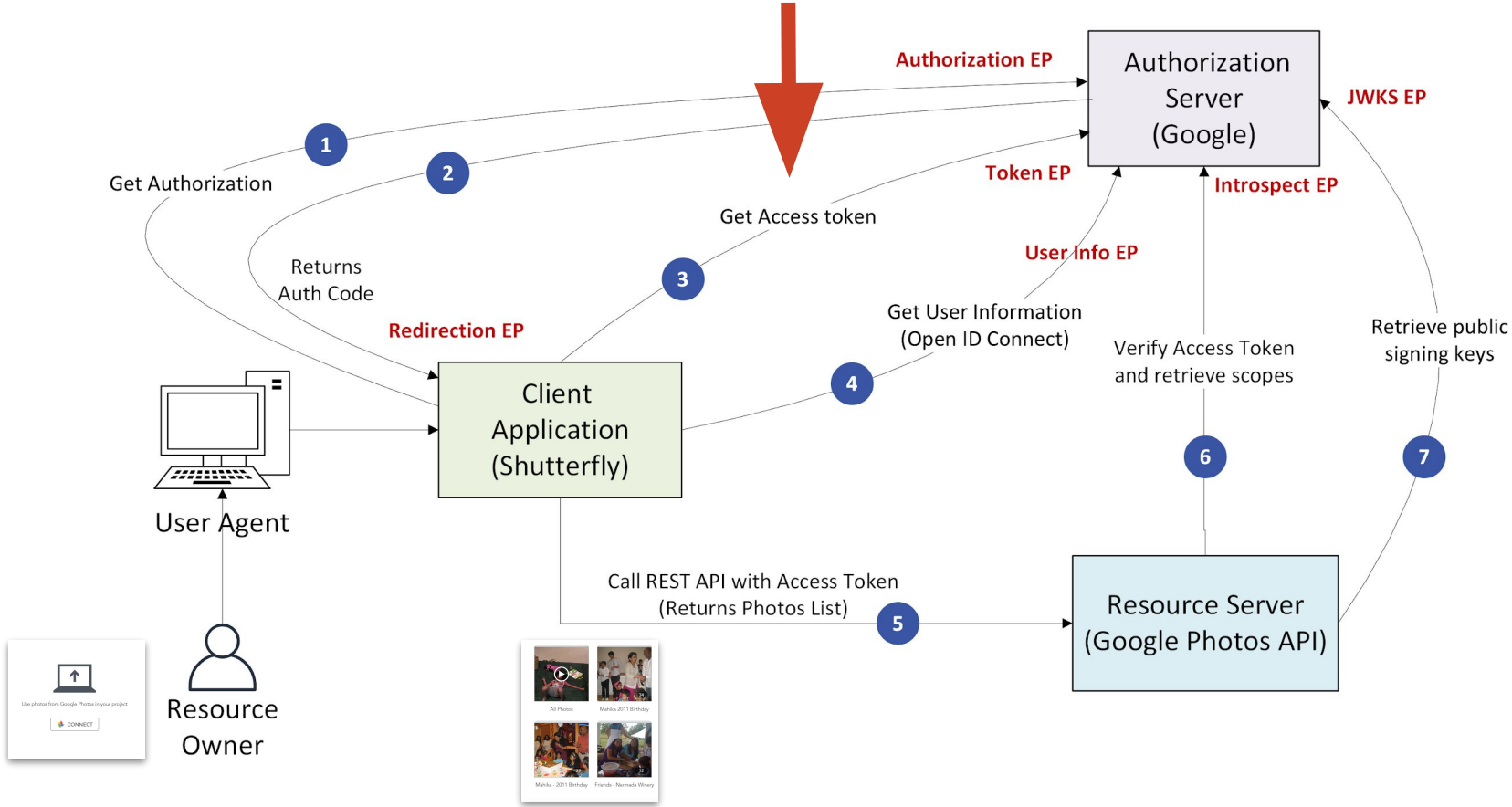
# OAuth 2.0 - OAuth Scopes



# OAuth 2.0 - OAuth Scopes

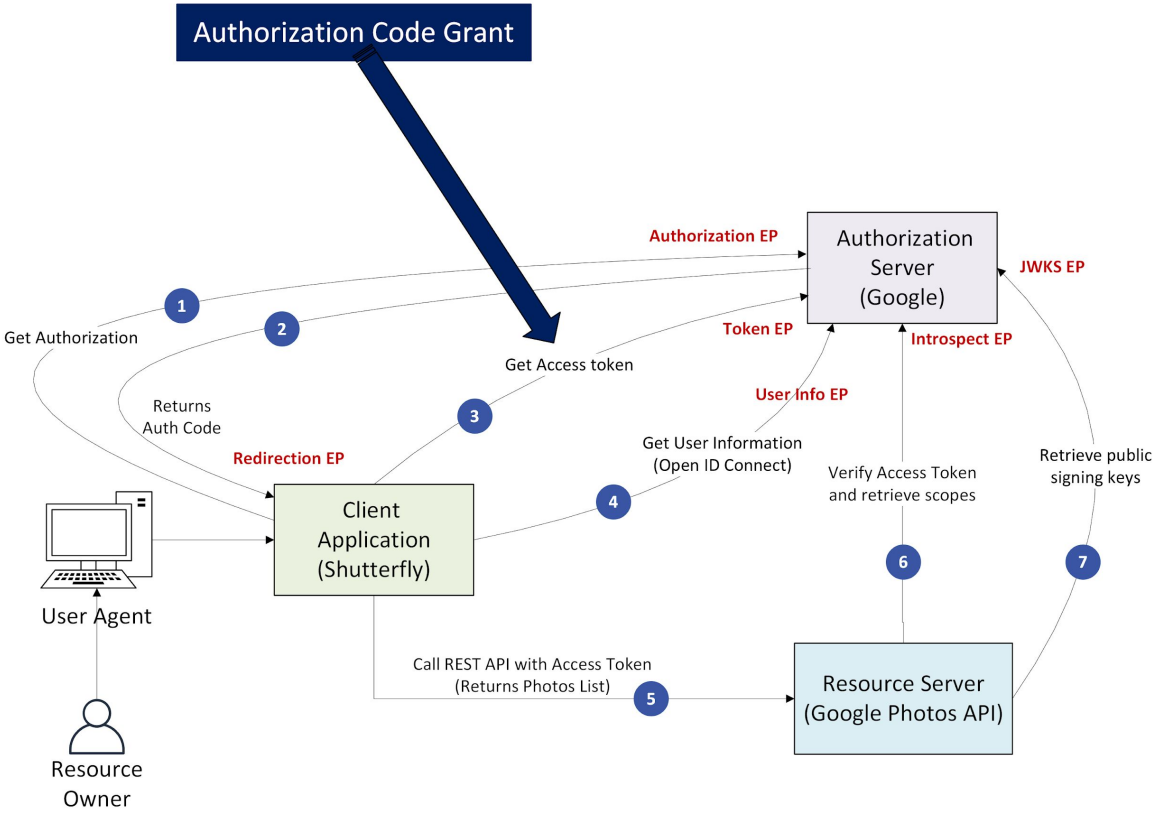


# OAuth 2.0 - Endpoints (EP)

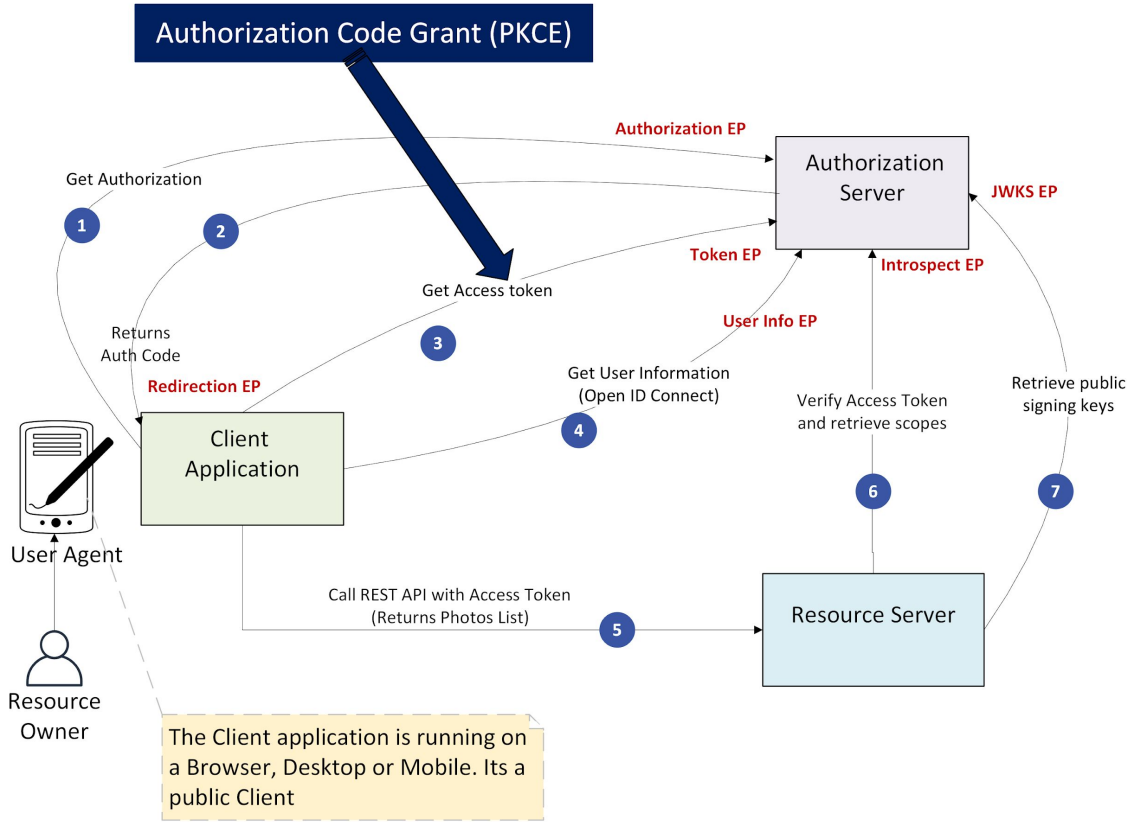


# Grant Types

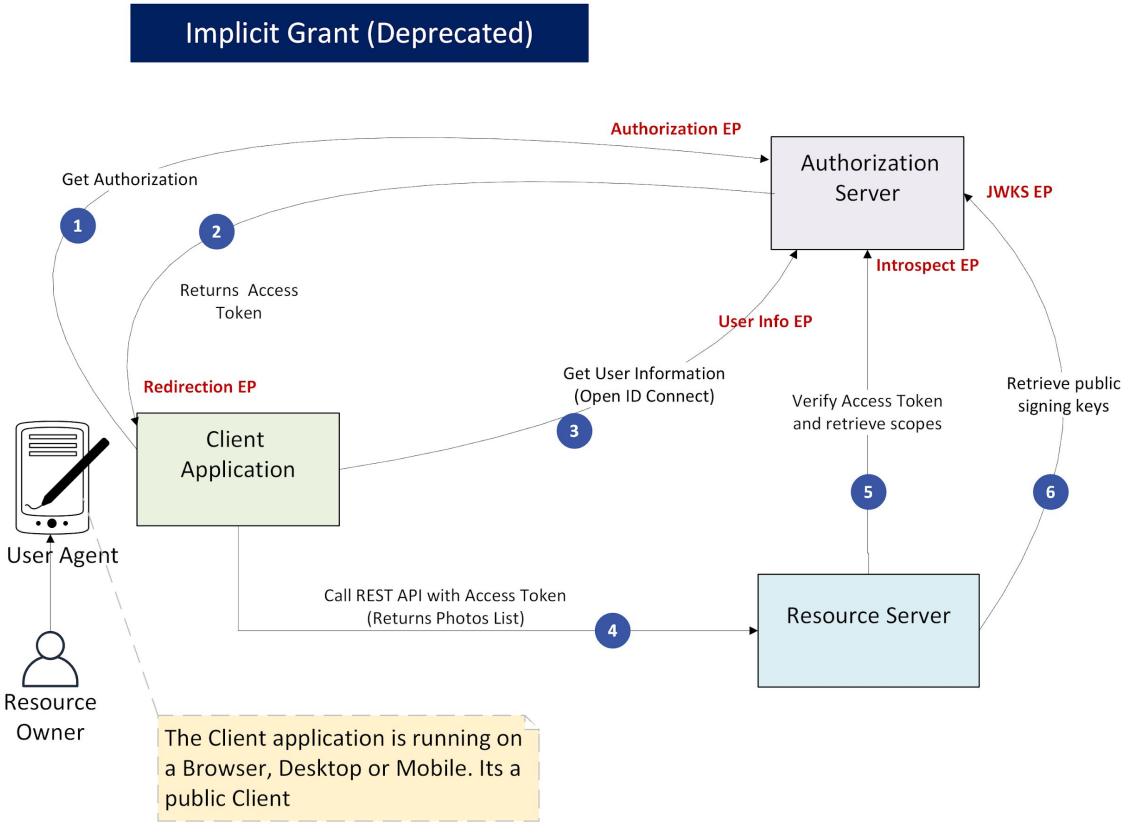
# OAuth 2.0 - Authorization Code



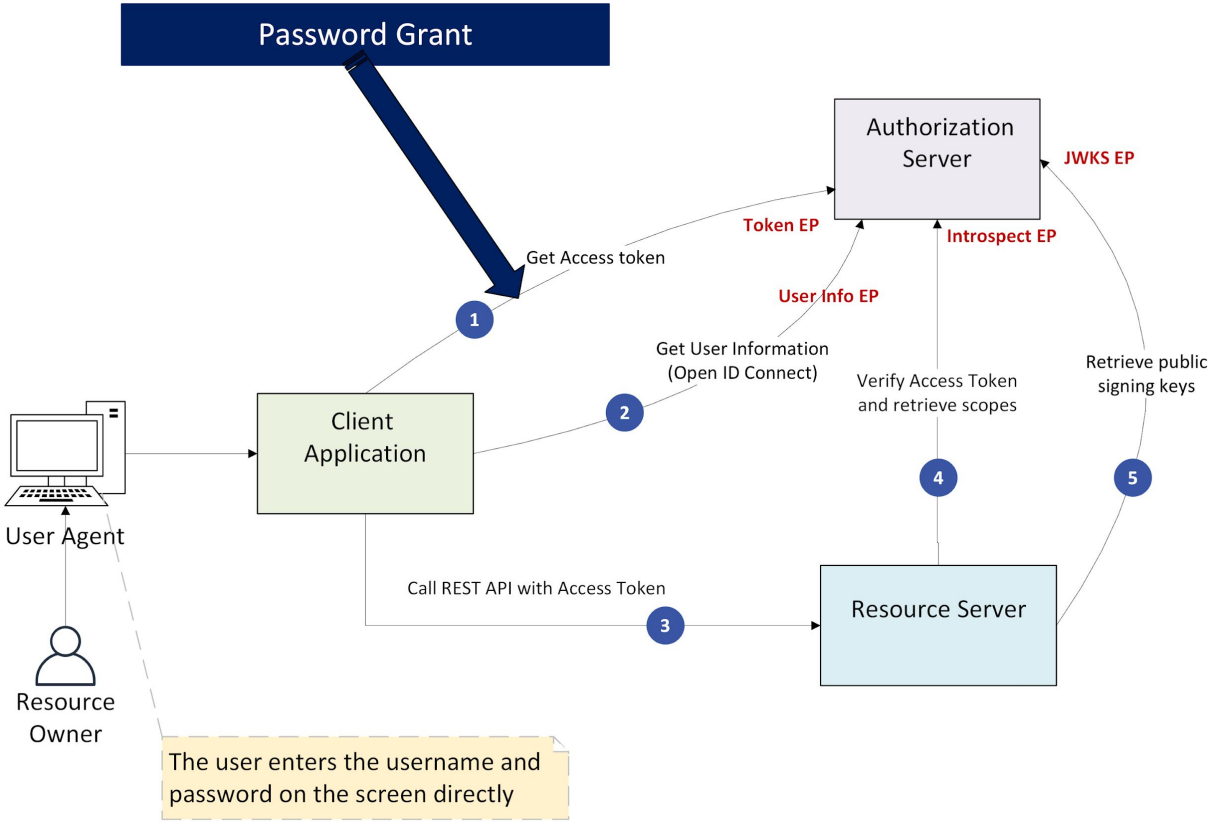
# OAuth 2.0 - Authorization Code (PKCE)



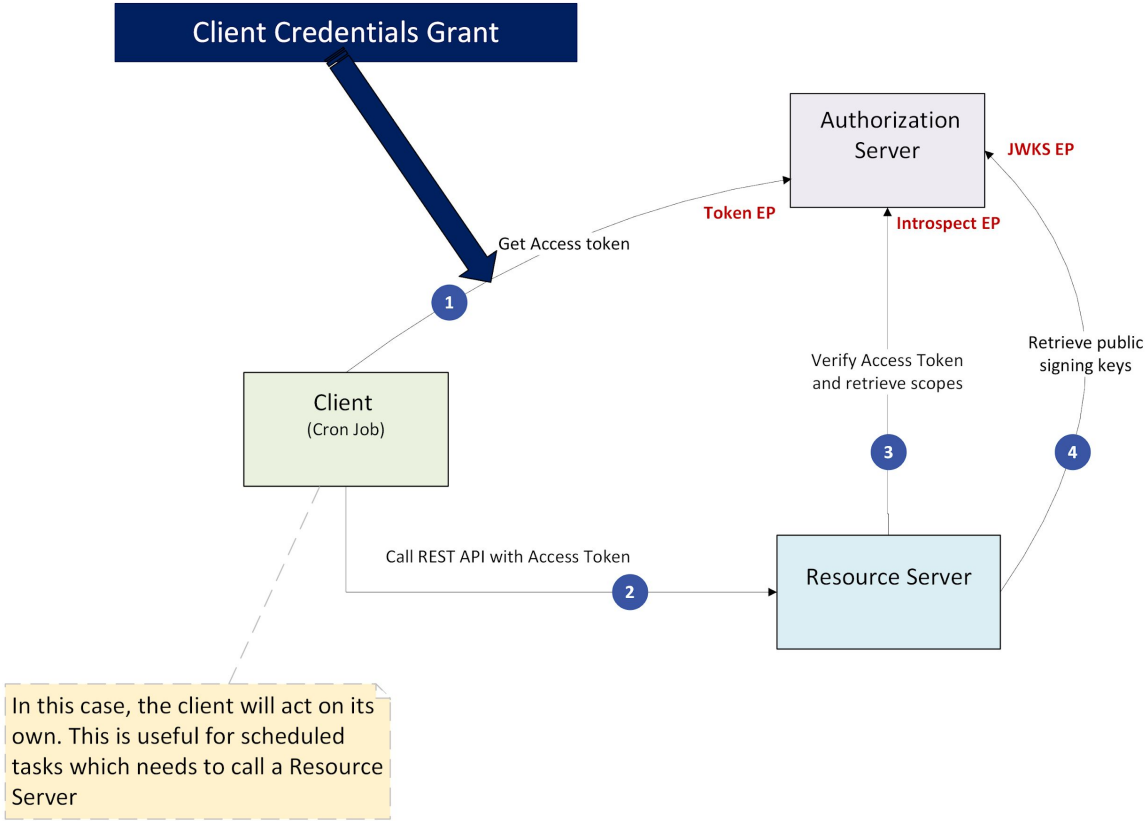
# OAuth 2.0 - Implicit



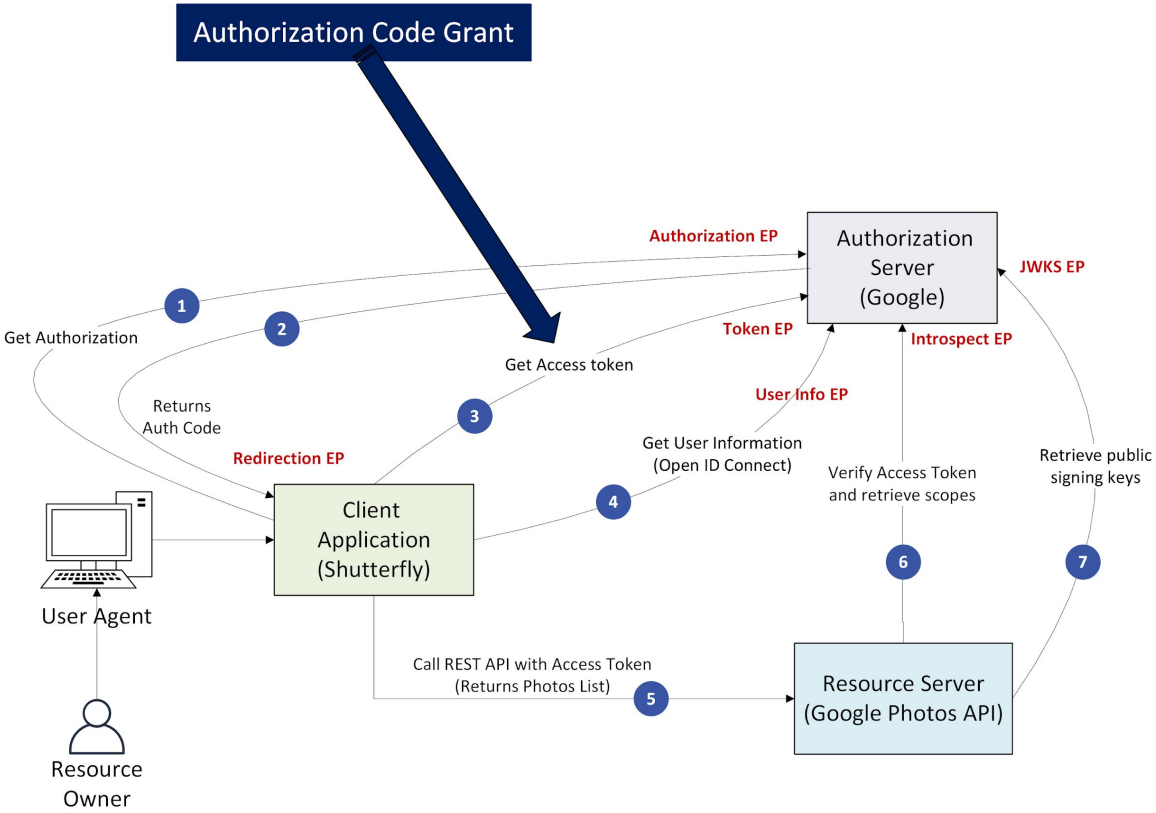
# OAuth 2.0 - Resource Owner Password Credentials



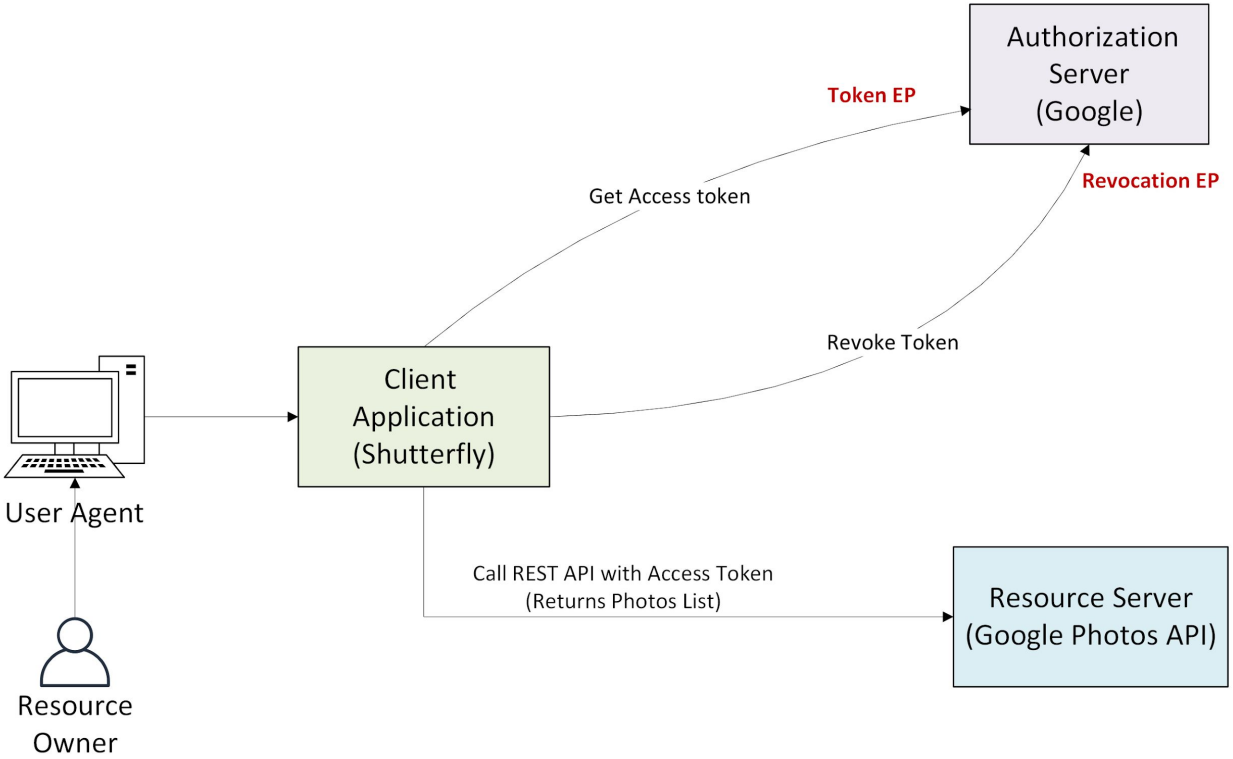
# OAuth 2.0 - Client Credentials



# OAuth 2.0 - Refresh Token



# OAuth 2.0 - Token Revocation



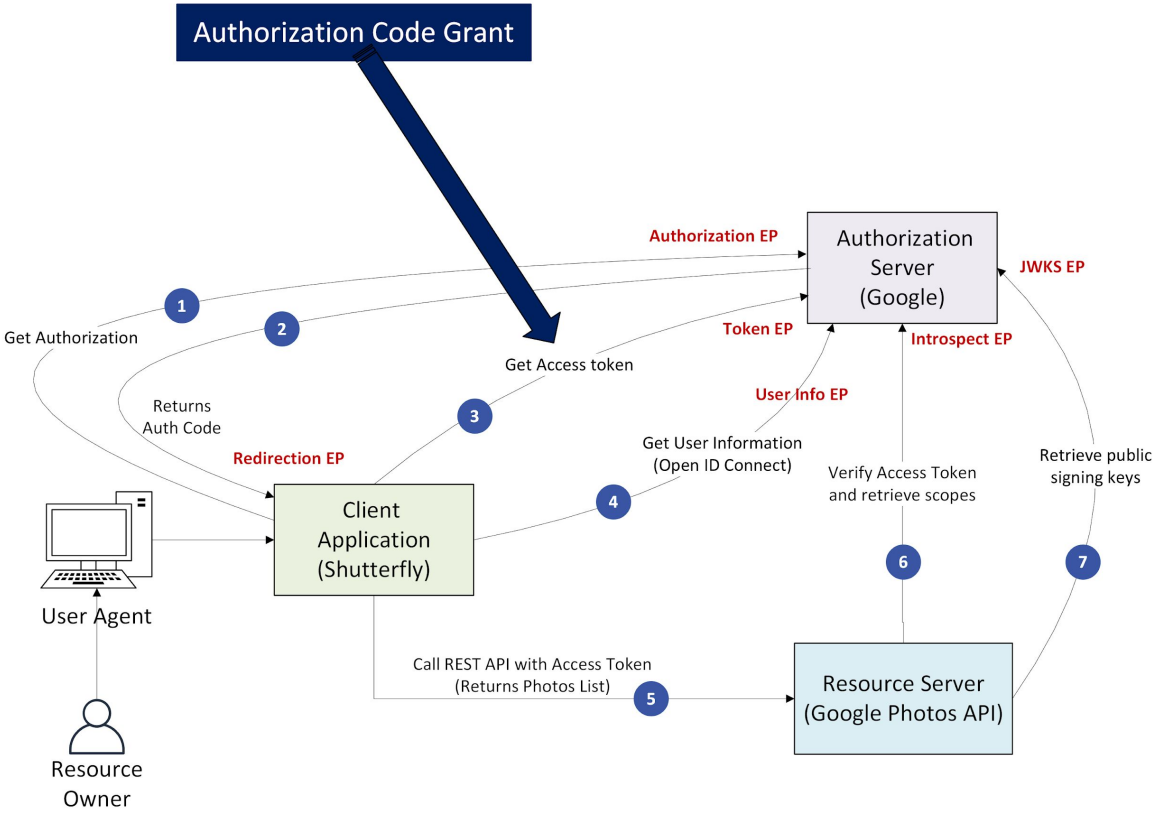
# OpenID Connect

# OAuth 2.0 - JWT Token Example

eyJraWQiOiJoWDBaVXliU3pzRjVLVVA4azltdGtDTkEy
X2l6WC03XzZuU1JTVFphLVRjIiwiYWxnIjoiUlMyNTYif
Q.eyJ2ZXIiOjEsImp0aSI6IkFULjY5dUl4V2w4Qi1KcTV
ueXg5SzgyclVQVENhbHdkLXJ4S3dZUmlEZ1RWZFk
ub2FyMWl3eXp5UWFrU2VyT2Y1ZDYiLCJpc3MiOiJo
dHRwczovL2Rldi0yMTQ4MjczLm9rdGEuY29tL29hdX
RoMi9kZWZhdWx0IiwiYXVkIjoiYXBpOi8vZGVmYXVs
dCIsImlhdCI6MTYxOTk4NjQ5MiwiZXhwIjoxNjE5OTg2
NzkyLCJjaWQiOiIwb2FtMmR2YnBQbDJJTUt1UTVkNi
IsInVpZCI6IjAwdTM2b3k4ZkN4eHhQV2hmNWQ2Iiwic
2NwIjpbIm9mZmxpbmVfYWNjZXNzIiwib3BlbmlkIiwiZ
mFrZWJvb2thcGkucmVhZCIsInByb2ZpbGUiLCJmYWt
lYm9va2FwaS5hZG1pbiIsImVtYWlsIl0sInN1YiI6InNoZ
XR0eS52aXJhakBnbWFpbC5jb20ifQ.FoDNU5IW48Jz
dpFwSDQmiA-c_IF23c-oEBjQuJ3X298RAW8ETbL8-5
hhcGV8n4g2kaH33xSaSHa5CZpZfVkfK4hmSb__vTe
DPPedWFfeypTu5kjTVPgsefdIxpUrkyKEd0ci9Q2sonf
R5G_7NW-e2jKXVMLDbWwAUmouKB1207VQQfmIb
VNPway9ZG-yg_j_XR42KMX1RcYTys53yQ71qweoQ
9e2xPynC2DNHGIL8fa060RtVTRQ-dUTn4EUdLjdvpv
EiioMnrzcZjfoxG7SUr5hwGLo_kygf0O4-PDL_I3-sU_o
zEnb7m6HZwxC2tPtA8w7qcv7harY61_4Ygmbvw


```
{
   "ver": 1,
   "jti": "AT.69uB8WI8B-Jq5nyx9K82rUPTCalwd-rxKwYRiDgTVdY.oar1iwyzyQakSerOf5d6",
   "iss": "https://dev-2148273.okta.com/oauth2/default",
   "aud": "api://default",
   "iat": 1619986492,
   "exp": 1619986792,
   "cid": "0oam2dvbpPl2IMKuQ5d6",
   "uid": "00u36oy8fCxxxPWhf5d6",
   "scp": [
      "offline_access",
      "openid",
      "fakebookapi.read",
      "profile",
      "fakebookapi.admin",
      "email"
   ],
   "sub": "shetty.viraj@gmail.com"
}
```

# OAuth 2.0 - Authorization Code



# OpenID Connect Core

## OpenID Scopes and Token

- **openid profile email address phone**
- ID Token contains User information

## /userinfo Response (scopes : openid profile email)


```
{
    "sub": "00u36oy8fCxxxPWhf5d6",
    "locale": "en-US",
    "zoneinfo": "America/Los_Angeles",
    "name": "Viraj Shetty",
    "preferred_username": "shetty.viraj@gmail.com",
    "given_name": "Viraj",
    "family_name": "Shetty",
    "updated_at": 1627238752,
    "email": "shetty.viraj@gmail.com",
    "email_verified": true
}
```

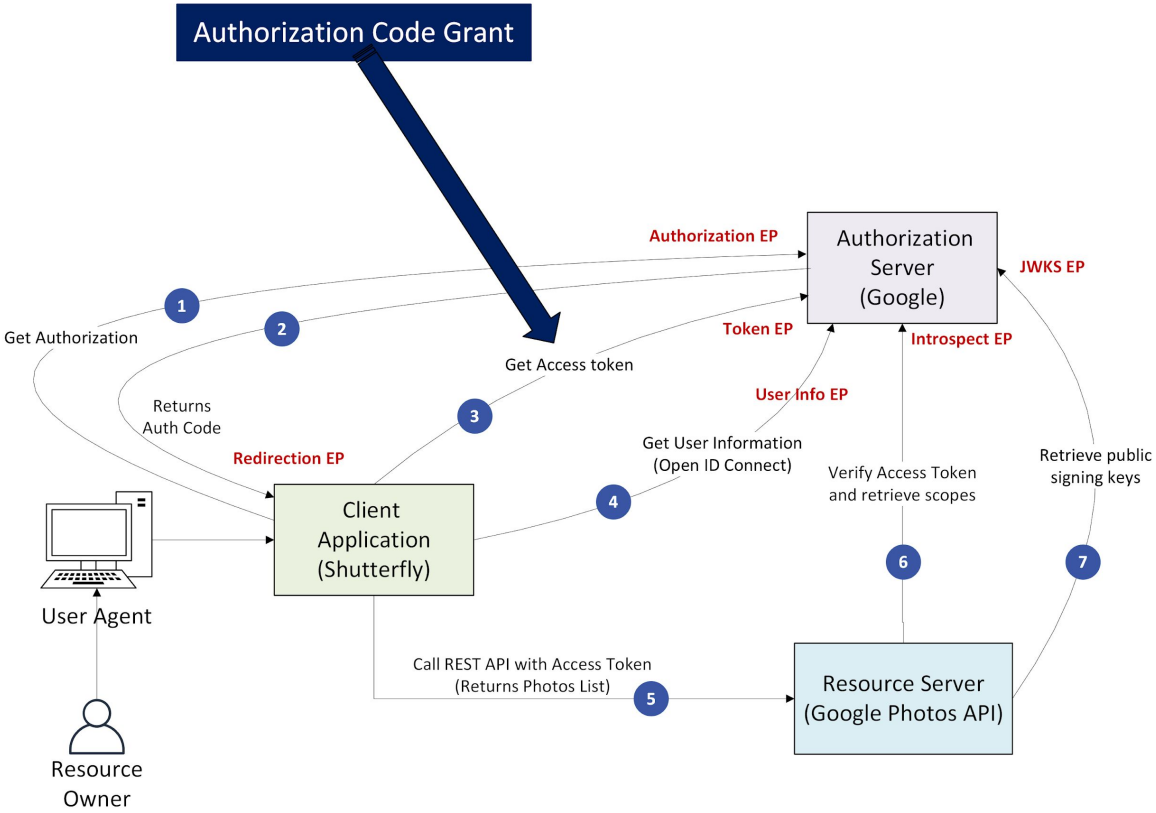

profile

email

| | | |
|---|---|---|
| Authorization Code | • Server side web applications<br>• Needs User Agent<br>• Can use Refresh tokens<br>• Very Safe | Confidential Client  |
| Implicit | • Single Page applications<br>• Needs User Agent<br>• Cannot use Refresh tokens<br>• Danger of Access token exposure<br>• Not recommended (Deprecated) | Public Client  |
| Authorization Code (PKCE extension) | • Recommended for public clients<br>• Needs User Agent<br>• Can use Refresh tokens | Public Client<br>Confidential Client  |
| Client Credential | • Use for Cron Jobs on the server<br>• No User Agent needed<br>• Cannot use Refresh tokens | Confidential Client  |
| Resource Owner Password Credentials | • Resource Server and Client must be from same Organization<br>• No User Agent needed<br>• Can use Refresh tokens<br>• Not Recommended (Deprecated) | Public Client<br>Confidential Client  |

# Grant Types : Deep Dive

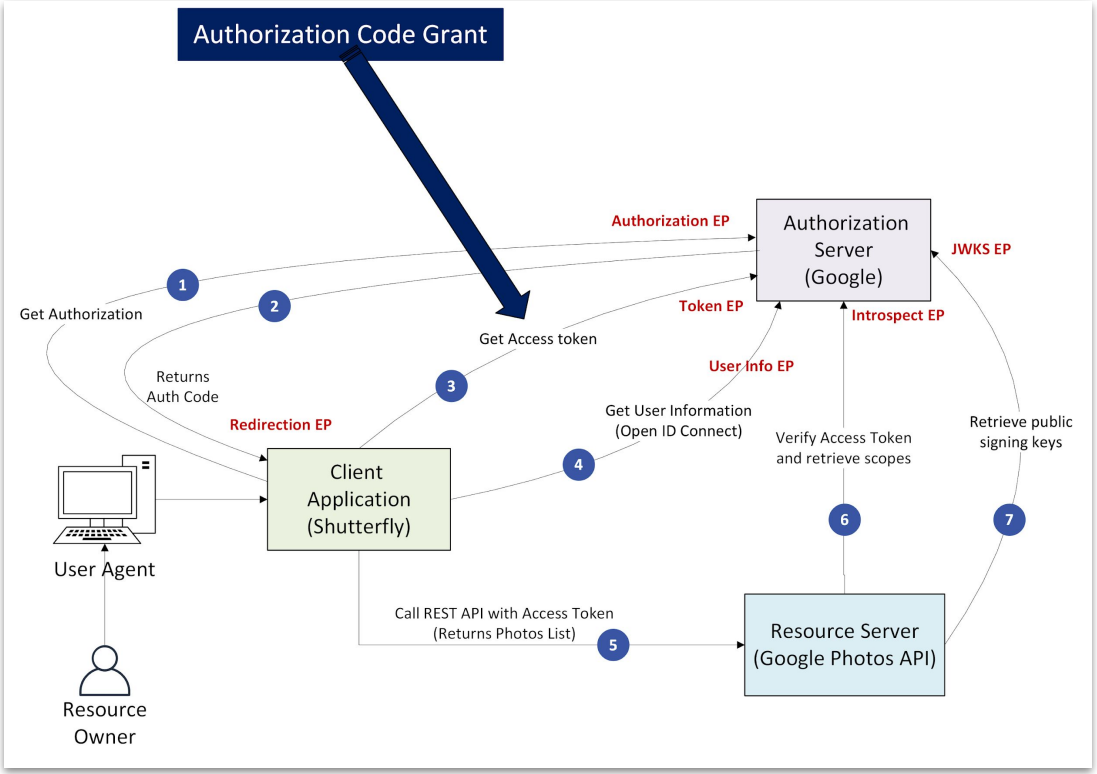
# OAuth 2.0 - Authorization Code



# OAuth 2.0 - Authorization Code

## STEPS

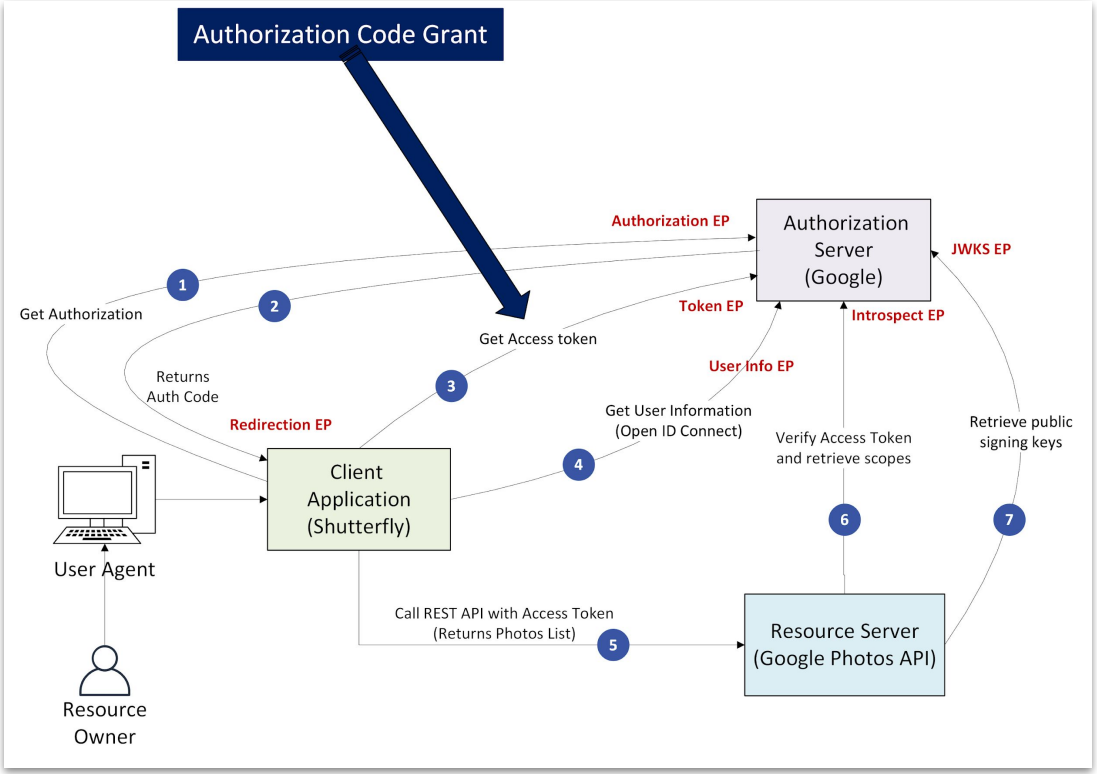
- 👉 Construct Authorization Request
- 👉 Send Request and extract Code
- 👉 Construct a Token Request
- 👉 Send Request and extract Token
- 👉 Send Google Photos Request
- 👉 Send Request and see one Album
- 👉 Send Request to see Album Photos



# OAuth 2.0 - OpenID Connect and userinfo

## MORE STEPS

- 👉 Send User Info Request
- 👉 Send request to get JWKS
- 👉 Dissect the ID Token



# OAuth 2.0 - PKCE Extension

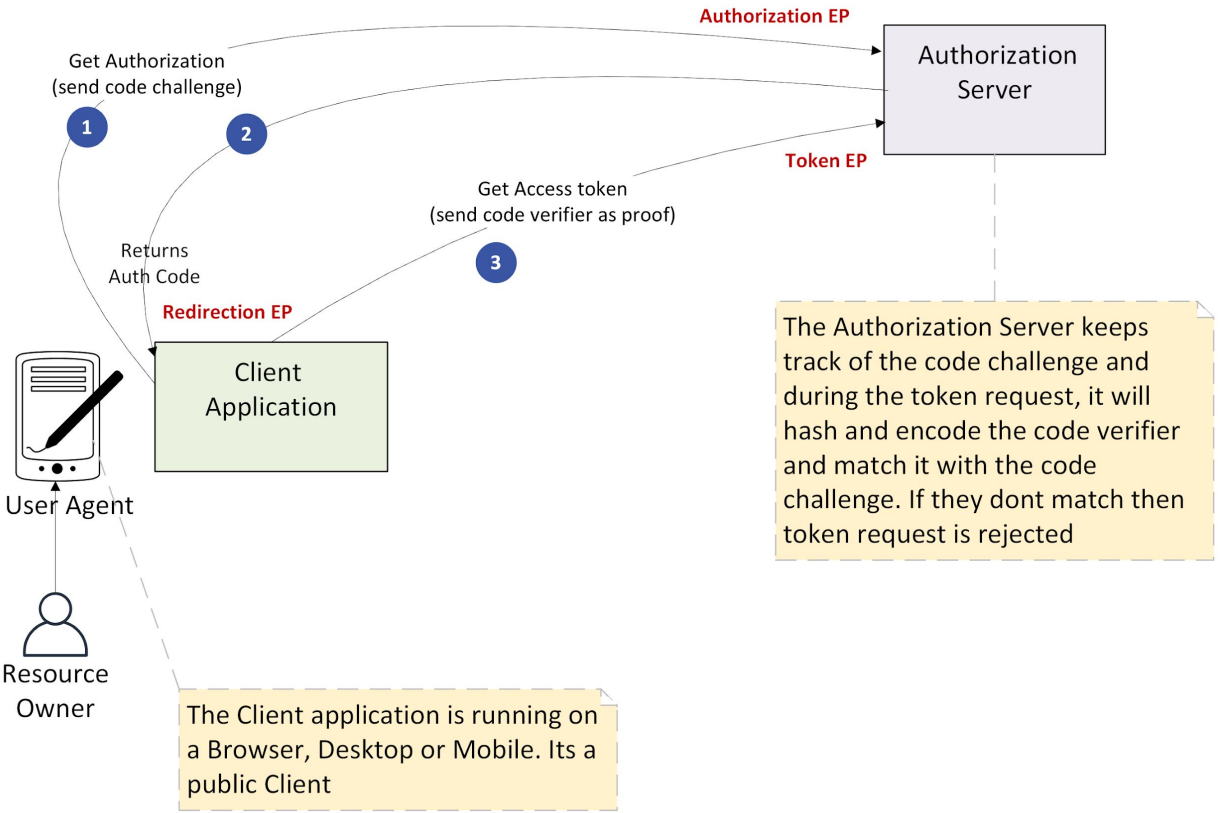
- Proof Key For Code Exchange
- Extension of the Authorization Code grant type
- Usually used by public clients

- Send **code_challenge** with authorize request

```
code_challenge = BASE64URL-ENCODE(SHA256(ASCII(code_verifier)))
```

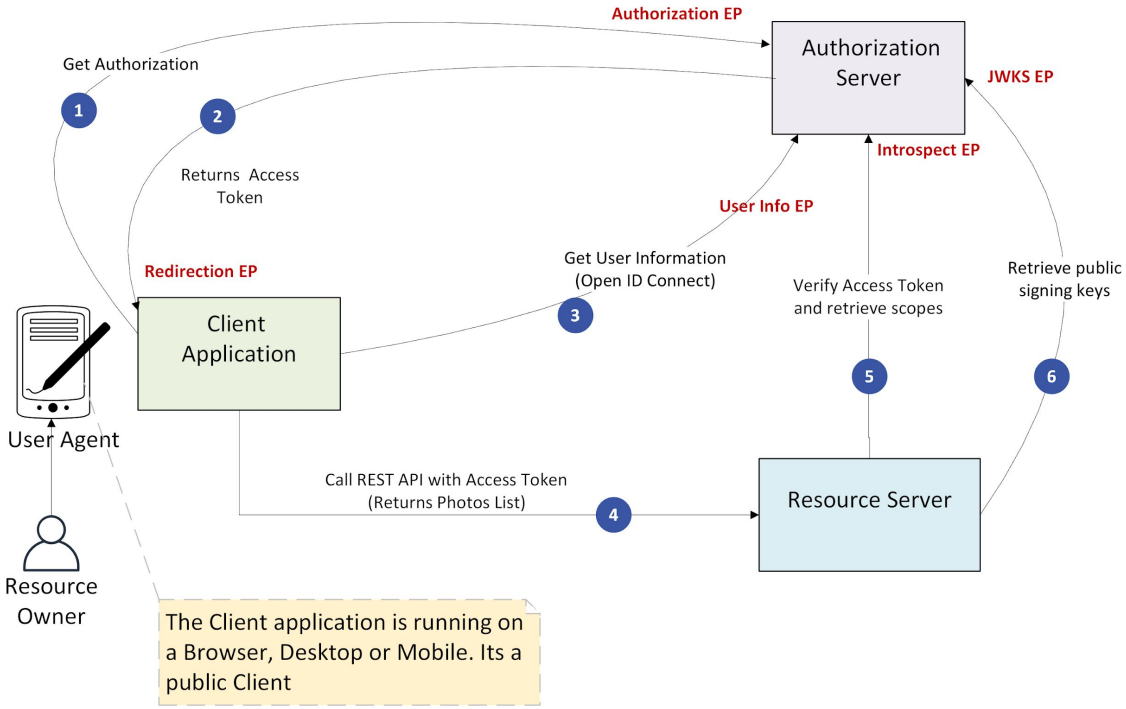
- Send **code_verifier** with token request for grant type = authorization code
  - That's the proof

# OAuth 2.0 - PKCE Extension



# OAuth 2.0 - Implicit Flow

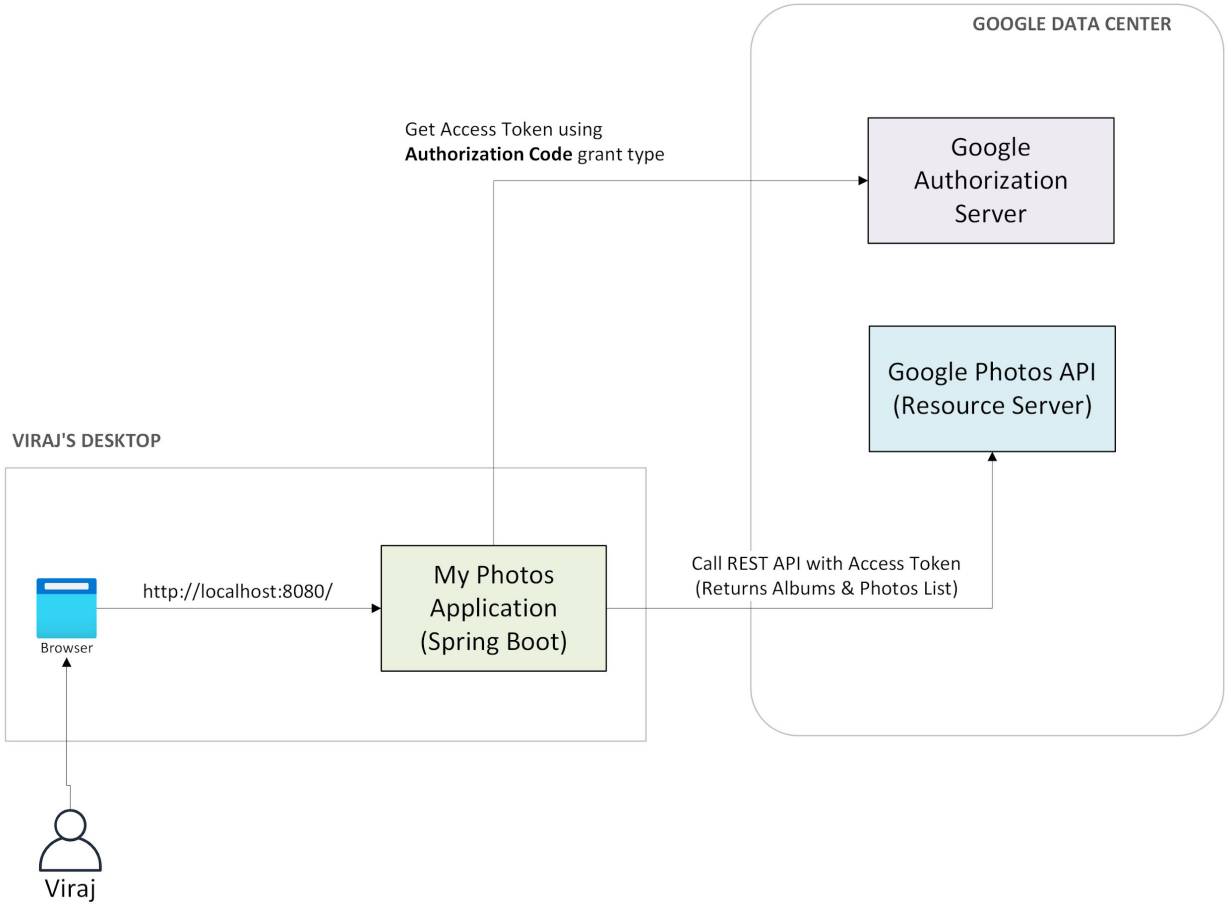
**Implicit Grant (Deprecated)**



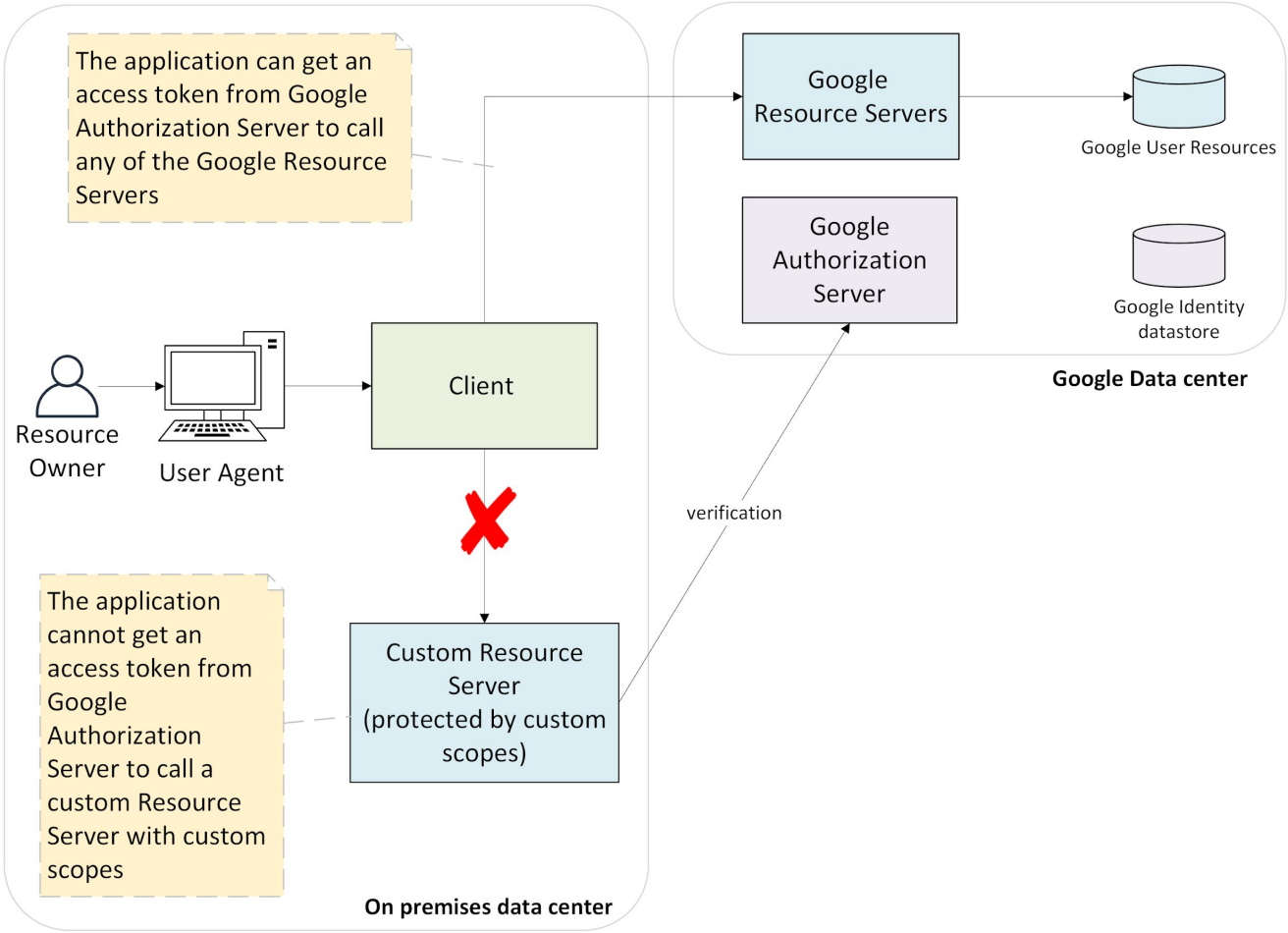
# OAuth 2.0 - Google Playground

# Google Coding Project - Demo and Setup

# Coding Project - Demo and Setup



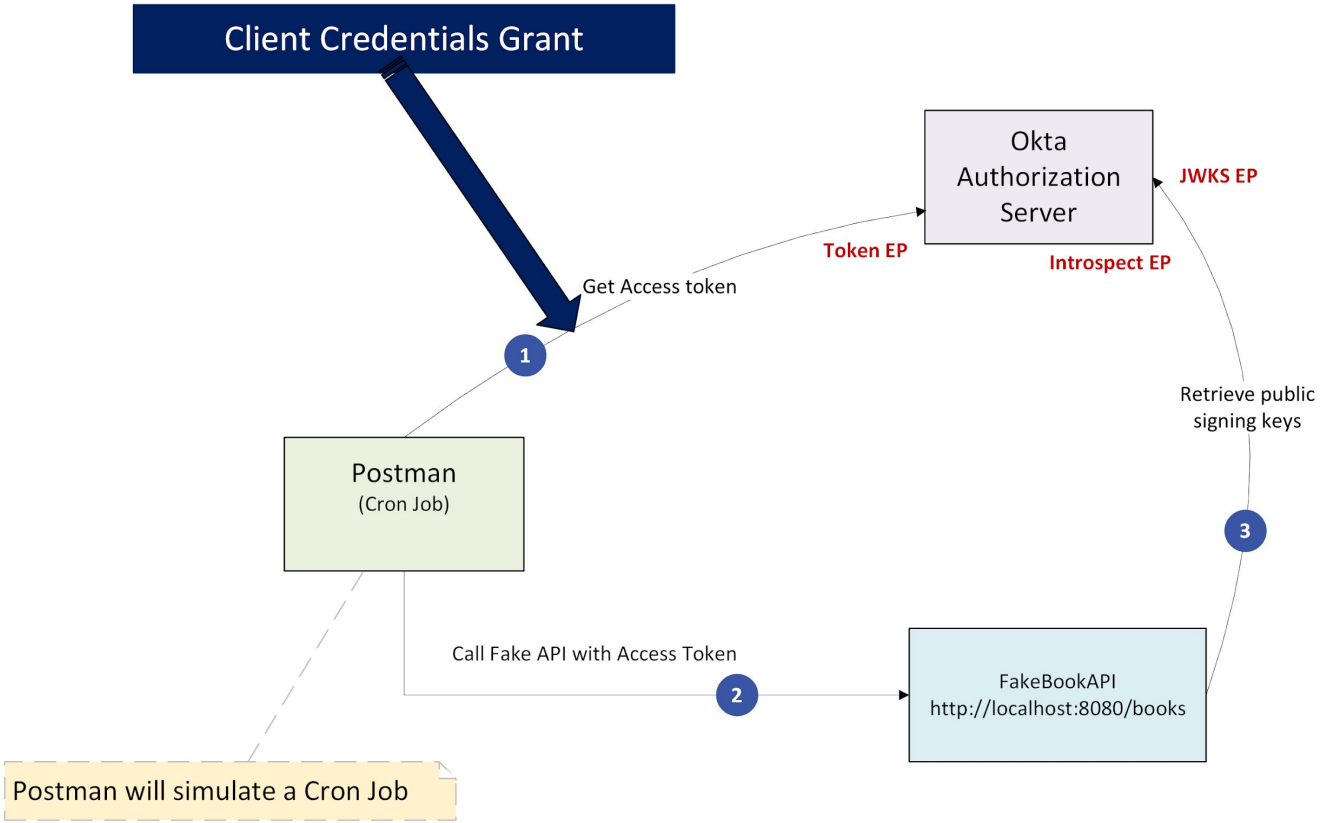
# OAuth 2.0 - Using Okta



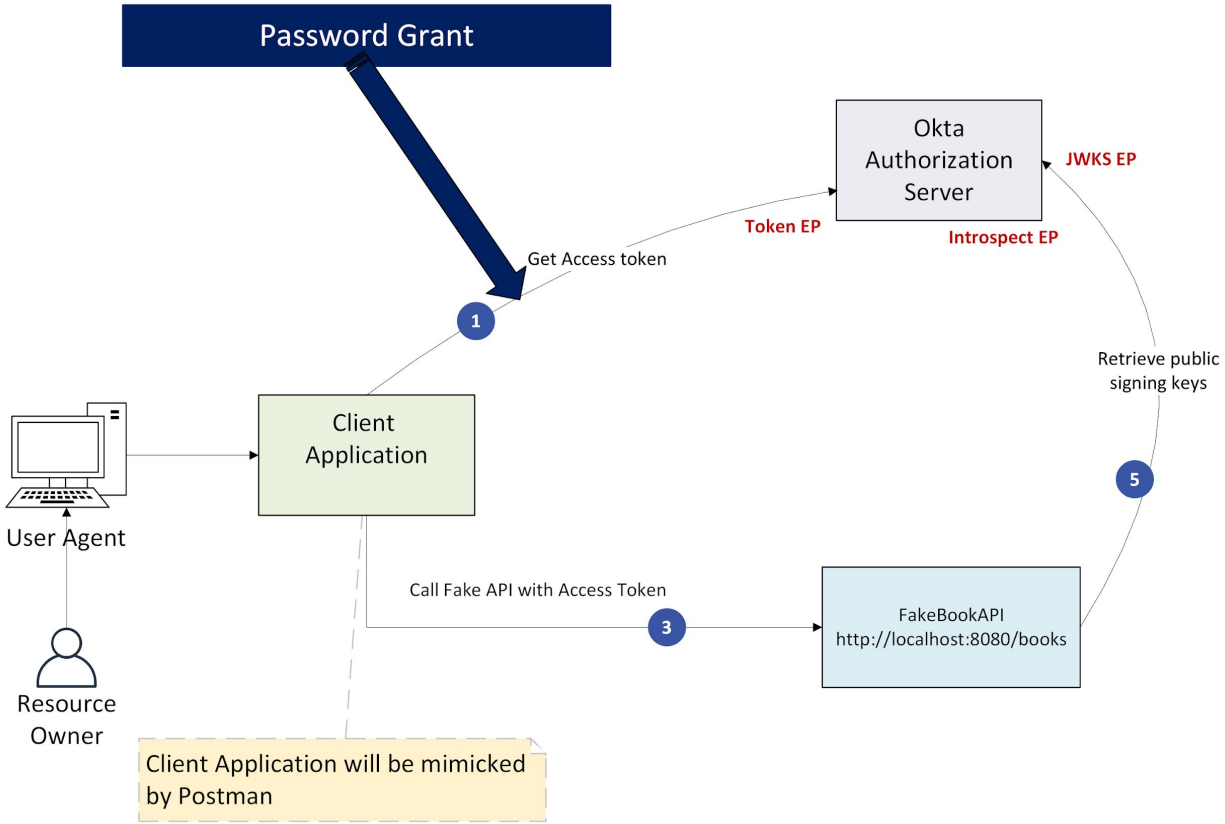
# Okta setup and Endpoints

# Okta Custom Resource Server (FakeBookAPI)

# OAuth 2.0 - Client Credentials



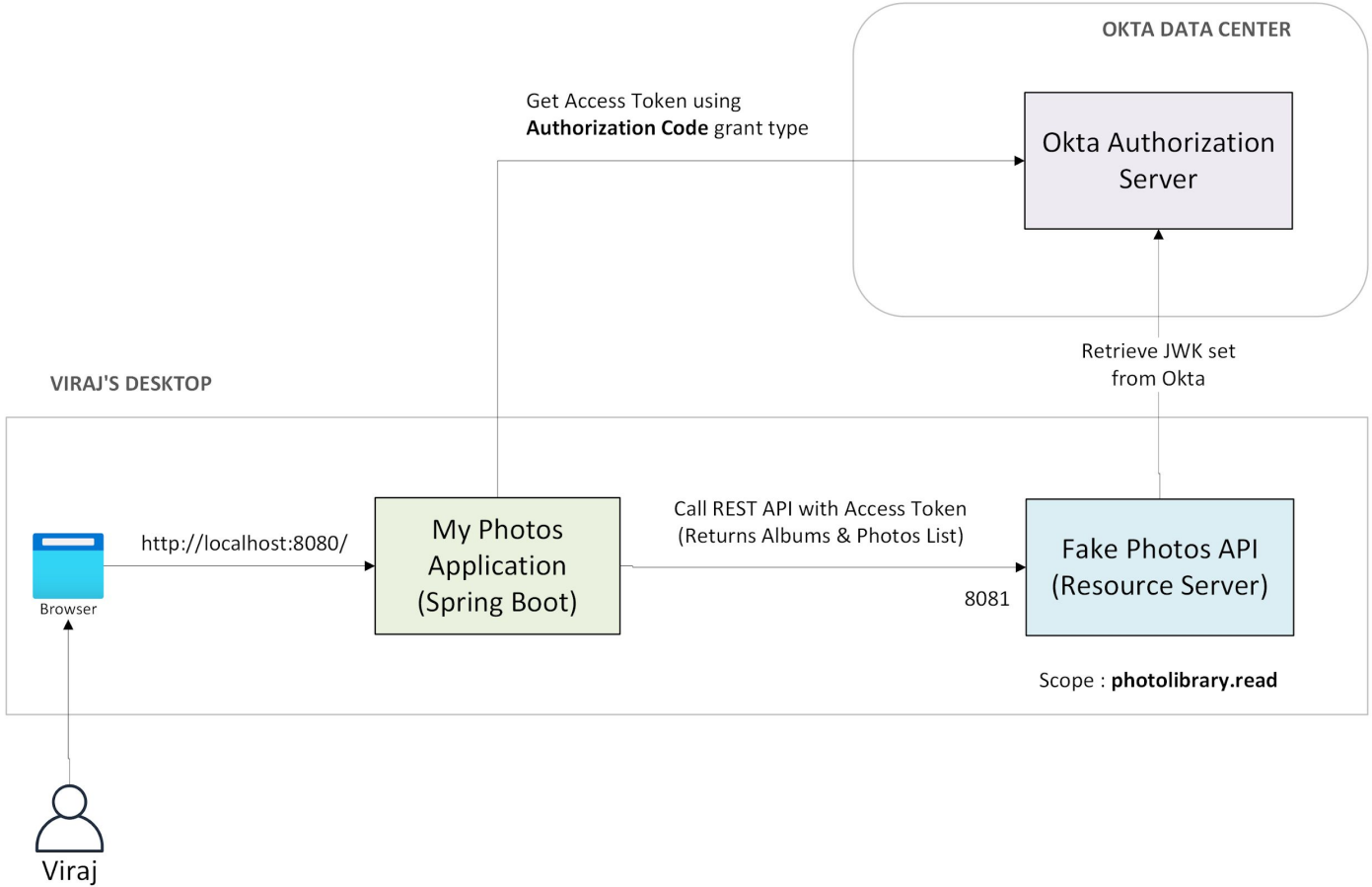
# OAuth 2.0 - Resource Owner Password



# Okta - Other Features

# Okta Coding Project - Demo and Setup

# Okta Coding Project - Demo and Setup



# JWT, Client Authentication and Other Titbits

- 👉 What's in the **JWT** ?
- 👉 Client Authentication methods
- 👉 Other Titbits

# OAuth 2.0 - JWT Token Example

eyJraWQiOiJoWDBaVXliU3pzRjVLVVA4azltdGtDTkEy
X2l6WC03XzZuU1JTVFphLVRjIiwiYWxnIjoiUlMyNTYif
Q.eyJ2ZXIiOjEsImp0aSI6IkFULjY5dUl4V2w4Qi1KcTV
ueXg5SzgyclVQVENhbHdkLXJ4S3dZUmlEZ1RWZFk
ub2FyMWl3eXp5UWFrU2VyT2Y1ZDYiLCJpc3MiOiJo
dHRwczovL2Rldi0yMTQ4MjczLm9rdGEuY29tL29hdX
RoMi9kZWZhdWx0IiwiYXVkIjoiYXBpOi8vZGVmYXVs
dCIsImlhdCI6MTYxOTk4NjQ5MiwiZXhwIjoxNjE5OTg2
NzkyLCJjaWQiOiIwb2FtMmR2YnBQbDJJTUt1UTVkNi
IsInVpZCI6IjAwdTM2b3k4ZkN4eHhQV2hmNWQ2Iiwic
2NwIjpbIm9mZmxpbmVfYWNjZXNzIiwib3BlbmlkIiwiZ
mFrZWJvb2thcGkucmVhZCIsInByb2ZpbGUiLCJmYWt
lYm9va2FwaS5hZG1pbiIsImVtYWlsIl0sInN1YiI6InNoZ
XR0eS52aXJhakBnbWFpbC5jb20ifQ.FoDNU5IW48Jz
dpFwSDQmiA-c_IF23c-oEBjQuJ3X298RAW8ETbL8-5
hhcGV8n4g2kaH33xSaSHa5CZpZfVkfK4hmSb__vTe
DPPedWFfeypTu5kjTVPgsefdIxpUrkyKEd0ci9Q2sonf
R5G_7NW-e2jKXVMLDbWwAUmouKB1207VQQfmIb
VNPway9ZG-yg_j_XR42KMX1RcYTys53yQ71qweoQ
9e2xPynC2DNHGIL8fa060RtVTRQ-dUTn4EUdLjdvpv
EiioMnrzcZjfoxG7SUr5hwGLo_kygf0O4-PDL_I3-sU_o
zEnb7m6HZwxC2tPtA8w7qcv7harY61_4Ygmbvw


```
{
   "ver": 1,
   "jti": "AT.69uB8WI8B-Jq5nyx9K82rUPTCalwd-rxKwYRiDgTVdY.oar1iwyzyQakSerOf5d6",
   "iss": "https://dev-2148273.okta.com/oauth2/default",
   "aud": "api://default",
   "iat": 1619986492,
   "exp": 1619986792,
   "cid": "0oam2dvbpPl2IMKuQ5d6",
   "uid": "00u36oy8fCxxxPWhf5d6",
   "scp": [
      "offline_access",
      "openid",
      "fakebookapi.read",
      "profile",
      "fakebookapi.admin",
      "email"
   ],
   "sub": "shetty.viraj@gmail.com"
}
```

## Encoded PASTE A TOKEN HERE


```
eyJraWQiOiI3dkZCVmdQM0dIRVB4NXdLbUgtTTZ
YTDJlOExNWHBjTFVmRFk1ZWctU2YwIiwiYWxnIj
oiUlMyNTYifQ.eyJ2ZXIiOjEsImp0aSI6IkFULn
FNaTNfODVpWXBrT2huQVd6ZkJSQzdWdnV4ak1NQ
zVDWnp6QjU2LVFQM3MiLCJpc3MiOiJodHRwczov
L2Rldi0yMTQ4MjczLm9rdGEuY29tL29hdXRoMi9
kZWZhdWx0IiwiYXVkIjoiYXBpOi8vZGVmYXVsdC
IsImlhdCI6MTY2ODg3MTc4NCwiZXhwIjoxNjY4O
DcyMDg0LCJjaWQiOiIwb2EzcmFnYmxuMVppa3BQ
eTVkNyIsInNjcCI6WyJmYWtlYm9va2FwaS5yZWF
kIiwiZmFrZWJvb2thcGkuYWRtaW4iXSwic3ViIj
oiMG9hM3JhZ2JsbjFaaWtwUHk1ZDcifQ.aJoaU9
8T7UKcH9aXvnIySDWCKrmYRZDh6C_Zqxwott5k4
A55X6gy5m3nOC9vuOW0oRGTe1HhNHgWZcJY9o5M
exsjDTxG0PIX0N409B8JN45w8kIkBUizWZRwBIL
yLeqENMn205R17Tp-
OlGR9d_YMyAWSM05hFYgWh2y1cf-
V49axDXppE8gDNSznYqdsfJRKdeKOjqjbuipcPK
mdzmXS-
teIzLfaM0Kzz9fSoX0dBvvi25qfG4y8lPADF1RH
aTf48hCh35-
a0gxZwAjhA1b2kIGOeaslkGvoBM23e4ujikhhTF
5w17p_FdAapl01YSNwoSFgTkW_257yGGG8REHXQ
```


## Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "7vFBVgP3GHEPx5wKmH-M6XL2e8LMXpcLUfDY5eg-Sf0",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
{
  "ver": 1,
  "jti": "AT.qMi3_85iYpkOhnAWzfBRC7VvuxjMMC5CZzzB56-
QP3s",
  "iss": "https://dev-2148273.okta.com/oauth2/default",
  "aud": "api://default",
  "iat": 1668871784,
  "exp": 1668872084,
  "cid": "0oa3ragbln1ZikpPy5d7",
  "scp": [
    "fakebookapi.read",
    "fakebookapi.admin"
  ],
  "sub": "0oa3ragbln1ZikpPy5d7"
}
```


VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  {
    "e": "AQAB",
```

# {encoded header}.{encoded payload}.{encoded signature}

Base64 URL Encoded

Base64 URL Encoded

Base64 URL Encoded


```
{
    "kid":
"7vFBVgP3GHEP....",
    "alg": "RS256"
}
```


**Header**


```
{
  "ver": 1,
  "jti": "AT.qMi3_85iYpkO....",
  "iss": "https://dev-214827...",
  "aud": "api://default",
  "iat": 1668871784,
  "exp": 1668872084,
  "cid": "0oa3ragbln1ZikpPy5d7",
  "scp": [
    "fakebookapi.read",
    "fakebookapi.admin"
  ],
  "sub": "0oa3ragbln1ZikpPy5d7"
}
```


**Payload**

Compute the Signature from **{Encoded header}.{Encoded payload}**

**Signature**

# Signing Algorithms Supported

"request_object_signing_alg_values_supported": [

"HS256",
"HS384",
"HS512",

HMAC with SHA-NNN (Use Secret Key)

"RS256",
"RS384",
"RS512",

RSA with SHA-NNN (Use Public/Private Key)

"ES256",
"ES384",
"ES512"

ECDSA with SHA-NNN (Use Public/Private Key)

]

# {encoded header}.{encoded payload}.{encoded signature}

Base64 URL Decoded

```
{
    "kid":
"7vFBVgP3GHEP....",
    "alg": "RS256"
}
```

**Header**

Base64 URL Decoded

```
{
  "ver": 1,
  "jti": "AT.qMi3_85iYpkO....",
  "iss": "https://dev-214827...",
  "aud": "api://default",
  "iat": 1668871784,
  "exp": 1668872084,
  "cid": "0oa3ragbln1ZikpPy5d7",
  "scp": [
    "fakebookapi.read",
    "fakebookapi.admin"
  ],
  "sub": "0oa3ragbln1ZikpPy5d7"
}
```

**Payload**

Base64 URL Decoded

The Signature is Verified by the recipient using **RS256** (extracted from header)

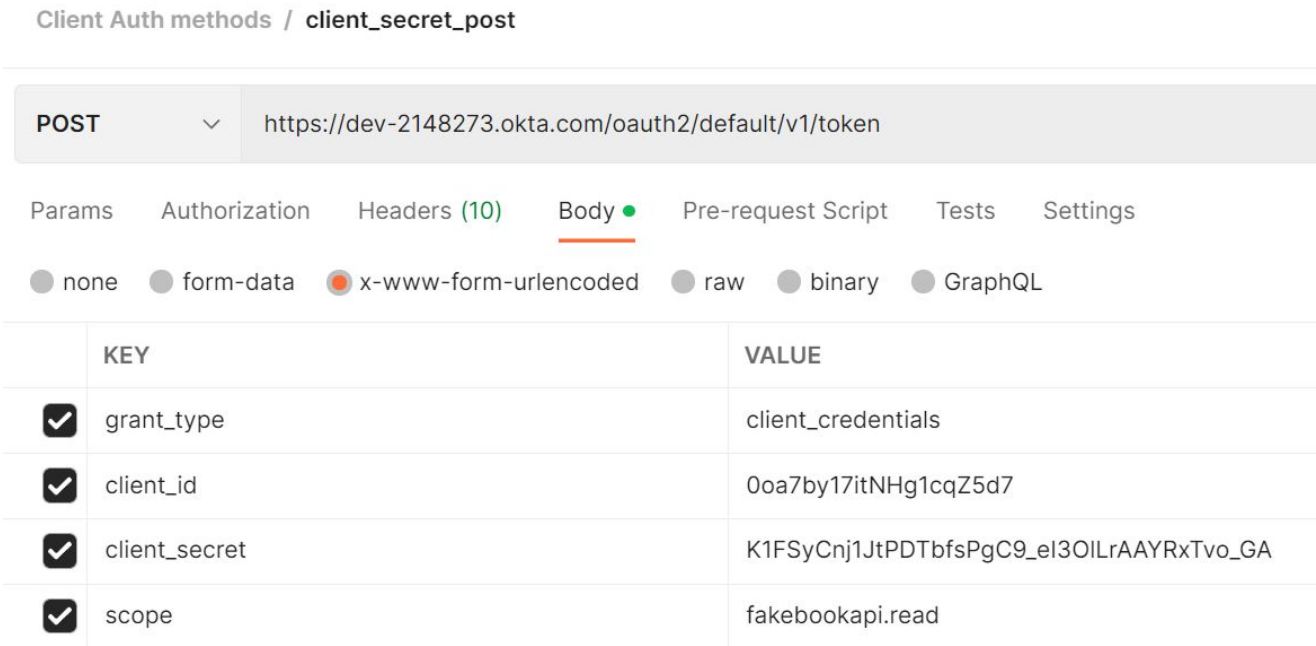
**Signature**

# Client Authentication

# Client Authentication Methods

- 👉 `Client_secret_post` (We have used this)
- 👉 `client_secret_basic`
- 👉 `client_secret_jwt`
- 👉 `private_key_jwt`
- 👉 `none`

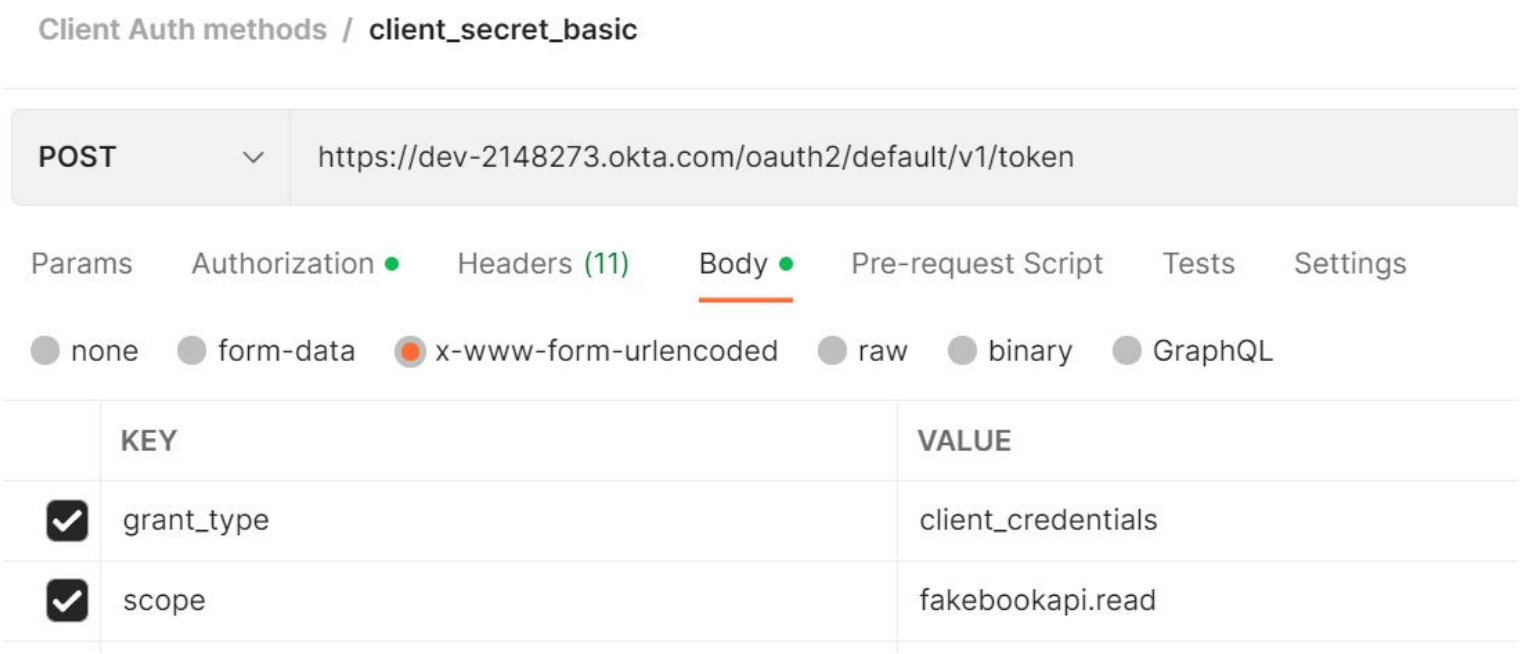
# client_secret_post



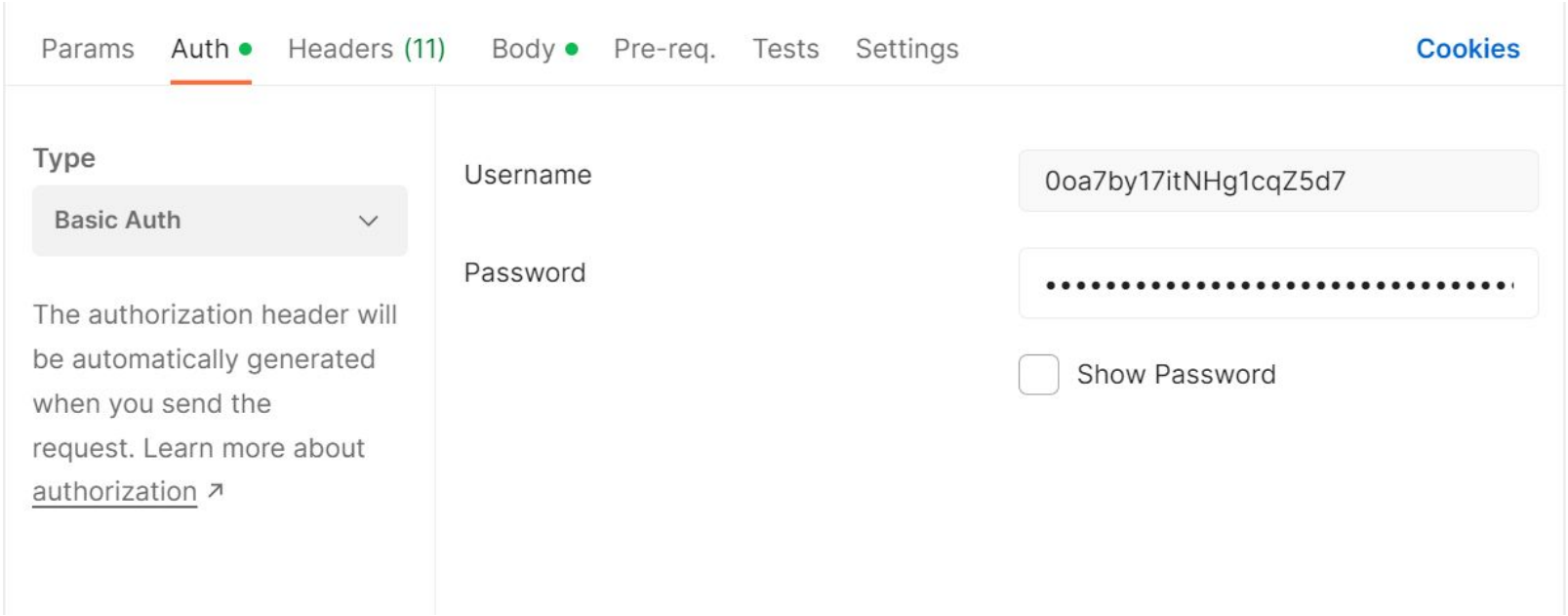
# client_secret_post

```
1  POST /oauth2/default/v1/token HTTP/1.1
2  Host: dev-2148273.okta.com
3  Content-Type: application/x-www-form-urlencoded
4  Cookie: DT=DI1uhgS4QsFQYm1nd4C5DnzGw; JSESSIONID=E4FF4CF8A4FB472E4FBF78C6A0C72724
5  Content-Length: 138
6
7  grant_type=client_credentials&client_id=0oa7by17itNHg1cqZ5d7&
       client_secret=K1FSyCnj1JtPDTbfsPgC9_eI3OlLrAAYRxTvo_GA&scope=fakebookapi.read
```

# client_secret_basic



# client_secret_basic



# client_secret_basic

```
1  POST /oauth2/default/v1/token HTTP/1.1
2  Host: dev-2148273.okta.com
3  Authorization: Basic
       MG9hN2J5MTdpdE5IZzFjcVo1ZDc6SzFGU3lDbmoxSnRQRFRiZnNQZ0M5X2VJM09sTHJBQVlSeFR2b19
       QQ==
4  Content-Type: application/x-www-form-urlencoded
5  Cookie: DT=DI1uhgS4QsFQYm1nd4C5DnzGw; JSESSIONID=7ED92357CB89857DB4BA4C1503F75347
6  Content-Length: 52
7
8  grant_type=client_credentials&scope=fakebookapi.read
```

# client_secret_jwt

## Encoded PASTE A TOKEN HERE


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JhdWQiOiJodHRwczovL2Rldi0yMTQ4MjczLm9rd
GEuY29tL29hdXRoMi9kZWZhdWx0L3YxL3Rva2Vu
IiwiaXNzIjoiMG9hN2J5MTdpdE5IZzFjcVo1ZDc
iLCJzdWIiOiIwb2E3YnkxN2l0TkhnMWNxWjVkNy
IsImV4cCI6IjE2Njg5Njk5OTcifQ.HJOtT7iBa9
QCy7QgtcO3UzPN6e3Waj9Xyyjz6LwuDUc
```


## Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```


PAYLOAD: DATA


```
{
  "aud": "https://dev-
2148273.okta.com/oauth2/default/v1/token",
  "iss": "0oa7by17itNHg1cqZ5d7",
  "sub": "0oa7by17itNHg1cqZ5d7",
  "exp": "1668969997"
}
```


VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  K1FSyCnj1JtPDTbfsPgC9_
) ☐ secret base64 encoded
```

# client_secret_jwt

Client Auth methods / client_secret_jwt

POST https://dev-2148273.okta.com/oauth2/default/v1/token

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

| KEY | VALUE |
| --- | --- |
| grant_type | client_credentials |
| scope | fakebookapi.read |
| client_assertion_type | urn:ietf:params:oauth:client-assertion-type:jwt-bearer |
| client_assertion | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOi... |

# client_secret_jwt

```
POST /oauth2/default/v1/token HTTP/1.1
Host: dev-2148273.okta.com
Content-Type: application/x-www-form-urlencoded
Cookie: DT=DI1uhgS4QsFQYm1nd4C5DnzGw; JSESSIONID=8852455106D9FC570A722483AE9AAB27
Content-Length: 424

grant_type=client_credentials&scope=fakebookapi.read&
    client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt
    bearer&client_assertion=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJhdWQiOiJodHRwczovL2Rldi0yMTQ4MjczLm9rdGEuY29tL29hdXRoMi9kZWZhdWx0L3YxL3Rva2V
    IiwiaXNzIjoiMG9hN2J5MTdpdE5IZzFjcVo1ZDciLCJzdWIiOiIwb2E3YnkxN2l0TkhnMWNxWjVkNyI
    ImV4cCI6IjE2Njg5Njk5OTcifQ.HJOtT7iBa9QCy7QgtcO3UzPN6e3Waj9Xyyjz6LwuDUc
```

# private_key_jwt

## Encoded PASTE A TOKEN HERE


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJodHRwczovL2Rldi0yMTQ4MjczLm9rdGEuY29tL29hdXRoMi9kZWZhdWx0L3YxL3Rva2VuIiwiaXNzIjoiMG9hN2J5MTdpdE5IZzFjcVo1ZDciLCJzdWIiOiIwb2E3YnkxN2l0TkhnMWNxWjVkNyIsImV4cCI6IjE2Njg5Njk5OTcifQ.snK-LcuPwXk8mbb3cizM6GOyCLlgrC1M4tOCkpf_NN6ad1MwQfb9ktVVyAVudKzI7jN-JfBznvTrgPcOWcCD4DgqNb8OSGsCNzhwySGSOvAoBicAkvXFBh-Tq9ZB1Vh7jQgCFptobEXMBi2jKi3K7XpW1alx3zqS5E7kdUggBpCMRxOiazxL4WfDKOEbzJdTO1iy-mEDB9sNHlNcTVmWiwHwr1cNPA_eyTt1HTLkduE9KHyYhnkr-f6Fxs15UF5mH2qS9a1W8BB9FhVoc37Mt64dyN7_HzNPbl13wo8FCqDJ6Xo385T23MLJ0q5ebwUUI_eidr-4pcfSPmtqyF0Dpg
```


## Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

Issuer (who created and signed this token)

```
{
  "aud": "https://dev-2148273.okta.com/oauth2/default/v1/token",
  "iss": "0oa7by17itNHg1cqZ5d7",
  "sub": "0oa7by17itNHg1cqZ5d7",
  "exp": "1668969997"
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  byrxSyvjUGcHMkf6zH0Ae27bYV
  GIM_aa4auoi_nXt6ynMAurASF2
  B2LeCfMbrOLrOrab4_vOOZD97E
  obxZYirqedI9KQ",
  -----BEGIN PRIVATE KEY-----
  MIIEvQIBADANBgkqhkiG9w0BAQ
  EFAASCBKcwggSjAgEAAoIBAQDl
  Plu2VChDuq6f
)
```

# private_key_jwt

Client Auth methods / client_secret_jwt

POST https://dev-2148273.okta.com/oauth2/default/v1/token

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

| | KEY | VALUE |
|---|---|---|
| ☑ | grant_type | client_credentials |
| ☑ | scope | fakebookapi.read |
| ☑ | client_assertion_type | urn:ietf:params:oauth:client-assertion-type:jwt-bearer |
| ☑ | client_assertion | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOi... |

# private_key_jwt

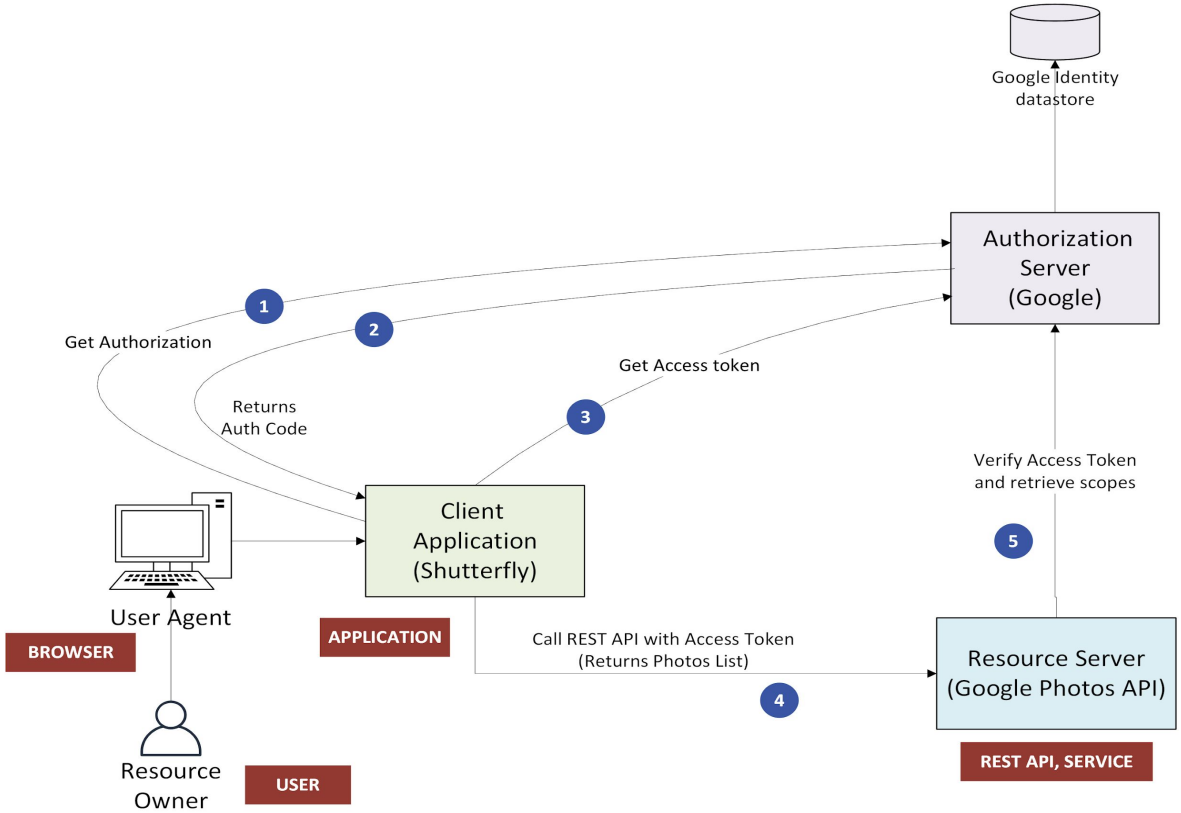
```
POST /oauth2/default/v1/token HTTP/1.1
Host: dev-2148273.okta.com
Content-Type: application/x-www-form-urlencoded
Cookie: DT=DI1uhgS4QsFQYm1nd4C5DnzGw; JSESSIONID=8852455106D9FC570A722483AE9AAB27
Content-Length: 424

grant_type=client_credentials&scope=fakebookapi.read&
    client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt
    bearer&client_assertion=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJhdWQiOiJodHRwczovL2Rldi0yMTQ4MjczLm9rdGEuY29tL29hdXRoMi9kZWZhdWx0L3YxL3Rva2V
    IiwiaXNzIjoiMG9hN2J5MTdpdE5IZzFjcVo1ZDciLCJzdWIiOiIwb2E3YnkxN2l0TkhnMWNxWjVkNyI
    ImV4cCI6IjE2Njg5Njk5OTcifQ.HJOtT7iBa9QCy7QgtcO3UzPN6e3Waj9Xyyjz6LwuDUc
```

# OAuth 2.0/OpenID Connect For the Enterprise

# OAuth 2.0
## Enterprise Versus Social Platform

# OAuth 2.0 - Enterprise Versus Social Platform

| OAuth 2.0 | Enterprise |
|---|---|
| Resource Owner | User |
| User Agent | Browser |
| Client | Application |
| Resource Server | Rest API, Service |



# OAuth 2.0 - Users, Groups, Scopes

## Social Platform

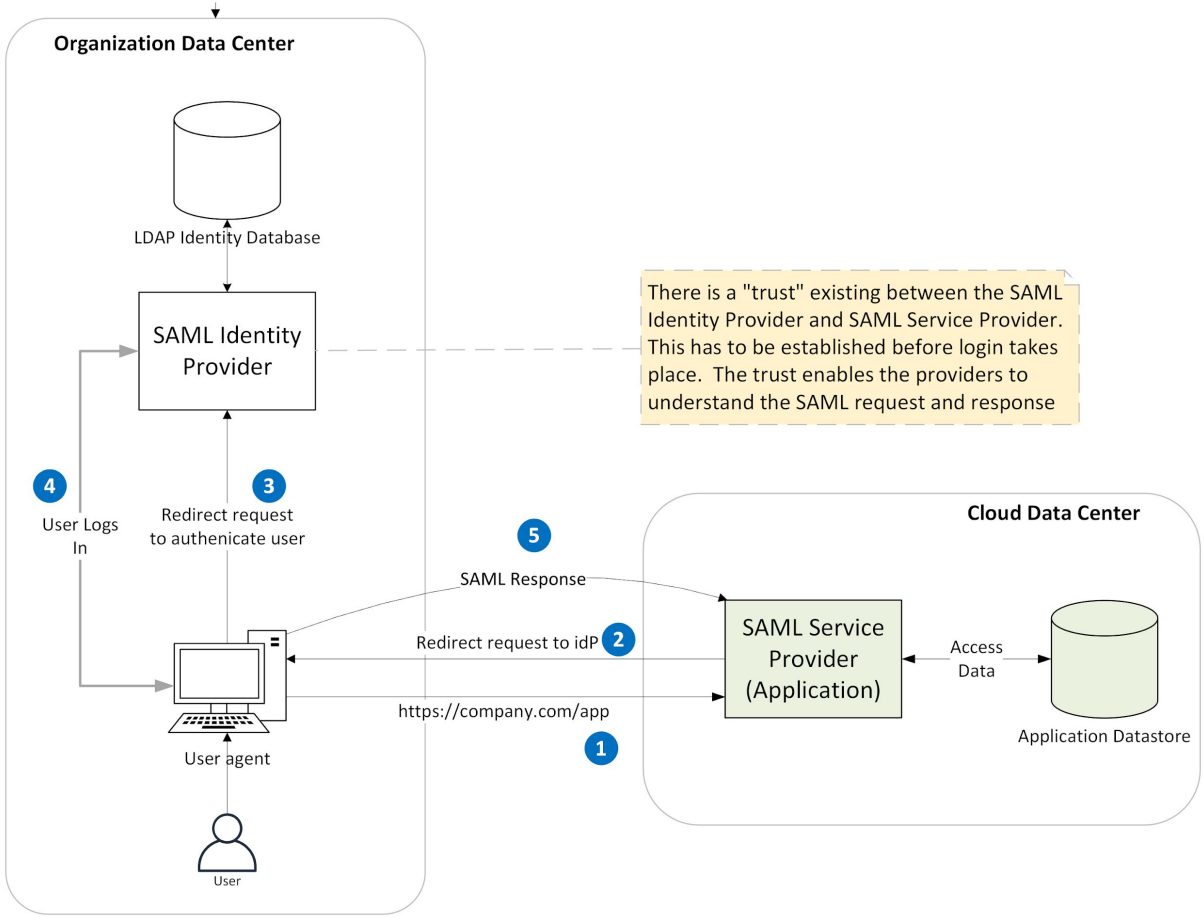
- 👉 User population endless
- 👉 User self registration
- 👉 No Groups but Scopes
- 👉 Scopes are associated at runtime

## Enterprise Platform

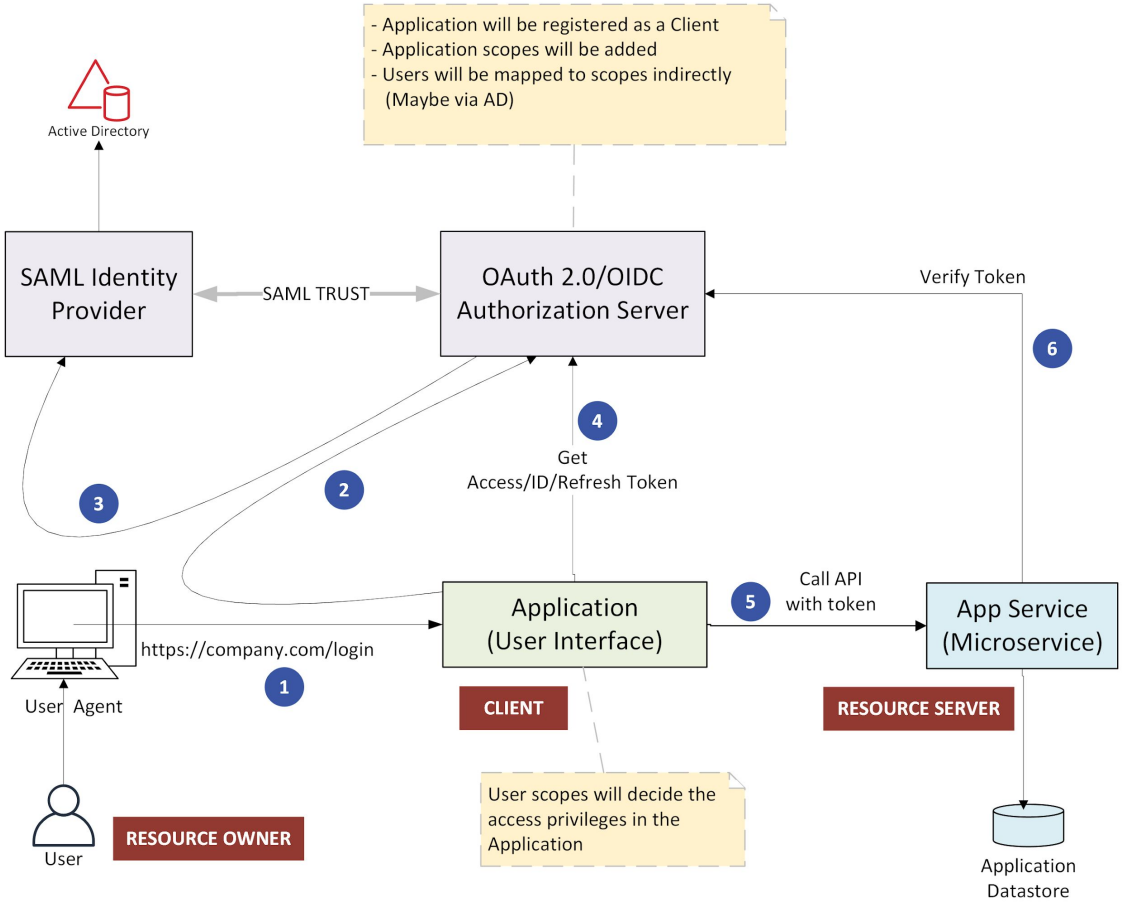
- 👉 User population is limited
- 👉 No self registration
- 👉 Users are assigned to groups, roles
- 👉 User associated to groups upfront
- 👉 Scopes can be used Optionally

# OAuth 2.0
## Typical Enterprise Setup On Premises

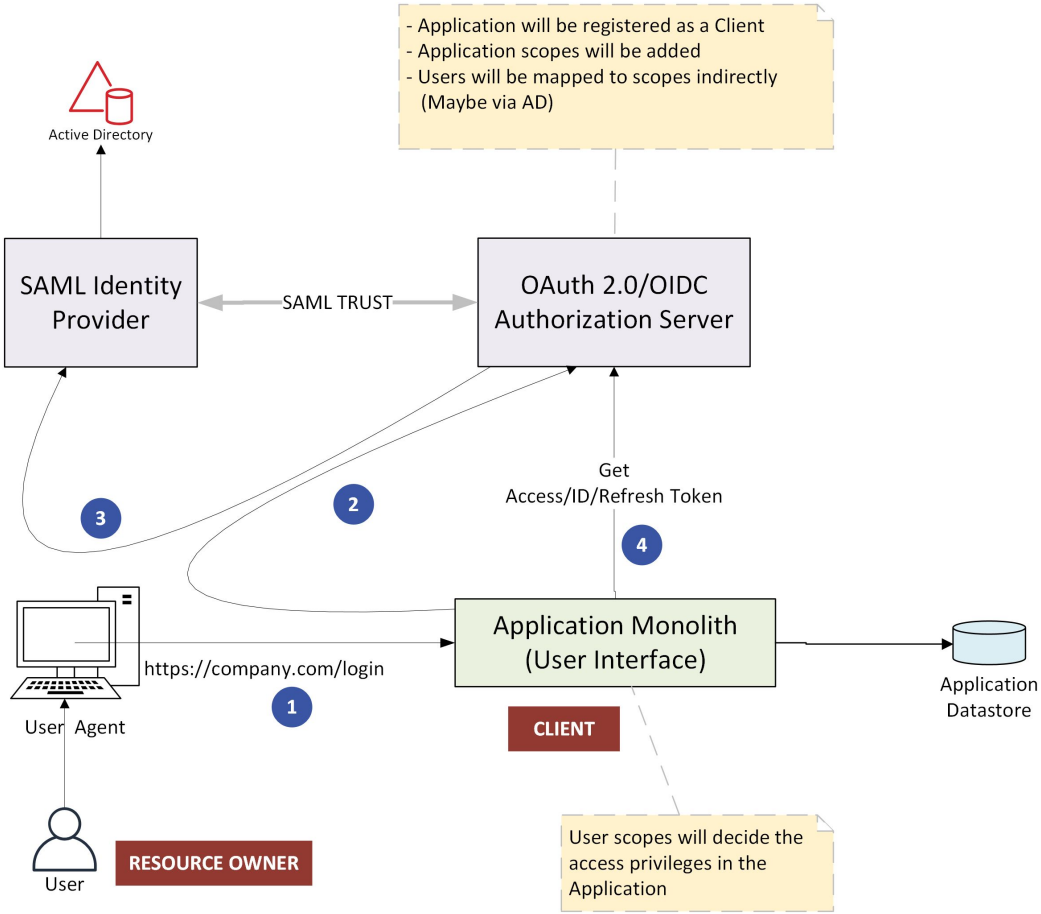
# Revisiting SAML Architecture



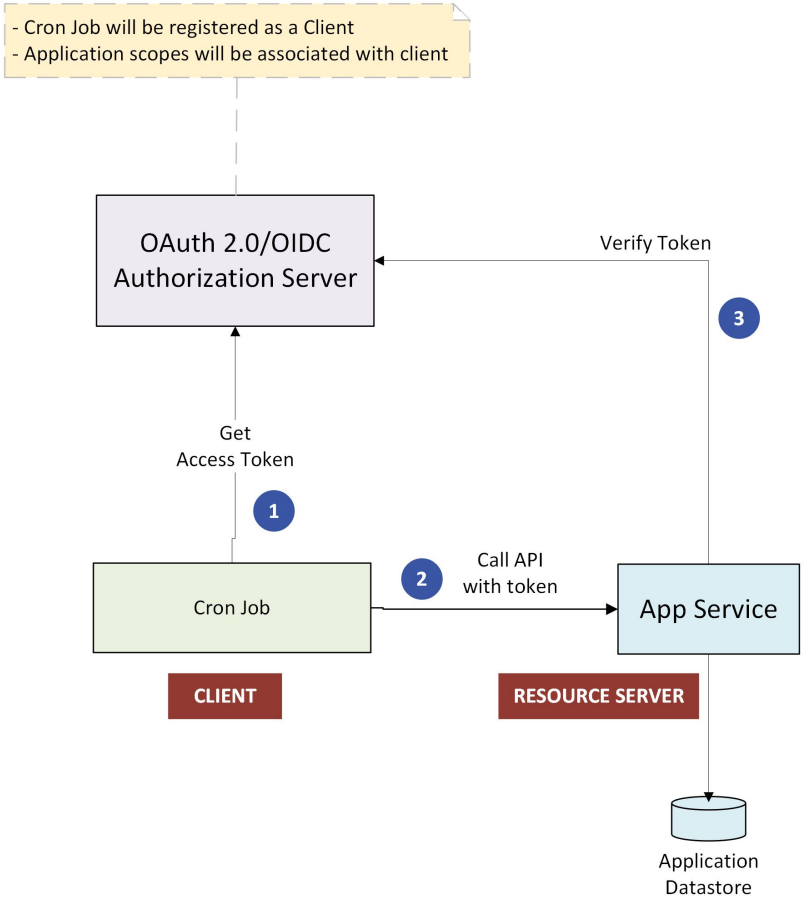
# OAuth 2.0 - Typical Enterprise Microservices Application



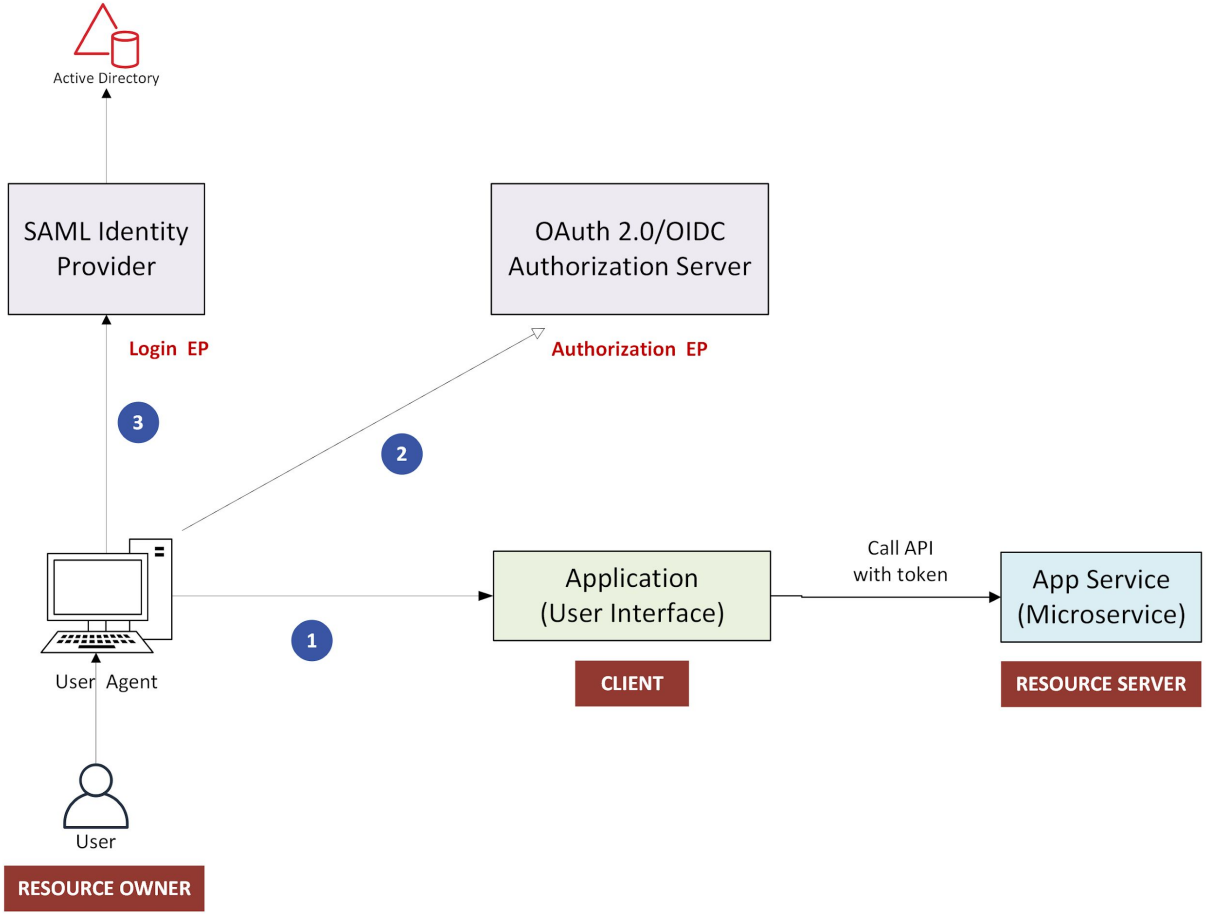
# OAuth 2.0 - Typical Enterprise Monolith Application



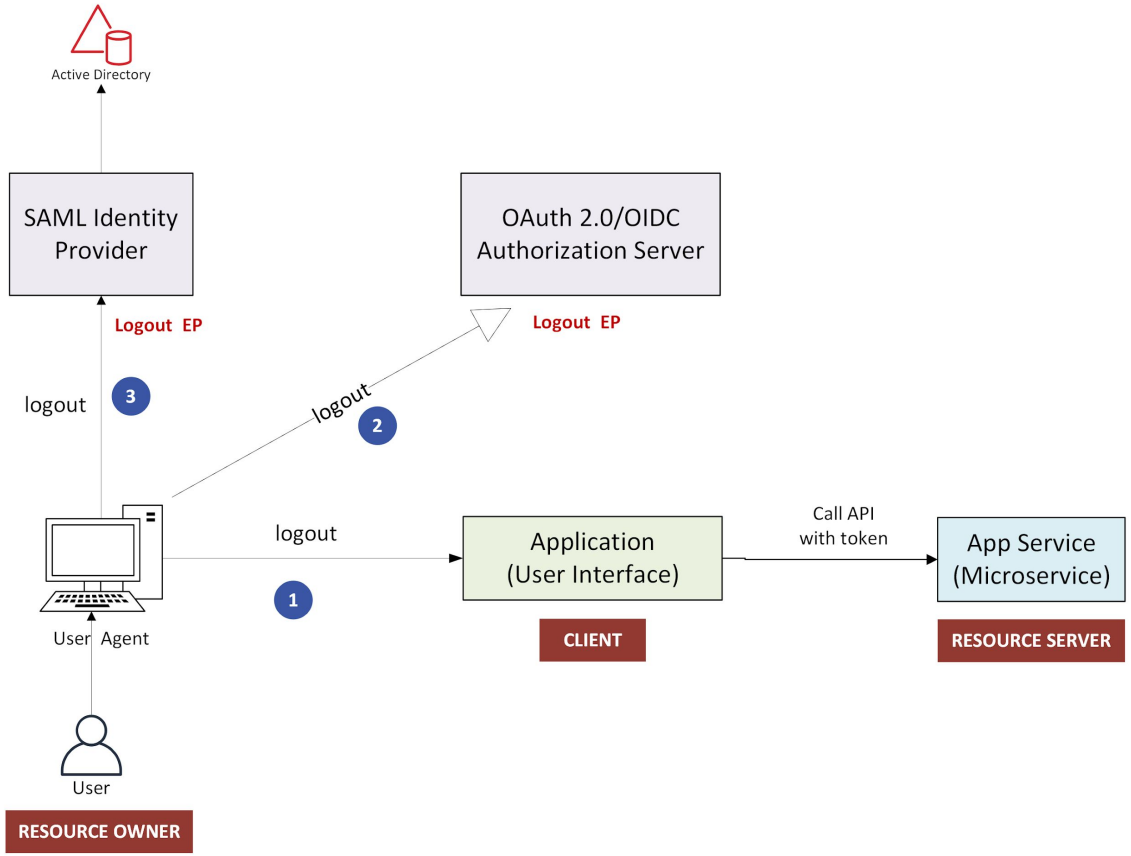
# OAuth 2.0 - Typical Enterprise Cron Job



# OAuth 2.0 - Multiple Sessions, Login and Logout

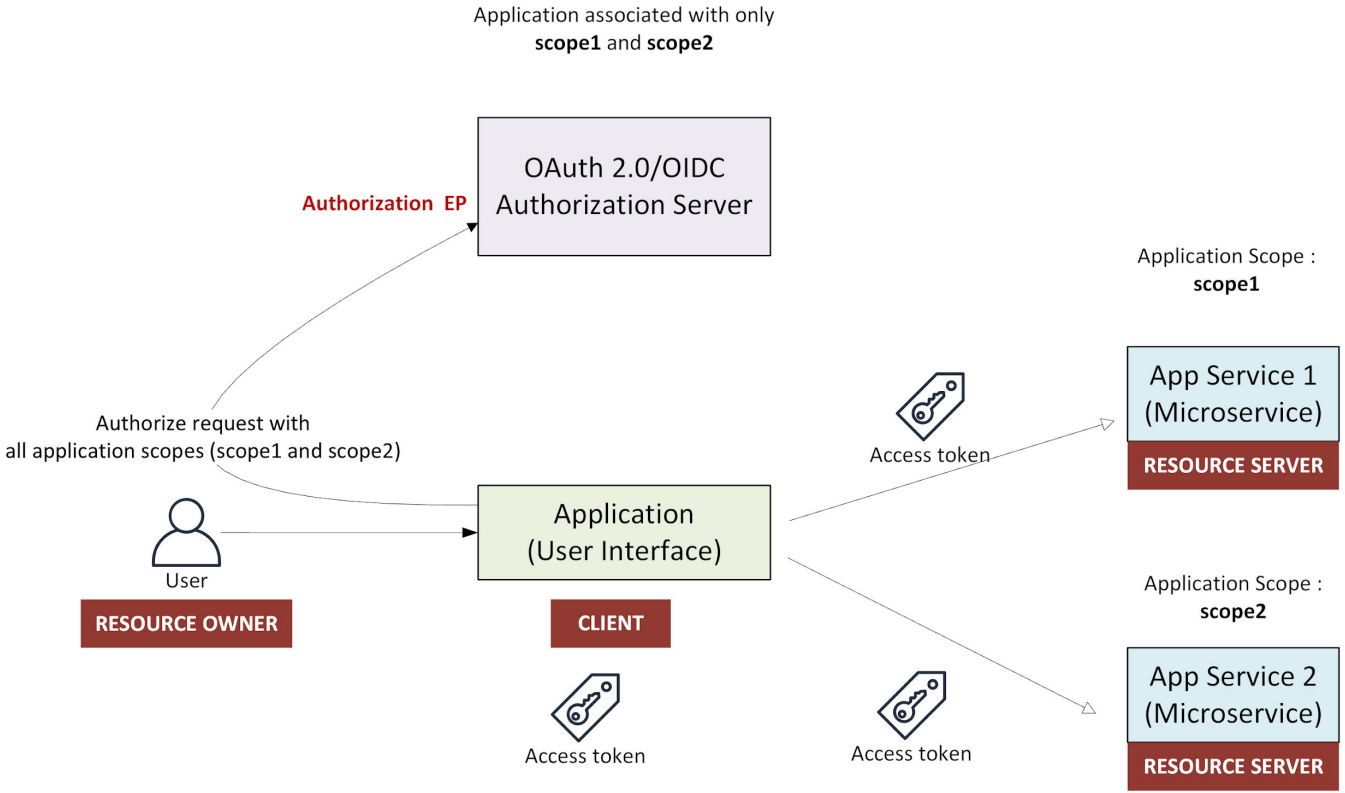


# OAuth 2.0 - Multiple Sessions, Login and Logout

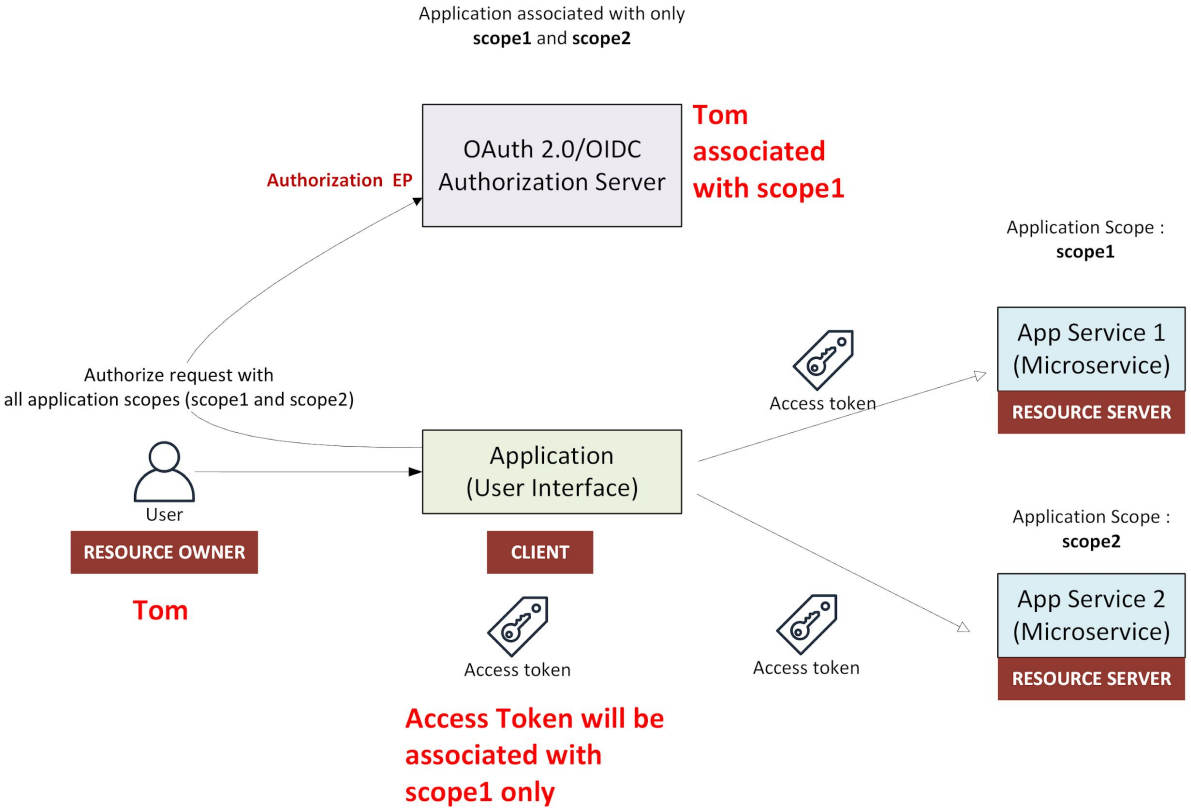


# OAuth 2.0 - User association to scopes

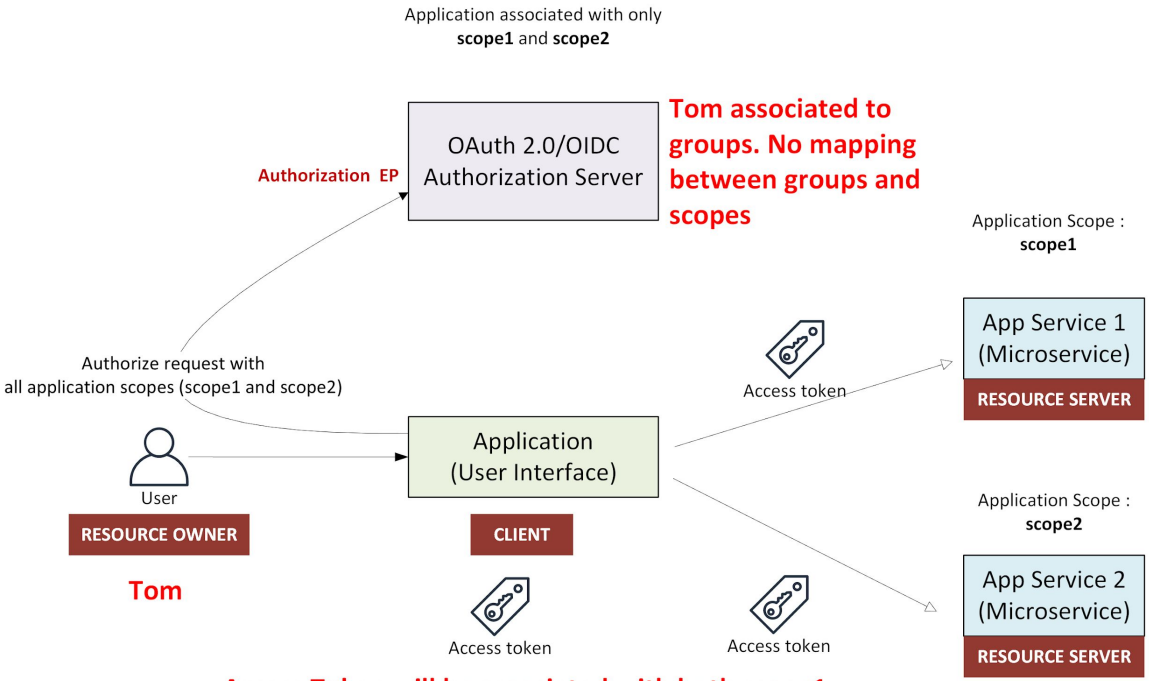
# OAuth 2.0 - User association to scopes



# OAuth 2.0 - Downscoping

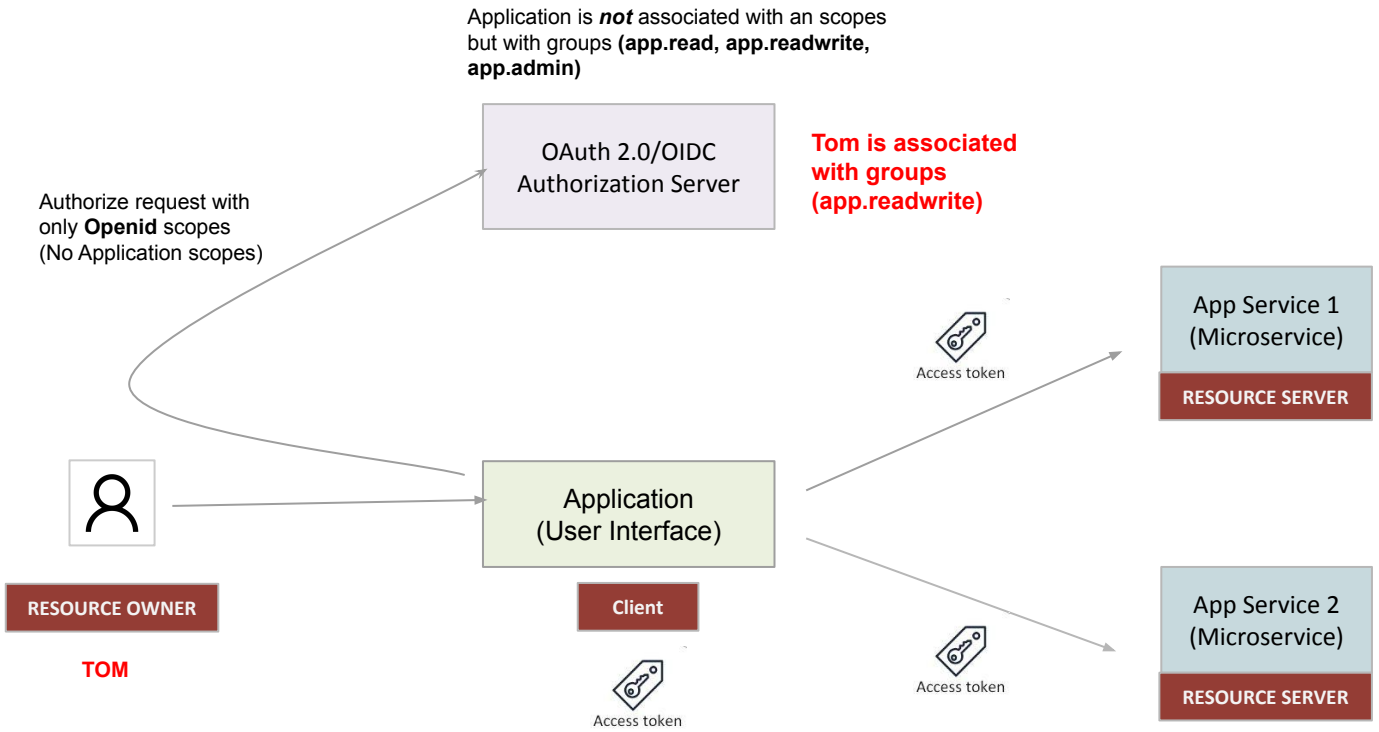


# OAuth 2.0 - Non Downscoping



# OAuth 2.0 - User association to Groups

# OAuth 2.0 - User association to Roles



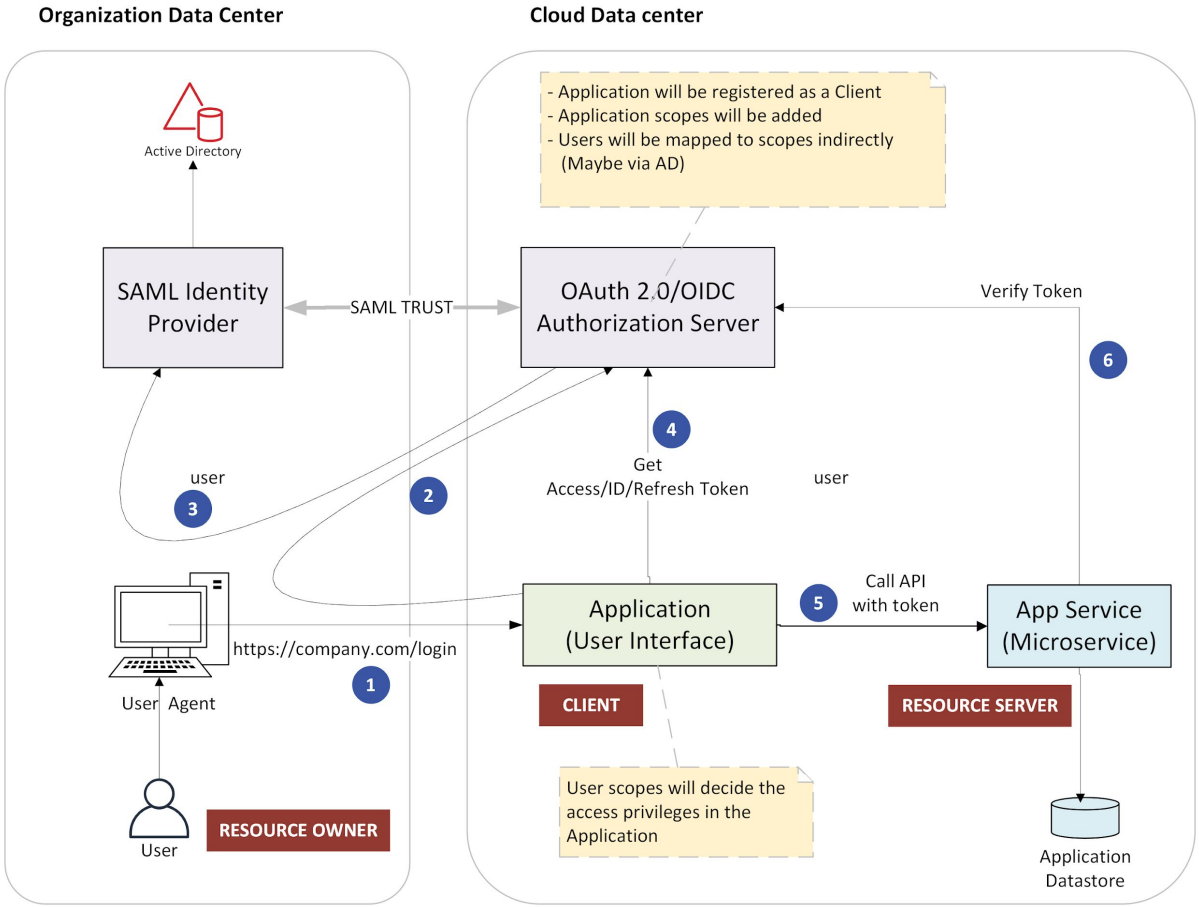
# OAuth 2.0 - User association to Groups

# OAuth 2.0 - User Association to Groups

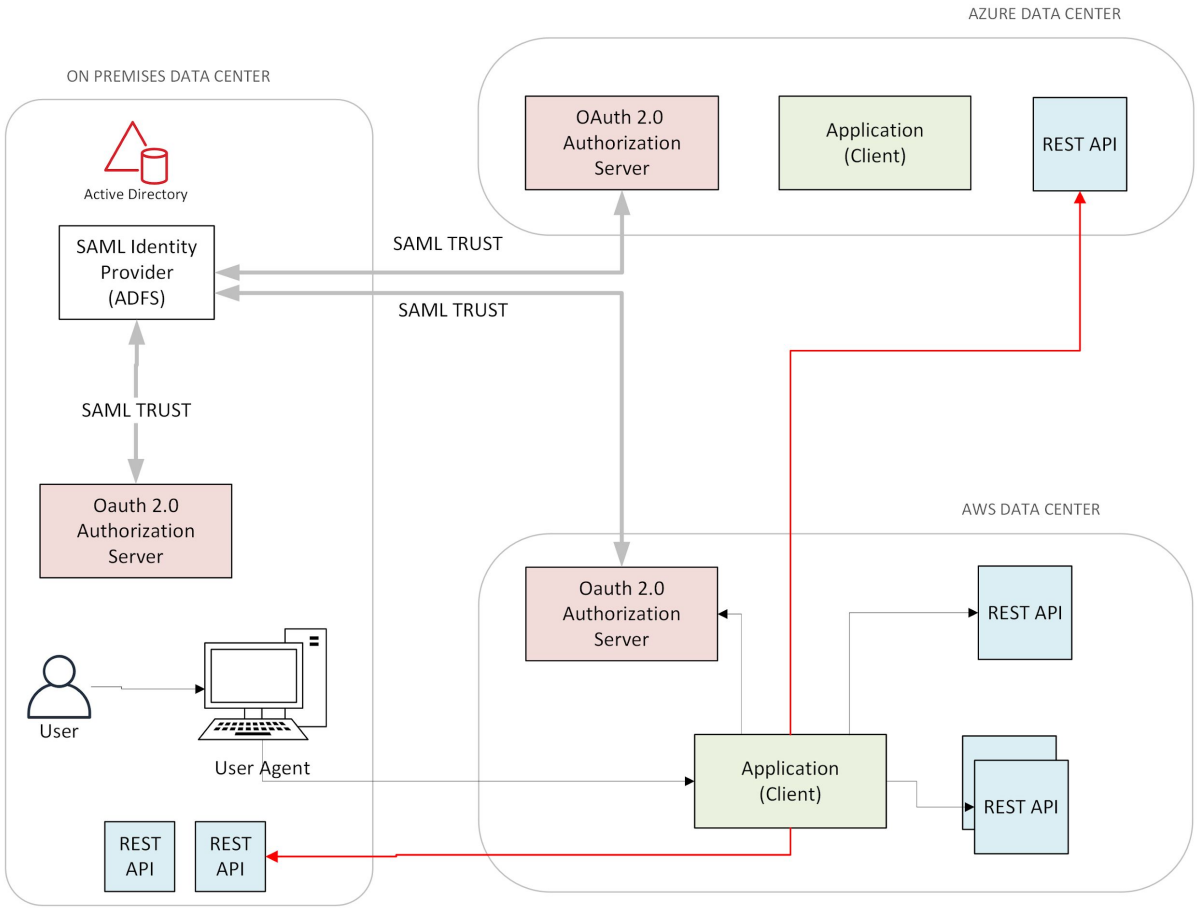
- 👉 Create the Application groups
- 👉 Assign groups to Application Users
- 👉 Add “groups” claim to Access Token
- 👉 Modify Application to look at Groups instead of Scopes

# OAuth 2.0 in the Cloud

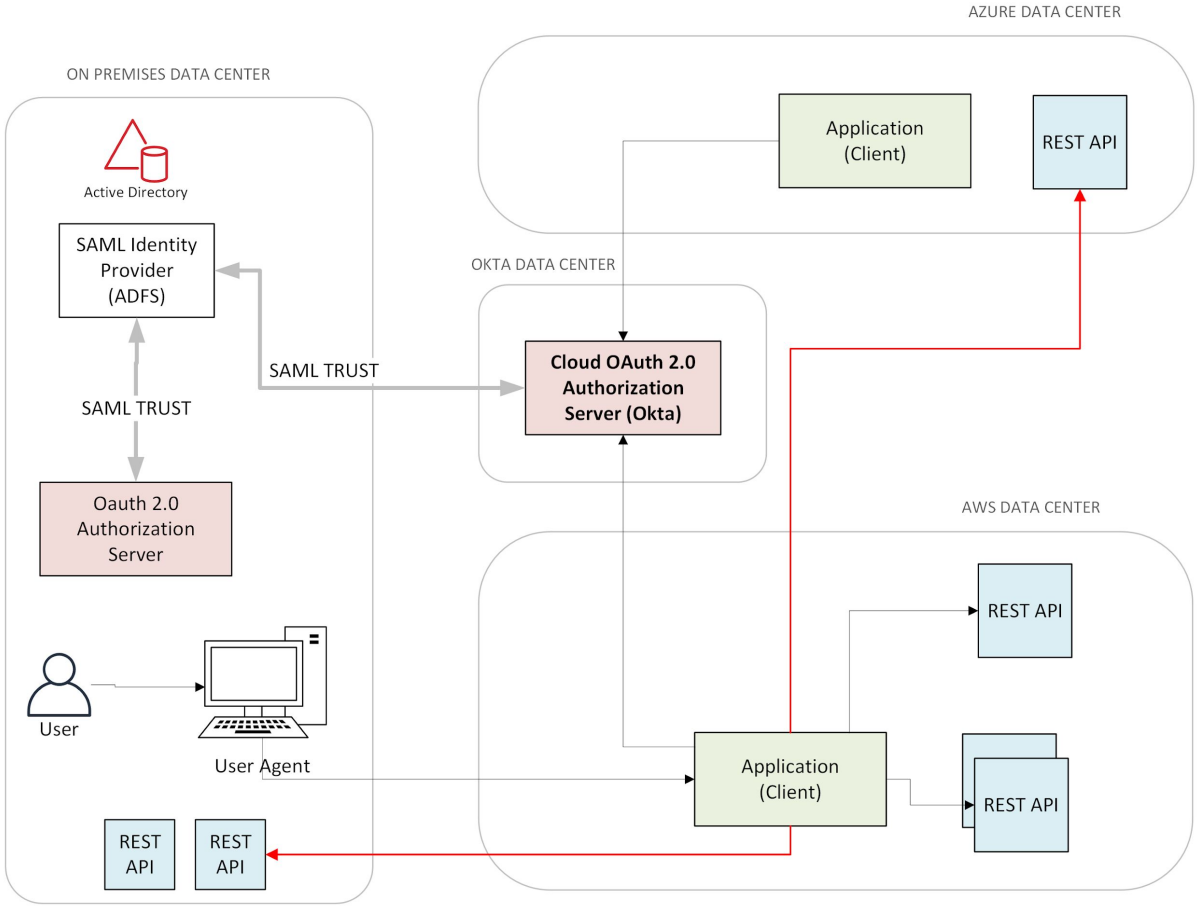
# OAuth 2.0 - Cloud Deployment



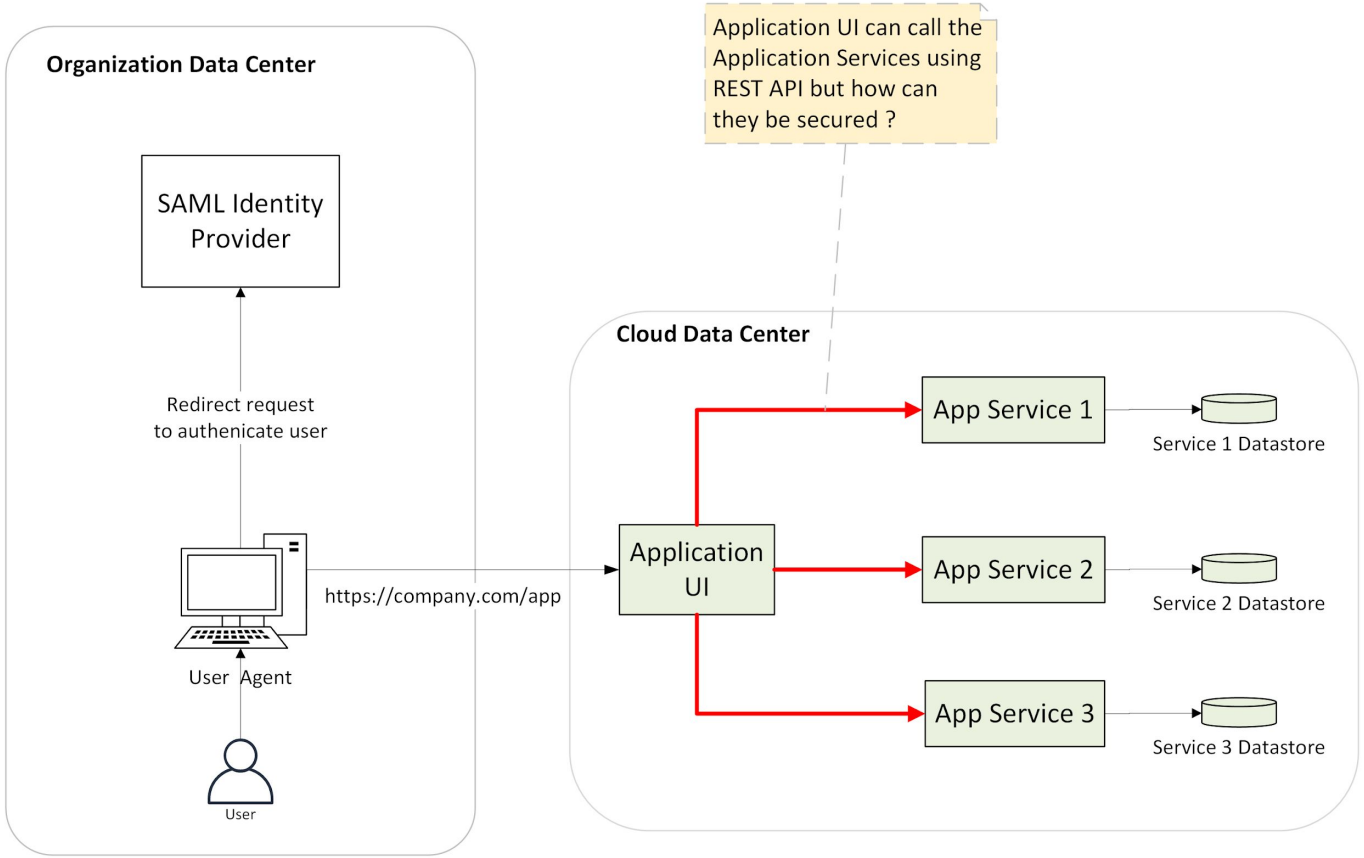
# OAuth 2.0 - Multi Cloud Deployment



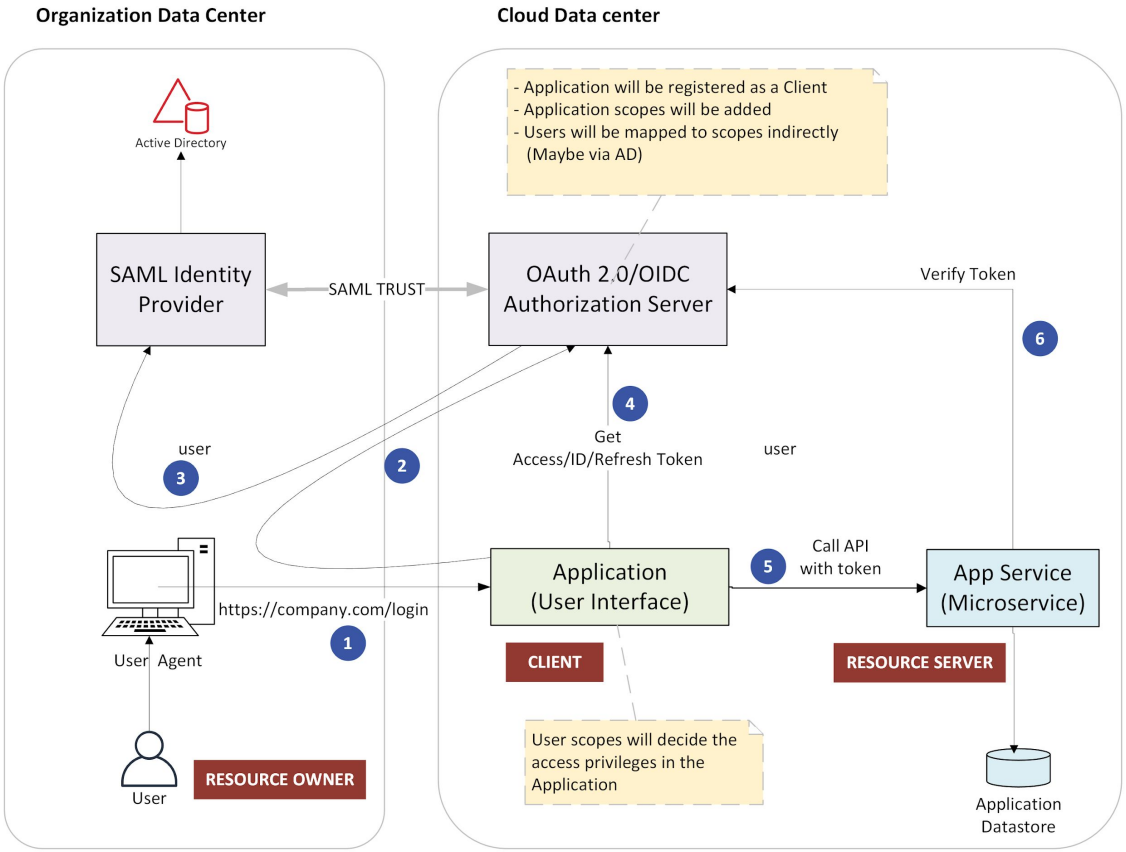
# OAuth 2.0 - Multi Cloud Deployment (OAuth As a Service)



# Enterprise problem 1 - Microservices

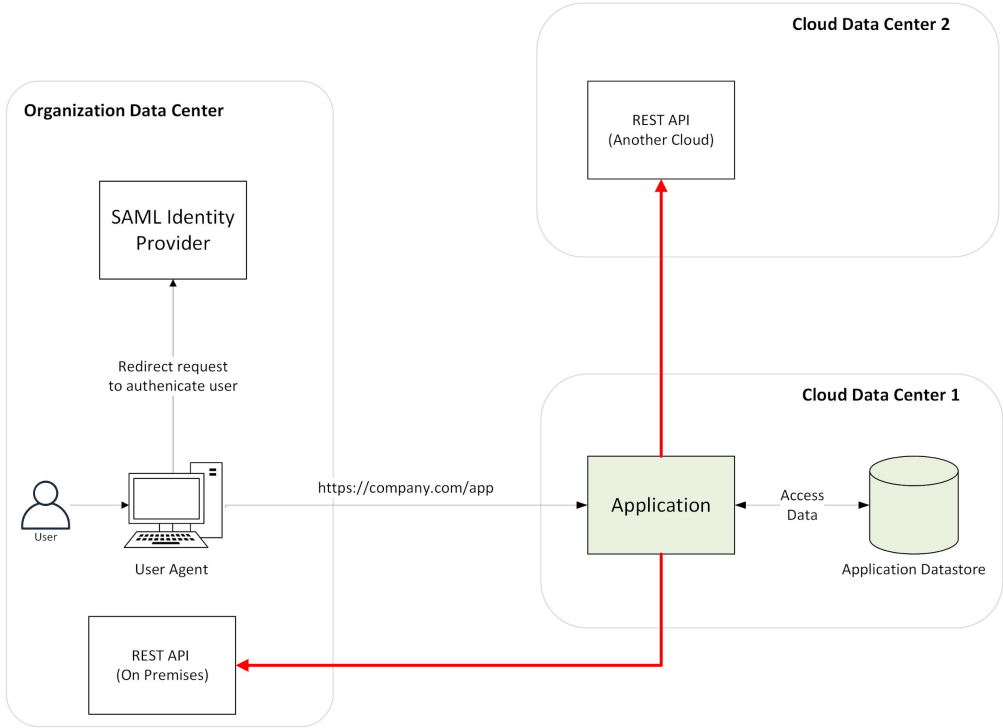


# Solution - Deploy OAuth 2.0/OIDC Authorization Server

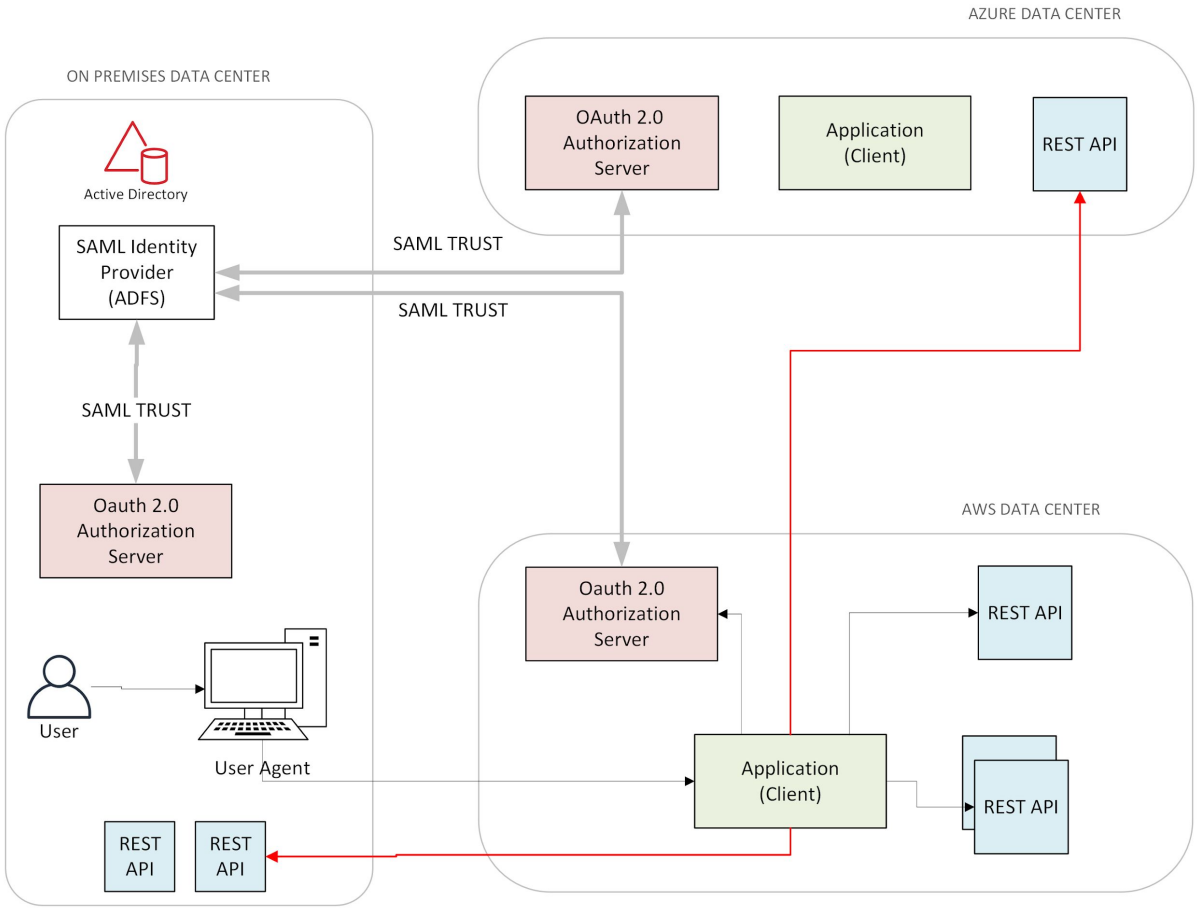


# Enterprise problem 2 - Cloud apps

- How does REST calls across network boundaries get secured ?

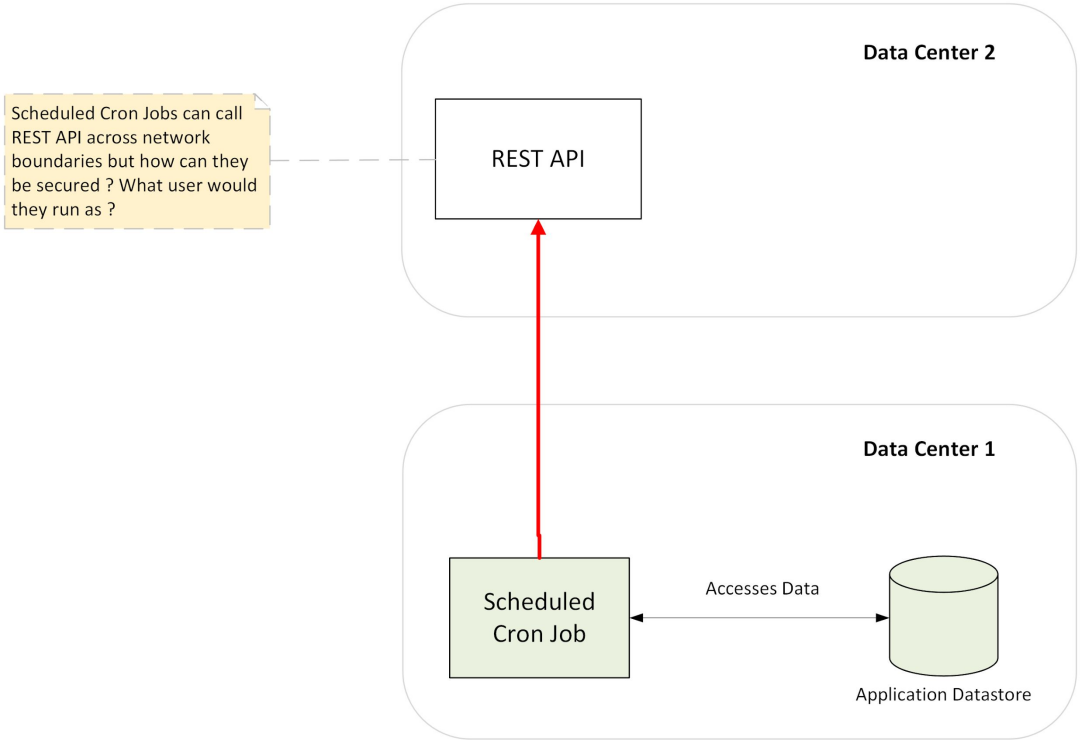


# Solution - Deploy OAuth 2.0/OIDC Authorization Server

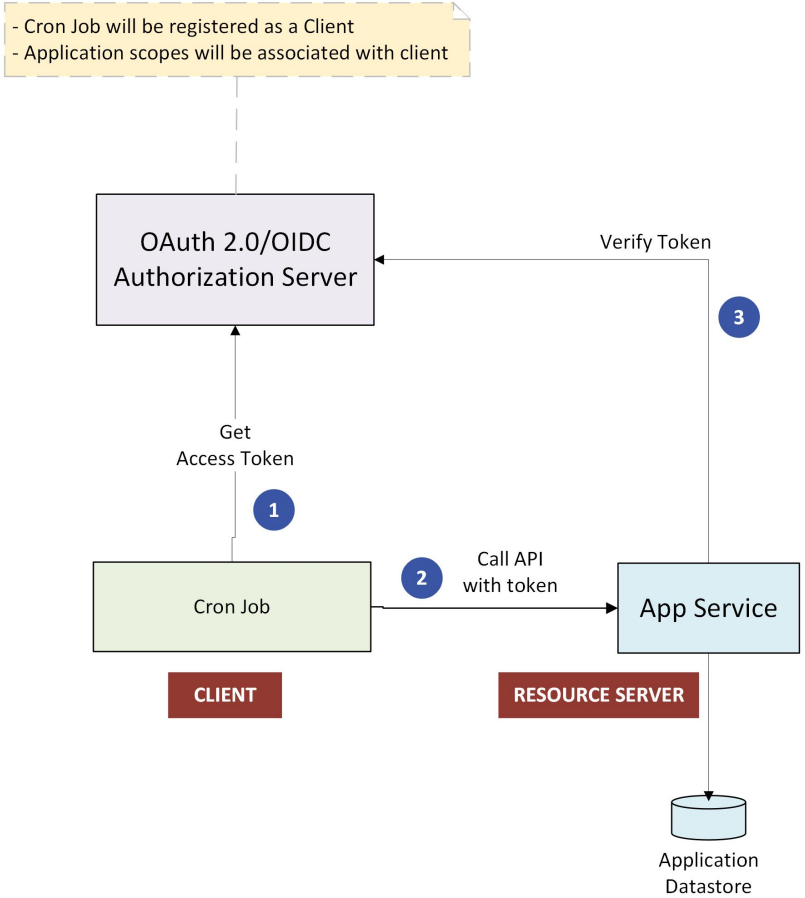


# Enterprise problem 3 - Machine to Machine

- Scheduled Tasks, Daemons sometimes need to call REST APIs. How are they secured ?
- No user involved



# Solution - Deploy OAuth 2.0/OIDC Authorization Server

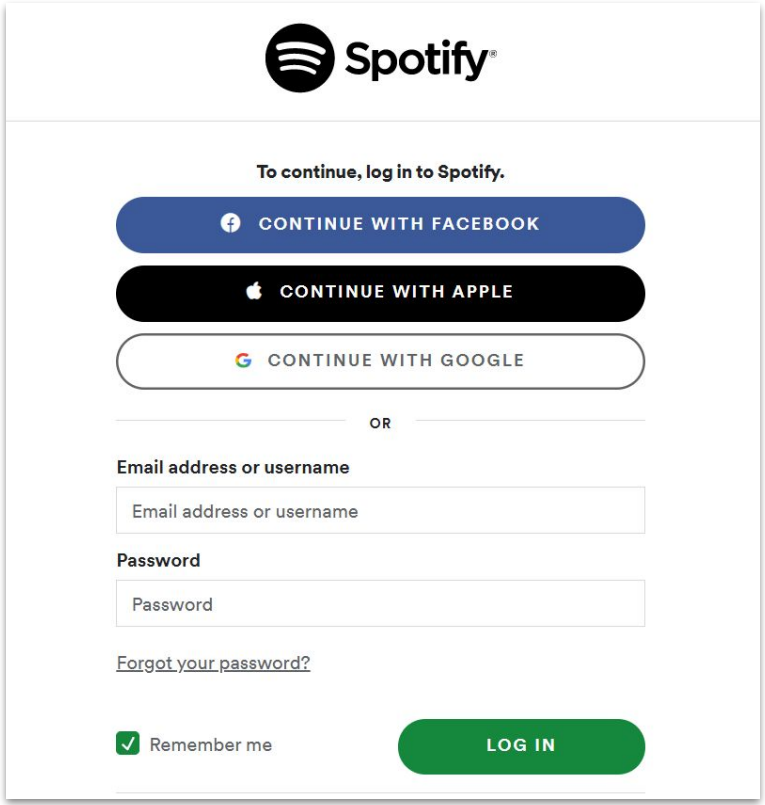


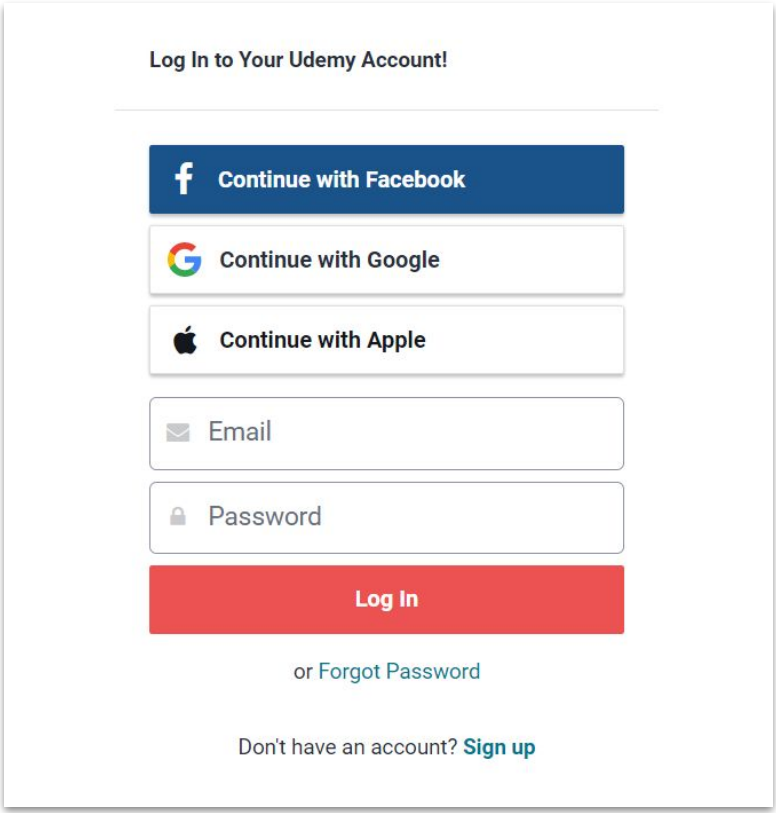
# OAuth 2.0 and OpenID Connect

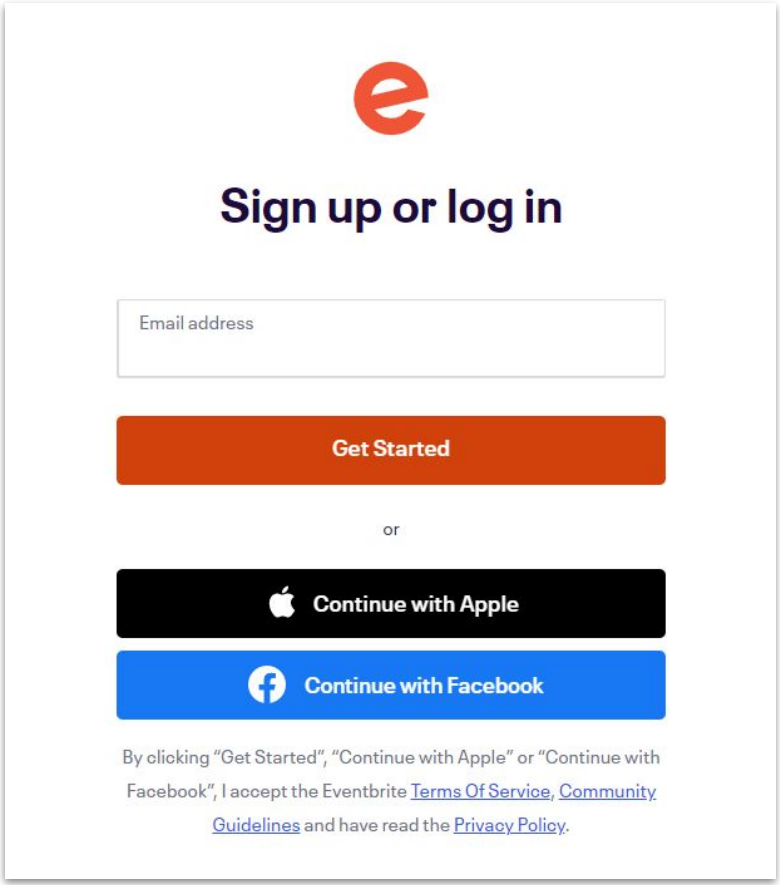
- OAuth 2.0 is for authorization
  - Access token should contain only authorization information
  - Scopes, Roles, Grant Types, Flows, Tokens
  - No ID Token
  - https://tools.ietf.org/html/rfc6749

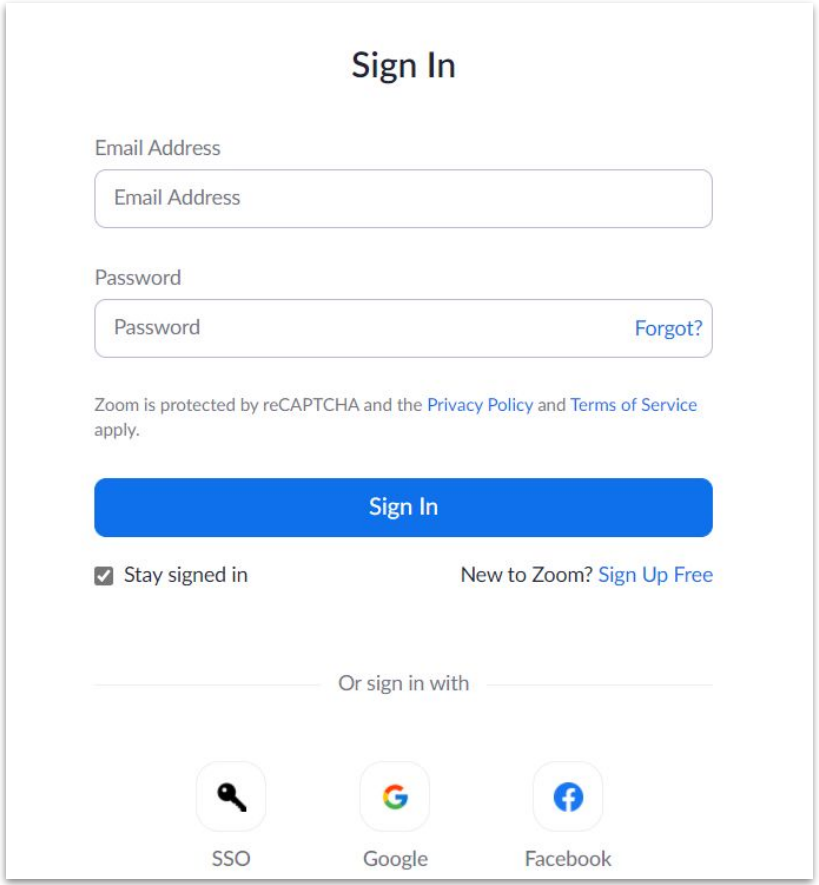
- OpenID Connect is the Identity layer on top of OAuth 2.0
  - ID Token contains user claims
  - /userinfo endpoint
  - Scopes : openid profile email
  - Can create custom claims
  - Adds more Response types
  - https://openid.net/specs/openid-connect-core-1_0.html

```
# Spring security OAuth 2.0 Client setup to access Google Albums with Authorization code flow
spring.security.oauth2.client.registration.google.client-name=myalbum
spring.security.oauth2.client.registration.google.client-id=70221534387-d8gsqqr16fhvbjlln55fcqsmrpke6v3n.apps.googleusercontent.com
spring.security.oauth2.client.registration.google.client-secret=BELlQoVg_2M_hAMp1C-Svp9D
spring.security.oauth2.client.registration.google.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.google.scope=openid,profile,email,https://www.googleapis.com/auth/photoslibrary.readonly

# Application specific constants
photolibrary.authorizer=Google
photolibrary.albums.uri=https://photoslibrary.googleapis.com/v1/albums
photolibrary.photos.uri=https://photoslibrary.googleapis.com/v1/mediaItems:search
photolibrary.logout.url=https://www.google.com/accounts/Logout
```

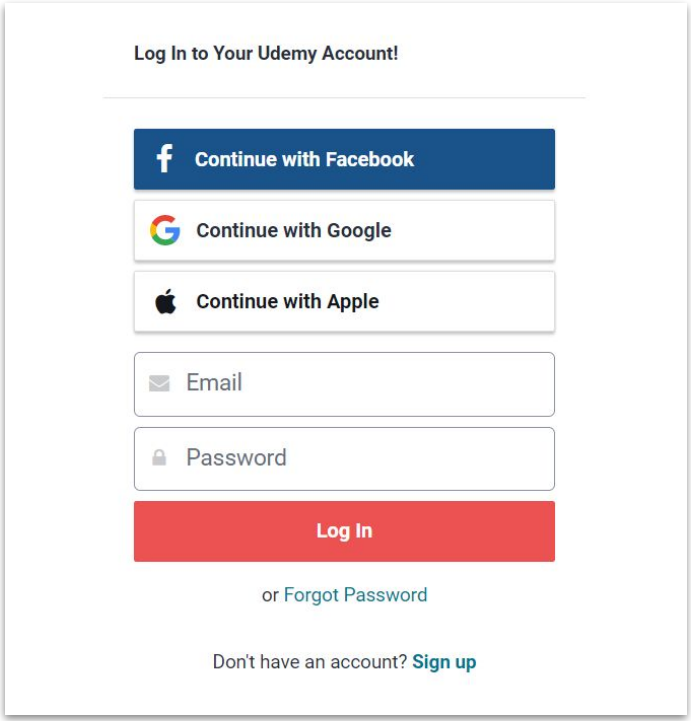


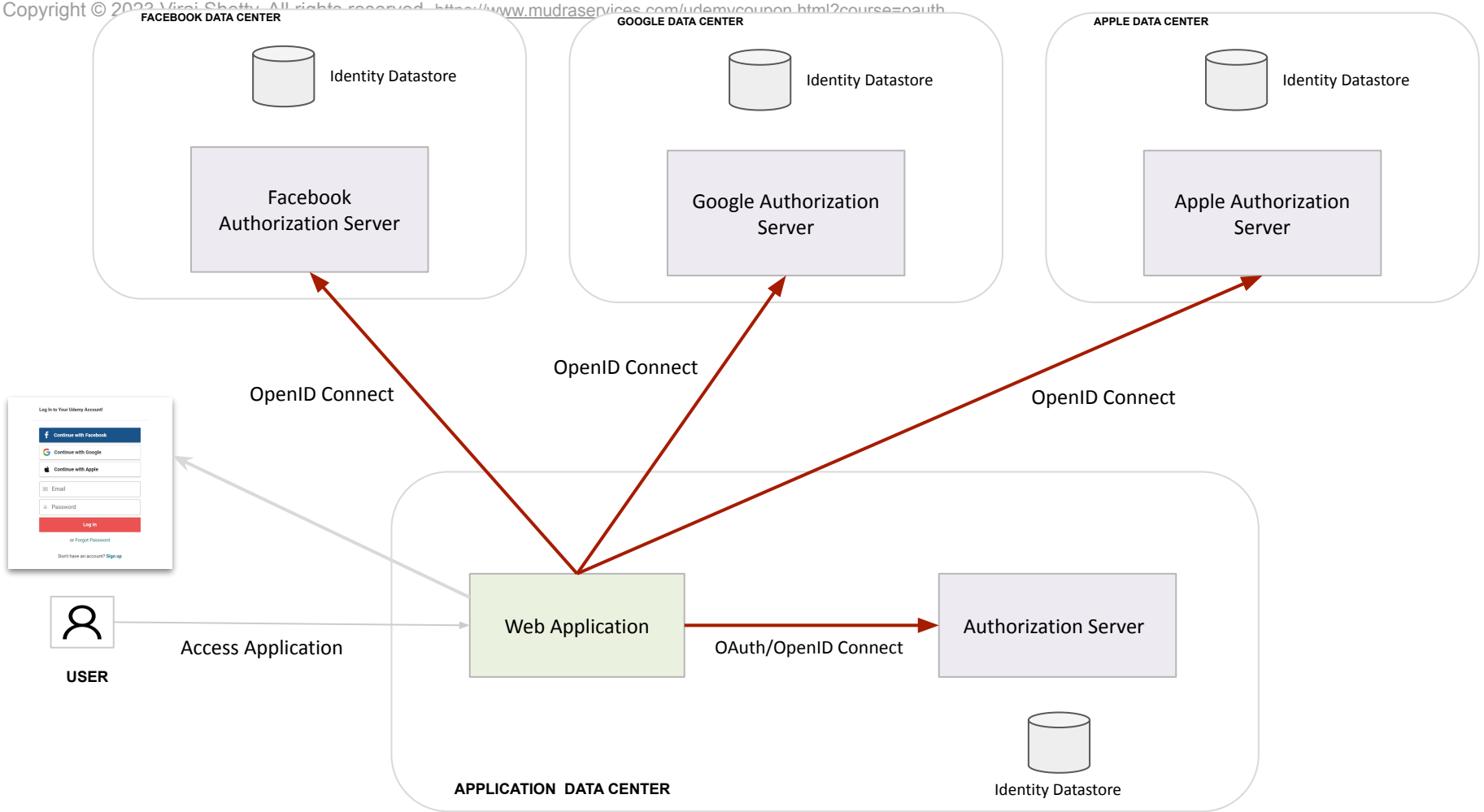



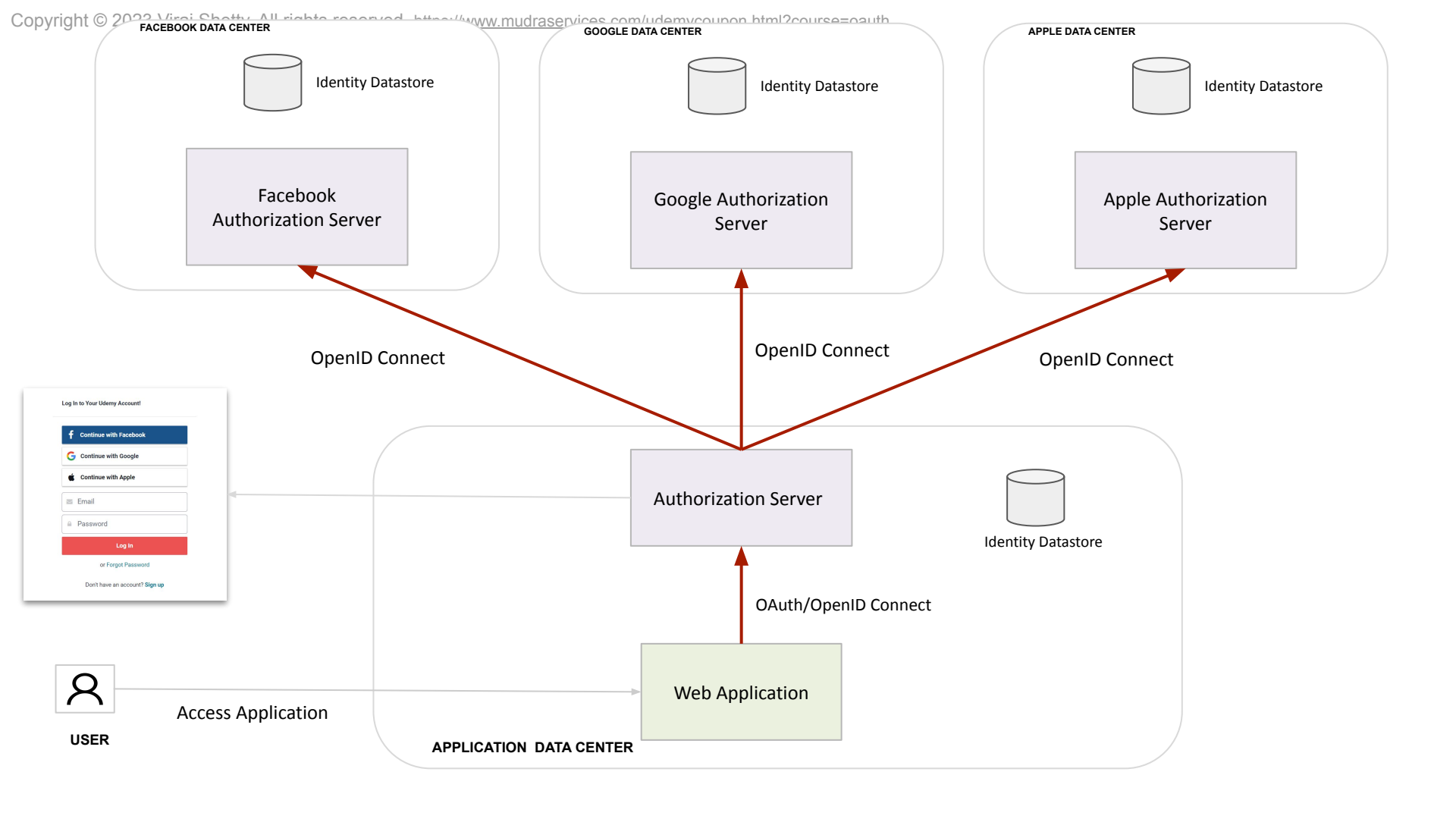


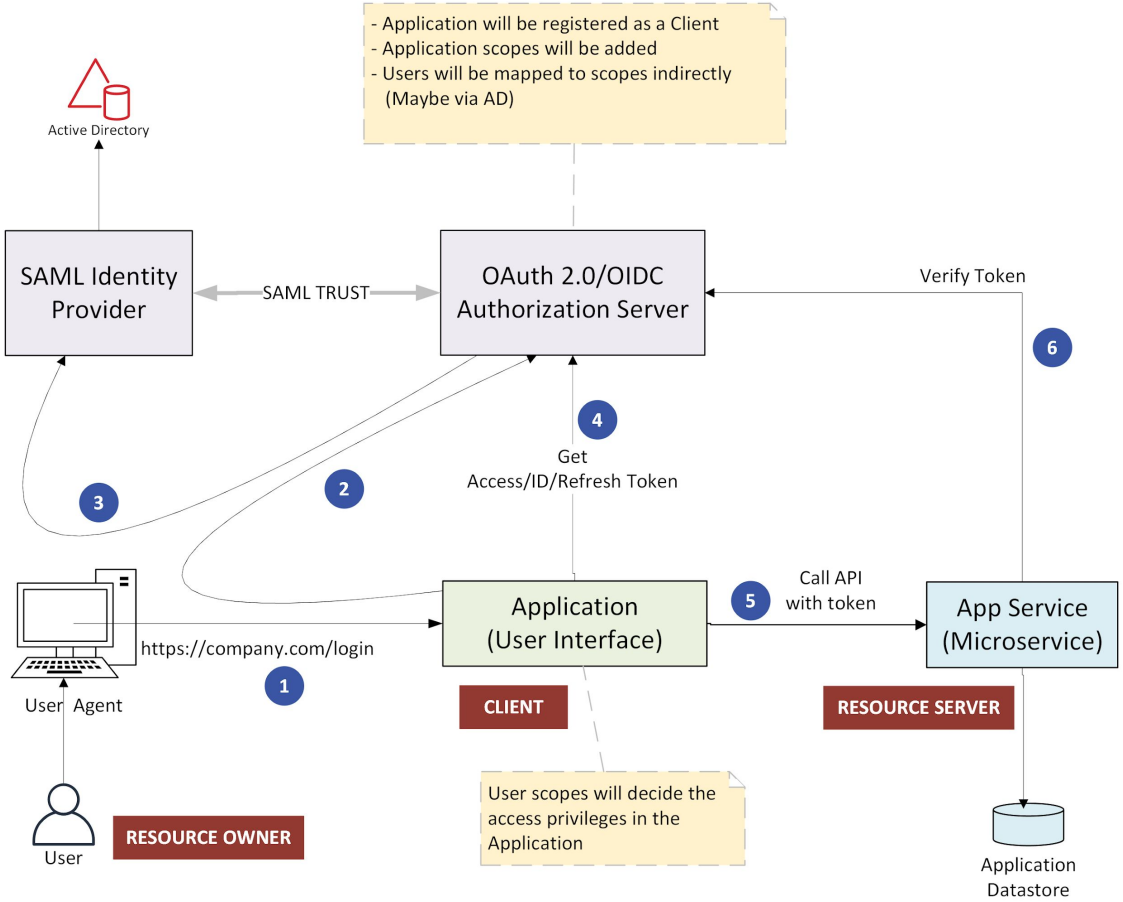
# Identity Brokers

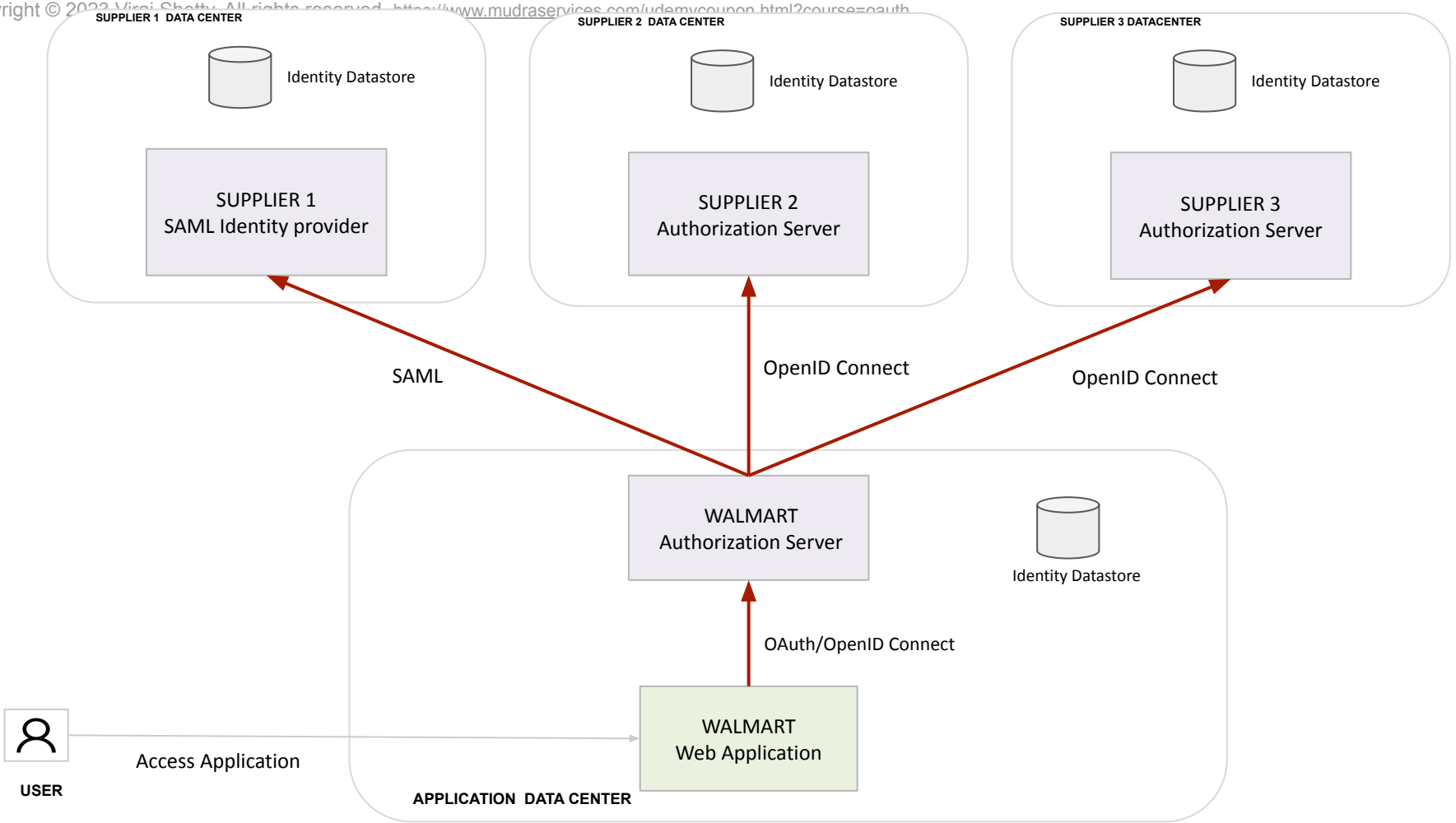






# OAuth 2.0 - Typical Enterprise Microservices Application

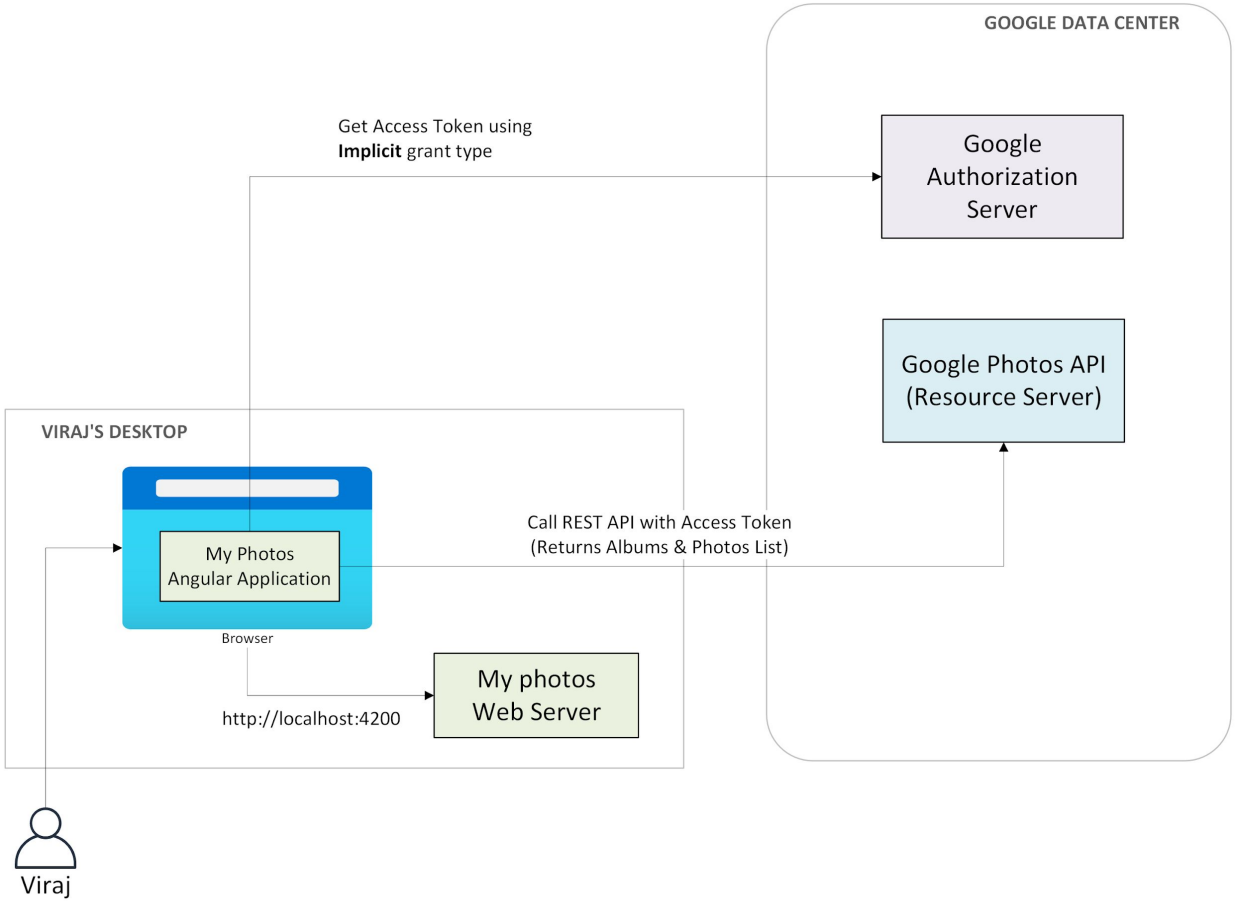


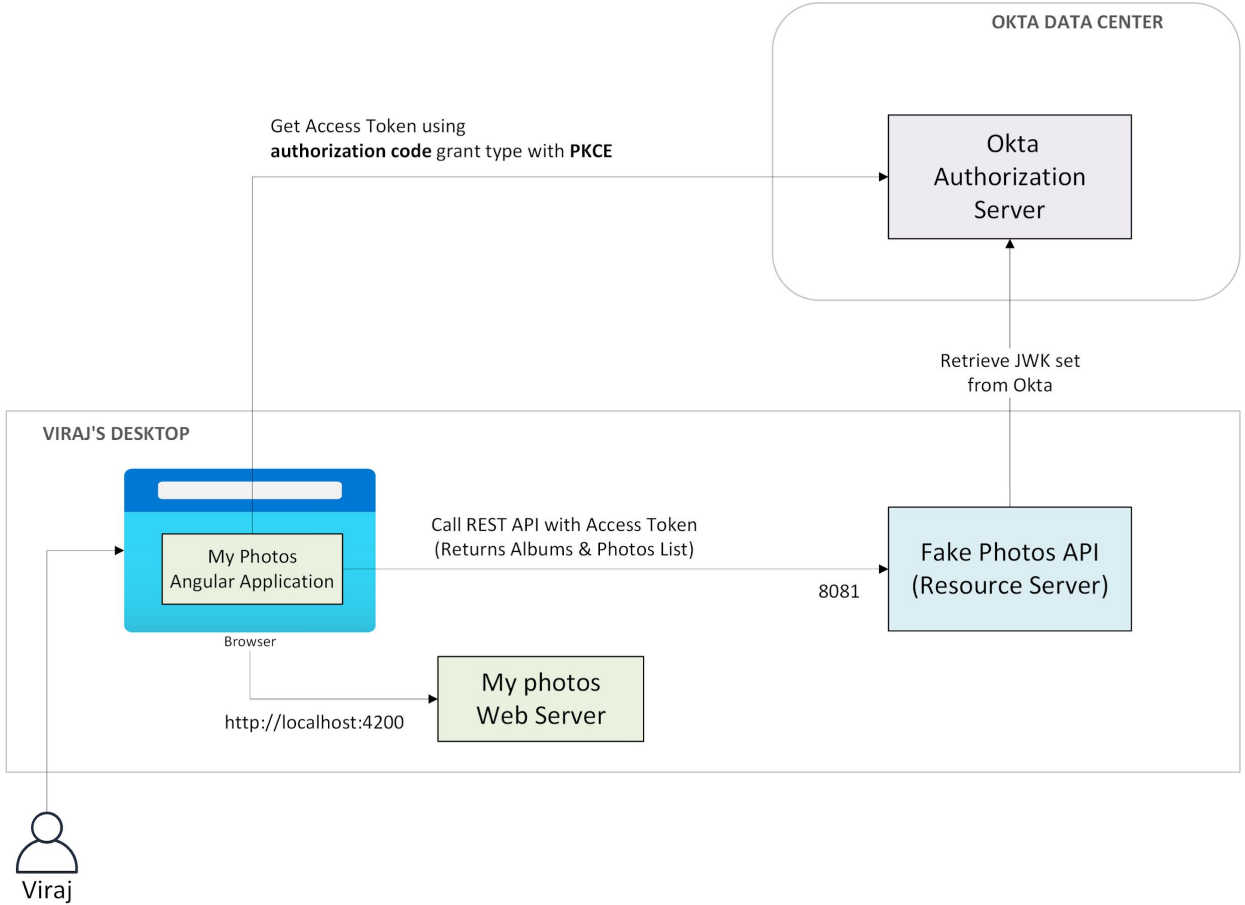


# OAuth 2.0 Best Practices

- Prefer Authorization Code Grant with PKCE
- Prefer Client Credentials Grant for Cron Jobs
- Avoid using Implicit Grant
- Avoid using Resource Owner Password Grant
- Store the secrets in a Safe place
- Rotate the secrets regularly
- Keep Access tokens short (5 min)
- Avoid using local users of the Authorization Server
- Do not associate users with more scopes than needed
- Use the enterprise logout (all sessions)
- Do not store tokens or secrets in the browser or Mobile devices

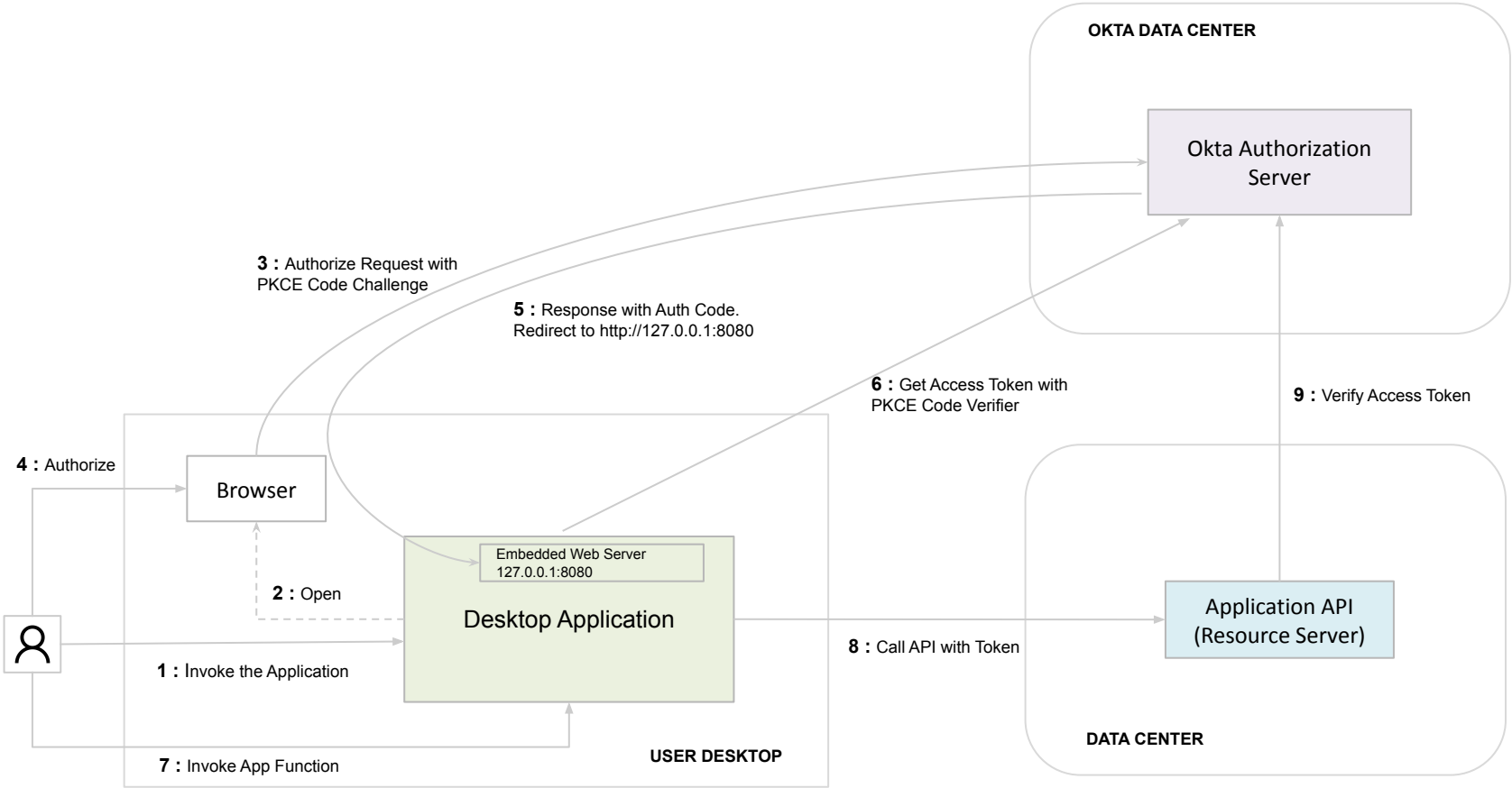
# Coding Project
## Single Page Application Using Angular





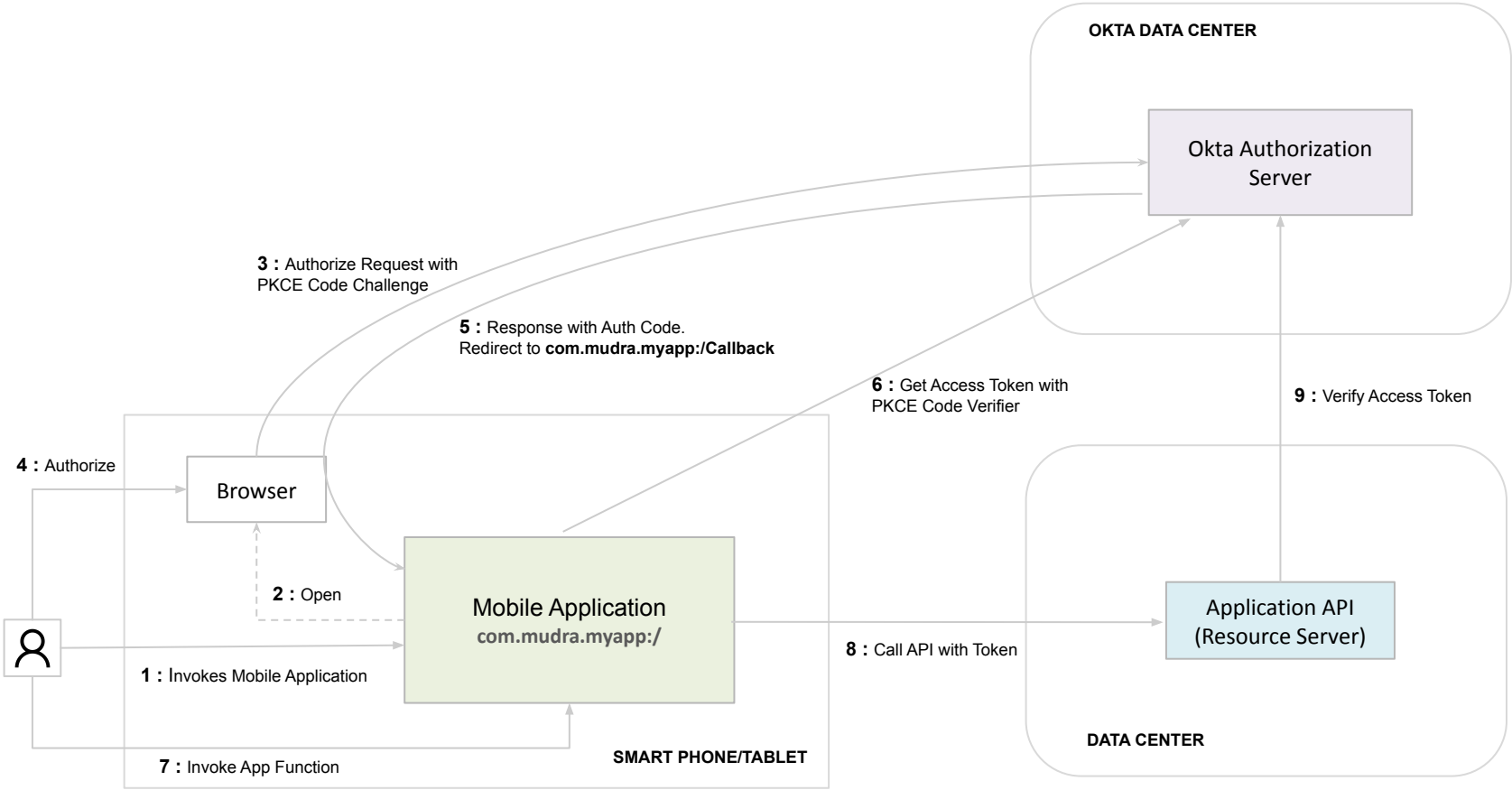
# Demo

# Desktop Applications

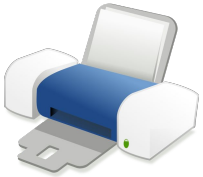


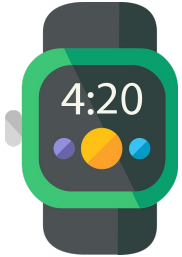
| | |
|---|---|
| Google Drive | googledrive:/ |
| Google Maps | googlemaps:/ |
| YouTube | youtube:/ |
| Netflix | nflx:/ |
| Apple iMovie | imovie:/ |
| Apple Maps | map:/ |

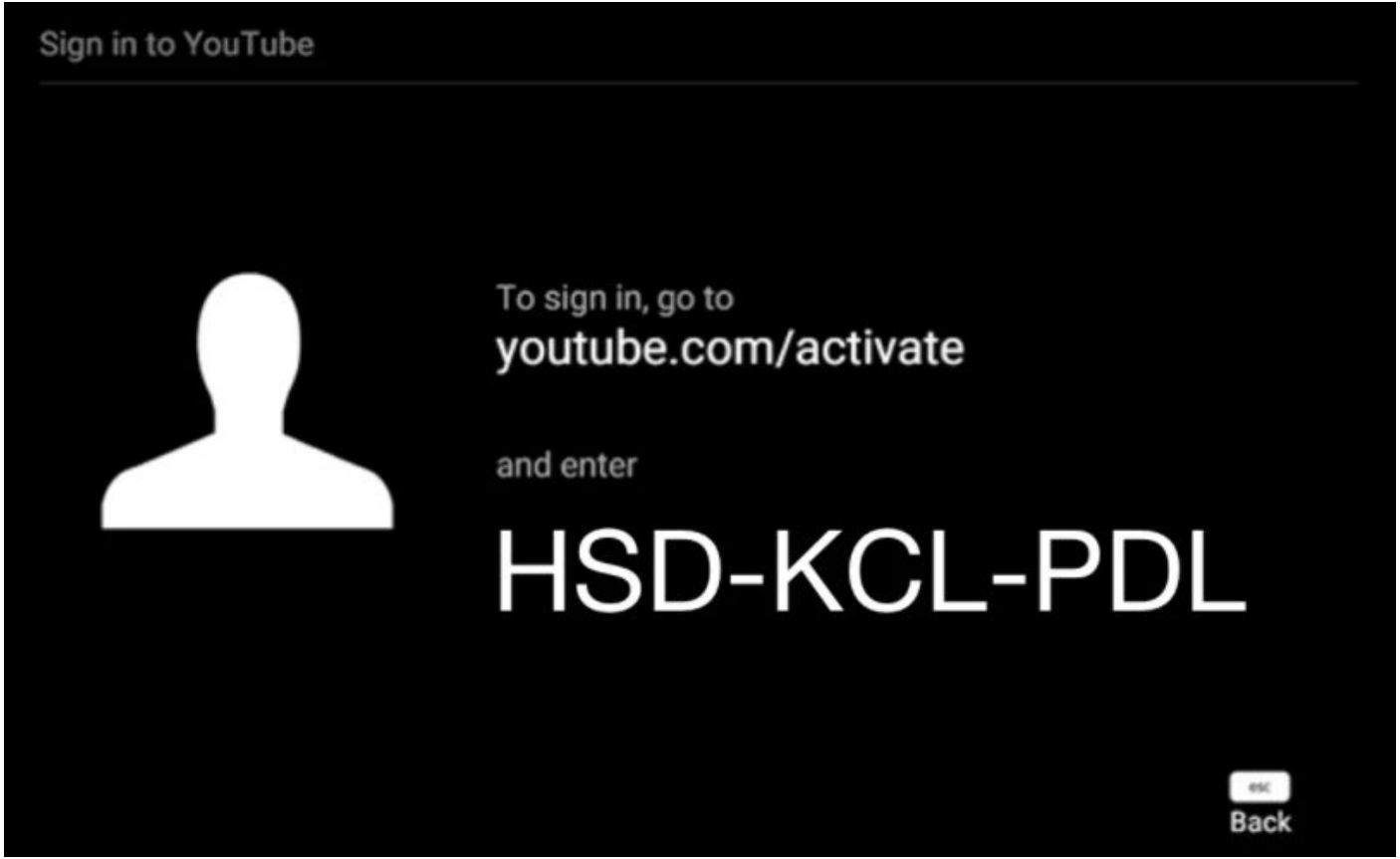
```
com.mudra.myphotosapp:/

com.mudra.myphotosapp:/Callback

com.mudra.myphotosapp:/Callback?state=xyz&code=SplxlOBeZQQYbYS6WxSbIA
```

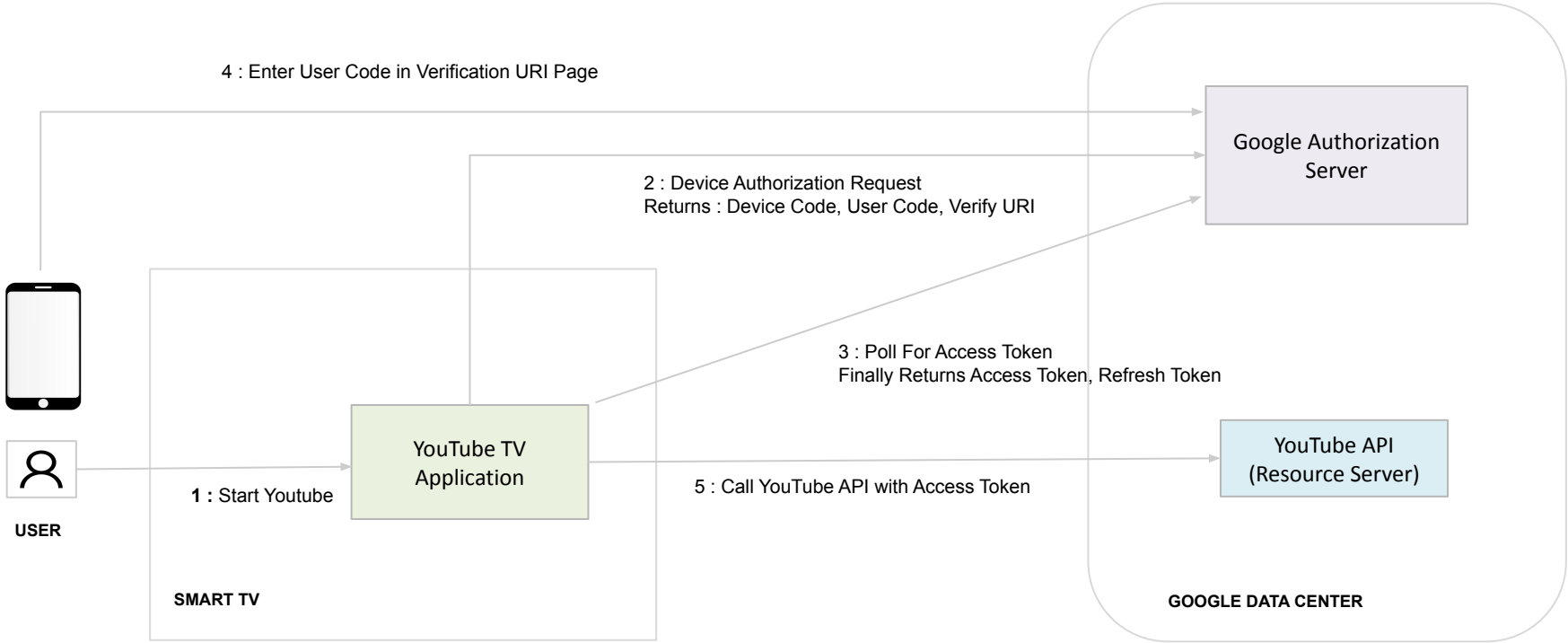


# Input Constrained Devices







# Conclusion