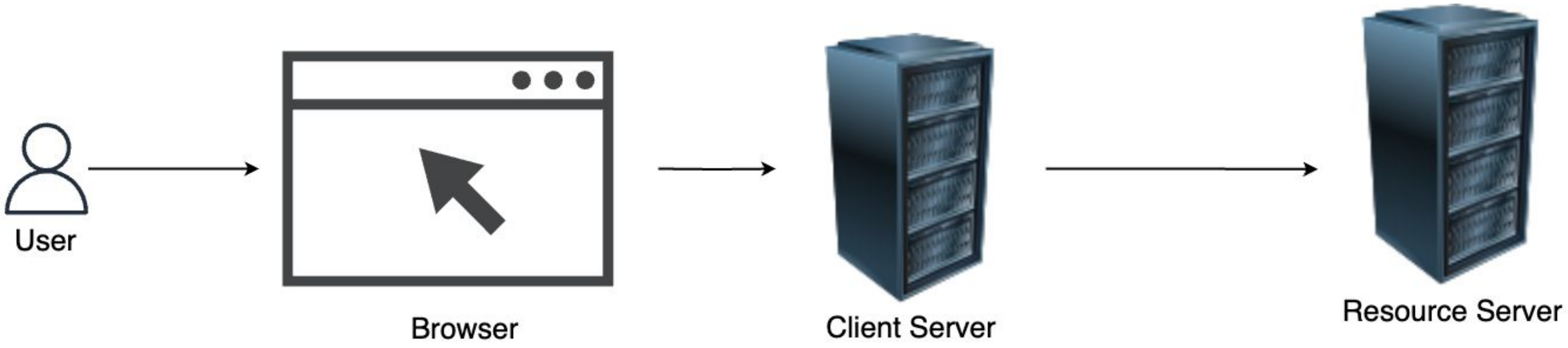
OAuth 2.0 Server Side (Confidential) Client Applications

youtube.com/@java-rush

🙋 What is it?

- ➢ Run on a Web Server
- ➢ Securely store client credentials
- ➢ Saft to use `client_secret`
- ➢ Use **Authorization Code Grant Flow** to get the access token
- ➢ All requests goes through Client Server



📜 Creating an Application

- Type of Application: Web Application
- Grant Type: Authorization Code
- Refresh Token: Yes (based on use case)
- Sign-in Redirect URIs
- Sign-out Redirect URIs
-

🤷 What we get?

- ➢ Client Id (Public Information)
- ➢ Client Secret (Keep it secret)

🫲 Sign-in Redirect URI

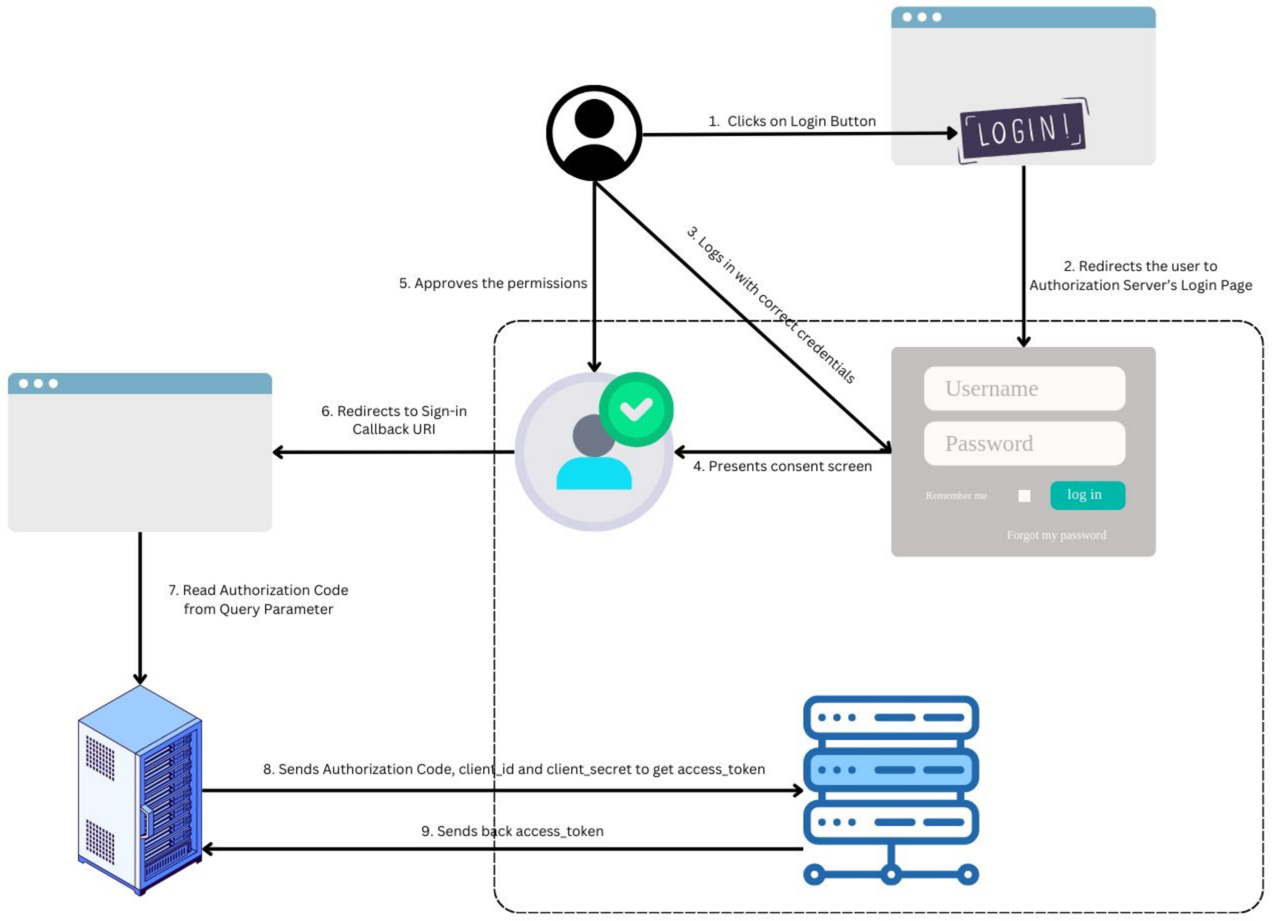
- ➢ Callback URI
- ➢ Must be registered
- ➢ To prevent redirection attacks
- ➢ May provide multiple values
- ➢ Must be https url

🏠 State

- ➢ String opaque to OAuth 2.0 service
- ➢ Client → Authorization Server
- ➢ Authorization Server → Client
- ➢ Advantages
  - Prevent CSRF using random string
  - Send user to correct location

🗝 Authorization Code Grant

- ➢ Client sends User to Authorization Server page to authorize
- ➢ User sees the information Client is requesting
- ➢ User approves the request
- ➢ Auth. Server redirects the user to Sign-in Redirect URI with Authorization Code
- ➢ Client receives the Authorization Code
- ➢ Client sends the Code to Auth. Server with **`client_id`** & **`client_secret`**
- ➢ Auth. Server verifies the information and sends the **`access_token`** to Client


1. Clicks on Login Button
LOGIN!
2. Redirects the user to Authorization Server's Login Page
3. Logs in with correct credentials
Username
Password
Remember me
log in
Forgot my password
4. Presents consent screen
5. Approves the permissions
6. Redirects to Sign-in Callback URI
7. Read Authorization Code from Query Parameter
8. Sends Authorization Code, client_id and client_secret to get access_token
9. Sends back access_token

🎉 Benefits of Authorization Code Grant Flow

- ➢ Reduce the risk of using authorization code by attacker
- ➢ Access token is not visible to the user

Responsibilities of OAuth 2 Client

1. Redirecting the user to Authorization Server
   a. Construct the authorization URL
   b. Send the user to the URL
2. Accept the Sign-in Redirect URL
   a. Read the authorization code from url
3. Get access token from the Authorization Server
   a. Send authorization code along with id and secret to Auth. Server
   b. Store the access token securely
4. Send requests to Resource Server
   a. Include access token along with request

Authorization URL

```
{AUTHORIZE_ENDPOINT}
    ?client_id={CLIENT_ID}
    &response_type={RESPONSE_TYPE}
    &state={STATE}
    &redirect_uri={REDIRECT_URI}
```

Sample Authorization URL 🔗

```
https://github.com/login/oauth/authorize
    ?client_id=Iv1.0b880a78910ade9d
    &response_type=code
    &state=bsoj89os
    &redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fauthorizati
on-code%2Fcallback
```

Response from Authorization

http://localhost:8080/Fauthorization-code/callback/?code={AUTHORIZATION_CODE}&state={STATE}

Get an access_token 🔑

Exchange authorization code for an access_token:

- ➢ grant_type: "authorization_code"
- ➢ code: Authorization code received after authorization
- ➢ redirect_uri: The Redirect URL
- ➢ Client Authentication: HTTP Basic Auth / POST Body Parameters

Access Token Url

```
{TOKEN_ENDPOINT}
    ?grant_type=code
    &code={AUTHORIZATION_CODE}
    &redirect_uri={REDIRECT_URI}
    &client_id={CLIENT_ID}
    &client_secret={CLIENT_SECRET}
```

Sample Access Token URL

```
https://github.com/login/oauth/access_token
    ?grant_type=code
    &code=abcdef123
    &redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fauthorizati
on-code%2Fcallback
    &client_id=Iv1.0b880a78910ade9d
    &client_secret=76ba18854
```

Access Token Response

```
{
    "access_token": “abcdlnsofn12e2…”,
    “refresh_token”: “jsojfo987q0uosn”,
    “expires_in”: 1800,
    “refresh_token_expires_in”: 3600,
    “scope”: <scopes>,
    “token_type”: “bearer”
}
```

Errors

{REDIRECT_URI}/?error={ERROR_CODE}&error_description={ERROR_DESCRIPTION}

- ➢ redirect_uri_mismatch
- ➢ Invalid client_id
- ➢ User denies the request
- ➢ Invalid Parameters
- ➢ invalid_scope

Using Access Token

```
GET {API_ENDPOINT}

    -H “Accept: application/json”

    -H “Authorization: Bearer {ACCESS_TOKEN 🔑}”

    …
```

⏭️ Next