

OAuth 2.0 with Keycloak

Use Keycloak as Identity Provider

TGIF, DKRZ, 11 September 2020



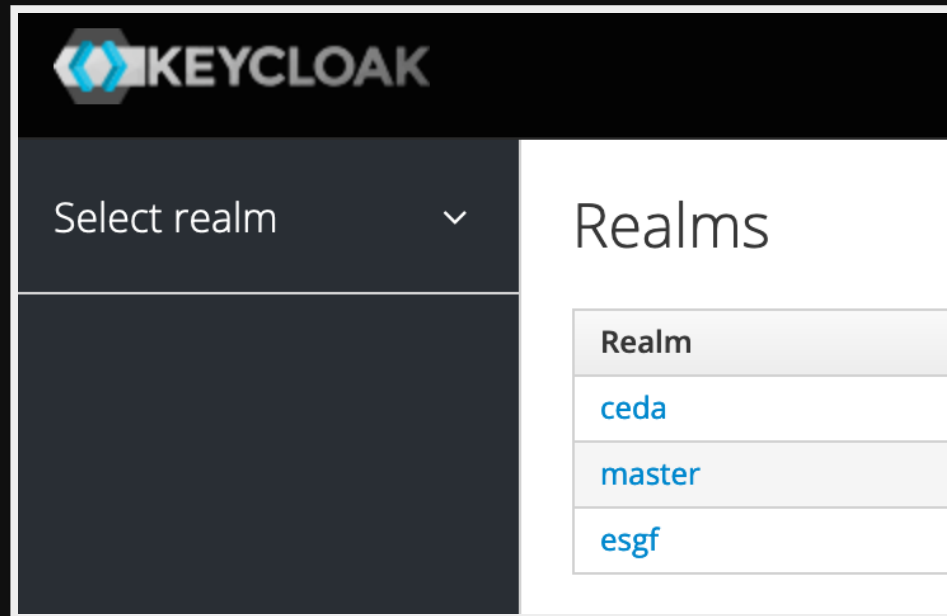
OAuth 2.0

- OAuth 2.0 is the modern standard for securing access to APIs.
- Used by Google, GitHub, etc
- First draft spec in 2010.
- A “framework” not a “protocol”.
- Still unfinished ... but most implementations very similar.
- <https://www.oauth.com/>

Keycloak

- Open Source Identity and Access Management
- Identity Provider - IDP
- Manage users and login to Portals.
- Protect resources: downloading in ESGF, access to processing.
- Supports OAuth2.
- <https://www.keycloak.org/>

Keycloak: ESGF Realm

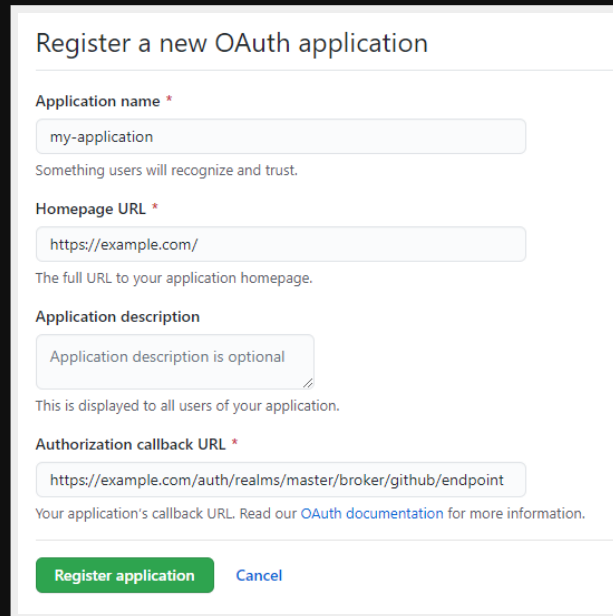


realm manages a set of users, credentials, roles,
and groups

Keycloak: Login to ESGF Portal

- Using Keycloak as Identity Provider
- Login to existing ESGF Portals
- Using OpenID-Connect ... OpenID Authentication using OAuth2

Register a Client



The screenshot shows a web form titled "Register a new OAuth application". It contains four main sections, each with a label, a text input field, and a descriptive note. The first section is "Application name" with the value "my-application" and the note "Something users will recognize and trust." The second is "Homepage URL" with the value "https://example.com/" and the note "The full URL to your application homepage." The third is "Application description" with the value "Application description is optional" and the note "This is displayed to all users of your application." The fourth is "Authorization callback URL" with the value "https://example.com/auth/realms/master/broker/github/endpoint" and the note "Your application's callback URL. Read our [OAuth documentation](#) for more information." At the bottom, there are two buttons: a green "Register application" button and a blue "Cancel" button.

Register a new OAuth application

Application name *
my-application
Something users will recognize and trust.

Homepage URL *
https://example.com/
The full URL to your application homepage.

Application description
Application description is optional
This is displayed to all users of your application.

Authorization callback URL *
https://example.com/auth/realms/master/broker/github/endpoint
Your application's callback URL. Read our [OAuth documentation](#) for more information.

Register application Cancel

- Clients (Apps) can request the Identity Provider to authenticate a user
- Relying Party (RP)

Client Details

1 user

Client ID

[REDACTED]

Client Secret

[REDACTED]

Revoke all user tokens

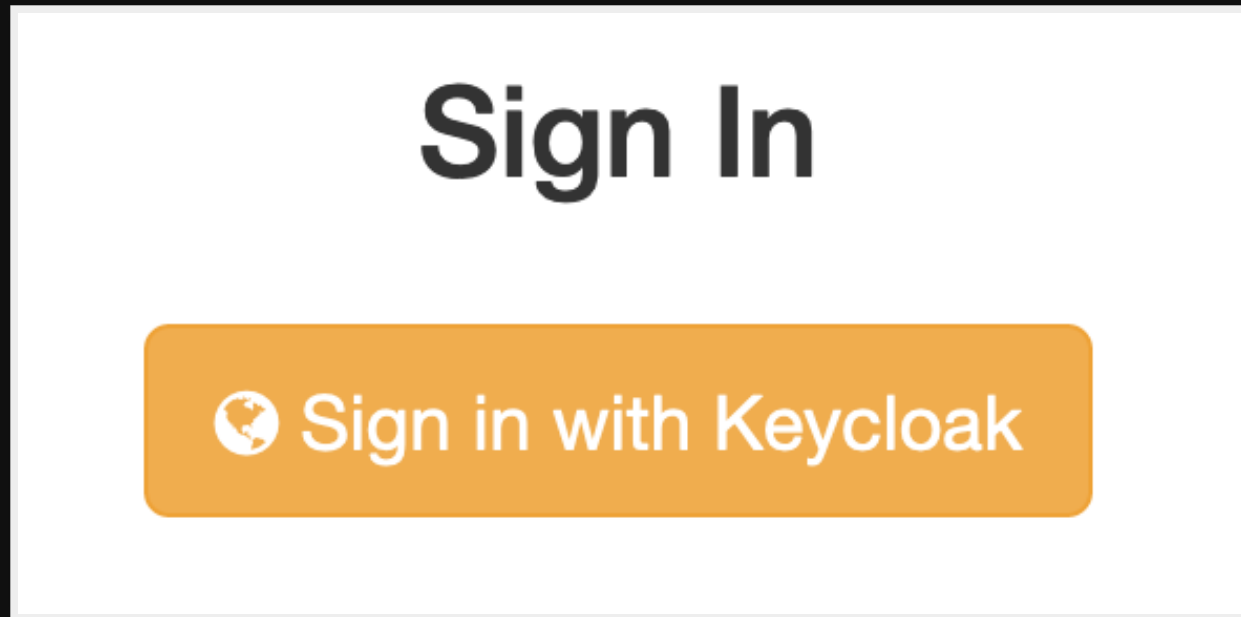
Reset client secret

User Registration

- Users are registered in Keycloak
- Username, Password, ...
- Multiple IDPs possible ... Sign-in with GitHub at ESGF IDP Proxy.

Sign-in to a Portal

<https://demo-phoenix.cloud.dkrz.de/>



Using redirection-based flow

Protect a Resource Server

Web Processing Service with token based access

Available Web Processing Services

 **Emu** 

WPS processes for testing and demos.

 **Hummingbird** 

A Web Processing Service for metadata compliance checks

Use an Access Token

Personal access token

Refresh Token

Access Token

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUiwiaw2IklIA6ICJ1VTdYTUpPWjRCejkzYTdGdmdaWEk1cDVXbWlwMnhmRk5YUm5PdmlXb0pNlnC
on5zS5dT1PdKST3ZbpejPFkyABOuZLuulnisauVE2PpuHJXuPIJQknz_2FGHMYGaiBswjfZw8J0PiZtWi4ebbmKKW-
e07W_AOqjDrQ0tWXdtbn6_VAy2syj-xw-
WATCJtD7BSu6bX7B8TCcXAB4MRHzVRSBhapgHtw4rpKHwBD5NqjdCGrQvxvEOq0fASHOVsDjKxhljewa2zk_Vp3CnQRWQw0o_mwp2

Expires at

2020-09-07 16:41:04 UTC

Use a string token valid for a short time.

JSON Web Tokens

- securely transmitting information between parties as a JSON.
- can be verified and trusted because it is digitally signed.

<https://jwt.io/>

Get a Token without redirect

```
import os
from oauthlib.oauth2 import BackendApplicationClient
from requests_oauthlib import OAuth2Session

os.environ['OAUTHLIB_INSECURE_TRANSPORT'] = '1'

client = BackendApplicationClient(client_id=client_id)
oauth = OAuth2Session(client=client)
token = oauth.fetch_token(
    token_url,
    scope='compute',
    client_id=client_id,
    client_secret=client_secret,
    include_client_id=True,
    verify=False)
```

From terminal or Jupyter notebook

Use Bearer Token to access Resource

Add token in header variable "Bearer" to access resource

Run request with birdy

```
: from birdy import WPSCClient
headers = {'Authorization': 'Bearer {}'.format(token['access_token'])}
emu = WPSCClient(url=base_url, headers=headers)

: response = emu.hello(name='Stranger')

: response.get()
```