OAuth 2.0 with Keycloak

Use Keycloak as Identity Provider

TGIF, DKRZ, 11 September 2020

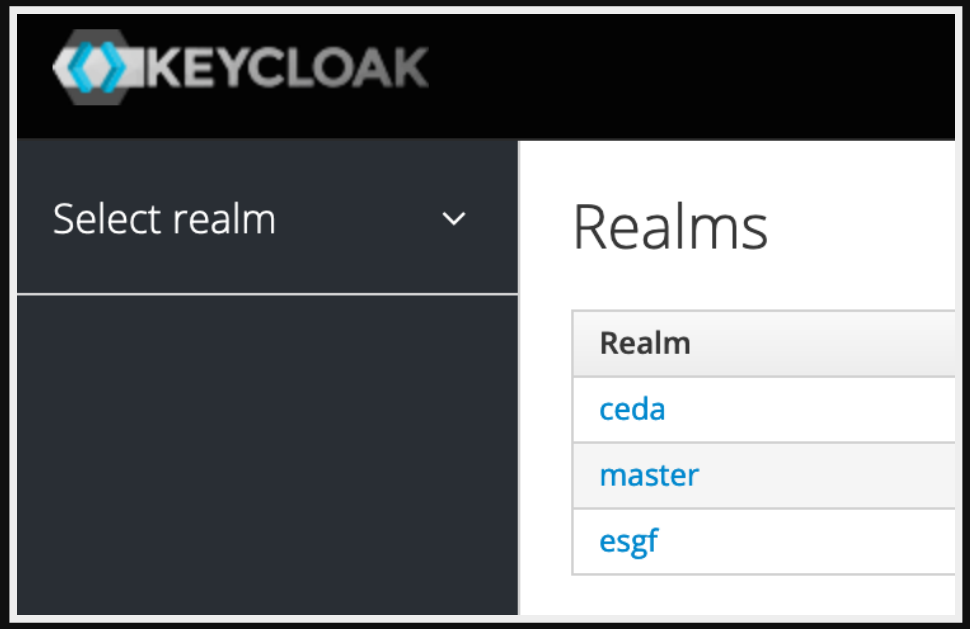
OAuth 2.0

- 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

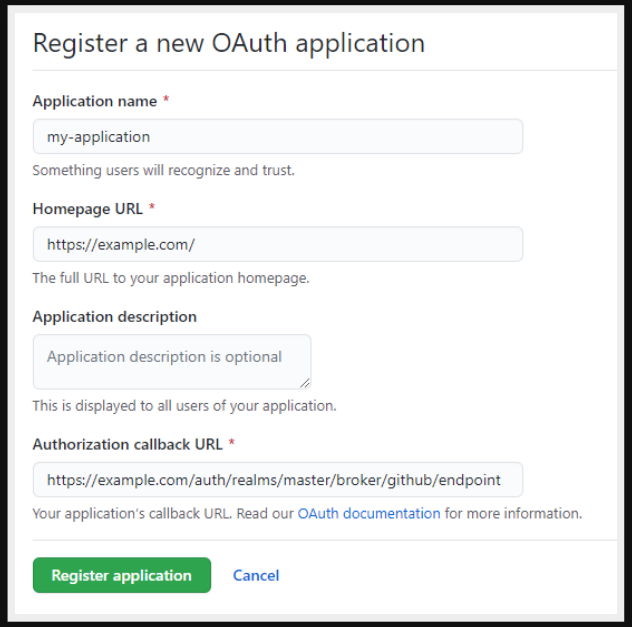


realm manages a set of users, credentials, roles, and groups

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



- 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

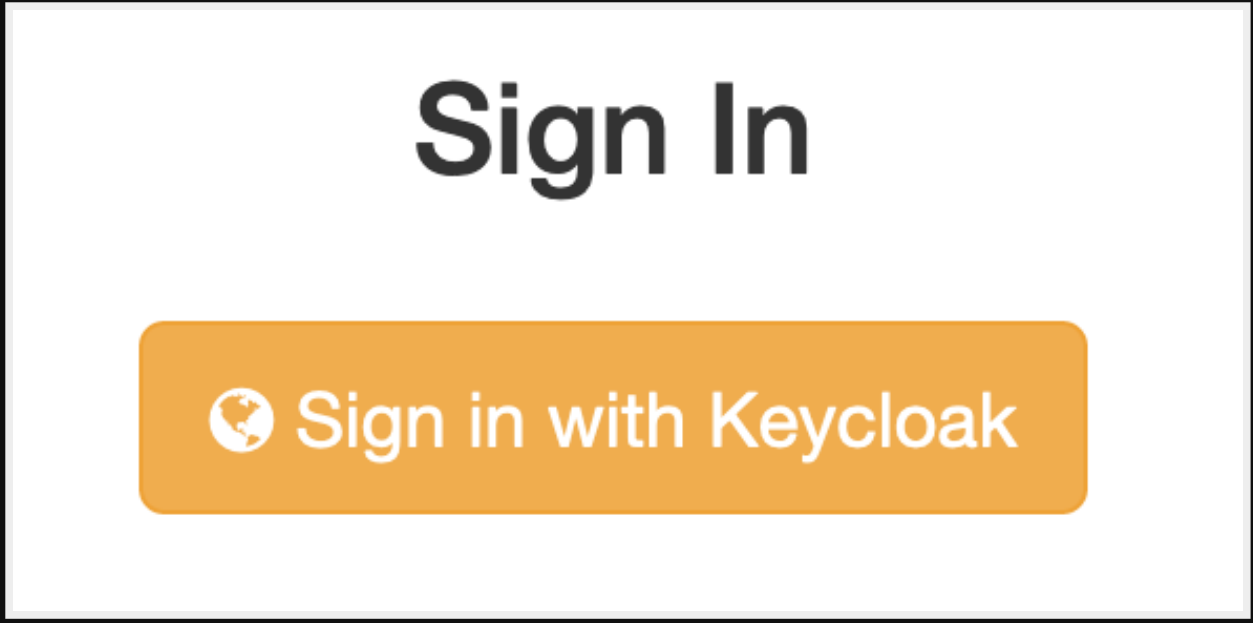
Reset client secret

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

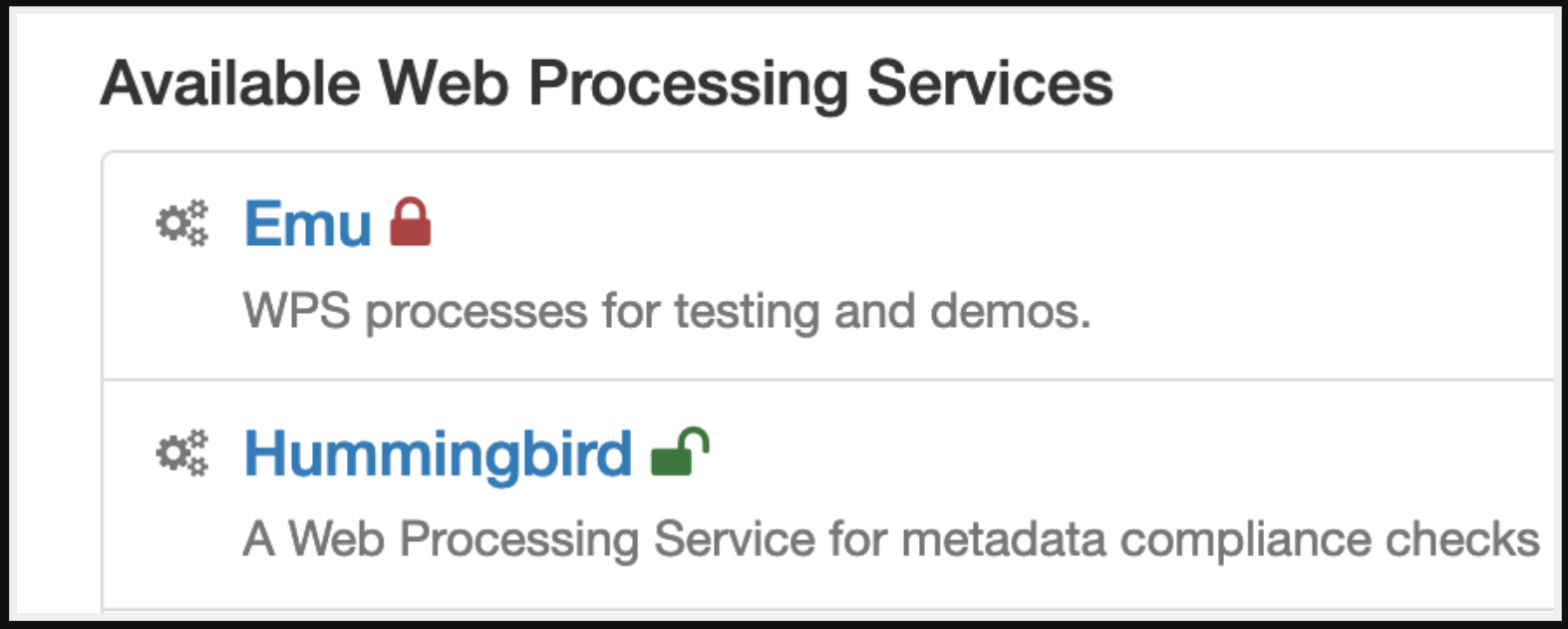
https://demo-phoenix.cloud.dkrz.de/



Using redirection-based flow

Protect a Resource Server

Web Processing Service with token based access



Use an Access Token

Personal access token

Refresh Token

Access Token

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJ1VTdYTUpPWjRCejkzYTdGdmdaWEk1cDVXbWlwMnhmRk5YUm5PdmlXb0pNIn0
on5zS5dT1PdKST3ZbpejPFkyABOuzLuuInisauVE2PpuHJXuPIJQknz_2FGHMYGaiBswjfZw8J0PiZtWi4ebbmKKW-
e07W_AOqjDrQ0tWXdtbn6_VAy2syj-xw-
WATCJtD7BSu6bX7B8TCcXAQb4MRHzVRSBhapgHtw4rpkHwBD5NqjdCGrQvxvEOq0fASHOVsDjKxhIjewa2zk_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

```python
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 WPSClient
headers = {'Authorization': 'Bearer {}'.format(token['access_token'])}
emu = WPSClient(url=base_url, headers=headers)
```

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

```
response.get()
```