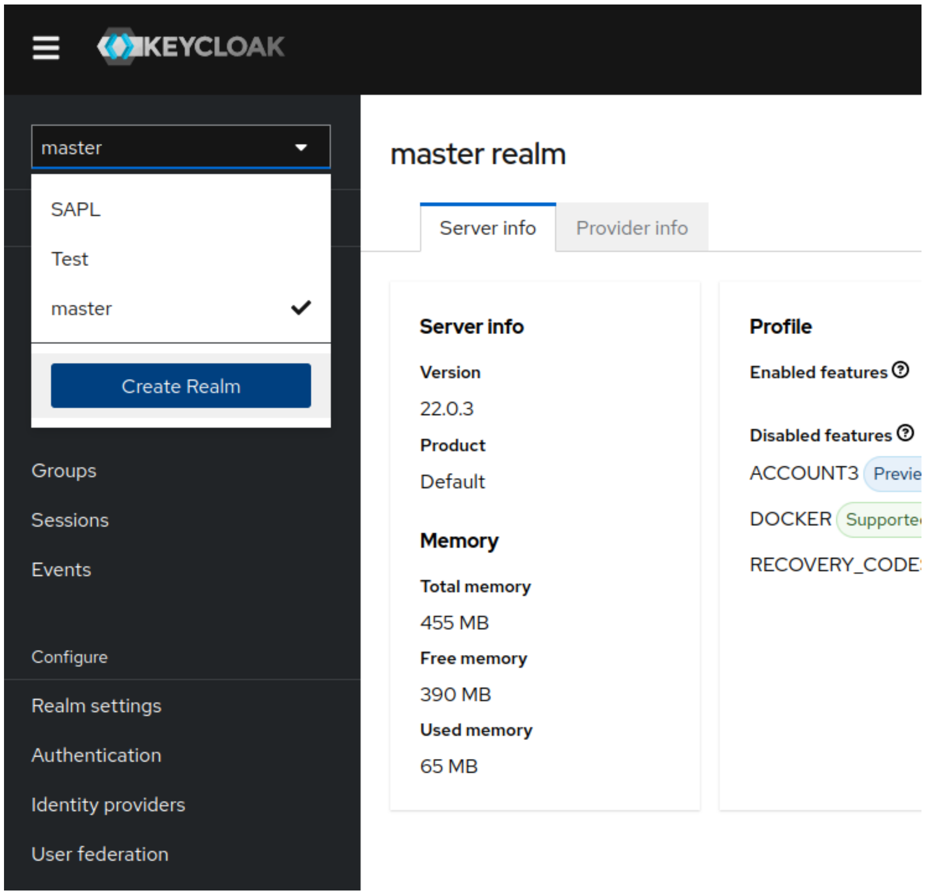
Configuring Keycloak

The SAPL Server CE can also be used with its own Keycloak realm. This means that existing users can be used for login. Keycloak still needs to be set up so that Keycloak can be used for the SAPL Server CE. The following steps show the necessary configurations for using OAuth2 with Keycloak.

In most cases, a Keycloak realm in which the SAPL client is to be created will already exist. If not, then the realm must first be created in Keycloak via 'Create Realm'.

Create a new realm

The first step is to create a realm if one does not already exist. With existing setups, it is possible that a realm has already been created for other clients that are shared:



Server info

Version
22.0.3

Product
Default

Memory

Total memory
455 MB

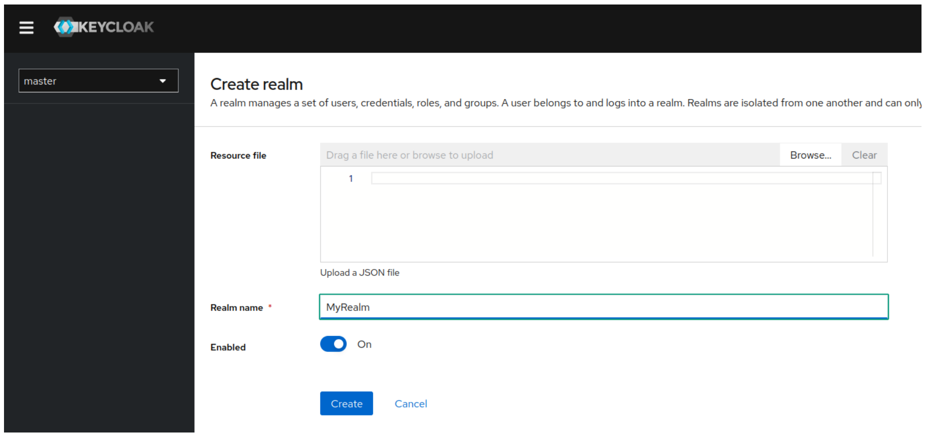
Free memory
390 MB

Used memory
65 MB

Profile

Enabled features

Disabled features
ACCOUNT3 Previe
DOCKER Supporte
RECOVERY_CODE



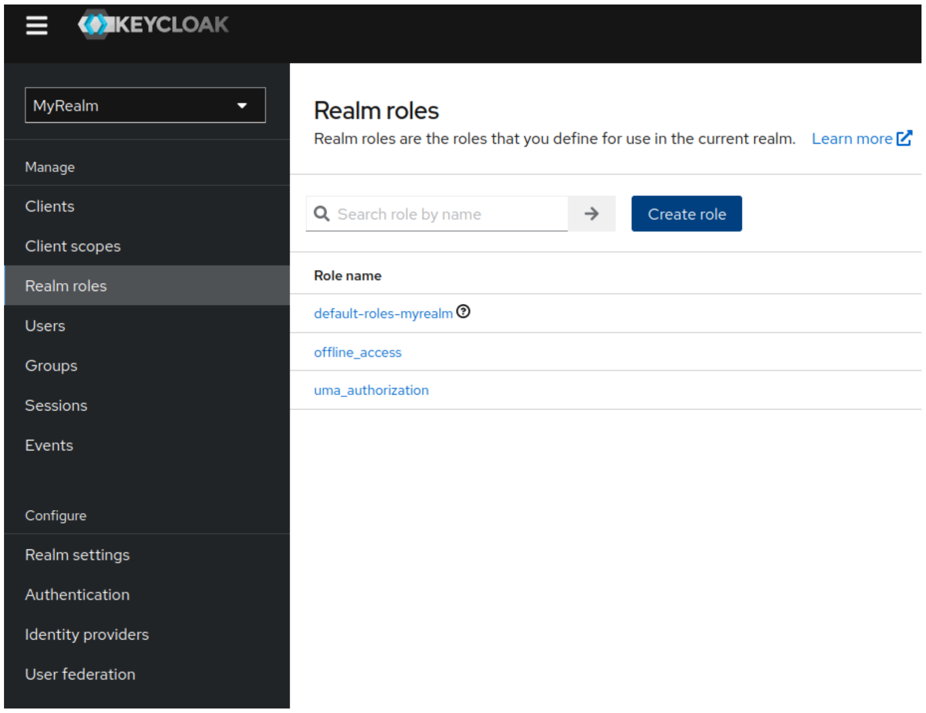
Create realm

A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only

Resource file	Drag a file here or browse to upload — Upload a JSON file
Realm name *	MyRealm
Enabled	On

Create the ADMIN role

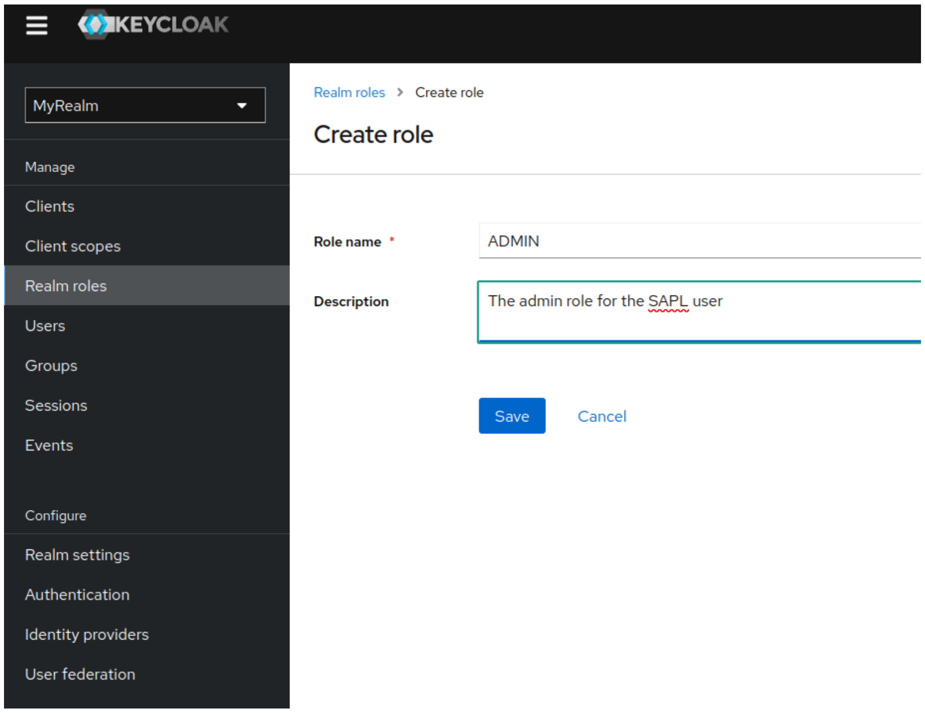
Now we can create the realm roles that are necessary so that users can be assigned the "ADMIN" role.



Realm roles

Realm roles are the roles that you define for use in the current realm. Learn more

Role name
default-roles-myrealm
offline_access
uma_authorization



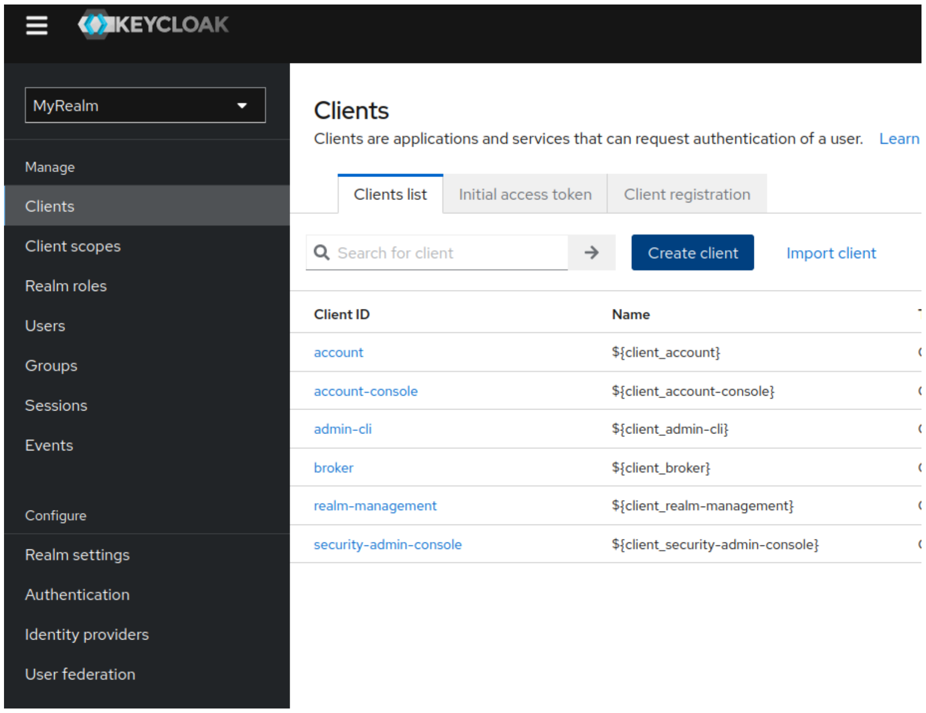
Create role

Role name *: ADMIN

Description: The admin role for the SAPL user

Creating the client and adding the client scope

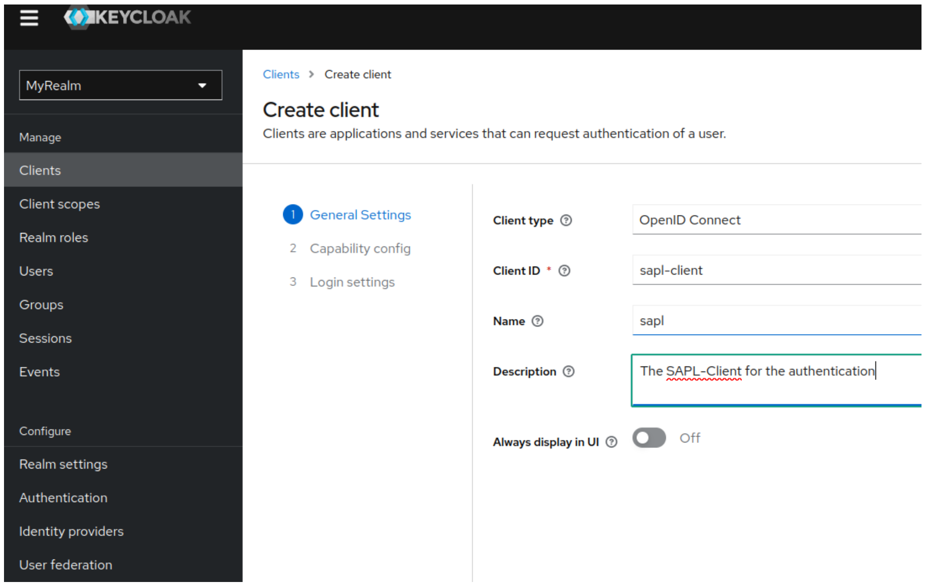
In the next step, we create the client in the realm, which we use to authenticate ourselves on the realm. We also change the client scope so that the correct attributes are sent when the token is transmitted.



Clients

Clients are applications and services that can request authentication of a user. Learn

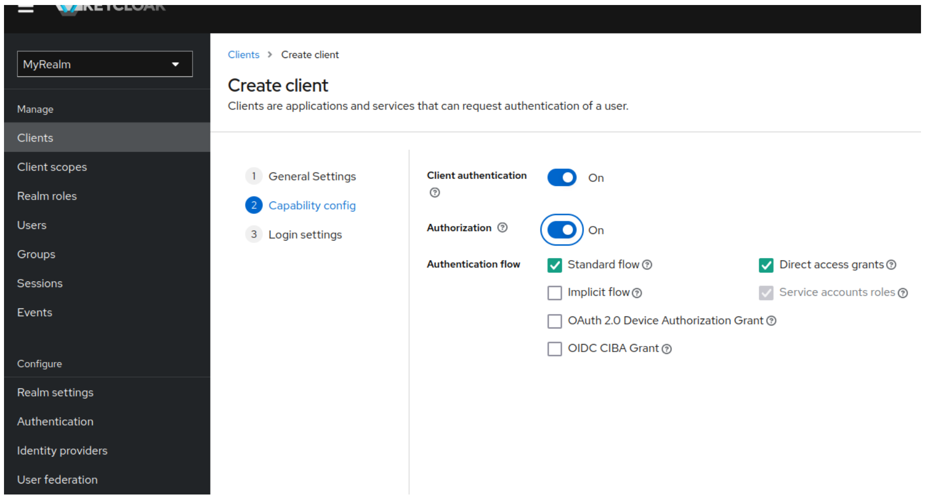
Client ID	Name
account	${client_account}
account-console	${client_account-console}
admin-cli	${client_admin-cli}
broker	${client_broker}
realm-management	${client_realm-management}
security-admin-console	${client_security-admin-console}



Create client

Clients are applications and services that can request authentication of a user.

Client type	OpenID Connect
Client ID *	sapl-client
Name	sapl
Description	The SAPL-Client for the authentication
Always display in UI	Off



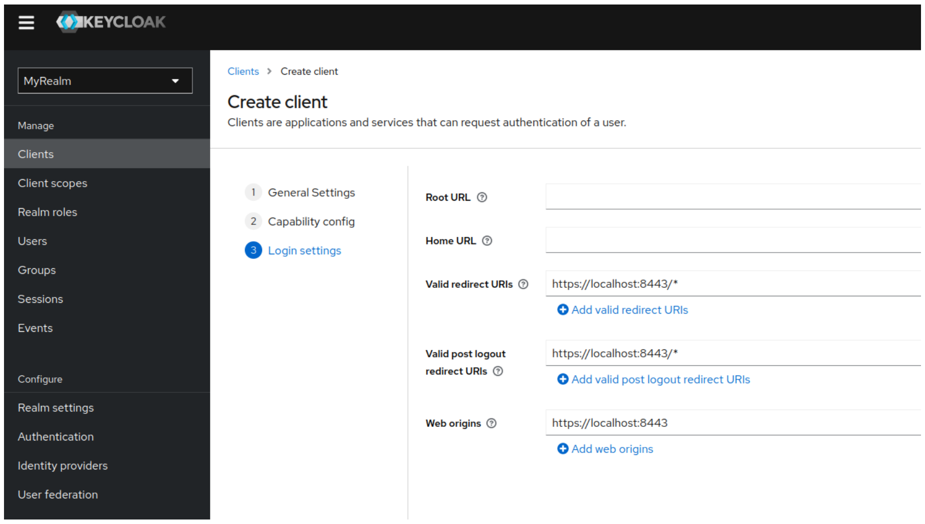
Create client

Clients are applications and services that can request authentication of a user.

Client authentication: On

Authorization: On

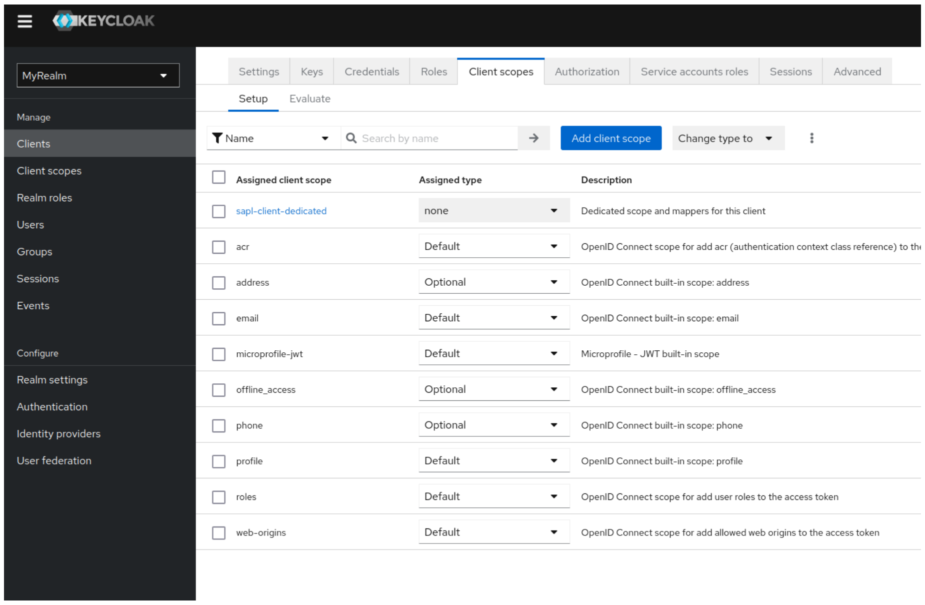
Authentication flow: [✓ Standard flow] [✓ Direct access grants] Implicit flow [✓ Service accounts roles] OAuth 2.0 Device Authorization Grant OIDC CIBA Grant



Create client

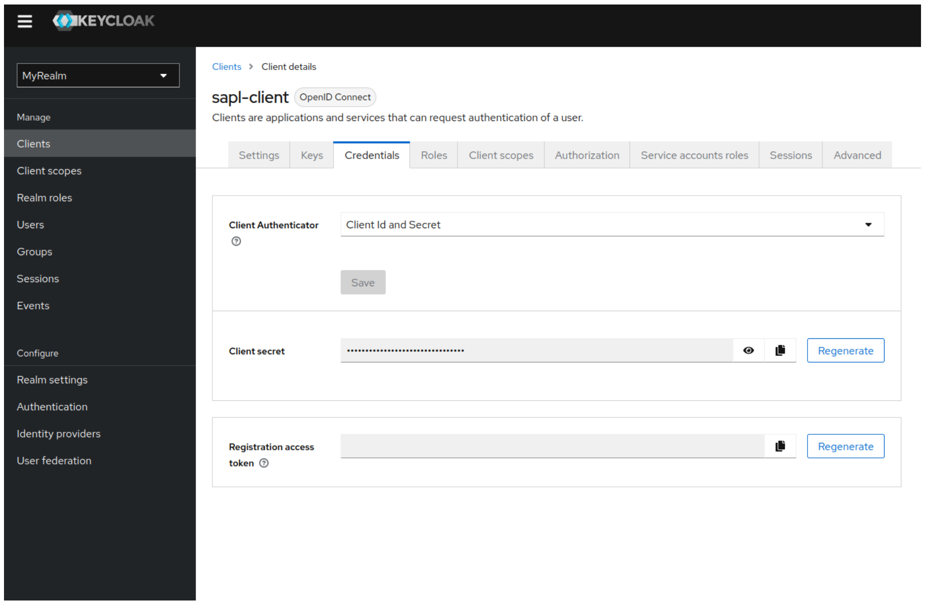
Clients are applications and services that can request authentication of a user.

Root URL	
Home URL	
Valid redirect URIs	https://localhost:8443/*
Valid post logout redirect URIs	https://localhost:8443/*
Web origins	https://localhost:8443



Assigned client scope	Assigned type	Description
sapl-client-dedicated	none	Dedicated scope and mappers for this client
acr	Default	OpenID Connect scope for add acr (authentication context class reference) to th
address	Optional	OpenID Connect built-in scope: address
email	Default	OpenID Connect built-in scope: email
microprofile-jwt	Default	Microprofile - JWT built-in scope
offline_access	Optional	OpenID Connect built-in scope: offline_access
phone	Optional	OpenID Connect built-in scope: phone
profile	Default	OpenID Connect built-in scope: profile
roles	Default	OpenID Connect scope for add user roles to the access token
web-origins	Default	OpenID Connect scope for add allowed web origins to the access token

Under the Credentials tab, we can copy the client secret, which we will need later for the configuration of the SAPL client in application.yml.



sapl-client OpenID Connect

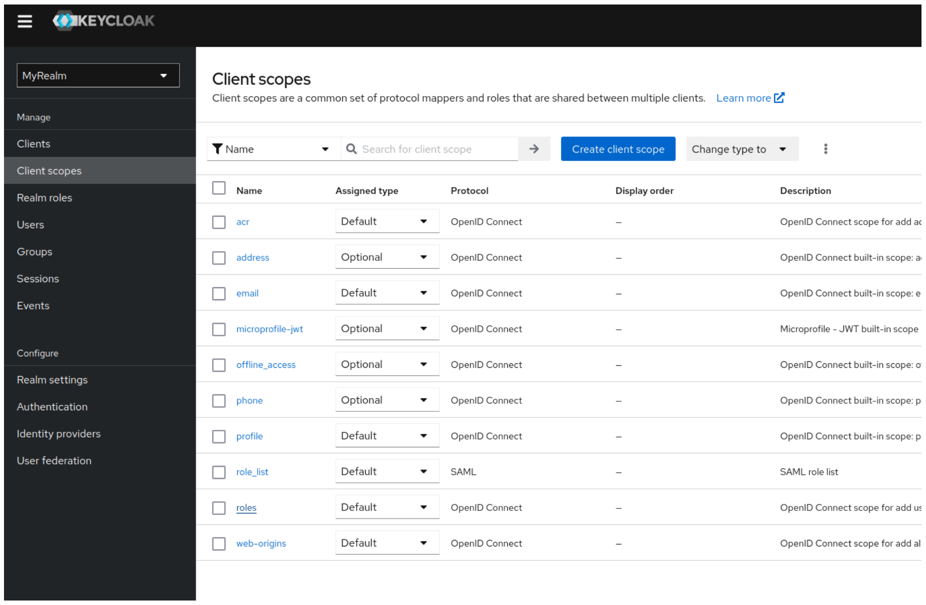
Clients are applications and services that can request authentication of a user.

Client Authenticator: Client Id and Secret

Client secret: ••••••••••••••••••••••••••••••••

Registration access token

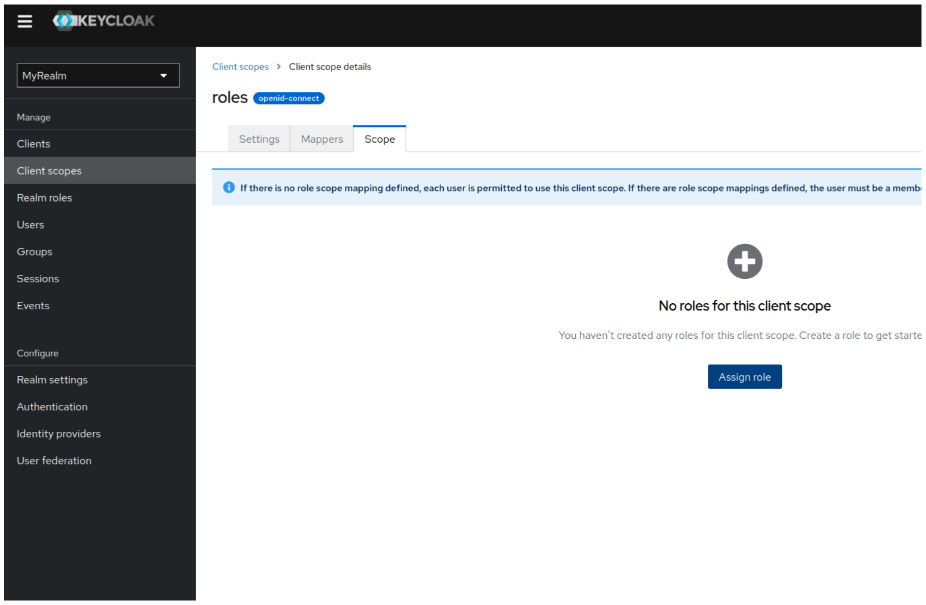
Now we add the ADMIN role to the client scope “roles”:



Client scopes

Client scopes are a common set of protocol mappers and roles that are shared between multiple clients. Learn more

Name	Assigned type	Protocol	Display order	Description
acr	Default	OpenID Connect	–	OpenID Connect scope for add ac
address	Optional	OpenID Connect	–	OpenID Connect built-in scope: a
email	Default	OpenID Connect	–	OpenID Connect built-in scope: e
microprofile-jwt	Optional	OpenID Connect	–	Microprofile - JWT built-in scope
offline_access	Optional	OpenID Connect	–	OpenID Connect built-in scope: o
phone	Optional	OpenID Connect	–	OpenID Connect built-in scope: p
profile	Default	OpenID Connect	–	OpenID Connect built-in scope: p
role_list	Default	SAML	–	SAML role list
roles	Default	OpenID Connect	–	OpenID Connect scope for add us
web-origins	Default	OpenID Connect	–	OpenID Connect scope for add al

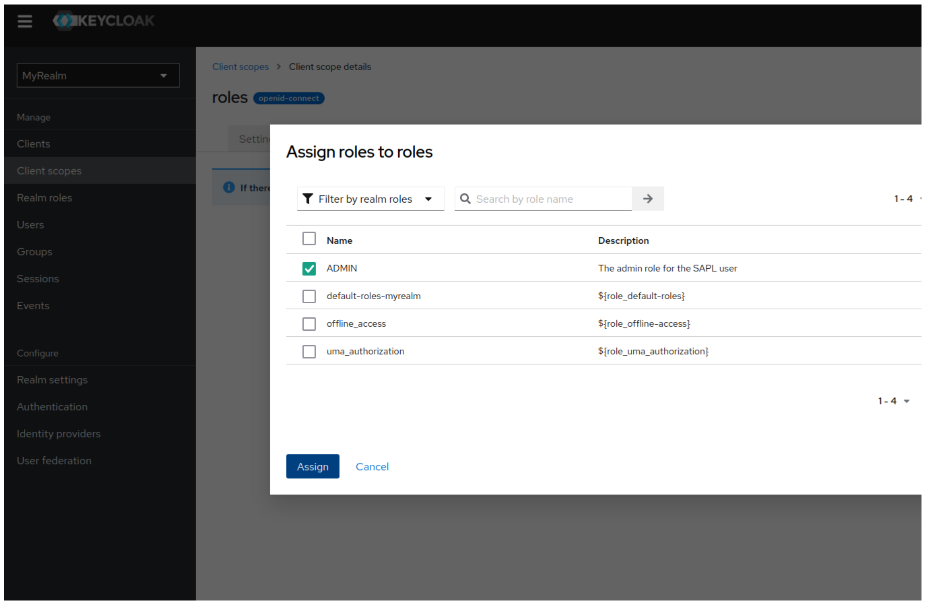


roles openid-connect

If there is no role scope mapping defined, each user is permitted to use this client scope. If there are role scope mappings defined, the user must be a memb

No roles for this client scope

You haven't created any roles for this client scope. Create a role to get starte



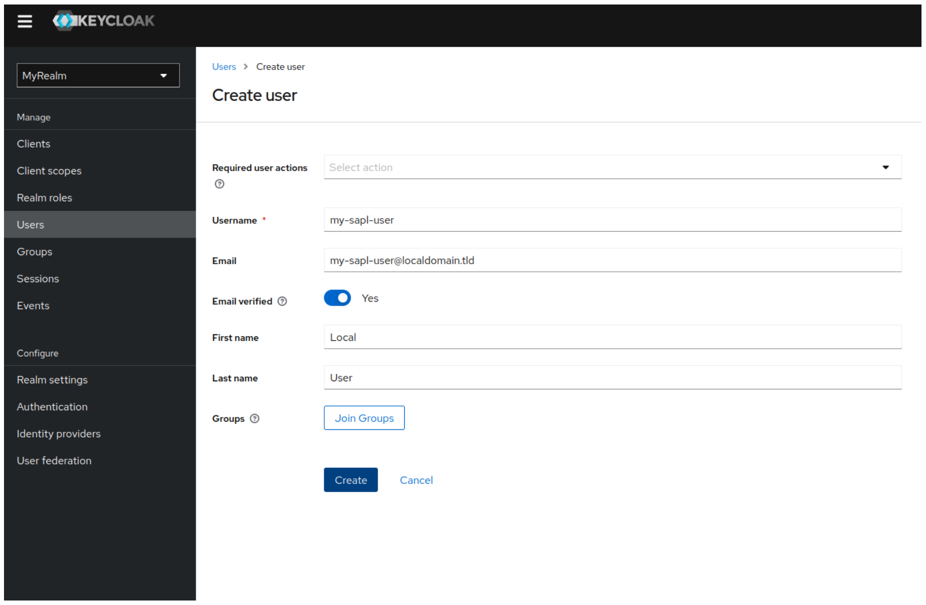
Assign roles to roles

Filter by realm roles | Search by role name | 1 - 4

	Name	Description
✓	ADMIN	The admin role for the SAPL user
	default-roles-myrealm	${role_default-roles}
	offline_access	${role_offline-access}
	uma_authorization	${role_uma_authorization}

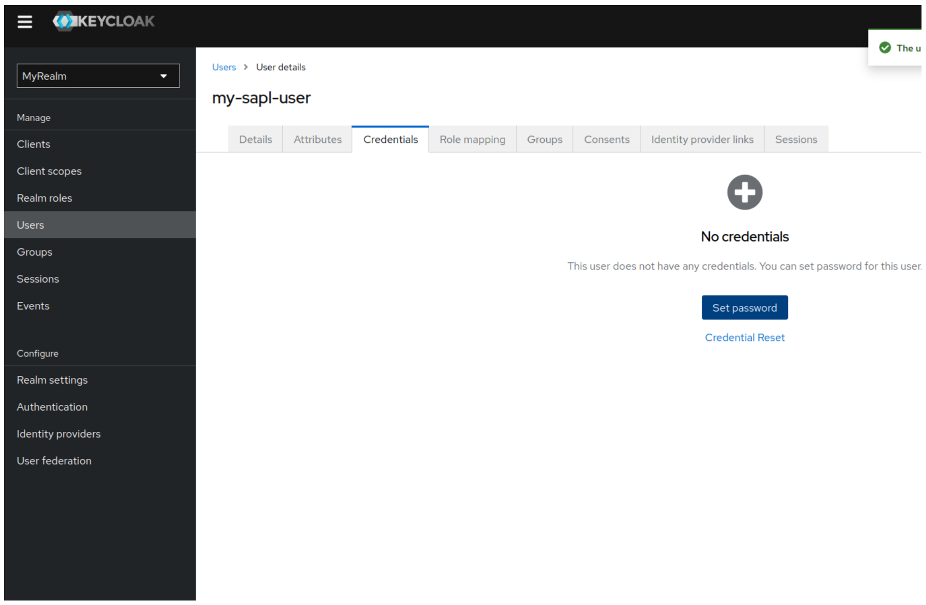
Creating a new user

We can now create a user in Keycloak for test purposes. We need this user to log in to the SAPL Server CE. We must explicitly set a password and assign the appropriate role to the user:



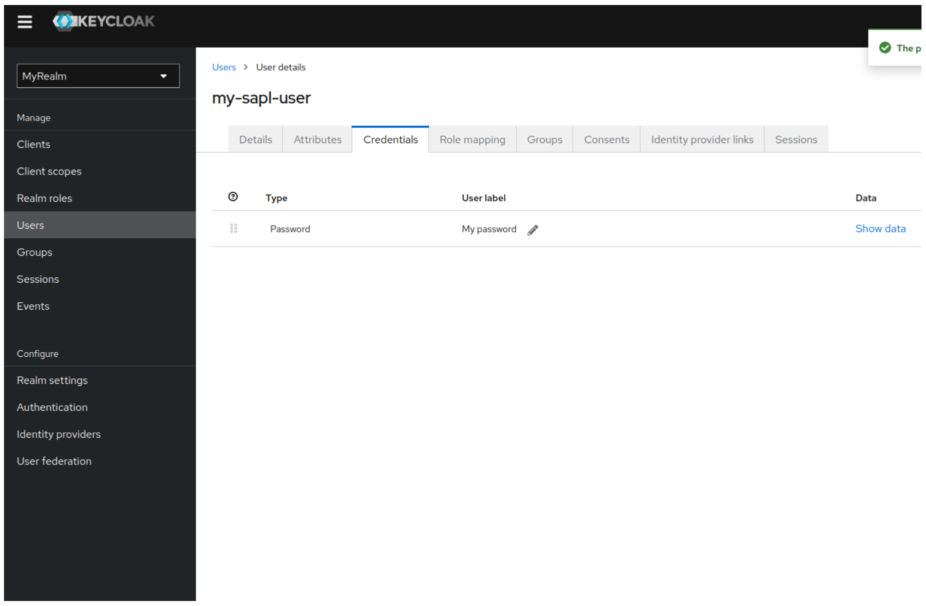
Create user

Required user actions	Select action
Username *	my-sapl-user
Email	my-sapl-user@localdomain.tld
Email verified	Yes
First name	Local
Last name	User
Groups	Join Groups

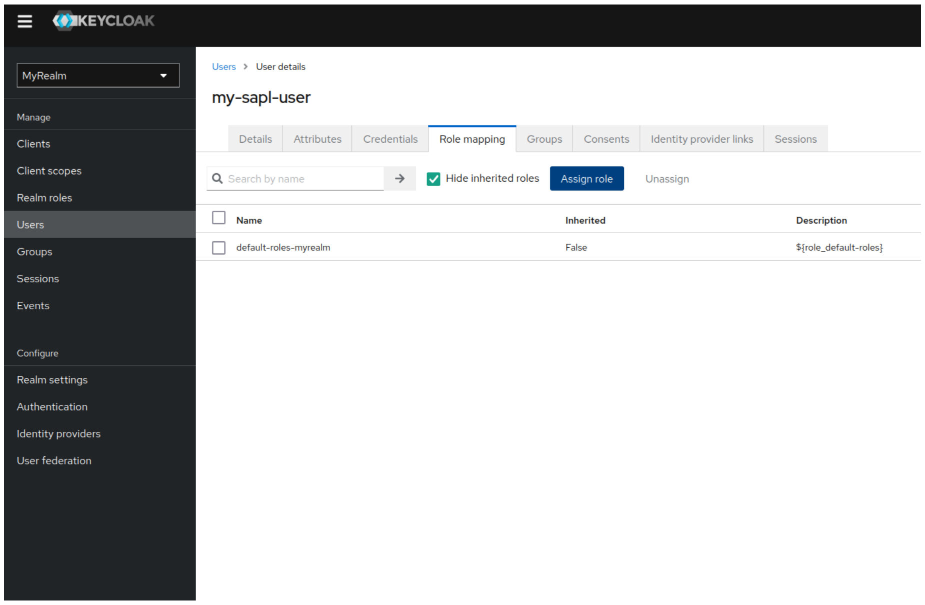


No credentials

This user does not have any credentials. You can set password for this user.

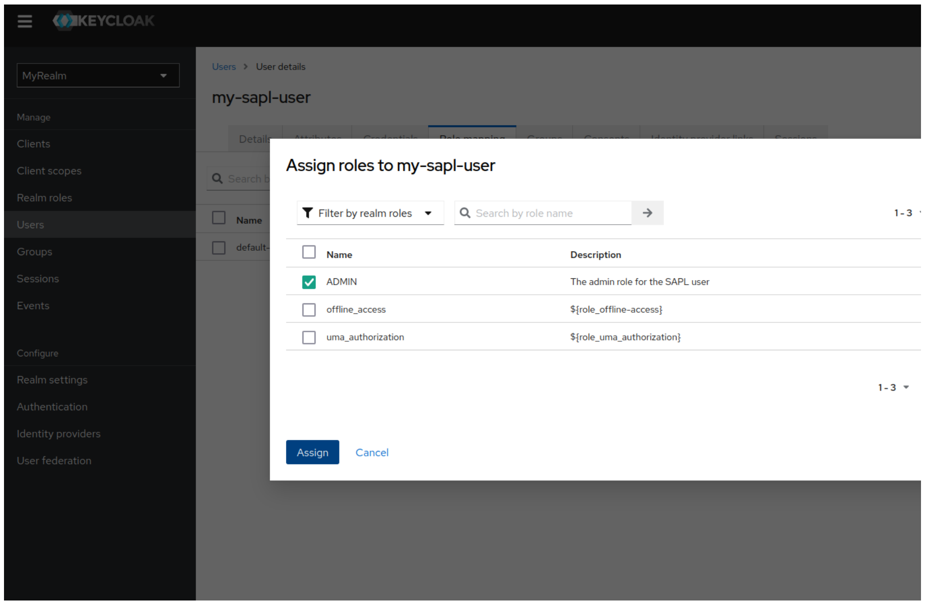


Type	User label	Data
Password	My password	Show data



my-sapl-user

Name	Inherited	Description
default-roles-myrealm	False	${role_default-roles}

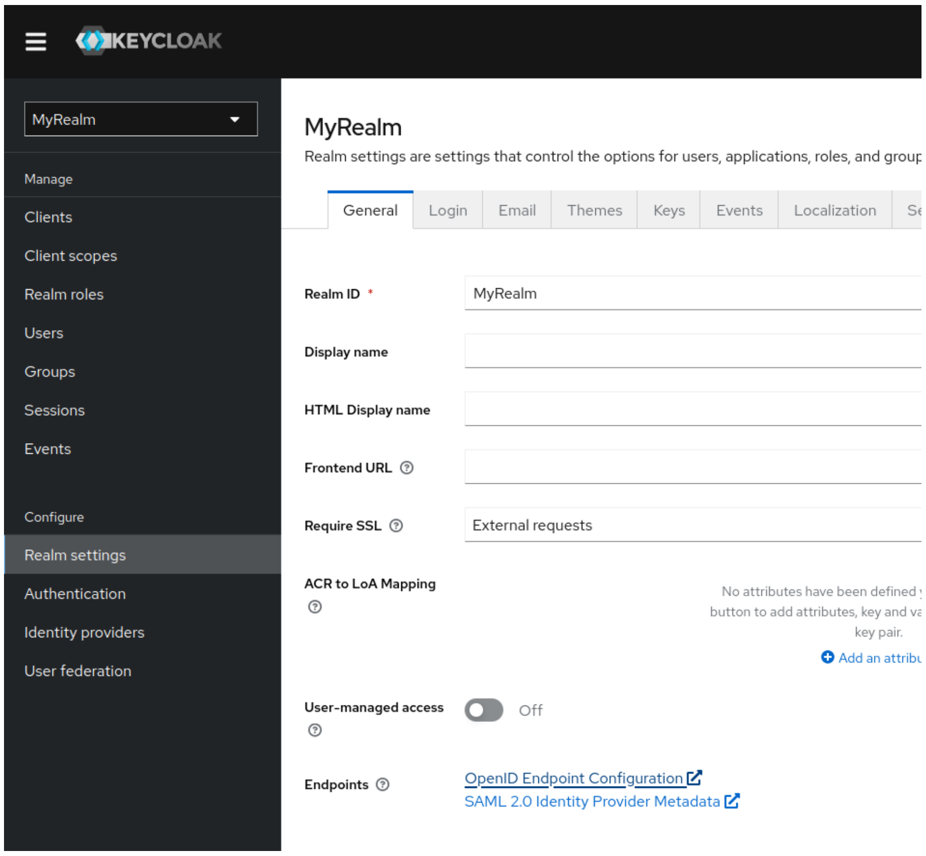


Assign roles to my-sapl-user

	Name	Description
✓	ADMIN	The admin role for the SAPL user
	offline_access	${role_offline-access}
	uma_authorization	${role_uma_authorization}

Display the realm config

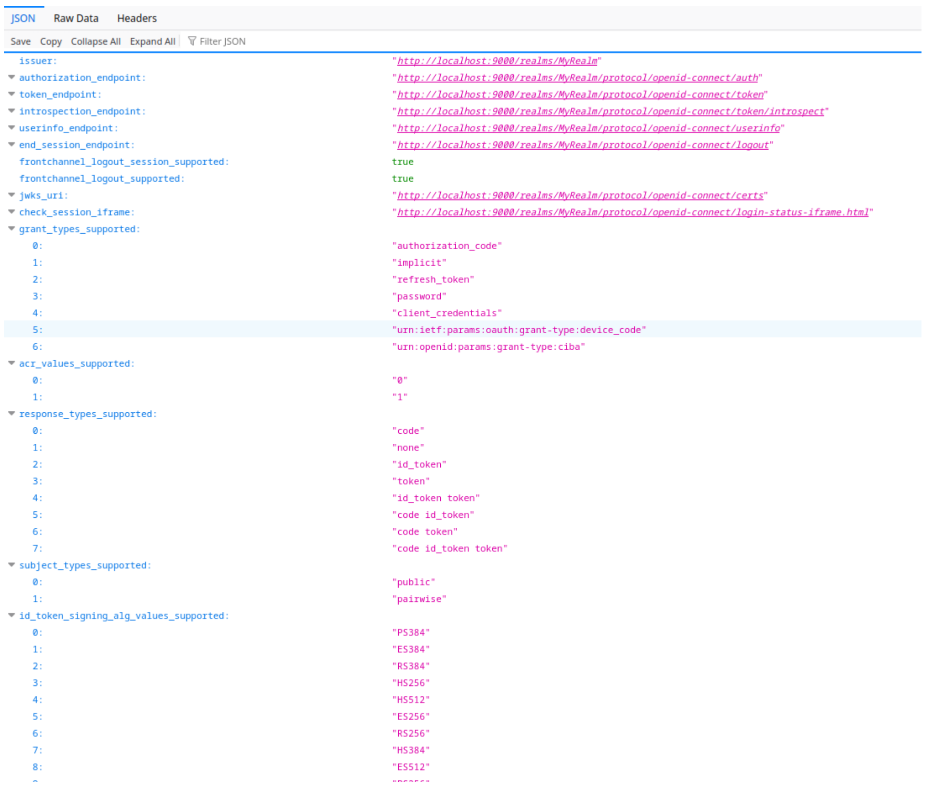
There is an overview of the realm configuration in Keycloak. This configuration is important as we need some values in our application.yml:



MyRealm

Realm settings are settings that control the options for users, applications, roles, and group

Realm ID *	MyRealm
Display name	
HTML Display name	
Frontend URL	
Require SSL	External requests
ACR to LoA Mapping	No attributes have been defined button to add attributes, key and va key pair. Add an attribu
User-managed access	Off
Endpoints	OpenID Endpoint Configuration SAML 2.0 Identity Provider Metadata



```
issuer:                                 "http://localhost:9000/realms/MyRealm"
authorization_endpoint:                 "http://localhost:9000/realms/MyRealm/protocol/openid-connect/auth"
token_endpoint:                         "http://localhost:9000/realms/MyRealm/protocol/openid-connect/token"
introspection_endpoint:                 "http://localhost:9000/realms/MyRealm/protocol/openid-connect/token/introspect"
userinfo_endpoint:                      "http://localhost:9000/realms/MyRealm/protocol/openid-connect/userinfo"
end_session_endpoint:                   "http://localhost:9000/realms/MyRealm/protocol/openid-connect/logout"
frontchannel_logout_session_supported:  true
frontchannel_logout_supported:          true
jwks_uri:                               "http://localhost:9000/realms/MyRealm/protocol/openid-connect/certs"
check_session_iframe:                   "http://localhost:9000/realms/MyRealm/protocol/openid-connect/login-status-iframe.html"
grant_types_supported:
  0:                                    "authorization_code"
  1:                                    "implicit"
  2:                                    "refresh_token"
  3:                                    "password"
  4:                                    "client_credentials"
  5:                                    "urn:ietf:params:oauth:grant-type:device_code"
  6:                                    "urn:openid:params:grant-type:ciba"
acr_values_supported:
  0:                                    "0"
  1:                                    "1"
response_types_supported:
  0:                                    "code"
  1:                                    "none"
  2:                                    "id_token"
  3:                                    "token"
  4:                                    "id_token token"
  5:                                    "code id_token"
  6:                                    "code token"
  7:                                    "code id_token token"
subject_types_supported:
  0:                                    "public"
  1:                                    "pairwise"
id_token_signing_alg_values_supported:
  0:                                    "PS384"
  1:                                    "ES384"
  2:                                    "RS384"
  3:                                    "HS256"
  4:                                    "HS512"
  5:                                    "ES256"
  6:                                    "RS256"
  7:                                    "HS384"
  8:                                    "ES512"
```

Configuring Keycloak for

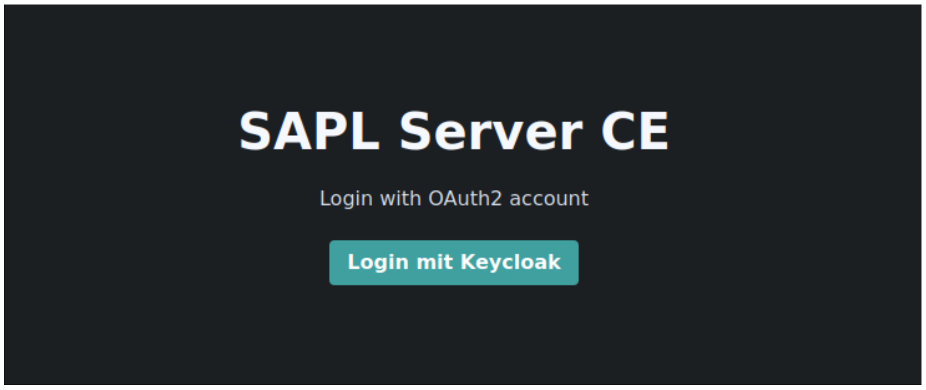
Now we have to configure the application.yml so that the SAPL Server CE also uses our Keycloak Client for authentication. The configuration looks like this:

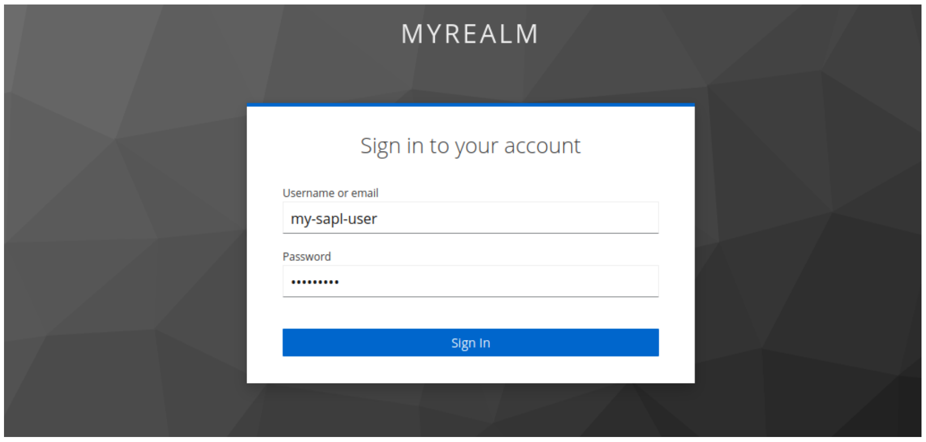
```
spring.security.oauth2.client:
  registration.keycloak:
    client-id: <Your SAPL client id e.g. sapl-client>
    client-secret: <Your SAPL client secret>
    client-authentication-method: client_secret_basic
    authorization-grant-type: authorization_code
    redirect-uri: "{baseUrl}/login/oauth2/code/keycloak"
    scope: openid, profile, email, roles
    provider: keycloak
  provider.keycloak:
    issuer-uri: <Issuer URI under issuer:>
    user-name-attribute: preferred_username
    jwk-set-uri: <JWK Set URI under jwks_uri:>
    authorization-uri: <Authorization URI under authorization_endpoint:>
    token-uri: <Token URI under token_endpoint:>
    user-info-uri: <User Info URI under userinfo_endpoint:>
```

The last necessary step is to set the parameter `allowOAuth2Login: True` in the application.yml. Now your SAPL Server CE will show an OAuth2 Login page.

Starting the server

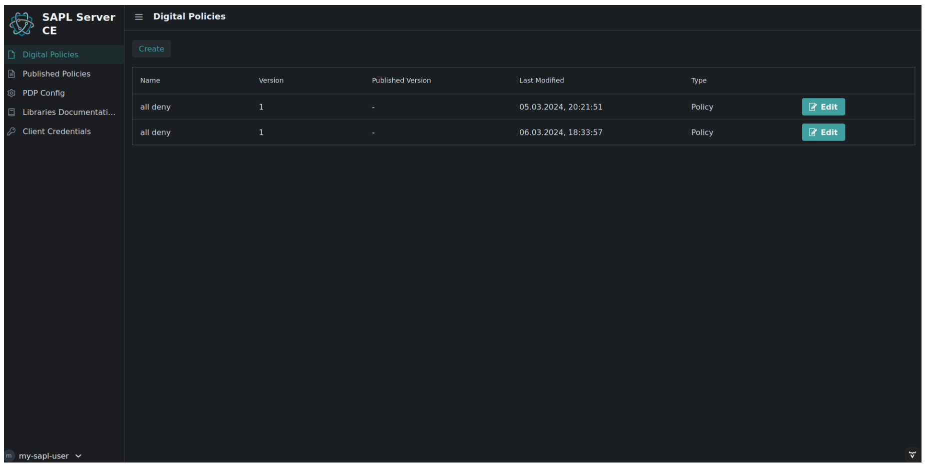
If the SAPL Server CE is now restarted, a new login page appears, which forwards you to the OAuth2 provider. We can now log in with our newly created user:





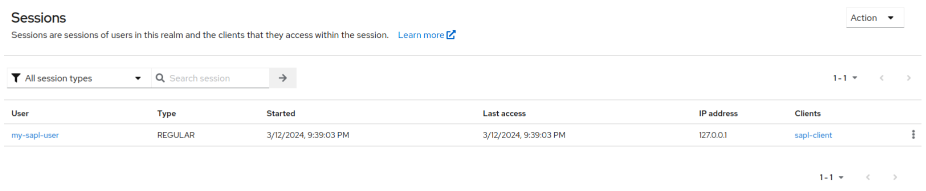
Username or email: my-sapl-user

Password: •••••••••



Name	Version	Published Version	Last Modified	Type
all deny	1	-	05.03.2024, 20:21:51	Policy
all deny	1	-	06.03.2024, 18:33:57	Policy

The user session now appears in the Keycloak Realm:



Sessions

Sessions are sessions of users in this realm and the clients that they access within the session. Learn more

User	Type	Started	Last access	IP address	Clients
my-sapl-user	REGULAR	3/12/2024, 9:39:03 PM	3/12/2024, 9:39:03 PM	127.0.0.1	sapl-client