

Royaume du Maroc
Université Mohammed Premier - Oujda
École Nationale des Sciences Appliquées - Oujda

Speciality: Information Security & Cyber Security

Personal Security Project Report

Under the theme:

Self-hosted IAM Solution Implementation

Realized by:

AMAHROUK Asmae

-2025-

Table of Contents

Table of Contents.....	1
Table of Figures.....	2
Summary	4
Introduction.....	5
Chapter 1: Implementing the User Authentication & Authorization part	15
Chapter 2: Implementing the API security part	44
Conclusion	50
Webography	51

Table of Figures

Figure 1: IdP workflow	8
Figure 2 : Single Sign-On workflow	9
Figure 3 : IAM Solution Architecture	12
Figure 4 : Updating the system.....	13
Figure 5 : Installing Docker	13
Figure 6 : Enabling Docker	13
Figure 7 : Docker version	14
Figure 8 : Installing OpenLDAP	16
Figure 9 : Installing OpenLDAP Admin GUI.....	17
Figure 10 : Starting phpldapadmin	17
Figure 11 : Listing OpenLDAP active containers	17
Figure 12 : OpenLDAP admin GUI interface	18
Figure 13 : Logged in OpenLDAP Admin GUI	19
Figure 14 : Adding user organizational unit step 1	20
Figure 15 : Adding user organizational unit step 2	20
Figure 16 : Adding a group step 1.....	21
Figure 17 : Adding a group step 2.....	21
Figure 18 : Adding a user step 1.....	22
Figure 19 : Adding a user step 2.....	22
Figure 20 : Entering john doe user's information	23
Figure 21 : Testing user existence using GUI interface.....	23
Figure 22 : Testing user existence using CLI.....	24
Figure 23 : Keycloak Installation	25
Figure 24 : Keycloak Container listing.....	25
Figure 25 : Keycloak Admin Console	26
Figure 26 : Keycloak admin console after logging in	26
Figure 27 : creating a realm in Keycloak.....	27
Figure 28 : creating a realm in Keycloak 2	27
Figure 29 : mycompany realm created in Keycloak	28
Figure 30 : Connect OpenLDAP with Keycloak step 1	28
Figure 31 : Connect OpenLDAP with Keycloak step 2	29
Figure 32 : Get OpenLDAP IP address.....	29
Figure 33 : Testing the existence of user jdoe in Keycloak.....	30
Figure 34 : User account log in interface.....	30
Figure 35 : Logging in as jdoe in Keycloak	31
Figure 36 : Logged in as the user John Doe.....	32
Figure 37 : Users and groups added in OpenLDAP	32
Figure 38 : Users accessed by Keycloak	33
Figure 39 : Implementing SSO in Keycloak.....	33
Figure 40 : Implementing MFA in Keycloak 1	34
Figure 41 : Implementing MFA in Keycloak 2	35
Figure 42 : Implementing MFA in Keycloak 3 – duplicating the Browser Flow	35
Figure 43 : Implementing MFA in Keycloak 4	36
Figure 44 : Implementing MFA in Keycloak 5 – adding OTP Form	36
Figure 45 : Implementing MFA in Keycloak 6 - OTP Form added	37
Figure 46 : Implementing MFA in Keycloak 7 - set duplicated flow to default	37

Figure 47 : Implementing MFA in Keycloak 8	37
Figure 48 : Implementing MFA in Keycloak 9 - changing OTP Policy settings	38
Figure 49 : Implementing MFA in Keycloak 10 - MFA implemented.....	38
Figure 50 : Implementing RBAC in Keycloak 1	39
Figure 51 : Implementing RBAC in Keycloak 2 - name the role	39
Figure 52 : Implementing RBAC in Keycloak 3 - assign roles to admin role	40
Figure 53 : Implementing RBAC in Keycloak 4 - assign admin role to afar	41
Figure 54 : Implementing RBAC in Keycloak 5 - assign admin role to afar	41
Figure 55 : Implementing RBAC in Keycloak 6 - creating a group	42
Figure 56 : Implementing RBAC in Keycloak 7 - joining a group	43
Figure 57 : Implementing RBAC in Keycloak 8 - joining a group	43
Figure 58 : APIcast installation	44
Figure 59 : APIcast container listing	45
Figure 60 : Link APIcast with Keycloak 1 - Client creation	45
Figure 61 : Link APIcast with Keycloak 2 - Client creation	46
Figure 62 : Link APIcast with Keycloak 3 - copy client secret.....	46
Figure 63 : Link APIcast with Keycloak 4 - oidc_config.json.....	47
Figure 64 : Link APIcast with Keycloak 5 - mount the OIDC configuration file.....	47
Figure 65 : Link APIcast with Keycloak 6 - get an Access Token from Keycloak	48
Figure 66 : Link APIcast with Keycloak 7 - use the Access Token to call APIcast	49

Summary

Identity and Access Management (IAM) is a framework of policies, technologies, and processes that ensures the right individuals and systems have appropriate access to digital resources at the right time.

IAM governs **who can access what, under what conditions, and for how long**. It enforces security policies and reduces the risk of unauthorized access, insider threats, and data breaches.

It is a critical component of cybersecurity that helps organizations manage user identities and control access to sensitive information.

This project aims to implement a **self-hosted IAM solution** leveraging open-source tools to provide secure authentication, authorization, and monitoring.

This project integrates multiple open-source tools to create a robust, self-hosted IAM solution that enhances security, access control, and monitoring.

By combining **identity management & authentication (Keycloak, OpenLDAP)**, **API security (APIcast)**, and **security monitoring (Wazuh)**, this architecture ensures a secure and scalable IAM framework suitable for various enterprise environments.

Introduction

IAM is a **critical security framework** that manages user identities, enforces access policies, and protects sensitive resources. By integrating IAM with **authentication, authorization, and monitoring tools**, organizations can strengthen security while maintaining operational efficiency.

1. Project scope

The objective of this project is to design and implement a scalable, **self-hosted IAM solution** that:

- Provides authentication and authorization services.
- Manages user identities and access policies.
- Integrates with various applications and services.
- Implements Zero Trust security principles.
- Monitors and detects security threats.

Among the benefits of implementing a self-hosted (on-premises) IAM solution, we mention:

- Offers **complete control** over security policies, configurations, and integrations. You can customize it extensively to meet specific business or compliance needs.
- Allows **full control over security** policies, making it easier to comply with **strict regulations** (e.g., GDPR, HIPAA, SOC 2).
- Allows **customized authentication and authorization** features.

2. Basic Concepts

a. IAM – Identity and Access Management

IAM is a framework of business processes, policies and technologies that facilitates the management of digital identities. With an IAM framework in place, IT security teams can control user access to critical information within their organizations.

i. IAM Deployment Models

IAM can be implemented in different environments:

- **On-Premises IAM:** Hosted within an organization's infrastructure (e.g., using **OpenLDAP**, **Keycloak**).
- **Cloud-Based IAM:** Hosted in the cloud (e.g., AWS IAM, Azure AD, Google Identity).
- **Hybrid IAM:** A mix of on-prem and cloud-based IAM systems.

j. Authentication Methods in IAM

Authentication verifies **who you are** before granting access. IAM supports multiple authentication methods:

- **Single-Factor Authentication (SFA):** Username & password.
- **Multi-Factor Authentication (MFA):** Requires additional verification (e.g., OTP, biometrics).
- **Passwordless Authentication:** Uses biometrics or security keys instead of passwords.
- **Federated Authentication:** Allows login via third-party identity providers (e.g., Google, Azure AD).

k. Authorization Methods in IAM

Authorization controls **what actions a user can perform** once authenticated:

- **Role-Based Access Control (RBAC):** Users get permissions based on roles (e.g., Admin, User, Guest).

- **Attribute-Based Access Control (ABAC):** Access is based on attributes like location, device, or department.
- **Policy-Based Access Control (PBAC):** Uses predefined policies for access management (e.g., Zero Trust security).

l. IAM Protocols & Standards

IAM solutions rely on industry standards to facilitate secure authentication and authorization:

- **LDAP (Lightweight Directory Access Protocol):** A directory service for storing identity information.
- **OAuth 2.0:** A framework for token-based authentication and authorization.
- **OpenID Connect (OIDC):** An identity layer on top of OAuth 2.0 for user authentication.
- **SAML (Security Assertion Markup Language):** Used for federated authentication and SSO.
- **Kerberos:** A network authentication protocol used in enterprise environments.

b. IdP – Identity Provider

An **Identity Provider (IdP)** is a **trusted system that creates, stores, and manages digital identities** for users and services. It is responsible for **authenticating users** and issuing identity information (such as tokens or credentials) to other applications or services requesting access.

Key Functions of an IdP :

- **User Authentication:** Verifies a user's identity using credentials (passwords, biometrics, MFA).
- **Identity Federation:** Allows users to log in with an external identity provider (e.g., Google, Microsoft Entra ID).
- **Single Sign-On (SSO):** Enables users to authenticate once and access multiple applications without logging in again.
- **Token Issuance:** Generates authentication and authorization tokens (e.g., SAML assertions, OAuth 2.0 access tokens, OpenID Connect ID tokens).

- **User Lifecycle Management:** Manages user accounts, password resets, role assignments, and access policies.

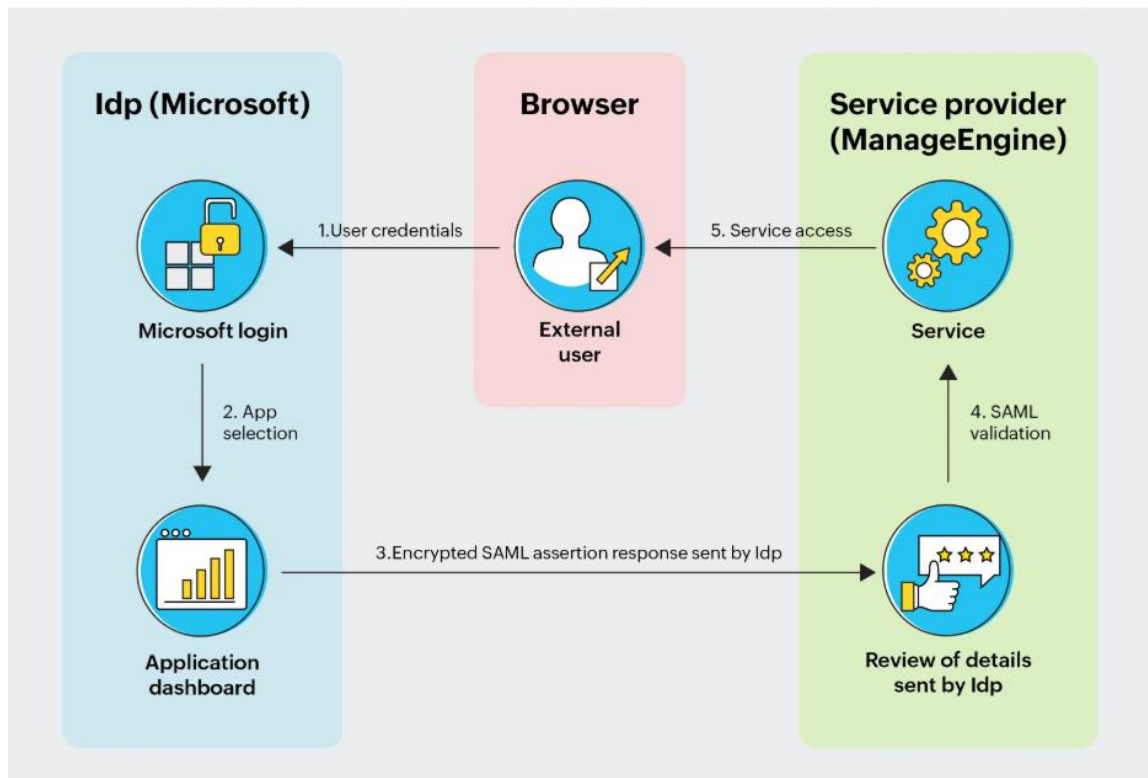


Figure 1: IdP workflow

c. SSO – Single Sign-On

SSO allows users to authenticate once and gain access to multiple applications without needing to re-enter credentials for each one. It enhances user convenience and security by reducing password fatigue and minimizing the risk of credential theft.

SSO delegates authentication to an **Identity Provider (IdP)** (e.g., Keycloak, Okta, Google, Microsoft Entra ID). After initial authentication, the IdP issues a **session/token** (OAuth 2.0 token, SAML assertion, or OpenID Connect ID token) to grant access to different applications without requiring another login.



Figure 2 : Single Sign-On workflow

3. Tools Utilized

a. Keycloak

Role: Identity Provider (IdP), MFA and Single Sign-On (SSO) solution.



Features:

- Supports OpenID Connect (OIDC) and SAML authentication.
- User federation through LDAP and other identity sources.
- Fine-grained access control with roles (RBAC) and groups.
- Multi-factor authentication (MFA) support.
- Integration with third-party applications via identity brokering.

b. OpenLDAP

Role: Centralized directory service for identity storage.



Features:

- Lightweight Directory Access Protocol (LDAP) support for identity storage and retrieval.
- Hierarchical organization of users, groups, and policies.
- Authentication backend for applications requiring LDAP integration.

c. APIcast

Role: acts as an **API gateway** that enforces authentication, authorization, and traffic control.

APIcast

Features:

- OAuth2 & OpenID Connect Integration – Supports token validation and secure API access.
- Rate Limiting & Traffic Control – Prevents API abuse with quotas, throttling, and burst limits.
- Reverse Proxy & API Gateway – Routes, balances, and secures API traffic.
- Logging & Monitoring – Tracks API requests for auditing and compliance.
- Multi-Tenant API Management – Manages multiple APIs with custom access rules.

d. Wazuh

Role: Security monitoring and threat detection.



Features:

- Intrusion Detection System (IDS) and log analysis.
- Real-time security monitoring of IAM services.
- Compliance auditing and reporting.

4. IAM Solution Architecture

a. User Authentication & Authorization

- Users access applications (web, mobile, APIs).
- **Keycloak** handles authentication (OAuth 2.0, OpenID Connect, SAML), implements MFA and RBAC.
- **Keycloak** verifies users via **OpenLDAP/FreeIPA** for user roles & credentials.
- **OpenLDAP** manages user directories.

b. API Security & Access Management

- **APIcast** ensures API rate limiting, token verification, and access control.

c. Security Monitoring & Compliance (Optional)

- **Wazuh** monitors IAM security, tracks unauthorized access attempts.
- **ELK Stack** collects IAM logs for real-time monitoring & forensic analysis.

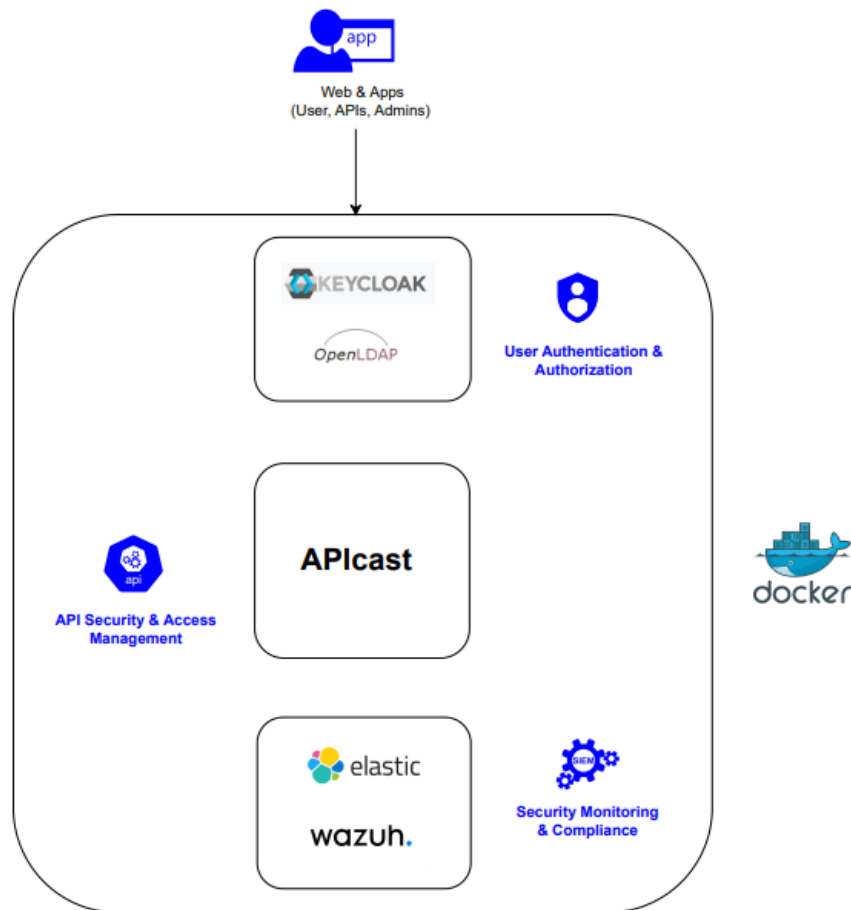


Figure 3 : IAM Solution Architecture

5. Setting Up Docker

a. What is Docker

Docker is an open platform for developing, shipping, and running applications.

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security let you run many containers simultaneously on a given host.

In this project, we will use Docker containers to run and set up the tools needed!

b. Setting up Docker

First of all, as a routine task, we will update our system before making any installation.

Command: ***sudo apt update***

```
(kali㉿kali)-[~]
$ sudo apt update
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [20.7 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [49.4 MB]
Get:4 http://kali.download/kali kali-rolling/non-free amd64 Packages [195 kB]
Get:5 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [880 kB]
Get:6 http://kali.download/kali kali-rolling/contrib amd64 Packages [115 kB]
Get:7 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [268 kB]
Fetched 71.6 MB in 1min 7s (1,075 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1795 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Figure 4 : Updating the system

Now that the system well updated and ready, we will install the Docker package.

Command: ***sudo apt install -y docker.io***

```
(kali㉿kali)-[~]
$ sudo apt install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-noto-color-emoji kali-wallpapers-2024 libabsl20230802 libaio1 libcbor0.1
  libnsl-dev libplist-2.0-4 libpwquality-common libtirpc-dev libucl1 m4 openssh-
  python3-pyinstaller-hooks-contrib python3-pypdf2 python3-requests-toolbelt pyt
```

Figure 5 : Installing Docker

After successfully installing Docker, we should make sure that is enabled.

Command: ***sudo systemctl enable docker --now***

```
(kali㉿kali)-[~]
$ sudo systemctl enable docker --now
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

Figure 6 : Enabling Docker

Now, we will just make sure that Docker is all set up.

Command: ***docker version***

```
(kali㉿kali)-[~]  
$ docker version  
Client:  
Version:           26.1.5+dfsg1  
API version:       1.45  
Go version:        go1.24.1  
Git commit:        a72d7cd  
Built:             Fri Mar 14 04:30:19 2025  
OS/Arch:           linux/amd64  
Context:           default  
permission denied while trying to connect to the Docker daemon socket at unix  
dial unix /var/run/docker.sock: connect: permission denied
```

Figure 7: Docker version

Now, we will be able to start installing and setting up the tools needed for the realization of this project (**Keycloak**, **OpenLDAP** and **APIcast**)

Chapter 1: Implementing the User Authentication & Authorization part

1. Introduction

This part of the project is the most important thing and plays a very important role on implementing that basis of our robust self-hosted IAM solution, by implementing:

- **User Authentication system:** is the method of verifying the identity of a consumer or system to ensure they're who they claim to be. It includes implementing;
 - Multiple Factor Authentication (MFA)
 - Federated Authentication
- **User Authorization system:** is the method of figuring out and granting permissions to a demonstrated user or system, specifying what assets they can access and what actions they're allowed to carry out. It includes implementing:
 - Role-Based Access Control (RBAC),
 - Attribute-Based Access Control (ABAC) or
 - Policy-Based Access Control (PBAC)

Tools that will be used in this part:

- **Keycloak:** as the main IAM and Identity Provider platform, in addition to that it will provide:
 - Multiple Factor Authentication (MFA)
 - Federated Authentication (By linking it with **OpenLDAP**)
 - Role-Based Access Control (RBAC)
- **OpenLDAP:** as the manager of user directories

2. Setting up OpenLDAP

OpenLDAP Software is an open-source implementation of the Lightweight Directory Access Protocol.

a. Installing OpenLDAP

```
(kali@kali)-[~]
$ sudo docker run -d --name openldap --restart always \
  -p 389:389 -p 636:636 \
  -e LDAP_ORGANISATION="MyCompany" \
  -e LDAP_DOMAIN="mycompany.com" \
  -e LDAP_ADMIN_PASSWORD="admin25" \
  osixia/openldap

[sudo] password for kali:
Unable to find image 'osixia/openldap:latest' locally
latest: Pulling from osixia/openldap

45b42c59be33: Pull complete
ae7fb8f59730: Pull complete
55443d9da5d5: Pull complete
edbb56ba2f49: Pull complete
8690d28b09a7: Pull complete
7f3b6edb2b51: Pull complete
176dedf70b8e: Pull complete
036fb52cada7: Pull complete
94404c4ec0d9: Pull complete
Digest: sha256:3f68751292b43564a2586fc29fb7337573e2dad692b92d4e78e49ad5c22e567b
Status: Downloaded newer image for osixia/openldap:latest
7f3f406efc473a560598712b3f0cc05cdc6dfb9975231543967d30d590a065fb
```

Container created !

Figure 8 : Installing OpenLDAP

Explanation:

- **LDAP_ORGANISATION:** The organization name.
- **LDAP_DOMAIN:** The LDAP domain (used in Distinguished Names).
- **LDAP_ADMIN_PASSWORD:** Password for the **admin user**
- **389:** Default LDAP port, **636:** Secure LDAP (LDAPS).

b. Installing OpenLDAP Admin GUI

This step is optional, but it will facilitate the process of creating and managing users and groups!

```
(kali㉿kali)-[~]
└─$ sudo docker run -d --name phpldapadmin --restart always \
  -p 8081:80 --link openldap:ldap-host \
  -e PHPLDAPADMIN_LDAP_HOSTS=ldap-host \
  -e PHPLDAPADMIN_HTTPS=false \
  osixia/phpldapadmin

Unable to find image 'osixia/phpldapadmin:latest' locally
latest: Pulling from osixia/phpldapadmin
1ab2bdf9e9778: Pull complete
0abcaf321aa9: Pull complete
6d688c3d4e02: Pull complete
454331b99b9a: Pull complete
5cada7c8cb4e: Pull complete
1c9252038144: Pull complete
5b1bb72ef8f6: Pull complete
b7637df040ab: Pull complete
f14c298e98cf: Pull complete
Digest: sha256:9831569a2f3d1d764aabcb5abe6e463771b9595f1565fe3007fe77c4c3979043
Status: Downloaded newer image for osixia/phpldapadmin:latest
58a30e950bc41784f555905fc8d4e17a2c982e078d74321cf0e61a6bb764d910
```

Figure 9 : Installing OpenLDAP Admin GUI

We will need to start **phpldapadmin** container so that we can access the GUI interface.

```
(kali㉿kali)-[~]
└─$ sudo docker start phpldapadmin
phpldapadmin
```

Figure 10 : Starting phpldapadmin

Now we have 02 containers that should be in UP state, both for the functioning of **OpenLDAP**:

```
(kali㉿kali)-[~]
└─$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
58a30e950bc4	osixia/phpldapadmin	"/container/tool/run"	17 minutes ago
Up 17 minutes	443/tcp, 0.0.0.0:8081→80/tcp, :::8081→80/tcp	phpldapadmin	
7f3f406efc47	osixia/openldap	"/container/tool/run"	20 minutes ago
Up 19 minutes	0.0.0.0:389→389/tcp, :::389→389/tcp, 0.0.0.0:636→636/tcp, :::636→636/tcp	openldap	

Figure 11 : Listing OpenLDAP active containers

Finally, we can access the GUI interface at: <http://localhost:8081>

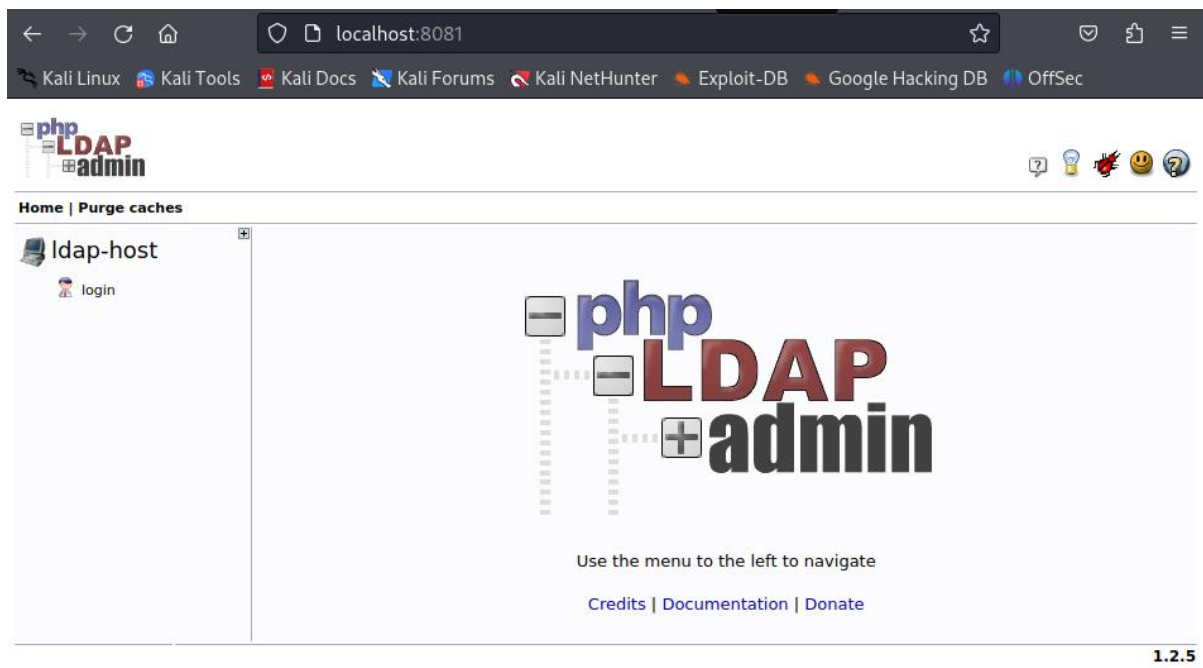


Figure 12 : OpenLDAP admin GUI interface

We should login using the following credentials:

- **Login DN:** cn=admin,dc=mycompany,dc=com
- **Password:** adminpassword (set at the installation time)

After successfully logged in, we get the following interface:

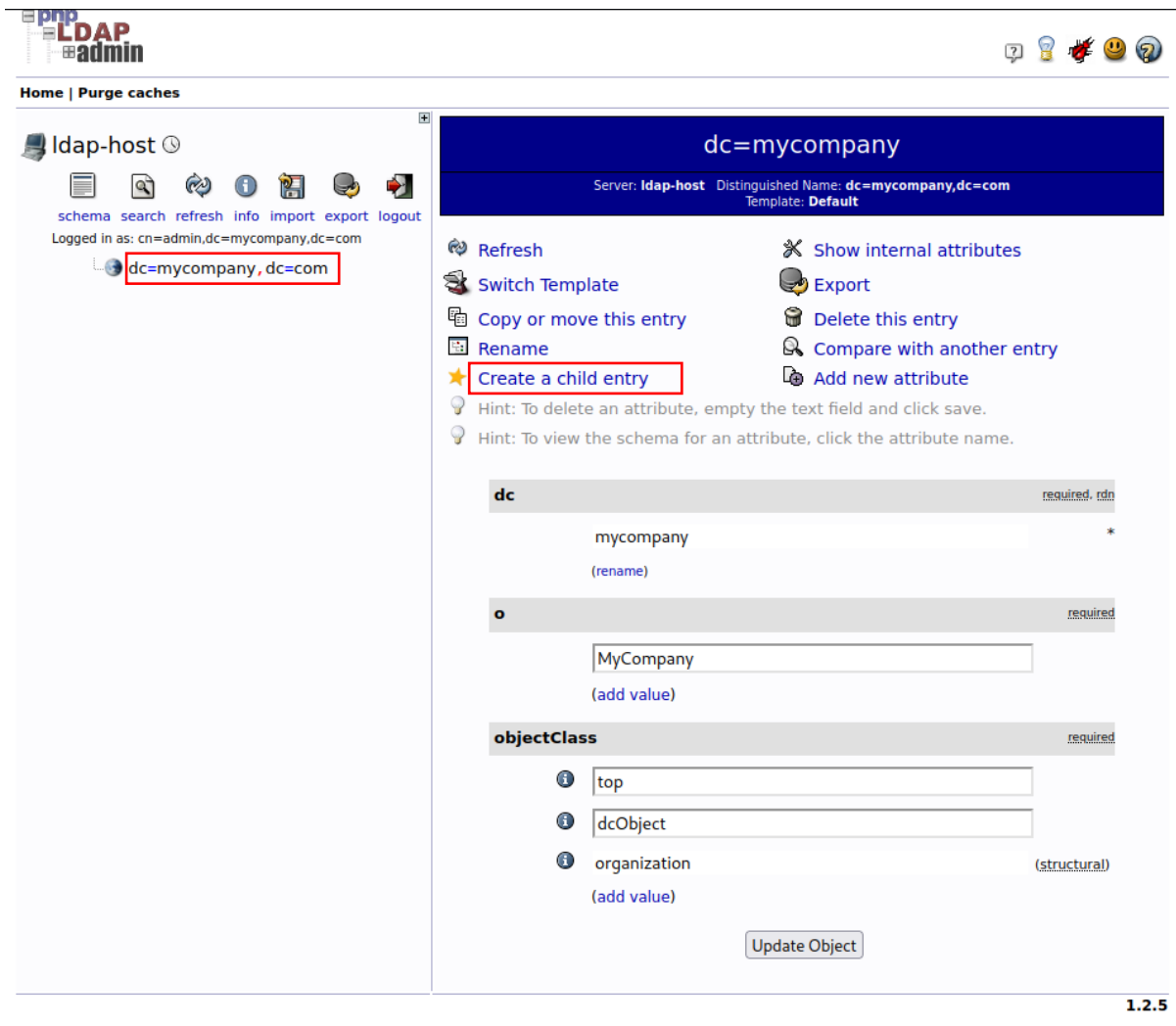


Figure 13 : Logged in OpenLDAP Admin GUI

And now we are ready to add and manage users and groups!

c. Adding users and groups

First of all, we have to create an **organisational unit** to hold the users that we will add.

Select our dc in this case is **dc=mycompany**, then click on “**create a child entry**”, we will get this interface in return:

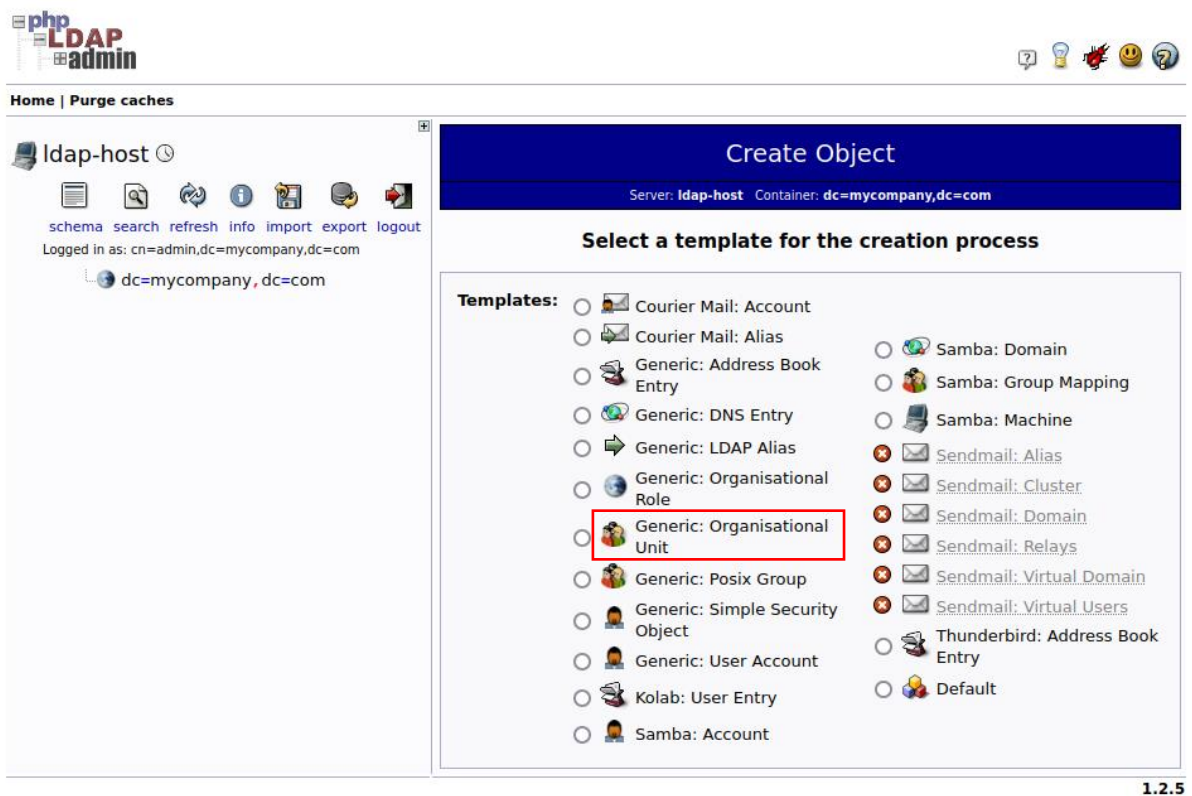


Figure 14 : Adding user organizational unit step 1

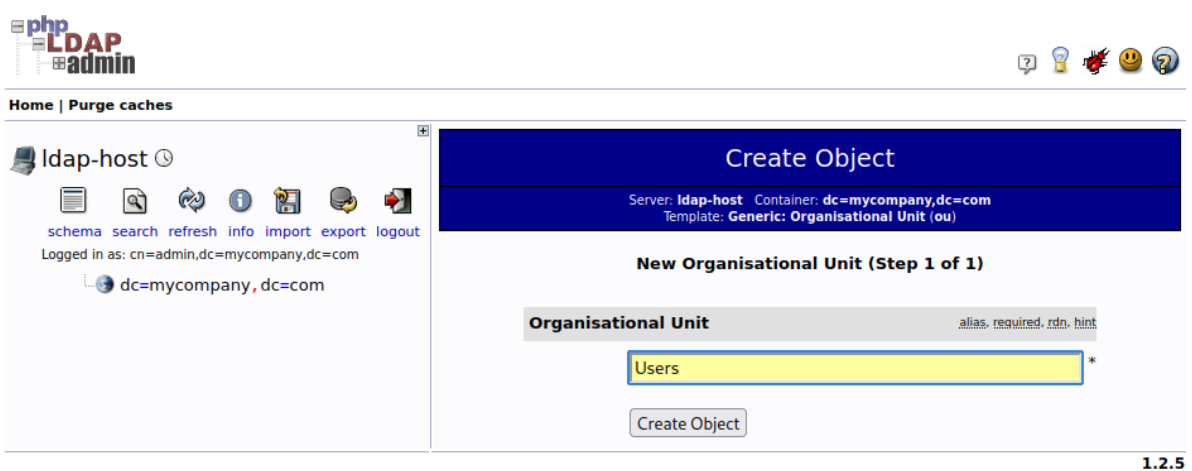


Figure 15 : Adding user organizational unit step 2

Before adding users, it is recommended to create a group first, since its ID will be needed during the user creation process.

So, we will come back to our **dc=mycompany**, and again select “**create a child entry**”, this time we will choose the “**Generic: Posix Group**” option.

Now we can enter the group Name as well as its GID as the following:

Create Object

Server: **ldap-host** Container: **dc=mycompany,dc=com**
Template: **Generic: Posix Group (posixGroup)**

New Posix Group (Step 1 of 1)

Group
alias, required, rdn

GID Number
alias, required, hint, re

Figure 16 : Adding a group step 1

Create LDAP Entry

Server: **ldap-host** Container: **dc=mycompany,dc=com**

Do you want to create this entry?

Attribute	New Value	Skip
cn=developers,dc=mycompany,dc=com		
Group	developers	<input type="checkbox"/>
GID Number	500	<input type="checkbox"/>
objectClass	posixGroup	<input type="checkbox"/>

1.2.5

Figure 17 : Adding a group step 2

After hitting **“Commit”**, the group will be successfully added !

Now after the unit and the groups have been created, we will select **ou=Users**, then click on **“create a child entry”** as shown in the following images:

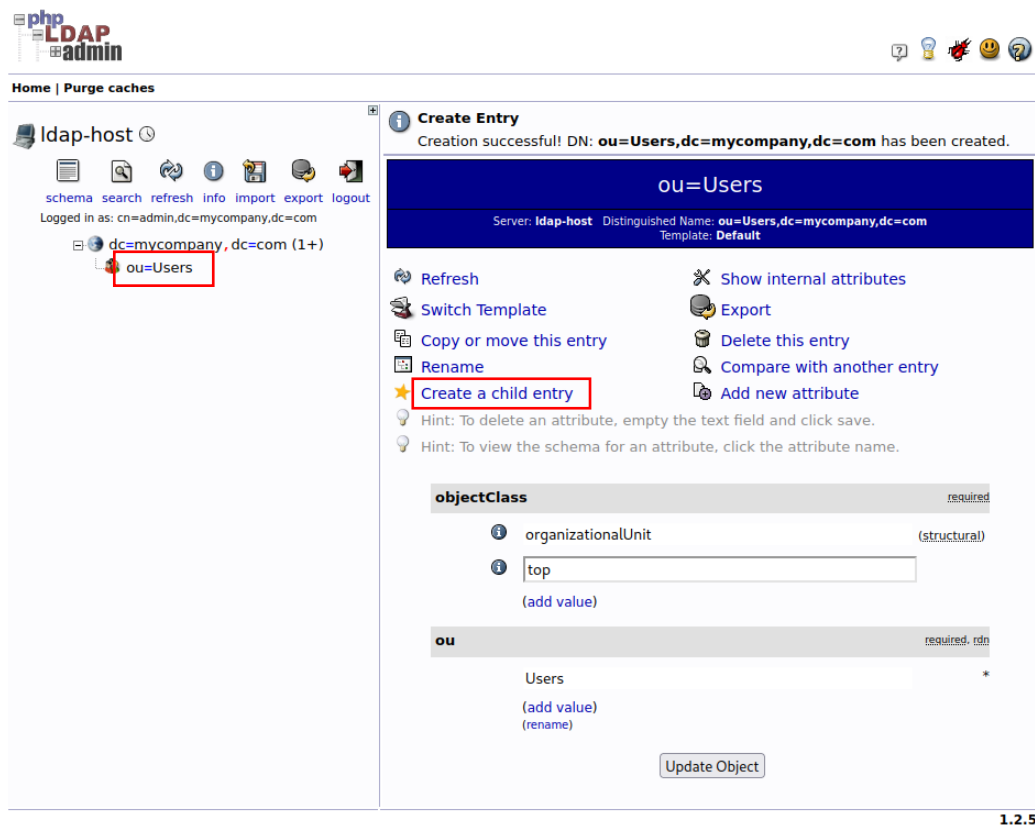


Figure 18 : Adding a user step 1

After we will choose the “Generic: User account” option

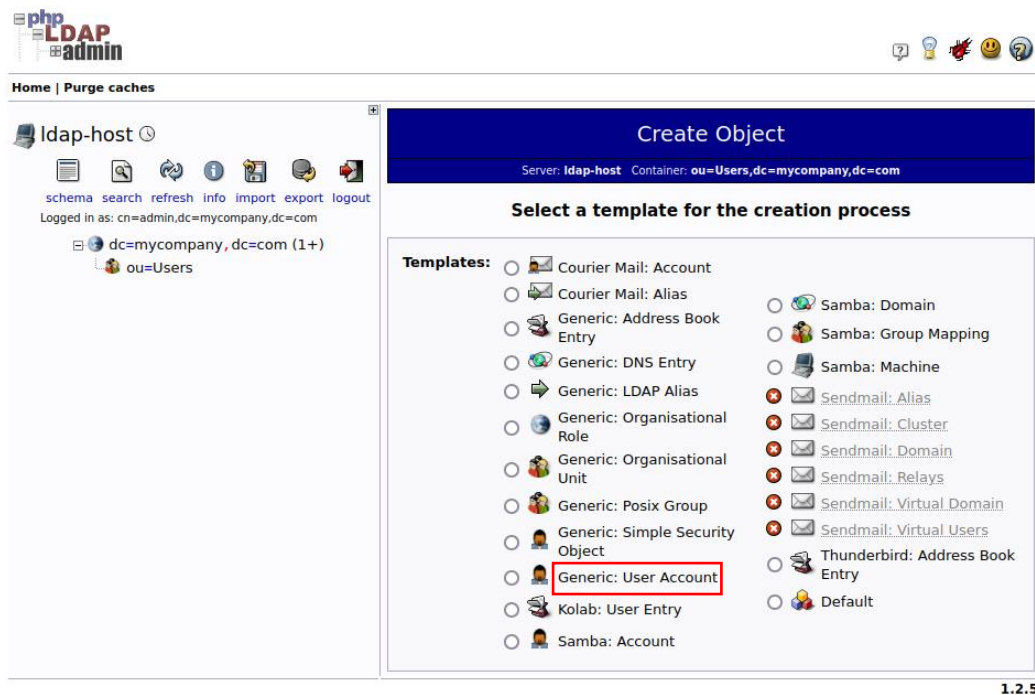


Figure 19 : Adding a user step 2

Now, all what we have to do is entering the user's information.

New User Account (Step 1 of 1)

First name

alias

John

Last name

alias, required

Doe

Common Name

alias, required, rdn

John.doe

User ID

alias, required

jdoe

Password

alias, hint

md5

(confirm)

Check password...

UID Number

alias, required, hint, no

1000

GID Number

alias, required, hint

developers

Home directory

alias, required

/home/users/jdoe

Login shell

alias

Bash

Create LDAP Entry

Server: ldap-host Container: ou=Users,dc=mycompany,dc=com

Do you want to create this entry?

Attribute	New Value	Skip
cn=John.doe,ou=Users,dc=mycompany,dc=com		
First name	John	<input type="checkbox"/>
Last name	Doe	<input type="checkbox"/>
Common Name	John.doe	<input type="checkbox"/>
User ID	jdoe	<input type="checkbox"/>
Password	*****	<input type="checkbox"/>
UID Number	1000	<input type="checkbox"/>
GID Number	500	<input type="checkbox"/>
Home directory	/home/users/jdoe	<input type="checkbox"/>
Login shell	/bin/bash	<input type="checkbox"/>
objectClass	inetOrgPerson posixAccount	<input type="checkbox"/>

Commit

Cancel

Figure 20 : Entering john doe user's information

Now for just make sure that everything went well, we will test the existence of the added user “**John Doe**”, for that we have 02 methods

By the GUI interface:



Figure 21 : Testing user existence using GUI interface

By the CLI:

```
(kali@kali)~$ ldapsearch -x -D "cn=admin,dc=mycompany,dc=com" -w admin25 -b "ou=Users,dc=mycompany,dc=com"
# extended LDIF
#
# LDAPv3
# base <ou=Users,dc=mycompany,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# Users, mycompany.com
dn: ou=Users,dc=mycompany,dc=com
ou: Users
objectClass: organizationalUnit
objectClass: top

# John.doe, Users, mycompany.com
dn: cn=John.doe,ou=Users,dc=mycompany,dc=com
givenName: John
sn: Doe
cn: John.doe
uid: jdoe
userPassword:: e01ENX1aWG5wYjNhNm9BZUhvb1pUaDJ4aEp3PT0=
uidNumber: 1000
gidNumber: 500
homeDirectory: /home/users/jdoe
loginShell: /bin/bash
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
```

Figure 22 : Testing user existence using CLI

Now, we can follow the same process to add more users and groups!

4. Setting up Keycloak

a. Installing Keycloak

```
(kali@kali)-[~]
$ sudo docker run -d --name keycloak --restart always \
  -p 8080:8080 \
  -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin \
  quay.io/keycloak/keycloak:latest start-dev

[sudo] password for kali:
Unable to find image 'quay.io/keycloak/keycloak:latest' locally
latest: Pulling from keycloak/keycloak
8a24962d3b26: Pull complete
7587ad96dcd2: Pull complete
300befaa4fd6: Pull complete
14ee67ce9949: Pull complete
581bd8557f4c: Pull complete
Digest: sha256:044a457e04987e1fff756be3d2fa325a4ef420fa356b7034ecc9f1b693c32761
Status: Downloaded newer image for quay.io/keycloak/keycloak:latest
7d867fff60d856b86e9488305f2b46703782c3ae2ef4cb07ab518fdc4ec2ff20
```

Figure 23 : Keycloak Installation

Now we can check if the **Keycloak** container is created and in an UP state:

```
(kali@kali)-[~]
$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS
NAMES
7d867fff60d8   quay.io/keycloak/keycloak:latest    "/opt/keycloak/bin/k..." 6 minutes ago
Up 6 minutes   8443/tcp, 0.0.0.0:8080→8080/tcp, :::8080→8080/tcp, 9000/tcp
keycloak
58a30e950bc4   osixia/phpldapadmin                "/container/tool/run"     About an hour ago
Up 41 minutes  443/tcp, 0.0.0.0:8081→80/tcp, :::8081→80/tcp
phpldapadmin
7f3f406efc47   osixia/openldap                    "/container/tool/run"     About an hour ago
Up 45 minutes  0.0.0.0:389→389/tcp, :::389→389/tcp, 0.0.0.0:636→636/tcp, :::636→636/
tcp openldap
```

Figure 24 : Keycloak Container listing

Now we can access the **Keycloak** Admin Console at:

<http://localhost:8080/admin>

b. Configure Keycloak to Use OpenLDAP as User Store

Now we will access the **Keycloak** admin console and log in as the admin using the following credentials:

- **Username:** admin
- **Password:** admin

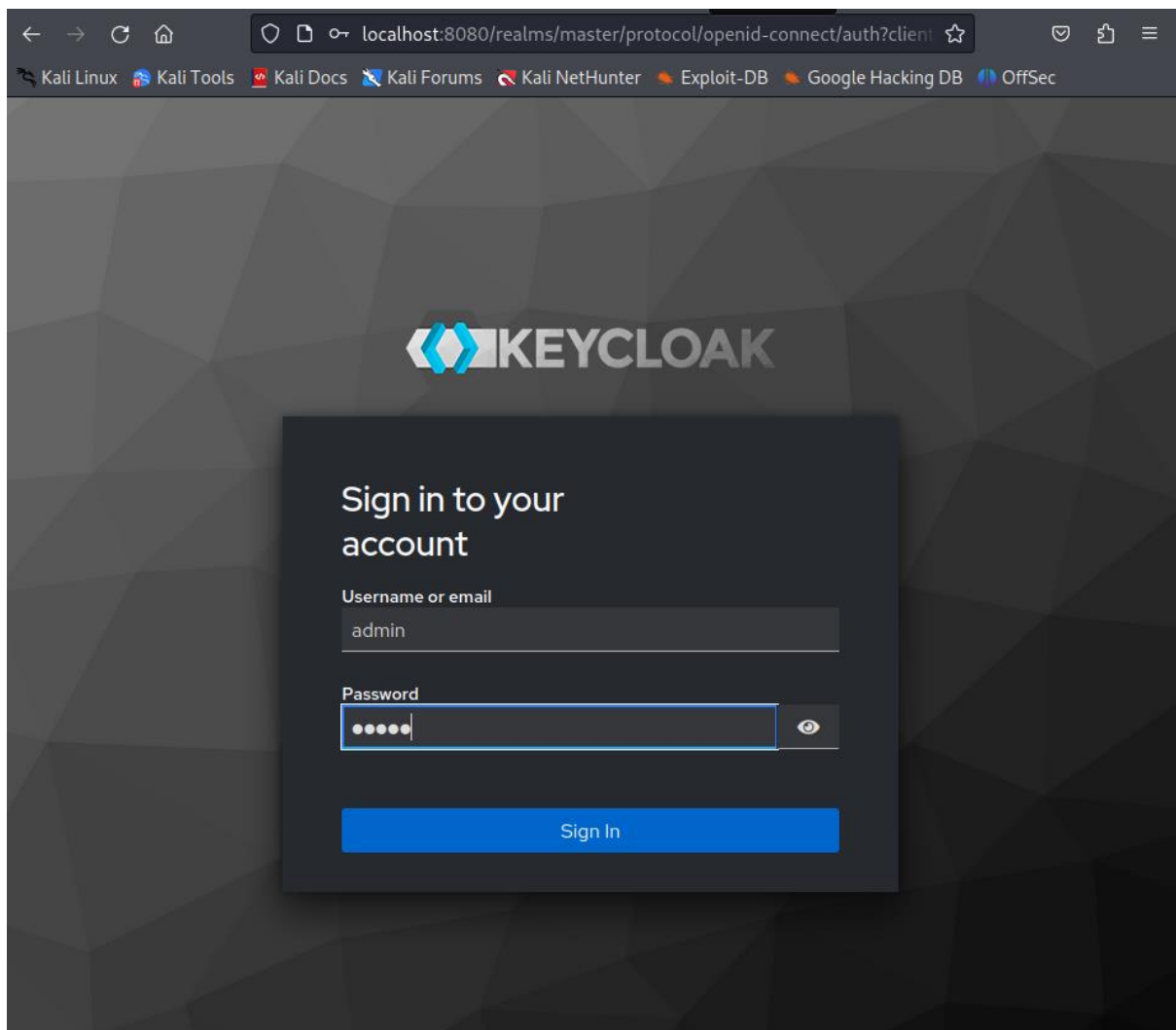


Figure 25 : Keycloak Admin Console

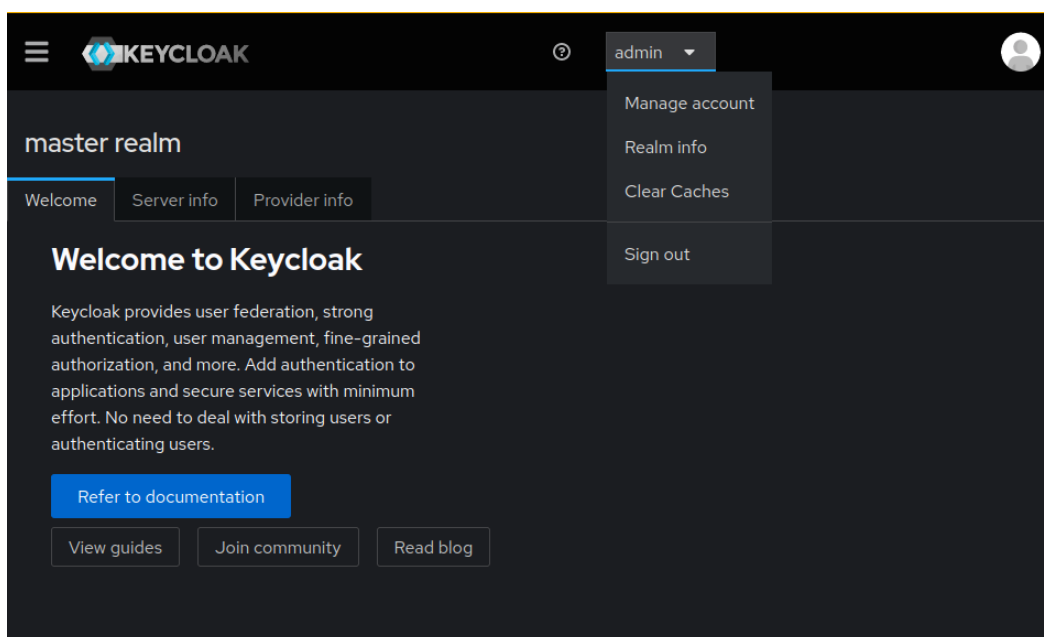


Figure 26 : Keycloak admin console after logging in

Now we should create a realm, and it's recommended to name it the same name as our dc in **OpenLDAP**.

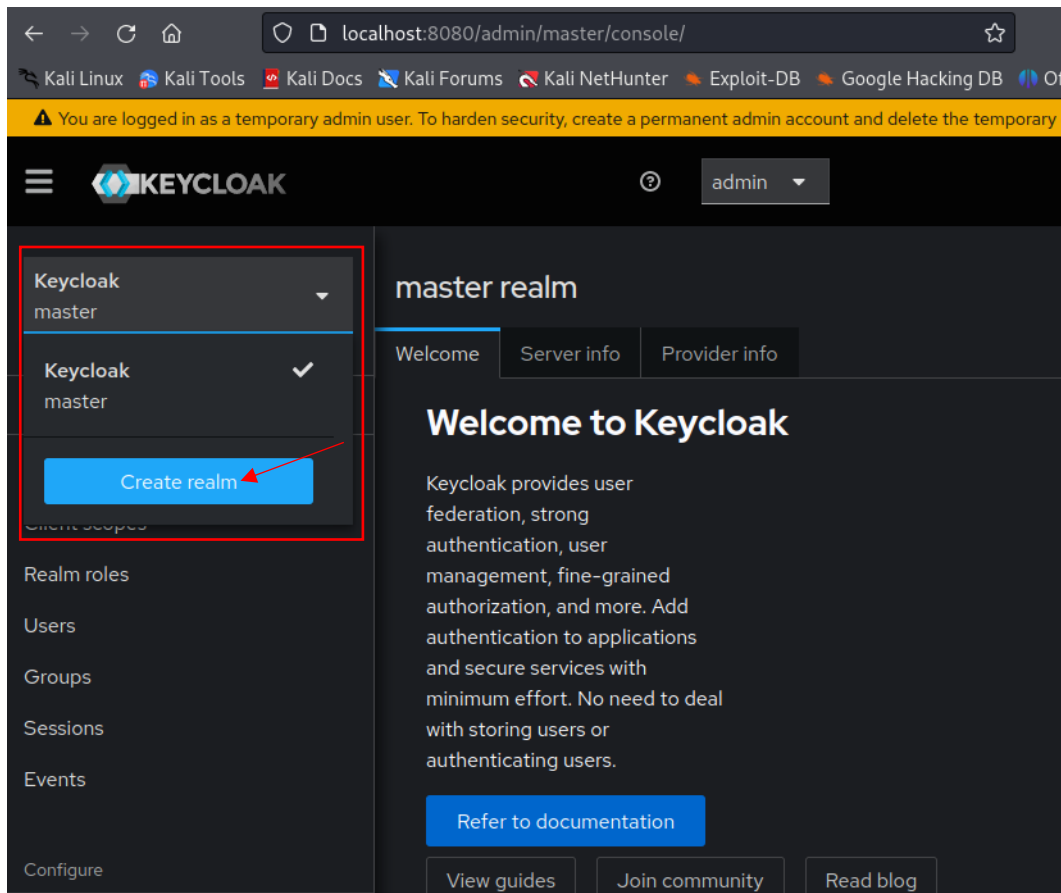


Figure 27 : creating a realm in Keycloak

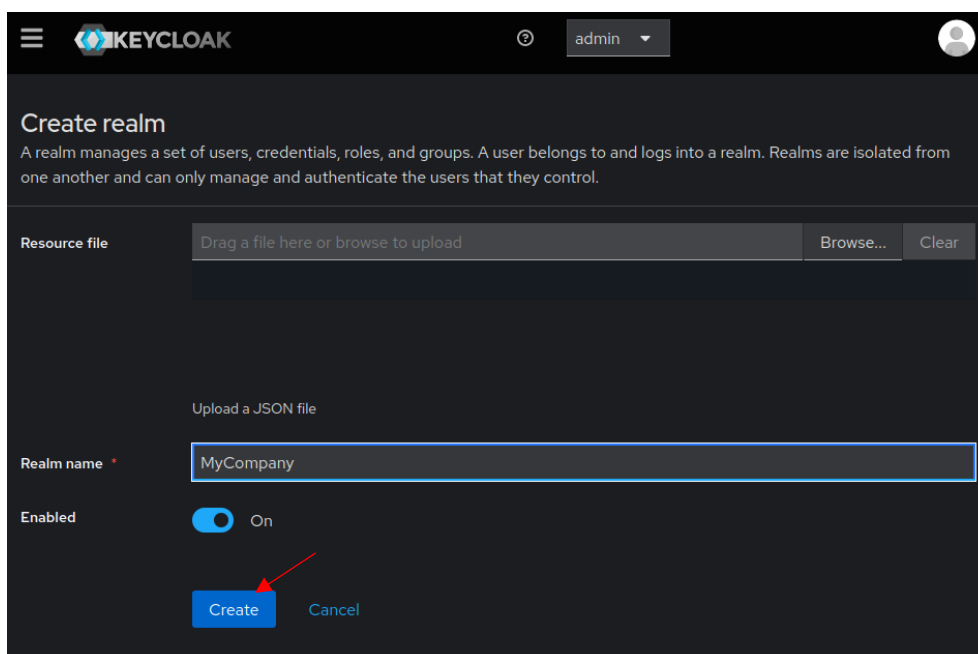


Figure 28 : creating a realm in Keycloak 2

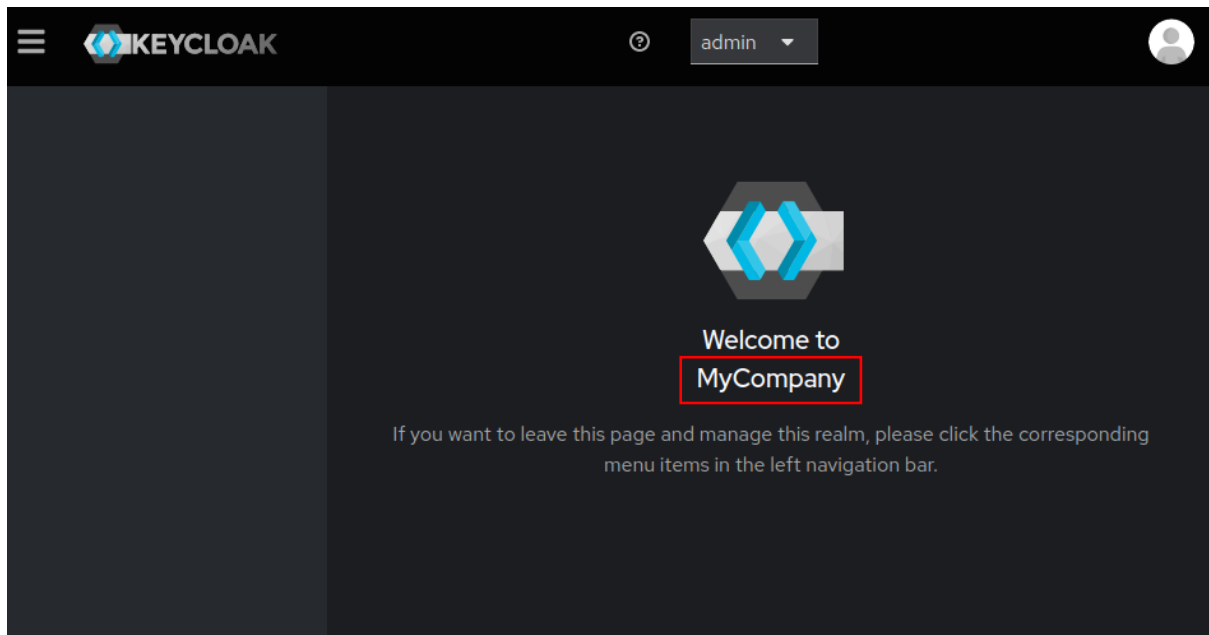


Figure 29 : mycompany realm created in Keycloak

Now after creating the realm for our **dc=mycomapny**, we will configure the realm to use our dc in **OpenLDAP** to store users.

To do so, we have to go to “**User Federation**”.

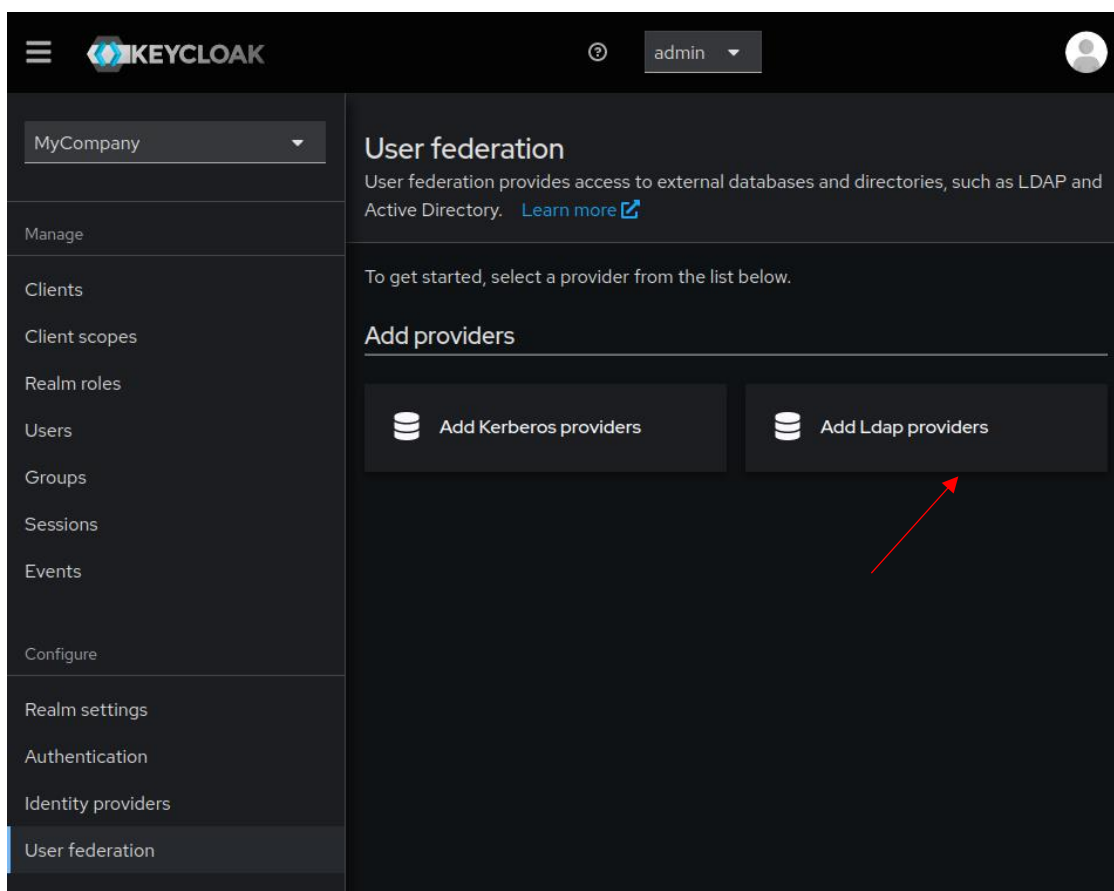


Figure 30 : Connect OpenLDAP with Keycloak step 1

We will find ourselves in front of this interface to enter information about our **OpenLDAP** server.

The screenshot shows the 'Add LDAP provider' configuration page in Keycloak. The 'Connection URL' field is highlighted with a red box and contains the value 'ldap://172.17.0.2:389'. Other fields include 'UI display name' (ldap), 'Vendor' (Other), 'Enable StartTLS' (Off), 'Use Truststore SPI' (Always), and 'Connection pooling' (On). A sidebar on the right lists sections: General options, Connection and authentication settings, LDAP searching and updating, Synchronization settings, Kerberos integration, Cache settings, and Advanced settings.

Figure 31 : Connect OpenLDAP with Keycloak step 2

To get the “**Connection URL**”, we need first to run the following command:

```
(kali㉿kali)-[~]
$ sudo docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' openldap
[sudo] password for kali:
172.17.0.2
```

Figure 32 : Get OpenLDAP IP address

After hitting “**save**” button, **OpenLDAP** and **Keycloak** should be connected, to test that we have to find the user created earlier in **OpenLDAP** “**John Doe**”, existing in the “**user**” section in **Keycloak**.

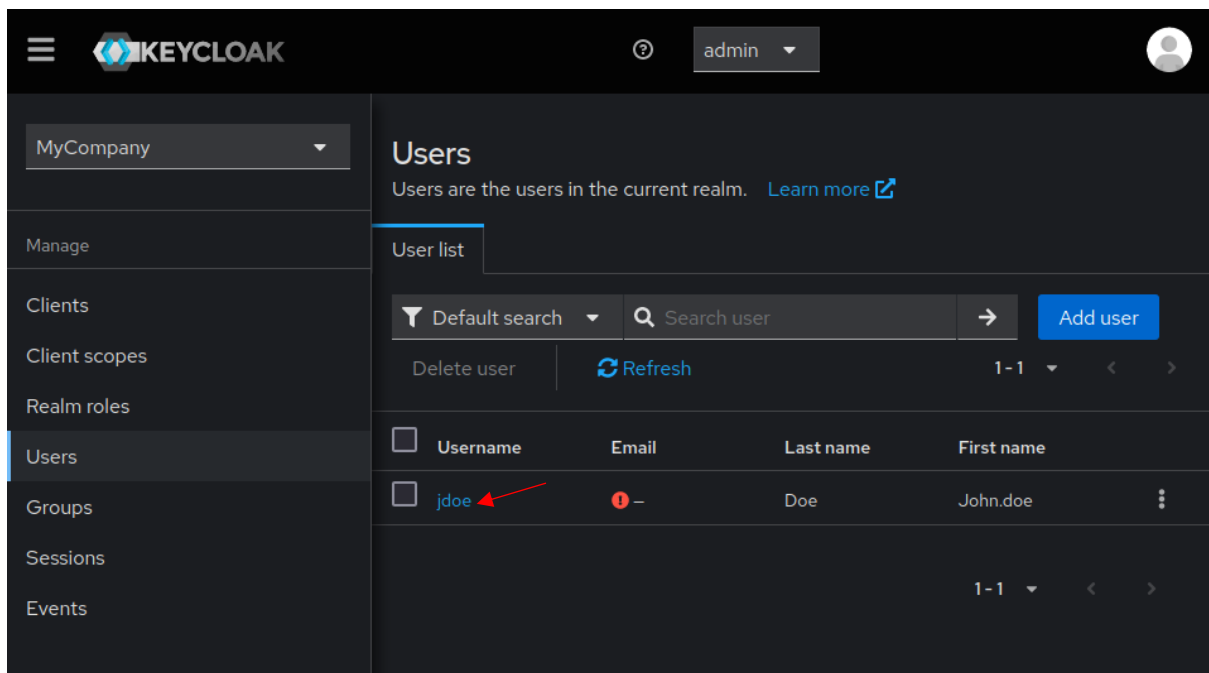


Figure 33 : Testing the existence of user jdoe in Keycloak

We can also verify the connection, by logging in as the user **John Doe** using the **Keycloak** accounts interface at: <http://localhost:8080/realms/MyCompany/account>

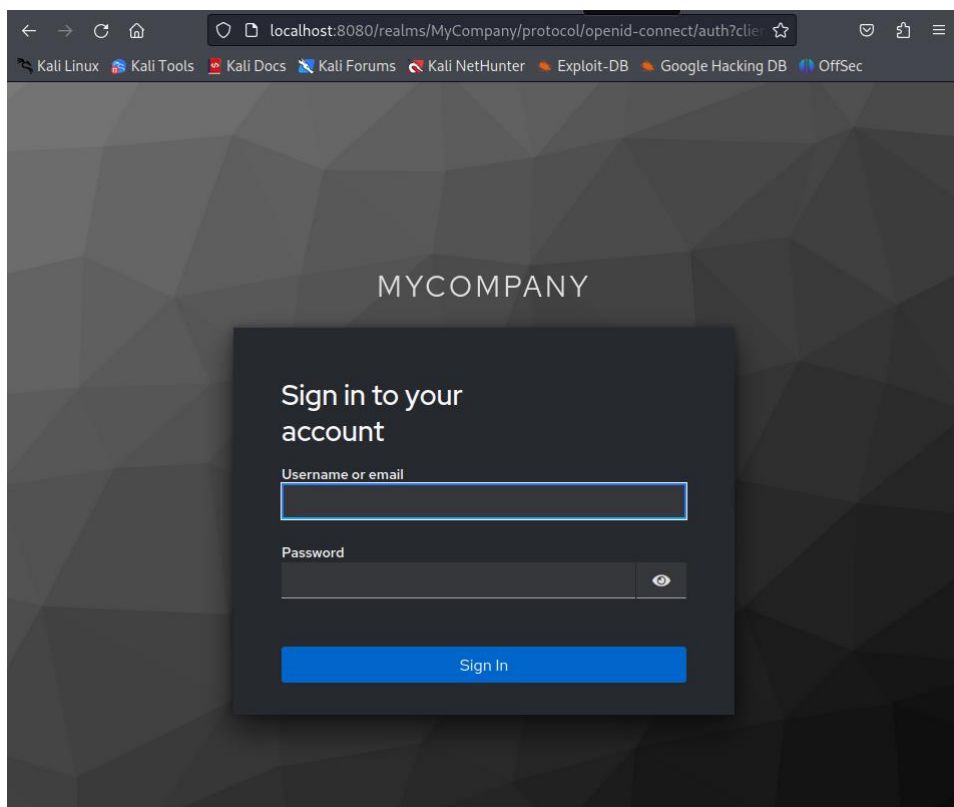


Figure 34 : User account log in interface

Now we have to enter the credentials of the user **John Doe**, we can find the credentials at **OpenLDAP** GUI interface.

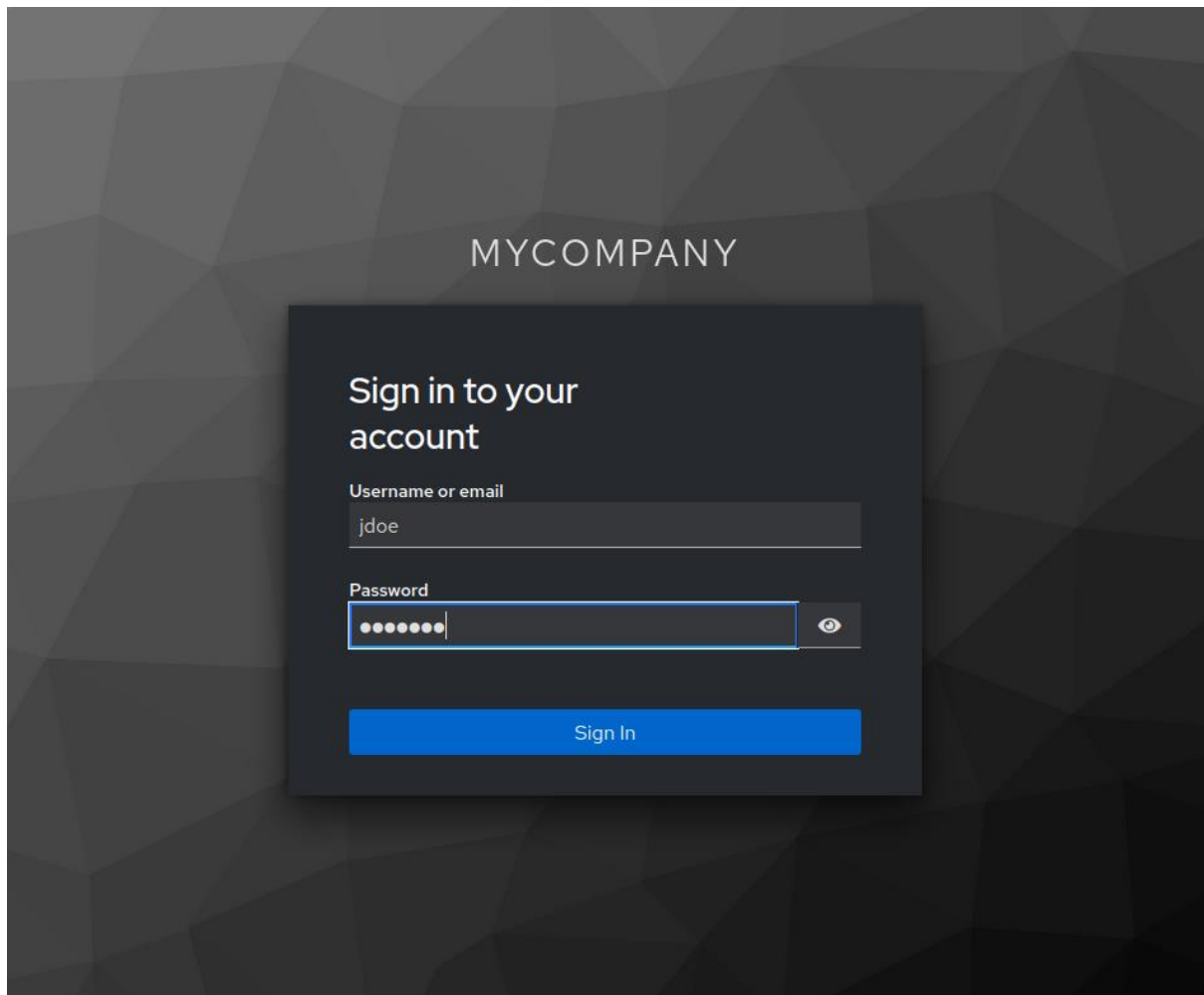


Figure 35 : Logging in as jdoe in Keycloak

If the connection between **OpenLDAP** and **Keycloak** has been successfully established, we should be able to log in as the **John Doe** user that we created earlier in **OpenLDAP**.

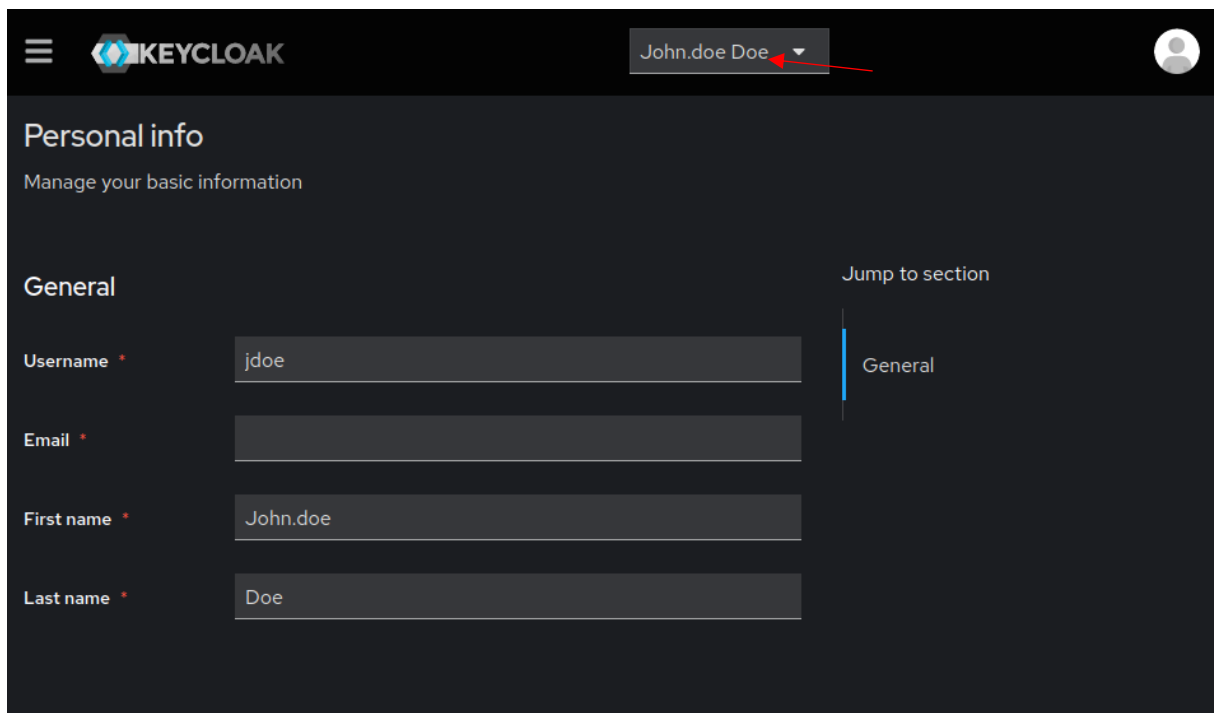


Figure 36 : Logged in as the user John Doe

As we can see, we successfully logged in as the “**John Doe**” user!

Following the same processes mentioned in this section, we added 08 users and 04 groups in **OpenLDAP** which can be accessed also by **Keycloak**.

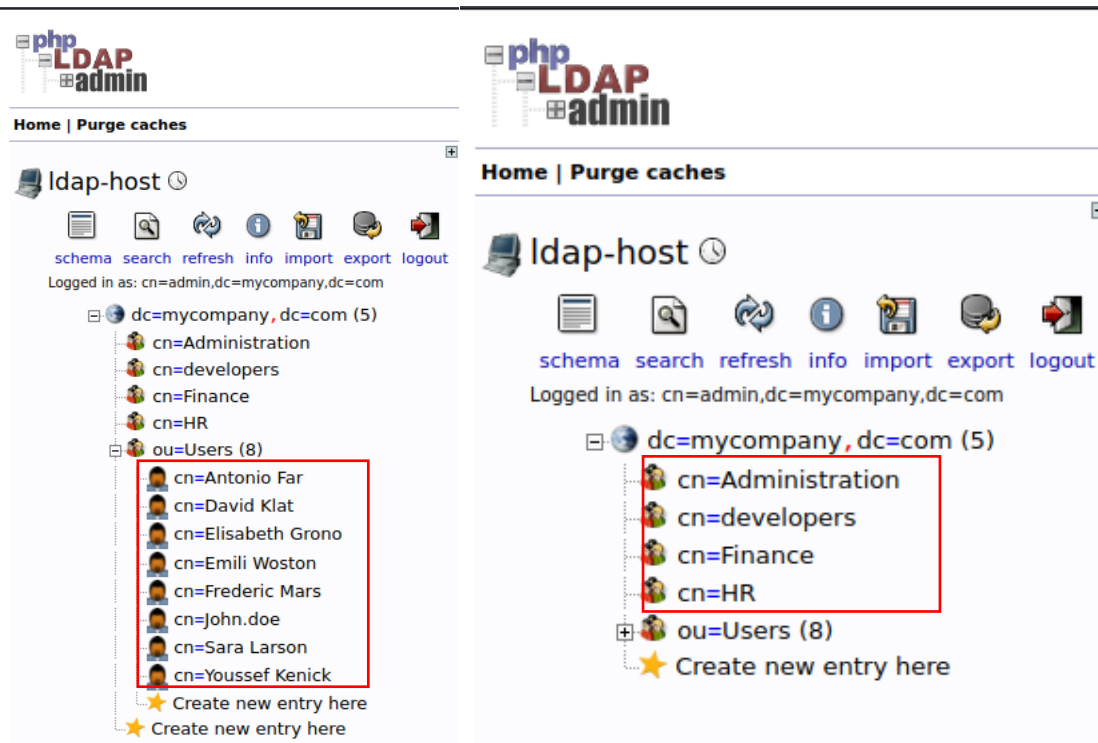


Figure 37 : Users and groups added in OpenLDAP

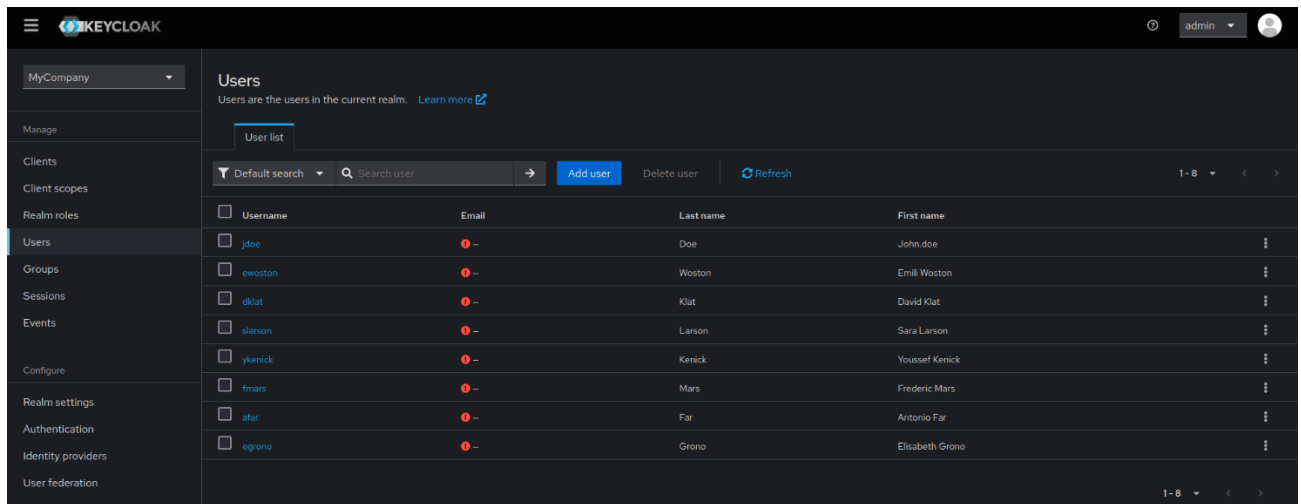


Figure 38 : Users accessed by Keycloak

5. Implementing Authentication and Authorization mechanisms in Keycloak

a. Implementing Single-Sign On (SSO)

SSO allows users to authenticate once and gain access to multiple applications without needing to re-enter credentials for each one. It enhances user convenience and security by reducing password fatigue and minimizing the risk of credential theft.

Obviously, **Keycloak** provide the SSO mechanism.

By selecting the realm in which we want to implement this mechanism, then go to “**session**”

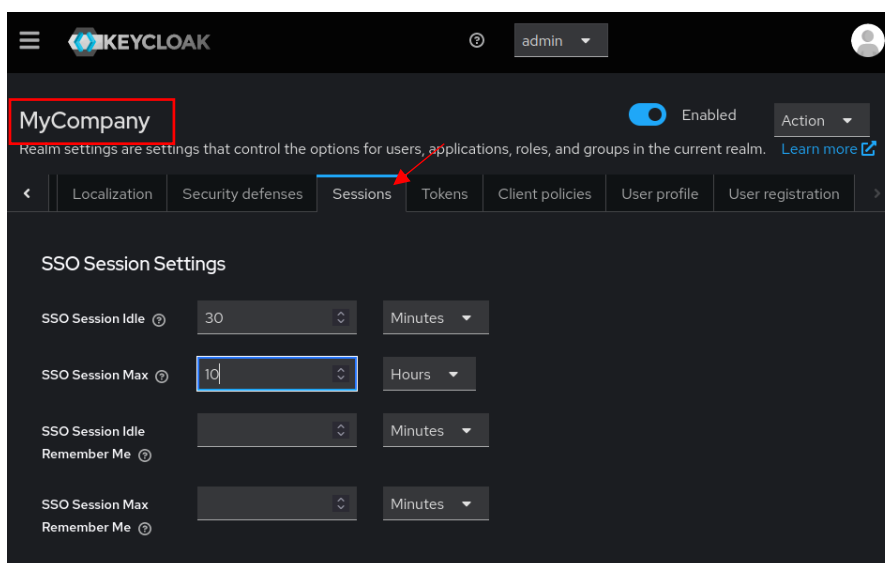


Figure 39 : Implementing SSO in Keycloak

b. Implementing Multiple Factor Authentication (MFA)

MFA is an authentication method that **requires the user to provide two or more verification factors** to gain access to a resource such as an application, online account, or a VPN.

MFA is a core component of a **strong identity and access management (IAM) policy**. Rather than just asking for a username and password, MFA requires one or more additional verification factors, which decreases the likelihood of a successful cyber-attack.

Keycloak offers also the MFA feature that can be easily implemented, by following those steps:

First of all, we gotta go to the **“Authentication”** section in our realm (MyCompany), and select the **“browser”** flow.

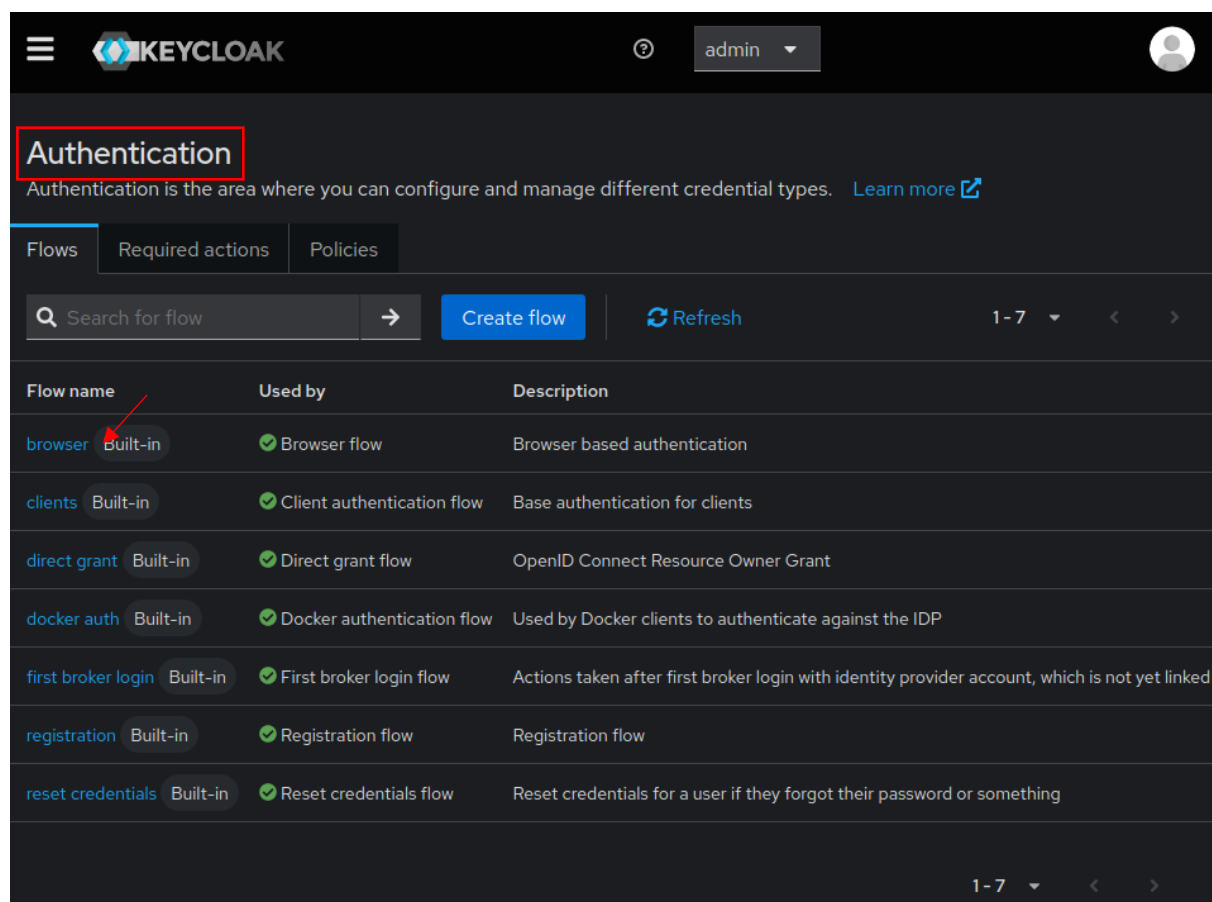


Figure 40 : Implementing MFA in Keycloak 1

After accessing the **“browser”** flow, we go to **“actions”** and select **“duplicate”** so that we can make changes that we want.

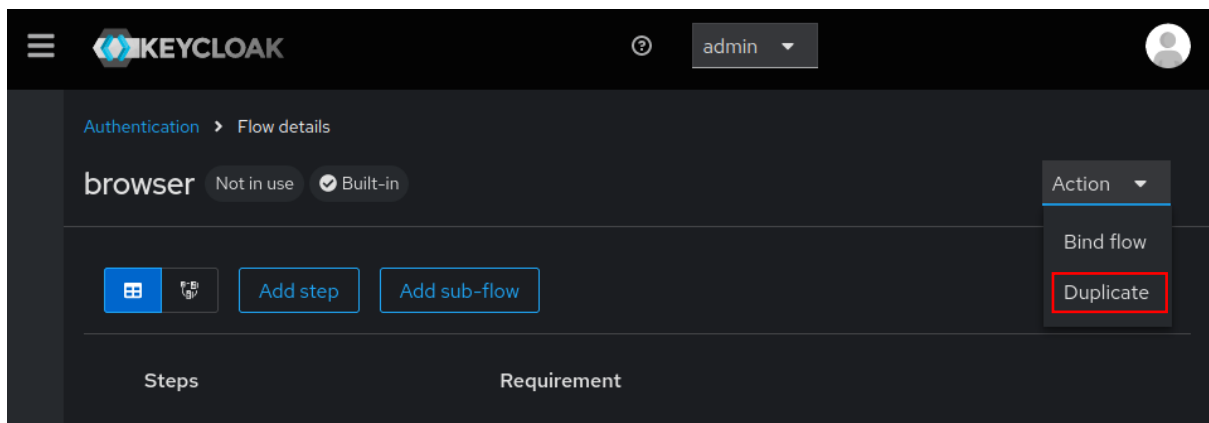


Figure 41 : Implementing MFA in Keycloak 2

Name it as you want and click on **“Duplicate”**.

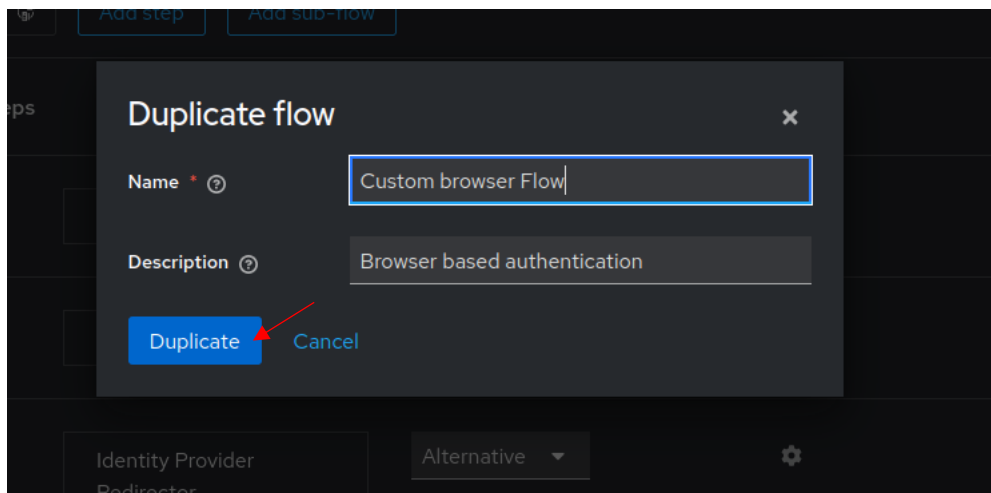


Figure 42 : Implementing MFA in Keycloak 3 – duplicating the Browser Flow

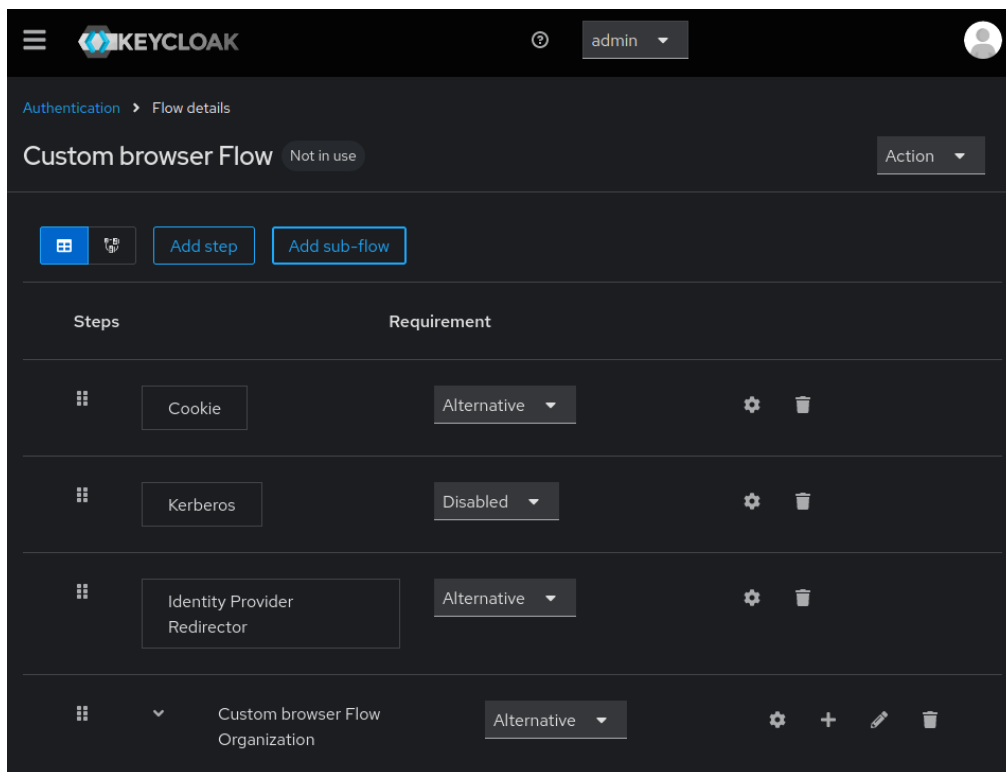


Figure 43 : Implementing MFA in Keycloak 4

Now after that the flow get duplicated, we will add the **“OTP Form”** step to implement the MFA process.

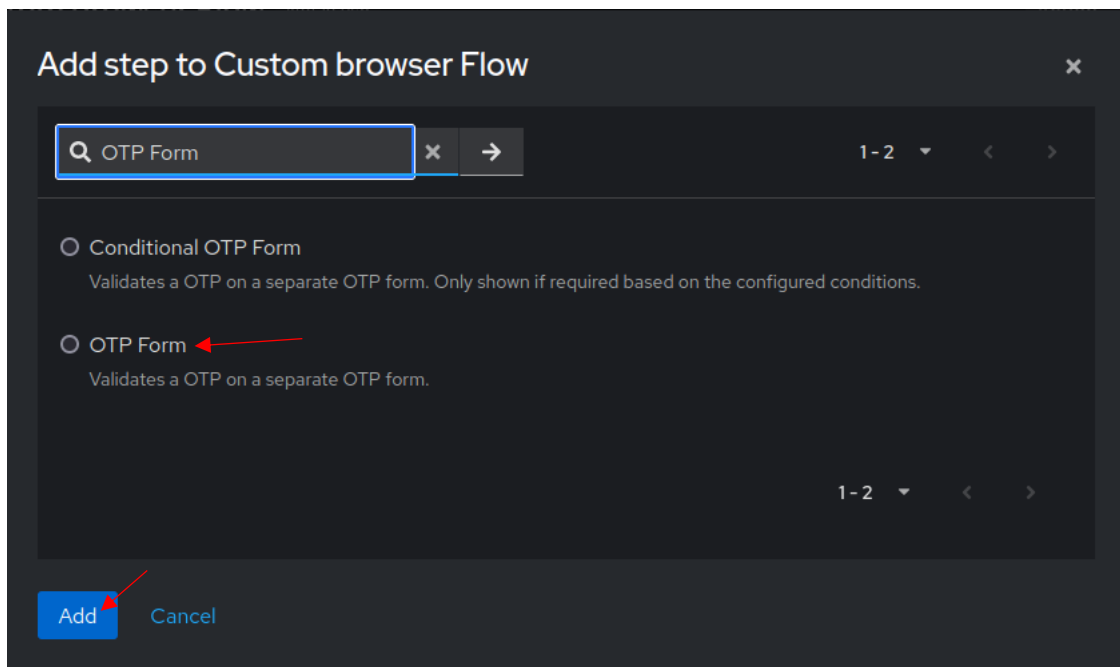


Figure 44 : Implementing MFA in Keycloak 5 – adding OTP Form

The OTP Form should be set to **“required”**!

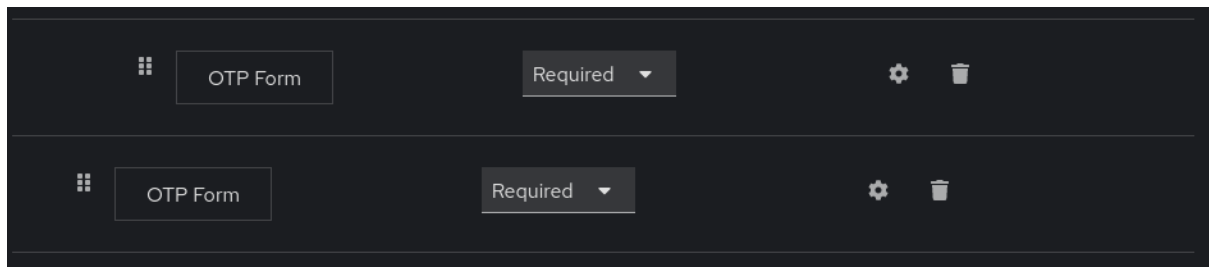


Figure 45 : Implementing MFA in Keycloak 6 - OTP Form added

Now we will go again to **“actions”**, and select **“Bind Flow”** so that the duplicated flow will be set to default!

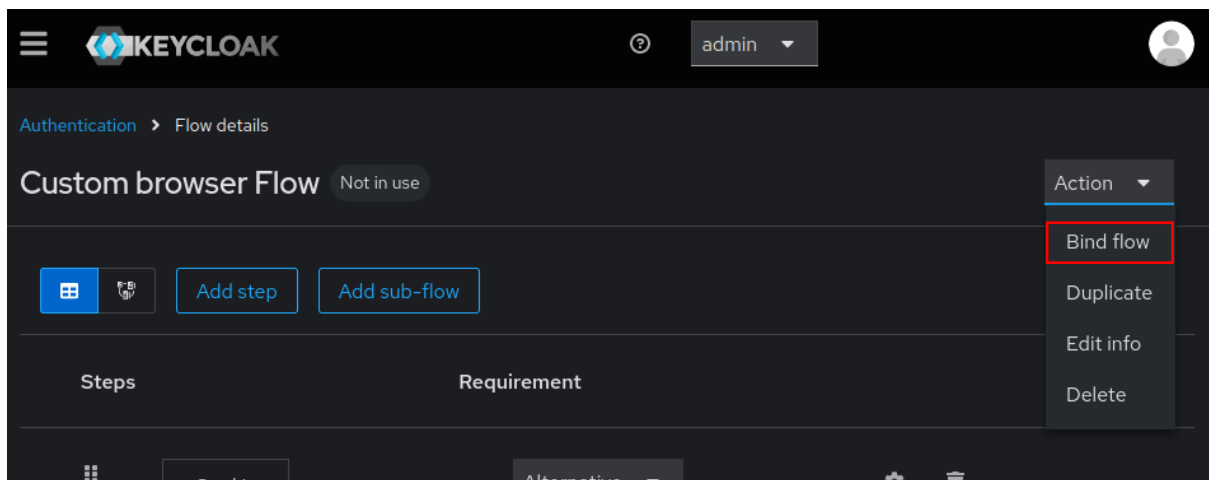


Figure 46 : Implementing MFA in Keycloak 7 - set duplicated flow to default

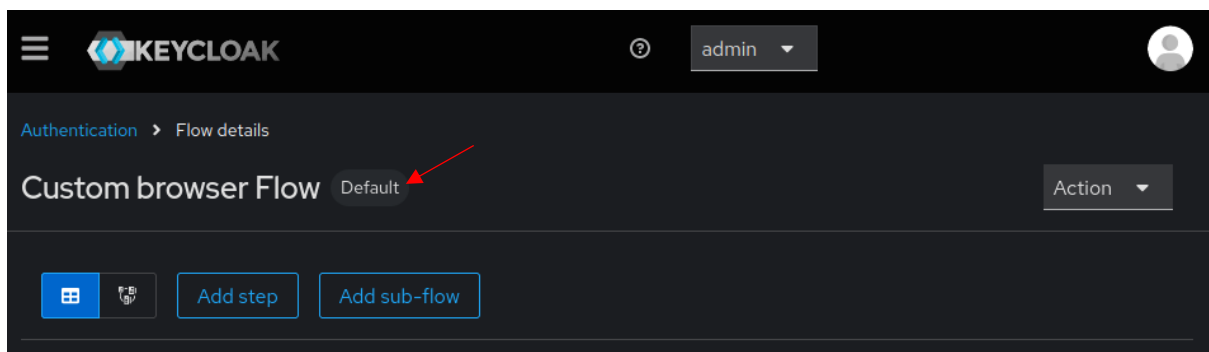


Figure 47 : Implementing MFA in Keycloak 8

Now to be able to change the OTP Policy settings, we can go to **Authentication → Policies → OTP Policy**

Authentication

Authentication is the area where you can configure and manage different credential types. [Learn more](#)

- Flows
- Required actions
- Policies**

- Password policy
- OTP Policy**
- Webauthn Policy
- Webauthn Passwordless Policy
- CIBA Policy

OTP type ☒ Time based ☐ Counter based

OTP hash algorithm SHA1

Number of digits ☒ 6 ☐ 8

Look around window - 1 +

OTP Token period 30 Seconds

Supported applications FreeOTP Google Authenticator Microsoft Authenticat...

Reusable token ☐ Off

Save Reload


Figure 48 : Implementing MFA in Keycloak 9 - changing OTP Policy settings

Now everything should be well set up, to test if the MFA mechanism has been successfully implemented, we will try to log in as one of the users.

MYCOMPANY

Mobile Authenticator Setup

1. Install one of the following applications on your mobile:
Google Authenticator
FreeOTP
Microsoft Authenticator
2. Open the application and scan the barcode:



Unable to scan?

3. Enter the one-time code provided by the application and click Submit to finish the setup.
Provide a Device Name to help you manage your OTP devices.

One-time code *

Device Name

☒ Sign out from other devices

Submit Cancel

Figure 49 : Implementing MFA in Keycloak 10 - MFA implemented

c. Implementing Role-Based Access Control (RBAC)

RBAC refers to the idea of **assigning permissions to users based on their role within an organization**. It offers a simple, manageable approach to access management that is less prone to error than assigning permissions to users individually.

When using RBAC for Role Management, you analyze the needs of your users and group them into roles based on common responsibilities.

Again, **Keycloak** offers the opportunity to implement the Role-based Access Control (RBAC) mechanism within its Identity Access and Management Process.

In our realm, to implement RBAC, we will first go to the “**Realm roles**” section then click on “**Create role**” button.

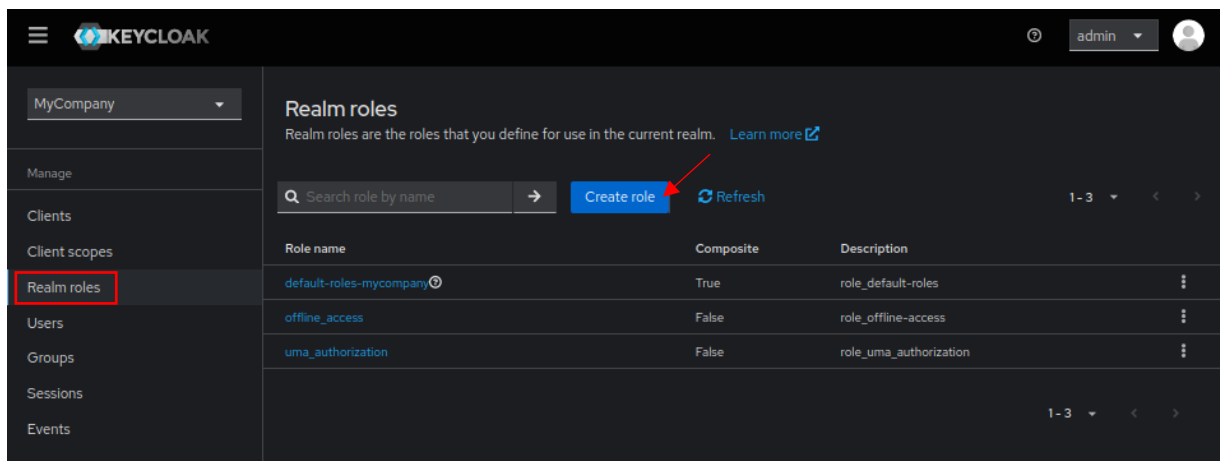


Figure 50 : Implementing RBAC in Keycloak 1

We will name the role, and it's recommended to give it a name that describes its role!

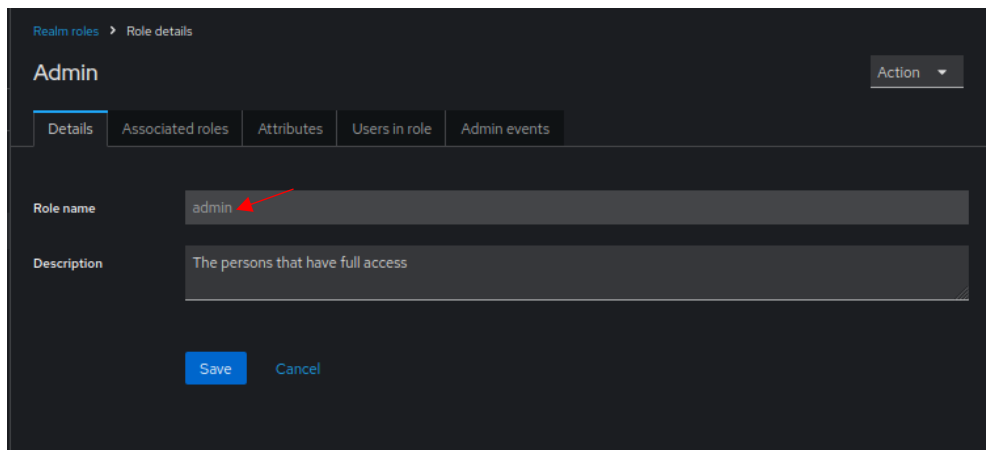


Figure 51 : Implementing RBAC in Keycloak 2 - name the role

Then go to “**Associated roles**” section and choose the roles to assign.

Of course we will give the admin role a full access.

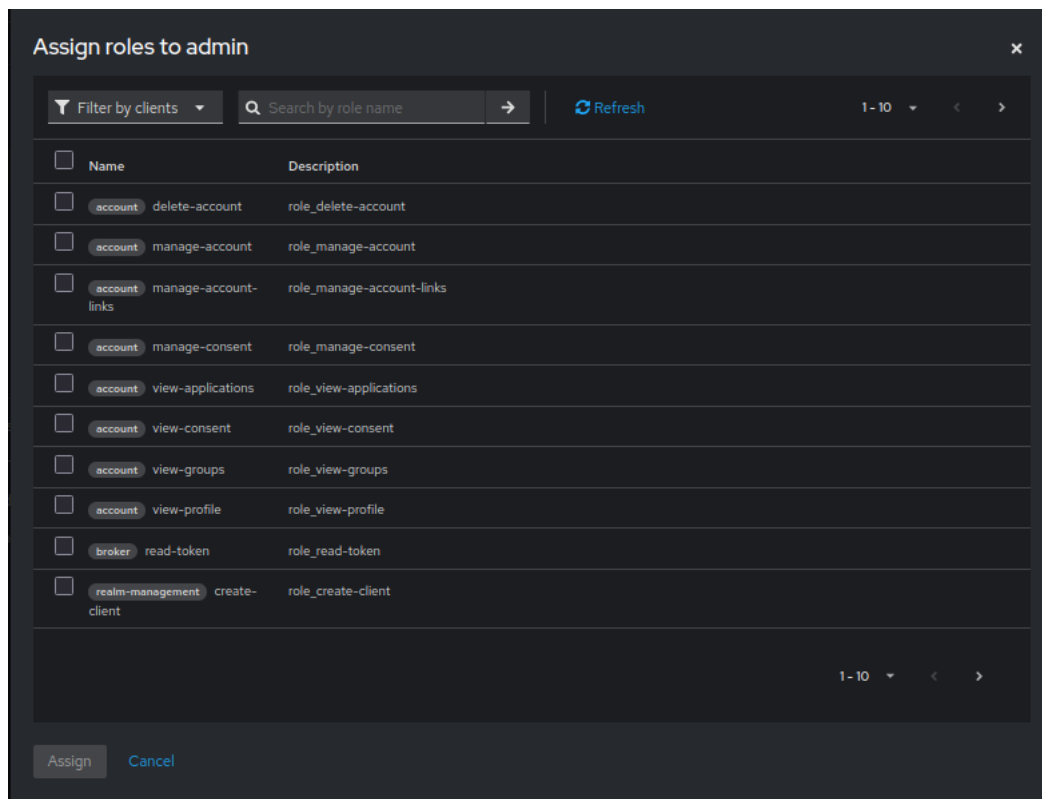


Figure 52 : Implementing RBAC in Keycloak 3 - assign roles to admin role

Then click “**Assign**”, and the role “**admin**” with full access will be successfully added.

Now to assign this role to users, we have 02 options:

- Assign the role admin to each user individually:

To do so we have to go to the wanted user, access the “**Role Mapping**” section and assign the created role.

For example, we will assign the “**admin**” role to the “**Antonio Far (afar)**” user:

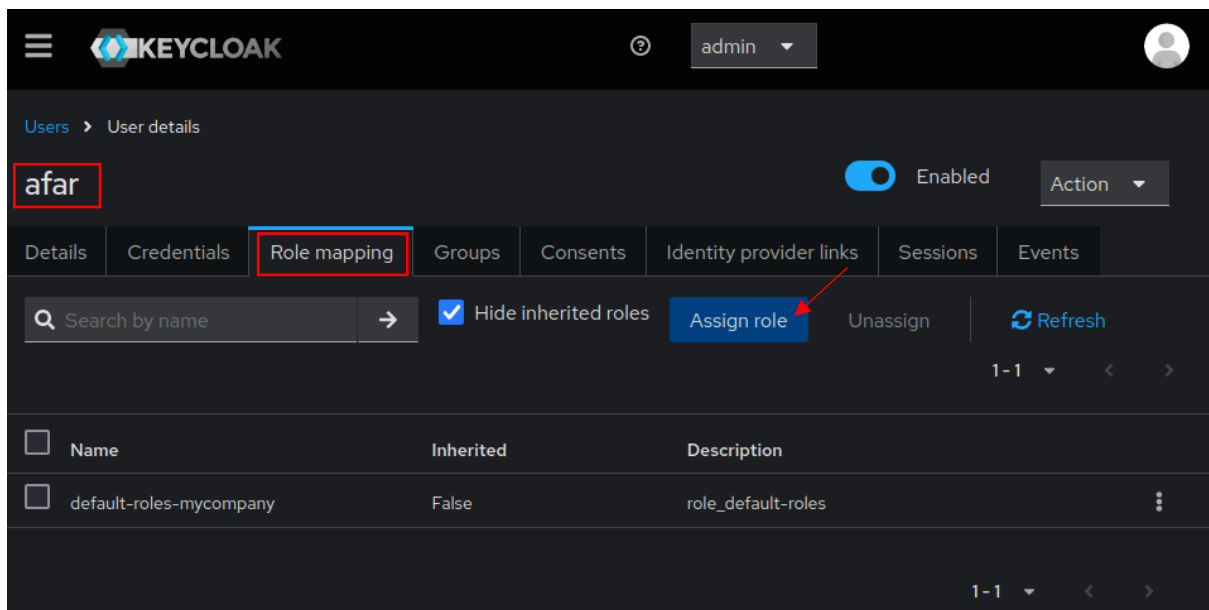


Figure 53 : Implementing RBAC in Keycloak 4 - assign admin role to afar

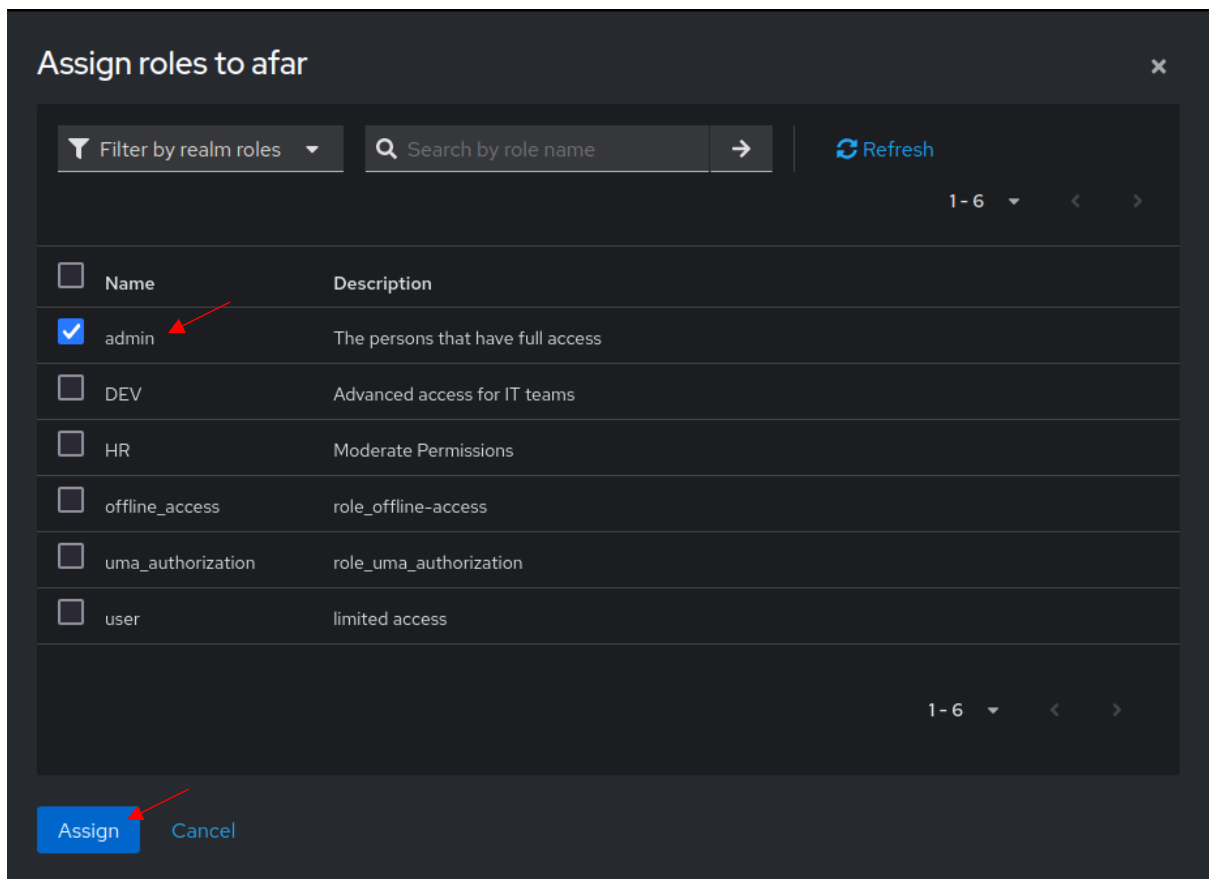


Figure 54 : Implementing RBAC in Keycloak 5 - assign admin role to afar

Now the **“admin”** role will be successfully assigned to the user **“afar”**!

- Create groups of users and then assign the role admin to each group:

To create a group, we will go to the “**Groups**” section in our realm, then click on “**Create a group**”.

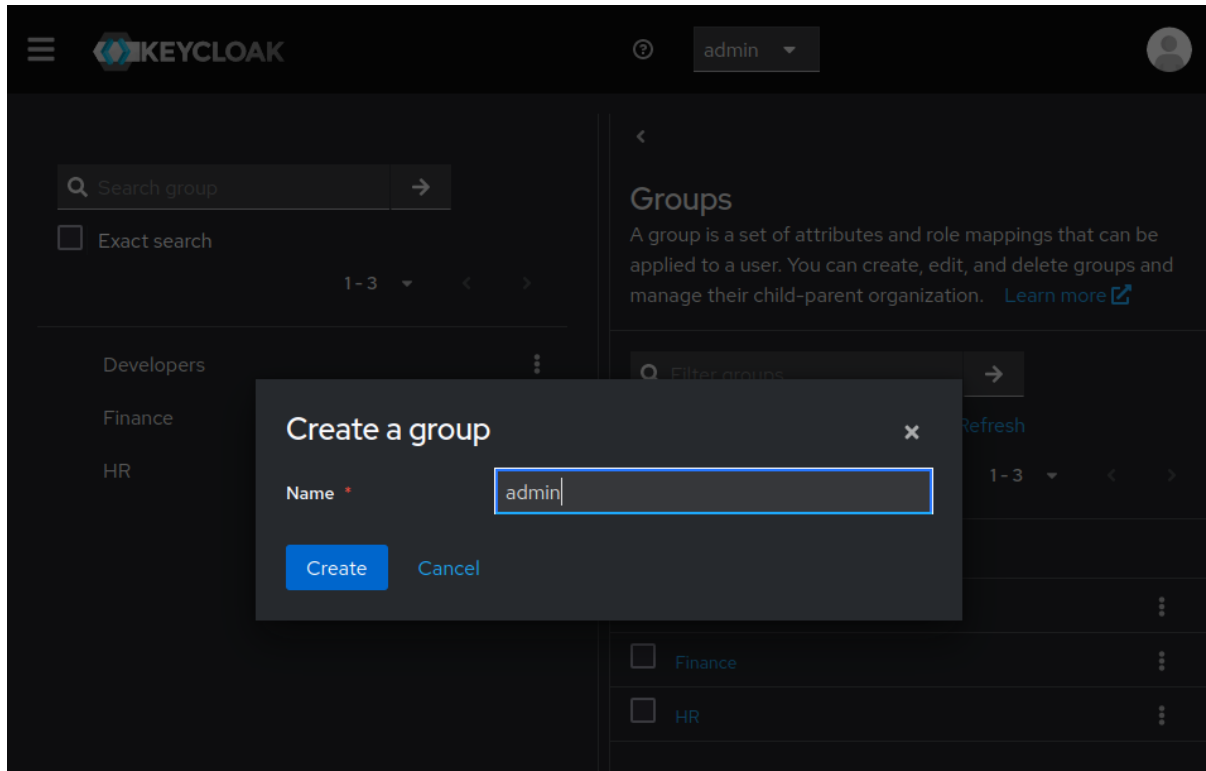


Figure 55 : Implementing RBAC in Keycloak 6 - creating a group

After the group has been created, we will go to the group **admin** → **Role Mapping** → **Assign Role** → **admin (role)** → **Assign**

Now that we assigned the “**admin**” role to the “**admin**” group, all what we have to do now is add the appropriate users to the “**admin**” group, and by that the “**admin**” role will be assigned to them automatically!

Go to wanted **user** → **Groups** → **Join Group** → **admin** → **Join**

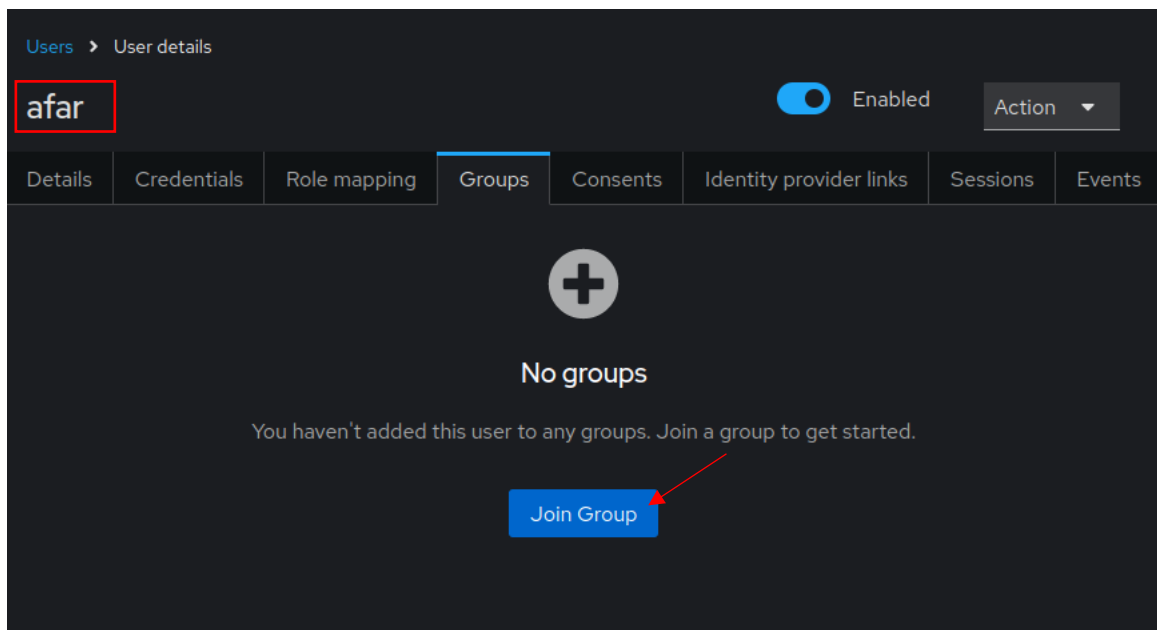


Figure 56 : Implementing RBAC in Keycloak 7 - joining a group

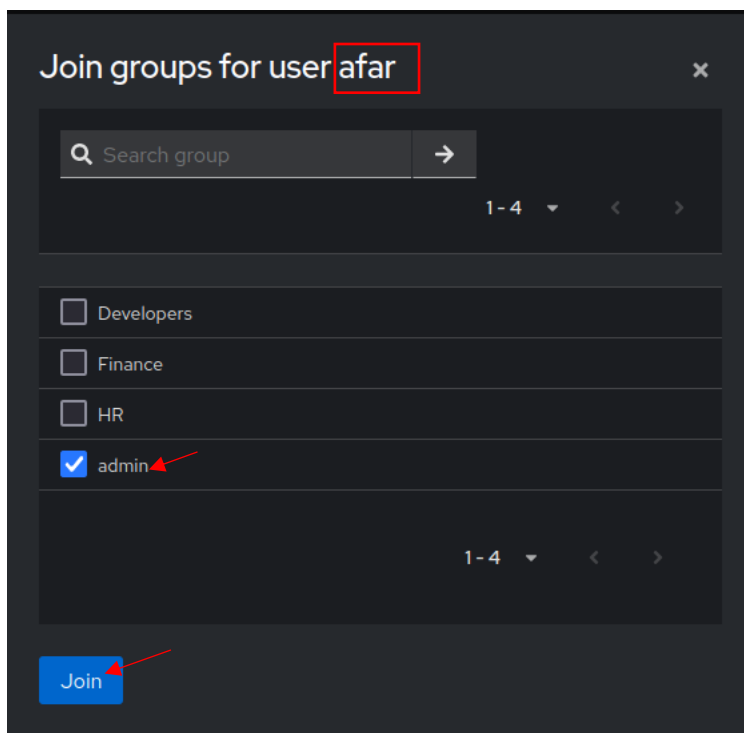


Figure 57 : Implementing RBAC in Keycloak 8 - joining a group

And by now, the admin role will be assigned to all the users the joined the **admin** group!

Chapter 2: Implementing the API security part

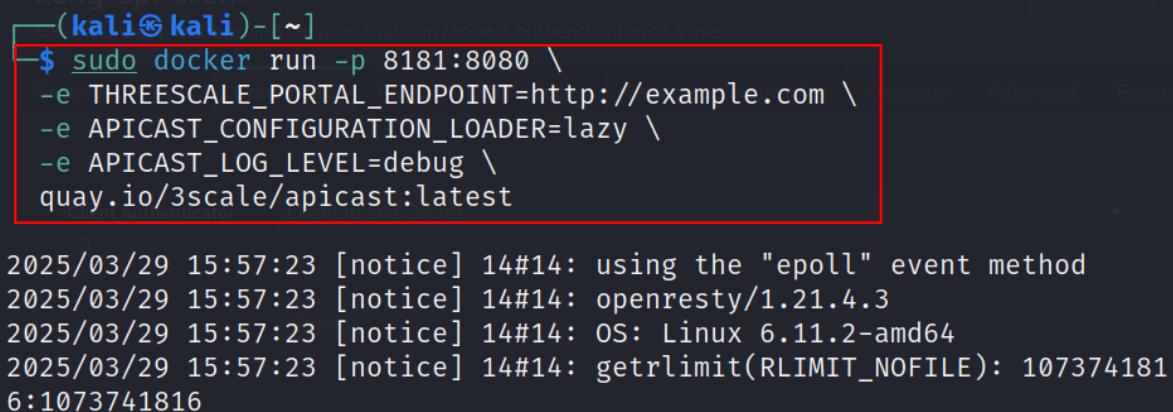
1. Introduction

In modern **Identity and Access Management (IAM)** solutions, securing and managing API access is a crucial component. **APIcast**, the API gateway from Red Hat 3scale, plays a key role in IAM by providing a scalable and secure way to expose, control, and monitor APIs. It acts as an intermediary between API consumers and backend services, enforcing **authentication**, **authorization**, and **rate limiting policies**.

By integrating **APIcast** with an IAM system such as **Keycloak**, organizations can implement robust API security mechanisms, including **OAuth2**, **OpenID Connect (OIDC)**, and **JWT-based authentication**. This ensures that only authorized users and applications can access sensitive APIs, reducing security risks such as **unauthorized access**, **token abuse**, and **API overuse**.

2. Implementing APIcast

a. Installing APIcast

A terminal window on a Kali Linux system showing the installation of APIcast using Docker. The command is: `sudo docker run -p 8181:8080 \ -e THREESCALE_PORTAL_ENDPOINT=http://example.com \ -e APICAST_CONFIGURATION_LOADER=lazy \ -e APICAST_LOG_LEVEL=debug \ quay.io/3scale/apicast:latest`. The output shows several status messages: `2025/03/29 15:57:23 [notice] 14#14: using the "epoll" event method`, `2025/03/29 15:57:23 [notice] 14#14: openresty/1.21.4.3`, `2025/03/29 15:57:23 [notice] 14#14: OS: Linux 6.11.2-amd64`, and `2025/03/29 15:57:23 [notice] 14#14: getrlimit(RLIMIT_NOFILE): 1073741816:1073741816`.

```
(kali㉿kali)-[~]  
$ sudo docker run -p 8181:8080 \  
-e THREESCALE_PORTAL_ENDPOINT=http://example.com \  
-e APICAST_CONFIGURATION_LOADER=lazy \  
-e APICAST_LOG_LEVEL=debug \  
quay.io/3scale/apicast:latest  
  
2025/03/29 15:57:23 [notice] 14#14: using the "epoll" event method  
2025/03/29 15:57:23 [notice] 14#14: openresty/1.21.4.3  
2025/03/29 15:57:23 [notice] 14#14: OS: Linux 6.11.2-amd64  
2025/03/29 15:57:23 [notice] 14#14: getrlimit(RLIMIT_NOFILE): 1073741816:1073741816
```

Figure 58 : APIcast installation

Let's check if the container has been successfully installed and in **UP** state.

```
(kali㉿kali)-[~]
$ sudo docker ps
CONTAINER ID   IMAGE                                STATUS      PORTS                               COMMAND
CREATED          STATUS      PORTS                               NAMES
1a4d48a43686   quay.io/3scale/apicast:latest       Up 5 seconds    0.0.0.0:8181→8080/tcp, :::8181→8080/tcp    "container-entrypoin...
gracious_carver
7d867fff60d8   quay.io/keycloak/keycloak:latest     Up 54 minutes   8443/tcp, 0.0.0.0:8080→8080/tcp, :
:::8080→8080/tcp, 9000/tcp             keycloak
58a30e950bc4   osixia/phpldapadmin                 Up 54 minutes   443/tcp, 0.0.0.0:8081→80/tcp, :::8081→80/tcp    phpldapadmin
7f3f406efc47   osixia/openldap                     Up 54 minutes   0.0.0.0:389→389/tcp, :::389→389/t
cp, 0.0.0.0:636→636/tcp, :::636→636/tcp    openldap
```

Figure 59 : APIcast container listing

b. Link APIcast with Keycloak

Now we have to configure Keycloak so it can use **APIcast** for API security and access management.

First of all, in our realm we will go to the “**Clients**” section then click “**Create client**” as shown in the following images:

Figure 60 : Link APIcast with Keycloak 1 - Client creation

Clients > Create client

Create client

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

2 Capability config

Client authentication ☒ On

Authorization ☐ Off

Authentication flow

- ☐ Standard flow
- ☐ Implicit flow
- ☐ OAuth 2.0 Device Authorization Grant
- ☐ Direct access grants
- ☒ Service accounts roles
- ☐ OIDC CIBA Grant

Figure 61 : Link APIcast with Keycloak 2 - Client creation

After that the client has been created, we will now need to access the client then go to “**Credentials**” and copy the “**Client Secret**”, because we will need it in the upcoming steps.

Clients > Client details

apicast-client OpenID Connect ☒ Enabled Action

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

Settings Keys **Credentials** Roles Client scopes Service accounts roles Sessions Advanced Events

Client Authenticator Client Id and Secret

Save

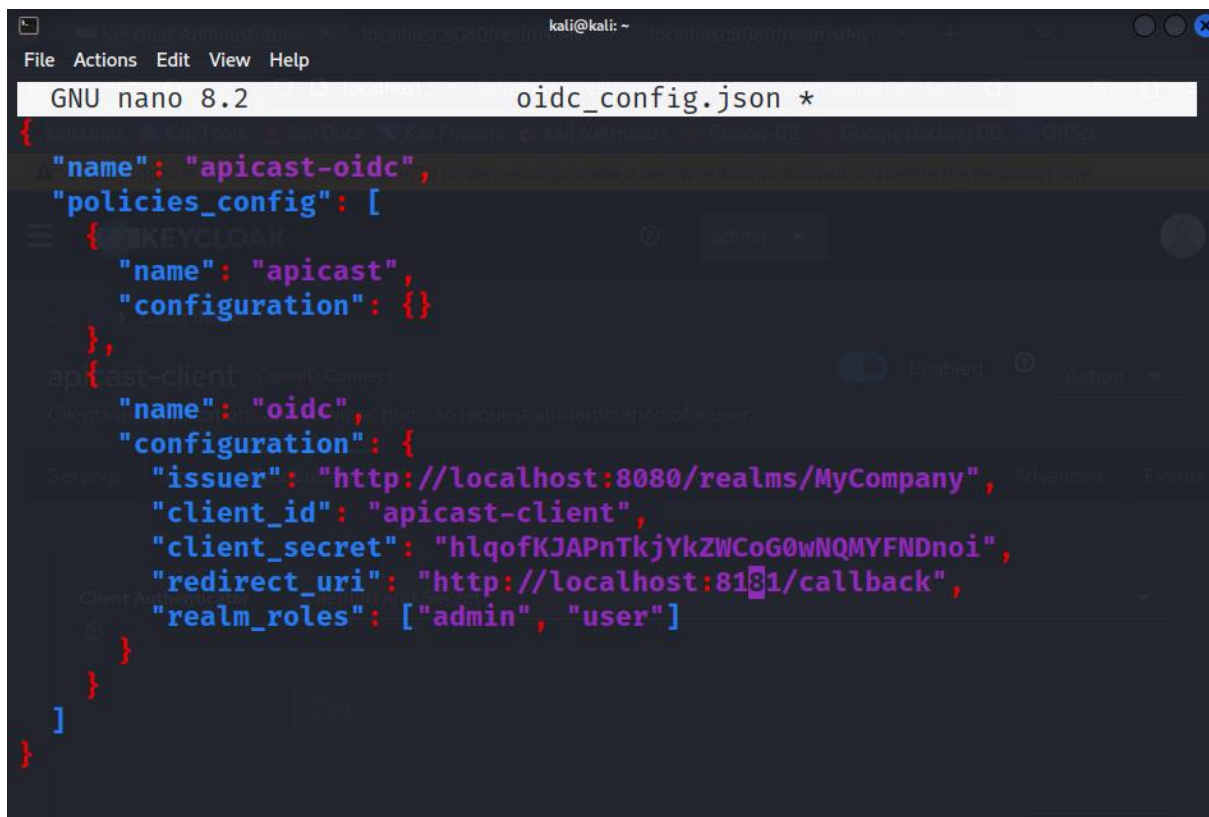
Client Secret [Redacted] [Copy] [Regenerate]

Registration access token [Redacted] [Copy] [Regenerate]

Figure 62 : Link APIcast with Keycloak 3 - copy client secret

Now, we have to configure **APIcast** to use **Keycloak** for Authentication.

To do so, we will create a file: **oidc_config.json**

A screenshot of a terminal window showing the nano text editor. The editor is open to a file named 'oidc_config.json'. The file contains a JSON configuration for linking APIcast with Keycloak. The configuration includes a 'name' field set to 'apicast-oidc', a 'policies_config' array with an object for 'apicast' (name: 'apicast', configuration: {}), and another object for 'oidc' (name: 'oidc', configuration: { issuer: 'http://localhost:8080/realms/MyCompany', client_id: 'apicast-client', client_secret: 'hlqofKJAPnTkjYkZWCoG0wNQMYFNDnoi', redirect_uri: 'http://localhost:8181/callback', realm_roles: ['admin', 'user'] }). The nano editor's status bar shows 'GNU nano 8.2' and the filename 'oidc_config.json *'.

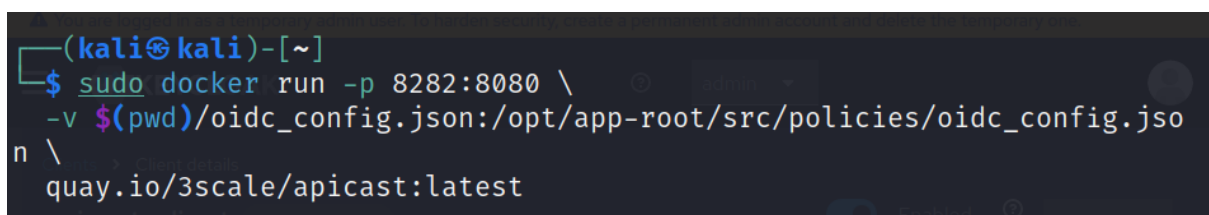
```
{
  "name": "apicast-oidc",
  "policies_config": [
    {
      "name": "apicast",
      "configuration": {}
    },
    {
      "name": "oidc",
      "configuration": {
        "issuer": "http://localhost:8080/realms/MyCompany",
        "client_id": "apicast-client",
        "client_secret": "hlqofKJAPnTkjYkZWCoG0wNQMYFNDnoi",
        "redirect_uri": "http://localhost:8181/callback",
        "realm_roles": ["admin", "user"]
      }
    }
  ]
}
```

Figure 63 : Link APIcast with Keycloak 4 - oidc_config.json

Replace:

- **Issuer:** By the URL to your realm in **Keycloak**
- **Client_id:** By the name of the **APIcast** client that you created in **Keycloak**
- **Client_secret:** by the value that we copied from the client's credentials
- **Redirect_uri:** just replace the port 8181 by the port that you assigned to **APIcast** at its installation

Now we have to mount the OIDC configuration file into the **APIcast** container.

A screenshot of a terminal window showing a command to run a Docker container. The command is: 'sudo docker run -p 8282:8080 -v \$(pwd)/oidc_config.json:/opt/app-root/src/policies/oidc_config.json quay.io/3scale/apicast:latest'. The prompt is '(kali@kali)-[~]' and the command is being entered line by line.

```
(kali@kali)-[~]
$ sudo docker run -p 8282:8080 \
-v $(pwd)/oidc_config.json:/opt/app-root/src/policies/oidc_config.json \
quay.io/3scale/apicast:latest
```

Figure 64 : Link APIcast with Keycloak 5 - mount the OIDC configuration file

Now we will use the **Access Token** to call **APIcast**:

[illegible]

Figure 66 : Link APIcast with Keycloak 7 - use the Access Token to call APIcast

If the authentication is successful, you will receive a valid response from your API.

Finally, we had successfully implemented, configured and Linked **APIcast** with **Keycloak** for providing a scalable and secure way to expose, control, and monitor APIs.

Conclusion

Implementing a **self-hosted IAM solution** using **Keycloak** and **OpenLDAP** has provided a robust and scalable authentication and authorization framework. By integrating **Keycloak** as the central identity provider, we successfully established **Single Sign-On (SSO)**, ensuring seamless user access across multiple applications. Additionally, **Role-Based Access Control (RBAC)** was enforced to define and manage user permissions effectively, while **Multi-Factor Authentication (MFA)** added an extra layer of security to protect against unauthorized access.

To extend IAM capabilities to API security and access management, we integrated **APIcast** as an API gateway. This allowed us to enforce **OAuth2-based authentication, token validation, and traffic control**, ensuring that only authorized clients could access protected resources. The combination of **Keycloak, OpenLDAP, and APIcast** provided a **comprehensive IAM solution**, balancing security, usability, and performance.

This self-hosted IAM architecture not only enhances **identity and access governance** but also ensures **API security and compliance** with modern authentication standards. By maintaining full control over identity management, authentication workflows, and API access, this solution is well-positioned to support scalable, secure, and efficient user and API interactions.

As a future improvement, **enhancing security monitoring** through the integration of **Wazuh** will strengthen threat detection, log analysis, and real-time security monitoring, allowing proactive defense against potential security incidents. By incorporating Wazuh into this IAM architecture, we aim to establish a **fully secured identity and access management ecosystem** with advanced monitoring and intrusion detection capabilities.

Regardless of the importance of security monitoring, this part is out of scope of this project, and it will be planned as part of an upcoming **Endpoint Detection and Response (EDR) implementation** project.

Webography

- What is Docker: <https://docs.docker.com/get-started/docker-overview/>
- Install Docker on Kali Linux: <https://www.kali.org/docs/containers/installing-docker-on-kali/>
- IAM explanation: <https://www.techtarget.com/searchsecurity/definition/identity-access-management-IAM-system>
- IAM Authentication methods: <https://www.manageengine.com/academy/iam-authentication-methods.html>
- Difference between Authentication and Authorization: <https://www.geeksforgeeks.org/difference-between-authentication-and-authorization/>
- MFA Explanation: [https://www.onelogin.com/learn/what-is-mfa#:~:text=Multi%2Dfactor%20Authentication%20\(MFA\)%20is%20an%20authentication%20method%20that,online%20account%2C%20or%20a%20VPN.](https://www.onelogin.com/learn/what-is-mfa#:~:text=Multi%2Dfactor%20Authentication%20(MFA)%20is%20an%20authentication%20method%20that,online%20account%2C%20or%20a%20VPN.)
- RBAC Explanation: [https://auth0.com/docs/manage-users/access-control/rbac#:~:text=Role%2Dbased%20access%20control%20\(RBAC,assigning%20permissions%20to%20users%20individually.](https://auth0.com/docs/manage-users/access-control/rbac#:~:text=Role%2Dbased%20access%20control%20(RBAC,assigning%20permissions%20to%20users%20individually.)
- Keycloak Official Documentation: <https://www.Keycloak.org/documentation>
- OpenLDAP Official Documentation: <https://www.OpenLDAP.org/software/>
- APIcast Documentation : https://docs.redhat.com/en/documentation/red_hat_3scale_api_management/2.3/html/deployment_options/APIcast-overview#configuring_your_service
- APIcast Github repository: <https://github.com/3scale/APIcast>