

Abdel-malik FOFANA
Mario EL DAHDAH
Thibault FOURMONT
Clarisse VERON

Projet 9: KEYCLOAK (Identity & Accès Management). Implémentations et scénario de test

1 Présentation de keycloak

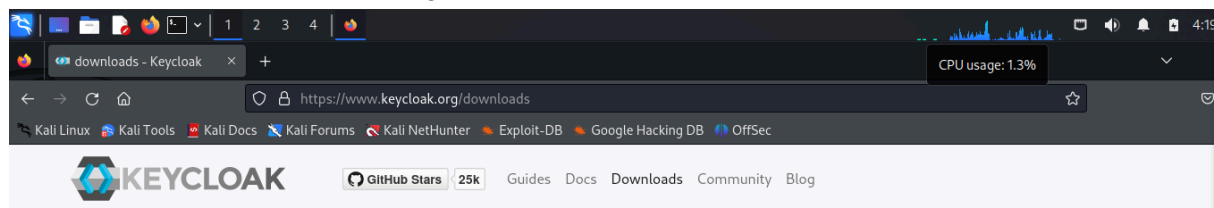
IAM (Identity and Access Management) : L'IAM est un ensemble de processus et de technologies pour gérer les identités numériques et contrôler l'accès aux ressources, garantissant ainsi que seules les bonnes personnes accèdent aux bonnes ressources au bon moment.

À quoi sert Keycloak : Keycloak est une solution de gestion d'identité et d'accès (IAM) qui permet de gérer l'authentification, les autorisations, et le SSO (Single Sign-On) pour les applications et les utilisateurs.

Comment on l'utilise en entreprise : En entreprise, Keycloak centralise la gestion des identités pour simplifier l'accès sécurisé aux applications, mettre en œuvre le SSO, gérer les rôles et permissions, et renforcer la sécurité grâce à des fonctionnalités comme l'authentification multi-facteurs (MFA).

2 Installation de Keycloak

On va sur le site web et télécharge le zip



Downloads **26.1.0**

For a list of community maintained extensions check out the [Extensions](#) page.

Server

Keycloak	Distribution powered by Quarkus	ZIP (sha1) TAR.GZ (sha1)
Container image	For Docker, Podman, Kubernetes and OpenShift	Quay
Operator	For Kubernetes and OpenShift	OperatorHub
Third-party licenses	License and source code information for third-party dependencies	HTML

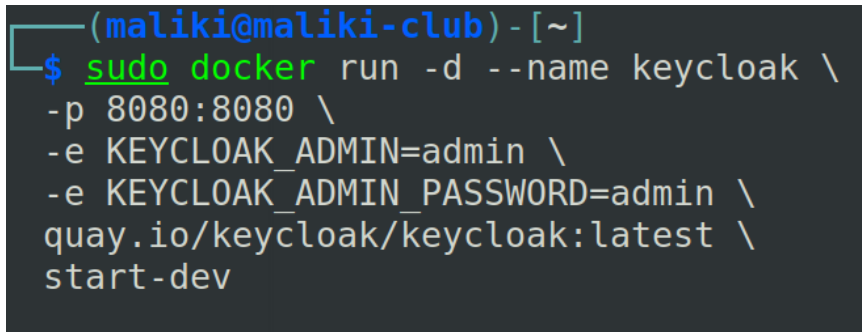
Quickstarts

Quickstarts distribution	GitHub ZIP
--------------------------	--

Client Adapters

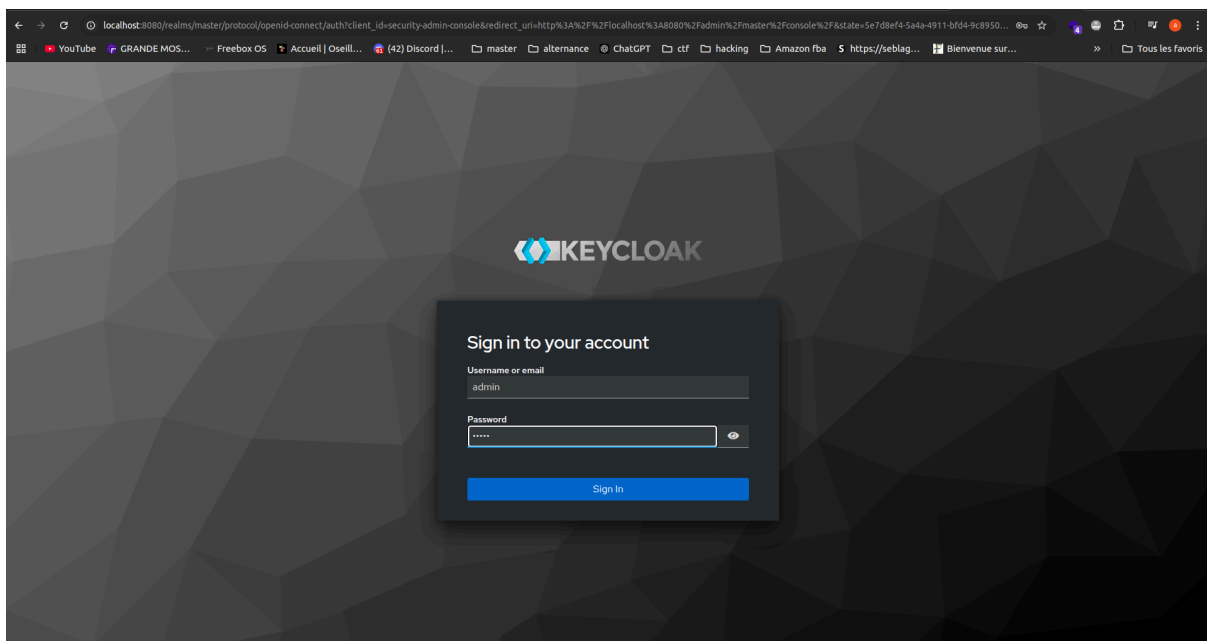
Puis on fait la commande cette commande docker pour pouvoir avoir le localhost:8080

```
docker run -d --name keycloak \  
-p 8080:8080 \  
-e KEYCLOAK_ADMIN=admin \  
-e KEYCLOAK_ADMIN_PASSWORD=admin \  
quay.io/keycloak/keycloak:latest \  
start-dev
```

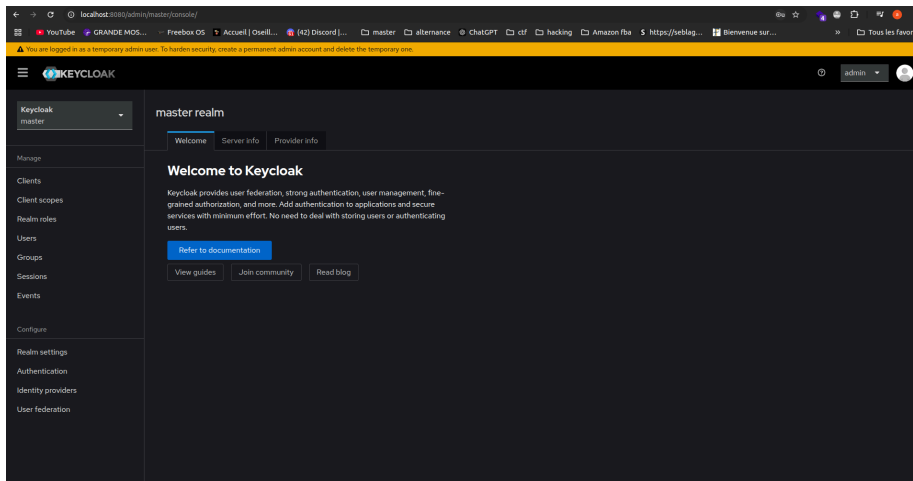


```
(maliki@maliki-club) - [~]  
$ sudo docker run -d --name keycloak \  
-p 8080:8080 \  
-e KEYCLOAK_ADMIN=admin \  
-e KEYCLOAK_ADMIN_PASSWORD=admin \  
quay.io/keycloak/keycloak:latest \  
start-dev
```

Et on a accès à l'interface



On se connecte avec admin admin

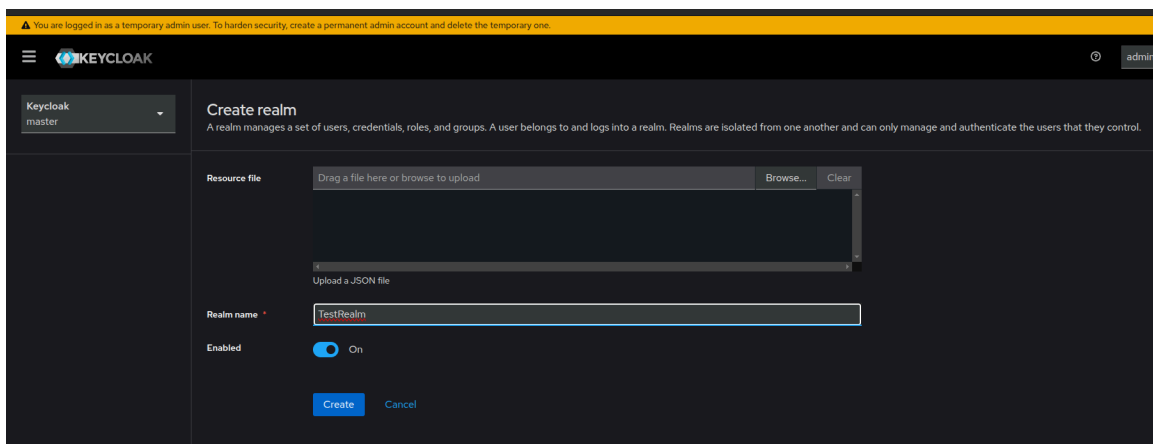


3 Configuration de base:

Création d'un Realm :

Un **realm** dans Keycloak est une instance isolée de gestion d'identité qui regroupe des utilisateurs, des applications, des rôles, et des configurations spécifiques, permettant de séparer les données et les règles d'accès entre différents environnements ou projets.

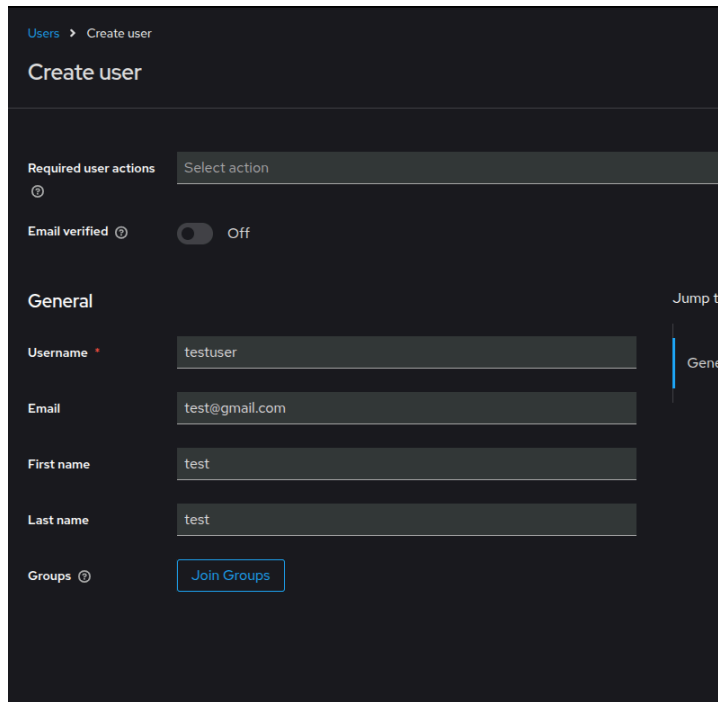
Real "TestRealm"



Ajout d'un utilisateur :

Un **user** (utilisateur) dans Keycloak est une entité représentant une personne ou un système qui peut s'authentifier et interagir avec des applications ou services gérés dans un **realm**,

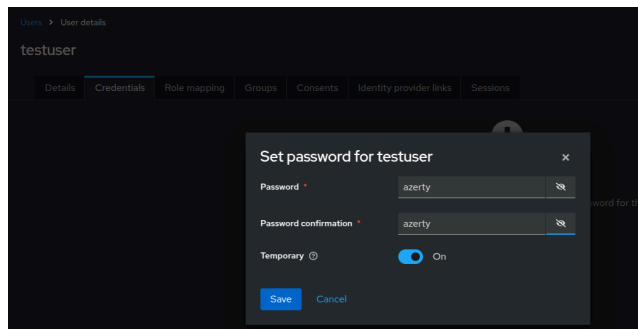
avec des informations comme un identifiant, des rôles et des attributs spécifiques.



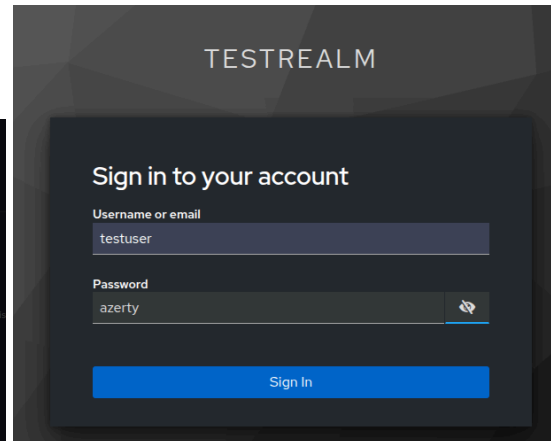
The screenshot shows the 'Create user' form in Keycloak. At the top, there's a breadcrumb 'Users > Create user' and the title 'Create user'. Below this, there are two sections: 'Required user actions' with a 'Select action' dropdown, and 'Email verified' with a toggle switch set to 'Off'. The 'General' section contains several input fields: 'Username' (filled with 'testuser'), 'Email' (filled with 'test@gmail.com'), 'First name' (filled with 'test'), and 'Last name' (filled with 'test'). At the bottom of the 'General' section, there's a 'Groups' section with a 'Join Groups' button. On the right side, there's a vertical sidebar with a 'Jump to' section and a 'General' link.

On crée un user “testuser” et “testadmin” et on ajoute un mot de passe à notre utilisateur et on peut voir si on peut se connecter sur le lien :

<http://localhost:8080/realms/TestRealm/account>



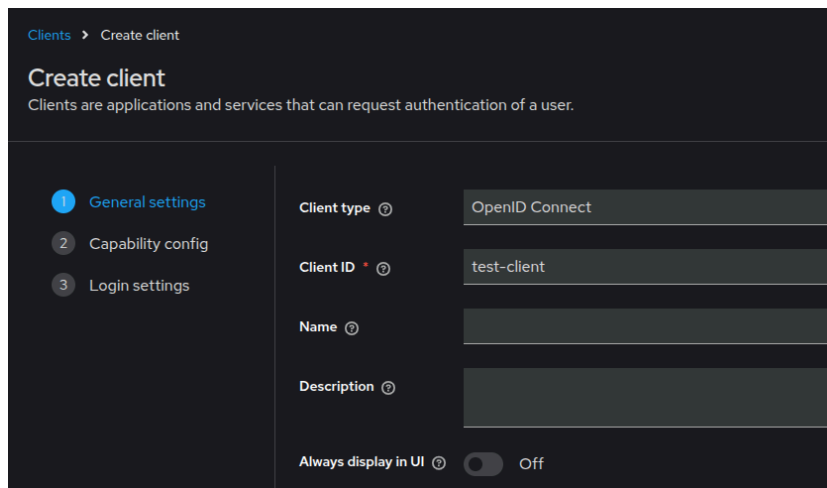
The screenshot shows the 'Set password for testuser' dialog box. It has a title bar with a close button. Inside, there are two input fields: 'Password' (filled with 'azerty') and 'Password confirmation' (filled with 'azerty'). Below these fields, there's a 'Temporary' toggle switch set to 'On'. At the bottom, there are 'Save' and 'Cancel' buttons.



The screenshot shows the 'Sign in to your account' page in Keycloak. The page has a dark background with a light gray header area containing the text 'TESTREALM'. The main content area has a title 'Sign in to your account' and two input fields: 'Username or email' (filled with 'testuser') and 'Password' (filled with 'azerty'). Below these fields, there's a blue 'Sign In' button.

Création des clients:

Les clients sont des entités qui peuvent demander à Keycloak d'authentifier un utilisateur. Le plus souvent, les clients sont des applications et des services qui souhaitent utiliser Keycloak pour se sécuriser et fournir une solution de connexion unique. Les clients peuvent également être des entités qui souhaitent simplement demander des informations d'identité ou un jeton d'accès afin de pouvoir invoquer de manière sécurisée d'autres services sur le réseau qui sont sécurisés par Keycloak.



Clients > Create client

Create client

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

- 1 General settings
- 2 Capability config
- 3 Login settings

Client type ⓘ OpenID Connect

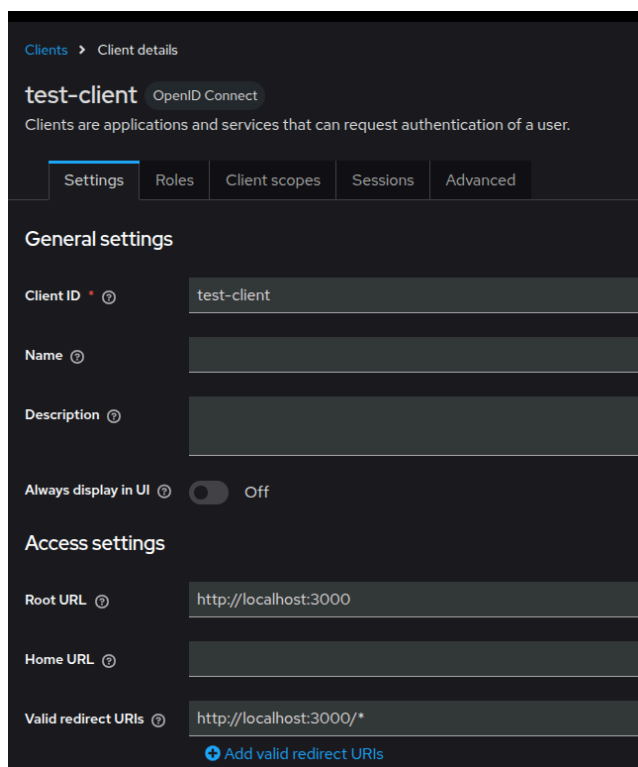
Client ID * ⓘ test-client

Name ⓘ

Description ⓘ

Always display in UI ⓘ ☐ Off

Le **Root URL** dans Keycloak est l'URL de base de votre application (le client). C'est là où Keycloak redirige les utilisateurs après qu'ils se soient authentifiés avec succès



Clients > Client details

test-client

OpenID Connect

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

- Settings
- Roles
- Client scopes
- Sessions
- Advanced

General settings

Client ID * ⓘ test-client

Name ⓘ

Description ⓘ

Always display in UI ⓘ ☐ Off

Access settings

Root URL ⓘ http://localhost:3000

Home URL ⓘ

Valid redirect URIs ⓘ http://localhost:3000/*

[+ Add valid redirect URIs](#)

4 Téléchargement de notre LDAP (AD simulé)

Pour télécharger un LDAP test on a utilisé cette commande docker

```
docker run -d --name ldap-server \
-p 389:389 \
-p 636:636 \
--env LDAP_ORGANISATION="My Organization" \
--env LDAP_DOMAIN="mydomain.local" \
--env LDAP_ADMIN_PASSWORD="admin" \
osixia/openldap:latest
```

et on a verifié que LDAP a bien été configuré avec la commande `ldapsearch -x -H ldap://localhost -b dc=mydomain,dc=local -D "cn=admin,dc=mydomain,dc=local" -w admin` et on peut voir que tout est bien configuré

```
(maliki@maliki-club) - [~]
$ ldapsearch -x -H ldap://localhost -b dc=mydomain,dc=local -D "cn=admin,dc=
domain,dc=local" -w admin

# extended LDIF
#
# LDAPv3
# base <dc=mydomain,dc=local> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# mydomain.local
dn: dc=mydomain,dc=local
objectClass: top
objectClass: dcObject
objectClass: organization
o: My Organization
dc: mydomain

# search result
```

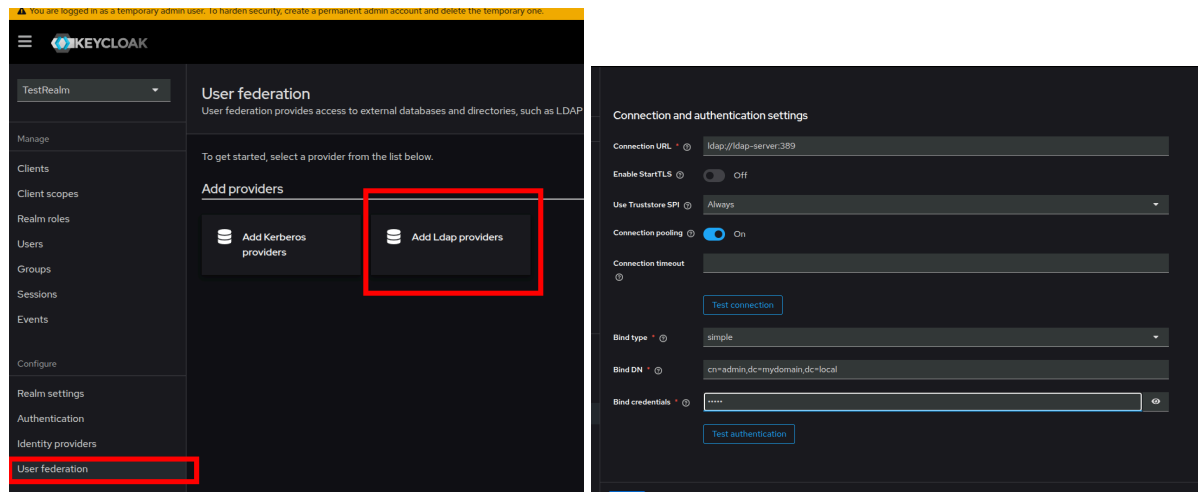
Notre LDAP, hébergé via Docker ([osixia/openldap](#)), utilise le domaine `dc=mydomain,dc=local` pour représenter **My Organization**. Accessible sur les ports **389** (LDAP) et **636** (LDAPS), il est administré par l'utilisateur **admin** et peut être intégré à Keycloak pour l'authentification et la gestion des rôles. Et surtout n'oublions pas de mettre sur le même network le ldap et keycloak

```
(maliki@maliki-club) - [~]
$ sudo docker network create ldap-network
e487df658be3eef0f646073c1f2d7fa03700eaec4b4151488256b97f23f423c
(maliki@maliki-club) - [~]
$ sudo docker network connect ldap-network ldap-server
(maliki@maliki-club) - [~]
$ sudo docker network connect ldap-network keycloak
```

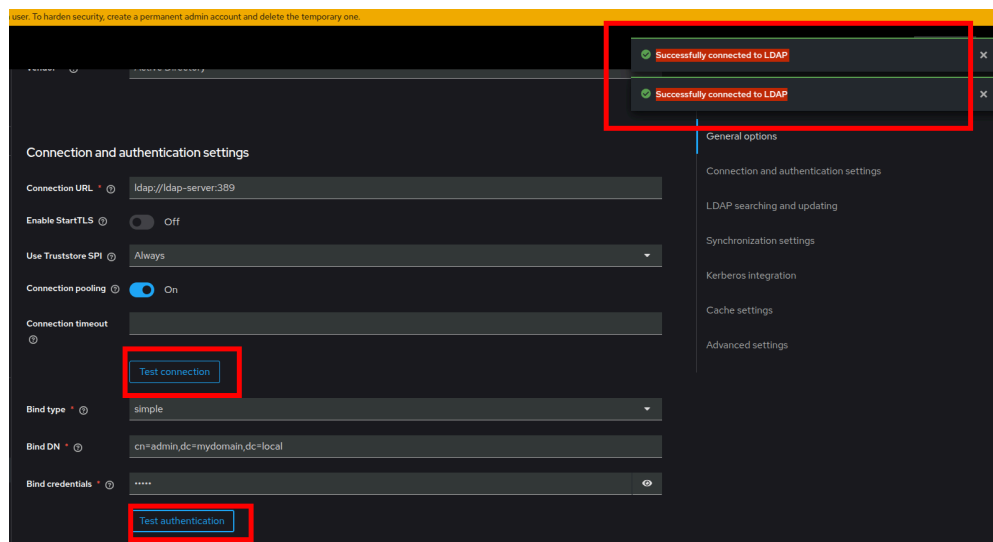
5 Configurer le lien entre Keycloak et LDAP (AD simulé)

Ensuite on va dans l'onglet "User federation" et on clique sur LDAP et on configure avec les informations de notre LDAP

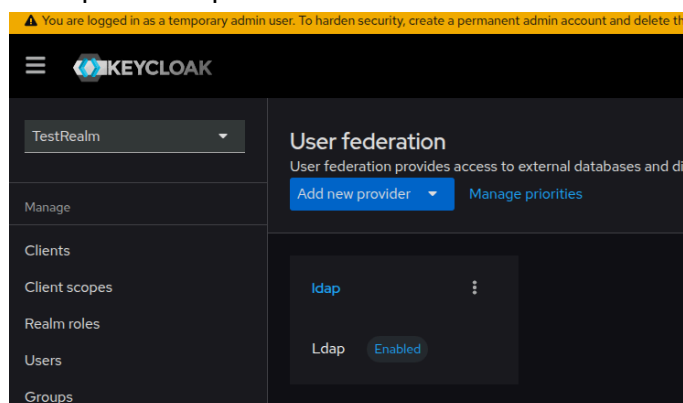
- **Vendor** : `Other` (car ce n'est pas un AD réel).
- **Connection URL** : `ldap://ldap-server:389`
- **Users DN** : `dc=mydomain,dc=local`
- **Bind DN** : `cn=admin,dc=mydomain,dc=local`
- **Bind Credential** : `admin`



Une fois les éléments rempli on peut voir que les test d'authentification et de connection fonctionnes



Et on peut voir que ca a marché



Nous allons maintenant ajouter un utilisateur appelé **testuser2** à notre serveur LDAP et vérifier qu'il est correctement synchronisé dans Keycloak dn: uid=testuser2,dc=mydomain,dc=local objectClass: inetOrgPerson sn: User2 cn: Test User2 uid: testuser2 userPassword: testpassword2 (on met ca dans le fichier testuser2.ldif) et on fait la commande

```
sudo docker exec -i ldap-server ldapadd -x -D "cn=admin,dc=mydomain,dc=local" -w admin < testuser2.ldif
```

```
(maliki@maliki-club) - [~/Téléchargements]
$ sudo docker exec -i ldap-server ldapadd -x -D "cn=admin,dc=mydomain,dc=local" -w admin < testuser2.ldif

[sudo] Mot de passe de maliki :
adding new entry "uid=testuser2,dc=mydomain,dc=local"

(maliki@maliki-club) - [~/Téléchargements]
$ ldapsearch -x -H ldap://localhost -b dc=mydomain,dc=local -D "cn=admin,dc=mydomain,dc=local" -w admin "(uid=testuser2)"

# extended LDIF
#
# LDAPv3
# base <dc=mydomain,dc=local> with scope subtree
# filter: (uid=testuser2)
# requesting: ALL
#
# testuser2, mydomain.local
dn: uid=testuser2,dc=mydomain,dc=local
objectClass: inetOrgPerson
```

essayons maintenant de nous connecter avec testuser2 et testpassword2 , on peut voir que testuser2 est là

Users

Users are the users in the current realm. [Learn more](#)


User list

Default search Search user → Add user Delete user Refresh 1-3 < >

<input type="checkbox"/>	Username	Email	Last name	First name	
<input type="checkbox"/>	admin	admin@test.com	min	ad	⋮
<input type="checkbox"/>	testuser	test@gmail.com	test	test	⋮
<input type="checkbox"/>	testuser2	—	User2	Test User2	⋮

1-3 < >

Et on arrive à se connecter avec testuser2 ajouter uniquement via LDAP




Sign in to your account

Username or email

testuser2

Password

..... 

Sign In

KEYCLOAK

Test User2 User2

Personal info

Account security

Applications

Personal info

Manage your basic information

General

Username *

testuser2

Email *

First name *

Test User2

Last name *

User2

Jump to section

General

6 Configurer le lien entre Keycloak et Kerberos

Configuration de Kerberos

On crée un réseau docker “auth-network”

```
thibaultfourmont@mac Desktop % docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
cdc48a1eaf38        auth-network        bridge              local
6455faa24168        bridge              bridge              local
c286509887c4        host                host                local
a8c2b376cae6        none                null                local
thibaultfourmont@mac Desktop %
```

Je crée aussi un conteneur Docker pour le serveur **Kerberos** :

```
docker run -d --name kerberos-server \
  --network auth-network \
  -h kerberos.example.com \
  -p 1088:88 \ -p 1464:464 \
  ubuntu bash -c "apt update && apt install -y krb5-kdc krb5-admin-server && sleep infinity"
```

On peut voir que le serveur Kerberos est fonctionnel

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	kerberos-server	d3c1918bbd21	ubuntu	1464:464 ↗ Show all ports (2)	0%	8 minutes ago	<input type="checkbox"/> ⋮ 🗑️

-On définit les permissions administratives avec la commande suivante :

echo "/admin *" > /etc/krb5kdc/kadm5.acl*

-On lance le service du KDC et de l'admin Server

```
root@kerberos:/# ls -l /tmp/http.keytab
-rw----- 1 root root 184 Jan 24 16:35 /tmp/http.keytab
root@kerberos:/# ps aux | grep krb5kdc
ps aux | grep kadmind
root      504  0.0  0.0  10700  3712 ?        Ss   16:33   0:00 krb5kdc
root      508  0.0  0.0  10700  3740 ?        Ss   16:33   0:00 krb5kdc
root      512  0.0  0.0  10700  3804 ?        Ss   16:34   0:00 krb5kdc
root      524  0.0  0.0   3124  1384 pts/0    S+   17:08   0:00 grep --co
lor=auto krb5kdc
root      514  0.0  0.0   5900  3244 ?        Ss   16:34   0:00 kadmind
root      526  0.0  0.0   3124  1384 pts/0    S+   17:08   0:00 grep --co
lor=auto kadmind
root@kerberos:/#
```

-On crée l'utilisateur testuser

kadmin.local -q "addprinc testuser"

```
root@kerberos:/# kadmin.local -q "listprincs"
Authenticating as principal root/admin@ATHENA.MIT.EDU with password.
HTTP/keycloak.example.com@ATHENA.MIT.EDU
K/M@ATHENA.MIT.EDU
kadmin/admin@ATHENA.MIT.EDU
kadmin/changepw@ATHENA.MIT.EDU
krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
testuser@ATHENA.MIT.EDU
root@kerberos:/#
```

-On crée le principal pour Keycloak et on genere le fichier keytab

kadmin.local -q "addprinc -randkey HTTP/keycloak.example.com@ATHENA.MIT.EDU"

*kadmin.local -q "ktadd -k /tmp/http.keytab
HTTP/keycloak.example.com@ATHENA.MIT.EDU"*

-Ensuite on copie le fichier keytab sur la machine hôte

docker cp kerberos-server:/tmp/http.keytab ~/Desktop/http.keytab

Deployment de keycloak et intégration

-On lance le conteneur Keycloak

```
docker run -d --name keycloak \  
  --network auth-network \  
  -p 8080:8080 \  
  -e KEYCLOAK_ADMIN=admin \  
  -e KEYCLOAK_ADMIN_PASSWORD=admin \  
  quay.io/keycloak/keycloak:latest start-dev
```

-On ajoute le fichier keytab dans keycloak, voici les commandes qu'on a utilisé :

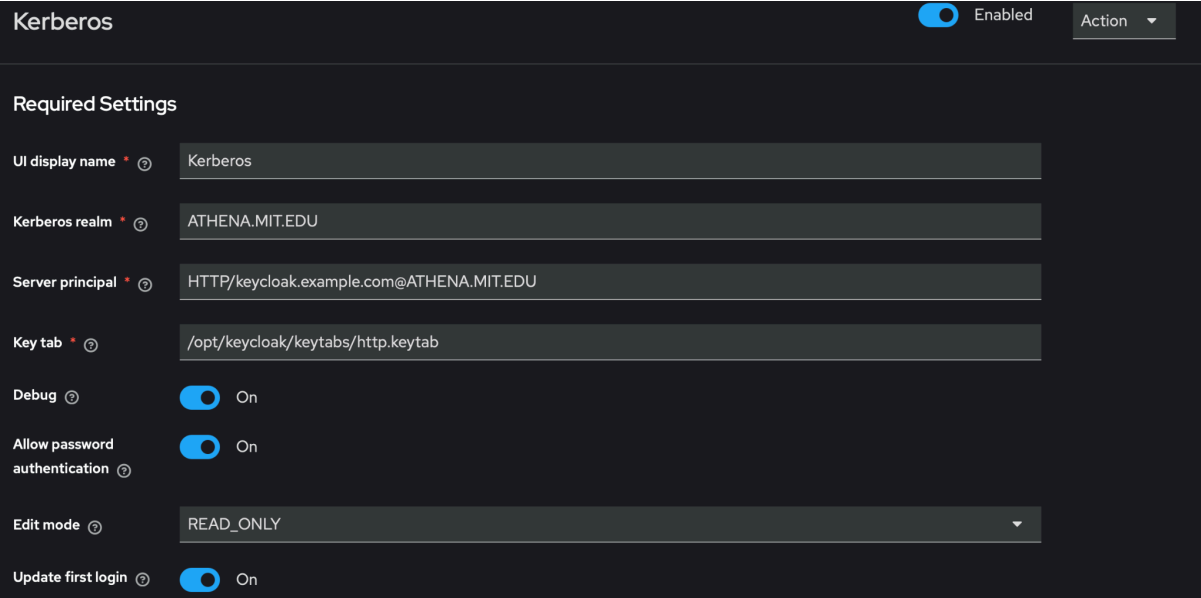
```
docker exec -it --user root keycloak mkdir -p /opt/keycloak/keytabs/
```

```
docker cp ~/Desktop/http.keytab keycloak:/opt/keycloak/keytabs/http.keytab
```

```
docker exec -it keycloak ls -l /opt/keycloak/keytabs/http.keytab
```

Configuration de KeyCloak pour utiliser Kerberos

-Une fois le fichier keytab bien placé, nous avons configuré Keycloak en passant par l'interface graphique.



The screenshot shows the 'Kerberos' configuration page in the Keycloak Admin Console. At the top right, there is a toggle switch labeled 'Enabled' which is turned on, and an 'Action' dropdown menu. Below this is the 'Required Settings' section. It contains several configuration fields: 'UI display name' with the value 'Kerberos', 'Kerberos realm' with 'ATHENA.MIT.EDU', 'Server principal' with 'HTTP/keycloak.example.com@ATHENA.MIT.EDU', and 'Key tab' with '/opt/keycloak/keytabs/http.keytab'. There are also three toggle switches: 'Debug' (On), 'Allow password authentication' (On), and 'Update first login' (On). Finally, the 'Edit mode' is set to 'READ_ONLY' via a dropdown menu.

Setting	Value
UI display name	Kerberos
Kerberos realm	ATHENA.MIT.EDU
Server principal	HTTP/keycloak.example.com@ATHENA.MIT.EDU
Key tab	/opt/keycloak/keytabs/http.keytab
Debug	On
Allow password authentication	On
Edit mode	READ_ONLY
Update first login	On

-Après on synchronise les utilisateurs

On remarque que la synchronisation a bien fonctionnée car l'utilisateur qu'on a créé "testuser@athena.mit.edu" apparaît dans les utilisateurs.

Username	Email	Last name	First name	
admin	—	—	—	⋮
testuser	testuser@athena.mit.edu	—	—	⋮

On lance keycloak sur le réseau docker grâce à la commande suivante

```
docker run -d --name keycloak \
  --network auth-network \
  -p 8080:8080 \
  -e KEYCLOAK_ADMIN=admin \
  -e KEYCLOAK_ADMIN_PASSWORD=admin \
  quay.io/keycloak/keycloak:latest \
  start-dev
```

On peut voir que KeyCloak tourne bien

keycloak	5d52aa686e6d	keycloak/keycloak 8080:8080	0.34%	7 minutes ago	⋮	🗑
----------	--------------	-----------------------------	-------	---------------	---	---

Screen du réseau docker

```
thibaultfourmont@mac Desktop % docker network inspect auth-network
[
  {
    "Name": "auth-network",
    "Id": "cdc48a1eaf38545b1128dde3f46da51980e9f2d2b58c8929194de5181d32e57c",
    "Created": "2025-01-22T21:05:08.342929541Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "5d52aa686e6d561b58f591123e173ca7d5376283bb4047f9f0ce0eb2e1f85fd5": {
        "Name": "keycloak",
        "EndpointID": "ace75f9496693f9967e6a6be3930aa98c4ee8b25bc6af01992f257c136906b13",
        "MacAddress": "02:42:ac:12:00:03",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
      },
      "d3c1918bbd215993e252ce5aa66d6e97fbdab637cf1b2809bc752d4e361540a7": {
        "Name": "kerberos-server",
        "EndpointID": "edddfecb03d943a592616afcf77d3a6351b27a9f3b0391bd3a7d3e8ab6a2e0b3",
        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
thibaultfourmont@mac Desktop %
```

Configuration du serveur Keberos

On rentre dans le container kerberos

```
thibaultfourmont@mac Desktop % docker exec -it kerberos-server bash
root@kerberos:/# dpkg -l | grep krb5
ii  krb5-admin-server      1.20.1-6ubuntu2.2      arm64      MIT Kerberos master server (kadmind)
ii  krb5-config            2.7                    all        Configuration files for Kerberos Version 5
ii  krb5-kdc               1.20.1-6ubuntu2.2      arm64      MIT Kerberos key server (KDC)
ii  krb5-locales           1.20.1-6ubuntu2.2      all        internationalization support for MIT Kerberos
ii  krb5-user              1.20.1-6ubuntu2.2      arm64      basic programs to authenticate using MIT Kerberos
ii  libgssapi-krb5-2:arm64  1.20.1-6ubuntu2.2      arm64      MIT Kerberos runtime libraries - krb5 GSS-API Mechanism
ii  libkrb5-3:arm64        1.20.1-6ubuntu2.2      arm64      MIT Kerberos runtime libraries
ii  libkrb5support0:arm64  1.20.1-6ubuntu2.2      arm64      MIT Kerberos runtime libraries - Support library
```

On crée la base de données

```
root@kerberos:/# ls -l /var/lib/krb5kdc/
total 16
-rw----- 1 root root 8192 Jan 24 13:59 principal
-rw----- 1 root root 8192 Jan 24 13:59 principal.kadm5
-rw----- 1 root root  0 Jan 24 13:59 principal.kadm5.lock
-rw----- 1 root root  0 Jan 24 13:59 principal.ok
root@kerberos:/#
```

Ensuite on démarre les services Kerberos

```
root@kerberos:/# krb5kdc
kadmind
[ps aux | grep krb5
root      160  0.0  0.0 10700  3772 ?        Ss   14:01   0:00 krb5kdc
root      172  0.0  0.0 10700  3724 ?        Ss   14:05   0:00 krb5kdc
root      176  0.0  0.0  3124  1388 pts/0    S+   14:05   0:00 grep --color=auto krb5
root@kerberos:/#
```

Ajout d'un utilisateur à Kerberos

Voici la commande que nous avons utilisé pour ajouter un utilisateur à Kerberos

`kadmin.local -q "addprinc testuser"`

```
root@kerberos:/# kadmin.local -q "addprinc testuser"
Authenticating as principal root/admin@EXAMPLE.COM with password.
No policy specified for testuser@EXAMPLE.COM; defaulting to no policy
Enter password for principal "testuser@EXAMPLE.COM":
Re-enter password for principal "testuser@EXAMPLE.COM":
Principal "testuser@EXAMPLE.COM" created.
root@kerberos:/#
```

Test de l'utilisateur Kerberos

On vérifie que l'utilisateur a bien été créé

```
[root@kerberos:/# kinit testuser@EXAMPLE.COM
[Password for testuser@EXAMPLE.COM:
root@kerberos:/#
```

Je vérifie que le ticket a bien été attribué à l'utilisateur

```
[root@kerberos:/# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: testuser@EXAMPLE.COM

Valid starting    Expires          Service principal
01/24/25 14:30:24 01/25/25 00:30:24  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        renew until 01/31/25 14:30:21
root@kerberos:/#
```

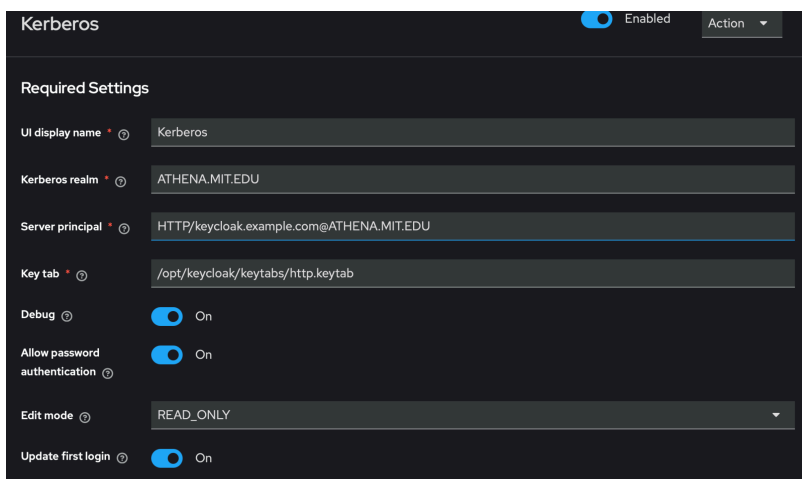
Intégration Kerberos a keycloak

On crée le principal

```
[root@kerberos:/# kadmin.local -q "addprinc -randkey HTTP/keycloak.example.com@EXAMPLE.COM"
Authenticating as principal testuser/admin@EXAMPLE.COM with password.
No policy specified for HTTP/keycloak.example.com@EXAMPLE.COM; defaulting to no policy
Principal "HTTP/keycloak.example.com@EXAMPLE.COM" created.
root@kerberos:/#
```

On crée le keytab

```
root@kerberos:/# kadmin.local -q "ktadd -k /etc/keycloak/http.keytab HTTP/keycloak.example.com@EXAMPLE.COM"
Authenticating as principal testuser/admin@EXAMPLE.COM with password.
kadmin.local: Key table file '/etc/keycloak/http.keytab' not found while adding key to keytab
root@kerberos:/#
```



The screenshot shows the 'Kerberos' configuration page in Keycloak. At the top right, there is a toggle switch labeled 'Enabled' which is turned on, and an 'Action' dropdown menu. Below this, the 'Required Settings' section contains several fields: 'UI display name' is set to 'Kerberos'; 'Kerberos realm' is set to 'ATHENA.MIT.EDU'; 'Server principal' is set to 'HTTP/keycloak.example.com@ATHENA.MIT.EDU'; 'Key tab' is set to '/opt/keycloak/keytabs/http.keytab'. There are also three toggle switches: 'Debug' (On), 'Allow password authentication' (On), and 'Update first login' (On). Finally, the 'Edit mode' dropdown is set to 'READ_ONLY'.

7 Sécurisation des API

Pour cette partie on a cree un nouveau Client (my-api) et user (apiuser).
On a fait une requete de connection normale pour voir si ca marche:

[illegible]

Vu que la requete marche, on va creer un serveur API pour pouvoir se connecter depuis.
On a ecrit le programme app.js:

```

(kali@kali)~[~/keycloak-login-api] All Forums < Kali NetHunter < Exploit-DB < Google Hacking DB < OffSec
$ cat app.js
const express = require('express');
const axios = require('axios');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());

const KEYCLOAK_URL = 'http://localhost:8080/realms/TestRealm/protocol/openid-connect/token';
const CLIENT_ID = 'my-api';
const CLIENT_SECRET = 'BzKGIH0R6cOAClu5X1gg8LYkw3peUfwf'; // Replace with your Keycloak client secret

// Login endpoint
app.post('/api/login', async (req, res) => {
  const { username, password } = req.body;

  if (!username || !password) {
    return res.status(400).json({ error: 'Username and password are required' });
  }

  try {
    // Send login request to Keycloak
    const response = await axios.post(
      KEYCLOAK_URL,
      new URLSearchParams({
        grant_type: 'password',
        client_id: CLIENT_ID,
        client_secret: CLIENT_SECRET,
        username,
        password,
      }),
      {
        headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
      }
    );

    // Respond with the access token
    res.status(200).json({
      access_token: response.data.access_token,
      refresh_token: response.data.refresh_token,
      expires_in: response.data.expires_in,
    });
  } catch (error) {
    // Handle errors from Keycloak
    if (error.response && error.response.data) {

```

app.js:

```
const express = require('express');
const axios = require('axios');
const bodyParser = require('body-parser');
```

```
const app = express();
app.use(bodyParser.json());

const KEYCLOAK_URL =
'http://localhost:8080/realms/TestRealm/protocol/openid-connect/token';
const CLIENT_ID = 'my-api';
const CLIENT_SECRET = 'BzKGIHOR6cOAClu5X1gg8lYkw3peUfwf'; // Replace with your
Keycloak client secret

// Login endpoint
app.post('/api/login', async (req, res) => {
  const { username, password } = req.body;

  if (!username || !password) {
    return res.status(400).json({ error: 'Username and password are required' });
  }

  try {
    // Send login request to Keycloak
    const response = await axios.post(
      KEYCLOAK_URL,
      new URLSearchParams({
        grant_type: 'password',
        client_id: CLIENT_ID,
        client_secret: CLIENT_SECRET,
        username,
        password,
      }),
      {
        headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
      }
    );

    // Respond with the access token
    res.status(200).json({
      access_token: response.data.access_token,
      refresh_token: response.data.refresh_token,
      expires_in: response.data.expires_in,
    });
  } catch (error) {
    // Handle errors from Keycloak
    if (error.response && error.response.data) {
      return res.status(error.response.status).json(error.response.data);
    }
    res.status(500).json({ error: 'Internal server error' });
  }
});
```



```
// Start the server
const PORT = 3000;
app.listen(PORT, () => {
  console.log(`API running at http://localhost:${PORT}`);
});
```

Puis on a lance le programme et l'API et en marche:

```
(kali㉿kali)-[~/keycloak-login-api]
$ node app.js
API running at http://localhost:3000
```

On a reessaye a faire une requete de l'API pour tester et voir si on a le meme resultat qu'avant:

[illegible]

L'API marche bien.

Dans une vraie situation, pour sécuriser encore plus cette API, la requête originale vers Keycloak ne devrait pas être publique et ne devrait être accessible qu'au serveur API. Les utilisateurs devront obligatoirement passer par l'API pour effectuer leur connexion. De plus, l'API doit inclure des mécanismes de vérification et de filtrage des données envoyées par les utilisateurs afin de prévenir toute tentative d'injection ou autre type d'attaque malveillante.