

TP : déployer Keycloak, activer MFA, configurer OAuth2/OIDC

Objectif : monter un IdP **Keycloak** local, forcer **MFA (TOTP/WebAuthn)** et tester un **login OIDC** "Authorization Code + PKCE".

Pré-requis

1. Docker / Docker Compose.
2. Un navigateur, Postman (ou curl) et un smartphone avec une app TOTP (Google Authenticator / Authy / etc.).
3. Ports libres 8080 (Keycloak), 3000 (appli de test si besoin).

Ressource

<https://www.keycloak.org/getting-started/getting-started-docker>

Objectifs pédagogiques

À l'issue de ce TP, l'étudiant devra être capable de :

1. Déployer une instance de Keycloak en local.
2. Comprendre la notion de *realm*, *client* et *utilisateur* dans Keycloak.
3. Configurer un mécanisme de **Multi-Factor Authentication (MFA)** via TOTP.
4. Mettre en place et tester un **flux OAuth2/OIDC (Authorization Code + PKCE)**.
5. Comprendre la différence entre **accès via token** et **session via cookies**.

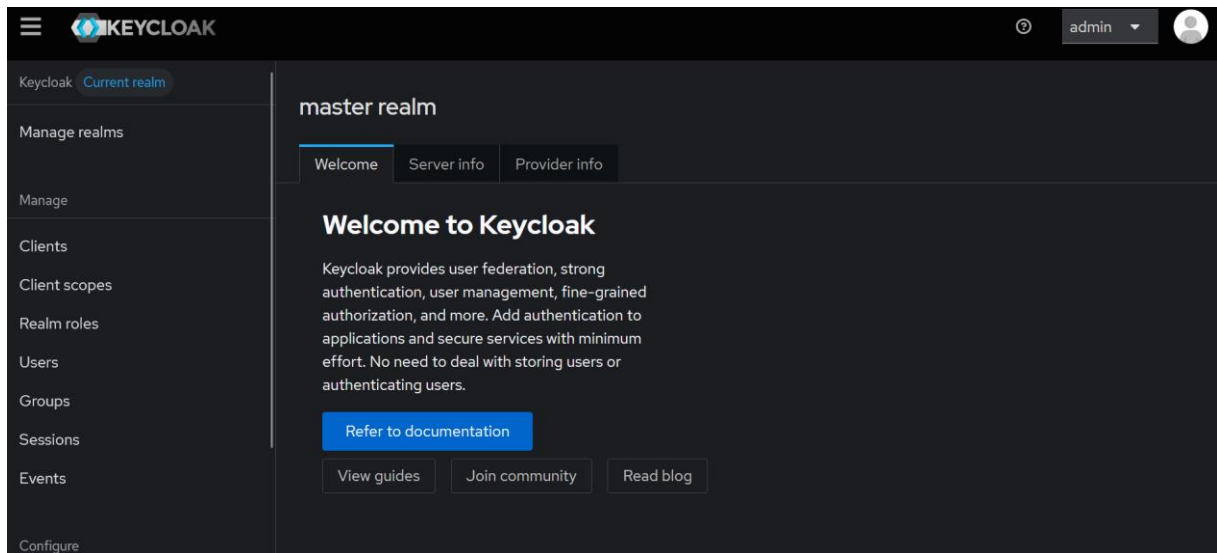
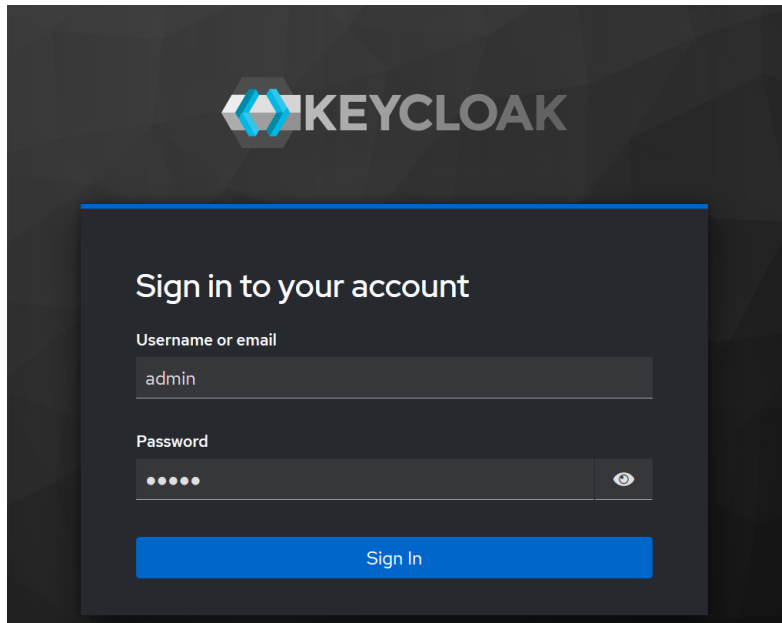
Étapes du TP

1. Déploiement rapide (Docker)

À partir d'un terminal, entrez la commande suivante pour démarrer Keycloak :

```
docker run -p 127.0.0.1:8080:8080 -e
KC_BOOTSTRAP_ADMIN_USERNAME=admin -e
KC_BOOTSTRAP_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:26.3.3
start-dev
```

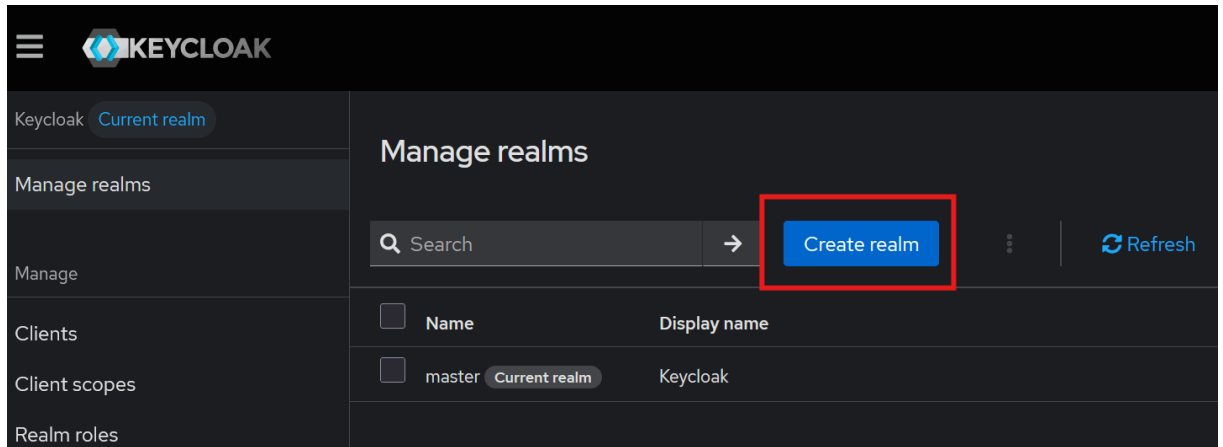
Cette commande démarre **Keycloak** exposé sur le port local **8080** et crée un utilisateur administrateur initial avec le nom d'utilisateur **admin** et mot de passe **admin**.



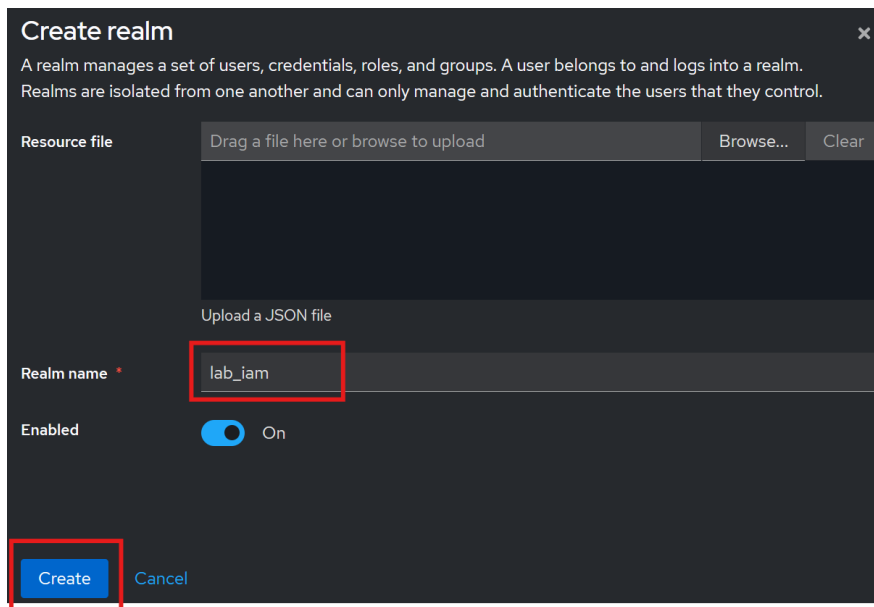
2. Créer le realm et un utilisateur

Un **realm** à Keycloak est un espace d'authentification indépendant équivalent à un domaine de sécurité. Chaque domaine permet à un administrateur de créer des groupes isolés d'applications et d'utilisateurs.

1. Dans la console, cliquer sur **Manage realms > Create Realm**.



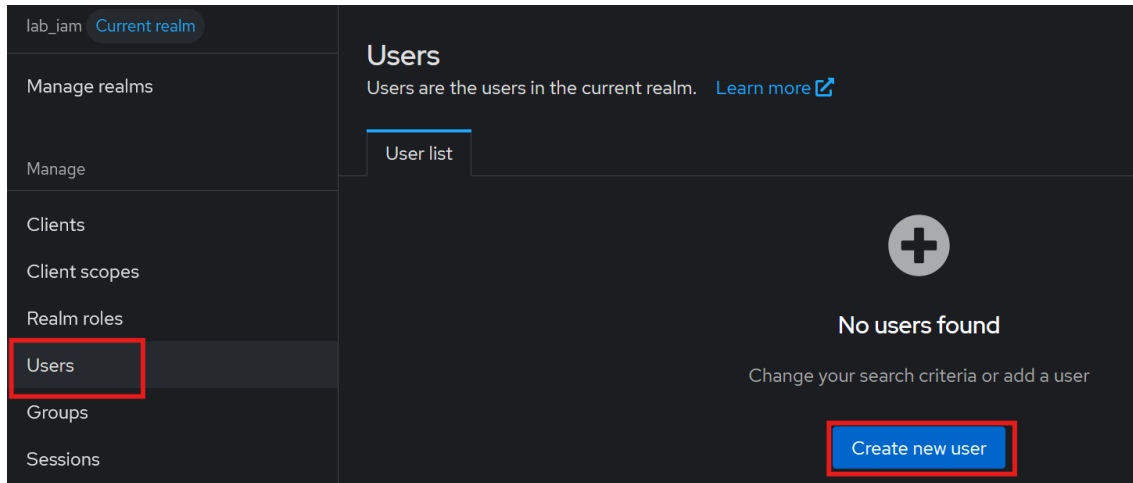
2. Nommer le realm : lab-iam
3. Valider.



3. Création d'un utilisateur

Un utilisateur représente une identité humaine ou applicative. Initialement, le realm n'a pas d'utilisateurs. Utiliser ces étapes pour créer un utilisateur :

1. Aller dans **Users** → **Add user**.



2. Renseigner :

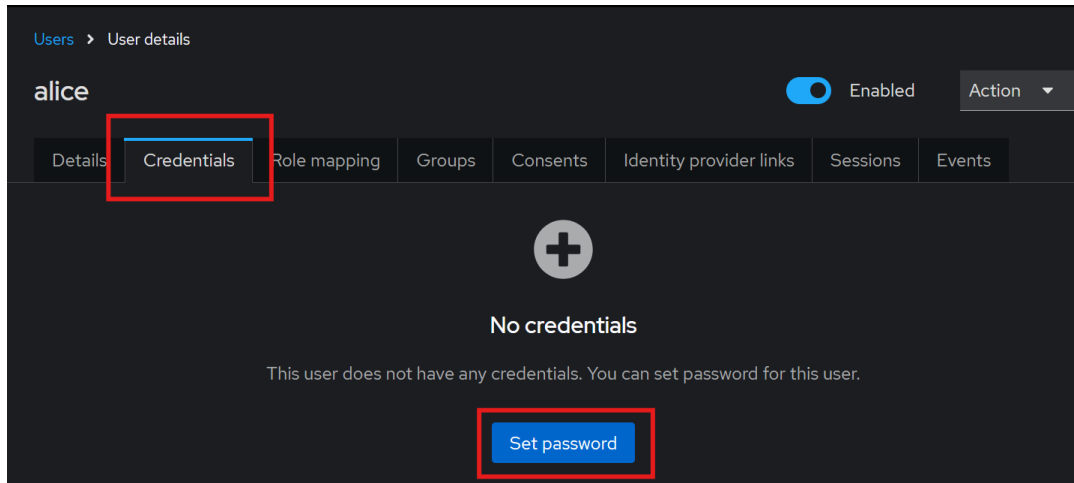
- a. **Username** : alice
- b. **Email** : alice@example.com
- c. **First Name** : Alice
- d. **Last Name** : User
- e. **Email Verified** : ON (permet d'éviter la validation par email).

3. Cliquer sur **Save**.

The screenshot shows the 'Create new user' form. At the top, there is a toggle for 'Email verified' which is currently 'On'. Below this is the 'General' section with several input fields: 'Username' (containing 'alice'), 'Email' (empty), 'First name' (containing 'Alice'), and 'Last name' (containing 'User'). There is also a 'Groups' section with a 'Join Groups' button. At the bottom of the form, there are two buttons: 'Create' and 'Cancel'.

4. Aller dans l'onglet **Credentials** → définir un mot de passe (ex. Passw0rd!).

- a. Cocher **Temporary** = **OFF** pour éviter de forcer le changement.



Set password for alice

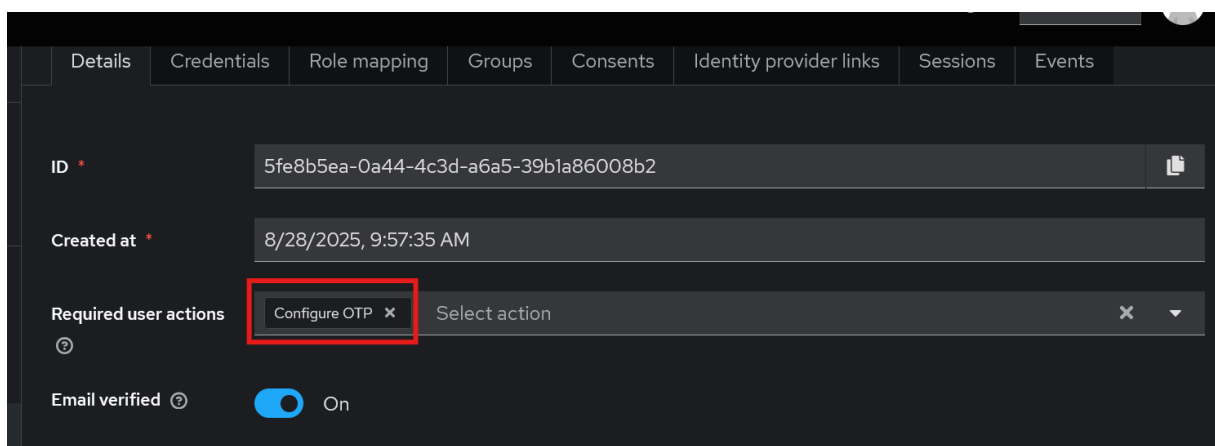
Password * password

Password confirmation * password

Temporary ② ☐ Off

Save Cancel

5. Dans **Required Actions**, ajouter **Configure OTP** → cela obligera l'utilisateur à configurer un second facteur lors de sa première connexion.

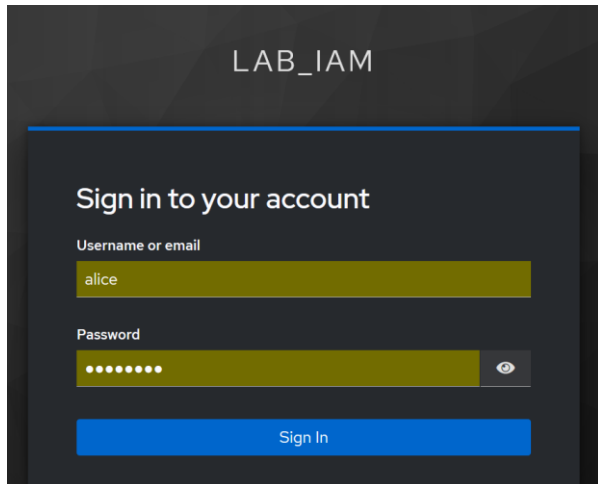


4. Configuration du MFA (OTP)

Le MFA renforce la sécurité en demandant **quelque chose que l'utilisateur connaît** (mot de passe) **et quelque chose qu'il possède** (application générant un code OTP).

Keycloak prend en charge le **TOTP (Time-based One-Time Password)** : un code à usage unique valable 30 secondes.

1. Dans **Authentication → Required Actions**, vérifier que **Configure OTP** est activé.
2. Lors de la première connexion, l'utilisateur verra un QR code à scanner avec son application OTP.
3. Aller dans la page de connexion du domaine "lab_iam" :
http://localhost:8080/realms/lab_iam/account

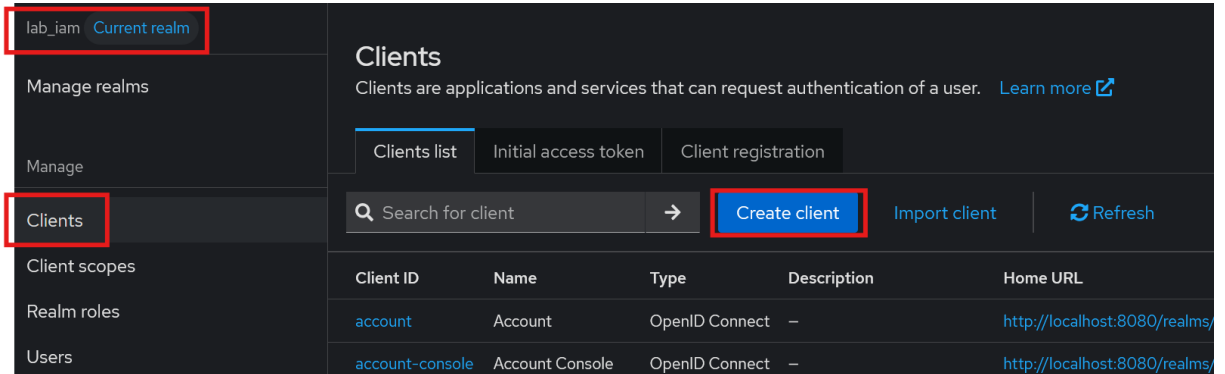


4. Comme l'action "Configure OTP" est activée, Keycloak affiche une page avec un **QR code**.
 5. Sur le smartphone, ouvrir une application OTP (Google Authenticator, Microsoft Authenticator, Authy, etc.).
 6. Ajouter un nouveau compte → **scanner le QR code** affiché.
 7. L'application génère un code à 6 chiffres, renouvelé toutes les 30 secondes.
 8. Saisir le **code OTP affiché** dans l'application.
 9. Cliquer sur **Submit**.
 10. Keycloak enregistre la configuration MFA pour Alice.
 11. Alice est maintenant connectée au portail utilisateur.
 12. Déconnecter Alice puis se reconnecter.
 13. Le processus d'authentification se déroule en deux étapes :
 - a. Mot de passe
 - b. Code OTP (nouveau code généré par l'application)
- La connexion est accordée uniquement si les **deux facteurs** sont corrects.

5. Créer un client **OIDC** "demo-spa"

Un client représente une application (SPA – Single-Page application), API, mobile, etc.) qui utilisera Keycloak pour l'authentification.

1. Aller dans **Clients** → **Create client**.



The screenshot shows the Keycloak Admin Console interface. On the left sidebar, the 'lab_iam' realm is selected, and the 'Clients' menu item is highlighted. The main panel displays the 'Clients' section, which includes tabs for 'Clients list', 'Initial access token', and 'Client registration'. The 'Clients list' tab is active, showing a search bar and a 'Create client' button. Below the search bar, a table lists existing clients:

Client ID	Name	Type	Description	Home URL
account	Account	OpenID Connect	–	http://localhost:8080/realms/
account-console	Account Console	OpenID Connect	–	http://localhost:8080/realms/

2. Renseigner :

- **Client ID** : demo-spa
- **Client type** : Public (pas de secret partagé, adapté aux SPA).
Ou **Client Authentication** : Off
- **Authentication flow** : Standard flow activé (correspond au flux **Authorization Code**).
- Cliquer sur **Next**.
- Dans **Valid redirect URIs**, ajouter : `https://www.keycloak.org/app/*`.
- Dans **Web origins**, ajouter : `“+”` pour autoriser toutes les origines (en pratique, restreindre).
- Sauvegarder.

The first screenshot shows the 'General settings' tab for a new client. The 'Client type' is set to 'OpenID Connect'. The 'Client ID' is 'demo_spa' and the 'Name' is 'client_spa'. The 'Always display in UI' toggle is turned off. The 'Next' button is highlighted in blue.

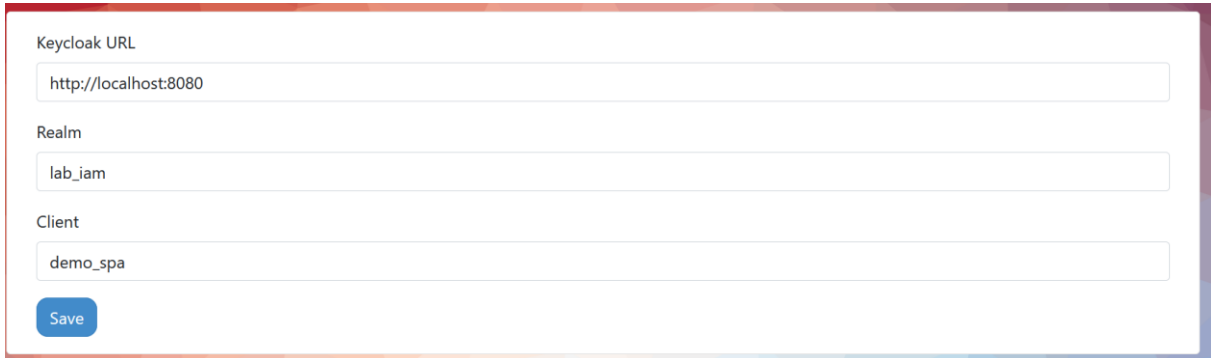
The second screenshot shows the 'Capability config' tab. 'Client authentication' and 'Authorization' are both turned off. Under 'Authentication flow', 'Standard flow' is selected. Other options like 'Direct access grants', 'Implicit flow', 'Standard Token Exchange', 'OAuth 2.0 Device Authorization Grant', and 'OIDC CIBA Grant' are unselected. The 'PKCE Method' is set to 'Choose...'. The 'Next' button is highlighted in blue.

The third screenshot shows the 'Access settings' tab. It contains fields for 'Root URL', 'Home URL', 'Valid redirect URIs' (with a value of 'https://www.keycloak.org/app/*'), 'Valid post logout redirect URIs', and 'Web origins'. Each field has a '+' icon to add more values and a '-' icon to remove them. The 'Add valid redirect URIs' and 'Add web origins' buttons are highlighted in blue.

6. Tester le login OIDC

Pour confirmer que le client a été créé avec succès, vous pouvez utiliser l'application de test SPA sur le [site web de Keycloak](https://www.keycloak.org/app/).

1. Ouvrir <https://www.keycloak.org/app/>.
2. Mettre à jour les champs **Realm** et **Client**.
3. Cliquez sur **Enregistrer** pour utiliser la configuration du TP.



A screenshot of a web form for configuring Keycloak. It has three input fields: 'Keycloak URL' with the value 'http://localhost:8080', 'Realm' with the value 'lab_iam', and 'Client' with the value 'demo_spa'. Below these fields is a blue 'Save' button.

4. Cliquez sur **Se connecter** pour authentifier cette application en utilisant le serveur Keycloak que vous avez lancé plus tôt.
5. Vous êtes redirigé vers la **page de login Keycloak**.
 - Saisir les identifiants de l'utilisateur *alice*.
 - Comme la MFA est activée, Keycloak demande aussi le **code OTP** généré par l'application Authenticator.

Une fois validé → retour automatique sur l'application de test.



A screenshot of the Keycloak login page. At the top, there are two buttons: 'Sign out' (blue) and 'Clear config' (grey). Below them, a white box displays the text 'Hello, Alice User'.

7. Tester et analyser les tokens OIDC (avec l'app de test Keycloak)

Jusqu'ici, l'application de test <https://www.keycloak.org/app/> affichait seulement **"Hello Alice"** après connexion. Mais pour bien comprendre OIDC, il est important de voir **les tokens échangés** entre Keycloak et l'application.

Voici comment procéder :

1. Se reconnecter avec Alice

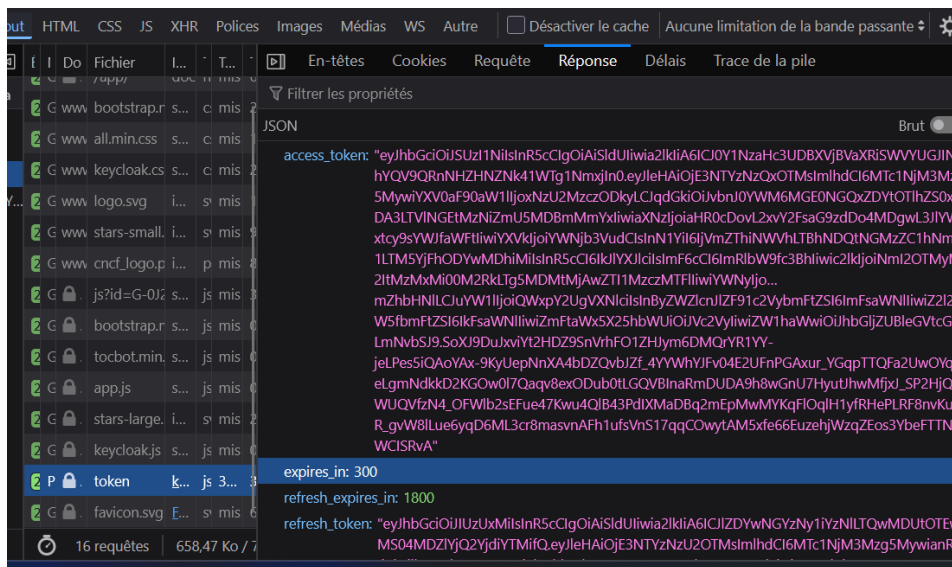
1. Ouvrir <https://www.keycloak.org/app/>
2. Cliquer sur **Change configuration**.
3. Entrer :
 - **Keycloak URL** : <http://localhost:8080>
 - **Realm** : lab-iam
 - **Client** : demo_spa
 - Cliquer sur **Save** puis se reconnecter avec l'utilisateur *Alice*.
 - Vous arrivez sur la page Hello Alice

2. Ouvrir les outils de développement (DevTools)

- Dans le navigateur, appuyer sur F12 (ou clic droit → Inspecter).
- Aller dans l'onglet Network (Réseau).
- Cocher l'option Preserve log (conserver le log) pour voir toutes les requêtes.

3. Observer l'échange de tokens

- Rafraîchir la page ou se reconnecter si nécessaire.
- Repérer une requête de type POST vers : `/realms/lab-iam/protocol/openid-connect/token`
- Cliquer sur cette requête → onglet **Response**.
- Vous verrez un JSON contenant :



4. Analyser les tokens

- **access_token** :
 - Sert à appeler une API protégée.
 - Contient les rôles, permissions, etc.
- **id_token** :
 - Sert à identifier l'utilisateur auprès de l'application.
 - Contient les claims d'identité (preferred_username, email, name).
- **refresh_token** :
 - Sert à obtenir un nouvel access_token sans redemander login + mot de passe.

Pour décoder et visualiser le contenu, copier un token JWT (comme `id_token`) et le coller sur <https://jwt.io>

8. Questions :

1. Observation des tokens

- a. Quels types de tokens avez-vous récupérés ? (citez-les et décrivez leur rôle : `access_token`, `id_token`, `refresh_token`).
- b. Quelle est la durée de vie de votre `access_token` ? Et celle de votre `refresh_token` ?

2. Analyse du contenu du `id_token`

- a. Quel est le `sub` (subject) de votre utilisateur Alice ?
- b. Quels attributs d'identité y trouvez-vous (`preferred_username`, `email`, etc.) ?
- c. Avez-vous vu des informations relatives aux rôles/permissions ?

3. Analyse du contenu du `access_token`

- a. Quelles informations de sécurité (scopes, rôles, `realm_access`, etc.) sont incluses ?
- b. Expliquez en quoi ce token peut être utilisé par une API.

4. Sécurité et gestion

- a. Pourquoi l'application a-t-elle besoin d'un `refresh_token` ?
- b. Quelles seraient les conséquences si un attaquant volait un `refresh_token` ?

5. Comparaison pratique

- a. En quoi l'`id_token` est-il différent de l'`access_token` ? Donnez un exemple concret de leur usage.