Gagner du temps et déporter votre authentification avec Keycloak

Ronan Morel
07/11/2019

Toujours long et fastidieux, la mise en place d'une authentification au sein d'une application est l'une des préoccupation majeure à la mise en place d'un projet. S'il l'on veut bien faire les choses, elle a toujours un coût important à l'initialisation du projet alors que c'est une chose que l'on a faite 1, 2 voire 10 fois, sans compter la maintenance sur le long terme et surement les failles de sécurité que l'on omet.

Alors pourquoi ne pas utiliser une solution déjà existante et maintenue par d'autres que nous ?

Dans cet article, je vais parler de la gestion de l'authentification via un SSO (Sigle Sign On), Keycloak.

Qu'est-ce que Keycloak ?

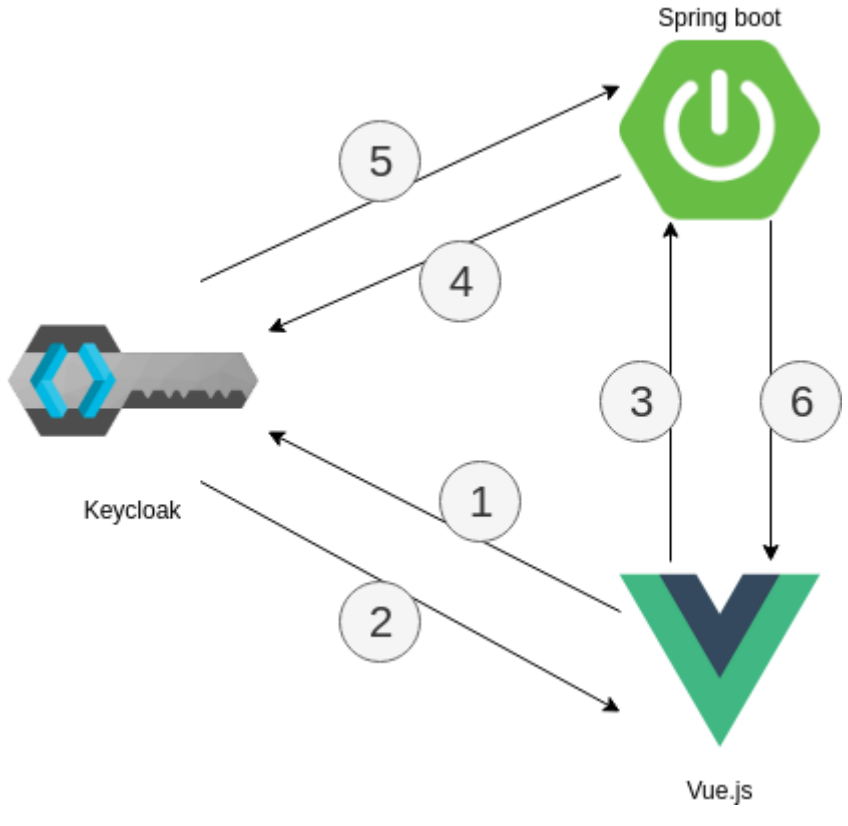
Keycloak est un outil de gestion des identités et d'authentification. Il est porté par Red Hat et est open-source. Cet outil est utilisé pour le SSO, c'est-à-dire, que pour plusieurs services, on délègue l'identification par Keycloak. Pour cela, il implémente les standards OAuth 2.0 avec les protocoles sous-jacents **OpenID** ou **SAML**.

Pour info, d'autres identity provider existent comme Google, GitHub, Facebook ou encore Twitter.

Un peu de contexte

Pour illustrer cette mise en place de l'authentification par Keycloak, nous allons d'abord parler des services à développer. Nous avons deux services : Une API qui ouvre accès aux données que l'on veut distribuer et un front-end qui les affichent. Nous avons aussi une notion de données public/private, et une notion de données user/admin.

Architecture de cette mini application



Pour cette application, le découpage est le suivant :

- Un backend en Kotlin avec Spring 5 / Spring boot 2.0
- Un frontend en TypeScript avec Vue.js 2.0
- Et Keycloak 6.0.1 comme service d'authentification

Le procédé sera le suivant :

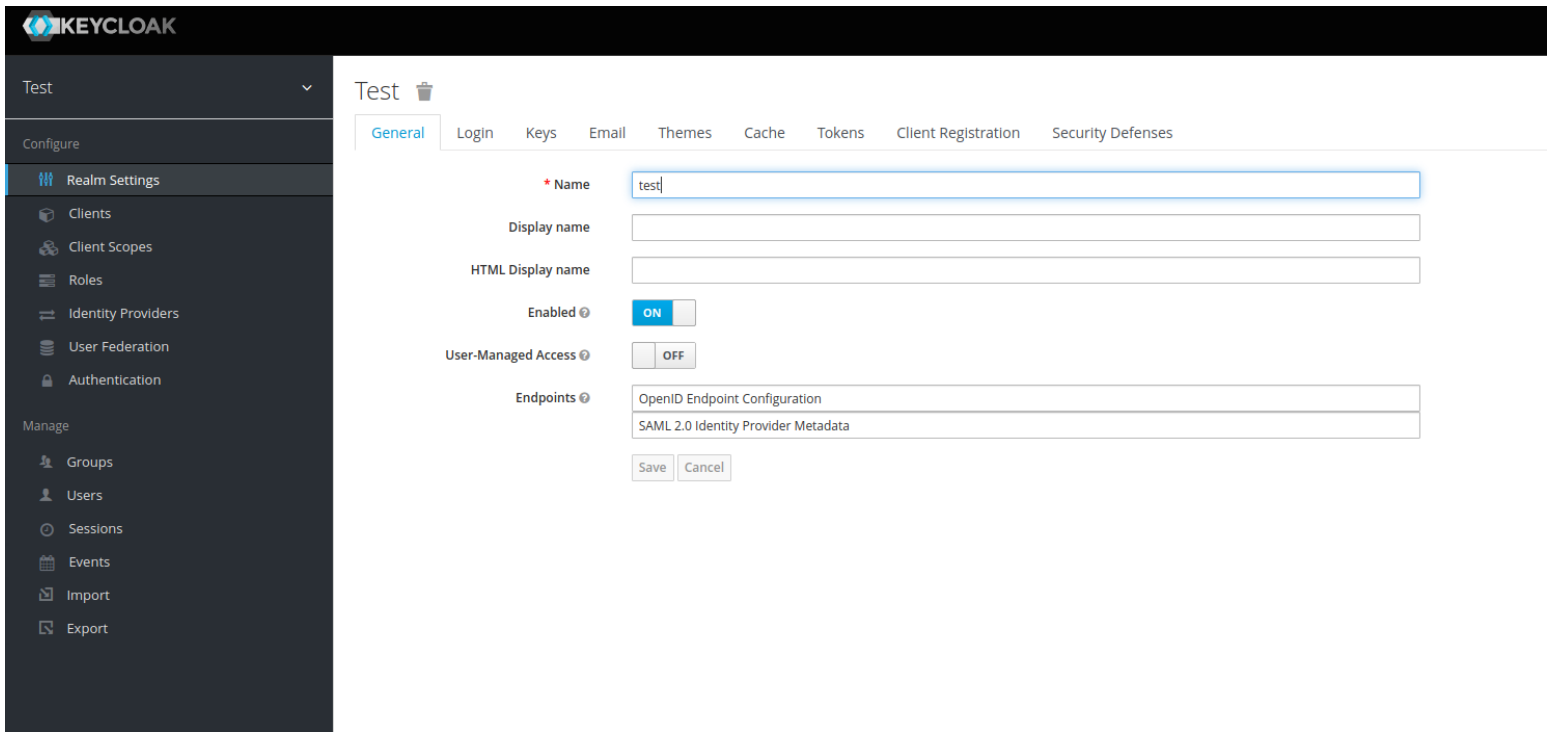
1. L'utilisateur arrive sur l'application front. Il peut intérargir avec plusieurs buttons et l'IHM lui donne les informations associées. Il peut aussi s'authentifier.
2. Après être bien identifier, Keycloak va envoyer au front, une clé d'accès à l'application appelée **access_token**
3. L'utilisateur va pour utiliser l'application front, avec les différentes page et les différents droits qu'il a (admin / user). Le front va aussi faire des appels REST pour récupérer les informations nécessaires au bon fonctionnement de l'application.
4. L'API, en recevant la requête REST du front-end, va vérifier s'il y a un access_token et demander à Keycloak si le token est bien valide.
5. Keycloak va confirmer ou non la bonne validité de l'access_token.
6. Et l'API, construira, en fonctionne des droits que dispose l'utilisateur, la response REST à la requête demandée.

Après le théorique, attaquons le technique maintenant.

Configuration de Keycloak

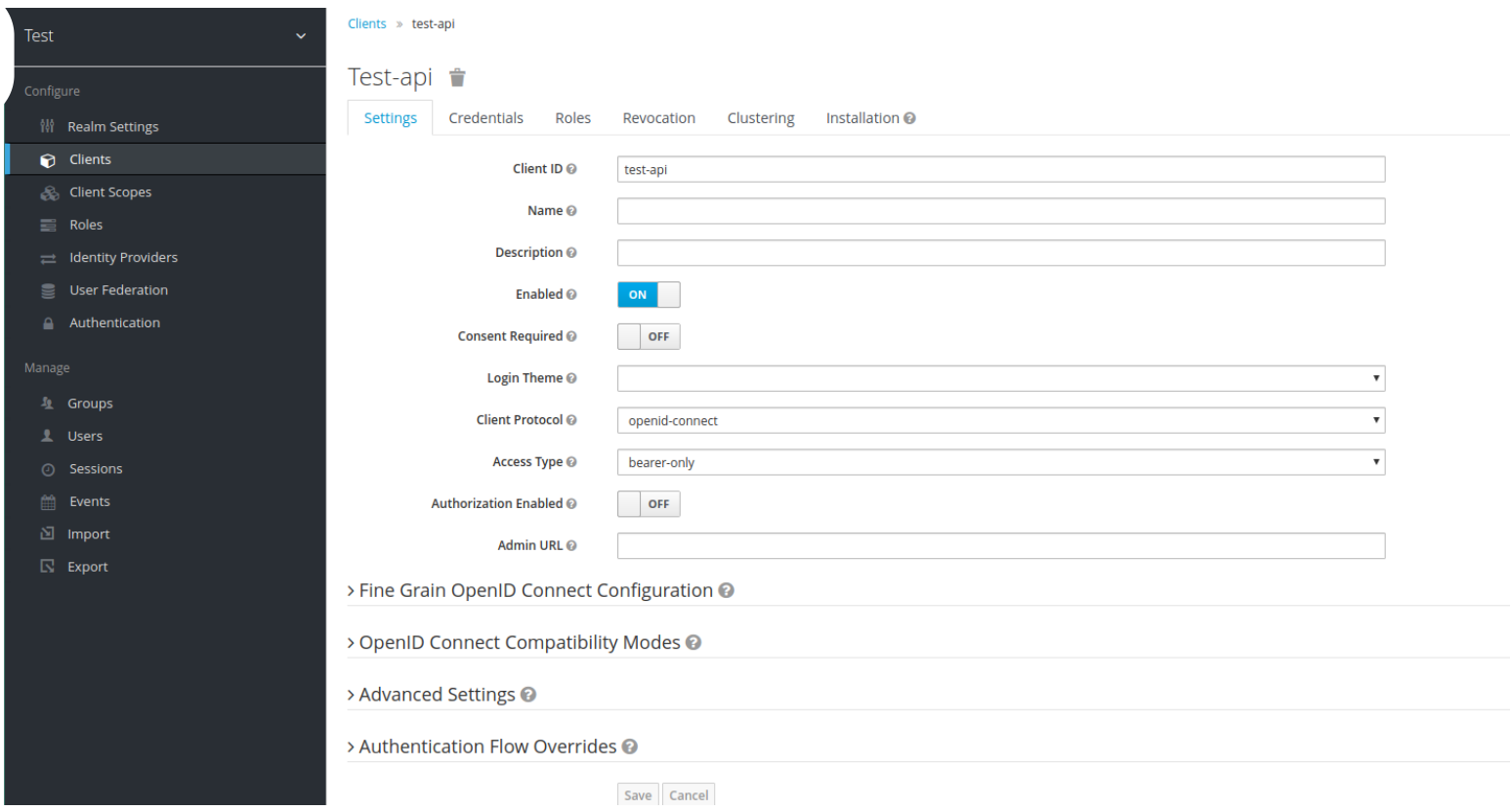
Création d'un **realm**

Créer un realm **test** dans Keycloak



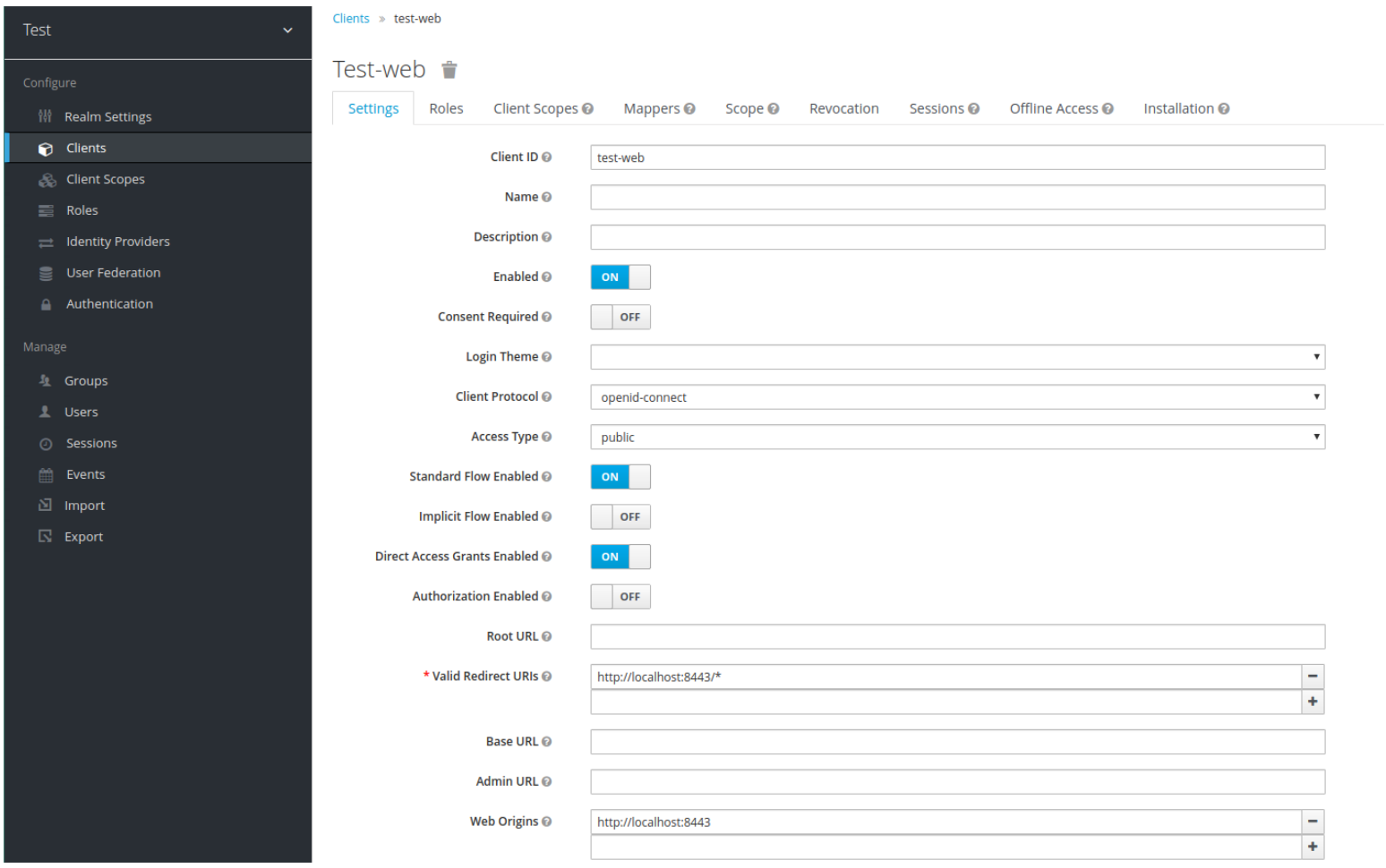
Création d’un client côté API

Créer un client nommé **test-api**, avec le protocole OpenID et avec comme type **bearer-only**



Création d’un client côté Web

Créer un client nommé **test-web**, avec le protocole OpenID et avec comme type **public**



Configuration côté API

Ajout des dépendances dans Gradle

```
dependencies {
...
implementation("org.keycloak:keycloak-spring-boot-starter:$versionKeycloak") implementation("org.keycloak.bom:keycloak-adapter-bom:$versionKeycloak") implementation("org.keycloak:keycloak-spring-boot-adapter:$versionKeycloak")
...
}
```

Ajout des configurations dans le fichier de ressource application.yml

```
server:
  port: $
keycloak:
  auth     er-url: $
    l       ed: none realm: $
    s       $
  bearer-only: true
  credentials:
    secret: $
security:
  basic:
    enabled: false

spring:
  main:
    allow-bean-definition-overriding: true
```

Ajout de la classe de configuration dans spring boot

https://gist.github.com/ronronan/50e3677e379113b44540bb3ee6e61d76#file-keycloakconfigurationadapter-kt

Configuration côté Front-end

Installation de la dépendance

```
npm install --save @dsb-norge/vue-keycloak-js
```

Ajout de la configuration dans le fichier main.ts

https://gist.github.com/ronronan/65589cd5f75d1dc31dfad3238adb7316#file-main-ts

Dans App.vue et TopBar.vue, utiliser l'objet $keycloak (example de App.vue)

- Pour vérifier authentication de l'utilisateur

  https://gist.github.com/ronronan/1462415a6a06cabf34b7c3dd7e3f6265#file-app-ts

- Pour se déconnecter :

  https://gist.github.com/ronronan/ea026f169dbdbd6d7b06a30d54f7ec31#file-logout-ts

Vidéo de présentation

Pour illustrer cette article, j'ai réaliser une vidéo où je mets en oeuvre ce qui est présenté ci-dessus par l'intermédiaire un pseudo live-coding.

Liens du projet

- Front-end : https://github.com/Slickteam/vue-spring-keycloak-front
- API : https://github.com/Slickteam/vue-spring-keycloak-back

Conclusion

Keycloak est un outil simple à mettre en place. Pour la mise en place rapide d'une authentication sur une application, je pense qu'utiliser
'o net de gagner des jours de développement et donc de l'argent, et aussi de s'abstraire pas mal de problème et surtout de sa
te t donc de pouvoir se concentrer sur l'application métier en tant que tel.

Sources

- https://github.com/dsb-norge/vue-keycloak-js
- https://github.com/keycloak/keycloak-documentation/blob/master/securing_apps/topics/oidc/java/spring-security-adapter.adoc