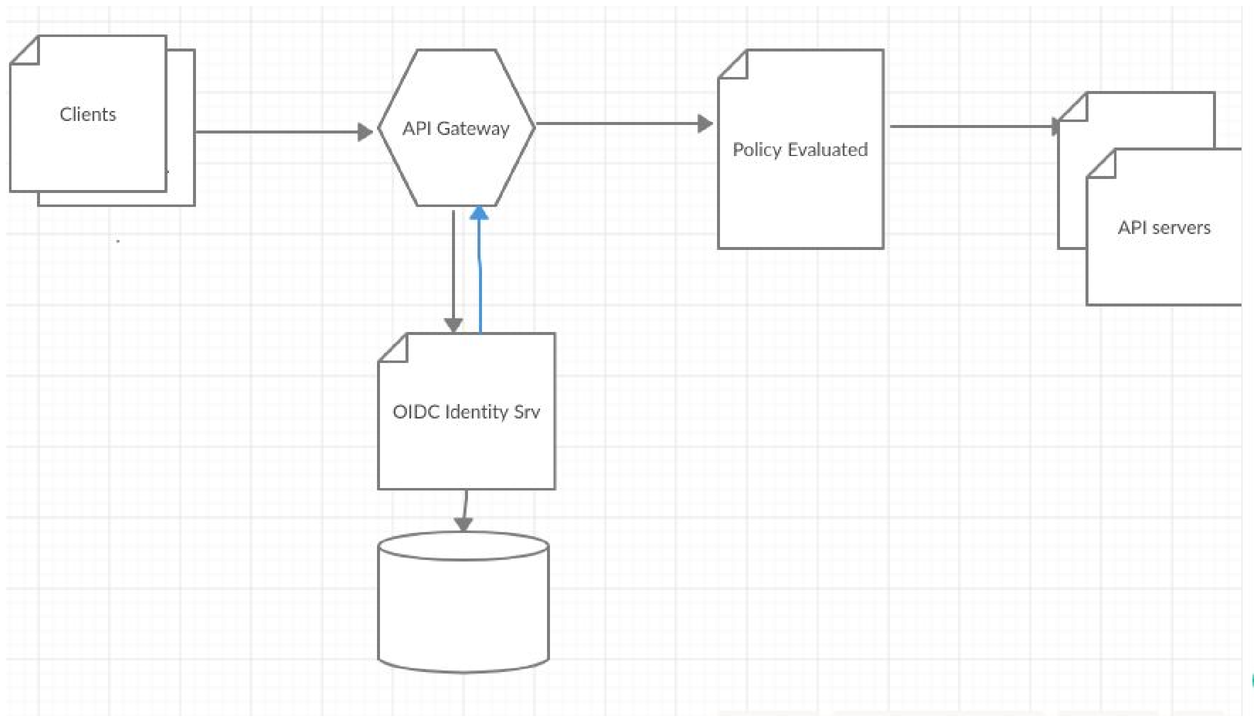
Case study: CS20180505
Name: Navneet Kumar

High level system architecture:



Use Case: I am using a setup with Keycloak as Identity Provider, Spring Cloud Gateway as API Gateway and multiple Microservices(API web servers). I can receive a JWT via my Gateway (redirecting to Keycloak) via `http://localhost:8050/auth/realms/dev/protocol/openid-connect/token`.
I can use the JWT to access a resource directly located at the Keycloak server (e.g. `http://localhost:8080/auth/admin/realms/dev/users`). But when I want to use the Gateway to relay me to the same resource `http://localhost:8050/auth/admin/realms/dev/users`) I get the Keycloak Login form as a response..

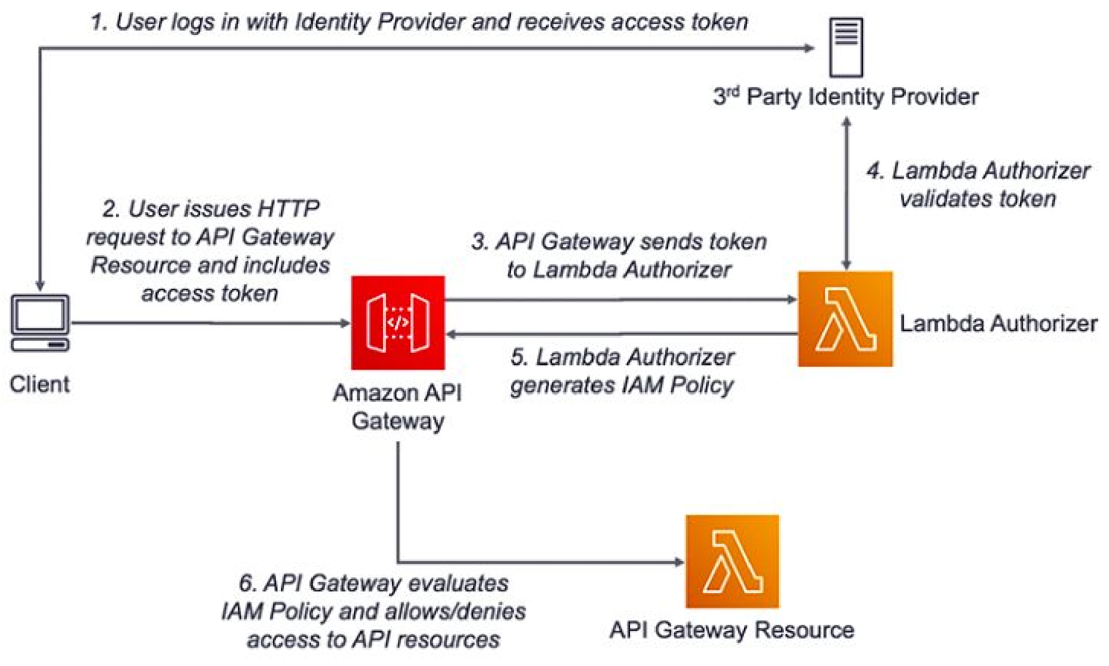
1. **List of APIs and rationale behind the design:** The API's servers(Resources) shall be as follows:
   /atms
   /branches
   /personal-current-accounts
   /business-current-accounts
   /unsecured-sme-loans
   /insurance
   /mortgage

/credit
/debit

2. **The implementation of one non functional requirement:**

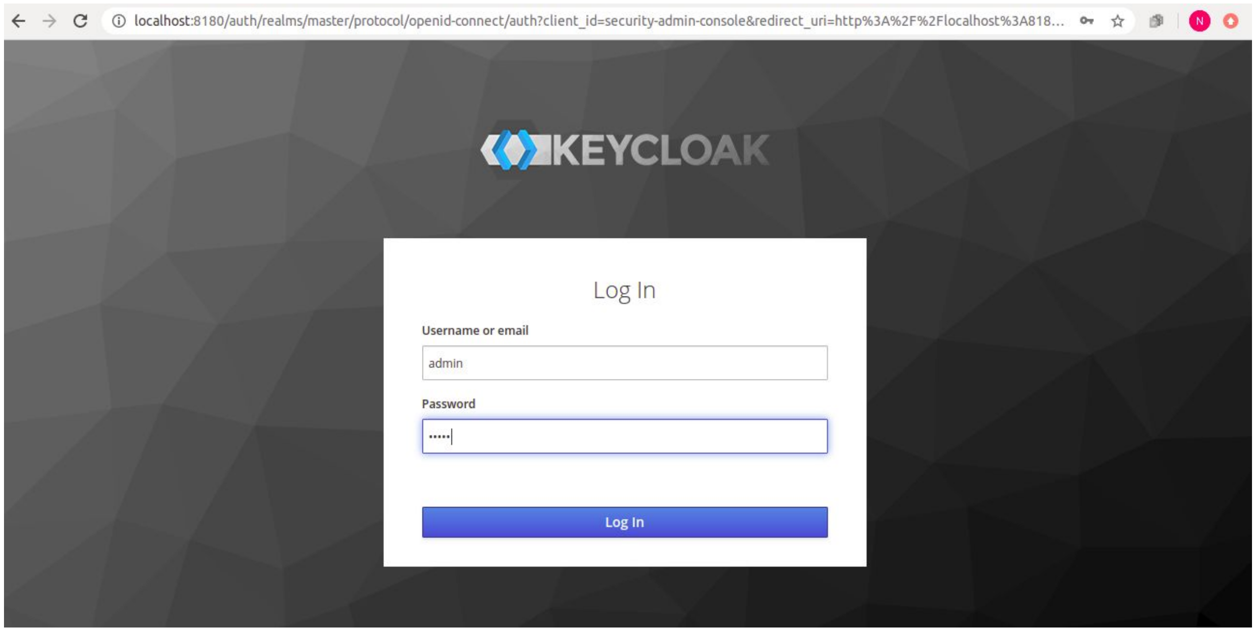
Prerequisite:

1) Keycloak distribution to be running in each cluster across DC
   Keycloak server to authenticate a user for all apps clients. After a successful login, the application will receive an identity token and an access token. The identity token contains information about the user such as username, email, and other profile information. The access token is digitally signed by the realm and contains access information (like user role mappings) that the application can use to determine what resources the user is allowed to access on the application.

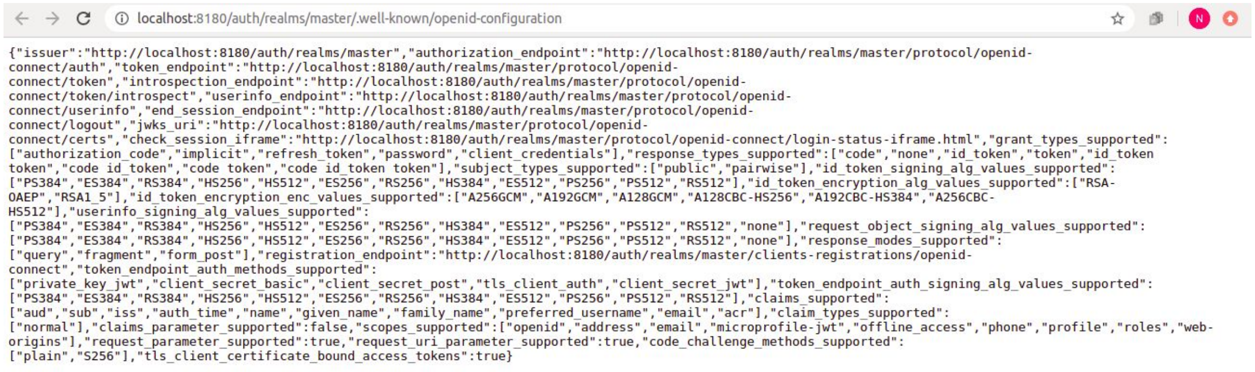

1. User logs in with Identity Provider and receives access token
3rd Party Identity Provider
4. Lambda Authorizer validates token
2. User issues HTTP request to API Gateway Resource and includes access token
3. API Gateway sends token to Lambda Authorizer
Lambda Authorizer
Client
Amazon API Gateway
5. Lambda Authorizer generates IAM Policy
6. API Gateway evaluates IAM Policy and allows/denies access to API resources
API Gateway Resource

Steps:

1. Create an initial admin username and password in keycloak. Login using the admin credentials


localhost:8180/auth/realms/master/protocol/openid-connect/auth?client_id=security-admin-console&redirect_uri=http%3A%2F%2Flocalhost%3A818...

2. Keycloak configure these endpoints


localhost:8180/auth/realms/master/.well-known/openid-configuration

```
{"issuer":"http://localhost:8180/auth/realms/master","authorization_endpoint":"http://localhost:8180/auth/realms/master/protocol/openid-
connect/auth","token_endpoint":"http://localhost:8180/auth/realms/master/protocol/openid-
connect/token","introspection_endpoint":"http://localhost:8180/auth/realms/master/protocol/openid-
connect/token/introspect","userinfo_endpoint":"http://localhost:8180/auth/realms/master/protocol/openid-
connect/userinfo","end_session_endpoint":"http://localhost:8180/auth/realms/master/protocol/openid-
connect/logout","jwks_uri":"http://localhost:8180/auth/realms/master/protocol/openid-
connect/certs","check_session_iframe":"http://localhost:8180/auth/realms/master/protocol/openid-connect/login-status-iframe.html","grant_types_supported":
["authorization_code","implicit","refresh_token","password","client_credentials"],"response_types_supported":["code","none","id_token","token","id_token
token","code id_token","code token","code id_token token"],"subject_types_supported":["public","pairwise"],"id_token_signing_alg_values_supported":
["PS384","ES384","RS384","HS256","HS512","ES256","RS256","HS384","ES512","PS256","PS512","RS512"],"id_token_encryption_alg_values_supported":["RSA-
OAEP","RSA1_5"],"id_token_encryption_enc_values_supported":["A256GCM","A192GCM","A128GCM","A128CBC-HS256","A192CBC-HS384","A256CBC-
HS512"],"userinfo_signing_alg_values_supported":
["PS384","ES384","RS384","HS256","HS512","ES256","RS256","HS384","ES512","PS256","PS512","RS512","none"],"request_object_signing_alg_values_supported":
["PS384","ES384","RS384","HS256","HS512","ES256","RS256","HS384","ES512","PS256","PS512","RS512","none"],"response_modes_supported":
["query","fragment","form_post"],"registration_endpoint":"http://localhost:8180/auth/realms/master/clients-registrations/openid-
connect","token_endpoint_auth_methods_supported":
["private_key_jwt","client_secret_basic","client_secret_post","tls_client_auth","client_secret_jwt"],"token_endpoint_auth_signing_alg_values_supported":
["PS384","ES384","RS384","HS256","HS512","ES256","RS256","HS384","ES512","PS256","PS512","RS512"],"claims_supported":
["aud","sub","iss","auth_time","name","given_name","family_name","preferred_username","email","acr"],"claim_types_supported":
["normal"],"claims_parameter_supported":false,"scopes_supported":["openid","address","email","microprofile-jwt","offline_access","phone","profile","roles","web-
origins"],"request_parameter_supported":true,"request_uri_parameter_supported":true,"code_challenge_methods_supported":
["plain","S256"],"tls_client_certificate_bound_access_tokens":true}
```

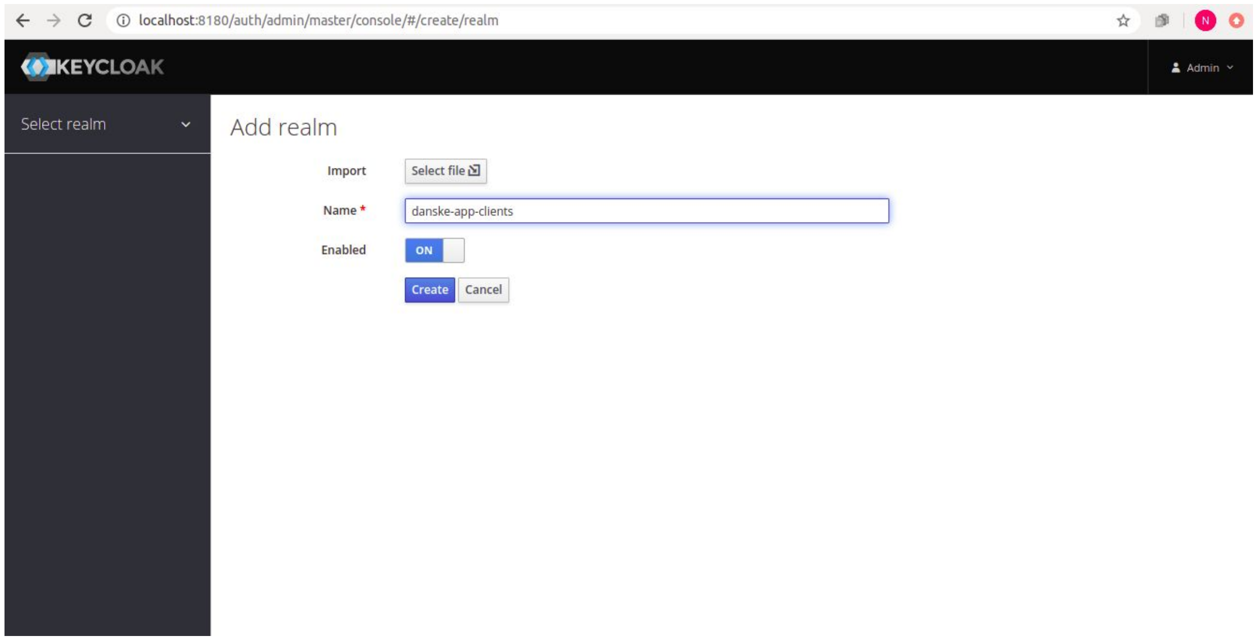
http://localhost:8180/auth/realms/master/protocol/openid-connect/auth
http://localhost:8180/auth/realms/master/protocol/openid-connect/token
http://localhost:8180/auth/realms/master/protocol/openid-connect/token/introspec
t
http://localhost:8180/auth/realms/master/protocol/openid-connect/userinfo
http://localhost:8180/auth/realms/master/protocol/openid-connect/certs
http://localhost:8180/auth/realms/master/protocol/openid-connect/logout
http://localhost:8180/auth/realms/master/clients-registrations/openid-connect

3. Health ping URL:

http://localhost:8180/auth/realms/SpringBootKeycloak/.well-known/openid-configuration

Creating a Realm

4. A successful login will take us to the console and open up the default Master realm for us. Here we'll focus on creating a custom realm. Let's navigate to the upper left upper corner to discover the **Add realm** button:



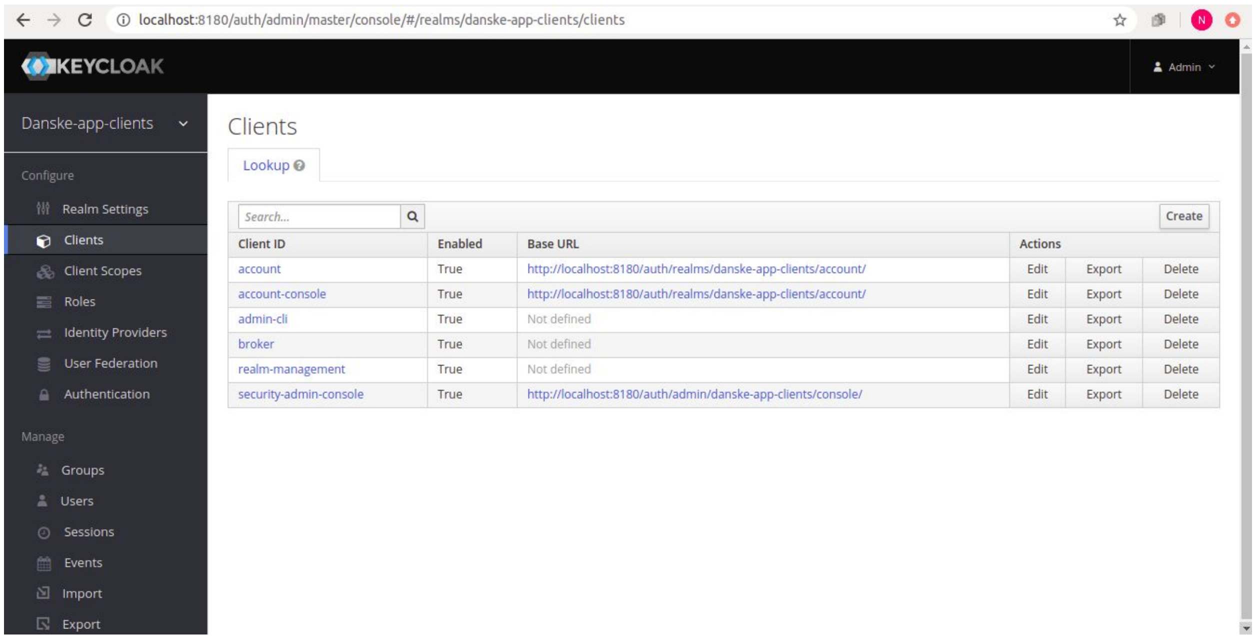
Name * danske-app-clients

Enabled ON

After clicking the Create button, a new realm will be created and we'll be redirected to it. All the operations in the next sections will be performed in this new danske-app-clients realm.

Creating a Client

Now we'll navigate to the Clients page. As we can see in the image below, **Keycloak comes with Clients that are already built-in**:

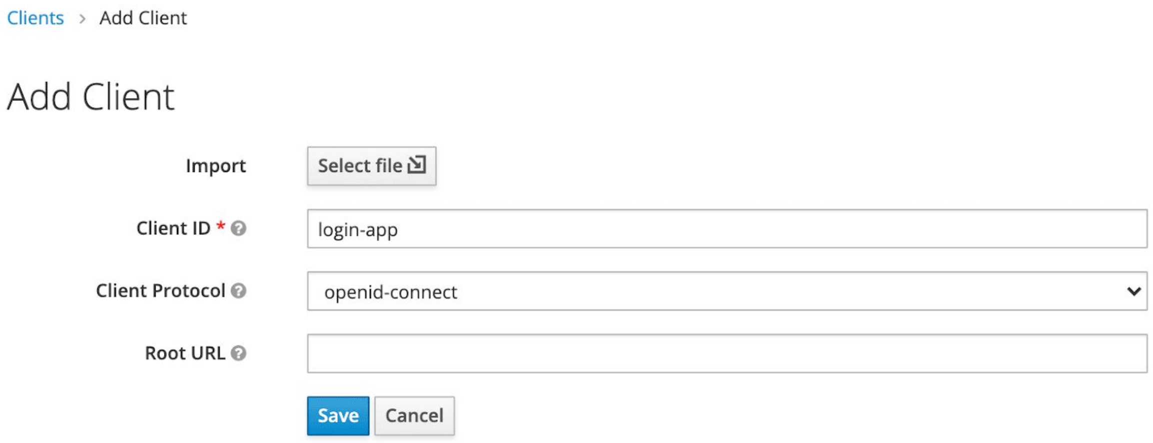


localhost:8180/auth/admin/master/console/#/realms/danske-app-clients/clients

Clients

Client ID	Enabled	Base URL	Actions		
account	True	http://localhost:8180/auth/realms/danske-app-clients/account/	Edit	Export	Delete
account-console	True	http://localhost:8180/auth/realms/danske-app-clients/account/	Edit	Export	Delete
admin-cli	True	Not defined	Edit	Export	Delete
broker	True	Not defined	Edit	Export	Delete
realm-management	True	Not defined	Edit	Export	Delete
security-admin-console	True	http://localhost:8180/auth/admin/danske-app-clients/console/	Edit	Export	Delete

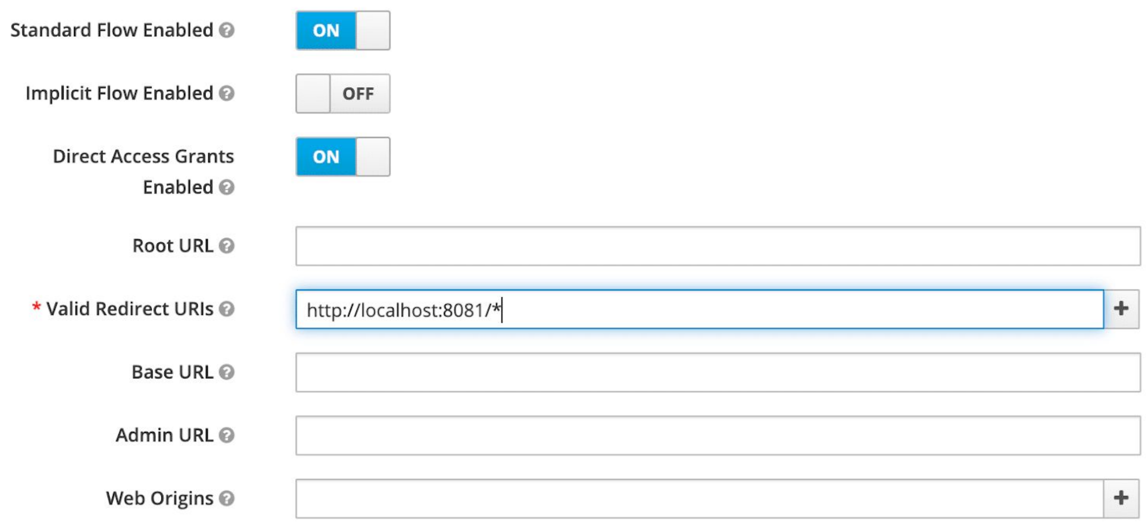
But we need to add a new client to our application, so we'll click Create. We'll call the new Client login-app:



Add Client

Import	
Client ID *	login-app
Client Protocol	openid-connect
Root URL	

In the next screen, for this tutorial, we'll be leaving all the defaults except the Valid Redirect URIs field. This field should contain the application URL(s) that will use this client for authentication:



Standard Flow Enabled: ON

Implicit Flow Enabled: OFF

Direct Access Grants Enabled: ON

Root URL:

* Valid Redirect URIs: http://localhost:8081/*

Base URL:

Admin URL:

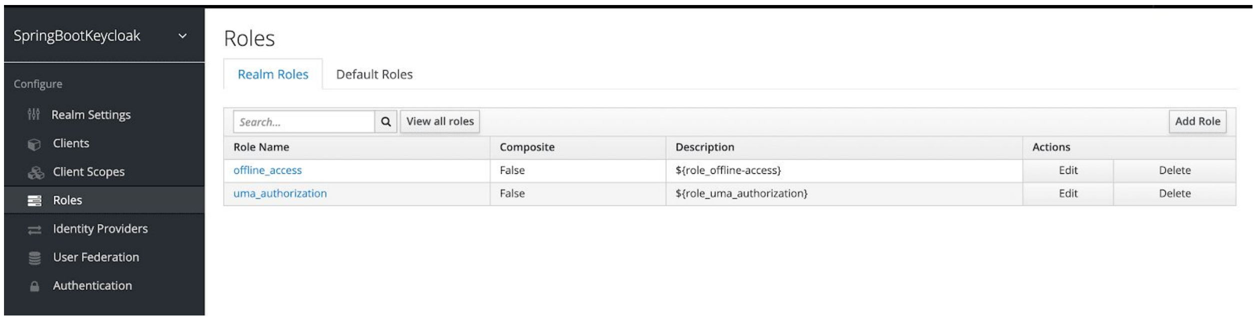
Web Origins:

Later on, we'll be creating a Spring Boot Application running at the port 8081 that'll use this client. Hence we've used a redirect URL of http://localhost:8081/* above.

Creating a Role and a User

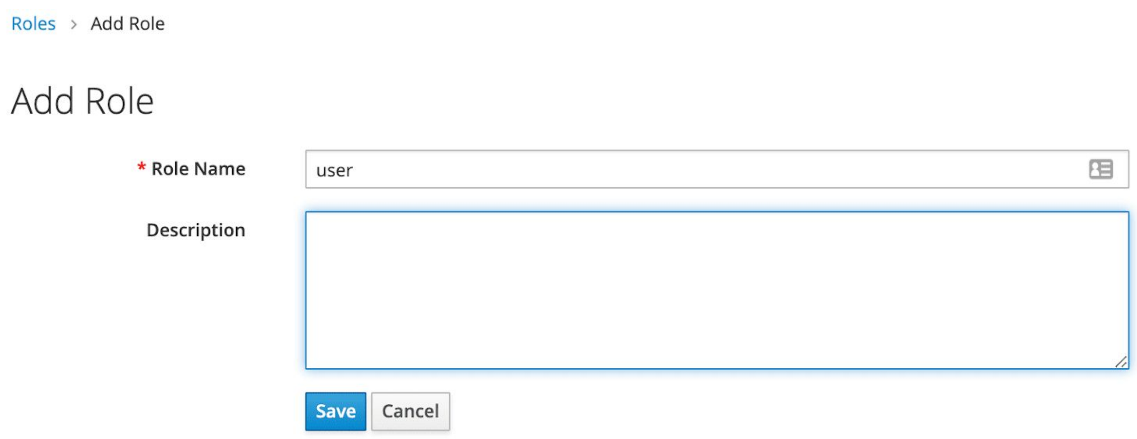
Keycloak uses Role-Based Access. Therefore, each user must have a role.

To do that, we need to navigate to the Roles page:



Role Name	Composite	Description	Actions	
offline_access	False	${role_offline-access}	Edit	Delete
uma_authorization	False	${role_uma_authorization}	Edit	Delete

Then, we'll add the user role:

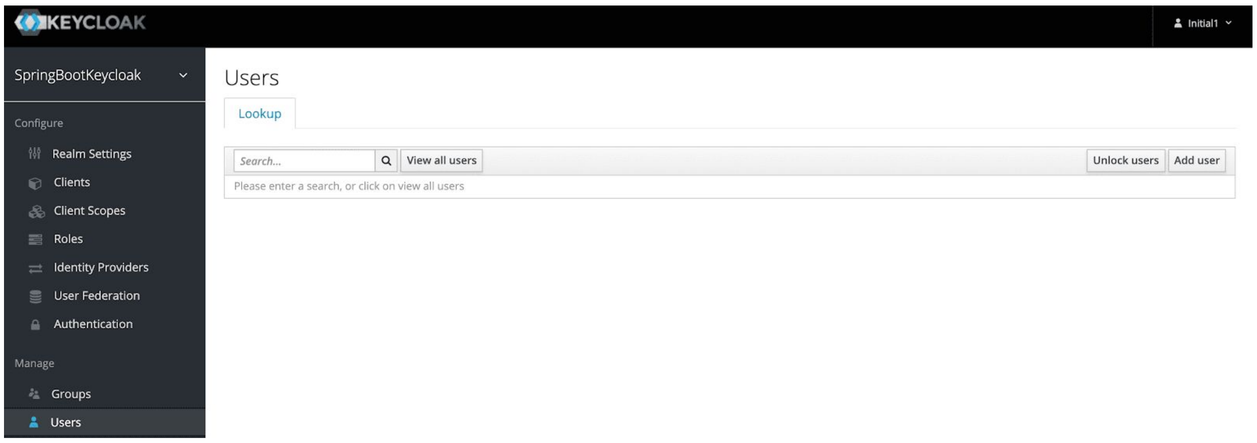

Roles > Add Role

Add Role

* Role Name: user

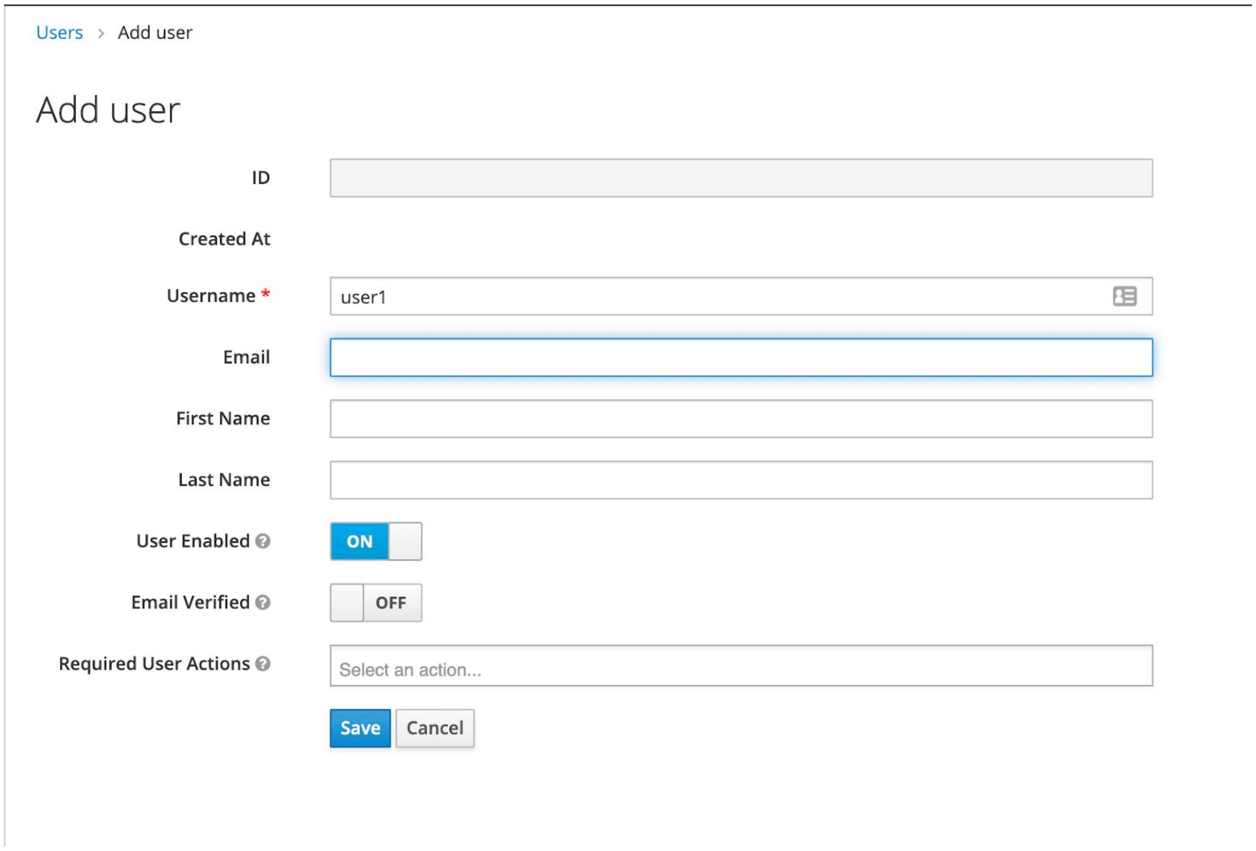
Description:

Now we've got a role that can be assigned to users, but there are no users yet. **So let's go the Users page and add one:**



Please enter a search, or click on view all users

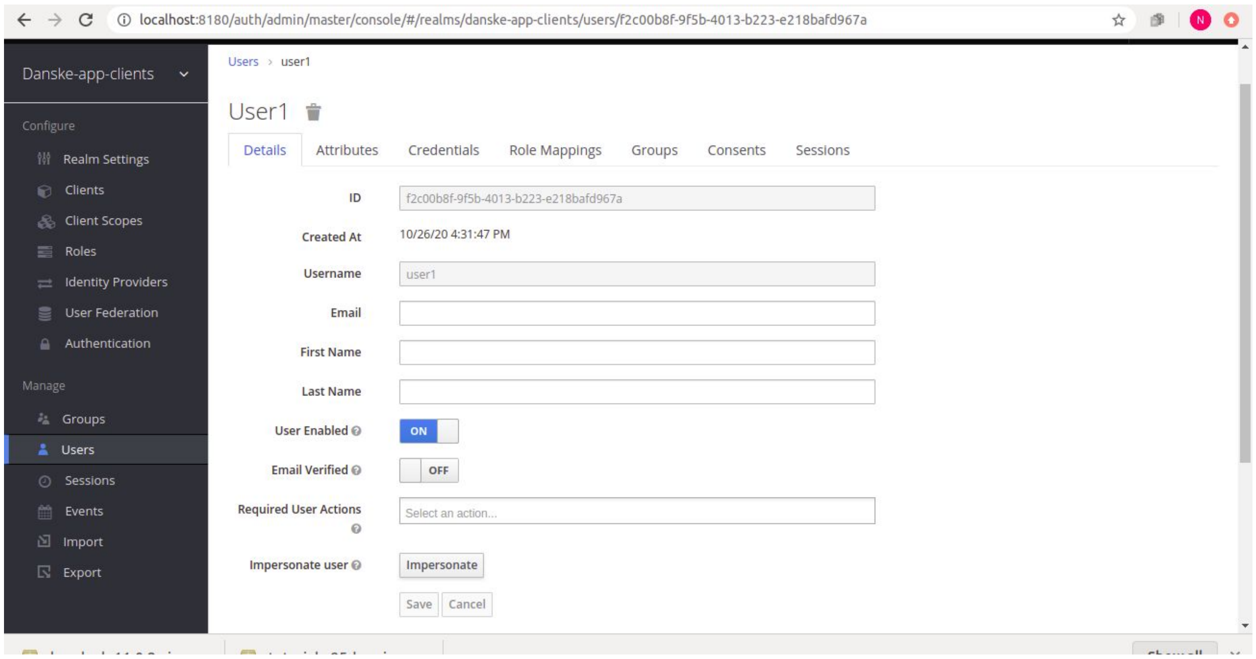
We'll add a user named user1:


Users > Add user

Add user

ID	
Created At	
Username *	user1
Email	
First Name	
Last Name	
User Enabled	ON
Email Verified	OFF
Required User Actions	Select an action...

Once the user is created, a page with its details will be displayed:

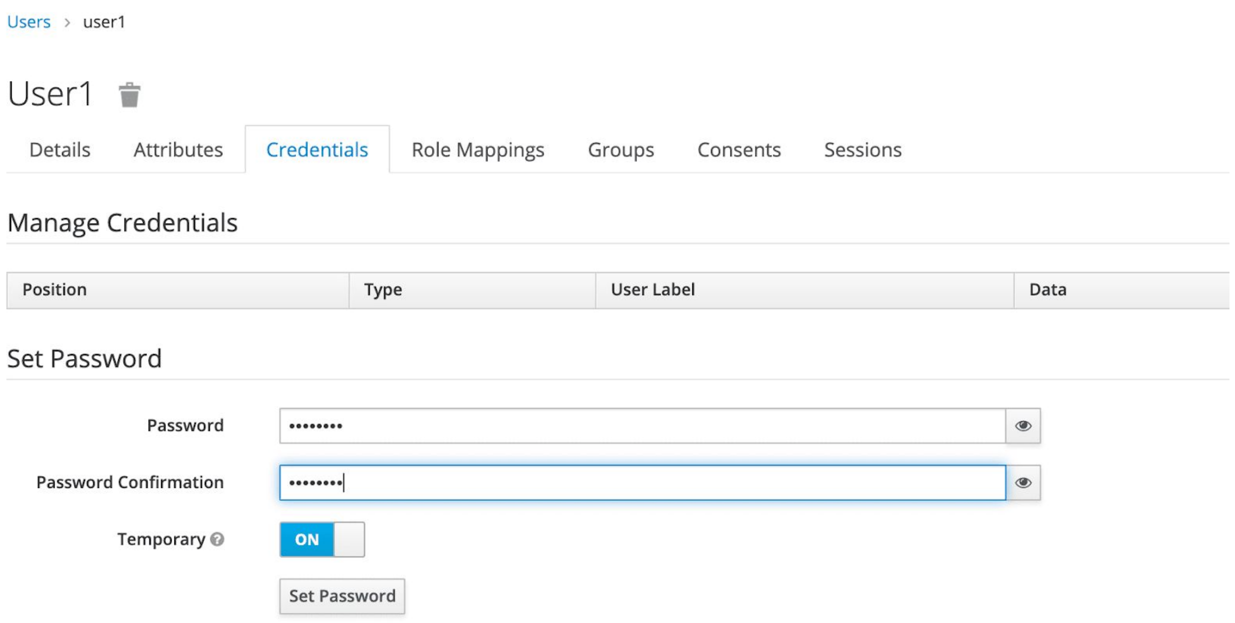

localhost:8180/auth/admin/master/console/#/realms/danske-app-clients/users/f2c00b8f-9f5b-4013-b223-e218bafd967a

Users > user1

User1

ID	f2c00b8f-9f5b-4013-b223-e218bafd967a
Created At	10/26/20 4:31:47 PM
Username	user1
Email	
First Name	
Last Name	
User Enabled	ON
Email Verified	OFF
Required User Actions	Select an action...
Impersonate user	Impersonate

We can now go to the Credentials tab. We'll be setting the initial password to Mig@jal_14


Users › user1

User1

Details | Attributes | Credentials | Role Mappings | Groups | Consents | Sessions

Manage Credentials

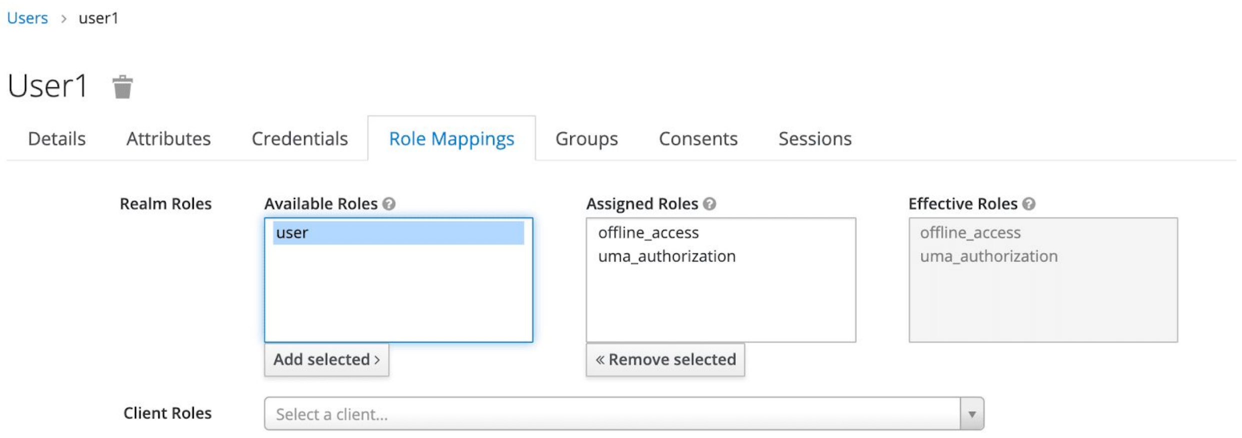
Position	Type	User Label	Data

Set Password

Password	••••••••
Password Confirmation	••••••••
Temporary	ON

Set Password

Finally, we'll navigate to the Role Mappings tab. We'll be assigning the user role to our user1:


Users › user1

User1

Details | Attributes | Credentials | Role Mappings | Groups | Consents | Sessions

Realm Roles

Available Roles	Assigned Roles	Effective Roles
user	offline_access	offline_access
	uma_authorization	uma_authorization

Add selected › | « Remove selected

Client Roles: Select a client...

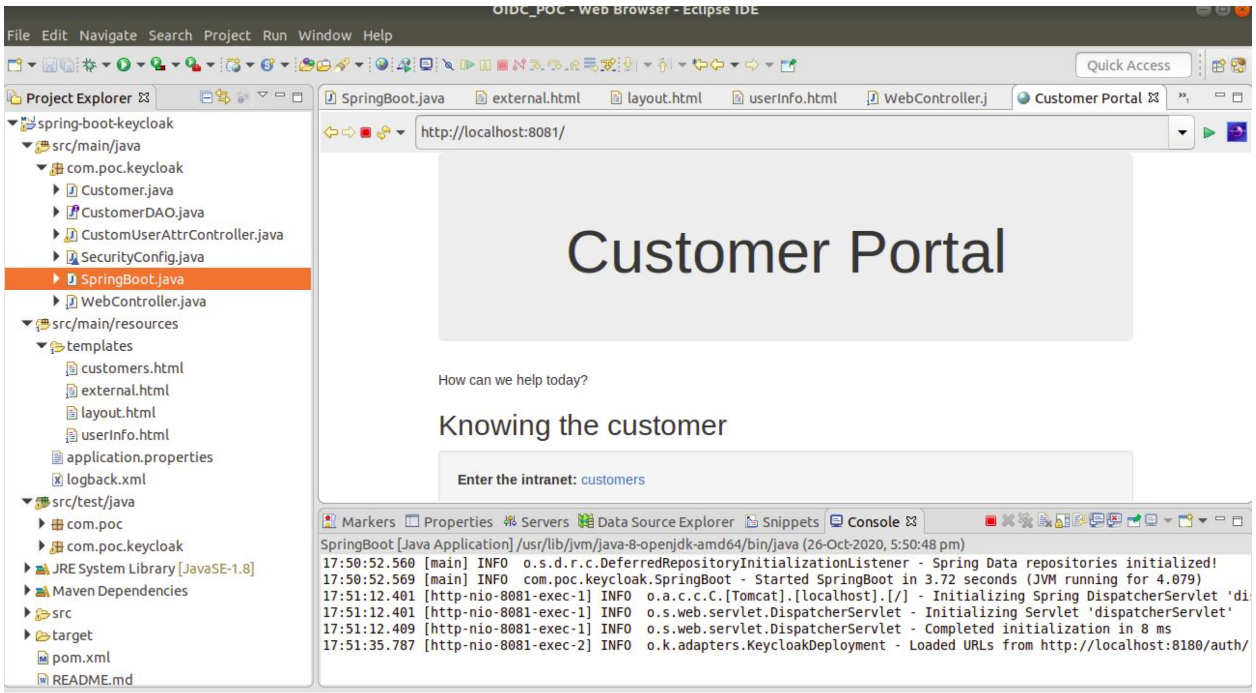
Click the Add selected button.

Application work flow:

1. mvn clean spring-boot:run

On visiting http://localhost:8081 we see:

Now we click customers to enter the intranet, which is the location of sensitive information.



Customer Portal

How can we help today?

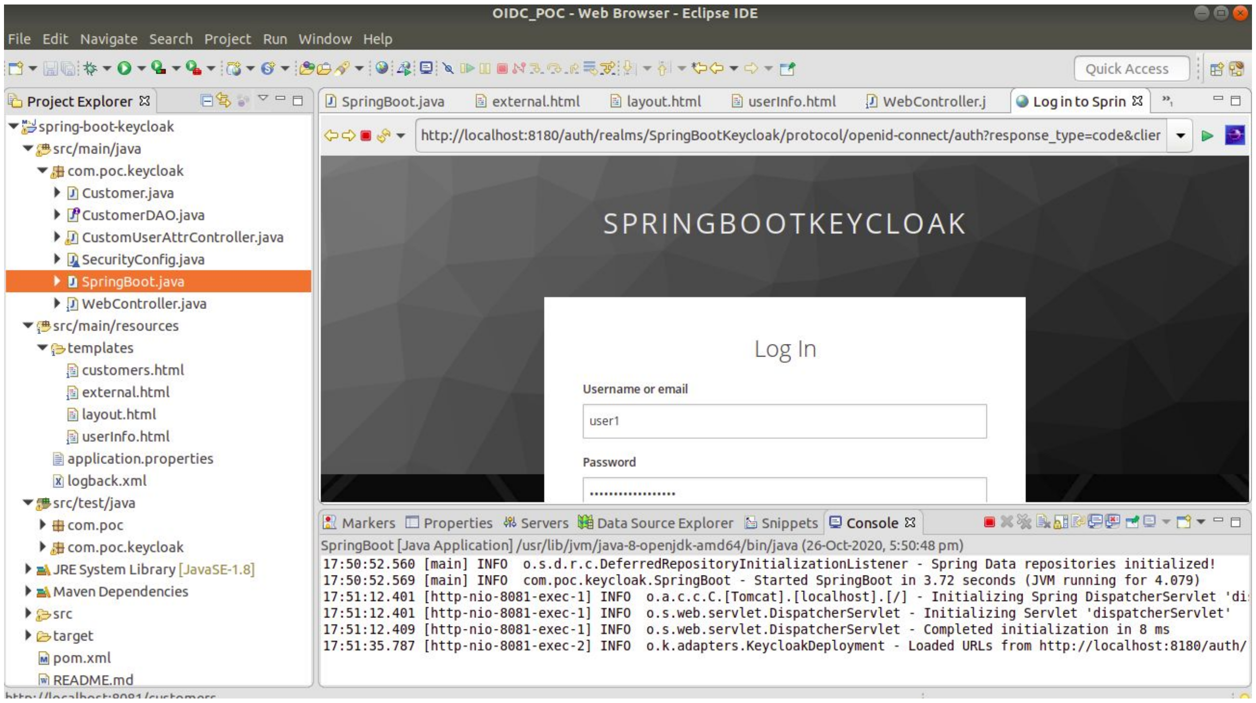
Knowing the customer

Enter the intranet: customers

```
SpringBoot [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (26-Oct-2020, 5:50:48 pm)
17:50:52.560 [main] INFO  o.s.d.r.c.DeferredRepositoryInitializationListener - Spring Data repositories initialized!
17:50:52.569 [main] INFO  com.poc.keycloak.SpringBoot - Started SpringBoot in 3.72 seconds (JVM running for 4.079)
17:51:12.401 [http-nio-8081-exec-1] INFO  o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring DispatcherServlet 'di
17:51:12.401 [http-nio-8081-exec-1] INFO  o.s.web.servlet.DispatcherServlet - Initializing Servlet 'dispatcherServlet'
17:51:12.409 [http-nio-8081-exec-1] INFO  o.s.web.servlet.DispatcherServlet - Completed initialization in 8 ms
17:51:35.787 [http-nio-8081-exec-2] INFO  o.k.adapters.KeycloakDeployment - Loaded URLs from http://localhost:8180/auth/
```

2. We can see that we've been redirected to authenticate through Keycloak to see if we're authorized to view this content:

http://localhost:8180/auth/realms/SpringBootKeycloak/protocol/openid-connect/auth?response_type=code&client_id=login-app&redirect_uri=http%3A%2F%2Flocalhost%3A8081%2Fcustomers&state=bfd5bd7c-18a6-4506-9b5b-cd6d5612859d&login=true&scope=openid



http://localhost:8180/auth/realms/SpringBootKeycloak/protocol/openid-connect/auth?response_type=code&clier

SPRINGBOOTKEYCLOAK

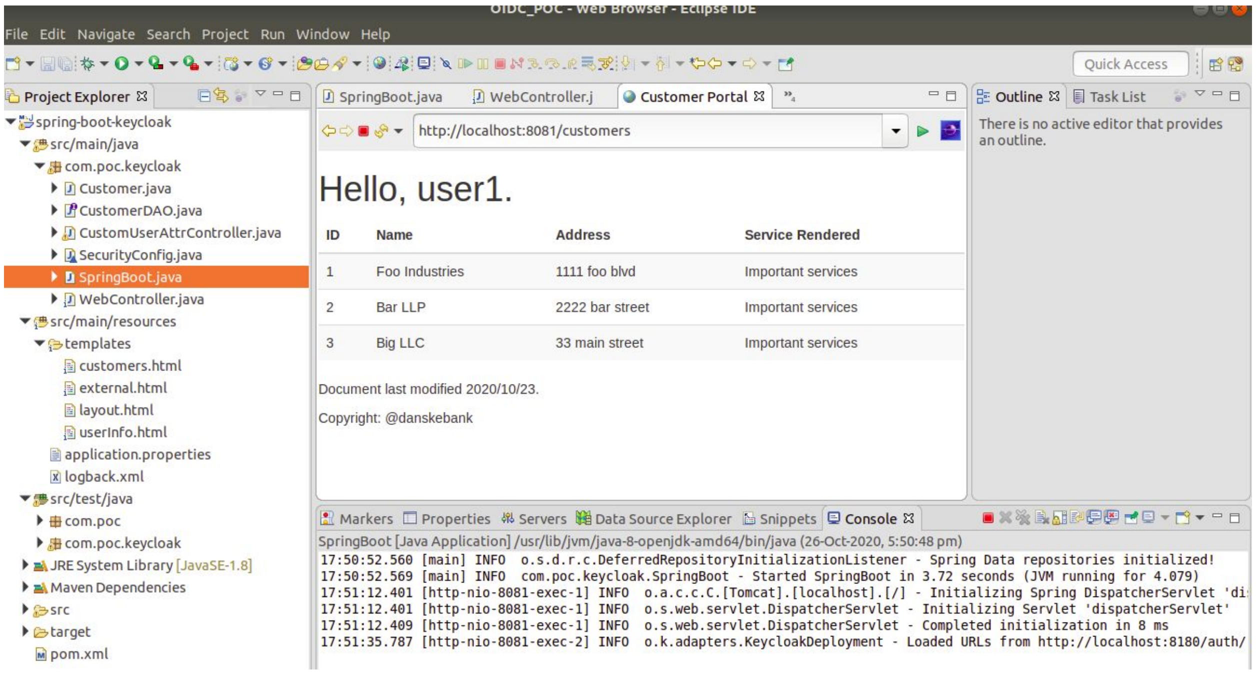
Log In

Username or email: user1

Password: ••••••••••••••••••

```
SpringBoot [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (26-Oct-2020, 5:50:48 pm)
17:50:52.560 [main] INFO  o.s.d.r.c.DeferredRepositoryInitializationListener - Spring Data repositories initialized!
17:50:52.569 [main] INFO  com.poc.keycloak.SpringBoot - Started SpringBoot in 3.72 seconds (JVM running for 4.079)
17:51:12.401 [http-nio-8081-exec-1] INFO  o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring DispatcherServlet 'di
17:51:12.401 [http-nio-8081-exec-1] INFO  o.s.web.servlet.DispatcherServlet - Initializing Servlet 'dispatcherServlet'
17:51:12.409 [http-nio-8081-exec-1] INFO  o.s.web.servlet.DispatcherServlet - Completed initialization in 8 ms
17:51:35.787 [http-nio-8081-exec-2] INFO  o.k.adapters.KeycloakDeployment - Loaded URLs from http://localhost:8180/auth/
```

3. Once we log in as user1, Keycloak will verify our authorization – that we have the user role – and we'll be redirected to the restricted customers page:



http://localhost:8081/customers

Hello, user1.

ID	Name	Address	Service Rendered
1	Foo Industries	1111 foo blvd	Important services
2	Bar LLP	2222 bar street	Important services
3	Big LLC	33 main street	Important services

Document last modified 2020/10/23.

Copyright: @danskebank

```
SpringBoot [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (26-Oct-2020, 5:50:48 pm)
17:50:52.560 [main] INFO  o.s.d.r.c.DeferredRepositoryInitializationListener - Spring Data repositories initialized!
17:50:52.569 [main] INFO  com.poc.keycloak.SpringBoot - Started SpringBoot in 3.72 seconds (JVM running for 4.079)
17:51:12.401 [http-nio-8081-exec-1] INFO  o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring DispatcherServlet 'di
17:51:12.401 [http-nio-8081-exec-1] INFO  o.s.web.servlet.DispatcherServlet - Initializing Servlet 'dispatcherServlet'
17:51:12.409 [http-nio-8081-exec-1] INFO  o.s.web.servlet.DispatcherServlet - Completed initialization in 8 ms
17:51:35.787 [http-nio-8081-exec-2] INFO  o.k.adapters.KeycloakDeployment - Loaded URLs from http://localhost:8180/auth/
```

Now we've finished the set up of connecting Spring Boot with Keycloak and demonstrating how it works.

As we can see, the entire process of calling the Keycloak Authorization Server was handled seamlessly by Spring Boot for us. We did not have to call the Keycloak API to generate the Access Token ourselves, or even send the Authorization header explicitly in our request for protected resources.

3. Current Limitation:

API Gateway has a limit of 10,000 RPS (requests per second), which might not be enough for some cases. Whereas Load Balancers or Cloud prodded ALB, on the other hand, is virtually unlimited. In fact, Some of the cloud providers specify no limits in terms of connections per second or concurrently. Though For Serverless applications, API Gateway was the only way to go until recently, when Cloud providers announced the integration of ALB with Lambda functions.

API Gateways are not offering rule-based routing mechanisms. Apart from supporting a URL path-based approach.