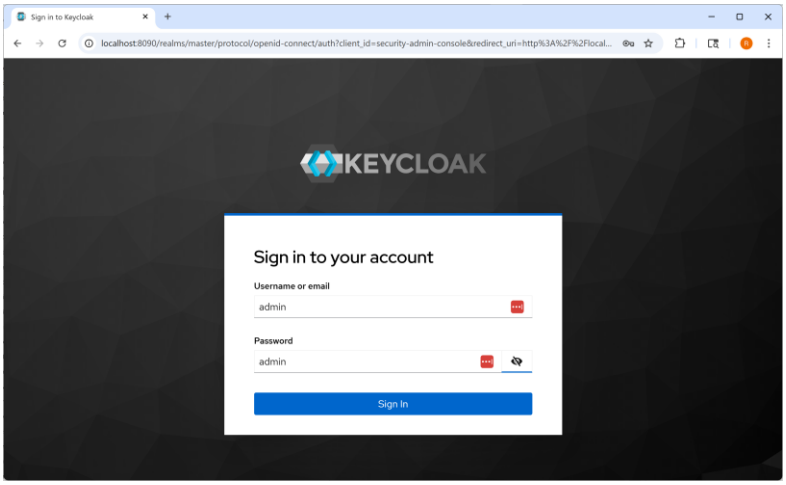
Lab 10

Part 1

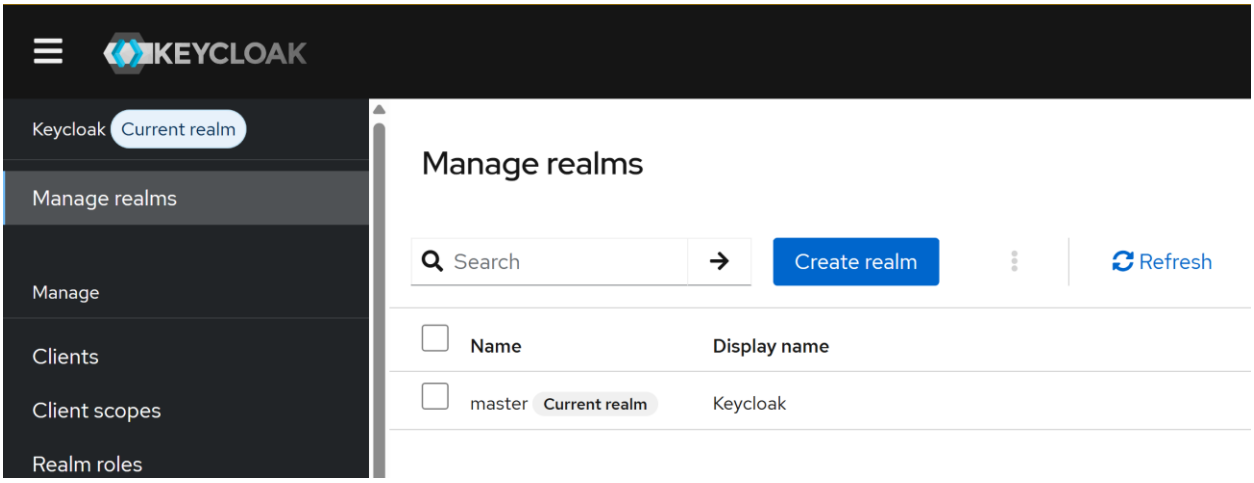
First start KeyCloak using docker:

docker run -d -p 8090:8080 --name keycloak -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:latest start-dev

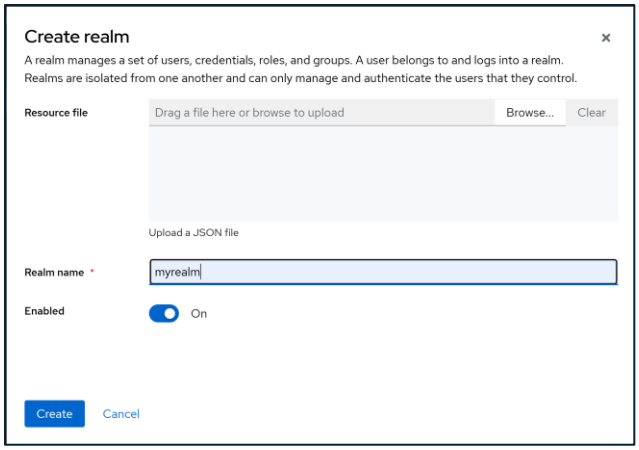
Then open **http://localhost:8090/**



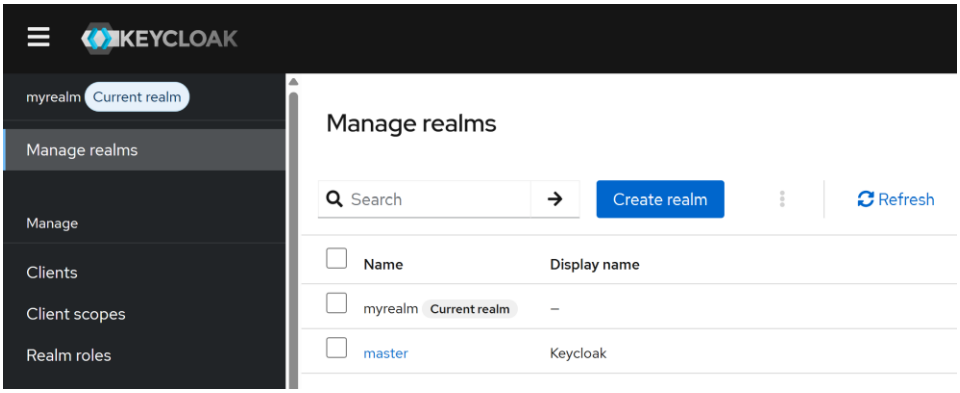
Login with Username **admin** and password **admin**.



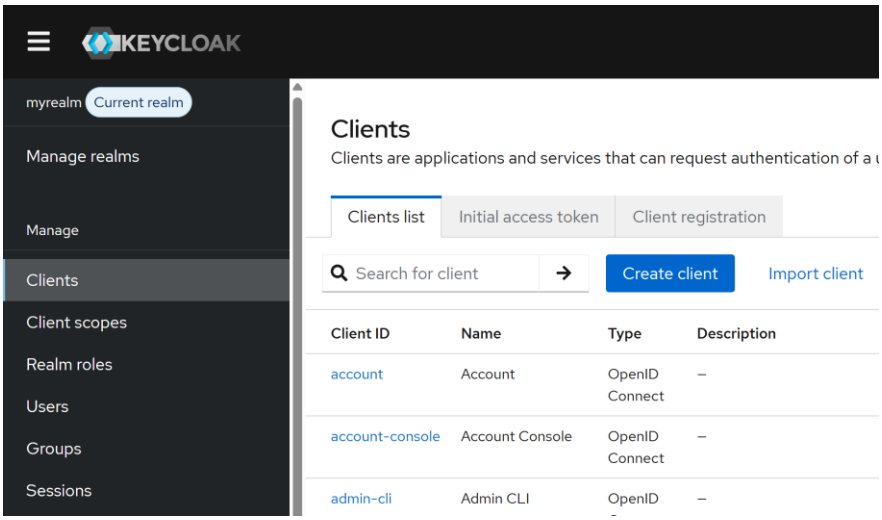
Select the **Manage realms** from the menu at the left, and click the **Create realm** button.



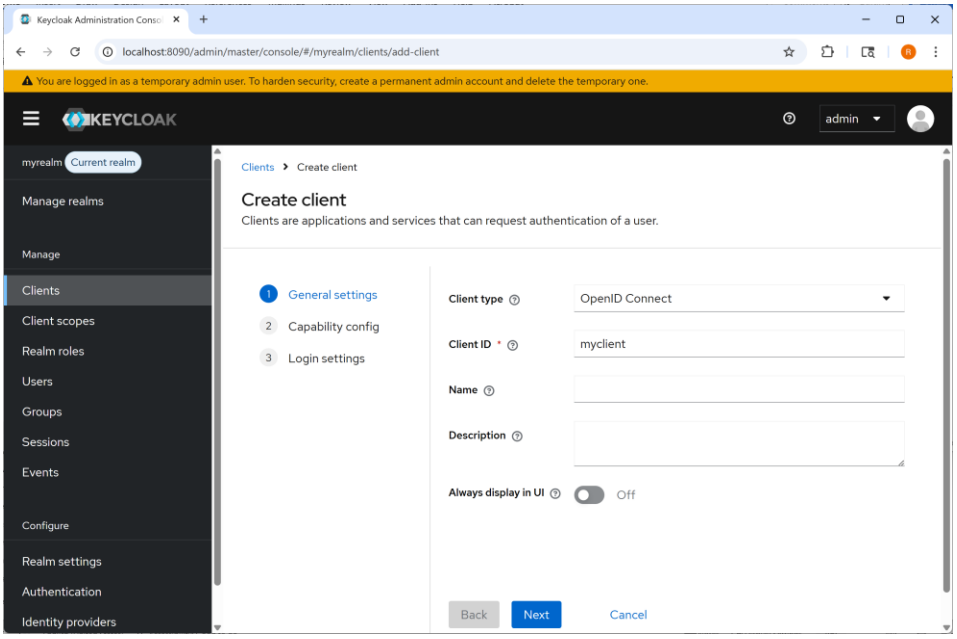
Name the realm **myrealm** and click the **Create** button.



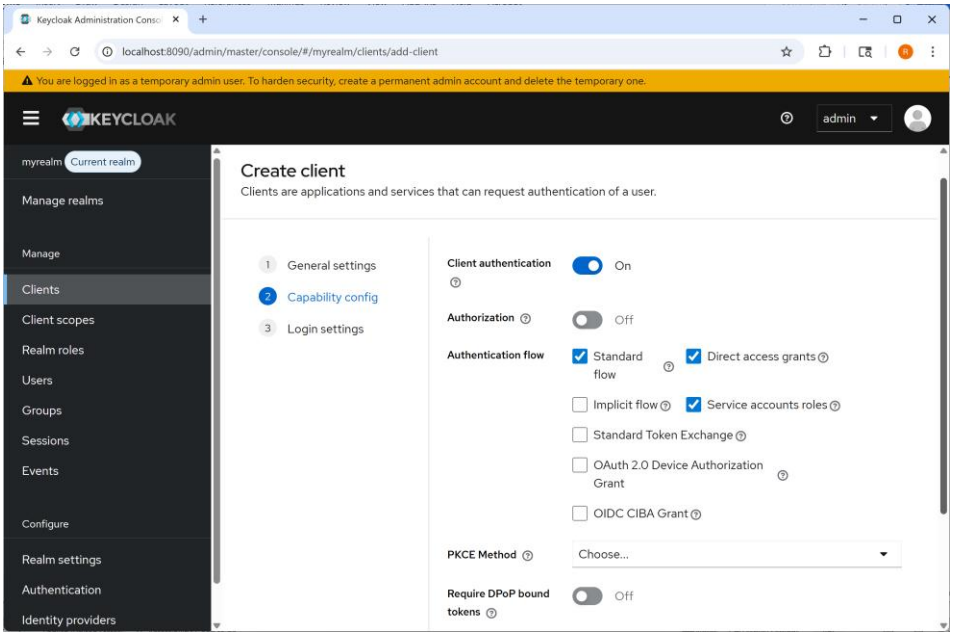
The realm **myrealm** is now created is the current active realm



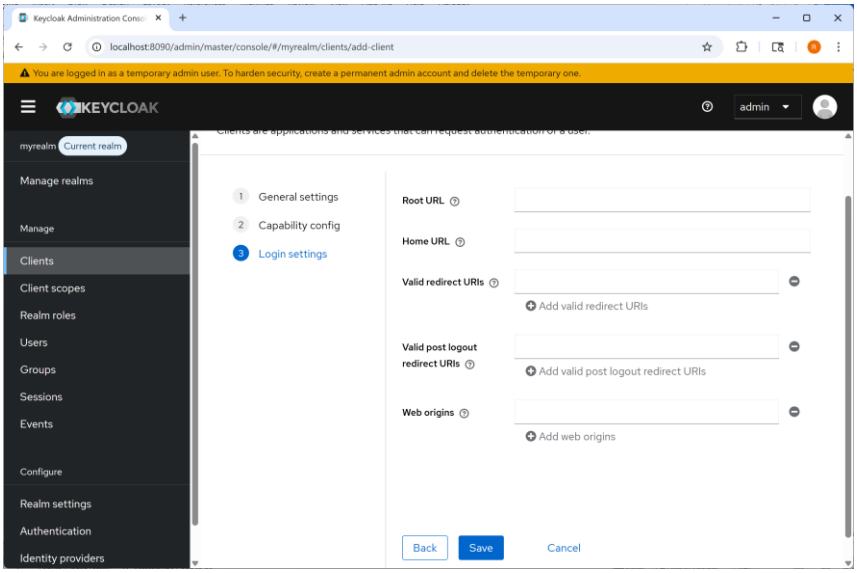
Select the **Clients** from the menu at the left, and click the **Create client** button.



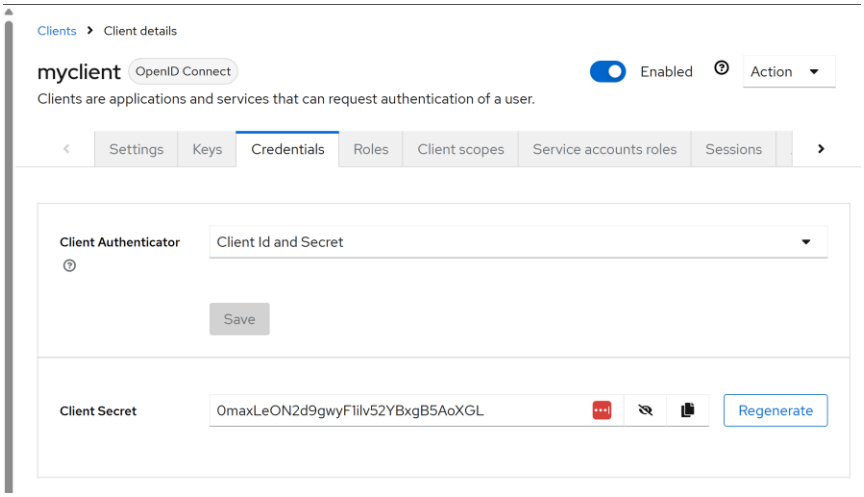
Fill in the Client ID **myclient** and click **Next**



Set **Client authentication** to **On** and select **Direct access grants** and **Service accounts roles.** Then click **Next**



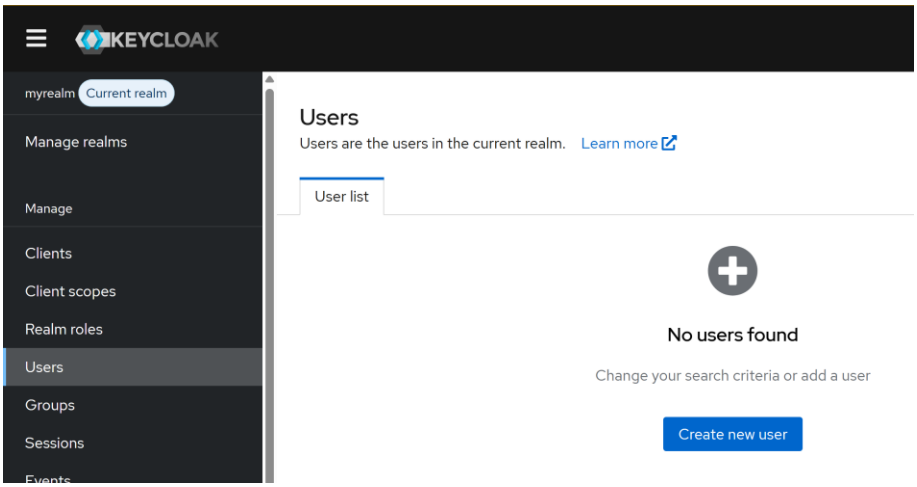
Click **Save**



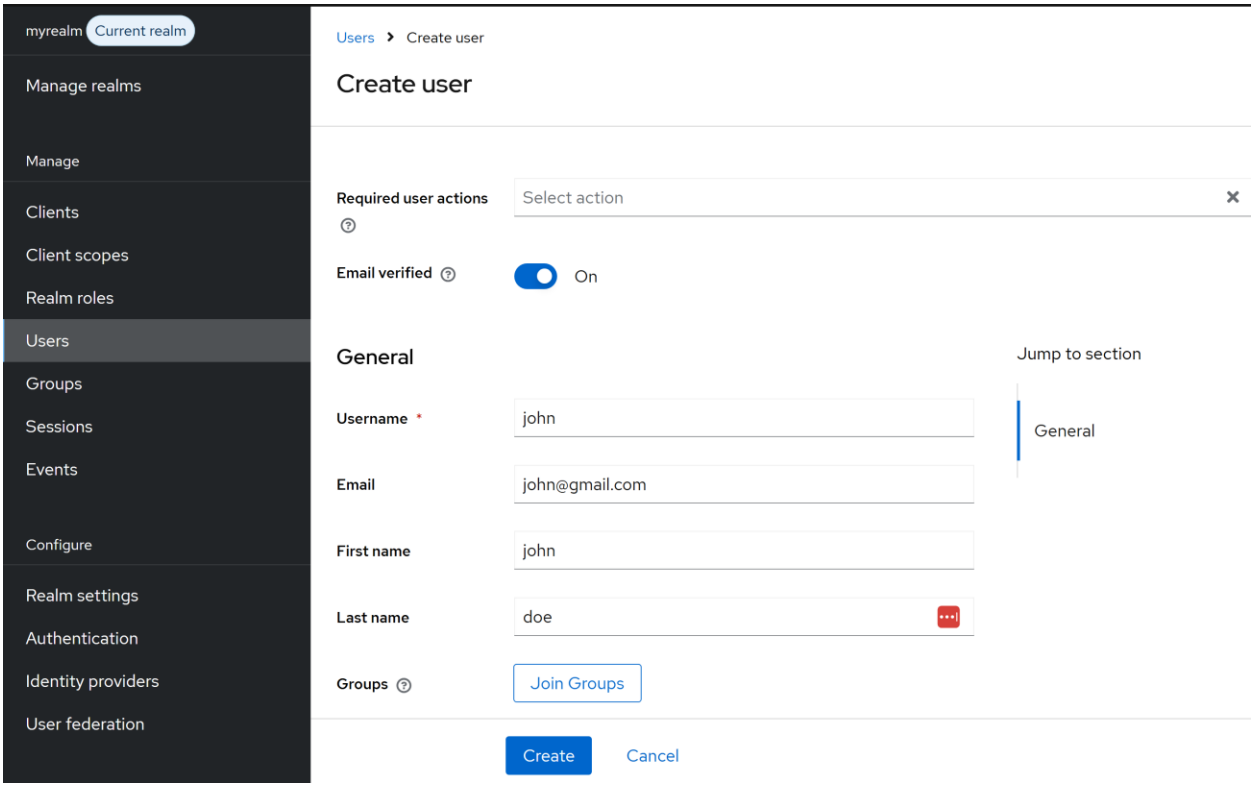
Select the **Credentials** tab and copy the **Client Secret**. Save this client secret somewhere (notepad).

We have now created a client with the Client ID **myclient** and a Client secret.

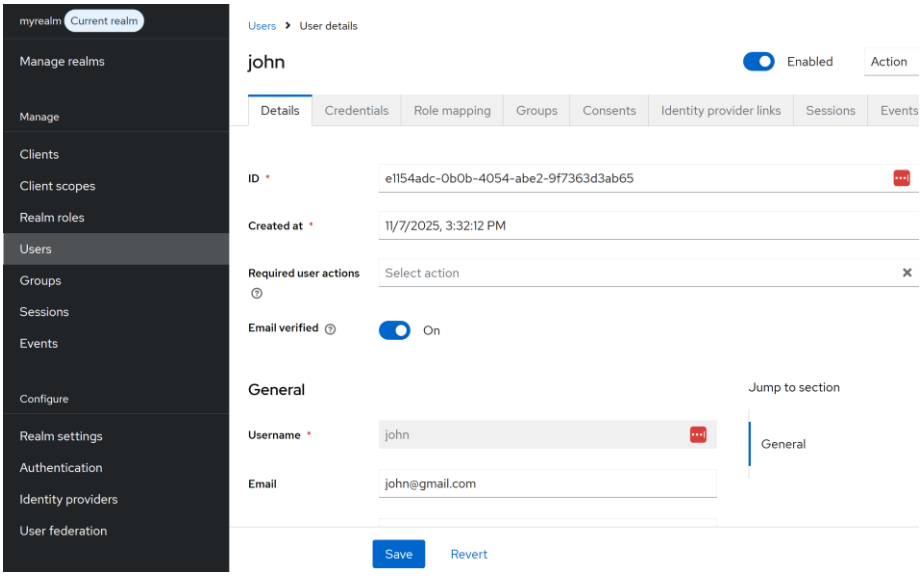
Now we are going to create Users.



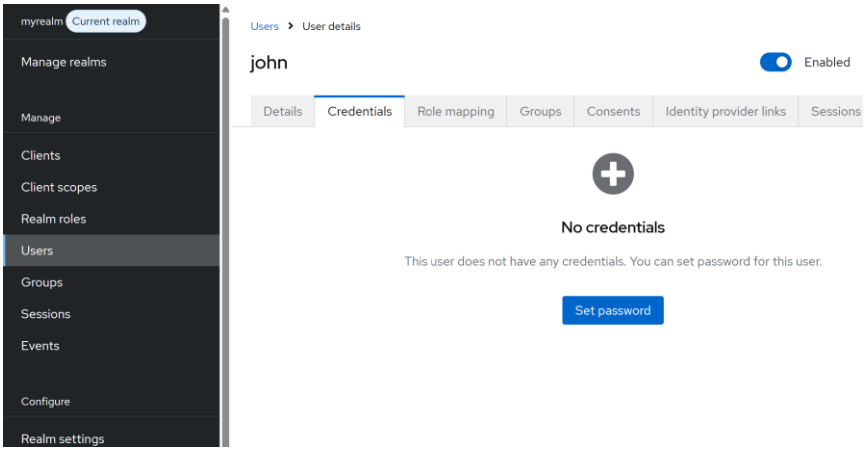
On the menu on the left select **Users** and click the **Create new user** button.



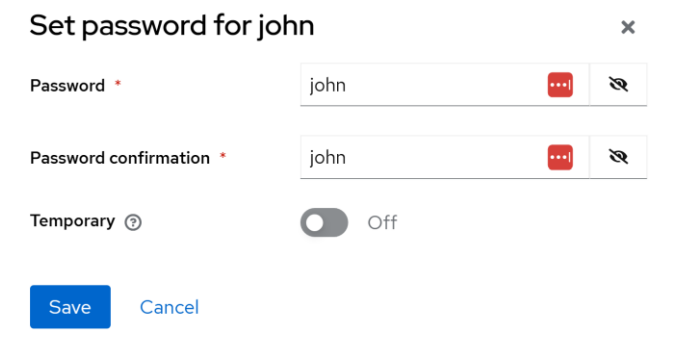
Fill in the **Username, Email, First name, Last name** and select the **Email verified** option. Then click **Create**



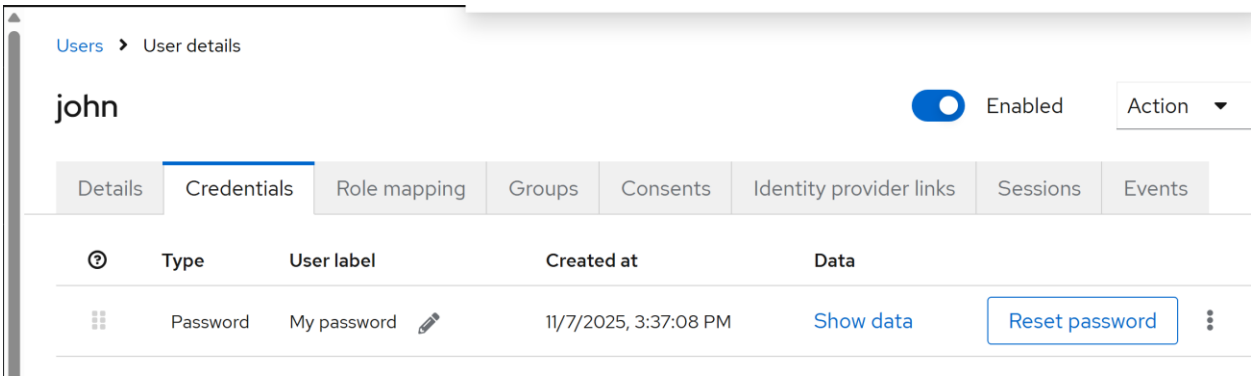
User **john** is created.



Select the **Credentials** tab and click the **Set password** button

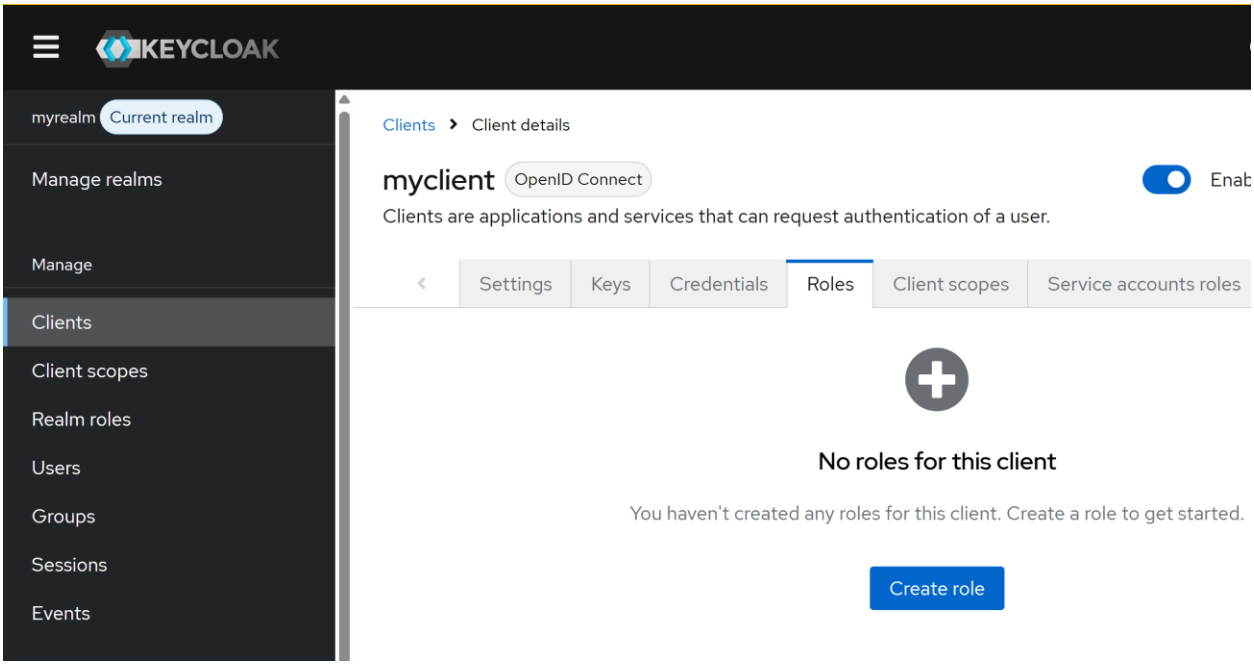


Fill in the password **john** and set **Temporary** to **Off**. Click the **Save** button.

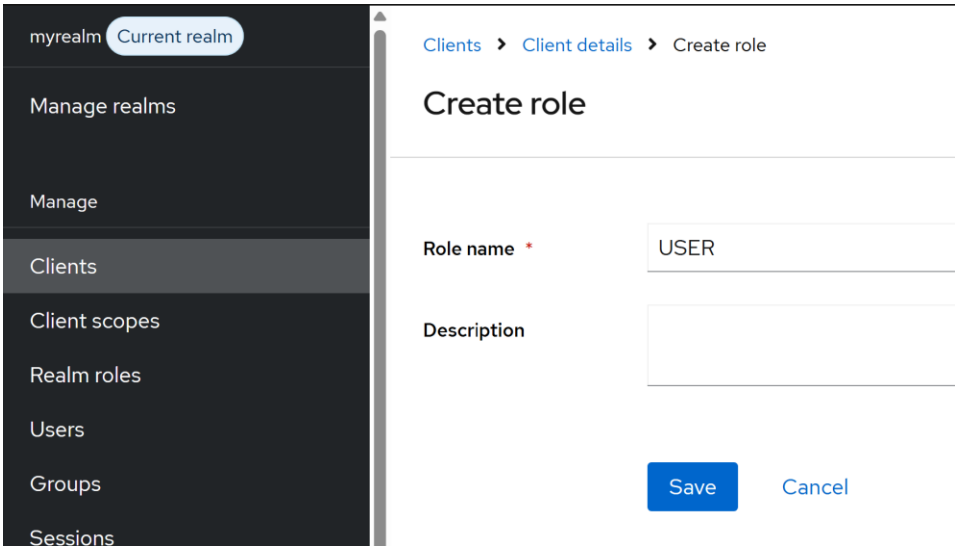


User john has now password john.

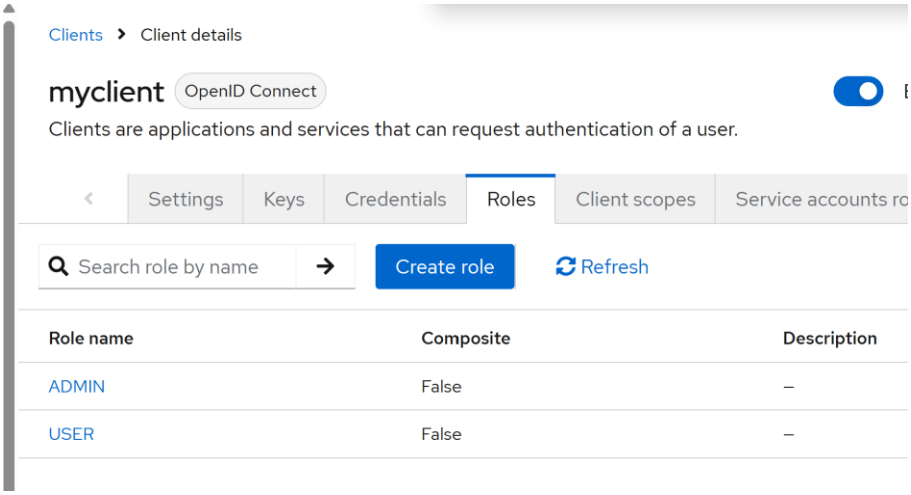
Now we need to create 2 roles: **user** and **admin**.



On the menu on the left, select **Clients** and select the roles tab for client **myclient**. click the **Create role** button.

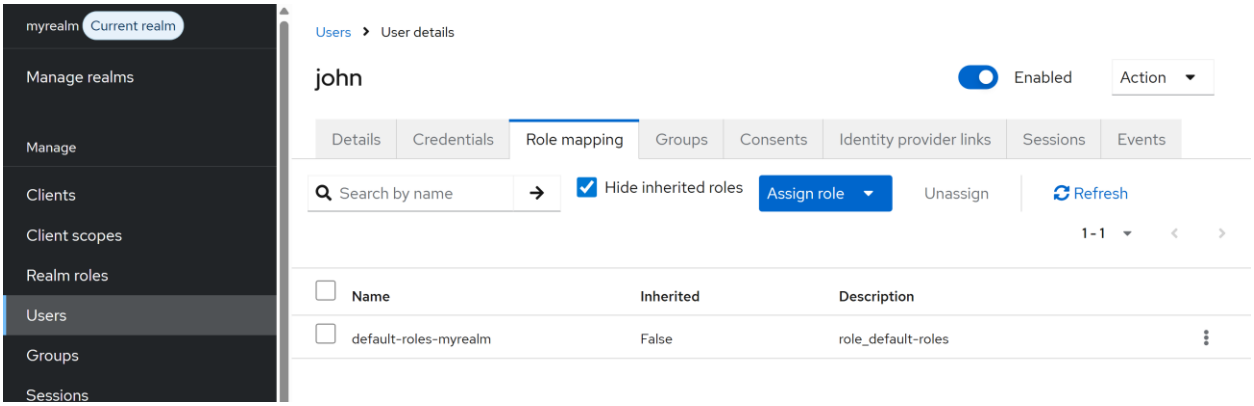


Make a **USER** role and click **Save**

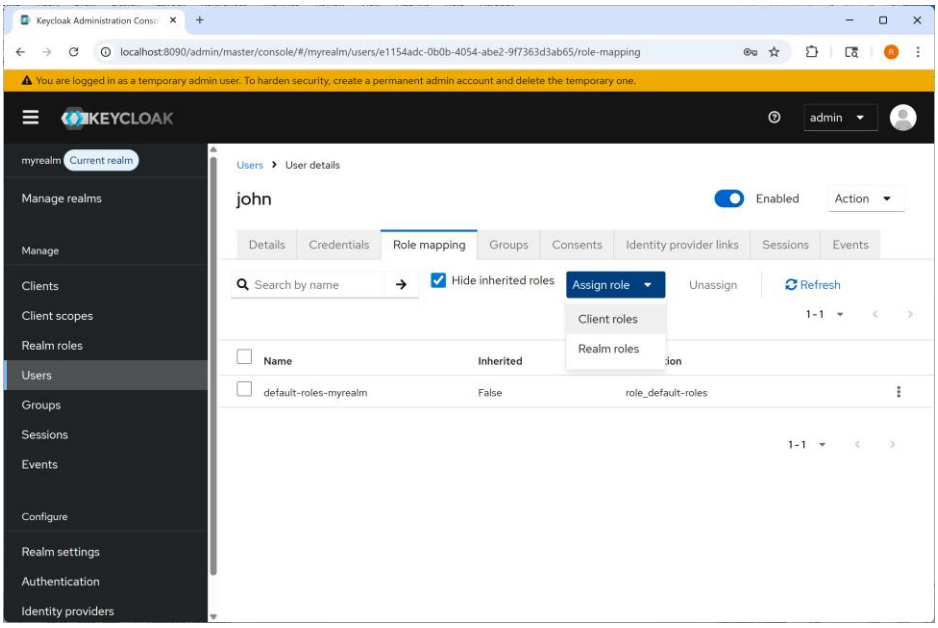


In the same way create an **ADMIN** role.

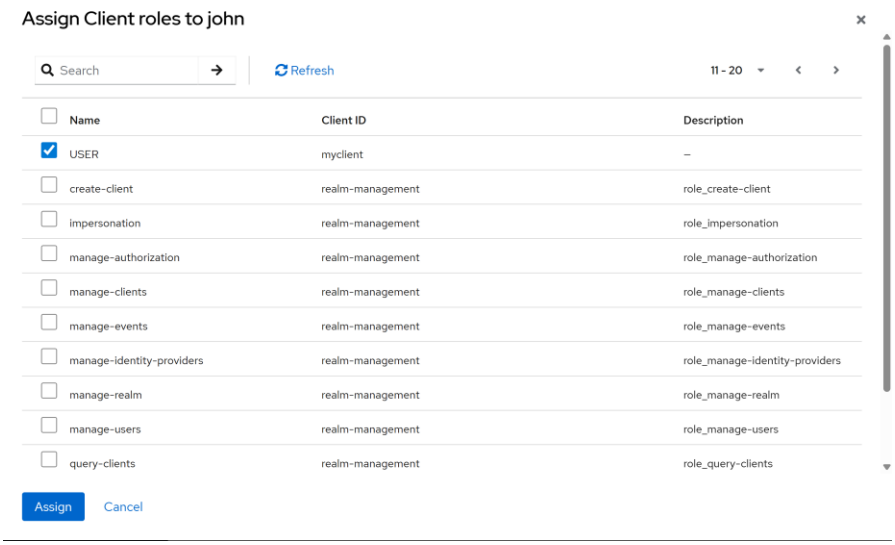
Now go back to the Users on the left menu and go to user john.



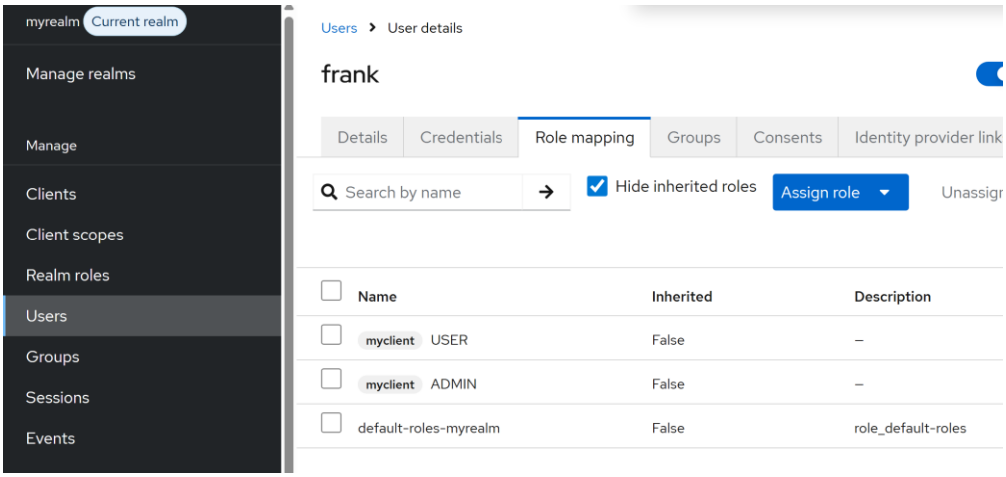
Select the **Role mapping** tab and click the **Assign role** pull down list.



Select **Client roles**.

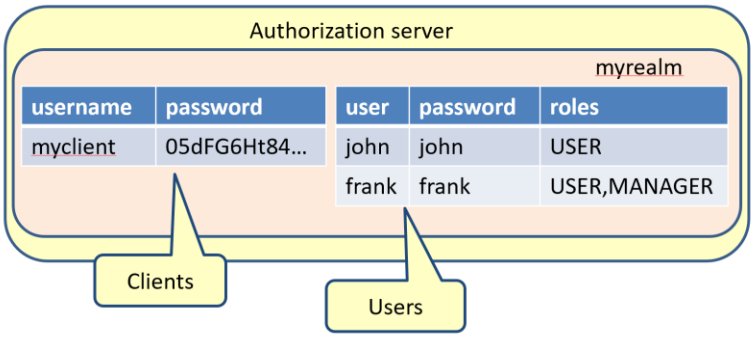


Select the **USER** role and click **Assign**.



In the same way create user **frank** with password **frank** and roles **USER** and **ADMIN**

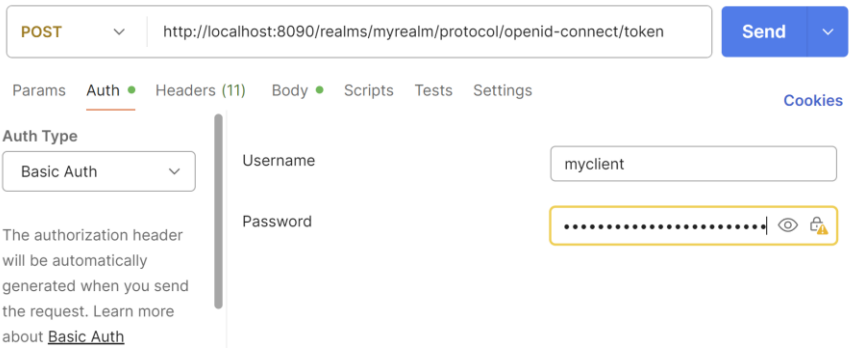
We now have the following configuration in KeyCloak:



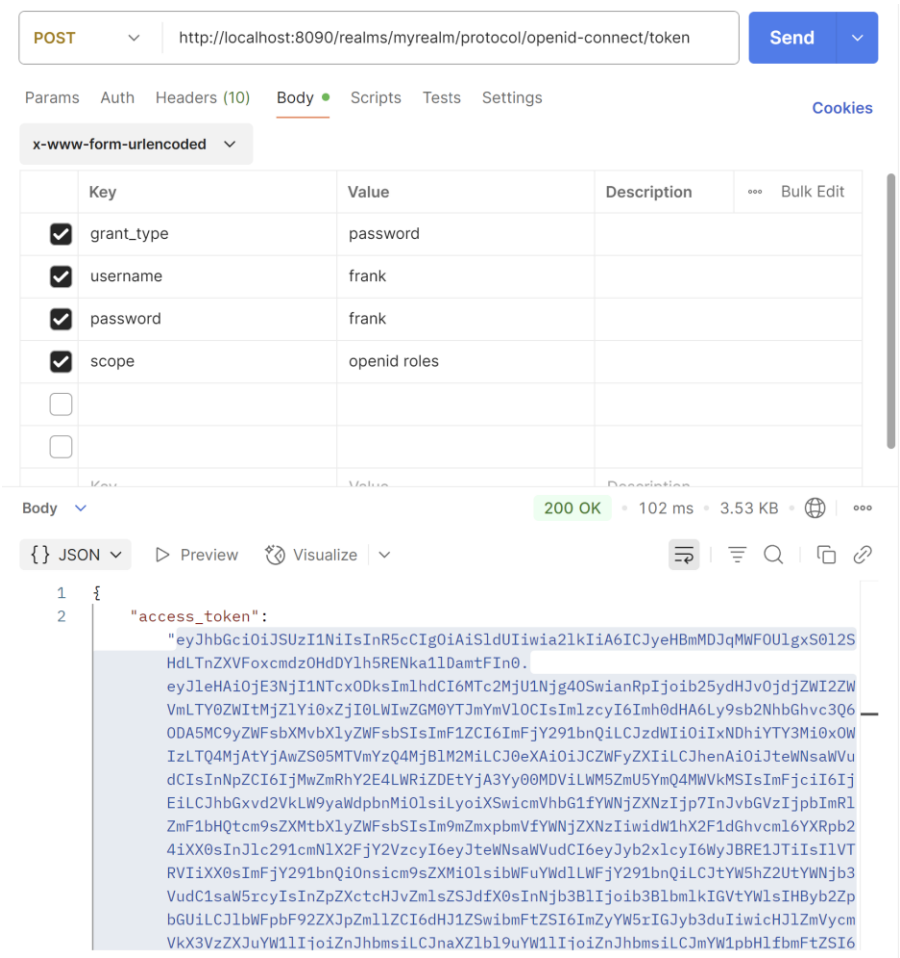
Now we are going to retrieve the token for users john and frank.

Open **Postman** and create a **POST** request for the URL

http://localhost:8090/realms/myrealm/protocol/openid-connect/token



In the **Auth** tab, select **Basic Auth** and fill in the **username** and **secret** of the **Client** configured in KeyCloak.

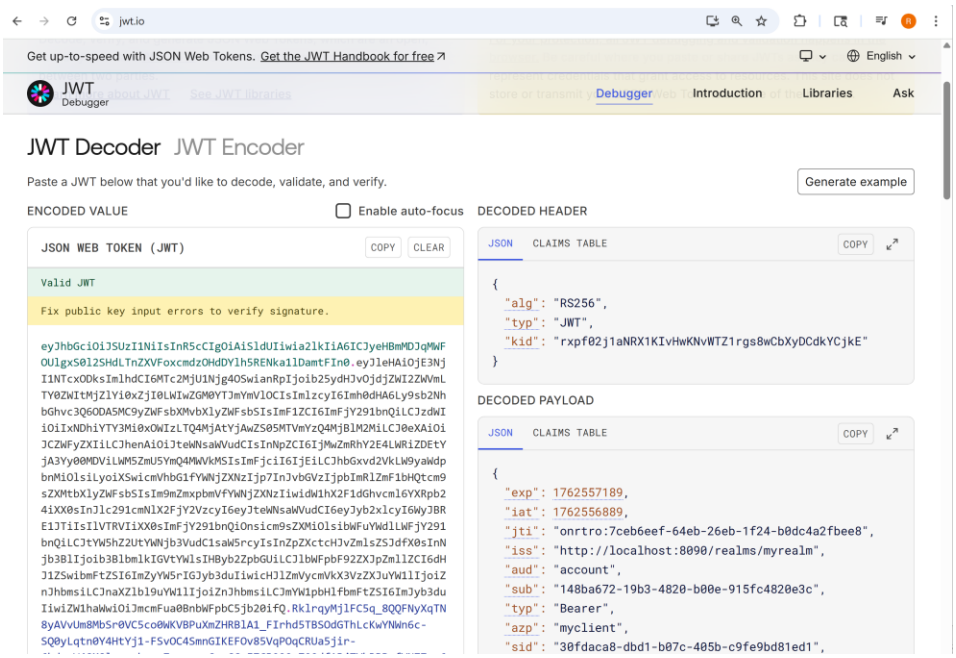


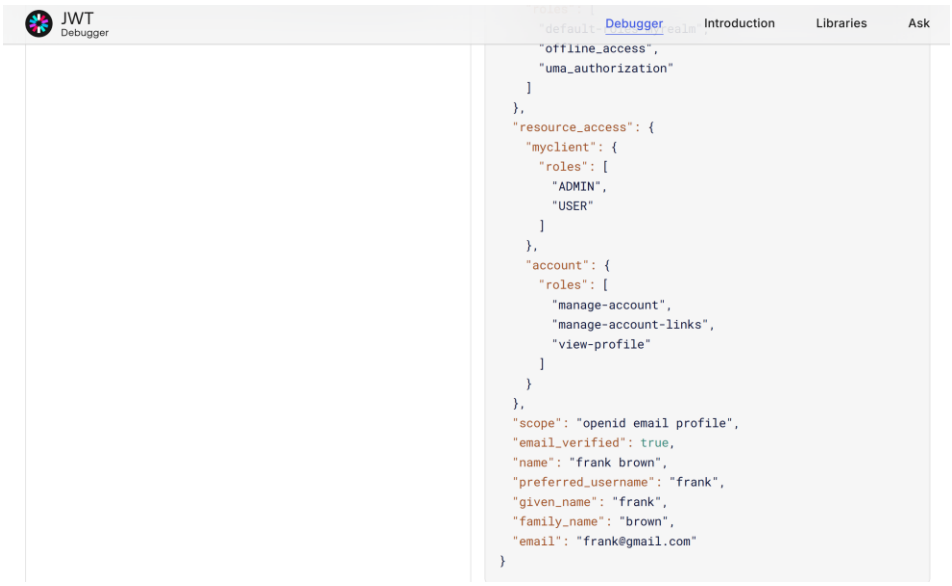
In the **Body** tab, select **x-www-form-urlencoded** and fill in:

- grant_type = client-credentials
- username = frank
- password = frank.

The click the **Send** button, and see that we receive an access token back from the authorization server.

Copy the access token and go to **jwt.io**





Notice that the token contains the **roles** for user **frank**.

Now check the token for user **john**

```
  "resource_access": {
    "myclient": {
      "roles": [
        "USER"
      ]
    },
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "scope": "openid email profile",
  "email_verified": true,
  "name": "john doe",
  "preferred_username": "john",
  "given_name": "john",
  "family_name": "doe",
  "email": "john@gmail.com"
```


Our KeyCloak authorization server is now working.

Now you are ready to run the given code for this lecture. Run the given applications and check if they work correctly.

Part 2

Now change the in-memory users so that we have 3 users:
One with the role customer, one with the roles customer and employee, and one with the roles customer, employee and manager.

Then create 3 microservices A, B and C.
C contains salary data
B contains employee contact data (phone)
A is our actual company service that is used by customers, employees and managers

In A you can call productdata that is accessible by all customers, employees and managers
In A you can call employee contact data that is accessible only by employees and managers. A will call B to get the actual employee contact data.
In A you can call salary data that is accessible only by managers. A will call C to get the actual salary data.

Check that everyone can call product data
Check that only employees and managers can call employee data
Check that only managers can call salary data