

1. Download keycloak zip from below link
<https://www.keycloak.org/downloads>

Downloads 21.0.1

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

Server

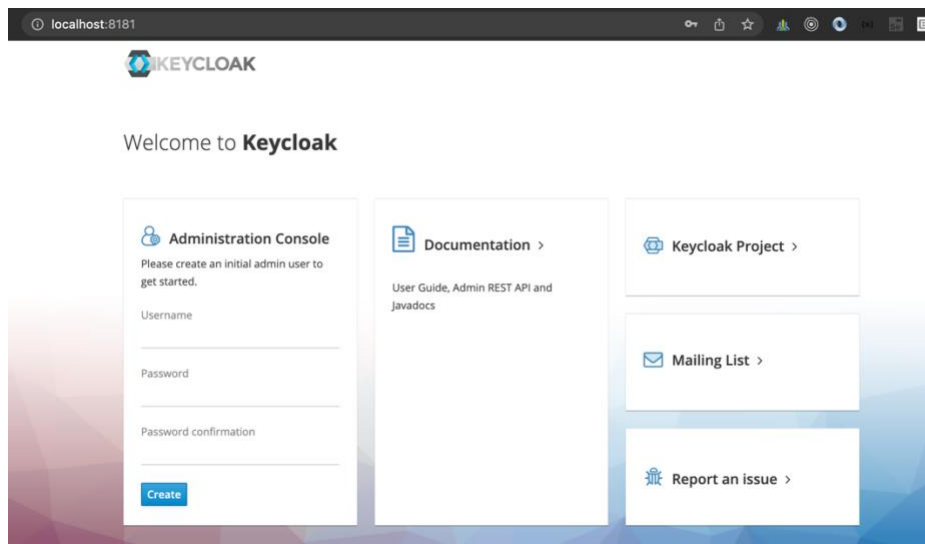
Keycloak	Distribution powered by Quarkus	ZIP (sha1) TAR.GZ (sha1)
Container image	For Docker, Podman, Kubernetes and OpenShift	Quay
Operator	For Kubernetes and OpenShift	OperatorHub

2. Extract the zip folder and open terminal / cmd from within the bin folder
3. Execute below command to start keycloak server
MAC -> `./kc.sh start-dev --http-port=8181`
WINDOWS -> `kc.bat start-dev --http-port=8181`

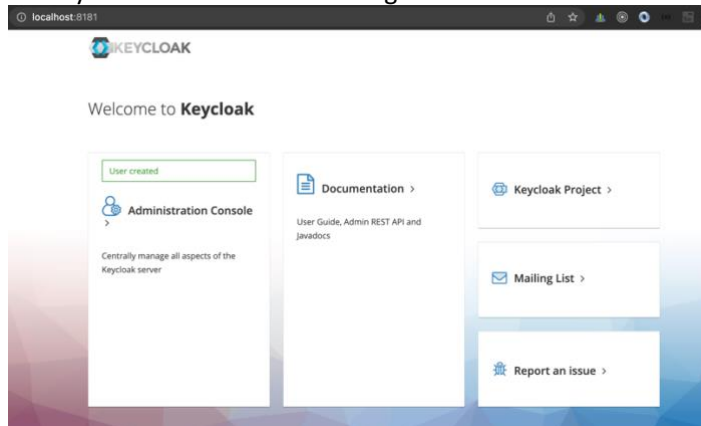
```
[(base) Manishs-MacBook-Pro:bin Shalini$ ./kc.sh start-dev --http-port=8181  
Updating the configuration and installing your custom providers, if any. Please wait.
```

NOTE : Please wait it will take some time to start the server

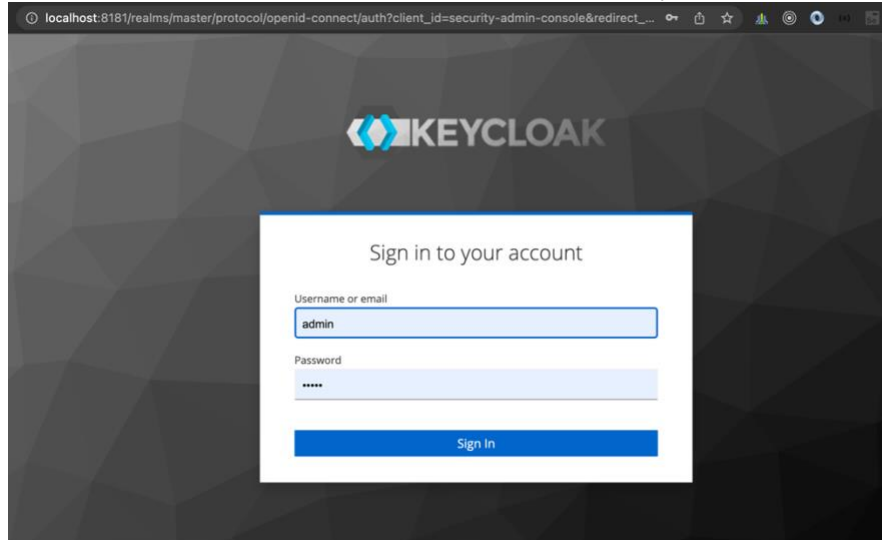
4. Open server on browser at url <http://localhost:8181>
5. Dashboard looks as follows:



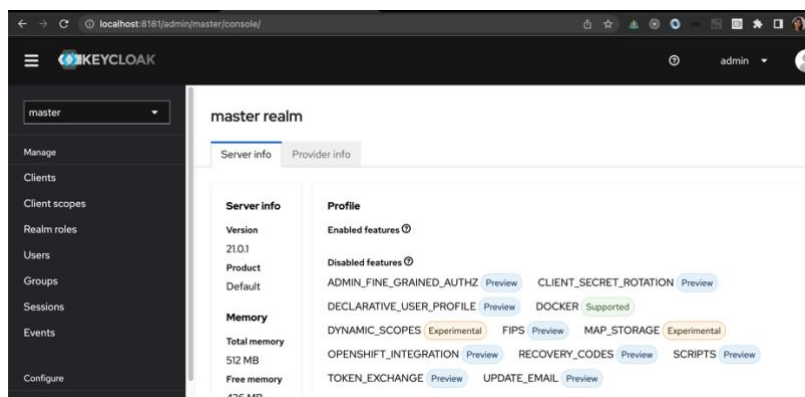
6. Enter username – admin, password – admin and click Create
7. Once you create on click following dashboard is seen



8. Click on Administration console and enter username and password – admin/ admin

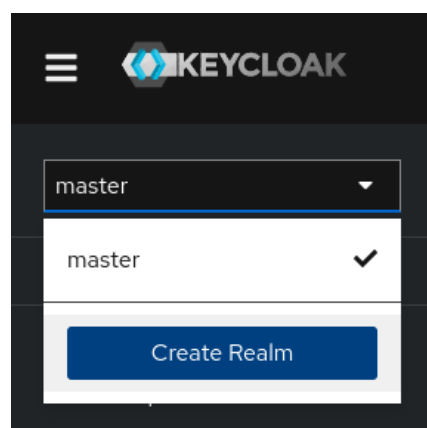


9. Once sign in -> dashboard looks as follows:



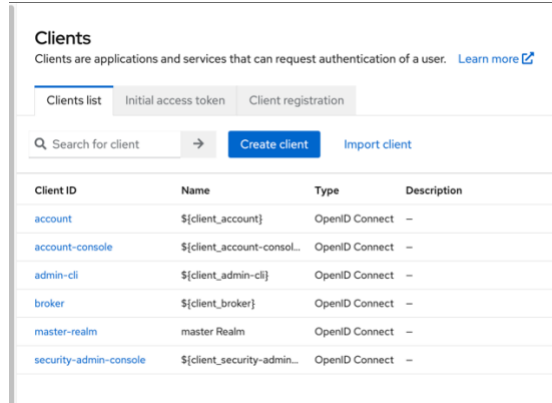
10. https://www.keycloak.org/getting-started/getting-started-docker#_create_a_realm

11. Use these steps to create the first realm.
- Open the Keycloak Admin Console.
 - Click the word master in the top-left corner, then click Create realm.
 - Enter **oauth-demo-realm** in the Realm name field.
 - Click Create.



12. Secure the first application

- To secure the first application, you start by registering the application with your Keycloak instance:
- Open the Keycloak Admin Console.
- Click Clients.
- Click Create client



- Fill in the form with the following values:

The screenshot shows the 'Create client' form in the Keycloak Admin Console. The 'Client type' is set to 'OpenID Connect'. The 'Client ID' field contains the value 'oauth2-client-credentials'. The 'Name' and 'Description' fields are empty. The 'Always display in UI' toggle is turned off. The 'Next', 'Back', and 'Cancel' buttons are at the bottom.

- Click Next – select the following

The screenshot shows the 'Create client' form in the Keycloak Admin Console, with the 'Capability config' tab selected. The 'Client authentication' toggle is turned on. The 'Authorization' toggle is turned off. The 'Authentication flow' section shows 'Standard flow' and 'Implicit flow' as unchecked, while 'Direct access grants' and 'Service accounts roles' are checked. The 'OAuth 2.0 Device Authorization Grant' and 'OIDC CIBA Grant' are unchecked. The 'Next', 'Back', and 'Cancel' buttons are at the bottom.

- Click Next and Save

- h. After save dashboard looks as follows: Click on Credentials tab for client secret

oauth2-client-credentials OpenID Connect Enabled Action

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

Settings Keys **Credentials** Roles Client scopes Service accounts roles Sessions Advanced

Client Authenticator Client Id and Secret

Save

Client secret Regenerate

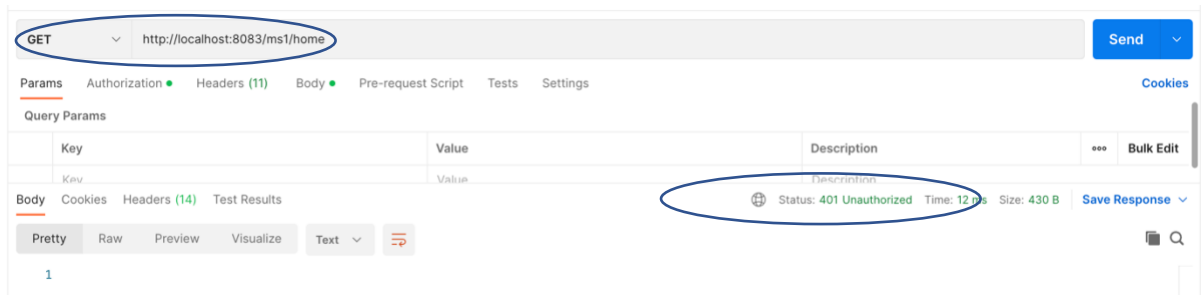
13. Create a spring boot application with following dependencies

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-oauth2-client</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-oauth2-jose</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
  <optional>true</optional>
</dependency>
```

14. The project will be available on github

15. Start the spring boot project. Once successfully started open postman and hit the rest controller.

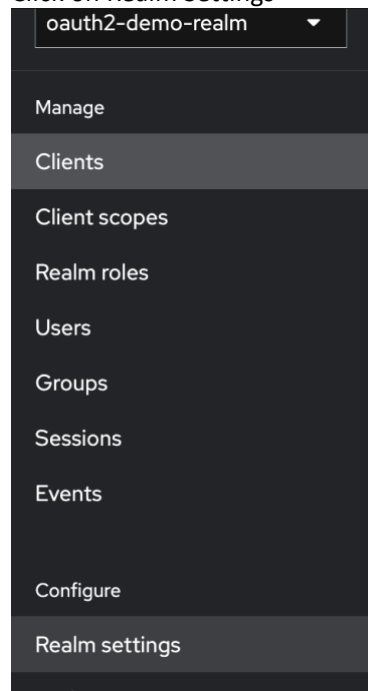
16. Should get unauthorized as follows:



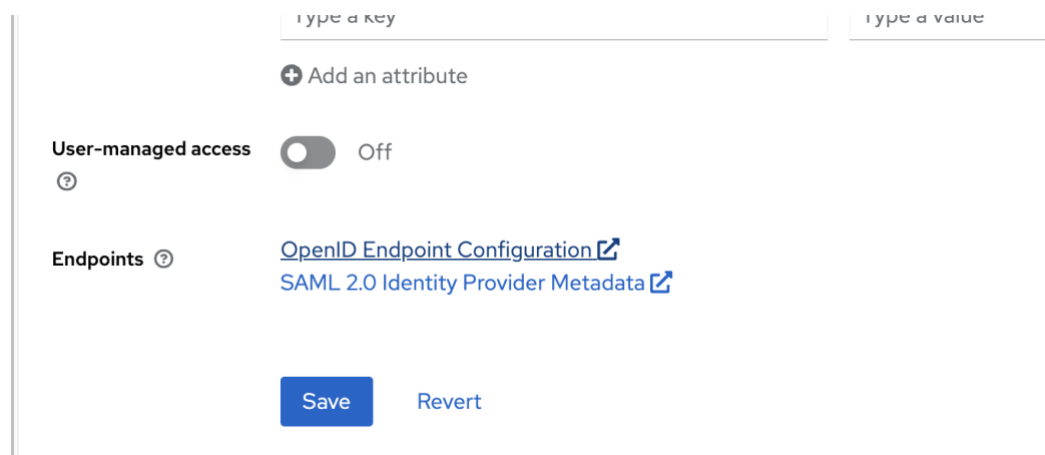
17. Need to get access token before hitting the url.

18. To access the configuration created by keycloak to register oauth-resource server

a. Click on Realm Settings



b. Click on OpenID Endpoint Configuration to open the configuration file:



19. The json looks as follows:

Issuer , jwks-uri and token_endppoint are what we will need

```

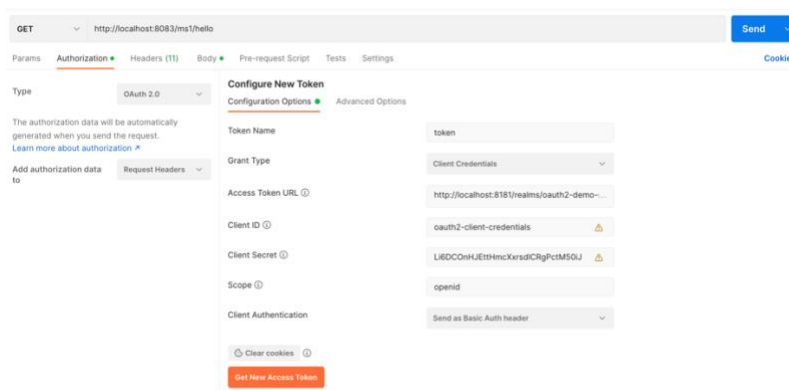
localhost:8181/realms/oauth2-demo-realm/.well-known/openid-configuration
{
  "issuer": "http://localhost:8181/realms/oauth2-demo-realm",
  "authorization_endpoint": "http://localhost:8181/realms/oauth2-demo-realm/protocol/openid-connect/auth",
  "token_endpoint": "http://localhost:8181/realms/oauth2-demo-realm/protocol/openid-connect/token",
  "introspection_endpoint": "http://localhost:8181/realms/oauth2-demo-realm/protocol/openid-connect/token/introspect",
  "userinfo_endpoint": "http://localhost:8181/realms/oauth2-demo-realm/protocol/openid-connect/userinfo",
  "end_session_endpoint": "http://localhost:8181/realms/oauth2-demo-realm/protocol/openid-connect/logout",
  "frontchannel_logout_supported": true,
  "frontchannel_logout_supported_uri": null,
  "jwks_uri": "http://localhost:8181/realms/oauth2-demo-realm/protocol/openid-connect/certs",
  "check_session_iframe": "http://localhost:8181/realms/oauth2-demo-realm/protocol/openid-connect/login-status-iframe.html",
  "grant_types_supported": [
    "authorization_code",
    "implicit",
    "refresh_token",
    "password",
    "client_credentials",
    "urn:ietf:params:oauth:grant-type:device_code",
    "urn:openid:params:grant-type:ciba"
  ],
  "acr_values_supported": [
    "0",
    "1"
  ],
  "response_types_supported": [
    "code",

```

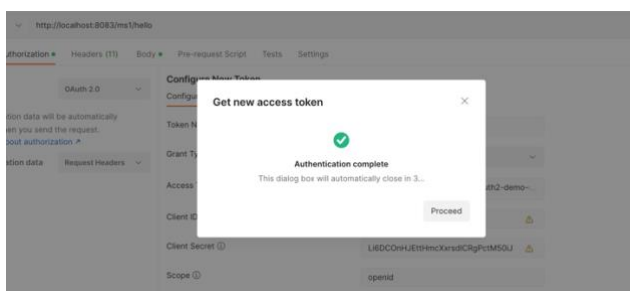
20. Open POSTMAN to get the access token to successfully connect to microservice

21. Enter details as follows

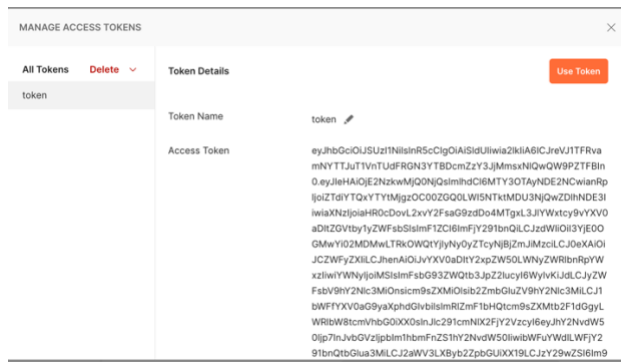
- Grant type : Client credentials
- Access token url – copy token-endpoint from json file
- Client Id : oauth2-client-credentials
- Client Secret – copy from keycloak Clients
- Scope – openid
- Client Authentication – keep it as it is
- Click on Get new access token



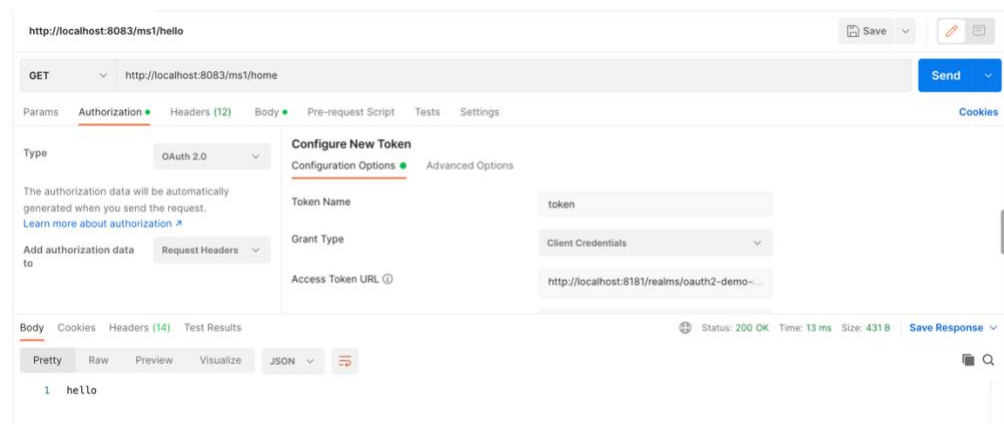
22. Should get Authentication Complete as follows:



23. Token looks as follows: Click Use Token



24. Now should get success and a hello message

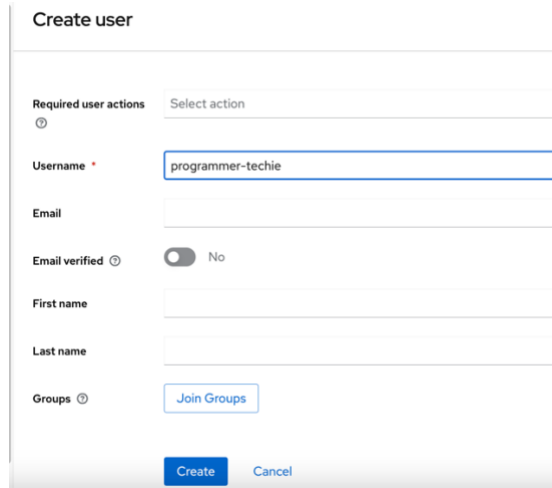


25. User is used if need to create and Oauth client for MVC applications

26. Create a user

- a. Initially, the realm has no users. Use these steps to create a user:
- b. Open the Keycloak Admin Console.
- c. Click Users in the left-hand menu.
- d. Click Create new user.
- e. Fill in the form with the following values:
 - i. Username: programmer-techie
 - ii. First name: any first name or optional
 - iii. Last name: any last name or optional

- f. Click Create.

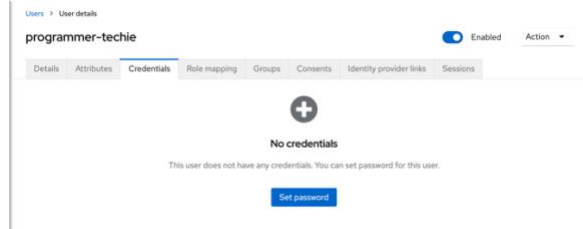


The 'Create user' form contains the following fields and controls:

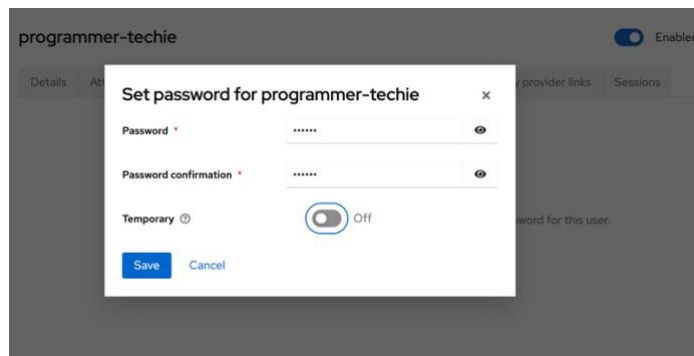
- Required user actions:** A dropdown menu with 'Select action'.
- Username:** A text input field containing 'programmer-techie'.
- Email:** An empty text input field.
- Email verified:** A toggle switch set to 'No'.
- First name:** An empty text input field.
- Last name:** An empty text input field.
- Groups:** A button labeled 'Join Groups'.
- Buttons:** 'Create' and 'Cancel' buttons at the bottom.

27. This user needs a password to log in. To set the initial password:

- Click Credentials at the top of the page.
- Fill in the Set password form with a password.
- Toggle Temporary Off so user does not need update this password at the first login.



The 'User details' page for 'programmer-techie' shows the 'Credentials' tab selected. It displays a '+ No credentials' message with a subtext: 'This user does not have any credentials. You can set password for this user.' and a 'Set password' button.



The 'Set password for programmer-techie' modal contains the following fields and controls:

- Password:** A text input field with masked characters (dots).
- Password confirmation:** A text input field with masked characters (dots).
- Temporary:** A toggle switch set to 'Off'.
- Buttons:** 'Save' and 'Cancel' buttons at the bottom.

- 28.