

1. For java 8
<https://www.keycloak.org/archive/downloads-13.0.1.html>
2. To start keycloak
standalone.bat -Djboss.http.port=8181
3. Download keycloak zip from below link
<https://www.keycloak.org/downloads>
[For java 11 and up](#)

Downloads 21.0.1

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

Server

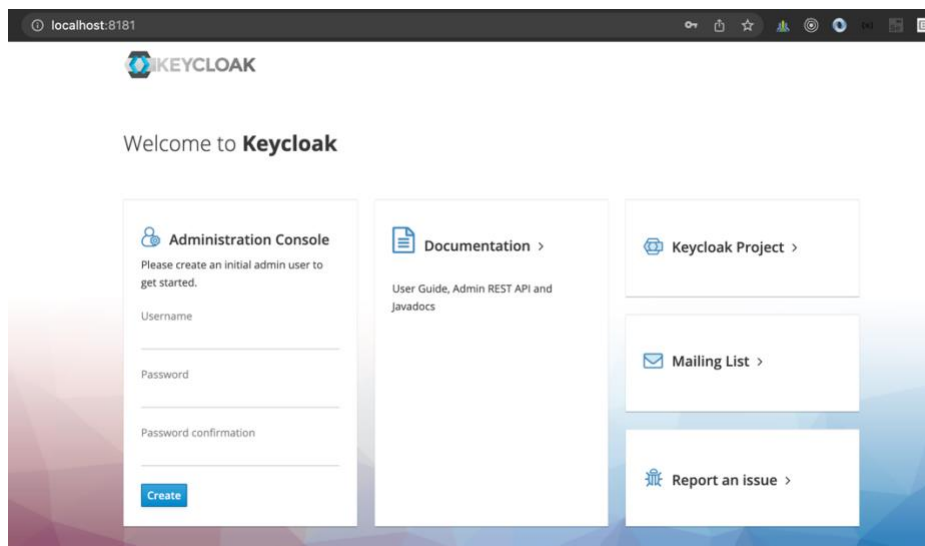
Keycloak	Distribution powered by Quarkus	ZIP (sha1) TAR.GZ (sha1)
Container image	For Docker, Podman, Kubernetes and OpenShift	Quay
Operator	For Kubernetes and OpenShift	OperatorHub

4. Extract the zip folder and open terminal / cmd from within the bin folder
5. 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.
```

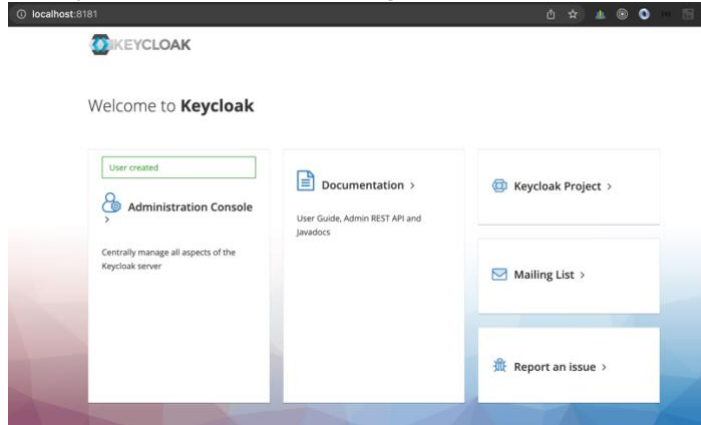
NOTE : Please wait it will take some time to start the server

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

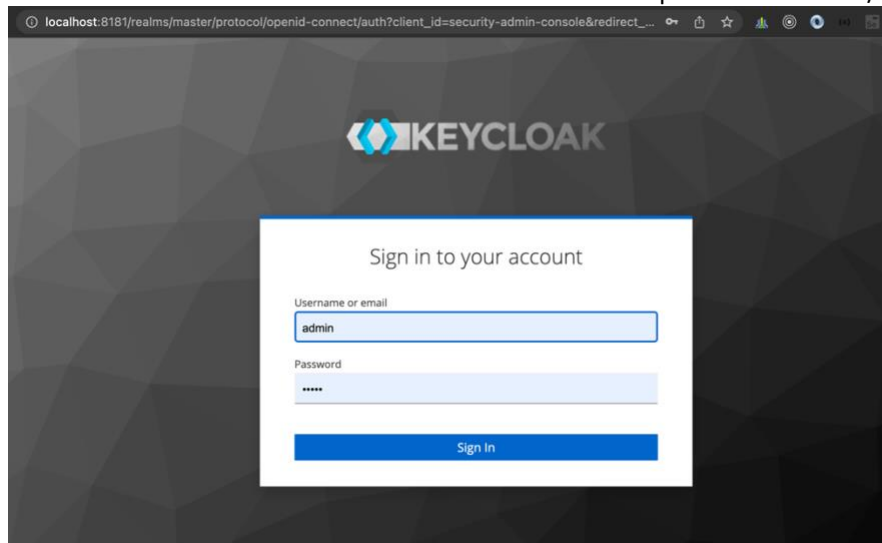


8. Enter username – admin, password – admin and click Create

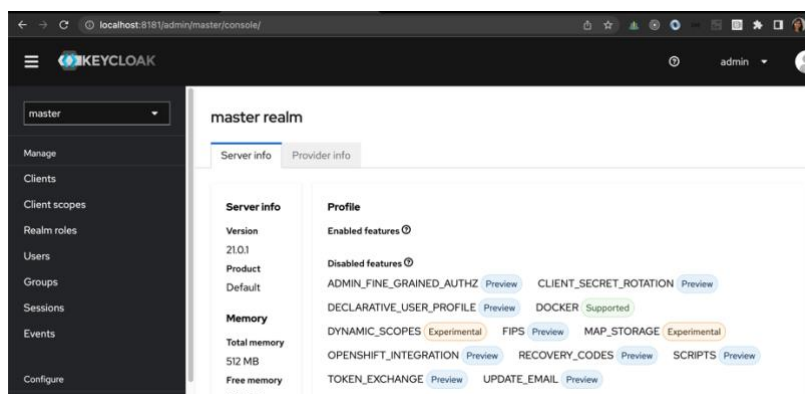
9. Once you create on click following dashboard is seen



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



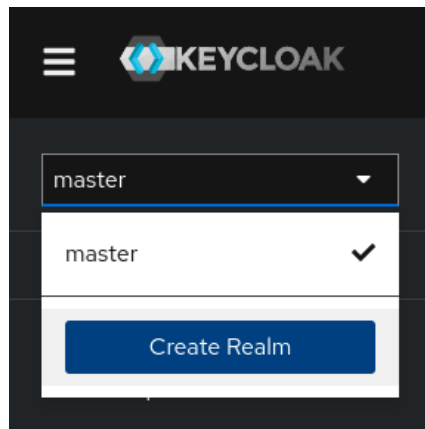
11. Once sign in -> dashboard looks as follows:



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

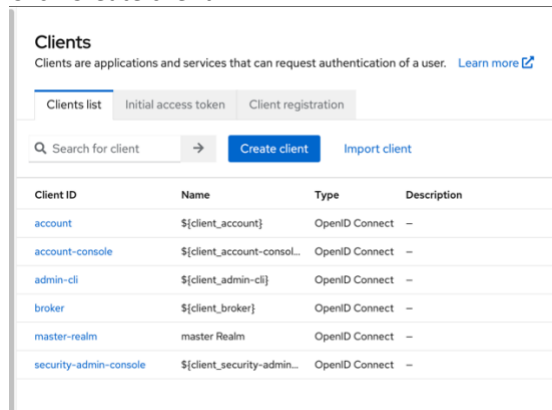
13. Use these steps to create the first realm.
- Open the Keycloak Admin Console.
 - Click the word master in the top-left corner, then click Create realm.
 - Enter **oauth-demo-realm** in the Realm name field.

- d. Click Create.

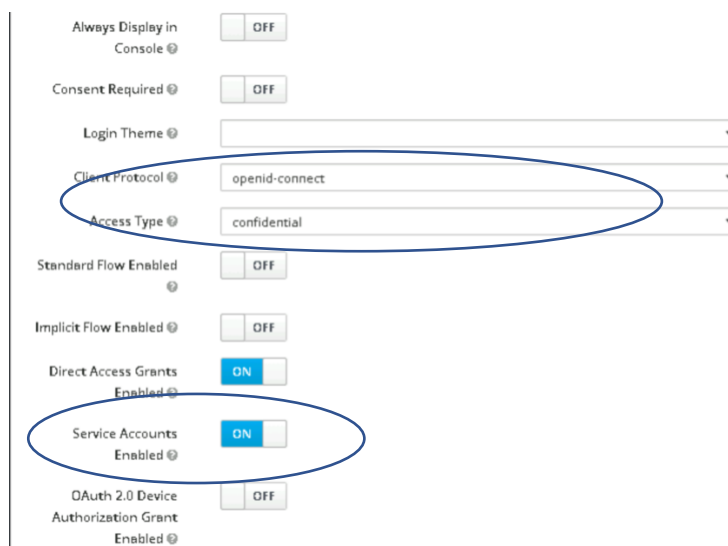


14. Secure the first application

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



e. For java 8



- f. Fill in the form with the following values:

Client type OpenID Connect

Client ID oauth2-client-credentials

Name

Description

Always display in UI ☐ Off

[Next](#) [Back](#) [Cancel](#)

- g. Click Next – select the following

Client authentication ☒ On

Authorization ☐ Off

Authentication flow

☐ Standard flow ☒ Direct access grants

☐ Implicit flow ☒ Service accounts roles

☐ OAuth 2.0 Device Authorization Grant

☐ OIDC CIBA Grant

[Next](#) [Back](#) [Cancel](#)

- h. Click Next and Save

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

oauth2-client-credentials OpenID Connect ☒ Enabled [Action](#)

Client Authenticator Client Id and Secret

[Save](#)

Client secret [Regenerate](#)

15. Create a spring boot application with following dependencies

<dependency>

<groupId>org.springframework.boot</groupId>

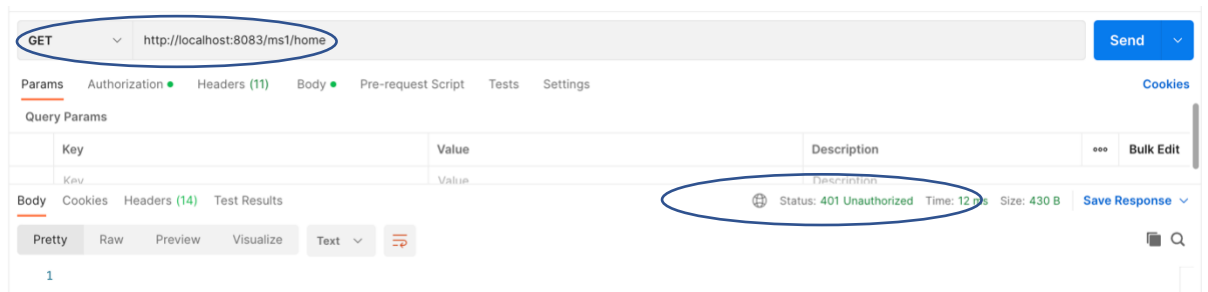
<artifactId>spring-boot-starter-web</artifactId>

```

</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>

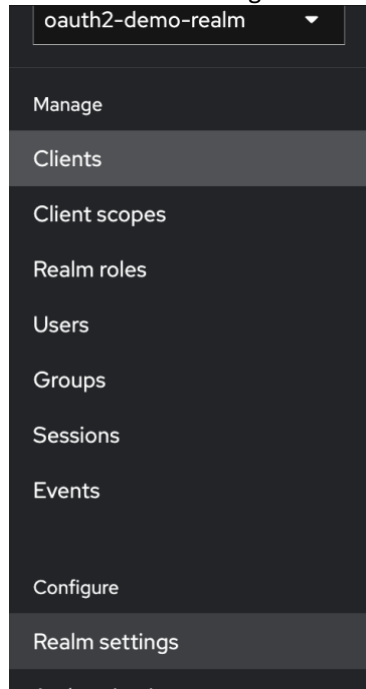
```

16. The project will be available on github
17. Start the spring boot project. Once successfully started open postman and hit the rest controller.
18. Should get unauthorized as follows:

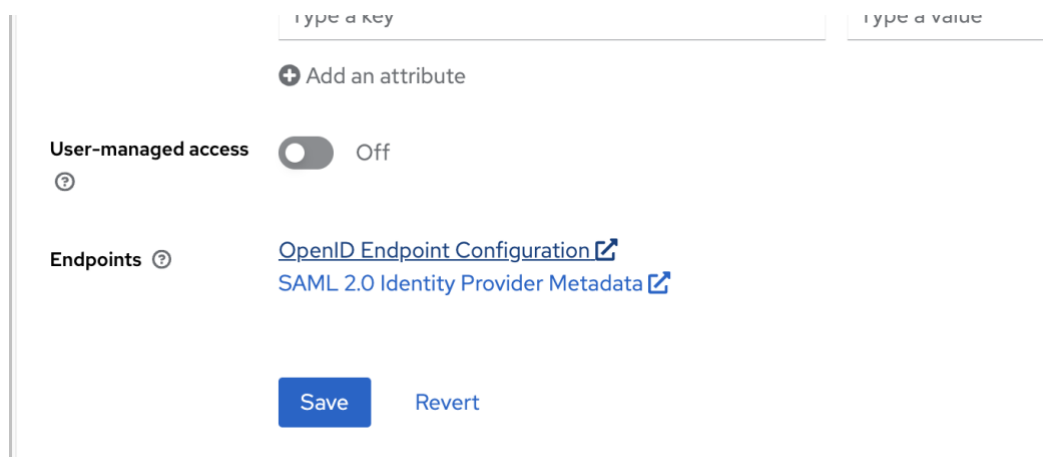


19. Need to get access token before hitting the url.
20. To access the configuration created by keycloak to register oauth-resource server

- a. Click on Realm Settings



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



21. The json looks as follows:

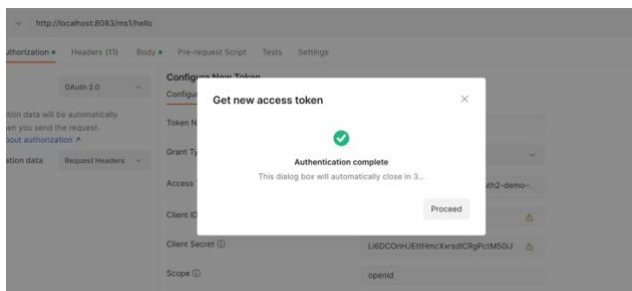
Issuer , jwks-uri and token_endppoint are what we will need

```
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_html": true,
  "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"
  ]
}
```

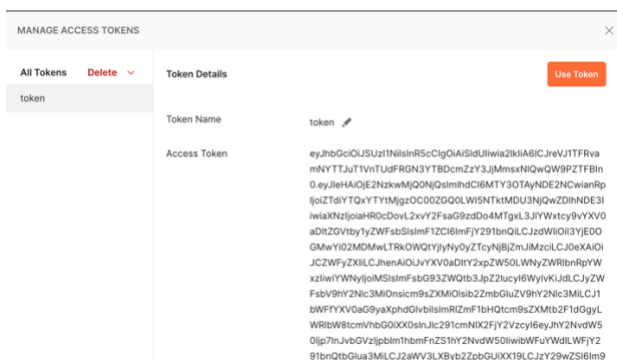
22. Open POSTMAN to get the access token to successfully connect to microservice
23. Enter details as follows

- 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

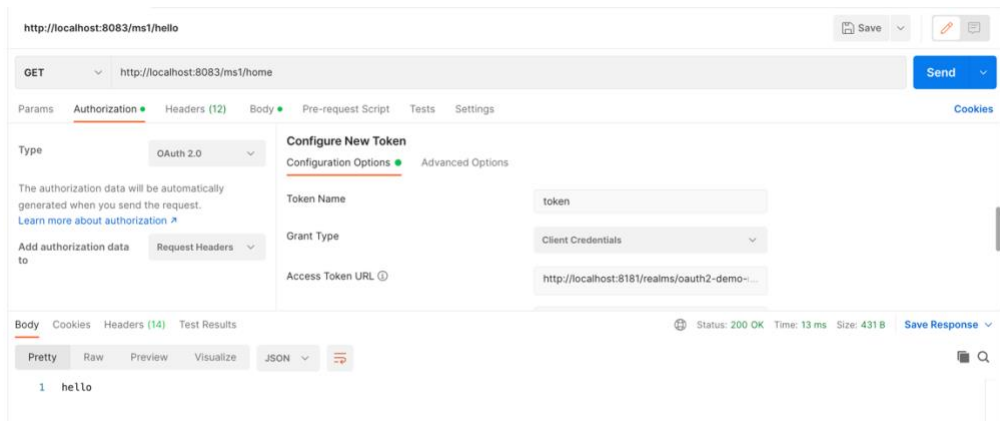
24. Should get Authentication Complete as follows:



25. Token looks as follows: Click Use Token



26. Now should get success and a hello message



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

28. 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 screenshot shows the 'Create user' form in the Keycloak Admin Console. The fields are filled as follows:

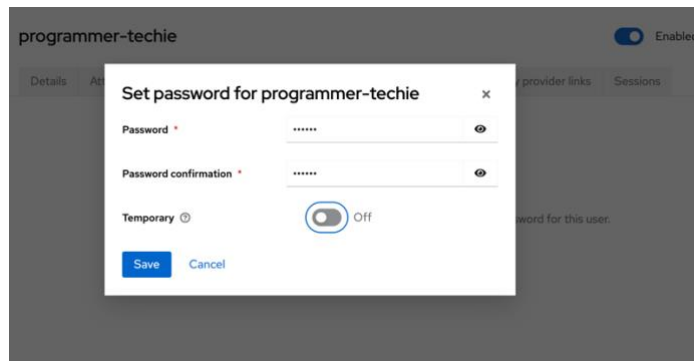
- Required user actions:** Select action
- Username:** programmer-techie
- Email:** (empty)
- Email verified:** No (toggle switch)
- First name:** (empty)
- Last name:** (empty)
- Groups:** Join Groups (button)

At the bottom, there are 'Create' and 'Cancel' buttons.

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

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

The screenshot shows the 'programmer-techie' user details page in the Keycloak Admin Console. The 'Credentials' tab is selected, and the page displays a message: "No credentials. This user does not have any credentials. You can set password for this user." Below the message is a 'Set password' button.



YOUTUBE LINKS:

1. <https://youtu.be/t9O99l4gjAc> - spring security basics - oauth
2. <https://youtu.be/mPPhcU7oWDU> - spring boot microservice crash course
3. <https://youtube.com/playlist?list=PLqg-6Pq4lTTYTEooakHchTGglSvkZAJnE> – complete security playlist [Kaushik – microservices playlist]
4. <https://youtu.be/996OiexHze0> - Best explanation for understanding oauth terminologies [OKTA]
- 5.