

OpenSource Keycloak Configuration Procedure

Introduction

Authorization and authentication are needed for medical devices. There are many providers online such as Azure App Service. However, OpenSource Keycloak is the ideal option for less configuration adoption in both on-premises and cloud settings.

Description

Keycloak is an open source software product to allow single sign-on with identity and access management aimed at modern applications and services.

The features of Keycloak include:

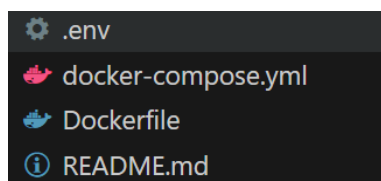
- User registration
- Social login
- Single sign-on/sign-off across all applications belonging to the same realm
- Two-factor authentication
- LDAP integration
- Kerberos broker
- Multitenancy with per-realm customizable skin
- Custom extensions to extend the core functionality

There are two main components of Keycloak:

- Keycloak server, including the API and graphical interface.
- Keycloak application adapter: a set of libraries to call the server.[8]

Steps for configuring Keycloak

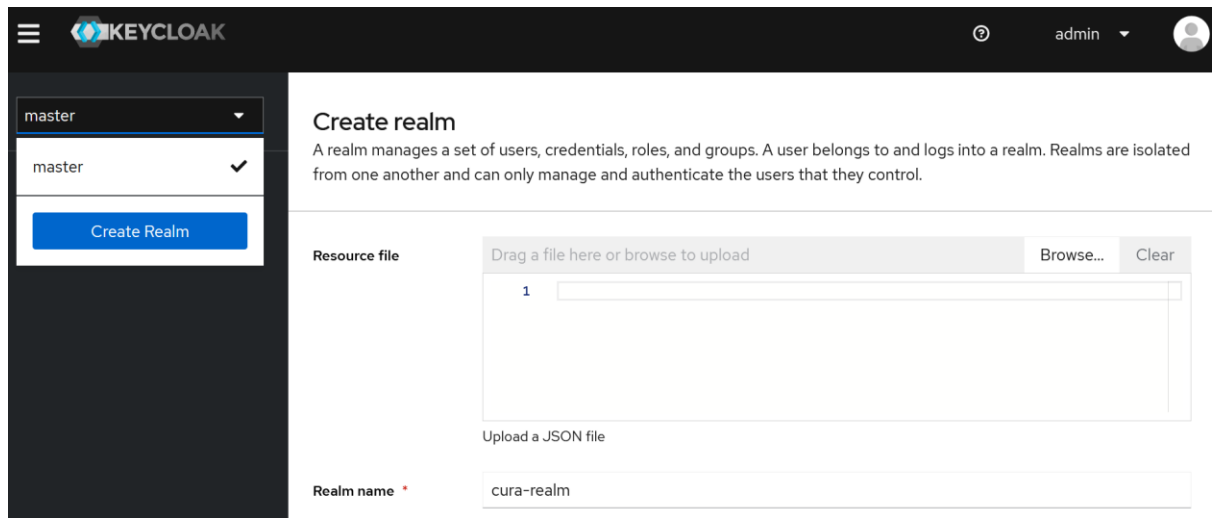
- 1) Ensure that the following files exist in your project folder.



- 2) Run the command “docker compose up” in terminal to download an image of Keycloak and run an instance of it as container.

```
PS C:\Users\vaitesswar\Desktop\keycloak new> docker compose up
[+] Running 2/0
✓ Container keycloaknew-identity-db-1 Running 0.0s
Attaching to keycloaknew-identity-1, keycloaknew-identity-db-1
Aborting on container exit...
[+] Stopping 2/2
✓ Container keycloaknew-identity-db-1 Stopped
```

3) Create a new realm.



The screenshot shows the Keycloak 'Create realm' form. On the left, a sidebar contains a dropdown menu with 'master' selected and a 'Create Realm' button. The main area has a title 'Create realm' and a description: 'A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.' Below this, there is a 'Resource file' section with a drag-and-drop area and a 'Browse...' button. A 'Realm name' field is at the bottom, containing the text 'cura-realm'.

master

master

Create Realm

Create realm

A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.

Resource file

Drag a file here or browse to upload

Browse... Clear

1

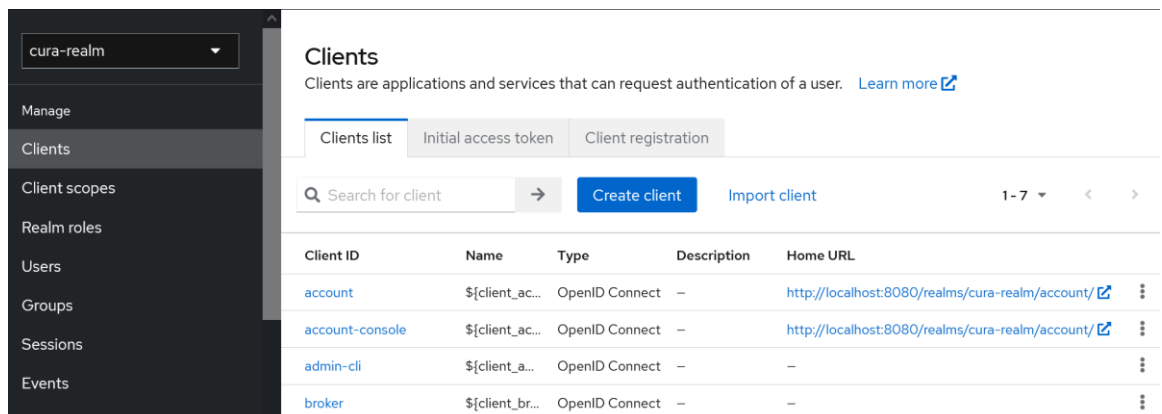
Upload a JSON file

Realm name *

cura-realm

4) Create a new client with the following details.

- Client Id: patient-api-client
- Name: patient-api-client
- Client authentication: **On**
- Authorization: **On**
- Valid redirect URIs: http://127.0.0.1:8001/*
- Valid post logout re-direct URIs: http://127.0.0.1:8001/*
- Web origins: <http://127.0.0.1:8001>



The screenshot shows the Keycloak 'Clients' page for the 'cura-realm'. The left sidebar has a dropdown menu with 'cura-realm' selected and a list of navigation items: 'Manage', 'Clients', 'Client scopes', 'Realm roles', 'Users', 'Groups', 'Sessions', and 'Events'. The main area has a title 'Clients' and a description: 'Clients are applications and services that can request authentication of a user. Learn more'. Below this, there are tabs for 'Clients list', 'Initial access token', and 'Client registration'. A search bar and a 'Create client' button are present. A table lists the existing clients.

cura-realm

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Clients

Clients are applications and services that can request authentication of a user. [Learn more](#)

Clients list Initial access token Client registration

Search for client → Create client Import client 1-7 < >

Client ID	Name	Type	Description	Home URL
account	\$_client_ac...	OpenID Connect	—	http://localhost:8080/realms/cura-realm/account/
account-console	\$_client_ac...	OpenID Connect	—	http://localhost:8080/realms/cura-realm/account/
admin-cli	\$_client_a...	OpenID Connect	—	—
broker	\$_client_br...	OpenID Connect	—	—

- 5) Go to Clients -> Roles. Create a new role with the following details.
- Role name: doctor-role

cura-realm

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Clients > Client details

patient-api-client OpenID Connect Enabled Action

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

< Settings Keys Credentials Roles Client scopes Authorization Service accounts >

Search role by name → Create role

Role name	Composite	Description
uma_protection	False	—

1-1 < >

- 6) Create a new group with the following details.
- Name: doctor-group

cura-realm

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Search... →

☐ Exact search

0-0 < >

Groups

A group is a set of attributes and role mappings that can be applied to a user. You can create, edit, and delete groups and manage their child-parent organization. [Learn more](#)

No groups in this realm

You haven't created any groups in this realm. Create a group to get started.

Create group

- 7) Under "Role mapping" tab, assign patient-api-client (doctor role) to this group.

cura-realm

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Search... →

☐ Exact search

1-1 < >

doctor... >

1-1 < >

Groups > Group details

doctor-group Action

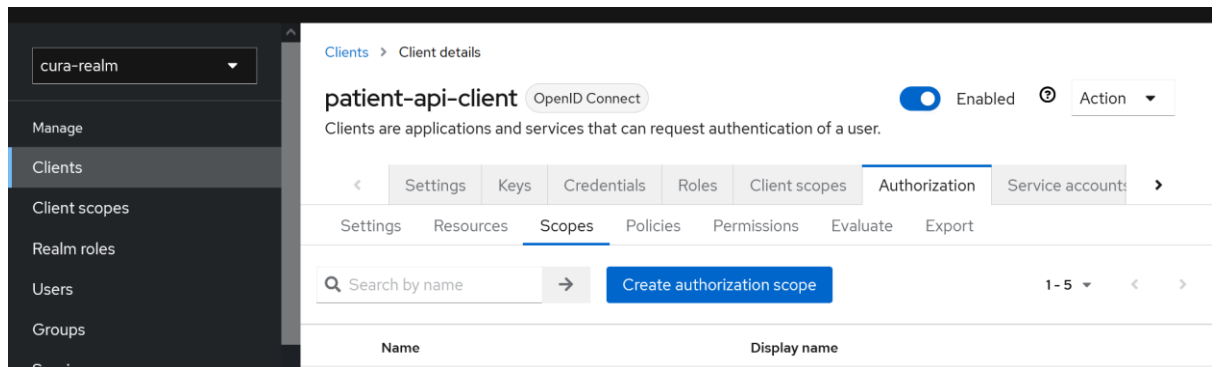
Child groups Members Attributes Role mapping

Search by name → ☒ Hide inherited roles

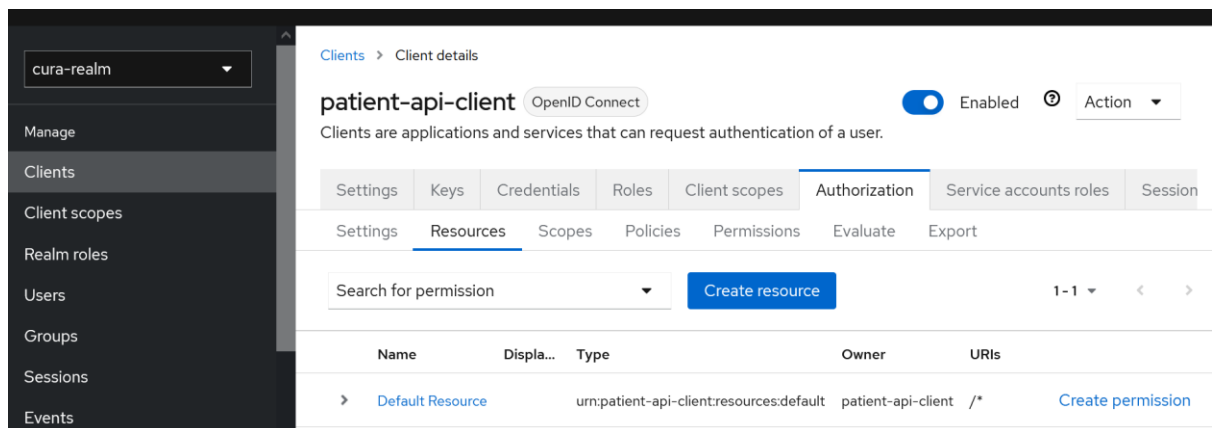
Assign role Unassign

<input type="checkbox"/> Name	Inherited	Description
<input type="checkbox"/> patient-api-client	doctor-role	False

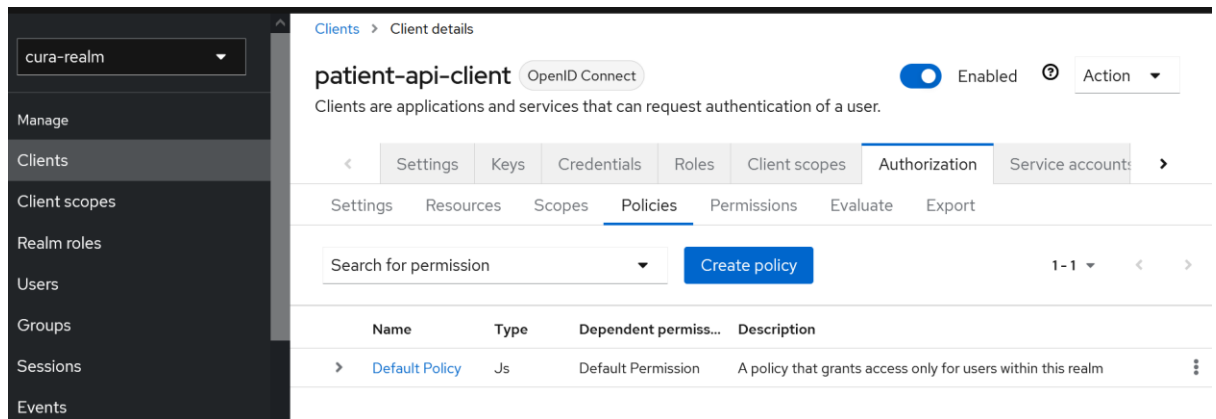
- 8) Go to Clients -> Authorization -> Scopes. Create a list of new authorization scopes as follows.
- delete-scope
 - register-scope
 - save-scope
 - update-scope
 - view-scope



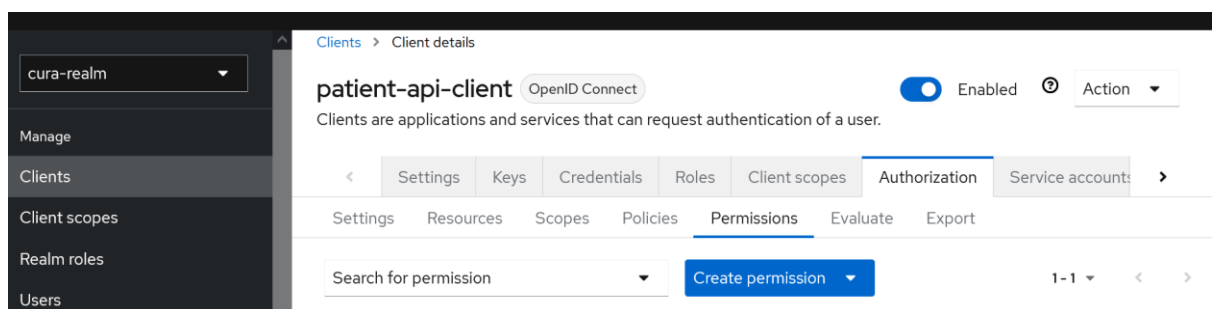
- 9) Go to Clients -> Authorization -> Resources. Create a new resource with the following details.
- Name: appointment-resource
 - Authorization scope: save-scope, view-scope, update-scope



- 10) Go to Clients -> Authorization -> Policies. Create a new policy with the following details.
- Choose policy type: Role
 - Name: doctor-role-policy
 - Description: doctor-role-policy
 - Add roles: doctor-role (patient-api-client)

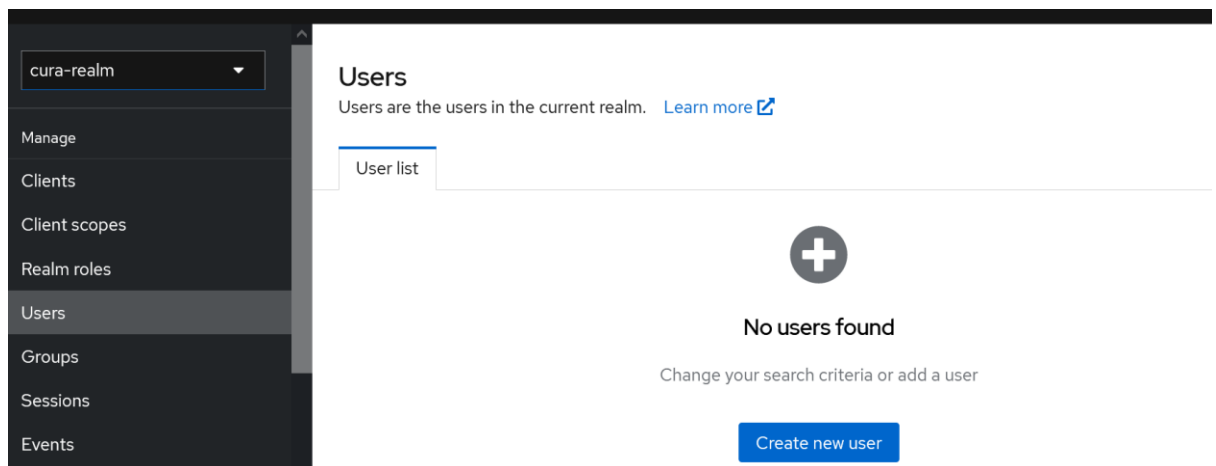


- 11) Go to Clients -> Authorization -> Permissions. Create new permissions with the following details.
- Permission 1
 - Type: Create scope-based permission
 - Name: appointment-advance-permission
 - Resources: appointment-resource
 - Authorization scopes: save-scope, update-scope
 - Policy: doctor-role-policy
 - Permission 2
 - Type: Create scope-based permission
 - Name: appointment-basic-permission
 - Resources: appointment-resource
 - Authorization scopes: view-scope
 - Policy: doctor-role-policy

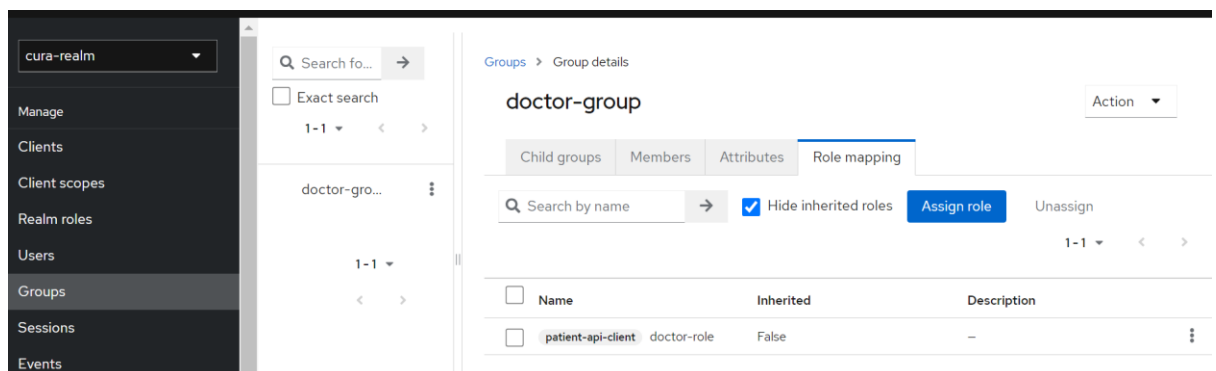


12) Create a new user with the following details.

- a. Username: doctor1
- b. Email: doctor1@example.com
- c. Change “Email verified” to “Yes”.
- d. Join “doctor-group”
- e. Create -> doctor1 -> Credentials
- f. password: *****
- g. Set temporary off



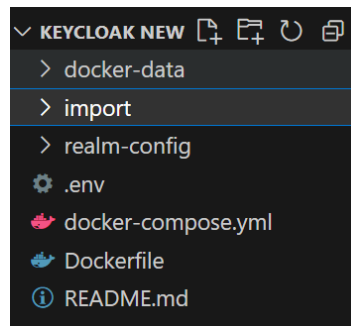
13) Go to Groups -> doctor-group -> Role Mapping. Assign “doctor role” in patient-api-client to this group.



14) The above-mentioned steps are for configuring Keycloak for backend. For frontend, only perform step 4 (i.e., create a new client) with the following details.

- a. Client Id: patient-app-client
- b. Name: patient-app-client
- c. Client authentication: **Off**
- d. Authorization: **Off**
- e. Valid redirect URIs: http://localhost:3000/*
- f. Valid post logout re-direct URIs: http://localhost:3000/*
- g. Web origins: <http://localhost:3000>

- 15) Create a new folder called “import” in directory.



- 16) Run “docker ps” command to get container id in VS Code terminal.

```
PS C:\Users\vaitesswar\Desktop\keycloak new> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
fd1c15de0da4	internalproject.azurecr.io/patient-app-identity:latest	"/opt/keycloak/bin/k..."	3 hours ago
Up 3 hours	0.0.0.0:8080->8080/tcp, 0.0.0.0:8443->8443/tcp	keycloaknew-identity-1	
ef9b18563b30	postgres:14	"docker-entrypoint.s..."	3 hours ago
Up 3 hours (healthy)	0.0.0.0:5433->5432/tcp	keycloaknew-identity-db-1	

- 17) Run “docker exec -it <container id> /bin/sh” command to access the docker container. In some cases, you might have to run “/bin/bash” instead if this command does not work.

```
PS C:\Users\vaitesswar\Desktop\keycloak new> docker exec -it fd1c15de0da4 /bin/sh
sh-5.1$
```

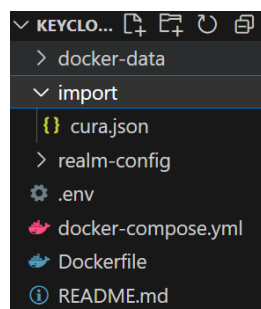
- 18) Run “cd /opt/keycloak/bin/” command.

```
sh-5.1$ cd /opt/keycloak/bin/
sh-5.1$
```

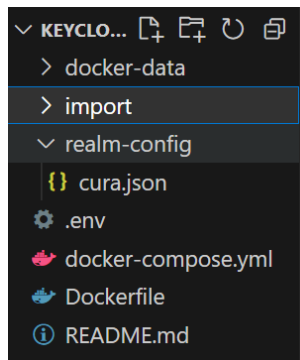
- 19) Run “cd /opt./kc.sh export --file /opt/keycloak/import/<realm-name>.json --realm <realm-name>.-realm” as shown below to get the realm configuration in a json file.

```
sh-5.1$ ./kc.sh export --file /opt/keycloak/import/cura.json --realm cura-realm
```

- 20) Ensure that the realm configuration file exists in “import” folder.

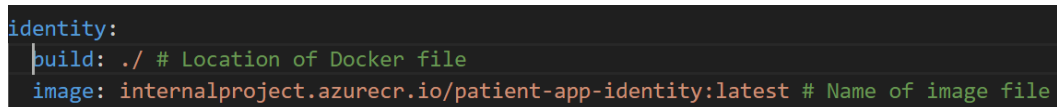
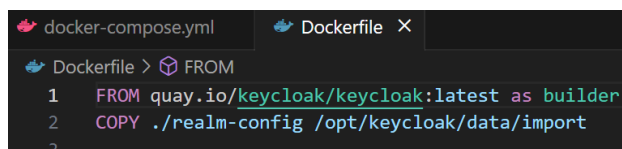


- 21) Shift the realm configuration file to “realm-config” folder. Delete both “import” and “docker-data” folders as they are unnecessary.



- 22) Delete “Default Policy” and “Default Permission” sessions in the realm configuration json file.

- 23) Uncomment all lines in all files. Delete all images and containers.



- 24) Run the command “docker compose up” in terminal to download an image of Keycloak and run an instance of it as a container using the saved realm configuration in realm-config folder.

