**14.Securing eazybank app using oauth2 and keycloak**

Introduction to OAUTH2 flow inside EazyBank web App

In the previous section, we have created a simple Spring Boot application

and try to secure it with the help of social logins like GitHub and Facebook.

But I can't use the same approach of social logins to secure my EazyBankSpringBoot web application. The reason is very simple.

Whatever auth servers that are available with the help of social logins like GitHub, Facebook, LinkedIn, Twitter, so all these auth servers, they have a very serious drawback or limitation due to which these auth servers, they can't be used inside an enterprise organization. The limitation is we will not have any control on the social login auth servers.

We can't create an role or we can't create an authority based upon our business requirements inside these auth servers. Whatever user related information

that we are getting from the social logins, we just have to accept them

and build our business logic using them. But in real applications, we can't live

with these kind of restrictions.

We may want to register thousands of end users and each of the end user may have their own authorities and roles defined inside the auth server. That's why the enterprise applications are bigger organizations, they are going to build their own auth server where they have complete control.

As a next step, we are going to enhance our EazyBankSpringBoot web application

with the OAuth 2 and by building our own auth server.

Resource Server:

So if you look at the setup that I have on this slide, first, we are going to convert our Spring Boot application into a Resource Server. So whatever Spring Boot application that we have right now, it has all the end user resources like accounts, loans, cards.

Since we want to secure all these end user resources, so the Spring Boot application server, it is going to act as a Resource Server.

client application:

And coming to the client application, I'm going to consider two types of client applications. The very first one you already know, which is Angular application.

We are going to consider this UI application as a client application inside the OAuth 2 flow.

I'm going to consider the Postman as an client application where I'm going to invoke the APIs available inside the Resource Server.

auth server using KeyCloak:

Now, coming to the auth server, we are not going to have any social login auth servers because they have so many restrictions.

we are going to build an auth server with the help of KeyCloak.

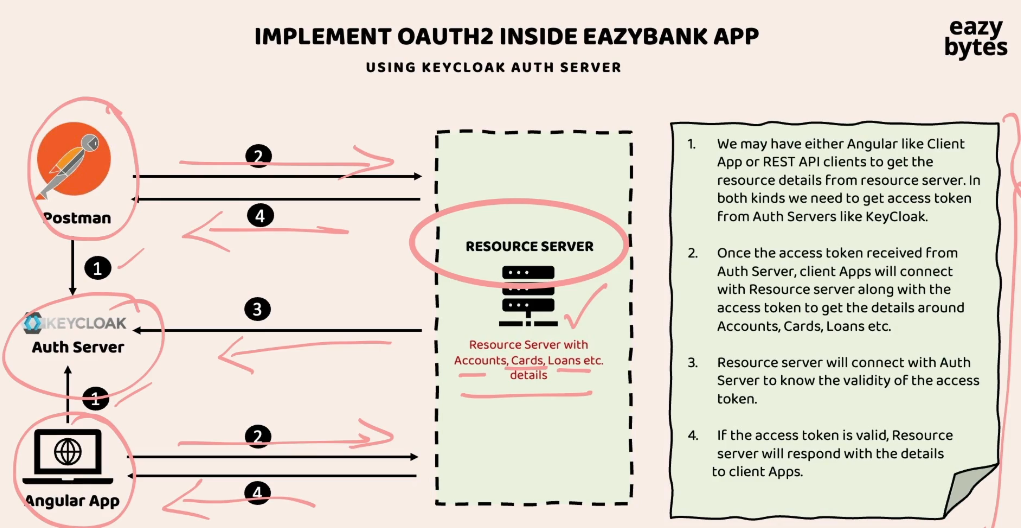
I'm going to give a good introduction about KeyCloak and we are going to set up a proper auth server with the help of KeyCloak.

In the coming sections, we're also going to build our own auth server with the help of Spring Auth Server Library.

Flow:

the client applications will reach out to the KeyCloak auth server to get an access token. When they're trying to get the access token from the auth server, they can involve end user as well if they're trying to follow the AuthorizationCode

AuthorizationCodeGrantType flow or PKCE flow. Otherwise, if they're simply following the ClientCredentialsGrantType flow, they don't have to involve any end user.

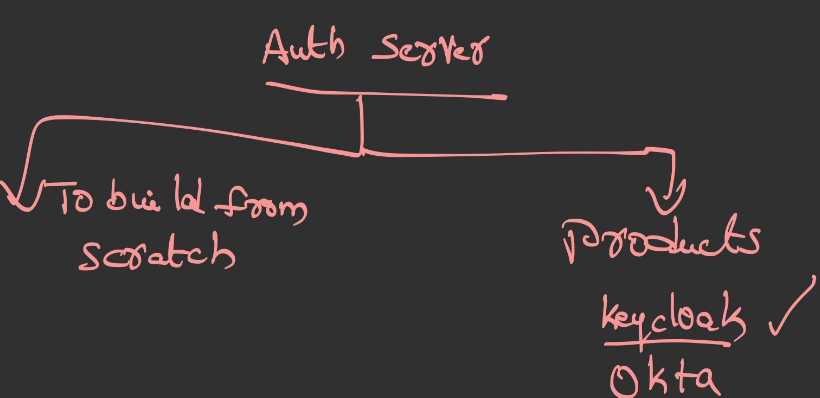


So once the access token is received from the auth server, these client applications,

they're going to send the access token to the Resource Server as part of the request 2. Once the access token is received by my Resource Server, the Resource Server, it is going to validate if the access token is valid or not in the step three.

So this validation, it is going to perform by connecting with the auth server, which is KeyCloak in our scenario. If the provided access token is valid, at last, my Resource Server is going to respond with the secured resources as part of the step four.

So as part of the step four, the client applications, they're going to receive the response from the Resource Server.



So whenever an organization is looking to build an auth server, they will have two options. The very first option is so to build from scratch.

The other option is they can leverage the products that are supporting the OAuth 2 and Open ID standards. So these products are like KeyCloak, which is an open source product, and similarly we have Okta. All the cloud providers like AWS, Azure, GCP, they also have products around the OAuth 2 and OpenID Connect.

But inside this course I'm going to use KeyCloak, which is the open source product.

Here you may have a question, which is "How organizations are going to choose

one of these option?" It all depends upon the organization requirements.

If an organization has huge manpower and budget and time, then they can build their own auth server from scratch by using libraries like Spring Auth Server.

Otherwise, they're going to adopt the readily available products like KeyCloak and Okta.

We're going to explore both of these two options inside this course.

First, with the help of KeyCloak, I'm going to show you the OAuth 2 flow

inside the EazyBank application.

In the coming sections, I'm also going to show you how to build our own auth server

with the help of the Spring Auth Server Library.

Introduction to KeyCloak Auth Server

Instructor: When we are talking about OAUTH 2 and OpenID framework for the very first time, I told you that OAUTH2 or OpenID, They are just standards or specifications. They're not actually a proper product, which can be leveraged by any organization.

That's why they are simply guidelines of standards, or specifications.

Based upon these OAUTH 2 standards or specifications or guidelines.

Bigger organizations like Facebook, GitHub, LinkedIn, Google, Netflix, Amazon,

they started building their own Auth server for their own internal purpose.

The most famous products:

The very first product I would recommend anyone to use is Keycloak,

because this is an open source product which is built and maintained by the Cloud Native Computing Foundation. Similarly, we also have some commercial products like Okta.

Okta is very famous commercial product. Any organization who are ready to pay few hundreds or few thousands of dollars, they can happily adopt these Okta.

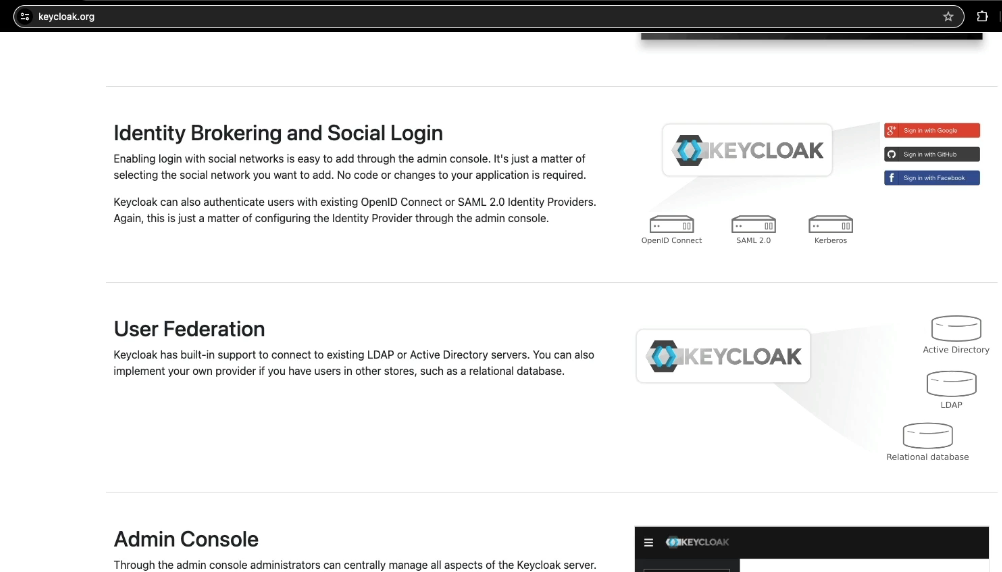
We also have other product with the name AuthO. This AuthO product is recently acquired by Okta itself. That's why you are able to see here AuthO by Okta,

this is also one of the famous readily available

Auth servers that organizations can leverage. At last, we also have Amazon Cognito from the AWS. If your organization is using AWS component to a great extent,

then using these product will be a good decision, because it has a good integration with the AWS components.

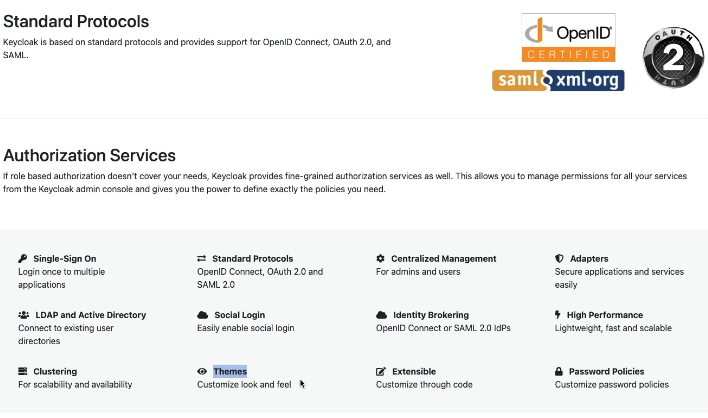
Even if someone is not using AWS, they can still use this product.



But inside this course, I'm going to use Keycloak. The reason is very simple.

It is an open source. With the help of this Keycloak, we should be able

to set up the Auth server inside our local, development and production system.



So if you try to understand what is there inside this website, the very first highlighted information here is, it is an open source identity and access management product.

It is going to support **single sign-on, Identity Brokering and Social Login.**

**User Federation. User Federation** means we can store all the user credentials

inside storage systems like relational database, LDAP, active directory, and we can point our Keycloak to these storage systems so that the Keycloak can perform the authentication and authorization based upon these configured storage systems.

And similarly, this product is also going to provide you admin console

where admins can set up some configurations related to the Auth servers.

It is also going to provide you Account Management console for normal users.

Next, here we have the most important information, which is it is going to support the standard protocols like Open ID Connect, OAuth 2.

And it is also going to support various authorization services. With the help of these Keycloak Auth server, we should be able to set up a cluster to handle any kind of traffic. If needed, we can also customize the login page of the Auth server based upon our own organization look and feel requirements.

But to get started with these Keycloak server first, we need to set up these inside our local system. Let's try to set up this Keycloak server.

Installation of KeyCloak server, setup admin account & realm

Let's try to set up Keycloak inside our local system.

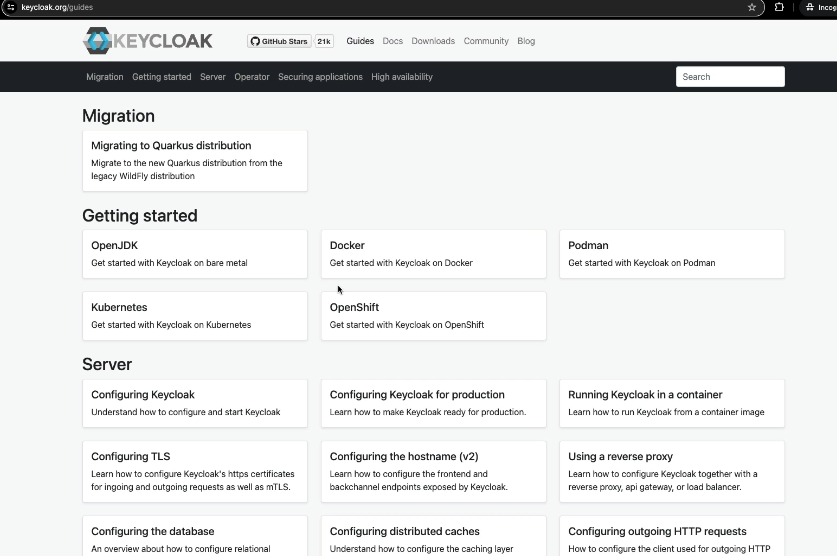
To know the installation steps, we need to click on this Get Started.

Inside these guides page you'll be having lot of information on how to set up Keycloak. You as a developer, you are not responsible to set up these Keycloak server.

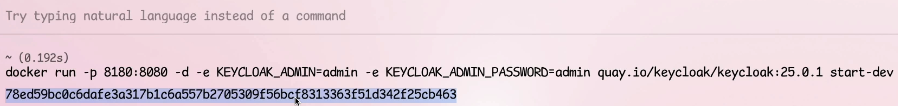
Either your DevOps team or platform team or deployment team, they're going to set up this Keycloak inside various environments and they're going to provide the URLs to the developers.

Using these URLs developers, they can start implementing the OAuth 2.0

inside their web applications.



Docker setup:

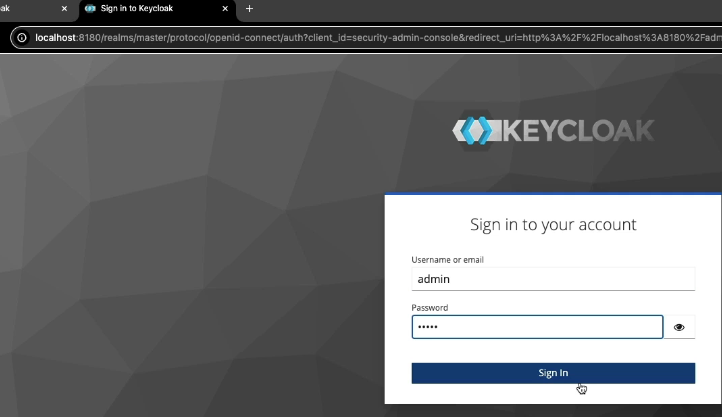


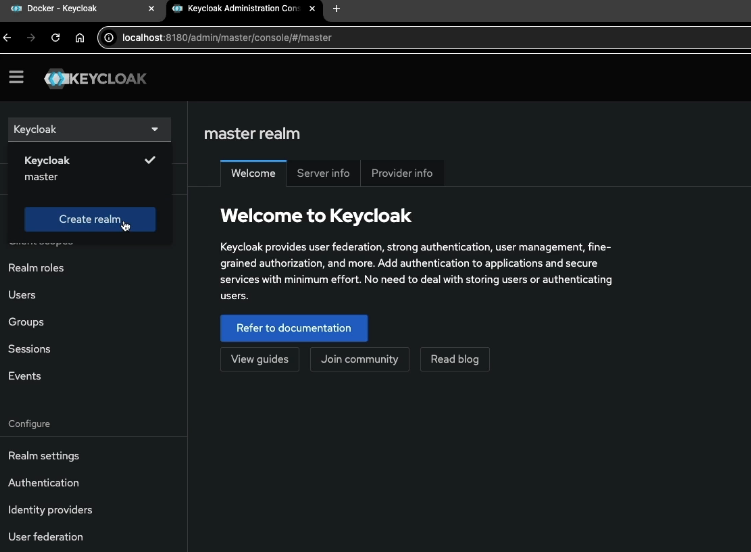
At the end of the command, we also have an option which is start-dev.

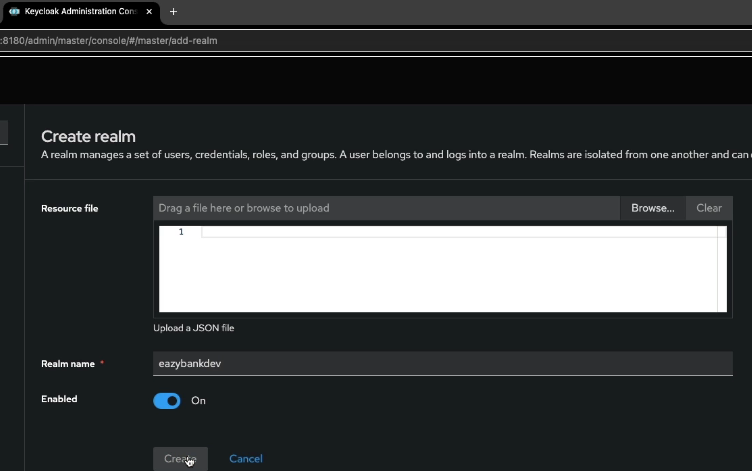
With this, we are telling to the Keycloak to start using a development profile or development mode. So whenever we start Keycloak using a development mode,

what is going to happen is it is going to have an embedded H2 database or in-memory database to store all the end user details and all other configurations that we are going to make.

So we should be fine with these in-memory database, but in real projects, your Keycloak admin, they're going to set up a proper storage system like MySQL database or LDAP or Active Directory based upon your organization requirements.







what is a realm?

You have details here. A realm is a space which is going to manage a set of users,

credentials, roles, and groups. This is very similar to the environments.

For example, if your organization have environments like dev, QA, production,

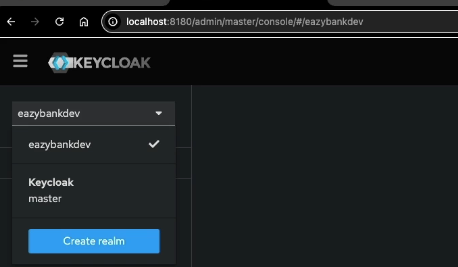
there is a very good chance we are not going to maintain the same users,

same credentials across multiple environments.

To support these kind of requirements or scenarios, every auth server that is built on top of OAuth 2.0, they're going to have a concept called **realm**.

So using the realm, we are going to create a space which is going to handle a specific environment. Let me provide the realm name, which is eazybankdev.

Using this realm, I'm going to maintain all the users, clients related to the eazybank application for the development environment.



Creating Client Credentials inside KeyCloak for API-API secured invocations

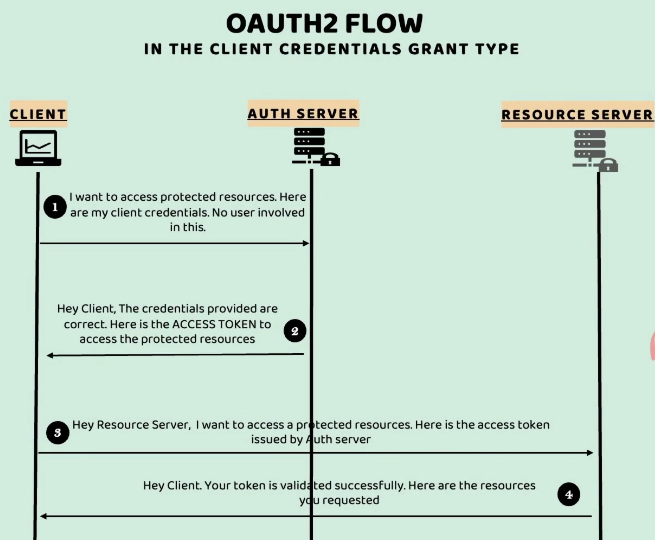
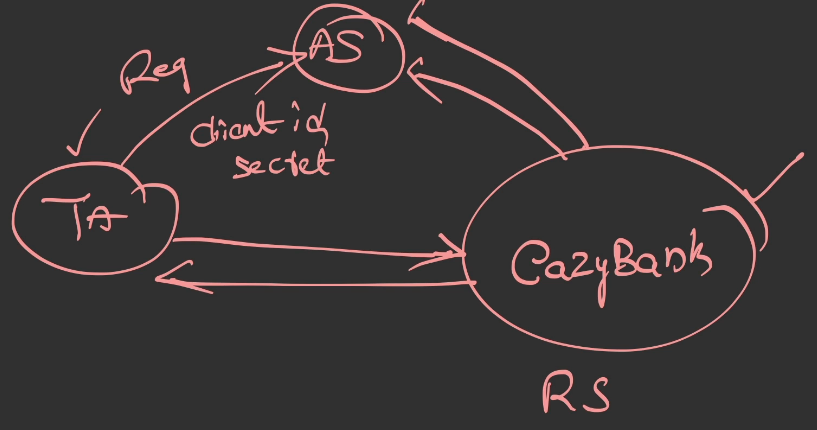
For now, let's focus on persisting client details inside the Keycloak auth server.

So the details of client application that are going to vary based upon

the OAuth2 grant type flow that you are trying to set up. We discuss many grant type flows, like authorization code grant type flow, PKCE grant type flow, client credentials grant type flow, refresh token grand type flow.

We're going to explore all these grant type flows one by one. Inside this lecture, let's try to focus on how to set up a client inside auth server whenever we are looking to leverage client credentials grant type flow.

Registering client using client credentials grant type flow: api to api communication

In these kind of scenarios, usually inside real projects, what is going to happen is,

whenever a third-party application or a backend application want to talk with the resource server, they're going to reach out to the leadership of this EazyBank application.

They will talk with them, they will explain the scenarios. Once proper approval is received from the leadership of the EazyBank application, the admin or the product owner of this EazyBank application, they're going to request Keycloak admin

to register the third-party application details and provide the client ID and client secret to this third-party application.

So this kind of registration process will happen behind the scenes,

and this is going to happen only once. Once the third-party application and the

client application registered with the auth server,

it can start sending requests to the EazyBank application by providing the proper access token.

Let's try to register a third-party application details inside Keycloak with the help of

Client Credentials Grant Type flow.

Login to Keycloak admin page, I'm going to enter the admin credentials.

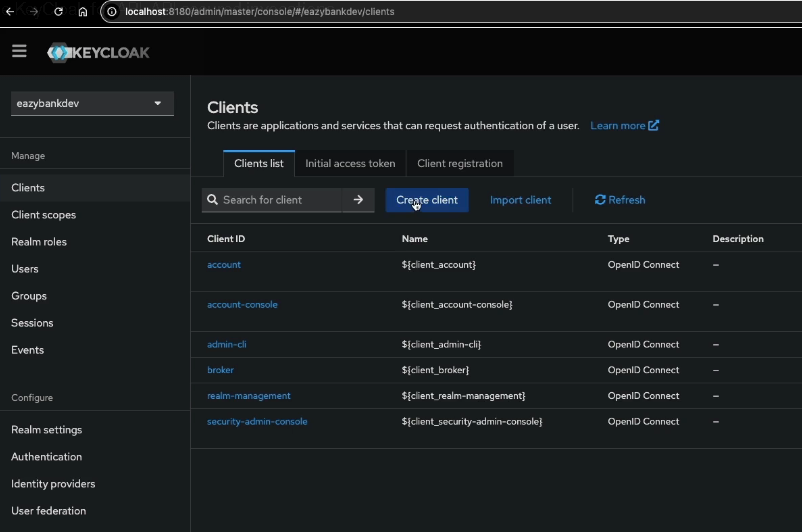
Once we click on this sign-in button, you can see by default, the Keycloak realm is selected.

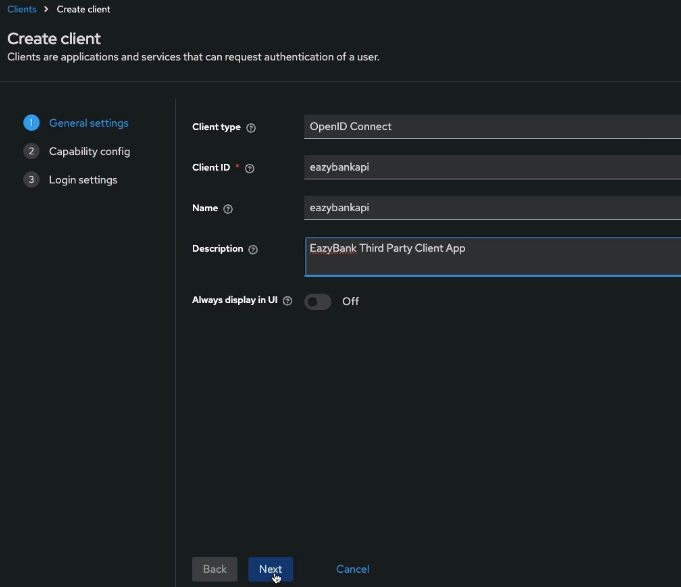
So please make sure you are selecting our eazybankdev as our realm.

Once you have selected this realm, you have an option here with the name "Clients".

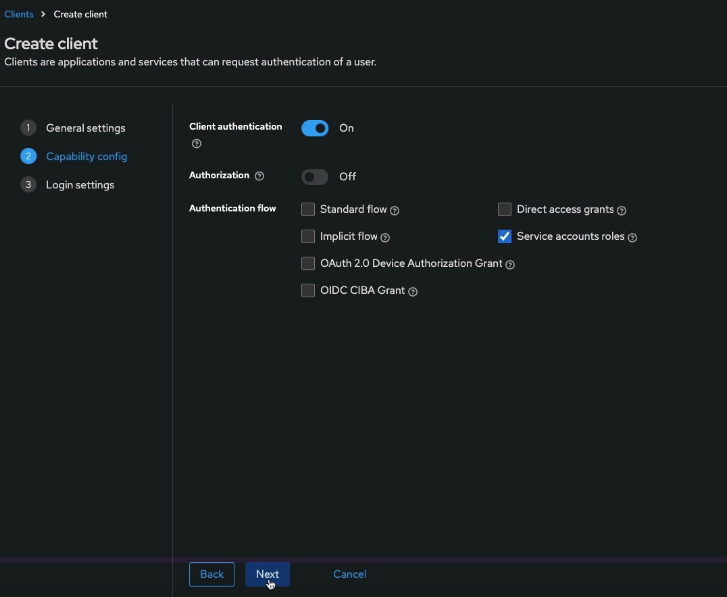
You need to click on the same. By default, these are all the various clients

that are available inside the Keycloak. But since we're looking to create our own client, you need to click on this "Create client".





Client id is like username here



we need to enable the client authentication. Because whenever the client application

wants to get an access token from the auth server, we want the client application to prove its identity. How it's going to prove its identity? By providing the client ID and client secret. That's why to enable the authentication of client, please make sure you have enabled this option.

Next, under the authentication flow, **by default, the standard flow and the direct access grants are selected.**

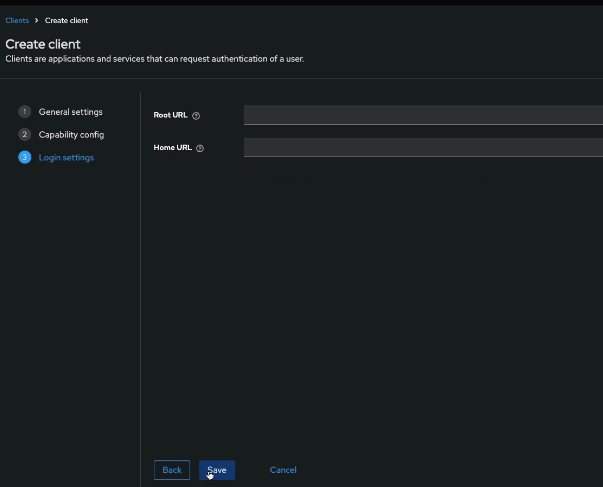
So if you try to click on this question mark after the standard flow, it is going to provide more details. So standard flow refers to the Authorization Grant Type flow

in the OAuth2 specification.

Similarly, Direct access grants means Resource Owner Password Credentials Grant.

We don't want to use both these grant type flows, for the client application that we are trying to create.

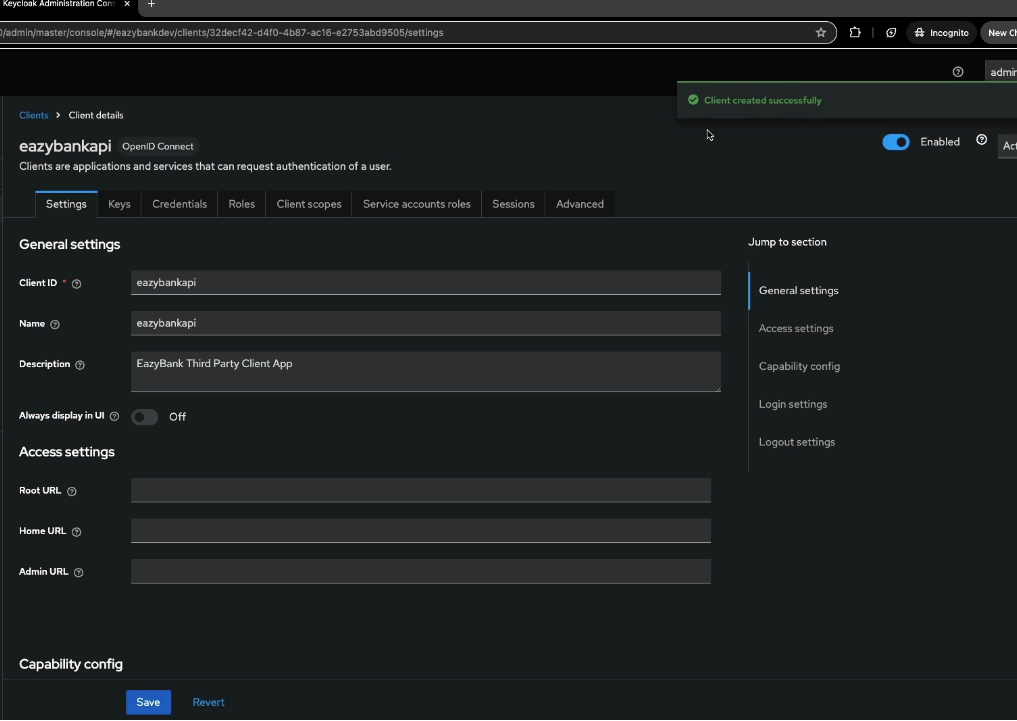
**So what is the grant type flow that we are looking? We are looking Client Credentials Grant Type flow. So if you try to click on this help button after the service accounts roles, you'll be able to see that service accounts roles means Client Credentials Grant Type flow.**

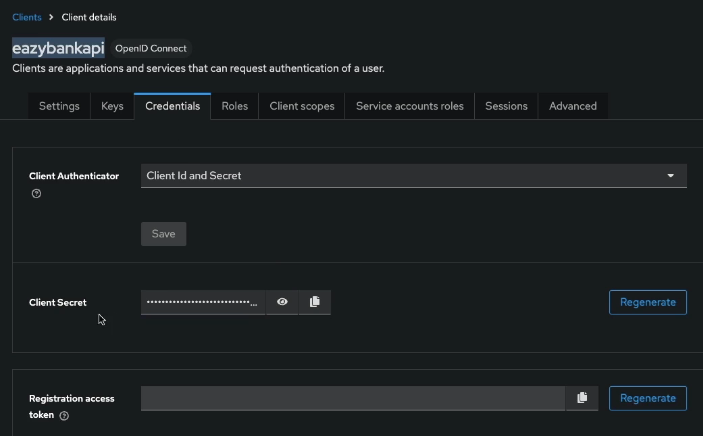


And you can ignore this root URL and home URL and click on the save button.

With this, you'll be able to see there is a message here, "Client created successfully",

and it is enabled.





As the next step, we want to know what is the secret for this client.

So you need to click on this credentials, and here you'll be able to see the secret,

or you can also copy the secret.

So this is the client ID, which is "eazybankapi", and this is the client secret that the client has to use whenever it wants to get an access token from this Keycloak auth server.