Kenny R N Wilson

KeyCloak

Introduction

THIS DOCUMENT COVERS

- Introduction

Cheat Sheet

This gives details on end points on the server

http://localhost:8080/auth/realms/master/.well-known/openid-configuration

Kenny R N Wilson

React and ASP.NET Core

This example shows how to use KeyCloak to secure a React front end and a .NET Core backend

Install KeyCloak.

The first step is to install KeyCloak and add an admin user as described here in the KeyCloak documentation.

https://www.keycloak.org/docs/latest/getting_started/index.html

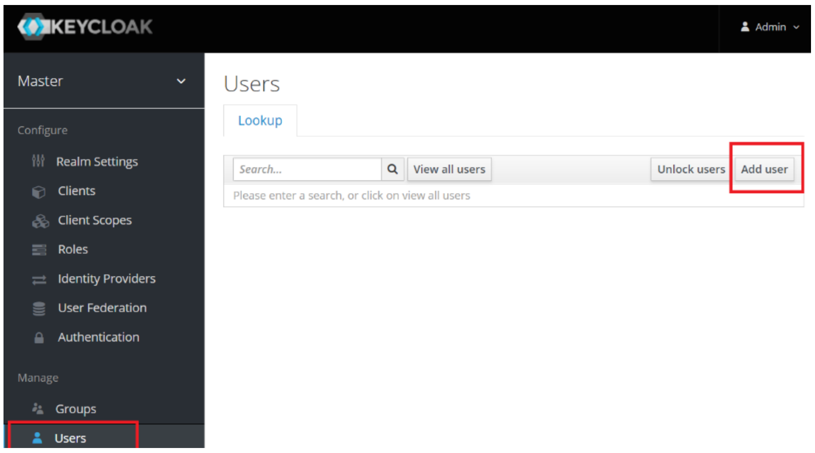
Configure KeyCloak.

For this tutorial we will just use the master realm.

ADD A USER.

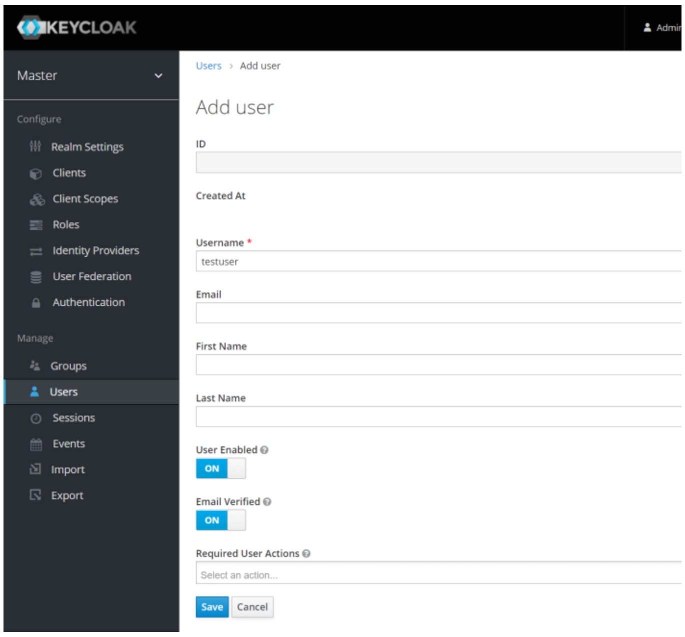
Open the Add user screen

We go to the Users tab and click add user.



Enter Name and turn on Email Verified

Now we enter the name as **testuser**, set **Email Verified** to “On” to indicate we do not need the user to verify the password we will set. Finally, we click save.



KEYCLOAK

Admin

Master

Configure: Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User Federation, Authentication

Manage: Groups, Users, Sessions, Events, Import, Export

Users › Add user

Add user

ID:

Created At:

Username *: testuser

Email:

First Name:

Last Name:

User Enabled: ON

Email Verified: ON

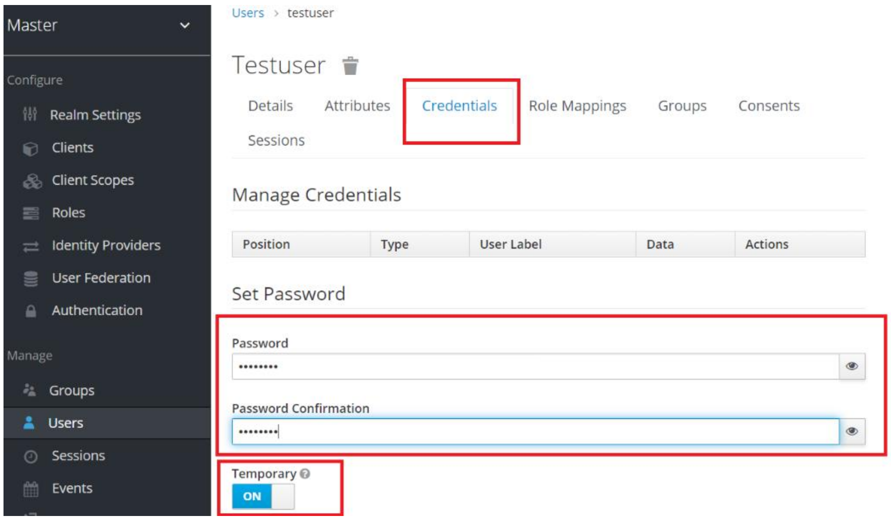
Required User Actions: Select an action...

Save | Cancel

Now go to the credential

Set the password for the user

Open the Credentials tab for the user and enter the password. Set the Temporary flag to false so we do not have to update it on first logon. Enter the password as t**estuser.**



Master

Configure: Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User Federation, Authentication

Manage: Groups, Users, Sessions, Events

Users › testuser

Testuser

Details | Attributes | Credentials | Role Mappings | Groups | Consents | Sessions

Manage Credentials

Position	Type	User Label	Data	Actions

Set Password

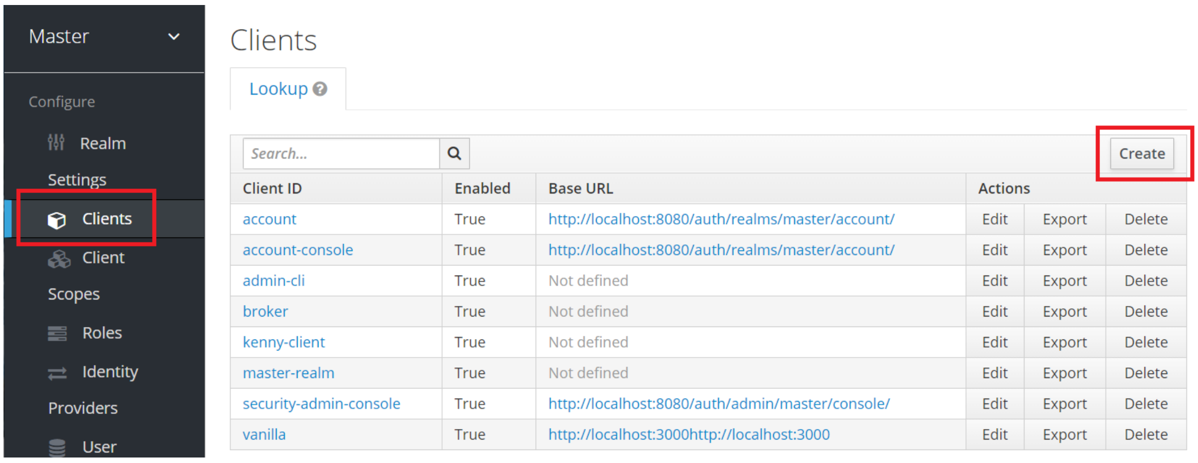
Password: ••••••••

Password Confirmation: ••••••••

Temporary: ON

Add a Client.

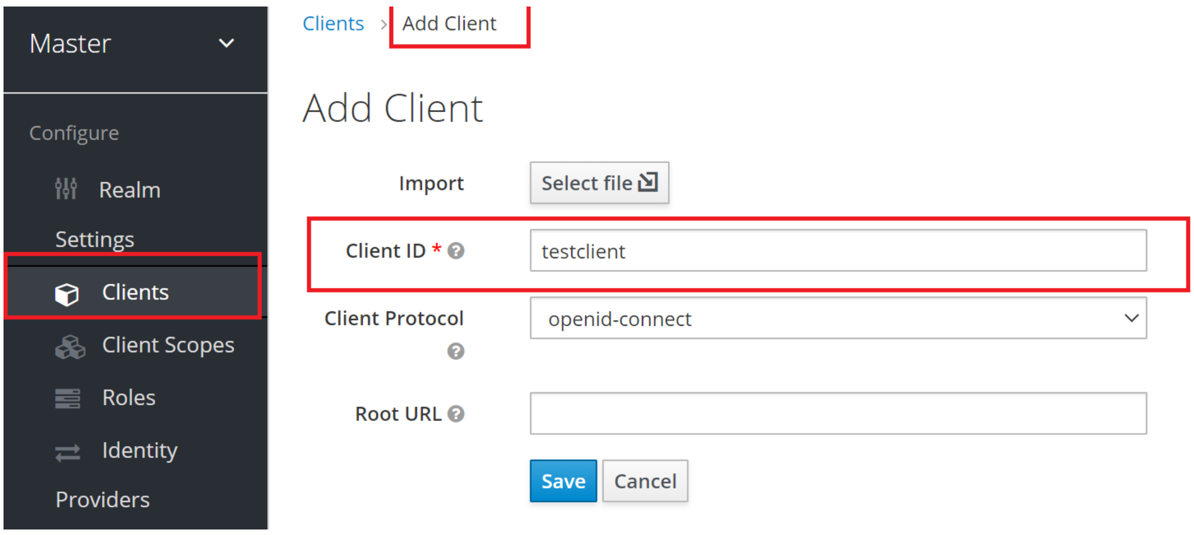
Open the Add Client screen



Client ID	Enabled	Base URL	Actions		
account	True	http://localhost:8080/auth/realms/master/account/	Edit	Export	Delete
account-console	True	http://localhost:8080/auth/realms/master/account/	Edit	Export	Delete
admin-cli	True	Not defined	Edit	Export	Delete
broker	True	Not defined	Edit	Export	Delete
kenny-client	True	Not defined	Edit	Export	Delete
master-realm	True	Not defined	Edit	Export	Delete
security-admin-console	True	http://localhost:8080/auth/admin/master/console/	Edit	Export	Delete
vanilla	True	http://localhost:3000http://localhost:3000	Edit	Export	Delete

Enter the Client

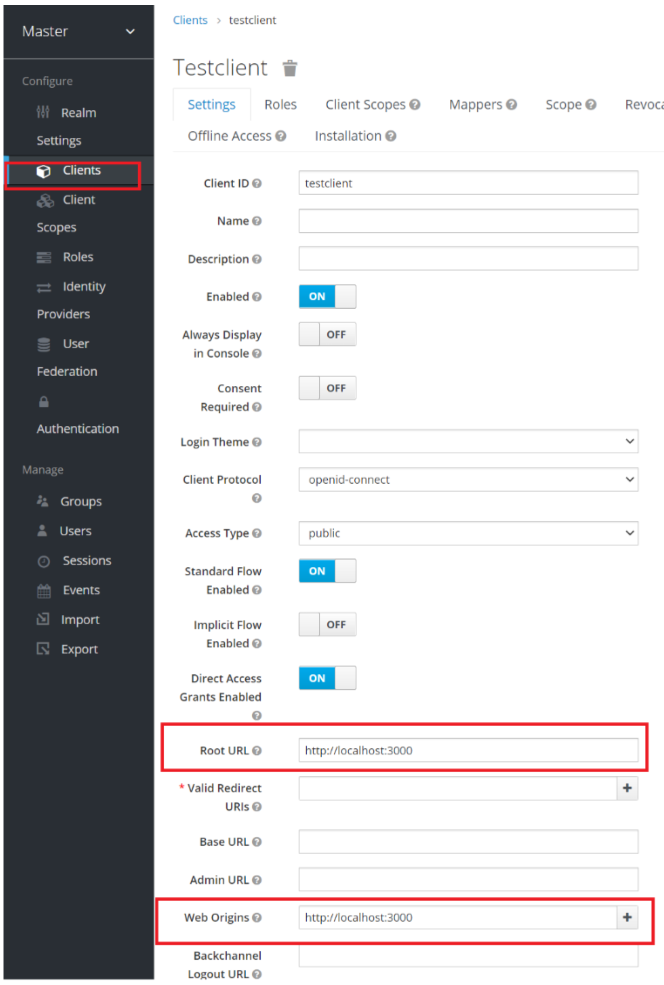
Set the name as **testclient and** click save.



Import	Select file
Client ID *	testclient
Client Protocol	openid-connect
Root URL	

Configure the Client

We need to set the base URL to the URL of the React client we will add later. As for this example we will be using create-react-app we will set this to http://localhost:3000. We add the same URL to the Web Origins so we don't end with CORS errors in the React client. Once entered click save.



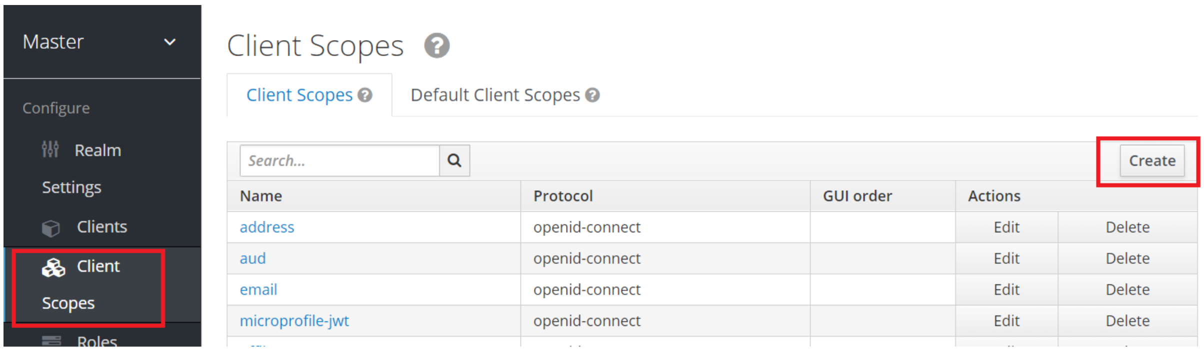
Client ID	testclient
Name	
Description	
Enabled	ON
Always Display in Console	OFF
Consent Required	OFF
Login Theme	
Client Protocol	openid-connect
Access Type	public
Standard Flow Enabled	ON
Implicit Flow Enabled	OFF
Direct Access Grants Enabled	ON
Root URL	http://localhost:3000
* Valid Redirect URIs	
Base URL	
Admin URL	
Web Origins	http://localhost:3000
Backchannel Logout URL	

Create a client scope and mapper.

This is the price that adds the client audience in the **aud** field of a generated token. This is super important.

Open Create Client Scope Screen

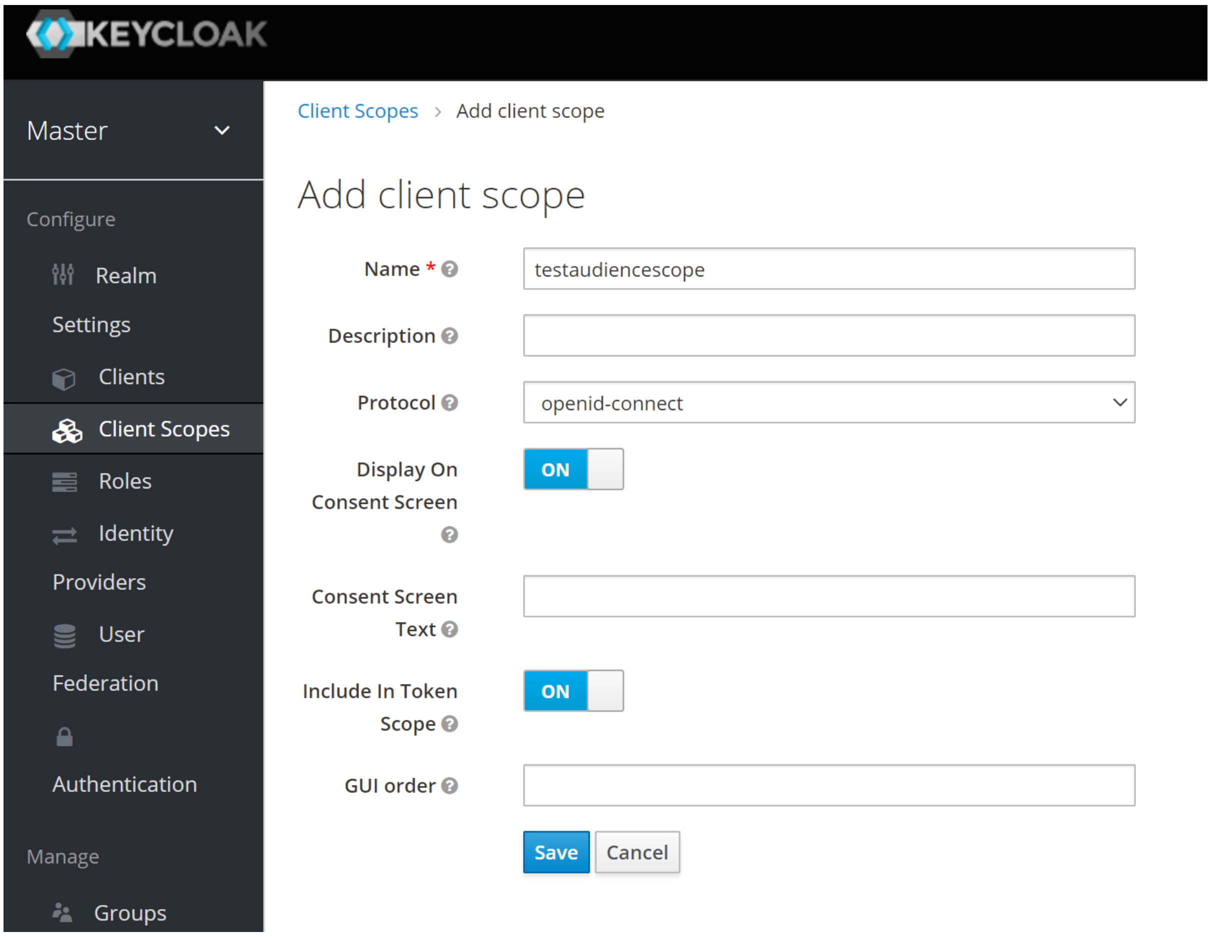
Click Create the from the Client Scopes screen.



Name	Protocol	GUI order	Actions	
address	openid-connect		Edit	Delete
aud	openid-connect		Edit	Delete
email	openid-connect		Edit	Delete
microprofile-jwt	openid-connect		Edit	Delete

Add a new Client Scope

Add a scope with name **testaudiencescope** and click save.

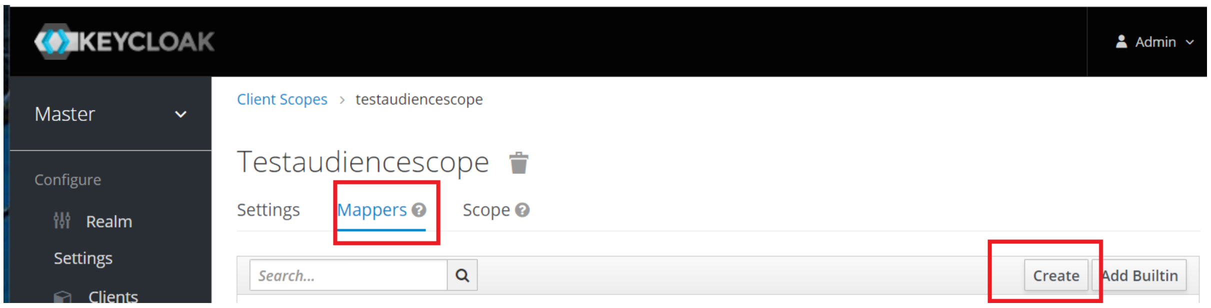


Add client scope

Name *	testaudiencescope
Description	
Protocol	openid-connect
Display On Consent Screen	ON
Consent Screen Text	
Include In Token Scope	ON
GUI order	

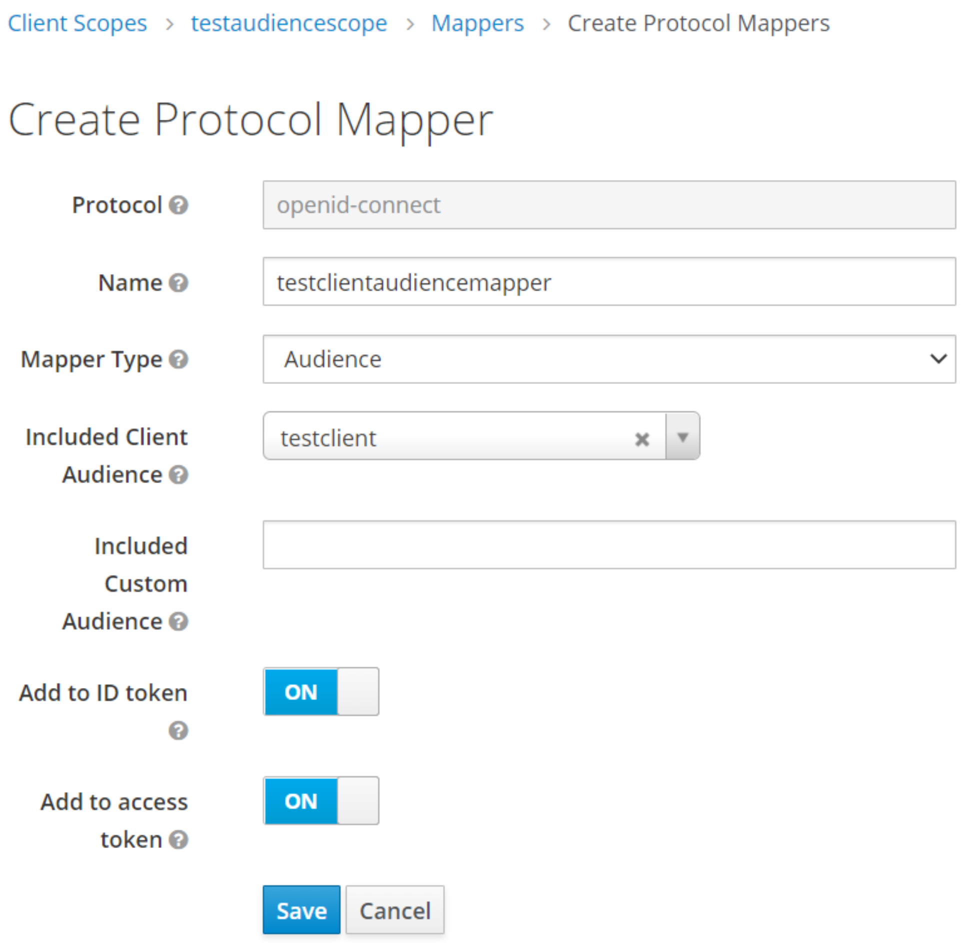
Open the Create Mapper Screen

From our new client scope move to the Mappers tab and click Create.



Configure the Mapper.

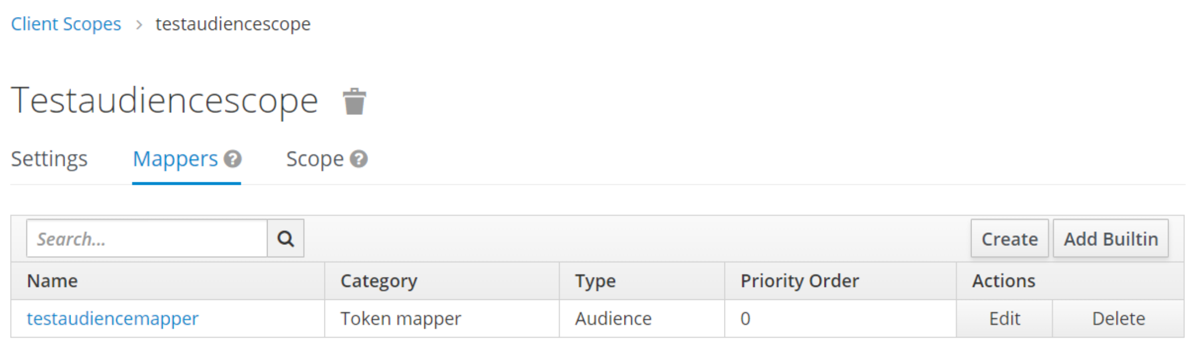
Configure the mapper as follows.


Client Scopes > testaudiencescope > Mappers > Create Protocol Mappers

Create Protocol Mapper

Protocol	openid-connect
Name	testclientaudiencemapper
Mapper Type	Audience
Included Client Audience	testclient
Included Custom Audience	
Add to ID token	ON
Add to access token	ON

The mappers tab should look like this now.

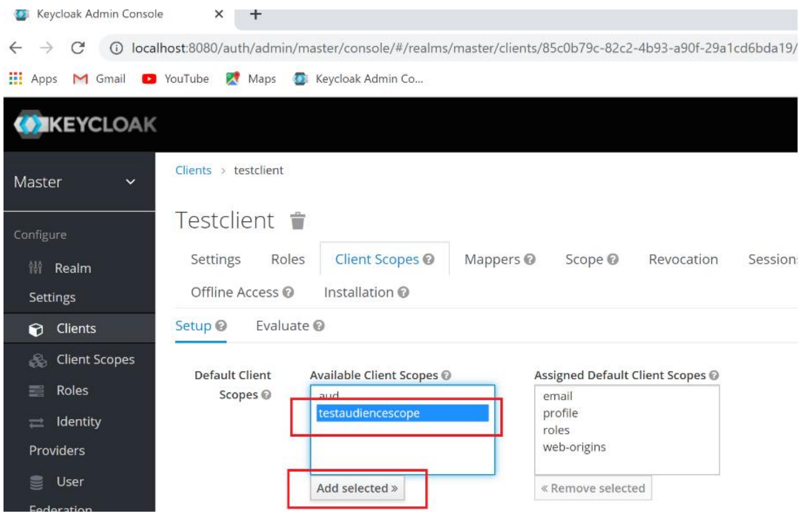

Client Scopes > testaudiencescope

Testaudiencescope

Name	Category	Type	Priority Order	Actions	
testaudiencemapper	Token mapper	Audience	0	Edit	Delete

Add Scope to client

Select the testaudiencescope and add to the Default Client Scopes



Create the Frontend react App.

Create the App

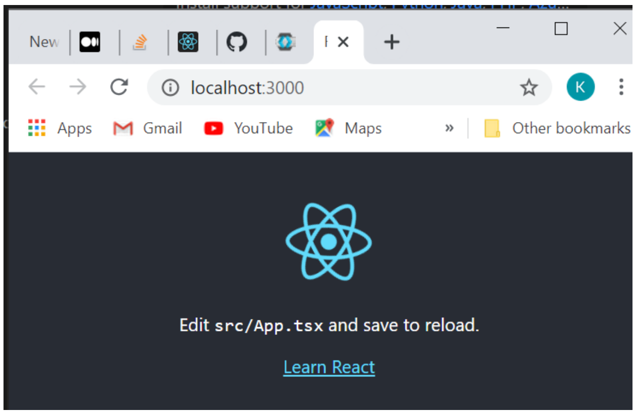
Assuming you have node and npm installed open a command prompt and enter the following command.

```
npx create-react-app ui --template typescript
```

Open visual studio in the new ui folder. Open a terminal and enter

```
Npm start
```

Make sure you can see the react app screen



Edit src/App.tsx and save to reload.

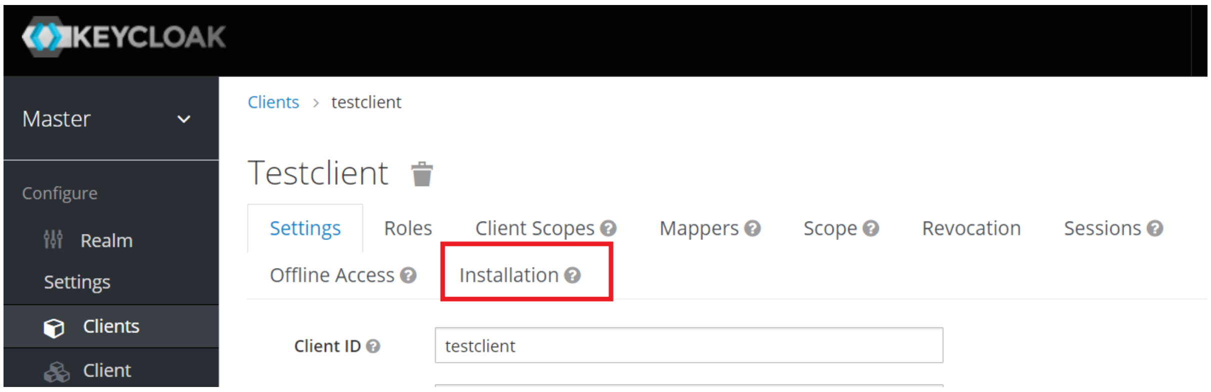
Add KeyCloak npm Package

From the command line run the command

```
npm install keycloak-js
```

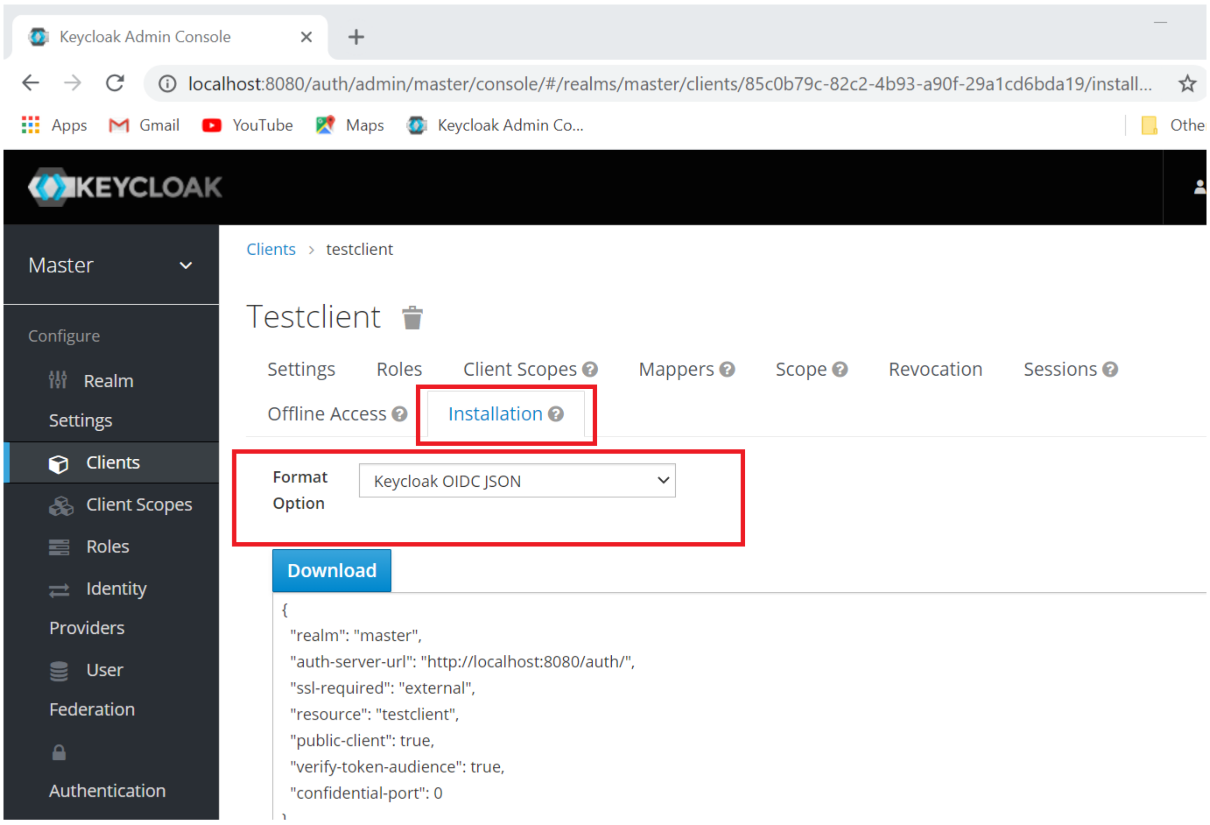
Add KeyCloak settings

From KeyCloak admin UI go to our client that we created in the previous sections and click installation.



Client ID testclient

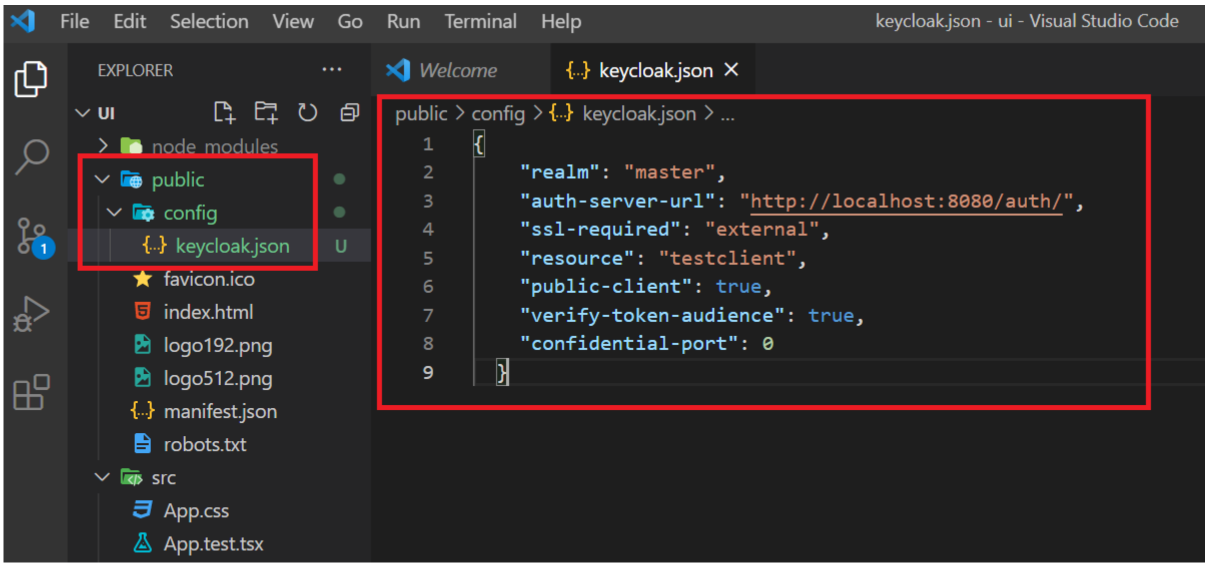
From installation screen select KeyCloak OIDC JSON from the Format Option drop down



Format Option: Keycloak OIDC JSON

```
{
  "realm": "master",
  "auth-server-url": "http://localhost:8080/auth/",
  "ssl-required": "external",
  "resource": "testclient",
  "public-client": true,
  "verify-token-audience": true,
  "confidential-port": 0
}
```

Copy the JSON into a file called keycloak.json in the React app's public/config folder



```
{
    "realm": "master",
    "auth-server-url": "http://localhost:8080/auth/",
    "ssl-required": "external",
    "resource": "testclient",
    "public-client": true,
    "verify-token-audience": true,
    "confidential-port": 0
}
```

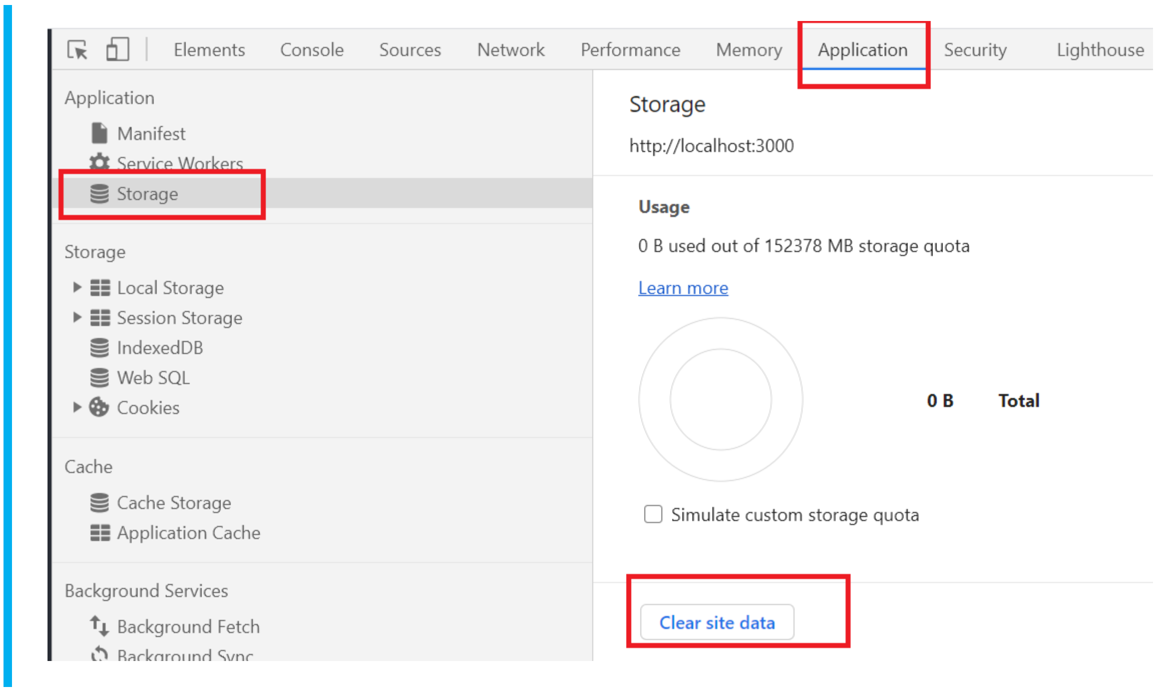
Add Logic to connect to KeyCloak

Replace index.tsx with the following code

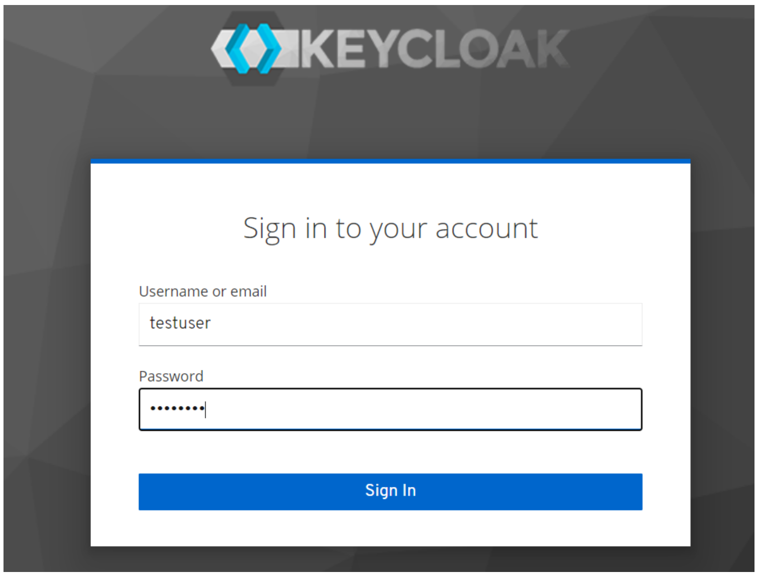
```
 import React, { ReactElement } from 'react';

import ReactDOM from 'react-dom';
import './index.css';
import reportWebVitals from './reportWebVitals';
import Keycloak from 'keycloak-js';

const keycloak = Keycloak(`/config/keycloak.json?ts=${new Date().getTime()}`);

async function DoWork()
{
  await keycloak.init({onLoad:'login-
required', enableLogging:true, checkLoginIframe:false});

  await  keycloak.updateToken(120);

  const token = keycloak.token;
  const tokenParsed = keycloak.tokenParsed;

  ReactDOM.render(
            <React.StrictMode>
              <App json={tokenParsed}></App>
            </React.StrictMode>,
            document.getElementById('root')
          );
}

function App(props:any) : ReactElement
{
  return <pre>{JSON.stringify(props.json,null, 2)}</pre>
}


DoWork();
```

When we reload the app we should be asked to login to KeyCloak.

> **Clear KeyCloak Cached Data**
>
> If we make changes we often need to clear out keycloak settings. We can do this in Chrome by opening developer settings. Going to Application Tab. Selecting Storage and Clear Site Data



Enter the username of testuser and password of testuser.



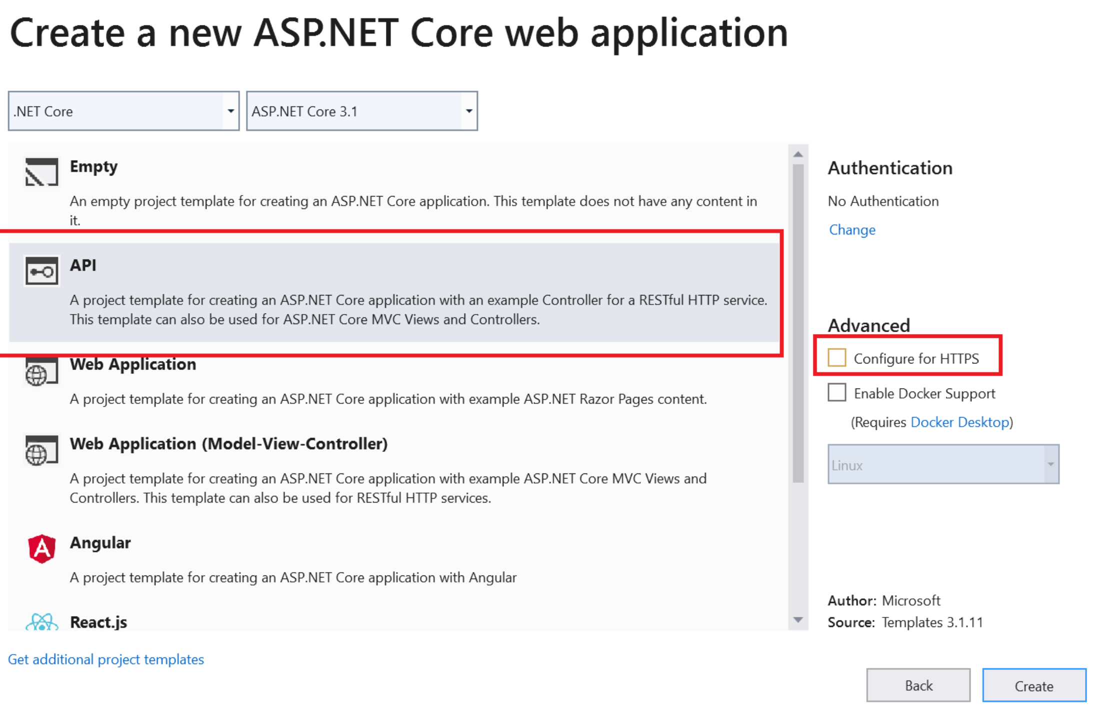
We should see a token similar to the following, Note we have the testclient in the audience. This is key to use the token from .net

```
{
  "exp": 1613042771,
  "iat": 1613042711,
  "auth_time": 1613042696,
  "jti": "a2c0dbd8-5f53-4cc8-9b23-9f7c3e633757",
  "iss": "http://localhost:8080/auth/realms/master",
    "aud": [
      "testclient",
    "master-realm",
    "account"
  ],
  "sub": "bd93aa68-622e-4907-aeb7-59967e7e1490",
  "typ": "Bearer",
  "azp": "testclient",
  "nonce": "c64e4324-f5ad-4ea0-9ba8-7942b5f4c242",
  "session_state": "badf9150-4718-4d4b-9693-565e6b3c0344",
  "acr": "0",
  "allowed-origins": [
    "http://localhost:3000"
  ],
  "realm_access": {
    "roles": [
      "create-realm",
      "offline_access",
      "admin",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "master-realm": {
      "roles": [
        "view-identity-providers",
        "view-realm",
        "manage-identity-providers",
        "impersonation",
        "create-client",
        "manage-users",
        "query-realms",
        "view-authorization",
        "query-clients",
        "query-users",
        "manage-events",
        "manage-realm",
        "view-events",
        "view-users",
        "view-clients",
        "manage-authorization",
        "manage-clients",
        "query-groups"
      ]
    },
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "scope": "openid profile email testaudiencescope",
  "email_verified": false,
  "preferred_username": "admin"
}
```

Create the Backend.

Create the Project

Open visual studio and create a new application using **ASP.NET Core Web Application**. Give the solution and project a name and then select the API template. Untick the configure HTTP checkbox



Create a new ASP.NET Core web application

.NET Core | ASP.NET Core 3.1

Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

Web Application (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Angular
A project template for creating an ASP.NET Core application with Angular

React.js

Get additional project templates

Authentication
No Authentication
Change

Advanced
☐ Configure for HTTPS
☐ Enable Docker Support
(Requires Docker Desktop)
Linux

Author: Microsoft
Source: Templates 3.1.11

Create Packages for JWT and OpenId

Make sure the project is targetint .NET 5.0 and add the following dependencies to the csproj file.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

	<PropertyGroup>
		<TargetFramework>net5.0</TargetFramework>
	</PropertyGroup>

	<ItemGroup>
		<PackageReference
Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="5.0.3" />
		<PackageReference
Include="Microsoft.AspNetCore.Authentication.OpenIdConnect" Version="5.0.3" />
	</ItemGroup>

</Project>
```

USE AUTHENTICATION ON ENDPOINT ACTION

Add the authorization attribute to the end point `WeatherForecastController.cs`

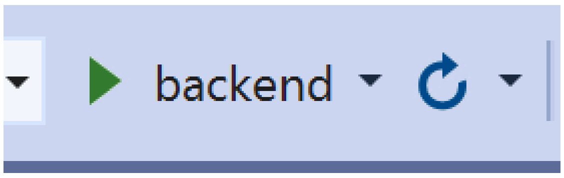
```
[HttpGet]
[Authorize]
public IEnumerable<WeatherForecast> Get()
{
    var rng = new Random();
    return Enumerable.Range(1, 5).Select(index => new WeatherForecast
    {
        Date = DateTime.Now.AddDays(index),
        TemperatureC = rng.Next(-20, 55),
        Summary = Summaries[rng.Next(Summaries.Length)]
    })
    .ToArray();
}
```

USE KEYCLOAK TO STARTUP.CS

```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    public void ConfigureServices(IServiceCollection services)
    {
        services.AddCors();

        services.AddControllers();

        var auth = services.AddAuthentication();

        auth.AddJwtBearer("myscheme", options =>
        {
            options.Authority = "http://localhost:8080/auth/realms/master";
            options.Audience = "testclient";
            options.RequireHttpsMetadata = false;
        });

        services.AddAuthorization(options =>
        {
            options.DefaultPolicy = new AuthorizationPolicyBuilder()
                .AddAuthenticationSchemes(new { "myscheme" })
                .RequireAuthenticatedUser()
                .Build();
        });
    }

    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.UseRouting();

        app.UseCors(builder =>
            builder.AllowAnyMethod()
                .AllowAnyHeader()
                .AllowCredentials()
                .SetIsOriginAllowed(s => true));

        app.UseAuthentication();
        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
     {
         endpoints.MapControllers();
     });
    }
}
```

Run

Make sure we run the project profile and not the IIS one.



Add Code To Front End to hit authenticated endpoint

```
import React, { ReactElement } from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import Keycloak from 'keycloak-js';

const keycloak = Keycloak(`/config/keycloak.json?ts=${new Date().getTime()}`);

async function DoWork() {
  await keycloak.init(
    {
      onLoad: 'login-required',
      enableLogging: true,
      checkLoginIframe: false
    });

  await keycloak.updateToken(120);
  const token = keycloak.token;
  const tokenParsed = keycloak.tokenParsed;

  const result = await fetch('http://localhost:5000/weatherforecast',
    {
      mode: "cors",
      headers: [
        ['authorization', `Bearer ${keycloak.token}`]
      ]
    }
  );

  ReactDOM.render(
    <React.StrictMode>
      <App json={await result.json()}></App>
    </React.StrictMode>,
    document.getElementById('root')
  );
}

function App(props: any): ReactElement {
  return <pre>{JSON.stringify(props.json, null, 2)}</pre>
}


DoWork();
```

Make Sure You Can See the Result

```
[
  {
    "date": "2021-02-12T11:40:36.7960094+00:00",
    "temperatureC": 30,
    "temperatureF": 85,
    "summary": "Freezing"
  },
  {
    "date": "2021-02-13T11:40:36.7960137+00:00",
    "temperatureC": 27,
    "temperatureF": 80,
    "summary": "Warm"
  },
  {
    "date": "2021-02-14T11:40:36.796014+00:00",
    "temperatureC": 28,
    "temperatureF": 82,
    "summary": "Chilly"
  },
  {
    "date": "2021-02-15T11:40:36.7960142+00:00",
    "temperatureC": -5,
    "temperatureF": 24,
    "summary": "Hot"
  },
  {
    "date": "2021-02-16T11:40:36.7960145+00:00",
    "temperatureC": 10,
    "temperatureF": 49,
    "summary": "Chilly"
  }
]
```

Using KeyCloak

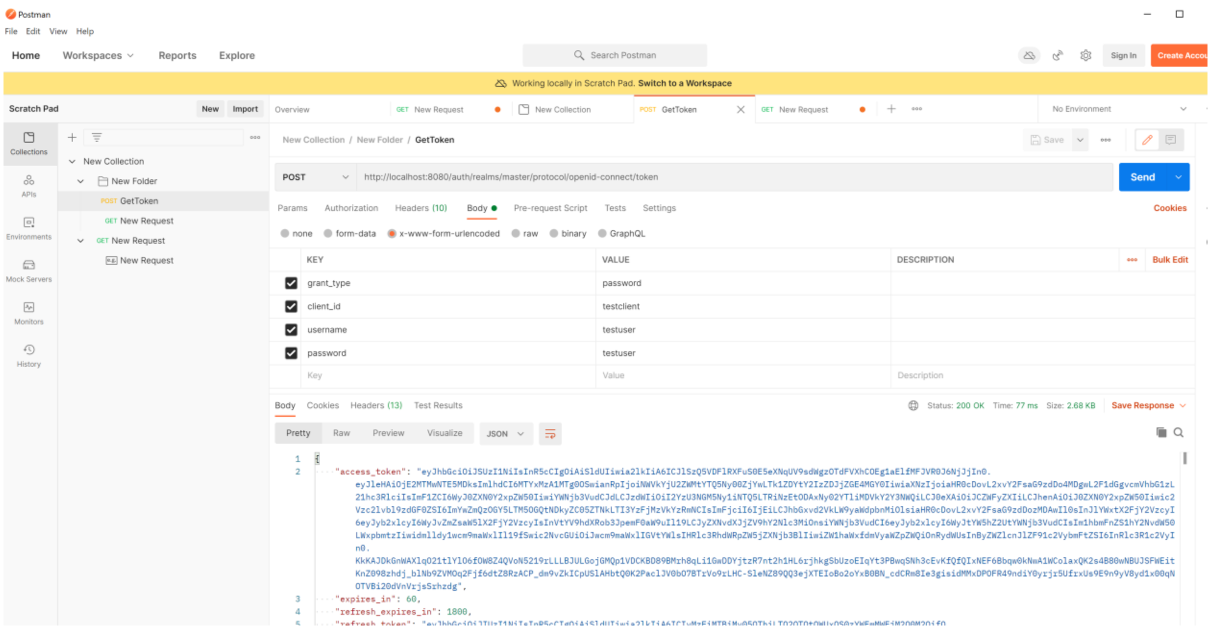
Get Server Endpoint details

This gives details on end points on the server

http://localhost:8080/auth/realms/master/.well-known/openid-configuration

Get Token.

The following shows how to use PostMan to get a token. Note the data is form encoded the HTTP verb is POST.


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad New Import
Collections APIs Environments Mock Servers Monitors History
New Collection
New Folder
POST GetToken
GET New Request
GET New Request
New Request
Overview
No Environment
New Collection / New Folder / GetToken
Save
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies
Bulk Edit
Body Cookies Headers (13) Test Results
Save Response
Pretty Raw Preview Visualize JSON

POST http://localhost:8080/auth/realms/master/protocol/openid-connect/token

Body: none / form-data / [✓ x-www-form-urlencoded] / raw / binary / GraphQL

	KEY	VALUE	DESCRIPTION
✓	grant_type	password	
✓	client_id	testclient	
✓	username	testuser	
✓	password	testuser	
	Key	Value	Description

Status: 200 OK Time: 77 ms Size: 2.68 KB

```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJlSzQ5VDFlRXFuS0E5eXNqUV9sdWgzOTdFVXhCOEg1aElfMFJVR0J6NjJjIn0.
        eyJleHAiOjE2MTMwNTE5MDksImlhdCI6MTYxMzA1MTg0OSwianRpIjoiNWVkYjU2ZWMtYTQ5Ny00ZjYwLTk1ZDYtY2IzZDJjZGE4MGY0IiwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdDo4MDgwL2F1dGgvcmVhbG1zL
        21hc3RlciIsImF1ZCI6WyJ0ZXN0Y2xpZW50IiwiYWNjb3VudCJdLCJzdWIiOiI2YzU3NGM5Ny1iNTQ5LTRiNzEtODAxNy02YTliMDVkY2Y3NWQiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJ0ZXN0Y2xpZW50Iiwic2
        Vzc2lvbl9zdGF0ZSI6ImYwZmQzOGY5LTM5OGQtNDkyZC05ZTNkLTI3YzFjMzVkYzRmNCIsImFjciI6IjEiLCJhbGxvd2VkLW9yaWdpbnMiOlsiaHR0cDovL2xvY2FsaG9zdDozMDAwIl0sInJlYWxtX2FjY2VzcyI
        6eyJyb2xlcyI6WyJvZmZsaW5lX2FjY2VzcyIsInVtYV9hdXRob3JpemF0aW9uIl19LCJyZXNvdXJjZV9hY2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsIm1hbmFnZS1hY2NvdW50
        LWxpbmtzIiwidmlldy1wcm9maWxlIl19fSwic2NvcGUiOiJwcm9maWxlIGVtYWlsIHRlc3RhdWRpZW5jZXNjb3BlIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsInByZWZlcnJlZF91c2VybmFtZSI6InRlc3R1c2VyI
        n0.
        KkKAJDkGnWAXlq021tlYl06fOW8Z4QVoN5219rLLLBJULGojGMQp1VDCKBD89BMrh8qLi1GwDDYjtzR7nt2h1HL6rjhkgSbUzoEIqYt3PBwqSNh3cEvKfQfQIxNEF6Bbqw0kNwA1WColaxQK2s4B80wNBUJSFWEit
        KnZ098zhdj_blNb9ZVMOq2Fjf6dtZ8RzACP_dm9vZkICpUSlAHbtQ0K2PaclJV0bO7BTrVo9rLHC-SleNZ89QQ3ejXTEIoBo2oYxB0BN_cdCRm8Ie3gisidMMxDPOFR49ndiY0yrjr5UfrxUs9E9n9yV8yd1x00qN
        OTVBi20dVnVrjsSrhzdg",
    "expires_in": 60,
    "refresh_expires_in": 1800,
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICIyMzFiMTBiMy05OThjLTQ2OTQtOWUxOS0zYWFmMWFiM2Q0M2QifQ
```