

### 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>

## Risk and Pricing Solutions

## 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](https://www.keycloak.org/docs/latest/getting_started/index.html)

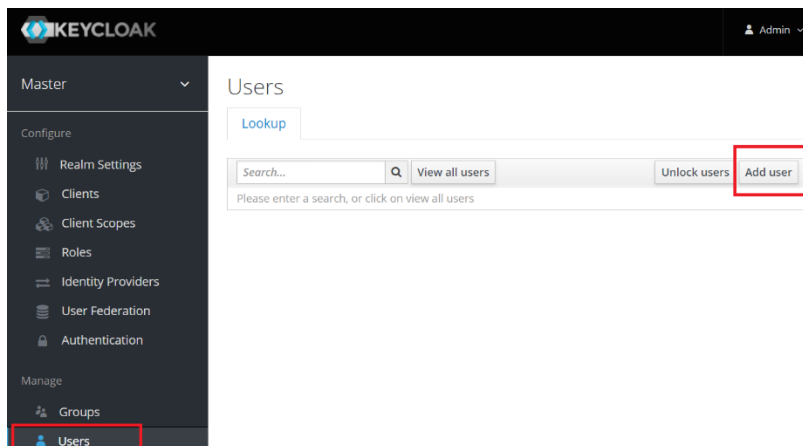
### Configure KeyCloak.

For this tutorial we will just use the master realm.

#### ADD A USER.

##### Open the Add user screen

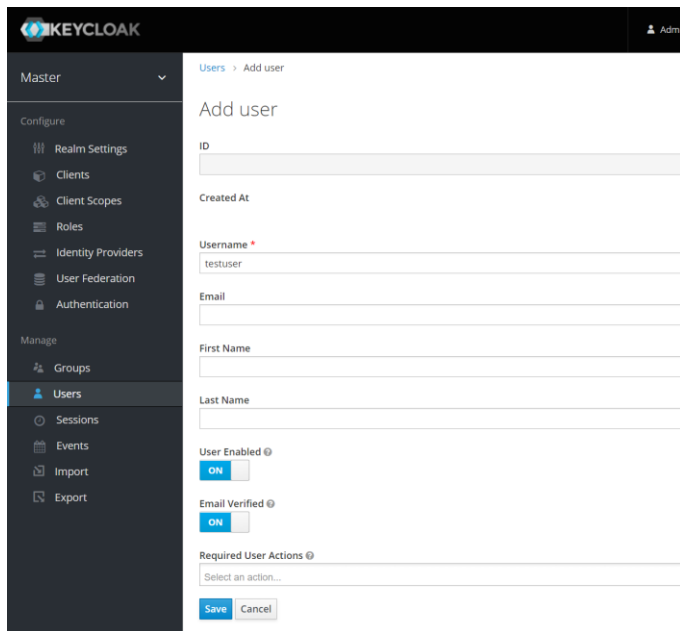
We go to the Users tab and click add user.



##### 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.

# Risk and Pricing Solutions

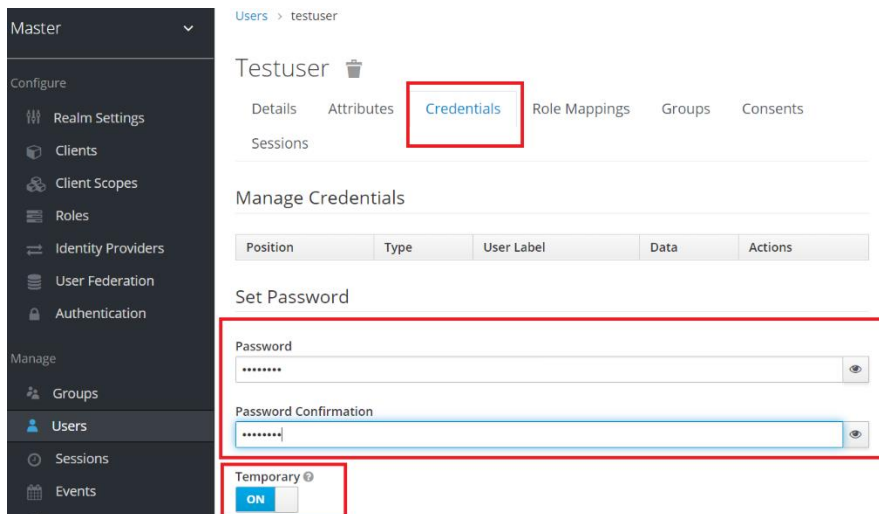


The image shows the 'Add user' form in the Keycloak administration console. The left sidebar contains navigation links for Master, Configure (Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User Federation, Authentication), and Manage (Groups, Users, Sessions, Events, Import, Export). The main content area is titled 'Add user' and includes fields for ID, Created At, Username (pre-filled with 'testuser'), Email, First Name, and Last Name. There are also toggle switches for 'User Enabled' and 'Email Verified', both set to 'ON'. A 'Required User Actions' dropdown is set to 'Select an action...'. At the bottom are 'Save' and 'Cancel' buttons.

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 login. Enter the password as **testuser**.



The image shows the 'Testuser' profile page in Keycloak, specifically the 'Credentials' tab. The left sidebar is the same as in the previous image. The main content area shows the user's name 'Testuser' and tabs for Details, Attributes, Credentials (highlighted with a red box), Role Mappings, Groups, and Consents. Below the tabs is a 'Manage Credentials' section with a table header: Position, Type, User Label, Data, and Actions. Underneath is a 'Set Password' section with two input fields: 'Password' and 'Password Confirmation', both containing masked characters and highlighted with a red box. At the bottom, there is a 'Temporary' toggle switch, which is currently set to 'ON' and also highlighted with a red box.

# Risk and Pricing Solutions

## ADD A CLIENT.

Open the Add Client screen

Master ▾

Configure

- Realm
- Settings
- Clients**
- Client
- Scopes
- Roles
- Identity
- Providers
- User

Clients

Lookup ⓘ

Search... Q

Create

Client ID	Enabled	Base URL	Actions		
account	True	<a href="http://localhost:8080/auth/realms/master/account/">http://localhost:8080/auth/realms/master/account/</a>	Edit	Export	Delete
account-console	True	<a href="http://localhost:8080/auth/realms/master/account/">http://localhost:8080/auth/realms/master/account/</a>	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	<a href="http://localhost:8080/auth/admin/master/console/">http://localhost:8080/auth/admin/master/console/</a>	Edit	Export	Delete
vanilla	True	<a href="http://localhost:3000http://localhost:3000">http://localhost:3000http://localhost:3000</a>	Edit	Export	Delete

## Enter the Client

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

Master ▾

Configure

- Realm
- Settings
- Clients**
- Client
- Scopes
- Roles
- Identity
- Providers

Clients > Add Client

Add Client

Import

Client ID \* ⓘ testclient

Client Protocol ⓘ openid-connect ▾

Root URL ⓘ

# Risk and Pricing Solutions

## 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.

The screenshot shows the Keycloak administration interface for configuring a client named 'Testclient'. The left sidebar contains a 'Clients' menu item, which is highlighted with a red box. The main content area shows the configuration for 'Testclient' with various tabs like 'Settings', 'Roles', 'Client Scopes', etc. The 'Settings' tab is active, showing fields for Client ID, Name, Description, Enabled, Always Display in Console, Consent Required, Login Theme, Client Protocol, Access Type, Standard Flow Enabled, Implicit Flow Enabled, Direct Access Grants Enabled, Root URL, Valid Redirect URIs, Base URL, Admin URL, Web Origins, Backchannel, and Logout URL. The 'Root URL' and 'Web Origins' fields are highlighted with red boxes, both containing the value 'http://localhost:3000'.

Field	Value
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	

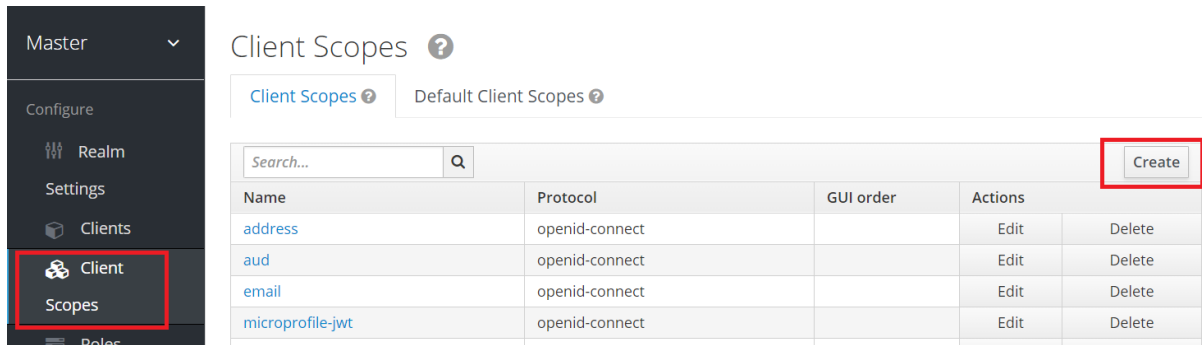
# Risk and Pricing Solutions

## 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.



Client Scopes ?

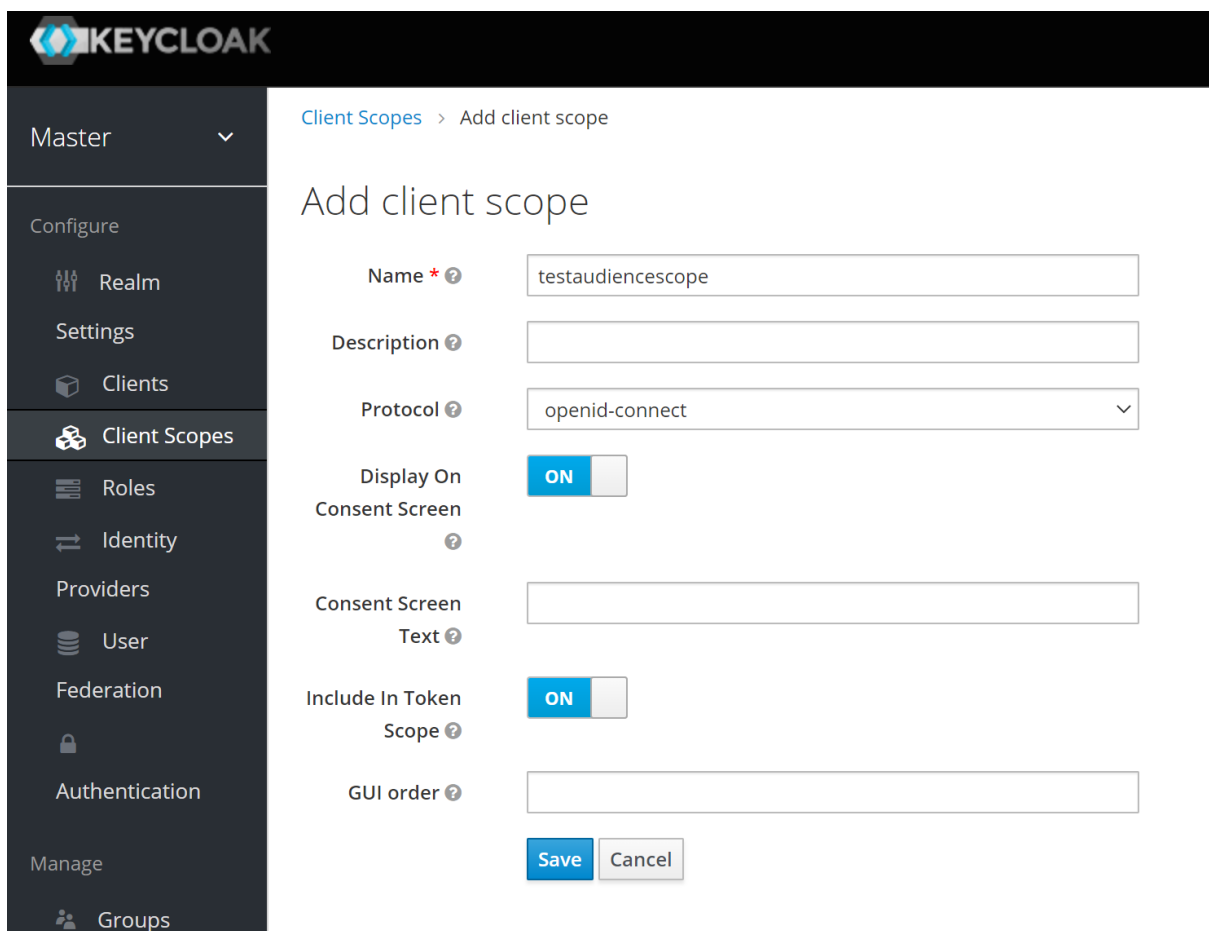
Client Scopes ? Default Client Scopes ?

Search... Q

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.



KEYCLOAK

Master

Configure

- Realm
- Settings
- Clients
- Client Scopes**
- Roles
- Identity
- Providers
- User
- Federation
- Authentication

Client Scopes > Add client scope

### Add client scope

Name \* ? testaudiencescope

Description ?

Protocol ? openid-connect

Display On Consent Screen ? ☒ ON

Consent Screen Text ?

Include In Token Scope ? ☒ ON

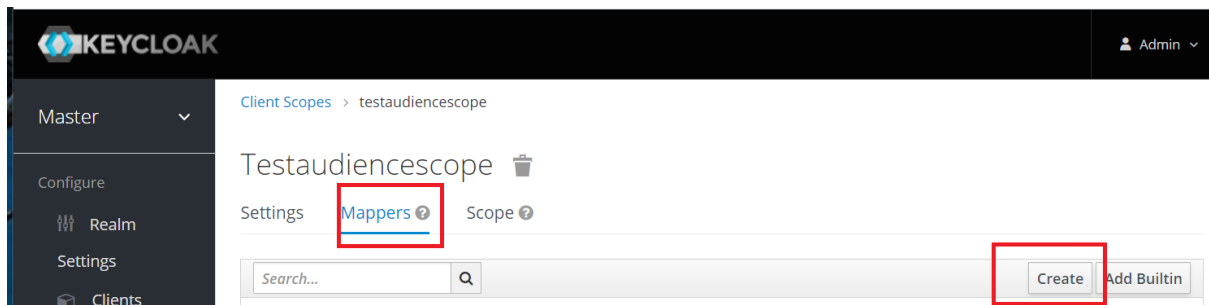
GUI order ?

Save Cancel

# Risk and Pricing Solutions

## 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 ?	<input type="text" value="openid-connect"/>
Name ?	<input type="text" value="testcliaudiencemapper"/>
Mapper Type ?	<input type="text" value="Audience"/>
Included Client Audience ?	<input type="text" value="testclient"/>
Included Custom Audience ?	<input type="text"/>
Add to ID token ?	<input checked="" type="checkbox"/>
Add to access token ?	<input checked="" type="checkbox"/>
	<input type="button" value="Save"/> <input type="button" value="Cancel"/>


# Risk and Pricing Solutions

The mappers tab should look like this now.

[Client Scopes](#) > testaudiencescope

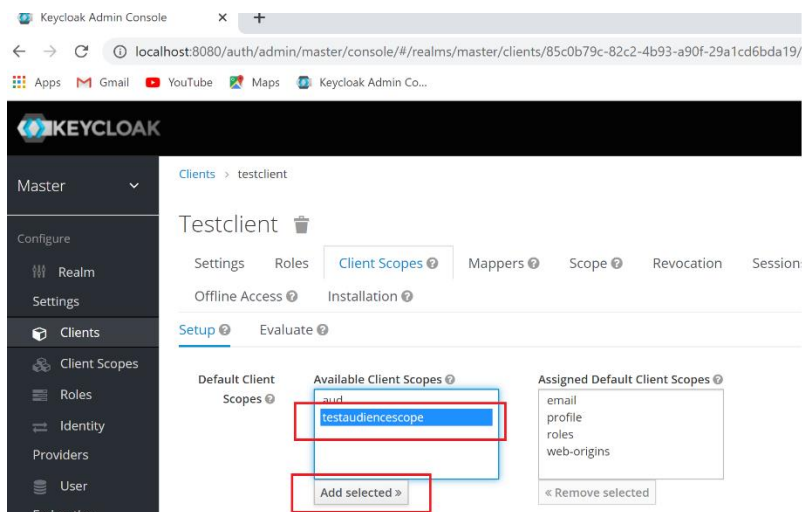
Testaudiencescope 

Settings [Mappers](#)  [Scope](#) 

<input type="text" value="Search..."/>					<a href="#">Create</a>	<a href="#">Add Builtin</a>
Name	Category	Type	Priority Order	Actions		
<a href="#">testaudiencemapper</a>	Token mapper	Audience	0	<a href="#">Edit</a>	<a href="#">Delete</a>	

## Add Scope to client

Select the testaudiencescope and add to the Default Client Scopes





## Risk and Pricing Solutions

### Create the Frontend react App.

#### CREATE THE APP

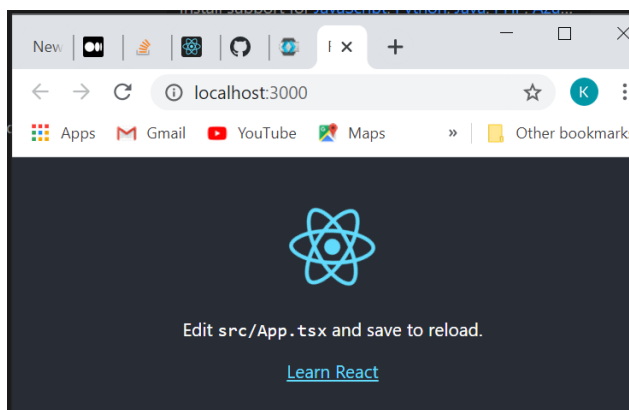
Assuming you have node and npm installed open a command prompt and enter the following command.

```
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



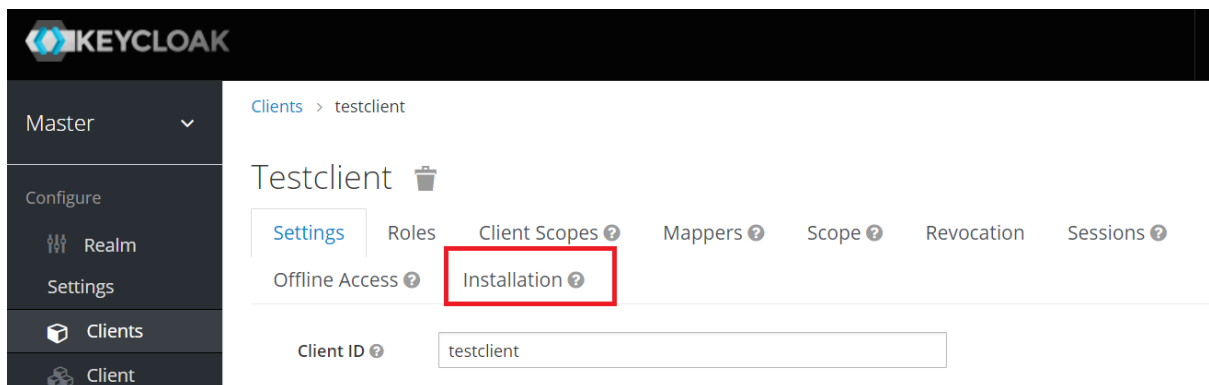
#### ADD KEYCLOAK NPM PACKAGE

From the command line run the command

```
npm install keycloak-js
```

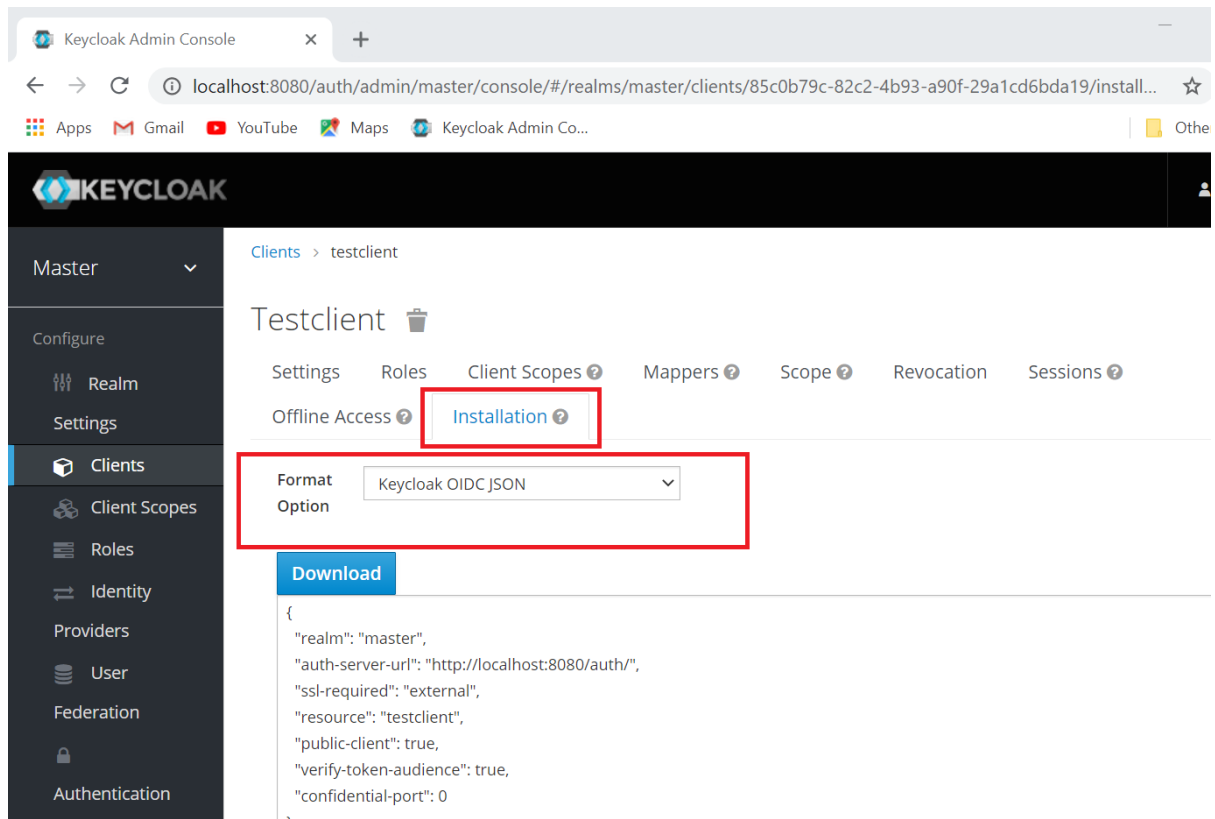
#### ADD KEYCLOAK SETTINGS

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

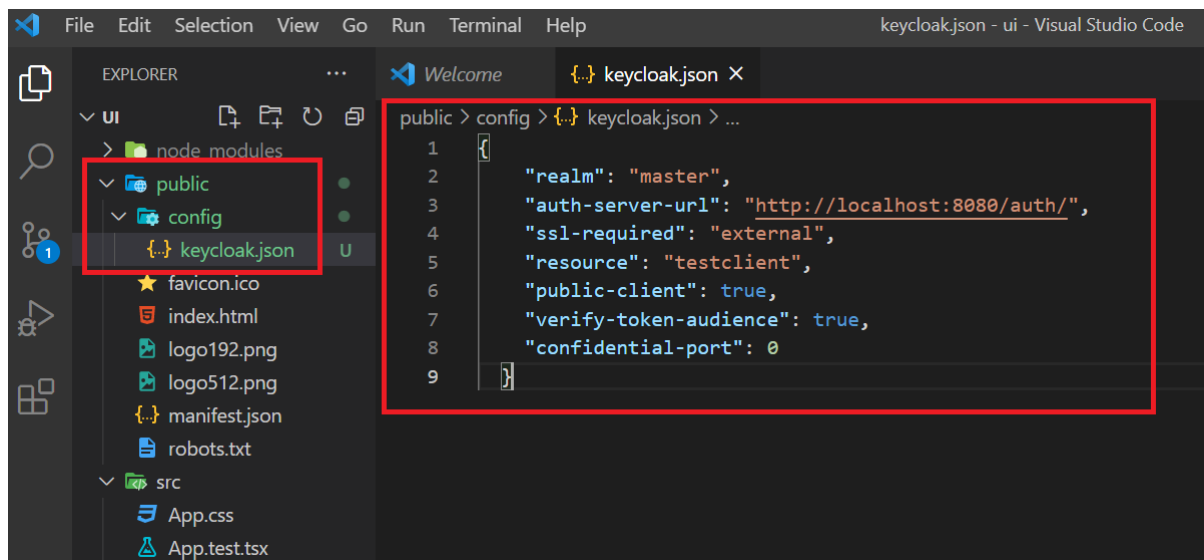


From installation screen select KeyCloak OIDC JSON from the Format Option drop down

## Risk and Pricing Solutions



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



# Risk and Pricing Solutions

## 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>
}

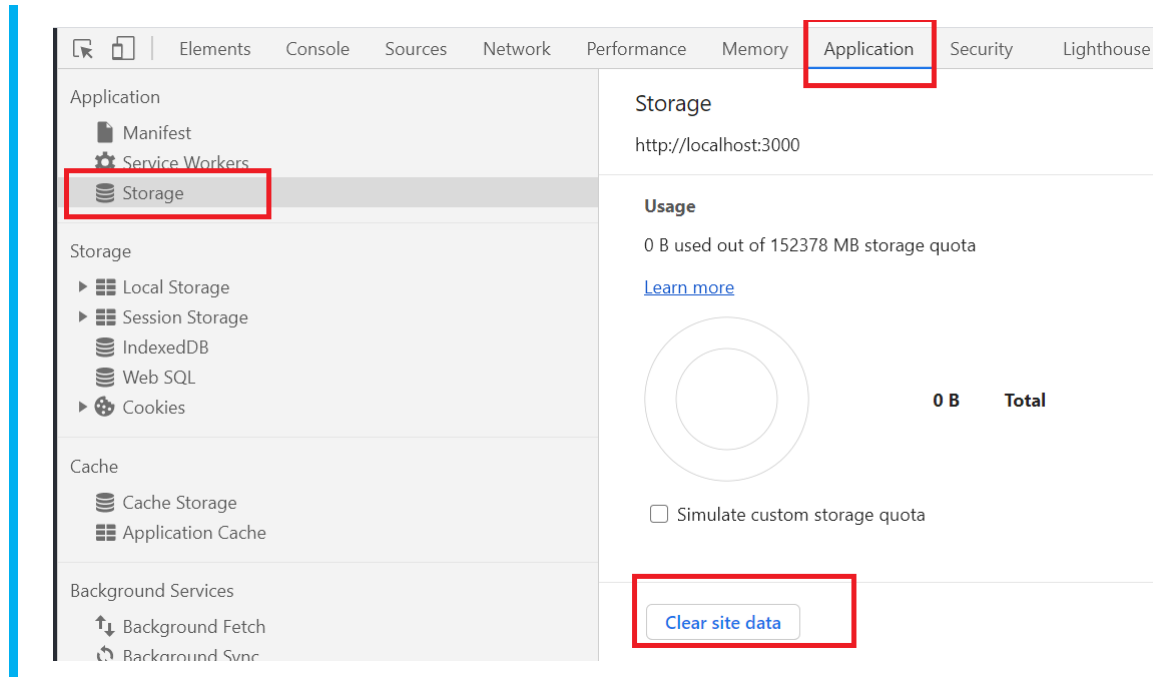
DoWork();
```

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

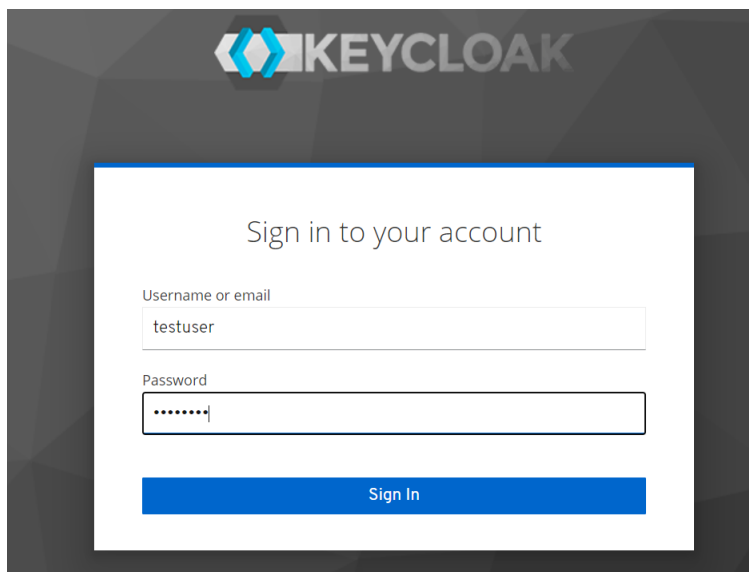
### CLEAR KEYCLOAK CACHED DATA

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

## Risk and Pricing Solutions



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

## Risk and Pricing Solutions

```
{
  "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"
}
```

## Risk and Pricing Solutions

### 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

The screenshot shows the 'Create a new ASP.NET Core web application' dialog in Visual Studio. At the top, there are two dropdown menus: '.NET Core' and 'ASP.NET Core 3.1'. Below these, a list of templates is shown: 'Empty', 'API', 'Web Application', 'Web Application (Model-View-Controller)', 'Angular', and 'React.js'. The 'API' template is selected and highlighted with a red box. To the right of the templates list, there are two sections: 'Authentication' and 'Advanced'. In the 'Authentication' section, 'No Authentication' is selected. In the 'Advanced' section, the 'Configure for HTTPS' checkbox is unchecked and highlighted with a red box. Below this, there is a checkbox for 'Enable Docker Support' which is also unchecked. At the bottom right, there are 'Back' and 'Create' buttons.

#### CREATE PACKAGES FOR JWT AND OPENID

Make sure the project is targeting .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>
```

# Risk and Pricing Solutions

## 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();
}
```

# Risk and Pricing Solutions

## 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();

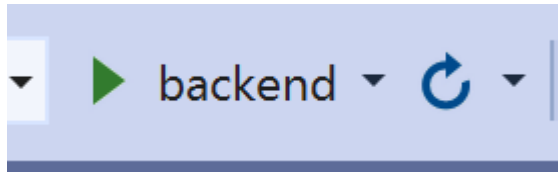
        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
```



## Risk and Pricing Solutions

### RUN

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



## Risk and Pricing Solutions

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();
```

## Risk and Pricing Solutions

### 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"
  }
]
```

