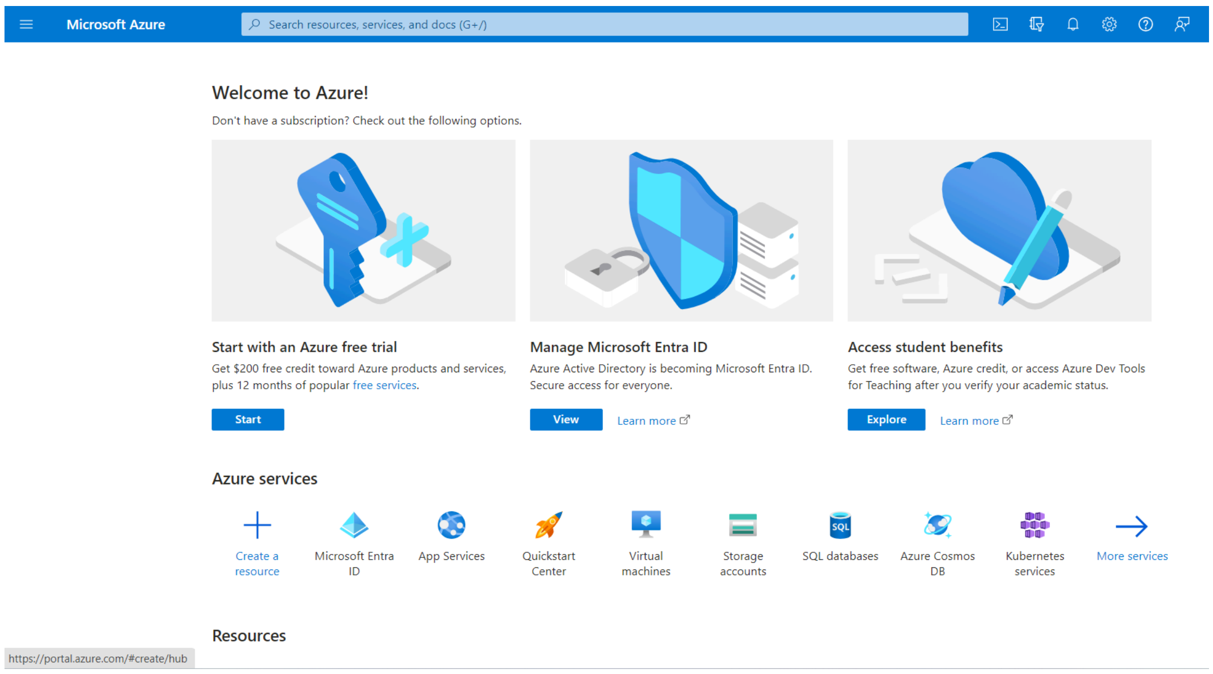
POC for SSO Integration with Azure using Java and React JS

Microsoft Azure Single Sign-On (SSO) is a cloud-based identity and access management service that enables users to securely sign in once and gain access to various applications and services without the need to re-authenticate. It simplifies user access to a multitude of applications, enhancing productivity and providing a seamless and secure authentication experience.

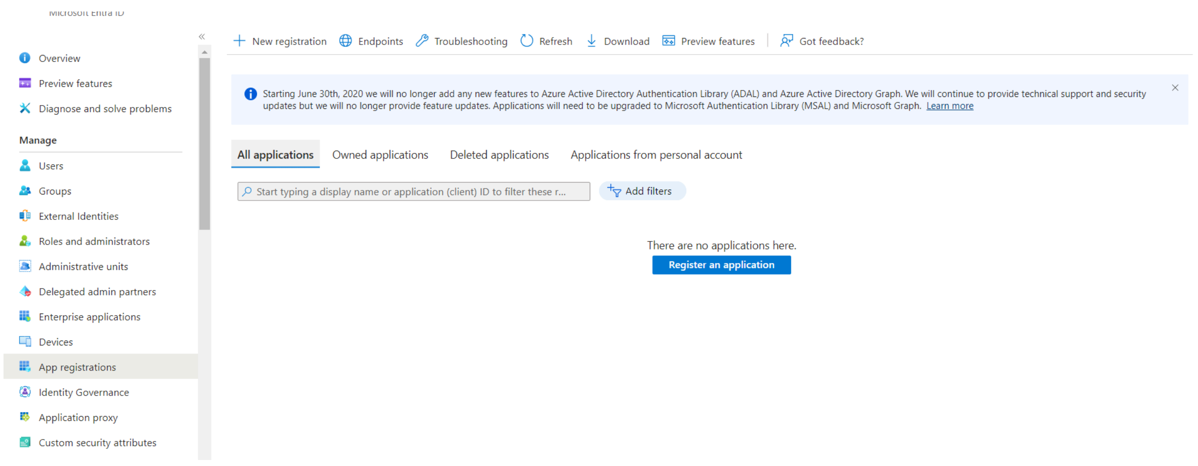
Here are few steps which required to setup the azure account

Setup Azure account with default directory

Sign in to azure portal by clicking this https://portal.azure.com/. After signing in, the home page will be displayed as shown in the image, and after that click on **Manage Microsoft Entra ID**.


Microsoft Azure
Search resources, services, and docs (G+/)
Welcome to Azure!
Don't have a subscription? Check out the following options.
Start with an Azure free trial
Get $200 free credit toward Azure products and services, plus 12 months of popular free services.
Start
Manage Microsoft Entra ID
Azure Active Directory is becoming Microsoft Entra ID. Secure access for everyone.
View
Learn more
Access student benefits
Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status.
Explore
Learn more
Azure services
Create a resource
Microsoft Entra ID
App Services
Quickstart Center
Virtual machines
Storage accounts
SQL databases
Azure Cosmos DB
Kubernetes services
More services
Resources
https://portal.azure.com/#create/hub

After clicking on Microsoft Entra Id, the Default directory will be displayed. Now go to **App Registration -> New Registration**

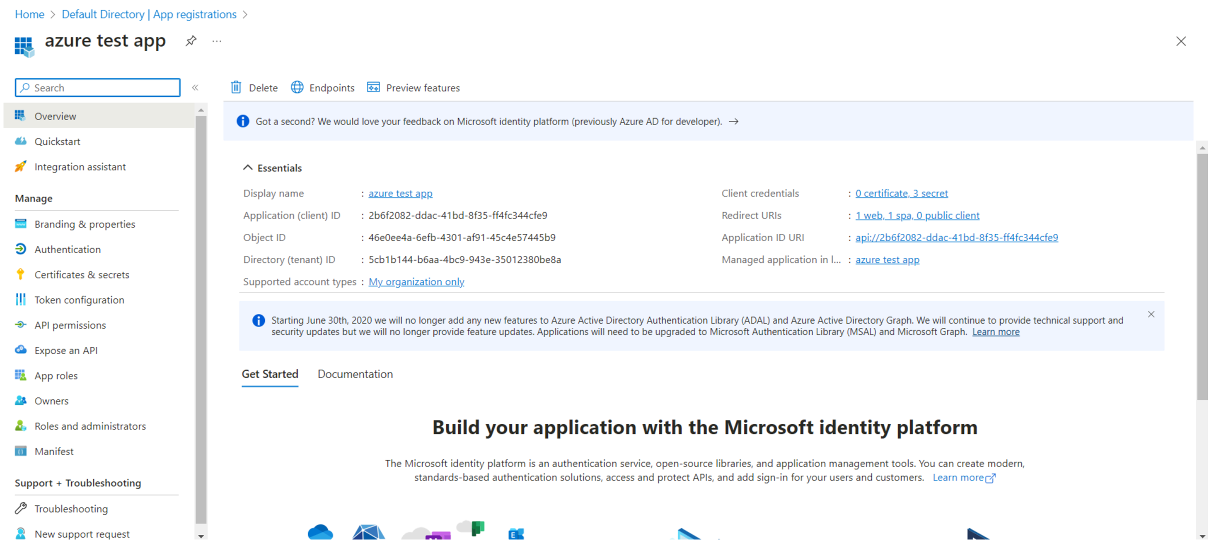


Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

All applications
Owned applications
Deleted applications
Applications from personal account
Start typing a display name or application (client) ID to filter these r...
Add filters

There are no applications here.

After successful application registration, the application overview page will be displayed. And application information will displayed like Client Id, Tenant Id



Got a second? We would love your feedback on Microsoft identity platform (previously Azure AD for developer). →

Essentials

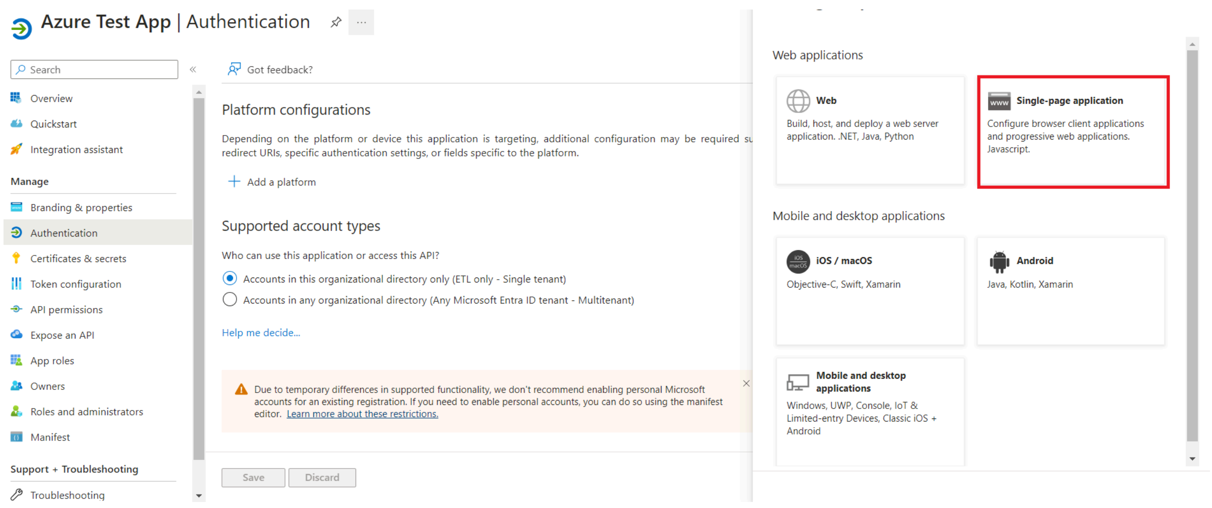
Display name	: azure test app	Client credentials	: 0 certificate, 3 secret
Application (client) ID	: 2b6f2082-ddac-41bd-8f35-ff4fc344cfe9	Redirect URIs	: 1 web, 1 spa, 0 public client
Object ID	: 46e0ee4a-6efb-4301-af91-45c4e57445b9	Application ID URI	: api://2b6f2082-ddac-41bd-8f35-ff4fc344cfe9
Directory (tenant) ID	: 5cb1b144-b6aa-4bc9-943e-35012380be8a	Managed application in l...	: azure test app
Supported account types	: My organization only		

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

Build your application with the Microsoft identity platform

The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern, standards-based authentication solutions, access and protect APIs, and add sign-in for your users and customers. Learn more

To set up a redirect(Callback) url, click on Authentication -> Add a platform. As we will create authentication from react so we will choose a **single page application** and provide a redirect URL. In this demo, we have provided the URL as http://localhost:3000/. And choose Access tokens for authorization endpoint.



Platform configurations

Depending on the platform or device this application is targeting, additional configuration may be required su redirect URIs, specific authentication settings, or fields specific to the platform.

Supported account types

Who can use this application or access this API?

- [✓ Accounts in this organizational directory only (ETL only - Single tenant)]
- Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)

Due to temporary differences in supported functionality, we don't recommend enabling personal Microsoft accounts for an existing registration. If you need to enable personal accounts, you can do so using the manifest editor. Learn more about these restrictions.

Web applications

Web
Build, host, and deploy a web server application. .NET, Java, Python

Single-page application
Configure browser client applications and progressive web applications. Javascript.

Mobile and desktop applications

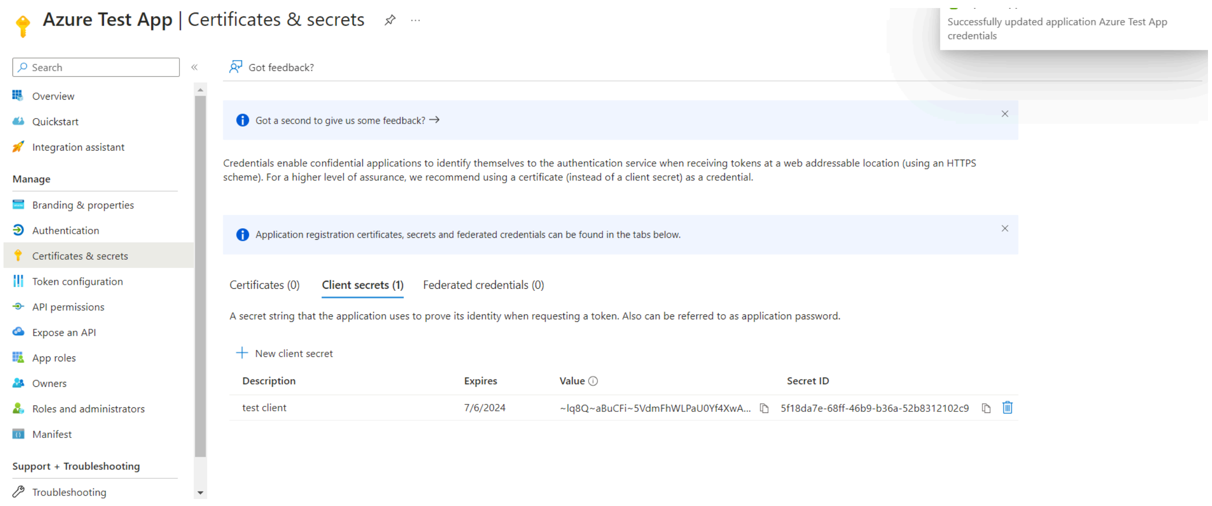
iOS / macOS
Objective-C, Swift, Xamarin

Android
Java, Kotlin, Xamarin

Mobile and desktop applications
Windows, UWP, Console, IoT & Limited-entry Devices, Classic iOS + Android

Generate client secret for authentication:

To generate client secret for authentication, go to **Certificates and Secrets** and generate client secret and copy that secret value.



Successfully updated application Azure Test App credentials

Got a second to give us some feedback? →

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

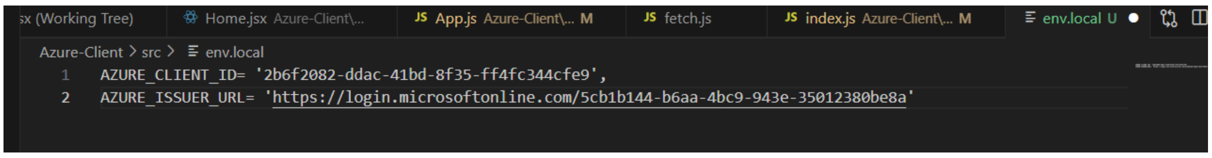
Application registration certificates, secrets and federated credentials can be found in the tabs below.

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

Description	Expires	Value	Secret ID
test client	7/6/2024	~lq8Q~aBuCFi~5VdmFhWLPaU0Yf4XwA...	5f18da7e-68ff-46b9-b36a-52b8312102c9

Add Azure Properties in React App:

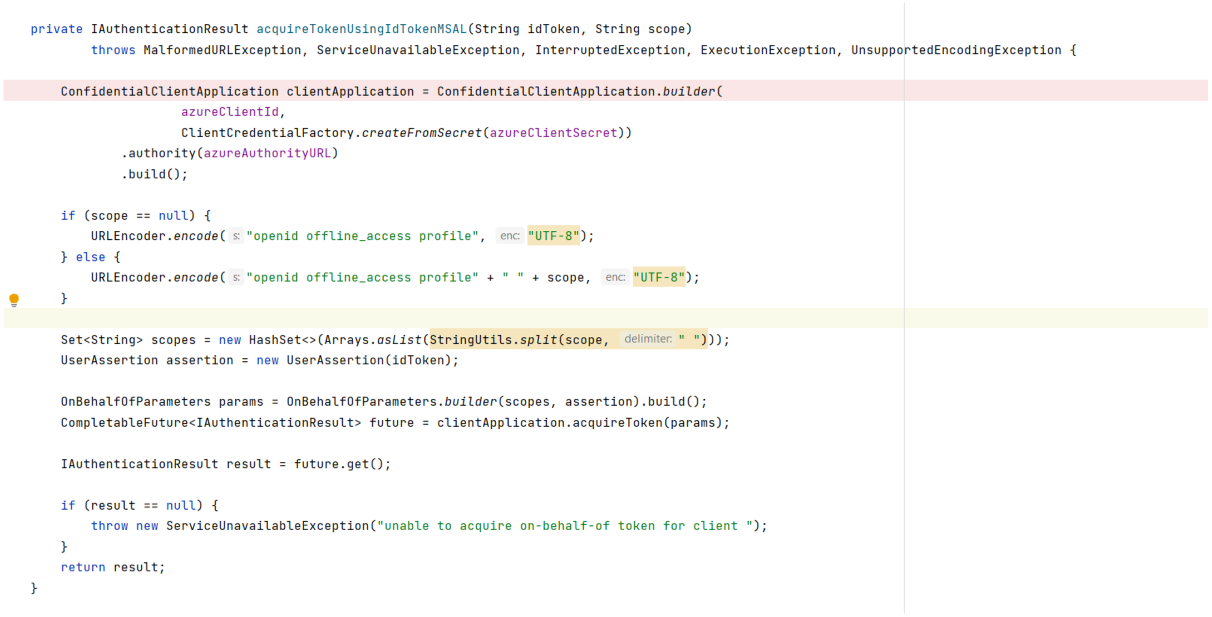
We need to create a react project and add azure configuration properties in the React application in the env file.


x (Working Tree)
Home.jsx Azure-Client\...
App.js Azure-Client\... M
fetch.js
index.js Azure-Client\... M
env.local U
Azure-Client > src > env.local

```
AZURE_CLIENT_ID= '2b6f2082-ddac-41bd-8f35-ff4fc344cfe9',
AZURE_ISSUER_URL= 'https://login.microsoftonline.com/5cb1b144-b6aa-4bc9-943e-35012380be8a'
```

So by running the react application, we will be able to login using Microsoft Azure and get the **ID Token** of azure.

And with the help of ID Token, we can get access tokens using msal4j dependency and using access tokens, we can access APIs of Microsoft Azure.

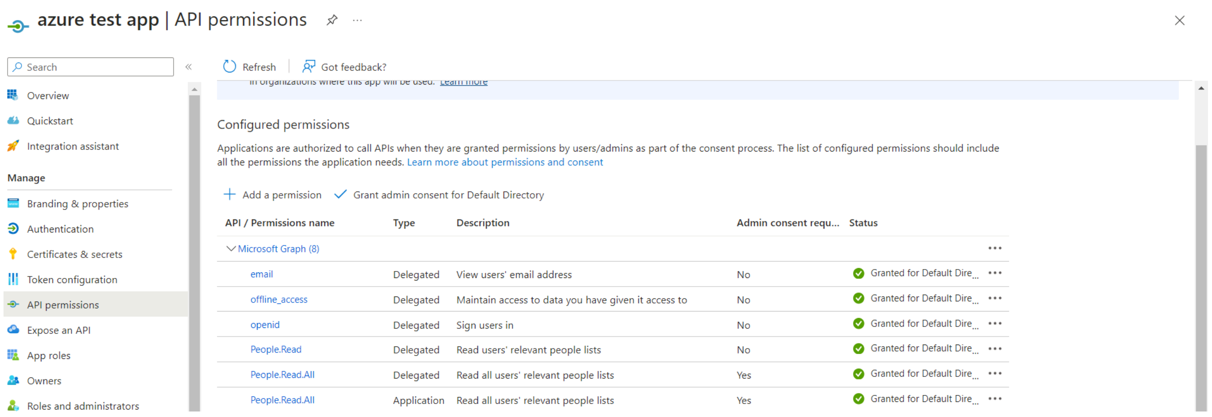


```
private IAuthenticationResult acquireTokenUsingIdTokenMSAL(String idToken, String scope)
        throws MalformedURLException, ServiceUnavailableException, InterruptedException, ExecutionException, UnsupportedEncodingException {

    ConfidentialClientApplication clientApplication = ConfidentialClientApplication.builder(
                    azureClientId,
                    ClientCredentialFactory.createFromSecret(azureClientSecret))
            .authority(azureAuthorityURL)
            .build();

    if (scope == null) {
        URLEncoder.encode( s: "openid offline_access profile",  enc: "UTF-8");
    } else {
        URLEncoder.encode( s: "openid offline_access profile" + " " + scope,  enc: "UTF-8");
    }

    Set<String> scopes = new HashSet<>(Arrays.asList(StringUtils.split(scope,  delimiter: " ")));
    UserAssertion assertion = new UserAssertion(idToken);

    OnBehalfOfParameters params = OnBehalfOfParameters.builder(scopes, assertion).build();
    CompletableFuture<IAuthenticationResult> future = clientApplication.acquireToken(params);

    IAuthenticationResult result = future.get();

    if (result == null) {
        throw new ServiceUnavailableException("unable to acquire on-behalf-of token for client ");
    }
    return result;
}
```

Here we have used Access Token in Bearer authentication to fetch the profile data of signed in user by using

API Endpoint: https://graph.microsoft.com/v1.0/me
Method: Post
Authorization: Bearer + AccessToken

To access the Azure APIs, we need to provide permissions from the App **Permission** section and provide admin consent to the directory.



Configured permissions

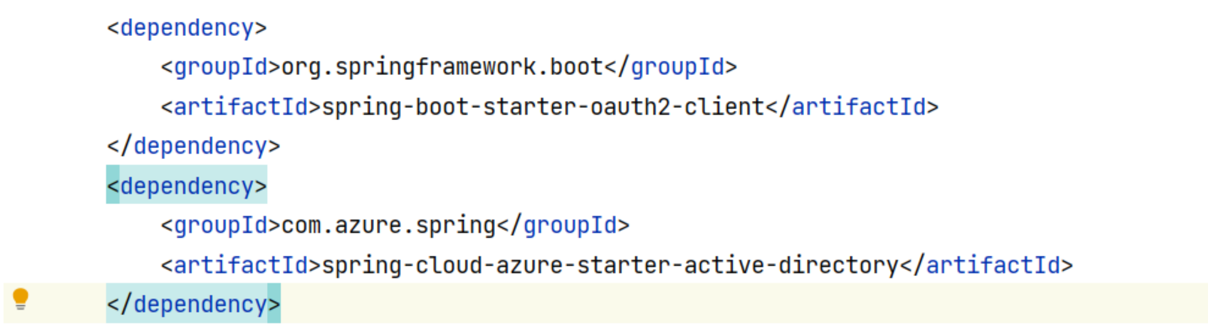
Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent

+ Add a permission ✓ Grant admin consent for Default Directory

API / Permissions name	Type	Description	Admin consent requ...	Status
Microsoft Graph (8)				
email	Delegated	View users' email address	No	✓ Granted for Default Dire...
offline_access	Delegated	Maintain access to data you have given it access to	No	✓ Granted for Default Dire...
openid	Delegated	Sign users in	No	✓ Granted for Default Dire...
People.Read	Delegated	Read users' relevant people lists	No	✓ Granted for Default Dire...
People.Read.All	Delegated	Read all users' relevant people lists	Yes	✓ Granted for Default Dire...
People.Read.All	Application	Read all users' relevant people lists	Yes	✓ Granted for Default Dire...

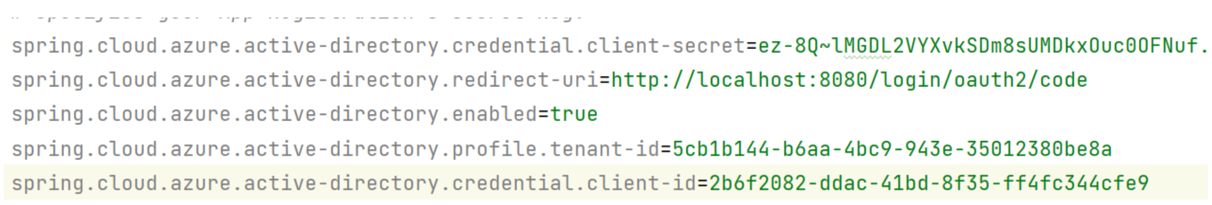
Authenticate using Java

To authenticate Microsoft Azure using java, we need to add below dependency to our pom.xml file



```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-oauth2-client</artifactId>
</dependency>
<dependency>
    <groupId>com.azure.spring</groupId>
    <artifactId>spring-cloud-azure-starter-active-directory</artifactId>
</dependency>
```

And add client secret value, redirect URL, tenant id and client Id



```
spring.cloud.azure.active-directory.credential.client-secret=ez-8Q~lMGDL2VYXvkSDm8sUMDkxOuc0OFNuf.
spring.cloud.azure.active-directory.redirect-uri=http://localhost:8080/login/oauth2/code
spring.cloud.azure.active-directory.enabled=true
spring.cloud.azure.active-directory.profile.tenant-id=5cb1b144-b6aa-4bc9-943e-35012380be8a
spring.cloud.azure.active-directory.credential.client-id=2b6f2082-ddac-41bd-8f35-ff4fc344cfe9
```

So by configuring azure properties, we can successfully login to azure using java and react.