**Demystifying OAuth Protocol in Azure using Azure Active Directory – Quick guide to Client Credentials flow**

OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords.

**OAuth Grant Types**

The OAuth framework specifies several grant types for different use cases, as well as a framework for creating new grant types.

The most common OAuth grant types are listed below.

* [Authorization Code](https://oauth.net/2/grant-types/authorization-code/)
* [PKCE](https://oauth.net/2/pkce/)
* [Client Credentials](https://oauth.net/2/grant-types/client-credentials/)
* [Device Code](https://oauth.net/2/grant-types/device-code/)
* [Refresh Token](https://oauth.net/2/grant-types/refresh-token/)

**Client Credentials**

The Client Credentials grant type is used by clients to obtain an access token outside of the context of a user.

This is typically used by clients to access resources about themselves rather than to access a user's resources.

In this article – Let us try and demystify the Client Credentials flow for two scenarios in Azure Active Directory – **Scenario 1 – Granting Bearer token based access to storage account blob from Postman & Scenario 2 – Protecting API using bearer token (access from Postman)**

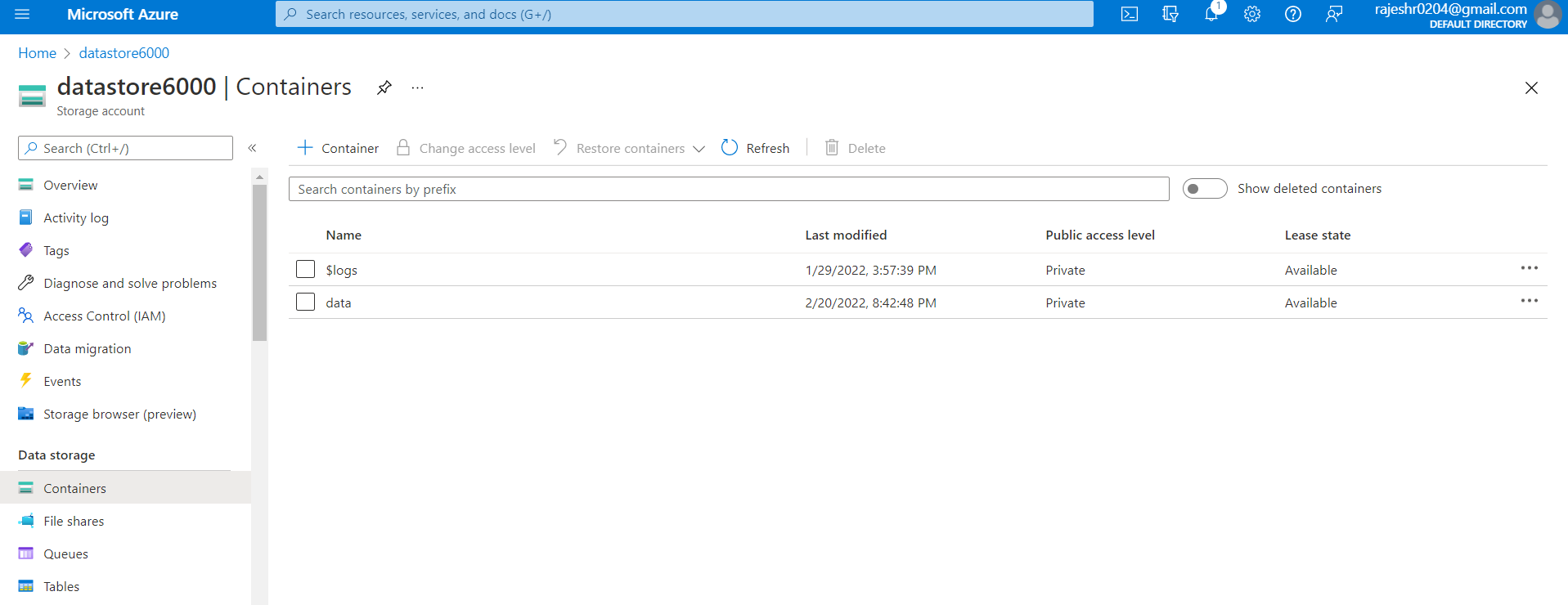
**Scenario 1** – Granting access to an Azure Storage Account resource from POSTMAN. This particular scenario does not have specific user context and has to be accomplished through App Registration process by registering PostMan as an application in the Azure Active Directory.

The high level flow is as follows –

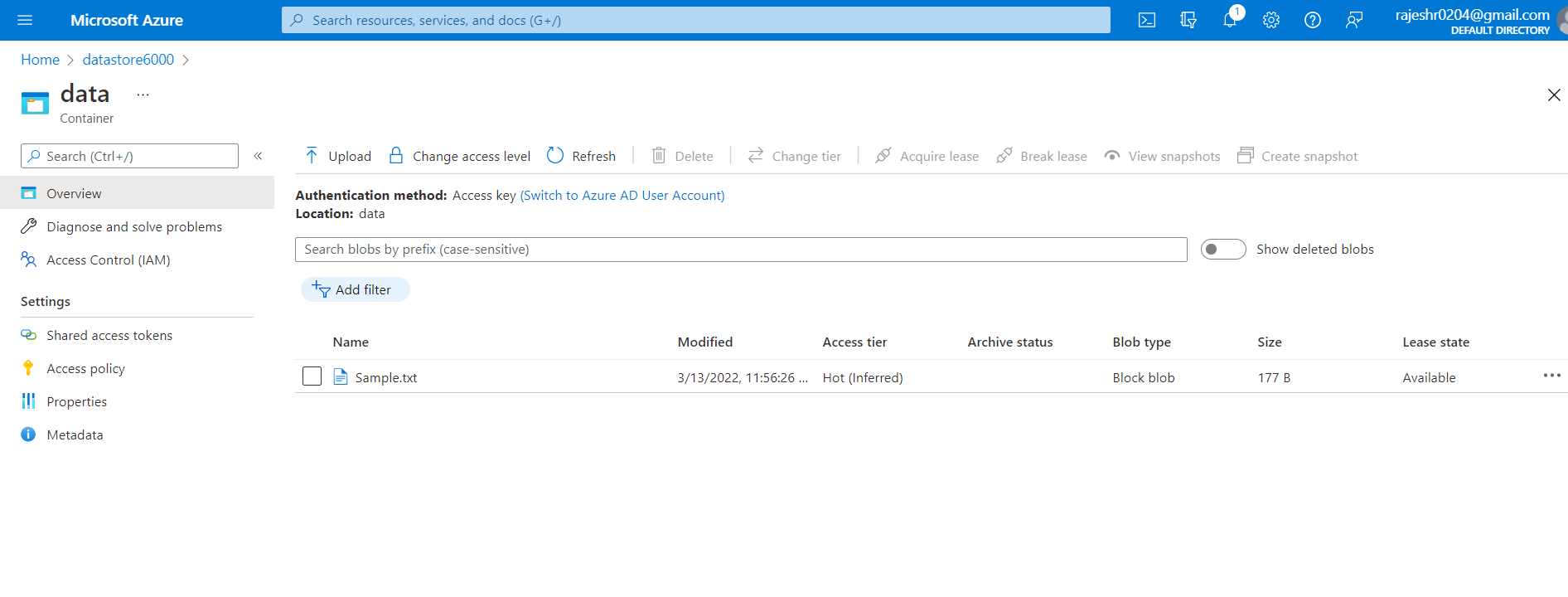
1. Register the Postman application as a virtual application in Azure Active Directory using tenant id, client and client secrets
2. Provide role assignments for Postman against the storage account for Reader and BlobStorage Reader.
3. Get the bearer token by hitting the Authorization token end point
4. Use the bearer token to read a JSON file stored in the Storage account

**Scenario 1 – Solution Steps**

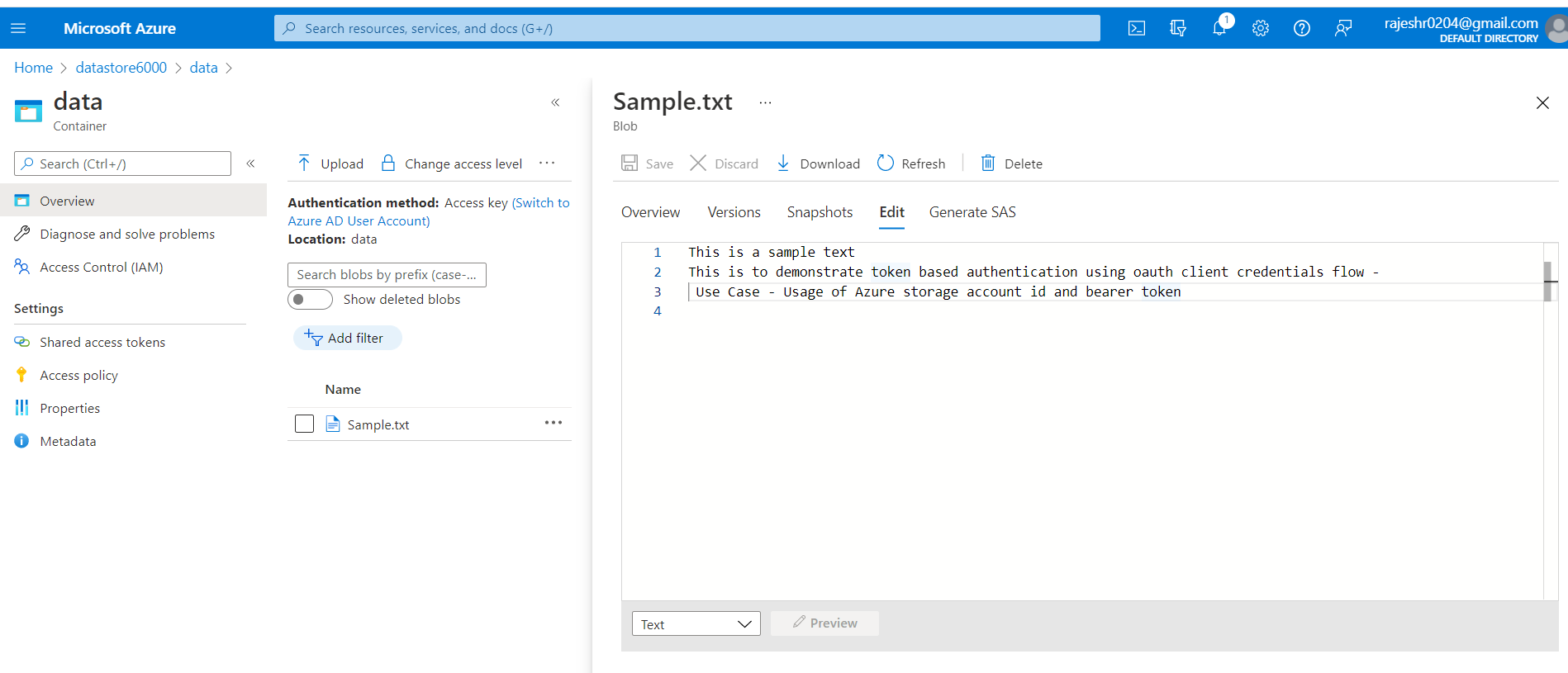
**Step 1** – Create a Azure Storage account and create containers under the storage account. In the below screenshot – You can view the datastorr6000 storage account and two containers called $logs (created by default) and container ‘data’ that contains the destination ‘Sample.txt’ that Postman needs to access. Make sure to create it as a Private Blob



**Step 2** – ‘Sample.txt’ present in the data folder in the storage account

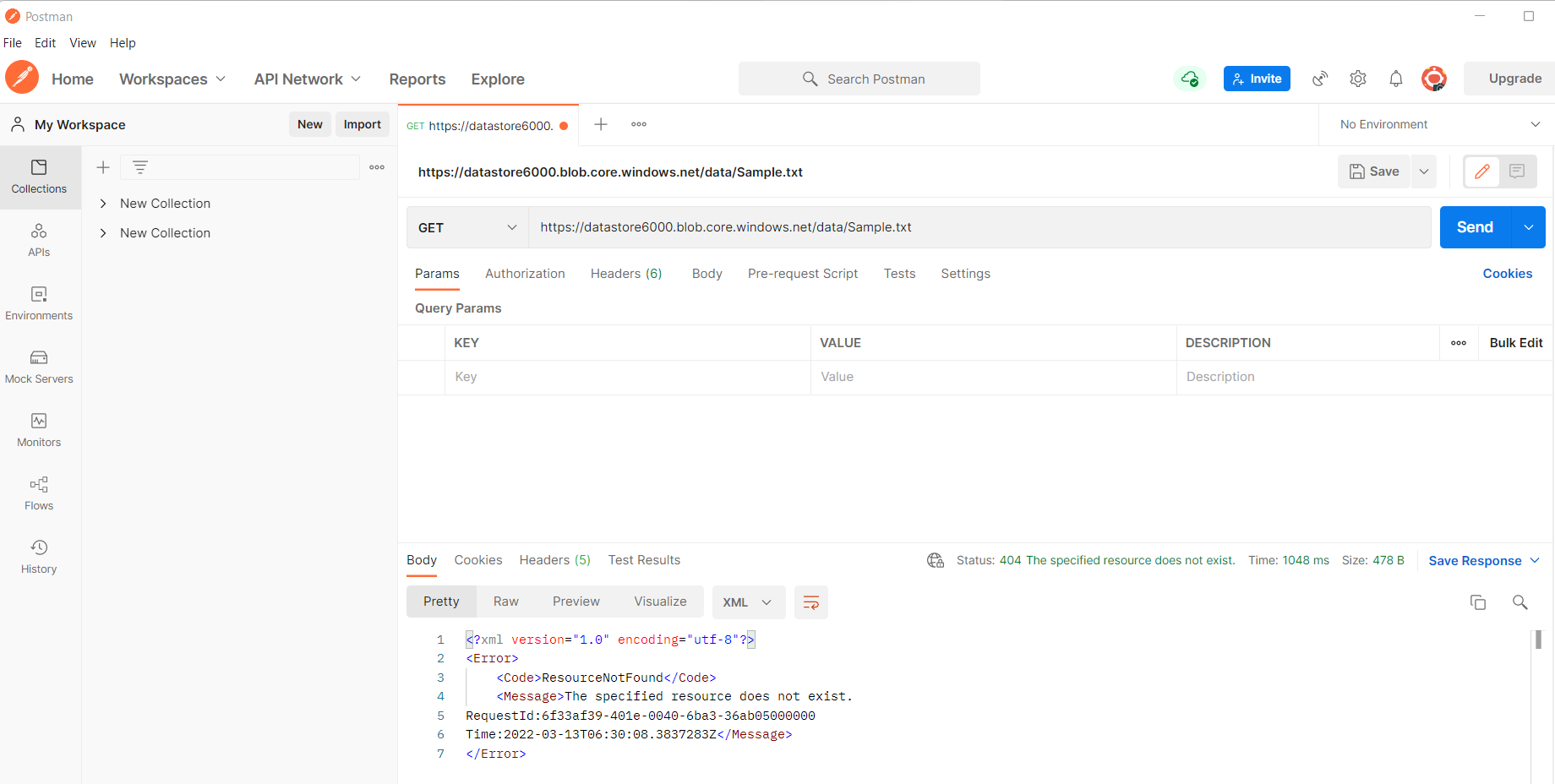


**Step 3** – Edit the Sample txt and provide a sample text

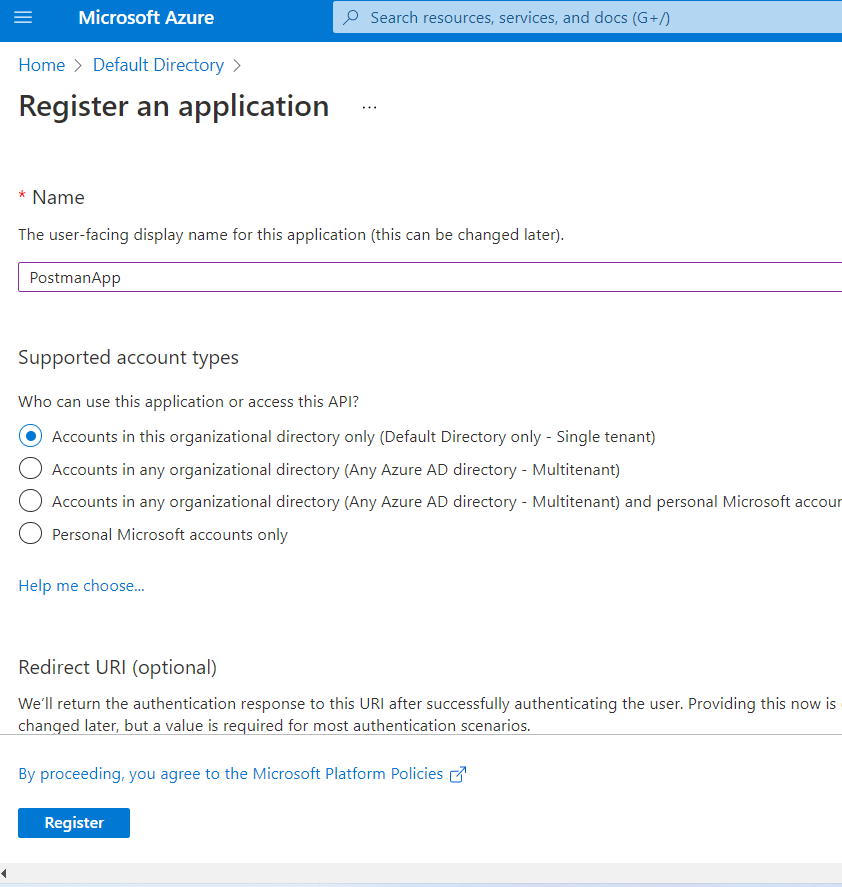


**Step 4** – Take the end point URL of the Sample.txt and enter it in Postman tool. As you observe, the resource is not accessible without a bearer token

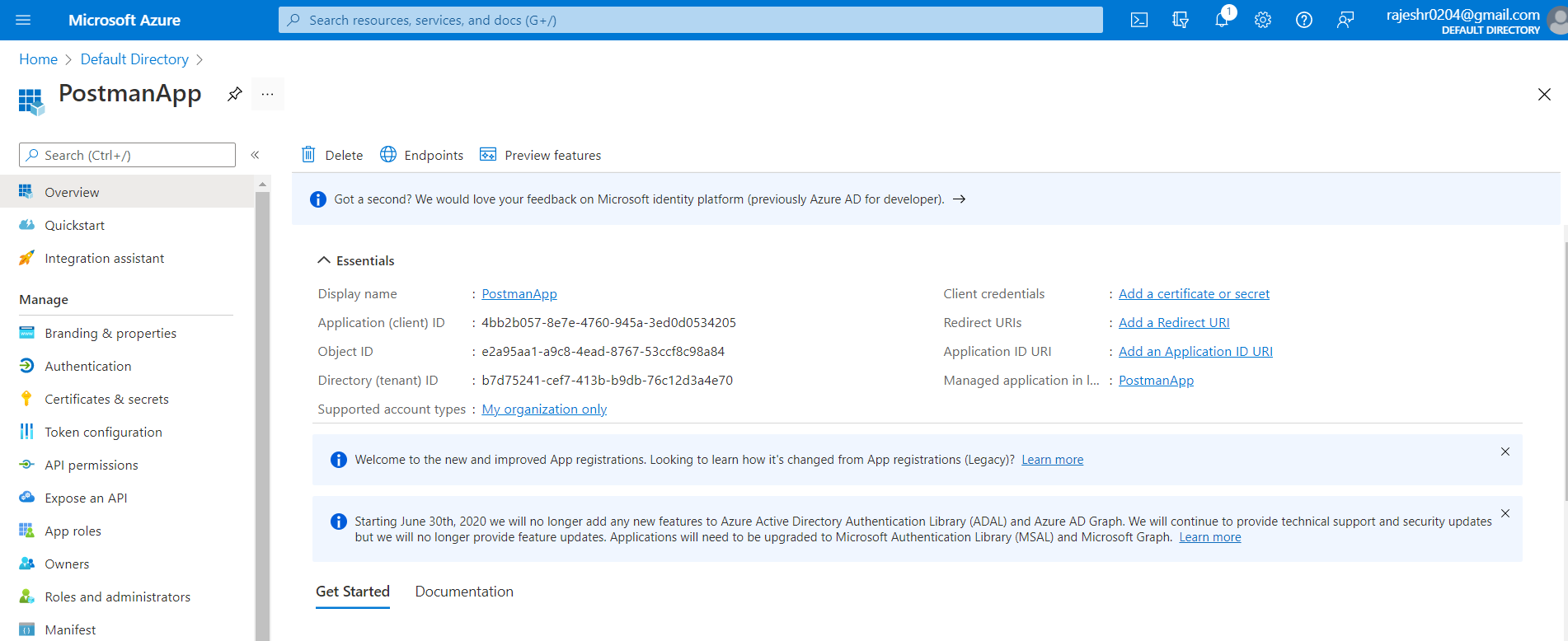
<https://datastore6000.blob.core.windows.net/data/Sample.txt>

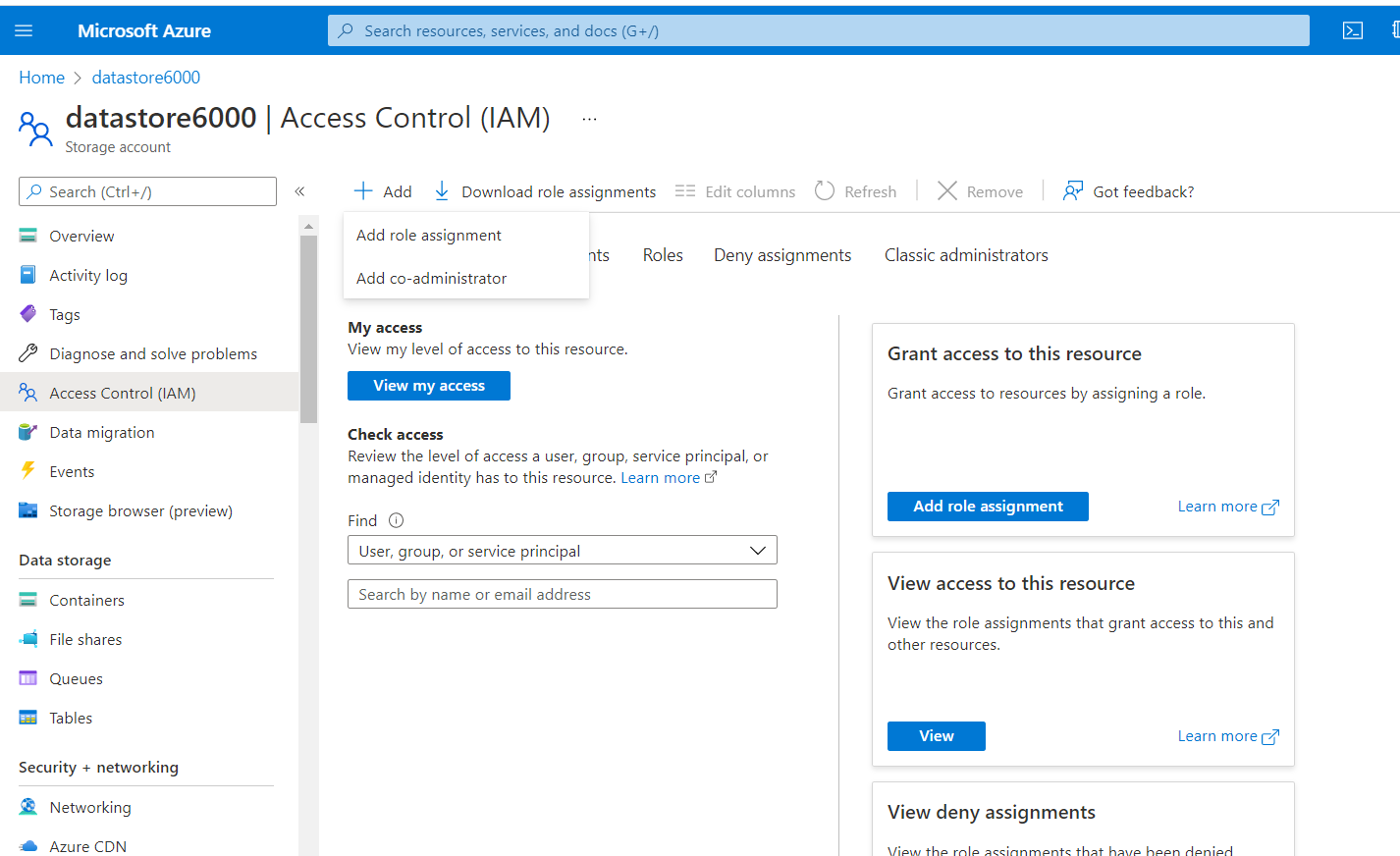


**Step 5** – Register the Postman app in Azure Active Directory->App Registrations

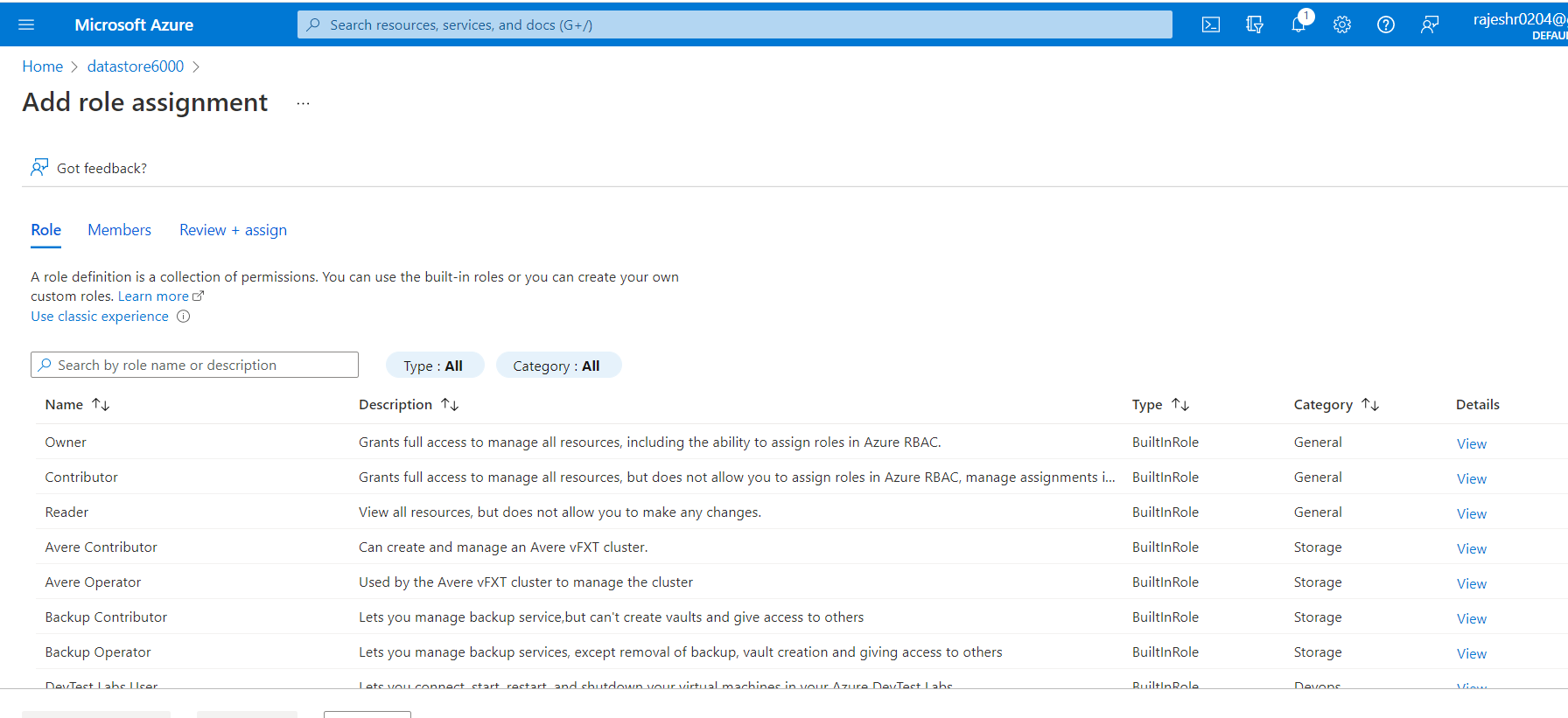


**Step 6** – When completed, the Postman app will appears as show below

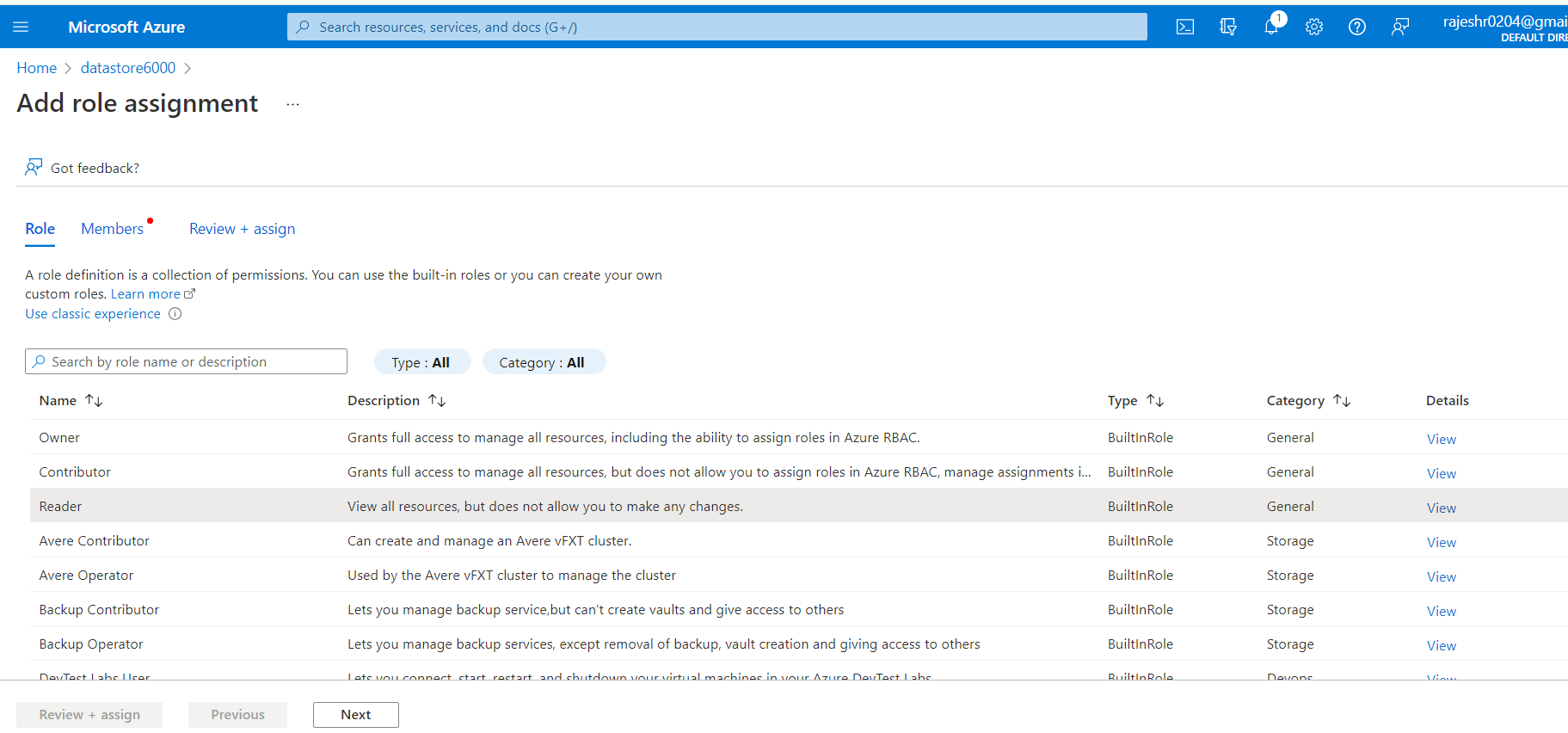


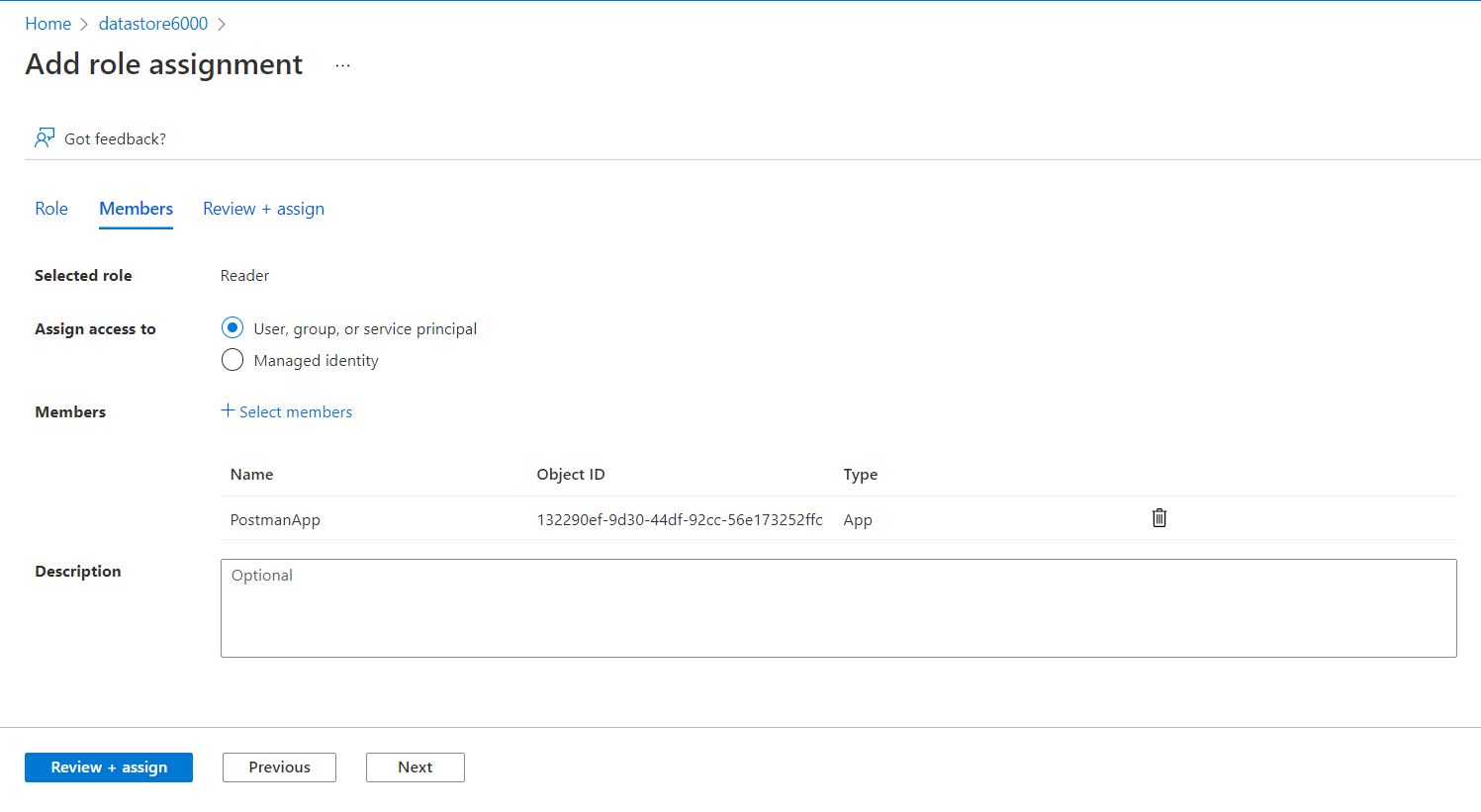
**Step 7** – Click on the datastore6000 storage account->Add Role assignment

You will find list of all role assignments

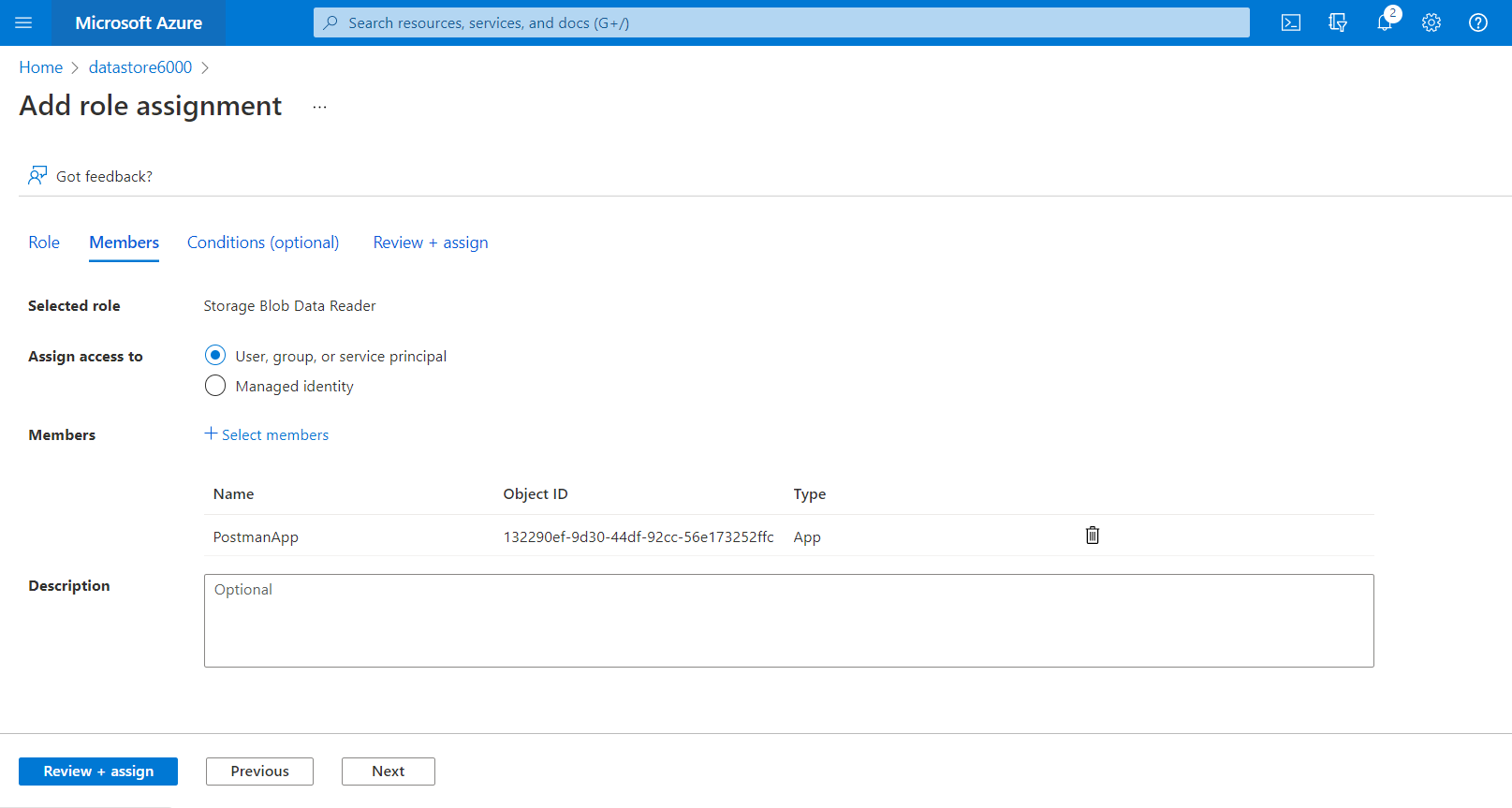


Select **Reader** role->Select Members->Postman App->Add role assignment and click on Review+assign



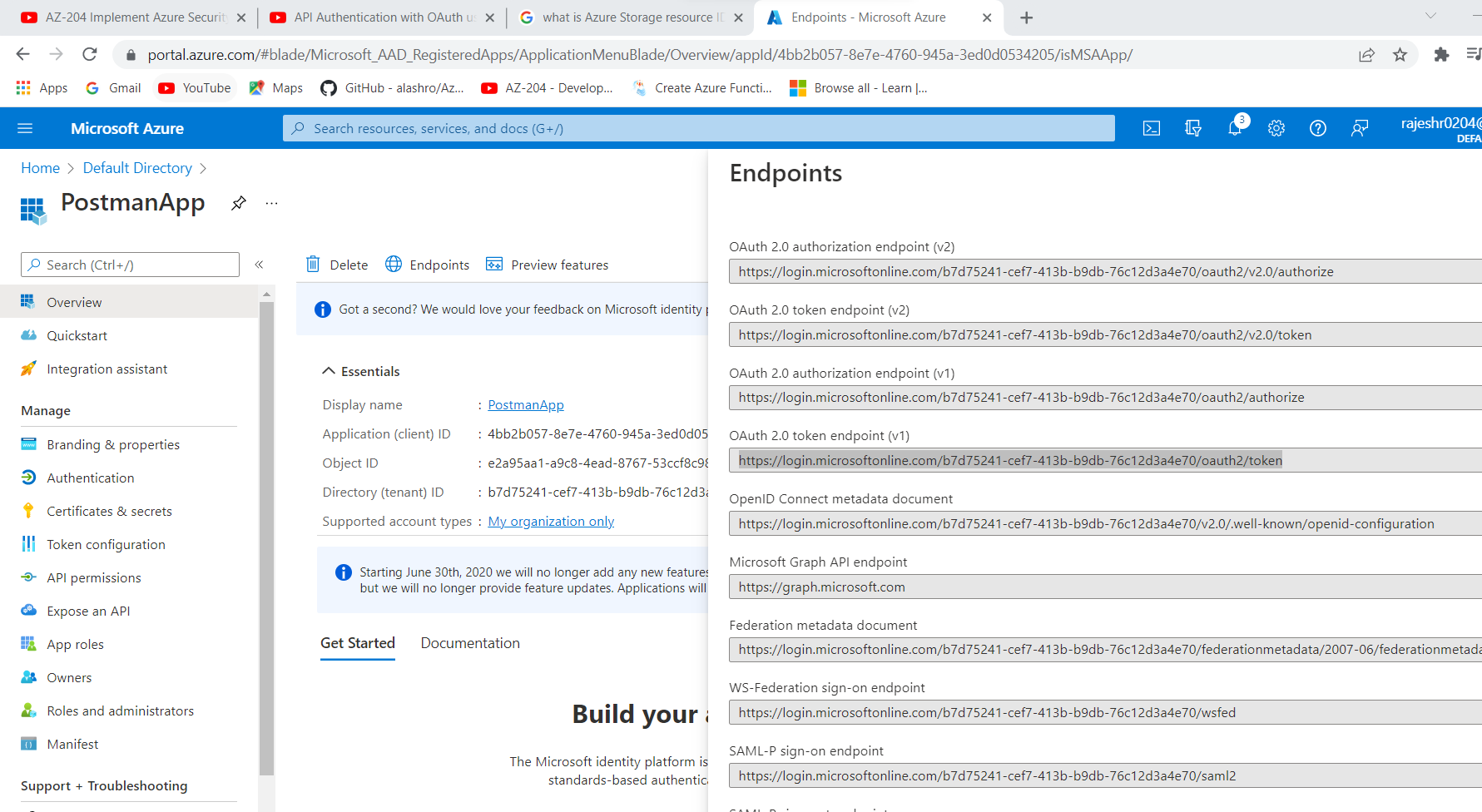


Similarly Select **Blob Data Reader** role->Select Members->Postman App->Add role assignment and click on Review+assign



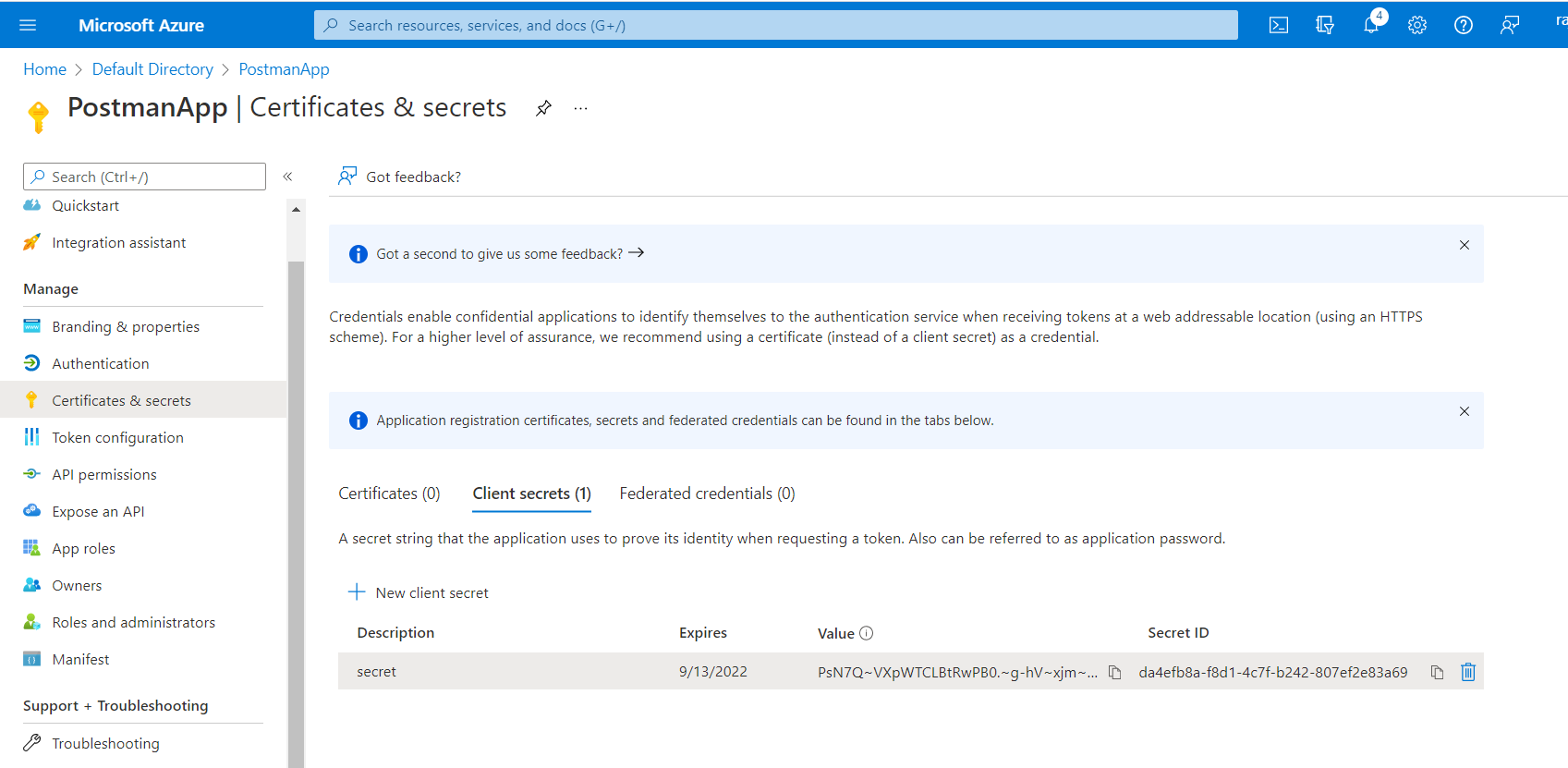
**Step 8** - When completed, the postman virtual app has been granted Reader and Blob Data Reader permissions to the Azure Storage account.

Now select the oAuth2.0 authorization end point on the PostmanApp. This is the URL that will provide the authorization bearer token for the Postman application



As a final configuration step- Create a client secret. The client id and client secret are important parameters to be used in the Postman application for getting the bearer token to get access to the ‘Sample.txt’ which resides on the Azure data storage account datastore6000

Copy and paste the secret as it vanishes in a few minutes after creation



**Step 9**

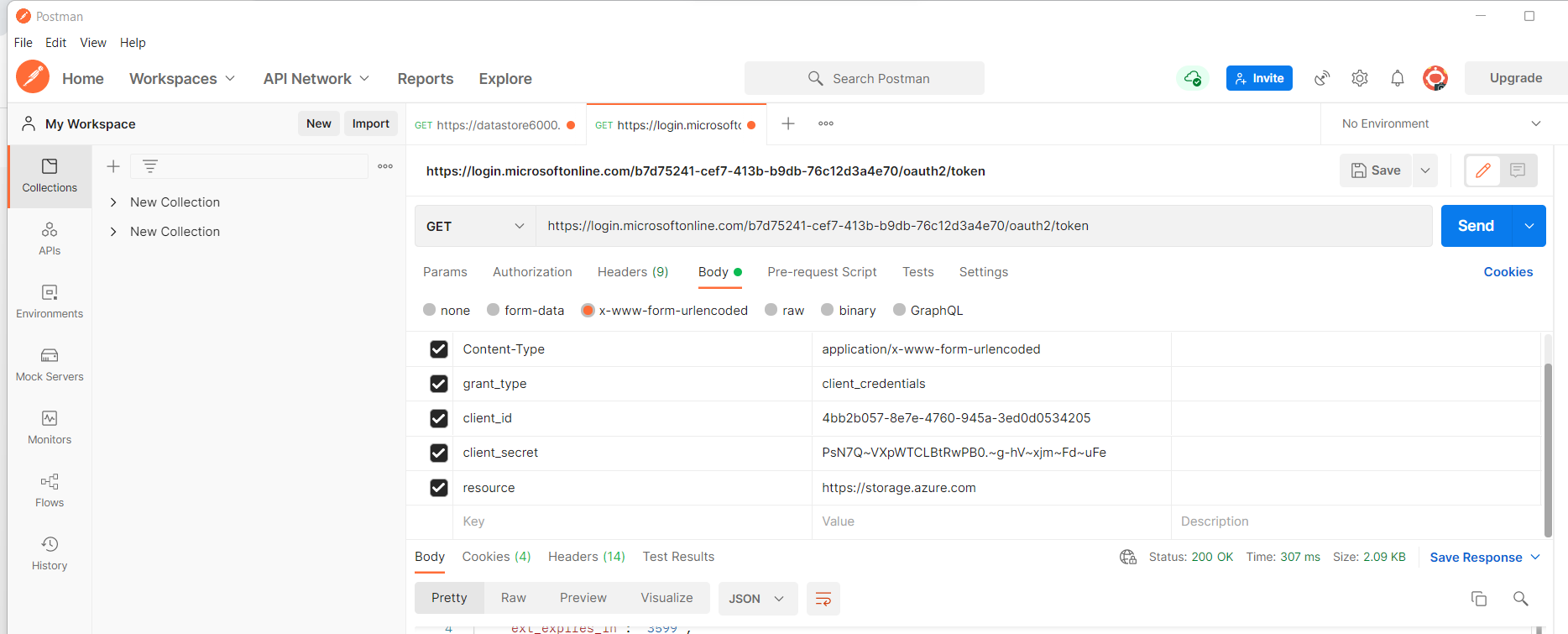
Open the Postman tool and paste the auth end point URL. In the ‘Body’ section, enter the following parameters –

**grant\_type** = client\_credentials

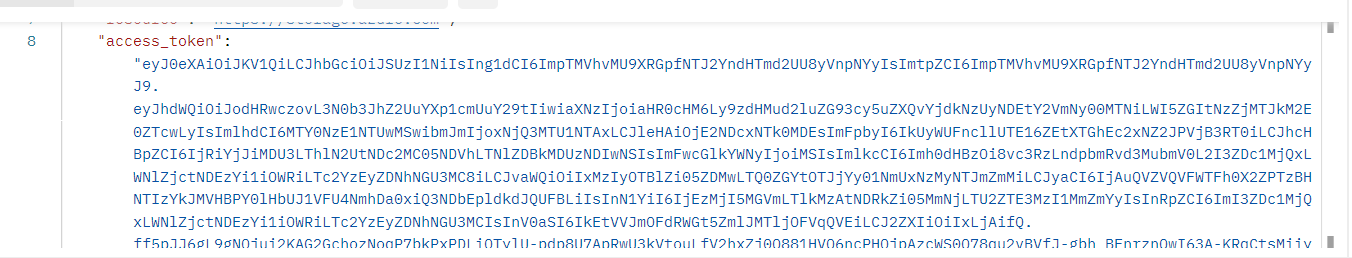
**client\_id** = <<Client id>> of Postman app that you will get from the portal

**client\_secret**=<<secret>> that was created in the earlier step

**resource** = <https://storage.azure.com> (since we are accessing a resource in the azure storage account)



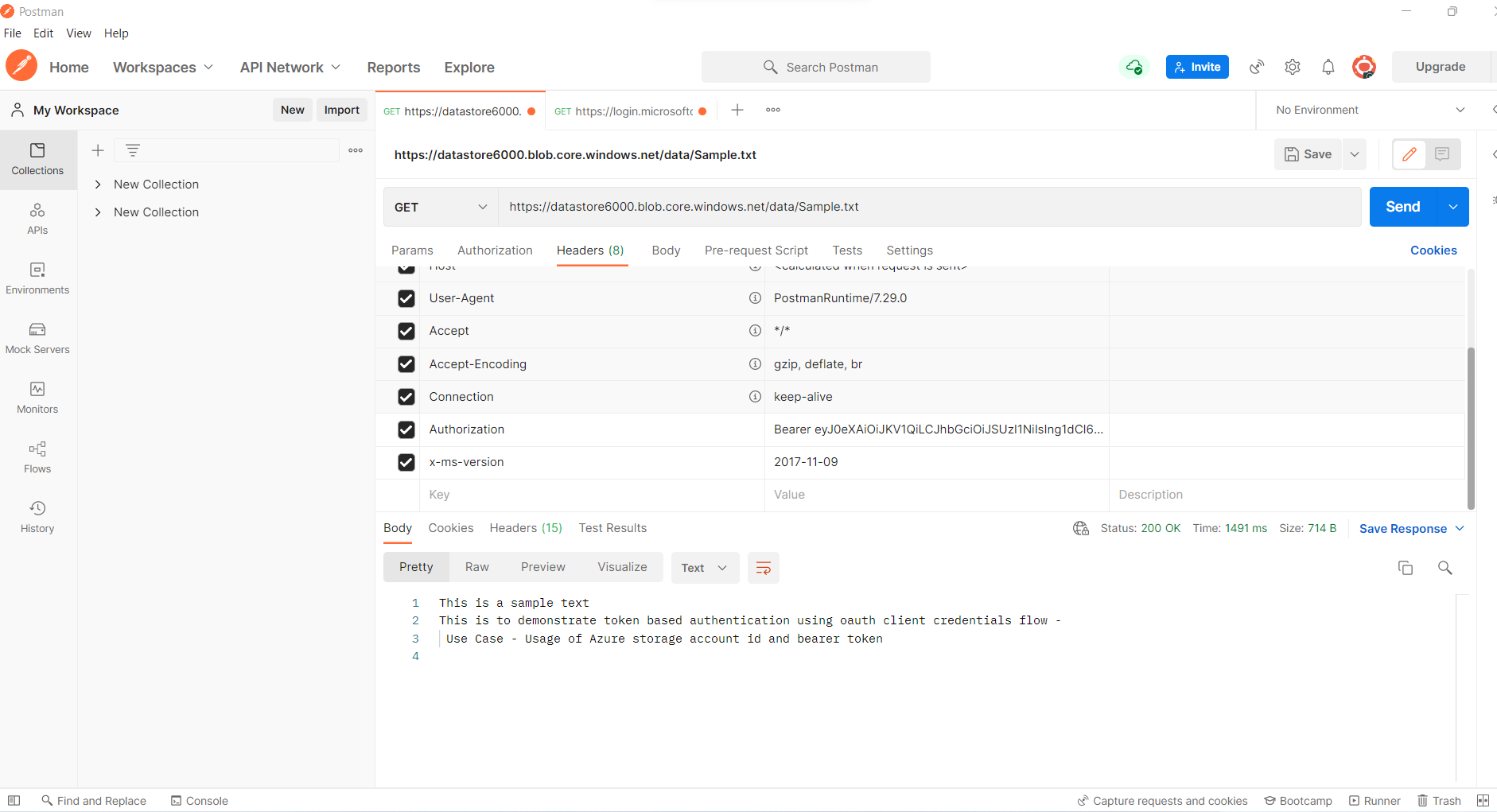
Click on “Send” and you will get the below bearer token



**Step 10** – Hit the end point URL of the Azure storage account (that contains the sample.txt file). Please the URL on Postman and enter the following parameters in the Headers section

**Authorization** – Bearer “<<paste token here>>”

**x-ms-version** – 2017-11-09



As observed – The bearer token has provided Read access to an ‘Private’ Blob on the Azure Storage Account

Do note – This is a sample for reading the Blob data. You can replicate this similarly for Write functions using appropriate role assignments.

**Scenario 2 – Protecting API’s**

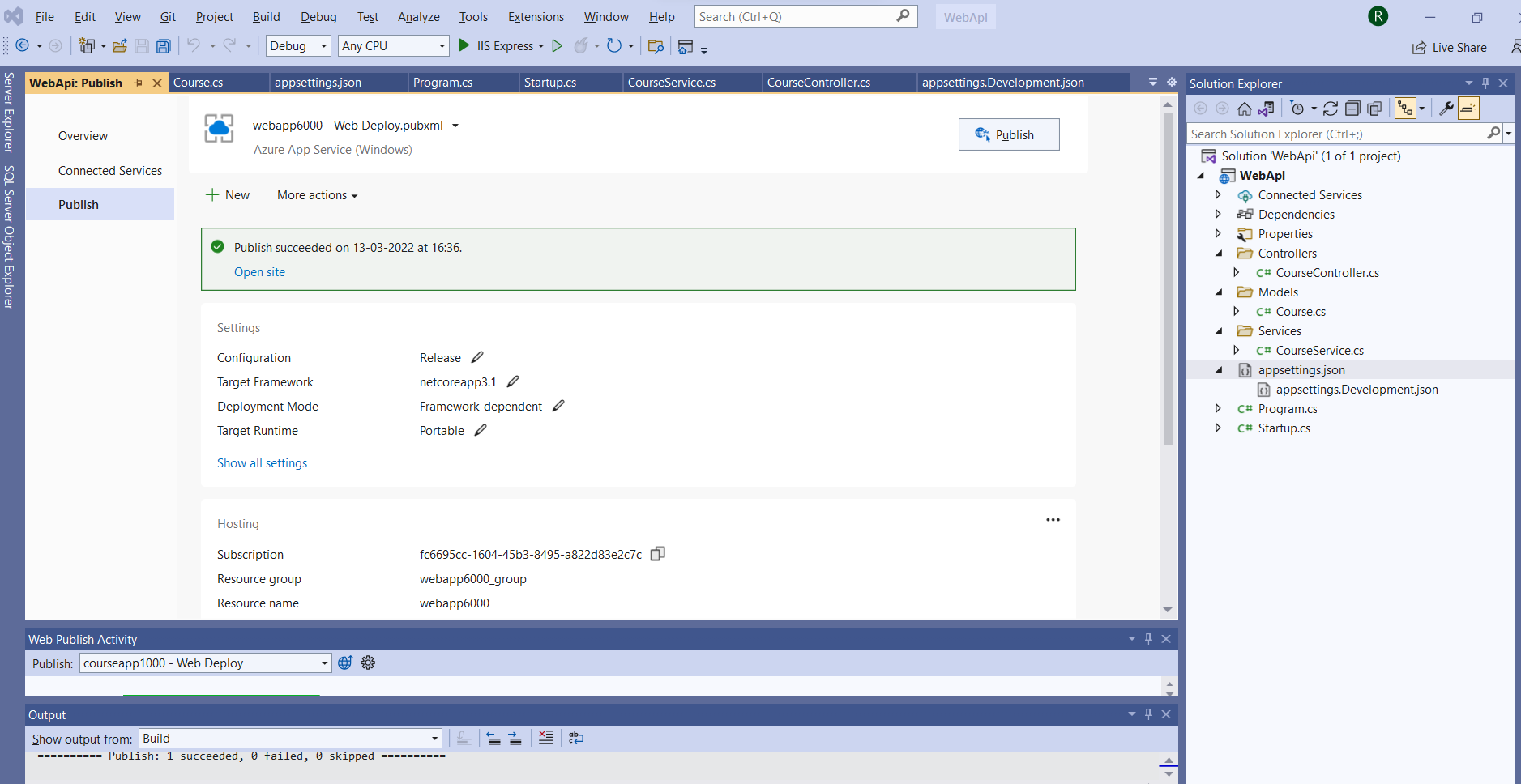
A very common scenario for API is typically when services are exposed as API’s and these API’s are used by third party consumers. In such cases, there is a need to protect the API using proper authorization and authentication.

In such cases, this can be efficiently accomplished through App Registrations. Let us take a simple example of an API being exposed through a Azure Web app. In this case, the Azure Web app would implement an authorize module to ensure that API end points are accessible only after due authorization.

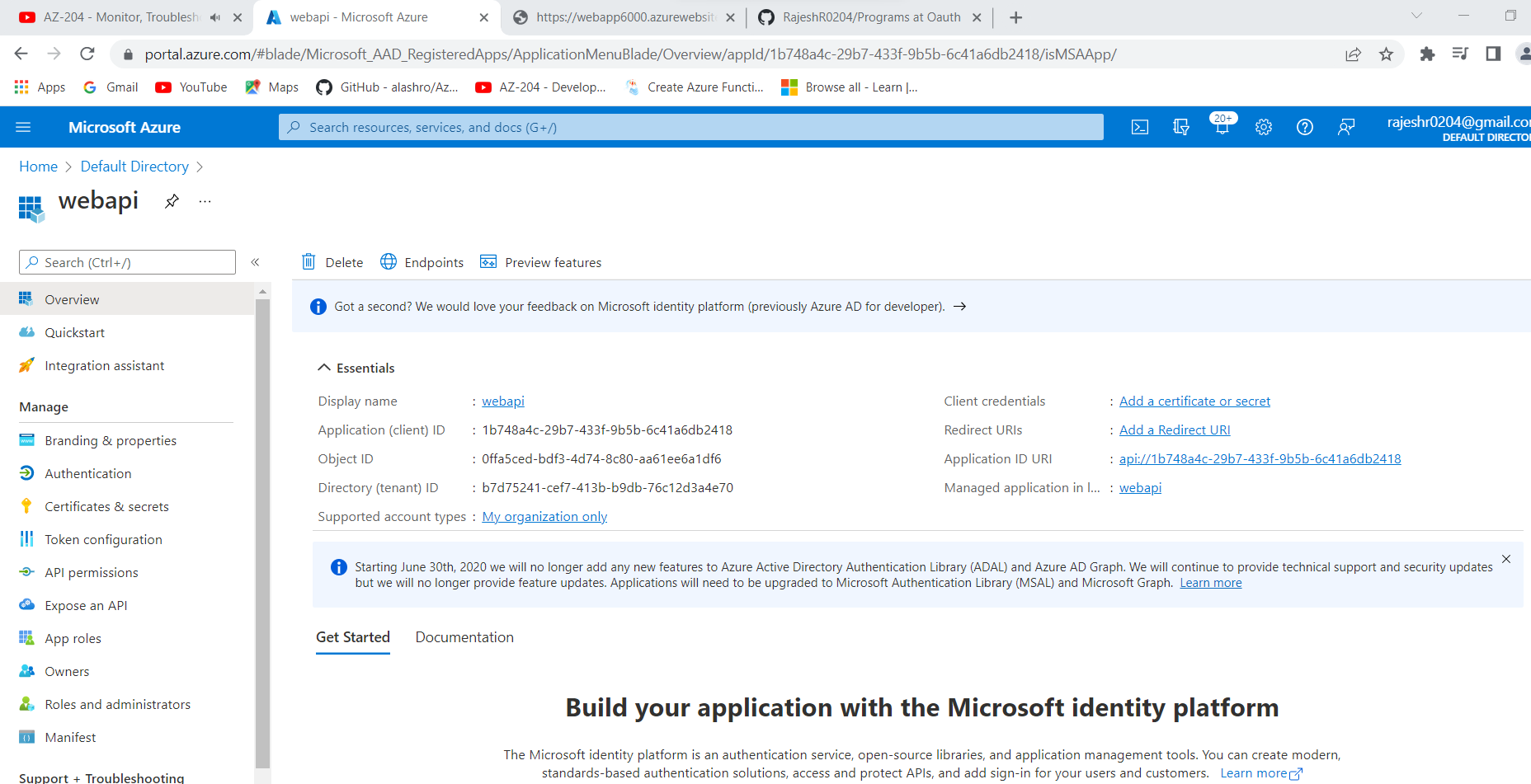
The authorization by itself can be accomplished outside the application code.

Simple REST API that was written on Visual Studio and published on Azure Web app

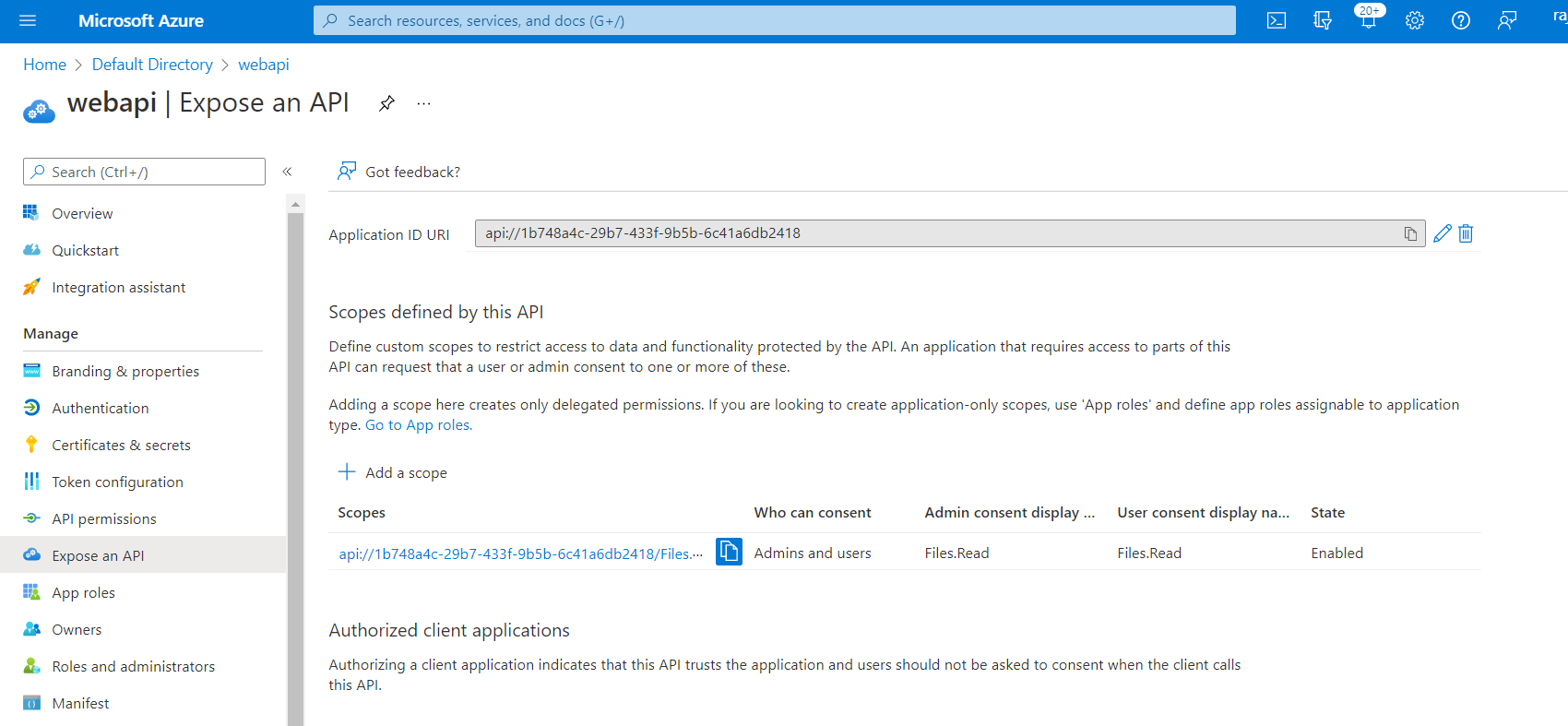
**Step 1** – Simple REST API that reads JSON data stored as a JSON file on a Azure storage account and returns the data to the user



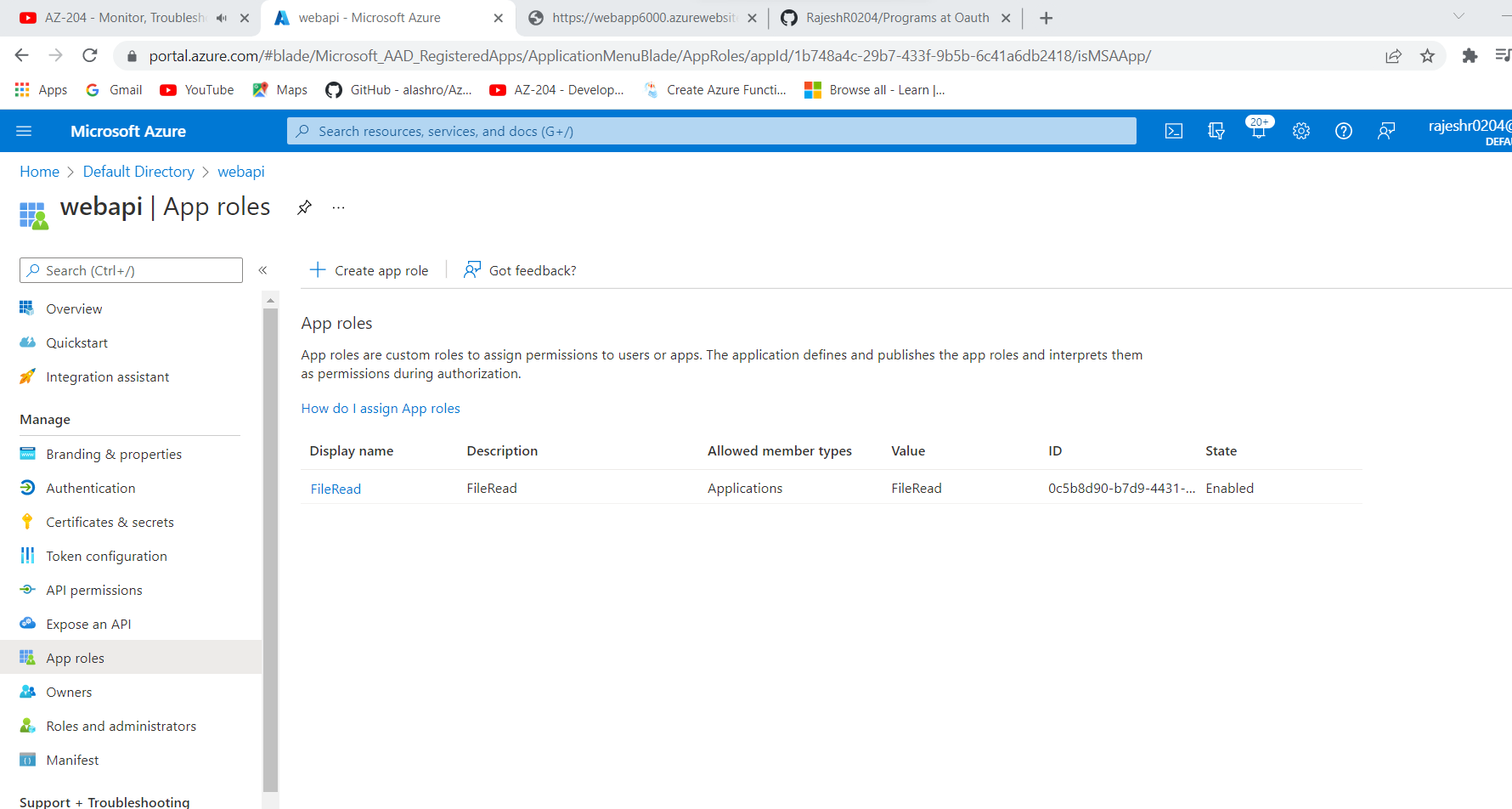
**Step 2 –** Create a Webapi virtual application in the Azure App Registrations section underAzure Active Directory



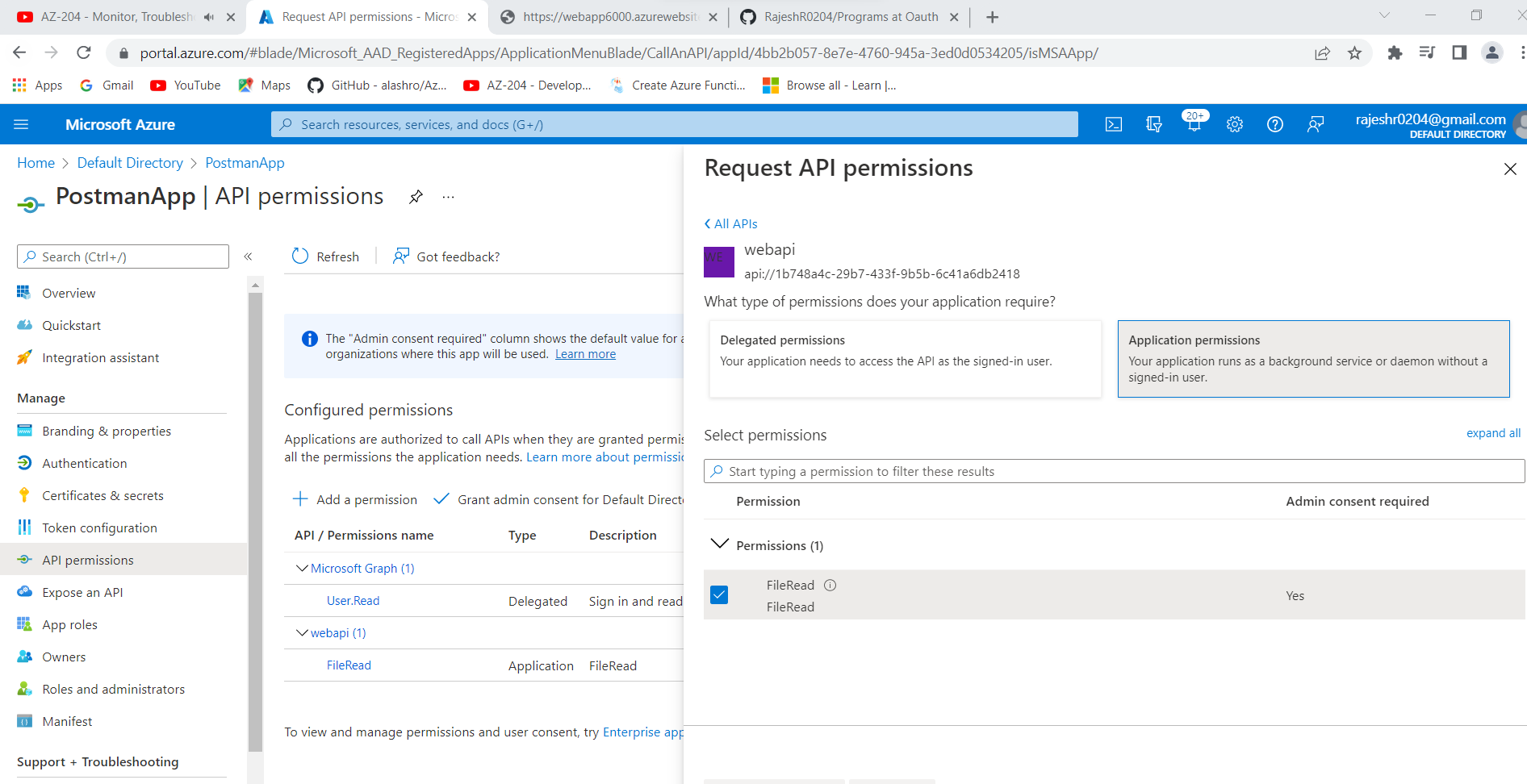
**Step 3** – Expose an API end point for API authorization – Provide File.Read as user permissions and enable the permissions



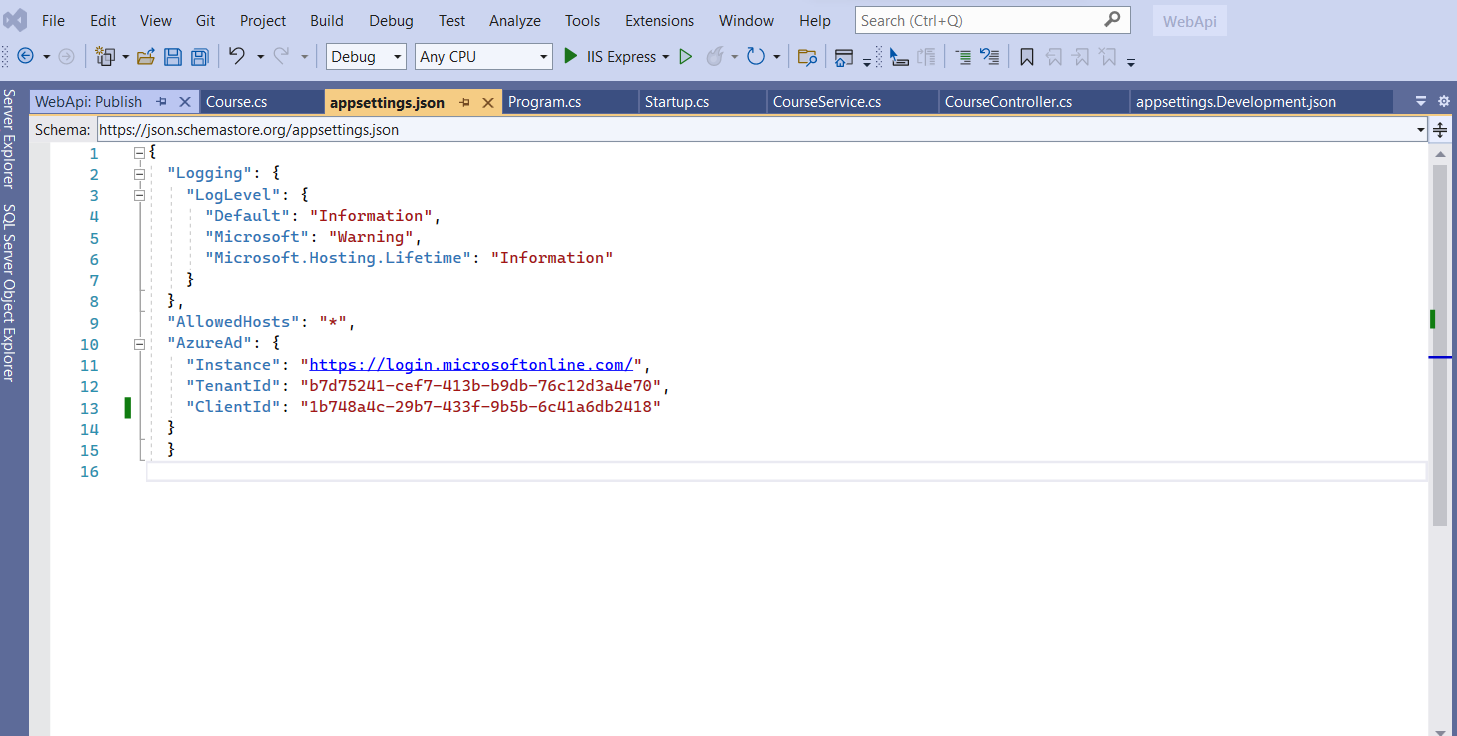
**Step 4** - Create App role for ‘Read ‘ permissions to the API



**Step 5 –** In the postman app – Provide API permissions to the webapi (Select File Read permissions that you had created in the webappi application registration)

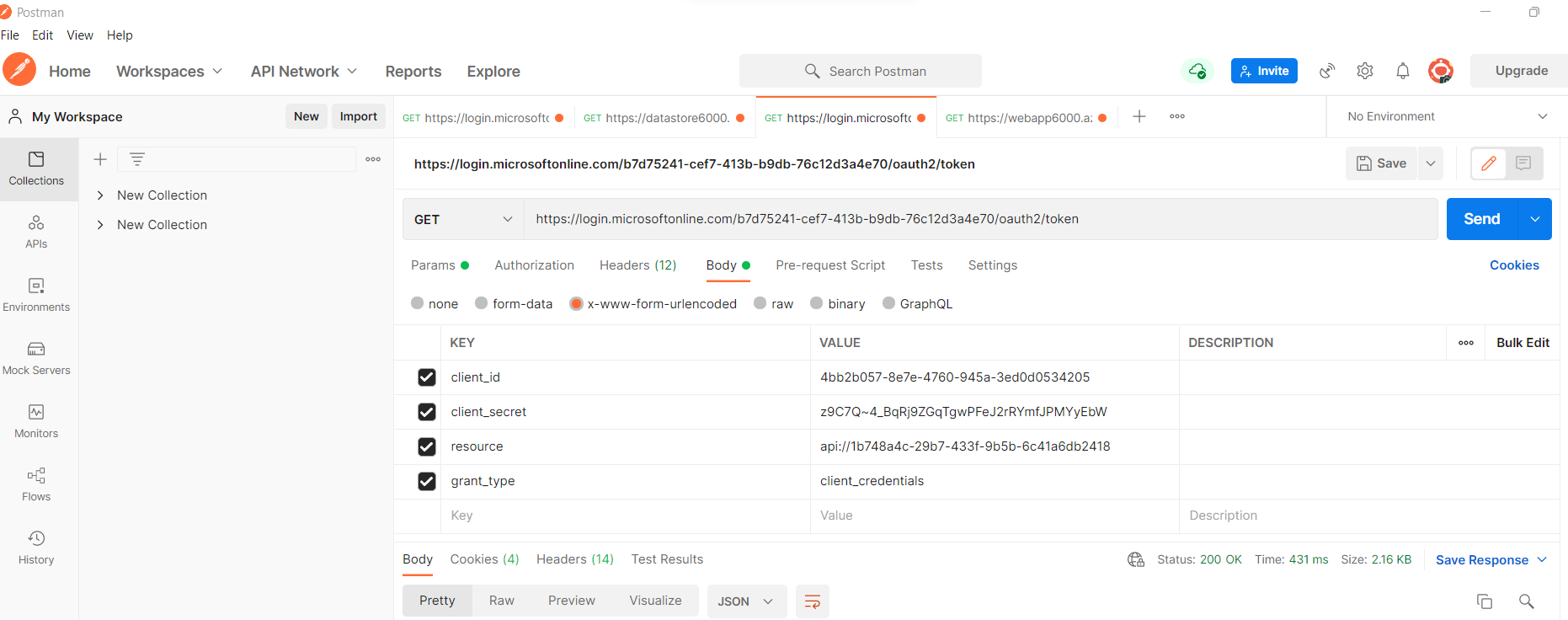


**Step 6** – Ensure that your API refers to the correct client id and tenant id as mentioned in the Azure portal (in appsettings.json)

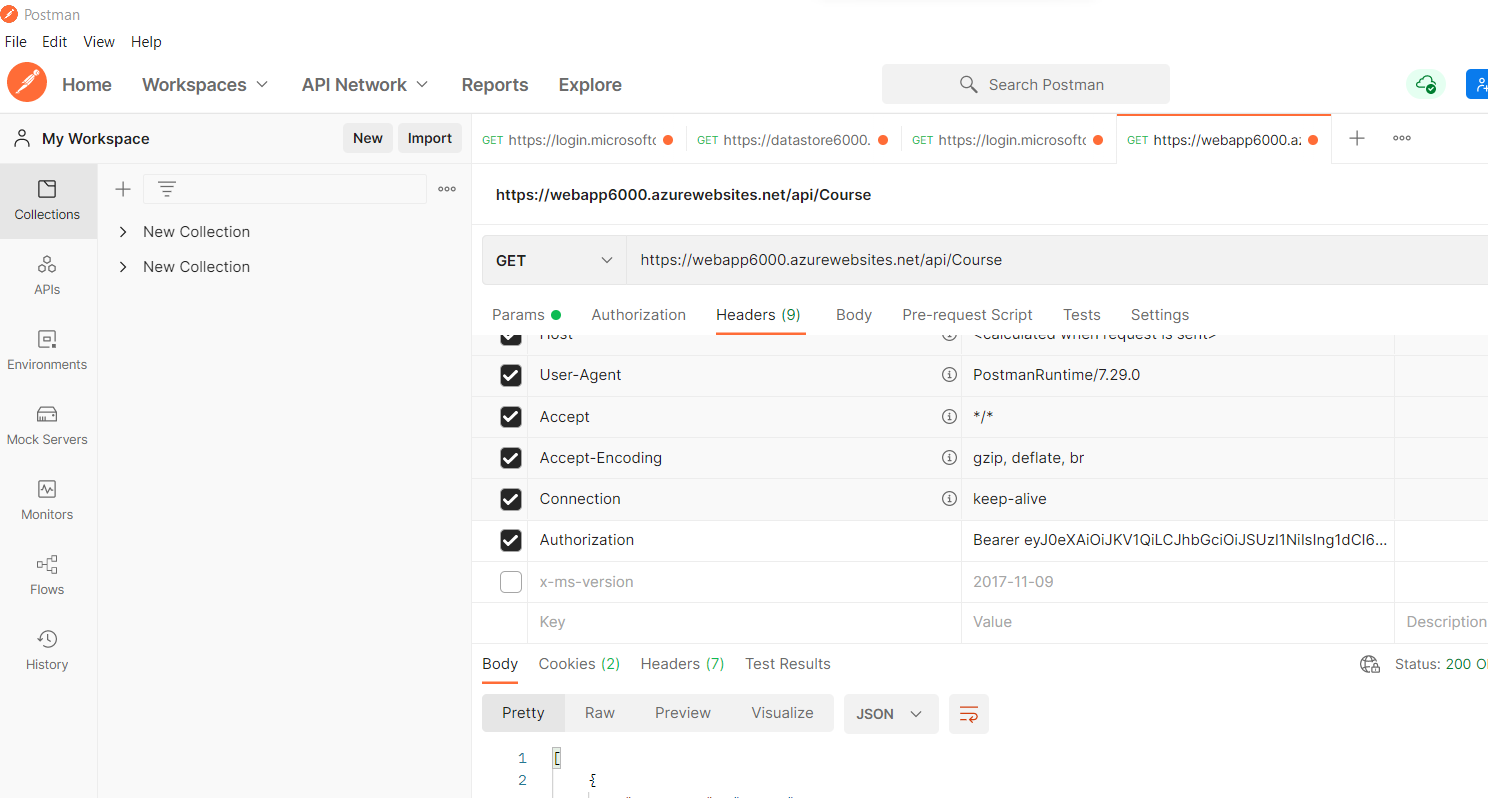


**Step 7** – Use the Postman Oauth end point token URL and enter details in postman

The client\_id and client\_secret in the below case is the client Id and secret of the app that represents the calling client (i.e POSTMAN)



**Step 8** -Use the bearer token to access the API



**Step 9** – Click on “Send” button on POSTMAN to get the API results - Final Outcome

