😊 Application Objects

In Azure Active Directory (Azure AD), application objects represent the apps or services that are integrated with Azure AD for authentication and authorization purposes. These applications can be cloud-based apps, on-premises apps, or custom-developed apps.

Here are some key points about application objects in Azure:

1. **Representation of Apps:** Application objects serve as a representation of the apps registered in Azure AD. Each application object contains metadata about the application, such as its name, ID, type (e.g., web application, native application), and other properties.
2. **Authentication and Authorization:** Application objects enable Azure AD to authenticate and authorize users for access to the registered apps. They define how users can sign in to the app (authentication) and what permissions they have (authorization) once authenticated.
3. **Single Sign-On (SSO):** Azure AD supports single sign-on for applications, allowing users to access multiple apps with a single set of credentials. Application objects facilitate the configuration of SSO settings for each registered app.
4. **Service Principal:** When an application is registered in Azure AD, a service principal object is created to represent the application's identity and permissions. The service principal is used to authenticate the application to Azure AD and access Azure resources.
5. **API Permissions:** Application objects define the API permissions required by the application to access Azure AD resources and other Microsoft services on behalf of users. These permissions are configured in the application's manifest or through the Azure portal.
6. **Lifecycle Management:** Application objects can be managed throughout their lifecycle, including creation, updating, and deletion. Administrators can configure settings, update permissions, and monitor usage for each application registered in Azure AD.

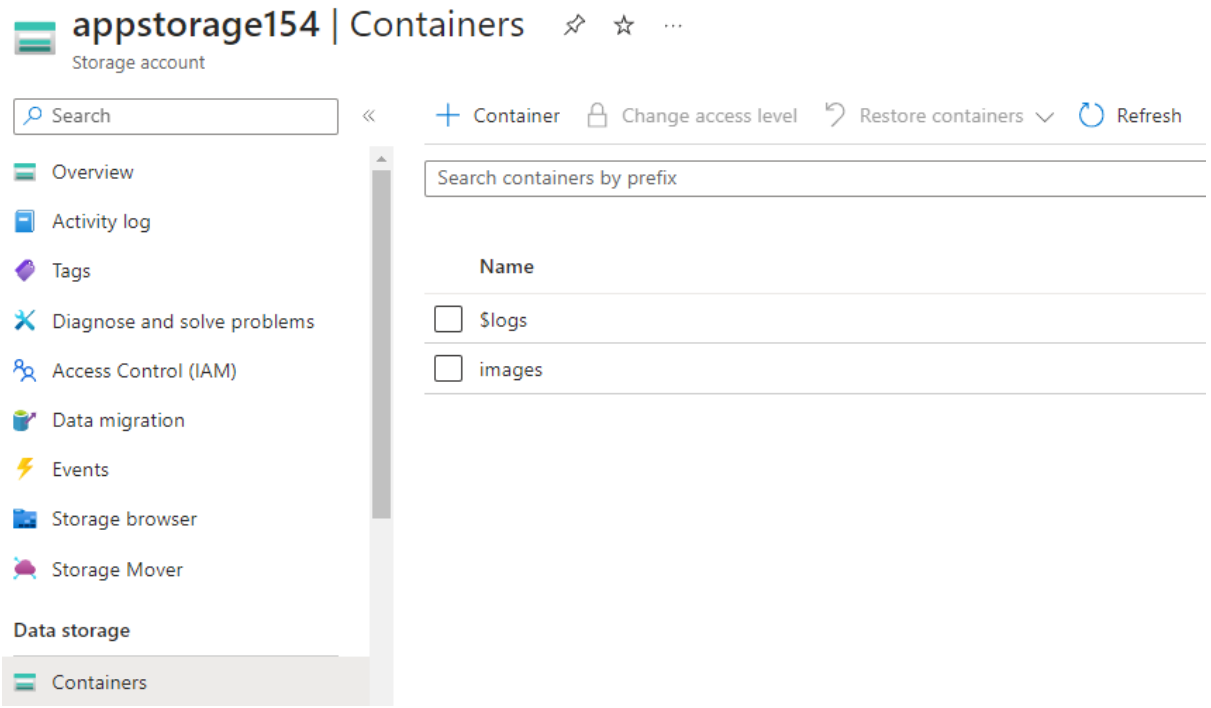
Overall, application objects in Azure AD play a crucial role in enabling secure and efficient access to applications and services, facilitating authentication, authorization, and identity management within organizations.

In this lab, we'll show you how to use Azure Active Directory's (Azure AD) application objects to let an application retrieve a blob file from an Azure Storage account.

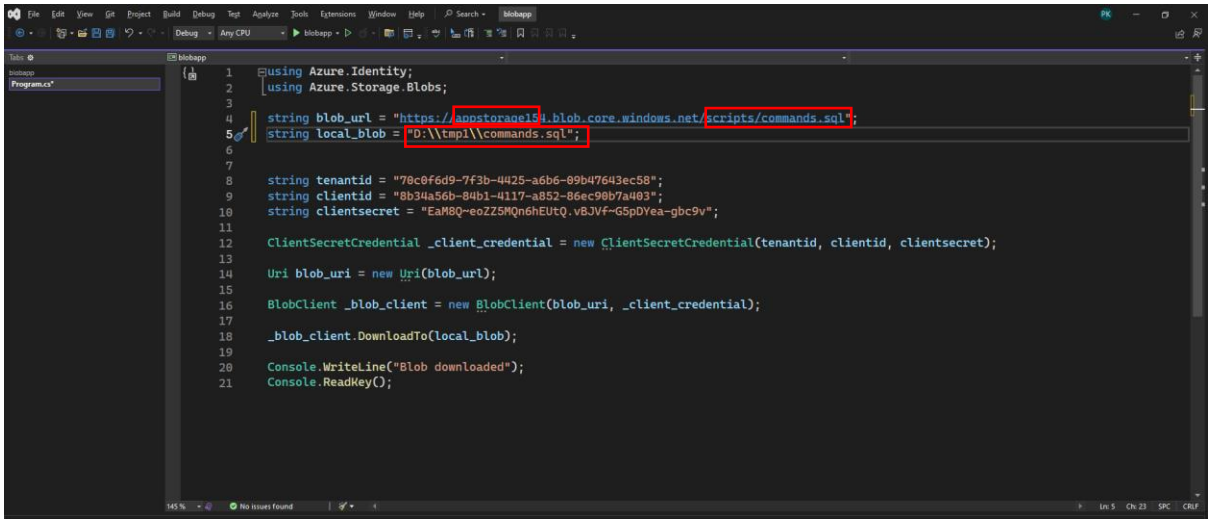
The ultimate objective is to demonstrate how application-based identity and access management in Azure AD is implemented in real-world scenarios. We make sure the application can interact with Azure resources in a regulated and safe way by registering it, giving it the necessary rights, and setting it up to access the storage account securely. This lab demonstrates how application objects provide identity and access control in Azure environments and allow safe resource access.

😄 To begin with the Lab:

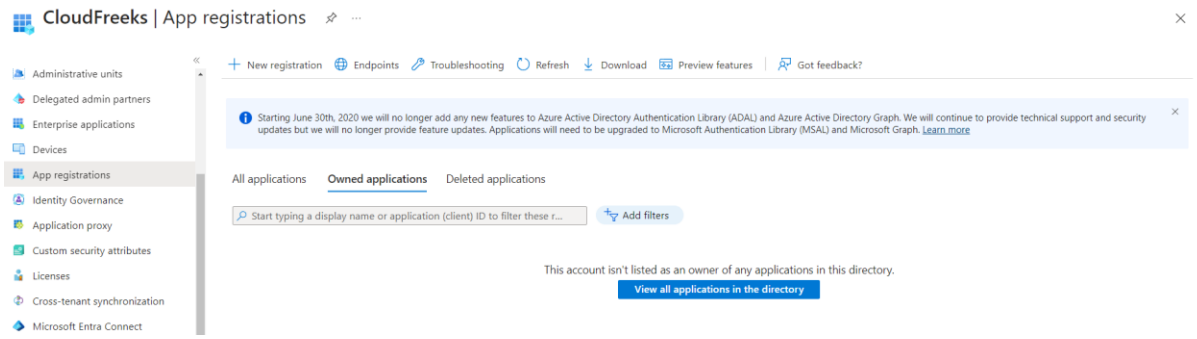
1. In this lab we are going to see an example of an application object where we are going to use an application that will be running on our laptop, and we will download a blob file from our Storage account to our laptop.
2. So, if you don't have a storage account then create a new one, then create a container and store a file in it.
3. Below you can see that we have a storage account in which we have a container and in that container, we have some files.
4. So, we will make use of our application and download a file from it.



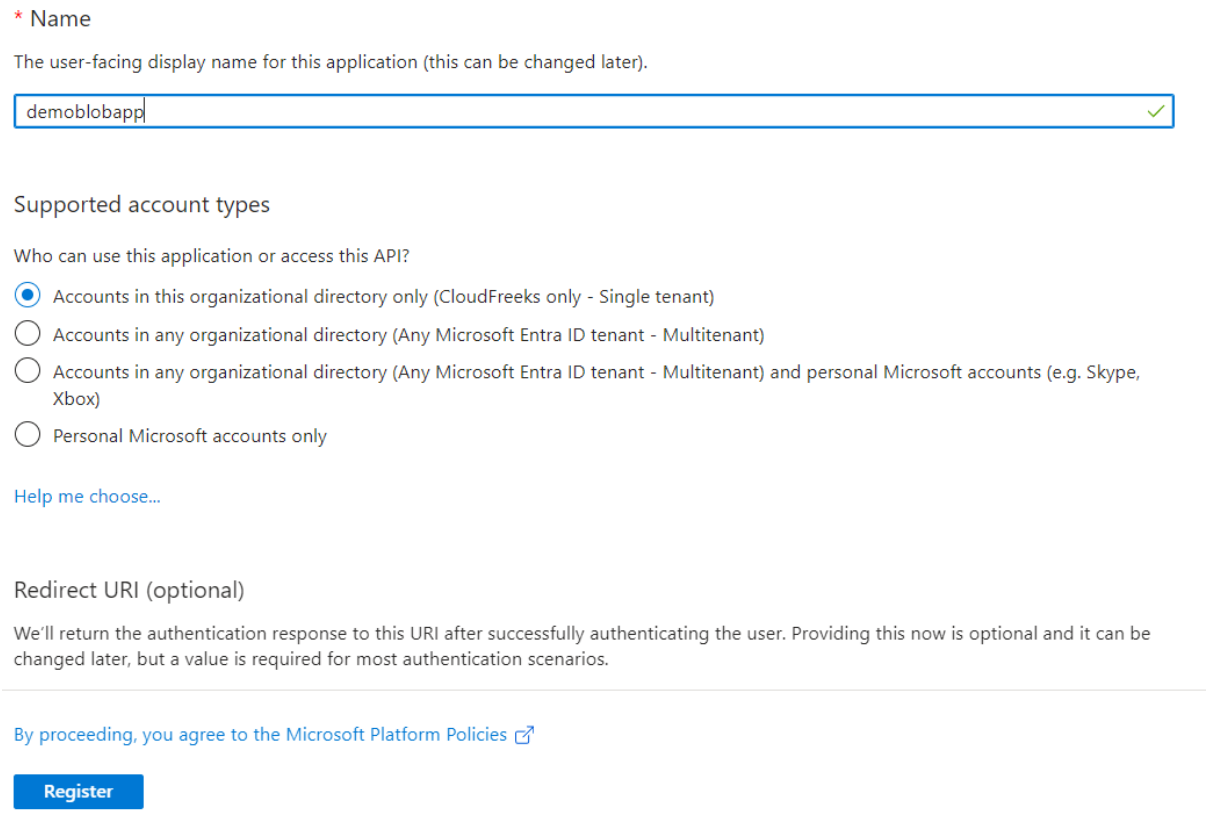
5. Now you can download the application from GitHub. After that open the application and you will see your application.
6. Now you need to change the name of your storage account, then you need to change the name of the container, give the file name, and give the location where you want to store the file in your laptop.



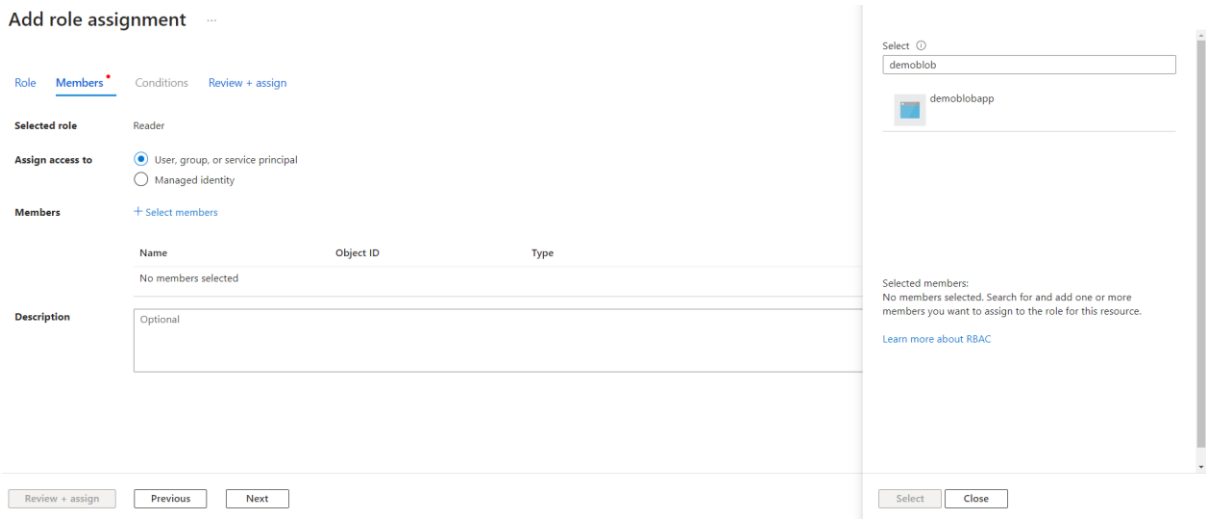
7. What we want to do is we want to create an application-based identity in the Microsoft Entra ID, give permissions for that application identity onto our Azure storage account, and that would give permissions to our program to download the blob.
8. To do this we need to navigate to Microsoft Entra ID, here from the left pane go to App Registration and choose New Registration.



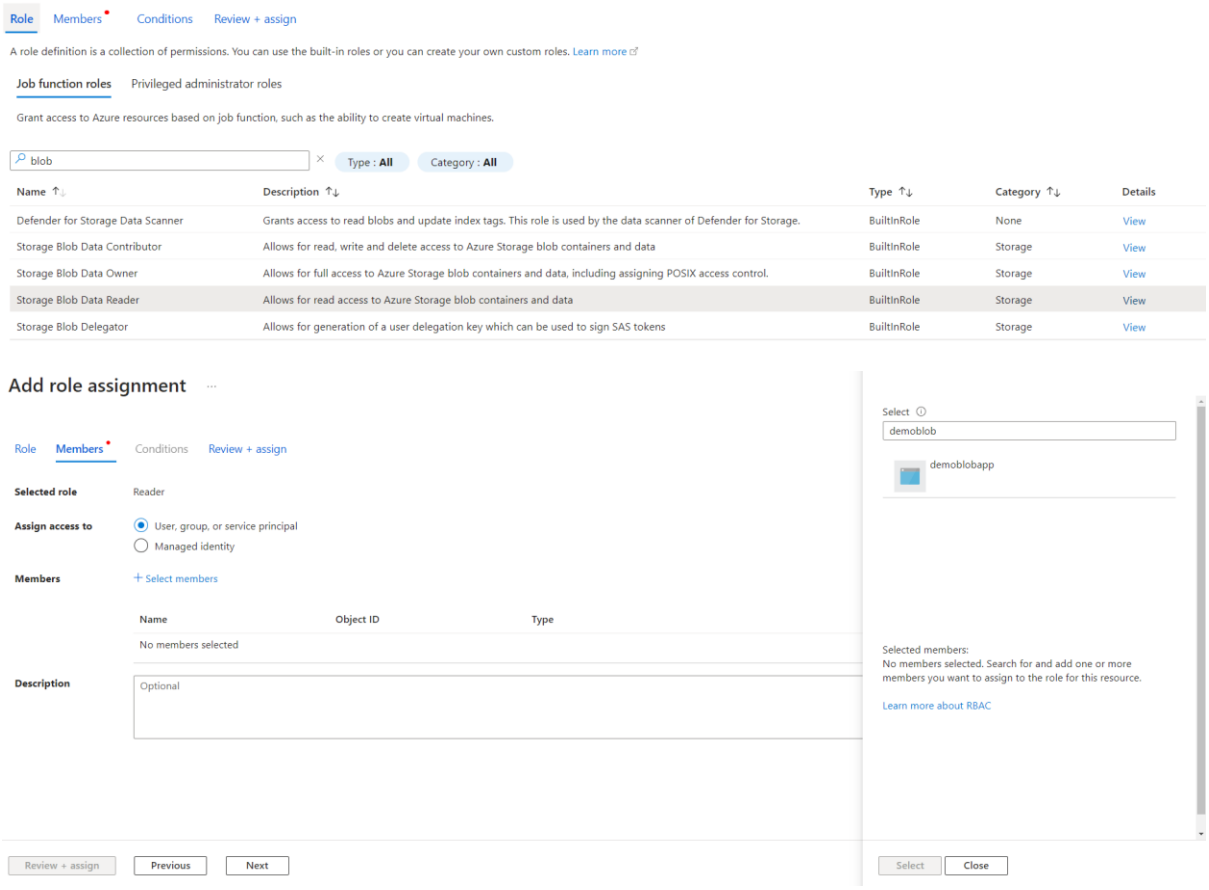
9. Now we just need to give a name and click on register.



10. We can now move on to access control (IAM) in our storage account. I can now apply role-based access control for that application object in a manner same as to adding it for a user.
11. First, we are going to add a **Reader role** and in the members section we need to add the demo blob app which we just created by app registration in Microsoft Entra ID.
12. After that just create the role assignment.



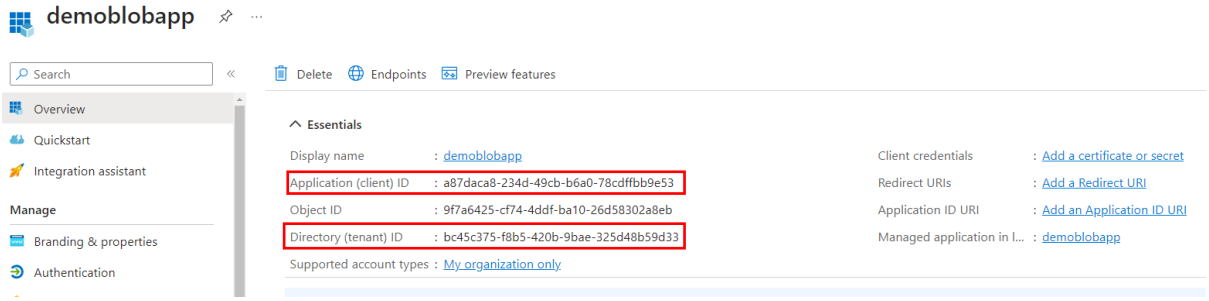
13. Now you need to add a role assignment again and choose storage blob data reader role and then add your demo blob app user, then just create your role assignment.



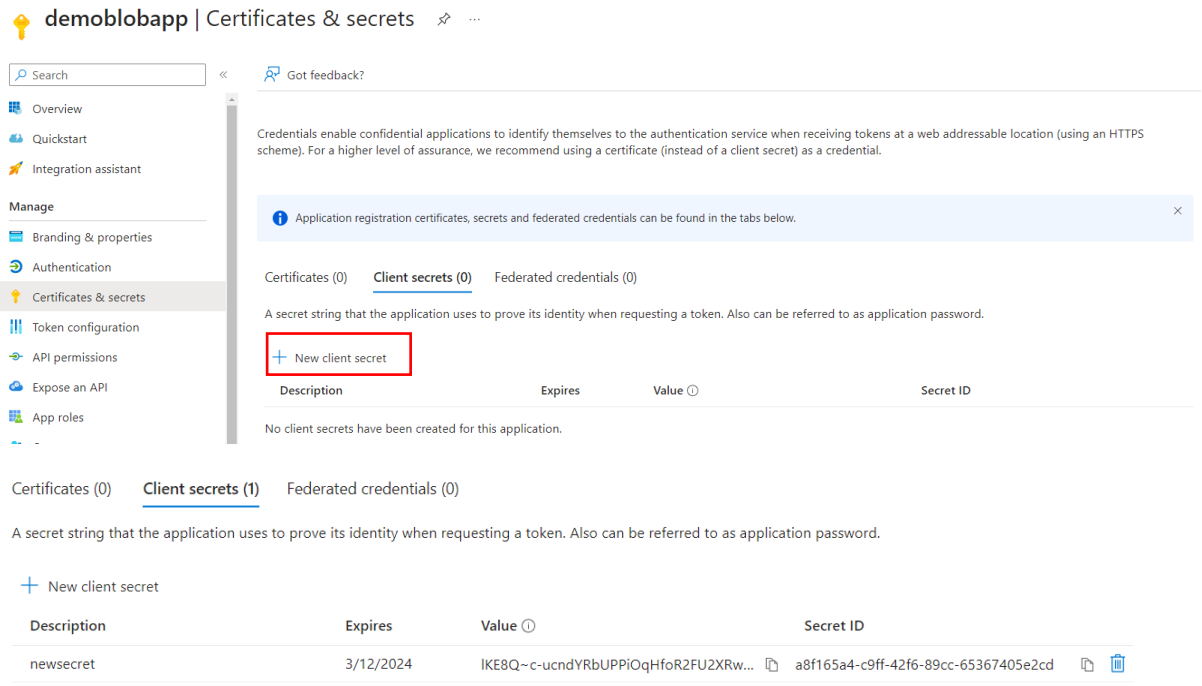
14. Now we need tenant id, client id, and client secret.

```
string tenantid = "70c0f6d9-7f3b-4425-a6b6-09b47643ec58";
string clientid = "8b34a56b-84b1-4117-a852-86ec90b7a403";
string clientsecret = "EaM8Q~eoZZ5MQn6hEUtQ.vBJVf~G5pDYea-gbc9v";
```

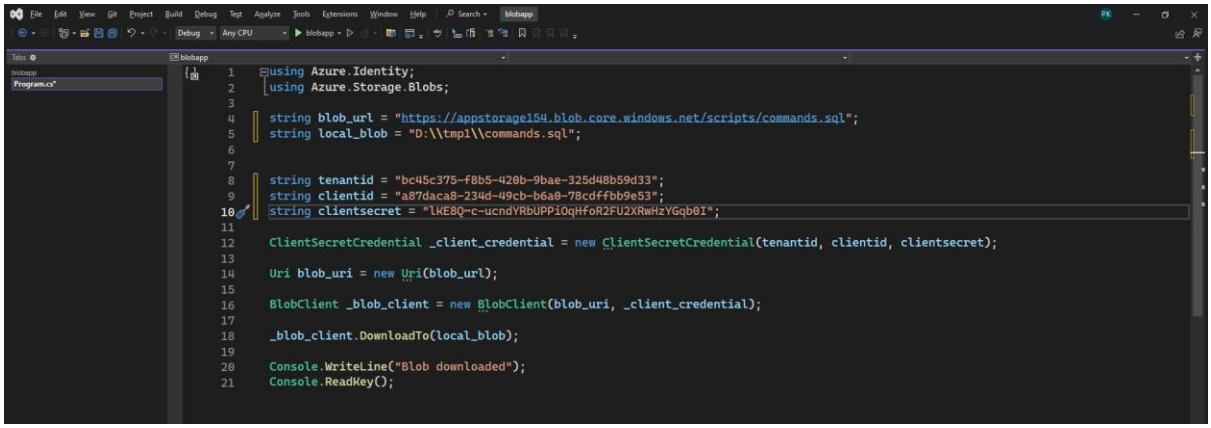
15. To get all these if you go to app registration, here you will see the tenant ID and the client ID. Now copy them and paste them into your application.
16. Now we just need client's secret, to generate that we need to go to certificate and secret in app registration.



17. Now create your client secret here.

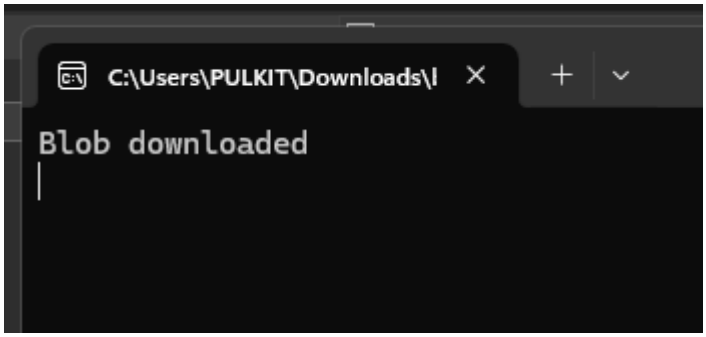


18. Once you have done everything now you just need to run the program locally.

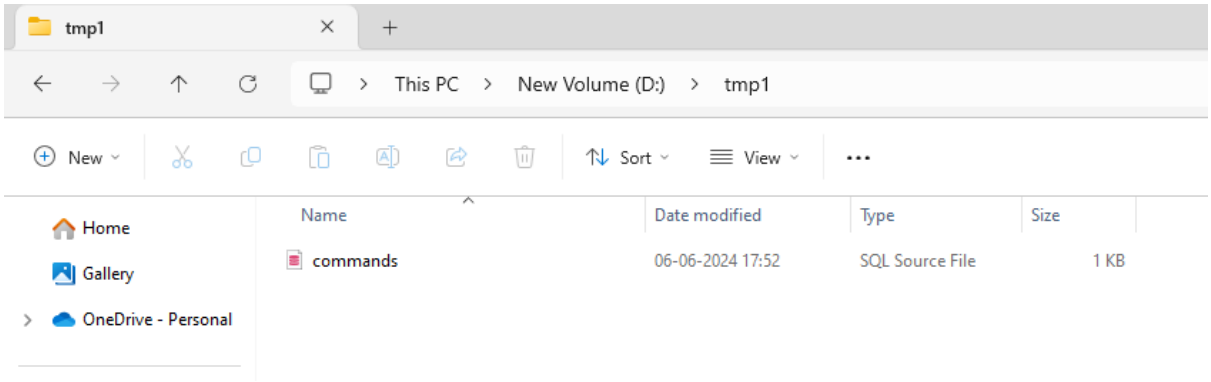


```
using Azure.Identity;
using Azure.Storage.Blobs;

string blob_url = "https://appstorage154.blob.core.windows.net/scripts/commands.sql";
string local_blob = "D:\\tmp1\\commands.sql";


string tenantid = "bc45c375-f8b5-420b-9bae-325d48b59d33";
string clientid = "a87daca8-234d-49cb-b6a0-78cdffbb9e53";
string clientsecret = "lKE8Q~c-ucndYRbUPPiOqHfoR2FU2XRwHzYGqb0I";

ClientSecretCredential _client_credential = new ClientSecretCredential(tenantid, clientid, clientsecret);

Uri blob_uri = new Uri(blob_url);

BlobClient _blob_client = new BlobClient(blob_uri, _client_credential);

_blob_client.DownloadTo(local_blob);

Console.WriteLine("Blob downloaded");
Console.ReadKey();
```

19. If your program executes successfully then you will get this message.



20. Now if you go to the location in your laptop, you will see that file has been downloaded successfully.


tmp1
This PC
New Volume (D:)
tmp1
New
Sort
View
Home
Gallery
OneDrive - Personal
Name
Date modified
Type
Size
commands
06-06-2024 17:52
SQL Source File
1 KB