



# 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.

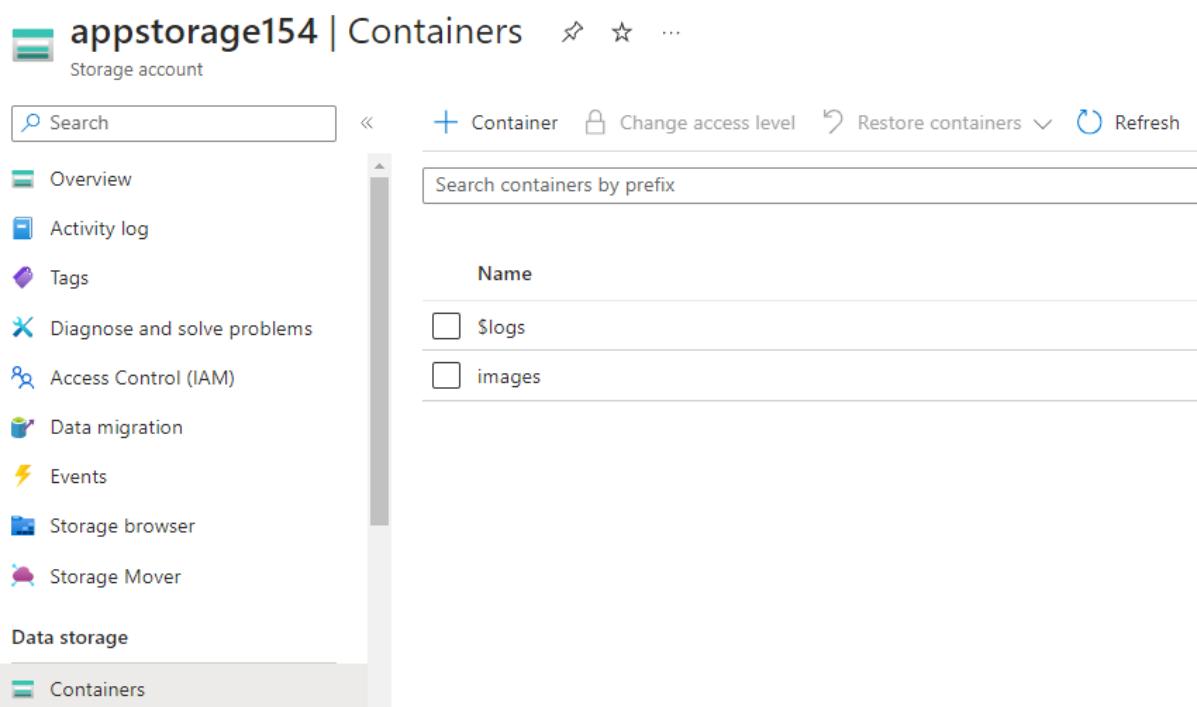
Overall, application objects in Azure AD play a crucial role in enabling secure and efficient access to applications and services, facilitating authentication, authorization, and identity management within organizations.

**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:

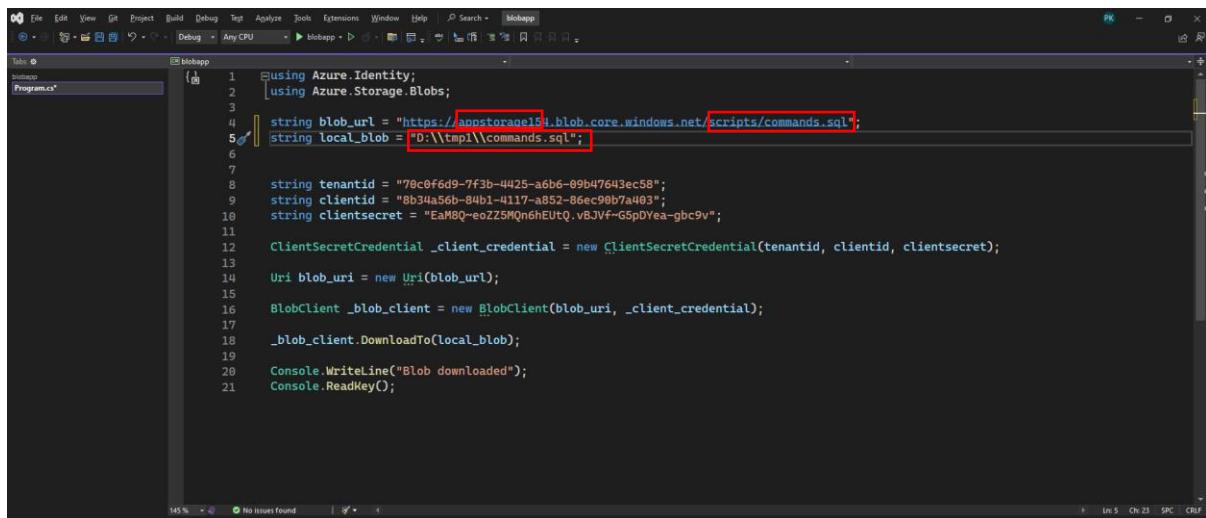
1. In this lab we are going to see an example of an application object where we are going to use an application that will be running on our laptop, and we will download a blob file from our Storage account to our laptop.
2. So, if you don't have a storage account then create a new one, then create a 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.



The screenshot shows the 'Containers' blade for the storage account 'appstorage154'. The left sidebar includes links for Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser, and Storage Mover. Under 'Data storage', the 'Containers' link is highlighted. The main area displays a table with two rows. The first row has a checkbox next to '\$logs' and the second row has a checkbox next to 'images'. A search bar at the top right says 'Search containers by prefix'.

Name
<input type="checkbox"/> \$logs
<input type="checkbox"/> images

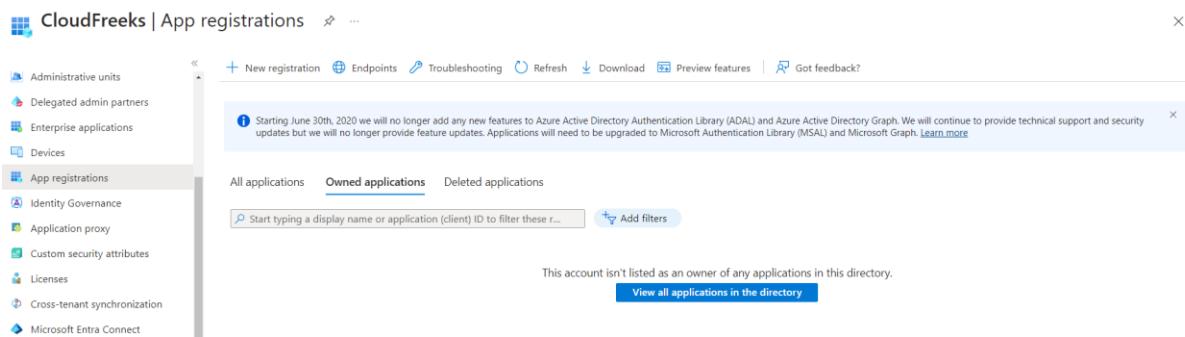
5. Now you can download the application from GitHub. After that open the application and you will see your application.
6. Now you need to change the name of your storage account, then you need to change the name of the container, give the file name, and give the location where you want to store the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



A screenshot of the Visual Studio IDE showing a C# code editor. The file is named 'blobapp' and contains a single file named 'Program.cs'. The code is a simple program that uses the Azure.Storage.Blobs library to download a blob from an Azure storage account. The URL of the blob is specified in line 4, and the local path where it will be saved is specified in line 5. Lines 4 and 5 are highlighted with red boxes.

```
1  using Azure.Identity;
2  using Azure.Storage.Blobs;
3
4  string blob_url = "https://anstorage124.blob.core.windows.net/scripts/commands.sql";
5  string local_blob = "D:\\tmp\\commands.sql";
6
7
8  string tenantid = "70c0f6d9-7f3b-4425-a6b6-09b47643ec58";
9  string clientid = "8b34a56b-84b1-4117-a852-86ec9eb7a403";
10 string clientsecret = "EaM8Q-eoZZ5MQn6hEUtQ.vB3Vf-G5pDYea-gbc9v";
11 ClientSecretCredential _client_credential = new ClientSecretCredential(tenantid, clientid, clientsecret);
12
13 Uri blob_uri = new Uri(blob_url);
14 BlobClient _blob_client = new BlobClient(blob_uri, _client_credential);
15
16 _blob_client.DownloadTo(local_blob);
17
18 Console.WriteLine("Blob downloaded");
19
20 Console.ReadKey();
```

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.



A screenshot of the Microsoft Entra ID portal under the 'App registrations' section. The left sidebar shows various administrative units and services. The 'App registrations' option is selected and highlighted. The main pane shows a list of applications. At the top, there are navigation links for 'New registration', 'Endpoints', 'Troubleshooting', 'Refresh', 'Download', 'Preview features', and 'Got feedback?'. A message at the top states: '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.' Below this, tabs for 'All applications', 'Owned applications' (which is selected), and 'Deleted applications' are shown. A search bar allows filtering by display name or application (client) ID. A note below the search bar says: 'This account isn't listed as an owner of any applications in this directory.' A blue button labeled 'View all applications in the directory' is present.

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

\* Name

The user-facing display name for this application (this can be changed later).

demoblobapp 

Supported account types

Who can use this application or access this API?

- Accounts in this organizational directory only (CloudFreaks only - Single tenant)
- Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)
- Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

By proceeding, you agree to the Microsoft Platform Policies 

[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.

Add role assignment ...

Role [Members\\*](#) Conditions Review + assign

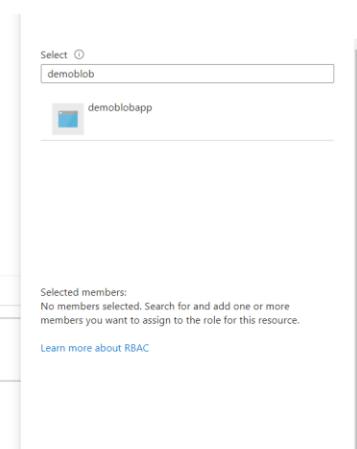
Selected role Reader

Assign access to  User, group, or service principal  Managed identity

Members [+ Select members](#)

Name	Object ID	Type
No members selected		

Description

  
Select  demoblobapp  
demoblobapp

Selected members:  
No members selected. Search for and add one or more members you want to assign to the role for this resource.  
[Learn more about RBAC](#)

[Review + assign](#) [Previous](#) [Next](#) [Select](#) [Close](#)

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.

The screenshot shows the 'Add role assignment' dialog in the Azure portal. At the top, there are tabs for 'Role', 'Members' (which is selected), 'Conditions', and 'Review + assign'. Below the tabs, a note says 'A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles.' Under 'Job function roles', it says 'Privileged administrator roles' and 'Grant access to Azure resources based on job function, such as the ability to create virtual machines.' A search bar at the top right contains 'blob'. The main table lists roles with columns for Name, Description, Type, Category, and Details. The 'Storage Blob Data Reader' role is highlighted. On the right, a search results pane shows 'demloblob' and 'demloblobapp'.

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.

The screenshot shows the 'demloblobapp' app registration page in the Azure portal. The left sidebar has sections for Overview, Quickstart, Integration assistant, Branding & properties, and Authentication. The main area shows the 'Essentials' section with fields for Display name (set to 'demloblobapp'), Application (client) ID (set to 'a87daca8-234d-49cb-b6a0-78cdffbb9e53'), Object ID (set to '9f7a6425-cf74-4ddf-ba10-26d58302a8eb'), and Directory (tenant) ID (set to 'bc45c375-f8b5-420b-9bae-325d48b59d33'). Other fields include Client credentials, Redirect URIs, Application ID URI, and Managed application in I... . The supported account types are set to 'My organization only'.

17. Now create your client secret here.

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.

**Certificates (0) Client secrets (0) Federated credentials (0)**

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

**+ New client secret**

Description	Expires	Value ⓘ	Secret ID
newsecret	3/12/2024	IKE8Q~c-ucndYRbUPPiOqHfoR2FU2XRw...	a8f165a4-c9ff-42f6-89cc-65367405e2cd

No client secrets have been created for this application.

**Certificates (0) Client secrets (1) Federated credentials (0)**

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

**+ New client secret**

Description	Expires	Value ⓘ	Secret ID
newsecret	3/12/2024	IKE8Q~c-ucndYRbUPPiOqHfoR2FU2XRw...	a8f165a4-c9ff-42f6-89cc-65367405e2cd

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

```

1  using Azure.Identity;
2  using Azure.Storage.Blobs;
3
4  string blob_url = "https://appstorage154.blob.core.windows.net/scripts/commands.sql";
5  string local_blob = "D:\tmp1\commands.sql";
6
7
8  string tenantid = "bc45c375-f0b5-420b-9bae-325d48b59d33";
9  string clientid = "a87daca8-234d-49cb-b6a0-78cfffb9e53";
10 string clientsecret = "lKE8Q~c-ucndYRbUPPiOqHfoR2FU2XRwHzYEqb0I";
11
12 ClientSecretCredential _client_credential = new ClientSecretCredential(tenantid, clientid, clientsecret);
13
14 Uri blob_uri = new Uri(blob_url);
15
16 BlobClient _blob_client = new BlobClient(blob_uri, _client_credential);
17
18 _blob_client.DownloadTo(local_blob);
19
20 Console.WriteLine("Blob downloaded");
21 Console.ReadKey();

```

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

```
Blob downloaded
```

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

