

1. Create or use a windows VM
2. Create or use a storage account
3. Copy Lab2/main.py to the target VM

Azure Blob Storage File Upload using Python

1. Initial Setup:
 - a. Python Environment: Set up a new Python project.
 - b. Tip: Consider using virtual environments (e.g., venv or virtualenv) to manage dependencies.
2. Package Installation:
 - a. Ensure the following packages are installed:
 - i. azure-identity: For Azure SDK authentication.
 1. Install with: pip install azure-identity
 - ii. azure-storage-blob: SDK for Azure Blob Storage operations.
 1. Install with: pip install azure-storage-blob
3. Code Setup:
 - a. Refer to the Lab2/main.py directory and integrate the provided Python code into your project.

Enable system-assigned managed identity on an existing VM

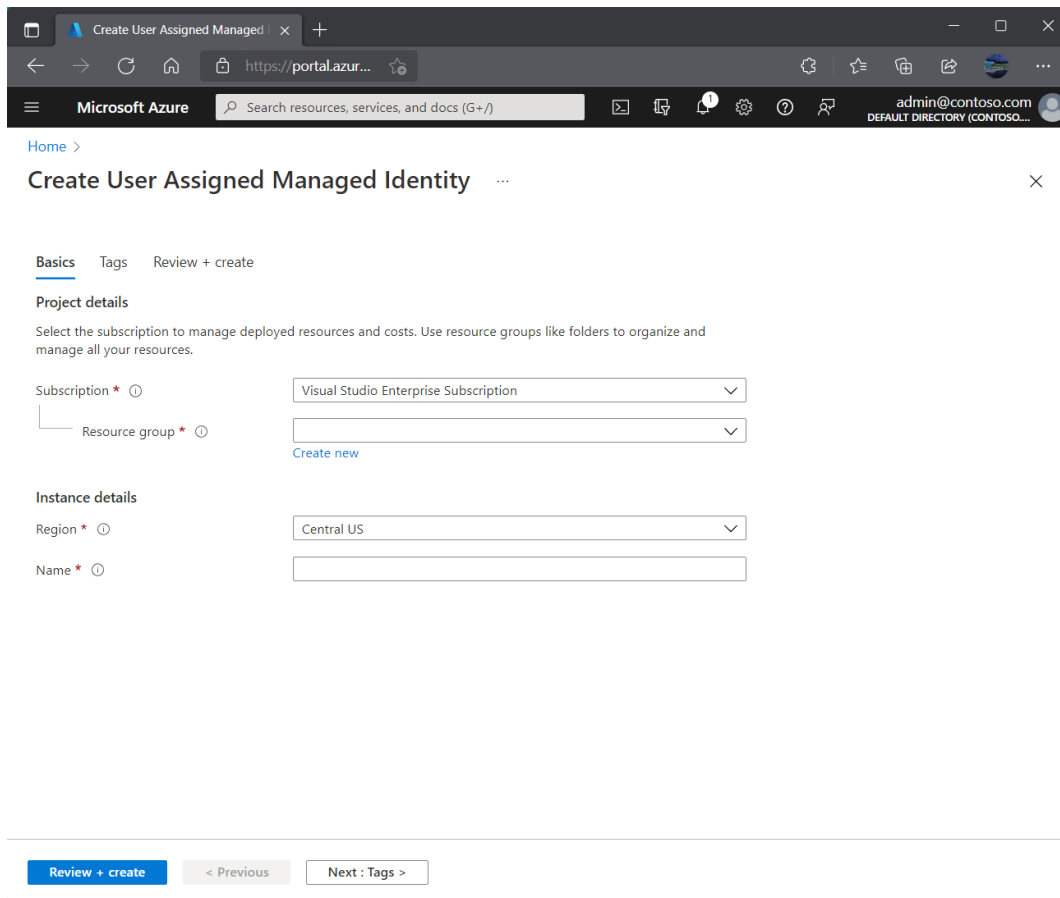
1. Sign in to the Azure portal using an account associated with the Azure subscription that contains the VM.
2. Navigate to the desired Virtual Machine and select Identity.
3. Under System assigned, Status, select On and then click Save
4. Give permissions to the new identity to write in the blob
5. Run the code

Remove system-assigned managed identity from a VM

1. Sign in to the Azure portal using an account associated with the Azure subscription that contains the VM.
2. Navigate to the desired Virtual Machine and select Identity.
3. Under System assigned, Status, select Off and then click Save:

Create a user-assigned managed identity

1. Sign in to the Azure portal.
2. In the search box, enter Managed Identities. Under Services, select Managed Identities.
3. Select Add, and enter values in the following boxes in the Create User Assigned Managed Identity pane:
 - a. Subscription: Choose the subscription to create the user-assigned managed identity under.
 - b. Resource group: Choose a resource group to create the user-assigned managed identity in, or select Create new to create a new resource group.
 - c. Region: Choose a region to deploy the user-assigned managed identity, for example, West US.
 - d. Name: Enter the name for your user-assigned managed identity, for example, UAI1.

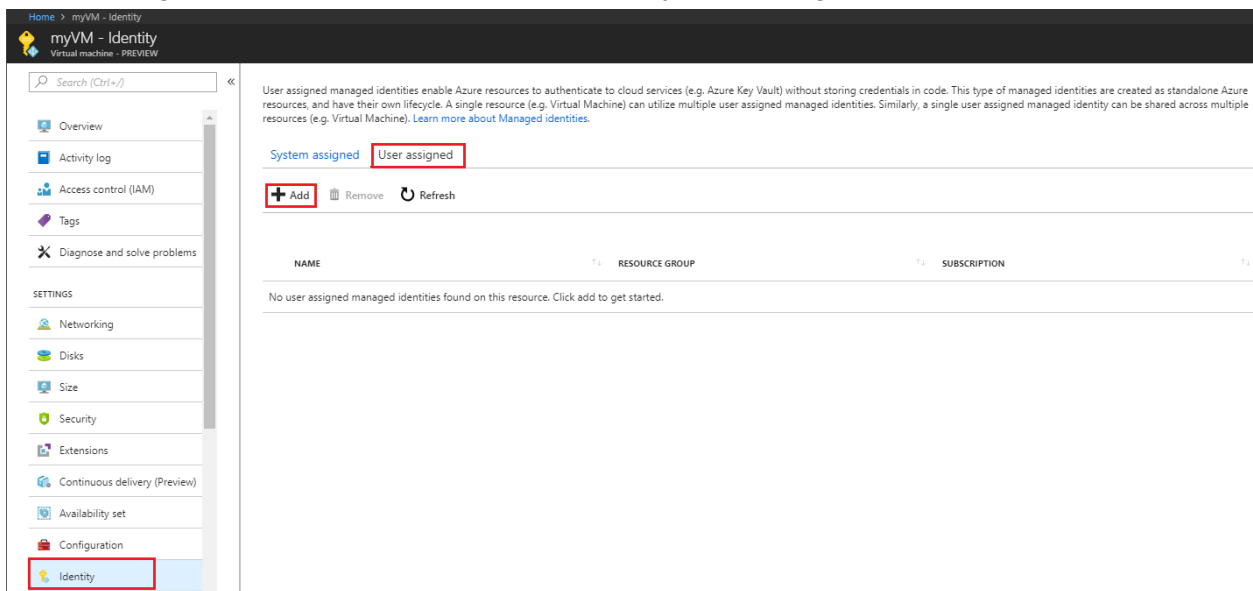


The screenshot shows the 'Create User Assigned Managed Identity' form in the Azure portal. The browser address bar shows 'https://portal.azure...'. The user is logged in as 'admin@contoso.com'. The form has three tabs: 'Basics', 'Tags', and 'Review + create'. The 'Basics' tab is active. Under 'Project details', there is a description: 'Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.' The 'Subscription' dropdown is set to 'Visual Studio Enterprise Subscription'. The 'Resource group' dropdown is empty, with a 'Create new' link below it. Under 'Instance details', the 'Region' dropdown is set to 'Central US'. The 'Name' field is empty. At the bottom, there are three buttons: 'Review + create' (blue), '< Previous' (grey), and 'Next : Tags >' (grey).

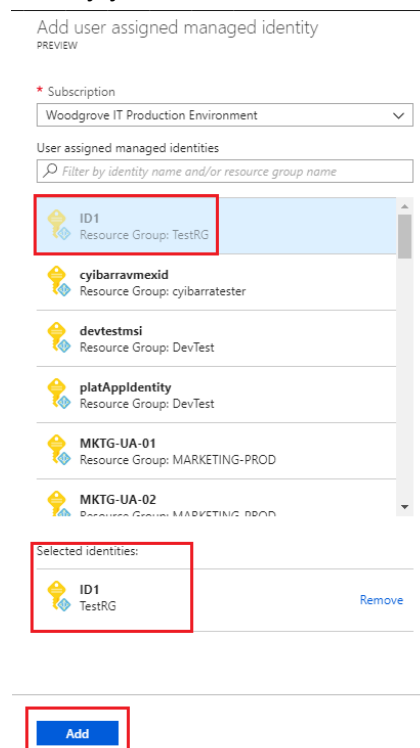
4. Select Review + create to review the changes.
5. Select Create.
6. Give it the appropriate permissions

Assign a user-assigned managed identity to an existing VM

1. Sign in to the Azure portal using an account associated with the Azure subscription that contains the VM.
2. Navigate to the desired VM and click Identity, User assigned and then +Add.



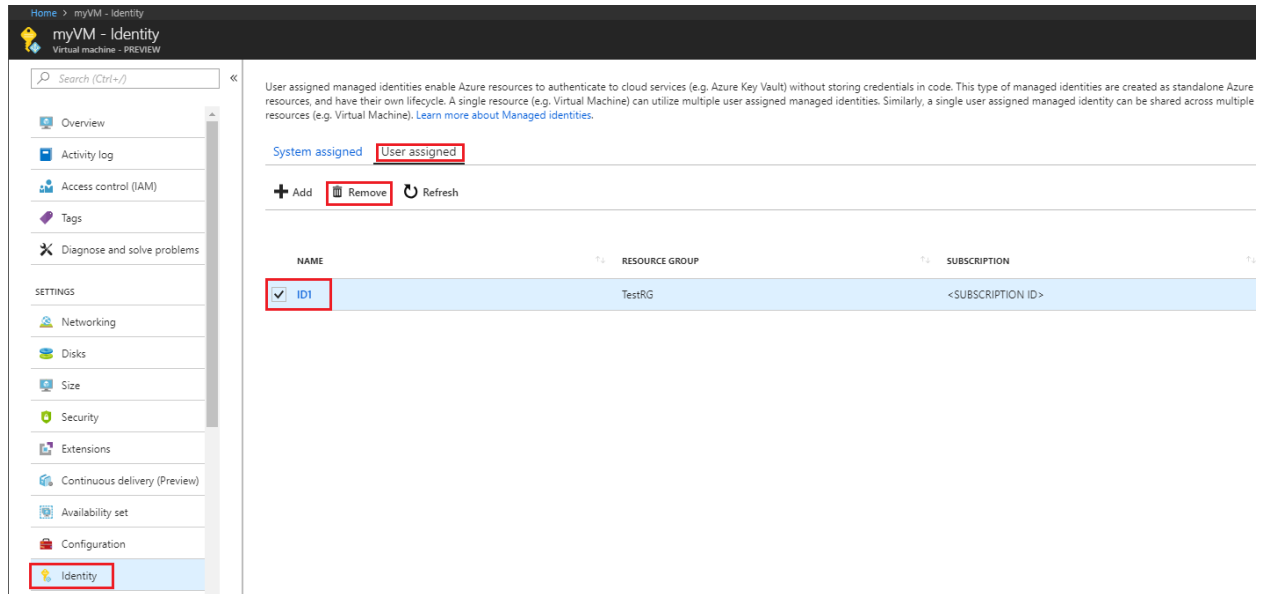
3. Click the user-assigned identity you want to add to the VM and then click Add.



4. Run the code again

Remove a user-assigned managed identity from a VM

1. Sign in to the Azure portal using an account associated with the Azure subscription that contains the VM.
2. Navigate to the desired VM and select Identity, User assigned, the name of the user-assigned managed identity you want to delete and then click Remove (click Yes in the confirmation pane).



Use service principal for authentication

Register an Application with Azure AD:

1. Go to the Azure portal.
2. Search for and select Azure Active Directory.
3. Go to "App registrations" and select "New registration".
4. Provide a name for the application, specify the supported account types, and the redirect URI (optional), then click "Register".

Create a Service Principal:

1. Once the application is registered, Azure AD will automatically create a service principal for it.

2. Navigate to the registered application in Azure AD.
3. Under "Manage," click on "Certificates & secrets".
4. Click on "New client secret," provide a description and an expiry period for the secret, and then click "Add".
5. Copy the values of the Application (client) ID, Directory (tenant) ID, and the newly generated secret, as you'll need them later.
6. Setup env variables inside the VM (AZURE_CLIENT_ID, AZURE_TENANT_ID, AZURE_CLIENT_SECRET).

Assign a Role to the Service Principal:

1. In the Azure portal, go to the subscription or resource group where the storage account is located .
2. Select "Access control (IAM)".
3. Click on "Add role assignment".
4. Select the role you want to assign to the service principal.
5. Search for the name of the app you registered and select it.
6. Click "Save" to assign the role.
7. Run the code

Register a new application

1. Sign in to the Azure portal, search for and select App Services, and then select your app. Note your app's URL. You'll use it to configure your Microsoft Entra app registration.
2. Navigate to your tenant in the portal:
3. On the "Overview" screen, make note of the Tenant ID, as well as the Primary domain.
4. From the left navigation, select App registrations > New registration.
5. In the Register an application page, enter a Name for your app registration
6. In Supported account types, select the account type that can access this application.
7. In the Redirect URIs section, select Web for platform and type `<app-url>.auth/login/aad/callback`. For example, `https://contoso.azurewebsites.net/.auth/login/aad/callback`.
8. Select Register.
9. After the app registration is created, copy the Application (client) ID and the Directory (tenant) ID for later.
10. Under Implicit grant and hybrid flows, enable ID tokens to allow OpenID Connect user sign-ins from App Service. Select Save.

11. (Optional) From the left navigation, select Branding & properties. In Home page URL, enter the URL of your App Service app and select Save.
12. From the left navigation, select Expose an API > Set > Save. This value uniquely identifies the application when it's used as a resource, allowing tokens to be requested that grant access. It's used as a prefix for scopes you create.

For a single-tenant app, you can use the default value, which is in the form `api://<application-client-id>`. You can also specify a more readable URI like `https://contoso.com/api` based on one of the verified domains for your tenant. For a multi-tenant app, you must provide a custom URI. To learn more about accepted formats for App ID URIs, see the app registrations best practices reference.

13. Select Add a scope.
 - a. In Scope name, enter `user_impersonation`.
 - b. In Who can consent, select Admins and users if you want to allow users to consent to this scope.
 - c. In the text boxes, enter the consent scope name and description you want users to see on the consent page. For example, enter `Access <application-name>`.
 - d. Select Add scope.
14. (Optional) To create a client secret:
 - a. From the left navigation, select Certificates & secrets > Client secrets > New client secret.
 - b. Enter a description and expiration and select Add.
 - c. In the Value field, copy the client secret value. It won't be shown again once you navigate away from this page.
15. (Optional) To add multiple Reply URLs, select Authentication.
 - a. Finish setting up your app registration:

Authentication setting: confidential vs. public

Select Authentication to review the settings. The default value for Allow public client flows is "No".

If you keep this default value, the application registration is a confidential client application and a certificate or secret is required.

Home > myhealthapisapp1

myhealthapisapp1 | Authentication ✕ ...

Search (Ctrl+/) < Save Discard Got feedback?

Overview
Quickstart
Integration assistant

Manage

Branding

Authentication

Certificates & secrets

Token configuration

API permissions

Expose an API

App roles

Owners

Roles and administrators | Preview

Manifest

Support + Troubleshooting

Troubleshooting

New support request

Platform configurations

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

+ Add a platform

Supported account types

Who can use this application or access this API?

☒ Accounts in this organizational directory only (b2xadorg only - Single tenant)

☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)

[Help me decide...](#)

Advanced settings

Allow public client flows ⓘ

Enable the following mobile and desktop flows:

Yes No

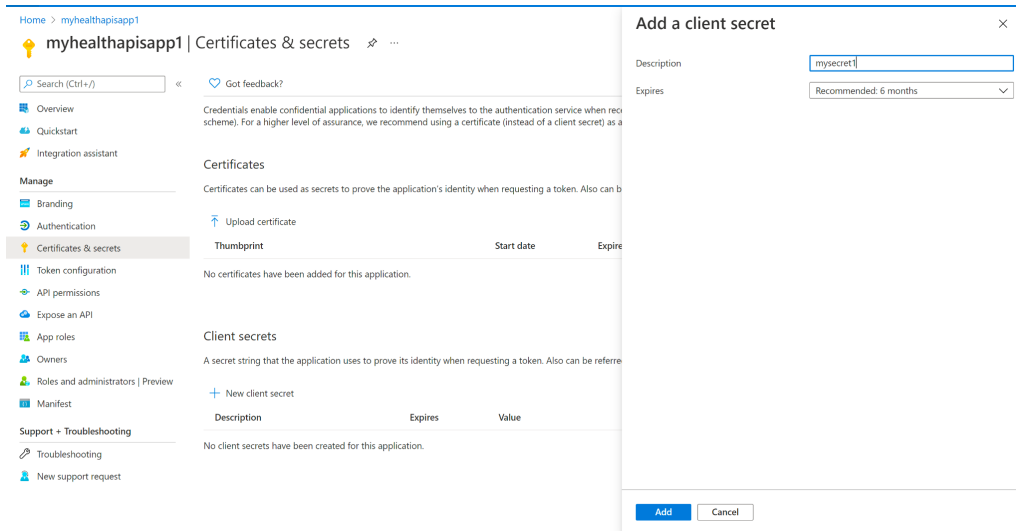
- App collects plaintext password (Resource Owner Password Credential Flow) [Learn more](#)
- No keyboard (Device Code Flow) [Learn more](#)
- SSO for domain-joined Windows (Windows Integrated Auth Flow) [Learn more](#)

If you change the default value to "Yes" for the "Allow public client flows" option in the advanced setting, the application registration is a public client application and a certificate or secret isn't required. The "Yes" value is useful when you want to use the client application in your mobile app or a JavaScript app where you don't want to store any secrets.

Certificates & secrets

Select Certificates & Secrets and select New Client Secret. Select Recommended 6 months in the Expires field. This new secret will be valid for six months. You can also choose different values such as:

- 03 months
- 12 months
- 24 months
- Custom start date and end date.

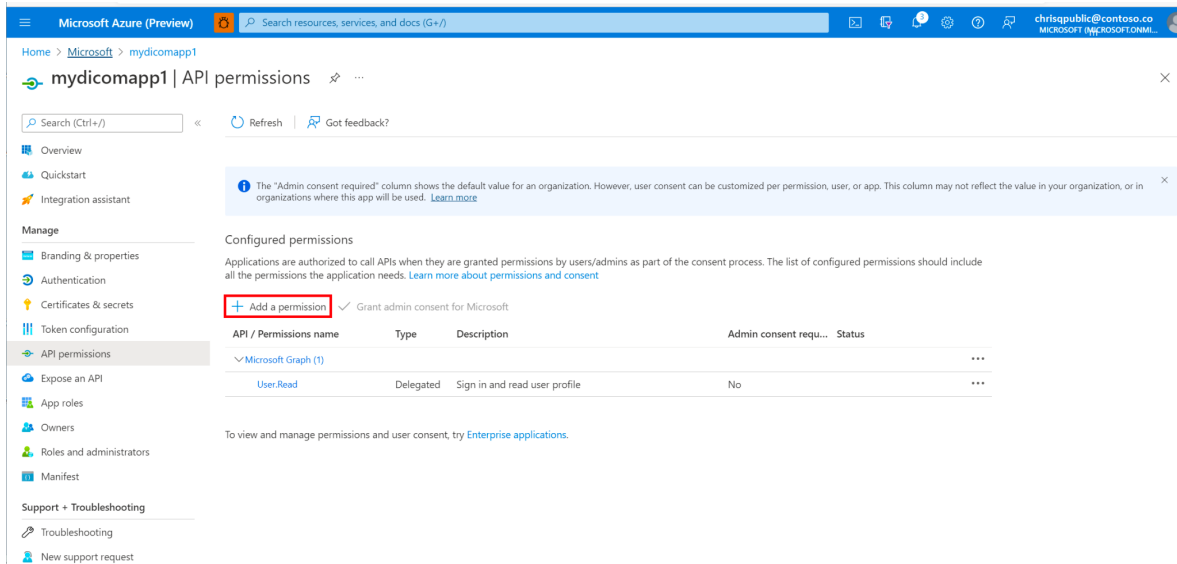


Optionally, you can upload a certificate (public key) and use the Certificate ID, a GUID value associated with the certificate. For testing purposes, you can create a self-signed certificate using tools such as the PowerShell command line, `New-SelfSignedCertificate`, and then export the certificate from the certificate stor

API permissions

The following steps are required for the DICOM service, but optional for the FHIR service. In addition, user access permissions or role assignments for the Azure Health Data Services are managed through RBAC. For more details, visit [Configure Azure RBAC for Azure Health Data Services](#).

1. Select the API permissions blade.
2. Select Add a permission.



Enable Microsoft Entra ID in your App Service app

1. Sign in to the Azure portal and navigate to your app.
2. From the left navigation, select Authentication > Add identity provider > Microsoft.
3. Select the Tenant type of the app registration you created.
4. Configure the app to use the registration you created, using the instructions for the appropriate tenant type:
5. For App registration type, choose one of the following:
 - a. Pick an existing app registration in this directory: Choose an app registration from the current tenant and automatically gather the necessary app information. The system will attempt to create a new client secret against the app registration and automatically configure your app to use it. A default issuer URL is set based on the supported account types configured in the app registration. If you intend to change this default, consult the following table.
 - b. Provide the details of an existing app registration: Specify details for an app registration from another tenant or if your account doesn't have permission in the current tenant to query the registrations. For this option, you must manually fill in the configuration values according to the following table.
6. When filling in the configuration details directly, use the values you collected during the app registration creation process

Field	Description
Application (client) ID	Use the Application (client) ID of the app registration.
Client Secret	Use the client secret you generated in the app registration. With a client secret, hybrid flow is used and the App Service will return access and refresh tokens. When the client secret isn't set, implicit flow is used and only an ID token is returned. These tokens are sent by the provider and stored in the App Service authentication token store.
Issuer URL	<p>Use <code><authentication-endpoint>/<tenant-id>/v2.0</code>, and replace <code><authentication-endpoint></code> with the authentication endpoint you determined in the previous step for your tenant type and cloud environment, also replacing <code><tenant-id></code> with the Directory (tenant) ID in which the app registration was created. For applications that use Azure AD v1, omit <code>/v2.0</code> in the URL.</p> <p>This value is used to redirect users to the correct Microsoft Entra tenant, as well as to download the appropriate metadata to determine the appropriate token signing keys and token issuer claim value for example. Any configuration other than a tenant-specific endpoint will be treated as multi-tenant. In multi-tenant configurations, no validation of the issuer or tenant ID is performed by the system, and these checks should be fully handled in your app's authorization logic.</p>
Allowed Token Audiences	This field is optional. The configured Application (client) ID is <i>always</i> implicitly considered to be an allowed audience. If your application represents an API that will be called by other clients, you should also add the Application ID URI that you configured on the app registration. There's a limit of 500 characters total across the list of allowed audiences.