100 XP

Introduction

3 minutes

A common challenge for developers is the management of secrets and credentials used to secure communication between different components making up a solution. Managed identities eliminate the need for developers to manage credentials.

After completing this module, you'll be able to:

- Explain the differences between the two types of managed identities
- Describe the flows for user- and system-assigned managed identities
- Configure managed identities
- Acquire access tokens by using REST and code

Next unit: Explore managed identities

Continue

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Explore managed identities

3 minutes

Managed identities provide an identity for applications to use when connecting to resources that support Azure Active Directory (Azure AD) authentication. Applications may use the managed identity to obtain Azure AD tokens. For example, an application may use a managed identity to access resources like Azure Key Vault where developers can store credentials in a secure manner or to access storage accounts.

Types of managed identities

There are two types of managed identities:

- A **system-assigned managed identity** is enabled directly on an Azure service instance. When the identity is enabled, Azure creates an identity for the instance in the Azure AD tenant that's trusted by the subscription of the instance. After the identity is created, the credentials are provisioned onto the instance. The lifecycle of a system-assigned identity is directly tied to the Azure service instance that it's enabled on. If the instance is deleted, Azure automatically cleans up the credentials and the identity in Azure AD.
- A **user-assigned managed identity** is created as a standalone Azure resource. Through a create process, Azure creates an identity in the Azure AD tenant that's trusted by the subscription in use. After the identity is created, the identity can be assigned to one or more Azure service instances. The lifecycle of a user-assigned identity is managed separately from the lifecycle of the Azure service instances to which it's assigned.

Internally, managed identities are service principals of a special type, which are locked to only be used with Azure resources. When the managed identity is deleted, the corresponding service principal is automatically removed.

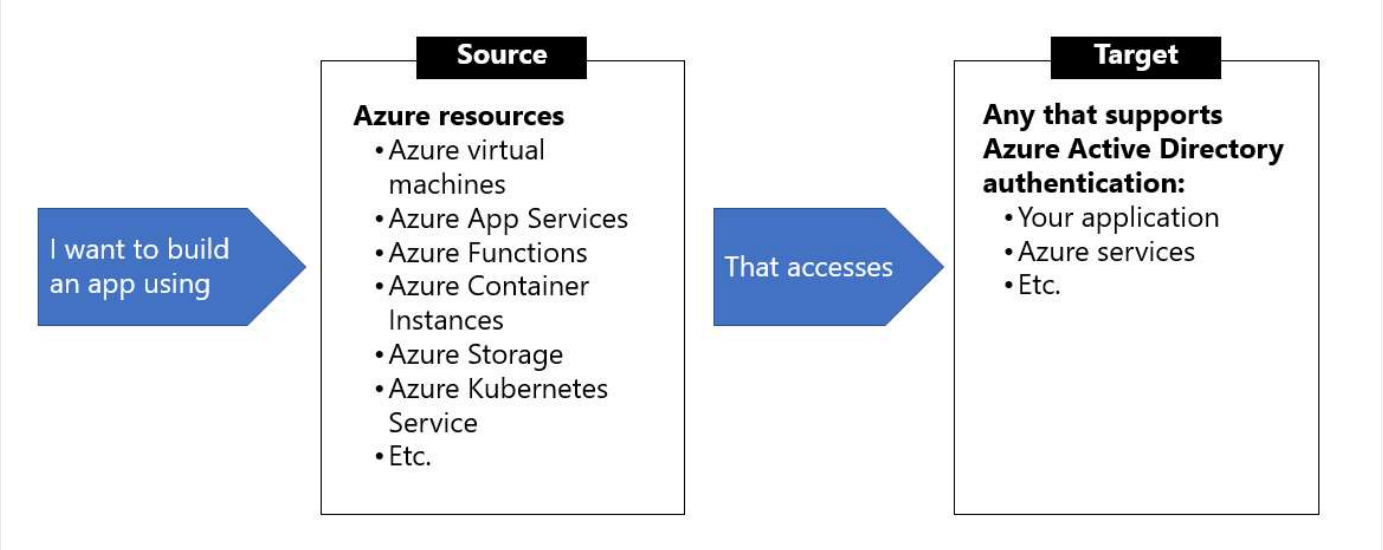
Characteristics of managed identities

The table below highlights some of the key differences between the two types of managed identities.

Characteristic	System-assigned managed identity	User-assigned managed identity
Creation	Created as part of an Azure resource (for example, an Azure virtual machine or Azure App Service)	Created as a stand-alone Azure resource
Lifecycle	Shared lifecycle with the Azure resource that the managed identity is created with. When the parent resource is deleted, the managed identity is deleted as well.	Independent life-cycle. Must be explicitly deleted.
Sharing across Azure resources	Cannot be shared, it can only be associated with a single Azure resource.	Can be shared, the same user-assigned managed identity can be associated with more than one Azure resource.

When to use managed identities

The image below gives an overview the scenarios that support using managed identities. For example, you can use managed identities if you want to build an app using Azure App Services that accesses Azure Storage without having to manage any credentials.



What Azure services support managed identities?

Managed identities for Azure resources can be used to authenticate to services that support Azure Active Directory authentication. For a list of Azure services that support the managed identities for Azure resources feature, visit Services that support managed identities for Azure resources.

The rest of this module will use Azure virtual machines in the examples, but the same concepts and similar actions can be applied to any resource in Azure that supports Azure Active Directory authentication.

Next unit: Discover the managed identities authentication flow

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Discover the managed identities authentication flow

3 minutes

In this unit, you learn how managed identities work with Azure virtual machines. Below are the flows detailing how the two types of managed identities work with with an Azure virtual machine.

How a system-assigned managed identity works with an Azure virtual machine

1. Azure Resource Manager receives a request to enable the system-assigned managed identity on a virtual machine.
2. Azure Resource Manager creates a service principal in Azure Active Directory for the identity of the virtual machine. The service principal is created in the Azure Active Directory tenant that's trusted by the subscription.
3. Azure Resource Manager configures the identity on the virtual machine by updating the Azure Instance Metadata Service identity endpoint with the service principal client ID and certificate.
4. After the virtual machine has an identity, use the service principal information to grant the virtual machine access to Azure resources. To call Azure Resource Manager, use role-based access control in Azure Active Directory to assign the appropriate role to the virtual machine service principal. To call Key Vault, grant your code access to the specific secret or key in Key Vault.
5. Your code that's running on the virtual machine can request a token from the Azure Instance Metadata service endpoint, accessible only from within the virtual machine:
   `http://169.254.169.254/metadata/identity/oauth2/token`
6. A call is made to Azure Active Directory to request an access token (as specified in step 5) by using the client ID and certificate configured in step 3. Azure Active Directory returns a JSON

Web Token (JWT) access token.

7. Your code sends the access token on a call to a service that supports Azure Active Directory authentication.

How a user-assigned managed identity works with an Azure virtual machine

1. Azure Resource Manager receives a request to create a user-assigned managed identity.
2. Azure Resource Manager creates a service principal in Azure Active Directory for the user-assigned managed identity. The service principal is created in the Azure Active Directory tenant that's trusted by the subscription.
3. Azure Resource Manager receives a request to configure the user-assigned managed identity on a virtual machine and updates the Azure Instance Metadata Service identity endpoint with the user-assigned managed identity service principal client ID and certificate.
4. After the user-assigned managed identity is created, use the service principal information to grant the identity access to Azure resources. To call Azure Resource Manager, use role-based access control in Azure Active Directory to assign the appropriate role to the service principal of the user-assigned identity. To call Key Vault, grant your code access to the specific secret or key in Key Vault.

   > **Note**
   >
   > You can also do this step before step 3.

5. Your code that's running on the virtual machine can request a token from the Azure Instance Metadata Service identity endpoint, accessible only from within the virtual machine:
   `http://169.254.169.254/metadata/identity/oauth2/token`
6. A call is made to Azure Active Directory to request an access token (as specified in step 5) by using the client ID and certificate configured in step 3. Azure Active Directory returns a JSON Web Token (JWT) access token.
7. Your code sends the access token on a call to a service that supports Azure Active Directory authentication.

Next unit: Configure managed identities

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Configure managed identities

3 minutes

You can configure an Azure virtual machine with a managed identity during, or after, the creation of the virtual machine. Below are some CLI examples showing the commands for both system- and user-assigned identities.

System-assigned managed identity

To create, or enable, an Azure virtual machine with the system-assigned managed identity your account needs the **Virtual Machine Contributor** role assignment. No additional Azure AD directory role assignments are required.

Enable system-assigned managed identity during creation of an Azure virtual machine

The following example creates a virtual machine named *myVM* with a system-assigned managed identity, as requested by the `--assign-identity` parameter, with the specified `--role` and `--scope`. The `--admin-username` and `--admin-password` parameters specify the administrative user name and password account for virtual machine sign-in. Update these values as appropriate for your environment:

Bash Copy

```
az vm create --resource-group myResourceGroup \
    --name myVM --image win2016datacenter \
    --generate-ssh-keys \
    --assign-identity \
    --role contributor \
    --scope mySubscription \
    --admin-username azureuser \
    --admin-password myPassword12
```

Enable system-assigned managed identity on an existing Azure virtual machine

Use `az vm identity assign` command enable the system-assigned identity to an existing virtual machine:

Bash Copy

```
az vm identity assign -g myResourceGroup -n myVm
```

User-assigned managed identity

To assign a user-assigned identity to a virtual machine during its creation, your account needs the **Virtual Machine Contributor** and **Managed Identity Operator** role assignments. No additional Azure AD directory role assignments are required.

Enabling user-assigned managed identities is a two-step process:

1. Create the user-assigned identity
2. Assign the identity to a virtual machine

Create a user-assigned identity

Create a user-assigned managed identity using `az identity create`. The `-g` parameter specifies the resource group where the user-assigned managed identity is created, and the `-n` parameter specifies its name.

Bash Copy

```
az identity create -g myResourceGroup -n myUserAssignedIdentity
```

Assign a user-assigned managed identity during the creation of an Azure virtual machine

The following example creates a virtual machine associated with the new user-assigned identity, as specified by the `--assign-identity` parameter, with the given `--role` and `--scope`.

Bash Copy

```
az vm create \
--resource-group <RESOURCE GROUP> \
--name <VM NAME> \
--image UbuntuLTS \
--admin-username <USER NAME> \
--admin-password <PASSWORD> \
--assign-identity <USER ASSIGNED IDENTITY NAME> \
--role <ROLE> \
--scope <SUBSCRIPTION>
```

Assign a user-assigned managed identity to an existing Azure virtual machine

Assign the user-assigned identity to your virtual machine using `az vm identity assign`.

Bash Copy

```
az vm identity assign \
    -g <RESOURCE GROUP> \
    -n <VM NAME> \
    --identities <USER ASSIGNED IDENTITY>
```

Azure SDKs with managed identities for Azure resources support

Azure supports multiple programming platforms through a series of Azure SDKs . Several of them have been updated to support managed identities for Azure resources, and provide corresponding samples to demonstrate usage.

SDK	Sample
.NET	Manage resource from a virtual machine enabled with managed identities for Azure resources enabled
Java	Manage storage from a virtual machine enabled with managed identities for Azure resources
Node.js	Create a virtual machine with system-assigned managed identity enabled

SDK	Sample
Python	Create a virtual machine with system-assigned managed identity enabled
Ruby	Create Azure virtual machine with an system-assigned identity enabled

Next unit: Acquire an access token

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Acquire an access token

3 minutes

A client application can request managed identities for Azure resources app-only access token for accessing a given resource. The token is based on the managed identities for Azure resources service principal. As such, there is no need for the client to register itself to obtain an access token under its own service principal. The token is suitable for use as a bearer token in service-to-service calls requiring client credentials.

This unit provides various code and script examples for token acquisition, as well as guidance on important topics such as handling token expiration and HTTP errors.

> **Important**
>
> All sample code/script in this unit assumes the client is running on a virtual machine with managed identities for Azure resources.

Acquire a token

The fundamental interface for acquiring an access token is based on REST, making it accessible to any client application running on the VM that can make HTTP REST calls. This is similar to the Azure AD programming model, except the client uses an endpoint on the virtual machine versus an Azure AD endpoint.

Sample request using the Azure Instance Metadata Service (IMDS) endpoint:

HTTP Copy

```
GET 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://management.azure.com/' HTTP/1.1 Metadata: true
```

Element	Description

Element	Description
`GET`	The HTTP verb, indicating you want to retrieve data from the endpoint. In this case, an OAuth access token.
`http://169.254.169.254/metadata/identity/oauth2/token`	The managed identities for Azure resources endpoint for the Instance Metadata Service.
`api-version`	A query string parameter, indicating the API version for the IMDS endpoint. Please use API version `2018-02-01` or greater.
`resource`	A query string parameter, indicating the App ID URI of the target resource. It also appears in the `aud` (audience) claim of the issued token. This example requests a token to access Azure Resource Manager, which has an App ID URI of `https://management.azure.com/`.
`Metadata`	An HTTP request header field, required by managed identities for Azure resources as a mitigation against Server Side Request Forgery (SSRF) attack. This value must be set to "true", in all lower case.

Sample response:

JSON Copy

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "access_token": "eyJ0eXAi...",
  "refresh_token": "",
  "expires_in": "3599",
  "expires_on": "1506484173",
  "not_before": "1506480273",
```

```
  "resource": "https://management.azure.com/",
  "token_type": "Bearer"
}
```

Get a token by using C#

The code sample below builds the request to acquire a token, calls the endpoint, and then extracts the token from the response.

C# Copy

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Web.Script.Serialization;

// Build request to acquire managed identities for Azure resources token
HttpWebRequest request =
(HttpWebRequest)WebRequest.Create("http://169.254.169.254/metadata/identity/oauth2/to
ken?api-version=2018-02-01&resource=https://management.azure.com/");
request.Headers["Metadata"] = "true";
request.Method = "GET";

try
{
    // Call /token endpoint
    HttpWebResponse response = (HttpWebResponse)request.GetResponse();

    // Pipe response Stream to a StreamReader, and extract access token
    StreamReader streamResponse = new StreamReader(response.GetResponseStream());
    string stringResponse = streamResponse.ReadToEnd();
    JavaScriptSerializer j = new JavaScriptSerializer();
    Dictionary<string, string> list = (Dictionary<string, string>)
j.Deserialize(stringResponse, typeof(Dictionary<string, string>));
    string accessToken = list["access_token"];
}
catch (Exception e)
{
    string errorText = String.Format("{0} \n\n{1}", e.Message, e.InnerException !=
null ? e.InnerException.Message : "Acquire token failed");
}
```

Token caching

While the managed identities for Azure resources subsystem does cache tokens, we also recommend to implement token caching in your code. As a result, you should prepare for scenarios where the resource indicates that the token is expired.

On-the-wire calls to Azure Active Directory result only when:

- Cache miss occurs due to no token in the managed identities for Azure resources subsystem cache.
- The cached token is expired.

Retry guidance

It is recommended to retry if you receive a 404, 429, or 5xx error code. Throttling limits apply to the number of calls made to the IMDS endpoint. When the throttling threshold is exceeded, IMDS endpoint limits any further requests while the throttle is in effect. During this period, the IMDS endpoint will return the HTTP status code 429 ("Too many requests"), and the requests fail.

Next unit: Knowledge check

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Knowledge check

3 minutes

Check your knowledge

1. Which of the following characteristics is indicative of user-assigned identities?

- (●) Shared lifecycle with an Azure resource

  ✗ **That's incorrect. User-assigned identities have an independent lifecycle from the resources they are associated with.**

- ( ) Independent life-cycle

  ✓ **That's correct. User-assigned identities exist independently from the resources they are associated with and must be explicitly deleted.**

- ( ) Can only be associated with a single Azure resource

2. A client app requests managed identities for an access token for a given resource. Which of the below is the basis for the token?

- (●) Oauth 2.0

  ✗ **That's incorrect. Oauth 2.0 is a protocol that can be used to acquire a token, but is not the basis for the token.**

- ( ) Service principal

  ✓ **That's correct. The token is based on the managed identities for Azure resources service principal.**

- ( ) Virtual machine

Next unit: Summary

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Summary

3 minutes

In this module, you learned how to:

- Explain the differences between the two types of managed identities
- Describe the flows for user- and system-assigned managed identities
- Configure managed identities
- Acquire access tokens by using REST and code

Module complete:

Unlock achievement

How are we doing? ☆ ☆ ☆ ☆ ☆