

# Microsoft Identity Platform for Developers

---

VAIBHAV GUJRAL  
CLOUD ARCHITECT | MICROSOFT AZURE MVP

# About me

---



Director, Global Microsoft Cloud CoE at Capgemini

Born and brought up in India and based out of Omaha, NE since 2016

Microsoft Azure MVP since 2020

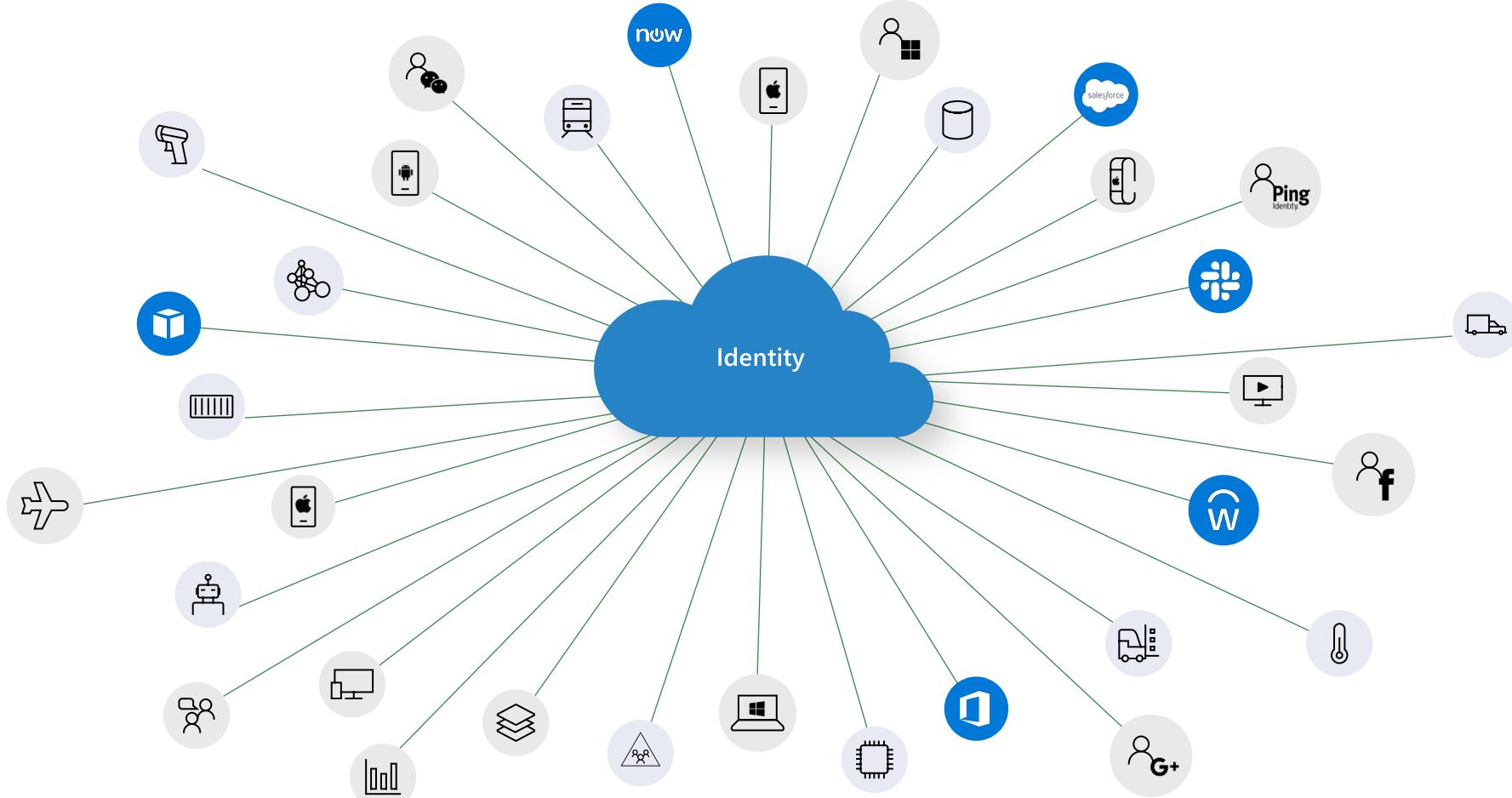
Leader, Omaha Azure User Group(<https://omahaazure.org>)

15+ cloud certifications and counting...

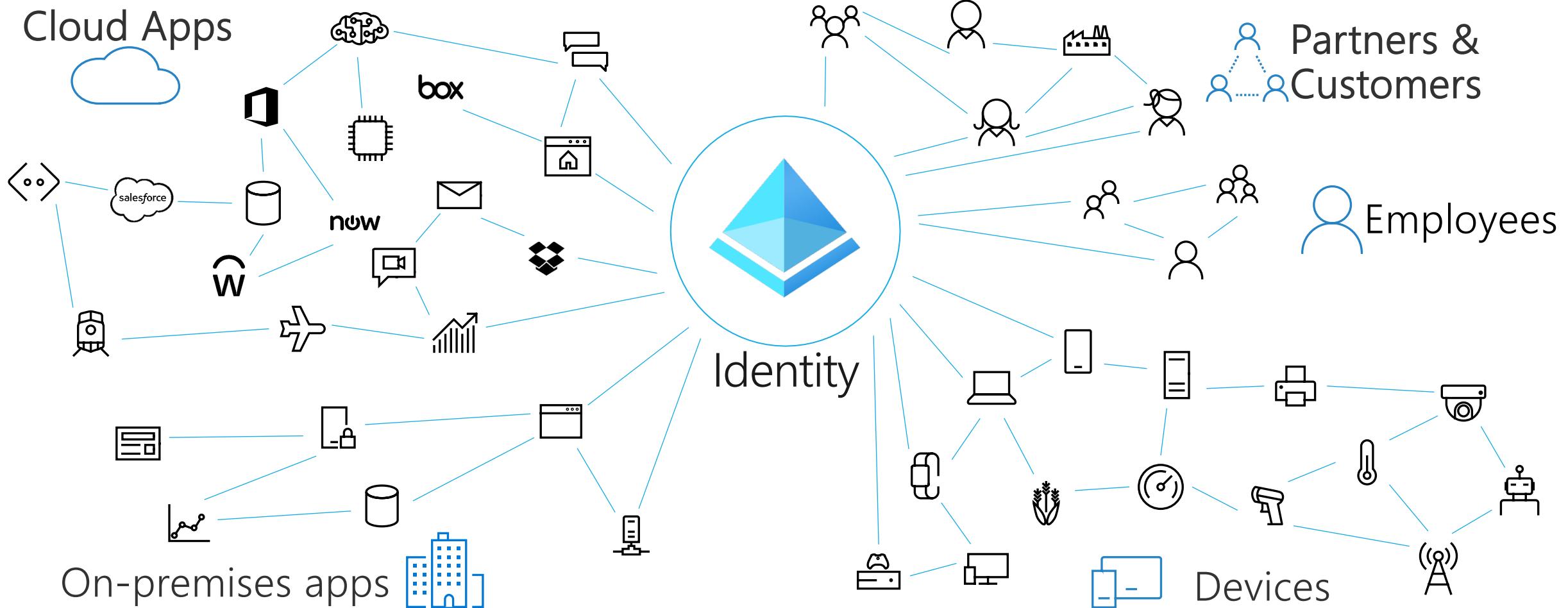


# Identity is the control plane

---



# Identity is the control plane



# What developers ask for with Identity & Access Management?



Can I leverage my company's existing identity and access management solution to save costs?



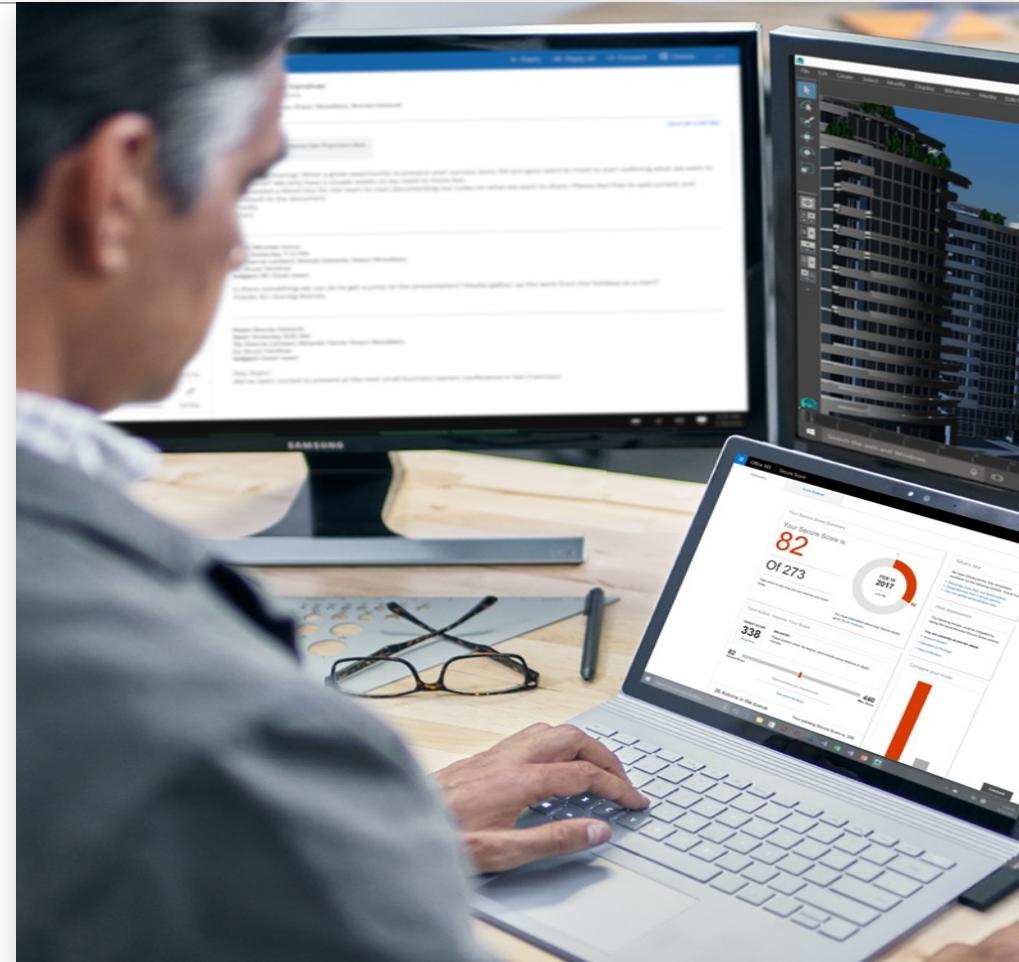
How easy is it to quickly get started and build authentication into my applications?



Does it support or integrate well with the platform, language and tools I use?



Does it support advanced security capabilities out-of-the-box, so I don't have to build it?



# Microsoft Identity Platform

---

Safeguard your apps with continuous identity innovation without any additional code



MFA



Passwordless



Seamless user experience



Access lifecycle management



Conditional Access

# What is Microsoft Identity Platform?

---



## Reduce sign-in friction

Simplify sign-in to your app and reach millions of users



## Access organizational data

Customize, extend or connect your apps to APIs such as Microsoft Graph



## Comply with IT

Meet enterprise security and compliance requirements



## Safeguard access

Protect access to your app to only authorized users

# Microsoft Identity Platform

---

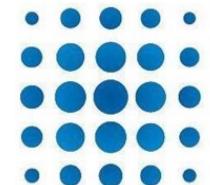


**SAML**<sup>v2.0</sup>

**SCIM**

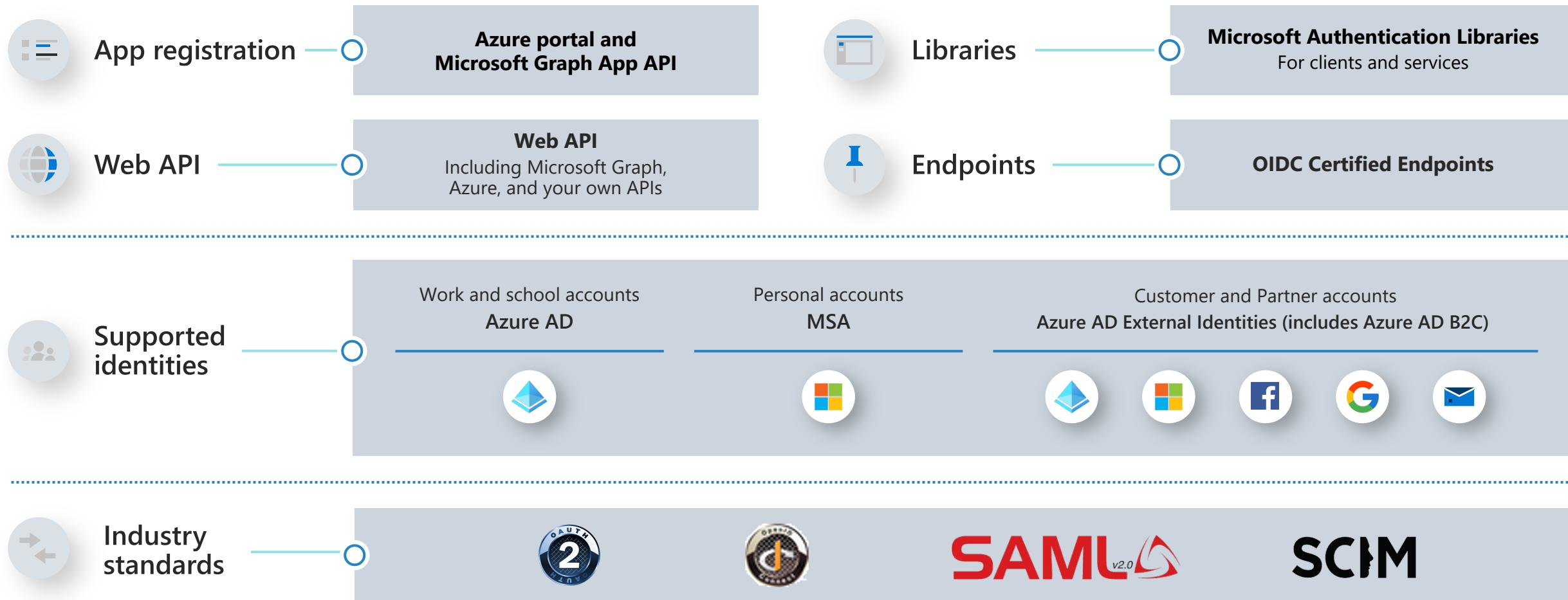
**fido**<sup>TM</sup>  
ALLIANCE

WebAuthn

 **DIF**

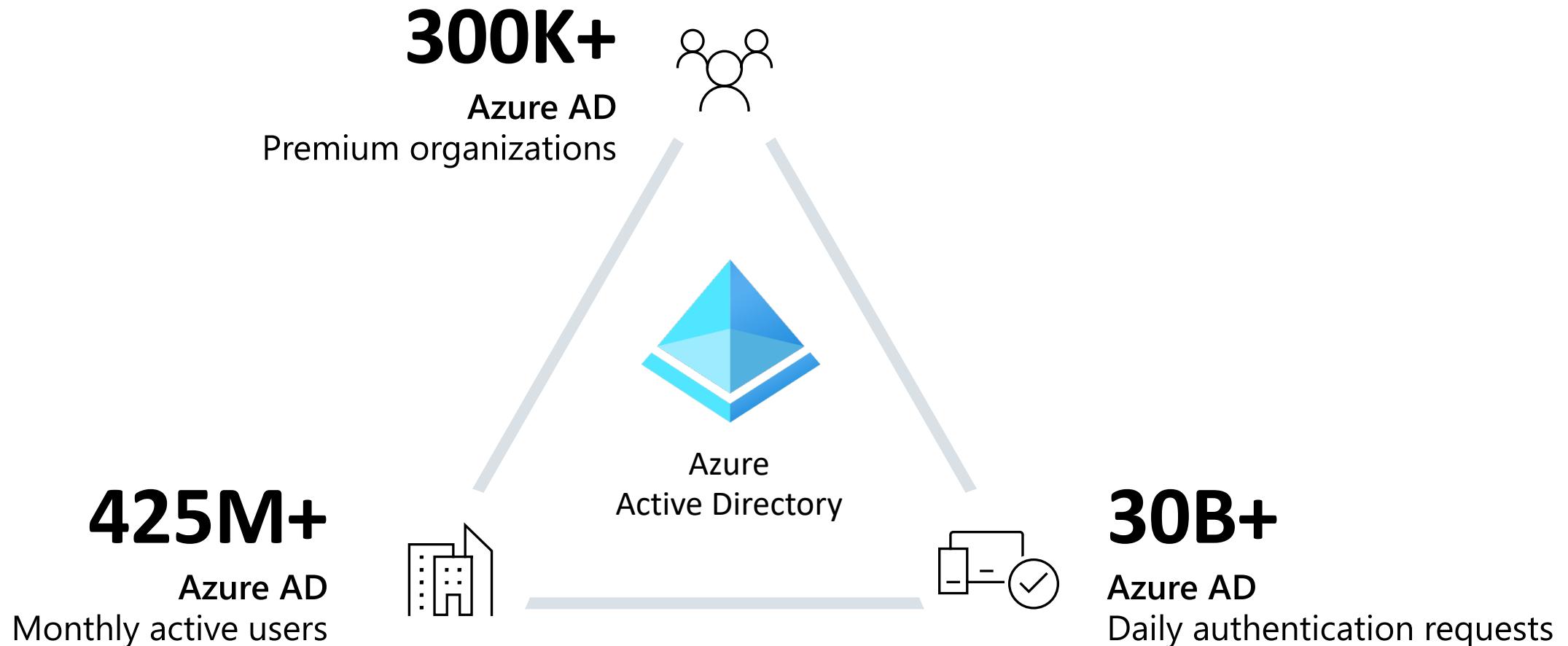
The DIF logo consists of a grid of blue dots of varying sizes followed by the letters "DIF".

# Microsoft identity platform for developers



# Azure AD – the world's largest cloud identity service

---



# Azure Active Directory



## Azure Active Directory



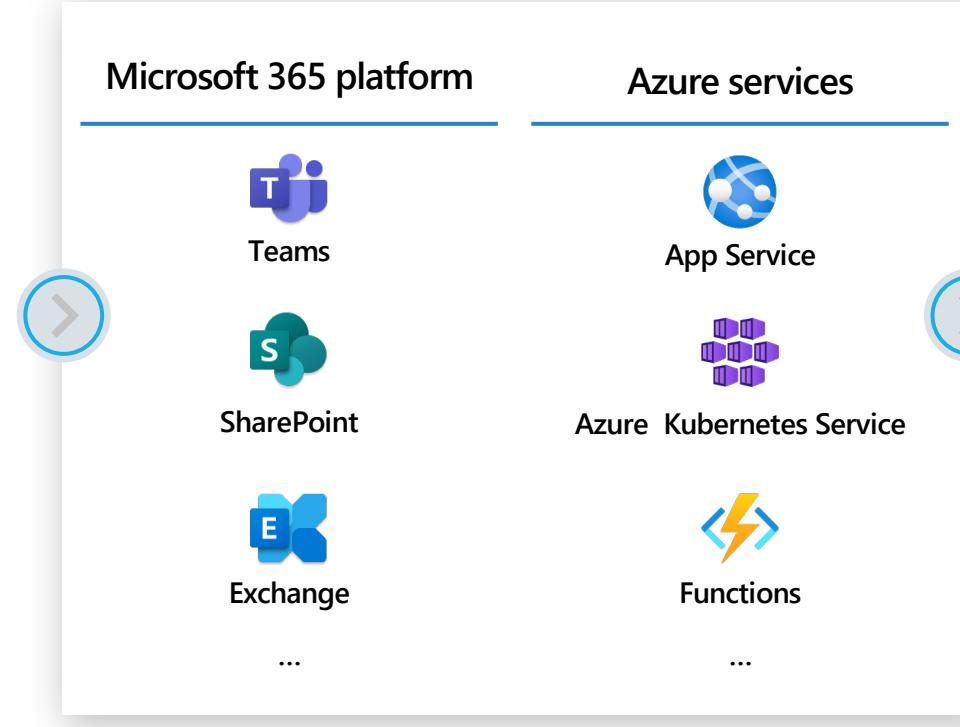
GitHub



Visual Studio



Visual Studio Code



# Azure Active Directory

Each **physical datacenter** protected with world-class, multi-layered protection, and engineered for maximum availability



**Global cloud infrastructure** with secure hardware and data segregation

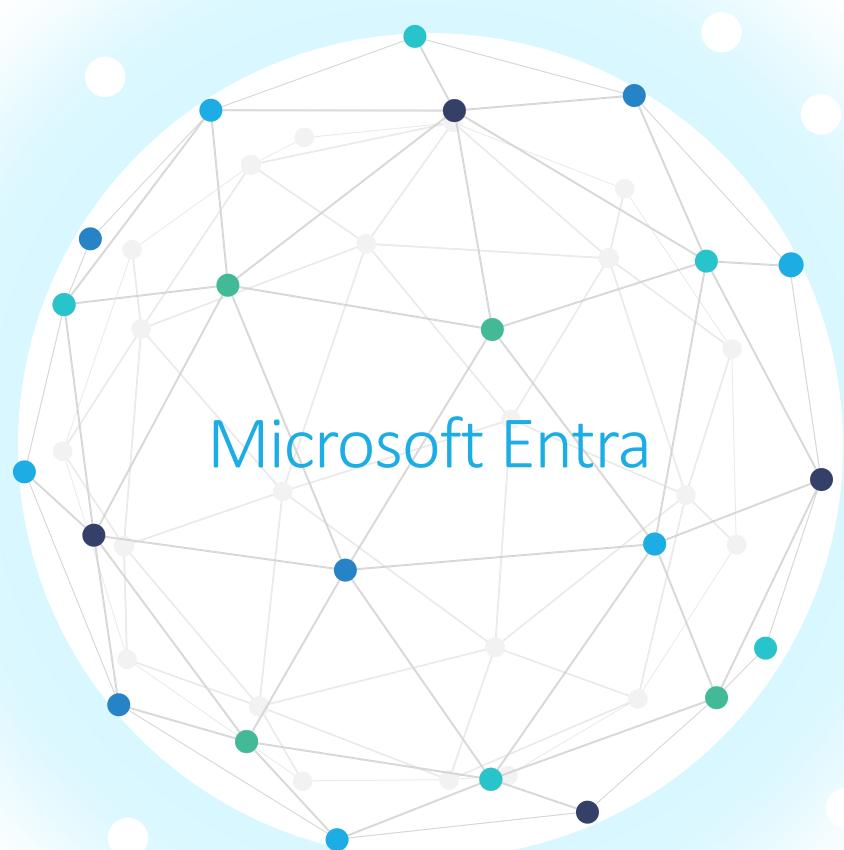
**99.99%**  
New Azure AD  
Service Level Agreement  
(in effect from April 1<sup>st</sup>, 2021)

Secured with cutting-edge **operational security**

- Restricted access
- 24x7 monitoring
- Global security experts

# Microsoft Entra

---



**Microsoft Entra is a product family name for Microsoft's identity and access solutions.**

**It is not a replacement of Azure Active Directory (Azure AD). Azure AD is the identity solution and part of Microsoft Entra.**

**A new name was introduced because Microsoft has expanded in several new categories in Security and needed a name to convey modern access security across broad range of products.**

# Microsoft Entra

---



Azure  
Active Directory

+



Microsoft Entra  
Permissions Management

+



Microsoft Entra  
Verified ID

+



Microsoft Entra  
Identity Governance

+



Microsoft Entra  
Workload Identities

# Authentication vs. Authorization

---

**Authentication** is the process of proving that you are who you say you are. This is achieved by verification of the identity of a person or device. It's sometimes shortened to *AuthN*.

Microsoft identity platform uses OpenID Connect protocol for handling authentication.

**Authorization** is the act of granting an authenticated party permission to do something. It specifies what data you're allowed to access and what you can do with that data. Authorization is sometimes shortened to *AuthZ*.

Microsoft identity platform uses OAuth 2.0 protocol for handling authorization.

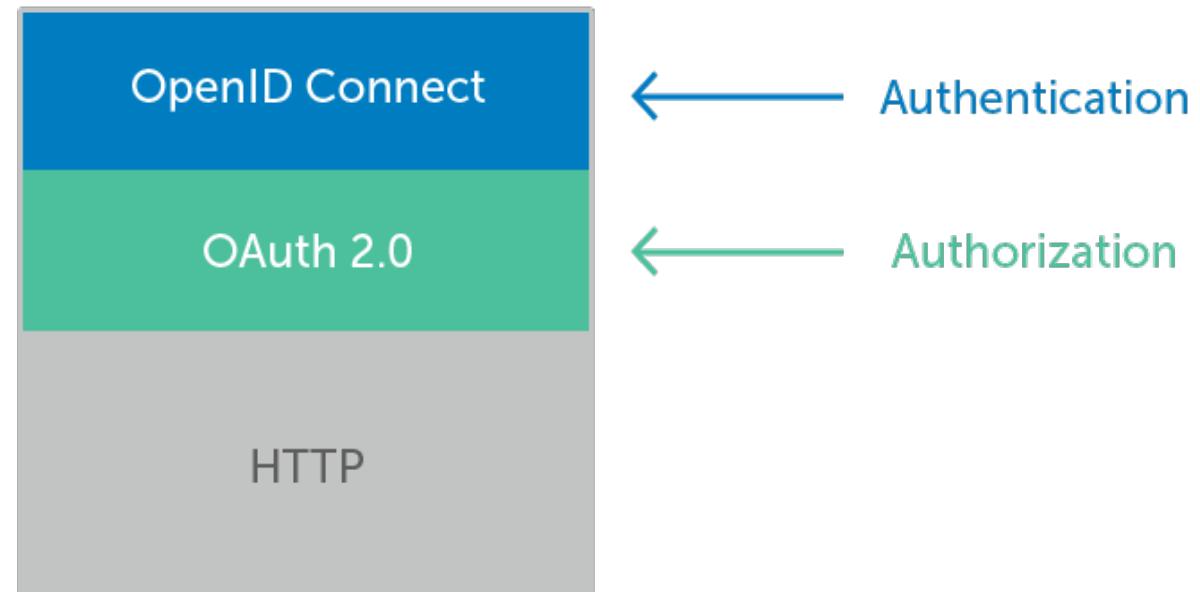
# OAuth 2.0 and OpenID Connect

**OAuth 2.0** is an authorization framework enables a third-party application to obtain limited access to an HTTP service.

**OAuth 2.0** specification: <https://www.rfc-editor.org/rfc/rfc6749>

**OpenID Connect 1.0** is a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

OpenID Connect Documentation -  
<https://openid.net/connect/>



# OAuth Roles

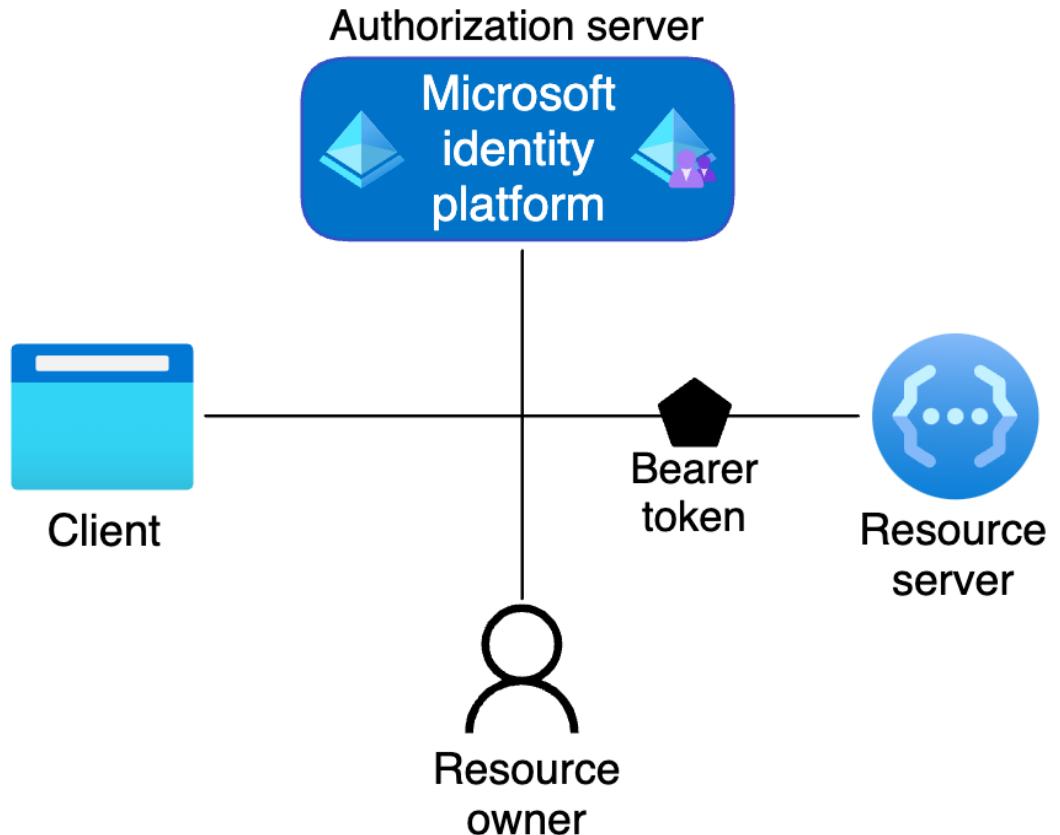
OAuth defines four roles:

**Resource owner:** An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user.

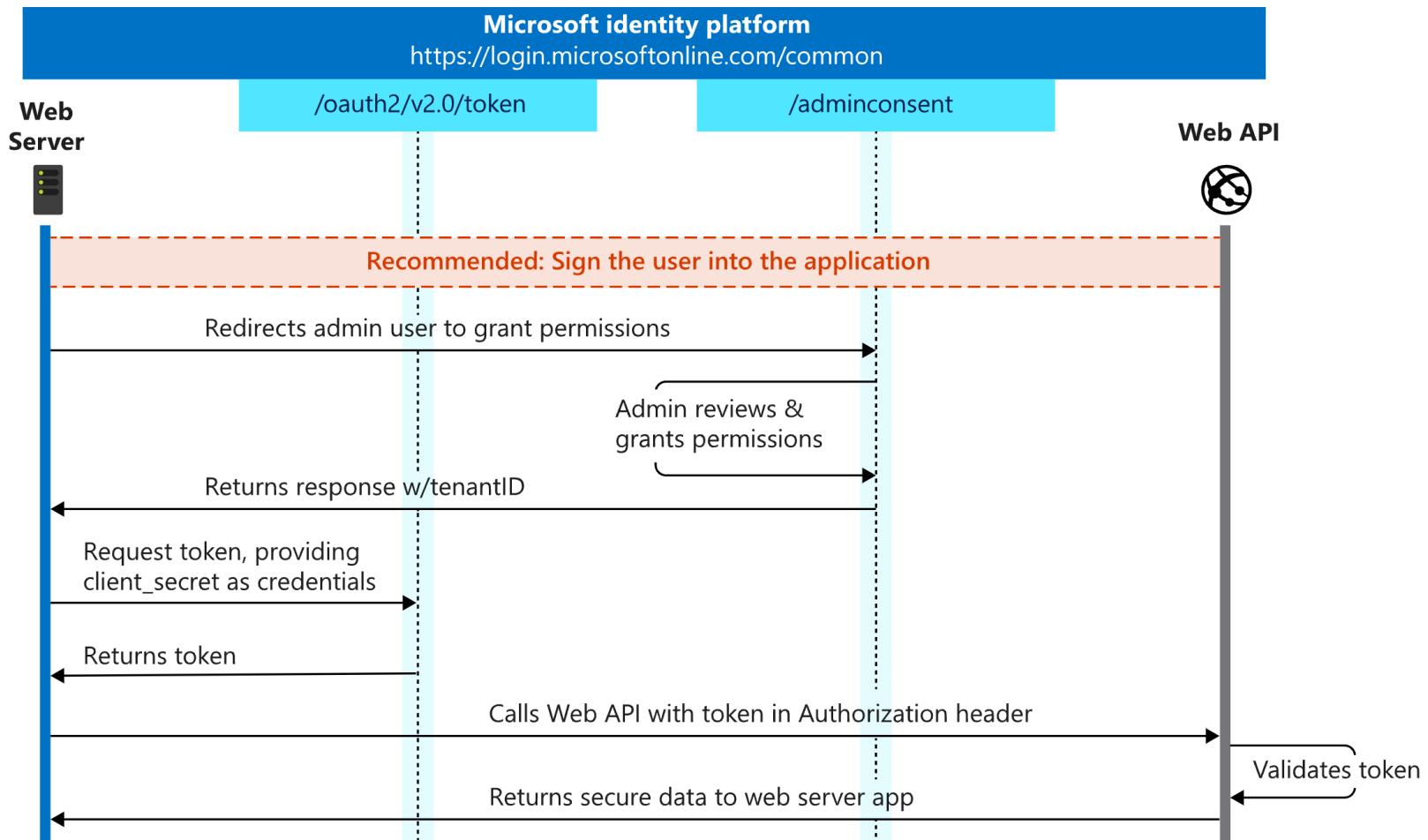
**Resource server:** The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.

**Client:** An application making protected resource requests on behalf of the resource owner and with its authorization. The term "client" does not imply any particular implementation characteristics (e.g., whether the application executes on a server, a desktop, or other devices).

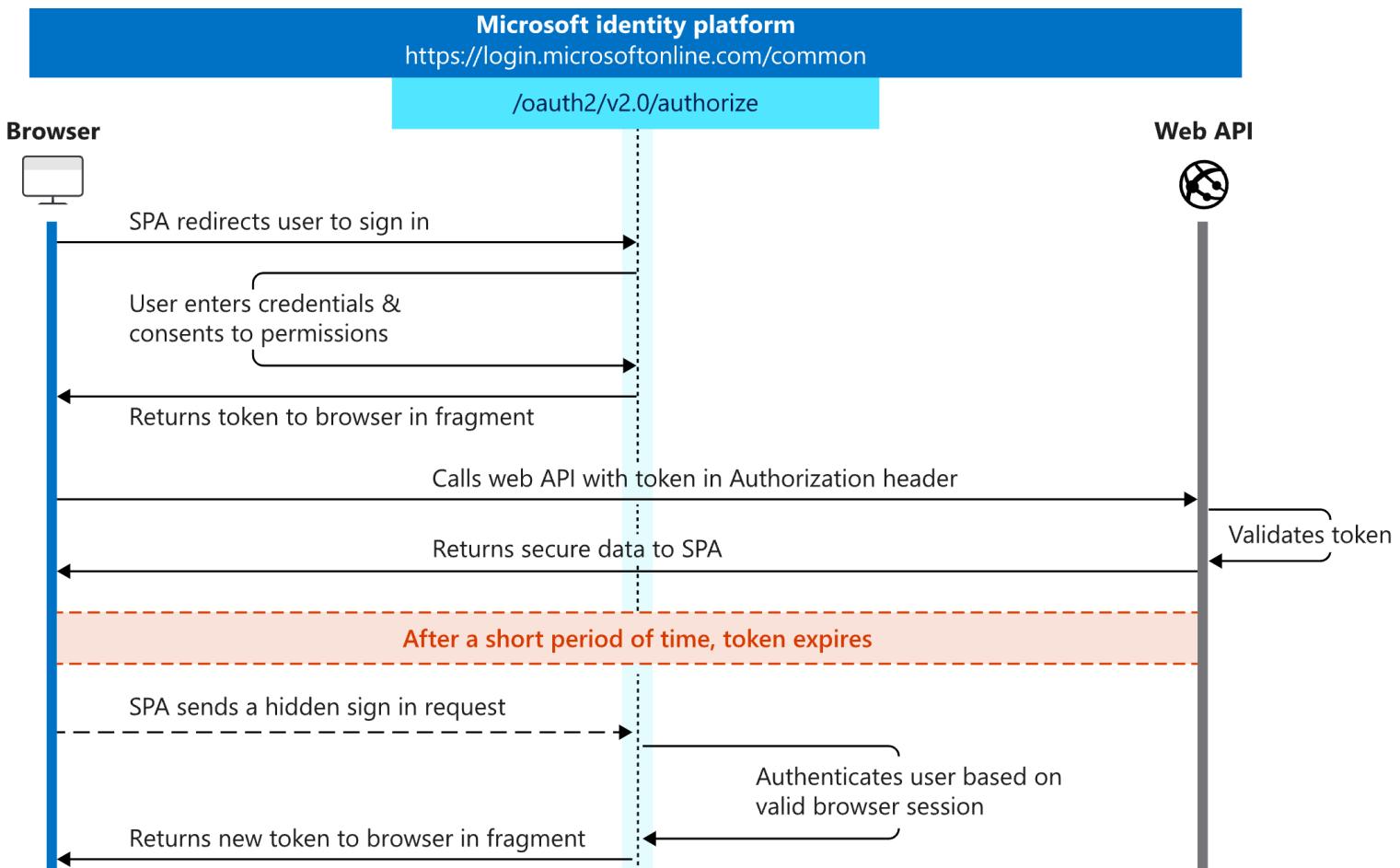
**Authorization server:** The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.



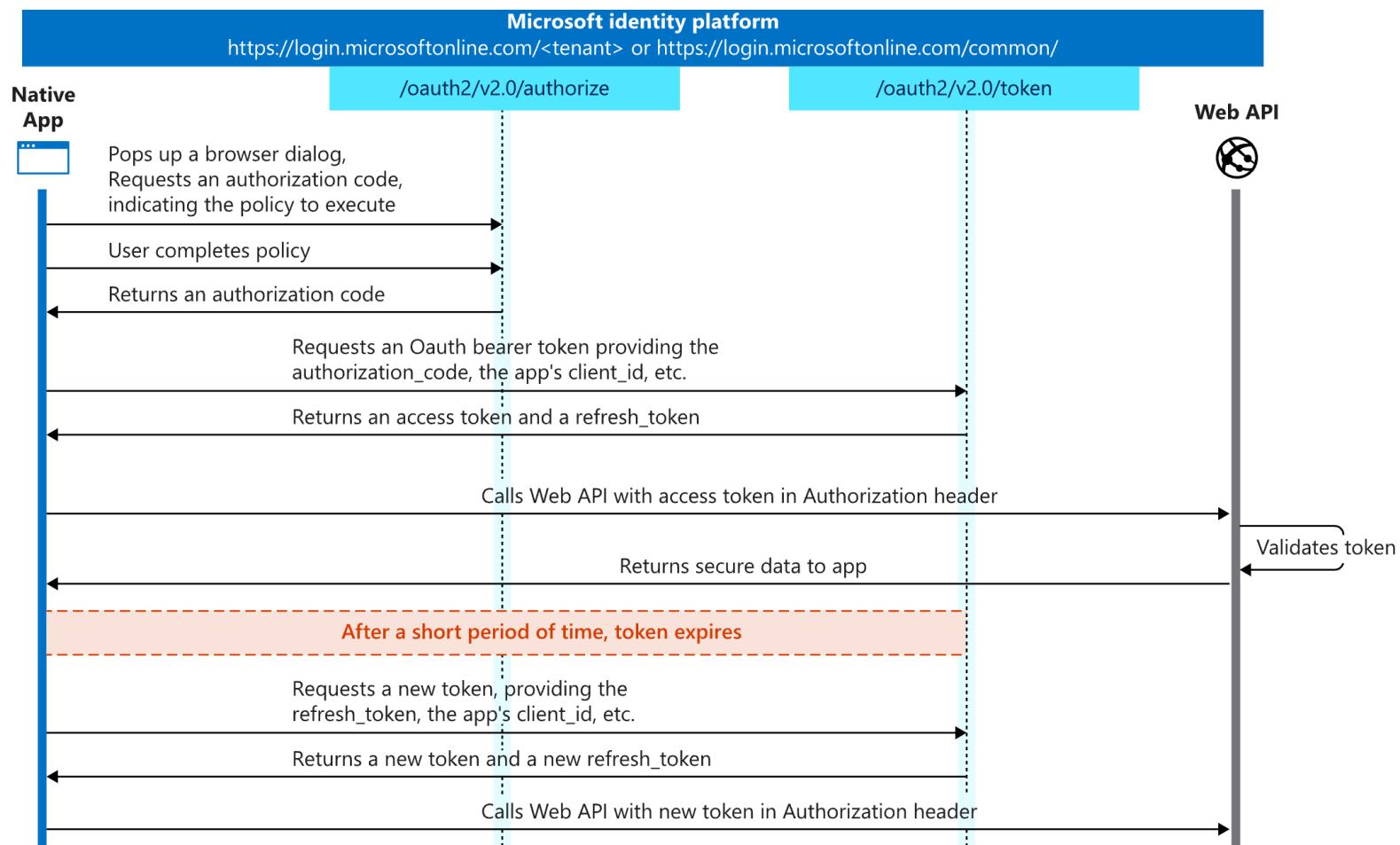
# Client Credentials Flow



# Implicit Grant Flow

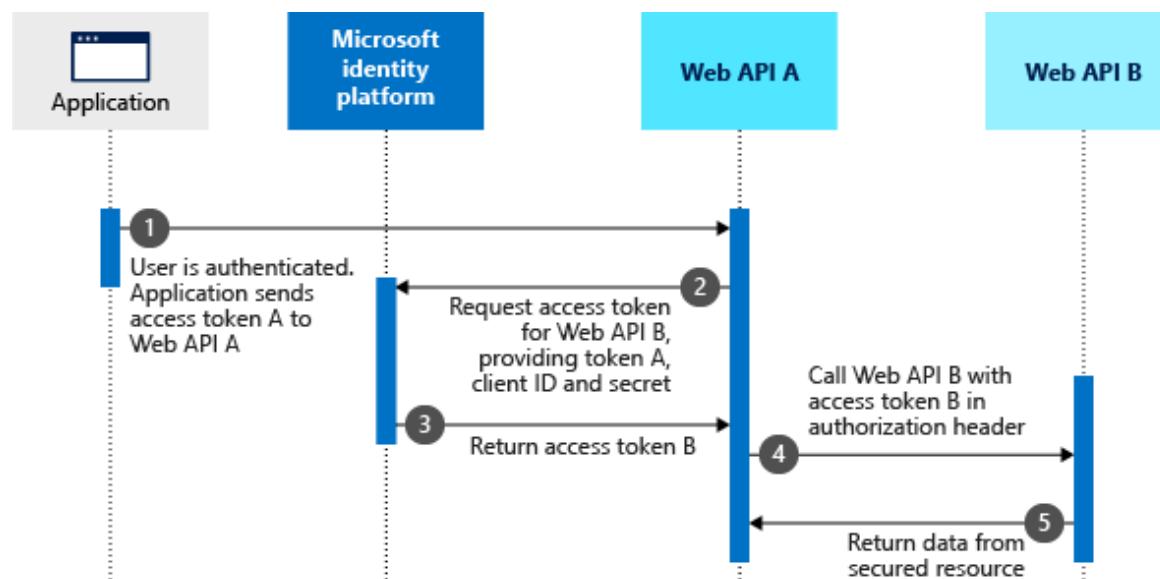


# Auth Code Flow



# On behalf of flow

---



# Bearer Token Types

---

**Access tokens** - Access tokens are issued by the authorization server to the client application. The client passes access tokens to the resource server. Access tokens contain the permissions the client has been granted by the authorization server.

**ID tokens** - ID tokens are issued by the authorization server to the client application. Clients use ID tokens when signing in users and to get basic information about them.

**Refresh tokens** - The client uses a refresh token, or *RT*, to request new access and ID tokens from the authorization server. Your code should treat refresh tokens and their string content as sensitive data because they're intended for use only by authorization server.

# JWT Format

---

JWTs are split into three pieces:

**Header** - Provides information about how to validate the token including information about the type of token and how it was signed.

**Payload** - Contains all of the important data about the user or application that's attempting to call the service.

**Signature** - Is the raw material used to validate the token.

Example -

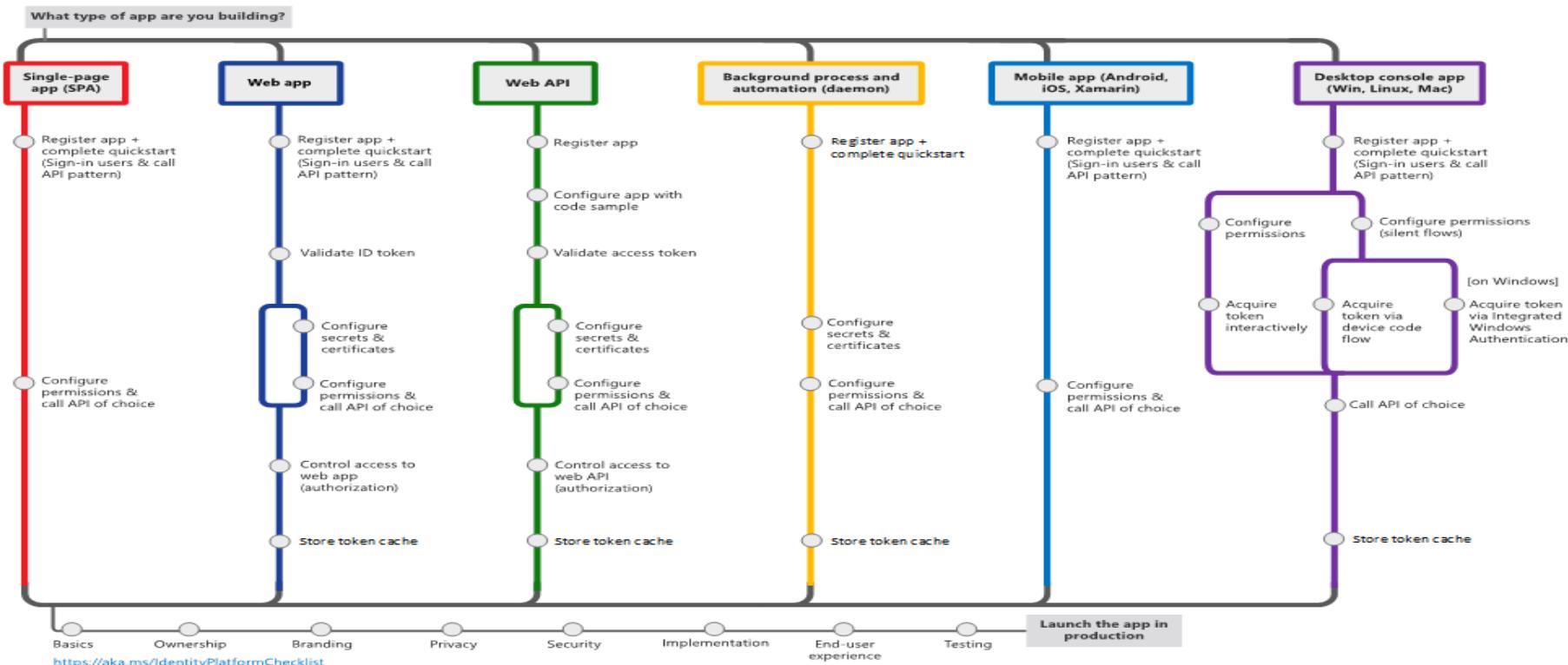
```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NlslmtpZCI6Imk2bEdrM0ZaenhSY1ViMkMzbkVRN3N5SEpsWSJ9.eyJhdWQiOi2ZTc0MTcyYi1iZTU2LTQ4NDMtOWZmNC1INjZhMzliYjEyZTMiLCJpc3MiOiJodHRwczovL2xvZ2luLm1pY3Jvc29mdG9ubGluZS5jb20vNzJmOTg4YmYtODZmMS00MWFMltkxYWItMmQ3Y2QwMTFkYjQ3L3YyLjAiLCJpYXQiOjE1MzcycMzEwNDgsIm5iZiI6MTUzNzlzMTA0OCwiZXhwIjoxNTM3MjM0OTQ4LCJhaW8iOjBWFBBaS84SUFBQUF0QWFaTG8zQ2hNaWY2S09udHRSQjdlQnE0L0RjY1F6amNKR3hQWXkvQzNqRGFOR3hYZDZ3TkIJVkdSz2hOUm53SjFsT2NBbk5aY2p2a295ckZ4Q3R0djMzMTQwUmlvT0ZKNGJDQ0dWdW9DYWcxdu9UVDIyMjlyZ0h3TFBZUS91Zjc5UVgrMETJaWpkcm1wNjIy3R6bVE9PSlsmF6cCl6ljZINzQxNzJiLWJINTYtNDg0My05ZmY0LWU2NmEzOWJiMTJiMyIsImF6cGFjciI6ljAiLCJuYw1IjoiQWJIIExpbmNvbG4iLCJvaWQiOjI2OTAyMjJiZS1mZjFhLTRkNTYtYyWjkMS03ZTRmN2QzOGU0NzQiLCJwcmVmZXJyZWRfdXNlcm5hbWUiOjhYmVsaUBtaWNyb3NvZnQuY29tliwicmgioIjIiwic2NwljoiYWNjZXNzX2FzX3VzZXliLCJzdWliOjJIS1pwZmFleVdhZGVpb3VzbGl0anJJLutmZIRtMjlyWDVyclYzeERxZktRliwidGlkljoiNzJmOTg4YmYtODZmMS00MWFMltkxYWItMmQ3Y2QwMTFkYjQ3liwidXRpljoiZnFpQnFYTFBqMGVRTgyUy1JWUZBQSlsInZlcil6ljluMCJ9.pj4N-w_3Us9DrBLfpCt
```

To decode, verify and generate JWT, check out - <https://jwt.io/>

# App Types

## Microsoft identity platform

<http://aka.ms/IdentityPlatform>



# Microsoft Authentication Libraries (MSAL)

---



JavaScript



.NET



Android



iOS



Python



Java



Angular



Microsoft  
Identity Web

# Simplify sign-in with MSAL



## Develop in your favorite language

Best in class authentication libraries that work with your platform or language of choice or use our OIDC certified endpoint.



## Secure by default

Applications using MSAL are secure by default and can comply with security policies implemented by IT.



## Build richer experiences

Secure access to users and data from Microsoft Graph, Azure or your own protected APIs.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with navigation links such as Home, Overview, Quickstart, Integration assistant (preview), Manage, Branding, Authentication, Certificates & secrets, Token configuration, API permissions (which is selected), Expose an API, Owners, Roles and administrators (Preview), Manifest, Support + Troubleshooting, Troubleshooting, and New support request. The main content area is titled 'Contoso Organizer | API permissions'. It shows a table with one row for 'Microsoft Graph (1)'. The row details are: API / Permissions name: Microsoft Graph, Type: User.Read, and Description: Delegated, Sign in. Below this table, it says 'Configured permissions' and 'Applications are authorized to call APIs when they are granted all the permissions the application needs.' A link 'Learn more about permissions' is provided. To the right, there's a section titled 'Request API permissions' with a heading 'Select an API'. It shows a list of 'Commonly used Microsoft APIs' with icons and names: Microsoft Graph, Azure Rights Management Services, Flow Service, OneNote, Skype for Business, and Yammer. Each item has a brief description below it. At the bottom right of the main content area, there's a link 'More Microsoft APIs'.

# Application Registration

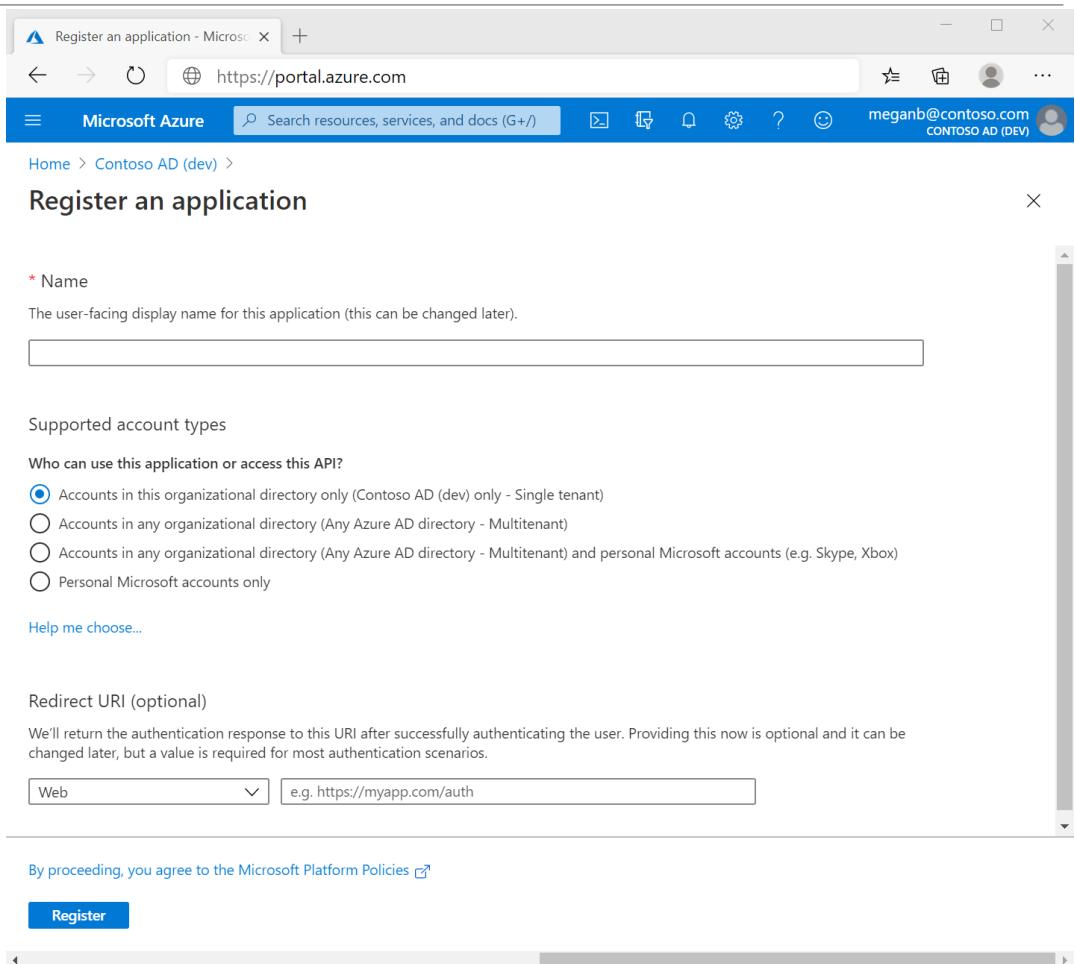
To leverage Microsoft identity platform for identity and access management (IAM), you must first register your application.

An application would be a client application like a web or mobile app, or a web API that backs a client app.

Registering an application establishes a trust relationship between the application and the identity provider, the Microsoft identity platform.

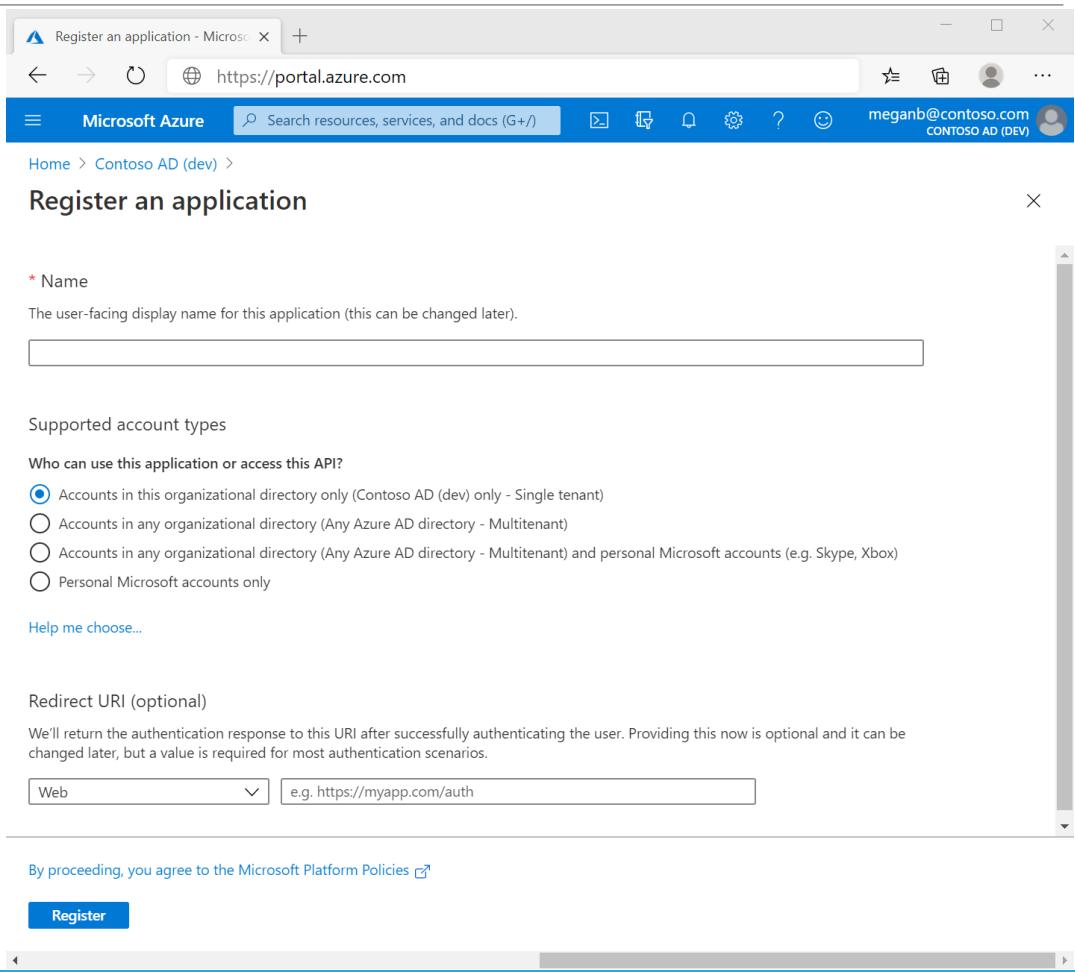
Two of the most commonly referenced app registration settings are:

- **Application (client) ID** - this value is assigned by the identity platform and is included in the security tokens the platform issues.
- **Redirect URI** - The authorization server uses a redirect URI to direct the resource owner's *user-agent* (web browser, mobile app) to another destination after completing their interaction. Not all client types use redirect URIs.



# Application Registration

Supported account types	Description
<b>Accounts in this organizational directory only</b>	Select this option if you're building an application for use only by users (or guests) in your tenant.  Often called a line-of-business (LOB) application, this app is a single-tenant application in the Microsoft identity platform.
<b>Accounts in any organizational directory</b>	Select this option if you want users in any Azure Active Directory (Azure AD) tenant to be able to use your application. This option is appropriate if, for example, you're building a software-as-a-service (SaaS) application that you intend to provide to multiple organizations.  This type of app is known as a multitenant application in the Microsoft identity platform.
<b>Accounts in any organizational directory and personal Microsoft accounts</b>	Select this option to target the widest set of customers.  By selecting this option, you're registering a multitenant application that can also support users who have personal Microsoft accounts.
<b>Personal Microsoft accounts</b>	Select this option if you're building an application only for users who have personal Microsoft accounts. Personal Microsoft accounts include Skype, Xbox, Live, and Hotmail accounts.



The screenshot shows the 'Register an application' page in the Microsoft Azure portal. The URL in the address bar is https://portal.azure.com. The top navigation bar includes links for Home, Contoso AD (dev), and a user profile for meganb@contoso.com. The main form has the following fields:

- Name:** A text input field labeled with an asterisk (\*).
- Supported account types:** A section with three radio button options:
  - Accounts in this organizational directory only (Contoso AD (dev) - Single tenant)
  - Accounts in any organizational directory (Any Azure AD directory - Multitenant)
  - Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
  - Personal Microsoft accounts only
- Who can use this application or access this API?**: A section with four radio button options:
  - Accounts in this organizational directory only (Contoso AD (dev) - Single tenant)
  - Accounts in any organizational directory (Any Azure AD directory - Multitenant)
  - Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
  - Personal Microsoft accounts only
- Help me choose...**: A link for assistance.
- Redirect URI (optional):** A section with a dropdown menu set to 'Web' and a text input field containing 'e.g. https://myapp.com/auth'.
- By proceeding, you agree to the Microsoft Platform Policies**: A link to the policies.
- Register**: A blue button at the bottom.

# Managing Application Registrations



## Manage all your apps in one place

See all your apps that sign-in users with Azure AD or Microsoft accounts in our unified app registration experience.



## Secure your app registration

Integration Assistant highlights best practices through your app integration's lifecycle and helps ensure every stage is properly configured.

## Azure Active Directory Roles

Users in following AAD roles can register an application:

- Application Administrator
- Application Developer
- Cloud Application administrator



The screenshot shows a Microsoft Azure laptop interface. The main window is titled 'Contoso app | Integration assistant (preview)'. It displays a sidebar with options like Overview, Quickstart, and Integration assistant (preview). The main content area is titled 'Here's the integration assistant for Contoso app' and shows the application type as 'Web app' and 'Calls APIs' as 'Yes'. Below this, there are tabs for Summary, Develop, Test, Release, and Monitor, with 'Summary' selected. The 'Recommended configurations' section lists items such as 'Configure a redirect URI for a web app.', 'Configure API permissions.', and 'Assign users that should be able to view and edit this application registration as owners.' Each item has a status indicator (Action required or Complete) and a 'Dismiss item' button. The 'Discouraged configurations' section lists items like 'Do not enable access tokens for the implicit grant flow.' and 'Do not configure an Application ID URI.', both marked as 'Complete'. At the bottom right, there are buttons for 'Go to page', 'Documentation', and 'Dismiss item'.

# Service Principals

---

To access resources that are secured by an Azure AD tenant, the entity that requires access must be represented by a security principal. This requirement is true for both users (user principal) and applications (service principal).

The security principal defines the access policy and permissions for the user/application in the Azure AD tenant.

Three types of service principal:

**Application** - The type of service principal is the local representation, or application instance, of a global application object in a single tenant or directory.

**Managed identity** - Managed identities provide an identity for applications to use when connecting to resources that support Azure AD authentication without the need for developers to manage credentials

**Legacy** - This type of service principal represents a legacy app, which is an app created before app registrations were introduced or an app created through legacy experiences.

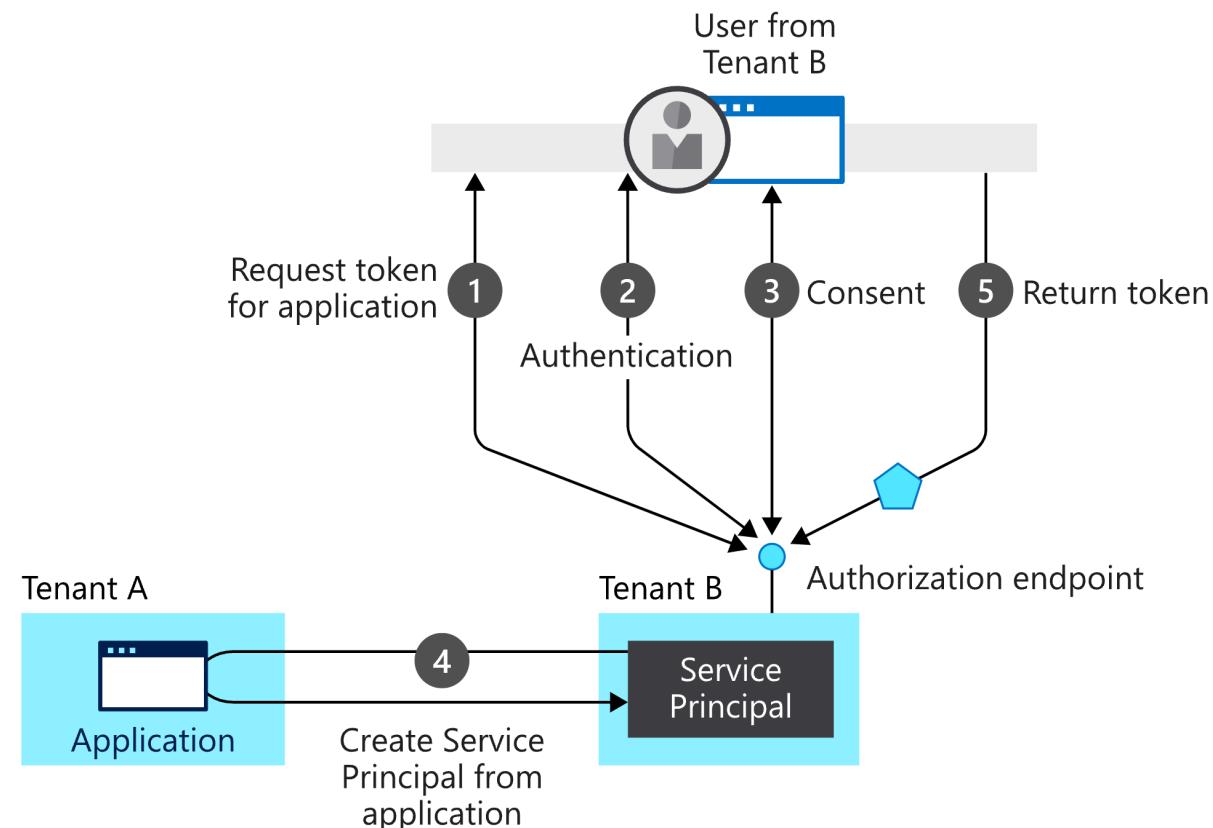
# App Registration vs Enterprise Application

The **Application Object** is seen under App Registrations in AAD where an application settings like API Permissions, Client Secrets, Branding, App Roles, etc. can be configured. All these customizations that you make to your app, get written to the app manifest file.

The application object describes three aspects of an application: how the service can issue tokens in order to access the application, resources that the application might need to access, and the actions that the application can take.

The **Service Principal Object** is what you see under the Enterprise Registration blade in AAD. Every Application Object would create a corresponding Service Principal Object in the Enterprise Registration blade of AAD. A service principal is a concrete instance created from the application object and inherits certain properties from that application object.

A service principal is created in each tenant where the application is used and references the globally unique app object. The service principal object defines what the app can actually do in the specific tenant, who can access the app, and what resources the app can access.



# App Roles

App roles can be used to implement claim-based authorization.

App roles are defined on an application registration representing a service, app or API.

App roles can be assigned to a user, to a group of users, to the service principal for another application, or to the service principal for a managed identity.

The screenshot shows the Azure portal interface for managing app roles. The URL in the address bar is [Home > saurabhmsft > MyApp](#). The main title is "MyApp | App roles". On the left, there's a sidebar with links like Overview, Quickstart, Integration assistant, Manage (with sub-links for Branding, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, App roles, Owners, Roles and administrators | Preview, and Manifest), and Support + Troubleshooting (with sub-links for Troubleshooting and New support request). A red box highlights the "Create app role" button at the top right of the main content area. Another red box highlights the "App roles" section below it. The "App roles" section contains a brief description: "App roles are custom roles to assign permissions to users or apps. The application defines and publishes the app roles and interprets them as permissions during authorization." It also includes a "How do I assign App roles?" link. Below this, a table lists three app roles:

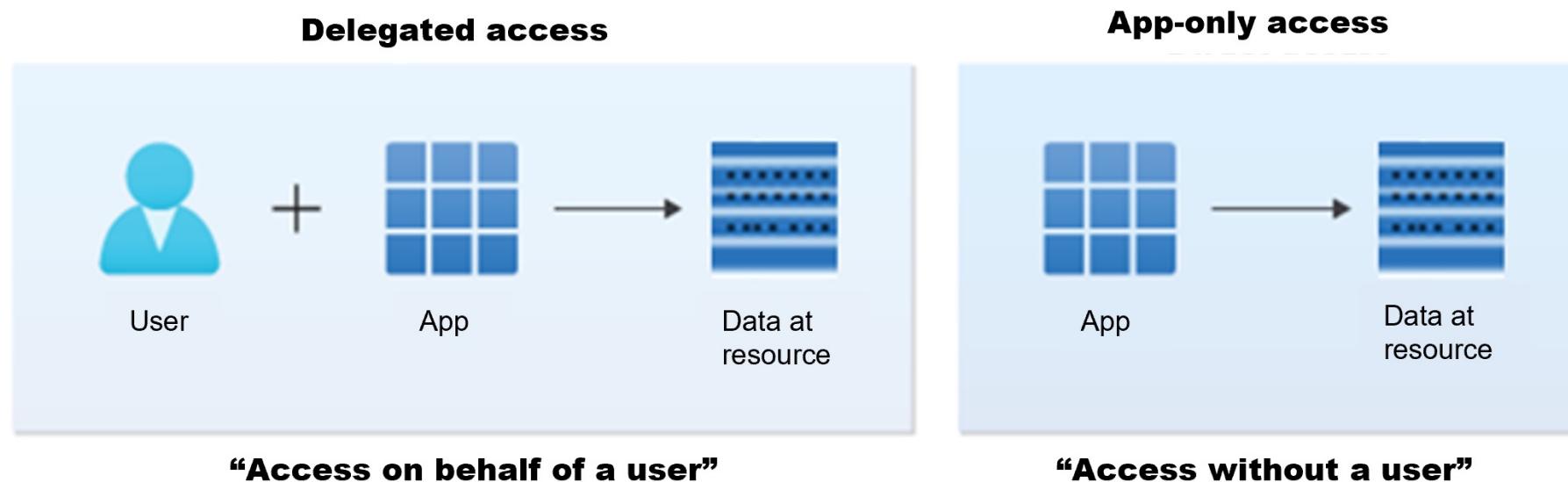
Display name	Description	Allowed member types	Value	ID	State
Writer	Writers can create surveys.	Users/Groups/Applications	Survey.Create	d4bbbe9b-96b4-4862-...	Enabled
Reader	Readers can read all surveys and reports	Users/Groups/Applications	Survey.Read	c74050d0-1eec-4244-a...	Enabled
Admin	Admins can moderate survey and publish re...	Users/Groups/Applications	Survey.Admin	39b164fc-295f-457e-8...	Enabled

# Delegated vs Application Permissions

Depending on how you want the app to access the data, you can provide access in one of the following two ways:

**Delegated access**, an app acting on behalf of a signed-in user.

**App-only access**, an app acting with its own identity.



# User Consent

Before an application can access your organization's data, a user must grant the application permissions to do so.

Different permissions allow different levels of access. By default, all users are allowed to consent to applications for permissions that don't require administrator consent.

Allow user consent only for applications that have been published by a verified publisher.

The screenshot shows two side-by-side windows from the Microsoft Azure portal.

**User consent for applications:** This window allows configuration of user consent. The "Allow user consent for apps" option is selected, indicated by a blue dot next to the radio button. Below it, a link says "7 permissions classified as low risk".

**Let this app access your info? (App info):** This window displays a consent dialog for an application. It asks if the user wants to "Read your profile". A note states: "Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. The publisher has not provided links to their terms for you to review. You can change these permissions at https://microsoft.com/consent. Show details". There are "No" and "Yes" buttons at the bottom, with "Yes" being highlighted in blue.

# Tenant-wide admin consent

Sometimes, you need to give the application access on behalf of the whole organization to the permissions requested (like role management, full access to all mailboxes or all sites, and full user impersonation).

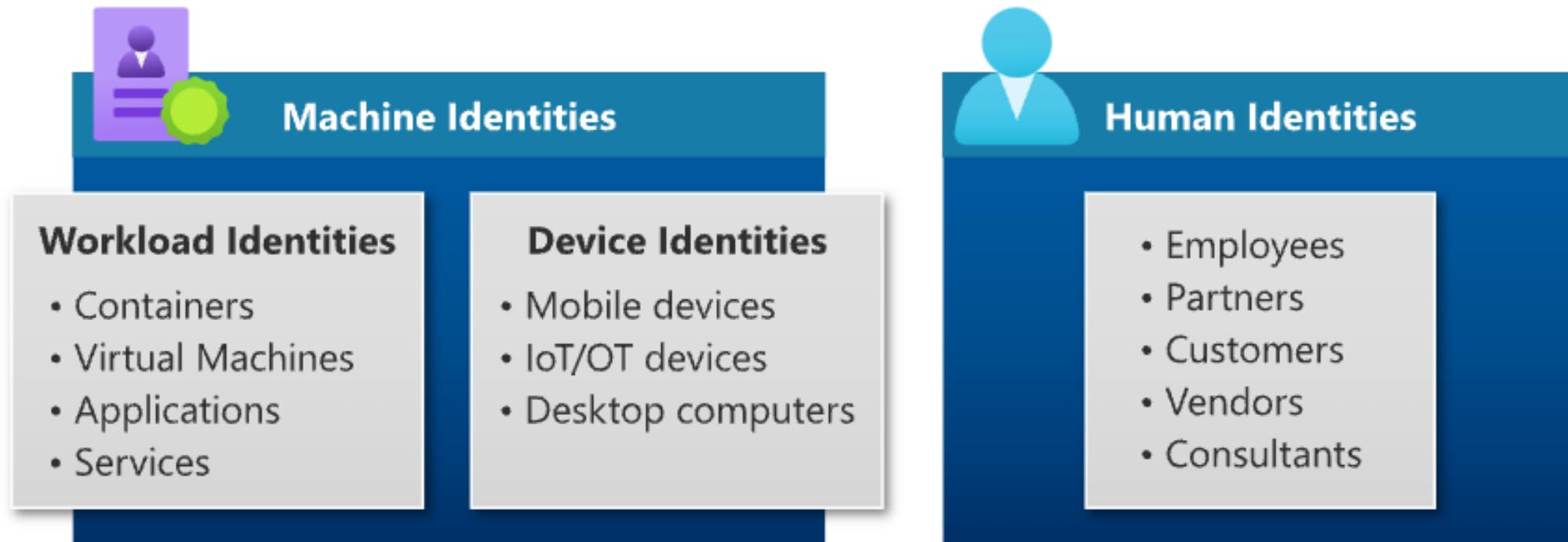
The screenshot shows the Microsoft Azure portal interface for managing API permissions. The left sidebar lists several categories: Overview, Quickstart, Integration assistant, Manage (with sub-options: Branding & properties, Authentication, Certificates & secrets, Token configuration), and API permissions (which is highlighted with a red box). The main content area is titled "Test Web App | API permissions". It includes a search bar, refresh button, and feedback link. A note states: "The 'Admin consent required' column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value in your organization, or in organizations where this app will be used." Below this is a section titled "Configured permissions" with a sub-note: "Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs." A "Grant admin consent for Fourth Coffee" button is shown with a checkmark and a plus sign. The table below lists the configured permissions:

API / Permissions name	Type	Description	Grant admin consent for Fourth Coffee	Admin consent req...	Status
Microsoft Graph (2)					...
Group.Read.All	Delegated	Read all groups	Yes	⚠️ Not granted for Fourth...	...
User.Read	Delegated	Sign in and read user profile	No		...

To view and manage permissions and user consent, try [Enterprise applications](#).

# Workload Identities

---



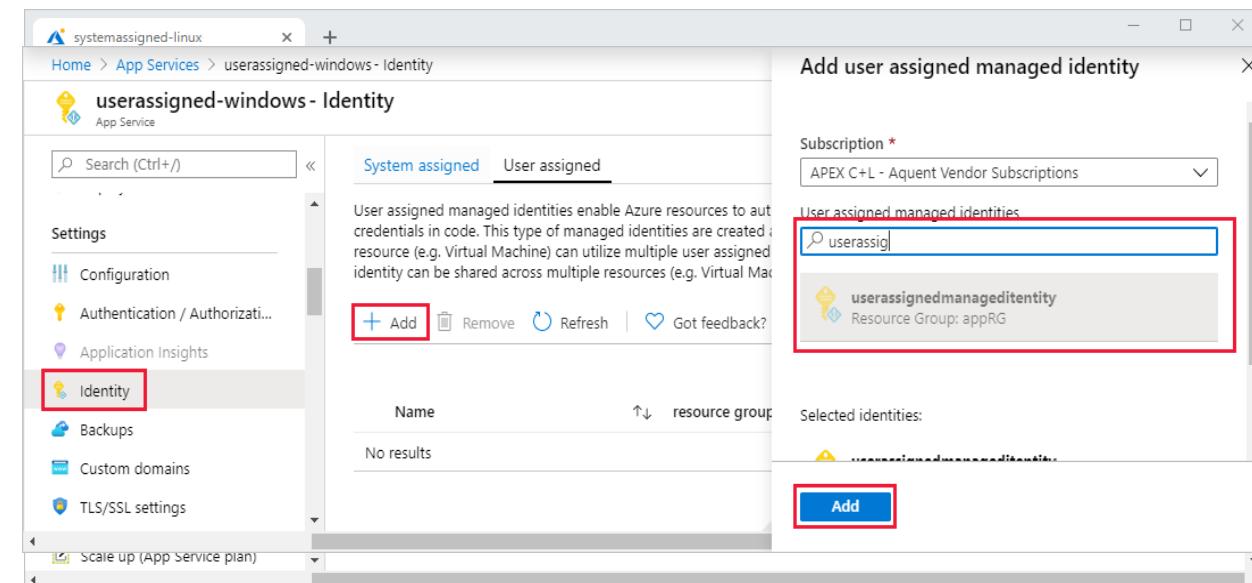
# Managed Identities

A managed identity from Azure Active Directory (Azure AD) allows your app to easily access other Azure AD-protected resources such as Azure Key Vault.

The identity is managed by the Azure platform and does not require you to provision or rotate any secrets.

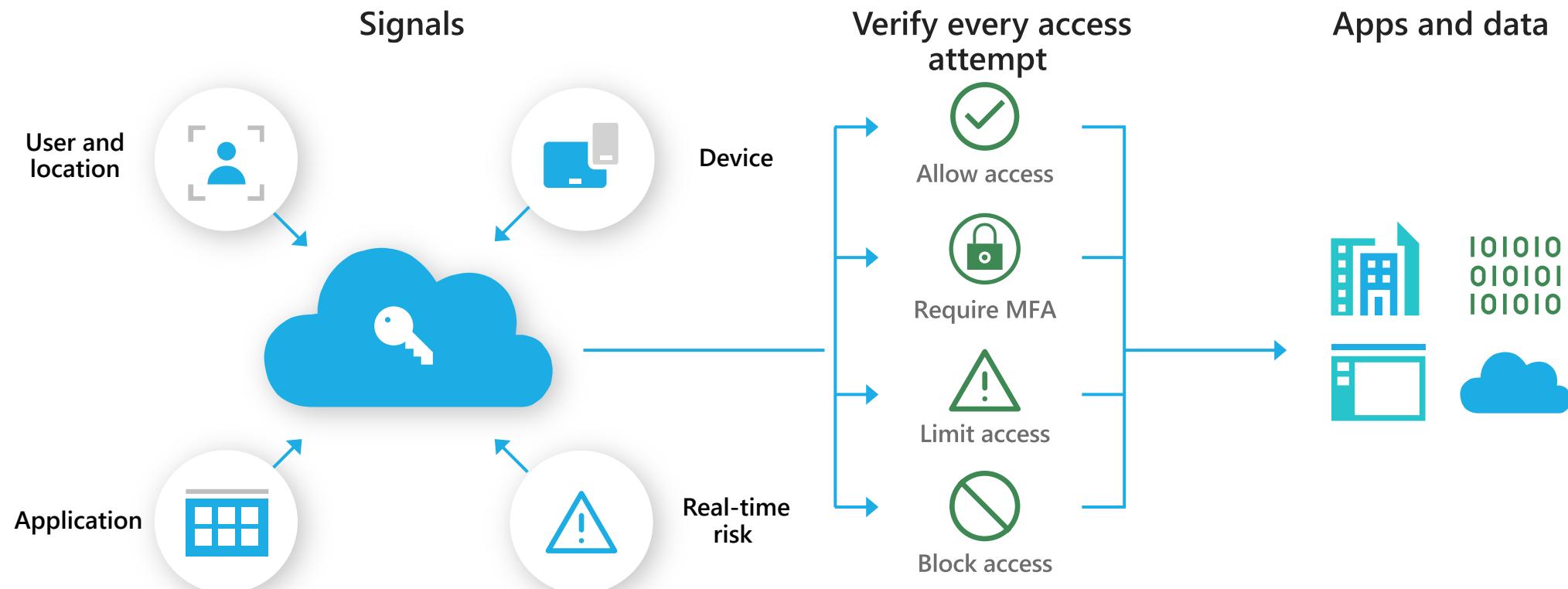
There are two types of managed identities:

- A **system-assigned identity** is tied to your application and is deleted if your app is deleted. An app can only have one system-assigned identity.
- A **user-assigned identity** is a standalone Azure resource that can be assigned to your app. An app can have multiple user-assigned identities.



# Conditional Access

Enforce smart protection policies and risk assessment to grant access to employees and partners



# Microsoft Graph

Extend Microsoft 365 experiences



Documents   Conversations

Portals

Timeline

Search

Build your experience



Web apps   Bots and agents

Device and native

Daemon apps

Workflow automation

Analytics apps

Microsoft Graph API



Your local data

Connectors



Microsoft Identity

Azure platform

Microsoft Graph data connect



# Microsoft Graph Explorer

The screenshot shows the Microsoft Graph Explorer interface. At the top, there's a navigation bar with links for Microsoft Graph, Explore, Graph Explorer, Docs, API, Learn, Developer Program, and Support. Below the navigation is a search bar and a dropdown for tenant selection.

The main area is titled "Graph Explorer" and has tabs for "Sample queries", "Resources", and "History". A search bar is present under the tabs. A message indicates "See more queries in the Microsoft Graph API Reference docs."

The left sidebar contains a tree view of available queries categorized by section:

- Getting Started (8):
  - GET my profile
  - GET my profile (beta)
  - GET my photo
  - GET my mail
  - GET list items in my drive
  - GET items trending around me
  - GET my manager
  - GET my To Do task lists
- Applications (8):
  - GET list all applications with count
  - GET search and count Service Principals with "teams" in the displayName
  - POST create a new application
  - GET retrieve application properties
  - PATCH update application properties
  - DELETE delete an application
  - GET retrieve a list of owners
  - POST create a new owner
- Batching (2)
- Compliance (beta) (10)
- Edge (4)
- Excel (7)

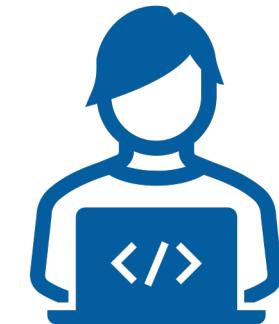
The central workspace shows a request configuration for a GET request to `https://graph.microsoft.com/v1.0/me`. It includes tabs for "Request body", "Request headers", "Modify permissions", and "Access token".

The bottom right shows a "Response preview" window with the following JSON output:

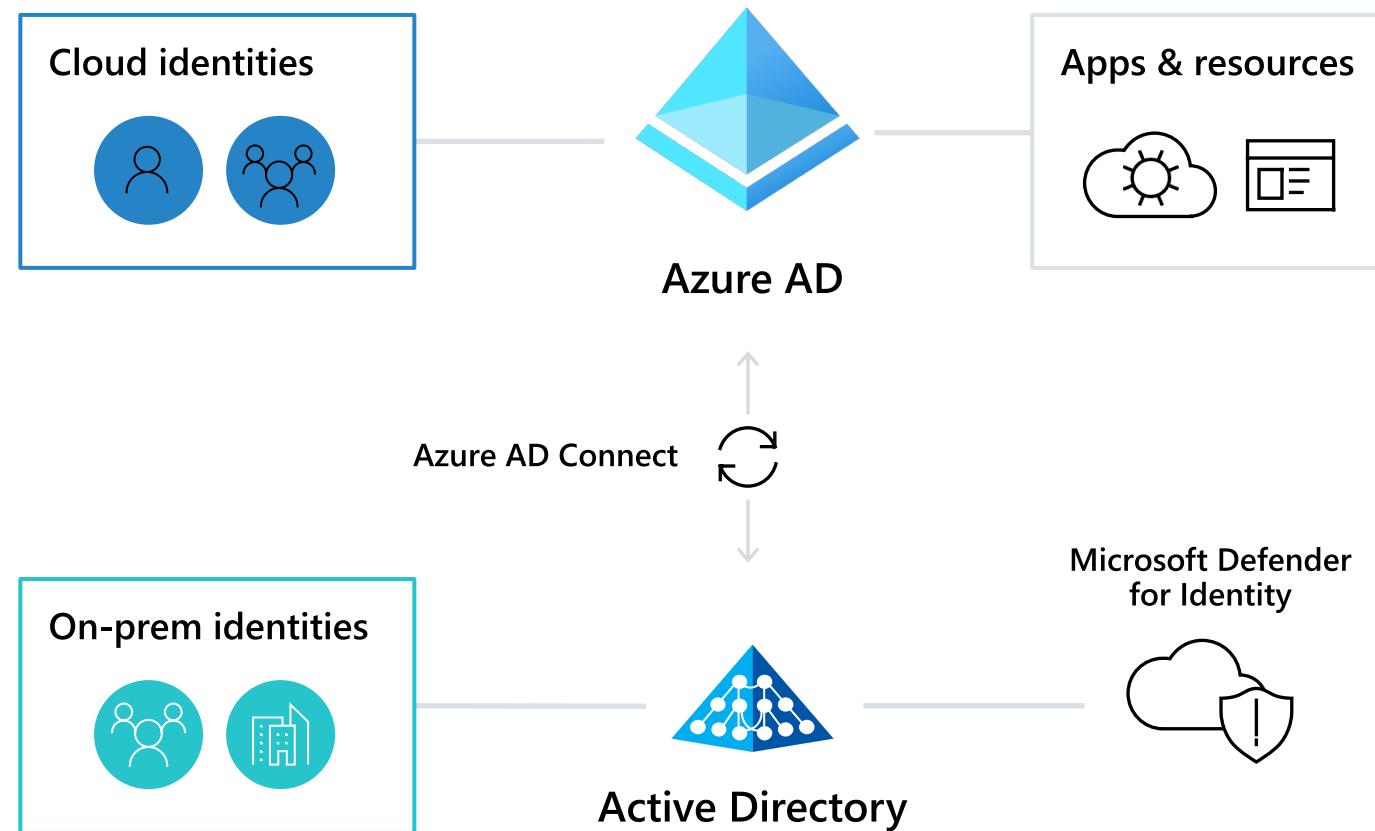
```
{ "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity", "businessPhones": [ "+1 412 555 0109" ], "displayName": "Megan Bowen", "givenName": "Megan", "jobTitle": "Associate", "mail": "Meghanbowen@msftx214355.onmicrosoft.com", "mobilePhone": null, "officeLocation": "12/111B", "preferredLanguage": "en-US", "surname": "Bowen", "userPrincipalName": "Meghanbowen@msftx214355.onmicrosoft.com", "id": "48d31887-5fad-4d73-a9f5-3c356e68a038" }
```

**DEMO**

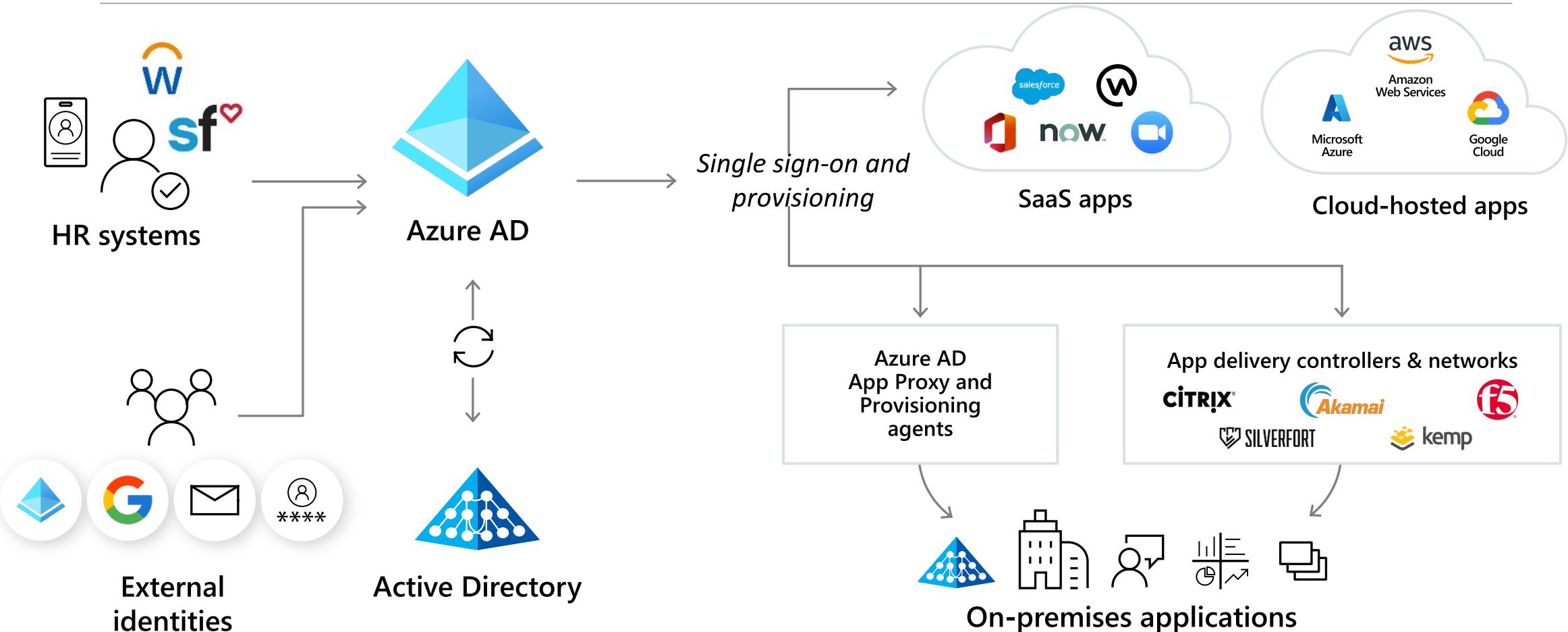
---



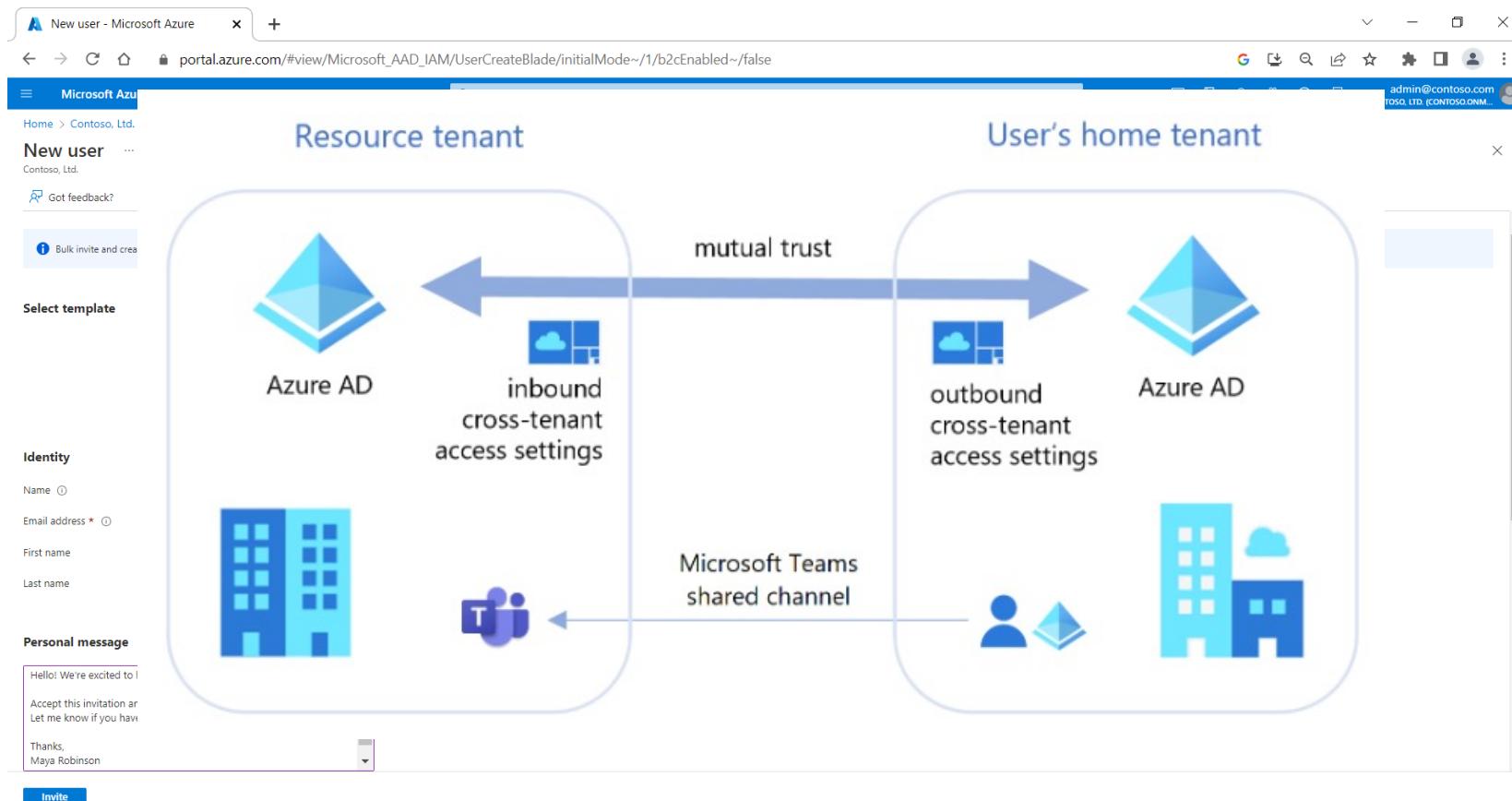
# Integration with Active Directory



# Single sign-on



# External Identities – Azure B2B



# Azure AD B2C

## Customers

Social IDs, email, or local accounts



Business & Government IDs

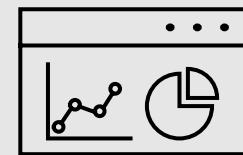


- Securely authenticate your customers using their preferred identity provider
- Capture login, preference, and conversion data for customers
- Provide branded (white-label) registration and login experiences

## Business



Apps and APIs



Analytics



External systems integration

# Azure AD B2C

---



**facebook.**

**LinkedIn**



**Google**

**twitter**



{ // }

Open standards



**amazon**

**GitHub**

**SAML**

{ JSON }

# Microsoft Identity Platform Best Practices

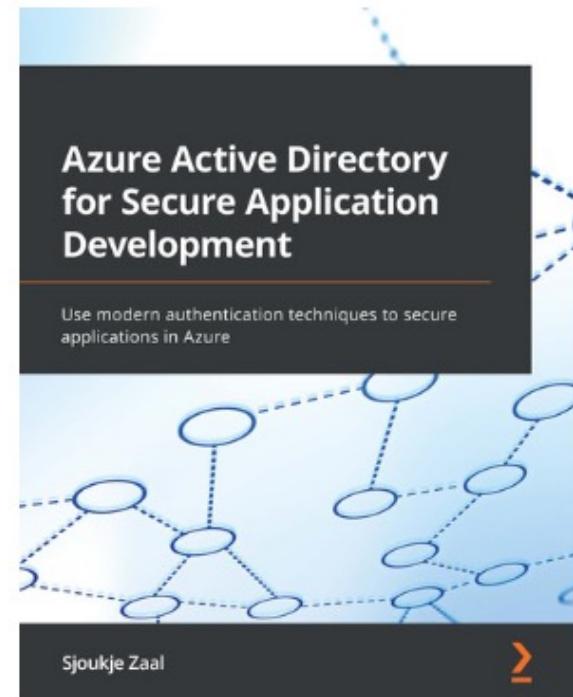
---

1. Maintain ownership of all your redirect URIs and keep the DNS records for them up-to-date.
2. Don't use wildcards(\*) in your URLs.
3. For web apps, make sure all URLs are secure and encrypted (https).
4. Review and trim all unused or unnecessary redirect URLs on a regular basis.
5. Regularly monitor the list of app registration owners.
6. Don't enable support for implicit grant flow unless explicitly required.
7. Make sure application requests the least privilege permissions.
8. Leverage MSAL instead of programming directly against OAuth 2 and Open ID.
9. Migrate existing apps from ADAL to MSAL.
10. Leverage Microsoft Graph API whenever possible.
11. .... <https://learn.microsoft.com/en-us/azure/active-directory/develop/identity-platform-integration-checklist>

# References / Further Reading

---

1. [What is the Microsoft Identity Platform?](#)
2. [OAuth 2.0](#)
3. [OpenID Conenct](#)
4. [Microsoft Identity Platform - Code Samples](#)
5. [Authentication flow support in MSAL](#)
6. [Microsoft Entra \(Azure AD\) Blog](#)
7. [Azure Roadmap - security and identity](#)



# Q&A

# Contact Information

---



<https://vaibhavgujral.com>

[@vaibhavgujral\\_](https://twitter.com/vaibhavgujral_)

<https://www.linkedin.com/in/vaibhavgujral/>

<https://www.youtube.com/@VaibhavGujral>

[vaibhav@vaibhavgujral.com](mailto:vaibhav@vaibhavgujral.com)



LinkedIn



Twitter



Email

# Slides

---



Thank  
You!