Microsoft Identity Platform for Developers

VAIBHAV GUJRAL
CLOUD ARCHITECT | MICROSOFT AZURE MVP

About me

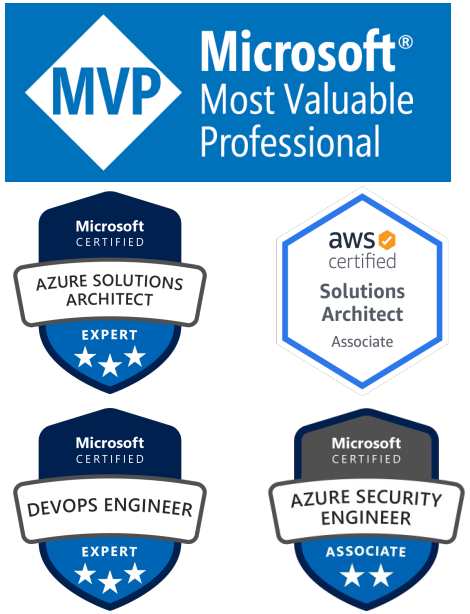
Director, Global Microsoft Cloud CoE at Capgemini

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

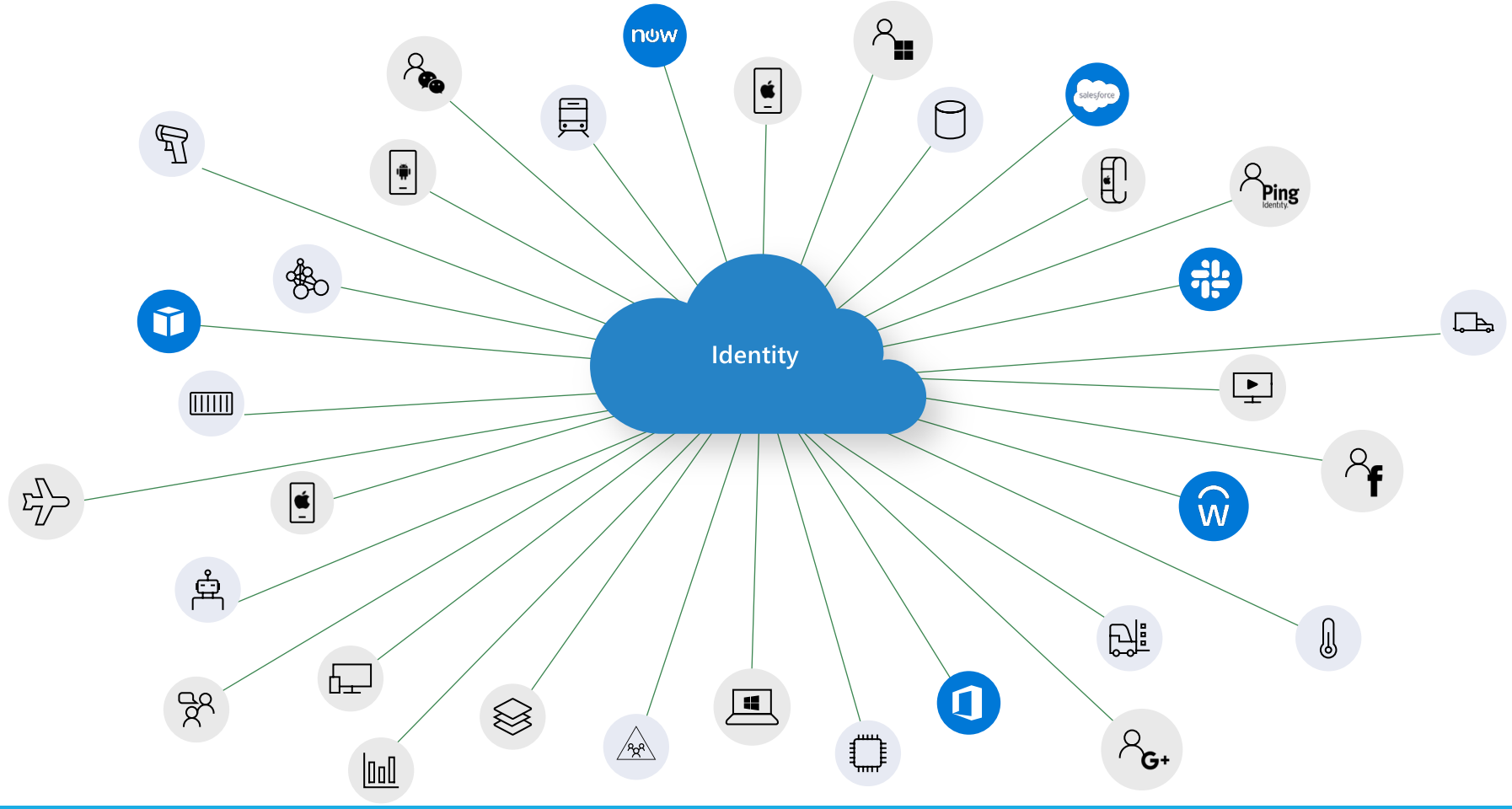
Microsoft Azure MVP since 2020

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

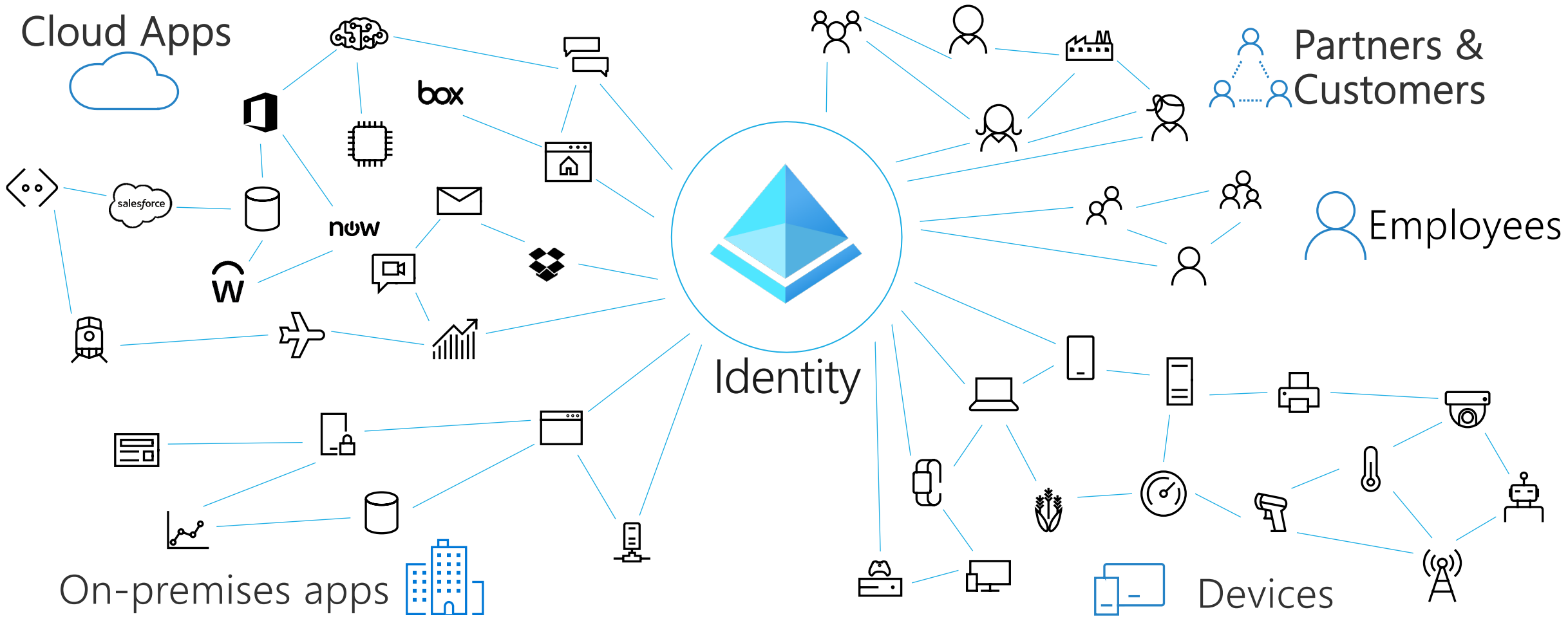
15+ cloud certifications and counting…



Identity is the control plane

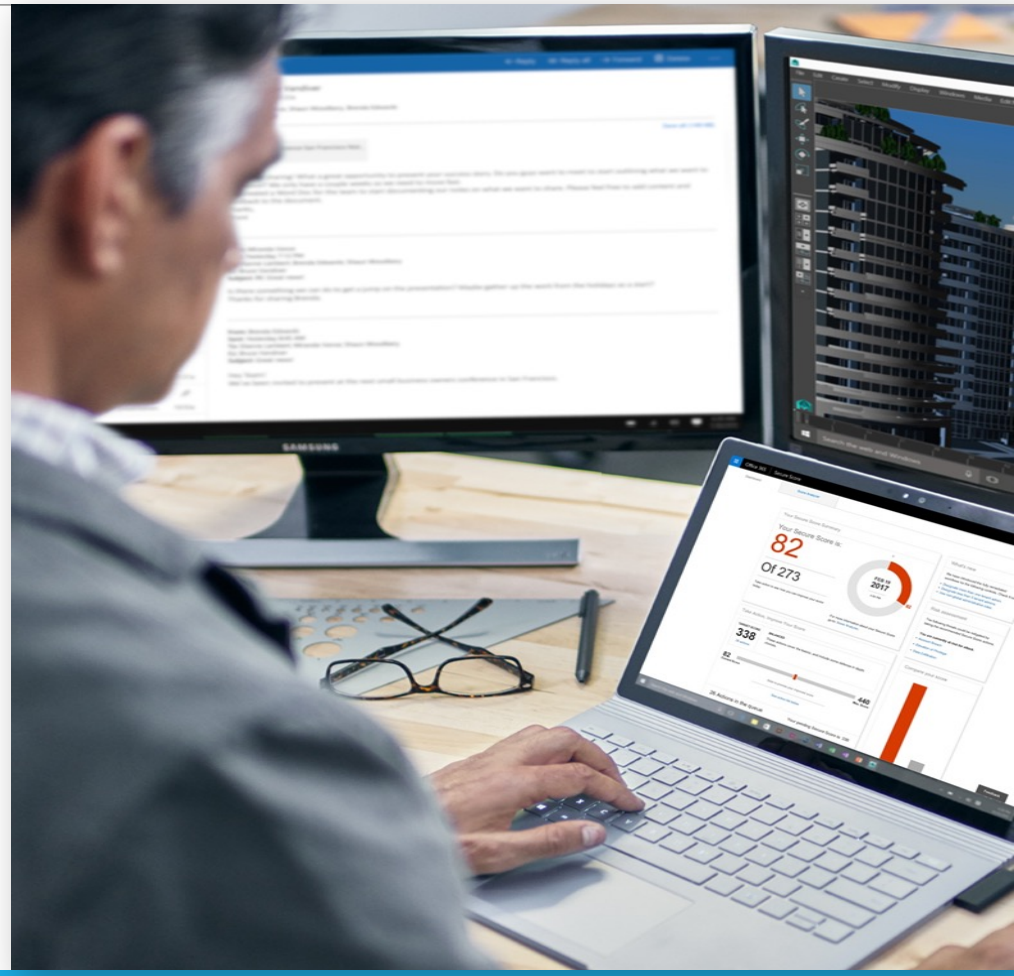


Identity is the control plane



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

SCIM

WebAuthn

Microsoft identity platform for developers

App registration
Azure portal and Microsoft Graph App API

Libraries
Microsoft Authentication Libraries
For clients and services

Web API
Web API
Including Microsoft Graph, Azure, and your own APIs

Endpoints
OIDC Certified Endpoints

Supported identities

Work and school accounts
Azure AD

Personal accounts
MSA

Customer and Partner accounts
Azure AD External Identities (includes Azure AD B2C)

Industry standards

OAuth 2 · OpenID Connect · SAML v2.0 · SCIM

Azure AD – the world's largest cloud identity service

300K+

Azure AD
Premium organizations

425M+

Azure AD
Monthly active users

30B+

Azure AD
Daily authentication requests

Azure Active Directory

Azure Active Directory

GitHub

Visual Studio

Visual Studio Code

Microsoft 365 platform

Teams

SharePoint

Exchange

...

Azure services

App Service

Azure Kubernetes Service

Functions

...

Power Automate

Power Apps

Power Virtual Agents

Azure Active Directory

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

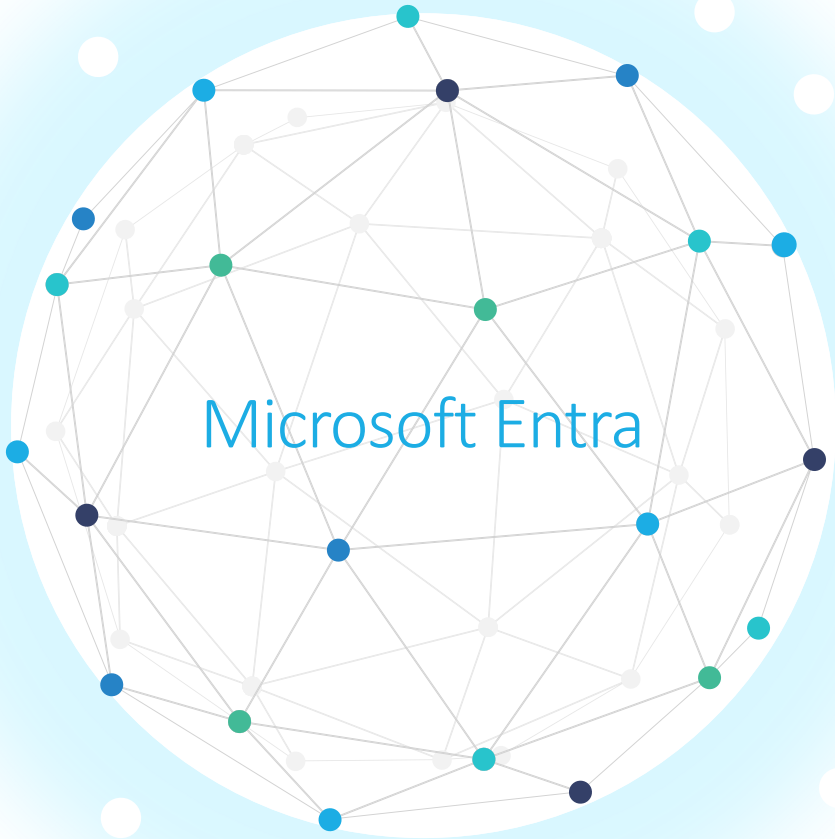
Global cloud infrastructure with secure hardware and data segregation

99.99%
New Azure AD Service Level Agreement
(in effect from April 1st, 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.

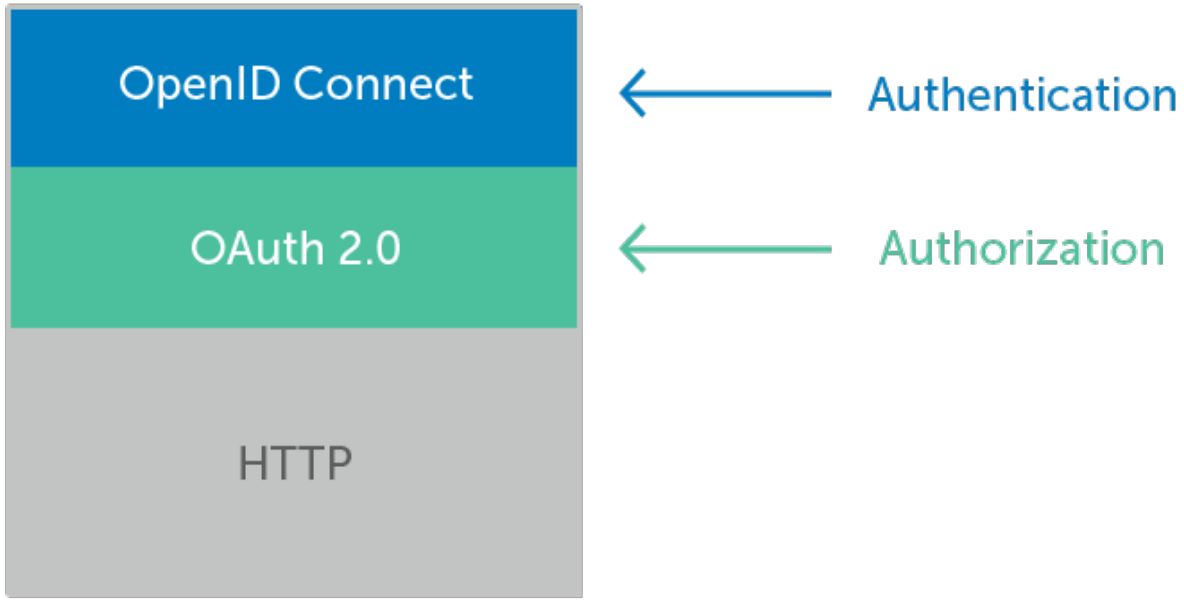
OAuth 2.0 and OpenID Connect

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

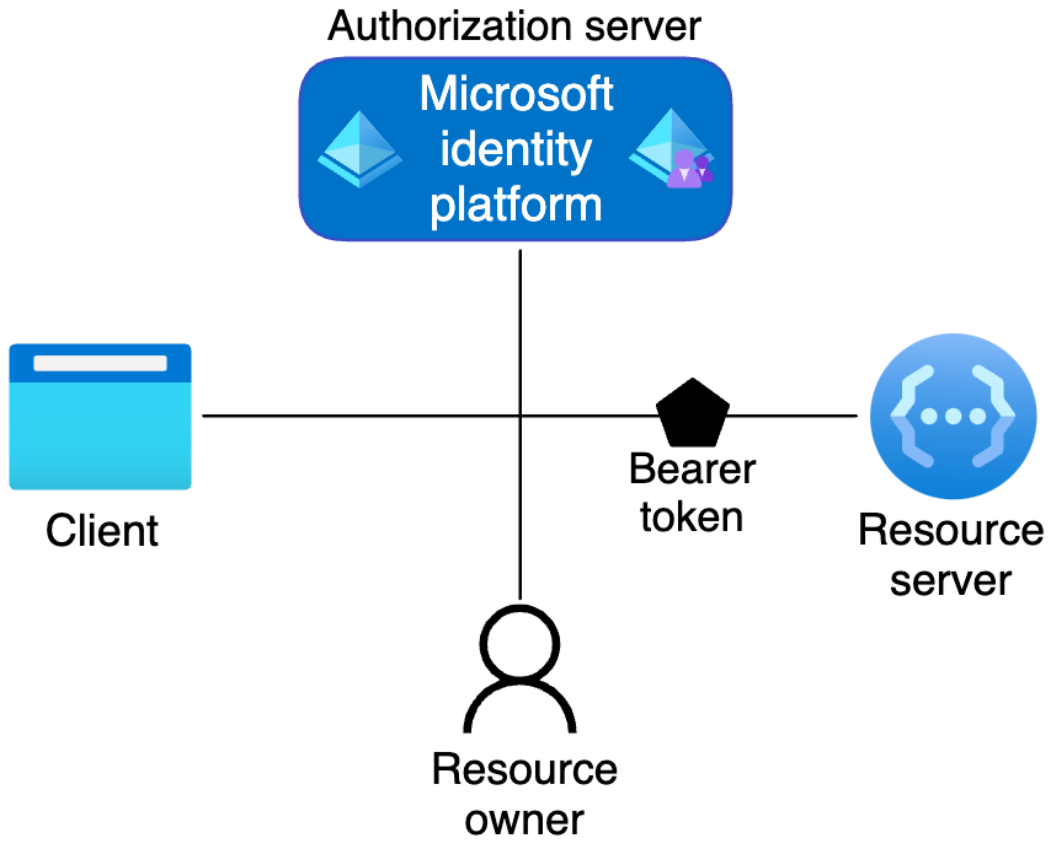
OAuth defines four roles:

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.

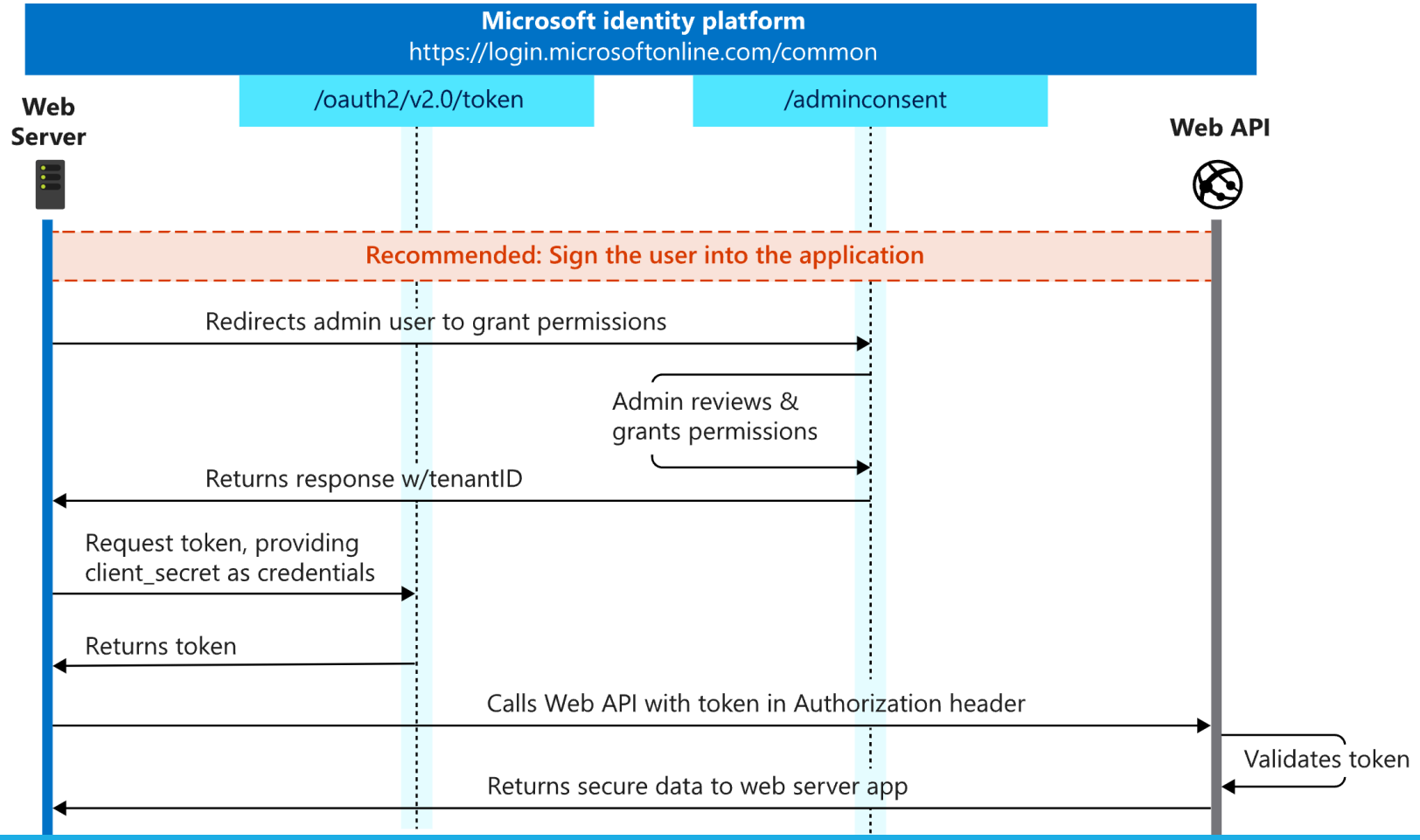
Resource server: The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.

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).

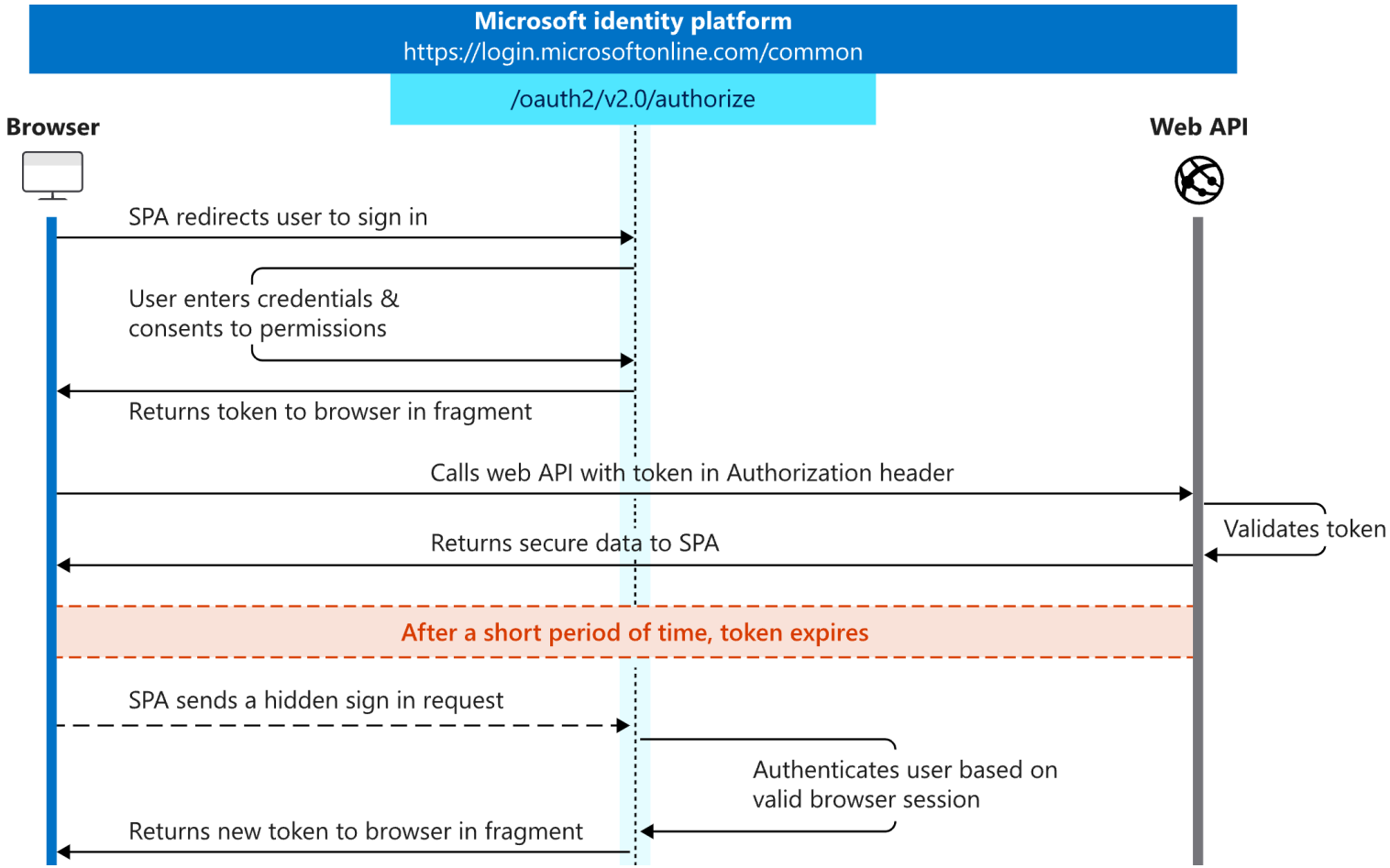
Authorization server: The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.



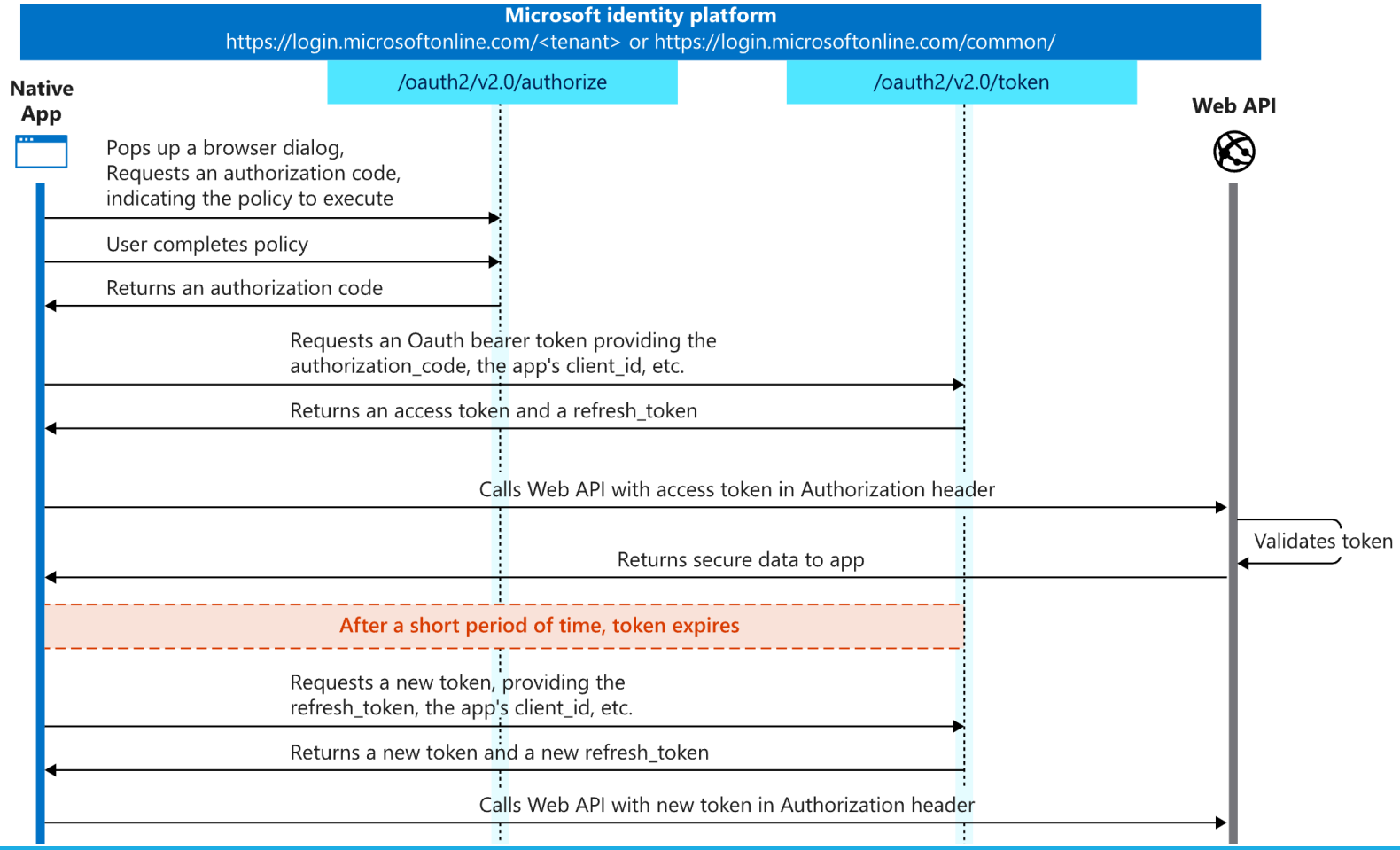
Client Credentials Flow



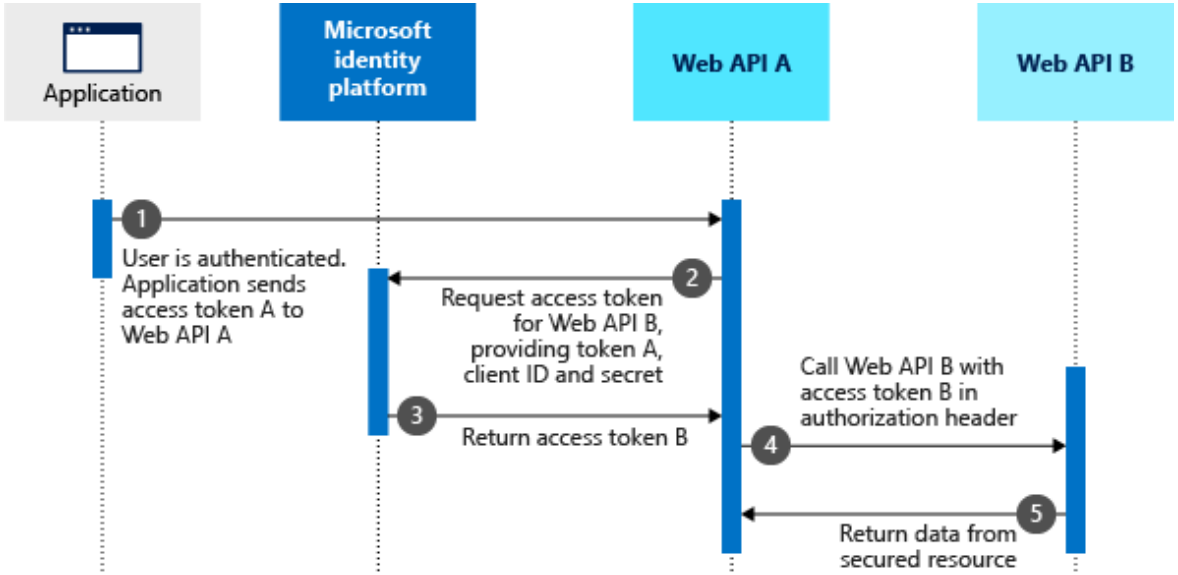
Implicit Grant Flow



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.

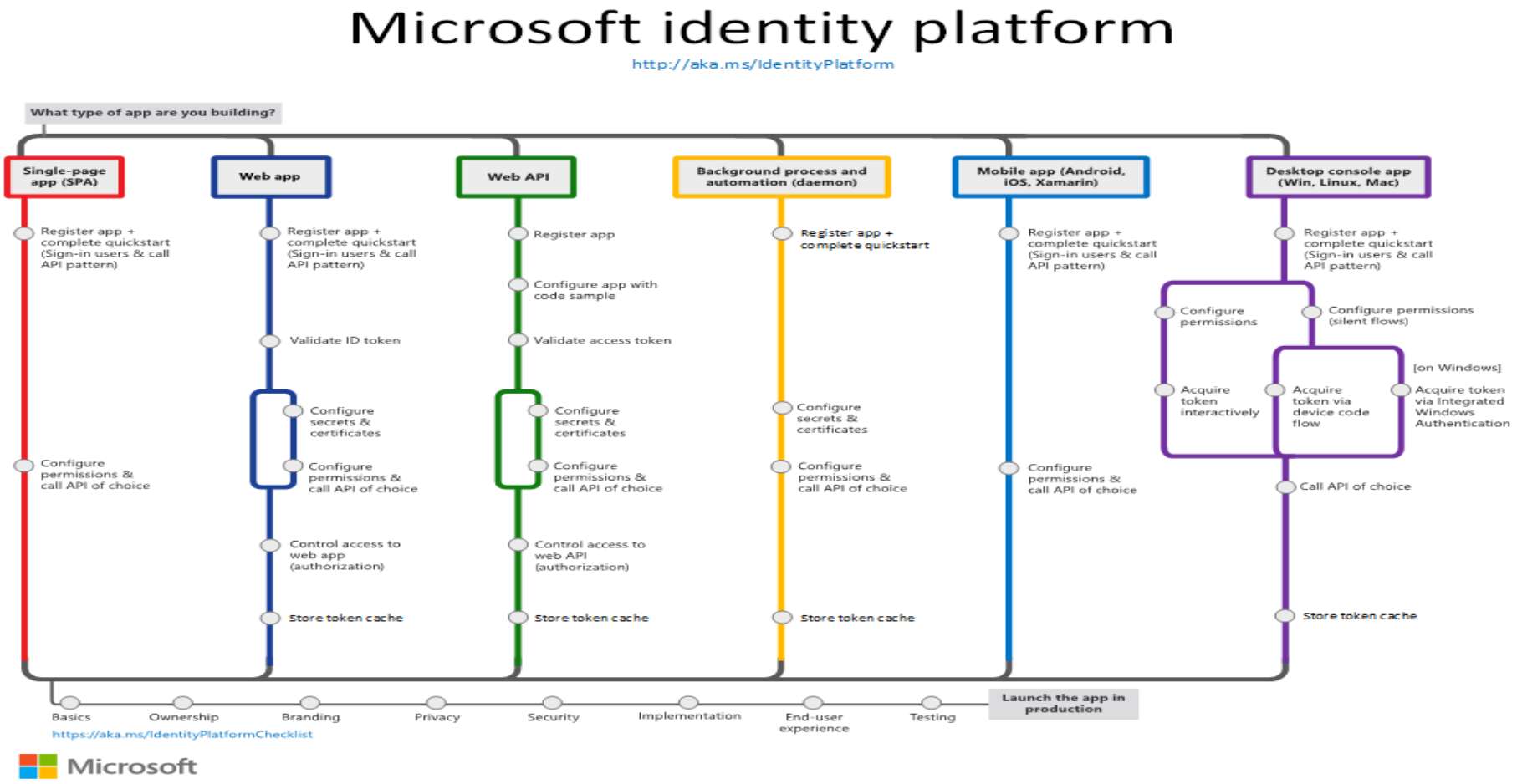
Signature - Is the raw material used to validate the token.

Example -

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Imk2bEdrM0ZaenhSY1ViMkMzbkVRN3N5SEpsWSJ9.eyJhdWQiOiI2ZTc0MTcyYi1iZTU2LTQ4NDMtOWZmNC1lNjZhMzliYjEyZTMiLCJpc3MiOiJodHRwczovL2xvZ2luLm1pY3Jvc29mdG9ubGluZS5jb20vNzJmOTg4YmYtODZmMS00MWFmLTkxYWItMmQ3Y2QwMTFkYjQ3L3YyLjAiLCJpYXQiOjE1MzcyMzEwNDgsIm5iZiI6MTUzNzIzMTA0OCwiZXhwIjoxNTM3MjM0OTQ4LCJhaW8iOiJBWFFBaS84SUFBQUF0QWFaTG8zQ2hNaWY2S09udHRSQjdlQnE0L0RjY1F6amNKR3hQWXkvQzNqRGFOR3hYZDZ3TklJVkdSZ2hOUm53SjFsT2NBbk5aY2p2a295ckZ4Q3R0djMzMTQwUmlvT0ZKNGJDQ0dWdW9DYWcxdU9UVDIyMjIyZ0h3TFBZUS91Zjc5UVgrMEtJaWpkcm1wNjlSY3R6bVE9PSIsImF6cCI6IjZlNzQxNzJiLWJlNTYtNDg0My05ZmY0LWU2NmEzOWJiMTJlMyIsImF6cGFjciI6IjAiLCJuYW1lIjoiQWJlIExpbmNvbG4iLCJvaWQiOiI2OTAyMjJiZS1mZjFhLTRkNTYtYWJkMS03ZTRmN2QzOGU0NzQiLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJhYmVsaUBtaWNyb3NvZnQuY29tIiwicmgiOiJJIiwic2NwIjoiYWNjZXNzX2FzX3VzZXIiLCJzdWIiOiJlS1pwZmFIeVdhZGVPb3VZbGl0anJJLUtmZlRtMjIyWDVyclYzeERxZktRIiwidGlkIjoiNzJmOTg4YmYtODZmMS00MWFmLTkxYWItMmQ3Y2QwMTFkYjQ3IiwidXRpIjoiZnFpQnFYTFBqMGVRYTgyUy1JWUZBQSIsInZlciI6IjIuMCJ9.pj4N-w_3Us9DrBLfpCt

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

App Types


Microsoft identity platform
http://aka.ms/IdentityPlatform
What type of app are you building?
Single-page app (SPA)
Web app
Web API
Background process and automation (daemon)
Mobile app (Android, iOS, Xamarin)
Desktop console app (Win, Linux, Mac)
Register app + complete quickstart (Sign-in users & call API pattern)
Register app
Register app + complete quickstart
Configure app with code sample
Validate ID token
Validate access token
Configure secrets & certificates
Configure permissions & call API of choice
Control access to web app (authorization)
Control access to web API (authorization)
Store token cache
Configure permissions
Configure permissions (silent flows)
Acquire token interactively
Acquire token via device code flow
[on Windows]
Acquire token via Integrated Windows Authentication
Call API of choice
Basics
Ownership
Branding
Privacy
Security
Implementation
End-user experience
Testing
Launch the app in production
https://aka.ms/IdentityPlatformChecklist
Microsoft

Microsoft Authentication Libraries (MSAL)

JavaScript

.NET

Android

iOS

Python

Java

Angular

Microsoft Identity Web

Simplify sign-in with MSAL

Develop in your favorite language

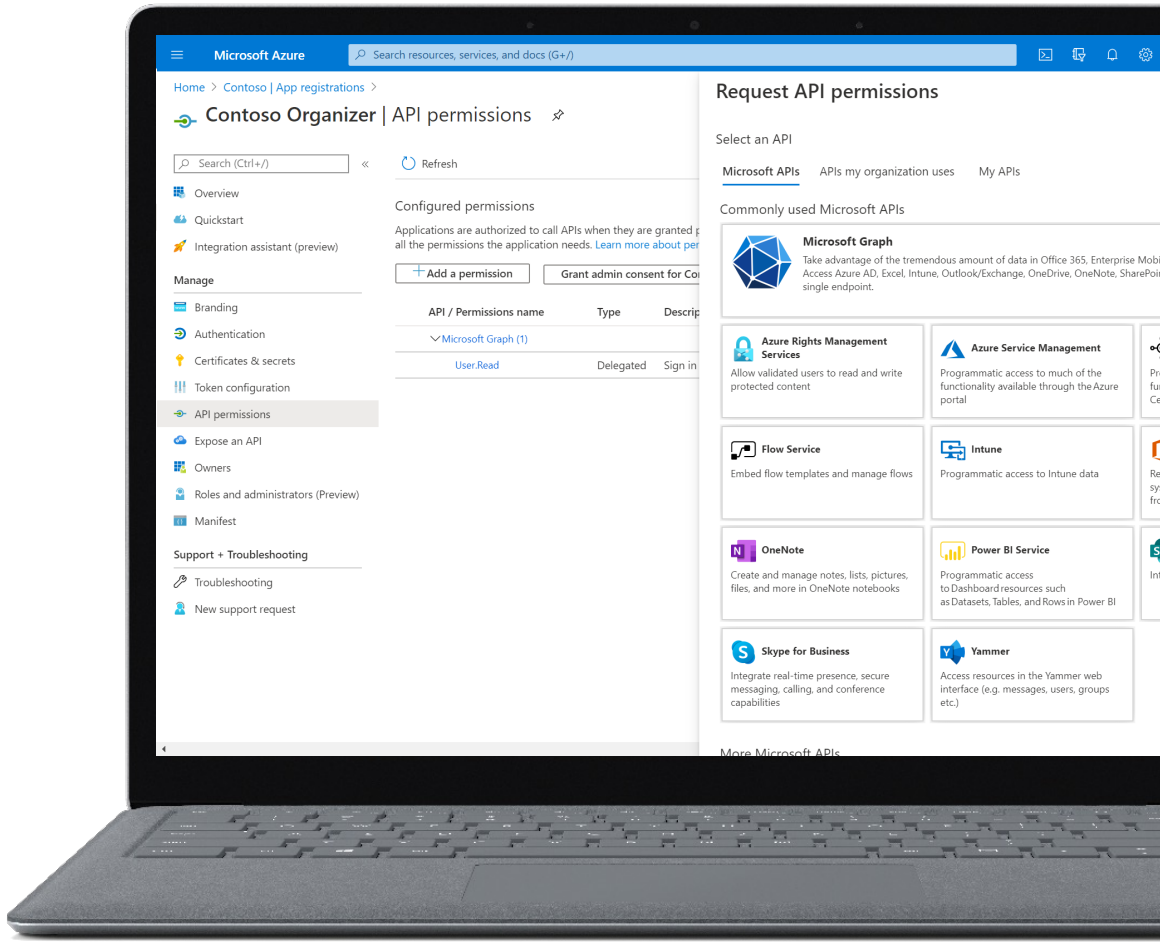
Best in class authentication libraries that work with your platform or language of choice or use our OIDC certified endpoint.

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.



Application Registration

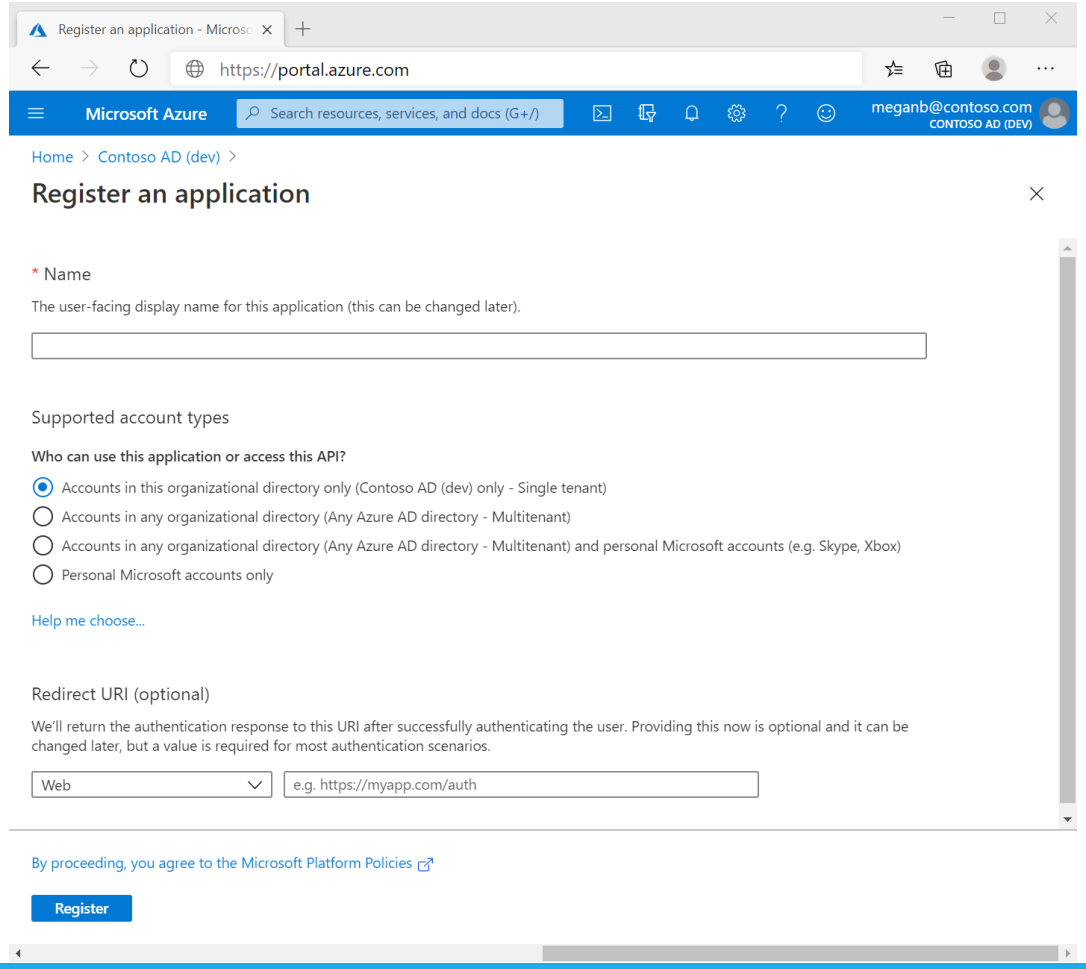
To leverage Microsoft identity platform for identity and access management (IAM), you must first register your application.

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.

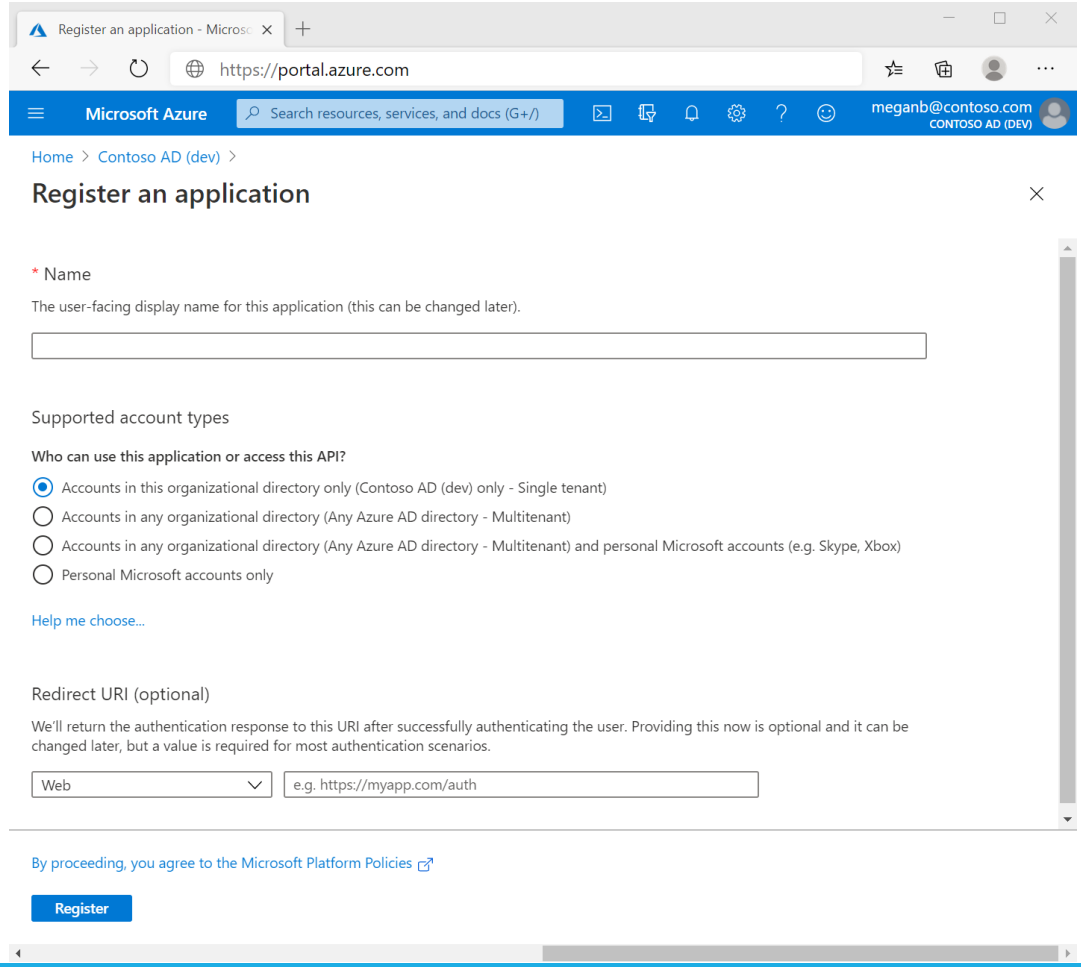
Two of the most commonly referenced app registration settings are:

- **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
Accounts in this organizational directory only	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.
Accounts in any organizational directory	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.
Accounts in any organizational directory and personal Microsoft accounts	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.
Personal Microsoft accounts	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.



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.

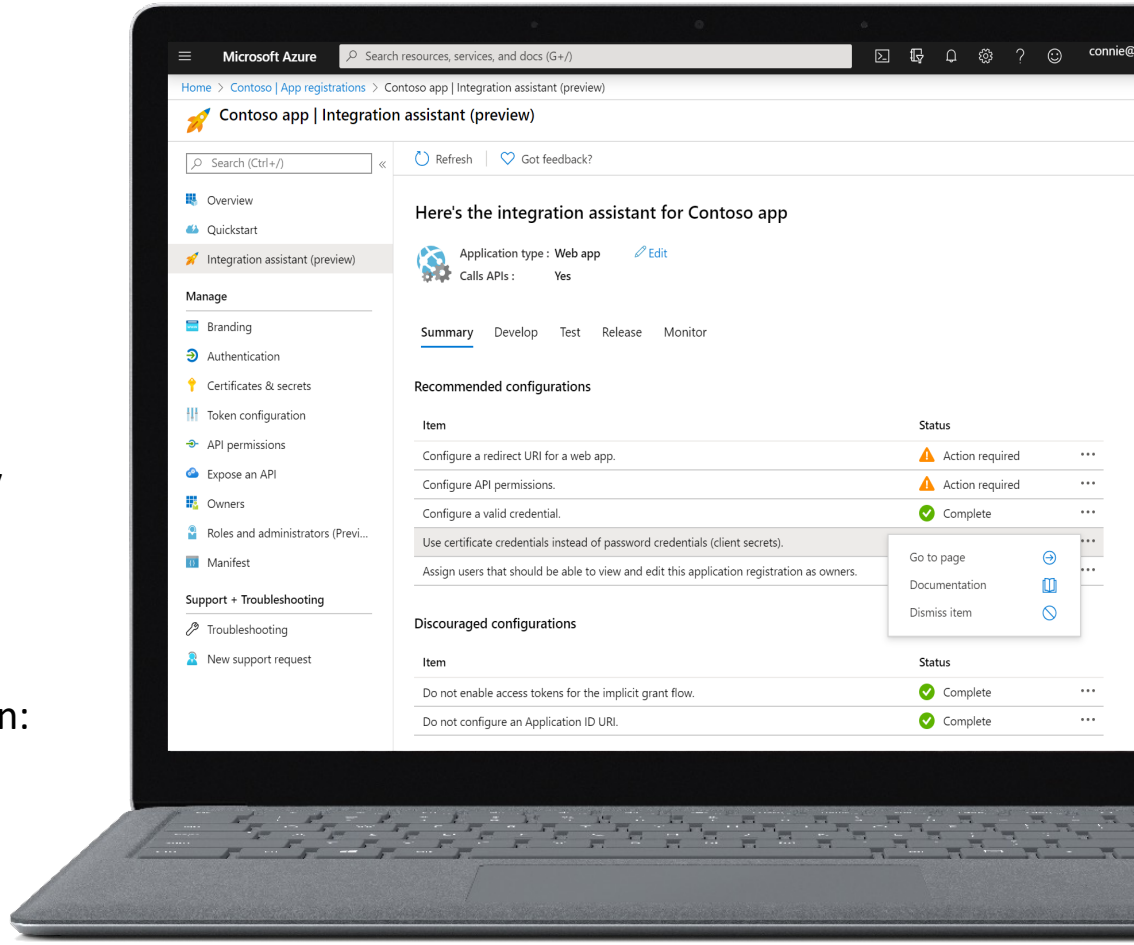
Secure your app registration

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



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.

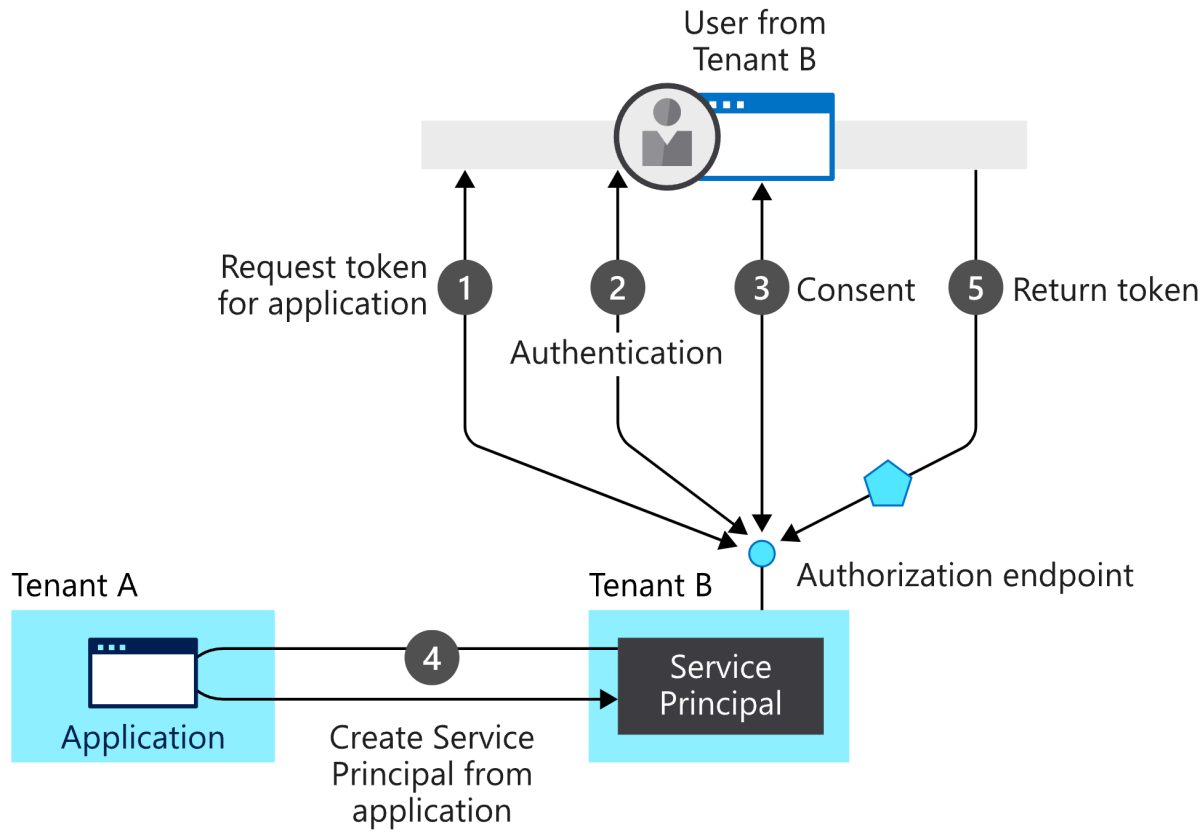
App Registration vs Enterprise Application

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.

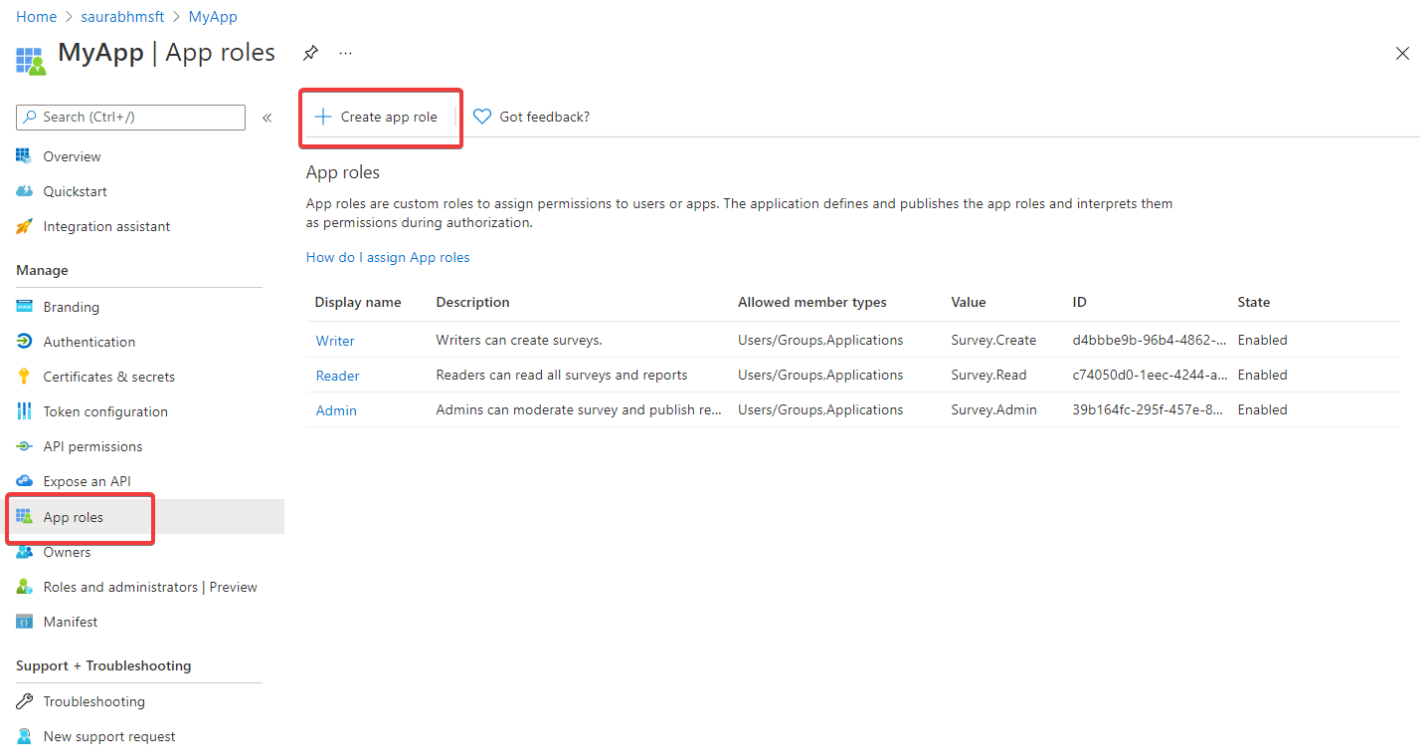


App Roles

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.

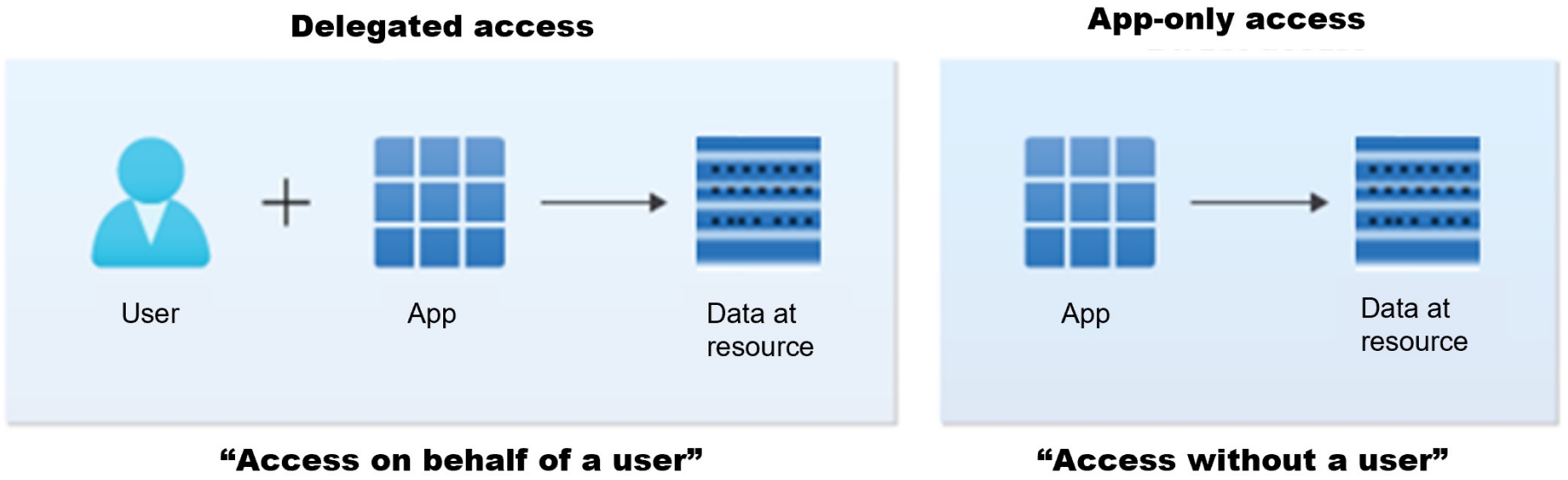


Delegated vs Application Permissions

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.

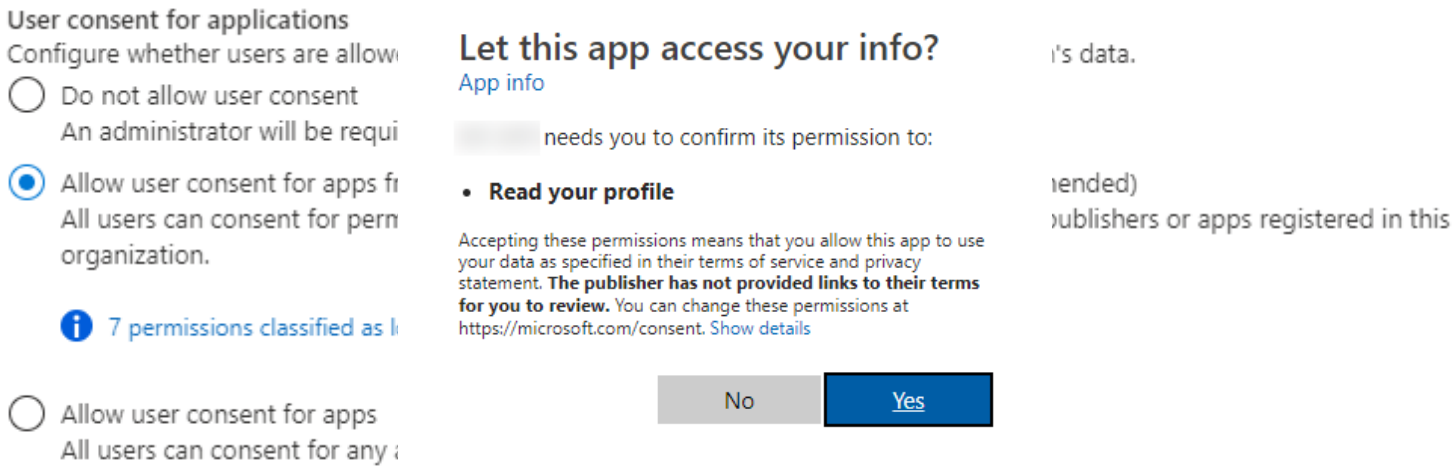


User Consent

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

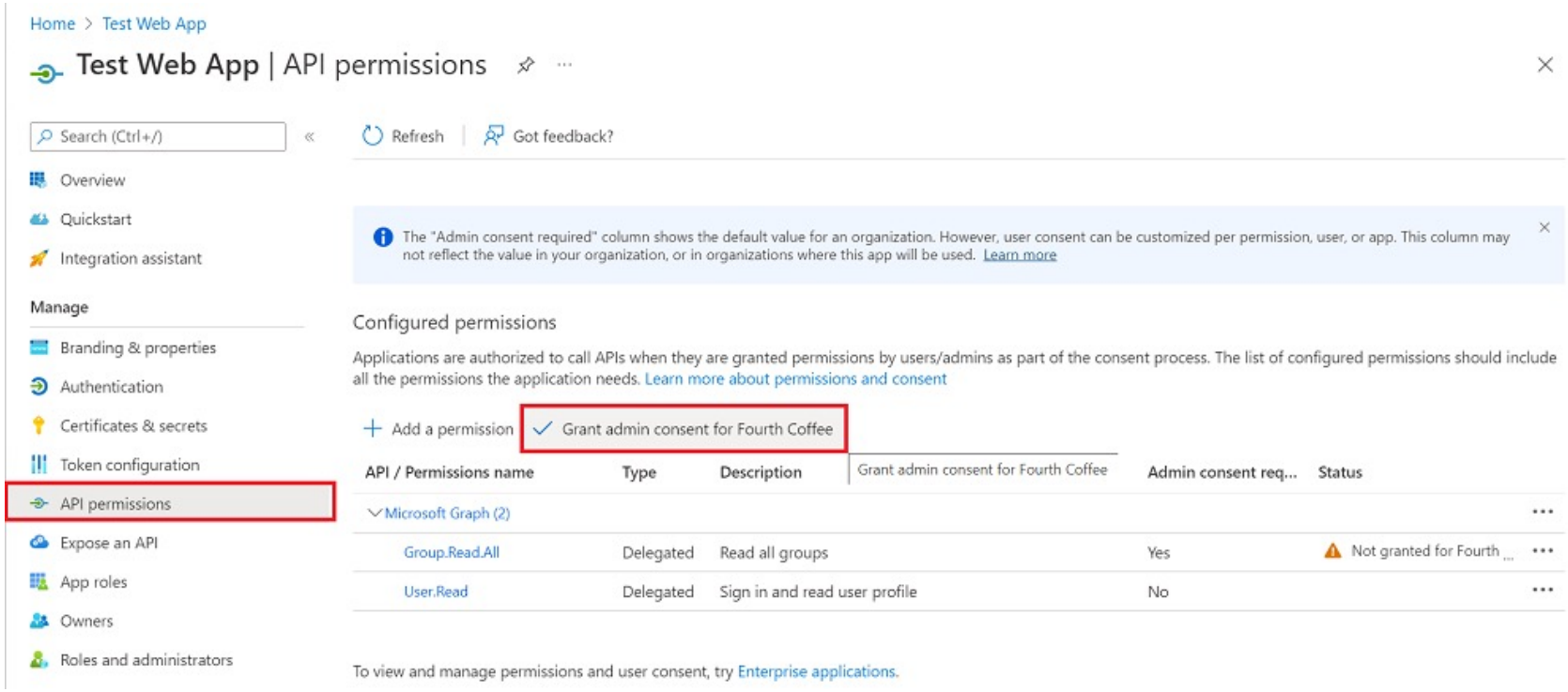
Different permissions allow different levels of access. By default, all users are allowed to consent to applications for permissions that don't require administrator consent.

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

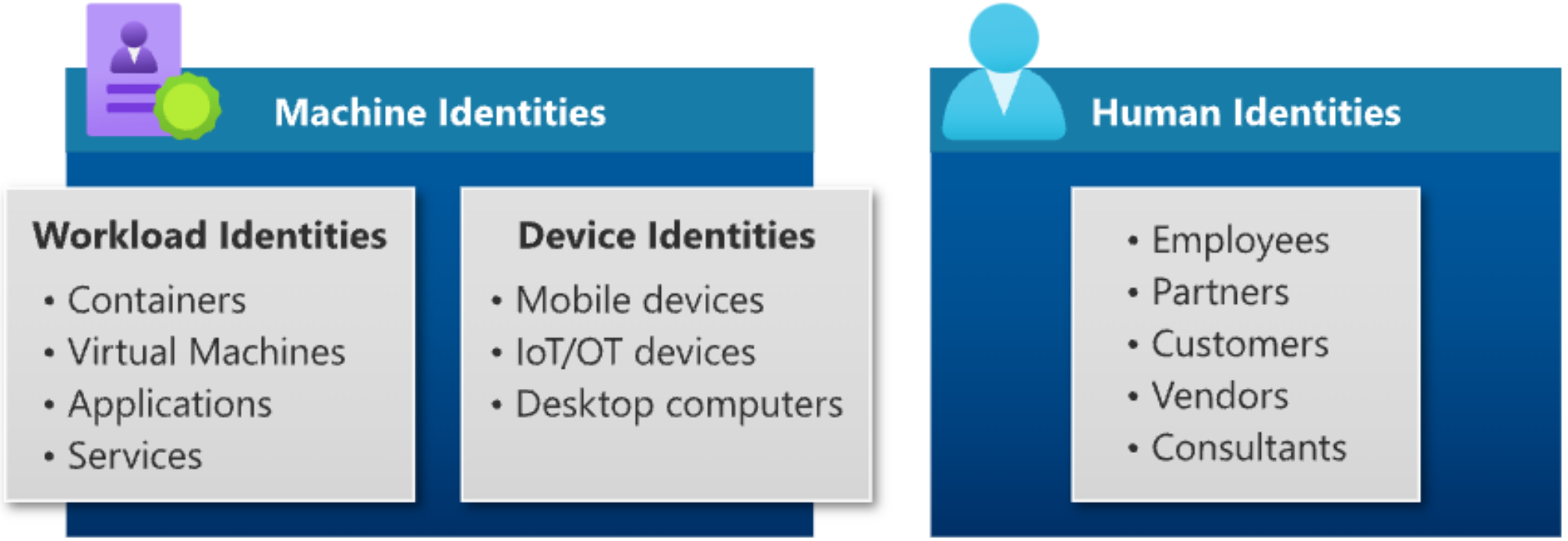


Tenant-wide admin consent

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).



Workload Identities



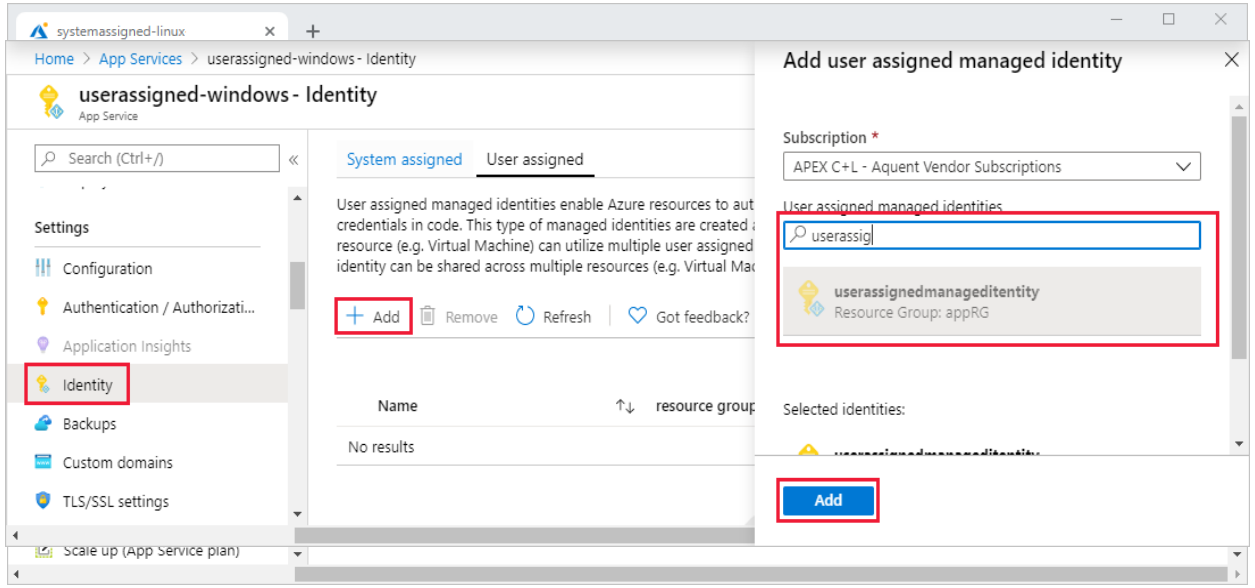
Managed Identities

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.

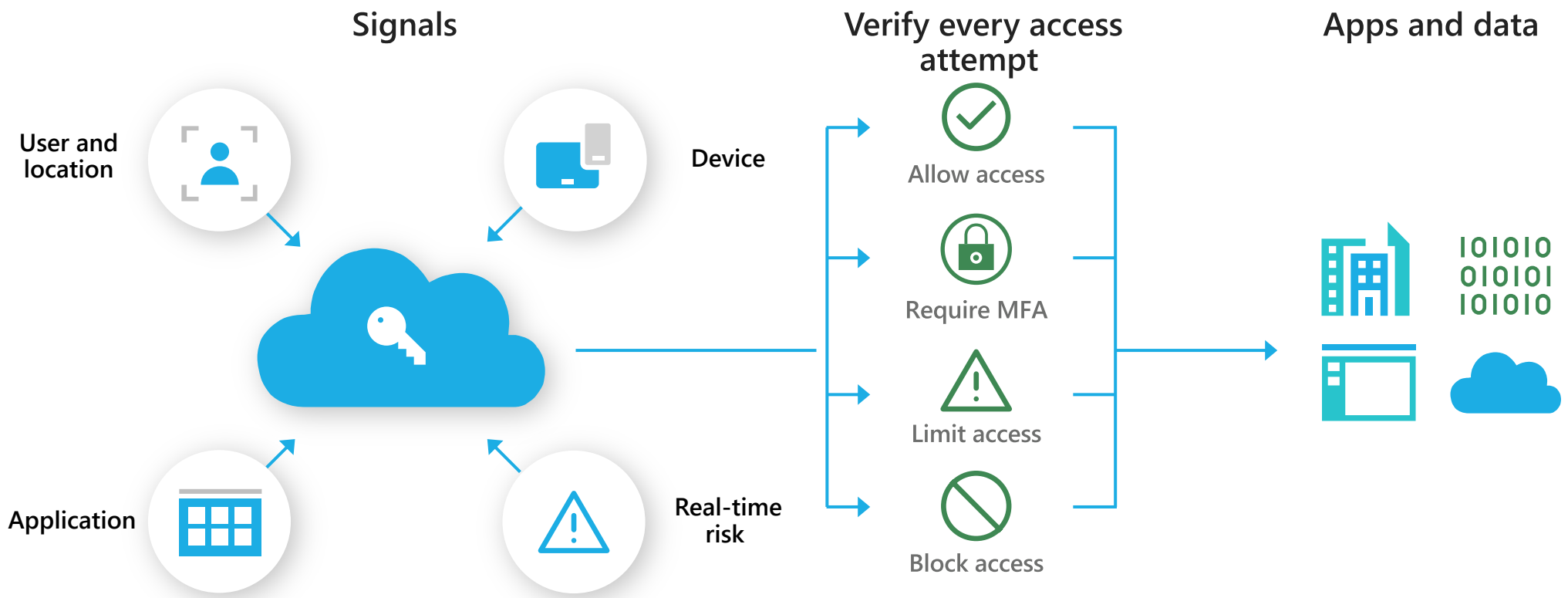
There are two types of managed identities:

- 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.

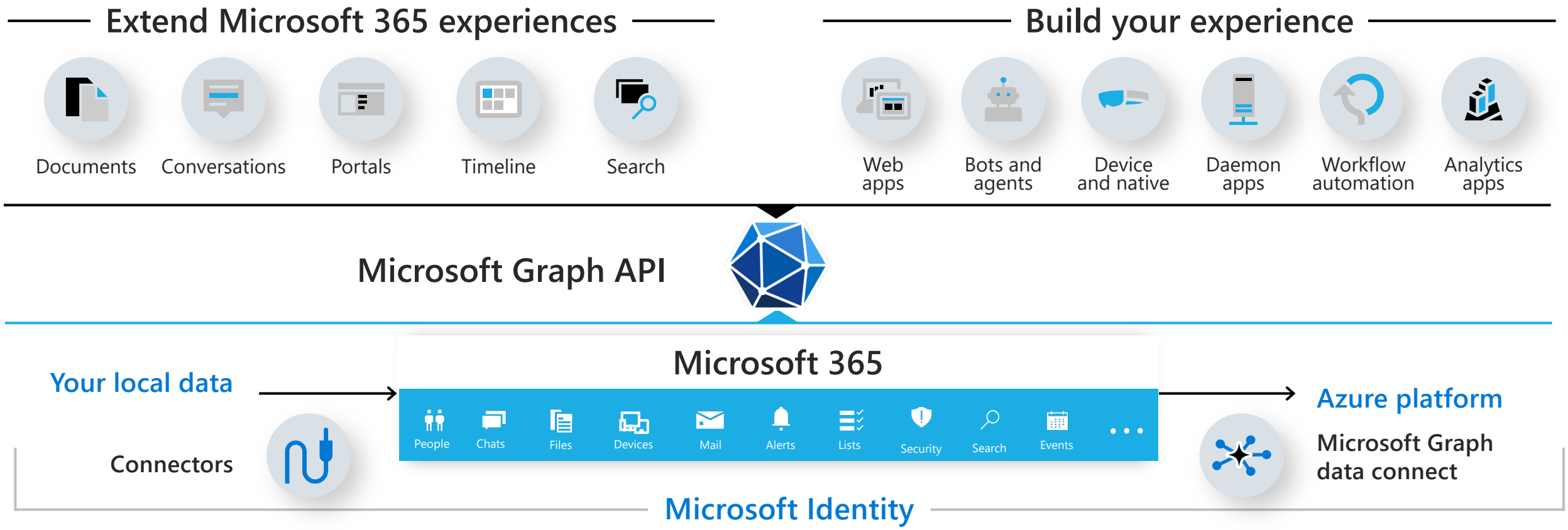


Conditional Access

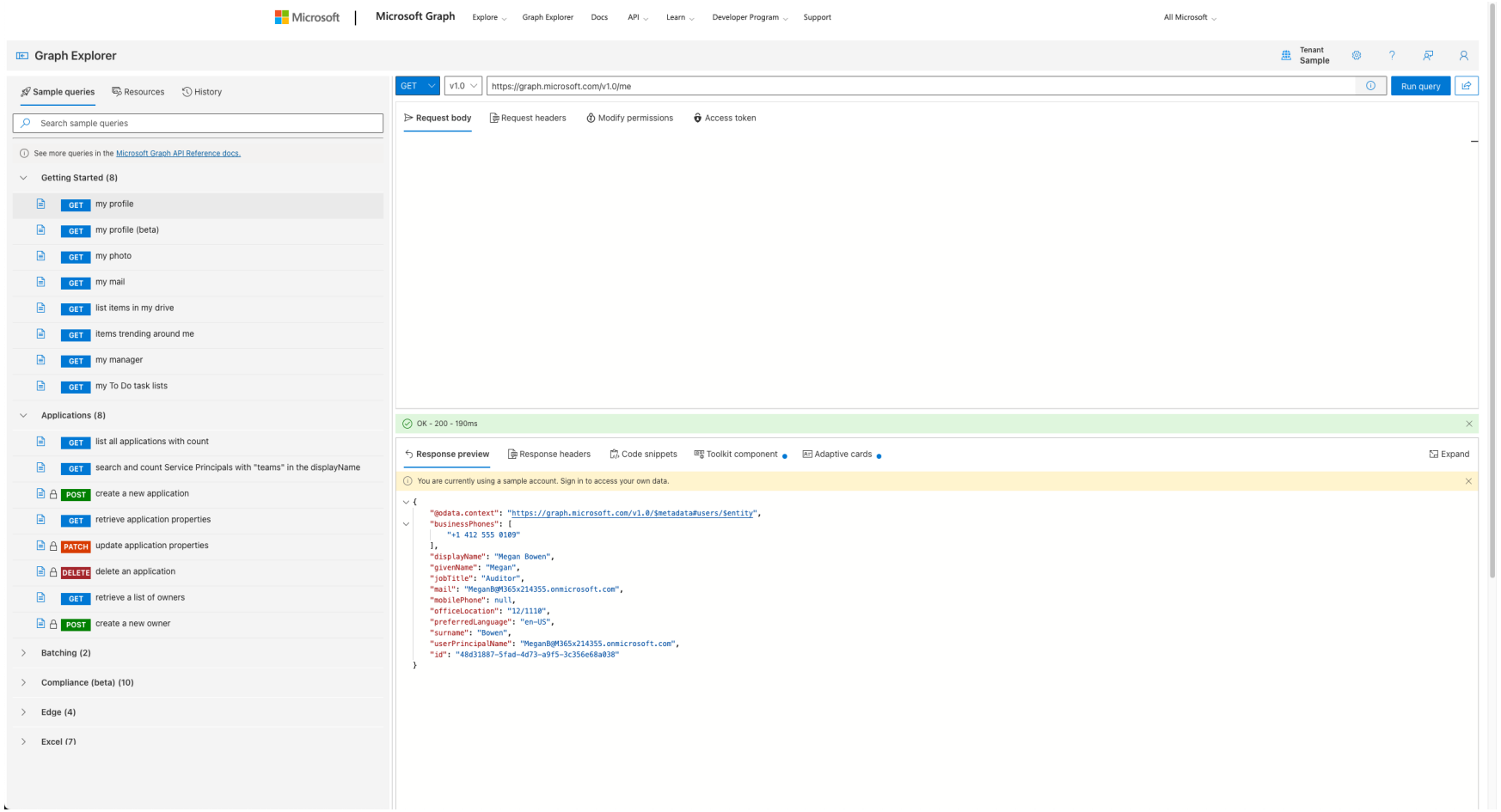
Enforce smart protection policies and risk assessment to grant access to employees and partners



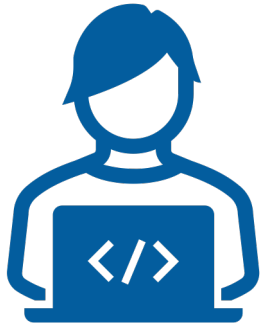
Microsoft Graph



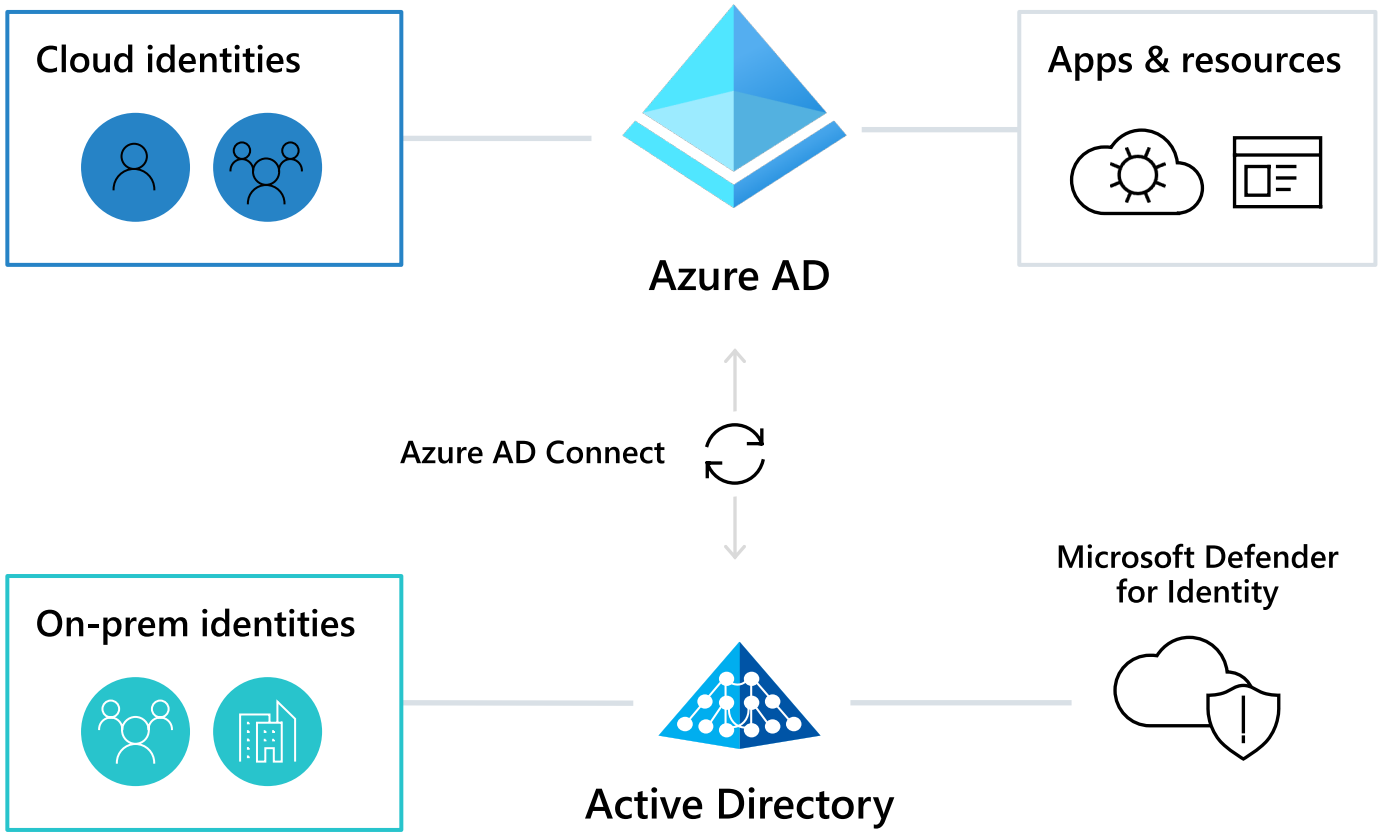
Microsoft Graph Explorer



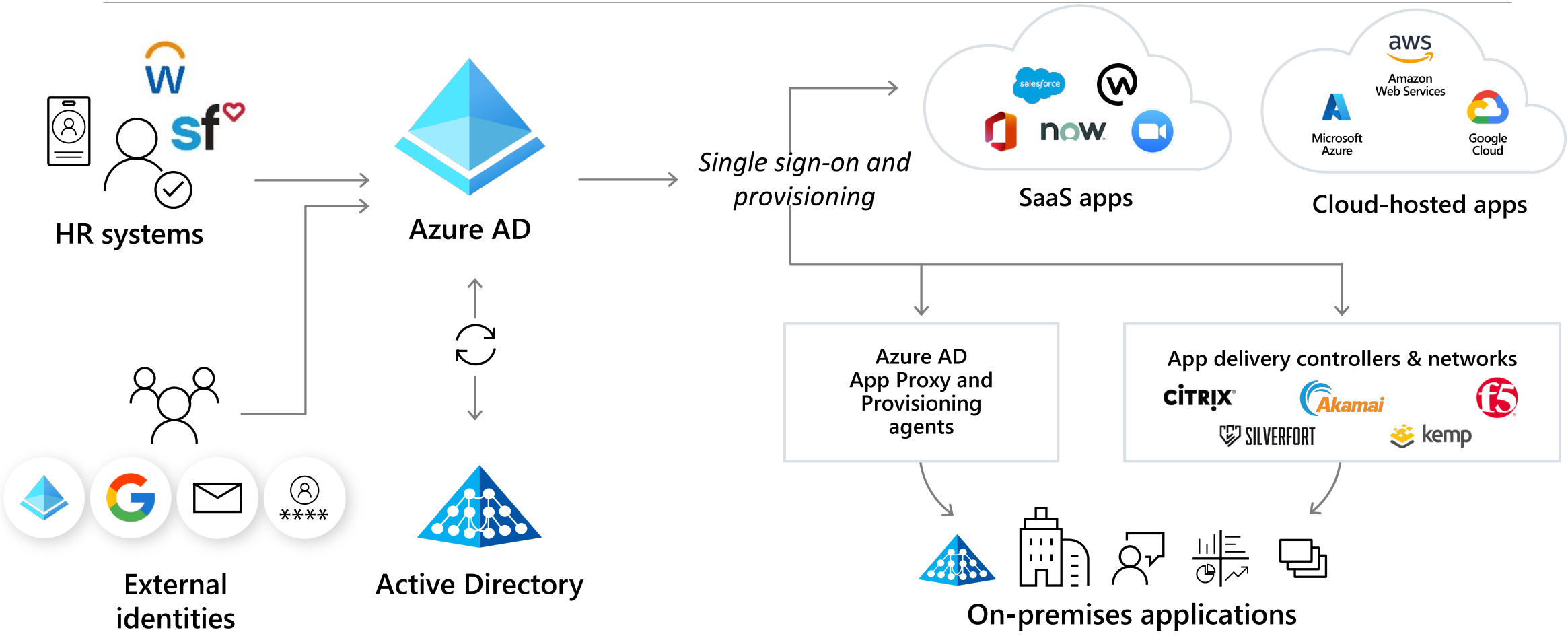

DEMO



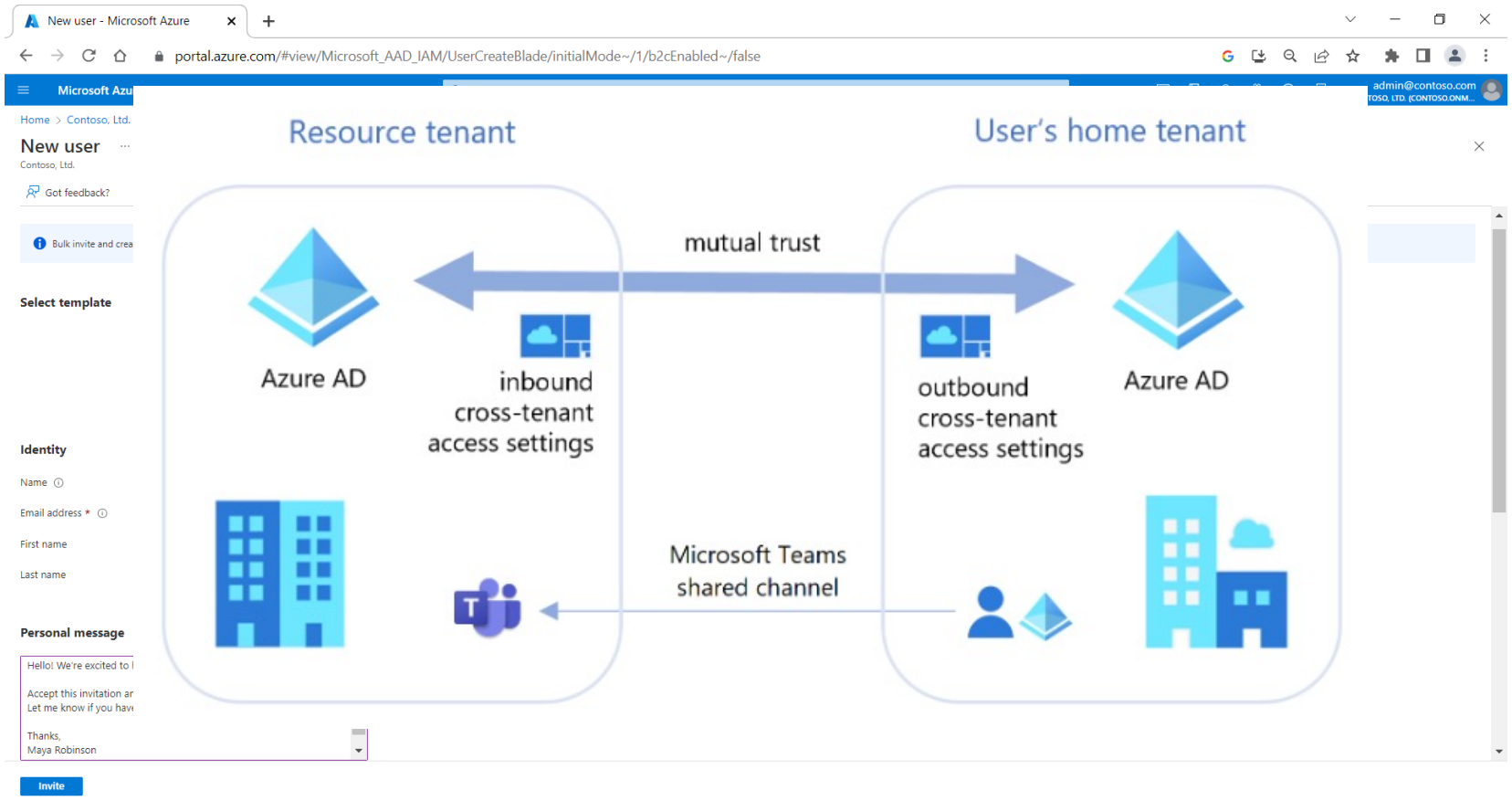
Integration with Active Directory



Single sign-on



External Identities – Azure B2B



Azure AD B2C

Customers

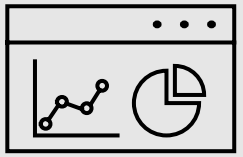
Social IDs, email, or local accounts

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

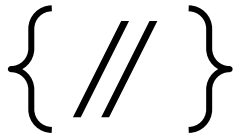
Apps and APIs



Analytics

External systems integration

Azure AD B2C



Open standards

{ JSON }

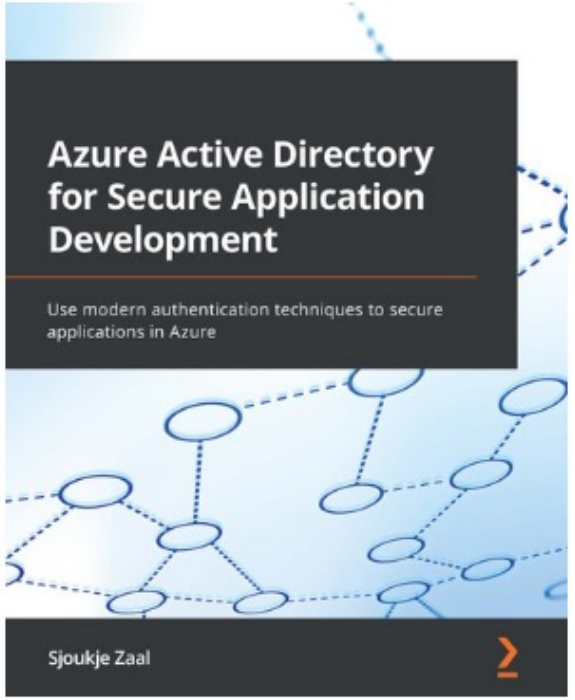
Microsoft Identity Platform Best Practices

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 URIs.
3. For web apps, make sure all URIs are secure and encrypted (https).
4. Review and trim all unused or unnecessary redirect URIs 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://www.linkedin.com/in/vaibhavgujral/

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

vaibhav@vaibhavgujral.com

LinkedIn

Twitter

Email

Slides


Thank
You!