Azure Identity and Access — Detailed Reference (Topics 34-39)

Topic 34: Microsoft Entra ID (formerly Azure AD)

Cloud-based identity and access management. The backbone of Azure security. Everything starts with identity.

What Entra ID Does

- **Authentication:** Verify who you are (username + password + MFA)
- **Authorization:** Determine what you can access (RBAC, app roles, Conditional Access)
- **Single Sign-On (SSO):** One login for all cloud apps (M365, Azure, SaaS apps)
- **Multi-Factor Authentication (MFA):** Additional verification beyond password
- **Conditional Access:** Policies that control access based on conditions (location, device, risk)
- **Device management:** Register and manage devices (Entra-joined, hybrid-joined)
- **B2B collaboration:** Invite guest users from other organizations
- **B2C:** Customer identity management (for customer-facing apps)

Entra ID Editions

Edition	Cost	Key Features
Free	Free	SSO for up to 10 apps per user, MFA (basic — security defaults), cloud-only users, basic group management
Microsoft Entra ID P1	~$6/user/month	Conditional Access, PIM (basic), dynamic groups, self-service password reset, Entra Connect Health, group-based licensing
Microsoft Entra ID P2	~$9/user/month	P1 + Identity Protection (risk-based MFA), PIM with approvals and access reviews, Entra Privileged Access Management

P2 is required for most enterprise security features. Budget for it. Identity Protection and PIM with approvals are worth the $3/user/month upgrade.

Free vs P1 vs P2 — Critical Differences:

Feature	Free	P1	P2
MFA	Security defaults only (MFA for all users, no granularity)	Conditional Access (granular MFA policies)	Same as P1 + risk-based MFA
Conditional Access	No	Yes	Yes
Dynamic groups	No	Yes	Yes
PIM	No	Basic (permanent + time-bound assignments)	Full (approvals, access reviews, just-in-time)
Identity Protection	No	No	Yes (sign-in risk, user risk detection)
Self-service password reset	No	Yes (with writeback)	Yes
Access Reviews	No	No	Yes (periodic review of group/app access)

Entra ID vs On-Prem Active Directory

Feature	On-Prem AD	Entra ID
Protocol	LDAP, Kerberos, NTLM	OAuth 2.0, OpenID Connect, SAML
Management	You manage domain controllers (install, patch, backup)	Microsoft manages (SaaS)
GPOs	Yes (Group Policy Objects — deep OS control)	No (use Intune/MDM for device management)
Machine trust	Domain join	Entra ID join / hybrid join / Entra registered
Schema	Extensible (add custom attributes)	Not extensible (use extension attributes)
Fine-grained auth	Kerberos delegation, constrained delegation	OAuth scopes, app roles, on-behalf-of flow
Query	LDAP queries, ADSI	Microsoft Graph API
Primary use	Internal network authentication, legacy apps	Cloud and modern authentication, SSO, B2B
DNS	DNS server, integrates with DHCP	DNS domain verification only

They are NOT the same thing. They serve different purposes. Most enterprises use both: - On-prem AD for legacy/Kerberos workloads, GPOs, internal DNS - Entra ID for cloud/modern workloads, SSO, Conditional Access - Connected via Entra Connect for hybrid identity

Users in Entra ID

Type	What it Is	How Created	Use Case
Cloud-only user	Exists only in Entra ID	Created in Entra ID portal, Microsoft 365 admin center, or Graph API	New cloud-only organizations, service accounts for cloud apps
Synced user	Created in on-prem AD, synced to Entra ID	Entra Connect syncs from on-prem AD	Organizations with existing on-prem AD that are moving to cloud
Guest user	External user from another Entra ID tenant	Invited via B2B collaboration	Partner access, contractor access, cross-

Type	What it Is	How Created	Use Case
			organization collaboration

Important: You cannot create a synced user directly in Entra ID. The source of authority is on-prem AD. Changes must be made on-prem and synced.

Guest user details: - Invited by email — they receive an invitation link - They authenticate with their home tenant credentials - You control what they can access in your tenant via RBAC and Conditional Access - Guest users can use MFA from their home tenant - You can restrict guest access with cross-tenant access policies

Groups in Entra ID

Type	How Members are Added	License Required	When to Use
Assigned (static)	Manually add/remove members	Any	Small teams, specific assignments
Dynamic user	Rule-based (e.g., department == “IT” -> add to IT group)	P1	Large organizations, auto-provisioning based on user attributes
Dynamic device	Rule-based for devices (e.g., deviceOSType == “Windows” -> add group)	P1	Auto-group devices for compliance policies

Dynamic group examples: - All users in Marketing department: `user.department -eq "Marketing"` - All users with job title containing “Engineer”: `user.jobTitle -contains "Engineer"` - All users in a specific office: `user.physicalDeliveryOfficeName -eq "NYC"` - All Windows devices: `device.deviceOSType -eq "Windows"`

Why dynamic groups are powerful: New employee joins, department=IT, they automatically get all IT group permissions. No manual assignment. When they change department, they automatically lose IT access and gain the new department’s access.

App Registrations

Register an application in Entra ID so it can authenticate users or authenticate itself.

What you configure in an app registration: - Name and branding - Supported account types: single tenant, multi-tenant, multi-tenant + personal Microsoft accounts - Redirect URIs (where Entra ID sends the auth response after login) - Client secret or certificate (for the app to authenticate itself) - API permissions: what Microsoft Graph and other APIs the app can access - Expose an API: define scopes that other apps can request from your app - Optional claims: configure what claims are included in tokens

App registration = the definition of the application. It does not grant access to anything by itself.

Enterprise Applications

An enterprise application is an instance of an app registration that has been deployed for use in your organization.

What you configure in an enterprise application: - User assignment required: only assigned users can access the app - Assignment: which users and groups can access the app - SSO configuration: SAML, OpenID Connect, password-based, or linked - Provisioning: automatic user account creation in the SaaS app (SCIM) - Conditional Access: apply access policies to this specific app

Service Principals

A service principal is the “identity” of an application in a specific tenant.

Think of it this way: - App registration = the application definition (what the app is) - Service principal = the application’s identity in your tenant (what the app can do in your organization) - Every app registration automatically creates a service principal in the home tenant - When a multi-tenant app is consented in another tenant, a service principal is created in that tenant

Service principals are used for: - Assigning RBAC roles (give the app access to Azure resources) - Assigning API permissions (give the app access to Microsoft Graph, etc.) - Configuring SSO (the service principal holds the SAML configuration)

Topic 35: Entra Connect (Hybrid Identity)

Synchronizes on-prem AD identities to Entra ID. Required for hybrid identity — users use the same credentials for both on-prem and cloud.

Authentication Methods — Detailed

Password Hash Sync (PHS):

How it works: 1. Entra Connect extracts the password hash from on-prem AD (the hash of the hash — MD4 hash extracted, then SHA256 hashed again, then salted) 2. The double-hashed value is sent over TLS to Entra ID 3. When a user signs in, Entra ID hashes the entered password the same way and compares 4. Authentication happens entirely in the cloud

Advantages: - Simplest to deploy (checkbox in Entra Connect) - No on-prem dependency for authentication (if on-prem goes down, users can still sign in to cloud apps) - Enables leaked credential detection (Microsoft checks hashes against known breached credentials) - Required for Identity Protection risk detection - Recommended by Microsoft as the default auth method

Disadvantages: - Password hashes exist in the cloud (double-hashed, but still present) - Some organizations have compliance requirements against any password hash leaving on-premises

Pass-through Authentication (PTA):

How it works: 1. User enters credentials in Entra ID sign-in page 2. Entra ID encrypts the credentials and places them in a queue 3. The PTA agent (installed on-prem on a Windows Server) picks up the request 4. The agent validates the credentials directly against on-prem AD (LDAP bind) 5. Result is sent back to Entra ID 6. Authentication happens on-prem, Entra ID just issues the token

Advantages: - No password hash leaves on-premises - On-prem password policies are enforced (expiration, complexity, lockout) - Supports on-prem smart card authentication (if using AD FS as the PTA agent backend)

Disadvantages: - Requires on-prem agents (high availability: deploy at least 2 agents) - If all agents go down, cloud authentication fails - Requires outbound HTTPS (agents connect to Entra ID, not the other way around)

Federation (AD FS):

How it works: 1. User tries to access a cloud app 2. Entra ID redirects the user to the on-prem AD FS endpoint 3. User authenticates directly with on-prem AD FS (Kerberos, certificate, smart card, etc.) 4. AD FS issues a SAML token 5. Entra ID validates the token and issues its own tokens for cloud apps

Advantages: - Full control over authentication policies - Supports complex scenarios (smart card, certificate-based auth, claims transformation) - Can authenticate third-party federation partners

Disadvantages: - Most complex to deploy and maintain - Requires AD FS infrastructure (servers, load balancers, SSL certificates) - On-prem dependency for all authentication - Requires significant expertise to troubleshoot

Microsoft recommendation: Use PHS as the primary method. It is the simplest, most resilient, and enables the most security features. Use PTA only if compliance prohibits any password hash in the cloud. Use federation only for complex authentication scenarios that PHS/PTA cannot handle.

Even with federation, enable PHS as a backup. If AD FS goes down, Entra ID can fall back to PHS for authentication.

Entra Connect Sync Configuration

Sync scope: - All users and groups (default) - Filtered by OU, domain, or attribute (reduce sync scope) - Cloud-only users are not affected (they exist only in Entra ID)

Sync interval: - Default: every 30 minutes - Can be forced manually via PowerShell: `Start-ADSyncSyncCycle -PolicyType Delta` - Full sync required after schema changes: `Start-ADSyncSyncCycle -PolicyType Initial`

What syncs: - Users (attributes: display name, email, department, job title, manager, phone, etc.) - Groups (security groups, distribution groups, Microsoft 365 groups) - Device objects (for hybrid Entra ID join) - Password hashes (if PHS is enabled)

What does NOT sync: - Group Policy Objects - Organizational Units (OU structure is not replicated — groups are flat in Entra ID) - Computer objects (device objects sync differently for hybrid join) - Service accounts (unless explicitly included)

Entra Connect Health

- Monitoring agent for Entra Connect, AD FS, and AD DS
- Dashboard showing sync errors, health status, last sync time
- Alerts for sync failures, export errors, password hash sync issues
- Requires P1 license
- Install on the same server as Entra Connect or AD FS

Topic 36: Role-Based Access Control (RBAC)

Controls who can do what on which Azure resources. The primary authorization mechanism in Azure.

How RBAC Works

Three components combine to form a role assignment:

1. Security Principal (who): - User (individual Entra ID user) - Group (Entra ID group — recommended for manageability) - Service principal (application identity) - Managed identity (auto-managed identity for Azure resources)

2. Role Definition (what): - A collection of permissions - Permissions are in the format: `Microsoft.{provider}/{resource}/{action}` - Example: `Microsoft.Compute/virtualMachines/read` = read VM properties - Example: `Microsoft.Compute/virtualMachines/write` = create or update VMs

3. Scope (where): - Management Group (applies to all subscriptions underneath) - Subscription (applies to all resource groups and resources) - Resource Group (applies to all resources in the RG) - Resource (applies to one specific resource)

Role Assignment = Principal + Role + Scope

```
User: john@company.com
Role: Virtual Machine Contributor
Scope: /subscriptions/xxxx/resourceGroups/rg-prod-eastus
→ John can manage VMs in rg-prod-eastus, but nothing else
```

Built-in Roles — Complete List of Commonly Used

Role	What it Allows	What it Does NOT Allow	When to Assign
Owner	Full access to all resources + can manage access	Nothing	Subscription/RG admins only. Minimize the number of Owners.
Contributor	Full access to all resources	Cannot manage access (cannot assign roles)	App/dev teams for their resources
Reader	View all resources	Cannot make any changes	Auditors, monitoring, reporting
Virtual Machine Contributor	Manage VMs, disks, extensions	Cannot manage VNet, storage accounts, RBAC	VM administrators
Network Contributor	Manage all networking resources	Cannot manage VMs, RBAC	Network team
Storage Account Contributor	Manage storage accounts	Cannot access data within the storage account	Storage admins
Key Vault Contributor	Manage Key Vault resources	Cannot access secrets, keys, certificates	Key Vault administrators
SQL Server Contributor	Manage SQL servers and databases	Cannot access data, cannot manage policies	DBAs
User Access Administrator	Manage RBAC assignments	Cannot manage resources	Identity admins
Monitoring Contributor	Read monitoring data + manage monitoring settings	Cannot manage resources	Operations team
Backup Contributor	Manage backup vaults and backups	Cannot manage resources being backed up	Backup operators

Important distinctions: - **Owner vs Contributor:** Owner can assign roles. Contributor cannot. This is the ONLY difference for resource access. - **Storage Account Contributor vs Storage Blob Data**

Owner: The first manages the resource (create/delete storage accounts). The second manages the data (read/write blobs). These are DIFFERENT things. - Never assign Owner broadly. Use Contributor + User Access Administrator if someone needs both resource management and role management.

Custom Roles

When built-in roles do not fit, create custom roles with specific permissions.

Example: VM Restart Only (no create, no delete):

```
{
  "Name": "VM Restarter",
  "Description": "Can restart VMs but not create or delete them",
  "Actions": [
    "Microsoft.Compute/virtualMachines/read",
    "Microsoft.Compute/virtualMachines/restart/action",
    "Microsoft.Compute/virtualMachines/deallocate/action",
    "Microsoft.Compute/virtualMachines/start/action"
  ],
  "NotActions": [],
  "AssignableScopes": ["/subscriptions/xxxx-xxxx-xxxx"]
}
```

Actions vs NotActions: - Actions: what the role ALLOWS - NotActions: what the role DENIES (subtracted from Actions) - NotActions is useful when a broad role needs one specific permission removed

Example using NotActions:

```
{
  "Name": "Contributor-No-Delete",
  "Actions": ["*"],  // All actions
  "NotActions": [
    "Microsoft.Compute/virtualMachines/delete",
    "Microsoft.Resources/subscriptions/resourceGroups/delete"
  ]
}
```

This gives all Contributor permissions but removes the ability to delete VMs and resource groups.

Scope Hierarchy

```
Management Group (Tenant Root)
  └── Subscription
      └── Resource Group
          └── Resource
```

How scope inheritance works: - Role assigned at MG → applies to ALL subscriptions, RGs, and resources below that MG - Role assigned at subscription → applies to ALL RGs and resources in that subscription - Role assigned at RG → applies to ALL resources in that RG - Role assigned at resource → applies to that ONE resource only

RBAC is additive: - If you have Reader at subscription level and Contributor at RG level, you get Contributor in that RG and Reader everywhere else - Multiple role assignments are combined — permissions are added together - There is no “deny” in RBAC (use Azure Policy or Azure Firewall for deny)

RBAC Limits:

Limit	Value
Role assignments per subscription	2,000
Custom roles per tenant	5,000
Scope levels	4 (MG, Sub, RG, Resource)
Max inherited assignments	No limit (all cascade down)

RBAC vs Azure Policy

Feature	RBAC	Azure Policy
What it controls	Who can do what	What can be created / how it must be configured
Example	John can create VMs	No VMs allowed outside East US
Enforcement	Per user/group	Per subscription/RG
Action	Allow/deny actions	Allow/deny/audit resource configurations
Scope	MG, Sub, RG, Resource	MG, Sub, RG
Can deny?	No (only allow)	Yes (Deny effect)

Use RBAC for access control. Use Azure Policy for resource governance. They complement each other.

Topic 37: Privileged Identity Management (PIM)

Manage, control, and monitor access to important resources. Provides just-in-time privileged access. The most important security tool most organizations underuse.

Why PIM

Without PIM: - 15 IT staff have permanent Owner role on production - Any one of them could accidentally or maliciously cause damage - No audit trail for why someone needed elevated access - No time limit on privileged access - Impossible to know who used what, when, and why

With PIM: - Nobody has permanent privileged access - They request it when needed, for a limited time, with approval and justification - All actions are logged and auditable - Automatic expiration prevents “set and forget” privileged access

Key Concepts

Concept	What it Means
Eligible assignment	User CAN activate the role, but does NOT have it by default. No permissions until activated.

Concept	What it Means
Active assignment	User HAS the role right now. Can be permanent or time-bound.
Activation	User requests to go from eligible → active. May require: MFA, approval, justification, ticket number.
Maximum duration	How long an activation lasts (e.g., 4 hours, 8 hours, 24 hours). Auto-expires.
Approval required	One or more approvers must approve the activation request before the role is granted.
Justification required	User must explain why they need the role (e.g., “Restarting prod-vm-01 for emergency patching”).
MFA on activation	User must complete MFA when activating the role. Prevents compromised accounts from activating privileged roles.
Access Reviews	Periodic review of who has role assignments. Reviewers can approve or remove access.

PIM Workflow — Step by Step

1. John has “Eligible” Virtual Machine Contributor on the production RG
2. John needs to restart a VM at 2 AM
3. John goes to Entra ID → Privileged Identity Management → My roles
4. John clicks “Activate” on Virtual Machine Contributor
5. PIM requires: MFA verification + justification + manager approval
6. John completes MFA, writes “Restarting app-vm-01 for emergency patching”
7. Manager receives notification, reviews, approves
8. John gets VM Contributor for 4 hours
9. John restarts the VM
10. After 4 hours, the role automatically deactivates
11. All actions John took during those 4 hours are logged in the PIM audit log

PIM for Azure Resources

Works on all RBAC roles at any scope: - Management Group roles (Owner, Contributor, User Access Administrator) - Subscription roles - Resource Group roles - Individual resource roles

PIM for Entra ID Roles

Works on directory roles: - Global Administrator (most critical — should ALWAYS be PIM-eligible, never permanent) - Privileged Role Administrator - Security Administrator - Exchange Administrator - SharePoint Administrator - Any other directory role

Critical: Global Administrator should have a maximum of 2-3 permanent holders (break-glass accounts). All other Global Admins should be PIM-eligible only.

PIM Best Practices

1. **All privileged roles should be eligible, not permanently active.** This is the core principle.
2. **Require MFA for every activation.** No exceptions.
3. **Require approval for high-privilege roles** (Owner, Global Admin, User Access Administrator).
4. **Set maximum activation duration to 8 hours or less.** Most tasks do not need more than 4 hours.
5. **Run quarterly access reviews.** Remove access that is no longer needed.
6. **Enable PIM alerts:** “Eligible assignments without MFA”, “Active assignments exceeding threshold”, “Users activating roles too frequently”.
7. **Use break-glass accounts** for emergency access. These are permanent Global Admin accounts with complex passwords stored in a safe. Only used when PIM is unavailable.
8. **Never assign permanent active roles** to individuals. If someone argues they need permanent access, set it to expire in 90 days (time-bound active assignment with renewal required).

PIM License Requirement

Entra ID P2 ($9/user/month). P1 gives basic PIM but without approvals, access reviews, or just-in-time for Azure resources. P2 is worth it.

Topic 38: Conditional Access

Entra ID feature that enforces access policies based on conditions. The enforcement engine for Zero Trust. The single most powerful security control in Azure.

How Conditional Access Works

```
IF (conditions are met) THEN (enforce controls)
```

Every sign-in is evaluated against all Conditional Access policies. If any policy blocks, the sign-in is blocked. If multiple grant policies apply, ALL grant controls must be satisfied.

Conditions (the IF)

Condition	What you Check	Examples
Users/Groups	Who is trying to access	All users, specific groups, excluded groups (break-glass accounts)
Cloud apps	Which application is being accessed	All cloud apps, Office 365, Azure Management, specific SaaS apps
Client apps	How they are accessing	Browser, mobile apps, desktop apps, Exchange

Condition	What you Check	Examples
		ActiveSync, legacy authentication
Device platform	Operating system	Windows, macOS, iOS, Android
Device state	Device compliance	Compliant, domain-joined, Entra-joined, hybrid-joined
Location	Geographic or IP-based	Named locations (countries, IP ranges), trusted locations (office IPs)
Sign-in risk	Probability that the sign-in is not the legitimate user	Low, Medium, High (from Identity Protection — requires P2)
User risk	Probability that the user account is compromised	Low, Medium, High (from Identity Protection — requires P2)
Authentication context	Triggered by a sensitive action within an app	Step-up authentication for sensitive operations
Filter for devices	Device attributes via Graph API	specific device properties

Controls (the THEN)

Grant controls (allow, but with conditions):

Control	What it Requires	When to Use
Require MFA	User must complete multi-factor authentication	Baseline for all users
Require compliant device	Device must be Intune-compliant	For accessing M365 or sensitive apps
Require domain-joined device	Device must be on-prem AD joined (hybrid)	Legacy requirement for some orgs
Require approved client app	Must use approved app (e.g., Outlook mobile instead of generic mail)	Prevent app sprawl, ensure managed apps
Require app protection policy	Intune app protection policy applied	Protect data on BYOD devices
Require password change	Force password reset	High user risk (account likely compromised)
Terms of use	User must accept terms	Guest access, contractor onboarding

Session controls (limit what they can do):

Control	What it Does	When to Use
Sign-in frequency	Force re-authentication every N hours	Contractors (4 hours), high-security apps (1 hour)
Persistent browser session	Allow/disallow “Stay signed in”	Disable for shared devices
Conditional Access App Control	Route traffic through Defender for Cloud Apps	Monitor and block risky file downloads
Disable resilience scope	Require fresh auth, no cached tokens	Most restrictive — use sparingly

Block access: Deny completely. No conditions, no exceptions. Use for: blocking legacy authentication, blocking specific countries, blocking specific apps.

Common Conditional Access Policies — Detailed

Policy 1: Require MFA for all users

Setting	Value
Users	All users
Cloud apps	All cloud apps
Conditions	None
Grant	Require MFA

This is the baseline policy. Every organization should have this. Start in report-only mode, monitor for a week, then enable.

Policy 2: Block legacy authentication

Setting	Value
Users	All users
Cloud apps	All cloud apps
Conditions → Client apps	Exchange ActiveSync clients, Other clients (legacy protocols)
Access	Block

Legacy protocols (POP3, IMAP, SMTP, older Office versions) do not support MFA. Attackers brute-force these protocols because they bypass MFA. Blocking them eliminates a major attack vector.

Policy 3: Require compliant device for M365

Setting	Value
Users	All users
Cloud apps	Office 365
Conditions → Device platforms	Any
Conditions → Device state	Require device to be marked as compliant
Grant	Require compliant device

Ensures only managed, compliant devices can access M365. Prevents access from personal/unmanaged devices.

Policy 4: MFA for risky sign-ins

Setting	Value
Users	All users
Cloud apps	All cloud apps
Conditions → Sign-in risk	Medium, High
Grant	Require MFA

When Identity Protection detects a risky sign-in (impossible travel, unfamiliar IP, malware-linked IP), require MFA. Requires P2.

Policy 5: Block access from untrusted countries

Setting	Value
Users	All users

Setting	Value
Cloud apps	All cloud apps
Conditions → Location	Any location EXCEPT trusted named locations
Access	Block

Create named locations for countries/regions where your users are located. Block access from everywhere else.

Policy 6: Require MFA for Azure Management

Setting	Value
Users	All users
Cloud apps	Microsoft Azure Management
Grant	Require MFA

Anyone accessing the Azure Portal, Azure CLI, or Azure PowerShell must complete MFA. Critical for protecting infrastructure access.

Policy 7: Session limits for contractors

Setting	Value
Users	Contractor group
Cloud apps	All cloud apps
Session	Sign-in frequency = 4 hours, no persistent browser session

Contractors must re-authenticate every 4 hours and cannot stay signed in.

Policy Evaluation

- All policies are evaluated for every sign-in
- If ANY policy blocks → blocked (no override)
- If multiple grant policies apply → ALL grant controls must be satisfied (AND logic)
  - Policy A requires MFA, Policy B requires compliant device → user must do both
- If no policies apply → access is allowed (no restrictions)
- Policies are evaluated in no particular order (all apply simultaneously)

Best Practices

- **Start in report-only mode.** Every new policy should start as report-only for at least 1 week. Monitor the log to see what would be blocked.
- **Exclude break-glass accounts** from ALL Conditional Access policies. These emergency admin accounts must never be blocked.
- **Start with two policies:** “Require MFA for all users” + “Block legacy auth”
- **Use named locations** for trusted IPs (your office ranges)

- **Exclude service accounts** from Conditional Access (they cannot complete MFA)
- **Do not create too many policies.** 10-15 well-designed policies cover most scenarios. Too many policies = management nightmare and troubleshooting difficulty.
- **Test with a small group first** before applying to all users

---

Topic 39: Managed Identities

Azure resources (VMs, App Service, Functions, Container Apps) can authenticate to other Azure services WITHOUT storing credentials. This is the single most important security practice for application authentication in Azure.

Why Managed Identities

The problem without managed identities: - You need a connection string to access Azure SQL: `Server=myserver.database.windows.net;User ID=myapp;Password=P@ssw0rd123` - This password is stored in a config file, environment variable, or code - If the password is compromised, an attacker can access the database - When the password expires, you must update it everywhere (all VMs, all config files) - No audit trail of which app accessed what, when

The solution with managed identities: - Azure automatically provides an identity for your resource - Your code requests an access token from the local metadata service - The token is used to authenticate to Azure services - No credentials in code, config files, or environment variables - Azure handles credential rotation automatically - Full audit trail in Entra ID sign-in logs

Two Types

Type	What it Is	Lifecycle	When to Use
System-assigned	Identity tied to ONE Azure resource. Created and deleted with the resource.	Same as the resource. If VM is deleted, identity is deleted.	Simple scenarios where one resource needs one identity. Most common choice.
User-assigned	Standalone identity resource. Can be assigned to MULTIPLE resources.	Independent. Survives resource deletion. Must be manually deleted.	Multiple resources sharing the same identity. Pre-created identity for deployment scripts.

System-assigned example: - VM “web-prod-01” gets identity “web-prod-01-identity” - If web-prod-01 is deleted, the identity is deleted automatically - Each VM gets its own unique identity

User-assigned example: - Create identity “my-app-identity” - Assign to VM “web-prod-01” and VM “web-prod-02” - Both VMs share the same identity and permissions - If one VM is deleted, the identity persists - Must manually delete “my-app-identity” when no longer needed

How it works — Technical Details

Step 1: Enable managed identity on the Azure resource (portal, CLI, or ARM)

```
# Enable system-assigned identity on a VM
az vm identity assign -g myResourceGroup -n myVM
```

Step 2: Assign RBAC roles to the managed identity

```
# Give the VM's identity read access to a storage account
az role assignment create --assignee <principal-id> --role "Storage Blob Data Reader" --scope <storage-account-resource-id>
```

Step 3: In your application code, request a token

```
# From inside the VM, call the IMDS endpoint
curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://storage.azure.com/' -H Metadata:true
```

Step 4: Use the token to authenticate

```
# Use the access token in the Authorization header
curl https://myaccount.blob.core.windows.net/mycontainer/myblob -H "Authorization: Bearer <access-token>"
```

Services that Support Managed Identity

Category	Services
Compute (can HAVE a managed identity)	VMs, VMSS, App Service, Functions, Container Apps, AKS, Azure Data Factory
Azure services (can USE a managed identity to access)	Azure Storage, Azure SQL, Cosmos DB, Key Vault, Event Hubs, Service Bus, Data Lake, Synapse

DefaultAzureCredential — The Easy Way

For .NET, Java, Python, and JavaScript applications, you do not need to call the IMDS endpoint directly. Use the Azure SDK with `DefaultAzureCredential`:

```
// C# example
var credential = new DefaultAzureCredential();
var blobClient = new BlobClient(new Uri("https://myaccount.blob.core.windows.net/mycontainer/myblob"), credential);
```

`DefaultAzureCredential` automatically tries multiple credential types in order: 1. Environment variables (for local development) 2. Managed Identity (when running in Azure) 3. Visual Studio credentials (when developing locally) 4. Azure CLI credentials (when developing locally)

This means the same code works locally during development AND in production with managed identity. No code changes.

Common Mistakes with Managed Identities

1. **Not assigning RBAC roles to the managed identity.** Enabling the identity is only half the job. You must also grant the identity permissions to the target resource via RBAC.
2. **Using access keys instead of managed identity.** If managed identity is supported, use it. No excuses.
3. **Forgetting that managed identity tokens are cached.** If you change RBAC assignments, the token may still have old permissions until it expires (~24 hours). Test after changes.
4. **Not using user-assigned identities for multi-resource scenarios.** If 10 VMs need the same permissions, assign them the same user-assigned identity. Do not create 10 system-assigned identities with duplicate RBAC assignments.