Week-12 | Azure IAM from Basics | Azure Managed Identities Demo with Microsoft Entra

Introduction to Identity and Access Management (IAM) in Azure

Identity and Access Management (IAM) is a fundamental aspect of cloud security in Microsoft Azure. IAM ensures that users and resources have the right level of access while minimizing security risks. It involves two key components:

- **Authentication** – Verifying the identity of a user or system before granting access.
- **Authorization** – Determining what actions the authenticated user or system can perform.

In an organization, different roles such as developers, DevOps engineers, and managers require specific levels of access to Azure resources. Instead of granting administrative access to everyone, IAM enforces **the principle of least privilege**, ensuring users have only the permissions required for their tasks.

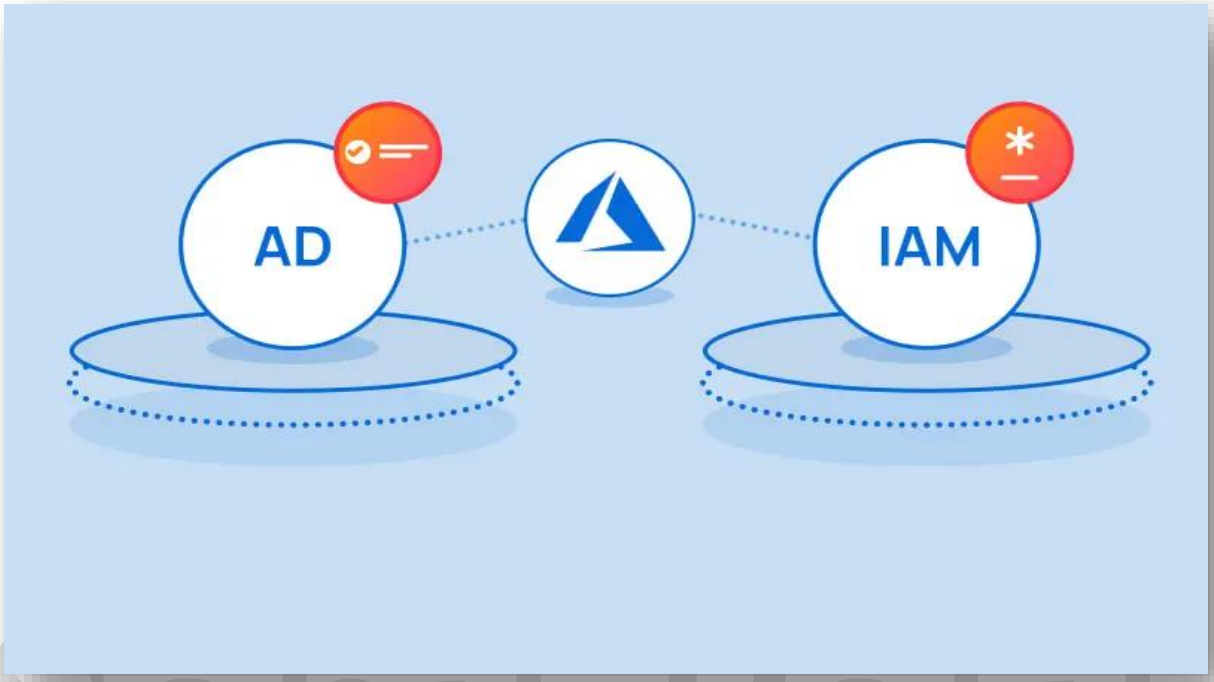
Understanding Microsoft Entra ID (Azure Active Directory - AAD)

To implement IAM effectively in Azure, **Microsoft Entra ID (previously known as Azure Active Directory - AAD)** is used. Microsoft Entra ID is a cloud-based identity and access management service that integrates with various Microsoft services.

Key Features of Microsoft Entra ID

1. **User Management** – Allows creating and managing user identities in the Azure portal.
2. **Groups** – Users with similar access requirements can be grouped, and roles can be assigned at the group level.
3. **Roles & Permissions** – Users and groups are assigned specific roles with predefined or custom permissions.
4. **App Registrations** – Enables applications to authenticate using Entra ID.

Creating a User and Assigning Roles



1. Navigate to **Microsoft Entra ID** in the Azure portal.
2. Create a new user with basic details such as username and password.
3. Assign a **role-based access control (RBAC)** permission. If a user does not have the necessary permissions, they will be denied access to certain Azure resources.
4. **Group-Based Role Management** – Instead of assigning roles to each user manually, they can be added to a group, and permissions are granted to the group.

Role-Based Access Control (RBAC) in Azure

Azure implements **Role-Based Access Control (RBAC)** to manage access efficiently. RBAC assigns roles to users, groups, or managed identities at different scopes such as:

- **Subscription level** – Access to all resources in a subscription.
- **Resource Group level** – Access to a specific set of resources within a resource group.
- **Resource level** – Granular access to a single resource like a virtual machine or storage account.

Common Built-in Azure Roles

1. **Owner** – Full access to all resources, including permission management.
2. **Contributor** – Can manage resources but cannot assign roles.

3. **Reader** – Read-only access to resources.
4. **User Access Administrator** – Can manage user access and permissions.

Custom Roles in Azure

- If default roles do not meet organizational requirements, custom roles can be created.
- Custom roles require an **Azure AD Premium P1 or P2** subscription.
- A JSON file defining permissions is used to create custom roles.

Managing Access Between Azure Resources

While IAM secures user access, Azure also needs to manage how different services communicate securely. For example, a virtual machine may need to access a file stored in Azure Blob Storage.

Service Principals and Managed Identities

Azure provides two methods to handle resource authentication:

1. **Service Principals** – A special type of identity that allows applications to authenticate and access resources. However, managing secrets for service principals requires manual intervention.
2. **Managed Identities** – An identity that is automatically managed by Azure, eliminating the need for credential rotation.

Understanding Managed Identities in Azure

Managed Identities provide a secure way for Azure resources to authenticate with other services. There are two types:

1. System-Assigned Managed Identity

- Tied directly to a specific Azure resource (e.g., a virtual machine).
- Automatically created and deleted with the resource.
- Cannot be shared with multiple resources.

2. User-Assigned Managed Identity

- A standalone identity that can be assigned to multiple resources.
- Created independently and managed separately from resources.

Advantages of Using Managed Identities

- **Eliminates the need for credentials** – No need to store credentials in applications.
- **Automatic token retrieval** – Azure manages token issuance.
- **Seamless authentication** – Works with services like Azure Storage, Key Vault, and Azure SQL Database.

Demo: Using Managed Identities to Access Azure Blob Storage

Step 1: Create an Azure Storage Account and Container

1. Navigate to **Azure Storage Accounts** in the portal.
2. Create a new storage account with a globally unique name.
3. Inside the storage account, create a **container** to store files.

Step 2: Create and Configure a Virtual Machine (VM)

1. Deploy a **Linux-based virtual machine** in Azure.
2. Use **SSH key authentication** for enhanced security.
3. Enable **System-Assigned Managed Identity** for the VM in the Azure portal.

Step 3: Assign the Managed Identity Storage Permissions

1. Navigate to the **Storage Account → Access Control (IAM)**.
2. Click **Add Role Assignment**.
3. Assign the **Storage Blob Data Reader** role to the VM's managed identity.

Step 4: Fetching an Access Token and Accessing Storage

1. **Login to the VM using SSH:**
2. ssh -i my-key.pem azureuser@vm-ip-address
3. **Install required utilities (JQ for parsing JSON):**
4. sudo apt update
5. sudo apt install jq -y
6. **Fetch the Managed Identity Access Token:**
7. TOKEN=$(curl -s -H "Metadata: true" "http://169.254.169.254/metadata/identity/oauth2/token?api-version=2019-08-01&resource=https://storage.azure.com" | jq -r .access_token)
8. **Use the Token to Access Azure Blob Storage:**
9. curl -H "Authorization: Bearer $TOKEN" \
10. -H "x-ms-version: 2020-06-12" \
11. "https://<storage-account-name>.blob.core.windows.net/<container-name>/<blob-name>"

Step 5: Automating the Process Using Shell Script

```
#!/bin/bash

# Fetch Access Token
TOKEN=$(curl -s -H "Metadata: true" "http://169.254.169.254/metadata/identity/oauth2/token?api-version=2019-08-01&resource=https://storage.azure.com" | jq -r .access_token)

# Define Storage Details
STORAGE_ACCOUNT="<storage-account-name>"
CONTAINER="<container-name>"
BLOB="<blob-name>"

# Access the File in Blob Storage
curl -H "Authorization: Bearer $TOKEN" \
    -H "x-ms-version: 2020-06-12" \
    "https://${STORAGE_ACCOUNT}.blob.core.windows.net/${CONTAINER}/${BLOB}"
```


Conclusion

- IAM is essential for securing access to Azure resources through authentication and authorization.
- Microsoft Entra ID provides centralized identity management.
- Role-Based Access Control (RBAC) simplifies permissions management.
- Managed Identities eliminate the need for storing credentials in applications.
- Using **system-assigned managed identities**, a virtual machine can securely access **Azure Blob Storage** without manually managing credentials.
- This approach improves security, reduces manual effort, and simplifies Azure resource authentication.

By following these best practices, organizations can ensure secure, efficient, and scalable IAM implementation in Microsoft Azure.