Entra (Active Directory) Basics

It is identity management service to manage Users and Permissions.

We have Tenants, Users, Groups, Accounts and Subscriptions as the basic units.

Tenants :

Tenants is an instance of Azure AD and represents the organisation. It helps manage the users, groups and access permissions to Azure resources.

After creating an Account, a new Azure AD Tenant is created automatically.

A Tenant can be thought of as a Container for multiple Subscriptions.

Subscriptions :

They act as a billing boundary for Azure Resources we create. It is a billing entity which tells us what resources we use. We can control the usage and budgets of the resources.

We can use multiple subscriptions each for dev, test and production under on Tenant.

Accounts :

An Azure account is simply the email ID you use to sign in to Azure. This account can be tied to multiple tenants or subscriptions.

- with this account, we can access multiple tenants and subscriptions if given proper permissions

User :

A user is an identity inside a tenant. Users are managed by Azure AD and assigned roles/permissions.

Groups :

This is self-explanatory as we add Users in to groups to logically group them according to the roles and permissions that needed to be assigned to them rather than having to assign them to individual users.

We can search for Microsoft Entra ID which will initially take us to the Default Directory Page. Default Dir is an instance of Active Dir. We will be brought to Default Directory page having a single User in it with the name we created the Azure Account with.

Azure AD vs Default Directory :

Azure AD (Microsoft Entra ID) is a cloud-based identity and access management service that provides authentication, user management, and security for Azure resources and applications.

The Default Directory is the first Azure AD tenant created automatically when you sign up for an Azure account. It's a specific instance of Azure AD.

1 : The User has several important fields like Object ID, roles (most powerful role : **Global Admin**).

2 : We can check the Sign-in logs of the user and most imp the Authentication Requirement as Single-factor Authentication which is a very basic one and not recommended.

3 : We even can add users to the Tenant by being in the Users Overview Section in the portal.

We can choose whether to let the newly added users change/reset their passwords or they should email the admin as it happens in the corporates.

We can use a feature called Self-service password reset under Password Reset Blade to achieve this.

<u>Adding a User to a Tenant :</u>

1 : Create a user by going into default dir and then Users blade.

2 : Select New user and add the UPN (User Principal Name) which is username followed by the tenant initial domain.

Every user added will have this initial domain which is assigned to the Tenant during the Azure Account creation.

3 : We can even invite an external user using his gmail id which will send him an invitation link.

Adding a User to a Group :

1 : Go to the Active Dir and select Groups.

2 : Then New Group tab.

3 : Group type => Security

Azure AD Licenses :

We have 3 license types :

1 : Free

2 : Premium 1 : 6$ / user / month

3 : Premium 2 : 9$ / user / month

Authentication Types :

1 : Something you know : username / pw

2 : Something you are : fingerprint | iris scan

3 : Something you have : phone | smart-card

Consider 3 factors when implementing :

1 : Feasibility : Whether all devices support fingerprint auth.

2 : Ease : Username/ Pw is faster

3 : Sensitivity : Biometric data as it is hard to hack

Two factor Authentication :

Combination of Auth Types by mixing any 2.

Security Defaults :

Inorder to prompt users to enable 2 factor auth or MFA, we can use security defaults which will do just that after user successfully logs in.

Enabling Security Defaults :

1 : Go to Entra ID Service => Properties

RBAC

[We assign roles to the users and groups]

[We assign Service Principal and Managed Identity to a Service that wants to access another service]

<u>They function similarly.</u>

If we were to assign roles and permission to a user or group like in the past, we would assign each group or a user the role and permissions individually.

eg : John can add items to the Inventory
David cannot be allow to read data from the DB

This granularity is generally very difficult to maintain. Hence, we have RBAC that helps resolve this issue.

We have :

Users : The actual user

Roles : The role can be Hotel Owner, Receptionist, Employee

Authorisations : Allowed Actions. Set Prices | Checkin | Checkout

Assigning Authorisations to roles logically.

Owner is allowed all the actions from Authorizations but the Receptionist can only Checkin and Checkout.

To sum it up :

We have to grant users the roles to create Resource Groups, Access data from SQL or see metrics of any application.

There are 3 types of roles :

1 *: Owner* : Performing any action / assigning roles

2 *: Contributor* : Perform any action / cannot assign roles.

3 : *Reader* : Can only view the data without changing anything.

eg :

Role name - VM Contributor

Role name - CosmosDB Account Reader

Role name - SQL Db Account Owner : Full Access

Scenario : Assigning a Contributor Role to a Group of Users who can contribute towards the RG by creating resources.

Every resource/service will have its own IAM settings which can be used to manage Role Assignments. By default, the Owner (usually) or Contributor role is typically assigned at the subscription or resource group level, which cascades permissions to the resources within them.

Roles need to be assigned at the Resource Level.

1 : Create a RG in the portal.

2 : IAM => Add Role Assignment

3 : Select the type of role to be assigned.

i : Job function role : (Recommended for Minimum Access to Resources for security and safety)

- For specific job responsibilities or specific to a service like pushing/ pulling from ACR

- User Admin : Manage Users and Groups

- App Developer : Register and manage app config

ii : Privileged Admin role : Owner | Contributor

4 : Owner | Contributor => Select User / Group

5 : Try signing in with any of the Group member creds and see whether we can create resources.

Note : We can only create resources in this RG as we have permission to contribute towards this RG

Managed Identity :

We can assign Azure AD's identity to Azure Resources. This way, the resources can connect to other resources using this identity without us needing to handle credentials.

When we use Managed Identity, what we essentially do is toggle the Identity switch on a service we need to create an identity for, and then after its identity has been created, we go to the other service's IAM and assign a role to the Identity we created by selecting one role and selecting the Resource ti whom we wish to give access to.

We have 2 types of Managed Identities :

1 : System Assigned : Managed by Azure which is tied to the resource's lifecycle. Deleting the resource deletes the identity.

2 : User Assigned : Managed by the User. The user can assign it to multiple resources. It is not tied to the lifecycle.

Scenario : 1

We will assign an identity to the App Service so that it can connect to our SQL Server without needing Username/Password.

1 : First we have to go to **Identity** blade in the Web app and assign a system assigned identity to it by setting the status toggle to on.

This will create a Unique Identity (User) in the MS Entra ID.

2 : Now we will add this User which is mapped to Web App created in Entra to the Sql server.

Before that, we should set an admin for the Db first and then proceed with authenticating the Web App.

Sql Server => Entra ID => Set Admin => User

User above is one with which we signed into the portal

This will set the User that was created during Azure Account Creation as an admin for the DB Server.

3 : We can now sign into the DB Server by going to the **query editor** and use **Entra Authentication.**

4 : Now we will create a User in the Database with the name of our Azure Service issuing commands one by one against the Sql DB using the query editor.

We do this because even though our App Service has a Managed Identity, SQL Server doesn't automatically recognize it as a database user.

Hence, we create a database user that maps to the App Service's Managed Identity.

The commands are :

CREATE USER [<azure service-name>] FROM EXTERNAL PROVIDER

ALTER ROLE db_datareader ADD MEMBER [<azure service-name>]

ALTER ROLE db_datawriter ADD MEMBER [<azure service-name>]

5 : Now we will edit the SQL Connection String using the App Service's Configuration blade and delete all the text after *Persist Security Info = False;* and then add the following text after *Persist Security Info = False;* :

Add this in value : *Authentication=Active Directory Mananged Identity*

Now we can easily access the DB Server from our App Service.

Scenario : 2

We will connect to a Storage Account using Managed Identity and try to render an html page from a Linux VM

1 : We will be creating a Managed Identity (role) for the VM for it to access the Storage Account. So for that, we should create an System-assigned Identity for the VM from the VM's Identity Blade, by toggling the status setting.

2 : This will create a Managed Identity under Entra ID. This will enable us to assign RBAC roles as well.

3 : Now in storage account, we will assign a role to the managed identity created in the above steps. For that, we will go to IAM and go to *add role assignment* and give an owner role or any other role.

4 : Then we will select the Members and under the managed Identity select our VM