



# SQL Server Security Model

Module 4

## Learning Units covered in this Module

- Lesson 1: SQL Security Overview
- Lesson 2: Authentication and Authorization
- Lesson 3: Azure Active Directory and RBAC
- Lesson 4: SQL MI Logins and Users
- Lesson 5: SQL MI Roles
- Lesson 6: SQL MI Permissions and Schemas

# Lesson 1: SQL Managed Instance (MI) Security Overview

# Objectives

After completing this learning, you will be able to:

- Understand SQL Server Managed Instance security basics
- Understand the difference between Principals and Securables



# The Security Gold Standard



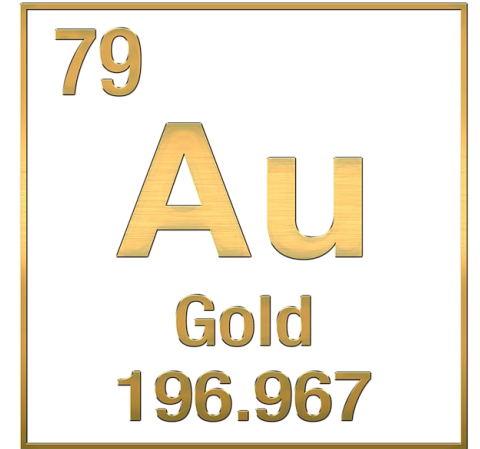
**AUTHENTICATION** – Verifies who you are



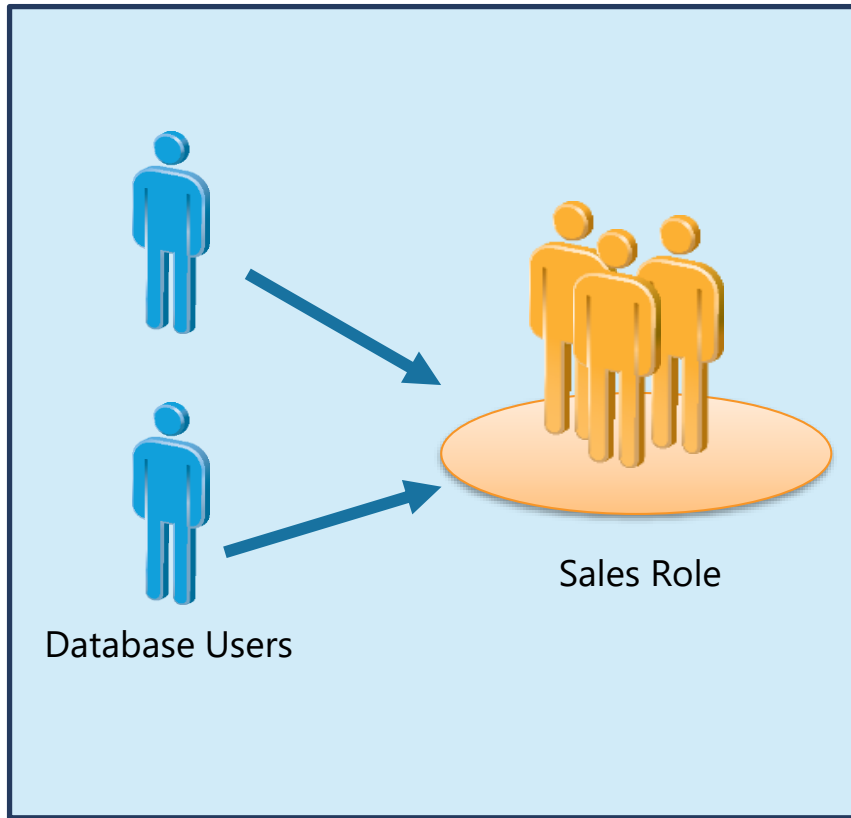
**AUTHORIZATION** – Assigns what you can do



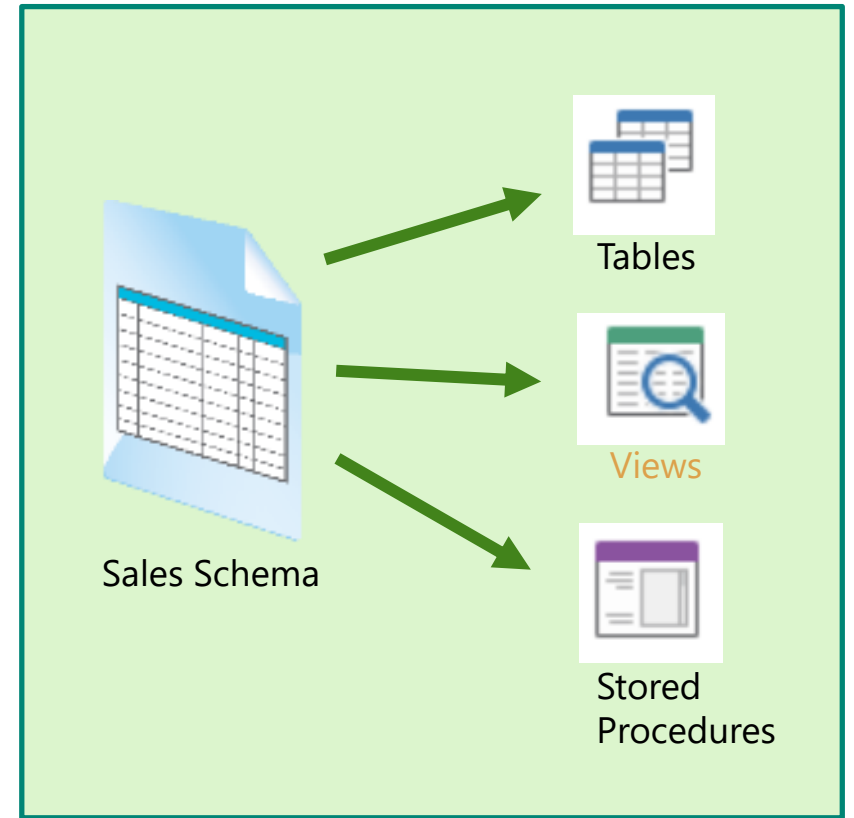
**AUDITING** – Monitors what you did



# Principals vs Securables

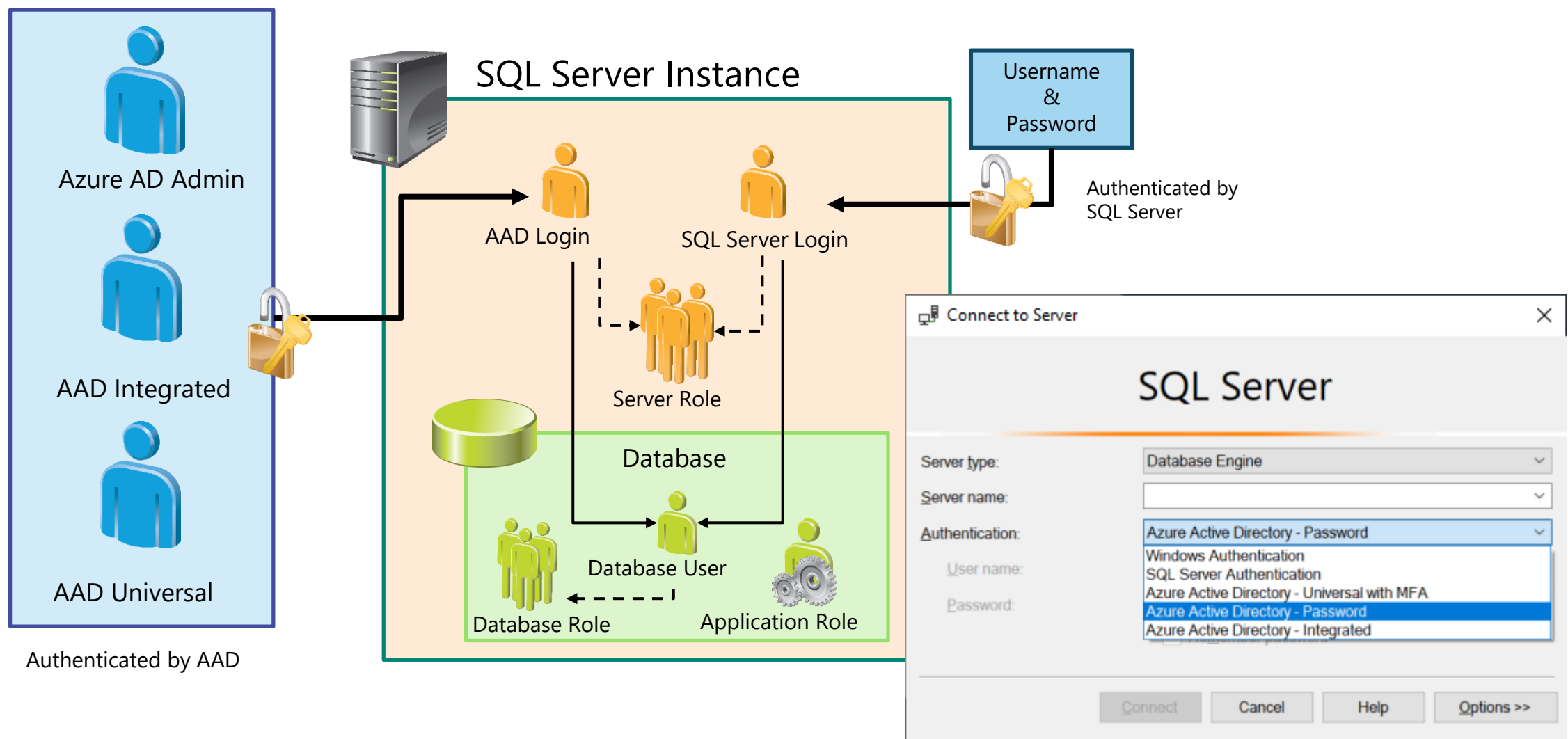


Principals

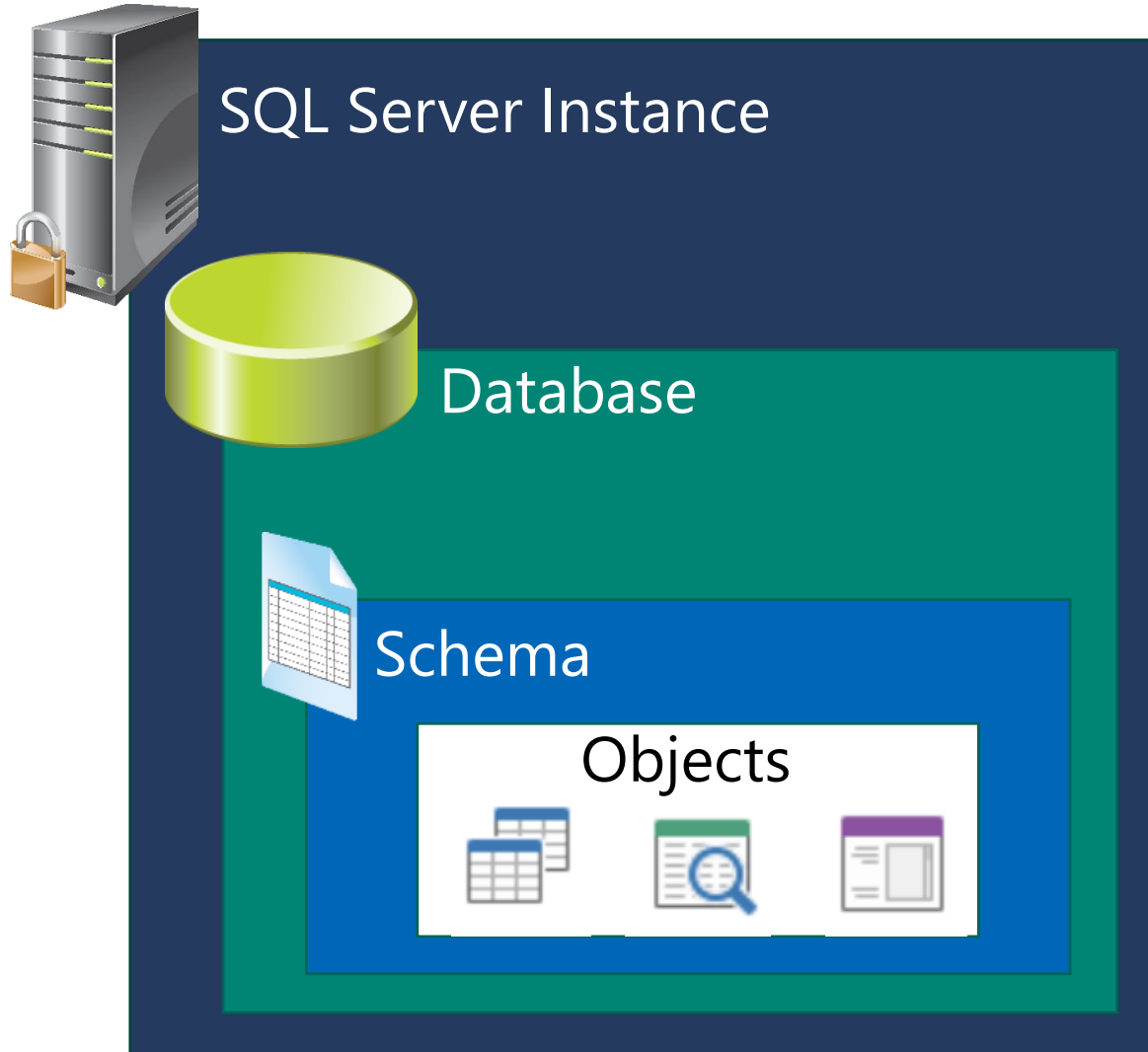


Securables

# SQL IaaS And SQL Managed Instance Security Principals



# Securables and the Four-Part Name of Objects



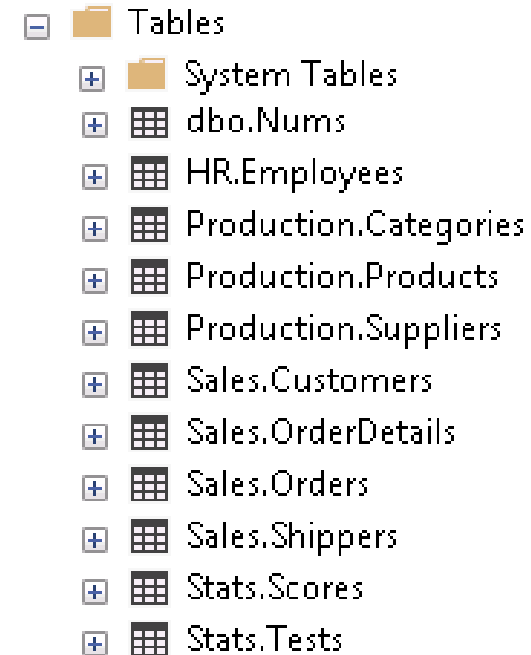
Server.Database.Schema.Object



# What is a Schema?

```
CREATE TABLE HR.Employees
(EmpID int IDENTITY PRIMARY KEY,
FirstName varchar(15) NOT NULL,
LastName varchar(20) NOT NULL,
JobTitle varchar(20) NOT NULL,
HireDate date DEFAULT GETDATE(),
BirthDate date NULL,
PhoneNumber varchar(15) UNIQUE,
DeptCode tinyint)
```

## Definition of an Object



## Organization of Objects

(For Management and Security)

Questions?



## Lesson 2: Authentication and Authorization

# Objectives

After completing this learning, you will be able to:

- Understand the authentication in Azure Managed Instance
- Understand the authorization process



# Authentication Available in Azure SQL Managed Instance

## Azure Active Directory Authentication

Uses identities managed by Azure Active Directory

## SQL Authentication

Authenticate a login on the local instance stored in the system master database

# Differences in Authentication Settings in SSMS IaaS vs PaaS

## Azure SQL Managed Instance

### Syntax

syntasql

Copy

```
-- Syntax for Azure SQL Managed Instance
CREATE LOGIN login_name [FROM EXTERNAL PROVIDER] { WITH <option_list> [,...]}

<option_list> ::=
    PASSWORD = {'password'}
    | SID = sid
    | DEFAULT DATABASE = database
```

Connect to Server

### SQL Server

Server type: Database Engine

Server name:

Authentication: Active Directory - Password  
Windows Authentication  
SQL Server Authentication  
Active Directory - Universal with MFA au  
Active Directory - Password  
Active Directory - Integrated

User name:

Password:

Connect Cancel Help

Server authentication

☐ Windows Authentication mode

☒ SQL Server and Windows Authentication mode

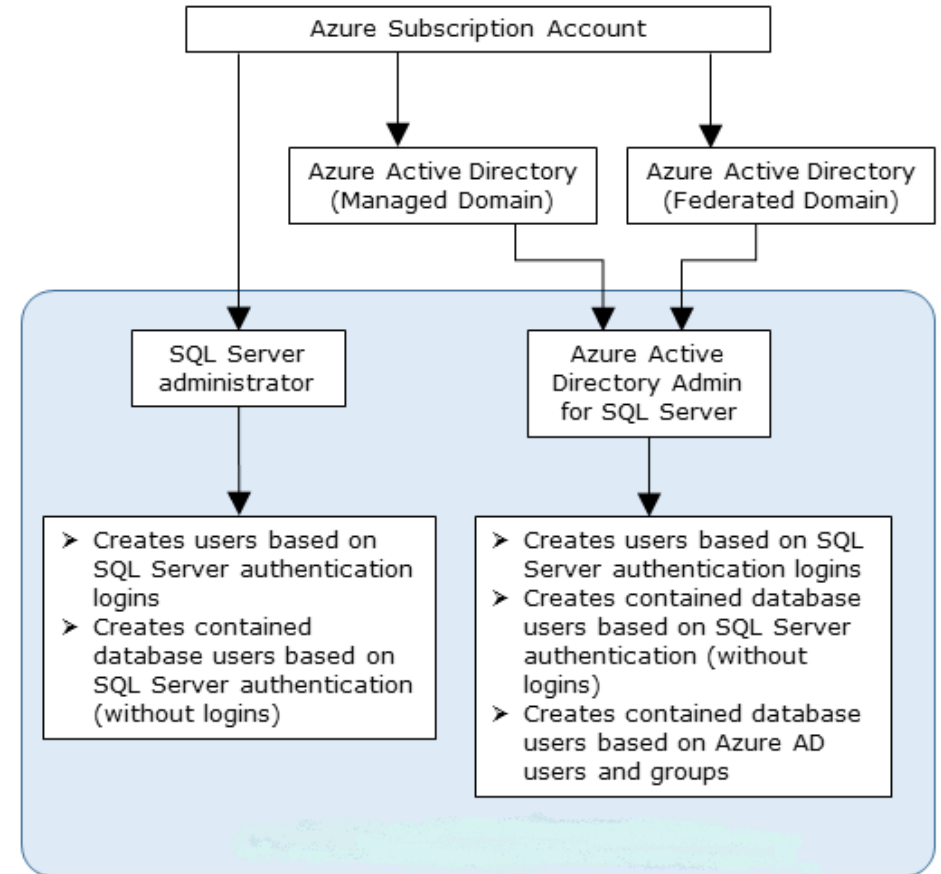
Login auditing

☐ None

☒ Failed logins only

☐ Successful logins only

☐ Both failed and successful logins



# Advantages of SQL Server Authentication



Allows applications that require SQL Server Authentication



Allows SQL Server to support environments with mixed operating systems, where all users are not authenticated by a centrally managed directory



Allows users to connect from unknown or untrusted systems



Allows SQL Server to support web-based applications where users create their own identities



Allows software developers to distribute their applications by using a complex permission hierarchy

# Authorization



Process by which SQL server decides whether a given principal can access a resource



Allows granting the specific permissions required rather than granting membership in a fixed role



Provides structural information and metadata of a securable only to those principals who have permission to access the securable



Allows creating custom permission sets



Works on the principle of *least privilege*



Questions?

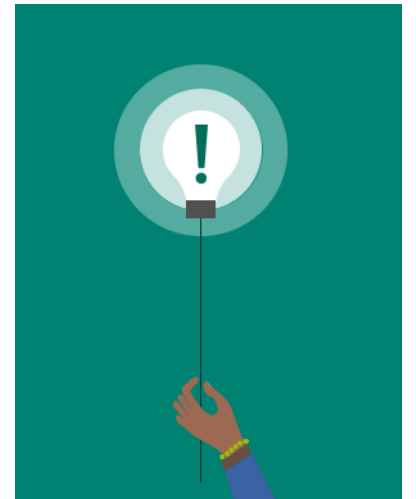


# Lesson 3: Azure Active Directory and RBAC

# Objectives

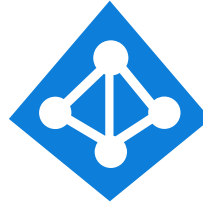
After completing this learning, you will be able to:

- Learn the Azure Active Directory and role-based access control in Azure SQL MI.



# Azure Active Directory

Your universal platform for managing and securing identities



## MODERNIZE ACCESS

Connect your users to any app

- Single Sign-On
- Azure AD Connect
- Automated User Lifecycle
- Self-Service for End Users
- Access from Anywhere



## SECURE & GOVERN

Safeguard user credentials

- Strong Authentication
- Conditional Access
- Identity Protection
- Privileged Identity Mgmt
- Identity Governance



## CONNECT & COLLABORATE

Interact with customers and partners

- Customer & Partner Identity and Access Management
- Cross-Tenant Collaboration
- Personalized Customer Journeys
- Connect with Any User

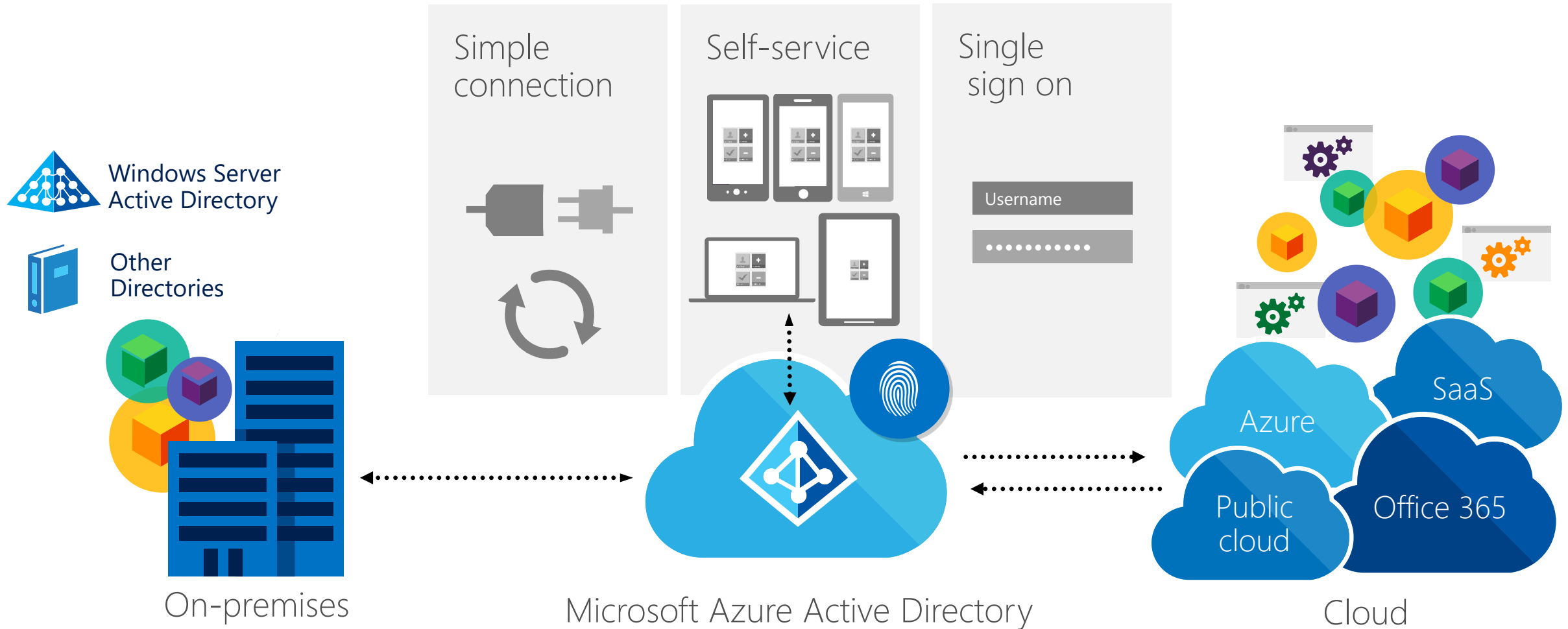


## DEVELOP & INTEGRATE

Accelerate adoption of your apps

- Open standards-based
- Identity platform
- App Integration
- Microsoft Graph
- Identity for IaaS

# Microsoft Azure Active Directory



# What's the difference between Active Directory and Azure Active Directory?

	Active Directory Domain Services	Azure Active Directory
User Management	Yes	Yes
Authentication	NTLM and Kerberos	OpenID Connect, SAML, OAuth
Groups	Yes	Yes
Object Hierarchy	Yes: X.500	No
Service Principals	Yes	Yes
Query AD programmatically	LDAP	AD Graph API (REST API)

# Azure role-based access control (Azure RBAC)

Azure role-based access control (Azure RBAC) helps you manage who has access to Azure resources, what they can do with those resources, and what areas they have access to.

Azure RBAC is an authorization system built on Azure Resource Manager that provides fine-grained access management of Azure resources.

- Allow one user to manage virtual machines in a subscription and another user to manage virtual networks
- Allow a DBA group to manage SQL databases in a subscription
- Allow a user to manage all resources in a resource group, such as virtual machines, websites, and subnets
- Allow an application to access all resources in a resource group

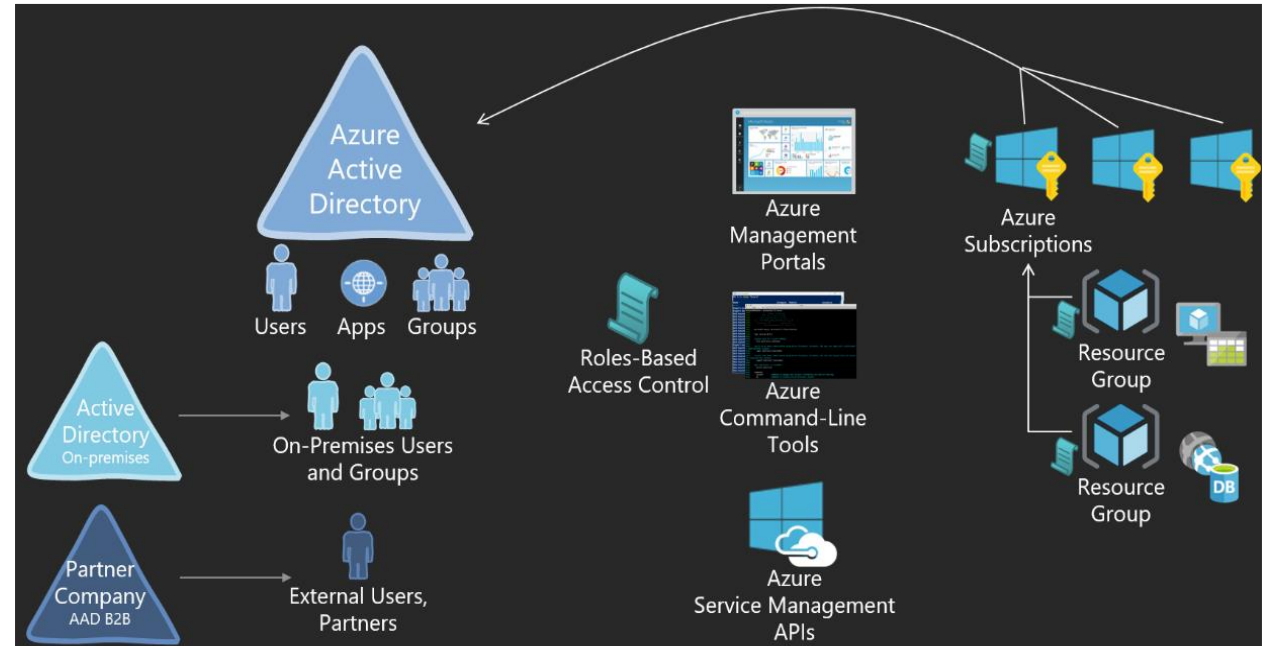
# Role Based Access Control

Used only for Azure administration

- Manage resources in Azure
- Azure AD is not an Azure resource

Roles composed of

- Actions
- Not Actions
- Scopes



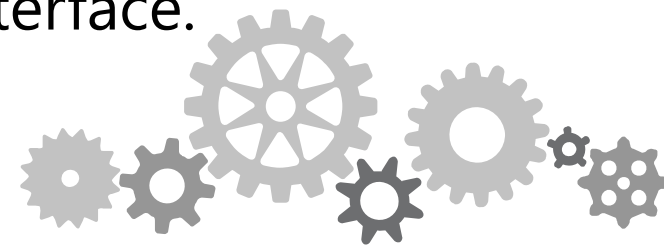


# RBAC – Roles

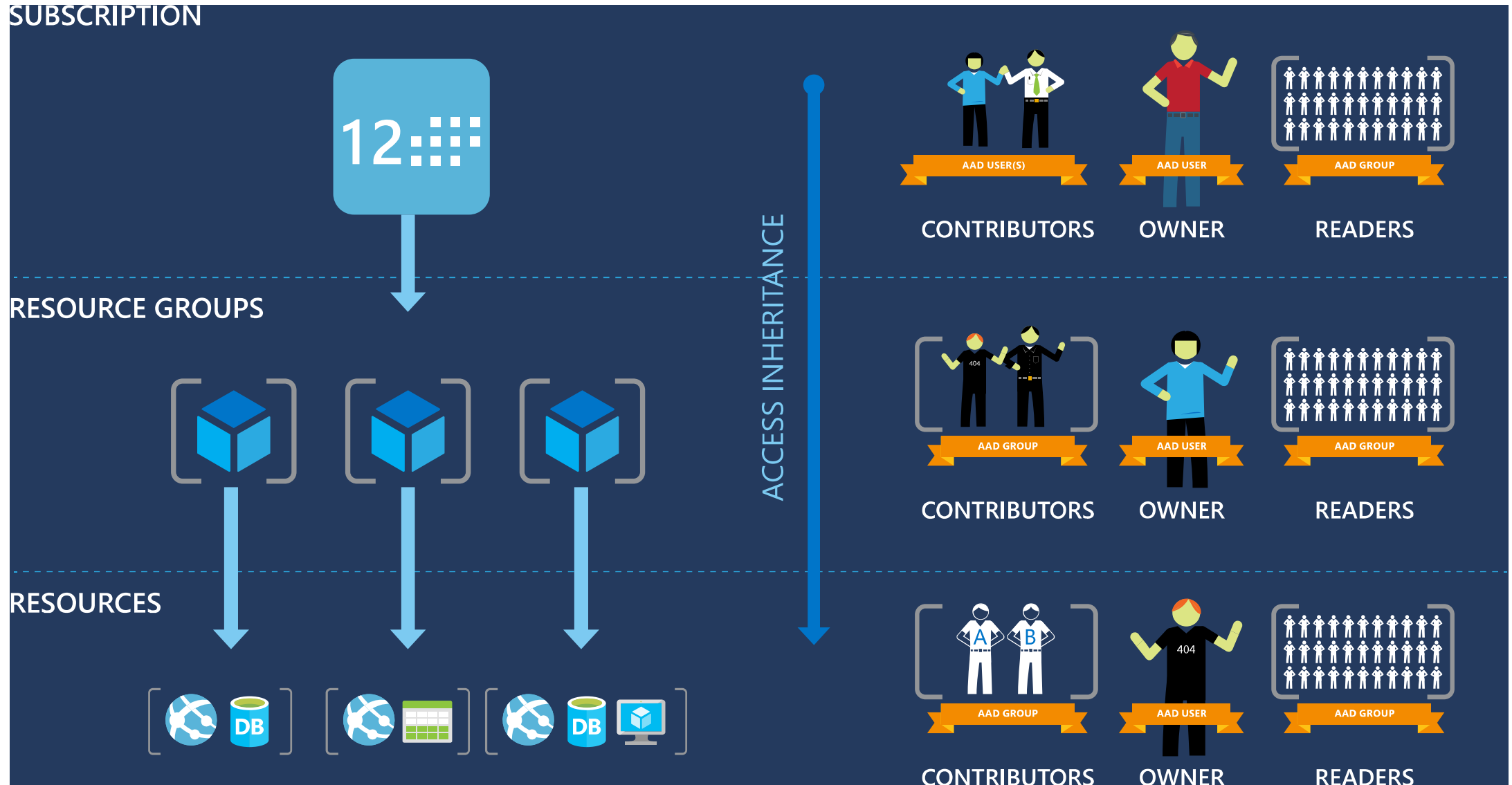
- **Built-in roles** - Azure RBAC has several built-in roles that you can assign to users , groups and service principals and managed identities. General and resource specific roles.

BUILT-IN ROLE	ACTIONS	NOT ACTIONS
Owner (allow all actions)	*	
Contributor (allow all actions except writing or deleting role assignments)	*	Microsoft.Authorization/*/Write, Microsoft.Authorization/*/Delete
Reader (allow all read actions)	*/Read	

- **Custom roles** - Custom roles can be created using Azure Portal , command-line tools in Azure PowerShell, and Azure Command-Line Interface.



# RBAC – Inheritance



# Azure AD Security Principals

## Roles can be assigned to:

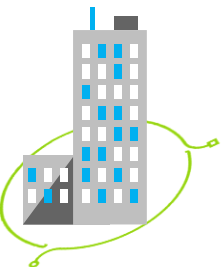
- Users
- Organizational users in Azure AD
- External Microsoft accounts (@outlook.com)

## Groups

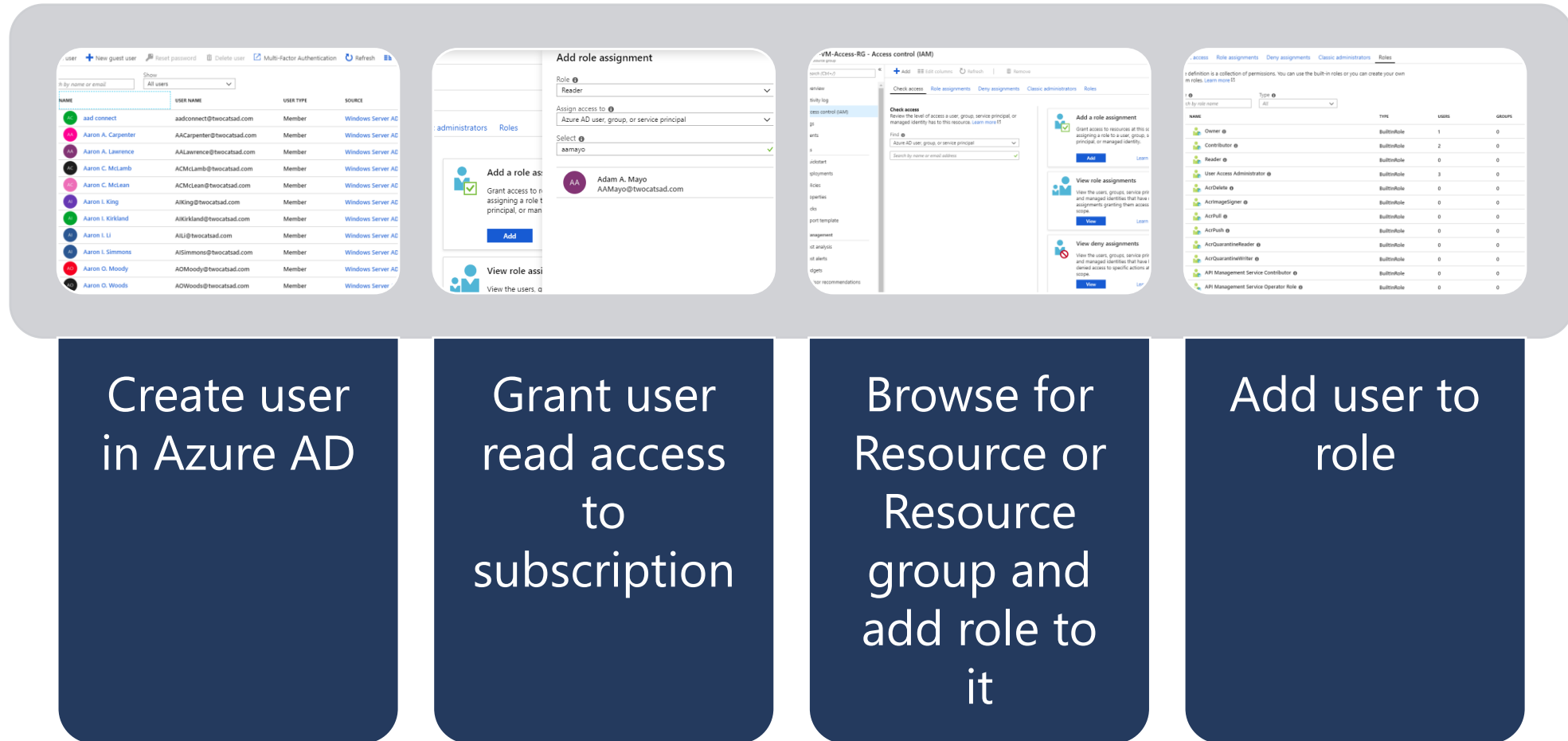
- Azure AD security groups
- Groups can be integrated with on-premises directories

## Service Principals

- Service identities are represented as service principals in Azure AD
- Assign to roles via Azure PowerShell cmdlets






# Basic Process for Adding Access



# RBAC Best Practices

Segregation of duties

Grant users the least privilege to get their work done

		Role			
		Reader	Resource-specific or custom role	Contributor	Owner
Scope	 Subscription	Observers	Users managing resources		Admins
	 Resource group				
	 Resource	Automated processes			

# Demonstration

## View user access to Azure Resources

- Show how to view and change the user access to Azure resources using IAM



Questions?



# Lesson 3: SQL MI Logins and Users



# Objectives

After completing this learning, you will be able to:

- Understand the difference between a login and user
- Understand how to create logins and users in SQL MI
- Understand the differences between logins and users



# Review Security Principals in SQL PaaS



Security principals are any login, user, group, or role within the server or database



Users within the databases are either mapped to a login, or contained users within the database



Contained users can be based on SQL Authentication or Azure Active Directory



Users can then be mapped to roles in order to give users centrally managed rights, or rights can be granted directly to a user

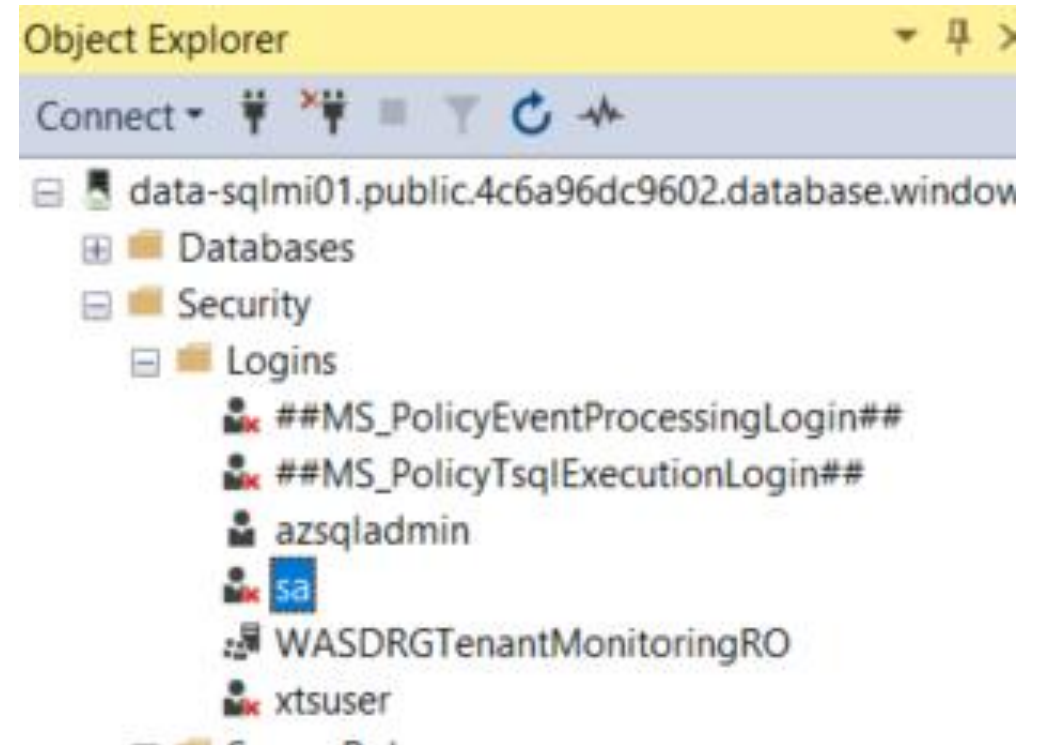
# Create accounts for non-administrator users

## Create a login:

- Create a SQL login in the master database
- `CREATE LOGIN login_name [FROM EXTERNAL PROVIDER] { WITH <option_list> [,..]}`

## Create a user account:

- Syntax can vary



# Demonstration

## Creating a Login for SQL MI



# Types of Administrative Logins for SQL MI

Azure Active Directory administrator account with full administrative permissions

Create SQL logins with full administrative permissions

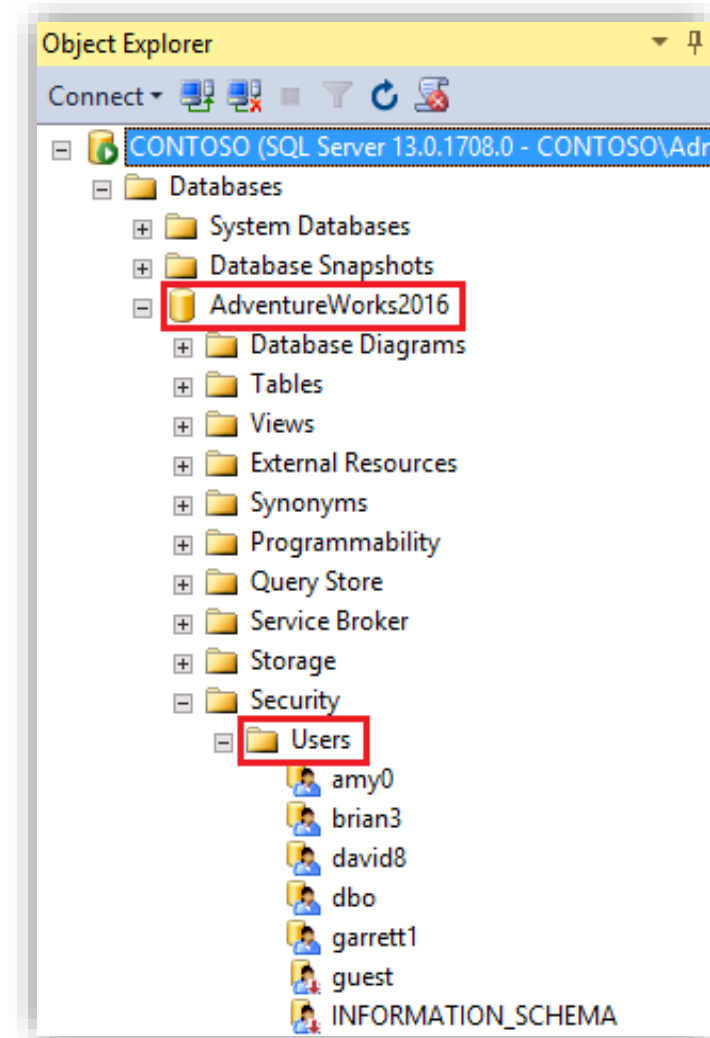
# Creating Users

Allow access to a database

Specific to a single database

Type of users:

- User based on logins in master
- SQL User with Password
- SQL User with Login
- SQL User without Password
- User mapped to a certificate
- User mapped to an asymmetric key
- Creating Azure AD contained users



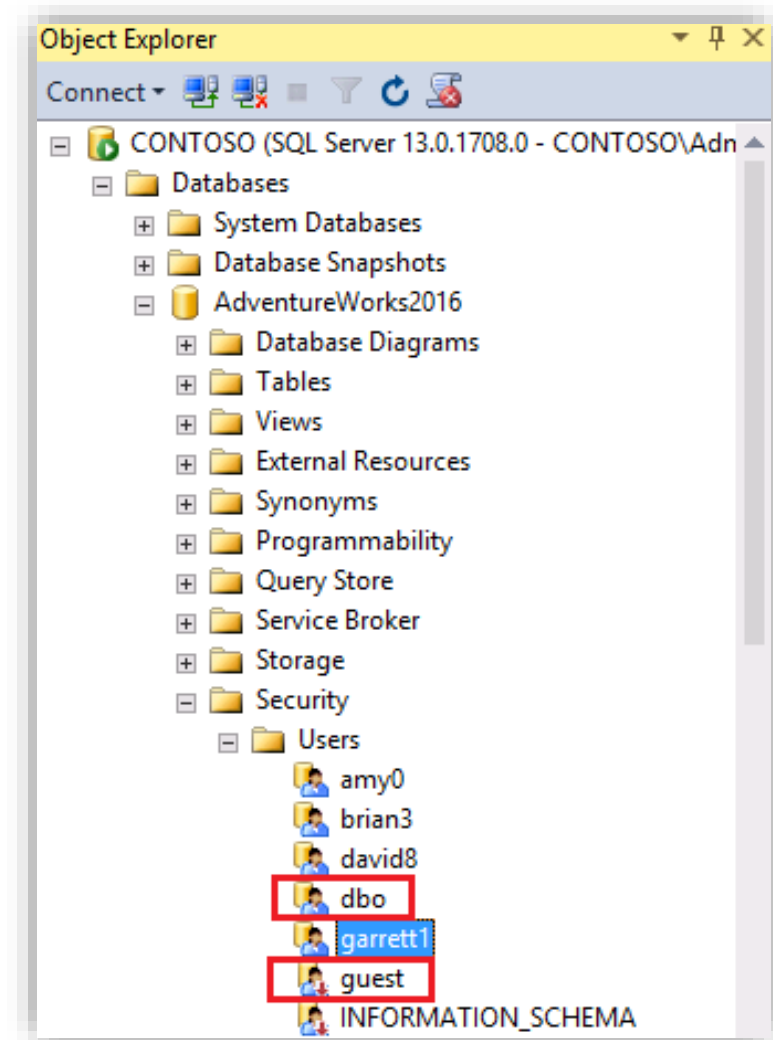
# DBO and Guest User

## DBO

- Performs all activities in the database
- Members of sysadmin role, SA login, and database owner are mapped to DBO
- Cannot be deleted

## Guest

- Allows logins without user accounts to access database
- Disabled by default in user databases
- Cannot be dropped but you can prevent it from accessing a database
- Must NOT be disabled in master and tempdb



Questions?





# Lesson 4: SQL MI Roles

# Objectives

After completing this learning, you will be able to:

- Create server and database roles
- Assign users and logins to roles
- Learn what are the default roles and when to use them
- Learn how to create credentials and how to use them



# Roles

## Server Roles

- Fixed server roles
- User-defined server roles

## Database Roles

- Fixed database roles
- User-defined database roles

## Application roles

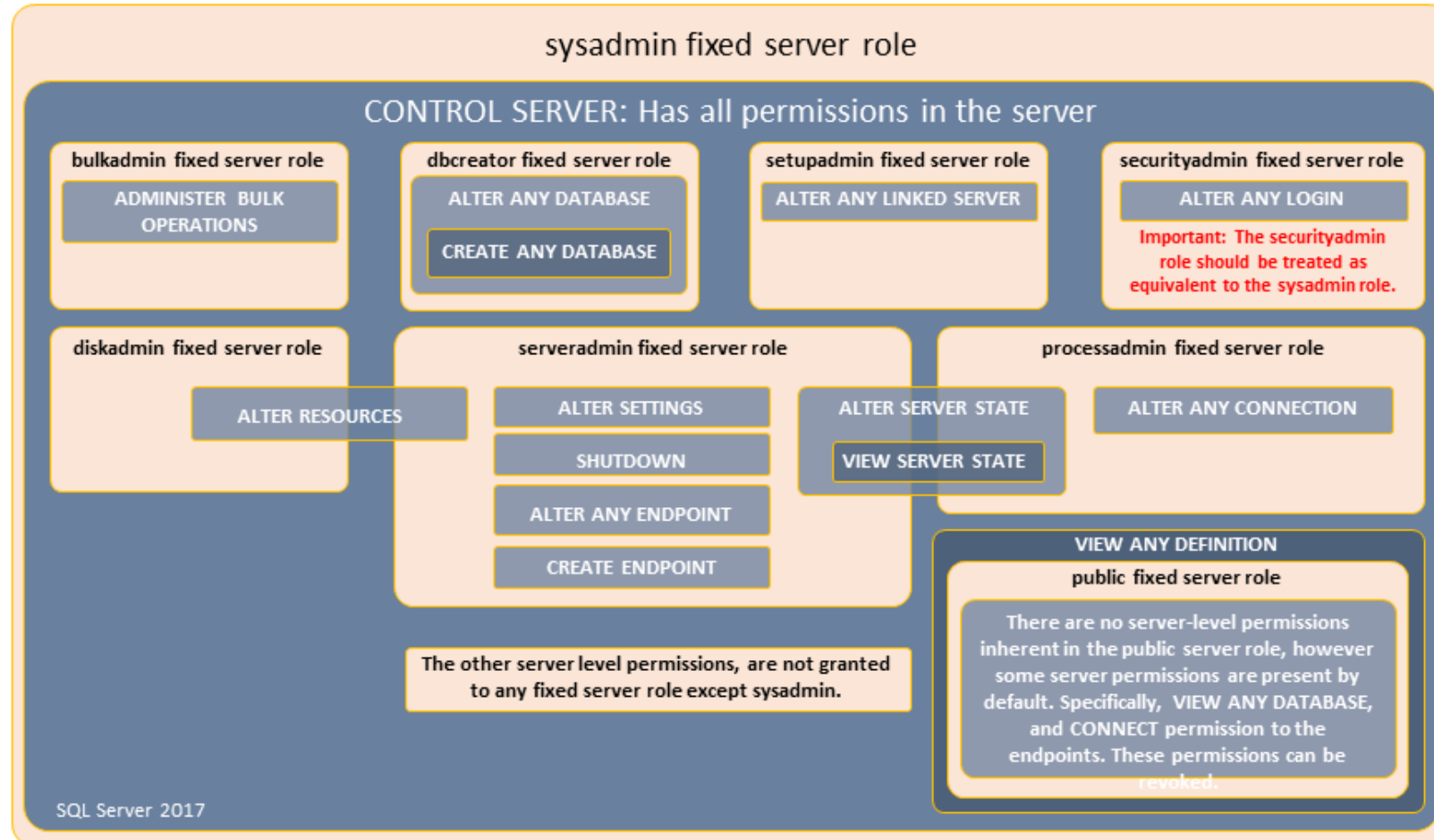
- Assign rights to applications instead of users

# Fixed server roles

sysadmin	Members of the sysadmin role can perform any activity on the server
Serveradmin	Members of the serveradmin role can change server-wide configuration settings (for example Max Server Memory) and can shutdown the server
Securityadmin	Members of the securityadmin role can manage logins and their properties (for example, changing the password of a login). This role should be treated as being equivalent to the sysadmin role
Processadmin	Members of the processadmin role can kill processes running inside of SQL Server
Setupadmin	Members of the setupadmin role can add and remove linked servers using T-SQL
Bulkadmin	Members of the bulkadmin role can run the BULK INSERT T-SQL statement
Diskadmin	Members of the diskadmin role have the ability to manage backup devices in SQL Server
Dbcreator	Members of the dbcreator role have the ability to create, restore, alter, and drop any database
Public	Every SQL Server login belongs to the public user role. Unlike the other fixed server roles, permissions can be granted, denied, or revoked from the public role



# Permissions of Fixed Server Roles

SERVER LEVEL ROLES AND PERMISSIONS: 9 fixed server roles, 34 server permissions



# Listing Server Level Permissions

```
SELECT * FROM sys.fn_builtin_permissions('SERVER')  
ORDER BY permission_name;
```

<div><div> Results</div><div> Messages</div></div>						
	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	SERVER	ADMINISTER BULK OPERATIONS	ADBO	CONTROL SERVER		
2	SERVER	ALTER ANY AVAILABILITY GROUP	ALAG	CONTROL SERVER		
3	SERVER	ALTER ANY CONNECTION	ALCO	CONTROL SERVER		
4	SERVER	ALTER ANY CREDENTIAL	ALCD	CONTROL SERVER		
5	SERVER	ALTER ANY DATABASE	ALDB	CONTROL SERVER		
6	SERVER	ALTER ANY ENDPOINT	ALHE	CONTROL SERVER		
7	SERVER	ALTER ANY EVENT NOTIFICATION	ALES	CONTROL SERVER		

# Public Role



Public is a special role that is at the server and database level.



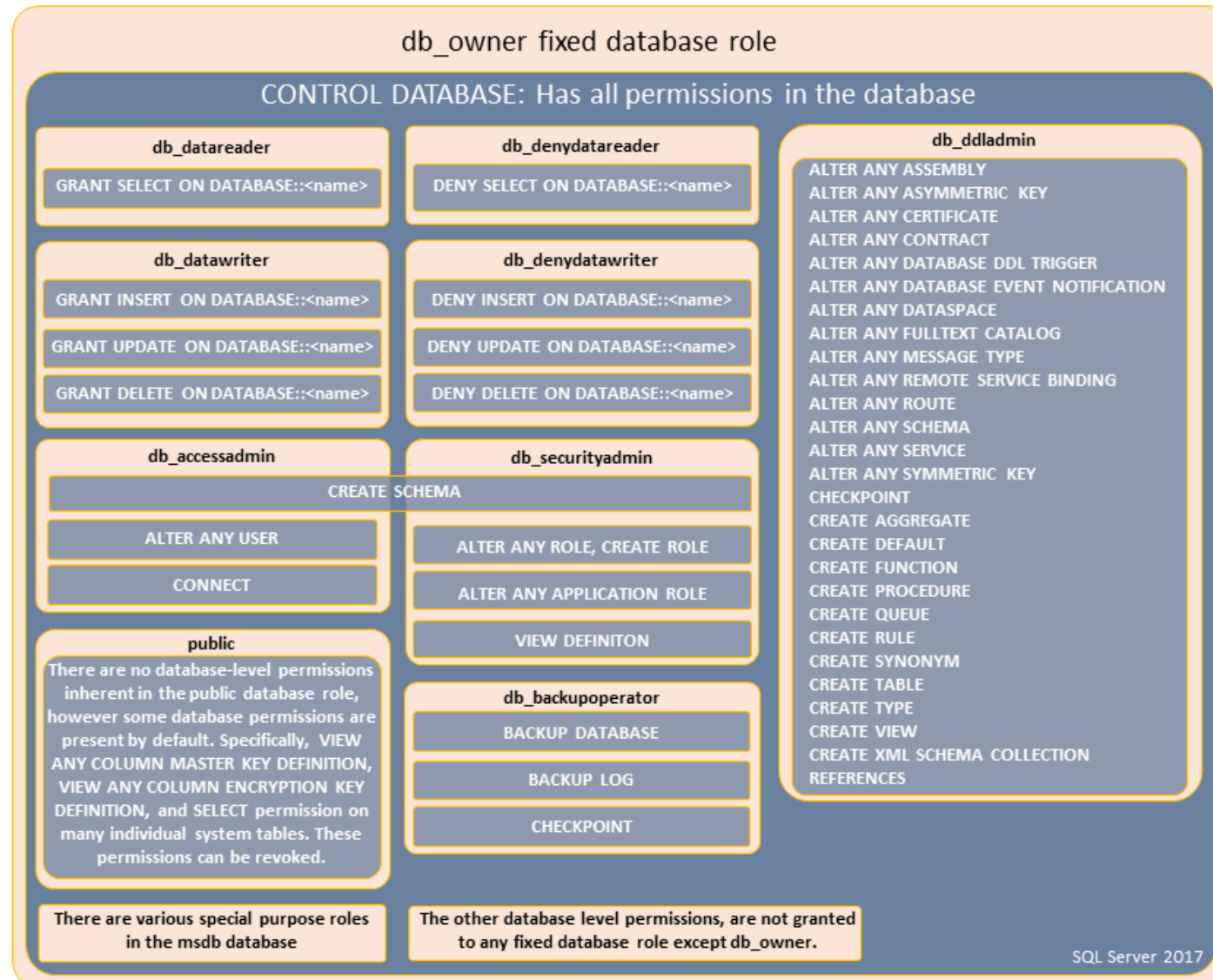
Every SQL Server login and user belongs to the Public role



Care must be taken when granting permissions to Public server role especially when granting server-level **permissions**.

# Permissions Assigned to the Fixed-Database Roles



DATABASE LEVEL ROLES AND PERMISSIONS: 11 fixed database roles, 77 database permissions





# Listing Database level permissions

```
SELECT * FROM sys.fn_builtin_permissions('Database')  
ORDER BY permission_name;
```

 Results  Messages						
	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	DATABASE	ALTER	AL	CONTROL	SERVER	ALTER ANY DATABASE
2	DATABASE	ALTER ANY APPLICATION ROLE	ALAR	ALTER	SERVER	CONTROL SERVER
3	DATABASE	ALTER ANY ASSEMBLY	ALAS	ALTER	SERVER	CONTROL SERVER
4	DATABASE	ALTER ANY ASYMMETRIC KEY	ALAK	ALTER	SERVER	CONTROL SERVER
5	DATABASE	ALTER ANY CERTIFICATE	ALCF	ALTER	SERVER	CONTROL SERVER
6	DATABASE	ALTER ANY COLUMN ENCRYPTION KEY	ALCK	ALTER	SERVER	CONTROL SERVER
7	DATABASE	ALTER ANY COLUMN MASTER KEY	ALCM	ALTER	SERVER	CONTROL SERVER

Questions?



# Lesson 5: SQL MI Permissions and Schemas

# Objectives

After completing this learning, you will be able to:

- Understand how to assign permissions to objects.
- Understand the concept of SQL Server schemas
- Understand how to apply security with schemas



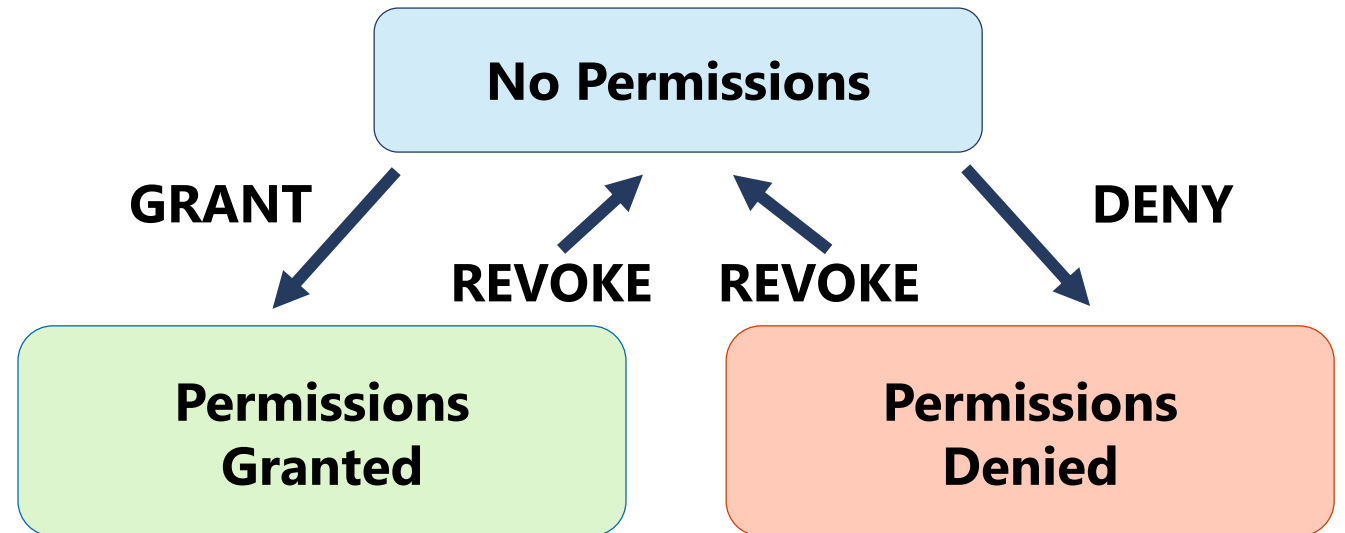
# Assigning Permissions

GRANT is used to assign a permission

DENY is used to explicitly deny a permission

- Used where permissions inherited through group or role membership
- Should only be used in exceptional circumstances

REVOKE removes either a GRANT or a DENY



# Assigning Permissions to Tables and Views

Grant with Grant allows the user to assign that permission.

Tables and Views can be assigned the same permissions.

Permissions for SELECT, UPDATE, and REFERENCES can also be set at the column level.

Select the Effective Tab to see what permissions have been granted.

Permissions for kenny:

Column Permissions...

Explicit Effective

Permission	Grantor	Grant	With Grant	Deny
Control		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Delete		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Insert		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
References		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Select		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Take ownership		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Update		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

# Security with Schemas

FQN has the form: ***server.database.schema.object***

In a database, all objects are created within a schema (dbo is default)

Allow their owners full control over objects within the schema

Permissions can be granted at the schema level

Can contain objects owned by multiple database users

Can be owned by any database principal

# Creating a Schema

```
CREATE SCHEMA Sprockets AUTHORIZATION Annik
CREATE TABLE NineProngs (source int, cost int)

GRANT SELECT ON SCHEMA::Sprockets TO Mandar
DENY SELECT ON SCHEMA::Sprockets TO Prasanna;
GO

ALTER SCHEMA Sprockets TRANSFER dbo.FourSporks
GO
```



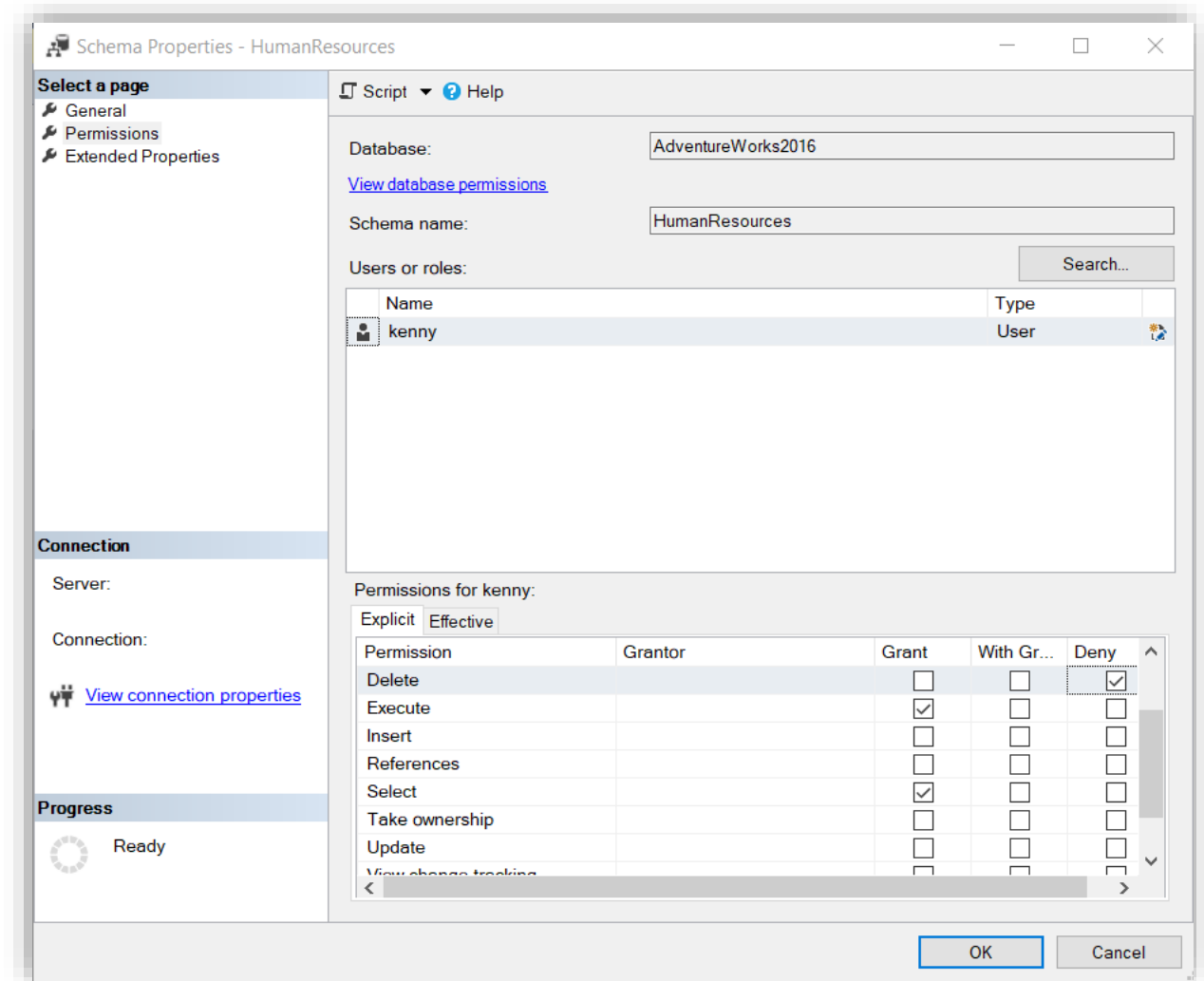
# Assign Permissions to a Schema

Permissions assigned at the schema level affect all objects belonging to that schema

Tables and Views can be assigned the same permissions

The Execute permission will be applied to all Stored Procedures in the schema

Select the Effective Tab to see what permissions have been granted



# Creating Synonyms

Creating a synonym will allow you to reference an object by an alternate name.

Useful for legacy applications that referenced objects with a dbo owner.

```
CREATE SYNONYM dbo.Employee FOR HumanResources.Employee
```

```
SELECT * FROM HumanResources.Employee
```

OR

```
SELECT * FROM dbo.Employee
```

Questions?



