# Identity Management

The Identity Management section defines the security control and standards for maintaining the security of ELC Information Technology to protect them from unauthorized access, modification, disclosure of identity management. The following industry good-practice frameworks, standards and guidelines were referenced from the standard section documents.

Section 3.1 of Azure Security Policies defines the IAM native control under “ELC Azure Security Policy Standards & Control Document of Azure Security Benchmark v2.1.0.”

# API Management should have username and password authentication disabled.

To enhance the security of your **developer portal** in **Azure API Management**, it’s advisable to **disable username and password authentication**. Instead, you can configure user authentication through **Azure AD** or **Azure AD B2C identity providers**. Let’s walk through the steps:

1. **Disable Default Username and Password Authentication**:
   * In your API Management instance, navigate to the **Developer portal**.
   * On the left menu, select **Identities**.
   * Choose **Username and password** as the **Provider type**.
   * [Click the context menu (usually represented by three dots), and then select Delete](https://learn.microsoft.com/en-us/azure/api-management/developer-portal-basic-authentication).
2. **Alternative Authentication Mechanisms**:
   * OAuth 2.0 is commonly used for API authorization. It secures communication between client apps, the API Management gateway, and backend APIs. [You can configure OAuth 2.0 authorization between the client and the gateway, the gateway, and the backend API, or both independently](https://learn.microsoft.com/en-us/azure/api-management/authentication-authorization-overview).
   * [Additionally, consider using TLS to protect credentials or tokens used for authentication or authorization](https://learn.microsoft.com/en-us/azure/api-management/authentication-authorization-overview).

Securing users access to APIs is just one aspect of safeguarding API Management environment. [For comprehensive security, explore the **Azure security baseline for API Management**](https://learn.microsoft.com/en-us/azure/api-management/authentication-authorization-overview)

Links –

* [Set up basic authentication to developer portal - Azure API Management | Microsoft Learn](https://learn.microsoft.com/en-us/azure/api-management/developer-portal-basic-authentication)
* [API authentication and authorization - Overview - Azure API Management | Microsoft Learn](https://learn.microsoft.com/en-us/azure/api-management/authentication-authorization-overview)

Note: - To supplement authentication and authorization, access to APIs should also be secured using TLS to protect the credentials or tokens that are used for authentication or authorization.

# An Azure Active Directory administrator should be provisioned for SQL servers.

To provision an **Azure Active Directory (AAD) administrator** for your SQL servers, follow these steps:

1. **Azure SQL Managed Instance with User-Assigned Managed Identity**:
   * If you’re using **Azure SQL Managed Instance**, you can create it with a **user-assigned managed identity** from **Microsoft Entra ID** (formerly Azure AD). This allows you to integrate Azure AD authentication with your SQL instance.
2. **Azure SQL Server on Azure VMs**:
   * For **SQL Server on Azure VMs**, you can set up **Microsoft Entra authentication**. Here’s how:
     + Use the **Azure CLI**, **PowerShell**, or an **ARM template** to set up a Microsoft Entra admin for your SQL Server.
     + In the Azure portal, select **SQL Server – Azure Arc**, and choose the instance for your SQL Server host.
     + Check the status of your SQL Server - Azure Arc resource to ensure it’s connected.
3. **Azure Active Directory Authentication for SQL Server 2022**:
   * If you’re using **Azure SQL Server 2022**, follow these steps:
     + In your **SQL Server – Azure Arc resource**, navigate to the **Azure Active Directory settings**.
     + Set up an **Azure AD administrator** for the SQL Server instance. This triggers the transfer of Azure AD administrator data to the Azure Arc agent, which then communicates it to the SQL Server instance.

Remember, integrating Azure AD administrators with your SQL servers enhances security and enables features like **Single Sign-On (SSO)** and **Multi-Factor Authentication (MFA)**. If you encounter any issues during the process, feel free to seek further assistance!

Links –

[Azure security baseline for Azure SQL | Microsoft Learn](https://learn.microsoft.com/en-us/security/benchmark/azure/baselines/azure-sql-security-baseline)

[Built-in policy definitions for Azure SQL Database - Azure SQL Database | Microsoft Learn](https://learn.microsoft.com/en-us/azure/azure-sql/database/policy-reference?view=azuresql)

[Microsoft Entra authentication - Azure SQL Database & Azure SQL Managed Instance & Azure Synapse Analytics | Microsoft Learn](https://learn.microsoft.com/en-us/azure/azure-sql/database/authentication-aad-overview?view=azuresql)

**Note -** For Azure SQL, Azure VMs, and SQL Server 2022, Microsoft Entra authentication only supports access tokens which originate from Microsoft Entra ID and doesn't support third-party access tokens. Microsoft Entra ID also doesn't support redirecting Microsoft Entra ID queries to third-party endpoints. This applies to all SQL platforms and all operating systems that support Microsoft Entra authentication.

# Virtual machines Guest Configuration extension should be deployed with system-assigned managed identity.

The Guest Configuration extension requires a system assigned managed identity. Azure virtual machines in the scope of this policy will be non-compliant when they have the Guest Configuration extension installed but do not have a system assigned managed identity.

When deploying the **Guest Configuration extension** for virtual machines, it’s essential to ensure that the VM has a **system-assigned managed identity**.

1. **Guest Configuration Extension**:
   * The Guest Configuration extension is a feature of **Azure Automanage** that performs audit and configuration operations inside virtual machines (VMs).
   * It checks policies within VMs, such as **Azure compute security baseline definitions** for both Linux and Windows.
   * To install this extension, you need to meet certain prerequisites.
2. **Prerequisites**:
   * Your VM must have a **system-assigned managed identity**.
   * You can satisfy this identity requirement by setting the "type": "SystemAssigned" property in the VM configuration. Here’s an example in JSON format:

**JSON**

"identity": {

"type": "SystemAssigned"

}

* + The machine configuration extension uses this identity to authenticate the VM as it reads and writes to the machine configuration service.

1. **Operating Systems and Internet Connectivity**:
   * The machine configuration extension supports both **Linux** and **Windows** operating systems.
   * The agent installed by the extension must be able to reach content packages listed by guest configuration assignments and report status to the machine configuration service.
   * The VM can connect via **outbound HTTPS over TCP port 443** or through a connection provided by **private networking**.
2. **Deployment Considerations**:
   * When installing and deploying the machine configuration extension, keep the following in mind:
     + **Instance Name**: Set the instance name of the extension to either "AzurePolicyforWindows" or "AzurePolicyforLinux". These specific strings are required for the security baseline definition policies.
     + **Versions**: By default, deployments update to the latest version. The autoUpgradeMinorVersion property helps with automatic upgrades when new versions are released.
     + **Automatic Upgrade**: You can enable automatic upgrades to the latest extension version.

System-assigned managed identity ensures proper authentication and compliance when using the Guest Configuration extension within your VMs.

Links –

[Azure Automanage machine configuration (guest configuration) - Azure Virtual Machines | Microsoft Learn](https://learn.microsoft.com/en-us/azure/virtual-machines/extensions/guest-configuration)

[Built-in policy definitions for Azure Virtual Machines - Azure Virtual Machines | Microsoft Learn](https://learn.microsoft.com/en-us/azure/virtual-machines/policy-reference)

[Azure Policy guest configuration extension - Cloud Adoption Framework | Microsoft Learn](https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/manage/azure-server-management/guest-configuration-policy)

[Managed identities for Azure resources - Managed identities for Azure resources | Microsoft Learn](https://learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/overview)

# Authentication to Linux machines should require SSH keys.

**SSH key-based authentication** is a highly secure method for accessing Linux servers via **Secure Shell (SSH)**.

1. **Generating SSH Keys**:
   * On your local computer, use the ssh-keygen utility (included with the standard OpenSSH suite of tools) to create an SSH key pair. By default, this generates a **3072-bit RSA key pair**.
   * Run the following command to generate the keys:
   * ssh-keygen
   * This will create both a **public key** (usually named id\_rsa.pub) and a **private key** (usually named id\_rsa) in your ~/.ssh/ directory.
2. **How SSH Keys Work**:
   * An SSH server can authenticate clients using various methods. Password authentication is common but less secure.
   * SSH key pairs consist of a public key (shared freely) and a private key (kept secret by the client).
   * When a client attempts to authenticate using SSH keys, the server checks if the client possesses the private key.
   * If successful, a shell session is spawned, or the requested command is executed.
3. **Setting Up SSH Key Authentication**:
   * Upload your **public key** to the remote server you want to log into. Add it to the ~/.ssh/authorized\_keys file within the user account you’ll be logging into.
   * The public key can be shared without negative consequences, as it only encrypts messages that the private key can decrypt.

Keep your **private key** secure and never share it. With SSH keys, you’ll enhance security and simplify server access!

**Links –**

[Use SSH keys to connect to Linux VMs - Azure Virtual Machines | Microsoft Learn](https://learn.microsoft.com/en-us/azure/virtual-machines/linux/ssh-from-windows)

# A maximum of 3 owners should be designated for your subscription.

This is a security best practice that reduces the potential for breach by a compromised. By monitoring the number of subscription owners using Azure Security Centre, you can enforce these best practices and always maintain a maximum of 3 subscription owners.

Links –

[Built-in policy definitions - Microsoft Defender for Cloud | Microsoft Learn](https://learn.microsoft.com/en-us/azure/defender-for-cloud/policy-reference)

[Security Control: Implement security best practices - Microsoft Community Hub](https://techcommunity.microsoft.com/t5/microsoft-defender-for-cloud/security-control-implement-security-best-practices/ba-p/2269914)

# Management ports of virtual machines should be protected with just-in-time network access control.

Protecting management ports of virtual machines is crucial for security. Let’s break down the recommendations for securing management ports:

1. **Network Security Groups (NSGs)**: These are essential for safeguarding your virtual machines. Consider restricting direct internet access to VMs by using Azure Policy to deny deployments with NICs containing public IP addresses. While there are valid reasons for having public-facing VMs, it’s important to protect them from RDP and SSH brute force attacks. NSGs can help restrict access to VMs and enhance security.
2. **Just-in-Time (JIT) Access**: Azure provides a feature called **Just-in-Time VM access**. By enabling JIT, you can block inbound traffic on specific ports. Defender for Cloud ensures “deny all inbound traffic” rules exist for selected ports in the network security group (NSG) and Azure Firewall rules. This restricts access to your VMs management ports and defends them from attacks.

Management ports don’t always need to be open and accessible. They should only be open when necessary for management or maintenance tasks. By implementing these recommendations, you’ll enhance your overall security posture and reduce exposure to potential threats.

**Links -**

[Security Control: Secure Management Ports - Microsoft Community Hub](https://techcommunity.microsoft.com/t5/microsoft-defender-for-cloud/security-control-secure-management-ports/ba-p/1505770)

[Understand just-in-time virtual machine access - Microsoft Defender for Cloud | Microsoft Learn](https://learn.microsoft.com/en-us/azure/defender-for-cloud/just-in-time-access-overview?tabs=defender-for-container-arch-aks)

[Enable just-in-time access on VMs - Microsoft Defender for Cloud | Microsoft Learn](https://learn.microsoft.com/en-us/azure/defender-for-cloud/just-in-time-access-usage)

# There should be more than one owner assigned to your subscription.

One of our best practices is to have more than one owner assigned to a subscription in order to have administrator access redundancy and. Additionally, we recommend having maximum of 3 owners, to reduce the potential for breach by a compromised owner. So, the recommendation is to have 2 owners per subscription.

When managing **Azure subscriptions**, it’s essential to consider access redundancy and security.

1. **Multiple Owners**:
   * While an **Azure subscription** can have only **one Account Administrator**, you can **add multiple owners**.
   * It’s a good practice to have **at least two owners** assigned to a subscription for redundancy and continuity.
2. **Azure Management Groups**:
   * If you have **many subscriptions**, consider creating a **management group hierarchy** to efficiently manage them.
   * Each management group can contain one or more subscriptions.
   * The hierarchy aligns with your organization’s structure and needs.
   * You can define up to **six levels** of management groups.
   * Policies and access controls applied at higher levels propagate down to lower levels, ensuring governance and security.
3. **Best Practices**:
   * **Create a root management group** and move existing subscriptions into it.
   * Consider creating separate management groups for **production** and **non-production** workloads.
   * As you add new subscriptions, they inherit appropriate controls from the management group hierarchy.

Having **multiple owners** helps distribute responsibilities and ensures smooth management of your Azure resources.

Links –

[Organize and manage multiple Azure subscriptions - Cloud Adoption Framework | Microsoft Learn](https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/organize-subscriptions)

# Audit usage of custom RBAC rules

When it comes to **auditing activity carried out by users with custom roles in Azure**, there are a few considerations:

1. **Azure Policy - “Audit usage of custom RBAC rules”**:
   * This policy aims to **audit built-in roles** such as ‘Owner,’ ‘Contributor,’ and ‘Reader,’ rather than custom RBAC (Role-Based Access Control) roles.
   * Custom roles are treated as an exception and require a **rigorous review and threat modeling** before implementation.
   * The JSON configuration for this policy enables auditing of custom roles, but the actual data logs are not specified.
2. **Types of Logs in Azure**:
   * To find audit actions performed by your RBAC permission holders, you can explore the following logs:
     + **Azure Active Directory reporting**: Provides information related to user activities.
     + **Platform logs**: Contain additional details about operations within Azure.
3. **Practical Steps**:
   * If you want to perform a non-grouped audit, consider using PowerShell:
     + Log into Azure and position yourself on the desired subscription.
     + Use the Invoke-AzureRmSubscriptionRBACAudit.ps1 script to gather relevant information.

Auditing custom RBAC roles helps maintain security and ensures proper access controls within your Azure environment.

Links –

[Azure Policy - Audit usage of custom RBAC rules - Microsoft Q&A](https://learn.microsoft.com/en-us/answers/questions/1200245/azure-policy-audit-usage-of-custom-rbac-rules)

[Azure Policy - "Audit usage of custom RBAC rules" and functionality - Microsoft Community Hub](https://techcommunity.microsoft.com/t5/azure/azure-policy-quot-audit-usage-of-custom-rbac-rules-quot-and/m-p/3789437)

# Role-Based Access Control (RBAC) should be used on Kubernetes Services

**Role-Based Access Control (RBAC)** is a crucial security mechanism in **Kubernetes (K8s)**. It allows you to define and enforce fine-grained access control policies for various resources within your Kubernetes cluster.

1. **What is RBAC?**
   * RBAC regulates access to computer or network resources based on the roles of individual users within your organization.
   * It uses the rbac.authorization.k8s.io API group to drive authorization decisions, allowing dynamic configuration of policies through the Kubernetes API
2. **RBAC Components:**
   * **Roles and ClusterRoles**:
     + An **RBAC Role** or **ClusterRole** contains rules representing a set of permissions.
     + Permissions are purely additive; there are no “deny” rules.
     + A **Role** sets permissions within a specific namespace, while a **ClusterRole** is non-namespaced.
     + ClusterRoles serve various purposes, including defining permissions on namespaced resources within individual namespaces or across all namespaces.
     + For example, a ClusterRole can allow a user to run kubectl get pods --all-namespaces.
   * **RoleBindings and ClusterRoleBindings**:
     + **RoleBindings** associate a **Role** with a user, group, or service account within a namespace.
     + **ClusterRoleBindings** link a **ClusterRole** to a user, group, or service account across the entire cluster.
     + These bindings grant access based on the defined roles.
3. **Example RBAC Configurations:**
   * **Role Example** (within a namespace):



* + **ClusterRole Example** (cluster-wide):

**A screenshot of a computer program

Description automatically generated**

This Cluster Role grants read access to secrets in any namespace or across all namespaces.

RBAC plays a critical role in managing and securing access to the Kubernetes API server and cluster resources. Implementing RBAC ensures proper authorization and helps maintain a secure Kubernetes environment.

Links –

[Using RBAC Authorization | Kubernetes](https://kubernetes.io/docs/reference/access-authn-authz/rbac/)

[Use Azure RBAC for Kubernetes Authorization - Azure Kubernetes Service | Microsoft Learn](https://learn.microsoft.com/en-us/azure/aks/manage-azure-rbac)

# &

# Function apps should use managed identity & App Service apps should use managed identity.

**Managed identities** are a powerful feature in **Azure** that allow your **App Service** and **Azure Functions** applications to securely access other resources without the need to manage credentials or secrets directly. Let’s dive into how you can set up managed identities for these services:

1. **System-Assigned Identity**:
   * A system-assigned identity is tied to your application and is deleted if your app is deleted.
   * An app can have only one system-assigned identity.
   * To enable it:
     + In the **Azure Portal**, navigate to your app’s page, scroll down to the **Settings** group, and select **Identity**.
     + Within the **System assigned** tab, switch the status to **On** and click **Save**.
     + Alternatively, you can use the **Azure CLI** or **Azure PowerShell** to create a system-assigned identity for your app.
   * This identity allows your app to easily access other resources, such as **Azure Key Vault**.
   * Keep in mind that managed identities don’t support cross-directory scenarios, so they won’t behave as expected if your app is migrated across subscriptions or tenants.
2. **User-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.
   * You can create and manage user-assigned identities separately from your app.
   * These identities are useful when you need to grant specific permissions to different resources.
   * To set up a user-assigned identity, create one in the Azure portal and then assign it to your app.
3. **Azure Functions and Managed Service Identity**:
   * With a managed identity from **Azure Active Directory (AAD)**, your **Azure Function App** can access other AAD-protected resources, such as **Key Vault**.
   * To enable it:
     + In the **Azure Portal**, navigate to your Function App, click **Platform features**, and then select **Identity**.
     + Switch the **System assigned** option to **On**.
   * This allows your Function App to securely interact with other resources without managing explicit credentials.

Managed identities simplify security and reduce the risk of credential leakage.

Links –

[Managed identities - Azure App Service | Microsoft Learn](https://learn.microsoft.com/en-us/azure/app-service/overview-managed-identity?tabs=portal%2Chttp)

[Using Managed Service Identity (MSI) to authenticate on Azure SQL DB - Microsoft Community Hub](https://techcommunity.microsoft.com/t5/azure-database-support-blog/using-managed-service-identity-msi-to-authenticate-on-azure-sql/ba-p/1288248)

[Managed Service Identity support for App Service and Azure Functions | Azure updates | Microsoft Azure](https://azure.microsoft.com/en-us/updates/managed-service-identity-support-for-app-service-and-azure-functions/)

# Accounts with write permissions on Azure resources should be MFA enabled.

Ensuring **Multi-Factor Authentication (MFA)** for accounts with write permissions on Azure resources is crucial for enhancing security. Here are the recommended steps to achieve this:

1. **Security Defaults (Free Option)**:
   * If you’re using the free edition of **Microsoft Azure AD**, consider enabling **security defaults**. This will enforce MFA for all users in your tenant.
   * Security defaults are suitable for basic MFA requirements and provide a simple way to enhance security.
2. **Per-User Assignment (Microsoft 365 Business, E3, or E5 Customers)**:
   * For Microsoft 365 customers, use **per-user assignment**.
   * With per-user assignment, you can enable or disable MFA for all users across all sign-in events. Management is done through the Office 365 portal.
3. **Conditional Access (CA) Policy (Microsoft Azure AD P1 or P2 Customers)**:
   * Upgrade to **Microsoft Azure AD P1 or P2** for an improved user experience and more granular control.
   * Configure a **CA policy** to prompt users for MFA during specific scenarios or events based on your business requirements.
   * Your CA policy should:
     + Enforce MFA.
     + Include the **Microsoft Azure Management app ID** (797f4846-ba00-4fd7-ba43-dac1f8f63013) or exclude it as needed.
     + Microsoft Azure AD P2 provides the strongest security features and an improved user experience.

MFA significantly reduces the risk of unauthorized access and helps protect your Azure resources. Implementing MFA ensures that even if passwords are compromised, an additional layer of authentication prevents unauthorized account access.

Links –

[Built-in policy definitions for Azure Resource Manager - Azure Resource Manager | Microsoft Learn](https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/policy-reference)

[Security recommendations for multifactor authentication - Microsoft Defender for Cloud | Microsoft Learn](https://learn.microsoft.com/en-us/azure/defender-for-cloud/multi-factor-authentication-enforcement)

# Azure SQL Database should have Microsoft Entra-only authentication.

**Microsoft Entra-only authentication** is a security feature available for **Azure SQL Database** and **Azure SQL Managed Instance**. When enabled, it ensures that only **Microsoft Entra** (formerly known as **Azure Active Directory**) authentication is allowed, and SQL authentication based on credentials is disabled at the server or managed instance level.

Here are the steps to enable Microsoft Entra-only authentication:

1. **Prerequisites**:
   * You need a **Microsoft Entra tenant**.
   * Ensure you have a **SQL Database** or **SQL Managed Instance** with at least one database and existing logins or users.
2. **Assign Role**:
   * Sign in to the **Azure portal** using a privileged user who can assign Microsoft Entra roles.
   * Navigate to your **SQL server resource**.
   * Go to **Access control (IAM)** in the menu.
   * Click **Add** and then select **Add role assignment** from the drop-down menu.
   * In the **Add role assignment** pane:
     + Choose the **Role SQL Security Manager**.
     + Select the user who should have the ability to enable or disable Microsoft Entra-only authentication (e.g., UserSqlSecurityManager@contoso.onmicrosoft.com).
     + Click **Save**.
3. **Enable Microsoft Entra-only authentication**:
   * You can enable it using the **Azure portal**, **Azure CLI**, or **PowerShell**.
   * For the Azure portal:
     + Sign in with the user having the **SQL Security Manager** role.
     + Go to your **SQL server resource**.
     + Under the **Settings** menu, select **Microsoft Entra ID**.
   * [Follow the specific steps for enabling it using the Azure CLI or PowerShell from the official tutorial](https://learn.microsoft.com/en-us/azure/azure-sql/database/authentication-azure-ad-only-authentication-tutorial?view=azuresql).

[Microsoft Entra-only authentication restricts access to Microsoft Entra users only, enhancing security for your Azure SQL resources](https://learn.microsoft.com/en-us/azure/azure-sql/database/authentication-azure-ad-only-authentication?view=azuresql).

Links –

[Enable Microsoft Entra-only authentication - Azure SQL Database & Azure SQL Managed Instance | Microsoft Learn](https://learn.microsoft.com/en-us/azure/azure-sql/database/authentication-azure-ad-only-authentication-tutorial?view=azuresql&tabs=azure-portal)

[Configure Microsoft Entra authentication - Azure SQL Database & SQL Managed Instance & Azure Synapse Analytics | Microsoft Learn](https://learn.microsoft.com/en-us/azure/azure-sql/database/authentication-aad-configure?view=azuresql&tabs=azure-powershell)

[Create server with Microsoft Entra-only authentication enabled - Azure SQL Database & Azure SQL Managed Instance | Microsoft Learn](https://learn.microsoft.com/en-us/azure/azure-sql/database/authentication-azure-ad-only-authentication-create-server?view=azuresql&tabs=azure-portal)

# Managed identities in Azure Data Factory.

1. **Overview of Managed Identity**:
   * Managed identities eliminate the need to manage credentials.
   * They provide an identity for the service instance when connecting to resources that support **Microsoft Azure authentication**.
   * For instance, a managed identity can be used to access resources like **Azure Key Vault**, where data admins securely store credentials or access storage accounts.
   * There are **two types** of supported managed identities:
     + **System-assigned**: Enabled directly on a service instance during its creation. An identity is created in Microsoft Azure tied to that service instance’s lifecycle. When the resource is deleted, Azure automatically removes the identity.
     + **User-assigned**: Created as a standalone Azure resource and can be assigned to one or more instances of a data factory. In this case, the identity is managed separately from the resources that use it.
2. **Benefits of Managed Identity**:
   * **Credential Storage**: Managed identity allows you to store credentials in **Azure Key Vault**.
   * **Authentication**: It enables access to data stores or computes using managed identity authentication for services like **Azure Blob storage**, **Azure Data Explorer**, **Azure Data Lake Storage Gen1**, **Azure Data Lake Storage Gen2**, **Azure SQL Database**, **Azure SQL Managed Instance**, **Azure Synapse Analytics**, REST APIs, Databricks activities, and more.
   * **Encryption**: Managed identity is also used for encrypting/decrypting data and metadata using the customer-managed key stored in Azure Key Vault, providing double encryption.
3. **System-Assigned Managed Identity**:
   * During service creation, you can allow a system-assigned managed identity.
   * An identity is created in Microsoft Azure tied to that service instance’s lifecycle.
   * When the resource is deleted, Azure automatically removes the identity.
   * This type of managed identity is also referred to as ‘Managed identity’ elsewhere in the documentation and in the Data Factory Studio for backward compatibility purposes.

If you’re setting up a linked service in Azure Data Factory for **Azure SQL**, consider using **system-assigned managed identity authentication**. Here are the steps:

1. Navigate to Azure Data Factory.
2. Click on **New Linked Service**.
3. Search for **Azure SQL Database**.
4. Specify your **Azure subscription**, **Azure SQL Server name**, and **Database name**.

Links –

[Compute environments - Azure Data Factory & Azure Synapse | Microsoft Learn](https://learn.microsoft.com/en-us/azure/data-factory/compute-linked-services)

[Using credentials - Azure Data Factory & Azure Synapse | Microsoft Learn](https://learn.microsoft.com/en-us/azure/data-factory/credentials?tabs=data-factory)

[Using Managed Service Identity (MSI) to authenticate on Azure SQL DB - Microsoft Community Hub](https://techcommunity.microsoft.com/t5/azure-database-support-blog/using-managed-service-identity-msi-to-authenticate-on-azure-sql/ba-p/1288248)

[Managed identity - Azure Data Factory | Microsoft Learn](https://learn.microsoft.com/en-us/azure/data-factory/data-factory-service-identity)

[az datafactory linked-service | Microsoft Learn](https://learn.microsoft.com/en-us/cli/azure/datafactory/linked-service?view=azure-cli-latest)

# Configure Azure Event Grid partner namespaces to disable local authentication.

To **disable local authentication** for your Azure Event Grid partner namespaces, follow these steps:

1. **Navigate** to your partner namespace in the **Azure portal**.
2. On the **Overview** page, in the **Essentials** section, select the current value for **Local Authentication**.
3. On the **Local Authentication** page, choose either **Enabled** or **Disabled** based on your requirements.
4. Select **OK** to close the **Local Authentication** page.

Additionally, consider implementing the following Azure Policy built-in definitions for enhanced security:

* **Azure Event Grid domains should have local authentication methods disabled**: This ensures that your Azure Event Grid domains exclusively require **Azure Active Directory identities** for authentication.
* **Azure Event Grid domains should use private link**: By mapping private endpoints to your Event Grid domain, you’ll be protected against data leakage risks.

Link –

[Built-in policy definitions for Azure Event Grid - Azure Event Grid | Microsoft Learn](https://learn.microsoft.com/en-us/azure/event-grid/policy-reference)

[az eventgrid partner configuration | Microsoft Learn](https://learn.microsoft.com/en-us/cli/azure/eventgrid/partner/configuration?view=azure-cli-latest)