

Connect via Private Endpoints

Last updated by | Vitor Tomaz | Mar 14, 2023 at 12:53 PM PDT

Contents

- [Readiness training](#)
- [Self-help content presented in Azure Portal](#)
 - [Learn how to connect an application to a Managed Instance...](#)
 - [When to use private endpoints](#)
 - [How does a private endpoint differ from VNet-local endpoint...](#)
 - [Limitations](#)
 - [1. Create a private endpoint in a virtual network](#)
 - [2. Review and approve a request to create a private endpoint...](#)
 - [3. Set up domain name resolution for private endpoint](#)
 - [Troubleshooting connectivity issues](#)
 - [Test connectivity using Azure SQL Connectivity Checker](#)
 - [Resources](#)

Readiness training

[SQL MI Private Link readiness session](#) 

Self-help content presented in Azure Portal



(This content was shown to the customer during case submission. It's also visible on 'Diagnose and solve problems' blade.)

Learn how to connect an application to a Managed Instance via Private Endpoints

A private endpoint is like extending a physical network cable from a computer running Azure SQL Managed Instance to another virtual network. This connectivity path is established virtually via Azure Private Link technology. This way, your Azure SQL Managed Instance becomes available in a different virtual network without having to set up network peering or turn on the instance's public endpoint.

Scan the following headings, and select one or more to learn more about how to connect an application to a Managed Instance via private endpoints.

When to use private endpoints

Private endpoints to Azure SQL Managed Instance allow you to implement important connectivity scenarios more easily and securely than by using [VNet-local endpoint](#)  or [public endpoint](#) . These scenarios include:

- **Airlock.** Private endpoints to Azure SQL Managed Instance are deployed in a virtual network with jump servers and an ExpressRoute gateway, providing security and isolation between on-premises and cloud

resources.

- **Hub and spoke topology.** Private endpoints in spoke virtual networks conduct traffic from SQL clients and applications to Azure SQL Managed Instances in a hub virtual network, establishing clear network isolation and separation of responsibility.
- **Publisher-consumer.** Publisher tenant (for example, an ISV) manages multiple Azure SQL Managed Instances in their virtual networks. Publisher creates private endpoints in other tenants' virtual networks to make instances available to their consumers.
- **Integration of Azure PaaS and SaaS services.** Some PaaS and SaaS services, like Azure Data Factory, can create and manage private endpoints to Azure SQL Managed Instance.

The benefits of using private endpoints over a VNet-local or public endpoint include:

- **IP address predictability:** a private endpoint to Azure SQL Managed Instance is assigned a fixed IP address from its subnet's address range. This IP address remains static even if the IP addresses of VNet-local and public endpoints change.
- **Granular network access:** a private endpoint is only visible inside its virtual network.
- **Strong network isolation:** In a peering scenario, peered virtual networks establish two-way connectivity, while private endpoints are unidirectional and don't expose network resources inside their network to Azure SQL Managed Instance.
- **Avoiding address overlap:** peering multiple virtual networks requires careful IP space allocation and can pose a problem when address spaces overlap.
- **Conserving IP address real estate:** a private endpoint only consumes one IP address from its subnet's address space.


There are a couple of caveats:

- Private endpoint only allows connections to port 1433, which means that more involved connectivity, automation and integration scenarios can't be implemented over it. These scenarios include linked server scenarios, MI Link, failover groups, and similar.
- In preview, private endpoints require special setup to configure the correct DNS resolution required for Azure SQL Managed Instance.

How does a private endpoint differ from VNet-local endpoint?

The default, VNet-local endpoint deployed with each Azure SQL Managed Instance behaves as if a computer running the service were physically attached to your virtual network. It allows near-complete traffic control via route tables, network security groups, DNS resolution, firewalls, and similar mechanisms. You can also use this endpoint to involve your instance in scenarios requiring connectivity on ports other than 1433, such as auto-failover groups, distributed transactions, and MI Link. However, great flexibility that this endpoint provides comes with complexity in configuring it for particular scenarios, especially those involving multiple virtual networks or tenants.

By contrast, setting up a private endpoint is like extending a physical network cable from a computer running Azure SQL Managed Instance to another virtual network. This connectivity path is established virtually via Azure Private Link technology. It only allows connections in one direction: from the private endpoint to Azure SQL Managed Instance; and it only carries traffic on port 1433 (the standard TDS traffic port). In this way, your Azure SQL Managed Instance becomes available in a different virtual network without having to set up network peering or turn on the instance's public endpoint.

For a more detailed discussion of the different types of endpoints supported by Azure SQL Managed Instance, see [Communication overview](#) .

Limitations




- Azure SQL Managed Instance requires the exact instance hostname to appear connection string. Using private endpoint's IP address instead will fail. To resolve, configure your DNS server, or use a private DNS zone.
- Automatic registration of DNS names is disabled while in preview.
- Private endpoints to SQL Managed Instance can only be used to connect to port 1433, the standard TDS port for SQL traffic. More complex connectivity scenarios requiring communication on other ports must be established via instance's VNet-local endpoint.

Section Set up a private endpoint to Azure SQL Managed Instance:

To set up a private endpoint to Azure SQL Managed Instance, you need to complete the following steps:

1. Create a private endpoint in a virtual network

Private endpoints can be created using the Azure portal, PowerShell, or the Azure API:

- [The portal](#) 
- [PowerShell](#) 
- [CLI](#) 

After creating a private endpoint, you may need to approve its creation in the target virtual network, and you'll need to configure domain name resolution. Otherwise, login attempts will fail.

2. Review and approve a request to create a private endpoint

After creating a private endpoint, you may need to approve its creation in the target virtual network.

Once a request to create a private endpoint is made, SQL administrator can manage the private endpoint connection to Azure SQL Managed Instance. The first step to managing a new private endpoint connection is to review and approve it. This step is automatic if the user or service creating the private endpoint has sufficient Azure RBAC permissions on the Azure SQL Managed Instance resource. If they don't, then the review and approval must be done manually:

1. Navigate to your Azure SQL Managed Instance in Azure portal.
2. In the sidebar, select Private endpoint connections.
3. Review the connections in Pending state and select one or more private endpoint connections to approve or reject.
4. Approve or reject selected private endpoint connection(s) with an optional text response.
5. After you approve or reject connections, the list will reflect the state of selected private endpoint connection(s) along with any text response.

3. Set up domain name resolution for private endpoint

After you create a private endpoint to Azure SQL Managed Instance, you'll need to configure domain name resolution. Otherwise, login attempts will fail. The method below works for virtual networks that use Azure DNS

resolution. If your virtual network is configured to use a custom DNS server, adjust the steps accordingly.



To set up domain name resolution for private endpoint to an instance whose FQDN is `<instance-name>.<dns-zone>.database.windows.net`, let's consider two different virtual networks:

- Instance virtual network: hosts Azure SQL Managed Instance.
- Endpoint virtual network: hosts the private endpoint to the Azure SQL Managed Instance.

Steps to set up domain name resolution differ depending on whether the instance and endpoint virtual network are different or the same (or peered).

Separate virtual networks

Follow these steps if the instance and endpoint virtual networks are different and not peered. After you complete these steps, SQL clients connecting to `<instance-name>.<dns-zone>.database.windows.net` from inside the endpoint virtual network will be transparently routed through the private endpoint.

1. Obtain the IP address of the private endpoint either by visiting Private Link Center or by performing the following steps:
 1. Navigate to your Azure SQL Managed Instance in Azure portal.
 2. In the sidebar, select Private endpoint connections.
 3. Locate the private endpoint connection in the table and select its name.
 4. In Overview, select the network interface.
 5. In Overview, Private IP address is shown in the Essentials section.
2. [Create a private Azure DNS zone](#)  named `privatelink.<dns-zone>.database.windows.net`.
3. [Link the private DNS zone to the endpoint virtual network](#) .
4. In the DNS zone, create a new record set with the following values:
 - Name: `<instance-name>`
 - Type: A
 - IP address: IP address of the private endpoint obtained in step 1.



Same or peered virtual networks

Follow these steps if the instance and endpoint virtual networks are the same or are peered.

After you complete these steps, SQL clients inside the endpoint virtual network whose connection string includes `Encrypt=false` and `TrustServerCertificate=true` can connect to the private endpoint. They can still connect to the VNet-local endpoint at `<instance-name>.<dns-zone>.database.windows.net` without modifications.

Note that when the instance and endpoint virtual networks are the same (or peered), you don't have the ability to establish trusted encrypted connections and to transparently re-route SQL clients via the private endpoint.

1. Obtain the IP address of the private endpoint either by visiting Private Link Center or by performing the following steps:
 1. Navigate to your Azure SQL Managed Instance in Azure portal.
 2. In the sidebar, select Private endpoint connections.
 3. Locate the private endpoint connection in the table and select its name.
 4. In Overview, select the network interface.


5. In Overview, Private IP address is shown in the Essentials section.
2. [Create a private Azure DNS zone](#)  named *anything other than* `privatelink.<dns-zone>.database.windows.net`; for example: `privatelink.site`. Do not name the private DNS zone `privatelink.<dns-zone>.database.windows.net` because this will disrupt the instance's internal connectivity and cause management operations to fail.
3. [Link the private DNS zone to the endpoint virtual network](#) .
4. In the DNS zone, create a new record set with the following values:
 - o Name: `<instance-name>`
 - o Type: A
 - o IP address: IP address of the private endpoint obtained in step 1.

Troubleshooting connectivity issues

Test connectivity using Azure SQL Connectivity Checker

The Azure SQL Connectivity Checker tool is a PowerShell script, run from the client machine, which automates a series of checks for the most common configuration problems. Most issues it detects will come with recommendations for how to resolve them.

Run the following PowerShell script from the Windows client computers where the error is occurring.

- To run the tests from Linux, from machines without internet access, or from a containerized environment, see [SQL Connectivity Checker on GitHub](#) .
1. Open Windows PowerShell ISE (in **Administrator mode** if possible).
Note: To collect a network trace along with the tests (`CollectNetworkTrace` parameter), you must run PowerShell as an administrator.
 2. Open a **New Script** window.
 3. Paste the following in the script window:

```

$parameters = @{
    # Supports Single, Elastic Pools and Managed Instance (provide FQDN, MI public endpoint is supported)
    # Supports Azure Synapse / Azure SQL Data Warehouse (*.sql.azuresynapse.net / *.database.windows.net)
    # Supports Public Cloud (*.database.windows.net), Azure China (*.database.chinacloudapi.cn), Azure Ge
    Server = '.database.windows.net' # or any other supported FQDN
    Database = '' # Set the name of the database you wish to test, 'master' will be used by default if n
    User = '' # Set the login username you wish to use, 'AzSQLConnCheckerUser' will be used by default i
    Password = '' # Set the login password you wish to use, 'AzSQLConnCheckerPassword' will be used by d



    ## Optional parameters (default values will be used if omitted)
    SendAnonymousUsageData = $true # Set as $true (default) or $false
    RunAdvancedConnectivityPolicyTests = $true # Set as $true (default) or $false, this will load the li
    ConnectionAttempts = 1 # Number of connection attempts while running advanced connectivity tests
    DelayBetweenConnections = 1 # Number of seconds to wait between connection attempts while running adv
    CollectNetworkTrace = $true # Set as $true (default) or $false
    #EncryptionProtocol = '' # Supported values: 'Tls 1.0', 'Tls 1.1', 'Tls 1.2'; Without this parameter
}

$ProgressPreference = "SilentlyContinue";
if ("AzureKudu" -eq $env:DOTNET_CLI_TELEMETRY_PROFILE) {
    $scriptFile = '/ReducedSQLConnectivityChecker.ps1'
} else {
    $scriptFile = '/AzureSQLConnectivityChecker.ps1'
}
$scriptUrlBase = 'http://raw.githubusercontent.com/Azure/SQL-Connectivity-Checker/master'
cls
Write-Host 'Trying to download the script file from GitHub (https://github.com/Azure/SQL-Connectivity-Che
try {
    [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12 -bor [Net.SecurityPro
    Invoke-Command -ScriptBlock ([Scriptblock]::Create((Invoke-WebRequest ($scriptUrlBase + $scriptFile)
    })
} catch {
    Write-Host 'ERROR: The script file could not be downloaded:' -ForegroundColor Red
    $_.Exception
    Write-Host 'Confirm this machine can access https://github.com/Azure/SQL-Connectivity-Checker/' -Fore
    Write-Host 'or use a machine with Internet access to see how to run this from machines without Intern
}
#end

```

4. Set the parameters on the script. You must set the server name and database name. Setting the user and password is best practice but optional.
5. Run the script. The results are displayed in the output window. If the user has permissions to create folders, a folder with the resulting log file is created along with a ZIP file (AllFiles.zip). When running on Windows, the folder opens automatically after the script completes.
6. Examine the output for any issues detected and any recommended steps to resolve the issue.
7. If the issue can't be resolved, send AllFiles.zip using the **File upload** option in the **Details** step of creating your support case.

Resources

- [Connectivity architecture for Azure SQL Managed Instance](#) 
- [Connect your application to Azure SQL Managed Instance](#) 
- [Configure connection types](#) 