Network settings

Last updated by | Vitor Tomaz | Nov 24, 2021 at 2:56 AM PST

Contents

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 configure public endpoint for data access from outside the virtual network, the minimum TLS version, select between proxy or redirect connection type and other networking settings on your instance.

Enable connectivity via public endpoint (public IP)

Public endpoint for a managed instance enables data access to your managed instance from outside the virtual network. You are able to access your managed instance from multi-tenant Azure services like Power BI, Azure Data Factory, Azure DevOps, Azure App Service, or an on-premises network. By using the public endpoint on a managed instance, you do not need to use a VPN.

See how to enable public endpoint and allow public endpoint traffic on the network security group at Configure public endpoint in Azure SQL Managed Instance.

Configure minimal TLS Version

The minimal TLS version enforced by the managed instance for inbound connections. See configure minimal TLS version in Azure SQL Managed Instance for commands to set the TLS version, or if you have further questions or concerns.

Set connection type (proxy or redirect)

Azure SQL Managed Instance supports the following two connection types:

- Redirect (recommended): Clients establish connections directly to the node hosting the database. To enable connectivity using redirect, you must open firewalls and Network Security Groups (NSG) to allow access on ports 1433, and 11000-11999. Packets go directly to the database, and hence there are latency and throughput performance improvements using redirect over proxy.

- Proxy (default): In this mode, all connections are using a proxy gateway component. To enable connectivity, only port 1433 for private networks and port 3342 for public connection need to be opened. Choosing this mode can result in higher latency and lower throughput, depending on the nature of the workload. We highly recommend the redirect connection policy over the proxy connection policy for the lowest latency and highest throughput.

The redirect connection type currently works only for a private endpoint. Regardless of the connection type setting, connections coming through the public endpoint would be through a proxy.

See more at connection type.

Learn more about TCP ports

Port 1433 for private networks and port 3342 for public connection are the only ports that must be open on the computer that hosts the client application to SQL Managed Instance when you connect using **Proxy** connection type.

However, to make connections using **Redirect** connection type, you must additionally open firewalls and Network Security Groups (NSG) to allow access on ports 11000-11999.

Learn more about firewalls

Azure SQL Managed Instance does not have IP-based firewall like Azure SQL Database. The access to SQL Managed Instance is controlled via Network Security Groups (NSG). See allow public endpoint traffic on the network security group for an example on how to add rules to an NSG.

Configure linked server

If you are experiencing issues with querying remote data using linked servers:

- Check the linked server constraints in Managed Instances to identify if you are using some unsupported feature (for example, connection to SSAS, Oracle, MySQL, and other unsupported data sources).

- Check if the **remote access** and **show advanced options** options are enabled in sys.configurations view.

- Check that you have correctly created the linked server with the correct remote server name/IP address, port, and account information (username and password).

- Make sure that you are using SQL Server Authentication, because Linked server in Managed Instance can't use Windows or Azure Active Directory authentication.

- If you are getting the error message "Server is not found or not accessible", check whether you can reach the remote server from Managed Instance:

  - Create a SQL Agent job with one PowerShell task that executes a command like `tnc <remote-server> -1433`. Run the job and check the job output in the job history. Confirm the DNS resolution is correct, and that port is reachable.
  - Check have you enabled the port that is used to communicate with the remote server. Port should be added in the Outbound security rules of the Network Security Group that controls the access to your Managed Instance. You also need to have matching inbound rules in target network.
- Check that you are using Distributed Transaction, because MS DTC is not supported in Managed Instance
- Script the linked server that you are using on Managed Instance, setup the identical linked server on SQL Server, then try to run the query there. If possible, try to place SQL Server in Azure Virtual machine in the same VNet where your Managed Instance is placed (in the different subnet) to ensure that you have similar networking environment.

Connect from Power BI

Public endpoint for a managed instance enables data access to your managed instance from outside the virtual network from multi-tenant services like Power BI. Also, you can use Azure service tags with Power BI to enable an Azure SQL Managed Instance (MI) to allow incoming connections from the Power BI service. In Azure, a service tag is a defined group of IP addresses that you can configure to be automatically managed, as a group, to minimize the complexity of updates or changes to network security rules. By using service tags with Power BI, you can enable a SQL Managed Instance to allow incoming connections from the Power BI service. See Using service tags with Power BI.

Another option is to install the on premise data gateway on a virtual machine that has virtual network connectivity to the managed instance.

Connect from Azure Data Factory

To access the SQL Managed Instance public endpoint, you can use an Azure Data Factory managed Azure integration runtime. Make sure that you enable the public endpoint and also allow public endpoint traffic on the network security group so that Azure Data Factory can connect to your database.

To access the SQL Managed Instance private endpoint, set up a self-hosted integration runtime that can access the database. If you provision the self-hosted integration runtime in the same virtual network as your managed instance, make sure that your integration runtime machine is in a different subnet than your managed instance. If you provision your self-hosted integration runtime in a different virtual network than your managed instance, you can use either a virtual network peering or a virtual network to virtual network connection. See Connect your application to SQL Managed Instance.

Connect from Azure App Services

In order to connect an application that's hosted by Azure App Service to managed instance through it's private IP address, you first need to make a connection between the application and the SQL Managed Instance virtual network. See Integrate your app with an Azure virtual network.

For troubleshooting, see Troubleshooting virtual networks and applications. If a connection cannot be established, try syncing the networking configuration.

A special case of connecting Azure App Service to SQL Managed Instance is when you integrate Azure App Service to a network peered to a SQL Managed Instance virtual network (they don't belong to the same VNet). That case requires the following configuration to be set up:

- SQL Managed Instance virtual network must not have a gateway.
- SQL Managed Instance virtual network must have the `Use remote gateways` option set.
- Peered virtual network must have the `Allow gateway transit` option set.

For data access to your managed instance from outside the virtual network, see Configure public endpoint in Azure SQL Managed Instance

Connect to an SMTP for Database Mail

Try to reach the mail server from a Managed Instance using an SQL Agent Job that tests the network connection:

- Make sure that you enabled the port that communicates with the email server. Add the Port in the Outbound security rules of the Network Security Group that controls access to your Managed Instance.

- Create a SQL Agent job that has one PowerShell task that runs a command like `tnc <your_email_server> -Port <your_email_server_port>`. Run the job and check the job output in the job history. Confirm the DNS resolution is correct, and that port is reachable.

Resources and tools

Resources

- Connectivity architecture for Azure SQL Managed Instance
- Connect your application to Azure SQL Managed Instance
- Configure connection types

Test connectivity using Azure SQL Connectivity Checker

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

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. User and password are optional, but best practices.

5. Run the script. Results are displayed in the output window. If the user has permissions to create folders, a folder with the resulting log file 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 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.

How good have you found this content?