VNET - SQL Login Workflows

Last updated by | Vitor Tomaz | Aug 5, 2020 at 12:40 PM PDT

Contents

- Background

SQL DB currently provides capability to filter incoming connections for SQL DB tenant on either server or database level (so called SQL DB firewall, described at https://azure.microsoft.com/enus/documentation/articles/sql-database-firewall-configure/).

It has proven insufficient for large, security-conscious enterprise customers which are concerned about recent HomeDepot/Target attacks and are required to minimize their security exposure. The current top customer ask, with 875 votes and 316 developers tracking, is to support virtual networks to provide the following missing capabilities:

- Remove the need for the customer hosted service(s) to have public IP addresses, instead be connectable only from ExpressRoute private IP space
- Lockdown all outgoing Internet access and prevent the services in VNET from connecting to SQL DBs of other tenants this is to prevent data exfiltration in the event of hosted services compromise

The objective of this feature is to allow customers to isolate their SQL Database and clients on a "virtual network" that protects from external intrusion while not needing to setup firewall rules for each client.

Background

SQL DB regional deployment is divided into 1 (2 in the near future) control rings and N (currently in 10s and projected to grow) tenant rings. Each ring is a Windows Azure hosted service with a publicly facing VIP.

Control ring serves provisioning and connection entry functions behind a load balanced input endpoint. Control ring is typically small (about 40 VMs) hosted service.

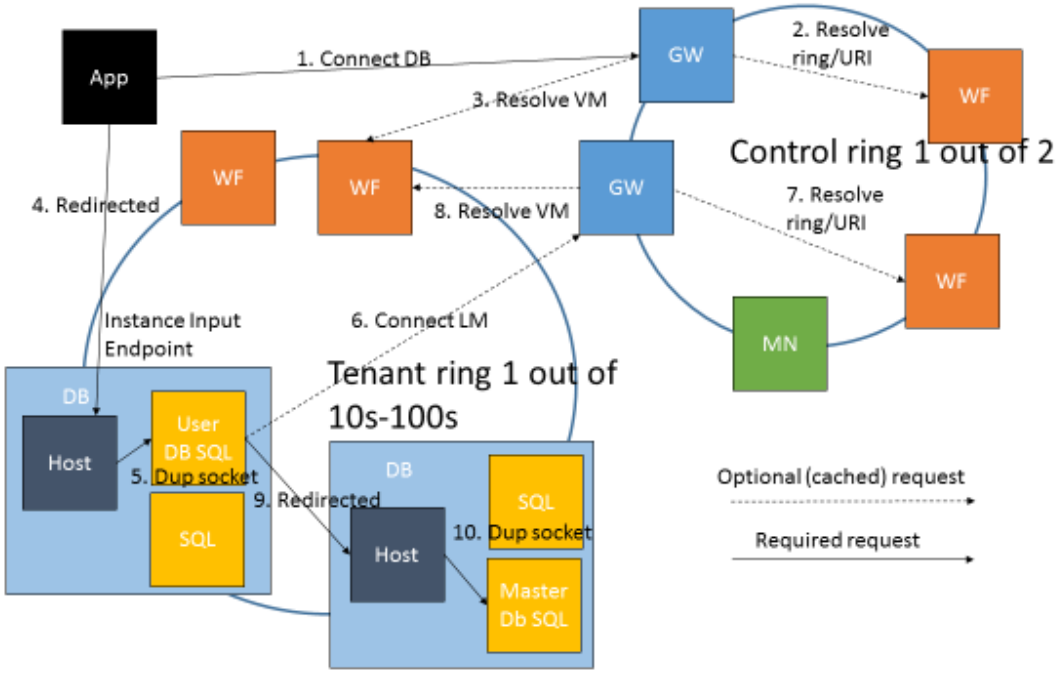
Tenant rings are where the customer workload actually runs and are sized to about 200-300 full VMs behind an input instance endpoint. There are multiple tenants inside a single tenant ring VM.

A minimum regional provisioning unit of SQL DB is so called “logical server” which gets a globally unique DNS name in the form of “myserver.database.windows.net ”. This DNS name ultimately resolves to the VIPs of the SQL DB control ring in the region where the logical server has been provisioned.

A logical server contains one or more databases of different editions and performance objectives. Those databases will likely be placed into different tenant rings (depending on hardware requirements for the specific database edition, available capacity, etc.) inside SQL instances. SQL DB never shares SQL instances cross tenant (logical server) boundary. Each SQL instance gets a unique DNS name that

in the form of <instancename>.trN.<region>.worker.database.windows.net mapped to the VIP of the tenant ring the SQL instance has been placed to.

The diagram below illustrates a typical connection flow.



1. Application initiates a connection to "*myserver.database.windows.net*", port 1433. The IP gets resolved to the VIP of the control ring(s) and a connection established to one of the SQL DB gateway

machines which acts as either a redirector or a proxy. An SSL session is established and the application sends myserver.database.windows.net as well as the target database name in TDS protocol

login packet (this is described in depth at https://msdn.microsoft.com/en-us/library/dd304019.aspx).

2. SQL DB gateway resolves the tenant ring name and the internal database identifier using the server and database names provided in TDS login packet. This resolution is based on cached metadata

available in the redirector.

3. SQL DB gateway connects to the target tenant ring and resolves the internal database identifier to the input instance endpoint port uniquely identifying the VM running the primary replica of the

database. Once the resolution is complete, the redirector sends back a TDS redirect message containing the instance name provisioned for that database and the port of the VM:

<instancename>.trN.<region>.worker.database.windows.net: <VM port>

4. Application connects using the provided DNS name and port and does SSL exchange again followed by TDS login message where it provides the target instance name *<instancename>.trN.<region>.worker.database.windows.net* and the target database name.

5. The application connection is accepted by a VM level service called 'socket duplicator' which then can identify the target SQL instance based on the information provided in the login packet.

The socket duplicator then performs socket and SSL shared session key duplication into the target SQL instance

6. At this point the login credential and firewall rule validation is done by the target SQL instance. Depending on authentication scheme used, this SQL instance may establish a connection to so called 'master' database to authenticate the customer and validate the firewall rules, this is done in Steps 6 through 10.

This connection architecture in SQL DB is used to minimize the latency of the application requests to SQL DB once fully connected by removing the frontend and socket duplicator process from the packet path once fully connected to allow easier on-boarding of existing latency sensitive SQL applications. The current observed IaaS VM to SQL DB "SELECT 1" roundtrip latency is about 0.8ms.

Currently SQL DB can only redirect connections for SQL client versions from ADO.NET 4.5 or higher. Older clients get proxied from the redirector.

The current SQL DB policy is to only redirect connections originating from Windows Azure IP space as:

7. For on-premise scenarios we have seen multiple cases where the increased IP range and port unpredictability caused by redirection broke applications after migration.
8. Reduced latency is not as relevant when connecting from outside of Windows Azure

It is important to point out that for some security conscious customers the default redirect policy has caused problems because the lock down outgoing IPs from inside their VNETs and it is more

difficult to manage the less predictable IP and port ranges – we would like the VNET solution described here remove the fragile configuration required.

Current SQL DB Login Path

In SQL DB Login Path, the existing IPv4 firewall rule check happens as part of the Authentication step. Authentication is driven by SQL User DB and can happen on two distinct places:

In Logical Master: If the XODBC Authentication Cache (which includes a cached copy of the Server-level IPv4 firewall rules) is not available when the login lands on SQL User DB, then SQL User DB executes a spec proc against Logical Master to validate the current login using the username, password and the source IP. Logical Master checks the source IP is allowed by the current sys.firewall_rules table.

In User databases: If the XODBC Authentication Cache is available, then SQL User DB checks whether the source IP is allowed by the current in-memory copy of sys.firewall_rules without having to execute a spec proc against master.

Connectivity changes

The design of VNET Service Tunneling will closely follow the current IP-based firewall rules semantics. This is the overview:

- VNET Firewall rules will have Allow-list semantics.
- The Allow/Deny enforcement checks will be done on SQL Server.

- This can be pushed up the connectivity stack (xdbhost/xdbgateway) on further iterations to protect the engine from malicious load.
- VNET Firewall Rules rules can be created/deleted only from ARM APIs/Portal.
- Create/Delete rules through T-SQL must be disabled to ensure Separation of Roles
- Customers can create Server-level firewall rules. Database-level firewall rules will not be enabled on this iteration. Server-level rules will be stored on Logical Master.
- If the server has at least one VNET Firewall Rule in the list, SQL DB will only honor VNET rules and not the existing IPv4 firewall rules.
- This needs to be implemented carefully considering the full spectrum [VNETserver] x [VNETdatabase, IPv4database] x [Login Cache].
- Logins originated from VNETs with Service Tunneling enabled will be blocked against database targets that have not explicitly allowed the source VNET on the firewall rules. If a login is originated from a VNET with Service Tunneling, only an explicit VNET firewall rule can allow it.
- VNET are integrated with current Login Cache functionality to avoid additional roundtrips to Logical Master.

How good have you found this content?