Create secondary server via T-SQL (cross-subscription(slash)cross-tenant)

Last updated by | Vitor Tomaz | Feb 18, 2021 at 2:30 AM PST

Contents

Issue

What we found is to establish GeoDR via T-SQL, the IP address of T-SQL client running the command must be allowed on the partner server. So we need to make sure your IP is allowed in the secondary server's firewall rule.

That said, I was able to connect to both servers in my previous test but I still had to open all the IP ranges in order to make this work. This was because two servers recognize different IP addresses probably due to different NAT when I try to connect although they are connecting from the same client machine. Let me illustrate this further. When I connect to my primary server, the IP used to connect and allowlisted is 108.235.201.152. But when I connect to my secondary server, the IP used to connect and allowed is 167.220.148.92. In this case, I have to also allow list 108.235.201.152 in my secondary server so that I can create the Geo secondary successfully. Otherwise I get insufficient permission to add secondary on server. Actually I don't have to allow list 167.220.148.92 if I only want to create the secondary database and don't need to access the secondary.

Thus, whatever IP address you used to connect to your primary server needs to be allow listed in the secondary server, regardless you are able to connect to secondary server or not. And don't forget to hit SAVE button when you update your firewall rule 😊.

This requirement applies when you create geo replication in the same subscription and across subscription. And good news is across subscription is supported when creating secondary via TSQL. However we will document this requirement better. Sorry for the inconvenience.

Following TSQL works for servers in the same subscription or across subscriptions

Highlighted in green is the permission requirement

--Run against master db in primary server

create login drtest with password = 'Some Password'

create user drtest from login drtest

ALTER ROLE dbmanager ADD MEMBER drtest;

--Run against user db in primary server

CREATE user drtest from login drtest

ALTER ROLE db_owner ADD MEMBER drtest;

select * from sys.database_principals

--get the sid

--Run against master db in secondary server called drtestcc

create login drtest with password = 'Some Password', SID = 0x0106000000000064000000000000000080E4E2B7AE1B4B46873F917B96645E03

create user drtest from login drtest

ALTER ROLE dbmanager ADD MEMBER drtest;

--Login using drtest account and run following against master db in primary server

ALTER DATABASE testdr ADD SECONDARY ON SERVER drtestcc WITH

(ALLOW_CONNECTIONS = ALL);

Without allow listing the IP, the error is:

Msg 45189, Level 16, State 1, Line 24 Insufficient permission to add secondary on server 'drtestcc'.

After allow listing the IP as talked above, it works fine

--TSQL command to stop the replication

ALTER DATABASE testdr

REMOVE SECONDARY ON SERVER secondtestwest

Workaround

To setup Active Geo-Replication between two databases belonging to different subscriptions (whether under the same tenant or not) you must follow a special procedure, described below.

The procedure is based on SQL commands and requires to:

- Create a privileged login on both servers
- allow list the IP address of the client performing the change on both servers (e.g. the IP address of the host running SQL Server Management Studio)

The client performing the changes needs network access to the primary server.

Although the same IP address of the client must be allow listed on the secondary server, network connectivity to the secondary server is not strictly required.

Steps 1 to 9 must be executed using the Azure SQL Database server admin user.

On the master of the primary server

1. Allow list the IP address of the client performing the changes (for details on how to add firewall rules, see https://docs.microsoft.com/en-us/azure/sql-database/sql-database-firewall-configure)
2. Create a login dedicated to setup Active Geo-Replication (adjust credentials as needed): create login geodrsetup with password='SomePassword01';
3. Create a corresponding user and assign it to the dbmanager role:

create user geodrsetup for login geodrsetup;

alter role dbmanager add member geodrsetup;

4. Take note of the SID of the new login using this query: select sid from sys.sql_logins where name = 'geodrsetup'

On the source database on the primary server

1. Create a user for the same login: create user geodrsetup for login geodrsetup;
2. Add the user to the db_owner role: alter role db_owner add member geodrsetup;

On the master of the secondary server

1. Allow list the IP address of the client performing the changes. It must be the same exact IP address of the primary server.
2. Create the same login of the primary server, use the same username and password of step 2, and use the SID you got in step 4: create login geodrsetup with password='SomePassword01', sid=0x01060000000000640000000000000000001C98F52B95D9C84BBBA8578FACE37C3E;
3. Create a corresponding user and assign it to the dbmanager role:

create user geodrsetup for login geodrsetup;

alter role dbmanager add member geodrsetup;

On the master of the primary server

1. Login to the master of the primary server using the new login.
2. Create a secondary replica of the source database on the secondary server (adjust database name and server name as needed): alter database dbrep add secondary on server mtlvwesql1;

After the initial setup, the users, logins and firewall rules creates as part of steps 1, 2, 3, 5, 7, 8 and 9 can be removed.

Classification

Root Cause: Azure SQL DB v2\GeoDR/AutoDR

How good have you found this content?