

How to check connectivity errors

Last updated by | Daniel Valero | Mar 23, 2023 at 9:20 AM PDT

Contents

- [Basic checks for connectivity cases](#)
- [Errors connecting due to missing firewall rules \(Server with...](#)
- [Troubleshooting other connection scenarios](#)

Basic checks for connectivity cases

IMPORTANT: Customer should always connect using the server FQDN. It is never recommended to connect using IP Address or the Private DNS Zone record A (Canonical name) for the server.

1. Always check if the server name can be resolved using

```
nslookup <server_FQDN>
```

If *nslookup* does not return a value, there might be an issue with name resolution (DNS) and the solution can be simple or very complex. Engage Networking team if necessary.

- For Public access it should look like

```
daniel@UbuntuServer2204A:~$ nslookup pgsqldvvr6.postgres.database.azure.com
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
Name:   pgsqldvvr6.postgres.database.azure.com
Address: 20.120.52.176
```

- For Private access it should look like (indicates the private DNS zone used)

```
daniel@UbuntuServer2204A:~$ nslookup pgdvvr8.postgres.database.azure.com
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
pgdvvr8.postgres.database.azure.com    canonical name = db1f8b98ef78.pgdvvr8.private.postgres.database.azure.com.
Name:   db1f8b98ef78.pgdvvr8.private.postgres.database.azure.com
Address: 10.5.3.4
```

You can verify the Private DNS Zone and private IP address used by the server by following instructions at [Get VNet/subnet, Private DNS Zone and private IP Address for a server set for Private Access \(VNET Integration\)](#).

If the Private DNS zone or IP Address returned by *nslookup* is different from the one you see in XTS following the document mentioned before, there could be a DNS configuration.

If the connection to the database server is being done from another Azure resources deployed on a VNET (such as VM or AKS) and name resolution is not working, make sure there is a Virtual Network Link between the source VNET and Private DNS zone used by the database server as explained at [Link the virtual network](#). You can check the Virtual Network Links for a Private DNS Zone in ASC

Subscriptions

Search for resources

Quick access

No items in quick access yet.

All resources

Resource Providers

- Microsoft.Compute
- Microsoft.DataFactory
- Microsoft.DBforMySQL
- Microsoft.DBforPostgreSQL
- Microsoft.DevTestLab
- Microsoft.Insights
- Microsoft.KeyVault
- Microsoft.Network
- Microsoft.Storage
- Microsoft.Web

networkInterfaces

networkSecurityGroups

networkWatchers

privateDnsZones

pgdvr8.private.postgres.database.azure.com

privateEndpoints

publicIPAddresses

routeTables

virtualNetworks

Max Number of Virtual Network Links: 1000

Number of Virtual Network Links: 3

Max Number of Virtual Network Links with Auto-Registration Enabled: 100

Number of Virtual Network Links with Auto-Registration Enabled: 0

Virtual Network Links: [Virtual Network Links \(download JSON\)](#)

Control Plane Link: [Link](#)

Virtual Network Links

Drag a column header and drop it here to group by that column

Name	Virtual Network	ProvisioningState	VirtualNetworkLinkState	RegistrationEnabled	Id
dnsvnetb	VNETB	Succeeded	Completed	False	
sdadas	Win10_group-vnet	Succeeded	Completed	False	
vbcwb7pevg23g	VNETA	Succeeded	Completed	False	

1 - 3 of 3 items

2. If name resolution is correct, check if the connection is possible using another client

For example: if the issue happens from a web app or from PgAdmin, try using *psql* cli.

If you can connect from the same source using another client, the issue might be on the app that fails.

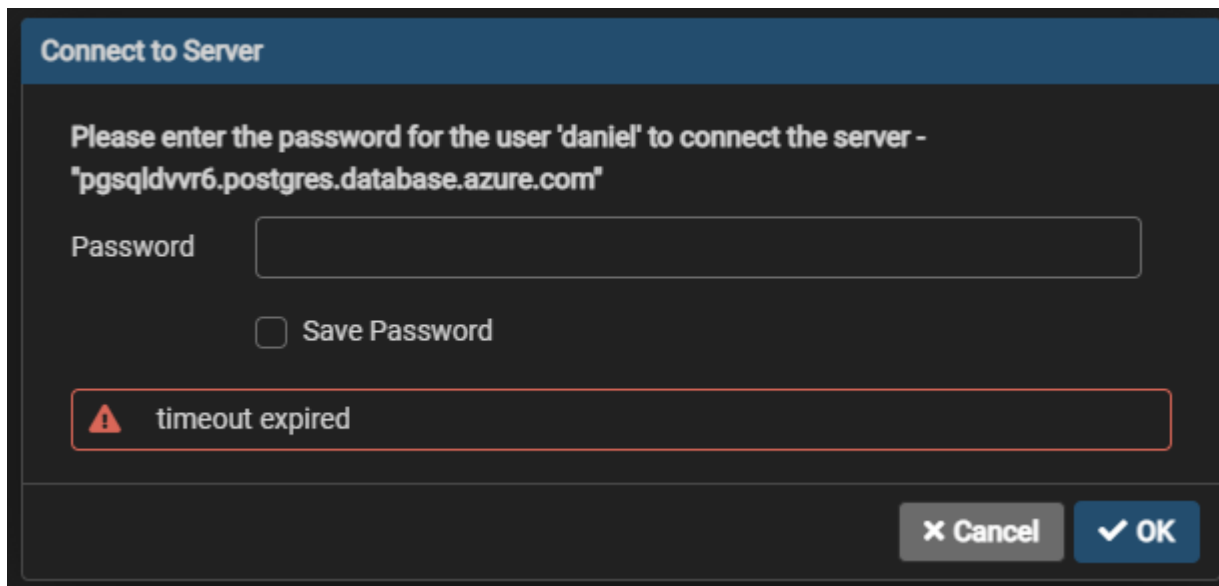
3. If name resolution is correct and it is not possible to connect from any app/client in the source, check if the connection is possible from another source in the same VNET

For example: if not possible to connect from any app/client in VM1, try to connect from VM2. If you can connect from VM2 in the same VNET as VM1, the issue is on VM1, not the source VNET

Errors connecting due to missing firewall rules (Server with Public Access)

If using Public Endpoint and no firewall rule set for the origin IP, the error you can get can be different depending on the client tool, some examples below:

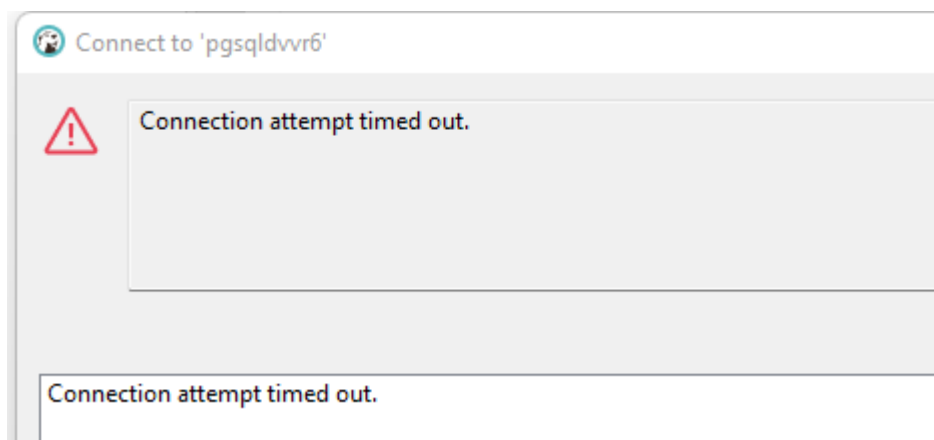
- pgAdmin shows "timeout expired"



- psql shows "psql: error: could not connect to server: could not connect to server: Connection timed out"

```
C:\Users\danielva>psql -h pgsqldvvr6.postgres.database.azure.com -d postgres -U daniel
psql: error: could not connect to server: could not connect to server: Connection timed out (0x0000274C/10060)
Is the server running on host "pgsqldvvr6.postgres.database.azure.com" (20.120.52.176) and accepting
TCP/IP connections on port 5432?
```

- Dbveaver shows "Connection attempt timed out"



IMPORTANT: Always check if the correct firewall rules are in place, as the error message is not clear to indicate there is a missing Firewall rule, as no view in Kusto will have no entry for that failed connection as never made it to the PostgreSQL server.

To get Firewall rules active in the server, refer to [Get firewall rules for servers that allow Public Access](#)

Troubleshooting other connection scenarios

1 Check if server is healthy using the Kusto query or if server is older than 8 days then via XTS

```
MonOBDockerContainerEvents
| where LogicalServerName == "YOUR-SERVER-NAME"
| where Event contains "health"
```

2. Check if any connections are reaching the server successfully using this kusto query


```
MonOBPgLogs
| where LogicalServerName contains "YOUR-SERVER-NAME"
| where TIMESTAMP > ago(1d)
| where message contains "connection received"
| where message !contains "127.0.0"
| project TIMESTAMP, message
```

if the above query result is empty, use the following. Unfortunately, the following query will also return localhost connections which is filtered out in the above query. Therefore, the result of the following may not be helpful as much;

```
MonPgLogs
| where LogicalServerName contains "YOUR-SERVER-NAME"
| where TIMESTAMP > ago(1d)
| where message_id contains "connection received"
| project TIMESTAMP, message_id
```


3. Check if there were any errors while logging in to PG:

```
MonOBPgLogin
| where LogicalServerName contains "YOUR SERVER NAME"
| where TIMESTAMP > ago(1d) | where peer_address != "127.0.0.x" | where is_success == "false"
| project TIMESTAMP, is_success, error, state, LogicalServerName
```

If this query returns non-zero rows, check the error code and state for the errors. You can find the exact error [here](#) 

If it returns zero rows, try with the following query;

```
MonPgLogs
| where LogicalServerName contains "YOUR SERVER NAME"
| where TIMESTAMP > ago(1d)
| where (sqlerrcode == '53300' and (funcname in ('CheckMyDatabase', 'InitPostgres', 'InitializeSessionUserId',
  (sqlerrcode in ('28000', '28P01') and funcname == 'auth_failed') or
  (sqlerrcode == '28000' and funcname == 'ClientAuthentication') or
  (sqlerrcode == '3D000' and funcname == 'InitPostgres') or
  ((sqlerrcode == '53200' or errorLevel == 'LOG') and funcname == 'BackendStartup') or
  (sqlerrcode == '57P03' and funcname == 'ProcessStartupPacket') or
  (sqlerrcode == '42501' and funcname == 'InitPostgres'))
| project TIMESTAMP, message_id, saved_errno, errorLevel, sqlerrcode, funcname
```

You can look the sqlerror code in PostgreSQL Error Code list at <https://www.postgresql.org/docs/11/errcodes-appendix.html> 

Some examples:

- wrong username/password

TIMESTAMP	message_id	saved_errno	errorLevel	sqlerrcode	funcname
2022-01-14 20:03:00.0000000	password authentication failed for user \"%s\"	0	FATAL	28P01	auth_failed
2022-01-14 20:03:00.0000000	no pg_hba.conf entry for host \"%s\", user \"%s\", database \"%s\", %s	11	FATAL	28000	ClientAuthentication

You can see the message:

```
password authentication failed for user "%s"
```

Keep in mind that, for a wrong username/password error, there will be an entry with message_id

```
no pg_hba.conf entry for host "%s", user "%s", database "%s", %s
```

It does NOT mean there is a missing firewall rule missing. Both message appear together. It is just how the PostgreSQL drivers manage the event. Below the image of a failed connection due to wrong username/password in a server where a firewall rule as exists to allow connections from 189.212.204.63

```
C:\Users\danielva>psql -h pgsqldevr6.postgres.database.azure.com -d postgres -U daniel
Password for user daniel:
psql: error: could not connect to server: FATAL: password authentication failed for user "daniel"
FATAL: no pg_hba.conf entry for host "189.212.204.63", user "daniel", database "postgres", SSL off
```

- non-SSL connection (using sslmode=disable) that failed as the server requires SSL

In the client app, you will get an error like

```
psql: error: could not connect to server: FATAL: no pg_hba.conf entry for host "189.212.204.63", user
"daniel", database "postgres", SSL off
```

In MonPgLogs, the message does not indicate it is due to a SSL issue

TIMESTAMP	message_id	saved_errno	errorLevel	sqlerrcode	funcname
2022-01-14 20:48:20.0000000	no pg_hba.conf entry for host \"%s\", user \"%s\", database \"%s\", %s	11	FATAL	28000	ClientAuthentication

- Connection limit reached (sorry, too many clients already)

TIMESTAMP	message_id	saved_errno	errorLevel	sqlerrcode	funcname
2022-01-14 21:26:20.0000000	sorry, too many clients already	0	FATAL	53300	InitProcess
2022-01-14 21:26:20.0000000	remaining connection slots are reserved for non-replication superuser connections	0	FATAL	53300	InitPostgres

For more detailed information on how to troubleshoot this scenario go to [ERROR: Too many connections or hit maximum connection limit](#)

- Connection to non existing database

TIMESTAMP	message_id	saved_errno	errorLevel	sqlerrcode	funcname
2022-01-14 21:56:20.0000000	database \"%s\" does not exist	0	FATAL	3D000	InitPostgres