Connection forcibly closed by the remote host

Last updated by | Daniel Valero | Jan 24, 2022 at 10:34 AM PST

When you see "*An existing connection was forcibly closed by the remote host*", that indicates your client closed the connection to the Postgres server

Read Troubleshoot PostgreSQL: 'An existing connection was forcibly closed by the remote host'

Other things you can check:

- Check if customer is using a connection pooler and if the pooler was restarted.

  If customer is using a connection pooler, and the pooler restarts, the connections to the database server are closed.

  For example, if PgBouncer is restarted or stooped, in its log you can see something like:

  > 2022-01-21 16:42:51.082 UTC [15590] LOG got SIGTERM, fast exit

  In MonLogin, you will see the error "An existing connection was forcibly closed by the remote host"

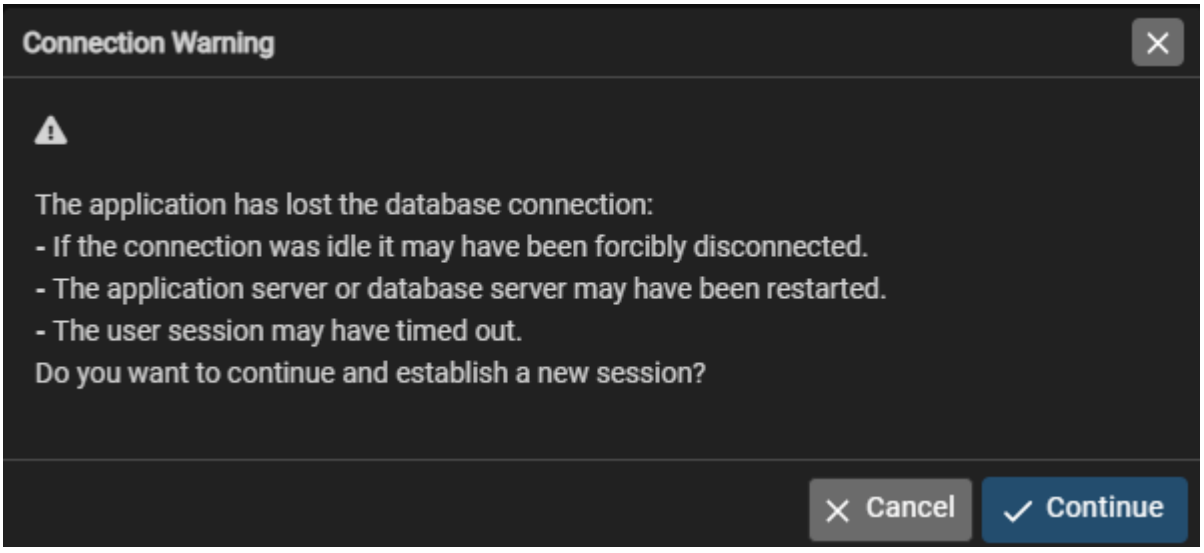
```
let TimeCheckStart = datetime('01/21/2022 16:40:00');
let TimeCheckEnd = datetime('01/21/2022 17:30:00');
MonRdmsPgSqlSandbox
| where TIMESTAMP >= TimeCheckStart and TIMESTAMP < TimeCheckEnd
| where LogicalServerName == "<server_name>"
| where text contains "An existing connection was forcibly closed by the remote host"
   or text contains " unexpected EOF on client connection with an open transaction"
| project-reorder originalEventTimestamp, text
| order by originalEventTimestamp asc
```

originalEventTimestamp	text
2022-01-21 16:40:32.5530236	2022-01-21 16:40:32 UTC-61eae1f3.ee8-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:41:41.5398001	2022-01-21 16:41:41 UTC-61eae21a.fe0-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:41:41.5398391	2022-01-21 16:41:41 UTC-61eae21b.10c8-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:41:41.5414250	2022-01-21 16:41:41 UTC-61eae21a.fe8-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1298870	2022-01-21 16:42:51 UTC-61eae252.14bc-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1309042	2022-01-21 16:42:51 UTC-61eae250.12c0-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1312267	2022-01-21 16:42:51 UTC-61eae253.14d4-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1318551	2022-01-21 16:42:51 UTC-61eae24f.11f8-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1323200	2022-01-21 16:42:51 UTC-61eae24d.10dc-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1328075	2022-01-21 16:42:51 UTC-61eae24d.1128-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1334242	2022-01-21 16:42:51 UTC-61eae24e.11a4-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1344574	2022-01-21 16:42:51 UTC-61eae251.13b4-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1361998	2022-01-21 16:42:51 UTC-61eae251.13c4-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 16:42:51.1371645	2022-01-21 16:42:51 UTC-61eae24c.10d4-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
[illegible]	2022-01-21 16:42:51 UTC-61eae24f.11ec-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.

In this situation "remote host" in the database log means the client. However, it can be misleading for the customer as the client application can show a messages saying the server closed the connections, but in this scenario, it means the pooler closed the connection, not Azure Database for PostgreSQL

```
postgres=*> start transaction;
server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
The connection to the server was lost. Attempting reset: Succeeded.
```

PgAdmin shows a different message in this situation



If there were open transaction at the time the pooler restarted, you could also see the message "unexpected EOF on client connection with an open transaction"

2022-01-21 17:05:41.2674996	2022-01-21 17:05:41 UTC-61eae7e4.1f50-LOG: could not receive data from client: An existing connection was forcibly closed by the remote hos
2022-01-21 17:05:41.2684265	2022-01-21 17:05:41 UTC-61eae7e4.1f50-LOG: unexpected EOF on client connection with an open transaction

- Check if customer is using a connection pooler and if the pooler is closing idle connections (below some example for PgBouncer)
  - PgBouncer drops server connections that have been idle more than **server_idle_timeout** seconds. By default this parameter is 600.

    When this happens, in the PgBouncer log you will see a message like

    > 2022-01-21 17:31:32.554 UTC [18689] LOG S-0x564c91f39060: postgres/daniel@pgdvvr1@40.71.8.203:5432 closing because: server idle timeout (age=649s)

    and MonLogin will show the error "An existing connection was forcibly closed by the remote host"

originalEventTimestamp	text
2022-01-21 17:31:32.5594795	2022-01-21 17:31:32 UTC-61eaeb6b.1fe0-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.

    in PgBouncer, you can increase the value for the parameter, or disable the server idle timeout by setting *server_idle_timeout* to 0

  - PgBouncer will close an unused server connection that has been connected longer than **server_lifetime** seconds. The default value is 3600 seconds (1hour). Setting it to 0 means the connection is to be used only once, then closed.

    When this happens, in the PgBouncer log you will see a message like

    > 2022-01-21 18:27:51.558 UTC [23250] LOG S-0x55d1a7c31a60: postgres/daniel@pgdvvr1@40.71.8.203:5432 closing because: server_lifetime (age=281s)

    and MonLogin will show the error "An existing connection was forcibly closed by the remote host"

originalEventTimestamp	text	TI
2022-01-21 18:27:51.5575837	2022-01-21 18:27:51 UTC-61eafa0e.2160-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.	20

  - PgBouncer will close any active connections that has been in “idle in transaction” state than **idle_transaction_timeout** seconds. By default, this timeout is disabled (0)

    When this happens, in the PgBouncer log you will see a message like

    > 2022-01-21 18:54:05.824 UTC [25204] LOG C-0x560a75af5850: postgres/daniel@pgdvvr1@189.212.204.63:3102 closing because: idle transaction timeout (age=105s)

    > 2022-01-21 18:54:05.824 UTC [25204] WARNING C-0x560a75af5850: postgres/daniel@pgdvvr1@189.212.204.63:3102 pooler error: idle transaction timeout

and MonLogin will show the error "An existing connection was forcibly closed by the remote host"

originalEventTimestamp	text
2022-01-21 18:54:05.8342776	2022-01-21 18:54:05 UTC-61eb00e4.28f0-LOG: could not receive data from client: An existing connection was forcibly closed by the remote host.
2022-01-21 18:54:05.8362191	2022-01-21 18:54:05 UTC-61eb00e4.28f0-LOG: unexpected EOF on client connection with an open transaction

In this scenario, a client app such as psql shows the message "server closed the connection unexpectedly". This can be misleading for the customer as the client application can show a messages saying the server closed the connections, but in this scenario, it means the pooler closed the connection, not Azure Database for PostgreSQL

```
postgres=> start transaction;
START TRANSACTION
postgres=*> select pg_backend_pid();
ERROR:  idle transaction timeout
server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
The connection to the server was lost. Attempting reset: Succeeded.
```

- PgBouncer disconnect client after **query_wait_timeout** seconds if the query is not assigned to a connections during that time. If all connections on the pool are in use, any new query must wait for a connection to be available, and it will wait **query_wait_timeout** seconds. If after **query_wait_timeout** seconds the query does not get a connection, the application can get an error like

```
postgres=> select pg_backend_pid();
ERROR:  query_wait_timeout
server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
The connection to the server was lost. Attempting reset: Succeeded.
```

When this happens, in the PgBouncer log you will see a message like

> 2022-01-21 19:08:34.768 UTC [26722] LOG C-0x55fd6907cc70:
> postgres/daniel@pgdvvr1@189.212.204.63:1664 closing because: query_wait_timeout (age=64s)

> 2022-01-21 19:08:34.768 UTC [26722] WARNING C-0x55fd6907cc70:
> postgres/daniel@pgdvvr1@189.212.204.63:1664 pooler error: query_wait_timeout

In this scenario, a client app shows the message "server closed the connection unexpectedly". This can be misleading for the customer as the client application can show a messages saying the server closed the connections, but in this scenario, it means the pooler closed the connection, not Azure Database for PostgreSWL. MonLogin will not have any entry about it as the issue from a disconnection from the app to PgBouncer and not database connection was closed on the database server.

Other actions you can take

An existing connection was forcibly closed by the remote host. This normally results if the peer application on the remote host is suddenly stopped, the host is rebooted, the host or remote network interface is disabled, or the

remote host uses a hard close (see setsockopt for more information on the SO_LINGER option on the remote socket). This error may also result if a connection was broken due to keep-alive activity detecting a failure while one or more operations are in progress. Operations that were in progress fail with WSAENETRESET. Subsequent operations fail with WSAECONNRESET.

Source: https://docs.microsoft.com/en-us/windows/win32/winsock/windows-sockets-error-codes-2

The description of WSAENETRESET, on the same page, indicates that the last two sentences apply only to the keep-alive failure case.)

- You are sending malformed data to the application (which could include sending an HTTPS request to an HTTP server)
- The network link between the client and server is going down for some reason
- Application has exhausted system resources

You can fire up Wireshark to see exactly what is happening on the wire to narrow down the problem.

Solution:

Pgbouncer; using a server level connection pooler like Pgbouncer which will uniform your connection activities.

OR

Send a dummy “select 1” signal to the backend database server to make sure those keep-alive signals are not dropped and can make their way to the backend server.

Other reasons for this issue:

- **Client machine is slow** – is there resource pressure (high CPU, high memory usage, high context switching) in the client side that might explain this slowness. We have seen this issue when the cloud service was overloaded.
  I would like to know if this issue occurs particularly from one client server or multiple machines.
- **Network path is slow** – this is complex piece to investigate. We will need to capture a network trace to investigate.

How to capture network trace:

1. Download Network Monitor - http://www.microsoft.com/en-us/download/details.aspx?id=4865
2. Install Network Monitor on the client experiencing the problem.
3. Using the NMCap command-line utility that comes with the Netmon 3.4 installation, start the chained trace (chained = create a new capture file after reaching the "Temporary capture file size" rather than overwriting the current capture)
   a. To do this, run the following commands in an elevated command prompt
      1. md C:\Netmon (This will be where the capture files will be stored. You can change it to another drive and folder if needed)
      2. cd C:\Program Files\Microsoft Network Monitor 3
      3. NMCap /network * /capture /file C:\Netmon\Netmon.chn:100M
      * The above command creates 100 MB chained files
      * You must use the .chn extension to capture chained traces.
      * The target folder must exist for NMCap to create the trace file.
4. Get the IP addresses of the client
   ipconfig /all on the client > client.txt
5. Wait until the issue reoccurs. please note the exact time the problem reproduces. Also, please note the exact error message.
6. Stop the trace by hitting Ctrl-c in the command-line window

7. Once you have the capture please ZIP the files into a single .zip file.

IMPORTANT: Please remember to let me know the IP addresses of the client, database name, exact error message as well as the time that the problem occurred.

Created with Microsoft OneNote 2016.

How good have you found this content?