MI database left in restoring state after failure

Last updated by | Balaji Barmavat | Nov 30, 2020 at 5:20 PM PST

Contents

Problem Description:

The customer is trying to do a SQL DMS Online migration of two databases from on-premise to a Managed Instance. One of the migration tasks is in the expected state, with current transaction logs being shipped to the target database. The other task however is continuously failing with the error: "The destination database name 'name_of_database' already exists on the server 'name_of_managed_instance'."

The customer checked the existing database on their Managed Instance and found two databases. Both had a GUID as name and not the expected names given by the customer:

SELECT name, create_date FROM sys.databases where database_id > 4

/*

e6452eb7-f691-4ff9-9214-08c833e4ff54	2018-11-19 09:59:06.177	
c731e714-6a83-437b-a89b-fedc6271d57e	2018-11-19 12:38:57.670	

*/

The customer reported that they had some instability in their Site-2-Site VPN during the initial DMS configuration attempts. One of the migration tasks had failed, and they reconfigured/restarted it but it failed with above error. The suspicion was that the old migration got stuck somehow.

They were unable to remove this {GUID] database themselves.

Troubleshooting:

CMS queries:

select * from managed_servers where name = 'managed_instance_name' or customer_subscription_id = 'Input SubscriptionId here'

select managed_server_id, managed_database_id, managed_database_name, sql_database_id, state, create_mode, create_time, last_update_time, dropped_time, first_backup_time, backup_retention_days, request_id

from managed_databases

where managed_server_id = 'fb4a5396-7a03-4292-801a-f3031c409c3d'

order by managed_database_name

managed_server_id	managed_database_id	managed_database_name	sql_database_id	
fb4a5396-7a03-4292-801a-f3031c409c3d	e6452eb7-f691-4ff9-9214-08c833e4ff54	database_name_1	8	WaitingForP
fb4a5396-7a03-4292-801a-f3031c409c3d	c731e714-6a83-437b-a89b-fedc6271d57e	database_name_2	10	WaitingForP

select * from managed_restore_requests where source_managed_server_id = 'fb4a5396-7a03-4292-801a-f3031c409c3d' or target_managed_server_id = 'fb4a5396-7a03-4292-801a-f3031c409c3d'

e6452eb7-f691-4ff9-9214-08c833e4ff54 Restore finished. The last restored backup is DB1_AutoCRM_Kijiji_Test_FULL_20181114_033040.bak.

c731e714-6a83-437b-a89b-fedc6271d57e Restore finished. The last restored backup is DB1_AutoImport_Kijiji_LOG_20181120_030000.trn.

The first entry is pointing to the failed configuration; the second entry is for the successful, active configuration for which we are restoring transaction logs (log shipping).

Cause:

This is expected state for the database to be in for an online migration.

If the migration task has failed, or if the database has been left behind after the migration task has been deleted, then the database will remain in the restoring state.

Solution / Mitigation:

The Managed Instance database can be dropped by the customer through either of the following methods.

Use the logical database name (the GUID) that is listed on sys.servers as the database to be deleted (do not use the name chosen by the user).

- ARM Delete Request

Examples:

DELETE https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/managedInstances/{managedInstanceName}/databases/{databaseName}?api-version=2017-03-01-preview

DELETE https://management.azure.com/subscriptions/00000000-1111-2222-3333-444444444444/resourceGroups/Default-SQL-SouthEastAsia/providers/Microsoft.Sql/managedInstances/managedInstance/databases/e6452eb7-f691-4ff9-9214-08c833e4ff54?api-version=2017-03-01-preview

- Powershell
- CLI

Example:

az sql midb delete -g mygroup --mi myinstance -n "e6452eb7-f691-4ff9-9214-08c833e4ff54" --yes

These mitigation steps will work for all failed restore workflows, where the database has been left behind in a restoring state.

How good have you found this content?