Backup & Recovery (Managed Instance)

Last updated by | Mohamed Baioumy | Dec 6, 2022 at 7:19 AM PST

Contents

Issue

The purpose of this document is to provide guidance on how to check, monitor and troubleshoot backup and recovery telemetry logs for SQL Managed Instance.

How to check, monitor and troubleshoot backup and recovery telemetry logs

Customer side

Customers can check their Azure SQL Managed Instance backup activity by querying the `msdb` database or by configuring an extended event (XEvent) session.

1. Querying the `msdb` database - To view backup activity, run this T-SQL query in the `msdb` database:

```
SELECT TOP (30) bs.machine_name, bs.server_name, DB_NAME(DB_ID(bs.database_name)) AS [Database Name], bs.
CONVERT (BIGINT, bs.backup_size / 1048576 ) AS [Uncompressed Backup Size (MB)],
CONVERT (BIGINT, bs.compressed_backup_size / 1048576 ) AS [Compressed Backup Size (MB)],
CONVERT (NUMERIC (20,2), (CONVERT (FLOAT, bs.backup_size) /
CONVERT (FLOAT, bs.compressed_backup_size))) AS [Compression Ratio], bs.has_backup_checksums, bs.is_copy_
DATEDIFF (SECOND, bs.backup_start_date, bs.backup_finish_date) AS [Backup Elapsed Time (sec)],
bs.backup_finish_date AS [Backup Finish Date], bmf.physical_device_name AS [Backup Location], bmf.physica
FROM msdb.dbo.backupset AS bs WITH (NOLOCK)
INNER JOIN msdb.dbo.backupmediafamily AS bmf WITH (NOLOCK)
ON bs.media_set_id = bmf.media_set_id
WHERE DB_ID(bs.database_name) = DB_ID()
AND bs.[type] = 'D'
ORDER BY bs.backup_finish_date DESC OPTION (RECOMPILE);
```

2. Configure an XEvent session

    The public documentation Configure XEvent session article details how to configure a simple tracking XEvent session, a verbose tracking XEvent session and how to monitor the backup progress.

    a) To configure a simple tracking XEvent session

Run this T-SQL to configure a simple XEvent session:

```
CREATE EVENT SESSION [Simple backup trace] ON SERVER
ADD EVENT sqlserver.backup_restore_progress_trace(
WHERE operation_type = 0
AND trace_message LIKE '%100 percent%')
ADD TARGET package0.ring_buffer
WITH(STARTUP_STATE=ON)
GO
ALTER EVENT SESSION [Simple backup trace] ON SERVER
STATE = start;
```

b) To configure a verbose tracking XEvent session

Run this T-SQL to configure a verbose XEvent session:

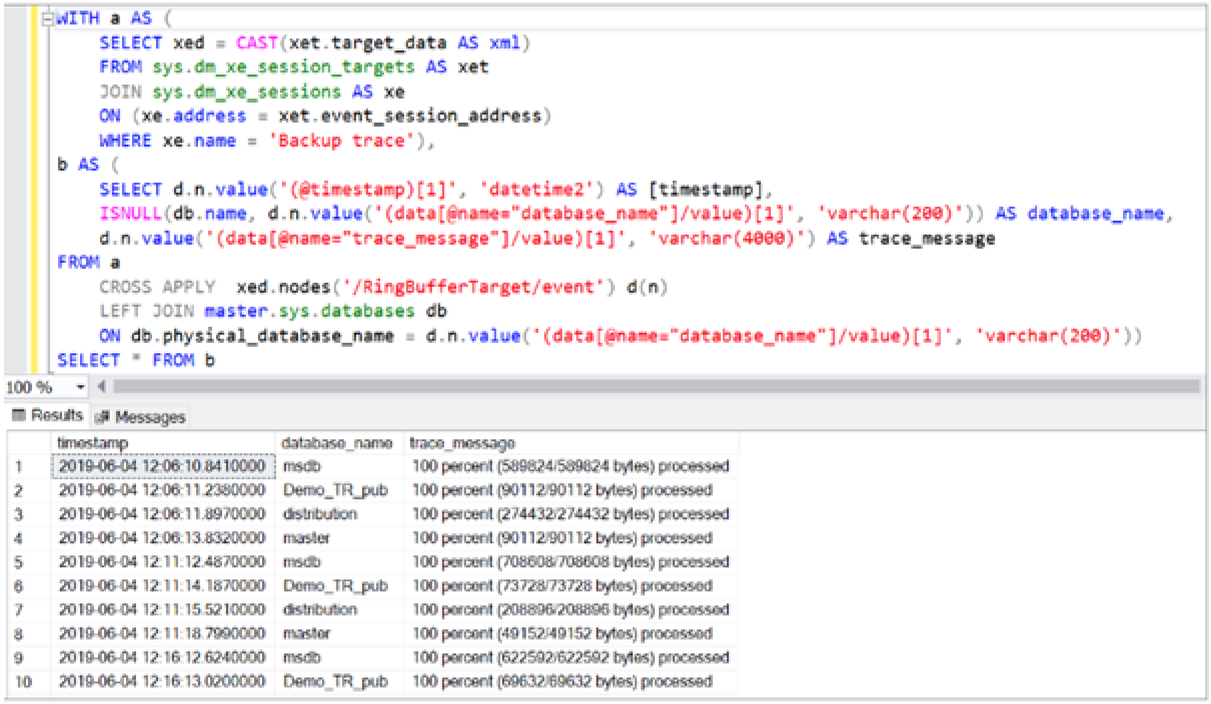
```
CREATE EVENT SESSION [Verbose backup trace] ON SERVER
ADD EVENT sqlserver.backup_restore_progress_trace(
WHERE (
              [operation_type]=(0) AND (
              [trace_message] like '%100 percent%' OR
              [trace_message] like '%BACKUP DATABASE%' OR [trace_message] like '%BACKUP LOG%'))
       )
ADD TARGET package0.ring_buffer
WITH (MAX_MEMORY=4096 KB,EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS,
       MAX_DISPATCH_LATENCY=30 SECONDS,MAX_EVENT_SIZE=0 KB,MEMORY_PARTITION_MODE=NONE,
       TRACK_CAUSALITY=OFF,STARTUP_STATE=ON)

ALTER EVENT SESSION [Verbose backup trace] ON SERVER
STATE = start;
```

3. To monitor the backup progress

This T-SQL statement queries the simple tracking XEvent session and returns the name of the database, the total number of bytes processed, and the time the backup completed.

```
WITH
a AS (SELECT xed = CAST(xet.target_data AS xml)
FROM sys.dm_xe_session_targets AS xet
JOIN sys.dm_xe_sessions AS xe
ON (xe.address = xet.event_session_address)
WHERE xe.name = 'Backup trace'),
b AS(SELECT
d.n.value('(@timestamp)[1]', 'datetime2') AS [timestamp],
ISNULL(db.name, d.n.value('(data[@name="database_name"]/value)[1]', 'varchar(200)')) AS database_name,
d.n.value('(data[@name="trace_message"]/value)[1]', 'varchar(4000)') AS trace_message
FROM a
CROSS APPLY  xed.nodes('/RingBufferTarget/event') d(n)
LEFT JOIN master.sys.databases db
ON db.physical_database_name = d.n.value('(data[@name="database_name"]/value)[1]', 'varchar(200)'))
SELECT * FROM b
```

The following screenshot shows an example output of the above query for the simple tracking XEvent session:



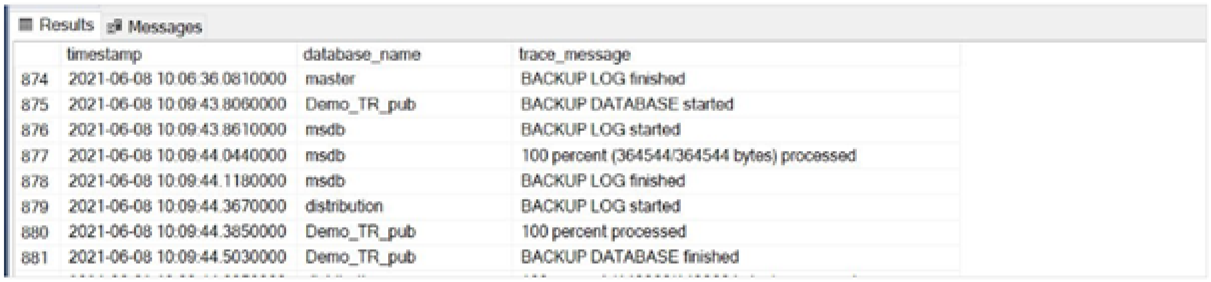
```
WITH a AS (
    SELECT xed = CAST(xet.target_data AS xml)
    FROM sys.dm_xe_session_targets AS xet
    JOIN sys.dm_xe_sessions AS xe
    ON (xe.address = xet.event_session_address)
    WHERE xe.name = 'Backup trace'),
b AS (
    SELECT d.n.value('(@timestamp)[1]', 'datetime2') AS [timestamp],
    ISNULL(db.name, d.n.value('(data[@name="database_name"]/value)[1]', 'varchar(200)')) AS database_name,
    d.n.value('(data[@name="trace_message"]/value)[1]', 'varchar(4000)') AS trace_message
FROM a
    CROSS APPLY  xed.nodes('/RingBufferTarget/event') d(n)
    LEFT JOIN master.sys.databases db
    ON db.physical_database_name = d.n.value('(data[@name="database_name"]/value)[1]', 'varchar(200)'))
SELECT * FROM b
```

	timestamp	database_name	trace_message
1	2019-06-04 12:06:10.8410000	msdb	100 percent (589824/589824 bytes) processed
2	2019-06-04 12:06:11.2380000	Demo_TR_pub	100 percent (90112/90112 bytes) processed
3	2019-06-04 12:06:11.8970000	distribution	100 percent (274432/274432 bytes) processed
4	2019-06-04 12:06:13.8320000	master	100 percent (90112/90112 bytes) processed
5	2019-06-04 12:11:12.4870000	msdb	100 percent (708608/708608 bytes) processed
6	2019-06-04 12:11:14.1870000	Demo_TR_pub	100 percent (73728/73728 bytes) processed
7	2019-06-04 12:11:15.5210000	distribution	100 percent (208896/208896 bytes) processed
8	2019-06-04 12:11:18.7990000	master	100 percent (49152/49152 bytes) processed
9	2019-06-04 12:16:12.6240000	msdb	100 percent (622592/622592 bytes) processed
10	2019-06-04 12:16:13.0200000	Demo_TR_pub	100 percent (69632/69632 bytes) processed

This T-SQL statement queries the verbose XEvent session returning the name of the database, as well as the start and finish time of the full, differential and log backups.

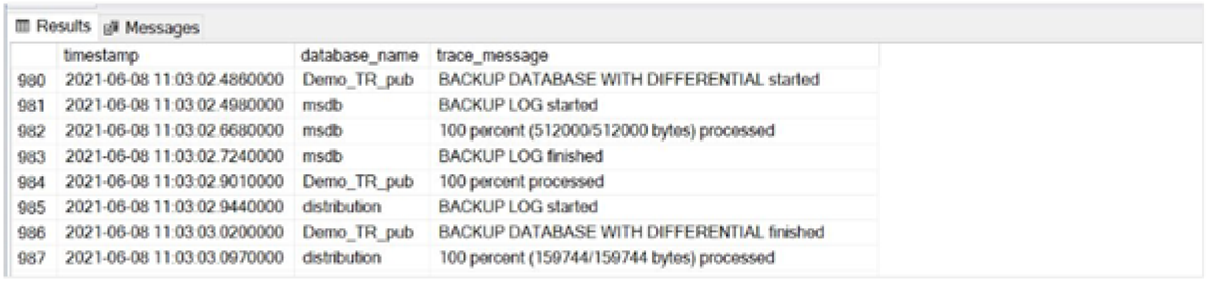
```
WITH
a AS (SELECT xed = CAST(xet.target_data AS xml)
FROM sys.dm_xe_session_targets AS xet
JOIN sys.dm_xe_sessions AS xe
ON (xe.address = xet.event_session_address)
WHERE xe.name = 'Verbose backup trace'),
b AS(SELECT
d.n.value('(@timestamp)[1]', 'datetime2') AS [timestamp],
ISNULL(db.name, d.n.value('(data[@name="database_name"]/value)[1]', 'varchar(200)')) AS database_name,
d.n.value('(data[@name="trace_message"]/value)[1]', 'varchar(4000)') AS trace_message
FROM a
CROSS APPLY  xed.nodes('/RingBufferTarget/event') d(n)
LEFT JOIN master.sys.databases db
ON db.physical_database_name = d.n.value('(data[@name="database_name"]/value)[1]', 'varchar(200)'))
SELECT * FROM b
```

The following screenshot shows an example of the output for a full backup from the XEvent session:



	timestamp	database_name	trace_message
874	2021-06-08 10:06:36.0810000	master	BACKUP LOG finished
875	2021-06-08 10:09:43.8060000	Demo_TR_pub	BACKUP DATABASE started
876	2021-06-08 10:09:43.8610000	msdb	BACKUP LOG started
877	2021-06-08 10:09:44.0440000	msdb	100 percent (364544/364544 bytes) processed
878	2021-06-08 10:09:44.1180000	msdb	BACKUP LOG finished
879	2021-06-08 10:09:44.3670000	distribution	BACKUP LOG started
880	2021-06-08 10:09:44.3850000	Demo_TR_pub	100 percent processed
881	2021-06-08 10:09:44.5030000	Demo_TR_pub	BACKUP DATABASE finished

The following screenshot shows an example of an output of a differential backup in the XEvent session:



	timestamp	database_name	trace_message
980	2021-06-08 11:03:02.4860000	Demo_TR_pub	BACKUP DATABASE WITH DIFFERENTIAL started
981	2021-06-08 11:03:02.4980000	msdb	BACKUP LOG started
982	2021-06-08 11:03:02.6680000	msdb	100 percent (512000/512000 bytes) processed
983	2021-06-08 11:03:02.7240000	msdb	BACKUP LOG finished
984	2021-06-08 11:03:02.9010000	Demo_TR_pub	100 percent processed
985	2021-06-08 11:03:02.9440000	distribution	BACKUP LOG started
986	2021-06-08 11:03:03.0200000	Demo_TR_pub	BACKUP DATABASE WITH DIFFERENTIAL finished
987	2021-06-08 11:03:03.0970000	distribution	100 percent (159744/159744 bytes) processed

Note: Unfortunately, monitoring SQL Managed Instance database recovery is not possible from the customer side.

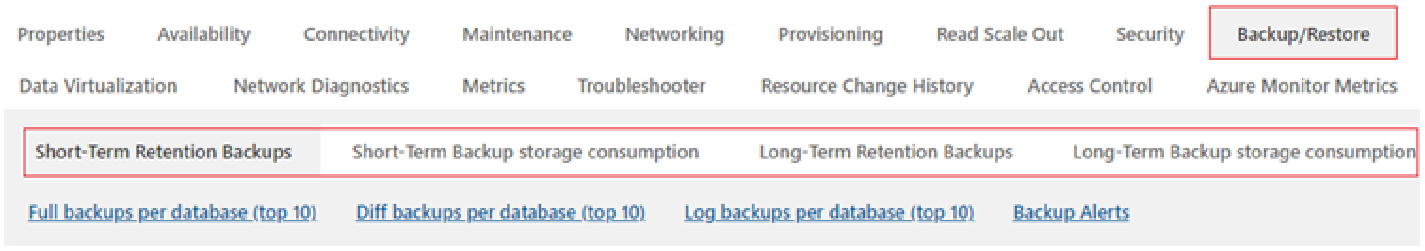
References

[Monitor backup activity for Azure SQL Managed Instance](#)
[Tips & Tricks #4: Monitoring Backup History for Azure SQL Database & Azure SQL Managed Instance](#)

CSS Side

This section details how to use ASC, Kusto and XTS to investigate and troubleshoot Managed Instance backup and restores.

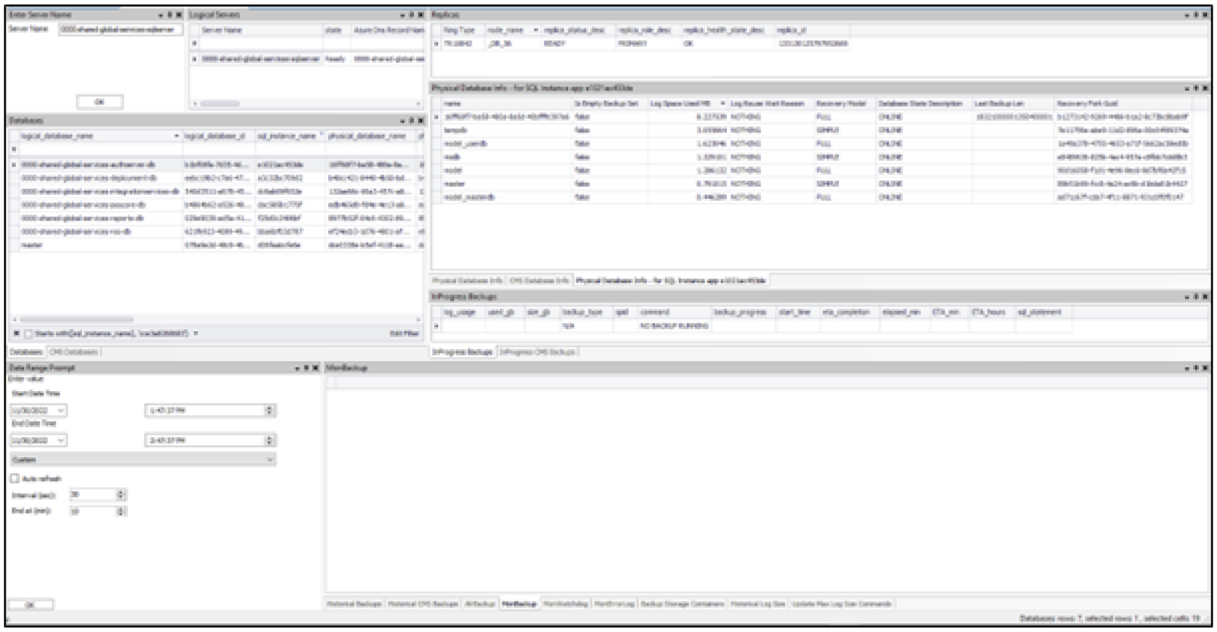
1. ASC
   From ASC navigate to the SQL Managed Instance at the server-level. Check the Backup/Restore tab. This screenshot from ASC shows the Short-Term Retention Backups (PITR backups) and the Long-Term Retention Backups (LTR backups) details.



2. Kusto
   The following Kusto queries provide details about the full backups per database, and backup alerts.

```
//Find backup per start/end timestamp
MonBackup
| where AppName == "{AppName}"
| where TIMESTAMP > datetime({StartTime}) and TIMESTAMP < datetime({EndTime})
| where  event_type == "BACKUP_METADATA_DETAILS" and (backup_type == "Full" or backup_type == "Diff")
| project originalEventTimestamp, NodeName, backup_type, backup_start_date, backup_end_date, backup_size,
uncompressed_backup_size, backup_path, AppName, logical_database_id, logical_database_name

//Full backups per database (top 10)
//Diff backups per database (top 10)
//Log backups per database (top 10)
let topNumber=10;
let backuptype = 'Full';//'Diff' //'Log';
let MIName = '({server_name})';
let AllDatabases = materialize( MonBackup
| where logical_server_name =~ '({server_name})'
| where backup_type =~ backuptype
| where event_type == 'BACKUP_METADATA_DETAILS'
| extend BackupSizeMB = round(todecimal(backup_size)/1024/1024,2)
| extend uncompressed_backup_size_MB = round(todecimal(uncompressed_backup_size)/1024/1024,2)
| extend backup_size_GB = round(todecimal(backup_size)/1024/1024/1024,2)
| extend uncompressed_backup_size_GB = round(todecimal(uncompressed_backup_size)/1024/1024/1024,2)
| project PreciseTimeStamp, logical_database_id=tolower(logical_database_name), BackupSizeMB, backup_star
| join kind=leftouter
(
MonAnalyticsDBSnapshot
| where logical_server_name =~ 'aeucceqfp2mi002'
| summarize arg_max(TIMESTAMP,*) by logical_database_id
| project logical_database_id=tolower(logical_database_id), logical_database_name
) on logical_database_id
| extend DatabaseName = iif(logical_database_name != '', logical_database_name, logical_database_id)
| project PreciseTimeStamp, DatabaseName, BackupSizeMB
| summarize BackupSizeMB = sum(BackupSizeMB) by DatabaseName, bin(PreciseTimeStamp,1d) );
let topX = (AllDatabases | summarize sum(BackupSizeMB) by DatabaseName
| top topNumber by sum_BackupSizeMB | project DatabaseName);
AllDatabases | where DatabaseName in (topX)
| project PreciseTimeStamp, DatabaseName, BackupSizeMB
| order by PreciseTimeStamp asc, DatabaseName asc

//Backup Alerts
MonBackup
| where originalEventTimestamp >= datetime(2022-11-30 01:53:00) and originalEventTimestamp <= datetime(20
| where LogicalServerName =~ 'aeucceqfp2mi002'
| where isempty('') or logical_database_id =~ ''
| where isnotempty(alert_type)
| project originalEventTimestamp, logical_server_name, logical_database_name, alert_type, br_error_detail
| extend logical_database_id = tolower(logical_database_name)
| join kind=leftouter (
MonAnalyticsDBSnapshot
| where logical_server_name =~ 'aeucceqfp2mi002'
| summarize arg_max(TIMESTAMP, logical_database_name) by logical_database_id
| extend logical_database_id = tolower(logical_database_id)
) on logical_database_id
| project-rename Database = logical_database_name, Timestamp = originalEventTimestamp, AlertType = alert_
| project Timestamp, Database, AlertType, ErrorDetails
| order by Timestamp asc
```

3. XTS
   The XTS view, `Sterlingbackuptroubleshooting.xts`, provides useful information to start troubleshooting database backups issues.



This CMS query returns the first backup time, and the backup retention in days.

```
XTS (CMS):
-- Check first_backup_time, backup_retention_days
select logical_database_name,state,first_backup_time,backup_retention_days
from logical_databases
Where logical_server_name = '0000-shared-global-services-sqlserver'
```

Troubleshooting Managed Database recovery

Databases sometimes go into recovery mode during SLO change, Geo-failover/replication initiation or planned and unplanned failovers. For example, during an SLO change the database is copied to the new hardware with the new tier set. The database by design remains accessible on the original hardware until the copy operation is completed, when connections are then switched to the new database at the new service tier. However, the size of the transaction log increases, depending on the customer workload, and the new database will not be ready until the copy operation is completed followed by transactions commit (aka the redo queue). This recovery can cause delays in switching connections to the new database. The transaction log at this point needs to replay to new database (the recovery phase).

Kusto investigation:

1. Check `MonManagementOperations` table, operation_type "FailoverManagedFailoverGroup"
2. Mark down the `request_id`
3. Check in `MonManagement` database `old_state` and `new_state` , reference `request_id` from step 1
4. When required check `MonSQLSystemHealth` table for recovery messages
5. Check `MonRecoveryTrace` table for `redo queue` or trace message "recovery completed for database"
6. To check `redo_queue_size` query Kusto table `MonDmHadrReplicationStates`

```
[Step 1]
let logicalServerName = ({servername});
MonManagementOperations
| where originalEventTimestamp > datetime({startime}) and originalEventTimestamp < datetime({endtime})
| where operation_type contains "FailoverManagedFailoverGroup"
| project originalEventTimestamp , request_id, event,operation_type, exception_type, message, error_message, s
operation_parameters
| order by originalEventTimestamp desc

[Step 3]
MonManagement
| where TIMESTAMP > datetime({StartTime})
| where TIMESTAMP < datetime({EndTime})
| where request_id == toupper({request_id})
| where isnotempty(old_state)
| where state_machine_type == "ManagedPhysicalDatabaseLinkStateMachine"
| where keys has "1cfb7431-833e-467d-8144-37b3c203da2a"
| summarize min(TIMESTAMP), max(TIMESTAMP) by old_state, new_state
| order by min_TIMESTAMP asc

[Step 4]
MonSQLSystemHealth
| where TIMESTAMP > datetime({StartTime})
| where TIMESTAMP < datetime({EndTime})
| where AppName == tolower({AppName})
| where message has ({request_id}) or message has ({logical_database_id}) //or message has "database id: 9"
| project TIMESTAMP, message

[Step 5]
MonRecoveryTrace
| where TIMESTAMP > datetime({StartTime})
| where TIMESTAMP < datetime({EndTime})
| where database_name == ({logical_db_name})
| where trace_message has "redo queue" or trace_message has "Recovery completed for database"
| project TIMESTAMP, trace_message

[Step 6]
MonDmDbHadrReplicaStates
| where TIMESTAMP > datetime({StartTime})
| where TIMESTAMP < datetime({EndTime})
| where AppName == tolower({AppName})
| where redo_queue_size > -1
| summarize arg_max(redo_queue_size, TIMESTAMP) by logical_database_name
| order by redo_queue_size desc
```

Additional Kusto queries

```
//Basic recovery info for database
MonRecoveryTrace
| where trace_message in ('Starting database recovery.', 'Redo phase done.')
    or trace_message startswith "Recovery completed for database "
    or trace_message startswith "Dirty Page Table summary"
    or trace_message startswith "Estimate: Redo ="
| where TIMESTAMP >= datetime("{StartTime}")
| where AppName == "{AppName}"
| where database_name == "{LogicalDatabaseId}"
| order by SubscriptionId, LogicalServerName, database_name, originalEventTimestamp asc
| parse trace_message with "Dirty Page Table summary: page count = " dirtyPages:int "."
| parse trace_message with "Estimate: REDO = " redoLogBytes:int64 " bytes of log, UNDO = " undoTransactions:in
| extend redoLogMb = redoLogBytes / 1048576.0

MonRecoveryTrace
| where AppName == "a40dc3f0b944"
| where TIMESTAMP >= ago(6h)
| where NodeName=="DB.33"
| where trace_message contains "Recovery of database"
| project TIMESTAMP, trace_message


//Duration of recovery
MonRecoveryTrace
| where AppName == "{AppName}"
| where database_name =~ "{LogicalDatabaseId}"
| where event == "database_recovery_times"
| project TIMESTAMP, recovery_time, recovery_step
```

How good have you found this content?