SQL Replication Kusto Queries

Last updated by | Holger Linke | Dec 20, 2022 at 5:29 AM PST

Contents

Kusto queries related to Transactional Replication

There are only a few Kusto queries that can provide relevant information related to replication. They may give you a quick idea on the customer's replication status. For troubleshooting, you will likely need to get further details from the customer though.

Replication Agent run status and performance metrics

The LogicalServerName should point to the Distributor instance - only set the short name, without DNS suffix or FQDN.
If you want to filter for failed executions, then run it with the runstatus/sync_status filter:

```
MonTranReplTraces
| where TIMESTAMP >= datetime(2022-04-12 12:00:00Z)
| where TIMESTAMP <= datetime(2022-04-13 18:00:00Z)
//| where SubscriptionId =~ "<insert subscription ID here>"
| where LogicalServerName =~ "<insert name of Distributor here>"
//| where runstatus in (6) or sync_status in (6) // runstatus=6 means: agent failed
| project TIMESTAMP, originalEventTimestamp, NodeName, AppName, LogicalServerName, db_id, logical_database_nam
| limit 3000
```

```
MonLogReaderTraces
| where TIMESTAMP >= datetime(2022-04-22 09:0:00)
| where TIMESTAMP <= datetime(2022-04-22 18:00:00)
//| where SubscriptionId =~ "<insert subscription ID here>"
| where LogicalServerName =~ "<insert name of Distributor here>"
//| where logical_database_name =~ "<insert publisher database name here>"
| project TIMESTAMP, NodeName, AppName, LogicalServerName, logical_database_name, phase_number, phase_state_na
| limit 5000
```

SQL Agent job status reports

This Kusto query will show you events and messages related to the SQL Agent job execution. The LogicalServerName should point to the Distributor instance - only set the short name, without DNS suffix or FQDN.

```
let srv = "servername";  // Distributor instance name
let startTime = datetime(2022-12-19 06:00:00Z);
let endTime = datetime(2022-12-20 23:00:00Z);
let timeRange = ago(7d);
MonSqlAgent
| where  TIMESTAMP >= startTime
| where  TIMESTAMP <= endTime
//| where  TIMESTAMP >= timeRange
| where LogicalServerName =~ srv
| where message_type in~ ("ERROR", "STARTUP_MSG", "WARNING")
//| where message_type !in~ ("INFORMATION", "STARTUP_MSG")
| project TIMESTAMP, AppName, LogicalServerName, event, resource_id, message_type, message
```

Error log details through MonSQLSystemHealth

The LogicalServerName should point to the Distributor instance - only set the short name, without DNS suffix or FQDN.
You will likely only see error_id=14151 "Replication agent failed" or error_id=14152 "Replication agent scheduled for retry", none of the other errors, but at least with an exact timestamp:

```
MonSQLSystemHealth
| where TIMESTAMP >= datetime(2022-04-12 12:00:00Z)
| where TIMESTAMP <= datetime(2022-04-13 18:00:00Z)
//| where SubscriptionId =~ "<insert subscription ID here>"
| where LogicalServerName =~ "<insert name of Distributor here>"
| where error_id > 0
| where error_id in (14151, 14152, 18856, 21411, 2601, 2627, 20598) //typical replication messages
| project TIMESTAMP, NodeName, AppName, instance_rg_size, error_id, message
| limit 1000
```

Performance metrics about commands applied to the Subscriber

Here is an example for monitoring the bulk insert operations when applying the snapshot to the Subscriber. For other scenarios, you'd rather check the ASC troublehshooter first to identify the query_id/query_hash values of slow queries.

```
// returns individual queries based on query_id and the performance statistics
// includes statement_type column to see if this is a select, insert, update, delete, bulk operation
// helpful to see if/when the snapshot from transactional replication has been applied

let srv = "servername";     // Subscriber server
let db  = "databasename";   // Subscriber database
let startTime = datetime(2022-11-23 06:00:00Z);
let endTime   = datetime(2022-11-24 23:00:00Z);
let timeRange = ago(7d);
MonWiQdsExecStats
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
// | where TIMESTAMP >= timeRange
| where LogicalServerName =~ srv
| where database_name =~ db
| where statement_type in ("x_estypInsertBulk")
//| where statement_type in ("x_estypInsertBulk", "x_estypInsert")
| extend interval_start_time_date = interval_start_time / 4294967296
| extend interval_start_time_time = interval_start_time - 4294967296 * interval_start_time_date
| extend interval_start = datetime(1900-1-1) + time(1d) * interval_start_time_date + time(1s) * (interval_star
| extend interval_start_time_date = interval_start_time / 4294967296
| extend interval_start_time_time = interval_start_time - 4294967296 * interval_start_time_date
| extend interval_start = datetime(1900-1-1) + time(1d) * interval_start_time_date + time(1s) * (interval_star
| extend interval_end_time_date = interval_end_time / 4294967296
| extend interval_end_time_time = interval_end_time - 4294967296 * interval_end_time_date
| extend interval_end = datetime(1900-1-1) + time(1d) * interval_end_time_date + time(1s) * (interval_end_time
| extend Average_cpu_time = cpu_time / execution_count,
        Average_logical_reads = logical_reads / execution_count,
        Average_logical_writes = logical_writes / execution_count,
        Average_physical_reads = physical_reads / execution_count,
        Average_elapsed_time = elapsed_time / execution_count,
        Average_log_bytes_used = log_bytes_used / execution_count,
        Average_rowcount = rowcount / execution_count
//| where query_id == "5139412"
//| where query_id in ("5138681","5138688","5138691","5138692")
//| where query_hash =~ "0x6519B34634C2ABE6"
//| where query_hash in ("0x6519B34634C2ABE6", "0x81A344CE3CACB706", "0x45BB4D864FDD5E06", "0x9E56C411B1E1A0DD
| project originalEventTimestamp, TIMESTAMP,LogicalServerName,database_name,query_id,query_hash,query_plan_has
Average_elapsed_time,Average_log_bytes_used,Average_rowcount ,logical_writes,logical_reads,min_logical_reads,m
```

How good have you found this content?