Logins (Managed Instance)

Last updated by | Vitor Tomaz | Aug 5, 2020 at 12:42 PM PDT

```
//*****************************************************************************************************************
// login
//*****************************************************************************************************************
// L.01
// Login success rate etc
MonLogin
|where TIMESTAMP >= datetime({StartTime})  and  TIMESTAMP <= datetime({EndTime})
| where LogicalServerName =~ "{LogicalServerName}" and AppName == "{AppName}"
| where event == "process_login_finish"   and package == "sqlserver"
| where is_user_error == 0
| extend success=case(is_success == 1, 1, 0)
| extend failure=case(is_success == 0, 1, 0)
| extend failure_enqueue_time_ms=case(is_success == 1, total_time_ms, 0)
| summarize Successful_Login_Count=sum(success), Failed_Login_Count=sum(failure),
FailureRate=sum(failure)*100/(sum(success)+sum(failure)),
avg_failure_enqueue_time_ms=sum(failure_enqueue_time_ms)/sum(failure),
Avg_Login_Time_ms=avg(total_time_ms),  Avg_Enqueu_Time_ms=avg(enqueue_time_ms) by
bin(originalEventTimestamp, 1min)
| render timechart

// L.02
// close look at login failures or long logings
MonLogin
|where TIMESTAMP >= datetime({StartTime})  and  TIMESTAMP <= datetime({EndTime})
| where LogicalServerName =~ "{LogicalServerName}" and AppName == "{AppName}"
| where event == "process_login_finish"   and package == "sqlserver"
| where (is_success == false and is_user_error == false) or total_time_ms > 14000
| project TIMESTAMP , enqueue_time_ms ,total_time_ms,  find_login_ms, message

// L.03
// Login resource usage comparison

// Use xdb for rgGroupPrefix


|where event == 'aggregated_workload_groups_plus_history'
| where AppName == "{AppName}" and NodeName =~ "{NodeName}" // and LogicalServerName =~ "
{LogicalServerName}}"
| where  end_aggregated_sample >= datetime({StartTime}) and  end_aggregated_sample <=
datetime({EndTime})
| where group_name contains "{rgGroupPrefix}" or ( "{rgGroupPrefix}" == "SloPri" and group_name  contains
"UserPrimaryGroup" )
//  or ( "{rgGroupPrefix}" == "SloSec" and group_name contains "UserSecondaryGroup" )
| where delta_total_cpu_usage_ms <> total_cpu_usage_ms
| project end_utc_date = end_aggregated_sample , name = group_name , start_utc_date =
```

```
start_aggregated_sample , delta_total_cpu_usage_ms,  delta_total_request_count, active_request_count =
active_request_count_max , active_session_count = active_session_count_max  , delta_writes_completed_total,
delta_reads_completed_total,
delta_log_bytes_used_total, delta_log_temp_db_bytes_used_total, used_databasepages_kb = 0    ,
physical_database_name
| extend duration =  (end_utc_date - start_utc_date)/time(1s)
| extend CpuCoreUsage = (delta_total_cpu_usage_ms/1000.0)/ duration
| extend TotalRequests = (delta_total_request_count  *1.0) /1.0
| extend ActiveRequests = (active_request_count *1.0)  /1.0
| extend ActiveSessions = (active_session_count *1.0) /1.0
| extend TotalIOPS = ((delta_reads_completed_total + delta_writes_completed_total)*1.0)/duration
| extend ReadIOPS = (delta_reads_completed_total*1.0)/duration
| extend WriteIOPS = (delta_writes_completed_total*1.0)/duration
| extend LogKbps = (delta_log_bytes_used_total)/(1024.0*duration)
| extend TempDbLogKbps = (delta_log_temp_db_bytes_used_total)/(1024.0*duration)
| extend UsedDatabasePagesMb = used_databasepages_kb*1.0/(1024)
| top-nested of bin(end_utc_date, 5min) by avg(CpuCoreUsage), top-nested 5 of name by
avg_cpucoreusage=avg(CpuCoreUsage) desc
| order by end_utc_date desc
//| project end_utc_date, name, CpuCoreUsage, TotalIOPS, ReadIOPS, WriteIOPS, LogKbps, TempDbLogKbps,
TotalRequests, ActiveRequests, ActiveSessions, UsedDatabasePagesMb
//| project end_utc_date, CpuCoreUsage, name
| project end_utc_date, avg_cpucoreusage, name
| render timechart
```

How good have you found this content?