Installing and using the ‘Pool Telemetry’ PowerShell scripts

This is a quick guide to using the PowerShell scripts provided for off-loading elastic pool and elastic database telemetry data into a separate telemetry database.

The scripts and this guide are copyright Microsoft Corporations and are provided as samples. They are not part of any Azure service and are not covered by any SLA or other Azure-related agreements. They are provided as-is with no warranties express or implied. Microsoft takes no responsibility for the use of the scripts or the accuracy of this document. Familiarize yourself with the scripts before using them.

This user guide applies to the PowerShell scripts, PoolTelemetryRunner.ps1 and PoolTelemetry.ps1.

# What do the scripts do?

The scripts are used to extract telemetry data associated with SQL Database elastic database pools and elastic databases and upload it to a separate telemetry database.

There is a **runner script**, PoolTelemetryRunner.ps1, which needs to be modified for your environment to identify one or more servers on which elastic pools are hosted and a telemetry database in which telemetry data is to be gathered. The runner script executes the **data collection script**, PoolTelemetry.ps1, and spawns a PowerShell job for each server.

Each data collection job executes in the background on a pre-determined schedule and if necessary creates the telemetry tables in the telemetry database. It then connects to the master database on the server and retrieves elastic pool telemetry data and loads that to the telemetry database. It then queries the master database to determine the current elastic databases on the server, resident in each of the pools identified in the prior step. It then connects to each database in turn and retrieves and loads telemetry data for that database. It then sleeps for a period before waking up and repeating the data collection cycle.

# Installing the scripts

Both scripts, PoolTelemetry.ps1 and PooltelemetryRunner.ps1, should be copied to the same directory.

You must have installed and imported the latest (1.x) Azure PowerShell modules and SQL PowerShell (sqlps) modules. The cmdlet Invoke-SQLCmd is used to execute SQL scripts.

# Customizing the PoolTelemetryRunner script

The runner script should be customized to provide information about the servers from which pool data will be retrieved, as well as to customize frequency and overall duration of data collection.

## Azure log in

The script requires you to log on to Azure with a Microsoft Id, either a personal Id or a work or school Id. The Id used must have read access to the servers in the subscription under which the pools and databases have been created (the telemetry database can be created under a different subscription).

Set the SubscriptionName. If the Microsoft Id used to login has access to multiple Azure subscriptions this ensures that the one under which the servers will be used. All servers to be reported on must be created under the same subscription.

$AzureSubscriptionName = ‘<subscription name>'

## SQL user names for source servers and the telemetry server

While the runner script uses ARM PowerShell cmdlets to gather information about resource groups and servers, the data collection script uses SQL queries to retrieve data. SQL user names and passwords must be provided at run time which will be passed to the data collection script job to access the source servers and databases using SQL DMVs. The scripts assume the same SQL user name and password are used for all source servers. The telemetry server user name and password are provided separately.

User credentials are gathered via dialog boxes at run time to avoid storing passwords in the script. You can specify the User Names for each dialog so that these do not need to be entered each time the script is run. To do this add a –UserName parameter to each of the two credentials.

$sourceCred = Get-Credential -Message 'User name and password for source server'   
–UserName '<user name>'

$outputServerCred = Get-Credential -Message 'User name and password for telemetry database’ –UserName '<user name>'

## Source resource group and server

Provide the resource group if data is to be gathered from all servers in a specific resource group or a specific server.

$resourceGroupName = '<resource group name >'

Provide the server name if data is to be gathered from all a specific server.

$serverName = '<server name>'

## Telemetry server and database

It is assumed that telemetry is to be loaded to an Azure SQL Database.

Provide the telemetry database server name.

$outputServerName = '<telemetry server name>’

Provide the telemetry database name.

$outputDatabaseName = '<telemetry database name>'

## Collection interval and job duration

Provide the interval in minutes. This defines both how far back the data collection will look on each execution and the interval between executions. A value between 15-30 minutes is probably most appropriate. Note that fine-grained database telemetry (15 second averages) is only retained in each database for 60 minutes, beyond that it based on 5 minute averages. Pool telemetry in the master database is always based on 5 minute averages. 5 minute averaged data is retained for 14 days.

$intervalMinutes = 15

Provide the job duration in minutes. This defines how long the job will execute for in the background. A value of zero will cause the job to execute once only. The value is best set to a multiple of the interval.

$durationMinutes = 0

## Specify the source server(s) to use

The script allows either a single server to be specified or multiple. Several sample queries are provided but in general only one should be used, the others should be commented out. The script requires that the $serverList variable is populated as input to the job execution. Either uncomment and use one of the queries that populates $serverList or use one of the queries that populates $resourceList and then uncomment the section in the script that translates the $resourceList into a $serverList. If not using $resourceList leave that section commented out.

# Executing the runner script

The runner script PoolTelemetryRunner.ps1 should be executed from within PowerShell.

The script will prompt for Azure login and the user name and password for the source servers and the user name and password for the telemetry server. It will then spawn a PowerShell job for each server that has been identified within the script. Each job will run in the background for the time specified in $durationMinutes. It will gather data for the most recent period defined by the interval value and load this, then sleep until the next data gathering point, wake up, gather and load more data and then sleep again, etc.

To manage the jobs created, in PowerShell use:

Get-Job - to see jobs in progress

Receive-Job <job id> -Keep to see the current console output from each job. If you don’t use   
–Keep the output is not retained (but doesn’t affect data collection)

Stop-Job \* to stop all jobs (or specify a job id)

Remove-Job \* to remove jobs (or specify a job id)

# Inspecting the telemetry that is collected

Use SSMS or other tools such as PowerBI to inspect and query the telemetry database. Data is gathered in two tables:

pool\_resource\_stats – has the resource usage data for all the elastic pools in the specified servers for the specified duration.

db\_resource\_stats -- has the resource usage data for all the databases in the elastic pools for the specified duration.

There is also a pre-defined Table-valued function to help querying the aggregate resource usage data for a specified time interval.

[dbo].[get\_aggregated\_pool\_metrics](@start datetime, @end datetime)

where @start and @end are the time interval of interest for querying.

For example:

-- Once the telemetry is started collecting, TVF can be called with the required time interval to get the top 10 elastic pools with highest average eDTU consumption in the specified time period.

select top 10 \* from [dbo].[get\_aggregated\_pool\_metrics]('04/29/2016 21:00:00', '04/29/2016 23:00:00')

order by avg\_DTU\_percent desc

Data can be queried while data collection is in progress. Note that if the scripts are stopped and started again within a short period they may add duplicate rows to the telemetry tables.