Azure Log Analytics

- KQL quick commands
- Write custom logs via PowerShell
- Query Azure Firewall Logs
- Create Workbooks with KQL queries
- Use of variables & dynamic content in Azure Workbooks
- Create log health dashboard using time based data

KQL quick commands

Count entries in table

```
<loganalyticstablename> | summarize count()
```

Get log storage usage

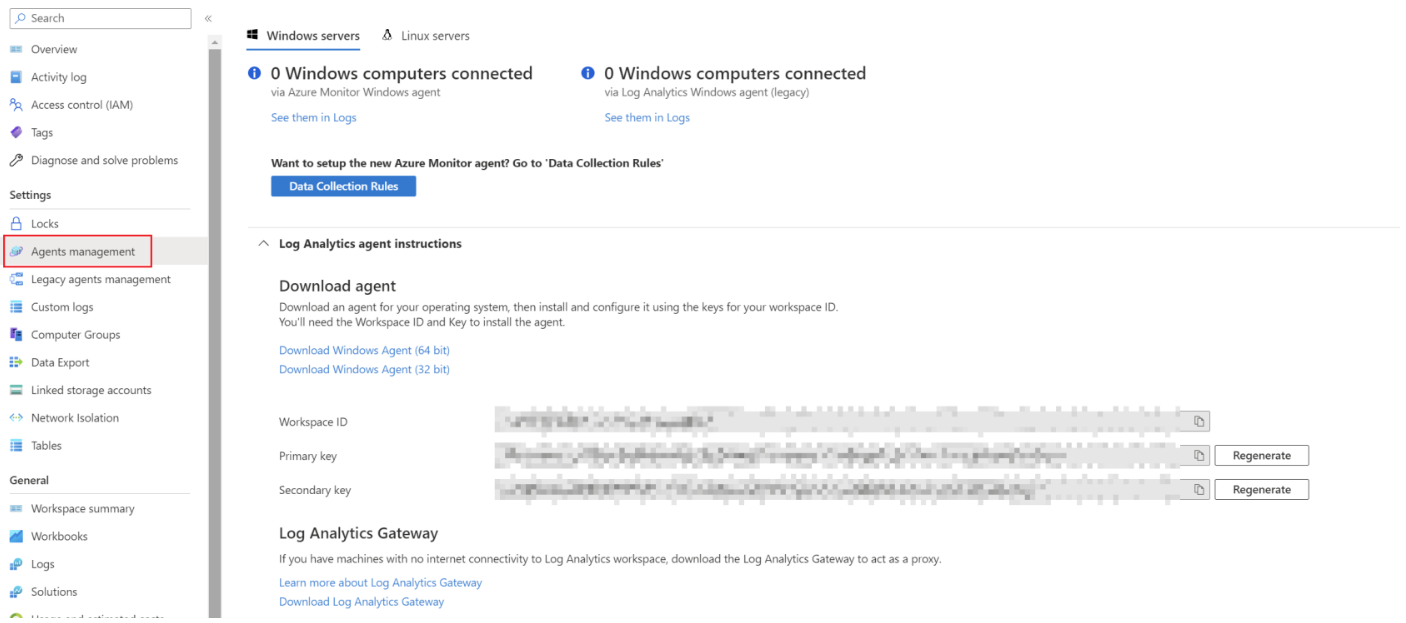
```
Usage | where IsBillable | summarize DataGB = sum(Quantity / 1000)
```

Write custom logs via PowerShell

With Log Analytics you can write custom logs. For this the API can be used to write new logs to the Log Analytics Workspace. The whole API is documented here: https://learn.microsoft.com/en-us/rest/api/loganalytics/create-request

Script template

This script template uses the first three variables. The first two can be found under "Agents management" -> "Log Analytics agent instructions".



0 Windows computers connected
via Azure Monitor Windows agent

0 Windows computers connected
via Log Analytics Windows agent (legacy)

Want to setup the new Azure Monitor agent? Go to 'Data Collection Rules'

Download agent

Download an agent for your operating system, then install and configure it using the keys for your workspace ID.
You'll need the Workspace ID and Key to install the agent.

Workspace ID

Primary key

Secondary key

Log Analytics Gateway

If you have machines with no internet connectivity to Log Analytics workspace, download the Log Analytics Gateway to act as a proxy.

The third variable "LogType" defines the type of log you are going to send. This can be the same as existing log entries or a completly new one. This type of log defines if a new table or an existing table is created or used to store the logs.

```
$customerId = ""
$sharedKey = ""
$LogType = ""


Function Build-Signature ($customerId, $sharedKey, $date, $contentLength, $method, $contentType,
$resource){
    $xHeaders = "x-ms-date:" + $date
    $stringToHash = $method + "`n" + $contentLength + "`n" + $contentType + "`n" + $xHeaders + "`n" + $resource

    $bytesToHash = [Text.Encoding]::UTF8.GetBytes($stringToHash)
    $keyBytes = [Convert]::FromBase64String($sharedKey)

    $sha256 = New-Object System.Security.Cryptography.HMACSHA256
    $sha256.Key = $keyBytes
    $calculatedHash = $sha256.ComputeHash($bytesToHash)
    $encodedHash = [Convert]::ToBase64String($calculatedHash)
    $authorization = 'SharedKey {0}:{1}' -f $customerId,$encodedHash
    return $authorization
}

Function Post-LogAnalyticsData ($customerId, $sharedKey, $body, $logType){
    $method = "POST"
    $contentType = "application/json"
    $resource = "/api/logs"
    $rfc1123date = ([DateTime]::UtcNow).ToString("r")
    $contentLength = $body.Length
    $signature = Build-Signature -customerId $customerId -sharedKey $sharedKey -date $rfc1123date -contentLength $contentLength -method $method -contentType $contentType -resource $resource

    $uri = "https://" + $customerId + ".ods.opinsights.azure.com" + $resource + "?api-version=2016-04-01"

    $headers = @{
        "Authorization" = $signature;
        "Log-Type" = $logType;
        "x-ms-date" = $rfc1123date;
    }

    $response = Invoke-WebRequest -Uri $uri -Method $method -ContentType $contentType -Headers $headers -Body $body -UseBasicParsing
    return $response.StatusCode
}

$Properties = [Ordered] @{
    "ComputerName"     = $env:computername
    "User"              = $env:Username
}

$CustomLogs = New-Object -TypeName "PSObject" -Property $Properties | ConvertTo-JSON -Depth 10



#Submit the data to the API endpoint
$params = @{
    CustomerId = $customerId
    SharedKey  = $sharedKey
    Body       = ([System.Text.Encoding]::UTF8.GetBytes($CustomLogs))
    LogType    = $LogType
}
$LogResponse = Post-LogAnalyticsData @params
$LogResponse
```

The variable "LogResponse" is in a successful execution filled with the value 200.

Use Case

A possible use case is for example is to write an Azure Function to provide a custom Rest API to write logs to a function. This has multiple advantages over writing directly to the Log Analytics Workspace. Everything from less code to credential leaks will be provided with a simple REST Call to the Azure Function. More on this in the chapter "Azure Function".

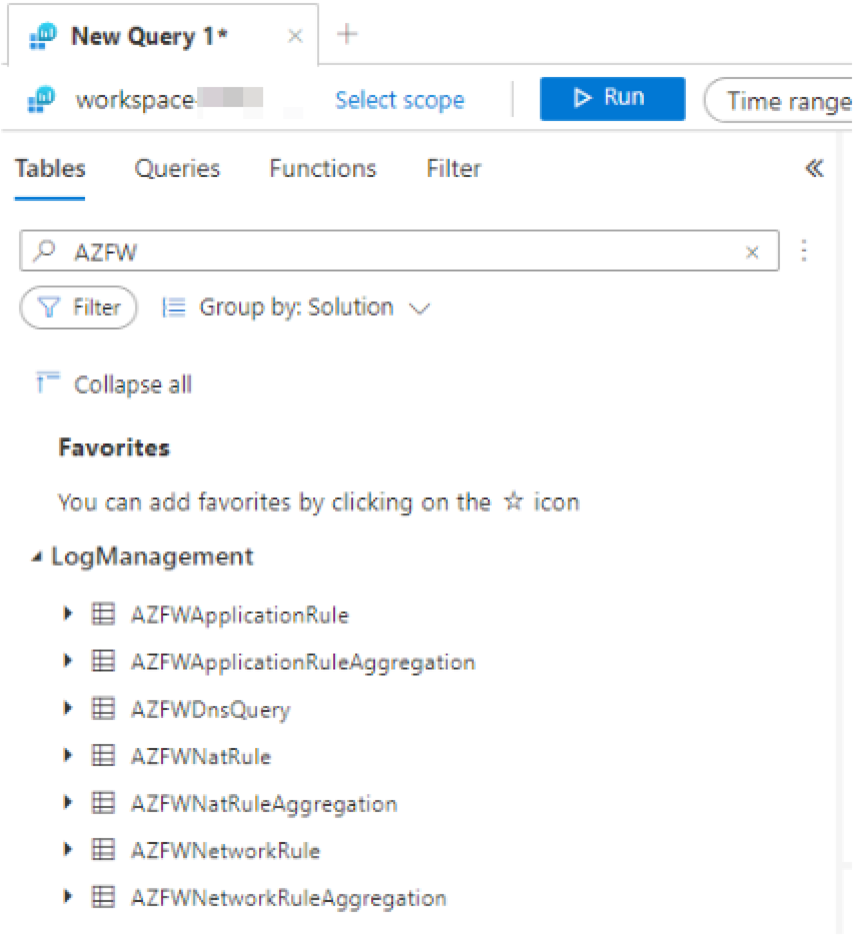
Query Azure Firewall Logs

Azure Firewall Logs can be stored in an Azure Log Analytics Workspace. This workspace then contains all status logs along with permitted and denied connections. So, to find out if a connection is wrongly blocked or to make a specific firewall request, we can use these logs to give us insights.

Find log tables

First of all you have to select the scope on which you want to search for the logs. You can choose the Log Analytics scope with "Select scope".

Azure Firewalls save logs to different tables. To find all the different log table you can search in the tables pane for "AZFW". These Tables contain the corresponding log data.



AZFW

Favorites

You can add favorites by clicking on the ☆ icon

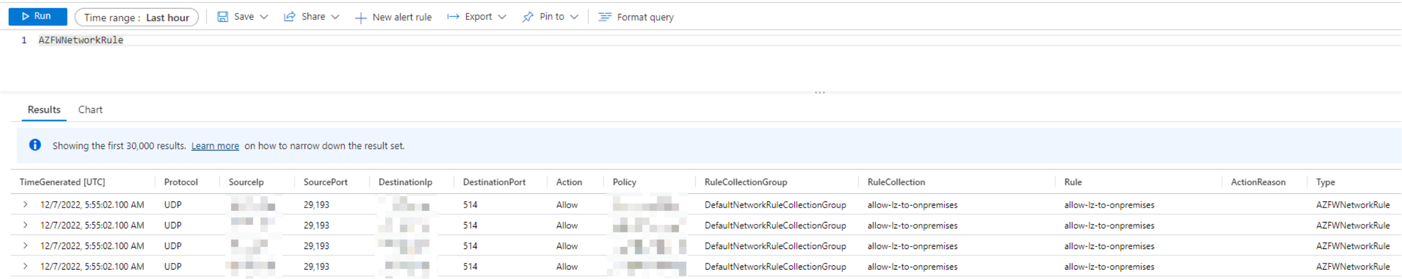
LogManagement

- AZFWApplicationRule
- AZFWApplicationRuleAggregation
- AZFWDnsQuery
- AZFWNatRule
- AZFWNatRuleAggregation
- AZFWNetworkRule
- AZFWNetworkRuleAggregation

Azure Firewall rule logs are stored within the "AZFWNetworkRule" table.

View whole table content

To view the whole table content you can write the name of the table into the KQL (Kusto Query Language) section. In this case "AZFWNetworkRule" is enough to see all the permitted and denied connections.



```
AZFWNetworkRule
```

Showing the first 30,000 results. Learn more on how to narrow down the result set.

	TimeGenerated [UTC]	Protocol	SourceIp	SourcePort	DestinationIp	DestinationPort	Action	Policy	RuleCollectionGroup	RuleCollection	Rule	ActionReason	Type
>	12/7/2022, 5:55:02.100 AM	UDP	[redacted]	29,193	[redacted]	514	Allow	[redacted]	DefaultNetworkRuleCollectionGroup	allow-lz-to-onpremises	allow-lz-to-onpremises		AZFWNetworkRule
>	12/7/2022, 5:55:02.100 AM	UDP	[redacted]	29,193	[redacted]	514	Allow	[redacted]	DefaultNetworkRuleCollectionGroup	allow-lz-to-onpremises	allow-lz-to-onpremises		AZFWNetworkRule
>	12/7/2022, 5:55:02.100 AM	UDP	[redacted]	29,193	[redacted]	514	Allow	[redacted]	DefaultNetworkRuleCollectionGroup	allow-lz-to-onpremises	allow-lz-to-onpremises		AZFWNetworkRule
>	12/7/2022, 5:55:02.100 AM	UDP	[redacted]	29,193	[redacted]	514	Allow	[redacted]	DefaultNetworkRuleCollectionGroup	allow-lz-to-onpremises	allow-lz-to-onpremises		AZFWNetworkRule

Filter logs after IP address

Most of the time we want to filter for specific addresses. These Firwall logs can be queried with the powerful KQL language. This language helps to explore data and discover patterns, identify anomalies and outliers, create statistical modeling, and more. The query uses schema entities that are organized in a hierarchy similar to SQL's: databases, tables, and columns.

Filter for source IPs

AZFWNetworkRule
| where SourceIp == "<yoursourceipadress>"

Filter for destination IPs

AZFWNetworkRule
| where DestinationIp == "<yourdestinationipadress>"

Create Workbooks with KQL queries

Introduction

In today's data-driven world, Azure provides a wealth of data waiting to be analyzed. Azure Log Analytics, powered by Kusto Query Language (KQL), enables users to explore and visualize data effectively. Let's dive into how to leverage KQL-Based Workbooks in Azure Log Analytics to gain valuable insights through compelling graphs and diagrams.

Implementing KQL-Based Workbooks

1. **Setting Up Azure Log Analytics:** Ensure you have an active Azure subscription and access to Azure Log Analytics. Create a Log Analytics workspace from the Azure portal and connect your data sources, such as Azure Monitor, Application Insights, or custom logs.
2. **Creating a Workbook:** Navigate to your Log Analytics workspace in the Azure portal, select "Workbooks" from the left-hand menu, and create a new workbook. You can also choose from existing templates to accelerate your analysis.
3. **Writing KQL Queries:** In the workbook, use Kusto Query Language (KQL) to craft queries that retrieve data from your Log Analytics workspace. KQL allows you to filter, aggregate, and transform data efficiently.
4. **Visualization with Graphs:** After querying the data, enhance your insights by creating visualizations with graphs. Choose from various graph types such as line charts, bar charts, pie charts, or area charts, depending on the data and the story you want to convey.
5. **Diagram Visualization:** Apart from traditional graphs, KQL-Based Workbooks enable you to create dynamic and interactive diagrams. Utilize diagrams like topology maps, network diagrams, or flowcharts to depict the relationships and dependencies within your data.
6. **Interactive Controls:** Enhance user experience by adding interactive controls like drop-down menus, time range selectors, or variable inputs. These controls allow users to customize and explore data on their terms.
7. **Sharing and Collaboration:** Share your workbook with relevant stakeholders, empowering them to view and interact with the data-driven insights. Collaborate with teams by granting them access to workbooks or exporting them for offline review.
8. **Automate with Scheduled Queries:** Automate data analysis by scheduling queries to run at regular intervals. This ensures that your workbooks stay up-to-date with the latest

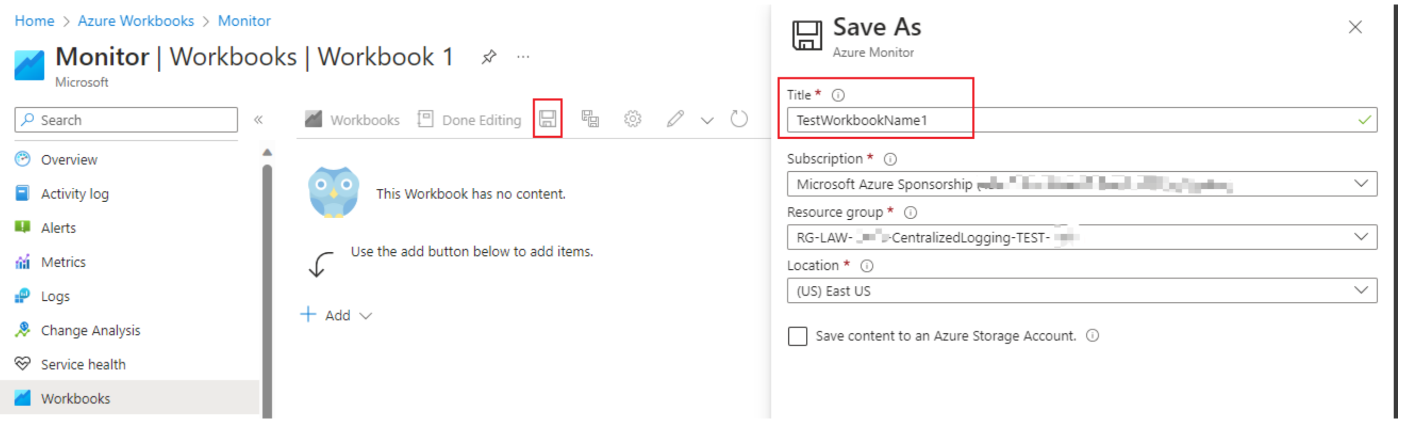
data without manual intervention.

Use of variables & dynamic content in Azure Workbooks

Within large Azure workbooks (dashboards), we want to be dynamic and allow users to specify their own parameters. This enables a more precise evaluation based on e.g. time period, resources and much more.

Create Azure Workbook

Search in Azure Portal for "Azure Workbooks" and create a new Workbook. It is recommended to save the workbook beforehand. For this step click on the save icon and set the name you want. Then choose the subscription and resource group parameters. in the end you have to specify a location.



This Workbook has no content.

Use the add button below to add items.

Title * TestWorkbookName1

Subscription * Microsoft Azure Sponsorship [redacted]

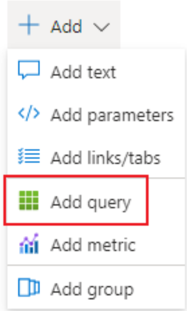
Resource group * RG-LAW-[redacted]-CentralizedLogging-TEST-[redacted]

Location * (US) East US

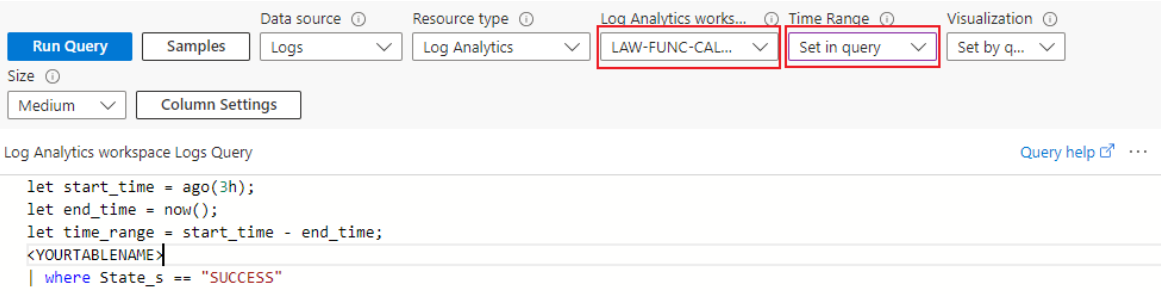
☐ Save content to an Azure Storage Account.

Create first data visualization

Inside the newly created we can start the first query. Click "Add" and choose "Add query".



Inside this query you have to specify the workspace as this is the source for the queries (your log analytics workspace or storage account of choice). Set the parameter "Time Range" to the value "Set in query" thus we will set this dynamically using Workbook parameters.



Data source: Logs
Resource type: Log Analytics
Log Analytics works...: LAW-FUNC-CAL...
Time Range: Set in query
Visualization: Set by q...
Size: Medium

Log Analytics workspace Logs Query

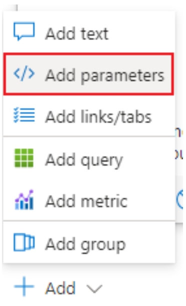
```
let start_time = ago(3h);
let end_time = now();
let time_range = start_time - end_time;
<YOURTABLENAME>
| where State_s == "SUCCESS"
```

Write your log query in KQL so that the output meets your expectations. AI Chatbots can be a big help for writing queries.

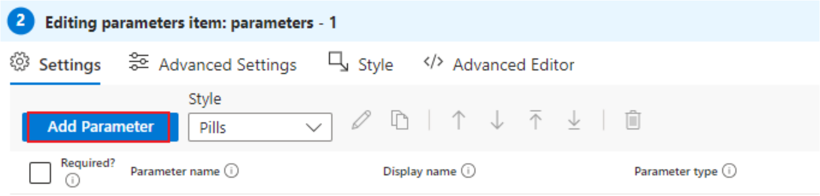
Add dynamic parameters

To display charts and tables based on the parameters we set, we can use the so-called "Parameter" functionality of Azure Workbooks.

First we have to create a parameter set. Click "Add" and choose "Add parameters":

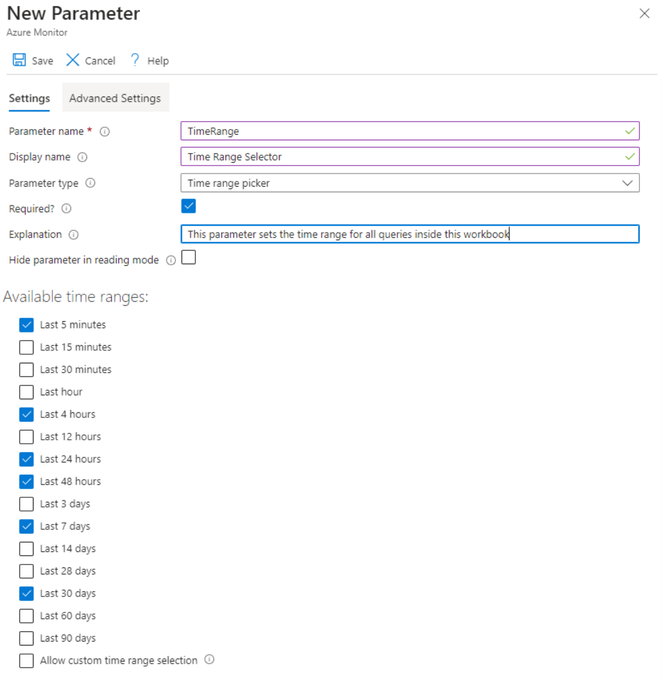


Click "Add Parameter":



Style: Pills

Fill out the form based on your needs:



Parameter name *	TimeRange
Display name	Time Range Selector
Parameter type	Time range picker
Required?	☑
Explanation	This parameter sets the time range for all queries inside this workbook
Hide parameter in reading mode	☐

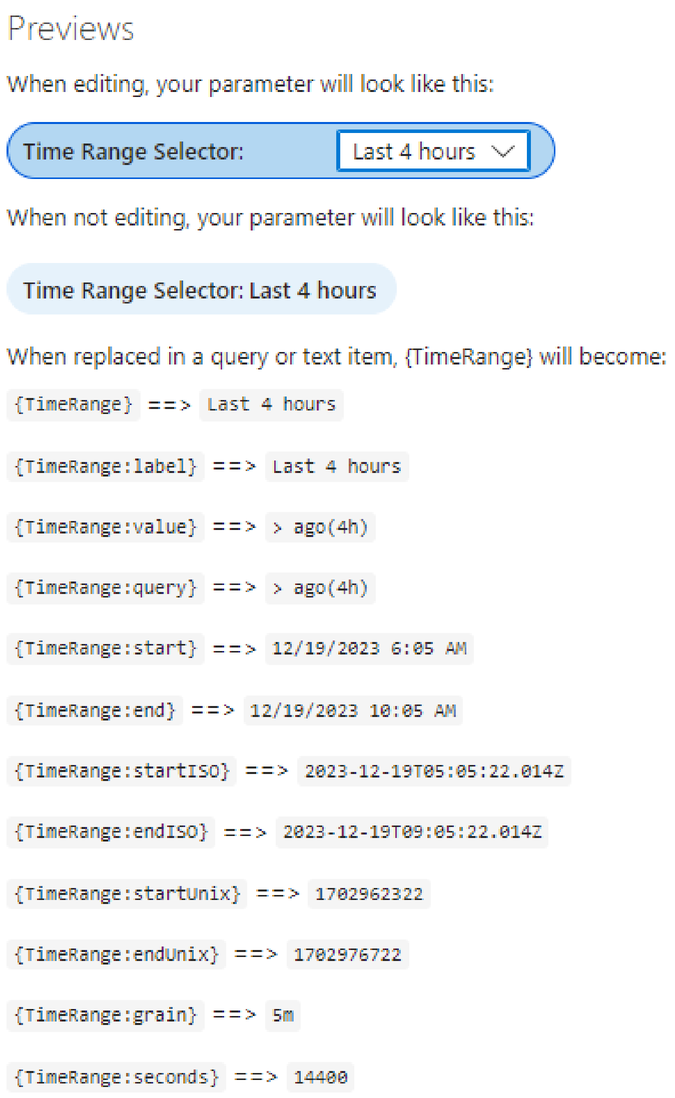
Available time ranges:

- ☑ Last 5 minutes
- ☐ Last 15 minutes
- ☐ Last 30 minutes
- ☐ Last hour
- ☑ Last 4 hours
- ☐ Last 12 hours
- ☑ Last 24 hours
- ☑ Last 48 hours
- ☐ Last 3 days
- ☑ Last 7 days
- ☐ Last 14 days
- ☐ Last 28 days
- ☑ Last 30 days
- ☐ Last 60 days
- ☐ Last 90 days
- ☐ Allow custom time range selection

Use of dynamic parameters

After we have created a dynamic parameter, we can use the values inside our KQL queries. The use is different based on the selected data type of the parameter.

Using the "Previews" pane on the editing page of the parameter we can select values and see their according value.



Previews

When editing, your parameter will look like this:

Time Range Selector: Last 4 hours

When not editing, your parameter will look like this:

Time Range Selector: Last 4 hours

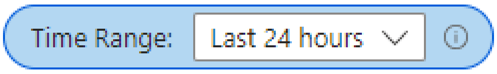
When replaced in a query or text item, {TimeRange} will become:

```
{TimeRange} ==> Last 4 hours

{TimeRange:label} ==> Last 4 hours

{TimeRange:value} ==> > ago(4h)

{TimeRange:query} ==> > ago(4h)

{TimeRange:start} ==> 12/19/2023 6:05 AM

{TimeRange:end} ==> 12/19/2023 10:05 AM

{TimeRange:startISO} ==> 2023-12-19T05:05:22.014Z

{TimeRange:endISO} ==> 2023-12-19T09:05:22.014Z

{TimeRange:startUnix} ==> 1702962322

{TimeRange:endUnix} ==> 1702976722

{TimeRange:grain} ==> 5m

{TimeRange:seconds} ==> 14400
```

So we can work with these variables inside our queries and update our query from before. This enables us to display the events based on the selected value of the parameter:

```
let start_time = {TimeRange:value};
let end_time = now();
let time_range = start_time - end_time;
<YOURTABLENAME>
| where State_s == "SUCCESS"
```

This will replace {TimeRange:value} with the value "ago(24h)" when the selection is the following:



Time Range: Last 24 hours

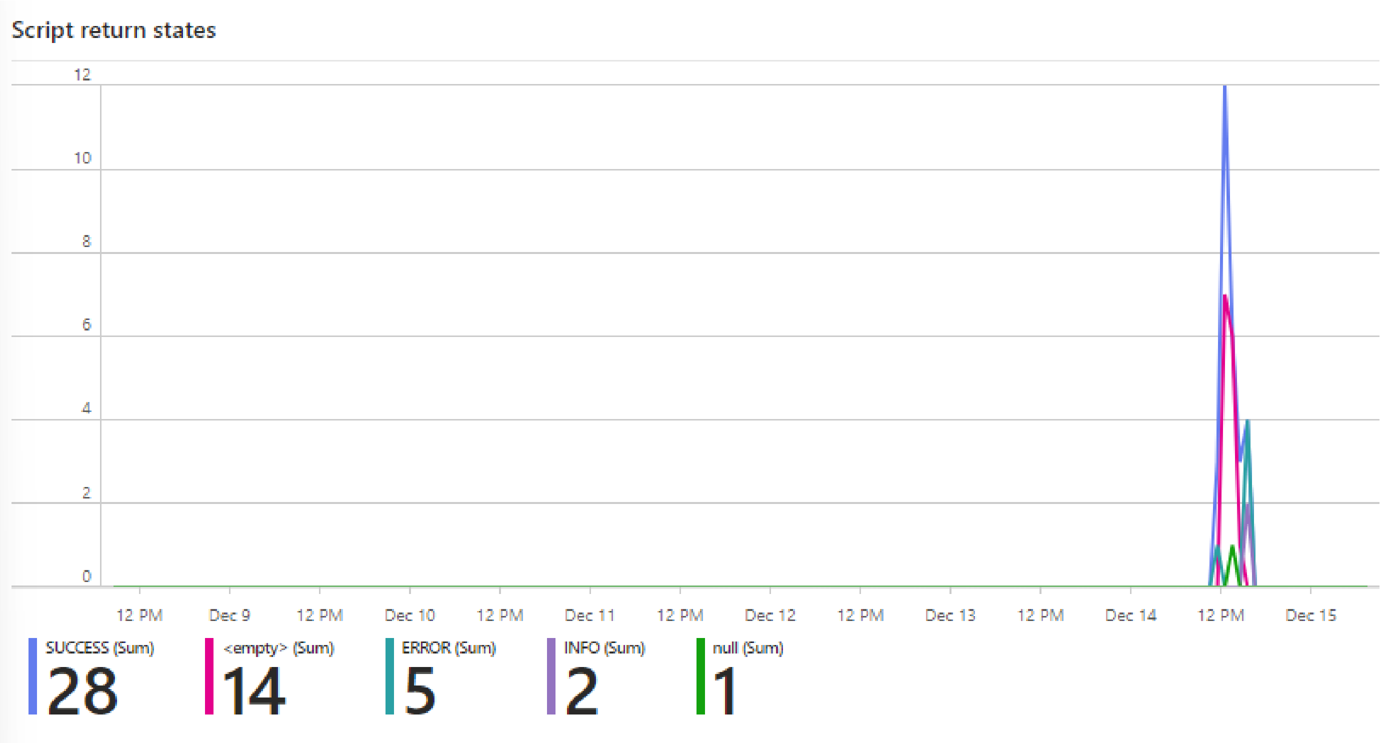
Create log health dashboard using time based data

This article is about how to create a dashboard using log data. We want to analyze relevant time-sensitive data in a dynamic and visually meaningful way.

Use case

An example application is when an application or script feeds data from various endpoints or microservices via a central logging solution and this data is then to be visualized. You can find more information about a simple and scalable solution for central log management here: Centralize log collect... | LNC DOCS (lucanoahcaprez.ch)

This example graph displays values of a PowerShell script that runs at the log in of a user. The script then has statuses, which are then written to an Azure Log Analytics Workspace. This diagram can now be used to see at what time which status was created.



Behind this view there is the following KQL query:

```
let start_time = {TimeRange:start};
let end_time = now();
let time_range = start_time - end_time;
<yourlogtable>
| where TimeGenerated >= start_time and TimeGenerated <= end_time
| make-series count() default=0 on TimeGenerated from start_time to end_time step 1h by State_s
| render timechart
```

This query uses the value of the variable "{TimeRange:start}" for dynamic change based on the selection. More on this dynamic strategy is described here: Use of variables & dyn... | LNC DOCS (lucanoahcaprez.ch)