NIC Cloud Connect

Oslo Spektrum

November 7 - 9

nic Cloud Connect

Morten Knudsen

LogAnalytics v2: Automating the Transition to Log Ingestion API & Data Collection Rules

Microsoft Azure

Mail Sept 19, 2023 at 10:13

We recently sent you an email regarding the retirement of Data Collector API. Our previous communication included an incorrect date in the title: 31 March 2026. This email contains the correct retirement date of 14 September 2026. We apologize for any confusion this error may have caused.

Data Collector API will be retired on 14 September 2026 – transition to DCR-based custom log ingestion.

You're receiving this email because you currently use the Data Collector API for ingestion of custom logs to Azure Monitor logs.

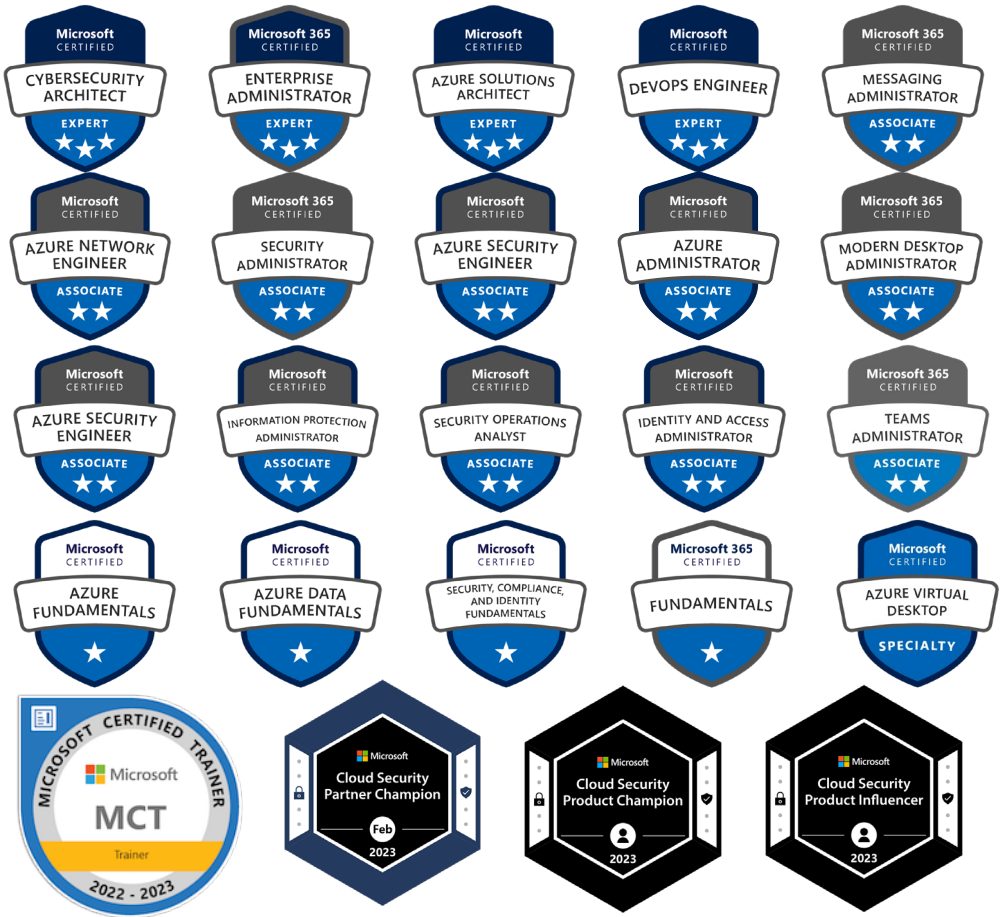
On 14 September 2026, we'll retire Data Collector API for ingesting custom logs to Azure Monitor logs. Before that date, you'll need to transition to the Data Collection Rules based log ingestion API that provides all the functionality of the Data Collector API, as well as new ones, including:

- Secure token-based authentication
- Full control over the shape of the destination table
- Ability to filter and transform data during ingestion

Required action

To avoid ingestion errors and data loss, transition to the DCR-based Log Ingestion API before 14 September 2026. Data Collector API endpoints will become unavailable after that date.

Speaker – Morten Knudsen



About | Morten Knudsen

Microsoft MVP Security & Azure Hybrid MVP

Microsoft Certified Trainer

Cloud & Security Architect

Microsoft Sentinel Black Belt

Microsoft Defender Black Belt

Microsoft Cloud Security Influencer

Microsoft Sentinel Influencer

Microsoft Defender for Cloud Influencer

MVP Microsoft® Most Valuable Professional

Award Categories
Security

First year awarded:
2023

Number of MVP Awards:
1

WHY ingest custom log data ?

Examples of Use Cases

- Are we in control? (Bitlocker, Local admins, Windows Update, etc)
- Detect local printers – replace with large multi - function printers (prepare business case)
- PC replacement – yearly budget – Lenovo / Dell warranty lookup
- SQL monitoring
- Collection of TSM Spectrum log - files
- Monitoring of Production environment

Are We In Control - *Operational & Security Inventory of Clients*

ClientInspector
FREE- community

CLIENT KPI STATUS | MANAGED D...

Shared dashboard

Create | Upload | Refresh | Full screen | Edit | Manage sharing | Export | Clone | Assign tags | Delete | Feedback

Auto refresh : **Off** | UTC Time : **Past 24 hours**

OTHER PRIMARY ANTIVIRUS (SECURITY CENTER) | CLIENTS | C...
log-platform-management-client-p (@ 3/19 3:11 PM)

3

INCOMPLIANT BITLOCKER | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

465

NO RESTART MORE THAN 7 DAYS DAY | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

8

LAST GROUP POLICY REFRESH MORE THAN 7 DAYS AGO | CLIE...
log-platform-management-client-p (@ 3/19 3:11 PM)

0

INCOMPLIANT LAPS | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

15

INCOMPLIANT DEFENDER AV | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

73

UNSUPPORTED WINDOWS OS BUILD, MDE | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

34

SOON UNSUPPORTED WINDOWS OS BUILD, MDE | CLIENTS | ...
log-platform-management-client-p (@ 3/19 3:11 PM)

1

DEVICES WITH NON-STANDARD MEMBERS LOCAL ADMINS G...
log-platform-management-client-p (@ 3/19 3:11 PM)

2,332

DEVICES WITH HIGH SEVERITY VULNERABILITIES >9 | CLIENTS ...
log-platform-management-client-p (@ 3/19 3:11 PM)

7

UPDATING O365 OFFICE NOT ENABLED | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

572

INCOMPLIANT UPDATE CHANNEL O365 OFFICE | CLIENTS | CO...
log-platform-management-client-p (@ 3/19 3:11 PM)

0

LEGACY OFFICE | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

0

STANDALONE OFFICE 2016 | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

0

INCOMPLIANT WINDOWS FIREWALL | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

1

LAST WINDOWS UPDATE MORE THAN 40 DAYS AGO | CLIENT...
log-platform-management-client-p (@ 3/19 3:11 PM)

134

CLIENTS WITH PENDING UPDATES OLDER THAN 40 DAYS | CLI...
log-platform-management-client-p (@ 3/19 3:11 PM)

33

CLIENTS WITH PENDING UPDATES OLDER THAN 40 DAYS, NO ...
log-platform-management-client-p (@ 3/19 3:11 PM)

25

UNEXPECTED SHUTDOWNS | CLIENTS | COUNT
log-platform-management-client-p (@ 3/19 3:11 PM)

819

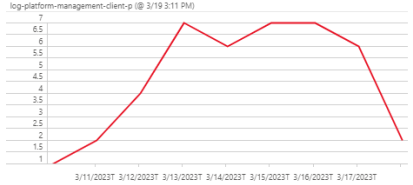
DEVICES WITH COLLECTION ISSUES (FIX WMI) | CLIENTS | COU...
log-platform-management-client-p (@ 3/19 3:11 PM)

14

DEVICES WITH CRITICAL OR HIGH RISKY BROWSER EXT | CLIE...
log-platform-management-client-p (@ 3/19 3:11 PM)

552

DEVICES WITH HIGH SEVERITY VULNERABILITY > 9 | CLIENTS | TIME 7 DAYS
log-platform-management-client-p (@ 3/19 3:11 PM)



total (Avg)

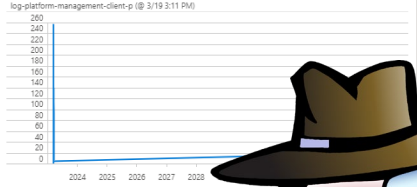
4.67

NO RESTART MORE THAN 7 DAYS DAY | CLIENTS | TIME 1 MONTH COLLECTI...
log-platform-management-client-p (@ 3/19 3:11 PM)

total (Avg)

1

UNEXPECTED SHUTDOWNS | CLIENTS | TIME 7 DAYS
log-platform-management-client-p (@ 3/19 3:11 PM)



total (Avg)

86

DEVICES WITH EXPIRED CERTIFICATES, PERSONAL, EXCL. MEM...
log-platform-management-client-p (@ 3/19 3:11 PM)

2,307

DEVICES WITH SOON TO EXPIRE CERTIFICATES (60 DAYS), PER...
log-platform-management-client-p (@ 3/19 3:11 PM)

316

WINDOWS UPDATE | CLIENTS | M...

Shared dashboard

Auto refresh : **Off** | UTC Time : **Past 24 hours**

PENDING UPDATES OLDER THAN 40 DAYS | CLIENTS | COUNT

log-management-client-demo1-t (@ 3/23 7:19 PM)

5

PENDING UPDATES, NO DRV, OLDER THAN 40 DAYS | CLIENTS | COUNT

log-management-client-demo1-t (@ 3/23 7:19 PM)

3

LAST WINDOWS UPDATE MORE THAN 40 DAYS AGO | CLIENTS...

log-management-client-demo1-t (@ 3/23 7:19 PM)

0

PENDING UPDATES OLDER THAN 40 DAYS | CLIENTS | SUM BY CLIENTS

log-management-client-demo1-t (@ 3/23 7:19 PM)

Search

Computer	total
STRV-ACW-LT-01	7
STRV-MEW-DT-02	6
STRV-CEW-LT-03	4
STRV-ACW-DT-01	2
STRV-MOK-LT-02	1

PENDING UPDATES, NO DRV, OLDER THAN 40 DAYS | CLIENTS | SUM BY CLIE...

log-management-client-demo1-t (@ 3/23 7:19 PM)

Search

Computer	total
STRV-ACW-LT-01	3
STRV-MEW-DT-02	2
STRV-CEW-LT-03	1

UPDATE SOURCE MICROSOFT UPDATE | CLIENTS | COUNT

log-management-client-demo1-t (@ 3/23 7:19 PM)

8

UPDATE SOURCE WINDOWS UPDATE | CLIENTS | LIST

log-management-client-demo1-t (@ 3/23 7:19 PM)

0

PENDING UPDATES OLDER THAN 40 DAYS | CLIENTS | LIST

log-management-client-demo1-t (@ 3/23 7:19 PM)

Search

Computer	UserLoggedOn	Title_	UpdateClassification	UpdateKBPublished	LastDeploymentChangeTime	CollectionTime	TimeGenerated	AutoDownload	AutoSelection	AutoSel
STRV-ACW-DT-01	2LINKIT\Anne	Lenovo Ltd. - Firmware - 1.0.0.110	Drivers	11/25/2022, 12:00:00 AM	11/25/2022, 12:00:00 AM	3/21/2023, 1:13:04 PM	3/21/2023, 1:13:08 PM	2	1	false
STRV-ACW-DT-01	2LINKIT\Anne	Hewlett-Packard - Imaging - Null Print - HP Photosmart 6...	Drivers	9/2/2018, 12:00:00 AM	9/2/2018, 12:00:00 AM	3/23/2023, 6:52:01 PM	3/23/2023, 6:52:03 PM	2	1	false
STRV-ACW-LT-01	2LINKIT\Anne	Apple, Inc. - USBDevice - 486.0.0.0	Drivers	11/20/2020, 12:00:00 AM	11/20/2020, 12:00:00 AM	3/22/2023, 12:15:17 PM	3/22/2023, 12:15:26 PM	2	1	false
STRV-ACW-LT-01	2LINKIT\Anne	Sikkerhedsopdatering 08-2022 til Windows 11 22H2 for x...	Security Updates	11/10/2022, 12:00:00 AM	11/10/2022, 12:00:00 AM	3/22/2023, 12:15:17 PM	3/22/2023, 12:15:26 PM	2	1	true
STRV-ACW-LT-01	2LINKIT\Anne	Microsoft Azure Information Protection Unified Labeling ...	Updates	11/28/2022, 12:00:00 AM	11/28/2022, 12:00:00 AM	3/22/2023, 12:15:17 PM	3/22/2023, 12:15:26 PM	0	0	false
STRV-ACW-LT-01	2LINKIT\Anne	Intel Corporation - System - 2.14.101.1	Drivers	11/1/2022, 12:00:00 AM	11/1/2022, 12:00:00 AM	3/22/2023, 12:15:17 PM	3/22/2023, 12:15:26 PM	2	1	false
STRV-ACW-LT-01	2LINKIT\Anne	Intel Corporation - SoftwareComponent - 2.17.100.2	Drivers	2/11/2023, 12:00:00 AM	2/11/2023, 12:00:00 AM	3/22/2023, 12:15:17 PM	3/22/2023, 12:15:26 PM	2	1	false
STRV-ACW-LT-01	2LINKIT\Anne	Lenovo - System - 1.2.0.11	Drivers	2/7/2023, 12:00:00 AM	2/7/2023, 12:00:00 AM	3/22/2023, 12:15:17 PM	3/22/2023, 12:15:26 PM	2	1	false
STRV-ACW-LT-01	2LINKIT\Anne	2023-01 Opdatering til Windows 11 Version 22H2 til x64-...	Critical Updates	1/18/2023, 12:00:00 AM	1/18/2023, 12:00:00 AM	3/22/2023, 12:15:17 PM	3/22/2023, 12:15:26 PM	2	1	true
STRV-CEW-LT-03	2LINKIT\Caroline	Realtek - Extension - 10.44.0.2	Drivers	11/11/2021, 12:00:00 AM	11/11/2021, 12:00:00 AM	3/23/2023, 2:04:46 PM	3/23/2023, 2:04:48 PM	2	1	false
STRV-CEW-LT-03	2LINKIT\Caroline	Lenovo - System - 1.2.0.11	Drivers	2/7/2023, 12:00:00 AM	2/7/2023, 12:00:00 AM	3/23/2023, 2:04:46 PM	3/23/2023, 2:04:48 PM	2	1	false

nic Cloud Connect

LOCAL ADMINS | CLIENTS | MANA...
Shared dashboard

Create | Upload | Refresh | Full screen | Edit | Manage sharing

Auto refresh : **Off** | UTC Time : **Past 24 hours**

DEVICES WITH NON-STANDARD MEMBERS LOCAL ADMINS G...
log-management-client-demo1-t (@ 3/23 7:21 PM)

8

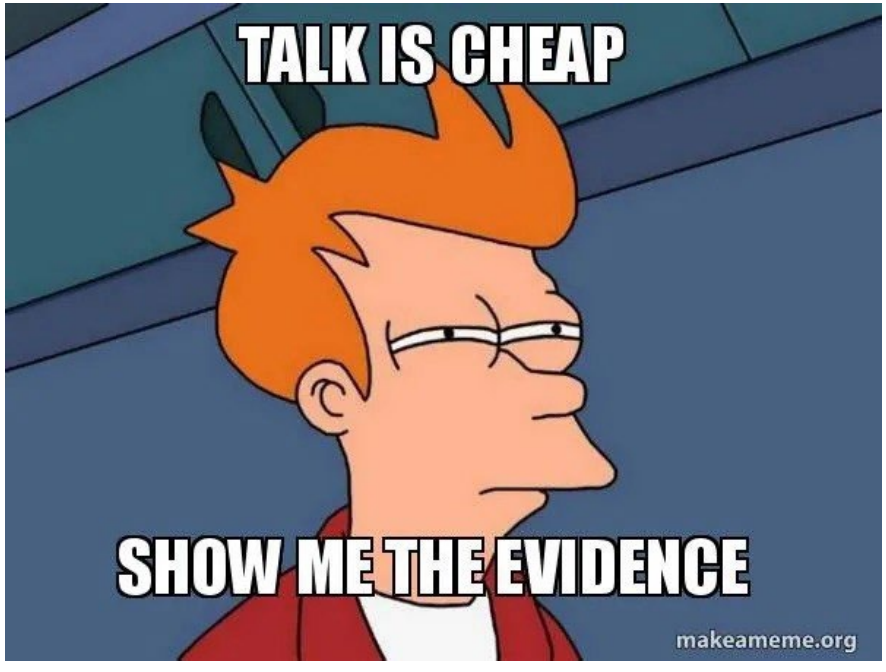
DEVICES WITH NON-STANDARD MEMBERS LOCAL ADMINS GROUP | CLIENT...
log-management-client-demo1-t (@ 3/23 7:21 PM)

Computer
STRV-ACW-DT-01
STRV-MOK-LT-02
HEIM-NEW-DT-01
STRV-MOK-DT-02
STRV-CEW-LT-03
STRV-MEW-LT-02
STRV-MEW-DT-02

MEMBERS LOCAL ADMINS GROUP | CLIENTS | GROUPED
log-management-client-demo1-t (@ 3/23 7:21 PM)

Search

Group	Computer
2LINKIT/Anne (2)	
WORKGROUP/STRV-MOK-LT-02/Morten Knudsen (1)	
2LINKIT/mok (2)	
2LINKIT/Niels.Waltorp (1)	
WORKGROUP/STRV-MOK-DT-02/mok (1)	
2LINKIT/Caroline (1)	
2LINKIT/magnus (2)	
WORKGROUP/STRV-ACW-LT-01/annew (1)	

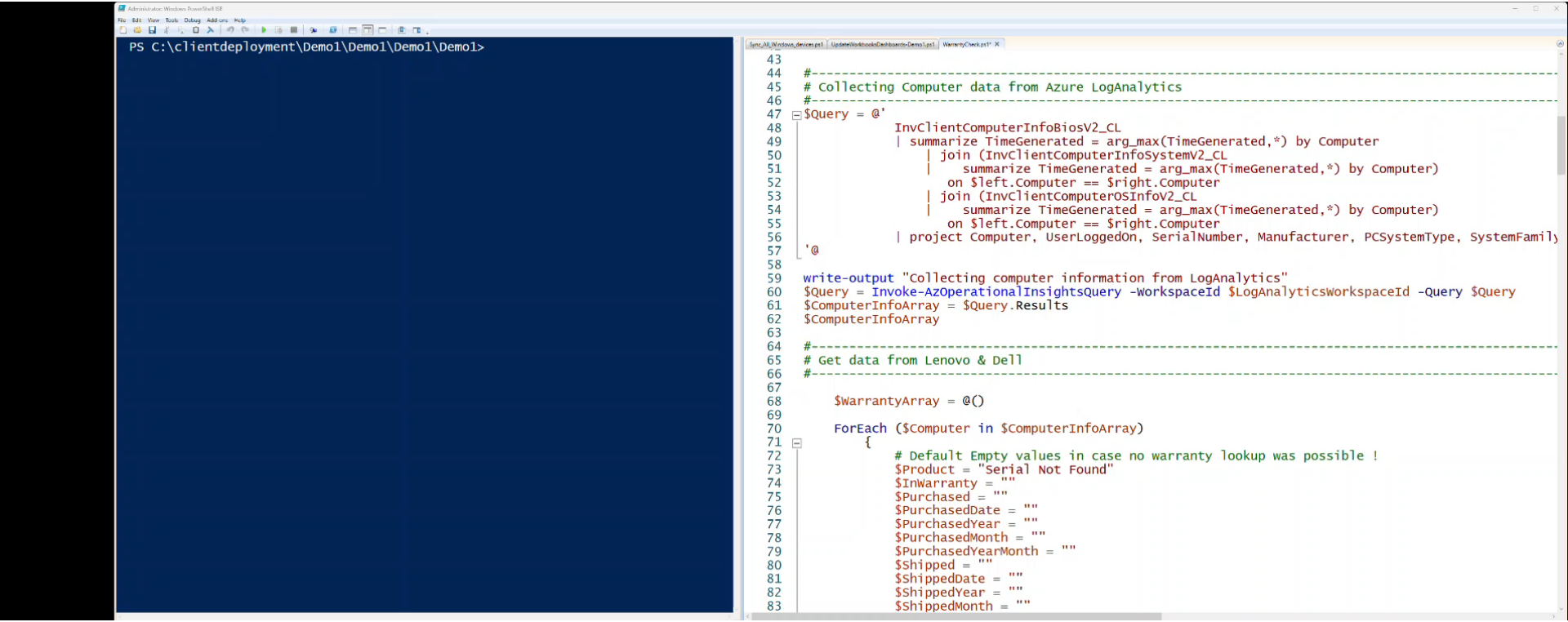


DKK 200 / USD 30
PER MONTH

500 devices /
daily inventory

PC replacement – Warranty Lookup

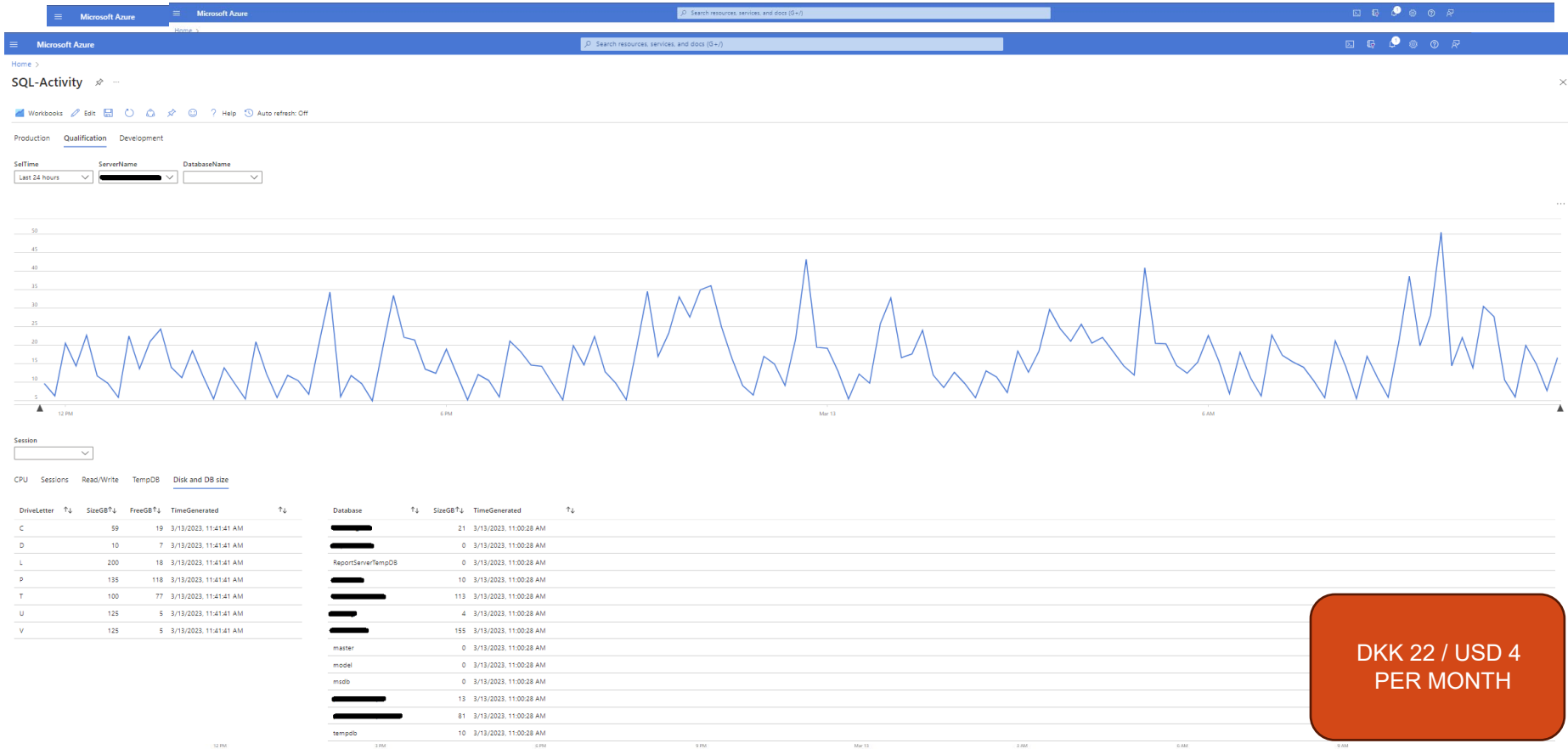
We re-use the inventory data from ClientInspector (serialnumber) to identify computers to replace




```
43
44  #-------------------------------------------------------------------------------------------------------------
45  # Collecting Computer data from Azure LogAnalytics
46  #-------------------------------------------------------------------------------------------------------------
47  $Query = @'
48              InvClientComputerInfoBiosV2_CL
49              | summarize TimeGenerated = arg_max(TimeGenerated,*) by Computer
50                  | join (InvClientComputerInfoSystemV2_CL
51                  |     summarize TimeGenerated = arg_max(TimeGenerated,*) by Computer)
52                    on $left.Computer == $right.Computer
53                  | join (InvClientComputerOSInfoV2_CL
54                  |     summarize TimeGenerated = arg_max(TimeGenerated,*) by Computer)
55                    on $left.Computer == $right.Computer
56              | project Computer, UserLoggedOn, SerialNumber, Manufacturer, PCSystemType, SystemFamily
57  '@
58
59  write-output "Collecting computer information from LogAnalytics"
60  $Query = Invoke-AzOperationalInsightsQuery -WorkspaceId $LogAnalyticsWorkspaceId -Query $Query
61  $ComputerInfoArray = $Query.Results
62  $ComputerInfoArray
63
64  #-------------------------------------------------------------------------------------------------------------
65  # Get data from Lenovo & Dell
66  #-------------------------------------------------------------------------------------------------------------
67
68      $WarrantyArray = @()
69
70      ForEach ($Computer in $ComputerInfoArray)
71          {
72              # Default Empty values in case no warranty lookup was possible !
73              $Product = "Serial Not Found"
74              $InWarranty = ""
75              $Purchased = ""
76              $PurchasedDate = ""
77              $PurchasedYear = ""
78              $PurchasedMonth = ""
79              $PurchasedYearMonth = ""
80              $Shipped = ""
81              $ShippedDate = ""
82              $ShippedYear = ""
83              $ShippedMonth = ""
```

Monitoring of 1200 SQL databases on 120 SQL -servers

Global Enterprise Danish Production company



DKK 22 / USD 4 PER MONTH

Backup Monitoring of IBM TSM Spectrum

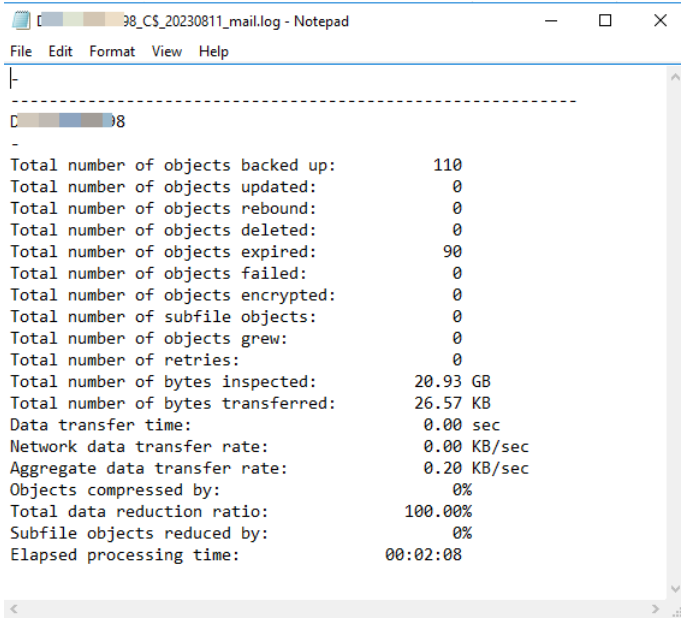
Central Log Collection

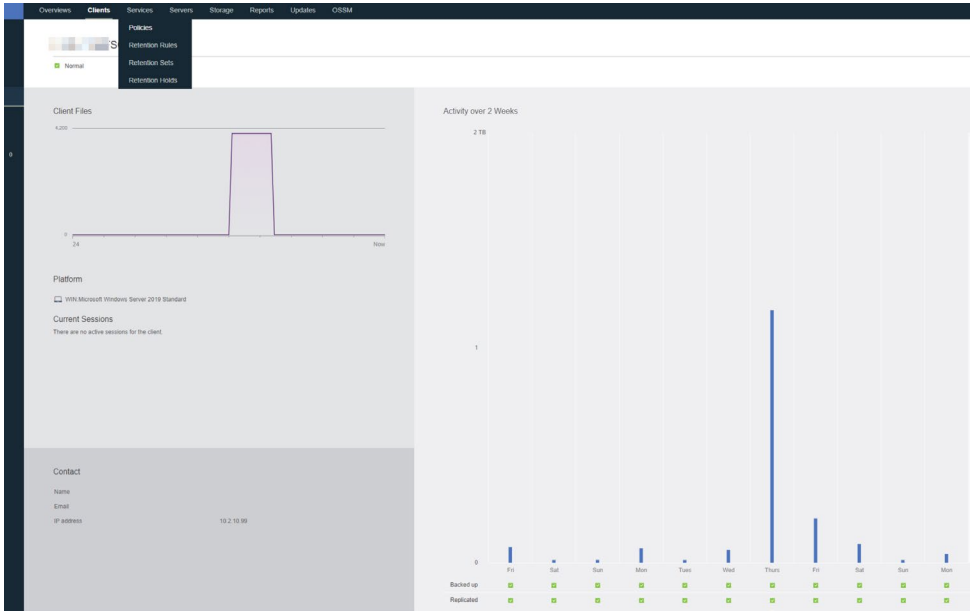
Large Enterprise German Production company

IBM Spectrum Protect aka. Tivoli TSM is used to perform backups especially on file -servers.

The backup agent writes a log file containing various information directly on the machine it runs.

Reporting capabilities of the TSM-Server are limited and do not allow detailed analysis.





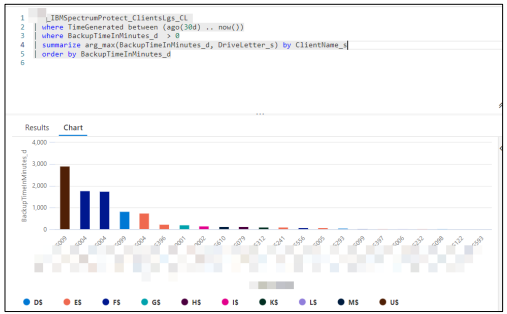
Backup Monitoring of IBM TSM Spectrum

Central Log Collection

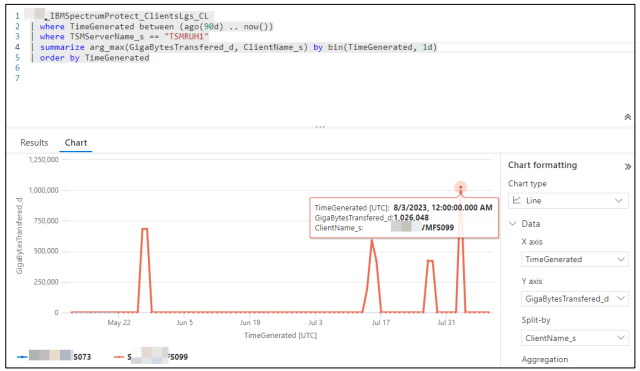
Large Enterprise German Production company

Building queries for Trending and Azure Monitor for Alerting

After having all information in the Log Analytics Workspace, KQL can be used for detailed analysis.:



Servers, sorted by time tacken for backup

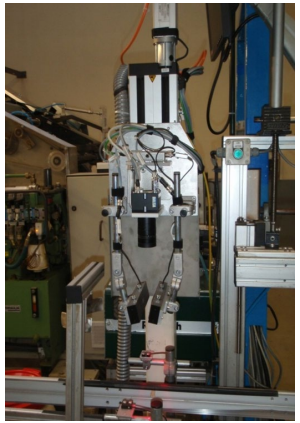


Plotting GBs transferred during backup

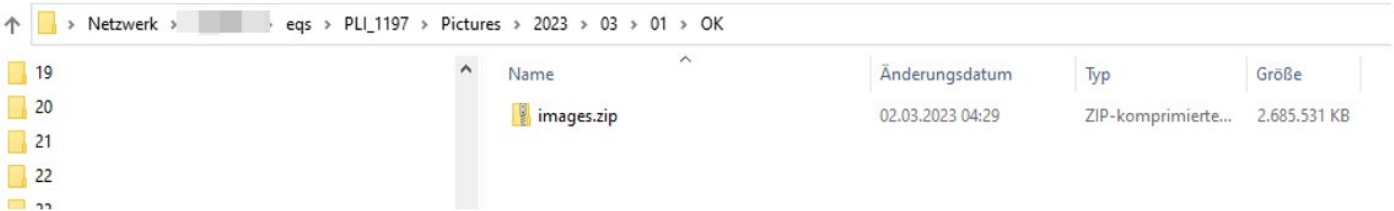
Servers with most failed objects during backup

Quality Inspection for Production (1/3)

Large Enterprise German Production company



XYZ is an image-processing-system. During one cycle about 10.000 pictures are taken and firstly stored on vendor proprietary hardware. A script runs regularly and copies the pictures to Fileserver on which data is than bundled to control the backup time.



Multiple production lines are equipped with those image-processing-systems. The script is re-used and only passes in different parameters to distinct between the vendor-hardware devices.

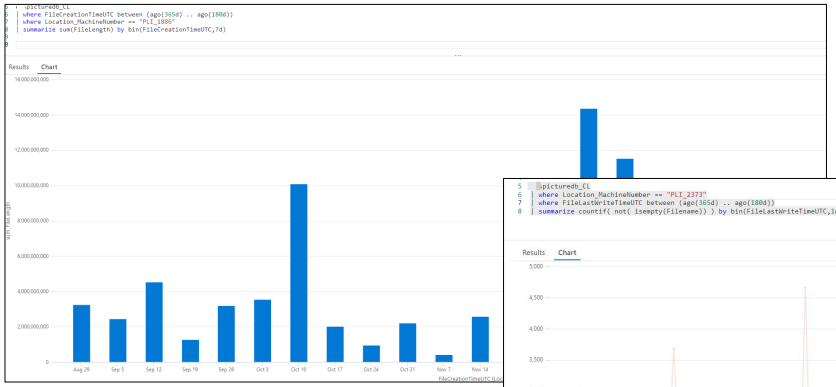
Unfortunately, it happens that pictures are missed. – From power outage on the camera, IP connectivity loss to seldom script errors. The script neither offers monitoring capabilities nor the tooling of the image-processing-system does.

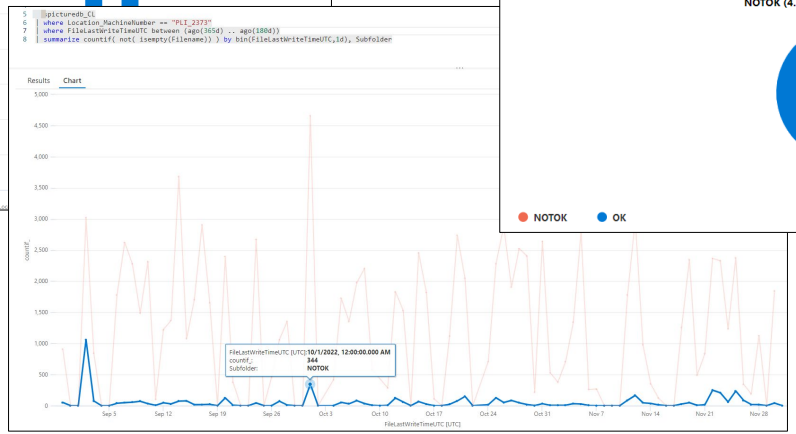
Quality Inspection for Production (3/3)

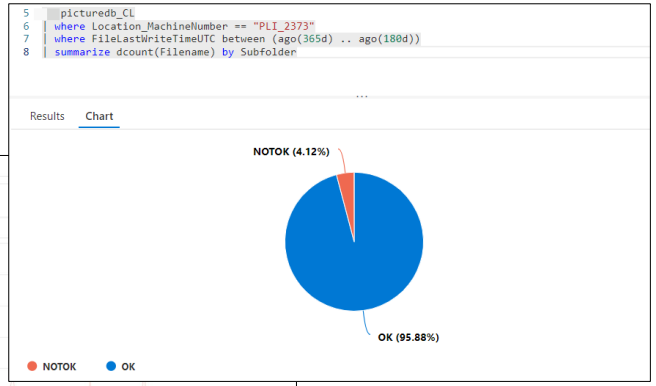
Large Enterprise German Production company

Building queries for Trending and Azure Monitor for Alerting

After having all information in the Log Analytics Workspace, KQL can be used for detailed analysis:







Topics we will cover …

- Comparing LogAnalytics V1 vs. V2
- Endpoint data collection using Azure Monitor Agent (AMA)
- Push & Pull data using Log Ingestion API, DCR & DCE
  - Challenges & options
  - Automate the transition using AzLogDcrIngestPS
  - Transformations: Data manipulations before & after sending data
- Migration strategies to Log Ingestion API

How many have tried Azure LogAnalytics before ?

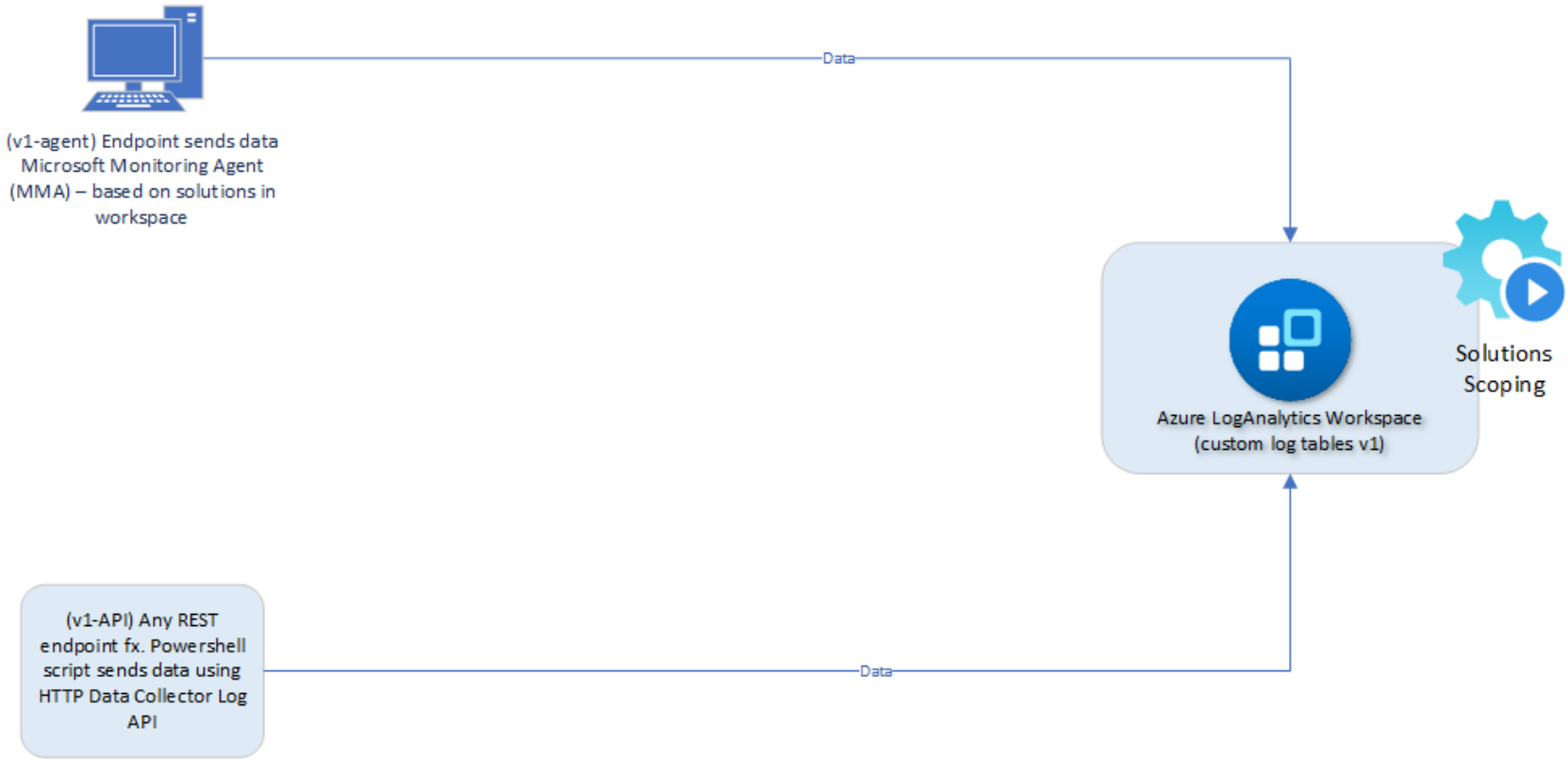
How many have used MMA agent to send data ?
…. AMA agent ?

How many have used HTTP Data Collector API before ?

How many have tried to play with Log Ingestion API ?

“V1” (legacy)

- LA Solutions
- MMA - Limited collection-capabilities – built- into agent
- Azure Monitor HTTP Data Collector API



Cloud Connect

(1) Endpoint sends data
Azure Monitor Agent (AMA)

Scope defined in Azure Data Collection Rules (DCR)

Performance
Event log
IIS logs
SNMP traps
Syslog
Change Tracking

Azure Pipeline
PMs, Seattle, USA

"V2"

- Azure Pipeline
- DCRs
- Transformation
- DCEs
- Custom tables (v2)

Azure Data Ingestion Pipeline
(shared service)

Azure Data Collection Rule (DCR)

DCR includes transformation to filter and manipulate data

transformed data

Azure LogAnalytics Workspace
(custom log tables v2)

(2) Any REST endpoint fx. Powershell script sends data using Log Ingest API

data

Sign In

Azure Data Collection Endpoint (DCE)

data

Azure DCR PMs &
Dev Lead, Seattle, USA

Azure LogAnalytics PMs, Israel

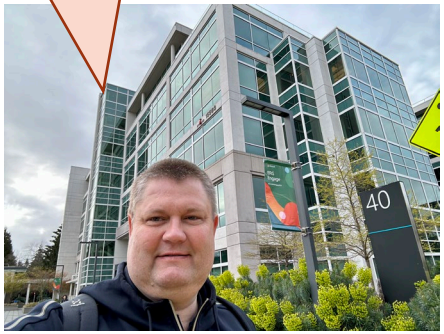
Azure Logging V2 – current data sources (endpoint-based)

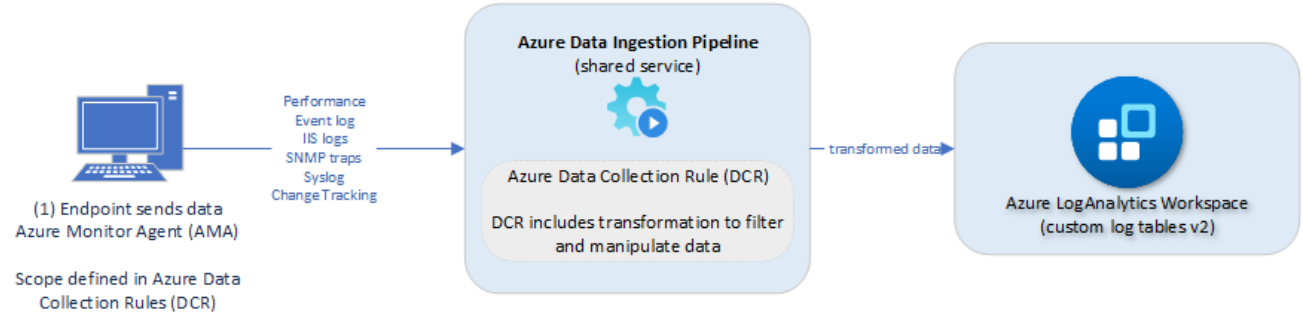
- Performance metrics & telemetry
- Performance Counters (SQL)
- Eventlogs
- IIS logs
- Windows Fw Logs
- Standard/Platform metrics/Telemetry (AzPaaS)
- Custom Logs (Ingestion API)
- Activity logs
- Platform logs (diagnostics)
- Custom metrics/telemetry (custom AzApp)
- Standard Logs (Ingestion API)
- Change Tracking
- Text logs
- SNMP Traps

Meet the Azure Monitor PMs (Seattle, USA)

I’m fan ☺

MS HQ - more really cool stuff is ‘in the oven being backed’ – coming soon ☺





- **remove data** before being sent into LogAnalytics (remove " noice " data/cost optimization, GDPR/compliance)
- **add data** before being sent into LogAnalytics
- **merge data** before being sent into LogAnalytics
- better data quality (**array data**), as array data is converted into dynamic - whereas the old MMA -method would convert array data into strings
- support to **send to other destinations**
- security is based on **Entra ID RBAC**
- **naming of data columns** are prettier, as they contain the actual name - and not for example ComputerName_s indicating it is a string value

How to get started with Azure Logging v2 on Azure Monitor Agent ??

12 new blogs + AzureLogLibrary.net (templates)

Blog: mortenknudsen.net

VMInsight - Linux - Performance_ServiceMap | Deploy DCR-rule to Azure

Deploy to Azure

Topic	Link
Understanding Azure logging capabilities in depth	https://mortenknudsen.net/?p=1433
How to do data transformation using Workspace transformation for legacy upload methods	https://mortenknudsen.net/?p=1436
Understanding the fundamentals of log -collection with Azure Monitor Agent & Azure Data Collection Rules	https://mortenknudsen.net/?p=1438
Understanding Azure Data Collection Endpoint	https://mortenknudsen.net/?p=1442
Collecting Security events using Azure Monitor Agent	https://mortenknudsen.net/?p=1444
Collecting System & Application events using Azure Monitor Agent	https://mortenknudsen.net/?p=1446
Collecting Performance data using Azure Monitor Agent, VMInsights and ServiceMap	https://mortenknudsen.net/?p=1449
Collecting IIS logs using Azure Monitor Agent	https://mortenknudsen.net/?p=1451
Collecting text logs using Azure Monitor Agent	https://mortenknudsen.net/?p=1453
Collecting Syslogs using Azure Monitor Agent	https://mortenknudsen.net/?p=1455
Collecting CEFSyslogs using Azure Monitor Agent	https://mortenknudsen.net/?p=1457
Tutorial – How to make data transformations using Data Collection Rules?	https://mortenknudsen.net/?p=1440

GiveAway Questions 1&2

When will Microsoft retire Microsoft Monitoring Agent (MMA) ? Month + Year

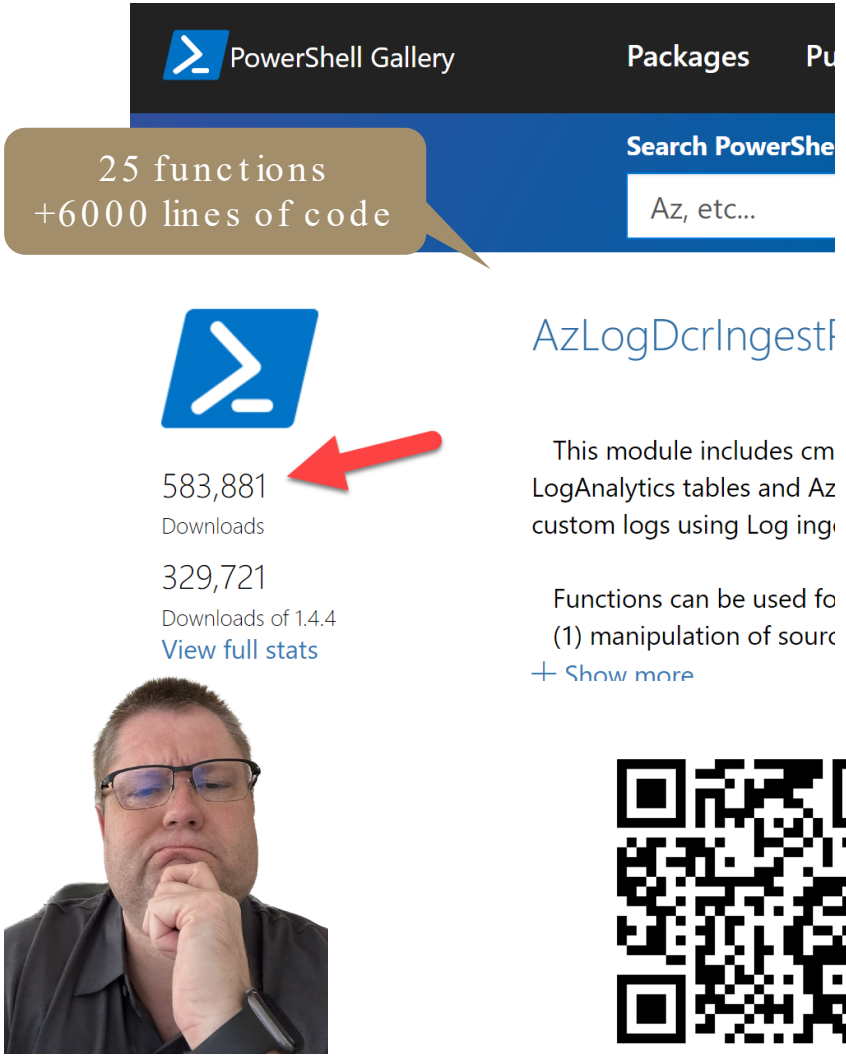
When will Microsoft retire HTTP Data Collector api ?

Please come up to me afterwards ☺

Challenges with v2 technologies

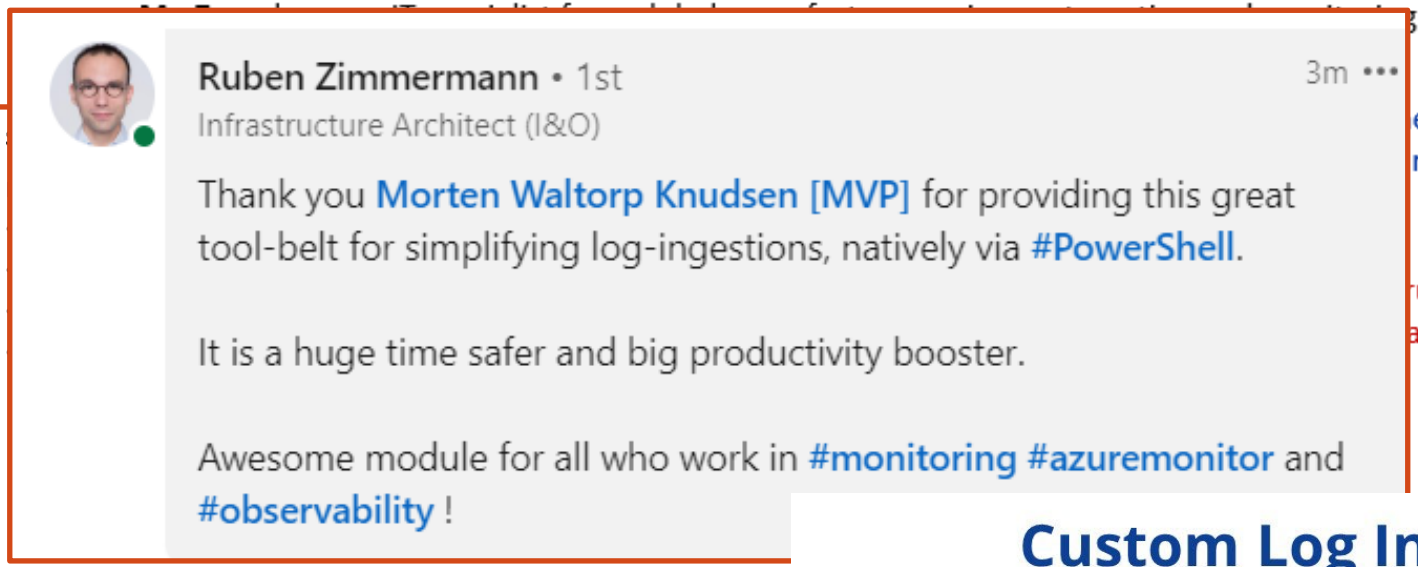
compared to V1 - HTTP Data Collector API

- Req: **Creation of DCR + tables** before sending data
- Req: **Data Collection EndPoint** must exist
- Req: **Schema** for data must be defined in both DCR and custom table (v2)
- **Naming conventions** &limitations / Prohibited names
- Handle new properties in source object (**merge, overwrite**)
- **Property changes** -> True (string) → True (Boolean) – "internal server 500"
- **Upload changes** (32 mb -> 1 mb) per JSON (batches, calculations)
- **Data manipulations** of source data (filtering, remove)
- **DCR limitations** with large schema
- **Azure seconds** - timing / delays – dependencies
- **Platform** – no-internet access, TLS incompatibility



Great feedback about AzLogDcrIngestPS ☺

Really happy it gives value to the community ☺



since his first day.

e new approach
ready about to

rustrated by the
acters and naming

Assistance by AzLogDcrIngestPS:

- No wasting time for using templates and guessing valid column and resour
- Simply passing objects to the module and knowing that data types match a happen

Custom Log Ingestions

AzLogDcrIngestPS - Simplification and Time-Saver!

Aug 2023

Official Microsoft documentation reference

https://learn.microsoft.com/en-us/azure/azure-monitor/logs/custom-logs-migrate

Migrate from the HTTP Data Collector API to the Log Ingestion API to send data to Azure Monitor Logs

Article • 06/19/2023 • 5 contributors

Feedback

In this article

- Advantages of the Log Ingestion API
- Prerequisites
- Create new resources required for the Log ingestion API
- Migrate existing custom tables or create new tables

Show 3 more

The Azure Monitor Log Ingestion API provides more processing power and greater flexibility in ingesting logs and managing tables than the legacy HTTP Data Collector API. This article describes the differences between the Data Collector API and the Log Ingestion API and provides guidance and best practices for migrating to the new Log Ingestion API.

ⓘ Note

As a Microsoft MVP, **Morten Waltorp Knudsen** contributed to and provided material feedback for this article. For an example of how you can automate the setup and ongoing use of the Log Ingestion API, see Morten's publicly available **AzLogDcrIngestPS PowerShell module**.

https://learn.microsoft.com/en-us/azure/azure-monitor/logs/set-up-logs-ingestion-api-prerequisites

Learn / Azure / Azure Monitor /

Set up resources required to send data to Azure Monitor Logs using the Logs Ingestion API

Article • 06/22/2023 • 2 contributors

Feedback

In this article

- Create resources and permissions
- PowerShell script
- Next steps

This article provides a PowerShell script that sets up all of the resources you need before you can send data to Azure Monitor Logs using the Logs ingestion API.

ⓘ Note

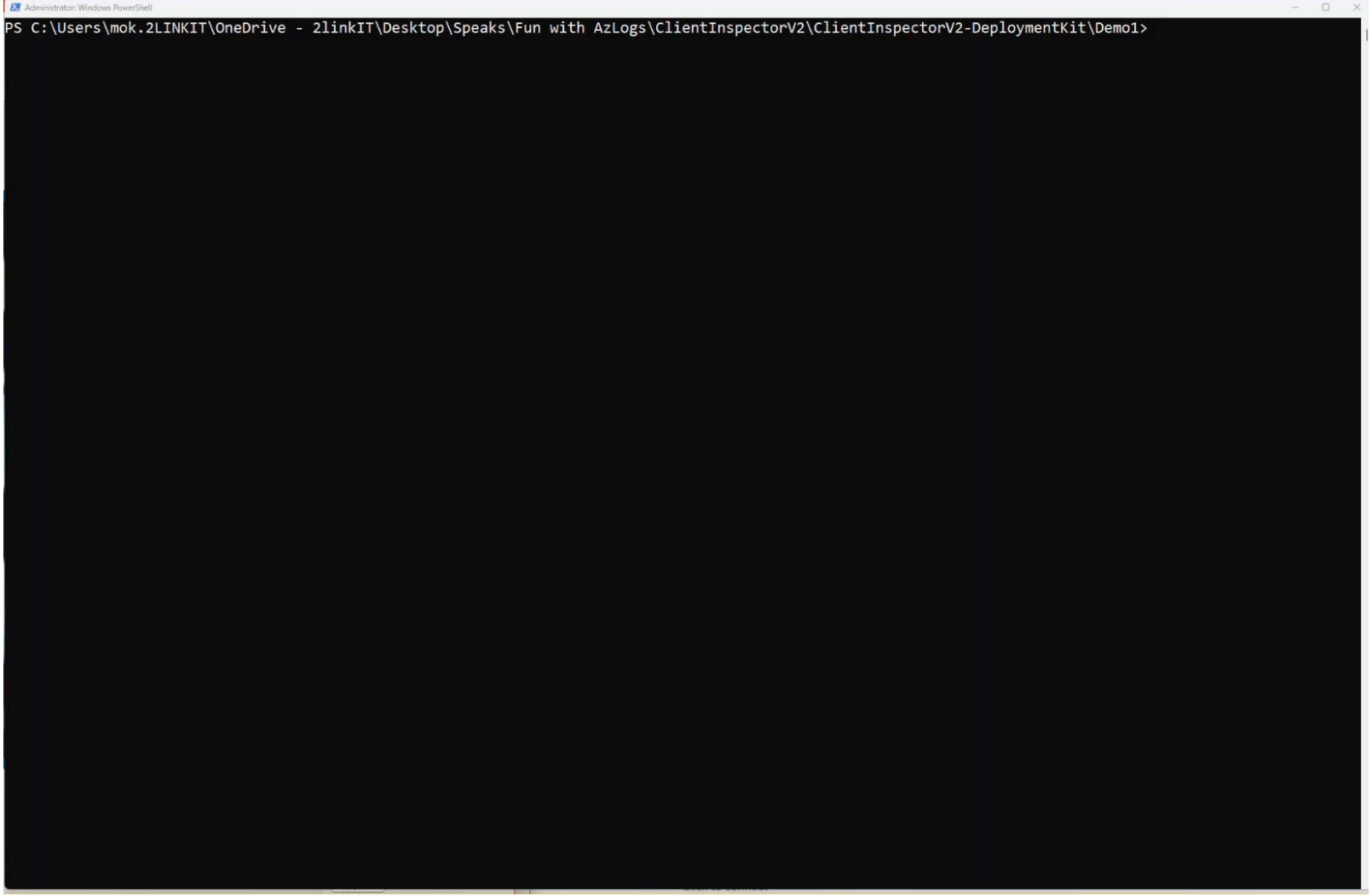
As a Microsoft MVP, **Morten Waltorp Knudsen** contributed to and provided material feedback for this article. For an example of how you can automate the setup and ongoing use of the Log Ingestion API, see Morten's **AzLogDcrIngestPS PowerShell module**.

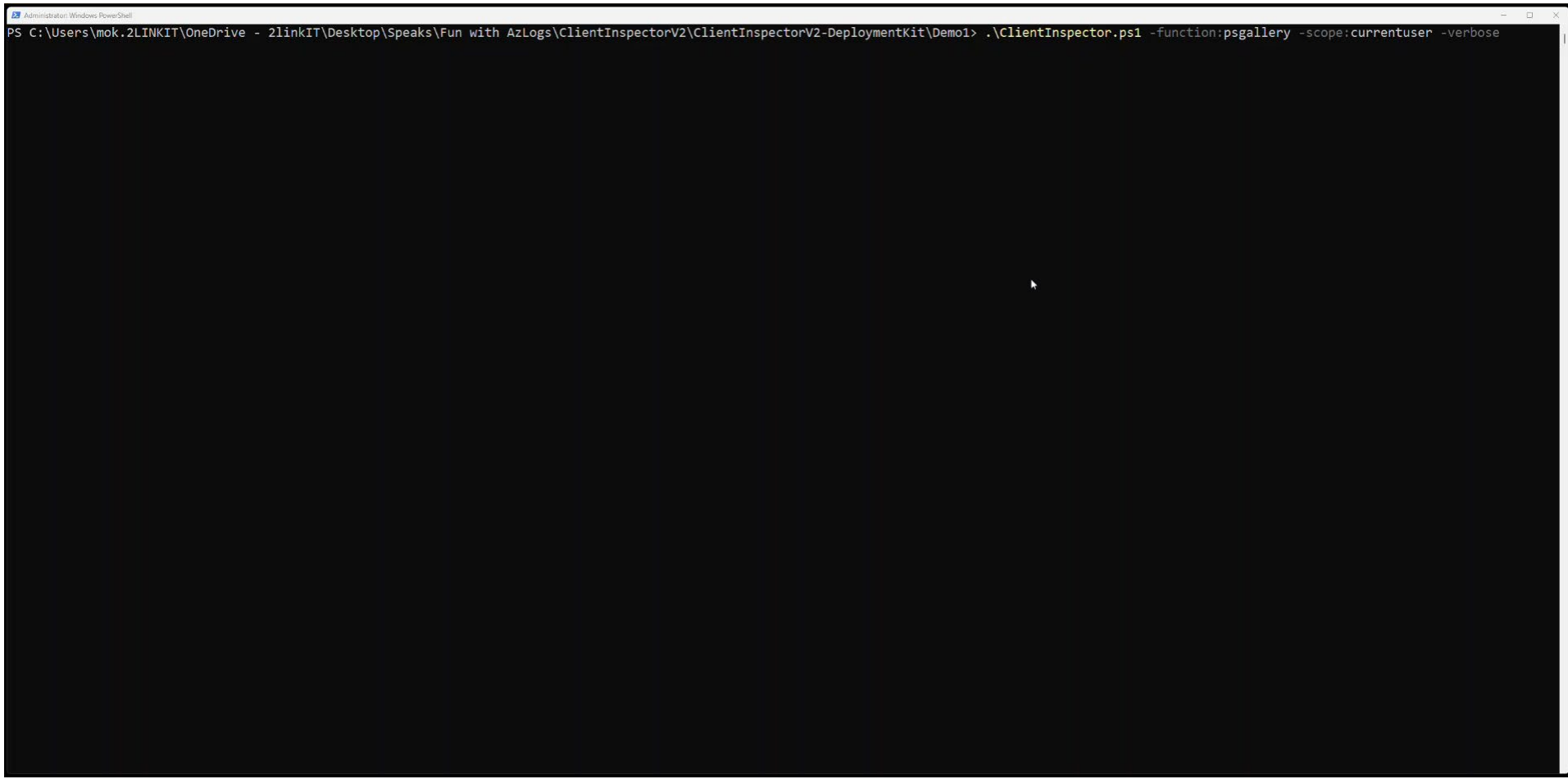
Create resources and permissions

The script creates these resources, if they don't already exist:

- A Log Analytics workspace and a resource group for the Log Analytics workspace.

You probably already have a Log Analytics workspace, in which case, provide the

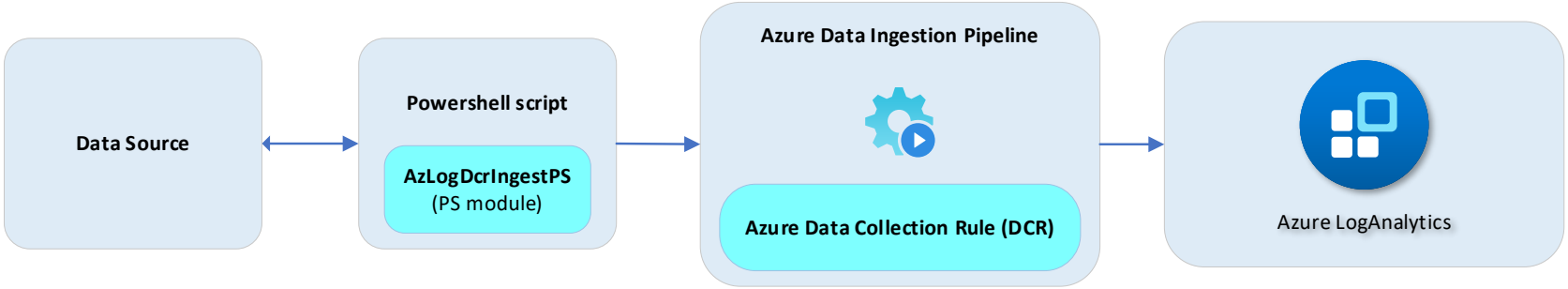

Administrator: Windows PowerShell
PS C:\Users\mok.2LINKIT\OneDrive - 2linkIT\Desktop\Speaks\Fun with AzLogs\ClientInspectorV2\ClientInspectorV2-DeploymentKit\Demo1>


Administrator: Windows PowerShell
PS C:\Users\mok.2LINKIT\OneDrive - 2linkIT\Desktop\Speaks\Fun with AzLogs\ClientInspectorV2\ClientInspectorV2-DeploymentKit\Demo1> .\ClientInspector.ps1 -function:psgallery -scope:currentuser -verbose

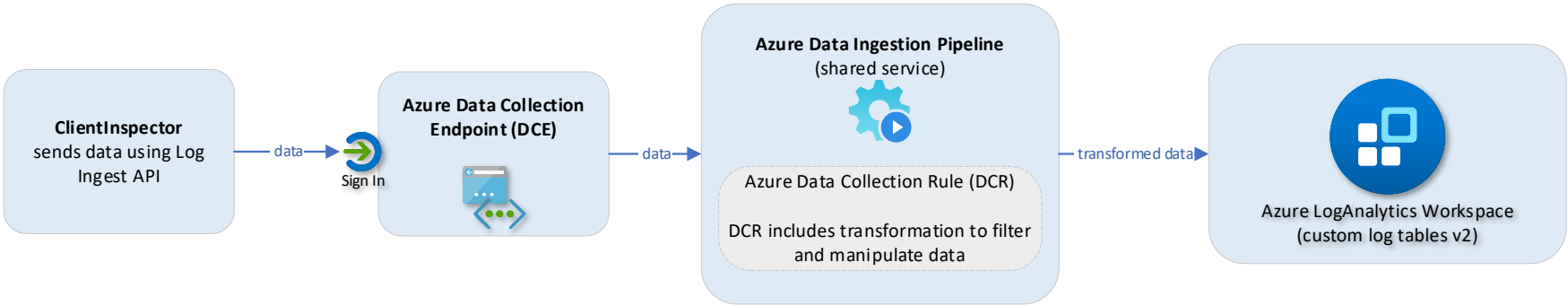
Use-Case | AzLogDcrIngestPS

"AnyConnector"

Pull - Retrieve data from any 3rd party source using PS script (intermediate)



Push - Collect data from endpoint - send data from endpoint (ClientInspector)

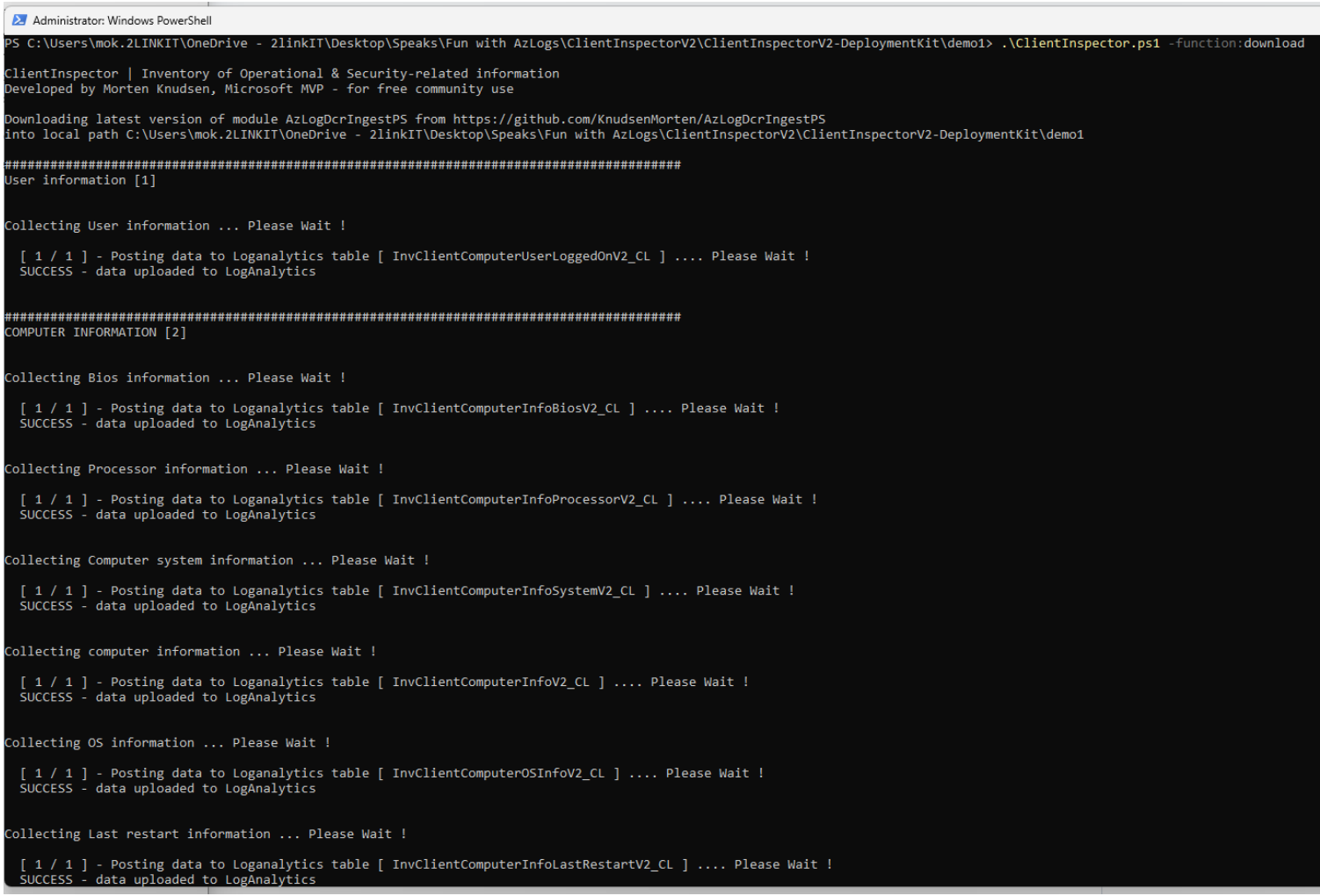


Cloud Connect

Run with

ConfigMgr
Intune
any deployment-tool

Servers:
RemotePS
CustomScript Extension



Cloud Connect

Name

ClientInspector_Detection_1.ps1

ClientInspector_Detection_2.ps1

ClientInspector_intune_script1.ps1

ClientInspector_intune_script2.ps1

Home > Endpoint analytics

Endpoint analytics | Proactive remediations

Search

Overview

Settings

Reports

Startup performance

Proactive remediations

Application reliability

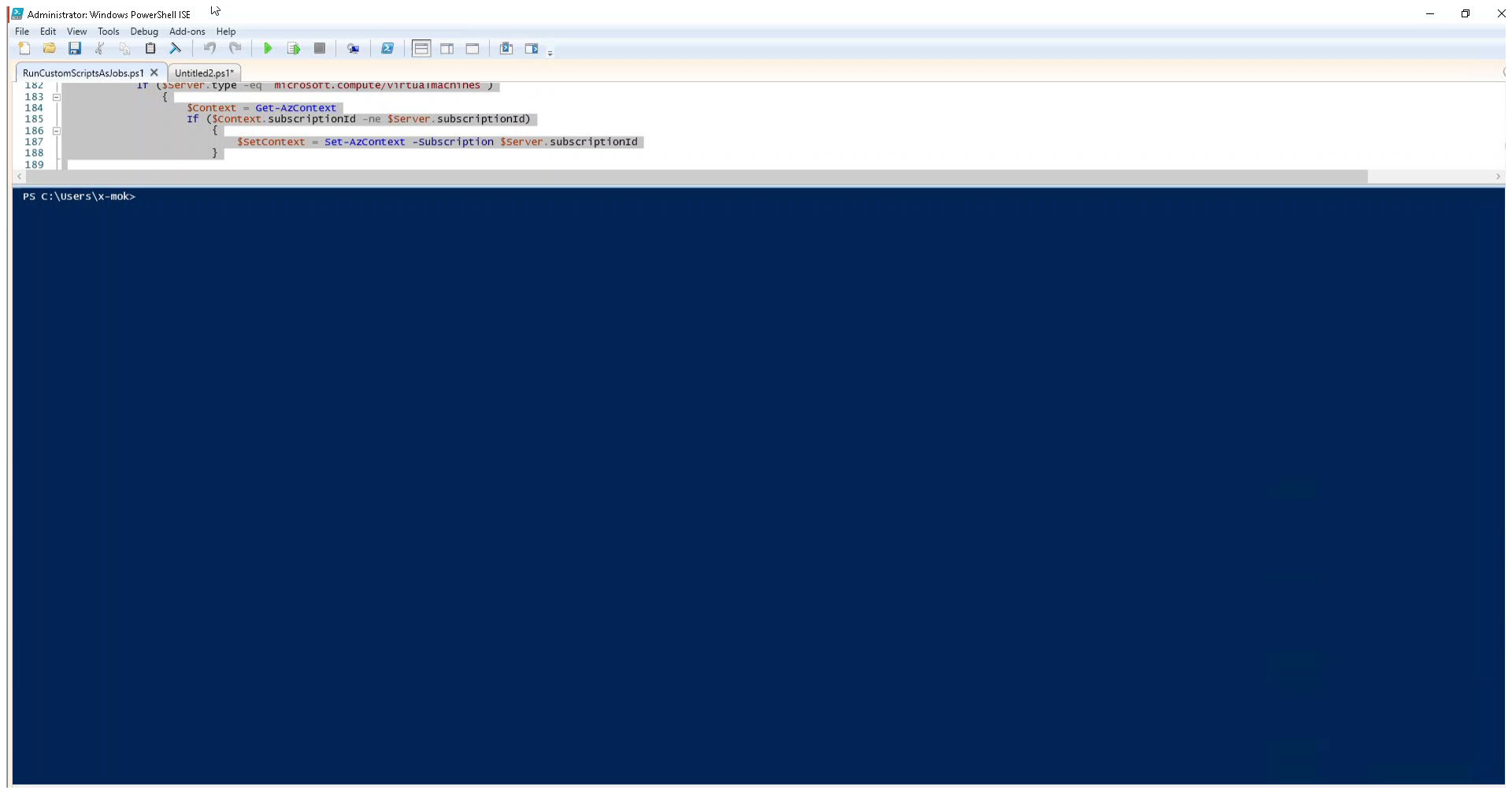
Work from anywhere

Refresh Create script package Columns

Create and run script packages on devices to proactively find and fix the top support issues in your organization. Use this table to see the status of your deployed script packages and to monitor the detection and remediation results. Results are shown as number of devices affected. Learn more.

Search by script package name

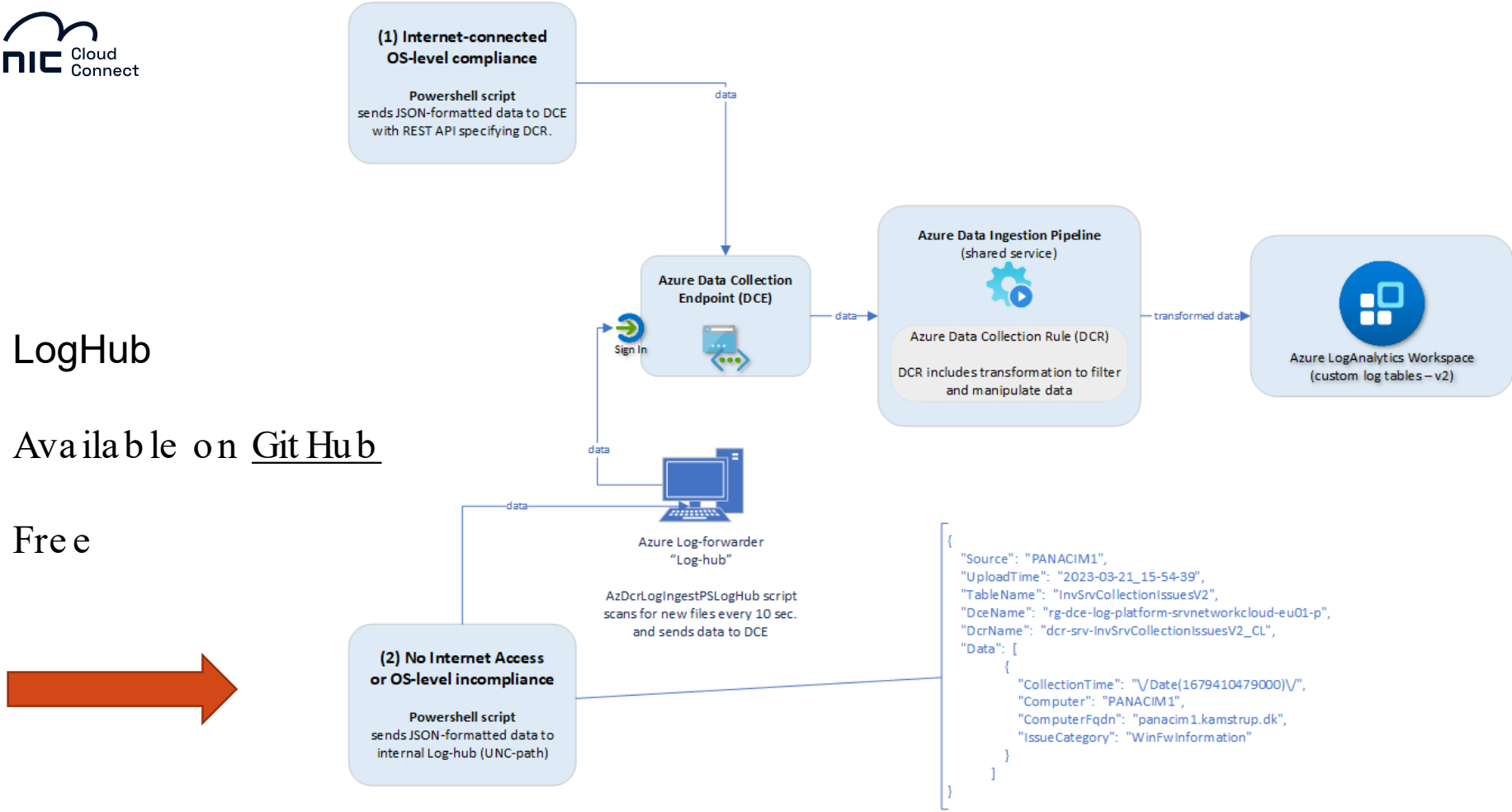
Script package name	Author	Status	Without issues	With issu
Restart stopped Office C2R svc	Microsoft	Not deployed	0	0
ClientInspector (daily) - part 2/2	Morten Knudsen	Active	18	7
ClientInspector (daily) - part 1/2	Morten Knudsen	Active	16	9
Remove Desktop Duplicates	GT	Active	23	1
Update stale Group Policies	Microsoft	Not deployed	0	0
Remove Built-in Teams - Windows 11	nlo	Active	58	4

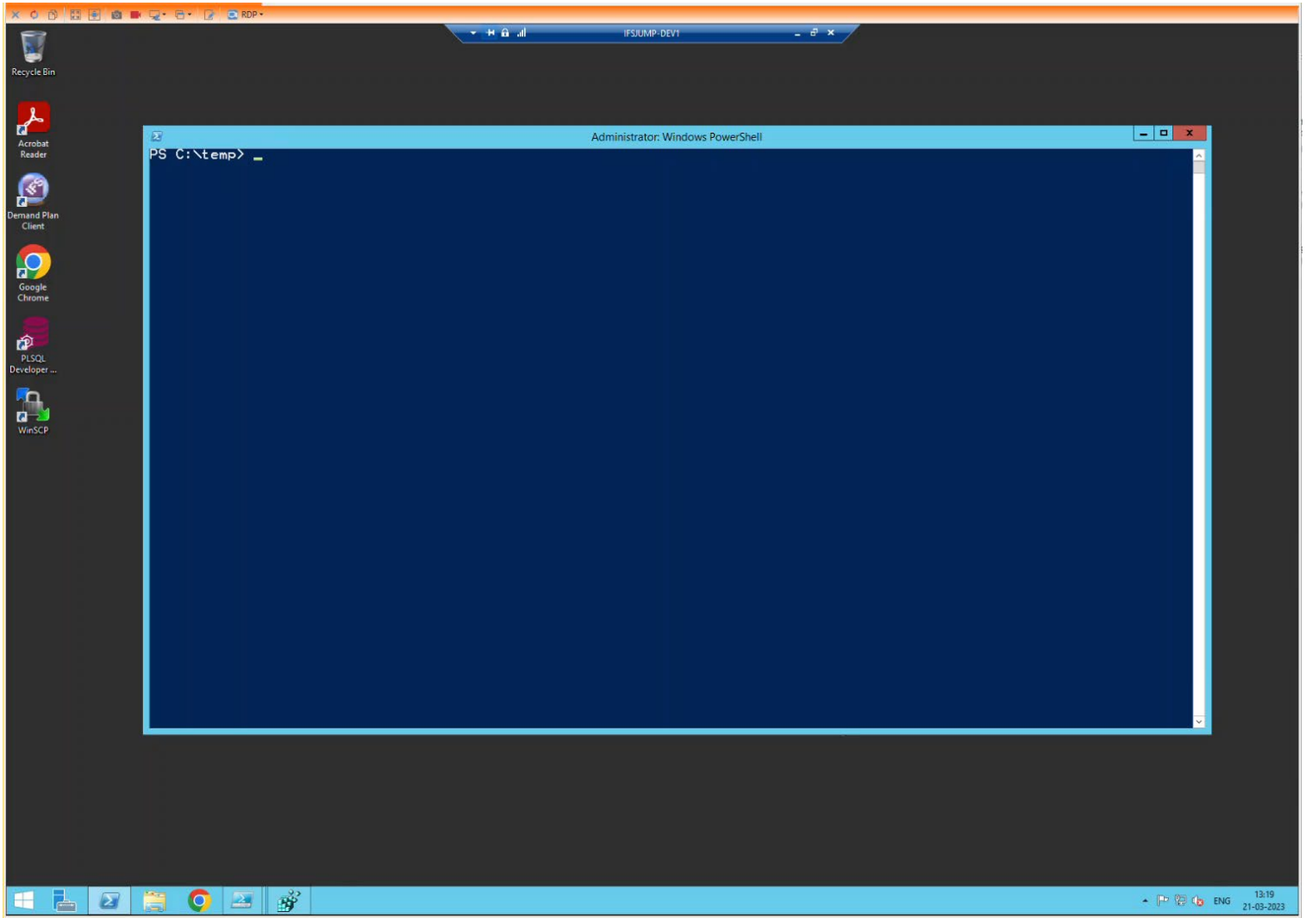

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
RunCustomScriptsAsJobs.ps1
Untitled2.ps1*
182 If ($Server.type -eq "microsoft.compute/virtualmachines")
183 {
184 $Context = Get-AzContext
185 If ($Context.subscriptionId -ne $Server.subscriptionId)
186 {
187 $SetContext = Set-AzContext -Subscription $Server.subscriptionId
188 }
189
PS C:\Users\x-mok>

LogHub

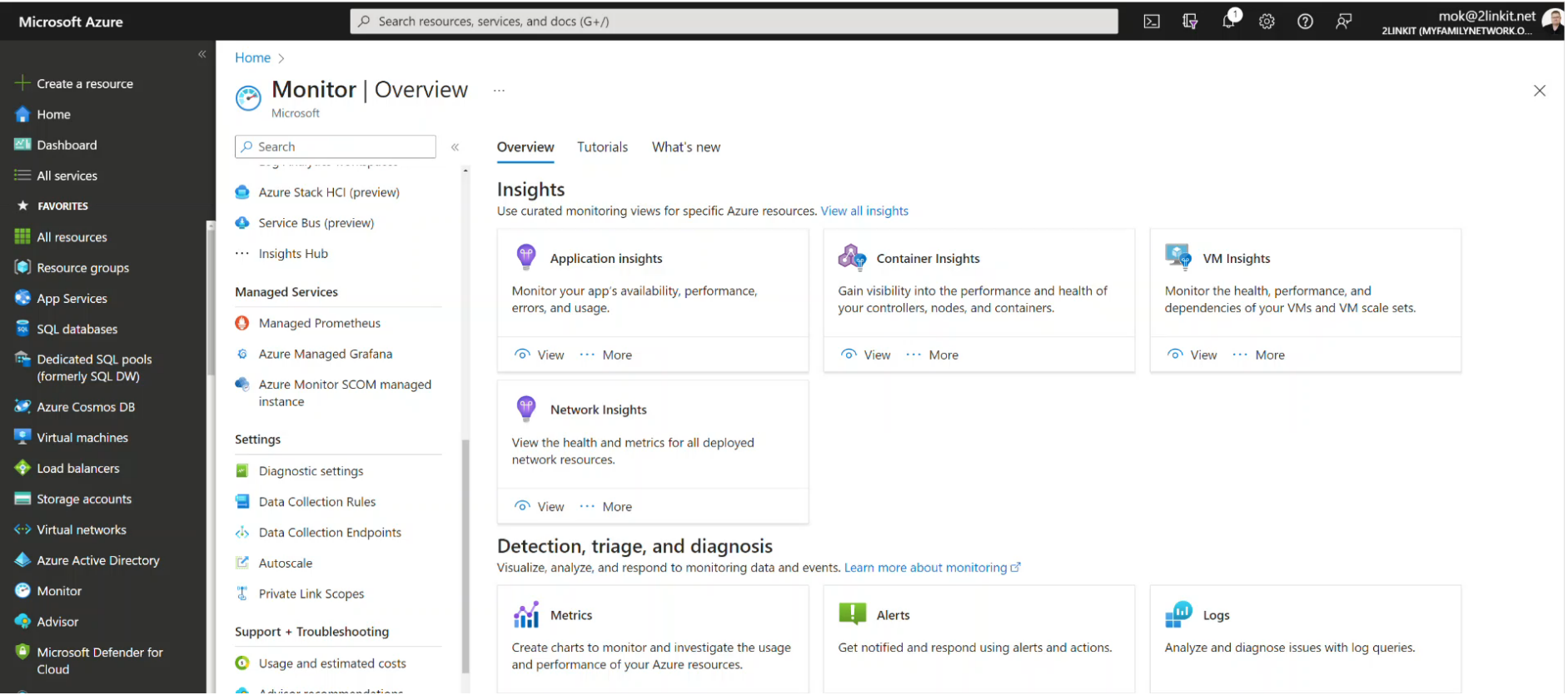
Available on GitHub

Free




RDP
IFSJUMP-DEV1
Recycle Bin
Acrobat Reader
Demand Plan Client
Google Chrome
PLSQL Developer ...
WinSCP
Administrator: Windows PowerShell
PS C:\temp> _
ENG
13:19
21-03-2023

Schema – getting "under the hood"



GiveAway Question #3

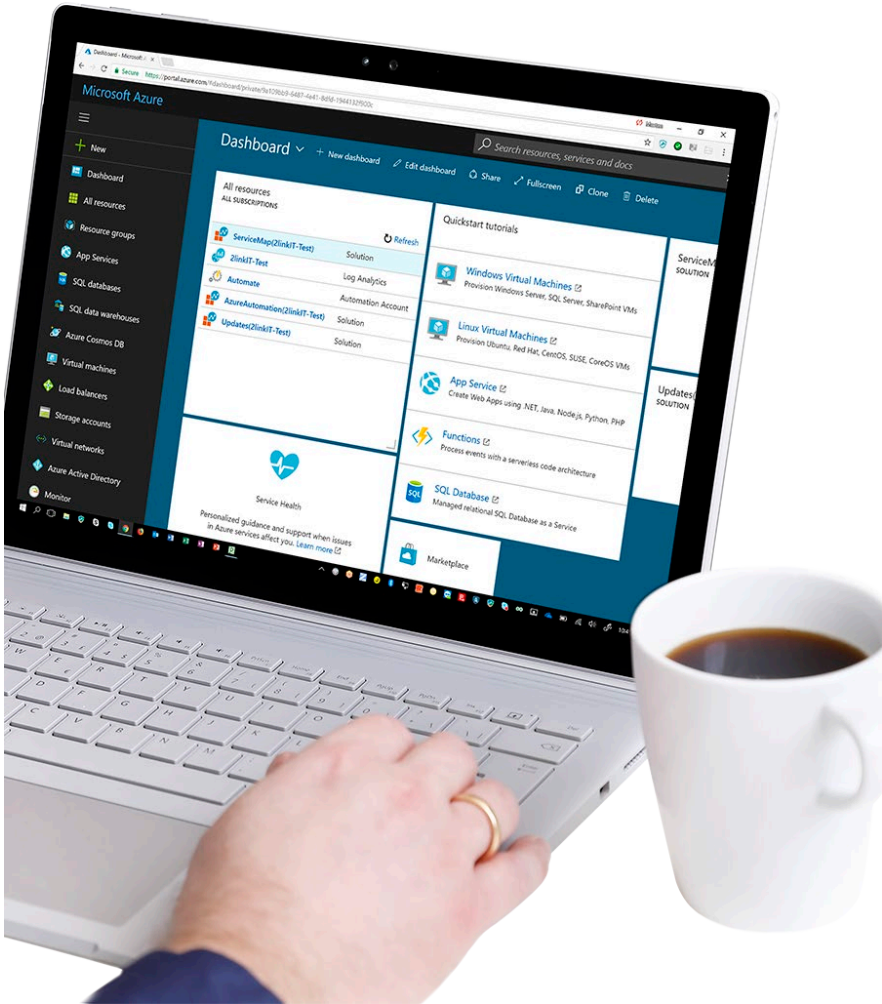
Who can mention the 2 main components you use when ingesting data through log ingestion api – besides Azure LogAnalytics custom table (v2) and Azure Pipeline ?

Hints:

They both have a 3 -letter acronym - both starting with D

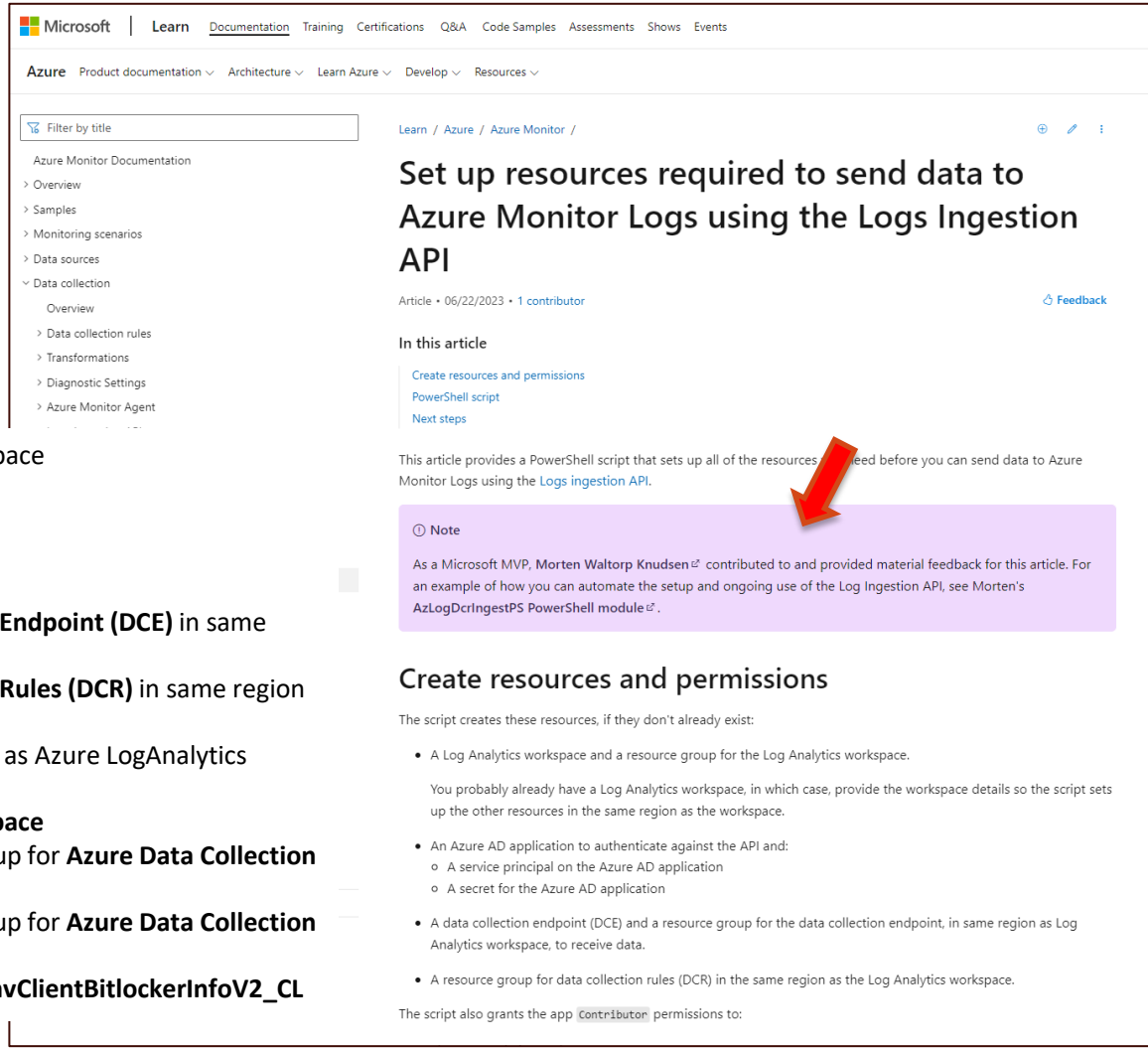
AzLogDcrIngestPS

“AnyConnector ”
deep - dive



Pre-req Environment for Log Analytics (v2)

1. create **Azure Resource Group** for Azure LogAnalytics Workspace
2. create **Azure LogAnalytics Workspace**
3. create **Azure App registration** used for upload of data
4. create **Azure service principal** on Azure App
5. create needed **secret** on Azure app
6. create the **Azure Resource Group** for **Azure Data Collection Endpoint (DCE)** in same region as Azure LogAnalytics Workspace
7. create the **Azure Resource Group** for **Azure Data Collection Rules (DCR)** in same region as Azure LogAnalytics Workspace
8. create **Azure Data Collection Endpoint (DCE)** in same region as Azure LogAnalytics Workspace
9. **delegate permissions** for Azure App on **LogAnalytics workspace**
10. **delegate permissions** for Azure App on Azure Resource Group for **Azure Data Collection Rules (DCR)**
11. **delegate permissions** for Azure App on Azure Resource Group for **Azure Data Collection Endpoints (DCE)**
12. define **naming convention for DCRs** – for example **dcr-clt-InvClientBitlockerInfoV2_CL**



Azure Monitor
(unique resource)

Data Collection Endpoint (DCE)

Properties:

immutableId

configurationAccess
(url for configuration)

logsIngestion
(url for data upload)

metricsIngestion
(url for monitoring)

networkAcls
publicNetworkAccess
= Enabled/Disabled


```
"properties": {
    "description": "DCE for LogIngest to LogAnalytics log-management-client-demo1-t",
    "immutableId": "dce-df70b659df0a47b99f09860ded66ee06",
    "configurationAccess": {
      "endpoint": "https://dce-log-management-client-demo1-t-2r31.westeurope-1.handler.control.monitor.azure.com"
    },
    "logsIngestion": {
      "endpoint": "https://dce-log-management-client-demo1-t-2r31.westeurope-1.ingest.monitor.azure.com"
    },
    "metricsIngestion": {
      "endpoint": "https://dce-log-management-client-demo1-t-2r31.westeurope-1.metrics.ingest.monitor.azure.com"
    },
    "networkAcls": {
      "publicNetworkAccess": "Enabled"
    },
    "provisioningState": "Succeeded"
  },
  "location": "westeurope",
  "id": "/subscriptions/fce4f282-fcc6-43fb-94d8-bf1701b862c3/resourceGroups/rg-dce-log-management-client-demo1-t/
provide
  "nam
  "typ
  "etag
  "syst
    "createdBy": "mok@2linkit.net",
    "createdByType": "User",
    "createdAt": "2023-03-19T10:57:58.6132975Z",
    "lastModifiedBy": "mok@2linkit.net",
    "lastModifiedByType": "User",
    "lastModifiedAt": "2023-03-19T10:57:58.6132975Z"
  }
}
```


Possible to optimize networking with
Azure Monitor Private Link Scope (AMPLS)

Flow with AzLogDcrIngestPS

Data In (“ AnyConnector ”)

- WMI, CIM, REST API, DBs, CSV, JSON, XML, etc.

Data Manipulate

- remove data from the source object, if there are colums of data you don't want to send
- convert source objects based on CIM or PS objects into PSCustomObjects /array
- add relevant information to each record like UserLoggedOn, Computer, CollectionTime

Table /DCR/Schema / Transformation management

- SchemaMode = Merge & Overwrite
- SchemaMode = Overwrite
  - create/update the DCRs and tables automatically - based on the source object schema
- SchemaMode = Merge
  - create/update the tables automatically - based on the source object schema (add new properties to existing).
  - create/update DCR based on table schema
- validate the schema for naming convention issues. If exist found, it will mitigate the issues
- auto -fix if something goes wrong with a DCR or table (DELETE/PUT)

Data Upload

- Upload data to LA
  - Automatic calculation of batches (json size)
  - Can be overwritten by batchAmount <number>

Support

- Security (token REST)
- List DCRs/DCEs
- Maintenance (delete)
- Transformations (GET, UPDATE, RESET)

Structure 1/5: Variables

naming - where to send the data

```
$TableName                                  = "V2CustomTable" + $DemoNumber

$TenantId                                   = "f0fa27a0-8e7c-4f63-9a77-ec94786b7c9e"
$LogIngestAppId                             = "8cd71693-be69-421c-a04f-54d45e1e305d"
$LogIngestAppSecret                         = "BgN8Q~a7-LvufU-PZmGyzSd7.ryOTeZNLmD2Yb_G"

$DceName                                    = "dce-log-loganalyticsv2-migration-demo"
$AzDcrResourceGroup                         = "rg-dcr-log-loganalyticsv2-migration-demo"

$LogAnalyticsWorkspaceResourceId            = "…/providers/Microsoft.OperationalInsights/workspaces/log-loganalyticsv2-demo"

$AzDcrSetLogIngestApiAppPermissionsDcrLevel = $false
$AzDcrLogIngestServicePrincipalObjectId     = "2f259c5a-503f-4c36-909d-9d97c4e5cf8d"

$AzLogDcrTableCreateFromReferenceMachine    = @()
$AzLogDcrTableCreateFromAnyMachine          = $true
$DcrName                                    = "dcr-demo-" + $TableName + "_CL"

$Verbose                                    = $true
```

```
#--------------------------------------------------------------------------
# Variables
#--------------------------------------------------------------------------

    $TableName  = 'InvClientDefenderAvV2'   # must not contain _CL
    $DcrName    = "dcr-" + $AzDcrPrefix + "-" + $TableName + "_CL"
```

Structure 2/5: Data Collection

```
#-------------------------------------------------------------------------
# Collecting data (in)
#-------------------------------------------------------------------------

    Write-Output ""
    Write-Output "Collecting Bios information ... Please Wait !"

    $DataVariable = Get-CimInstance -ClassName Win32_BIOS

"
```

```
#-------------------------------------------------------------------------
# Collecting data (in)
#-------------------------------------------------------------------------

    Write-Output ""
    Write-Output "Collecting Microsoft Defender Antivirus information ... Please Wait !"

    $MPComputerStatus = Get-MpComputerStatus
    $MPPreference = Get-MpPreference
```

```
AMEngineVersion                 : 1.1.23050.3
AMProductVersion                : 4.18.23050.3
AMRunningMode                   : Normal
AMServiceEnabled                : True
AMServiceVersion                : 4.18.23050.3
AntispywareEnabled              : True
AntispywareSignatureAge         : 0
AntispywareSignatureLastUpdated : 11-06-2023 06:18:21
AntispywareSignatureVersion     : 1.391.1111.0
AntivirusEnabled                : True
AntivirusSignatureAge           : 0
AntivirusSignatureLastUpdated   : 11-06-2023 06:18:21
AntivirusSignatureVersion       : 1.391.1111.0
BehaviorMonitorEnabled          : True
CimClass                        :
ROOT/Microsoft/Windows/Defender:MSFT_MpComputerStatus
CimInstanceProperties           : {AMEngineVersion,
AMProductVersion, AMRunningMode, AMServiceEnabled...}
CimSystemProperties             :
Microsoft.Management.Infrastructure.CimSystemProperties
CollectionTime                  : 11-06-2023 13:02:42
Computer                        : STRV-MOK-DT-02
ComputerFqdn                    : STRV-MOK-DT-02.WORKGROUP
ComputerID                      : A39E177B-9D86-A3AE-93EC-
A9FA04AC9191
ComputerState                   : 0
DefenderSignaturesOutOfDate     : False
DeviceControlDefaultEnforcement : Default Allow
DeviceControlPoliciesLastUpdated : 27-03-2023 00:34:19
DeviceControlState              : Disabled
FullScanAge                     : 83
FullScanEndTime                 : 20-03-2023 01:35:18
FullScanOverdue                 : False
FullScanRequired                : False
FullScanSignatureVersion        : 1.385.490.0
FullScanStartTime               : 19-03-2023 22:31:53
IoavProtectionEnabled           : True
IsTamperProtected               : True
IsVirtualMachine                : False
LastFullScanSource              : 2
LastQuickScanSource             : 2
NISEnabled                      : True
NISEngineVersion                : 1.1.23050.3
NISSignatureAge                 : 0
NISSignatureLastUpdated         : 11-06-2023 06:18:21
NISSignatureVersion             : 1.391.1111.0
OnAccessProtectionEnabled       : True
ProductStatus                   : 524288
PSComputerName                  :
QuickScanAge                    : 0
QuickScanEndTime                : 11-06-2023 07:16:51
QuickScanOverdue                : False
QuickScanSignatureVersion       : 1.391.1075.0
QuickScanStartTime              : 11-06-2023 03:52:21
RealTimeProtectionEnabled       : True
RealTimeScanDirection           : 0
RebootRequired                  : False
SmartAppControlExpiration       :
SmartAppControlState            : Off
TamperProtectionSource          : Intune
TDTMode                         : N/A
TDTSiloType                     : N/A
TDTStatus                       : N/A
TDTTelemetry                    : N/A
TroubleShootingDailyMaxQuota    : 480
TroubleShootingDailyQuotaLeft   : 480
TroubleShootingEndTime          : INFINITE
TroubleShootingExpirationLeft   : INFINITE
TroubleShootingMode             : Disabled
TroubleShootingModeSource       : Service
TroubleShootingQuotaResetTime   : N/A
TroubleShootingStartTime        : N/A
UserLoggedOn                    : 2LINKIT\mok
```

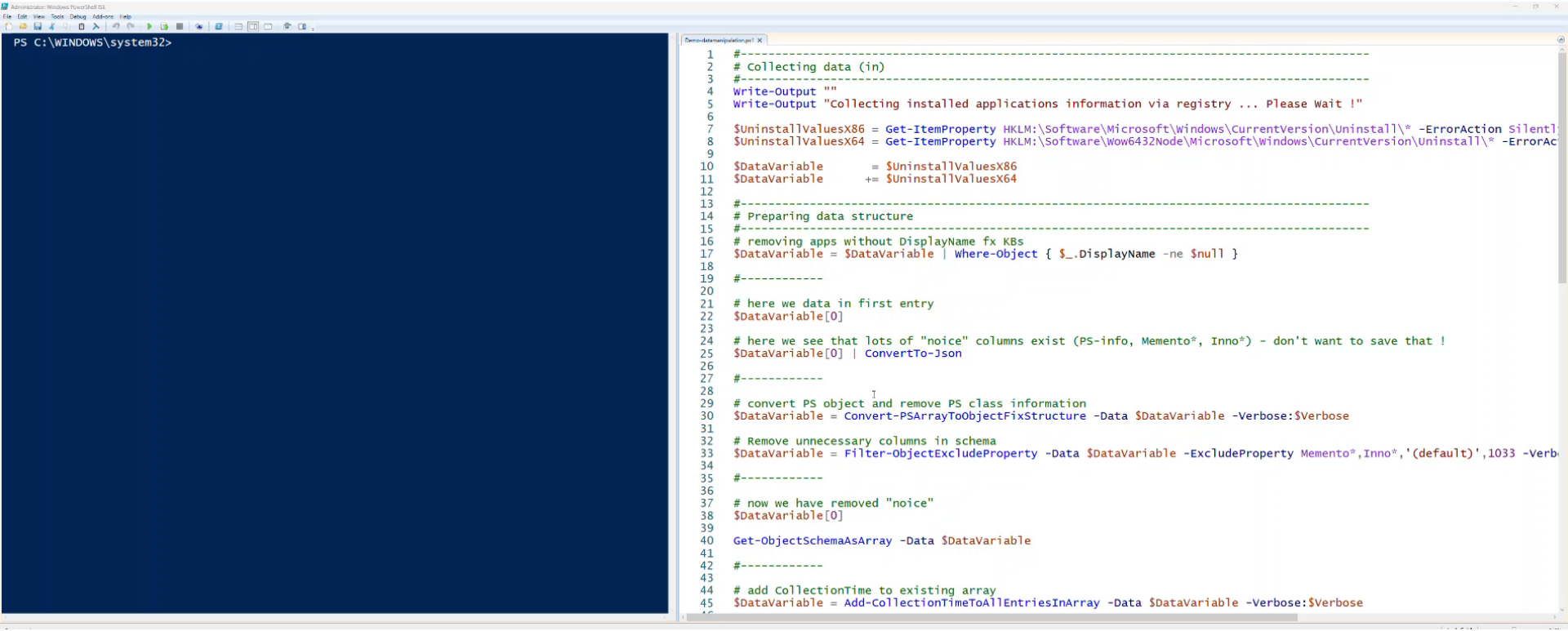
Structure 3/5: Data Manipulation

ensure data is in correct format and any "noise" was removed and relevant information has been added

```powershell
# removing apps without DisplayName fx KBs
$DataVariable = $DataVariable | Where-Object { $_.DisplayName -ne $null }

# convert PS object and remove PS class information
$DataVariable = Convert-PSArrayToObjectFixStructure -Data $DataVariable -Verbose:$Verbose

# add CollectionTime to existing array
$DataVariable = Add-CollectionTimeToAllEntriesInArray -Data $DataVariable -Verbose:$Verbose

# add Computer, ComputerFqdn & UserLoggedOn info to existing array
$DataVariable = Add-ColumnDataToAllEntriesInArray -Data $DataVariable -Column1Name Computer -Column1Data $Env:ComputerName -Column2Name ComputerFqdn -Column2

# Get insight about the schema structure of an object BEFORE changes. Command is only needed to verify columns in schema
# $SchemaBefore = Get-ObjectSchemaAsArray -Data $DataVariable

# Remove unnecessary columns in schema
$DataVariable = Filter-ObjectExcludeProperty -Data $DataVariable -ExcludeProperty Memento*,Inno*,'(default)',1033 -Verbose:$Verbose

# Validating/fixing schema data structure of source data
$DataVariable = ValidateFix-AzLogAnalyticsTableSchemaColumnNames -Data $DataVariable -Verbose:$Verbose

# Aligning data structure with schema (requirement for DCR)
$DataVariable = Build-DataArrayToAlignWithSchema -Data $DataVariable -Verbose:$Verbose
```

```
VERBOSE: Getting Data Collection Rules from Azure Resource Graph .... Please Wait !
VERBOSE: POST with -1-byte payload
VERBOSE: received 1468-byte response of content type application/json; charset=utf-8
VERBOSE: POST with -1-byte payload
VERBOSE: received 11713-byte response of content type application/json; charset=utf-8
VERBOSE:   Adding CollectionTime to all entries in array .... please wait !
VERBOSE:   Adding columns to all entries in array .... please wait !
VERBOSE:   Validating schema structure of source data ... Please Wait !
VERBOSE:   SUCCESS - No issues found in schema structure
VERBOSE:   Aligning source object structure with schema ... Please Wait !
```

Data Manipulation demo



Structure 4/5: Table & DCR management

```
CheckCreateUpdate-TableDcr-Structure -AzLogWorkspaceResourceId $LogAnalyticsWorkspaceResourceId -SchemaMode Merge `
                                     -AzAppId $LogIngestAppId -AzAppSecret $LogIngestAppSecret -TenantId $TenantId -Verbose:$Verbose `
                                     -DceName $DceName -DcrName $DcrName -DcrResourceGroup $AzDcrResourceGroup -TableName $TableName `
                                     -LogIngestServicePricipleObjectId $AzDcrLogIngestServicePrincipalObjectId `
                                     -Data $DataVariable -AzDcrSetLogIngestApiAppPermissionsDcrLevel $AzDcrSetLogIngestApiAppPermissionsDcrLevel `
                                     -AzLogDcrTableCreateFromAnyMachine $AzLogDcrTableCreateFromAnyMachine `
                                     -AzLogDcrTableCreateFromReferenceMachine $AzLogDcrTableCreateFromReferenceMachine
```

```
VERBOSE:   Checking LogAnalytics table and Data Collection Rule configuration .... Please Wait !
VERBOSE: POST with -1-byte payload
VERBOSE: received 1468-byte response of content type application/json; charset=utf-8
VERBOSE: GET with 0-byte payload
VERBOSE:   LogAnalytics table wasn't found !
VERBOSE:   DCR was not found [ dcr-demo-V2CustomTable5778_CL ]
VERBOSE: GET with 0-byte payload
VERBOSE: POST with -1-byte payload
VERBOSE: received 1468-byte response of content type application/json; charset=utf-8
VERBOSE: GET with 0-byte payload
VERBOSE:
VERBOSE: Trying to update existing LogAnalytics table schema for table [ V2CustomTable5778_CL ] in
VERBOSE: /subscriptions/fce4f282-fcc6-43fb-94d8-bf1701b862c3/resourceGroups/rg-loganalyticsv2demo/providers/Microsoft.OperationalInsights/workspaces/log-loganalyticsv2-migration-demo
VERBOSE: PUT with -1-byte payload
VERBOSE: received 8133-byte response of content type application/json; charset=utf-8
VERBOSE: POST with -1-byte payload
VERBOSE: received 1468-byte response of content type application/json; charset=utf-8
VERBOSE: Found required DCE info using Azure Resource Graph
VERBOSE:
VERBOSE: GET with 0-byte payload
VERBOSE: received 893-byte response of content type application/json; charset=utf-8
VERBOSE: Found required LogAnalytics info
VERBOSE:
VERBOSE: GET with 0-byte payload
VERBOSE:
VERBOSE: Creating/updating DCR [ dcr-demo-V2CustomTable5778_CL ] with limited payload
VERBOSE: /subscriptions/fce4f282-fcc6-43fb-94d8-bf1701b862c3/resourceGroups/rg-dcr-log-loganalyticsv2-migration-demo/providers/microsoft.insights/dataCollectionRules/dcr-demo-V2CustomTable5778_CL
VERBOSE: PUT with -1-byte payload
VERBOSE: received 2089-byte response of content type application/json; charset=utf-8
VERBOSE:
VERBOSE: Updating DCR [ dcr-demo-V2CustomTable5778_CL ] with full payload
VERBOSE: /subscriptions/fce4f282-fcc6-43fb-94d8-bf1701b862c3/resourceGroups/rg-dcr-log-loganalyticsv2-migration-demo/providers/microsoft.insights/dataCollectionRules/dcr-demo-V2CustomTable5778_CL
VERBOSE: PUT with -1-byte payload
VERBOSE: received 4805-byte response of content type application/json; charset=utf-8
VERBOSE:
VERBOSE: Waiting 10 sec to let Azure sync up so DCR rule can be retrieved from Azure Resource Graph
```

Structure 5/5: Upload data using Log Ingestion API

```
Post-AzLogAnalyticsLogIngestCustomLogDcrDce-Output -DceName $DceName `
                                                   -DcrName $DcrName `
                                                   -Data $DataVariable `
                                                   -TableName $TableName `
                                                   -AzAppId $LogIngestAppId `
                                                   -AzAppSecret $LogIngestAppSecret `
                                                   -TenantId $TenantId `
                                                   -Verbose:$Verbose
```

Automatic calculation of size of JSON

```
VERBOSE: Getting Data Collection Rules from Azure Resource Graph .... Please Wait !
VERBOSE: POST with -1-byte payload
VERBOSE: received 1468-byte response of content type application/json; charset=utf-8
VERBOSE: POST with -1-byte payload
VERBOSE: received 11713-byte response of content type application/json; charset=utf-8
VERBOSE: POST with -1-byte payload
VERBOSE: received 1468-byte response of content type application/json; charset=utf-8
VERBOSE: POST with -1-byte payload
VERBOSE: received 1302-byte response of content type application/json; charset=utf-8
```

Upload - mitigation of POST error 513 (dataset too large)

Upload data using DCR / DCE/ Log Ingestion API

```
Post-AzLogAnalyticsLogIngestCustomLogDcrDce-Output -DceName $DceName `
                                                   -DcrName $DcrName `
                                                   -Data $DataVariable `
                                                   -TableName $TableName `
                                                   -AzAppId $LogIngestAppId `
                                                   -AzAppSecret $LogIngestAppSecret `
                                                   -TenantId $TenantId `
                                                   -Verbose:$Verbose `
                                                   -BatchAmount <number of rows>
```

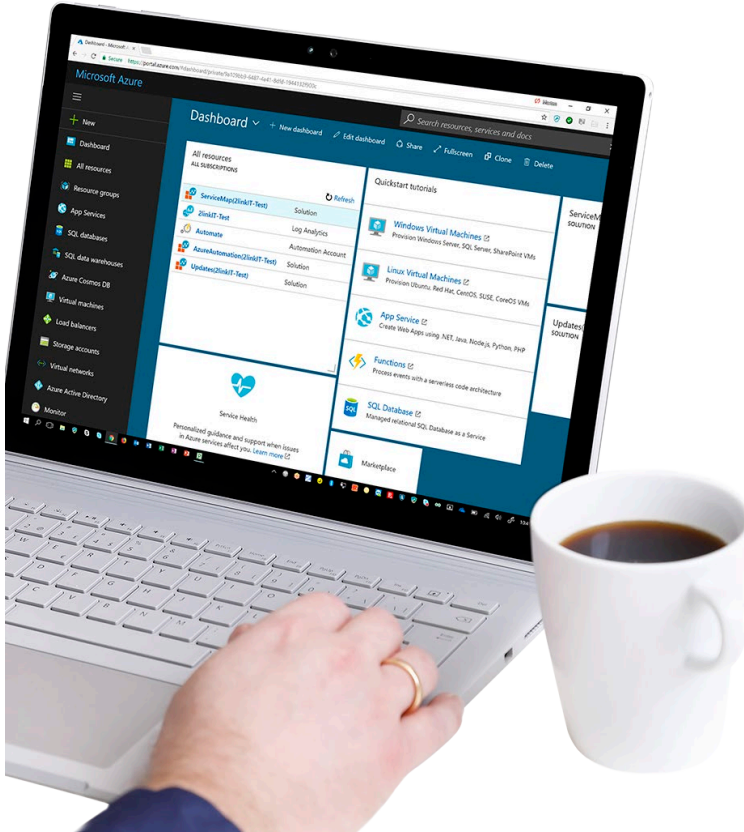
Multiple records in data -set with different size – force specific batch -amounts per POST

Limit	Value
Maximum size of API call	1 MB for both compressed and uncompressed data.
Maximum data/minute per DCR	2 GB for both compressed and uncompressed data. Retry after the duration listed in the Retry-After header in the response.
Maximum requests/minute per DCR	12,000. Retry after the duration listed in the Retry-After header in the response.

nic Cloud Connect

Migration to LogAnalytics (v2) using API

Scenarios



Migration Scenarios from v1 to v2

Side-by-Side

- New table with nice naming convention
- Any integrations, dashboards, etc. can be adjusted before “turn -key”
- Migration can be done ‘per table’
- Old data will exist until retention period runs out

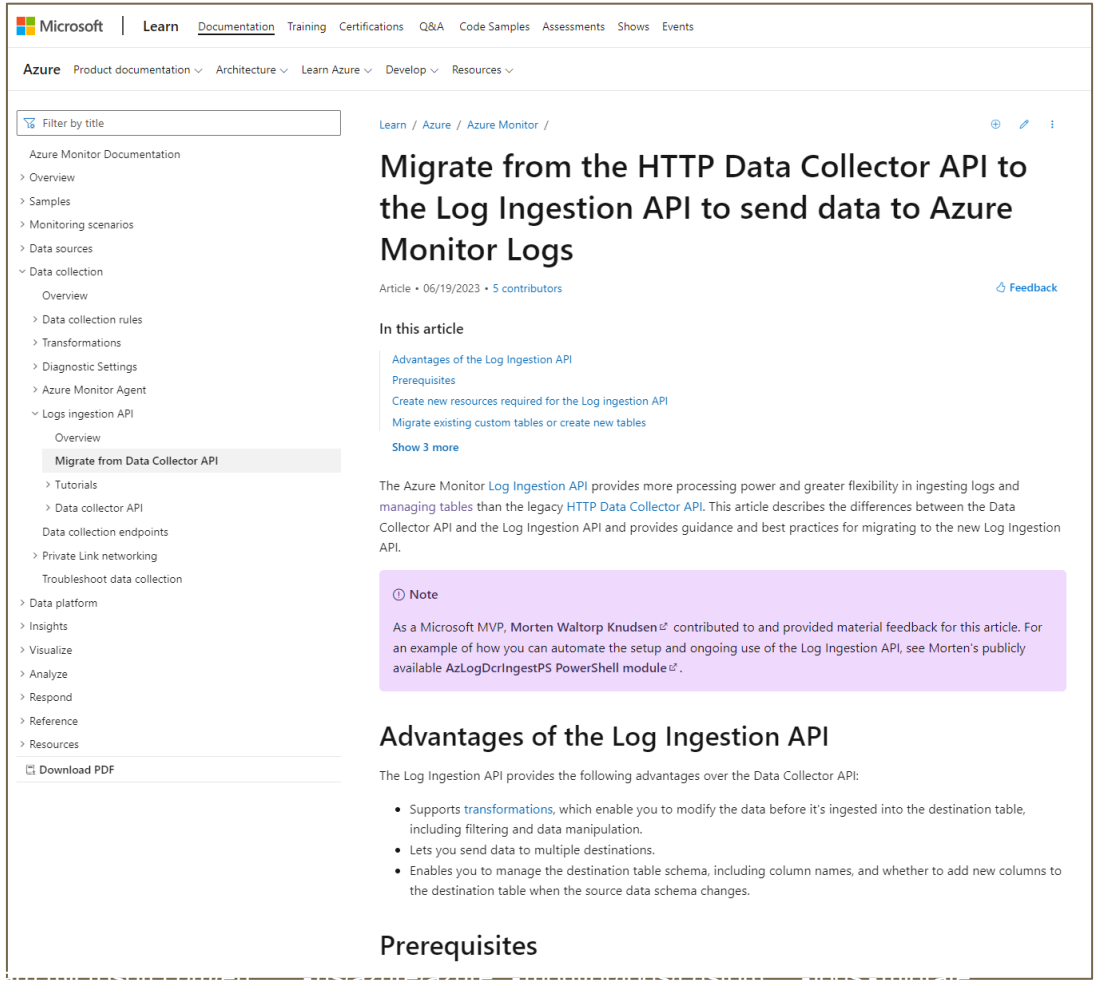
Video 8:30 min

Migrate existing table to DataCollectionRules -format

- Re-use existing table name
- Naming – 2 options
- Add new naming convention to columns + make transformation into old columns
- Re-use old naming
- “Big bang” per table (not possible to revert back)
- After migration of table
- Sending using HTTP Data Collector with SAME schema is allowed – data will be accepted.
- No schema changes using old HTTP Data Collector method
- Schema-changes can only be made using DCR -based method

Video 6:00 min

Migration guide



Microsoft | Learn Documentation Training Certifications Q&A Code Samples Assessments Shows Events

Azure Product documentation Architecture Learn Azure Develop Resources

Filter by title

- Azure Monitor Documentation
- Overview
- Samples
- Monitoring scenarios
- Data sources
- Data collection
  - Overview
  - Data collection rules
  - Transformations
  - Diagnostic Settings
  - Azure Monitor Agent
  - Logs ingestion API
    - Overview
    - Migrate from Data Collector API
    - Tutorials
    - Data collector API
  - Data collection endpoints
  - Private Link networking
  - Troubleshoot data collection
- Data platform
- Insights
- Visualize
- Analyze
- Respond
- Reference
- Resources

Download PDF

Learn / Azure / Azure Monitor /

Migrate from the HTTP Data Collector API to the Log Ingestion API to send data to Azure Monitor Logs

Article • 06/19/2023 • 5 contributors Feedback

In this article

- Advantages of the Log Ingestion API
- Prerequisites
- Create new resources required for the Log ingestion API
- Migrate existing custom tables or create new tables

Show 3 more

The Azure Monitor Log Ingestion API provides more processing power and greater flexibility in ingesting logs and managing tables than the legacy HTTP Data Collector API. This article describes the differences between the Data Collector API and the Log Ingestion API and provides guidance and best practices for migrating to the new Log Ingestion API.

Note

As a Microsoft MVP, **Morten Waltorp Knudsen** contributed to and provided material feedback for this article. For an example of how you can automate the setup and ongoing use of the Log Ingestion API, see Morten's publicly available **AzLogDcrIngestPS PowerShell module**.

Advantages of the Log Ingestion API

The Log Ingestion API provides the following advantages over the Data Collector API:

- Supports transformations, which enable you to modify the data before it's ingested into the destination table, including filtering and data manipulation.
- Lets you send data to multiple destinations.
- Enables you to manage the destination table schema, including column names, and whether to add new columns to the destination table when the source data schema changes.

Prerequisites

Get Started - Detailed Documentation

https://github.com/KnudsenMorten/AzLogDcrIngestPS

Videos

I have provided 4 demos for you to try, but if you want to see it first using video, check out these videos:

Video 3m 19s - Running ClientInspector using commandline (normal mode)
Video 1m 40s - Automatic creation of 2 tables & DCRs (verbose mode)
Video 1m 37s - Automatic creation of 2 tables & DCRs (normal mode)
Video 1m 34s - See schema of DCR and table)
Video 2m 19s - Data manipulation
Video 1m 58s - Kusto queries against data
Video 3m 01s - Dashboards
Video 0m 48s - Sample usage of data - lookup agai
Video 7m 25s - Deployment via ClientInspector Dep

Download latest version

You can download latest version of AzLogDcrIngestP

Install AzLogDcrIngestPS from Powershell Gallery

```
install-module AzLogDcrIngestPS
```

Download AzLogDcrIngestPS module from this Gith

Quick links for more information


```
PS C:\Users\mok.2LINKIT> get-command -module azlogdcringestps

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Add-CollectionTimeToAllEntriesInArray              1.2.36     azlogdcringestps
Function        Add-ColumnDataToAllEntriesInArray                  1.2.36     azlogdcringestps
Function        Build-DataArrayToAlignWithSchema                   1.2.36     azlogdcringestps
Function        CheckCreateUpdate-TableDcr-Structure               1.2.36     azlogdcringestps
Function        Convert-CimArrayToObjectFixStructure               1.2.36     azlogdcringestps
Function        Convert-PSArrayToObjectFixStructure                1.2.36     azlogdcringestps
Function        CreateUpdate-AzDataCollectionRuleLogIngestCusto... 1.2.36     azlogdcringestps
Function        CreateUpdate-AzLogAnalyticsCustomLogTableDcr       1.2.36     azlogdcringestps
Function        Delete-AzDataCollectionRules                       1.2.36     azlogdcringestps
Function        Delete-AzLogAnalyticsCustomLogTables               1.2.36     azlogdcringestps
Function        Filter-ObjectExcludeProperty                       1.2.36     azlogdcringestps
Function        Get-AzAccessTokenManagement                        1.2.36     azlogdcringestps
Function        Get-AzDataCollectionRuleTransformKql               1.2.36     azlogdcringestps
Fun
```

```
PS C:\Users\mok.2LINKIT> get-help Add-CollectionTimeToAllEntriesInArray -full

NAME
    Add-CollectionTimeToAllEntriesInArray

SYNOPSIS
    Add property CollectionTime (based on current time) to all entries on the object


SYNTAX
    Add-CollectionTimeToAllEntriesInArray [-Data] <Array> [<CommonParameters>]


DESCRIPTION
    Gives capability to do proper searching in queries to find latest set of records with same collection ti
    Time Generated cannot be used when you are sending data in batches, as TimeGenerated will change
    An example where this is important is a complete list of applications for a computer. We want all applic
    ions to
    show up when queriying for the latest data


PARAMETERS
    -Data <Array>
        Object to modify

        Required?                    true
        Position?                    1
        Default value
        Accept pipeline input?       false
        Accept wildcard characters?  false

    <CommonParameters>
        This cmdlet supports the common parameters: Verbose, Debug,
        ErrorAction, ErrorVariable, WarningAction, WarningVariable,
        OutBuffer, PipelineVariable, and OutVariable. For more information, see
        about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

INPUTS
    None. You cannot pipe objects


OUTPUTS
    Updated object with CollectionTime
```

KnudsenMorten / AzLogDcrIngestPS Public

Notifications | Fork 0 | Star 0

Code | Issues | Pull requests | Discussions | Actions | Projects | Security

main | AzLogDcrIngestPS / RELEASENOTES | Go to file

KnudsenMorten def — Latest commit bc2916f 2 days ago — History

1 contributor

29 lines (18 sloc) | 1.35 KB — Raw | Blame

```
RELEASE NOTES
v1.2.38  Changed the logic around SchemaMode = Merge, so the schema in DCR will be based on LogAnalytics tabl

v1.2.37  Adding another check to SchemaMode = Merge, so it will support existing properties are changing type

v1.2.36  Bugfix

v1.2.35  Changing so $AzDcrLogIngestServicePrincipalObjectId is not a mandatory parameter

v1.2.34  Bugfix - schemamode = merge wasn't updating table correctly

v1.2.33  Bugfix - signature was by mistake in source file (removed)

v1.2.32  Bugfix - skip TimeGenerated in DCR table schema when doing a merge. TimeGenerated only exist in tabl

v1.2.31  Added parameter SchemaMode = Merge/Overwrite for functions CheckCreateUpdate-TableDcr-Structure, Cre
                        SchemaMode = Merge (default)
                        It will do a merge/union of new properties and existing schema properties. DCR will i

                        SchemaMode = Overwrite
                        It will overwrite existing schema in DCR/table - based on source object schema
                        This parameter can be useful for separate overflow work
```

Step 1 - Get demo environment up and running.

Download the Powershell script Step1-Deployment-DemoEnvironment

Modify the SubscriptionId and TenantId in the header before running the deployment

The deployment-script will setup the following tasks:

- create Azure Resource Group for Azure LogAnalytics Workspace
- create Azure LogAnalytics Workspace
- create Azure App registration used for upload of data by demo-upload script
- create Azure service principal on Azure App
- create needed secret on Azure app
- create the Azure Resource Group for Azure Data Collection Endpoint (DCE) in same region as Azure LogAnalytics Workspace
- create the Azure Resource Group for Azure Data Collection Rules (DCR) in same region as Azure LogAnalytics Workspace
- create Azure Data Collection Endpoint (DCE) in same region as Azure LogAnalytics Workspace
- delegate permissions for Azure App on LogAnalytics workspace
- delegate permissions for Azure App on Azure Resource Group for Azure Data Collection Rules (DCR)
- delegate permissions for Azure App on Azure Resource Group for Azure Data Collection Endpoints (DCE)

Step 2 - Adjust the demo-script with the needed variables (sample below).

Demo-script can also be found here

```
##########################################
# VARIABLES
##########################################

<# ----- onboarding lines ----- BEGIN #>

$TenantId                               = "xxxxxxxxxxxxf63-9a77-ec94786b7c9e"
$LogIngestAppId                         = "xxxxxxxxxxxx-b45b-fe5e78392285"
$LogIngestAppSecret                     = "xxxxxxxxxxxx_NJFrBH_o-QdNR1Ga.T"

$LogAnalyticsWorkspaceResourceId        = "/subscriptions/xxxxxx/resourceGroups/rg-logworkspaces-client

$DceName                                = "dce-log-management-client-demo1-t"
$AzDcrResourceGroup                     = "rg-dcr-log-management-client-demo1-t"
$AzDcrSetLogIngestApiAppPermissionsDcrLevel  = $false

$AzDcrPrefix                            = "clt" # used to make it easy to find the DCRs when searching

# Used so schema changes will only happen on reference machines and not from other machines, if AzLogDcrInges
$AzLogDcrTableCreateFromReferenceMachine     = @()   # you will add your machine like @("MyDeviceName")
$AzLogDcrTableCreateFromAnyMachine           = $true # should be $false when you are ready for production. I

$global:Verbose                         = $true  # can be removed from script and added as parameter (

<# ----- onboarding lines ----- END  #>
```

How to get started ? (demo)

Step 3 - Run demos

You can now run the different sections in the script and see the demos. The demos will use most functions in AzLogDcrIngestPS

Start by running lines 1-275, which will load the initial header and build variables

Demo #1 will demonstrate data manipulation + show schema content

Demo #2 will demonstrate collection data + create LogAnalytics table + DCR + send data

Demo #3 will demonstrate collection of data, remove unnecessary data-properties, create schema with modified structure

Demo #4 will demonstrate schema change of existing table

NOTE:
Have patience :-)

Making schema changes + creating new pipelines will require 10-15 min delays right now. Data WILL come - have patience.

When the DCR + table + schema is in place, normal upload of data will happen very fast afterwords.

I have outlined the things to notice during the demos - run the lines one by one (sample below)

Thank You for today ☺

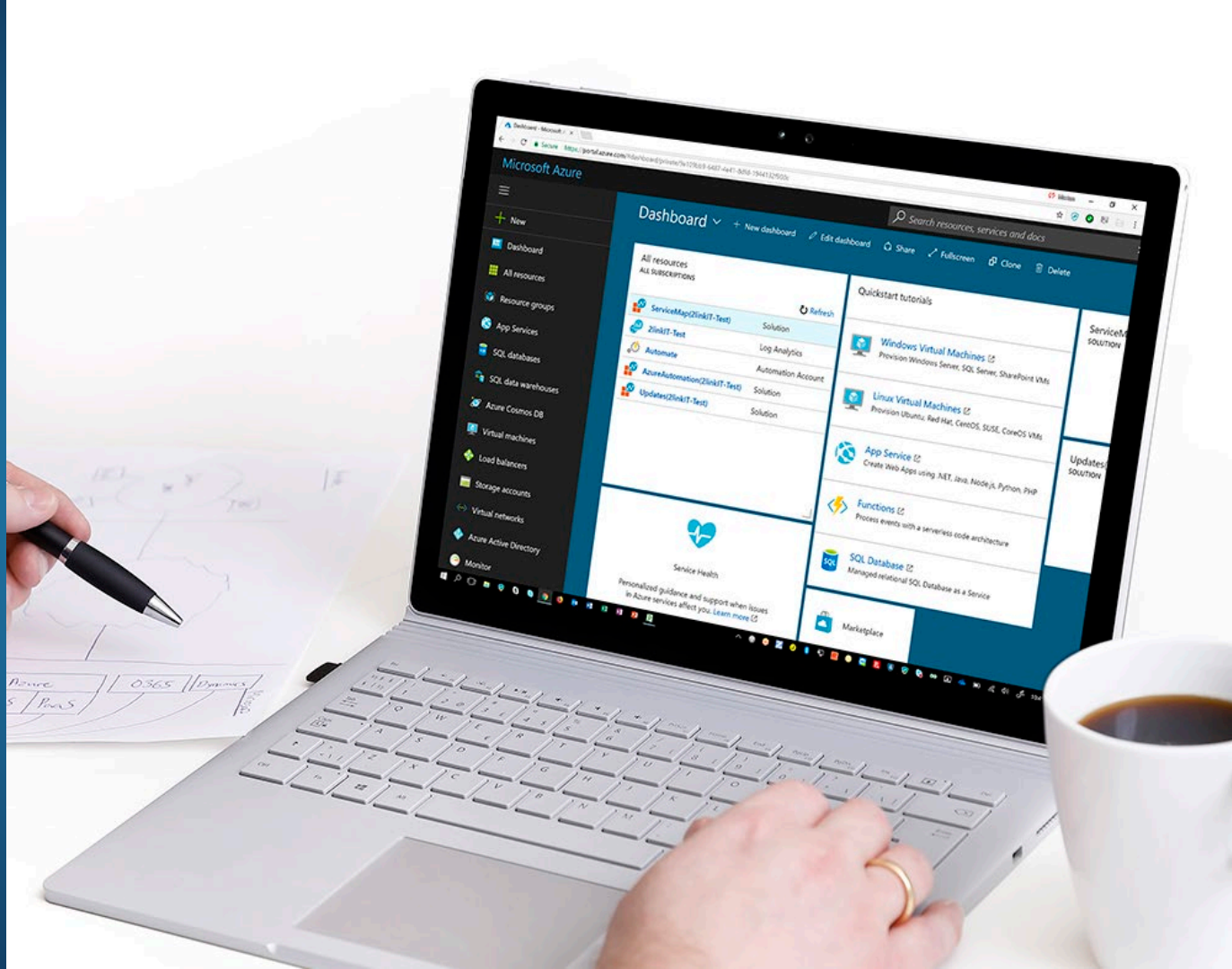
- Blog: https://mortenknudsen.net/
- Mail: mok@mortenknudsen.net | mok@2linkit.net
- LinkedIn: https://www.linkedin.com/in/mortenwaltorpknudsen/
- GitHub: https://github.com/KnudsenMorten
- Twitter: https://twitter.com/knudsenmortendk

This Presentation

Topic	Link
ClientInspector	https://github.com/KnudsenMorten/ClientInspectorV2
ClientInspector DeploymentKit	https://github.com/KnudsenMorten/ClientInspectorV2-DeploymentKit
Powershell module AzLogDcrIngestPS	https://github.com/KnudsenMorten/AzLogDcrIngestPS
LogHub (AzLogDcrIngestPSLogHub)	https://github.com/KnudsenMorten/AzLogDcrIngestPSLogHub
Deep-dive about Azure Data Collection Rules (DCRs)	https://github.com/KnudsenMorten/AzLogDcrIngestPS#deep-dive-about-azure-data-collection-rules-dcrs
Deep-dive about Log Ingestion API	https://github.com/KnudsenMorten/AzLogDcrIngestPS#deep-dive-about-log-ingestion-api

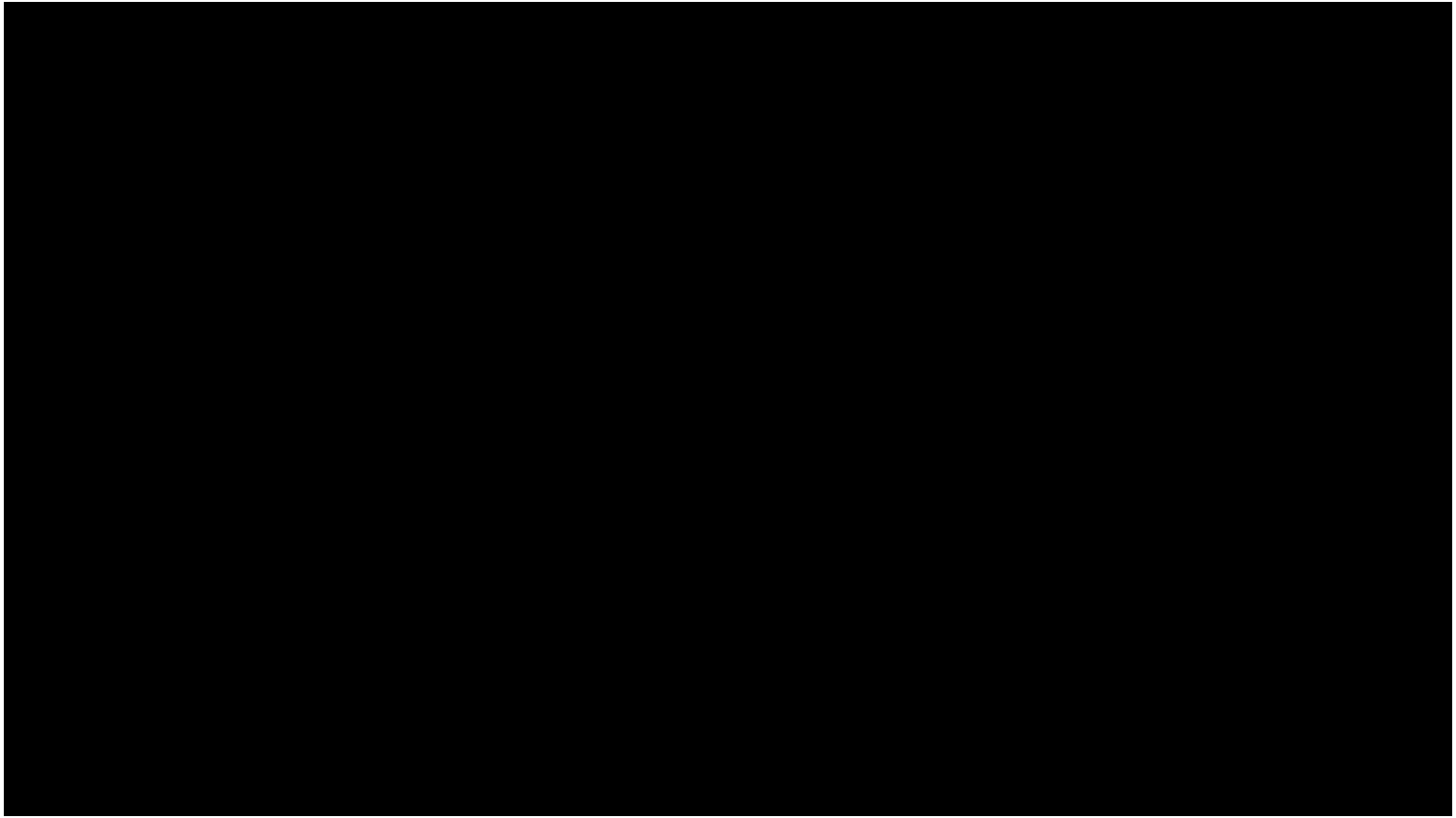
More deep -dive information, if people are interested

(not part of main presentation)



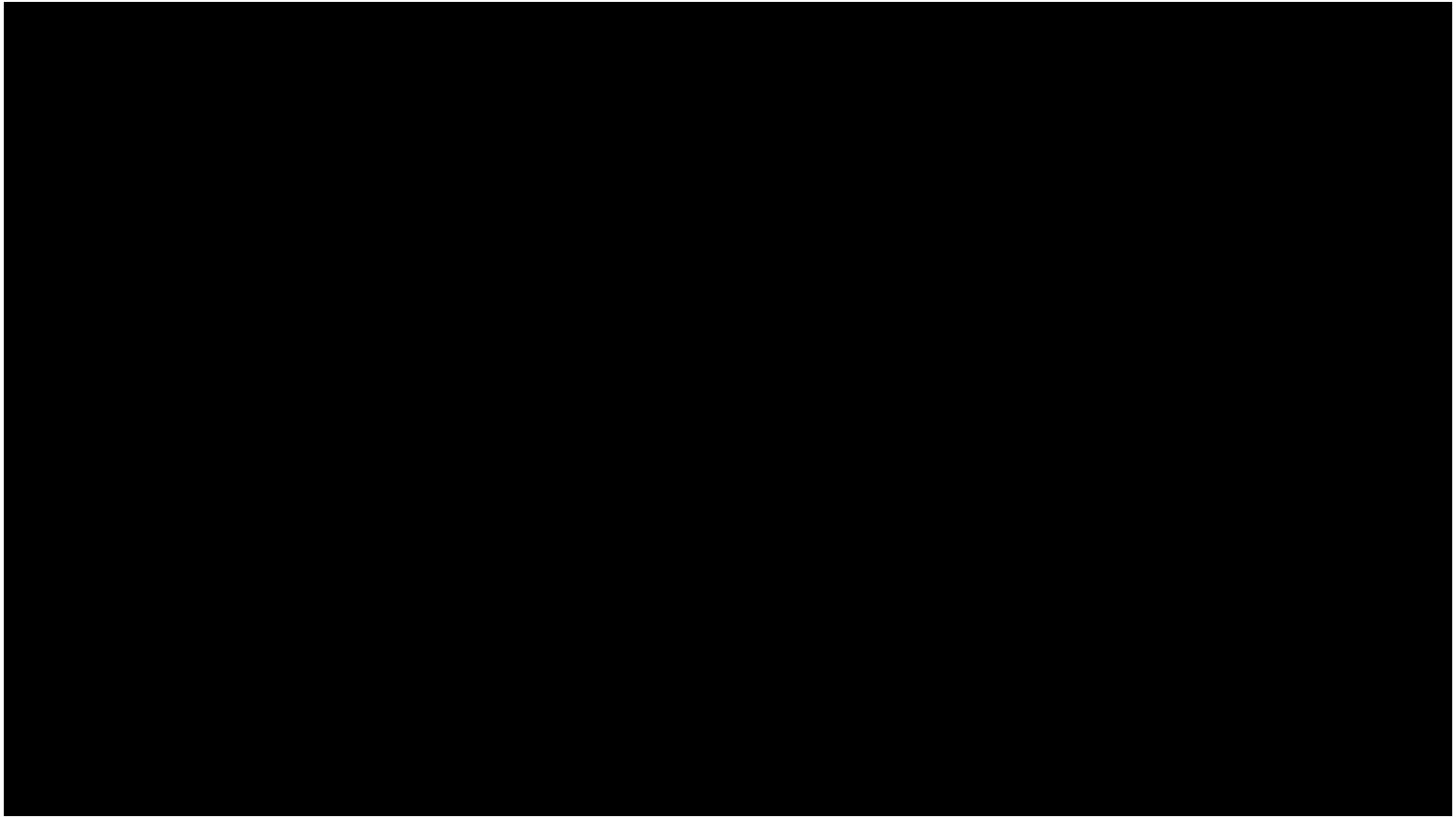
Side-by-Side Migration using AzLogDcrIngestPS

Video
(8:30)



Migrate existing table to DataCollectionRules-format

Video
(6:00)



Migrate existing table to DataCollectionRules -format

(Step 1/5) - Migration of existing table to V2-format (DCR-based)

```powershell
$Headers = Get-AzAccessTokenManagement -AzAppId $LogIngestAppId `
                                       -AzAppSecret $LogIngestAppSecret `
                                       -TenantId $TenantId `
                                       -Verbose:$Verbose

# Get existing LA table info
$TableUrl = "https://management.azure.com" + $LogAnalyticsWorkspaceResourceId + "/tables?api-version=2021-12-01-preview"
$tbl = invoke-restmethod -UseBasicParsing -Uri $TableUrl -Method GET -Headers $Headers
$res = $tbl.value.properties | Where-Object { $_.schema.name -like "*$($DemoNumber)*" }
$res.schema
$res.schema.columns

# Migrate table
$Uri         = "https://management.azure.com" + $LogAnalyticsWorkspaceResourceId + "/tables/$($TableName)_CL/migrate?api-version=2021-12-01-preview"
$Response    = invoke-webrequest -UseBasicParsing -Method POST -Uri $Uri -Headers $Headers


# Get existing LA table info
$TableUrl = "https://management.azure.com" + $LogAnalyticsWorkspaceResourceId + "/tables?api-version=2021-12-01-preview"
$tbl = invoke-restmethod -UseBasicParsing -Uri $TableUrl -Method GET -Headers $Headers
$res = $tbl.value.properties | Where-Object { $_.schema.name -like "*$($DemoNumber)*" }
$res.schema
$res.schema.columns
```

```
tableSubType            : Classic
name                    : V1CustomTable2997_CL
tableType               : CustomLog
```

```
tableSubType            : DataCollectionRuleBased
name                    : V1CustomTable2997_CL
tableType               : CustomLog
```

Migrate existing table to DataCollectionRules -format

(Step 2/5) - Add new table structure/naming

```powershell
#----------------------------------------
# (M1) Data to Upload (sample)
#----------------------------------------

$DataVariable = @()

$item = New-Object PSObject
$item | Add-Member -type NoteProperty -Name 'Computer' -Value $Env:ComputerName
$item | Add-Member -type NoteProperty -Name 'ColumnString' -Value 'StringText'
$item | Add-Member -type NoteProperty -Name 'ColumnDate' -Value (Get-date)
$item | Add-Member -type NoteProperty -Name 'ColumnNumber' -Value (Get-random -max 10000)
#$item | Add-Member -type NoteProperty -Name 'ColumnStringExtra' -Value "Extra"

$DataVariable += $item


#-----------------------------------------------------------------------------------------
# (M2) Get info about DCEs & DCRs
#-----------------------------------------------------------------------------------------

# building global variable with all DCEs, which can be viewed by Log Ingestion app
$global:AzDceDetails = Get-AzDceListAll -AzAppId $LogIngestAppId -AzAppSecret $LogIngestAppSecret -TenantId $TenantId -Verbose:$Verbose

# building global variable with all DCRs, which can be viewed by Log Ingestion app
$global:AzDcrDetails = Get-AzDcrListAll -AzAppId $LogIngestAppId -AzAppSecret $LogIngestAppSecret -TenantId $TenantId -Verbose:$Verbose
```

Migrate existing table to DataCollectionRules -format

(Step 3/5) - Add new table structure/naming

```
#-----------------------------------------
# (M3) Data Manipulation
#-----------------------------------------

# add CollectionTime to existing array
$DataVariable = Add-CollectionTimeToAllEntriesInArray -Data $DataVariable -Verbose:$Verbose

# add Computer, ComputerFqdn & UserLoggedOn info to existing array
$DataVariable = Add-ColumnDataToAllEntriesInArray -Data $DataVariable -Column1Name Computer -Column1Data $Env:ComputerName -Column2Name
ComputerFqdn -Column2Data $DnsName -Column3Name UserLoggedOn -Column3Data $UserLoggedOn -Verbose:$Verbose

# Validating/fixing schema data structure of source data
$DataVariable = ValidateFix-AzLogAnalyticsTableSchemaColumnNames -Data $DataVariable -Verbose:$Verbose

# Aligning data structure with schema (requirement for DCR)
$DataVariable = Build-DataArrayToAlignWithSchema -Data $DataVariable -Verbose:$Verbose
```

Migrate existing table to DataCollectionRules -format
(Step 4/5) - Add new table structure/naming

```
#-------------------------------------------------------------------------------------------
# (M4) Create/Update Schema for LogAnalytics Table & Data Collection Rule schema
#-------------------------------------------------------------------------------------------

CheckCreateUpdate-TableDcr-Structure -AzLogWorkspaceResourceId $LogAnalyticsWorkspaceResourceId -SchemaMode Migrate `
                                     -AzAppId $LogIngestAppId -AzAppSecret $LogIngestAppSecret -TenantId $TenantId -Verbose:$Verbose `
                                     -DceName $DceName -DcrName $DcrName -DcrResourceGroup $AzDcrResourceGroup `
                                     -TableName $TableName -Data $DataVariable `
                                     -LogIngestServicePricipleObjectId $AzDcrLogIngestServicePrincipalObjectId `
                                     -AzDcrSetLogIngestApiAppPermissionsDcrLevel $AzDcrSetLogIngestApiAppPermissionsDcrLevel `
                                     -AzLogDcrTableCreateFromAnyMachine $AzLogDcrTableCreateFromAnyMachine `
                                     -AzLogDcrTableCreateFromReferenceMachine $AzLogDcrTableCreateFromReferenceMachine
```

LogAnalytics table schema will be extended with any new data (merge)
DCR will be created with schema based on (updated) LogAnalytics table

Table + DCR is now ready to accept new data !

Migrate existing table to DataCollectionRules -format

(Step 5/5) - Transformation

```
$transformKql = "source | extend TimeGenerated = now(), ColumnDate_value_t = ColumnDate, ColumnString_s = ColumnString"

$DcrResourceId = ($global:AzDcrDetails | Where-Object { $_.name -eq $DcrName }).id
Update-AzDataCollectionRuleTransformKql -DcrResourceId $DcrResourceId -transformKql $transformKql -Verbose:$Verbose

Get-AzDataCollectionRuleTransformKql -DcrResourceId $DcrResourceId
```

Migrate existing table to DataCollectionRules -format

(Step 1/5) - Migration of existing table to V2-format (DCR-based)

```powershell
$Headers = Get-AzAccessTokenManagement -AzAppId $LogIngestAppId `
                                       -AzAppSecret $LogIngestAppSecret `
                                       -TenantId $TenantId `
                                       -Verbose:$Verbose

# Get existing LA table info
$TableUrl = "https://management.azure.com" + $LogAnalyticsWorkspaceResourceId + "/tables?api-version=2021-12-01-preview"
$tbl = invoke-restmethod -UseBasicParsing -Uri $TableUrl -Method GET -Headers $Headers
$res = $tbl.value.properties | Where-Object { $_.schema.name -like "*$($DemoNumber)*" }
$res.schema
$res.schema.columns

# Migrate table
$Uri         = "https://management.azure.com" + $LogAnalyticsWorkspaceResourceId + "/tables/$($TableName)_CL/migrate?api-version=2021-12-01-preview"
$Response    = invoke-webrequest -UseBasicParsing -Method POST -Uri $Uri -Headers $Headers


# Get existing LA table info
$TableUrl = "https://management.azure.com" + $LogAnalyticsWorkspaceResourceId + "/tables?api-version=2021-12-01-preview"
$tbl = invoke-restmethod -UseBasicParsing -Uri $TableUrl -Method GET -Headers $Headers
$res = $tbl.value.properties | Where-Object { $_.schema.name -like "*$($DemoNumber)*" }
$res.schema
$res.schema.columns
```

```
tableSubType            : Classic
name                    : V1CustomTable2997_CL
tableType               : CustomLog
```

```
tableSubType            : DataCollectionRuleBased
name                    : V1CustomTable2997_CL
tableType               : CustomLog
```

Migrate existing table to DataCollectionRules -format

(Step 2/5) - Add new table structure/naming

```powershell
#----------------------------------------
# (M1) Data to Upload (sample)
#----------------------------------------

$DataVariable = @()

$item = New-Object PSObject
$item | Add-Member -type NoteProperty -Name 'Computer' -Value $Env:ComputerName
$item | Add-Member -type NoteProperty -Name 'ColumnString' -Value 'StringText'
$item | Add-Member -type NoteProperty -Name 'ColumnDate' -Value (Get-date)
$item | Add-Member -type NoteProperty -Name 'ColumnNumber' -Value (Get-random -max 10000)
#$item | Add-Member -type NoteProperty -Name 'ColumnStringExtra' -Value "Extra"

$DataVariable += $item


#-----------------------------------------------------------------------------------------
# (M2) Get info about DCEs & DCRs
#-----------------------------------------------------------------------------------------

# building global variable with all DCEs, which can be viewed by Log Ingestion app
$global:AzDceDetails = Get-AzDceListAll -AzAppId $LogIngestAppId -AzAppSecret $LogIngestAppSecret -TenantId $TenantId -Verbose:$Verbose

# building global variable with all DCRs, which can be viewed by Log Ingestion app
$global:AzDcrDetails = Get-AzDcrListAll -AzAppId $LogIngestAppId -AzAppSecret $LogIngestAppSecret -TenantId $TenantId -Verbose:$Verbose
```

Migrate existing table to DataCollectionRules -format

(Step 3/5) - Add new table structure/naming

```
#-----------------------------------------
# (M3) Data Manipulation
#-----------------------------------------

# add CollectionTime to existing array
$DataVariable = Add-CollectionTimeToAllEntriesInArray -Data $DataVariable -Verbose:$Verbose

# add Computer, ComputerFqdn & UserLoggedOn info to existing array
$DataVariable = Add-ColumnDataToAllEntriesInArray -Data $DataVariable -Column1Name Computer -Column1Data $Env:ComputerName -Column2Name
ComputerFqdn -Column2Data $DnsName -Column3Name UserLoggedOn -Column3Data $UserLoggedOn -Verbose:$Verbose

# Validating/fixing schema data structure of source data
$DataVariable = ValidateFix-AzLogAnalyticsTableSchemaColumnNames -Data $DataVariable -Verbose:$Verbose

# Aligning data structure with schema (requirement for DCR)
$DataVariable = Build-DataArrayToAlignWithSchema -Data $DataVariable -Verbose:$Verbose
```

Migrate existing table to DataCollectionRules -format

(Step 4/5) - Add new table structure/naming

```
#-------------------------------------------------------------------------------------------
# (M4) Create/Update Schema for LogAnalytics Table & Data Collection Rule schema
#-------------------------------------------------------------------------------------------

CheckCreateUpdate-TableDcr-Structure -AzLogWorkspaceResourceId $LogAnalyticsWorkspaceResourceId -SchemaMode Migrate `
                                     -AzAppId $LogIngestAppId -AzAppSecret $LogIngestAppSecret -TenantId $TenantId -Verbose:$Verbose `
                                     -DceName $DceName -DcrName $DcrName -DcrResourceGroup $AzDcrResourceGroup `
                                     -TableName $TableName -Data $DataVariable `
                                     -LogIngestServicePricipleObjectId $AzDcrLogIngestServicePrincipalObjectId `
                                     -AzDcrSetLogIngestApiAppPermissionsDcrLevel $AzDcrSetLogIngestApiAppPermissionsDcrLevel `
                                     -AzLogDcrTableCreateFromAnyMachine $AzLogDcrTableCreateFromAnyMachine `
                                     -AzLogDcrTableCreateFromReferenceMachine $AzLogDcrTableCreateFromReferenceMachine
```

LogAnalytics table schema will be extended with any new data (merge)
DCR will be created with schema based on (updated) LogAnalytics table

Table + DCR is now ready to accept new data !

Migrate existing table to DataCollectionRules -format
(Step 5/5) - Transformation

```
$transformKql = "source | extend TimeGenerated = now(), ColumnDate_value_t = ColumnDate, ColumnString_s = ColumnString"

$DcrResourceId = ($global:AzDcrDetails | Where-Object { $_.name -eq $DcrName }).id
Update-AzDataCollectionRuleTransformKql -DcrResourceId $DcrResourceId -transformKql $transformKql -Verbose:$Verbose

Get-AzDataCollectionRuleTransformKql -DcrResourceId $DcrResourceId
```

ClientInspector - What is being collected ?

- User Logged On to Client
- Computer information
  - bios, processor, hardware info, Windows OS info, OS information, last restart
- Installed applications
  - both using WMI and registry
- Antivirus Security Center from Windows
  - default antivirus, state , configuration
- Microsoft Defender Antivirus
  - all settings including ASR,exclusions , realtime protection , etc
- Office - version, update channel config , SKUs
- VPN client - version, product
- LAPS – version
- Admin By Request (3rd party) – version
- Bitlocker - configuration
- Windows Update
  - last result (when), windows update source information (where), pending updates , last installations (what)
- Eventlog
  - look for specific events including logon events, blue screens , etc.
- Network adapters - configuration , installed adapters
- IP information for all adapters
- Local administrators group membership
- Windows firewall
  - settings for all 3 modes
- Group Policy - last refresh
- TPM information
  - relevant to detect machines with/ without TPM

```
PS  get-command -module AzLogDcrIngestPS

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Add-CollectionTimeToAllEntriesInArray              1.1.17     AzLogDcrIngestPS
Function        Add-ColumnDataToAllEntriesInArray                  1.1.17     AzLogDcrIngestPS
Function        Build-DataArrayToAlignWithSchema                   1.1.17     AzLogDcrIngestPS
Function        CheckCreateUpdate-TableDcr-Structure               1.1.17     AzLogDcrIngestPS
Function        Convert-CimArrayToObjectFixStructure               1.1.17     AzLogDcrIngestPS
Function        Convert-PSArrayToObjectFixStructure                1.1.17     AzLogDcrIngestPS
Function        CreateUpdate-AzDataCollectionRuleLogIngestCusto... 1.1.17     AzLogDcrIngestPS
Function        CreateUpdate-AzLogAnalyticsCustomLogTableDcr       1.1.17     AzLogDcrIngestPS
Function        Delete-AzDataCollectionRules                       1.1.17     AzLogDcrIngestPS
Function        Delete-AzLogAnalyticsCustomLogTables               1.1.17     AzLogDcrIngestPS
Function        Filter-ObjectExcludeProperty                       1.1.17     AzLogDcrIngestPS
Function        Get-AzAccessTokenManagement                        1.1.17     AzLogDcrIngestPS
Function        Get-AzDceListAll                                   1.1.17     AzLogDcrIngestPS
Function        Get-AzDcrDceDetails                                1.1.17     AzLogDcrIngestPS
Function        Get-AzDcrListAll                                   1.1.17     AzLogDcrIngestPS
Function        Get-AzLogAnalyticsTableAzDataCollectionRuleStatus  1.1.17     AzLogDcrIngestPS
Function        Get-ObjectSchemaAsArray                            1.1.17     AzLogDcrIngestPS
Function        Get-ObjectSchemaAsHash                             1.1.17     AzLogDcrIngestPS
Function        Post-AzLogAnalyticsLogIngestCustomLogDcrDce        1.1.17     AzLogDcrIngestPS
Function        Post-AzLogAnalyticsLogIngestCustomLogDcrDce-Output 1.1.17     AzLogDcrIngestPS
Function        Update-AzDataCollectionRuleDceEndpoint             1.1.17     AzLogDcrIngestPS
Function        Update-AzDataCollectionRuleResetTransformKqlDef... 1.1.17     AzLogDcrIngestPS
Function        Update-AzDataCollectionRuleTransformKql            1.1.17     AzLogDcrIngestPS
Function        ValidateFix-AzLogAnalyticsTableSchemaColumnNames   1.1.17     AzLogDcrIngestPS
```

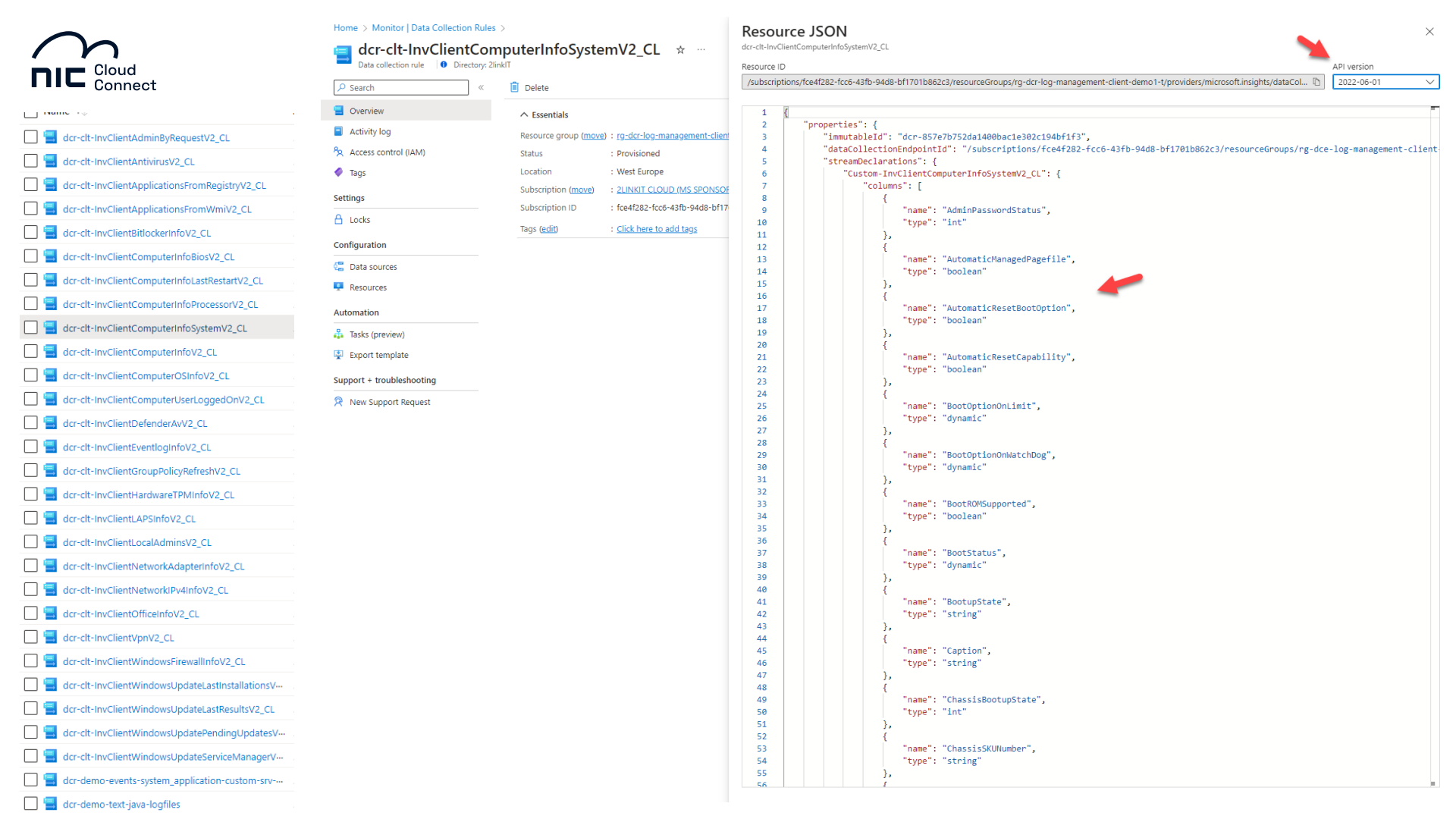
```
get-help Add-CollectionTimeToAllEntriesInArray -full

NAME
    Add-CollectionTimeToAllEntriesInArray

SYNOPSIS
    Add property CollectionTime (based on current time) to all entries on the object


SYNTAX
    Add-CollectionTimeToAllEntriesInArray [-Data] <Array> [<CommonParameters>]


DESCRIPTION
    Gives capability to do proper searching in queries to find latest set of records with same collection tim
    Time Generated cannot be used when you are sending data in batches, as TimeGenerated will change
    An example where this is important is a complete list of applications for a computer. We want all applica
    show up when queriying for the latest data


PARAMETERS
    -Data <Array>
        Object to modify

        Required?                    true
        Position?                    1
        Default value
        Accept pipeline input?       false
        Accept wildcard characters?  false

    <CommonParameters>
        This cmdlet supports the common parameters: Verbose, Debug,
        ErrorAction, ErrorVariable, WarningAction, WarningVariable,
        OutBuffer, PipelineVariable, and OutVariable. For more information, see
        about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

INPUTS
    None. You cannot pipe objects


OUTPUTS
    Updated object with CollectionTime


    -------------------------- EXAMPLE 1 --------------------------

    PS C:\>#-------------------------------------------------------------------------------------------------

    # Variables
    #-------------------------------------------------------------------------------------------------
    $Verbose                  = $true  # $true or $false

    #-------------------------------------------------------------------------------------------------
    # Collecting data (in)
    #-------------------------------------------------------------------------------------------------
    $DNSName                  = (Get-CimInstance win32_computersystem).DNSHostName +"." + (Get-CimInstance w
    $ComputerName             = (Get-CimInstance win32_computersystem).DNSHostName
```

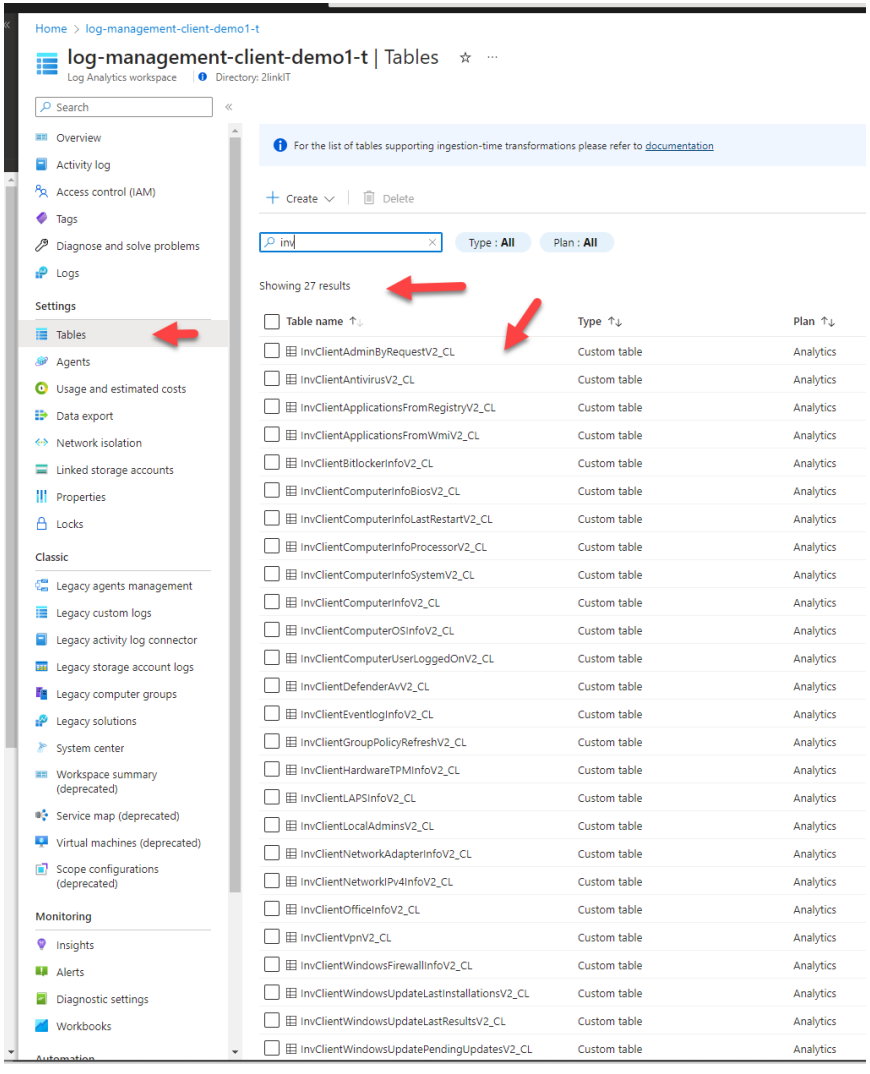
```
.\ClientInspector.ps1 -verbose:$false -function:download

ClientInspector | Inventory of Operational & Security-related information
Developed by Morten Knudsen, Microsoft MVP - for free community use

Downloading latest version of module AzLogDcrIngestPS from https://github.com/KnudsenMorten/CientInspectorV2
into local path D:\scripts\ClientInspectorV2
```

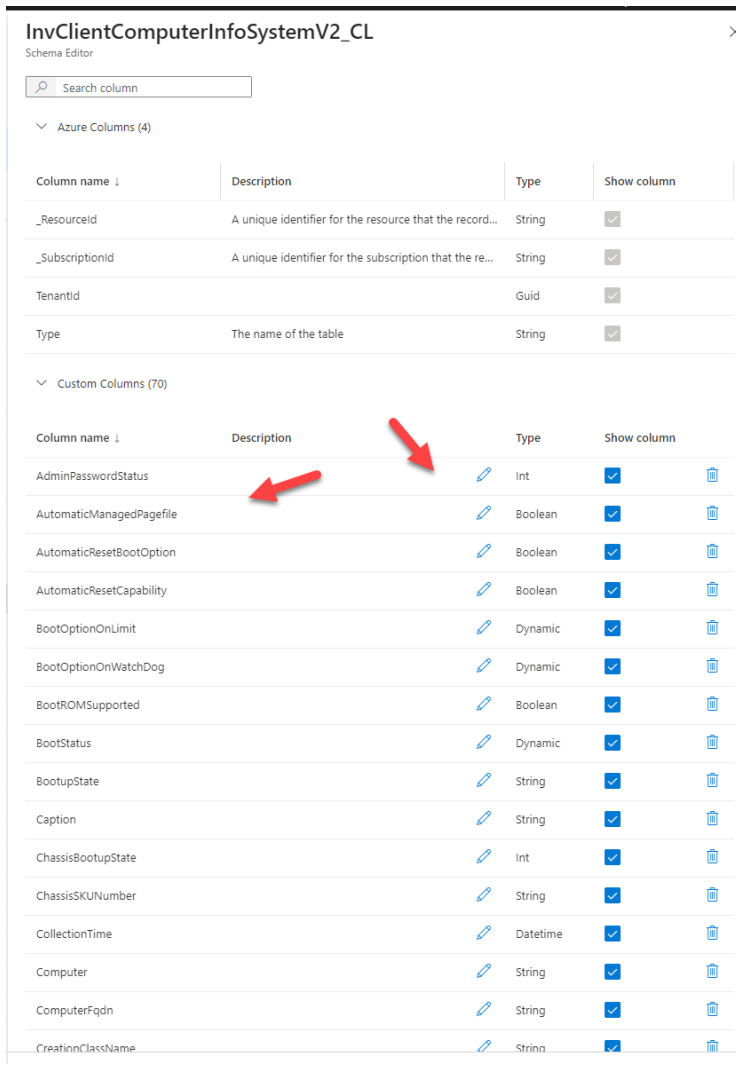
```
.\ClientInspector.ps1 -verbose:$false -function:localpath

ClientInspector | Inventory of Operational & Security-related information
Developed by Morten Knudsen, Microsoft MVP - for free community use

Using AzLogDcrIngestPS module from local path D:\scripts\ClientInspectorV2
```

```
.\ClientInspector.ps1 -verbose:$false -function:PSGallery -scope:CurrentUser

ClientInspector | Inventory of Operational & Security-related information
Developed by Morten Knudsen, Microsoft MVP - for free community use

Powershell module was not found !
Installing in scope currentuser .... Please Wait !
```

```
.\ClientInspector.ps1 -verbose:$false -function:PsGallery -scope:currentuser

ClientInspector | Inventory of Operational & Security-related information
Developed by Morten Knudsen, Microsoft MVP - for free community use

Checking latest version at PsGallery for AzLogDcrIngestPS module
OK - Running latest version
```

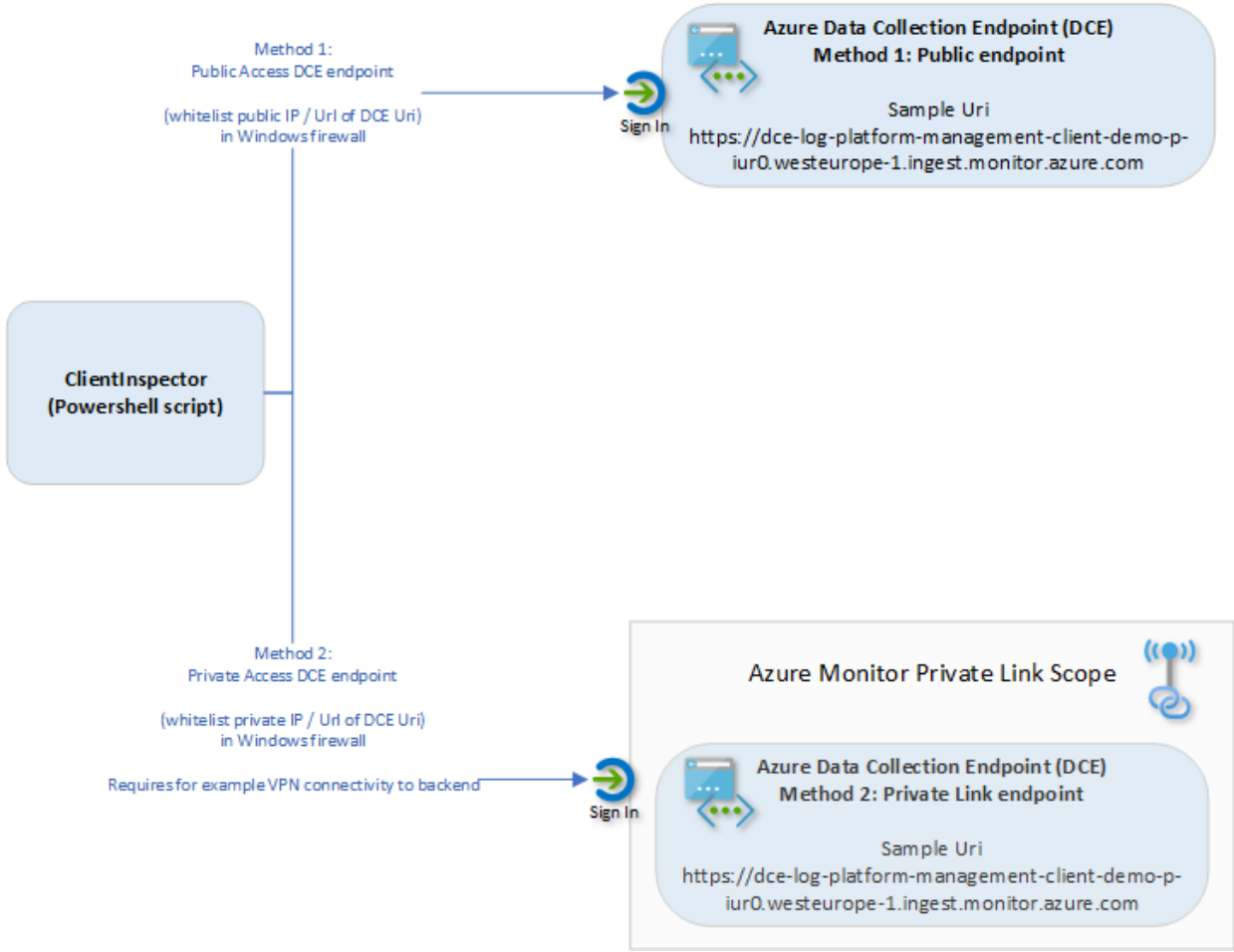

Cloud Connect
dcr-clt-InvClientAdminByRequestV2_CL
dcr-clt-InvClientAntivirusV2_CL
dcr-clt-InvClientApplicationsFromRegistryV2_CL
dcr-clt-InvClientApplicationsFromWmiV2_CL
dcr-clt-InvClientBitlockerInfoV2_CL
dcr-clt-InvClientComputerInfoBiosV2_CL
dcr-clt-InvClientComputerInfoLastRestartV2_CL
dcr-clt-InvClientComputerInfoProcessorV2_CL
dcr-clt-InvClientComputerInfoSystemV2_CL
dcr-clt-InvClientComputerInfoV2_CL
dcr-clt-InvClientComputerOSInfoV2_CL
dcr-clt-InvClientComputerUserLoggedOnV2_CL
dcr-clt-InvClientDefenderAvV2_CL
dcr-clt-InvClientEventlogInfoV2_CL
dcr-clt-InvClientGroupPolicyRefreshV2_CL
dcr-clt-InvClientHardwareTPMInfoV2_CL
dcr-clt-InvClientLAPSInfoV2_CL
dcr-clt-InvClientLocalAdminsV2_CL
dcr-clt-InvClientNetworkAdapterInfoV2_CL
dcr-clt-InvClientNetworkIPv4InfoV2_CL
dcr-clt-InvClientOfficeInfoV2_CL
dcr-clt-InvClientVpnV2_CL
dcr-clt-InvClientWindowsFirewallInfoV2_CL
dcr-clt-InvClientWindowsUpdateLastInstallationsV...
dcr-clt-InvClientWindowsUpdateLastResultsV2_CL
dcr-clt-InvClientWindowsUpdatePendingUpdatesV...
dcr-clt-InvClientWindowsUpdateServiceManagerV...
dcr-demo-events-system_application-custom-srv-...
dcr-demo-text-java-logfiles
Home > Monitor | Data Collection Rules >
dcr-clt-InvClientComputerInfoSystemV2_CL
Data collection rule
Directory: 2linkIT
Search
Delete
Overview
Activity log
Access control (IAM)
Tags
Settings
Locks
Configuration
Data sources
Resources
Automation
Tasks (preview)
Export template
Support + troubleshooting
New Support Request
Essentials
Resource group (move) : rg-dcr-log-management-client
Status : Provisioned
Location : West Europe
Subscription (move) : 2LINKIT CLOUD (MS SPONSOR
Subscription ID : fce4f282-fcc6-43fb-94d8-bf17
Tags (edit) : Click here to add tags
Resource JSON
dcr-clt-InvClientComputerInfoSystemV2_CL
Resource ID
/subscriptions/fce4f282-fcc6-43fb-94d8-bf1701b862c3/resourceGroups/rg-dcr-log-management-client-demo1-t/providers/microsoft.insights/dataCol...
API version
2022-06-01
{
"properties": {
"immutableId": "dcr-857e7b752da1400bac1e302c194bf1f3",
"dataCollectionEndpointId": "/subscriptions/fce4f282-fcc6-43fb-94d8-bf1701b862c3/resourceGroups/rg-dce-log-management-client
"streamDeclarations": {
"Custom-InvClientComputerInfoSystemV2_CL": {
"columns": [
{
"name": "AdminPasswordStatus",
"type": "int"
},
{
"name": "AutomaticManagedPagefile",
"type": "boolean"
},
{
"name": "AutomaticResetBootOption",
"type": "boolean"
},
{
"name": "AutomaticResetCapability",
"type": "boolean"
},
{
"name": "BootOptionOnLimit",
"type": "dynamic"
},
{
"name": "BootOptionOnWatchDog",
"type": "dynamic"
},
{
"name": "BootROMSupported",
"type": "boolean"
},
{
"name": "BootStatus",
"type": "dynamic"
},
{
"name": "BootupState",
"type": "string"
},
{
"name": "Caption",
"type": "string"
},
{
"name": "ChassisBootupState",
"type": "int"
},
{
"name": "ChassisSKUNumber",
"type": "string"
},

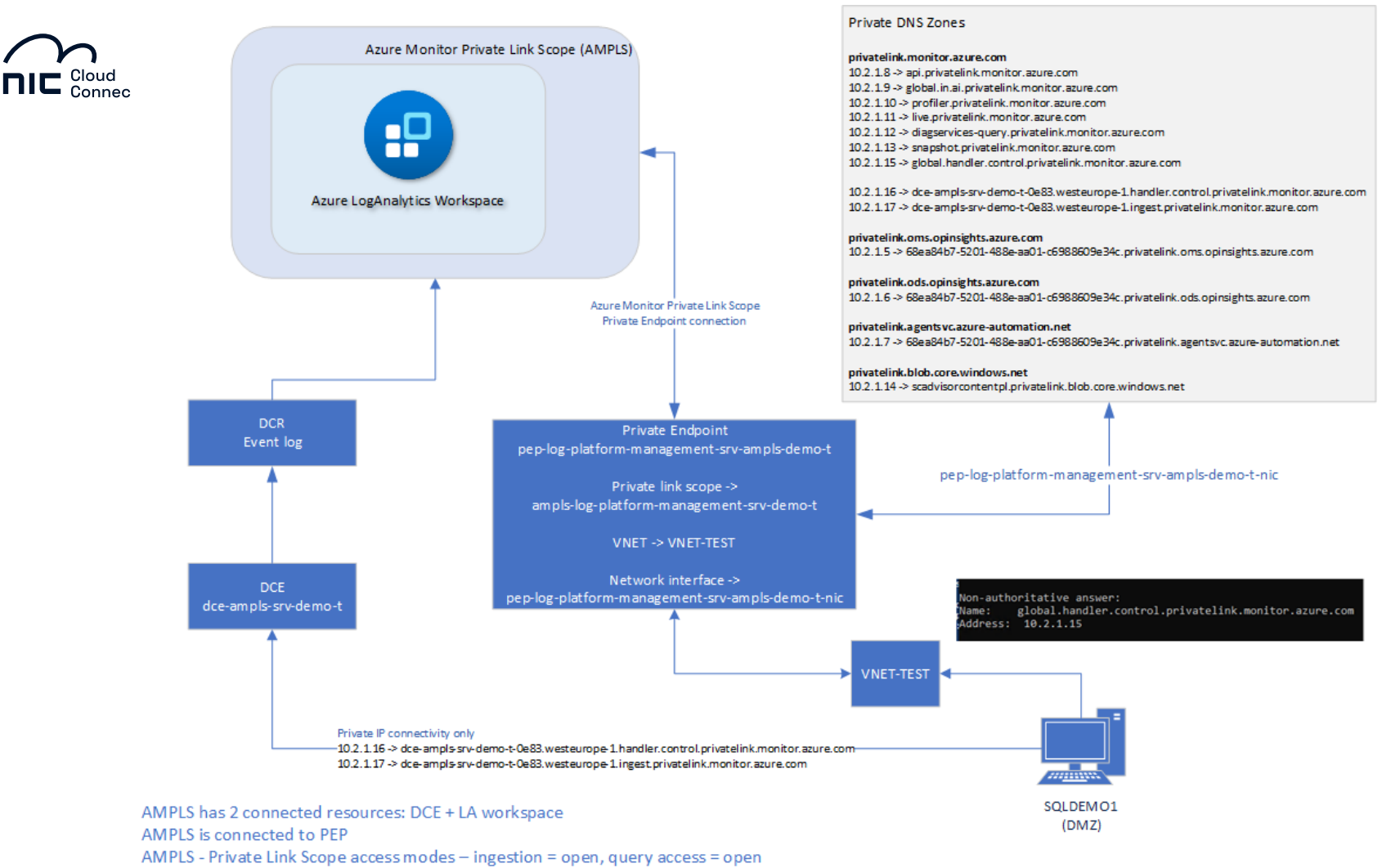

Home > log-management-client-demo1-t
log-management-client-demo1-t | Tables
Log Analytics workspace
Directory: 2linkIT
Search
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Logs
Settings
Tables
Agents
Usage and estimated costs
Data export
Network isolation
Linked storage accounts
Properties
Locks
Classic
Legacy agents management
Legacy custom logs
Legacy activity log connector
Legacy storage account logs
Legacy computer groups
Legacy solutions
System center
Workspace summary (deprecated)
Service map (deprecated)
Virtual machines (deprecated)
Scope configurations (deprecated)
Monitoring
Insights
Alerts
Diagnostic settings
Workbooks
Automation
For the list of tables supporting ingestion-time transformations please refer to documentation
Create
Delete
inv
Type : All
Plan : All
Showing 27 results
Table name
Type
Plan
InvClientAdminByRequestV2_CL Custom table Analytics
InvClientAntivirusV2_CL Custom table Analytics
InvClientApplicationsFromRegistryV2_CL Custom table Analytics
InvClientApplicationsFromWmiV2_CL Custom table Analytics
InvClientBitlockerInfoV2_CL Custom table Analytics
InvClientComputerInfoBiosV2_CL Custom table Analytics
InvClientComputerInfoLastRestartV2_CL Custom table Analytics
InvClientComputerInfoProcessorV2_CL Custom table Analytics
InvClientComputerInfoSystemV2_CL Custom table Analytics
InvClientComputerInfoV2_CL Custom table Analytics
InvClientComputerOSInfoV2_CL Custom table Analytics
InvClientComputerUserLoggedOnV2_CL Custom table Analytics
InvClientDefenderAVV2_CL Custom table Analytics
InvClientEventlogInfoV2_CL Custom table Analytics
InvClientGroupPolicyRefreshV2_CL Custom table Analytics
InvClientHardwareTPMInfoV2_CL Custom table Analytics
InvClientLAPSInfoV2_CL Custom table Analytics
InvClientLocalAdminsV2_CL Custom table Analytics
InvClientNetworkAdapterInfoV2_CL Custom table Analytics
InvClientNetworkIPv4InfoV2_CL Custom table Analytics
InvClientOfficeInfoV2_CL Custom table Analytics
InvClientVpnV2_CL Custom table Analytics
InvClientWindowsFirewallInfoV2_CL Custom table Analytics
InvClientWindowsUpdateLastInstallationsV2_CL Custom table Analytics
InvClientWindowsUpdateLastResultsV2_CL Custom table Analytics
InvClientWindowsUpdatePendingUpdatesV2_CL Custom table Analytics


InvClientComputerInfoSystemV2_CL
Schema Editor
Search column
Azure Columns (4)
Column name
Description
Type
Show column
_ResourceId A unique identifier for the resource that the record... String
_SubscriptionId A unique identifier for the subscription that the re... String
TenantId Guid
Type The name of the table String
Custom Columns (70)
Column name
Description
Type
Show column
AdminPasswordStatus Int
AutomaticManagedPagefile Boolean
AutomaticResetBootOption Boolean
AutomaticResetCapability Boolean
BootOptionOnLimit Dynamic
BootOptionOnWatchDog Dynamic
BootROMSupported Boolean
BootStatus Dynamic
BootupState String
Caption String
ChassisBootupState Int
ChassisSKUNumber String
CollectionTime Datetime
Computer String
ComputerFqdn String
CreationClassName String

Networking

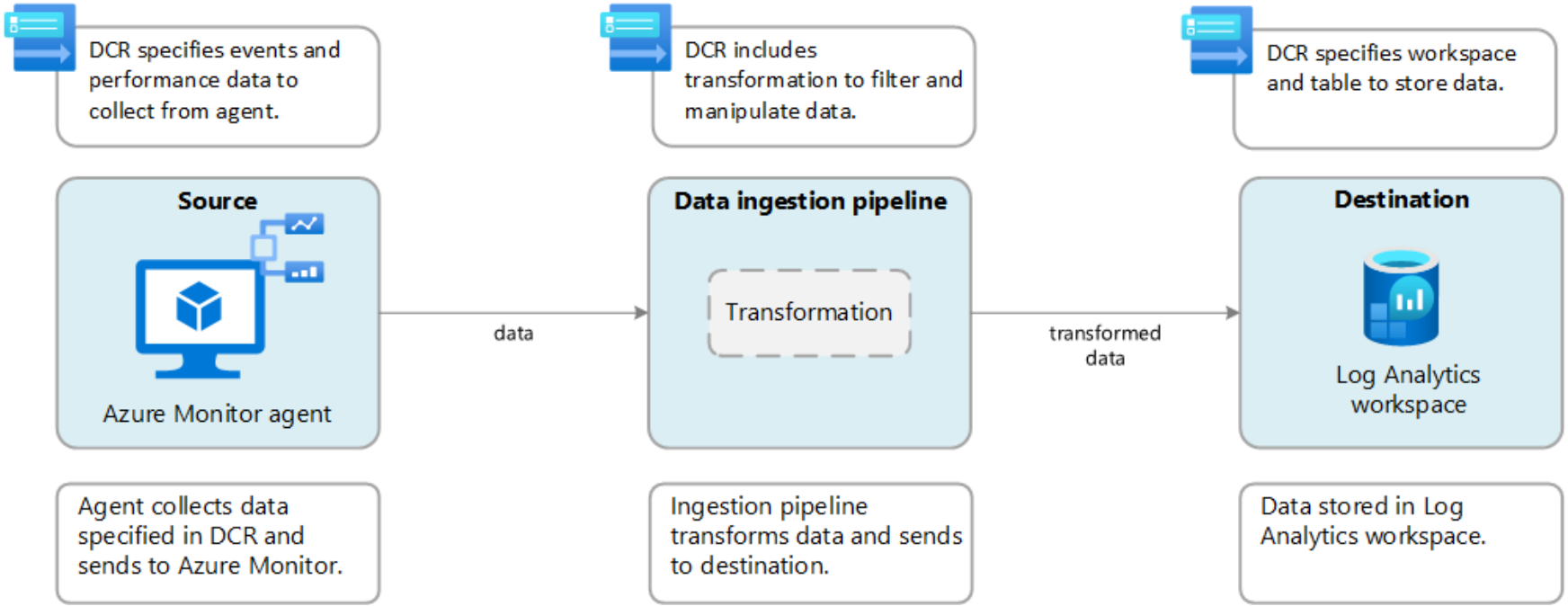
Client endpoint (REST api) sends data to DCE



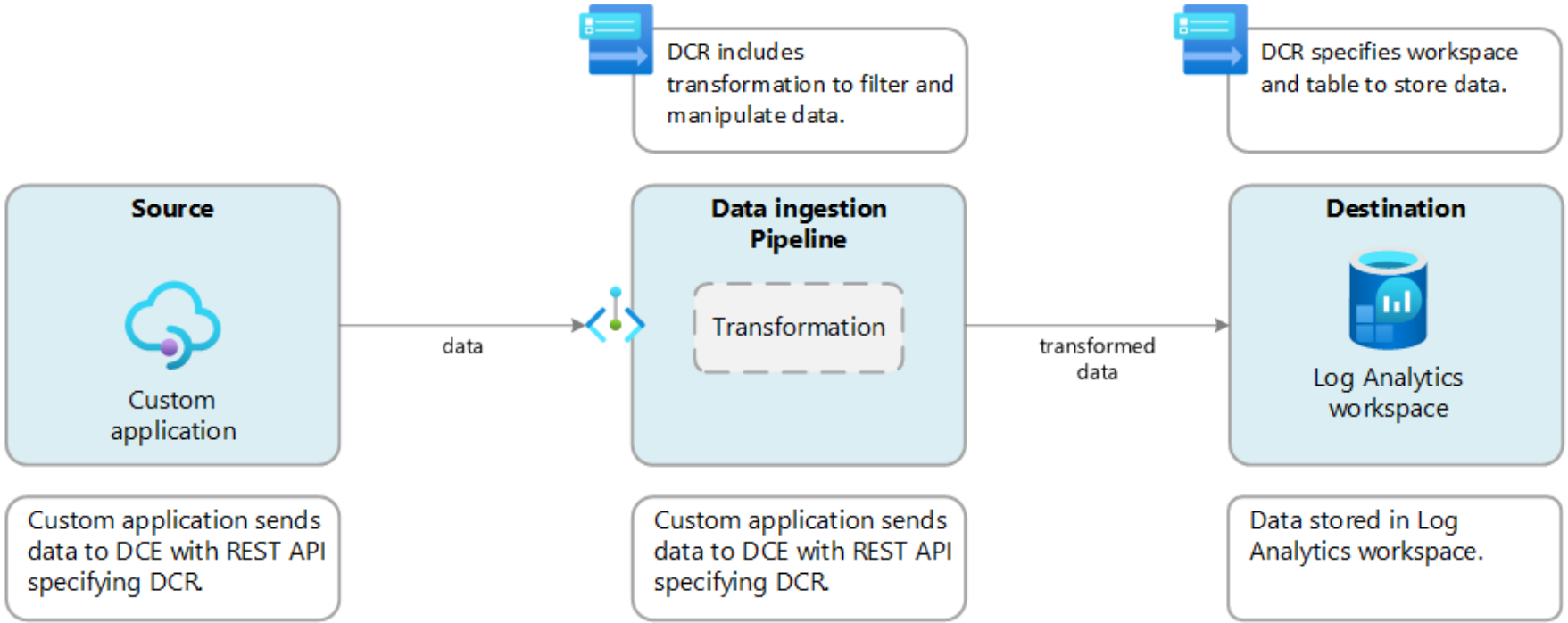


AMPLS has 2 connected resources: DCE + LA workspace
AMPLS is connected to PEP
AMPLS - Private Link Scope access modes – ingestion = open, query access = open

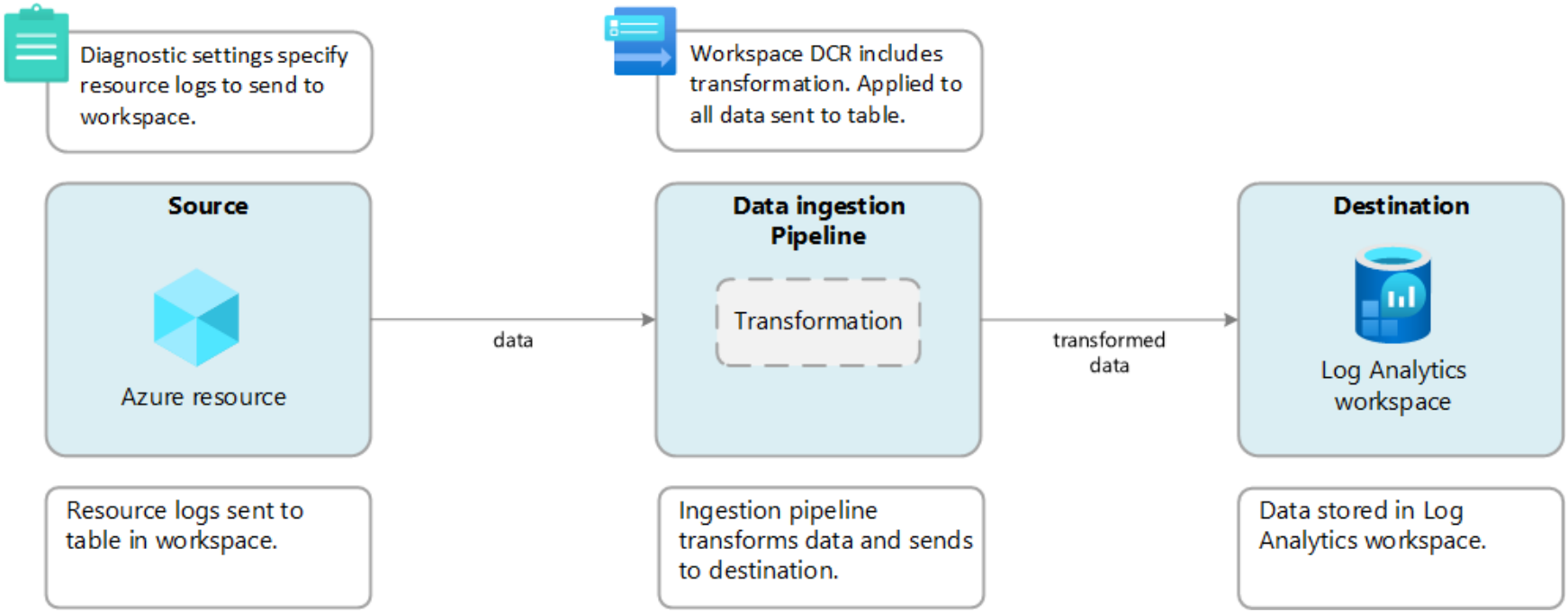
Transformation using AMA & DCR



Transformation using Log ingestion API, DCR & DCE



Workspace Transformation DCR



Collection Source	Technologies needed	Resource Association	Target (today)
Performance Eventlog Syslog	AMA / DCR	Required (DCR)	Standard table (LogAnalytics)
Text logs IIS logs Windows Firewall logs (preview)	AMA / DCR /DCE	Required (DCR)	Custom Log table (LogAnalytics)
SNMP traps	Linux with SNMP trap receiver -and- AMA (syslog file) / DCR - or - AMA (syslog stream) / DCR	Required (DCR)	(LogAnalytics)
Change Tracking (legacy	Change Tracking extension (FIM) / DCR	Required (DCR)	Standard table (LogAnalytics)
Custom logs (Log Ingest API)	DCR /DCE	N/A	Custom Log table (LogAnalytics)
Standard logs (Log Ingest API) (*)	DCR /DCE	N/A	Standard table (LogAnalytics)
Standard/Platform Metrics/Telemetry (Azure Paas)	DCR (built-in, non-manageable, hidden)	N/A	Azure Monitor Metrics
Custom Metrics/Telemetry (custom app)	AMA / DCR -or – Azure Diagnostics extension -or- Azure Monitor REST API -or- Linux: InfluxData Telegraf agent (Linux) + Azure Monitor output plugin	Required (DCR)	Azure Monitor Metrics
Platform Logs (diagnostics per resource) • AllMetrics (don't send to LA – you already have data via Azure Monitor Metrics) • Resource logs (alllogs , audit)	Azure Policy (Diagnostics) DCR (built-in, non-manageable, hidden)	Required (Policy)	Standard table (LogAnalytics)
Activity logs (audit per subscription)	Azure Policy (Diagnostics) DCR (built-in, non-manageable, hidden)		Standard table (LogAnalytics)

Collection Source	Technologies needed	Flow
Performance Eventlog Syslog	AMA / DCR	AMA -> DCR Ingestion Pipeline (backend) → LA
Text logs IIS logs Windows Firewall logs (preview)	AMA / DCR / DCE	AMA -> DCE -> DCR Ingestion Pipeline (backend) → LA
SNMP traps	Linux with SNMP trap receiver -and- AMA (syslog file) / DCR - or - AMA (syslog stream) / DCR	AMA-> DCR Ingestion Pipeline (backend) → LA
Change Tracking (legacy	Change Tracking extension (FIM) / DCR	FIM -> DCR Ingestion Pipeline (backend) → LA
Custom logs (Log Ingest API)	DCR / DCE	REST EndPoint -> DCE -> DCR Ingestion Pipeline (backend) → LA
Standard logs (Log Ingest API) (*)	DCR / DCE	REST EndPoint -> DCE -> DCR Ingestion Pipeline (backend) → LA
Platform (Standard) Metrics/Telemetry (Azure Paas)	DCR (built-in, non-manageable)	Azure Resource -> DCR Ingestion Pipeline (backend) -> Azure Monitor Metrics
Custom Metrics/Telemetry (custom app)	AMA / DCR -or – Azure Diagnostics extension -or- Azure Monitor REST API -or- Linux: InfluxData Telegraf agent (Linux) + Azure Monitor output plugin	AMA -> DCR Ingestion Pipeline (backend) → LA AzDiag Ext -> LA REST EndPoint -> DCE -> DCR Ingestion Pipeline (backend) → LA InfuxData Telegraf -> Azure Monitor output plugin -> LA
Platform Logs (diagnostics per resource) • AllMetrics • Resource logs (alllogs, audit)	Azure Policy (Diagnostics) DCR	Azure Resource -> DCR -> LA
Activity logs (audit per subscription)	Azure Policy (Diagnostics) DCR	Azure Resource -> DCR -> LA