



OCTOBER 28, 2021

Julie Koesmarno (t: [@MsSQLGirl](#) | [MsSQLGirl.com](#))

From Oops to Ops: Incident Response with Jupyter Notebooks





Julie Koesmarno

@MsSQLGirl | MsSQLgirl.com

Bringing value to data and insights through experiences users love

Principal Program Manager
at Microsoft

From Ops to Ops: Incident Response with Jupyter Notebooks

What if you can apply software engineering practices to your troubleshooting guides (TSGs) / knowledge base for your team's on-call or for your customers? What if you can reduce stress and mistakes in your incident response workflow and activate a more scientific approach?

Join this session to learn more about TSG Ops framework using Jupyter Notebook for executable and automatable troubleshooting guides / knowledge base. TSG Ops innovates incident response approach by applying software engineering practice in curating TSGs and activating our scientific approach when troubleshooting. We will share our learnings and our journey in implementing TSG Ops using open source technology, internally in Azure Data and externally for our Azure Data customers.



Slide deck /
session notes

Our Learning Journey Today

1. Incident Response – why is it hard (to scale)?
2. Re-framing our approach to troubleshooting guides: software artefacts & Jupyter Notebooks
3. Implementation: Executable, reusable & automatable troubleshooting guides
4. Takeaways and other resources

“TSG”

TSG = Troubleshooting Guide

SOP = Standard Operating Procedure

Playbook = steps to identify issues

Runbook = procedures to achieve specific outcome

KB = Knowledge Base

Incident Response



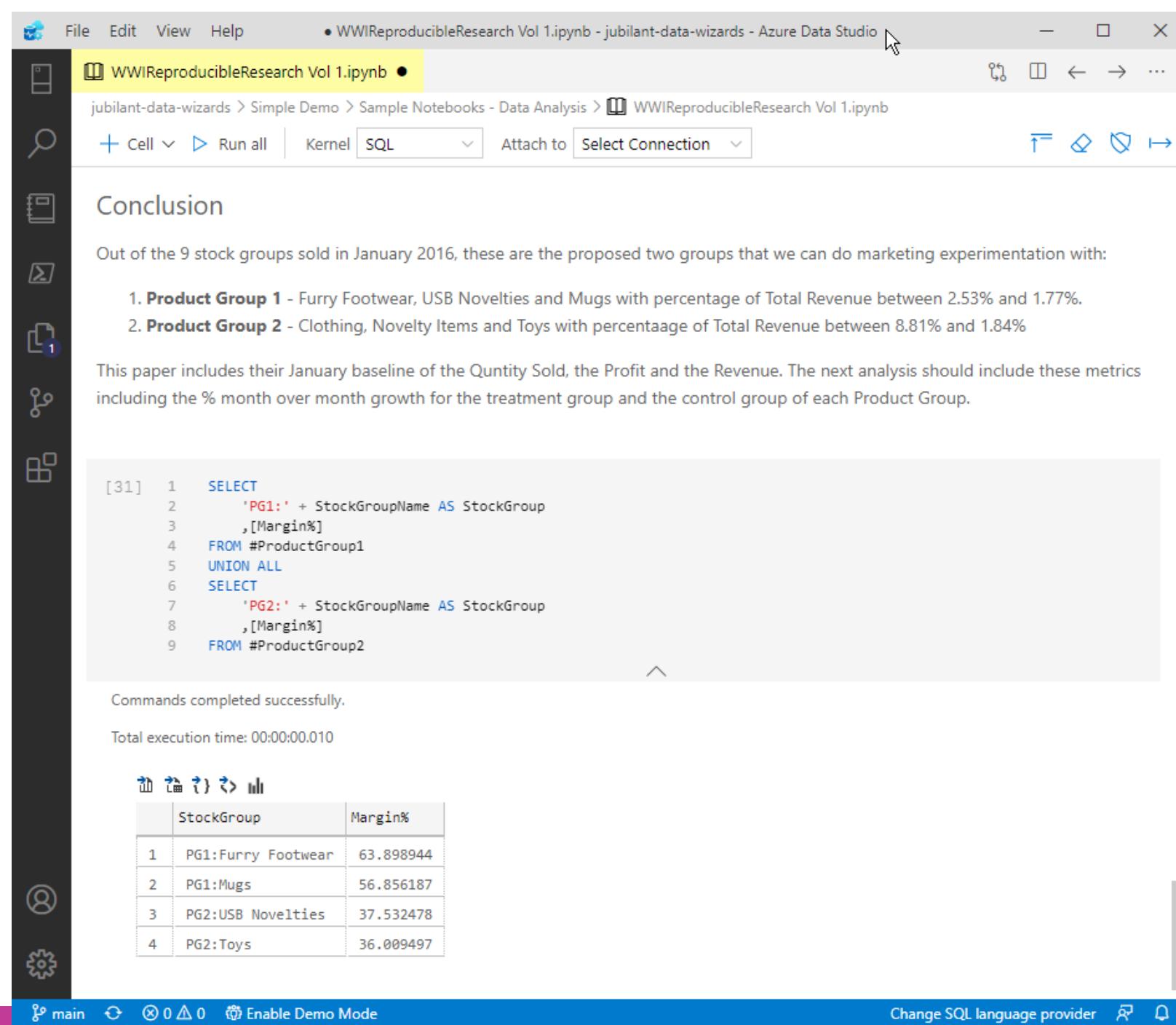
Learn more about notebooks

New to Jupyter Notebooks?

Watch Julie's Jupyter Notebooks
for the Mere Mortals
youtu.be/-akGNOsaMg0

New to parameterized Notebooks?

Watch Aaron's Parameterization
in Azure Data Studio (for
PowerShell)
youtu.be/5DDDeSb-mHP0



The screenshot shows the Azure Data Studio interface with a Jupyter Notebook open. The notebook title is "WWIReproducibleResearch Vol 1.ipynb". The code cell [31] contains the following SQL query:

```
[31] 1 SELECT
2     'PG1:' + StockGroupName AS StockGroup
3     ,[Margin%]
4 FROM #ProductGroup1
5 UNION ALL
6 SELECT
7     'PG2:' + StockGroupName AS StockGroup
8     ,[Margin%]
9 FROM #ProductGroup2
```

Below the code, a message indicates "Commands completed successfully." and "Total execution time: 00:00:00.010". A table is displayed with the following data:

	StockGroup	Margin%
1	PG1:Furry Footwear	63.898944
2	PG1:Mugs	56.856187
3	PG2:USB Novelties	37.532478
4	PG2:Toys	36.009497

At the bottom of the interface, there are navigation icons and status indicators.

1. Incident Response – why is it so hard (to scale)?



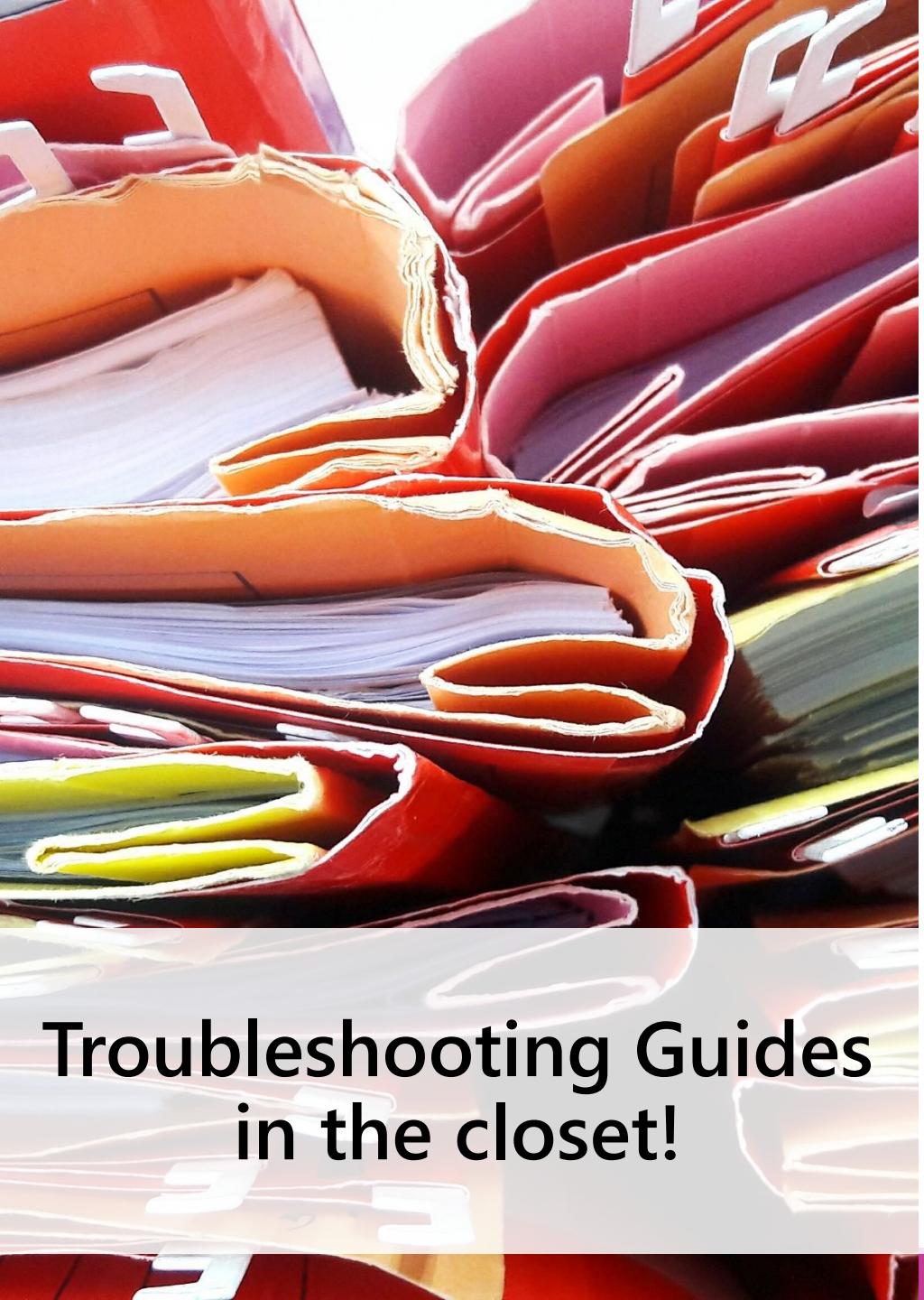
Where do you store
your TSGs today?





Ctrl C + Ctrl V

Toil! Subtle but can be erroneous



Troubleshooting Guides
in the closet!

Challenges:

1. Static text in OneNote / wiki resulting in copy and paste code to execute
2. Discoverability - siloed TSG scripts
3. No version control to keep track quality
4. Sharing code and results with context is hard to do with OneNote or text files
5. Hard to crowd source code
6. Hard to search
7. Not reusable
8. Not automatable

“... There is no such thing as “push through this phase and it will get significantly better.” Instead, we should take steps to make our work sustainable. That is done through process improvement work and automation, which will **reduce toil** and make the work **repeatable, consistent, fast, scalable, and auditable**. It will also free us up to do new, creative work.

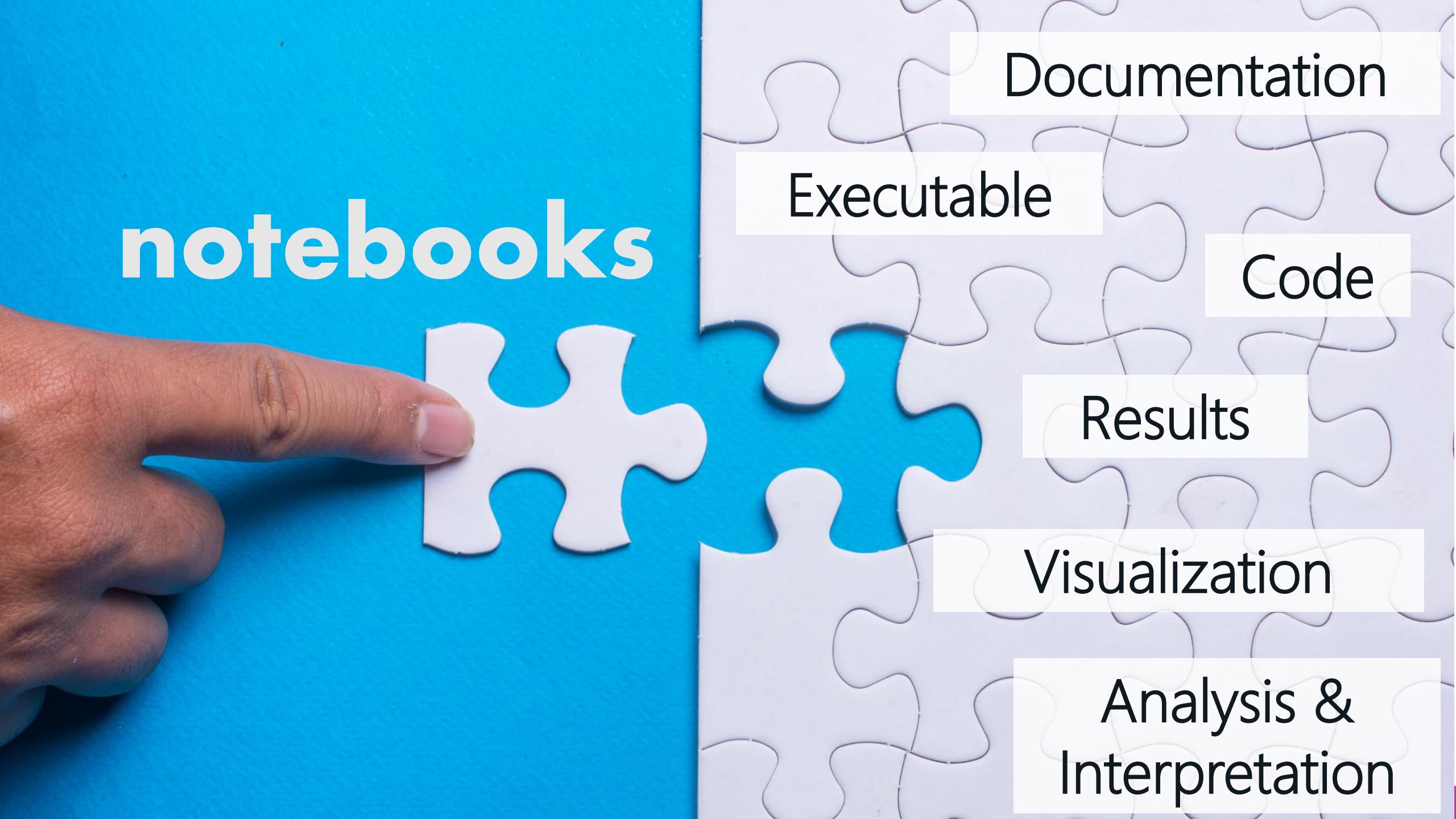
State of DevOps 2019, p61

2. Re-framing our approach to troubleshooting guides: software artefacts & Jupyter Notebooks

Troubleshooting Guides as **software artifacts**

- executable
- reusable
- automatable



A close-up photograph of a person's hand placing a single white puzzle piece onto a blue surface. To the right of the hand, a completed white puzzle is visible, divided into several rectangular sections. Each section contains a different word: "Documentation", "Executable", "Code", "Results", "Visualization", and "Analysis & Interpretation".

notebooks

Documentation

Executable

Code

Results

Visualization

Analysis &
Interpretation



Jupyter Notebook

Jupyter Notebook open source format

Jupyter Notebook dev tools – examples:

- Azure Data Studio
- VSCode
- Jupyter Lab

Jupyter Notebook viewers – examples:

- GitHub native notebook viewer
- Azure DevOps Jupyter Preview



Azure Data Studio

Modern data experiences, multi data-platforms

SQL: SQL Server, PostgreSQL, Azure SQL Edge, Azure SQL*

KQL: Azure Data Explorer

Environment: On-premises, poly-cloud

Client tool on multi-platforms: Windows, Linux, macOS

Extensible tool

Git support

Jupyter Notebook experiences

... and more ...

<http://aka.ms/getAzureDataStudio>

Notebooks as TSGs

Bring engineering discipline to incident response content

Notebooks can be checked into GIT repos to provide version control

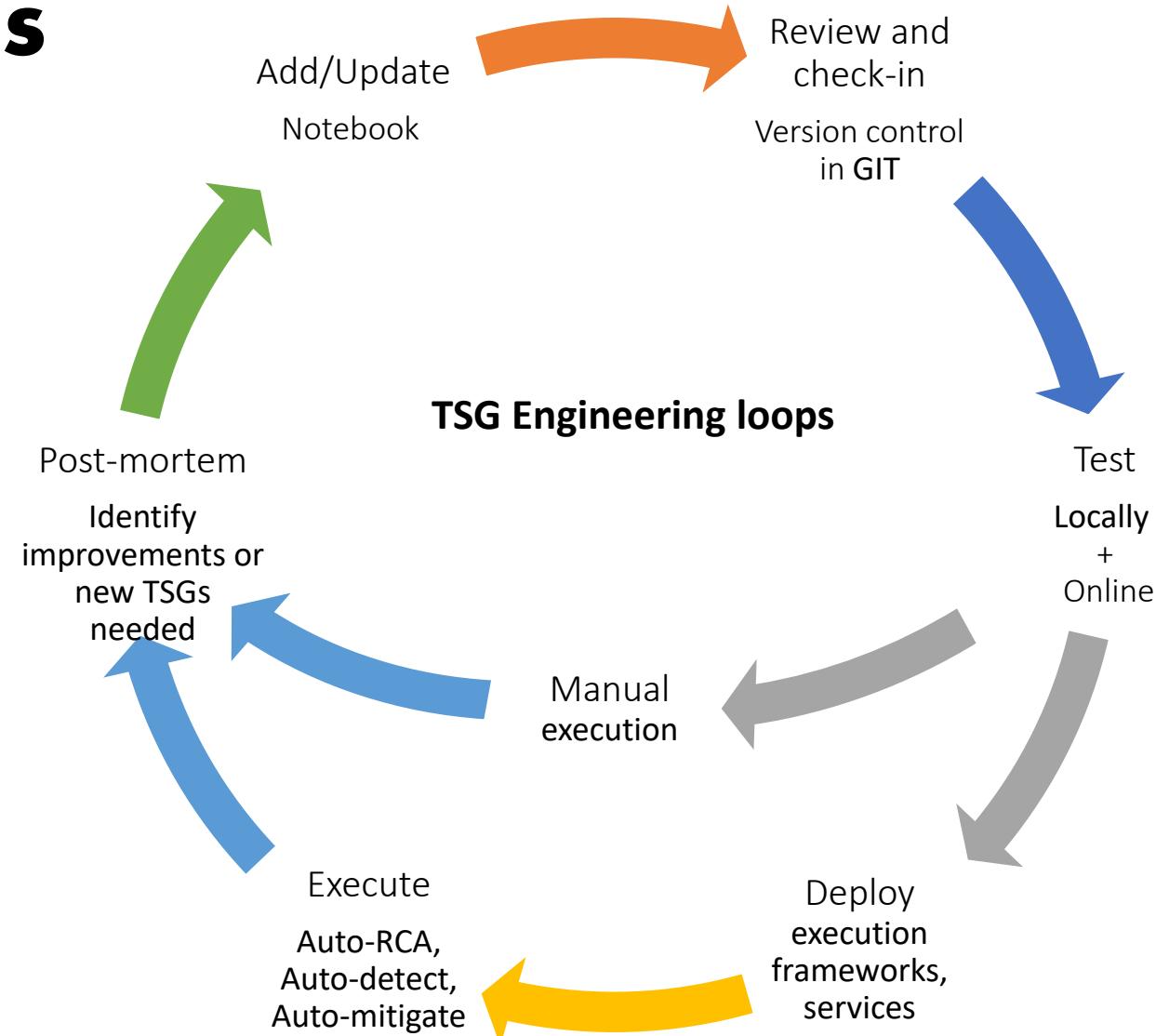
Review, test, deploy, similar to code

A simple system, with flexibility to create once and use everywhere

Notebooks can be shared easily

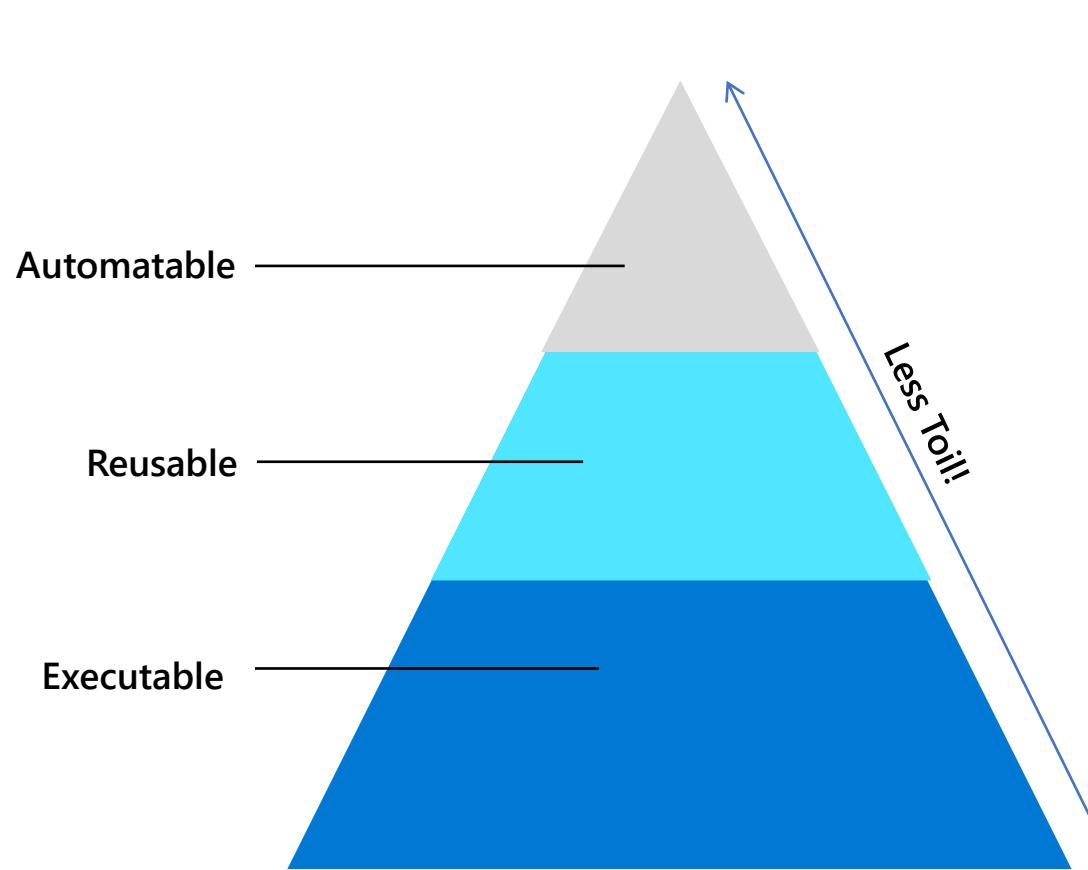
Notebooks inherently record every step executed with results/failures

A repeatable environment for debugging and investigation



3. Implementation: Executable, reusable & automatable troubleshooting guides

Notebooks – executable & beyond



Executable

Can be run on a UI

Reusable

Can be parameterized

Automatable

Can be run in command-line (no UI needed)

- Use `Invoke-SqlNotebook` to run SQL notebooks via PowerShell
- Use `Invoke-ExecuteNotebook` to run PowerShell notebooks via PowerShell
- Use `Papermill` for Python notebooks via Python

What can we automate?

1. Diagnosis / Root Cause Analysis (RCA)
2. Detection and mitigation



A simple demo to illustrate a Support Case System with automated TSG execution

The screenshot shows a Microsoft To-Do task card for a "Slow Query [cont...]" ticket. The card has the following details:

- Title:** Slow Query [cont...]
- Last changed:** moments ago by you
- Assignee:** Unassigned (dropdown menu open, showing Julie Koesmarno and Shafiq Rahman)
- Status:** started
- Priority:** Medium
- Notes:** My query runs slow on my database. Help ...
- Checklist:** 0 / 1
 - https://jktsgnotebooks.blob.core.windows.net/blobjktsgnotebooks/DBDiagnostic_20210501211232.ipynb
 - Add an item

Two purple callout boxes highlight specific features:

- A purple callout box points to the "Assign" dropdown menu with the text: "Automatically execute pre-defined notebook when assigned to me!"
- A purple callout box points to the notes section with the text: "Update the task with the pre-executed **notebooks** for me to view". A blue arrow points from this callout to the link in the notes section.

Email with link to pre-executed notebook and the task

Slow Query [jukoesmasqlldb.database.windows.net]



Cat-toso 'Sup-Bot'

To Julie Koesmarno

This message was sent with Low importance.

Diagnostics output (TSG Notebook): [Click to view in Azure Data Studio](#)

Link to Planner: [Slow Query \[jukoesmasqlldb.database.windows.net\]\[WideWorldImporters\]](#)

The screenshot shows the Azure Data Studio interface. At the top, there's a toolbar with File, Edit, View, Help, and a tab for DBDiagnostic_2021-09-16.ipynb. Below the toolbar, there's a search bar and a dropdown for Kernel (set to SQL). A button for 'Run all' is also present. To the right of the kernel dropdown is 'Attach to Select Connection'. On the far right of the header are standard window control buttons (minimize, maximize, close).

The main area displays a query result titled "Find single-use, ad-hoc and prepared queries that are bloating the plan cache (Query 13) (Ad hoc Queries)". The query itself is:

```
1 -- Find single-use, ad-hoc and prepared queries that are bloating the plan cache (Query 13) (Ad hoc Queries)
2 SELECT TOP(10) DB_NAME(t.[dbid]) AS [Database Name], t.[text] AS [Query Text],
3 cp.objecttype AS [Object Type], cp.cacheobjtype AS [Cache Object Type],
4 cp.size_in_bytes/1024 AS [Plan Size in KB]
5 FROM sys.dm_exec_cached_plans cp WITH (NOLOCK)
6 CROSS APPLY sys.dm_exec_sql_text(plan_handle) AS t
7 WHERE cp.cacheobjtype = N'Compiled Plan'
8 AND cp.objecttype IN ('N Adhoc', 'N Prepared')
9 AND cp.usescounts = 1
10 ORDER BY cp.size_in_bytes DESC, DB_NAME(t.[dbid]) OPTION (RECOMPILE);
```

Below the query text, it says "Total execution time: 00:00:00.0700913".

Below the text, there's a table titled "Database N...". The table has columns: Database Name, Query Text, Object Type, Cache Obj..., and Plan Size The data in the table is as follows:

Database N...	Query Text	Object Type	Cache Obj...	Plan Size ...	
1	WideWorldImport...	(@_msparam_0 n...	Prepared	Compiled Plan	2104
2	WideWorldImport...	(@_msparam_0 n...	Prepared	Compiled Plan	864
3	WideWorldImport...	SELECT [...	Adhoc	Compiled Plan	680
4	WideWorldImport...	(@_msparam_0 n...	Prepared	Compiled Plan	544
5	WideWorldImport...	(@_msparam_0 n...	Prepared	Compiled Plan	472
6	WideWorldImport...	SELECT [...	Adhoc	Compiled Plan	368
7	WideWorldImport...	SELECT * FROM [...	Adhoc	Compiled Plan	320
8	WideWorldImport...	(@_msparam_0 n...	Prepared	Compiled Plan	304
9	WideWorldImport...	(@_msparam_0 n...	Prepared	Compiled Plan	288
10	WideWorldImport...	(@_msparam_0 n...	Prepared	Compiled Plan	232

Open the pre-executed notebook in **Azure Data Studio**

Search resources, services, and docs (G+)

Home > DemoTSGWorkflow Logic app

Search (Ctrl+ /) Run Trigger Refresh Edit Delete Disable Update Schema Clone Open in mobile Export ...

Introducing the new portable Logic Apps runtime that supports local development and debugging. Click to learn more. →

View Cost | JSON View

Overview Activity log Access control (IAM) Tags Diagnose and solve problems

Development Tools

- Logic app designer
- Logic app code view
- Versions
- API connections
- Quick start guides

Settings

- Workflow settings
- Authorization

Resource group (change) jukoesma_exe_rg

Location West US 2

Subscription (change)

Subscription ID

Definition 1 trigger, 5 actions

Status Enabled

Runs last 24 hours 5 successful, 2 failed

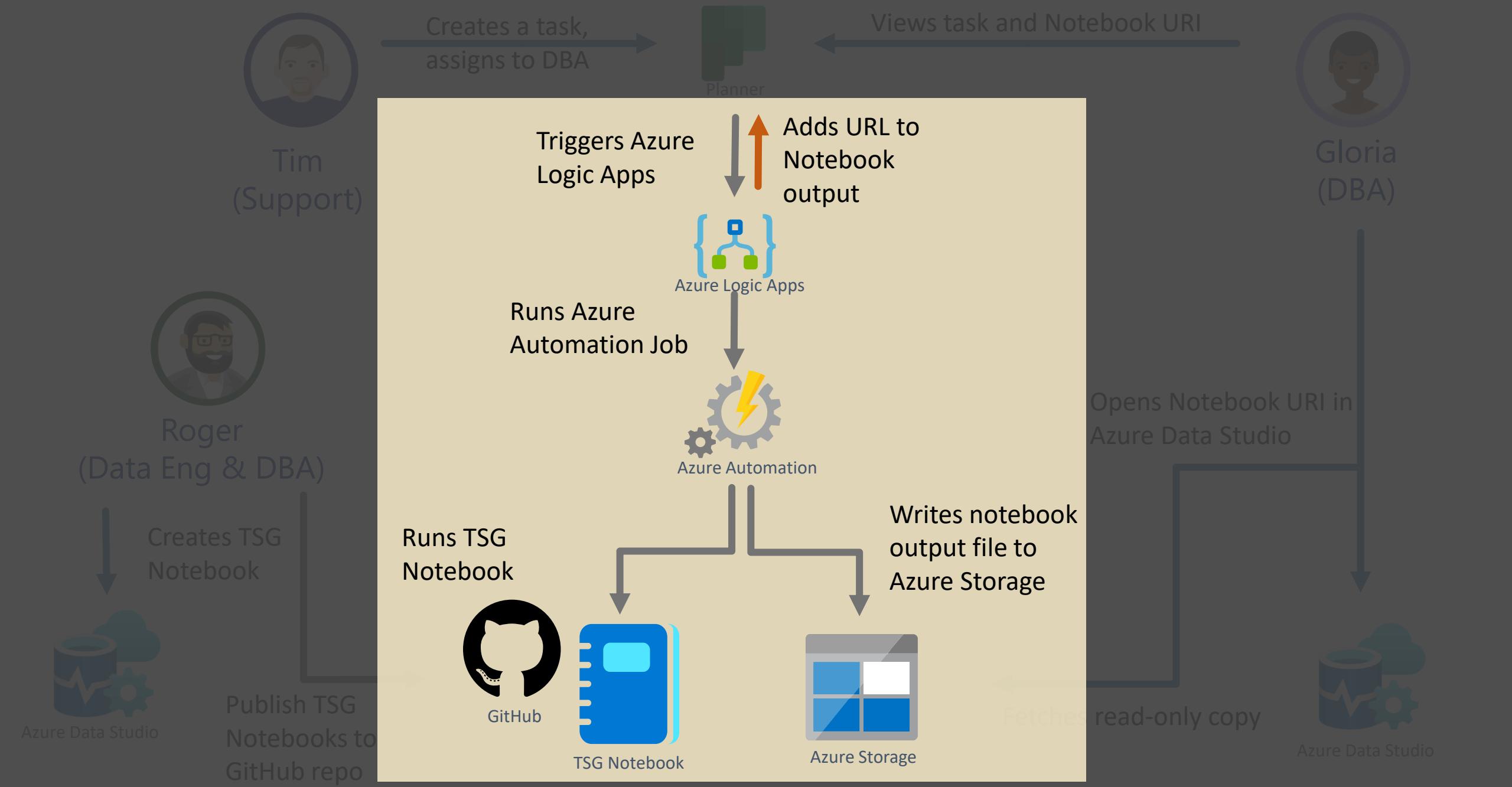
Integration Account ---

Get started Runs history Trigger history Metrics

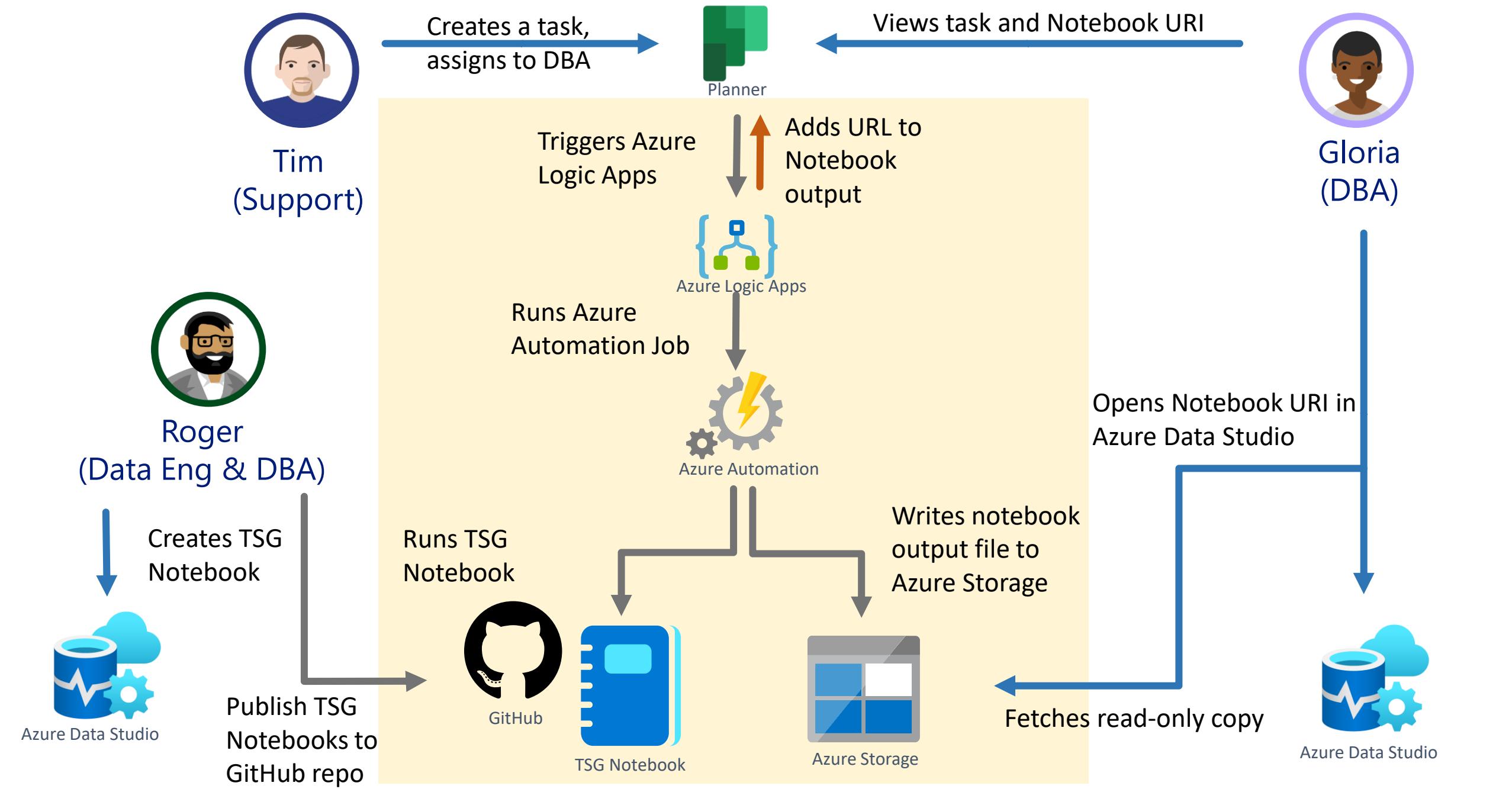
All Start time earlier than Pick a date Pick a time

Specify the run identifier to open monitor view directly

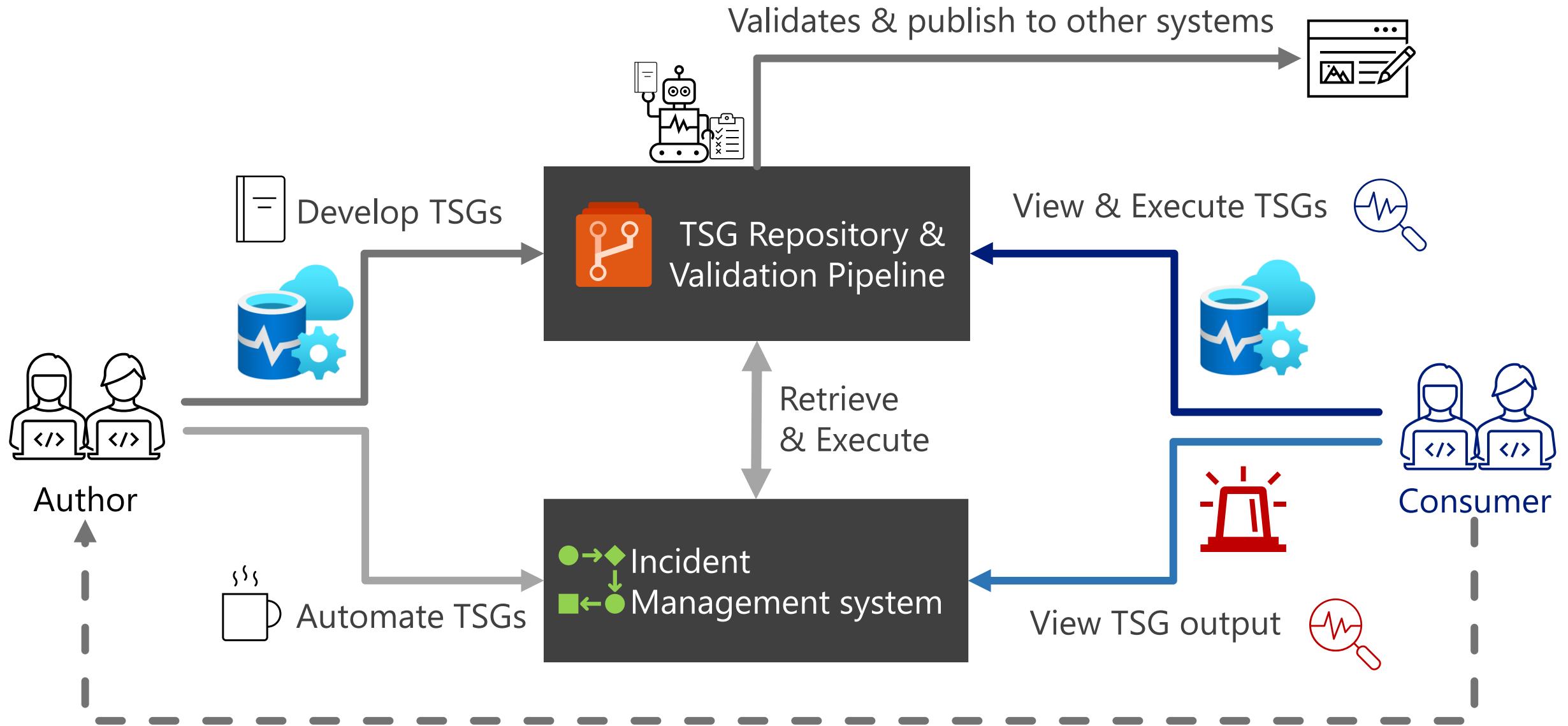
Status	Start time	Identifier	Duration	Static Results
Succeeded	5/1/2021, 1:39 PM	08585817052871593324217...	1.34 Minutes	
Succeeded	5/1/2021, 12:56 PM	08585817078908637089165...	45.74 Seconds	



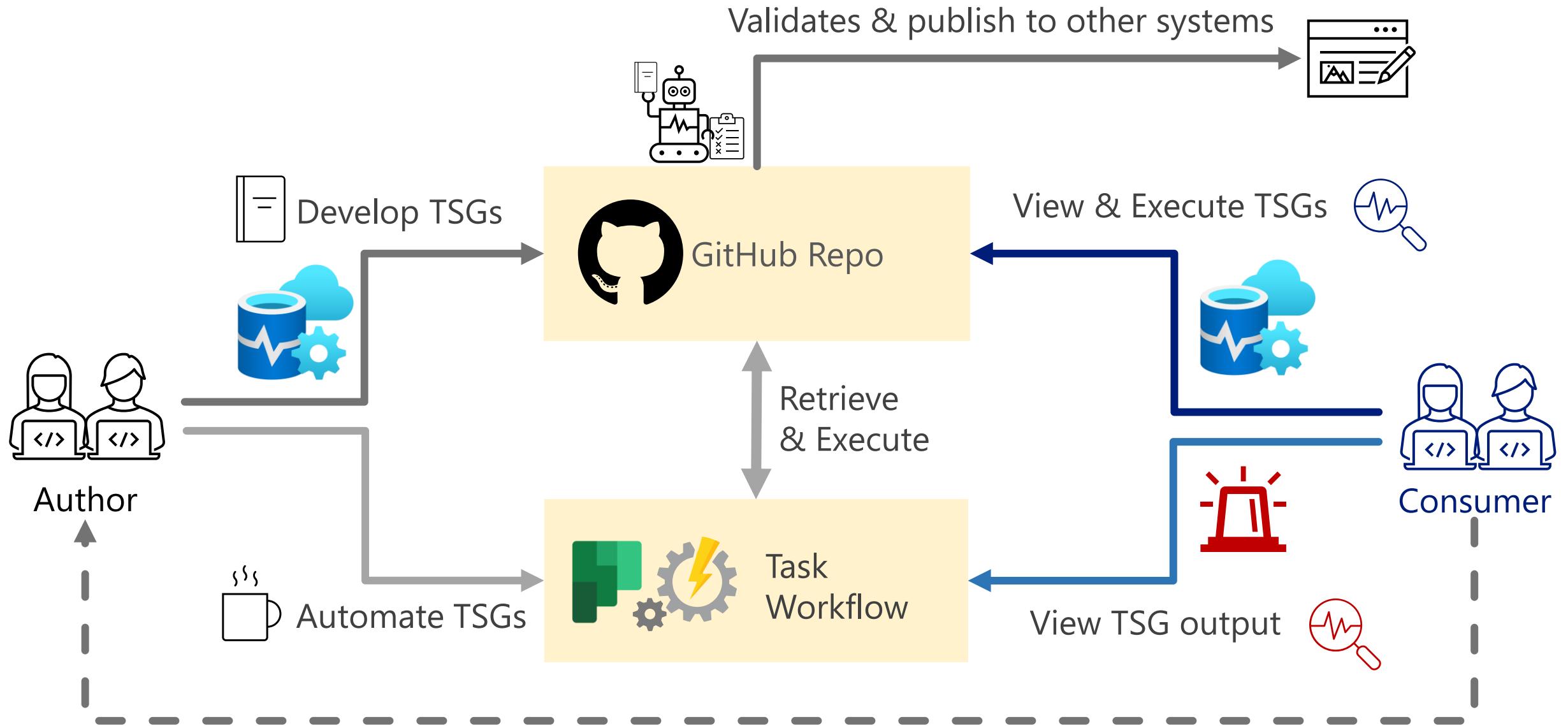
Demo: Automated task workflow with notebooks



Demo: Automated task workflow with notebooks



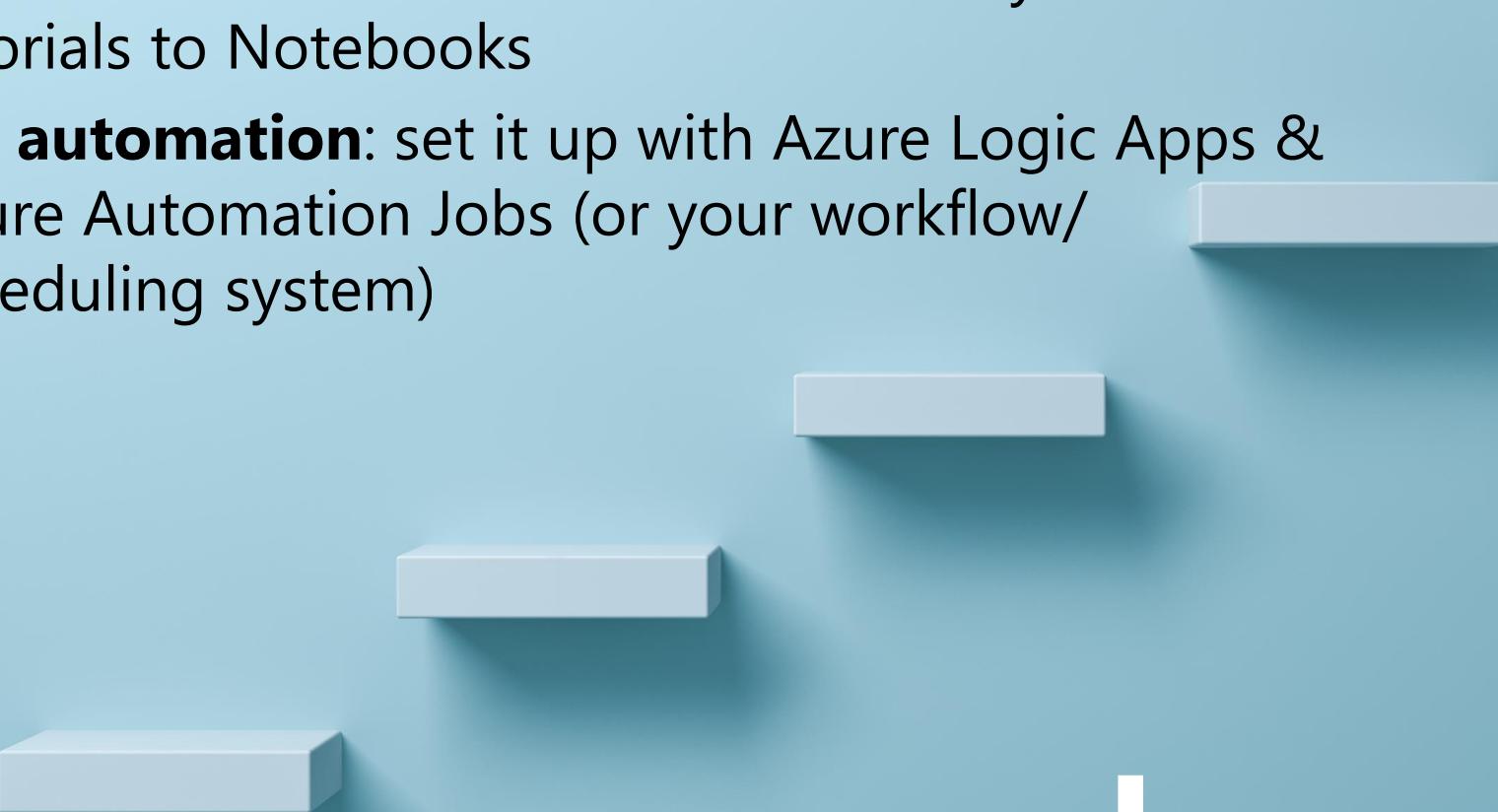
Demo: Automated task workflow with notebooks



Demo: Automated task workflow with notebooks

4. Takeaways & Resources

1. **Learn Jupyter Notebooks:** use Azure Data Studio, VSCode, JupyterLab and many others.
2. **Executable & Reusable TSGs:** Convert your TSGs or tutorials to Notebooks
3. **Try automation:** set it up with Azure Logic Apps & Azure Automation Jobs (or your workflow/scheduling system)



a change agent

Useful resources

Azure Data Studio: <http://aka.ms/AzureDataStudio>
<https://github.com/microsoft/azuredatastudio/issues>

SQL Assessment Toolkit: <https://github.com/microsoft/sql-server-samples/tree/master/samples/manage/sql-assessment-api/notebooks>

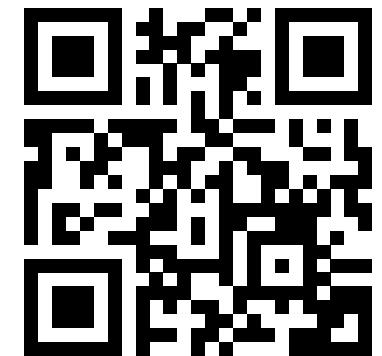
Glenn Berry's Diagnostic Notebooks for SQL Server and Azure SQL DB:
<https://glenncsqlperformance.com/resources/>

Rob Sewell's Notebook <http://sqldbawithabeard.com>
<https://github.com/SQLDBAWithABeard/JupyterNotebooks>

The SQL Diagnostic (Jupyter) Book by Emanuele Meazzo
<https://tsql.tech/the-sql-diagnostic-jupyter-book/>



**Session Notes at
MsSQLGirl.com**



Notebooks 101:
<https://bit.ly/2Ryu9uW>



OCTOBER 28, 2021

Thank You!





Jump start to Notebooks!

Bulk conversions -

- SQL Scripts to Notebooks with [ConvertTo-SqlNoteBook](#)
- PowerShell scripts to Notebooks with [ConvertTo-PowerShellNotebook](#)

Demo: <https://youtu.be/80L-UT0Ikew?t=1627>

Individual conversions -

- Export as Notebook UI in Azure Data Studio from a SQL Script
- [PowerShell history to a Notebook](#)

```
Get-History 7,14 | % comm* | Export-  
AsPowerShellNotebook -OutputNotebook  
d:\temp\testthis.ipynb
```

Tips: Notebook Parameterization

Watch how to use Notebooks parameterization in Azure Data Studio: <https://youtu.be/5DDDeSb-mHPO>

Notebook Language or Kernel	Execution Method	Parameterization support
Python	Papermill	Yes – all https://docs.microsoft.com/sql/azure-data-studio/notebooks/parameterize-papermill
PowerShell or .net interactive PowerShell	Invoke-ExecuteNotebook from PowerShellNotebook module built by Doug Finke	Yes – must be executed on PowerShell v7.1. https://github.com/dfinke/PowerShellNotebook#parameterizing-a-powershell-notebook
SQL	Invoke-SqlNotebook from SqlServer module	Limited – parameters supported are database connection details. https://docs.microsoft.com/powershell/module/sqlserver/invoke-sqlnotebook

Invoke SQL notebook from a command line

[Invoke-SqlNotebook](#)

```
dir 'SQLSERVER:\SQLRegistration\Database Engine Server Group' |
WHERE { $_.Mode -ne 'd' } |
foreach {
    $datetime = Get-Date -Format yyyyMMddhhmm;
    Get-SqlInstance -ServerInstance $_.Name |
    foreach {
        Invoke-SqlNotebook -ServerInstance $_.Name -Database master -InputFile
        '$home\Documents\SQL Server Management Studio\BPCheck.ipynb' ` 
        -OutputFile "BPCheck_output_$($_.NetName)_$($datetime).ipynb";
    }
}
```

Invoke PowerShell notebook from a command line

[Invoke-ExecuteNotebook](#) by Doug Finke

<https://github.com/dfinke/PowerShellNotebook#executing-a-notebook>

```
Import-Module PowerShellNotebook
```

```
Invoke-ExecuteNotebook  
  -InputNotebook 'path/to/input.ipynb'  
  -OutputNotebook 'path/to/output.ipynb'  
  -Parameters @{ alpha=0.6; ratio=0.1 }
```