



**Data  
Brothers**

# **Orchestration Best Practices in Microsoft Fabric**

28.08.2025 | Manchester – Power BI & Fabric User Group

# ŠTĚPÁN REŠL (Steve)

Lead technical consultant



**Data  
Brothers**



**Microsoft®**  
Most Valuable  
Professional



**Power BI**  
kafíčko



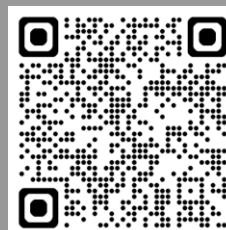
**Data  
Meerkat**



**LINKEDIN**



**X**



**BLUESKY**



# Agenda

What I will be talking about

Why we should care about orchestrations

Schedulers

Power Automate / API

Pipeline

Orchestration of notebooks



# Why we should care about orchestrations

alias... why we are here today

# Coordination and Dependencies

## Coordination

**Coordination** is a process that ensures the **harmonious** and **efficient** cooperation of all parts of a system. It is essential for the **correct sequencing** of task execution. Without adequate coordination, errors and inefficient resource utilization may occur. Coordination involves **planning** and **managing** individual **steps** of the process, as well as handling **dependencies between tasks**. Within the medallion architecture, coordination is crucial for integrating data across different layers. It also includes the implementation of **failover scenarios** to ensure continuity.

# Coordination and Dependencies

## Dependencies

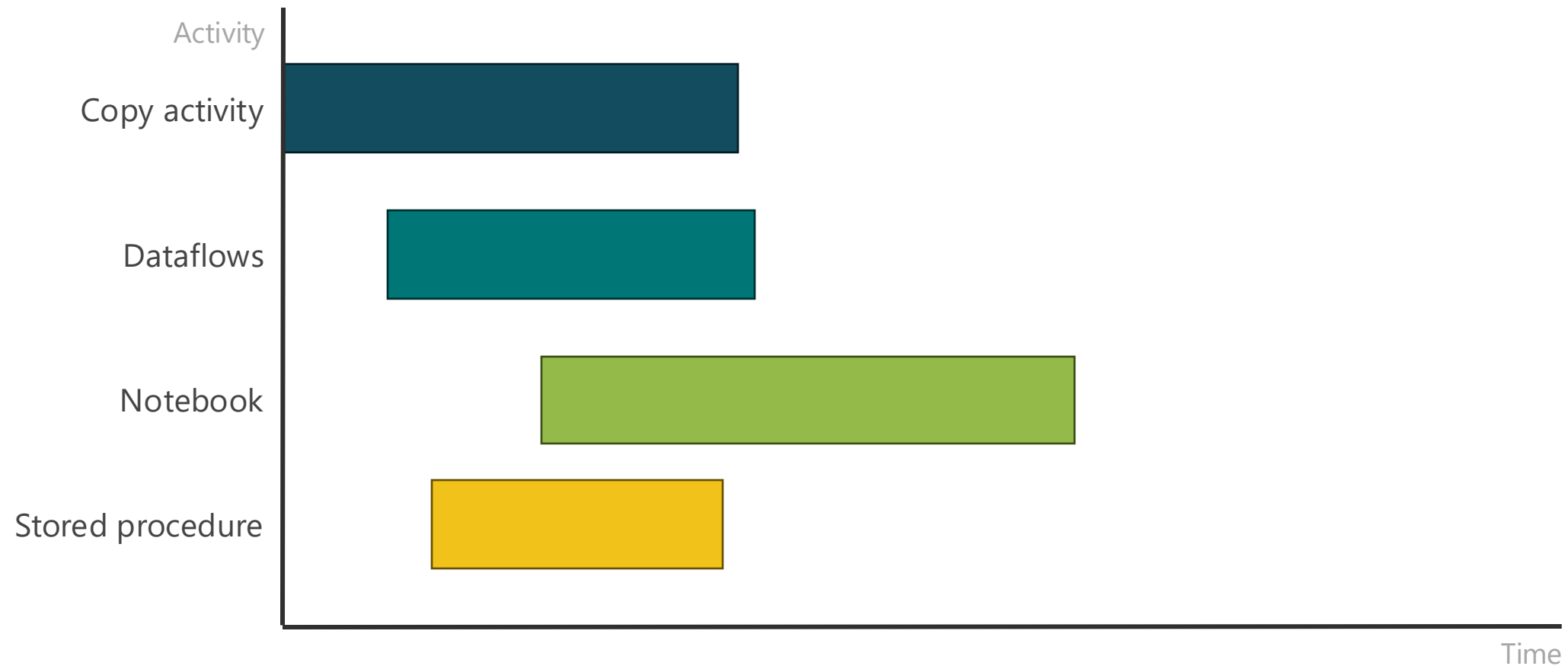
**Coordination** is a process that ensures the **harmonious** and **efficient** cooperation of all parts of a system. It is essential for the **correct sequencing** of task execution. Without adequate coordination, errors and inefficient resource utilization may occur. Coordination involves **planning** and **managing** individual **steps** of the process, as well as handling **dependencies between tasks**. Within the medallion architecture, coordination is crucial for integrating data across different layers. It also includes the implementation of **failover scenarios** to ensure continuity.

**Dependencies** in a pipeline are essential for ensuring the **correct order** and **coordination** of individual tasks. Each task in the pipeline may depend on other tasks, which means that some tasks must be completed before others can begin. Dependencies determine the execution **order** of tasks. If multiple tasks depend on the same preceding task, the pipeline must efficiently resolve these conflicts and ensure proper execution of all dependent tasks. Proper dependency management can significantly improve pipeline performance by **minimizing wait times** and maximizing resource utilization.

# Coordination and Dependencies

How to handle timing

**Timing** is crucial for ensuring the **correct order** and scheduling of individual steps.

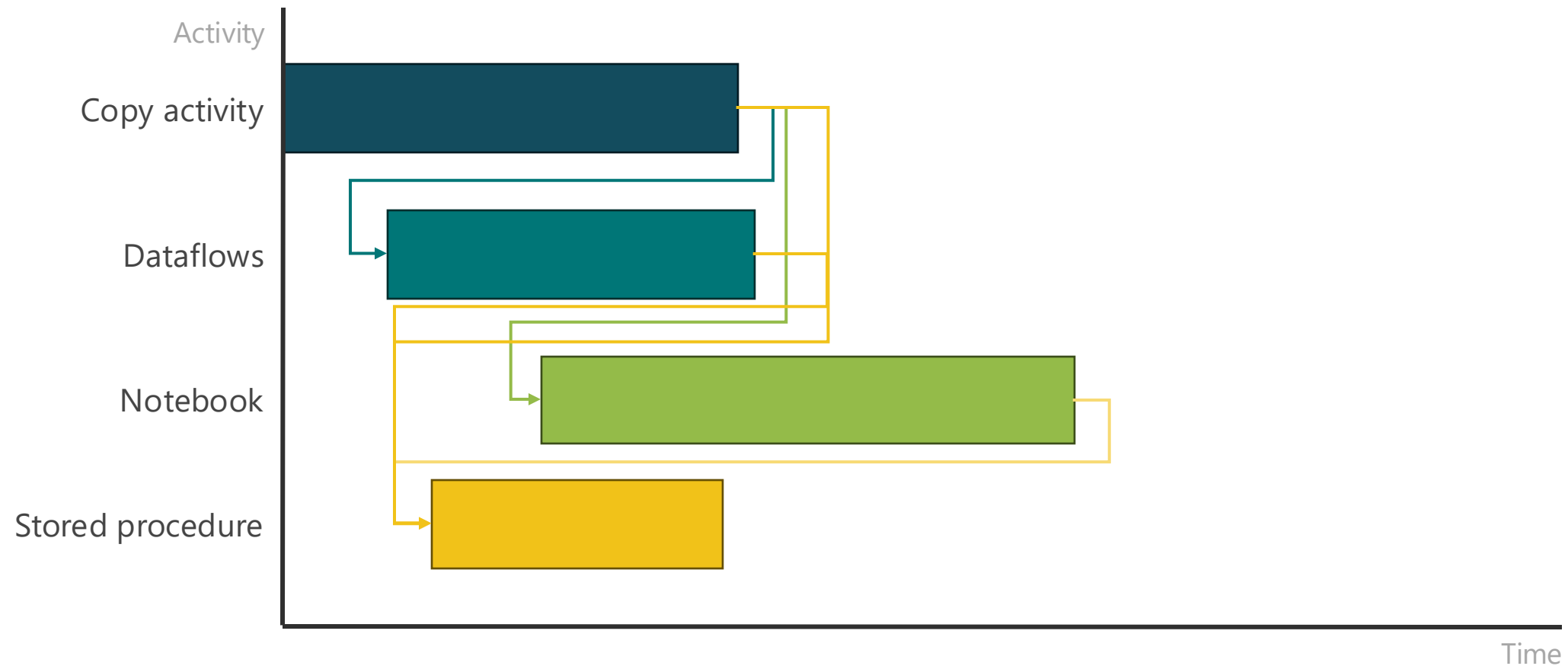




# Coordination and Dependencies

Disharmony can occur immediately when the timing is not correct.

**Timing** is crucial for ensuring the **correct order** and scheduling of individual steps.

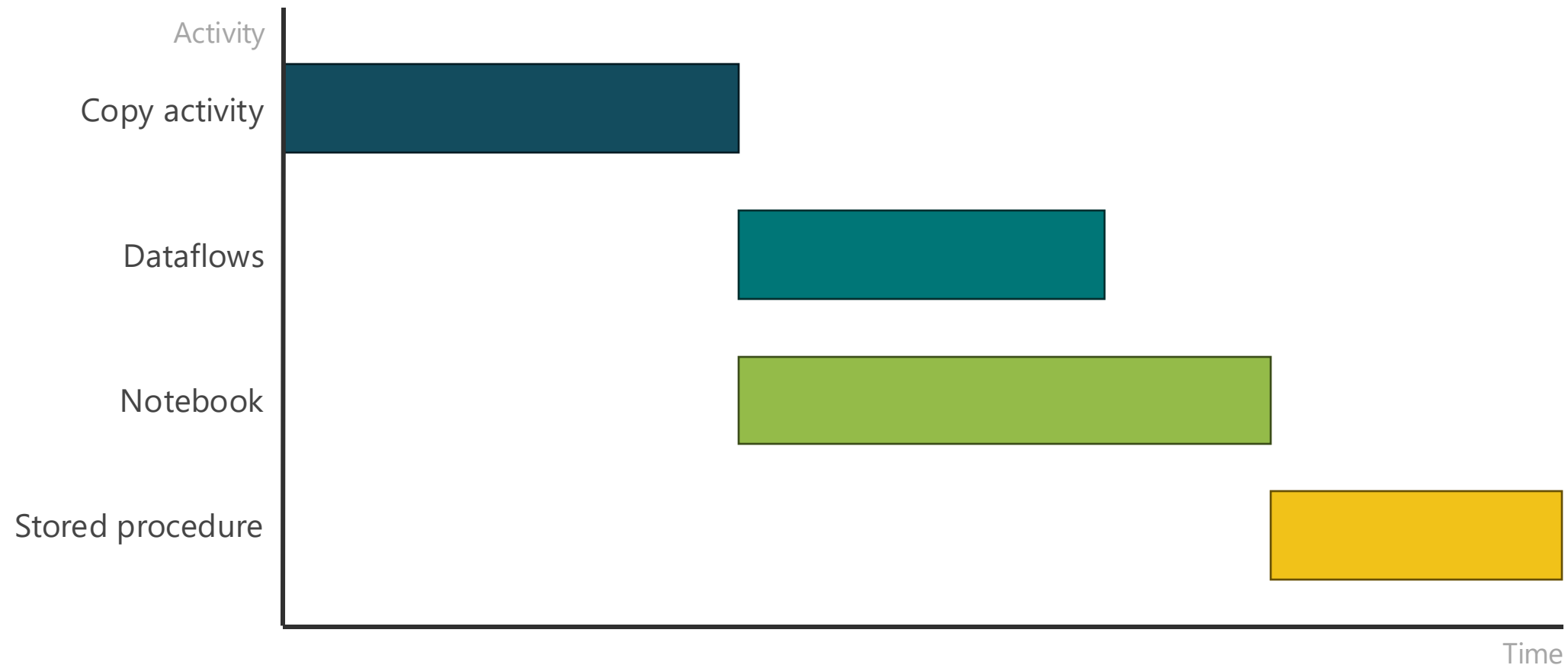




# Coordination and Dependencies

## Harmonization

**Timing** is crucial for ensuring the **correct order** and scheduling of individual steps.



# Global paralilism of executed items

How many items can be executed at one time?

- If they fit within the capacity... = not a fixed number
  - Semantic Model
  - Dataflow
  - Pipeline
  - Python Notebooks
  - ...
- Strictly limited number depending on purchased capacity
  - PySpark, Spark (Scala), R, HTML Notebooks -> Depends on the number of available sessions

# Restriction of global parallelism of PySpark Notebooks

and this can cause a lot of issues

Fabric capacity SKU	Equivalent Power BI SKU	Spark VCores	Max Spark VCores with Burst Factor	Queue limit
F2	-	4	20	4
F4	-	8	24	4
F8	-	16	48	8
F16	-	32	96	16
F32	-	64	192	32
F64	P1	128	384	64
F128	P2	256	768	128
F256	P3	512	1536	256
F512	P4	1024	3072	512
F1024	-	2048	6144	1024
F2048	-	4096	12288	2048
Trial Capacity	P1	128	128	NA

# On-Demand request vs. Triggered request

How is it counted, and into what?

If a Fabric item is started through the UI or via the REST API, it is executed as an *On-Demand request*. These requests can only be started **if there is currently available capacity**. For notebooks, this often results in the following error:

```
HTTP Response code 430: This Spark job can't be run because you have hit a Spark compute or API rate limit. To run this Spark job, cancel an active Spark job through the Monitoring hub, or choose a larger capacity SKU or try again later.
```

However, if a Fabric item is started through a scheduler or from an already running activity, it is executed as a *Triggered request*. These requests can be placed into a queue (provided there is space, according to the mentioned limit), from which they will be executed as soon as capacity becomes available.











# Local parallelisation of executed item

wait... what? Can something be running multiple times at once?

- 1 item = 1 active instance (*0 instances in queue*)
  - Semantic Model
  - Dataflow
  - Pipeline
  - Data stream
- 1 item = 1 active instance (*X instances in queue*)
  - Pipeline
  - Python Notebooks
  - PySpark, Spark (Scala), R, HTML Notebooks

## Local parallelisation of executed item

wait... what? Can something be running multiple times at once?

Knowledge Standup_b413e46f-feac-4d0a-99f3-49a44affcc9e	✓ Succeeded	 Notebook
Employees_46a3203c-ec58-4c1b-8d3c-5d00c5549b3f	✓ Succeeded	 Notebook
Buddies_24ee05bc-ea2a-4c47-a6ab-0427987dd2d0	✓ Succeeded	 Notebook
Run_Load_148b5025-335c-4539-aed2-226fab54cca6	✓ Succeeded	 Notebook
Power BI Cards_934c6b8a-1277-4d77-aadf-c5cf98e776e3	✓ Succeeded	 Notebook
Run_Load_bbe9a829-d937-447f-b1a2-44e52eb78af6	✓ Succeeded	 Notebook
Certifications_e28a8c90-e0ee-4c95-b115-3302bb9c0af5	✓ Succeeded	 Notebook
Weekly_Priorities_4f49a134-3845-4419-ab49-3fd7ece5a190	✓ Succeeded	 Notebook
Clockify_10c13752-d257-4c2d-9c4c-670291a1ea6f	✓ Succeeded	 Notebook
Bronze	✓ Succeeded	 Data pipeline

**ID of local instance or... Livy\_ID**

# High Concurrency Sessions

As a means of sharing resources of a single session for notebooks

They allow sharing a session across multiple notebooks simultaneously. This is very useful when we want to be considerate of other users and processes. To enable session sharing, the following rules from the documentation must be followed:

- The notebooks must use the **same default Lakehouse**,
- they must use the **same Spark configuration**,
- they must use the **same libraries (i.e., environment)** – additional libraries can then be installed within the notebook



# High Concurrency Sessions

As a means of sharing resources of a single session for notebooks

## Workspace settings

⚙️ General

💎 License info

☁️ Azure connections

💻 System storage

🔗 Git integration

🌐 OneLake

👤 Workspace identity

🔒 Network security

Power BI

Delegated Settings

</> OneLake settings

Data Engineering/Science

**⚙️ Spark settings**

Data Factory

### Spark settings

Configure and manage settings for Spark workloads and the default environment for the workspace.

Pool   Environment   **High concurrency**   Automatic log

#### For notebooks

When high concurrency for notebooks is on, multiple notebooks can use the same Spark application to reduce the start time for each session. [Learn more about running notebooks in high concurrency mode](#)

☒ On

#### For pipeline running multiple notebooks

When high concurrency for pipelines is on, multiple notebooks can use the same Spark application to reduce the start time for each session. [Learn more about running pipelines in high concurrency mode](#)

☐ Off

# Schedulers

Native option for timers. But are they reliable?

# Semantic Models

A simple timer allowing execution  $8\times$  to  $48\times$  per day (depending on the license).

## Refresh

### Time zone

① Time zone configuration is applied not only to determine the schedule refresh time but also to establish the current date and time for incremental refresh models during on-demand and API refreshes. [Learn more](#)

(UTC+01:00) Belgrade, Bratislava, Budapest

### Configure a refresh schedule

Define a data refresh schedule to import data from the data source into the semantic model. [Learn more](#)

☒ On

### Refresh frequency

Daily

### Time

7 00 AM X

8 00 AM X

9 00 AM X

[Add another time](#)

### Send refresh failure notifications to

☒ Semantic model owner

☐ These contacts:

Enter email addresses

# Dataflows

Does this mean it will run exactly at the specified time?

Dataflow 4

### Nastavení pro Dataflow 4

Tento tok dat naposledy upravil(a) [roza@w3x1w.onmicrosoft.com](mailto:roza@w3x1w.onmicrosoft.com).

Probíhá aktualizace...  
[Historie aktualizace](#)

#### Připojení brány

Místní brány toku dat se v tuto chvíli dají upravovat přes online prostředí Power Query. [Jak provádět úpravy](#)

► Přihlašovací údaje ke zdroji dat

⚙ Aktualizovat

#### Časové pásmo

ⓘ Konfigurace časového pásma se používá nejen k určení času aktualizace plánu, ale také k určení aktuálního data a času pro modely přírůstkové aktualizace během aktualizací na vyžádání a aktualizací rozhraní API. [Další informace](#)

Koordinovaný univerzální čas (UTC, ▼)

#### Konfigurace plánu aktualizace

Definujte plán aktualizace dat pro import dat ze zdroje dat do sémantického modelu. [Další informace](#)

☒ Zap.

#### Četnost aktualizace

Denně ▼

#### Čas

7 ▼ 00 ▼ 00 ▼ ×

10 ▼ 00 ▼ 00 ▼ ×

[Přidat jiný čas](#)

#### Odeslat oznámení o selhání aktualizace

☒ Vlastník toku dat

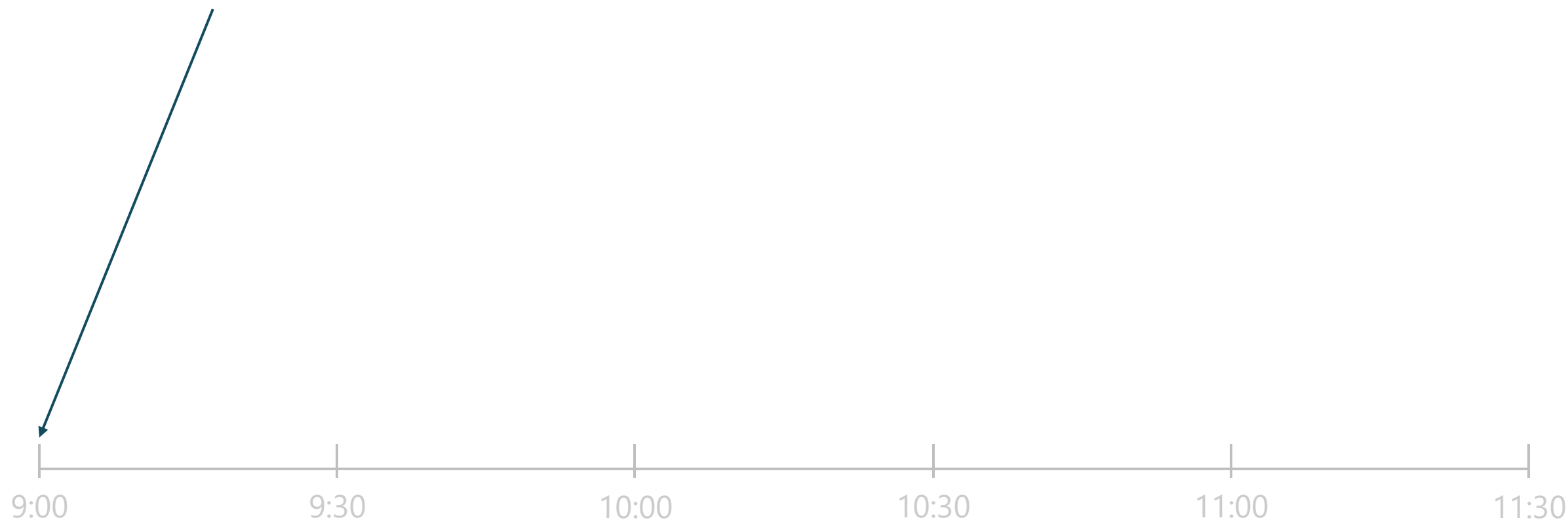
☐ Tyto kontakty:

► Doporučení

# Timing

How does the timing for these schedulers work?

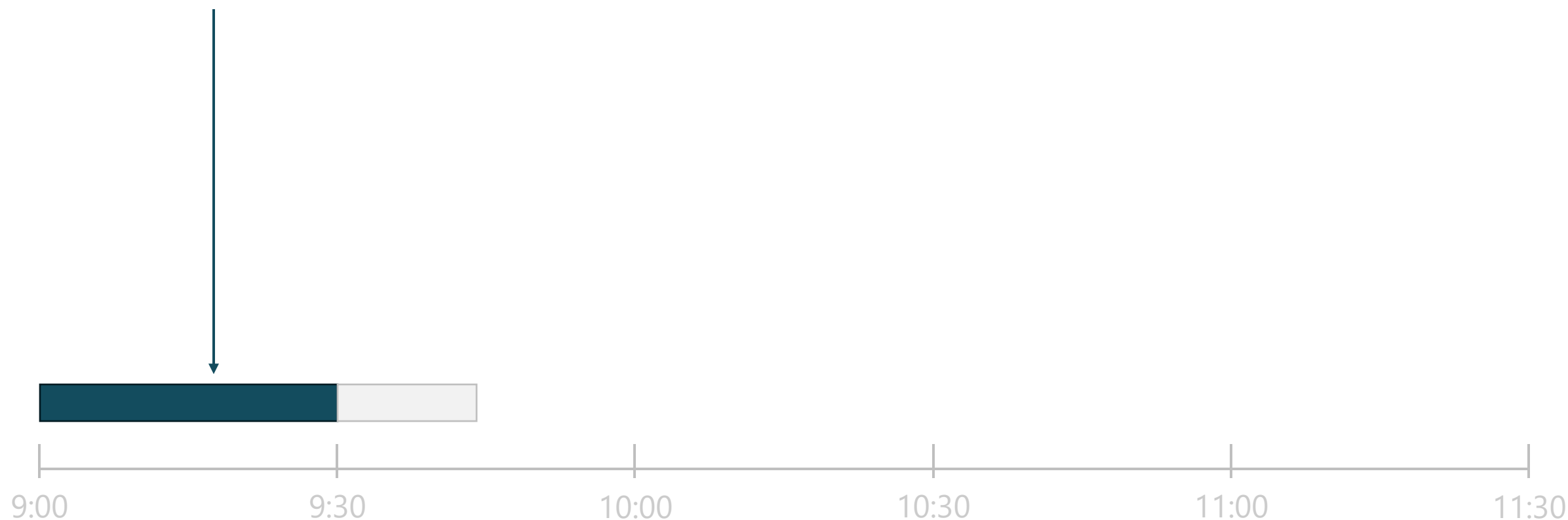
Refresh time: **9:00**



# Timing

How does the timing for these schedulers work?

Refresh time: **9:00**



# Timing

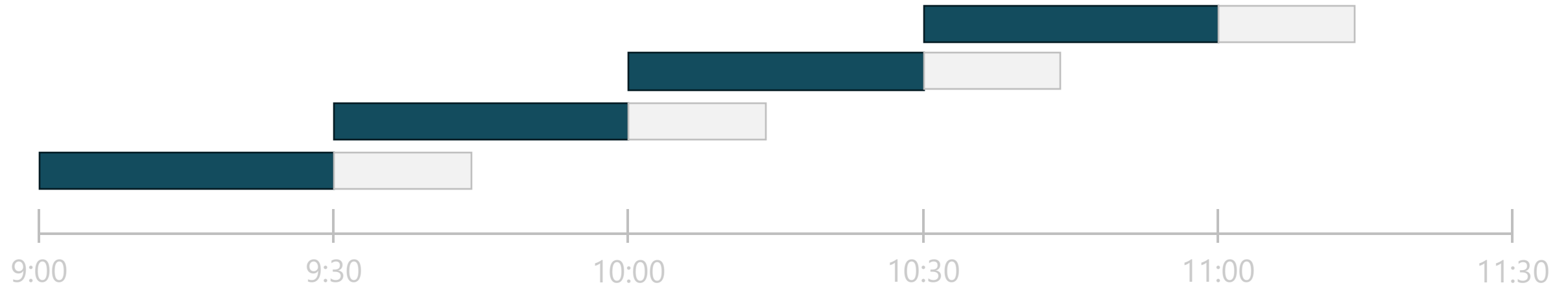
How does the timing for these schedulers work?

Refresh time: **9:00**

Refresh time: **9:30**

Refresh time: **10:00**

Refresh time: **10:30**





# Timing

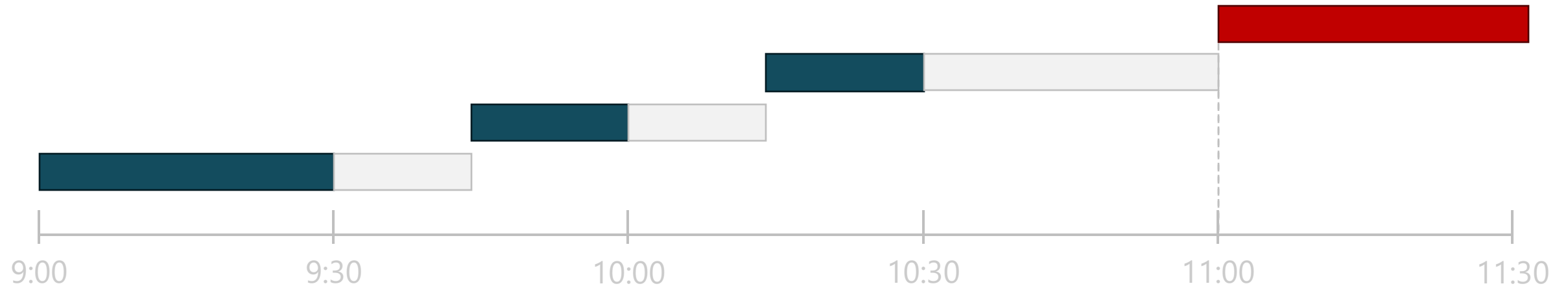
How does the timing for these schedulers work?

Refresh time: **9:00**

Refresh time: **9:30**


Refresh time: **10:00** (*Refresh was longer then expected*)

Refresh time: **10:30** (*Skipped*)



# Data Pipeline

It also has a scheduler, but a different one...


 **Medallion**  
Data pipeline

[About](#)  
[Endorsement](#)  
**[Schedule](#)**

**Last success is in**  
September 13, 2024 at 4:08:07 AM  
(UTC+01:00) Brussels, Copenhagen, Madrid, Paris

**Next refresh in**  
44 minute(s)

[Run](#)

 Schedule



**Scheduled run**

☒ On ☐ Off

**Repeat**



Daily ▾

**Time**

06:00 AM  

[+ Add a time](#)

**Start date and time** **End date and time**

09/18/2024  09/20/2024 


**Time zone**

(UTC+01:00) Brussels, Copenhagen, Madric ▾

[Apply](#) [Discard](#)

# Notebook

But what if I were executing a notebook that will run longer than the timeframe for the next trigger

 **sleeper**  
Notebook

About

Sensitivity label

Endorsement


Tags (preview)

**Schedule**

Other people in your organization may have access to this notebook in this workspace. Carefully review this item before scheduling it.

No previous history

Next refresh in  
1 minute(s)

 Schedule

Scheduled run

☒ On ☐ Off


Repeat

By the minute ▾


Every

15 minute(s)

Start date and time

12.03.2025 08:27 

End date and time

12.03.2025 08:30 

Time zone

(UTC+01:00) Brussels, Copenhagen, Madric ▾







Apply

```
1 import time

1 time.sleep(120)
```

# Notebook

They will be inserted into the queue

Activity name	Status	Item type	Start time	Submitted by	Location
sleep	⌘ Not started	 Notebook	03/12/2025, 8:29 AM	Štěpán Rešl	 [T] Štěpán - AppInNotebookDemo
sleep	⌘ Not started	 Notebook	03/12/2025, 8:28 AM	Štěpán Rešl	 [T] Štěpán - AppInNotebookDemo
sleep_5ca8ca67-bf50-4b7f-b48f-607805f2e37d	🕒 In progress	 Notebook	03/12/2025, 8:27 AM	Štěpán Rešl	 [T] Štěpán - AppInNotebookDemo

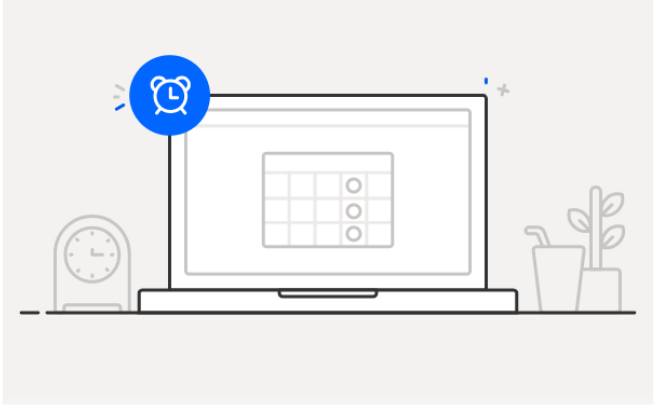
# Power Automate / API

When we need an exact update time in cases where the scheduler is not sufficient.

# Power Automate

Custom scheduler... (it also supports executing every minute, but... don't trust it...)

### Build a scheduled cloud flow



Stay on top of what's important without the effort—you choose when and how often the flow runs.

Examples:

- Automate team reminders to submit expense reports
- Auto-backup data to designated storage on a regular basis

Flow name

SemanticModel Scheduler

Run this flow \*

Starting

3/12/25

at

10:00 AM

Repeat every

1

Hour

This flow will run:

Every hour


Skip

Create

Cancel

# Power Automate

Pre-created action for refreshing a semantic model under your connection.

 Refresh a dataset ⋮ ⏪

Parameters

Settings

Code view

Testing


About

Workspace <sup>\*</sup>

[D] Admin Monitor ⌵

Dataset <sup>\*</sup>

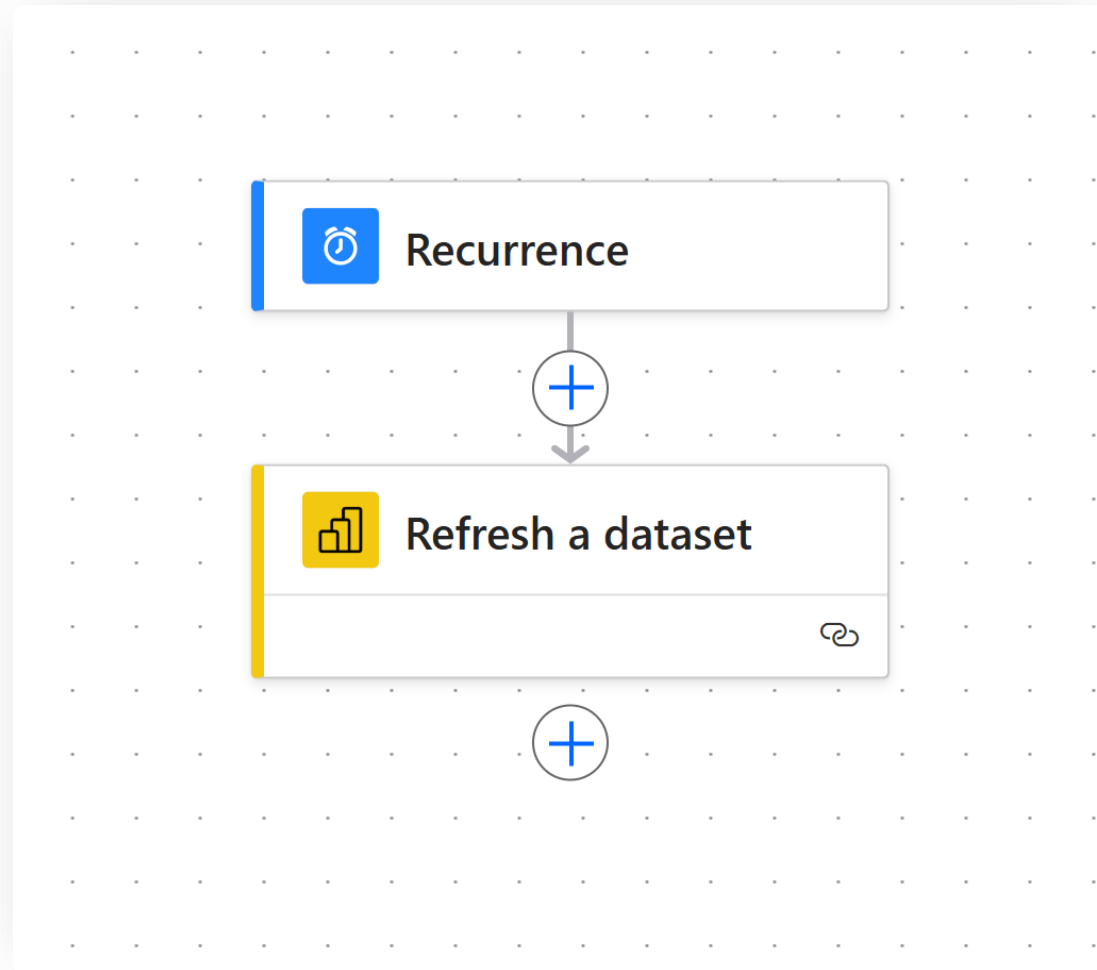
Power BI Admin Monitor ⌵

 Connected to stepan.resl@databrothers.cz. [Change connection](#)




# Power Automate

Trivial run of scheduler...



# Power Automate

The same works here for the Dataflows...

 Refresh a dataflow ⋮ ⏪

Parameters

Settings

Code view

Testing

About

Group Type \*

Workspace

Group \*

[Df] [I] Public Dataflows

Dataflow \*

DateKey

🔗 Connected to stepan.resl@databrothers.cz.

[Change connection](#)

# API

In other words, when the native features in Power Automate or the scheduler options are not sufficient...

- POST <https://api.powerbi.com/v1.0/myorg/groups/{groupId}/datasets/{datasetId}/refreshes>
- POST <https://api.powerbi.com/v1.0/myorg/groups/{groupId}/dataflows/{dataflowId}/refreshes>

- 
- POST <https://api.fabric.microsoft.com/v1/workspaces/{groupId}/items/{itemId}/jobs/instances?jobType=RunNotebook>
  - POST <https://api.fabric.microsoft.com/v1/workspaces/{groupId}/items/{itemId}/jobs/instances?jobType=Pipeline>
  - ...

# Refresh Dataset in Group

An API that enables much more than one might expect.

- POST <https://api.powerbi.com/v1.0/myorg/groups/{groupId}/datasets/{datasetId}/refreshes>
- Authorization: User, Service Principal (*Dataset.ReadWrite.All*)

Name	Required	Type	Description
notifyOption	True	Notify Option	Mail notification options. This parameter is not applicable to enhanced refreshes or API operations with a service principal.
applyRefreshPolicy		boolean	Determine if the policy is applied or not
commitMode		Dataset Commit Mode	Determines if objects will be committed in batches or only when complete
effectiveDate		string	If an incremental refresh policy is applied, the <code>effectiveDate</code> parameter overrides the current date.
maxParallelism		integer	The maximum number of threads on which to run parallel processing commands
objects		Dataset Refresh Objects[]	An array of objects to be processed
retryCount		integer	Number of times the operation will retry before failing. Temporary internal errors may trigger a retry of the refresh, even when this parameter is set to 0.
timeout		string	If a <code>timeout</code> is specified, each data refresh attempt on the semantic model will adhere to that timeout. Note that a single refresh request can include multiple attempts if <code>retryCount</code> is specified, which may cause the total refresh duration to exceed the specified timeout. For instance, setting a <code>timeout</code> of 1 hour with a <code>retryCount</code> of 2 could result in a total refresh duration of up to 3 hours. Users can adjust the <code>timeout</code> to shorten the refresh duration for faster failure detection or extend it beyond the default 5 hours for more complex data refreshes. However, the total refresh duration, including retries, cannot exceed 24 hours.
type		Dataset Refresh Type	The type of processing to perform

# Refresh Dataset in Group

An API that enables much more than one might expect.

```
{
  "type": "full",
  "commitMode": "transactional",
  "objects": [
    { "table": "Customer", "partition": "Robert" }
  ],
  "applyRefreshPolicy": "false",
  "timeout": "05:00:00"
}
```

Name	Type	Description
Automatic	string	If the object needs to be refreshed and recalculated, refresh and recalculate the object and all its dependents. Applies if the partition is in a state other than Ready.
Calculate	string	Recalculate this object and all its dependents, but only if needed. This value doesn't force recalculation, except for volatile formulas.
ClearValues	string	Clear values in this object and all its dependents
DataOnly	string	Refresh data in this object and clear all dependents
Defragment	string	Defragment the data in the specified table. As data is added to or removed from a table, the dictionaries of each column can become polluted with values that no longer exist in the actual column values. The defragment option will clean up the values in the dictionaries that are no longer used.
Full	string	For all partitions in the specified partition, table, or database, refresh data and recalculate all dependents. For a calculation partition, recalculate the partition and all its dependents.

Name	Type	Description
PartialBatch	string	Commit the refresh operation in batches. When utilizing <code>partialBatch</code> mode, the refresh operation does not occur within a transaction. Consequently, each command will be committed individually, and in the event of a failure, the model may end up in a state where only a subset of the data is loaded, or the table is left empty. If you desire to guarantee the preservation of previous data in case of a failure, you should execute the operation with <code>commitMode = transactional</code> .
Transactional	string	Commit the whole refresh operation as a transaction

# Power Automate + REST API

And suddenly, it's possible to run exactly what we need!

The image displays the Power Automate interface, split into two panels. The left panel shows the configuration for a REST API connector named 'refreshModel'.

**Parameters:**

- URI \*: `https://api.powerbi.com/v1.0/myorg/groups/ce1c84e1-c337-4e1f-a903-982e8057387c/datasets/2b798f0e-2f43-48f0-b3c6-29b1dd9e76e6/refreshes`
- Method \*: `POST`
- Headers:
  - Content-Type: `application/json`
  - Authorization: `concat(...)`
- Queries:
  - Enter key: `Enter key`
  - Enter value: `Enter value`
- Body: 

```
{
  "type": "full",
  "commitMode": "transactional",
  "retryCount": 3,
  "timeout": "02:00:00"
}
```
- Cookie: `Enter HTTP cookie`
- Advanced parameters: Showing 0 of 1

The right panel shows a flow diagram with the following steps:

- Recurrence
- imageOne
- repeaterValue
- Get secret
- inputs
- Variables
- oAuth-token
- tokenReciever
- refreshModel

# Execute Fabric Notebook

An API that enables much more than one might expect.

- POST <https://api.fabric.microsoft.com/v1/workspaces/{groupId}/items/{itemId}/jobs/instances?jobType=RunNotebook>
- Authorization: User, Service Principal

## Required Delegated Scopes


For item APIs use these scope types:

- Generic scope: `Item.Execute.All`
- Specific scope: `itemType.Execute.All` (for example: `Notebook.Execute.All`)

for more information about scopes, see: [scopes article](#).



## Microsoft Entra supported identities

This API supports the Microsoft [identities](#) listed in this section.

 Expand table

Identity	Support
User	Yes
Service principal and Managed identities	Yes

## Interface

HTTP  Copy  Try It

```
ft.com/v1/workspaces/{workspaceId}/items/{itemId}/jobs/instances?jobType={jobType}
```



# Pipeline

From Azure Data Factory to Azure Synapse, all the way to Microsoft Fabric

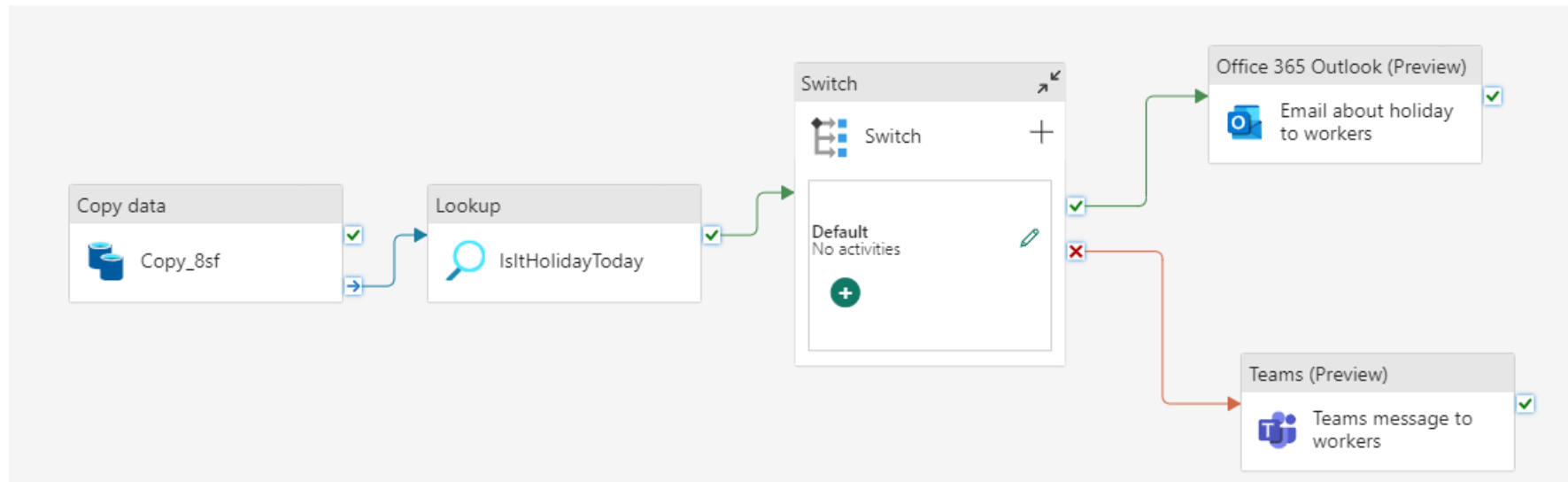
# Data Pipeline

## Short recapitulation

A Data Pipeline functions as an **orchestration component** that can trigger other items and services, and it can be scheduled to run at a specific time.

**Activities** are executable tasks within a Pipeline. You can define the flow of activities by linking them in **sequence**. **The outcome** of a particular activity (success, failure, or completion) can then be used to **direct the flow** to the next activity in the sequence.

A data pipeline enables the combination of various operations within a single framework and executes them over time as part of a unified process.



# Data Pipeline

Bins of activities

## Data movement activities

**Used to transfer data from point A to point B.** It leverages Azure Data Movement Services.

## Data transformation activities

**Used for data transformations.** These activities can be parameterized within the pipeline, allowing control over what is transformed and when.

## Control flow activities

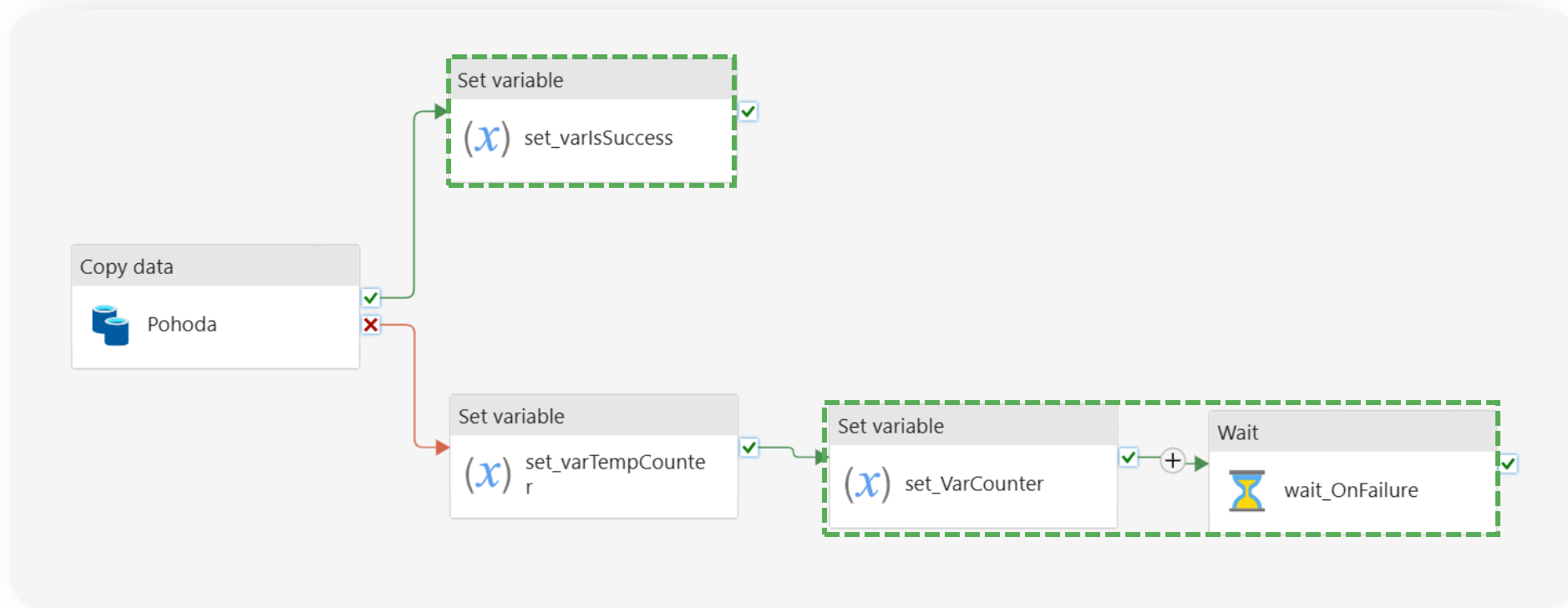
**Used to control the pipeline flow,** i.e., "what should happen when."

# Coordination and consequences

## Consequences - On Success

**On Success** is triggered when an activity completes successfully.

It is used to define the next steps in the pipeline that should occur **only if the preceding activity was successful**. For example, after successfully copying data, you may proceed with transforming it.

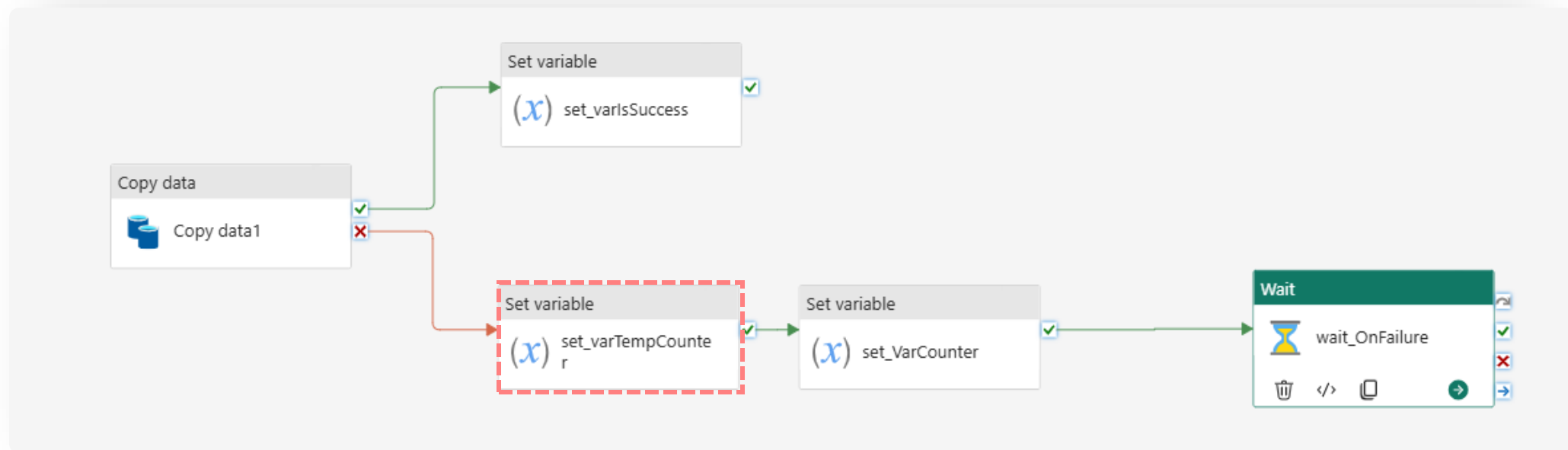


# Coordination and consequences

## Návaznost - On Fail

**On Failure** is triggered when an activity fails.

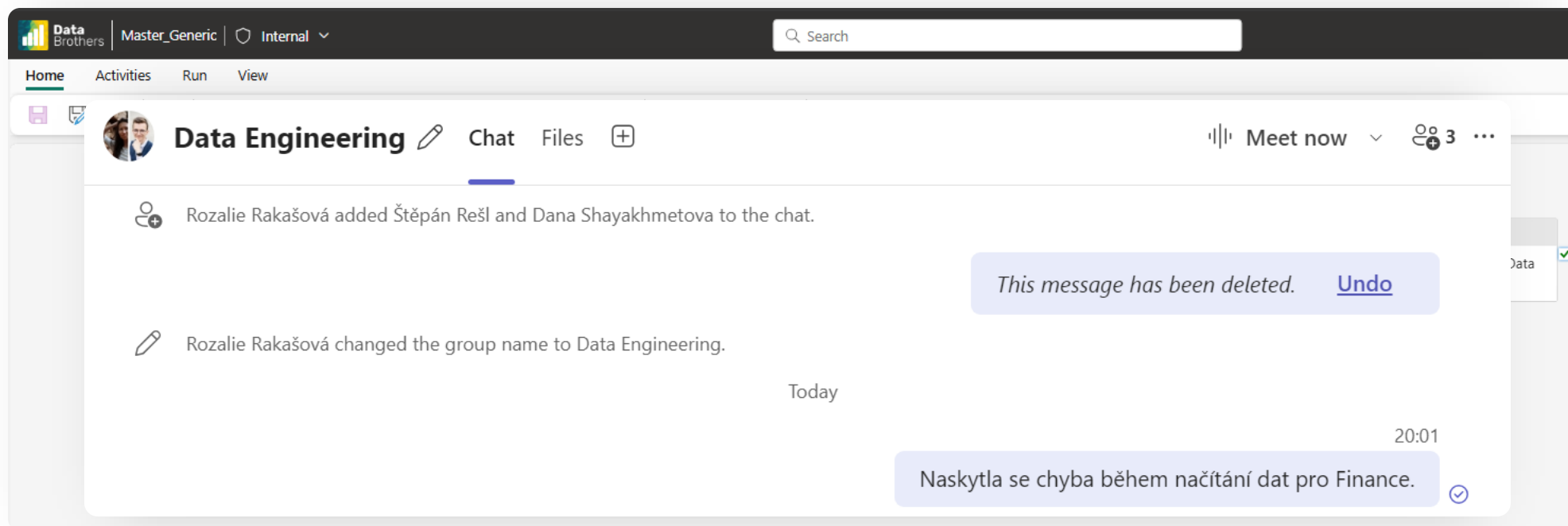
It is used for error handling and defining alternative actions when an activity does not complete as expected. This may include logging the error, sending a notification, or triggering another set of activities to handle the failure.



# Coordination and consequences

## Zaslání notifikace

In the event of an operation failure, it is possible to configure notifications to be sent to an email address or **Microsoft Teams**. By utilizing dynamic content from helper variables and filters, you can send information about the specific error source, eliminating the need to review all sources to locate the error.



# Coordination and consequences

Creating a variable that will contain result status of activities.

```
@createArray(  
  concat(  
    'Bronze | ',  
    activity('Bronze').Status  
  ),  
  concat(  
    'Silver | ',  
    activity('Silver').Status  
  ),  
  concat(  
    'Gold | ',  
    activity('Gold').Status  
  )  
)
```

## Výstup:

Bronz | Failed  
Silver | Failed  
Gold | Failed

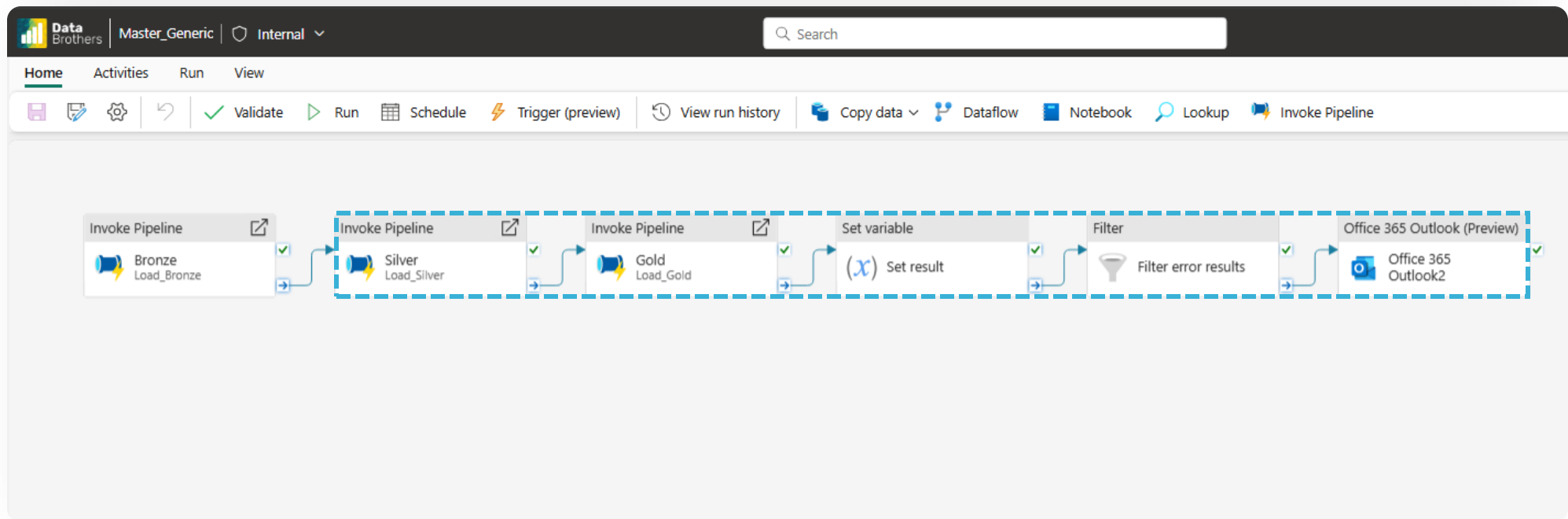
# Coordination and consequences

## Consequences - On Completion

**On Completion** is triggered when an activity finishes, regardless of whether it succeeded or failed.

It is used to define steps that should follow after the previous activity has finished, no matter the outcome (e.g., after a data copy step completes—whether successfully or with an error—perform cleanup operations or log the result).

However, it does not include the **On Skip** condition.





# Coordination and consequences

## Consequences - On Skip

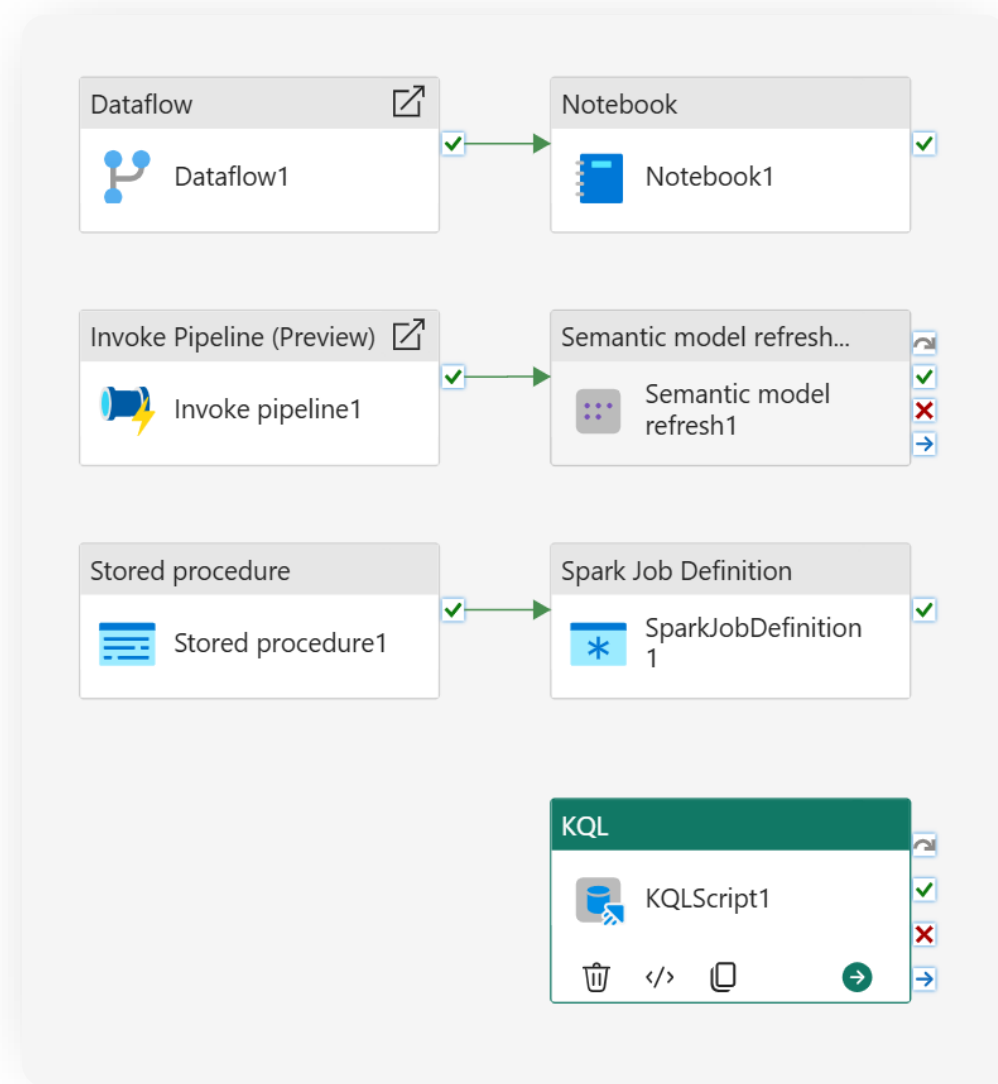
**On Skip** is triggered when an activity is skipped.

This can happen due to conditional logic in the pipeline that determines the activity is not needed.

It can be useful in scenarios where certain steps are required only under specific conditions. For example, if a data source is empty, you may skip the data transformation step.

# Coordination and consequences


Invoke of... almost anything...











# Coordination and consequences

## Specification of actions

Copy data

 Zdroj 1

General Source Destination Mapping Settings

Name \*

Zdroj 1 [Learn more](#)

Description

Refresh 1. zdroje dat

Activity state ⓘ

☒ Activated ☐ Deactivated

Timeout ⓘ

0.2:00:00

Retry ⓘ

3

Advanced

Retry interval (sec) ⓘ

600

Secure output ⓘ

☐

Secure input ⓘ

☐

# Orchestration of notebooks

Directed Acyclic Graph (DAG). POZOR! Neplést s DAX!

## %run magic command

But before the DAG arrives, the wizard appears...

This is one of the ways to have one notebook trigger another. Unfortunately, it is not a very transparent method of running notebooks, as in many cases, you may want to run multiple notebooks in a specific order due to dependencies. This approach requires placing the triggering code in each of the notebooks, which leads to the mentioned lack of clarity. Moreover, this variant only supports up to **5 levels of nesting**. As soon as a fifth notebook attempts to trigger another one, an error will be returned.

Within this execution, however, all resources are shared, because no actual separate notebook run occurs — instead, its code is injected into the current one. This means that if the included notebook contains libraries or variables, these will also be transferred into the calling notebook and may cause conflicts or overwrite existing variables. In short, this creates a very high probability of collisions.

```
%run [-b/--builtin -c/--current] [script_file.py/.sql] [variables ...]
```

## notebookutils.notebook

And this wizard has quite a few more tricks up his sleeve.

The most important helpers for optimal notebook orchestration:

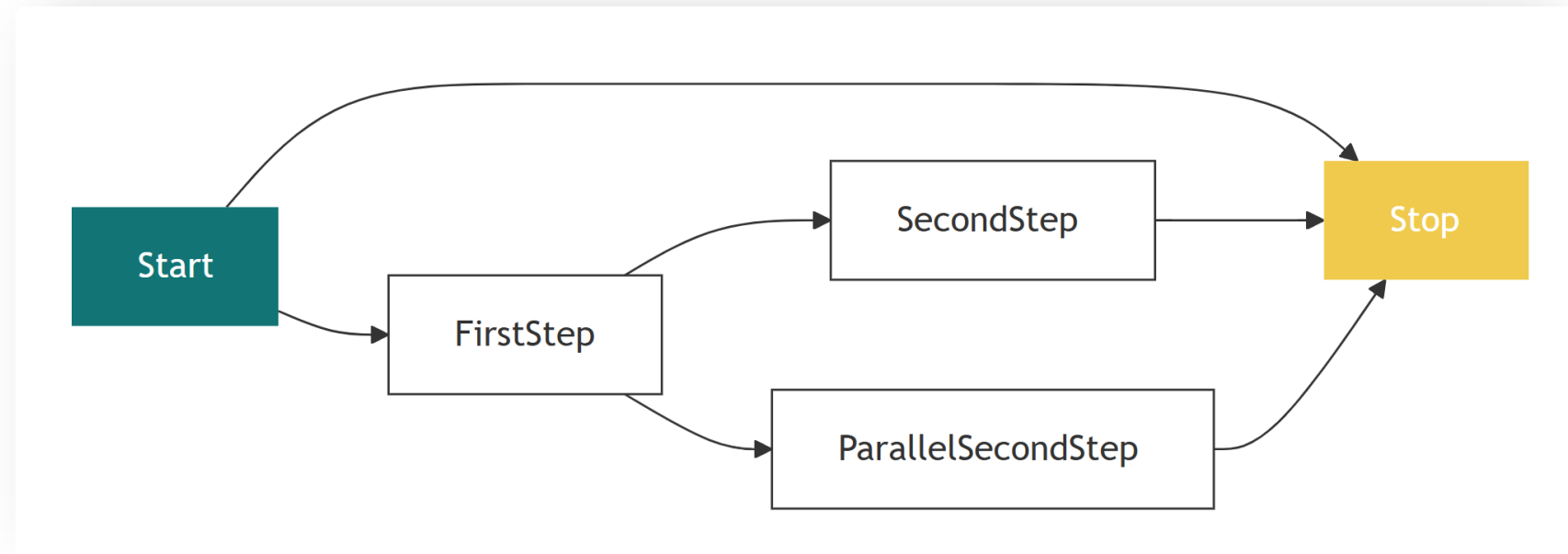
```
notebookutils.notebook.run("notebook name", <timeoutSeconds>, <parameterMap>, <workspaceId>)
```

```
notebookutils.notebook.runMultiple(<DAG>, <config>)
```

When a notebook is triggered, **a new session is not created!** Everything runs under the existing session of the notebook that initiated the execution, and the workload is “only” redistributed across available nodes within the active session.

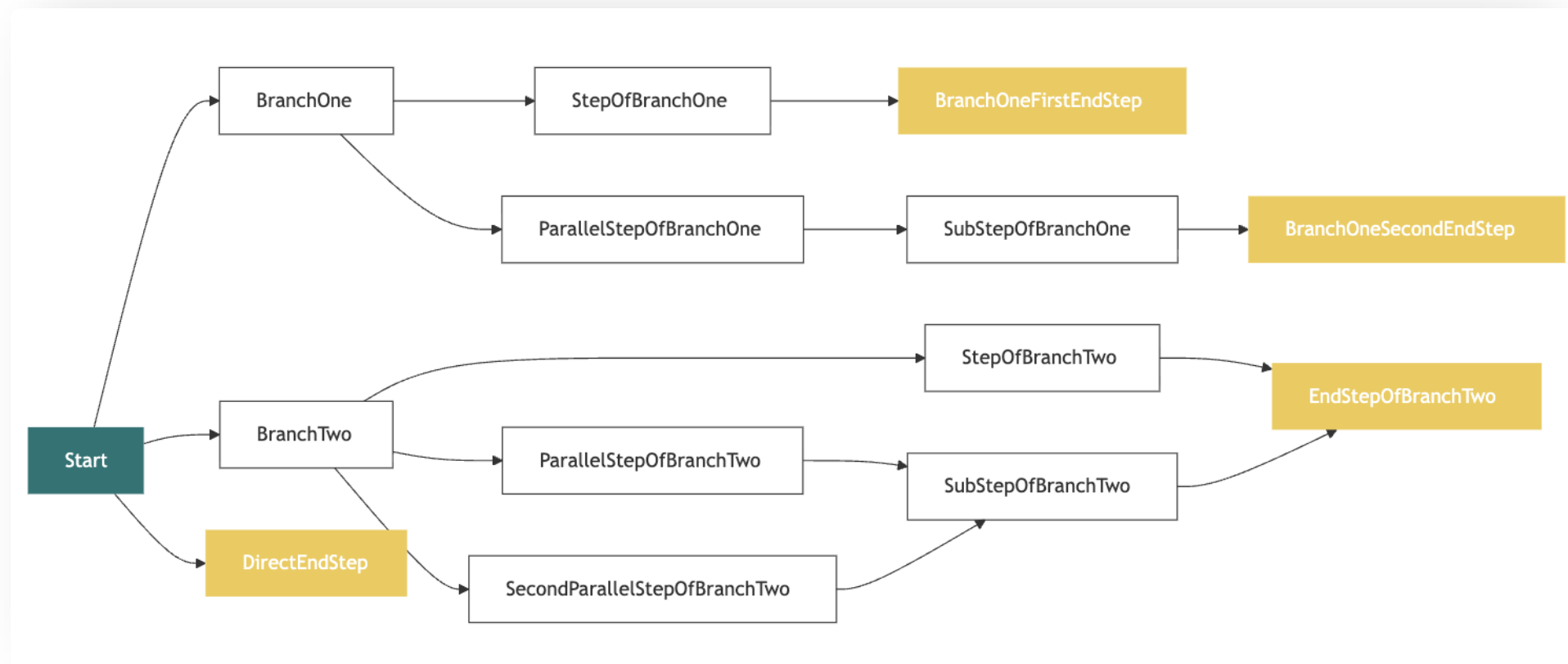
# Showcase of DAG

Illustration for better understanding



# Showcase of DAG

Illustration for better understanding





# Supported definition of DAG

And this is how we write it down...

```
DAG = {
  "activities": [
    {
      "name": "notebook1", # activity name, must be unique
      "path": "notebook1", # notebook path
      "timeoutPerCellInSeconds": 90, # max timeout for each cell, default to 90 seconds
      "args": {"param1": "value1"}, # notebook parameters
      "workspace": "workspace1", # workspace name, default to current workspace
      "retry": 0, # max retry times, default to 0
      "retryIntervalInSeconds": 0, # retry interval, default to 0 seconds
      "dependencies": [] # list of activity names that this activity depends on
    },
    {
      "name": "notebook2",
      "path": "notebook2",
      "timeoutPerCellInSeconds": 120,
      "args": {
        "useRootDefaultLakehouse": True, # set useRootDefaultLakehouse as True
        "param1": "@activity('notebook1').exitValue()" # use exit value of notebook1
      },
      # to ignore that child notebook attach a different default lakehouse.
      "retry": 1,
      "retryIntervalInSeconds": 10,
      "dependencies": ["notebook1"]
    }
  ],
  "timeoutInSeconds": 43200, # max timeout for the entire pipeline, default to 12 hours
  "concurrency": 50 # max number of notebooks to run concurrently, default to 50, 0 means unlimited
}

notebookutils.notebook.runMultiple(DAG, {"displayDAGViaGraphviz": False, "showArgs": False, "showTime": False})
```

# Supported definition of DAG

## Example of implementation

```
1 silver_load_flow_dag = {
2     "activities": [
3         {"name": "Load_To_Silver_Apps", "path": "Load_To_Silver_Apps", "timeoutPerCellInSeconds": 600, "dependencies": []},
4         {"name": "Load_To_Silver_Capacities", "path": "Load_To_Silver_Capacities", "timeoutPerCellInSeconds": 600, "dependencies": []},
5         {"name": "Load_To_Silver_Domains", "path": "Load_To_Silver_Domains", "timeoutPerCellInSeconds": 600, "dependencies": []},
6         {"name": "Load_To_Silver_Links", "path": "Load_To_Silver_Links", "timeoutPerCellInSeconds": 600, "dependencies": []},
7         {"name": "Load_To_Silver_Deployment_Pipelines", "path": "Load_To_Silver_Deployment_Pipelines", "timeoutPerCellInSeconds": 600, "dependencies": []},
8         {"name": "Load_To_Silver_Fact_Link", "path": "Load_To_Silver_Fact_Link", "timeoutPerCellInSeconds": 7200, "dependencies": ["Load_To_Silver_Links"]},
9         {"name": "Load_To_Silver_Users", "path": "Load_To_Silver_Users", "timeoutPerCellInSeconds": 600, "dependencies": []},
10        {"name": "Load_To_Silver_Gateways", "path": "Load_To_Silver_Gateways", "timeoutPerCellInSeconds": 600, "dependencies": []},
11        {"name": "Load_To_Silver_Groups", "path": "Load_To_Silver_Groups", "timeoutPerCellInSeconds": 600, "dependencies": []},
12        {"name": "Load_To_Silver_GroupsUsers", "path": "Load_To_Silver_GroupsUsers", "timeoutPerCellInSeconds": 600, "dependencies": []},
13        {"name": "Load_To_Silver_UserLicenses", "path": "Load_To_Silver_UserLicenses", "timeoutPerCellInSeconds": 600, "dependencies": []},
14        {"name": "Load_To_Silver_ServicePrincipals", "path": "Load_To_Silver_ServicePrincipals", "timeoutPerCellInSeconds": 600, "dependencies": []},
15        {"name": "Load_To_Silver_Workspaces", "path": "Load_To_Silver_Workspaces", "timeoutPerCellInSeconds": 600, "dependencies": ["Load_To_Silver_Capacities", "Load_To_Silver_Domains"]},
16        {"name": "Load_To_Silver_SensitivityLabels", "path": "Load_To_Silver_SensitivityLabels", "timeoutPerCellInSeconds": 600, "dependencies": ["Load_To_Silver_Users", "Load_To_Silver_Groups"]},
17        {"name": "Load_To_Silver_Imports", "path": "Load_To_Silver_Imports", "timeoutPerCellInSeconds": 600, "dependencies": ["Load_To_Silver_Subscriptions"]},
18        {"name": "Load_To_Silver_ActivityEvents", "path": "Load_To_Silver_ActivityEvents", "timeoutPerCellInSeconds": 3600, "dependencies": ["Load_To_Silver_Subscriptions"]},
19        {"name": "Load_To_Silver_Subscriptions", "path": "Load_To_Silver_Subscriptions", "timeoutPerCellInSeconds": 600, "dependencies": ["Load_To_Silver_Workspaces", "Load_To_Silver_SensitivityLabels"]},
20        {"name": "Load_To_Silver_Datasources", "path": "Load_To_Silver_Datasources", "timeoutPerCellInSeconds": 7200, "dependencies": ["Load_To_Silver_Users", "Load_To_Silver_Groups"]},
21        {"name": "Load_To_Silver_WS_Scanner", "path": "Load_To_Silver_WS_Scanner", "timeoutPerCellInSeconds": 7200, "dependencies": ["Load_To_Silver_Datasources", "Load_To_Silver_Users", "Load_To_Silver_Groups"]}
22    ]
23 }
```

[21] ✓ PySpark (Python) ▾

... ▾

+ Code + Markdown

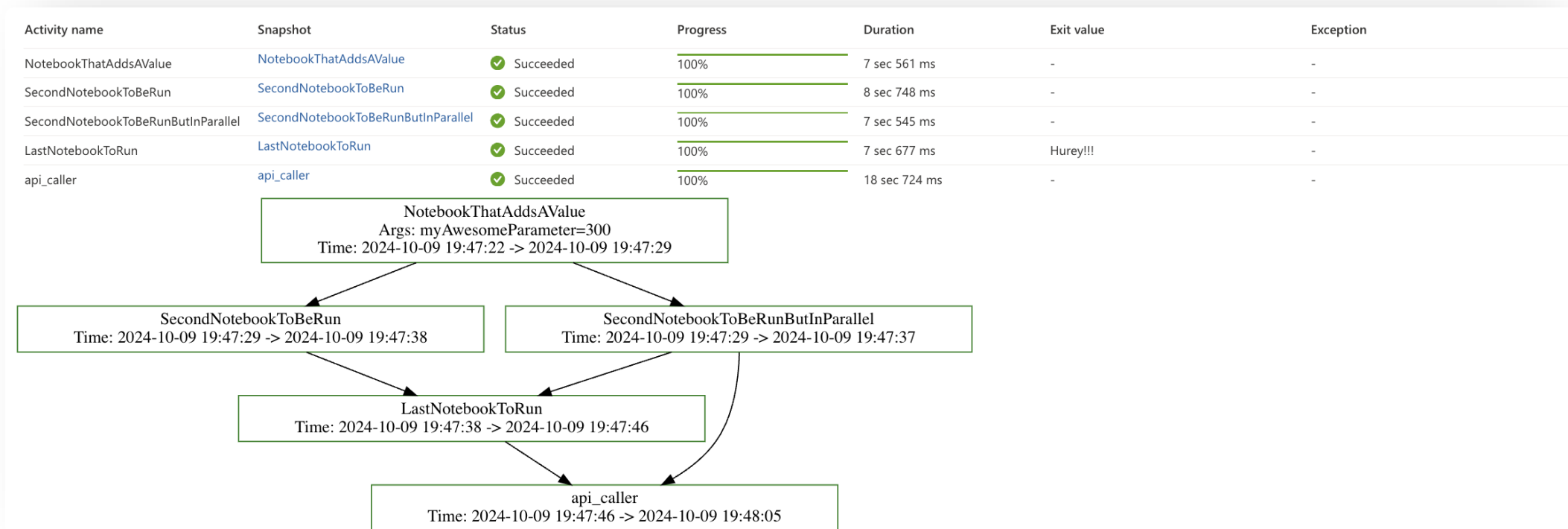
Run load for all notebooks to silver

```
1 notebookutils.notebook.runMultiple(silver_load_flow_dag, {"displayDAGViaGraphviz": False})
```

PySpark (Python) ▾

# Displayed DAG chart after execution

with a graphical chart addition...



# There is a catch

one more... that you need to have in our mind.

The PySpark notebook can't execute Python notebooks.

Python notebooks can't execute PySpark notebooks.

The screenshot shows a PySpark notebook interface. The top toolbar includes icons for file operations, a 'Run all' button, session management (Standard session), environment selection (PySpark (Python)), workspace selection (Workspace default), Data Wrangler, and Copilot. The left sidebar shows the 'Explorer' view. The main code editor contains a Python DAG definition:

```
1 import notebookutils
2
3 DAG = {
4     "activities": [
5         {
6             "name": "python_sleeper",
7             "path": "python_sleeper",
8             "timeoutPerCellInSeconds": 90,
9             "retry": 0,
10            "retryIntervalInSeconds": 0,
11            "dependencies": []
12        },
13    ],
14    "timeoutInSeconds": 43200,
15    "concurrency": 2
16 }
17
18
19 notebookutils.notebook.runMultiple(DAG)
```

Below the code, the execution output shows a runtime error:

```
[1] 16 sec - Session ready in 11 sec 995 ms. Command executed in 4 sec 955 ms by Štěpán Rešl on 10:44:19 AM, 8/28/25
...
331 "An error occurred while calling {0}{1}{2}. Trace:\n{3}\n".
332 format(target_id, ".", name, value))

Py4JJavaError: An error occurred while calling z:notebookutils.notebook.runMultiple.
: com.microsoft.spark.notebook.msutils.NotebookExecutionException: Fetch notebook content for python_sleeper failed with exception: Request to
preprocess?api-version=1 failed with status code: 400, response:ChildNotebook compute type
JupyterCompute does not match to main notebook compute type SparkCompute., response headers: Array(Content-Type: text/plain; charset=utf-8, Date: Thu, 28 Aug 2025
09:44:15 GMT, Server: Kestrel, Transfer-Encoding: chunked, Request-Context: appId=, x-ms-nbs-activity-spanId: 61d66239dale7926, x-ms-nbs-activity-traceId:
```

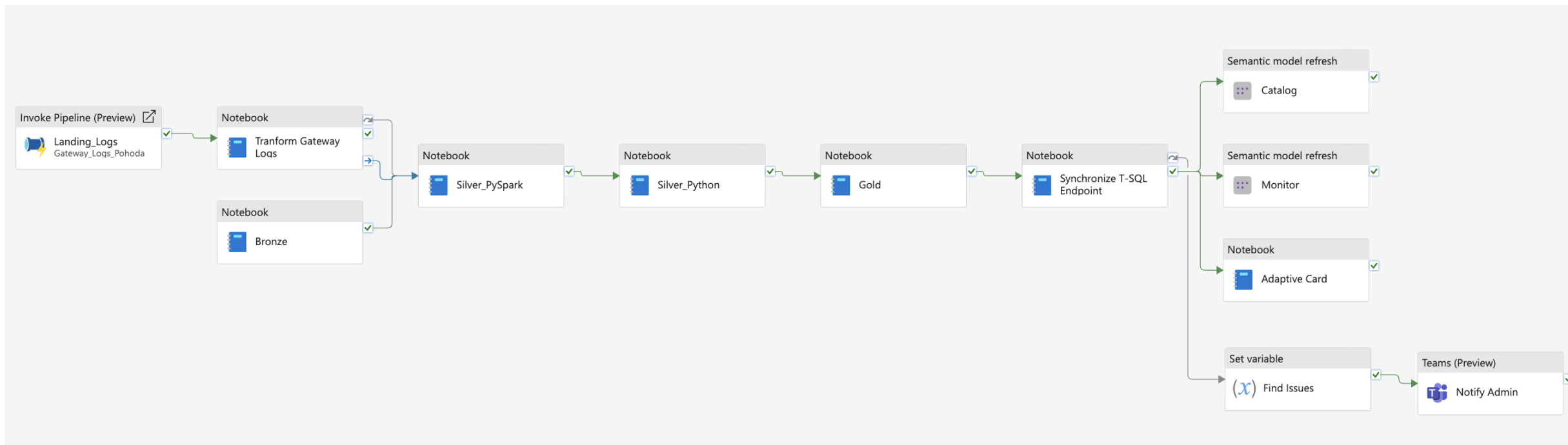
Abstract yellow geometric shapes, including a large triangle and a smaller parallelogram, on a light yellow background.

# All together

How it can look like...

# One pipeline that orchestrates data ingest and transformations

... including notification to admins and cards with changes in the MS Teams





**Data  
Brothers**

# **Thats all folks**

Thank you for attention.