



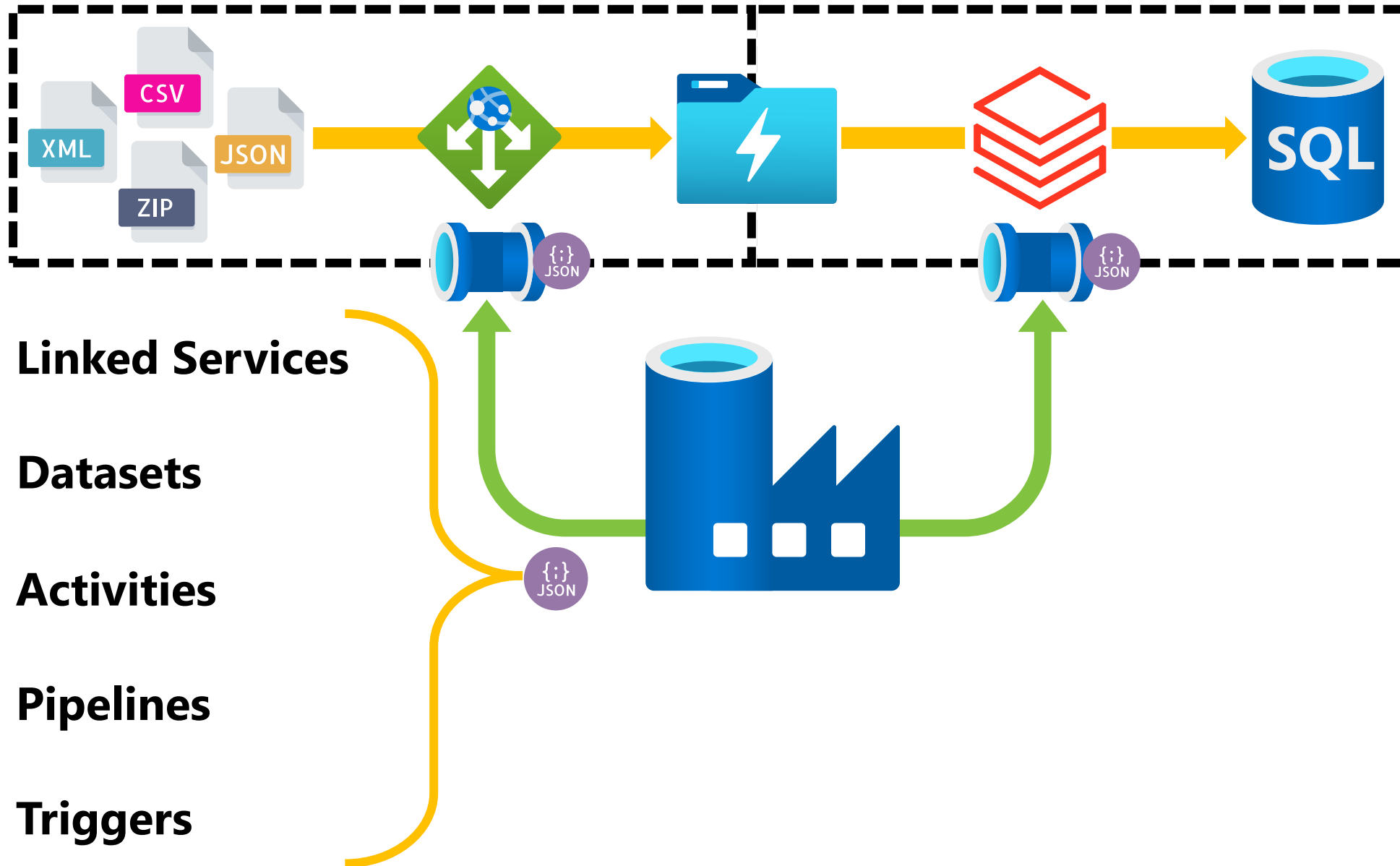
Module 5: Metadata Driven Pipelines

- Expressions
- Dynamic Pipeline
- Orchestration Framework
- procfwk.com

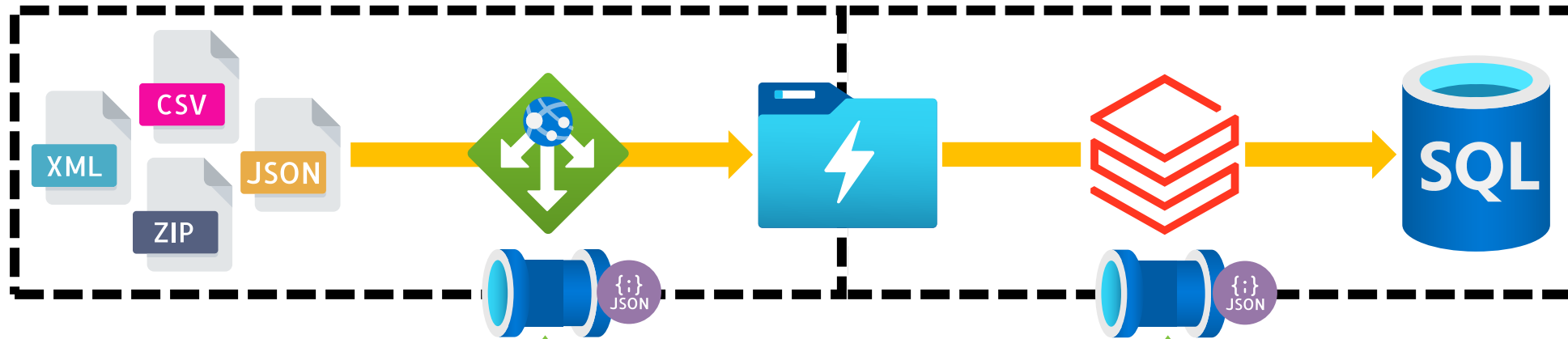
Expressions



Data Factory Components



Data Factory Components – Add Dynamic Content



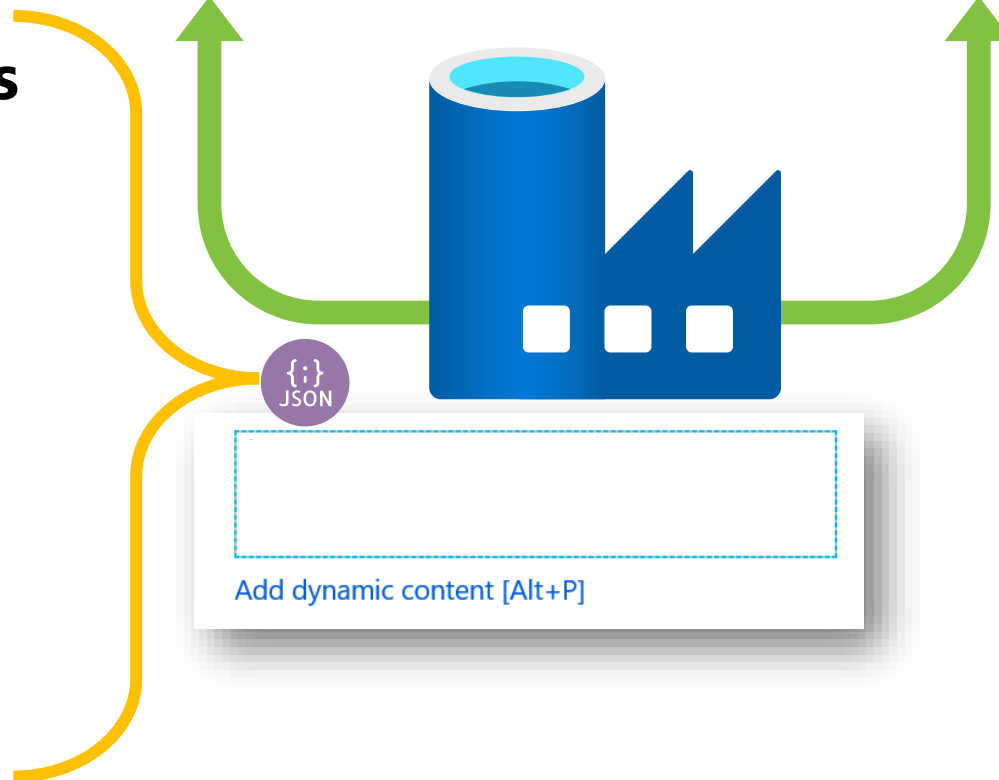
1 **Linked Services**

2 **Datasets**

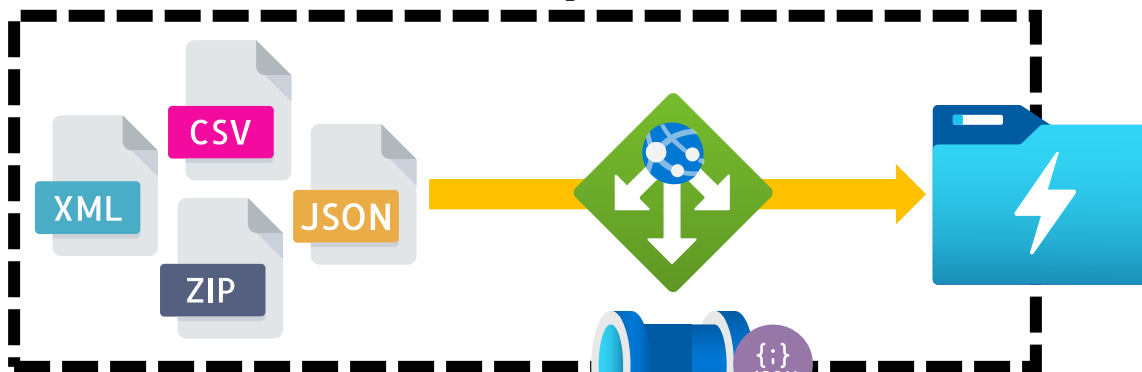
3 **Activities**

4 **Pipelines**

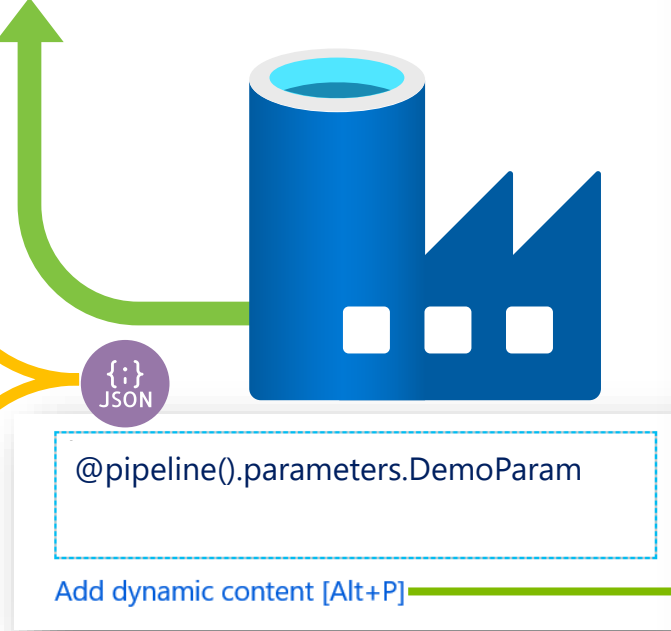
5 **Triggers**



Control Flow Expressions



- 1 **Linked Services**
- 2 **Datasets**
- 3 **Activities**
- 4 **Pipelines**
- 5 **Triggers**



Add dynamic content

`@pipeline().parameters.DemoParam`

[Clear contents](#)

[Filter...](#)

Use [expressions](#), [functions](#) or refer to [system variables](#).

System variables

Data factory name

Name of the data factory the pipeline run is running within

Pipeline Name

Name of the pipeline

Pipeline run ID

ID of the specific pipeline run

Pipeline trigger ID

ID of the trigger that invokes the pipeline

Pipeline trigger name

Name of the trigger that invokes the pipeline

Pipeline trigger time

Time when the trigger that invoked the pipeline. The trigger time is the actual fired time, not the sch...

Pipeline trigger type

Type of the trigger that invoked the pipeline (Manual, Scheduler)

Functions

✖ [Expand all](#)

▸ [Collection Functions](#)

▸ [Conversion Functions](#)

▸ [Date Functions](#)

▸ [Logical Functions](#)

▸ [Math Functions](#)

▸ [String Functions](#)

Control Flow Expressions

 Save  Save as template  Validate  Debug

```
"SomethingDynamic": {  
  "value": "@pipeline().parameters.DemoParam",  
  "type": "Expression"  
}
```

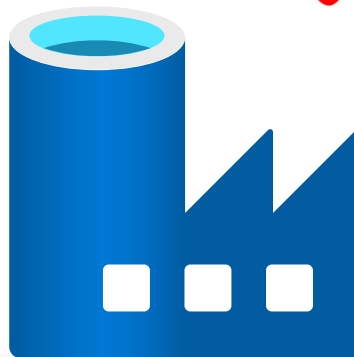
```
"SomethingDynamic": "@pipeline().parameters.DemoParam"
```

String Interpolation

```
"SomethingDynamic": "Hello World"
```

```
"parameters": {  
  "DemoParam": {  
    "type": "string",  
    "defaultValue": "World"  
  }  
}
```

```
"SomethingDynamic": "Hello @{pipeline().parameters.DemoParam}"
```



@pipeline().parameters.DemoParam

Add dynamic content [Alt+P]

Add dynamic content

@pipeline().parameters.DemoParam

[Clear contents](#)



Use [expressions](#), [functions](#) or refer to [system variables](#).

System variables

Data factory name

Name of the data factory the pipeline run is running within

Pipeline Name

Name of the pipeline

Pipeline run ID

ID of the specific pipeline run

Pipeline trigger ID

ID of the trigger that invokes the pipeline

Pipeline trigger name

Name of the trigger that invokes the pipeline

Pipeline trigger time

Time when the trigger that invoked the pipeline. The trigger time is the actual fired time, not the sch...

Pipeline trigger type

Type of the trigger that invoked the pipeline (Manual, Scheduler)

Functions

 [Expand all](#)

▸ [Collection Functions](#)

▸ [Conversion Functions](#)

▸ [Date Functions](#)

▸ [Logical Functions](#)

▸ [Math Functions](#)

▸ [String Functions](#)

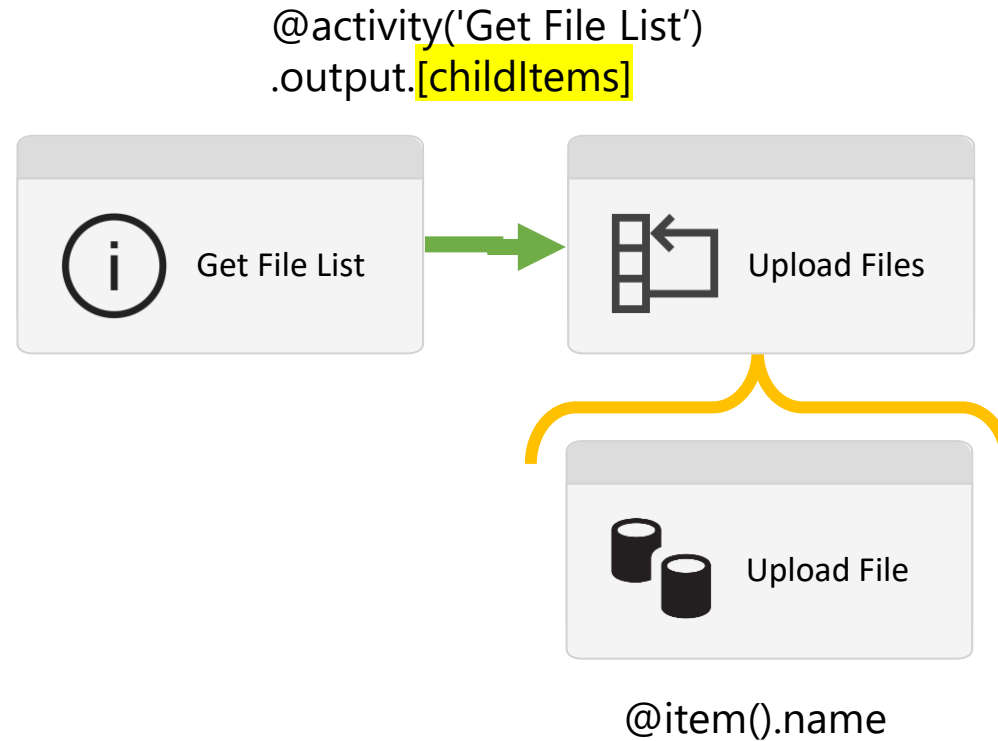
Dynamic Pipelines



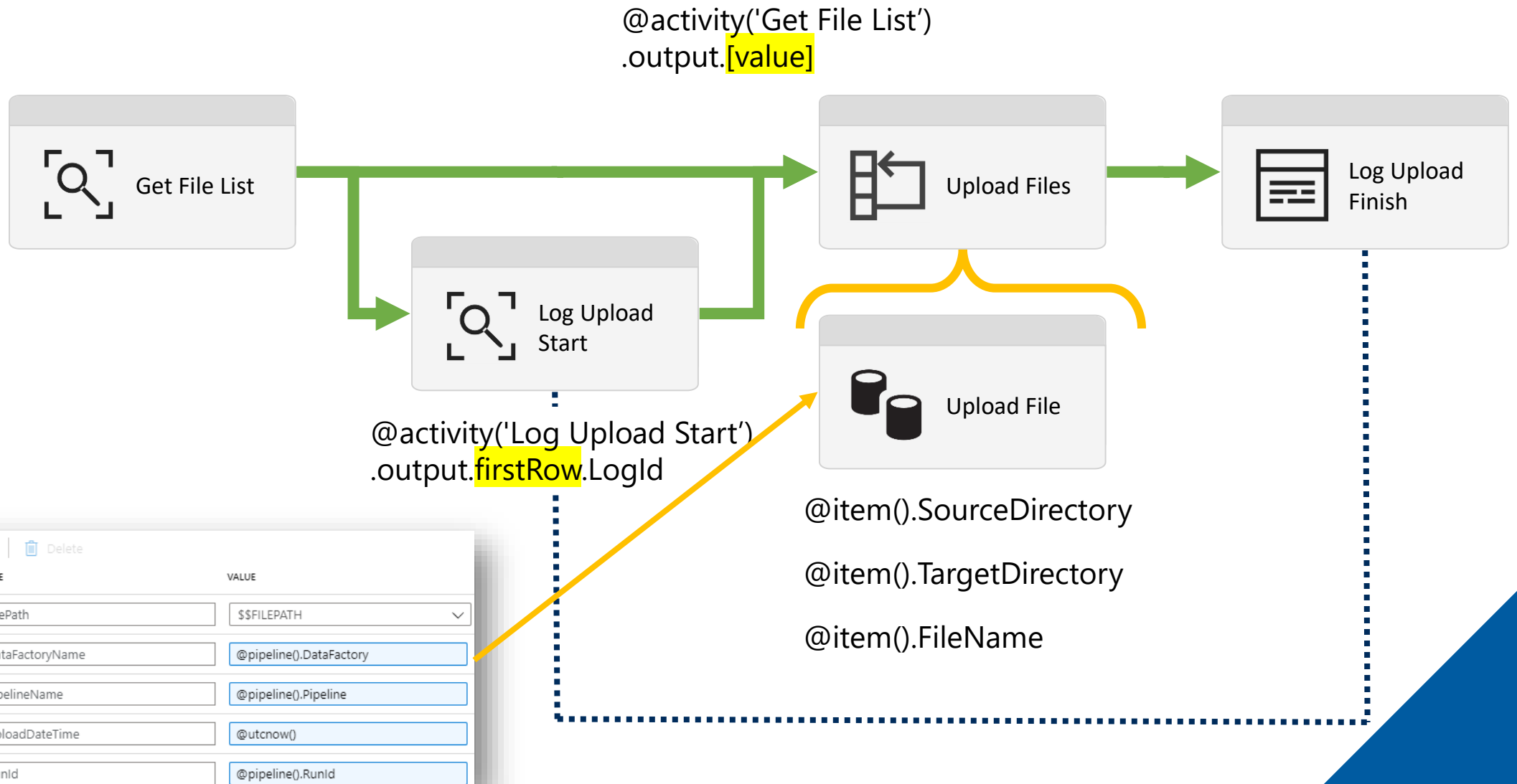
A blue cylinder is shown from a top-down perspective. The top of the cylinder is a light blue ring. Inside this ring, the word "DEMO" is written in white, bold, sans-serif capital letters. The rest of the cylinder is a solid blue color.

DEMO

Demo Pipeline 1 – Data Discovery and Upload



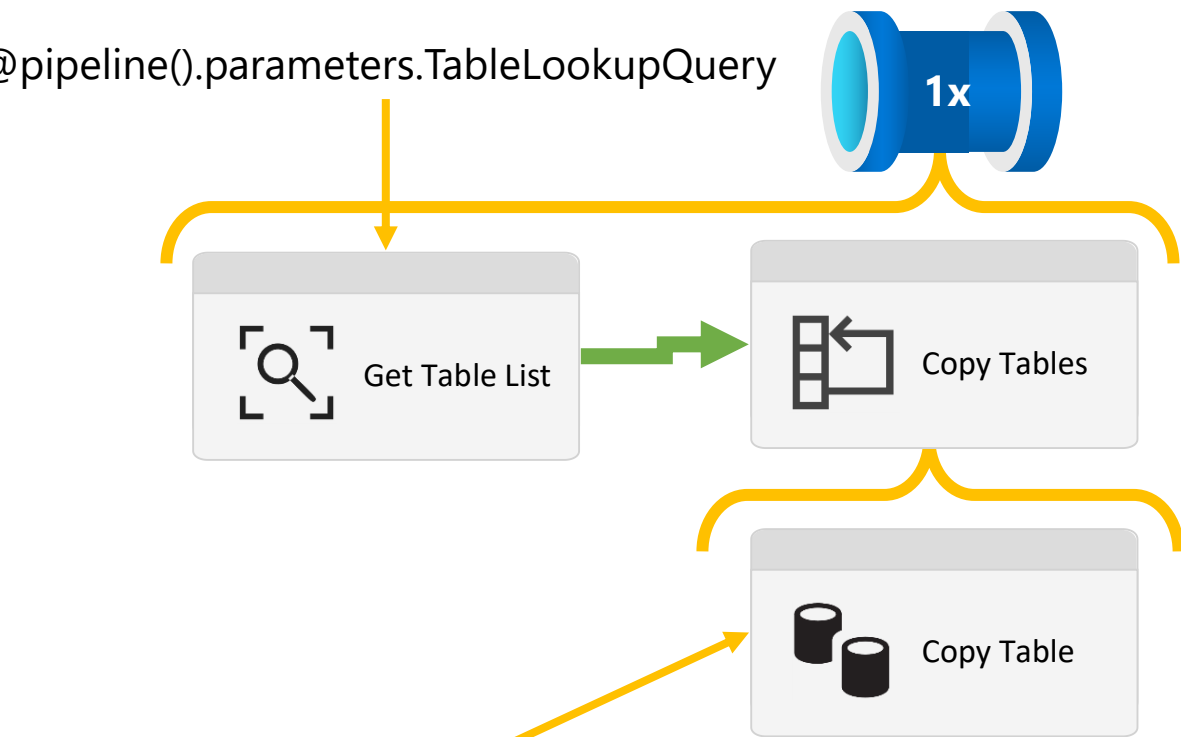
Demo Pipeline 2 – Simple Metadata and Upload



Demo Pipeline 3 – Lazy SQLDB Replication

```
SELECT s.name AS SchemaName, o.name AS TableName FROM sys.objects o  
INNER JOIN sys.schemas s ON o.schema_id = s.schema_id WHERE o.[type] = 'U'
```

@pipeline().parameters.TableLookupQuery



```
IF OBJECT_ID(  
    '@{item().SchemaName}.@{item().TableName}'  
) IS NOT NULL
```

```
TRUNCATE TABLE @item().SchemaName.@item().TableName
```

1x

1x

@pipeline().parameters.SourceConnectionSecret

@pipeline().parameters.TargetConnectionSecret

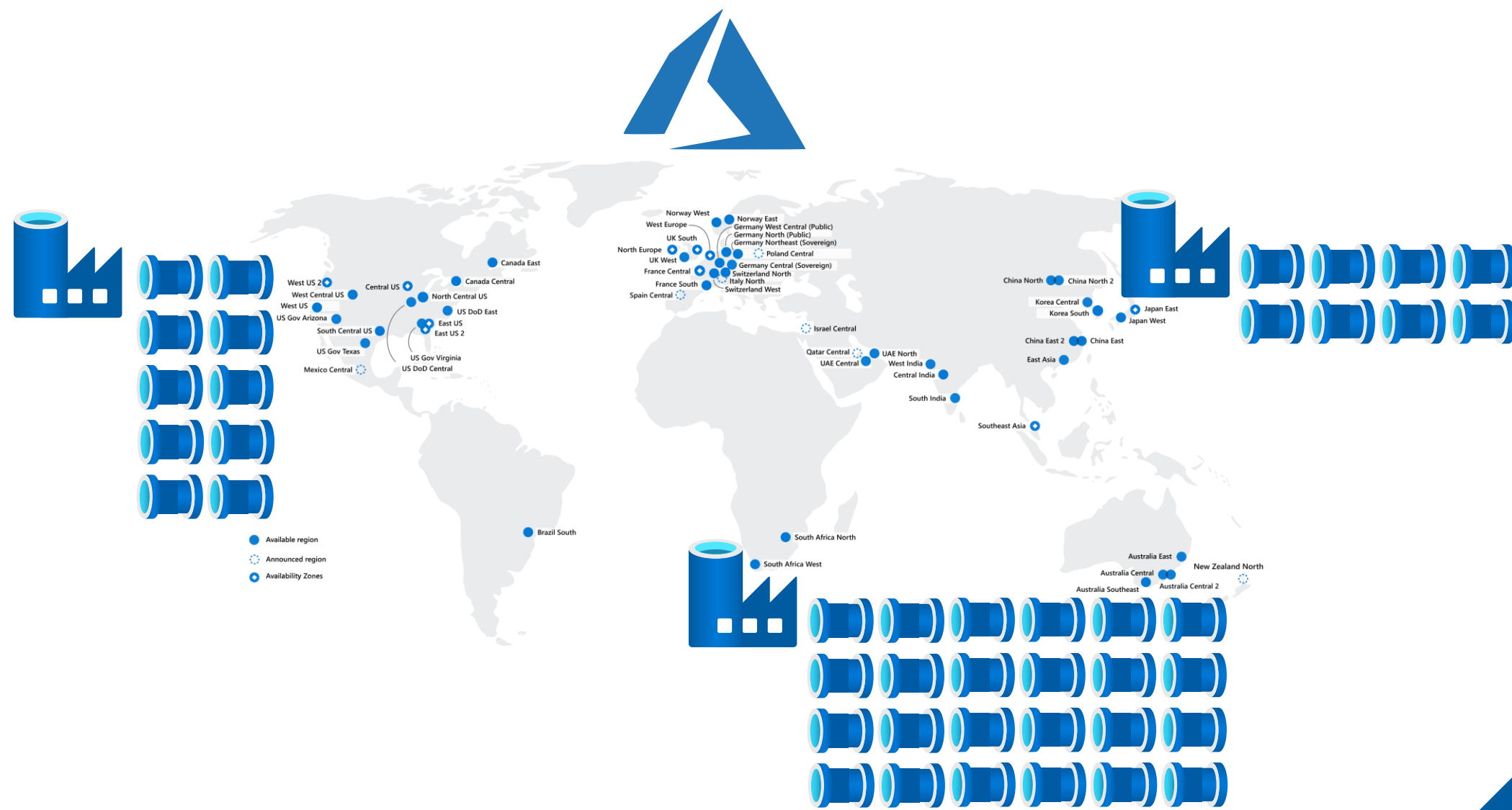
@dataset().LinkedServiceConnectionSecret

@linkedService().DBConnectionSecret

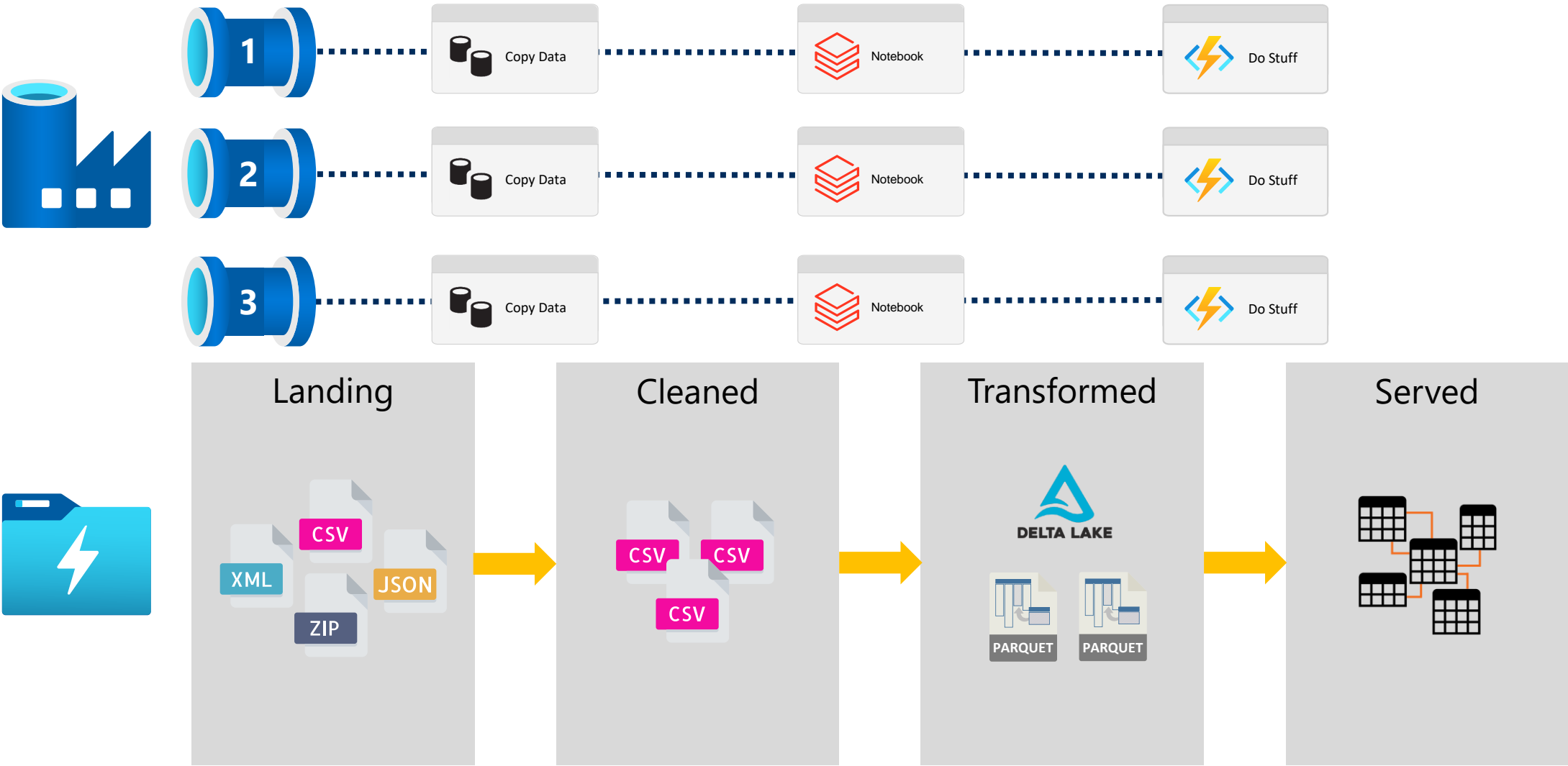
Orchestration Framework

A solid blue diagonal shape that starts from the bottom-left and extends towards the top-right corner of the slide, creating a modern, abstract background element.

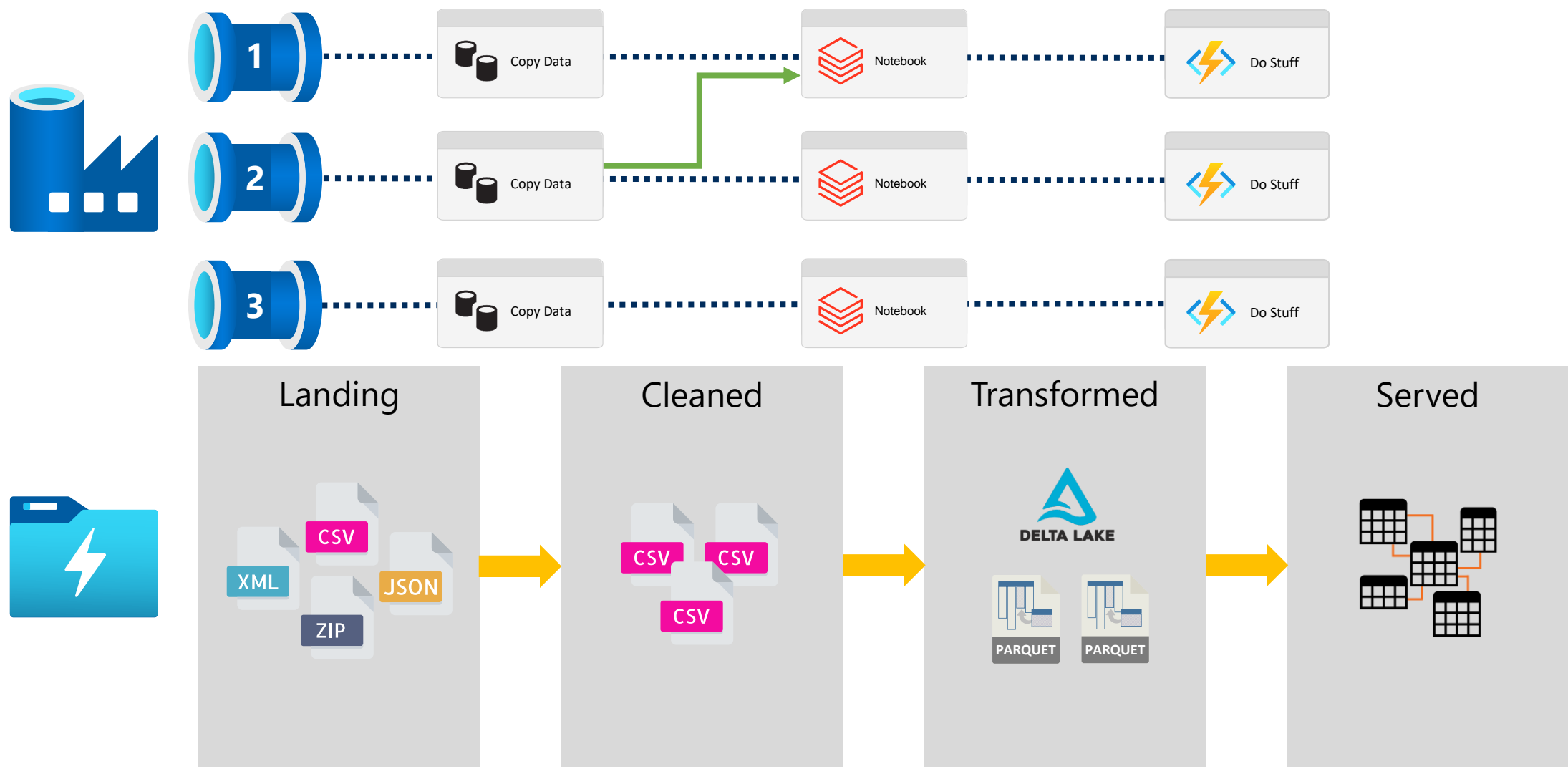
Problem: How should we structure our Data Factory Pipelines?



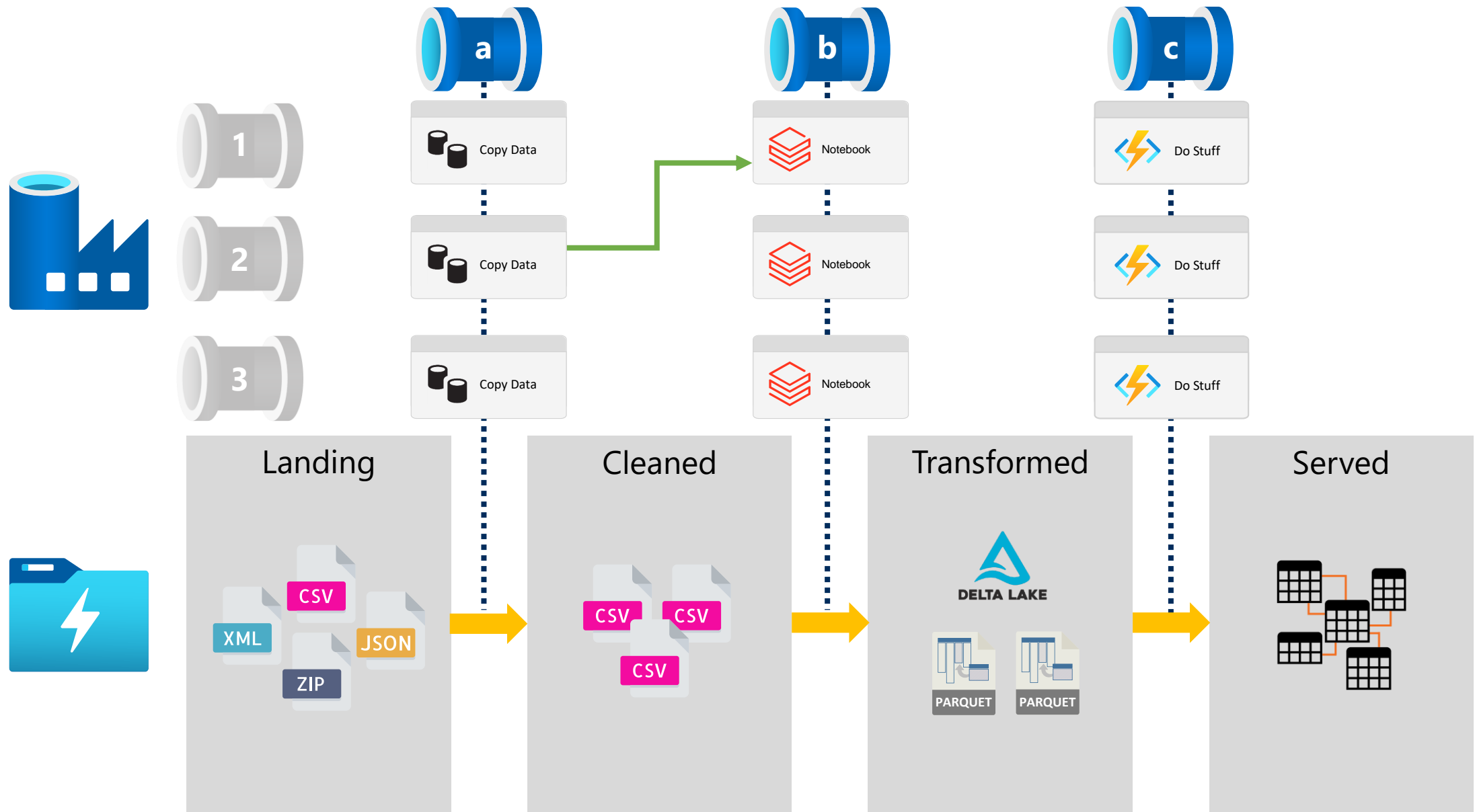
Problem



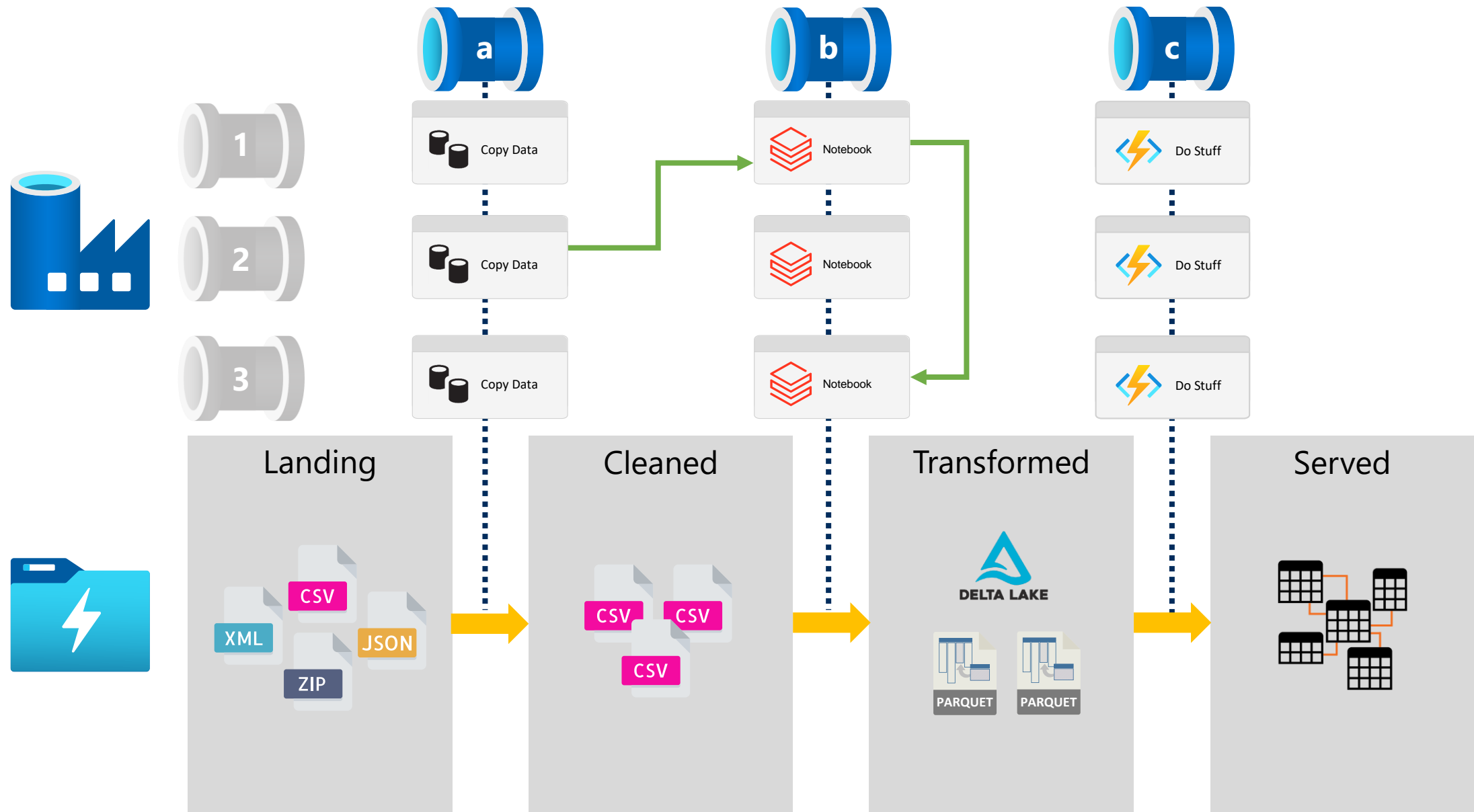
Problem



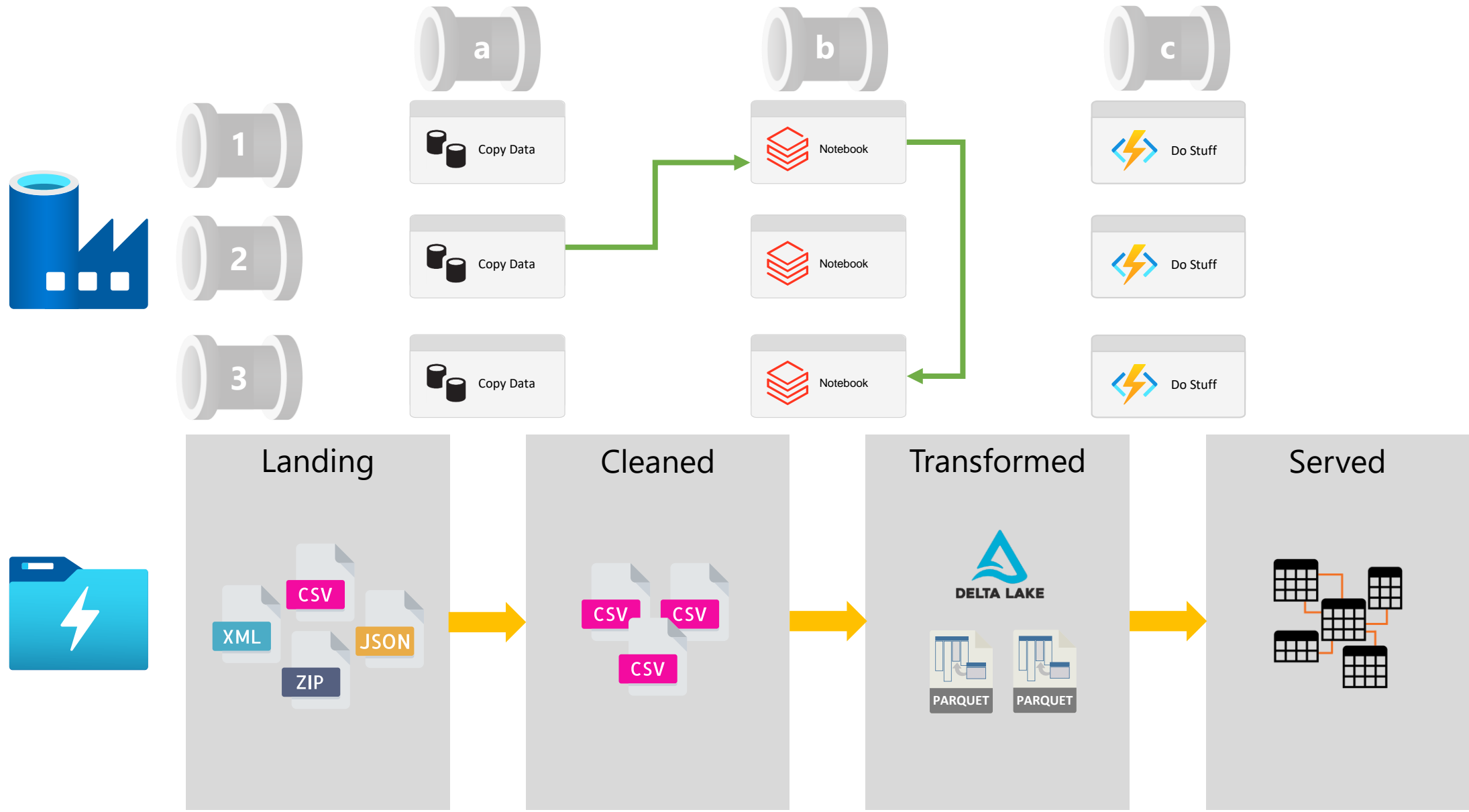
Problem




Problem

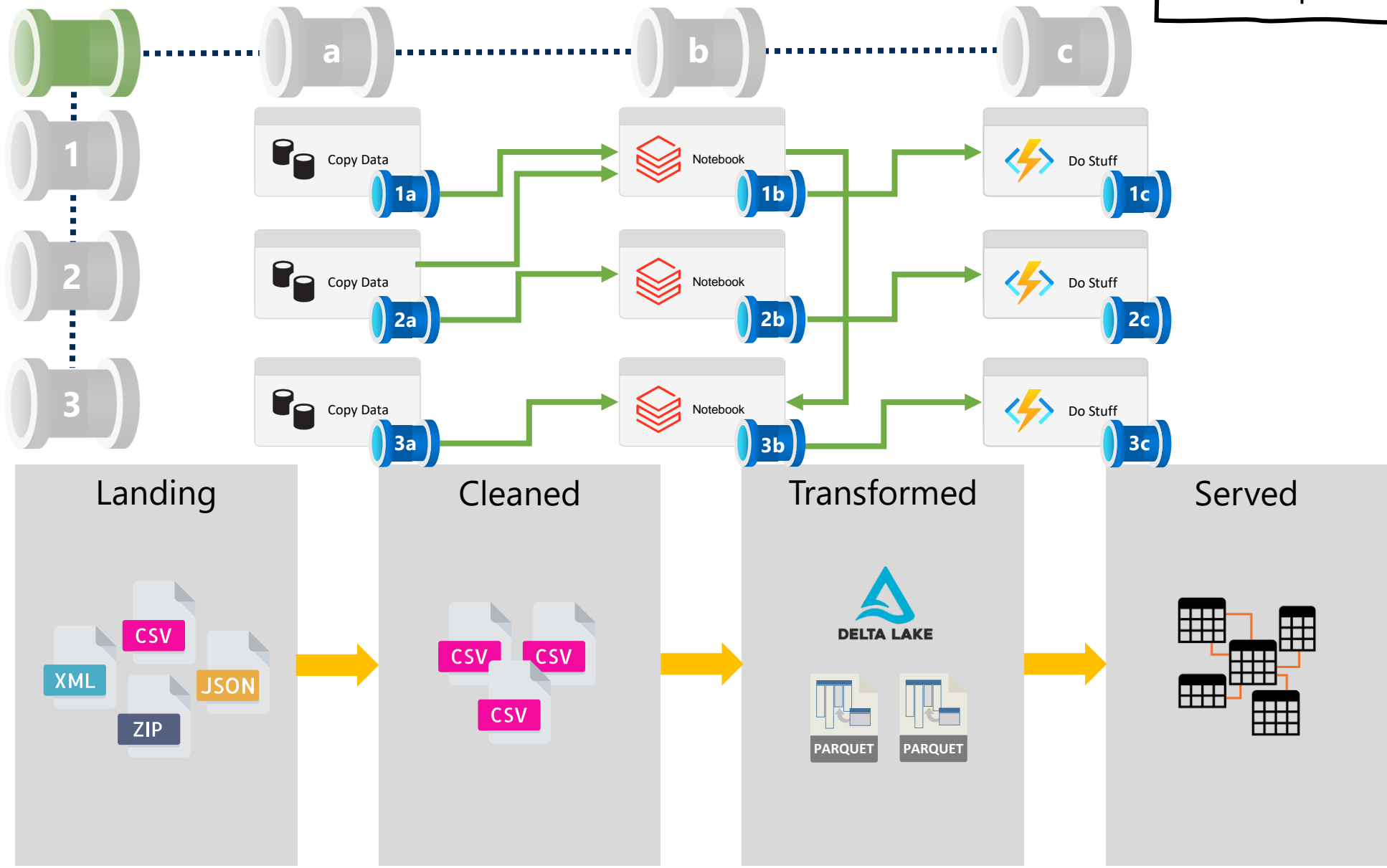
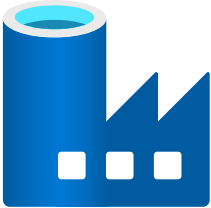


Problem

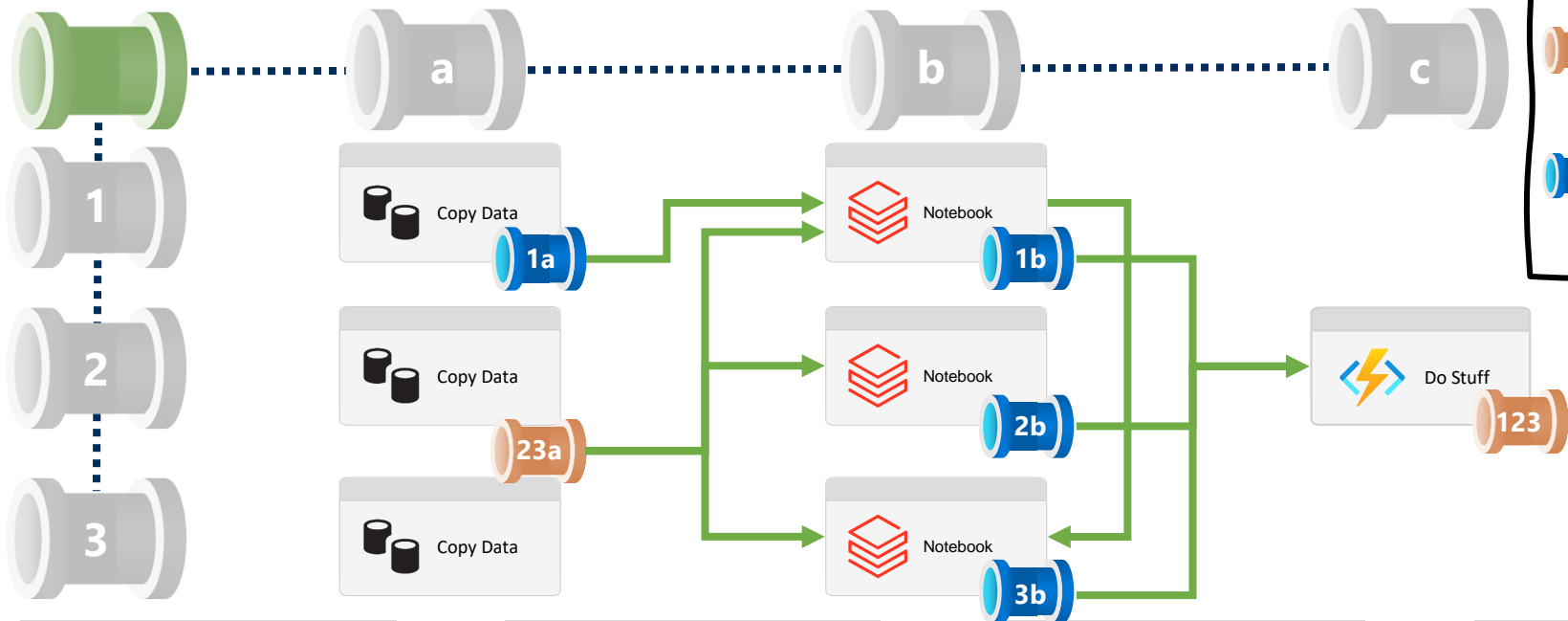
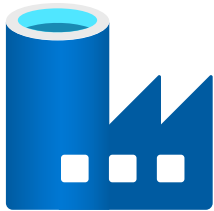


Problem

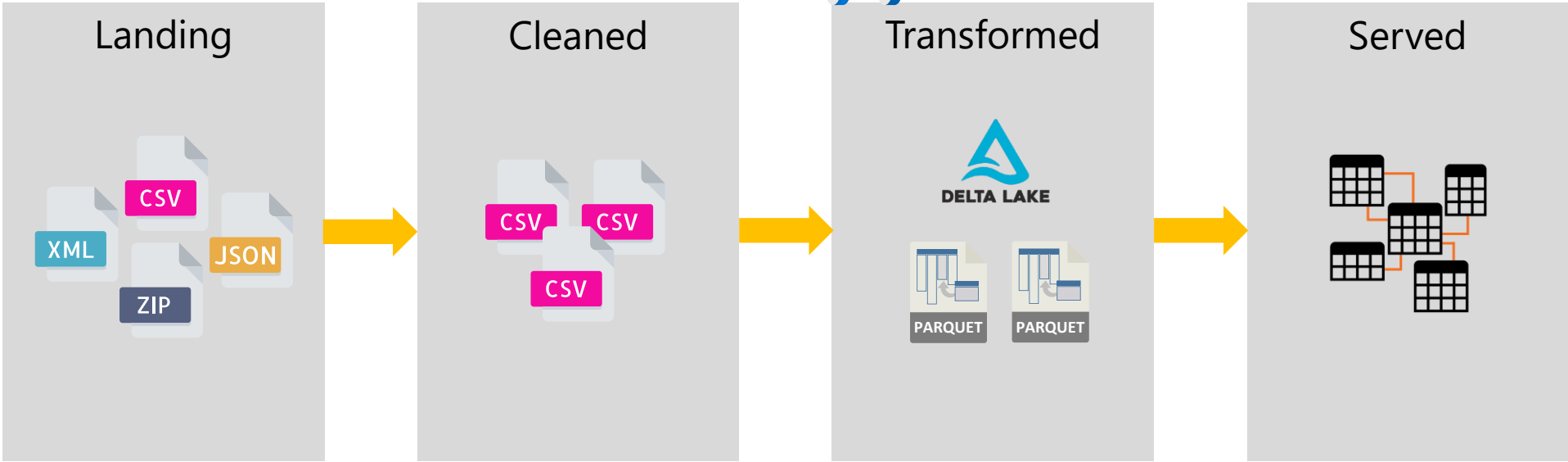
 Only 40 Activities per Pipeline.



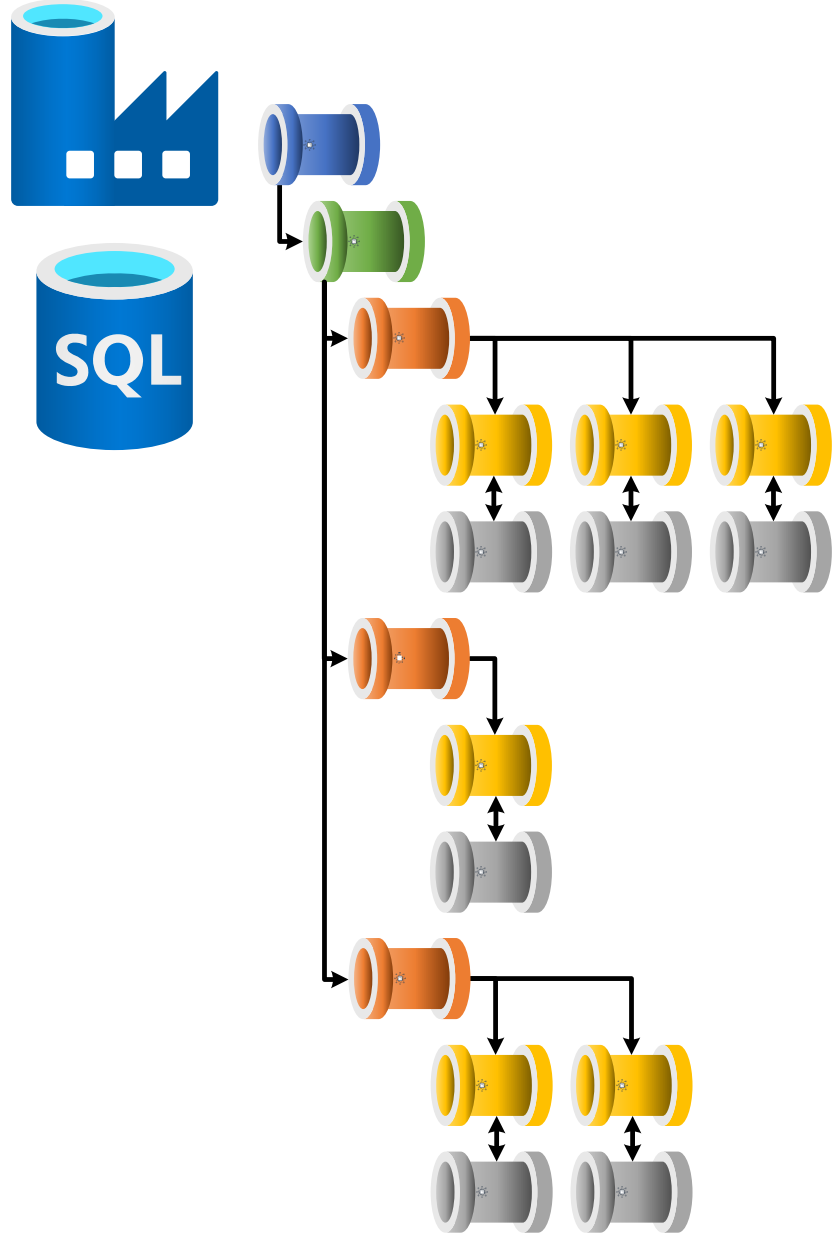
Problem



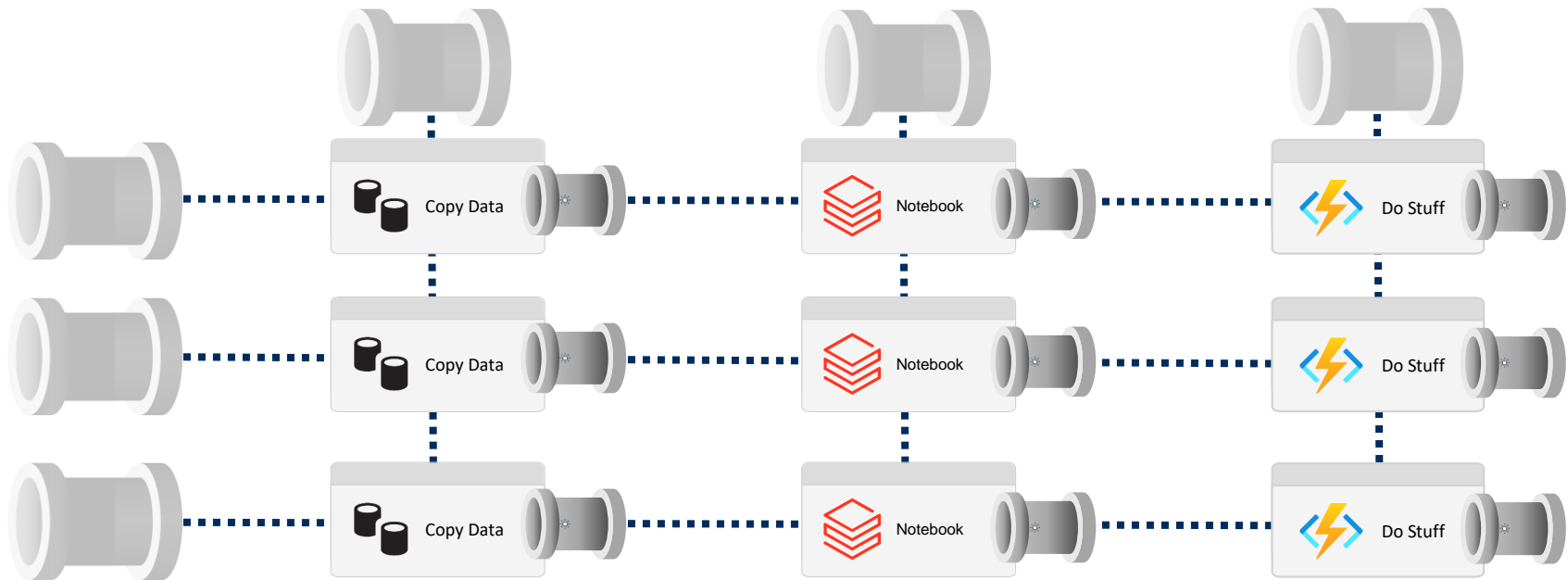
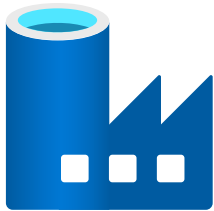
- Grandparent pipeline for all processing.
- Parent pipeline to consolidate work.
- Child pipelines for low level dependencies.



Solution: Use Metadata to Drive Data Factory Pipelines

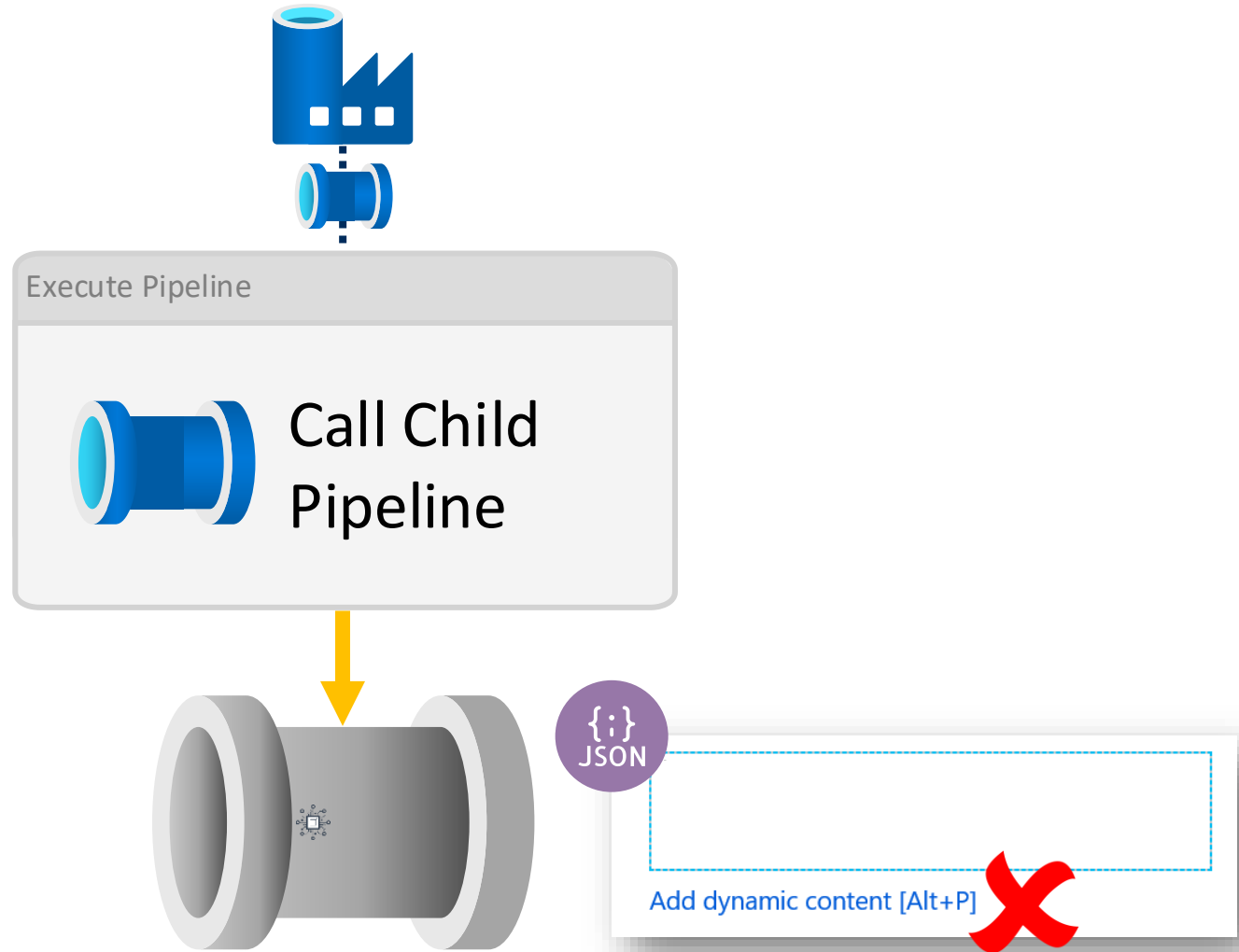


Solution

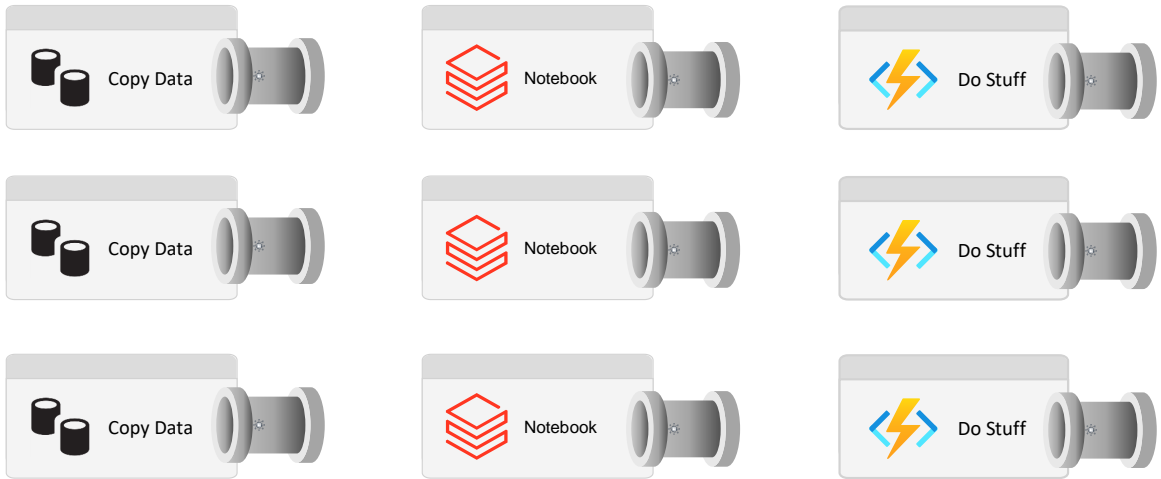
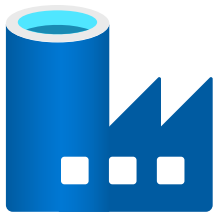


Stages	Pipelines	Stage	Pipeline
1	a	1	a
2	b	1	b
3	c	1	c
	d	2	d
	e	2	e
	f	3	f
	g	3	g
	h	3	h
	i	3	i

One More Problem



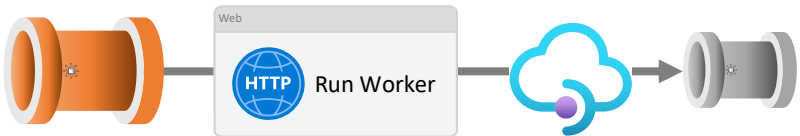
Calling Our Worker Pipelines



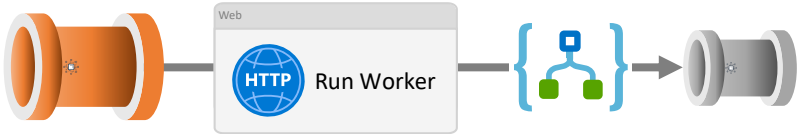
Stages	Pipelines
1	a
2	b
3	c
	d
	e
	f
	g
	h
	i

Stage	Pipeline
1	a
1	b
1	c
2	d
2	e
3	f
3	g
3	h
3	i

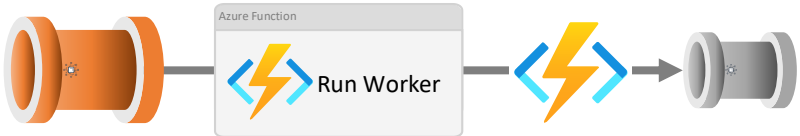
Option 1:



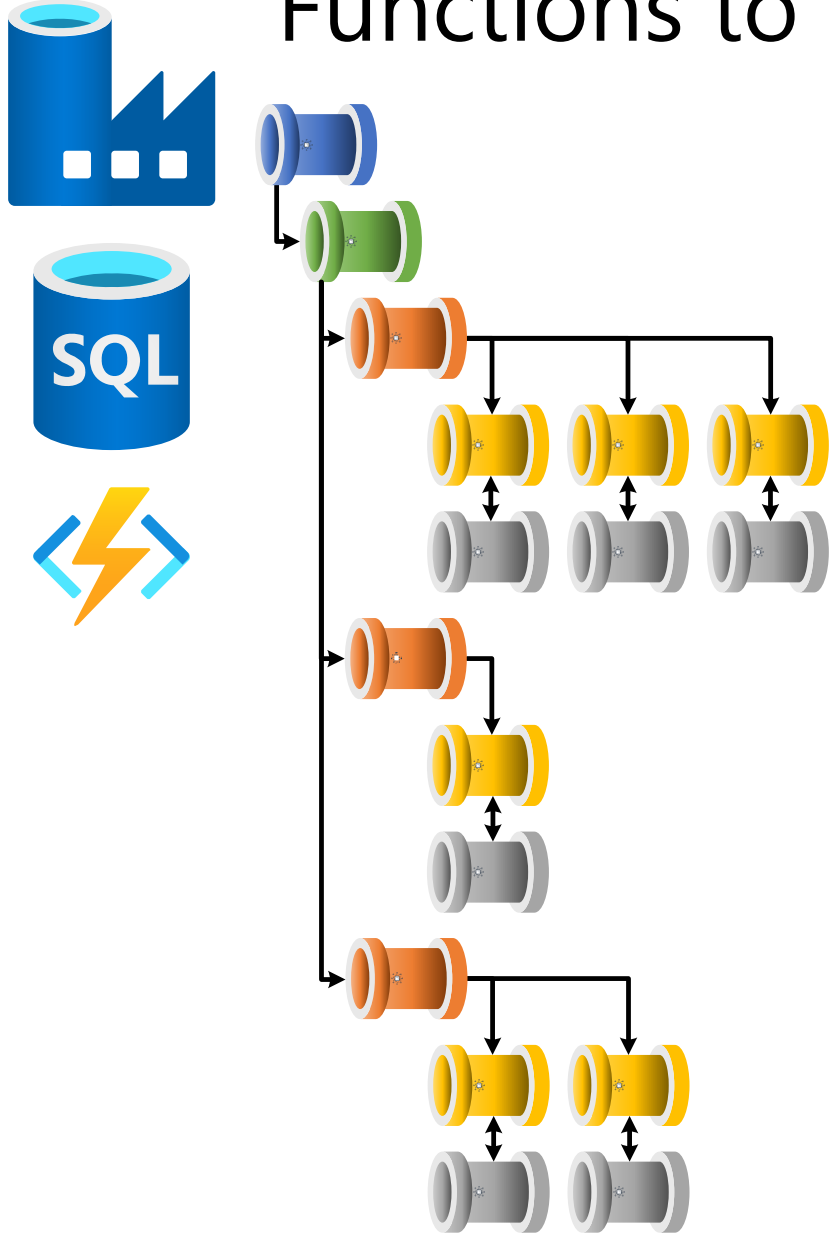
Option 2:



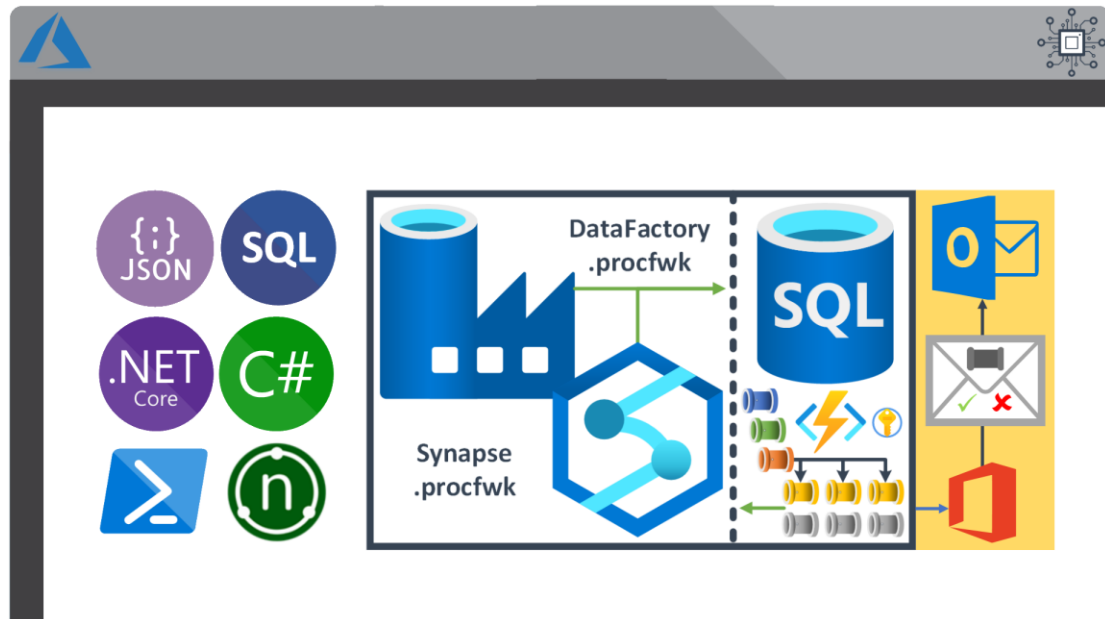
Option 3:



Solution: Use Metadata to Drive Data Factory Pipelines & Functions to Handle the Worker Execution



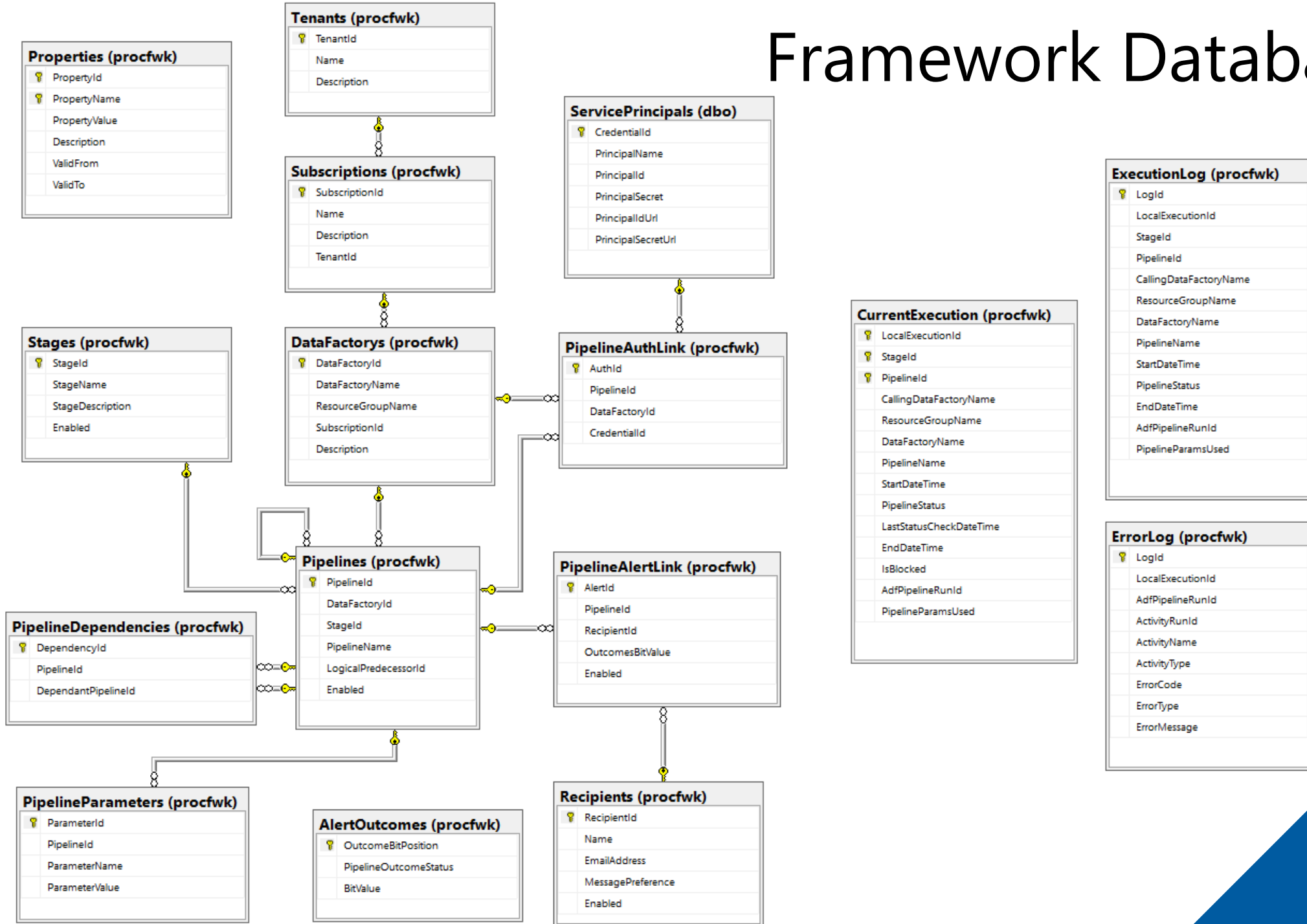
Introducing procfwk.com



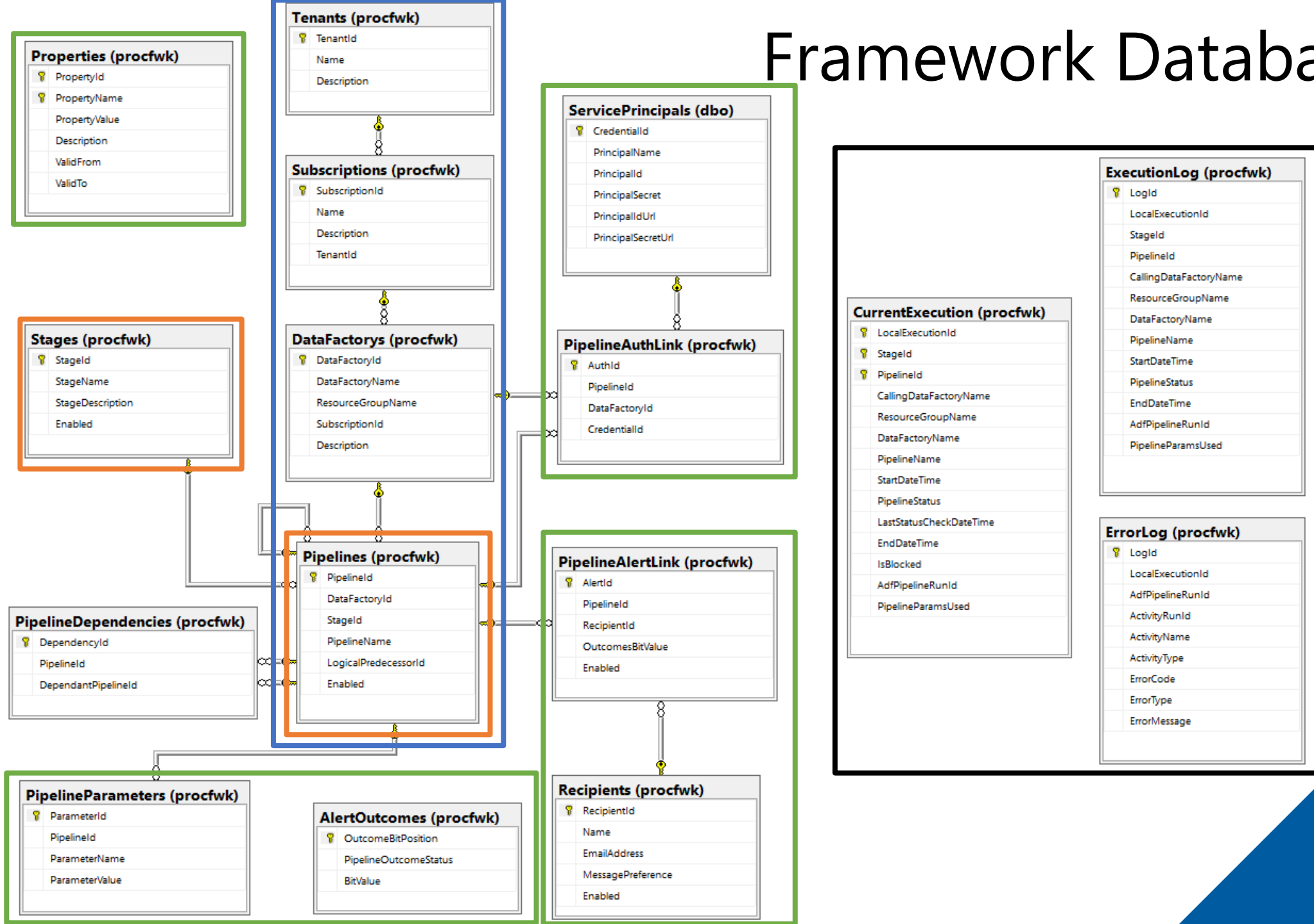
Procfwk Features

- 00 Granular metadata control.
- 00 Metadata integrity checking.
- 00 Global properties.
- 00 Complete pipeline dependency chains.
- 00 Execution restart-ability.
- 00 Parallel execution.
- 00 Full execution and error logs.
- 00 Operational dashboards.
- 00 Low cost orchestration.
- 00 Disconnection between framework and worker pipelines.
- 00 Cross Tenant/Subscription/Data Factory control flows.
- 00 Pipeline parameter support.
- 00 Simple troubleshooting.
- 00 Easy deployment.
- 00 Email alerting.
- 00 Automated testing.
- 00 Azure Key Vault integration.

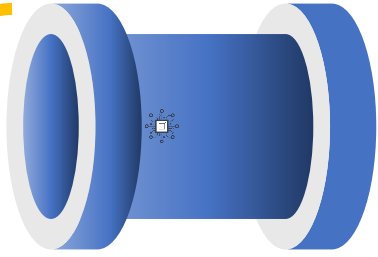
Framework Database



Framework Database

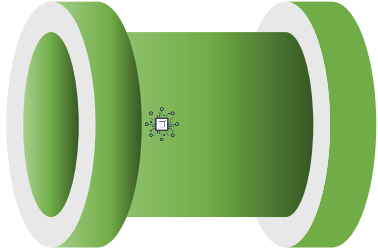


Pipeline Hierarchy



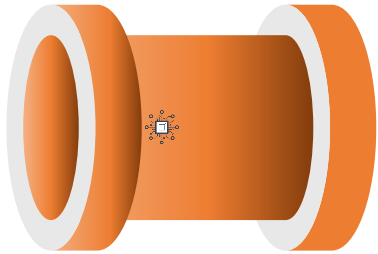
- Grandparent

Role: Optional level platform setup, for example, scale up/out compute services ready for the framework to run.



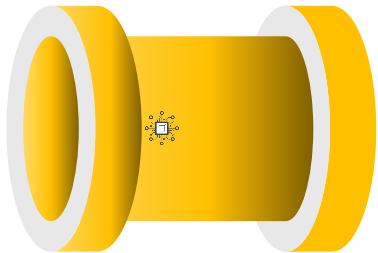
- Parent

Role: Execution run wrapper and execution stage iterator.



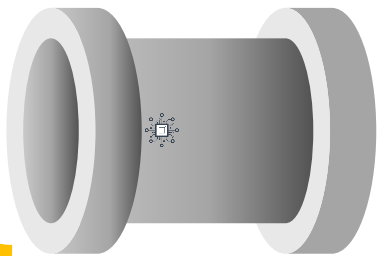
- Child

Role: Scale out triggering of worker pipelines within the execution stage.



- Infant

Role: Worker executor, monitor and reporting of the outcome for the single worker pipeline.



- Worker

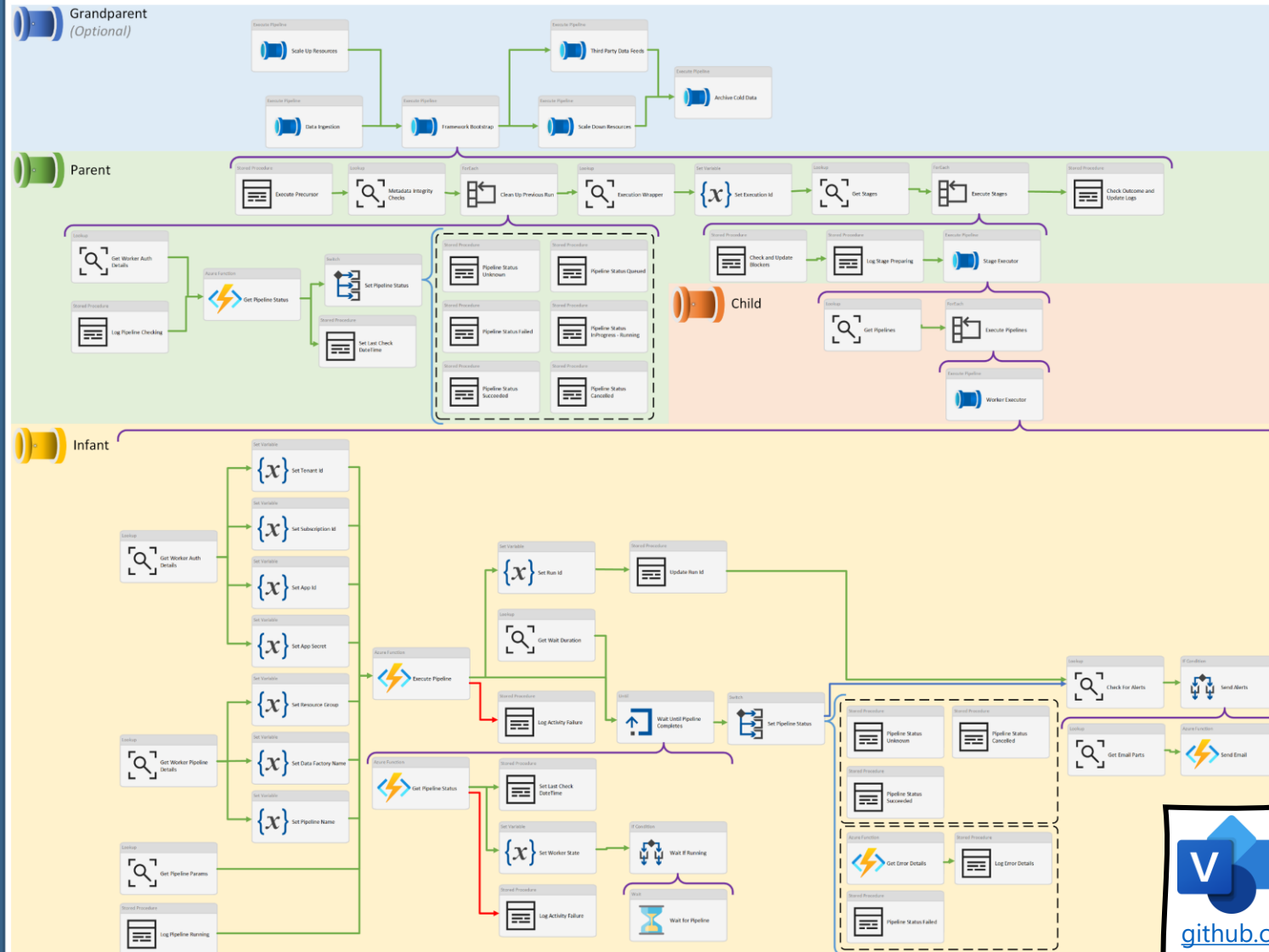
Role: Anything specific to the process needing to be performed.

Processing Framework - Activity Chain

Orchestration Framework Support Resources



Orchestration Framework



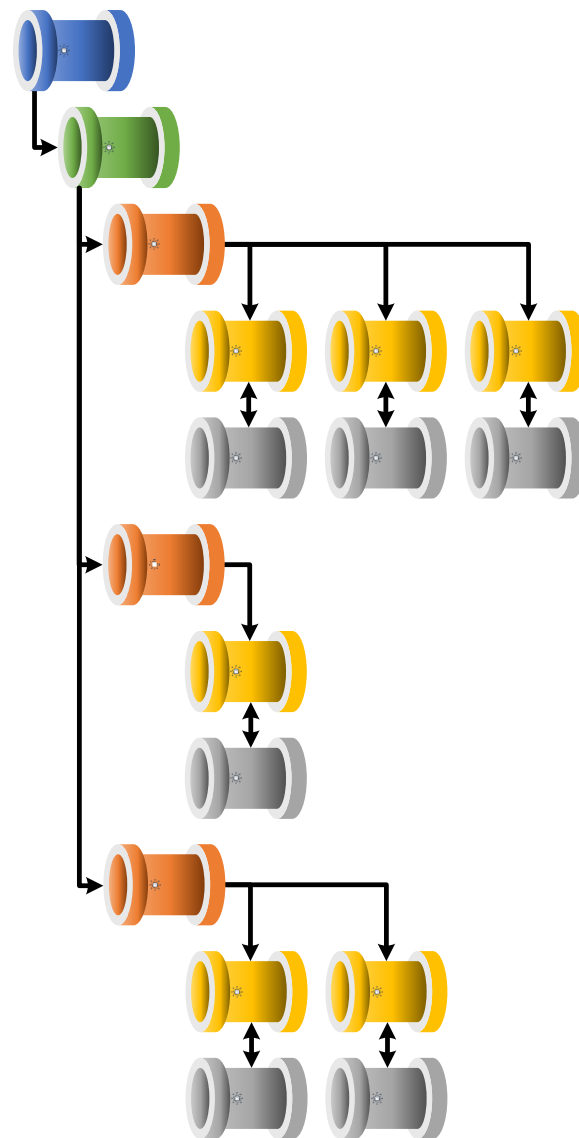
Worker Pipelines

- Worker 1 - Extract
- Worker 2 - Clean
- Worker 3 - Transform
- Worker 4 - Load
- Worker 5 - Serve
- Worker n -



Go to Visio file in
GitHub:

github.com/mrpaulandrew/procfwk/blob/master/Images



Module 5:

Metadata Driven Pipelines

☐☐ Expressions



☐☐ Dynamic Pipeline



☐☐ Orchestration Framework

