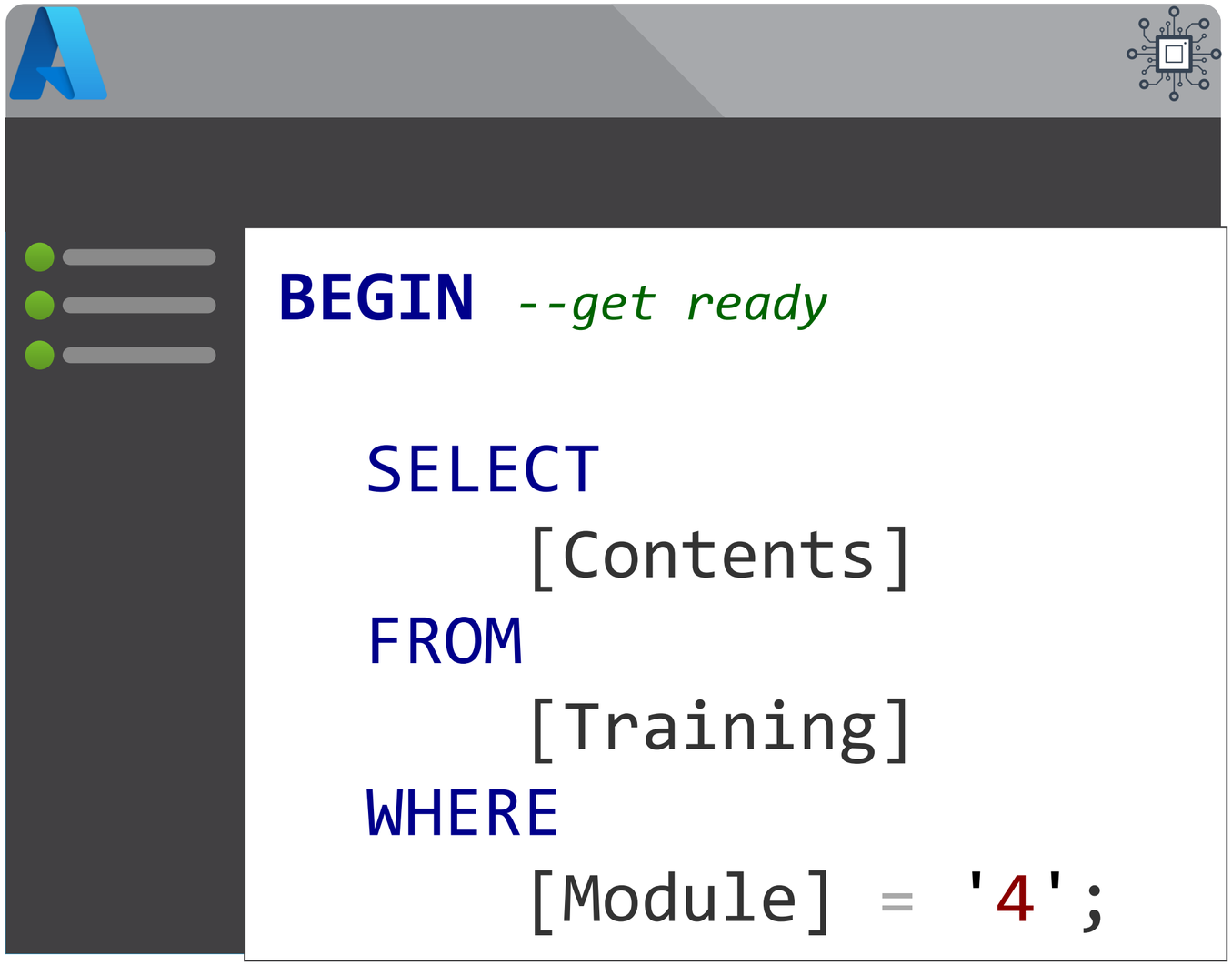
Module 4

Dynamic Pipelines



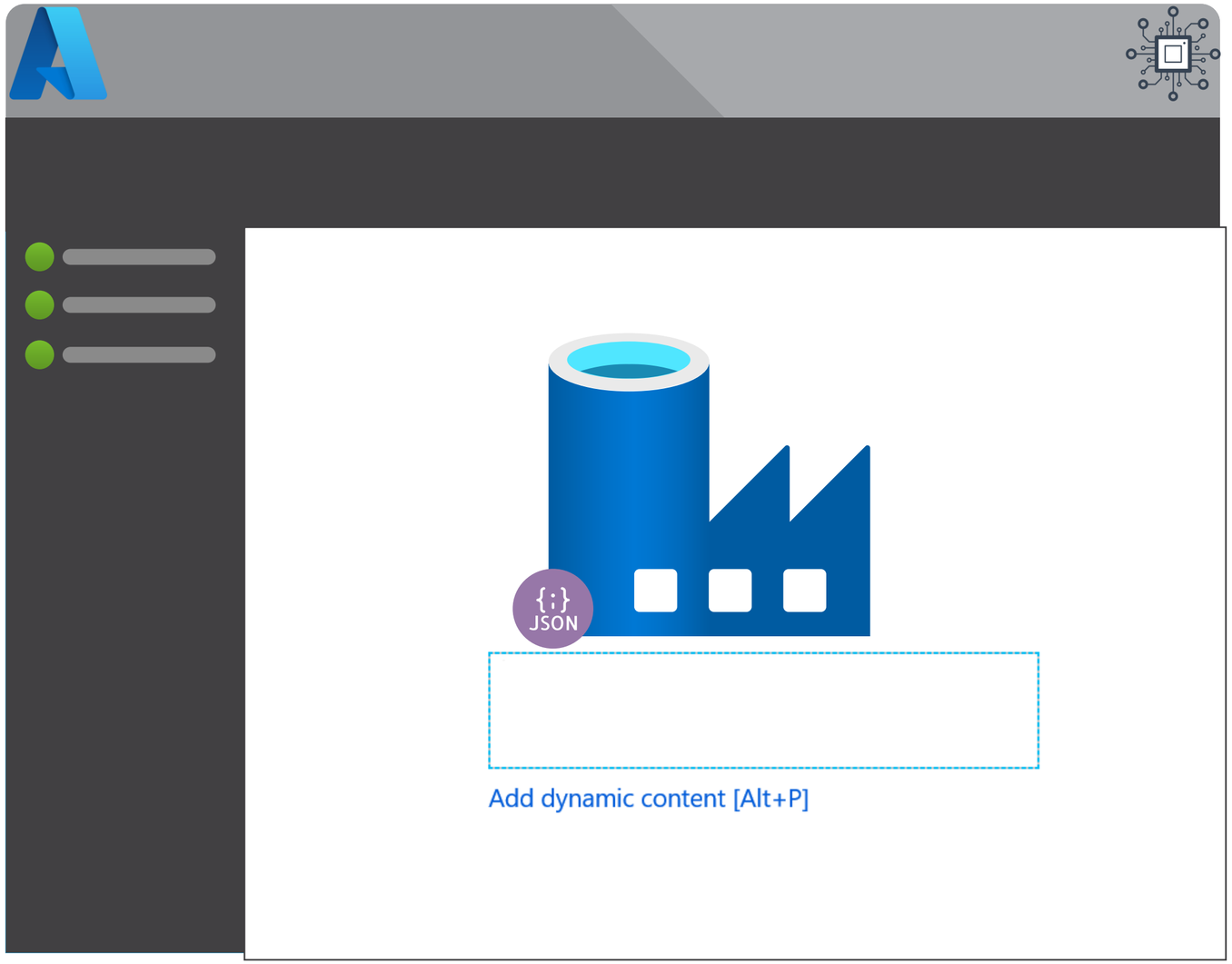
```
BEGIN --get ready

    SELECT
        [Contents]
    FROM
        [Training]
    WHERE
        [Module] = '4';
```

- Expressions & Interpolation
- Simple Metadata Driven Execution
- Dynamic Content Chains
- Reference Names

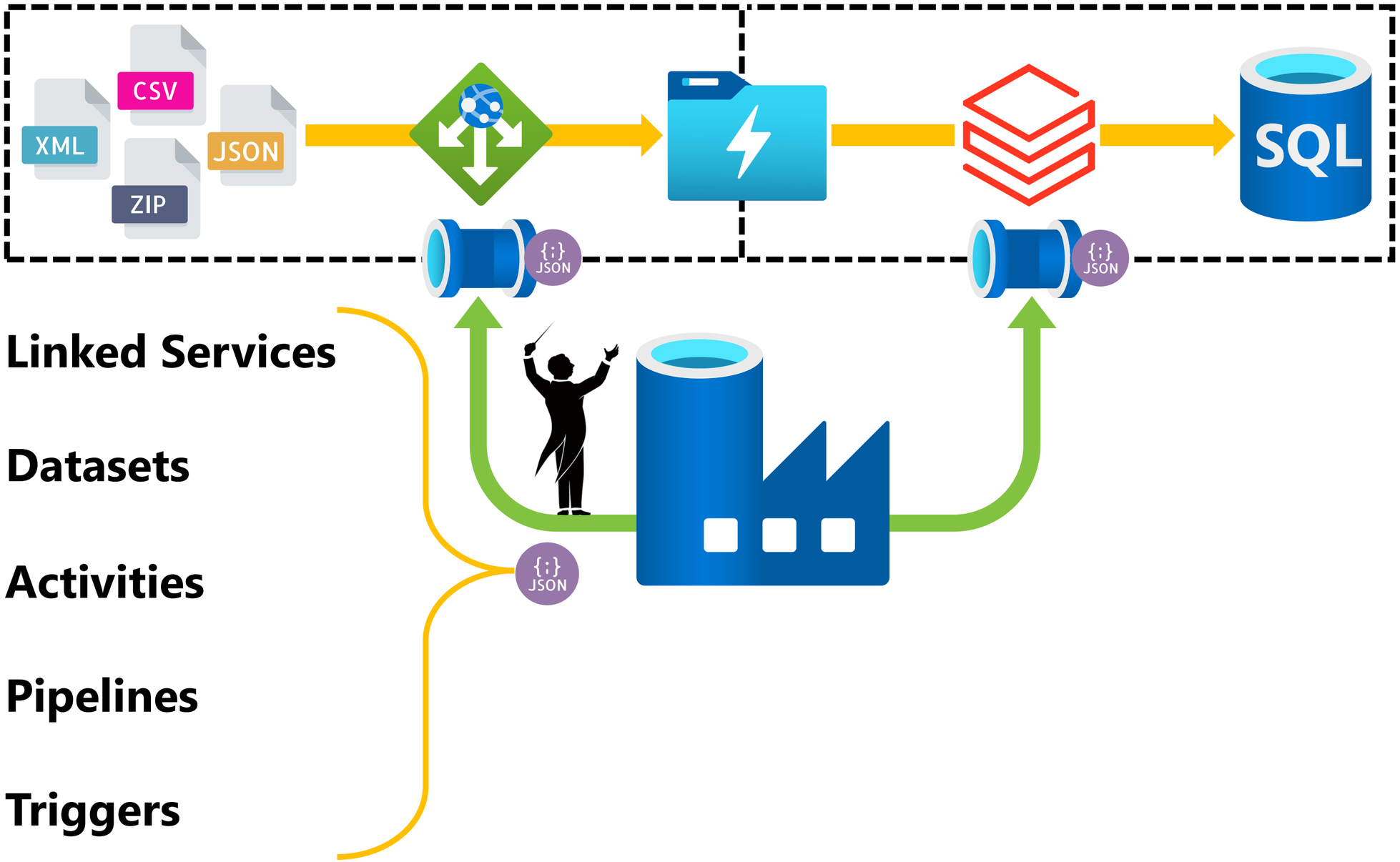
Module 4

Dynamic Pipelines



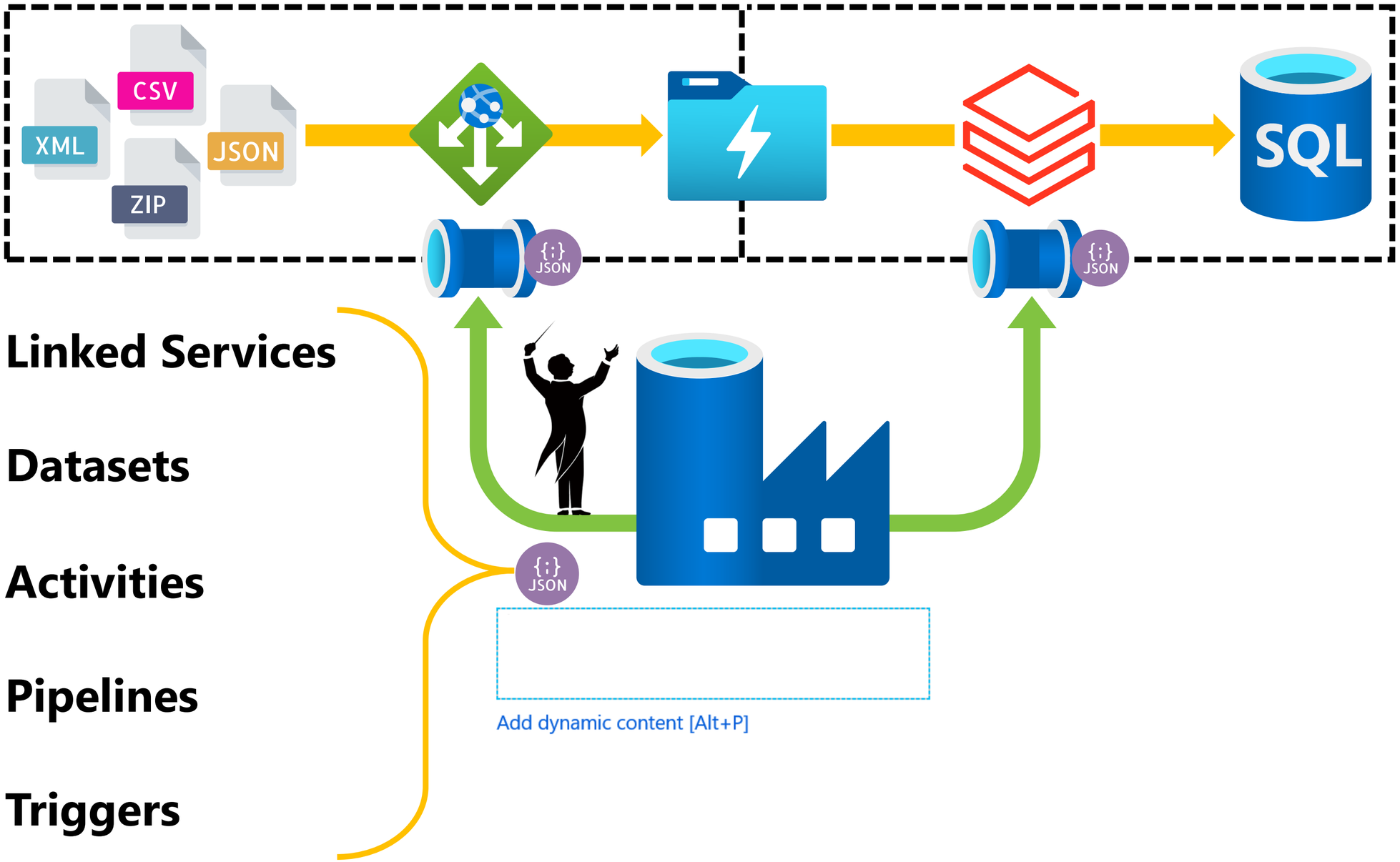
- Expressions & Interpolation
- Simple Metadata Driven Execution
- Dynamic Content Chains
- Reference Names

Data Factory Components



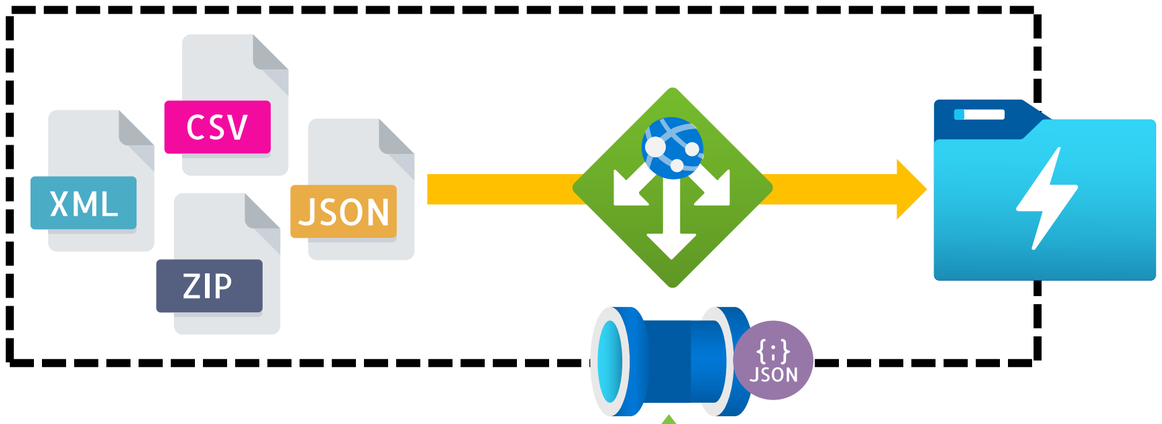
1. **Linked Services**
2. **Datasets**
3. **Activities**
4. **Pipelines**
5. **Triggers**

Data Factory Components – Add Dynamic Content

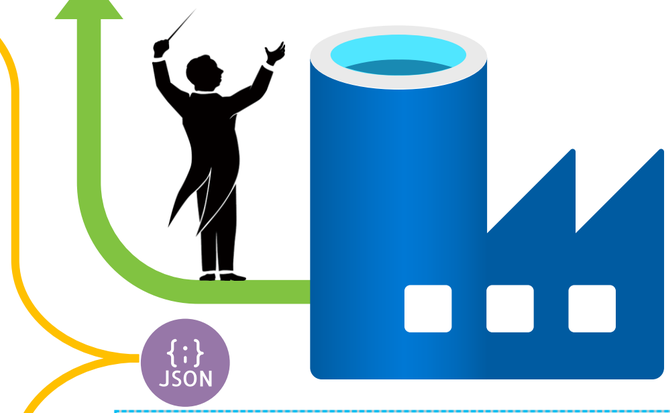


1. **Linked Services**
2. **Datasets**
3. **Activities**
4. **Pipelines**
5. **Triggers**

Control Flow Expressions



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



@pipeline().parameters.DemoParam

Add dynamic content [Alt+P]

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

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-expression-language-functions

Control Flow Expressions

Save | Save as template | Validate | Debug

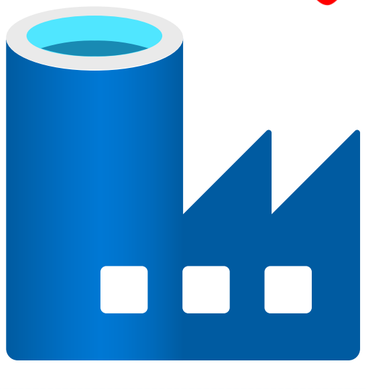
```
"SomethingDynamic": {
    "value": "@pipeline().parameters.DemoParam",
    "type": "Expression"
}
```

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

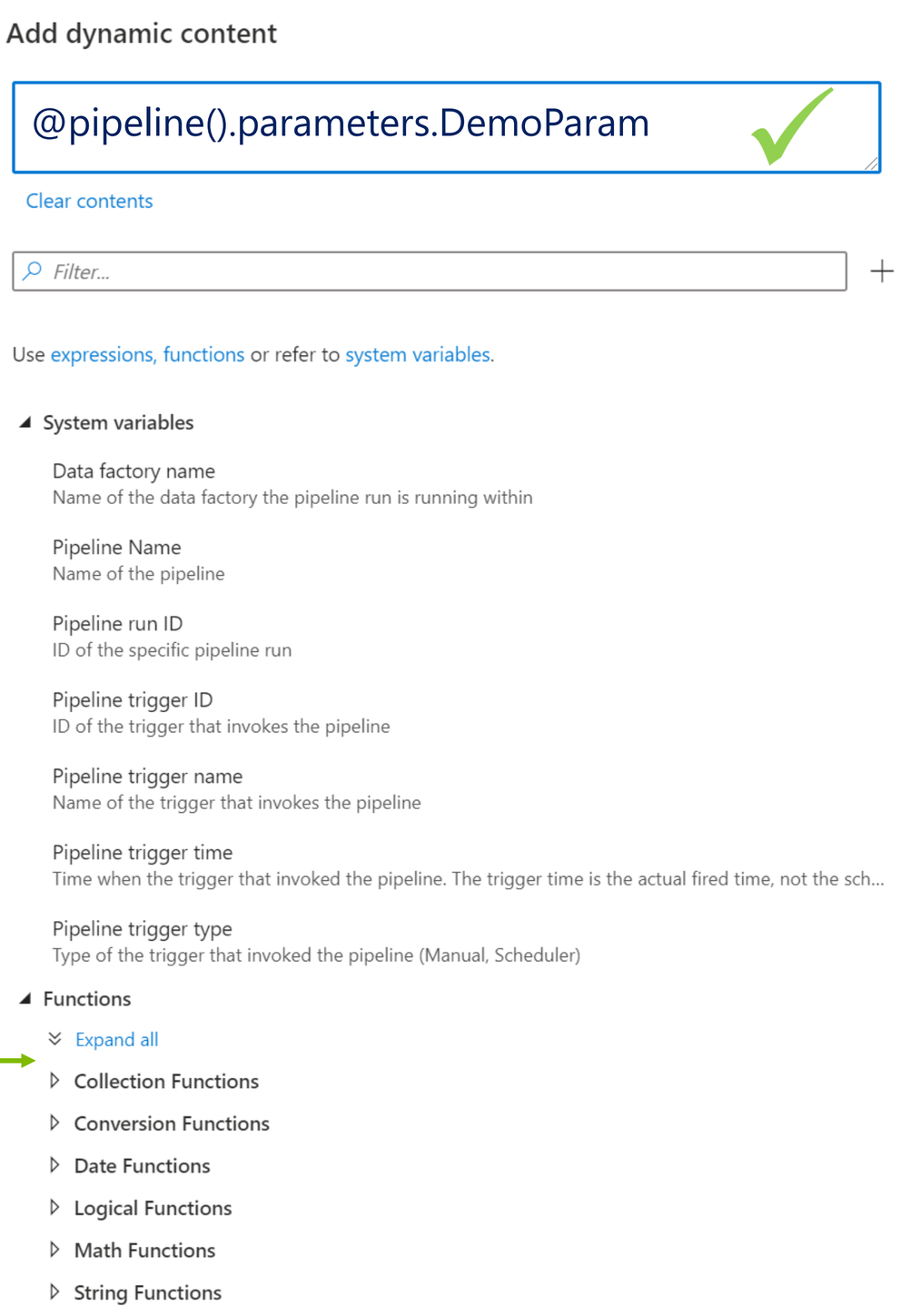
String Interpolation

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

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

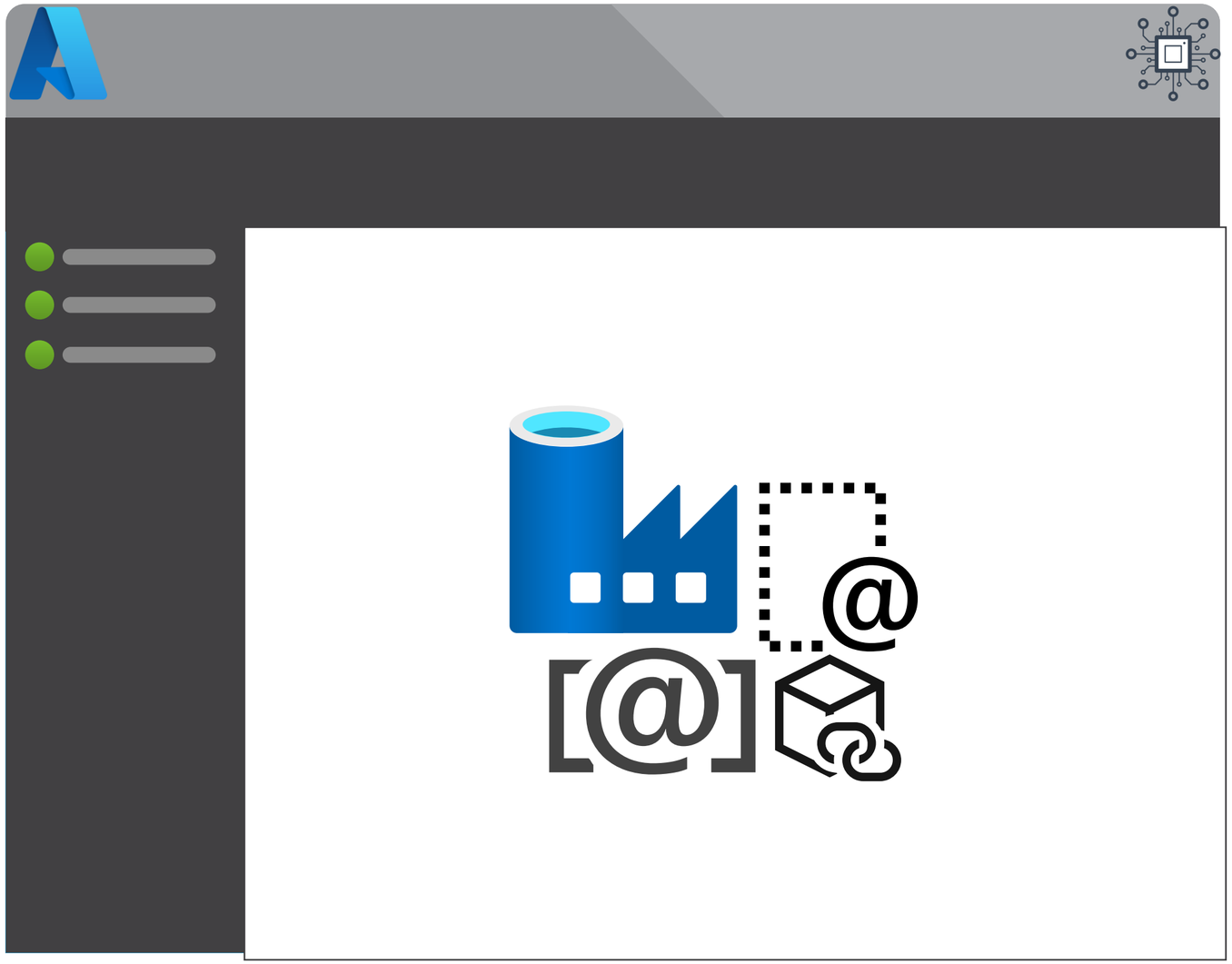



```
"SomethingDynamic": "Hello @{pipeline().parameters.DemoParam}"
```



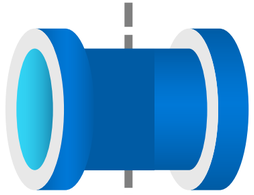
Module 4

Dynamic Pipelines

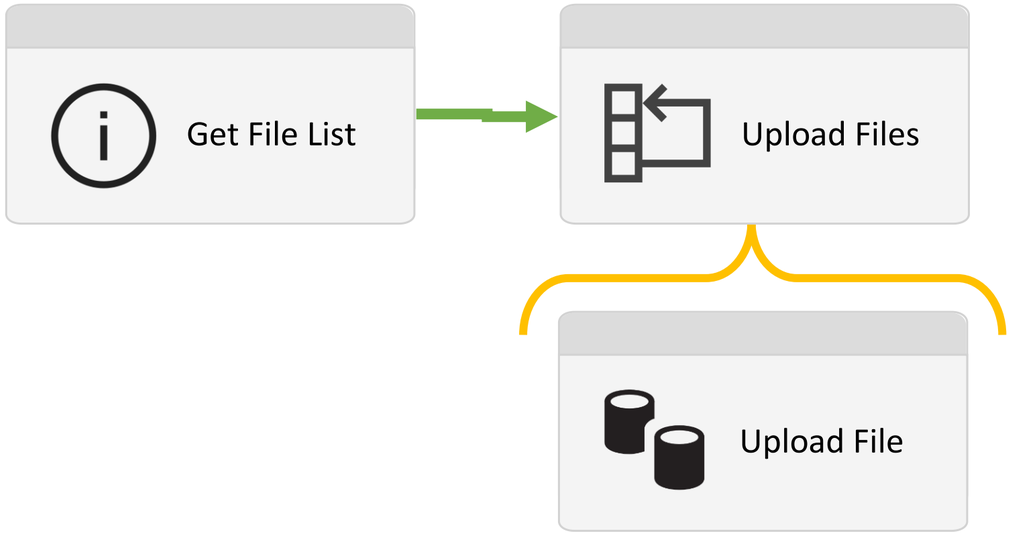


- Expressions & Interpolation
- Simple Metadata Driven Execution
  - Outputs
  - Global Parameters
  - Variables
- Dynamic Content Chains
- Reference Names

Data Discovery and Upload

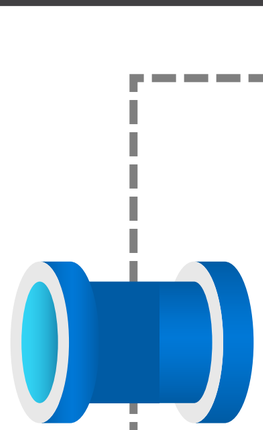


@activity('Get File List')
.output.[childItems]



@item().name

Simple Metadata and Upload



@activity('Get File List')
.output.[value]

Get File List

Log Upload Start

@activity('Log Upload Start')
.output.firstRow.LogId

Upload Files

Upload File

@item().SourceDirectory

@item().TargetDirectory

@item().FileName

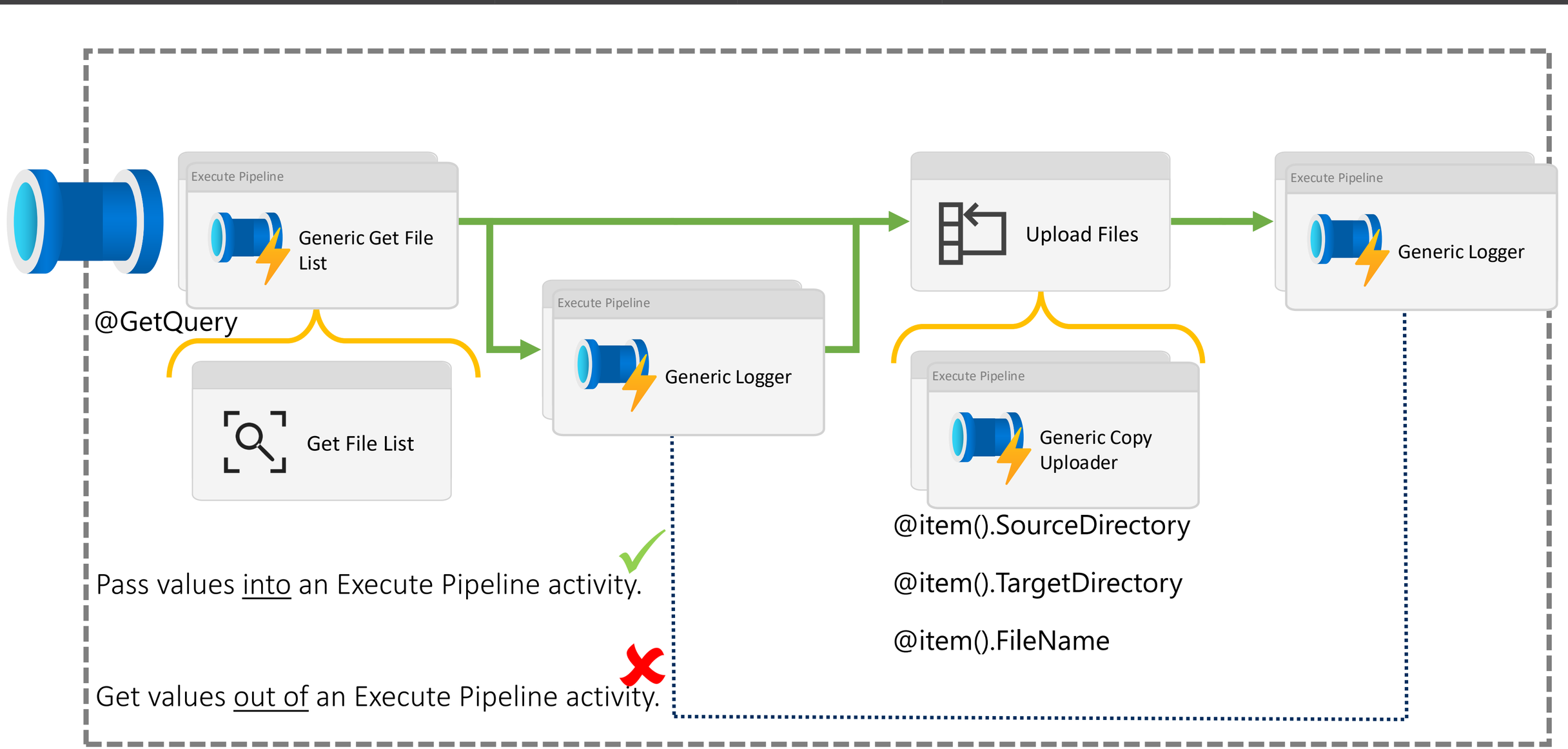
Log Upload Finish

Additional columns

New | Delete

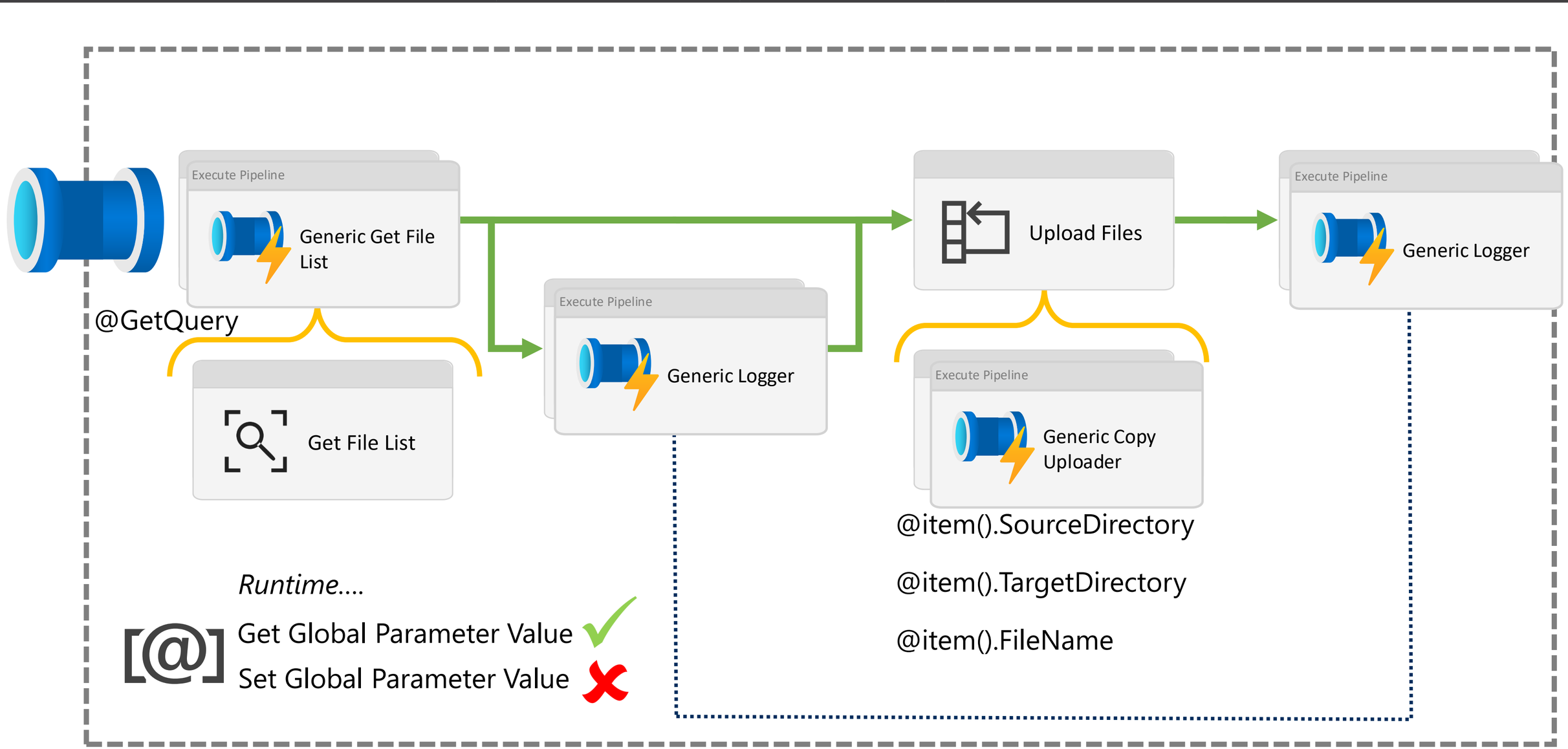
NAME	VALUE
FilePath	$$FILEPATH
DataFactoryName	@pipeline().DataFactory
PipelineName	@pipeline().Pipeline
UploadDateTime	@utcnow()
RunId	@pipeline().RunId

Using Pipelines as Worker Components



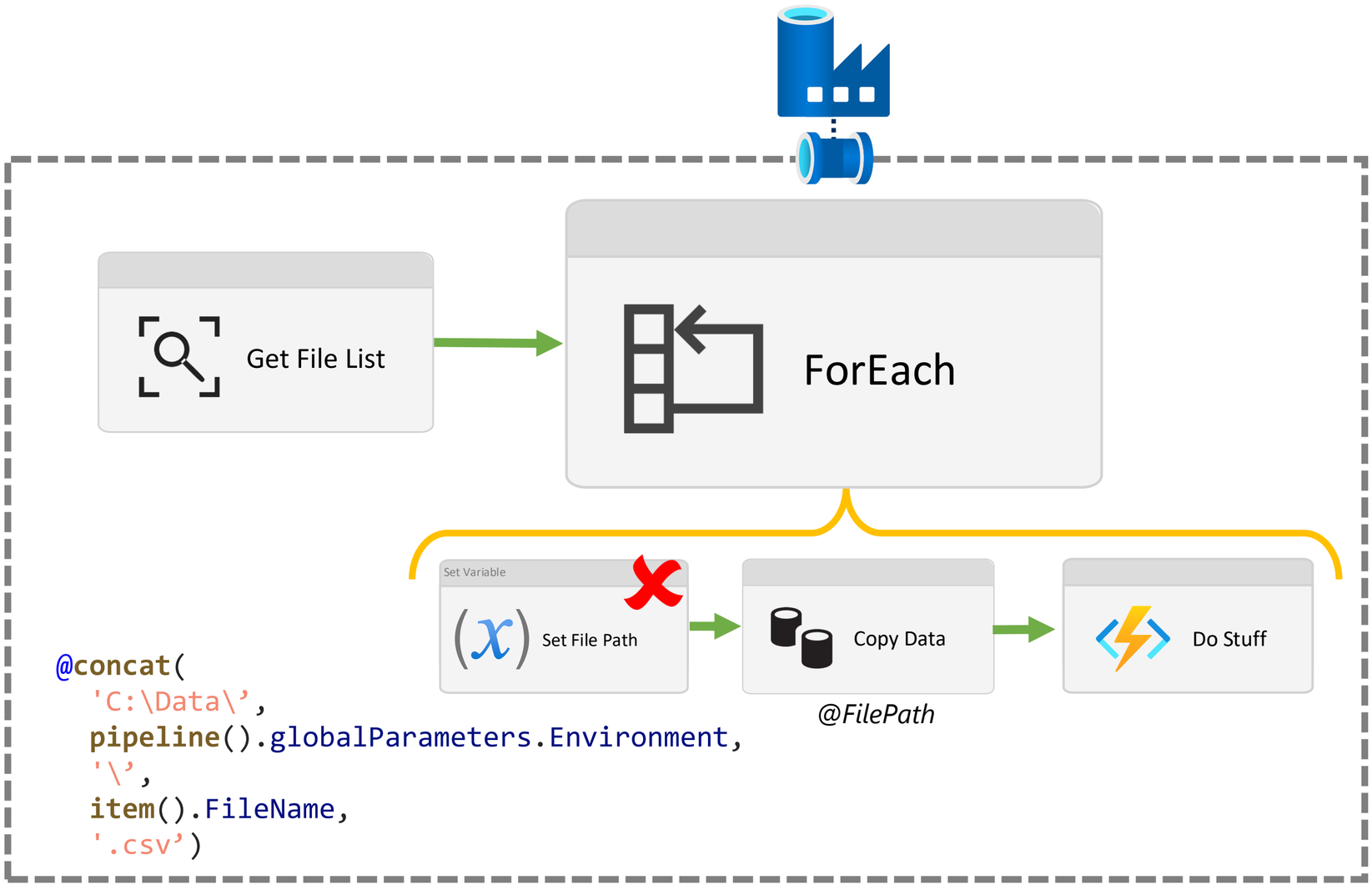
Using Pipelines as Worker Components

With a Global Parameters

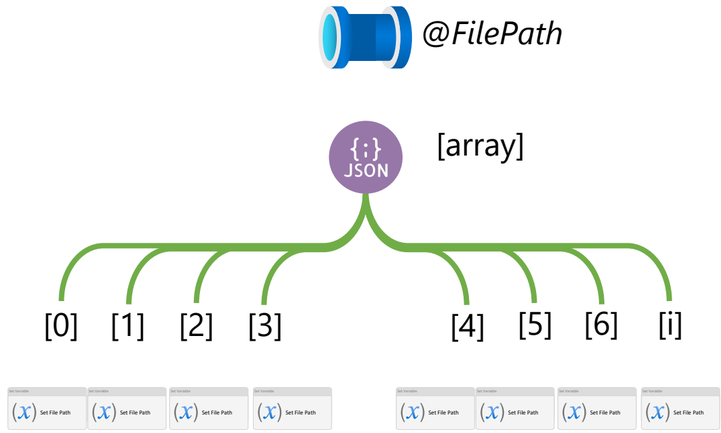


Variable Scope

Thread Safe



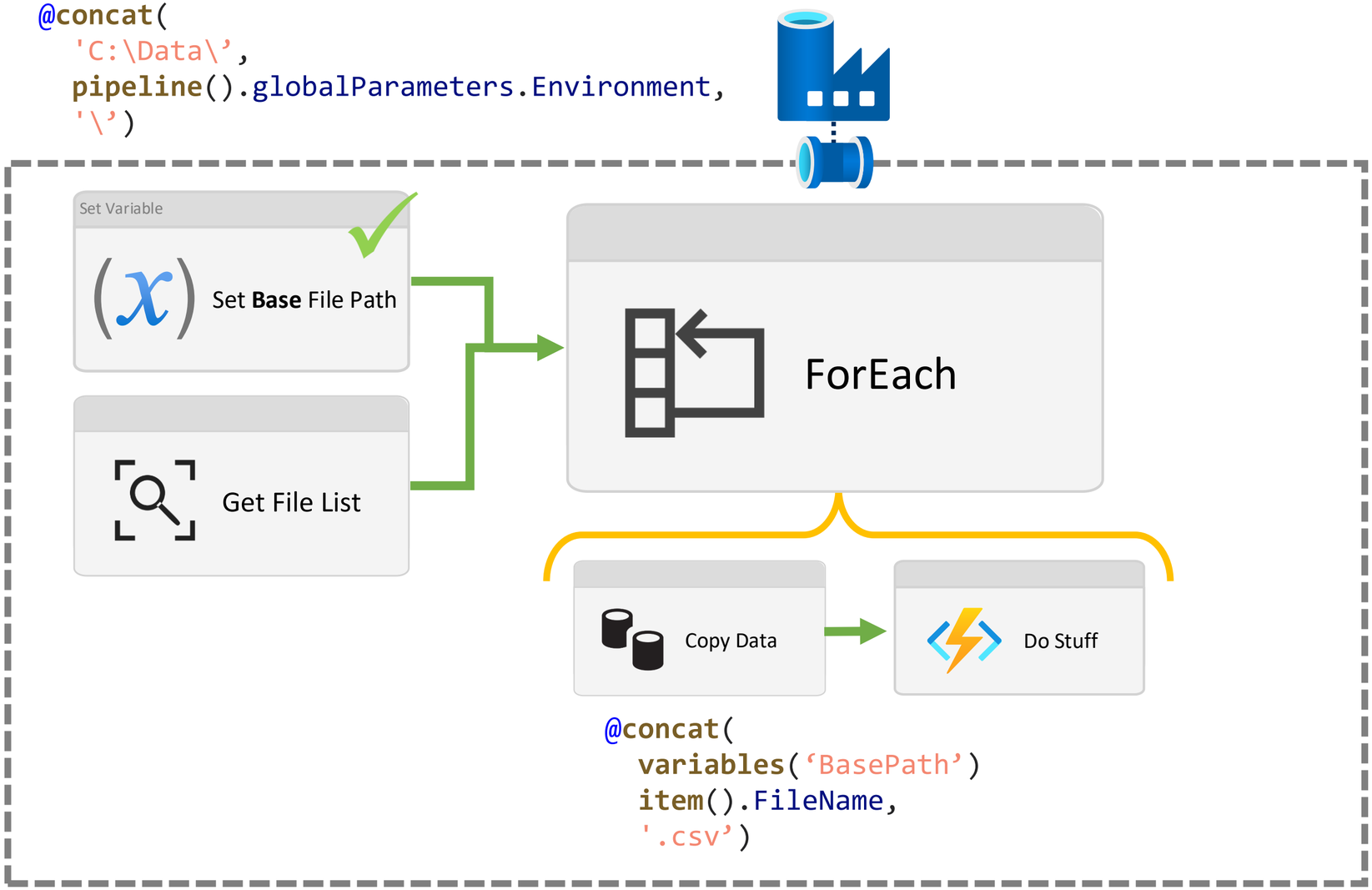

```
@concat(
   'C:\Data\',
   pipeline().globalParameters.Environment,
   '\',
   item().FileName,
   '.csv')
```



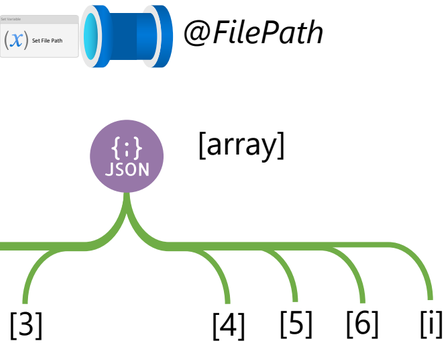
Variable Scope

Layering Expressions

```
@concat(
  'C:\Data\',
  pipeline().globalParameters.Environment,
  '\')
```




```
@concat(
  variables('BasePath')
  item().FileName,
  '.csv')
```



Module 4

Dynamic Pipelines

- Expressions & Interpolation
- Simple Metadata Driven Execution
- Dynamic Content Chains
- Reference Names

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

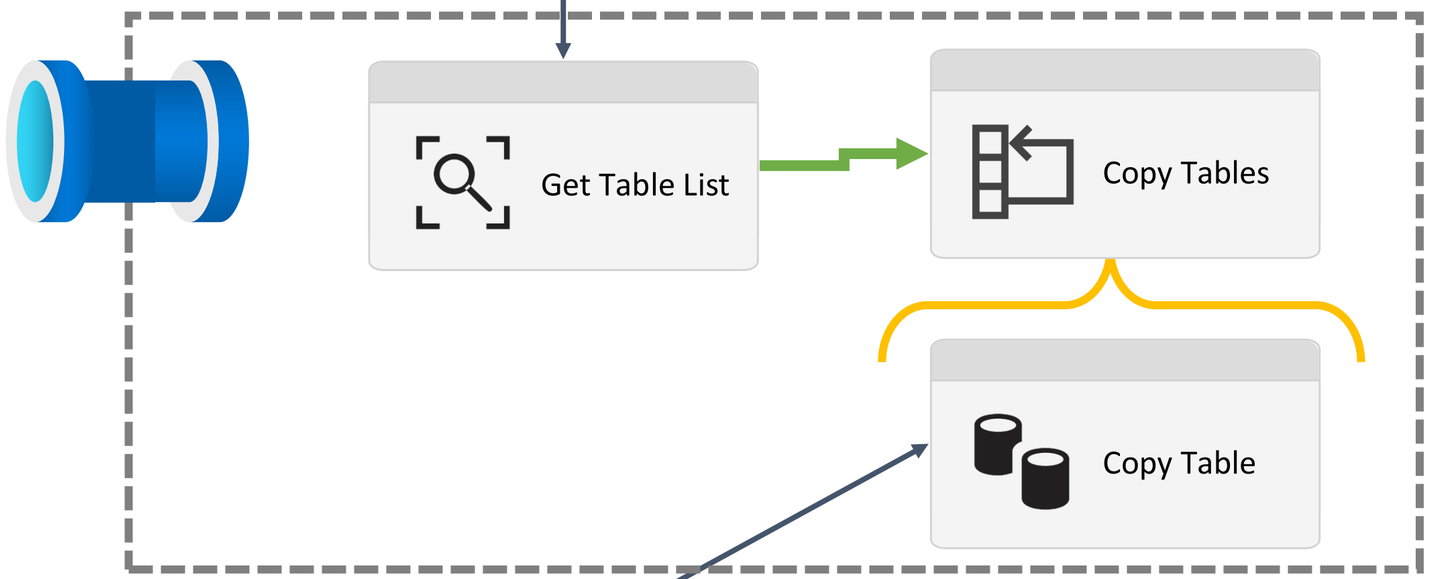


Table option ○ None ◉ Auto create table ⓘ

1x

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

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

1x

@pipeline().parameters.SourceConnectionSecret

@pipeline().parameters.TargetConnectionSecret

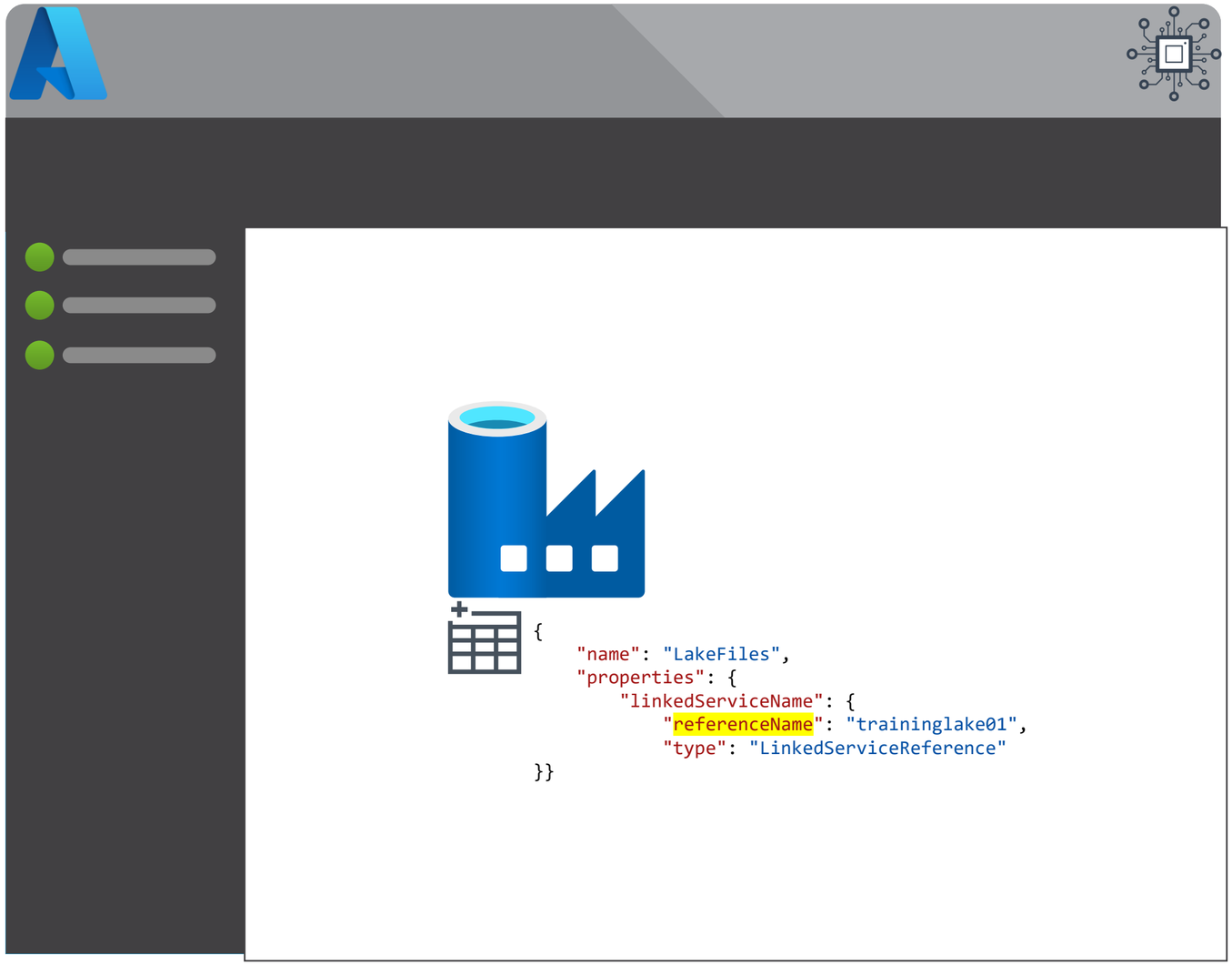
@dataset().LinkedServiceConnectionSecret

@linkedService().DBConnectionSecret

Demo

Module 4

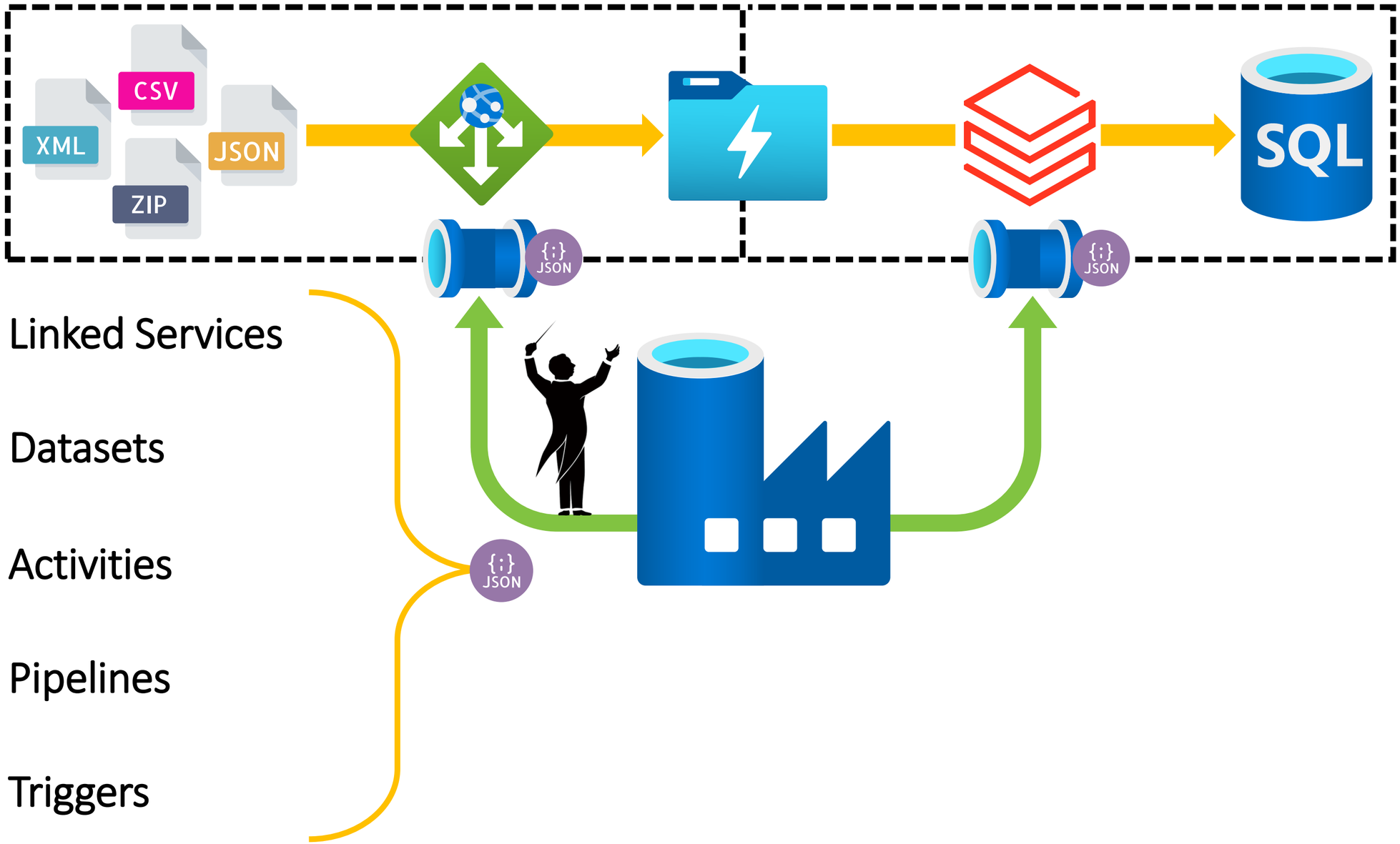
Dynamic Pipelines



```
{
    "name": "LakeFiles",
    "properties": {
        "linkedServiceName": {
            "referenceName": "traininglake01",
            "type": "LinkedServiceReference"
}}
```

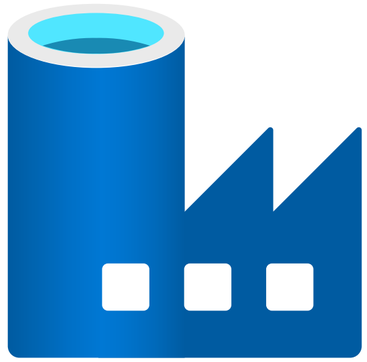
- Expressions & Interpolation
- Simple Metadata Driven Execution
- Dynamic Content Chains
- Reference Names

Reference Names



1. Linked Services
2. Datasets
3. Activities
4. Pipelines
5. Triggers

Reference Names



1. Linked Services
2. Datasets
3. Activities
4. Pipelines
5. Triggers

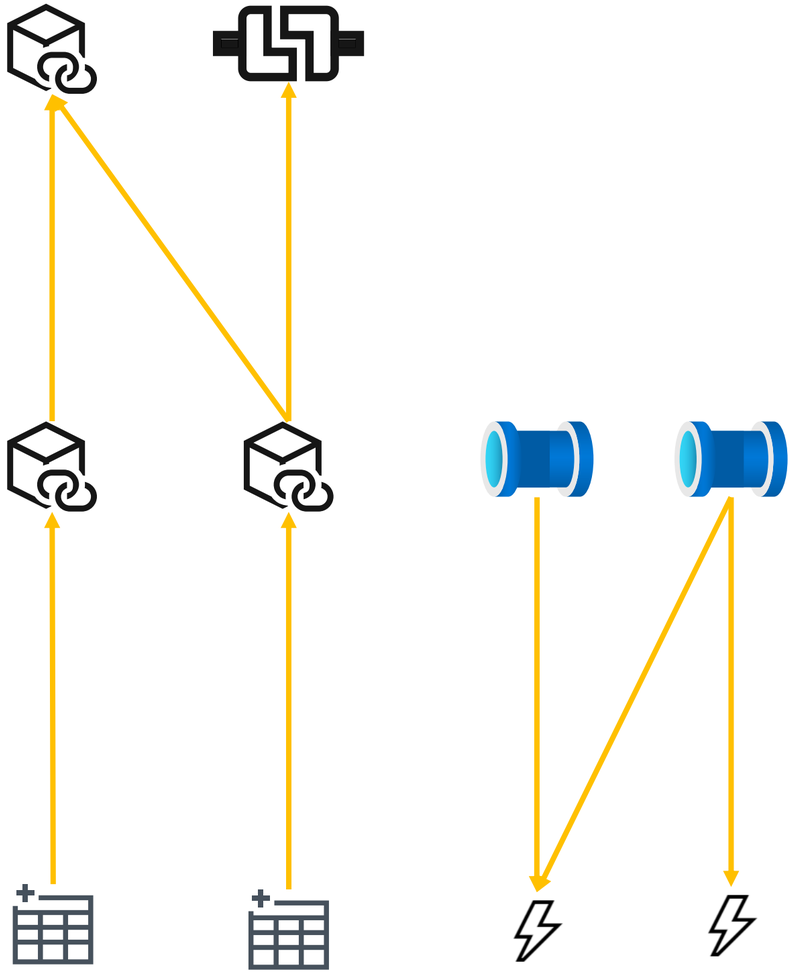
```
{
    "name": "TrainingKeys01",
    "type": "Microsoft.DataFactory/factories/linkedservices",
    "properties": {
        "annotations": [],
        "type": "AzureKeyVault",
        "typeProperties": {
            "baseUrl": "https://TrainingKeys01.vault.azure.net/"
}}}
```

```
{
    "name": "trainglake01",
    "properties": {
        "typeProperties": {
            "accountKey": {
                "type": "AzureKeyVaultSecret",
                "store": {
                    "referenceName": "TrainingKeys01",
                    "type": "LinkedServiceReference"
}}}}}
```

```
{
    "name": "LakeFiles",
    "properties": {
        "linkedServiceName": {
            "referenceName": "trainglake01",
            "type": "LinkedServiceReference"
}}
```

Reference Names

1. Linked Services
2. Datasets
3. Activities
4. Pipelines
5. Triggers



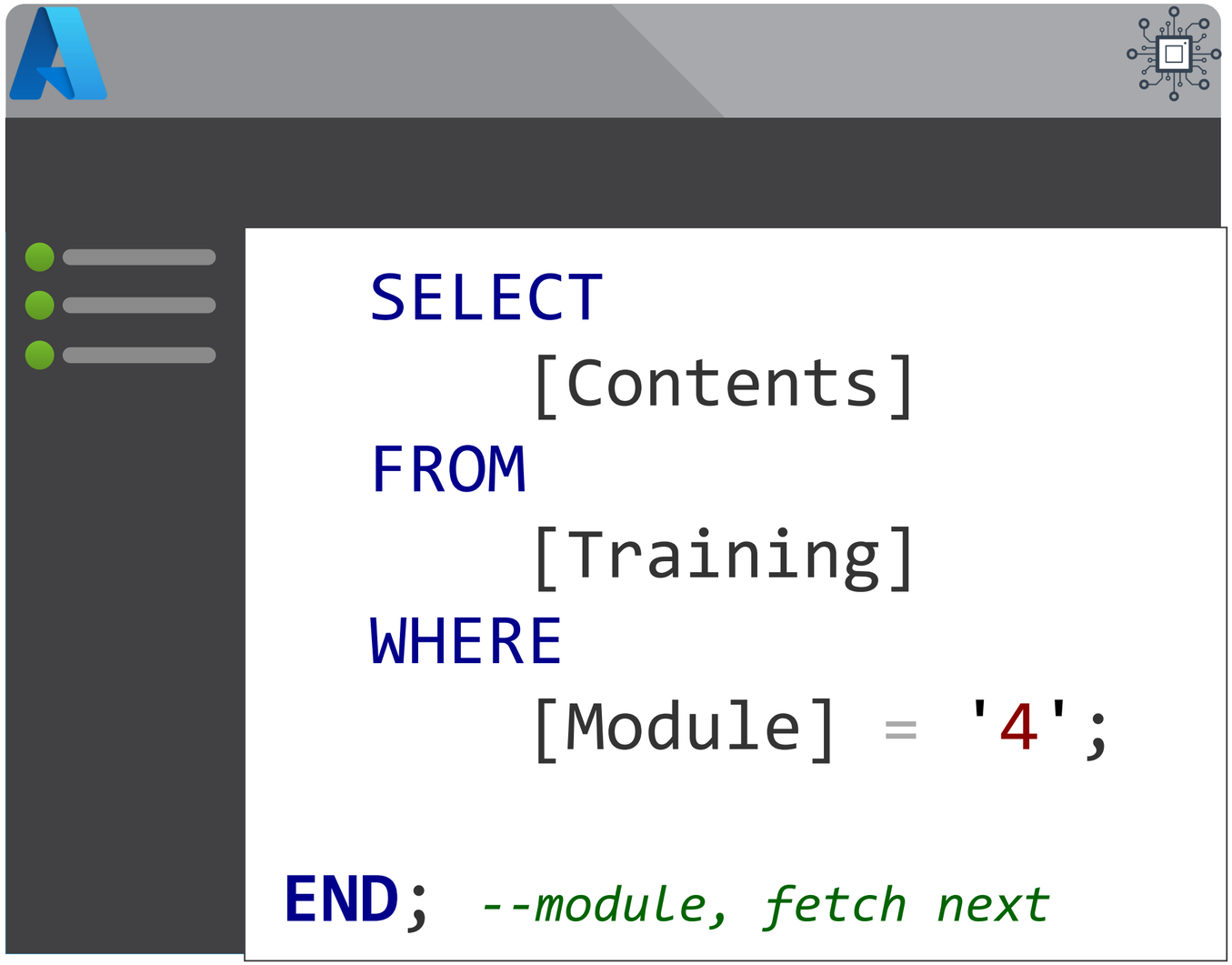
Reference Names **cannot** be dynamic.

- Not at development time.
- Not at runtime.
- At deployment time if <u>being careful</u>.

They are used internally by Data Factory to validate artifact dependencies.

Module 4

Dynamic Pipelines



```
    SELECT
        [Contents]
    FROM
        [Training]
    WHERE
        [Module] = '4';


END; --module, fetch next
```

- Expressions & Interpolation
- Simple Metadata Driven Execution
- Dynamic Content Chains
- Reference Names