Microsoft

Building Dynamic Data Pipelines in Azure Data Factory

Cathrine Wilhelmsen, *Inmeta*
@cathrinew | cathrinew.net

BRK2103

Session Abstract

You already know how to build, orchestrate, and monitor data pipelines in Azure Data Factory (ADF). But how do you go from basic, hardcoded data pipelines to making your solution dynamic and reusable?

In this session, we dive straight into some of the more advanced features of Azure Data Factory. How do you parameterize your linked services, datasets, and pipelines? What is the difference between parameters and variables, and when should you use them? And how do the expression language and built-in functions really work?

We answer these questions by going through an existing solution step-by-step and gradually making it dynamic and reusable. Along the way, we cover best practices and lessons learned.

cathrine
WILHELMSEN

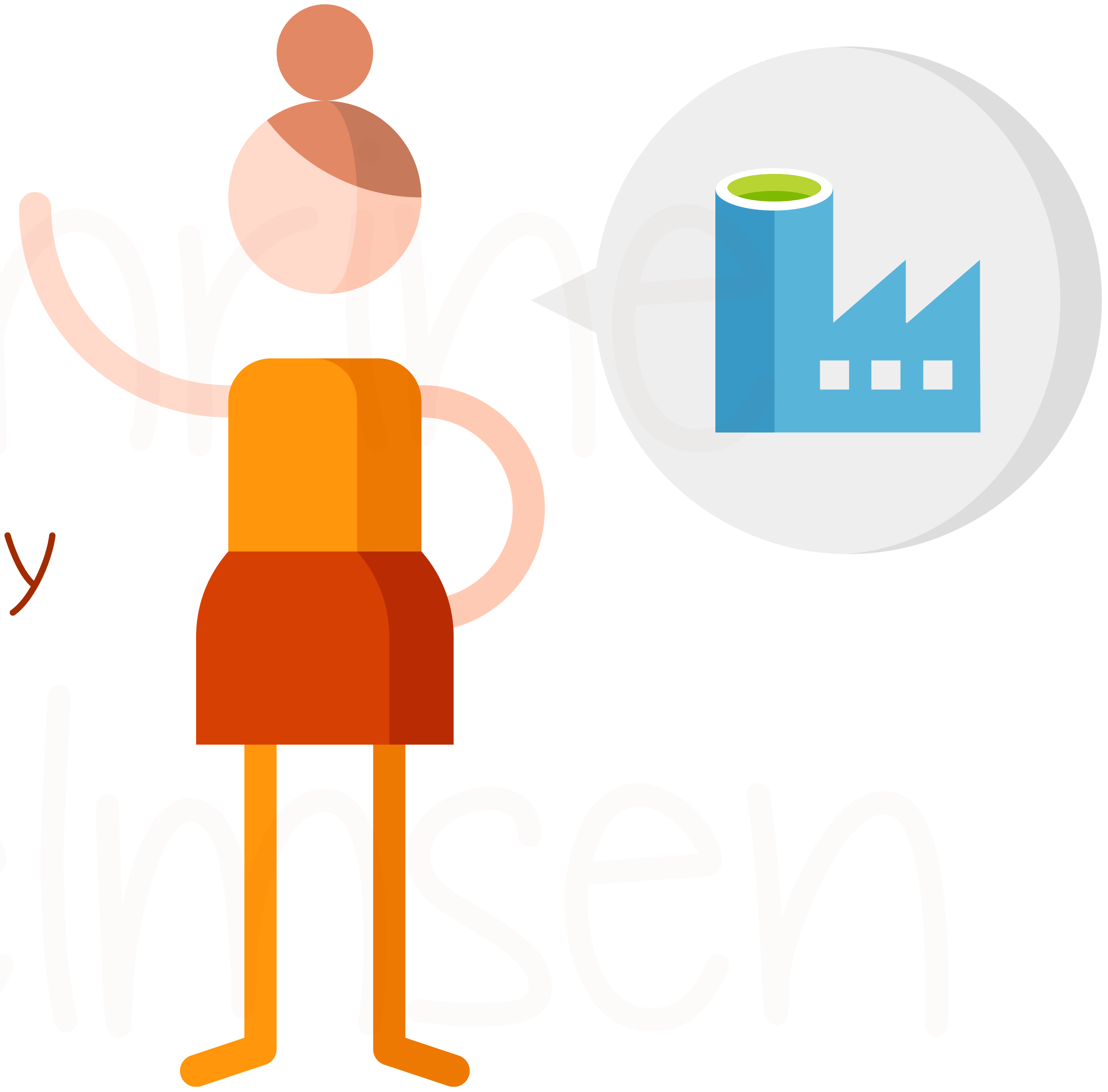
inmeta
EVANGELIST
MICROSOFT
AZURE

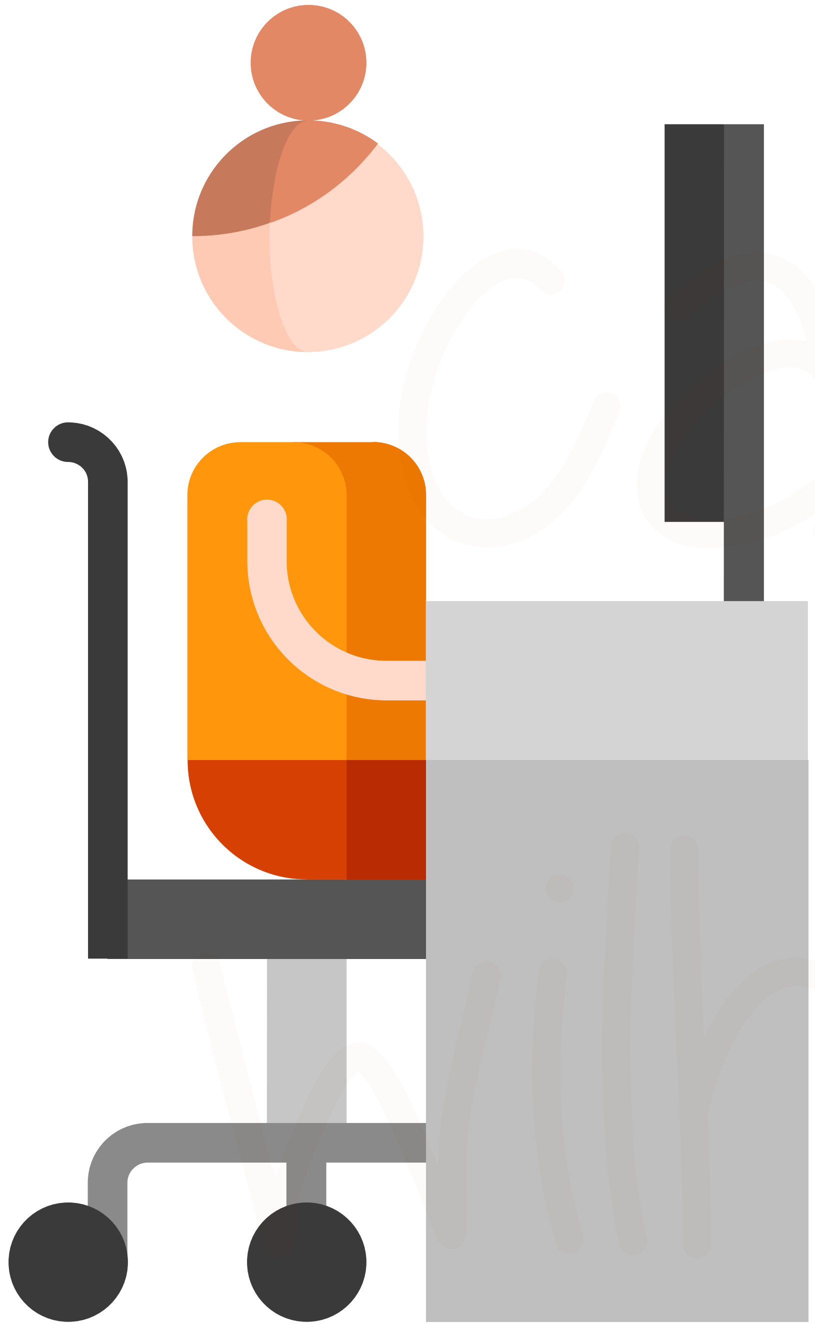
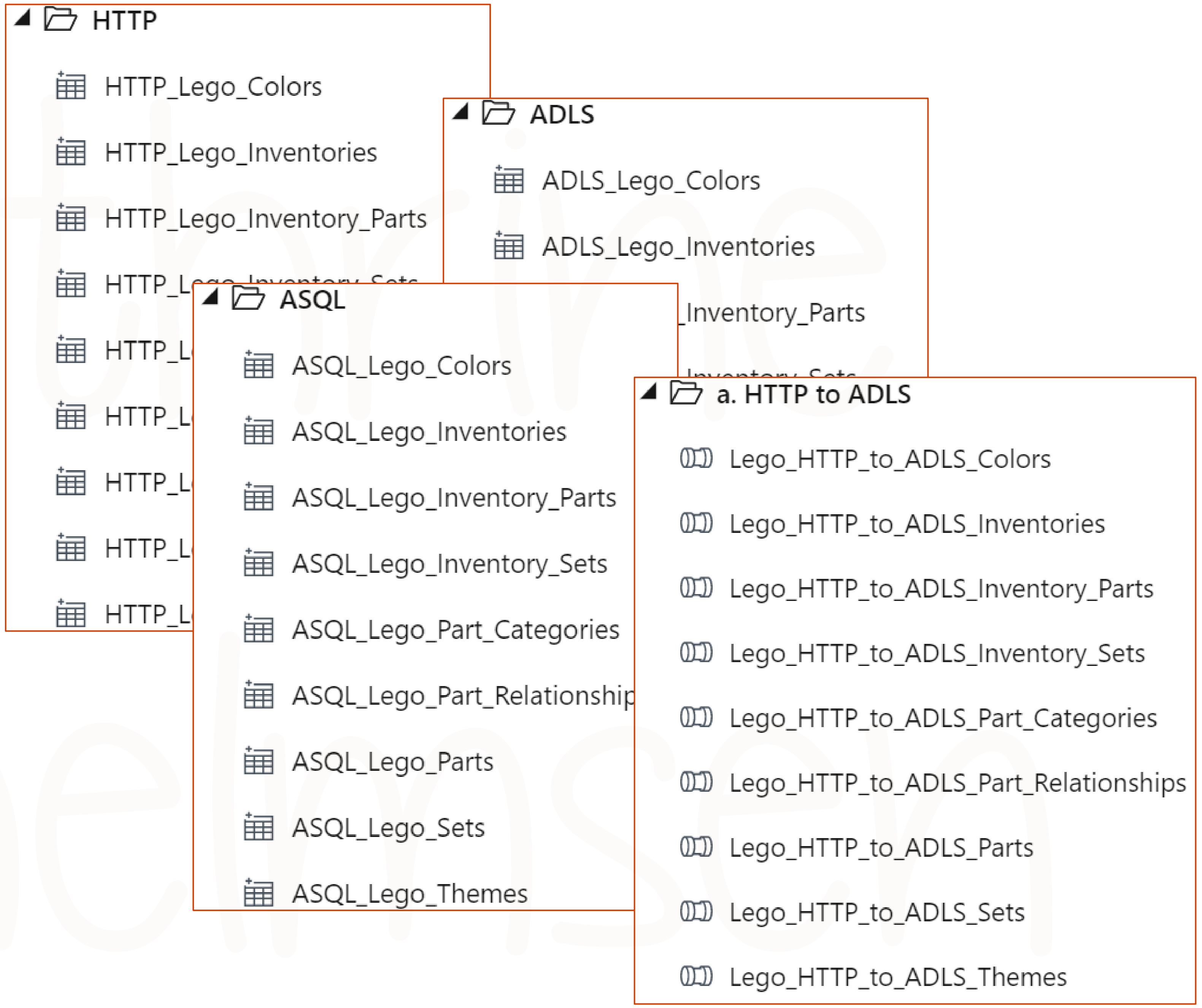
MVP
Microsoft®
Most Valuable
Professional

@cathrinew

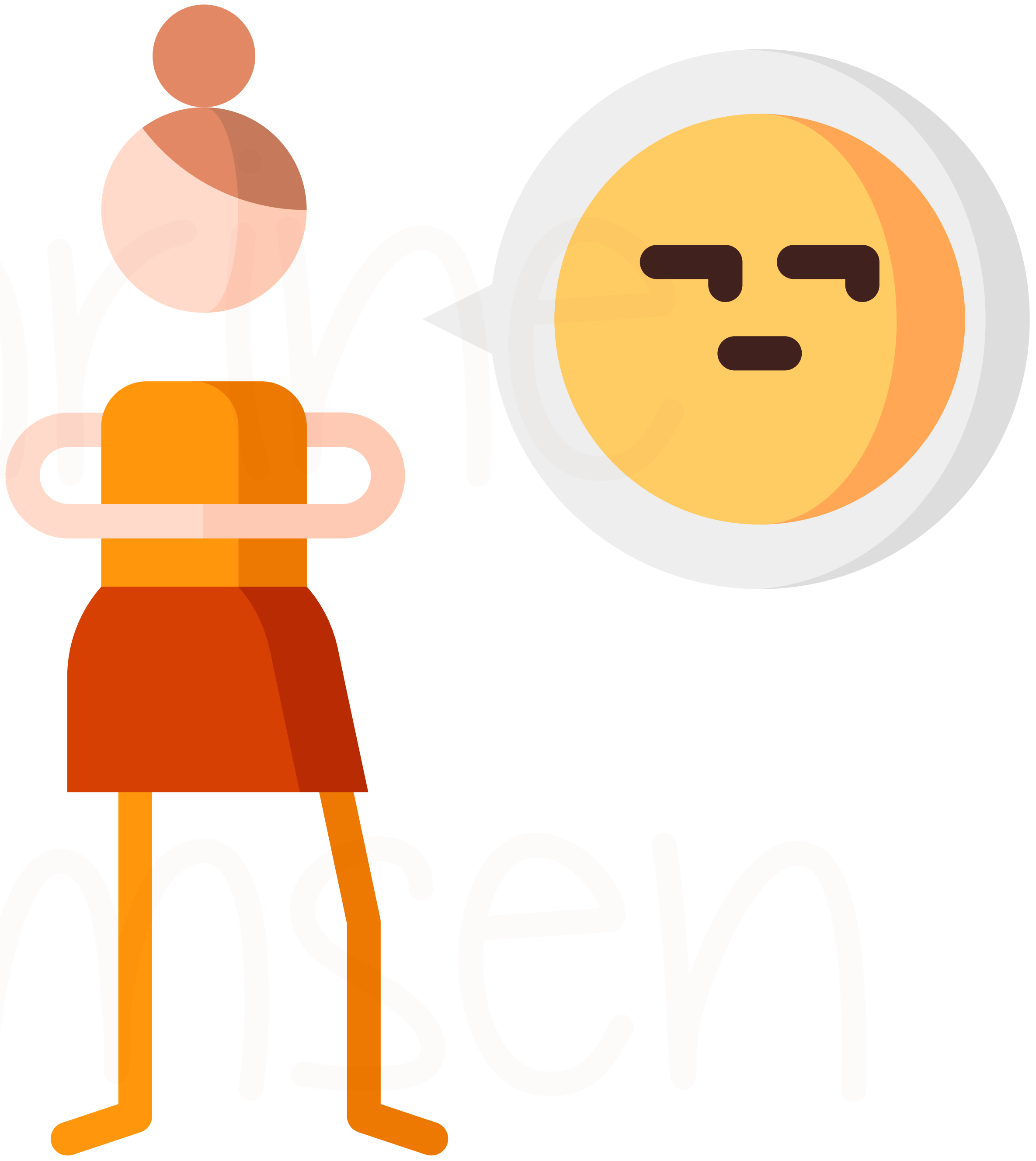
CW cathrinew.net

You are an
Azure Data Factory
developer

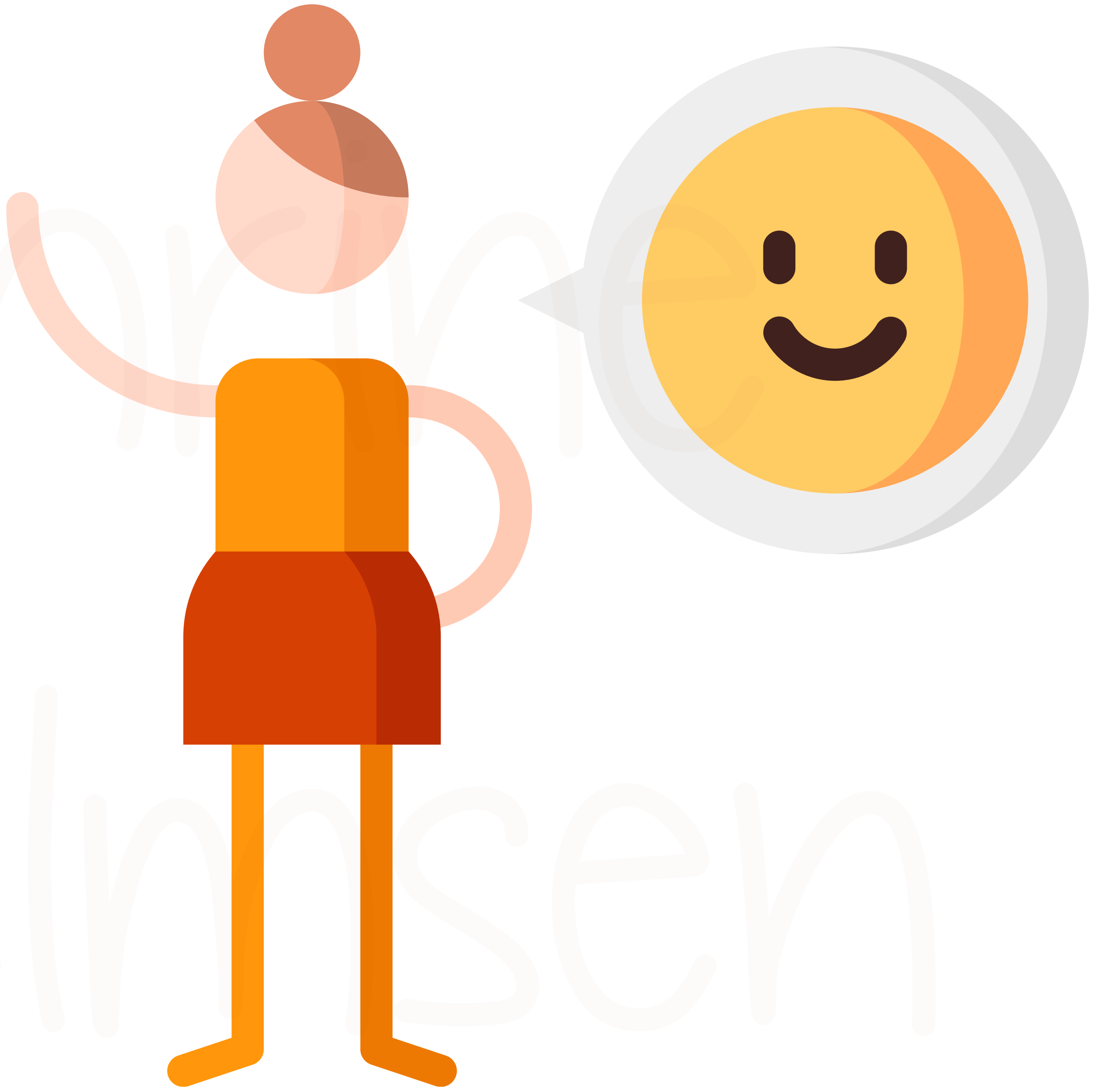




HTTP
HTTP_Lego_Colors
HTTP_Lego_Inventories
HTTP_Lego_Inventory_Parts
HTTP_L
HTTP_L
HTTP_L
HTTP_L
HTTP_L
HTTP_L
ADLS
ADLS_Lego_Colors
ADLS_Lego_Inventories
_Inventory_Parts
ASQL
ASQL_Lego_Colors
ASQL_Lego_Inventories
ASQL_Lego_Inventory_Parts
ASQL_Lego_Inventory_Sets
ASQL_Lego_Part_Categories
ASQL_Lego_Part_Relationship
ASQL_Lego_Parts
ASQL_Lego_Sets
ASQL_Lego_Themes
a. HTTP to ADLS
Lego_HTTP_to_ADLS_Colors
Lego_HTTP_to_ADLS_Inventories
Lego_HTTP_to_ADLS_Inventory_Parts
Lego_HTTP_to_ADLS_Inventory_Sets
Lego_HTTP_to_ADLS_Part_Categories
Lego_HTTP_to_ADLS_Part_Relationships
Lego_HTTP_to_ADLS_Parts
Lego_HTTP_to_ADLS_Sets
Lego_HTTP_to_ADLS_Themes

This is getting pretty tedious...



There has to be a better way!



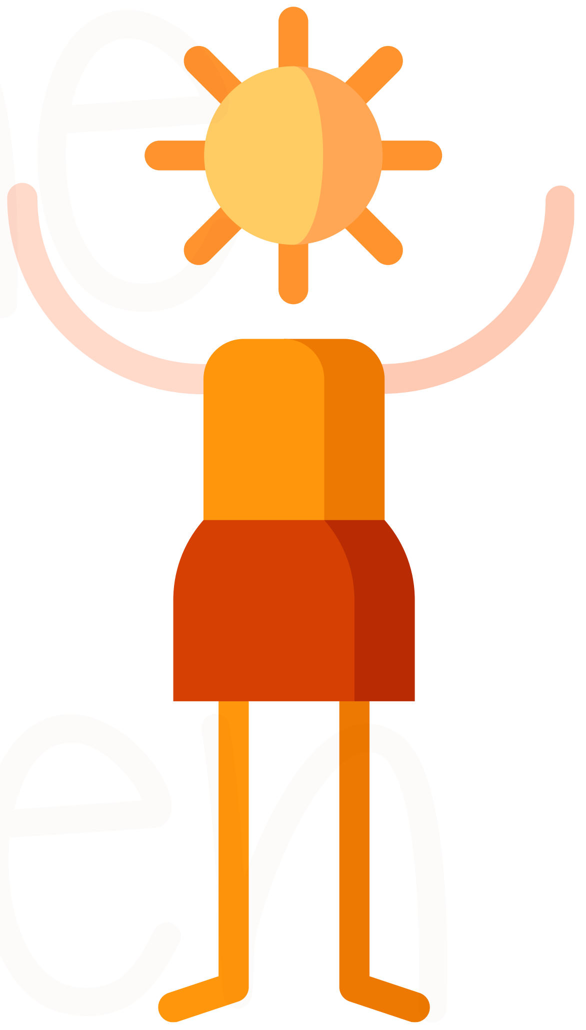
Dynamic Solutions

Why would you use dynamic solutions?

Reduce development time

Reuse patterns for similar tasks

Lower risk of manual errors

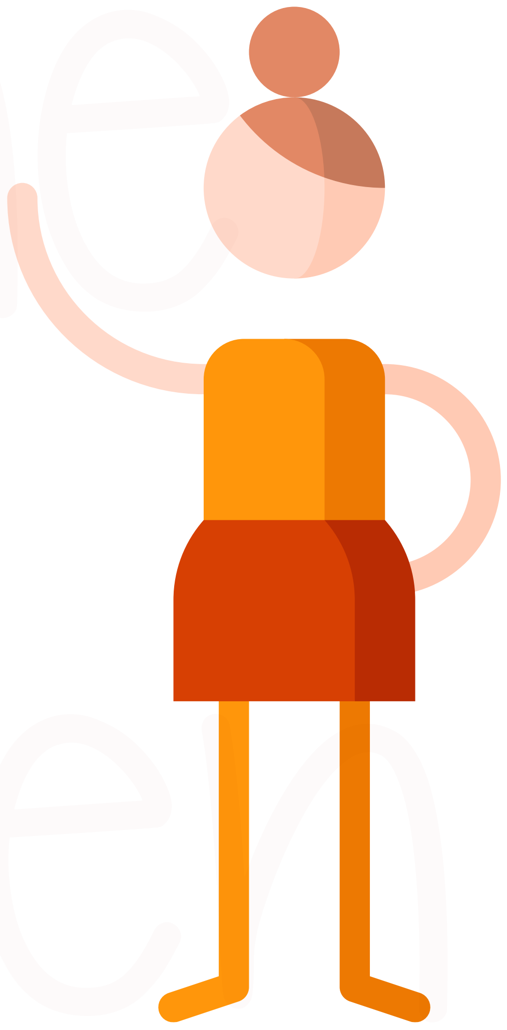


How dynamic should the solution be?

Dynamic enough that you save time on development, but not so dynamic that it becomes difficult to understand

How dynamic should the solution be?

...don't try to make a solution that is generic enough to solve *everything* :)



What can make a solution dynamic?

Parameters and Variables:

Pass input values and set or update values during runtime

Expressions and Functions:

Modify the content of values during runtime

Loops and Lookups:

Control logic and executions based on external configuration values

Parameters and Variables

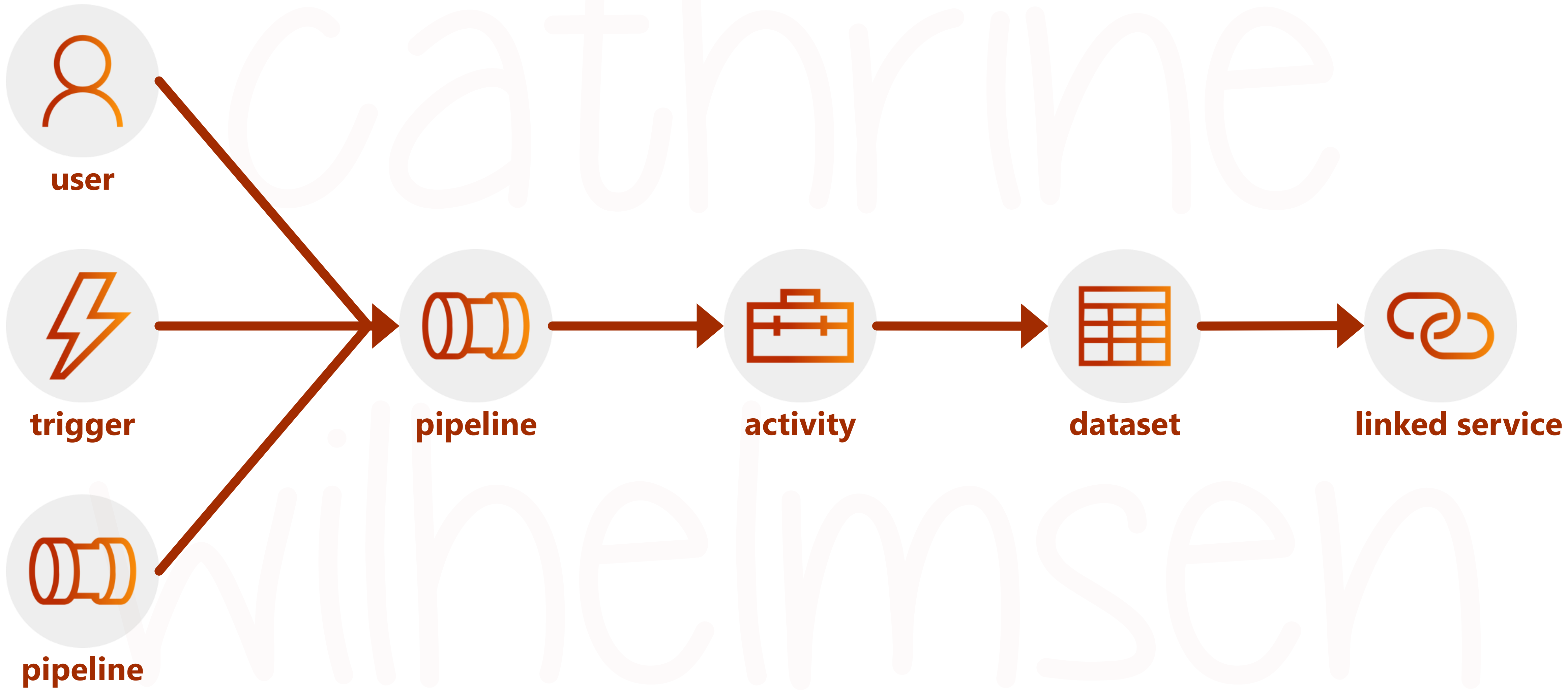
What are Parameters?

Use parameters to pass external values into:

- Pipelines
- Datasets
- Linked Services
- Mapping Data Flows

Once passed, parameter values cannot be changed

How are Parameters passed?



Parameters

@pipeline().parameters. ParameterName

@dataset(). ParameterName

@linkedService(). ParameterName

Parameters

@pipeline().**parameters**.ParameterNa

@dataset().Param Name

@linkedService().ParameterName

Parameters and System Variables

@pipeline().parameters. ParameterName

@pipeline().DataFactory

@pipeline().TriggerTime

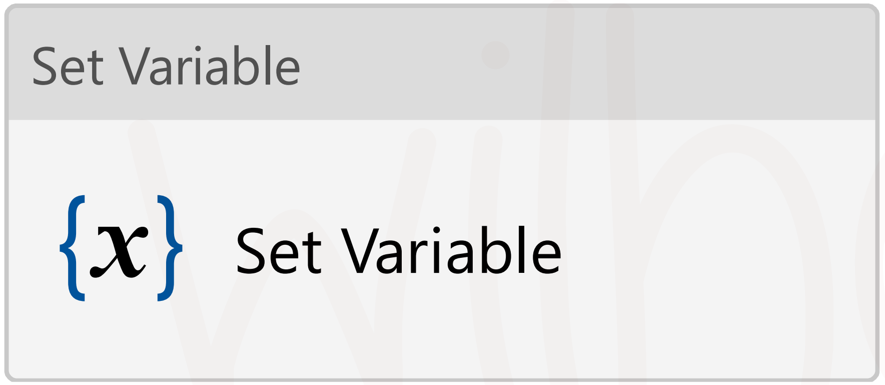
What are Variables?

Parameters are external values passed into pipelines, while variables live inside pipelines

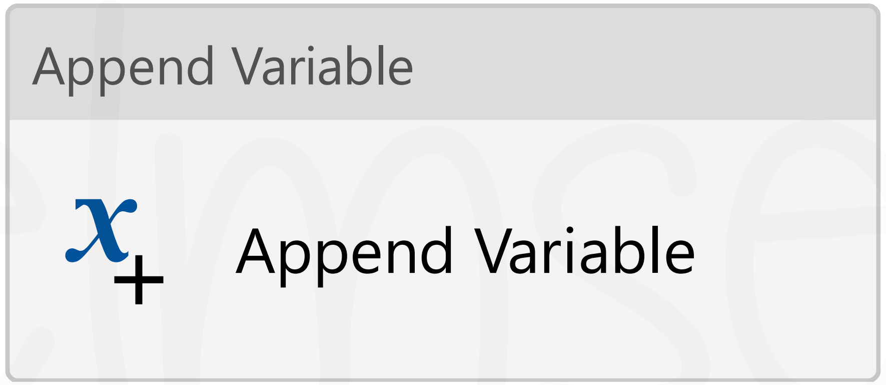
Use variables when you need to change values during pipeline execution

How are variables controlled?

Use **Set Variable** and **Append Variable** activities to control values during pipeline execution


Set Variable

Set Variable


Append Variable

Append Variable

Variables

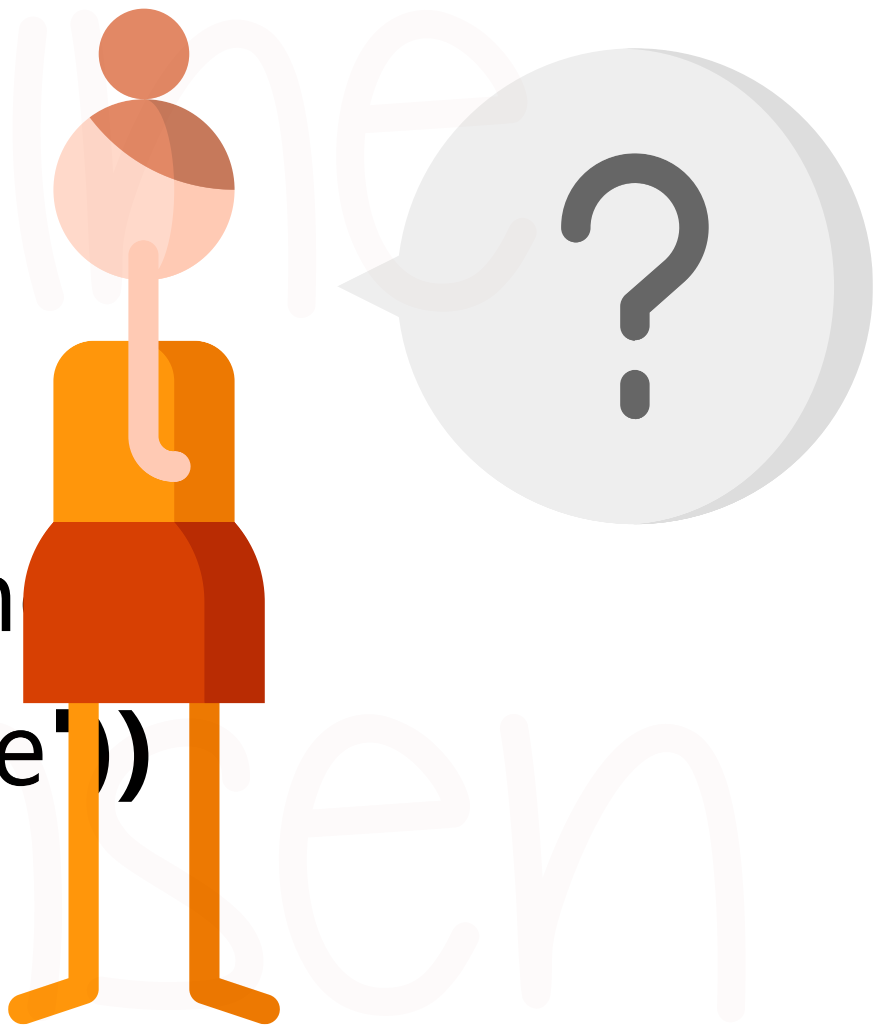
```
@variables('VariableName')

@first(variables('VariableName'))
@last(variables('VariableName'))
```

Variables

```
@variables('VariableName')

@first(variables('VariableNam
@last(variables('VariableName'))
```



Expressions and Functions

What are Expressions?

Use expressions to modify values during runtime

The entire expression starts with the **@** symbol:

```
"@toUpper(pipeline().parameters.FileName)"
```

At runtime, expressions are evaluated to literal string values:

```
"COLORS.CSV"
```

What are Functions?

String:	concat, substring, startswith, endswith …
Date:	adddays, addhours, formatDateTime …
Collection:	contains, first, last, length …
Logical:	if, and, or, equals, less, greater …
Conversion:	createArray, binary, json, xml, …
Math:	add, sub, mul, div, min, max, mod, rand …

Combining Strings

How to combine strings?

A common task is to combine strings, for example multiple parameters or static text with a parameter

Combine strings in two ways:

- String Concatenation: **`@concat()`**
- String Interpolation: **`@{…}`**

What is String Concatenation?

```
@concat(
    ' TRUNCATE TABLE dbo. ',
    pipeline().parameters.TableName
)
```

What is String Interpolation?

```
TRUNCATE TABLE dbo.
@{pipeline().parameters.TableName}
```

DEMO

Let's add some parameters!

Loops

ForEach Loop Input

The ForEach loop iterates over a *collection*:

- Array
- JSON Object

Reference the current iteration using **`@item()`**

Array Items

Array Items

```
["colors", "parts", "sets"]
```

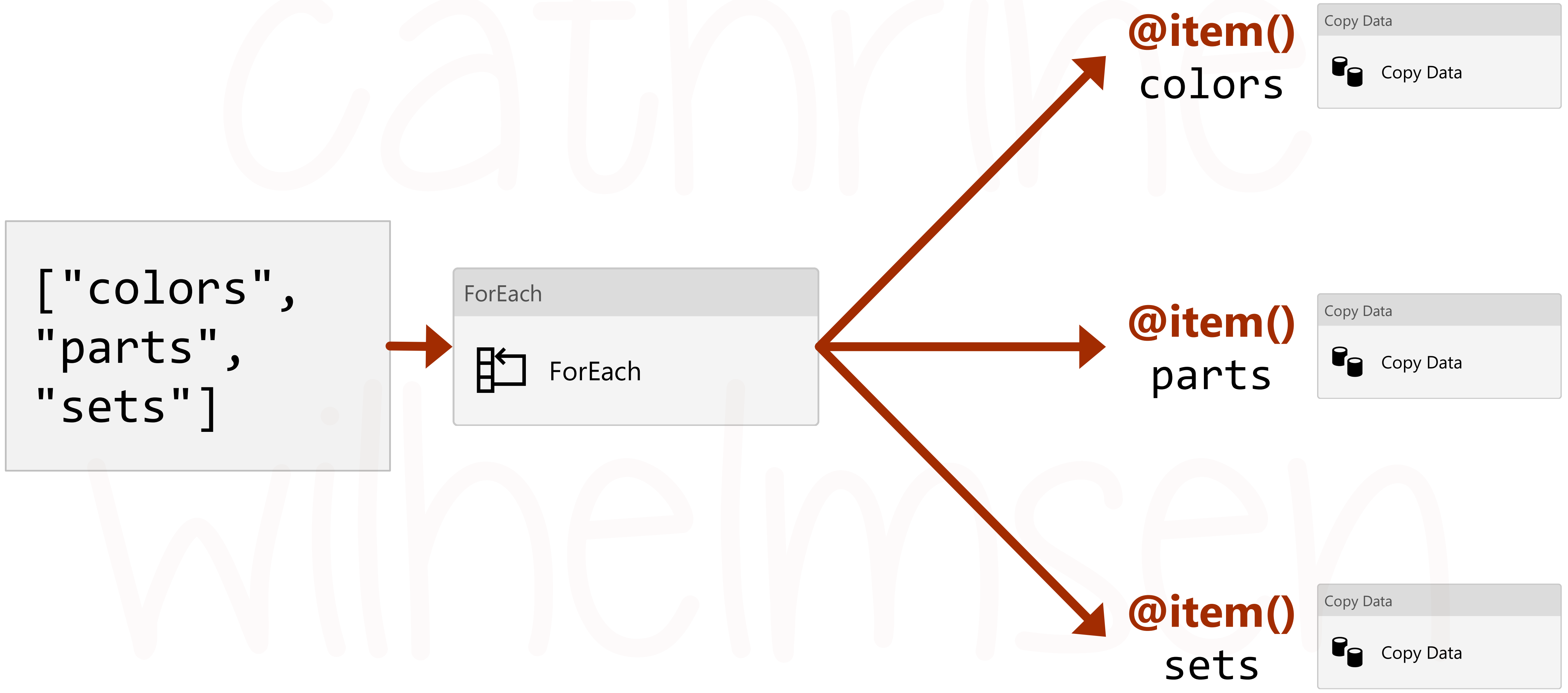
Array Items

```
["colors", "parts", "sets"]
```

↑

@item()

Array Items


["colors",
"parts",
"sets"]
ForEach
ForEach
@item()
colors
@item()
parts
@item()
sets
Copy Data
Copy Data
Copy Data
Copy Data
Copy Data
Copy Data

Object Items

Object Items

```
{ "Files": [
   { "Name": "colors", "Extension": "csv" },
   { "Name": "parts",  "Extension": "csv" },
   { "Name": "sets",   "Extension": "csv" }
] }
```

Object Items

@item()

```
{ "Files": [
   { "Name": "colors",  "Extension": "csv" },
   { "Name": "parts",   "Extension": "csv" },
   { "Name": "sets",    "Extension": "csv" }
] }
```

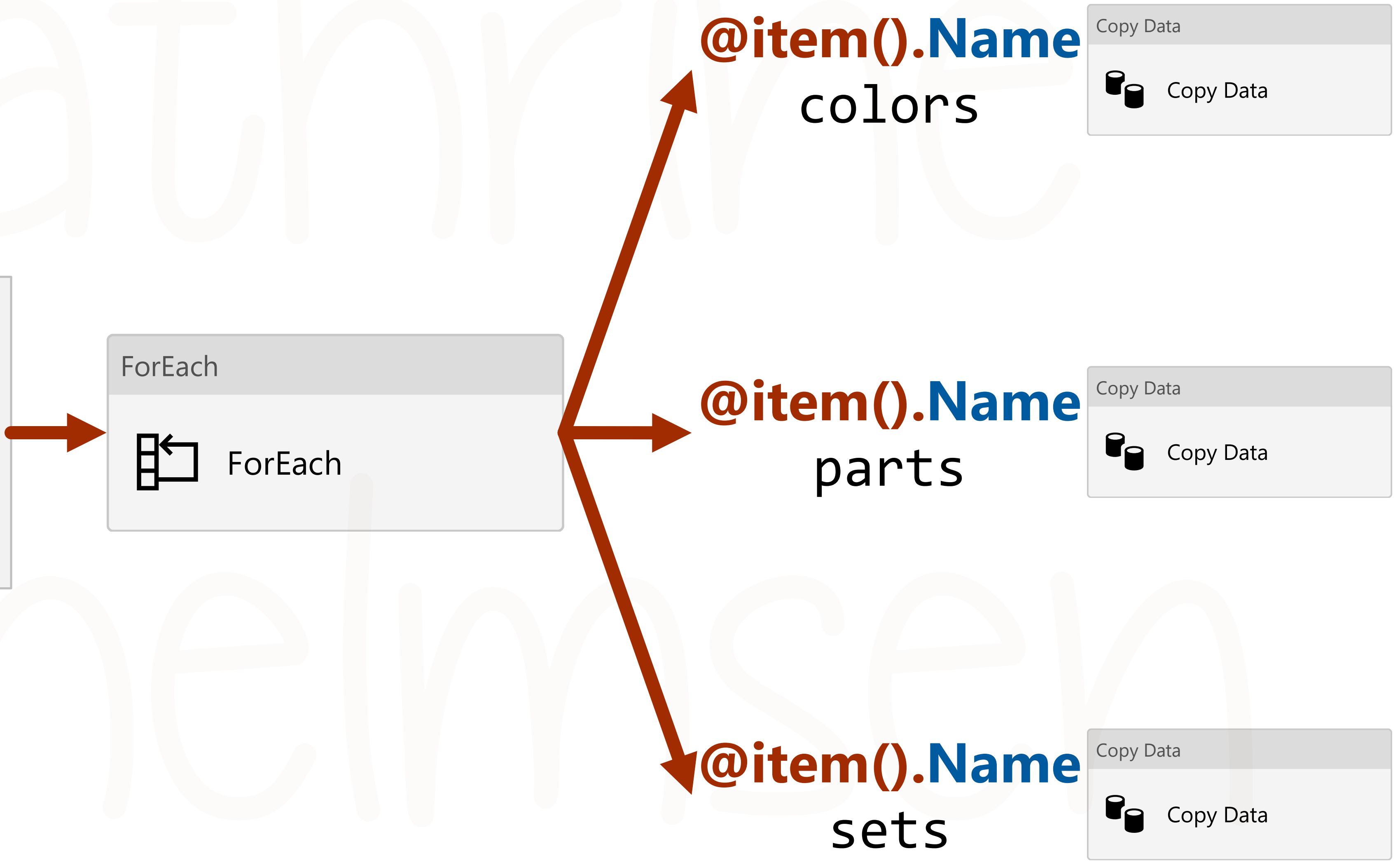
Object Items

```
{ "Files": [
    { "Name": "colors", "Extension": "csv" },
    { "Name": "parts",  "Extension": "csv" },
    { "Name": "sets",   "Extension": "csv" }
] }
```

@item().Name

Object Items

```
{ "Files": [
  { "Name": "colors" … },
  { "Name": "parts" … },
  { "Name": "sets" … }
] }
```



DEMO

Let's add a loop!

Lookups

Lookup

Retrieve dataset from:

- Configuration File / Table
- Query
- Stored Procedure

Use to dynamically get objects to operate on in subsequent activity

Lookup Output

Two types of outputs:

- First Row Only
- All Rows

Lookup Output

```
{
  "firstRow" :
    {
      "Name" : "colors",
      "Extension" : "csv"
    }
}
```

```
{
  "count" : "2",
  "value" : [
    {
      "Name" : "parts",
      "Extension" : "csv"
    },
    {
      "Name" : "sets",
      "Extension" : "csv"
    }
  ]
}
```

Lookup Output: First Row Only

```
{
  "firstRow" :
    {
      "Name" : "colors",
      "Extension" : "csv"
    }
}
```

@{activity('Lookup')
.output.firstRow}

Lookup Output: First Row Only

```
{
  "firstRow" :
    {
      "Name" : "colors",
      "Extension" : "csv"
    }
}
```

@{activity('Lookup')
.output.firstRow
.**Name**}

Lookup Output: All Rows

@{activity('Lookup')
.output.value}

```
{
  "count" : "2",
  "value" : [
    {
      "Name" : "parts",
      "Extension" : "csv"
    },
    {
      "Name" : "sets",
      "Extension" : "csv"
    }
  ]
}
```

Lookup Output

First Row Only:

Use to pass single values to another activity

```
@{activity('Lookup').output.firstRow.ColumnName}
```

All Rows:

Use to pass entire collection to another activity

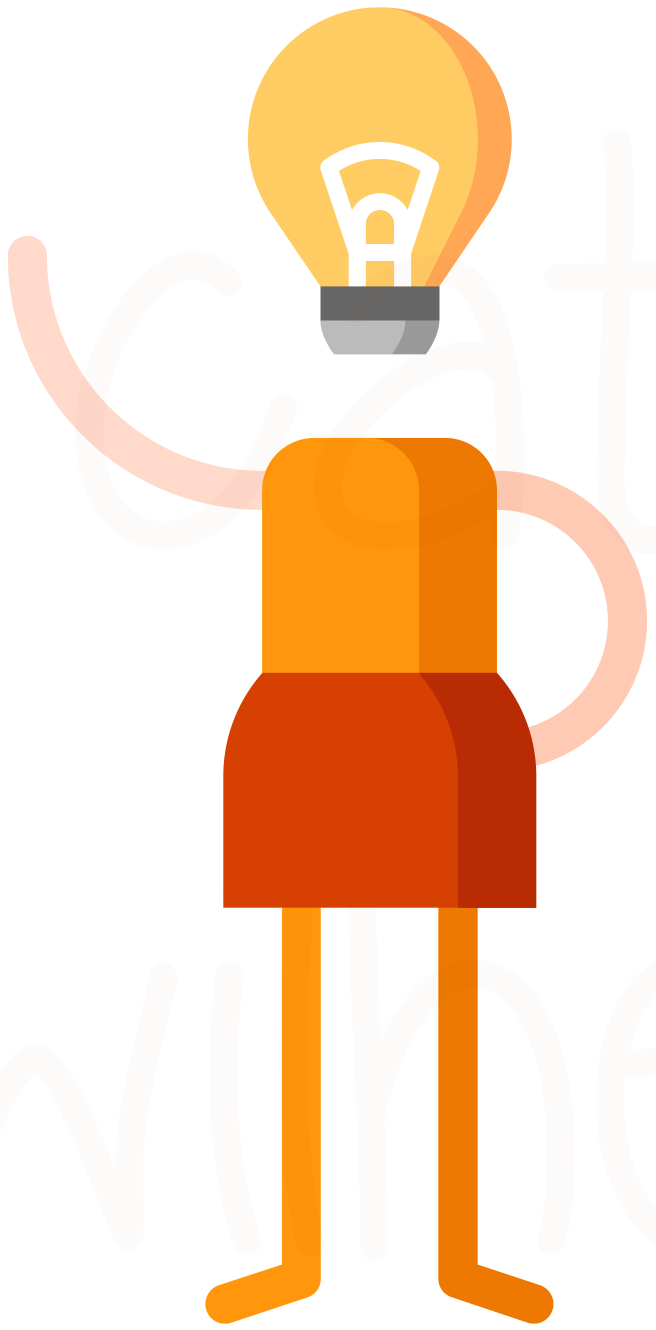
```
@{activity('Lookup').output.value}
```

DEMO

Let's add a lookup!

How do you make dynamic solutions?

1. Create hardcoded solution to test pattern
2. Replace hardcoded properties with parameters
3. Execute the parameterized solution in a loop
4. Control loop from configuration table



Good luck!

thank you!

hi@cathrinew.net

@cathrinew

CW cathrinew.net

inmeta EVANGELIST MICROSOFT AZURE

Microsoft