Lab 3 – Create a reusable pipeline

The pipeline you authored in Lab 2 copied a file from a website – GitHub – into your data lake. In this lab you will use linked service and dataset **parameters**, in combination with ADF pipeline **expressions**, to build a reusable pipeline – a single pipeline that can perform this task for any of the AdventureWorks files stored in GitHub.

Lab 3.1 – Create a generic HTTP linked service & dataset

In Lab 2.1 you created a linked service containing the full URL path to the AdventureWorks "Products.csv" file. We'll improve on that by creating a linked service and dataset that can refer to any of the AdventureWorks sample files on GitHub.

1. Navigate to the Manage hub, open the "Linked services" page and click "+ New". Choose a linked service of type "HTTP" and click "Continue".

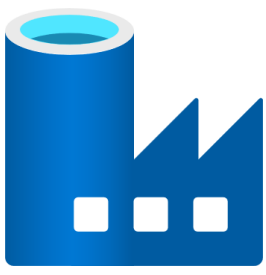
2. Configure the linked service like this:

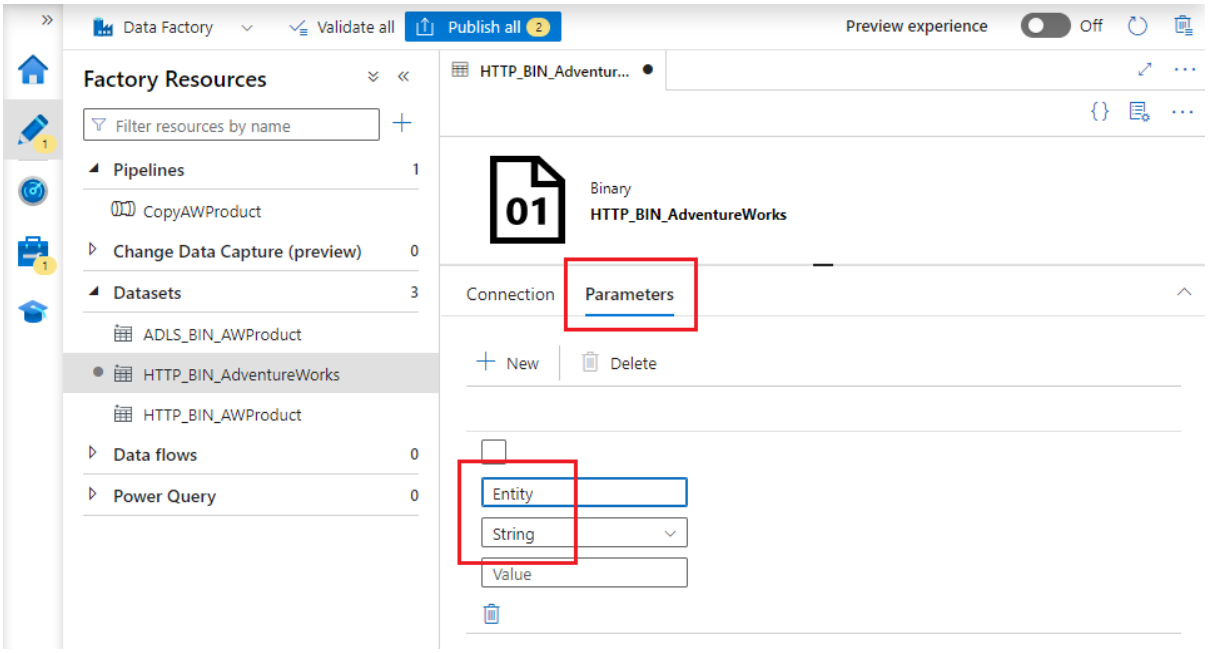
    - Give it the **Name** "HTTP_AWGitHub".
    - Set its **Base URL** to "https://raw.githubusercontent.com/microsoft/sql-server-samples/master/samples/databases/adventure-works/oltp-install-script/". This is the first part of the URL you used in Lab 2.1 – the **folder** containing AdventureWorks data files.
    - Set **Authentication type** to "Anonymous".
    - Click "Create".

3. Navigate to the Authoring experience and use the "Dataset Actions" menu to create a new dataset.

    - Choose the "HTTP" data store.
    - Choose the "Binary" file format.
    - Name the dataset "HTTP_BIN_AdventureWorks", and select the new "HTTP_AWGitHub" linked service.
    - Leave "Relative URL" blank again – we will come back to it in a moment – and click OK.

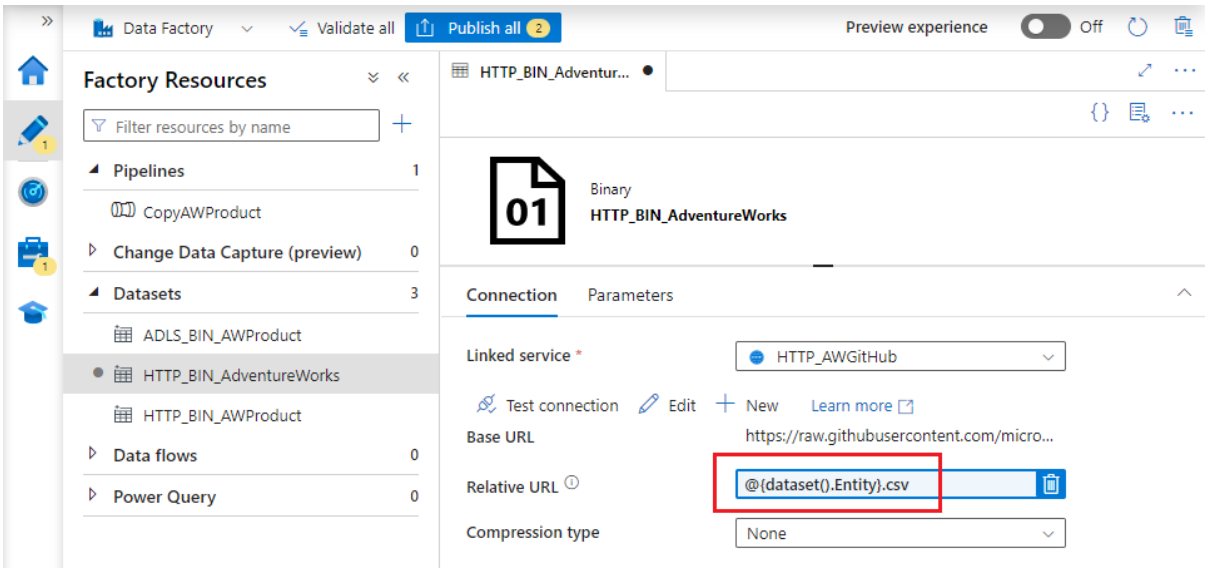
4. To reuse the dataset, we need to be able to pass in different relative URL values at runtime, depending which file is required. We can do this using a **dataset parameter**. Use the "Parameters" configuration tab to create a new String parameter called "Entity".





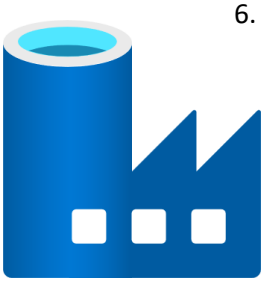
5. To use the parameter value passed in at runtime, the relative URL will require a pipeline **expression** that references the parameter. Every file in the collection has a ".csv" extension, so we can also include that in the expression (and save users of the pipeline from having to specify the same extension repeatedly).

   Enter an expression to achieve this in the "Connection" tab's "Relative URL" field. A possible implementation is **`@{dataset().Entity}.csv`** – this expression uses string interpolation to avoid a less readable `@concat` function.



   You must enter the expression using the Pipeline expression builder, accessed by clicking the "Add dynamic content" link. The link appears below the "Relative URL" text field when you click into the box. A field which contains an expression has a blue background (as in the screenshot above) – if your field background is white, you have entered a string constant value.

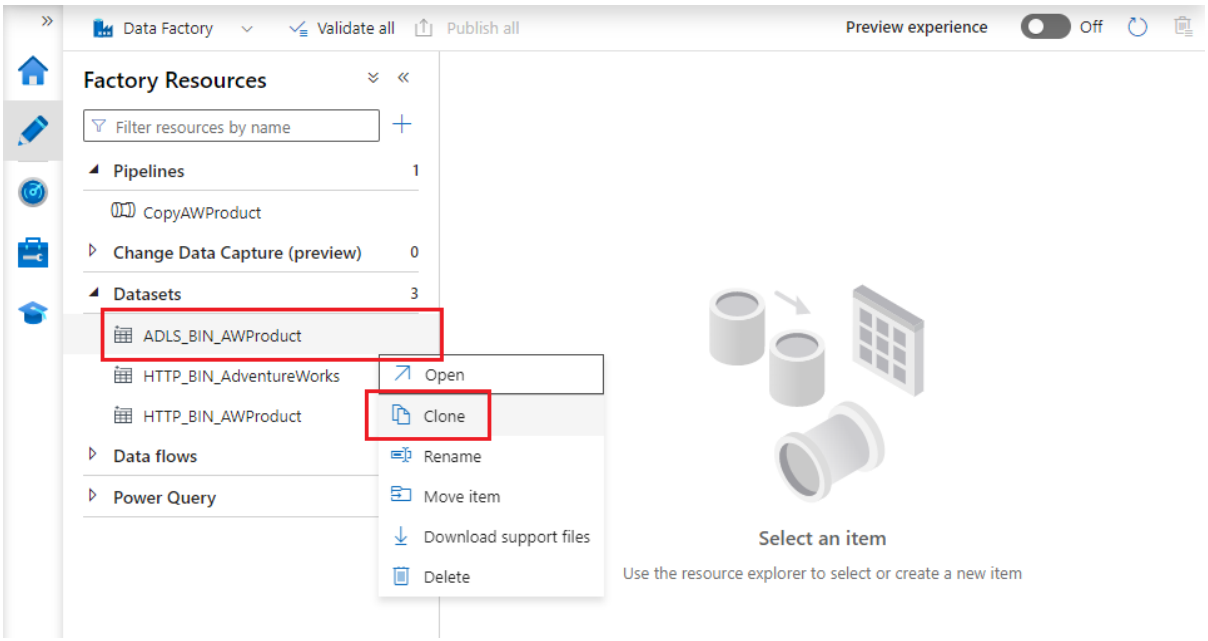
6. Use the "Publish all" button to publish your new resources to ADF.



Lab 3.2 – Create a generic data lake dataset

By parameterising the HTTP dataset you created in Lab 3.1, you enabled it to accept different file names at runtime. We need now to perform the equivalent task for the data lake dataset.

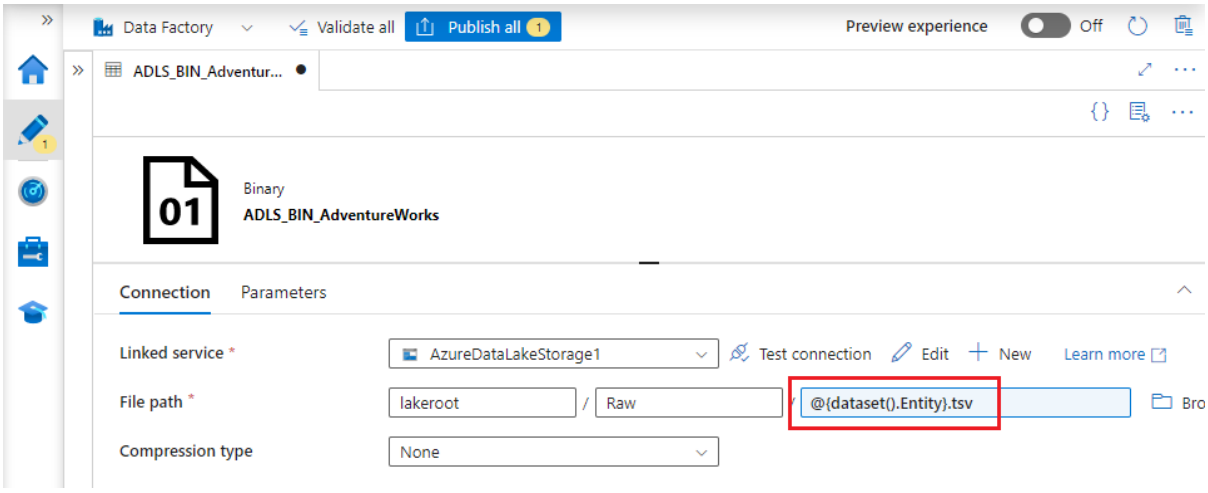
1. Create a new dataset, this time by cloning the "ADLS_BIN_AWProduct" dataset created in Lab 2. Locate the dataset in the "Factory Resources" menu, then use the "Clone" option on its "Actions" menu.



2. Rename the cloned dataset "ADLS_BIN_AdventureWorks"

3. Create a dataset parameter, called "Entity" and of type String.

4. Replace "Product.csv" in the "File name" section of the "Connection" tab's "File path" with an expression. In the screenshot we have changed the extension of the data lake file to "tsv", indicating more clearly that the file is tab-separated.



5. Publish the dataset to ADF.

Lab 3.3 – Create a generic pipeline

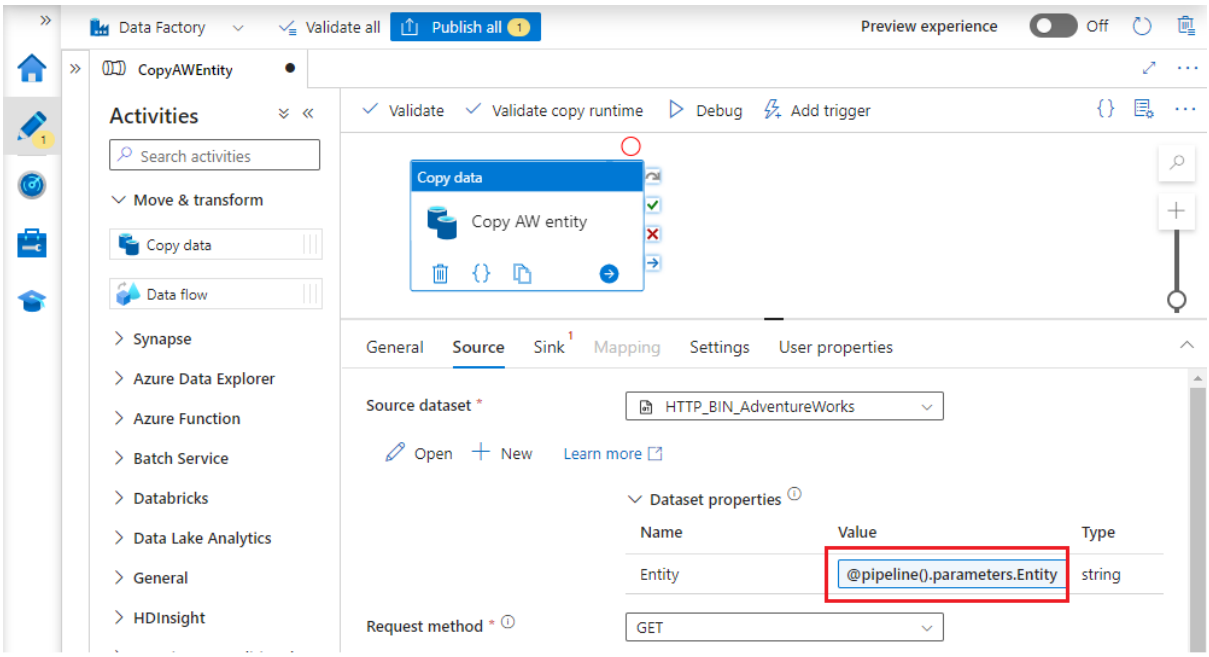
A pipeline using the two datasets you have created must provide runtime values for their parameters. Instead of hard coding them into the pipeline definition, use **pipeline parameters** to make the pipeline generic as well.

1. Create a new ADF pipeline called “CopyAWEntity”.

2. On the pipeline’s “Parameters” tab, create a String parameter called “Entity”.

3. Drag a “Copy” activity from the “Move & transform” section of the activity toolbox and drop it onto the pipeline canvas. Give the activity a meaningful name.

4. On the activity’s “Source” tab, choose dataset “HTTP_BIN_AdventureWorks”. A field in which to specify a value for the **dataset**’s parameter will appear – supply an expression to pass in the **pipeline** parameter value.



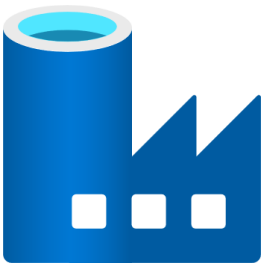
The expression in the screenshot passes the pipeline’s parameter value without modification – remember that the file extension will be appended by the “HTTP_BIN_AdventureWorks” dataset.

5. Configure the activity’s “Sink” tab similarly, choosing dataset “ADLS_BIN_AdventureWorks” and passing the pipeline’s parameter value to the dataset.

6. Publish the pipeline, then run it a few times in “Debug” mode, specifying appropriate parameter values, e.g.:

   - BillOfMaterials
   - BusinessEntity
   - ContactType

   Verify that the expected filenames and contents appear in the data lake’s “Raw” folder.

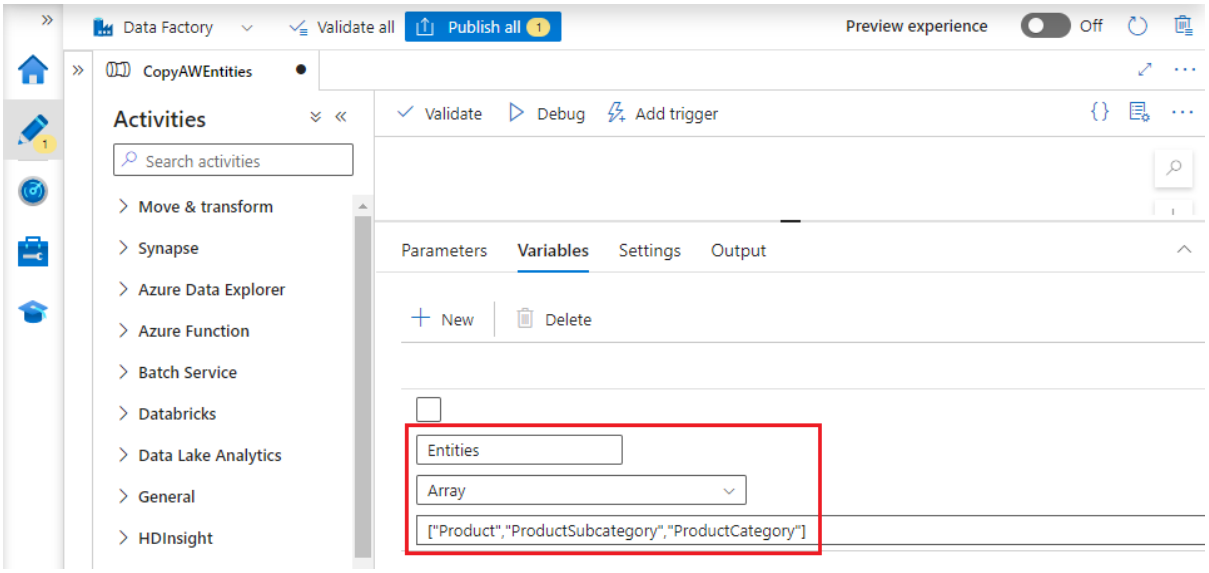


Lab 3.4 – Reuse the generic pipeline

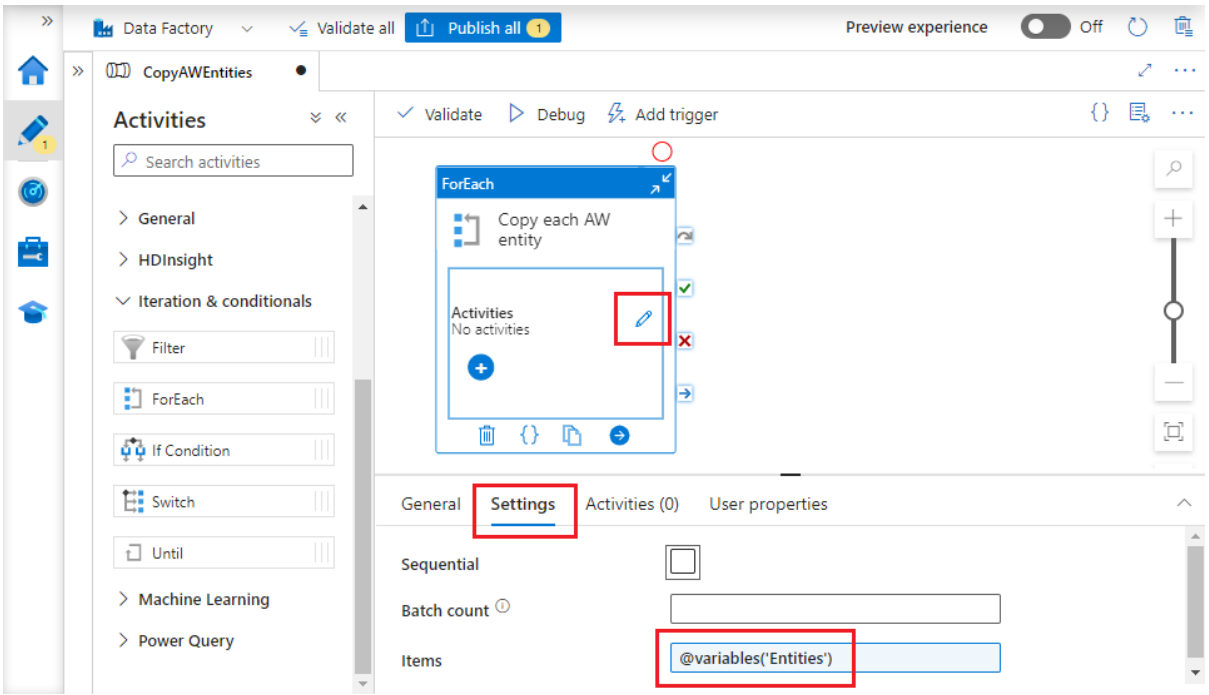
Your generic pipeline can be used to extract any ".csv" file from the AdventureWorks sample data folder on GitHub. In this lab you will automate the extraction of multiple files.

1. Create a new pipeline called "CopyAWEntities".

2. The new pipeline will store a list of entities in an array variable, then use a ForEach activity to iterate over the array. Use the pipeline's "Variables" configuration tab to create an array variable with the value `["Product","ProductSubcategory","ProductCategory"]`.

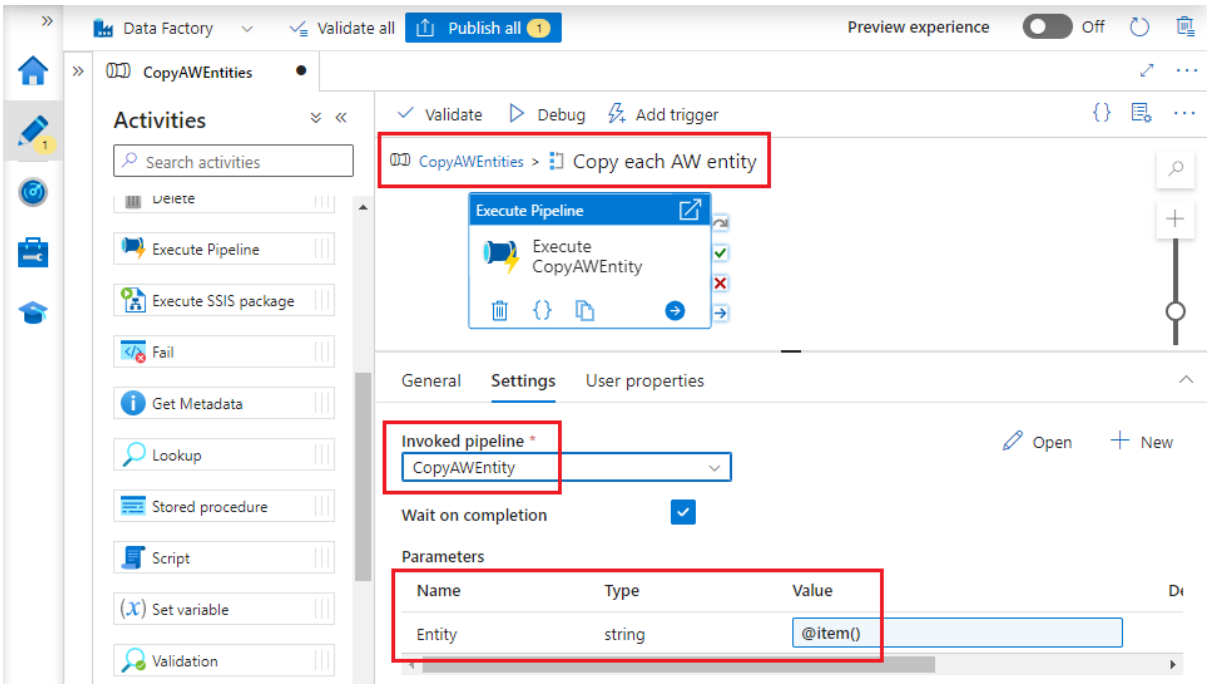


3. Drag a ForEach activity from the "Iteration & conditionals" section of the Activities toolbox and onto the pipeline canvas. Use its "Settings" tab to configure the "Items" collection – this is the collection of items over which the activity will iterate.



4. Click the "pencil" icon on the ForEach activity (indicated in the screenshot) to open the sub-pipeline to be executed at each iteration. Drag an "Execute pipeline" activity onto the sub-pipeline canvas and name it appropriately.

5. Configure the activity to call the generic "CopyAWEntity" pipeline. A value is required for that pipeline's "Entity" parameter – use the **`@item()`** function to refer to the array element under consideration during each iteration.



6. Publish your changes and run the pipeline. When you run the pipeline using "Debug", ADF Studio warns you that the ForEach activity will handle its input collection sequentially. This is a debug-specific feature – default ForEach behaviour in published pipelines is to handle array elements in parallel. This makes the activity a powerful way to execute multiple data processing activities simultaneously.

7. When the pipeline's execution has successfully completed, inspect the "lakeroot" container's "Raw" folder to verify that it now contains the three files: Product.tsv, ProductSubcategory.tsv and ProductCategory.tsv.

Recap

In Lab 3 you have:

- created reusable datasets to represent different source and data lake files generically
- created a generic pipeline using those generic datasets
- reused the generic pipeline in a ForEach activity, enabling a single pipeline to download multiple source files using only two activities.