Microsoft Fabric: Pipelines

Scenario2: We want to copy all the files which are there in github folder, so we need to use Dynamic pipelines or parameterized pipeline.

Create Parameterized Pipelines to fetch all the files from github in one go

Fetch multiple files from GitHub dynamically and load them into OneLake using a **parameterized pipeline** in Fabric.

☑ Step-by-Step Breakdown

◇ Step 1: Create and Upload JSON File

You created a JSON like this:

```
json
CopyEdit
[
  {
    "p_rel_url": "pragyachoukade/Fabric-
Tutorial/refs/heads/main/Data/AdventureWorks_Customers.csv",
    "p_sink_folder": "AdventureWorks_Customers",
    "p_sink_file": "AdventureWorks_Customers.csv"
  },
  ...
]
```

✅ You uploaded this file into your **Lakehouse → Files** section, e.g.:

```
bash
CopyEdit
/Files/metadata/github_file_list.json
```

◇ Step 2: Create a New Data Pipeline

1. Go to your **Fabric Workspace**
2. Click **New → Data Pipeline**
3. Name it something like `GitHubToOneLake_Dynamic`

◇ Step 3: Add a Lookup Activity

1. Drag a **Lookup** activity onto the canvas
2. Configure it as:

🔧 *Source:*

- **Source type**: Lakehouse
- **File path**:

```
bash
CopyEdit
/Files/metadata/github_file_list.json
```

- **File format**: JSON
- **First row as header**: ⛔ (Not needed for JSON)
- **Data type detection**: Enable

✅ This reads your JSON file and loads it as a dynamic list of file definitions.

◇ Step 4: Add a ForEach Activity

1. Drag a **ForEach** activity
2. Connect the output of the Lookup to ForEach
3. In **Settings** tab of ForEach:
    a. **Items**:

```
kotlin
CopyEdit
@activity('Lookup1').output.value
```

✅ This loops through each object in the JSON array (each file row).

◇ Step 5: Add a Copy Data Activity Inside ForEach

1. Drag a **Copy Data** activity inside ForEach

🔧 *Source Configuration:*

- **Source type**: HTTP (GitHub)
- Use a **parameterized relative URL**:

```
kotlin
CopyEdit
@concat('https://raw.githubusercontent.com/', item().p_rel_url)
```

✅ This dynamically builds the Raw GitHub URL for each file.

📝 *Example:*

```
text
CopyEdit
https://raw.githubusercontent.com/pragyachoukade/Fabric-Tutorial/main/Data/AdventureWorks_Customers.csv
```

⚠️ Tip: Your current `"p_rel_url"` values contain `refs/heads` which are part of Git API URLs, not Raw URLs.
🔁 You should **replace**:

```
css
CopyEdit
refs/heads/main
→
main
```

✅ Corrected example:

```
json
CopyEdit
"p_rel_url": "pragyachoukade/Fabric-Tutorial/main/Data/AdventureWorks_Customers.csv"
```

🔧 *Sink Configuration:*

- **Sink type**: Lakehouse
- **Destination folder**: Use dynamic value:

```
java
CopyEdit
@item().p_sink_folder
```

- **File name**:

```
java
CopyEdit
@item().p_sink_file
```

✅ This writes each file into its respective subfolder or table.

◇ Step 6: Run and Monitor the Pipeline

- Click **Validate** to check everything is configured correctly
- Click **Run**
- Open the **Monitor tab** to view pipeline execution
- Confirm the files show up in your **Lakehouse → Files** or **Tables**

Scenario 2: Now we will fetch data from Azure Data Lake to Fabric Lakehouse

☑ Prerequisites

Before we start:

- You must have access to both:
  - Azure Storage Account (ADLS Gen2)
  - Fabric workspace with a **Lakehouse**
- Recommended: **SAS token** or **role-based access** from ADLS to Fabric user identity
- Know your container, storage account name, and file path

🧿 STEP-BY-STEP: Fetch Data from Azure Data Lake → Fabric Lakehouse

◇ Step 1: Open Microsoft Fabric Workspace

1. Go to https://app.fabric.microsoft.com
2. Select your Fabric-enabled workspace

◇ Step 2: Create or Open a Lakehouse

1. Go to **Lakehouse** tab
2. Create a new Lakehouse or use an existing one (note the name)

◇ Step 3: Create a Data Pipeline

1. Click **"New"** → Choose **"Data pipeline"**
2. Drag a **Copy Data** activity to the canvas

◇ Step 4: Configure Source (Azure Data Lake)

1. Click on **Source** in the Copy Activity
2. Click **+ New** to create a **Linked Service**
3. Choose **Azure Data Lake Storage Gen2**
4. Provide:
    a. **Name**: e.g., `ADLS_Storage_Link`
    b. **Authentication type**:
        i. If using identity → Sign in using the correct Azure account (must have role)
        ii. If using SAS → paste full URL with token
5. After connecting, **browse to your container & file** (CSV/Parquet/etc.)
6. Select **file format** and delimiter (if CSV)

◇ Step 5: Configure Sink (Lakehouse)

1. Click on **Sink**
2. Choose your **existing Lakehouse**
3. Select destination folder (or type it — like `Files/RawData`)
4. Set file format: usually **DelimitedText (CSV)** or **Parquet**

◇ Step 6: Validate and Run Pipeline

1. Click **Validate** on the top toolbar
2. If no errors, click **Publish all**
3. Then click **Run**

◇ Step 7: Monitor Progress

1. Go to the **Monitor tab**
2. Track pipeline run status (Success/Failed)
3. On success, go back to your **Lakehouse** → check `Files` or `Tables` folder

We will create a Dataflow to transform the raw data that has been ingested into the Lakehouse.

🧭 OVERVIEW OF STEPS:

1. **Create a Dataflow Gen2** to transform source data
2. **Define transformations** (like remove columns, split, filter, etc.)
3. **Add Lakehouse as destination** in Dataflow
4. **Save and publish Dataflow**
5. **Create a Data Pipeline**, add **Dataflow activity**
6. **Run pipeline to write transformed data to Lakehouse**

☑ STEP-BY-STEP GUIDE

◇ STEP 1: Create Dataflow Gen2

1. Go to your Fabric Workspace
2. Click **"New" → "Dataflow Gen2"**
3. Choose **"Blank Dataflow Gen2"**
4. Click **"Add new data"** → choose **Lakehouse** as source
5. Browse and select your **existing Lakehouse**
6. Choose the CSV or table you just ingested
7. Click **"Next"** → it will open in **Power Query Online (transform editor)**

◇ STEP 2: Apply Transformations in Power Query

You're now inside the Power Query editor for Dataflow.

Here you can:

- Remove columns
- Rename columns
- Change data types
- Filter rows

- Replace nulls
- Add calculated columns

Example:

- Right-click column → **Remove**
- Use “Transform” tab to change data types
- Use “Add column” → Custom column (like DAX logic)

⚠️ If ribbon options like "Remove Rows" are missing, use formula bar or right-click menus.

◇ STEP 3: Add Destination (Lakehouse)

Once your transformation is done:

1. Click on **"Add destination"**
2. Choose **"Lakehouse"** as destination
3. Select the **same Lakehouse** or a different one if desired
4. Provide:
    a. **Destination Table Name** (e.g., `Product_Transformed`)
    b. Choose: **Replace** or **Append** if table already exists
5. Click **Next** → then **Save & Close**

◇ STEP 4: Publish the Dataflow

1. Name your Dataflow (e.g., `ProductData_Transform`)
2. Click **Publish** (top-right)

🎉 You’ve now created and published a Dataflow that will load transformed data to Lakehouse.

◇ STEP 5: Create a Pipeline to Run the Dataflow

Now let’s automate it.

1. Go back to the **Fabric workspace**
2. Click **“New → Data Pipeline”**
3. Drag **“Dataflow” activity** onto canvas
4. Click on the activity → In Settings:
    a. **Choose the Dataflow Gen2** you just published (`ProductData_Transform`)
    b. Select **Workspace** and **Dataflow name**

◇ STEP 6: Publish and Run the Pipeline

1. Click **"Publish all"**
2. Click **"Run"**
3. Go to the **Monitor tab** to track progress

✅ Once it runs, the transformed data will be available inside your Lakehouse as a **table** or **CSV in Files folder**, based on your Dataflow output settings.

By All these pipelines copying data from git, from ADLS and then to push transformed data using Dataflow pipeline we will now create a parent pipeline that automates the task

✅ Steps to Use Invoke Pipeline (Preview) in Microsoft Fabric

🎯 Goal:

To call (trigger) existing child pipelines from a **master pipeline** using **Invoke Pipeline (Preview)** activity.

◇ STEP 1: Open or Create the Master Pipeline

1. In your Fabric workspace, click **New → Data Pipeline**
2. Name it something like `Master_Orchestration_Pipeline`

◇ STEP 2: Add "Invoke Pipeline (Preview)" Activities

1. From the toolbox (left panel), search or scroll to find
   ➤ **Invoke Pipeline (Preview)**
2. Drag **three instances** of it to the canvas
   ➤ One for Git copy
   ➤ One for Azure Data Lake copy
   ➤ One for Dataflow execution

◇ STEP 3: Configure Each Invoke Pipeline Activity

🔧 For Each Activity:

1. Click on the **Invoke Pipeline** block
2. Go to the **Settings** tab
3. Under **Pipeline**, click **Select a pipeline**
4. Choose the corresponding child pipeline:
    a. GitHub Copy → `Git_DynamicCopy_Pipeline`
    b. Azure Copy → `Azure_ADLS_Copy_Pipeline`
    c. Dataflow → `Dataflow_Execution_Pipeline`

✅ If the pipeline has parameters, they'll appear below — you can map them here using dynamic content or static values.

◇ STEP 4: Arrange Sequence (Optional)

1. Connect the activities in the order you want (e.g., Git → Azure → Dataflow)
2. Drag arrows to control **execution flow**:
    a. Sequential (1 → 2 → 3)
    b. Parallel (trigger all at once)

◇ STEP 5: Publish and Run

1. Click **Publish All**
2. Then click **Run**
3. Go to the **Monitor** tab to see the run progress and logs

We can apply trigger after this

☑ How to Apply a Trigger to Your Master Pipeline

◇ Step 1: Open the Master Pipeline

1. Go to your Fabric workspace

2. Open the pipeline you want to schedule (e.g., `Master_Orchestration_Pipeline`)

◇ Step 2: Click on "Add Trigger"

1. In the top ribbon, click **"Add Trigger"**
2. Select **"New/Edit"**

◇ Step 3: Configure the Trigger

You'll see a trigger configuration screen.

Choose one of the following:

Trigger Type	Use When
Scheduled	You want to run the pipeline on a fixed schedule (daily, weekly, etc.)
Event-based *(Coming soon)*	When reacting to file uploads or changes (not yet fully supported in Fabric)
Manual	Run on-demand only

For Scheduled Trigger:

- **Name**: `DailyPipelineRun`
- **Recurrence**: e.g., Every 1 day at 8 AM
- **Time Zone**: Make sure to select `(UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi` (for IST)

◇ Step 4: Attach the Trigger

1. After setting the schedule, click **Next**
2. Confirm it's attached to the correct pipeline
3. Click **Finish**

Now the trigger is live ✅