**AZURE DATA FACTORY**

**INDEX**

|  |  |
| --- | --- |
| **S.No** | **Topics** |
| **1** | **What is Azure Data Factory**  Why do we need ADF |
| **2** | **What is resource group**  Why we use resource group |
| **3** | **Components of ADF**   Pipelines   Activities   Data Flows   Data sets   Linked Services   Integration Run times   Triggers |
| **4** | **Core Components of ADF**   Data Source   Data Movement   Data Transformation   Monitoring |
| **5** | **Prerequisites of project flow**   Creating Storage Account   Creating resource group |
| **6** | **Project flow**   Connecting to the integration runtime   Connecting to linked services   Connecting to the datasets   Creating Pipelines   Scheduling and triggering |
| **7** | **Data Flow** |
| **8** | **Control Flow** |
| **9** | **Variables** |
| **10** | **Functions** |

**Azure Data Factory**

Azure Data Factory (ADF) is a cloud-based data integration service provided by Microsoft. It helps you create work flows (called pipelines) to move, transform, and load data from various sources to a target destination. It is commonly used for ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform) processes in modern data engineering.

Before Azure Data Factory (ADF), Microsoft SQL Server Integration Services (SSIS) served as the primary ETL tool for data integration and transformation tasks.ADF can be considered an evolution of SSIS, designed for modern cloud-based data work flows with enhanced scalability and flexibility.

Now, Microsoft Fabric builds upon ADF’s capabilities, offering an integrated platform that includes all key components of ADF alongside additional advanced features. Fabric provides a unified analytics ecosystem for data integration, transformation, and analysis, streamlining the entire data life cycle within a single environment.

**Why do we need ADF**

**Connect to multiple data sources**: On-premises, cloud, or hybrid.

**Automate tasks**: Easily schedule and trigger data work flows.

**Handle complex work flows**: Perform data cleaning, transformation, and loading.

**Cost-effective:** No need to maintain infrastructure manually.

**Scalable and reliable**: Process large data sets with ease.

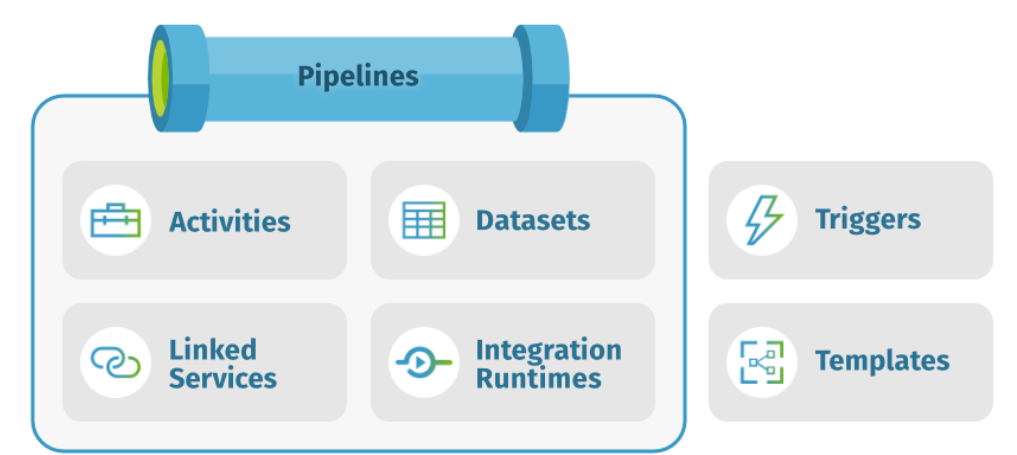
**Resource Group**

A container in Azure that holds related resources for an application, making it easier to manage, organize, and monitor resources collectively.

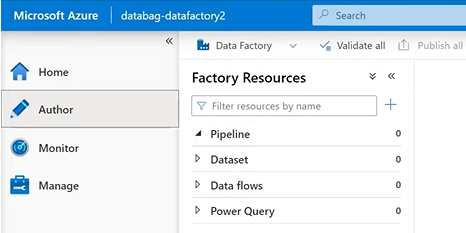
**Why We Use Resource Group** : We use a resource group in Azure to logically organize and manage resources like Azure Data Factory (ADF) for easier provisioning, management, and access control. It allows us to group related services, apply consistent policies, and manage them as a unit, streamlining deployment and lifecycle management.

**Components of Azure Data Factory**

* Pipelines
* Activities
* Data Flows
* Data sets
* Linked Services
* Integration Run times
* Triggers

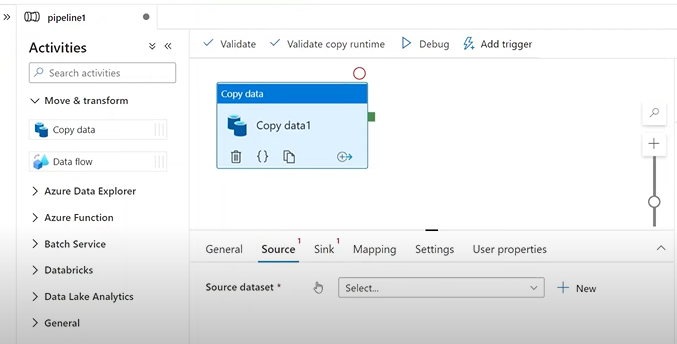


**Pipelines :**   
  
A data factory might have one or more pipelines. A group of activities is called a Pipeline.A pipeline is a logical grouping of activities that perform a unit of work. Together, the activities in a pipeline perform a task.



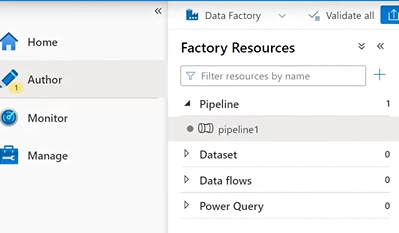
**Activity :**

Activities represent a processing step in a pipeline. For example, you might use a copy activity to copy data from one data store to another data store.

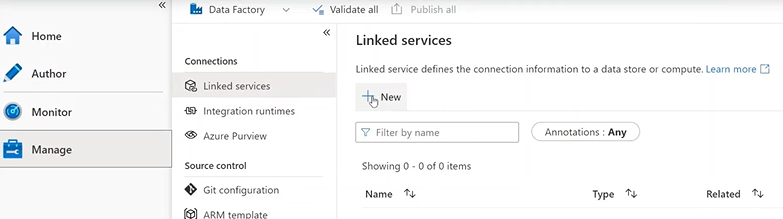
**Data Sets :**

Data sets represent data structures within the data stores, which simply point to or reference the data you want to use in your activities as inputs or outputs.



**Linked Services :**

Linked services are much like connection strings, which define the connection information that’s needed for data factory to connect to external sources.  
Eg: A company uses **Azure Blob Storage** to store raw customer data and wants to copy it into an **Amazon S3 bucket** for collaboration with external partners.  
**Azure Blob Storage --> Amazon S3 bucket**



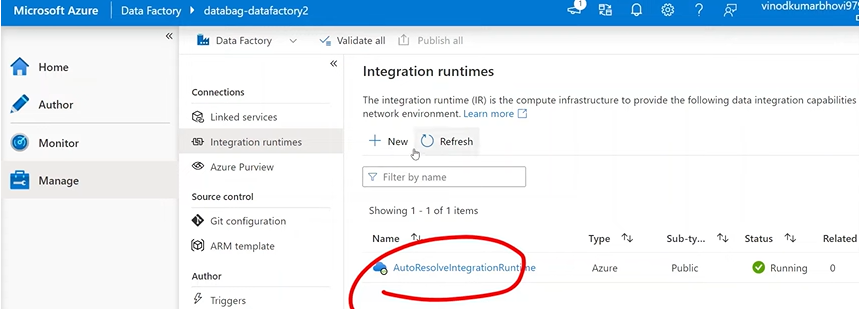
**Why Linked Services :**

In Azure Data Factory, **Linked Services** are used to define the connection information for both the source and destination data stores or services. They act as connection strings or bridges that enable Azure Data Factory to connect and interact with these resources.

**Integration Runtime :**

In data factory, an activity defines the action to be performed. A linked service defines a target data store or a compute device. An integration runtime provides the bridge between the activity and linked services. Data Factory automatically creates a integration runtime we don’t want to create one. We have three types of Integration runtimes.

* Azure Integration runtime.
* Self-hosted Integration runtime.
* Azure-SQL Server Integrated Services (SSIS) Integration runtime.



**Why Integration runtime** : In **Azure Data Factory**, **Integration Runtime (IR)** is the compute infrastructure used to perform data movement, transformation, and activity execution. It acts as a bridge between Azure Data Factory and the connected resources.

Azure Integration runtime : The compute resource for an azure integration runtime is fully managed elastically in Azure.  
Why : This is a fully managed compute environment in Azure. It is used for **data movement** and **data transformation** activities within the cloud.

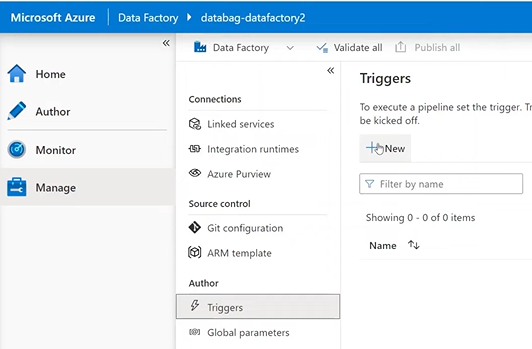
Self-hosted Integration runtime : Self-hosted Integration runtime can run copy activities between a cloud data store and a data store in a private network.  
Why : This is used to connect on-premises data sources or private networks to Azure Data Factory.

Azure-SQL Server Integratedservices (SSIS) Integration runtime : Azure SSIS IR is a fully managed cluster of Azure Virtual Machines (VMS or nodes) dedicated to run SSIS package.  
Why : This runtime is specifically used to **lift and shift SQL Server Integration Services (SSIS) packages** into Azure Data Factory.

**Triggers :**

Triggers represent the unit of processing that determines when a pipeline execution needed to be kicked off. There are different types of triggers for different types of events.

* Schedule trigger - Invoke a pipeline on a wall-clock Schedule.
* Tumbling Window trigger- Operates on a periodic interval at retaining state also.
* Event-based trigger- A trigger that responds to an event.
* Custom trigger - Activates a pipeline based on events triggered by a web application.



**Core Components**

**Data Source :**

**Data sources** are the origin points where data resides. These could be cloud-based or on-premises systems that you want to access and integrate within Azure Data Factory (ADF).

**Data Movement :**

**Data movement** refers to the process of transferring data from one location to another within or outside Azure, using various data movement activities and tools.

**Data Transformation :**

Once data is moved, it often needs to be transformed before it can be used for analysis, reporting, or storage. **Data transformation** refers to the process of changing the format, structure, or values of data to make it suitable for downstream tasks.

**Monitoring :**

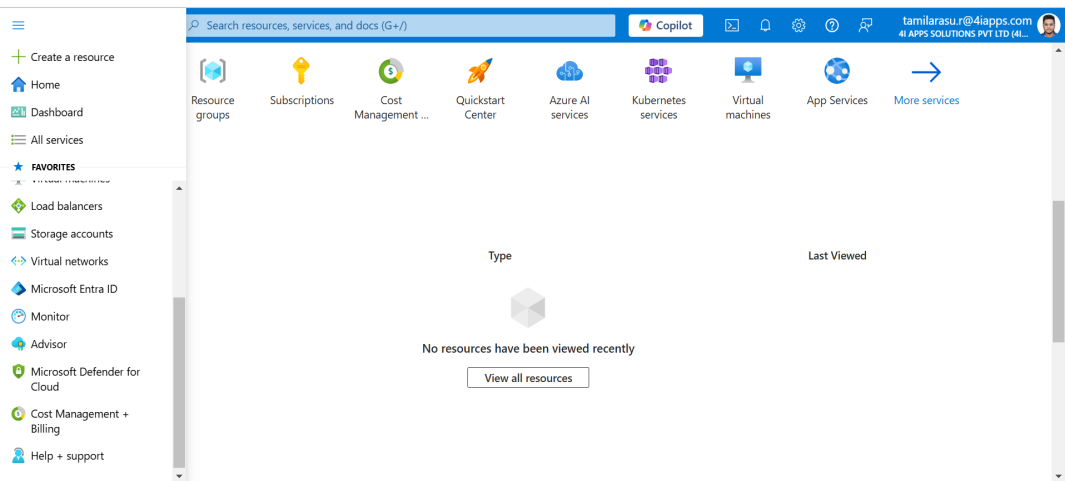
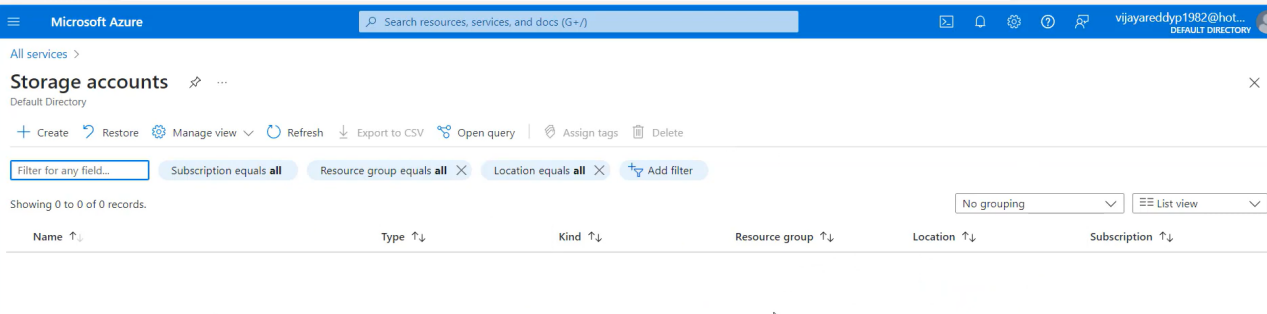
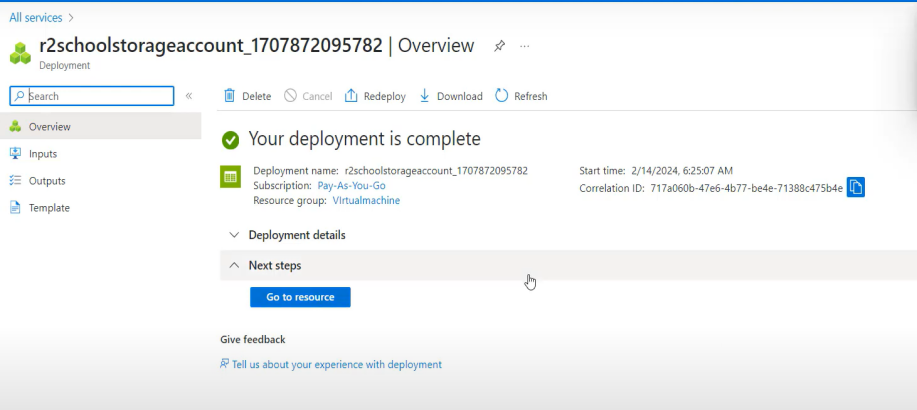
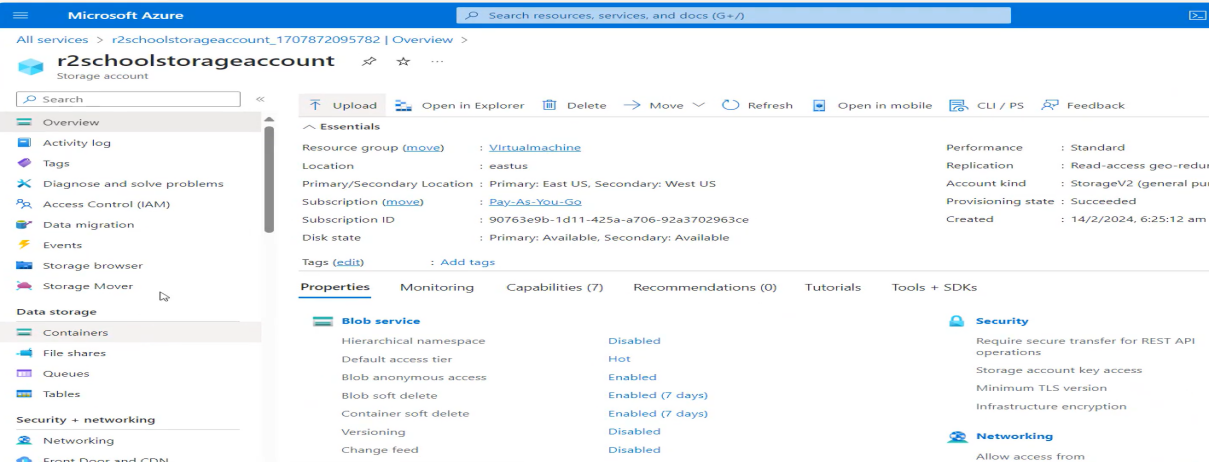
**Monitoring** provides visibility into the health and status of your data workflows. It helps you track the execution of pipelines, identify errors, and ensure that everything is running smoothly.

**Prerequisites for Project flow**

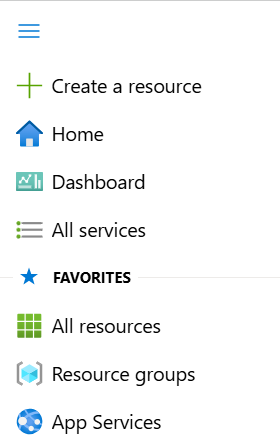
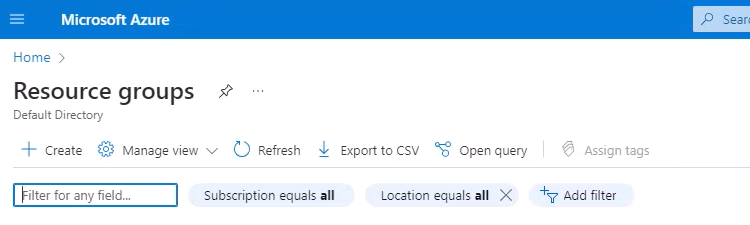
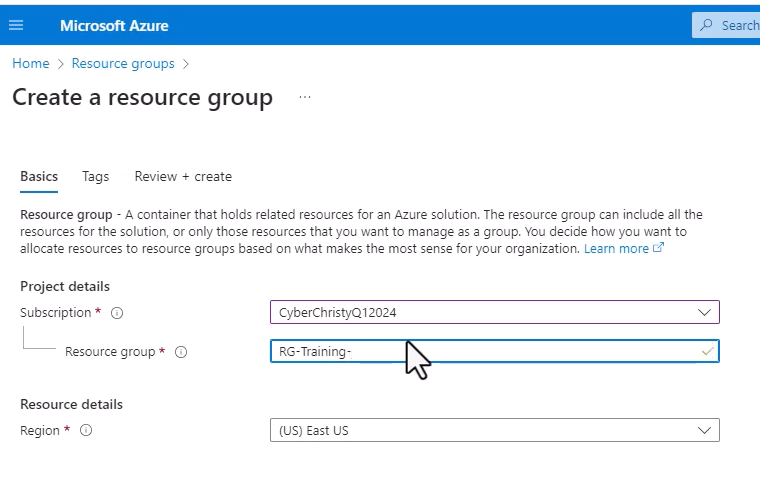
Before starting the project we need some prerequisites.

\* Creating Azure Blob Storage Account.  
\* Creating Resource group.

**Creating Storage Account :**

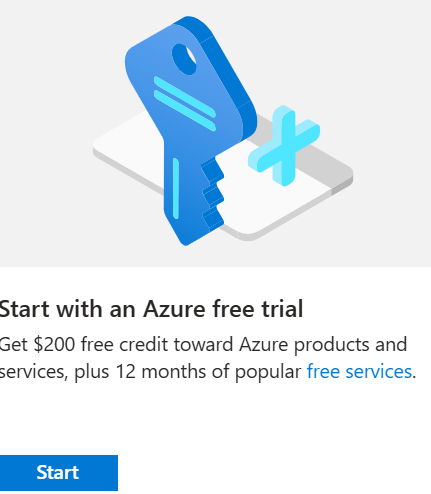
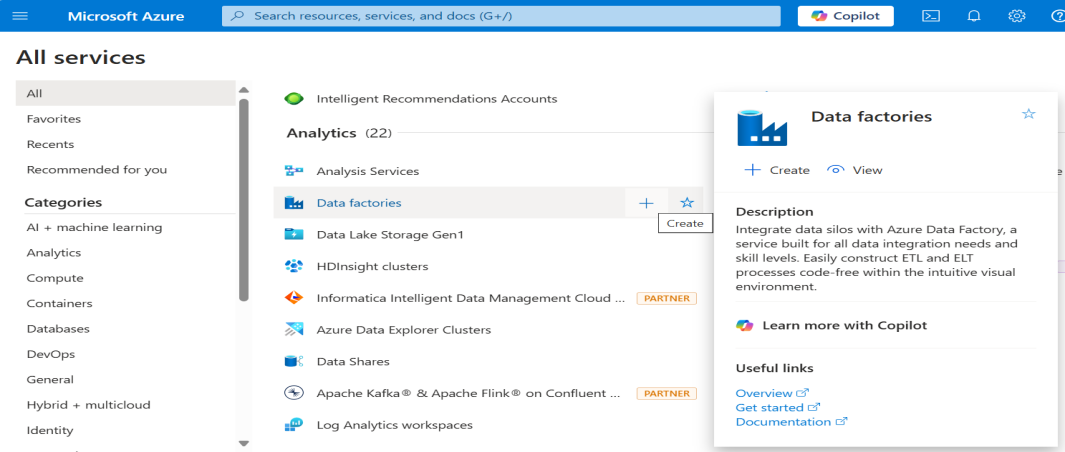
* Login to the azure portal and click the left corner button in that select storage account.  
    
  
* Click **Create** to Start creating Storage Account.  
    
  
* We need to fill the basic details to create the storage account for Eg: Subscription name, Resource group, Storage account name, Region etc..,.For Resource group we can able to create a new resource group also and if we already have a resource group we can create storage account in that group.  
  
* After review select **Create** the storage account then the overview is like  
    
  
* Click **Go to resource** it will take to Resource which we gave at the time of creating storage account either a new one or existing one.  
    
  
* In a storage account, before uploading any files, we need to create a container. For example, if we want to upload three or four images, we can create a container named images and store all the images in it. Similarly, if we are uploading more than five Excel files, we can create a container named Excel files and upload all the files into that container.

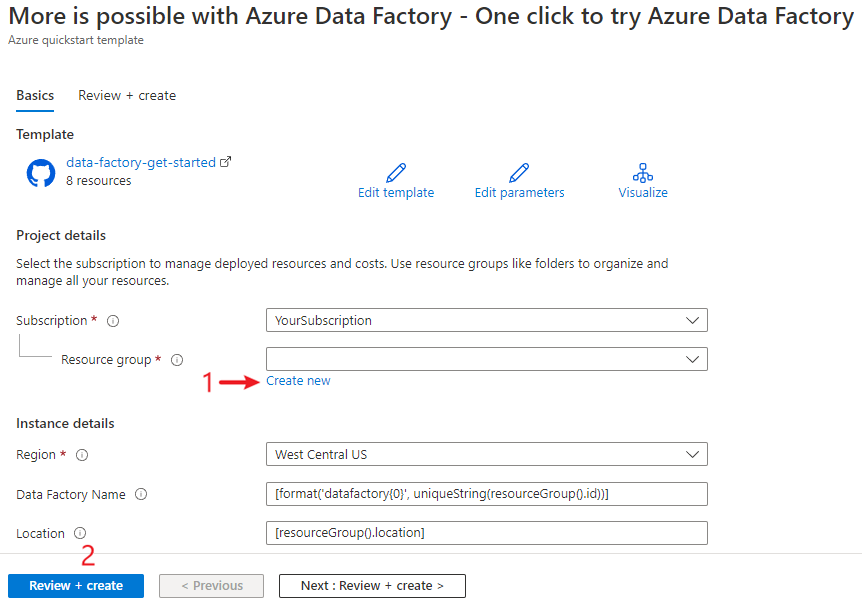
**Creating Resource Group :**

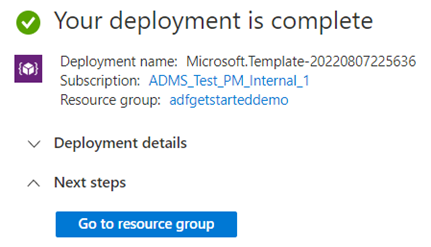
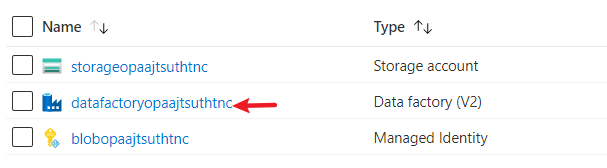
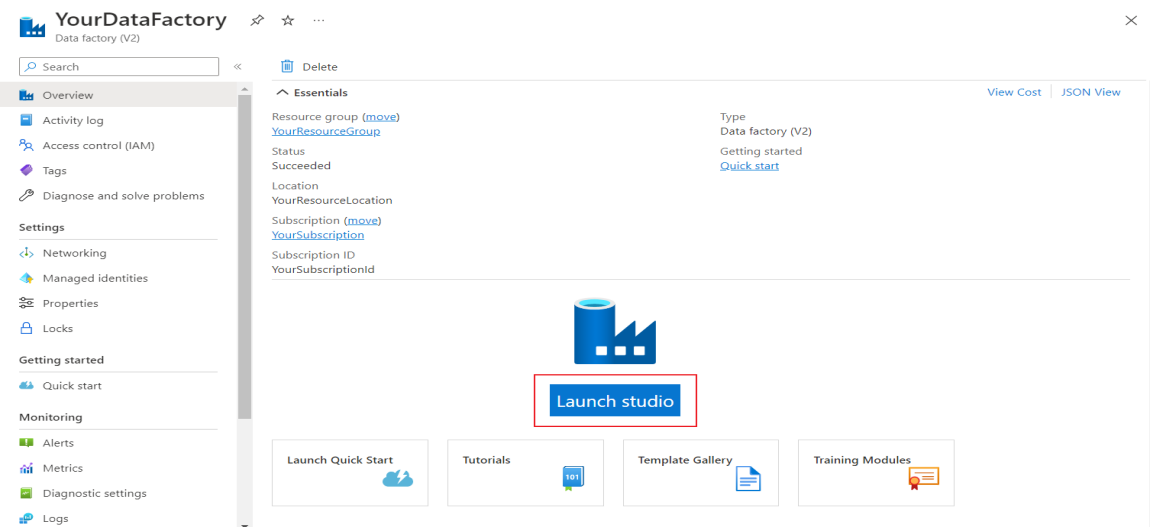
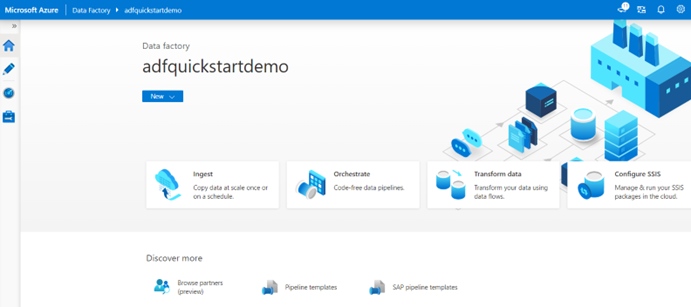
* Login to the Azure portal click left corner icon and select the **Resource group** to create.  
    
  
* After selecting resource group it appears one page to create it.  
    
  
* Click **Create** and fill in the details, such as the subscription name and a new resource group name. Then, click the **Review + Create** option. This is typically useful for storing resources like a storage account, Azure Data Factory (ADF), etc.  
    
  

**Project Flow**

**Get started with azure data factory :**

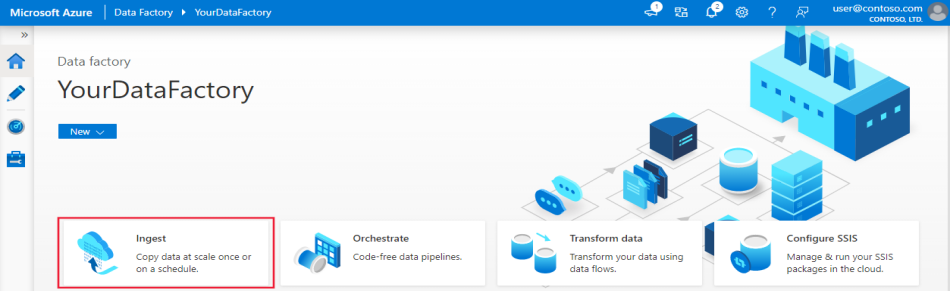
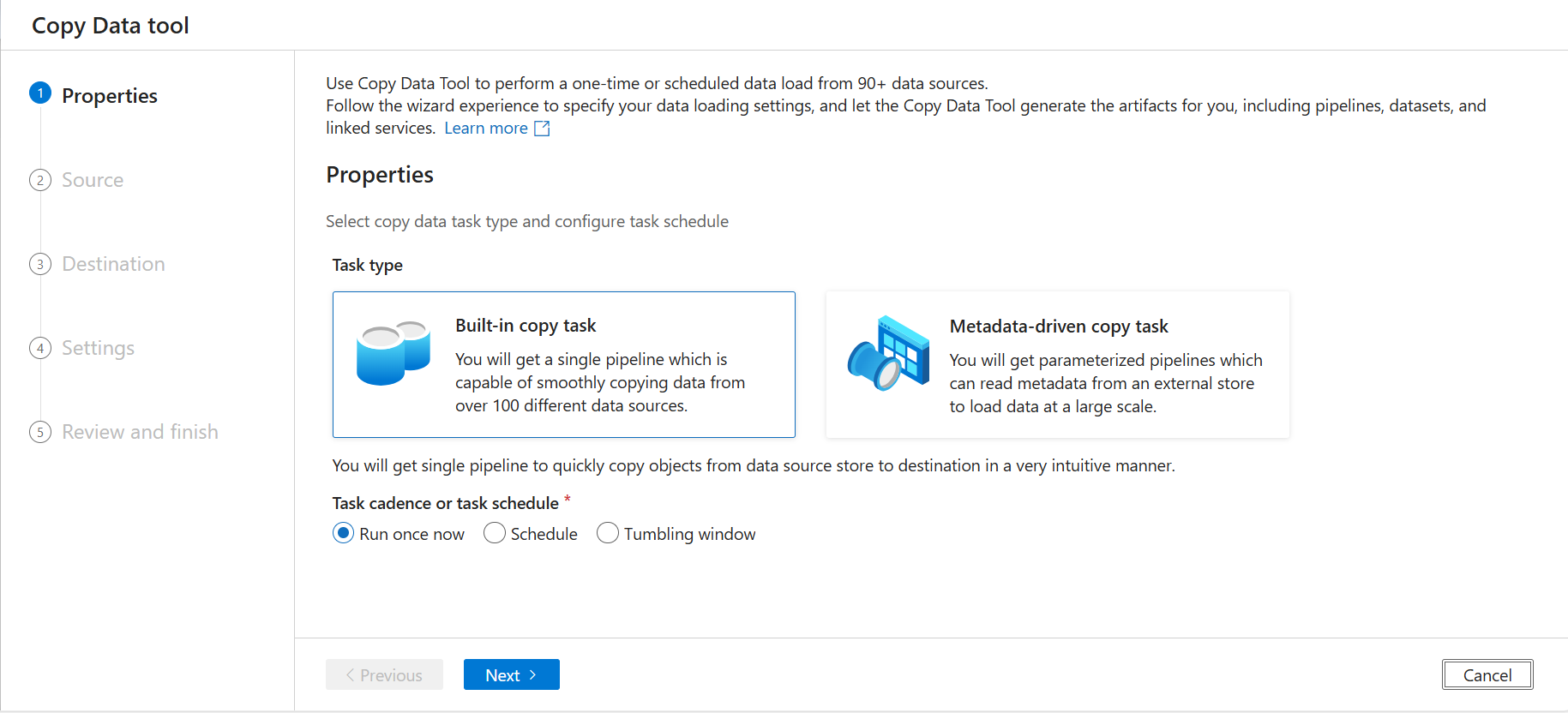
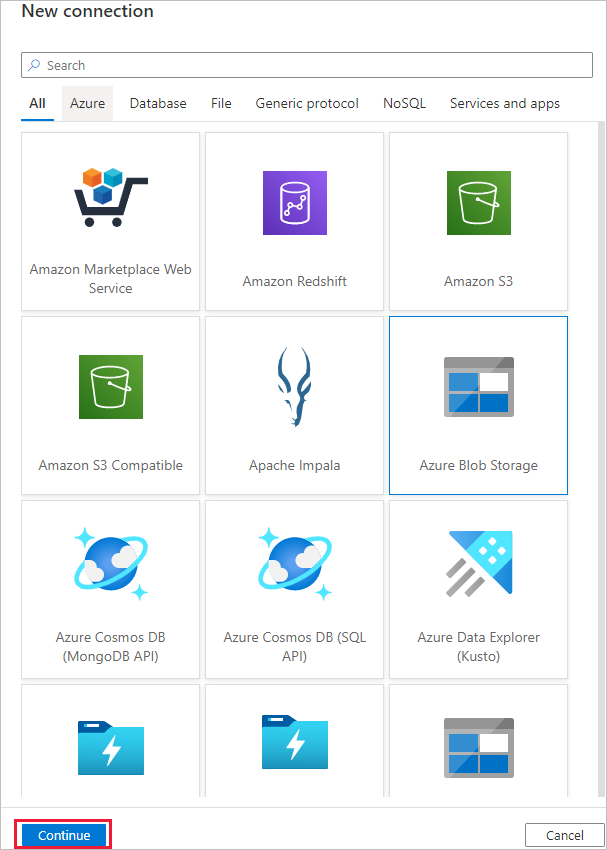
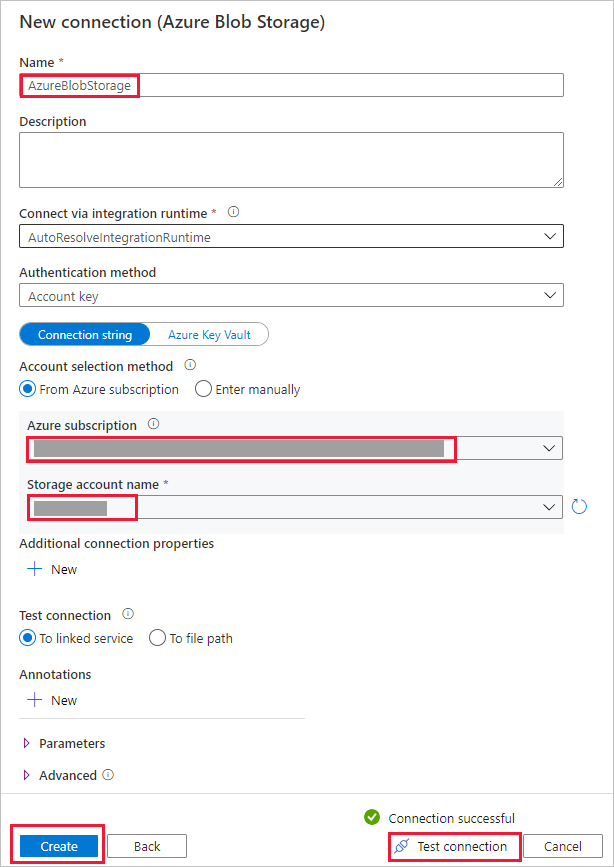
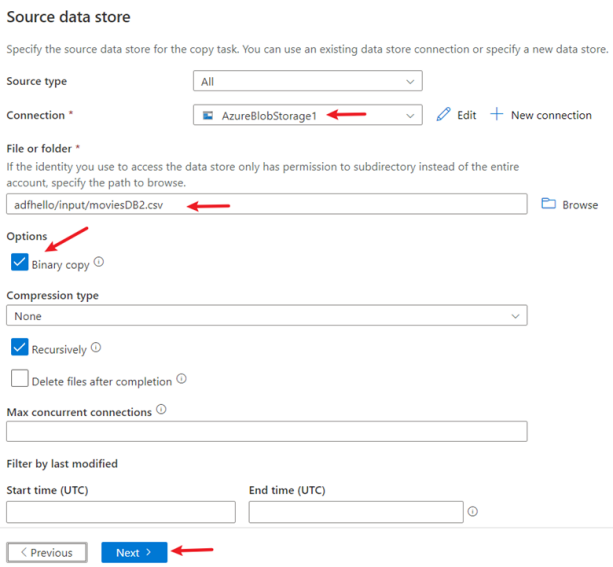
* Before we begin a **free subscription** for Azure is needed.  
    
  
* Search for Azure data factory and fill details of creating resource group and started working on ADF.  
    
  
* It will redirect to the configuration page and there we need to create a new resource group. After filling the details then click **Review + Create** and click **Create** to deploy the resources.

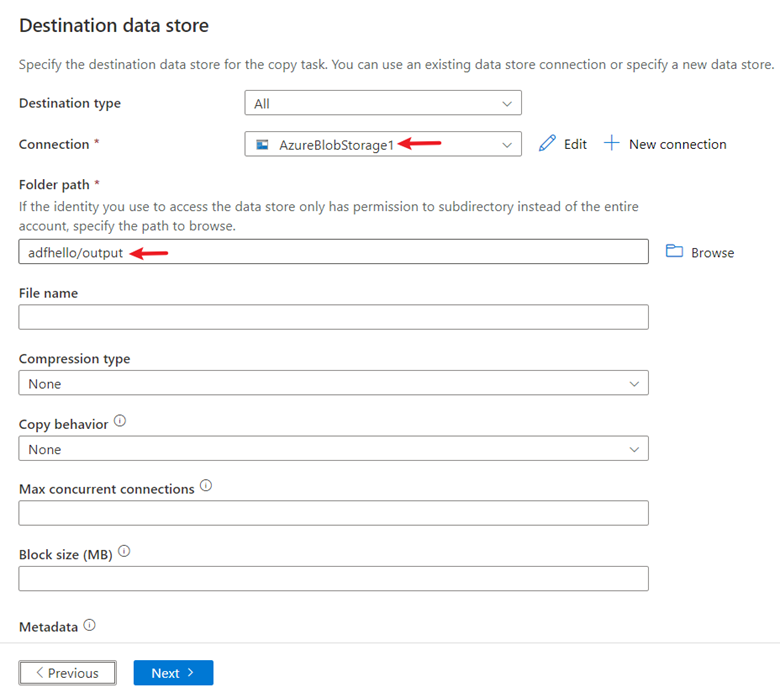


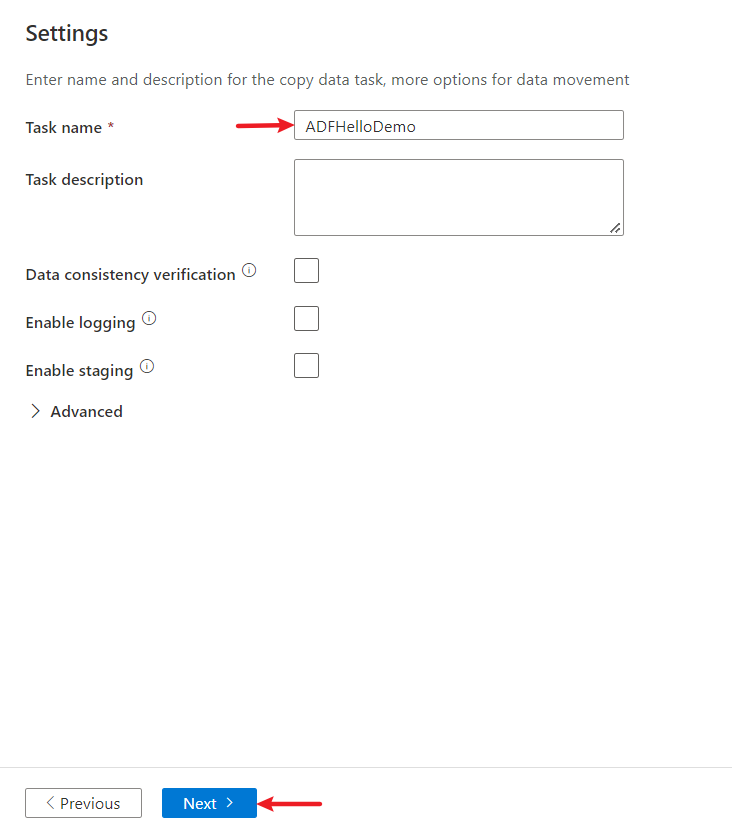
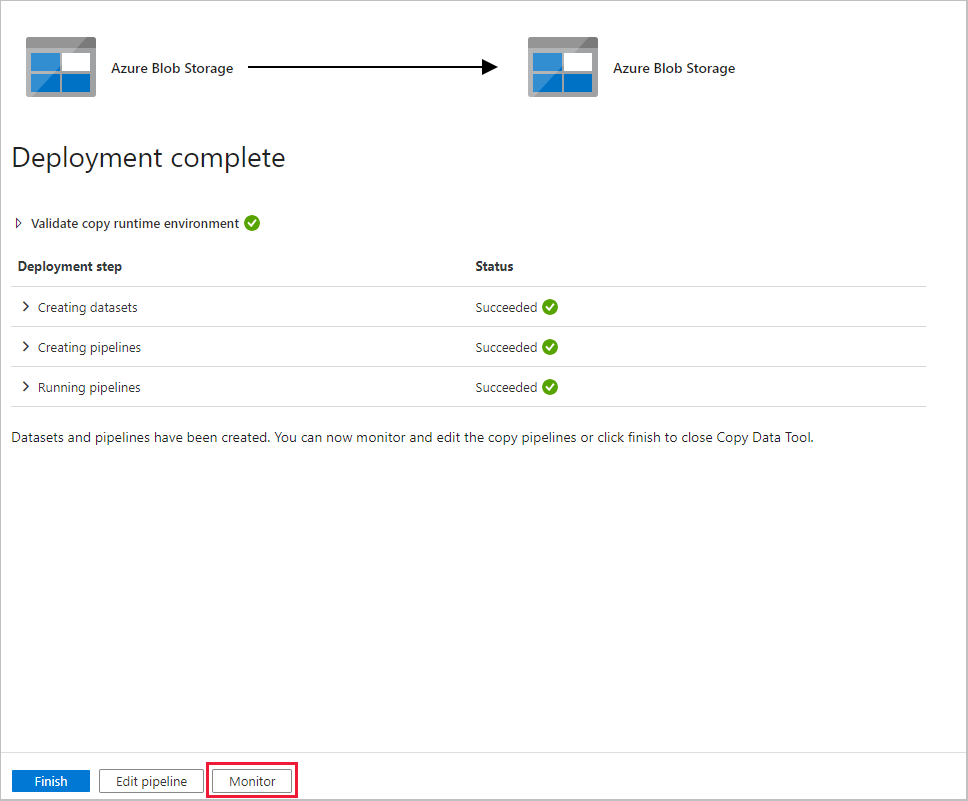
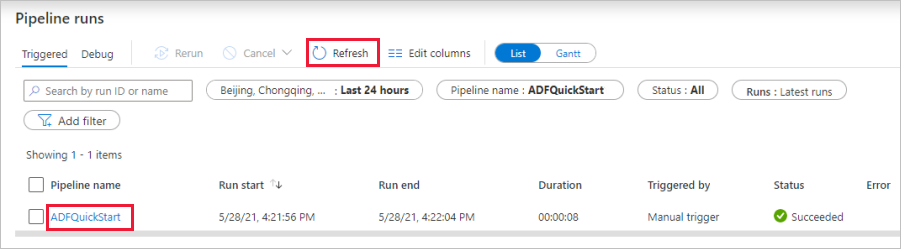
* Select **Go to resource group** after completing the deployment.  
    
  
* In the resource group we can see the new data factory that we created at the time of deployment.  
    
  
* Select the data factory in the resource group and click launch studio button to get started.  
    
  
* After launching the Azure data factory studio the home page of ADF appears.  
    
  

**Using Copy data tool to copy data :**

In Azure Data Factory, there are two ways to copy data. One way is by directly using the **Ingest** tile to copy data, and the other way is by creating a pipeline in Azure Data Factory Studio to perform the data copy.

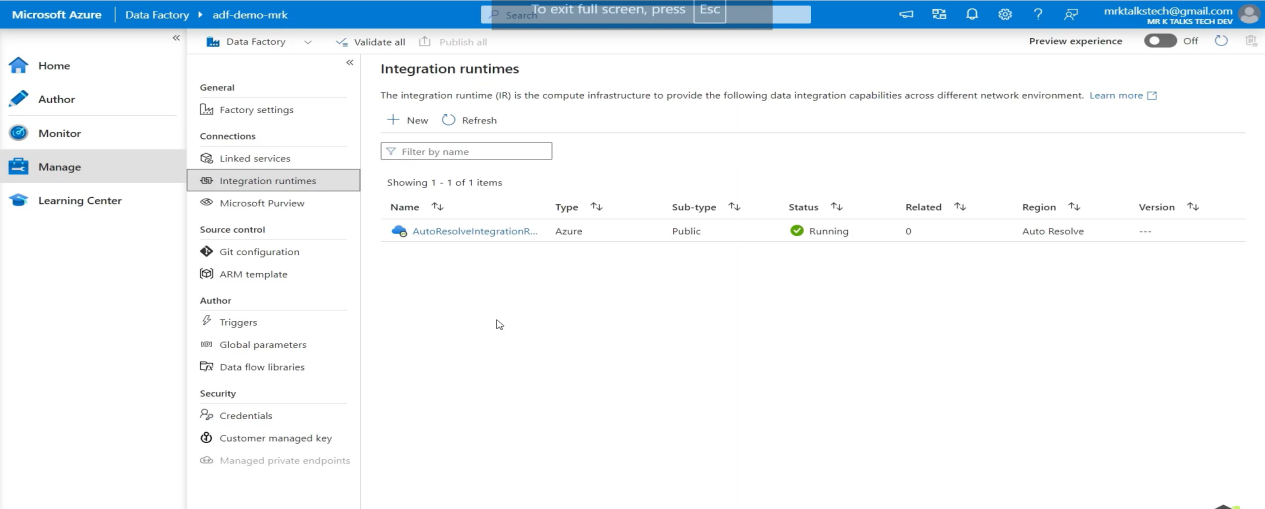
* On the home page of Azure data factory select the **Ingest** tile to start the copy data tool.  
    
  
* On the **Properties** page of the copy data tool, Choose **Built-in copy task** under **Task type**, then select **Next**.  
    
  
* After that, we need to select a connection for the source. Click **+ Create new connection** to add a connection, and then select the linked service for the connection. For example, if I want to copy data from Blob Storage, I would select Azure Blob Storage as the linked service and specify the destination.Then select **Continue**.  
    
  
* On the **New Connection (Azure Blob Storage)** specify name for the Connection. Select the **Azure Subscription** and storage account name after filling the details **Test connection** if it is succeeded then select **Create**.  
    
  
* For the **Source** newly created connection we have to take and give the file path and Click **OK**. Then select the Binary Copy Checkbox to copy the file and select **Next**.  
    
  
* For the **Destination** select the Azure blob storage connection that we have created in Connection block there itself we are giving destination but in another folder path. In the folder path section select the destination path for copying data into it and select **Next**.



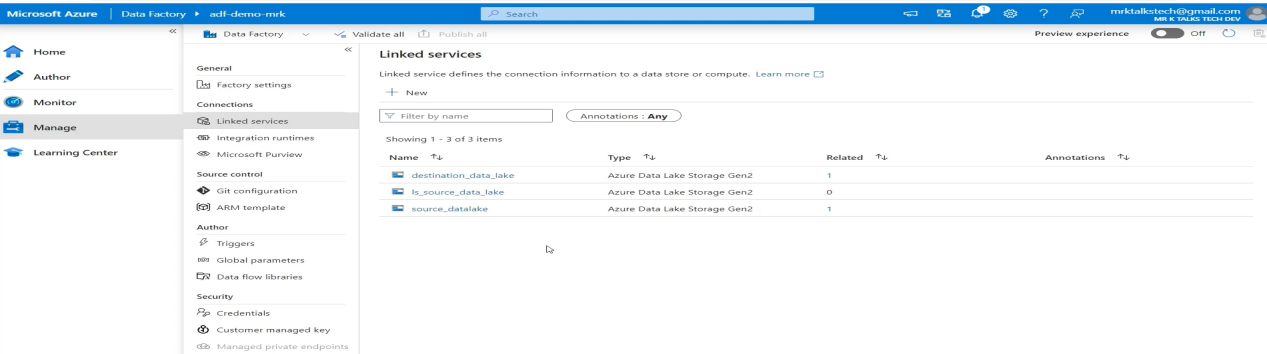
* On the **Settings** page, specify a name for the pipeline and its description, the select **Next**.  
    
  
* On the Summary page, review all settings, then select Next.
* On the deployment Complete page select Monitor to monitor the pipeline that we created.  
    
  
* It will switch to the **Monitor** tab there we can see the status of the pipeline. Then select **Refresh** and select the name of the pipeline to see the activity run details.  
    
  

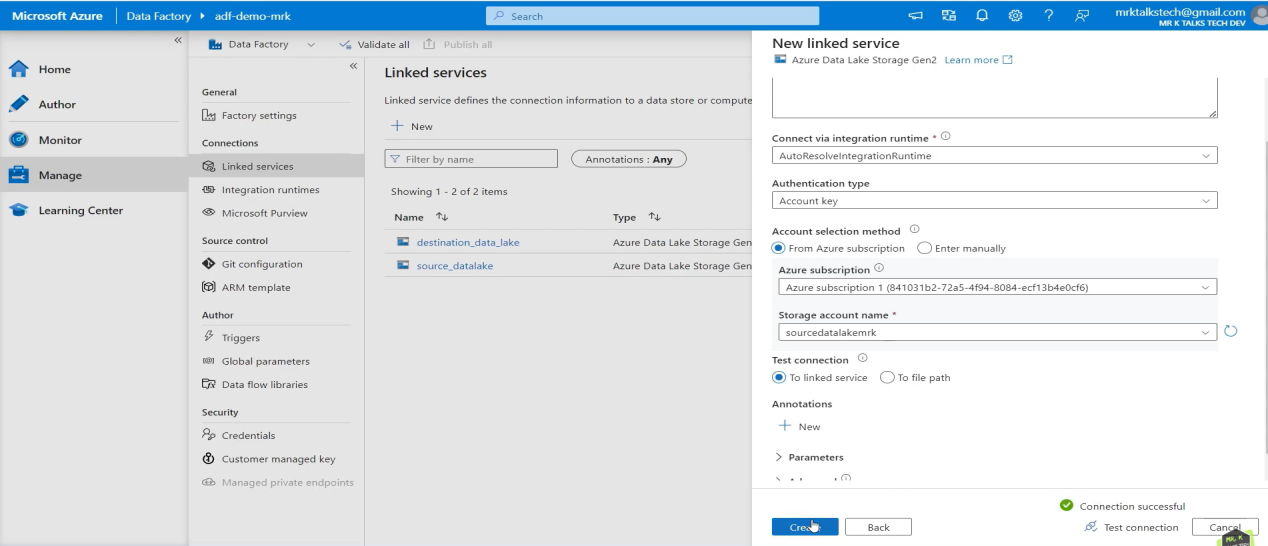
Copying data by using Pipeline :

* To copy data from source to destination through the pipeline we need to add integration runtime, linked services and datasets.

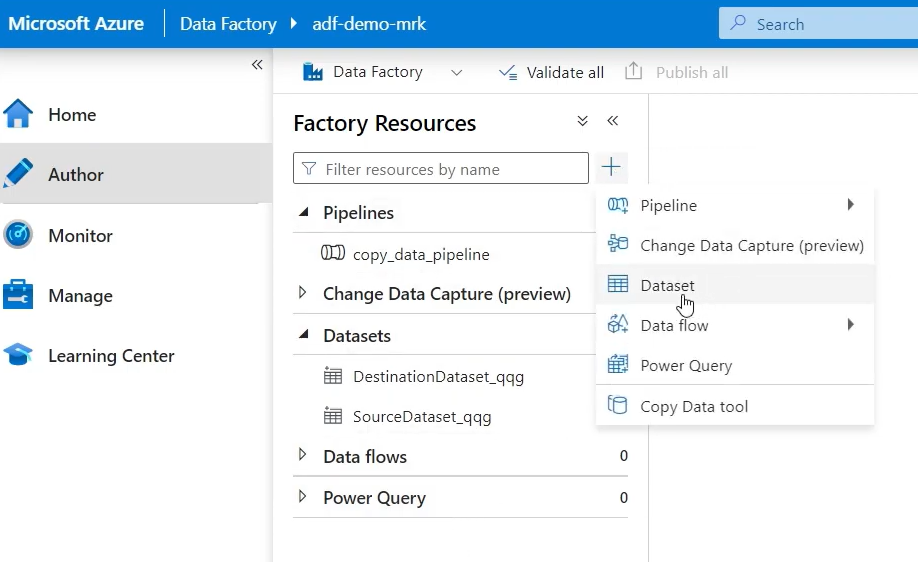
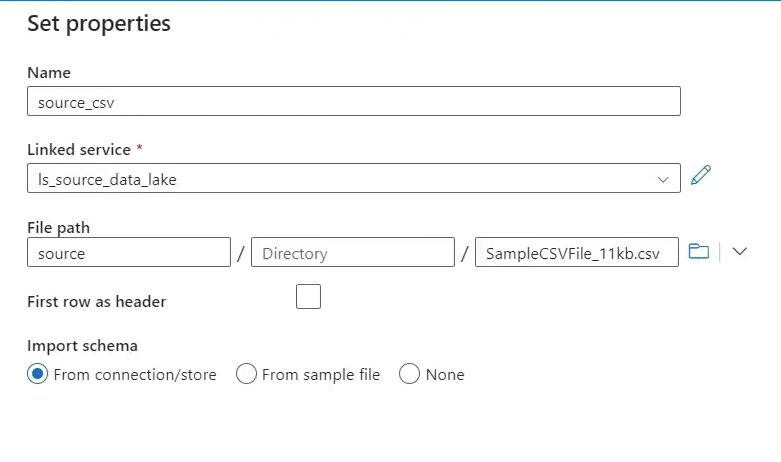
**Step 1 :**First, we need to check the **Integration Runtimes** for the dataset copying process. Azure provides a default Integration Runtime, but if needed, we can create other runtimes, such as for SSIS or for a single user. Integration Runtimes in ADF are customizable based on the requirements.  
  
****

**Step 2 :**

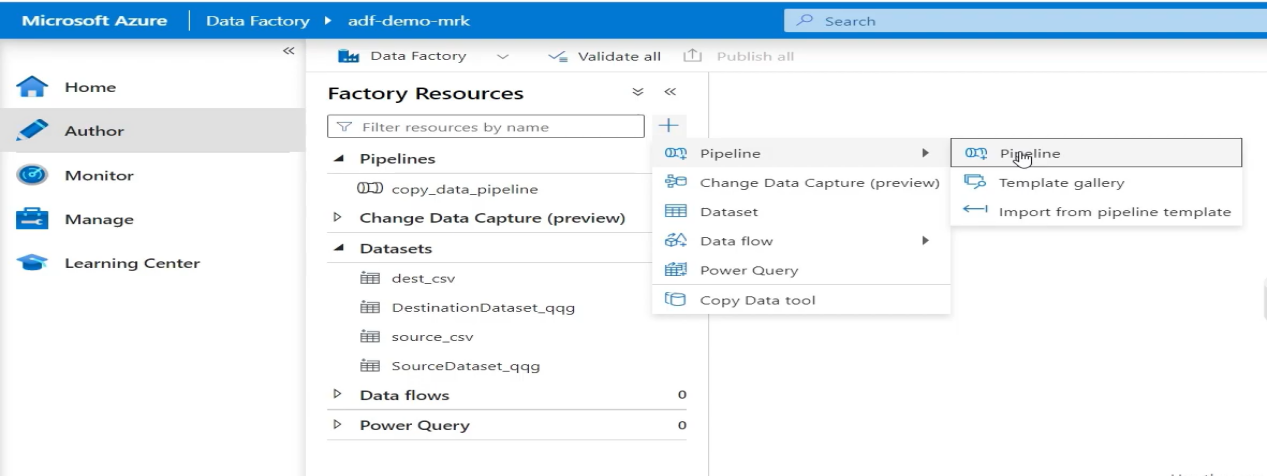
Next, we need to link the services. To do this, go to the **Manage** tab and select **Linked Services**. Then, configure the source and destination linked services. For example, if I have two connections in a storage account—one as the source and the other as the destination—I would connect both resources in the linked services.Then select **Create**.  
  
  
Select the **New** option in Linked Services and fill in the details, such as the storage account name where the source file is located, the integration runtime, subscription details, etc. Similarly, create a linked service for the destination as well.



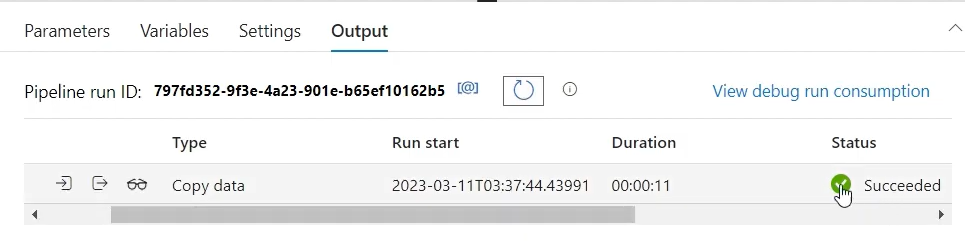
**Step 3 :**

Next, we need to create datasets for the source and destination paths. To do this, go to the **Author** tab and select **Dataset**. Choose the Blob Storage option and specify the file type, such as CSV or Excel, for the source dataset. Similarly, create a dataset for the destination by specifying the path and the desired file type. For example, if I’m copying data from a CSV file and want the destination data in Excel format, I would configure the dataset accordingly.   
  
In **Set properties** we need to give linked services that we have created for source and destination previously and then click **OK**.  


**Step 4 :**

Lastly, we create the pipeline to copy data from one location to another. To do this, go to the **Author** tab and select **Pipeline**.  


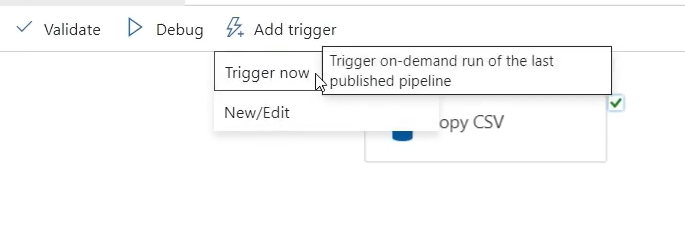
There, we need to select the source and destination paths that were defined in the datasets previously.  


After that, we have two options to run the pipeline: **Debug** and **Trigger**. **Debug** is a **Manual method** where as **Trigger** is a **Automatic method**.Now, let’s see how to run the pipeline using the **Debug** option.  
  
  
  
After successfully running the pipeline, we need to check the Blob Storage path to verify whether the file was copied or not.

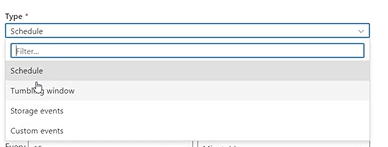
**Scheduling and Triggering**A triggering in ADF is a mechanism that starts an ADF Pipeline automatically at a specified time or when a specific event occurs.There are different types of triggers are available in ADF.

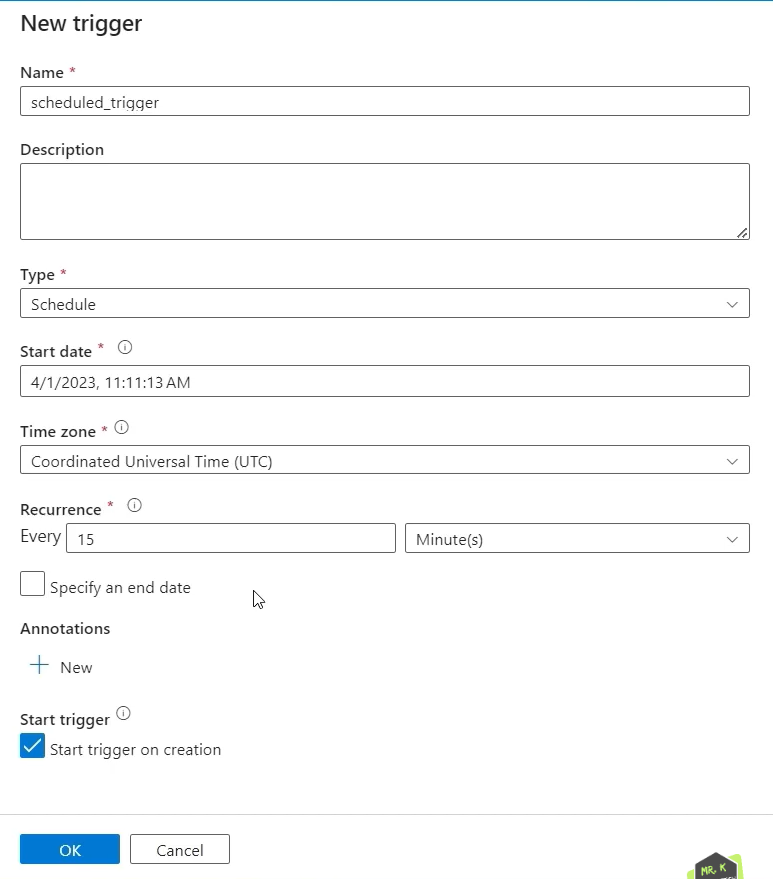
1) Schedule Trigger

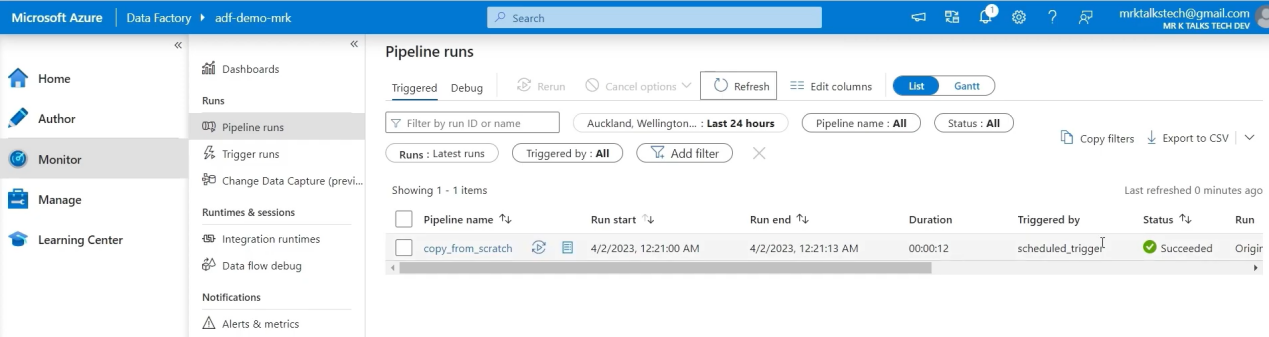
1. Storage Events Trigger
2. Tumbling Window Trigger
3. Custom Triggers

Schedule Trigger : This trigger runs a pipeline on a specific schedule, such as hourly,weekly, daily and monthly.Schedule triggers can be created and configured using the ADF portal.  
  


Previously, we ran the pipeline using the **Debug** option. Now, we’ll run the pipeline using the **Trigger** option. After selecting **Trigger**, there are two choices: to run the pipeline immediately, click **Trigger.Now** to schedule the pipeline, select **New/Edit**.



As mentioned before, there are different types of triggers available in ADF. Select the trigger type based on the requirement, provide a name for the trigger, specify the time and frequency of execution, fill in all the necessary details, and then click **OK**.  
  


After triggering the pipeline, its status will be displayed in the **Monitor** tab. To view only triggered pipelines, go to the **Monitor** tab and select **Trigger Runs**. If you want to delete triggers, navigate to the **Manage** tab, select **Triggers**, where you can see the triggered pipelines, and delete them as needed.  
  


Schedule Trigger : Runs pipelines on a pre-defined schedule.  
Why : For time-based pipeline exceutions.

Storage Events Trigger : Starts pipelines based on events in Azure Blob Storage.  
Why : Runs pipelines at fixed time intervals with retry support.  
  
Tumbling Window Trigger : Processes pipelines in fixed time intervals without overlaps.  
Why : Runs pipelines when an event happens, like a file arrival.  
Custom Trigger : Trigger pipelines based on external events or custom conditions via APIs or integrations.  
Why : Runs pipelines based on custom logic or external control.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trigger** | **Usecase** | **Execution Frequency** | **Event based** | **State Retention** |
| **Schedule Trigger** | Executes on a predefined schedule (e.g., daily, weekly). | Recurring at set times (e.g., 9 AM daily). | No | No |
| **Storage Event Trigger** | Triggered by events like file creation or modification in a storage account. | When specific events occur (e.g., new file upload). | Yes | No |
| **Tumbling Window Trigger** | Executes in fixed, non-overlapping intervals for time-based processing | Periodic with defined window sizes (e.g., hourly). | No | Yes(Tracks state per window) |
| **Custom Trigger** | Invoked by external, user-defined conditions. | Custom defined by user (e.g., event from external system). | Yes | Depends on the custom logic. |

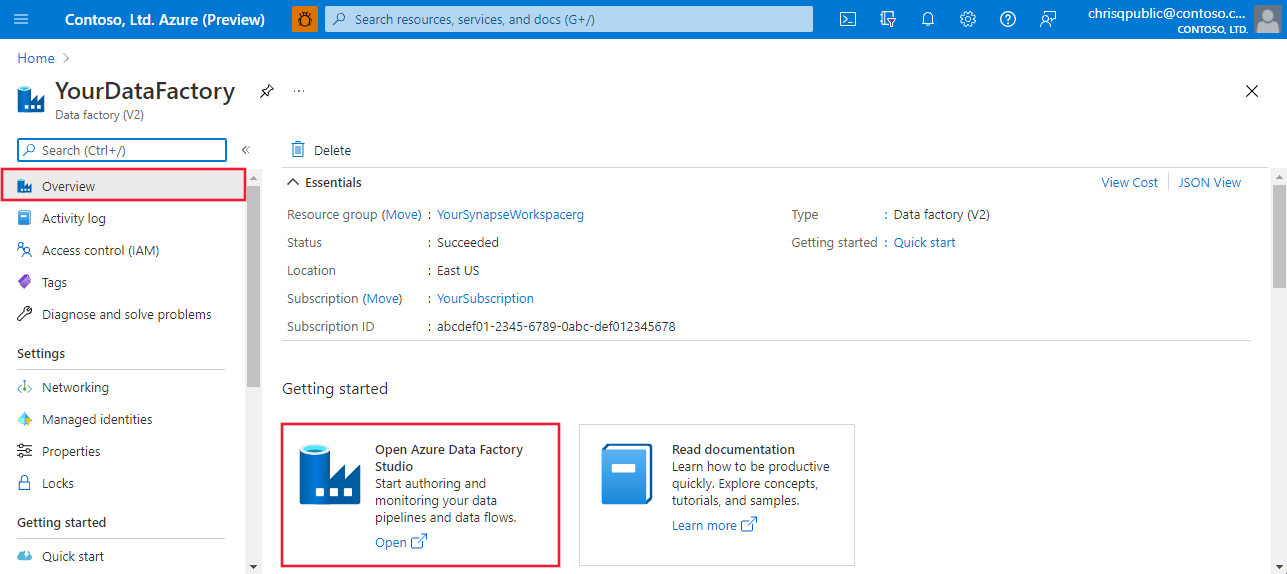
**Data Flow** :   
  
A **Data Flow** in Azure Data Factory is a **visual tool** used to design and perform **data transformations** without writing code. It allows you to move and transform data at scale within the cloud.Designed for data transformation and ETL/ELT processes.  
  
**Why** : When you need to **clean, prepare, or transform data** before moving it to a destination, such as for reporting or analysis.In short, Data Flow simplifies complex **ETL (Extract, Transform, Load)** processes visually and efficiently.

### **Features of Data Flow :**

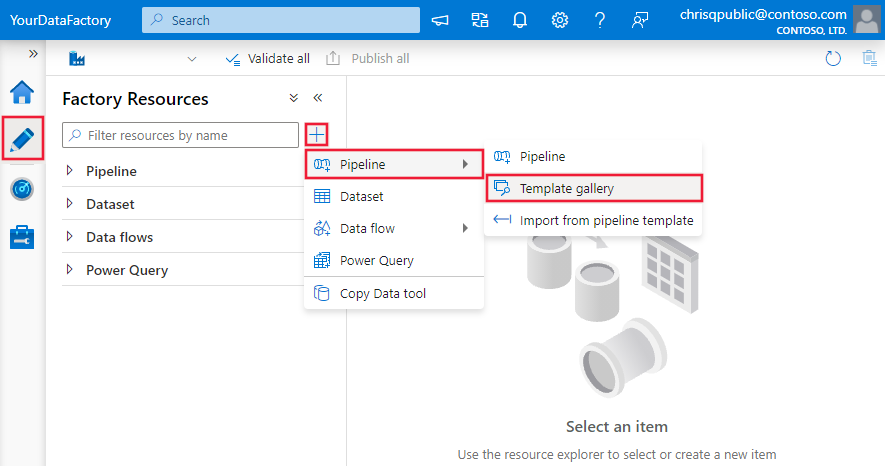
* **Visual Design** : Drag-and-drop interface to build transformation logic.
* **Transformation** : Allows operations like filter, join, aggregate, sort, and more.
* **Code-Free** : No need to write code—logic is built visually.
* **Scalable** : Runs on Azure's **Spark clusters** to handle large data volumes.

**Creating Data Flow :**

Get started by first [creating a new V2 Data Factory](https://learn.microsoft.com/en-us/azure/data-factory/quickstart-create-data-factory-portal) from the Azure portal. After creating your new factory, select the **Open **Azure Data Factory Studio**** tile in the portal to launch the Data Factory Studio.

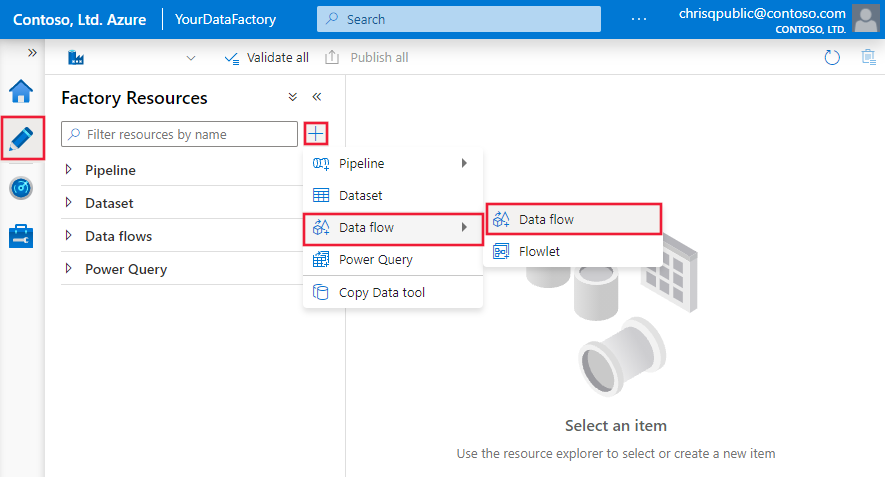


You can add sample Data Flows from the template gallery. To browse the gallery, select the **Author** tab in Data Factory Studio and click the plus sign to choose ****Pipeline** | **Template Gallery****.

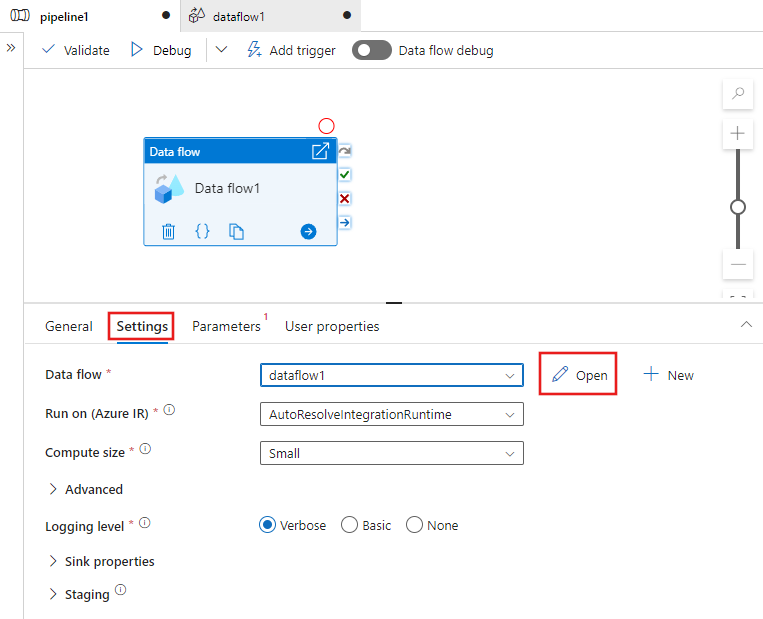


Select the Data Flow category there to choose from the available templates. 

You can also add data flows directly to your data factory without using a template. Select the ****Author**** tab in Data Factory Studio and click the plus sign to choose ****Data Flow**** | ****Data Flow****.



In the ****Data Flow**** pop-up, select ****Create new Data Flow**** and then name your data flow . Click Finish when done.

  
In the top bar of the pipeline canvas, slide the ****Data Flow debug**** slider on.  
IMG_256  
Once you create your Data Flow, you'll be automatically sent to the data flow canvas. In case you are not redirected to data flow canvas, in the panel below the canvas, go to ****Settings**** and select ****Open****, located beside data flow field. This will open the dataflow canvas.  


**Control Flow** :

A **Control Flow** in Azure Data Factory defines the **execution order** and **logic** for activities within a pipeline. It orchestrates how different tasks run and interact with each other.Focuses on managing and sequencing activities in a pipeline.

**Why :**

When you need to **orchestrate tasks**, manage their sequence, and include conditions or loops for workflows.In short, Control Flow manages the **workflow logic** of activities within a pipeline.

### **Features of Control Flow:**

* **Activity Sequencing**: Determines the order in which activities execute.
* **Conditional Logic**: Allows "if-else" conditions, loops, and error handling.
* **Branching**: Supports running multiple activities in parallel.
* **Trigger Execution**: Controls when and how pipelines start.

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Data Flow** | **Control Flow** |
| Purpose | Handles **data transformations** and processing. | Manages the **coordination** and execution of tasks. |
| Focus | Focuses on moving and transforming data within a pipeline. | Focuses on managing the order and flow of activities in a pipeline |
| Componenets | Uses **Data Flows** (Mapping Data Flow, Wrangling Data Flow). | Uses activities like **Copy, ForEach, Execute Pipeline, and Web**. |
| Data Transformations | Performs complex **ETL/ELT transformations** like joins, aggregations, and lookups. | Does not handle direct transformations; focuses on controlling and sequencing activities. |
| Execution | Runs within the **data flow runtime** using compute (Azure IR, SHIR). | Executes on the **pipeline runtime** to manage tasks and dependencies. |
| Usecase | Ideal for transforming raw data into actionable insights. | Ideal for controlling the flow of tasks and managing workflows. |
| Performance | Optimized for **data-intensive operations** with high scalability. | Optimized for task coordination and dependency management. |

In Control Flow, activities are the building blocks used to define actions within a pipeline. Pipelines in tools like Azure Data Factory or SSIS orchestrate the execution of these activities. Each pipeline contains a sequence of activities that define the control flow, allowing you to manage and monitor the execution of processes such as data movement, data transformation, or custom operations. You can view and manage the control flow of these activities within the pipeline designer interface.

Under Control flow the activities include :

\* Execute Pipeline

\* For each loop

\* If condition

\* wait

\* Set variable

\* Append variable

\* Filter

\* Switch

\* Lookup

\* Web

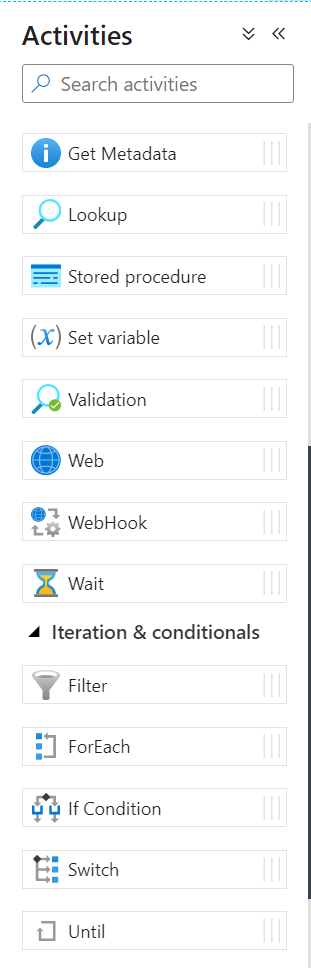
\* Execute SSIS Package

\* Get Metadata

\* Validation

\* Until

\* Stored Procedure



* **Get Metadata Activity :**

The Get Metadata activity retrieves metadata properties of a file, folder, or dataset.

**Steps to Configure :**

1. Drag the **Get Metadata** activity into the pipeline.
2. Specify the dataset or file for which metadata is needed.
3. Select the required metadata fields (e.g., size, last modified).

**When to Use Get Metadata Activity :**  
Use this activity to check properties like file existence, size, or modification date before further processing.

* **Exeute Pipeline :** The Execute Pipeline activity allows you to run another pipeline within a pipeline.

**Steps to Configure :**

1. Drag and drop the ****Execute Pipeline**** activity into the pipeline canvas.
2. Go to the **Settings** tab and select the pipeline you want to execute.
3. Optionally, define parameters to pass values to the child pipeline.

**When to Use Execute Pipeline Activity :**  
Use this activity when you want to modularize your workflows by dividing tasks into separate pipelines and execute them as needed. This is especially helpful for reusability and managing large workflows.

* **For Each Activity :**

The For Each activity iterates over a collection of items and executes a set of activities for each item.

**Steps to Configure :**

1. Drag the **For Each** activity into the pipeline.
2. In the **Settings** tab, specify the collection (array or list) to loop through.
3. Inside the activity, define the actions or sub-activities to execute for each item.
4. Configure concurrency if needed (default is sequential execution).

**When to Use For Each Activity :**  
Use this activity when you need to perform repetitive actions, such as processing files from a folder or performing operations on a list of values.

* **If Condition Activity :**

The If Condition activity evaluates a boolean expression and executes different sets of activities based on the result (True/False).

**Steps to Configure :**

1. Drag the **If Condition** activity into the pipeline.
2. Define the condition expression in the **Activities** tab (e.g., @greater (variables('value'), 10)).
3. Add activities under the **True** and **False** branches.

**When to Use If Condition Activity :**  
Use this activity for branching logic when you need to execute different actions based on a condition, such as checking file availability or validating input values.

* **Wait Activity :**

The Wait activity pauses pipeline execution for a specified duration.

**Steps to Configure :**

1. Drag the **Wait** activity into the pipeline.
2. Specify the wait time in seconds in the **Settings** tab.

**When to Use wait Activity :**  
Use this activity when you need to introduce a delay in your pipeline, such as waiting for a dependent system or process to complete.

* **Set Variable Activity :**

The Set Variable activity assigns a value to a pipeline variable.

**Steps to Configure :**

1. Drag the **Set Variable** activity into the pipeline.
2. Select the variable you want to set from the dropdown menu.
3. Define the value to assign using expressions or dynamic content.

**When to Use Set Variable Activity :**  
Use this activity to dynamically set or update variables during pipeline execution, such as assigning runtime values or results.

* **Append Variable Activity :**

The Append Variable activity adds a value to an existing array-type variable.

**Steps to Configure :**

1. Drag the **Append Variable** activity into the pipeline.
2. Select the array variable you want to append to.
3. Specify the value to append.

**When to Use Append Variable Activity :**  
Use this activity when you need to build a collection of values during pipeline execution, such as gathering filenames or error messages.

* **Filter Activity :**

The Filter activity filters a collection based on a specified condition.

**Steps to Configure :**

1. Drag the **Filter** activity into the pipeline.
2. In the **Settings** tab, specify the input array and filter condition.
3. Use dynamic expressions to define the filter logic.

**When to Use Filter Activity :**  
Use this activity to create a subset of items from a collection based on specific criteria, such as filtering files by extension or date.

* **Switch Activity :**

The Switch activity evaluates a single expression and executes one branch of activities based on the value.

**Steps to Configure :**

1. Drag the **Switch** activity into the pipeline.
2. Define the expression to evaluate.
3. Configure cases and the corresponding activities for each case.
4. Optionally, add a **Default** branch for unmatched cases.

**When to Use Switch Activity :**  
Use this activity for decision-making scenarios where multiple branches are needed, such as routing data processing based on file type or input value.

* **Lookup Activity :**

The Lookup activity retrieves a single row or a set of rows from a data source.

**Steps to Configure :**

1. Drag the **Lookup** activity into the pipeline.
2. Select the data source and configure the query or dataset to retrieve data.
3. Optionally, enable the "First Row Only" setting if you need just one row.

**When to Use Lookup Activity :**  
Use this activity to retrieve configuration values, metadata, or any specific data from a database or file for use in subsequent pipeline activities.

* **Web Activity :**

The Web activity makes HTTP calls to a REST endpoint.

**Steps to Configure :**

1. Drag the **Web** activity into the pipeline.
2. Specify the endpoint URL and HTTP method (GET, POST, etc.).
3. Configure headers and body as needed.
4. Test the connection to ensure it works.

**When to Use Web Activity :**  
Use this activity to interact with external REST APIs, such as sending notifications, triggering external systems, or retrieving data.

* **Execute SSIS Package Activity :**

The Execute SSIS Package activity runs an SSIS package deployed in Azure-SSIS Integration Runtime.

**Steps to Configure :**

1. Drag the **Execute SSIS Package** activity into the pipeline.
2. Specify the SSIS package location, environment, and parameters.
3. Configure the execution runtime.

**When to Use Execute SSIS Package Activity :**  
Use this activity to leverage existing SSIS packages for ETL operations as part of your pipeline.

* **Validation Acitivity :**

The Validation activity ensures that a file or dataset exists before continuing.

**Steps to Configure :**

1. Drag the **Validation** activity into the pipeline.
2. Specify the dataset to validate.
3. Configure timeout and retry policies if needed.

**When to Use Validation Activity :**  
Use this activity to validate the availability of input data or files before executing dependent activities.

* **Until Activity :**

The Until activity repeats a set of activities until a specified condition is met.

**Steps to Configure :**

1. Drag the **Until** activity into the pipeline.
2. Define the condition to evaluate in the **Settings** tab.
3. Place the activities to repeat inside the Until block.

**When to Use Until Activity :**  
Use this activity for looping scenarios where the exit condition depends on dynamic values, such as waiting for a process to complete or reaching a threshold.

* **Stored Procedure Activity :**

The Stored Procedure activity executes a stored procedure in a database.

**Steps to Configure :**

1. Drag the **Stored Procedure** activity into the pipeline.
2. Specify the database connection and stored procedure name.
3. Provide input parameters if required.

**When to Use Stored Procedure Activity :**  
Use this activity to call database procedures for operations like data transformation, auditing, or custom processing.

**Variables**

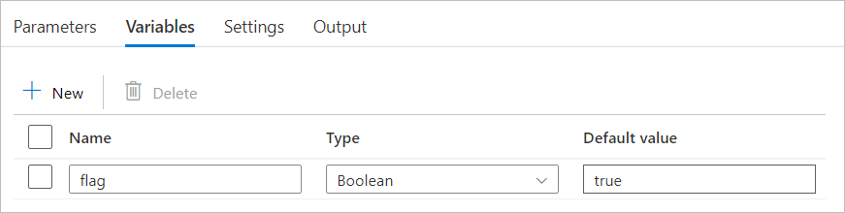
Variables in Azure Data Factory are used to store and update values during pipeline execution. They hold temporary data that can change as the pipeline runs.

**Why We Use Variables :**

* To store intermediate results during pipeline execution.
* To pass values between activities within the same pipeline.
* To enable dynamic decision-making based on runtime data.

**When to Use Variables :**

* When you need to dynamically update values during pipeline execution, such as tracking a counter or storing status information.
* For scenarios where values need to change based on activity outputs (e.g., appending file names to a list or updating a flag based on a condition).



**Parameters :**

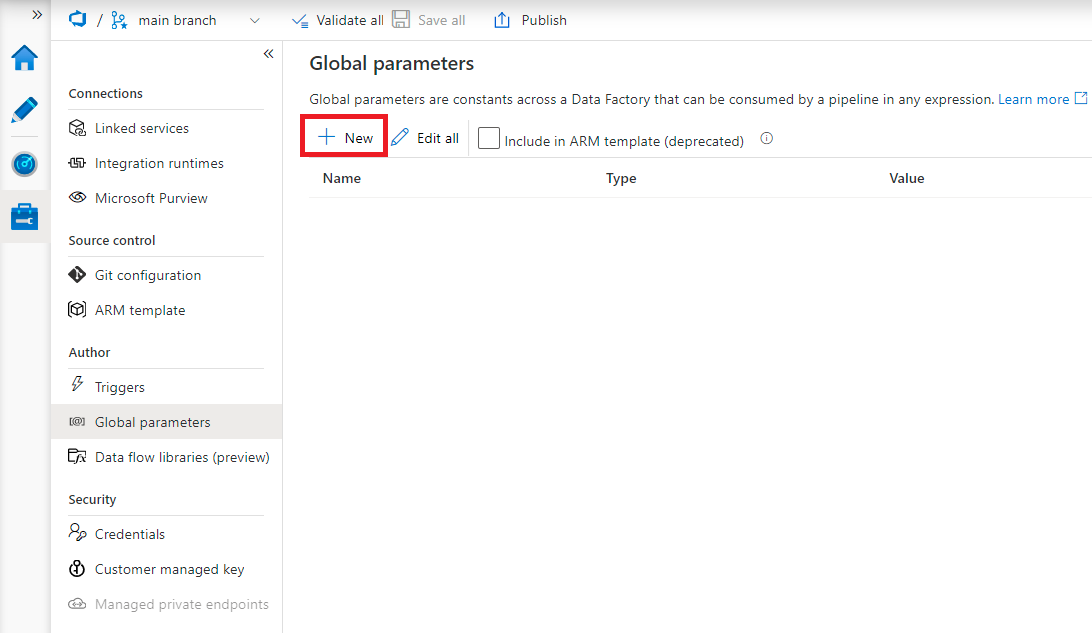
Parameters are placeholders used to pass values into pipelines, datasets, or linked services at runtime. These values are immutable during the execution of a pipeline.

**Why We Use Parameters :**

* To make pipelines reusable and configurable by passing external inputs (e.g., file names, dates, or database names).
* To avoid hardcoding values, making pipelines more flexible and adaptable.

**When to Use Parameters :**

* When you want to reuse the same pipeline for different datasets or scenarios, and the inputs vary (e.g., processing files from different folders or running queries for different dates).
* When you need to pass values into a pipeline from a trigger or parent pipeline.



**Functions :**

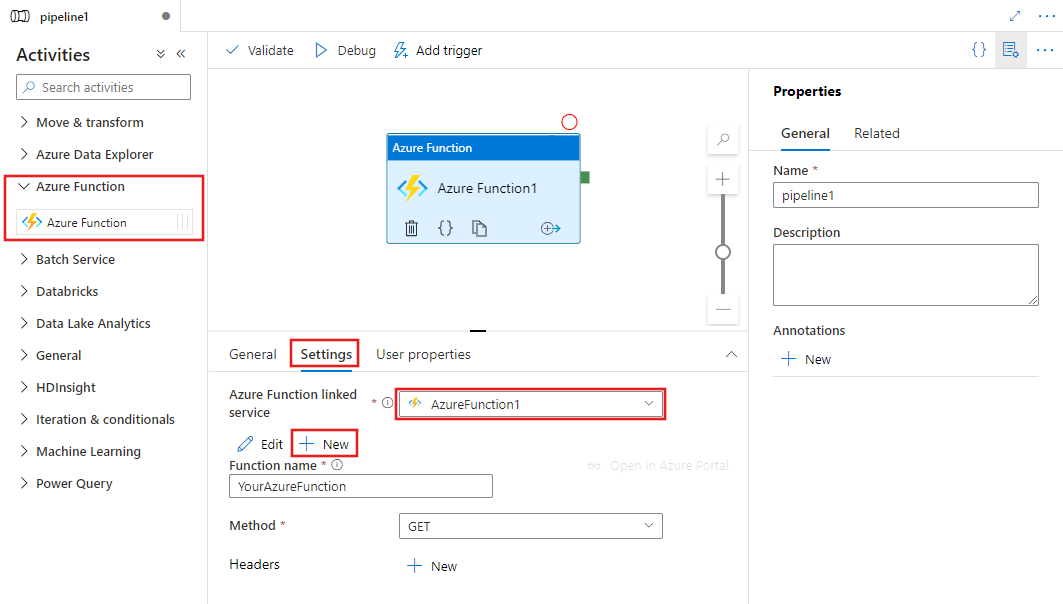
Functions in Azure Data Factory are predefined expressions used to perform transformations, manipulations, or evaluations on data. They enable dynamic and conditional logic in pipelines and datasets.

**Why We Use Functions :**

* To dynamically transform or manipulate data during runtime (e.g., formatting dates, converting strings, or performing arithmetic operations).
* To evaluate conditions and make decisions within pipelines.
* To build dynamic content using expressions.

**When to Use Functions :**

* When you need to create dynamic dataset paths or connection strings.
* When you need to apply transformations to variables or parameters at runtime.
* When building expressions for conditional logic in activities like **If Condition** or **Switch**.



|  |  |
| --- | --- |
| **Features** | **When to use** |
| **Variables** | To store and update values temporarily during pipeline execution. |
| **Parameters** | To pass external, immutable inputs into pipelines or datasets at runtime for reusability and configurability. |
| **Functions** | To dynamically transform, manipulate, or evaluate data and enable dynamic behavior in pipelines and datasets. |