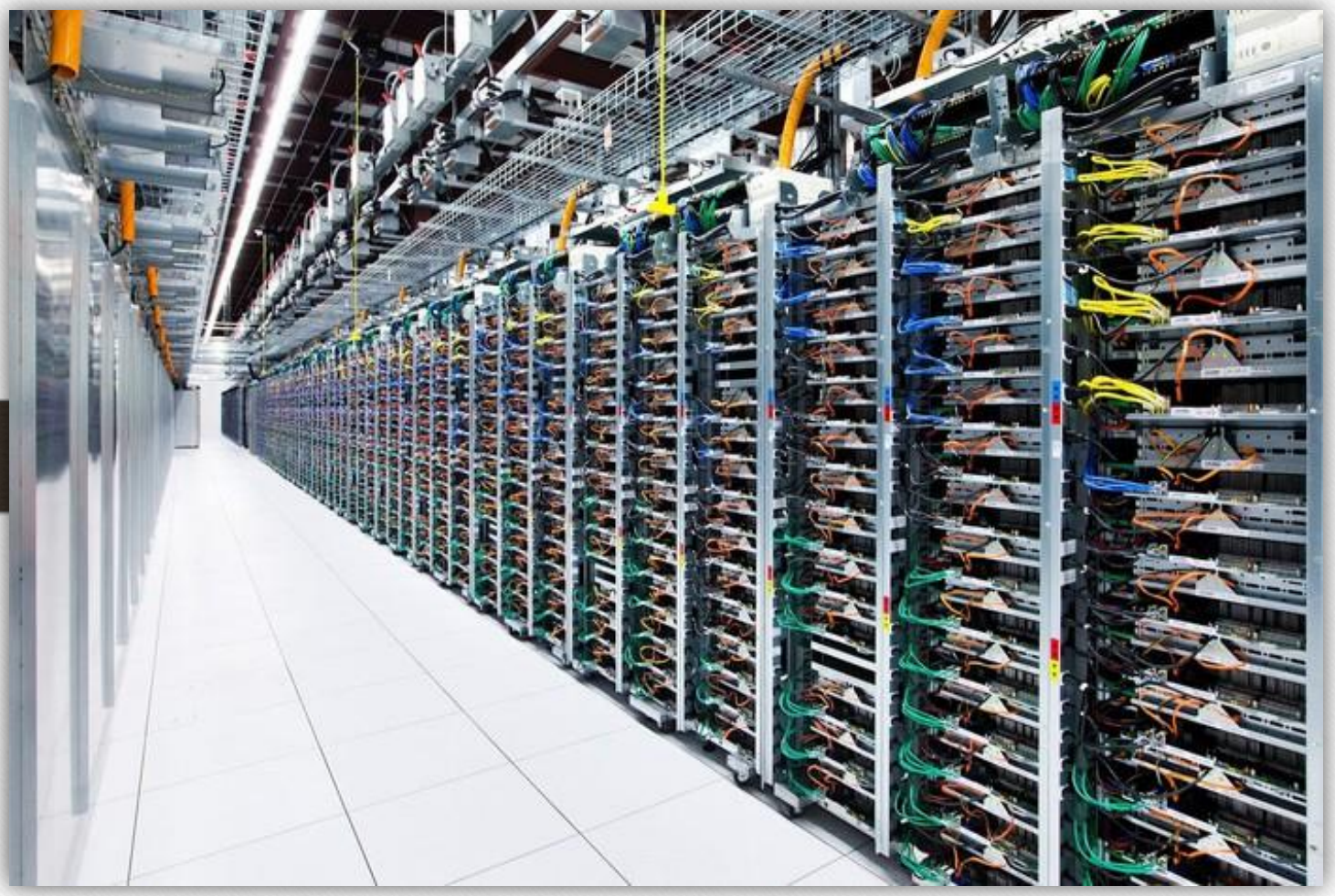
Youtube: TechLake

Introduction To Azure Cloud And Data Factory Blob & ADLS

Source : Azure portal and Microsoft standard documentation

Azure Data Center

Azure Data Center Racks



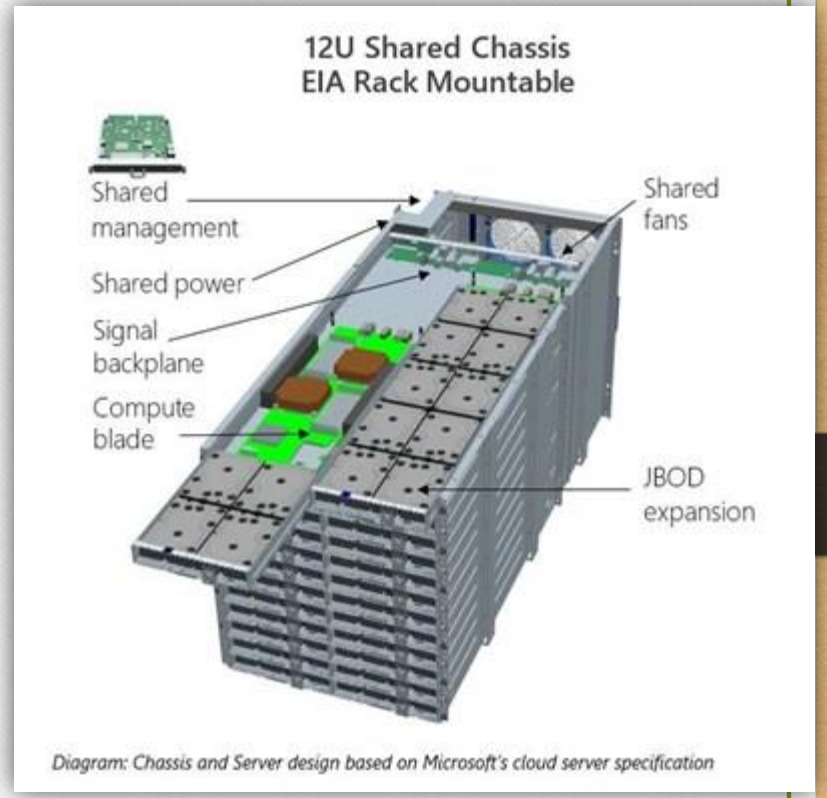


Diagram: Chassis and Server design based on Microsoft's cloud server specification

Azure Data Center Capsule

Types of Cloud Services

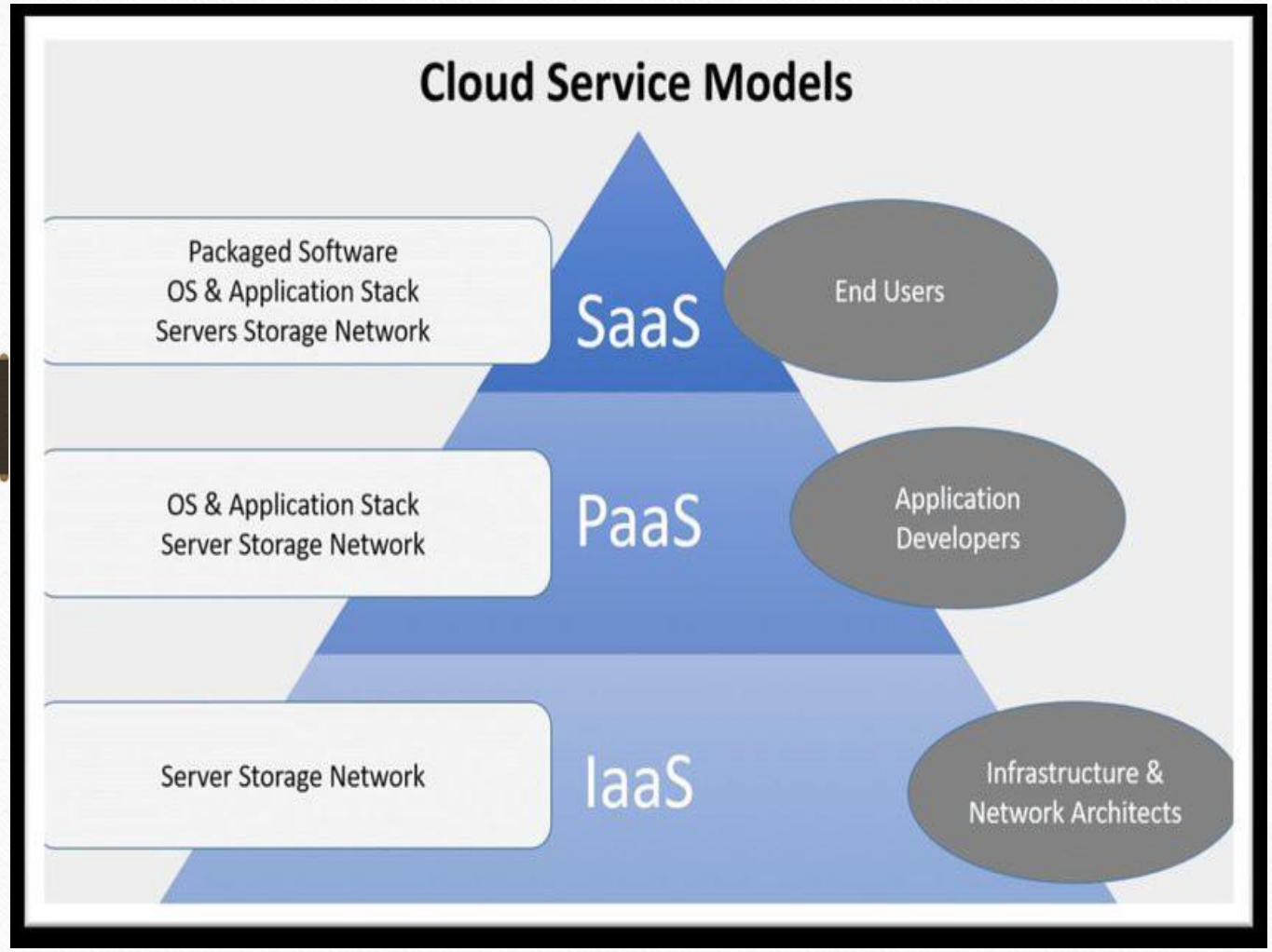
IaaS: Most flexible category, giving full control on the hardware that runs the app.

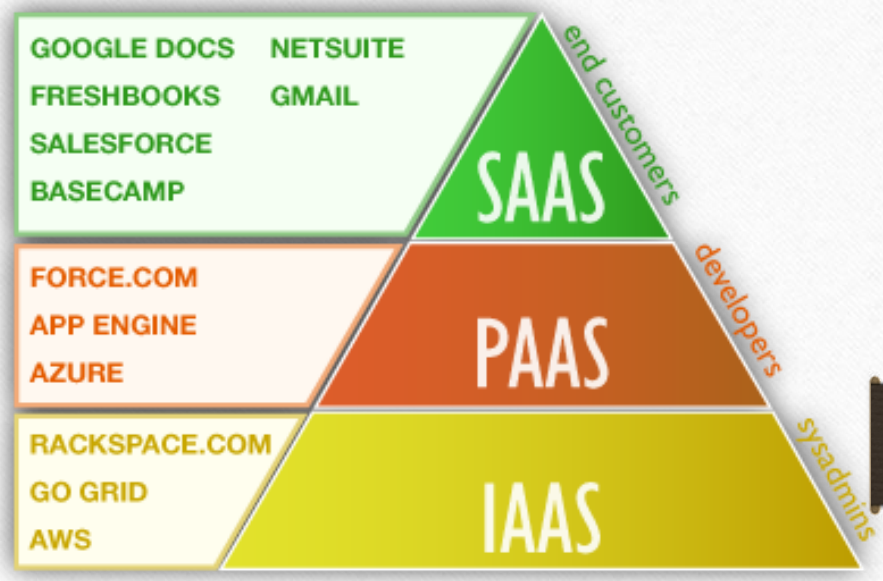
Shared responsibility model: Ensuring that a service is running is a shared responsibility between the cloud provider (in charge of the infrastructure) and the customer (who needs to set up the proper configurations). This is the usual scenario when migrating workloads - as you can mimic on-premise setup, testing and deployment for new apps and for storage, backup and recovery.

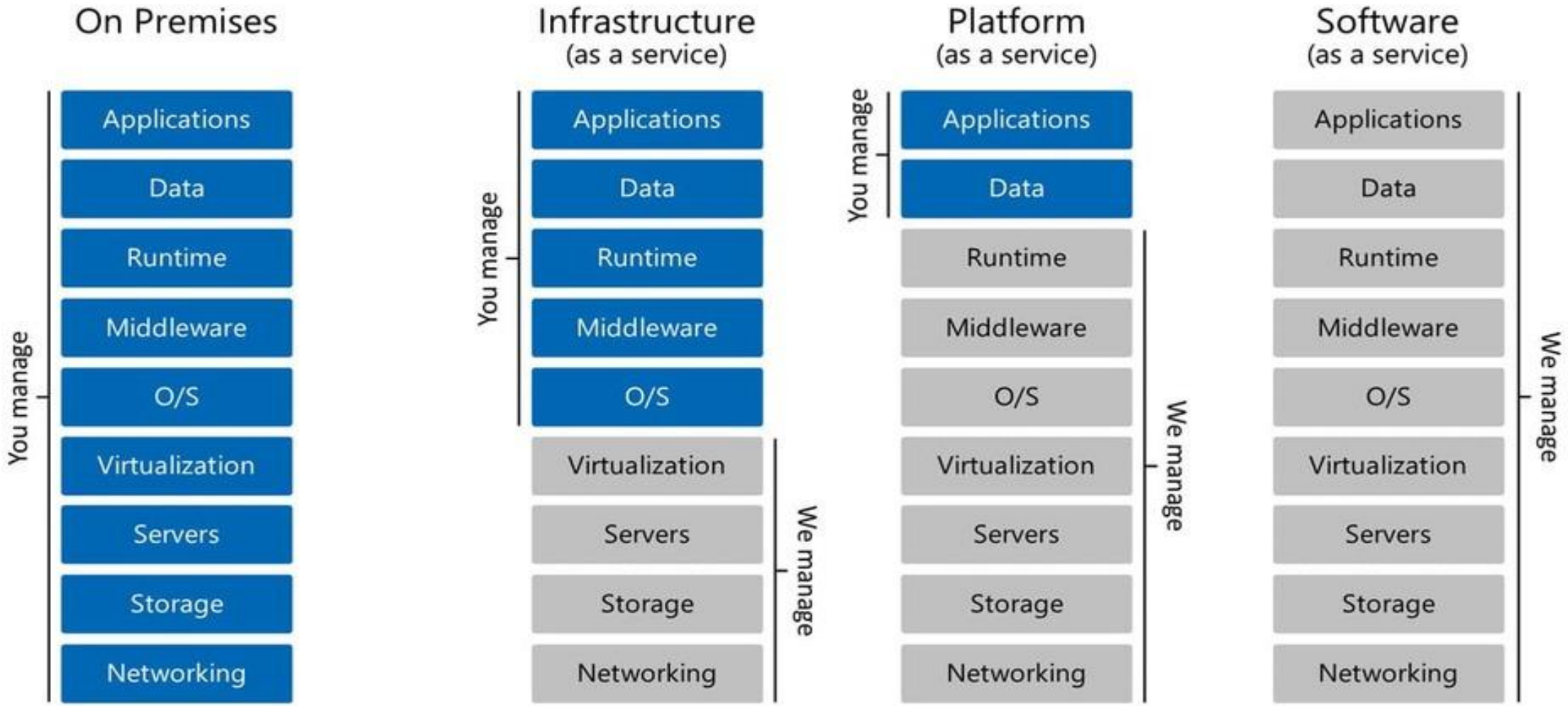
PaaS: Provides an environment for building, testing and deploying software applications, where one does not need to care about the underlying infrastructure. It is useful as it brings a framework that developers can work upon and also for analytics or BI tasks.

SaaS: Software hosted and managed for the customer, who acts as end user.

Cloud Service Models







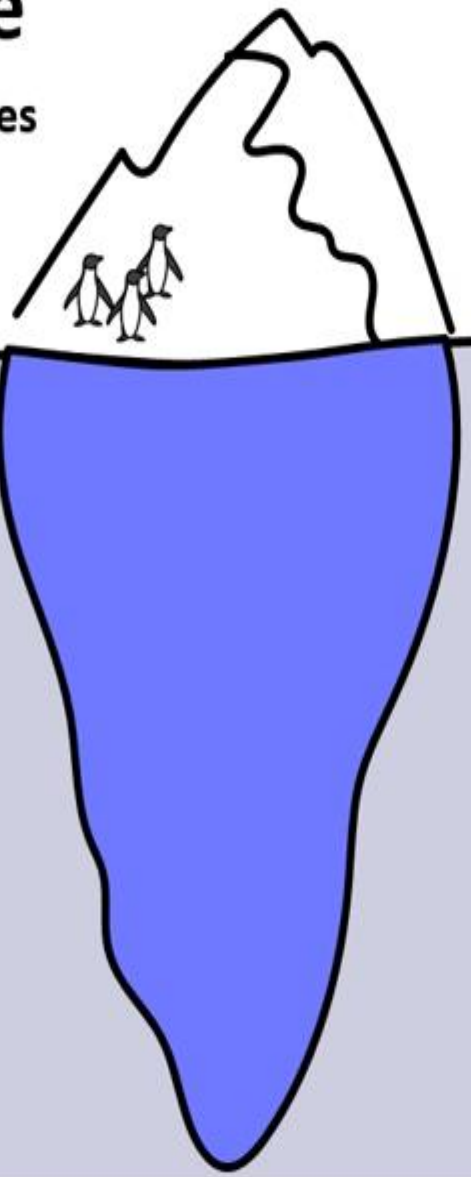
	Cost	Security	Level of Configuration	Technical Knowledge
Public Cloud	👍 Most cost-effective	👍 Security Controls by Default 👎 Might not meet security requirements	👎 Limited based on what the Cloud Service Provider exposes to you.	👍 You don't need in-depth knowledge of underlying infrastructure
Private Cloud	👎 Most expensive	👎 no guarantee its secure 👍 can meet any security compliance requirement if you put in the work.	👍 You can configure the infrastructure however you like.	👎 You need to know in-depth how to configure all levels of your infrastructure
Hybrid	👍👎 Could be more cost-effective based on what you offload to the cloud.	👎 you now have to secure your connection to the cloud 👍 can meet all security requirements	👍 You get the best of both worlds.	👎 You need to know in-depth how to configure all levels of your infrastructure and know the CSPs services.

CAPEX

On-Premise

Software license Fees

- Implementation
- Configuration
- Training
- Physical Security
- Hardware
- IT Personal
- Maintenance

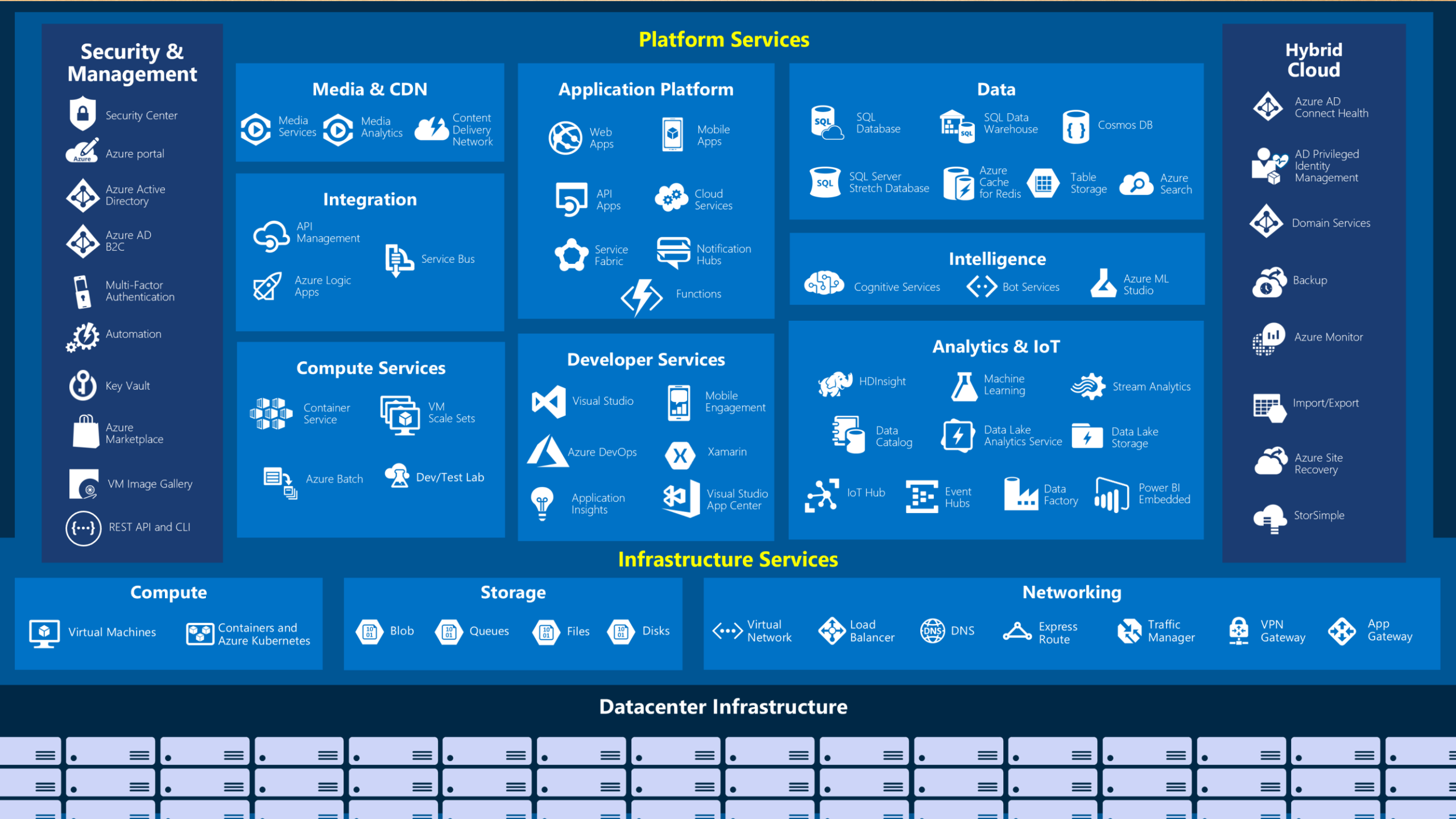


OPEX

Azure

Subscription Fees

- Implementation
- Configuration
- Training


Security & Management
Security Center
Azure portal
Azure Active Directory
Azure AD B2C
Multi-Factor Authentication
Automation
Key Vault
Azure Marketplace
VM Image Gallery
REST API and CLI
Platform Services
Media & CDN
Media Services
Media Analytics
Content Delivery Network
Integration
API Management
Service Bus
Azure Logic Apps
Compute Services
Container Service
VM Scale Sets
Azure Batch
Dev/Test Lab
Application Platform
Web Apps
Mobile Apps
API Apps
Cloud Services
Service Fabric
Notification Hubs
Functions
Developer Services
Visual Studio
Mobile Engagement
Azure DevOps
Xamarin
Application Insights
Visual Studio App Center
Data
SQL Database
SQL Data Warehouse
Cosmos DB
SQL Server Stretch Database
Azure Cache for Redis
Table Storage
Azure Search
Intelligence
Cognitive Services
Bot Services
Azure ML Studio
Analytics & IoT
HDInsight
Machine Learning
Stream Analytics
Data Catalog
Data Lake Analytics Service
Data Lake Storage
IoT Hub
Event Hubs
Data Factory
Power BI Embedded
Hybrid Cloud
Azure AD Connect Health
AD Privileged Identity Management
Domain Services
Backup
Azure Monitor
Import/Export
Azure Site Recovery
StorSimple
Infrastructure Services
Compute
Virtual Machines
Containers and Azure Kubernetes
Storage
Blob
Queues
Files
Disks
Networking
Virtual Network
Load Balancer
DNS
Express Route
Traffic Manager
VPN Gateway
App Gateway
Datacenter Infrastructure

Your ability to **increase your capacity** based on the increasing demand of traffic, memory and computing power

Vertical Scaling

Scaling **Up**

Horizonal Scaling

Scaling **Out**

Your ability to **automatically** increase or decrease your capacity based on the current demand of traffic, memory and computing power

Azure VM Scale Sets
Automatically increase or decrease in response to demand or a defined schedule.

SQL Server Stretch Database
Dynamically stretch warm and cold transactional data from Microsoft SQL Server 2016 to Microsoft Azure

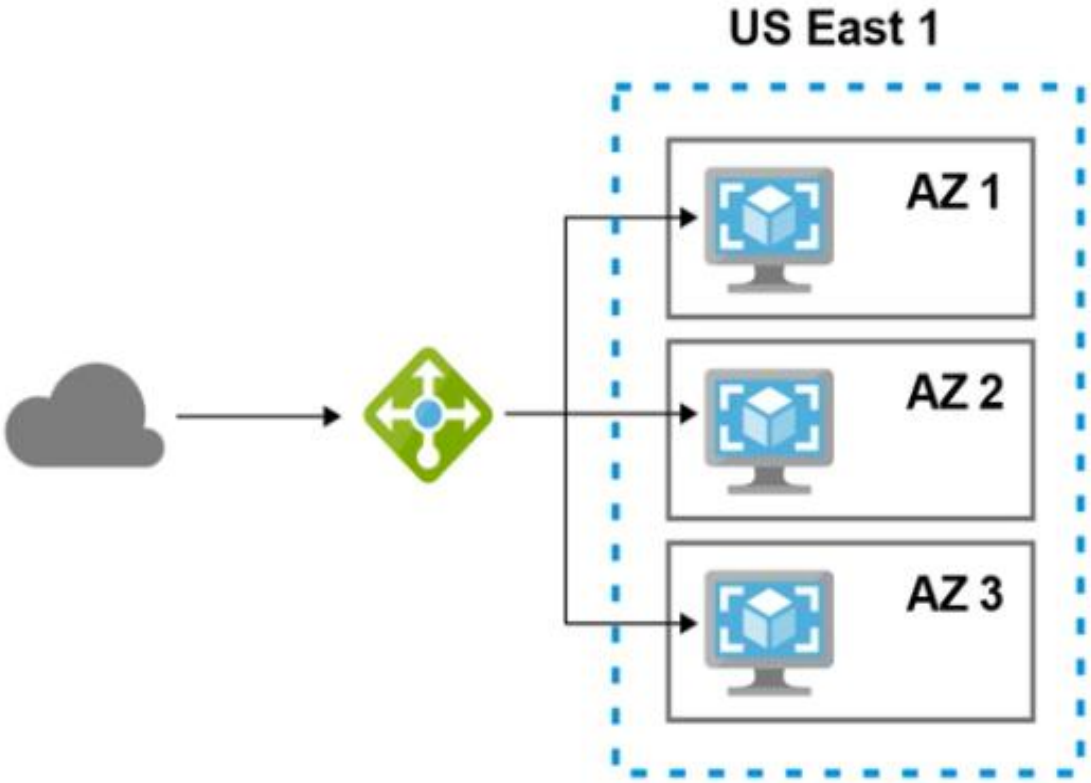
Horizonal Scaling

Scaling **Out** — Add more servers of the same size

Scaling **In** — Removing more servers of the same size

Vertical Scaling is generally hard for traditional architecture so you'll usually only see horizontal scaling described with Elasticity.

Your ability for your service to **remain available** by ensuring there is ***no single point of failure** and/or ensure a certain level of performance



Azure Load Balancer

A load balancer allows you to evenly distribute traffic to multiple servers in one or datacenter. If a datacenter or server becomes unavailable (unhealthy) the load balancer will route the traffic to only available datacenters with servers.

Running your workload across multiple **Availability Zones** ensures that if 1 or 2 **AZs** become unavailable your service / applications remains available.

Datacenters and Regions

Cloud providers are built upon datacenters around the globe, where the physical hardware is located.

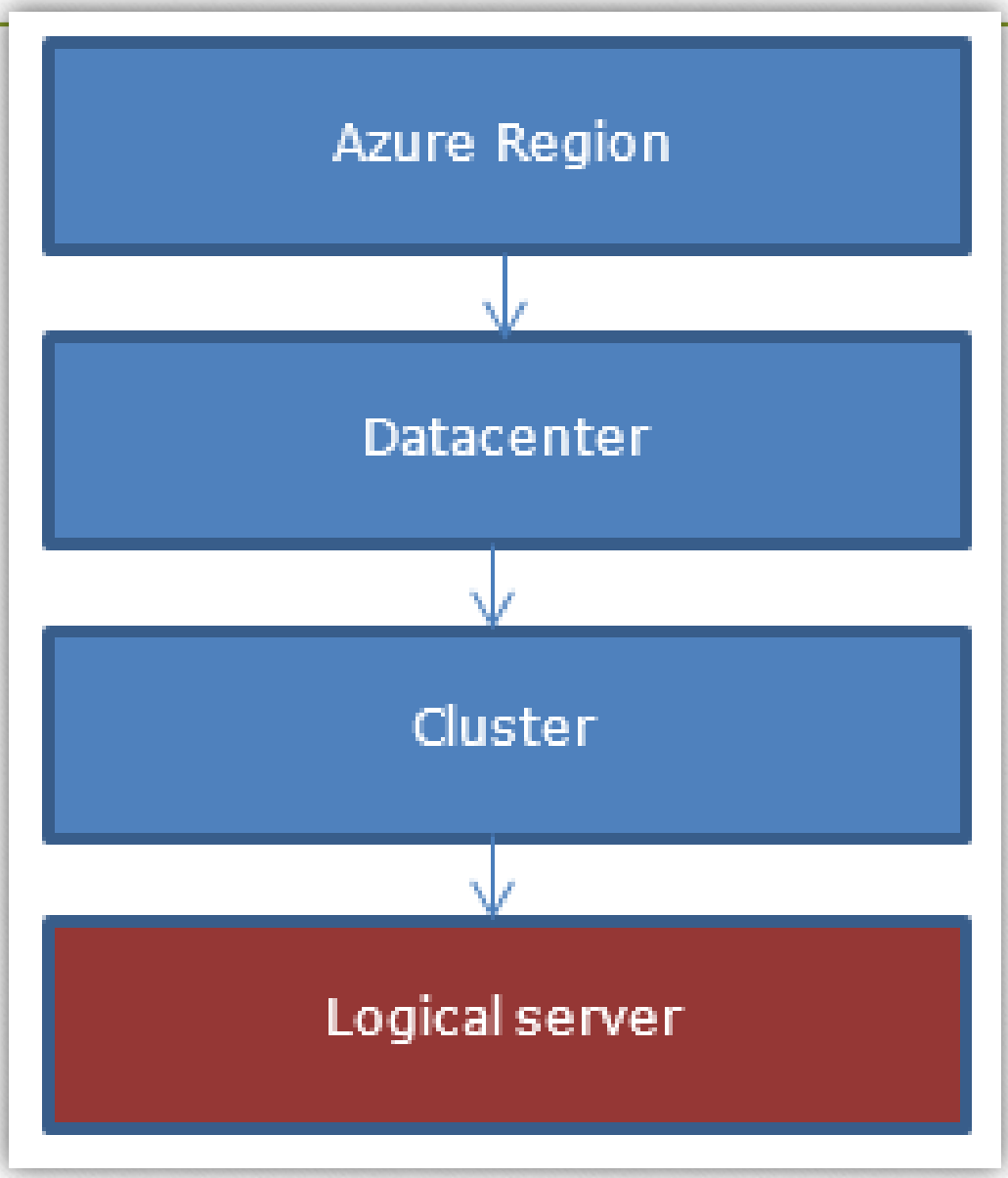
Regions are geographical areas on the planet containing +1 datacenter. This partition into regions gives flexibility to provide services which are close - and thus ensure lower latency - to the user.

Geographies

Azure divides the world into geographies that are defined by geopolitical boundaries. Each geography preserve data residency and compliance needs required by those countries. Moreover, they are fault-tolerant to withstand complete region failure. We have the following geographies:

1. Americas
2. Europe
3. Asia Pacific
4. Middle East and Africa

Each region belongs to a single geography and has specific service availability, compliance, and data residency/sovereignty rules applied to it


Azure Region
Datacenter
Cluster
Logical server

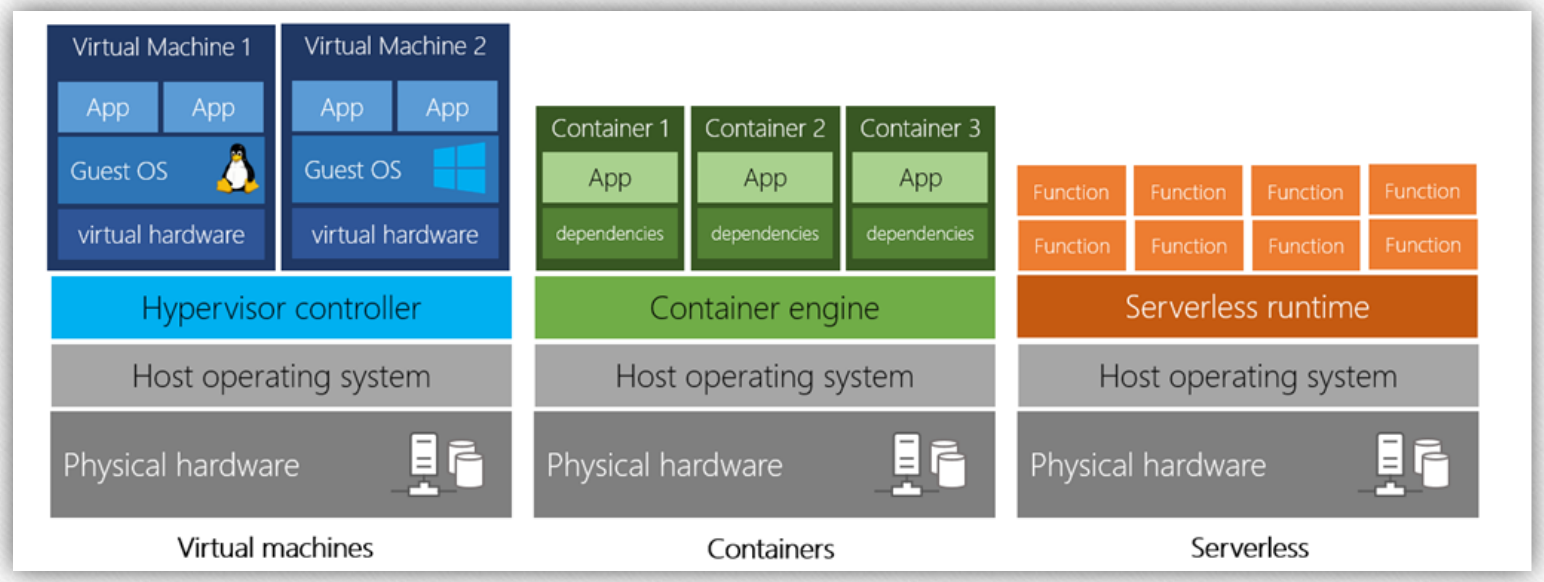
Azure Compute Power

It is the processing capability offered by cloud-based servers. Depending on our needs, we can chose among three different options:

Virtual Machines (VM): You are provided the hardware and the OS, but have to control and maintain the rest of the software. This VM will run on a physical server based on a datacenter and overall resources will be shared with other VMs in an isolated and secure fashion.

Containers: Such as Docker, offer an independent, consistent and isolated execution environment for applications. They do not require any OS - as they just contain the minimum dependencies to run the app - they are easy and fast to spin up, move to a different machine and scale.

Serverless Computing: Is the ability to run an application without creating, configuring nor maintaining a server. By breaking the whole application into smaller pieces, we can define a workflow and a set of triggers to execute the different parts of the application. As per pricing, with VMs and Containers, we pay for as long the server is up, eventhough the application is idle. With Serverless Computing though, you only pay for the processing time of each function.



Benefits of Cloud Computing

Cost-Effective: with a pay-as-you-go model you don't need to make huge upfronts investments to start offering your services. Moreover, you don't need to manage any kind of infrastructure, freeing people to focus on other tasks.

Scalable: It's easy to increase / decrease the resources used either manually or automatically.
Vertical Scaling or scaling up can be used to increase the power of an existing machine.
Horizontal Scaling or scaling out adds more servers to function together as a unit.

Elastic: Can automatically adapt to workload changes by adding or removing resources.

Current: By removing the need of managing hardware and other IT tasks, you can focus on just building and deploying your apps.

Reliable: As it offers backups, disaster recovery and data replication services to make sure that your data is always safe. If on top you follow redudancy on your architectures you can avoid single points of failure.

Global: With datacenters all over the globe, you can have presence close to your customers and reduce response time.

Secure: Offering policies, technologies and controls.

Accounts and subscriptions

An Azure account is an identity in either Azure Active Directory (Azure AD), or a directory that is trusted by Azure AD, such as a work or school organization. It holds information such as:

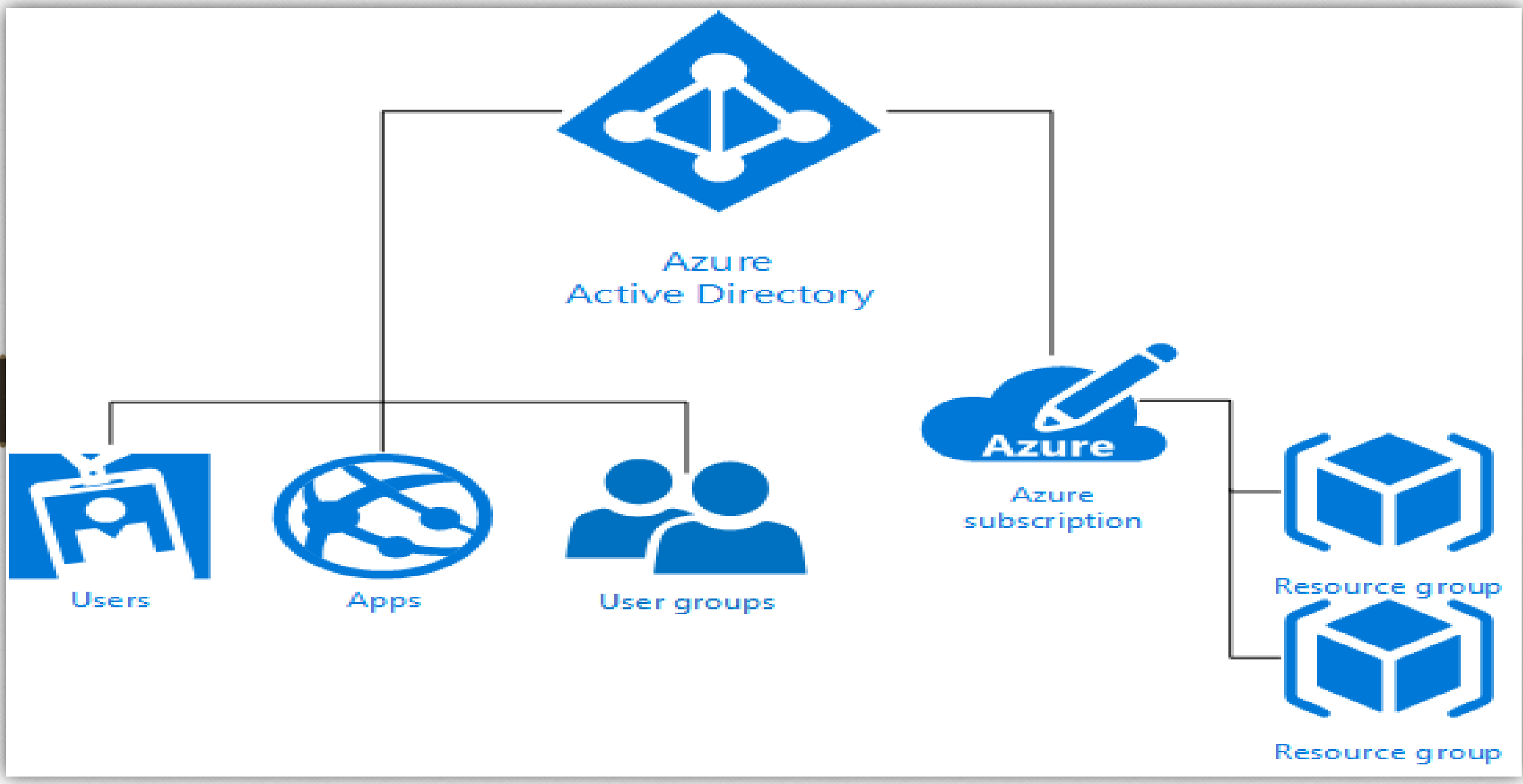
Name, email, and contact preferences , Billing information such as a credit card
You use an Azure account to sign in to the Azure website and administer or deploy services. Every Azure account is associated with one or more subscriptions.
An Azure subscription is a logical container used to provision resources in Azure and is associated with Azure AD.

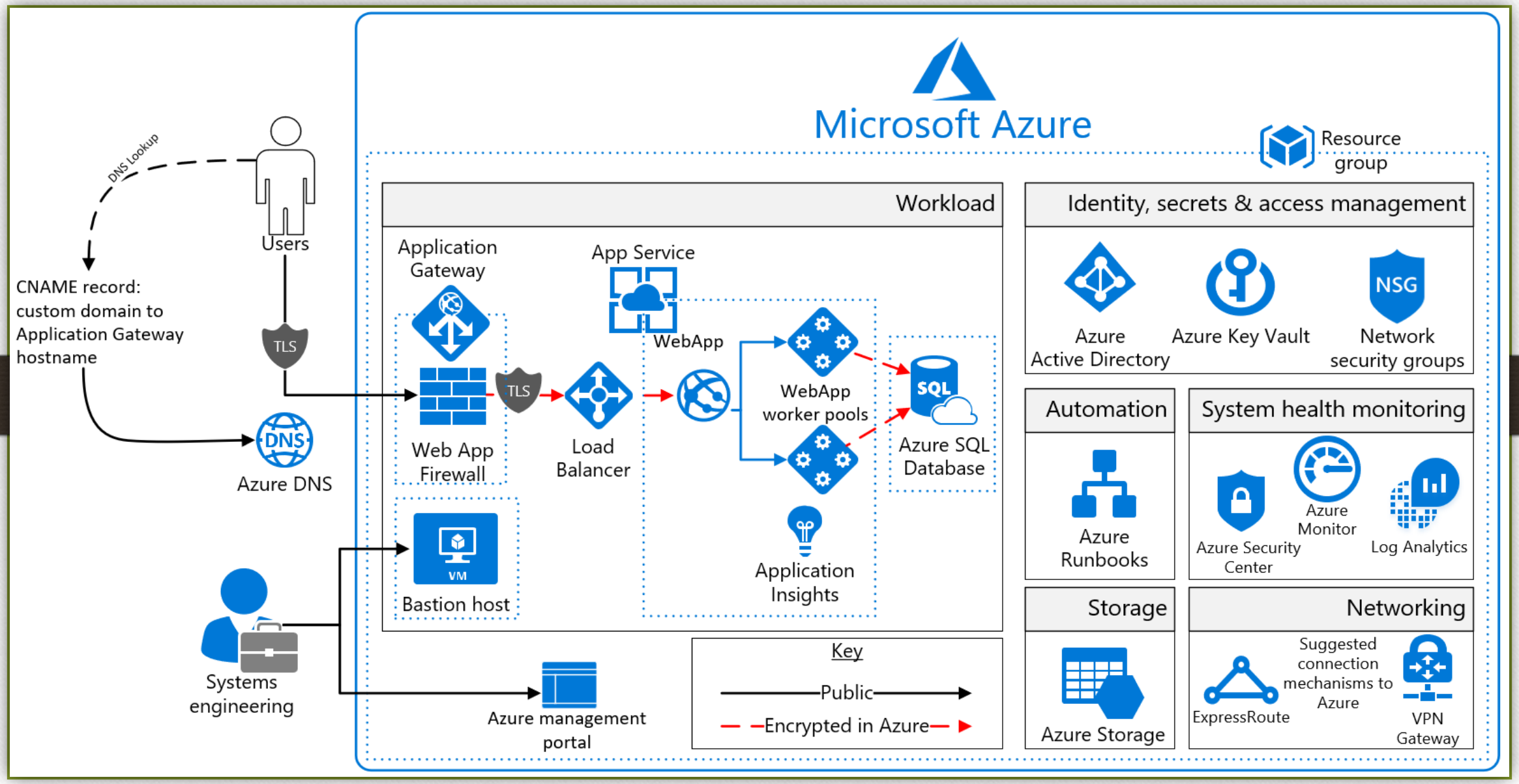
There are four different subcription types:

1. Free
2. Pay-As-You-Go
3. Enterprise Agreement
4. Student

When to use multiple Azure subscriptions

Sometimes it is useful to separate resources at a subcription level, as billing and access control happen at the subscription. We could use this to separate DEV and PROD environments for billing and security reasons or even to isolate some resources for compliance.


Azure
Active Directory
Users
Apps
User groups
Azure
Azure
subscription
Resource group
Resource group


Microsoft Azure
Resource group
DNS Lookup
Users
CNAME record:
custom domain to
Application Gateway
hostname
TLS
DNS
Azure DNS
Workload
Application Gateway
Web App Firewall
TLS
Load Balancer
App Service
WebApp
WebApp worker pools
Application Insights
SQL
Azure SQL Database
VM
Bastion host
Systems engineering
Azure management portal
Key
Public
Encrypted in Azure
Identity, secrets & access management
Azure Active Directory
Azure Key Vault
NSG
Network security groups
Automation
Azure Runbooks
System health monitoring
Azure Security Center
Azure Monitor
Log Analytics
Storage
Azure Storage
Networking
ExpressRoute
Suggested connection mechanisms to Azure
VPN Gateway

Azure management options

There is a broad selection of tools that can be used to configure and manage Azure.

Azure portal for interacting with Azure via a Graphical User Interface (GUI). It uses a blades model for navigation, where a blade is a slide-out panel containing the UI for a single level in a navigation sequence.
Azure PowerShell and Azure Command-Line Interface (CLI) for command line and automation-based interactions with Azure.
Azure Cloud Shell for a web-based command-line interface.
Azure mobile app for monitoring and managing your resources from your mobile device.

Azure Data Factory

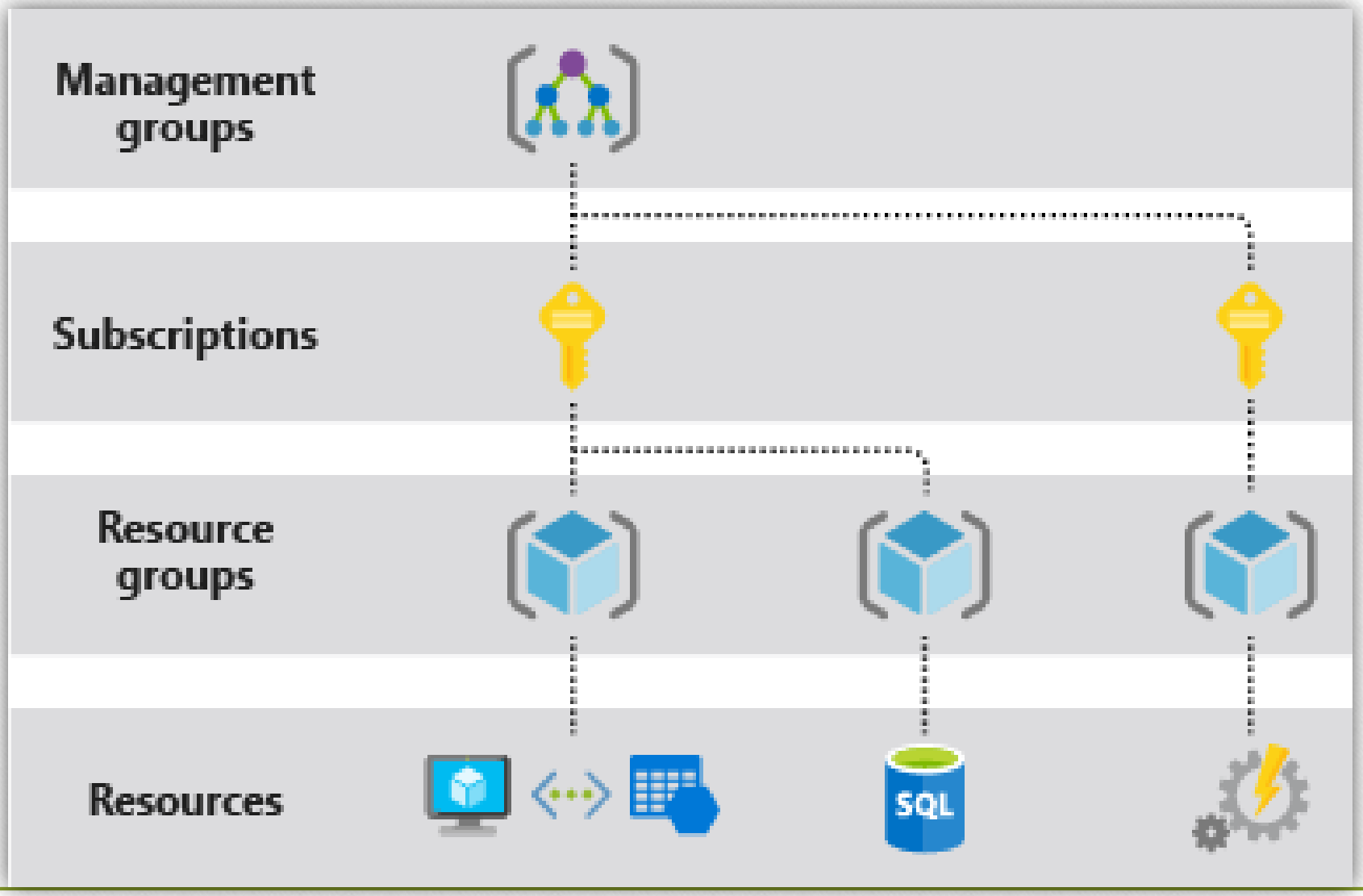
Introduction To Azure Data Factory Content

- Introduction To Azure Components Related to ADF
- ADF Architecture
- ADF V2 Components
- Configuring ADF for Copying Data
- Scheduling a Pipeline
- Transforming data
- Copying Data from Different Sources

Azure Cloud Components/Resources

- Azure Blob Storage
- Data Lake Store
- Azure HDInsight Hadoop Cluster
- Azure Data Factory V2
- Azure SQL DB
- Azure Databricks

Azure Cloud Subscription & Resources.



AZURE Cloud Subscription & Resources

What is Azure Subscription?

A Windows Azure subscription grants you access to Windows Azure services and to the Windows Azure Platform Management Portal.

What is Resource Group?

Resource Group is a container that holds related resources for an Azure solution. The resource group includes those resources that you want to manage as a group. You decide how to allocate resources to resource groups based on what makes the most sense for your organization.

What is Resource?

Resource is a manageable item that is available through Azure. Virtual machines, storage accounts, web apps, databases, and virtual networks are examples of resources.

What is Azure Data Factory?

It is a cloud-based data integration service that allows you to create data-driven workflows in the cloud for orchestrating and automating data movement and data transformation.

Benefits of using Azure to store data

1. Automated backup and recovery: mitigating the risk of losing data.
2. Replication across the globe: protecting your data.
3. Support for data analytics
4. Encryption capabilities: making it secure. You also have control over who can access the data.
5. Multiple data types: for both relational and NoSQL.
6. Data storage in virtual disks: with up to 8TB capacity per disk.
7. Storage Tiers: to prioritize access to data based on frequently used vs. rarely used information.

Types of Data

Structured data: data that adhers to a schema so all of it has the same fields or properties. Also known as relational data.

Semi-structured data: which does not neatly fit into tables, rows and columns. Uses keys to organize and provide a hierarchy. Also known as non-relational or NoSQL data.

Unstructured data: No structure means no restrictions. For example, a blob can hold a PDF, JPG, JSON, video...

Azure data storage and features

1. **Azure SQL**: As a Database as a Service (DaaS), it gives the latest stable SQL server engine in a high performance, reliable, fully managed and secure way.
2. **Azure Cosmos DB**: Distributed database service which supports schema-less data. This means that it supports applications with changing data.
3. **Azure Blob Storage**: It is unstructured, so no restrictions applied. Blobs are scalable and it's easy to rowk with blobs with applications. They can manage thousands of simultaneous uploads and massive amounts of data.
4. **Azure Data Lake Storage**: Large repository hosting both structured and unstructured data which allows to perform analytics on the data. Combines the scalability and cost benefits of object storage with the reliability and performance of the Big Data file system capabilities.
5. **Azure Files**: Offers fully managed file shares in the cloud that are accessible via the industry standard Server Message Block (SMB) protocol. Machines can mount these to have access to the data.
6. **Azure Queue**: Service for storing large numbers of message. It also can be used to help build flexible applications and separate function s for better durability across large workloads and provides asynchronous message queueing for communication between application components.
7. **Disk Storage**: Disk storage provides disks for virtual machines, applications, and other services to access and use as they need, similar to how they would in on-premises scenarios.
8. **Storage Tiers:** Azure offers three storage tiers for blob object storage:
9. **Hot storage tier**: optimized for storing data that is accessed frequently.
10. **Cool storage tier**: optimized for data that are infrequently accessed and stored for at least 30 days.
11. **Archive storage tier:** for data that are rarely accessed and stored for at least 180 days with flexible latency requirements.
12. **Encryption and replication:** Azure provides security and high availability to your data through encryption and replication features:
13. **Encryption for storage services:** We can choose between SSE (Azure Storage Service Encryption) which encrypts data before writing and decrypts when reading transparently to the user or Client-side encryption, where the data is already encrypted by the client libraries.
14. **Replication for storage availability**: ensuring that data is durable and available.

What is Azure Data Lake?

Azure Data Lake Storage Gen1 is an enterprise-wide hyper-scale repository for big data analytic workloads. Azure Data Lake enables you to capture data of any size, type, and ingestion speed in one single place for operational and exploratory analytics.

Data Lake Storage Gen1 can be accessed from Hadoop (available with HDInsight cluster) using the WebHDFS-compatible REST APIs. It's designed to enable analytics on the stored data and is tuned for performance for data analytics scenarios. Data Lake Storage Gen1 includes all enterprise-grade capabilities: security, manageability, scalability, reliability, and availability.

Azure Data Lake Store Features

1) Open Source Compatibility
2) Unlimited Data Storage- Petabytes - Size files and Trillions of objects
3) Massively Parallel Processing (MPP)
4) Scales instantly , Pay as per Jobs
5) Enterprise Grade Security
6) Integrates Seamlessly with existing infrastructure
7) hybrid extract-transform-load (ETL), extract-load-transform (ELT) Supports

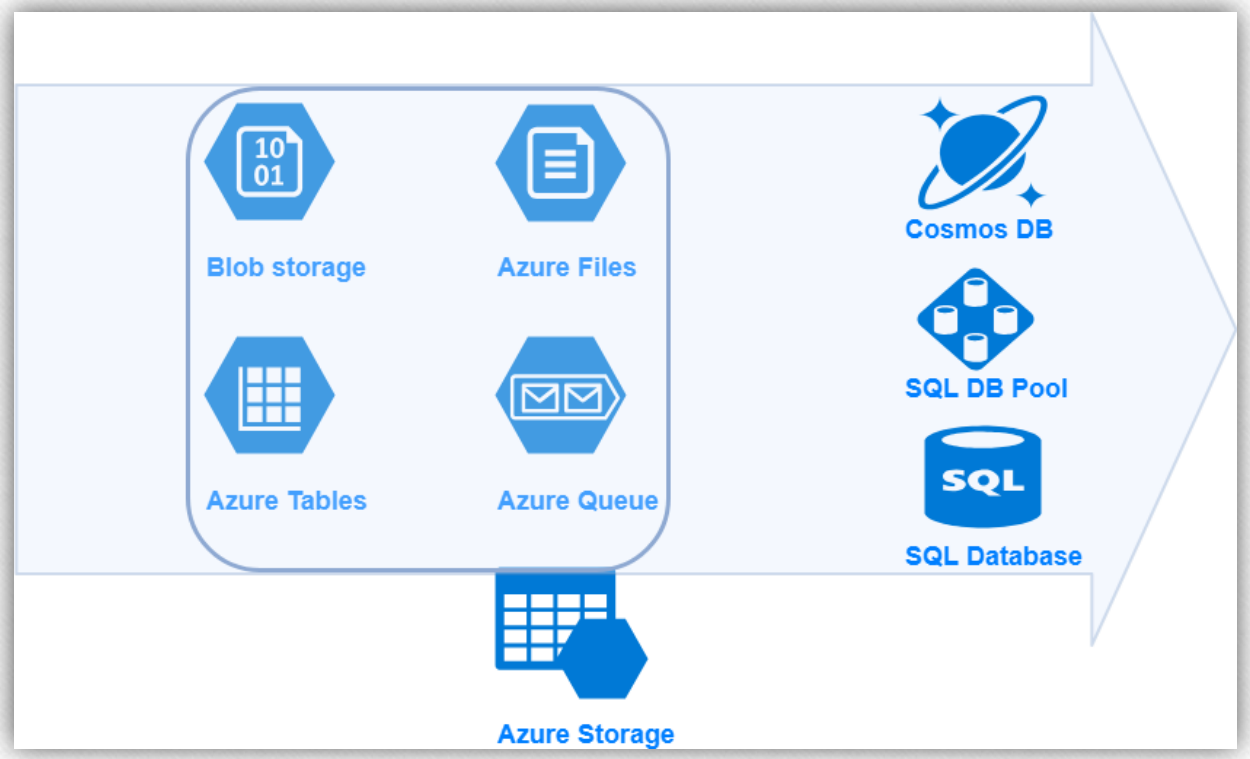
Comparison between Azure data storage and on-premises storage

Storing data in the cloud gives the following benefits vs. on prem solutions:

Cost effectiveness: as we don't need an upfront expense and just **pay-as-you-go**.

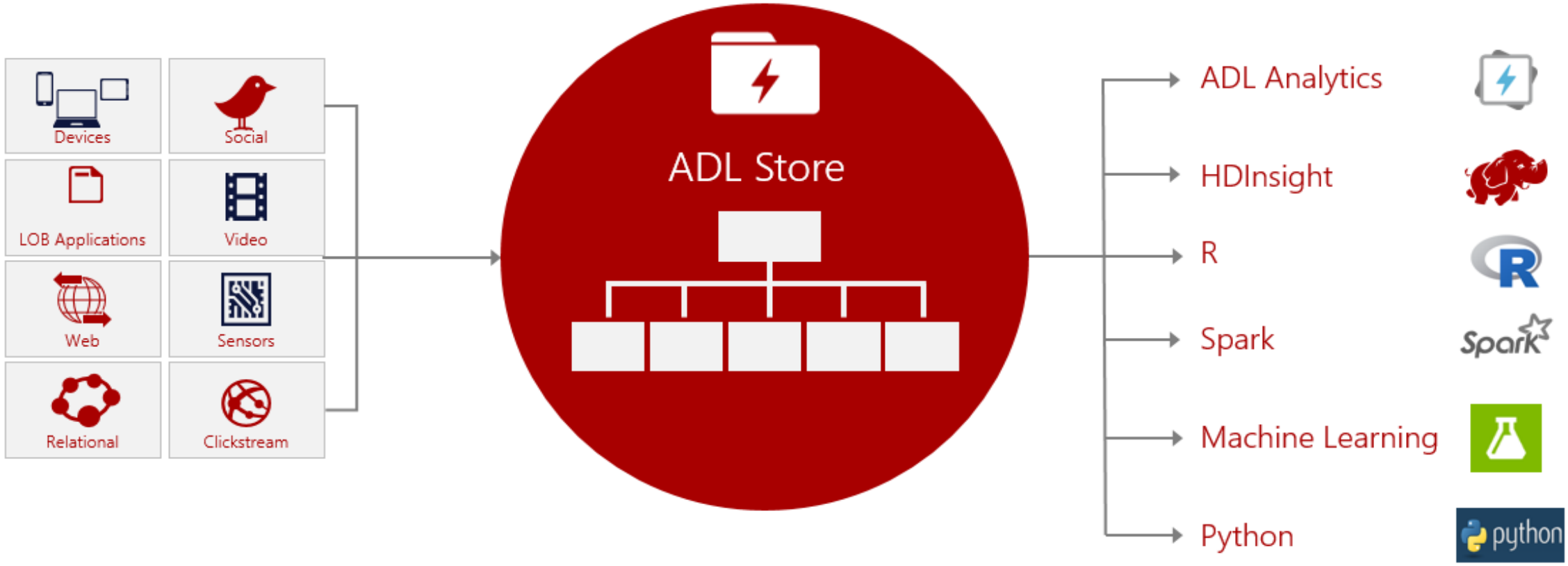
Reliability: Azure data storage provides data backup, load balancing, disaster recovery, and data replication as services to ensure data safety and high availability without as much investment as in on premise.

Multiple storage types depending on your needs. The agility to change the technologies used.

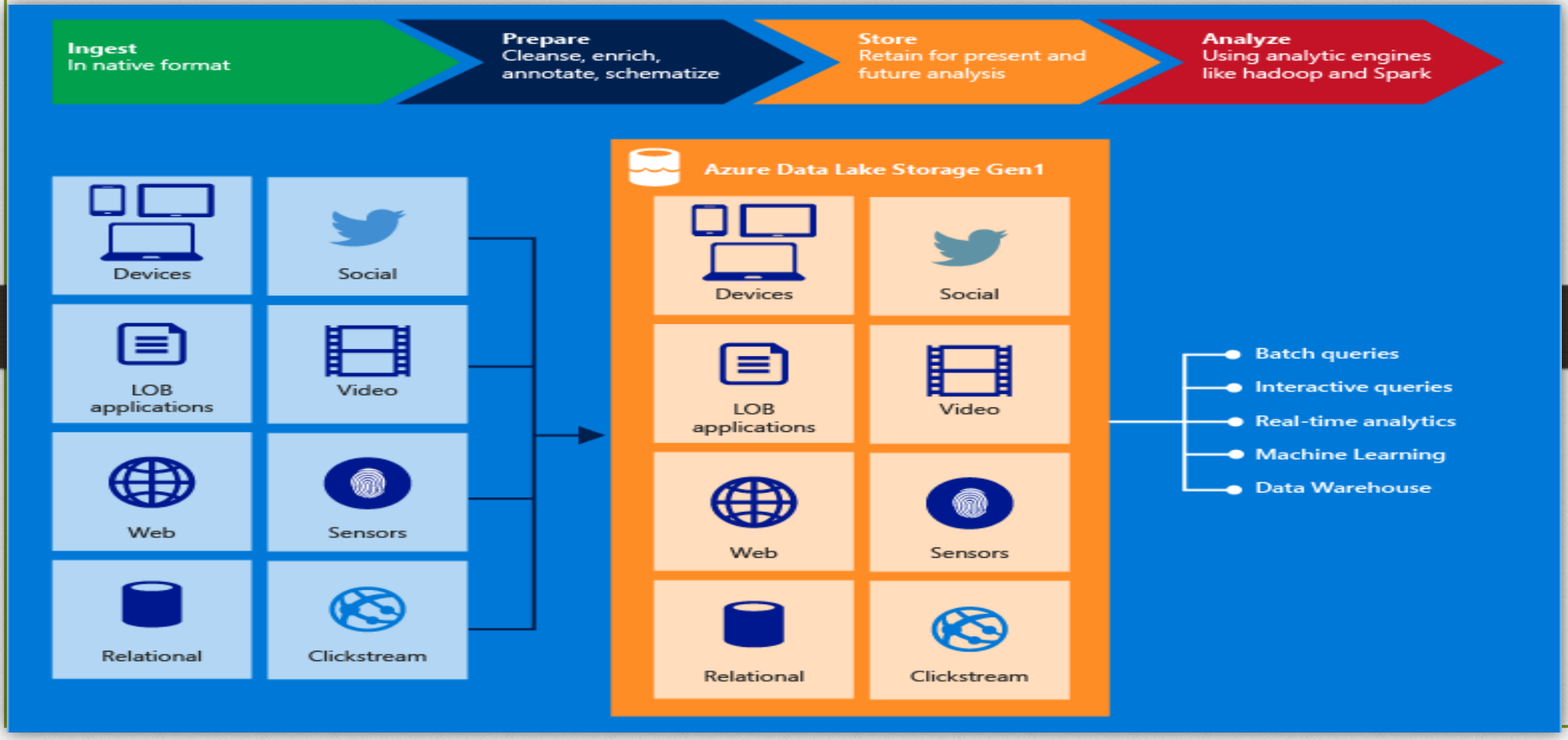


ADLS: work with multiple analytic frameworks

A highly scalable, distributed, parallel file system in the cloud specifically designed to work with multiple analytic frameworks

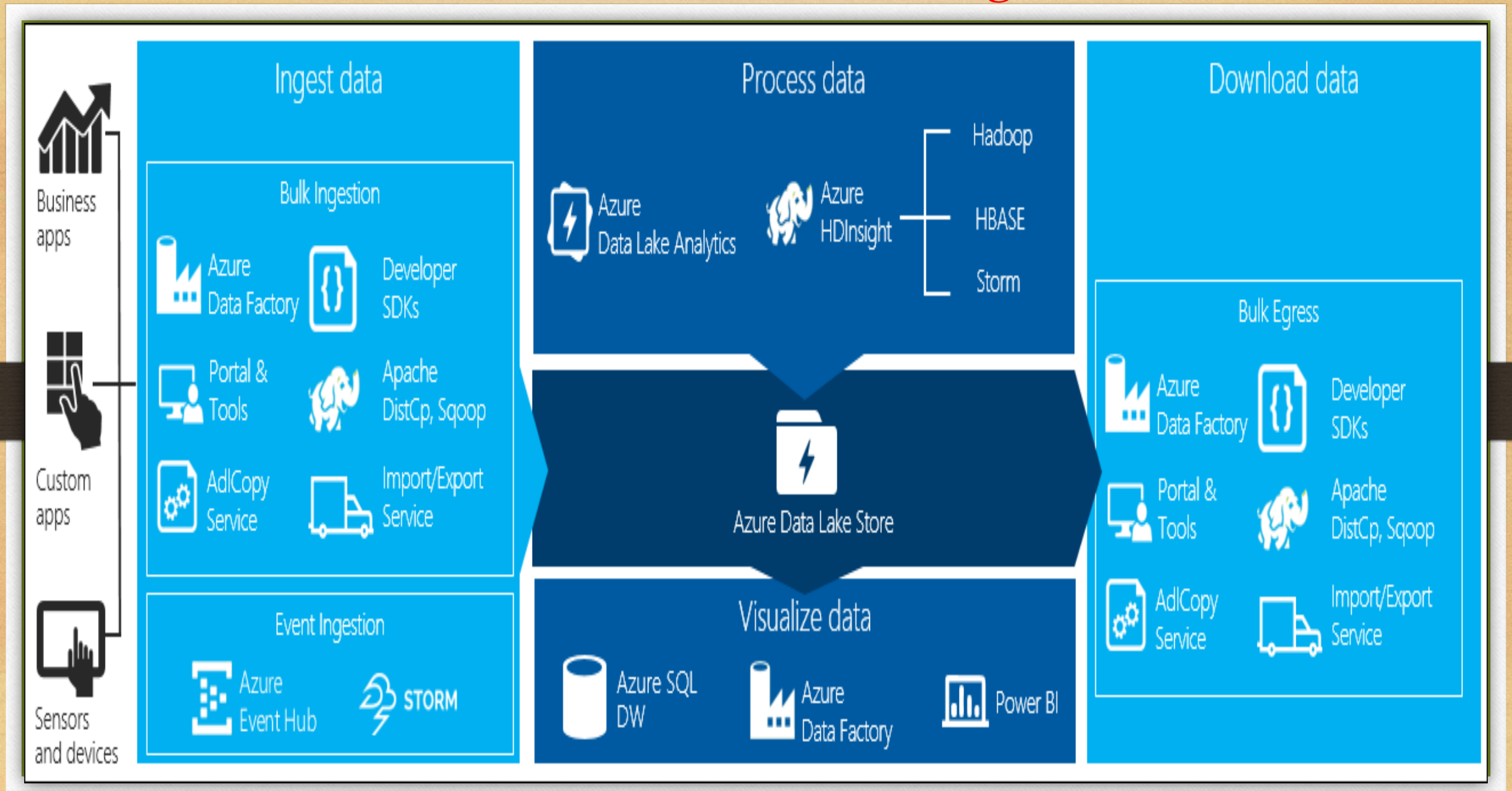


Azure Data Lake store uses



Azure Data Lake Store usage

Youtube : techlake



Azure Data Lake V2 Features

ADLS Gen2 includes most of all features from both ADLS Gen1 and Azure Blob Storage. These features include:

- The ability to store and analyse data of any kind and size.
- Enterprise-grade security with Azure Active Directory.
- Hierarchical File System (HFS)
- Read-access geo-redundant storage
- 5 TB file size limit
- Blob tiers (Hot, Cool, Archive)
- Multiple access methods including U-SQL, Spark, Hive, HBase, and Storm.
- Built on YARN and HDFS.
- Dynamic scaling to match your business priorities.

Azure Data Lake Pricing

Data Lake Store is offers Pay as you go monthly bases pricings as like below one of the region Central US.

Region: Central US

Currency: US Dollar ($)

Storage prices

Storage is available in pay-as-you-go and monthly commitment packages.

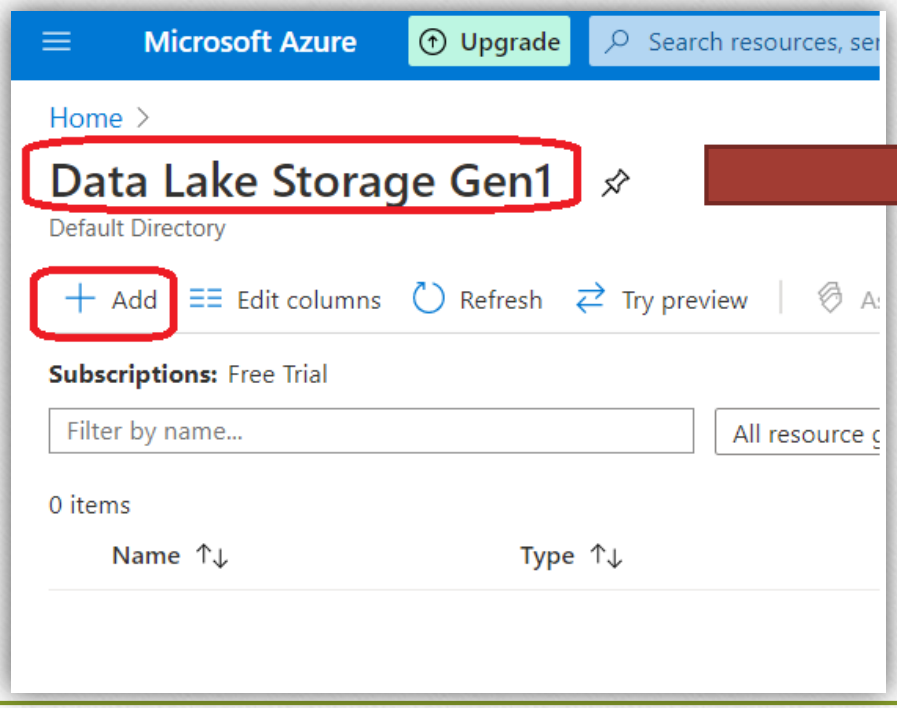
Pay-as-you-go

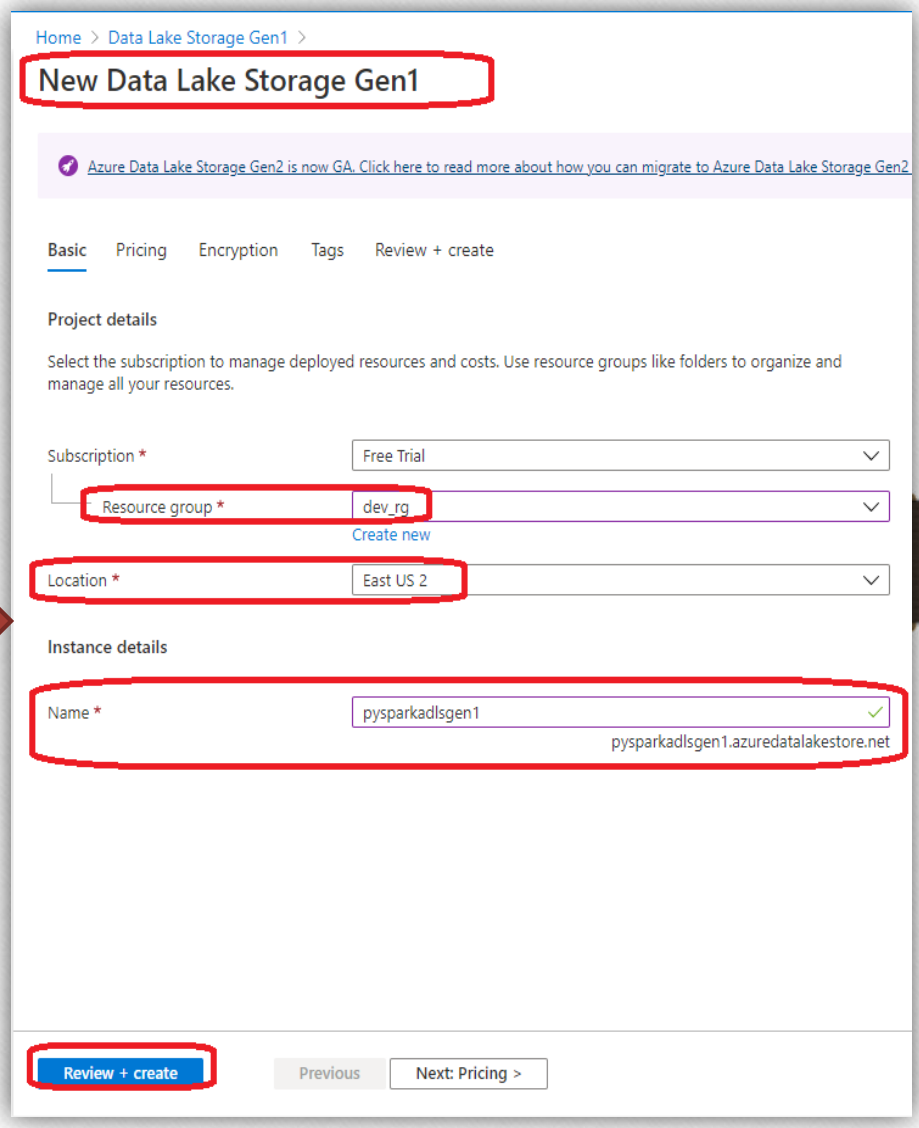
USAGE	PRICE/MONTH
First 100 TB	$0.039 per GB
Next 100 TB to 1,000 TB	$0.038 per GB
Next 1,000 TB to 5,000 TB	$0.037 per GB
Over 5,000 TB	Contact Us

1GB = 1024^3 bytes

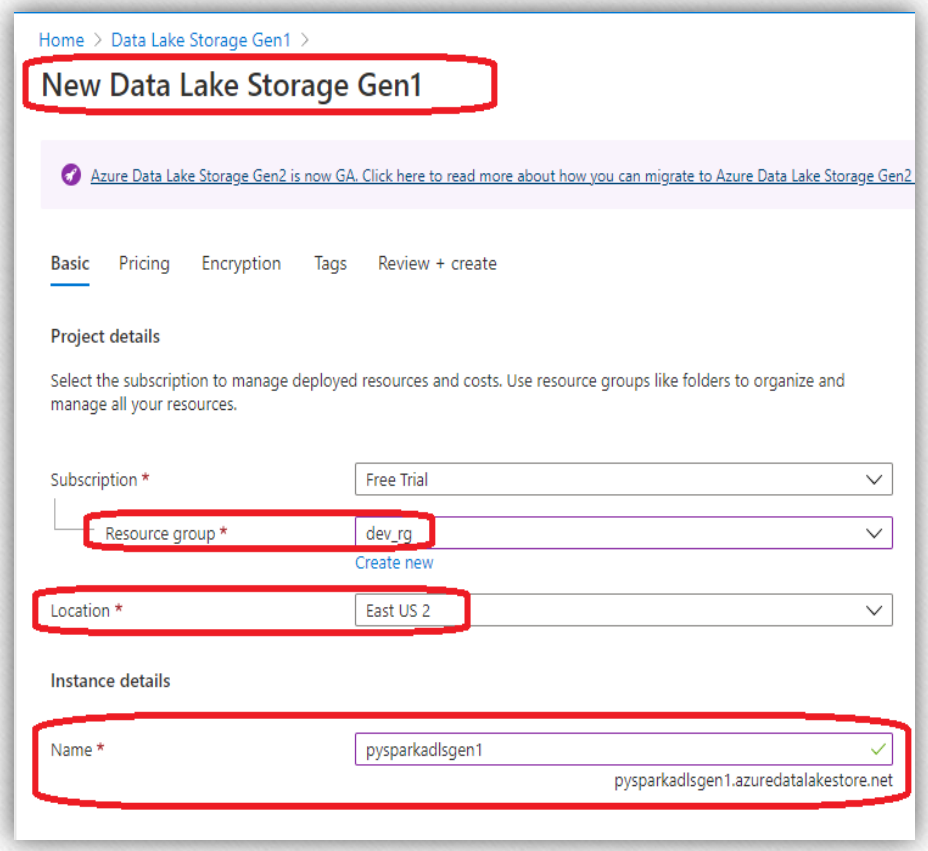
Create an Azure Data Lake Store account

1.Sign in to the new Azure Portal .
2.Click **NEW** , **Data + Storage** , then **Azure Data Lake Store** . Read the information in the **Azure Data Lake Store** dashboard , then click **Create** in the lower left corner of the dashboard.
3.In the **New Data Lake Store** panel , provide the values, as shown in the screenshot below:



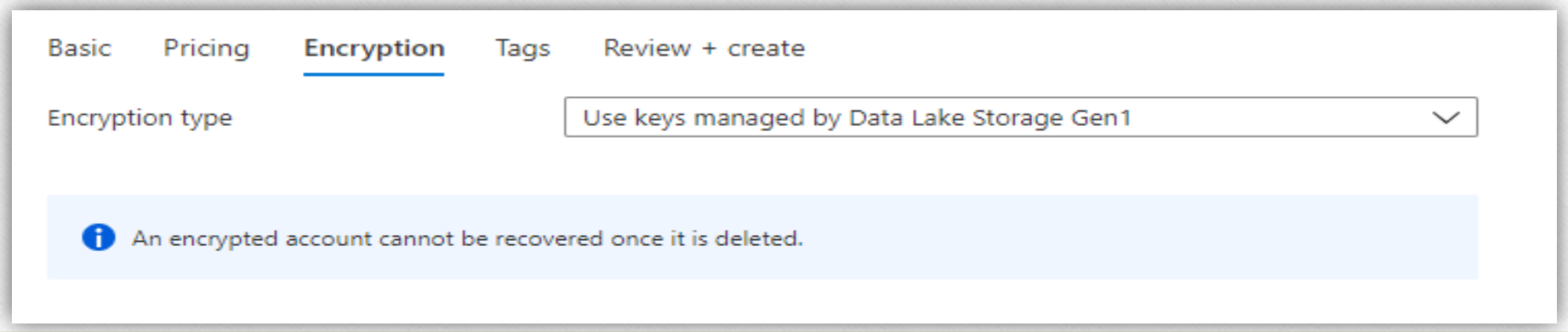


Name . Enter a unique name for the Data Lake Store account.
Subscription . Select the subscription under which you want to create a new Data Lake Store account.
Resource Group . Select an existing resource group or select the **Create new** option to create one. A resource group is a container that holds related resources for an application.
Location : Select a location where you want to create the Data Lake Store account.
Encryption settings . You can choose whether to encrypt your Data Lake Store account. If you choose to encrypt, you can also specify how to manage the master encryption key that you want to use to encrypt your account data

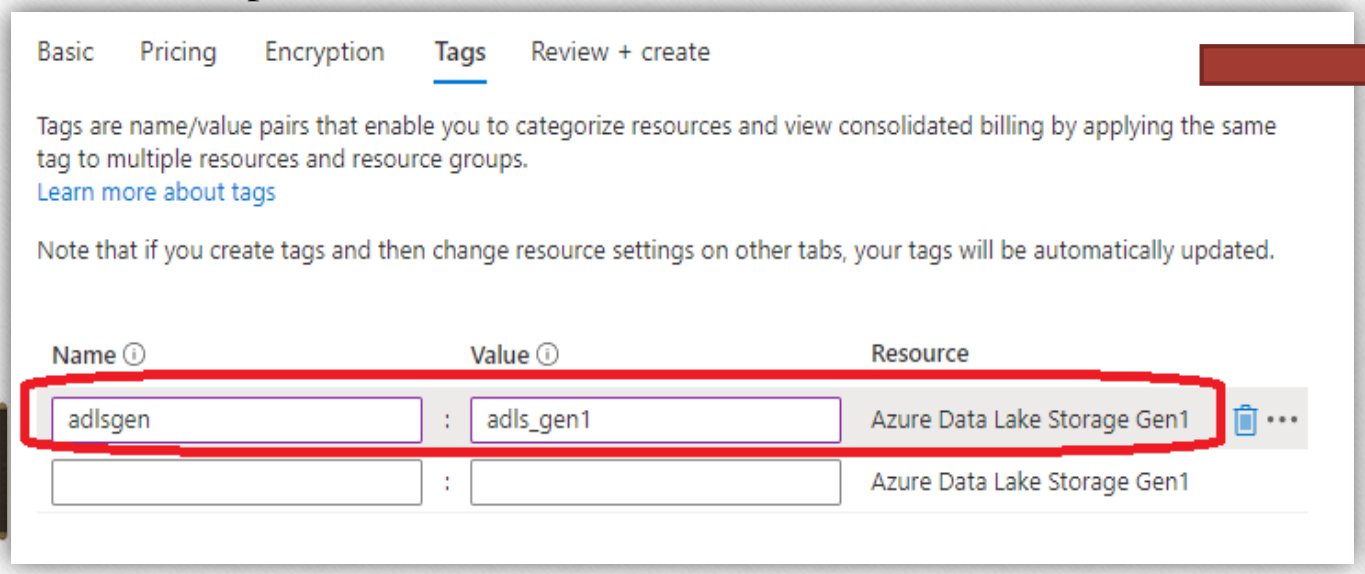


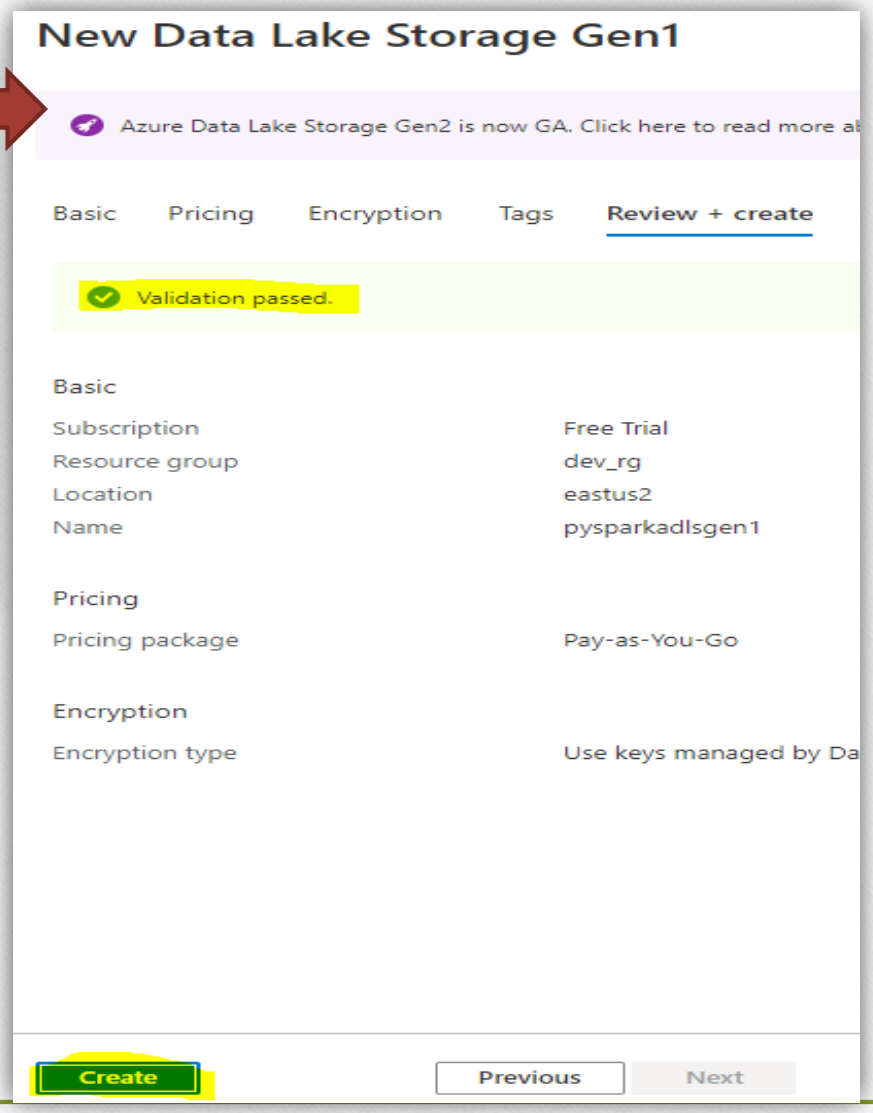
Pricing: select pricing option. 1) Pay-As-You-Go or 2) Monthly Commitment

(Optional) Select **Do not enable encryption** from the drop-down menu to not enable encryption.
(Default) Select **Use the keys managed by Azure Data Lake** if you want Azure Data Lake Store to manage your encryption keys.



Tags: Tags are names or value pair that enable you to categorize resources in azure portal For searching azure resources in portal.





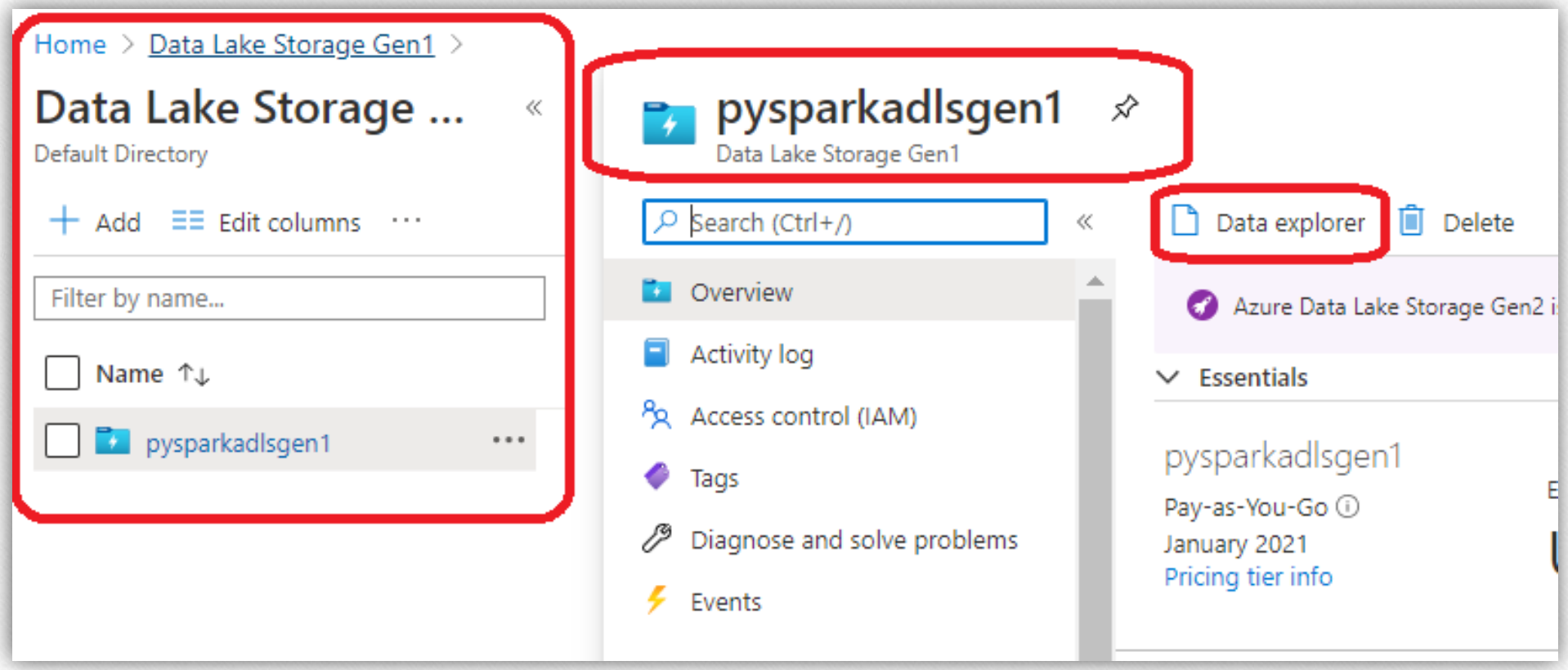
Review & Create: Finally validation all the inputs And once Validation passed we can proceed for Creating **Azure Data Lake Store Gen1.**

Create folders in your Azure Data Lake Store account

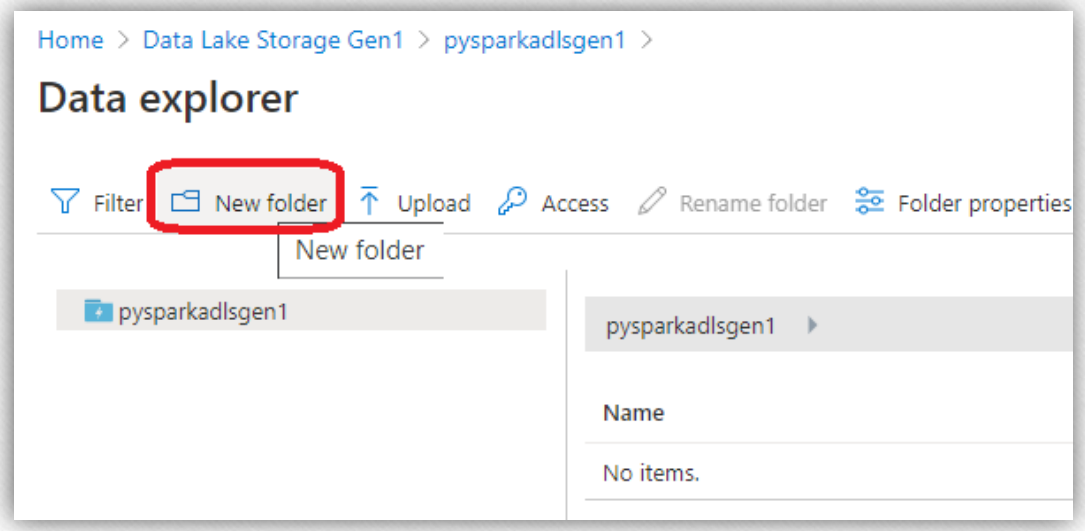
You can create folders in your Data Lake Store account to manage and store data.

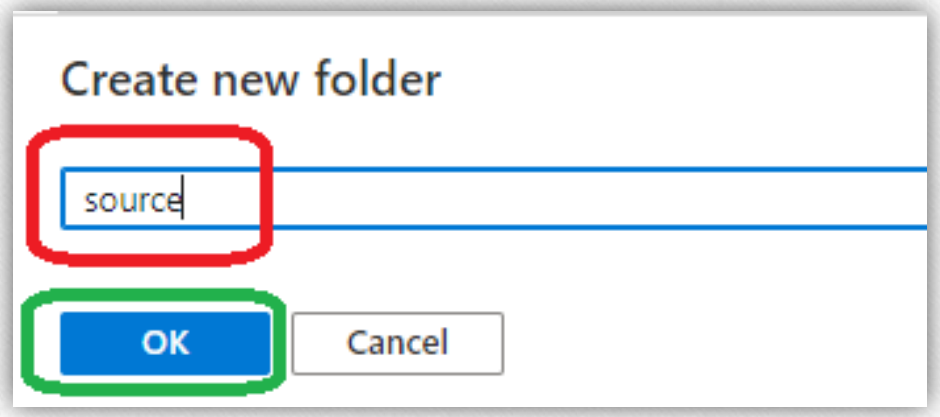
1.Open the Data Lake Store account you just created. In the left pane, click **Browse** , click **Data Lake Store,** and then, in the Data Lake Store pane, click the name of the account under which you want to create the folders. If you attached the account to the start board, click on the mosaic for that account.

2.In the Data Lake Store account dashboard, click **Data Explorer** .

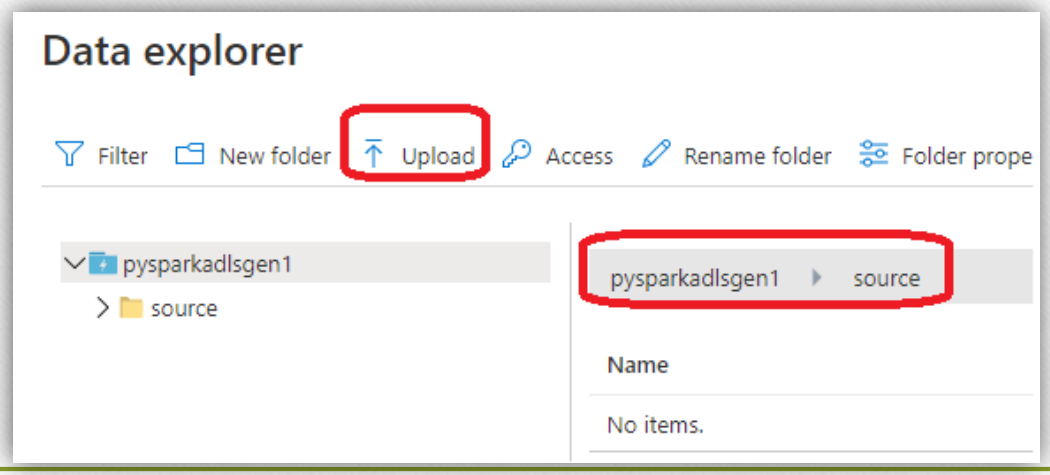


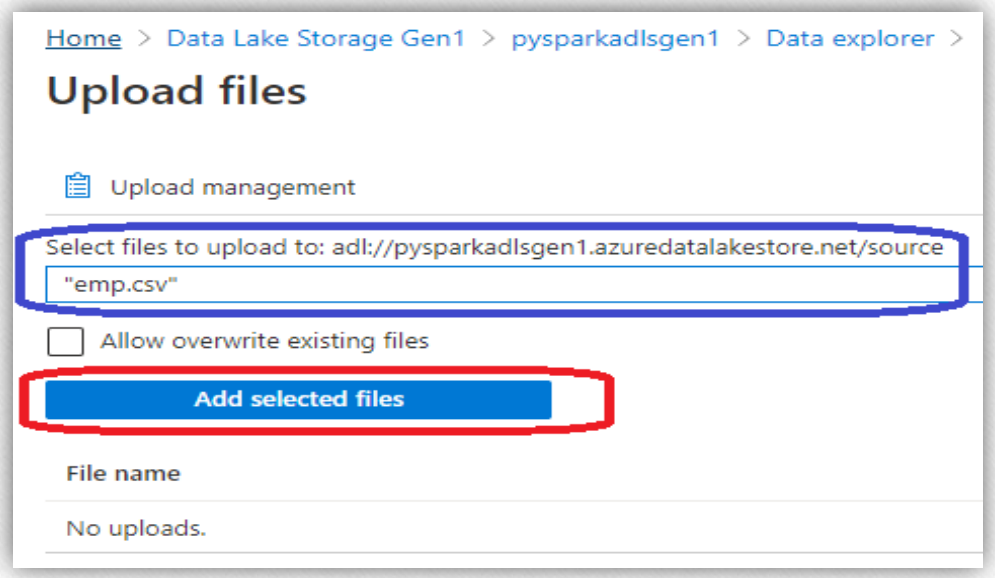
3. In the Data Lake Store account panel, click **New Folder** , enter a name for the new folder, and then click **OK** .





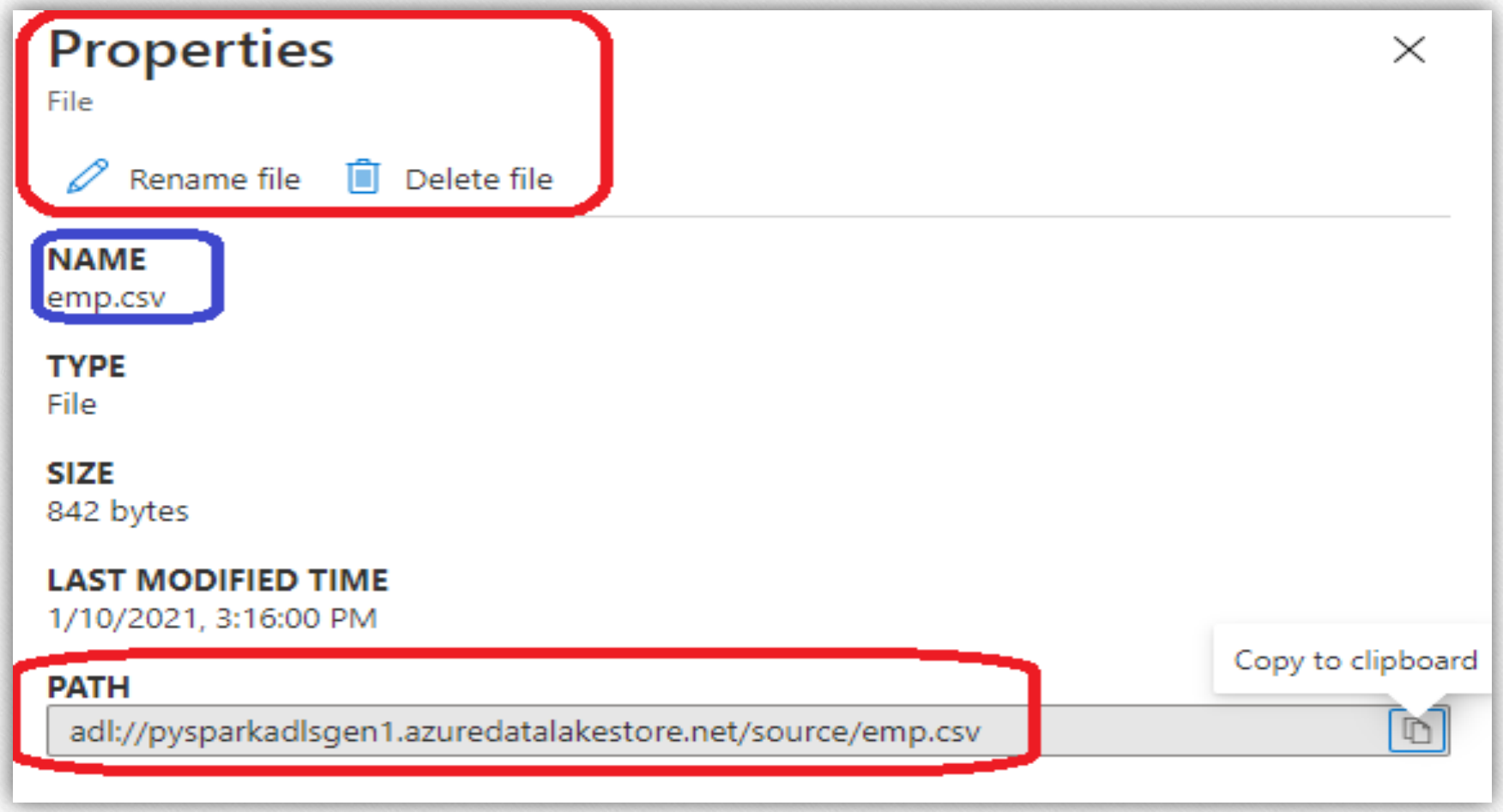
4. Upload a file manually using upload option.



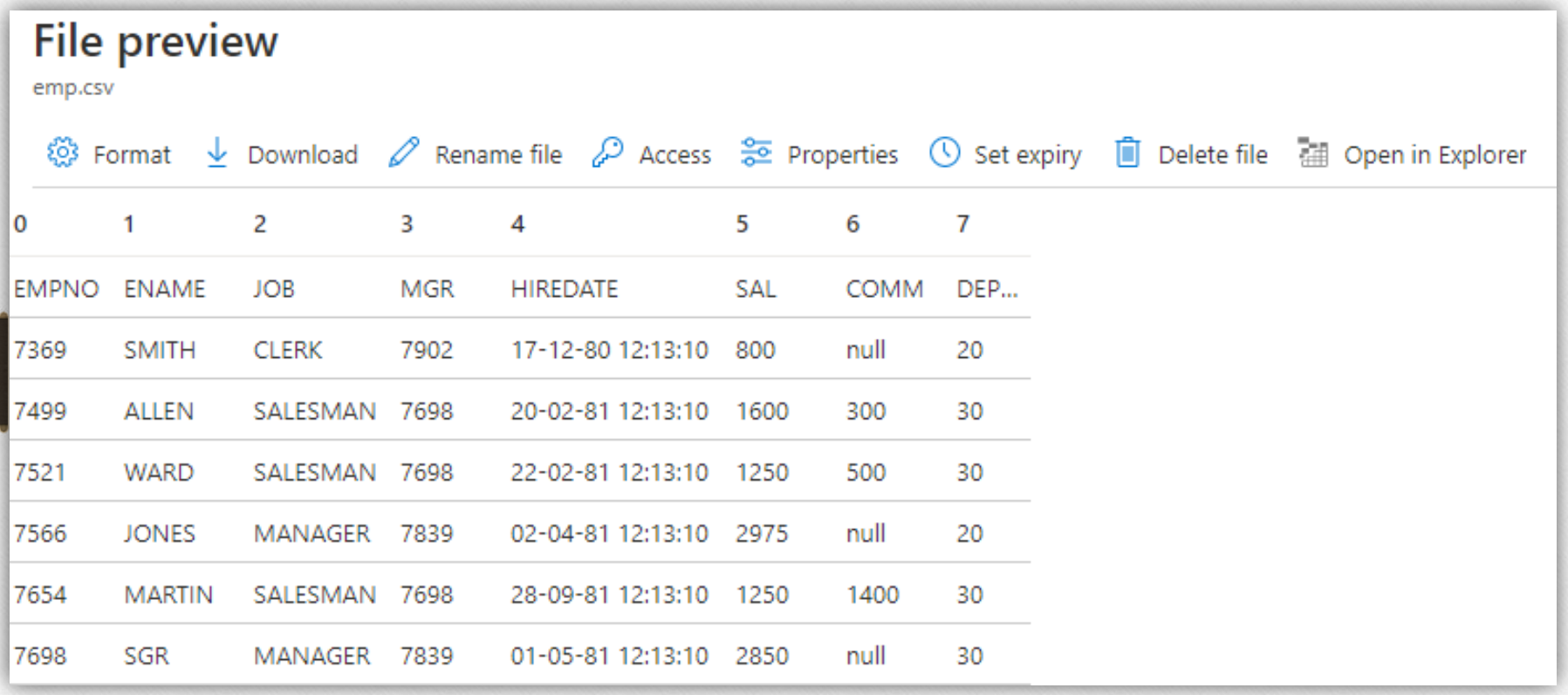


Properties and actions available on stored data

Click on the newly added file to open the **Properties** panel . The properties associated with the file and the actions you can take on the file are available in this panel. You can also copy the full path to the file in your Azure Data Lake Store account, highlighted in the red box in the screenshot below.



Click **Preview** to see a preview of the file, directly from the browser. You can also specify the preview format. Click **Preview** , click **Format** in the **File Preview** panel and, in the **File Preview Format** panel **,** specify options such as the number of lines to display, the encoding to use, the delimiter to use, etc. .



0	1	2	3	4	5	6	7
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEP...
7369	SMITH	CLERK	7902	17-12-80 12:13:10	800	null	20
7499	ALLEN	SALESMAN	7698	20-02-81 12:13:10	1600	300	30
7521	WARD	SALESMAN	7698	22-02-81 12:13:10	1250	500	30
7566	JONES	MANAGER	7839	02-04-81 12:13:10	2975	null	20
7654	MARTIN	SALESMAN	7698	28-09-81 12:13:10	1250	1400	30
7698	SGR	MANAGER	7839	01-05-81 12:13:10	2850	null	30

- Click **Download** to download the file to your computer.
- Click **Rename the file** to rename the file.
- Click **Delete file** to delete the file.

Securing data stored in Azure Data Lake Store

Securing data in Azure Data Lake Store is a three-step approach.

1.Start by creating security groups in Azure Active Directory (**AAD**). These security groups are used to implement role-based access control (**RBAC**) in Azure Portal.

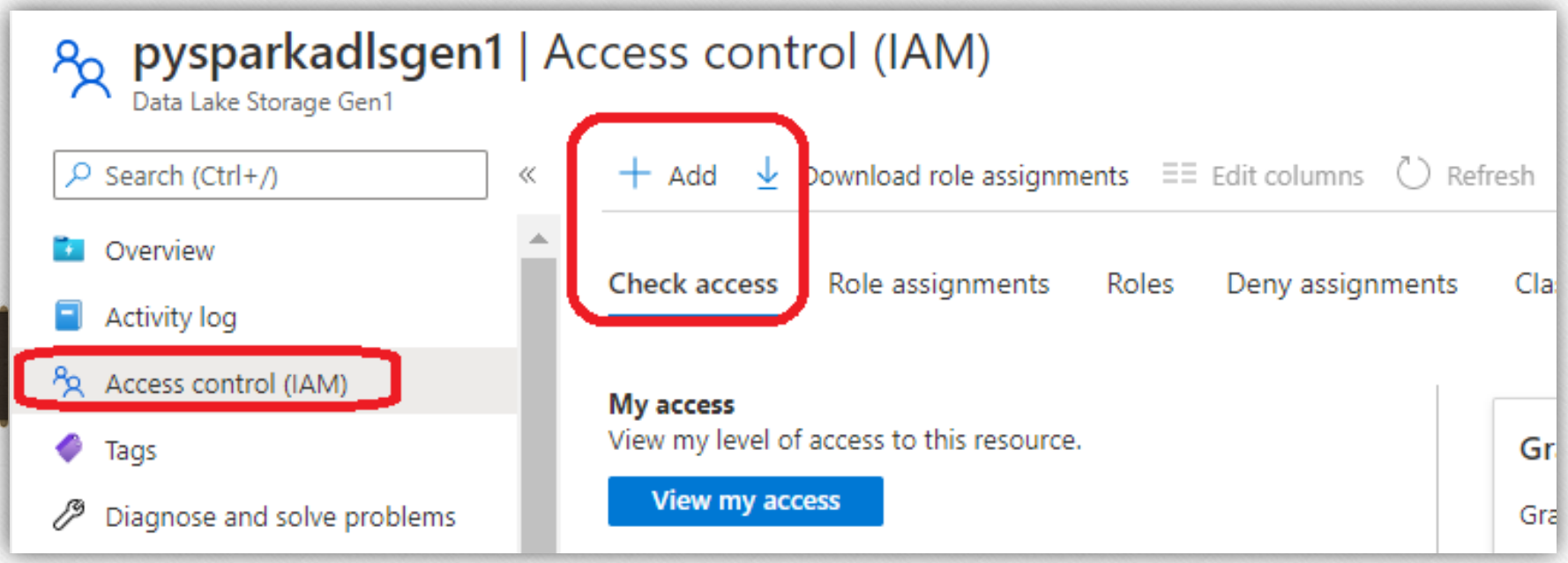
2.Assign the AAD security groups to the Azure Data Lake Store account. This controls access to the Data Lake Store account from the portal and management operations from the portal or APIs.

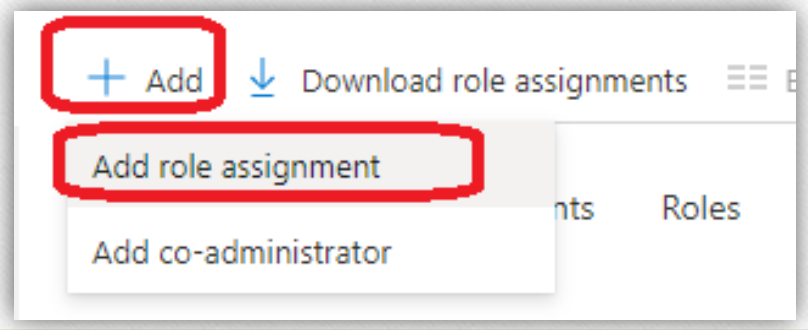
3.Assign the AAD security groups as access control lists (**ACLs**) on the Data Lake Store file system.

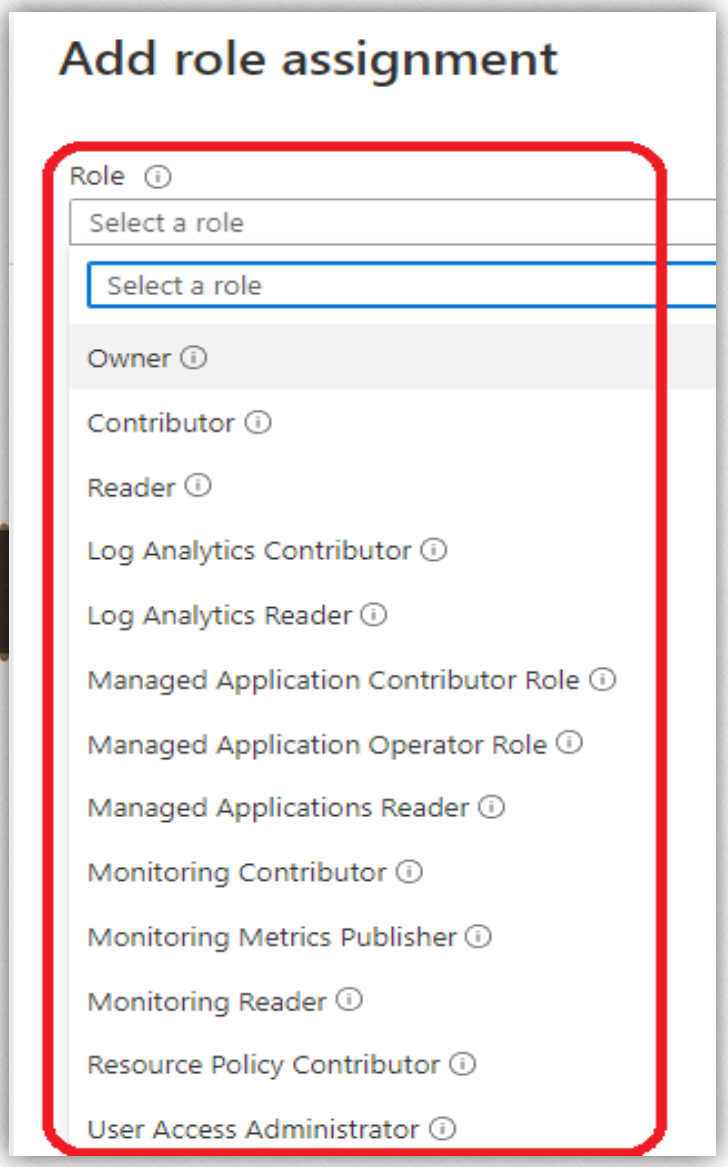
4.Additionally, you can also set an IP address range for clients that can access the data in Data Lake Store.

role-based access control (**RBAC**) in Azure Portal.

1. Select **Access Control (IAM)** from the menu on the left.
2. The Access Control panel lists all users, groups and applications with access to the resource group.





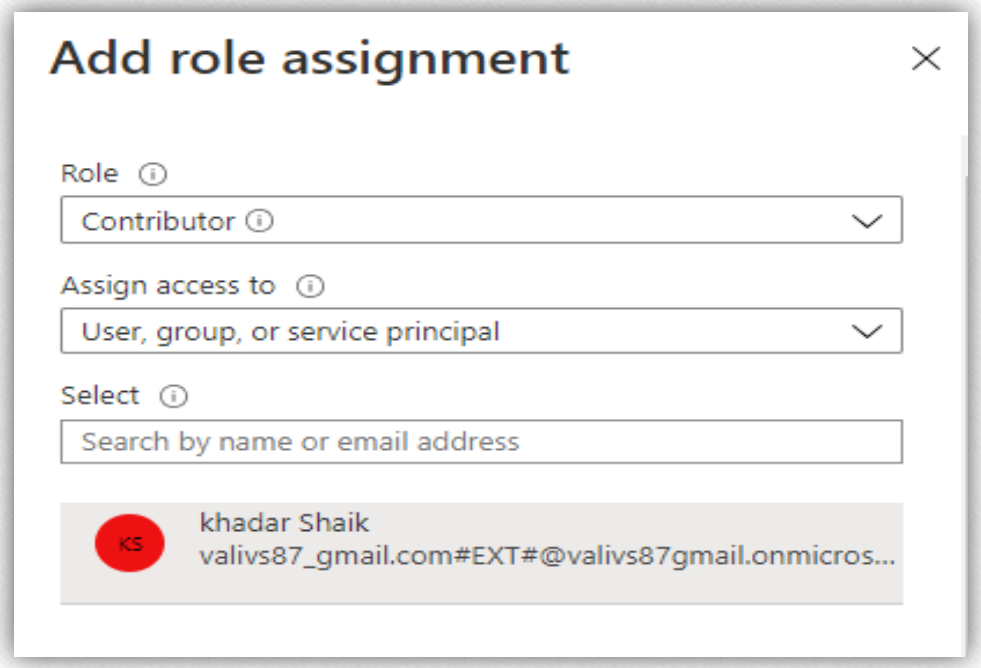


Owner: Grants full access to manage all resources, including the ability to assign roles in Azure RBAC.

Contributor: Grants full access to manage all resources, but does not allow you to assign roles in Azure RBAC, manage assignments in Azure Blueprints, or share image galleries.

Reader: View all resources, but does not allow you to make any changes.

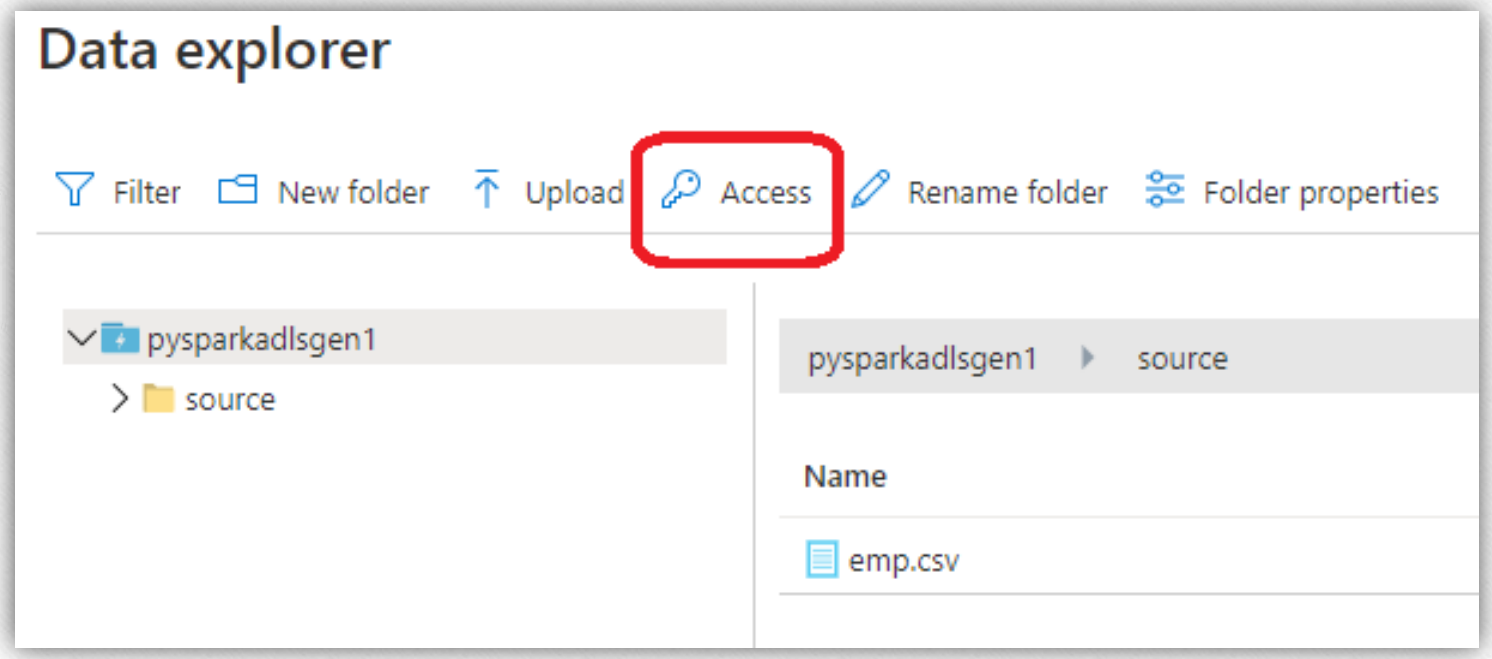
User Access Administrator : Lets you manage user access to Azure resources.



Assign users or security group as ACLs to the Azure Data Lake Store file system

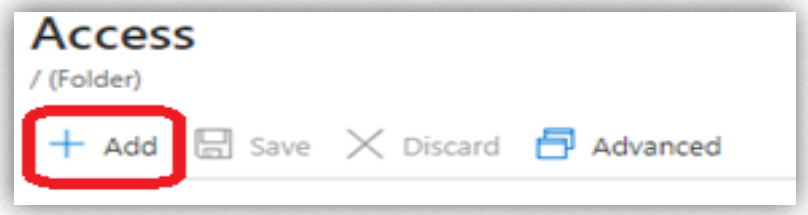
By assigning user/security groups to the Azure Data Lake file system, you set access control on the data stored in Azure Data Lake Store.

1. In your Data Lake Store account blade, click Data Explorer.
2. In the Data Explorer blade, click the root of your account, and then in your account blade, click the Access icon.

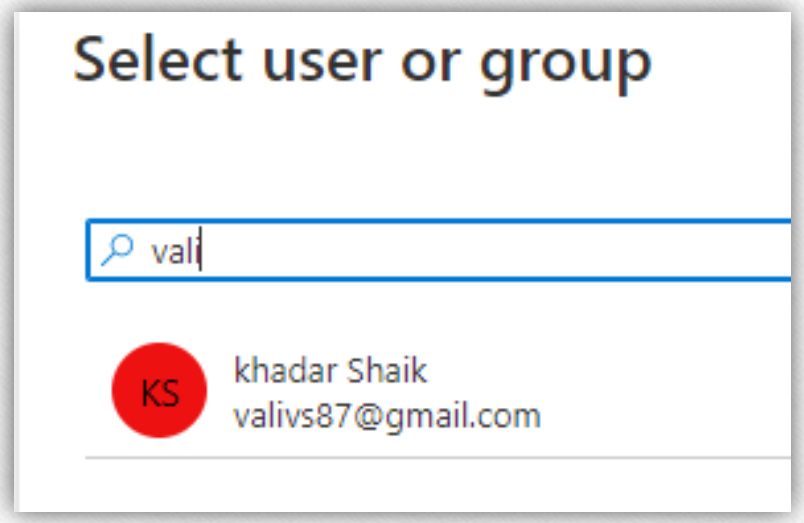


The **Access** blade lists the standard access and custom access already assigned to the root. Click the **Add** icon to add custom-level ACLs.

The **Access** blade lists the standard access and custom access already assigned to the root. Click the **Add** icon to add custom-level ACLs.



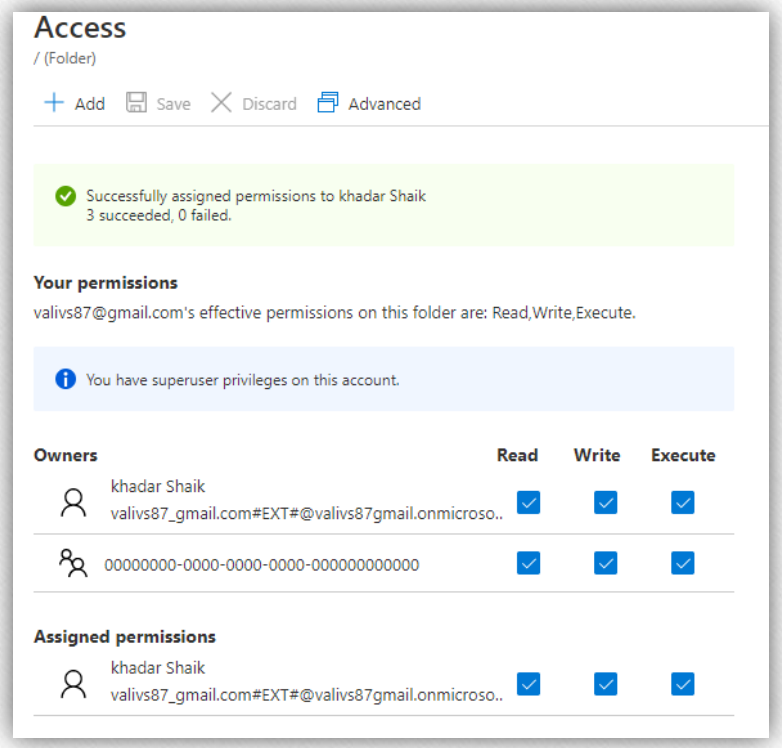
Standard access is the UNIX-style access, where you specify read, write, execute (rwx) to three distinct user classes: owner, group, and others.



Permissions

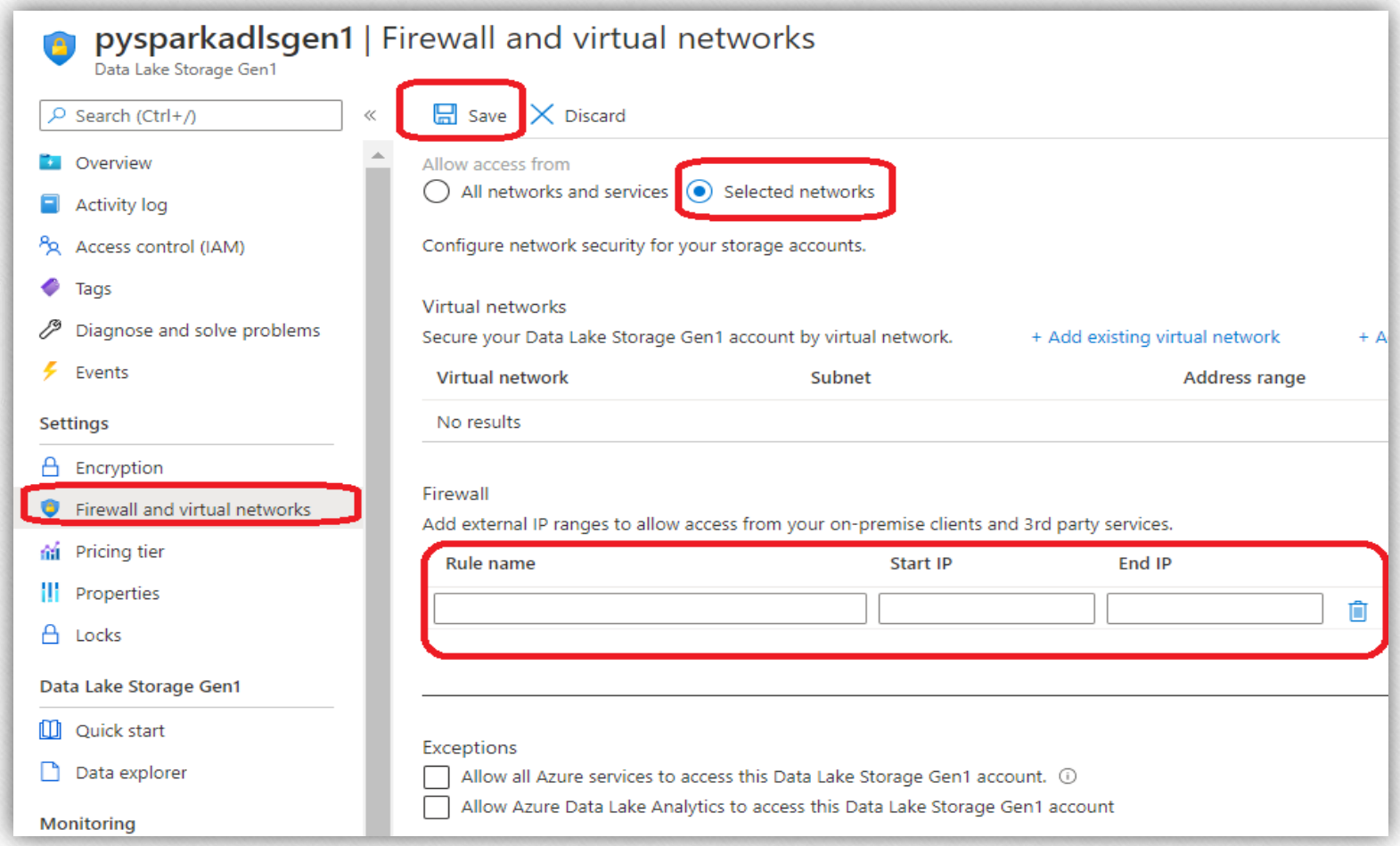
The permissions on a file system object are **Read** , **Write** and **Execute** and can be used on files and folders as shown in the table below.

	File	Folder
Reading (R)	You can edit the contents of a file	Requires **Read** and **Execute** to list the contents of the folder.
Writing (W)	You can write or add to a file	Requires **Writing and Execution** to create subordinate items in a folder.
Execution (X)	It has no meaning in the context of the Data Lake Store	It is necessary to go through the subordinate items of a folder.



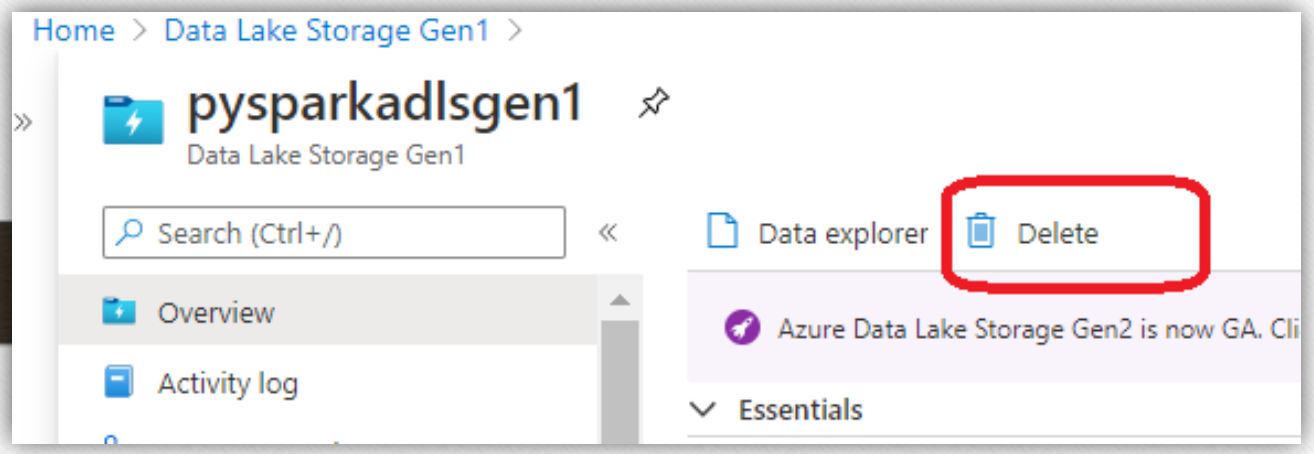
Set IP address range for data access

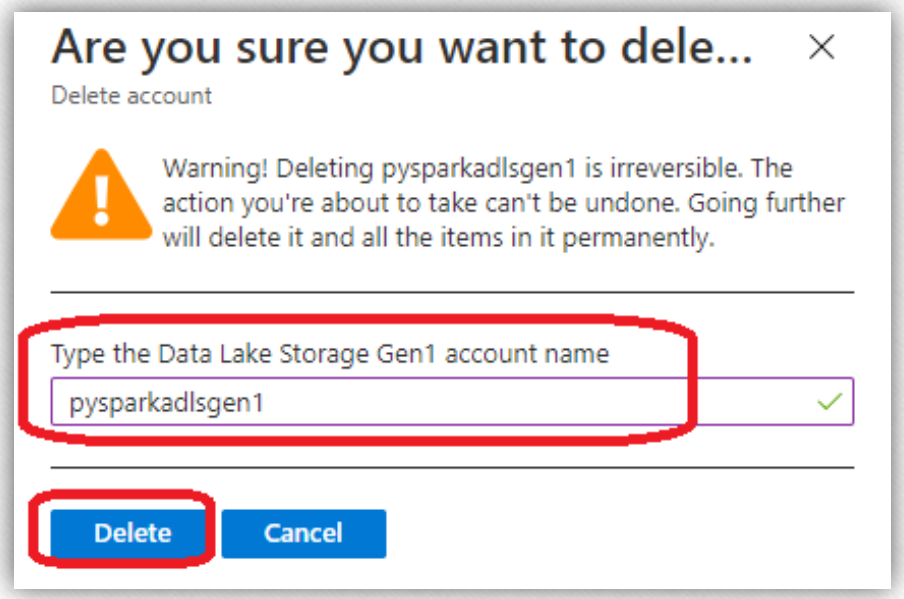
Azure Data Lake Store enables you to further lock down access to your data store at network level. You can enable firewall, specify an IP address, or define an IP address range for your trusted clients. Once enabled, only clients that have the IP addresses within defined range can connect to the store.



Delete Azure Data Lake Store account

To delete an Azure Data Lake Store account, from the Data Lake Store panel, click **Delete** . To confirm the action, you will be asked to enter the name of the account you want to delete. Enter the account name, then click **Delete** .





Azure Blob Storage (BLOB)

Blob storage

Azure Blob storage is Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of unstructured data, such as text or binary data.

- Serving images or documents directly to a browser
- Storing files for distributed access
- Streaming video and audio
- Storing data for backup and restore, disaster recovery, and archiving
- Storing data for analysis by an on-premises or Azure-hosted service

Azure Storage services

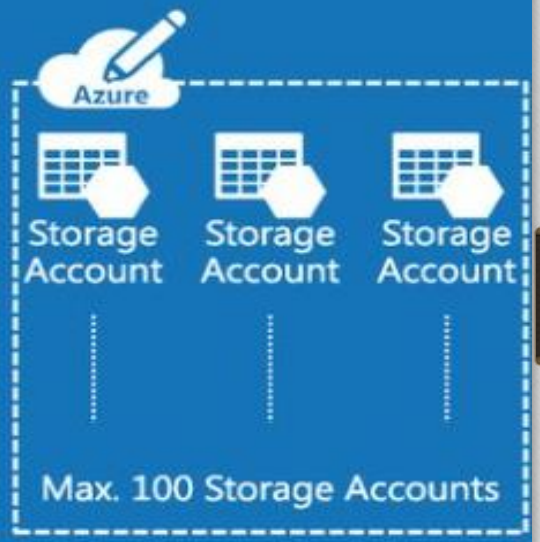
Azure Storage includes these data services:

Azure Blobs: A massively scalable object store for text and binary data.
Azure Files: Managed file shares for cloud or on-premises deployments.
Azure Queues: A messaging store for reliable messaging between application components.
Azure Tables: A NoSQL store for schema less storage of structured data.

Azure Storage Accounts (BLOB)

Azure Storage Accounts

- An Azure Subscription can have up to 100 storage accounts.
  - Soft limit – contact support if you need more
- Store up to 500TB per storage account
- A storage account is uniquely addressable.
  https://<account name>.<service name>.core.windows.net
- Available from anywhere using REST API's
- Open source client libraries available for .NET, Native C++, Java, Android, Node.js, PHP, Python, Ruby, PowerShell, iOS



Azure Storage Account Replication

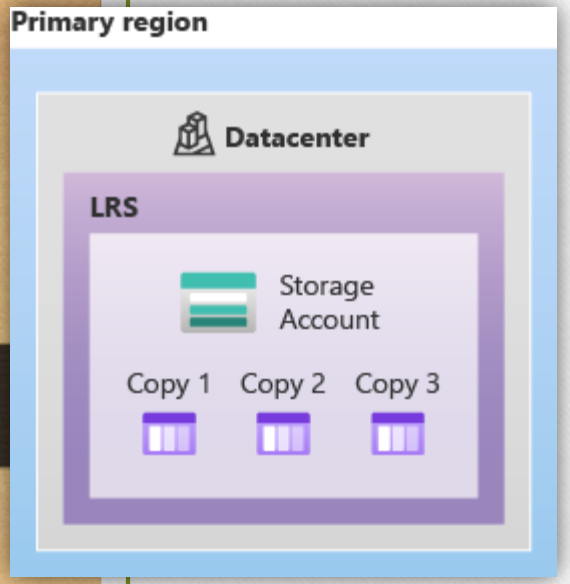
Storage Replication

Replication Strategy	LRS	ZRS*	GRS	RA-GRS
Data is replicated across multiple facilities	No	Yes	Yes	Yes
Data can be read from the secondary location as well as from the primary location	No	No	No	Yes
Number of copies of data maintained on separate nodes	3	3	6	6

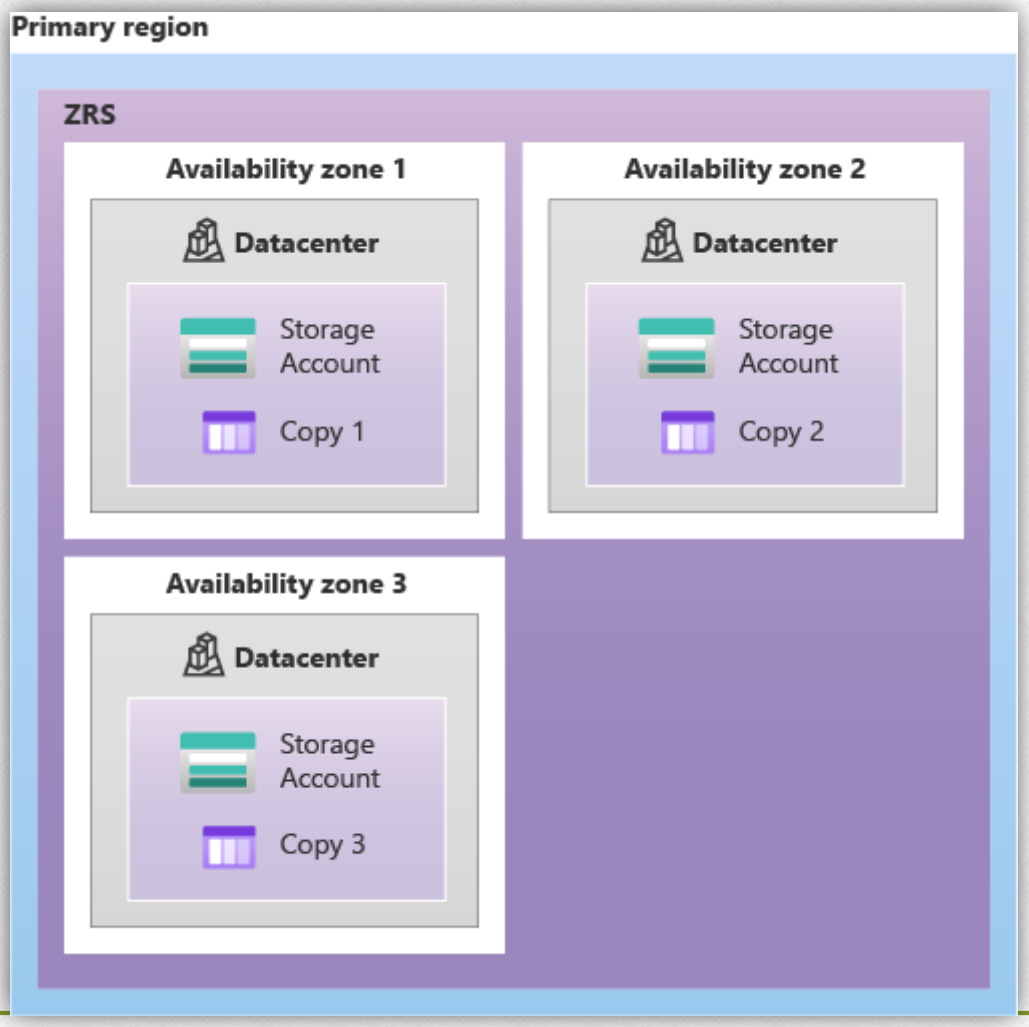
LRS – Locally Redundant
*ZRS - Zone Redundant (only for Block Blob)
GRS – Globally Redundant
RA-GRS – Read Access Geo Redundant

Azure Storage Account Replication

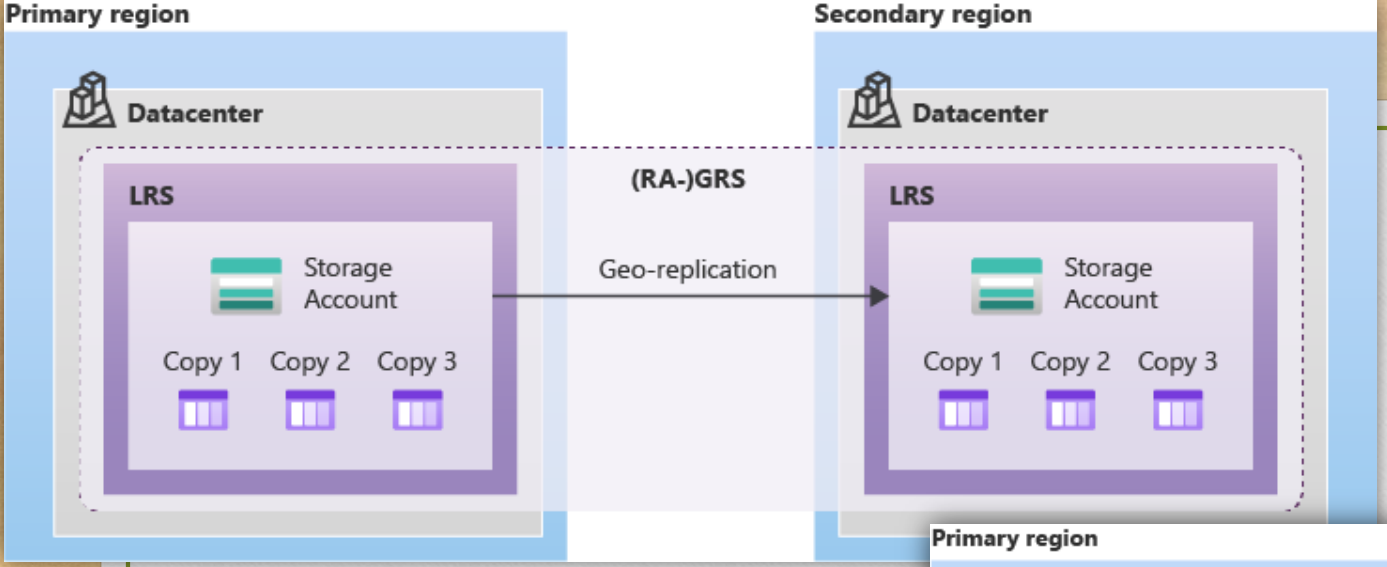
LRS

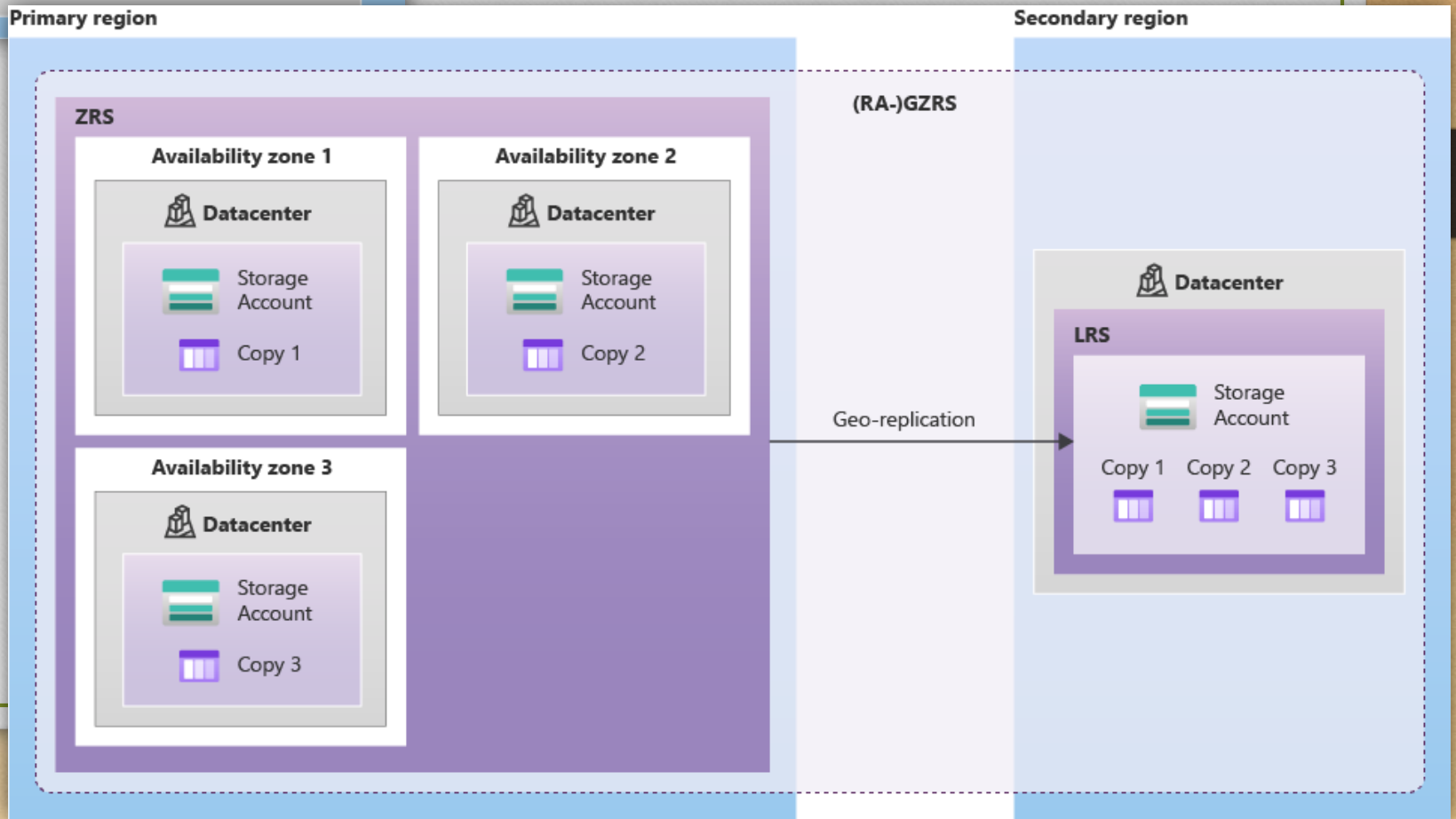


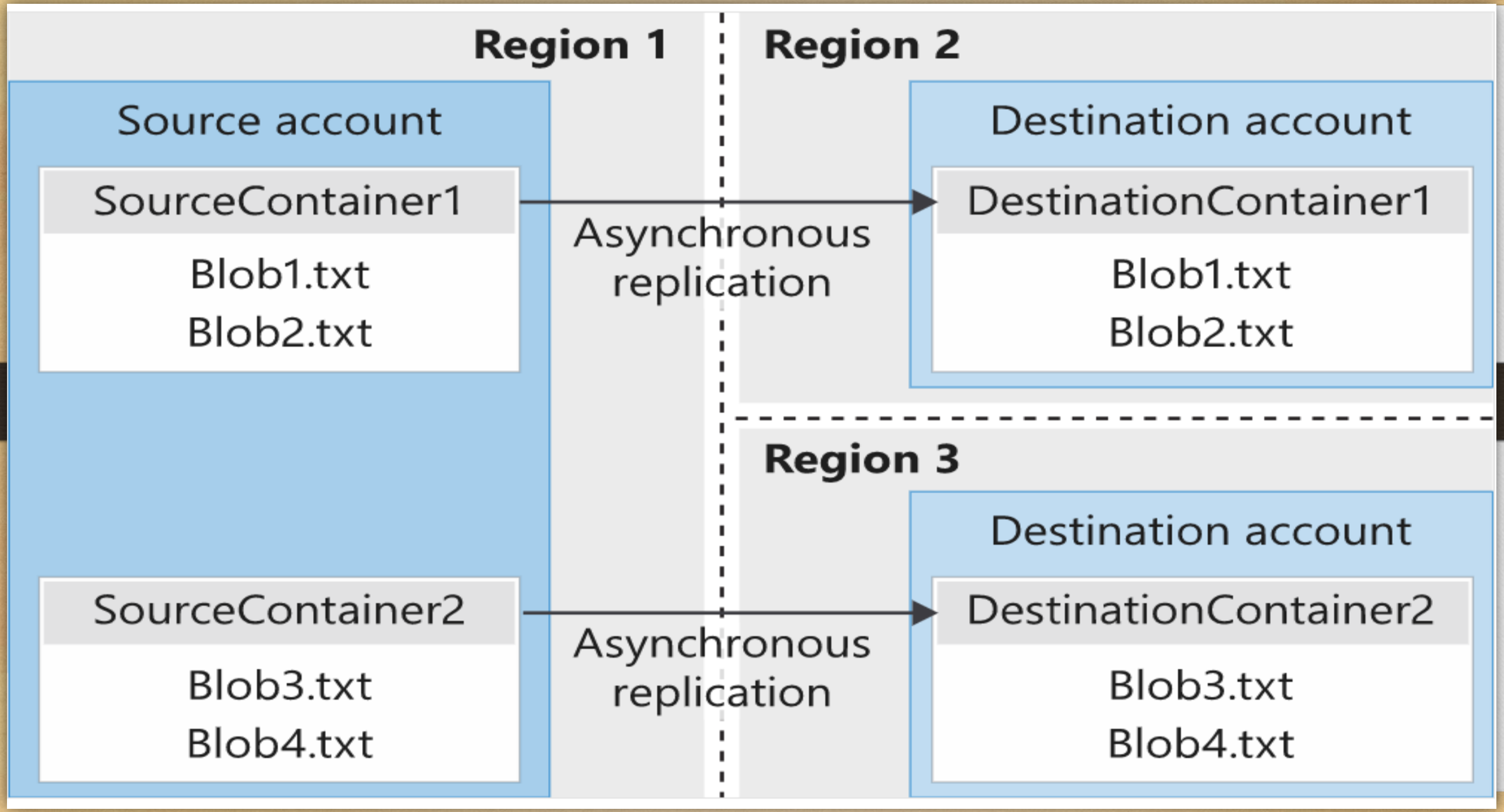
Zone-redundant storage



Azure Storage Account Replication






Region 1
Region 2
Source account
SourceContainer1
Blob1.txt
Blob2.txt
Asynchronous replication
Destination account
DestinationContainer1
Blob1.txt
Blob2.txt
Region 3
SourceContainer2
Blob3.txt
Blob4.txt
Asynchronous replication
Destination account
DestinationContainer2
Blob3.txt
Blob4.txt

Storage Service Types

Azure Storage Account Services

Blobs

Block: Text or binary data (.log, .exe, .jpg, etc.).

Up to 200GB.

Page: Optimized for disks (.vhd). Supports random read-write. Up to 1TB.

Append Blob: Writes to end of the blob (4MB max) up to 50k times (~195GB)

File Shares

Supports SMB 3.0 protocol.

Can be accessed like a traditional file share.

Share files between multiple Virtual Machines.

A single file share can be up to 5TB.

Tables

NoSQL storage of structured data (entities).

Key/value storage.

A single entity can have up to 255 properties and be up to 1MB.

Queues

Durable messaging.

Provides asynchronous communication between application tiers and components.

A single message can be up to 64KB.

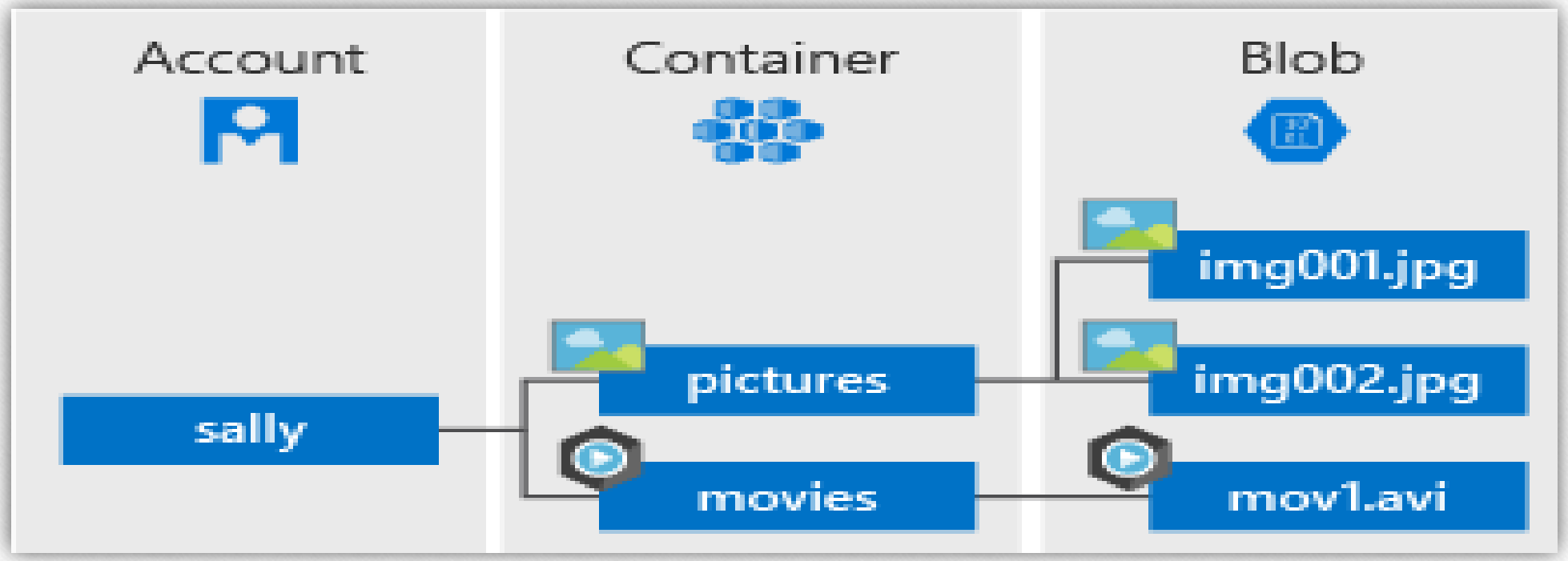
Blob Storage resources

Blob storage resources

Blob storage offers three types of resources:

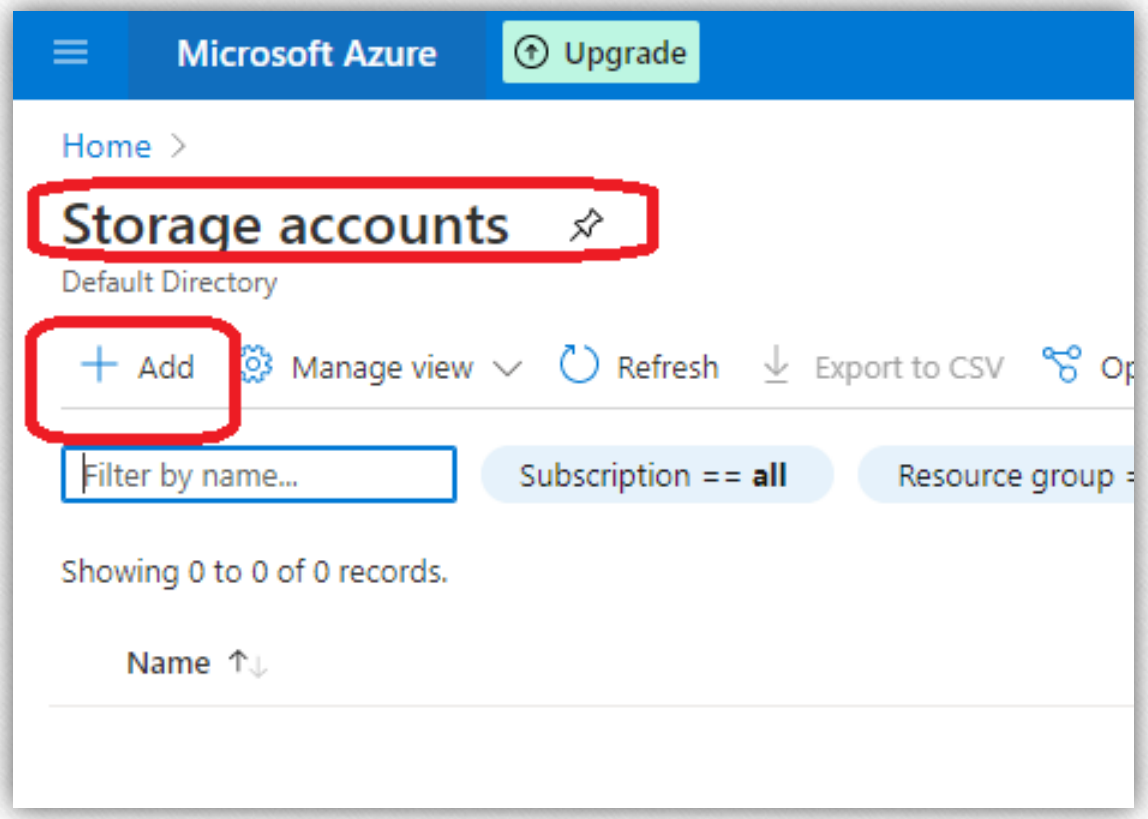
- The storage account.
- A container in the storage account
- A blob in a container

The following diagram shows the relationship between these resources.



Create a Blob Storage account through the Azure portal

1. Sign in to the azure porta.
2. From the Hub menu, select **New** > **Data + Storage** > **Storage Account** .
3. Enter a name for the Storage account.
4. This name must be globally unique; is used as part of the URL used to access objects in the storage account.



Home > Storage accounts >

Create storage account

Basics | Networking | Data protection | Advanced | Tags | Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below.
Learn more about Azure storage accounts

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Free Trial

Resource group * : dev_rg

Create new

Instance details

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. Choose classic deployment model

Storage account name * : pysparkstorage

Location * : (US) East US

Performance : Standard / Premium

Account kind : StorageV2 (general purpose v2)

Replication : Read-access geo-redundant storage (RA-GRS)

Review + create | < Previous | Next : Networking >

1.Select the subscription for which you want to create the new Storage account.
2.Specify a new resource group or select an existing resource group
3.Enter the Storage Account Name and it should unique name in entire cloud.
4.Select the Location for your storage account
5.From the Account Type(Performance)
select **Blob Storage** .This is where you can select the type of storage account. Tiered storage is not available for general purpose storage; it is only available in the Blob Storage type account. Note that when you select this option, the performance layer is set to Standard. Tiered storage is not available with the Premium performance tier.
6.Select the replication option for the Storage account: **LRS** ,**ZRS**, **GRS** or **RA-GRS** .
The default is **RA-GRS** .
LRS = locally redundant storage
ZRS = Zone Redundant Storage
GRS = geo-redundant storage (2 regions)
RA-GRS is georedundant storage with read access (2 regions with read access for the second).

Basics | **Networking** | Data protection | Advanced | Tags | Review + create

Network connectivity

You can connect to your storage account either publicly, via public IP addresses or service endpoints, or privately, using a private endpoint.

Connectivity method *

- (•) Public endpoint (all networks)
- ( ) Public endpoint (selected networks)
- ( ) Private endpoint

ⓘ All networks will be able to access this storage account.
Learn more about connectivity methods

Network routing

Determine how to route your traffic as it travels from the source to its Azure endpoint. Microsoft network routing is recommended for most customers.

Routing preference * ⓘ

- (•) Microsoft network routing (default)
- ( ) Internet routing

Basics | Networking | **Data protection** | Advanced | Tags | Review + create

Recovery

☐ **Turn on point-in-time restore for containers**

Use point-in-time restore to restore one or more containers to an earlier state. If point-in-time restore is enabled, then versioning, change feed, and blob soft delete must also be enabled. Learn more

☐ **Turn on soft delete for blobs**

Soft delete enables you to recover blobs that were previously marked for deletion, including blobs that were overwritten. Learn more

☐ **Turn on soft delete for containers**

Soft delete enables you to recover containers that were previously marked for deletion. Learn more

Sign up is required on a per-subscription basis to use container soft delete. Sign up for container soft delete

☐ **Turn on soft delete for file shares**

Soft delete enables you to recover file shares that were previously marked for deletion. Learn more

Tracking

☐ **Turn on versioning for blobs**

Use versioning to automatically maintain previous versions of your blobs for recovery and restoration. Learn more

☐ **Turn on blob change feed**

Keep track of create, modification, and delete changes to blobs in your account. Learn more

Basics | Networking | Data protection | **Advanced** | Tags | Review + create

Security

Secure transfer required ⓘ — ○ Disabled ◉ Enabled

Minimum TLS version ⓘ — Version 1.0

Infrastructure encryption ⓘ — ◉ Disabled ○ Enabled

ⓘ Sign up is currently required to enable infrastructure encryption on a per-subscription basis. Sign up for infrastructure encryption

Blob storage

Allow Blob public access ⓘ — ○ Disabled ◉ Enabled

Blob access tier (default) ⓘ — ○ Cool ◉ Hot

NFS v3 ⓘ — ◉ Disabled ○ Enabled

ⓘ Sign up is currently required to utilize the NFS v3 feature on a per-subscription basis. Sign up for NFS v3

Data Lake Storage Gen2

Hierarchical namespace ⓘ — ◉ Disabled ○ Enabled

Azure Files

Large file shares ⓘ — ◉ Disabled ○ Enabled

Tables and Queues

Customer-managed keys support ⓘ — ◉ Disabled ○ Enabled

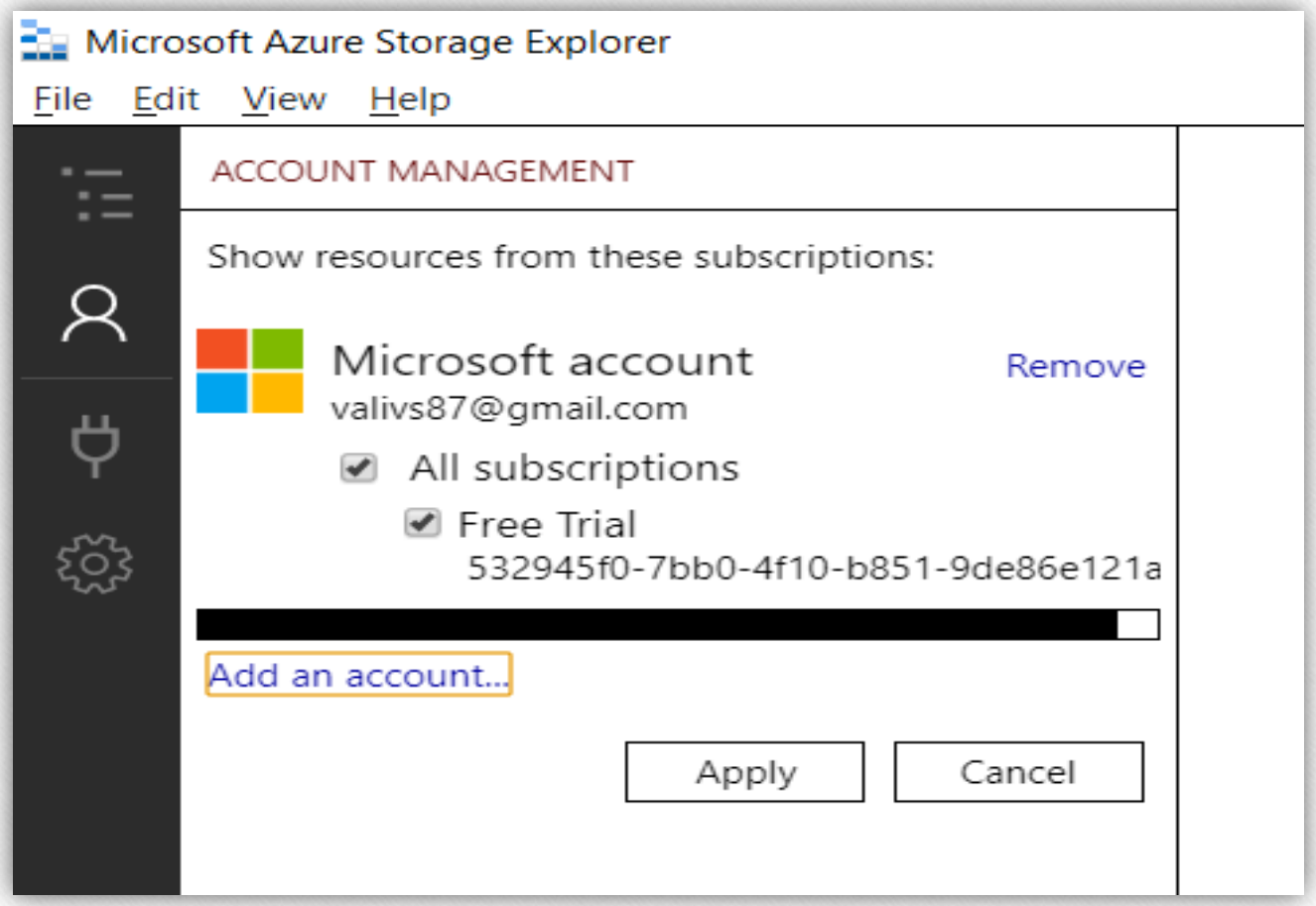
ⓘ Sign up is currently required to enable customer-managed keys support for tables and queues on a per-subscription basis. Sign up for CMK support

Diff Between ADLS & BLOB

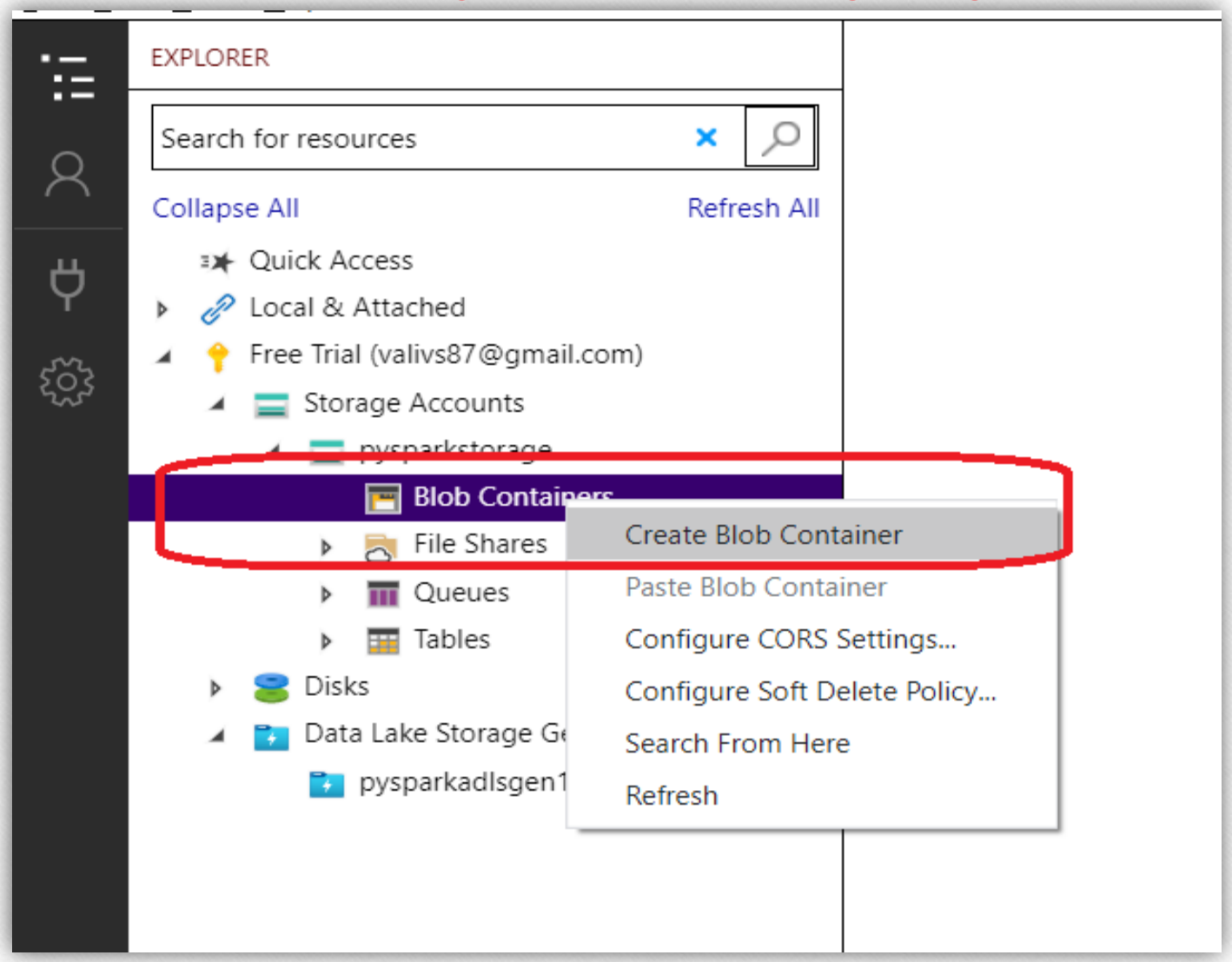
USAGE	AZURE DATA LAKE	AZURE BLOB STORAGE
Structure	Hierarchical file system	Object store with flat namespace
Purpose	Optimized storage for big data analytics workloads	General purpose object store for a wide variety of storage, including big data analytics
API	REST API over HTTPS	REST API over HTTP/HTTPS
Server-side API	WebHDFS-compatible REST API	Azure Blob Storage REST API
Authentication	Based on Azure Active Directory Identities	Based on shared secrets - Account Access Keys and Shared Access Keys.
Size limits	No limits on account sizes, file sizes or number of files	Specific limits for containers sizes and files in blob. Please refre ms document.
Sanpshots	No snapshot options.	You can create no of snapshots in blob.

Install Storage Explorer in windows to manage Storage

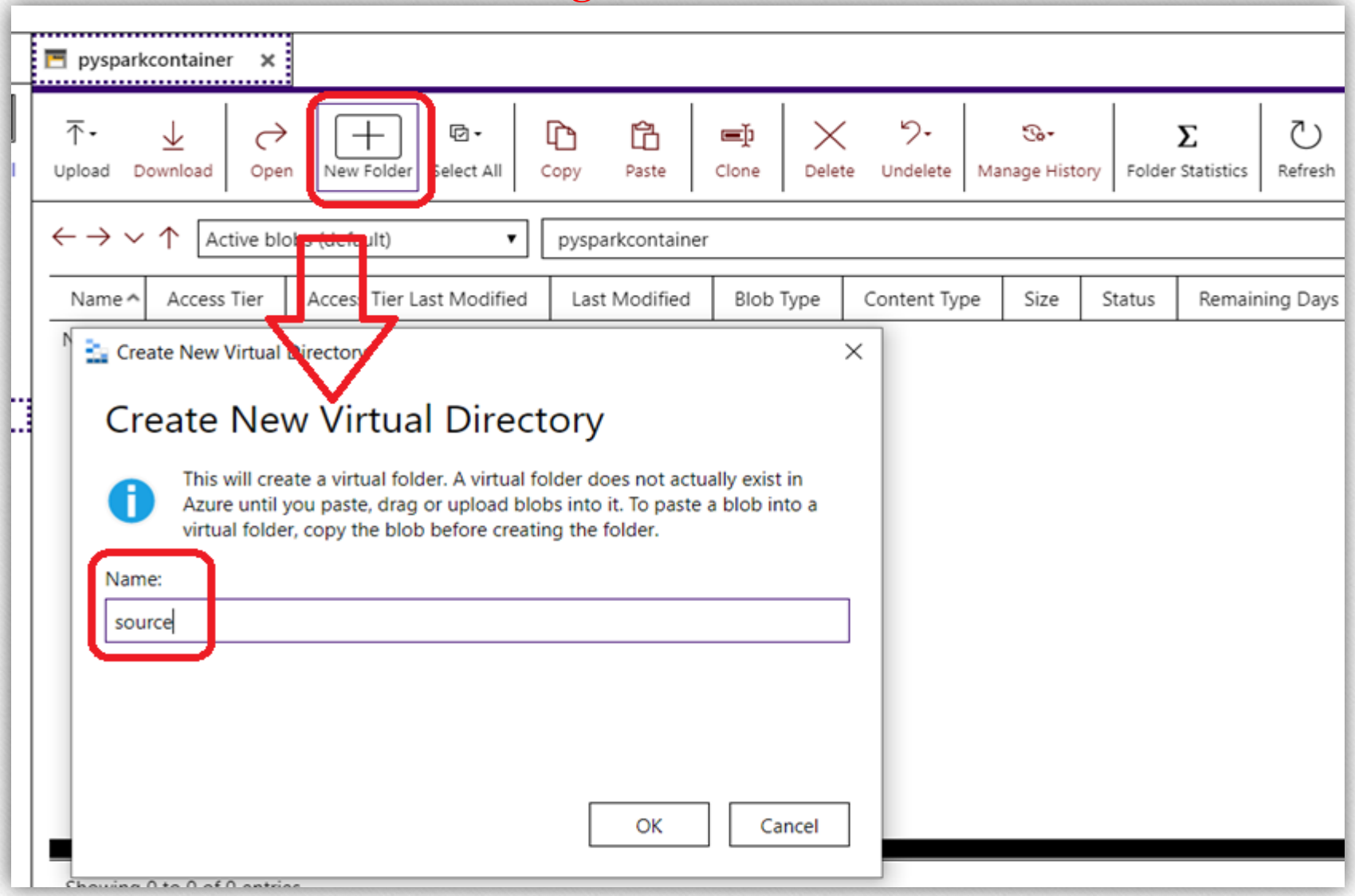
Login with Azure portal credentials and select your subscription to access all storage Accounts and data lakes storages and manage storages.



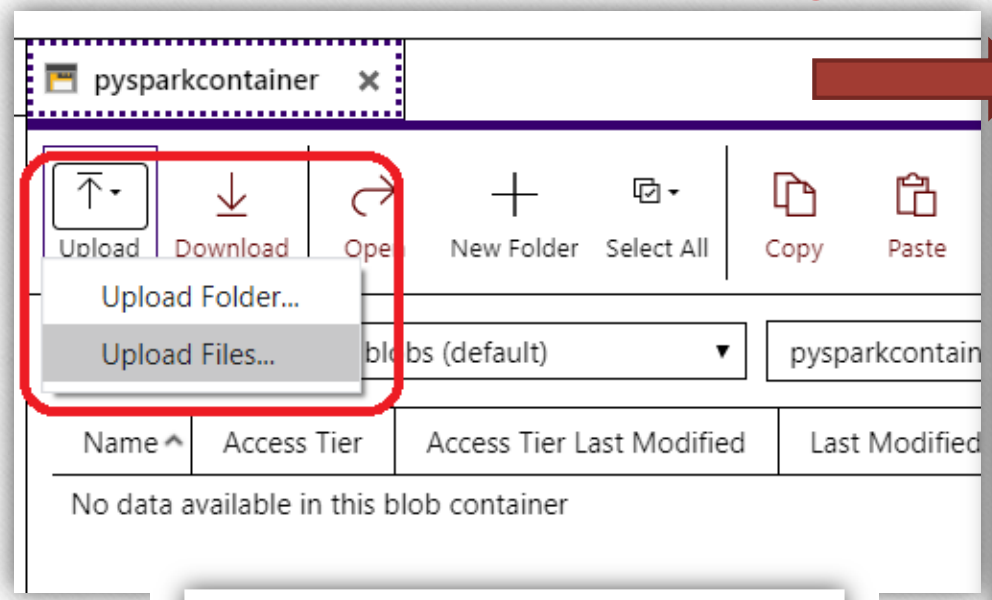
Creating Blob Container using Storage Explorer

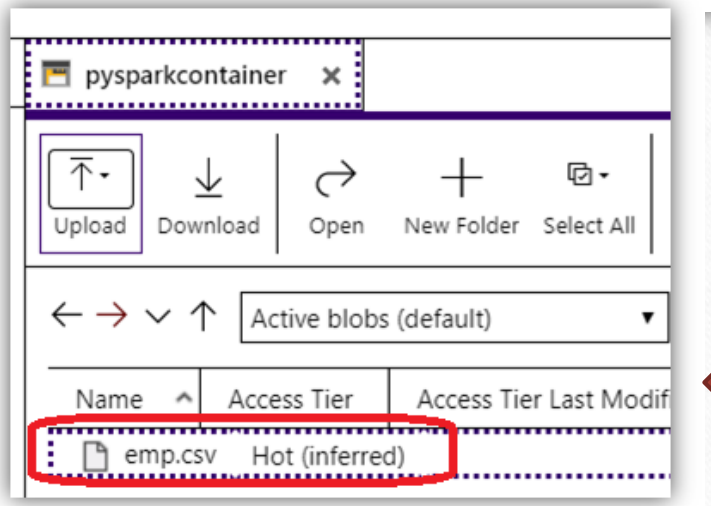


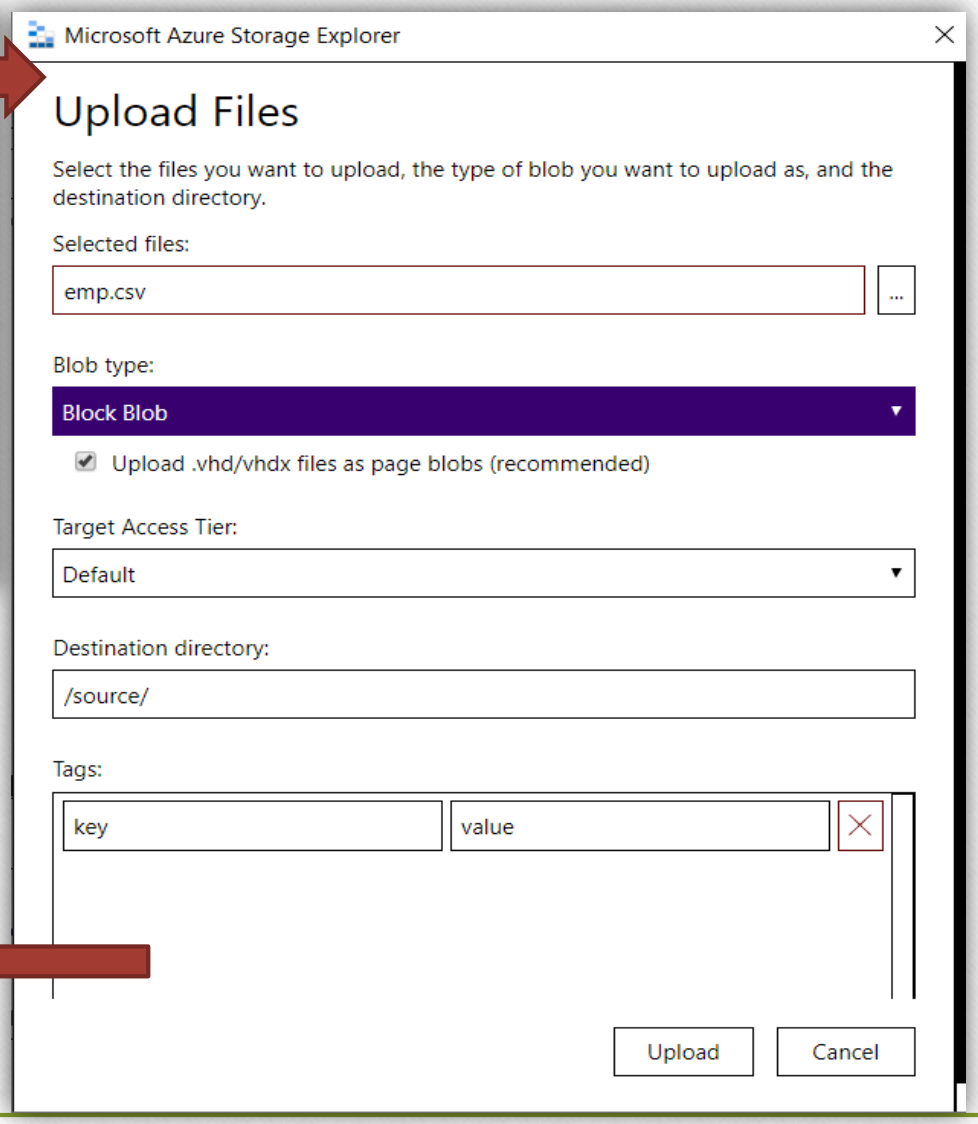
Creating Folder in Container



Uploading files into folder







AZURE ADF V2 Features

1) Execute **SSIS** Package
2) **Integration Runtime** (IR) – (Azure, Self – Hosted, Azure-SSIS)
3) **Control Flow** (branching, looping and conditional processing)
4) **Linked Services**
5) **Triggers** For Pipeline Scheduling
6) **OMS** Monitoring (Operational Monitoring Suite)
7) **Data Flow**
8) **Expressions** and **Functions**
9) **Parameterized** Pipelines
10) **on-demand Spark linked** service
11) Support for more **data stores**
12) **Invoking a pipeline** from another pipeline (Execute Pipeline Activity)
13) **Custom state passing** (current activity getting previous activity output @activity('NameofPreviousActivity').output.value)
14) **Incremental Load** Flow (**Delta Load Flow** – based on watermark)
15) V2 Supports to execute **Custom Activity** for .Net script or executable.
16) **Git Repository & Azure Devops** Integration.
17) **ARM Templates** for deploying Azure ADF to different environments
18) **Key Vault** Integration
19) Scale-out with **on-demand processing power**
20) **Debugging** Pipelines to identify errors and view the data view.
21) **System Variables** (@pipeline().DataFactory, @trigger().ScheduleTime)
22) **COPY Activity** (Dynamically Creation Pipe lines, data sets)

Code-Free ETL as a Service

INGEST

- Multi-cloud and on-prem hybrid copy data
- 90+ native connectors
- Serverless and auto-scale
- Use wizard for quick copy jobs

CONTROL FLOW

- Design code-free data pipelines
- Generate pipelines via SDK
- Utilize workflow constructs: loops, branches, conditional execution, variables, parameters, ...

DATA FLOW

- Code-free data transformations that execute in Spark
- Scale-out with Azure Integration Runtimes
- Generate data flows via SDK
- Designers for data engineers and data analysts

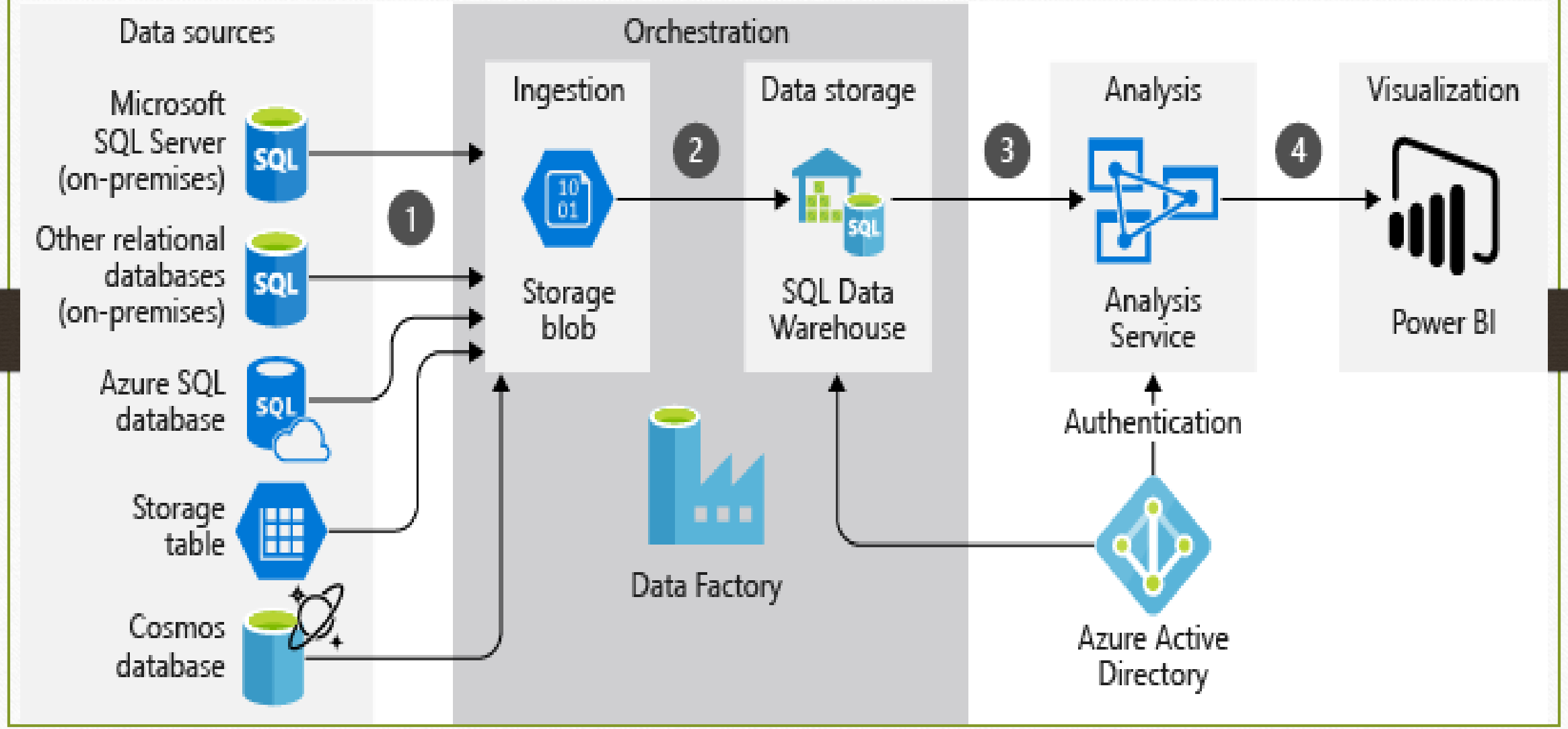
SCHEDULE

- Build and maintain operational schedules for your data pipelines
- Wall clock, event-based, tumbling windows, chained

MONITOR

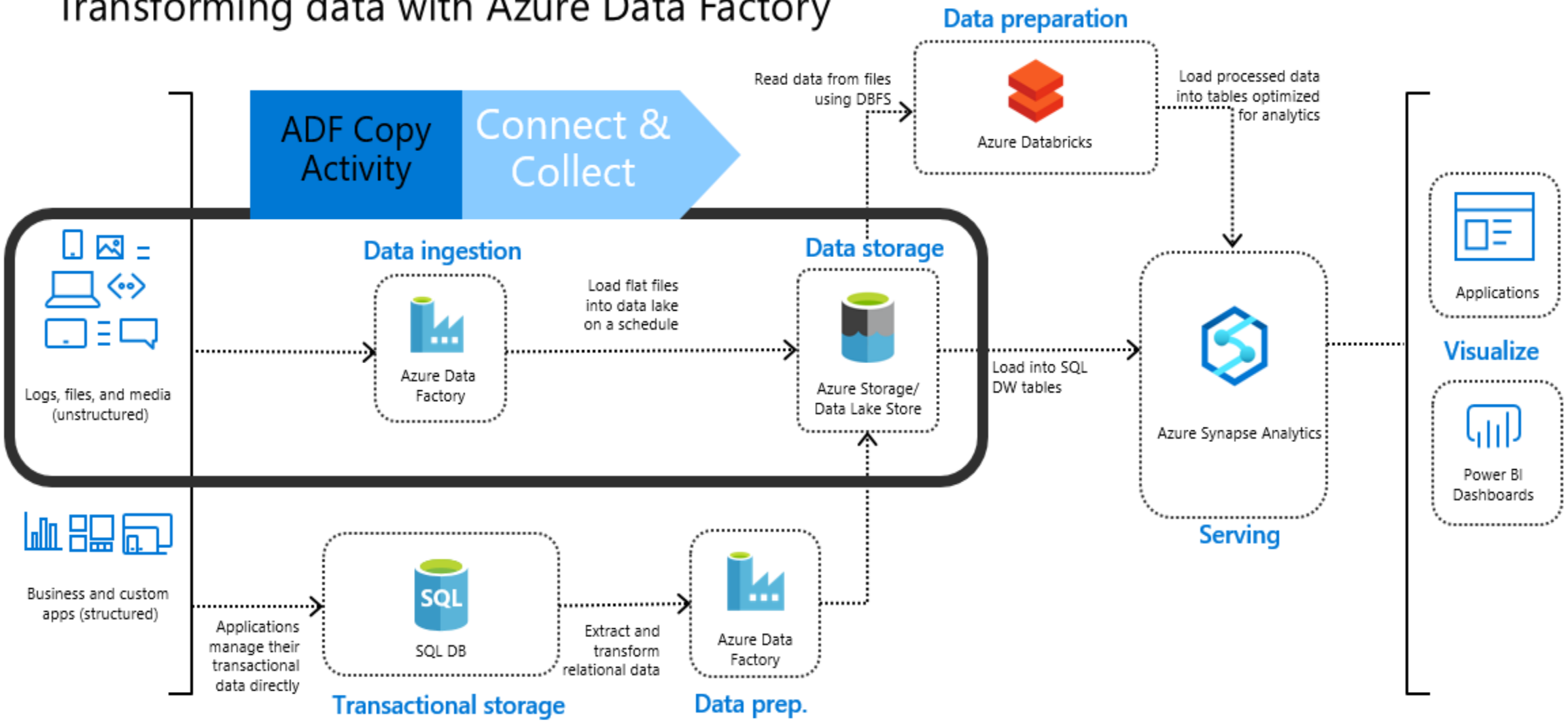
- View active executions and pipeline history
- Detail activity and data flow executions
- Establish alerts and notifications

ADF Architecture Data Flow Diagram

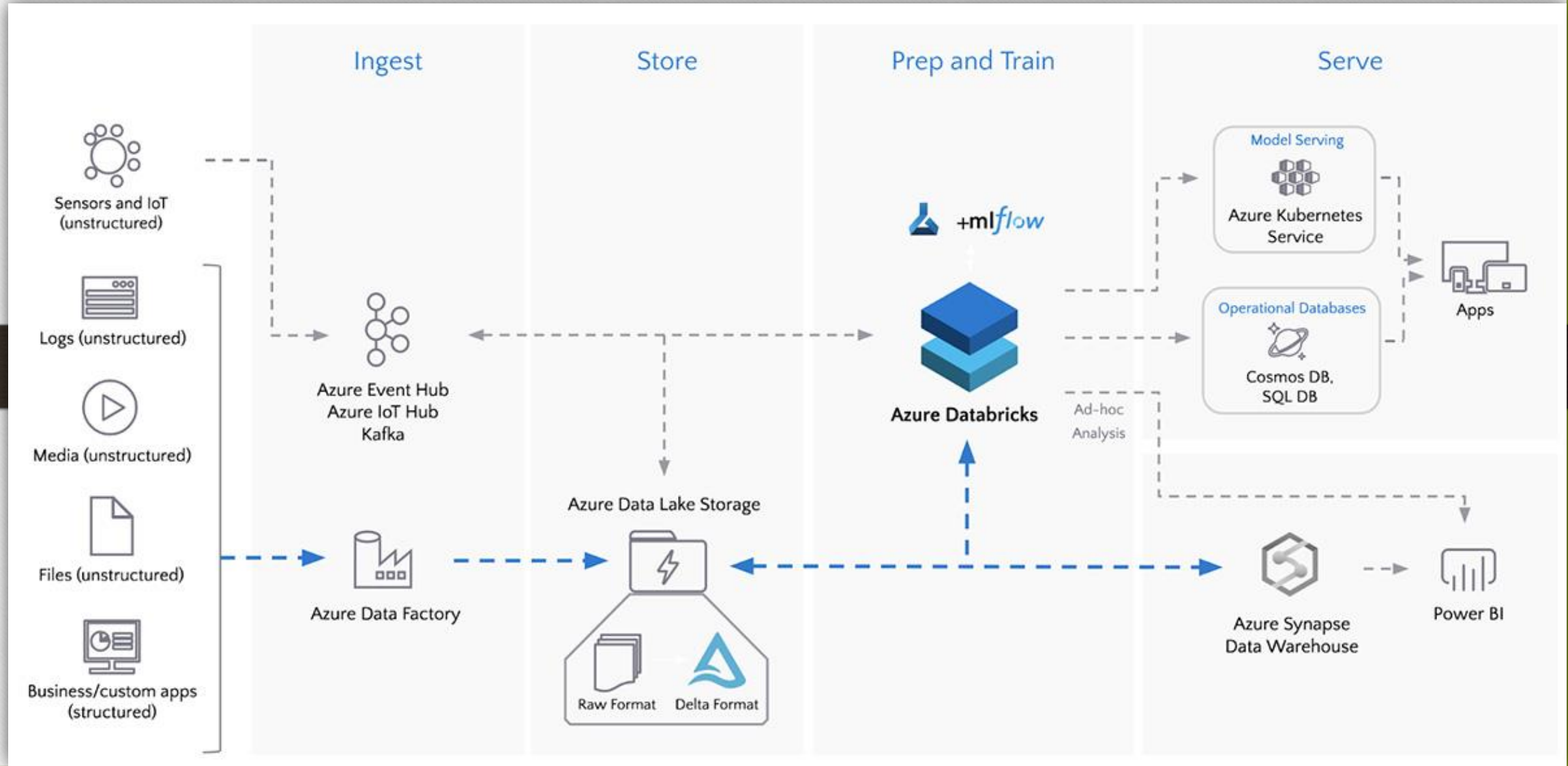


Data transformation in Azure

Transforming data with Azure Data Factory



Azure ADF With BigData Data flow Architecture



ADF V2 Components

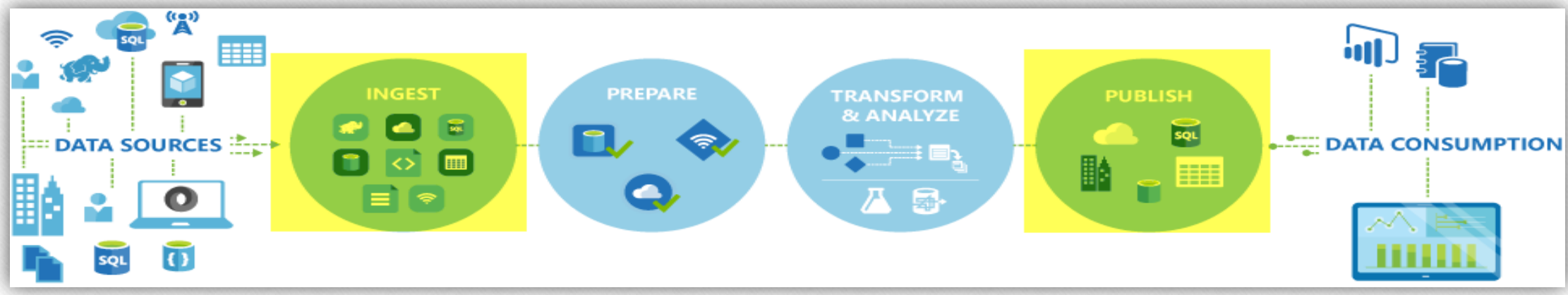
- Azure Data Factory is composed of Five key components. These components work together to provide the platform on which you can compose data-driven workflows with steps to move and transform data.

1. Pipelines
2. Datasets
3. Linked Services (Connections)
4. Integration Runtime
5. Mapping Data Flow

AZURE ADF V2 COMPONENTS FLOW

Big data requires service that can orchestrate and operationalize processes to refine these enormous stores of raw data into actionable business insights. Azure Data Factory is a managed cloud service that's built for these complex hybrid extract-transform-load (ETL), extract-load-transform (ELT), and data integration projects.

The pipelines (data-driven workflows) in Azure Data Factory typically perform the following four steps:

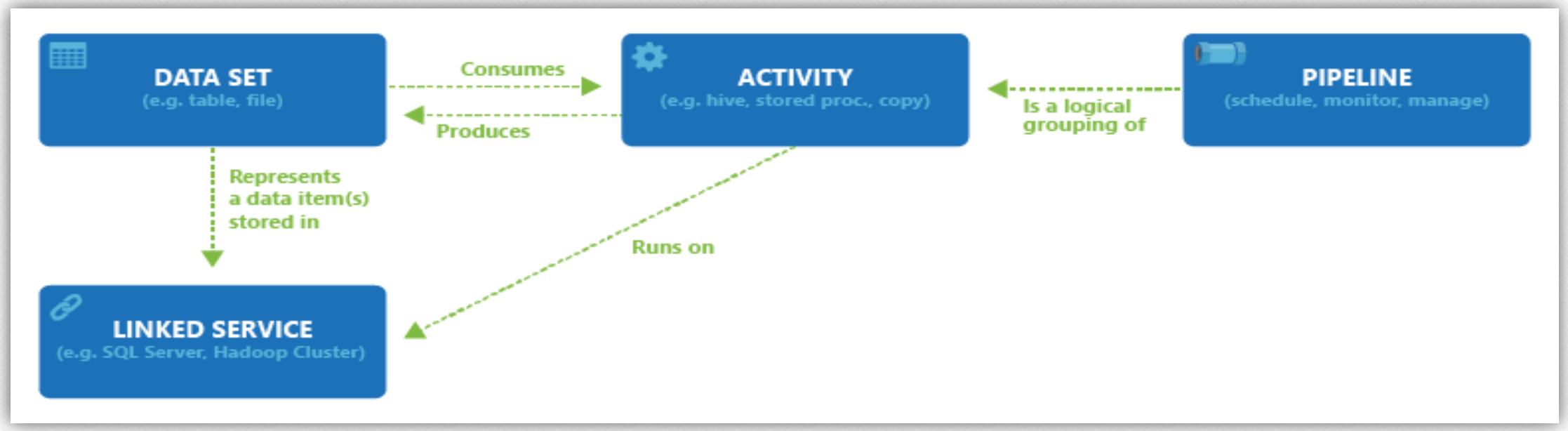


ADF V2 Components Relation

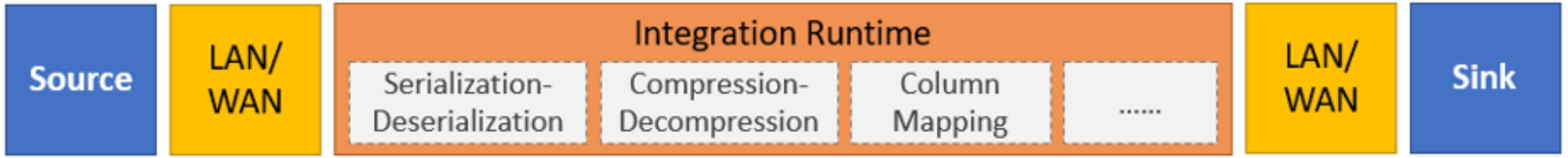
Data Factory supports three types of activities:

- data movement activities
- data transformation activities
- control activities

An activity can take zero or more input datasets and produce one or more output datasets. The following diagram shows the relationship between **pipeline**, **activity**, and **dataset** in Data Factory:



Copy Activity process



- Reads data from a source data store.
- Performs serialization/deserialization, compression/decompression, column mapping, and so on. It performs these operations based on the configuration of the input dataset, output dataset, and Copy activity.
- Writes data to the sink/destination data store

Copy files with the Copy Activity

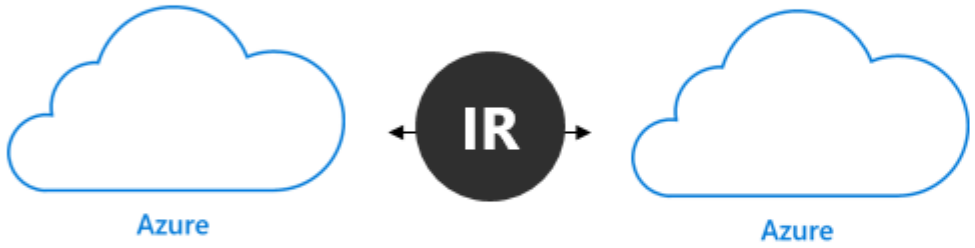
Supported file formats:

- Text
- JSON
- Avro
- ORC
- Parquet

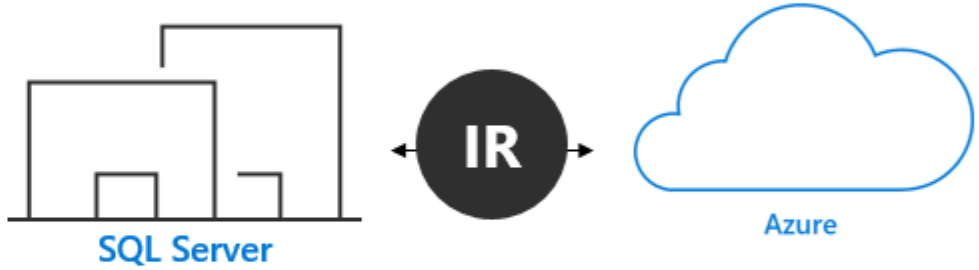
Copy activity can compress and decompress files with The following codecs:

- Gzip
- Deflate
- Bzip2
- ZipDeflate

Integration Runtime



Azure Integration Runtime



Self-hosted Integration Runtime

ADF V2 IR Types

What is Integration Runtime?

The Integration Runtime (IR) is the compute infrastructure used by Azure Data Factory to provide the data integration capabilities across different network environments like

Data Flow, Data Movement, Activity Dispatch and SSIS Package Execution.

Integration runtime types

Data Factory offers three types of Integration Runtime, and you should choose the type that best serve the data integration capabilities and network environment needs you are looking for.

1. Azure
2. Self-hosted
3. Azure-SSIS

IR Use Cases

- Copying between two cloud data source and targets
- Copying between a cloud data source and a data source in private network
- Copying between two data sources in private network

AZURE ADF V2 Control Flow Activities

Append Variable: Append Variable activity could be used to add a value to an existing array variable defined in a Data Factory pipeline.

Set Variable: Set Variable activity can be used to set the value of an existing variable of type String, Bool, or Array defined in a Data Factory pipeline.

Execute Pipeline: The Execute Pipeline activity allows a Data Factory pipeline to invoke another pipeline.

If Condition: If Condition activity allows directing pipeline execution, based on evaluation of certain expressions.

Get Metadata: Get Metadata activity can be used to retrieve metadata of any data in Azure Data Factory.

ForEach: The ForEach activity defines a repeating control flow in your pipeline.

Lookup: Lookup activity can retrieve a dataset from any of the Azure Data Factory supported data sources.

Filter: Filter activity can be used in a pipeline to apply a filter expression to an input array.

Until: Until activity executes a set of activities in a loop until the condition associated with the activity evaluates to true.

Wait: Wait activity allows pausing pipeline execution for specified time period.

Web: Web activity can be used to call a custom REST endpoint from a Data Factory pipeline.

WebHook: Using the webhook activity, you can call an endpoint and pass a callback URL.

Azure Function: The Azure Function activity allows you to run Azure Functions in a Data Factory pipeline.

Home >

Create Data Factory

Changes on this step may reset later selections you have made. Review all options prior to deployment.

Basics | Git configuration | Networking | Advanced | Tags | Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Free Trial

Resource group * : dev_rg

Create new

Instance details

Region * : East US

Name * : pysparkadfv2

Version * : V2

Review + create | < Previous | Next : Git configuration >

Create Data Factory

Basics | **Git configuration** | Networking | Advanced | Tags | Review + create

Azure Data Factory allows you to configure a Git repository with either Azure DevOps or GitHub. Git is a version control system that allows for easier change tracking and collaboration.
Learn more about Git integration in Azure Data Factory

Configure Git later ⓘ ☐

Repository Type * ⓘ ○ Azure DevOps ◉ GitHub

GitHub account * ⓘ raveendratal ✓

Repo name * ⓘ pysparktelugu ✓

Branch name * ⓘ master ✓

Root folder * ⓘ /

Review + create | < Previous | Next : Networking >

Create Data Factory

Validation Passed

Basics | Git configuration | Networking | Advanced | Tags | **Review + create**

TERMS

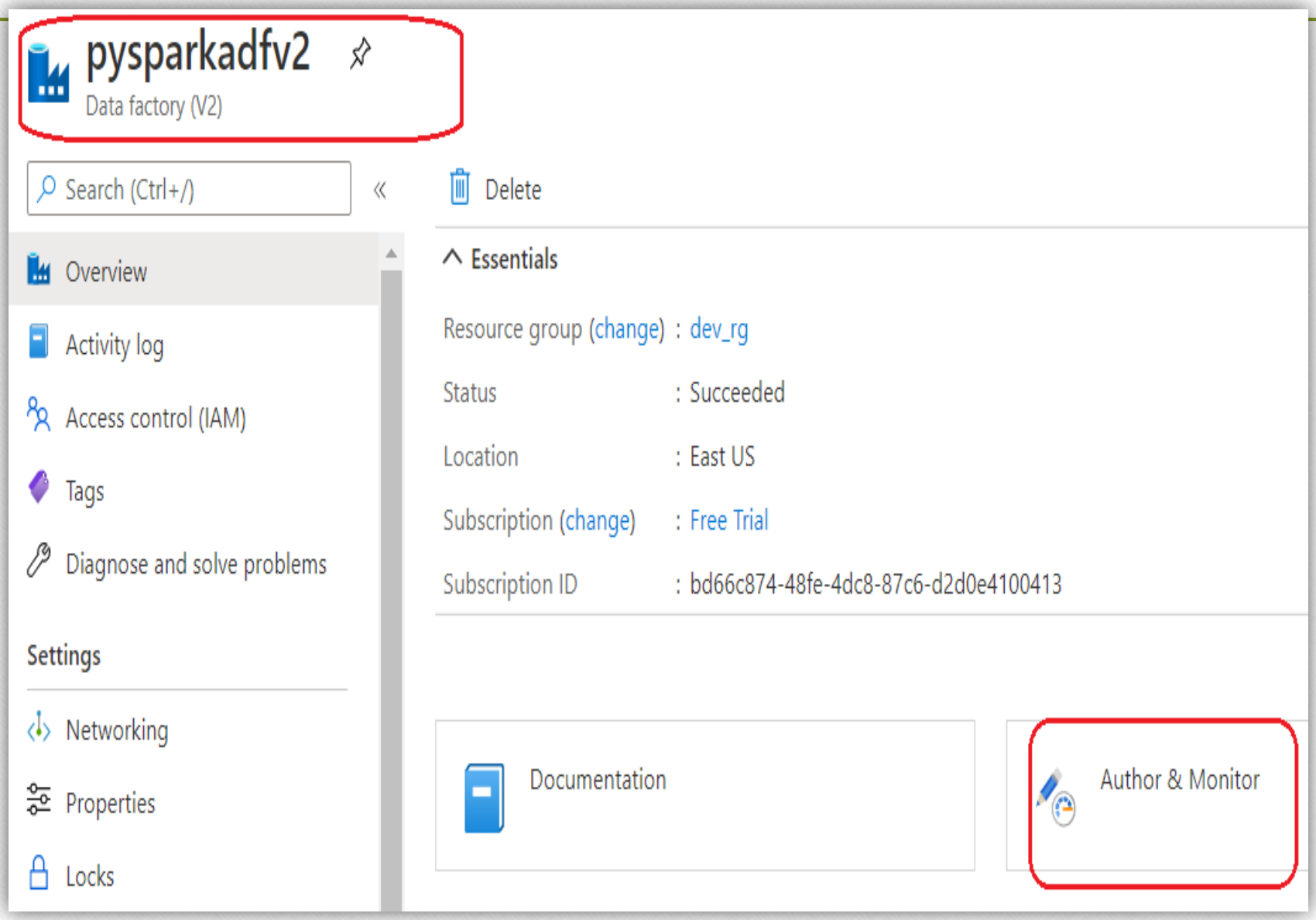
By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the Azure Marketplace Terms for additional details.

Basics

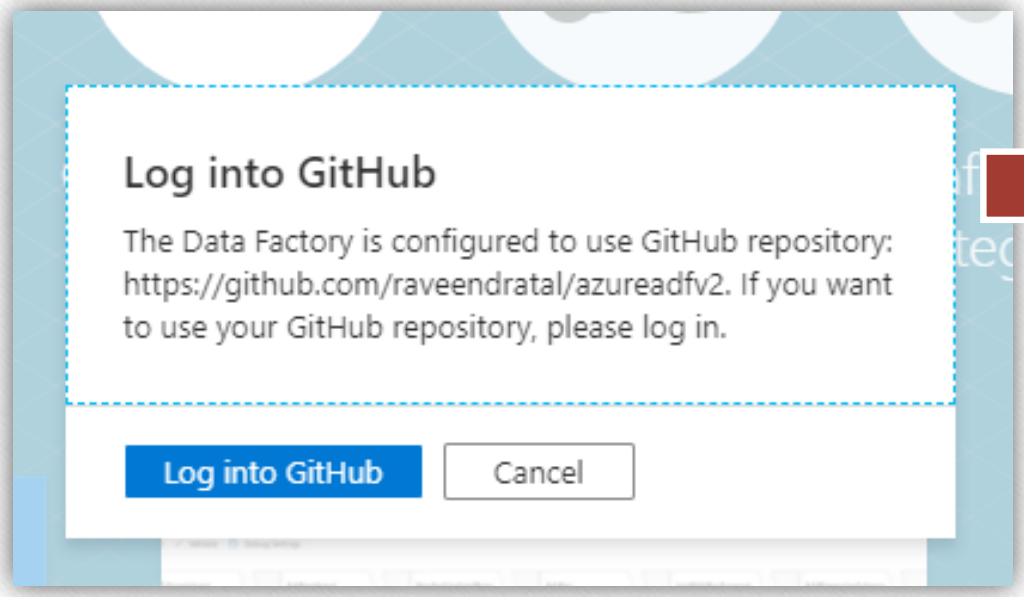
Subscription	Free Trial
Resource group	dev_rg
Region	East US
Name	pysparkadfv2
Version	V2

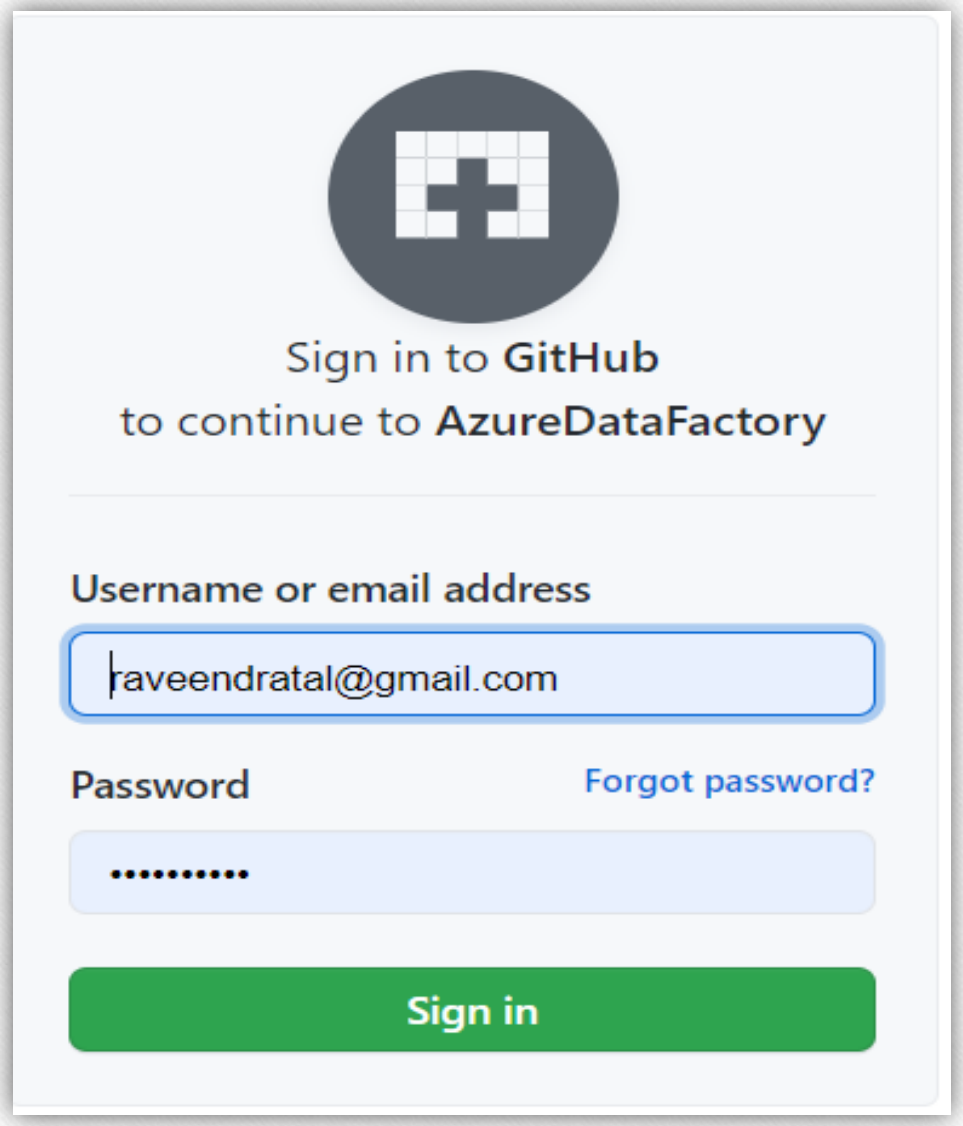
Git configuration

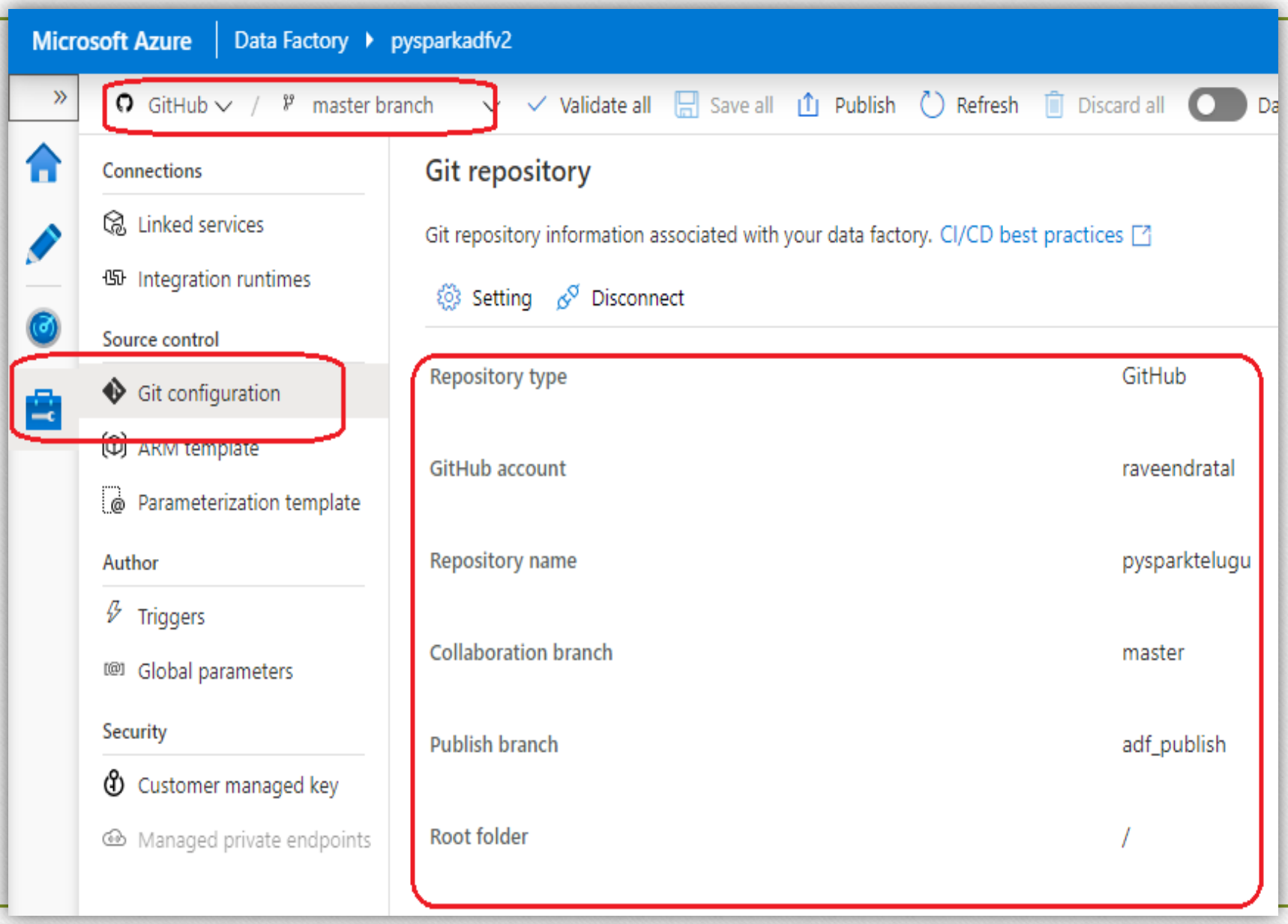
Create | < Previous | Next | Download a template for automation


pysparkadfv2
Data factory (V2)
Search (Ctrl+/)
Delete
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Networking
Properties
Locks
Essentials
Resource group (change) : dev_rg
Status : Succeeded
Location : East US
Subscription (change) : Free Trial
Subscription ID : bd66c874-48fe-4dc8-87c6-d2d0e4100413
Documentation
Author & Monitor

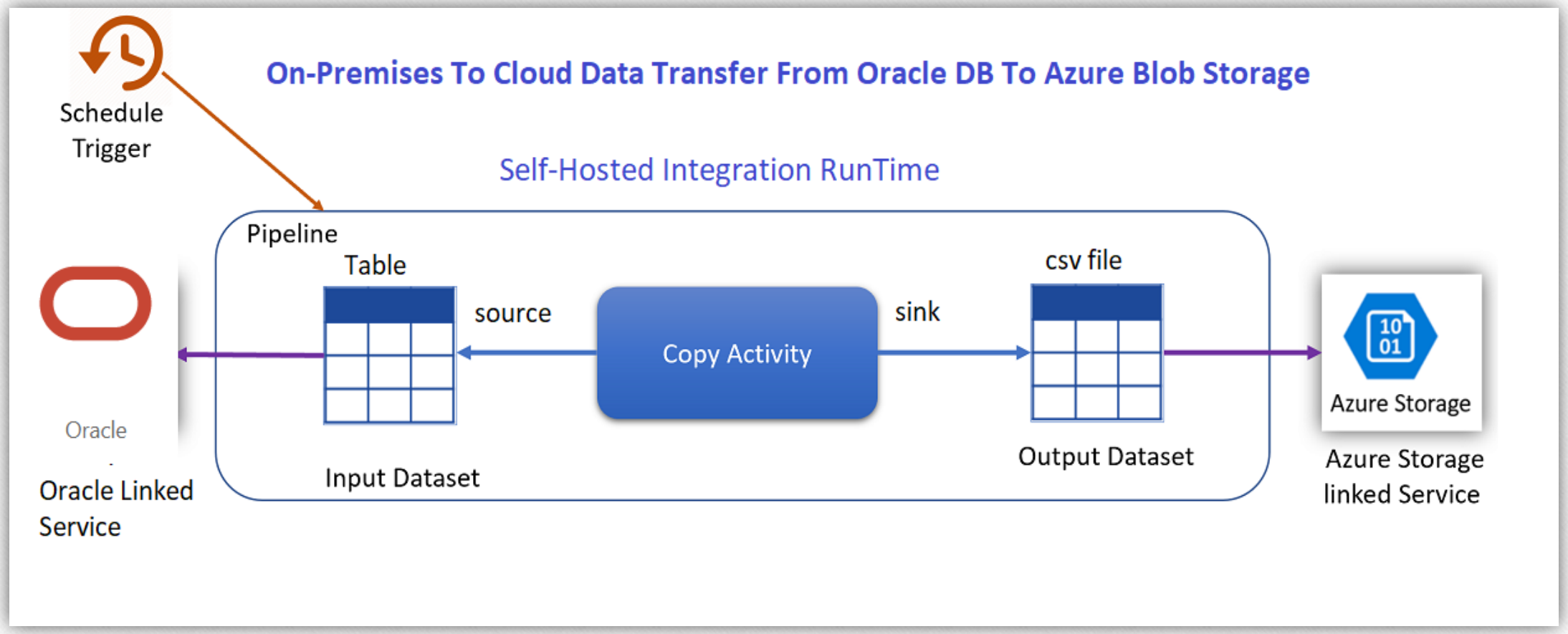
Github Integration in Azure ADF






Microsoft Azure
Data Factory
pysparkadfv2
GitHub
master branch
Validate all
Save all
Publish
Refresh
Discard all
Connections
Linked services
Integration runtimes
Source control
Git configuration
ARM template
Parameterization template
Author
Triggers
Global parameters
Security
Customer managed key
Managed private endpoints
Git repository
Git repository information associated with your data factory. CI/CD best practices
Setting
Disconnect
Repository type
GitHub
GitHub account
raveendratal
Repository name
pysparktelugu
Collaboration branch
master
Publish branch
adf_publish
Root folder
/

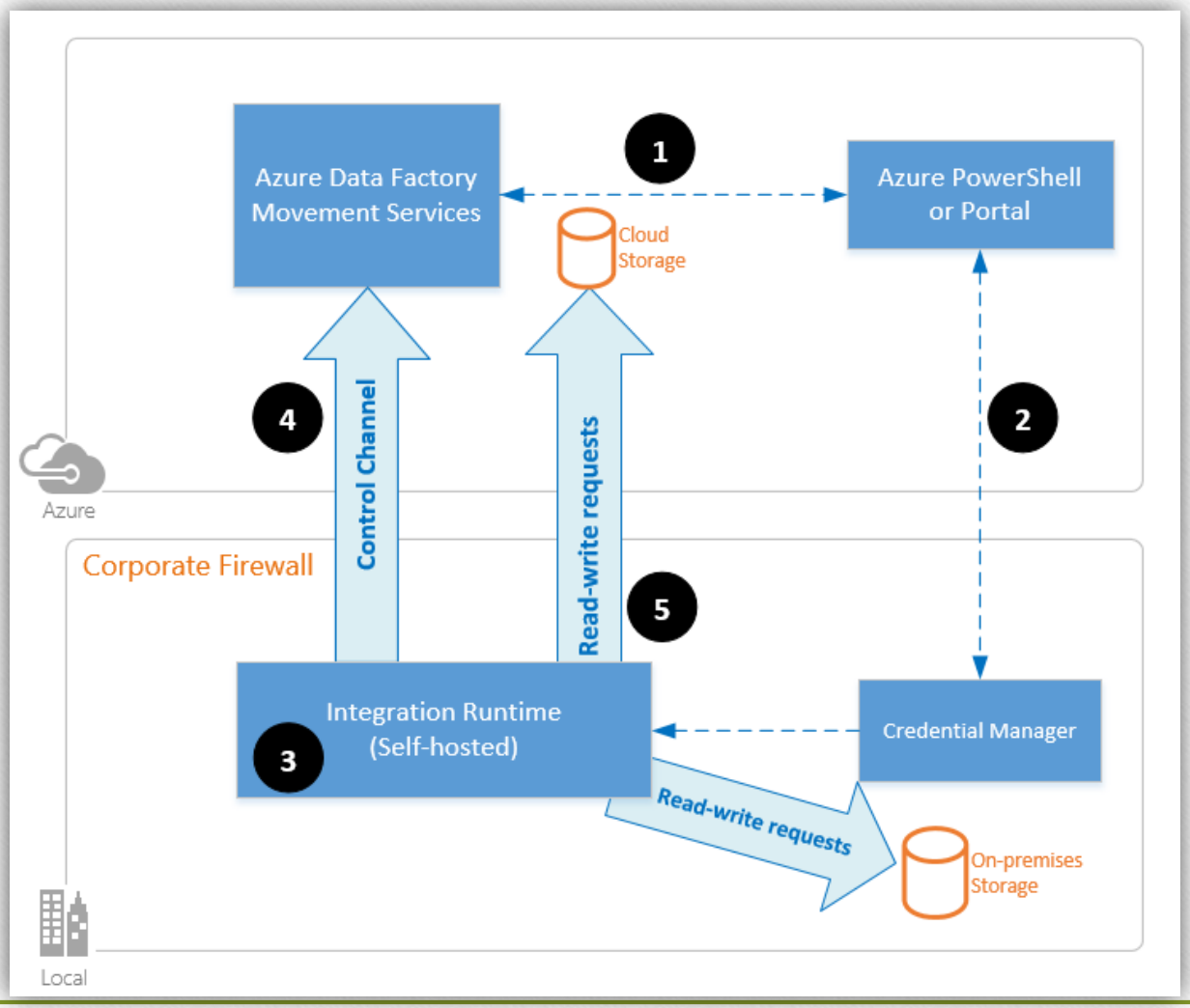
Loading Data from On-Premises Oracle To Azure Blob Storage



1) Creating Self-Hosted Integration Runtime
2) Creating Linked Service for Source Oracle And Target Blob Storage.
3) Creating Datasets for Source Oracle Table And Target Blog File store.

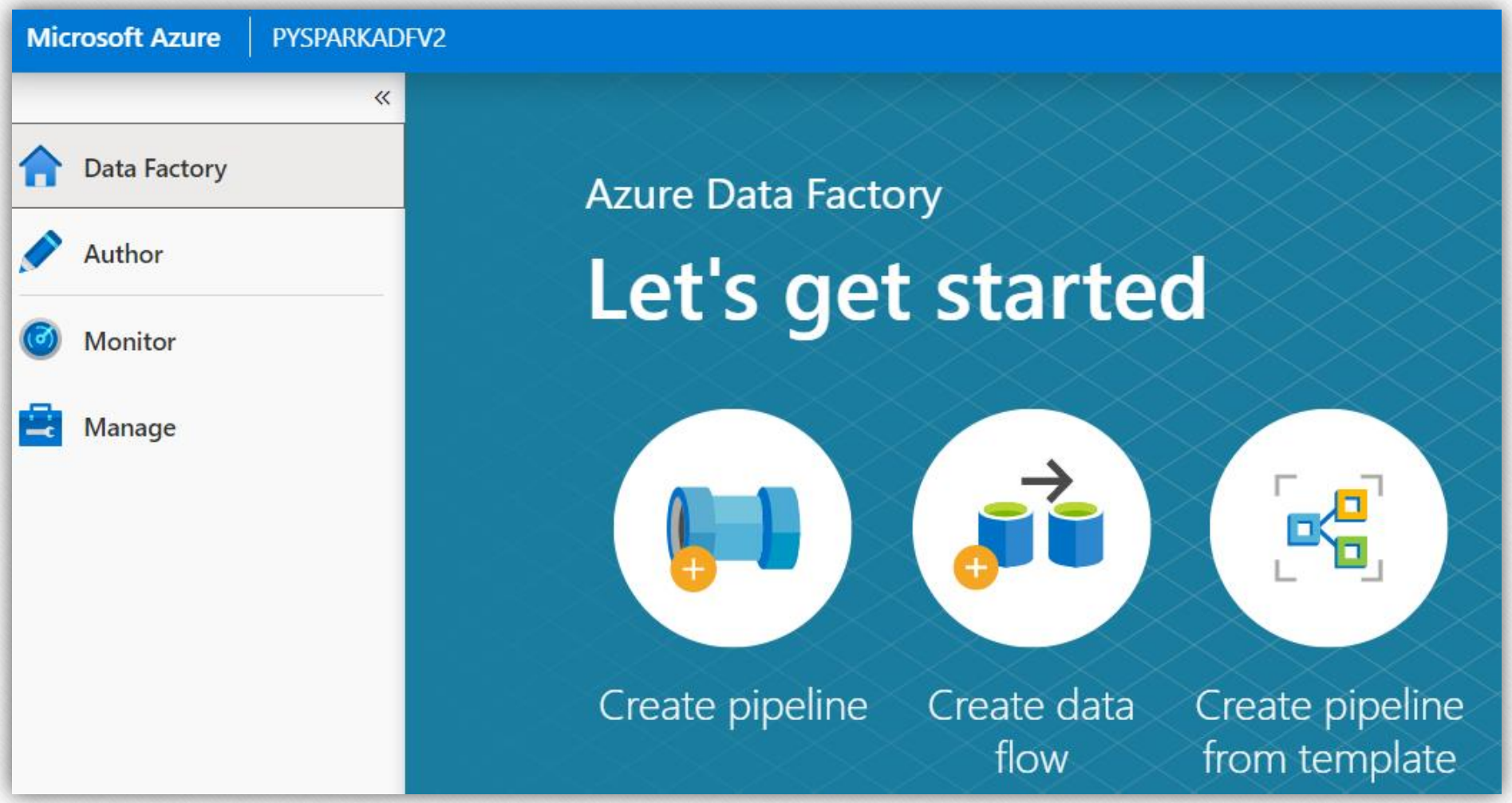
Create and configure a self-hosted integration runtime

A self-hosted integration runtime can run copy activities between a cloud data store and a data store in a private network. It also can dispatch transform activities against compute resources in an on-premises network or an Azure virtual network. The installation of a self-hosted integration runtime needs an on-premises machine or a virtual machine inside a private network.

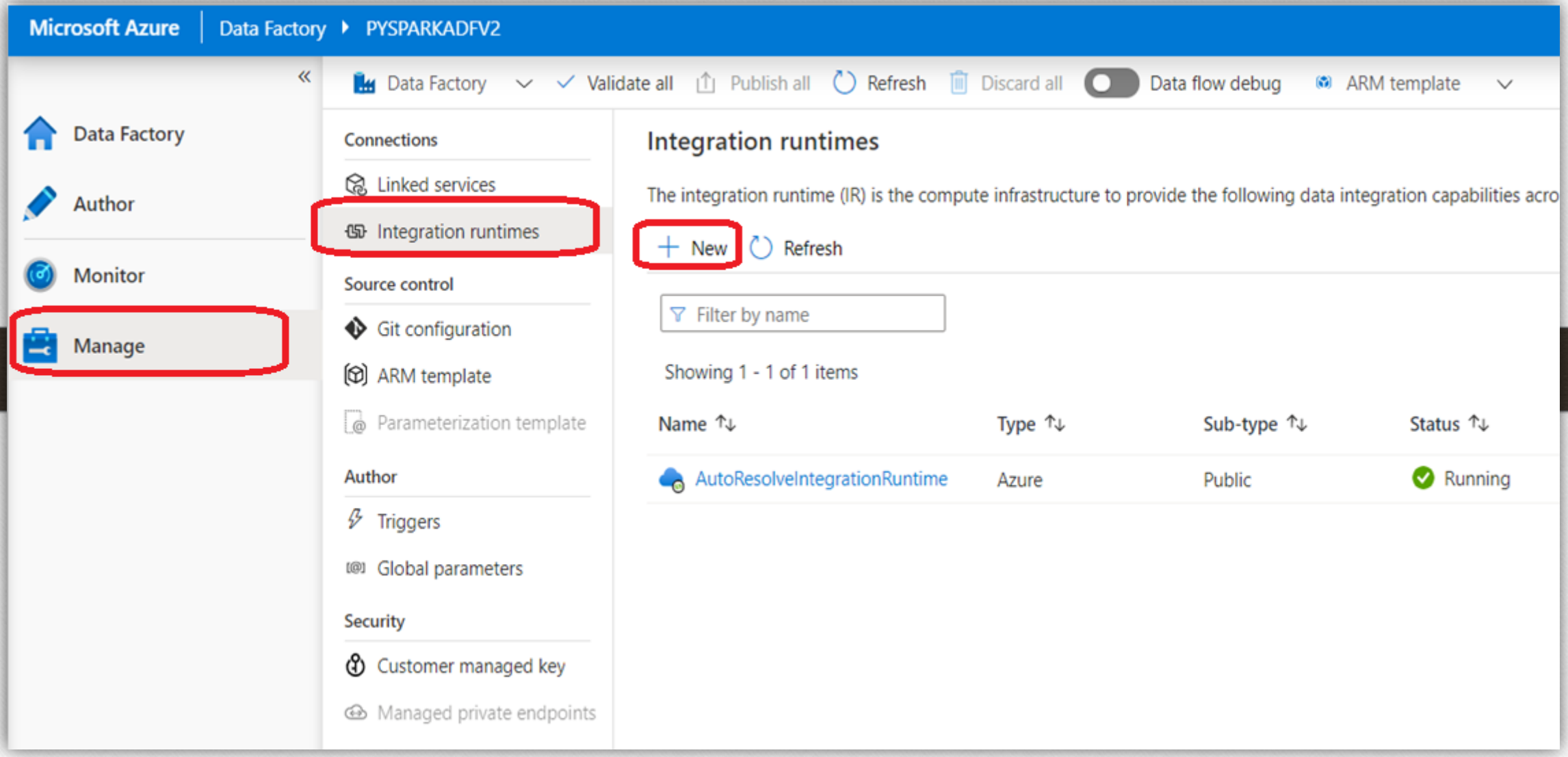


Creating Self-Hosted Integration Runtime

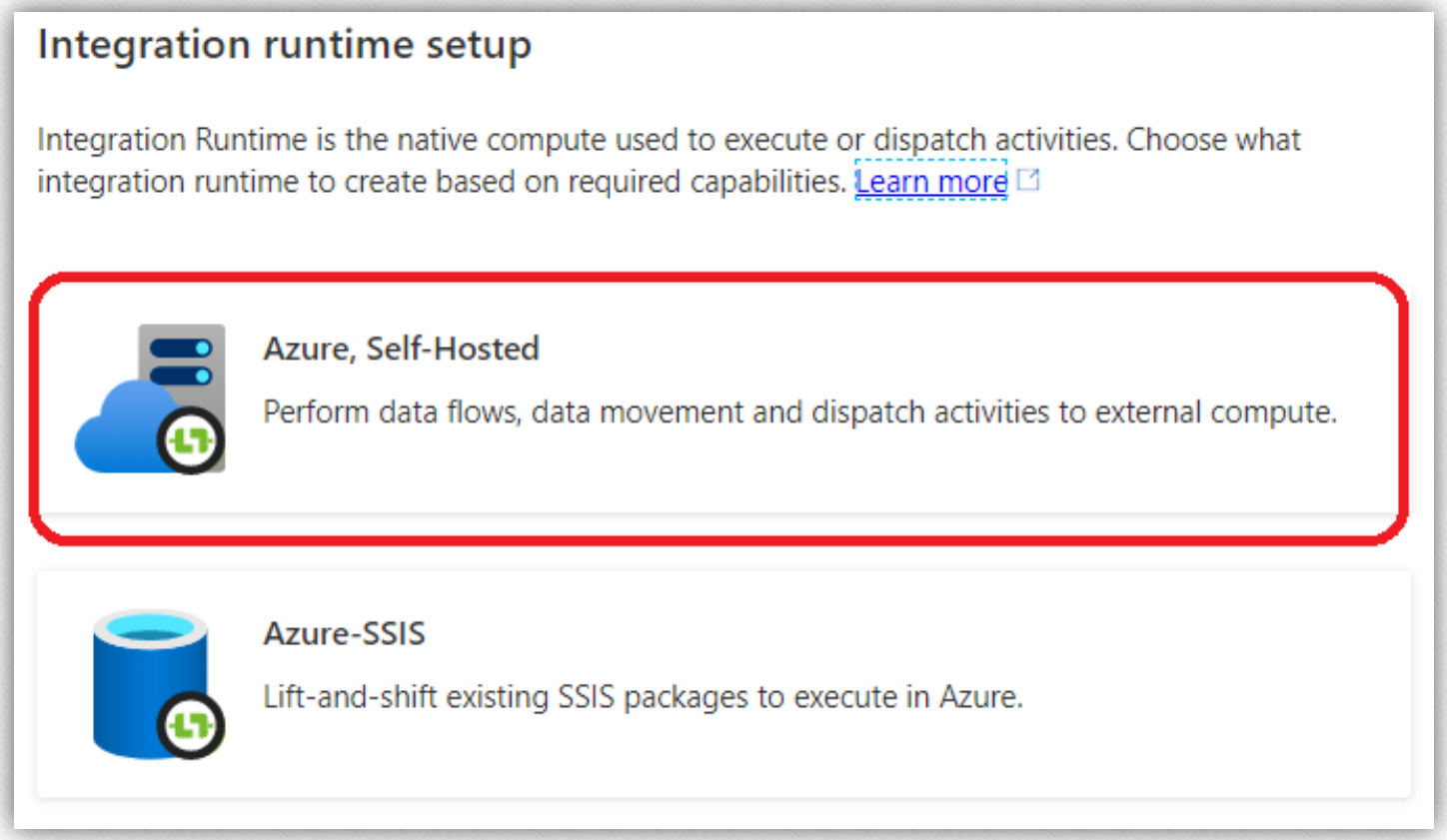
On the **Let's get started** page of Azure Data Factory UI, select the **Manage** Tab from the leftmost pane.



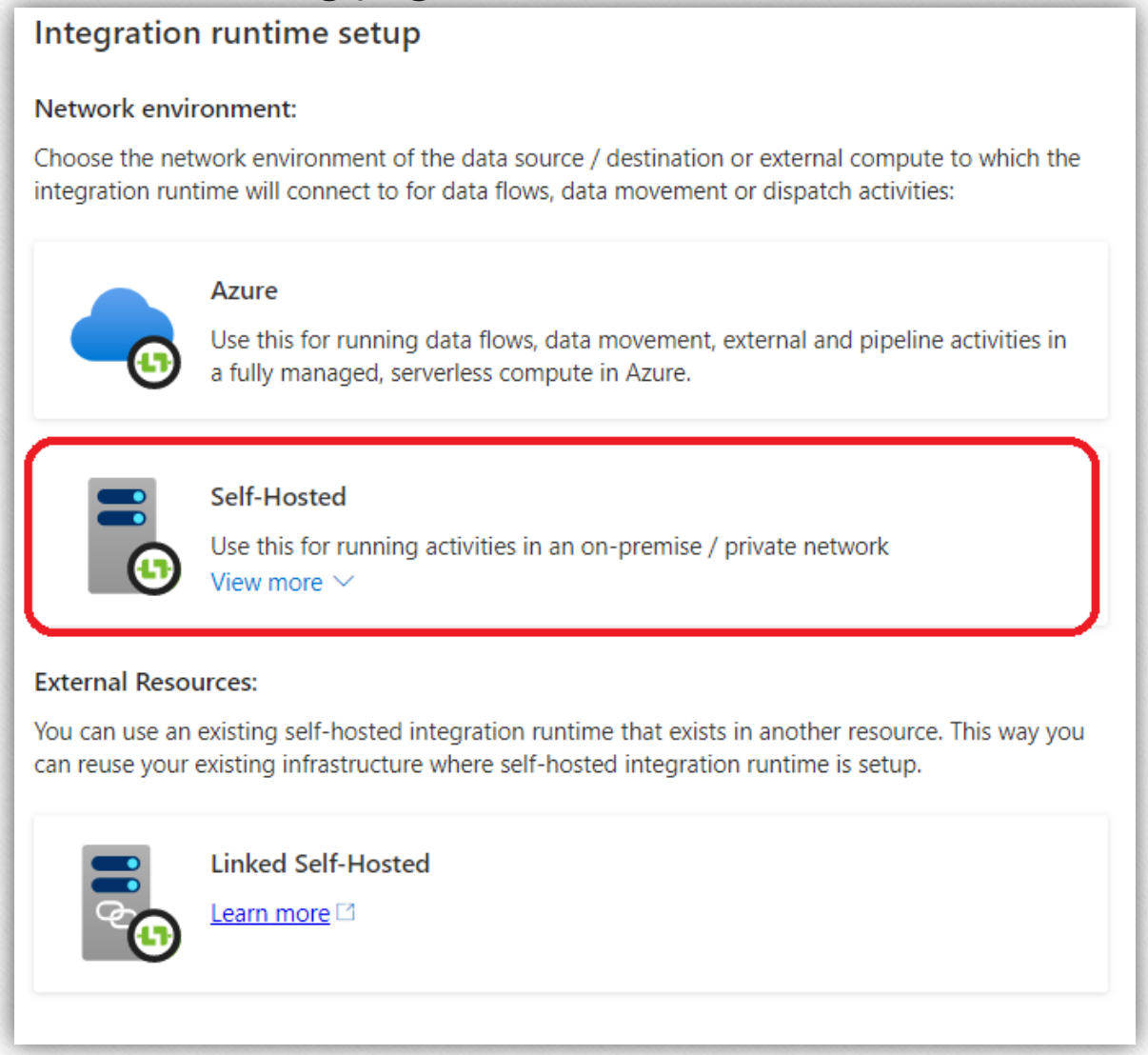
Select **Integration runtimes** on the left pane, and then select **+New**.



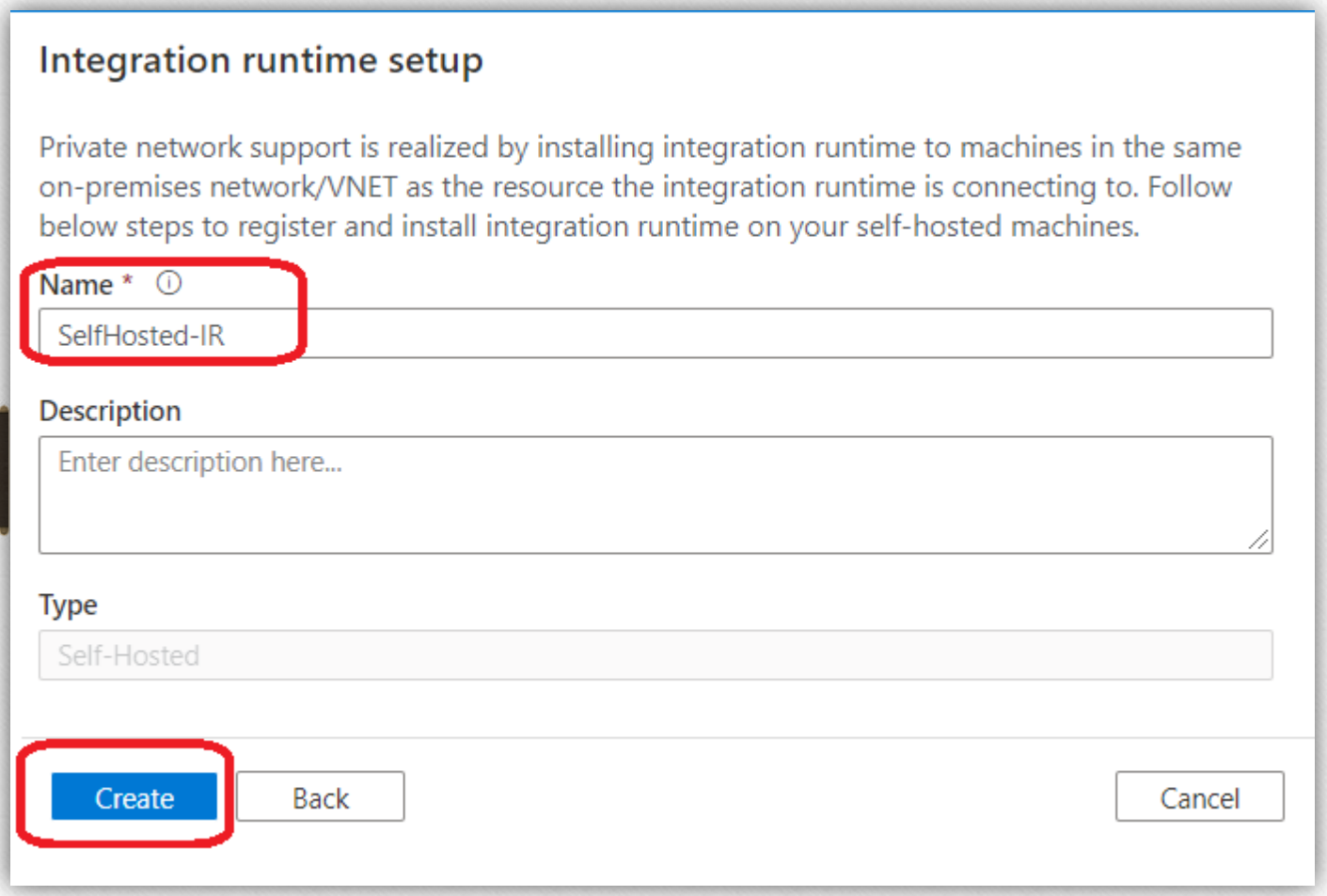
On the **Integration runtime setup** page, select **Azure, Self-Hosted**, and then select **Continue**.



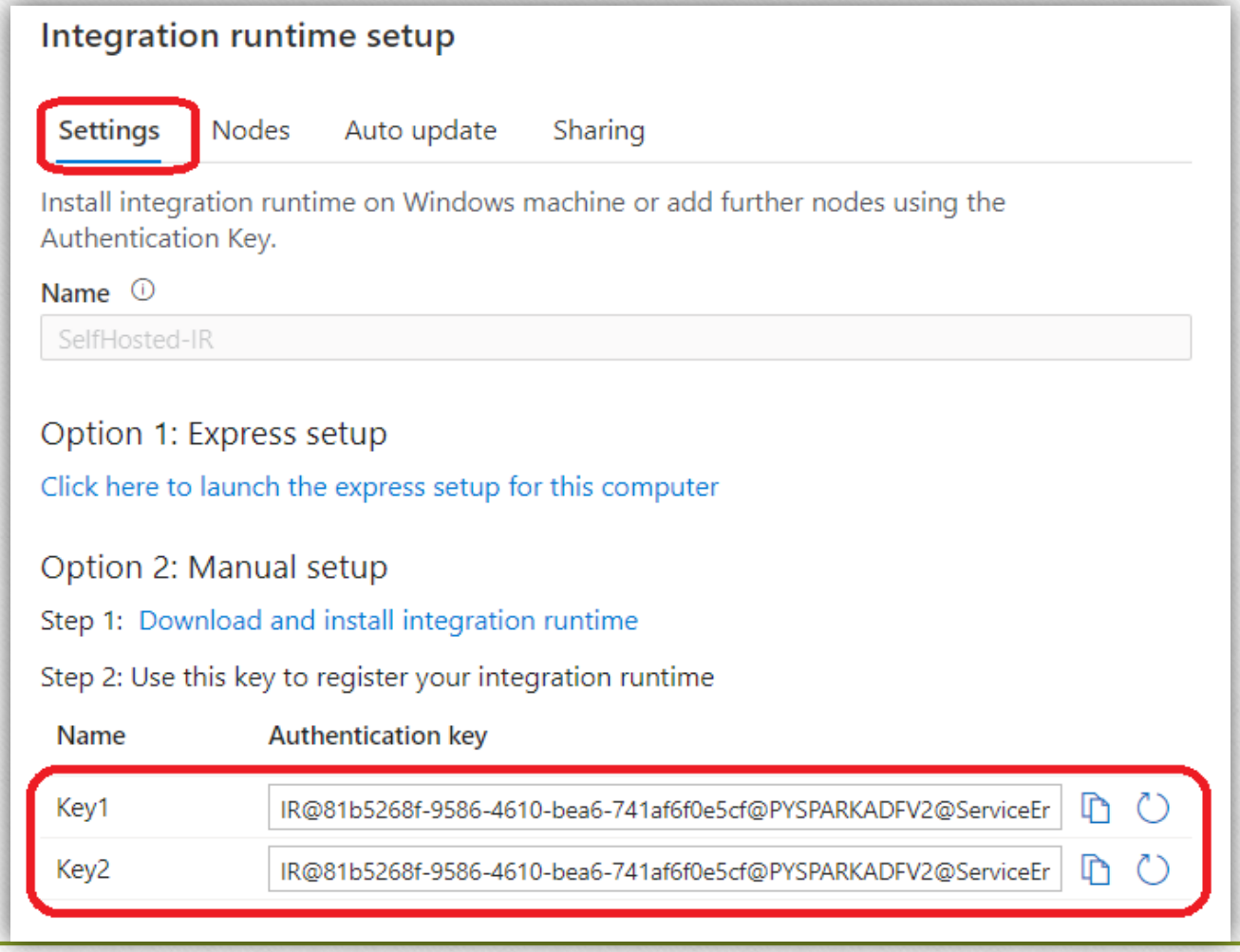
On the following page, select **Self-Hosted** to create a Self-Hosted IR, and then select **Continue**.



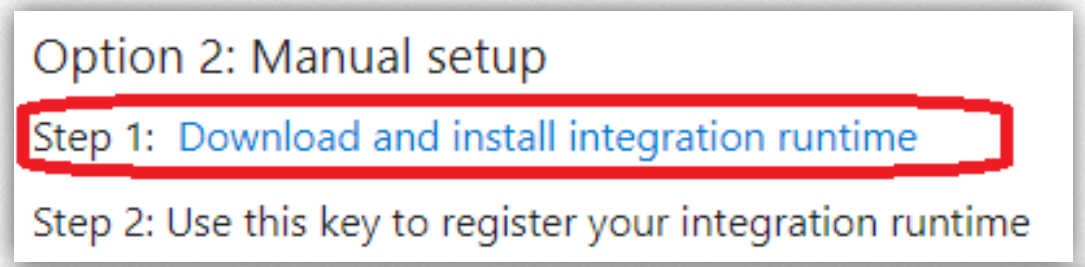
Enter a name for your IR, and select **Create**.

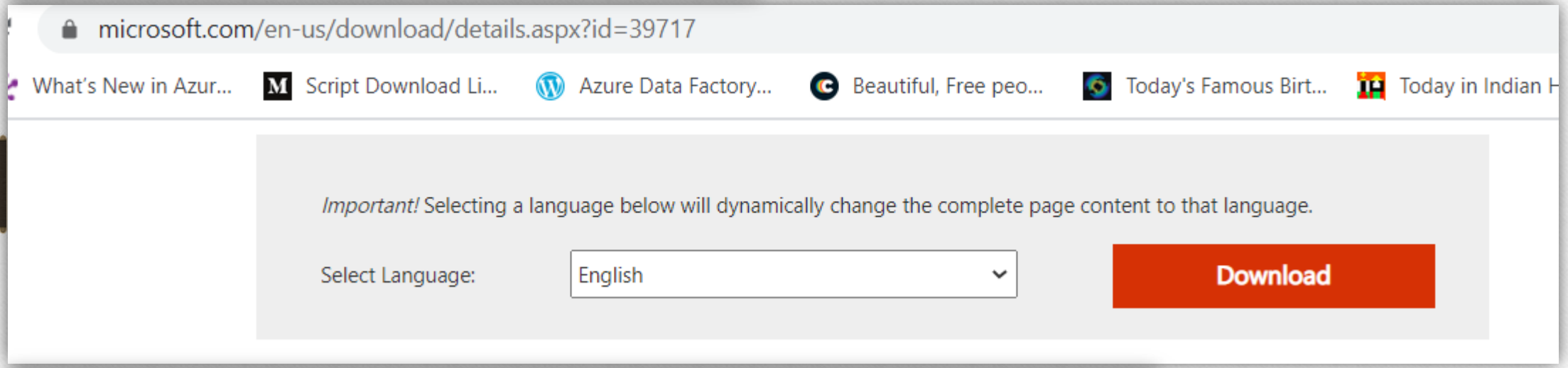


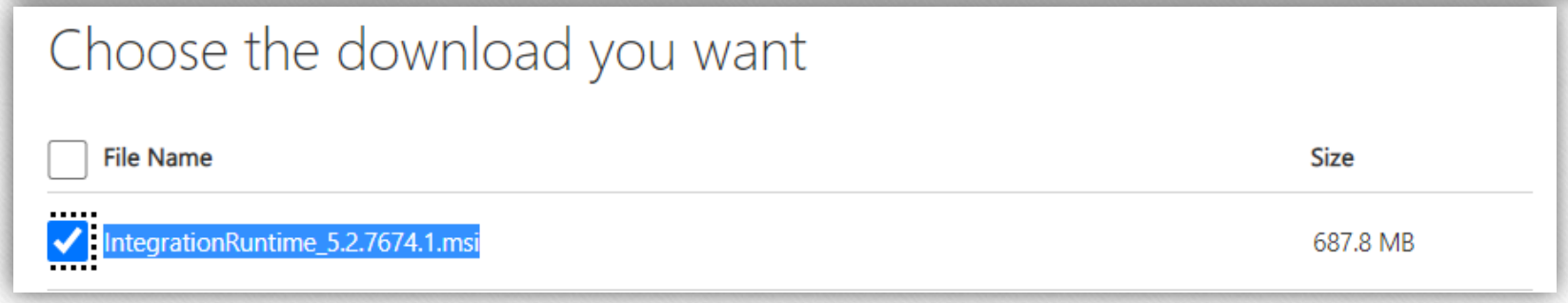
On the **Integration runtime setup** page, select the link under **Option 1** to open the express setup on your computer. Or follow the steps under **Option 2** to set up manually. The following instructions are based on manual setup:



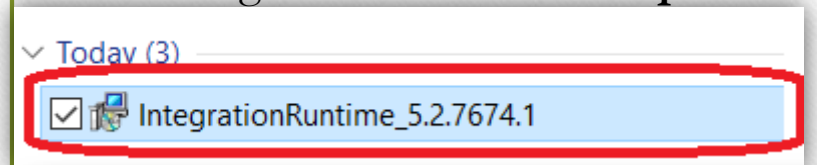
Option 2 to set up manually. Click on download from below link and install

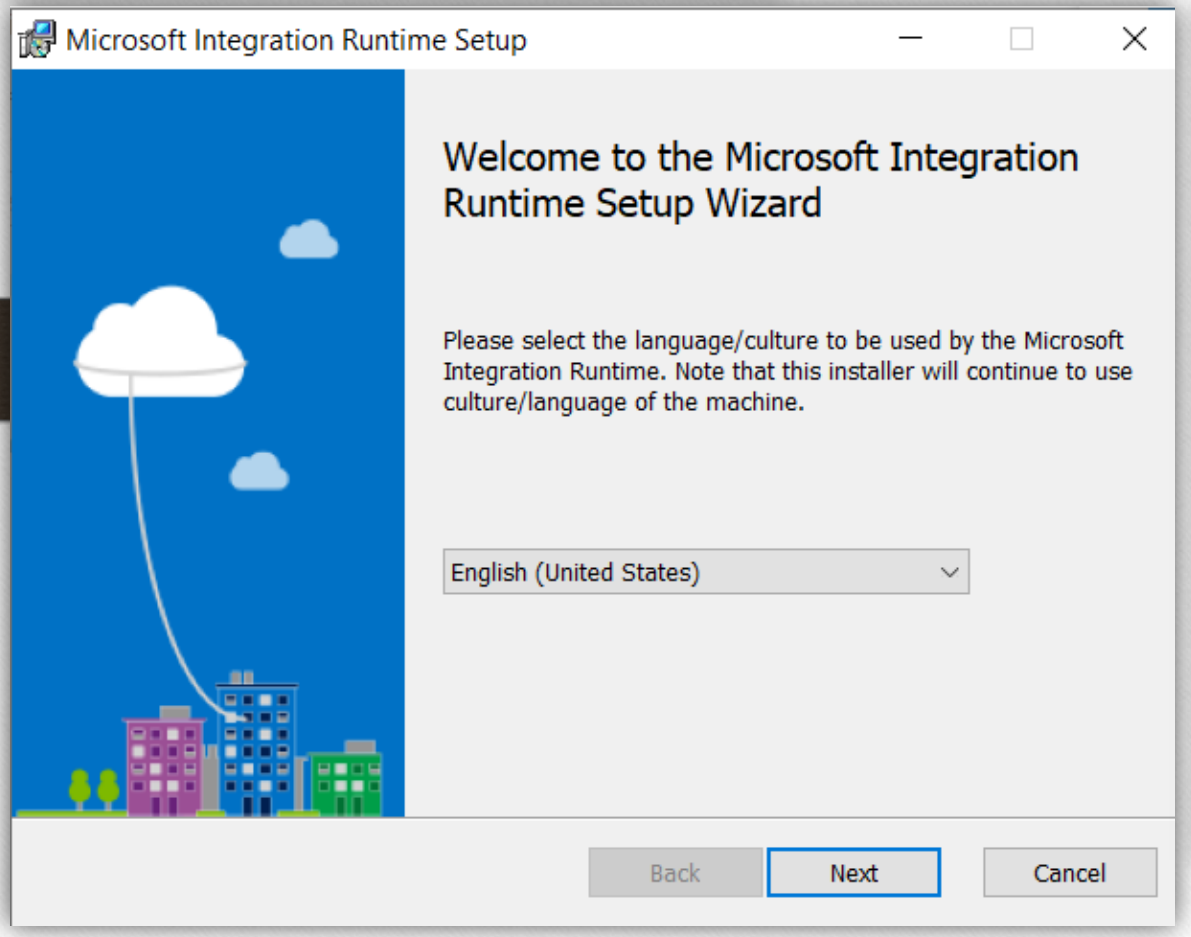


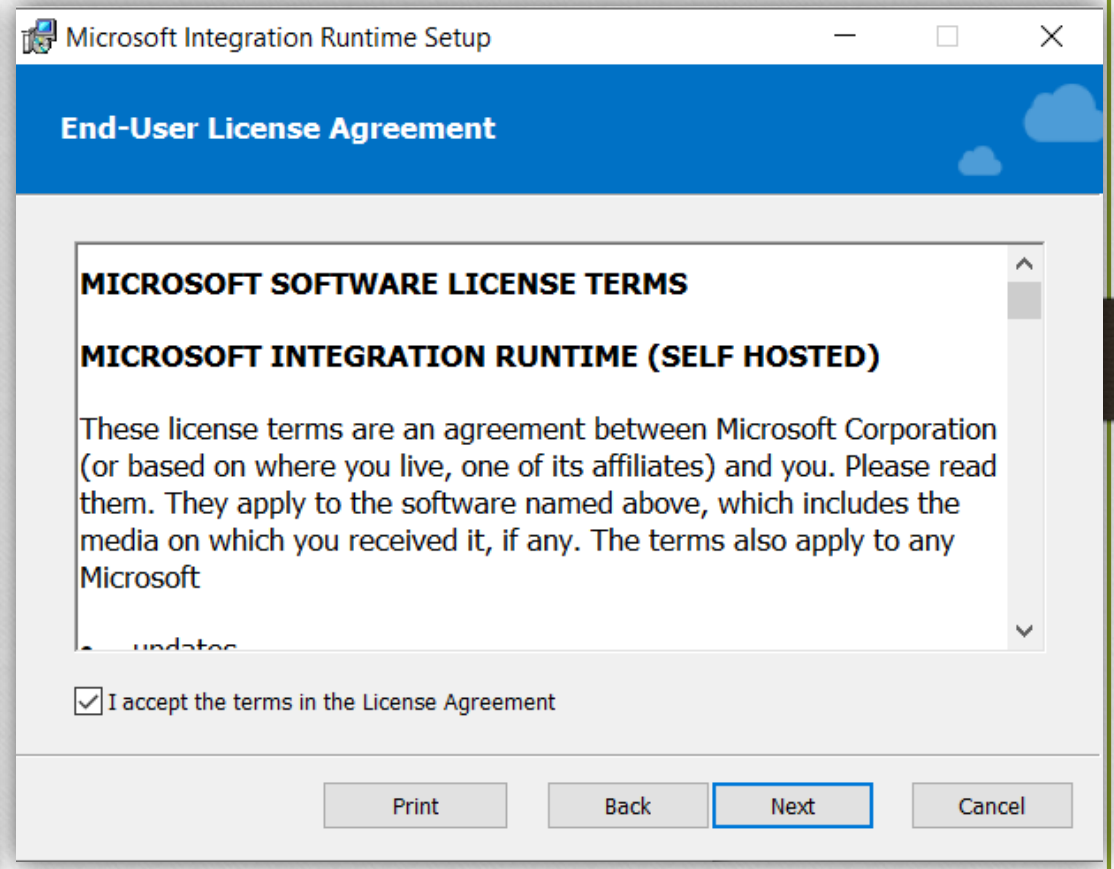




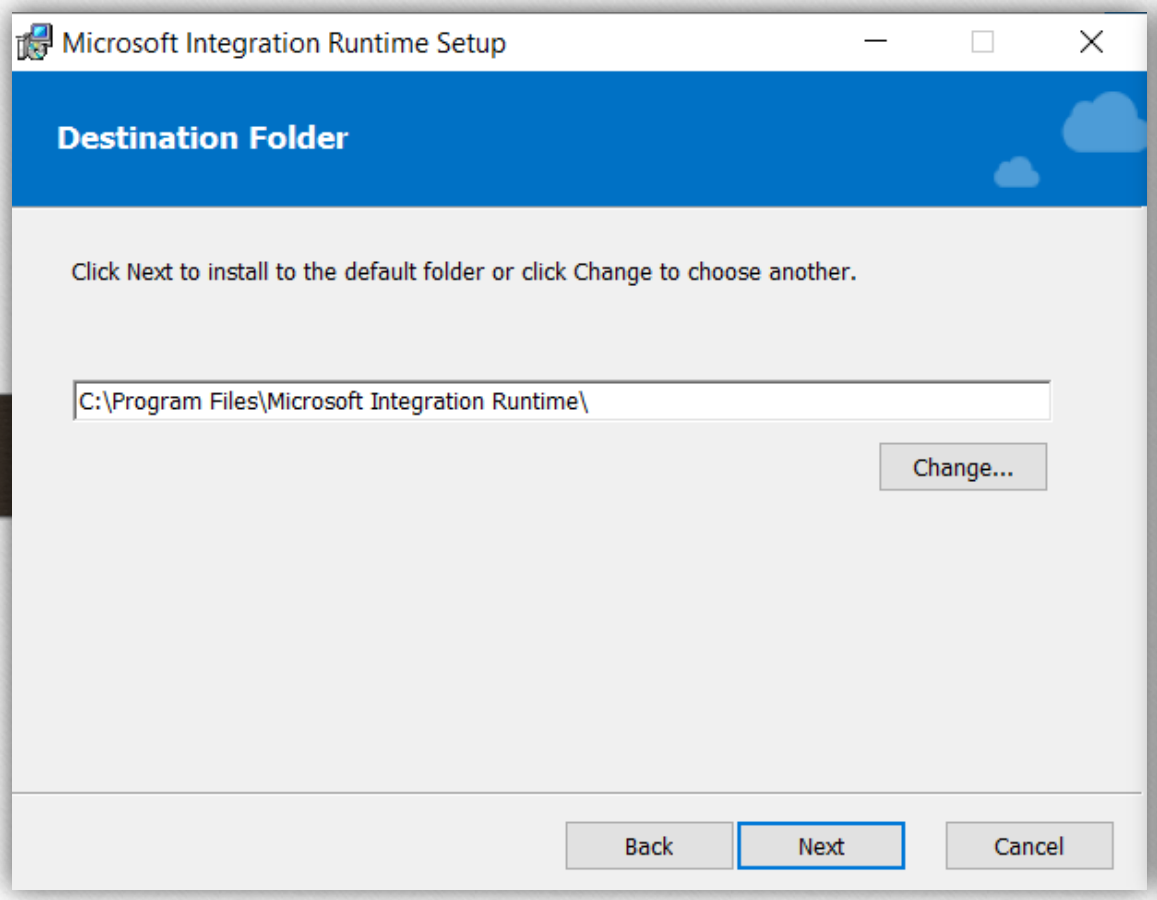
Install IntegrationRuntime in **on-premises**.

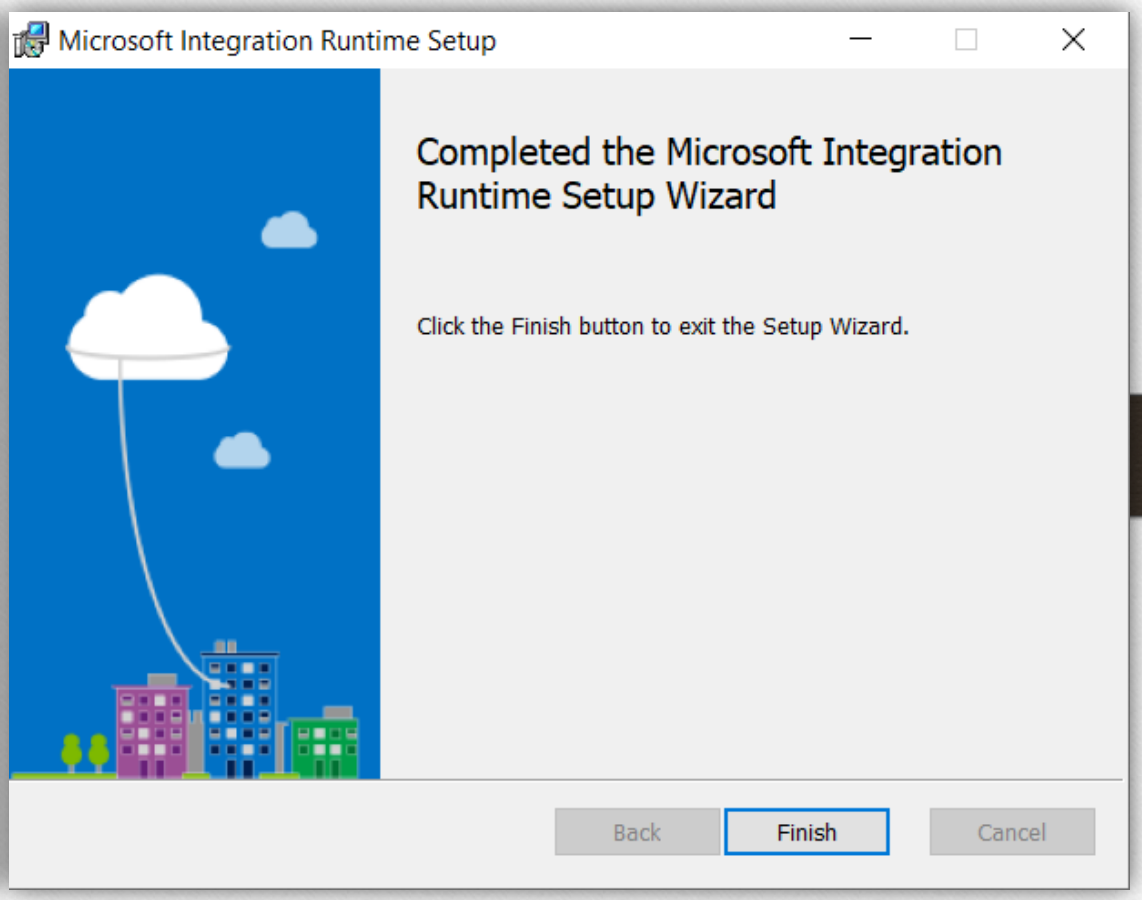




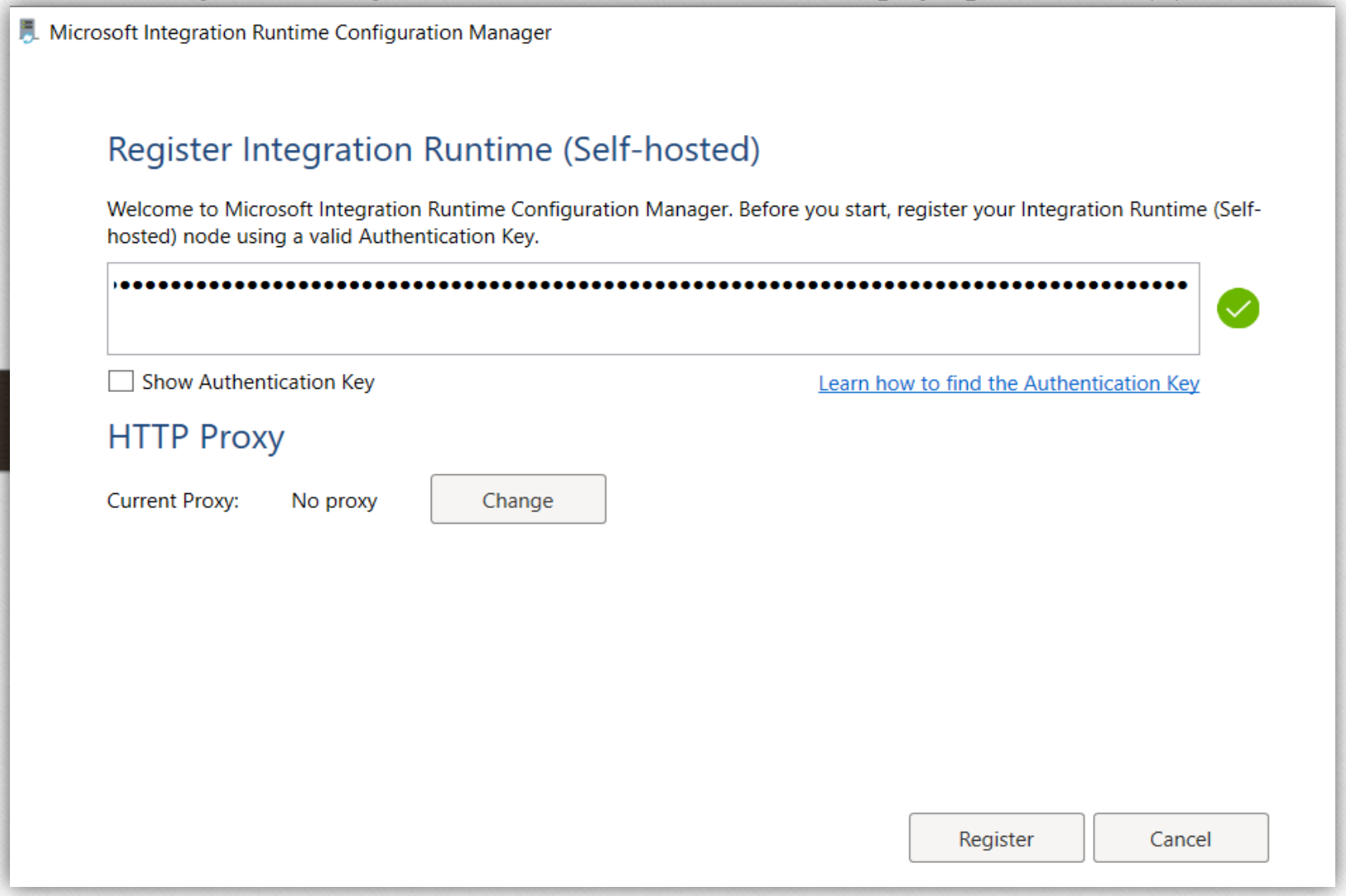


Select install location and install Integration Runtime

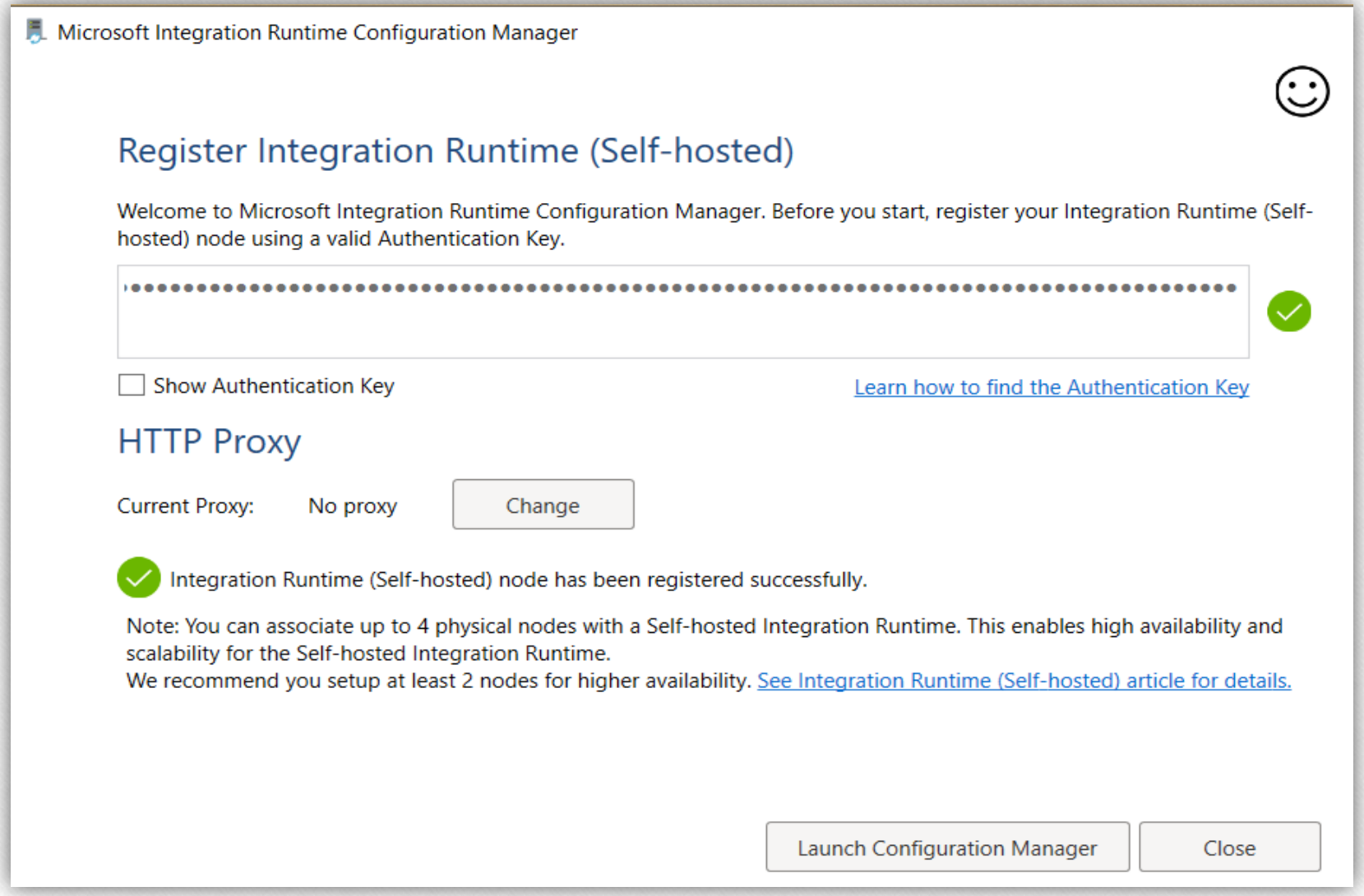




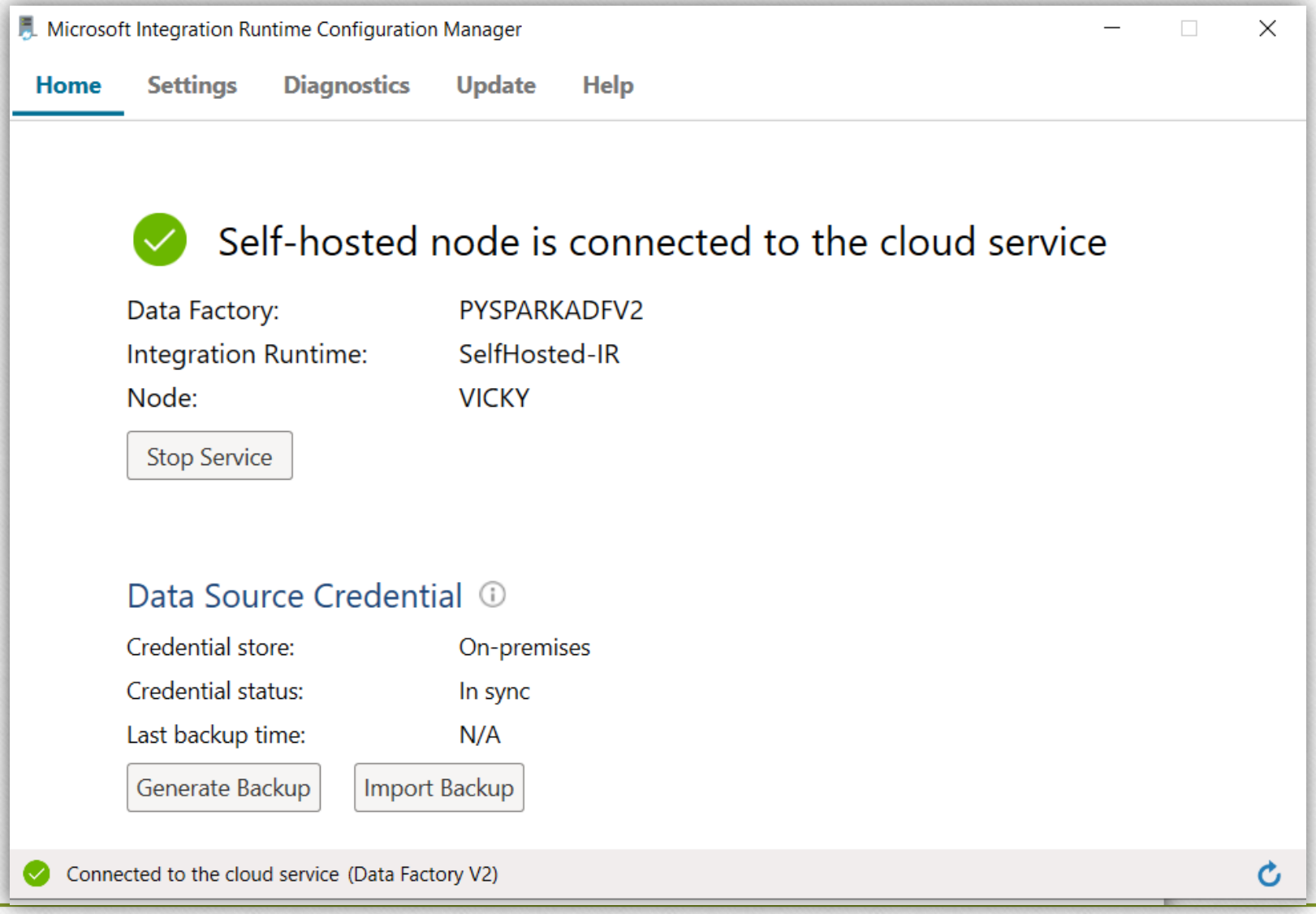
On the **Register Integration Runtime (Self-hosted)** page, paste the key you saved earlier, and select **Register**.



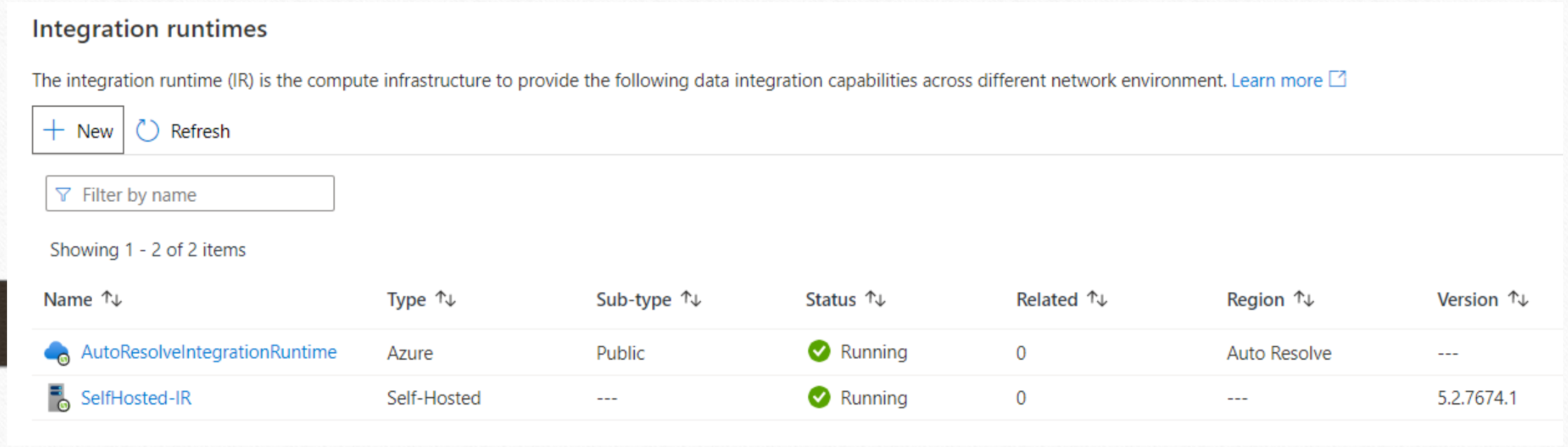
After the self-hosted integration runtime is registered successfully, you see the following window



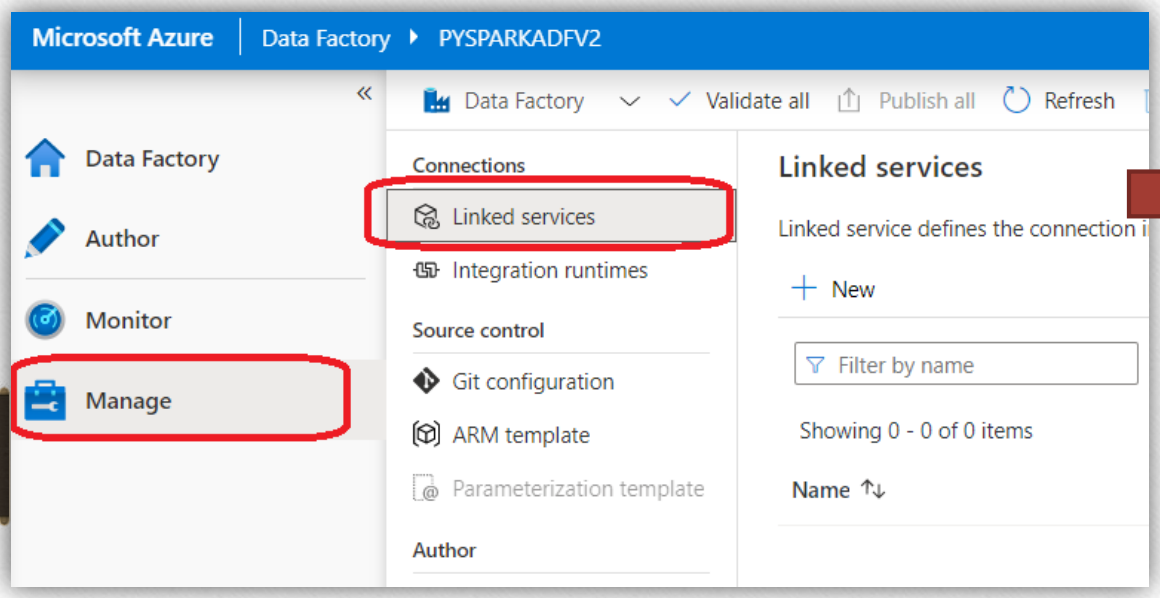
After click on Launch Configuration Manager.

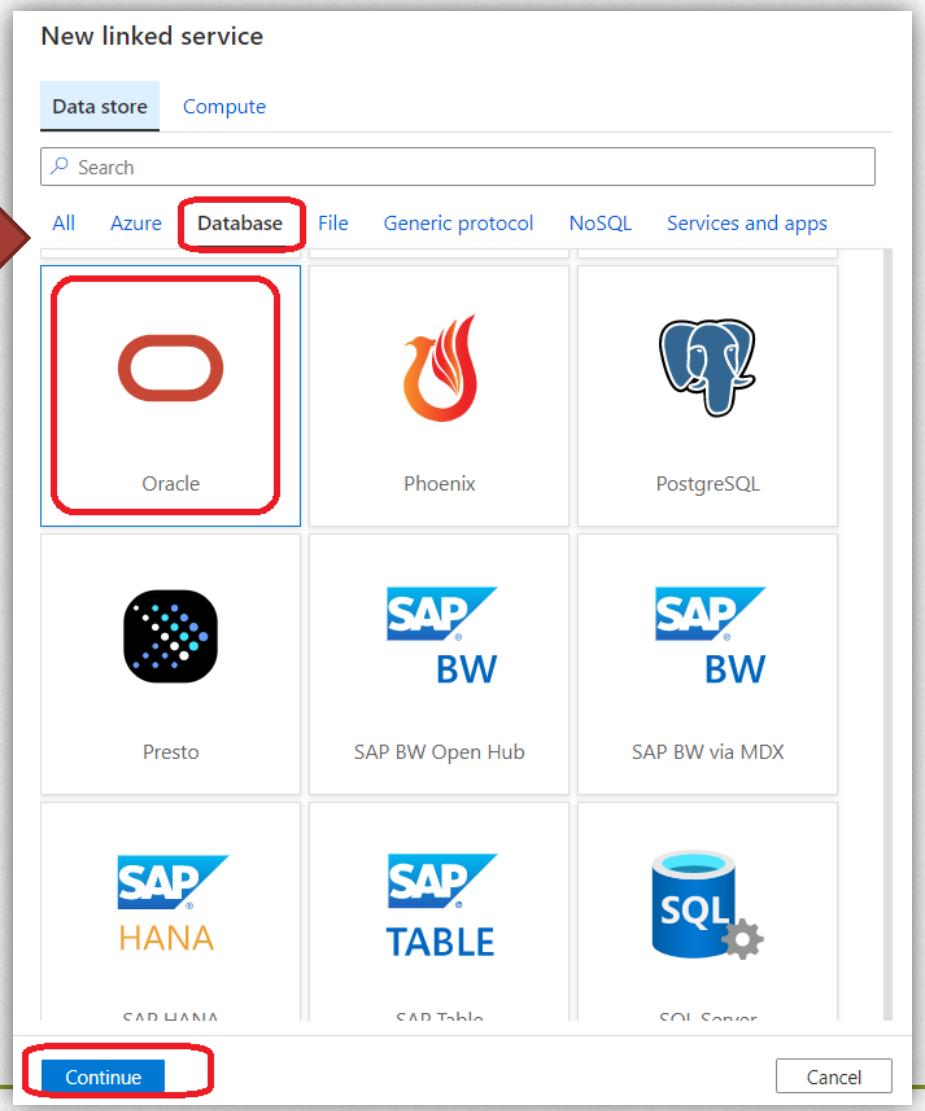


Self Hosted Integration Runtime configured and its connected On-Premises And status showing as Running.
Now we can transfer Data From On-Premises Oracle Database To Azure Blob Store.

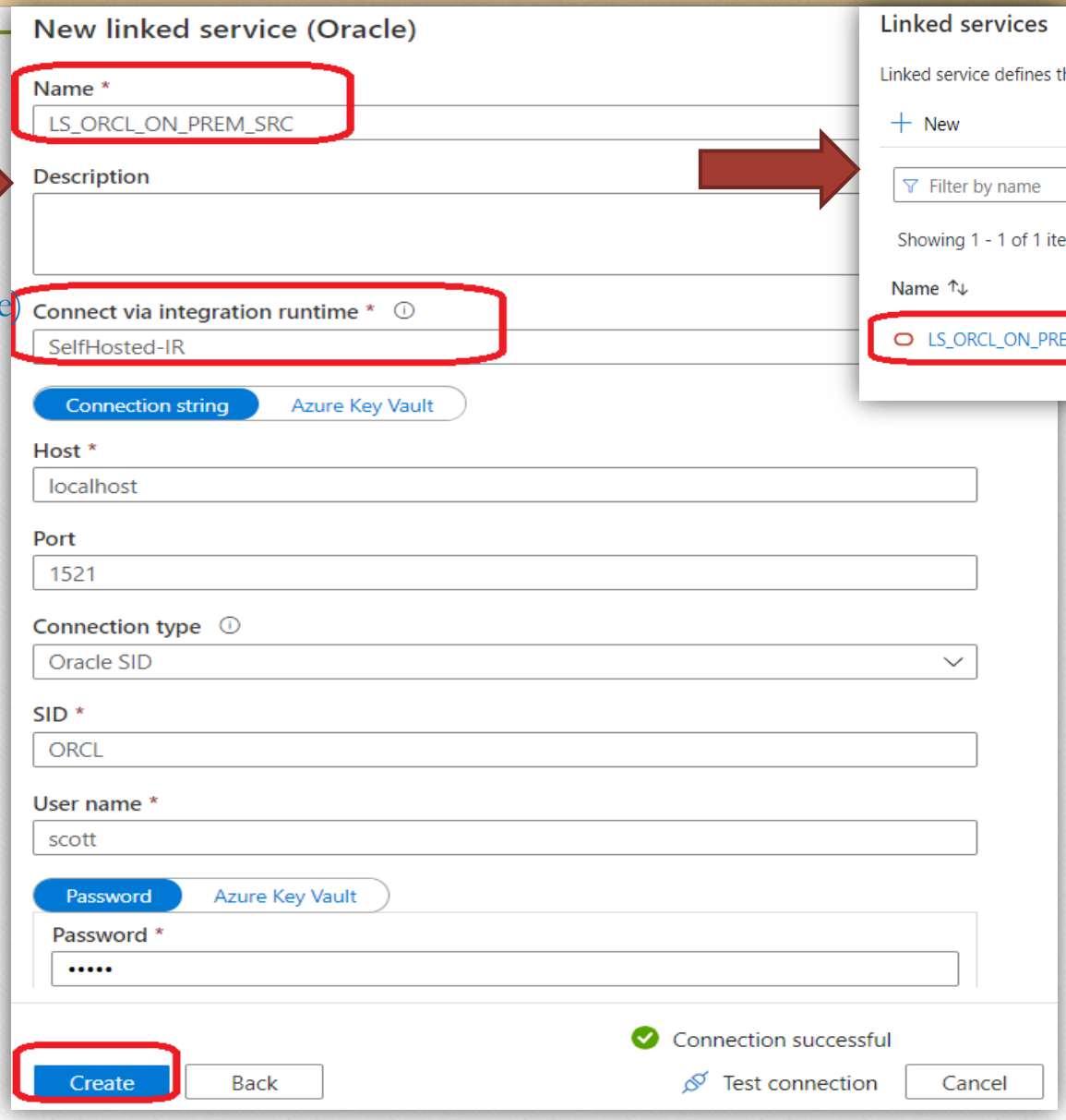


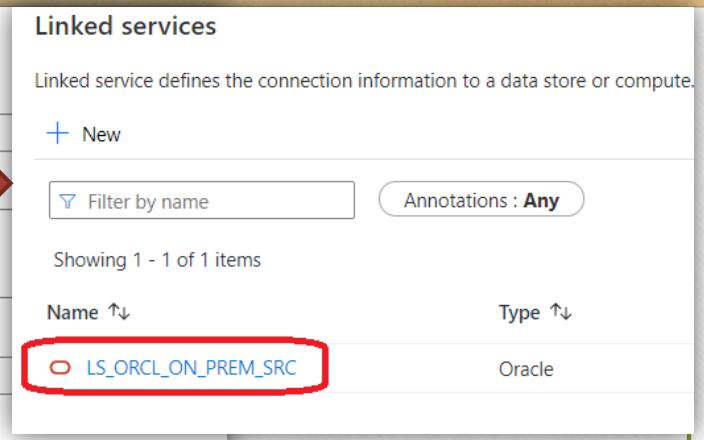
Creating Linked Service To Connect On-Premises Oracle Database to connect and transfer data from on-premises To Azure Blob Storage.





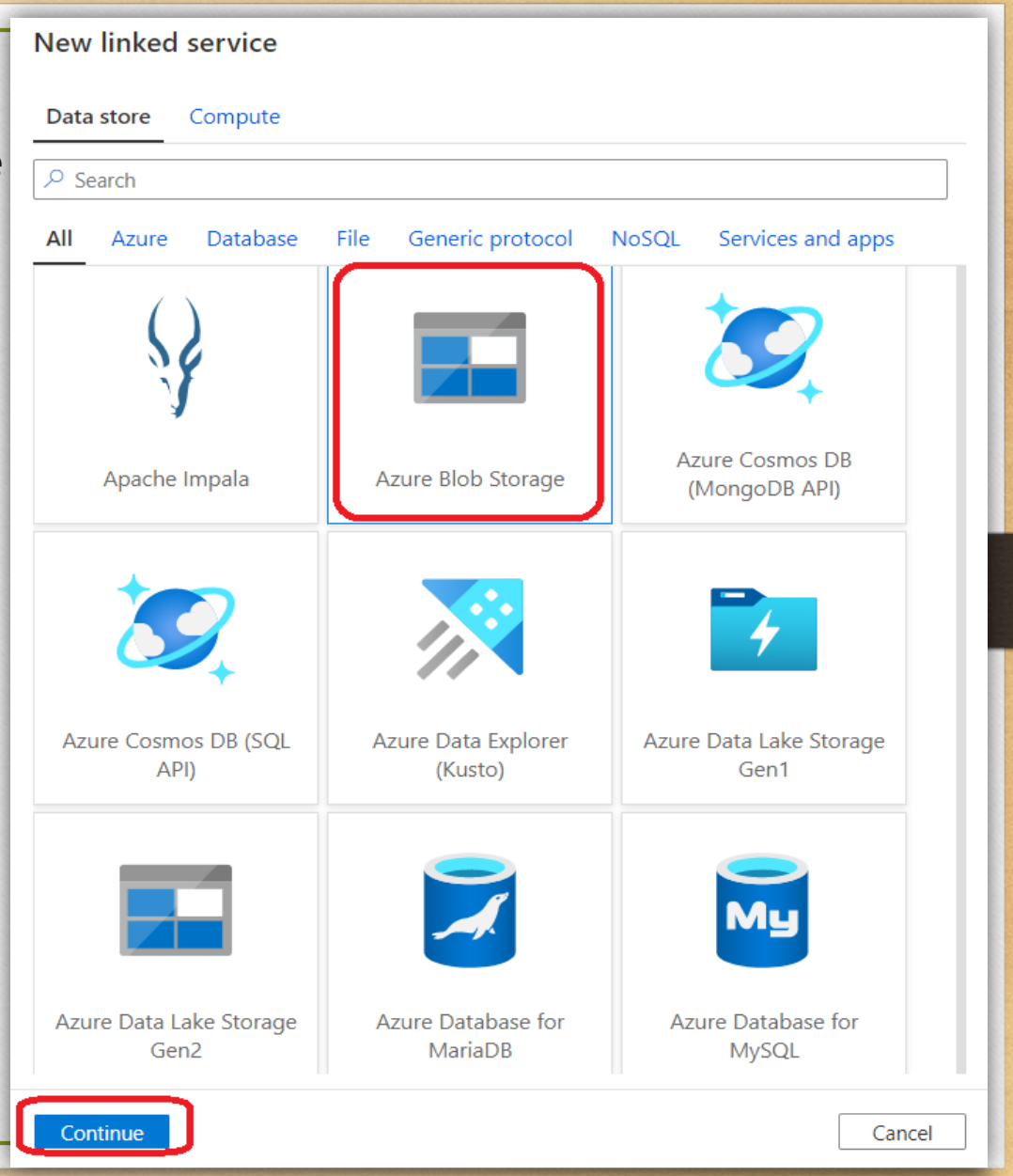
Name: User Defined Name
LS_ORCL_ON_PREM_SRC
Integration Runtime:
SelfHosted-IR
Oracle Host: (IP or Hostname)
Localhost (my local Oracle Host Name)
Port: (Oracle Port Number)
1521 (Oracle Default Port Number)
SIR: (Oracle Service ID)
ORCL
User Name:
Scott
Password:
scott



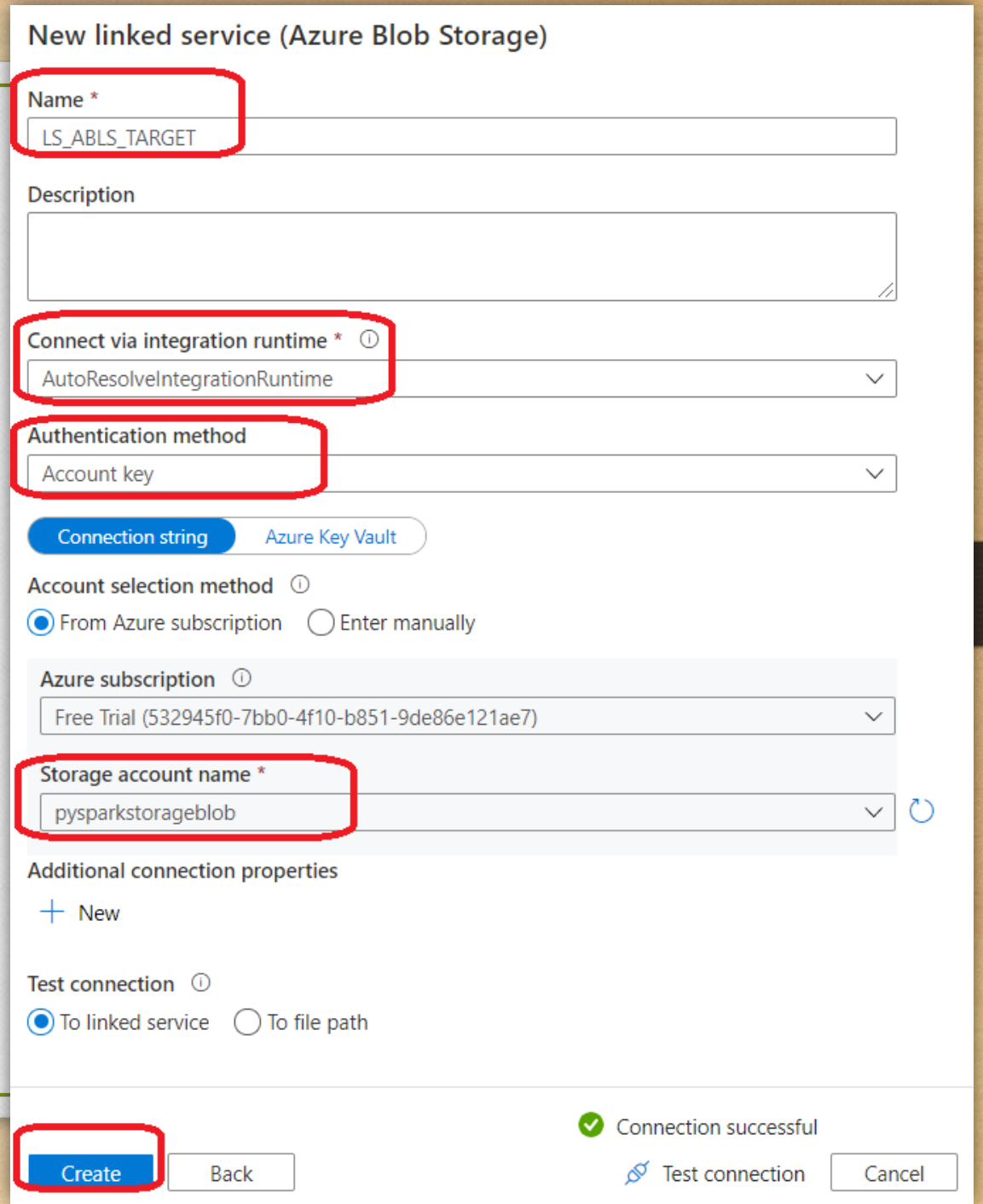


Create a linked service for Azure Blob Storage as Target

Create linked services in a data factory to link your data stores and compute services to the data factory. Create an Azure Storage linked service that is used as both the source and sink stores. The linked service has the connection information that the Data Factory service uses at runtime to connect to it and select data as **source** or store data as **target**.



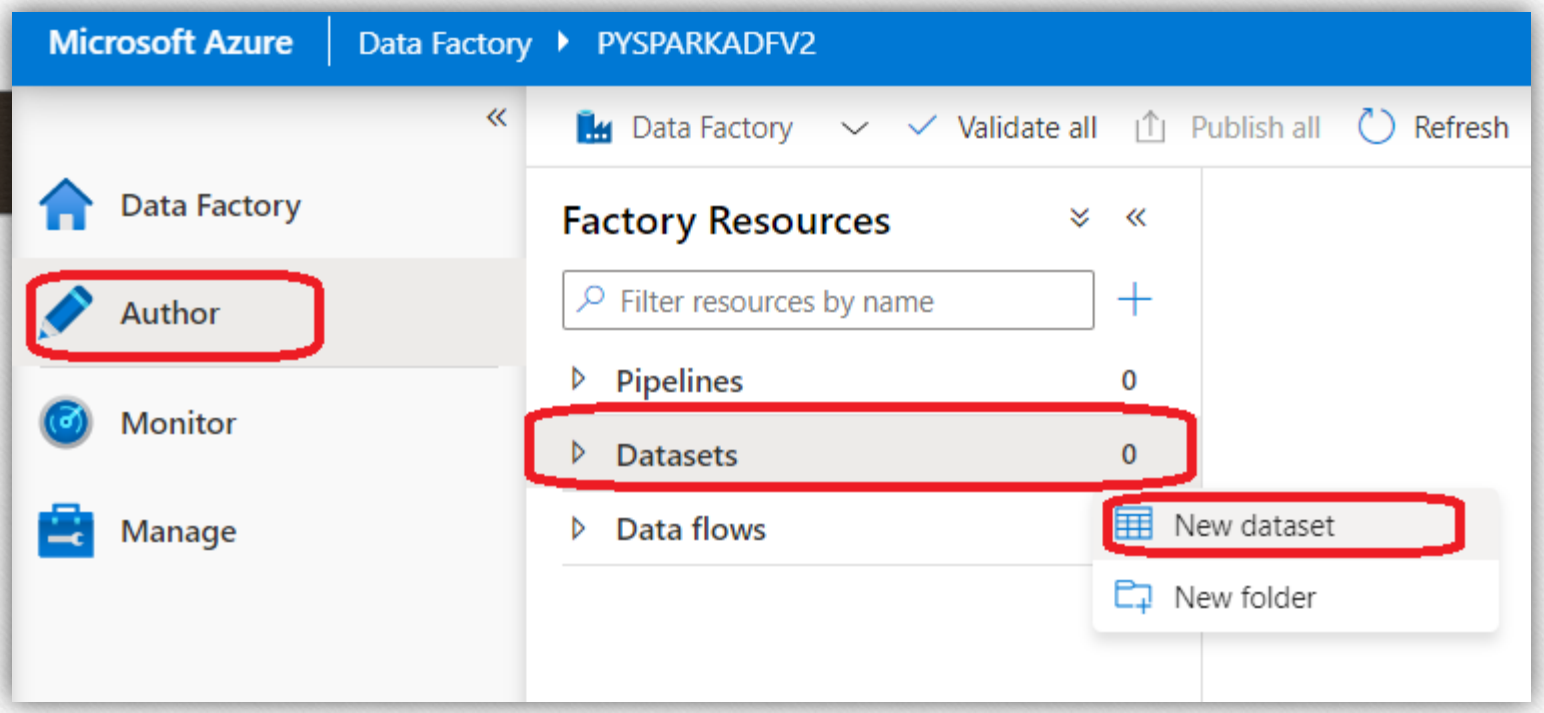
1.On the **New Linked Service** page, select **Azure Blob Storage**, and then select **Continue**.
2.On the New Linked Service (Azure Blob Storage) page, complete the following steps:
3.For **Name**, enter **LS_ABLS_TARGET**.
4.Authentication method : **Account Key**
5.For **Storage account name**, select the name of your Azure Storage account.
6.Select **Test connection** to confirm that the Data Factory service can connect to the storage account.
7.Select **Create** to save the linked service.



Creating Two Datasets for **Source** And **Target**.

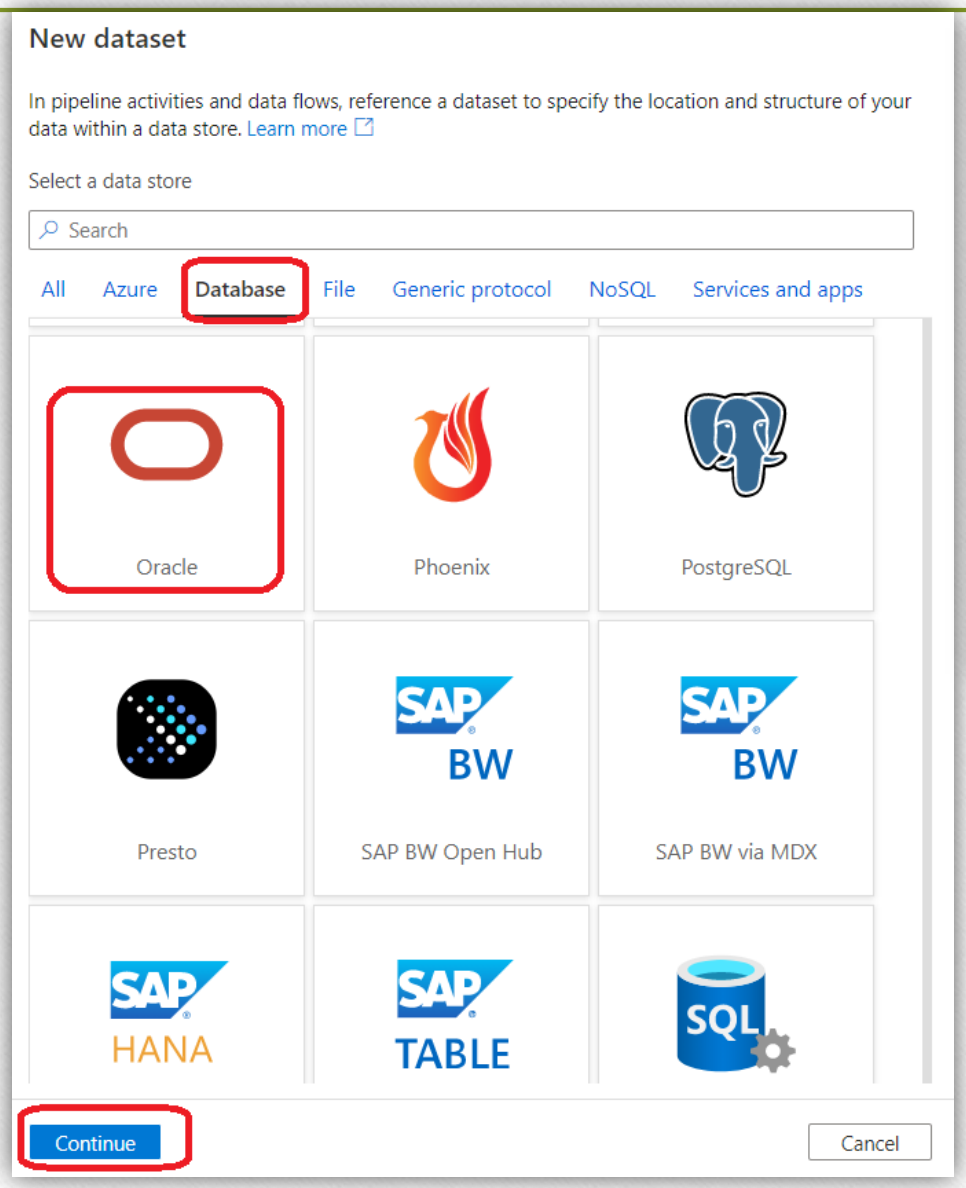
1) One **dataset** for Source Oracle Database Table (scott.emp table)
2) One dataset for Target Azure Blog Storage File (creating target data file as emp.csv)

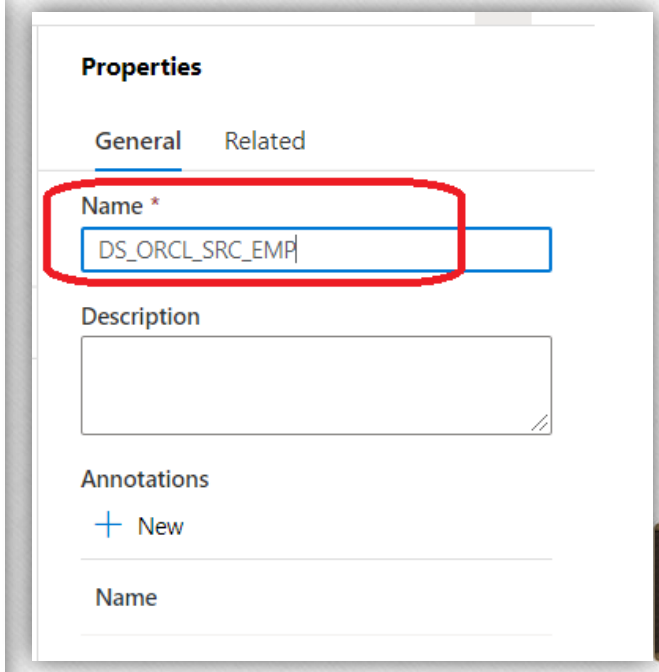
Go To=> Author => Datasets => New Dataset



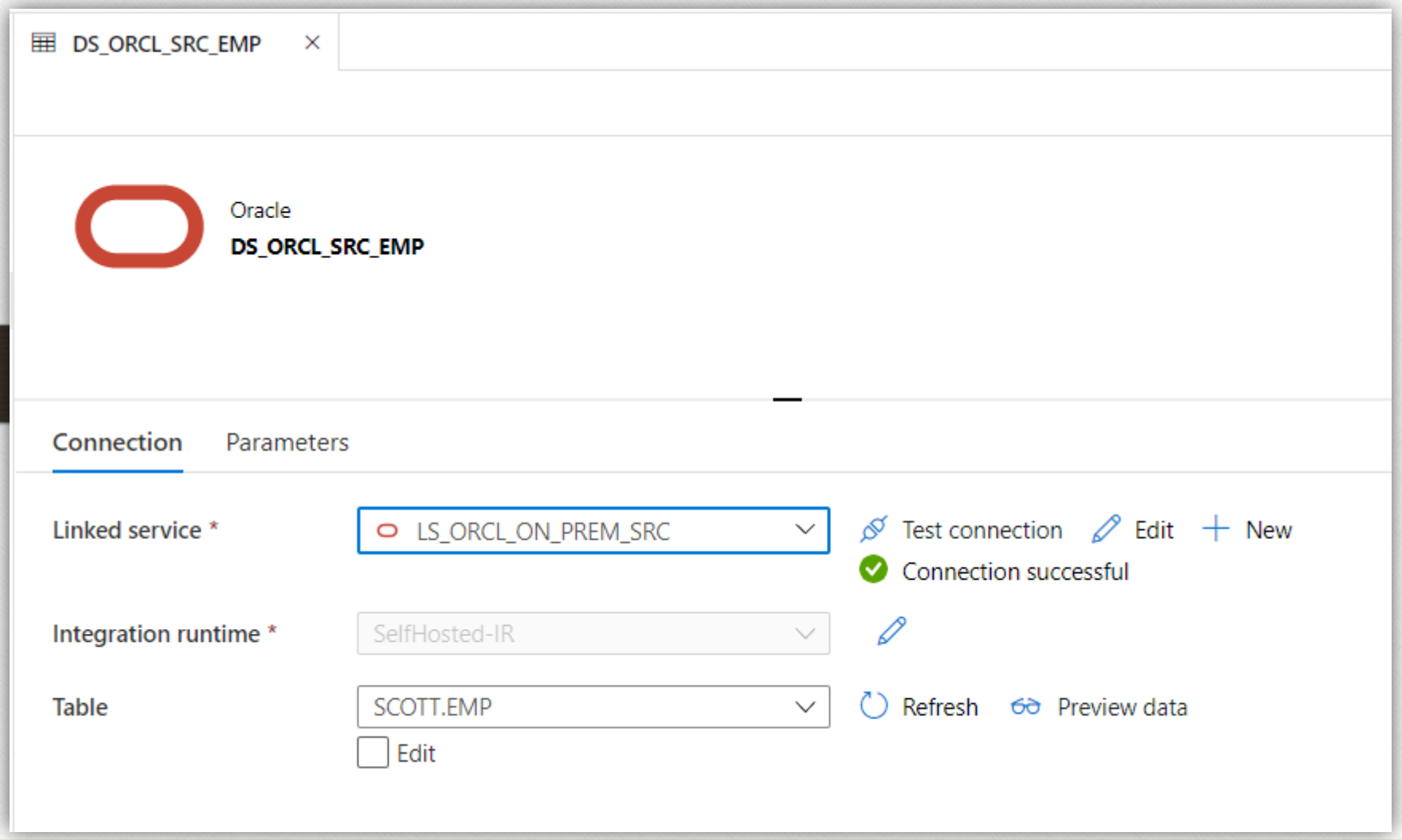
Goto Database tab =>
Select Oracle And Click on Continue.

Enter The Dataset Name As
DS_ORCL_SRC_EMP



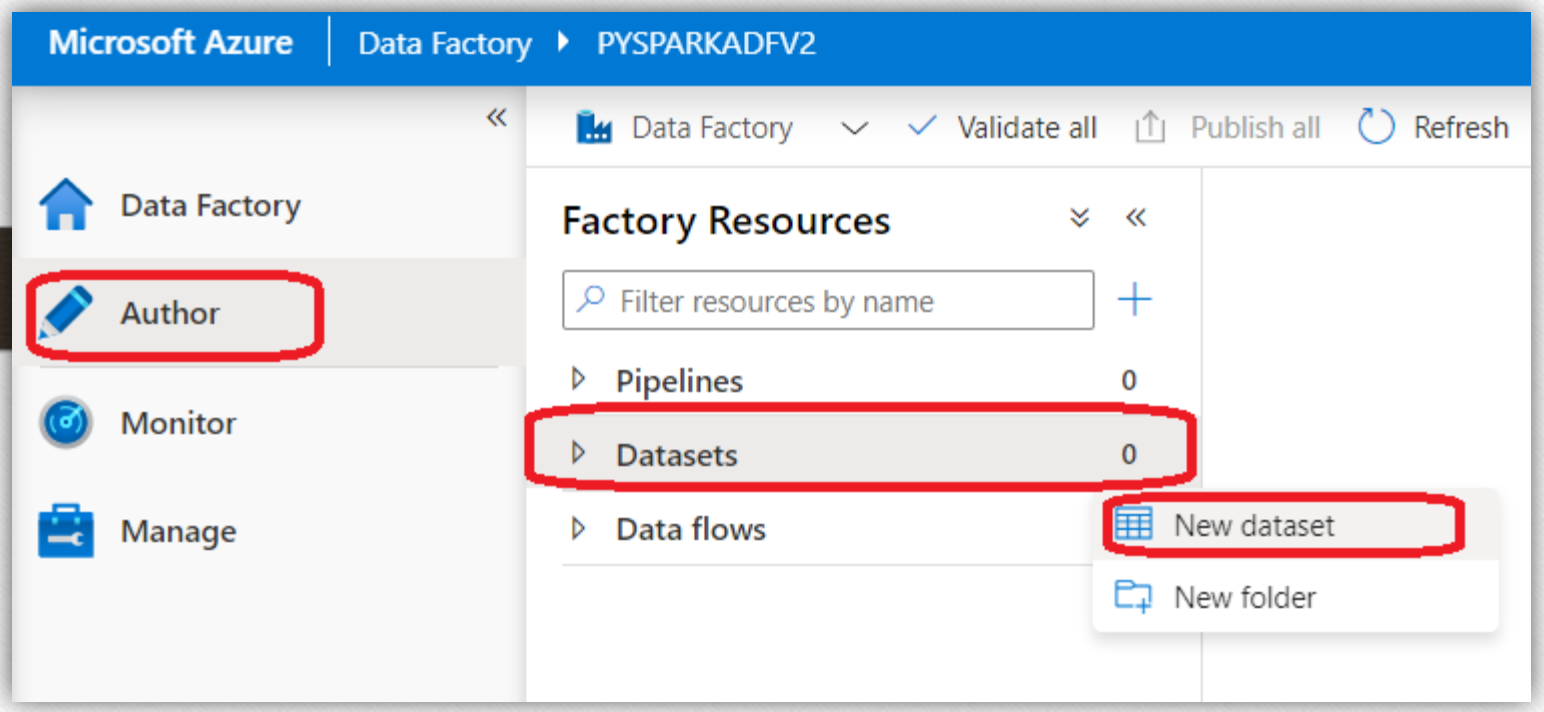


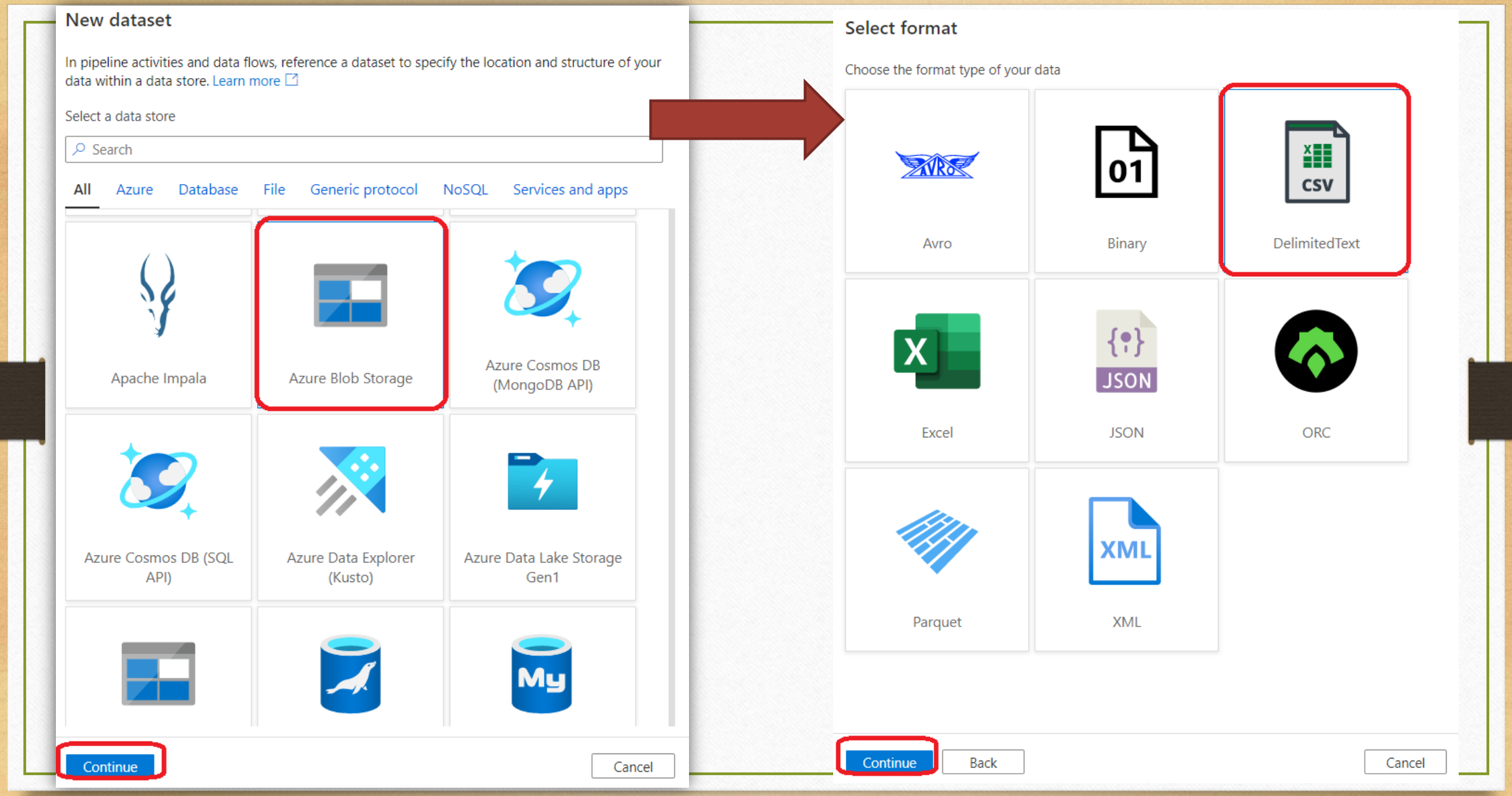
Select The Linked Service Name as : **LS_ORCL_ON_PREM_SRC** And Select The Table **SCOTT.EMP**



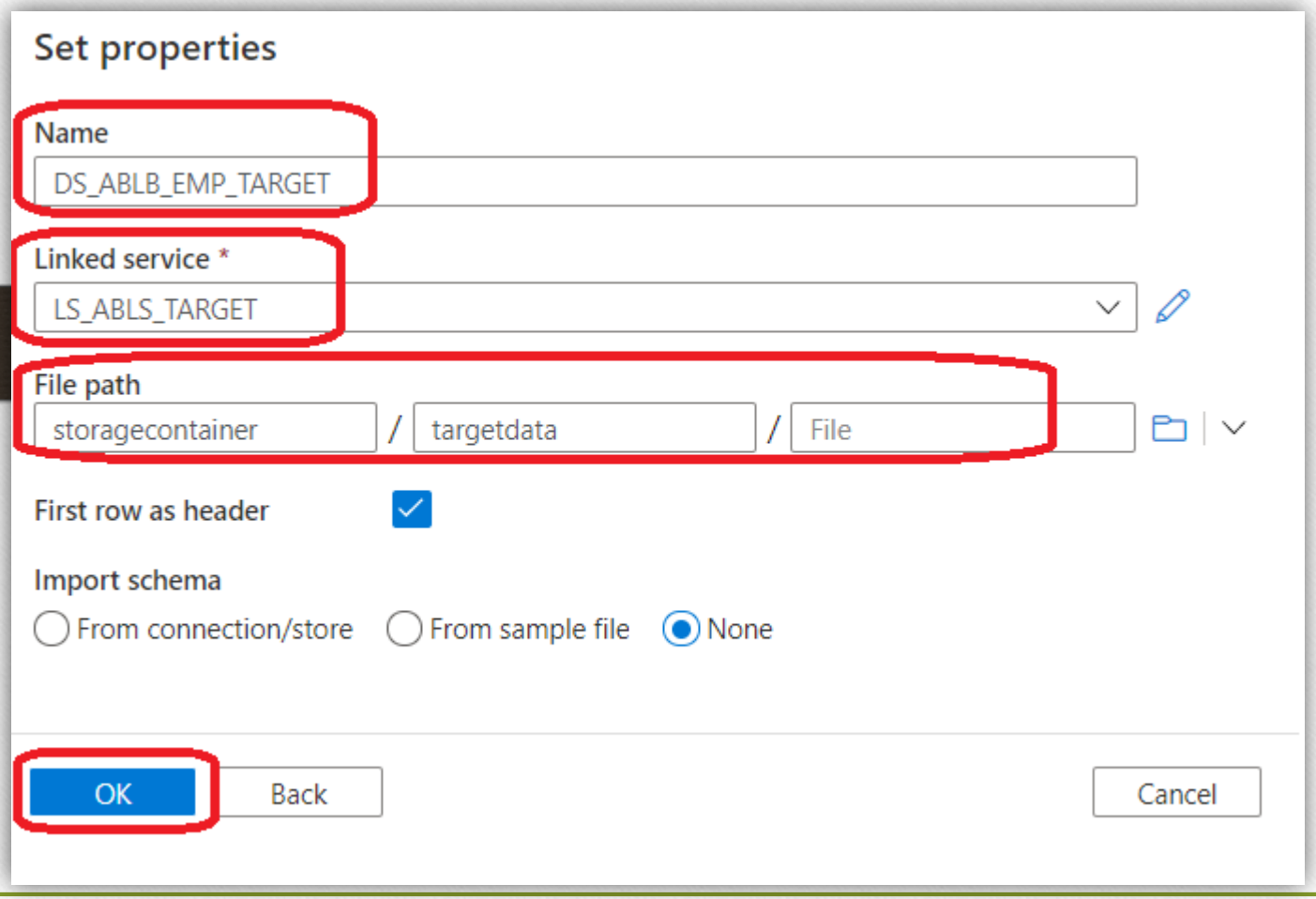
Creating Another Dataset For Target File in Azure Blob Storage.

Go To=> Author => Datasets => New Dataset

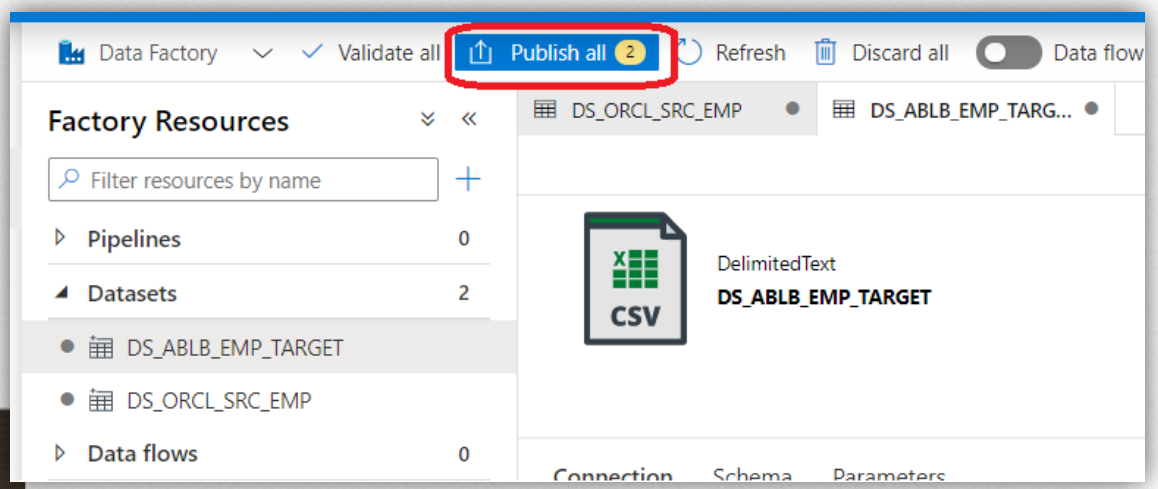


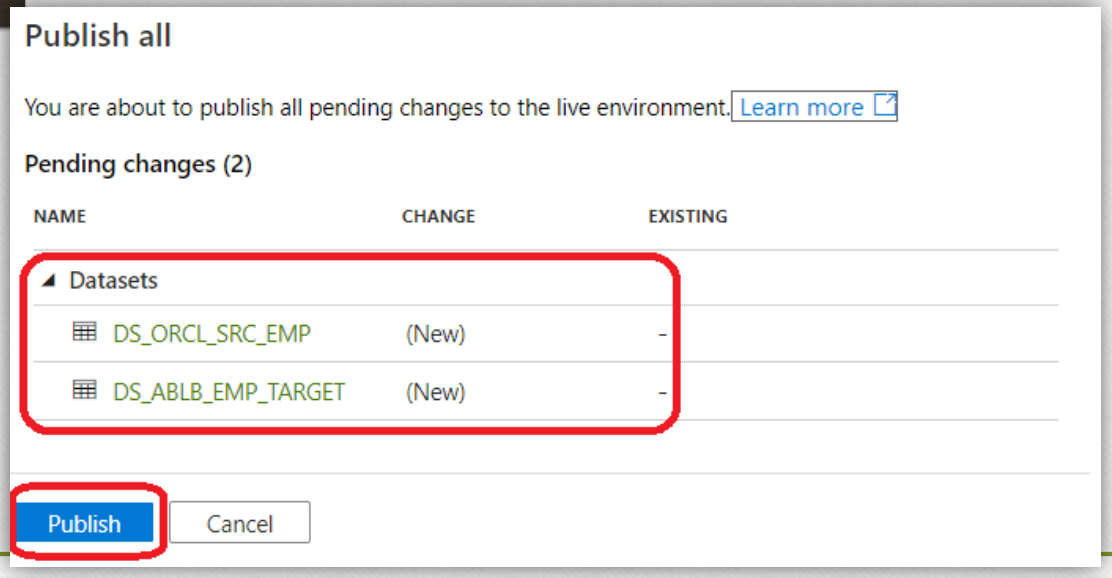

New dataset
In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. Learn more
Select a data store
Search
All
Azure
Database
File
Generic protocol
NoSQL
Services and apps
Apache Impala
Azure Blob Storage
Azure Cosmos DB (MongoDB API)
Azure Cosmos DB (SQL API)
Azure Data Explorer (Kusto)
Azure Data Lake Storage Gen1
Continue
Cancel
Select format
Choose the format type of your data
Avro
Binary
DelimitedText
Excel
JSON
ORC
Parquet
XML
Continue
Back
Cancel

Creating Dataset for Azure blob storage file.
Name: DS_ABLB_EMP_TARGET
Linked Service: LS_ABLS_TARGET And Container Name: select existing container and folder name to store the file.



Click on Publish All to save created Datasets.





Create a pipeline

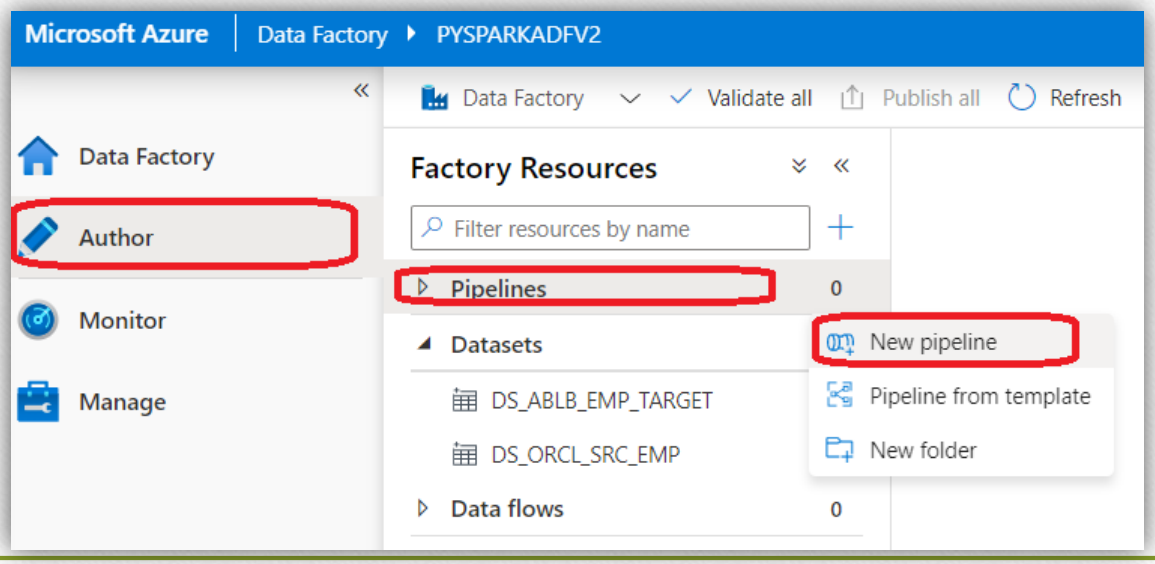
In this procedure, you create and validate a pipeline with a copy activity that uses the input and output datasets. The copy activity copies data from the file you specified in the input dataset settings to the file you specified in the output dataset settings. If the input dataset specifies only a folder (not the file name), the copy activity copies all the files in the source folder to the destination.

1.Select the **+** (plus) button, and then select **Pipeline**.

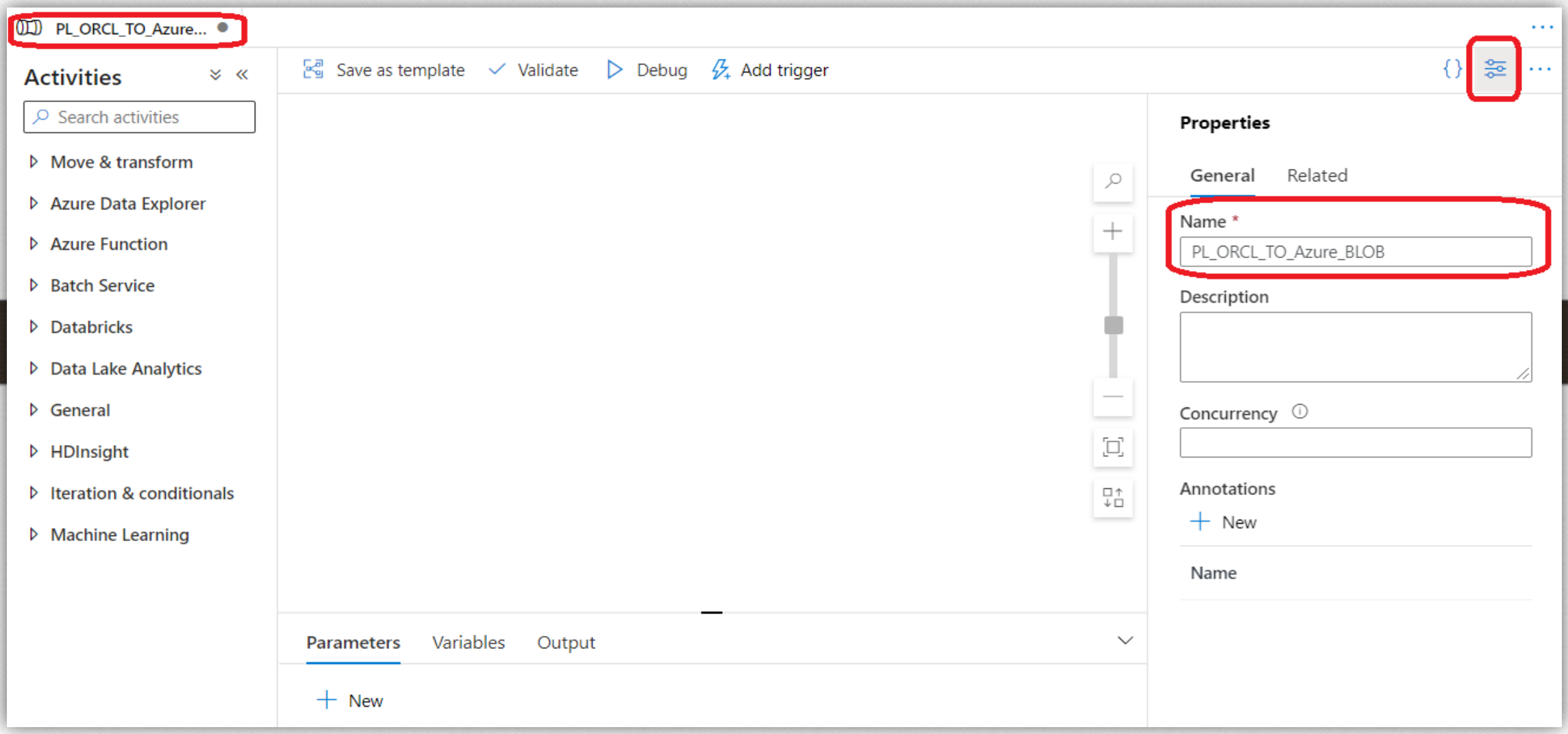
2.In the General panel under **Properties**, specify **CopyPipeline** for **Name**. Then collapse the panel by clicking the Properties icon in the top-right corner.

3.In the **Activities** toolbox, expand **Move & Transform**. Drag the **Copy Data** activity from the **Activities** toolbox to the pipeline designer surface. You can also search for activities in the **Activities** toolbox.

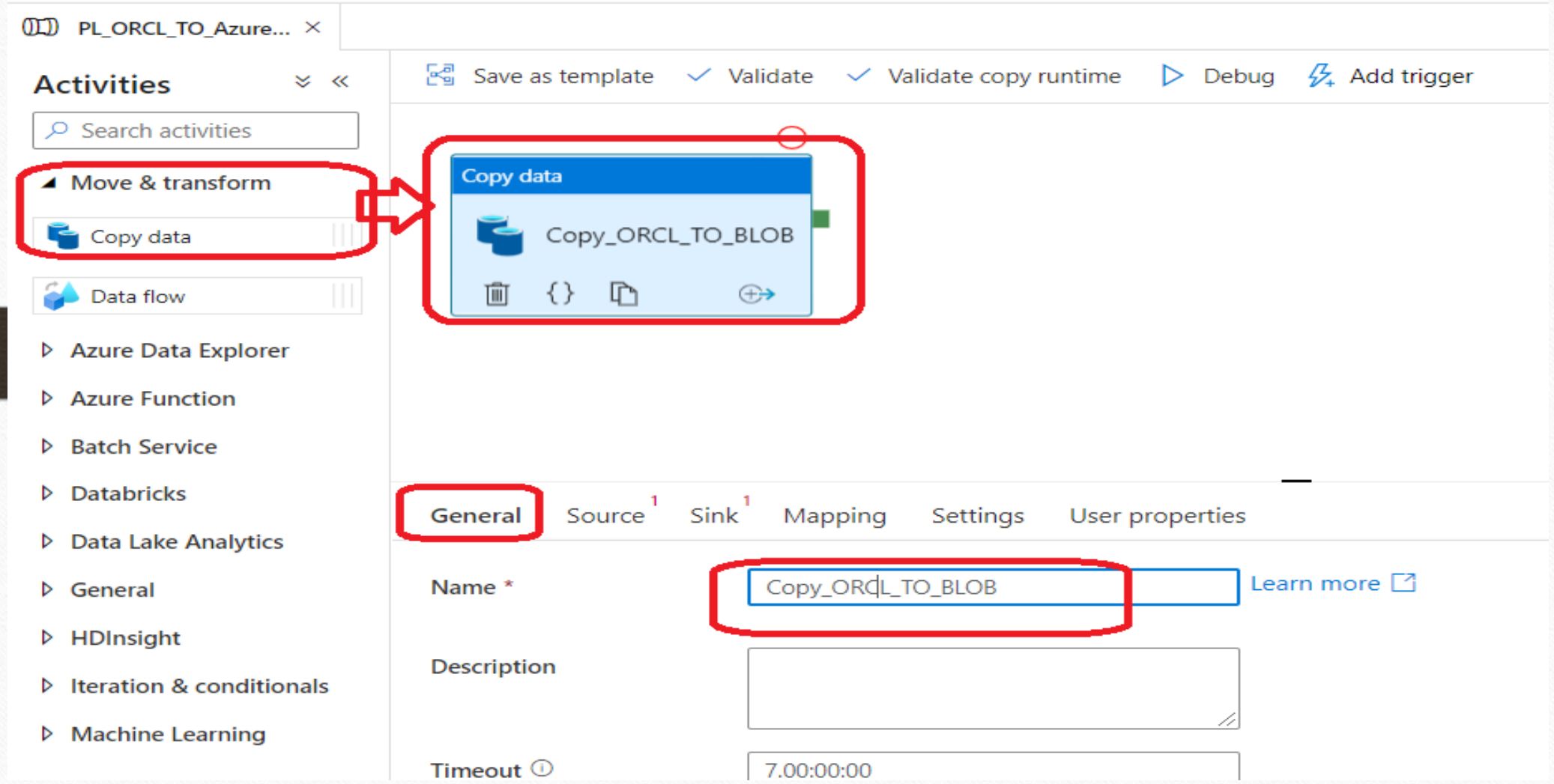
Specify **CopyFromBlobToBlob** for **Name**.



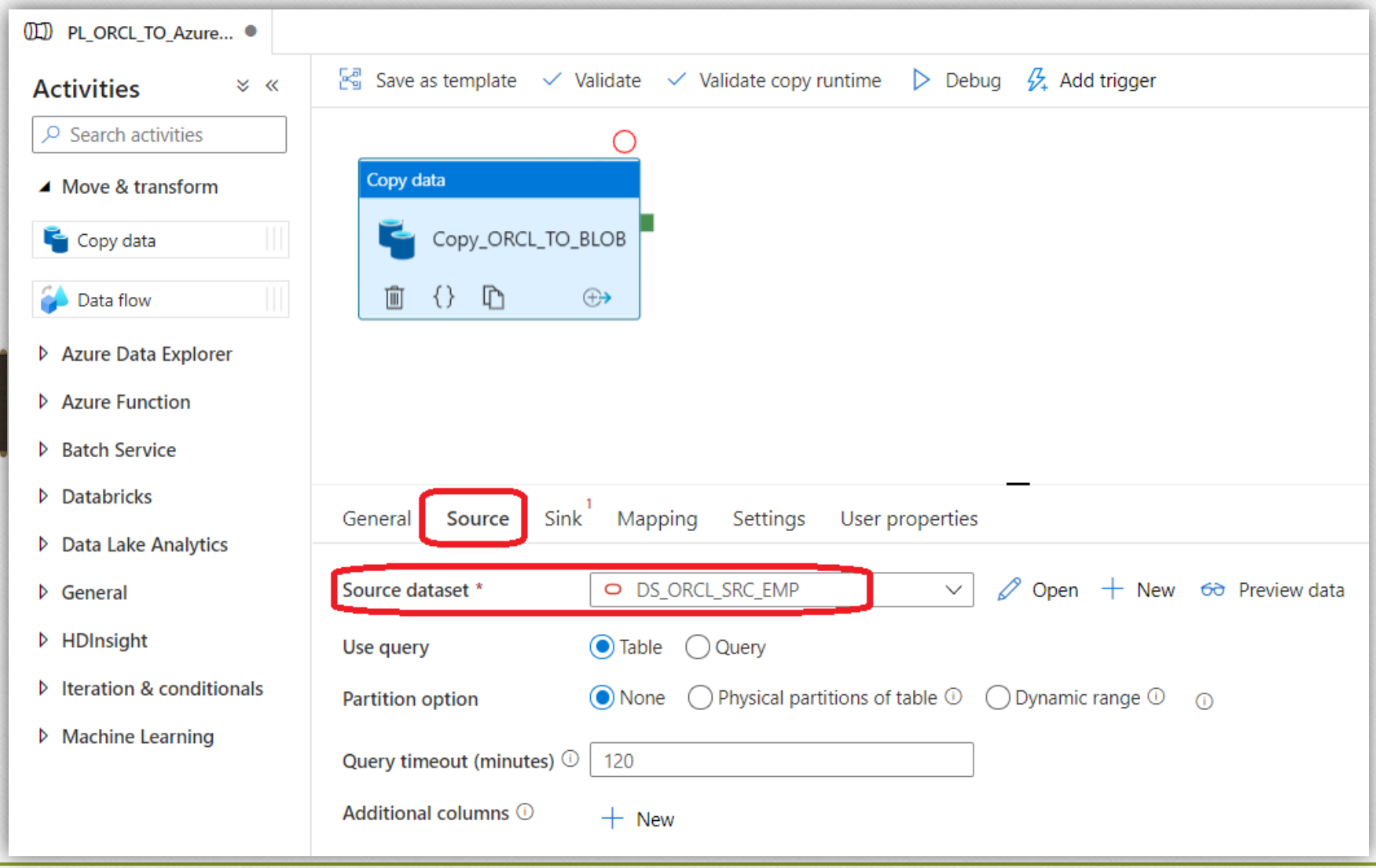
Enter The Pipeline Name: **PL_ORCL_TO_Azure_BLOB**



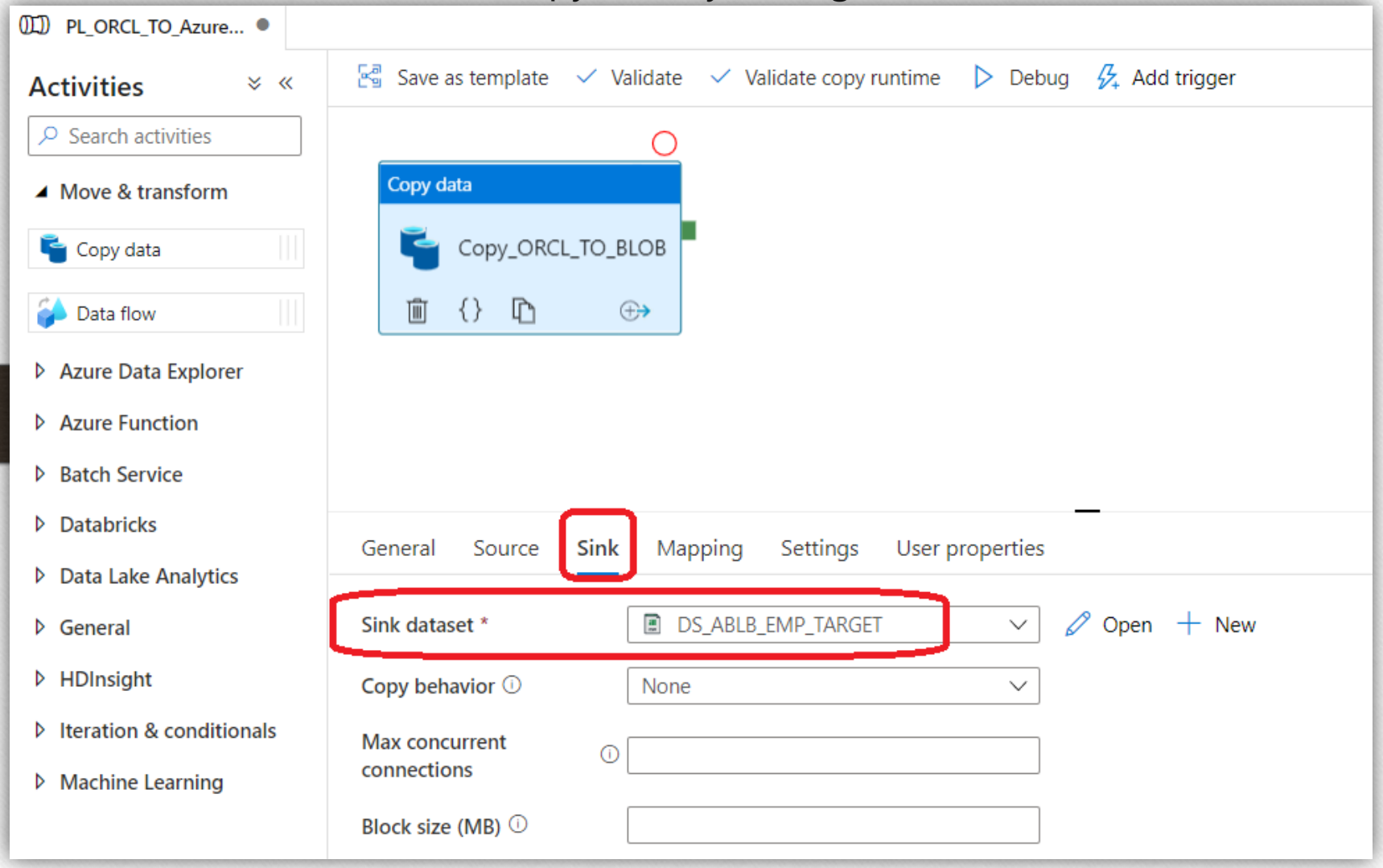
In the **Activities** toolbox, expand **Move & Transform**. Drag the **Copy Data** activity from the **Activities** toolbox to the pipeline **designer surface**. You can also search for activities in the **Activities** toolbox. Specify **Copy_ORCL_TO_BLOB** for **Name**.



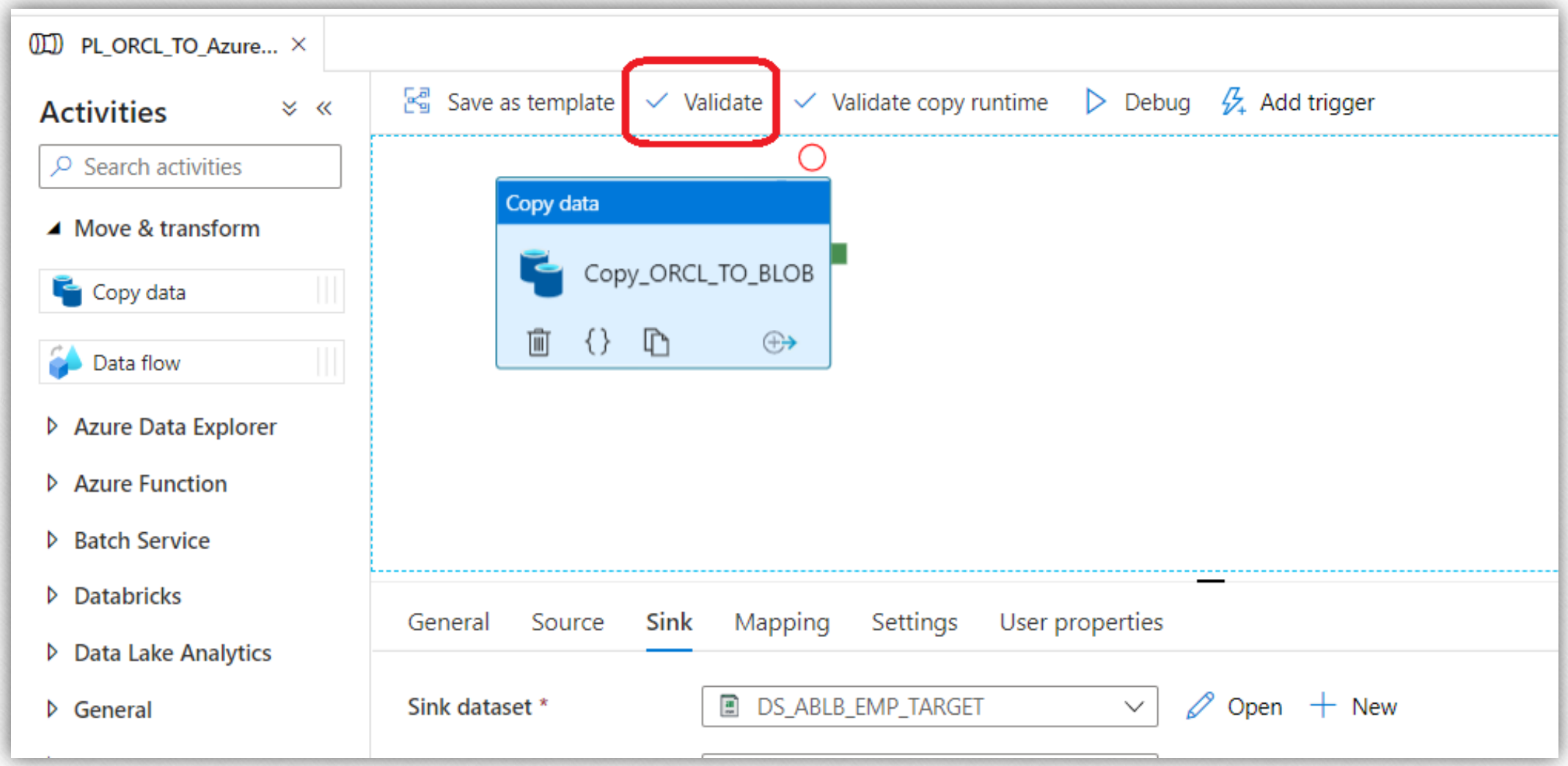
Switch to the **Source** tab in the copy activity settings, and select **DS_ORCL_SRC_EMP** for **Source Dataset**.



Switch to the **Sink** tab in the copy activity settings, and select **DS_ABLB_EMP_TARGET** for **Sink Dataset**.



Click **Validate** on the pipeline toolbar above the canvas to validate the pipeline settings. Confirm that the pipeline has been successfully validated. To close the validation output, select the Validation button in the top-right corner.

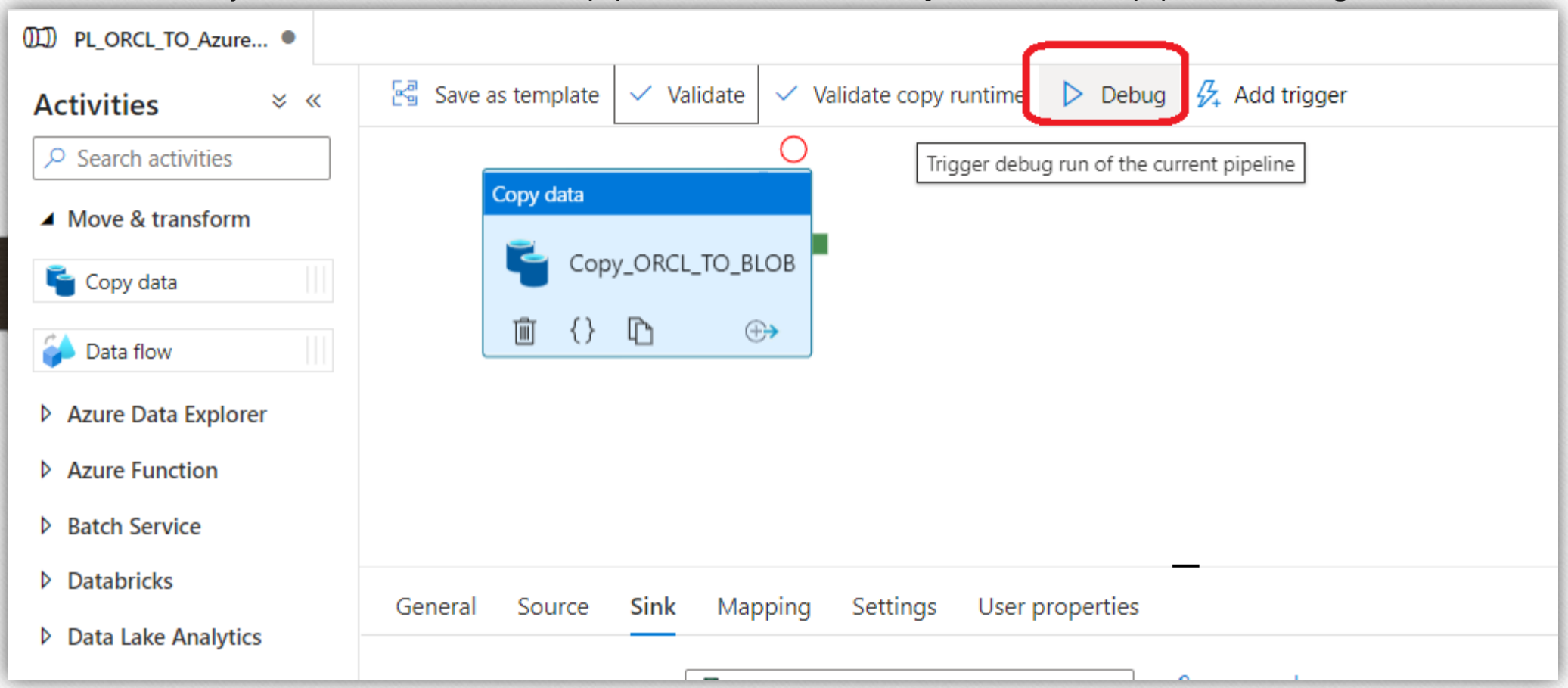


Debug the pipeline

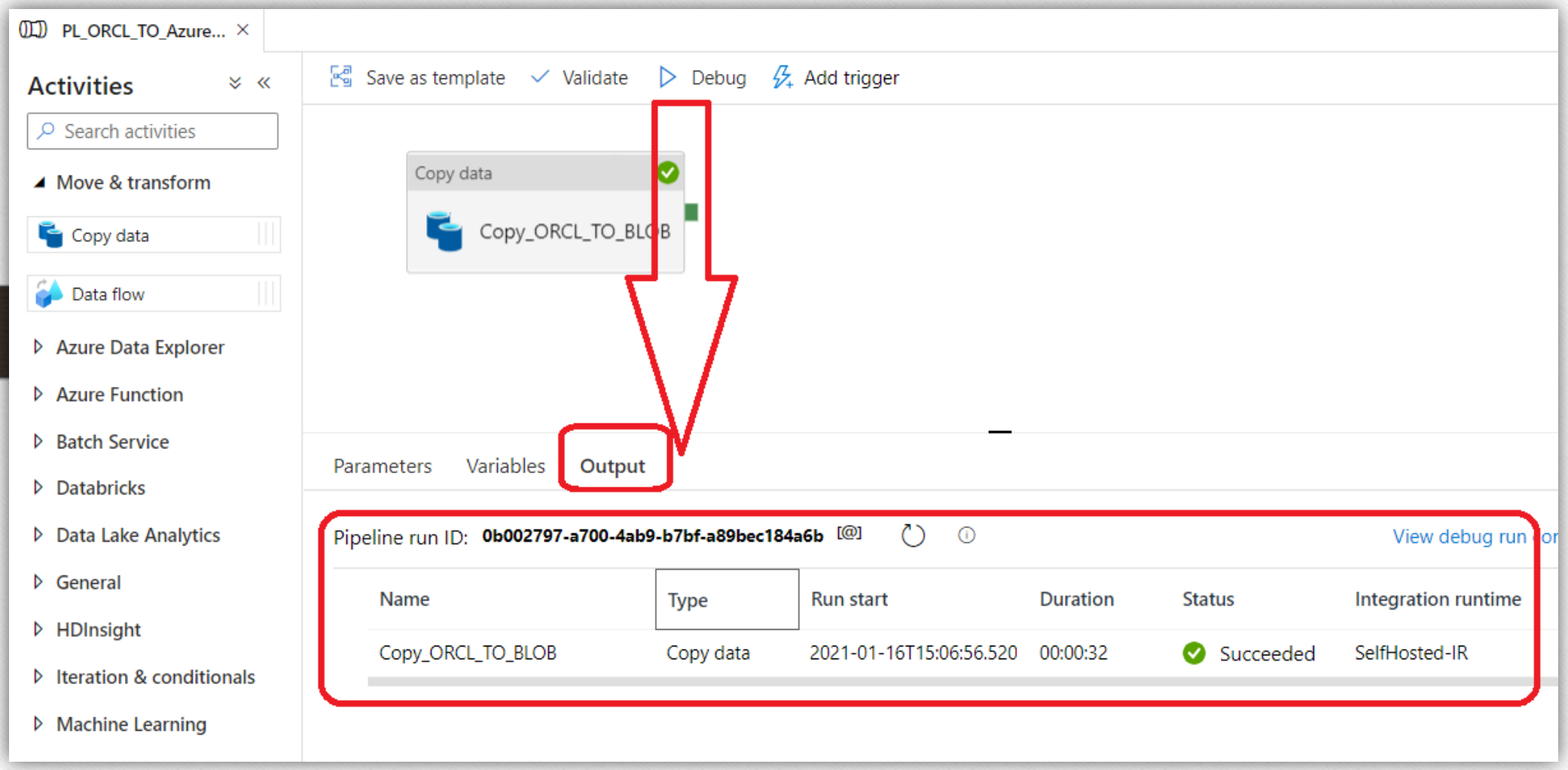
In this step, you debug the pipeline before deploying it to Data Factory.

1.On the pipeline toolbar above the canvas, click **Debug** to trigger a test run.

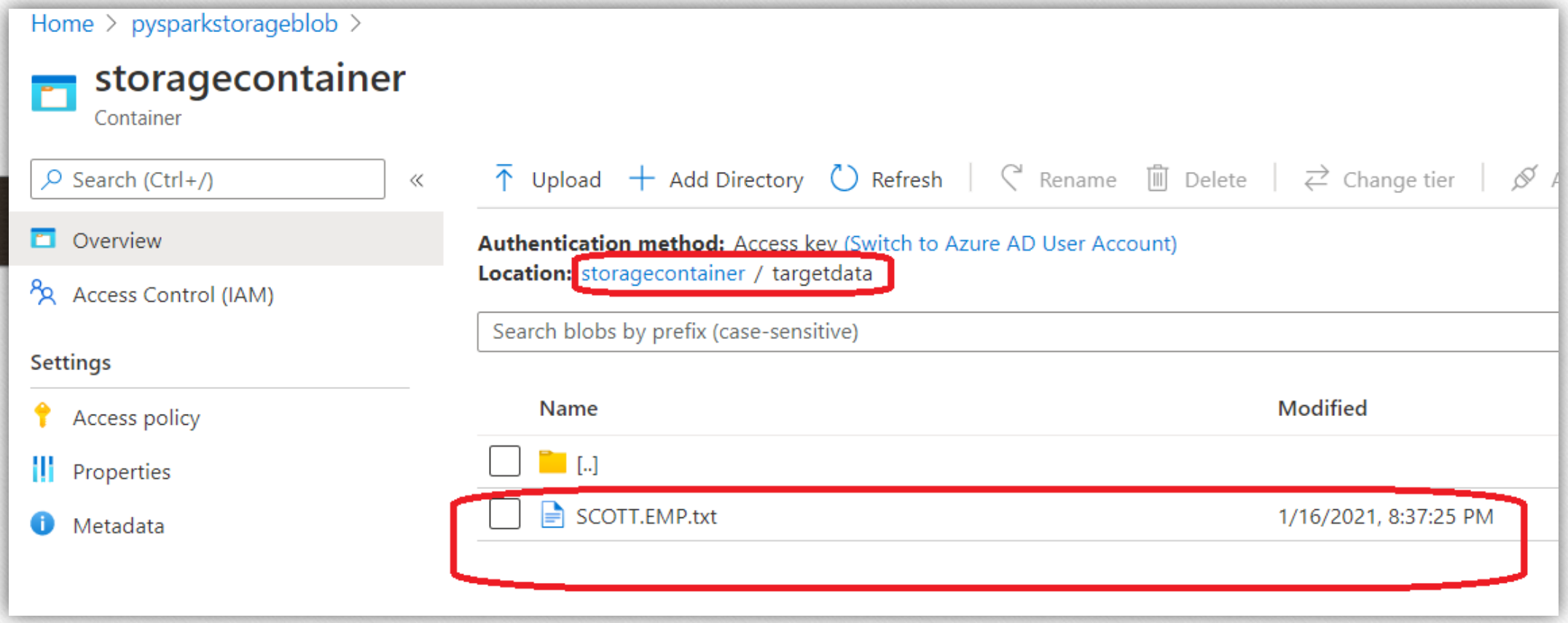
2.Confirm that you see the status of the pipeline run on the **Output** tab of the pipeline settings at the bottom



Confirm that you see an output file in the **targetdata** folder of the **storagecontainer** container. If the output folder doesn't exist, the Data Factory service automatically creates it.

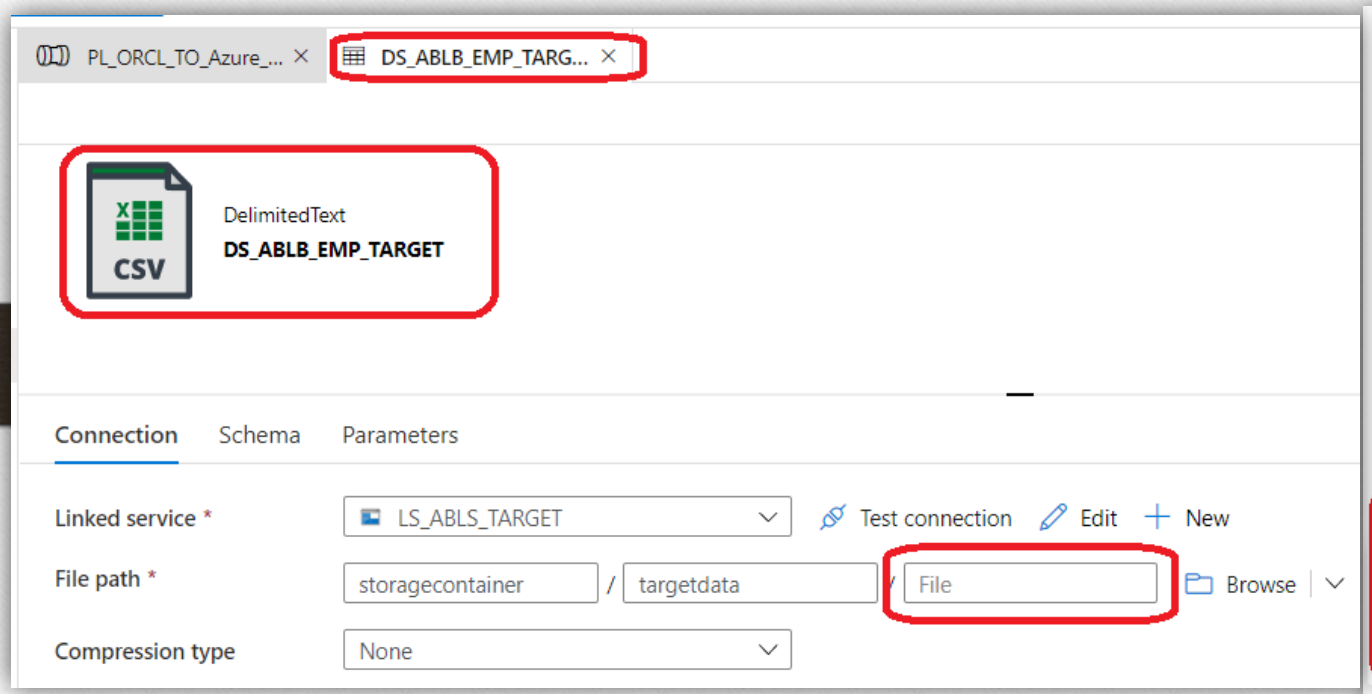


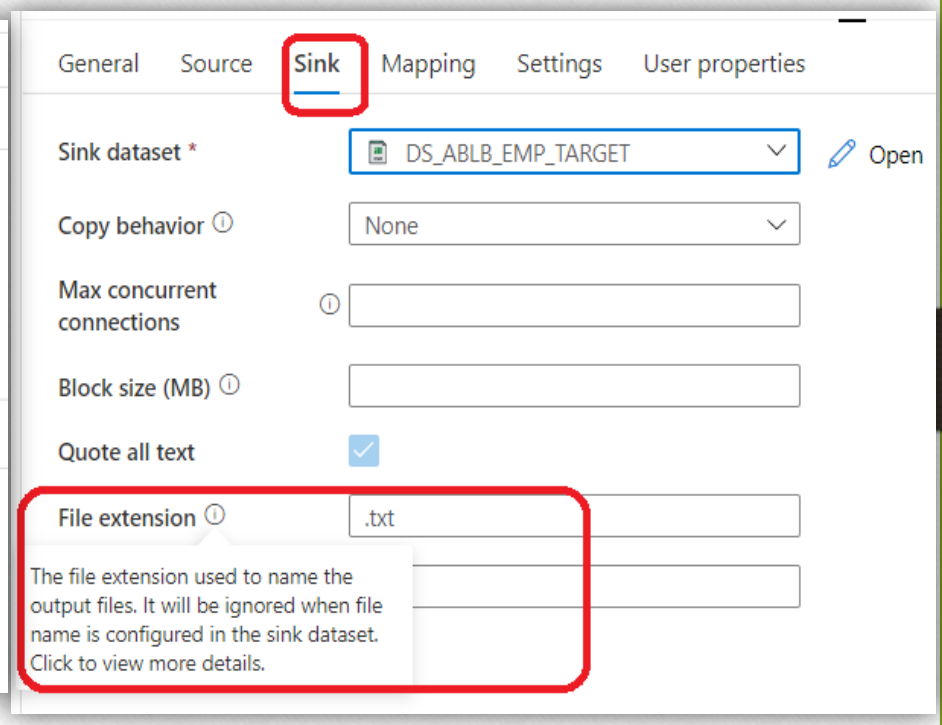
Verify Output Data file in target blob location as per Dataset **DS_ABLB_EMP_TARGET**



Target File Name with Extension: If you are not specified Target file name in Dataset, it will consider in Copy Activity Level File Extention.

If we specify at Target Dataset level filename, then it will be ignored.



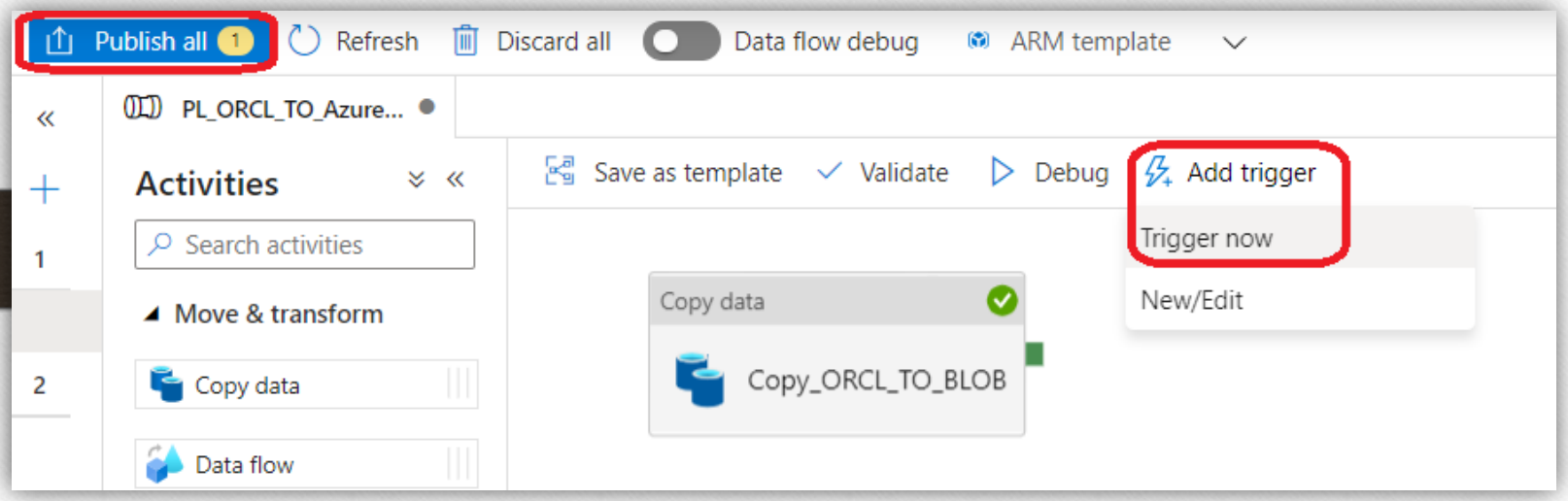


Trigger the pipeline manually

In this procedure, you deploy entities (linked services, datasets, pipelines) to Azure Data Factory. Then, you manually trigger a pipeline run.

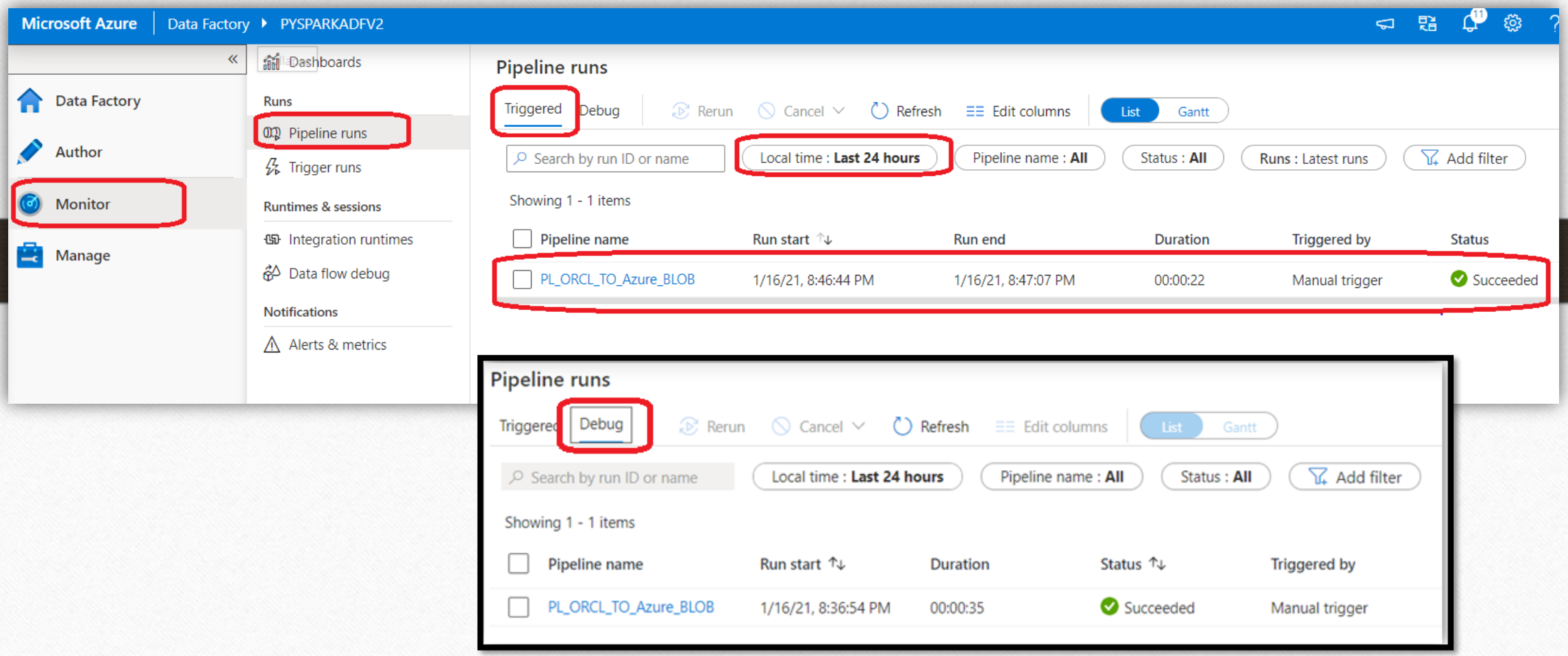
1.Before you trigger a pipeline, you must publish entities to Data Factory. To publish, select **Publish all** on the top.

2.To trigger the pipeline manually, select **Add Trigger** on the pipeline toolbar, and then select **Trigger Now**. On the **Pipeline run** page, select **OK**.

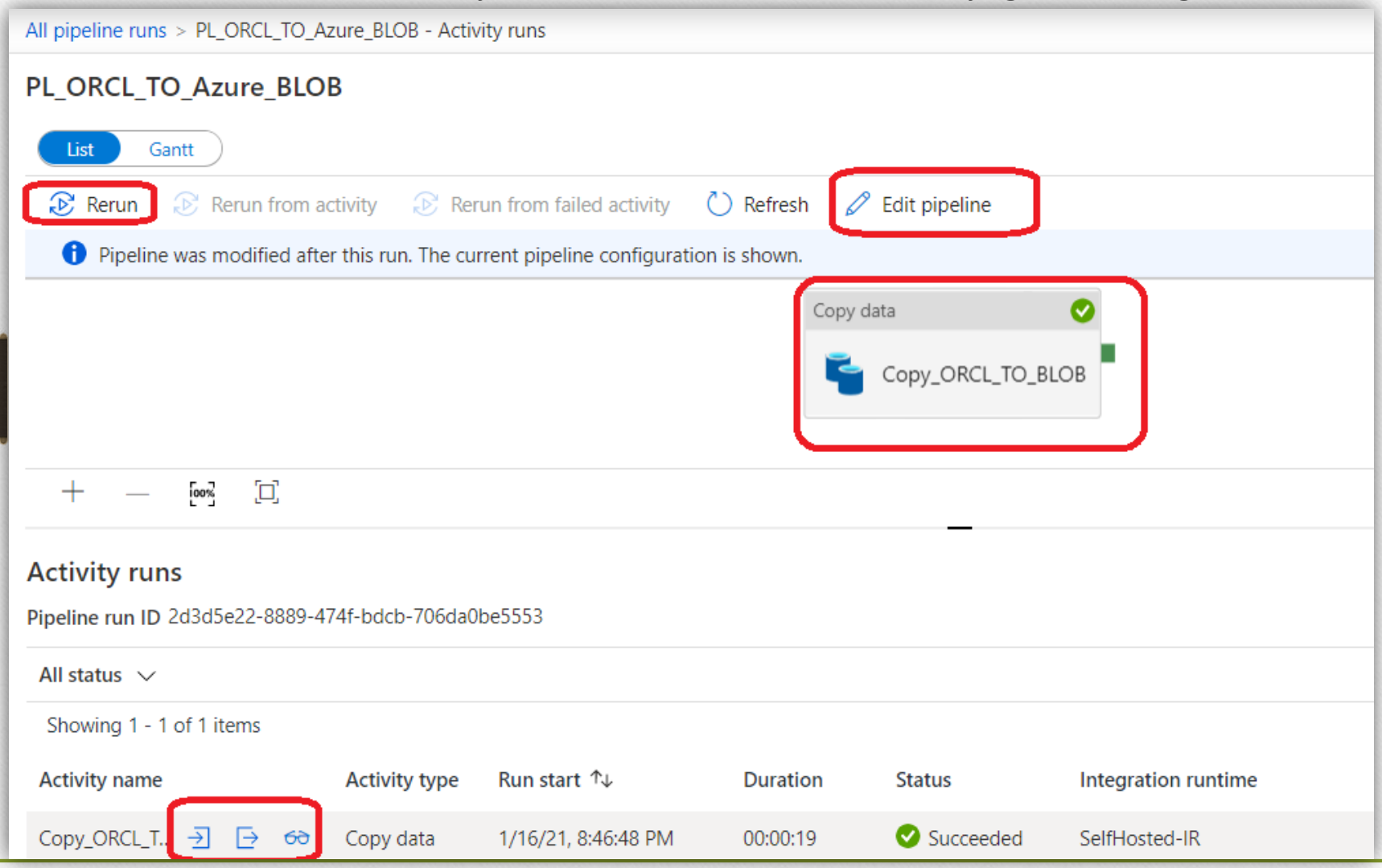


Monitor the pipeline

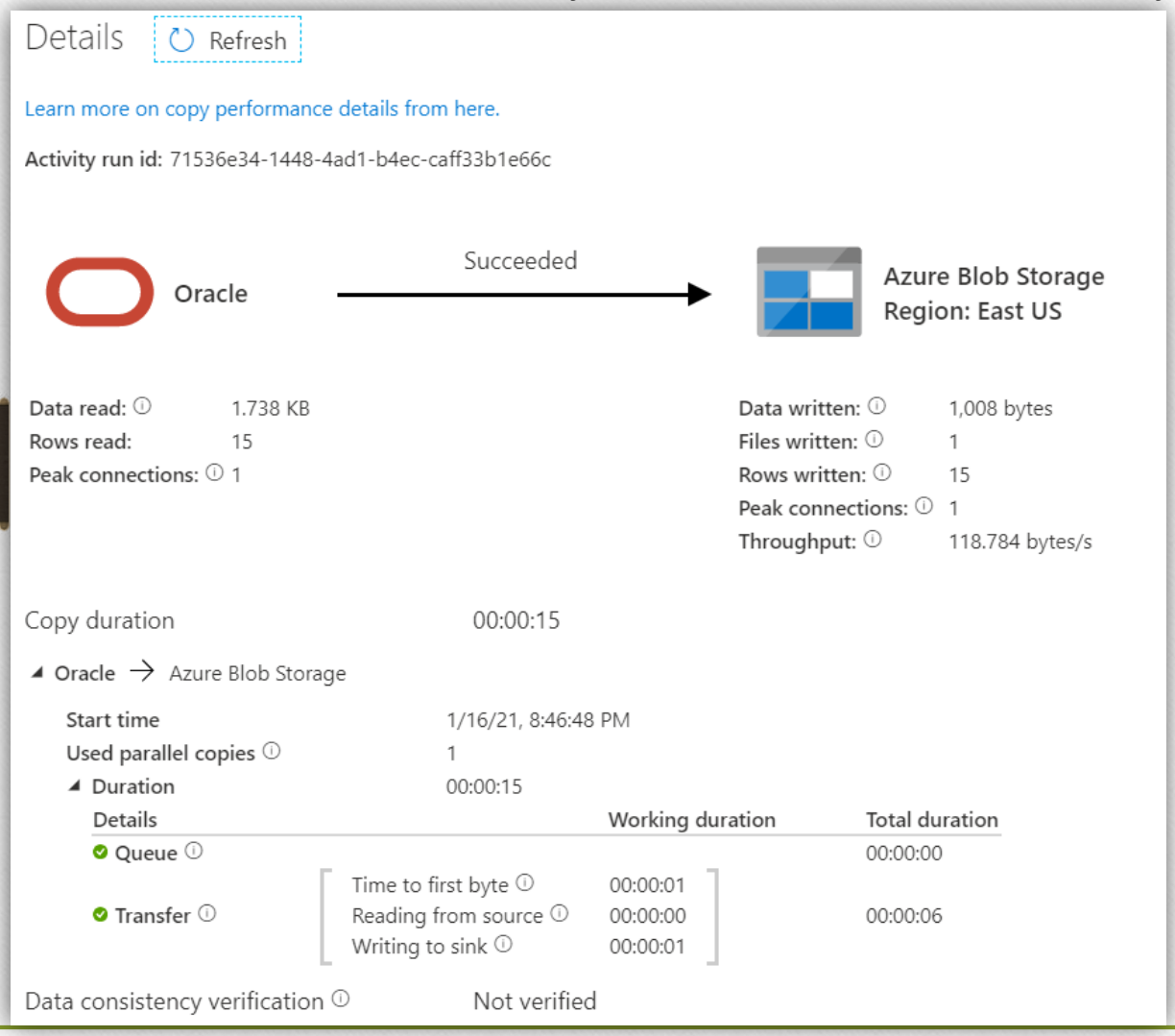
Switch to the **Monitor** tab on the left. Use the **Refresh** button to refresh the list. Triggered log will be in triggered tab and Debug log will be in Debug tab.



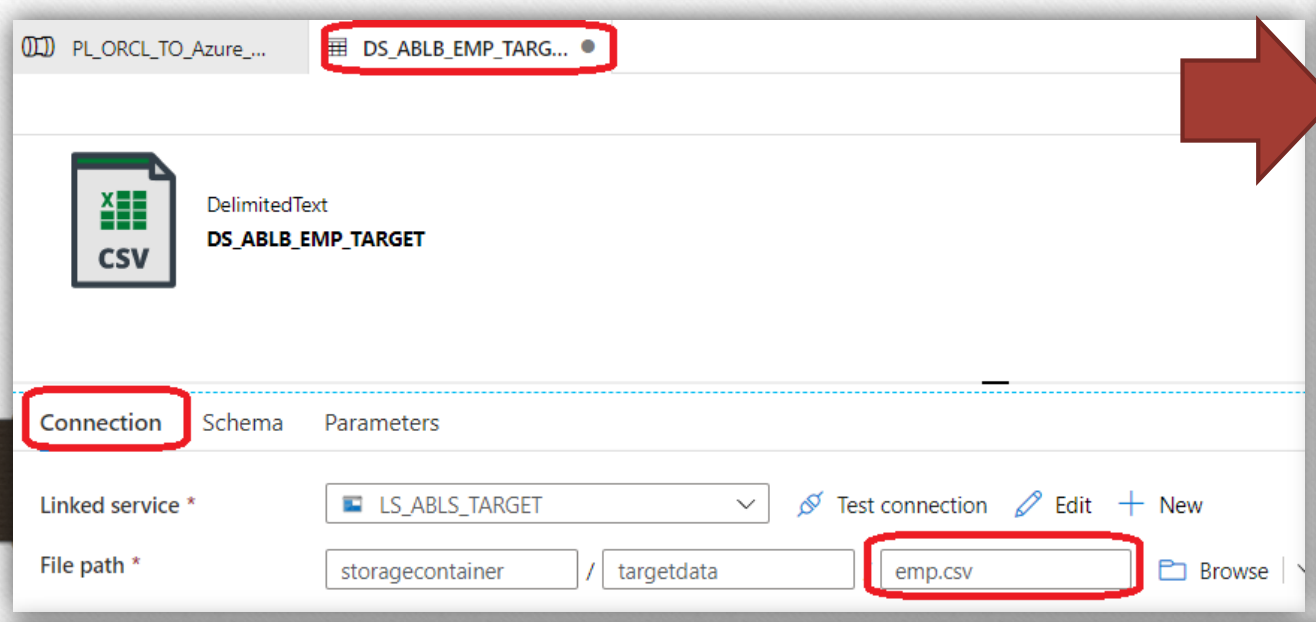
1.Select the **Copy_ORCL_TO_BLOB** link, you'll see the status of the copy activity run on this page.
2.To view details about the copy operation, select the **Details** (eyeglasses image) link. For details about the properties

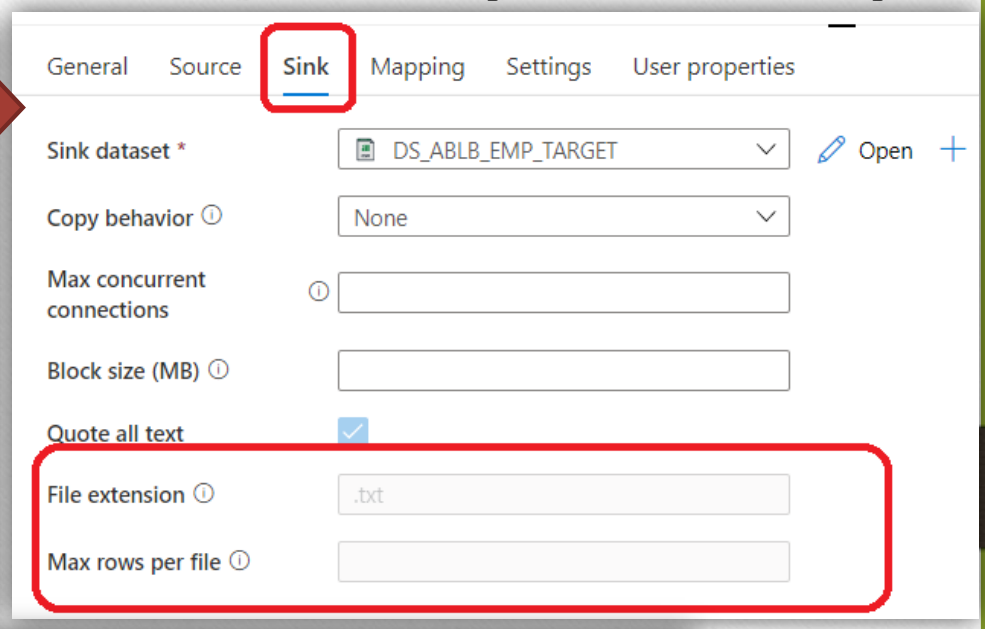


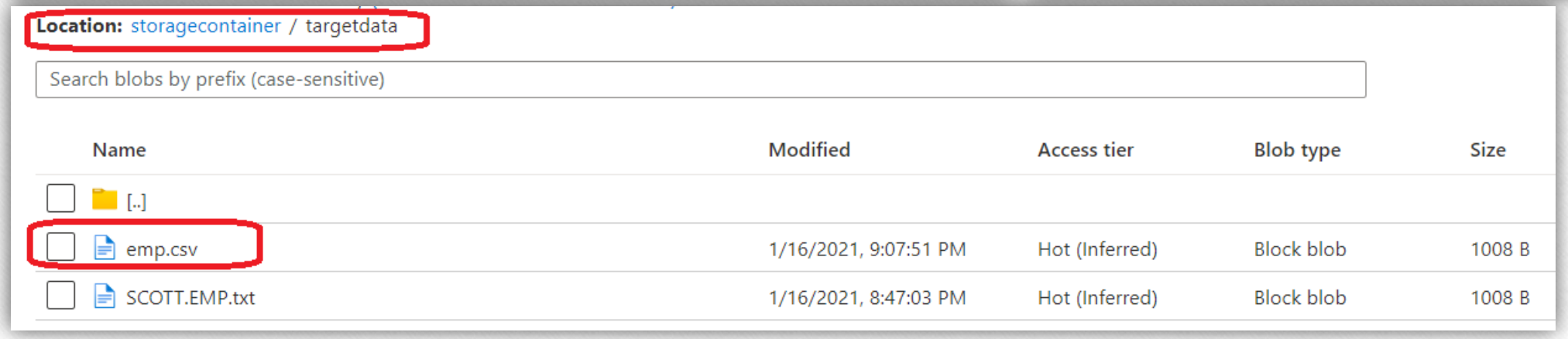
1.Select the **Copy_ORCL_TO_BLOB** link, you'll see the status of the copy activity run on this page.
2.To view details about the copy operation, select the **Details** (eyeglasses image) link. For details about the properties



Target File Name with Extension: If we specify Target file at **DATASET** level, then by default file extension edit Option will be disabled at copy activity sink level. Changed the filename with extension at dataset level as **emp.csv.** And output file is created as emp.csv







Trigger the pipeline on a schedule

This procedure is optional in this tutorial. You can create a *scheduler trigger* to schedule the pipeline to run periodically (hourly, daily, and so on). In this procedure, you create a trigger to run every minute until the end date and time that you specify.

1.Switch to the **Author** tab.

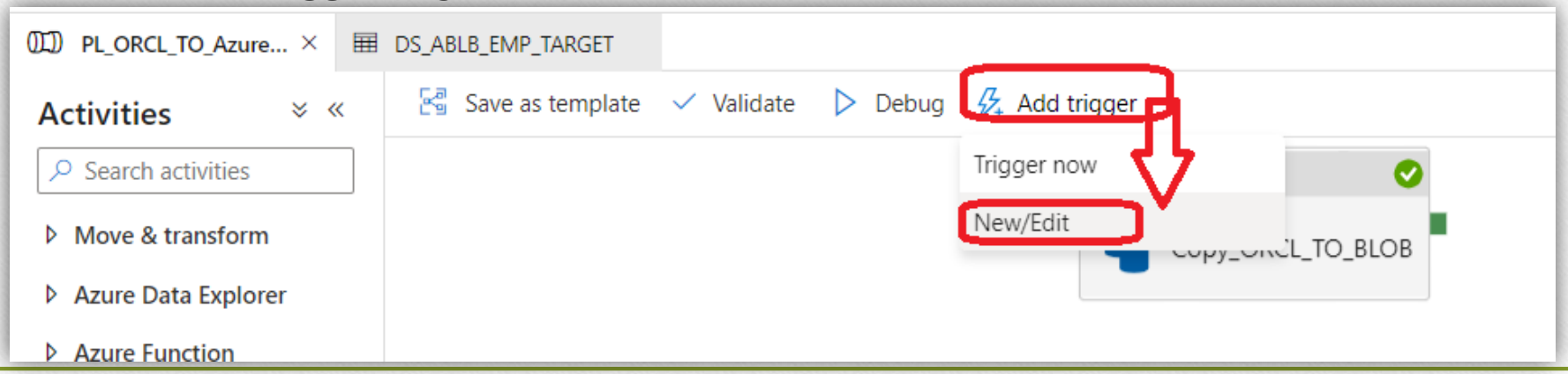
2.Go to your pipeline, select **Add Trigger** on the pipeline toolbar, and then select **New/Edit**.

3.On the **Add Triggers** page, select **Choose trigger**, and then select **New**.

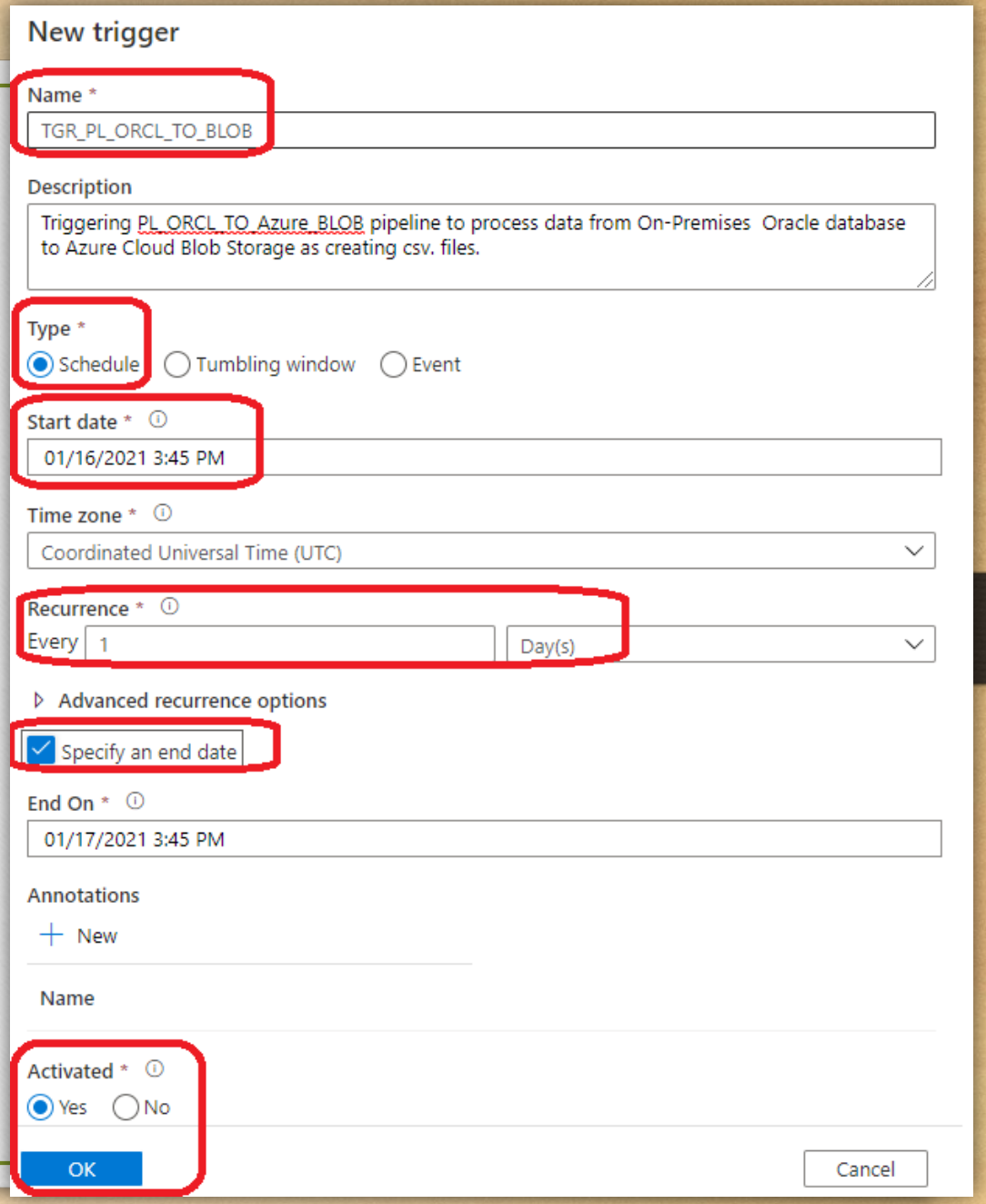
4.On the **New Trigger** page, under **End**, select **On Date**, specify an end time a few minutes after the current time, and then select **OK**.

5.A cost is associated with each pipeline run, so specify the end time only minutes apart from the start time. Ensure that it's the same day. However, ensure that there's enough time for the pipeline to run between the publish time and the end time. The trigger comes into effect only after you publish the solution to Data Factory, not when you save the trigger in the UI.

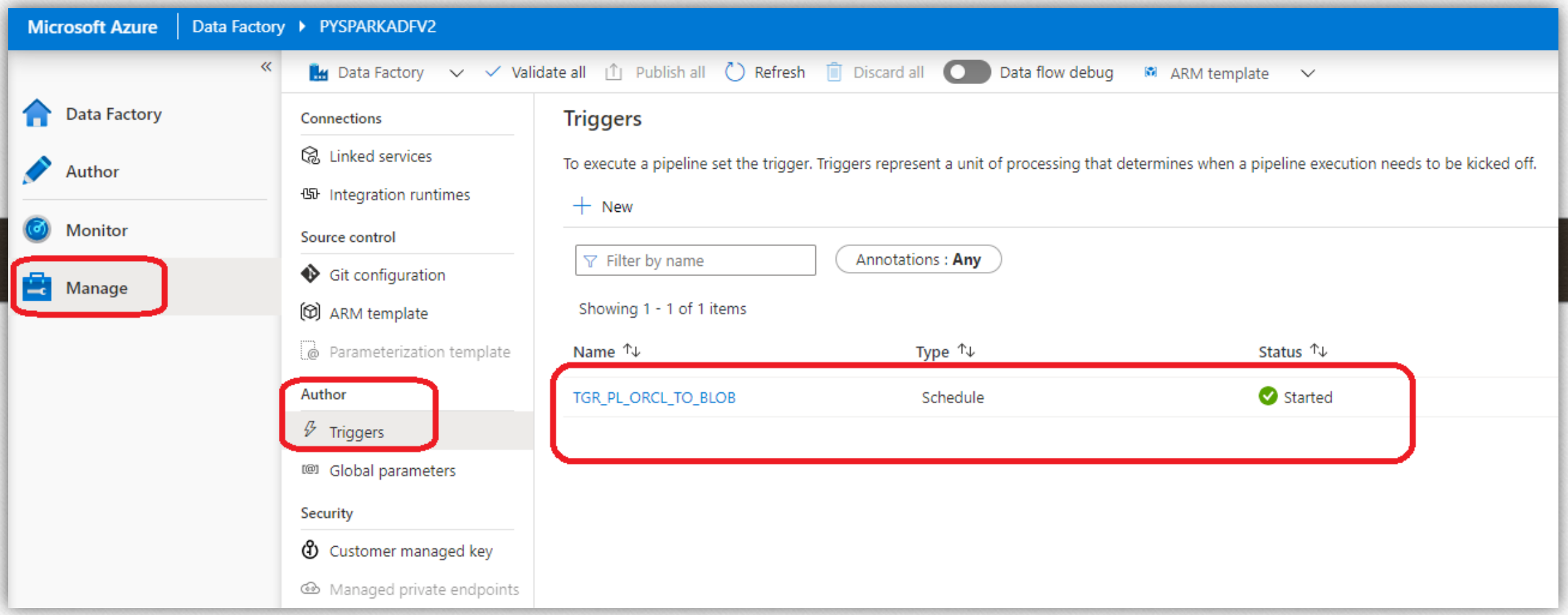
6.On the **New Trigger** page, select the **Activated** check box, and then select **OK**.



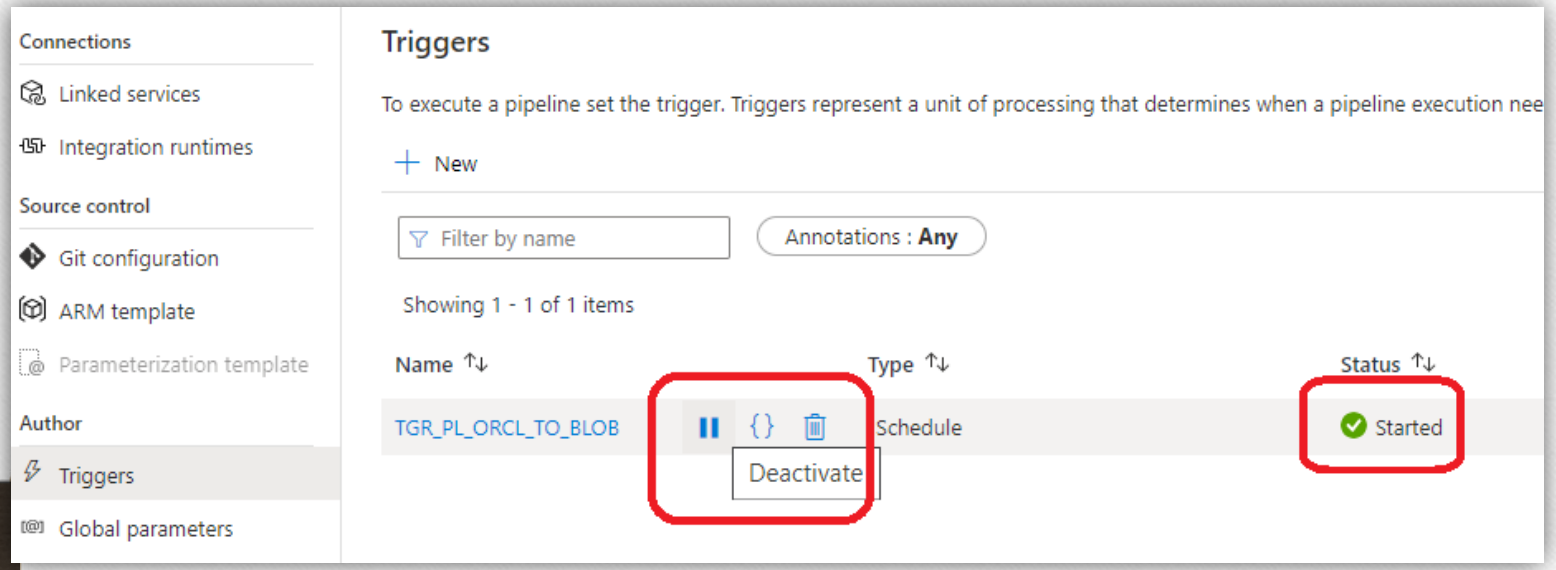
1. Review the warning message, and select **OK**.
2. Select **Publish all** to publish changes to Data Factory.
3. Switch to the **Monitor** tab on the left. Select **Refresh** to refresh the list. You see that the pipeline runs once every minute from the publish time to the end time.
4. Notice the values in the **TRIGGERED BY** column. The manual trigger run was from the step (**Trigger Now**) that you did earlier.
5. Switch to the **Trigger runs** view.
6. Confirm that an output file is created for every pipeline run until the specified end date and time in the **output** folder.



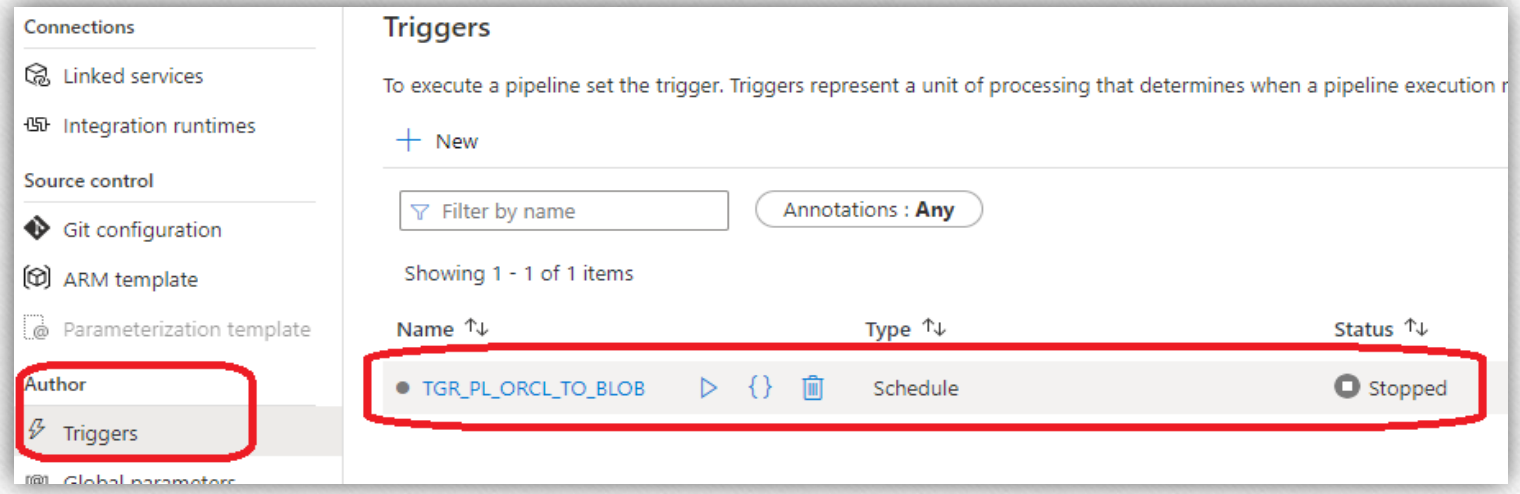
Verify All Created Triggers in Manage tab => Triggers window.
Here we can find all created triggers both active and inactive triggers.



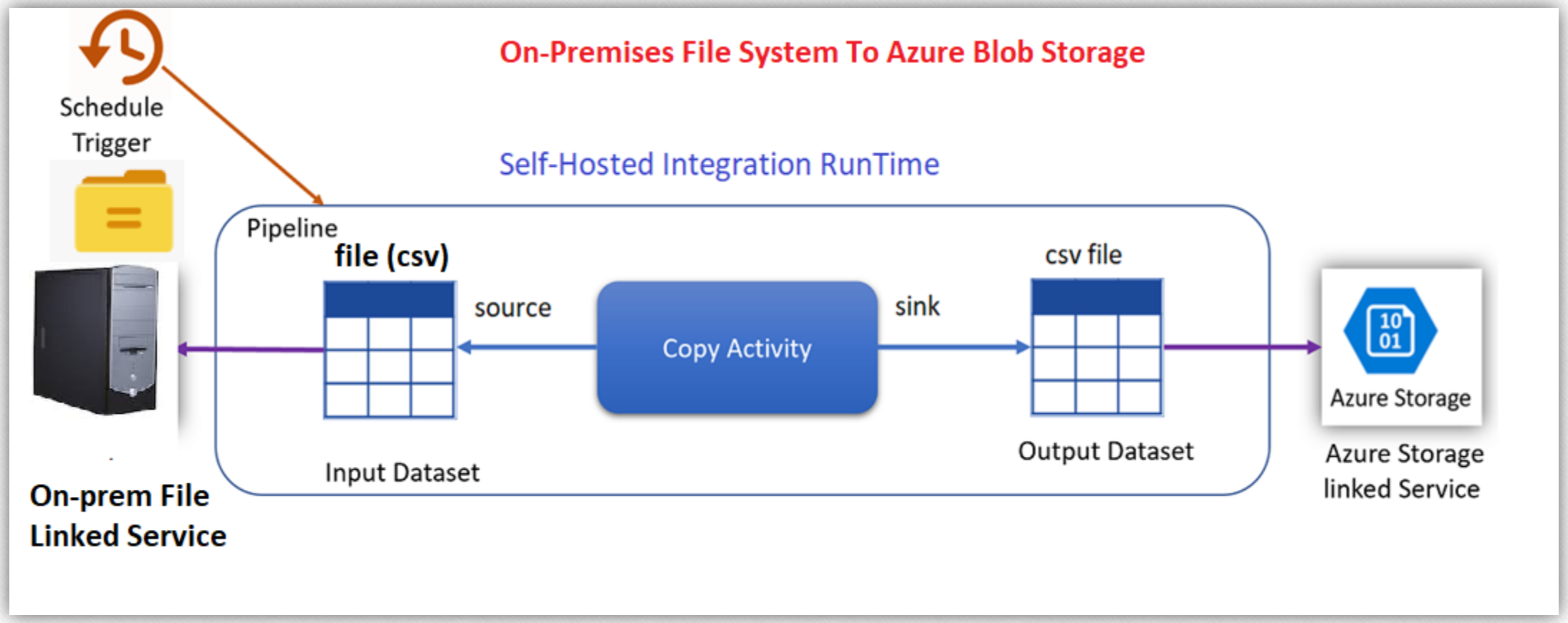
Deactivate triggers using **Deactivate** option and trigger will be **disabled**.



Trigger is deactivated and we can see the status as **stopped**



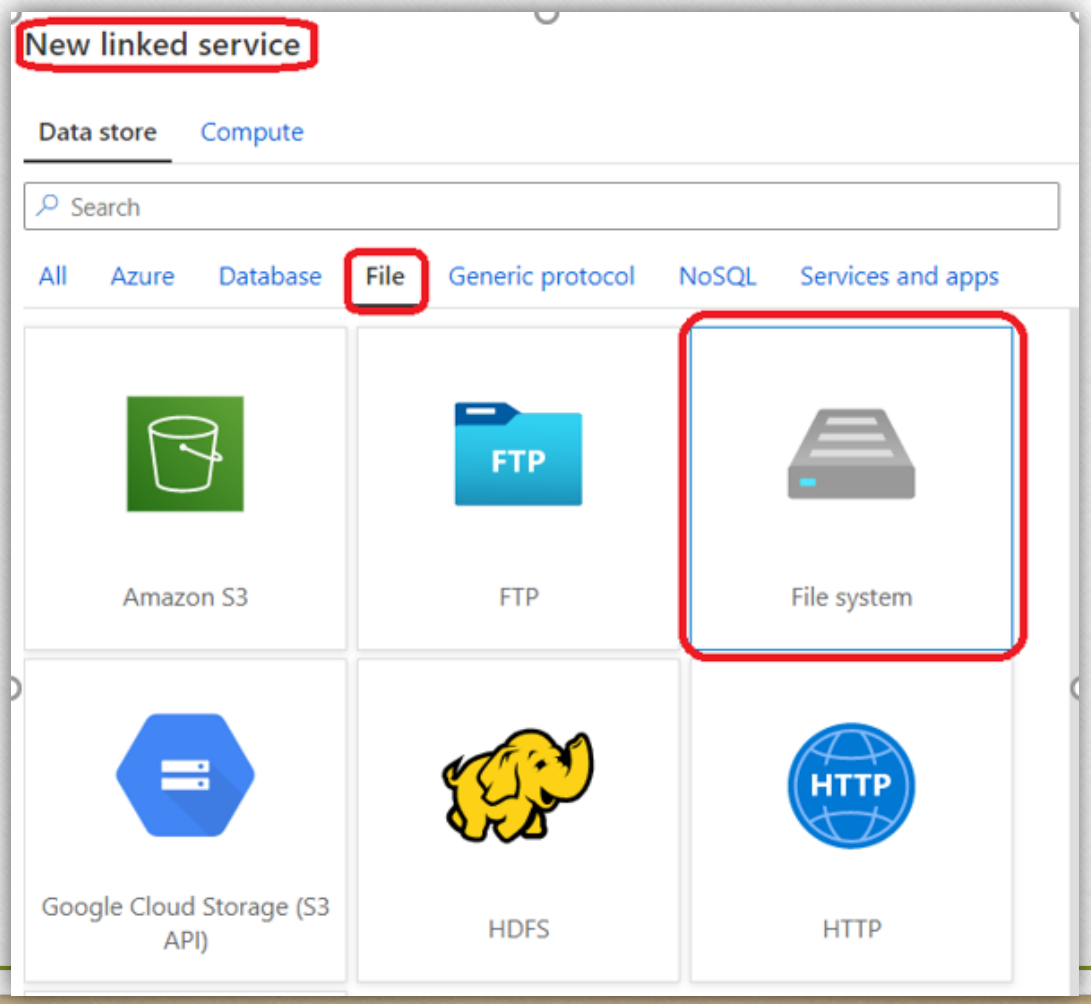
Loading Data from On-Premises File To Azure Blob Storage



1) Creating Self-Hosted Integration Runtime
2) Creating Linked Service for File And Target Blob Storage.
3) Creating Datasets for Source File And Target Blog File store.

Creating linked service for Source On-Premises File System.

1) Click on New Linked Service => Goto File => Select File System = > Continue



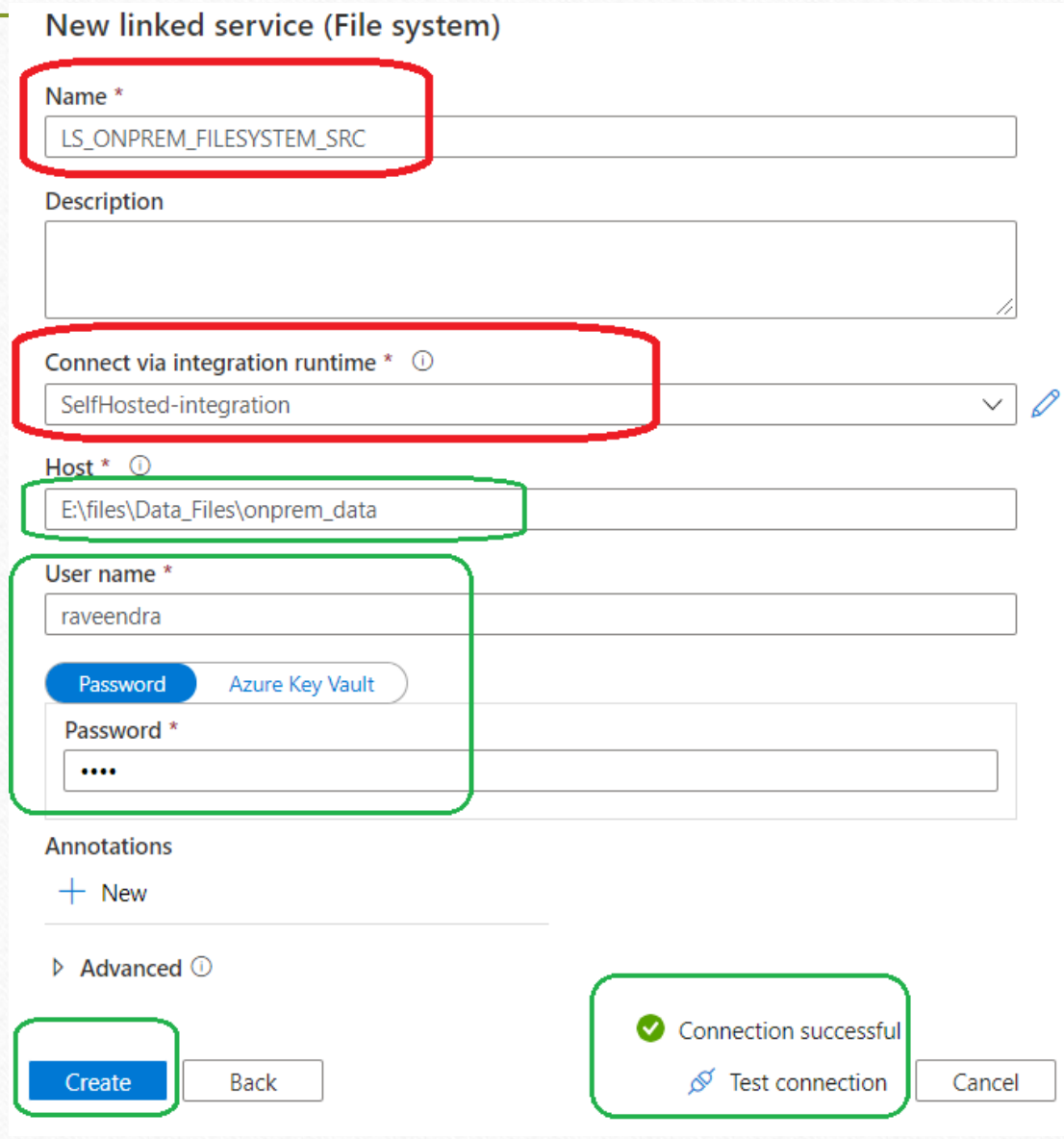
Creating linked service for Source On-Premises File System.

Name: LS_ONPREM_FILESYSTEM_SRC
Integration runtime : SelfHosted-Integration
Host: E:\files\data_files\onprem_data

Username: Raveendra
Password: mypassword

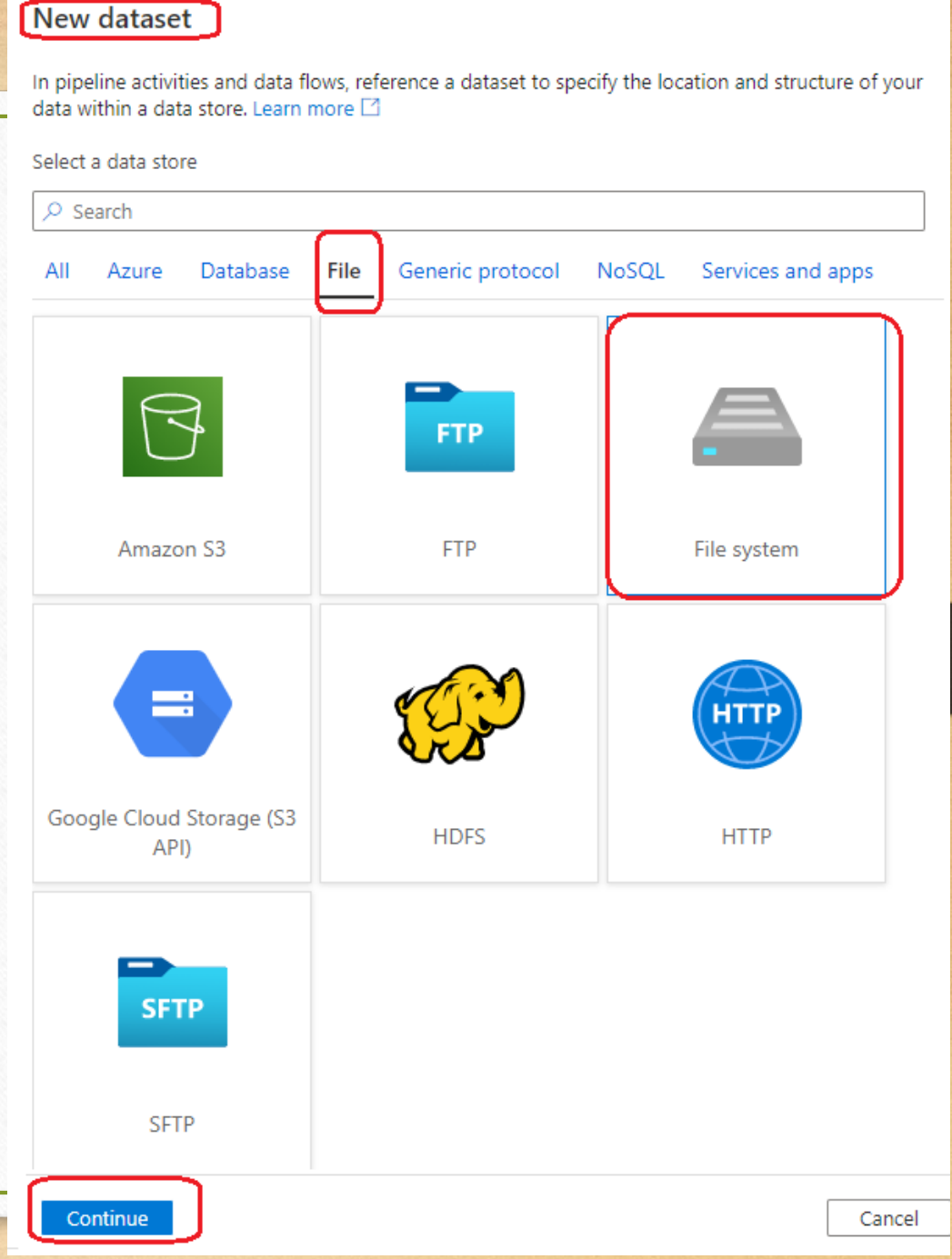
Click on **Test Connection**

Create Linked service to connecting on-premises File system to copy files from onprem to azure Blob storage.



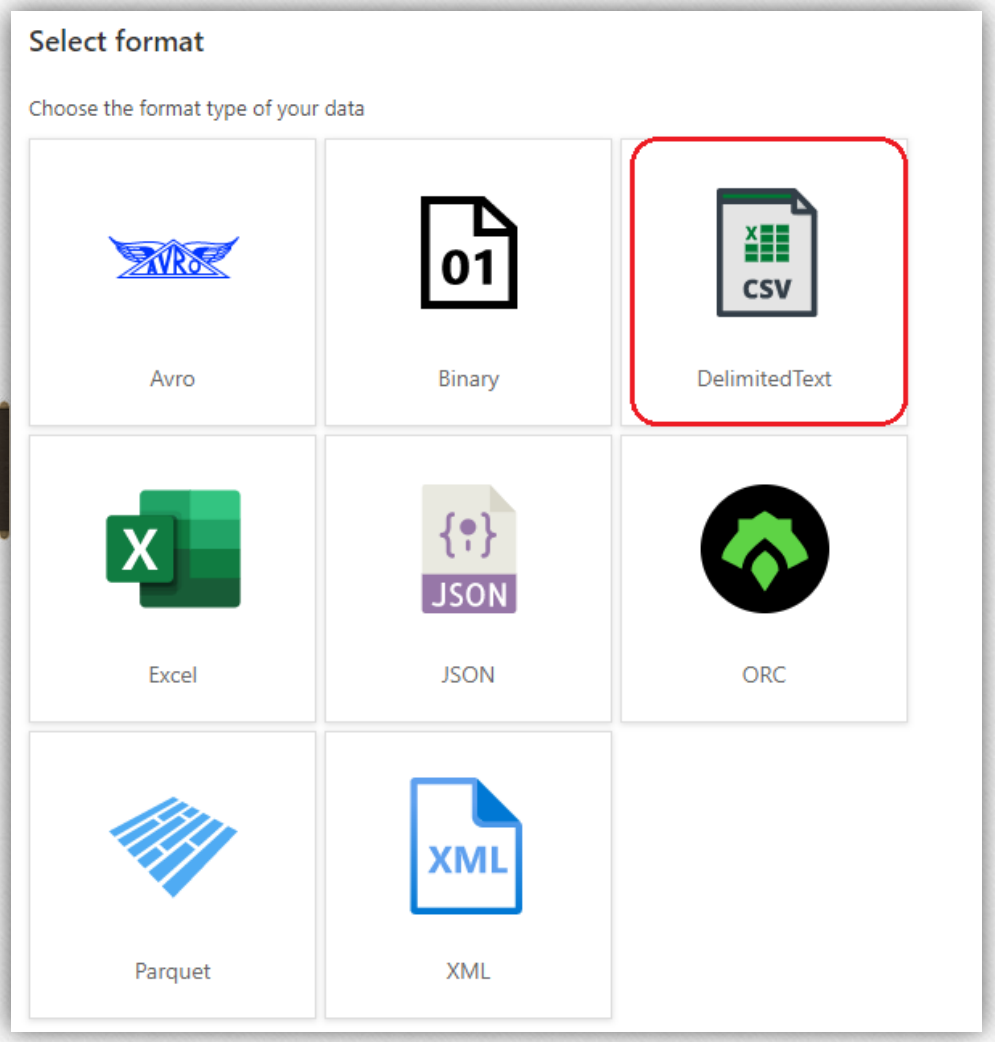
Creating DataSet for On-premises Filesystem

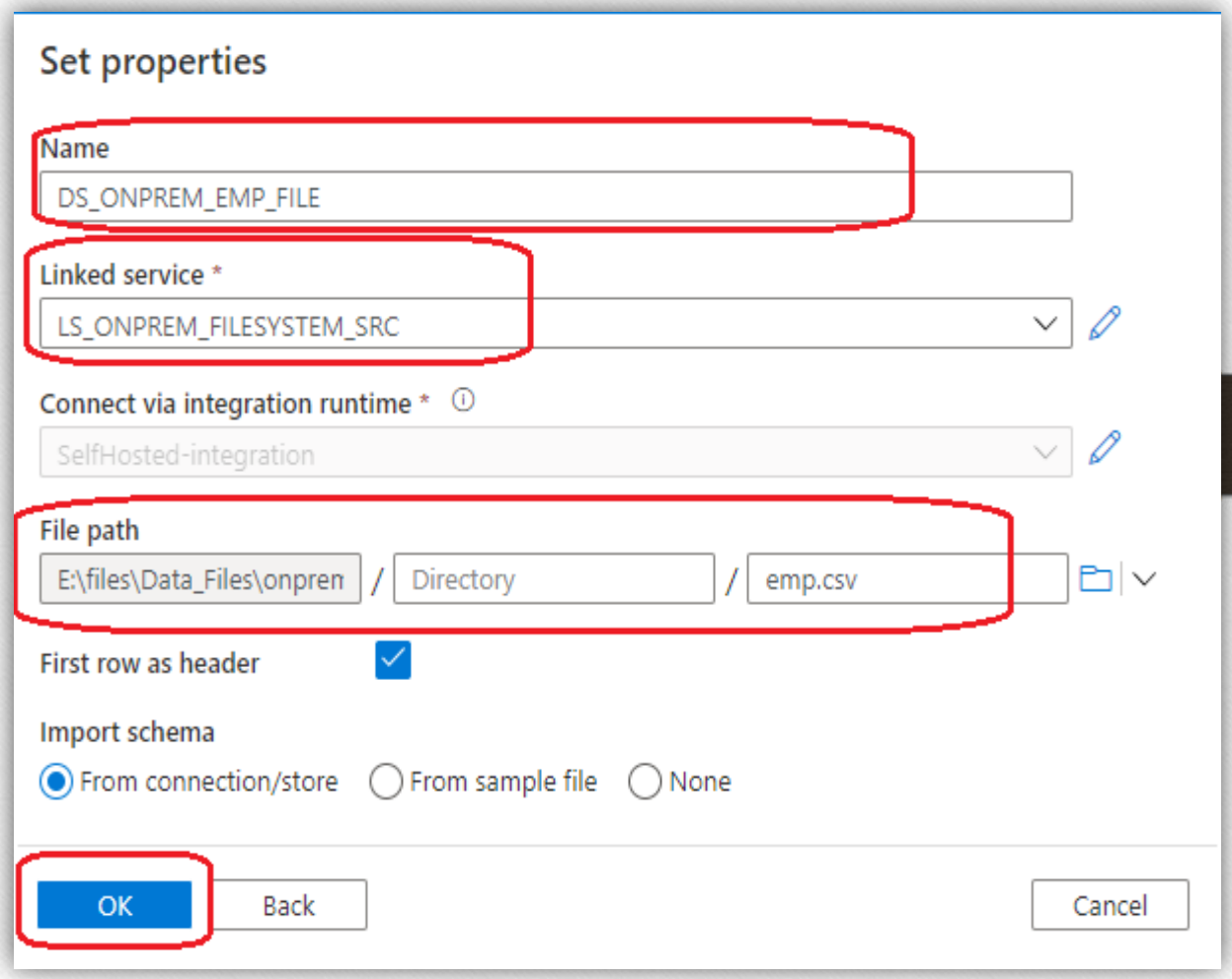
1) Click on New Dataset => Goto File => File System =>Select => Continue



Creating DataSet for On-premises Filesystem

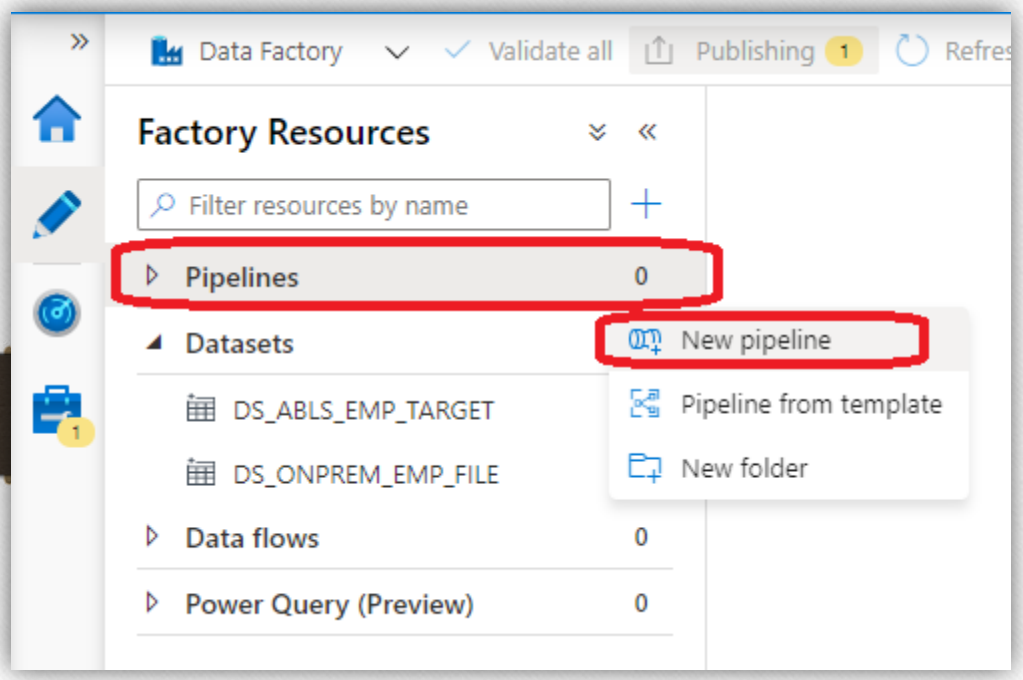
Select Dataset File Format as delimited Text (CSV) file And Enter the Dataset name and linked services and ok.

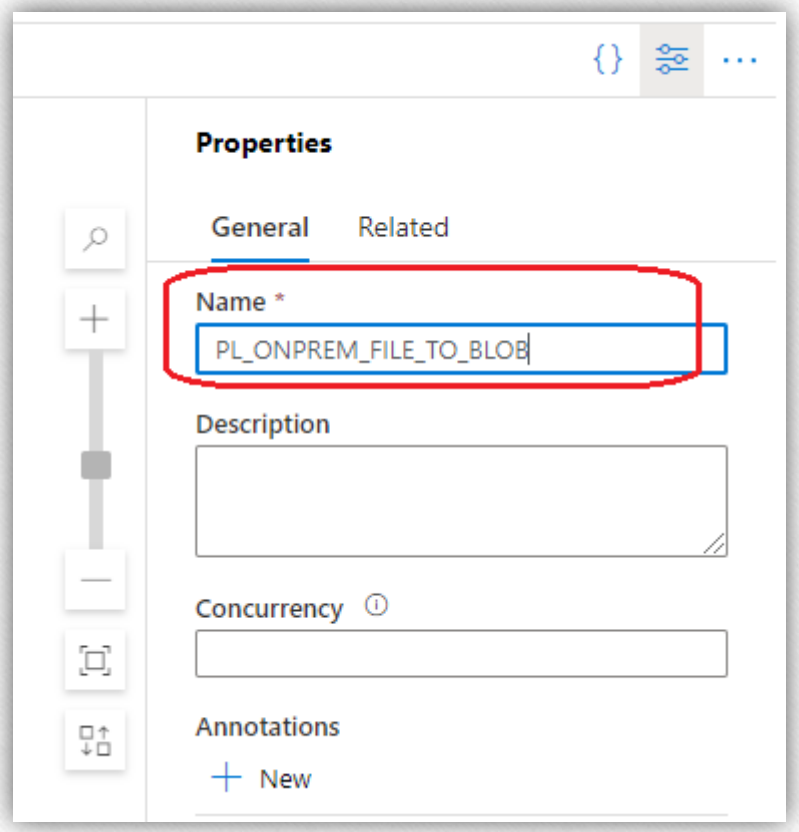




Creating DataSet for On-premises Filesystem

Click on New Pipeline for transferring data from On-premises File System to Azure Cloud Blob Storage.

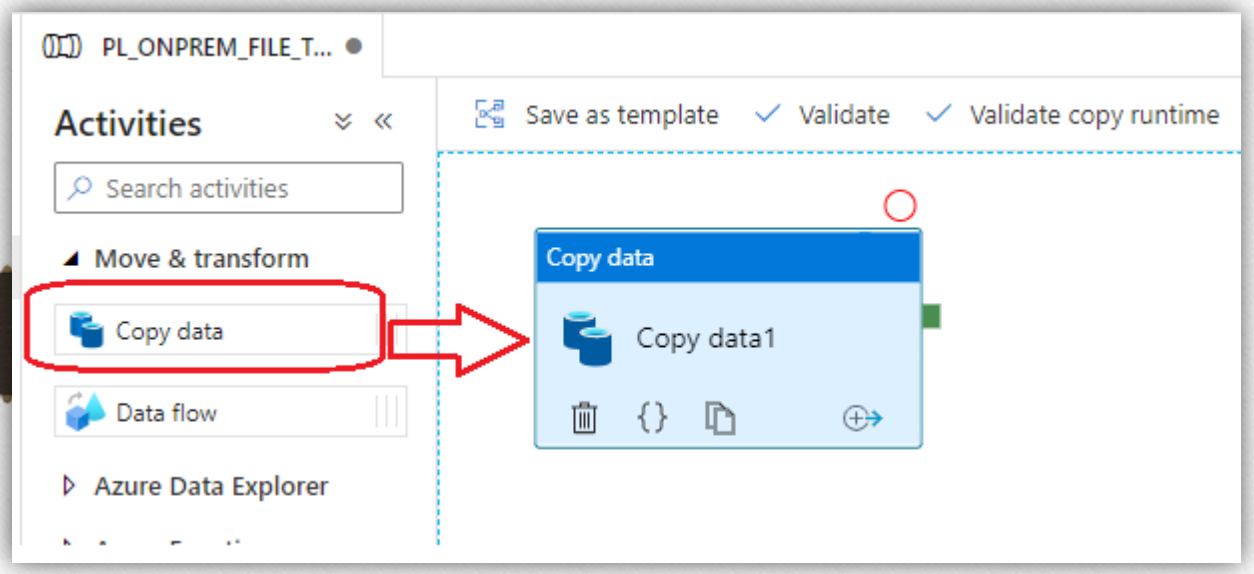


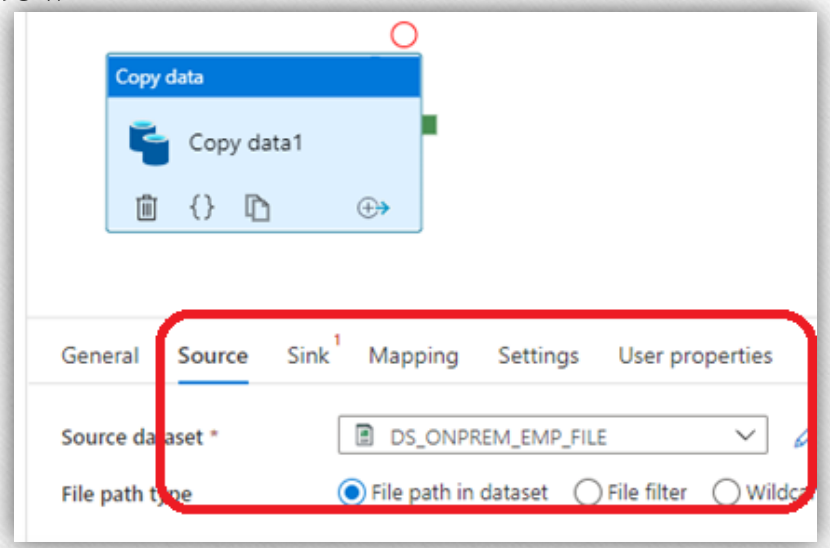


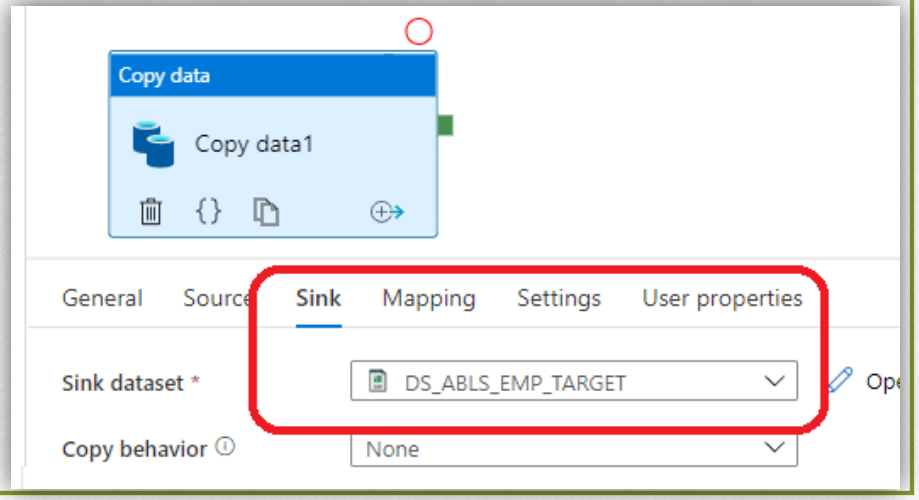
Creating DataSet for On-premises Filesystem

Drag and Drop Copy Activity from activities list to pipeline diagram window.

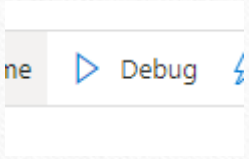
1) Select Source dataset as DS_ONPREM_EMP_FILE
2) Target Dataset as DS_ABLS_EMP_TARGET



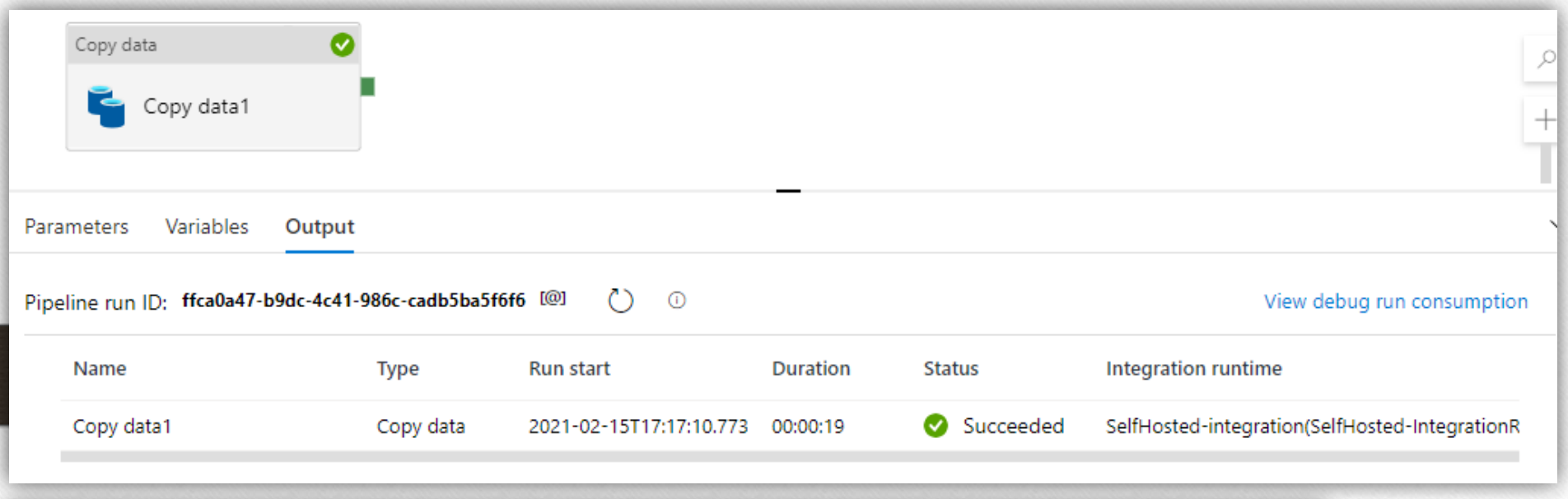


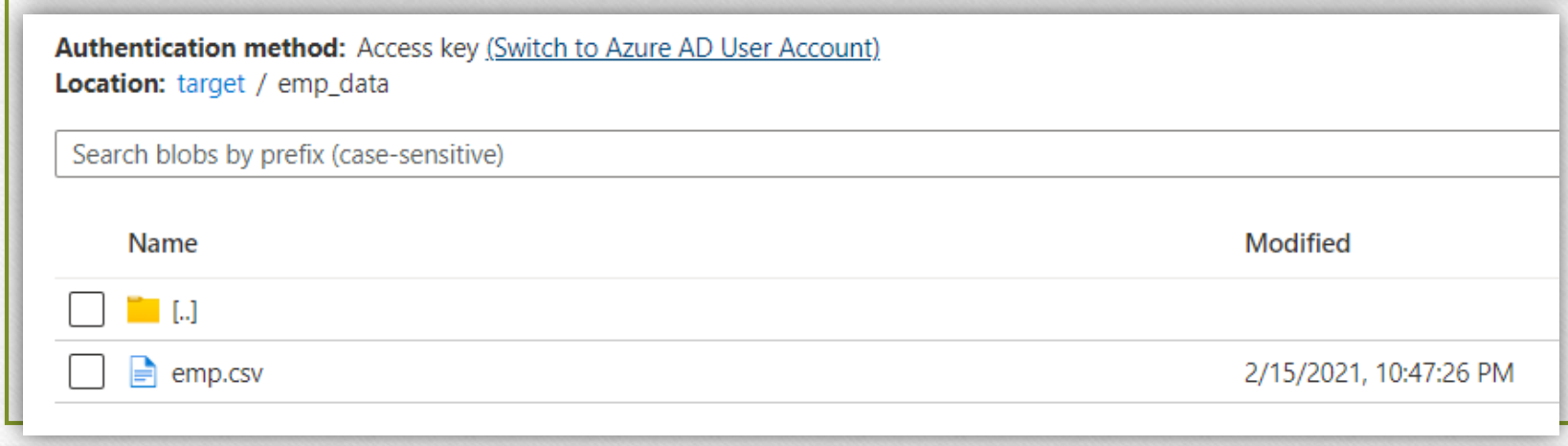


Click on Debug Option To verify transferring data from on-prem To azure blob



On-prem file is successfully copied to Azure blob storage.





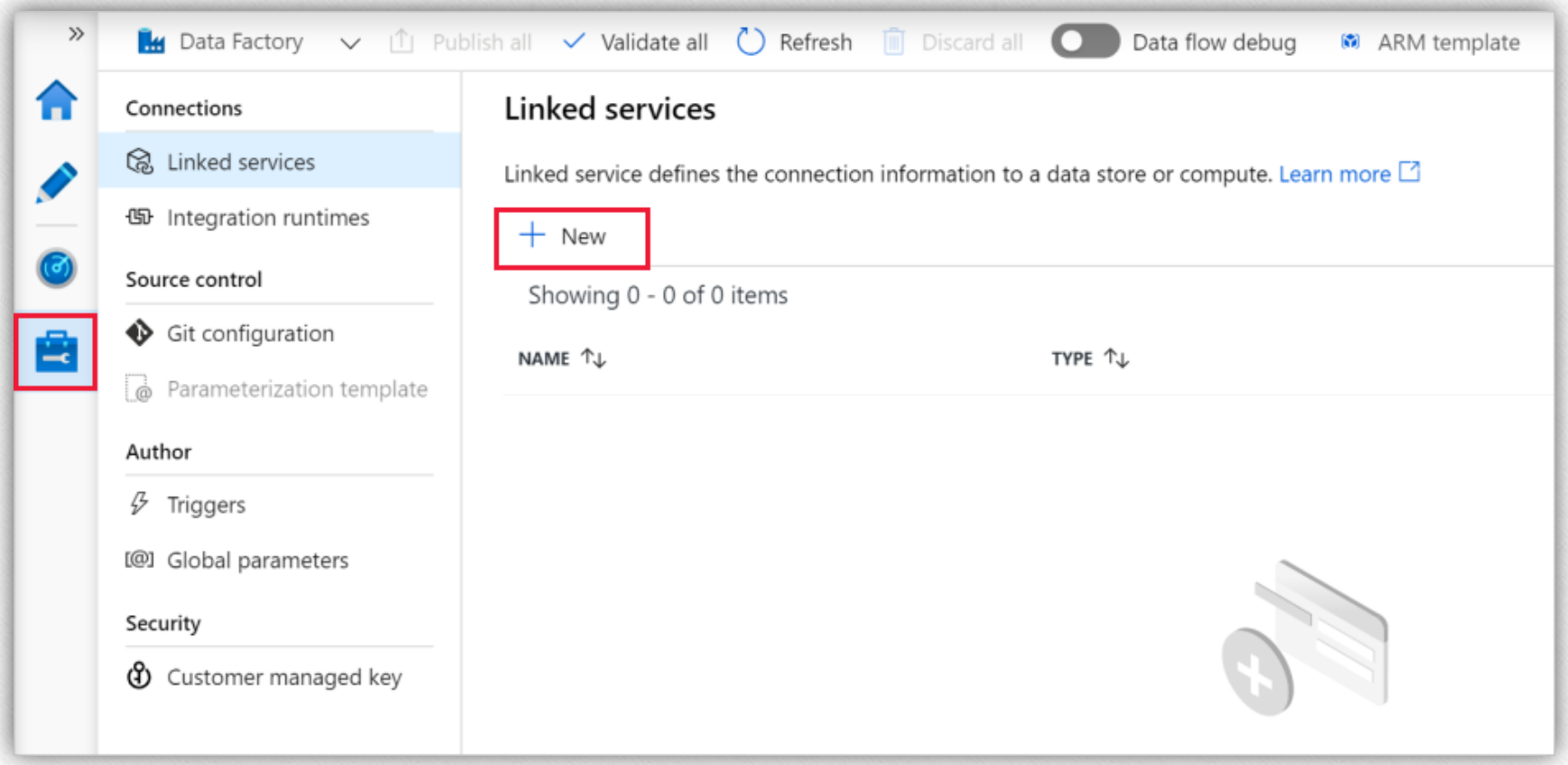
Creating Liked Services for Source And Target

Create a linked service For Data Lake as Source Linked services

In this procedure, you create a linked service to link your Azure Storage account to the data factory. The linked service has the connection information that the Data Factory service uses at runtime to connect to it.

1.On the Azure Data Factory UI page, open **Manage** tab from the left pane.

2.On the Linked services page, select **+New** to create a new linked service.



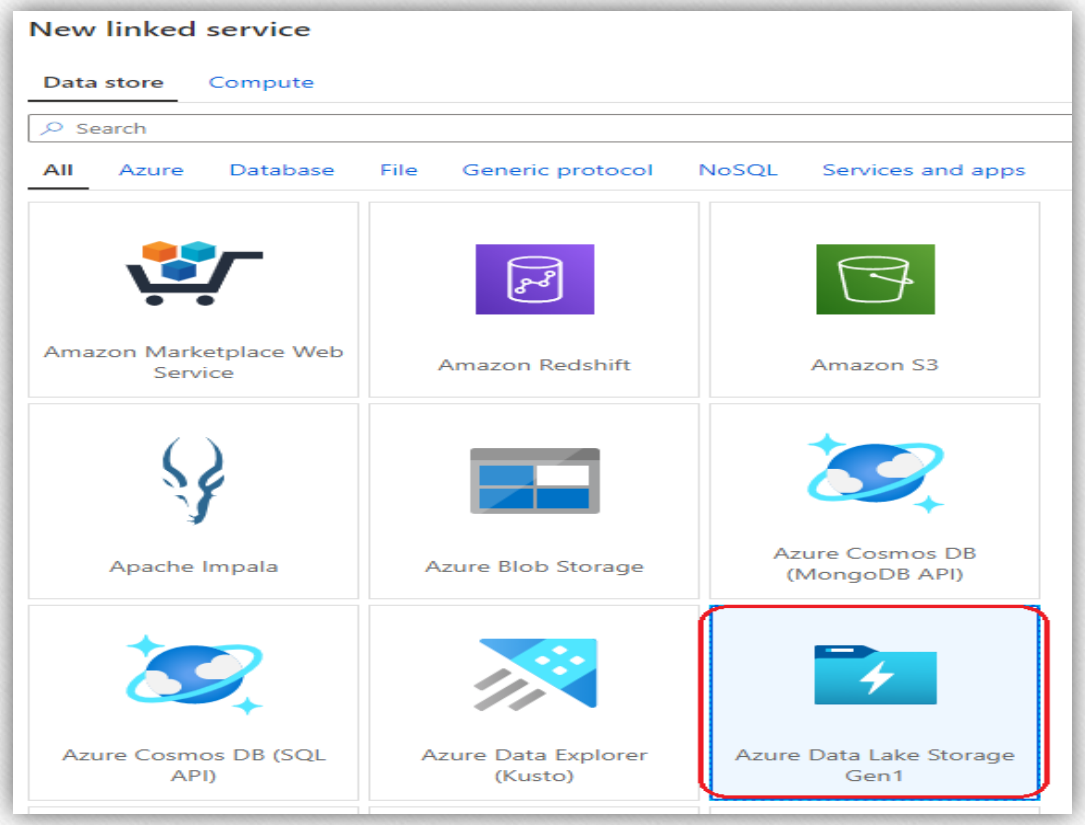
Creating Liked Services for Source And Target

Create a linked service For Data Lake as Source Linked services

In this procedure, you create a linked service to link your Azure Storage account to the data factory. The linked service has the connection information that the Data Factory service uses at runtime to connect to it.

1.On the Azure Data Factory UI page, open Manage tab from the left pane.

2.On the Linked services page, select **+New** to create a new linked service.



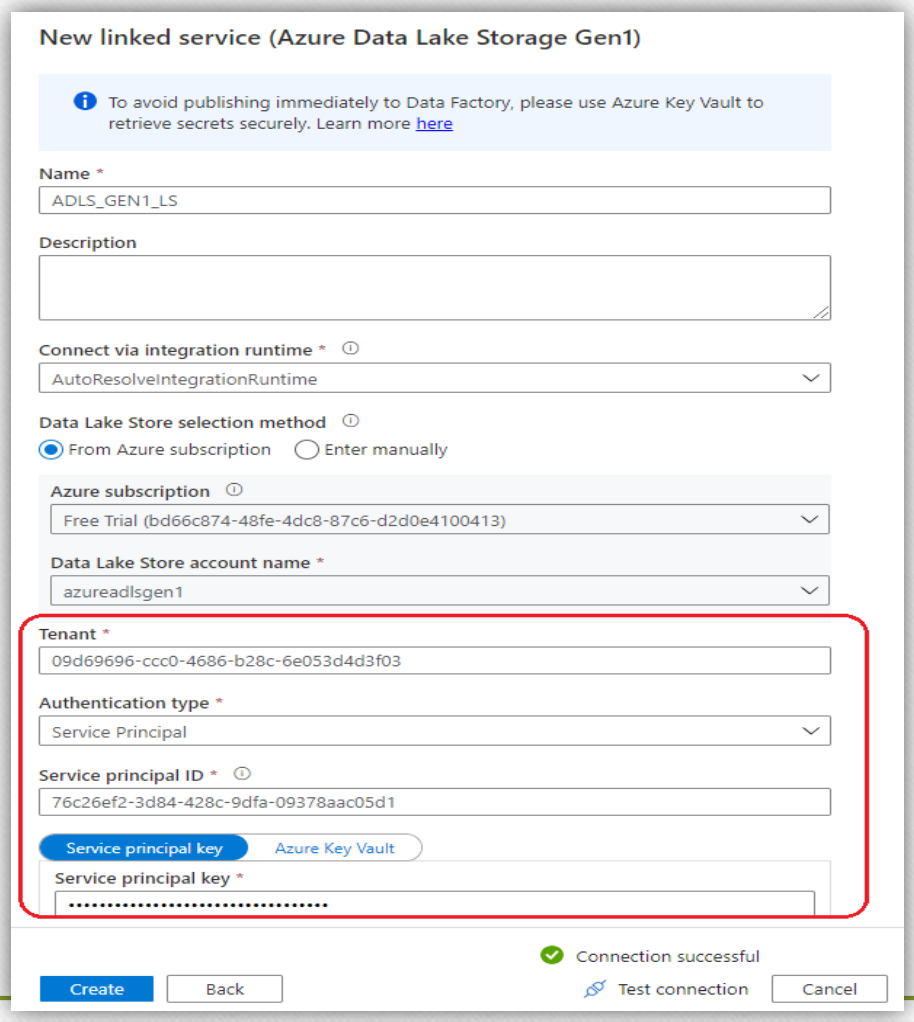
Creating Liked Services for Source And Target

In the **New Linked Service (Azure Data Lake Storage Gen1)** page, do the following steps:

1.Select your Data Lake Storage Gen1 account for the **Data Lake Store account name**.

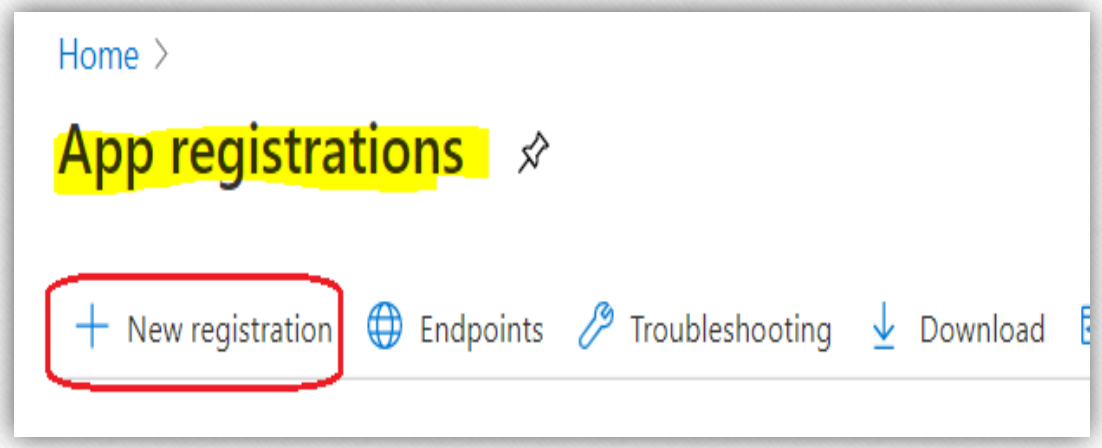
2.Specify the **Tenant**, and select Finish.

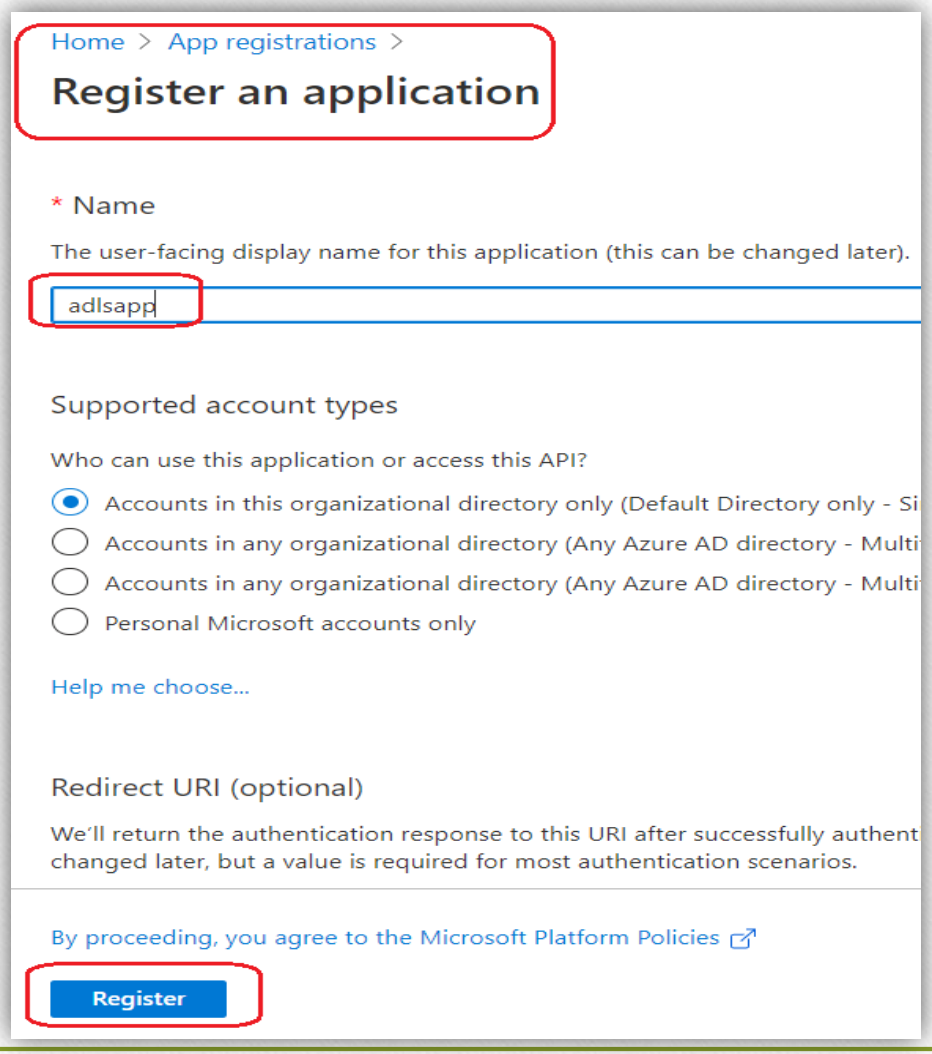
3.Select **Next**.



Create an Active Directory web application

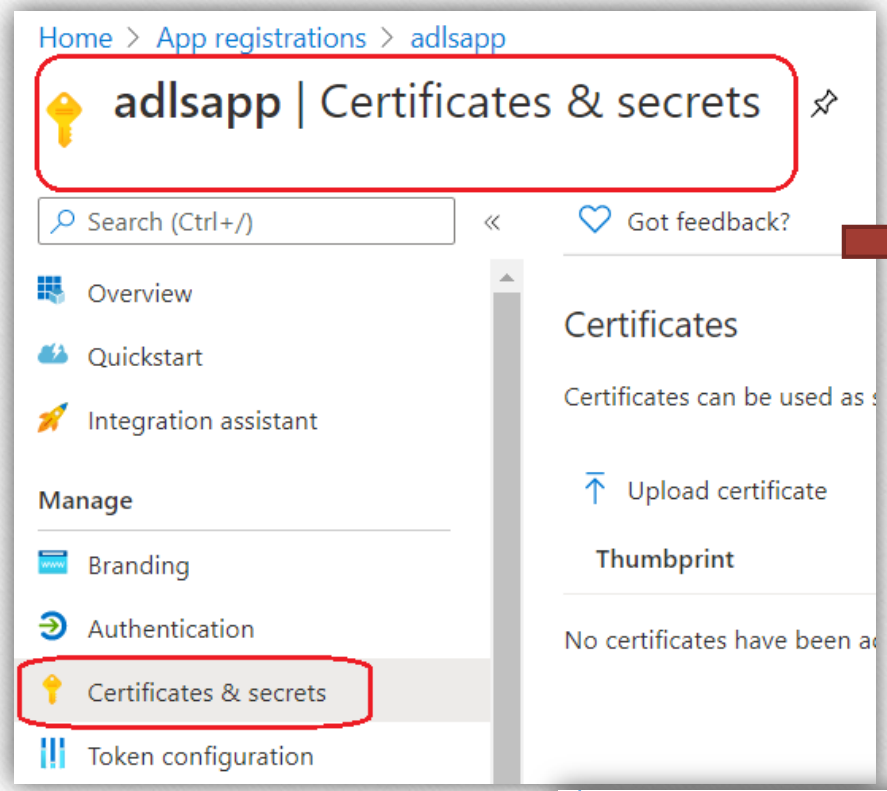
Create and configure an Azure AD web application for service-to-service authentication with Azure Data Lake Storage Gen1 using Azure Active Directory.

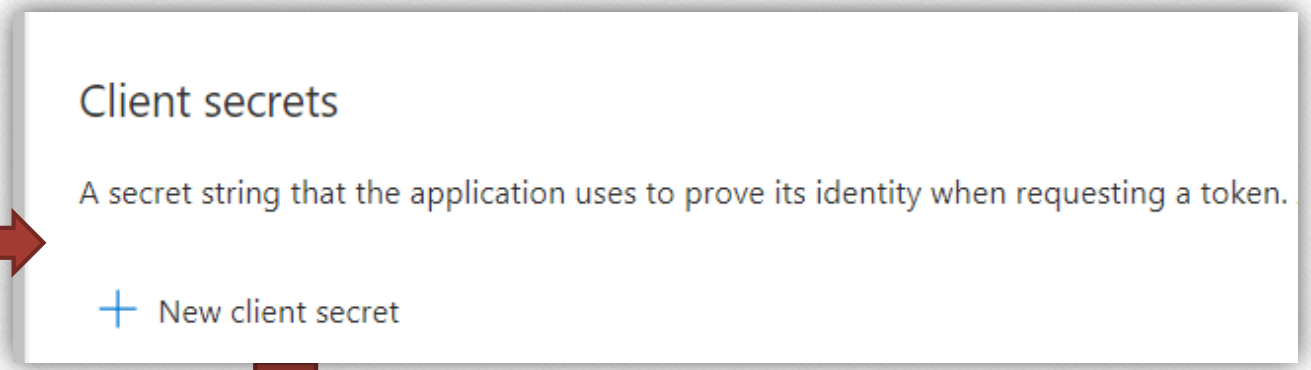


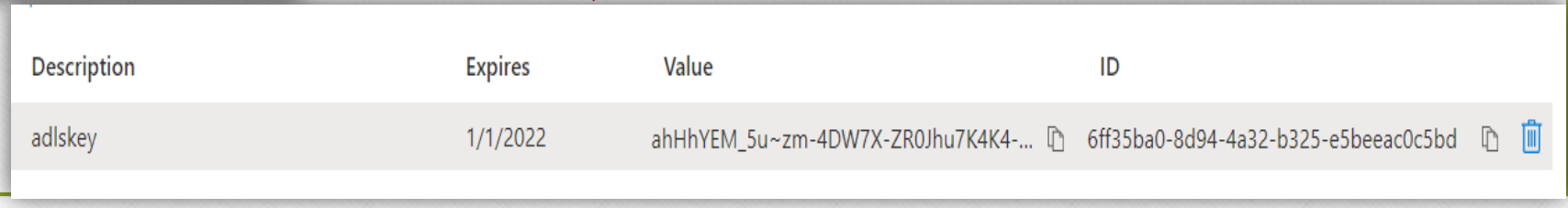


Create an Active Directory web application

Create and configure an Azure AD web application for service-to-service authentication with Azure Data Lake Storage Gen1 using Azure Active Directory.





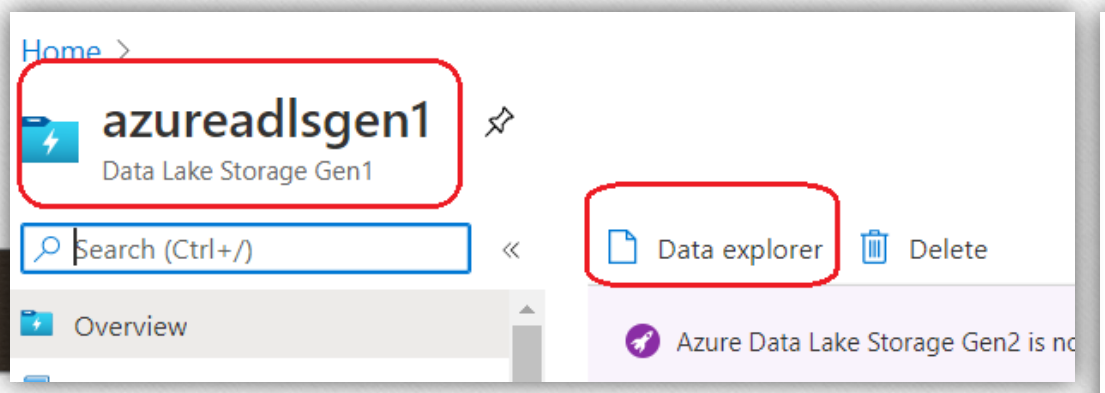


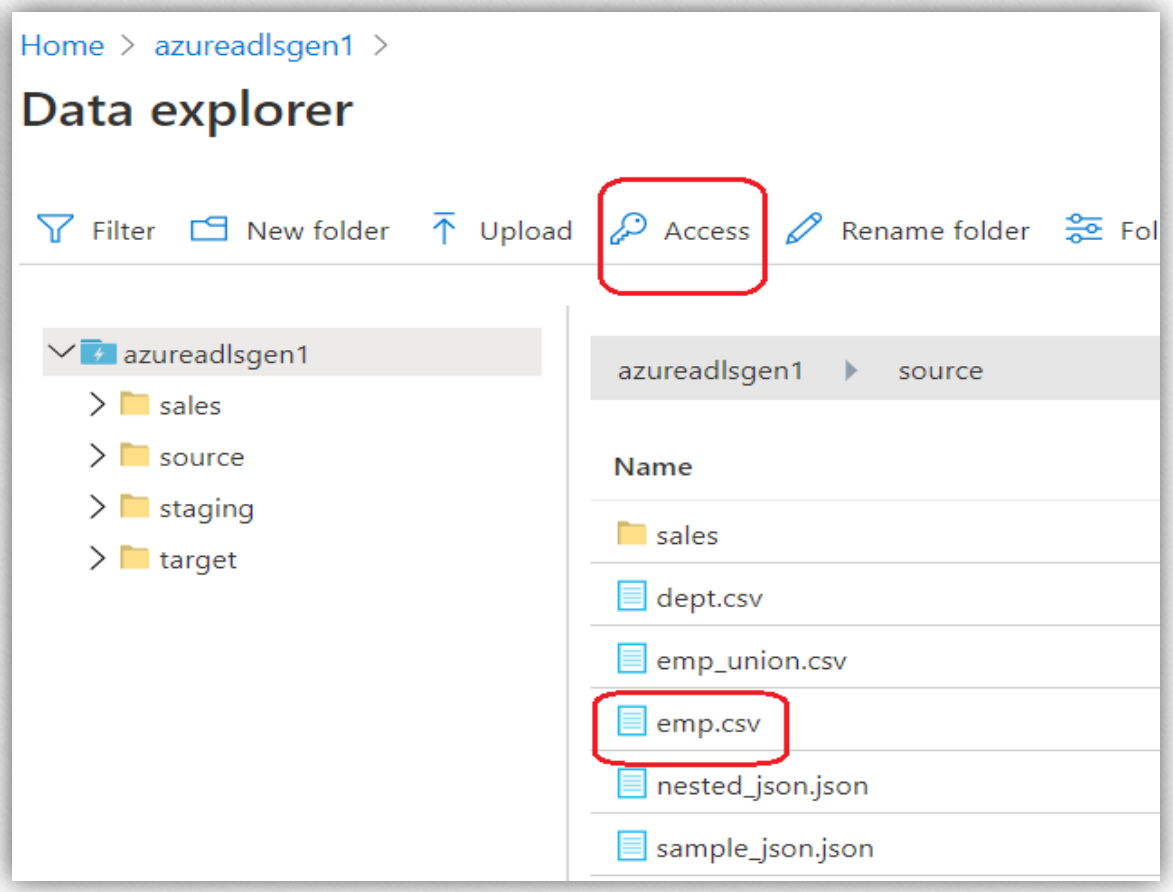
Create an Active Directory web application

Assign the Azure AD application to the Azure Data Lake Storage Gen1 account file or folder

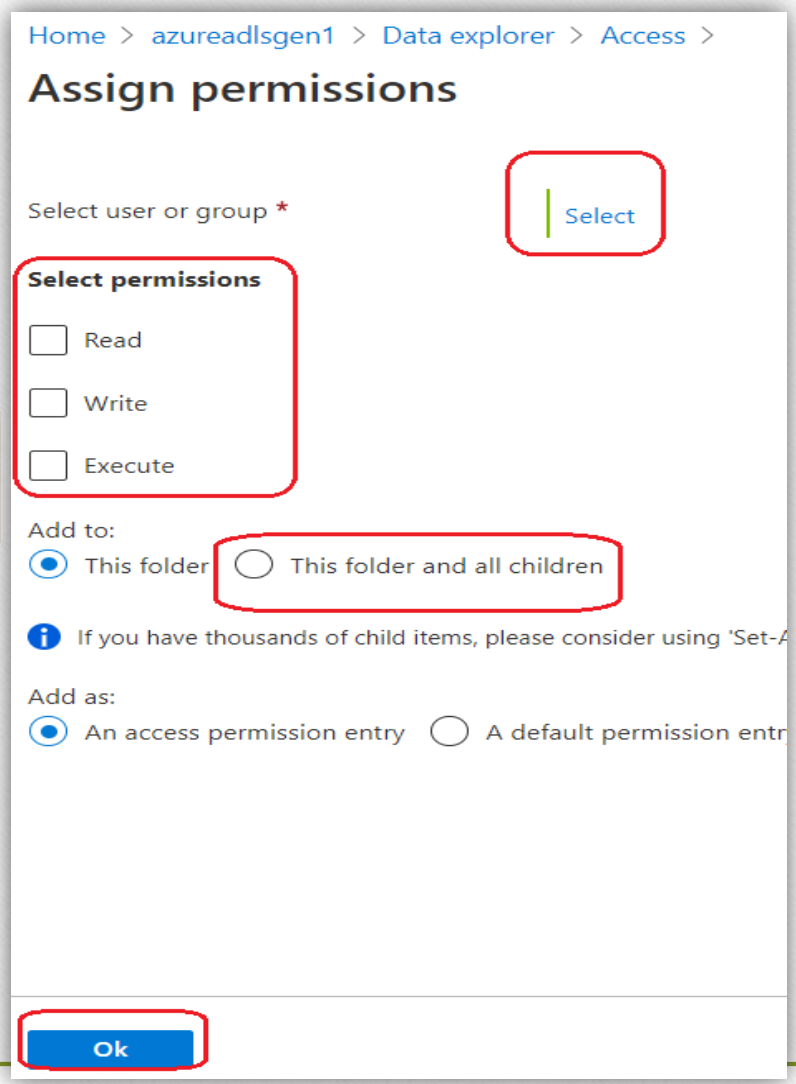
1.Sign on to the Azure portal. Open the Data Lake Storage Gen1 account that you want to associate with the Azure Active Directory application you created earlier.

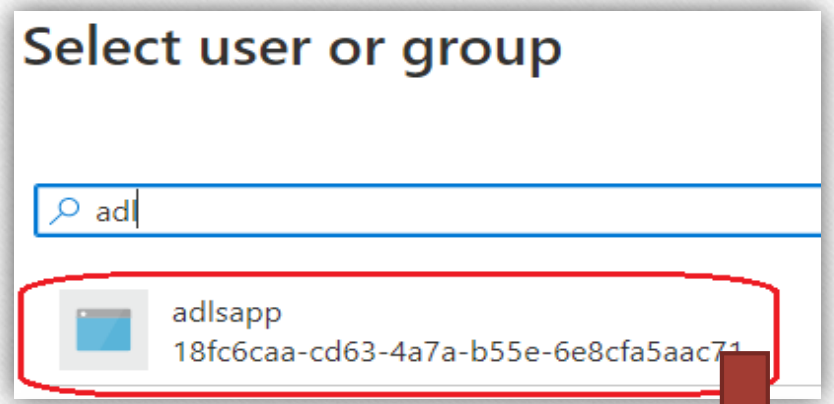
2.In your Data Lake Storage Gen1 account blade, click **Data Explorer**.

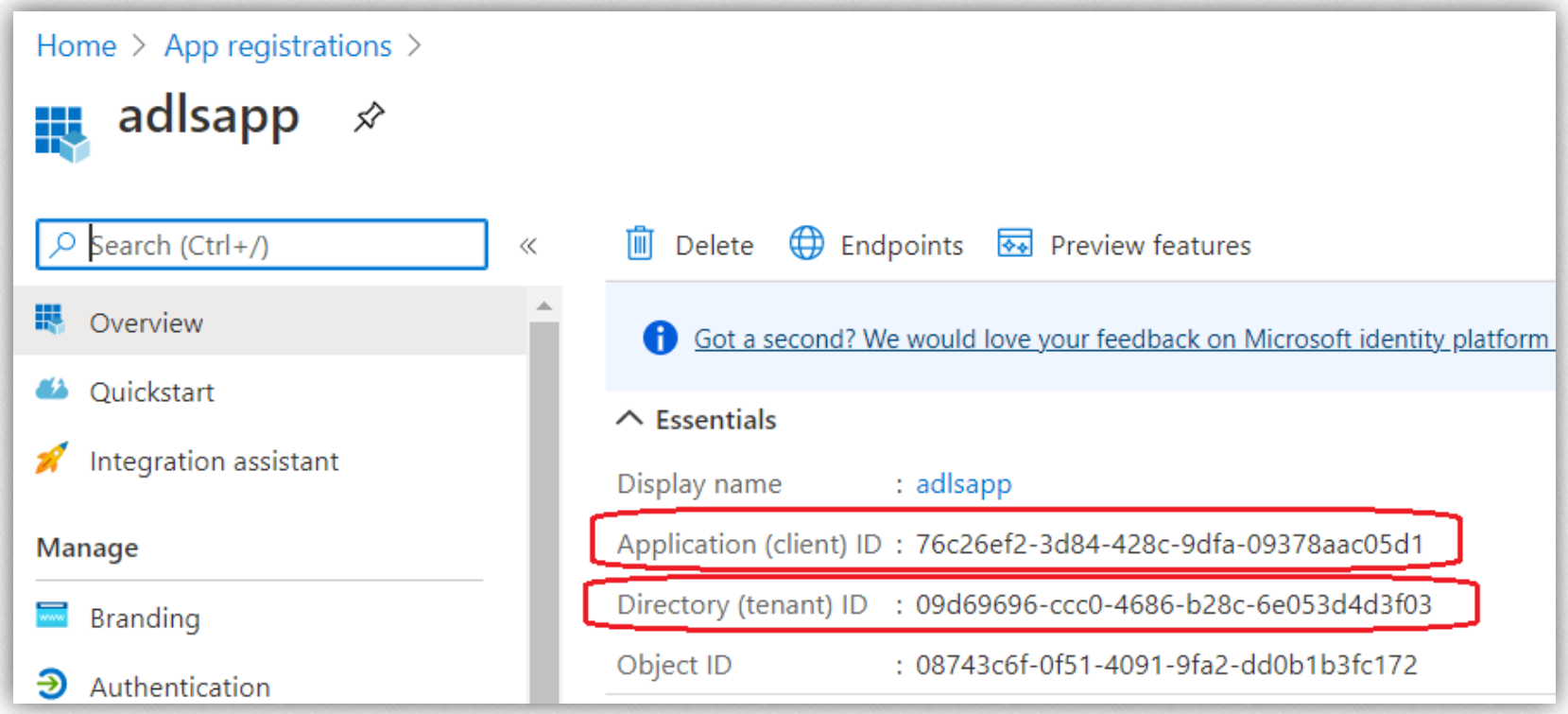




The **Access** blade lists the standard access and custom access already assigned to the root. Click the **Add** icon to add custom-level ACLs.







app key

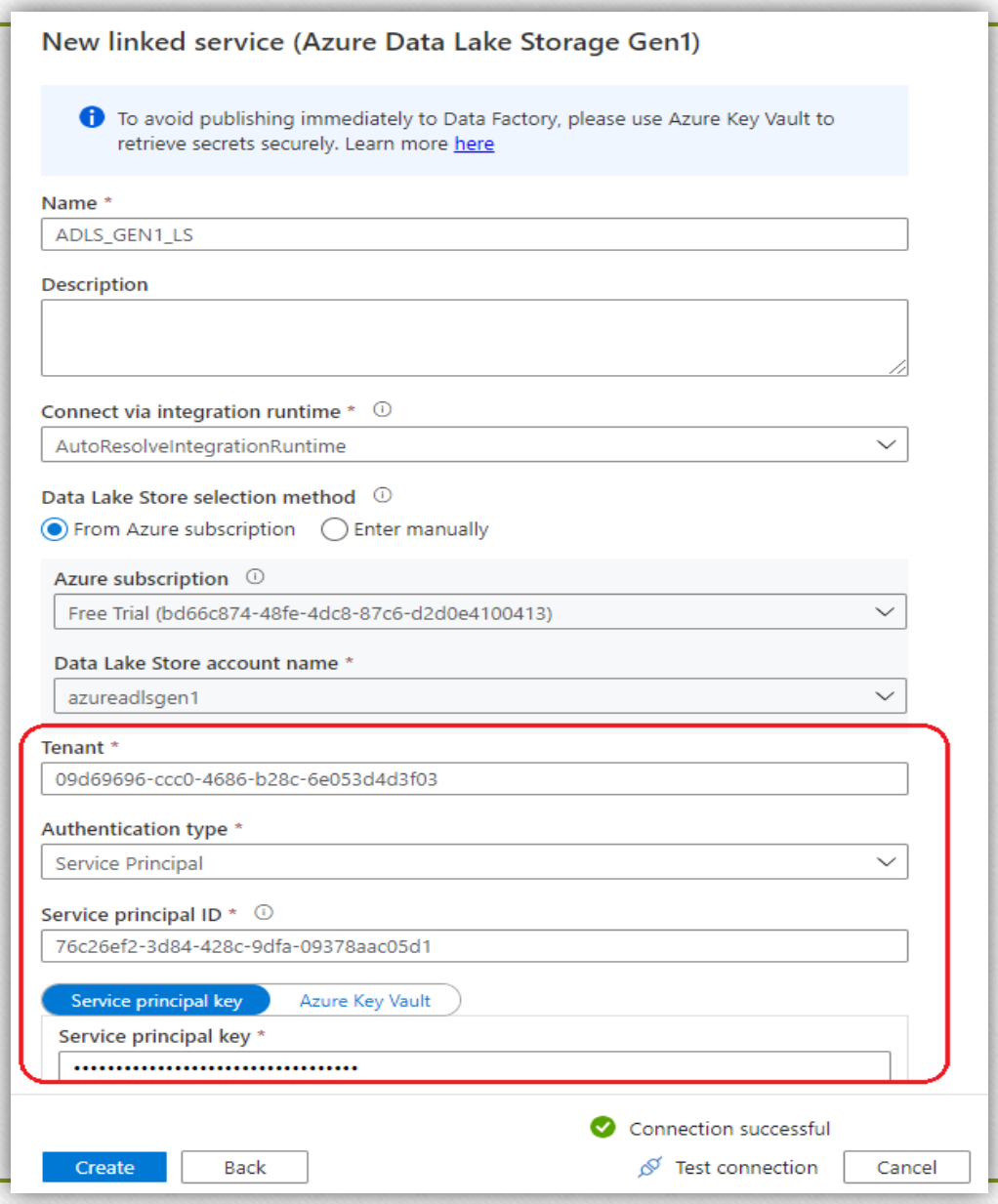
ahHhYEM_5u~zm-4DW7X-ZR0Jhu7K4K4-AL

Service Principle id(application id)

76c26ef2-3d84-428c-9dfa-09378aac05d1

tenant id

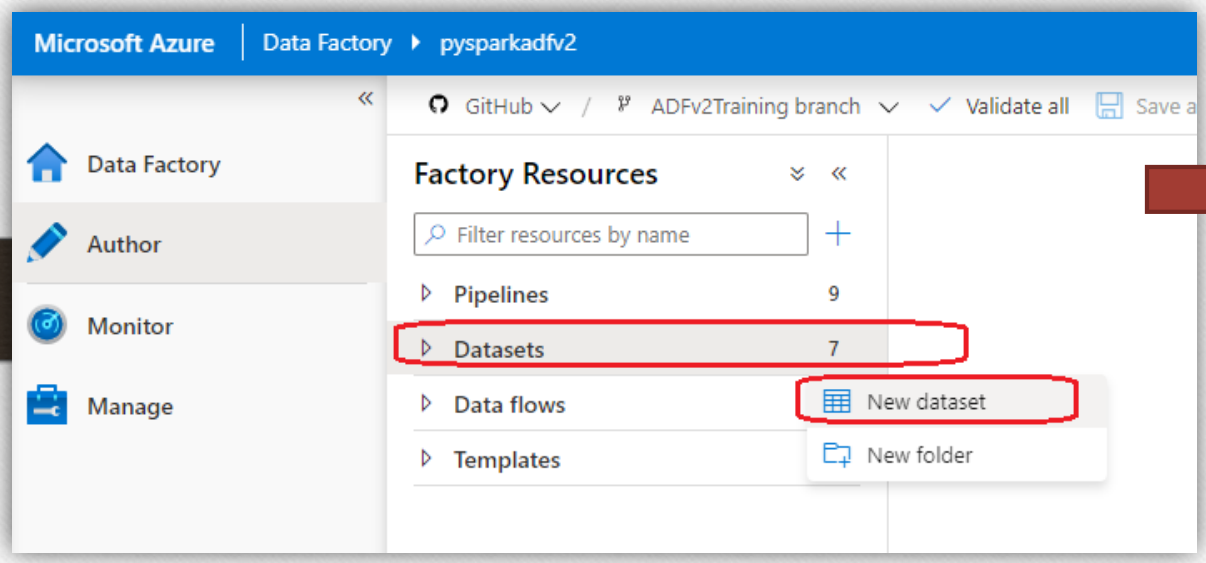
09d69696-ccc0-4686-b28c-6e053d4d3f03

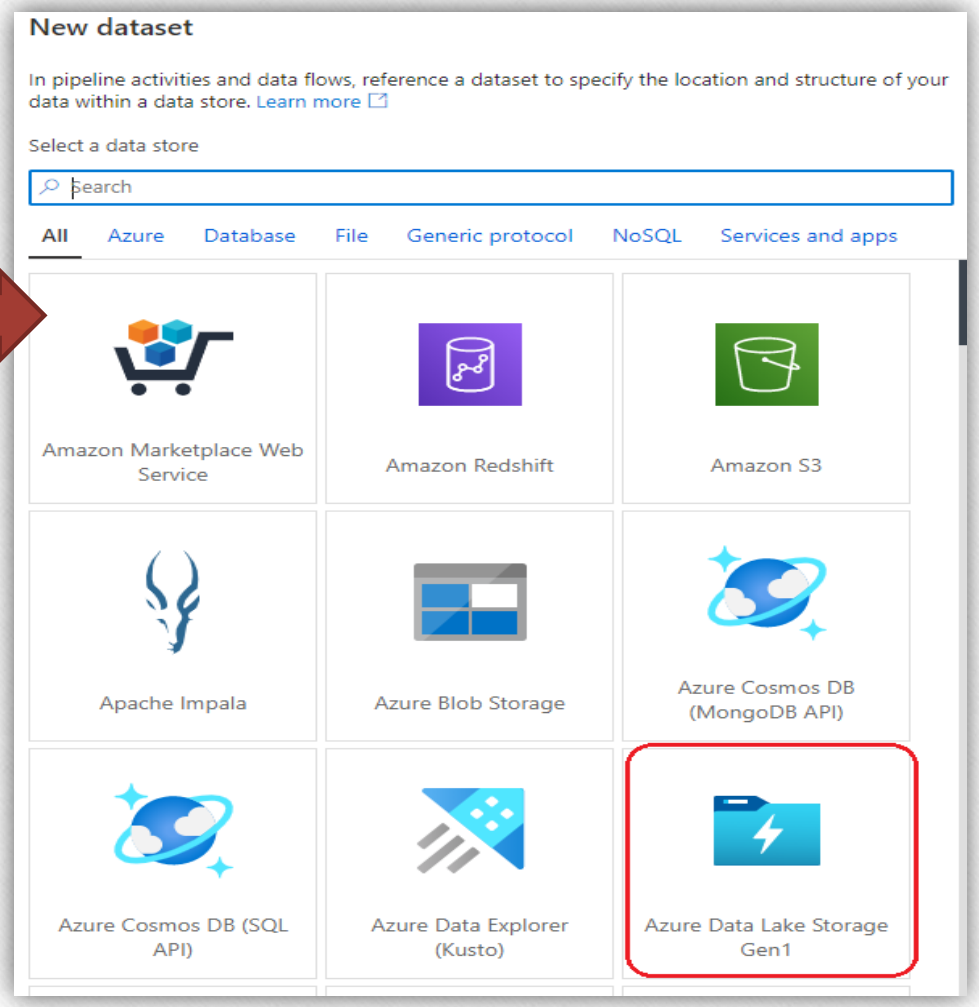

New linked service (Azure Data Lake Storage Gen1)
To avoid publishing immediately to Data Factory, please use Azure Key Vault to retrieve secrets securely. Learn more here
Name *
ADLS_GEN1_LS
Description
Connect via integration runtime *
AutoResolveIntegrationRuntime
Data Lake Store selection method
From Azure subscription
Enter manually
Azure subscription
Free Trial (bd66c874-48fe-4dc8-87c6-d2d0e4100413)
Data Lake Store account name *
azureadlsgen1
Tenant *
09d69696-ccc0-4686-b28c-6e053d4d3f03
Authentication type *
Service Principal
Service principal ID *
76c26ef2-3d84-428c-9dfa-09378aac05d1
Service principal key
Azure Key Vault
Service principal key *
Connection successful
Create
Back
Test connection
Cancel

Creating DataSet

In this procedure, you create two datasets: **InputDataset** and **OutputDataset**. These datasets are of type **DataLake And Blobstorage**. They refer to the Azure Storage linked service that you created in the previous section

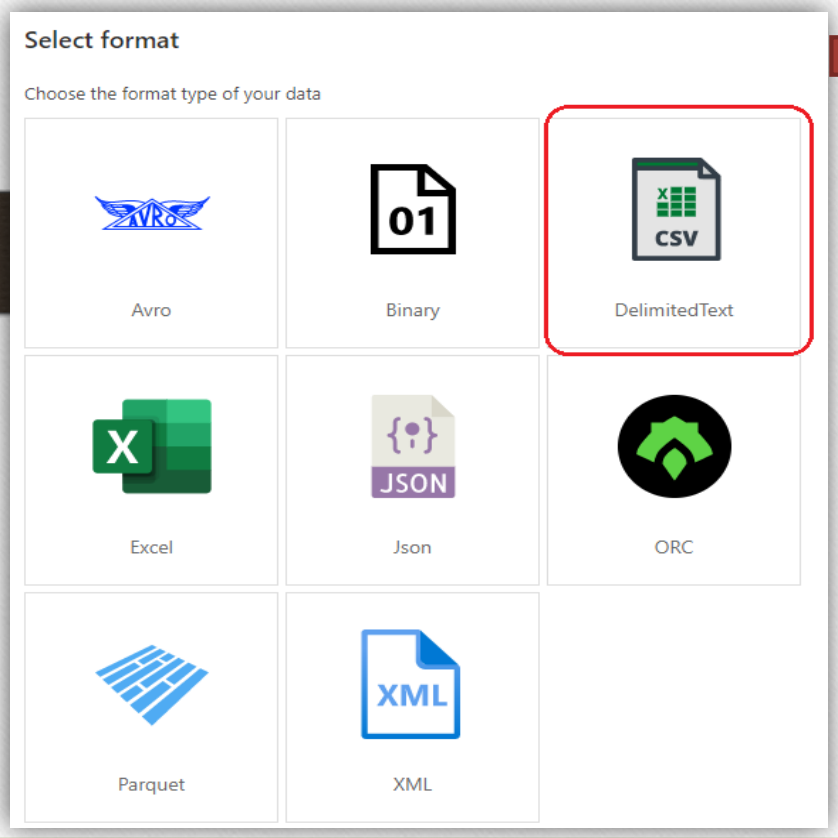
1.Select **Author** tab from the left pane.
2.Select the **+** (plus) button, and then select **Dataset**.

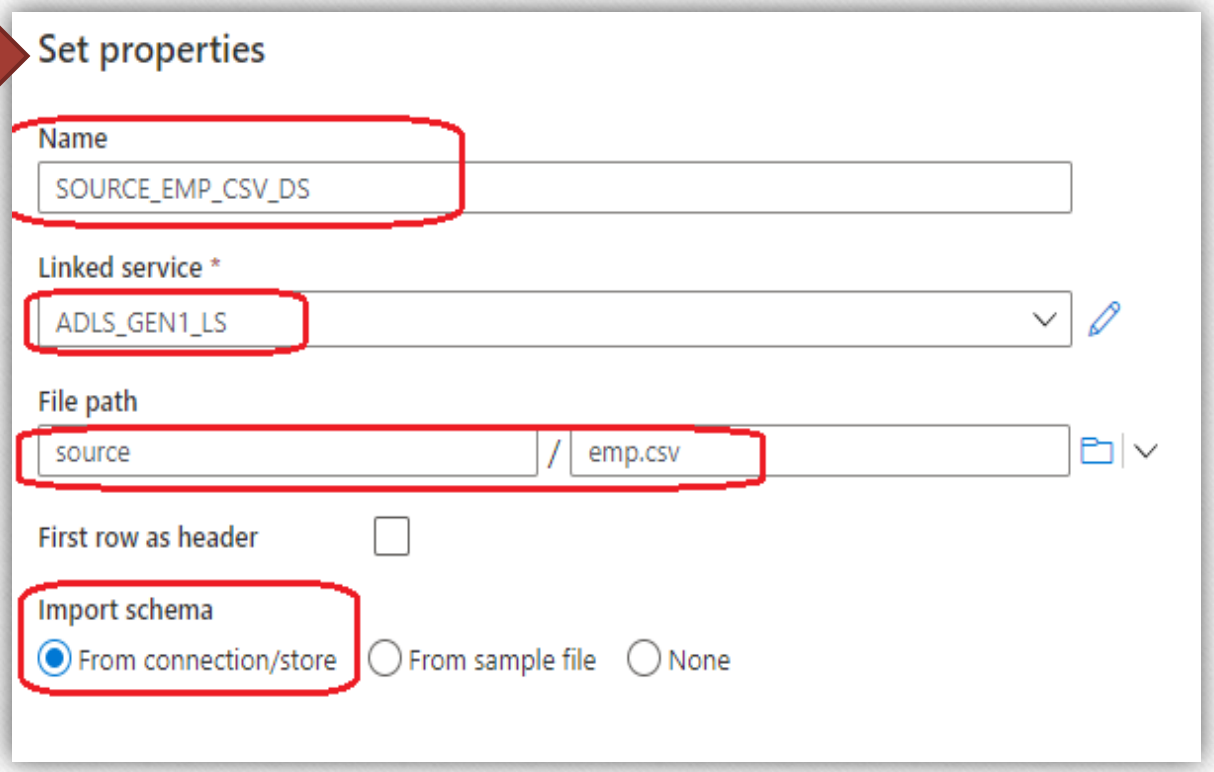




Creating DataSet

On the **Set Properties** page, complete following steps:
a. Under **Name**, enter **SourceDataset** as **SOURCE_EMP_CSV_DS**
b. For **Linked service**, select **AzureADLSLinkedService**. (**ADLS_GEN1_LS**)
c. For **File path**, select the **Browse** button.
d. In the **Choose a file or folder** window, browse to the **SOURCE** folder select the **emp.csv** file, and then select **OK**.
e. Select **OK**.





Mapping Data Flow

ADF Mapping Data Flow is to provide a fully visual experience with no coding required. Your Data Flows will execute on your own Azure Databricks cluster for scaled-out data processing using Spark. ADF handles all of the code translation, Spark optimization and execution of your data flow jobs.

Mapping Data Flows in ADF provide a way to transform data at scale without any coding required. Design a data transformation job in the ADF Data Flow designer by constructing a series of Source Transformations, followed by data transformation steps, then sink your results in a Sink(target) Transformation.

Start by creating data flows, then add a **Data Flow activity** to your pipeline to execute and test your data flow in Debug mode in the pipeline or use **"Trigger Now"** in the pipeline to test your Data Flow from a pipeline Activity.

You will then operationalize your Data Flow by **scheduling** and monitoring your **ADF pipeline** that is executing the Data Flow activity.

There is also a **Debug Mode** toggle switch on the Data Flow design surface to allow you to interactively build your data transformations in an interative data prep environment.

Mapping Data Flow

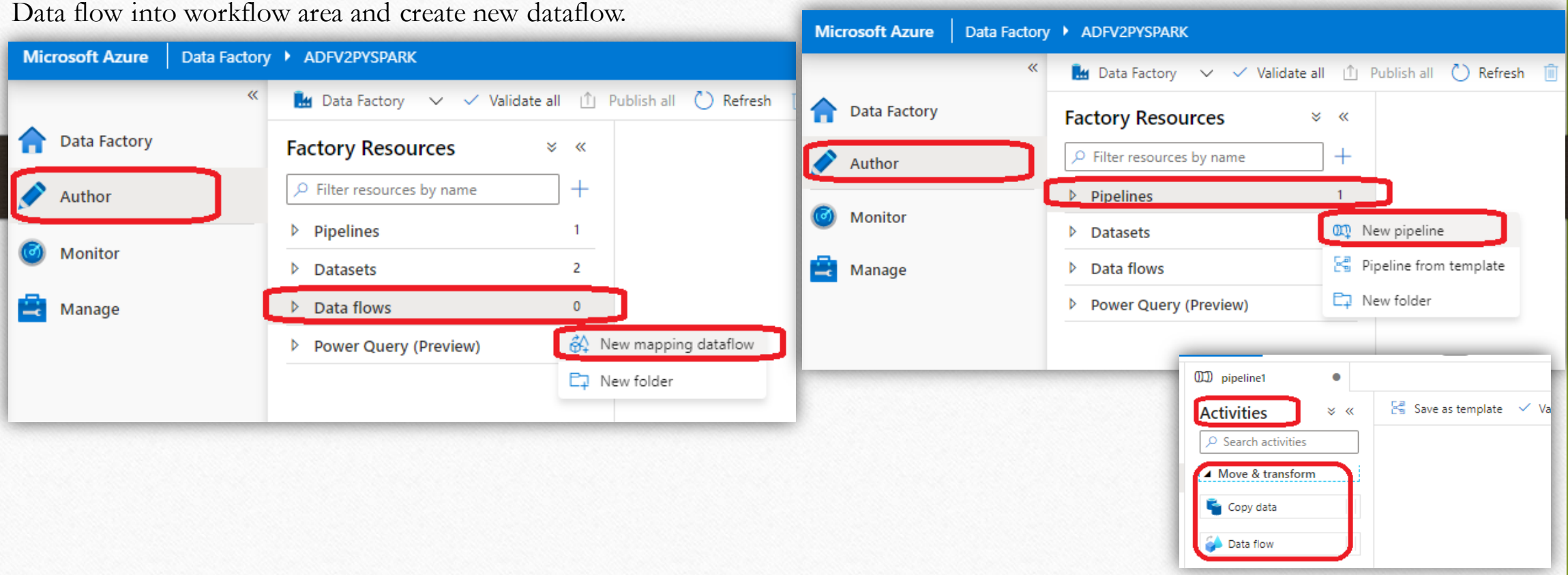
Creating Data Flow in Azure Data Factory.

Option 1:

Goto => Data Factory =>Author => Data Flows => Click on … => Click on New Mapping Dataflow.

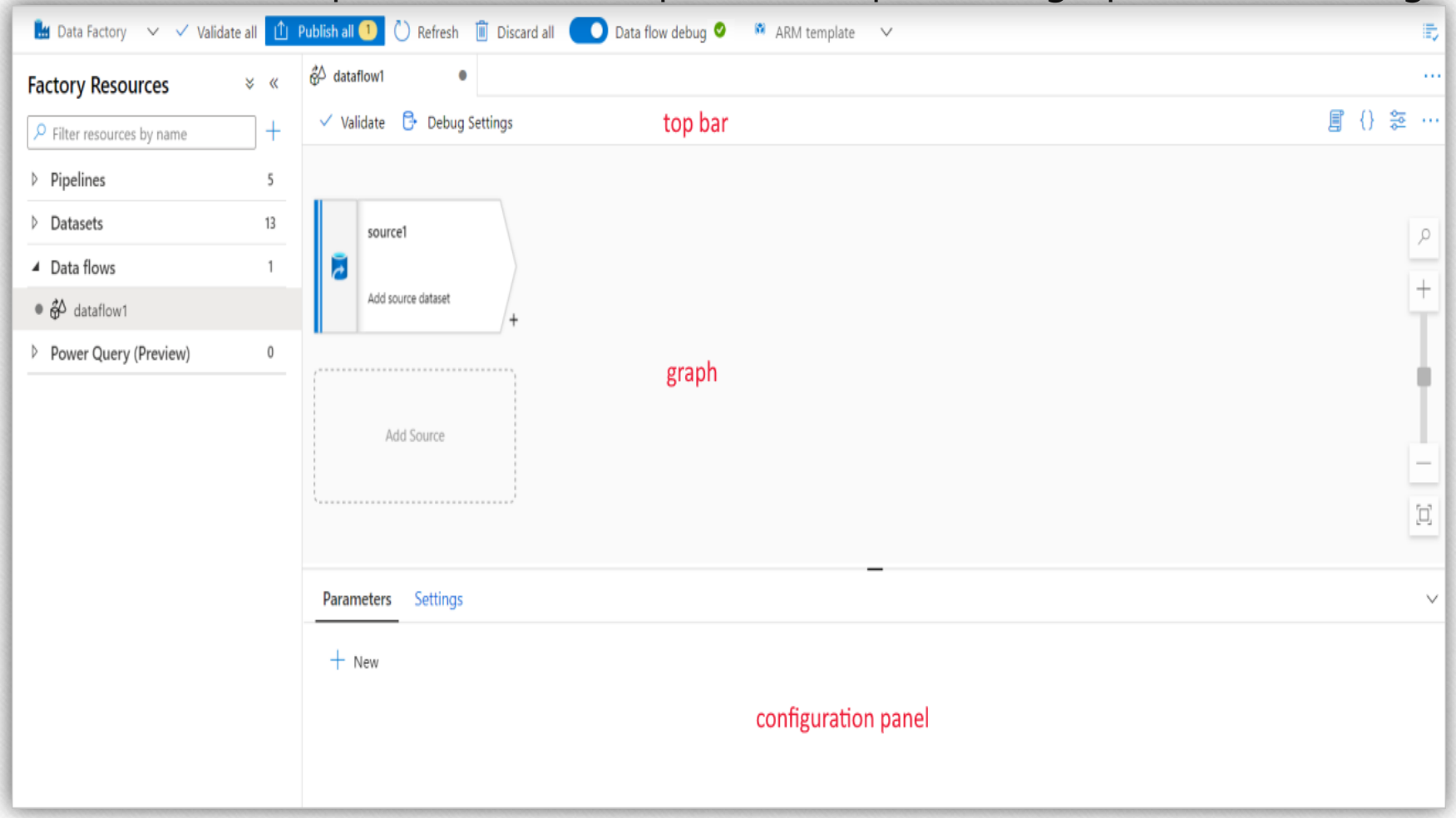
Option 2:

Goto => Data Factory => Author => Pipelines=> click on … => New Pipeline => Activities => Data Flow => Drag And Drop Data flow into workflow area and create new dataflow.



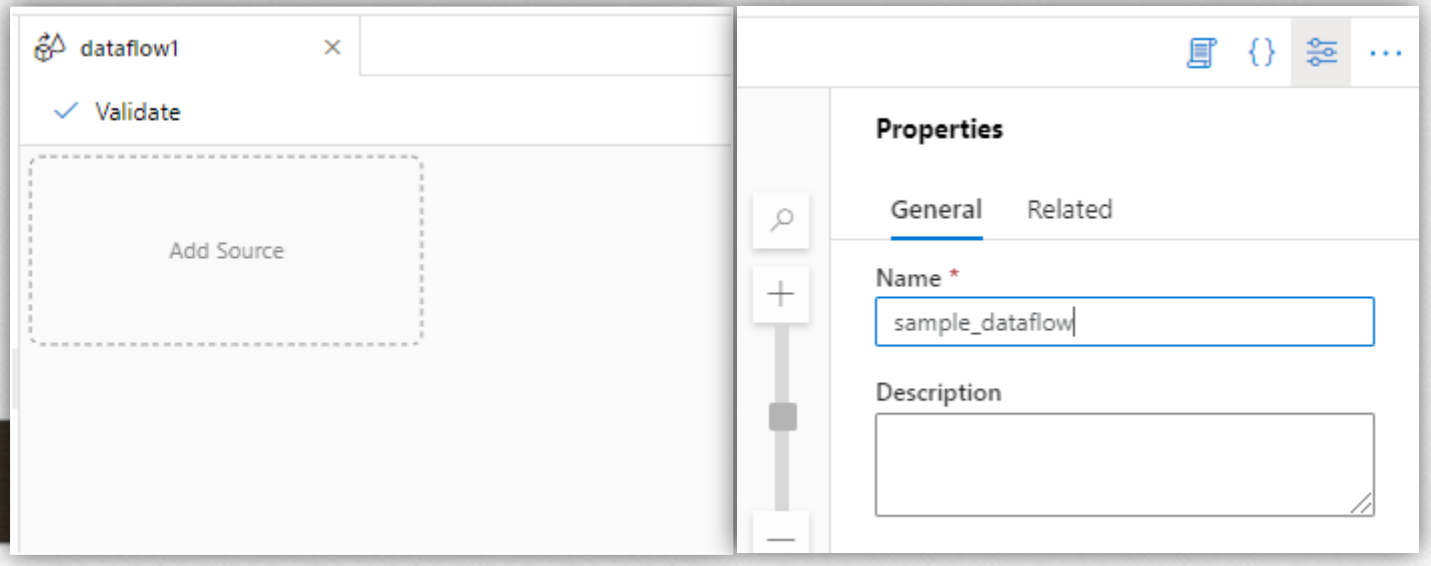
Using data flows Activity

Mapping data flow has a unique authoring canvas designed to make building transformation logic easy. The data flow canvas is separated into three parts: the top bar, the graph, and the configuration panel.

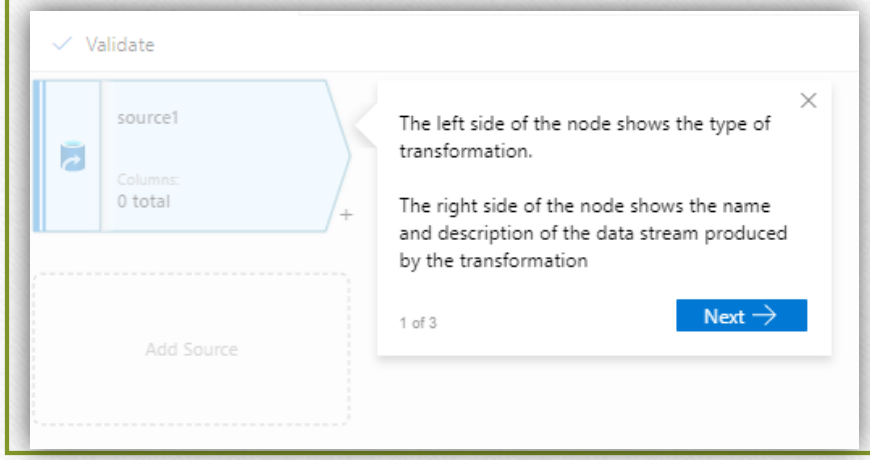


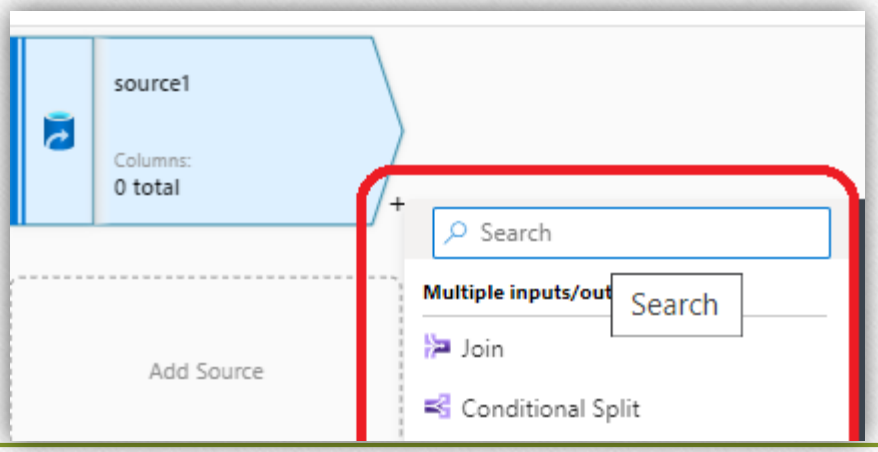
Creating Data Flow and Adding Source

Enter The Dataflow Name in properties – General tab as sample_dataflow  Click on this it will hive the window.



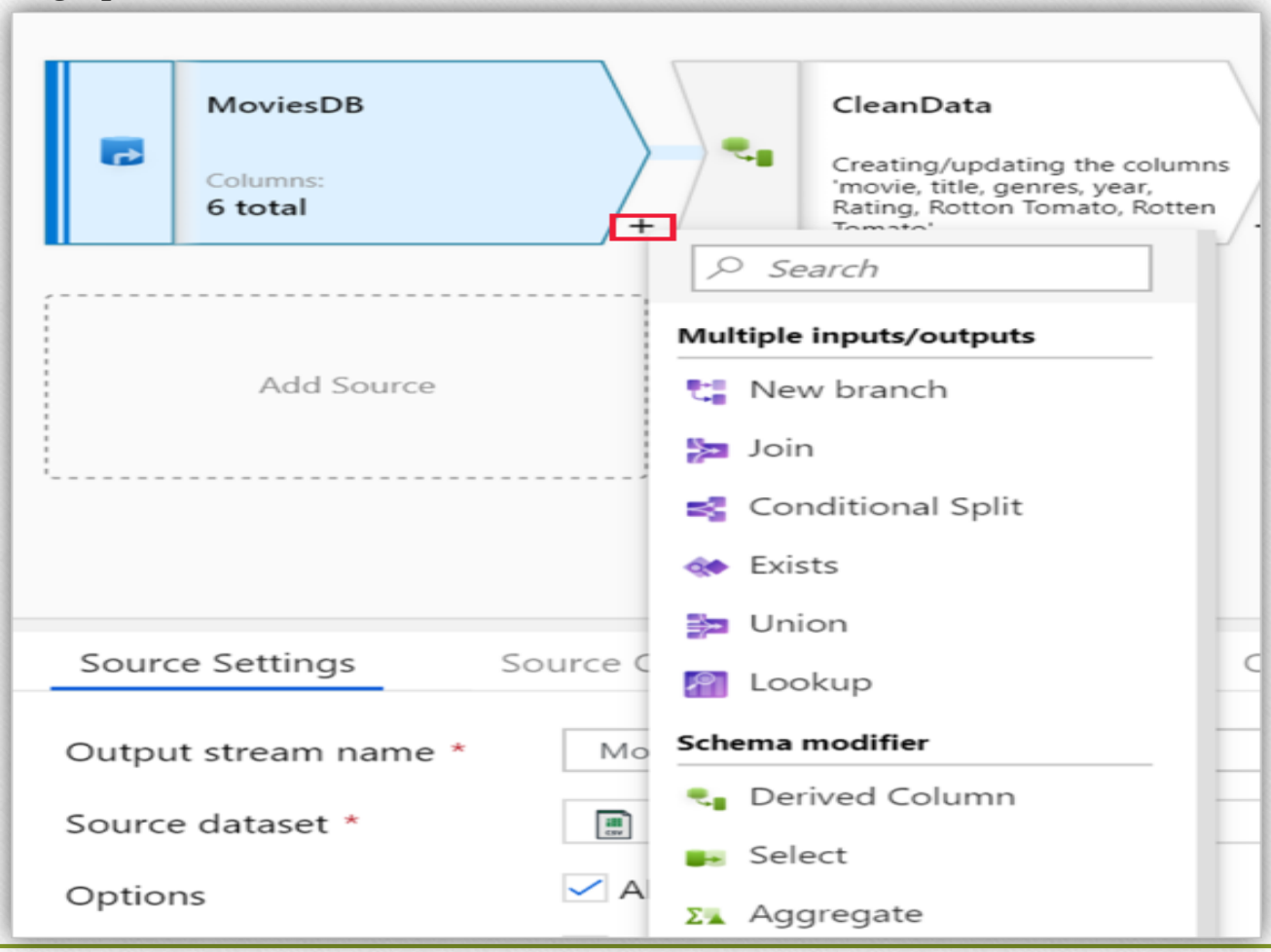
Click On **Add Source** box and select Source dataset. Click **+** button to add Transformations on top of Source.





Graph

The graph displays the transformation stream. It shows the lineage of source data as it flows into one or more sinks. To add a new source, select Add source. To add a new transformation, select the plus sign on the lower right of an existing transformation. Learn more on how to manage the data flow graph.

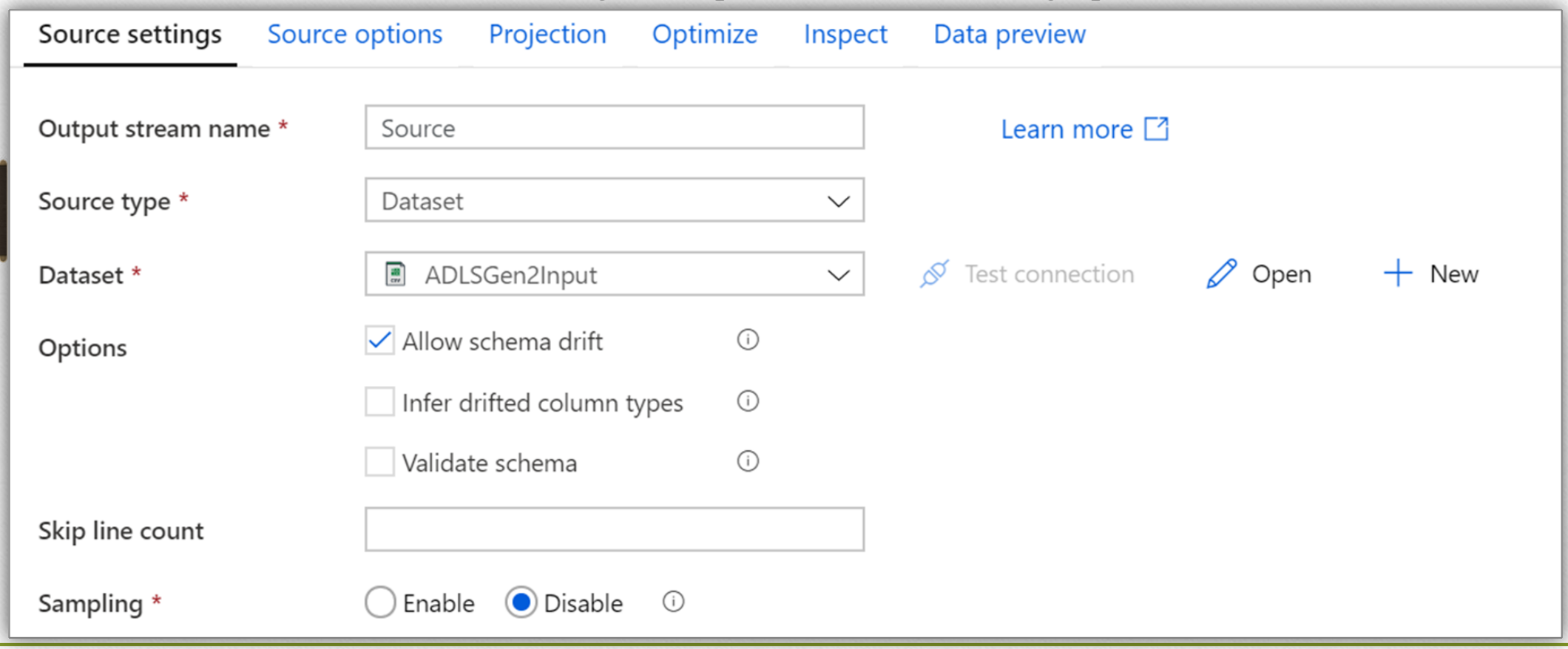


Configuration panel

The configuration panel shows the settings specific to the currently selected transformation. If no transformation is selected, it shows the data flow. In the overall data flow configuration, you can add parameters via the Parameters tab.

Transformation settings

The first tab in each transformation's configuration pane contains the settings specific to that transformation.



Source Settings Options:

Output stream name: The name of the source transformation.

Source type: Choose whether you want to use an inline dataset or an existing dataset object.

Test connection: Test whether or not the data flow's Spark service can successfully connect to the linked service used in your source dataset. Debug mode must be on for this feature to be enabled.

Schema drift: Schema drift is the ability of Data Factory to natively handle flexible schemas in your data flows without needing to explicitly define column changes.

1. Select the **Allow schema drift** check box if the source columns will change often. This setting allows all incoming source fields to flow through the transformations to the sink.
2. Selecting **Infer drifted column types** instructs Data Factory to detect and define data types for each new column discovered. With this feature turned off, all drifted columns will be of type string.

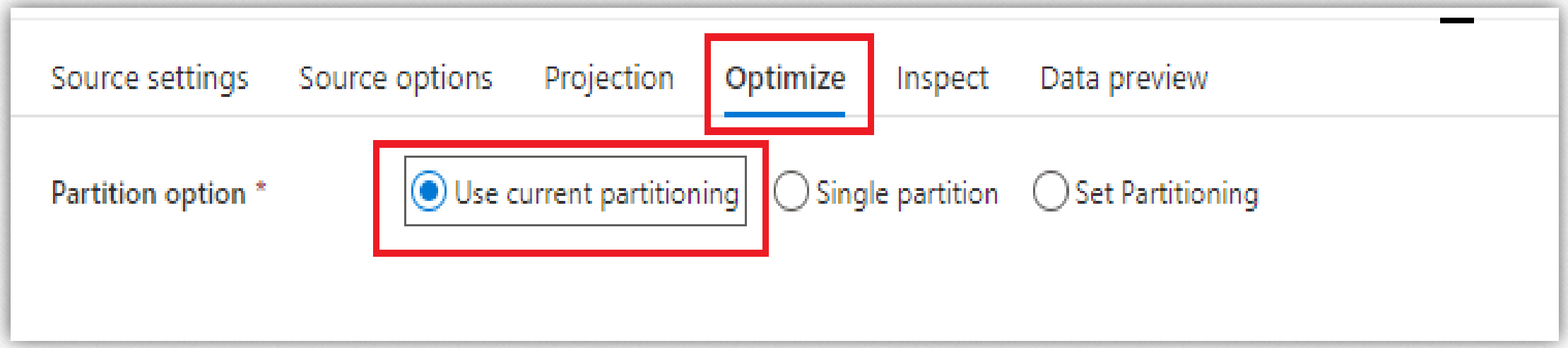
Validate schema: If **Validate schema** is selected, the data flow will fail to run if the incoming source data doesn't match the defined schema of the dataset.

Skip line count: The **Skip line count** field specifies how many lines to ignore at the beginning of the dataset.

Sampling: Enable **Sampling** to limit the number of rows from your source. Use this setting when you test or sample data from your source for debugging purposes.

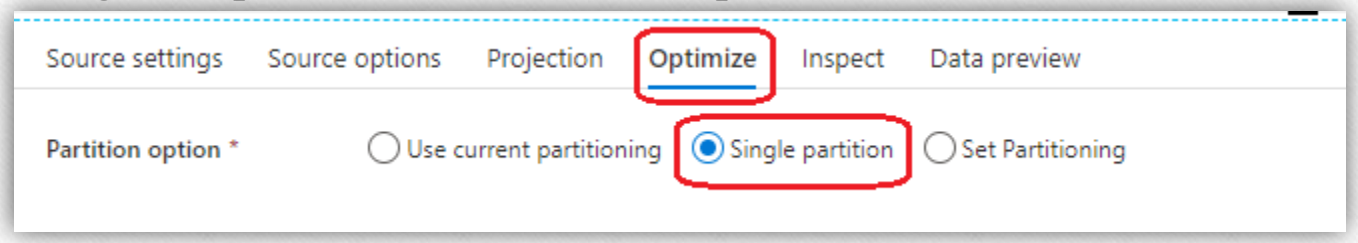
Optimize

The Optimize tab contains settings to configure the partitioning scheme of the Spark cluster. This tab exists in every transformation of data flow and specifies whether you want to repartition the data after the transformation has completed. Adjusting the partitioning provides control over the distribution of your data across compute nodes and data locality optimizations that can have both positive and negative effects on your overall data flow performance.

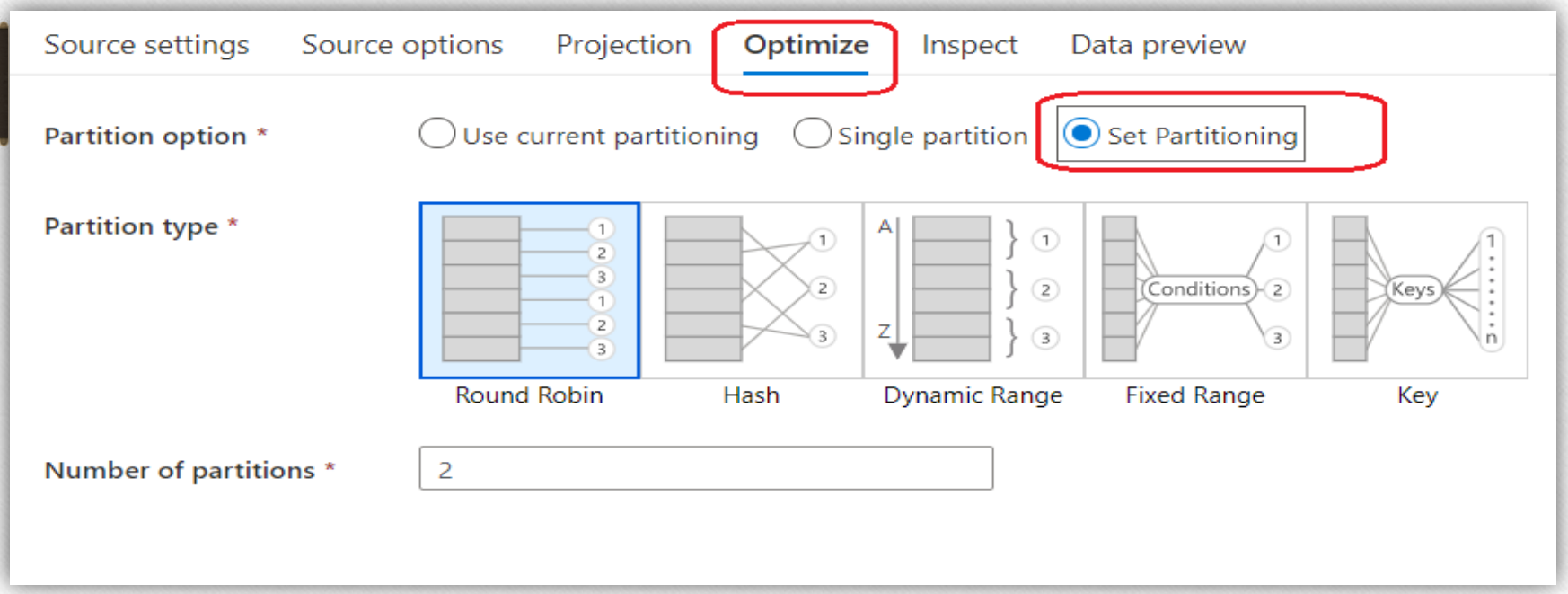


By default, Use current partitioning is selected which instructs Azure Data Factory keep the current output partitioning of the transformation. As repartitioning data takes time, Use current partitioning is recommended in most scenarios. Scenarios where you may want to repartition your data include after aggregates and joins that significantly skew your data or when using Source partitioning on a SQL DB.

Single partition combines all the distributed data into a single partition. This is a very slow operation that also significantly affects all downstream transformation and writes. The Azure Data Factory highly recommends against using this option unless there is an explicit business reason to do so.



Set Partitioning



Set Partitioning

Round robin

Round robin distributes data equally across partitions. Use round-robin when you don't have good key candidates to implement a solid, smart partitioning strategy. You can set the number of physical partitions.

Hash

Azure Data Factory produces a hash of columns to produce uniform partitions such that rows with similar values fall in the same partition. When you use the Hash option, test for possible partition skew. You can set the number of physical partitions.

Dynamic range

The dynamic range uses Spark dynamic ranges based on the columns or expressions that you provide. You can set the number of physical partitions.

Fixed range

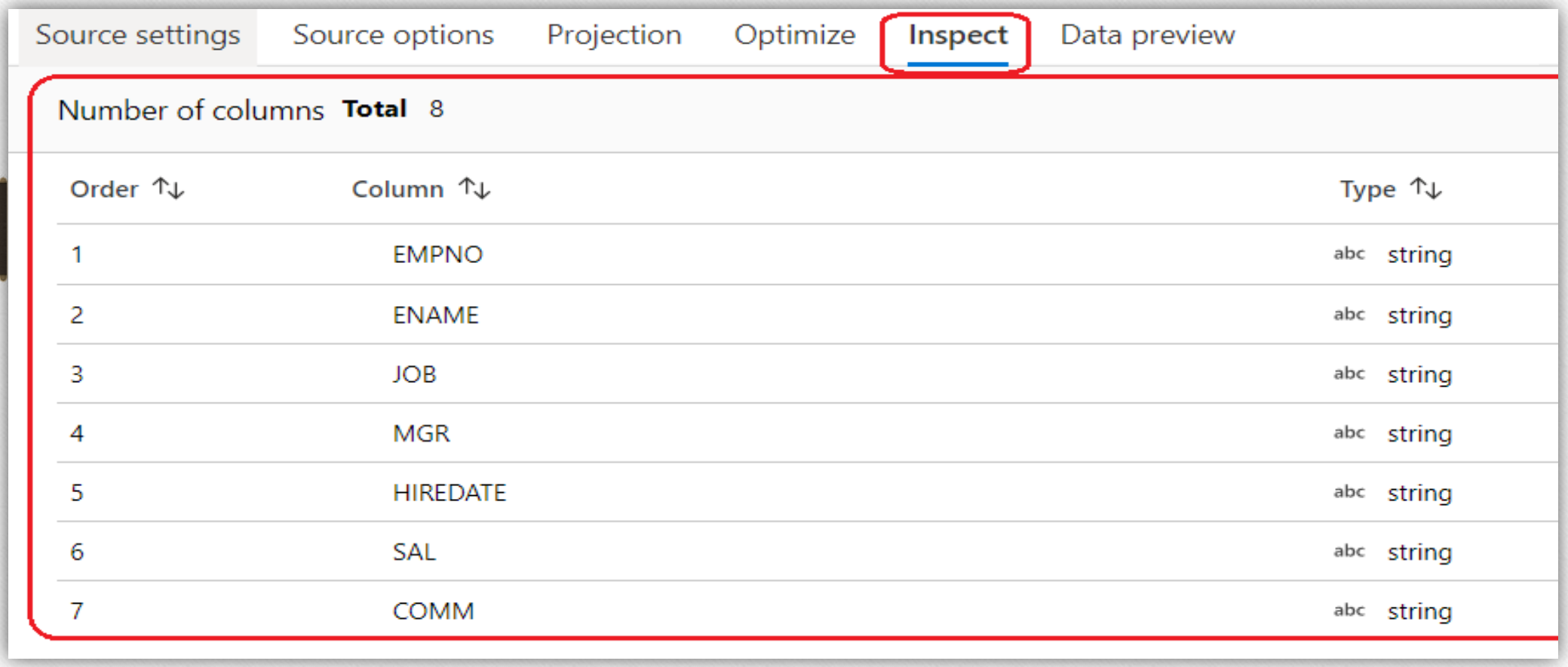
Build an expression that provides a fixed range for values within your partitioned data columns. To avoid partition skew, you should have a good understanding of your data before you use this option. The values you enter for the expression are used as part of a partition function. You can set the number of physical partitions.

Key

If you have a good understanding of the cardinality of your data, key partitioning might be a good strategy. Key partitioning creates partitions for each unique value in your column. You can't set the number of partitions because the number is based on unique values in the data.

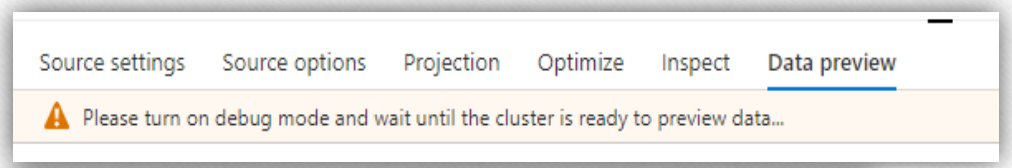
Inspect

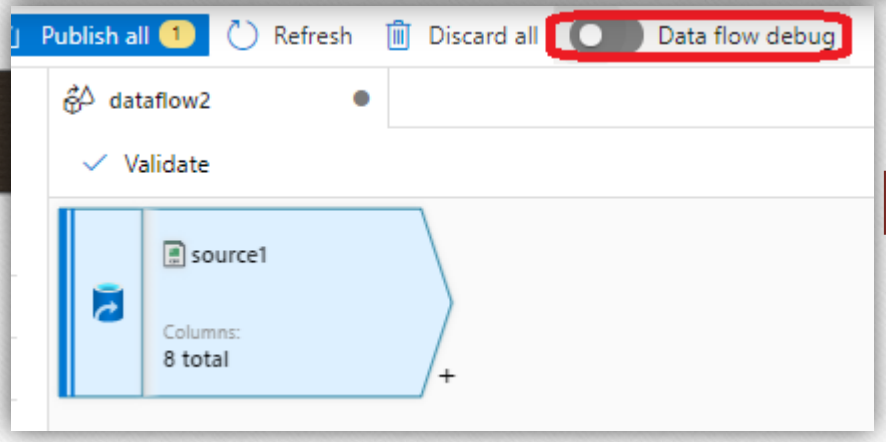
The Inspect tab provides a view into the metadata of the data stream that you're transforming. You can see column counts, the columns changed, the columns added, data types, the column order, and column references. Inspect is a read-only view of your metadata. You don't need to have debug mode enabled to see metadata in the Inspect pane.

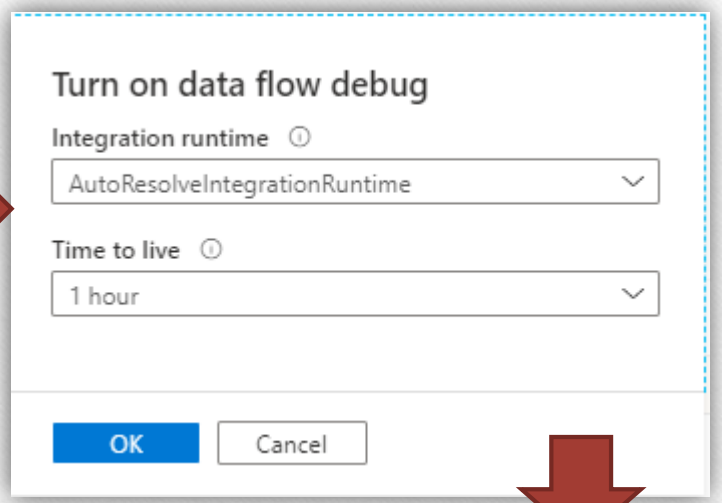


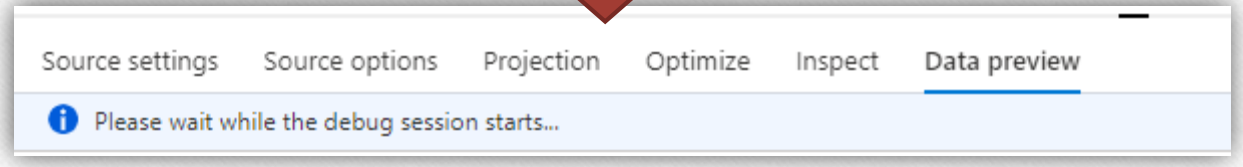
Data Preview

With debug on, the Data Preview tab will light-up on the bottom panel. Without debug mode on, Data Flow will show you only the current metadata in and out of each of your transformations in the Inspect tab. The data preview will only query the number of rows that you have set as your limit in your debug settings. Click **Refresh** to fetch the data preview.



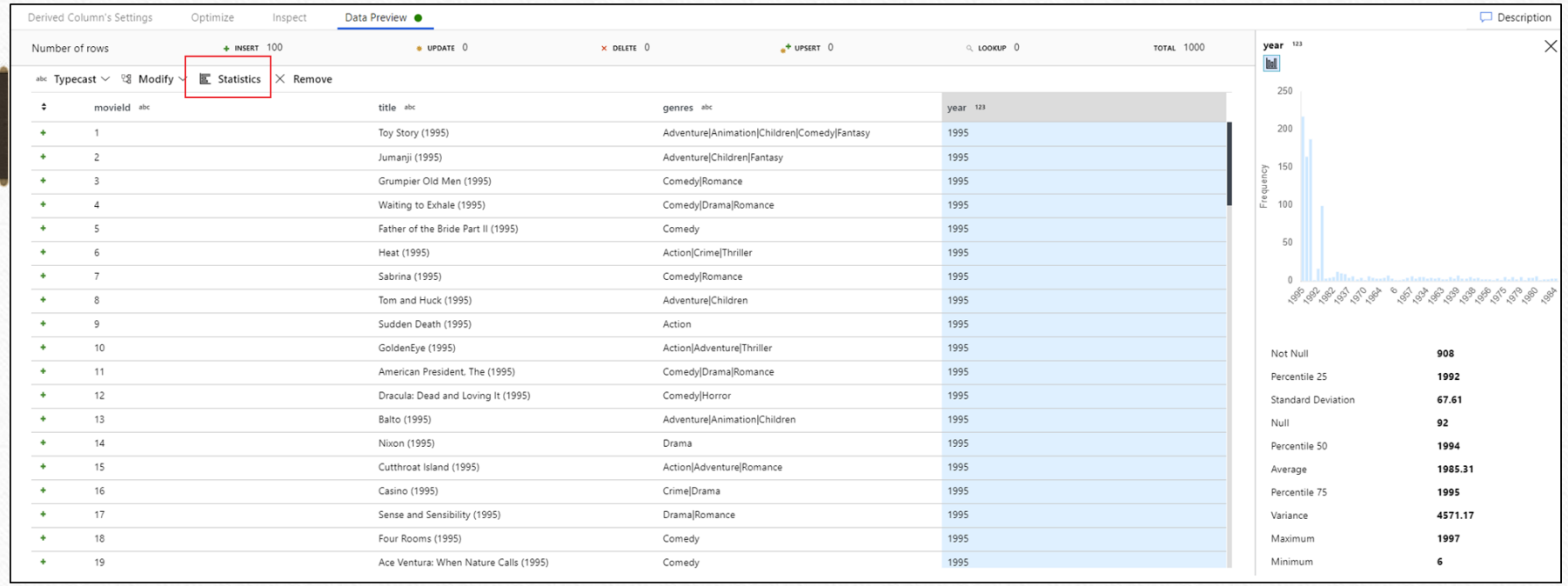






Data profiling

Selecting a column in your data preview tab and clicking **Statistics** in the data preview toolbar will pop up a chart on the far-right of your data grid with detailed statistics about each field. Azure Data Factory will make a determination based upon the data sampling of which type of chart to display. High-cardinality fields will default to NULL/NOT NULL charts while categorical and numeric data that has low cardinality will display bar charts showing data value frequency. You'll also see max/len length of string fields, min/max values in numeric fields, standard dev, percentiles, counts, and average.



MAPPING DATA FLOW TRANSFORMATIONS

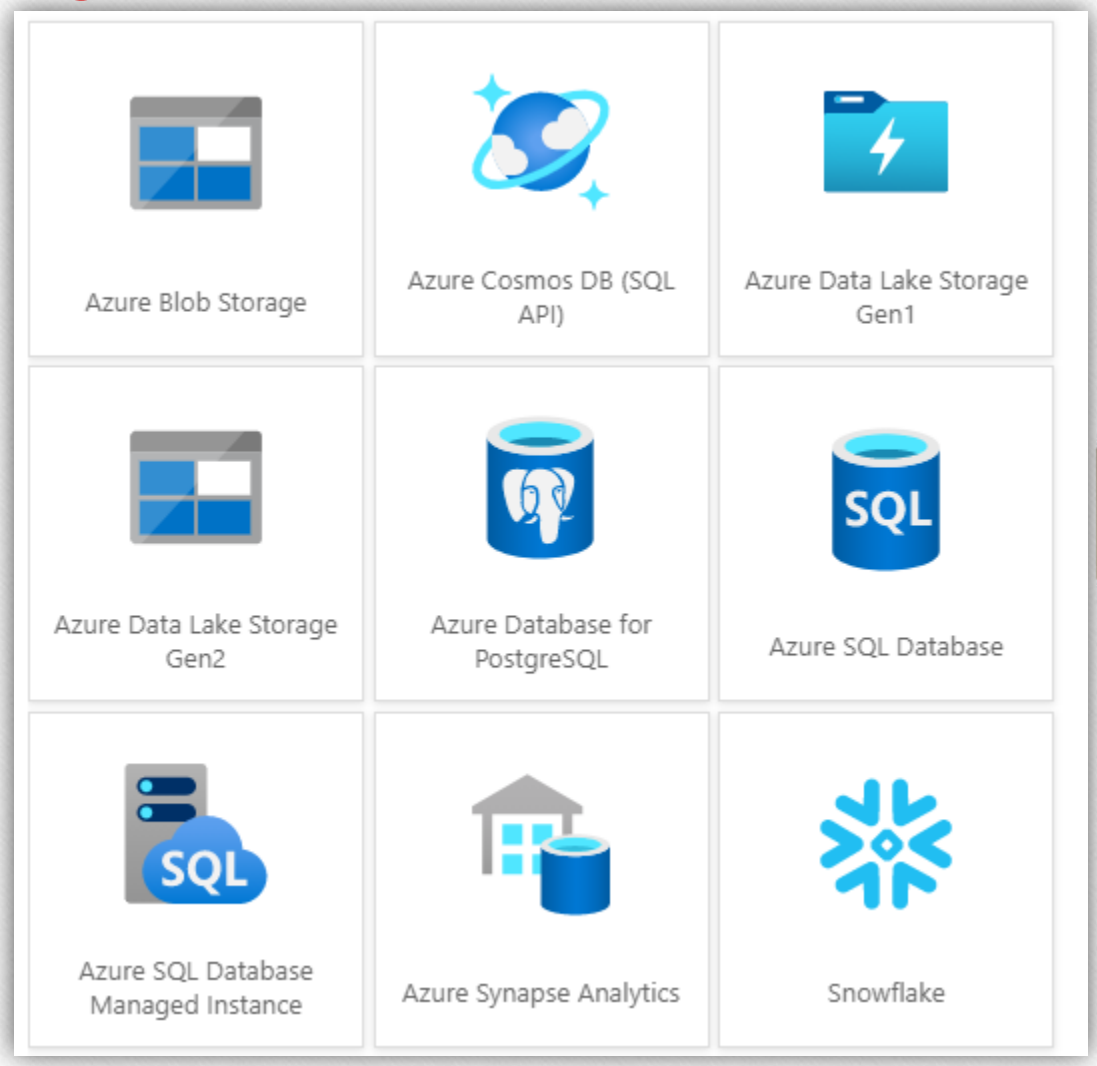
Type	Icon	Transform	Description
Dataset		Source	Source for your dataflow.
		Sink	Destination for your dataflow.
Multiple Inputs/Outputs		New Branch	Create a new flow branch with the same data.
		Join	Join data from two streams based on a condition.
		Conditional Split	Route data into different streams based on conditions.
		Union	Collect data from multiple streams.
		Lookup	Lookup additional data from another stream.
Schema Modifier		Derived Column	Compute new columns based on existing ones.
		Aggregate	Calculate aggregations on the stream.
		Surrogate Key	Adds a surrogate key column to output stream from a specific value.
		Pivot	Row values transformed into individual columns.
		Unpivot	Column values transformed into individual rows.
Row Modifier		Exists	Check the existence of data in another stream.
		Select	Choose columns to flow to the next stream.
		Filter	Filter rows in the stream based on a condition.
		Sort	Order data in the stream based on column(s).

Data Flow Supporting Sources

Data Flow currently **allowing Azure Blob Storage, Azure Cosmos DB, Azure ADLS Gen1, Azure ADLS Gen2, Azure Database for PostgresSQL, Azure SQL DB, Azure SQL DW(synapse)** and **Snowflake** natively for **Source** and **Sink**.

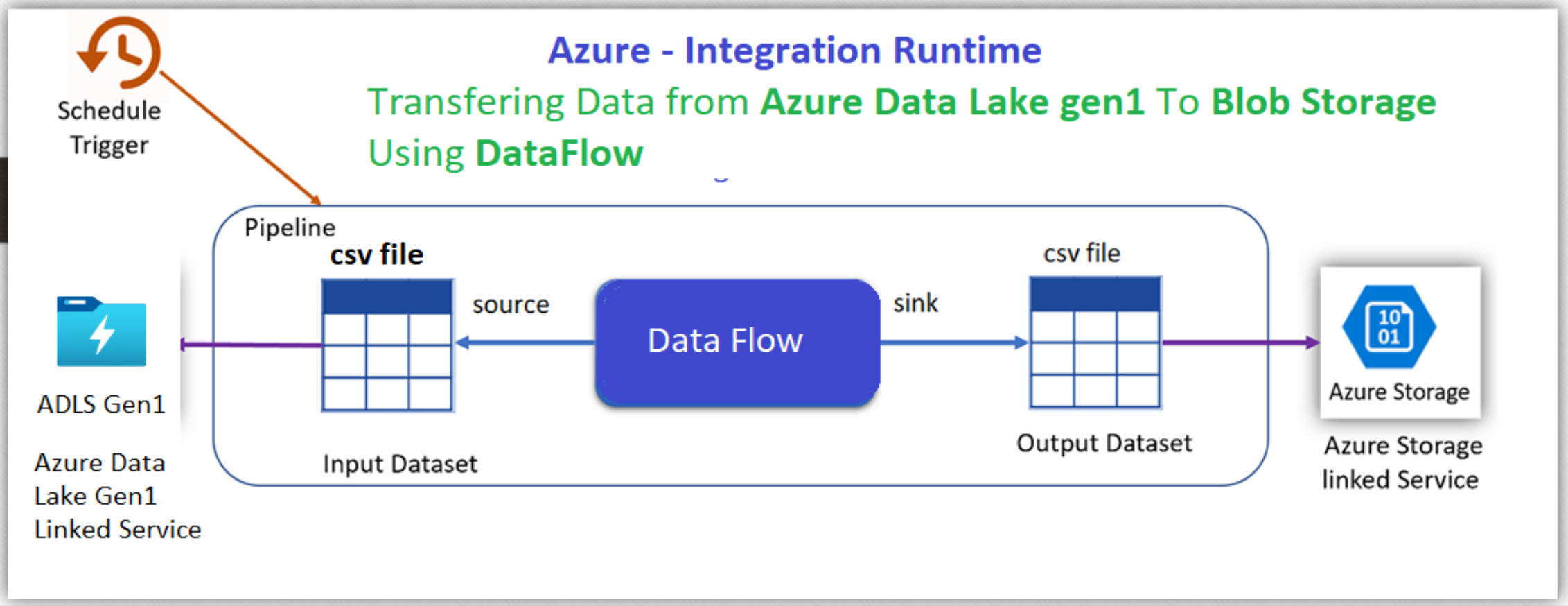
How To connect Other Sources in Data Flow?

Use the Copy Activity to stage data from any of the other connectors and then execute a Data Flow activity to transform data after it's been staged. For example, your pipeline will first Copy into Blob and then a Data Flow activity will use a dataset in Source to transform that data.

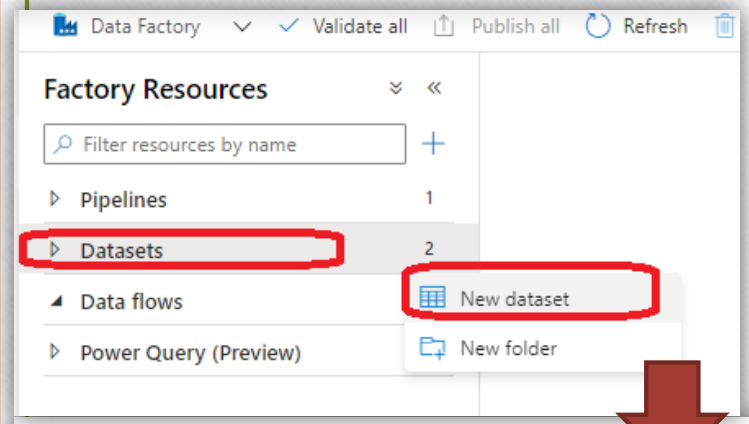


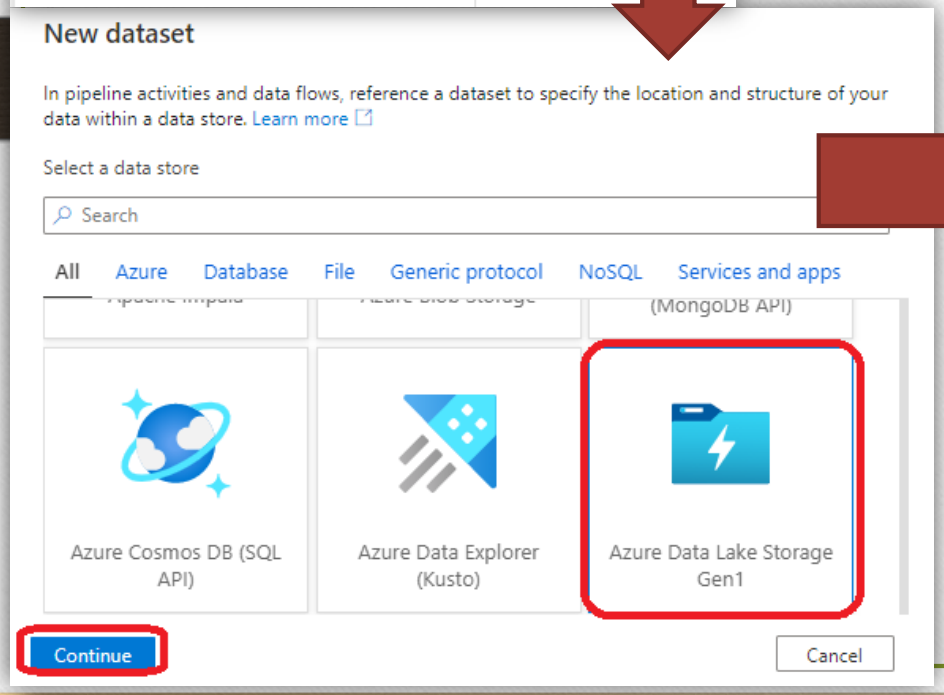
Azure Data Factory Data Flow Activity

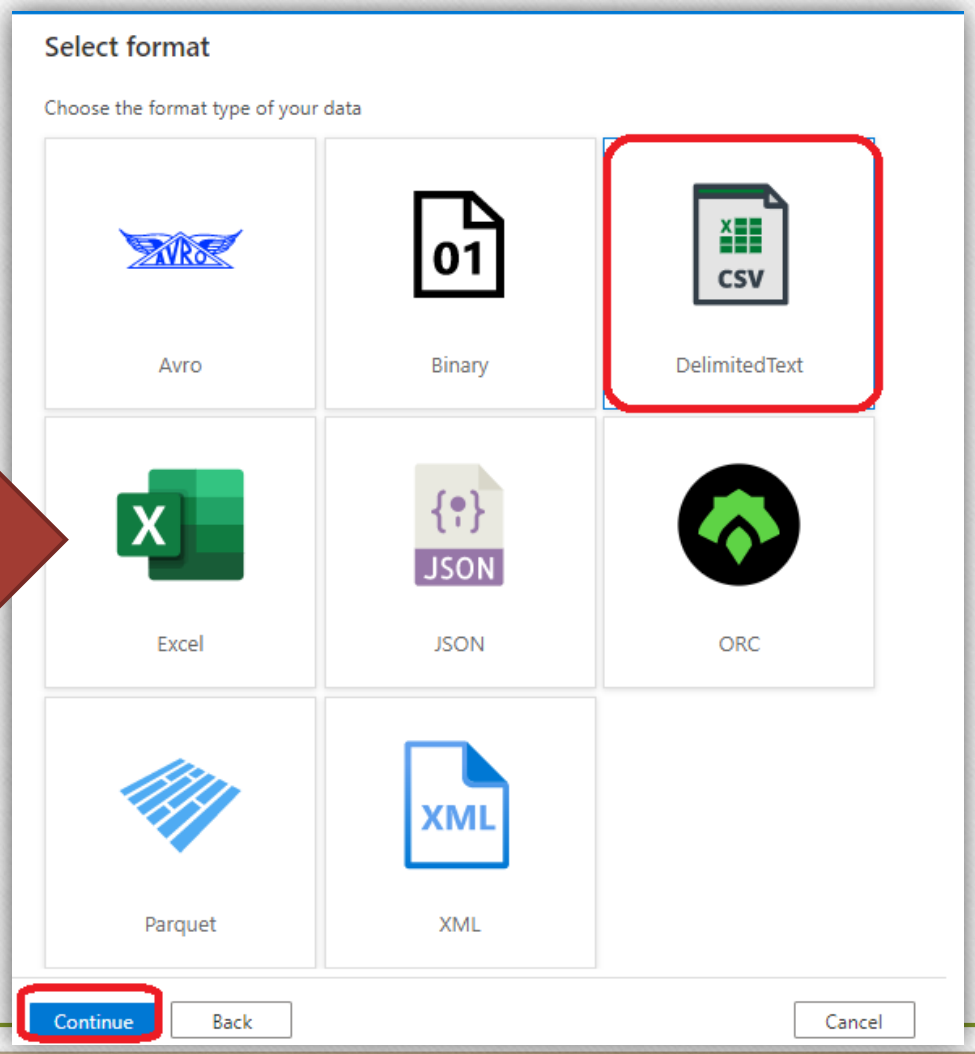
1) Creating two source datasets for emp and dept files to understand the joins and those types.



Creating Dataset for Source files. One for emp.csv file and another for dept.csv. Follow below steps for creating dataset.
Datasets => New Dataset=> Select Azure Data Lake Storage Gen1 => Select Delimited Text file format => Continue







Linked Service for ADLS Gen1

Create Two Datasets for source. Emp.csv and dept.csv files.

Set properties

Name
DS_ADLS_EMP_SRC_CSV

Linked service *
LS_ADLS_SRC

File path
landing / emp.csv

First row as header ☐

Import schema
(•) From connection/store ( ) From sample file ( ) None

Set properties

Name
DS_ADLS_DEPT_SRC_CSV

Linked service *
LS_ADLS_SRC

File path
landing / dept.csv

First row as header ☑

Import schema
(•) From connection/store ( ) From sample file ( ) None

New linked service (Azure Data Lake Storage Gen1)

Name *
LS_ADLS_SRC

Description

Connect via integration runtime * ⓘ
AutoResolveIntegrationRuntime

Data Lake Store selection method ⓘ
(•) From Azure subscription ( ) Enter manually

Azure subscription ⓘ
Free Trial (532945f0-7bb0-4f10-b851-9de86e121ae7)

Data Lake Store account name *
databricksadlsgen1

Tenant *
b38c808f-390b-4056-9a18-d5a9b571cf98

Authentication type *
Service Principal

Service principal ID * ⓘ
9768d5c8-5df2-4125-818b-37b0d3472218

Service principal key | Azure Key Vault

Service principal key *
••••••••••••••••••••••••••••••••••••

Azure cloud type ⓘ
Data Factory's cloud type

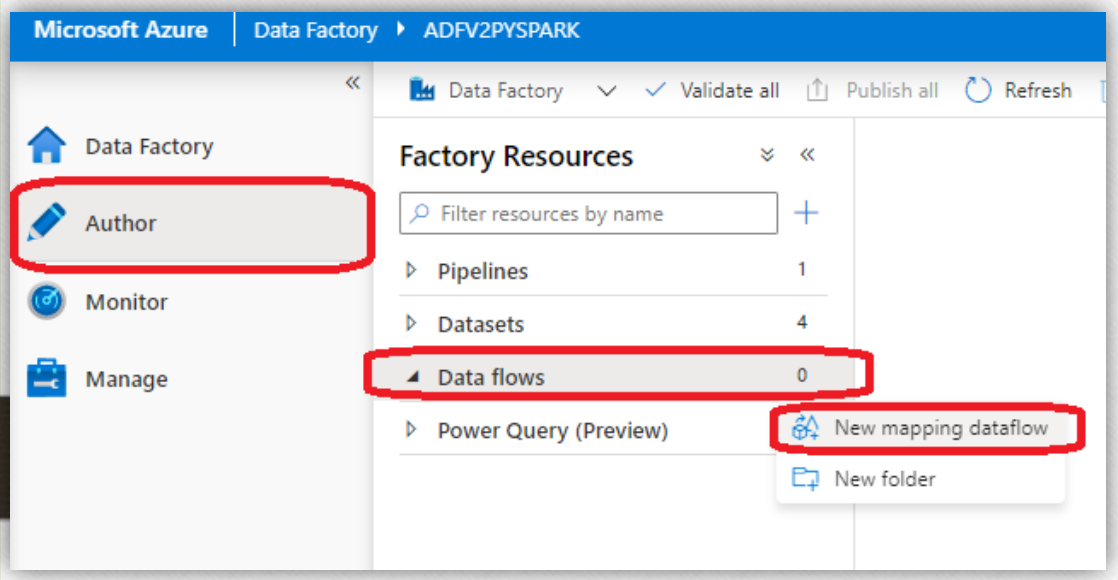
Annotations
+ New

Connection successful

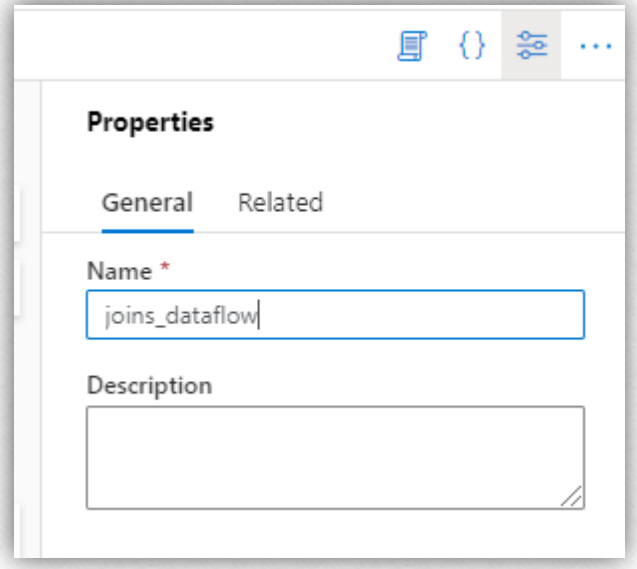
Create | Test connection | Cancel

Creating New Mapping Data Flow.

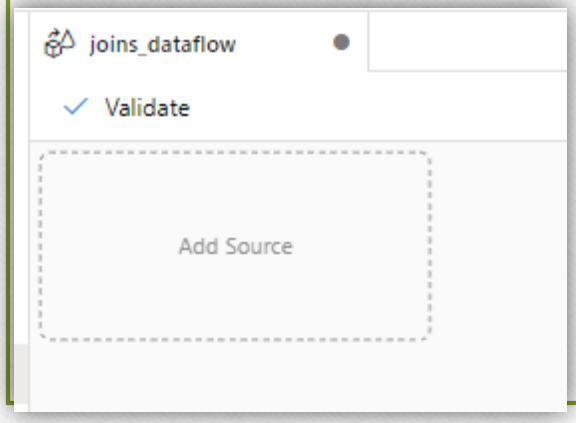
Author => Data Flows => click on New Mapping dataflow



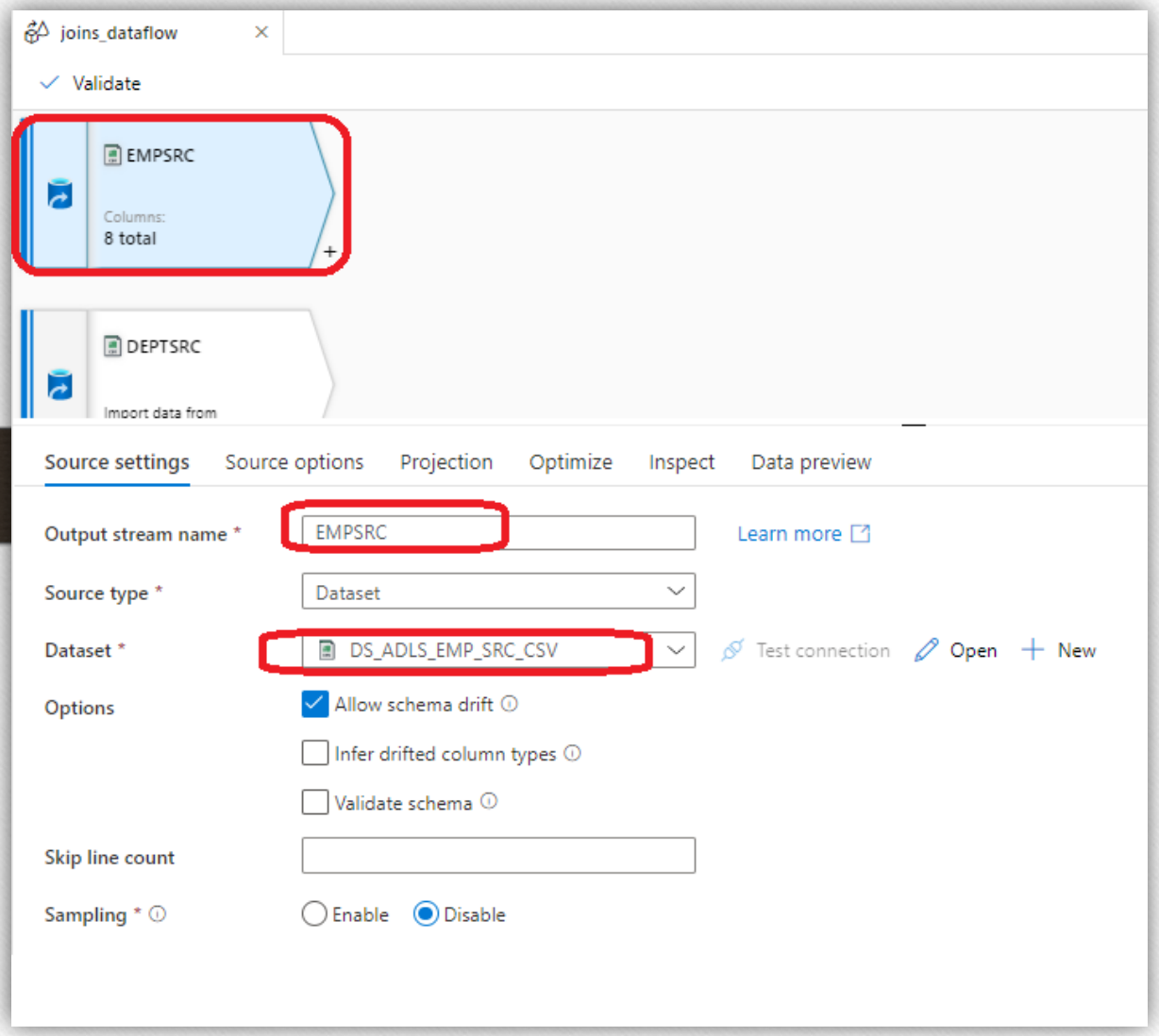
Rename Dataflow name. I have created as joins_dataflow



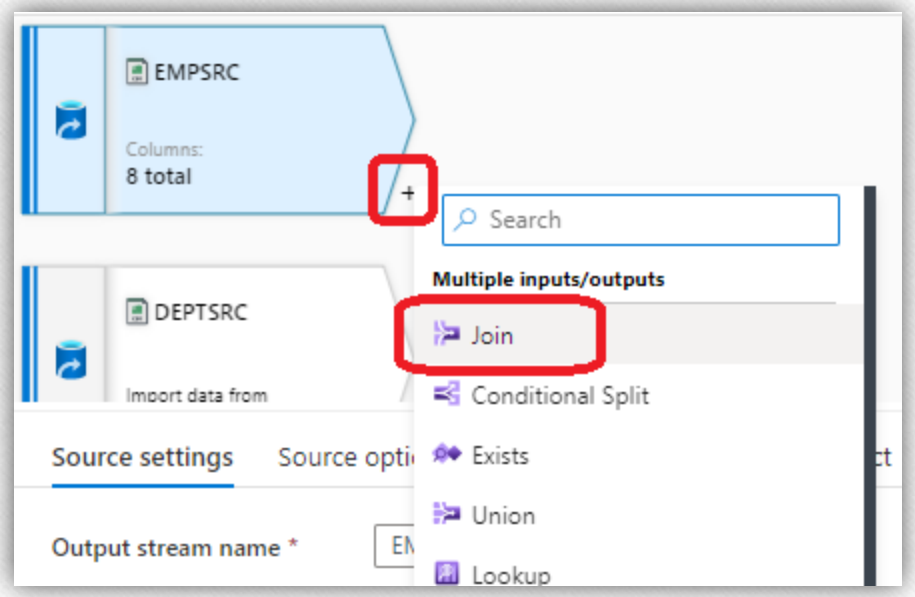
Click on **Add Source** and select Source EMP dataset.



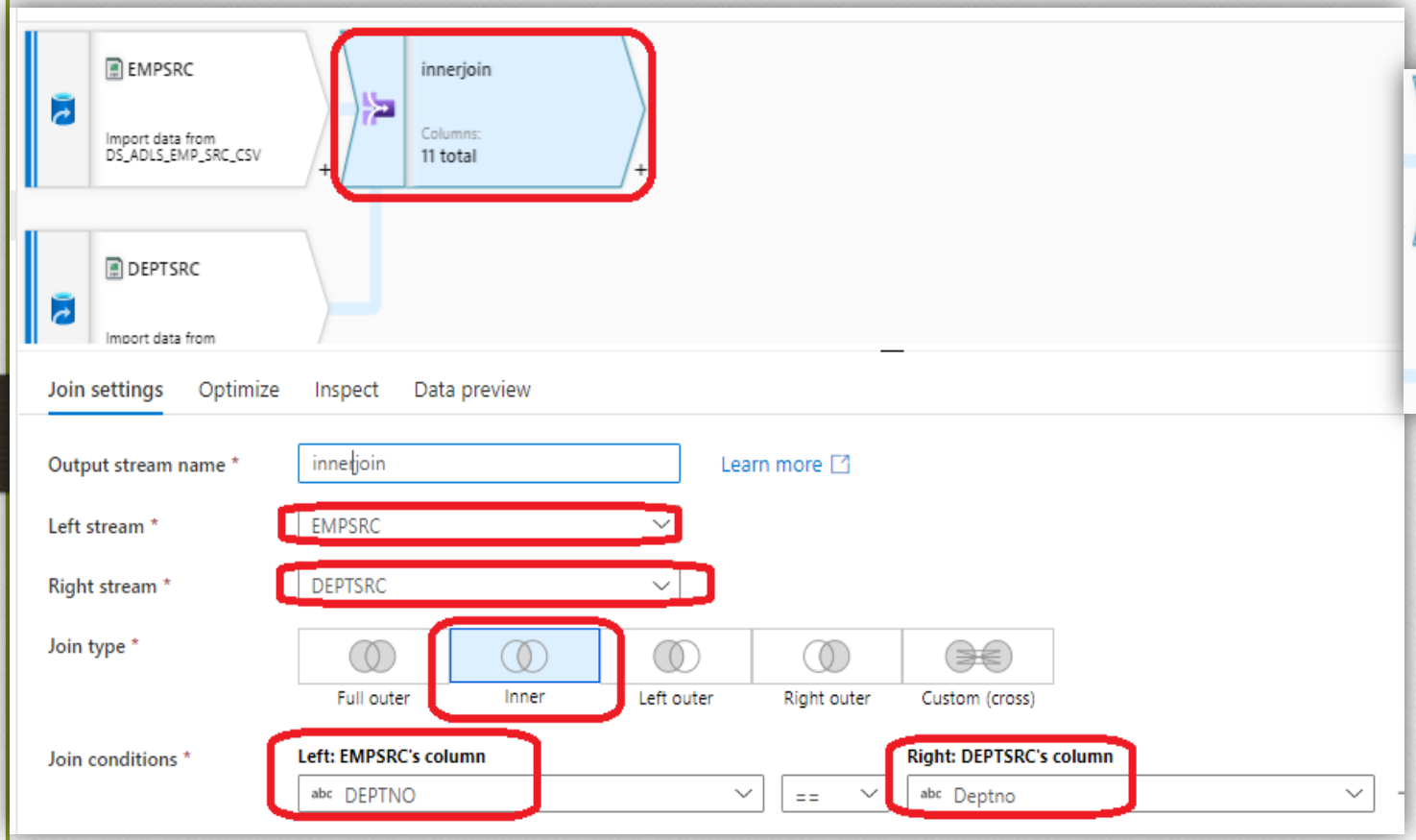
Create Two Sources. One for emp.csv file dataset and another for dept.csv file dataset.



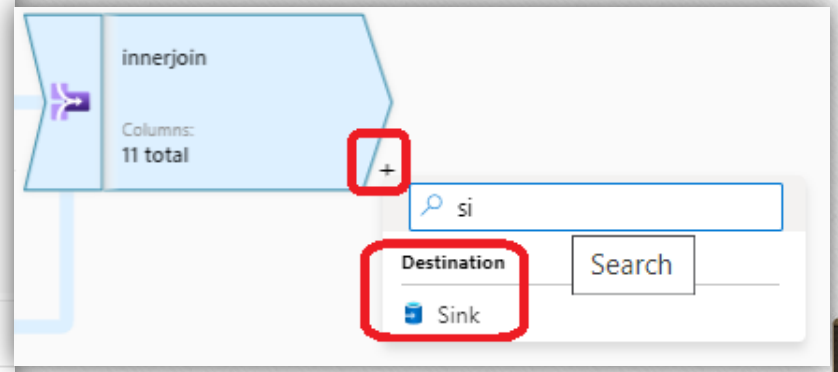
Click on + in source dataset. Select join it will add **Join Transformation**



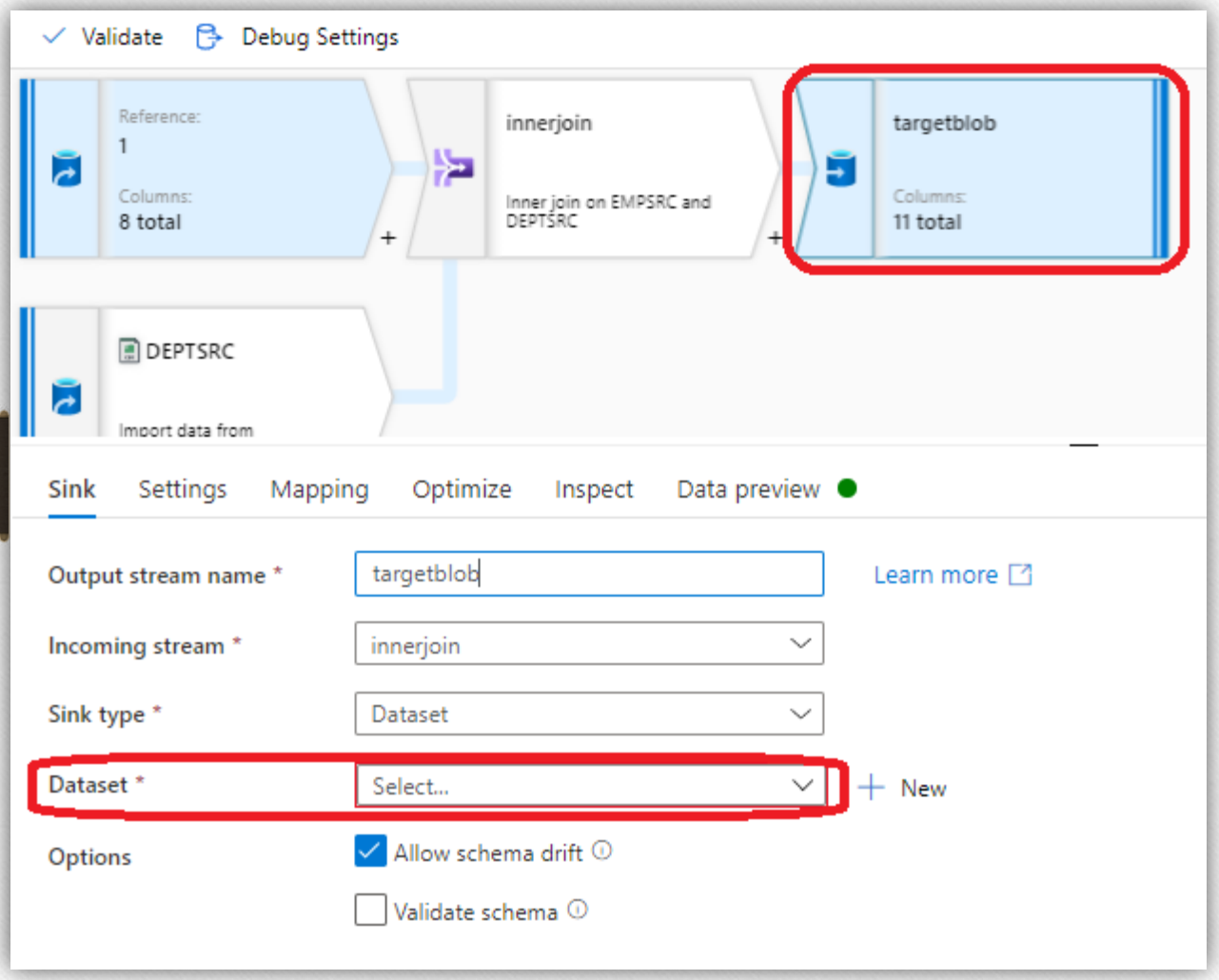
Select **left** And **Right stream** datasets and select join type as **inner join**. Select Join condition on left side and right side.



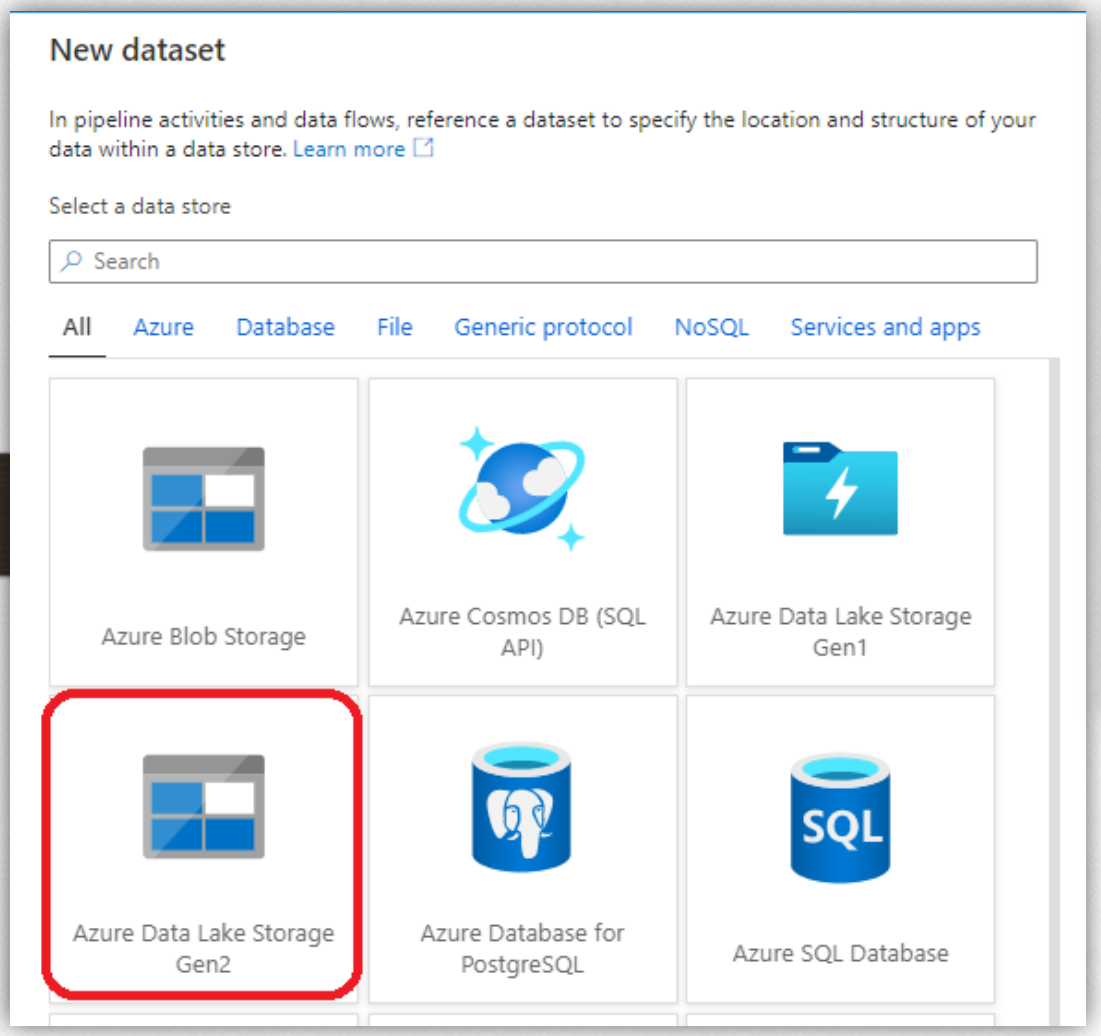
Click on + in Join to add target (sink)

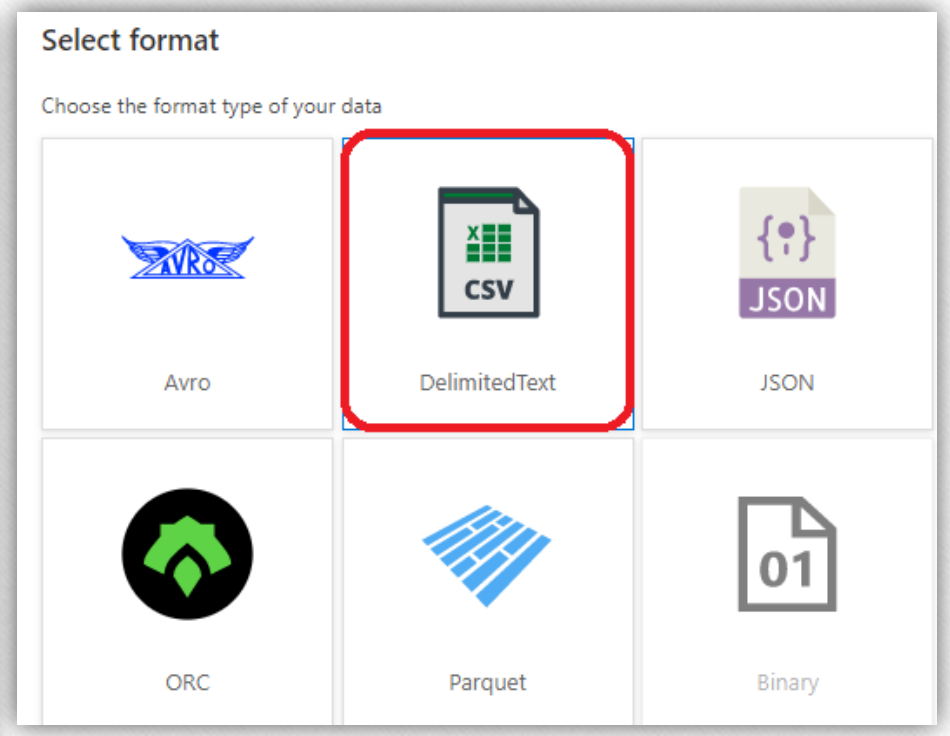


Select **sink** and create **new dataset** for target location which is **Azure Blog storage gen2**



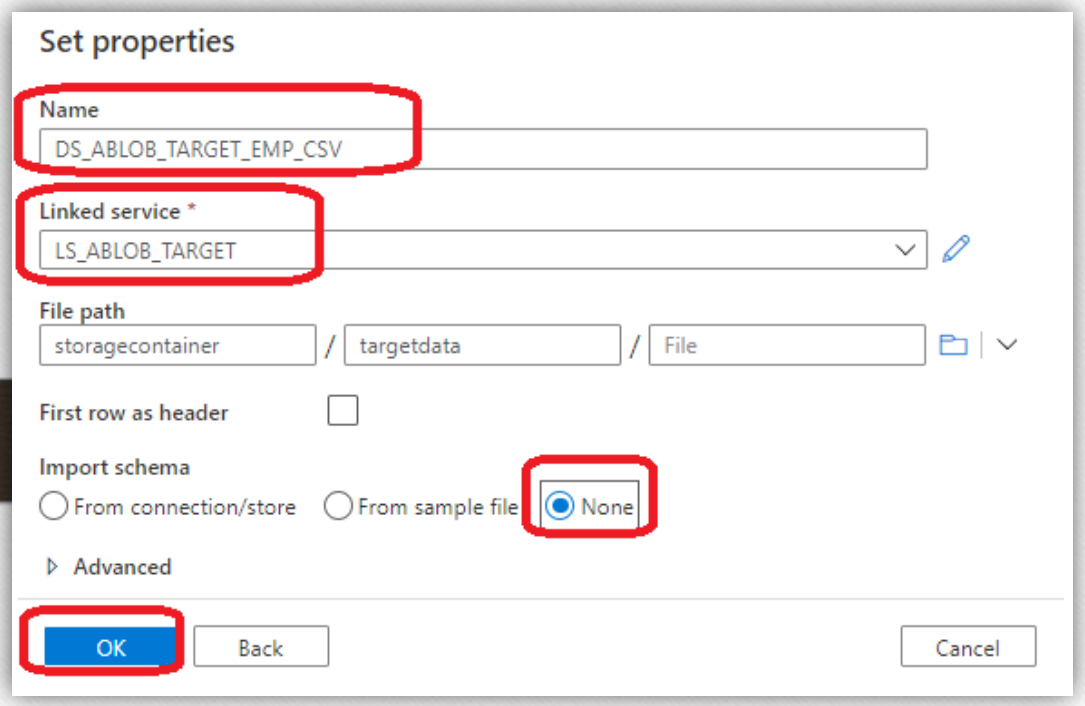
Select **sink dataset** – Target output file storage location as blob dataset.

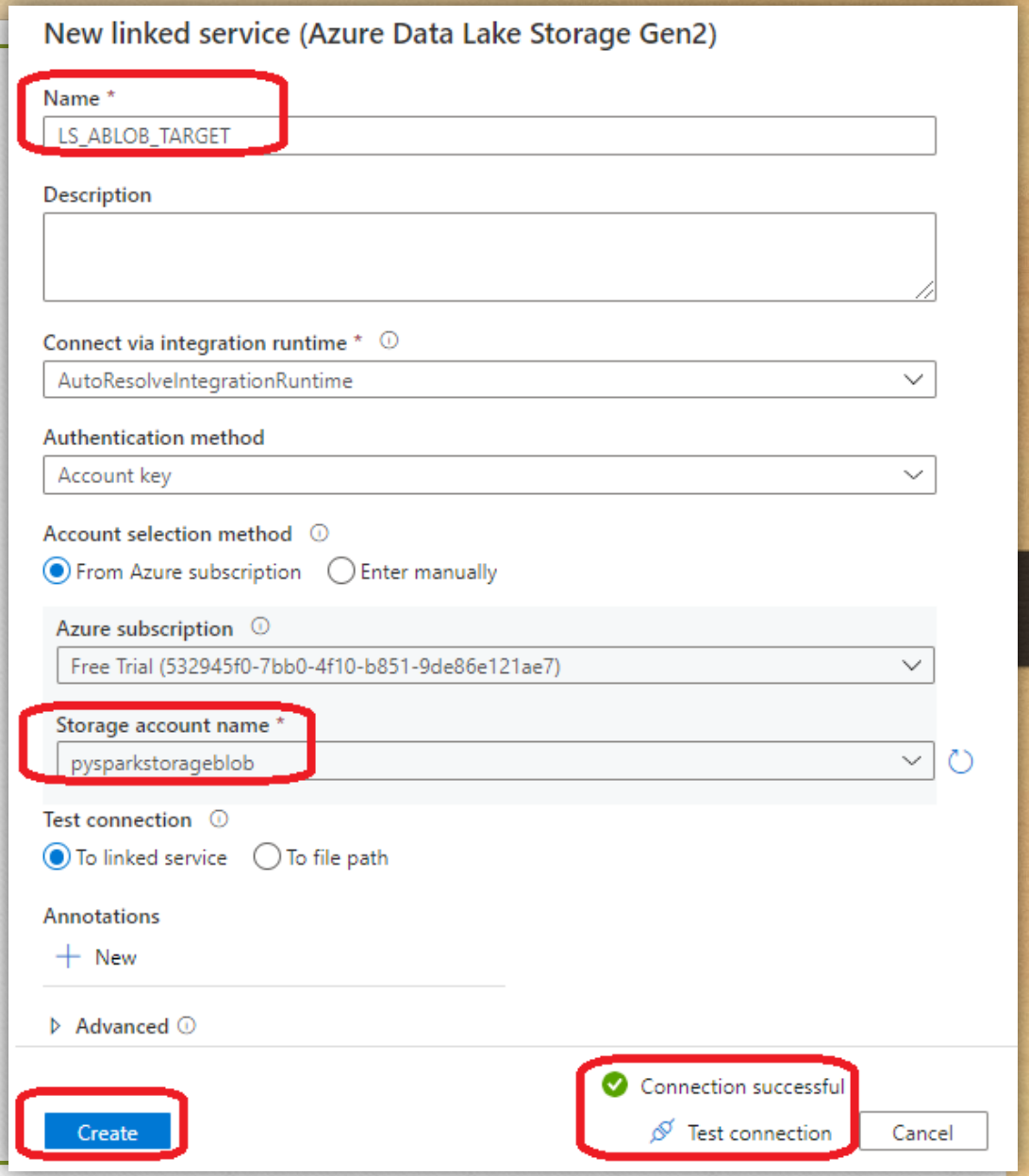




Creating Linked Service for Azure Data Lake Store gen2

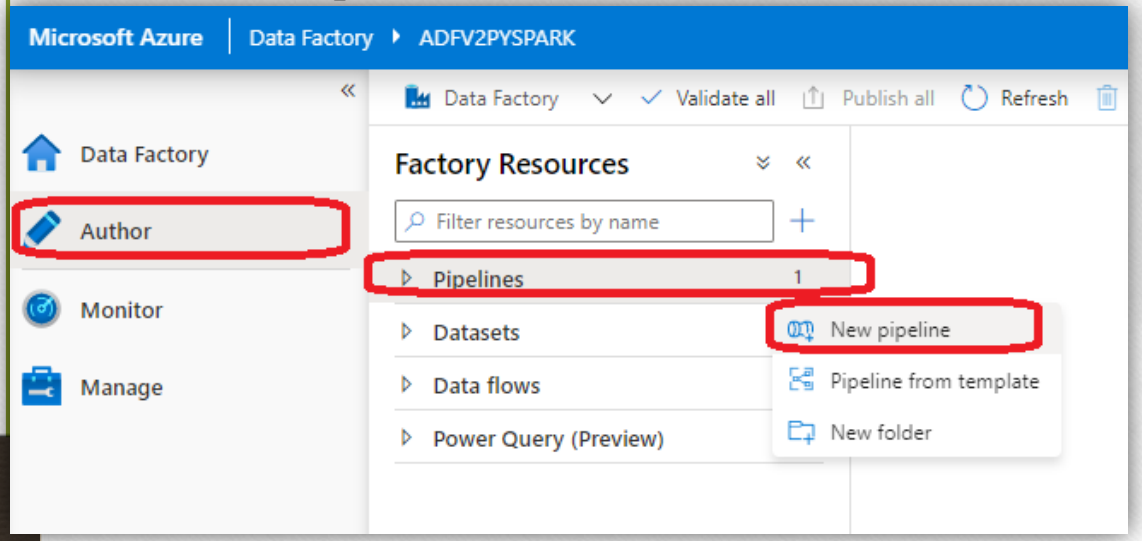
Creating Target Dataset for azure blob storage adls gen2.

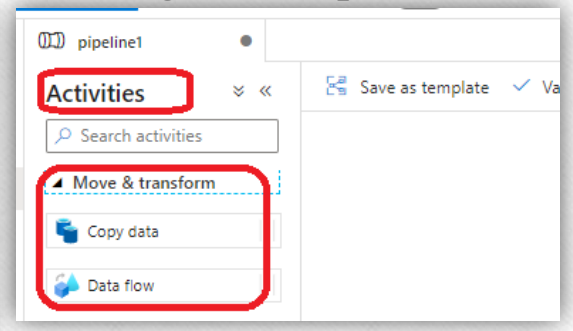




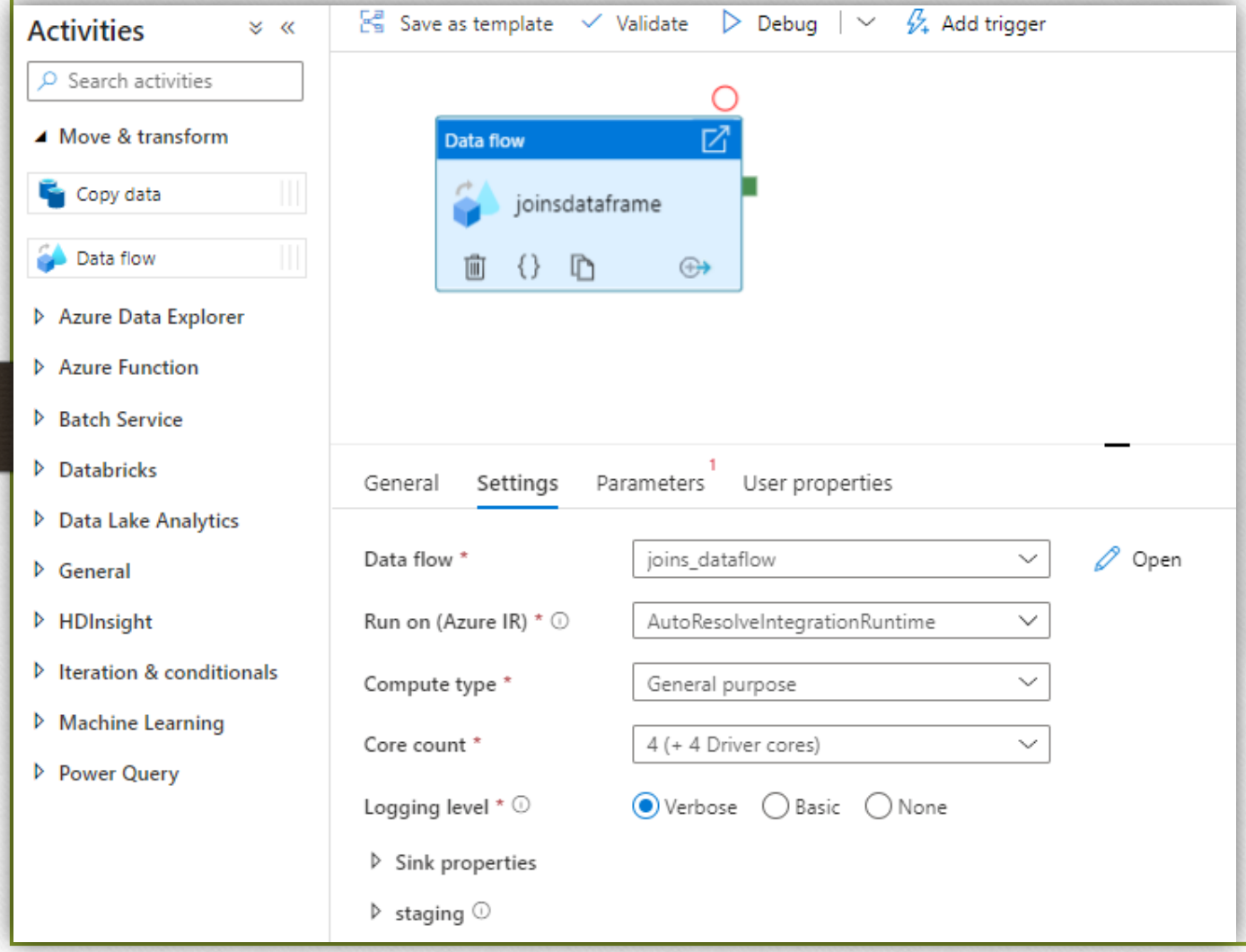
Creating New Pipeline executing data flow to transfer data from Source To Target

Create New Pipeline=> In Activities => Move Transform => drag and drop **Data Flow**

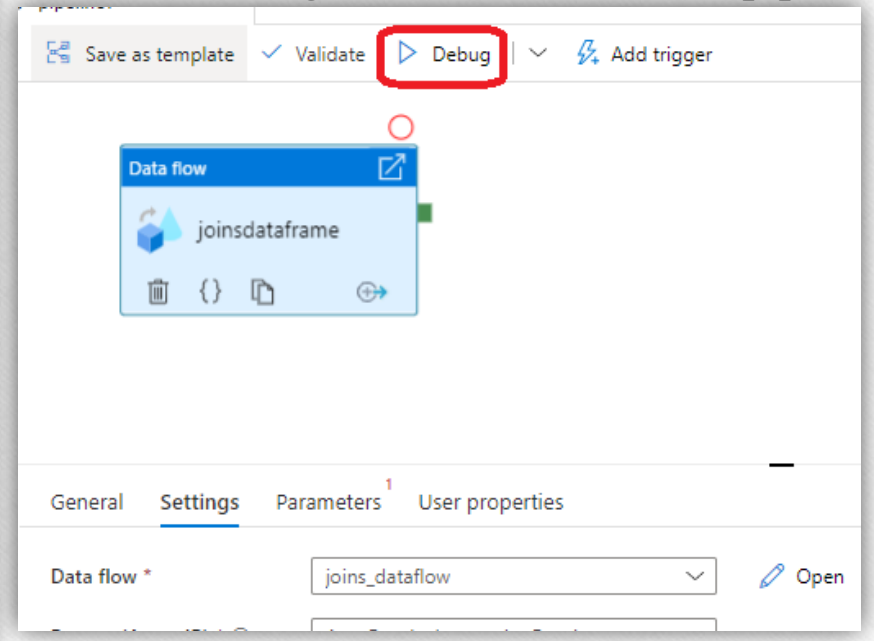




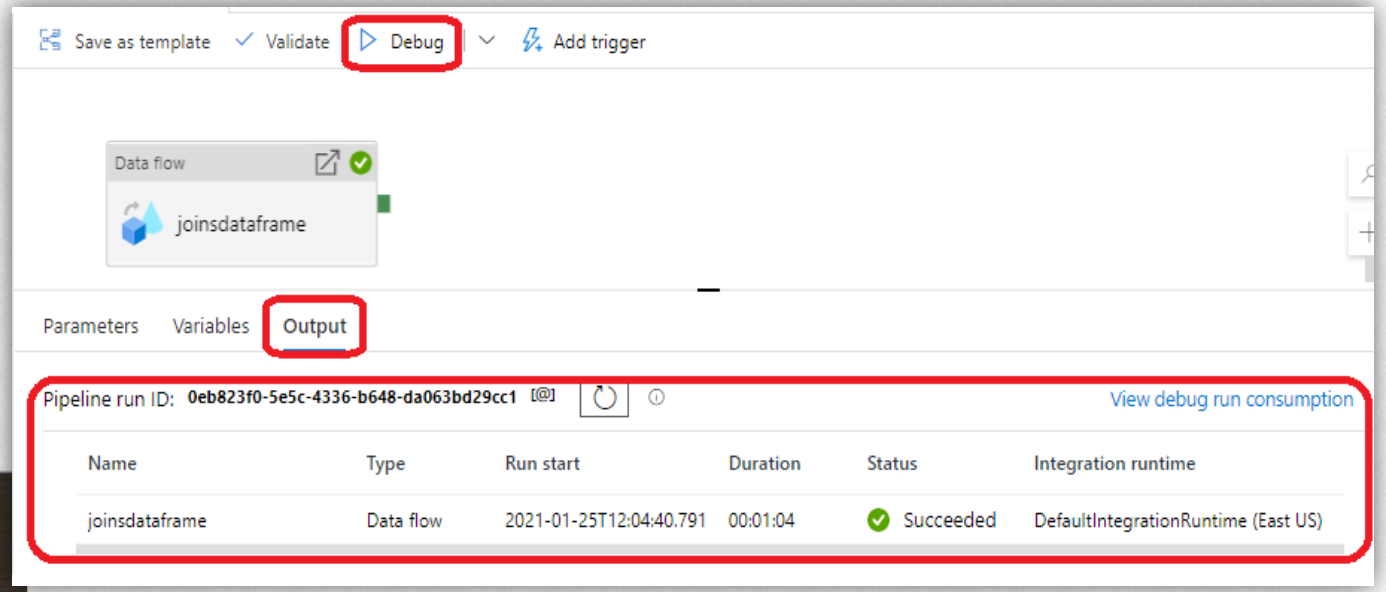
Rename Dataflow Activity name as joinsdataframe And Goto Settings - Select Joins_dataflow which we created recently



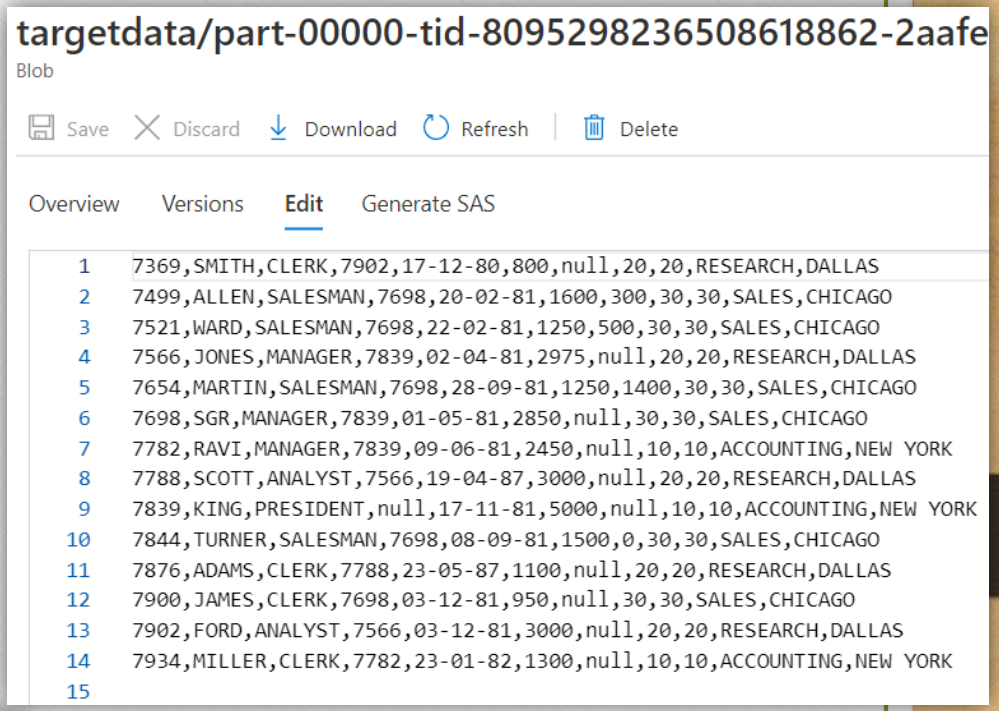
Click on debug button to execute the pipeline.



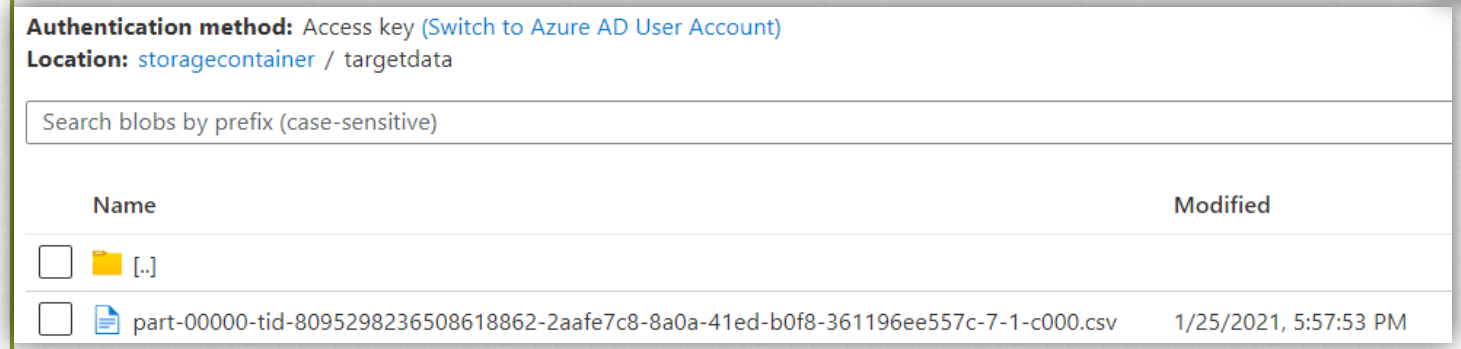
Click on Dubug option to run the pipe line with dataflow activity.



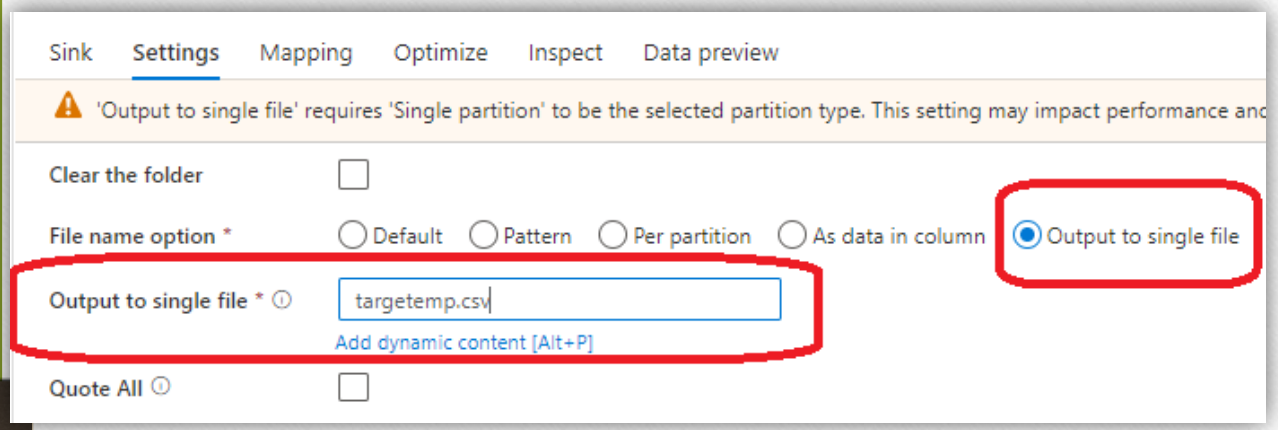
Target output file



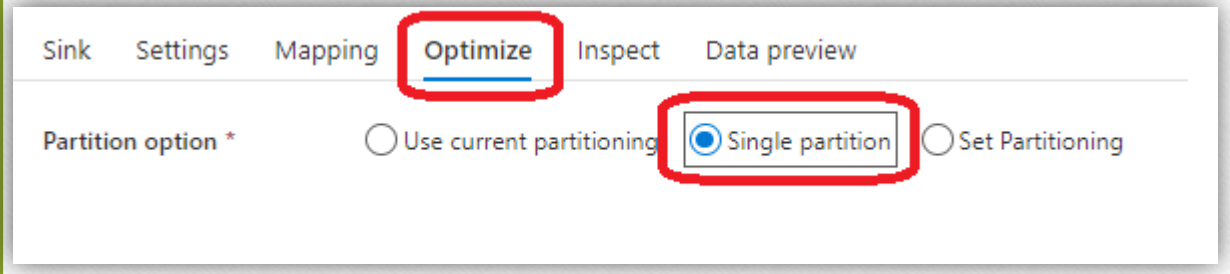
With Default option it will create part files.

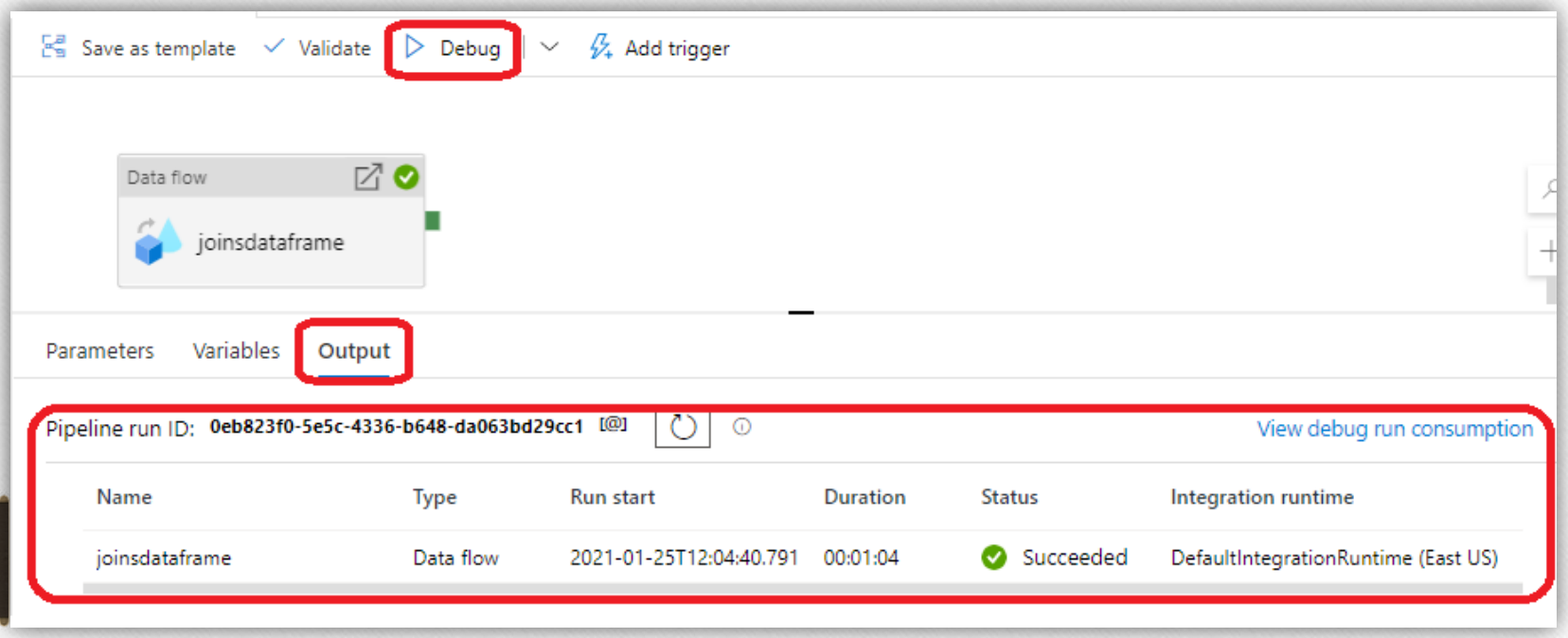


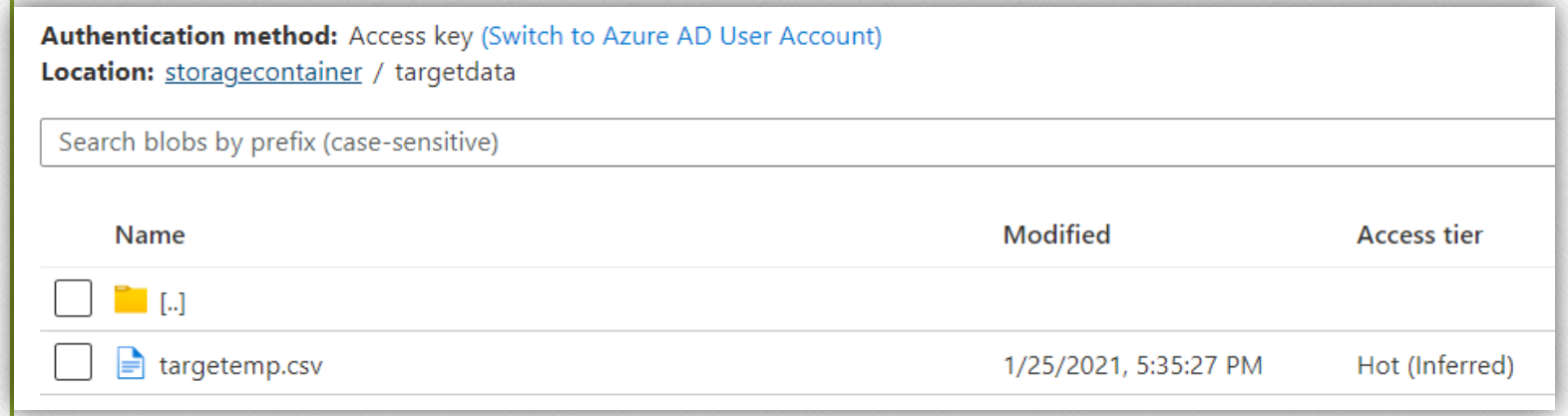
If we want store target file name as customized name. use **Output to single file** in **Settings** Then enter the file name In below **Output to single file as :** targetemp.csv



While creating single file choose **single partition** option in **optimize tab.**

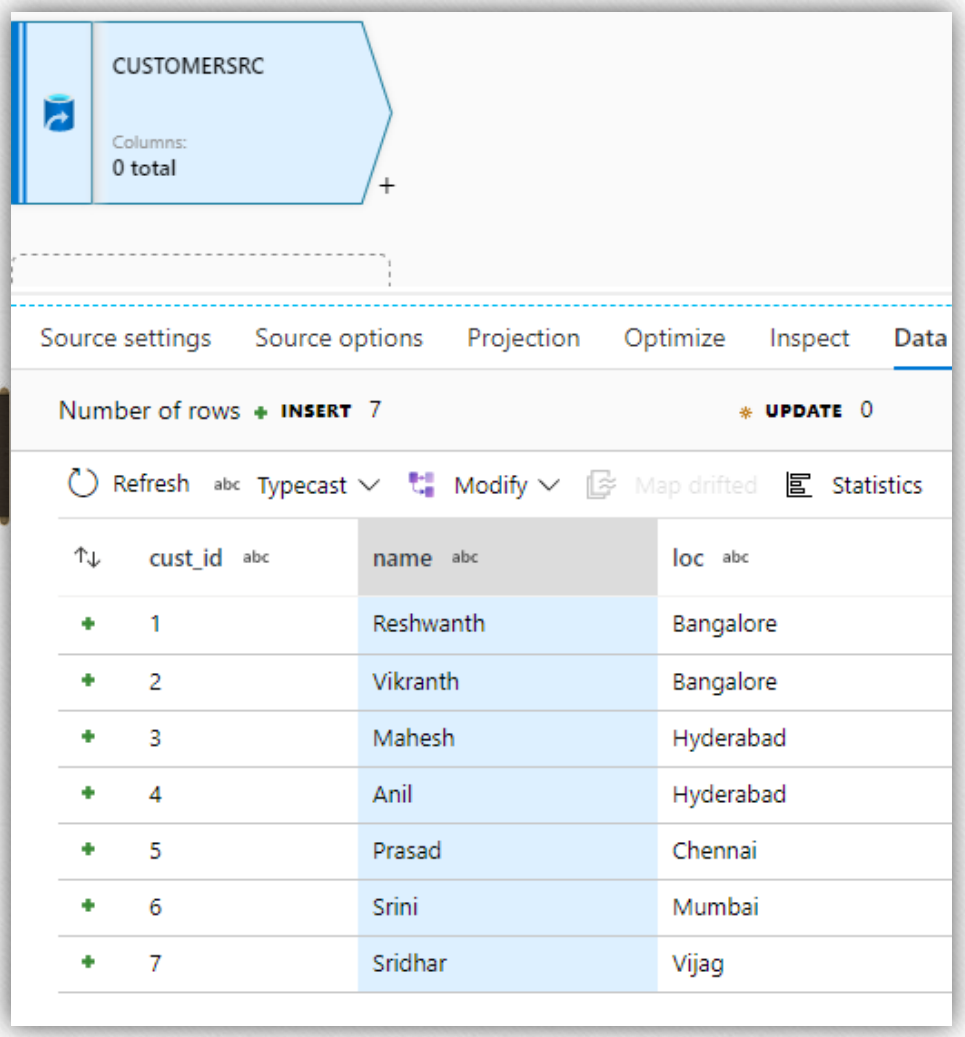



Save as template
Validate
Debug
Add trigger
Data flow
joinsdataframe
Parameters
Variables
Output
Pipeline run ID: 0eb823f0-5e5c-4336-b648-da063bd29cc1
View debug run consumption
Name
Type
Run start
Duration
Status
Integration runtime
joinsdataframe
Data flow
2021-01-25T12:04:40.791
00:01:04
Succeeded
DefaultIntegrationRuntime (East US)

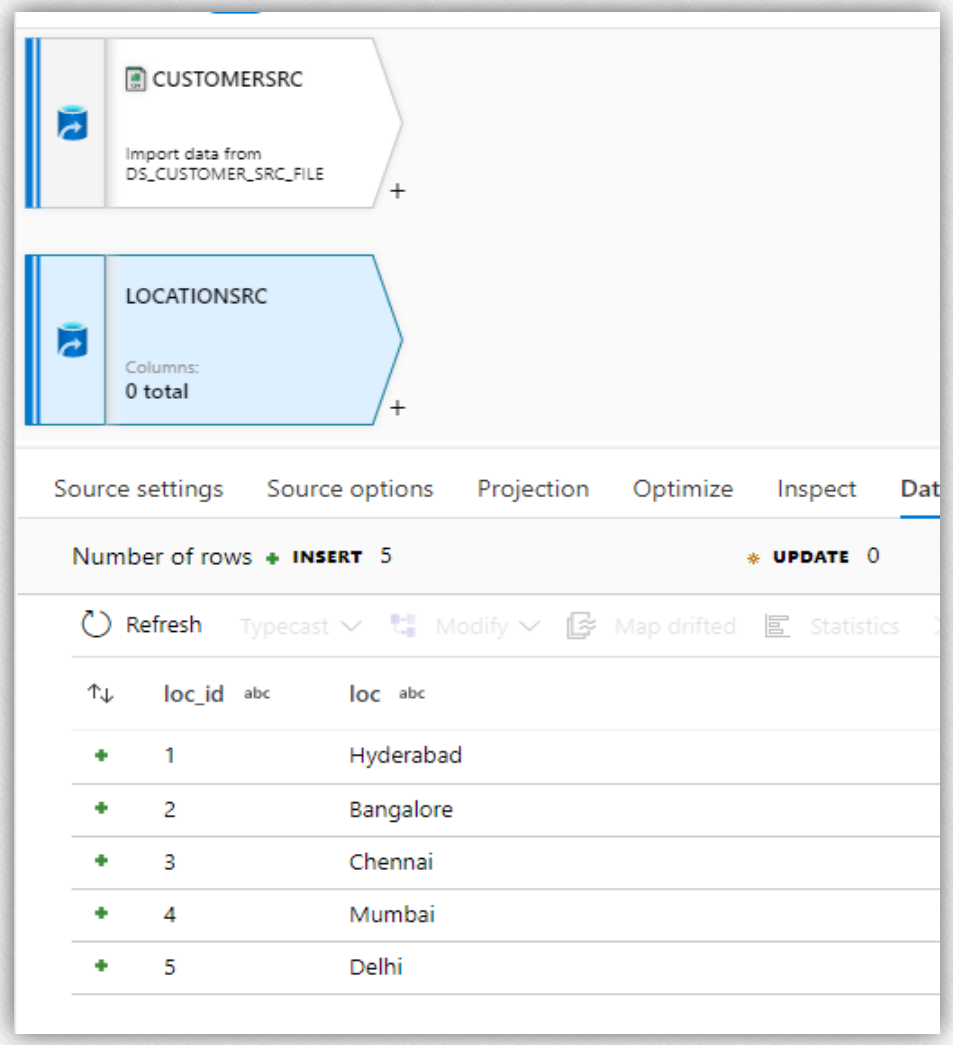

Authentication method: Access key (Switch to Azure AD User Account)
Location: storagecontainer / targetdata
Search blobs by prefix (case-sensitive)
Name
Modified
Access tier
[..]
targetemp.csv
1/25/2021, 5:35:27 PM
Hot (Inferred)

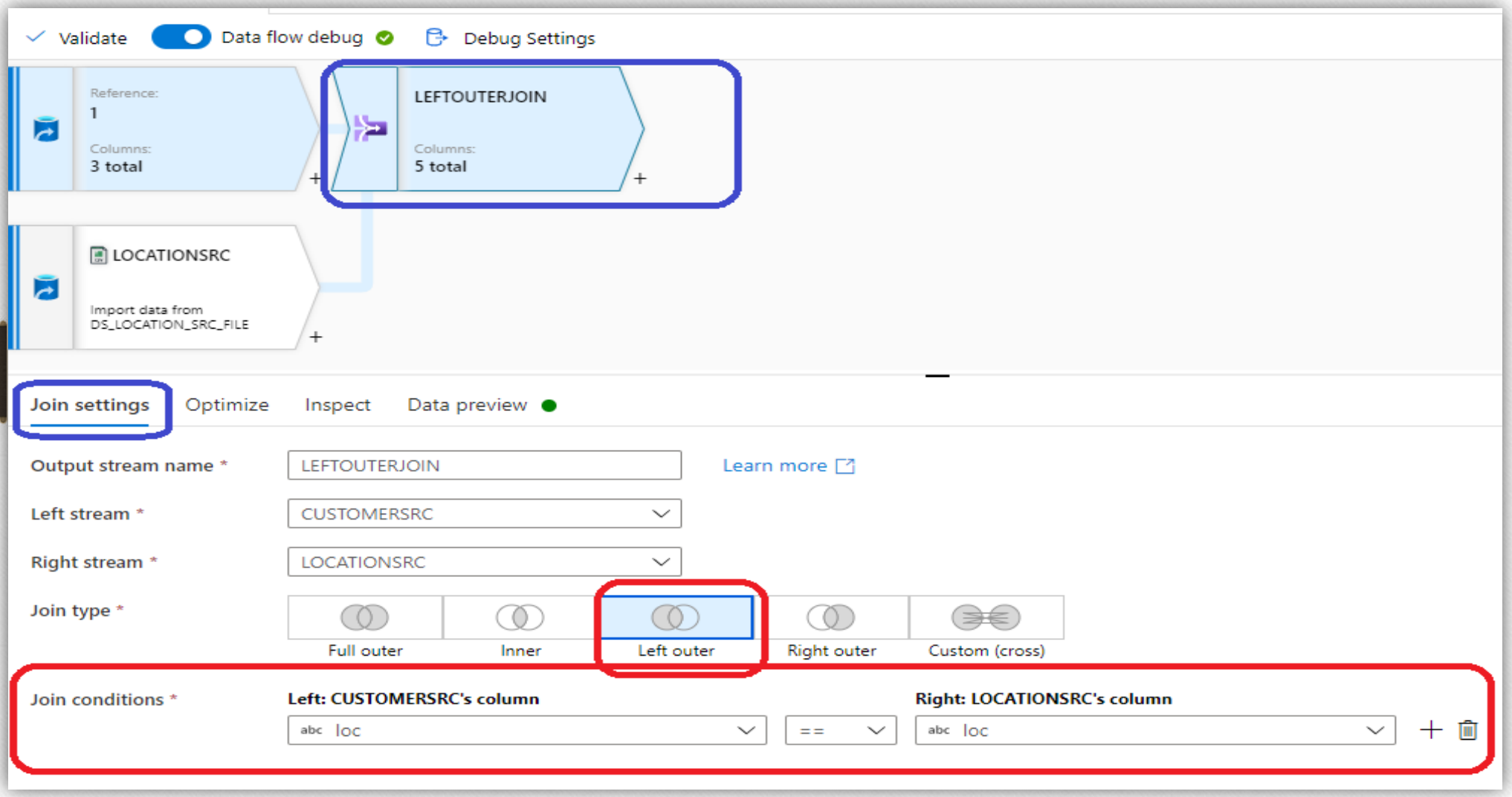
Creating Left Outer Join

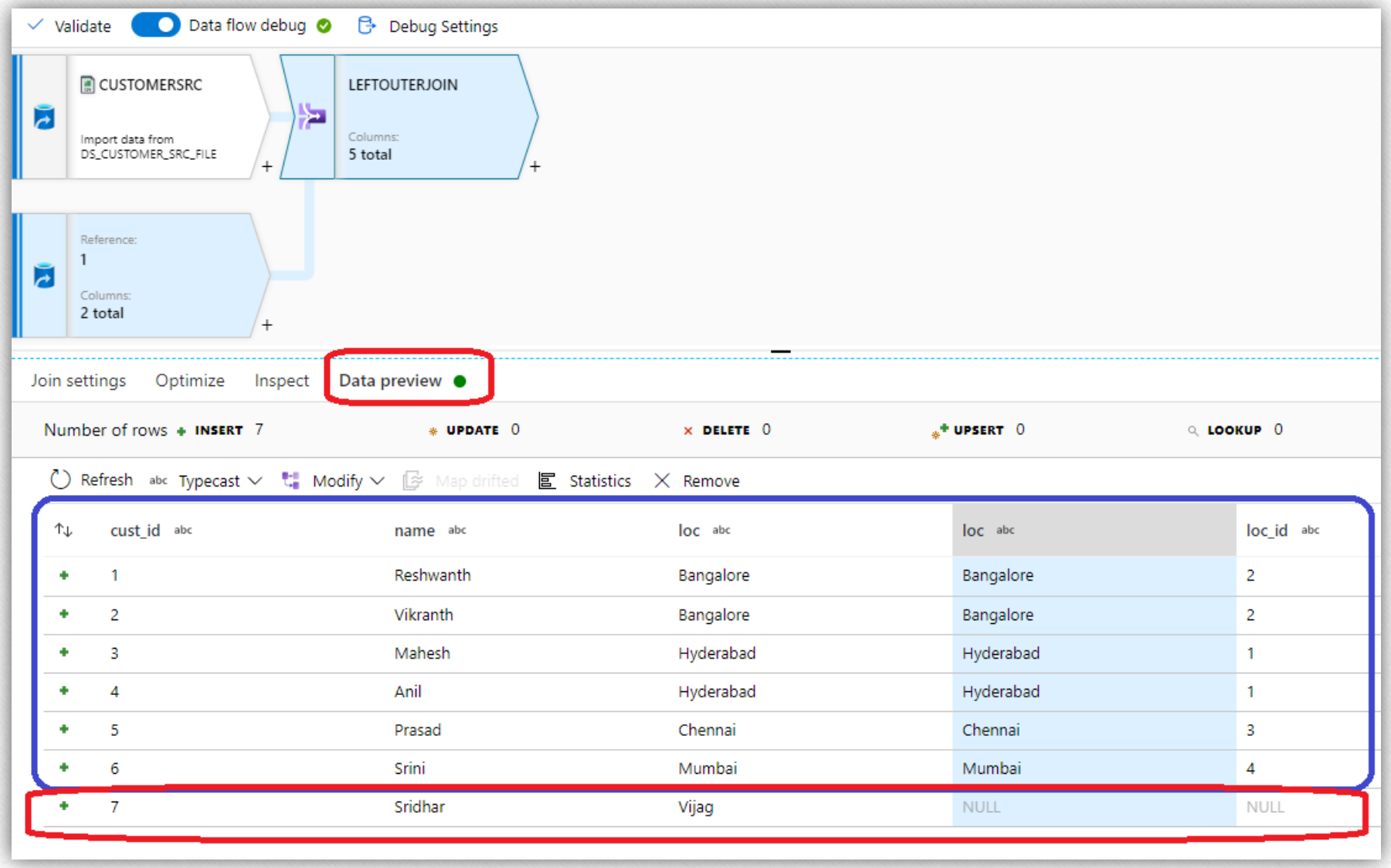
Left Table data

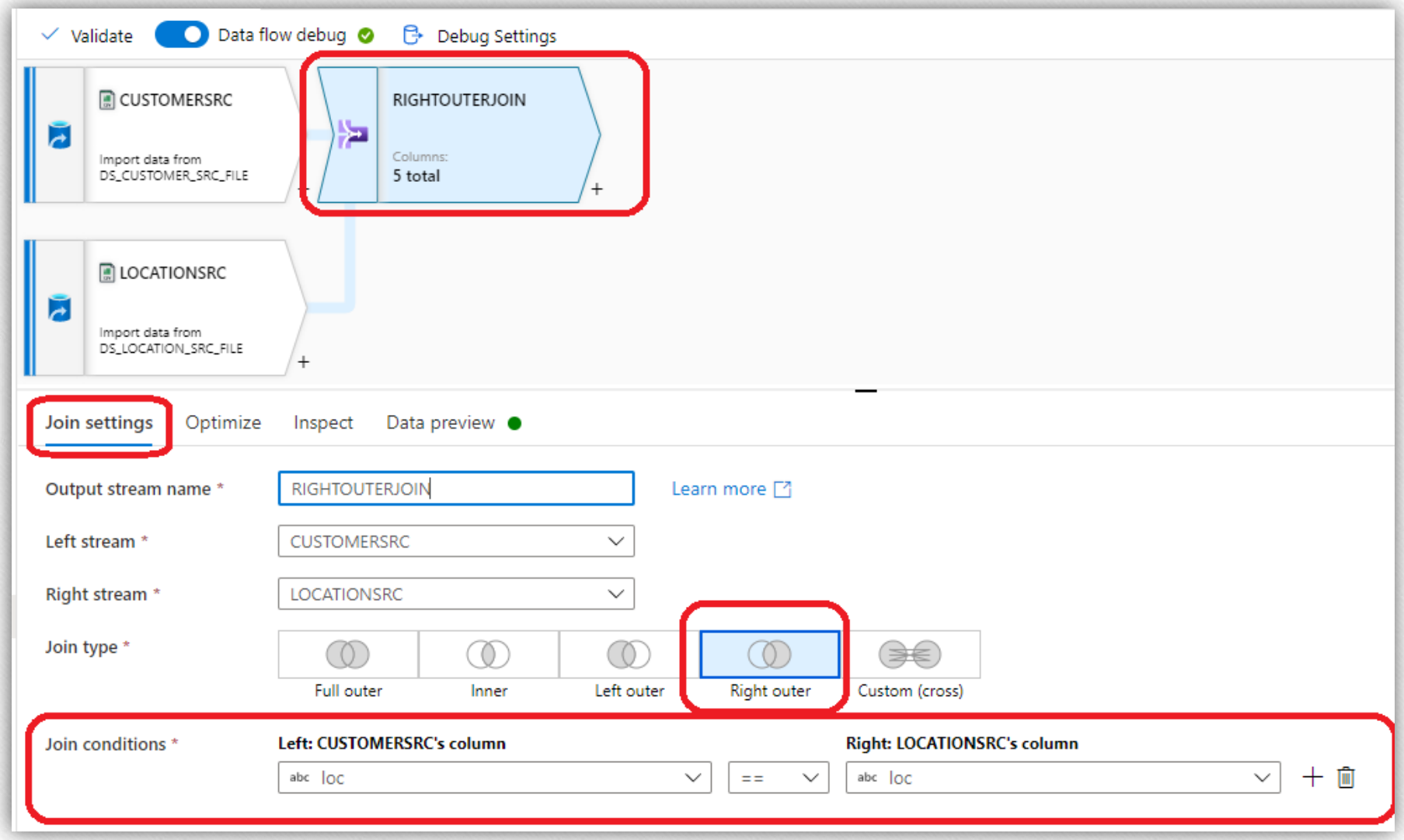


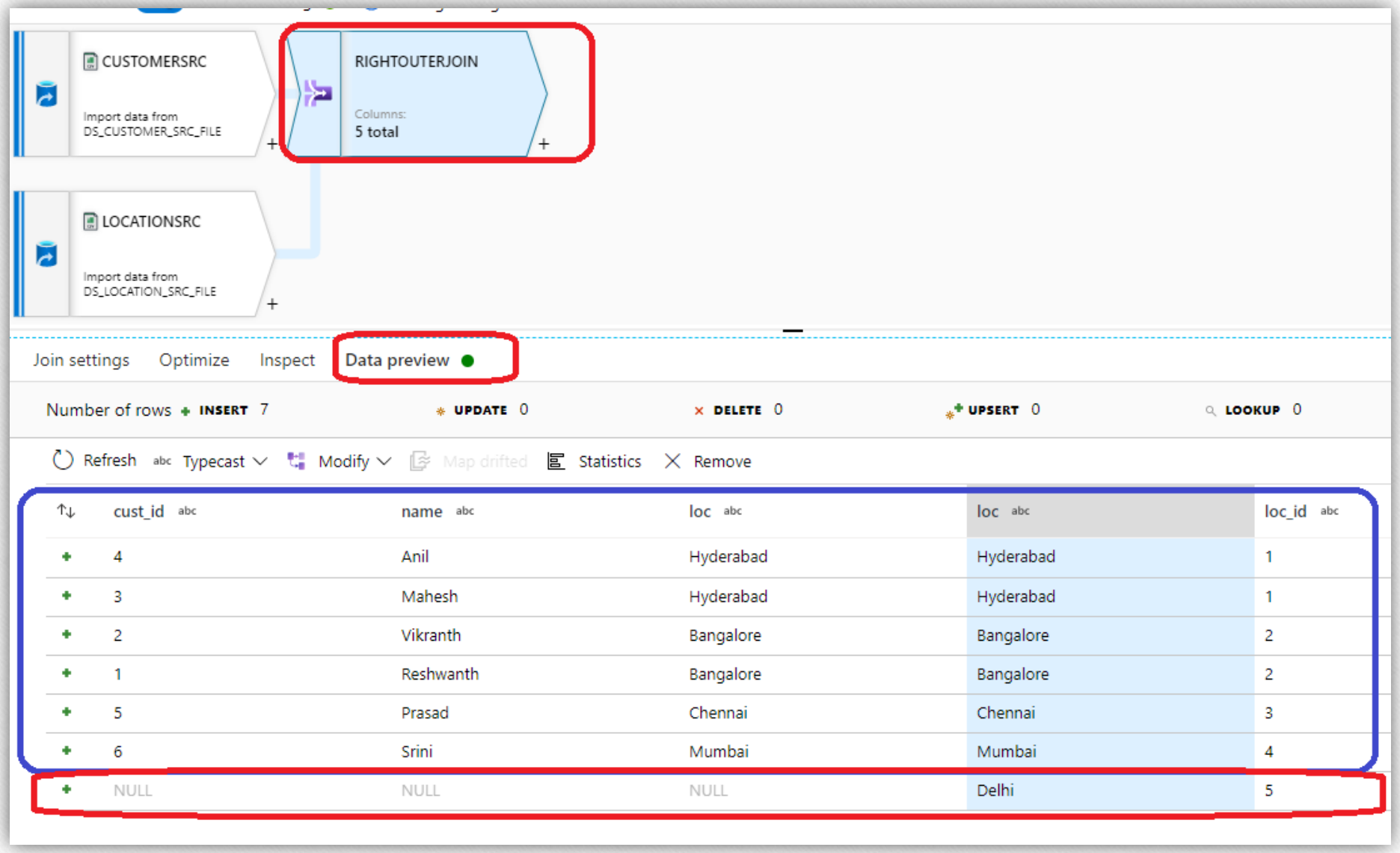
Right Table Data



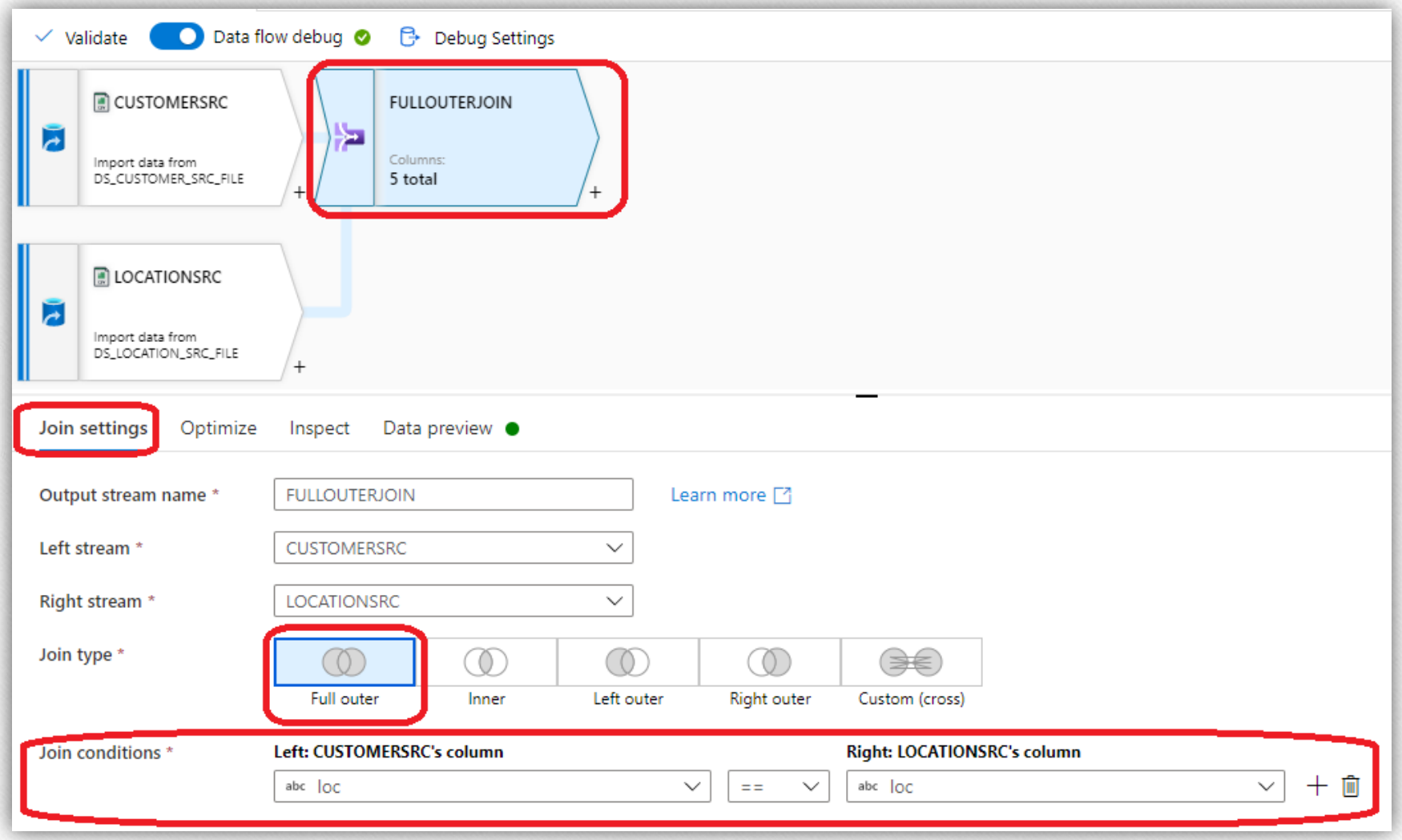

Validate
Data flow debug
Debug Settings
Reference:
1
Columns:
3 total
LEFTOUTERJOIN
Columns:
5 total
LOCATIONSRC
Import data from
DS_LOCATION_SRC_FILE
Join settings
Optimize
Inspect
Data preview
Output stream name *
LEFTOUTERJOIN
Learn more
Left stream *
CUSTOMERSRC
Right stream *
LOCATIONSRC
Join type *
Full outer
Inner
Left outer
Right outer
Custom (cross)
Join conditions *
Left: CUSTOMERSRC's column
abc loc
==
Right: LOCATIONSRC's column
abc loc

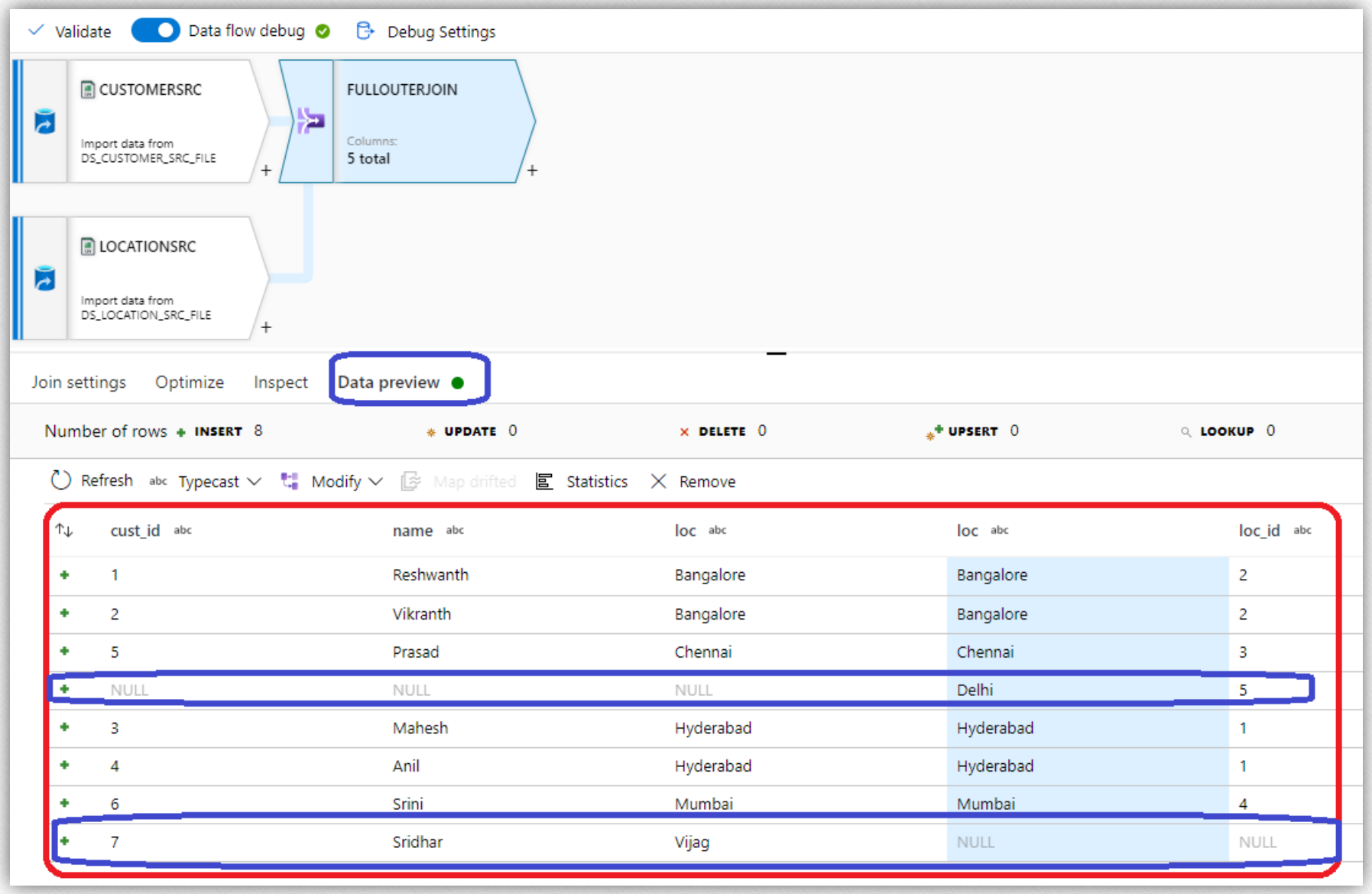

Validate
Data flow debug
Debug Settings
CUSTOMERSRC
Import data from DS_CUSTOMER_SRC_FILE
LEFTOUTERJOIN
Columns: 5 total
Reference: 1
Columns: 2 total
Join settings
Optimize
Inspect
Data preview
Number of rows INSERT 7
UPDATE 0
DELETE 0
UPSERT 0
LOOKUP 0
Refresh
Typecast
Modify
Map drifted
Statistics
Remove
cust_id abc
name abc
loc abc
loc abc
loc_id abc
1 Reshwanth Bangalore Bangalore 2
2 Vikranth Bangalore Bangalore 2
3 Mahesh Hyderabad Hyderabad 1
4 Anil Hyderabad Hyderabad 1
5 Prasad Chennai Chennai 3
6 Srini Mumbai Mumbai 4
7 Sridhar Vijag NULL NULL


Validate
Data flow debug
Debug Settings
CUSTOMERSRC
Import data from DS_CUSTOMER_SRC_FILE
RIGHTOUTERJOIN
Columns:
5 total
LOCATIONSRC
Import data from DS_LOCATION_SRC_FILE
Join settings
Optimize
Inspect
Data preview
Output stream name *
RIGHTOUTERJOIN
Learn more
Left stream *
CUSTOMERSRC
Right stream *
LOCATIONSRC
Join type *
Full outer
Inner
Left outer
Right outer
Custom (cross)
Join conditions *
Left: CUSTOMERSRC's column
Right: LOCATIONSRC's column
abc loc
==
abc loc

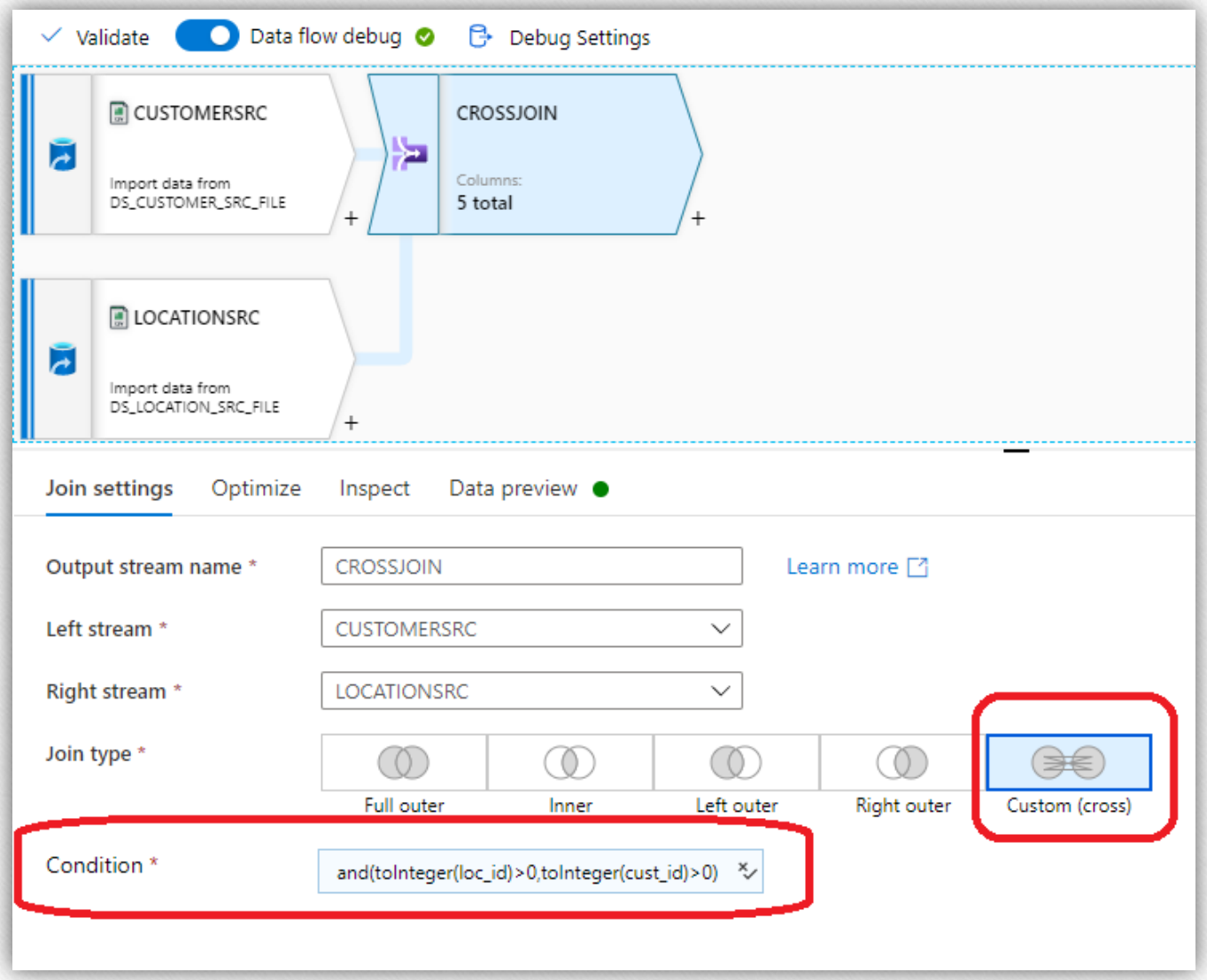


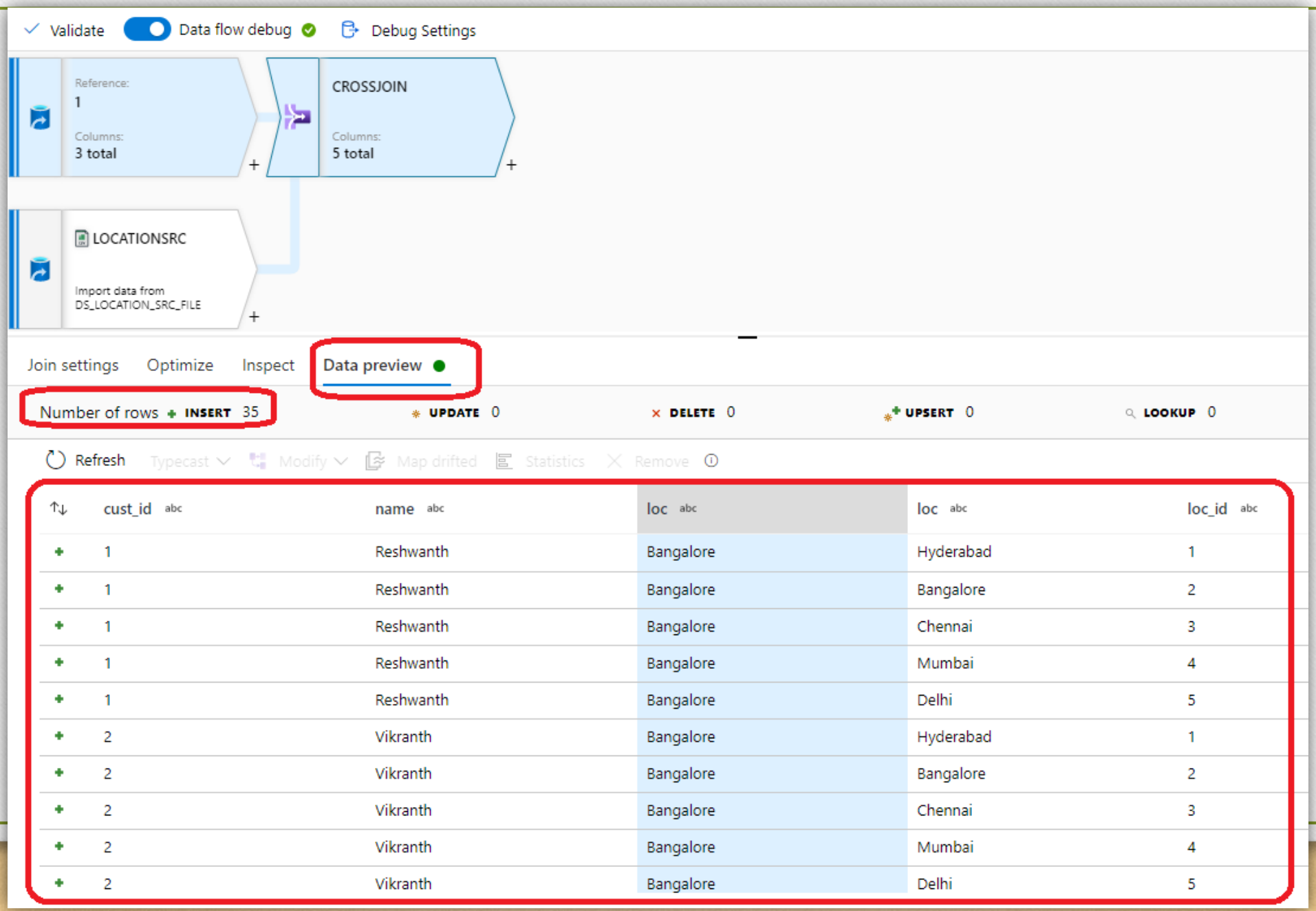
cust_id	name	loc	loc	loc_id
4	Anil	Hyderabad	Hyderabad	1
3	Mahesh	Hyderabad	Hyderabad	1
2	Vikranth	Bangalore	Bangalore	2
1	Reshwanth	Bangalore	Bangalore	2
5	Prasad	Chennai	Chennai	3
6	Srini	Mumbai	Mumbai	4
NULL	NULL	NULL	Delhi	5


Validate
Data flow debug
Debug Settings
CUSTOMERSRC
Import data from
DS_CUSTOMER_SRC_FILE
FULLOUTERJOIN
Columns:
5 total
LOCATIONSRC
Import data from
DS_LOCATION_SRC_FILE
Join settings
Optimize
Inspect
Data preview
Output stream name *
FULLOUTERJOIN
Learn more
Left stream *
CUSTOMERSRC
Right stream *
LOCATIONSRC
Join type *
Full outer
Inner
Left outer
Right outer
Custom (cross)
Join conditions *
Left: CUSTOMERSRC's column
Right: LOCATIONSRC's column
abc loc
==
abc loc



cust_id	name	loc	loc	loc_id
1	Reshwanth	Bangalore	Bangalore	2
2	Vikranth	Bangalore	Bangalore	2
5	Prasad	Chennai	Chennai	3
NULL	NULL	NULL	Delhi	5
3	Mahesh	Hyderabad	Hyderabad	1
4	Anil	Hyderabad	Hyderabad	1
6	Srini	Mumbai	Mumbai	4
7	Sridhar	Vijag	NULL	NULL


Validate
Data flow debug
Debug Settings
CUSTOMERSRC
Import data from
DS_CUSTOMER_SRC_FILE
CROSSJOIN
Columns:
5 total
LOCATIONSRC
Import data from
DS_LOCATION_SRC_FILE
Join settings
Optimize
Inspect
Data preview
Output stream name *
CROSSJOIN
Learn more
Left stream *
CUSTOMERSRC
Right stream *
LOCATIONSRC
Join type *
Full outer
Inner
Left outer
Right outer
Custom (cross)
Condition *
and(toInteger(loc_id)>0,toInteger(cust_id)>0)


Validate
Data flow debug
Debug Settings
Reference:
1
Columns:
3 total
CROSSJOIN
Columns:
5 total
LOCATIONSRC
Import data from
DS_LOCATION_SRC_FILE
Join settings
Optimize
Inspect
Data preview
Number of rows INSERT 35
UPDATE 0
DELETE 0
UPSERT 0
LOOKUP 0
Refresh
Typecast
Modify
Map drifted
Statistics
Remove
cust_id abc
name abc
loc abc
loc abc
loc_id abc
1 Reshwanth Bangalore Hyderabad 1
1 Reshwanth Bangalore Bangalore 2
1 Reshwanth Bangalore Chennai 3
1 Reshwanth Bangalore Mumbai 4
1 Reshwanth Bangalore Delhi 5
2 Vikranth Bangalore Hyderabad 1
2 Vikranth Bangalore Bangalore 2
2 Vikranth Bangalore Chennai 3
2 Vikranth Bangalore Mumbai 4
2 Vikranth Bangalore Delhi 5

Window Functions

The Window transformation is where you will define window-based aggregations of columns in your data streams. In the Expression Builder, you can define different types of aggregations that are based on data or time windows (aka SQL OVER clause) such as LEAD, LAG, NTILE, CUMEDIST, RANK, etc ...) and create a new field in your output that includes these aggregations with optional group-by fields.

Over

This where you will set the partitioning of column data for your window transformation. This is equivalent to the Partition By in the Over clause in SQL. If you wish to create a calculation or create an expression to use for the partitioning, you can do that by hovering over the column name and select "computed column".

Sort

Also part of the Over clause is setting the Order By, or sort order. As is the case with the "Over" column selector, you can also create an expression for a calculate value in this column field for sorting.

Range By

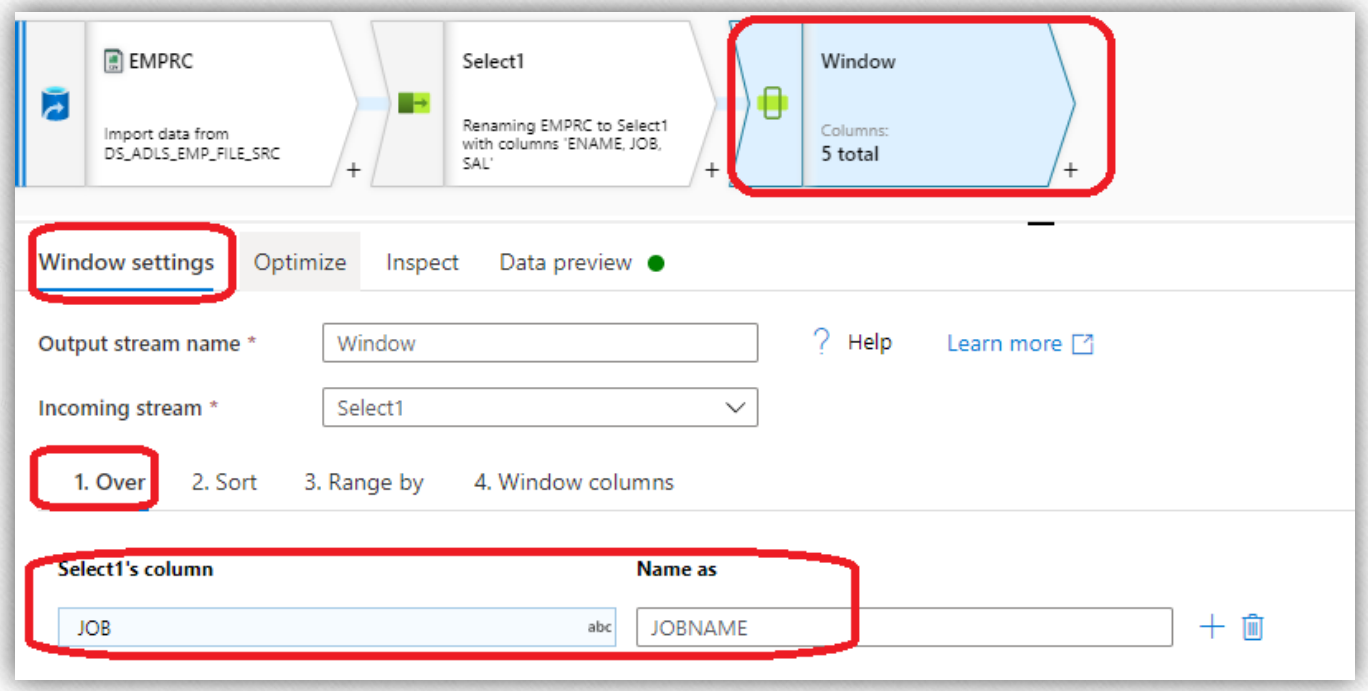
Next, set the window frame as Unbounded or Bounded. To set an unbounded window frame, set the slider to Unbounded on both ends. If you choose a setting between Unbounded and Current Row, then you must set the Offset start and end values. Both values will be positive integers. You can use either relative numbers or values from your data.

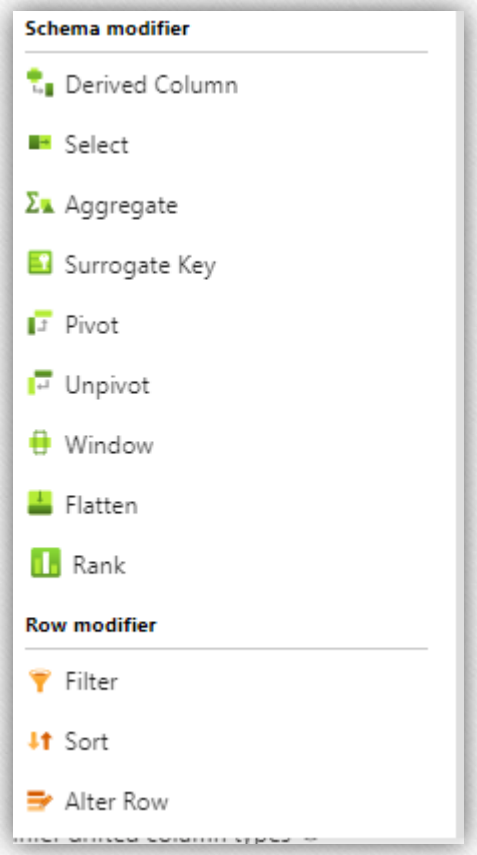
The window slider has 2 values to set: the values before the current row and the values after the current row. The Start and End offset matches the 2 selectors on the slider.

Window Functions

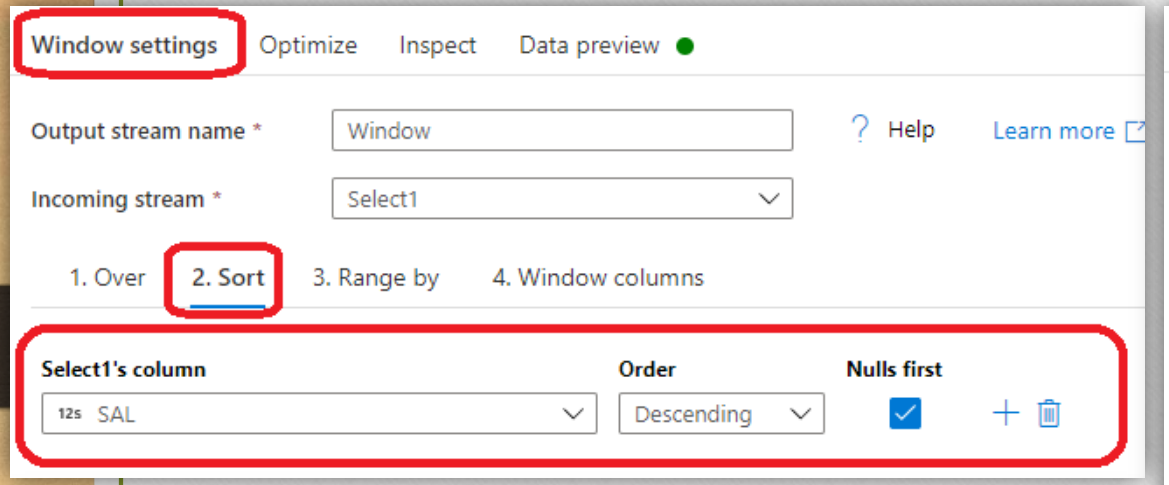
Window Columns

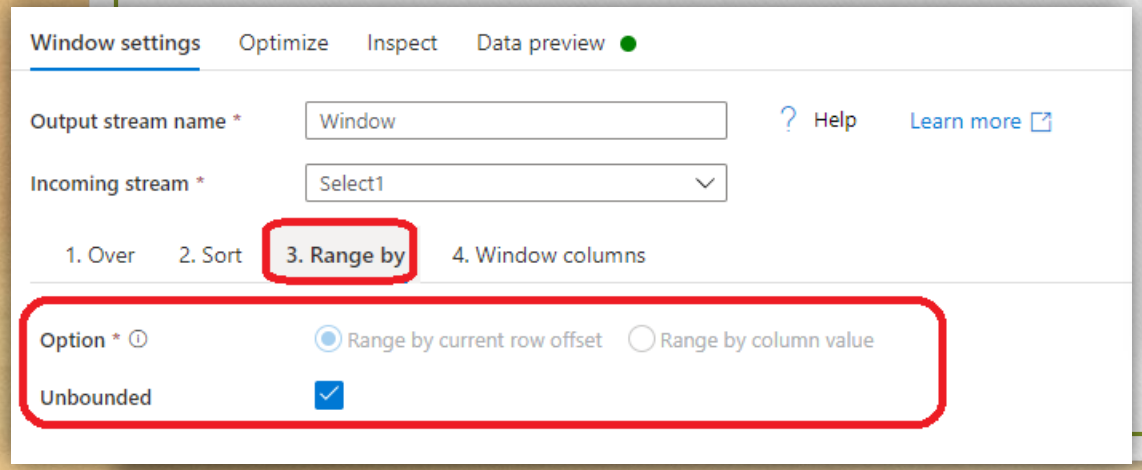
Lastly, use the Expression Builder to define the aggregations you wish to use with the data windows such as RANK, COUNT, MIN, MAX, DENSE RANK, LEAD, LAG, etc.

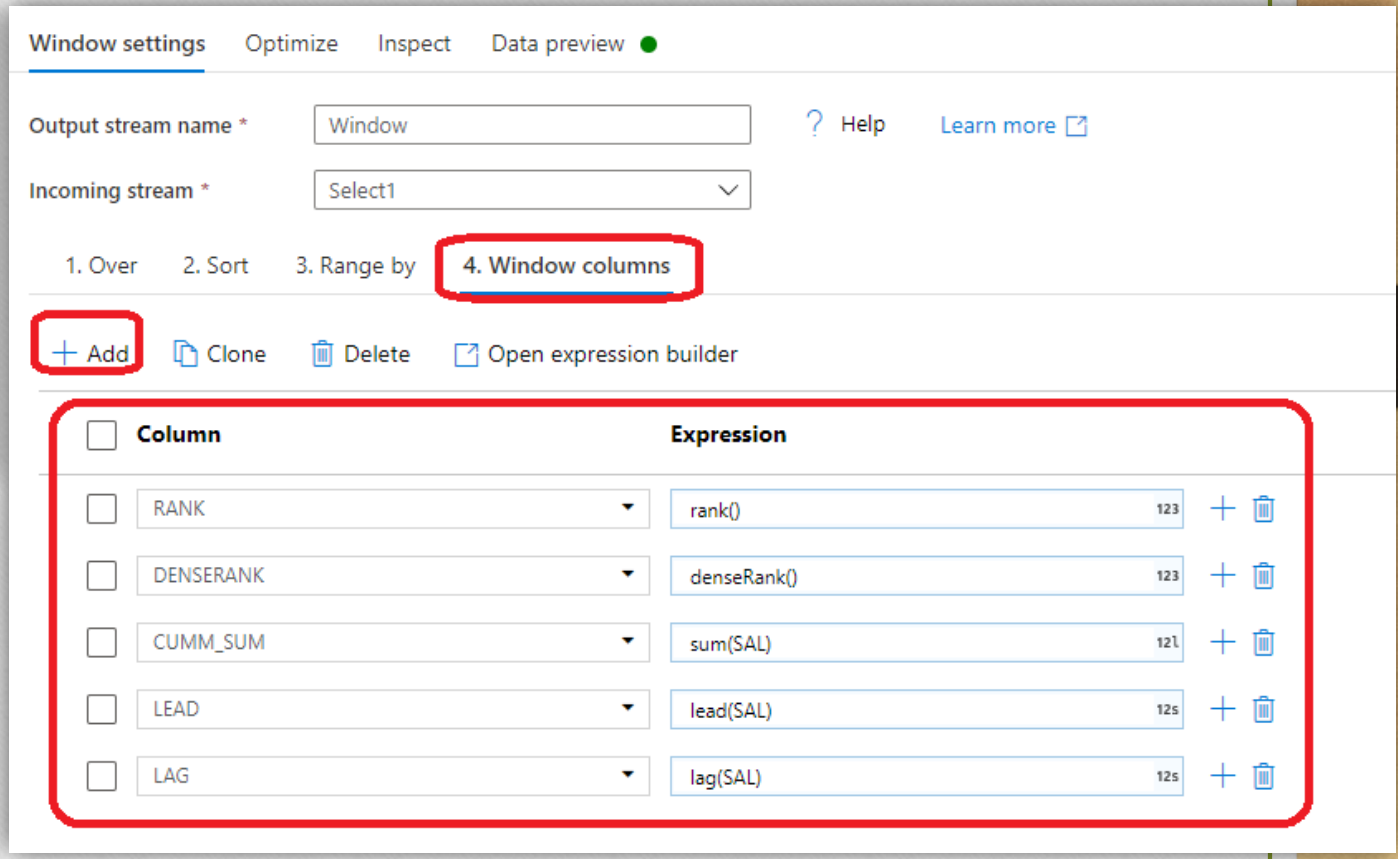


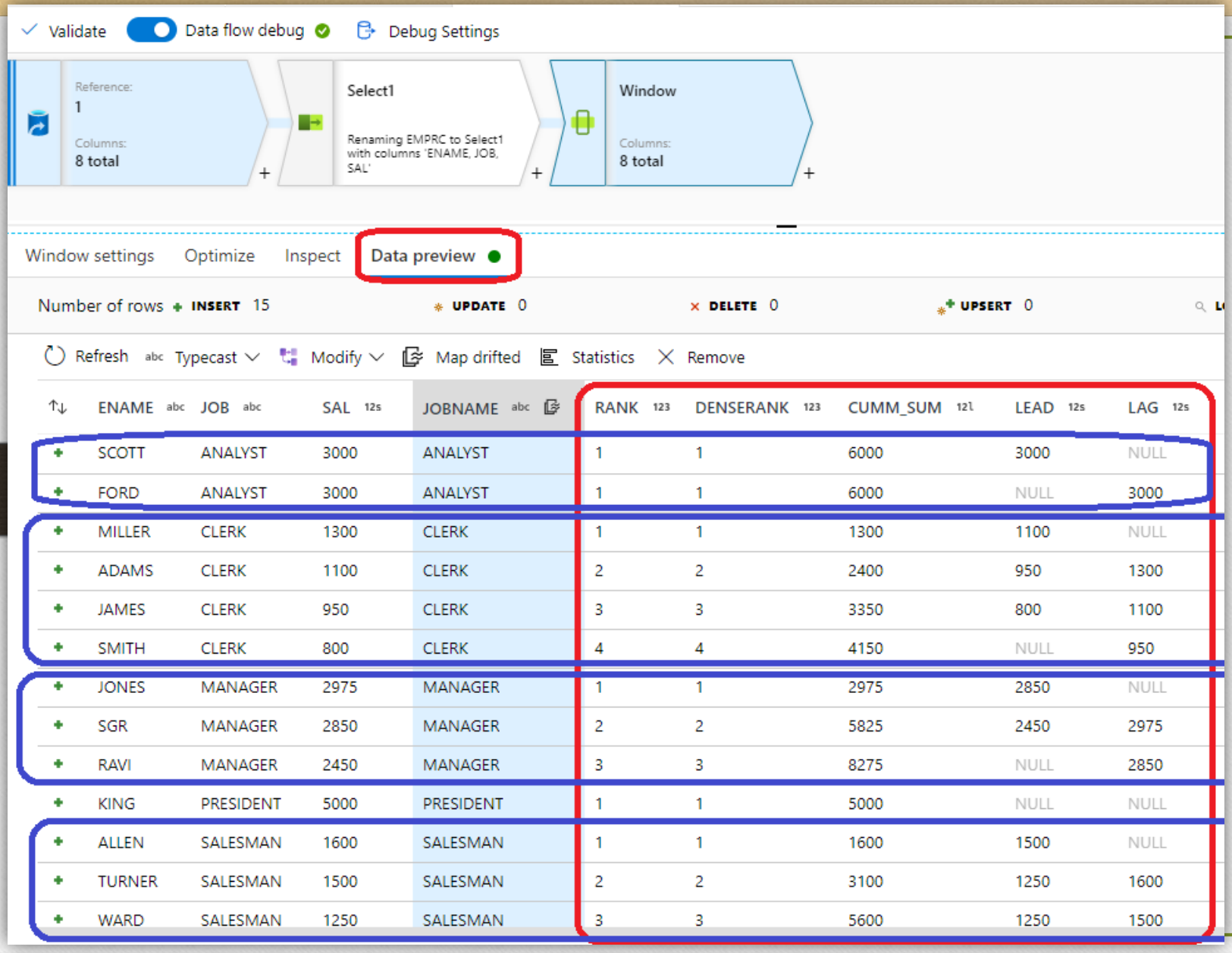


Window Functions









ENAME	JOB	SAL	JOBNAME	RANK	DENSERANK	CUMM_SUM	LEAD	LAG
SCOTT	ANALYST	3000	ANALYST	1	1	6000	3000	NULL
FORD	ANALYST	3000	ANALYST	1	1	6000	NULL	3000
MILLER	CLERK	1300	CLERK	1	1	1300	1100	NULL
ADAMS	CLERK	1100	CLERK	2	2	2400	950	1300
JAMES	CLERK	950	CLERK	3	3	3350	800	1100
SMITH	CLERK	800	CLERK	4	4	4150	NULL	950
JONES	MANAGER	2975	MANAGER	1	1	2975	2850	NULL
SGR	MANAGER	2850	MANAGER	2	2	5825	2450	2975
RAVI	MANAGER	2450	MANAGER	3	3	8275	NULL	2850
KING	PRESIDENT	5000	PRESIDENT	1	1	5000	NULL	NULL
ALLEN	SALESMAN	1600	SALESMAN	1	1	1600	1500	NULL
TURNER	SALESMAN	1500	SALESMAN	2	2	3100	1250	1600
WARD	SALESMAN	1250	SALESMAN	3	3	5600	1250	1500

Aggregate

The **Aggregate transformation** is where you will define aggregations of columns in your data streams. In the Expression Builder, you can define different types of aggregations (i.e. SUM, MIN, MAX, COUNT, etc ...) and create a new field in your output that includes these aggregations with optional group-by fields.

NOTE: Aggregate transforms will only output the columns used in the aggregation. I.e. only the fields used in group-by and the aggregated fields will be passed on to the next transformation in your data flow. If you wish to include the previous columns in your flow, use a New Branch from the previous step and use the self-join pattern to connect the flow with the original metadata.

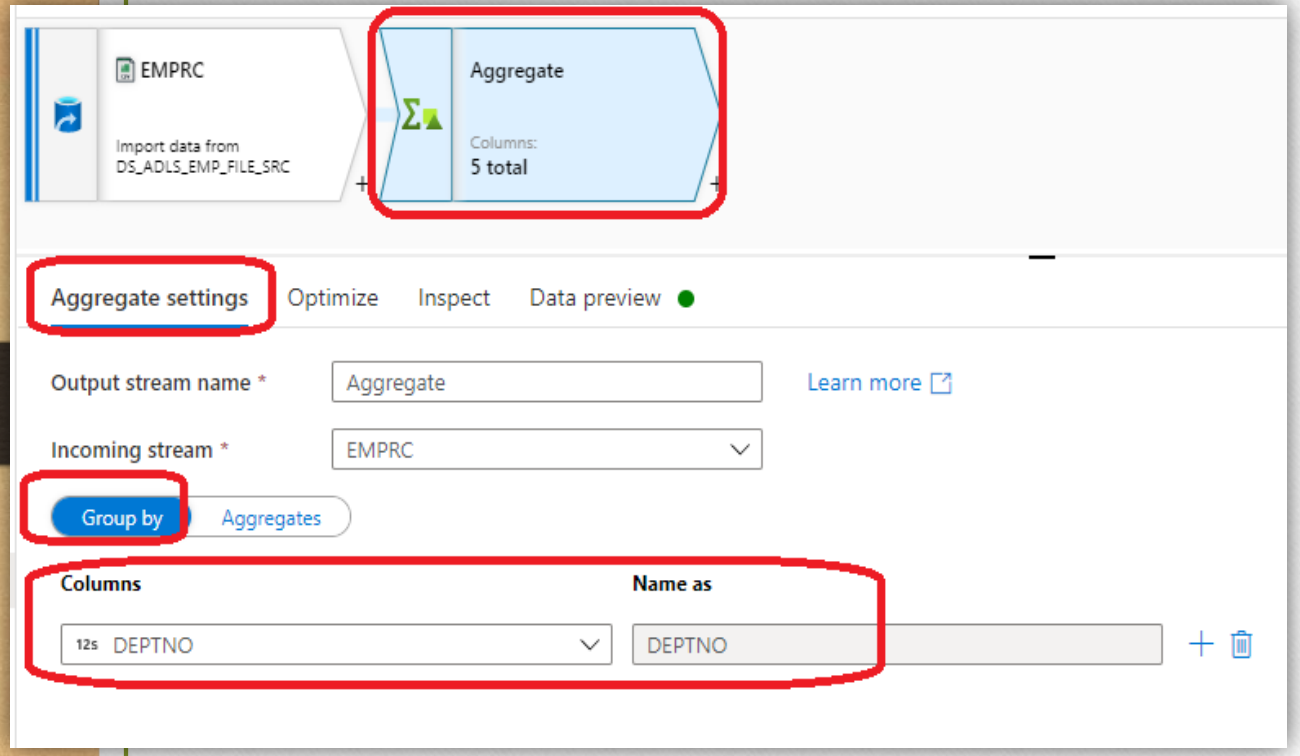
Group By

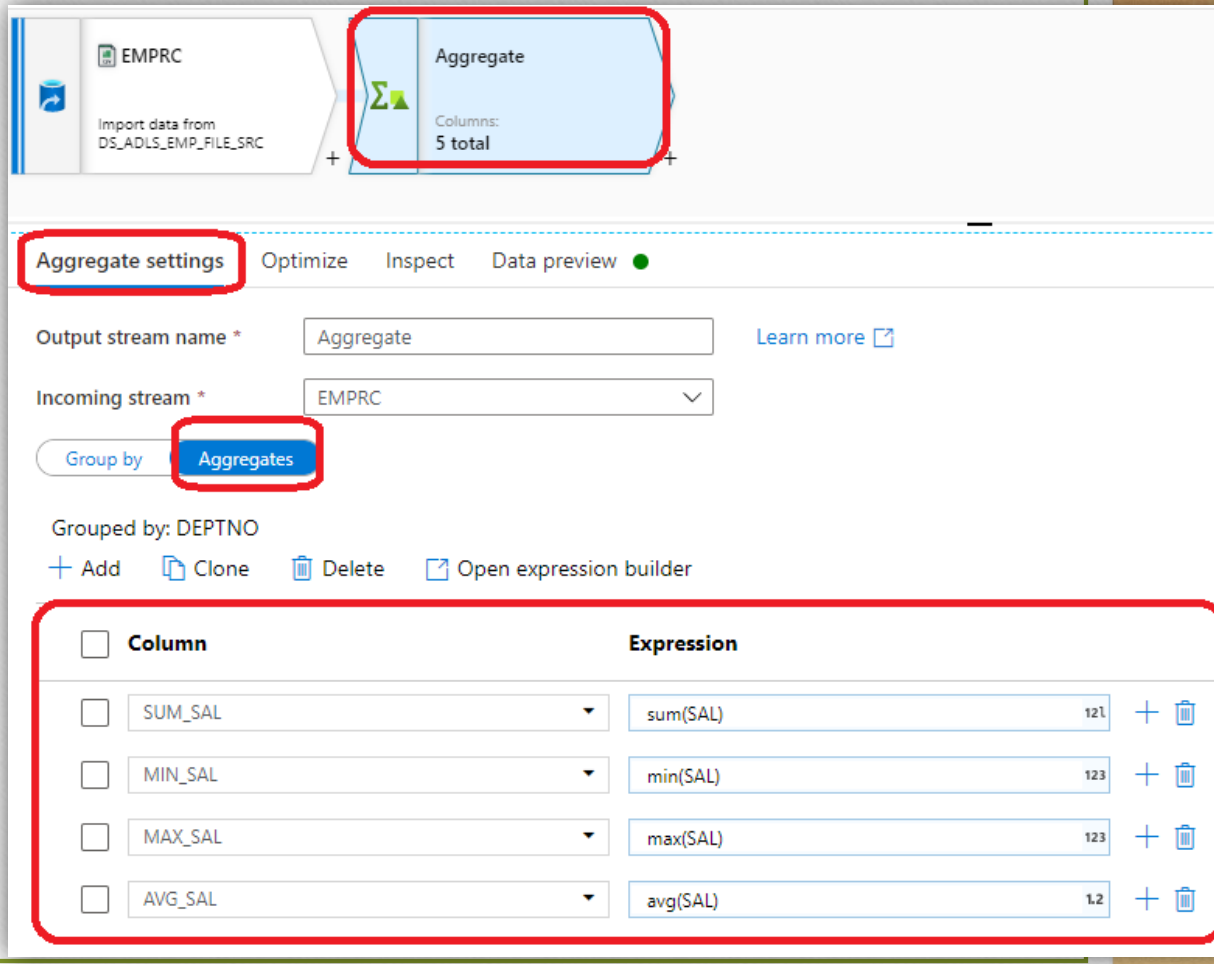
(Optional) Choose a Group-by clause for your aggregation and use either the name of an existing column or a new name. Use "Add Column" add more group-by clauses and click on the text box next to the column name to launch the Expression Builder to either select just an existing column, combination of columns or expressions for your grouping.

The Aggregate Column tab

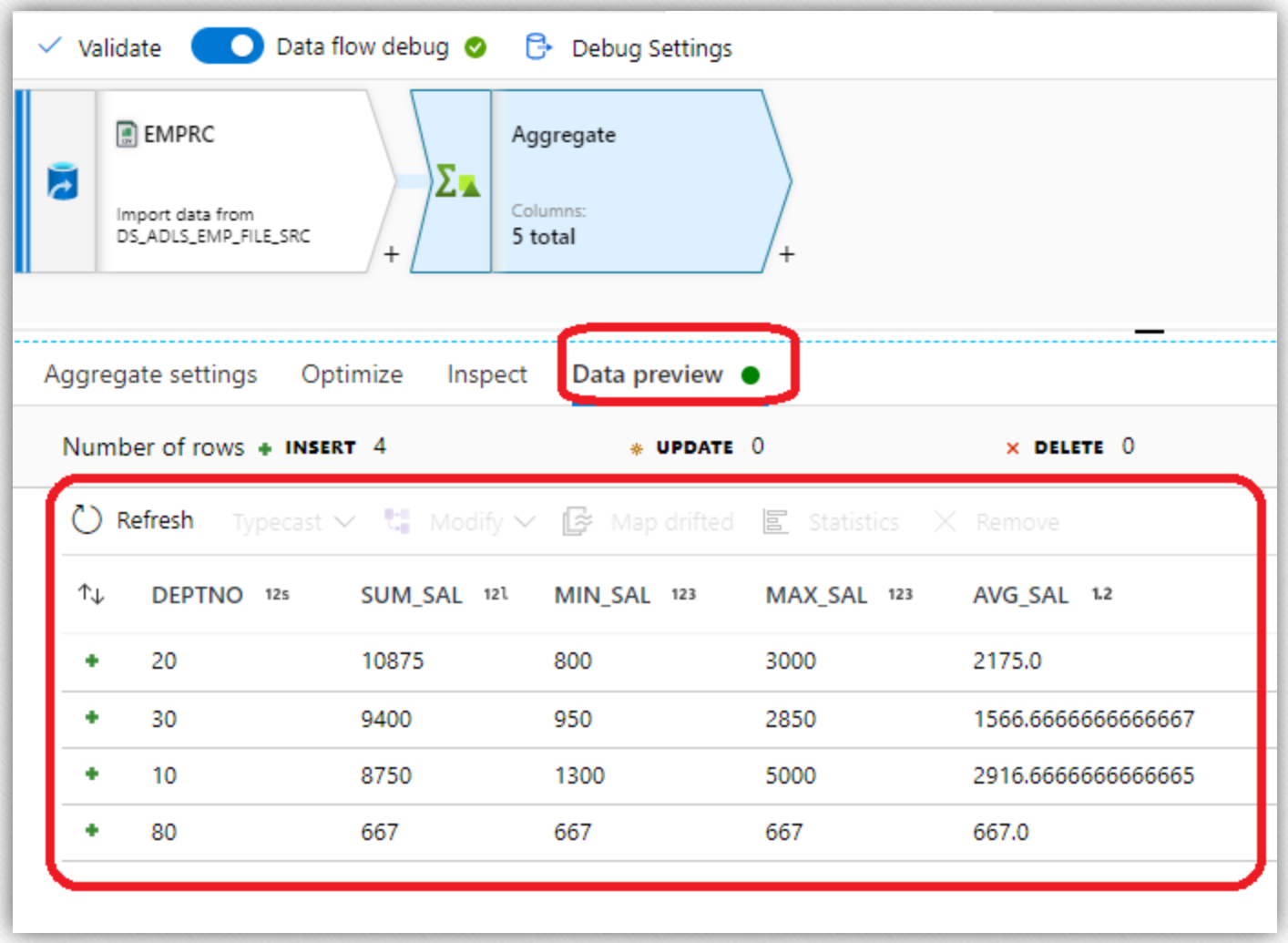
(Required) Choose the Aggregate Column tab to build the aggregation expressions. You can either choose an existing column to overwrite the value with the aggregation, or create a new field with the new name for the aggregation. The expression that you wish to use for the aggregation will be entered in the right-hand box next to the column name selector. Clicking on that text box will open up the Expression Builder.

Aggregate


EMPRC
Import data from
DS_ADLS_EMP_FILE_SRC
Aggregate
Columns:
5 total
Aggregate settings
Optimize
Inspect
Data preview
Output stream name *
Aggregate
Learn more
Incoming stream *
EMPRC
Group by
Aggregates
Columns
Name as
12s DEPTNO
DEPTNO


EMPRC
Import data from
DS_ADLS_EMP_FILE_SRC
Aggregate
Columns:
5 total
Aggregate settings
Optimize
Inspect
Data preview
Output stream name *
Aggregate
Learn more
Incoming stream *
EMPRC
Group by
Aggregates
Grouped by: DEPTNO
Add
Clone
Delete
Open expression builder
Column
Expression
SUM_SAL
sum(SAL)
MIN_SAL
min(SAL)
MAX_SAL
max(SAL)
AVG_SAL
avg(SAL)

Aggregate

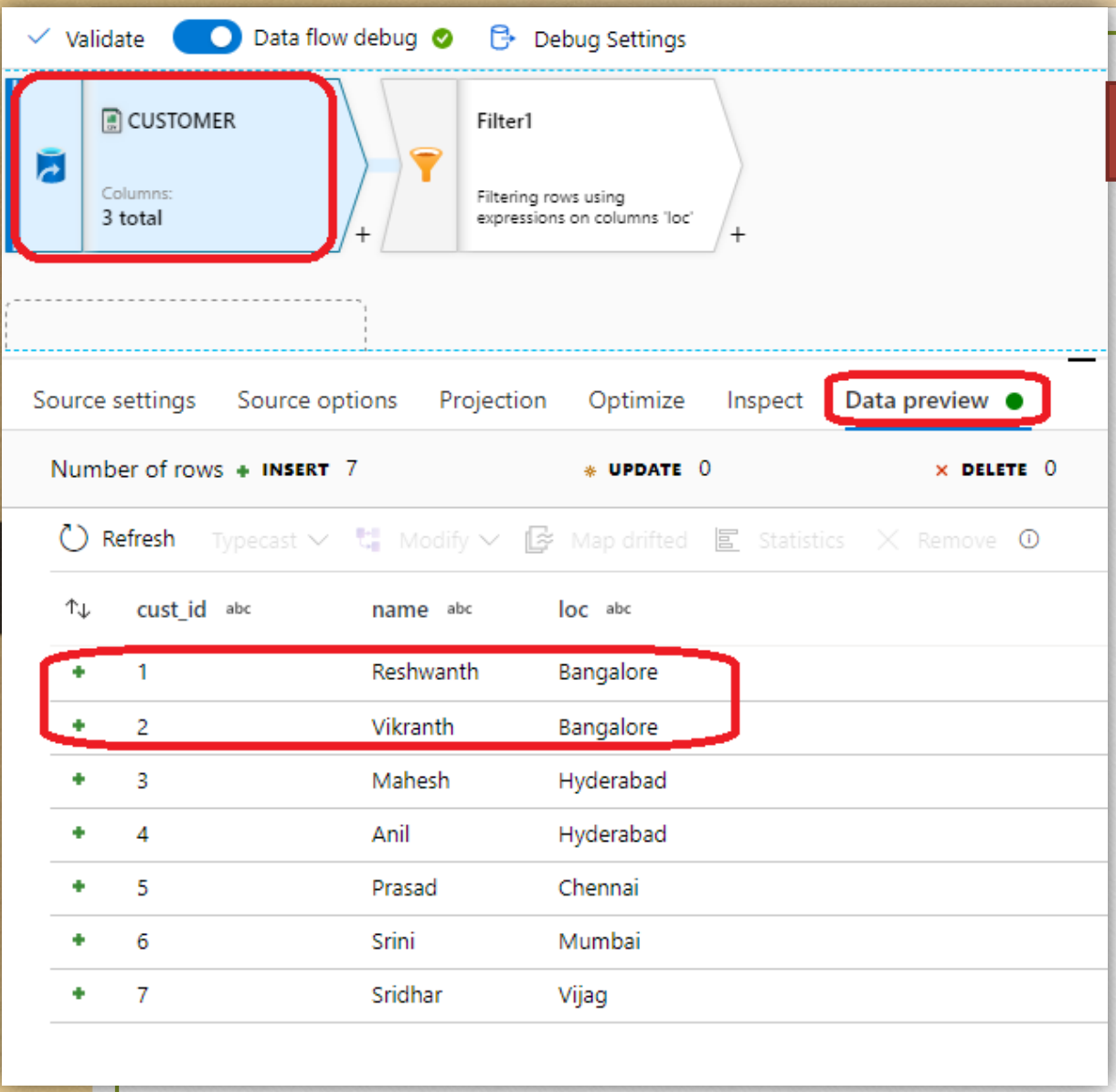

Validate
Data flow debug
Debug Settings
EMPRC
Import data from DS_ADLS_EMP_FILE_SRC
Aggregate
Columns:
5 total
Aggregate settings
Optimize
Inspect
Data preview
Number of rows
INSERT 4
UPDATE 0
DELETE 0
Refresh
Typecast
Modify
Map drifted
Statistics
Remove
DEPTNO
SUM_SAL
MIN_SAL
MAX_SAL
AVG_SAL
20 10875 800 3000 2175.0
30 9400 950 2850 1566.6666666666667
10 8750 1300 5000 2916.6666666666665
80 667 667 667 667.0

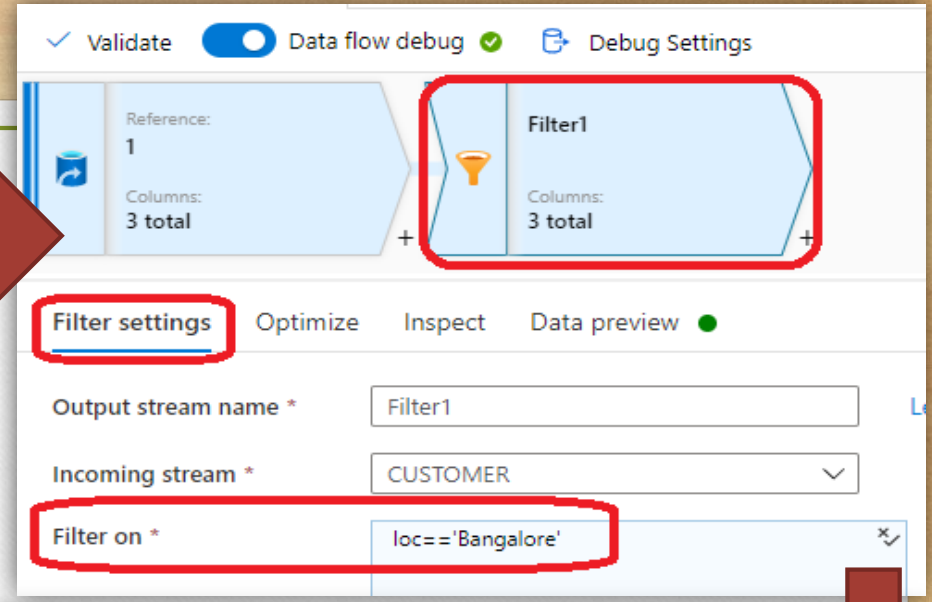
Filter

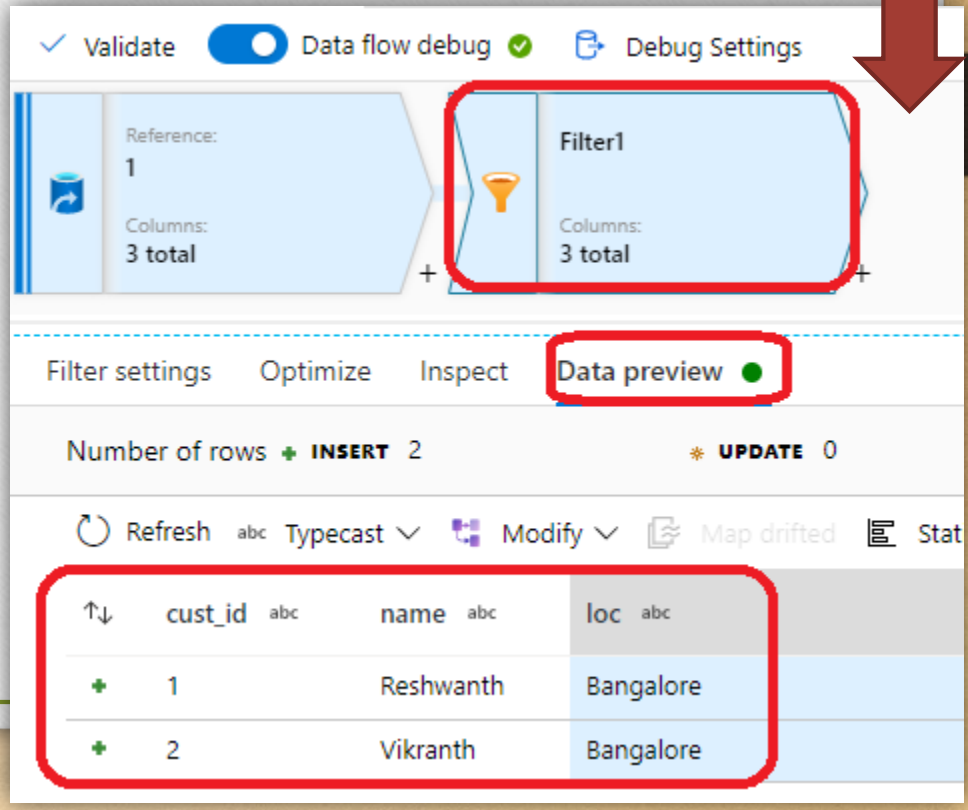
The Filter transforms is a row filtering transform that takes an expression as its parameter. Click in the text box to launch the Expression Builder. This is where you can build a filter expression which allows you to control which rows from the current data stream are allowed to pass through (filter) to the next transformation.

Tips for using Filter transform

The Filter transform can be very useful when used directly after a Source. You can use the Filter transform as the query predicate to a full table query from the source. ADF Data Flow is smart enough to take your end-to-end flows and optimize the execution utilizing pushdown techniques when available. So, using a Filter transform against what appears like a complete table scan in the design view may not actually execute as such when you attach your Data Flow to a pipeline. It is best to experiment with different techniques and use the Monitoring in ADF to gather timings and partition counts on your Data Flow activity executions.


Validate
Data flow debug
Debug Settings
CUSTOMER
Columns:
3 total
Filter1
Filtering rows using expressions on columns 'loc'
Source settings
Source options
Projection
Optimize
Inspect
Data preview
Number of rows
INSERT 7
UPDATE 0
DELETE 0
Refresh
Typecast
Modify
Map drifted
Statistics
Remove
cust_id abc
name abc
loc abc
1 Reshwanth Bangalore
2 Vikranth Bangalore
3 Mahesh Hyderabad
4 Anil Hyderabad
5 Prasad Chennai
6 Srini Mumbai
7 Sridhar Vijag


Validate
Data flow debug
Debug Settings
Reference:
1
Columns:
3 total
Filter1
Columns:
3 total
Filter settings
Optimize
Inspect
Data preview
Output stream name *
Filter1
Incoming stream *
CUSTOMER
Filter on *
loc=='Bangalore'


Validate
Data flow debug
Debug Settings
Reference:
1
Columns:
3 total
Filter1
Columns:
3 total
Filter settings
Optimize
Inspect
Data preview
Number of rows
INSERT 2
UPDATE 0
Refresh
Typecast
Modify
cust_id abc
name abc
loc abc
1 Reshwanth Bangalore
2 Vikranth Bangalore

Exists Transformation

The Exists transformation is a row filter transforms that allows you to filter rows in your data that exists (SQL WHERE EXISTS) or do not exist (SQL WHERE NOT EXISTS). The resulting rows from your data stream after this transformation will either include all rows where column values from source 1 exist in source 2 or do NOT exist in source 2.

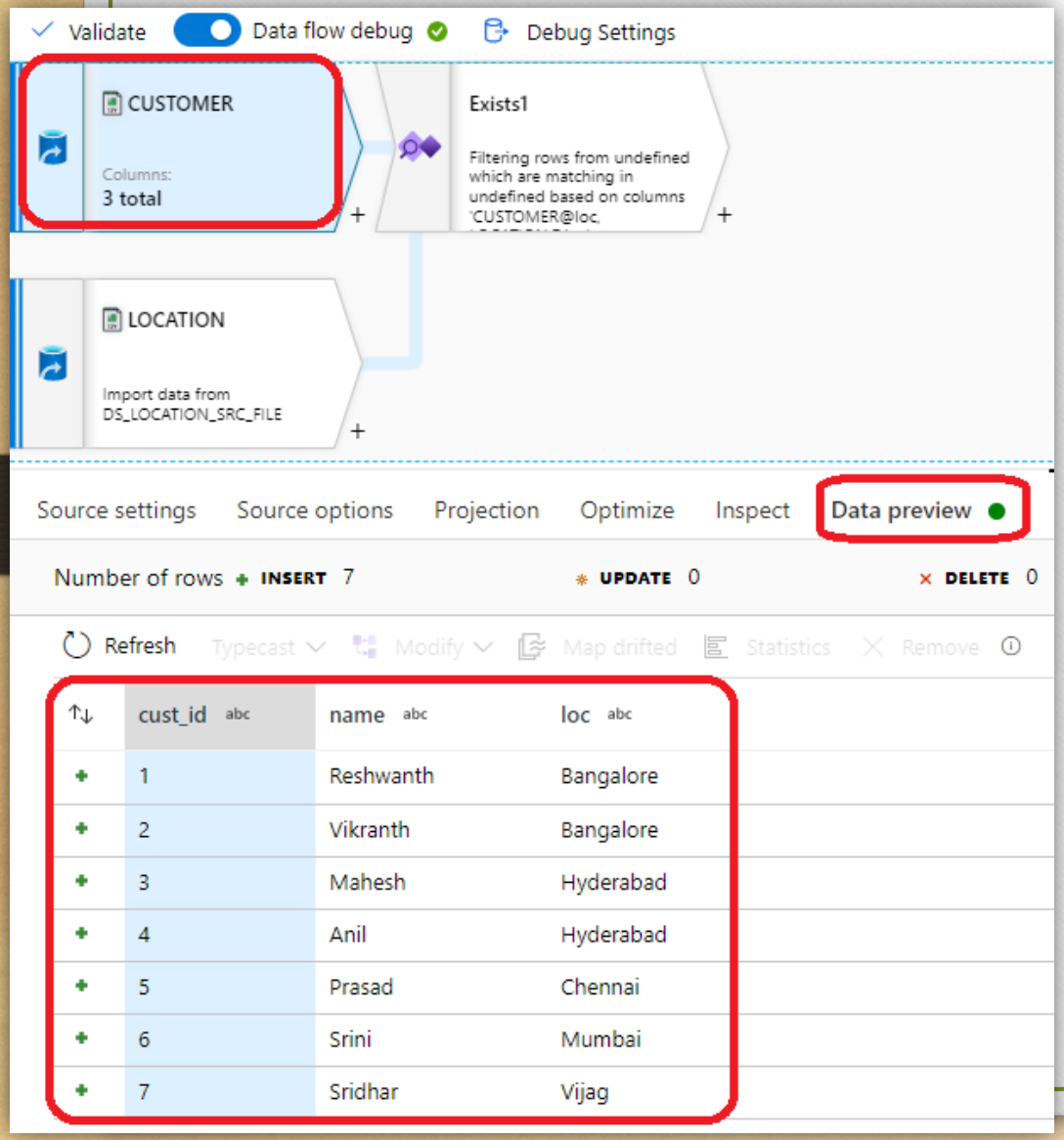
Settings

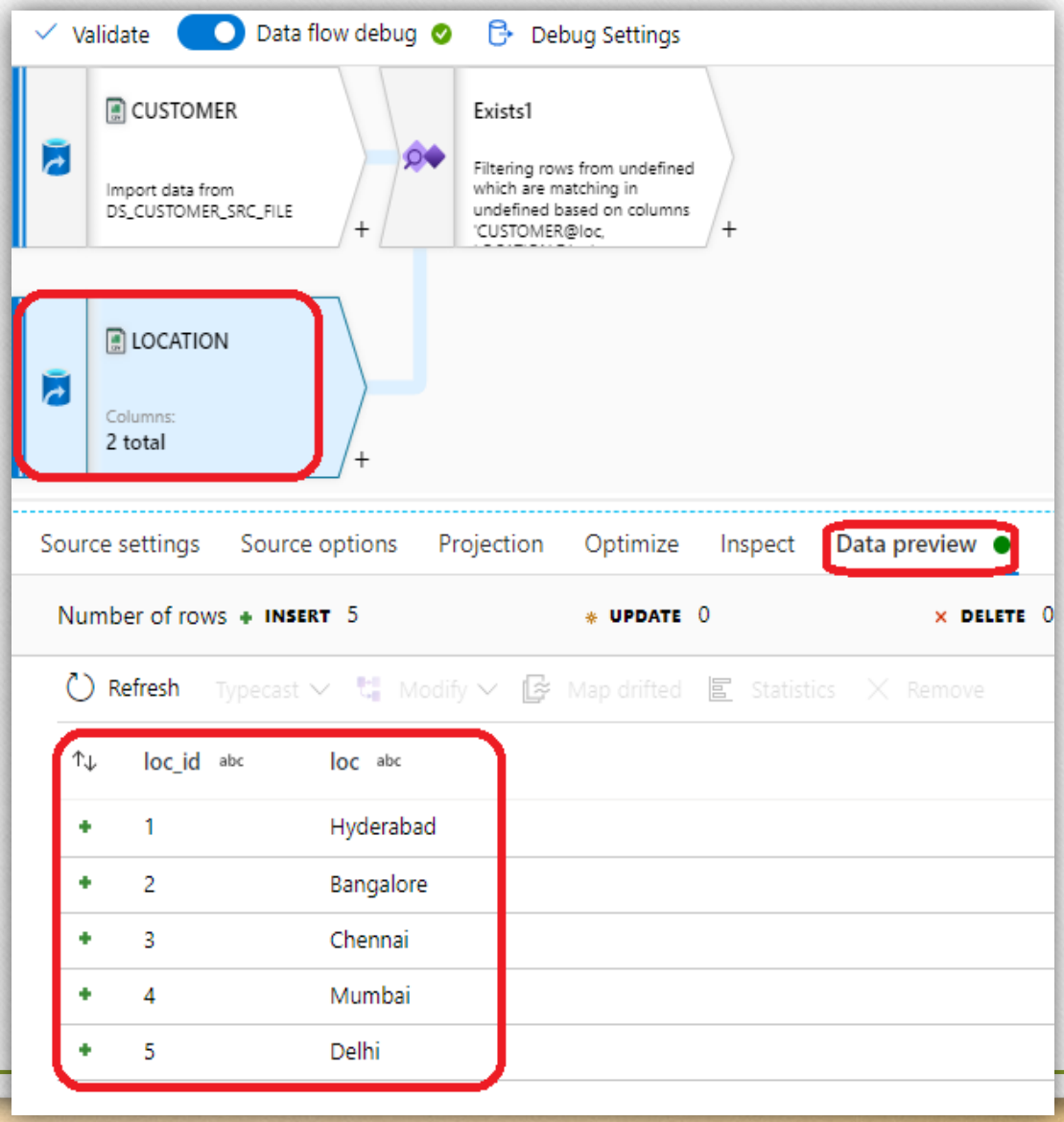
Fed by

Choose the 2nd source for your Exists so that Data Flow can compare values from Stream 1 against Stream 2

Select the column from Source 1 and from Source 2 whose values you wish to check against for Exists or Not Exists.

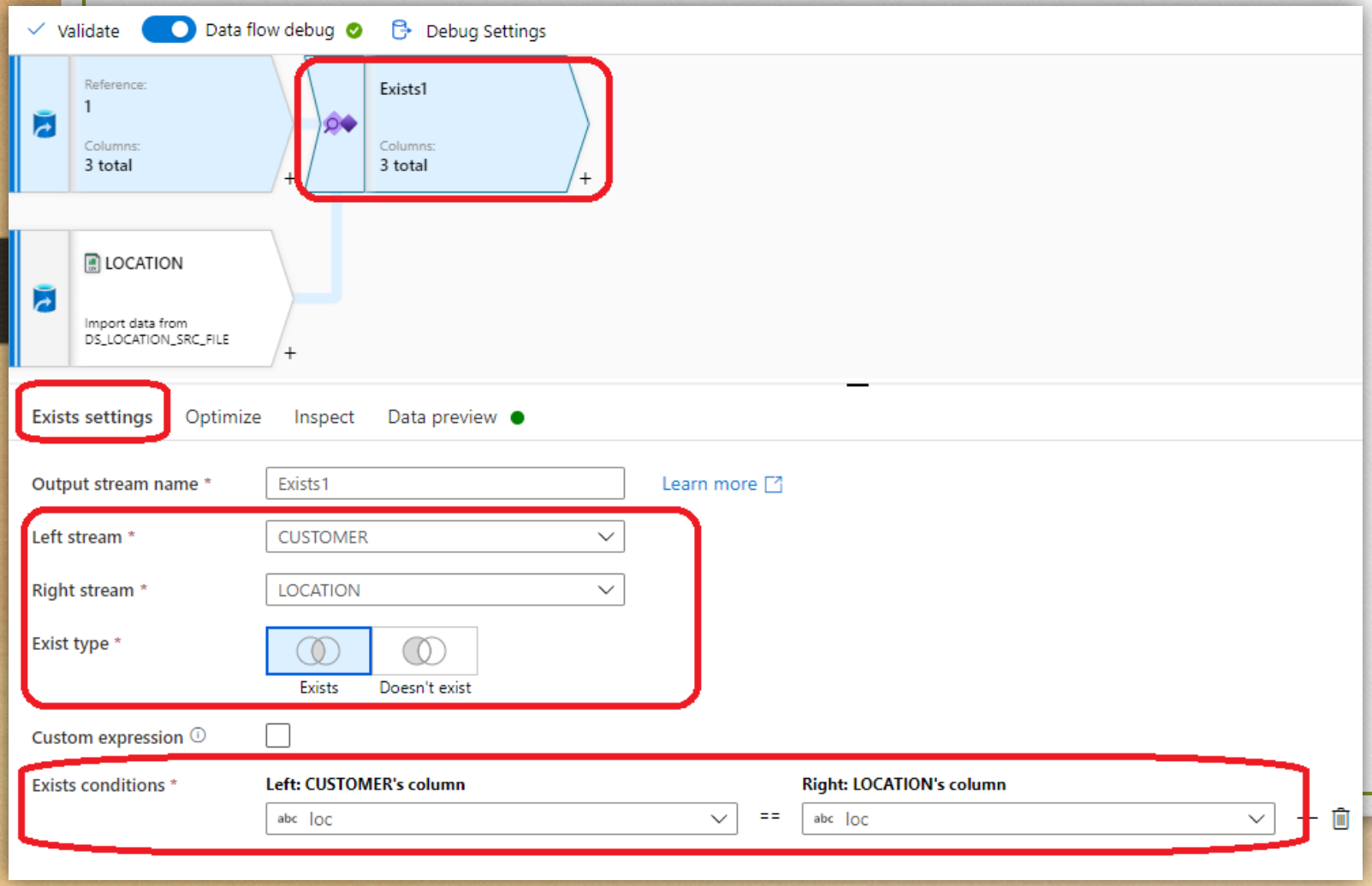
Exists Transformation

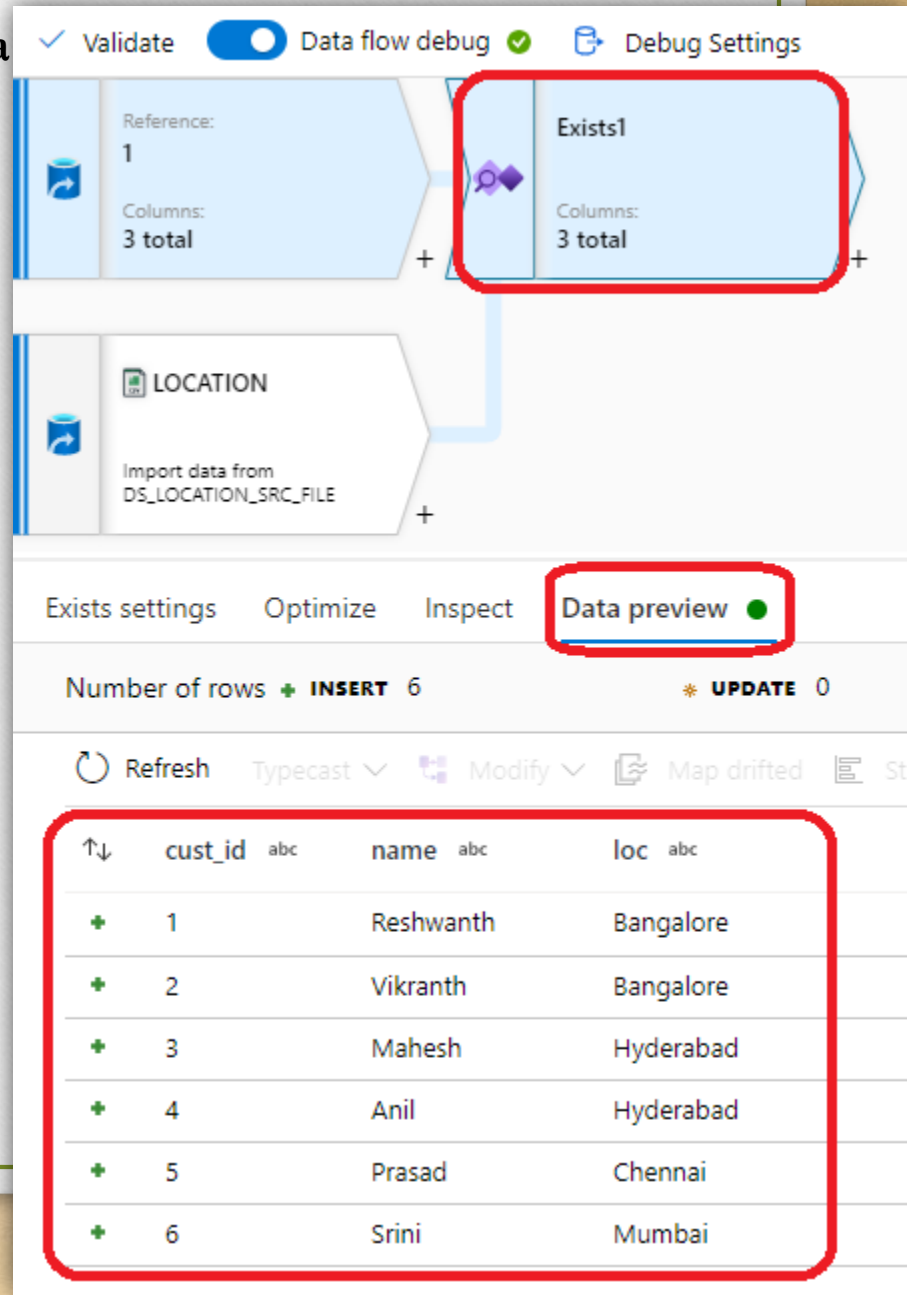




Exists Transformation

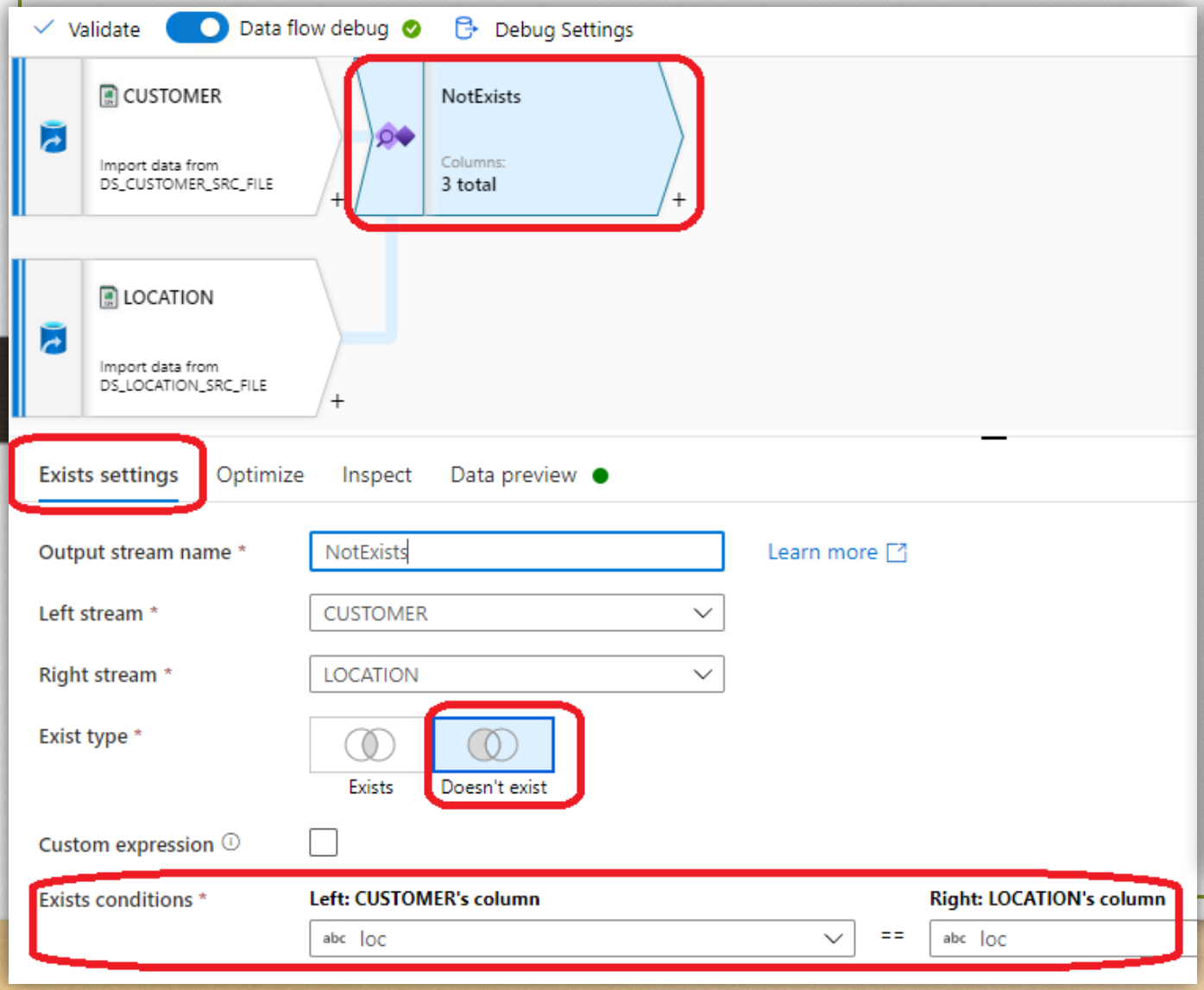
Exists Transformation is equal to **Left Semi join**. It will **display matching data** From left table. In this example it will display only matching data from Customer table it will give.

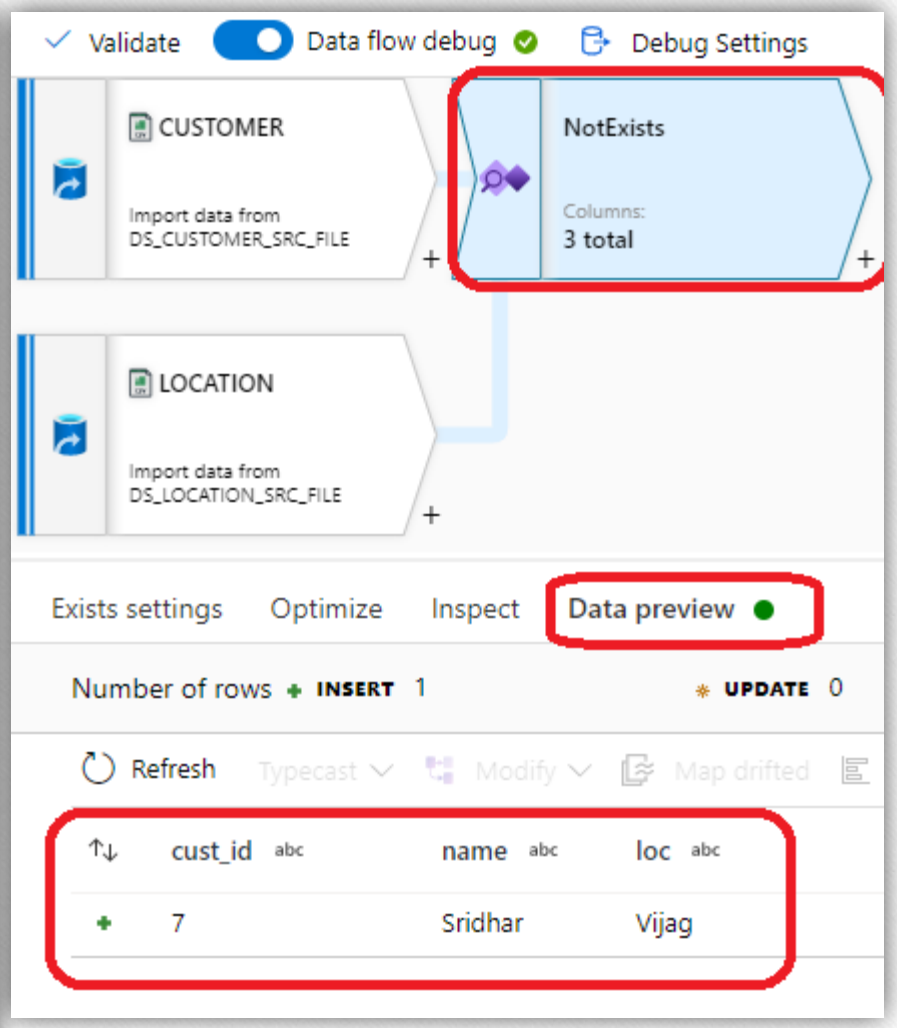




Doesn't Exists Transformation

Not Exists is equal to Left Anti Join. Which will display unmated data from left table. Here un-matched data from Customer Table.





Unpivot

Use Unpivot as a way to turn a non-normalized dataset into a more normalized version by expanding values from multiple columns in a single record into multiple records with the same values in a single column

Ungroup By

First, set the columns that you wish to group by for your pivot aggregation. Set more than one column to ungroup by using the + sign next to the column list.

Unpivot Key

The Pivot Key is the column that ADF will pivot from row to column. By default, each unqiue value in the dataset for this field will pivot to a column. However, you can optionally enter the values from the dataset that you wish to pivot to column values.

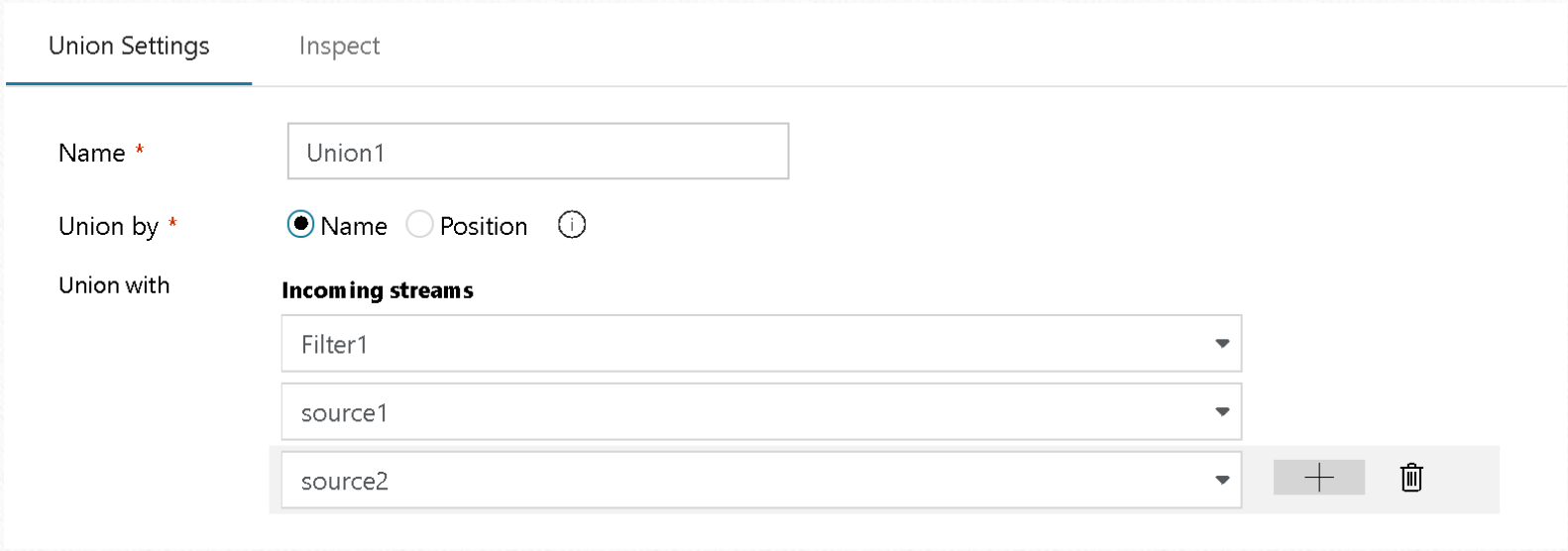
Unpivoted Columns

Lastly, you will choose the aggregation that you wish to use for the pivoted values and how you would like the columns to be displayed in the new output projection from the transformation.

(Optional) You can set a naming pattern with a prefix, middle, and suffix to be added to each new column name from the row values.
For instance, pivoting "Sales" by "Region" would simply give you new column values from each sales value. For example: "25", "50", "1000", ... However, if you set a prefix value of "Sales", then the pivoted columns will include "Sales" in the values.

UNION Transformation

Union will combine multiple data streams into one, with the SQL Union of those streams as the new output from the Union transformation.



Surrogate Key Transformation

Use the Surrogate Key Transformation to add an incrementing non-business arbitrary key value to your data flow rowset. This is useful when designing dimension tables in a star schema analytical data model where each member in your dimension tables needs to have a unique key that is a non-business key, part of the Kimball DW methodology.

"Key Column" is the name that you will give to your new surrogate key column.

"Start Value" is the beginning point of the incremental value.

Sort

Sort settings

The Sort transformation allows you to sort the incoming rows on the current data stream. The outcoming rows from the Sort Transformation will subsequently follow the ordering rules that you set. You can choose invidual columns and sort them ASC or DEC, using the arrow indicator next to each field. If you need to modify the column before applying the sort, click on "Computed Columns" to launch the expression editor. This will provide with an opportunity to build an expression for the sort operation instead of simply applying a column for the sort.

You can turn on "Case insensitive" if you wish to ignore case when sorting string or text fields.

"Sort Only Within Partitions" leverages the Spark data partitioning capability to sort incoming data only within each partition as opposed to the entire data stream.

Each of the sort conditions in the Sort Transformation can be re-arranged. So if you need to move a column higher in the sort precendence, grab that row with your mouse and move it higher or lower in the sorting list.

Partitioning effects on Sort
ADF Data Flow is executing on big data Spark clusters in the backend that distributes data across multiple nodes and multiple data partitions. It is important to keep this in mind when architecting your data flow in ADF where you are depending on the Sort transform to keep data in that same order. If you choose repartition your data in a subsequent transformation, you may lose your sorting due to that reshuffling of data.

Select Transformation

Use this transformation for column selectivity (reducing number of columns) or to alias columns and stream names.

In ADF Data Flow, all of your columns will automatically propagate throughout your streams. As your columns accumulate, you can use the Select transform to "select" the columns that you wish to keep. You will see stream names appended to the column names to indicate the origins and lineage of those columns to help you make the proper determination.

While you can always choose column selectivity in the Sink transform at the end of your Data Flow, maintaining column hygiene may help you to prune your column lists. The downside to this approach is that you will lose that metadata downstream and will not be able to access it once you've dropped it from your metadata with a Select.

The Select transform allows you to alias an entire stream, or columns in that stream, assign different names (aliases) and then reference those new names later in your data flow. This is very useful for self-join scenarios. The way to implement a self-join in ADF Data Flow is to take a stream, branch it with "New Branch", then immediately afterward, add a "Select" transform. That stream will now have a new name that you can use to join back to the original stream, creating a self-join:

In the above diagram, the Select transform is at the top. All it's doing is aliasing the original stream to "OrigSourceBatting". In the higlighted Join transform below it you can see that we use this Select alias stream as the right-hand join, allowing us to reference the same key in both the Left & Right side of the Inner Join.

Select can also be used as a way de-select columns from your data flow. For example, if you have 6 columns defined in your sink, but you only wish to pick a specific 3 to transform and then flow to the sink, you can select just those 3 by using the select transform.

NOTE: You must switch off Select All to pick only specific columns
Options
The default setting for "Select" is to include all incoming columns and keep those original names. You can alias the stream by setting the name of the Select transform.

To alias individual columns, deselect "Select All" and use the column mapping at the bottom.

ADF Data Flow Pivot Transformation

Use Pivot in ADF Data Flow as an aggregation where one or more grouping columns has its distinct row values transformed into individual columns. Essentially, you can Pivot row values into new columns (turn data into metadata).

Group By

First, set the columns that you wish to group by for your pivot aggregation. You can set more than 1 column here with the + sign next to the column list.

Pivot Key

The **Pivot Key** is the column that ADF will pivot from row to column. By default, each unqiue value in the dataset for this field will pivot to a column. However, you can optionally enter the values from the dataset that you wish to pivot to column values.

Lastly, you will choose the aggregation that you wish to use for the pivoted values and how you would like the columns to be displayed in the new output projection from the transformation.

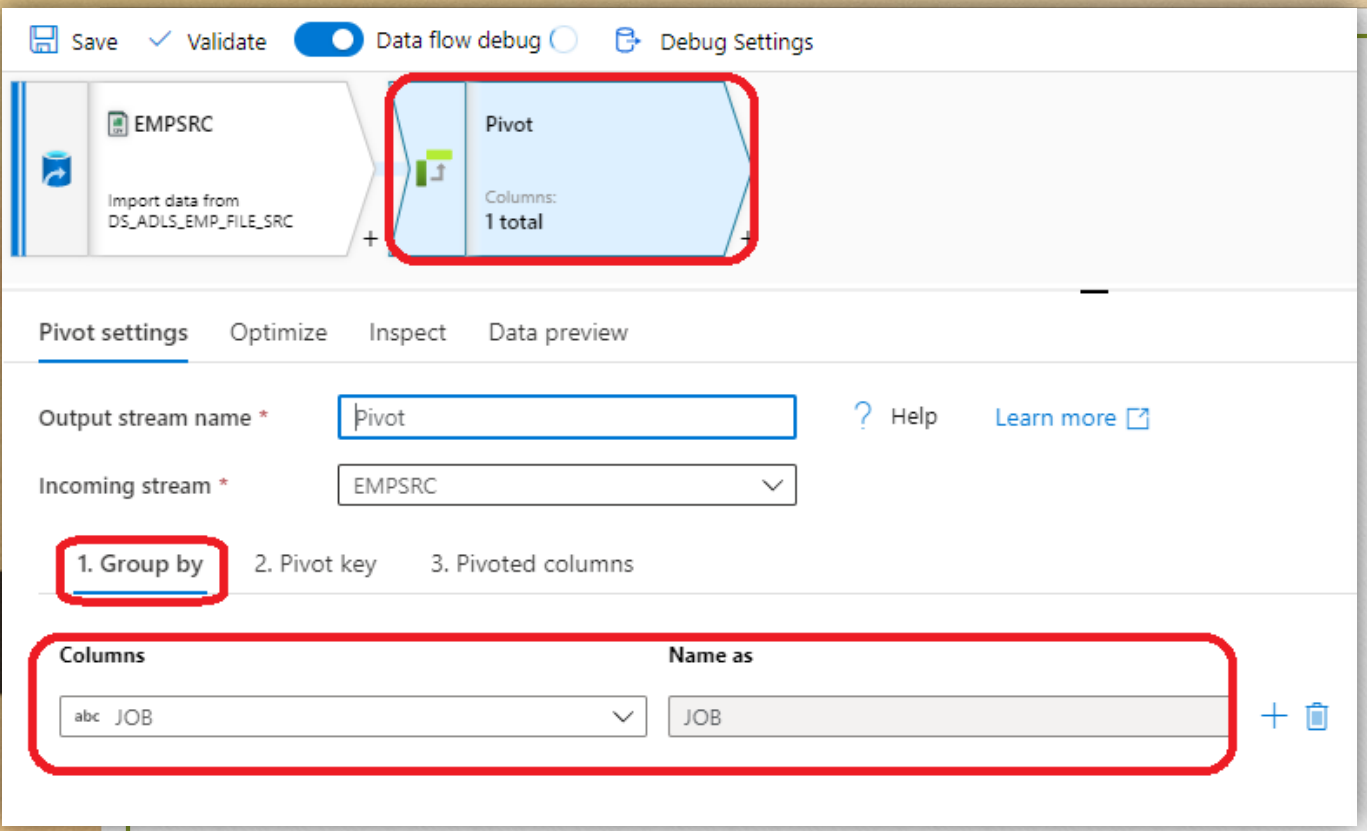
(Optional) You can set a naming pattern with a prefix, middle, and suffix to be added to each new column name from the row values.

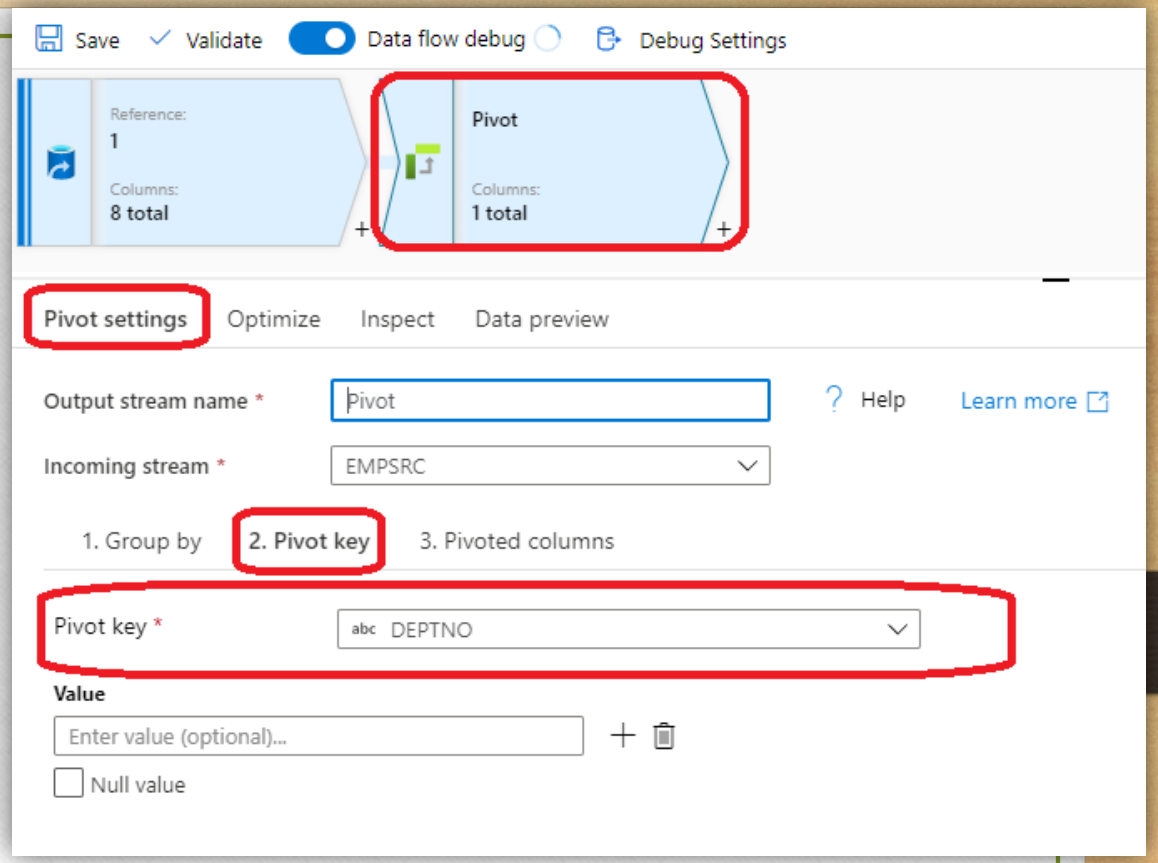
For instance, pivoting "Sales" by "Region" would simply give you new column values from each sales value, i.e. "25", "50", "1000", etc. However, if you set a prefix value of "Sales "

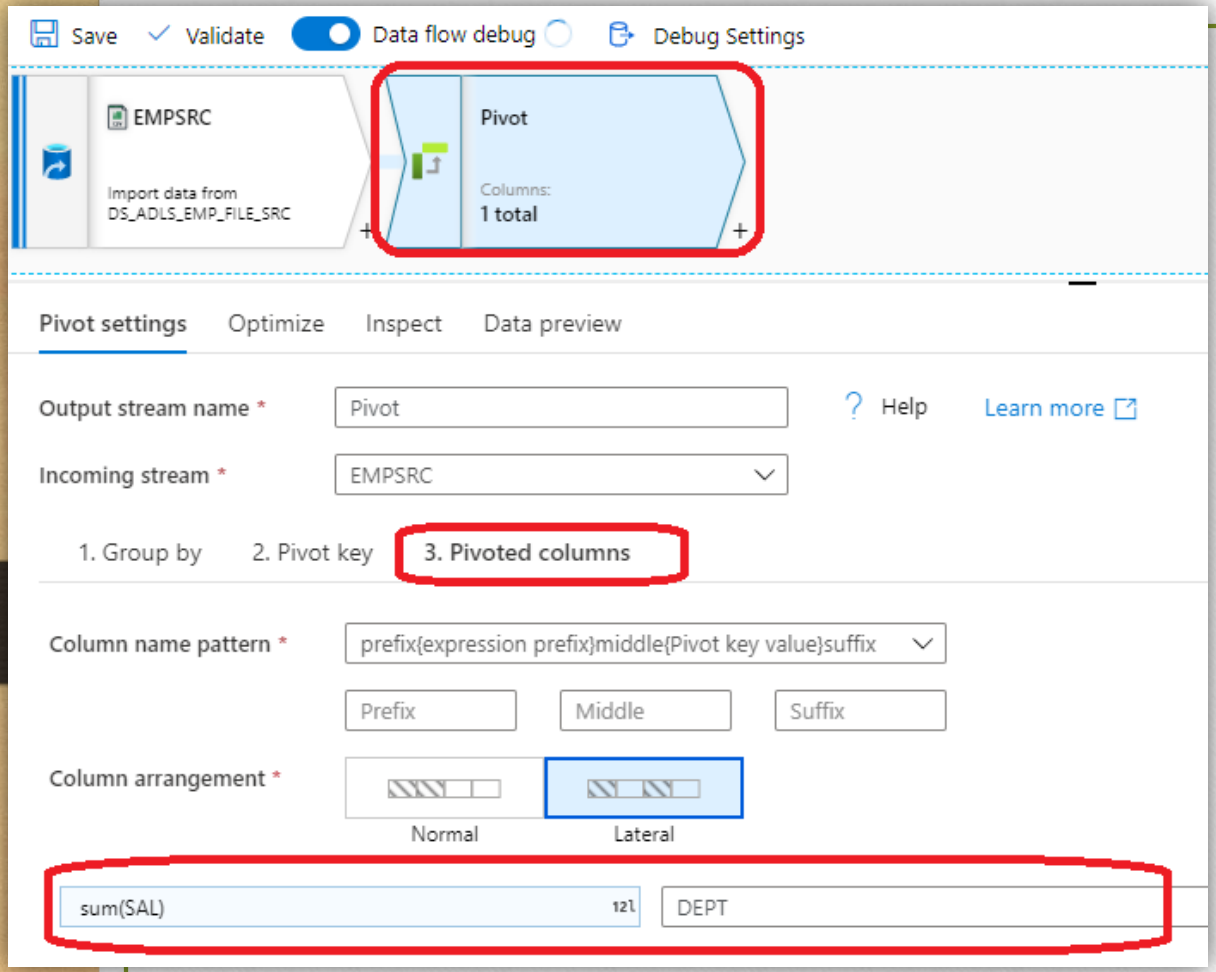
Setting the Column Arrangement to "Normal" will group together all of the pivoted columns with their aggregated values. Setting the columns arrangment to "Lateral" will alternate between column and value.

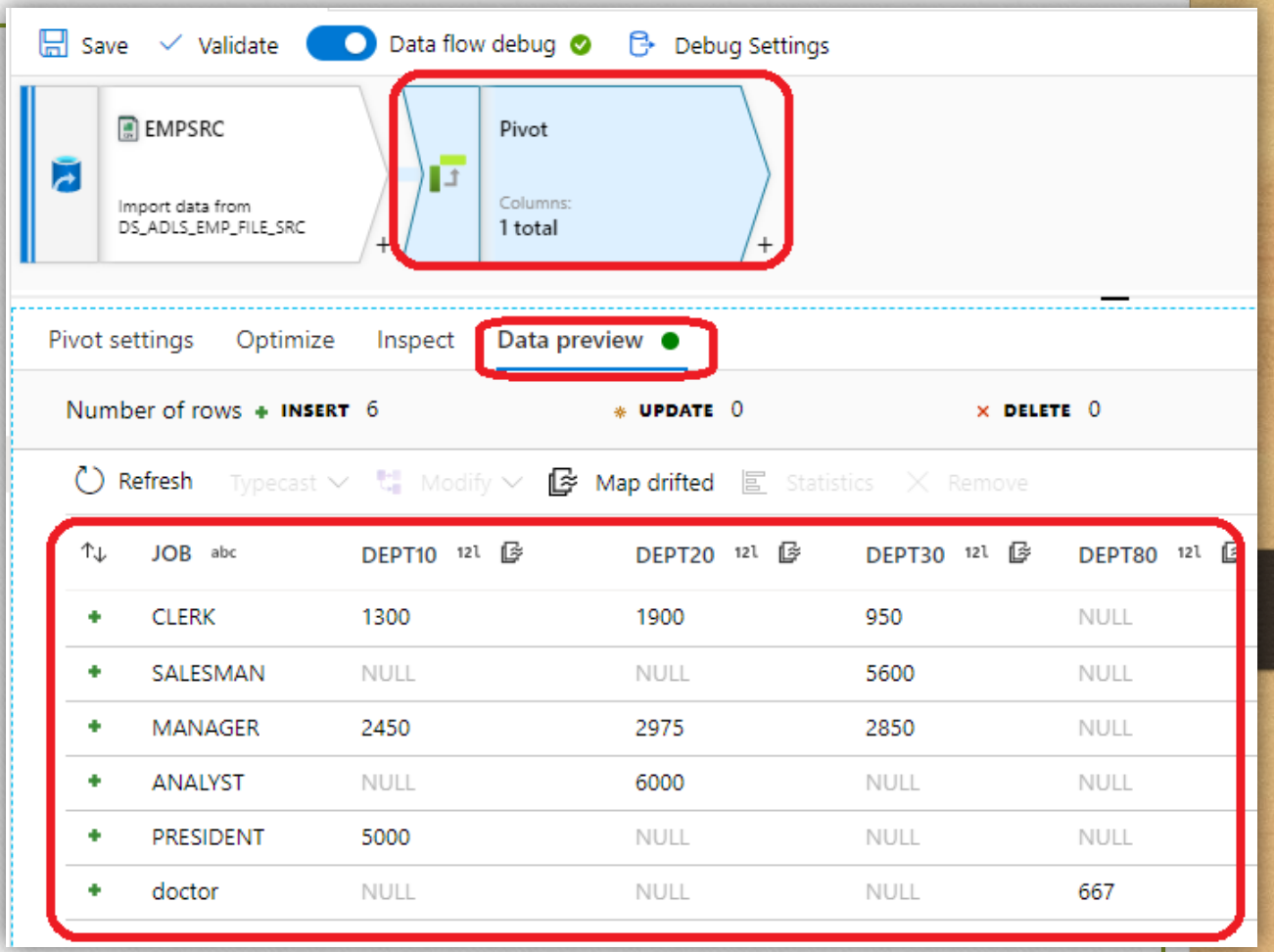
Aggregation
To set the aggregation you wish to use for the pivot values, click on the field at the bottom of the Pivoted Columns pane. You will enter into the ADF Data Flow expression builder where you can build an aggregation expression and provide a descriptive alias name for your new aggregated values.

Use the ADF Data Flow Expression Language to describe the pivoted column transformations in the Expression Builder
How to rejoin original fields
NOTE: The Pivot transformation will only project the columns used in the **aggregation**, **grouping**, and **pivot action**. If you wish to include the other columns from the previous step in your flow, use a New Branch from the previous step and use the self-join pattern to connect the flow with the original metadata


Save
Validate
Data flow debug
Debug Settings
EMPSRC
Import data from
DS_ADLS_EMP_FILE_SRC
Pivot
Columns:
1 total
Pivot settings
Optimize
Inspect
Data preview
Output stream name *
Pivot
Help
Learn more
Incoming stream *
EMPSRC
1. Group by
2. Pivot key
3. Pivoted columns
Columns
Name as
JOB
JOB


Save
Validate
Data flow debug
Debug Settings
Reference:
1
Columns:
8 total
Pivot
Columns:
1 total
Pivot settings
Optimize
Inspect
Data preview
Output stream name *
Pivot
Help
Learn more
Incoming stream *
EMPSRC
1. Group by
2. Pivot key
3. Pivoted columns
Pivot key *
DEPTNO
Value
Enter value (optional)...
Null value


Save
Validate
Data flow debug
Debug Settings
EMPSRC
Import data from DS_ADLS_EMP_FILE_SRC
Pivot
Columns:
1 total
Pivot settings
Optimize
Inspect
Data preview
Output stream name *
Pivot
Help
Learn more
Incoming stream *
EMPSRC
1. Group by
2. Pivot key
3. Pivoted columns
Column name pattern *
prefix{expression prefix}middle{Pivot key value}suffix
Prefix
Middle
Suffix
Column arrangement *
Normal
Lateral
sum(SAL)
DEPT


Save
Validate
Data flow debug
Debug Settings
EMPSRC
Import data from DS_ADLS_EMP_FILE_SRC
Pivot
Columns:
1 total
Pivot settings
Optimize
Inspect
Data preview
Number of rows
INSERT 6
UPDATE 0
DELETE 0
Refresh
Typecast
Modify
Map drifted
Statistics
Remove
JOB
DEPT10
DEPT20
DEPT30
DEPT80
CLERK 1300 1900 950 NULL
SALESMAN NULL NULL 5600 NULL
MANAGER 2450 2975 2850 NULL
ANALYST NULL 6000 NULL NULL
PRESIDENT 5000 NULL NULL NULL
doctor NULL NULL NULL 667

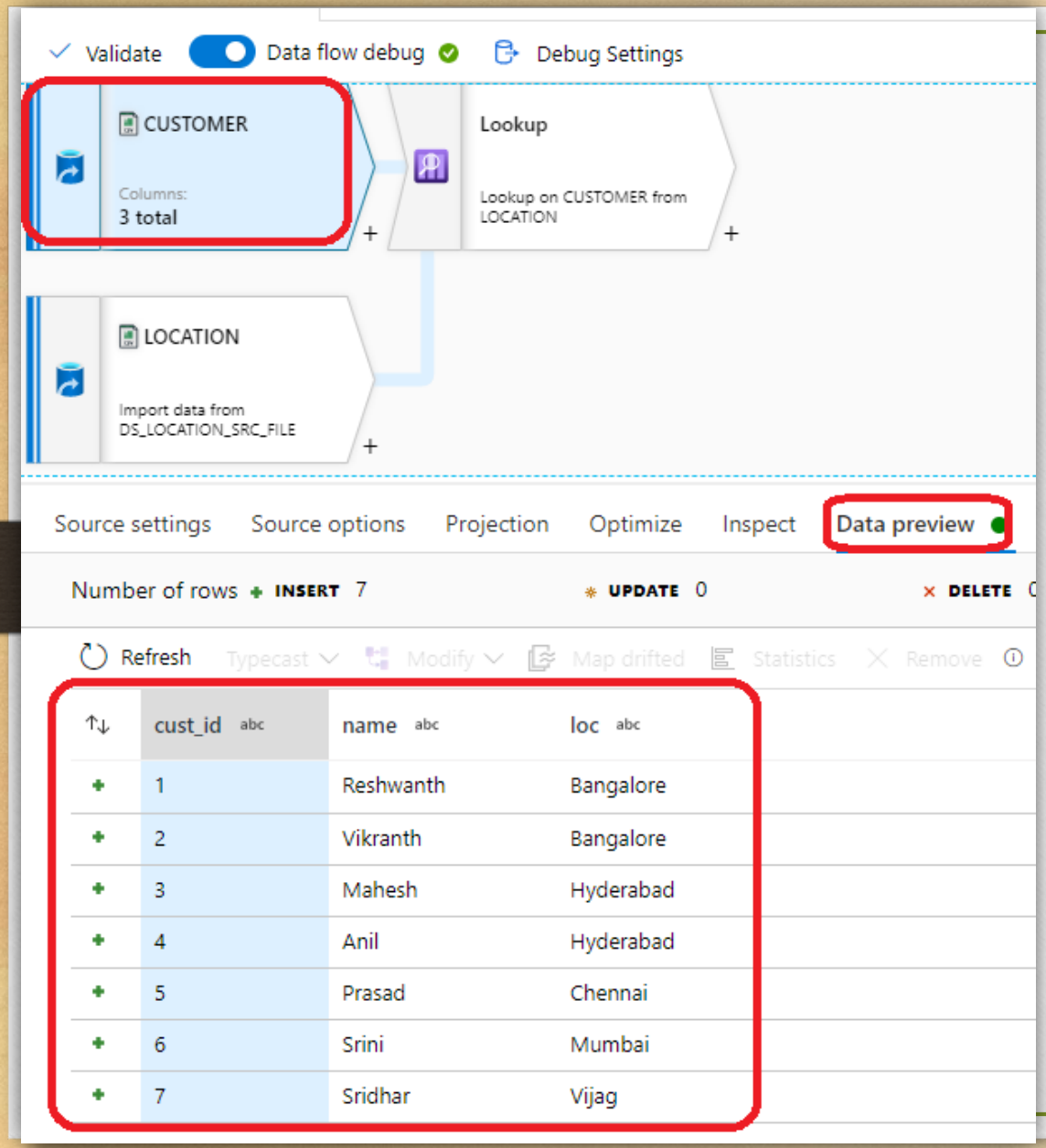
Lookup

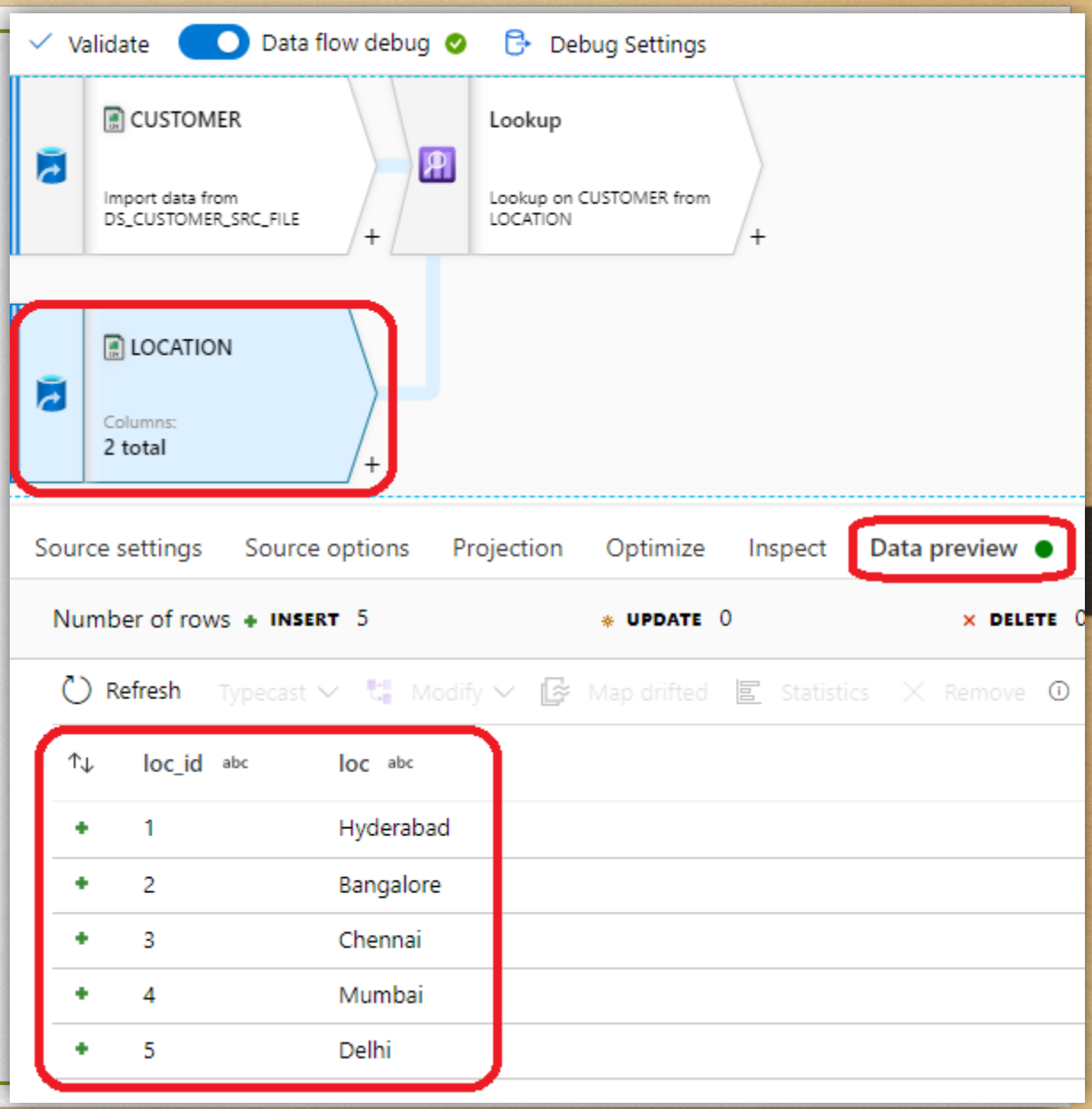
Use Lookup to add reference data from another source to your Data Flow. The Lookup transform requires a defined source that points to your reference table and matches on key fields.

Select the key fields that you wish to match on between the incoming stream fields and the fields from the reference source. You must first have created a new source on the Data Flow design canvas to use as the right-side for the lookup.

When matches are found, the resulting rows and columns from the reference source will be added to your data flow. You can choose which fields of interest that you wish to include in your Sink at the end of your Data Flow or use the Select tranformation for column selectivity.

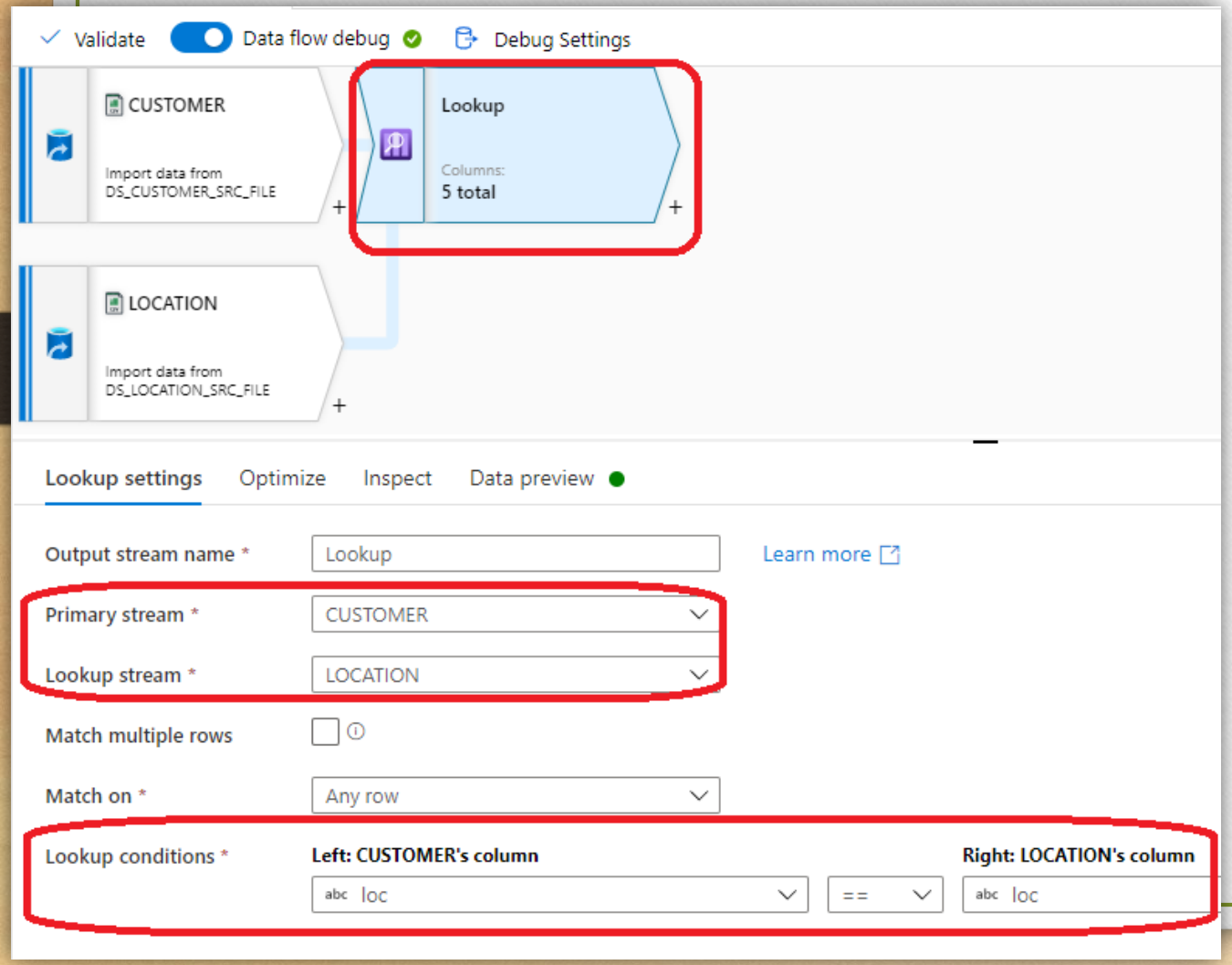
Lookup

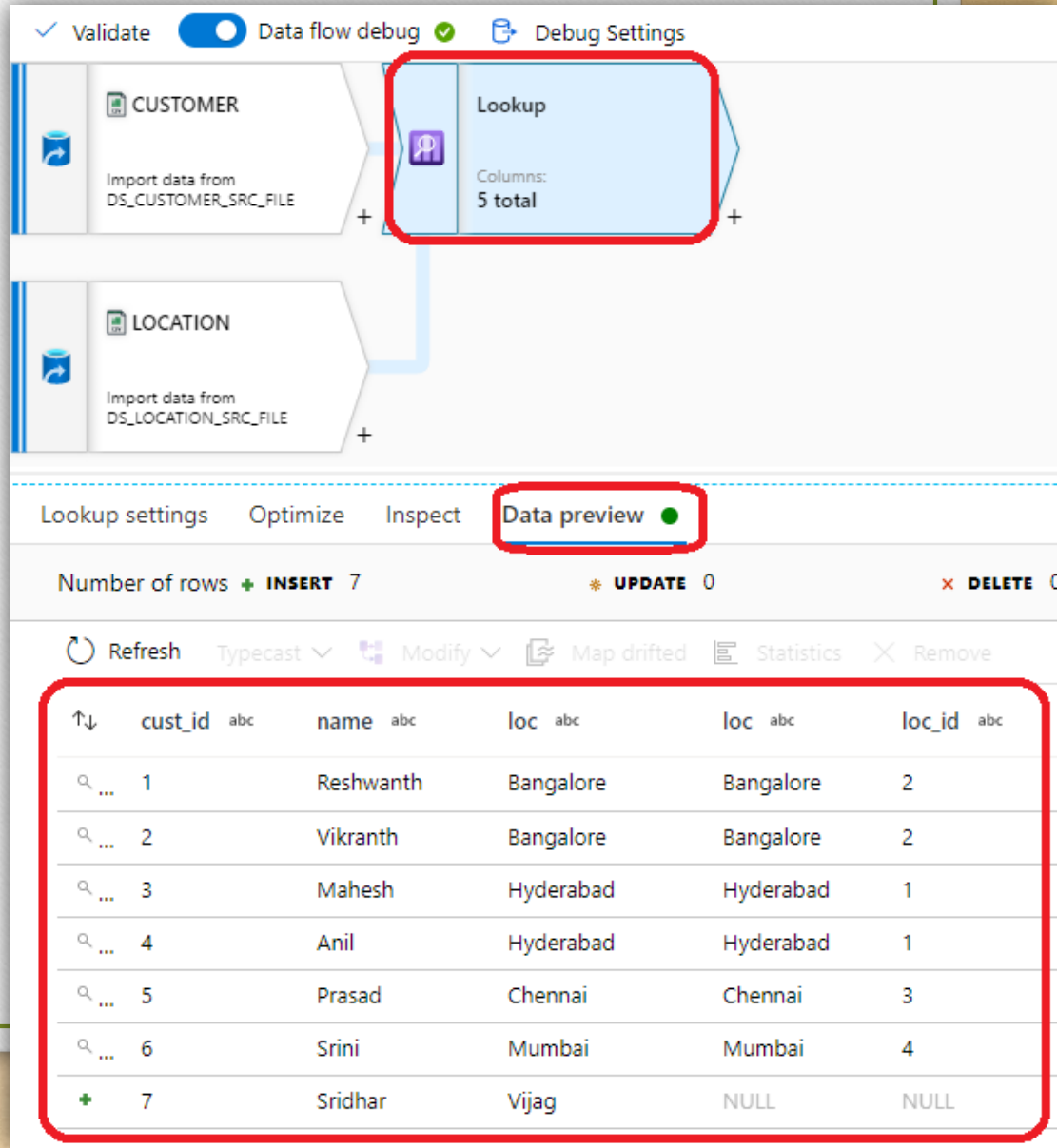

Validate
Data flow debug
Debug Settings
CUSTOMER
Columns:
3 total
Lookup
Lookup on CUSTOMER from LOCATION
LOCATION
Import data from DS_LOCATION_SRC_FILE
Source settings
Source options
Projection
Optimize
Inspect
Data preview
Number of rows INSERT 7
UPDATE 0
DELETE
Refresh
Typecast
Modify
Map drifted
Statistics
Remove
cust_id abc
name abc
loc abc
1 Reshwanth Bangalore
2 Vikranth Bangalore
3 Mahesh Hyderabad
4 Anil Hyderabad
5 Prasad Chennai
6 Srini Mumbai
7 Sridhar Vijag


Validate
Data flow debug
Debug Settings
CUSTOMER
Import data from DS_CUSTOMER_SRC_FILE
Lookup
Lookup on CUSTOMER from LOCATION
LOCATION
Columns:
2 total
Source settings
Source options
Projection
Optimize
Inspect
Data preview
Number of rows INSERT 5
UPDATE 0
DELETE
Refresh
Typecast
Modify
Map drifted
Statistics
Remove
loc_id abc
loc abc
1 Hyderabad
2 Bangalore
3 Chennai
4 Mumbai
5 Delhi

Lookup

Use Lookup to add reference data from another source to your Data Flow. The Lookup transform requires a defined source that points to your reference table and matches on key fields.





Branching will take the current data stream in your data flow and replicate it to another stream. This allows you to perform multiple sets of different operations and transformations against the same data stream.

Example: You have a Source Transform that includes a selected set of columns with data type conversions and then place a Derived Column immediately following that Source. In the Derived Column, you've create a nwe field that combines first name and last name to make a new "full name" field.

You can treat that new stream with a set of transformations and a sink on one row and use New Branch to create a copy of that stream to perform a completely different set of transformations and sink on another row.

** NOTE: "New Branch" will only show as an action on the + Transformation menu when there is a subsequent transformation following the current location where you are attempting to branch. i.e. You will not see a "New Branch" option at the end here until you add another transformation after the Select:

Join

Use Join to combine data from 2 tables in your Data Flow. Add the Join transform with the Plus icon on the data stream where you would like to join and then inside the transform you will select the 2nd Join stream.

Join types

Selecting Join Type is required for the Join transformation

Inner Join

Inner join will pass through only rows that match the column conditions from both tables

Left Outer

All rows from the left stream not meeting the join condition are passed through, and output columns from the other table are set to NULL in addition to all rows returned by the inner join.

Right Outer

All rows from the right stream not meeting the join condition are passed through, and output columns that correspond to the other table are set to NULL, in addition to all rows returned by the inner join.

Full Outer

Full Outer produces all columns and rows from both sides with NULL values for columns that are not present in the other table

Cross Join

Specific the cross product of the 2 streams with an expression and use this option to write a free-form expression for other JOIN types.

Specify Join Conditions

The Left Join condition is from the data stream connected to the left of your Join. The Right Join condition is the second data stream connected to your Join on the bottom, which will either be a direct connector to another stream or a reference to another stream.

You are required to enter at least 1 (1..n) join conditions. They can be either directly-referenced fields selected from the drop-down menu, or expressions.

Join Peformance Optimizations

Please note that unlike Merge Join in tools like SSIS, ADF's Join in Data Flow is not a mandatory merge join operation. Therefore, the join keys do not need to be sorted first. The Join operation will occur in Spark using Databricks based on the optimal join operation in Spark: Broadcast / Map-side join:

If your dataset can fit into the Databricks worker node memory, we can optimize your Join performance. You can also specify partitioning of your data on the Join operation to create sets of data that can fit better into memory per worker.

Self-Join

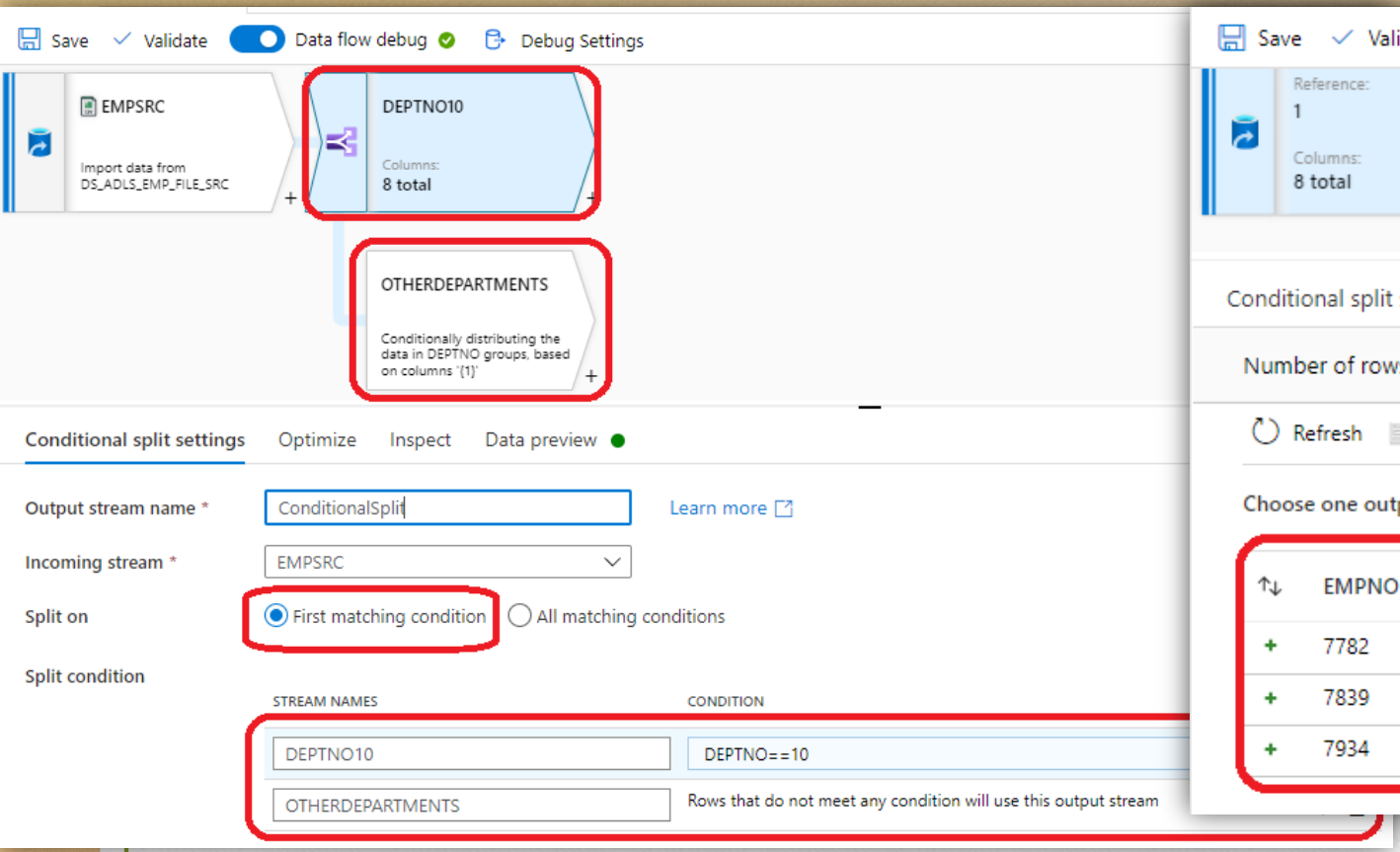
You can achieve self-join conditions in ADF Data Flow by using the Select transformation to alias an existing stream. First, create a "New Branch" from a stream, then add a Select to alias the entire original stream.

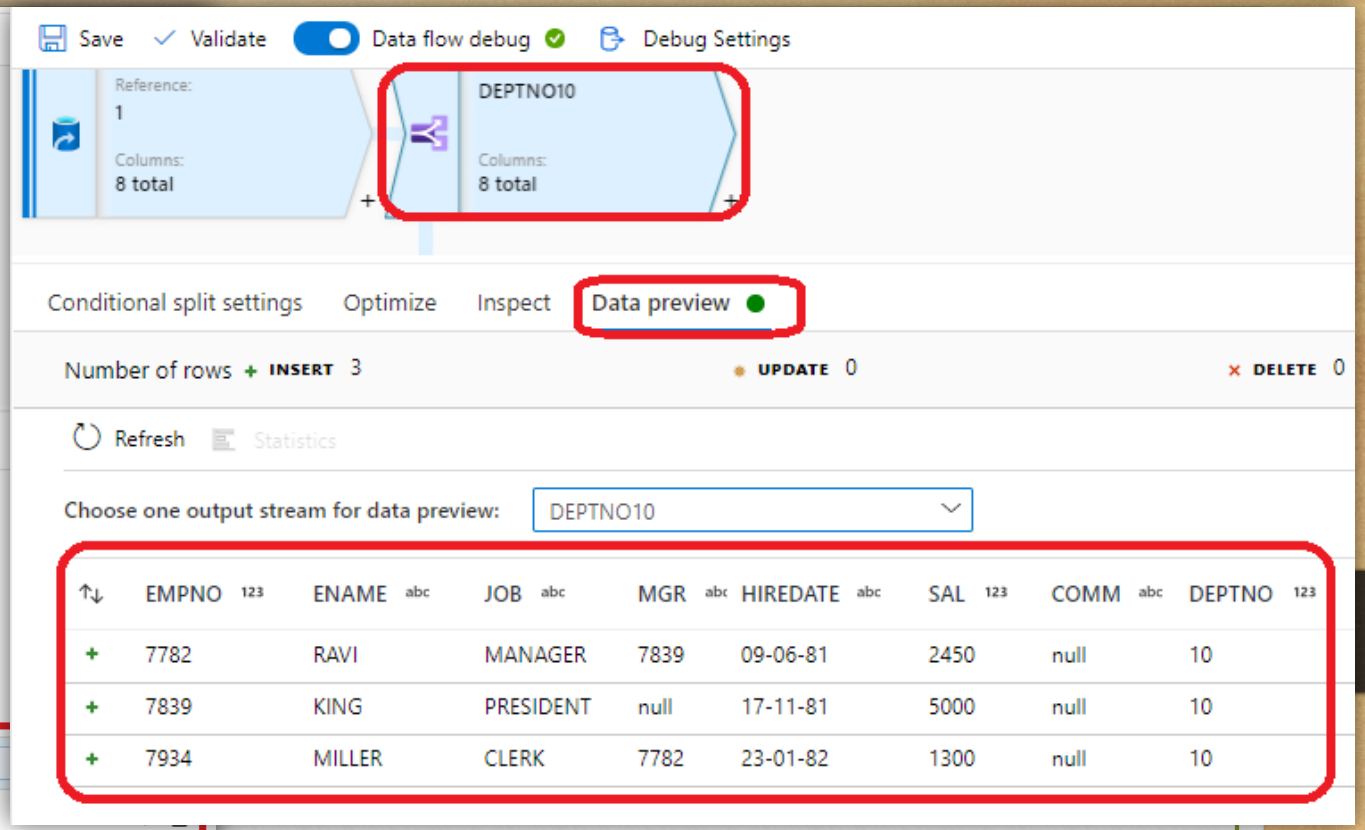
In the above diagram, the Select transform is at the top. All it's doing is aliasing the original stream to "OrigSourceBatting". In the higlighted Join transform below it you can see that we use this Select alias stream as the right-hand join, allowing us to reference the same key in both the Left & Right side of the Inner Join.

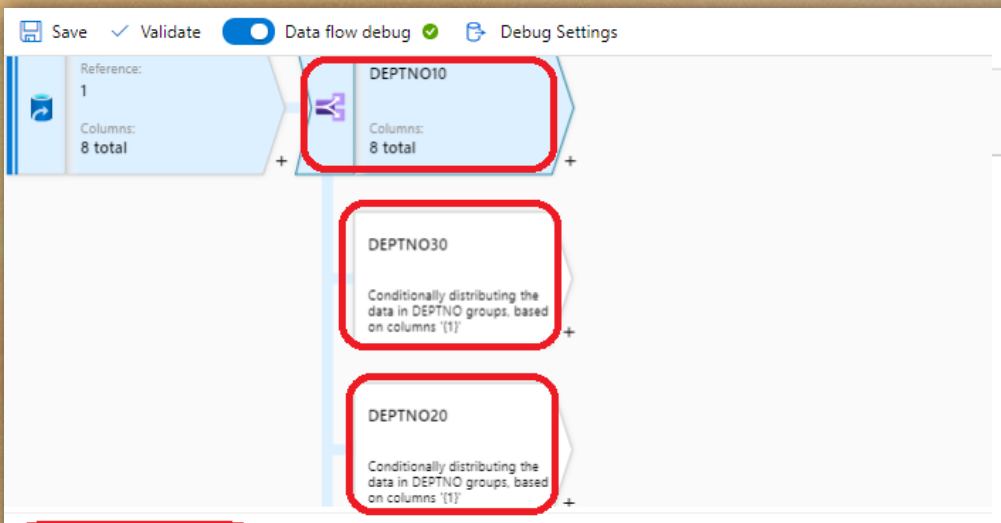
Conditional Split

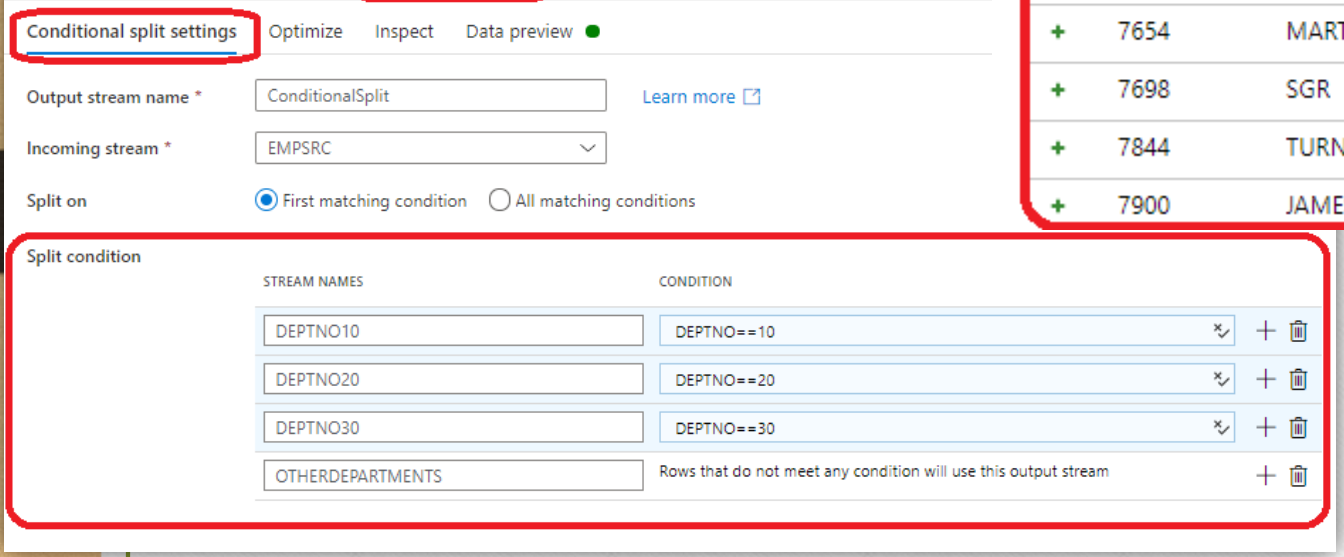
The Conditional Split transformation can route data rows to different streams depending on the content of the data. The implementation of the Conditional Split transformation is similar to a CASE decision structure in a programming language. The transformation evaluates expressions, and based on the results, directs the data row to the specified stream. This transformation also provides a default output, so that if a row matches no expression it is directed to the default output.

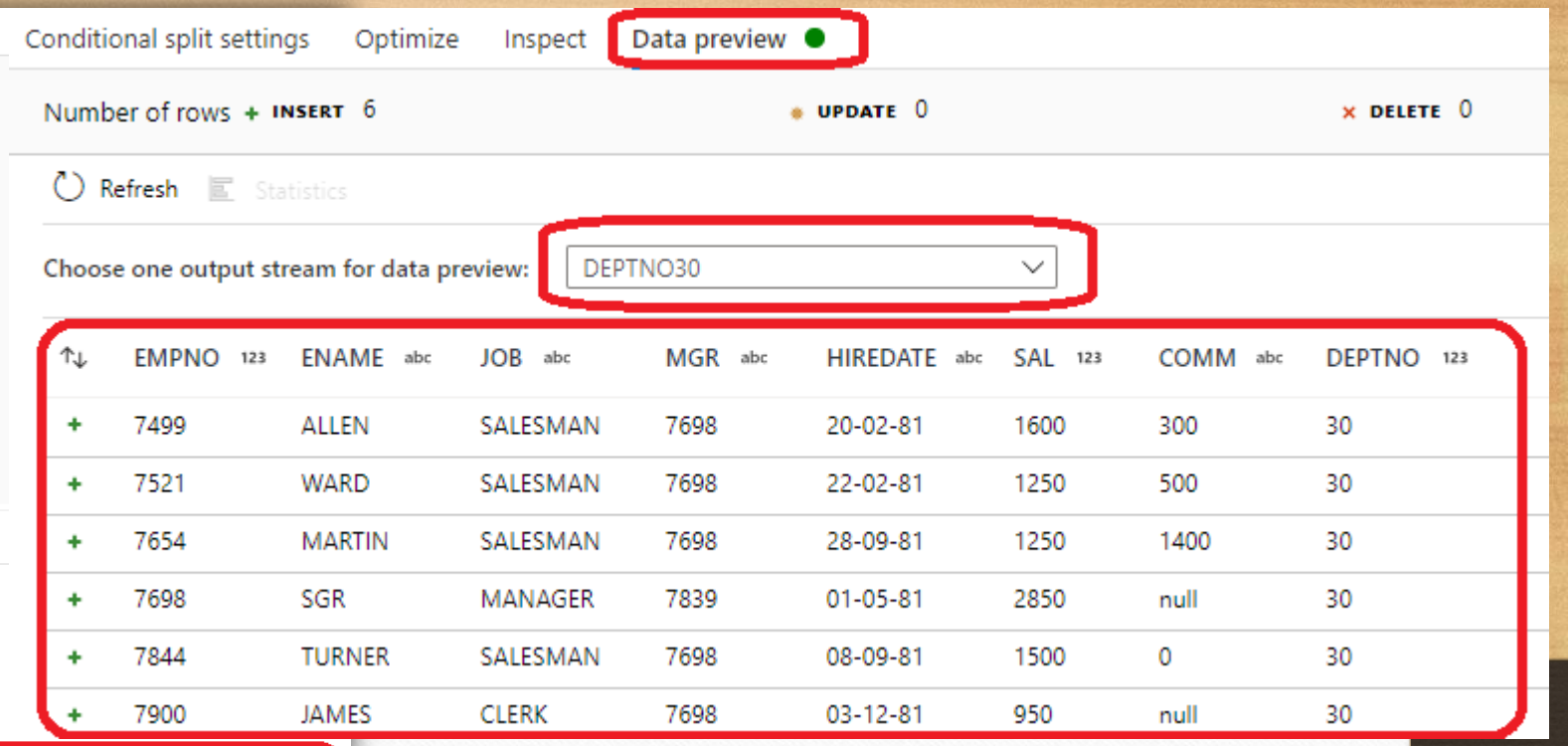
To add additional conditions, select "Add Stream" in the bottom configuration pane and click in the Expression Builder text box to build your expression.


Save
Validate
Data flow debug
Debug Settings
EMPSRC
Import data from DS_ADLS_EMP_FILE_SRC
DEPTNO10
Columns:
8 total
OTHERDEPARTMENTS
Conditionally distributing the data in DEPTNO groups, based on columns '(1)'
Conditional split settings
Optimize
Inspect
Data preview
Output stream name *
ConditionalSplit
Learn more
Incoming stream *
EMPSRC
Split on
First matching condition
All matching conditions
Split condition
STREAM NAMES
CONDITION
DEPTNO10
DEPTNO==10
OTHERDEPARTMENTS
Rows that do not meet any condition will use this output stream


Save
Validate
Data flow debug
Debug Settings
Reference:
1
Columns:
8 total
DEPTNO10
Columns:
8 total
Conditional split settings
Optimize
Inspect
Data preview
Number of rows
INSERT 3
UPDATE 0
DELETE 0
Refresh
Statistics
Choose one output stream for data preview:
DEPTNO10
EMPNO 123
ENAME abc
JOB abc
MGR abc
HIREDATE abc
SAL 123
COMM abc
DEPTNO 123
7782 RAVI MANAGER 7839 09-06-81 2450 null 10
7839 KING PRESIDENT null 17-11-81 5000 null 10
7934 MILLER CLERK 7782 23-01-82 1300 null 10

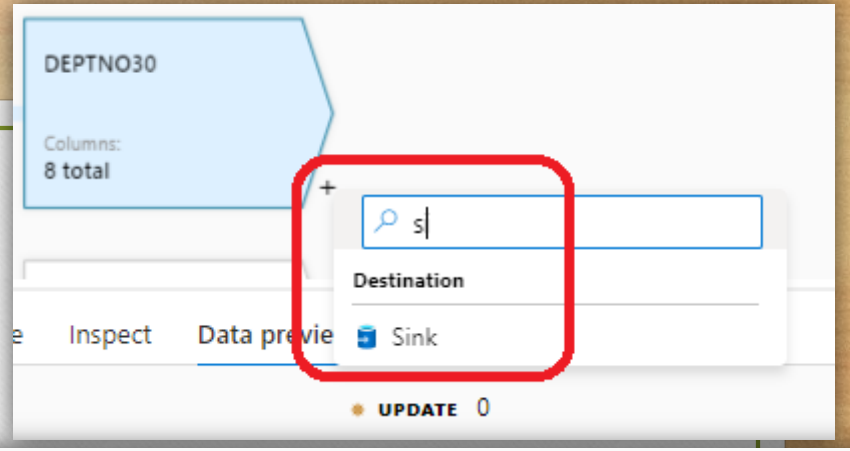


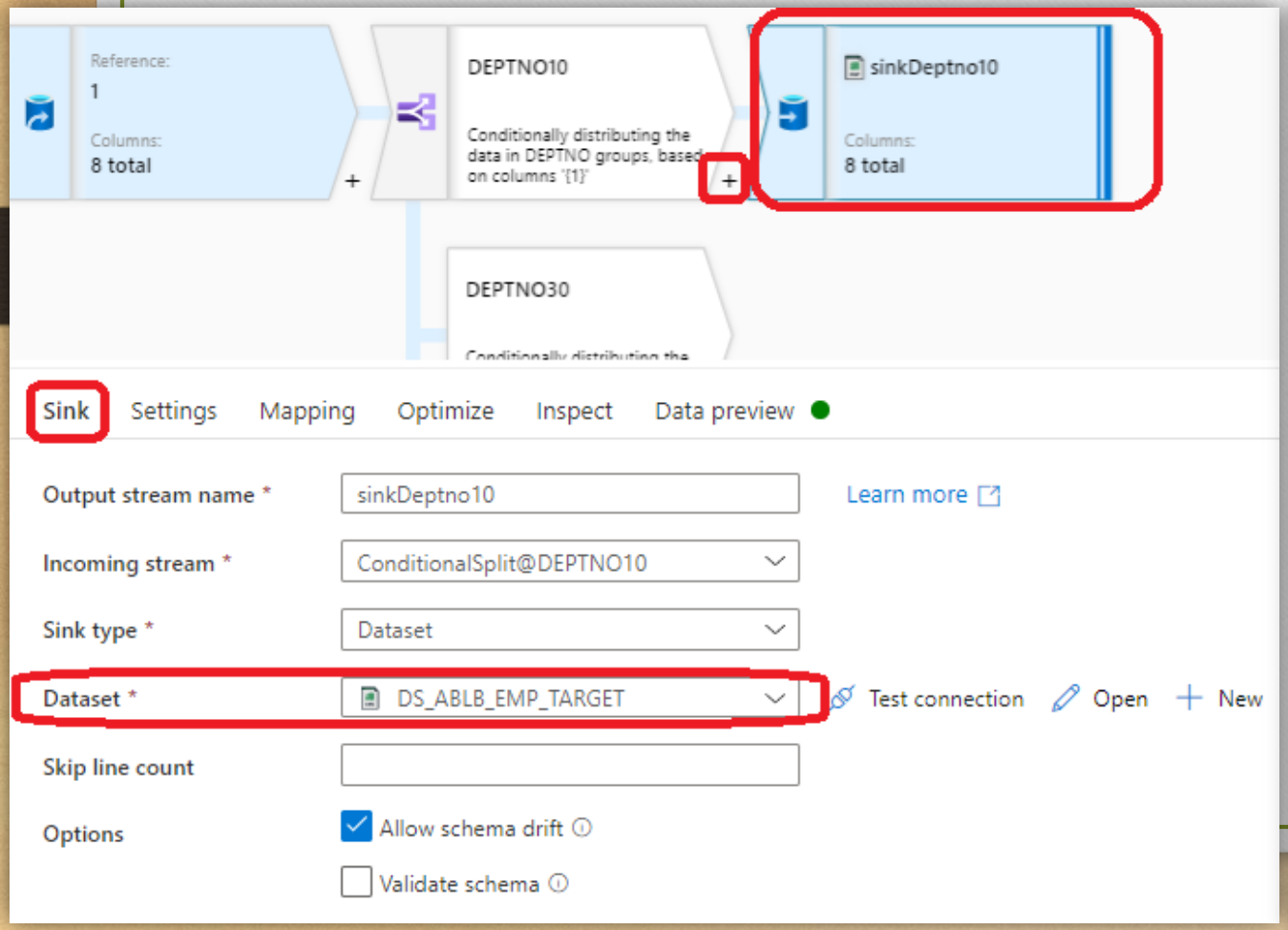


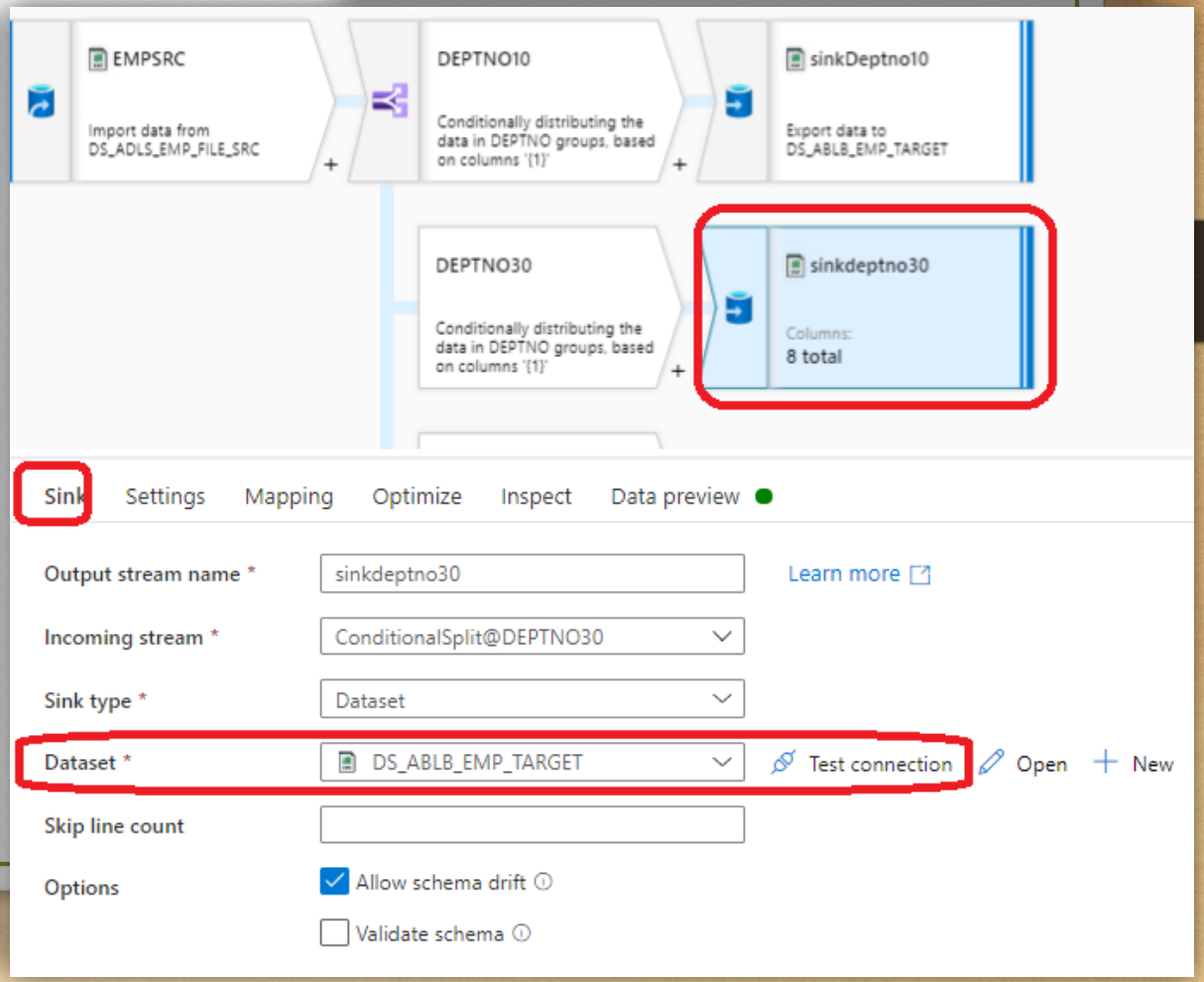


EMPNO 123	ENAME abc	JOB abc	MGR abc	HIREDATE abc	SAL 123	COMM abc	DEPTNO 123
7499	ALLEN	SALESMAN	7698	20-02-81	1600	300	30
7521	WARD	SALESMAN	7698	22-02-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-09-81	1250	1400	30
7698	SGR	MANAGER	7839	01-05-81	2850	null	30
7844	TURNER	SALESMAN	7698	08-09-81	1500	0	30
7900	JAMES	CLERK	7698	03-12-81	950	null	30

Each Conditional Split Transformation can add to multiple Targets. Using **Sink Transformation** we can transfer to target Azure Blob File.







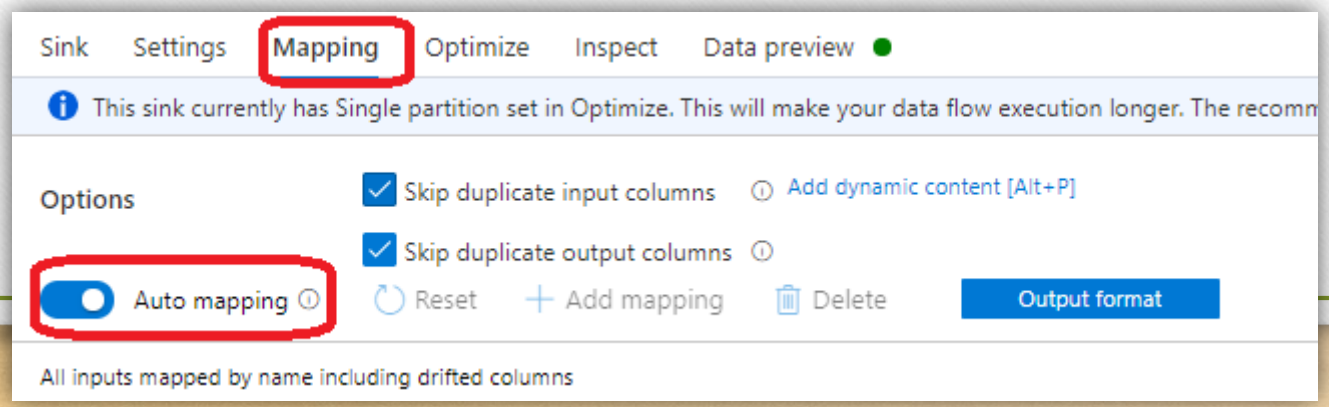
Sink Transformation

At the completion of your data flow transformation, you can **sink** your transformed data into a destination dataset. In the Sink transformation, you can choose the dataset definition that you wish to use for the destination output data. Data Flow debug mode does not require a sink. No data is written and no files are moved or deleted in Data Flow debug mode. You must execute your data flow using the Exec Data Flow activity for the Sink to execute. You are required to have at least 1 Sink in order to publish your data flow for pipeline execution.

A common practice to account for changing incoming data and to account for schema drift is to sink the output data to a folder without a defined schema in the output dataset. You can additionally account for all column changes in your sources by selecting "Allow Schema Drift" at the Source and then auto-map all fields in the Sink.

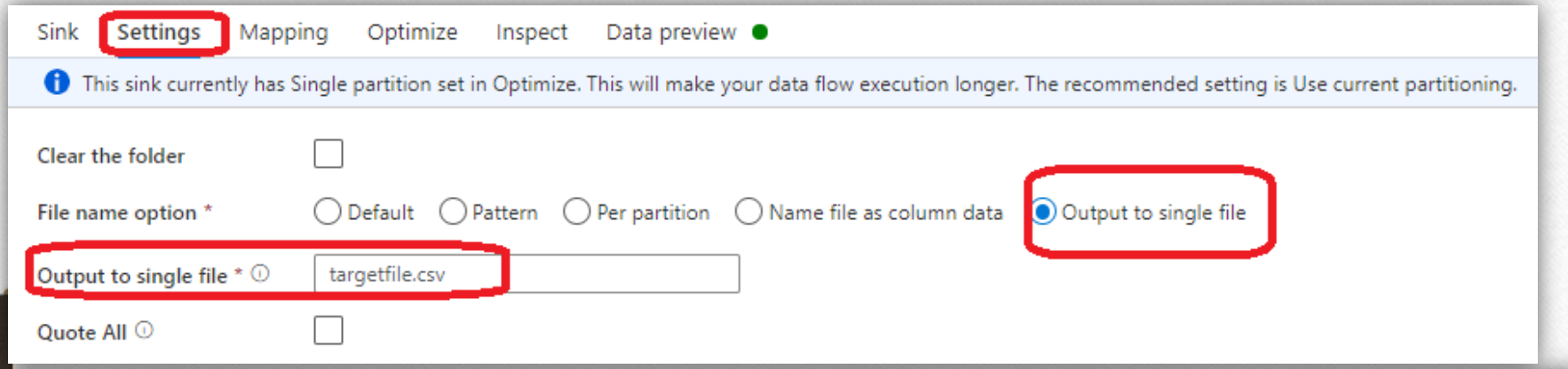
You can choose to overwrite, append, or fail the data flow when sinking to a dataset.

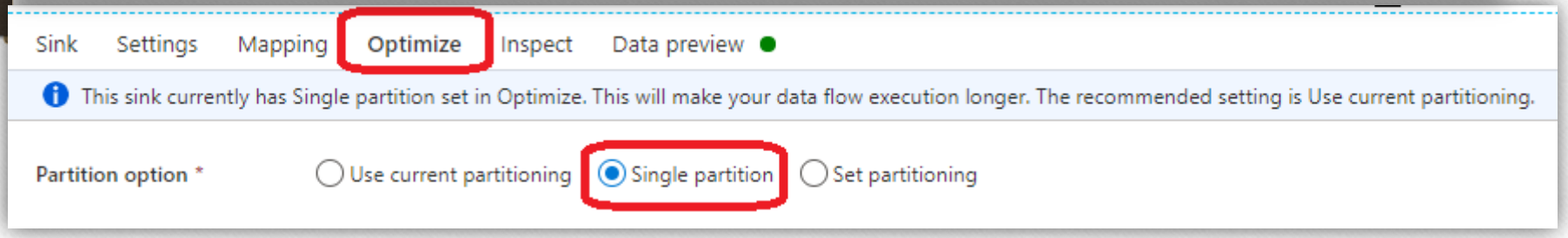
You can also choose "automap" to simply sink all incoming fields. If you wish to choose the fields that you want to sink to the destination, or if you would like to change the names of the fields at the destination, choose "Off" for "automap" and then click on the Mapping tab to map output fields:



Output to single File

For Azure Storage Blob or Data Lake sink types, you will output the transformed data into a folder. Spark will generate partitioned output data files based on the partitioning scheme being used in the Sink transform. You can set the partitioning scheme by dicking on the "Optimize" tab. If you would like ADF to merge your output into a single file, click on the "Single Partition" radio button.





Data Lake Folders

When Sinking your data transformations to Azure Blob Store or ADLS, choose a data lake folder as your destination folder path, not a file. ADF Data Flow will generate the output files for you in that folder.

Azure Blob Folders

When sinking your data to Azure Blob Store datasets, make sure to choose a blob folder inside of a container, i.e.: container/folder. Do not land your data direct lying in a container, create an output folder inside your container.

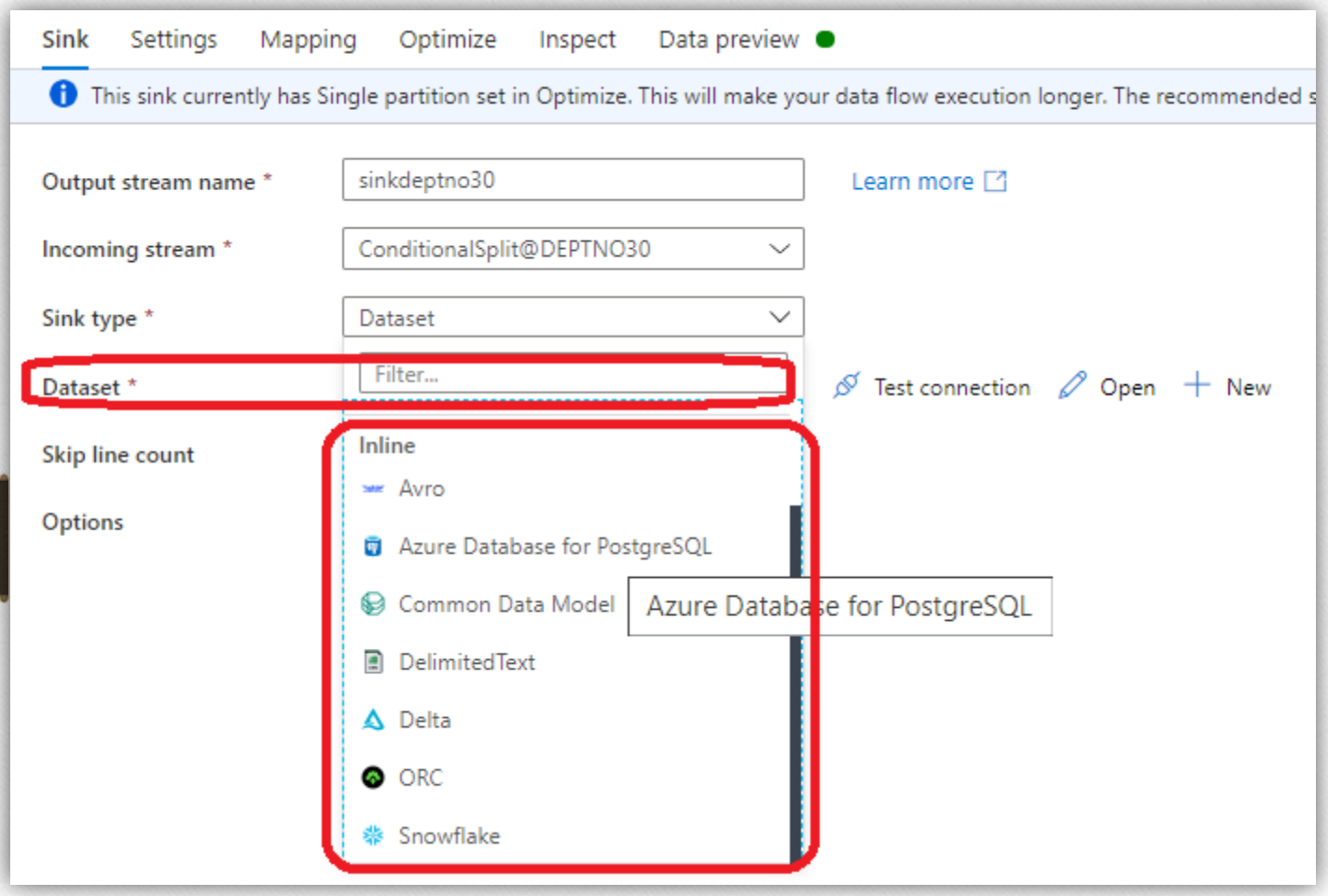
Azure SQL Data Warehouse and SQL Database Sink Datasets

If you prefer to sink your transformed data directly into Azure SQL DW or Azure SQL DB instead of the Lake approach of landing transformed data into Blob or ADLS first, you can use Sink Datasets for Data Flow that are Azure SQL DB or DW. This will allow you to land your transformed data directly into Azure SQL DW within Data Flow without the need of adding a Copy Activity in your pipeline.

Start by creating an ADW dataset, just as you would for any other ADF pipeline, with a Linked Service that includes your ADW credentials and choose the database that you wish to connect to. In the table name, either select an existing table or type in the name of the table that you would like Data Flow to auto-create for you. A new table will be generated in the target database using the incoming metadata schema.

NOTE: At this time, we are not supporting SQL Server square brackets " [ ] ", so please use the "Edit" link on the table name field and remove the brackets

Azure SQL DW Datasets require staging locations to be specified because ADF uses Polybase behind the scenes. You'll select the Storage account you wish to use for staging the data for the Polybase load into ADW. The path field is of the format: "containername/foldername".


Sink
Settings
Mapping
Optimize
Inspect
Data preview
This sink currently has Single partition set in Optimize. This will make your data flow execution longer. The recommended s
Output stream name *
sinkdeptno30
Learn more
Incoming stream *
ConditionalSplit@DEPTNO30
Sink type *
Dataset
Dataset *
Filter...
Test connection
Open
New
Skip line count
Options
Inline
Avro
Azure Database for PostgreSQL
Common Data Model
DelimitedText
Delta
ORC
Snowflake
Azure Database for PostgreSQL

Save Policy
Overwrite will truncate the table if it exists, then recreate it and load the data. Append will simply insert the new rows. If the table from the Dataset table name does not exist at all in the target ADW, Data Flow will create the table, then load the data.

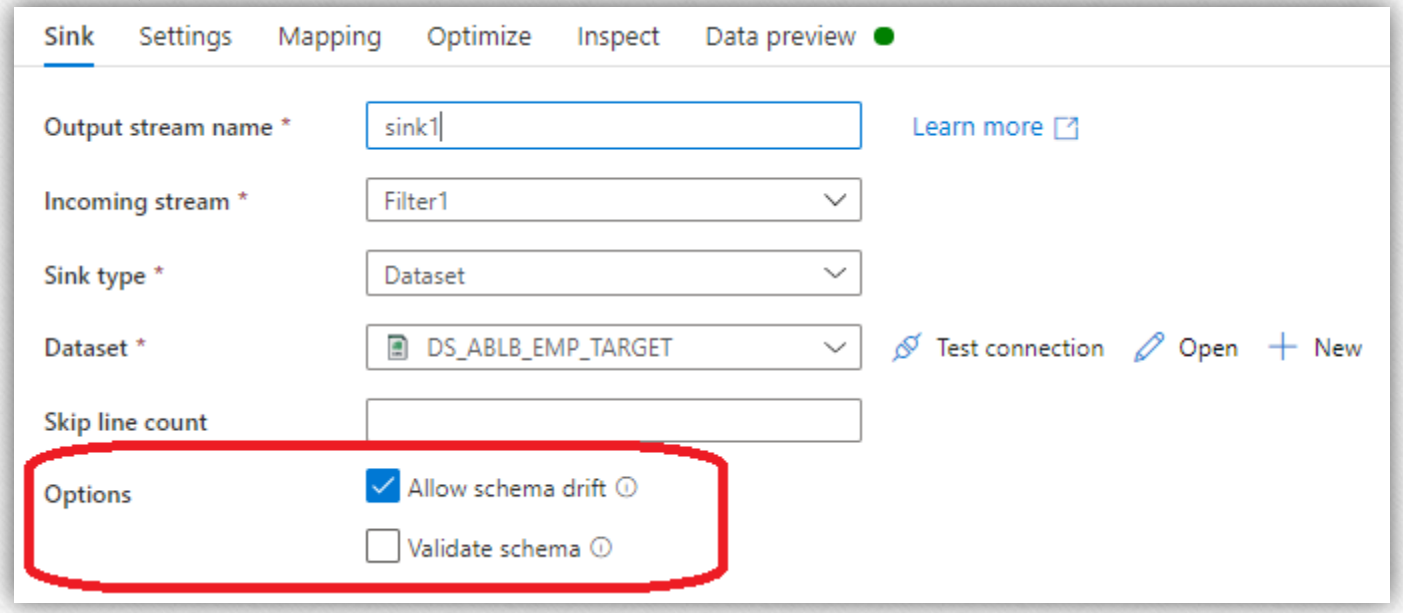
Field Mapping
On the Mapping tab of your Sink transformation, you can map the incoming (left side) columns to the destination (right side). When you sink data flows to files, ADF will always write new files to a folder. When you map to a database dataset, you can choose to either generate a new table with this schema (set Save Policy to "overwrite") or insert new rows to an existing table and map the fields to the existing schema.

You can use multi-select in the mapping table to Link multiple columns with one click, Delink multiple columns or map multiple rows to the same column name.

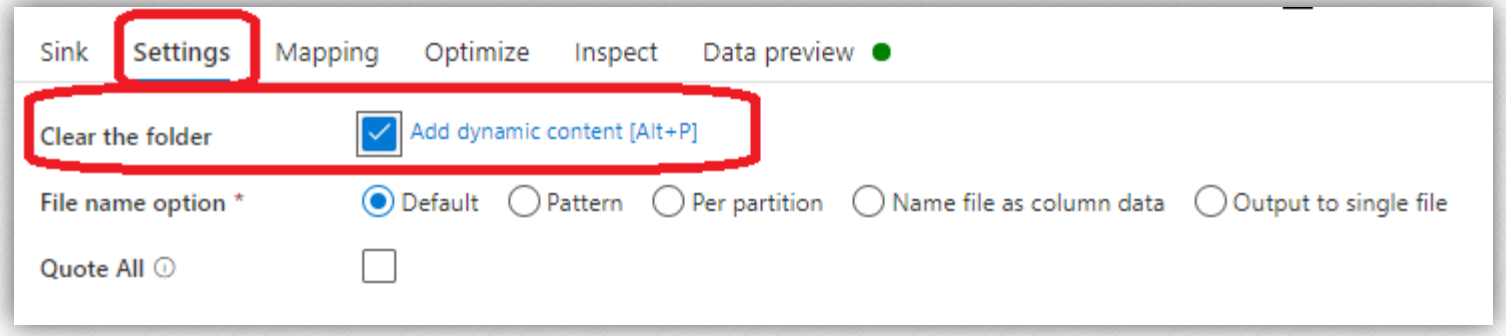
If you'd like to reset your columns mappings, press the "Remap" button to reset the mappings.

Max Concurrent Connections
You can set the maximum concurrent connections in the Sink transformation when writing your data to an Azure database connection.

Allow Schema Drift and Validate Schema options are now available in **Sink**. This will allow you to instruct ADF to either fully accept flexible schema definitions (**Schema Drift**) or fail the Sink if the **schema changes** (Validate Schema).

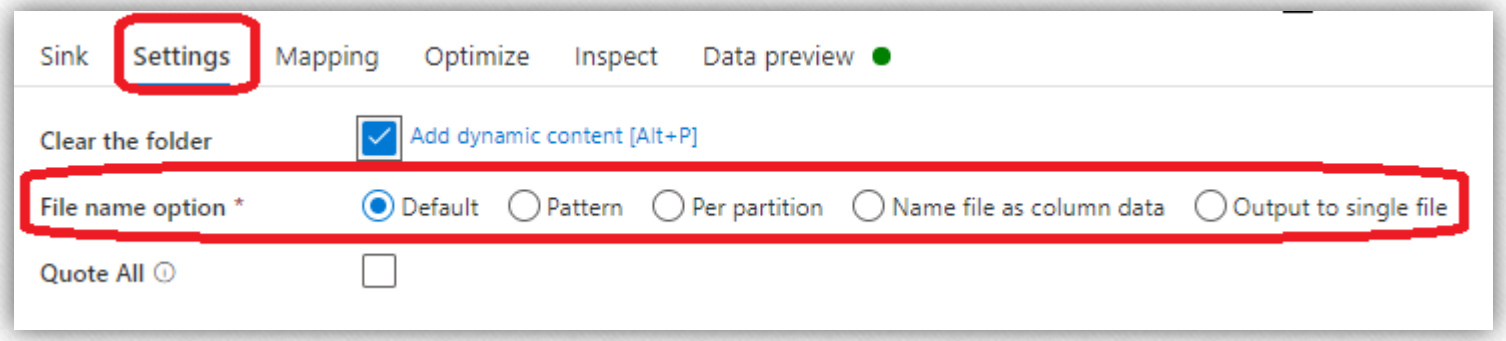


Clear the Folder. ADF will truncate the sink folder contents before writing the destination files in that target folder.



File name options

Default: Allow Spark to name files based on PART defaults
Pattern: Enter a name for your output files
Per partition: Enter a file name per partition
As data in column: Set the output file to the value of a column
NOTE: File operations will only execute when you are running the Execute Data Flow activity, not while in Data Flow Debug mode

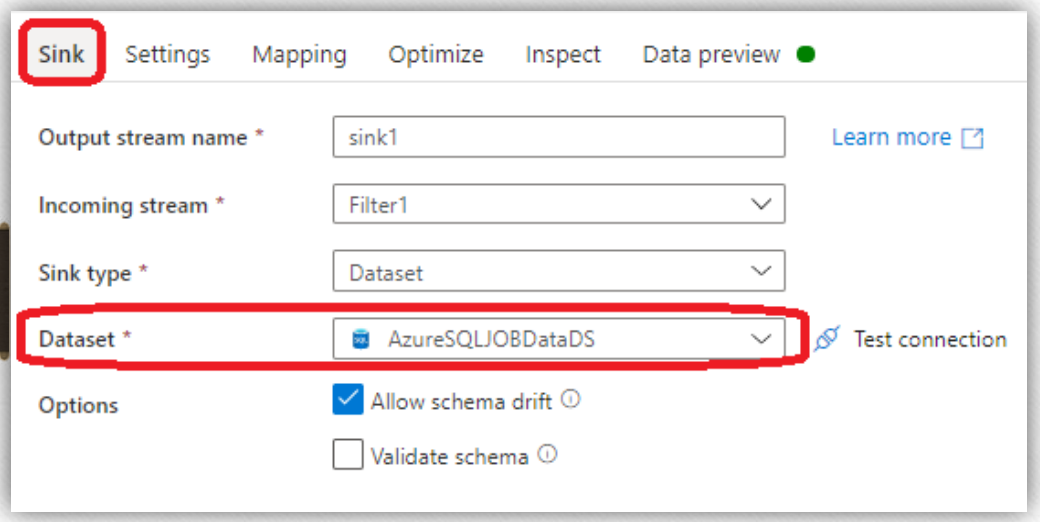


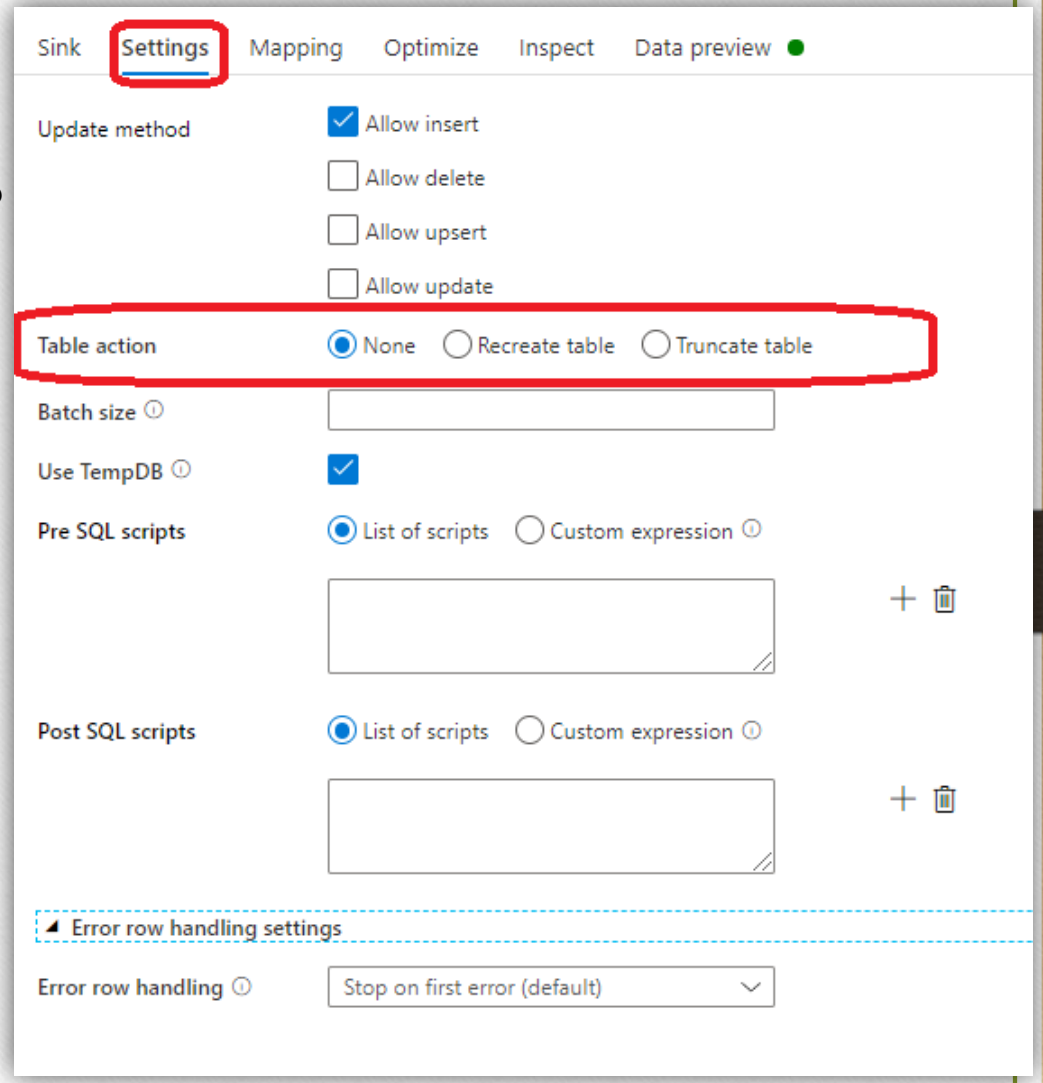
With the SQL sink types, you can set:

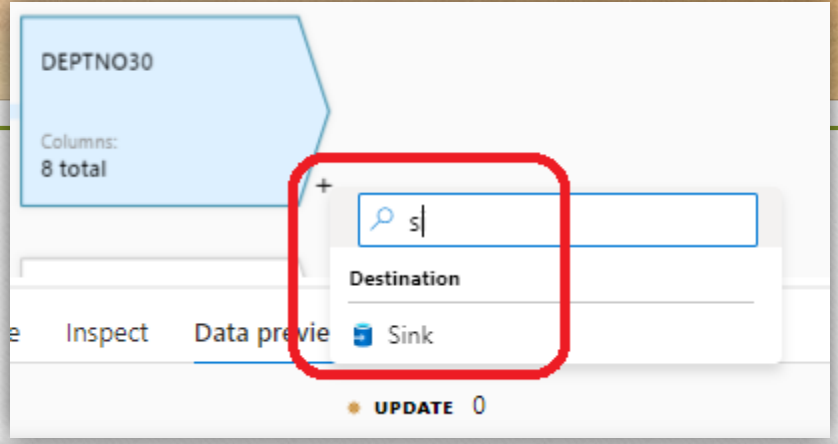
Truncate table

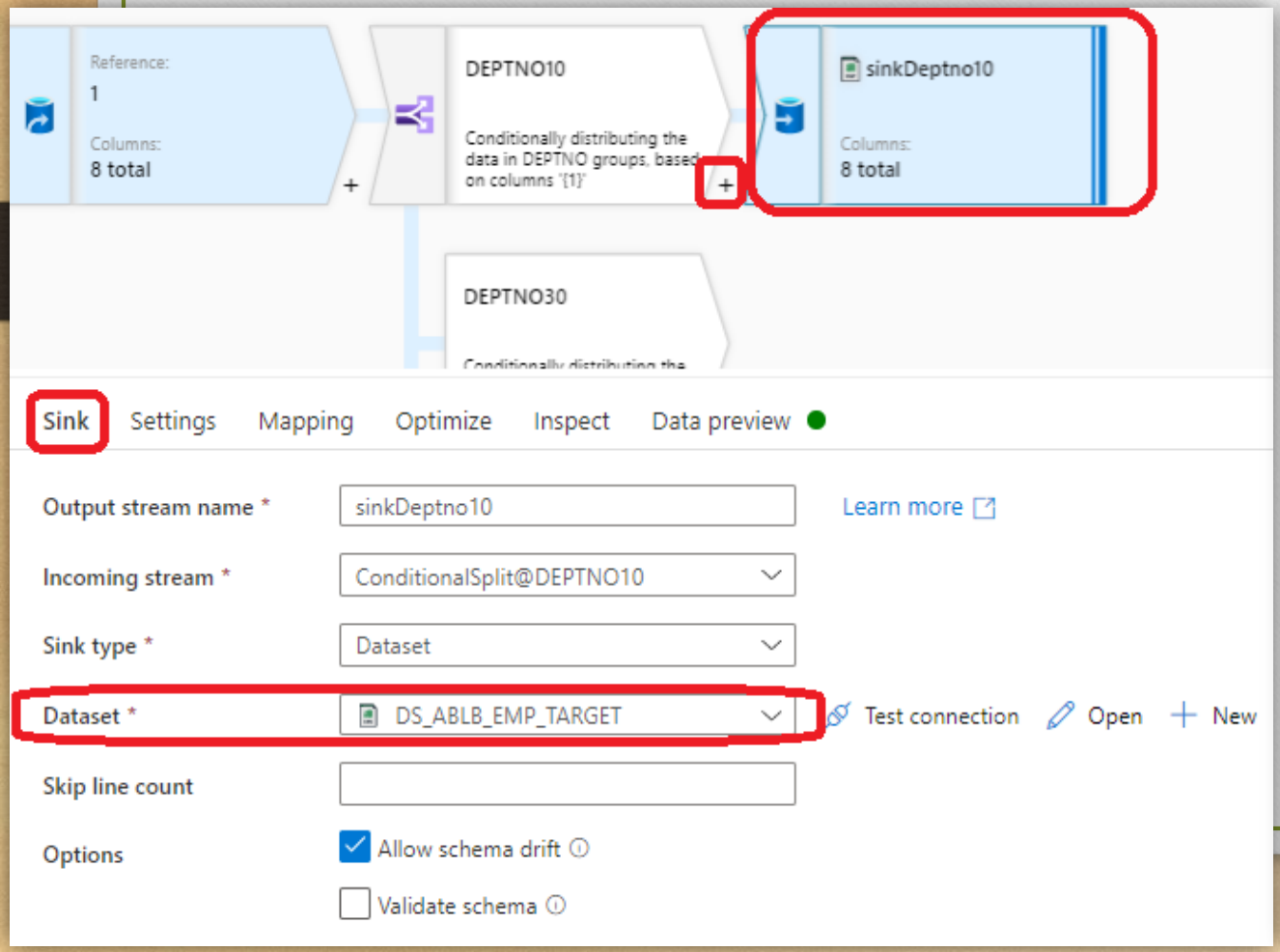
Recreate table (performs drop/create)

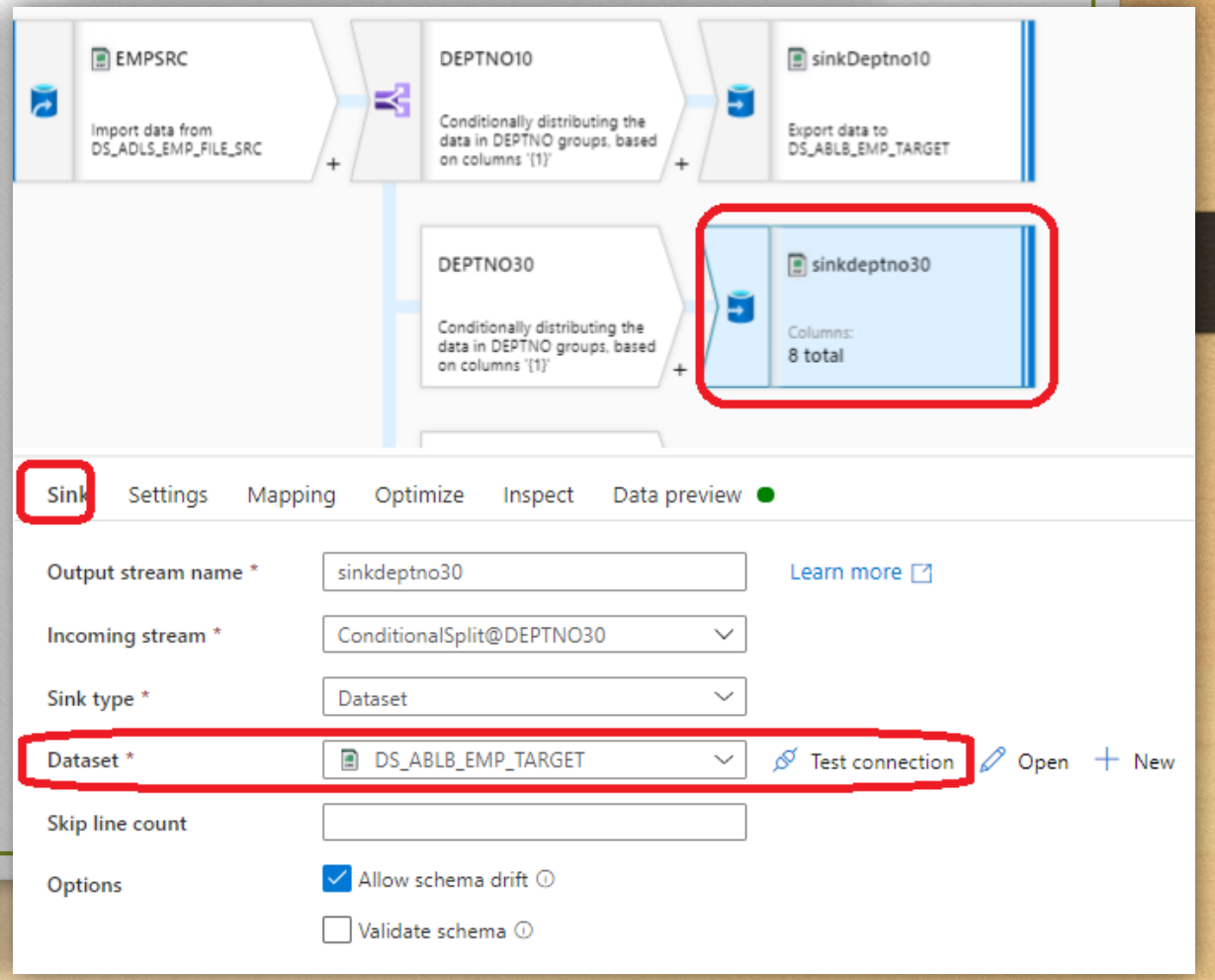
Batch size for large data loads. Enter a number to bucket writes into chunks.



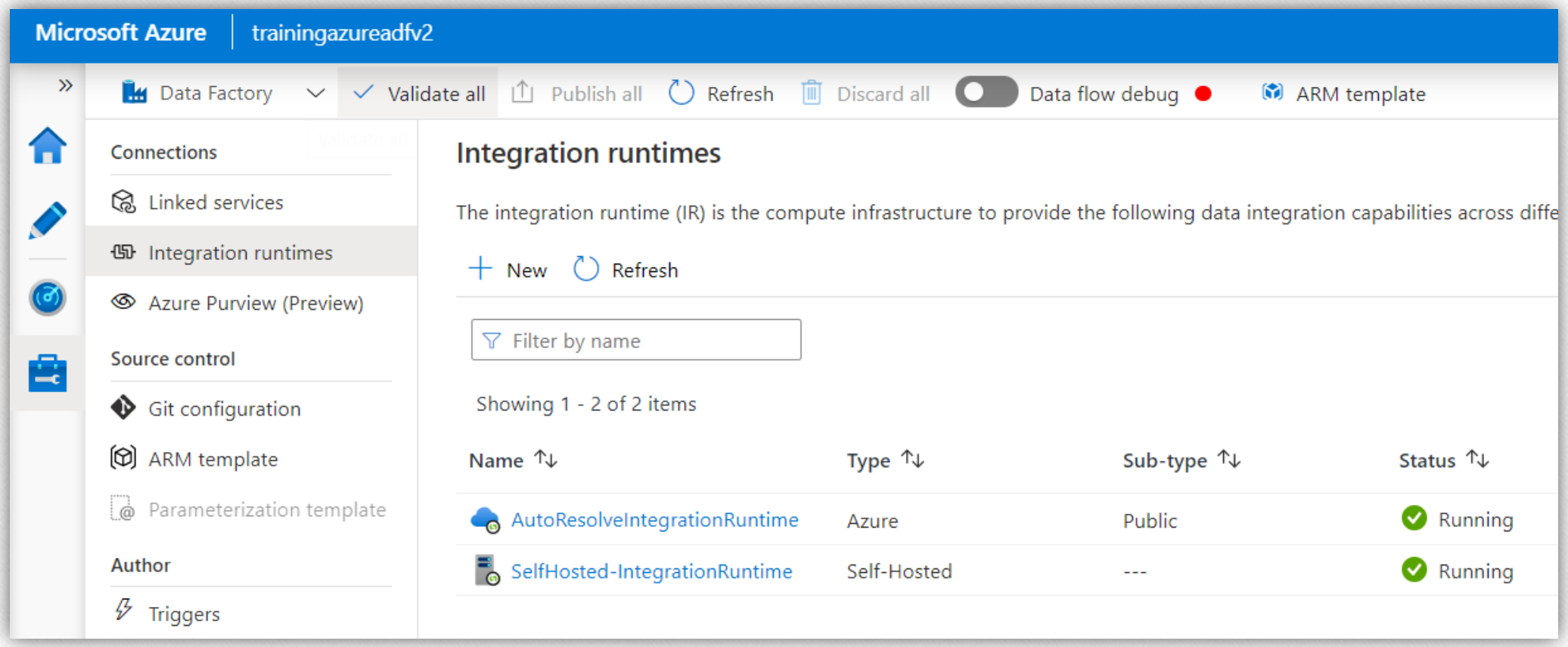



DEPTNO30
Columns:
8 total
s
Destination
Sink
Inspect
Data previe
UPDATE 0


Reference:
1
Columns:
8 total
DEPTNO10
Conditionally distributing the data in DEPTNO groups, based on columns '{1}'
sinkDeptno10
Columns:
8 total
DEPTNO30
Sink
Settings
Mapping
Optimize
Inspect
Data preview
Output stream name *
sinkDeptno10
Learn more
Incoming stream *
ConditionalSplit@DEPTNO10
Sink type *
Dataset
Dataset *
DS_ABLB_EMP_TARGET
Test connection
Open
New
Skip line count
Options
Allow schema drift
Validate schema

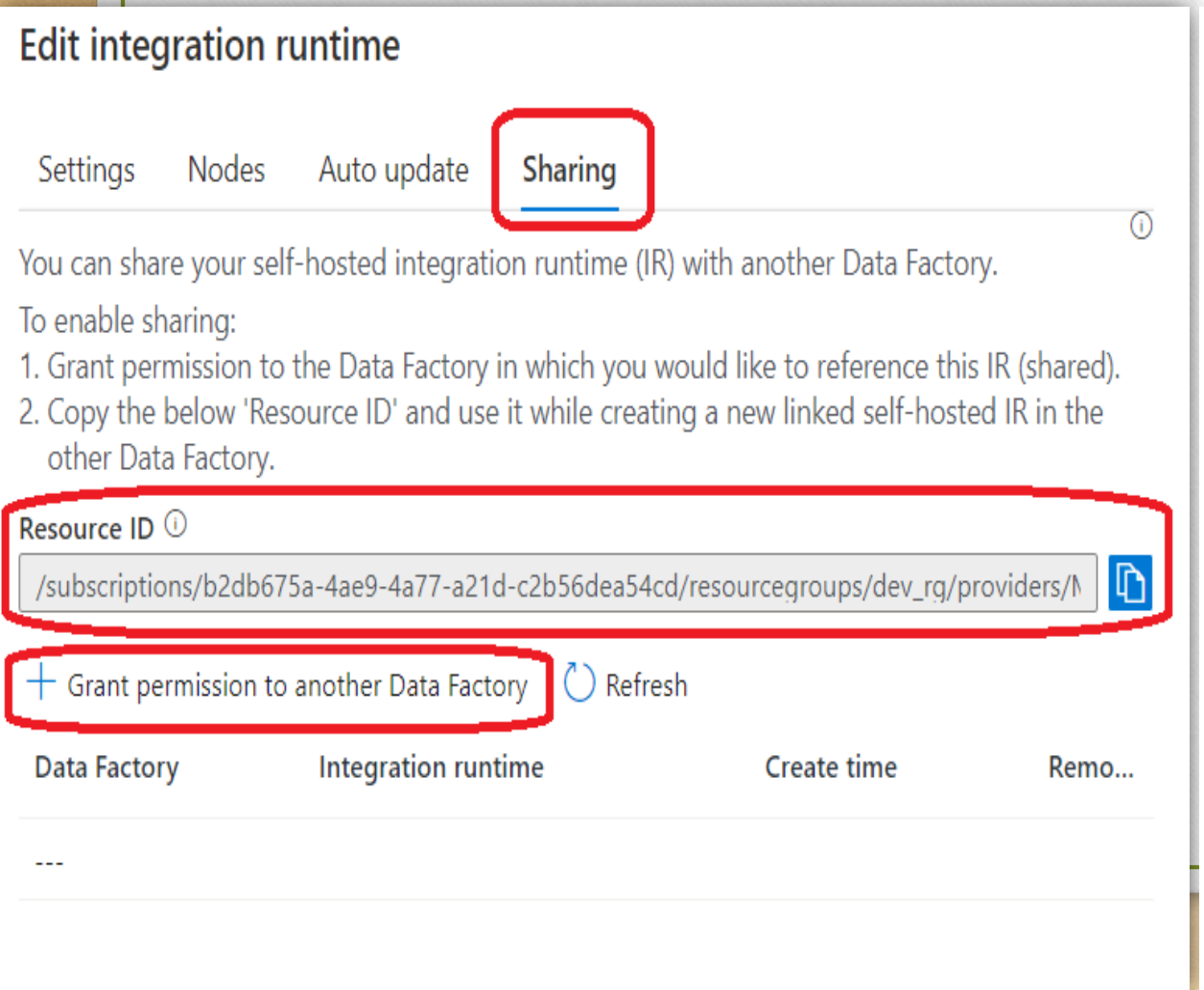

EMPSRC
Import data from DS_ADLS_EMP_FILE_SRC
DEPTNO10
Conditionally distributing the data in DEPTNO groups, based on columns '{1}'
sinkDeptno10
Export data to DS_ABLB_EMP_TARGET
DEPTNO30
Conditionally distributing the data in DEPTNO groups, based on columns '{1}'
sinkdeptno30
Columns:
8 total
Sink
Settings
Mapping
Optimize
Inspect
Data preview
Output stream name *
sinkdeptno30
Learn more
Incoming stream *
ConditionalSplit@DEPTNO30
Sink type *
Dataset
Dataset *
DS_ABLB_EMP_TARGET
Test connection
Open
New
Skip line count
Options
Allow schema drift
Validate schema

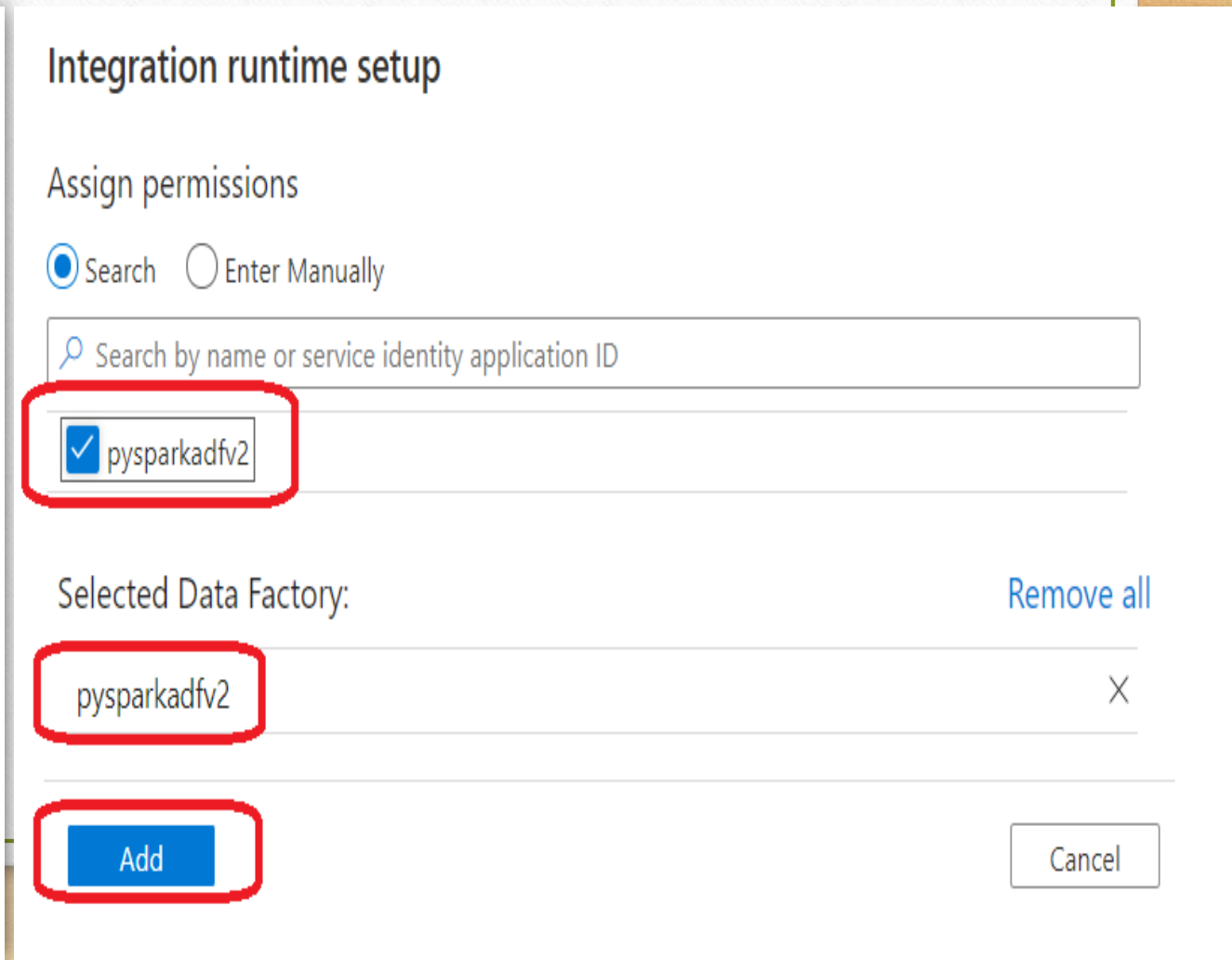
Create a shared self-hosted integration runtime in Azure Data Factory



To create a shared self-hosted IR using Azure Data Factory UI, you can take following steps:

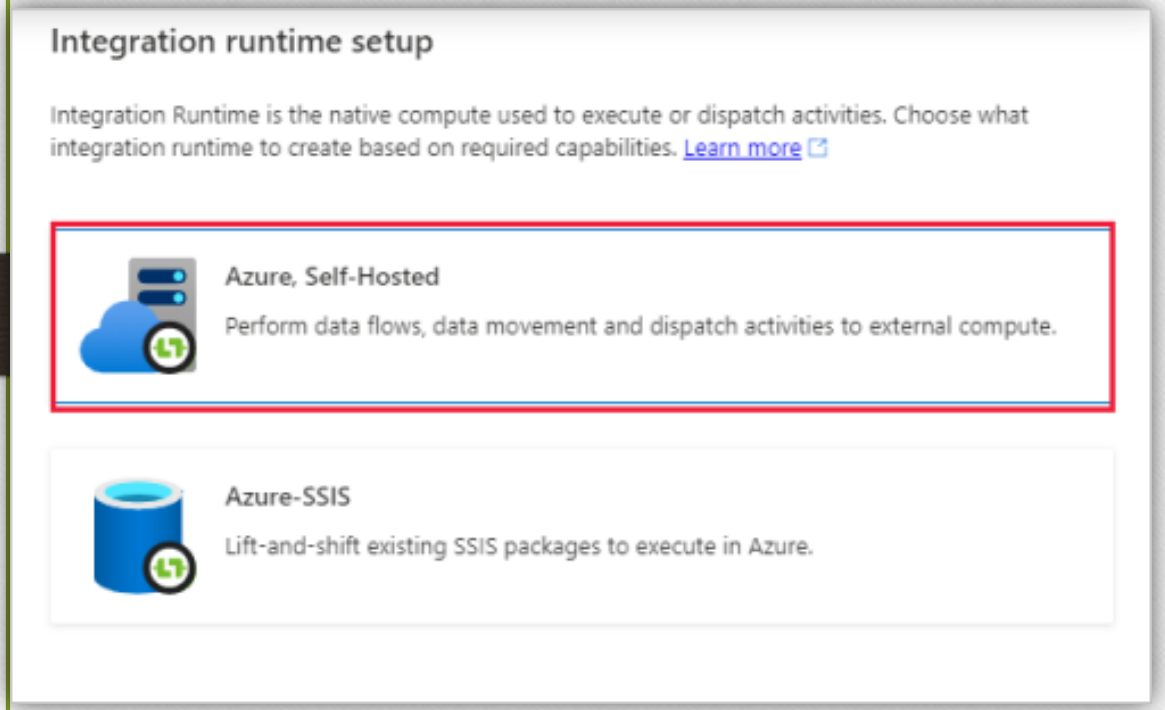
1.In the self-hosted IR to be shared, select **Grant permission to another Data factory** and in the "Integration runtime setup" page, select the Data factory in which you want to create the linked IR.

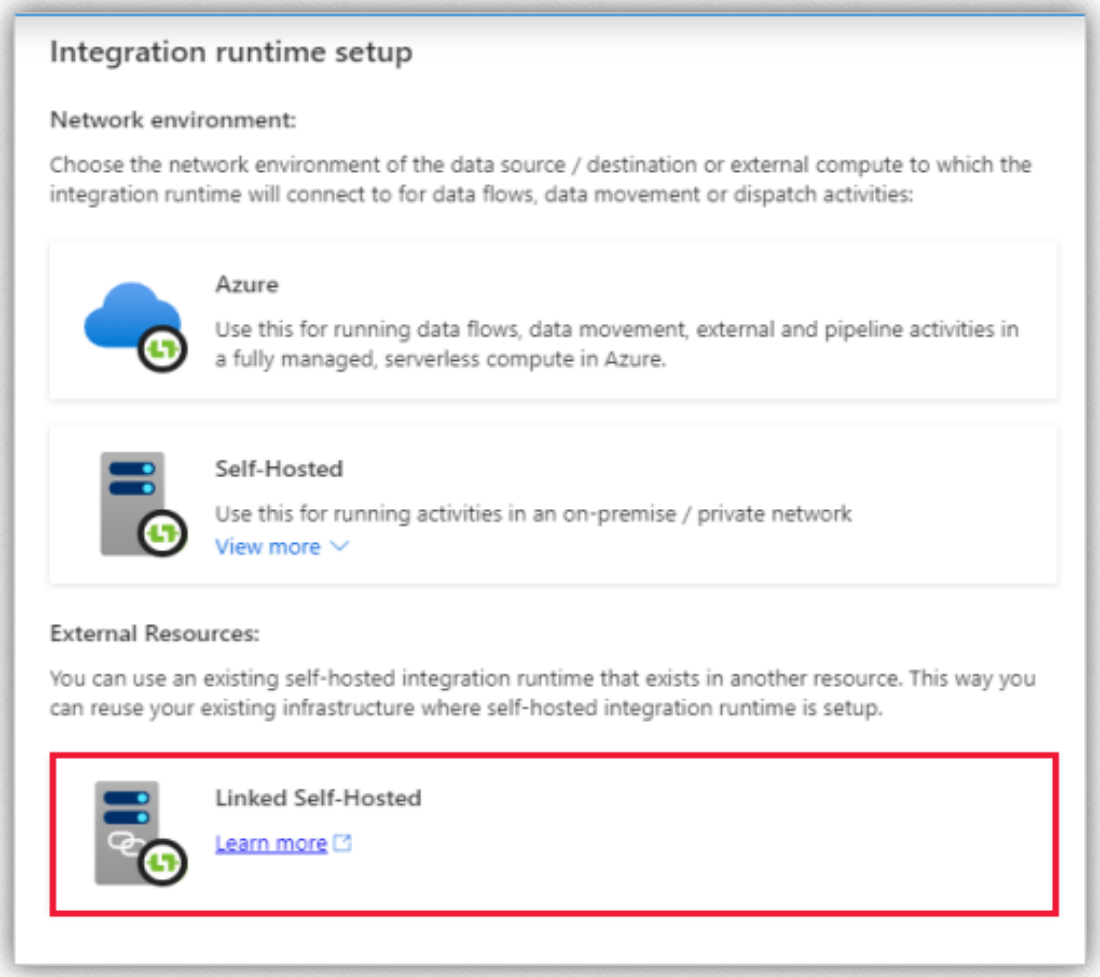




Note and copy the above "Resource ID" of the self-hosted IR to be shared.

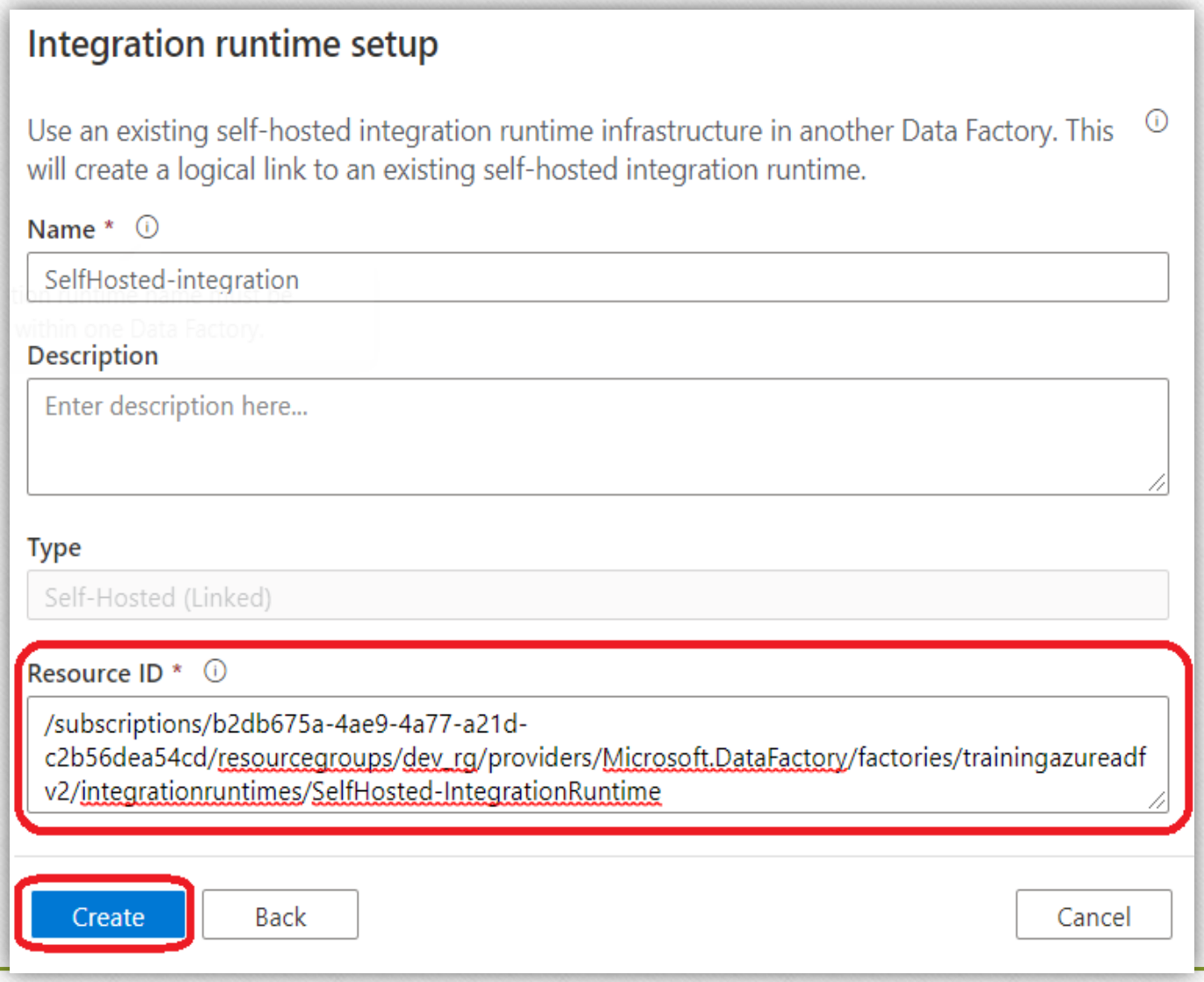
In the data factory to which the permissions were granted, create a new self-hosted IR (linked) and enter the resource ID.



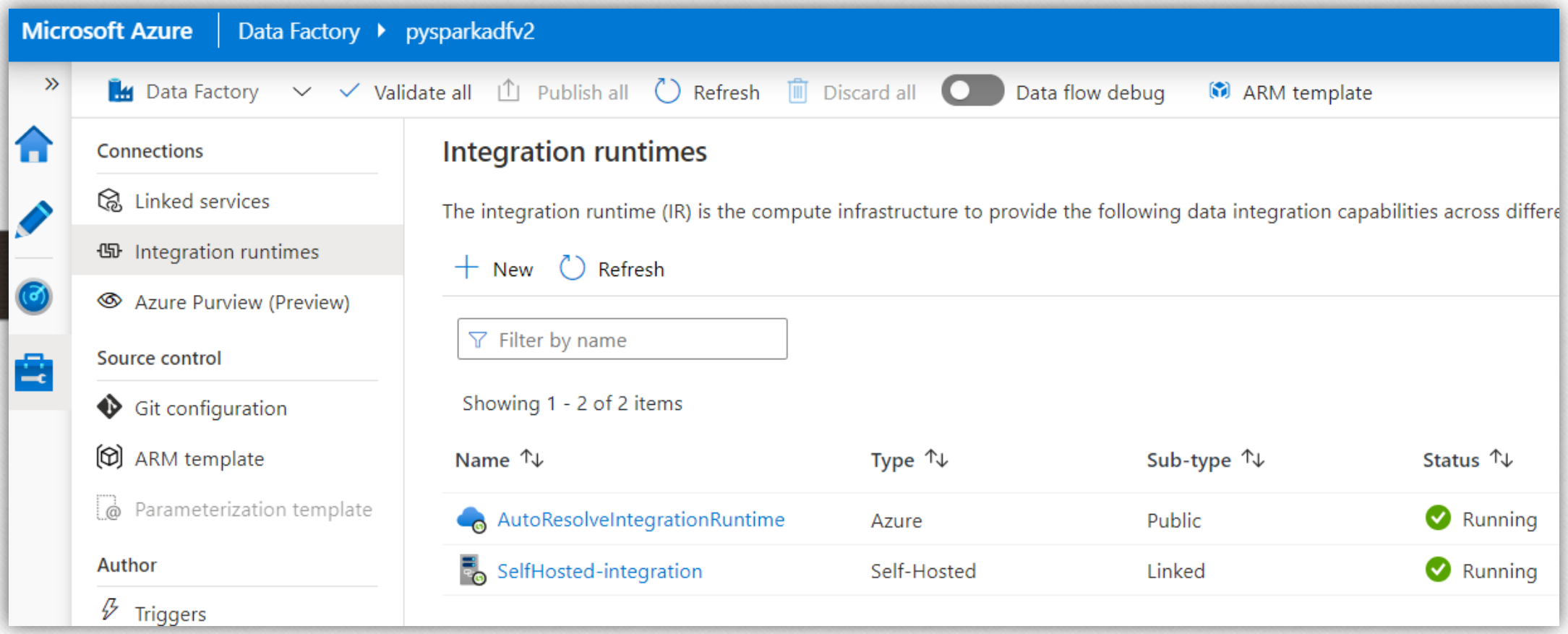


Rename Integration tuntime name as " **SelfHosted-Integration** " And Paste Resource ID in down window.
Then Create.

It will be created new Share self hosted Integration runtime in another Data factory.



Finally New **Shared Self-hosted integration** Run time is working in another datafactory.

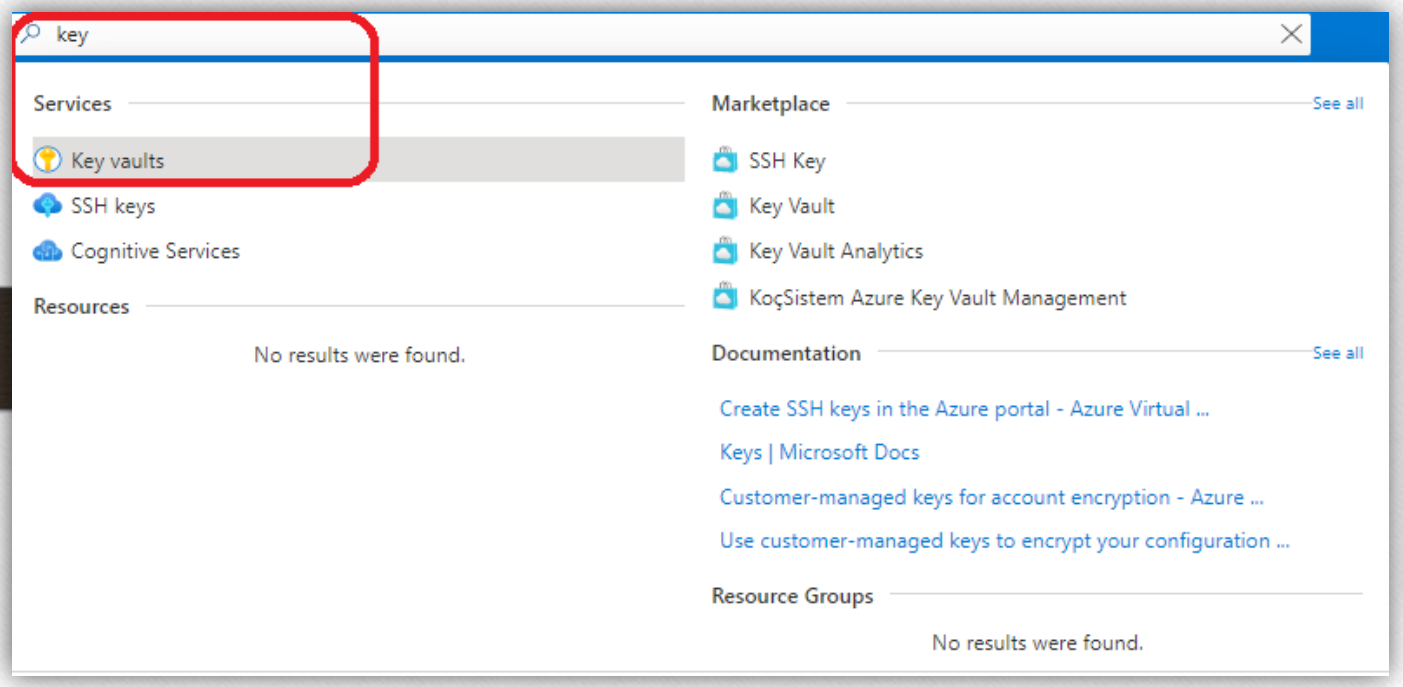


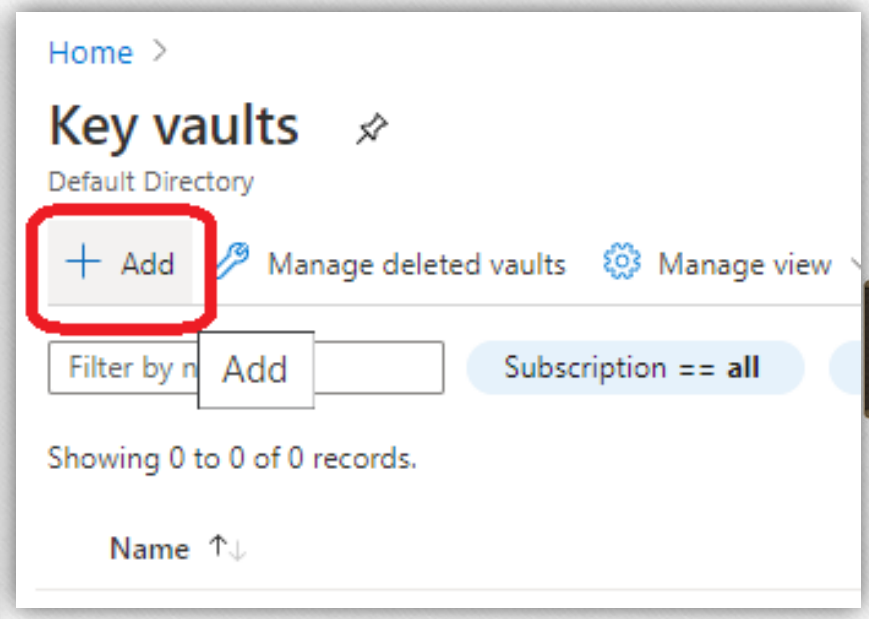
Deploy Azure Key Vault

Azure Key Vault: Centralizing storage of application secrets in Azure Key Vault allows you to control their distribution. Key Vault greatly reduces the chances that secrets may be accidentally leaked. When using Key Vault, application developers no longer need to store security information in their application. Not having to store security information in applications eliminates the need to make this information part of the code. For example, an application may need to connect to a database. Instead of storing the connection string in the app's code, you can store it securely in Key Vault

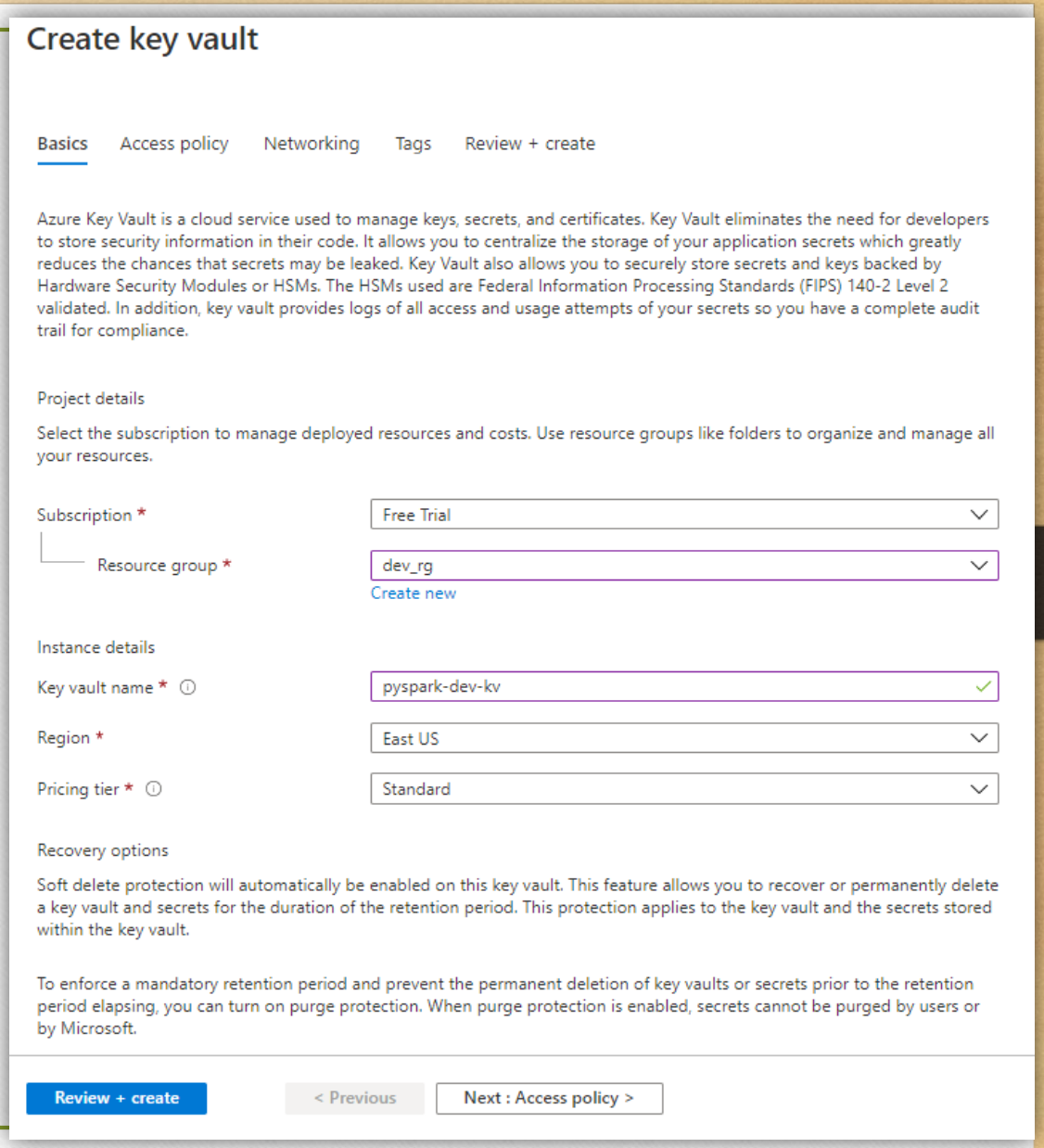
1. Search for Azure Key Vault
2. Click Add
3. Provide subscription, resource group
4. **For Key Vault Name use: adf-<your name>-dev-kv for example: adf-pyspark-dev-kv**
5. Select Location
6. Leave the rest as it is
7. Click “Review + Create” and then “Create”

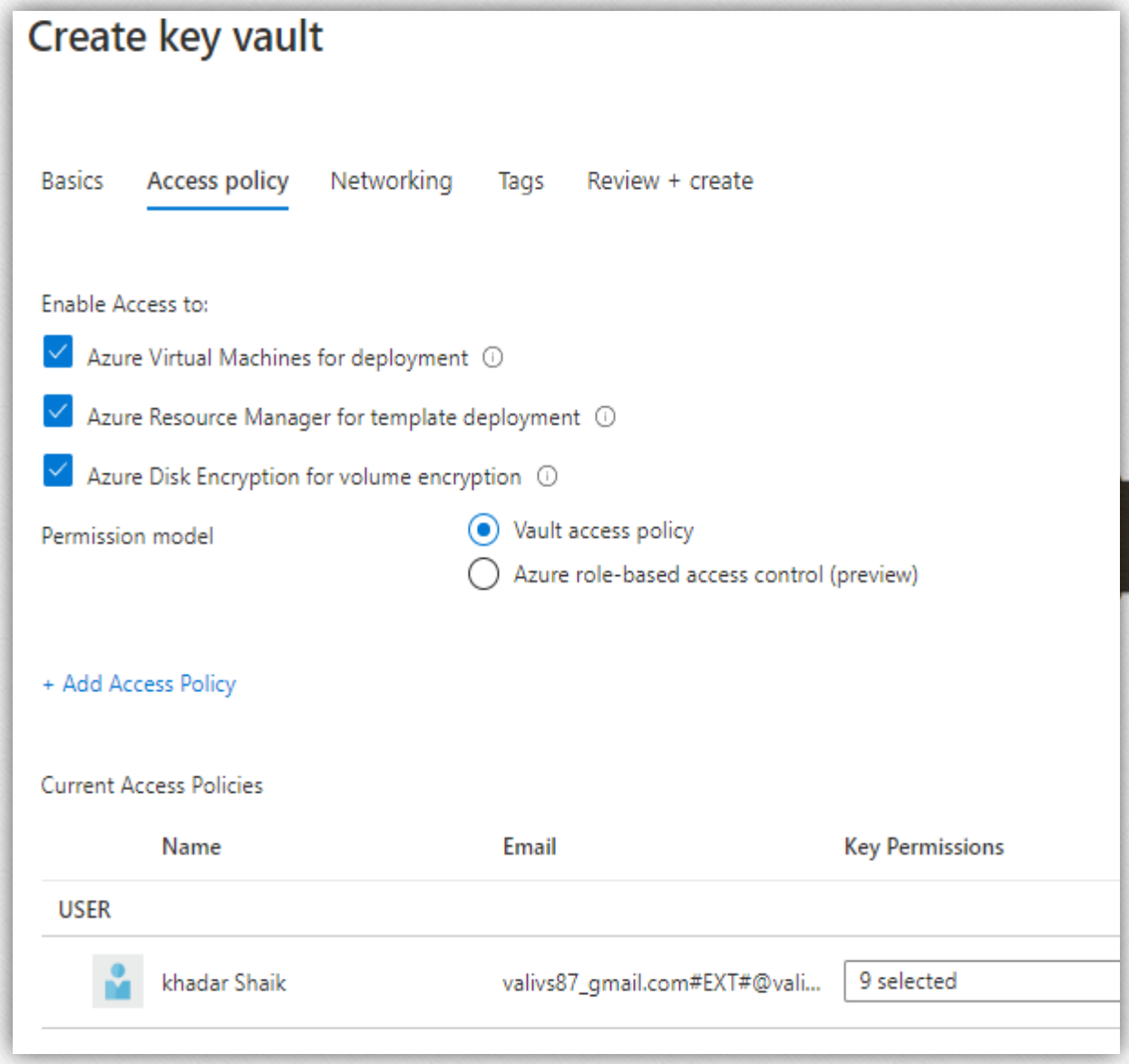
Search key vaults in search box. It will display key vaults resource and click on that.
Click on **+Add** to create new Key vaults





1.Choose a **Subscription** from the list of available subscriptions. All subscriptions that offer the Key Vault service are displayed in the drop-down list.
2.Select an existing **Resource Group**, or create a new one.
3.Select the **Pricing tier**. In the Azure Stack Development Kit (ASDK), key vaults support **Standard** SKUs only.
4.Choose one of the existing **Access policies** or create a new one. An access policy allows you to grant permissions for a user, an app, or a security group to perform operations with this vault.
5.Optionally, choose an **Advanced access policy** to enable access to features. For example: virtual machines (VMs) for deployment, Resource Manager for template deployment, and access to Azure Disk Encryption for volume encryption.
6.After you configure the settings, select **OK**, and then select **Create**. This step starts the key vault deployment.




Create key vault
Basics
Access policy
Networking
Tags
Review + create
Enable Access to:
Azure Virtual Machines for deployment
Azure Resource Manager for template deployment
Azure Disk Encryption for volume encryption
Permission model
Vault access policy
Azure role-based access control (preview)
+ Add Access Policy
Current Access Policies
Name
Email
Key Permissions
USER
khadar Shaik
valivs87_gmail.com#EXT#@vali...
9 selected

```
/*****************************************
** Creating ETL Load Status table in SQL DB...
*****************************************/
CREATE TABLE [dbo].[JOB_LIST](
[id] [varchar](100) NULL,
[jobname] [varchar](100) NULL,
[src_folderpath] [varchar](100) NULL,
[tgt_folderpath] [varchar](100) NULL,
[directory] [varchar](200) NULL,
[status] [varchar](100) NULL,
[processedon] [datetime] NULL
)
GO

/*********************************************************
**  INSERT SCRIPT FOR LIST OF JOBS
*********************************************************/

insert into [dbo].[JOB_LIST] values(1,'channels_dim','/staging','/sales/projectz/source','channels','',GETDATE());
insert into [dbo].[JOB_LIST] values(2,'countries_dim','/staging','/sales/projectz/source','countries','',GETDATE());
insert into [dbo].[JOB_LIST] values(3,'customers_dim','/staging','/sales/projectz/source','customers','',GETDATE());
insert into [dbo].[JOB_LIST] values(4,'product_dim','/staging','/sales/projectz/source','product','',GETDATE());
insert into [dbo].[JOB_LIST] values(5,'promotions_dim','/staging','/sales/projectz/source','promotions','',GETDATE());
insert into [dbo].[JOB_LIST] values(6,'tims_dim','/staging','/sales/projectz/source','times','',GETDATE());
insert into [dbo].[JOB_LIST] values(7,'sales_fact','/staging','/sales/projectz/source','sales','',GETDATE());
```

```
/*****************************************
** Creating ETL Load Status table in SQL DB...
*****************************************/
CREATE TABLE [dbo].[JOB_LIST](
[id] [varchar](100) NULL,
[jobname] [varchar](100) NULL,
[src_folderpath] [varchar](100) NULL,
[tgt_folderpath] [varchar](100) NULL,
[directory] [varchar](200) NULL,
[status] [varchar](100) NULL,
[processedon] [datetime] NULL
)
GO

/*********************************************************
**  INSERT SCRIPT FOR LIST OF JOBS
*********************************************************/

insert into [dbo].[JOB_LIST] values(1,'channels_dim','/staging','/sales/projectz/source','channels','',GETDATE());
insert into [dbo].[JOB_LIST] values(2,'countries_dim','/staging','/sales/projectz/source','countries','',GETDATE());
insert into [dbo].[JOB_LIST] values(3,'customers_dim','/staging','/sales/projectz/source','customers','',GETDATE());
insert into [dbo].[JOB_LIST] values(4,'product_dim','/staging','/sales/projectz/source','product','',GETDATE());
insert into [dbo].[JOB_LIST] values(5,'promotions_dim','/staging','/sales/projectz/source','promotions','',GETDATE());
insert into [dbo].[JOB_LIST] values(6,'tims_dim','/staging','/sales/projectz/source','times','',GETDATE());
insert into [dbo].[JOB_LIST] values(7,'sales_fact','/staging','/sales/projectz/source','sales','',GETDATE());
```

```
/*****************************************
** Creating ETL Load Status table in SQL DB...
*****************************************/
CREATE TABLE [dbo].[ETL_LOAD_STATUS](
[DataFactoryName] [varchar](100) NULL,
[PipelineName] [varchar](100) NULL,
[RunId] [varchar](100) NULL,
[activityname] [varchar](200) NULL,
[readrows] [varchar](100) NULL,
[writerows] [varchar](100) NULL,
[activitystatus] [varchar](200) NULL,
[ErrorMessage] [varchar](1000) NULL,
[starttime] [datetime] NULL,
[endtime] [datetime] NULL,
[Createdon] [datetime] NULL
)
GO
```

```
/*****************************************************************
** Creating PROCEDURE FOR ETL Load Status updating table in SQL DB...
*****************************************************************/
CREATE PROCEDURE [dbo].[PROC_ETL_LOAD_STATUS] (@DataFactoryName varchar(100), @PipelineName
varchar(100), @runid varchar(100),@activityname varchar(100),@readrows varchar(100),@writerows
varchar(100),
@activitystatus varchar(100),@starttime datetime,@endtime datetime,@ErrorMessage
varchar(1000),@Createdon datetime)
AS BEGIN
INSERT INTO [dbo].[ETL_LOAD_STATUS]
(
[DataFactoryName],[PipelineName],[RunId],[activityname],[readrows],[writerows],[activitystatus],
[ErrorMessage],[starttime],[endtime],[Createdon]
)VALUES
(@DataFactoryName, @PipelineName, @runid, @activityname, @readrows, @writerows,
@activitystatus,@ErrorMessage,
@starttime,@endtime,@Createdon
)
END
GO
```

Create key vault

Validation passed

Basics | Access policy | Networking | Tags | Review + create

Review + create

Basics

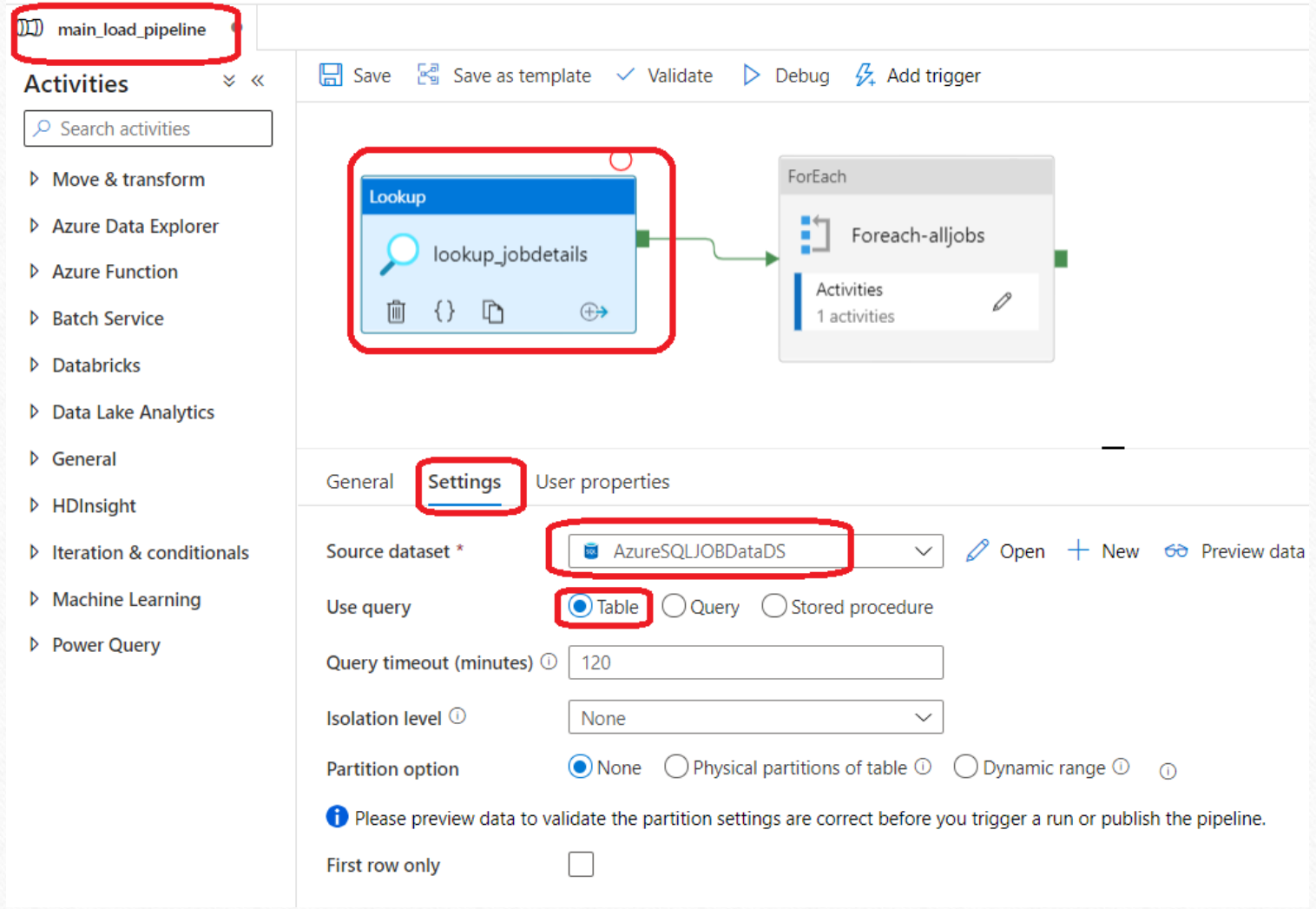
Subscription	Free Trial
Resource group	dev_rg
Key vault name	pyspark-dev-kv
Region	East US
Pricing tier	Standard
Soft-delete	Enabled
Purge protection during retention period	Disabled
Days to retain deleted vaults	90 days

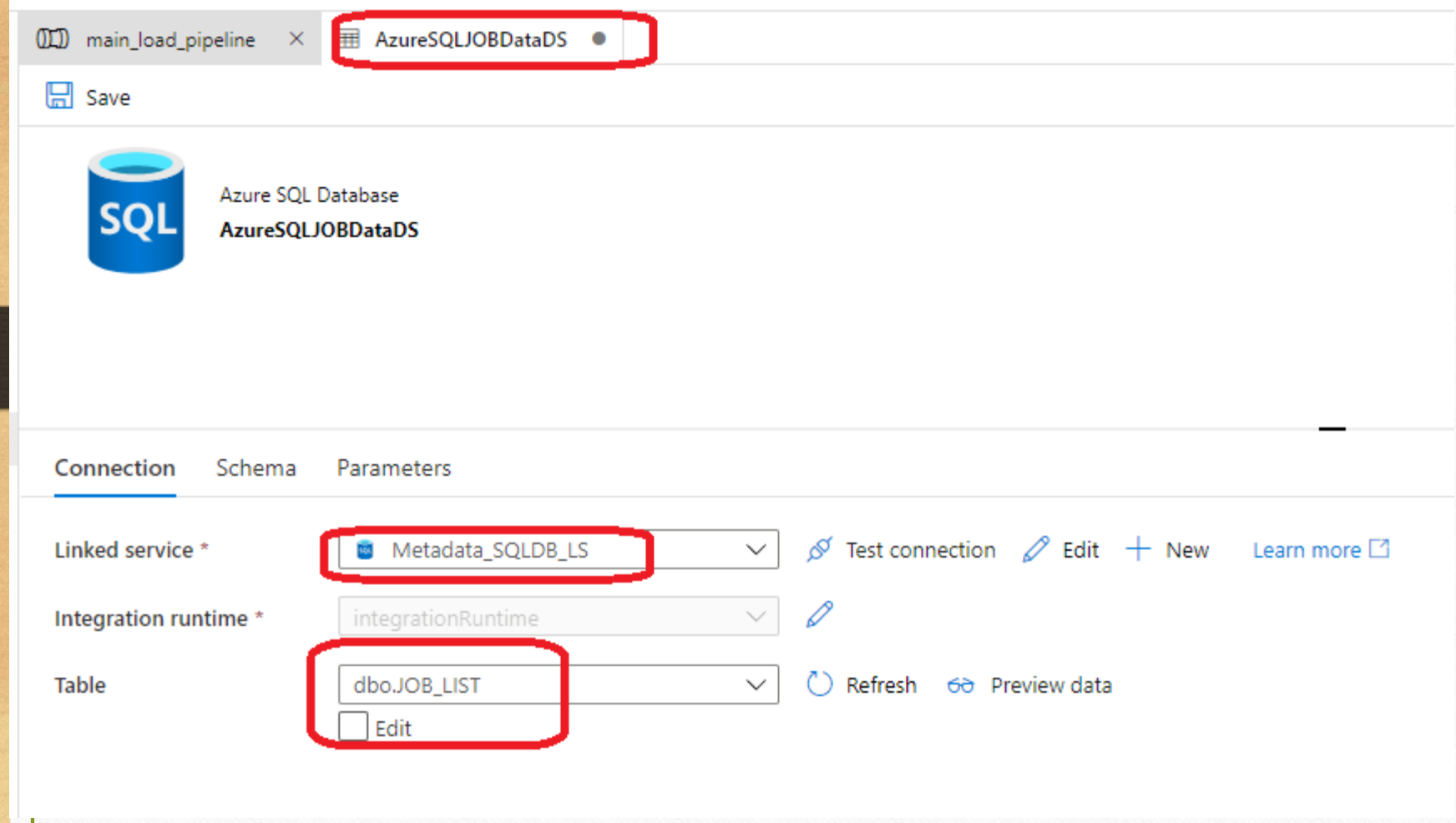
Access policy

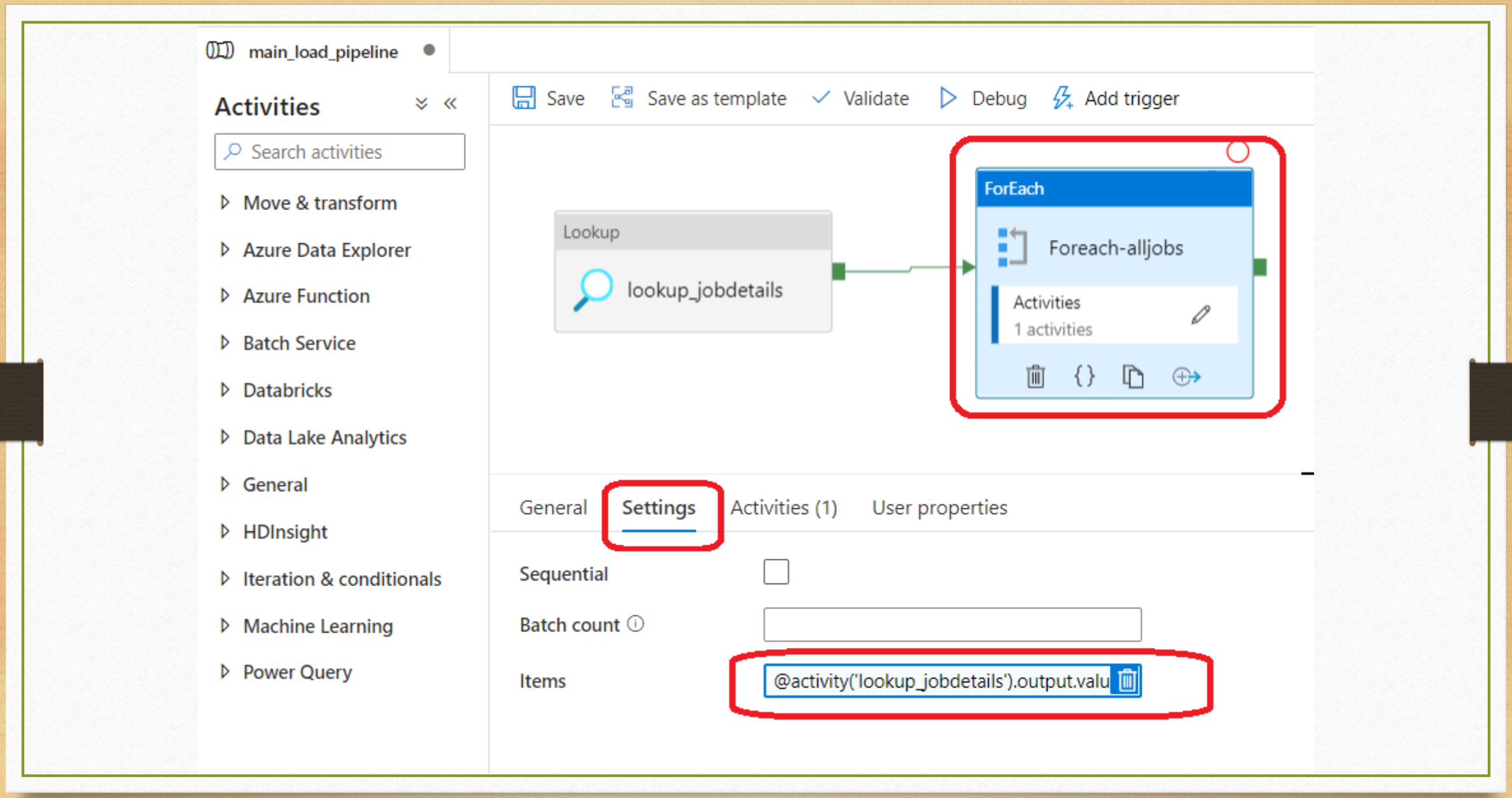
Azure Virtual Machines for deployment	Enabled
Azure Resource Manager for template deployment	Enabled
Azure Disk Encryption for volume encryption	Enabled
Permission model	Vault access policy
Access policies	1

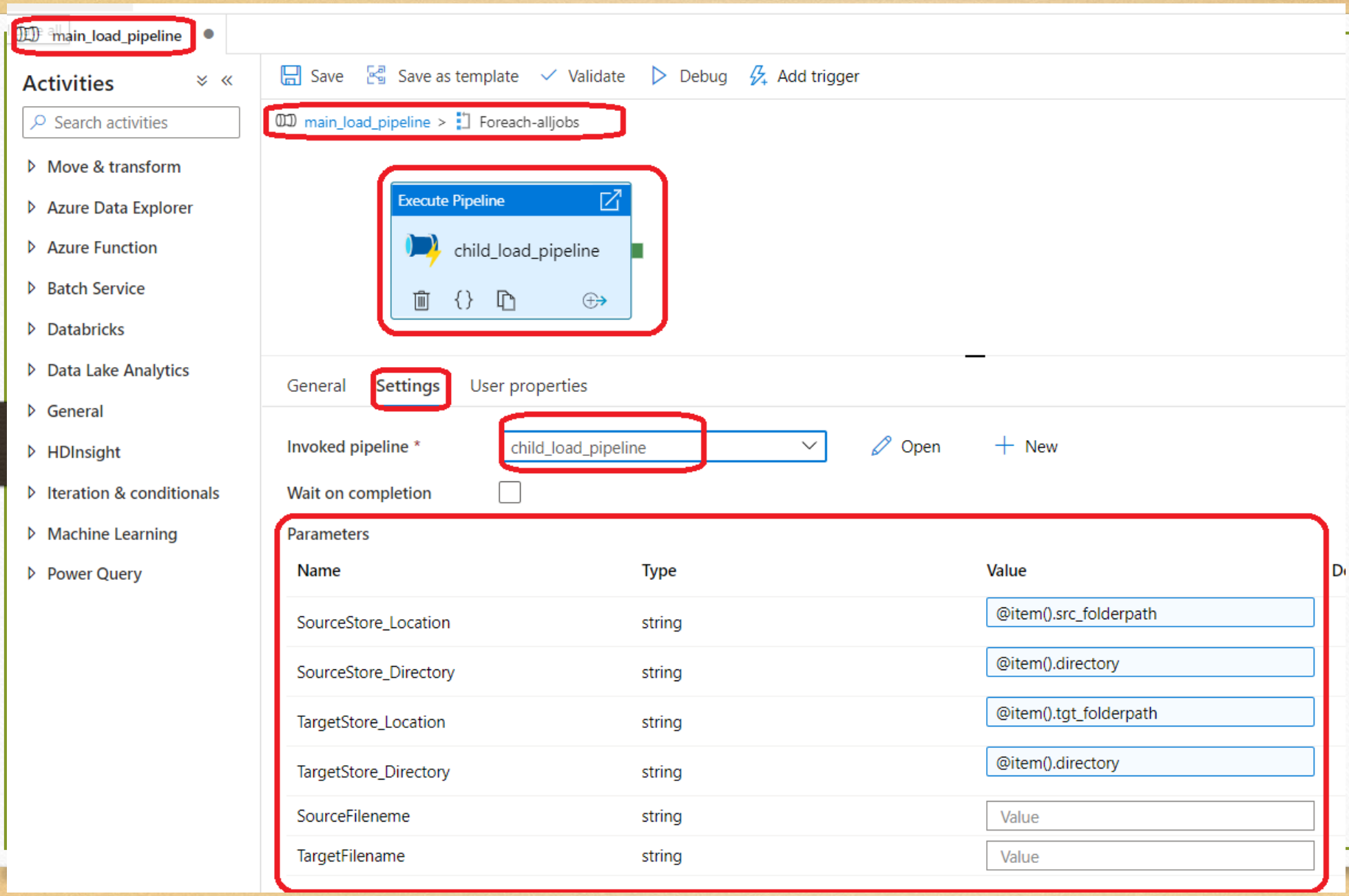
Networking

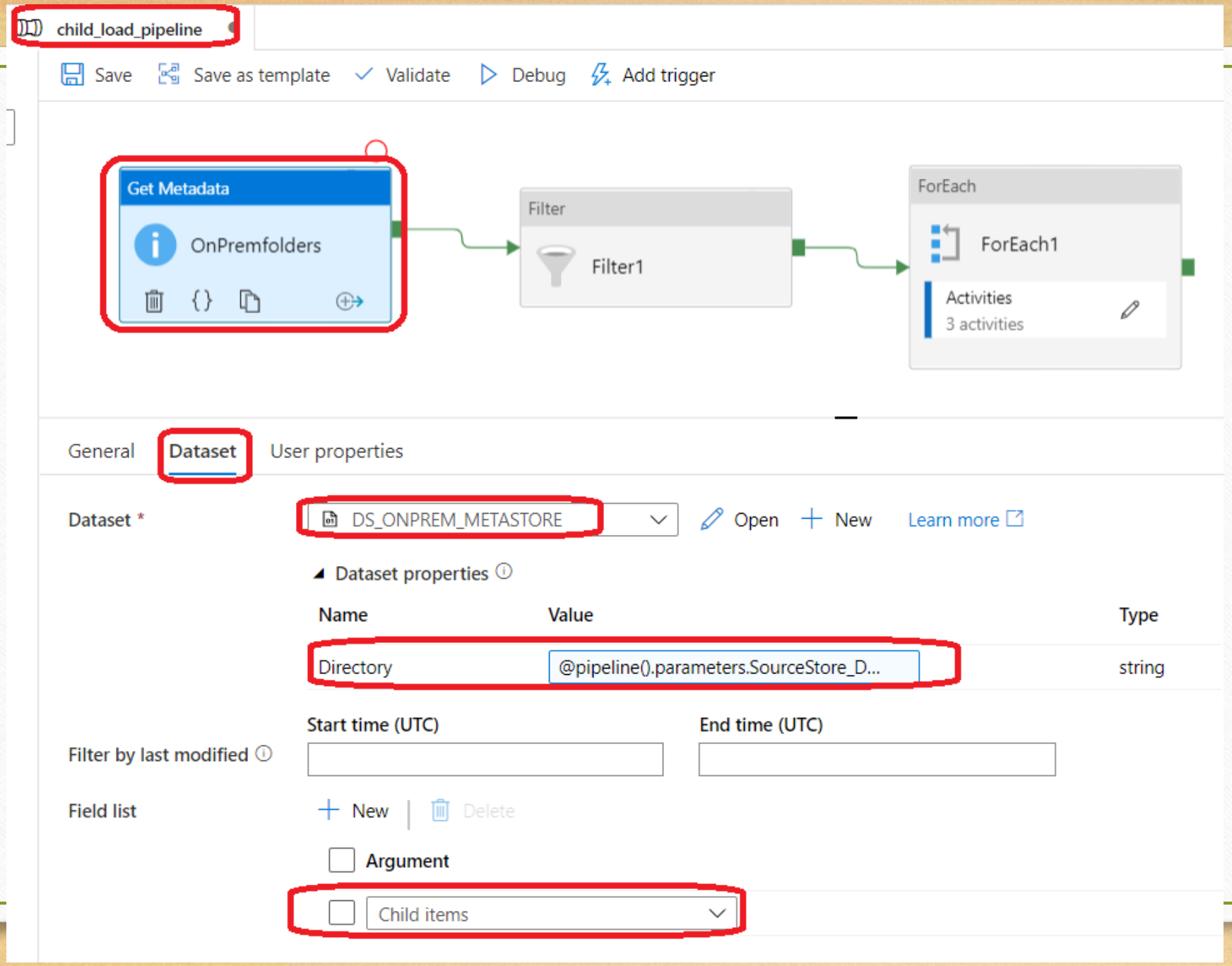
Connectivity method	Public endpoint (all networks)

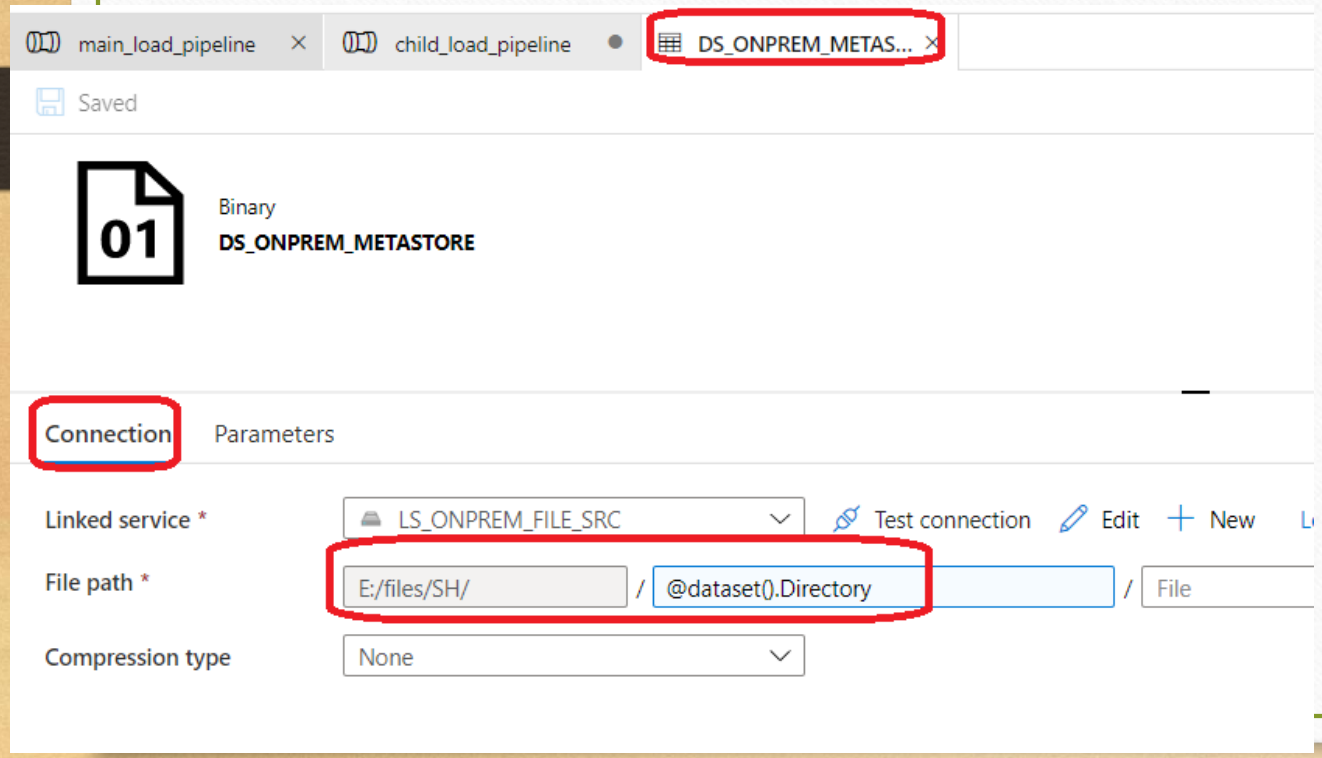

main_load_pipeline
Activities
Search activities
Move & transform
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Save
Save as template
Validate
Debug
Add trigger
Lookup
lookup_jobdetails
ForEach
Foreach-alljobs
Activities
1 activities
General
Settings
User properties
Source dataset *
AzureSQLJOBDataDS
Open
New
Preview data
Use query
Table
Query
Stored procedure
Query timeout (minutes)
120
Isolation level
None
Partition option
None
Physical partitions of table
Dynamic range
Please preview data to validate the partition settings are correct before you trigger a run or publish the pipeline.
First row only

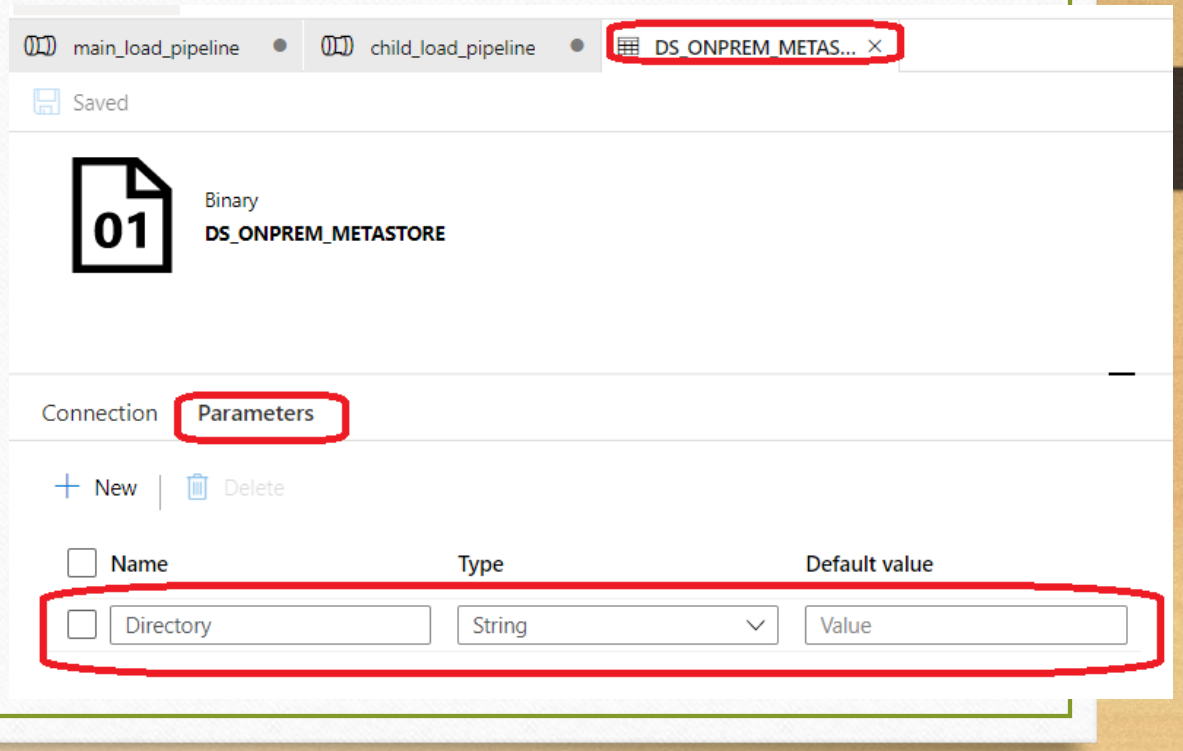

main_load_pipeline
AzureSQLJOBDataDS
Save
Azure SQL Database
AzureSQLJOBDataDS
Connection
Schema
Parameters
Linked service *
Metadata_SQLDB_LS
Test connection
Edit
New
Learn more
Integration runtime *
integrationRuntime
Table
dbo.JOB_LIST
Refresh
Preview data
Edit

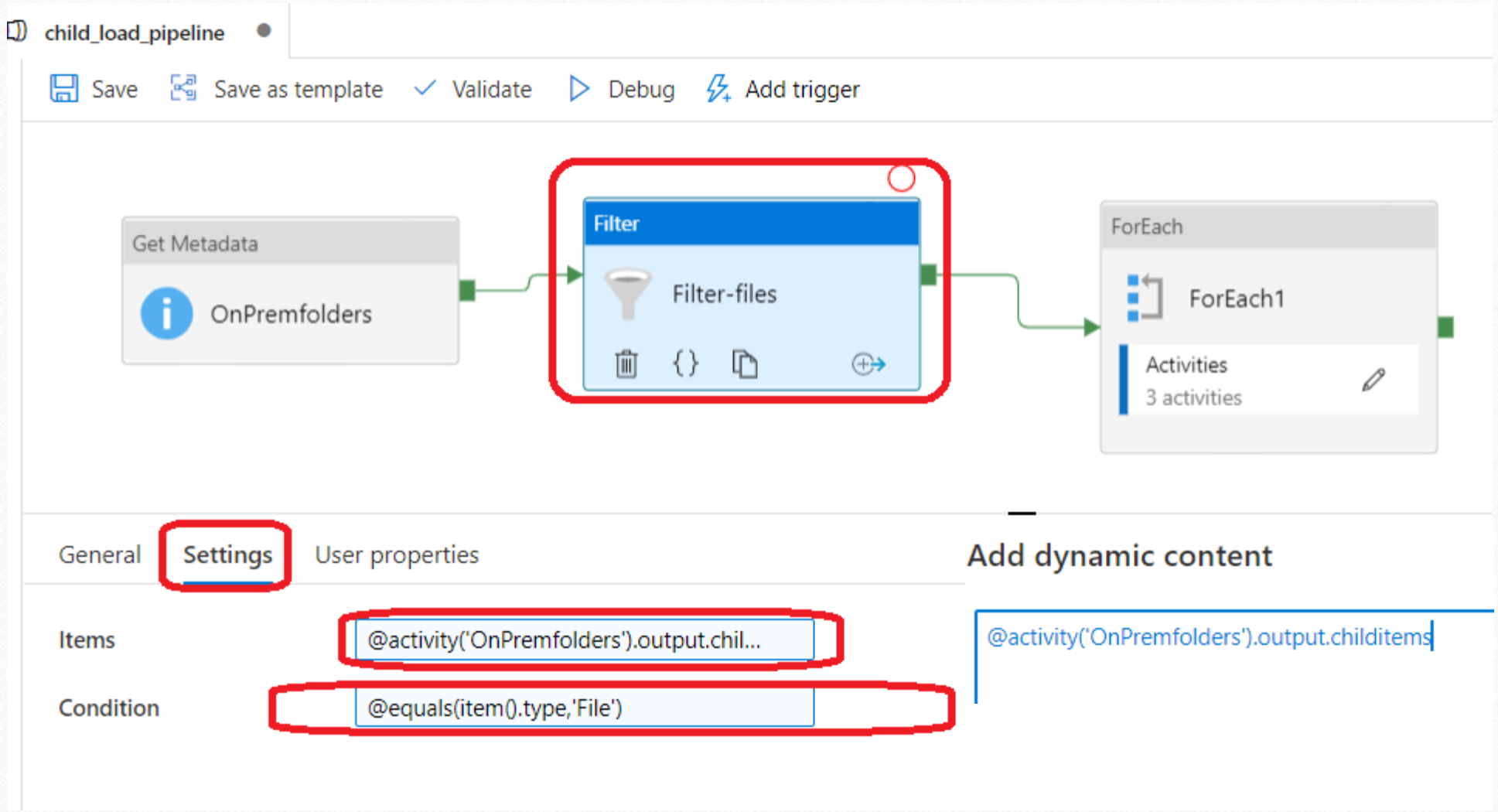

main_load_pipeline
Activities
Search activities
Move & transform
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Save
Save as template
Validate
Debug
Add trigger
Lookup
lookup_jobdetails
ForEach
Foreach-alljobs
Activities
1 activities
General
Settings
Activities (1)
User properties
Sequential
Batch count
Items
@activity('lookup_jobdetails').output.valu

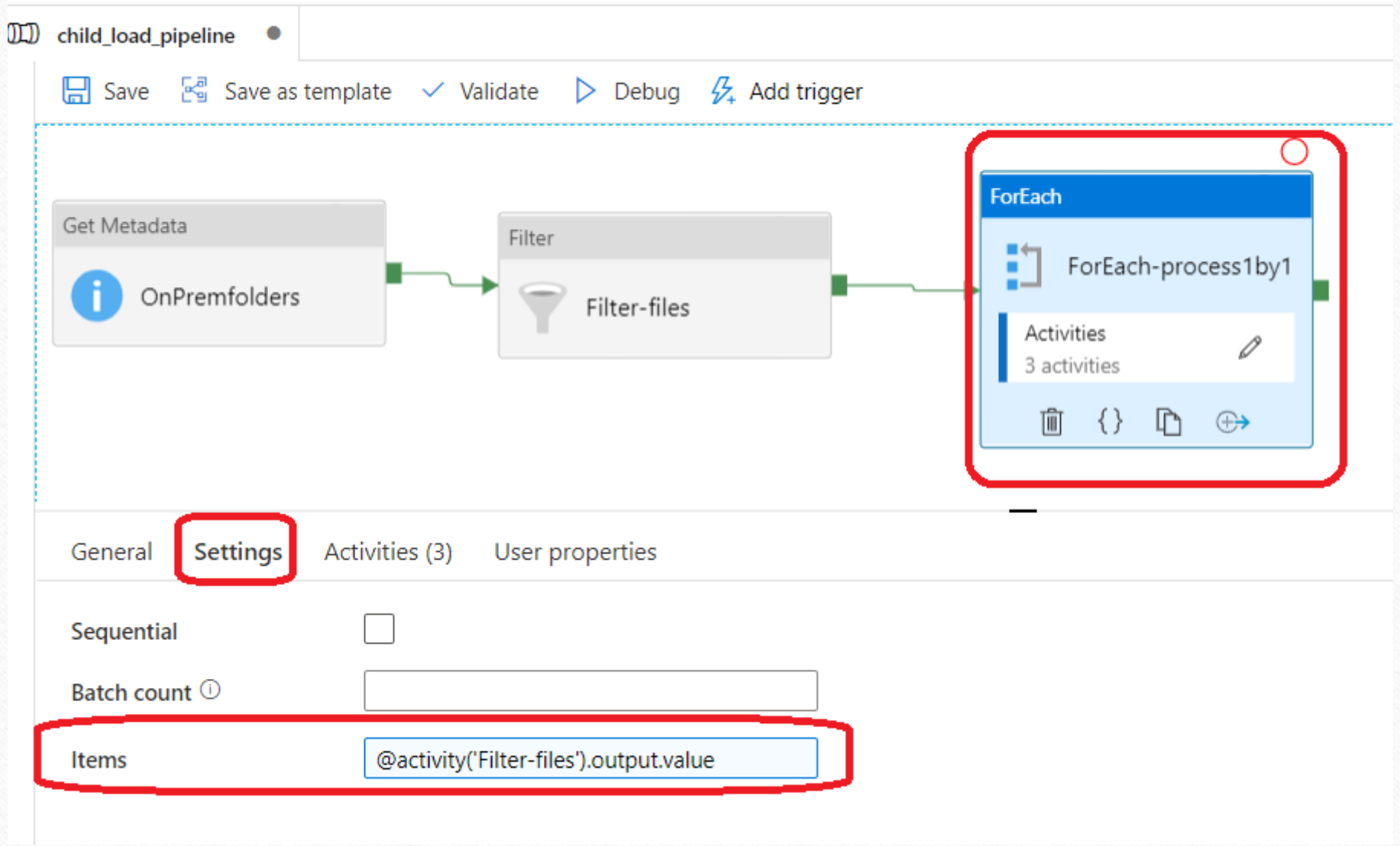

main_load_pipeline
Activities
Search activities
Move & transform
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Save
Save as template
Validate
Debug
Add trigger
main_load_pipeline > Foreach-alljobs
Execute Pipeline
child_load_pipeline
General
Settings
User properties
Invoked pipeline *
child_load_pipeline
Open
New
Wait on completion
Parameters
Name
Type
Value
SourceStore_Location
string
@item().src_folderpath
SourceStore_Directory
string
@item().directory
TargetStore_Location
string
@item().tgt_folderpath
TargetStore_Directory
string
@item().directory
SourceFileneme
string
Value
TargetFilename
string
Value

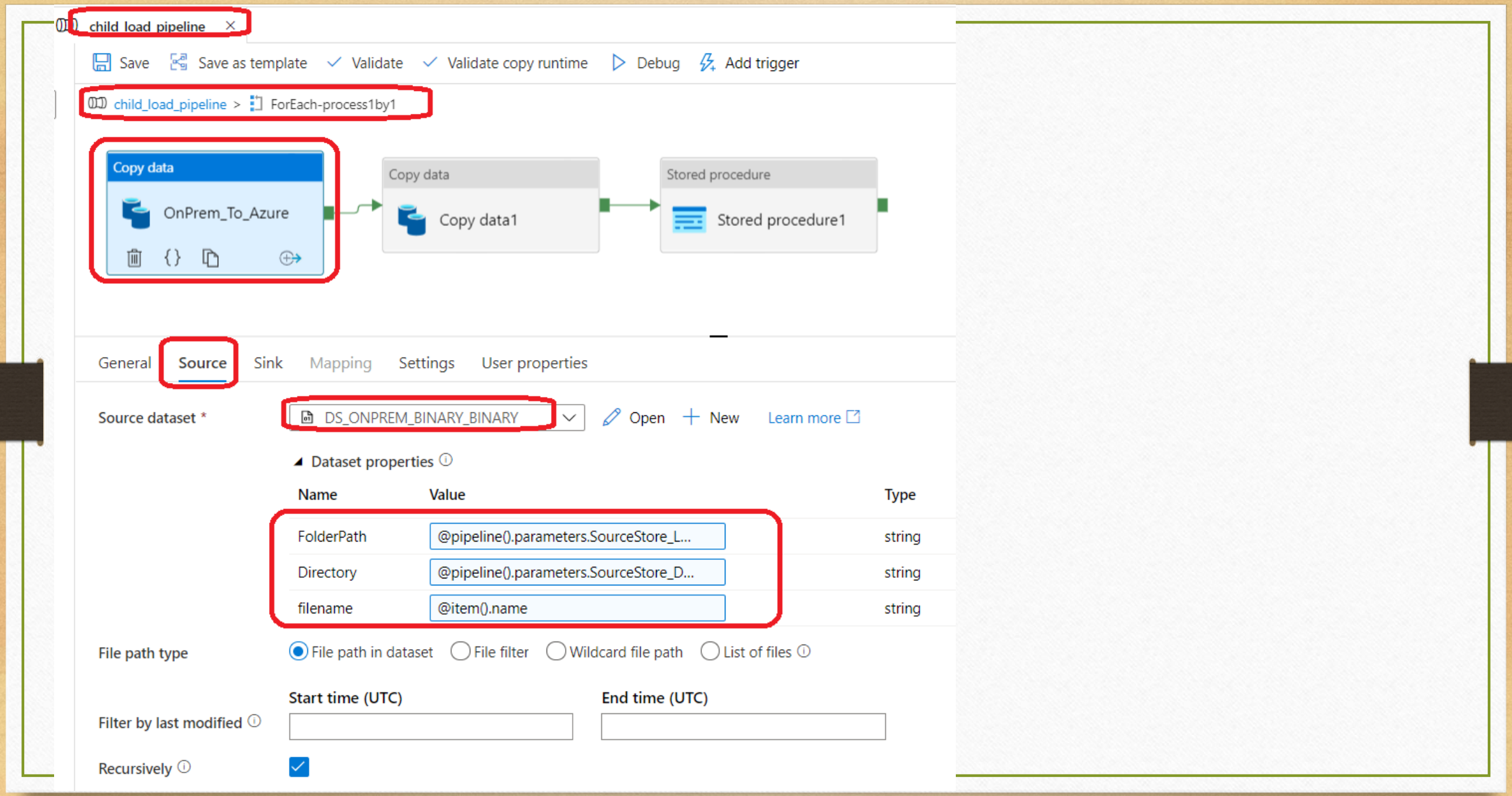

child_load_pipeline
Save
Save as template
Validate
Debug
Add trigger
Get Metadata
OnPremfolders
Filter
Filter1
ForEach
ForEach1
Activities
3 activities
General
Dataset
User properties
Dataset *
DS_ONPREM_METASTORE
Open
New
Learn more
Dataset properties
Name
Value
Type
Directory
@pipeline().parameters.SourceStore_D...
string
Start time (UTC)
End time (UTC)
Filter by last modified
Field list
New
Delete
Argument
Child items

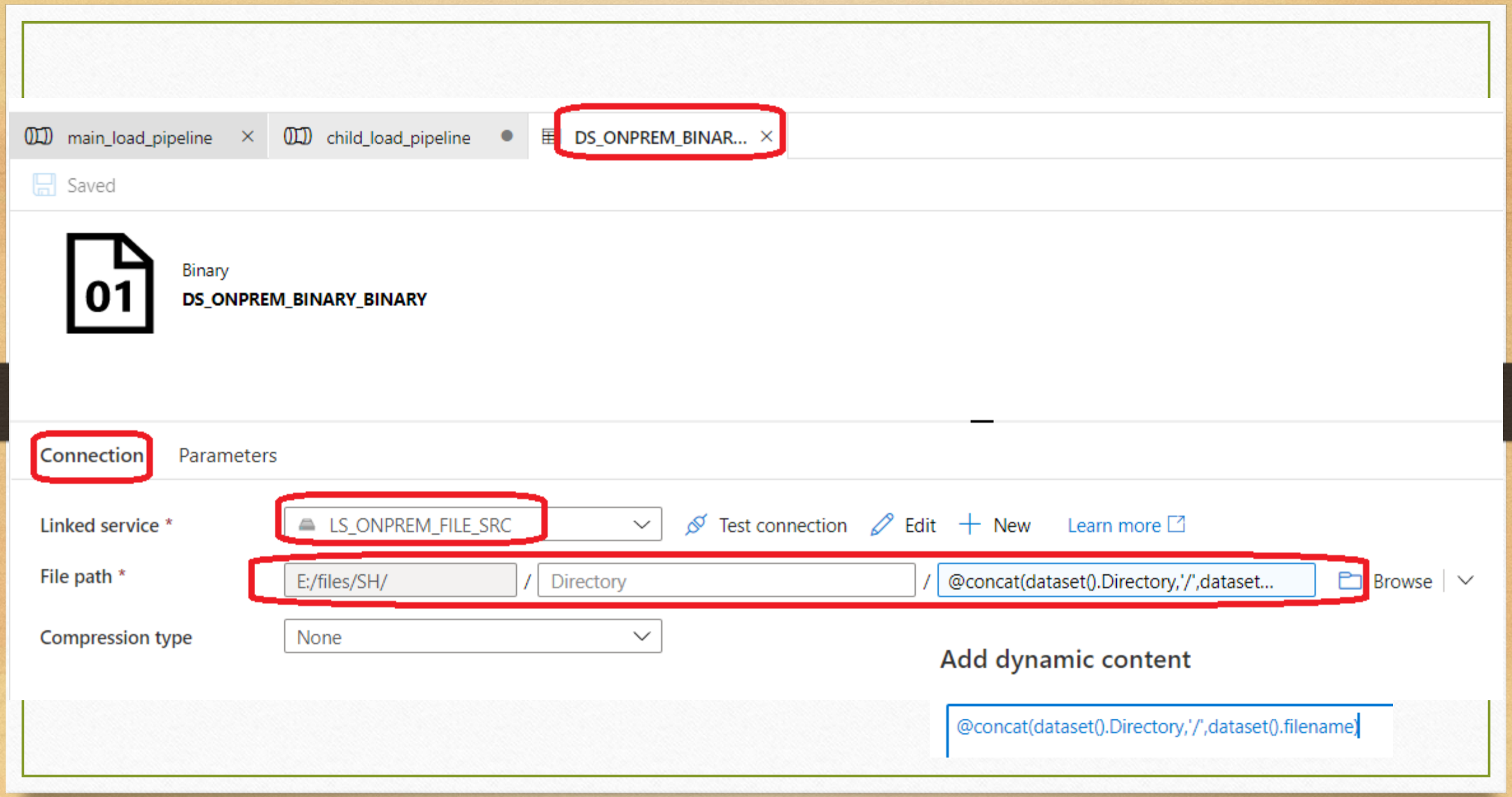

main_load_pipeline
child_load_pipeline
DS_ONPREM_METAS...
Saved
Binary
DS_ONPREM_METASTORE
Connection
Parameters
Linked service *
LS_ONPREM_FILE_SRC
Test connection
Edit
New
File path *
E:/files/SH/
@dataset().Directory
File
Compression type
None

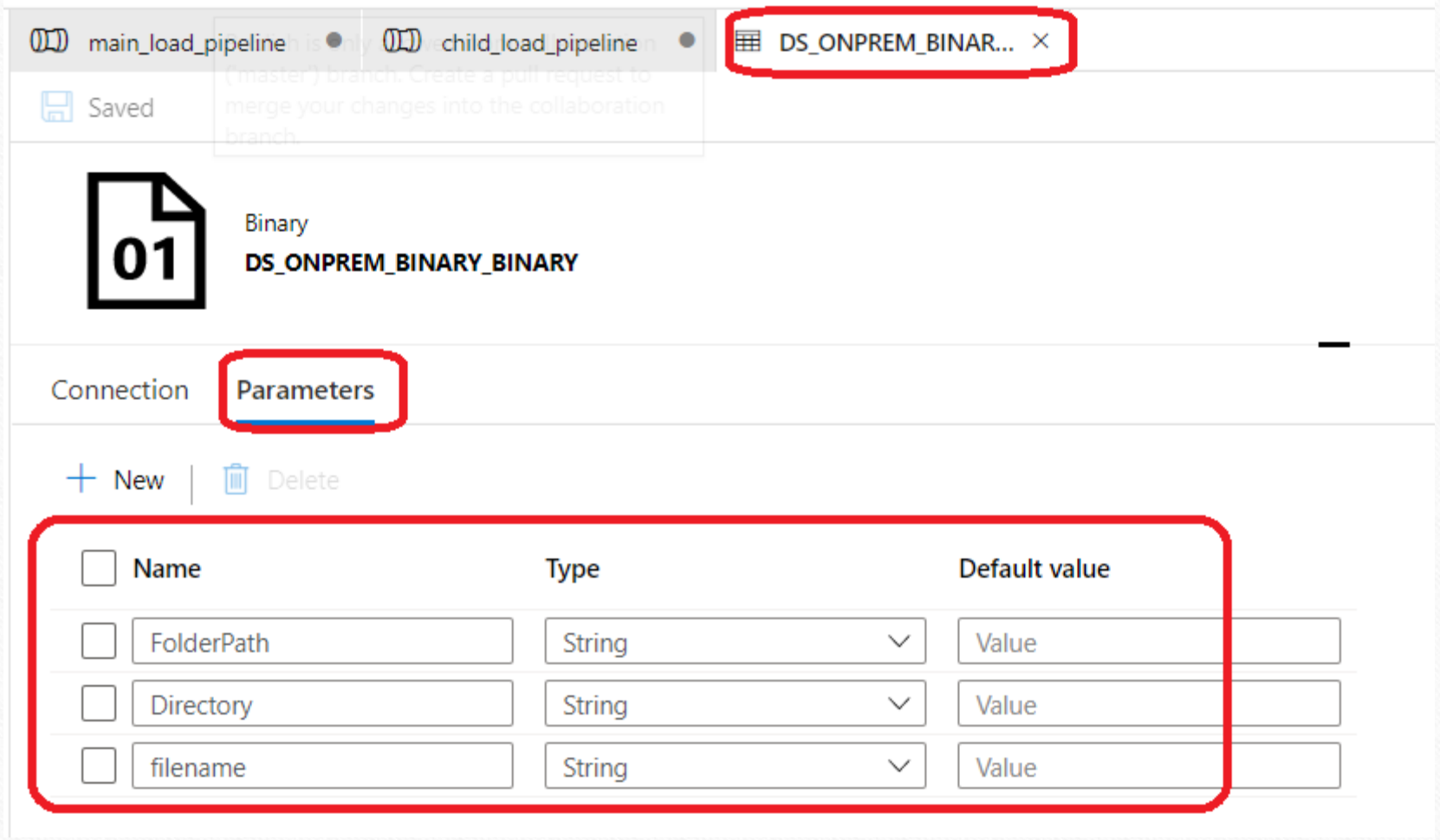

main_load_pipeline
child_load_pipeline
DS_ONPREM_METAS...
Saved
Binary
DS_ONPREM_METASTORE
Connection
Parameters
New
Delete
Name
Type
Default value
Directory
String
Value

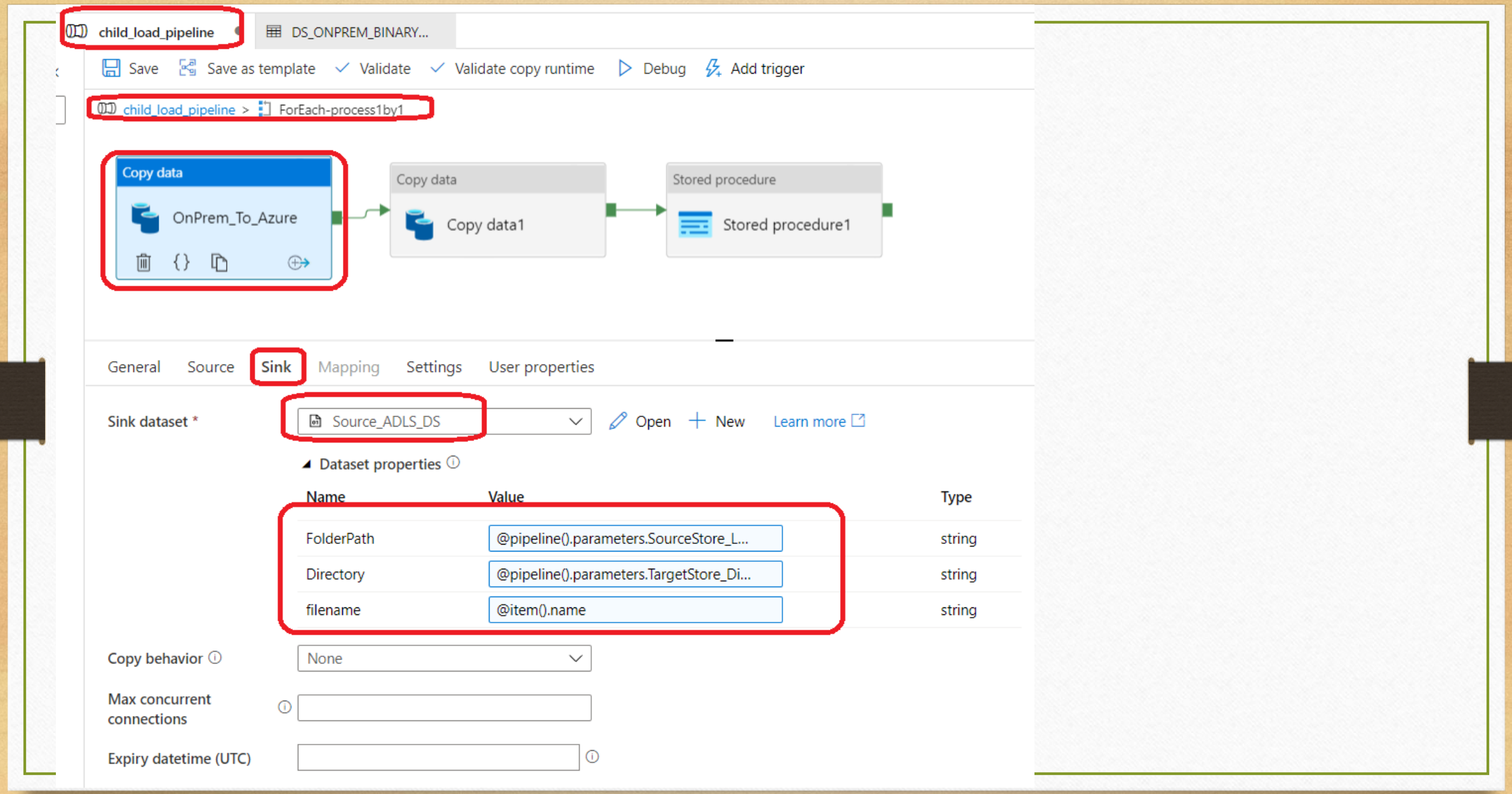

child_load_pipeline
Save
Save as template
Validate
Debug
Add trigger
Get Metadata
OnPremfolders
Filter
Filter-files
ForEach
ForEach1
Activities
3 activities
General
Settings
User properties
Items
@activity('OnPremfolders').output.chil...
Condition
@equals(item().type,'File')
Add dynamic content
@activity('OnPremfolders').output.childitems

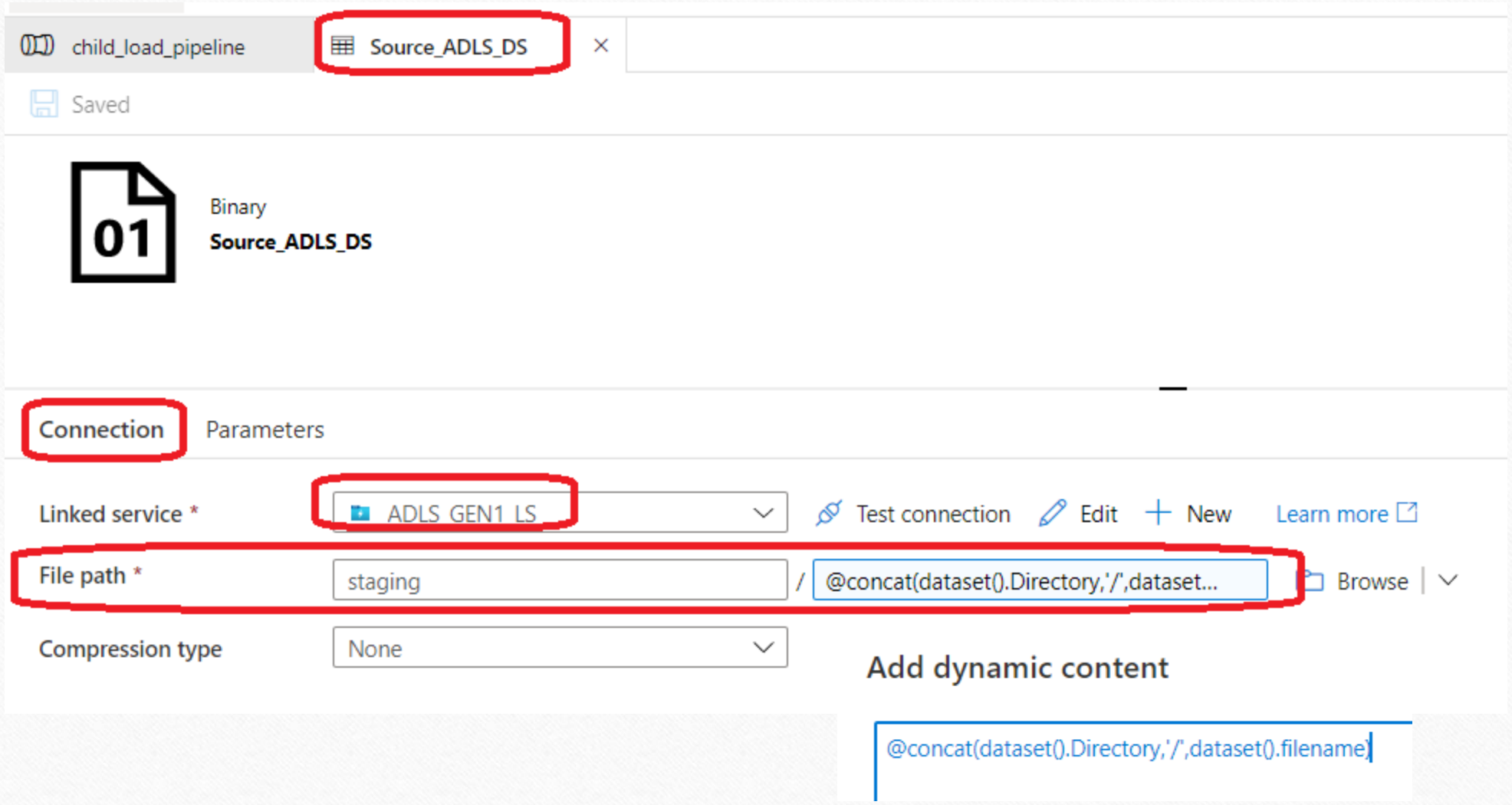

child_load_pipeline
Save
Save as template
Validate
Debug
Add trigger
Get Metadata
OnPremfolders
Filter
Filter-files
ForEach
ForEach-process1by1
Activities
3 activities
General
Settings
Activities (3)
User properties
Sequential
Batch count
Items
@activity('Filter-files').output.value

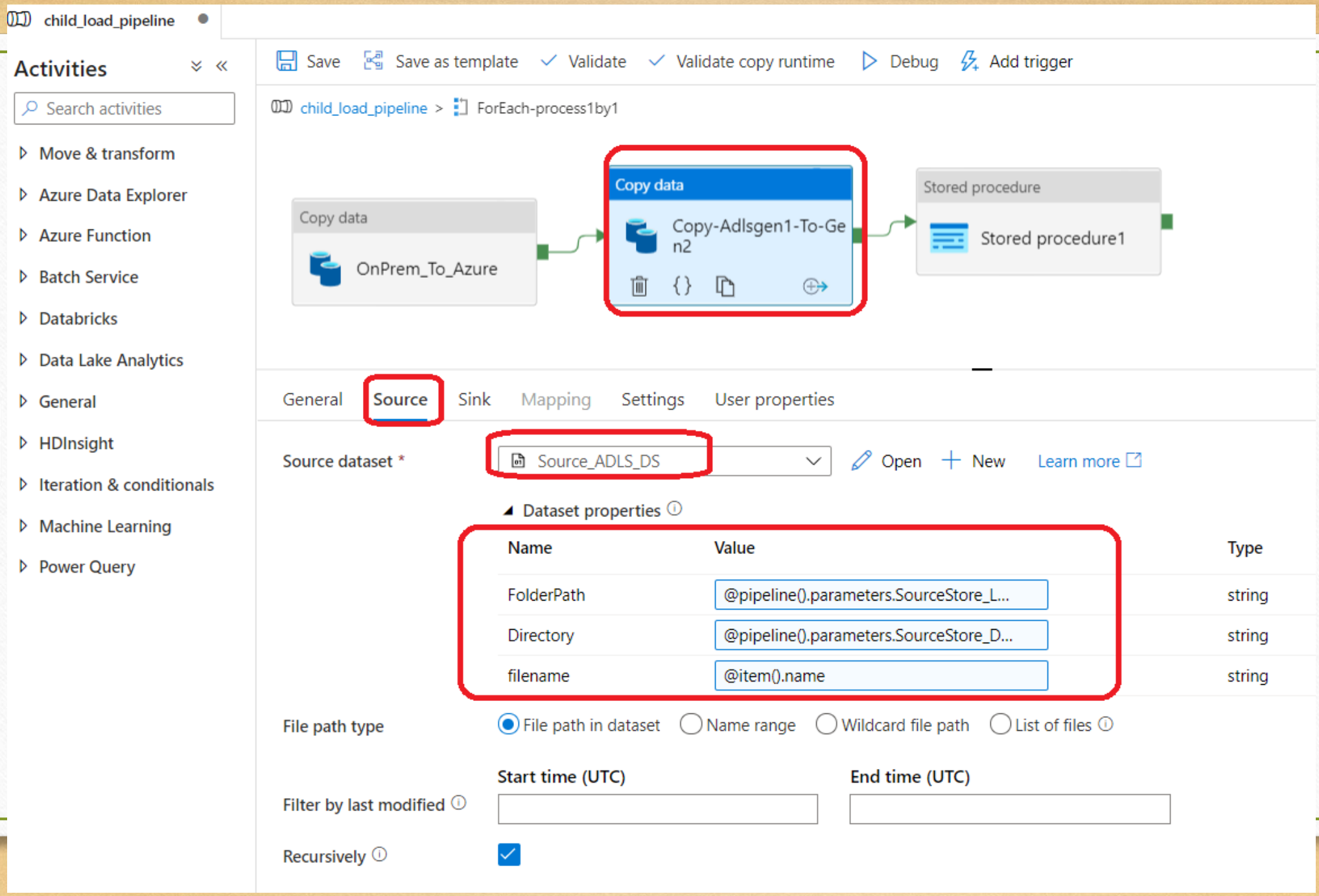

child_load_pipeline
Save
Save as template
Validate
Validate copy runtime
Debug
Add trigger
child_load_pipeline > ForEach-process1by1
Copy data
OnPrem_To_Azure
Copy data
Copy data1
Stored procedure
Stored procedure1
General
Source
Sink
Mapping
Settings
User properties
Source dataset *
DS_ONPREM_BINARY_BINARY
Open
New
Learn more
Dataset properties
Name
Value
Type
FolderPath
@pipeline().parameters.SourceStore_L...
string
Directory
@pipeline().parameters.SourceStore_D...
string
filename
@item().name
string
File path type
File path in dataset
File filter
Wildcard file path
List of files
Filter by last modified
Start time (UTC)
End time (UTC)
Recursively

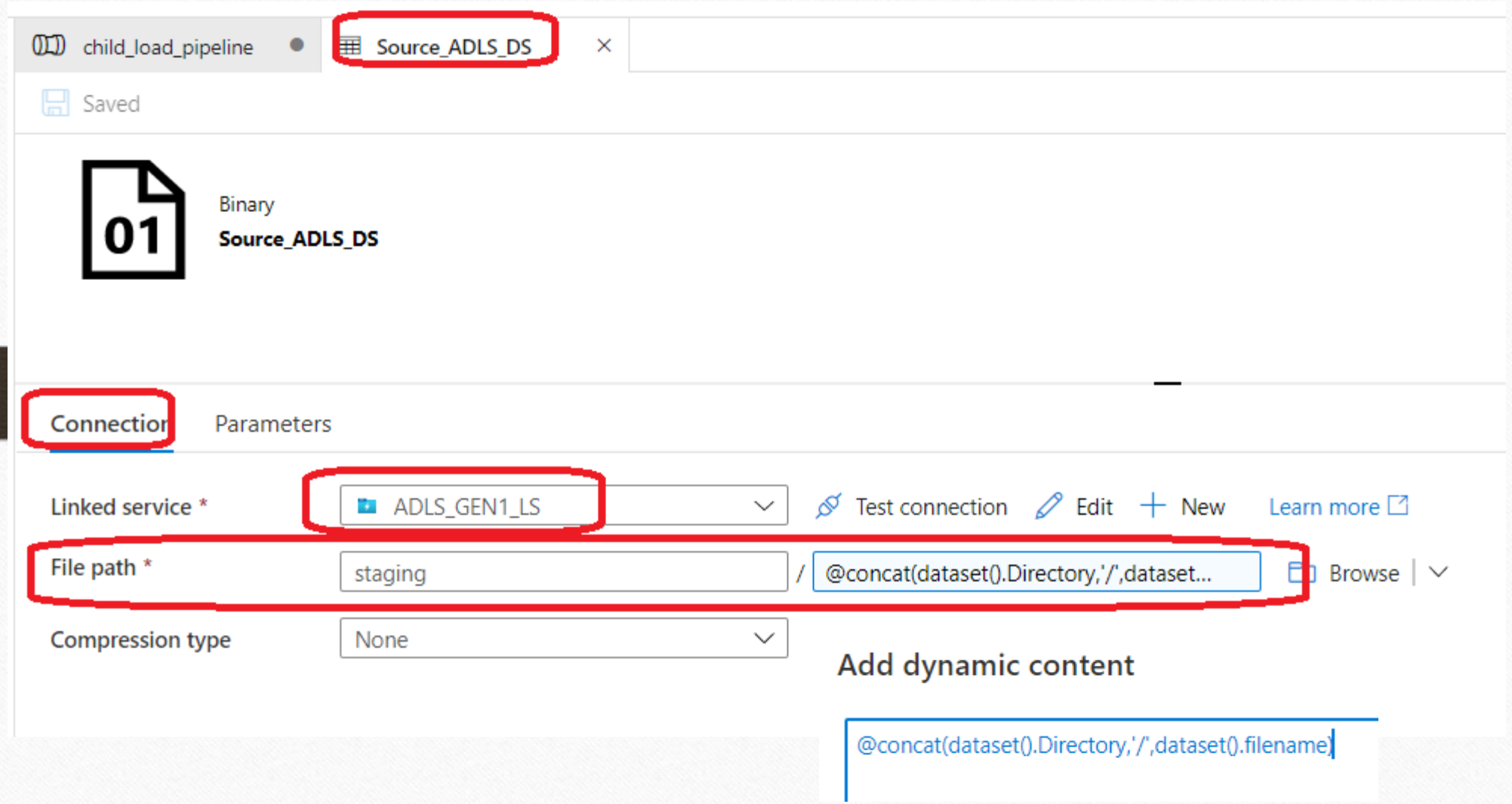

main_load_pipeline
child_load_pipeline
DS_ONPREM_BINAR...
Saved
01
Binary
DS_ONPREM_BINARY_BINARY
Connection
Parameters
Linked service *
LS_ONPREM_FILE_SRC
Test connection
Edit
New
Learn more
File path *
E:/files/SH/
Directory
@concat(dataset().Directory,'/',dataset...
Browse
Compression type
None
Add dynamic content
@concat(dataset().Directory,'/',dataset().filename)

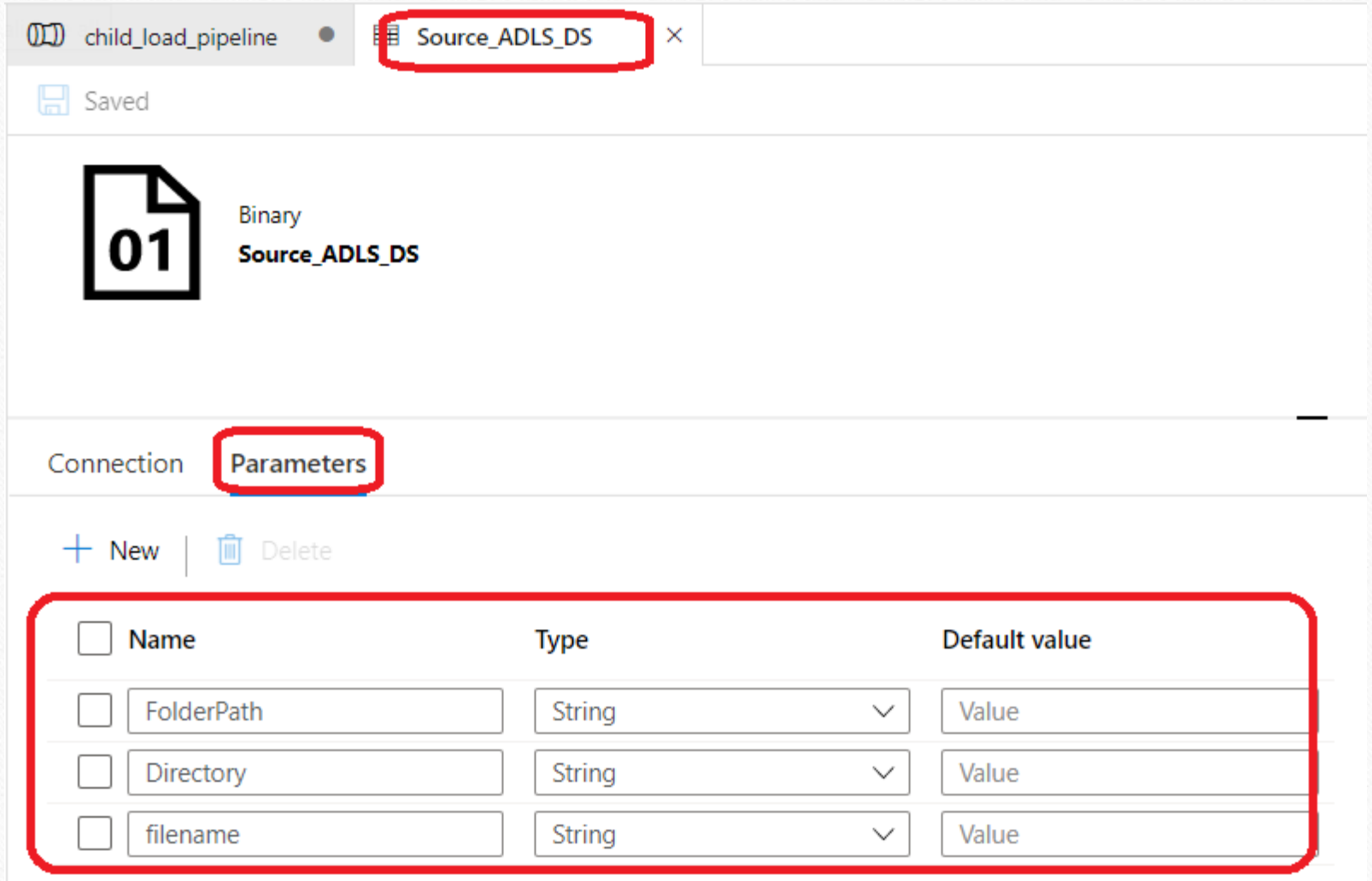

main_load_pipeline
child_load_pipeline
DS_ONPREM_BINAR...
Saved
Binary
DS_ONPREM_BINARY_BINARY
Connection
Parameters
New
Delete
Name
Type
Default value
FolderPath
String
Value
Directory
String
Value
filename
String
Value

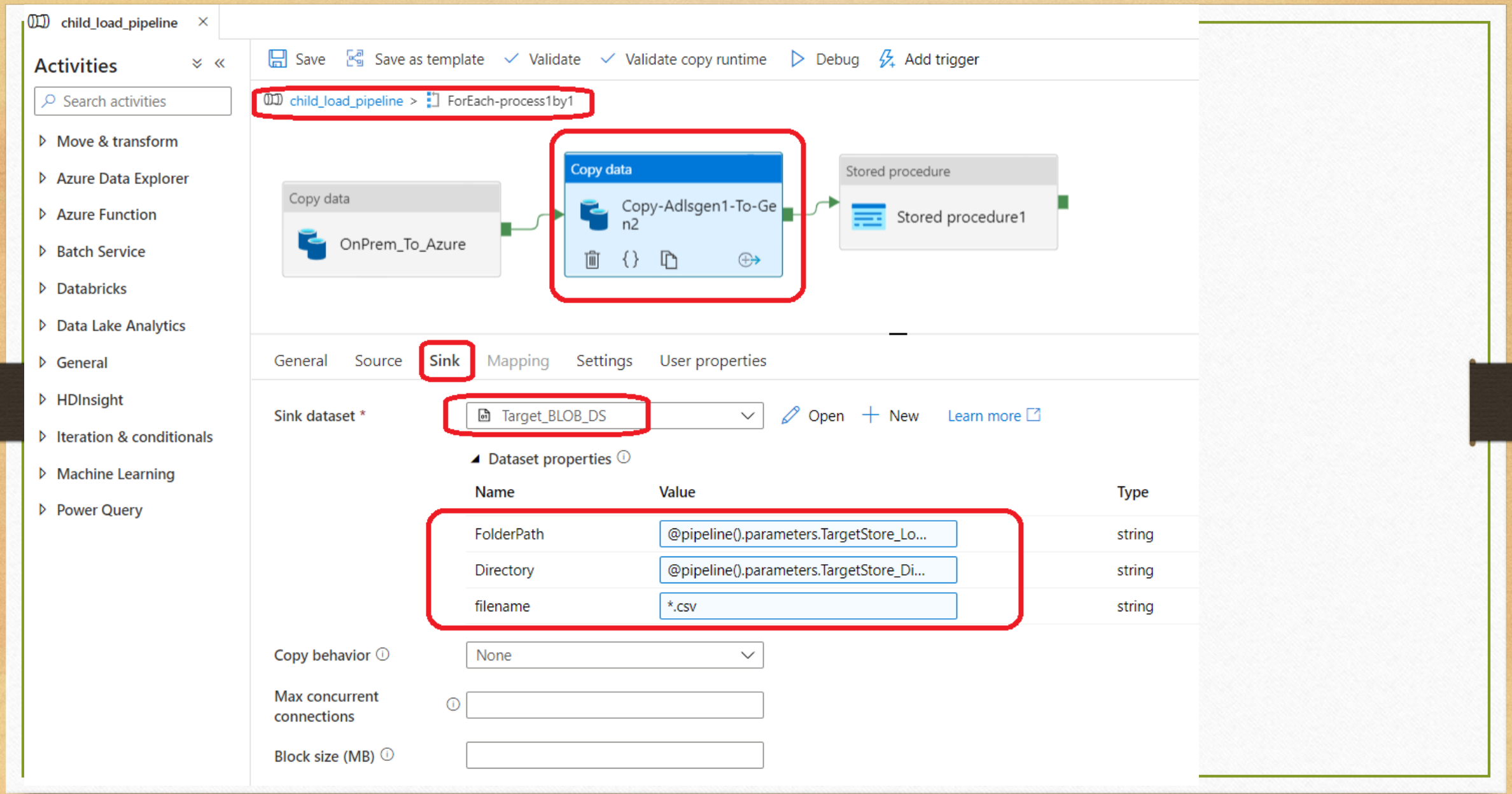

child_load_pipeline
DS_ONPREM_BINARY...
Save
Save as template
Validate
Validate copy runtime
Debug
Add trigger
child_load_pipeline > ForEach-process1by1
Copy data
OnPrem_To_Azure
Copy data
Copy data1
Stored procedure
Stored procedure1
General
Source
Sink
Mapping
Settings
User properties
Sink dataset *
Source_ADLS_DS
Open
New
Learn more
Dataset properties
Name
Value
Type
FolderPath
@pipeline().parameters.SourceStore_L...
string
Directory
@pipeline().parameters.TargetStore_Di...
string
filename
@item().name
string
Copy behavior
None
Max concurrent connections
Expiry datetime (UTC)

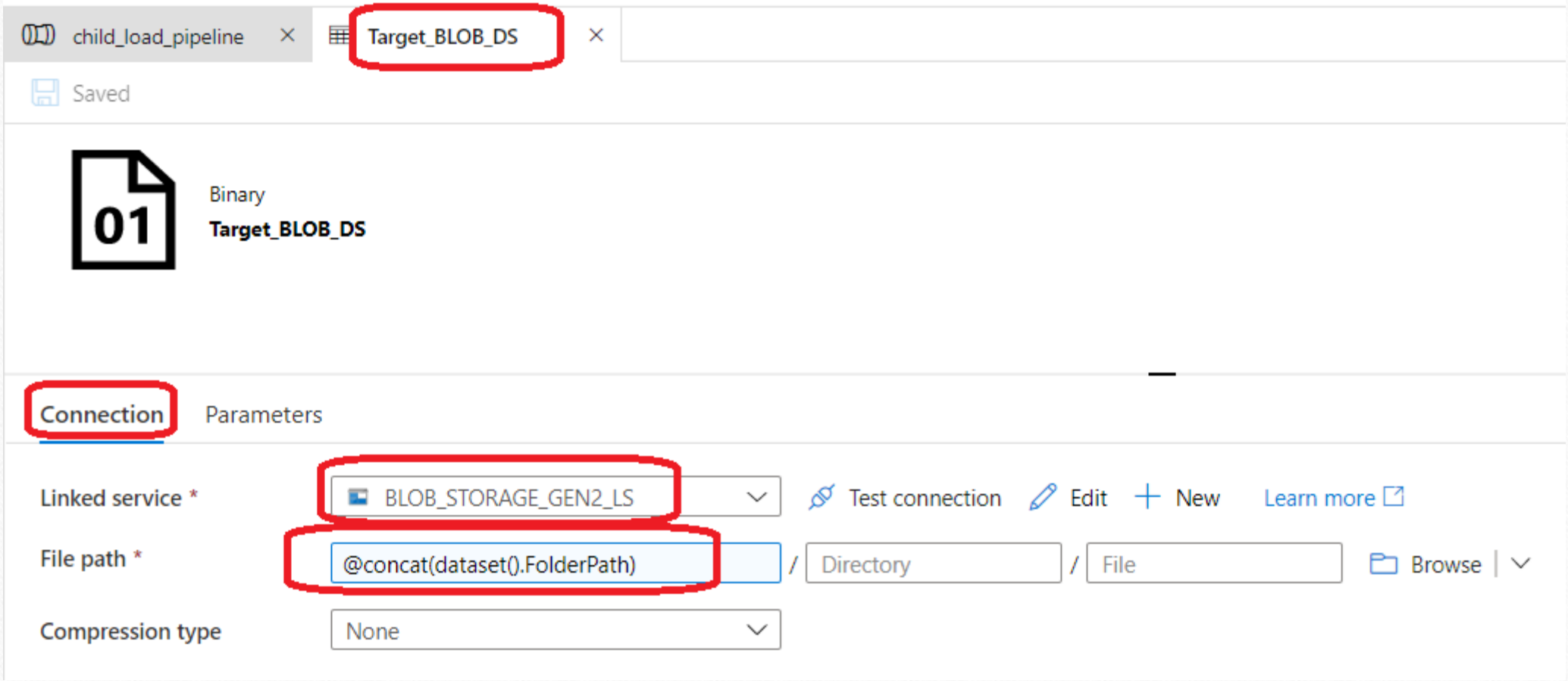

child_load_pipeline
Source_ADLS_DS
Saved
Binary
Source_ADLS_DS
Connection
Parameters
Linked service *
ADLS_GEN1_LS
Test connection
Edit
New
Learn more
File path *
staging
@concat(dataset().Directory,'/',dataset...
Browse
Compression type
None
Add dynamic content
@concat(dataset().Directory,'/',dataset().filename)

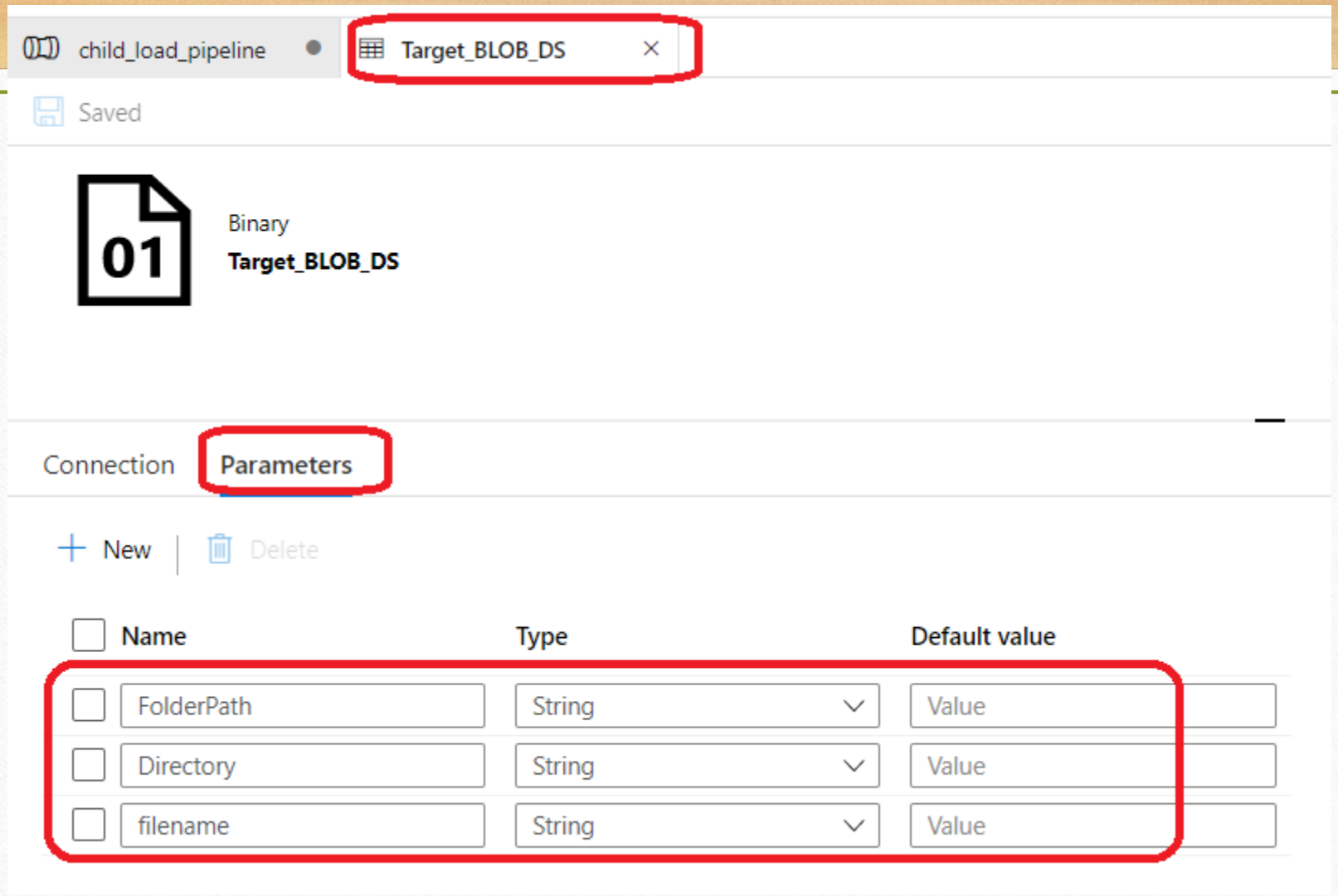

child_load_pipeline
Activities
Search activities
Move & transform
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Save
Save as template
Validate
Validate copy runtime
Debug
Add trigger
child_load_pipeline > ForEach-process1by1
Copy data
OnPrem_To_Azure
Copy data
Copy-Adlsgen1-To-Gen2
Stored procedure
Stored procedure1
General
Source
Sink
Mapping
Settings
User properties
Source dataset *
Source_ADLS_DS
Open
New
Learn more
Dataset properties
Name
Value
Type
FolderPath
@pipeline().parameters.SourceStore_L...
string
Directory
@pipeline().parameters.SourceStore_D...
string
filename
@item().name
string
File path type
File path in dataset
Name range
Wildcard file path
List of files
Filter by last modified
Start time (UTC)
End time (UTC)
Recursively

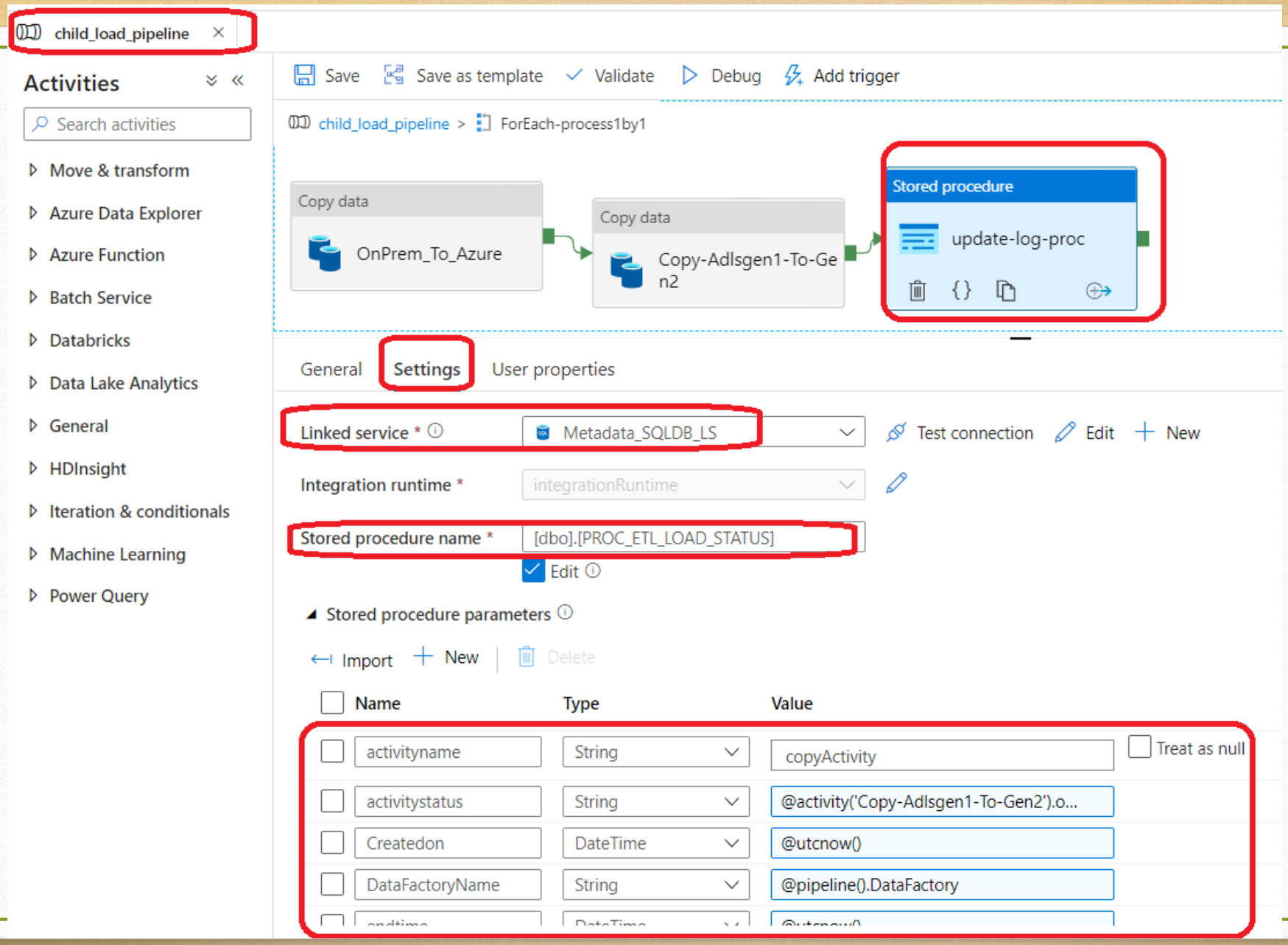

child_load_pipeline
Source_ADLS_DS
Saved
Binary
Source_ADLS_DS
Connection
Parameters
Linked service *
ADLS_GEN1_LS
Test connection
Edit
New
Learn more
File path *
staging
@concat(dataset().Directory,'/',dataset...
Browse
Compression type
None
Add dynamic content
@concat(dataset().Directory,'/',dataset().filename)

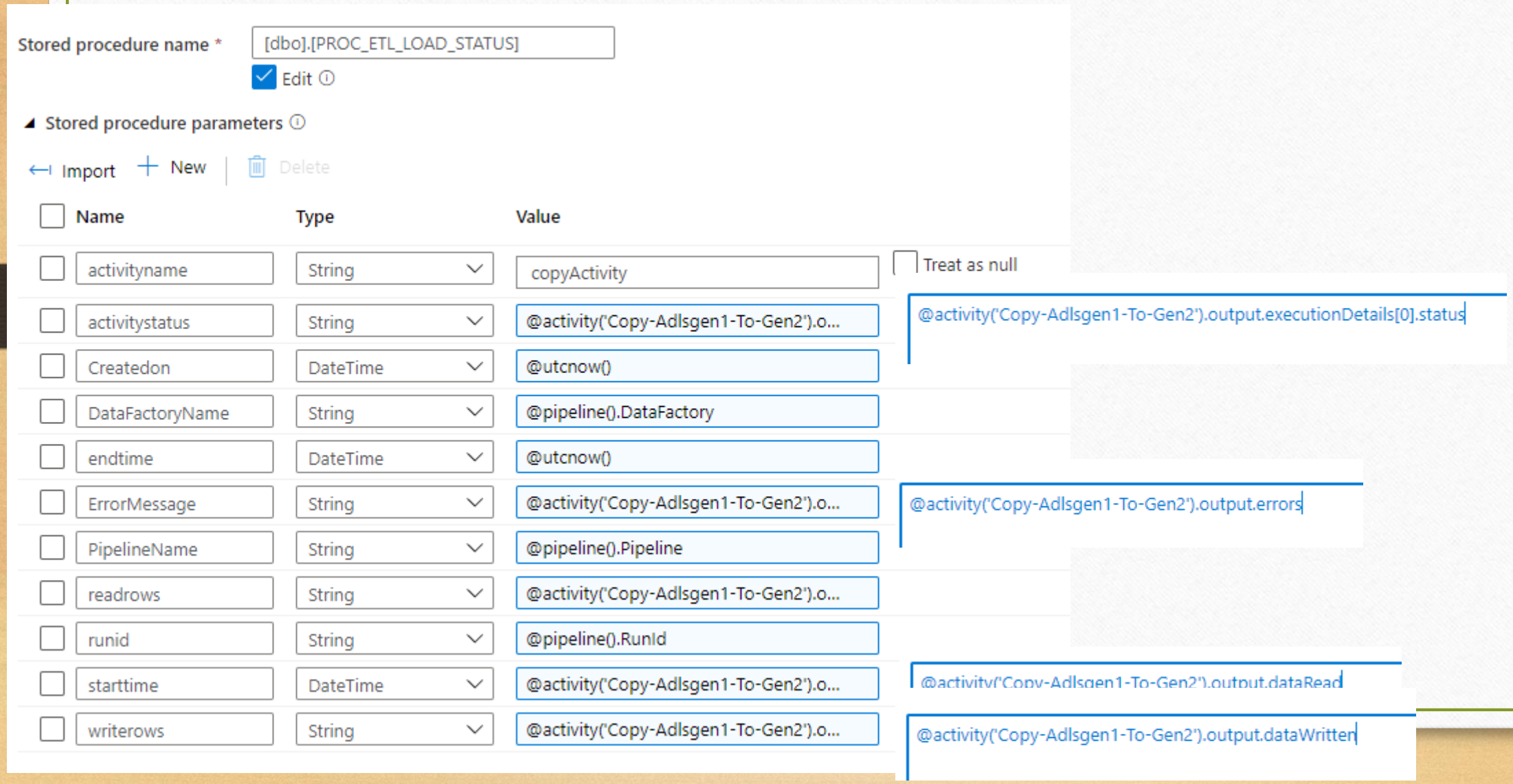

child_load_pipeline
Source_ADLS_DS
Saved
01
Binary
Source_ADLS_DS
Connection
Parameters
New
Delete
Name
Type
Default value
FolderPath
String
Value
Directory
String
Value
filename
String
Value


child_load_pipeline
Activities
Search activities
Move & transform
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Save
Save as template
Validate
Validate copy runtime
Debug
Add trigger
child_load_pipeline > ForEach-process1by1
Copy data
OnPrem_To_Azure
Copy data
Copy-Adlsgen1-To-Gen2
Stored procedure
Stored procedure1
General
Source
Sink
Mapping
Settings
User properties
Sink dataset *
Target_BLOB_DS
Open
New
Learn more
Dataset properties
Name
Value
Type
FolderPath
@pipeline().parameters.TargetStore_Lo...
string
Directory
@pipeline().parameters.TargetStore_Di...
string
filename
*.csv
string
Copy behavior
None
Max concurrent connections
Block size (MB)


child_load_pipeline
Target_BLOB_DS
Saved
01
Binary
Target_BLOB_DS
Connection
Parameters
Linked service *
BLOB_STORAGE_GEN2_LS
Test connection
Edit
New
Learn more
File path *
@concat(dataset().FolderPath)
Directory
File
Browse
Compression type
None


child_load_pipeline
Target_BLOB_DS
Saved
Binary
Target_BLOB_DS
Connection
Parameters
New
Delete
Name
Type
Default value
FolderPath
String
Value
Directory
String
Value
filename
String
Value


child_load_pipeline
Activities
Search activities
Move & transform
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Save
Save as template
Validate
Debug
Add trigger
child_load_pipeline > ForEach-process1by1
Copy data
OnPrem_To_Azure
Copy data
Copy-Adlsgen1-To-Gen2
Stored procedure
update-log-proc
General
Settings
User properties
Linked service *
Metadata_SQLDB_LS
Test connection
Edit
New
Integration runtime *
integrationRuntime
Stored procedure name *
[dbo].[PROC_ETL_LOAD_STATUS]
Edit
Stored procedure parameters
Import
New
Delete
Name
Type
Value
activityname
String
copyActivity
Treat as null
activitystatus
String
@activity('Copy-Adlsgen1-To-Gen2').o...
Createdon
DateTime
@utcnow()
DataFactoryName
String
@pipeline().DataFactory


Stored procedure name *
[dbo].[PROC_ETL_LOAD_STATUS]
Edit
Stored procedure parameters
Import
New
Delete
Name
Type
Value
activityname
String
copyActivity
Treat as null
activitystatus
String
@activity('Copy-Adlsgen1-To-Gen2').o...
@activity('Copy-Adlsgen1-To-Gen2').output.executionDetails[0].status
Createdon
DateTime
@utcnow()
DataFactoryName
String
@pipeline().DataFactory
endtime
DateTime
@utcnow()
ErrorMessage
String
@activity('Copy-Adlsgen1-To-Gen2').o...
@activity('Copy-Adlsgen1-To-Gen2').output.errors
PipelineName
String
@pipeline().Pipeline
readrows
String
@activity('Copy-Adlsgen1-To-Gen2').o...
runid
String
@pipeline().RunId
starttime
DateTime
@activity('Copy-Adlsgen1-To-Gen2').o...
writerows
String
@activity('Copy-Adlsgen1-To-Gen2').o...
@activity('Copy-Adlsgen1-To-Gen2').output.dataWritten

Naming Conversions

Type	Linked Service	Name	Linked Service2	Dataset
Azure	Azure Blob storage	ABLB_	LS_ABLB_	DS_ABLB_
	Azure Data Lake Store	ADLS_	LS_ADLS_	DS_ADLS_
	Azure SQL Database	ASQL_	LS_ASQL_	DS_ASQL_
	Azure SQL Data Warehouse	ASDW_	LS_ASDW_	DS_ASDW_
	Azure Table storage	ATBL_	LS_ATBL_	DS_ATBL_
	Azure DocumentDB	ADOC_	LS_ADOC_	DS_ADOC_
	Azure Search Index	ASER_	LS_ASER_	DS_ASER_
Databases	SQL Server*	MSQL_	LS_SQL_	DS_SQL_
	Oracle*	ORAC_	LS_ORAC_	DS_ORAC_
File	File System*	FILE_	LS_FILE_	DS_FILE_
	HDFS*	HDFS_	LS_HDFS_	DS_HDFS_
	Amazon S3	AMS3_	LS_AMS3_	DS_AMS3_
	FTP	FTP_	LS_FTP_	DS_FTP_
Others	Salesforce	SAFC_	LS_SAFC_	DS_SAFC_
	Generic ODBC*	ODBC_	LS_ODBC_	DS_ODBC_
	Generic OData	ODAT_	LS_ODAT_	DS_ODAT_
	Web Table (table from HTML)	WEBT_	LS_WEBT_	DS_WEBT_
	GE Historian*	GEHI_	LS_GEHI_	DS_GEHI_

Type	Name	Action	Example
Data movement Activity	PL_DATA_	NA	PL_DATA_DS_SQL_Person_To_DS_ABLB_Person
Data transformation pipeline	PL_TRAN_	SPRC – Stored Procedure	PL_TRAN_SPRC_CleanDimAccount
	PL_TRAN_	DNET – Script	PL_TRAN_DNET_AggregateSales
	PL_TRAN_	ADLK – Azure Data Lake	PL_TRAN_ADLK_AggregateSales
	PL_TRAN_	HIVE – Hive	PL_TRAN_HIVE_AggregateSales
	PL_TRAN_	PIG – Pig	PL_TRAN_PIG_AggregateSales
	PL_TRAN_	MAPR – MapReduce	PL_TRAN_MAPR_AggregateSales
	PL_TRAN_	HADP – Hadoop Stream	PL_TRAN_HADP_StreamData
	PL_TRAN_	AML – Azure Machine Learning	PL_TRAN_AML_CalculateMonthlyChurn

Edit trigger

Name *

trigger4

Description

Type *

TumblingWindowTrigger

Start Date (UTC) *

05/25/2021 10:50 AM

Recurrence * ⓘ

Every 15 Minute(s)

Specify an end date

Advanced

Annotations

+ New

Activated * ⓘ

Yes No

Edit trigger

Name *

trigger5

Description

Type *

TumblingWindowTrigger

Start Date (UTC) *

05/25/2021 11:52 AM

Recurrence * ⓘ

Every 15 Minute(s)

Specify an end date

Advanced

Add dependencies

New | Delete

TRIGGER	OFFSET	WINDOW SIZE
trigger4	0.00:13:00	0.00:10:00

Delay ⓘ

00:00:00

All | Schedule | **Tumbling window** | Storage events | Custom events (preview) | Refresh | Edit columns

Chennai, Kolkata, Mu... : **Last 24 hours** | Trigger name : **All** | Status : **All** | **Runs** : Latest runs

Showing 1 - 100 items

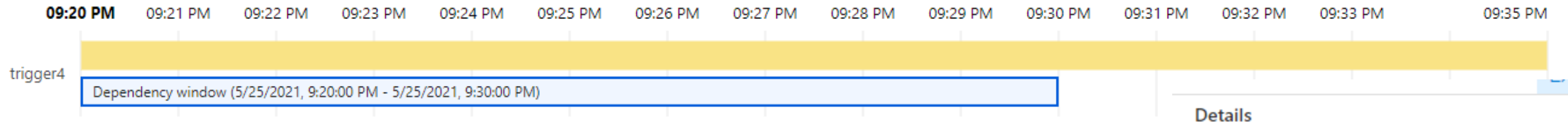
Trigger name	Window start time	Window end time	Trigger time	Status	Pipelines
trigger5	5/25/21, 9:07:00 PM	5/25/21, 9:22:00 PM	5/25/21, 9:22:00 PM	Waiting on dependency	0
trigger4	5/25/21, 9:05:00 PM	5/25/21, 9:20:00 PM	5/25/21, 9:20:00 PM	Succeeded	1
trigger5	5/25/21, 8:52:00 PM	5/25/21, 9:07:00 PM	5/25/21, 9:06:59 PM	Succeeded	1
trigger4	5/25/21, 8:50:00 PM	5/25/21, 9:05:00 PM	5/25/21, 9:04:59 PM	Succeeded	1
trigger5	5/25/21, 8:37:00 PM	5/25/21, 8:52:00 PM	5/25/21, 8:52:00 PM	Succeeded	1
trigger4	5/25/21, 8:35:00 PM	5/25/21, 8:50:00 PM	5/25/21, 8:50:00 PM	Succeeded	1
trigger5	5/25/21, 8:22:00 PM	5/25/21, 8:37:00 PM	5/25/21, 8:37:00 PM	Succeeded	1
trigger4	5/25/21, 8:20:00 PM	5/25/21, 8:35:00 PM	5/25/21, 8:35:00 PM	Succeeded	1

trigger5

Window (5/25/2021, 9:07:00 PM - 5/25/2021, 9:22:00 PM)

Refresh | Rerun | Time zone : Chennai, Kolkata, Mumbai, New ... | List | Gantt

Upstream run dependencies



Details

Run ID

085857964986573215962233743 65CU24

Window

5/25/2021, 9:07:00 PM - 5/25/2021, 9:22:00 PM

Run start

5/26/2021, 2:52:00 AM

Status

Waiting on dependency

Dependent triggers

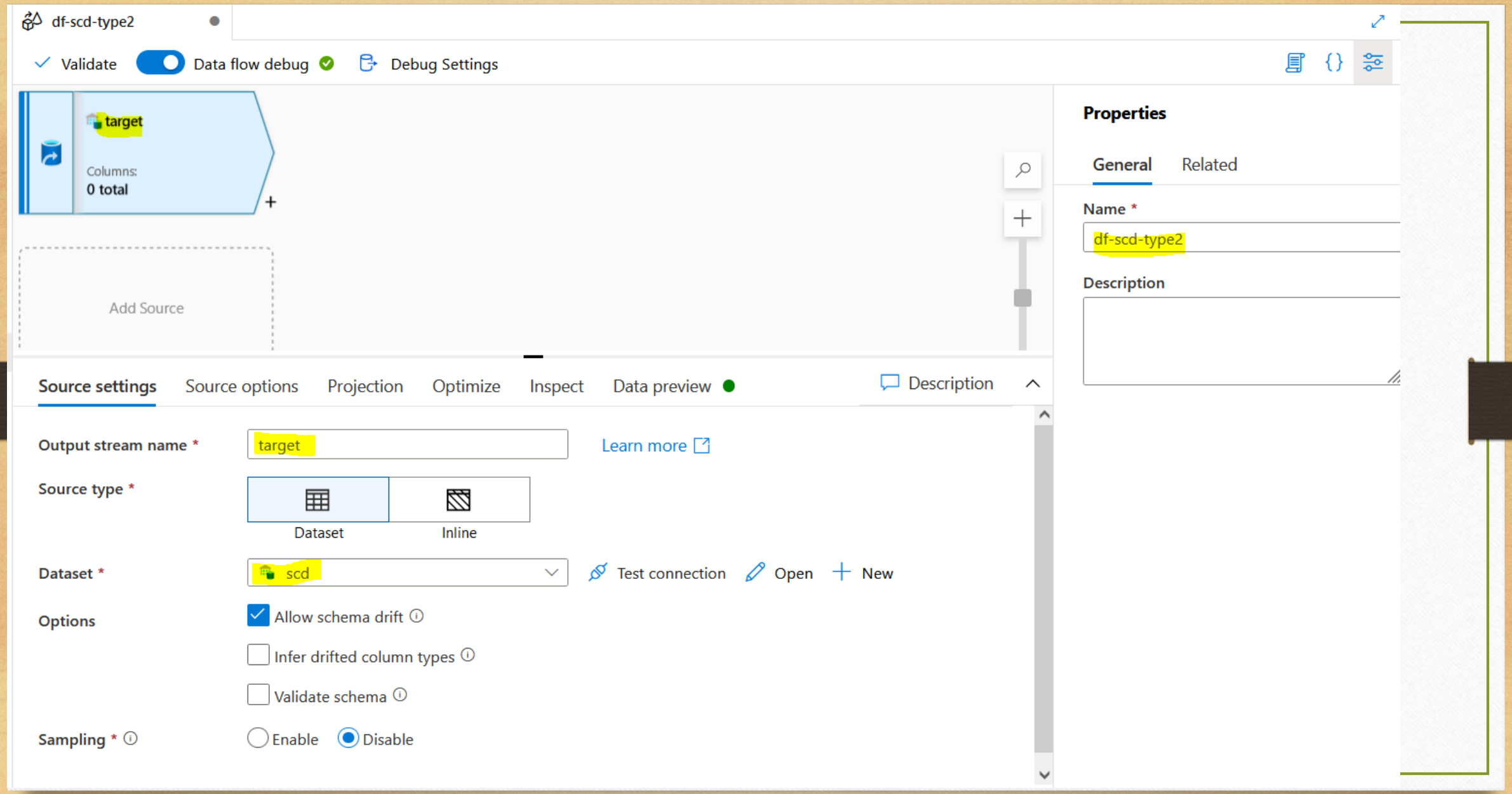
1

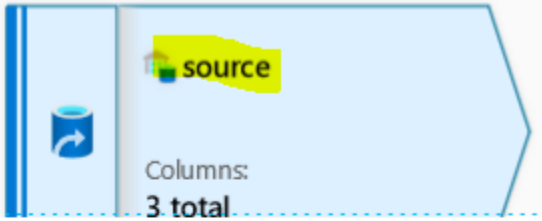
Type

TumblingWindowTrigger

Triggered pipeline run

SCD-type 2 Implementation


df-scd-type2
Validate
Data flow debug
Debug Settings
target
Columns:
0 total
Add Source
Properties
General
Related
Name *
df-scd-type2
Description
Source settings
Source options
Projection
Optimize
Inspect
Data preview
Description
Output stream name *
target
Learn more
Source type *
Dataset
Inline
Dataset *
scd
Test connection
Open
New
Options
Allow schema drift
Infer drifted column types
Validate schema
Sampling *
Enable
Disable



Source settings | Source options | Projection | Optimize | Inspect

Output stream name * source

Source type * Dataset | Inline

Dataset * scd_source

Options Allow schema drift

Exists settings | Optimize | Inspect | Data preview

Output stream name * Exists1

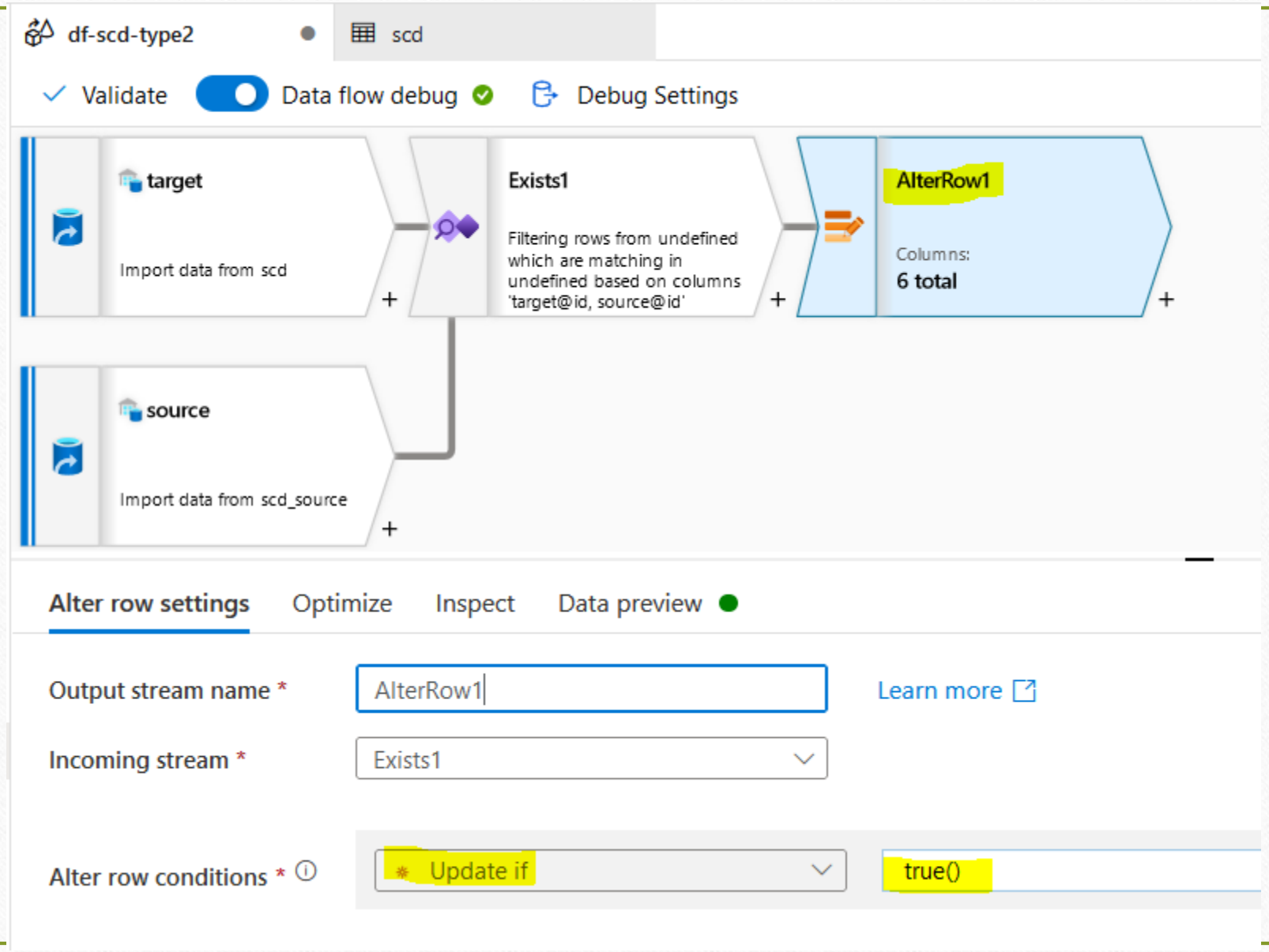
Left stream * target

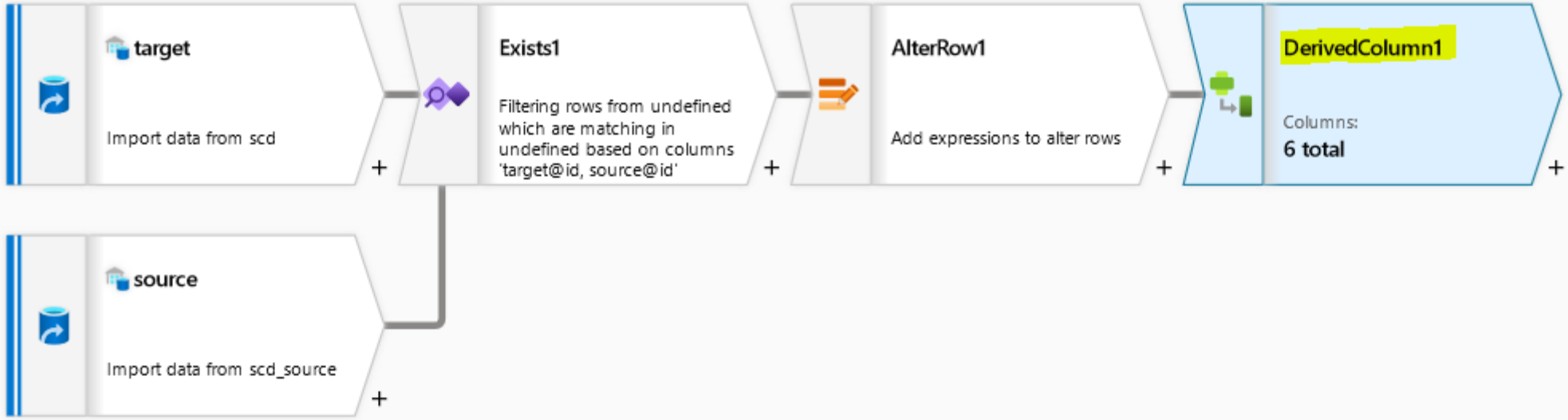
Right stream * source

Exist type * Exists | Doesn't exist

Custom expression

Exists conditions * and(target@id==source@id,target@loc!=source@loc)


df-scd-type2
scd
Validate
Data flow debug
Debug Settings
target
Import data from scd
Exists1
Filtering rows from undefined which are matching in undefined based on columns 'target@id, source@id'
AlterRow1
Columns:
6 total
source
Import data from scd_source
Alter row settings
Optimize
Inspect
Data preview
Output stream name *
AlterRow1
Learn more
Incoming stream *
Exists1
Alter row conditions *
Update if
true()



Derived column's settings | Optimize | Inspect | Data preview

Output stream name * DerivedColumn1 Learn more

Incoming stream * AlterRow1

+ Add | Clone | Delete | Open expression builder

Columns *

Column	Expression
status	0
end_time	currentTimestamp()

Derived column's settings | Optimize | Inspect | Data preview

Number of rows | INSERT 0 | UPDATE 1 | DELETE 0 | UPSERT 0 | LOOKUP 0

Refresh | Typecast | Modify | Map drifted | Statistics | Remove

id 123	name abc	loc abc	status 123	start_date	end_time
4	Prasad	Pune	0	2020-01-01 00:00:00	2021-06-20 12:07:20

Sink | Settings | Mapping | Optimize | Inspect | Data preview

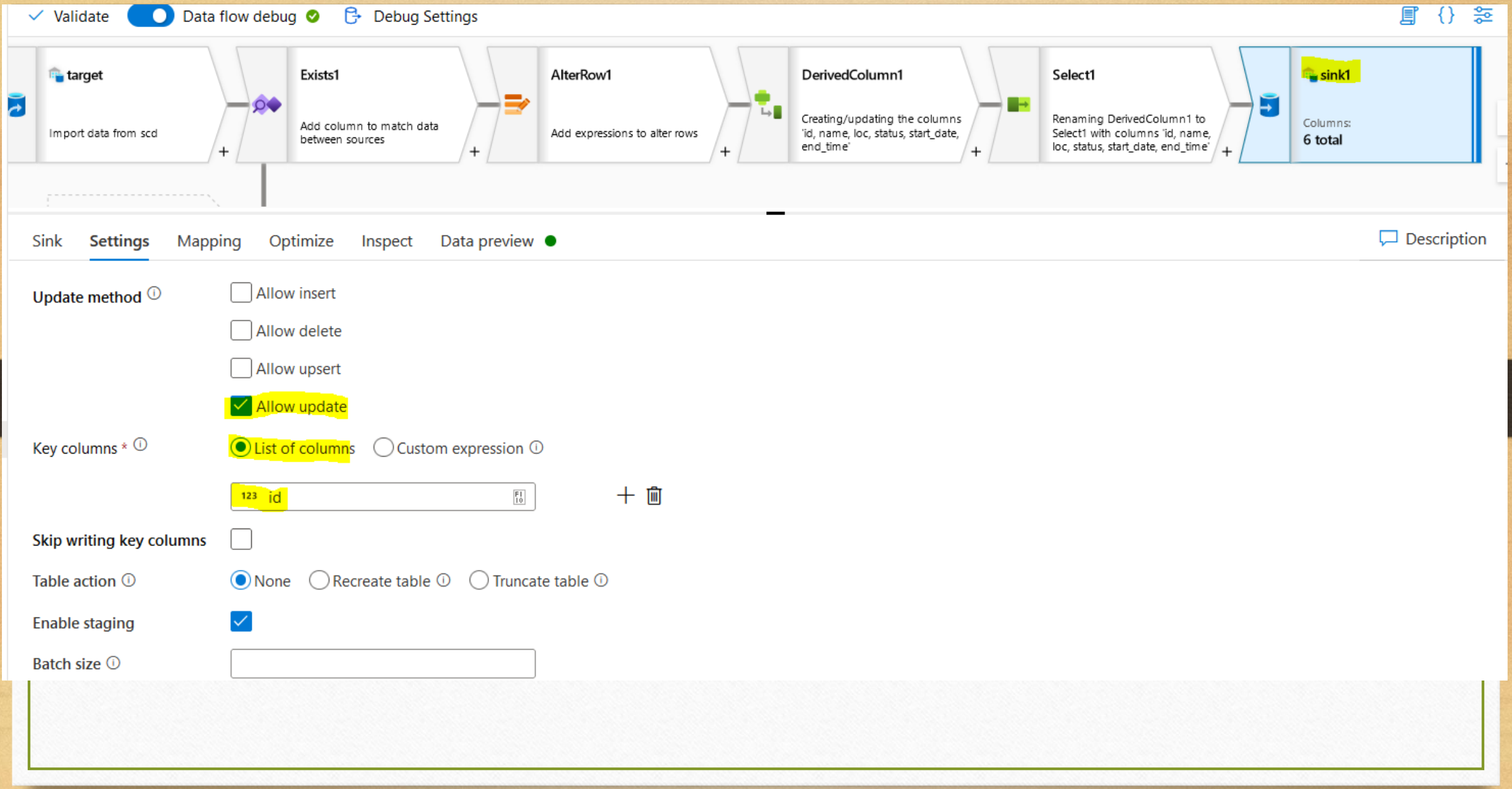
Output stream name * sink1 Learn more

Incoming stream * DerivedColumn1

Sink type * Dataset | Inline | Cache

Dataset * scd Test connection | Open | New

Options
- [x] Allow schema drift
- [ ] Validate schema


Validate
Data flow debug
Debug Settings
target
Import data from scd
Exists1
Add column to match data between sources
AlterRow1
Add expressions to alter rows
DerivedColumn1
Creating/updating the columns 'id, name, loc, status, start_date, end_time'
Select1
Renaming DerivedColumn1 to Select1 with columns 'id, name, loc, status, start_date, end_time'
sink1
Columns:
6 total
Sink
Settings
Mapping
Optimize
Inspect
Data preview
Description
Update method
Allow insert
Allow delete
Allow upsert
Allow update
Key columns *
List of columns
Custom expression
123 id
Skip writing key columns
Table action
None
Recreate table
Truncate table
Enable staging
Batch size

Derived column's settings | Optimize | Inspect | Data preview

Output stream name * DerivedColumn2 Learn more

Incoming stream * source

+ Add | Clone | Delete | Open expression builder

Columns *

Column	Expression
status	1
start_date	currentTimestamp()
end_date	toTimestamp('9999-12-31 00:00:00.000')

Derived column's settings | Optimize | Inspect | **Data preview** ● | Descri

Number of rows + INSERT 4 | * UPDATE 0 | × DELETE 0 | UPSERT 0 | LOOKUP 0 | TOTA

Refresh | Typecast | Modify | Map drifted | Statistics | Remove | ⓘ

	id 123	name abc	loc abc	status 123	start_date	end_date
+	6	Reshwanth	Bangalore	0	2021-06-20 12:10:58	9999-12-31 00:00:00
+	4	Prasad	Chennai	0	2021-06-20 12:10:58	9999-12-31 00:00:00
+	7	Sekhar	Bangalore	0	2021-06-20 12:10:58	9999-12-31 00:00:00
+	5	Vikranth	Bangalore	0	2021-06-20 12:10:58	9999-12-31 00:00:00

Select settings | Optimize | Inspect | Data preview

Output stream name * Select Learn more

Incoming stream * DerivedColumn2

Options
- Skip duplicate input columns
- Skip duplicate output columns

Input columns * Auto mapping | Reset | Add mapping | Delete 6

DerivedColumn2's column	Name as
id	id
name	name
loc	loc
status	status
start_date	start_date
end_date	end_date

Sink | Settings | Mapping | Optimize | Inspect | Data preview

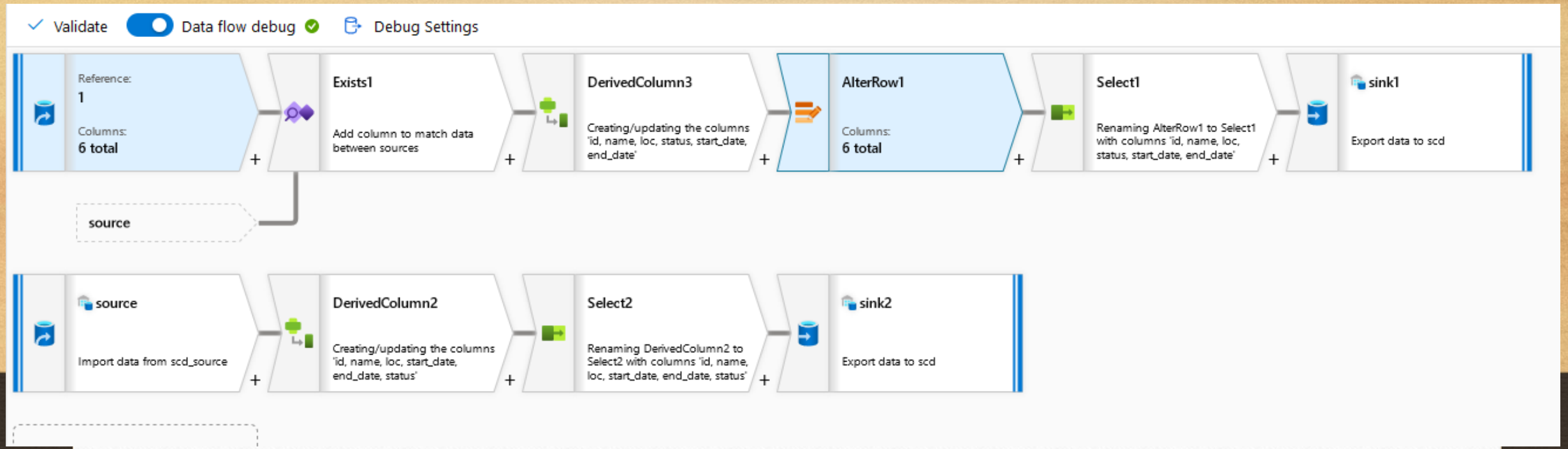
Output stream name * scdtarget Learn more

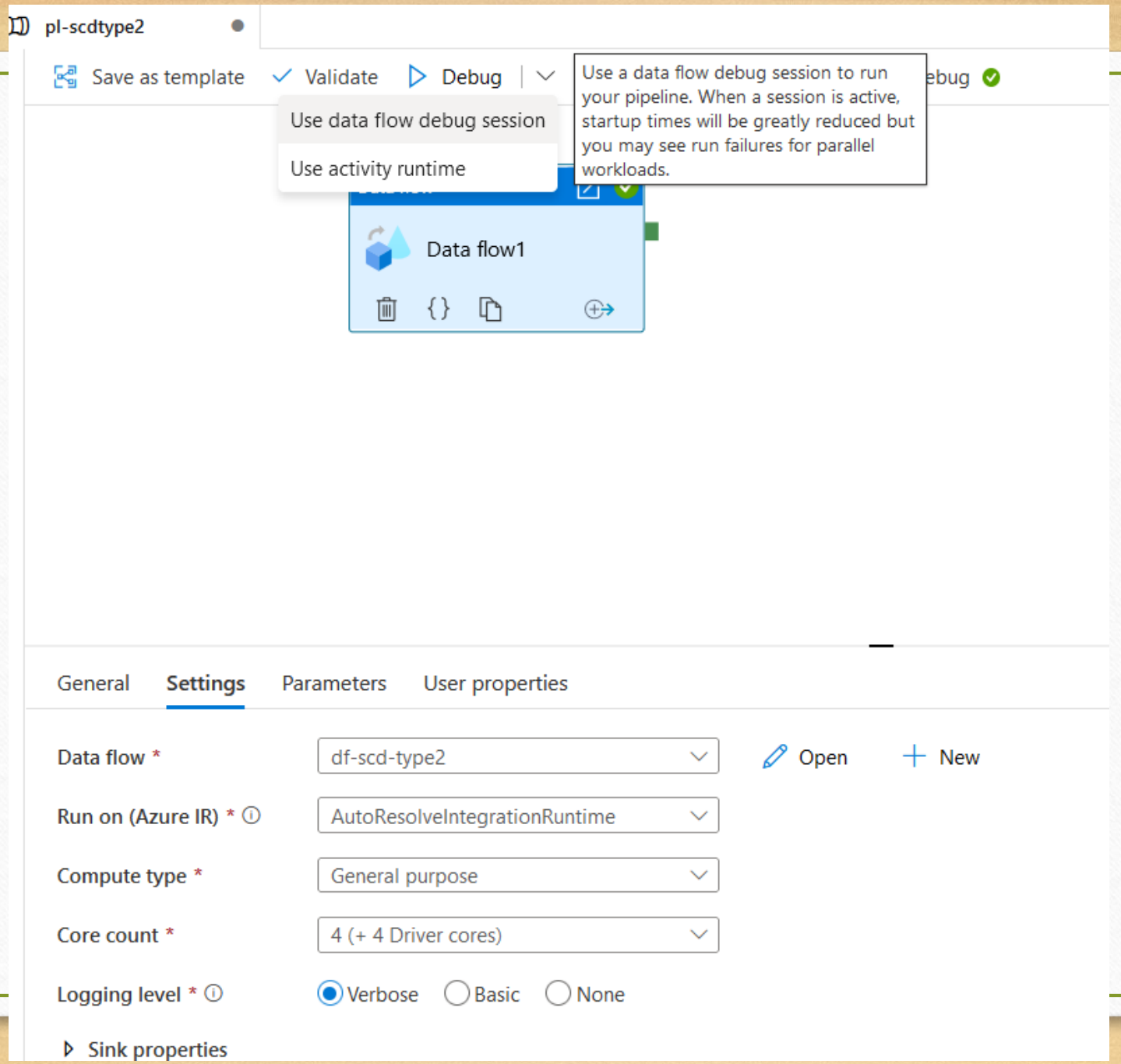
Incoming stream * Select

Sink type * Dataset | Inline | Cache

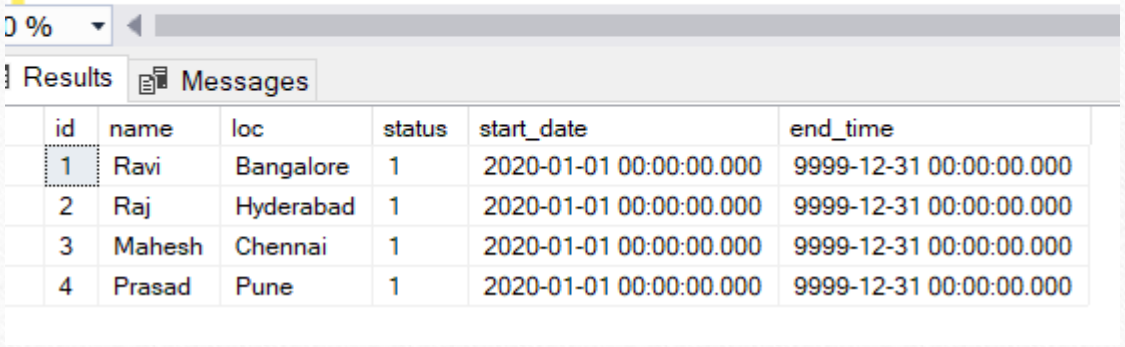
Dataset * scd Test connection | Open | New

Options
- [x] Allow schema drift
- [ ] Validate schema


Validate
Data flow debug
Debug Settings
Reference:
1
Columns:
6 total
Exists1
Add column to match data between sources
DerivedColumn3
Creating/updating the columns 'id, name, loc, status, start_date, end_date'
AlterRow1
Columns:
6 total
Select1
Renaming AlterRow1 to Select1 with columns 'id, name, loc, status, start_date, end_date'
sink1
Export data to scd
source
source
Import data from scd_source
DerivedColumn2
Creating/updating the columns 'id, name, loc, start_date, end_date, status'
Select2
Renaming DerivedColumn2 to Select2 with columns 'id, name, loc, start_date, end_date, status'
sink2
Export data to scd


pl-scdtype2
Save as template
Validate
Debug
Use data flow debug session
Use activity runtime
Use a data flow debug session to run your pipeline. When a session is active, startup times will be greatly reduced but you may see run failures for parallel workloads.
Data flow1
General
Settings
Parameters
User properties
Data flow *
df-scd-type2
Open
New
Run on (Azure IR) *
AutoResolveIntegrationRuntime
Compute type *
General purpose
Core count *
4 (+ 4 Driver cores)
Logging level *
Verbose
Basic
None
Sink properties

```
select* from cust_dim order by id
```



Results | Messages

id	name	loc	status	start_date	end_time
1	Ravi	Bangalore	1	2020-01-01 00:00:00.000	9999-12-31 00:00:00.000
2	Raj	Hyderabad	1	2020-01-01 00:00:00.000	9999-12-31 00:00:00.000
3	Mahesh	Chennai	1	2020-01-01 00:00:00.000	9999-12-31 00:00:00.000
4	Prasad	Pune	1	2020-01-01 00:00:00.000	9999-12-31 00:00:00.000