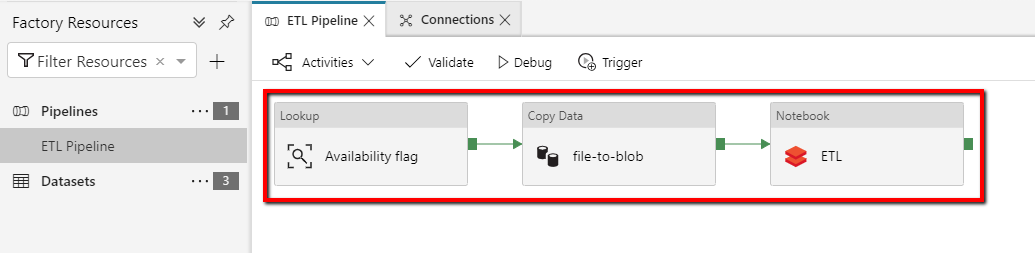
ETL with Azure Databricks – Setup Instructions

# Overview

In this lab, you will create an end-to-end pipeline containing **Lookup**, **Copy** and **Databricks** **notebook** activities in ADF.

* **Lookup** or GetMetadata activity is used to ensure the source dataset is ready for downstream consumption, before triggering the copy and analytics job.
* **Copy activity,** copies the source file/ dataset to the sink storage. The sink storage is mounted as DBFS in the databricks notebook so that the dataset can be directly consumed by Spark.
* **Databricks notebook activity,** triggers the databricks notebook that transforms the dataset, and adds it to a processed folder/ SQL DW.

We will not create a scheduled trigger in this template for simplicity, and you may add that if required.

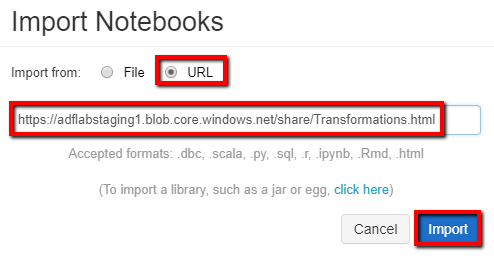


# Prerequisite:

1. Create a **blob storage account** and a container called ‘**sinkdata**’ to be used as **sink**. Keep a note of the **storage account name**, **container name** and **Access key** will be referenced later in the template.
2. Ensure you have an **Azure Databricks workspace** or create a new one.
   1. Import the below Transform notebook to the databricks workspace. (need not be the same location as below, just remember the path for later usage)

**ACTION REQUIRED** – **Import the notebook for ETL**. Import the notebook from URL. Please enter the below URL in the URL field: <https://adflabstaging1.blob.core.windows.net/share/Transformations.html>. Select Import.

Machine generated alternative text:
C O https://eastus.azuredatabricks.net/?o=6964761357957243 
Microsoft Azure 
Workspace 
Azure 
Databricks 
Home 
Recents 
Workspace 
? Documentation 
</> Release Notes 
*Training & Tutorials 
Shared 
Users 
adftutorial 
adftutorial 
Create 
Clone 
Rename 
g abc.jar 
Move 
g custom 
Move to Trash 
g custom 
Import 
Datase 
xpo 
ETL-P 
Permissions 
ETL-P 
tabricks 
utorial 



1. Now let’s update the **'Transformation'** notebook with your **storage connection information** (name and access key). Go to **command 5** in the imported notebook above, replace it with the below code snippet after replacing the highlighted values. *Ensure this is the same storage account created earlier and contains the* ***sinkdata*** *container.*
2. #Supply storageName and accessKey values
3. storageName = "<storage name>"
4. accessKey = "<access key>"
6. **try**:
7. dbutils.fs.mount(
8. source = "wasbs://sinkdata@"+storageName+".blob.core.windows.net/",
9. mount\_point = "/mnt/adfdata",
10. extra\_configs = {"fs.azure.account.key."+storageName+".blob.core.windows.net":
11. accessKey})
12. **except** Exception as e:
13. # The error message has a long stack trace.  This code tries to print just the relevent line indicating what failed.
14. **import** re
15. result = re.findall(r"^\s\*Caused by:\s\*\S+:\s\*(.\*)$", e.message, flags=re.MULTILINE)
16. **if** result:
17. **print** result[-1] # Print only the relevant error message
18. **else**:
19. **print** e # Otherwise print the whole stack trace.

1. Generate a **databricks access token** for ADF to access databricks. **Save the access token** for later use in creating a databricks linked service, which looks something like 'dapi32db32cbb4w6eee18b7d87e45exxxxxx'

Machine generated alternative text:
Microsoft Azure 
Azure 
Databricks 
Horne 
PORTAL 
e Azure Databricks 
abnarain@microsoft.com 
Signed in as 
abnarain@mic oft.com 
user Setfings 
Acmin Console 
Manage Account 
Log Out 

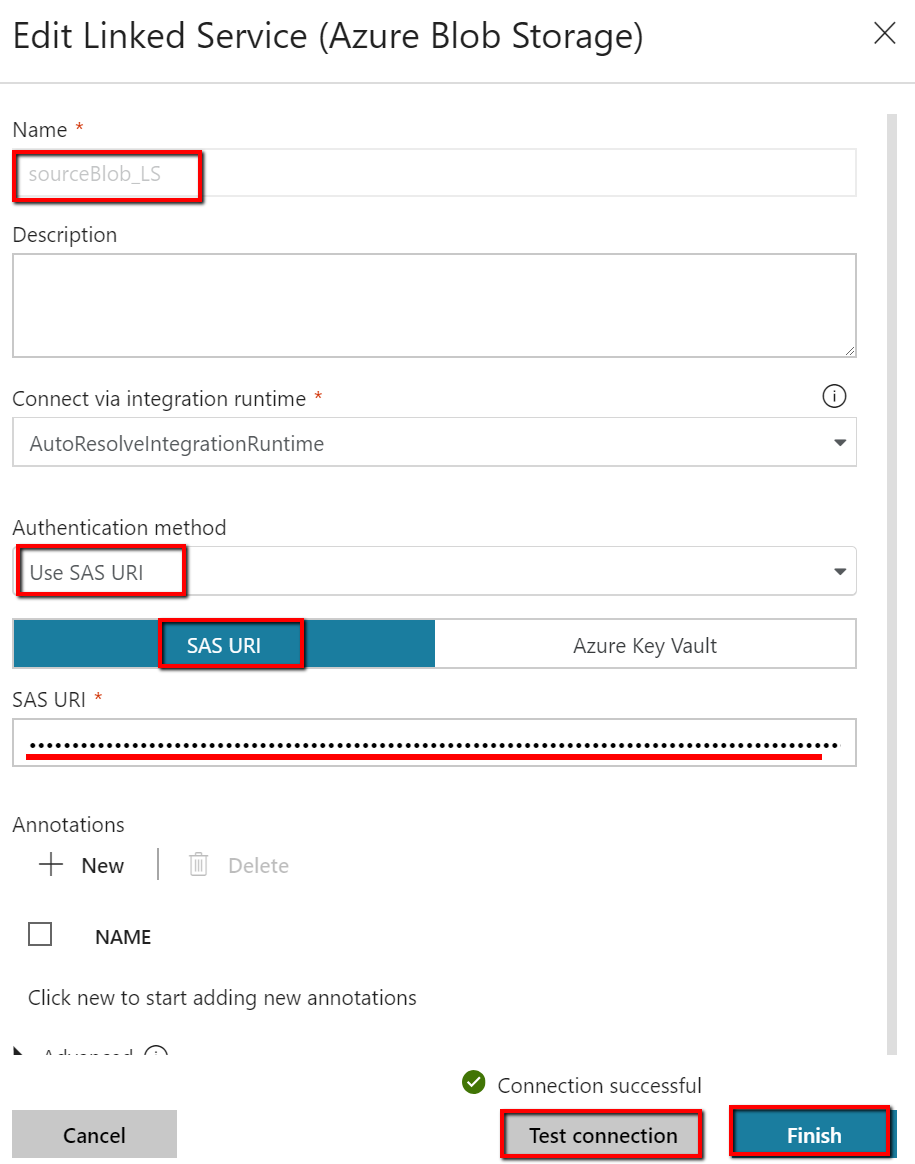
Machine generated alternative text:
User Settings 
Access Tokens Git Integration 
Notebook Settings 
Personal access tokens can be used for secure authentication to the Databricks API 
Generate New Token 
Token ID 
69fe4ecab7859f39beaOa4d0895285b277cc767a8f2f2ae3e948.. 
Obi a29900d6e432aObca060c1da4.. 
82dbc1262306cbac86e3abec5f12b4eb871f8cfc10111fa61bd8... 
12a0362362c358d3fOOa358007bd1df10d09920ab1fa0526609.. 
Comment 
s 
databricksAcce 
asa 
. for adf 
Generate New Token 
Comment 
adf access 
Lifetime (days) O 
Cancel 
Generate 

## **Create Linked Services and datasets**

1. Create new **linked services** in ADF UI by going to *Connections 🡪 Linked services 🡪 + new*
   1. **Source** – for accessing source data. You can use the public blob storage containing the source files for this sample.

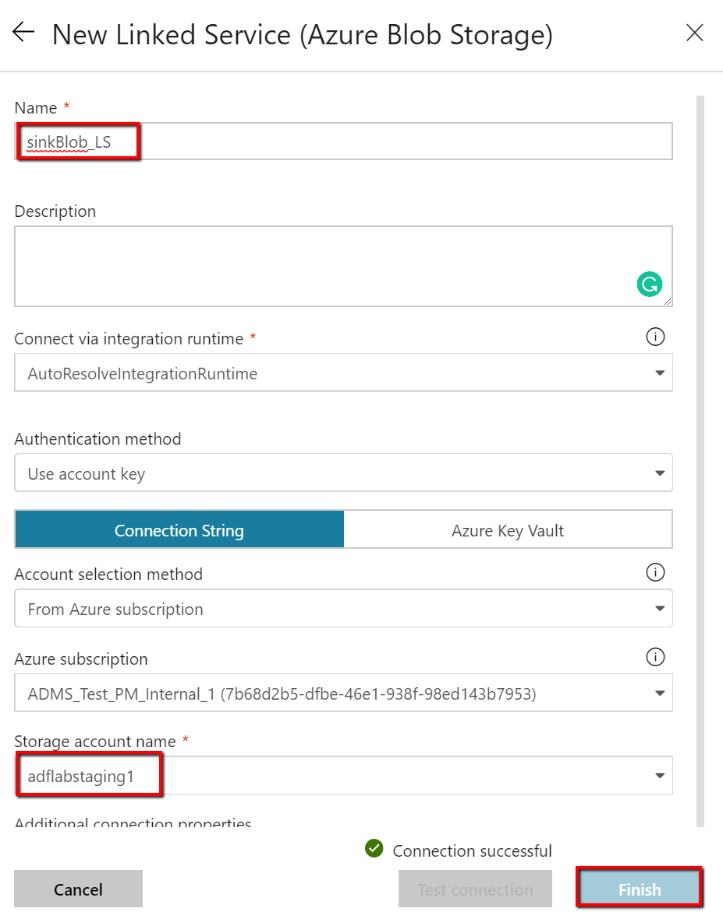
Select **Blob Storage**, use the below **SAS URI** to connect to source storage (read-only access).

*https://storagewithdata.blob.core.windows.net/?sv=2017-11-09&ss=b&srt=sco&sp=rl&se=2019-12-31T21:40:53Z&st=2018-10-24T13:40:53Z&spr=https&sig=K8nRio7c4xMLnUV0wWVAmqr5H4P3JDwBaG9HCevI7kU%3D*



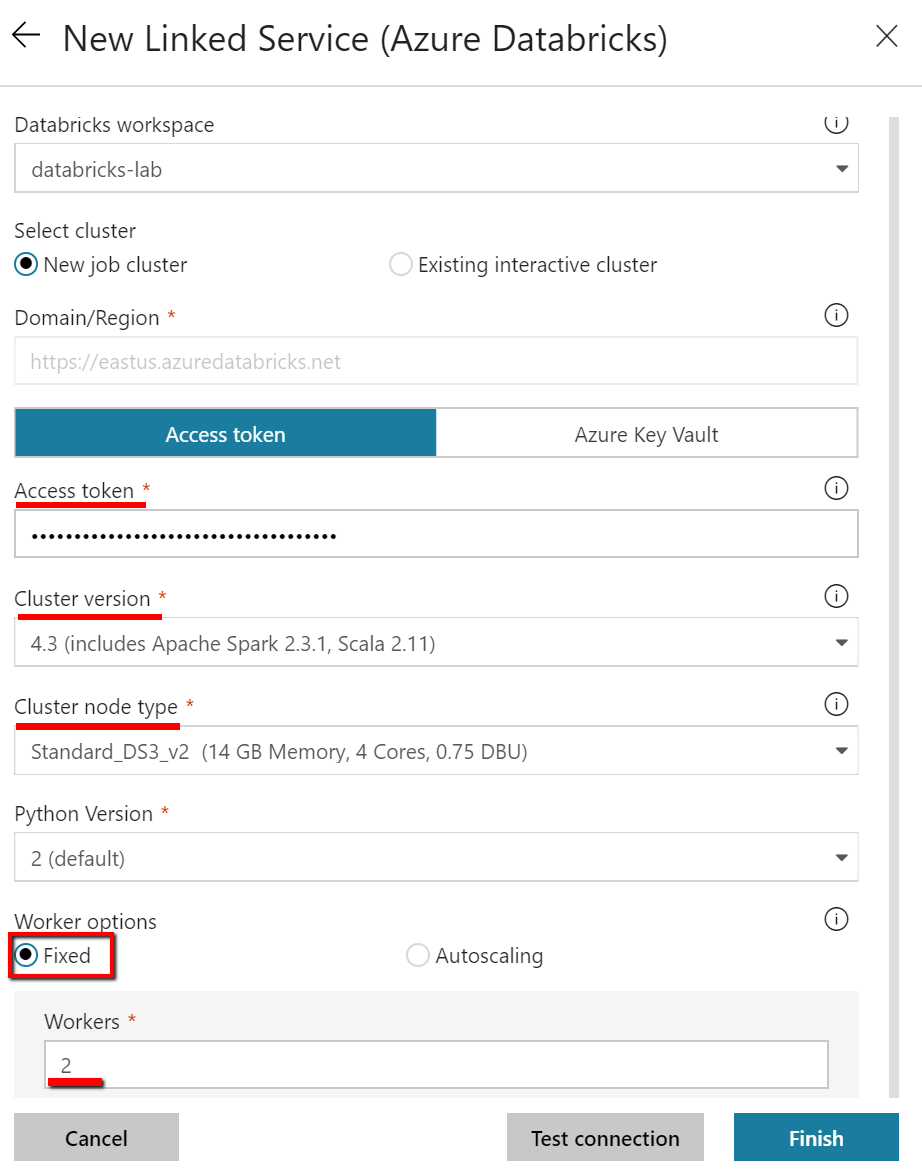
* 1. **Sink** – for copying data into.

Please select a storage created in the [prerequisite 1](#_Prerequisite:), in the sink linked service.



* 1. **Databricks** – for connecting to databricks cluster

Create a databricks linked service using access key generated in [prerequisite 2.c](#_Prerequisite:). If you have an *interactive cluster*, you may select that. (in this case we use a *New job cluster* option)



1. Create **datasets**
   1. Create **'sourceAvailability\_Dataset'** to check if source data is available

Machine generated alternative text:
Azure Blob Storage 
sourceAvaiIabiIity_Dataset 
General 
Connection 
Linked service * 
Schema 
sourceBlob LS 
data/source 
None 
Parameters 
success 
Test connection 
Browse 
Edit 
+ New 
File path 
Compression Type 
[3 Binary Copy O 
Q) 
Preview data 

1. **Source dataset –** for copying the source data (using binary copy)

Machine generated alternative text:
Azure Blob Storage 
sourceFiIes Dataset 
General 
Connection 
Linked service * 
Schema 
sourceBlob LS 
data/source 
None 
Parameters 
Product.csv 
Test connection 
Browse 
Edit 
+ New 
File path 
Compression Type 
Binary Copy O 
Q) 
Preview data 

1. **Sink dataset** – for copying into the sink/ destination location
   1. Linked service - select 'sinkBlob\_LS' created in 1.b
   2. File path - 'sinkdata/staged\_sink'

Machine generated alternative text:
Azure Blob Storage 
sinkRawFiIes Dataset 
General 
Connection 
Linked service * 
Schema 
sinkBlob LS 
sinkdata/staged sink 
None 
Parameters 
File 
Test connection 
Browse 
Edit 
+ New 
File path 
Compression Type 
Binary Copy O 
Q) 
Preview data 

**Create activities**

1. a. Create a Lookup activity '**Availability flag**' for doing a Source Availability check (Lookup or GetMetadata can be used). Select 'sourceAvailability\_Dataset' created in 2.a.

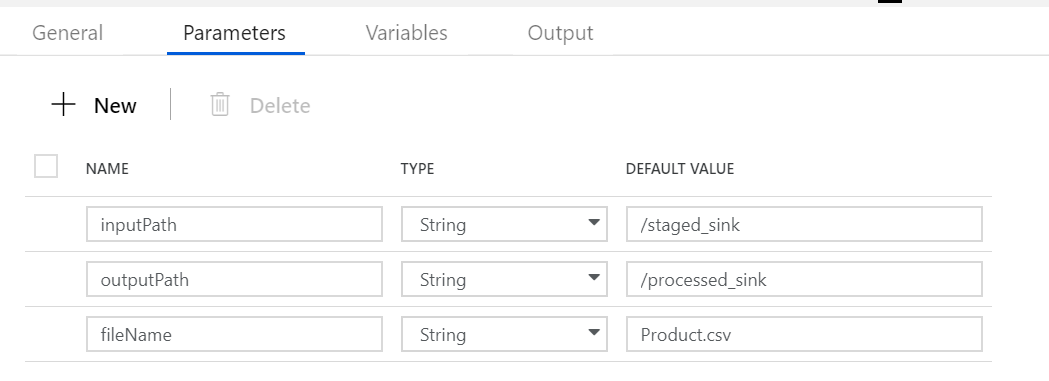
Machine generated alternative text:
v/ Validate 
Lookup 
Debug 
General 
Availability flag 
Settings 
Trigger 
O 
User Properties 
sourceAvailability Dataset 
Source Dataset * 
First row only 
Edit 
+ New 
Preview data 

1. Create a Copy activity '**file-to-blob**' for copying dataset from source to sink. In this case the data is binary file. Reference the below screenshots for source and sink configurations in the copy activity.

Machine generated alternative text:
v/ Validate 
Lookup 
Debug 
General 
Availability flag 
Source 
Trigger 
Sink 
sourceFiles 
Copy Data 
file-to-blob 
Mapping 
Dataset 
O 
9.5 
Settings 
Edit 
User Properties 
+ New 
Source Dataset * 
Copy file recursively O 
Preview data 

Machine generated alternative text:
v/ Validate 
Lookup 
Debug 
Trigger 
Copy Data 
file-to-blob 
Availability flag 
General 
Sink Dataset * 
Copy behavior 
Source 
Sink 
sinkRawFiles 
None 
Mapping 
Dataset 
O 
9.5 
Settings 
Edit 
User Properties 
+ New 

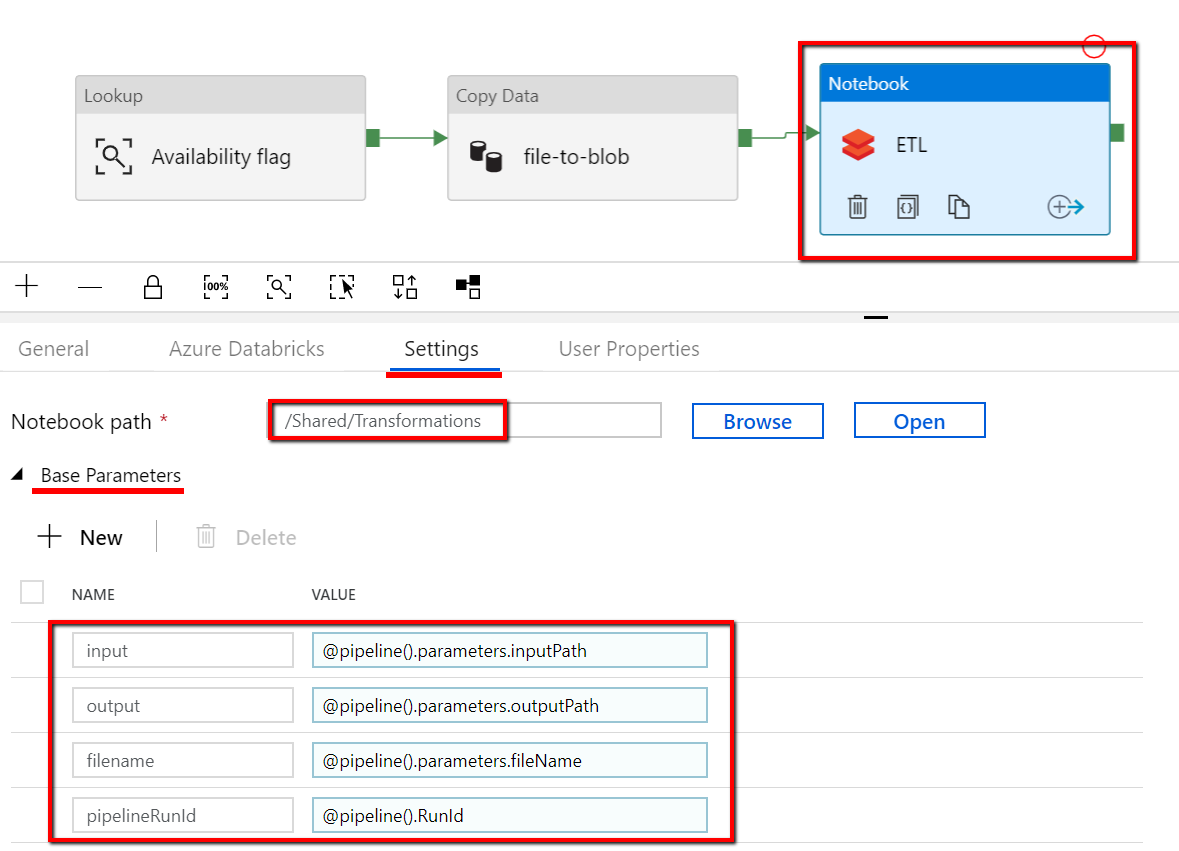
1. Define **pipeline parameters**



1. Create a **databricks activity**
   * Select the linked service created in 1.c

Machine generated alternative text:
Lookup 
Availability flag 
Copy Data 
file-to-blob 
General 
Azure Databricks 
Settings 
User Properties 
Databricks Linked Service * 
AzureDatabricks LS 
Notebook 
e Connection successful 
Test connection 
Edit 
+ New 

* Configure the **settings**. Create **Base Parameters** as shown in the screenshot and create parameters to be passed to the databricks notebook from ADF. Browse and **select** the **correct notebook path** uploaded in **prerequisite 2**.



* **Run the pipeline**. You can find link to databricks logs for more detailed spark logs.

Machine generated alternative text:
v/ Validate 
Lookup 
Debug 
Ck Availability flag 
Parameters 
Trigger 
Variables 
Copy Data 
file-to-blob 
Output 
Notebook 
ETL 
Output 
'runPageUrl": "https://eastus.azuredatabricks.net/? 
0-6964761357957243#job/233/run/1" 
General 
Pipeline Run ID: Oc2e9db7-27eb-45e9-8ff8-966304408d7e 
Name 
ETC 
file-to-blob 
Availability flag 
Type 
DatabricksNoteb0( 
Copy 
Lookup 
Run Start 
10/24/2018 6:04 PM 
10/24/2018 6:03 PM 
10/24/2018 6:03 PM 
O 
Duration 
16 
15 
e ec we n egra Ion un me : 
(East US) 
"executionDuration": 41962 
Succeeded 
Succeeded 
Succeeded 
e au n egra Ion untime 
4924ffOO-67c9-4383-ac1 f62 
242b2aOf-1e09-4b78-9995-Oc732bc8cc31 
c6e37c26-8e4d-40fa-8702-7fd05dea5219 

* You can also verify the data file using storage explorer. (For correlating with Data Factory pipeline runs, we appended pipeline run id from data factory to the output folder. This way you can track back the files generated via each run)

Machine generated alternative text:
Upload Download 
New Folder 
Open 
copy URL 
sinkdata 
Select All 
Copy 
Paste 
Rename 
v Active blobs (default) 
Name 
•s 
processed_sink 
Access Tier 
Hot (inferred) 
Access Tier Last Modified 
C} Product Oc2e9db7-27eb-45e9-8ff8-966304408d7e 