

We will use the Autoloader feature to read data from source to bronze location.

NOTE: We need to make sure the schemaLocation and checkpointLocation path should be the same..

Our Recommendation for Easy ETL

To enable schema inference and evolution, the following conditions need to be met:

1. Do **not** provide a schema for your data
2. Provide a location to store your inferred schemas using the "cloudFiles.schemaLocation" option in your DataStreamReader. Here we show using the checkpoint path, which is recommended
3. Set the option "mergeSchema" to True in your DataStreamWriter.

```
# Reading data as a stream using Autoloader
## schemaLocation and checkpointLocation has to be match
df = spark.readStream.format("cloudFiles") \
    .option("cloudFiles.format", "parquet") \
    .option("cloudFiles.schemaLocation", "abfss://bronze@databricks.dfs.core.windows.net/checkpoint_orders") \
    .load("abfss://source@databricks.dfs.core.windows.net/orders")
```

▶ (1) Spark Jobs

▶ df: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 5 more fields]

Data Writing

```
# Writing the stream data
## schemaLocation and checkpointLocation has to be match
### trigger(once = True) is used to read the data once and write it to the sink and if we want make it continuous we can remove the trigger option
df.writeStream.format("parquet") \
    .outputMode("append") \
    .option("checkpointLocation", "abfss://bronze@databricks.dfs.core.windows.net/checkpoint_orders") \
    .option("path", "abfss://bronze@databricks.dfs.core.windows.net/orders") \
    .trigger(once = True) \
    .start()
```

▼ (1) Spark Jobs

▶ Job 25 [View](#) (Stages: 1/1)

▶ d0748524-3147-41ff-bc09-6cacf8e16e15 *Last updated: 7 minutes ago*

<pyspark.sql.streaming.query.StreamingQuery at 0x7f49dac8620>

[home](#) > [databricks](#) | [Containers](#)



bronze

Container

[Upload](#)

[Add Directory](#)

Overview

[Diagnose and solve problems](#)

[Access Control \(IAM\)](#)

[Settings](#)

Authentication method: Access ke

Location: bronze

Name

☐ **checkpoint_orders**

☐ orders

Location: **bronze** / checkpoint_orders

Search blobs by prefix (case-sensitive)

Name	
<input type="checkbox"/>	[-.]
<input type="checkbox"/>	__tmp_path_dir
<input type="checkbox"/>	_schemas
<input type="checkbox"/>	commits
<input type="checkbox"/>	offsets
<input type="checkbox"/>	sources
<input type="checkbox"/>	metadata

Authentication method: Access key (Switch to Microsoft Entra user account)

Location: **bronze** / checkpoint_orders / sources / 0 / rocksdb

Search blobs by prefix (case-sensitive)

Name	
<input type="checkbox"/>	[-.]
<input type="checkbox"/>	__tmp_path_dir
<input type="checkbox"/>	logs
<input type="checkbox"/>	SSTs
<input type="checkbox"/>	0.zip
<input type="checkbox"/>	1.zip

Make our Notebook dynamic using parameters(widgets), so that we can use the same script to process data from different folders. So, in order to do that we've created a notebook which contains the values for passing parameters.

bronze_parameters Python Tabs: OFF ☆

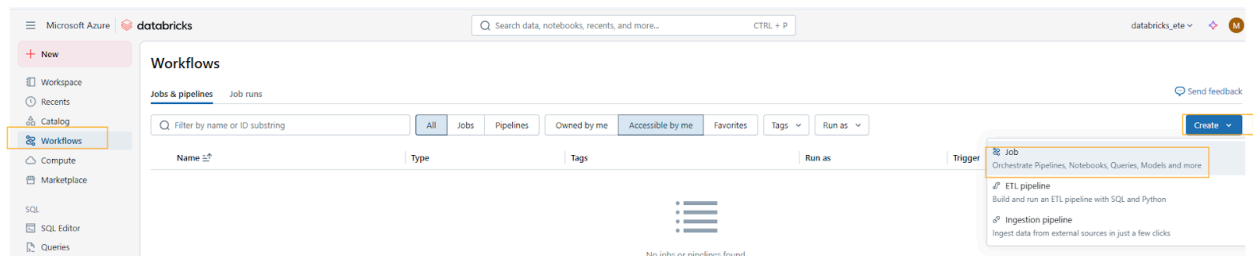
File Edit View Run Help Last edit was 8 hours ago

```
datasets = [
    {'file_name': 'orders'},
    {'file_name': 'customers'},
    {'file_name': 'products'}
]
```

```
dbutils.jobs.taskValues.set('output_datasets', datasets)
```

Creating the job to run bronze layer:

Workflows→Create→Job



First, we need to create a task for parameters and then for ingestion script

Task for Parameters:

Scheduling the notebook from Git repos:

Task name*	<input type="text" value="Parameters_Passing"/>		
Type*	<input type="text" value="Notebook"/>		
Source*	<input type="text" value="Git provider (main)"/>		<input type="button" value="Edit"/>
Path*	<input type="text" value="scripts/bronze_scripts/bronze_parameters"/>		
Compute*	<input type="text" value="Medishetty's Cluster 16 GB · 4 Cores · DBR 16.3 · Spark 3.5.2 · Scala 2.12"/>		

Jobs running on all-purpose clusters are considered all-purpose compute. [Learn more](#)

When you select the Git provider as Source, we will get the pop-up like below...

Git information

Task for Ingestion Script:

Task name* ⓘ

Type*

Source* ⓘ [Edit](#) ▼

Path* ⓘ

Compute* ⓘ 16 GB · 4 Cores · DBR 16.3 · Spark 3.5.2 · Scala 2.12 [Learn more](#)

ⓘ Jobs running on all-purpose clusters are considered all-purpose compute. [Learn more](#)

Depends on

Parameters ⓘ

Key	Value
<input type="text" value="file_name"/>	<input type="text" value="{}"/>

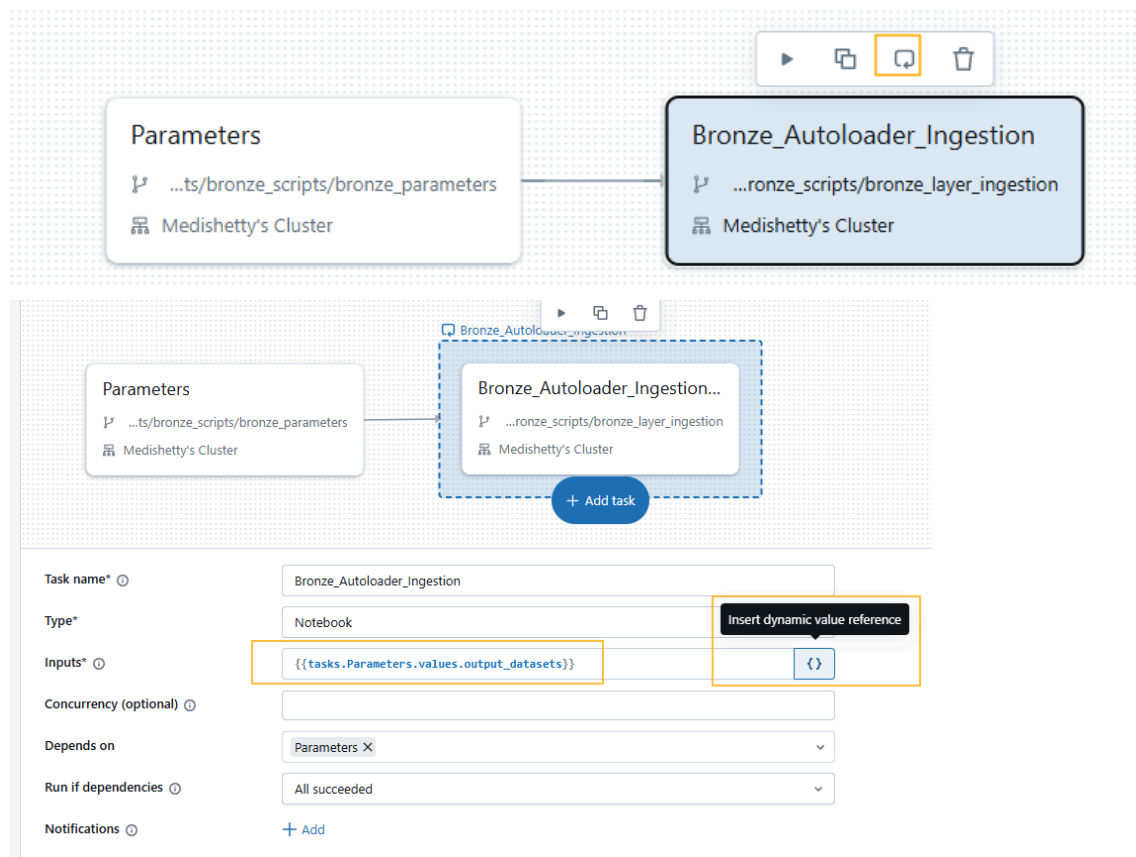
+ Add

Notifications ⓘ + Add

Retries ⓘ + Add

[Cancel](#) [Create task](#)

Tasks have been created for both parameters and ingestion script. So, we need to loop the ingestion script task for every parameter in parameters task..

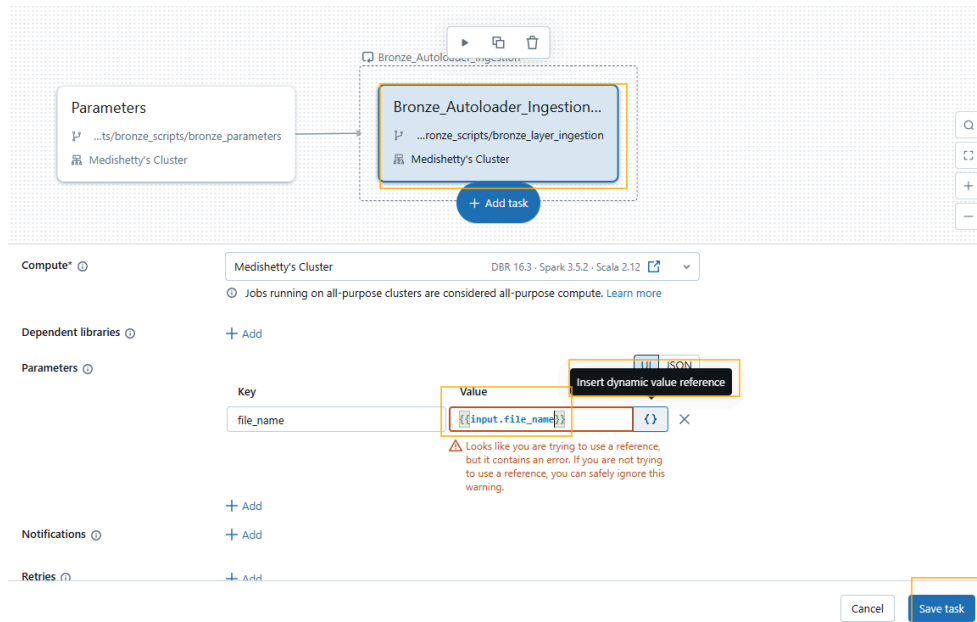


Step 1: Click on loop (like forEach Activity)

Step 2: Click on *Insert dynamic value reference*

Step 3: Give the text in **inputs**

Step 4: Save Task



Step 5: Select the task inside the loop

Step 6: click on *insert dynamic value reference*

Step 7: Give the text in **Value**

Step 8: Save Task

Running the Job:

