

# Lab 15: Auto Loader – Incrementally Load CSV Files from ADLS

Author: Dr. Sandeep Kumar Sharma

---

## Lab Overview

Welcome to your first Auto Loader hands-on lab! In this exercise, you will learn how Databricks Auto Loader processes **new CSV files incrementally** from Azure Data Lake Storage (ADLS) using streaming reads.

This lab is written in a simple, beginner-friendly style — exactly like a classroom session.

---

## Learning Objective

By the end of this lab, you will be able to:

- Configure Auto Loader to read CSV files incrementally from ADLS
- Understand the purpose of `cloudFiles` format
- Use checkpoint folders to track ingestion
- Continuously process incoming files without reprocessing old ones
- Write results into a Delta table (Bronze layer)

---

## Prerequisites

Before you begin, ensure:

1. You have access to ADLS with **Managed Identity**, **Service Principal**, or SAS Key (any method works)
2. You have a working ADLS path like:

```
abfss://raw@yourstorageaccount.dfs.core.windows.net/incoming-sales/
```

3. You can read files using normal Spark read commands

---

## Dataset Setup

Imagine new CSV files are continuously arriving in ADLS under:

```
abfss://raw@yourstorageaccount.dfs.core.windows.net/incoming-sales/
```

Example files:

```
sales_20240101.csv  
sales_20240101_01.csv  
sales_20240101_02.csv
```

Your task is to use Auto Loader to load each new file **only once**.

---

---

## STEP 1 — Define Storage Path and Checkpoint Location

---

Create variables for the input folder and checkpoint folder.

```
# ADLS path where new CSV files arrive  
input_path = "abfss://raw@yourstorageaccount.dfs.core.windows.net/incoming-  
sales/"  
  
# Checkpoint folder for Auto Loader\checkpoint_path = "dbfs:/mnt/checkpoints/  
autoloader_sales/"
```

Checkpointing is **critical** because Auto Loader stores its internal index here.

---

---

## STEP 2 — Create the Auto Loader Stream (Read Stream)

---

Use `cloudFiles` to read incrementally.

```
autoloader_df = (spark.readStream
    .format("cloudFiles")
    .option("cloudFiles.format", "csv")
    .option("header", "true")
    .load(input_path))
```

Explanation: - `format("cloudFiles")` → tells Spark to use Auto Loader - `cloudFiles.format="csv"`  
 → allows CSV incremental reading - `load(input_path)` → watches the ADLS folder for *new* files

---

## STEP 3 — Write the Stream to a Bronze Delta Table

---

We now write the incremental data to a Delta table.

```
output_bronze = "dbfs:/mnt/bronze/sales_bronze_delta/"

query = (autoloader_df.writeStream
    .format("delta")
    .option("checkpointLocation", checkpoint_path)
    .outputMode("append")
    .start(output_bronze))
```

Explanation: - `outputMode("append")` → new data is continuously appended - `checkpointLocation`  
 → keeps track of processed files - `.start()` → starts a streaming job in Databricks

---

## STEP 4 — Monitor the Stream

---

Use the Databricks UI:

1. Notebook top-right → **Streaming** tab

2. View progress, batches, processed files

OR in code:

```
query.status
```

You should see:

```
"message": "Processing new files"
```

---

## STEP 5 — Test Incremental Behavior

---

Upload a new CSV file into ADLS:

```
sales_20240101_03.csv
```

Auto Loader will automatically pick it up.

Run:

```
display(spark.read.format("delta").load(output_bronze))
```

You will see the new file's data merged.

Upload another file:

```
sales_20240101_04.csv
```

Only this new file will be ingested.

Auto Loader **never** reprocesses old files.

---

---

## STEP 6 — Stop the Stream (When Done)

---

```
query.stop()
```

This gracefully stops the streaming job.

---

---

## Understanding What Just Happened (Simple Summary)

---

1. Auto Loader watched your ADLS folder.
2. It detected only new files.
3. It converted CSV into a DataFrame stream.
4. It stored progress in your checkpoint folder.
5. It wrote output incrementally into a Delta table.

This is 100% production-grade ingestion.

---

## End of Lab 15

You have successfully created an **incremental ingestion pipeline** using Auto Loader.

If you want, next we can create: - **Lab 16 — Auto Loader with Schema Evolution** (automatic column detection) - **Lab 17 — Auto Loader with Complex JSON** - **Lab 18 — Auto Loader Bronze → Silver → Gold Pipeline**

Tell me which lab you want next!