## Databricks Best Practise & Learning’s

**# Databricks Streaming Pipeline (Step-by-Step)**

# Sample file: people.csv (fields: id, name, department, event\_time)

from pyspark.sql.functions import col, upper, current\_timestamp, input\_file\_name, window

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, TimestampType

# 1. Define Input & Output Paths

source\_path = "/mnt/data/autoloader/incoming/"

schema\_path = "/mnt/data/autoloader/schema/people/"

checkpoint\_path = "/mnt/data/autoloader/checkpoints/people/"

target\_path = "/mnt/data/bronze/people/"

# 2. Define Schema Explicitly (for clarity)

schema = StructType([

StructField("id", IntegerType(), True),

StructField("name", StringType(), True),

StructField("department", StringType(), True),

StructField("event\_time", TimestampType(), True)

])

# 3. Read Files Using Auto Loader (cloudFiles)

df = (

spark.readStream

.format("cloudFiles")

.option("cloudFiles.format", "csv")

.option("cloudFiles.inferColumnTypes", "true")

.option("cloudFiles.schemaLocation", schema\_path)

.option("header", "true")

.load(source\_path)

)

# 4. Apply Watermark (late data tolerance)

df\_watermarked = df.withWatermark("event\_time", "10 minutes")

# 5. (Optional) Time-Based Windowing (for aggregations)

df\_windowed = df\_watermarked.groupBy(

window(col("event\_time"), "30 minutes"),

col("department")

).count()

# 6. Deduplicate Based on id + source file path

df\_deduped = df.dropDuplicates(["id", "name", "event\_time"])

# 7. Transformations (enrichment, metadata stamping)

df\_transformed = df\_deduped \

.withColumn("department", upper(col("department"))) \

.withColumn("ingestion\_timestamp", current\_timestamp()) \

.withColumn("source\_file", input\_file\_name())

# 8. Write to Delta Table with Checkpointing

(

df\_transformed.writeStream

.format("delta")

.outputMode("append")

.option("checkpointLocation", checkpoint\_path)

.start(target\_path)

)

# Notes:

# - This pipeline processes incoming CSV files with schema inference

# - Watermark ensures late data (up to 10 minutes) is processed

# - Checkpoints track state for recovery

# - Deduplication avoids duplicates in the Bronze table

Appendix :

## Most Common .option("cloudFiles.\*") Settings

|  |  |
| --- | --- |
| **Option** | **Description** |
| **cloudFiles.format** | Required. Specifies the file format: csv, json, parquet, avro, orc, text, binaryFile |
| **cloudFiles.schemaLocation** | Required. Stores inferred or user-defined schema persistently for efficient re-use |
| **cloudFiles.inferColumnTypes** | Auto-detects data types from files (for CSV and JSON). Defaults to false |
| **cloudFiles.includeExistingFiles** | Ingests already-existing files when the stream starts. Set to true to include historical files |
| **cloudFiles.useNotifications** | Enables event-based discovery (via S3, ADLS Gen2, or GCS) instead of directory listing |
| **cloudFiles.maxBytesPerTrigger** | Limits total size (in bytes) processed per micro-batch |
| **cloudFiles.maxFilesPerTrigger** | Limits number of new files per micro-batch (default: unlimited) |
| **cloudFiles.backfillInterval** | Controls rate of discovery for old files when includeExistingFiles=true |
| **cloudFiles.partitionColumns** | Used to specify partition columns (especially for Hive-style partitions) |
| **cloudFiles.resourceGroup** | Azure-specific: used for Event Grid setup |
| **cloudFiles.subscriptionId** | Azure-specific: used for Event Grid setup |
| **cloudFiles.connectionString** | Azure-specific: connection to Event Grid topic for file notifications |
| **cloudFiles.validateOptions** | Validates all options at runtime. Default is true |

## 📌 Format-Specific Options

Some options also depend on the file type you're ingesting:

### For CSV:

* header → "true" if the file has headers
* delimiter → "," or "|", etc.
* quote, escape, nullValue, etc.

### For JSON:

* multiline → "true" for JSON lines with nested objects
* allowBackslashEscapingAnyCharacter

### For Parquet/Avro/ORC:

* Typically schema is embedded; fewer format-specific options needed