End to End Monitoring of Data Pipeline

amadeus

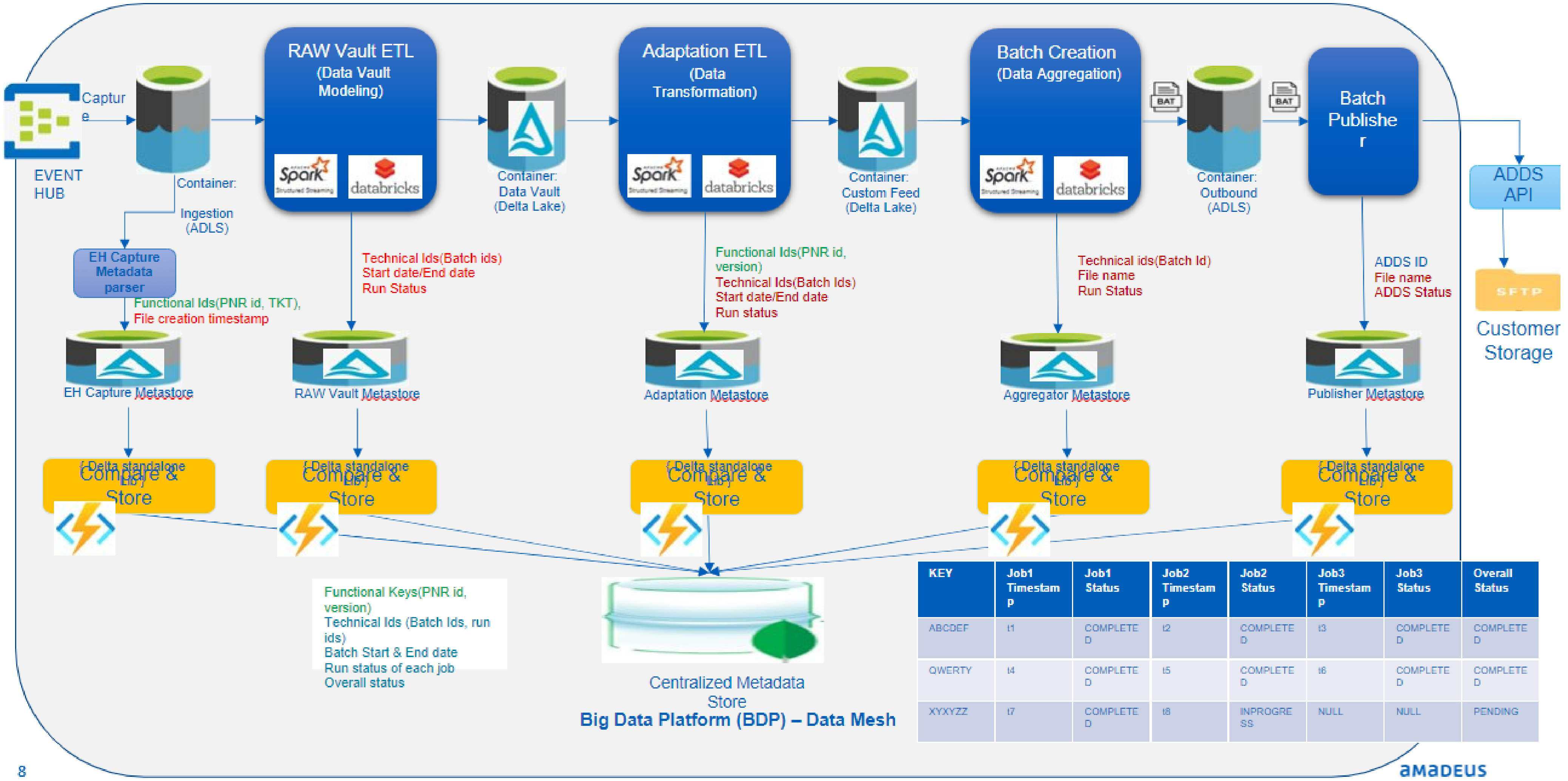
Agenda

1.

Introduction

Context and Goal of Internship

- Create an end-to-end data pipeline monitoring framework, which can help to monitor batch execution status based on functional keys across different jobs in a big data pipeline.


RAW Vault ETL (Data Vault Modeling)
Adaptation ETL (Data Transformation)
Batch Creation (Data Aggregation)
Batch Publisher
Captur e
EVENT HUB
Container: Ingestion (ADLS)
Container: Data Vault (Delta Lake)
Container: Custom Feed (Delta Lake)
Container: Outbound (ADLS)
ADDS API
SFTP
Customer Storage
EH Capture Metadata parser
Functional Ids(PNR id, TKT), File creation timestamp
Technical Ids(Batch ids) Start date/End date Run Status
Functional Ids(PNR id, version) Technical Ids(Batch Ids) Start date/End date Run status
Technical ids(Batch Id) File name Run Status
ADDS ID File name ADDS Status
EH Capture Metastore
RAW Vault Metastore
Adaptation Metastore
Aggregator Metastore
Publisher Metastore
Compare & Store (Delta standalone Lib)
Functional Keys(PNR id, version) Technical Ids (Batch Ids, run ids) Batch Start & End date Run status of each job Overall status
Centralized Metadata Store
Big Data Platform (BDP) – Data Mesh
8

KEY	Job1 Timestamp	Job1 Status	Job2 Timestamp	Job2 Status	Job3 Timestamp	Job3 Status	Overall Status
ABCDEF	t1	COMPLETED	t2	COMPLETED	t3	COMPLETED	COMPLETED
QWERTY	t4	COMPLETED	t5	COMPLETED	t6	COMPLETED	COMPLETED
XYXYZZ	t7	COMPLETED	t8	INPROGRESS	NULL	NULL	PENDING

Introduction

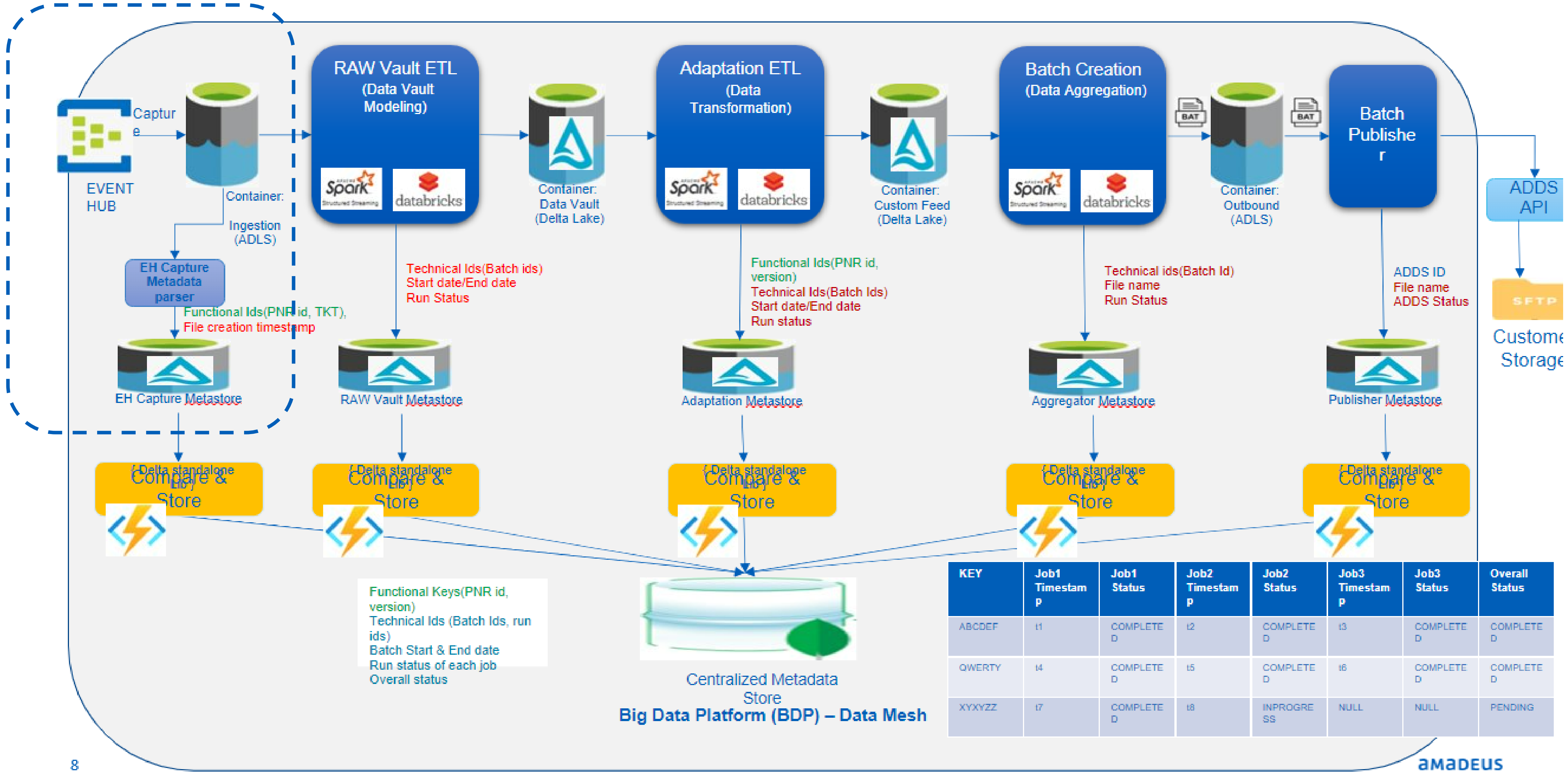
This whole process of end-to-end monitoring is diving into four main modules

- EH Capture Metadata storage
- Metadata capture for ETL job (Generic Library)
- Batch Publisher metadata storage
- Compare and Store

3.

Demo

EH Capture Metadata storage



Functional Keys(PNR id, version)
Technical Ids (Batch Ids, run ids)
Batch Start & End date
Run status of each job
Overall status

KEY	Job1 Timestamp	Job1 Status	Job2 Timestamp	Job2 Status	Job3 Timestamp	Job3 Status	Overall Status
ABCDEF	t1	COMPLETED	t2	COMPLETED	t3	COMPLETED	COMPLETED
QWERTY	t4	COMPLETED	t5	COMPLETED	t6	COMPLETED	COMPLETED
XYXYZZ	t7	COMPLETED	t8	INPROGRESS	NULL	NULL	PENDING

EH Capture Metadata storage

Introduction

EVENT HUB

- Event Hub is an Azure service that enables in processing large amounts of event data from connected devices and applications.

AUTO LOADER

- Databricks Autoloader is an **Optimized File Source** that can automatically perform incremental data loads from your Cloud storage as it arrives into the **Delta Lake Tables**.
- Databricks Autoloader presents a new Structured Streaming Source called **cloudFiles**.

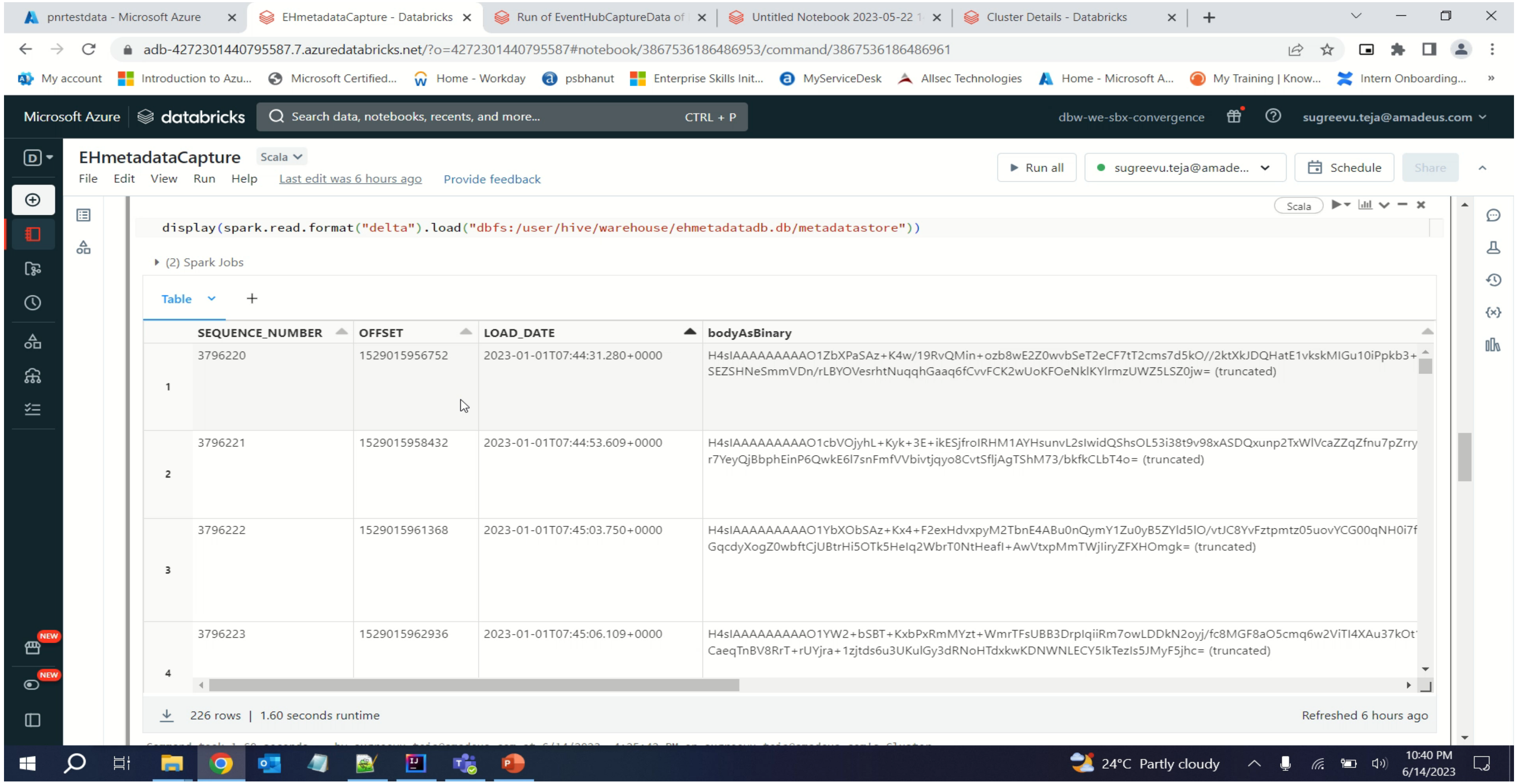
Uses of AUTO LOADER

- No file state management
- Scalable & Easy to use
- Schema inference & evolution support

EH Capture Metadata storage Demo Flow

- Ingestion of files (AVRO) incrementally and efficiently from azure blob storage container
- Consumption of the Avro files in the FrameWork(Spark Structured Streaming) and Performing deserialization and Uncompression into the Json Format.
- Traversing the json document to derive columns for the meta datatable.
- And pushing the Avro files after Consolidating them into the single record to those Related FeedType Database.
- Creating the different Database-tables for related Functional feedTypes in the container in the form of delta table.
- This Single application jar can work with the other feedtype Modules by Changing the config file while running the job.

EH Capture Metadata storage


EHmetadataCapture - Databricks
adb-4272301440795587.7.azuredatabricks.net/?o=4272301440795587#notebook/3867536186486953/command/3867536186486961
EHmetadataCapture
Scala
File Edit View Run Help
Last edit was 6 hours ago
Provide feedback
Run all
Schedule
Share
Table

```
display(spark.read.format("delta").load("dbfs:/user/hive/warehouse/ehmetadatadb.db/metadatastore"))
```

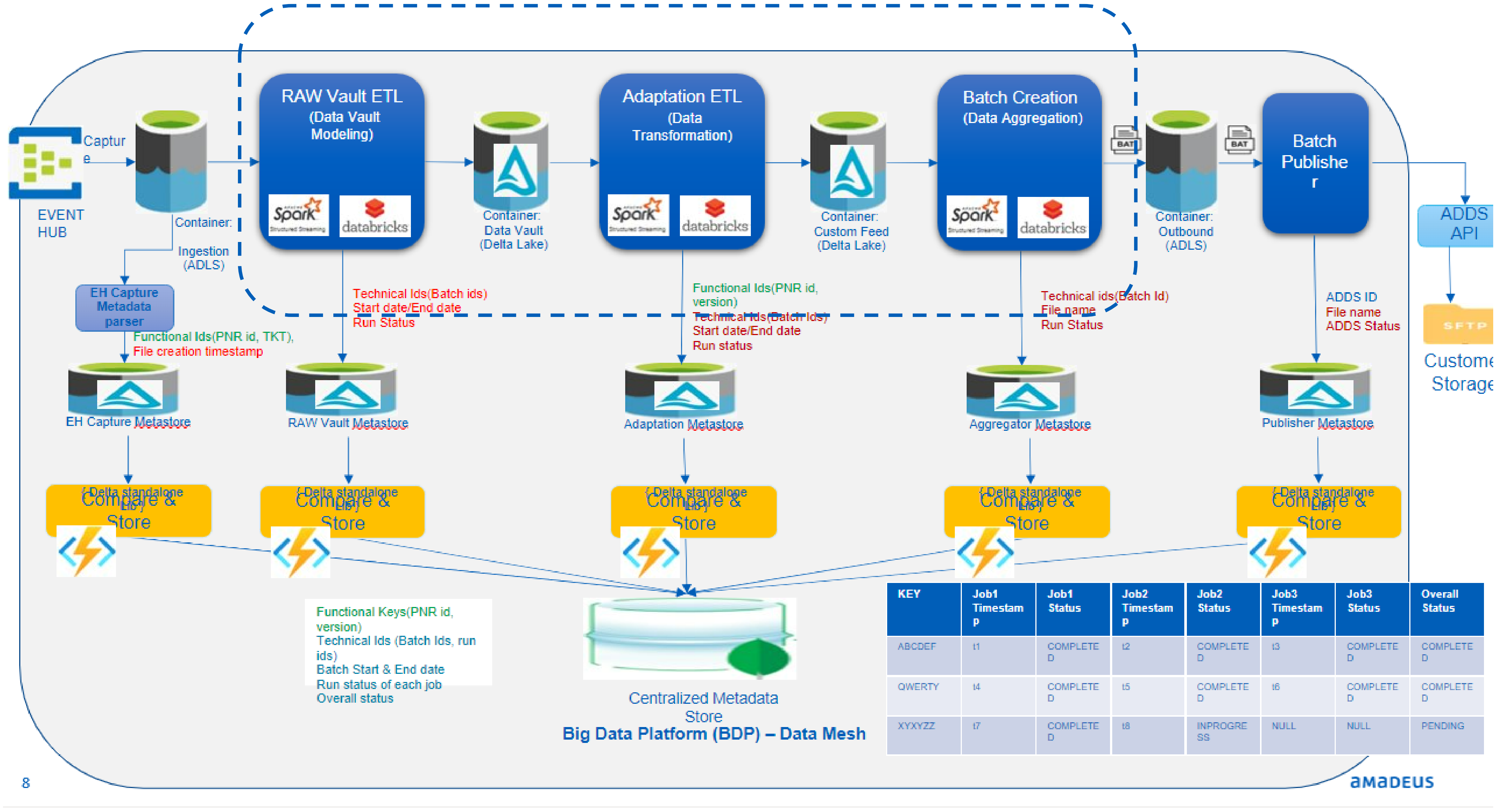
(2) Spark Jobs

	SEQUENCE_NUMBER	OFFSET	LOAD_DATE	bodyAsBinary
1	3796220	1529015956752	2023-01-01T07:44:31.280+0000	H4sIAAAAAAAAAO1ZbXPaSAz+K4w/19RvQMin+ozb8wE2Z0wvbSeT2eCF7tT2cms7d5kO//2ktXkJDQHatE1vkskMIGu10iPpkb3+ SEZSHNeSmmVDn/rLBYOVesrhtNuqqhGaaq6fCvvFCK2wUoKFOeNklKYlrmzUWZ5LSZ0jw= (truncated)
2	3796221	1529015958432	2023-01-01T07:44:53.609+0000	H4sIAAAAAAAAAO1cbVOjyhL+Kyk+3E+ikESjfroIRHM1AYHsunvL2sIwidQShsOL53i38t9v98xASDQxunp2TxWlVcaZZqZfnu7pZrry r7YeyQjBbphEinP6QwkE6l7snFmfVVbivtjqyo8CvtSfljAgTShM73/bkfkCLbT4o= (truncated)
3	3796222	1529015961368	2023-01-01T07:45:03.750+0000	H4sIAAAAAAAAAO1YbXObSAz+Kx4+F2exHdvxpyM2TbnE4ABu0nQymY1Zu0yB5ZYld5lO/vtJC8YvFztpmtz05uovYCG00qNH0i7f GqcdyXogZ0wbftCjUBtrHi5OTk5HeIq2WbrT0NtHeafl+AwVtxpMmTWjIiryZFXHOmgk= (truncated)
4	3796223	1529015962936	2023-01-01T07:45:06.109+0000	H4sIAAAAAAAAAO1YW2+bSBT+KxbPxRmMYzt+WmrTFsUBB3DrpIqiiRm7owLDDkN2oyj/fc8MGF8aO5cmq6w2ViTI4XAu37kOt CaeqTnBV8RrT+rUYjra+1zjtds6u3UKulGy3dRNoHTdxkwKDNWNLECY5IkTezIs5JMyF5jhc= (truncated)

226 rows | 1.60 seconds runtime

Refreshed 6 hours ago

Generic library creation



KEY	Job1 Timestamp	Job1 Status	Job2 Timestamp	Job2 Status	Job3 Timestamp	Job3 Status	Overall Status
ABCDEF	t1	COMPLETED	t2	COMPLETED	t3	COMPLETED	COMPLETED
QWERTY	t4	COMPLETED	t5	COMPLETED	t6	COMPLETED	COMPLETED
XYXYZZ	t7	COMPLETED	t8	INPROGRESS	NULL	NULL	PENDING

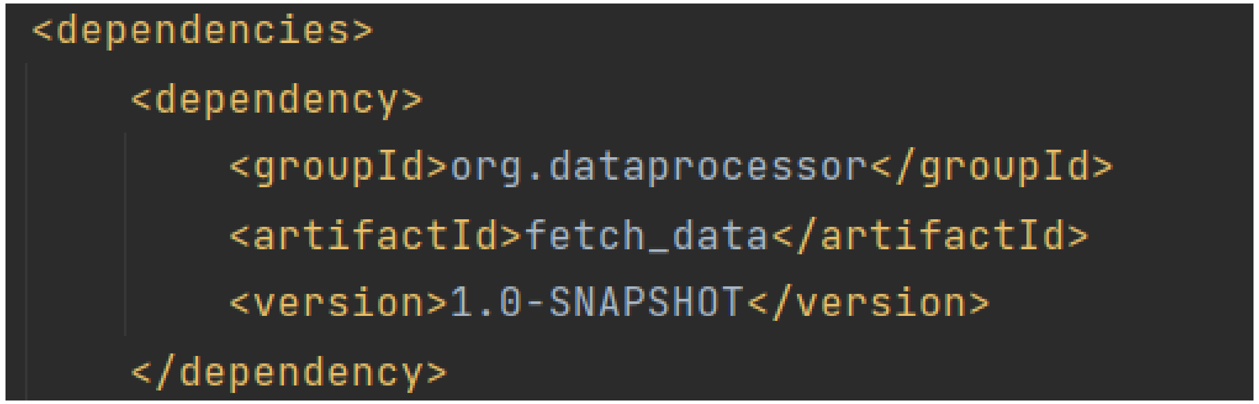
Generic library creation

Introduction

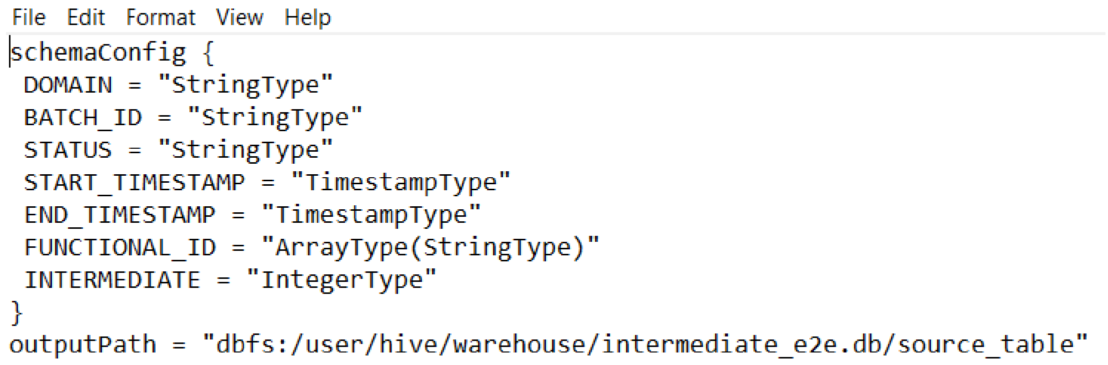
_ Fetches data from modules

_ Takes schema for the fields from the configuration file

_ Writes the output inside a table after creating a data frame

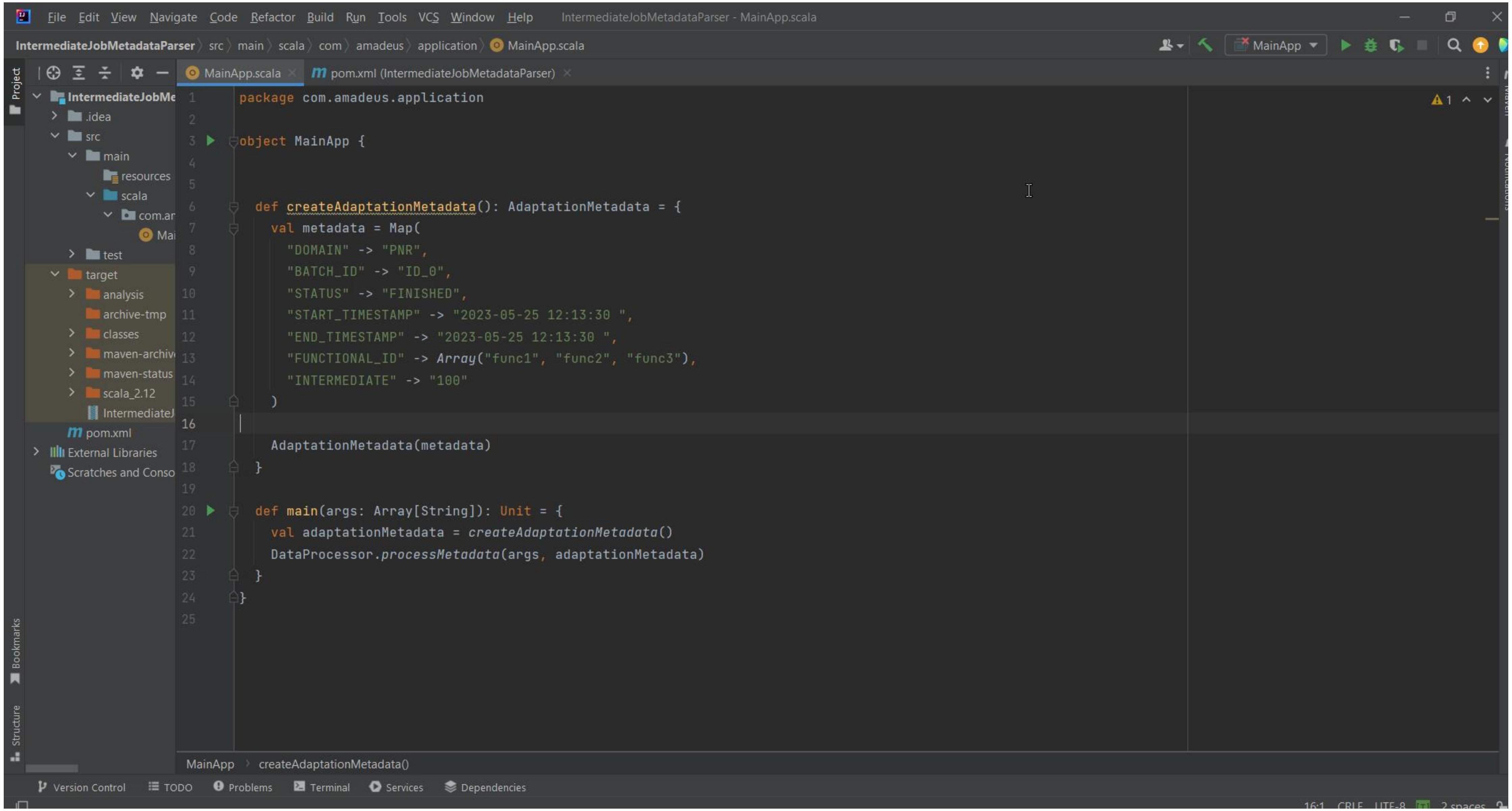


```
<dependencies>
    <dependency>
        <groupId>org.dataprocessor</groupId>
        <artifactId>fetch_data</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
```



```
schemaConfig {
 DOMAIN = "StringType"
 BATCH_ID = "StringType"
 STATUS = "StringType"
 START_TIMESTAMP = "TimestampType"
 END_TIMESTAMP = "TimestampType"
 FUNCTIONAL_ID = "ArrayType(StringType)"
 INTERMEDIATE = "IntegerType"
}
outputPath = "dbfs:/user/hive/warehouse/intermediate_e2e.db/source_table"
```

Generic library creation


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
IntermediateJobMetadataParser - MainApp.scala
IntermediateJobMetadataParser › src › main › scala › com › amadeus › application › MainApp.scala
MainApp.scala
pom.xml (IntermediateJobMetadataParser)
IntermediateJobMe
.idea
src
main
resources
scala
com.ar
Mai
test
target
analysis
archive-tmp
classes
maven-archiv
maven-status
scala_2.12
IntermediateJ
pom.xml
External Libraries
Scratches and Conso
MainApp › createAdaptationMetadata()
Version Control
TODO
Problems
Terminal
Services
Dependencies

```scala
package com.amadeus.application

object MainApp {


  def createAdaptationMetadata(): AdaptationMetadata = {
    val metadata = Map(
      "DOMAIN" -> "PNR",
      "BATCH_ID" -> "ID_0",
      "STATUS" -> "FINISHED",
      "START_TIMESTAMP" -> "2023-05-25 12:13:30 ",
      "END_TIMESTAMP" -> "2023-05-25 12:13:30 ",
      "FUNCTIONAL_ID" -> Array("func1", "func2", "func3"),
      "INTERMEDIATE" -> "100"
    )

    AdaptationMetadata(metadata)
  }

  def main(args: Array[String]): Unit = {
    val adaptationMetadata = createAdaptationMetadata()
    DataProcessor.processMetadata(args, adaptationMetadata)
  }
}
```

aDDS Batch Publisher Monitoring

Introduction

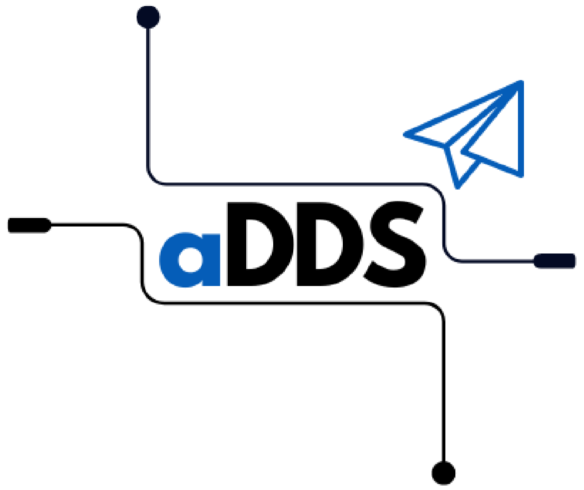
_ aDDS for transferring files

_ aDDS provides different endpoints:

- aDDS API Stream
- aDDS API Notify

_ aDDS is separate team

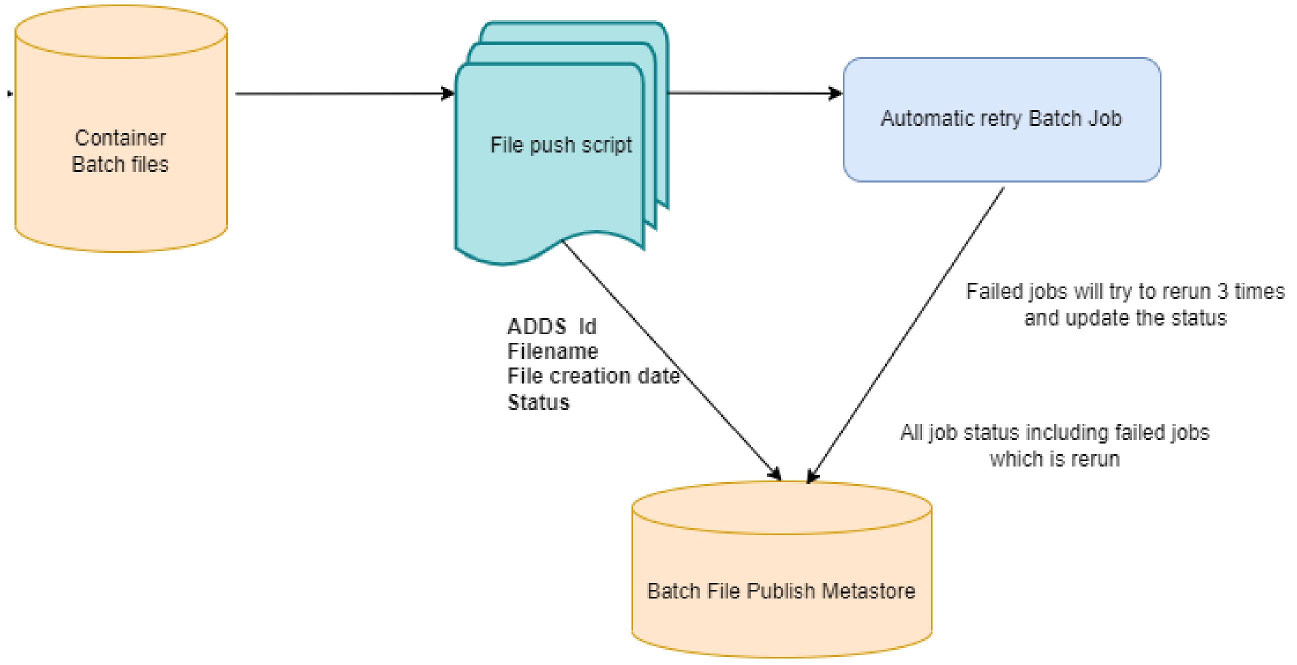
_ Files status need to be monitored



Batch Publisher Library

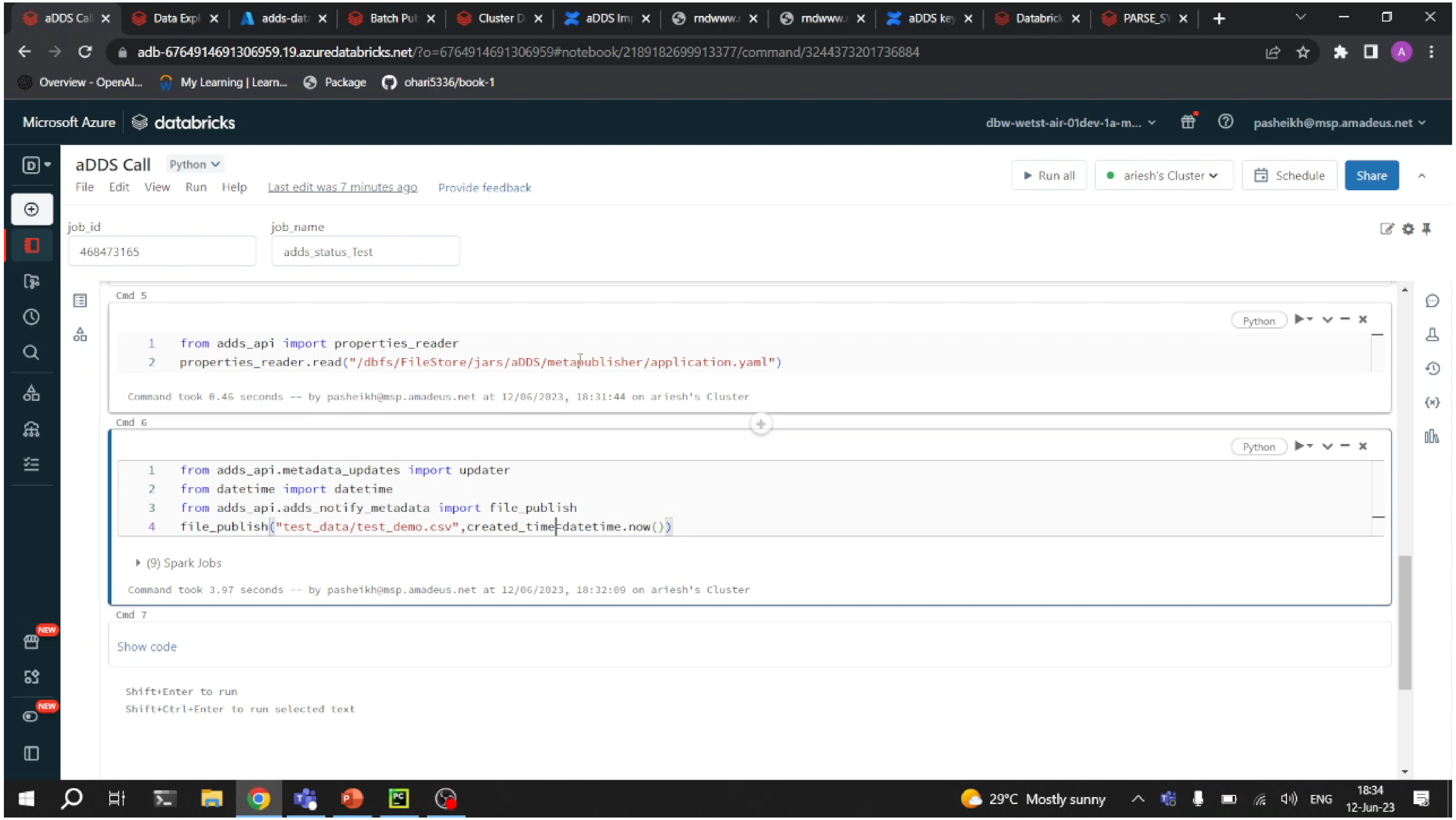
_ Already a batch publisher lib to transfer files from storage account to external sftp.

_ Now it will also monitor "Batch Publisher Metastore" metadata of files.


Container
Batch files
File push script
Automatic retry Batch Job
ADDS Id
Filename
File creation date
Status
Failed jobs will try to rerun 3 times and update the status
All job status including failed jobs which is rerun
Batch File Publish Metastore

Batch File Monitoring

Batch Publisher Library



job_id: 468473165

job_name: adds_status_Test

Cmd 5

```
from adds_api import properties_reader
properties_reader.read("/dbfs/FileStore/jars/aDDS/metapublisher/application.yaml")
```

```
Command took 0.46 seconds -- by pasheikh@msp.amadeus.net at 12/06/2023, 18:31:44 on ariesh's Cluster
```

Cmd 6

```
from adds_api.metadata_updates import updater
from datetime import datetime
from adds_api.adds_notify_metadata import file_publish
file_publish("test_data/test_demo.csv",created_time=datetime.now())
```

(9) Spark Jobs

```
Command took 3.97 seconds -- by pasheikh@msp.amadeus.net at 12/06/2023, 18:32:09 on ariesh's Cluster
```

Cmd 7

Show code

```
Shift+Enter to run
Shift+Ctrl+Enter to run selected text
```

amadeus

Compare & Store

Delta Standalone

- The Delta Standalone library is a Java library that can be used to read from and write to Delta tables
- Standalone doesn’t depend on Apache Spark
- ACID guarantees

MongoDB

- MongoDB is document-oriented database
- Collection: Collection is a group of MongoDB documents, which is similar to table
- Document: Document is a set of key-value pairs, which is similar to row

Azure Functions

- Azure functions allows you to schedule the execution of your functions
- Integrates with most of the development tools
- Automatically scales the execution environment

Demo Flow

_ Reading Delta table data & schema by provided configuration

_ Creating MongoDB connection to access the maximum timestamp value

_ Comparing MongoDB timestamp with timestamps from delta table

_ Perform Transformations on the table record which has early timestamp

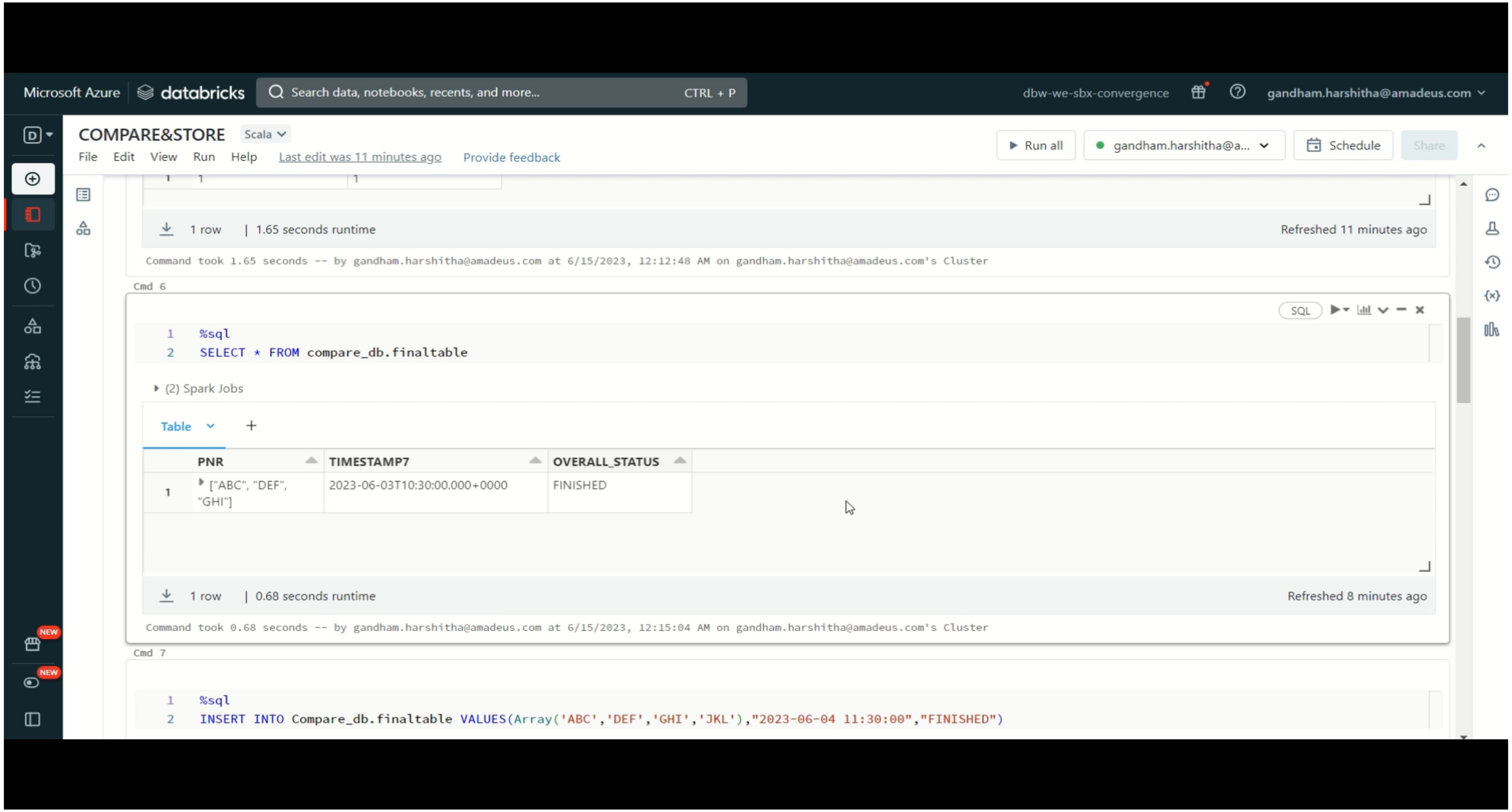
_ Check if the exploded ID is present in MongoDB

_ If ID exists, update the document with values

_ If ID doesn’t exist, insert the values into new document

_ If the timestamp is later, end the workflow

Compare & Store



1 row | 1.65 seconds runtime   Refreshed 11 minutes ago

```
Command took 1.65 seconds -- by gandham.harshitha@amadeus.com at 6/15/2023, 12:12:48 AM on gandham.harshitha@amadeus.com's Cluster
```

Cmd 6

```
%sql
SELECT * FROM compare_db.finaltable
```

(2) Spark Jobs

Table

	PNR	TIMESTAMP7	OVERALL_STATUS
1	["ABC", "DEF", "GHI"]	2023-06-03T10:30:00.000+0000	FINISHED

1 row | 0.68 seconds runtime   Refreshed 8 minutes ago

```
Command took 0.68 seconds -- by gandham.harshitha@amadeus.com at 6/15/2023, 12:15:04 AM on gandham.harshitha@amadeus.com's Cluster
```

Cmd 7

```
%sql
INSERT INTO Compare_db.finaltable VALUES(Array('ABC','DEF','GHI','JKL'),"2023-06-04 11:30:00","FINISHED")
```

4.

Key Takeaways

Key Takeaways

- ➢Developed a deep understanding of Scala and Spark programming.
- ➢Gained experience using Databricks and IntelliJ to run spark jobs.
- ➢Developed proficiency in querying and manipulating data using SQL.
- ➢Understood how data works with respect to high level
- ➢Understood about real-time streaming using Autoloader

Thank you!

amadeus