Building a BI Solution in the Cloud for a Startup

Marius Costin - Data & Analytics Lead, eMAG / Freshful

Marius Costin

Data & Analytics Lead, eMAG / Freshful

In eMAG since 2016:
- DataWarehouse & Big Data Developer – 9 months
- DataWarehouse Architect – 4.5 years
- Data & Analytics Team Lead – 10 months

Bogdan Miclaus

Cloud Data Engineer, eMAG / Freshful

In eMAG since 2017:
- Datawarehouse Developer – 2 years
- Senior DataWarehouse Developer – 2 years
- Cloud Engineer – 9 months

eMAG is a fast-growing online marketplace and ecommerce leader in south-eastern Europe headquartered in Romania. It is part of the Naspers Group.

Freshful is the newest addition to the eMAG group, the first exclusive online grocery retailer in Romania, delivering fresh products from our own warehouse in under 2 hours.

Agenda

BI & Analytics in the Cloud

1. Data Sources & Targets
2. Extract, Load, Transform
3. Scheduling & Orchestration
4. Visualization
5. Elastic Computing
6. Short Demo
7. Stats
8. Costs

Modern vs. Classic Approach

Which way should we head?

The Classical Approach

What we know best

It would assume creating a MSSQL DataWarehouse, use SSIS as the ETL tool and use Qlikview as a Reporting Tool.

This had some drawbacks:

- Very high costs
- Low scalability
- Can’t harvest cluster processing power
- Data is growing fast and the old tools can’t keep up

It would assume using a modern data stack in the cloud. The drawback was our lack of experience in this area. The benefits would be:

- Getting to learn & use new and exciting technologies
- Using open source tools and open standards
- Low cost at the beginning of the startup
- Highly scalable
- Eliminate the maintenance of infrastructure

The Modern Approach

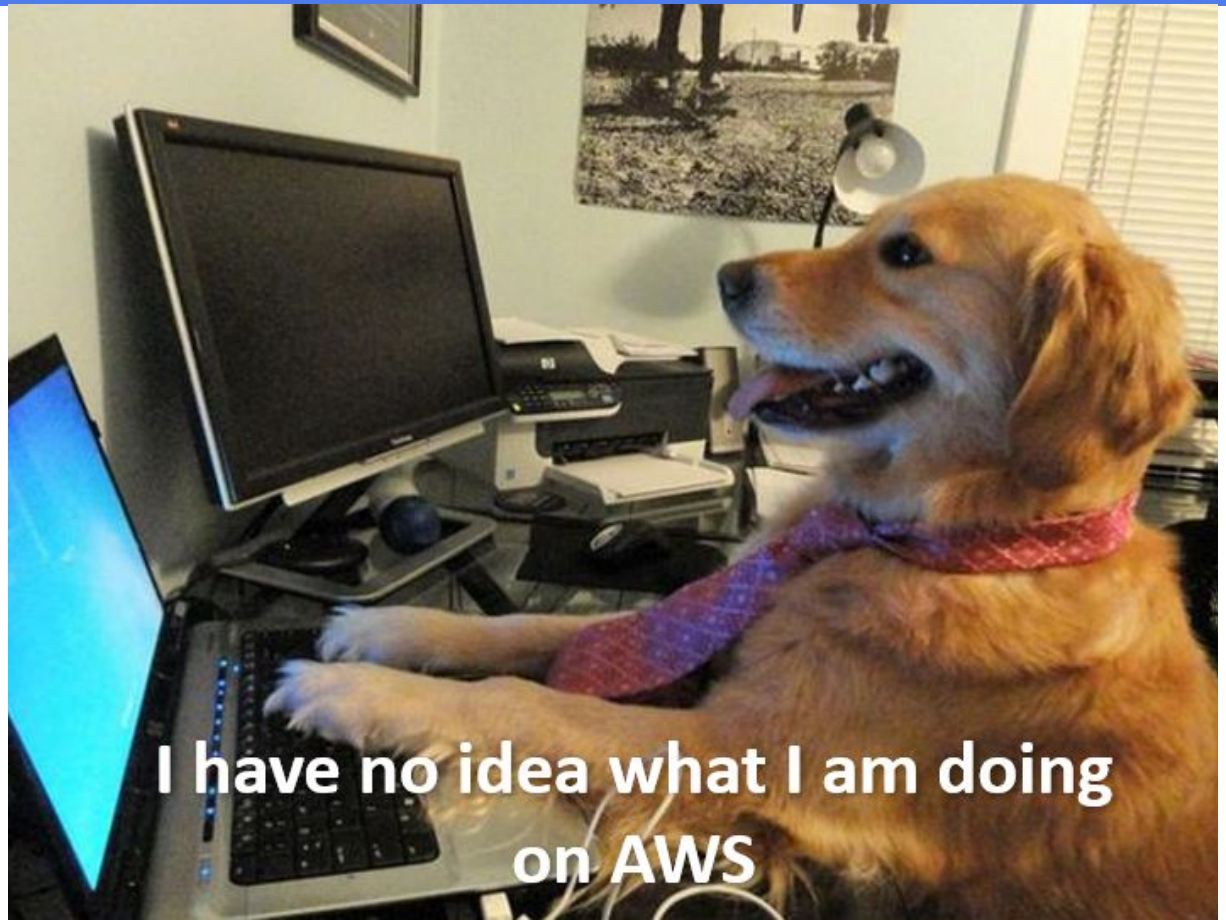
We were not experts, but we had time to learn.

A bit of time.

As we had a fresh start and the chance to build something from scratch, we decided to go with the Modern Approach…

And it was fun doing it!

Getting Started



Sources & Targets

Determining where the data lies and where the data will be stored

Data Sources & Targets

Traffic Data

Generated by the app & site:

- Source: Big Query
- Target: Parquet files in S3 buckets

Master & Financial Data

Generated by the ERP system:

- Source: SAP
- Target: Parquet files in S3 buckets
- Exported into Rabbit MQ queues

Orders Data

Generated by the app & site:

- Source: Amazon RDS (MySQL)
- Target: Parquet files in S3 buckets
- Also, queried directly for real time purposes

Extract, Load, Transform

Shifting from the ETL paradigm to ELT & PLT

Extract & Ingest

Bringing the raw data from the Sources to the Staging S3 Bucket

What we chose: **Python scripts with Pandas**

- Execute Python custom scripts to import data into our Staging Layer

What we also tried:

- Airbyte
- Amazon Data Pipelines
- Amazon Glue

Transform (small workloads)

Bringing the raw data from the Staging S3 Bucket to the Reporting S3 Bucket

What we chose: **Python scripts with Pandas**

- Scripts reading files from the staging S3 bucket, deduplicate the data & writes the data into S3 partitioned files
- For example: 1 script for Rabbit queues

What we also tried:

- Amazon Glue & Redshift
- Vertica

Transform (big workloads)

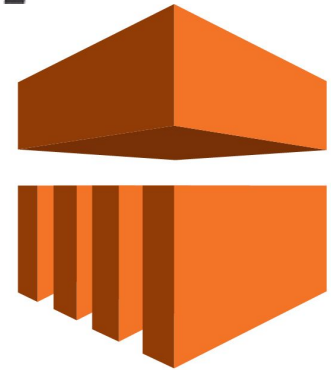
Bringing the raw data from the Staging S3 Bucket to the Reporting S3 Bucket

What we chose: **Amazon EMR / Spark**

- Launching an EMR Cluster from Airflow
- Submitting a Spark job that reads the files from the staging S3 bucket, calculate / aggregate, write the result in parquet file using Iceberg
- Having an Airflow Sensor to check the status of the job

What we also tried:

- DeltaLake
- Pandas



Amazon EMR

Transform (Live Query)

In order to read the data from S3, provide fast queries & enable data discovery for power users, we need a query engine.

What we chose: **Dremio**

- Provides fast queries directly over S3 using the Iceberg table formats
- Uses SQL Language, easy to use and implement business logic
- Enhanced end user experience

What we also tried:

- Amazon Athena, Redshift Spectrum
- Hive & Impala

Scheduling & Orchestration

I just need my flows to run at a given time and in a given order

Scheduling & Orchestration

What we chose: **Airflow MWAA**

- We create DAG's that are importing sets of tables
- Each DAG calls 2 dynamic scripts
- The DAG's are easy to create & maintain
- We can start importing data very fast
- Scalable infrastructure managed by AWS

What we also tried:

- Amazon Glue
- Amazon Data Pipeline

Visualization

Showing the data to the world

Visualization

What we chose: **Tableau**

- Used for creating dashboards & analytics
- Directly connects to the Dremio cluster to run queries
- Has slick visualizations
- Is customizable & highly dynamic
- Very user friendly and offers data exploration capabilities

What we also tried:

- Power BI
- Superset

Computing Power

We need to run our workloads somewhere…

Computing Power

Tableau Instance

- m4.4xlarge – 64 GB RAM, 16 CPU's
- Scalability options: backup & restore on a more powerful machine

Dremio Instance

- Coordinator: m5d.2xlarge – 16 GB RAM, 4 CPU's
- Executors: 2 x r5d.4xlarge – 128 GB RAM, 32 CPU's
- Scalability options: from Dremio, you can launch as many engines and executors as you need. Or you can migrate to Dremio Cloud.

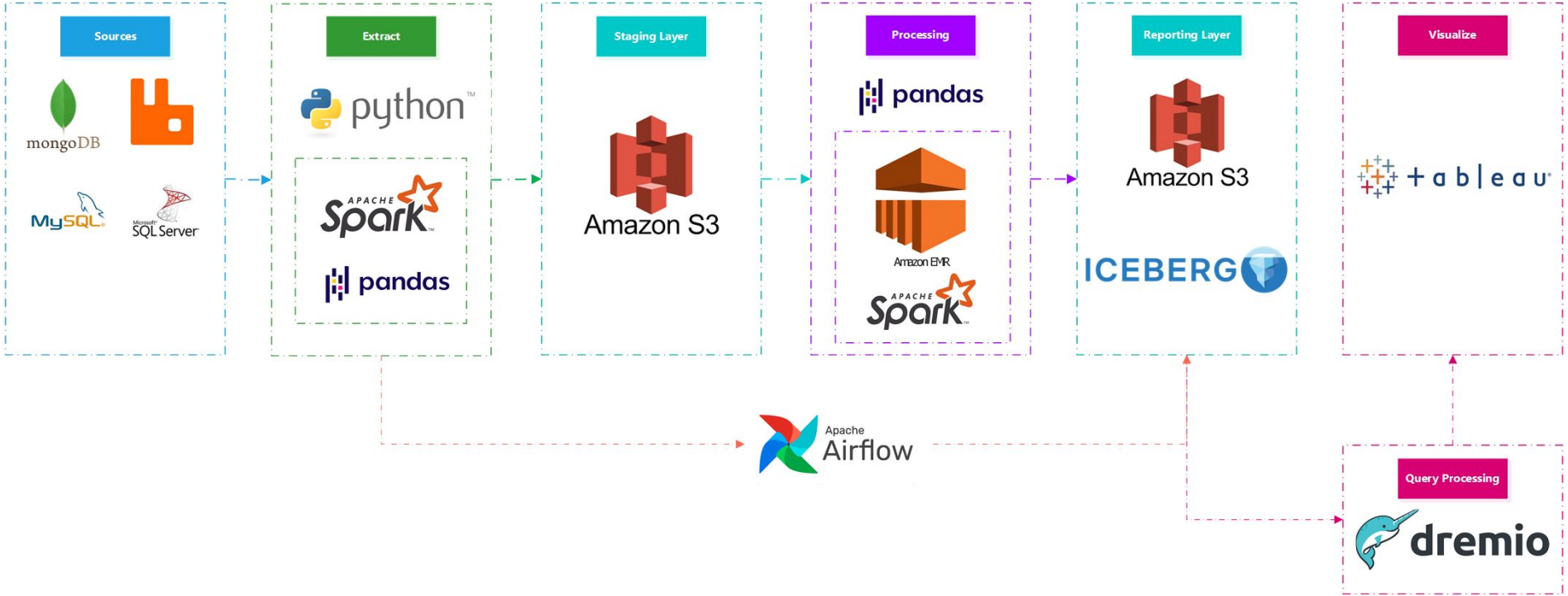
Airflow Instance

- mw1.large – 8 GB RAM, 4 CPU's
- Scalability options: from the MWAA environment you can select the type of instance you need

Putting it all together

How will our BI infrastructure look like?

BI Architecture



Short Demo

Let’s see it in action

Rabbit MQ Queue

Queue dwh.vbrk_delta

Overview

Queued messages last minute ?

Ready	51
Unacked	0
Total	51

Queue dwh.vbrp_delta

Overview

Queued messages last minute ?

Ready	1,116
Unacked	0
Total	1,116

VBRK = Billing Header

VBRP = Billing Line

Airflow Dag Parameters

```
#declare the path where the called script is
#running_on_airflow = 1 =>> runs only on aws // = 0 on local
running_on_airflow = 1
is_test_queue = 0
empty_queue = 1
is_full = 0
move_to_archive = 1
```

Parameters that help us run different environments in one dag

```
#parameters to be filled
dag_id = 'Import_Billing_Data'
table_name = ['VBRK','VBRP']
uq = [['VBELN'], ['VBELN','POSNR']]
columns = [[], []]

partition_key = ['ERDAT','ERDAT']
partition_type = ['YearMonth','YearMonth']
```

Parameters that help us manipulate & merge the data in the tables

```
query = ['SELECT * FROM QueryAnalysis.Apps.AirflowBillingHeaderValidation', 'SELECT * FROM QueryAnalysis.Apps.AirflowBill
```

Parameters used to execute data quality checks

```
sender = ["airflow@freshful.ro"]
recipients = [["marius.costin@emag.ro"]]
subject = ["[High] Data Validation for Billing Header", "[High] Data Validation for Billing Lines"]
body = ["Data Validation Checks failed for Billing Header", "Data Validation Checks failed for Billing Line"]
```

Airflow DAG

```
with DAG(dag_id, schedule_interval = '55 * * * *', default_args = default_args, catchup = False) as dag:

    start = BashOperator(task_id = 'start', bash_command = 'sleep 1',execution_timeout=timedelta(seconds=900), retries = 2,

    with TaskGroup(Task1) as run_task_1:
        import_rabbitConsumer = PythonOperator(task_id = 'import_{}'.format(table_name[0]),execution_timeout = timedelta(sec
            python_callable = import_rabbit,
            op_kwargs={'queue':queue_name[0], 'table_name' : table_name[0], 'file_name' : file_names[0],'is_test_queue': is_

                                                                                                    = timedelta(seconds

            python_callable = pds_refresh,

        import_rabbitConsumer >> process_sap_data >> refresh_pds_dremio >> data_checks
```

Step 3: Calls the **pds_refresh** script, which triggers a Dremio API call, telling it to refresh all the reflections that contains that table.

Step 1: Calls the script **import_rabbit**, which reads & imports the data from the queues into the staging bucket.

Ess

Step 4: Calls the **query_datasets** script, which triggers a Dremio API call, running some scripts that check the data. An email will be sent with the results if issues are detected.

Step 2: Calls the **process_sap** script, which reads the imported files from Staging, merges the data based on the unique key, partitions the data by a date column and writes the partitioned file in the Processed folder. Essential parameters: table_name, unique key & partition key.

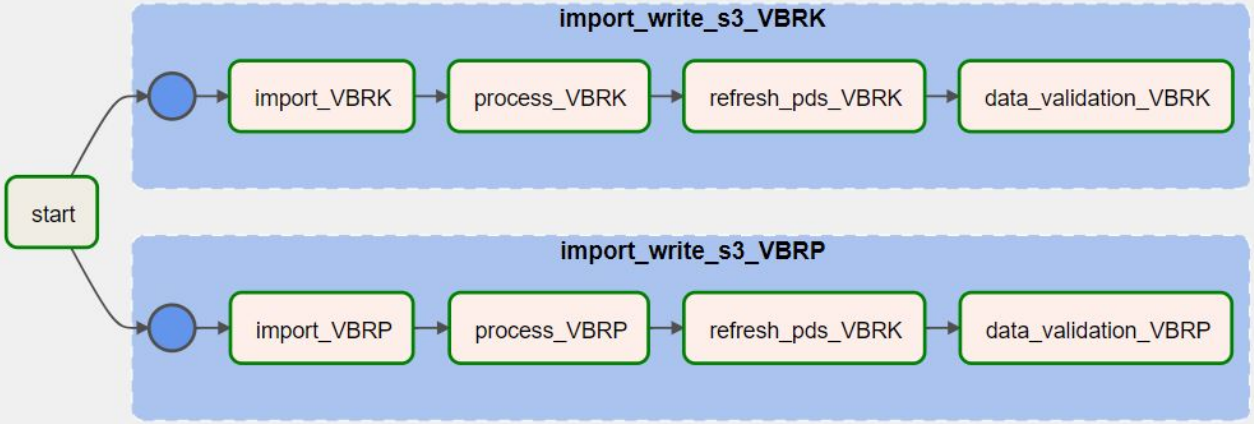
Airflow DAG
How the code actually looks like

DAG: Import_Billing_Data

Tree View | Graph View | Calendar View | Task Duration | Task Tries | Landing Times | Gantt | Details | Code

2022-02-09T09:52:41Z | Runs 25 | Run manual__2022-02-09T09:52:40.327465+00:00 | Layout Left > Right | Update

BashOperator PythonOperator



S3 Staging Layer
Where the data lands first

Amazon S3 > bi-staging > SAP/ > VBRK/

Staging Bucket / Data Source / Table Name

VBRK/

Objects | Properties

Parquet filename with unix timestamp

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Find objects by prefix

1

Name	Type	Last modified	Size	Storage class
VBRK_1644400567_4613926.parquet	parquet	February 9, 2022, 12:25:59 (UTC+02:00)	66.4 KB	Standard

S3 Buffer Layer
Checking for corruption

Amazon S3 > bi-buffer > SAP/ > VBRK/

Buffer Bucket / Data Source / Table Name

VBRK/

Partitioned merged file checked for corruption

Objects | Properties

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Find objects by prefix

1

Name	Type	Last modified	Size	Storage class
VBRK_202202.parquet	parquet	February 9, 2022, 12:27:40 (UTC+02:00)	671.8 KB	Standard

S3 Reporting Layer
Where the magic happens

Amazon S3 > processed > SAP/ > VBRK/

Processed Bucket / Data Source / Table Name

VBRK/

Objects | Properties

The files that create the Physical Dataset In Dremio

Objects (7)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

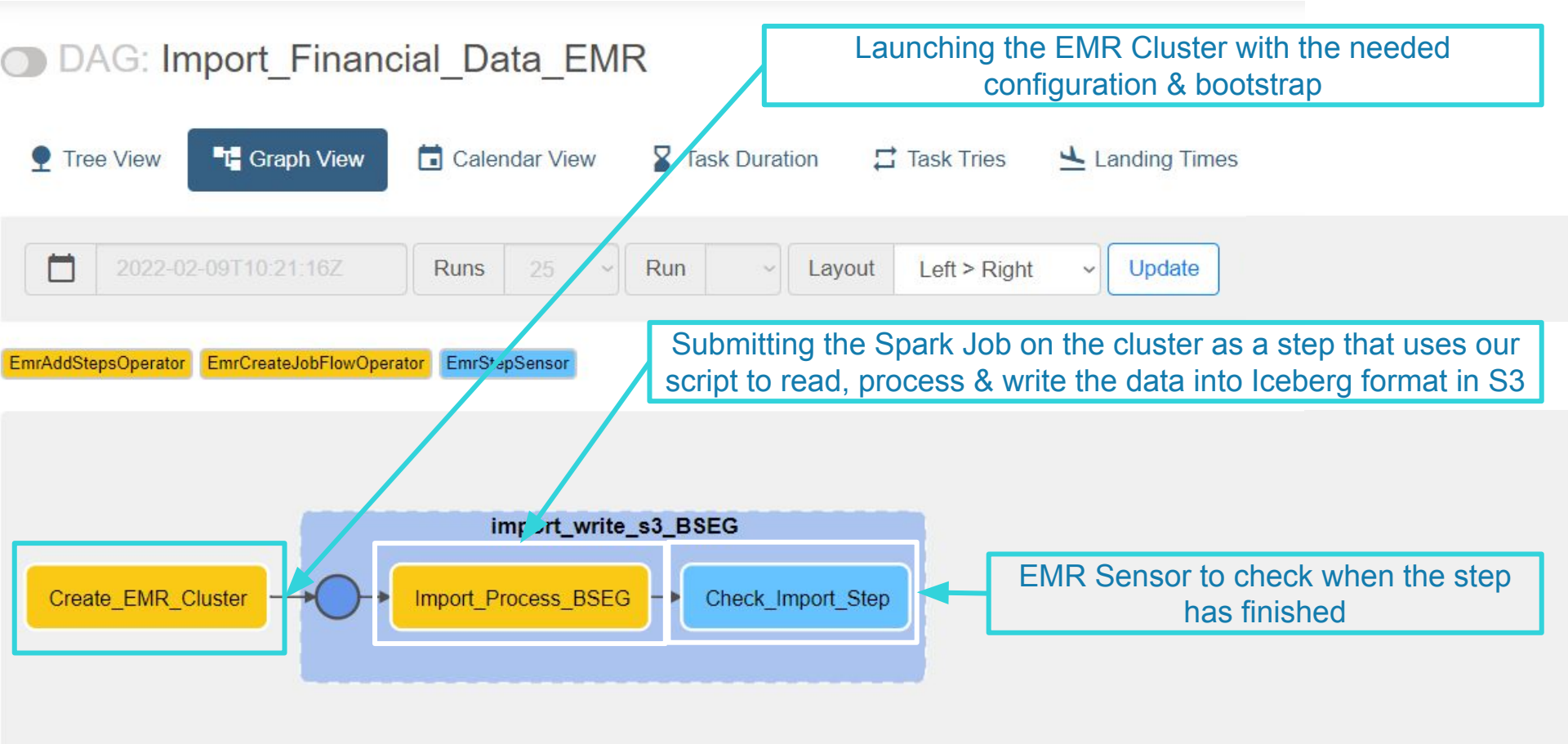
Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Find objects by prefix

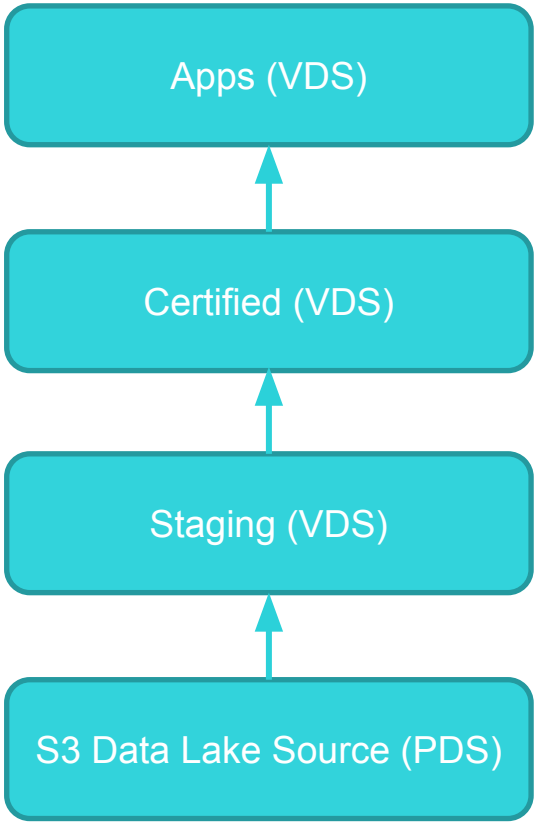
1

Name	Type	Last modified	Size	Storage class
VBRK_202108.parquet	parquet	October 26, 2021, 15:03:47 (UTC+03:00)	74.1 KB	Standard
VBRK_202109.parquet	parquet	October 26, 2021, 15:03:48 (UTC+03:00)	83.3 KB	Standard
VBRK_202110.parquet	parquet	November 11, 2021, 14:55:21 (UTC+02:00)	284.9 KB	Standard
VBRK_202111.parquet	parquet	November 30, 2021, 21:55:16 (UTC+02:00)	728.8 KB	Standard
VBRK_202112.parquet	parquet	January 25, 2022, 16:55:17 (UTC+02:00)	1.4 MB	Standard
VBRK_202201.parquet	parquet	January 31, 2022, 23:55:22 (UTC+02:00)	1.9 MB	Standard
VBRK_202202.parquet	parquet	February 9, 2022, 11:56:35 (UTC+02:00)	671.8 KB	Standard

EMR DAG
Launching an EMR Cluster & Processing a Spark job



Dremio Folder Structure



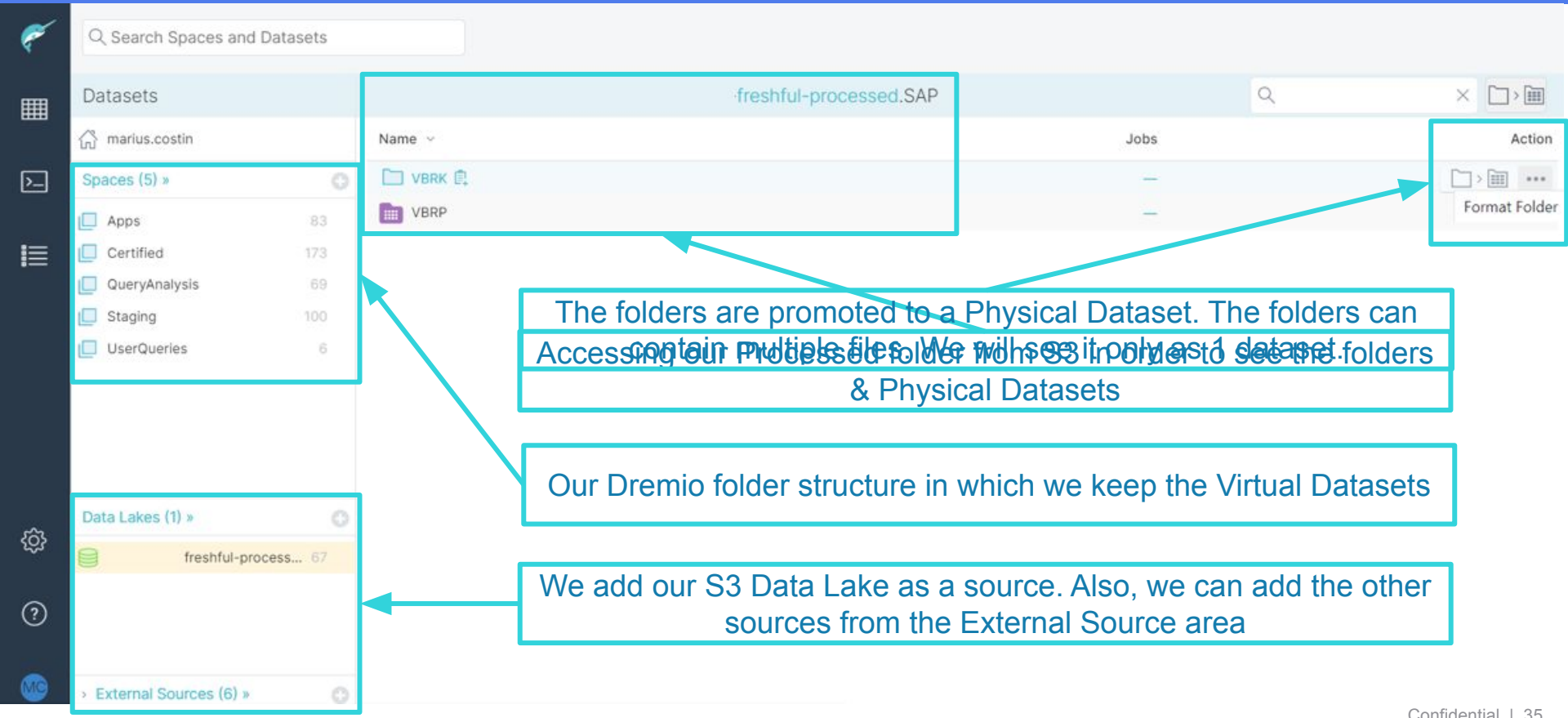
Virtual datasets to be used in Tableau containing a select from the Certified Layer

Intermediary virtual datasets that contains business logic & reflections

Virtual datasets containing a select from the PDS with the needed columns

Physical Datasets over the S3 folders in the data lake

S3 Data Lake Source



The folders are promoted to a Physical Dataset. The folders can contain multiple files. We will see it only as 1 dataset.

Accessing our Processed folder from S3 in order to see the folders & Physical Datasets

Our Dremio folder structure in which we keep the Virtual Datasets

We add our S3 Data Lake as a source. Also, we can add the other sources from the External Source area

Staging Layer

Search Spaces and Datasets

Datasets

marius.costin

Spaces (5) »

Apps	83
Certified	173
QueryAnalysis	69
Staging	100
UserQueries	6

Staging.SAP

Name

VBRK

VBRP

VBRK (edited)
Staging.SAP

Data

Catalog

Reflections

SQL Editor

```
1 SELECT MANDT, VBELN, FKART, FKTYP, VBTYP, WAERK, KNUMV, FKDAT,
2 NETWR,ERDAT, XBLNR, ZUONR, MWSBK, KUNAG,ERZET
3 FROM VBRK
```

Certified Layer

Datasets

marius.costin

Spaces (5) »

Apps	83
Certified	173
QueryAnalysis	69
Staging	100
UserQueries	6

Certified.Financial

Name

FactBilling

FactBilling (edited)
Certified.Financial

Data | Catalog | Reflections

SQL Editor

```
SELECT A.VBELN AS InvoiceIdSAP, B.POSNR AS InvoiceLineIdSAP,
A.FKDAT as BillingDate,A.ERDAT as InvoiceDate,
A.KUNAG as CustomerId,
B.MATNR  as ProductId, C.ProductName,
B.FKIMG as Quantity
FROM Staging.SAP.VBRK A LEFT JOIN Staging.SAP.VBRP B ON A.VBELN = B.VBELN
LEFT JOIN Certified.MasterData.DimProducts C ON B.MATNR = C.ProductId
```

Apps Layer

Datasets

marius.costin

Spaces (5) »

Apps	84
Certified	173
QueryAnalysis	69
Staging	100
UserQueries	6

Apps.Financial

Name

FactBilling

FactBilling
Apps.Financial

Data

Catalog

Reflections

SQL Editor

```
1 SELECT InvoiceIdSAP,InvoiceLineIdSAP,BillingDate,InvoiceDate,
2 ProductId, ProductName,
3 Quantity
4 FROM Certified.Financial.FactBilling
```

Tableau Creator

Connections Add

Apps.Financial.FactBilling
Other Databases (ODBC)

Database

DREMIO

Table

Enter table name

Exact Contains Starts with

New Custom SQL

Apps.Financial.FactBilling

Migrated Data

Need more data?

Drag tables here to relate them. Learn more

Migrated Data

7 fields 1386333 rows

Name

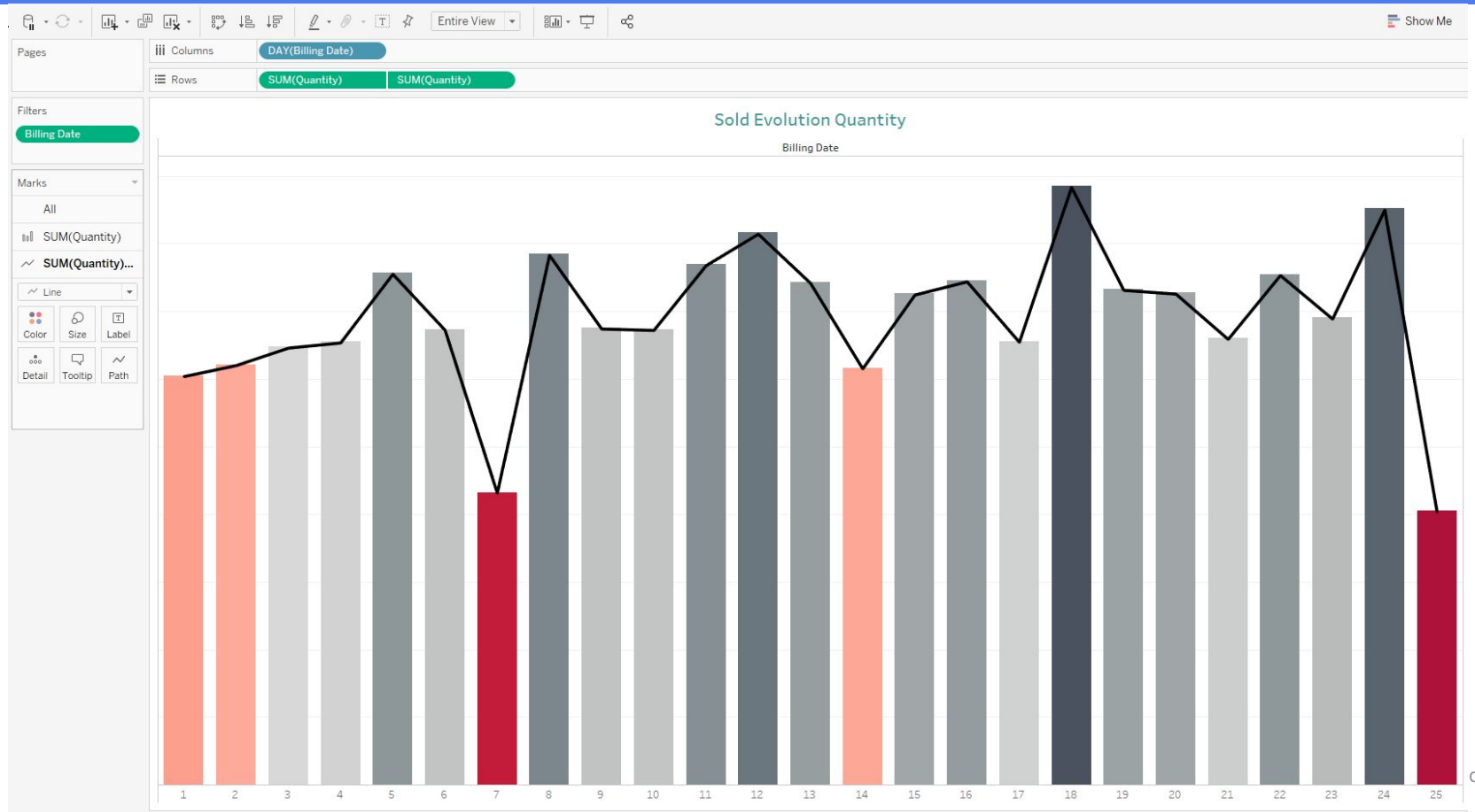
Migrated Data

Fields

Type	Field Name	Physical Table	Remote ...
Abc	Invoice Id Sap	FactBilling	InvoiceIdS...

Abc FactBilling Invoice Id Sap	# FactBilling Invoice Line Id Sap	Abc FactBilling Billing Date	Abc FactBilling Invoice Date	Abc FactBilling Batch Id	Abc FactBilling Product Id	# FactBilling Quantity
					SP00000	1.0000
					100101240	1.0000
					100101241	1.0000
					100086879	1.0000
					100071254	1.0000

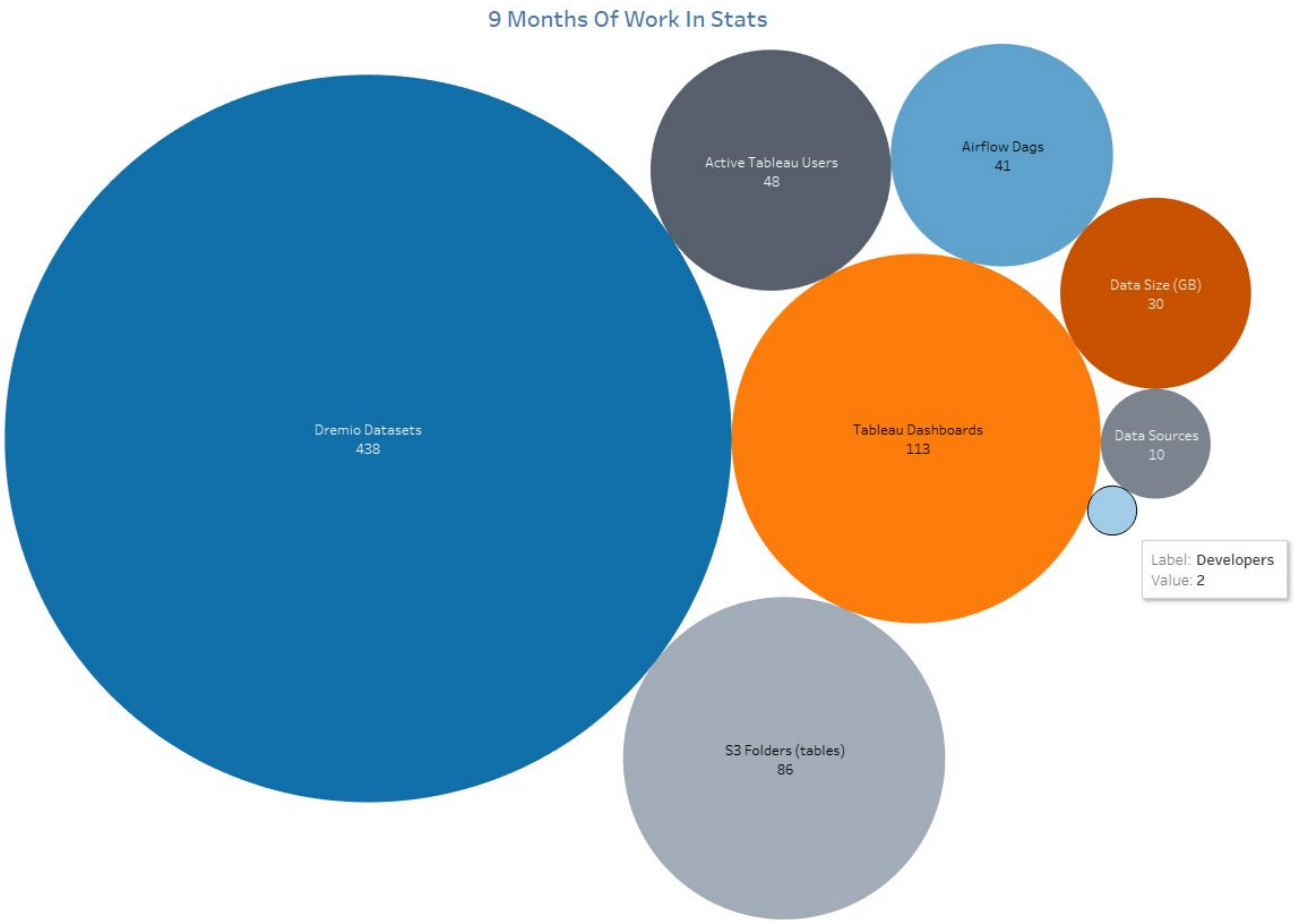
Small Tableau Evolution Chart



Some Stats

From our journey

9 Months Into Our Journey

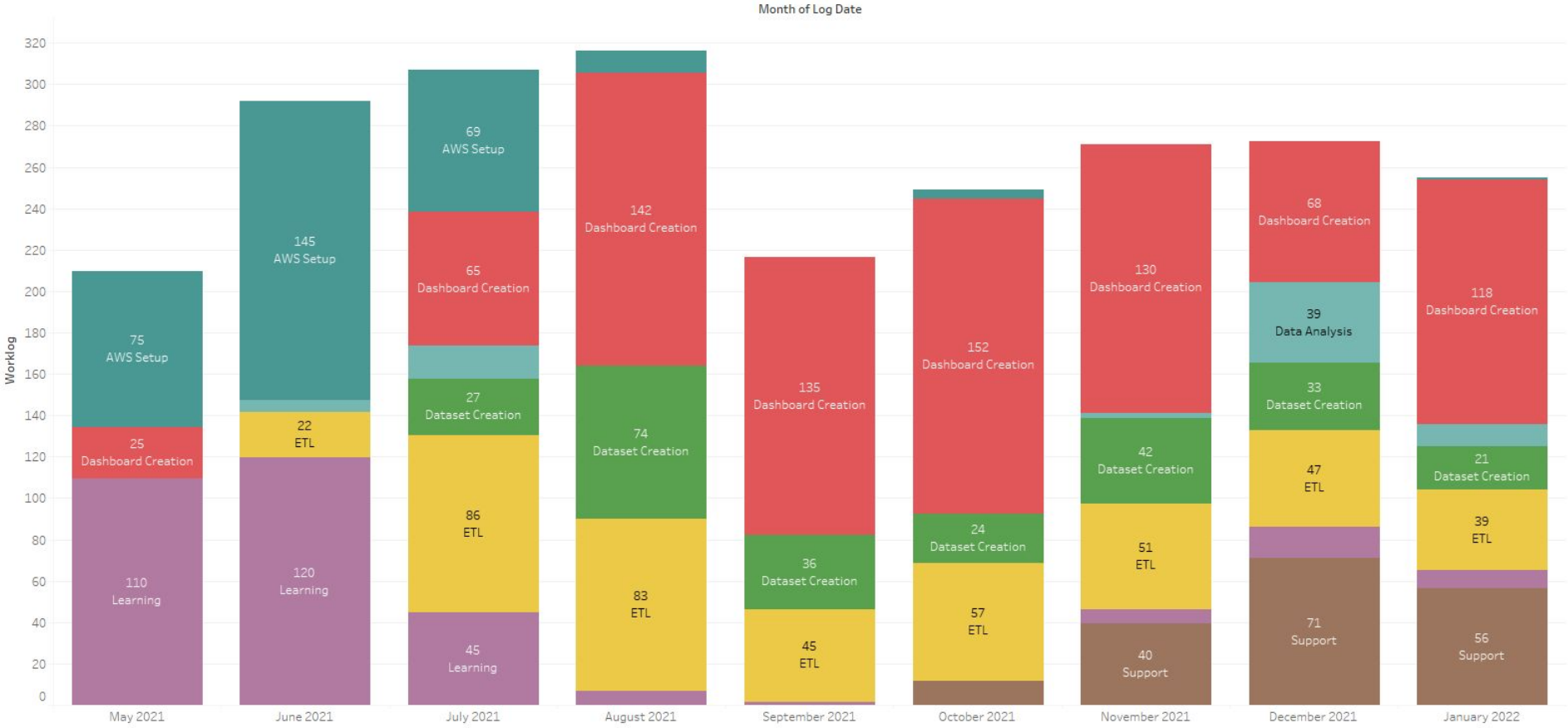


2
Number Of Devs

315
Hours Spent Learning

28 Days
First Dashboard In Production

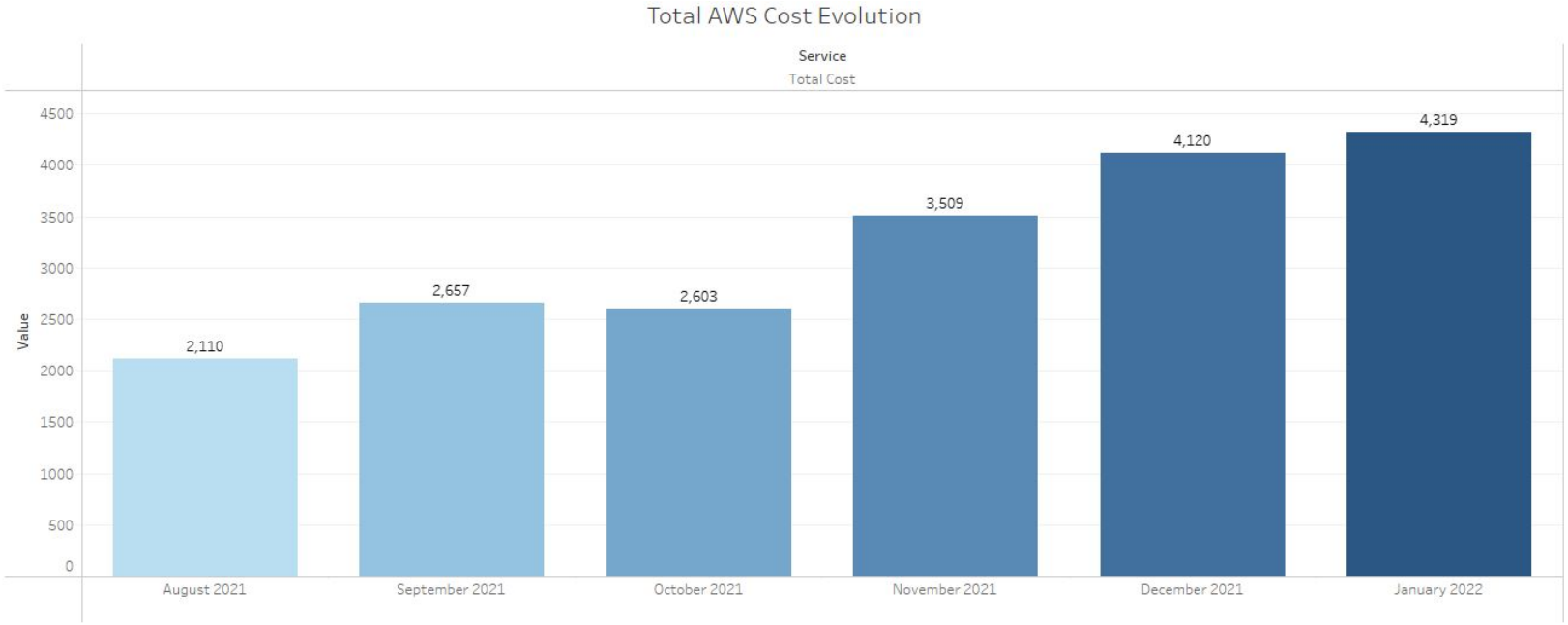
Number Of Hours Spent Working On Different Task Types



Costs

Everything has a price

Costs



Monthly AWS Costs ($)

Service	August 2021	September 2021	October 2021	November 2021	December 2021	January 2022
EC2 Instances	1,219	1,206	1,570	2,098	2,383	2,258
MWAA	364	352	403	530	549	617
RDS	31	30	32	54	146	162
S3	50	568	81	121	173	251
Tableau Licenses	446	951	517	706	869	1,031
Grand Total	2,110	3,107	2,603	3,509	4,120	4,319

Future Evolutions

What are we looking at for the future

- Integrating Debezium & Iceberg for real time data pipelines
- Analyzing the prospect of migrating to Dremio Cloud
- Creating Airflow Sensors for our PLT Dags
- Analyzing the prospect of bringing in a new ETL tool
- Focusing on data lineage & data catalog

Resources

Useful links for further reading

- https://aws.amazon.com/managed-workflows-for-apache-airflow/
- https://docs.dremio.com
- https://www.tableau.com/learn/get-started
- https://aws.amazon.com/blogs/big-data/orchestrating-analytics-jobs-on-amazon-emr-notebooks-using-amazon-mwaa/
- https://public.tableau.com/
- https://aws.amazon.com/ec2/pricing/on-demand/
- https://debezium.io/blog/2021/10/20/using-debezium-create-data-lake-with-apache-iceberg/

Subsurface LIVE

The Cloud Data Lake Conference

Thank You

LinkedIn: https://www.linkedin.com/in/mariuscostin/
Email: Marius.costin@emag.ro