

FINAL PROJECT – BIG DATA CONCEPTS: **REPORT (CLOUD COMPUTING)**

Name: Jonathan Hansen

Date: 11.21.2024

INTRODUCTION

This project was developed to explore a big data cloud environment and cloud computing resources, namely in Azure. It is geared toward building the initial stages of a cloud computing pipeline that can be used for machine learning models. The aim of this project is to implement a portion of a data pipeline within the cloud environment, convey some of the complexities that come along with developing in the cloud, and show how the resources leveraged in the cloud environment provide excellent data handling at a large scale. I chose azure specifically because it is one of the top tier cloud computing platforms, offers a wide range of services, is used often within the geolocation market (Nordics) I am interested in working, and provides great documentation for how to use the service.

The benefits of using Azure or any cloud computing resource are the flexibility, increased capacity, potential cost savings, increased scalability, and increased availability and reliability that it provides versus other on-prem systems or hybrid systems. While there are some risks involved in using cloud services for many applications, the benefits outweigh the risks. For many machine learning applications the usefulness and benefits of cloud services do provide substantial benefits.

BACKGROUND

WHY THIS PROBLEM OR TOPIC WAS SELECTED

Modern data machine learning pipelines often require vast amounts of data and an architecture that can clean, transform, and store that data on very a large scale efficiently and be used widely. The backbone of machine learning models today is often the ability to deal with data on a very large scale effectively. This is why I have had the goal of learning Azure Cloud Services and why cloud computing was chosen to investigate.

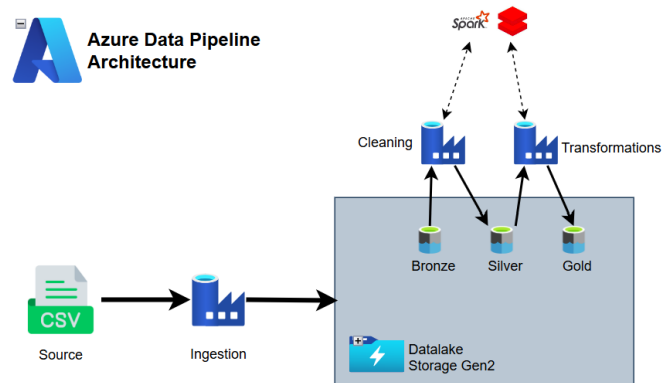
WHY IT IS INTERESTING / IMPORTANT TO ADDRESS,

Focusing on the benefits of cloud computing allows us to see why cloud computing is so important in today's modern data pipelines. These benefits include flexibility, increased capacity, cost savings, increased scalability, and increased availability and reliability. The flexibility of the cloud gives us the ability to transform and manage our data pipelines from anywhere and from different devices. This is important in today's world where many companies are spread throughout the world and there are multiple employees working on parts of the infrastructure. In machine learning today it is important to be able to handle big data. The cloud resources today are created to provide large scale handling of without having to worry about limitations. While it is important to be careful, often using cloud services can provide significant cost savings vs an on-premises resource. Cloud services dynamically scale if demand goes up or down, and having this scalability is very helpful in saving money on resources when you are not using them. It also provides the ability to not worry about capacity at peak times. The redundancy measures built into cloud services are very important and these are mentioned later when creating our datalake storage. This redundancy provides backups from data in case of disruptions on multiple different kinds of events.

METHODOLOGY

The project consists of a data pipeline built within the Azure Cloud environment. The data used was from the 'Bureau of Transportation Statistics'. The portion of the data pipeline focused on in this project is the Extract, Load, and Transform (ELT). The pipeline would run by extracting data from a source and loading the data into a datalake in Azure. Next the data would be sent to the Data Factory for some cleaning and processing of the data before being dumped into the silver container. After this the data would once again be sent to the Data Factory for further transformations and splitting of the data into categories appropriate for machine learning models and sent back to the datalake into the gold container.

Data pipeline architecture for Big Data – Final Project



My goal in this project was to build the groundwork for a machine learning model in Azure Cloud Services. The implementation of this data pipeline in Azure could be set to automatically run. For cost reasons this pipeline was not set up to have fully automated runs. However, there are parts of the pipeline that auto run when triggered manually. To automate the remaining portions of the pipeline are straight forward given necessary monetary resources to do so.

In this project several concepts from the modules were consulted and used in choosing which services were best suited for the project. Cloud Computing being the main technology used, with supporting modules being, data cleaning via Data Factory, distributed processing via Spark Cluster accessed by Data Factory, distributed storage via Azure datalake gen2 storage and data pipelines was consulted when thinking about the project as a whole.

Since there is no code to be shared, below you will find the detailed steps taken that show how services and resources were created, and how the data was processed.

ENVIRONMENT SETUP:

When planning to move to a cloud-based option it is important to note whether this will save the user money. There are great calculators built into Azure that will assist in predicting how much expense will occur for setup and running of the Azure resources.

Step 1 (After creating the Azure Cloud Services account and logging in to the account): Create Resource Group – click on Create Resource Group in the top left-hand corner of the Azure portal and fill out the following information. It's best to select the region at this stage in the region that your other resources will be created. So, if the app or product will be used mostly in eastern united states then this should be the region or if the app will be used mostly in Europe then a European region would be the best fit.

Create a resource group

Basics Tags Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

Project details







Subscription *

Resource group *

Resource details

Region *

Resources

Recent	Favorite
Name	
 datalakestorage-dep	
 jh-data-engineering-project	
 af-jhdep-01	
 adf-jhdep-01	
 db-jhdep-01	
 kv-jhdep-01	

Step 2: After creating your Resource Group I created the Azure Services I wanted to use in my data pipeline.

Here is the list of resources I created for the pipeline. The ones highlighted in red are the ones used in this pipeline.

Step 3: Creating these resources is relatively straightforward. First, from the home page I clicked on create a resource and searched for the resource that I wanted to create. I clicked the 'Create' button next to the icon of the resource I wanted to create.

Fill out the necessary information from the screenshot.

For each resource there are differing configurations for the creation of the resources. It was important to know these differences, and, in some cases, why different selections would be more beneficial.

For Azure Data Factory and Azure Key Vault some of these included whether to allow public or private endpoints, tags, instance region and version.

For Azure Storage Account Gen2 (Data Lake) the important configurations included:

- 'Azure Blob Storage or Azure Data Lake Storage Gen 2' – This gives the option to make the storage account a data lake that has a hierarchical namespace.
- 'Region' – select regions closest to where the Data Lake is most used.
- 'Redundancy' – Can choose locally-redundant storage, Geo-redundant storage, zone-redundant storage, or geo-redundant storage. For this project I chose Geo-redundant storage as it was a low-cost option that had a secondary regional backup. This way if there is a regional catastrophe in the primary region there is another secondary region that has a backup.
- Under the 'Advanced Section' be sure to select 'enable hierarchy namespace' this will create the datalake storage instead of a blob storage.

Microsoft Azure

Home > Create a resource >

Create Data Factory

Basics Git configuration Networking Advanced Tags Review + create

One-click to create data factory with sample pipeline and datasets. [Try it](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group *

Instance details

Name *

Region *

Version *

INGESTION:

Creating an Integration Runtime on local device:

There were a couple of methods I tried to implement. The first of these was ingesting from a local file.

Step 1: Open Azure Data Factory

Step 2: Go to the 'Manage' tab and click on 'Integration Runtimes'.

Step 3: Click on '+ New'. This will open a panel on the right-hand side of the screen.

Step 4: Click on 'Azure, Self-Hosted' tab and continue.

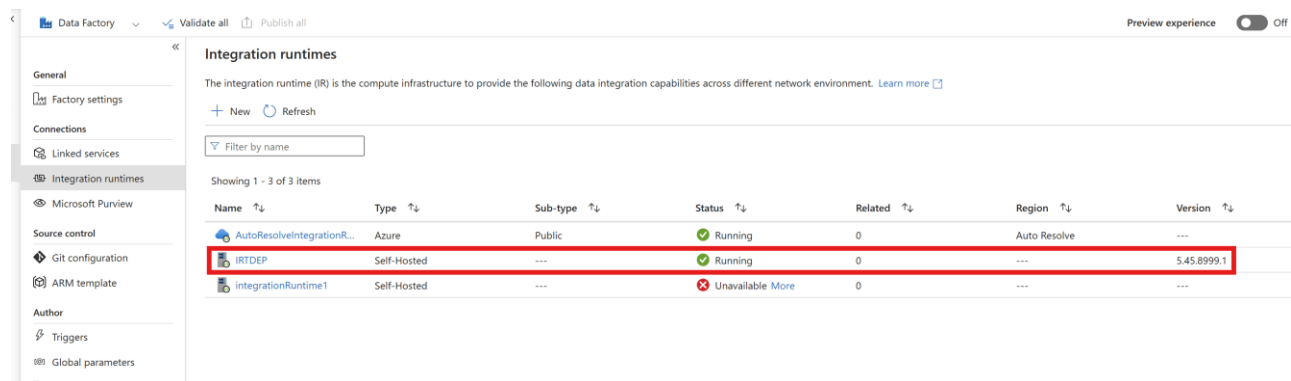
Step 5: Click on 'Self-Hosted' if using for an on-premises or private network or you can alternately use a existing self-hosted integration runtime that is used for another resource and click 'Continue'.

Step 6: Click 'Create' on the next page. This will initiate the runtime and create the icon in your azure data factory. Note this does not connect your integration runtime to your data factory listing. You will set up the runtime integration on the next page.

Step 7: On this page you will select 'Option 2: Manual setup' and click on the link called 'Download and install integration runtime'. Be sure to not close this page as you will need the listed 'Key1' and 'Key2' when setting up the integration runtime.

Step 8: You will follow the directions to download and install the integration runtime on your local device. Copy the authentication key from step 7 into the box before finishing.

This is what it will look like when complete:



Name	Type	Sub-type	Status	Related	Region	Version
AutoResolveIntegrationR...	Azure	Public	Running	0	Auto Resolve	---
IRTDPEP	Self-Hosted	---	Running	0	---	5.45.8999.1
integrationRuntime1	Self-Hosted	---	Unavailable More	0	---	---

Connecting to a Local Folder:

Step 1: Go to 'Linked Service' under the 'Manage' tab. Then click on the '+ New' button which will open panel on the righthand side.

Step 2: Select the 'File system' linked service.

New linked service

File system [Learn more](#)

Name *

FileServer1

Description

Connect via integration runtime • ⓘ

AutoResolveIntegrationRuntime

Host • ⓘ

e.g. \\ServerName\SharedFolder\<Folder>, \\<storage name>.file.core.windows.net\file service

User name *

Password *

Annotations

+ New

> Parameters

> Advanced ⓘ

Step 3: This will take you to the next page.

You will need to input the file pathway and assure proper sharing has been enabled on your file. Then you will add the username of your computer and the password for your computer.

(This is the part that did not work for me. When I input this information it gave me an error, however it gives very little information. Looking up the error code it looks like a problem with my username password combination. I could see this might be this issue as my login is facial recognition. I tried other methods to get this working with no success)

TRANSFORMATIONS:

Initially I planned to mount my data lake from a Databricks compute cluster, however this did not end up work, I believe this was because the IU account I was using lacked proper administrator permissions to create a Microsoft Entra ID service principle and provide it access to the azure storage accounts. There was an additional issue with Databricks not having a compute cluster accessible via the student account credits I had available. Because of this issue I investigated other solutions Azure Functions App – which would allow me to write in a notebook but looked difficult to configure – and Azure Data Factory – which was easier to configure and commonly used, but I did not know how to use data processing and transformations in Data Factory. I selected the latter of these two as I was able to find a lot of good tutorials of how to implement data processing and transformation steps.

Connecting to containers:

▲ Datasets 4

ImportfromBronze

● ImportfromSilver1

● SinktoGold

● SinktoSilver

I created 2 datasets. The first one 'ImportfromBronze' connects to the csv file in the datalake bronze container. The second one is connecting to the datalake silver container.

Here are the steps to connect to the datalake container. In this case I am Importing from the Silver Container.

Step 1: Click on the 'Author' tab on the left-hand side of the screen. Then go to datasets and click on 'ellipses' and click on New Dataset.

Step 2: A panel on the right-hand side will open and I clicked on the 'Azure Data Lake Storage Gen2'.

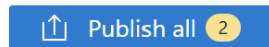
Step 3: Select the file format to import. I selected the 'DelimitedText'.

Step 4: I Filled out the information as listed in the corresponding picture with the Linked service and corresponding file path.

After pressing 'OK' I tested the connection which was successful.

Step 5: Can view schema by clicking on the 'Schema' tab.

Step 6: It is necessary to click on the publish tab in the top left-



Set properties

Name
ImportfromSilver

Linked service *
AzureDataLakeStorage1

File path
silver / Directory / File name

First row as header ☒

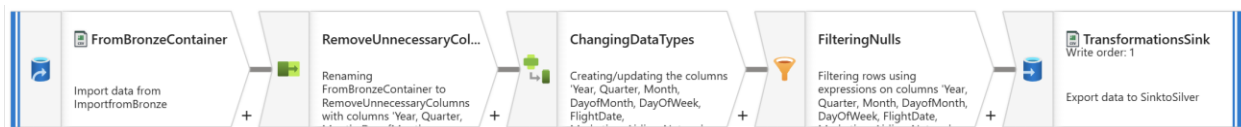
Import schema
☒ From connection/store
 ☐ From sample file
 ☐ None

hand corner for changes to take effect.

This completes connecting a dataset to data factory and publishing to sink from data factory.

Data Flows in Data Factory (Data Processing and Transformations):

The first data flow I worked on was the data processing step.



Step 1: To create this data flow I first connected my dataset called 'ImportfromBronze'

Source settings | Source options | Projection | Optimize | Inspect | Data

Output stream name *
FromBronzeContainer

Description
Import data from ImportfromBronze

Source type *
Dataset | Inline

Dataset *
ImportfromBronze

Options
☒ Allow schema drift ⓘ
☐ Infer drifted column types ⓘ
☐ Validate schema ⓘ

Skip line count

Sampling * ⓘ
☐ Enable ☒ Disable

Step 2: I removed all unnecessary columns that either had no information or very little information and contained mostly nulls values. I did this by clicking on the trash can button signified below. This step condensed my dataset from 119 columns to 66 columns. The goal is to minimize this even more from the Silver to Gold processing, separating this dataset into specified sets for specific machine learning model predictions.

Select settings Optimize Inspect Data preview

Output stream name * RemoveUnnecessaryColumns [Learn more](#)

Description Renaming FromBronzeContainer to RemoveUnnecessaryColumns with columns 'Year, Quarter, Month, [Reset](#)

Incoming stream * FromBronzeContainer

Options ☒ Skip duplicate input columns ☒ Skip duplicate output columns

Input columns * ☐ Auto mapping [Reset](#) [Add mapping](#) [Delete](#) 66 mappings

FromBronzeContainer's column	Name as	
abc Year	Year	+
abc Quarter	Quarter	+
abc Month	Month	+
abc DayofMonth	DayofMonth	+
abc DayOfWeek	DayOfWeek	+

Step 3: In this step I changed the columns to the correct formatting. The majority of columns were changed to and integer format while date was changed to a date format.

Derived column's settings Optimize Inspect Data preview

Output stream name * ChangingDataTypes [Learn more](#)

Description Creating/updating the columns 'Year, Quarter, Month, DayofMonth, DayOfWeek, FlightDate, [Reset](#)

Incoming stream * RemoveUnnecessaryColumns

[Add](#) [Clone](#) [Delete](#) [Open expression builder](#)

Columns * ①

Column	Expression
FlightDate	toDate(FlightDate)
Year	toInteger(Year)
Quarter	toInteger(Quarter)
Month	toInteger(Month)
DayofMonth	toInteger(DayofMonth)
DayOfWeek	toInteger(DayOfWeek)
Marketing_Airline_Network	toInteger(Marketing_Airline_Network)

Step 4: I next filtered null values. I did this using the expression builder this code 'isNull(column_name)' to filter the null values in columns.

Filter settings Optimize Inspect Data preview

Output stream name * FilteringNulls [Learn more](#)

Description Filtering rows using expressions on columns 'Year, Quarter, Month, DayofMonth, DayOfWeek, FlightDate, [Reset](#)

Incoming stream * ChangingDataTypes

Filter on * `isNull(Year) && isNull(Quarter) && isNull(Month) && isNull(DayofMonth) && isNull(DayOfWeek) && isNull(FlightDate) && isNull(Marketing_Airline_Network) &&`

Step 5: In this step of the dataflow, I ended the flow with a sink connecting the flow to the 'SinktoSilver' dataset, exporting the file in batch to the silver container.

Sink Settings Errors Mapping Optimize Inspect Data preview

Output stream name * [Learn more](#)

Description [Reset](#)

Incoming stream *

Sink type * Dataset Inline Cache

Dataset * [Test connection](#) [Open](#) [New](#)

Skip line count

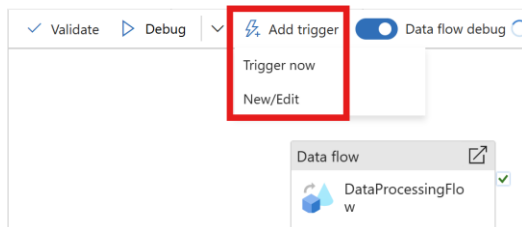
Options ☒ Allow schema drift [?](#) ☒ Validate schema [?](#)

Step 6: Publish all changes.

Note: Starting the data preview helped to preview the data at each step to allow me to confirm the data processing steps before moving on.

Pipelines In Data Factory:

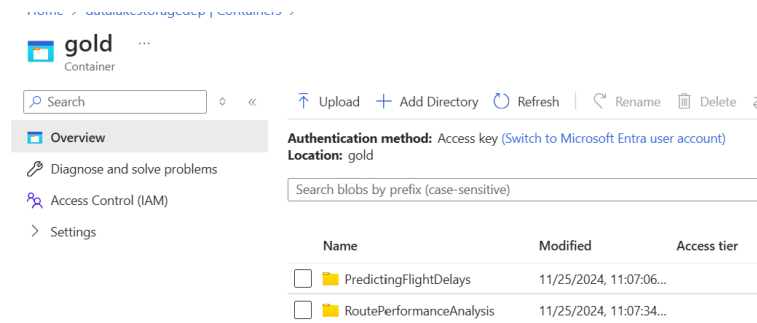
To run a dataflow in Data Factory it is necessary to create a pipeline. This is easily done once the dataflow has been created. Simply select the dataflow (in my case 'DataProcessingFlow') from the 'Data Flows' from within the 'Author' tab and drag and drop it into the Pipelines builder. It is then easily possible to add triggers that are either specified by time or they can be manually triggered from the 'add trigger' button from the pipeline builder.



By pressing the trigger now button we can see the resulting output of 4 files into the silver container along with a success notification.

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
part-00000-c2db0340-7557-42e2-8a43-70f08270bd76-c000.csv	11/20/2024, 3:37:26 PM	Hot (Inferred)		Block blob	0 B	Available
part-00001-c2db0340-7557-42e2-8a43-70f08270bd76-c000.csv	11/20/2024, 3:37:25 PM	Hot (Inferred)		Block blob	47.57 MiB	Available
part-00002-c2db0340-7557-42e2-8a43-70f08270bd76-c000.csv	11/20/2024, 3:37:25 PM	Hot (Inferred)		Block blob	47.67 MiB	Available
part-00003-c2db0340-7557-42e2-8a43-70f08270bd76-c000.csv	11/20/2024, 3:37:25 PM	Hot (Inferred)		Block blob	46.55 MiB	Available
part-00003-c2db0340-7557-42e2-8a43-70f08270bd76-c000.csv	11/20/2024, 3:37:25 PM	Hot (Inferred)		Block blob	45.14 MiB	Available
SUCCESS	11/20/2024, 3:37:26 PM	Hot (Inferred)		Block blob	0 B	Available

The transformations from silver to gold were completed in the same fashion as the above transformations from bronze to silver. I did add directories to the gold container that would hold the fully processed data that is ready for data modeling.



SECURITY/AZURE KEY VAULT:

The azure key vault manages access keys, connection strings, and passwords that are use within the data pipeline. It stores the created credentials in one centralized location not hard coded into the pipeline or scripts. The key vault reduces the potential for secrets to accidentally leak. By not having these keys as part of the code and more widely distributed this creates a safer environment for the pipeline and the data. Logs of when keys are accessed or used can be sent to the Azure Monitor to keep an eye on what is being utilized.

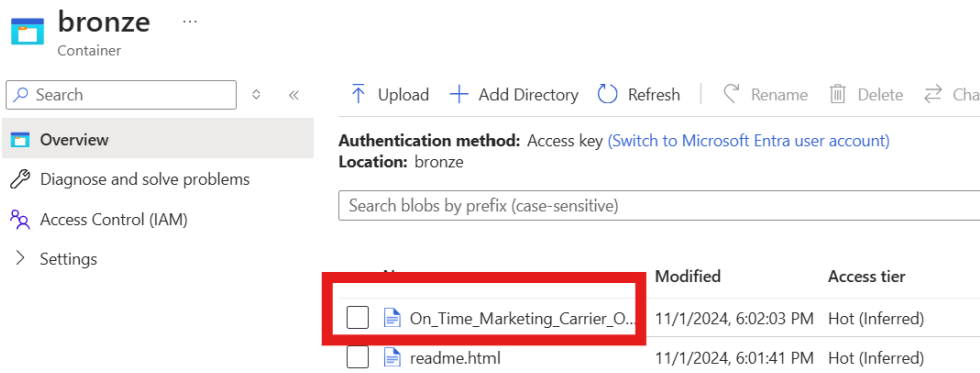
RESULTS

Many of the results are shown naturally above in the documentation of the process and the resulting output of the documentation. However, here I will recap some of the results and review other results.

Datalake: The environment was built to be able to store a large amount of data. This was and can be done on the datalake in an unstructured format. The medallion model of containers is popular for machine learning. The first container (bronze) for unstructured data, the second (silver) for more structured cleaned data, and the third (gold) for data that is fully prepared for machine learning models. You can see below the medallion containers in the datalake along with a default container and the datalake logs container.

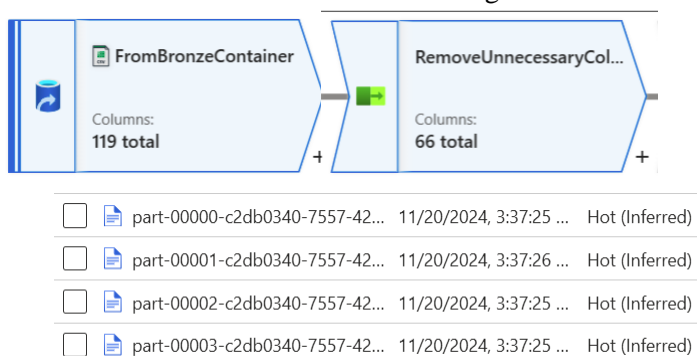
Name	Last modified	Anonymous access level	Lease state
<input type="checkbox"/> \$logs	11/1/2024, 1:19:52 PM	Private	Available
<input type="checkbox"/> bronze	11/1/2024, 6:00:41 PM	Private	Available
<input type="checkbox"/> datalake	11/1/2024, 1:40:03 PM	Private	Available
<input type="checkbox"/> gold	11/1/2024, 6:01:19 PM	Private	Available
<input type="checkbox"/> silver	11/1/2024, 6:01:13 PM	Private	Available

Ingestion: the csv was ingested into the bronze layer. Although, there were several hurdles I eventually simply loaded the csv to the container directly instead of loading through an automated flow in Data Factory. You can see below the csv in the bronze container of the datalake.



Cleaning: The output of cleaning dataflow was a file that had 66 rows instead of the original 119. In this first pass several of the rows had the nulls removed.

I did leave nulls in the rows that had to do with delayed flights as removing nulls from the whole dataset left me with very little data. This portion will have to be done in the silver to gold transformations and cleanings when the dataset is split into categories for machine learning models. Some of the columns were changed from string type to integer or date type depending on whether that data was string, integer, or date. The data was split into four csv files for faster processing in the spark cluster. This was done by selecting a button in the dataflow processing to either have use current partitioning, single partitioning, or set partitioning. I used current partitioning for processing in the spark cluster.



The Results from the silver to gold transformations removed all the remaining nulls assuring, data types were correct and selected only columns that would be needed for the models to predict. While there could be other transformations like normalization or standardization and potentially one-hot-encoding the transformations completed are a good steppingstone. You can see below the output data from the 'PredictingFlightDelays' directory in the gold container.

Year	Quarter	Month	DayofMonth	DayOfWeek	Origin	Dest	CRSDepTime	DepDelayMinutes	CRSArrTime	ArrDelayMinutes	CarrierDelay	WeatherDelay	NASDelay	SecurityDelay	LateAircraftDelay
2024	1	1	30	2	PHX	BOS	2225	2	500	19	0	0	17	0	2
2024	1	1	30	2	BOS	PHX	1720	85	2134	17	4	0	0	0	13
2024	1	1	30	2	LGA	FLL	929	30	1236	23	0	0	0	0	23
2024	1	1	30	2	FLL	LGA	1630	0	1929	20	0	0	20	0	0
2024	1	1	30	2	BOS	LGA	1900	124	2029	97	97	0	0	0	0
2024	1	1	30	2	BOS	DCA	700	0	851	24	0	0	24	0	0

You can see the below output data from the 'RoutePerformanceAnalysis' directory in the gold container.

Origin	Dest	DepDelayMinutes	CRSArrTime	ArrDelayMinutes	ActualElapsedTime	AirTime	Distance	CRSDepTime
ROC	JFK	0	1659	0	70	51	264	1538
MCO	HPN	0	2029	0	150	129	972	1744
PHX	BOS	2	500	19	292	278	2300	2225
BOS	PHX	85	2134	17	306	285	2300	1720
LAX	JFK	0	830	0	334	314	2475	2359

DISCUSSION

In this section includes a review of the technologies used, why they were chosen, what considerations were taken, and problems encountered. The discussion of how the technologies were employed is included more naturally in the methodology section of this report.

The main consideration taken into account for this project was to build a pipeline that meets the modern-day data pipeline aspects. This project got as close to this goal as possible with the resources of time, finances, and knowledge available. While this was a pipeline on a budget, it did succeed in proving that there are advantages to using cloud computing for big data processing and management.

Setting up a storage account came with many considerations. There are many options but settled on the datalake because it is widely used in today's modern data pipeline. This is because you can put structured and unstructured data into a datalake in a semi organized fashion because of the hierarchical namespace. The datalake can hold a very large number of pictures, audio, video, and data structured or unstructured, making it very useful for collecting various types of data for machine learning models. It is also an inexpensive option when compared to a data warehouse.

There are many different options to choose from when working with data pipelines in the cloud. When one service didn't completely meet the needs of the pipeline, there were a great variety of options to choose from. For example, when it came to ingesting Data Factory and Synapse Analytics were considered. In only Data Factory there were options to ingest data from many different sources in many ways, each with their own idiosyncrasies.

These options came as a blessing and a curse as the many services provided options to choose from but proved to be very difficult to get the applications to set up the environment correctly so that they each worked seamlessly together. For example, a wrong selection on the creation of the datalake could mean that different credentialing would be needed to access the container storage. This could also be the case when setting up the Data Factory dataflows or datasets.

In this project the main difficulties were configuring the ingestion, which was reviewed in the methodology section. Besides the ingestions there were also difficulties configuring the access of the datalake storage to pull the data for transformations. Databricks notebook had to be discarded due to authorization limitations of datalake access and cluster accessibility. This option seemed at first to be the best option as creating code within notebooks is something I have done often. However, when this option was laid aside there were other options to look at. These included Data Factory, Synapse Analytics, and Azure Functions App; there are other options, but these are the ones that were considered and looked at to provide some easier integration. Ultimately, I decided on Data Factory because of its ease of integration with the datalake and good tutorials, that showed this could be a successful implementation. Another point was that Data Factory also accesses spark clusters to do its data processing and transformations. Which, from our modules, have shown efficiency and power in analyzing large datasets.

The modules in this class assisted in understanding how the data pipeline may benefit from features of various services. For instance, understanding that Data Factory would be using a spark cluster and knowing what a spark cluster is good for was very helpful. Additionally, when choosing a storage option understanding structured and unstructured data, scalability of cloud resources, costs related to storage, and redundancy was very paramount in assessing option. It was useful to understand the lifecycles and pipelines and the monitoring in Data Factory, the datalake, and azure services as a whole the Azure provides. While this was only a piece of a whole puzzle that would be a pipeline it was good to think

about the pipeline as whole and how data quality, backup and security, and documentation could be more fully integrated on the pipeline.

CONCLUSION

The overall benefit of this project was the exploration showed how cloud services can be used to ingest and process data at a large scale. This project communicated some of the benefits of using different applications within azure to provide easy access, processing power using spark, scalability of applications, wide variety resources (applications), cost saving potentials, the ability to increase capacity at peak times, and the reliability of services given geo redundancy. This was the first step in developing a fully functioning data pipeline that can handle big data use cases. I personally learned about some of the difficulties in implementing these processes, but more importantly it brought into focus much of the information learned through this course. The project demonstrates how Azure services can simplify data processing and storage while preparing datasets for machine learning at scale.

REFERENCES

1. <https://code.benco.io/icon-collection/azure-icons/>
2. <https://learn.microsoft.com/en-us/azure/data-factory/introduction>
3. https://miro.medium.com/v2/resize:fit:800/1*qn2gItW3lIvbDtrKlQgtww.jpeg
4. https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGK&QO_fu146_anzr=b0-gvzr
5. Data Factory: https://www.youtube.com/watch?v=Oggcql1euGJs&t=780s&ab_channel=BitsAnalytics
6. Data Factory: https://www.youtube.com/watch?v=ZILTV-BSAIE&t=2s&ab_channel=SkillCurb
7. Data Factory: https://www.youtube.com/watch?v=Oggcql1euGJs&t=780s&ab_channel=BitsAnalytics
8. Data Factory: https://www.youtube.com/watch?v=Gt3XQ5ZxiU&t=7s&ab_channel=AnalyticswithNags
9. Data Factory Transformations: https://www.youtube.com/watch?v=75NTbT2QMaw&ab_channel=DataCafe
10. Data Factory Filter, sort, sink Transformation: <https://www.youtube.com/playlist?list=PLcwrIWK7WBcTnbJludJBBHIKDvZaK3aQ>
11. Resources and Pipeline: https://www.youtube.com/watch?v=ygJ11fzq_ik&ab_channel=Luke-CloudConsultant
12. Azure Data Factory: https://youtube.com/watch?v=U5uYOM6j5II&ab_channel=LearnITEveryDay-AzureETLSolutionmadeEasy
13. Mounting Azure Data Lake: https://www.youtube.com/watch?v=uIueSLiQ8_o&ab_channel=SkillsPragati
14. Microsoft Documentation: <https://learn.microsoft.com/en-us/azure/?product=popular>