



Data Engineering Exercise

AZURE

Document Version: [V2]

Date: [05/24/2021]

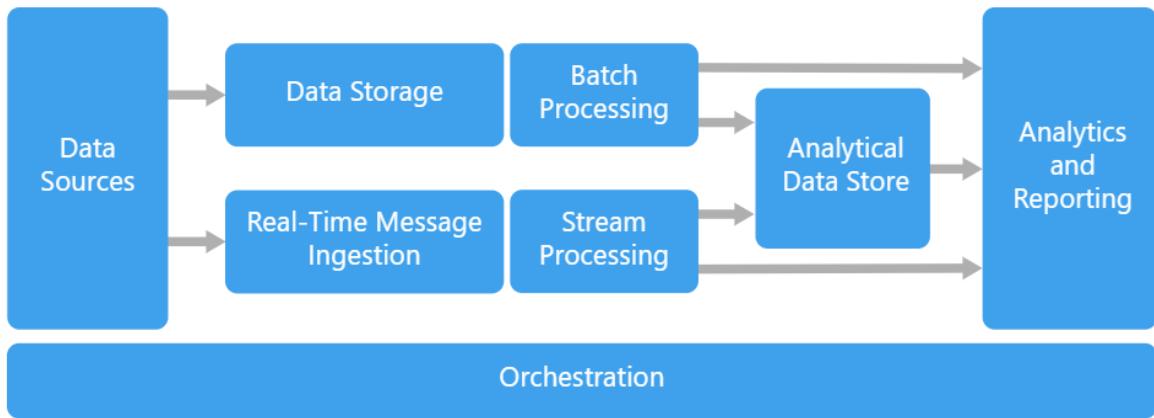
Contents

1. BIG DATA ARCHITECTURE STYLE.....	3
1.1. Technical architecture for Big Data	4
1.2. Common Data Flow for Big Data Project.....	4
2. DATA ENGINEERING EXERCISE.....	6
2.1. Exercise 1	6
2.2. Exercise 2	7
2.3. Exercise 3	19
2.4. Exercise 4	21
2.5. Exercise 5	25

1. Big data architecture style

This section describes the common Big data architecture pattern for big data on public cloud (Azure).

A big data architecture is designed to handle the ingestion, processing, and analysis of data that is too large or complex for traditional database systems



Most of Big data solutions typically involve one or more of the following types of workload:

- Batch processing of big data sources at rest.
- Real-time processing of big data in motion.
- Interactive exploration of big data.
- Predictive analytics and machine learning.

Use Case -

- Store and process data in volumes too large for a traditional database.
- Transform unstructured data for analysis and reporting.
- Capture, process, and analyze unbounded streams of data in real time, or with low latency.
- Use Azure Machine Learning or Microsoft Cognitive Services.

Benefits -

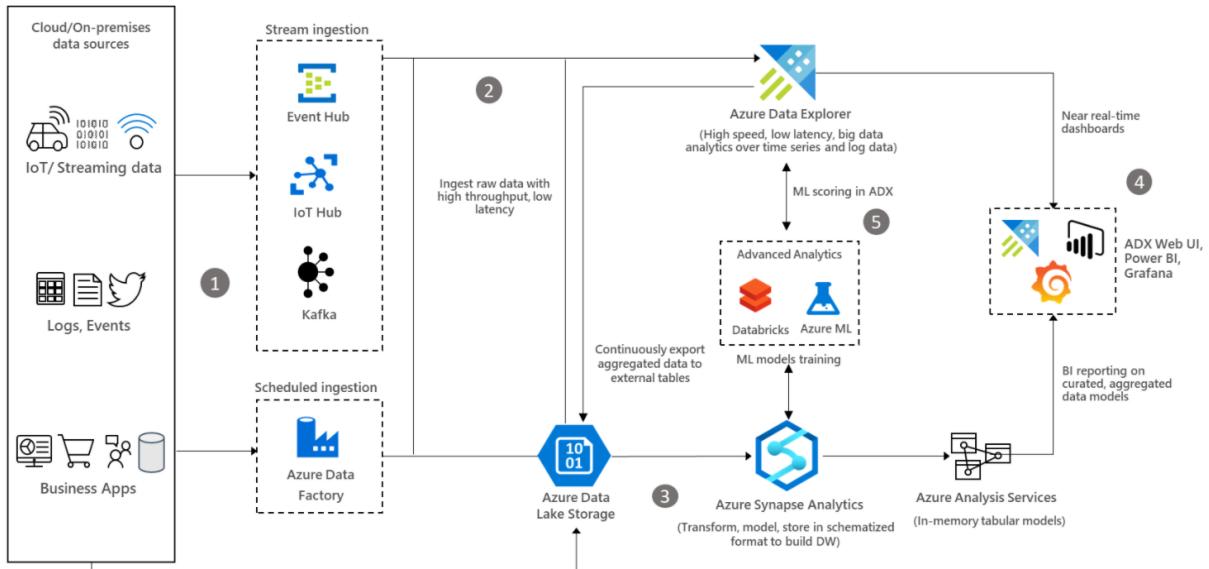
Technology choices- You can mix and match Azure managed services and Apache technologies in HDInsight clusters to capitalize on existing skills or technology investments.

Performance through parallelism- Big data solutions take advantage of parallelism enabling high-performance solutions that scale to large volumes of data.

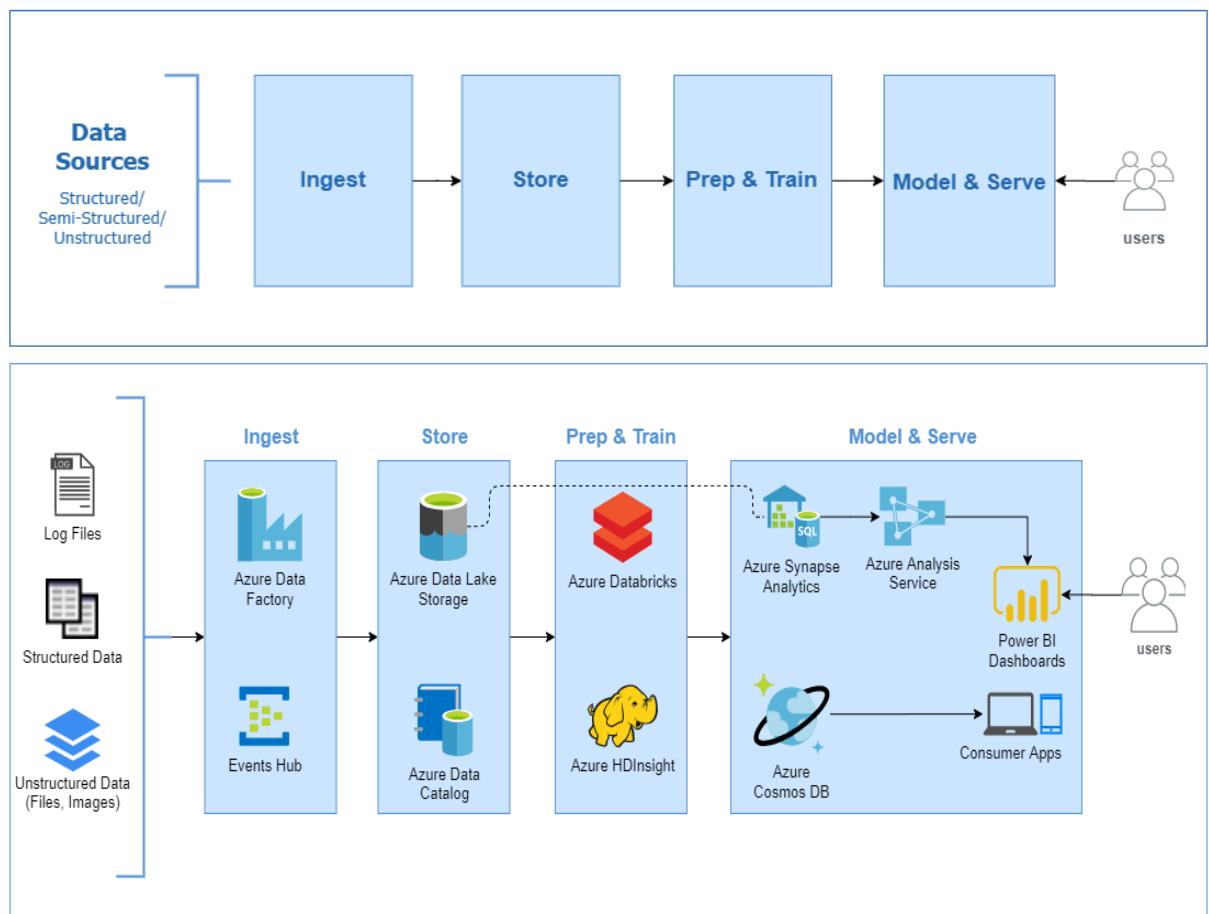
Elastic scale- All the components in the big data architecture support scale-out provisioning so that you can adjust your solution to small or large workloads and pay only for the resources that you use.

Interoperability with existing solutions- The components of the big data architecture are also used for IoT processing and enterprise BI solutions, enabling you to create an integrated solution across data workloads.

1.1. Technical architecture for Big Data



1.2. Common Data Flow for Big Data Project



Ingest- The data is ingested using Azure Data Factory. The Azure Data Factory is a cloud based ETL tool for data integration and transformation. For streaming data ingestion, Azure

Events Hub service is used to build real-time data pipelines. It can stream millions of events per second from any source and seamlessly integrate with other Azure data services.

Store- For storage, Azure Data Lake Storage is the ideal candidate to provide a secure and massively scalable data store for all kinds of data sets with capabilities dedicated to big data analytics. Whereas the Data Catalog service is used to create a metadata catalog which stores information about business, technical and operational metadata and helps simplifying data asset discovery.

Prep & Train- For data preparation and training, both Azure HDInsight and Azure Databricks provide capabilities to not only clean, transform and enrich unstructured datasets but also apply machine learning techniques for building and training models and get deeper data insights. The users can create in-built notebooks with support of programming languages i.e. Scala, Python or R.

Model & Serve- The results of data processing are then stored in the data warehouse and analytics applications such as Azure Synapse Analytics to be used by Power BI reports and dashboards generated by the user. While the insights gained from machine-learning models are moved to Azure Cosmos DB for access through consumer applications.

2. Data Engineering Exercise

This section describes the DE exercise for Data solution on Azure public cloud.

2.1. Exercise 1

Write a python script to simulate the following three Tables:

Table 1: High frequency timeseries accelerometer (3-axis) and gyroscope (3-axis) data (50 Hz, duration of 5 min) for 200 subjects

Table 2: Demographic information for 200 subjects (Any 5 demographic variables)

Table 3: Programming information including program name, amplitude, frequency and pulse width for 200 subjects

Use random variable generators to create above tables and make sure the range of values are reasonable.

Type – Sample source data generation

Solution – Approach

Step 1 – Create Python virtual environment

Step 2 – Install required libs (Planning to use Faker and pandas-profiling). *Faker* is a Python package that generates fake data for any data project.

Step 3 – Build a python script to generate 200 Unique subject Data (Demographic and multiple Programs for Subjects).

Step 4 – Use previously generated 200 random subjects to generate timeseries : High frequency data.

Step 5 – Generate profile result for all three data sets

Step 6 – Save sample data sets in .csv format along with profile results(.HTML format)

Code Location

GitHub Link - https://github.com/vishu22314558/ABT_DE_Exercise_V2-main.git

Code repo Name - [ABT_DE_Exercise_V2-main](#)

Script to generate Sample Data -

- [ABT_DE_Exercise_V2-main/PythonCode/Exercise-1.py](#)

Script for generating Source data profile (jupyter notebook) -

[ABT_DE_Exercise_V2-main/PythonCode/Source_Data_Profile.ipynb](#)

Generated Sample data - [ABT_DE_Exercise_V2-main/Sample_Data/](#)

Data Set Name - [Subject_Program.csv](#) , [Subject_Demographic.csv](#) and

[Subject_Accelerometer.csv](#)

HTML Profile result - [ABT_DE_Exercise_V2-main/Profile_Result_Source_Data/](#)

Profile Name - [df_Subject_Program.html](#) , [df_Subject_Demographic.html](#) and

[df_Subject_Accelerometer.html](#)

2.2. Exercise 2

Write the three Tables (from step 1) to Azure SQL database (Secured with firewall)

Type – Data Ignition and Storage

Solution – Approach

We have three sample data files, So I will be trying two or three different approaches to load data into SQL Server.

Infrastructure Setup –

Create resource group “ABT-Exercise-rg” using PowerShell

Create AZURE SQL Serve instance (**abtsqlserver1**) and create database(**abtsample**) using PowerShell or UI and block all access except trusted IP.

Databases in SQL Database are protected by firewalls in Azure. By default, all connections to the server and database are rejected.

Other Security Features

Transparent data encryption - The encryption feature automatically encrypts your data at rest and requires no changes to applications accessing the encrypted database. It is on by default.

Dynamic data masking - The data masking feature will automatically hide sensitive data in your database.

Auditing - The auditing feature tracks database events and writes events to an audit log in either Azure storage, Azure Monitor logs, or to an event hub. Auditing helps maintain regulatory compliance, understand database activity, and gain insight into discrepancies and anomalies that could indicate potential security violations.

Transport Layer Security (Encryption-in-transit) - SQL Database, SQL Managed Instance, enforce encryption (SSL/TLS) at all times for all connections. This ensures all data is encrypted "in transit" between the client and server irrespective of the setting of **Encrypt** or **TrustServerCertificate** in the connection string.

Sever Setup -

Azure SQL Sever - instance (abtsqlserver1) and database(abtsample)

Showing 1 to 10 of 10 records. <input type="checkbox"/> Show hidden types		
<input type="checkbox"/> Name ↑↓	Type ↑↓	Location ↑↓
<input type="checkbox"/> abtbloblink	Private endpoint	Central US
<input type="checkbox"/> abtbloblink.nic.57efed35-a128-4acd-a351-7d2d5ef373b8	Network interface	Central US
<input type="checkbox"/> abtKeyVault	Key vault	Central US
<input type="checkbox"/> abtlink	Private endpoint	Central US
<input type="checkbox"/> abtlink.nic.aac9f67a-2848-4f48-b04c-196975af8cf0	Network interface	Central US
<input type="checkbox"/> abtsample (abtsqlserver1/abtsample)	SQL database	Central US
<input type="checkbox"/> abtsg	Storage account	Central US
<input type="checkbox"/> abtsqlserver1	SQL server	Central US
<input type="checkbox"/> abtVNet	Virtual network	Central US
<input type="checkbox"/> privatelink.blob.core.windows.net	Private DNS zone	Global

Database -

The screenshot shows the Azure portal interface for the 'abtsample' database. The top navigation bar includes 'Home > ABT-Exercise-rg > abtsqlserver1 >'. The main title is 'abtsample (abtsqlserver1/abtsample)'. Below the title, it says 'SQL database'. The left sidebar has links for Overview, Activity log, Tags, Diagnose and solve problems, Quick start, and Query editor (preview). The right pane displays database properties:

Resource group (change)	: ABT-Exercise-rg
Status	: Online
Location	: Central US
Subscription (change)	: Azure pay as you go
Subscription ID	: c81bdfb0-4263-4ec8-bb74-de870247fe28
Tags (change)	: Click here to add tags
Server name	: abtsqlserver1.database.windows.net
Elastic pool	: No elastic pool
Connection strings	: Show database connection strings
Pricing tier	: Basic
Earliest restore point	: No restore point available

Network Connectivity -
Firewall - No Public Access
Connected to Vent - abtVNet
Allowed Trusted IP

The screenshot shows the Azure portal interface for the 'abtsqlserver1' SQL server. The top navigation bar includes 'Home > ABT-Exercise-rg > abtsample (abtsqlserver1/abtsample) > abtsqlserver1'. The main title is 'abtsqlserver1 | Firewalls and virtual networks'. The left sidebar has links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Settings (Active Directory admin, SQL databases, SQL elastic pools, DTU quota, Properties, Locks), Data management (Backups, Deleted databases, Failover groups), and Firewall rules. The right pane displays firewall settings and a list of virtual network rules:

Deny public network access	Yes						
Click here to create a new private endpoint.	Create Private Endpoint						
Minimum TLS Version	1.0 1.1 1.2						
Connection Policy	Default Proxy Redirect						
Allow Azure services and resources to access this server	Yes						
Client IP address	72.183.53.89						
Rule name	Start IP	End IP					
Clientip-2021-5-22_10-29-13	72.183.53.89	72.183.53.89					
Virtual networks	+ Add existing virtual network + Create new virtual network						
Rule name	Virtual network	Subnet	Address Range	Endpoint status	Resource group	Subscription	State
newVnetRule1	abtVNet	app	10.0.0.0/24	Enabled	ABT-Exercise-rg	Azure pay as you go	Read

Private Link between Blob and SQL Server

The screenshot shows the Azure portal interface for a list of private link resources. The top navigation bar includes 'Filter for any field... Type == all Location == all Add filter'. The main area shows a table with columns: Name, Type, and Location. There are 10 records listed:

Name	Type	Location
abtbloblink	Private endpoint	Central US
abtbloblink.nic.57efed35-a128-4acd-a351-7d2d5ef373b8	Network interface	Central US
abtKeyVault	Key vault	Central US
abtlink	Private endpoint	Central US
abtlink.nic.aac9f67a-2848-4f48-b04c-196975af8cf0	Network interface	Central US
abtsample (abtsqlserver1/abtsample)	SQL database	Central US

Transparent data encryption - Encryption at rest

The screenshot shows the Azure portal interface for managing database encryption. The left sidebar has 'Storage' selected. The main area is titled 'Transparent data encryption'. It contains a note: 'Transparent data encryption encrypts your databases, backups, and logs at rest without any changes to your application. To enable encryption, go to each database. [Learn more](#)' and a button 'Service-managed key'. A message box states: 'You've chosen to use a service-managed key. Azure will automatically generate a key to encrypt your databases, and manage key rotations.'

File 1 (Subject_Demographic.csv) -> SQL Severe

- Step 1 – Generate DDL for first table and execute it on SQL Server.
- Step 2 - Build a python script to read .csv file and make connection to sql server and ingest the data.
- Step 3 – login into SQL Server(Database) instance and verify the data in table.

Code Location

GitHub Link - https://github.com/vishu22314558/ABT_DE_Exercise_V2-main.git
Code repo Name - [ABT_DE_Exercise_V2-main](#)

DDL Location - [ABT_DE_Exercise_V2-main](#)/DataBaseObjects/
Script Name - [SubjectDemographic.ddl](#)

Python Script Location - [ABT_DE_Exercise_V2-main](#)/PythonCode/
Script Name - [Exercise-2-File1.py](#)

Note - Used System OS variable to secure the SQL Server credential

```
SQL_SERVER = os.environ.get("SQLSERVER")
logging.info("SQL Server Name :%s" , SQL_SERVER)
DATABASE_NAME = os.environ.get("DATABASE")
logging.info("SQL DATABASE Name :%s" , DATABASE_NAME)

USER_NAME = os.environ.get("SQLUSERNAME")
PASSWORD = os.environ.get("PASSWORD")
```

Note - Another Approach would be to create VM in Vnet and use private link to read data from Blob and ingest into SQL Server

```

1 SELECT [SubjectFirstName]
2      ,[SubjectLastName]
3      ,[SubjectGender]
4      ,[SubjectBirthdate]
5      ,[SubjectId]
6      ,[SubjectCity]
7      ,[SubjectZipcode]
8      ,[SubjectState]
9  FROM [dbo].[SubjectDemographic]

```

SubjectFirstName	SubjectLastName	SubjectGender	SubjectBirthdate	SubjectId	SubjectCity	SubjectZipcode	SubjectState
Kamryn	McDermott	M	1947-06-14	02991182-176d-4a90-b647-6...	Andersonton	83979	Oregon
Hebert	Schumm	F	1994-12-22	0497919c-4f20-419d-b172-c...	West Elizabethmouth	69511	Georgia
Deondre	Denesik	F	1977-07-12	04dc15a1-653e-412b-af58-9...	Michaelberg	24078	Texas
Greg	Von	F	1950-08-20	05cad1e2-20bf-4380-b68f-9...	East Jose	40151	Iowa
Carisa	White	M	1969-11-30	061866e1-165d-415c-9ab1-f...	New Lisa	86935	Colorado
Merlyn	Jerde	M	1999-01-06	06ebc0be-f04c-4dfd-a62e-2...	Hortonchester	61741	Nebraska
Burnice	Cole	F	2001-05-29	0706404b-19ad-4078-bc76-a...	North Kenneth	11957	Missouri
Elia	Gutmann	M	2002-10-18	07c91a9b-1348-486d-ad37-4...	Port Kenneth	13815	Kentucky
Curtis	VonRueden	F	1970-05-23	098dcc15-d5da-47e2-ab58-9...	East David	15250	Arizona
Conard	Hansen	M	1962-04-02	0a76a765-821c-4cd-8cf5-4...	South Meganville	85537	Idaho
Ladonna	Johnston	NULL	2003-12-11	0aa4b0cb-8668-49c0-bda1-a...	West Robinstad	18774	Vermont
Destany	Jenkins	M	1968-03-15	0ab0cffc-9318-4b74-9bae-e...	Elizabethtown	70168	Connecticut
Melissa	Altenwerth	F	1960-02-06	0ad37f77-4a55-44eb-a05c-3...	Hodgesstad	88701	Tennessee
Williams	Schultz	F	1975-08-28	0bbe3568-7aa2-4fe7-b8a7-9...	Millerhaven	26306	South Dakota
Lydia	Hagenes	M	1960-10-22	0cd5a4b-fc3e-4eba-a436-9...	West Jennifer	68161	Alabama
Tollie	Herman	M	2001-11-13	0cedefaf-5561-442e-8ac6-1...	East Timothyview	74592	New Jersey
Shelva	Bahringer	M	2003-05-03	117897a2-edcf-4ad6-8619-b...	Perezberg	68292	Virginia

```

1 SELECT count(*)
2  | FROM [dbo].[SubjectDemographic]

```

(No column name)
1 200

File 2 (Subject_Accelerometer.csv) -> Blob -> Azure data factory ->SQL Server

Step 1 – Create Blob or ADLS gen2 storage account using power shell in ABT-Exercise-rg resource group

Step 2 – Use Apache Nifi to ingest data into Blob storage using Shared access signature or access keys.

Step 3 – Create Azure data Factory flow to load data into SQL Server. This flow will create the target table as part of data pipeline.

Step 4- login into SQL Server instance and verify the data in table.

Note – This data ingestion flow can be scheduled to run on a particular type of event or on schedule basis. We can also use ADLS Gen2 storage type to store the data if data volume is huge.

ADLS Gen2 (Azure Data Lake Storage) - Optimized storage for big data analytics workloads and can be used for Batch, interactive, streaming analytics and machine learning data such as log files, IoT data, click streams, large datasets.

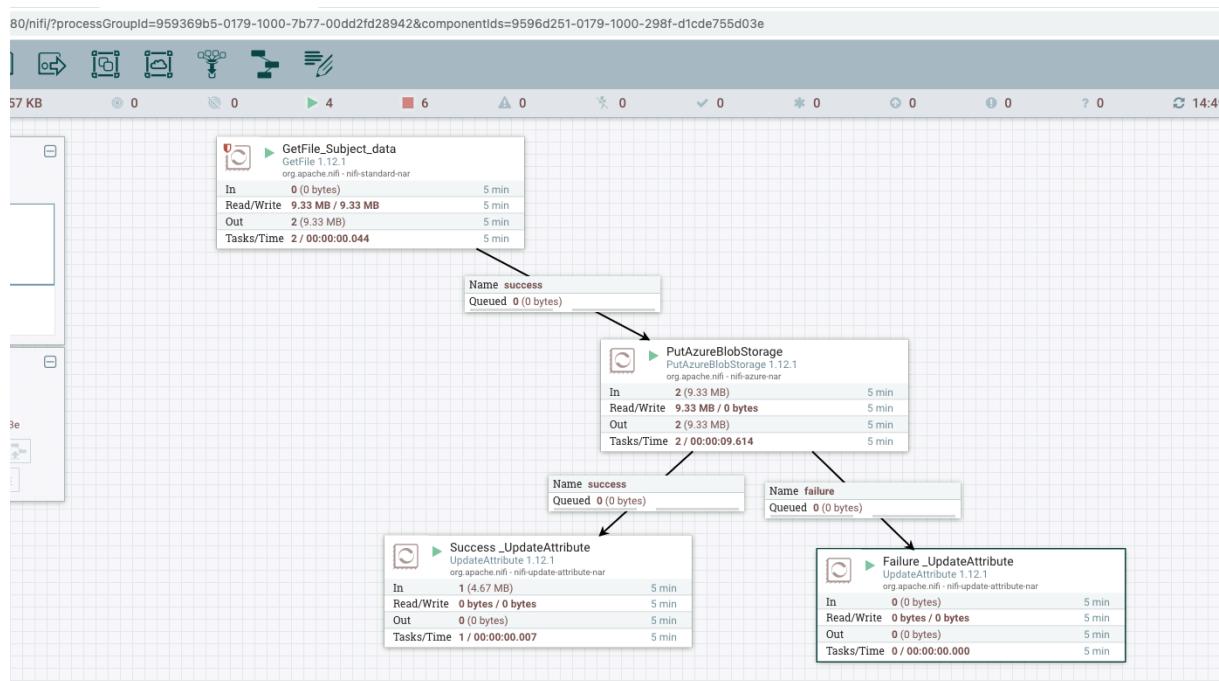
Code Location

GitHub Link - https://github.com/vishu22314558/ABT_DE_Exercise_V2-main.git
 Code repo Name - [ABT_DE_Exercise_V2-main](#)

Data Factory Code - [ABT_DE_Exercise_V2-main/Azure_Code/CopyPipeline_DF/](#)

Apache NiFi supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic.

Apache NiFi flow to ingest data files into Blob storage
 PutAzureBlobStorage - utilizes SAS key to authenticate



The screenshot shows the Azure Storage Explorer interface for a container named 'abtsample'. The left sidebar includes links for Home, abtsg, Overview, Diagnose and solve problems, Access Control (IAM), Settings, Shared access signature, Access policy, and Donations.

The main pane displays blob storage details for the 'abtsample' container. It shows two blobs:
 - Subject_Accelerometer.csv: Modified 5/22/2021, 2:45:54 PM, Access tier Hot (Inferred), Blob type Block blob, Size 4.67 MiB, Lease state Available.
 - Subject_Program.csv: Modified 5/22/2021, 2:11:22 PM, Access tier Hot (Inferred), Blob type Block blob, Size 55.15 KiB, Lease state Available.

At the top, there are buttons for Upload, Change access level, Refresh, Delete, Change tier, Acquire lease, Break lease, View snapshots, Create snapshot, and a search bar for blobs by prefix.

Create Copy data pipeline in DF

Target Table creation is part of Pipeline. This pipe line can be event/trigger driven or schedule basis.

Azure Data Factory can help us to complex spark ELT pipeline using transient **Databricks** or **Hdinsight** cluster.

The screenshot shows the Azure Resource Group 'ABT-Exercise-rg'. The 'Essentials' section displays a single resource: 'Subscription (change) : Azure pay as you go' with ID 'c81bdff0-4263-4ec8-bb74-de870247fe28'. The 'Tags' section shows 'Tags (change) : Click here to add tags'. The 'Data factory (V2)' section lists one item: 'abtDF' under 'Data factory (V2)' with 'Central US' location. Other sections like 'Deployments', 'Security', and 'Policies' are also visible.

The screenshot shows the 'Copy Data tool' interface. On the left, a vertical navigation pane lists steps: Properties, Source, Destination, Settings, Review and finish, Review, and Deployment. The main area shows a flow from 'Azure Blob Storage' to 'Azure SQL Database'. Below the flow, a message says 'Deployment complete'. A table details the deployment steps: 'Creating datasets' (Succeeded), 'Creating pipelines' (Succeeded), and 'Running pipelines' (Succeeded). A note at the bottom states: 'Datasets and pipelines have been created. You can now monitor and edit the copy pipelines or click finish to close Copy Data Tool.'

The screenshot shows the 'Pipeline runs' page in the Microsoft Azure Data Factory interface. The left sidebar shows 'Runs', 'Pipeline runs', 'Trigger runs', 'Runtimes & sessions', and 'Integration runtimes'. The main area displays a table of pipeline runs. One run is listed: 'CopyPipeline_ABTSQ' with a run start of '5/22/21, 2:57:29 PM', a run end of '5/22/21, 2:58:11 PM', a duration of '00:00:41', triggered by 'Manual trigger', status 'Succeeded', and type 'Original'.

The screenshot shows the Azure Data Factory pipeline editor. On the left, there's a tree view of datasets: 'SourceDataset_kok' (1 item), 'ABTSQL' (4 items), and 'review' (0 items). The main area displays a 'CopyPipeline_ABTS...' pipeline with a single 'Copy' activity named 'Copy_iel'. The activity configuration pane shows the following settings:

- General**: Name is 'Copy_iel', Description is empty, Timeout is '7:00:00:00', Retries is '0'.
- Source**: Not defined.
- Sink**: Not defined.
- Mapping**: Not defined.
- Settings**: Not defined.
- User properties**: Not defined.

Security - Pipeline in encrypted End to End - Default

All Source Connection using Azure **Service Principle** to grant access - Read only

We can also use Azure Key vault to secure the credentials.

The screenshot shows the 'Enterprise applications | All applications' page in Azure Active Directory. The left sidebar includes 'Overview', 'Diagnose and solve problems', 'Manage' (with 'All applications' selected), 'User settings', 'Security', 'Conditional Access', and 'Consent and permissions'. The main area lists registered applications:

Name	Homepage URL	Object ID	Application ID
abtblob		39e31c43-47d2-42e4-940d-4a52a6e7cd5b	9e2089c3-1823-4bd1-9146-f6e906ff62ed
Azure DevOps	http://azure.com/devops	7145d142-c0e5-4e94-b8bf-d6a136f00040	499b84ac-1321-427f-aa17-267ca6975798
AzureDataB		889a77e8-54b4-4806-bb65-81ca6f391b35	67a1c6b8-b393-4661-9f3c-e7eb73e98891
DFStorage		a5a42b73-bb18-4396-8bc7-f89958e10f8c	1db419f8-d9b0-4987-9af0-dfb06dba1b28

The screenshot shows the 'Storage Blob Data Reader' configuration page. It lists two entries:

App	Storage Blob Data Reader	This resource	Add
abtblob	Storage Blob Data Reader ⓘ	This resource	Add
DFStorage	Storage Blob Data Reader ⓘ	This resource	Add

Target Table -

CONNECTIONS

- SERVERS**
 - sourcedbdk.database.windows.net, <default> (admin1...)
 - abtsqlserver1.database.windows.net, <default> (abtad...)
- Databases**
 - System Databases
 - abtsample**
 - Tables
 - dbo.BuildVersion
 - dbo.ErrorLog
 - dbo.SubjectAccelerometer
 - dbo.SubjectDemographic
 - dbo.SubjectProgram
 - SalesLT.Address
 - SalesLT.Customer
 - SalesLT.CustomerAddress
 - SalesLT.Product
 - SalesLT.ProductCategory
 - SalesLT.ProductDescription
 - SalesLT.ProductModel
 - SalesLT.ProductModelProductDescription
 - SalesLT.SalesOrderDetail
 - SalesLT.SalesOrderHeader
 - Views
 - Synonyms
 - Programmability
 - External Resources

Welcome SQLQuery_1 - abtsql...min123) ×

Run Cancel Disconnect Change Connection abtsample ▾ |

```

1  SELECT TOP (1000) [SubjectId]
2    , [ProgramName]
3    , [HFTime]
4    , [Gyroscope]
5   FROM [dbo].[SubjectAccelerometer]

```

Results Messages

	SubjectId	ProgramName	HFTime	Gyroscope
1	719edd16-226c-44a8-9e5f-2...	Amplatz PFO Occluder	1621664116.687975	775
2	719edd16-226c-44a8-9e5f-2...	Quadra Allure MP CRT-P	1621664116.688126	673
3	719edd16-226c-44a8-9e5f-2...	Endurity Pacemaker	1621664116.688168	714
4	e9001419-63e8-4aab-8fca-9...	Amplatz PFO Occluder	1621664116.688199	759
5	e9001419-63e8-4aab-8fca-9...	Quadra Allure MP CRT-P	1621664116.688227	920
6	e9001419-63e8-4aab-8fca-9...	Endurity Pacemaker	1621664116.688253	956
7	8b4ef897-87b9-44bb-a793-b...	Amplatz PFO Occluder	1621664116.688279	872
8	8b4ef897-87b9-44bb-a793-b...	Quadra Allure MP CRT-P	1621664116.688305	956
9	8b4ef897-87b9-44bb-a793-b...	Endurity Pacemaker	1621664116.688331	830
10	4e4ebc3a-d127-4898-baa3-a...	Amplatz PFO Occluder	1621664116.688357	688
11	4e4ebc3a-d127-4898-baa3-a...	Quadra Allure MP CRT-P	1621664116.688385	888
12	4e4ebc3a-d127-4898-baa3-a...	Endurity Pacemaker	1621664116.68841	849

CONNECTIONS

- SERVERS**
 - sourcedbdk.database.windows.net, <default> (admin1...)
 - abtsqlserver1.database.windows.net, <default> (abtad...)
- Databases**
 - System Databases
 - abtsample**
 - Tables
 - dbo.BuildVersion
 - dbo.ErrorLog
 - dbo.SubjectAccelerometer
 - dbo.SubjectDemographic
 - dbo.SubjectProgram
 - SalesLT.Address
 - SalesLT.Customer
 - SalesLT.CustomerAddress
 - SalesLT.Product
 - SalesLT.ProductCategory
 - SalesLT.ProductDescription

Welcome SQLQuery_1 - abtsql...min123) ●

Run Cancel Disconnect Change Connection abtsample ▾ |

```

1  SELECT count(*)
2   | FROM [dbo].[SubjectAccelerometer]

```

Results Messages

	(No column name)
1	60000

File 3 (Subject_Program.csv) -> Blob ->Databricks(Pyspark)->SQL Server

Step 1 – Create Blob or ADLS gen2 storage account using power shell in ABT-Exercise-rg resource group. - **Completes in previous task**

Step 2 – Use Apache Nifi to ingest data into Blob storage using Shared access signature or access keys. - **Completes in previous task**

Step 3 – Deploy Databricks in ABT-Exercise-rg resource group.

Step 4 - Create small one node cluster.

Step 4 - Create Python Notebook(abtSubjectProgram)

Step 5 - Build pyspark code to read data from Blob (using azure service principal) and ingest into SQL Sever.

Step 6- login into SQL Server instance and verify the data in table.

Step 7 - Terminate the cluster

Code Location

GitHub Link - https://github.com/vishu22314558/ABT_DE_Exercise_V2-main.git

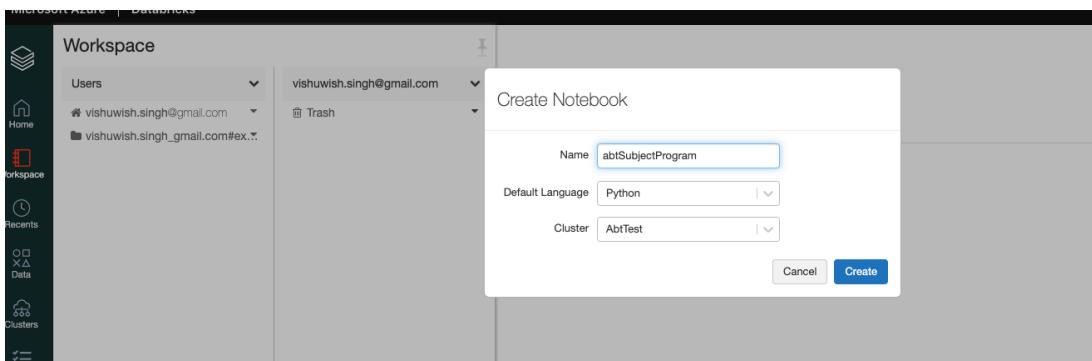
Code repo Name - [ABT_DE_Exercise_V2-main](#)

Spark Code - [Exercise-2-SubjetProgram_Spark.ipynb](#)

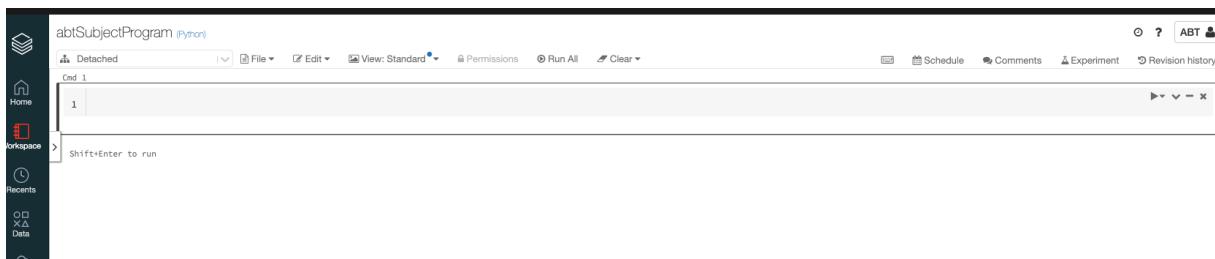
Spark Cluster Setup

The screenshot shows the 'Create Cluster' page in the Microsoft Azure Databricks interface. The left sidebar has icons for Home, Workspace, Recents, Clusters (selected), Jobs, Models, and Search. The main area is titled 'New Cluster' with a 'Create Cluster' button. It includes fields for 'Cluster Name' (AbtTest), 'Cluster Mode' (Single Node), 'Pool' (None), 'Databricks Runtime Version' (Runtime: 8.2 (Scala 2.12, Spark 3.1.1)), and 'Autopilot Options' (Terminate after 10 minutes of inactivity). A note states: 'Note: Databricks Runtime 8.x uses Delta Lake as the default table format.' The 'Node Type' is set to Standard_F4s.

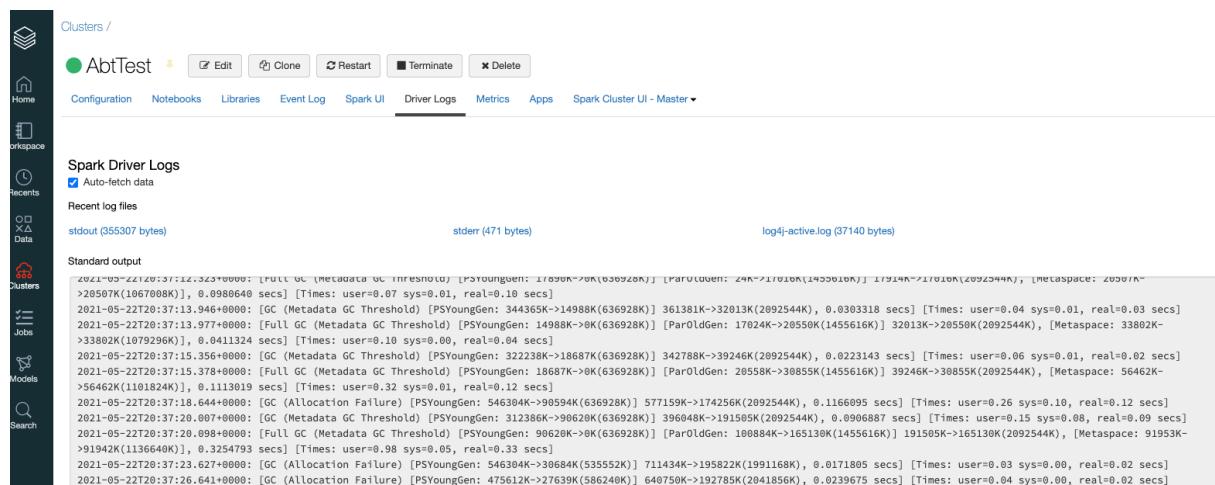
Spark Notebook Setup



Notebook -



Cluster Driver Log -



Resource Group -

Home > ABT-Exercise-rg < ...

ABT-Exercise-rg Resource group

Search (Cmd+ /) < + Add < Edit columns Delete resource group Refresh Export to CSV Open query Feedback Open in mobile Assign tags Move ... Delete ...

Essentials

Subscription (change) : Azure pay as you go

Subscription ID : c81bdfb0-4263-4ec8-bb74-de870247fe28

Tags (change) : Click here to add tags

Deployments : 21 Succeeded

Location : Central US

Filter for any field... Type == all Location == all Add filter

No grouping List view

Type	Location
Azure Databricks Service	Central US
Data factory (V2)	Central US
Key vault	Central US
Private endpoint	Central US
Network interface	Central US
SQL database	Central US

PySpark code -

```

#GDSsubjectProgram.py
# 1. Read csv data from point point
# 2. df = spark.read.format('csv').options(header='true').load("mnt/abt/subject_program.csv")
# 3. df.show()

# (1) Spark Jobs
# df = pySpark.readDataFrame.DataFrame([Subject string, ProgramName string, ... more fields])
# Command took 0.57 seconds -- by vishwakshi.singh@gmail.com at 5/22/2021, 11:13:03 PM on abt
# Cmd 4
# df.show()
# Show cell
# Cmd 5
# df.createOrReplaceTempView("SubjectProgram")
# Command took 0.05 seconds -- by vishwakshi.singh@gmail.com at 5/22/2021, 11:13:14 PM on abt
# Cmd 6
# %sql
# select ProgramName , count(*) from SubjectProgram group by ProgramName
# (2) Spark Jobs
# ProgramName
# 100% 0%
# 0% 100%
# 1. If you want to query this data as a table, you can simply register it as a view or a table.
# 2. df.createOrReplaceTempView("SubjectProgram")
# Command took 1.06 seconds -- by vishwakshi.singh@gmail.com at 5/22/2021, 11:13:14 PM on abt
# Cmd 7
# %sql
# select Pulse , count(*) from SubjectProgram group by Pulse

```

Insert data into SubjectProgram table -



Target Table -

Run Cancel & Disconnect Change Connection abtsample

```

1  SELECT TOP (1000) [AccelerometerId]
2    , [SubjectId]
3    , [ProgramName]
4    , [Amplitude]
5    , [Frequency]
6    , [Pulse]
7  FROM [dbo].[SubjectProgram]

```

Results Messages						
	AccelerometerId	SubjectId	ProgramName	Amplitude	Frequency	Pulse
1	1	719edd16-226c-44a8-9e5f-2...	Amplatzer PFO Occluder	3	58MHz	64
2	2	719edd16-226c-44a8-9e5f-2...	Quadra Allure MP CRT-P	4	44MHz	89
3	3	719edd16-226c-44a8-9e5f-2...	Endurity Pacemaker	7	52MHz	83
4	4	719edd16-226c-44a8-9e5f-2...	Engage TR Introducer	5	51MHz	66
5	5	e9001419-63e8-4aab-8fca-9...	Amplatzer PFO Occluder	4	46MHz	63
6	6	e9001419-63e8-4aab-8fca-9...	Quadra Allure MP CRT-P	3	48MHz	61
7	7	e9001419-63e8-4aab-8fca-9...	Endurity Pacemaker	8	46MHz	88
8	8	e9001419-63e8-4aab-8fca-9...	Engage TR Introducer	4	54MHz	81
9	9	8b4ef897-87b9-44bb-a793-b...	Amplatzer PFO Occluder	8	41MHz	77
10	10	8b4ef897-87b9-44bb-a793-b...	Quadra Allure MP CRT-P	5	52MHz	100
11	11	8b4ef897-87b9-44bb-a793-b...	Endurity Pacemaker	7	42MHz	83
12	12	8b4ef897-87b9-44bb-a793-b...	Engage TR Introducer	3	59MHz	72
13	13	4e4ebc3a-d127-4898-baa3-a...	Amplatzer PFO Occluder	7	56MHz	64
14	14	4e4ebc3a-d127-4898-baa3-a...	Quadra Allure MP CRT-P	3	42MHz	91
15	15	4e4ebc3a-d127-4898-baa3-a...	Endurity Pacemaker	3	47MHz	80
16	16	4e4ebc3a-d127-4898-baa3-a...	Engage TR Introducer	4	44MHz	97
17	17	53551f47-14cd-4c65-b371-6...	Amplatzer PFO Occluder	8	57MHz	64
18	18	53551f47-14cd-4c65-b371-6...	Quadra Allure MP CRT-P	3	46MHz	79

Terminate cluster -

Clusters /

AbtTest Edit Clone Delete

Configuration Notebooks Libraries Event Log Spark UI Driver Logs Metrics Apps Spark Cluster UI - Master

Cluster Mode Single Node

Databricks Runtime Version 8.2 (includes Apache Spark 3.1.1, Scala 2.12)

Autopilot Options Terminate after 10 minutes of inactivity

Node Type Standard_F4s 8.0 GB Memory, 4 Cores, 0.5 DBU

Advanced Options

2.3. Exercise 3

Create an Azure Blob Storage and integrate it with VNet.

Type – Infrastructure and Security

Solution – Approach

Step 1 – Create Blob storage account (abtstg) using power shell in ABT-Exercise-rg resource group.

Step 2 – Create abtVNet VPC using power shell.

Step 3 – Integrate abtstg with abtVNet using service endpoint and/or private endpoint

Note - Service endpoints allow us to secure your critical Azure service resources to only your virtual networks. Traffic from your VNet to the Azure service always remains on the Microsoft Azure backbone network.

Private Endpoint is a network interface that connects you privately and securely to a service powered by Azure Private Link. Private Endpoint uses a private IP address from your VNet, effectively bringing the service into your VNet.

Both appear to allow a direct connection to the services but the key difference here appears to be in the wording. A service endpoint extends your VNet to the Azure service, whereas the private endpoints extend the service into your VNet (using an IP address in your subnet).

Code Location

GitHub Link - https://github.com/vishu22314558/ABT_DE_Exercise_V2-main.git

Code repo Name - [ABT_DE_Exercise_V2-main](#)

PowerShell Code - [.ABT_DE_Exercise_V2-main/Azure_Code/Exercise-3.ps1](#)

Blob Storage - Firewall and Virtual Network - Service Endpoint

Service Endpoint -

Below screenshot shows connectivity between BLOB and VNET using service endpoint and trusted Client IP.

The screenshot shows the Azure portal interface for managing a storage account named 'abtstg' within the 'ABT-Exercise-rg' resource group. The 'Networking' blade is open, specifically the 'Firewalls and virtual networks' section. Under 'Allow access from', the 'Selected networks' option is selected. Below this, there is a table listing a single virtual network entry:

Virtual Network	Subnet	Address range	Endpoint Status	Resource Group	Subscription
abtVNet	1			ABT-Exercise-rg	Azure pay as you go

At the bottom of the blade, there is a 'Firewall' section where an IP address or CIDR range '72.183.53.89' is listed in the 'Address range' field.

Blob Storage Private Endpoint with Vent - (abtlink)

Below screenshot shows connectivity between BLOB and VNET using private endpoint and .

The screenshot shows the 'Private endpoint connections' tab in the Azure Storage account networking settings. A single connection named 'abtlink' is listed, showing it is approved and connected to a private endpoint.

Connection name	Connection state	Private endpoint	Description
abtsg.8fb92074-1ad4-4bda-812f-65...	Approved	abtlink	Auto-Approved

Resource Group - ABT-Exercise-rg

Below screenshot shows resource group - Blob, Vnet , endpoint , Niccard and Private DNS zone for connectivity.

The screenshot shows the 'Overview' tab in the Azure Resource Group 'ABT-Exercise-rg'. It lists several resources including a Key vault, a Private endpoint, a Network interface, a Storage account, a Virtual network, and a Private DNS zone.

Name	Type	Location
abtKeyVault	Key vault	Central US
abtlink	Private endpoint	Central US
abtlink.nic.aac9f67a-2848-4f48-b04c-196975af8cf0	Network interface	Central US
abtsg	Storage account	Central US
abtVNet	Virtual network	Central US
privatelink.blob.core.windows.net	Private DNS zone	Global

2.4. Exercise 4

Write an Event trigger function app (both python and .Net) to parse the Json files (attached example file) from the Blob Storage (step 3) and write it to the Azure SQL database.

Type – Serverless processing

Note - Due to time constrains, I have not created .net function.

Solution – Approach

Step 1 – Analyze JOSN file.

Step 2 – Create 3 normalize tables (main, locdata and hkdata) into SQL Server (**abtsample**)

Step 3 - Ingest JSON data into blob storage account using Nifi

Step 4 – Install all plugins needed for Azure Function development on VS code(local).

Step 5 – Create Function project (Blog Trigger).

Step 6 – Parse JOSN file and insert records in SQL tables (linking key - **guid**)

Step 7 – Login to SQL server and verify tables data

Note – This approach will work only if incoming data set is smaller in size. For big data set we should consider spark engine- **Databricks** or **Hdinsight** for compute requirement.
Another option would be utilizing the **Azure Kubernetes Service (AKS)**.

“Instead on Nifi loading data we can have another HHTP trigger Azure function to upload json data and link with second function”

[HTTP Trigger Function] -> [Blob Trigger Function] -> SQL Server

Code Location

GitHub Link - https://github.com/vishu22314558/ABT_DE_Exercise_V2-main.git

Code repo Name - [ABT_DE_Exercise_V2-main](#)

Py Azure Function Code - [Exercise-4-Azure-Function-Python](#)

Azure Function -

The screenshot shows the Azure portal interface for the resource group 'ABT-Exercise-rg'. The left sidebar lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', and 'Events'. Under 'Settings', there are sections for 'Deployments', 'Security', 'Policies', 'Properties', and 'Locks'. 'Cost Management' includes 'Cost analysis' and 'Cost alerts (preview)'. The main content area displays a table of resources:

Name	Type	Location
abtDF	Data factory (V2)	Central US
abtsample (abtsqlserver1/abtsample)	SQL database	Central US
abtsq	Storage account	Central US
abtsqlserver1	SQL server	Central US
CentralUSLinuxDynamicPlan	App Service plan	Central US
funcABTBlobSQLv1	Function App	Central US
funcabtblobsqlv1insight	Application Insights	Central US

funcABTBlobSQLv1 | Functions

Name: funcABTBlobSQL

Trigger: Blob

Status: Enabled

```

import logging
import azure.functions as func
import json
import pandas as pd
import pymssql

def main(myblob: func.InputStream):
    logging.info(f"Python blob trigger function processed blob \n"
                 f"Name: {myblob.name}\n"
                 f"Blob Size: {myblob.length} bytes")
    json_data = myblob.read()
    sample_json = json.loads(json_data.decode("utf-8"))
    #logging.info(f"This is Updated JSON Data : {sample_json}")
    if sample_json == None or sample_json == '':
        logging.info("empty or blank message received")
    else:

```

Trigger / Execution Log -

Completed	timestamp [UTC]	message	logLevel
5/24/2021, 5:11:01.262 AM	Insert HK Dataframe into SQL Server :abtsqlserver1.database.windows.net	Information	
5/24/2021, 5:11:01.310 AM	HK Dataframe - All record Inserted	Information	
5/24/2021, 5:11:01.310 AM	Insert locdata Dataframe into SQL Server :abtsqlserver1.database.windows.net	Information	
5/24/2021, 5:11:01.338 AM	LOCDATA Dataframe - All record Inserted	Information	
5/24/2021, 5:11:01.338 AM	Insert packetdata into SQL Server :abtsqlserver1.database.windows.net	Information	
5/24/2021, 5:11:01.349 AM	SQL Server Connection Closed	Information	
5/24/2021, 5:11:01.355 AM	Executed 'Functions.funcABTBlobSQL' (Succeeded, Id=f919021a-8f5c-4d6e-aa71-b0e6861f64db, Duration=507ms)	Information	

Target Tables -

CONNECTIONS

- ... database.windows.net
- SQLQuery_1 - abtsql...min123
- SQLQuery_2 - abtsql...min123
- SQLQuery_3 - abtsql...min123

SERVERS

- > sourcedblik.database.windows.net, <default> (admin1...)
- > abtsqlserver1.database.windows.net, <default> (abtad...)
- > abtsample
- > Tables
 - > dbo.BuildVersion
 - > dbo.ErrorLog
 - > dbo.hkdata
 - > dbo.locdata
 - > dbo.packetdata
 - > dbo.SubjectAccelerometer
 - > dbo.SubjectDemographic
 - > dbo.SubjectProgram
 - > SalesLT.Address
 - > SalesLT.Customer
 - > SalesLT.CustomerAddress
 - > SalesLT.Product
 - > SalesLT.ProductCategory
 - > SalesLT.ProductDescription
 - > SalesLT.ProductModel
 - > SalesLT.ProductModelProductDescription
 - > SalesLT.SalesOrderDetail
 - > SalesLT.SalesOrderHeader
 - > Views
 - > Synonyms
 - > Programmability
 - > External Resources
 - > Storage

Run □ Cancel ⌂ Disconnect ⌂ Change Connection abtsample ▾ |

```

1  SELECT TOP (1000) [quantity]
2    , [quantityUnit]
3    , [startTIme]
4    , [endTIme]
5    , [quantityType]
6    , [guid]
7   FROM [dbo].[hkdata]
8
9
10 delete from hkdata

```

Results Messages

	quantity	quantityUnit	startTIme	endTIme	quantityType	guid
1	67	count/min	1618596043, 6129725	1618596043, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
2	1	min	1618595977, 8532996	1618596037, 8532996	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
3	71	count/min	1618596051, 6129725	1618596051, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
4	72	count/min	1618596053, 6129725	1618596053, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
5	82	count/min	1618596058, 6129725	1618596058, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
6	89	count/min	1618596067, 6129725	1618596067, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
7	87	count/min	1618596070, 6129725	1618596070, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
8	85	count/min	1618596073, 6129725	1618596073, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
9	85	count/min	1618596078, 6129725	1618596078, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
10	84	count/min	1618596083, 6129725	1618596083, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
11	81	count/min	1618596088, 6129725	1618596088, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
12	83	count/min	1618596097, 6129725	1618596097, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
13	80	count/min	1618596102, 6129725	1618596102, 6129725	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...
14	1	min	1618596037, 8532996	1618596097, 8532996	HKQuantityTypeIdentifierH...	BC5714C5-B134-40FD-8A86-1...

CONNECTIONS

- ... y_1 - abtsql...min123
- SQLQuery_2 - abtsql...min123
- SQLQuery_3 - abtsql...min123
- SQLQuery_4 - abtsql...min123

SERVERS

- > sourcedblik.database.windows.net, <default> (admin1...)
- > abtsqlserver1.database.windows.net, <default> (abtad...)
- > abtsample
- > Tables
 - > dbo.BuildVersion
 - > dbo.ErrorLog
 - > dbo.hkdata
 - > Columns
 - > Keys
 - > Constraints
 - > Triggers
 - > Indexes
 - > Statistics
 - > dbo.locdata
 - > Columns
 - > Keys
 - > Constraints
 - > Triggers
 - > Indexes
 - > Statistics
 - > dbo.packetdata
 - > dbo.SubjectAccelerometer

Run □ Cancel ⌂ Disconnect ⌂ Change Connection abtsample ▾ |

```

1  SELECT TOP (1000) [latitude]
2    , [longitude]
3    , [time]
4    , [altitude]
5    , [guid]
6   FROM [dbo].[locdata]

```

Results Messages

	latitude	longitude	time	altitude	guid
1	23.04071233826276	-86.82193541693611	1619052598.0062962	311.535945892334	BC5714C5-B134-40FD-8A86-1...
2	23.04146693026753	-86.81978777717225	1619052613.50745	309.0	BC5714C5-B134-40FD-8A86-1...
3	23.04061054579175	-86.82213727635401	1619052613.8562012	308.5008125305176	BC5714C5-B134-40FD-8A86-1...
4	23.040839406053738	-86.82204809869896	1619052630.508529	308.50928497314453	BC5714C5-B134-40FD-8A86-1...
5	23.04011262648462	-86.82193140104995	1619052632.7033482	305.97312927246094	BC5714C5-B134-40FD-8A86-1...
6	23.04039222742726	-86.82212143456536	1619052633.314462	308.50148010253906	BC5714C5-B134-40FD-8A86-1...
7	23.04039222742726	-86.82212143456536	1619052642.314462	308.50148010253906	BC5714C5-B134-40FD-8A86-1...
8	23.04018375838427	-86.82204182011985	1619052644.5876489	305.9652709609375	BC5714C5-B134-40FD-8A86-1...
9	23.0404235820543	-86.82219299807798	1619052644.98462	309.59000396728516	BC5714C5-B134-40FD-8A86-1...

The screenshot shows the SSMS interface with the following details:

- Connections:** y_2 - abtsql...min123, SQLQuery_3 - abtsql...min123, SQLQuery_4 - abtsql...min123, SQLQuery_5 - abtsql...min123.
- Servers:** sourcedblk.database.windows.net, abtsqlserver1.database.windows.net, abtsample.
- Databases:** System Databases, abtsample.
- Tables:** dbo.BuildVersion, dbo.ErrorLog, dbo.hkdata, dbo.locldata, dbo.packetdata.
- Query Editor:** A T-SQL script is pasted into the editor:

```
1 SELECT TOP (1000) [guid]
2 ,[filename]
3 ,[packetnumber]
4 ,[data]
5 FROM [dbo].[packetdata]
```
- Results Grid:** The results of the query are displayed in a table with columns: guid, filename, packetnumber, data. One row is shown:

	guid	filename	packetnumber	data
1	BC5714C5-B134-40FD-8A86-1...	BC5714C5-B134-40FD-8A86-1...	453	[]

2.5. Exercise 5

The attached csv file contains IOT data from several sensors. "SubjectID" column is the unique identifier for each subject.

Write a python script to clean the data (missing and duplicate values), find the most important sensor and model the output variable. Try several techniques to handle the missing data and propose the optimized method.

Type – Machine learning

I will be utilizing Jupiter Notebook to build model

Solution – Approach

Setup Notebook Env (Python) and install python libs like sklearn, seaborn ..

Note – This will be a learning exercise for me and I will be trying RandomForestClassifier to model the output variable.

Step 1 - Define Problem Statement

Step 2 - Analyze Source data

Step 3 - Define data strategy to handle missing values

Step 4- Build RandomForestClassifier model

Step 5 - Implement different data strategy and capture accuracy

Step 6 - Implement Variable importance graph to find most important sensor

Step 7 - Plan for improvements and implementation

Code Location

GitHub Link - https://github.com/vishu22314558/ABT_DE_Exercise_V2-main.git
Code repo Name - [ABT_DE_Exercise_V2-main](#)

NoteBook - [Exercise-5.ipynb](#)

Quick Source Data Analysis -

```
[16]: df = pd.read_csv("IOT_data.csv")
[17]: df.head()
[17]:   Sensor 1  Sensor 2  Sensor 3  Sensor 4  Sensor 5  Sensor 6  Sensor 7  Sensor 8      Time  Output  SubjectID
  0    NaN     NaN     NaN     NaN     NaN    64.0     NaN     NaN  3/13/2021 5:54      2  8812EC1F-22DF-4729-A699-C7E639847E11
  1    NaN     NaN     NaN     NaN     NaN   95.0     NaN     NaN  3/16/2021 13:44      2  8812EC1F-22DF-4729-A699-C7E639847E11
  2    NaN     NaN     NaN     NaN     NaN   73.0     NaN     NaN  3/14/2021 11:29      2  8812EC1F-22DF-4729-A699-C7E639847E11
  3    NaN     NaN     NaN     NaN     NaN   65.0     NaN     NaN  3/14/2021 7:50      2  8812EC1F-22DF-4729-A699-C7E639847E11
  4    NaN     NaN     NaN     NaN     NaN   73.0     NaN     NaN  3/13/2021 1:31      2  8812EC1F-22DF-4729-A699-C7E639847E11
```

Data Description

"SubjectID" column is the unique identifier for each subject. Sensors 1 to 8 represent 8 values from 8 different sensors. Here is some statistics on missing value:

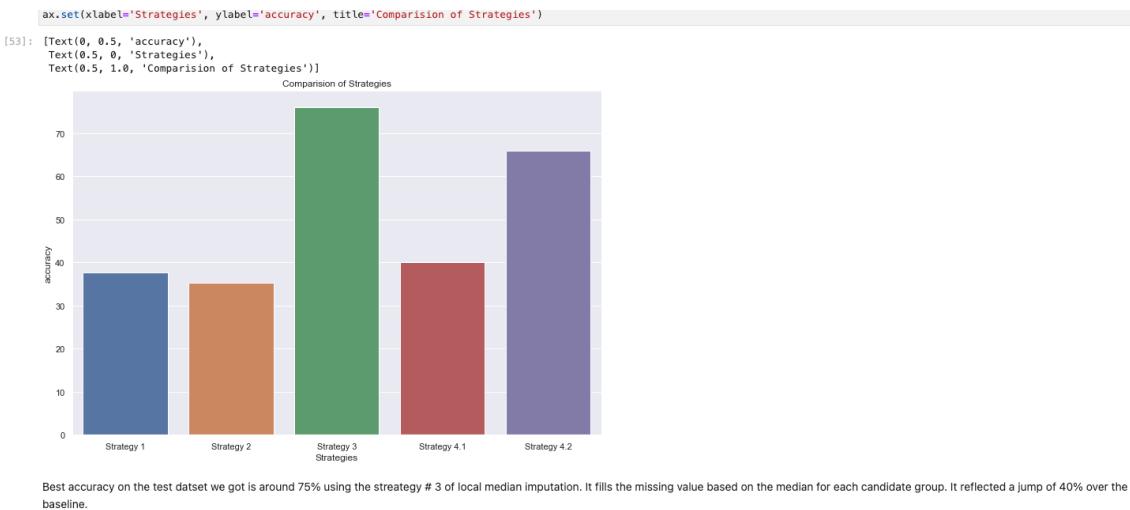
```
[18]: df.isna().mean().round(4)*100
[18]:   Sensor 1    99.97
        Sensor 2    99.53
        Sensor 3    99.10
        Sensor 4   100.00
        Sensor 5    99.99
        Sensor 6    1.44
        Sensor 7    99.97
        Sensor 8    99.10
        Time       0.00
        Output      0.00
        SubjectID   0.00
        dtype: float64
```

Different Strategies -

Strategy (Missing Value)

0. Drop missing values. Not acceptable here since it drops 100% of our data.
1. Replacing them with a distinctive number '-1' in our case.
2. Adding global mean, median with missing value indicator
3. Adding local mean, median as compared to each candidate respectively
4. Finding out missing value based on time series
5. Finding out missing value based on another model

Model accuracy based on data different Strategies -



Most Important Sensor -

```
[51]: eli5.show_weights(perm, feature_names=X_columns)
```

	Weight	Feature
[51]:	0.2818 ± 0.0067	Sensor 7
	0.2412 ± 0.0074	Sensor 5
	0.1536 ± 0.1275	Sensor 3
	0.0597 ± 0.0070	Sensor 6
	0.0559 ± 0.0039	Sensor 1
	0.0294 ± 0.1166	Sensor 2
	0.0004 ± 0.0006	Sensor 8

Sensor 7 and Sensor 5 are the most important

Future improvements and implementation

Chosen strategy however is difficult to implement realtime. Using spark streaming and Kafka, we can keep track of last known value of the sensor and use that value in case of any missing element. We can also keep track of certain statistics based on the time of the day or hour of the day if that is important for that sensor.

1:

