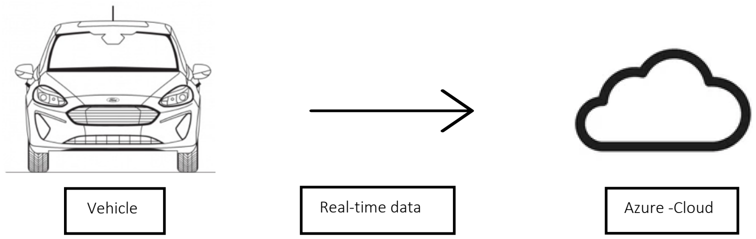
Connected Vehicles Project

Business Case:

Ford is a one of the leading Heavy Vehicle manufacturing companies. To improve their services, they are planning to rollout new features in the coming models. These features are based on IOT solutions, that detect hazardous and pre-malfunction conditions to alert the driver in advance.

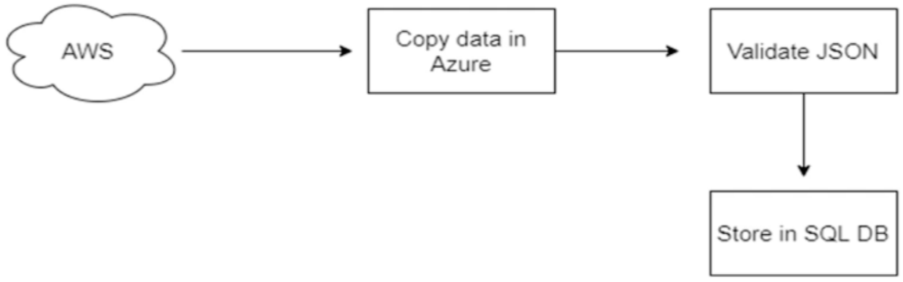
For this to work Ford ties up with a third party that provides them with a device that can be plugged into the car, the **sensors** gather real-time data on *fuel consumption, engine temperature, fluid levels, run time etc.*



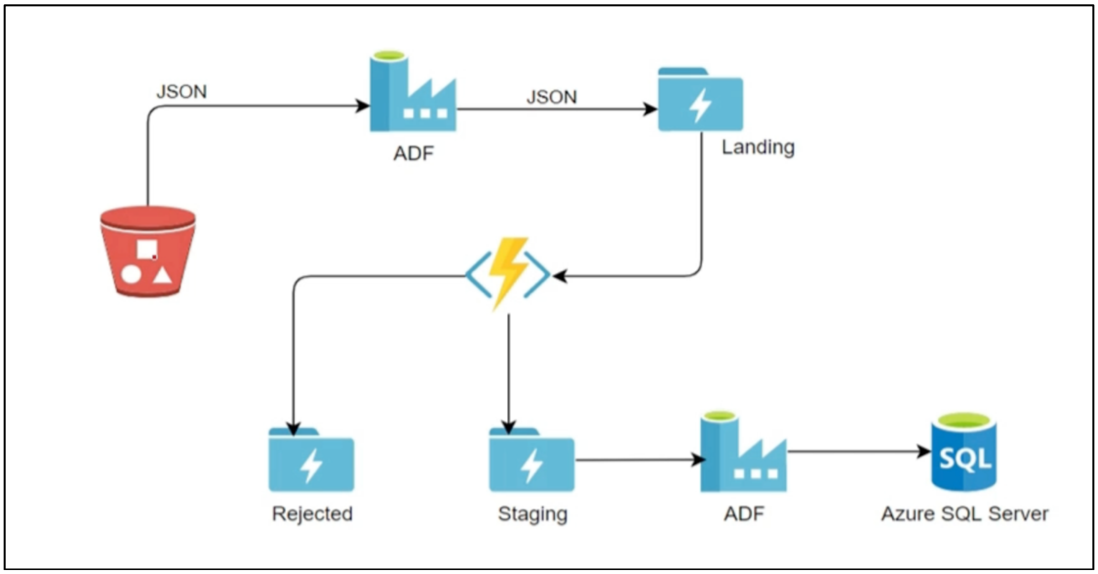
Task:

1. The IOT device sends all the telemetry data (JSON format) into the AWS cloud.
2. The data engineering team needs to migrate that data into the Ford Azure cloud.
3. Also need to validate the files and check for format discrepancies and incomplete files which need rejection.
4. Once the data gets validated it needs to be stored in a SQL data base, which will be utilized by the data science team.

High-level Scope:



Architecture Modelling:

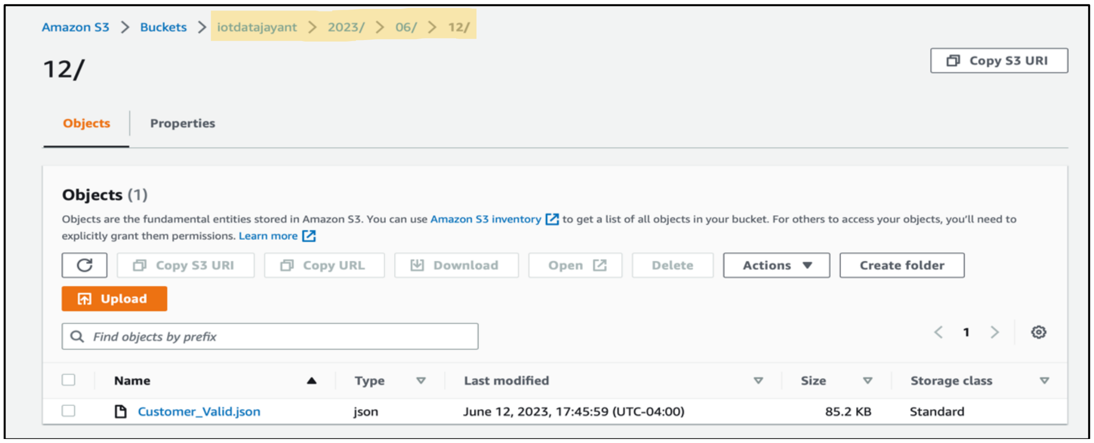


A diagrammatic representation of the workflow

Phase 1 - Setting up basic building blocks

Create S3 bucket:

1. Login to your AWS portal and search for S3, hit create bucket option and give a suitable name.
2. Enable public Access and acknowledge the same on this bucket and hit create.
3. The data will be updated on a **daily basis**, so we will create a folder **hierarchy 'yyyy/mm/dd'**
4. Upload source data file in this path.



Objects (1)

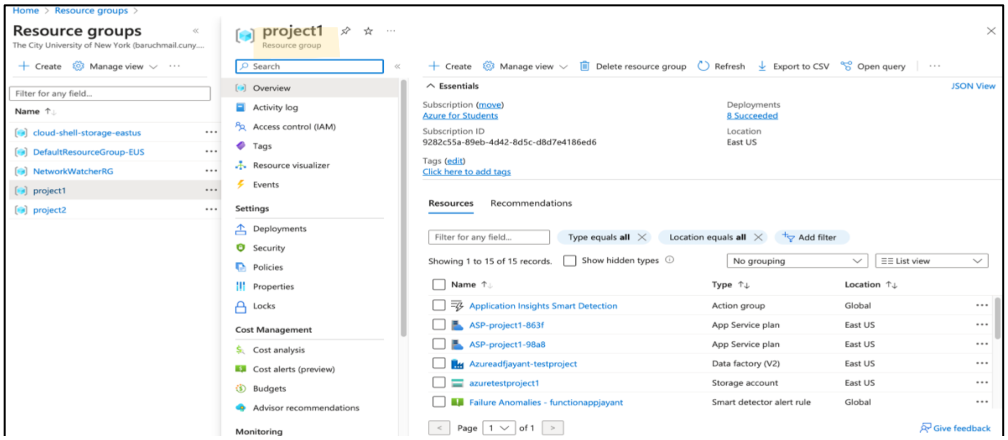
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	Customer_Valid.json	json	June 12, 2023, 17:45:59 (UTC-04:00)	85.2 KB	Standard

Create Data-lake storage & Landing Folder:

Resource group:

1. Once the data is migrated from "**AWS-s3**" it'll need to be placed in a '<u>Landing Folder</u>', for that we will create a '<u>resource group</u>' and then make a '<u>storage account</u>' inside it in Azure cloud.
2. Resource group encapsulates all the different services that you create inside azure, it makes them easy to manage, we have named ours '**project1**' and assigned a student subscription.



Subscription (move)
Azure for Students

Subscription ID
9282c55a-89eb-4d42-8d5c-d8d7e4186ed6

Tags (edit)
Click here to add tags

Deployments
8 Succeeded

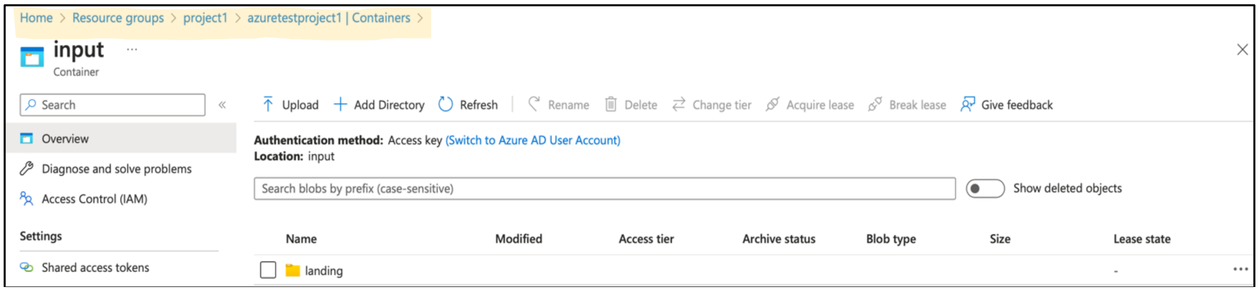
Location
East US

Showing 1 to 15 of 15 records. □ Show hidden types

□	Name	Type	Location
□	Application Insights Smart Detection	Action group	Global
□	ASP-project1-863f	App Service plan	East US
□	ASP-project1-98a8	App Service plan	East US
□	Azureadfjayant-testproject	Data factory (V2)	East US
□	azuretestproject1	Storage account	East US
□	Failure Anomalies - functionappjayant	Smart detector alert rule	Global

Storage account:

1. Inside the resource group, lets create a **storage account**, giving a suitable name.
2. Enable '**<u>hierarchical namespace</u>'**, to make it into an **Azure data lake storage gen2** account.
3. Review+Create to create the storage account.
4. Inside the storage account click in '**container**' to create a storage container and call it 'input'
5. Inside 'input' container we will create our **landing folder** where we want out s3 data to show.



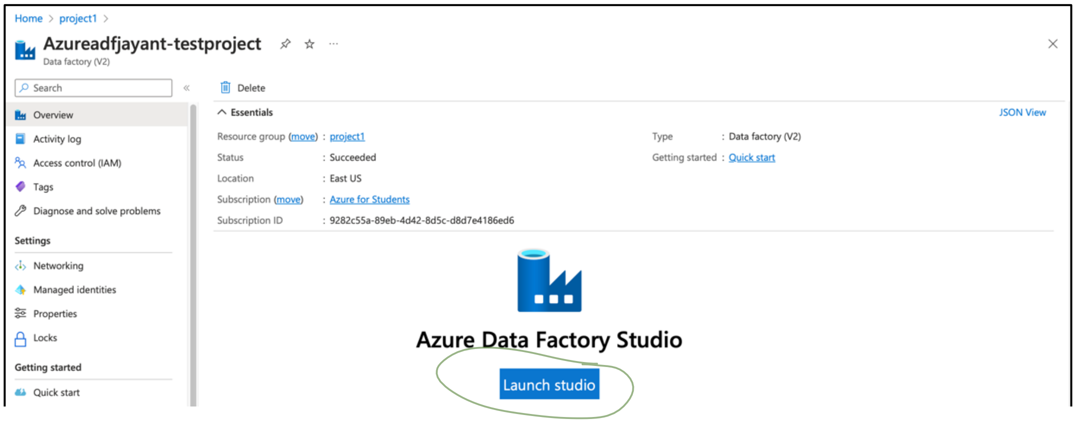
Authentication method: Access key (Switch to Azure AD User Account)
Location: input

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
landing						-

Create Azure Data Factory account:

"We want to move s3 data into azure cloud and for that we'll need an ADF account"

1. Go to azure portal, search for data factories and hit create, provide resource group that we created (project 1).
2. Give a suitable name, give Git configuration later and hit review + create.
3. Go to resource and launch ADF, where we will create pipelines to migrate data.



Resource group (move) : project1
Status : Succeeded
Location : East US
Subscription (move) : Azure for Students
Subscription ID : 9282c55a-89eb-4d42-8d5c-d8d7e4186ed6
Type : Data factory (V2)
Getting started : Quick start

Create Azure Key Vault, secrets & Access Policies :

"We require Acess key & id to able to connect to s3, we will store these secrets in key vault"

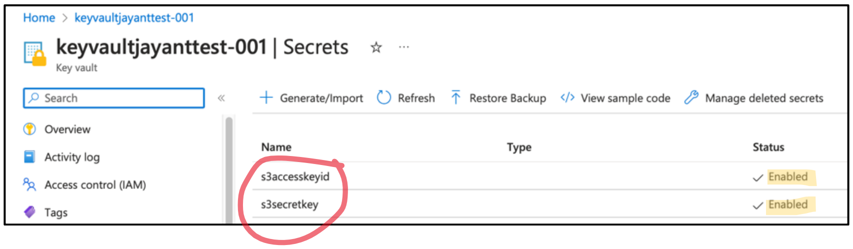
Azure key vault:

1. Go to azure portal, search for 'Key vault' and hit create, provide resource group (project 1).

2. Give a suitable name, review +create

<u>Secrets:</u>

1. Go to 'Secrets' and hit 'generate' to store the AWS 'access key id' and 'secret key'
2. We go to AWS > IAM> Manage Access keys> create access key >copy Key ID
3. Come back to key vault and paste the value and give a name.
4. Create another secret for the 'secret access key', use the same process.

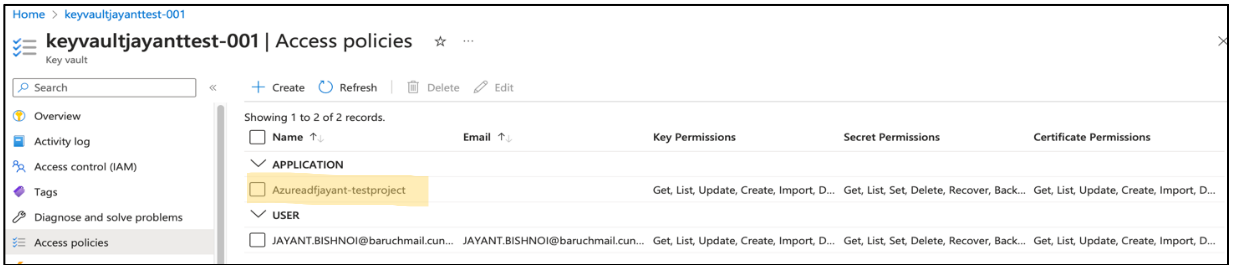


Name	Type	Status
s3accesskeyid		✓ Enabled
s3secretkey		✓ Enabled

<u>Access Policiy:</u>

"For Azure Data Factory to access the secrets, we assign access policies"

1. Access policies > Add access policy > Configure from template > Key, Secret &Certificate Management
2. 'Select principal' which will be our ADF account, paste the name of your account and hit add


Home > keyvaultjayanttest-001
keyvaultjayanttest-001 | Access policies ☆ ···
Key vault
Search
«
+ Create
Refresh
Delete
Edit
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Access policies

Showing 1 to 2 of 2 records.

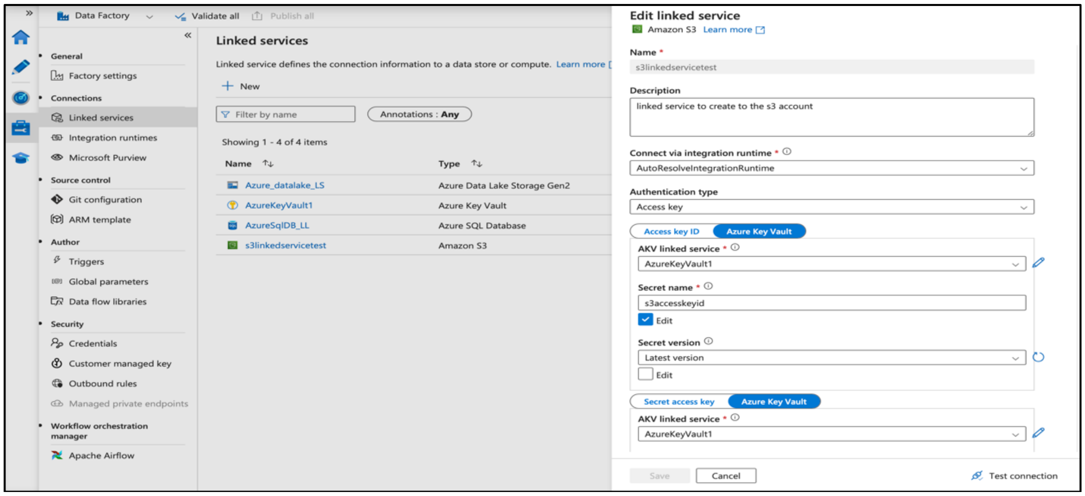
□ Name ↑↓	Email ↑↓	Key Permissions	Secret Permissions	Certificate Permissions
APPLICATION				
□ Azureadfjayant-testproject		Get, List, Update, Create, Import, D...	Get, List, Set, Delete, Recover, Back...	Get, List, Update, Create, Import, D...
USER				
□ JAYANT.BISHNOI@baruchmail.cun...	JAYANT.BISHNOI@baruchmail.cun...	Get, List, Update, Create, Import, D...	Get, List, Set, Delete, Recover, Back...	Get, List, Update, Create, Import, D...

Phase 2 – Migrating Data

Create Data pipelines in ADF:

Linked Service to connect to s3:

- Click Manage > new > select s3 > give a name > in authentication, choose key vault >new AKV linked service>select your key vault > test connection (we added policy to ADF to connect to key vault) > create>give secret name > test connection.
- If connection is successful it means, ADF can successfully connect to s3.



Linked services

Linked service defines the connection information to a data store or compute. Learn more

Showing 1 - 4 of 4 items

Name	Type
Azure_datalake_LS	Azure Data Lake Storage Gen2
AzureKeyVault1	Azure Key Vault
AzureSqlDB_LL	Azure SQL Database
s3linkedservicetest	Amazon S3

Edit linked service

Amazon S3 Learn more

Name *: s3linkedservicetest

Description: linked service to create to the s3 account

Connect via integration runtime *: AutoResolveIntegrationRuntime

Authentication type: Access key

Access key ID | Azure Key Vault

AKV linked service *: AzureKeyVault1

Secret name *: s3accesskeyid

☑ Edit

Secret version: Latest version

☐ Edit

Secret access key | Azure Key Vault

AKV linked service *: AzureKeyVault1

Linked Service to Azure Data Lake storage:

- Click Manage > new > select Data Lake gen2 > give a name > in authentication, choose account >select storage account>select your key vault > test connection.
- If connection is successful it means, ADF can successfully connect to data lake storage account.

Pipeline:

- Author tab > hit '+' > give a name >copy activity
- In copy activity give source and sink
- In source **since we don't have an existing dataset**, we need to create one > select s3>json format>give name> select link service>browse s3 bucket name >select only till year

"Because data is updated daily, and folders keep changing we need to parameterize data set to dynamically pull data"

- Click on **advanced and open dataset** > go to parameter> '+' > under name "folderpath"
- Under connection > in directory, add dynamic content> choose 'folderpath' parameter

- Now go back to pipeline > in **source**, add dynamic content to folder path
- ## @concat(formatDateTime(utcnow(), 'yyyy'), '/', formatDateTime(utcnow(), 'MM'), '/', formatDateTime(utcnow(), 'dd'), '/') ##

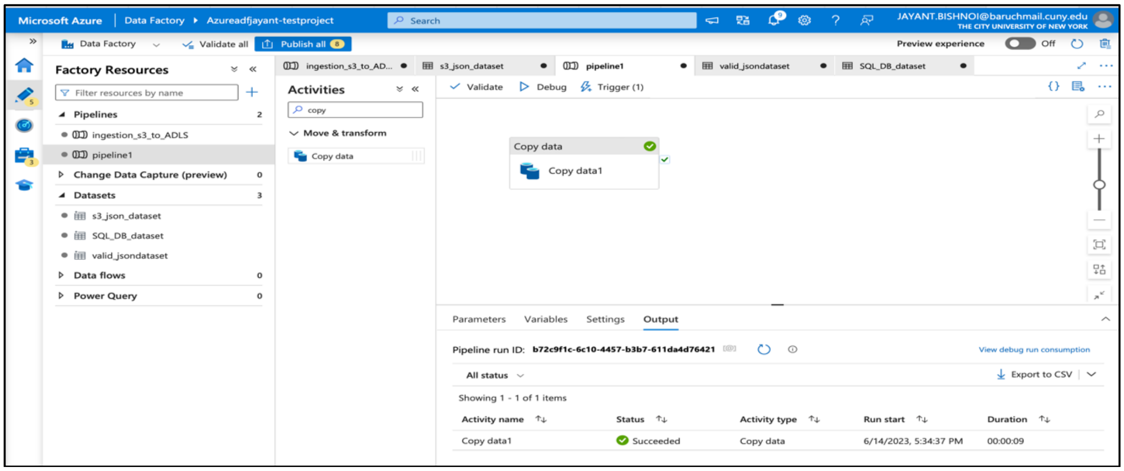
- File path type > **wildcard file path** (this will pull any of a given format, .json here)

"The folder path will now be created dynamically and is not a constant date."

- In the pipeline **sink** > create dataset> ADLS gen2 > json > give name> select linked service>select input container (folder we will select dynamically)
- open dataset > create parameter (landingFolder)
- in connection > in file path >add dynamic directory> selecting landingFolder parameter

- Now go back to pipeline > in **sink**, add dynamic content to folder path, to have hierarchical folder layout in out landing folder
- ## @concat('landing/',formatDateTime(utcnow(), 'yyyy'), '/', formatDateTime(utcnow(), 'MM'), '/', formatDateTime(utcnow(), 'dd'), '/') ##

- Hit debug to run pipeline and make sure the s3 folder path starts **from ' current day's date'**

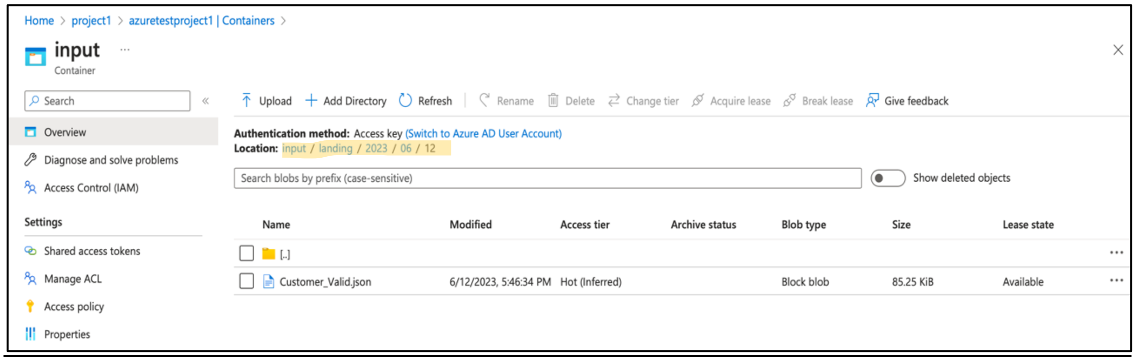


Pipeline run ID: b72c9f1c-6c10-4457-b3b7-611da4d76421

Showing 1 - 1 of 1 items

Activity name	Status	Activity type	Run start	Duration
Copy data1	✓ Succeeded	Copy data	6/14/2023, 5:34:37 PM	00:00:09

(We can see the landing folder has been successfully created.)



Authentication method: Access key (Switch to Azure AD User Account)
Location: input / landing / 2023 / 06 / 12

	Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
□	[..]						
□	Customer_Valid.json	6/12/2023, 5:46:34 PM	Hot (Inferred)		Block blob	85.25 KiB	Available

Phase 3 – Data Validation

Creating Azure function:

"We need to define ***when*** *the function will run, so we will create it with a* ***BLOB TRIGGER*** *so that as soon as a file comes in the 'landingFolder' from s3 on a daily basis, the azure function is triggered to determine whether the file has a* ***valid*** *JSON format or not and moves into respective Staging or RejectedFolder"*

1. Function app > '+' > subscription>resourcegroup > give name>publish as code > runtime stack (node.js)
2. Hosting > *"Azure function app will by default create and use a separate storage account"* >Serverless consumption > review and create.

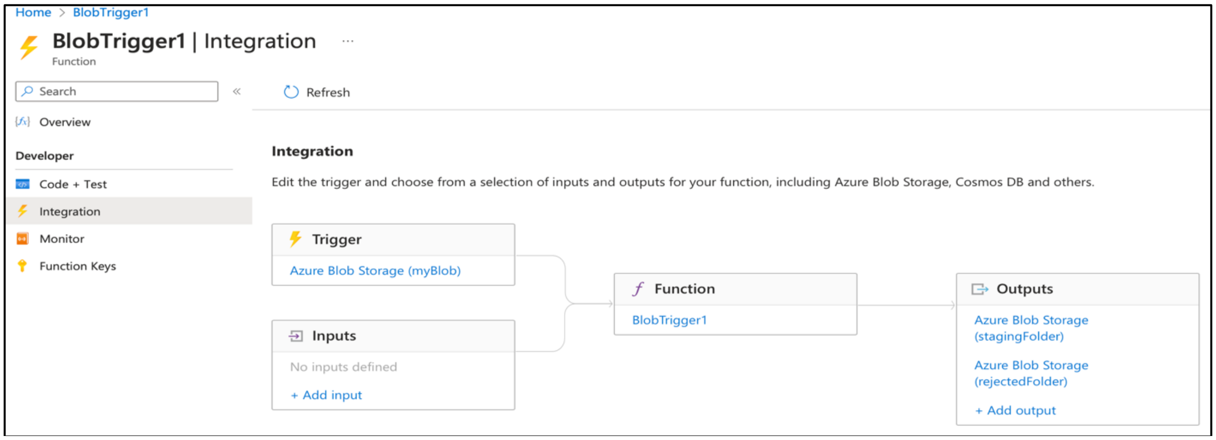
1. In the function app go to **function tab** > '+' >develop in portal > Azure Blob storage trigger (*This means the function will get executed as soon as blob gets created)*

2. In integration tab > trigger>path – input/landing > storage account name>save.

   (*We will add 2 outputs if the function execution is successful)*

2.1 storage account name>Parameter name- stagingFolder > path- input/staging/{rand-guid}.json

2.2 storage account name>Parameter name- rejectedFolder > path-input/rejectedFolder/{rand-guid}.json



Integration

Edit the trigger and choose from a selection of inputs and outputs for your function, including Azure Blob Storage, Cosmos DB and others.

Trigger
Azure Blob Storage (myBlob)
Inputs
No inputs defined
+ Add input
Function
BlobTrigger1
Outputs
Azure Blob Storage (stagingFolder)
Azure Blob Storage (rejectedFolder)
+ Add output

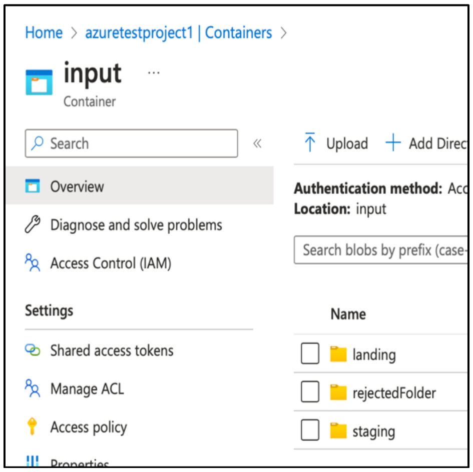
In the code + test tab, write the following code --

```
module.exports = async function (context, myBlob) {
    context.log("JavaScript blob trigger function processed blob \n Blob:");
    context.log("********Azure Function Started********");
    var result =true;
    try{
        context.log(myBlob.toString());
        JSON.parse(myBlob.toString().trim().replace('\n', ' '));
    }catch(exception){
        context.log(exception);
        result =false;
    }
    if(result){
        context.bindings.stagingFolder = myBlob.toString();
        context.log("********File Copied to Staging Folder Successfully********");
    } else{

        context.bindings.rejectedFolder = myBlob.toString();
        context.log("********Inavlid JSON File Copied to Rejected Folder
Successfully********");
    }

    context.log("*******Azure Function Ended Successfully*******");

};
```



Authentication method: Acc
Location: input

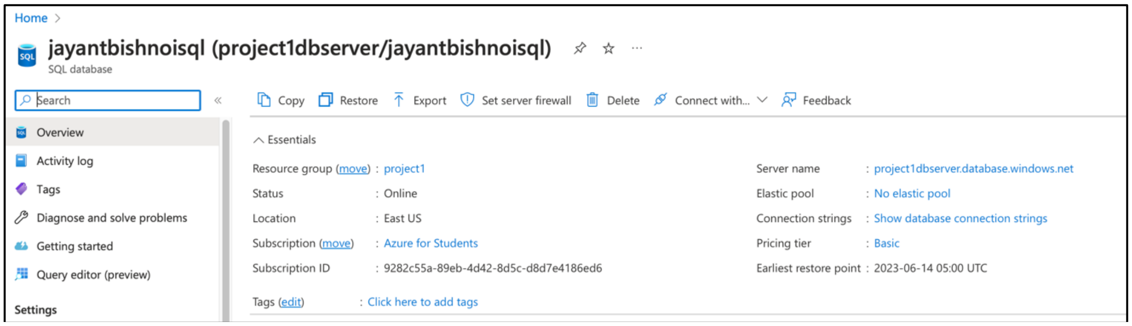
Name
□ landing
□ rejectedFolder
□ staging

Phase 4 – Data Storage

Create SQL Database:

1. Create > subscription > resource group > database name.
2. Create new server and give name > SQL authentication > define username and password > compute + storage (basic tier)> LRS> public endpoint> allow access to DB> review +create.

(Go to resource and go to query editor, give login and password, allow your IP to firewall)



jayantbishnoisql (project1dbserver/jayantbishnoisql)
SQL database

Resource group (move) : project1
Status : Online
Location : East US
Subscription (move) : Azure for Students
Subscription ID : 9282c55a-89eb-4d42-8d5c-d8d7e4186ed6
Tags (edit) : Click here to add tags
Server name : project1dbserver.database.windows.net
Elastic pool : No elastic pool
Connection strings : Show database connection strings
Pricing tier : Basic
Earliest restore point : 2023-06-14 05:00 UTC

Create Pipeline to move staging data to SQL Database:

*"We will create a **storage event trigger** so that the pipeline is executed only as some data comes into the staging folder."*

1. Click '+' and create one pipeline > search 'copy.'
2. In source > create new dataset (+ sign) > gen 2 > json > linked service (data lake ls) / path- input/staging
3. Create another dataset for destination> Azure SQl DB > name.
4. Create new linked service > test connection >project server>username>password>test conection>create

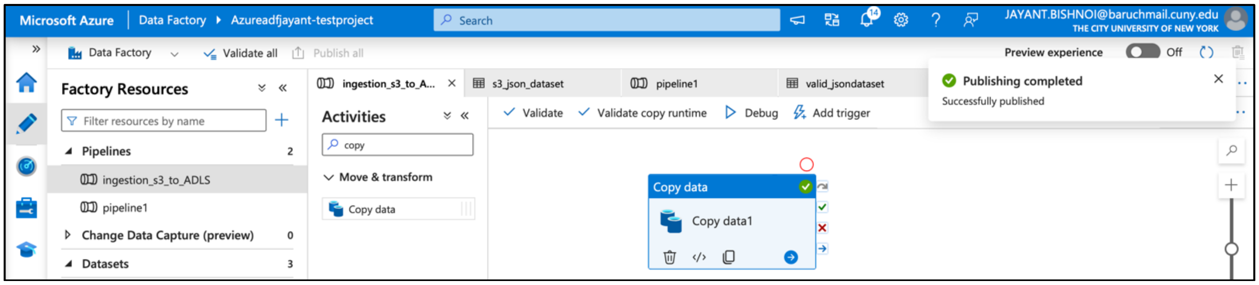
5. Create a table (contains the data from ADLS into SQL DB):

```
create table [dbo]. [VehicleData1] (
VehicleID nvarchar (100),
latitude decimal,
longitude decimal,
City nvarchar (100),
temperature int,
speed int
)
```

6. Refresh and put table name
7. In the pipeline under source put > valid json dataset > wildcard file path < under sink create SQL DB dataset

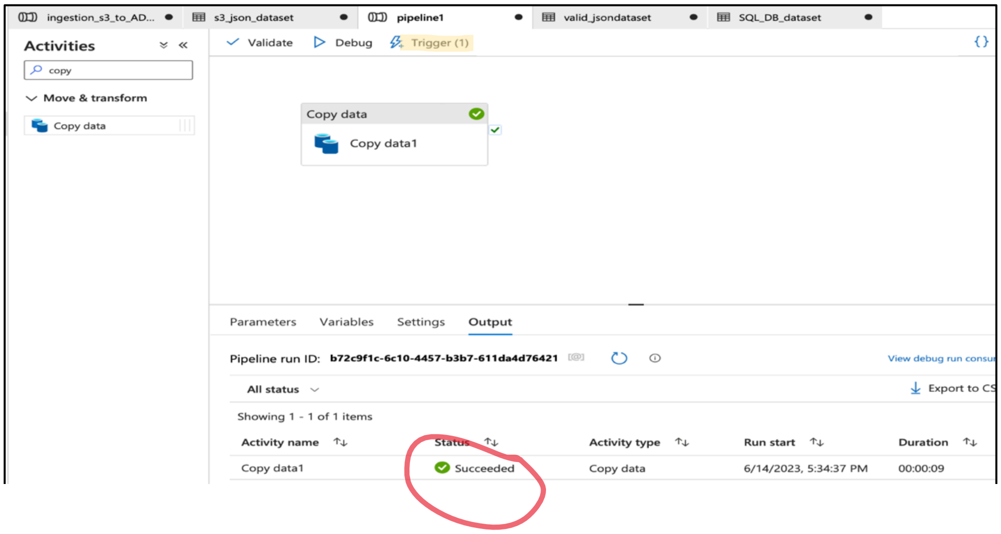
8. Click on add trigger button > new > name > type -storage events > storage account name > container name – input > path begins – staging/ > event- Blobcreated > start trigger on creation> continue

9. Publish.



Publishing completed

Successfully published



Pipeline run ID: b72c9f1c-6c10-4457-b3b7-611da4d76421

Showing 1 - 1 of 1 items

Activity name	Status	Activity type	Run start	Duration
Copy data1	✓ Succeeded	Copy data	6/14/2023, 5:34:37 PM	00:00:09

THE END