**MICROSOFT AZURE : END TO END PROJECT**

**Step 1 : Create a resource group**

Tags – Used to categorise the resources

**Step 2 : Create a data lake i.e Storage Account**

Redundancy – By default the data will be put under GRS – Geo redundant storage. Because it replicates our data within different regions. But we do not need to do that. We will choose LRS – Locally redundant storage i.e store the data in our local.

By default, the storage account is a BLOB storage account, it does not create a data lake. In order to create the data lake we have to select Hierarchical Namespace. Blob storage does not have hierarchical namespace i.e container within a container.

Access tier – 3 types : Hot, Cool , Cold . Hot is used when we have to frequently query the data. Cool is used when we need low cost data. Cold is used when we have minimal use of the data and we have to pay more if we want use more. Since we frequently use the data, we select Hot.

**PHASE 1 – AZURE DATA FACTORY – Loading raw data into bronze layer**

**Step 3 : Create Azure Data Factory**

Click launch studio

Side panel contains: Author – to create pipelines, Monitor to check the status of our pipelines, manage and learning center

**Step 4: Create containers – 1 each for bronze, silver and gold zone**

Go to resources and click on our data lake -> left side panel -> data storage

There are 4 types of data storage : Containers (popularly store data of data lake), file shares (store all files of the organisation) , queues (for nosql data like JSON data), tables (nosql databases , very good if you want some analysis on sem structured data i.e keys and value pairs)

Click on container – create three containers – bronze, silver, gold

**Step 5: Perform data loading into the bronze layer – Create Static pipeline**

Go to ADF -> To create a pipeline , we need a source and destination

Author -> New Pipeline -> Name : GitToRaw

Activities -> Move and Transform -> Copy Data (click and drag)

We should always keep our linked services Ready

Create linked services : Go to the github file -> click on raw : The url will have two parts – Base url (until github.com) and rest is relative url

We need two linked services : 1) Github 2)Data Lake

Manage -> Linked Services ->

1. Search for HTTP(because this is a url from github) -> give name, base url -> Authentication type : Anonymous -> Test connection -> Successfully created the linked service which will pick our dataset -> Create
2. Search for Data lake -> Azure Datalake Storage Gen 2 -> Name : storagedatalake -> Storage Account -> Select your datalake -> Test connection -> Successful -> Create

Now we are ready to create our datasets

Author -> Select the copy data -> Source -> New -> HTTP -> select type of file (CSV, AVRO, ORC, Excel, parquet, ect) -> Name: ds\_http -> Linked Services -> select the http linked service -> Provide the relative url -> Ok

Do the same for the datalake linked service. Source ->New-> Data lake gen 2 -> file type : CSV (because we need to provide the data as such in the bronze layer – no transformations) -> Name: ds\_raw -> Linked service -> Select the storagedatalake service -> Provide the file path: bronze/products/products.csv -> OK

Debug the copy data -> FIRST STATIC PIPELINE CREATED SUCCESSFULLY

Publish all -> To save all the work

**Step 6: Create dynamic pipelines – To fetch multiple files**

Create 3 parameters – relative url, folder, file. These three are the dynamic components which will keep on changing -> “for each activity”

Create a for each activity and the copy activity will be placed inside the for each activity.

**Step 6.1:** Drag Copy Data -> Name: DyamicCopyData

Source -> New -> HTTP -> Filetype: CSV -> Name: ds\_git\_dynamic -> linked service : select httplinkedservice that we created (because the base url will remain the same, only the relative url will change)

Click Advanced -> click relative url -> add dynamic content -> add new parameter -> Name: p\_relative\_url -> Type: String(default) -> Save -> Pick the parameter -> @dataset.p\_relative\_url -> Ok

Go back to the copy activity -> The source will now contain the parameter. Do the same for the sink . Create two parameters for the folder and file name.

Next we have to specify the value for the parameters. For that we will use for each activity.

Drag for each activity into the board.

**Step 6.2:** Create a json file with the relative url, folder and file name as a dictionary of key value pairs. Upload the json file in storage account : Home -> click on the awsstoragedatalake -> create a new container named parameters -> Upload the file

**Step 6.3:** Create a lookup activity to pull the data. Settings -> New -> data lake -> File type: JSON -> Name : ds\_git\_parameters-> Linked service: storagedatalake-> Select the file path using the browse option.

Deselect first row only – because it will run the loop only for the first dictionary, we need all the data .

Run only the lookup, by deactivating the other components to check the status . Click on the lookup and click on the output icon -> it will contain all the values that we need

**Step 6.4:** Then connect the lookup to for each activity. For each activity -> Settings -> Tick sequential -> Items -> select the lookup activity and add .value

Step 6.5: Embed the copy activity within the for each activity

Cut the copy activity-> go to activities in the for each activity -> paste the copy activity

Source -> p\_relative\_url -> value -> Add dynamic content -> @item.p\_relative\_url

Sink -> p\_sink\_folder -> value -> Add dynamic content -> @item.p\_sink\_folder

Sink -> p\_file\_name -> value -> Add dynamic content -> @item.p\_file\_name

Challenges :

* ensure to keep the correct key name when setting parameter
* ensure that the base url ends with ‘/’
* ensure to provide the raw relative url itself – it should not be blob as , blob is only viewer access

**PIPELINE SUCCESSFULLY CREATED. DATA LOADED IN BRONZE CONTAINER . CHECK BRONZE !!!!**

**PHASE 2 – DATABRICKS – Data transformation and ingestion into silver layer**

Step 7 : Create Databricks

Home -> Create Resource -> Azure Databricks -> Create -> Select AWPROJECT -> Give Workspace name: adb\_aw\_project -> Pricing Tier : Premium -> Managed Resource Group Name : managed\_adb\_awproject (Same name as workspace)-> Next-> Review+Create -> Create

Resource group manages the VMs, because we work in clusters and the clusters are managed by Databricks. All the cluster information will be stored in Managed Resource Group

Launch workspace

Compute – Where we create clusters

Workflows – For orchestration

Step 8: Create cluster

Give Name -> Policy: Unrestricted -> Single Node -> Runtime: Select anytime with LTS (Long Time Support) -> Terminate after :120 minutes -> Access mode: No isolation shared

How will databricks access the data stored in datalake? -> Using a key

We will create an application known as Service Level Application. This application will have access to the datalake. Databricks can access datalake using a service level application credentials.

Step 8: Create Microsoft Entra ID -> Create app registration -> Name: aw\_projectapp

Step 8.1: Create an application

Save the application details:-

Display name : [aw\_project\_app](https://portal.azure.com/#blade/Microsoft_AAD_RegisteredApps/BrandingBlade/appId/ce8644ee-409f-4039-b062-01f9d635e4ba/objectId/12ad33ee-2d35-49ce-aef2-2d190f45ab89/isMSAApp/)

Application (client) ID : ce8644ee-409f-4039-b062-01f9d635e4ba

Object ID: 12ad33ee-2d35-49ce-aef2-2d190f45ab89

Directory (tenant) ID :01e4ece0-99af-4324-8920-45e79afc6425

Supported account types:[My organization only](https://portal.azure.com/#blade/Microsoft_AAD_RegisteredApps/AuthenticationV2Blade/appId/ce8644ee-409f-4039-b062-01f9d635e4ba/objectId/12ad33ee-2d35-49ce-aef2-2d190f45ab89/isMSAApp/)

Client credentials :[Add a certificate or secret](https://portal.azure.com/#blade/Microsoft_AAD_RegisteredApps/CredentialsBlade/appId/ce8644ee-409f-4039-b062-01f9d635e4ba/objectId/12ad33ee-2d35-49ce-aef2-2d190f45ab89/isMSAApp/)

Redirect URIs :[Add a Redirect URI](https://portal.azure.com/#blade/Microsoft_AAD_RegisteredApps/AuthenticationV2Blade/appId/ce8644ee-409f-4039-b062-01f9d635e4ba/objectId/12ad33ee-2d35-49ce-aef2-2d190f45ab89/isMSAApp/)

Application ID URI :[Add an Application ID URI](https://portal.azure.com/#blade/Microsoft_AAD_RegisteredApps/WebApiBlade/appId/ce8644ee-409f-4039-b062-01f9d635e4ba/objectId/12ad33ee-2d35-49ce-aef2-2d190f45ab89/isMSAApp/)

Managed application in local directory :[aw\_project\_app](https://portal.azure.com/#blade/Microsoft_AAD_IAM/ManagedAppMenuBlade/objectId/d71404d3-5f0b-4771-8eb9-d84e4f6f5a3e/appId/ce8644ee-409f-4039-b062-01f9d635e4ba)

Certifications & secret -> New client secret -> Name: awproject -> Copy the value and secret id

Secret value: ~0H8Q~MIC-oj.qXtownRKW81rcfz2uMYPfRqxcMv

Secret ID: 06b019f0-7589-4849-ade1-2677d58c8563

Step 8.2: Assign a role to the application

Home -> awstoragelakemr -> access control -> add role assignment -> storage data blob contributor (allows to read, write and delete data stored in blob containers ) -> add member -> select our aw\_projectapp -> Review + assign . Now this will add the role to your app

[Databricks Tutorial: Connecting to Azure Data Lake Storage Gen2 & Blob Storage | Mount Azure Storage](https://www.youtube.com/watch?v=VkjqViooMtQ)

I connected to azure databricks using unity catalog.Without secret key

Transformations:-

* Read all data
* Basic added columns month & year
* Write the data in parquet format – optimised for data reads , because it is columnar format
  + Modes – 4 different modes are there : append(), overwrite(), error(),ignore()
  + Append() – if we want to merge data
  + Overwrite() – to overwrite
  + error () – If data is already there , it will show an error
  + ignore() – If data is already there, it will not throw an error , but it will not write the data as well.
* Concat first name and last name into full name