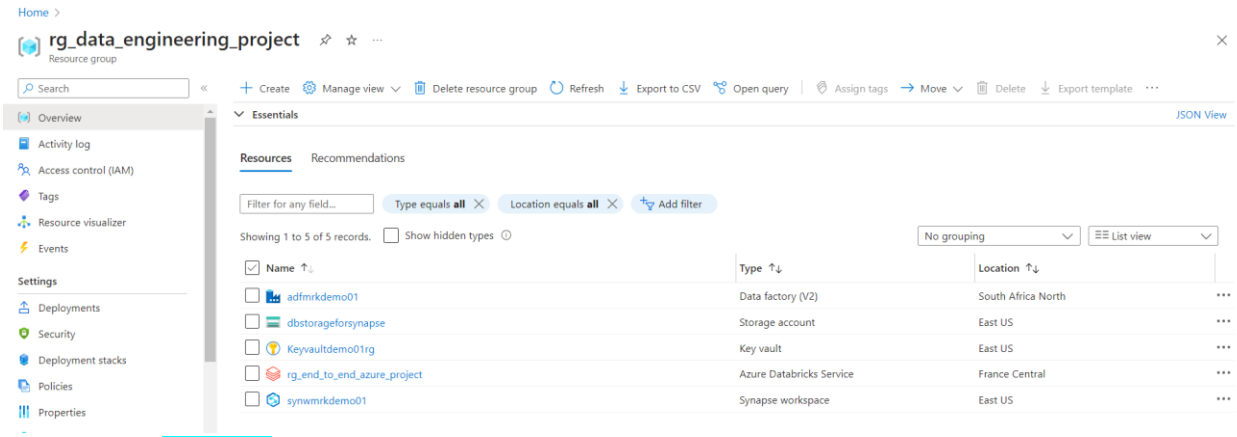
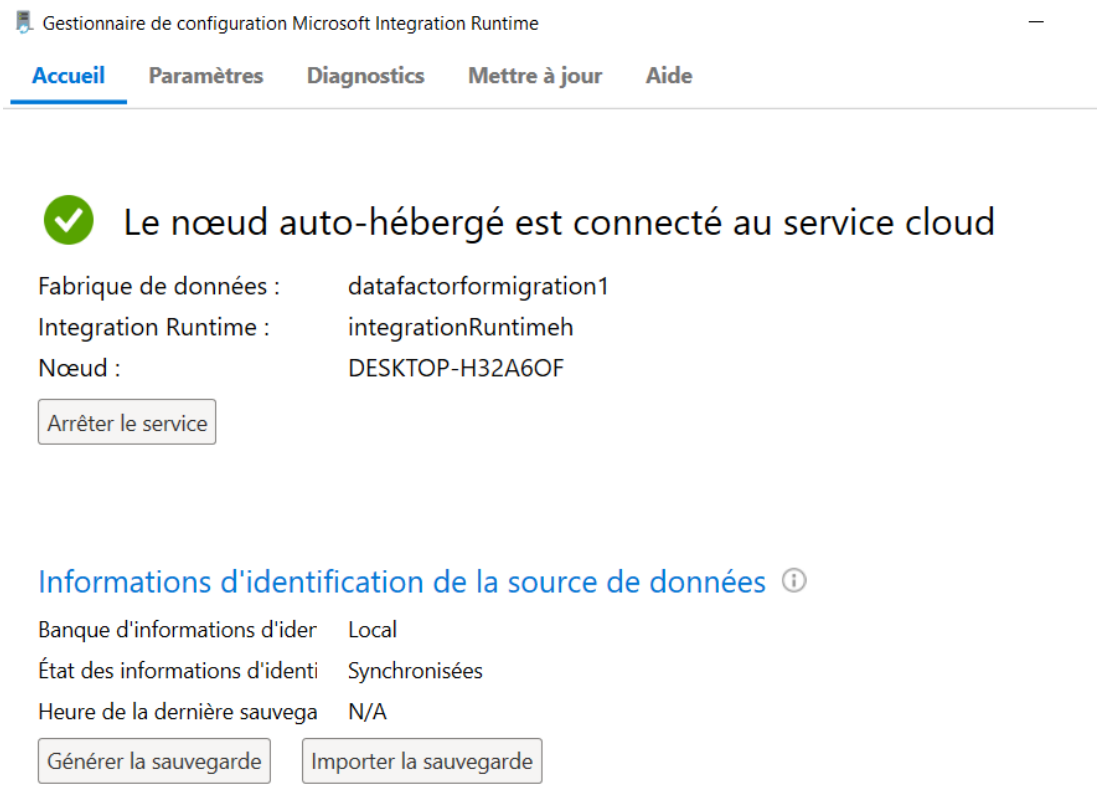
Explain the steps of this task:

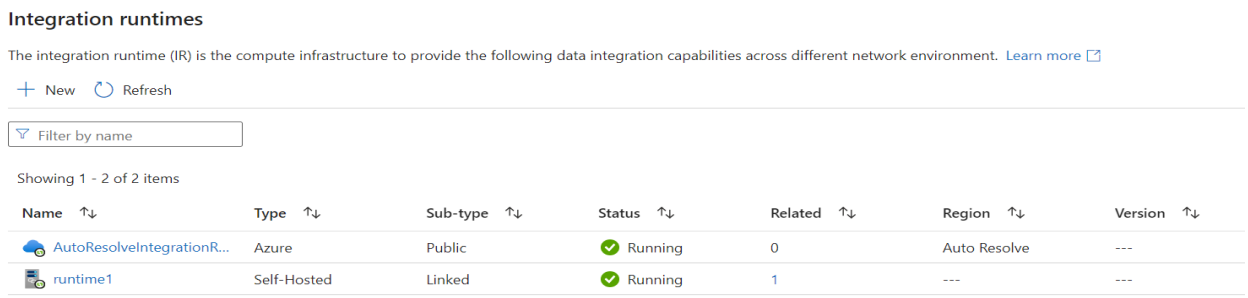
First, we create resource group for Dev environment include all services we will use for ingestion, transformation, and loading



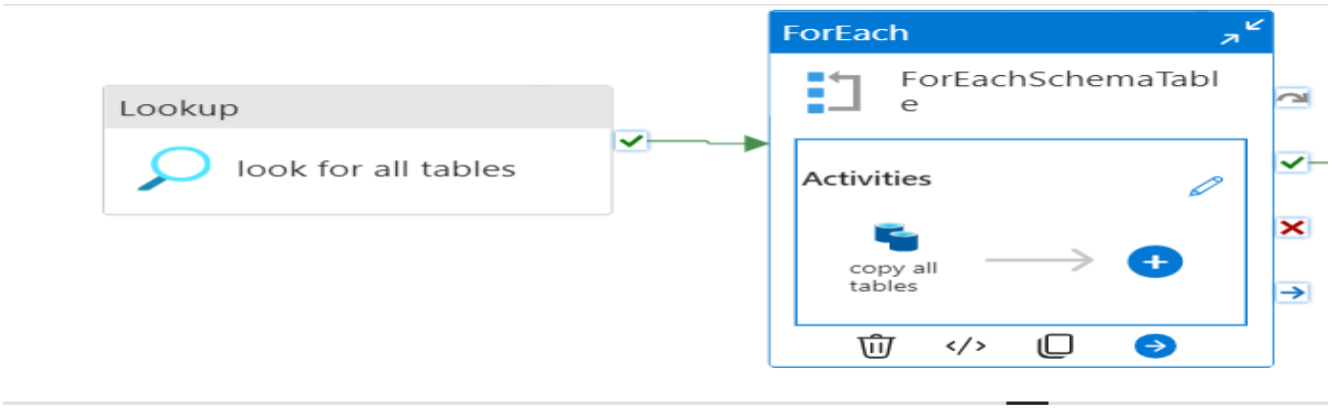
Starting with ingestion

So, we need to set up the host runtime to access the on-premises PostgreSQL DB

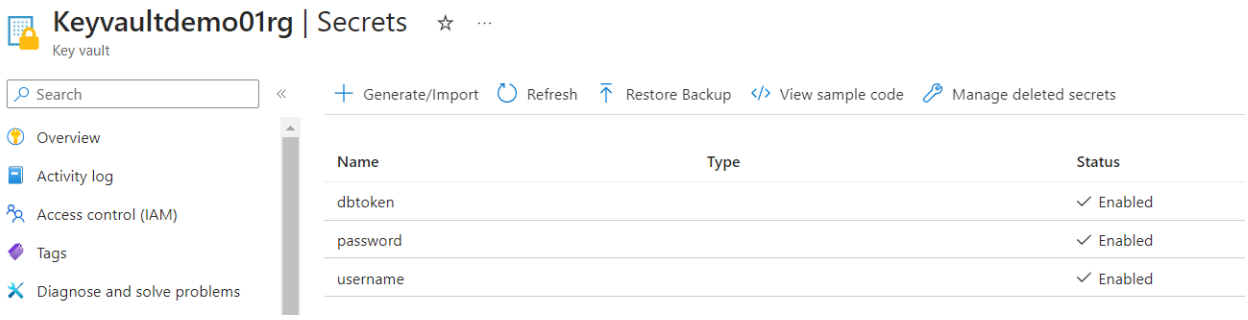




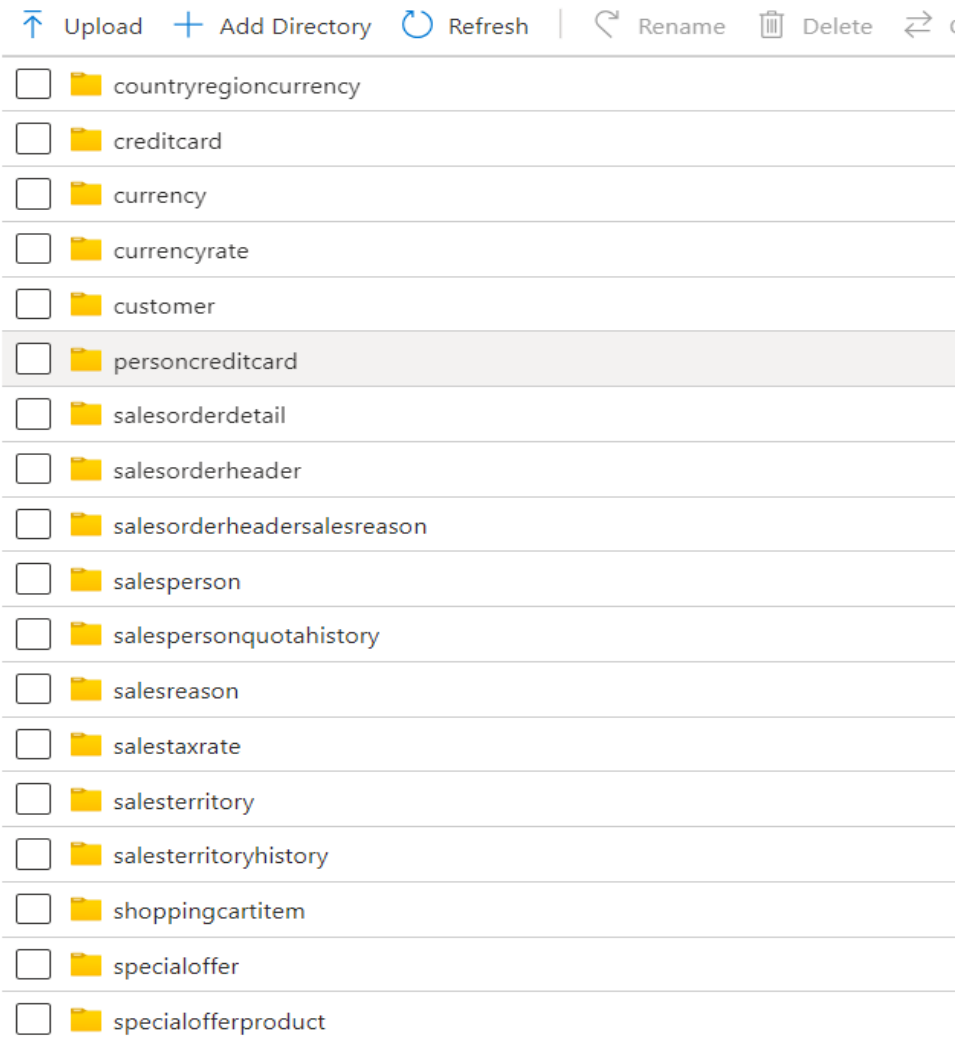
Then we try to copy tables from PostgreSQL to azure data lake Gen2 in parquet format (column-oriented data file format) using Azure Data Factory



For that we use Key vault service for security (we put here our username and password to access the PostgreSQL DB)

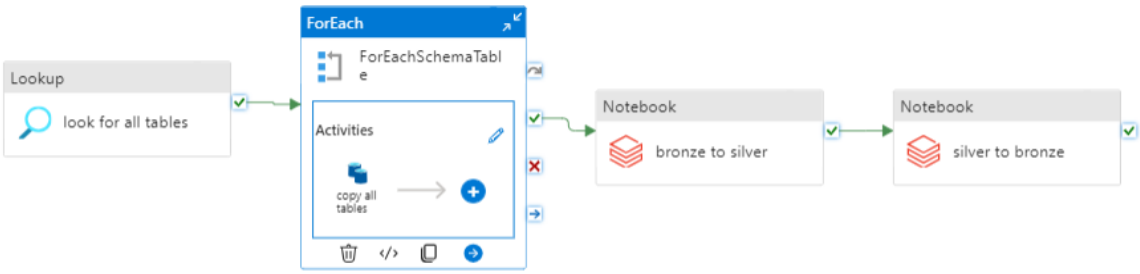


after copy all tables which are now look like this inside the azure Data Lake Gen2 storage:



Then we connect to data bricks to make transformation for our data

So, the pipeline will look like this



The linked services will look like

Linked services

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

+ New

Filter by name | Annotations : **Any**

Showing 1 - 4 of 4 items

Name	Type	Related
AzureDatabricks1	Azure Databricks	1
AzureDataLakeStorage1	Azure Data Lake Storage Gen2	3
AzureKeyVault1	Azure Key Vault	2
PostgreSql1	PostgreSQL	3

+ Container | Change access level | Restore containers | Refresh | Delete | Give feedback

Search containers by prefix | Show deleted containers

Name	Last modified	Anonymous access level	Lease state
bronze	1/31/2024, 9:09:51 PM	Private	Available
filesystemforsynapse	1/30/2024, 7:04:38 PM	Private	Available
gold	2/24/2024, 12:10:12 PM	Private	Available
silver	2/24/2024, 12:09:58 PM	Private	Available

Role assignment for the resource group also including the Azure Active Directory for authentication

rg_data_engineering_project | Access control (IAM)

Resource group

Search

Overview
Activity log
Access control (IAM)
Tags
Resource visualizer
Events
ngs
Deployments
Security
Deployment stacks
Policies
Properties
Locks
Management
Cost analysis
Cost alerts (preview)

+ Add | Download role assignments | Edit columns | Refresh | Remove | Feedback

Search by name or email | Type : **All** | Role : **All** | Scope : **All scopes** | Group by : **Role**

5 items (3 Users, 1 Groups, 1 Service Principals)

Name	Type	Role	Scope	Condition
Owner (2)				
SS SOUHAYLA SOUHAYLA sou834184_gmail.com#EXT#...	User	Owner	Subscription (Inherited)	None
SS SOUHAYLA SOUHAYLA sou834184_gmail.com#EXT#...	User	Owner	This resource	Add
Contributor (1)				
sou834184-databricks cicd-ed4c	App	Contributor	This resource	None
Application Group Contributor (1)				
RG rg_end_to_end_azure_project	Group	Application Group Contributor	This resource	None
User Access Administrator (1)				
SS SOUHAYLA SOUHAYLA sou834184_gmail.com#EXT#...	User	User Access Administrator	This resource	Add

more about the transformation

Here we try to do storage mounting

Cell 1 Python

```python
config = {
  "fs.azure.account.auth.type": "CustomAccessToken",
  "fs.azure.account.custom.token.provider.class": spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}

# Optionally, you can add <directory-name> to the source URI of your mount point.
dbutils.fs.mount(
  source = "abfss://bronze@dbstorageforsynapse.dfs.core.windows.net/",
  mount_point = "/mnt/bronze",
  extra_configs = config)
```

Cell 2

```python
dbutils.fs.ls("/mnt/bronze/sales")
```

Cell 3 Python

```python
config = {
  "fs.azure.account.auth.type": "CustomAccessToken",
  "fs.azure.account.custom.token.provider.class": spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}

# Optionally, you can add <directory-name> to the source URI of your mount point.
dbutils.fs.mount(
  source = "abfss://silver@dbstorageforsynapse.dfs.core.windows.net/",
  mount_point = "/mnt/silver",
  extra_configs = config)
```

Cell 4

```python
config = {
  "fs.azure.account.auth.type": "CustomAccessToken",
  "fs.azure.account.custom.token.provider.class": spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}

# Optionally, you can add <directory-name> to the source URI of your mount point.
dbutils.fs.mount(
  source = "abfss://gold@dbstorageforsynapse.dfs.core.windows.net/",
  mount_point = "/mnt/gold",
  extra_configs = config)
```

Then make transformation from bronze layer to silver layer (we only modified date data type columns)

Then from silver to gold layer the finale data

Cell 11 Python

```python
from pyspark.sql.functions import from_utc_timestamp, date_format
from pyspark.sql.types import TimestampType

for i in table_name:
    path = '/mnt/bronze/sales/' + i + '/' + i + '.parquet'
    df = spark.read.format('parquet').load(path)
    columns = df.columns

    for col in columns:
        if "Date" in col or "date" in col:
            df = df.withColumn(col, date_format(from_utc_timestamp(df[col].cast(TimestampType()), "UTC"), "yyyy-MM-dd"))

    output_path = '/mnt/silver/sales/' + i + '/'
    df.write.format('delta').mode("overwrite").save(output_path)
```

Cell 12

```python
display(df)
```

From silver to gold layer

```python
table_name=[]
for i in dbutils.fs.ls('/mnt/silver/sales'):
  table_name.append(i.name.split('/')[0])
```

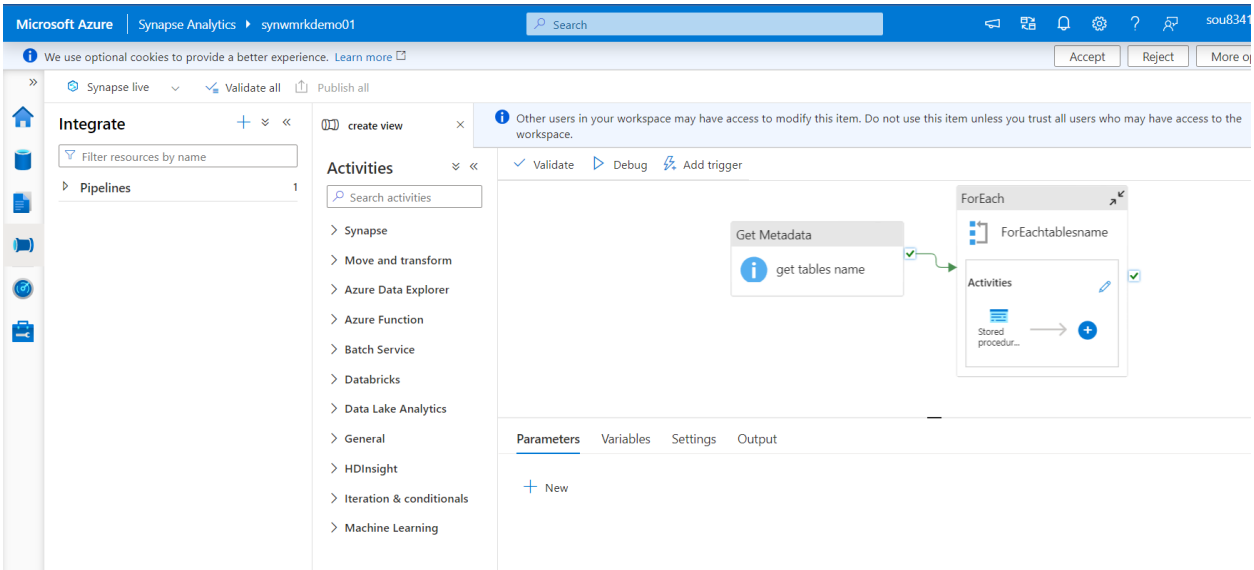
```python
for name in table_name:
    path = '/mnt/silver/sales/' + name
    df = spark.read.format('delta').load(path)
    columns = df.columns

    for old_col_name in columns:
        new_col_name = "".join("_" + char.lower() if char.isupper() and not old_col_name[i-1].isupper() else char for i, char in enumerate
        (old_col_name)).lstrip("_")
        df = df.withColumnRenamed(old_col_name, new_col_name)


    output_path = '/mnt/gold/sales/' + name + '/'
    df.write.format('delta').mode("overwrite").save(output_path)
```

So the three notebooks we have are:

bronze to silver	Notebook	SOUHAYLA SOUHAYLA	2024-02-24 12:34:07
silver to gold	Notebook	SOUHAYLA SOUHAYLA	2024-02-24 12:49:29
storagemounting	Notebook	SOUHAYLA SOUHAYLA	2024-02-24 11:50:42

For the loading task:

We use azure synapse analytics



then we connect power bi to synapse analytic for data reporting (you can explore the dashboard folder to visualize the report or for dynamic use you can use the .pbit file for this)