

**On-prem SQL server Database** is connected to **Azure Data Factory (ETL tool)**. ADF will be used to ingest data into Azure Data Lake gen2 platform.

The Data lake architecture has 3 layers- **Bronze, Silver, and Gold.**

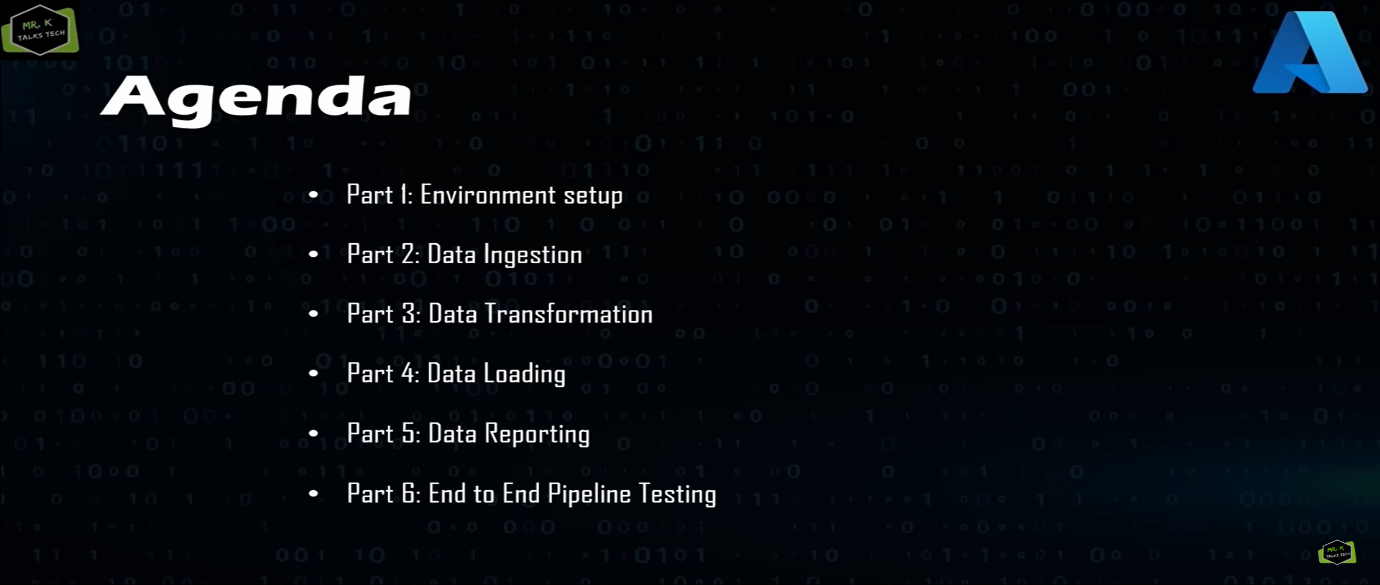
**Bronze**- The data here is exactly identical to the data in on-prem DB. This layer can be used for Data retrieval in case of failures.

**Silver-** The data after the first set of simple transformations such as Column name correction is stored in the Silver layer.   
  
**Gold-** The data in the silver layer are further transformed and stored in Gold layer.

All the transformations are done by using **Azure Data bricks**. The Final transformed data is stored in the DBs created using **Azure Synapse Analytics (ASA).**

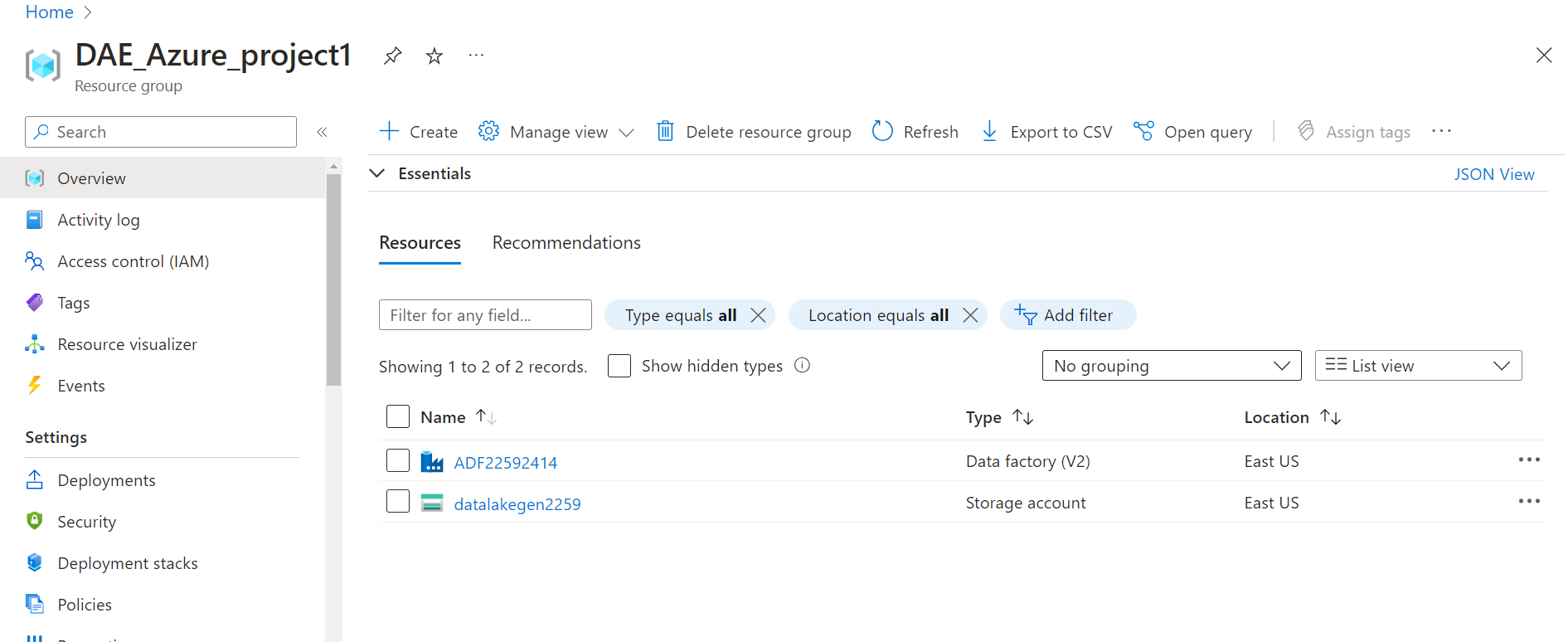
Finally, the DBs in ASA are connected to Power BI to create attractive visuals to present to the stakeholders. We are also making use of Security and Governance tools like Azure AD to ensure secure access to users. Azure key vault securely stores user name and passwords.

Self- hosted to integrate with DBs not on cloud, auto-resolve to integrate with DBs on cloud

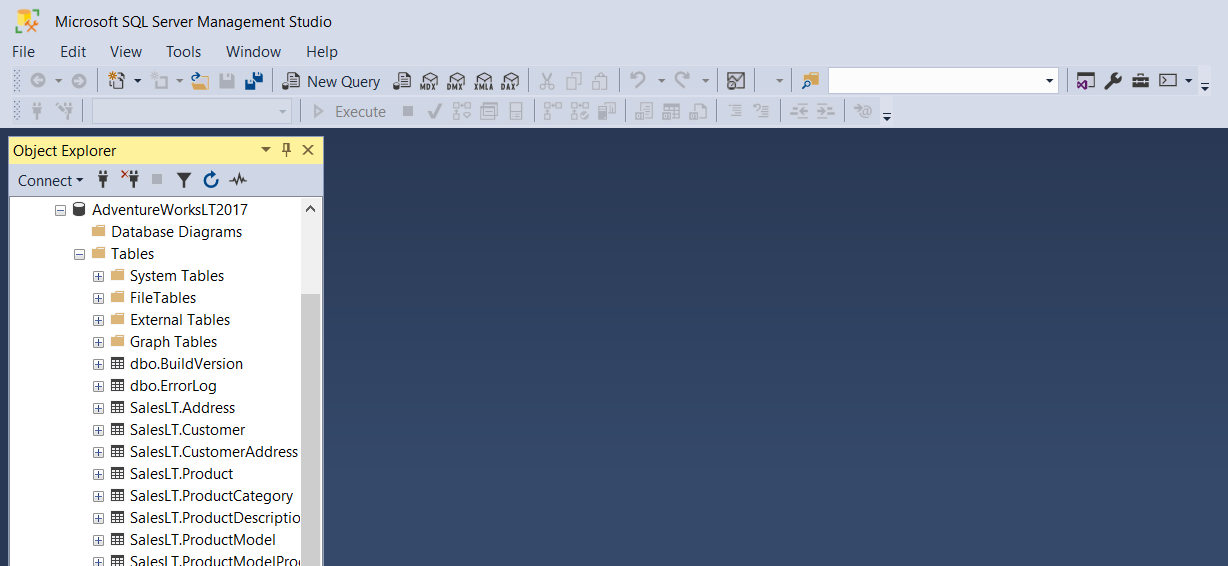


Part 1:

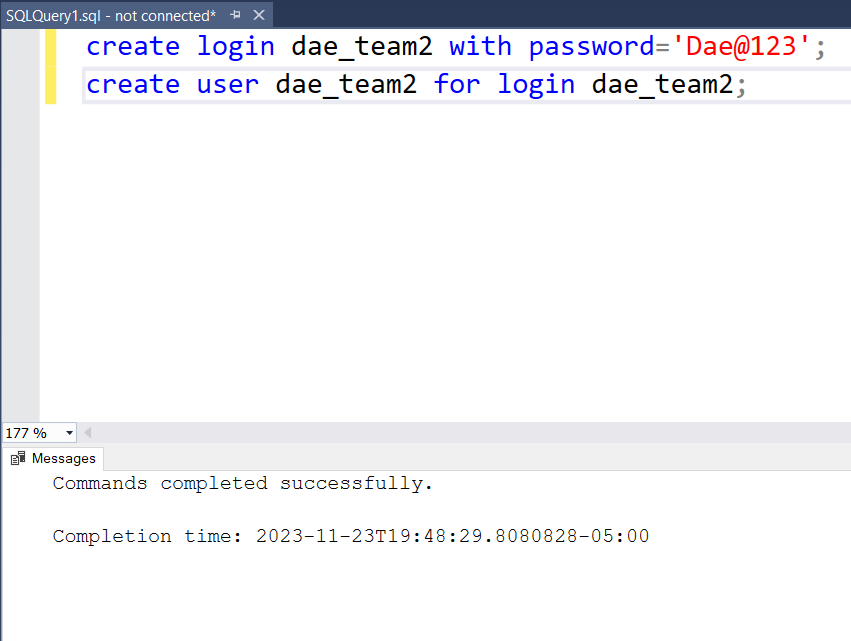
* Downloaded the database schema from [Datalink](https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms)
* Downloaded [SQL server](https://www.microsoft.com/sql-server/sql-server-downloads) and [SQL server management studio](https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16%20)
* Created a resource group and added the necessary services to it



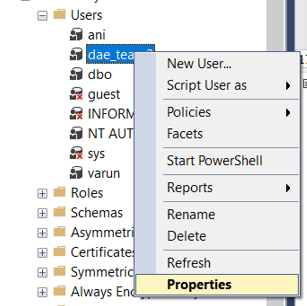
* Moved the .bak file to your SQL Server backup location. This varies depending on your installation location, instance name and version of SQL Server. For example, the default location for a default instance of SQL Server 2019 (15.x) is: C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup.
* Opened SSMS and connect to your SQL Server instance.
* Right-click **Databases** in **Object Explorer** > **Restore Database...** to launch the **Restore Database** wizard.
* Selected **Device** and then select the ellipses **(...)** to choose a device.
* Selected **Add** and then choose the .bak file you recently moved to the backup location. If you moved your file to this location but you're not able to see it in the wizard, this typically indicates a permissions issue - SQL Server or the user signed into SQL Server doesn't have permission to this file in this folder.
* Selected **OK** to confirm your database backup selection and close the **Select backup devices** window.
* Checked the **Files** tab to confirm the **Restore as** location and file names match your intended location and file names in the **Restore Database** wizard.
* Selected **OK** to restore your database.

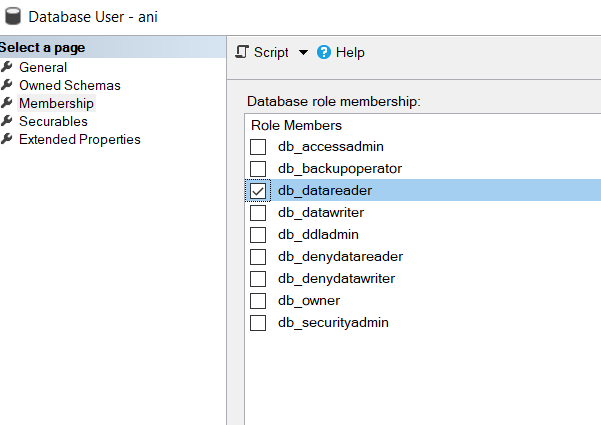


* Created a login and user for the SQL server with a password to establish a connection between cloud and on-premise server (Login: dae\_team2, user-dae\_team2, password: Dae@123)

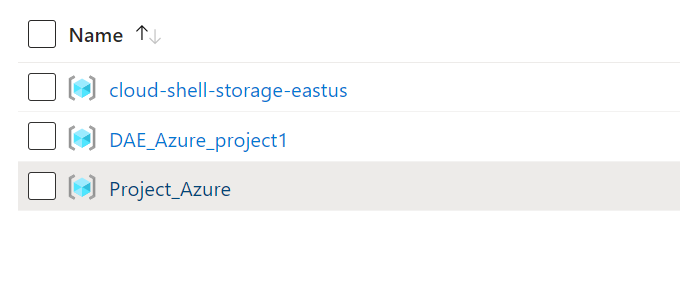


* Provide the right level of access to the created user. Expand security-> right click on the created user and select properties.
* Under the membership tab provide database reader to the created user and save the settings.

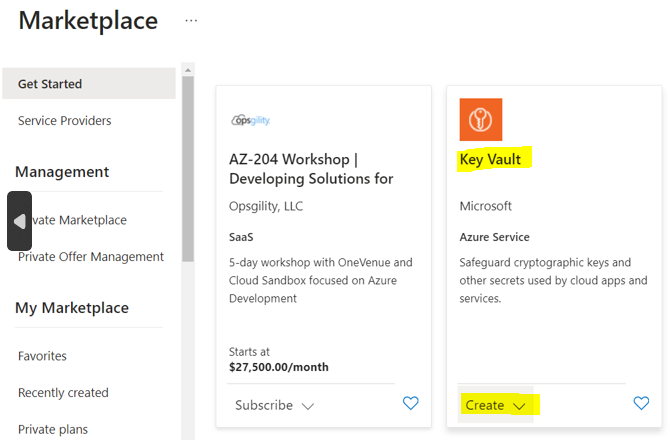




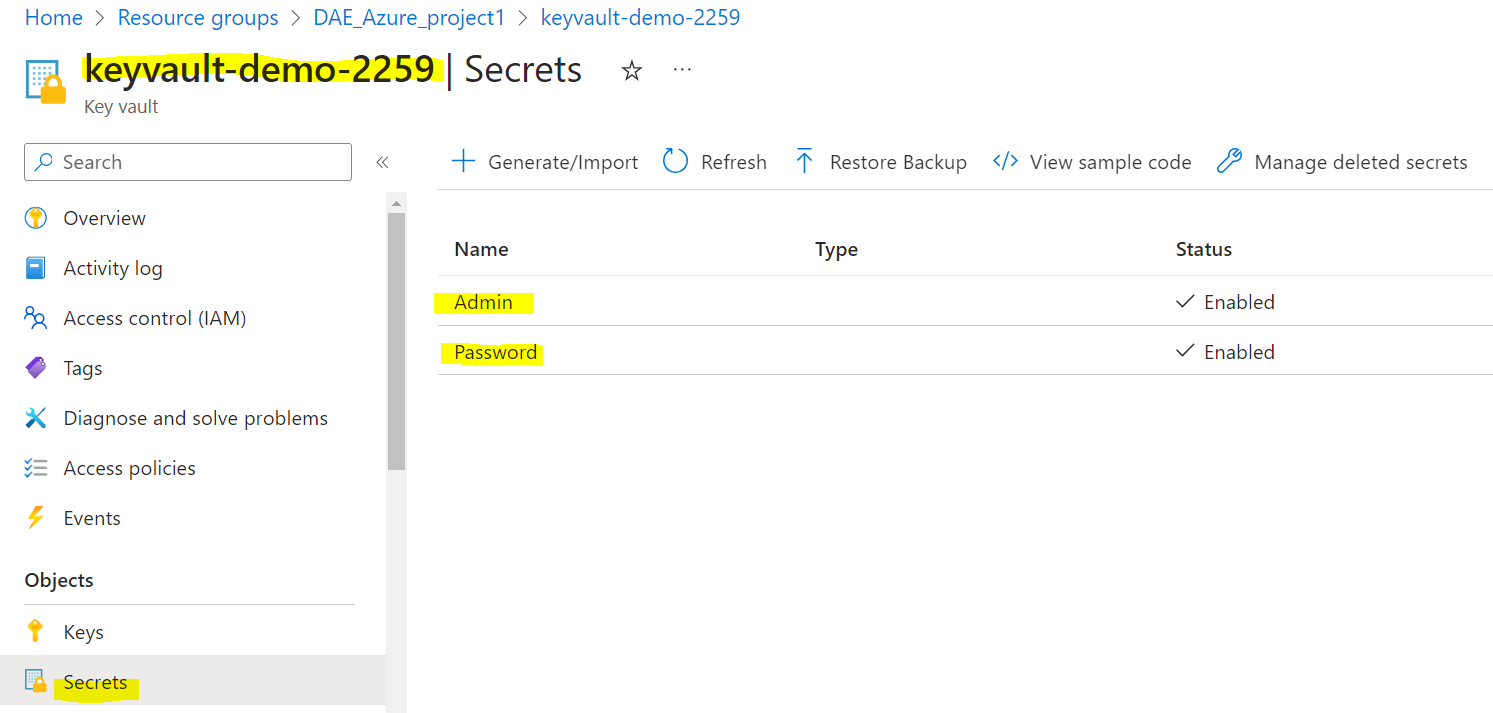
* Created a new resource group in Azure portal called as Dae\_Azure\_project1



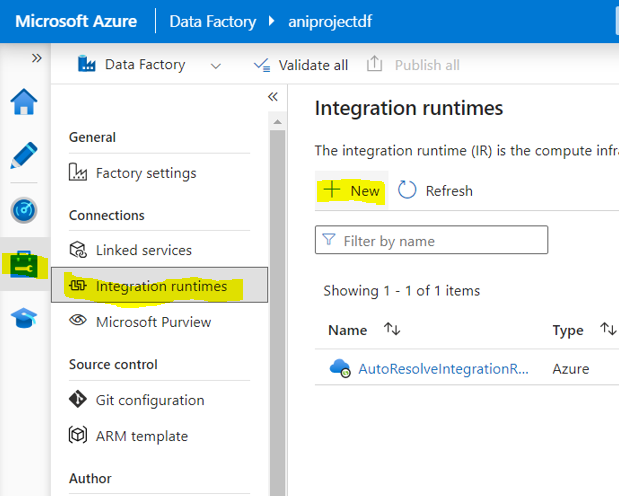
* All the passwords would be stored as secrets in Azure secret vault.
* Created a new Key vault resource inside the resource group from the marketplace.



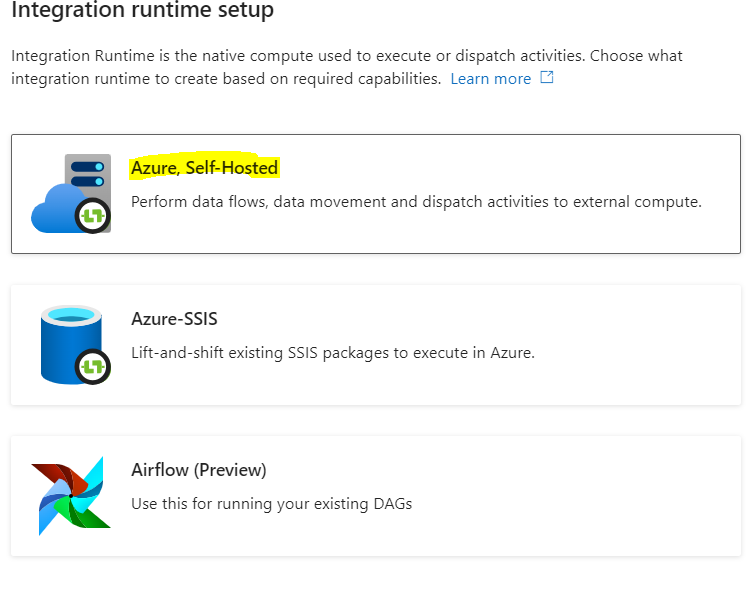
* Created and configured a key vault named “[eyvault-demo-2259](https://portal.azure.com/#@northeastern.onmicrosoft.com/resource/subscriptions/b3131675-25f3-4b2f-95e2-1f368e1bff30/resourceGroups/DAE_Azure_project1/providers/Microsoft.KeyVault/vaults/keyvault-demo-2259)**”**.
* The permission model is Azure RBAC, hence I need to provide the right role to your account to create secrets in Key vault.
* Navigate to access configuration tab and click on GO to access control(IAM)
* In IAM, in the Grant access to resource section click on **Add role assignment**.
* In the role section, select Key vault administrator and in member section add the email ID associated to your portal account. This will allow you to create secrets in key vault.
* This setting needs to performed if we are using RBAC, but for this project we will go with vault access policy, where access can be manages in Access policies tab of the vault

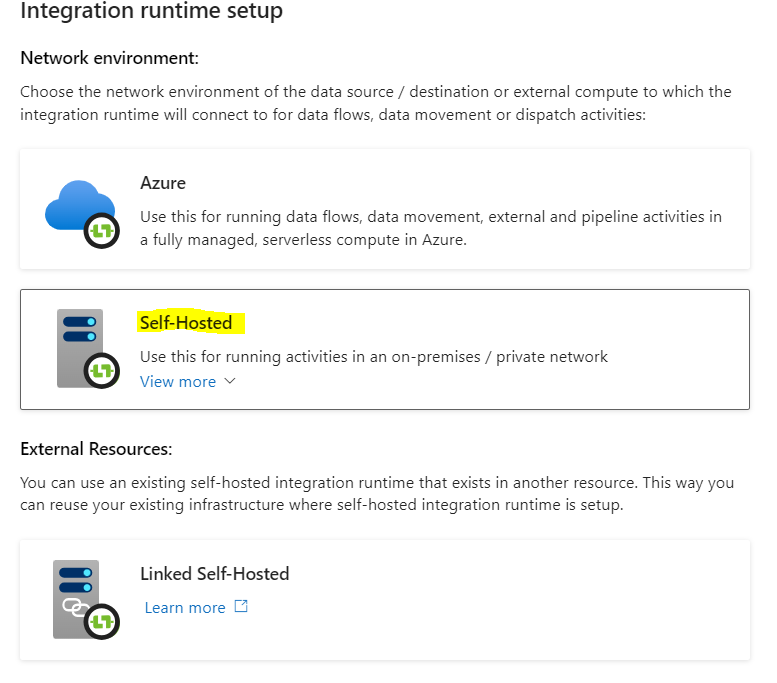


* Click on the key vault resource and go to the Secrets tab and select Generate/Import.
* Create two secrets- **admin** for the created user **dae\_team2** and **password** for the password created for user **dae\_team2**
* This ensures that the actual username and passwords are not used anywhere and only their encrypted alias are used.
* Go to the Marketplace in the created resource group and create and configure the Azure data factory.
* To establish a connection between Azure data factory and on-premise server we need to install self-hosted integration runtime on the device hosting the on-prem server.
* By default we have an Auto-resolve integration runtime installed in ADF to connect to cloud-based databases like Azure DataLake Gen 2.
* To create a self-hosted integration runtime, go to the **manage** tab in ADF, select **integration runtimes**, and click on **+ New**.

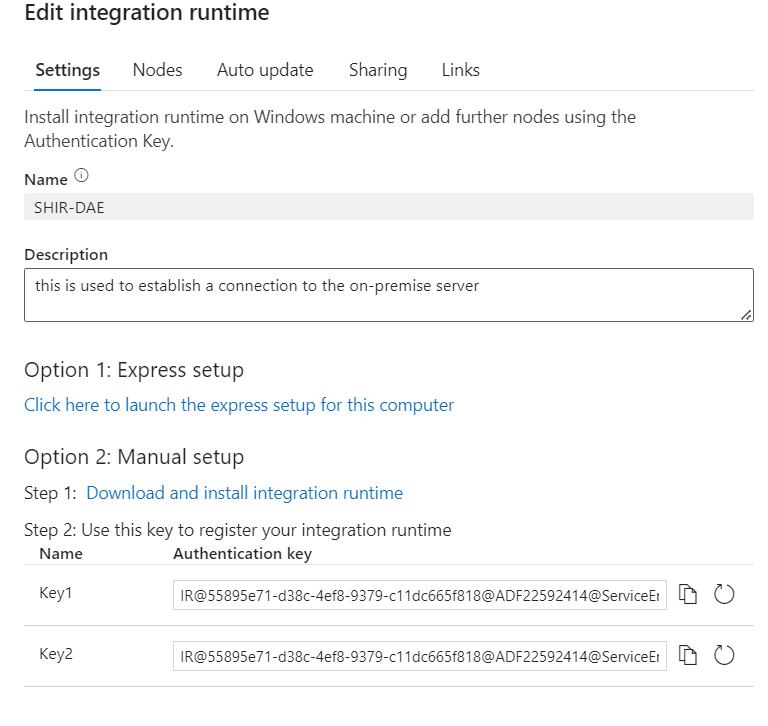


* Select Azure, self-hosted option click on continue. In the next tab select Self-hosted and click on continue

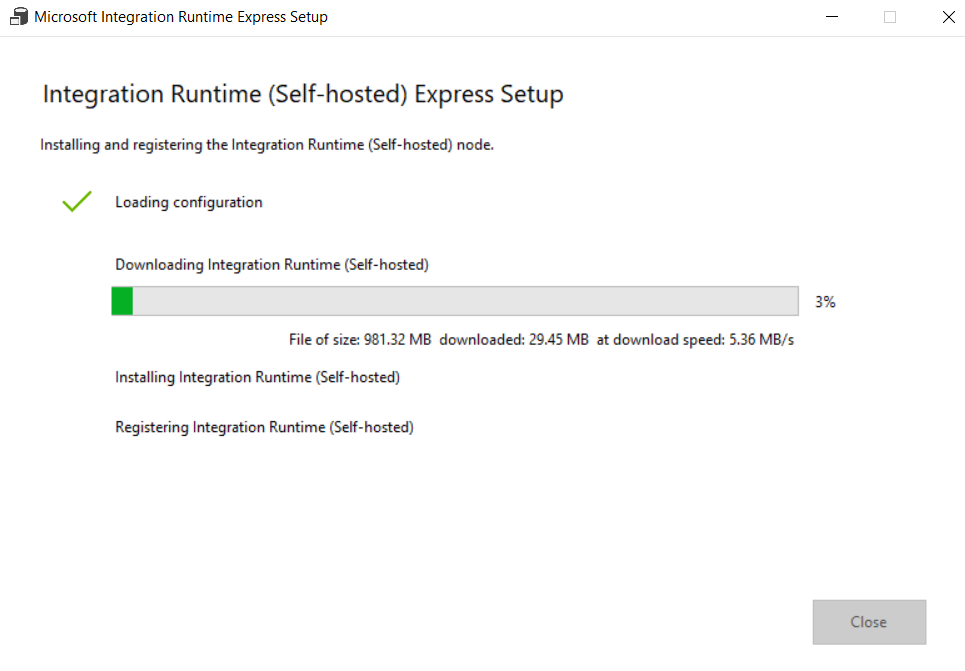


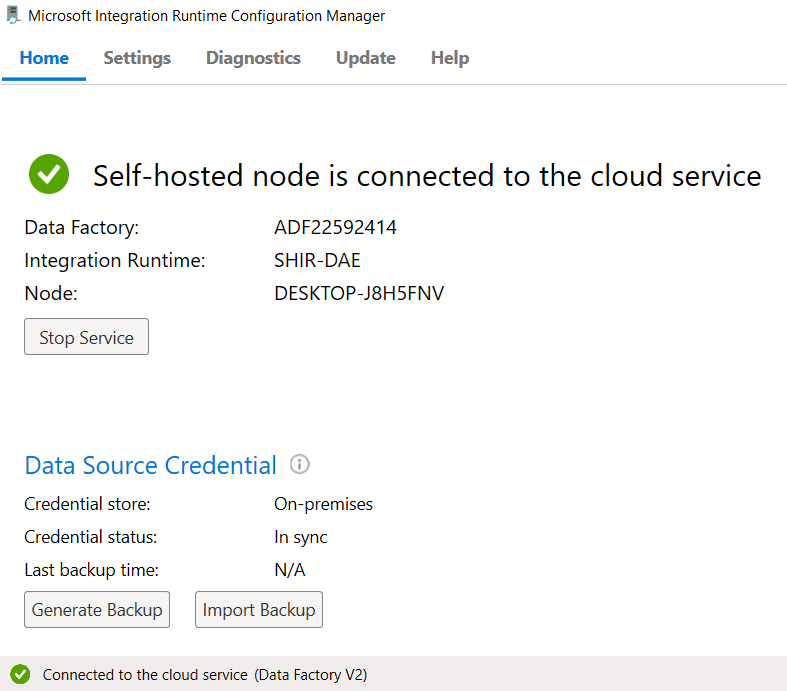


* In the next tab give a name and description for the integration run time.

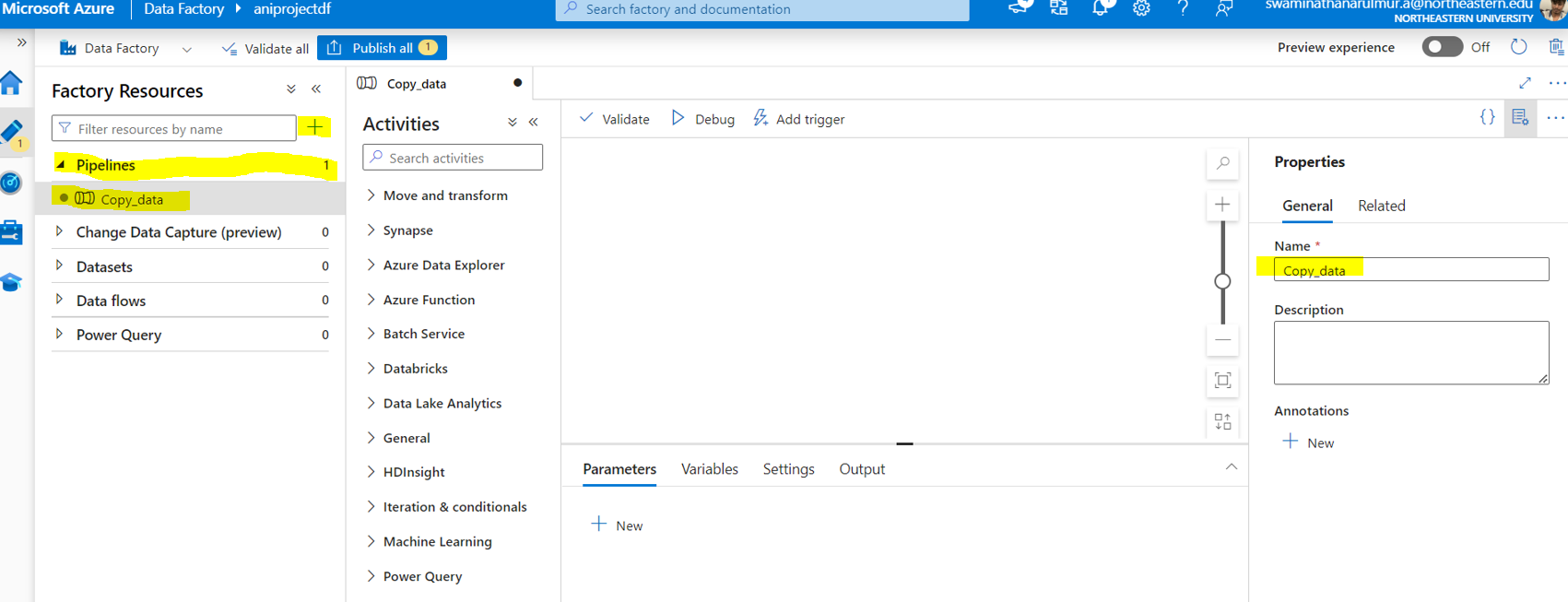


* We will have two options to install the run time namely express and manual
* Express allows you to install the same machine in which you are running Azure and no key is needed and manual allows to install the runtime in a different machine where the server is hosted and a key is needed.
* Integration runtimes setup the required compute infrastructure for the ADF
* Since the Azure portal and the onprem server in this project is on the same machine, we are going with the express setup. Once the config file is downloaded it is launched for further setup.

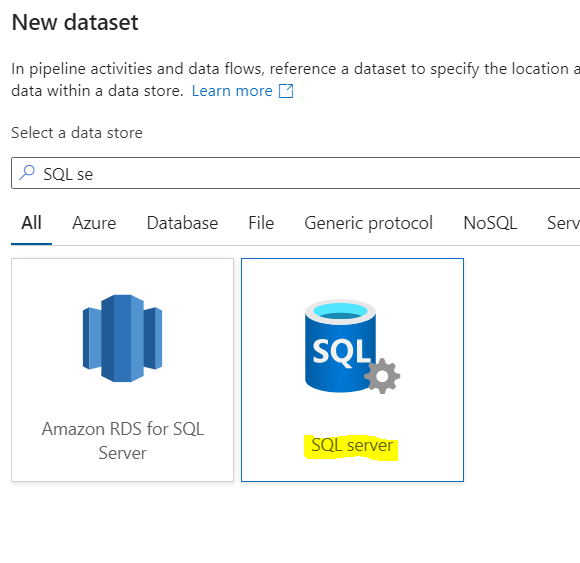




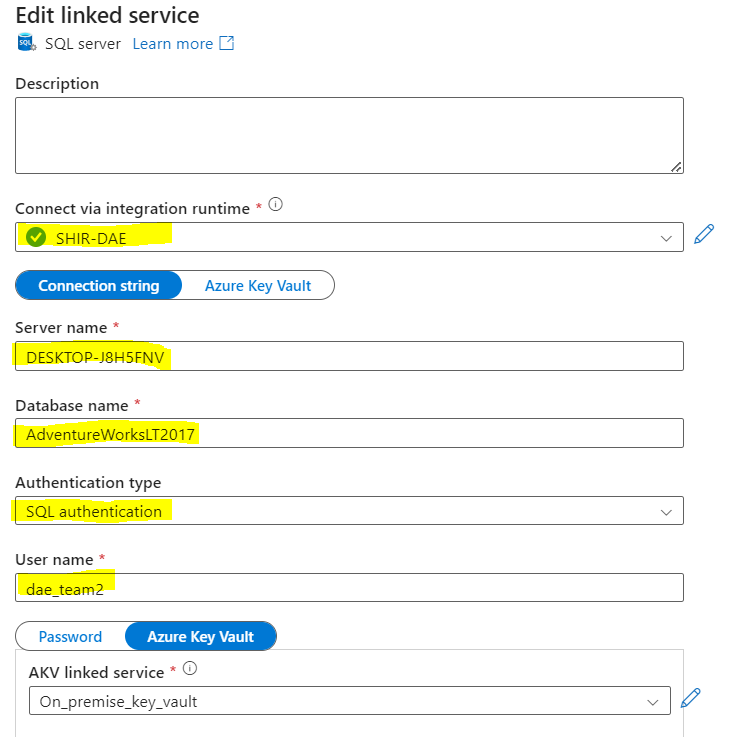
* To begin with the ingestion process, let’s create the sink for the ingestion which is the Azure Data lake gen 2 from the Marketplace.
* In the market place type storage account and click create and configure the storage account
* Now in the ADF, go to Author and click on + to create a new pipeline and name it as Copy data.



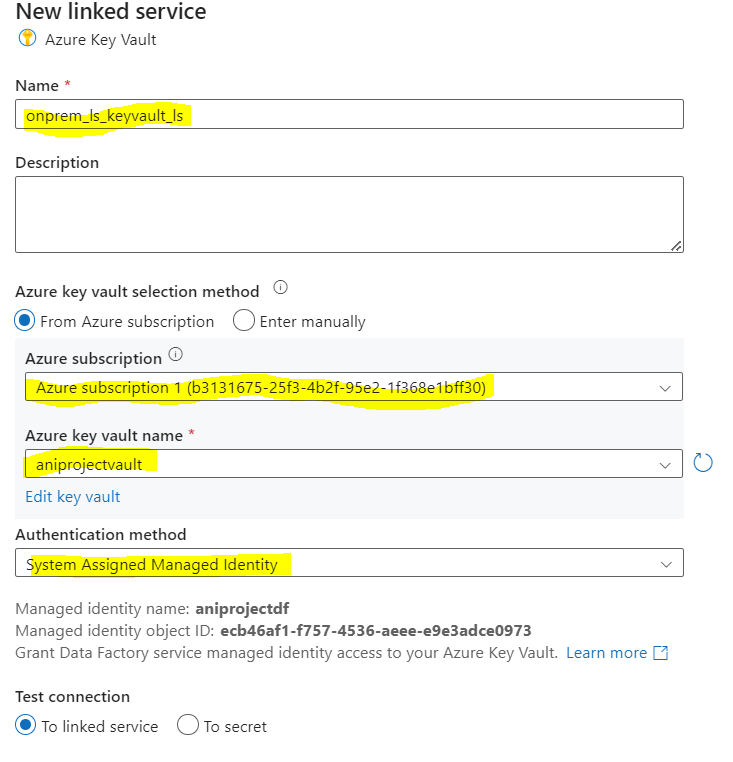
* Create a lookup activity in the pipeline and name it as onprem\_lookup
* In the settings in the data source option, click on + and select SQL server



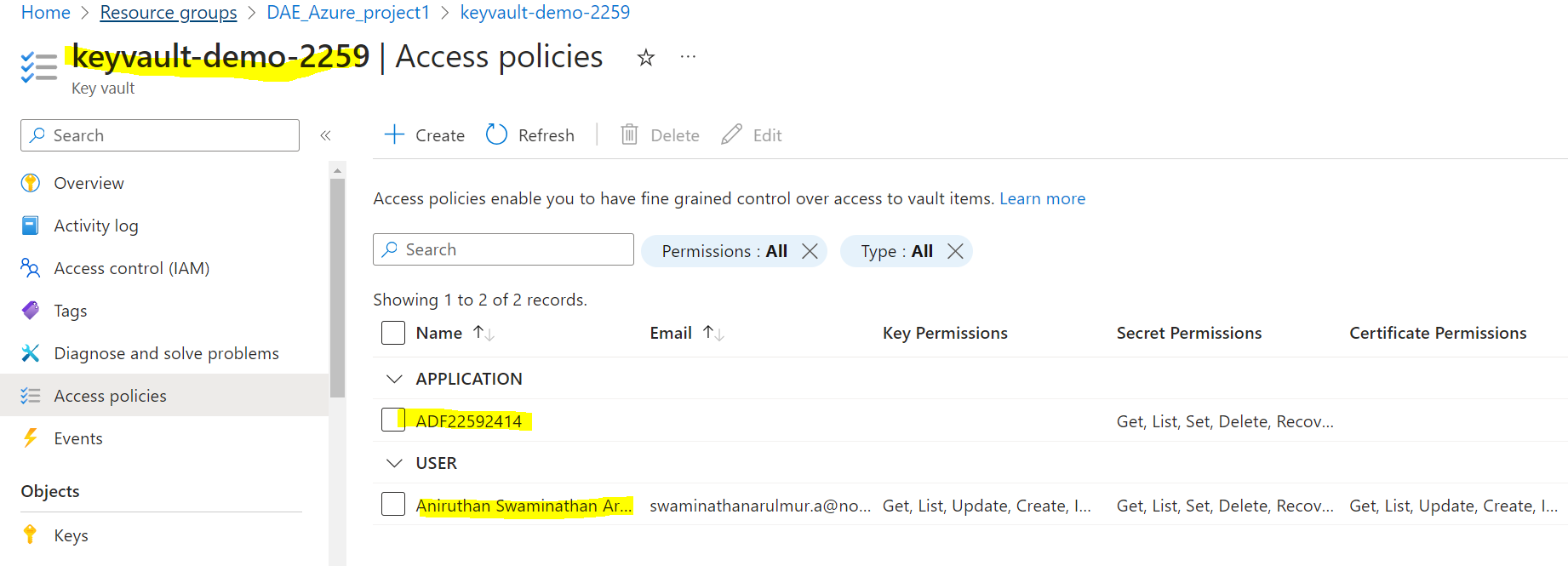
* Nam the source dataset as sqlServerTable2.
* Create a new linked service as sqlServer1.
* Configure the linked service with the details as mentioned in below image

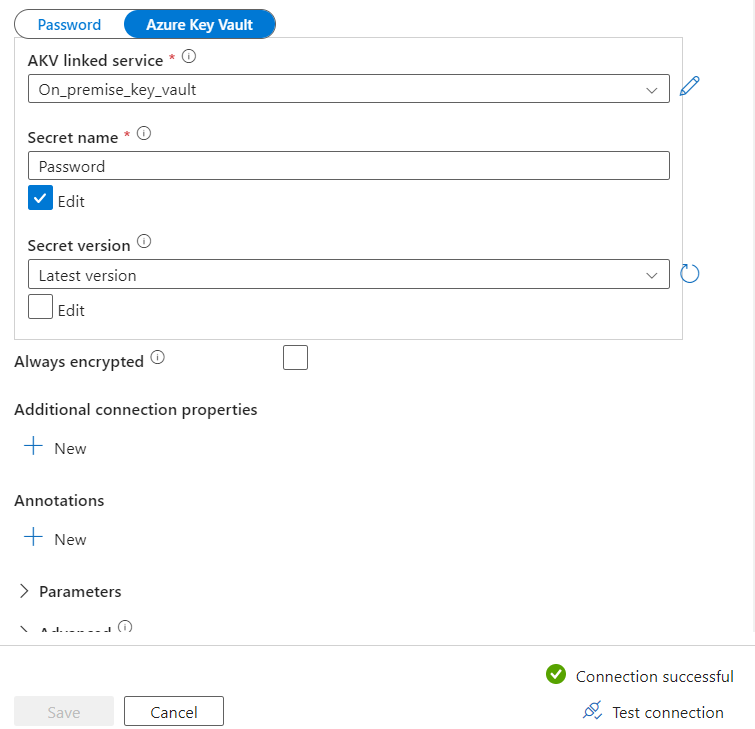


* We are using the keyvault for the passwords. To use the stored secrets in keyvault first we need to create a linked service for the keyvault
* Configure the Azure keyvault linked service as mentioned in the image



* After configuring it, you can test the connection by clicking on the test connection option
* Even after the keyvault linked service is created the sqlServer1 won’t be able to see all the secrets because we haven’t provided ADF with the access to read the keyvault secrets.
* To provide access go to keyvault-> access policies and create a new access for ADF.





* After giving the access, we will be able to establish connection to the on-premise SQL server through the linked service
* There are 10 tables in the on-prem server, to get the names of the Tables and Schema, run the below SQL script on the query section of lookup activity settings

SELECT

s.name AS SchemaName,

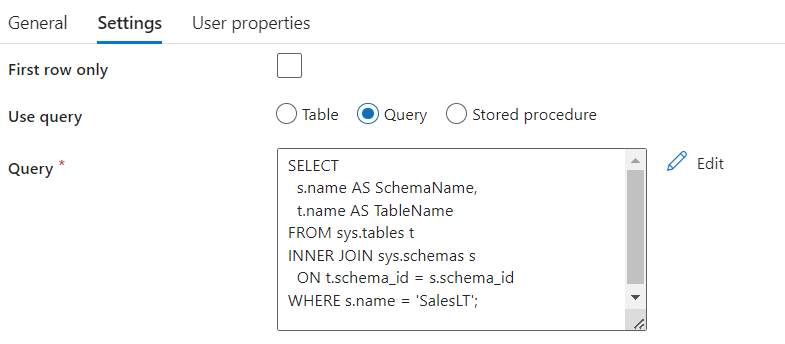
t.name AS TableName

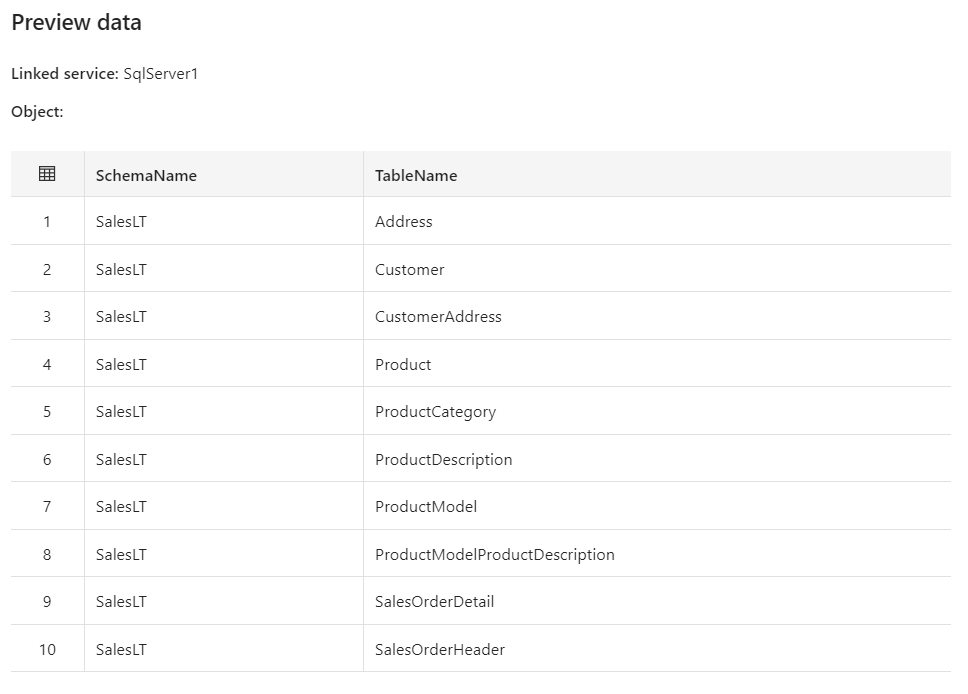
FROM sys.tables t

INNER JOIN sys.schemas s

ON t.schema\_id = s.schema\_id

WHERE s.name = 'SalesLT';





* After running the query, we can click on preview the data to see the above table which shows the Schema Name and Table Name.
* Select the Lookup activity and click on debug option on the top.
* The O/P of the aactivity will look like below

{

"count": 10,

"value": [

{

"SchemaName": "SalesLT",

"TableName": "Address"

},

{

"SchemaName": "SalesLT",

"TableName": "Customer"

},

{

"SchemaName": "SalesLT",

"TableName": "CustomerAddress"

},

{

"SchemaName": "SalesLT",

"TableName": "Product"

},

{

"SchemaName": "SalesLT",

"TableName": "ProductCategory"

},

{

"SchemaName": "SalesLT",

"TableName": "ProductDescription"

},

{

"SchemaName": "SalesLT",

"TableName": "ProductModel"

},

{

"SchemaName": "SalesLT",

"TableName": "ProductModelProductDescription"

},

{

"SchemaName": "SalesLT",

"TableName": "SalesOrderDetail"

},

{

"SchemaName": "SalesLT",

"TableName": "SalesOrderHeader"

}

],

"effectiveIntegrationRuntime": "SHIR-DAE",

"billingReference": {

"activityType": "PipelineActivity",

"billableDuration": [

{

"meterType": "SelfhostedIR",

"duration": 0.016666666666666666,

"unit": "Hours"

}

],

"totalBillableDuration": [

{

"meterType": "SelfhostedIR",

"duration": 0.016666666666666666,

"unit": "Hours"

}

]

},

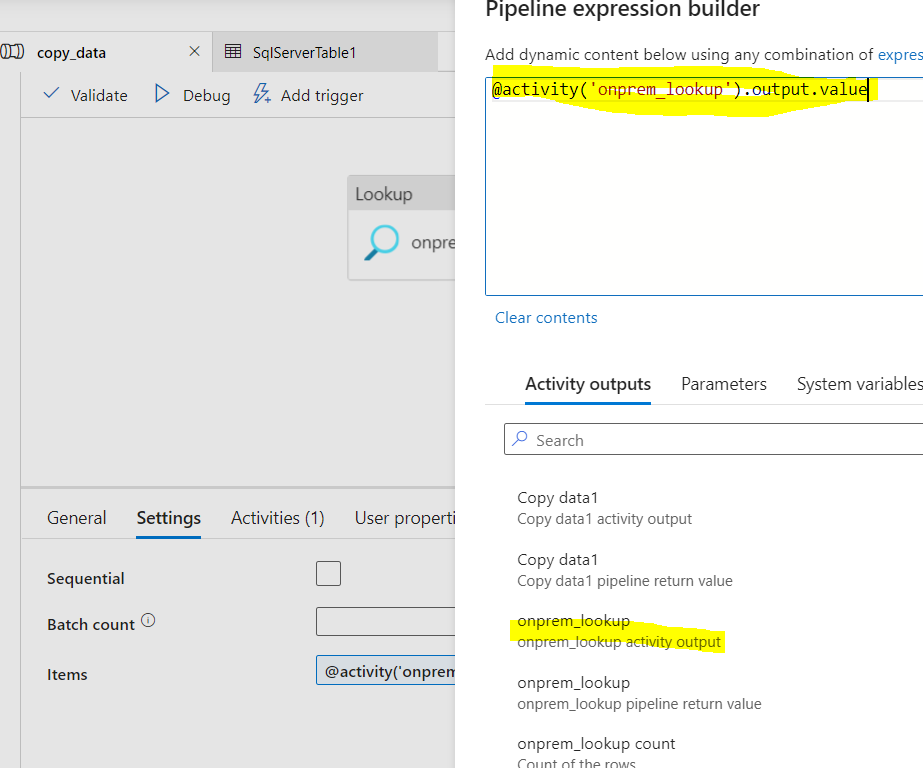
"durationInQueue": {

"integrationRuntimeQueue": 2

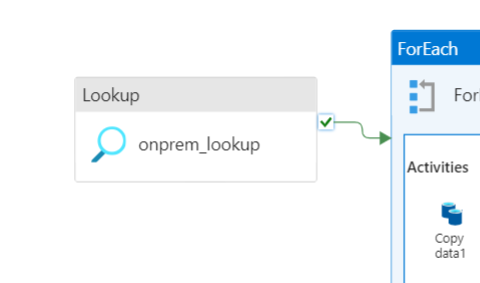
}

}

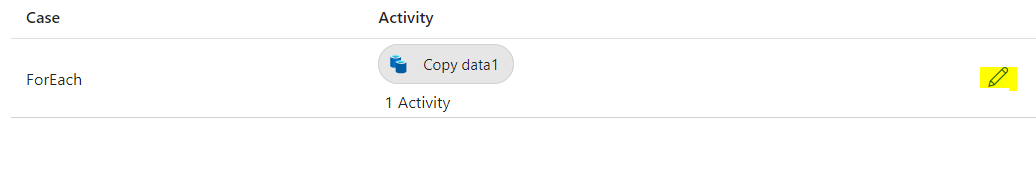
* We will use the above JSON structure output to copy each data table from the source and ingest it into the datalake.
* We will use the next activity named For each which is similar to for each loop in programming language.
* This will go through each item in the o/p list of the lookup activity.



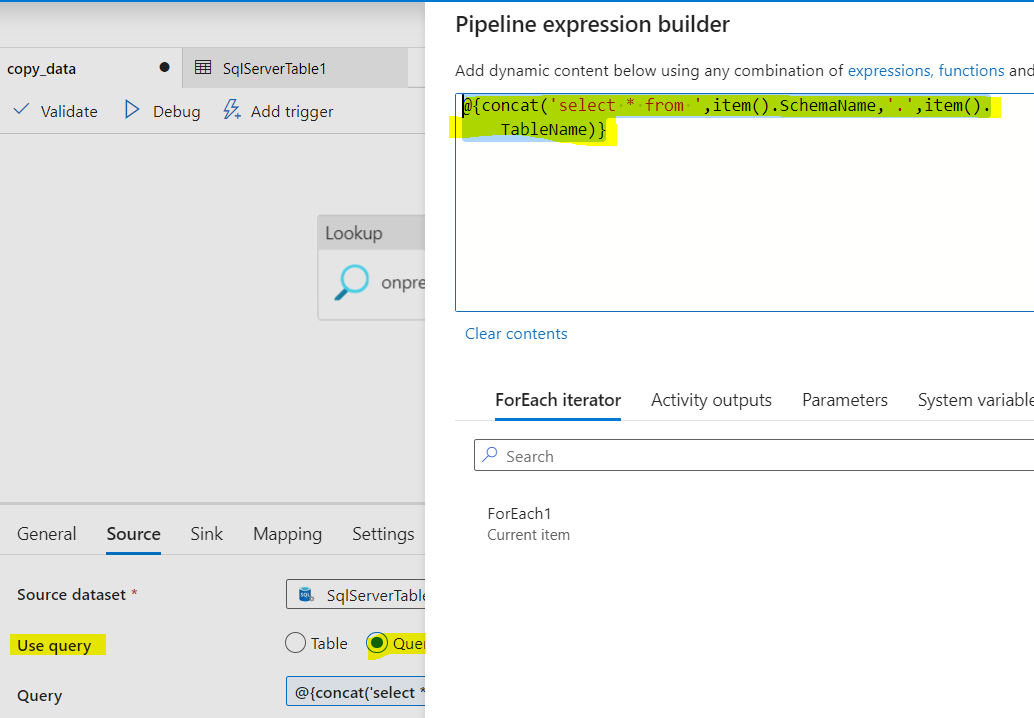
* Go the settings in for each activity-> click on dynamic selection in the items tab and mention the above command as show in the above figure.
* This will allow us to get each schema name and Table name.



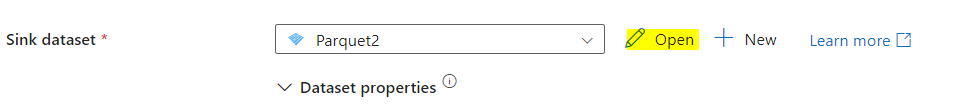
* We need to join the lookup activity to the For Each activity on success.



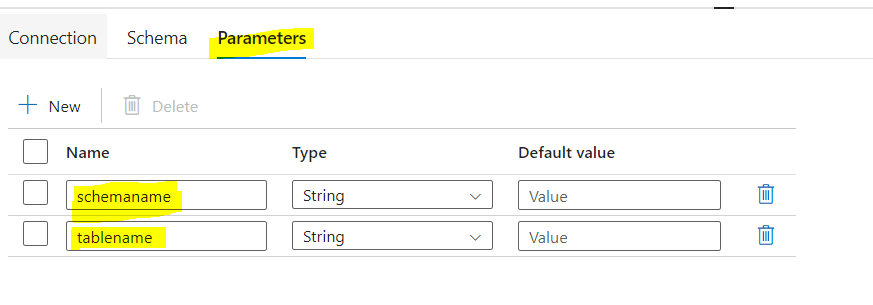
* In the activity tab, click on the tab to get inside the get activity tab.
* Add copy data activity inside the for each activity. We have named it as copy\_data1
* Now select the **copy\_data1** activity, go to source settings. Click on **+ New** and select SQL server. The name is set to **SQLServerTable1** and we use the same linked service created for lookup activity which is **SqlServer1**.
* Now go to the use query settings and set it to query and select the dynamic content option.



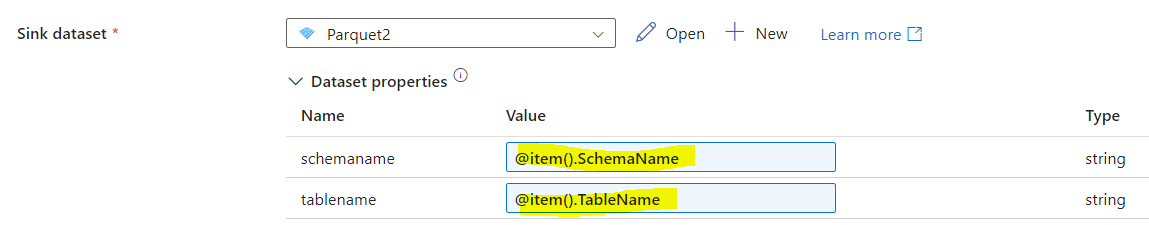
* Use the **@{concat('select \* from',item().SchemaName,'.',item().TableName)}**pipe line builder command.
* This is similar to select \* from schemaName.TableName command but it takes the Schema name and Table name from each item from the lookup activity.
* Now go to the sink settings and create a +New sink. Select Microsoft Azure Data Lake Gen2.
* Named the Sink as Parquet2. Selected the sink folder as bronze in the storage file path and click on save.
* We want to save the tables in the following file paths bronze/Schema/TableName/TableName.parquet
* In order to do this click on open option next to parquet 2



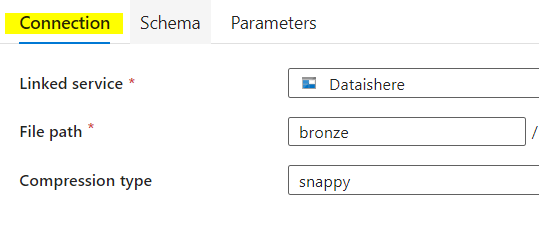
* Go to parameters, and create two new parameters schemaname and tablename



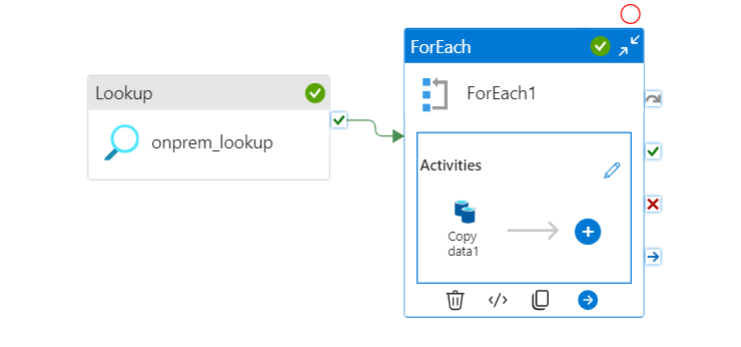
* Now go to sink’s setting and assign value to the created parameters.
* We have to add dynamic content to both the parameters



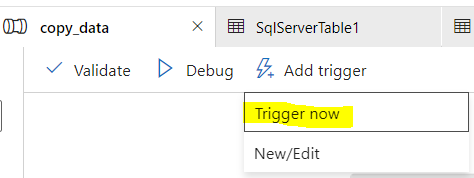
* Assign @item().SchemaName and @item().TableName to the schemaname and tablename parameters.
* These will allow the parameters to dynamically take the lookup activity item values one by one using for each activity.
* Now click on the open option next to the sink dataset and go to connection settings



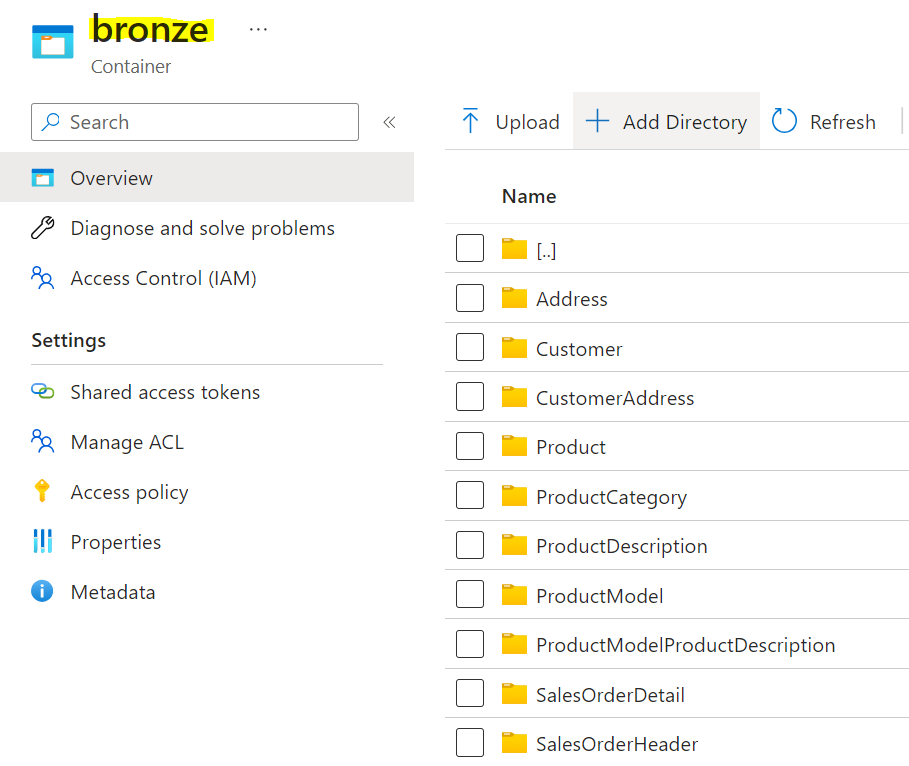
* We need to add dynamic value content to both the directory and file sections in the file path.
* After adding the dynamic content the final file path will look like bronze/@{concat(dataset().schemaname,'/',dataset().tablename)}/ @{concat(dataset().tablename,'.parquet')}
* Ensure publish all the changes. The final pipeline will look like this



* Now click on Add trigger and select trigger now to run the pipeline



* After the successful execution of the pipeline, you can see the tables being ingested into Datalake gen2.



**Additional information:**

Create a self service integration run time to set up the infrastructure which shall enable ADF to move data from the on-pr

* emise server
* Launch ADF-> create a new pipeline-> add a copy data task
* Mention the name of the task, and move the source settings
* Under source settings first create a linked service
* Linked service is used to mention the properties/ strings that are required to establish a connection to a data source
* Then while mentioning the credentials, we use the key vault for the password. We need to create a linked service to the key vault to establish a connection between the copy data activity and the key vault. Then in the key vault access policies settings, we need to provide the right access to the ADF to access the secrets created. Then use the password secret as the password for establishing a connection to the on-premises server.
* Uner lookup settings under use query select query and use the below sql query:

SELECT

s.name AS SchemaName,

t.name AS TableName

FROM sys.tables t

INNER JOIN sys.schemas s

ON t.schema\_id = s.schema\_id

WHERE s.name = 'SalesLT';

* Select the debug button and the query in the lookup activity will be executed

{

"count": 10,

"value": [

{

"SchemaName": "SalesLT",

"TableName": "Address"

},

{

"SchemaName": "SalesLT",

"TableName": "Customer"

},

{

"SchemaName": "SalesLT",

"TableName": "CustomerAddress"

},

{

"SchemaName": "SalesLT",

"TableName": "Product"

},

{

"SchemaName": "SalesLT",

"TableName": "ProductCategory"

},

{

"SchemaName": "SalesLT",

"TableName": "ProductDescription"

},

{

"SchemaName": "SalesLT",

"TableName": "ProductModel"

},

{

"SchemaName": "SalesLT",

"TableName": "ProductModelProductDescription"

},

{

"SchemaName": "SalesLT",

"TableName": "SalesOrderDetail"

},

{

"SchemaName": "SalesLT",

"TableName": "SalesOrderHeader"

}

],

"effectiveIntegrationRuntime": "SHIR-DAE",

"billingReference": {

"activityType": "PipelineActivity",

"billableDuration": [

{

"meterType": "SelfhostedIR",

"duration": 0.016666666666666666,

"unit": "Hours"

}

],

"totalBillableDuration": [

{

"meterType": "SelfhostedIR",

"duration": 0.016666666666666666,

"unit": "Hours"

}

]

},

"durationInQueue": {

"integrationRuntimeQueue": 7

}

}  
  
Then set up for each

* Put copy data activity inside the for each activity  
  @concat{('select \* from', item().SchemaName,'.',item().TableName)}- use this to get the each table from the on premise server
* @{concat(dataset().schemaname,'/',dataset().tablename)}
* @concat{(dataset().tablename,'.parquet')}