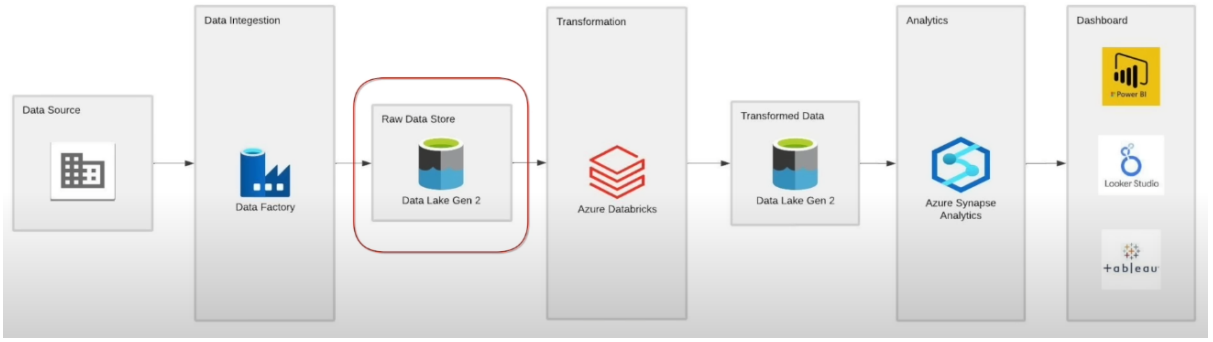
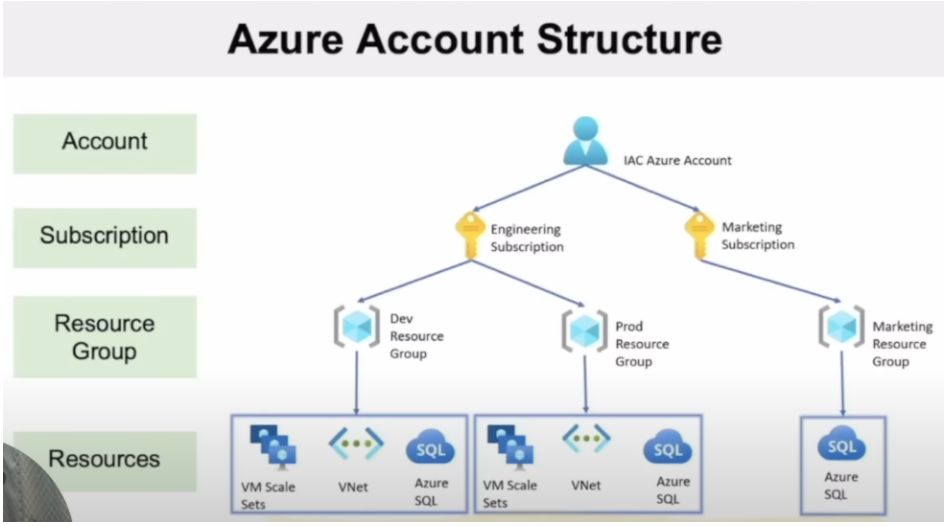
Tokyo Olympic Data Analytics | End-To-End Azure Data Engineering Project





Data Factory

Data integration service that enables you to create, schedule, and manage data pipelines for efficient data movement and transformation between various sources and destinations in Azure and beyond. It simplifies ETL (Extract, Transform, Load) and data integration tasks.

Data Lake Gen 2

Data lake solution that combines the capabilities of a data lake with the power of Azure Blob Storage, allowing you to store and analyze large volumes of structured and unstructured data with enhanced performance, security, and analytics capabilities.

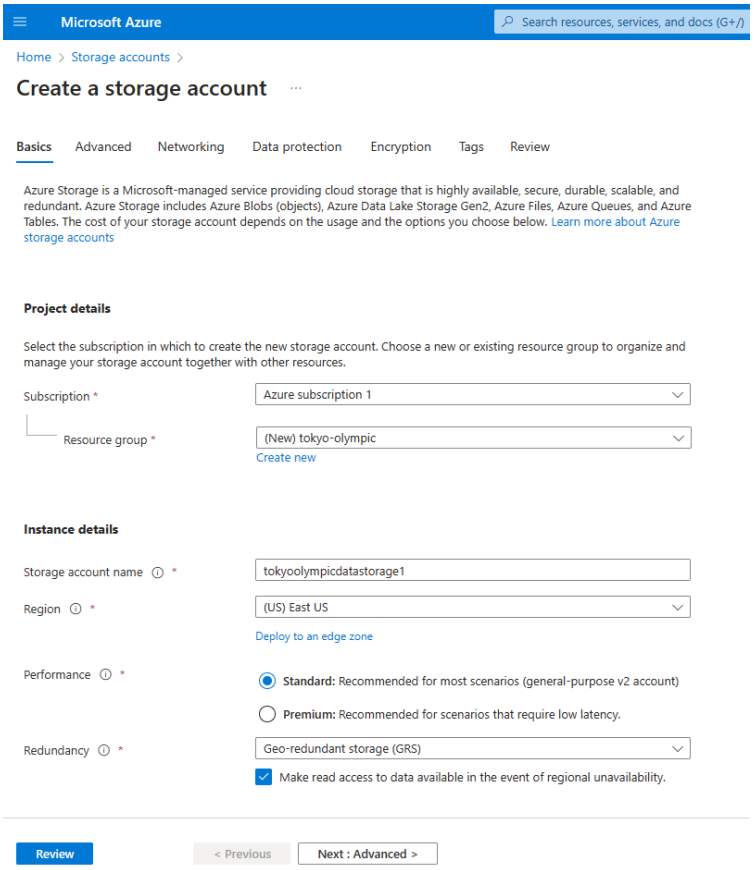
Azure Databricks

Databricks is a unified analytics platform built on top of Apache Spark, designed to help data engineers and data scientists collaborate on big data processing and machine learning tasks. It provides tools for data exploration, data processing, and building machine learning models in a collaborative and scalable environment.

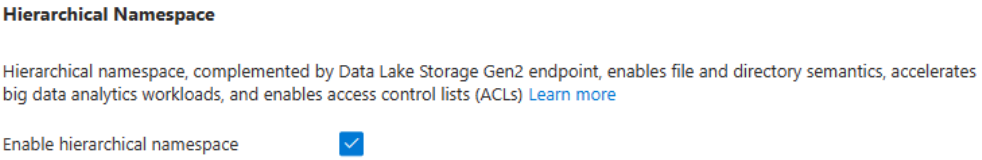
Synapse Analytics

SQL Data Warehouse, is a cloud-based analytics service provided by Microsoft Azure. It combines big data and data warehousing into a single integrated platform, allowing organizations to analyze and process large volumes of data for business intelligence and data analytics purposes.

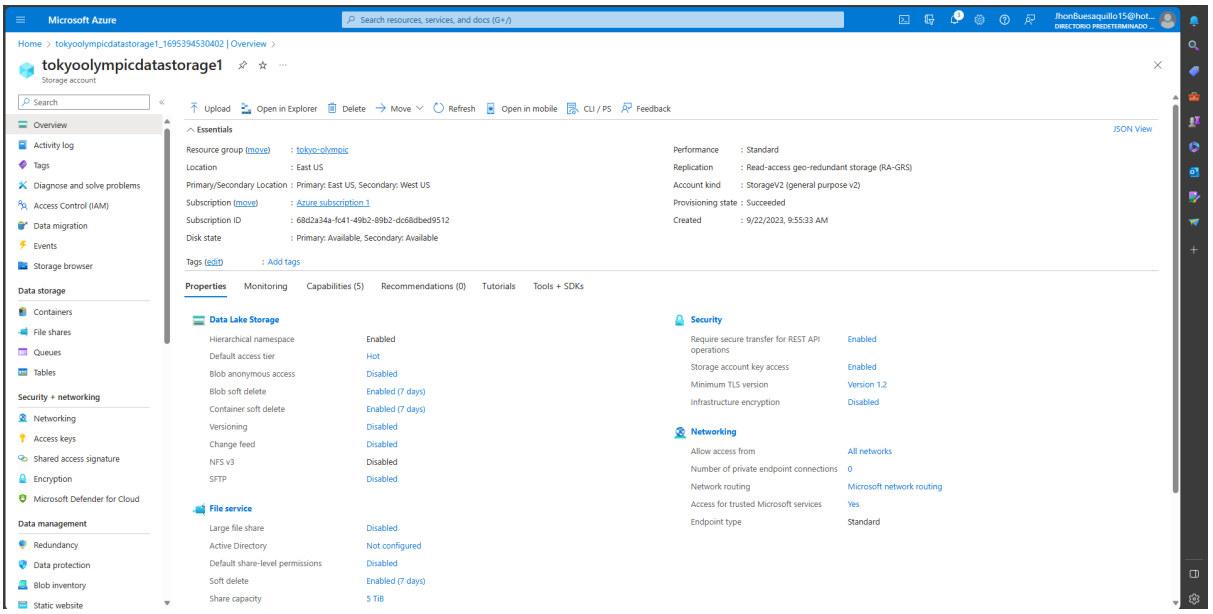
Ingresamos a nuestra cuenta Azure y creamos una storage account.



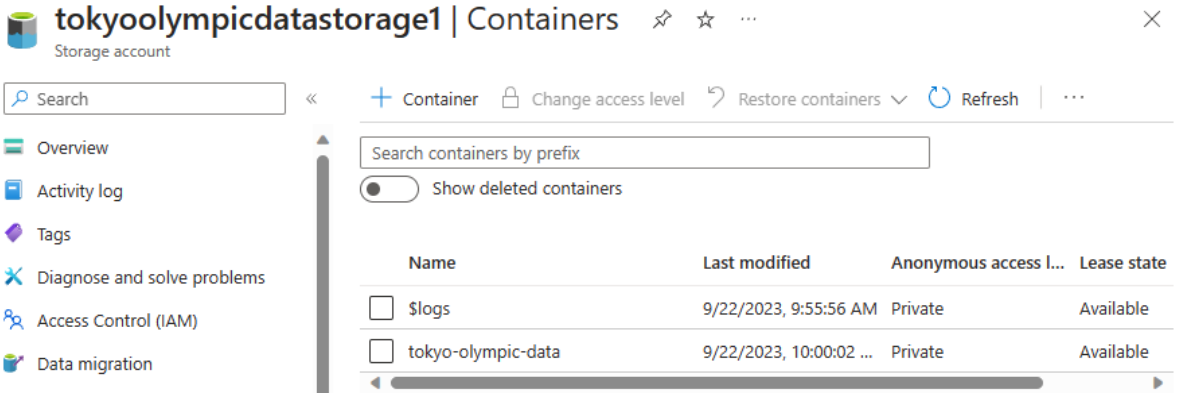
En la sección Advanced habilitamos el Hierarchical Namespace, con el fin de organizar nuestros archivos como se hace normalmente en nuestro directorio local.



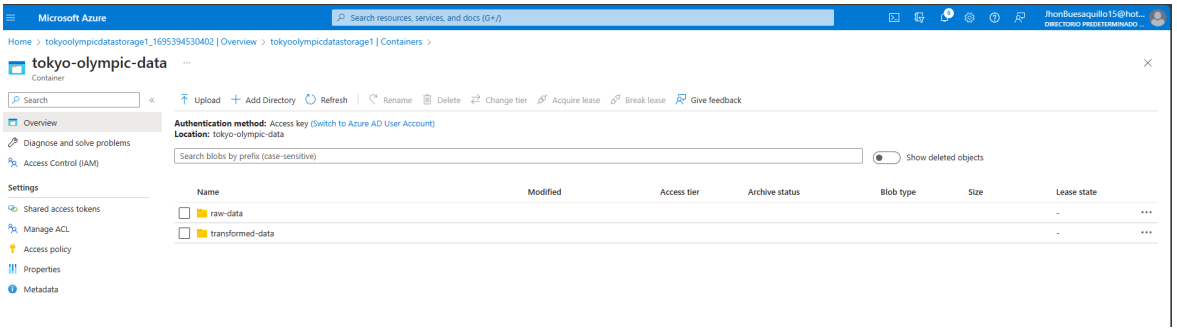
Los demás valores no los modificamos. Finalmente creamos la cuenta de almacenamiento.



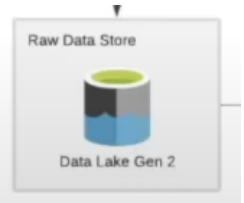
En este laboratorio nos concentraremos en la sección de Containers. Creamos uno nuevo llamado tokyo-olympic-data.



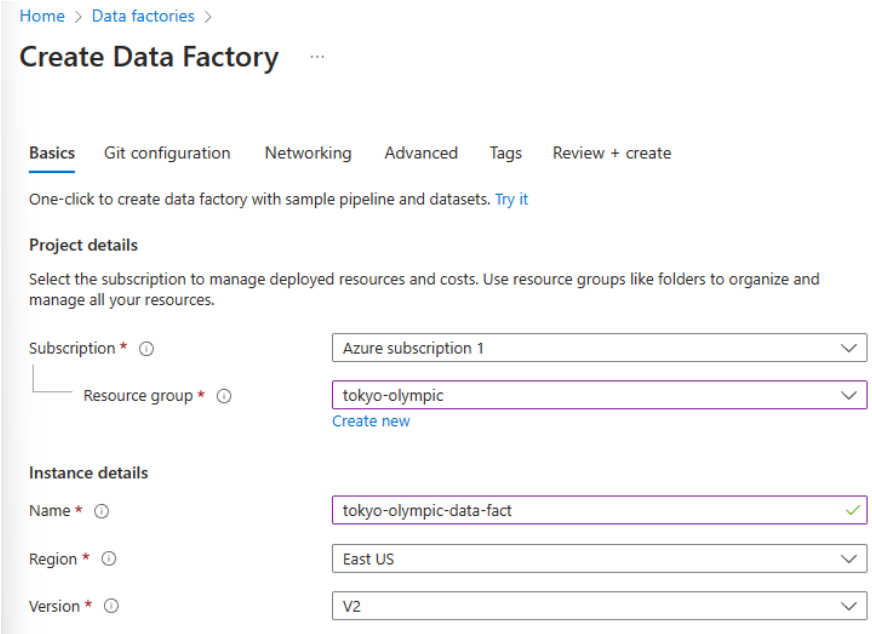
Dentro del container creamos dos directorios para almacenar los datos.



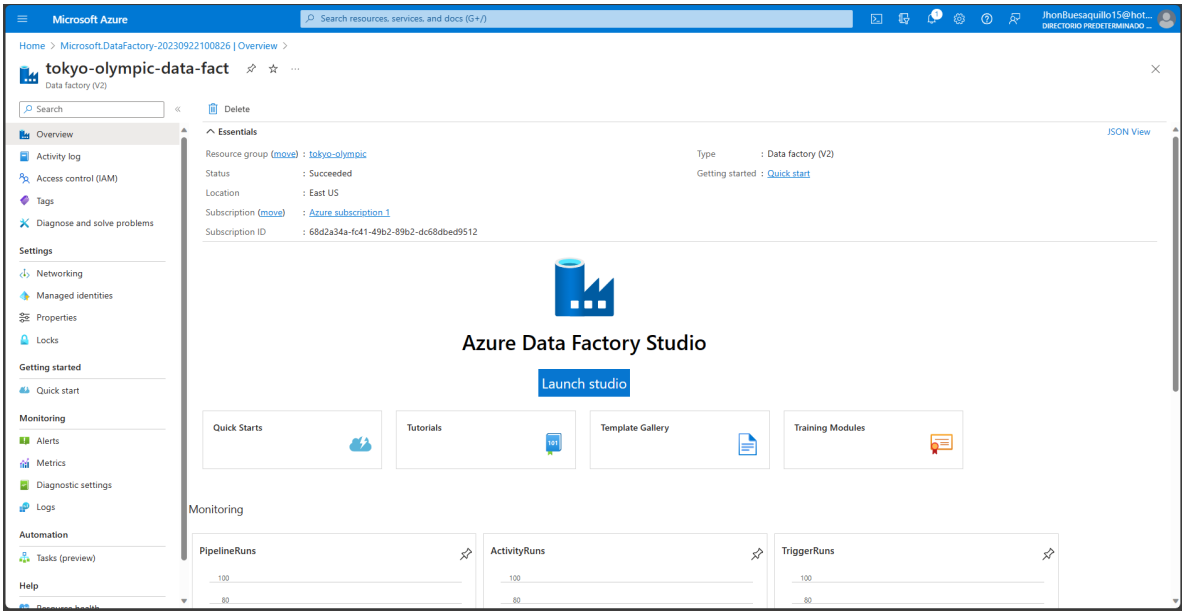
Lo que hemos hecho hasta ahora se encuentra abarcado en esta parte del esquema.



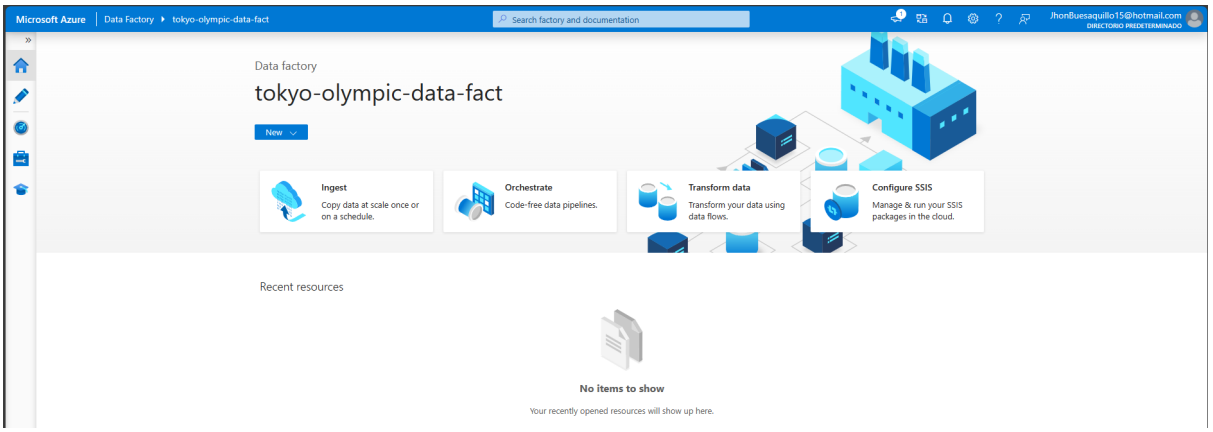
Ahora, ingresamos al servicio de Data factories y creamos una nueva Data Factory.



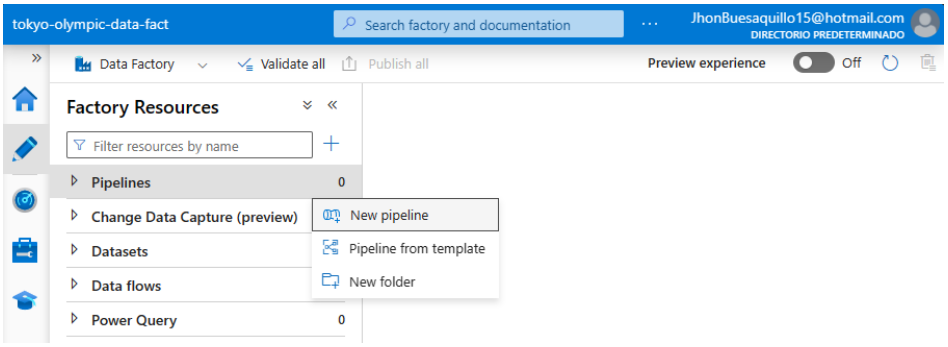
El servicio creado se puede ver a continuación.



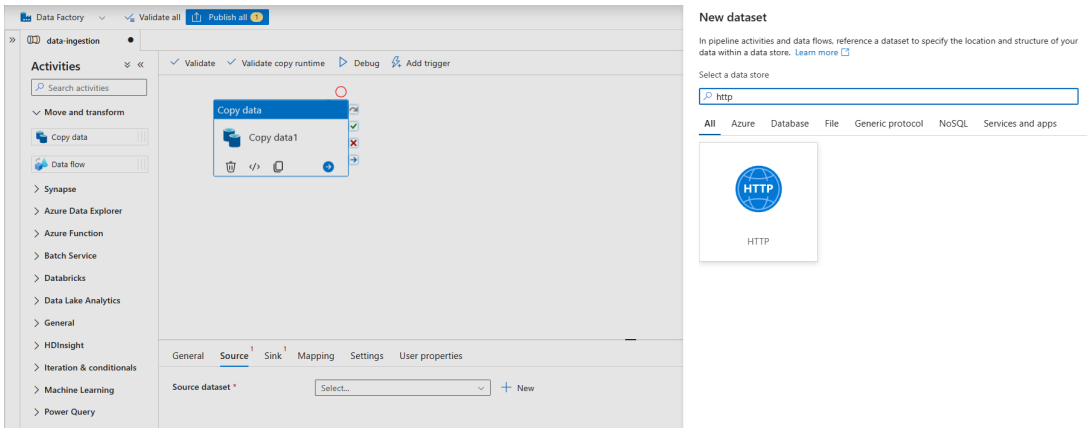
Ingresamos a Data Factory Studio.



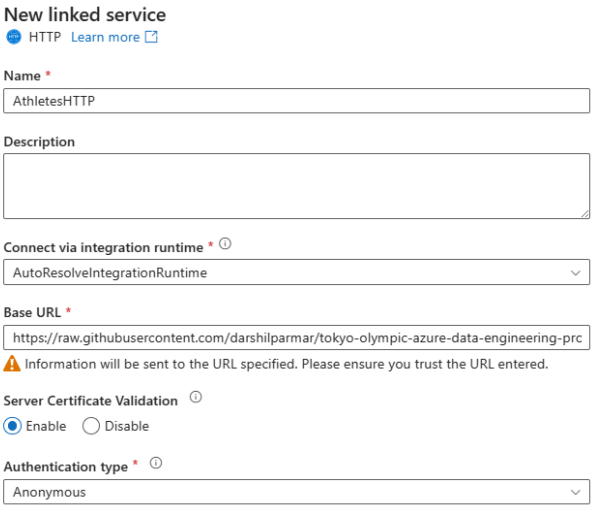
Vamos a la sección Author y creamos un nuevo pipeline.



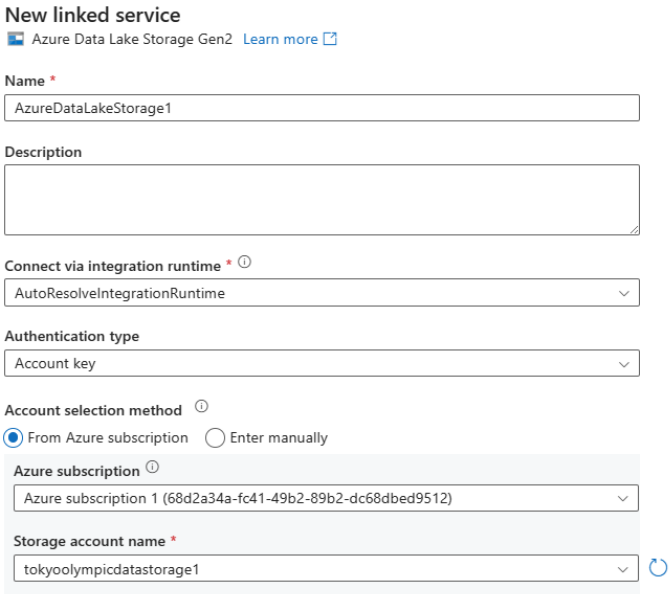
Agregamos un bloque de Copy data. En la sección *source* seleccionamos HTTP como fuente. Luego, en formato, elegimos DelimitedText, ya que nuestros archivos son .csv.

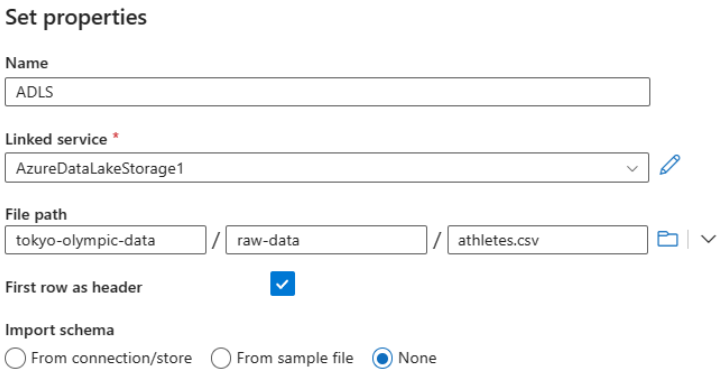


En la configuración, ingresamos la url del dataset y una autenticación anónima, ya que es información abierta al público. Finalmente queda creado este vínculo. Podemos ver la data con la opción *Preview data*.

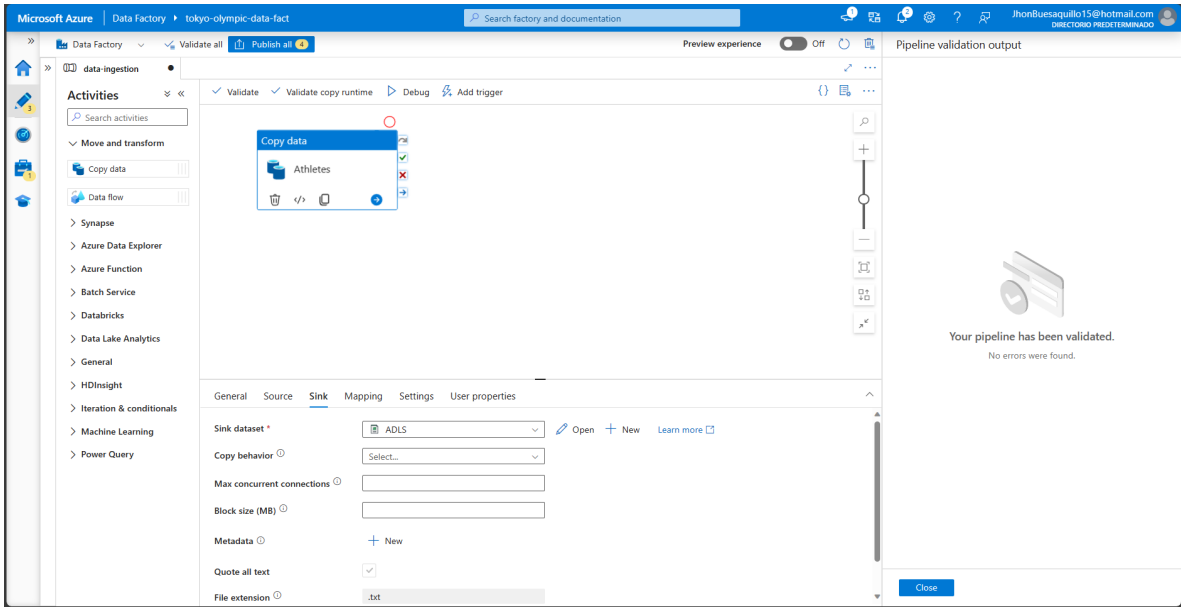


Ahora, para cargar nuestros datos en la storage account ya creada, vamos a la sección sink y seleccionamos Azure Data Lake Storage Len2. Luego, en formato, elegimos DelimitedText.

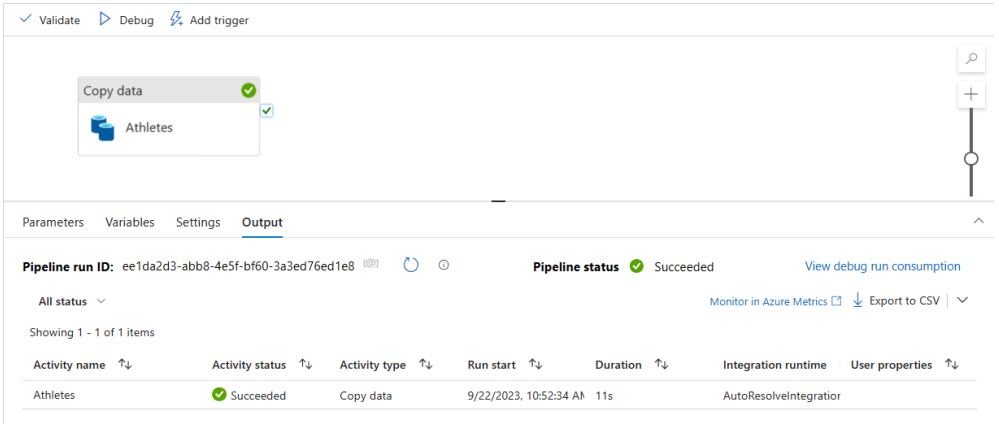


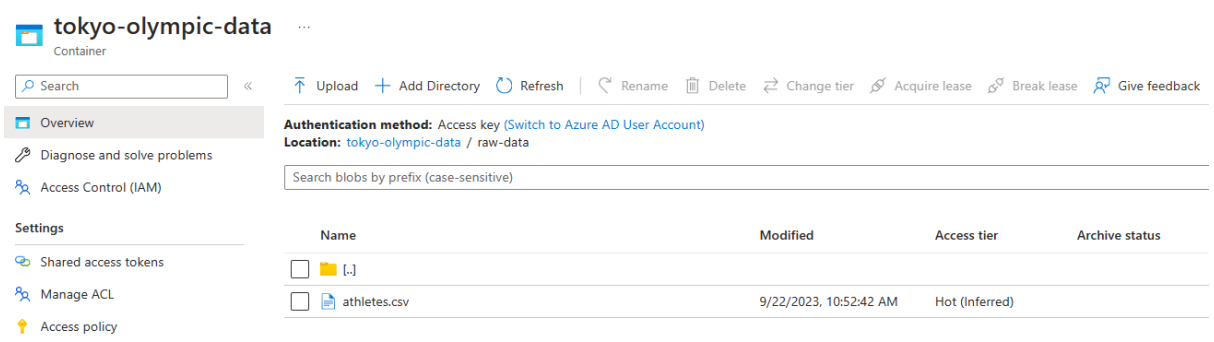


Validamos nuestro pipeline.

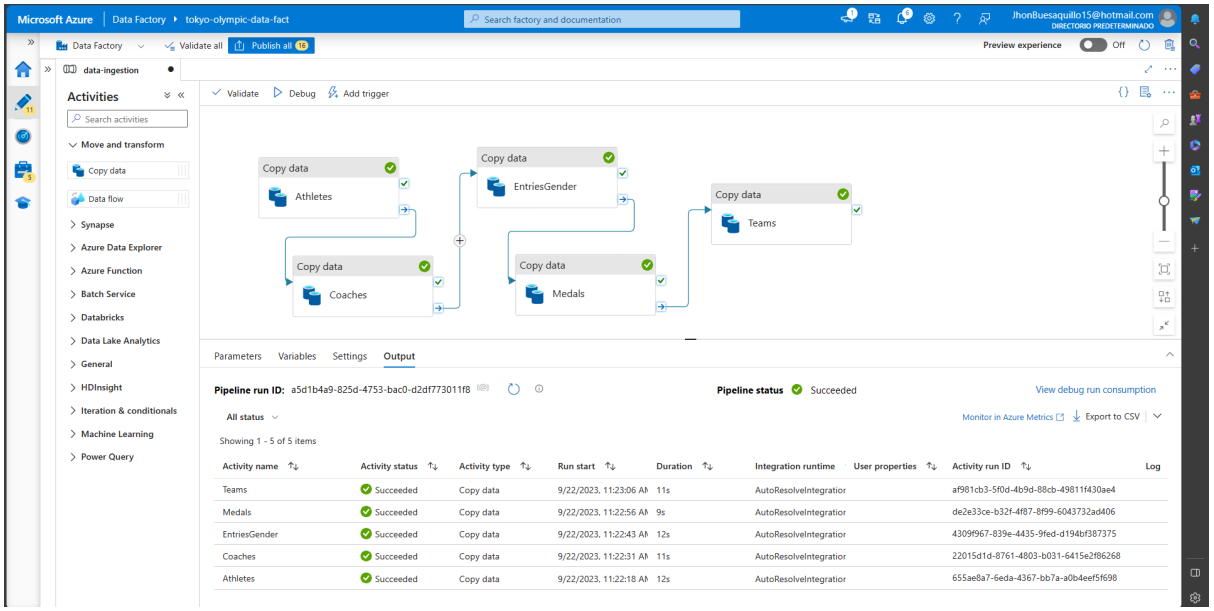


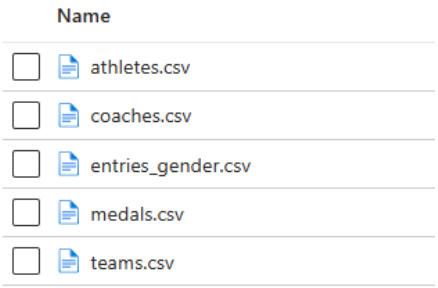
Hacemos el debug y verificamos que todo se ejecute correctamente.



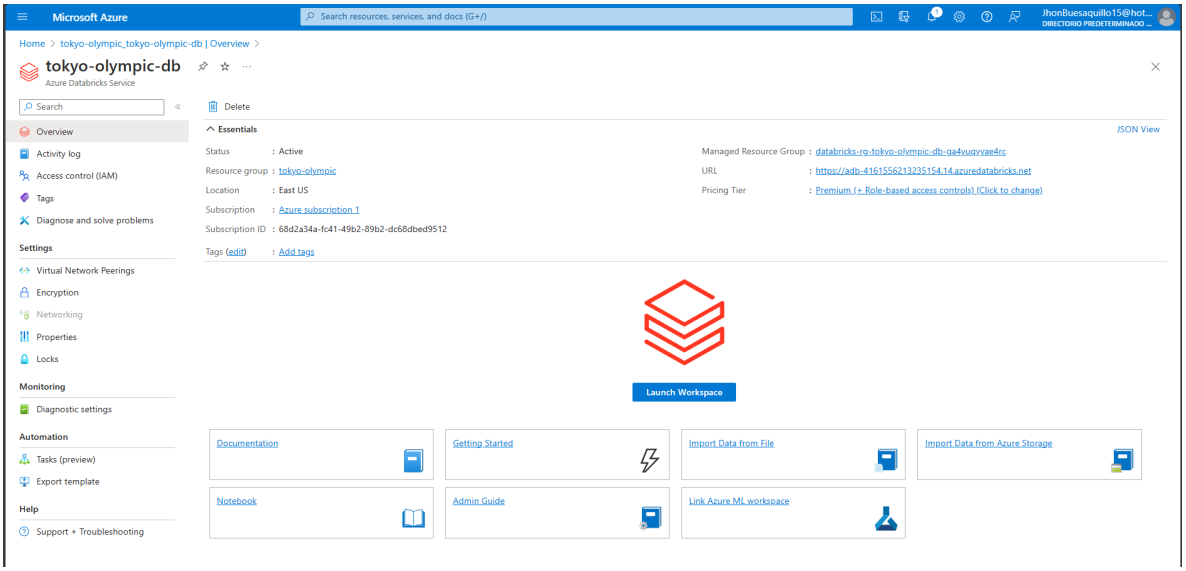


Realizamos el mismo proceso para los demás datasets, reutilizando el sink de Azure Data Lake Storage Len2 para todos.

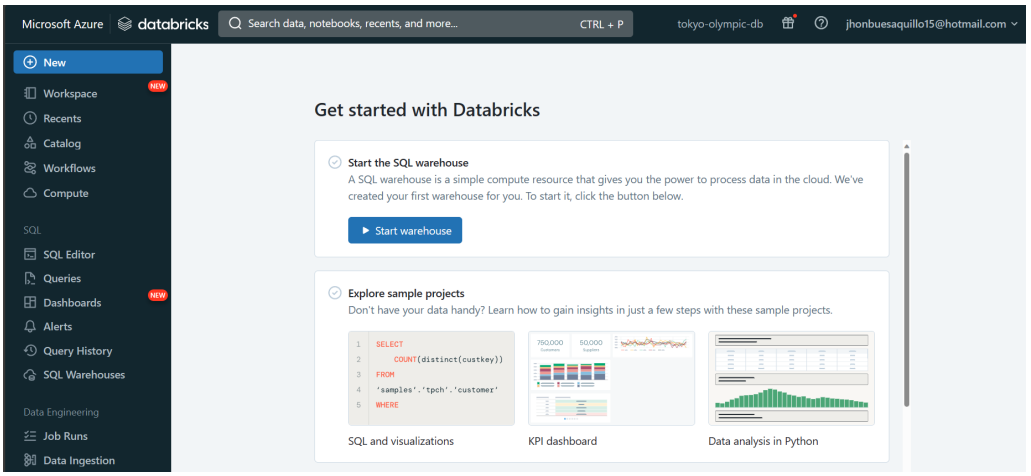




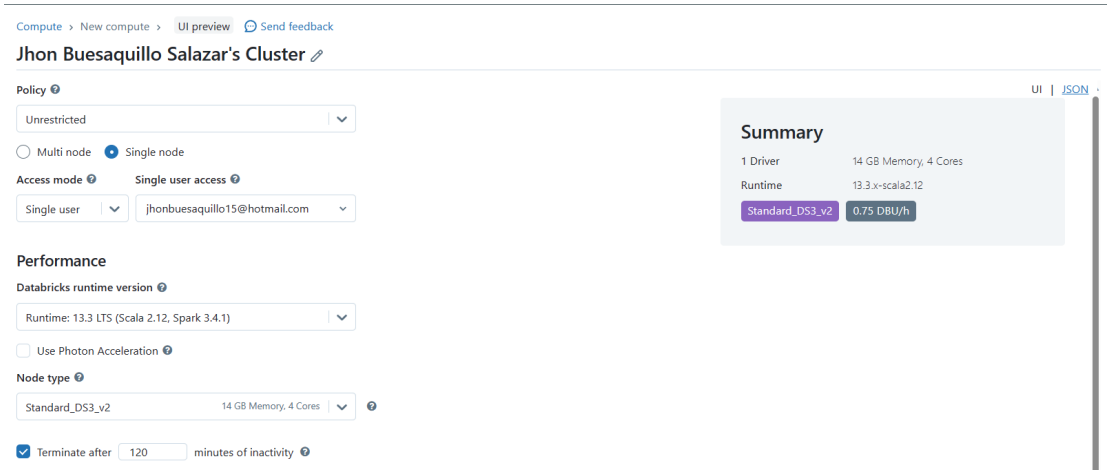
Continuando con la arquitectura, ahora creamos un servicio Azure Databricks.



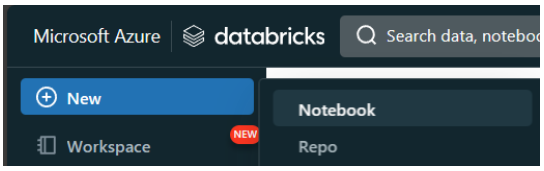
Lanzamos el workspace.



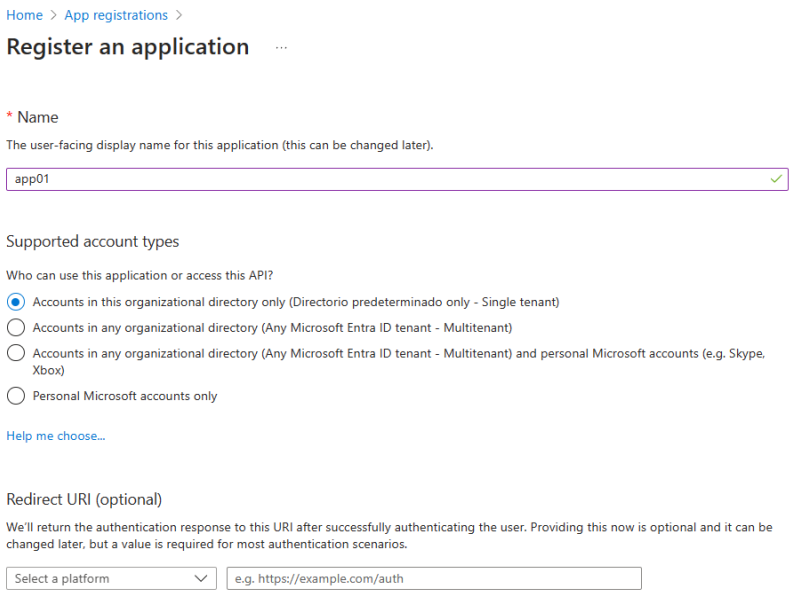
Creamos una instancia Compute para poder correr Spark.



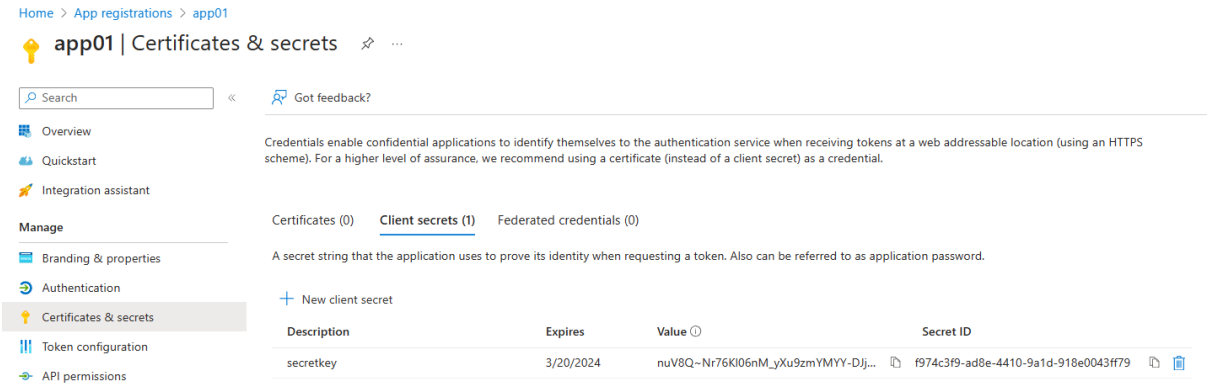
Luego, creamos un nuevo notebook para desarrollar nuestro código Spark.



Creamos un registro de app para poder hacer una conexión entre Azure Databricks y ADLS.



Creamos un nuevo secret.

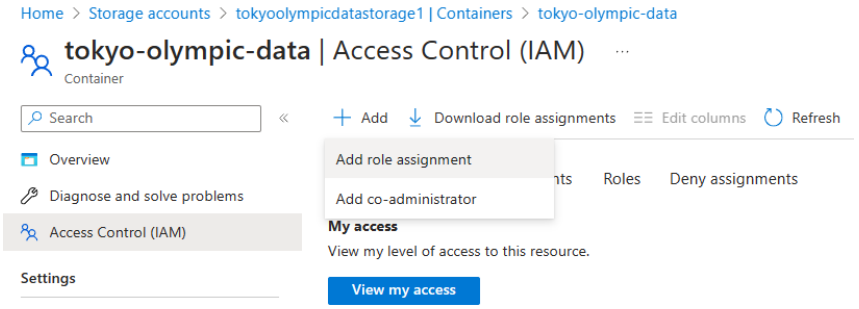


Tomamos el Client ID y Tenant ID del registro, como también el Value del secret para crear la conexión.

```
configs = {"fs.azure.account.auth.type": "OAuth",
"fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
"fs.azure.account.oauth2.client.id": "6ea29bf7-ef62-4955-a342-1cdbeae07f74",
"fs.azure.account.oauth2.client.secret": 'nuV8Q~Nr76Kl06nM_yXu9zmYMYY-DJjmbHI6ebbh',
"fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/22b59d6a-4ebf-4603-a1c8-65dfc64b4f04/oauth2/token"}
```

Estos valores pueden ser protegidos usando un Key Vault de Azure, para tener en cuenta en un ambiente empresarial.

La conexión ahora nos exige crear un rol en el container.



Asignamos la configuración correspondiente.

Storage Blob Data Contributor

☐	app01	App	Storage Blob Data Contributor

Realizamos la transformación de los datos con Pyspark. Una vez hecho esto, enviamos estos archivos al directorio transformed-data de nuestro container.

Location: tokyo-olympic-data / transformed-data / athletes

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier
[..]		
_committed_877905915125892948	9/23/2023, 10:39:22 ...	Hot (Inferred)
_started_877905915125892948	9/23/2023, 10:39:22 ...	Hot (Inferred)
_SUCCESS	9/23/2023, 10:39:22 ...	Hot (Inferred)
part-00000-tid-877905915125892948-a881b9a0-c39b-4a22-92dc-412257dea06...	9/23/2023, 10:39:22 ...	Hot (Inferred)

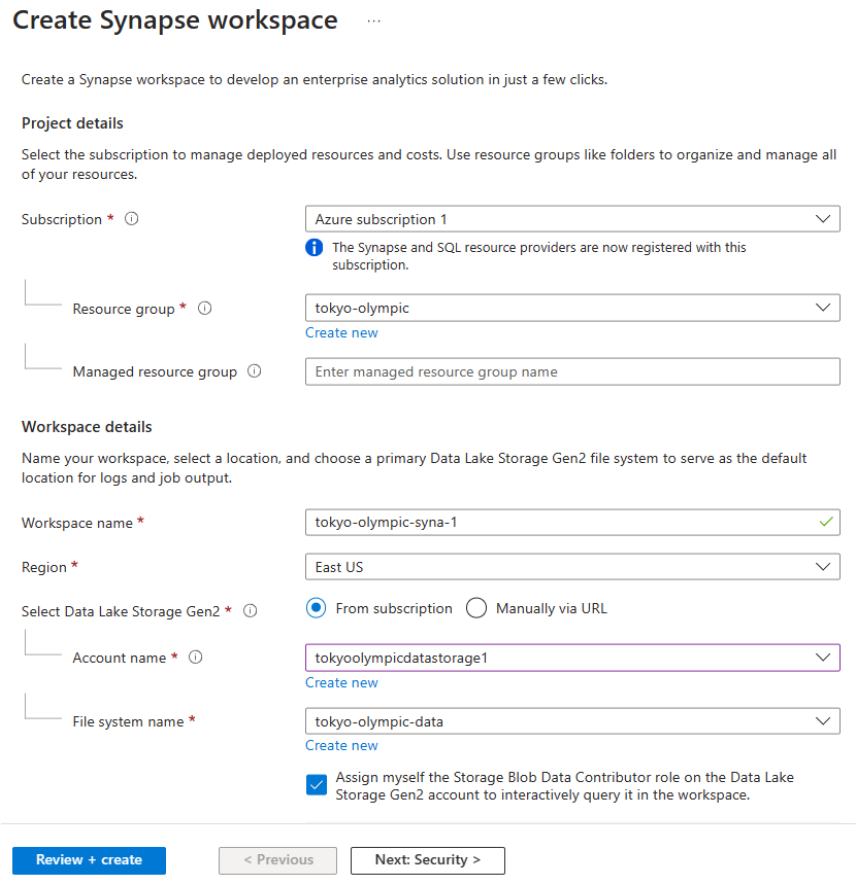
Location: tokyo-olympic-data / transformed-data

Search blobs by prefix (case-sensitive)

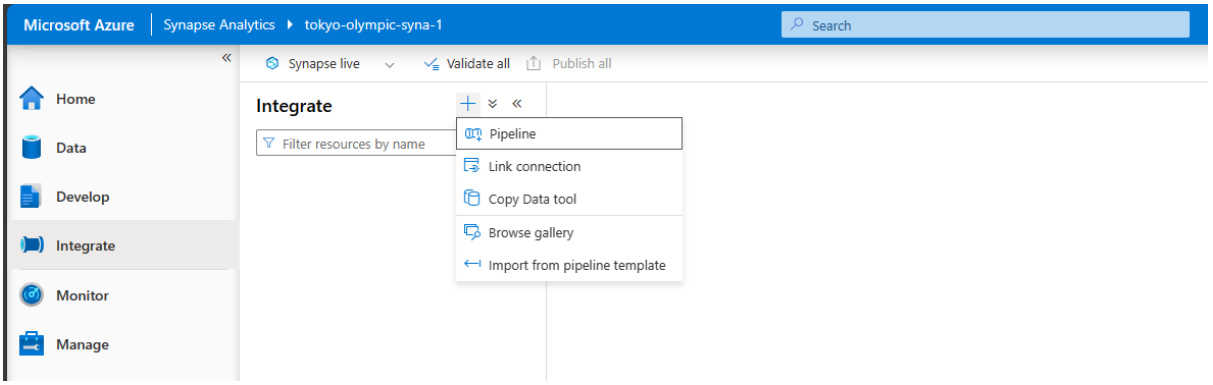
Show deleted objects

Name	Modified
athletes	
coaches	
entries_gender	
medals	
teams	

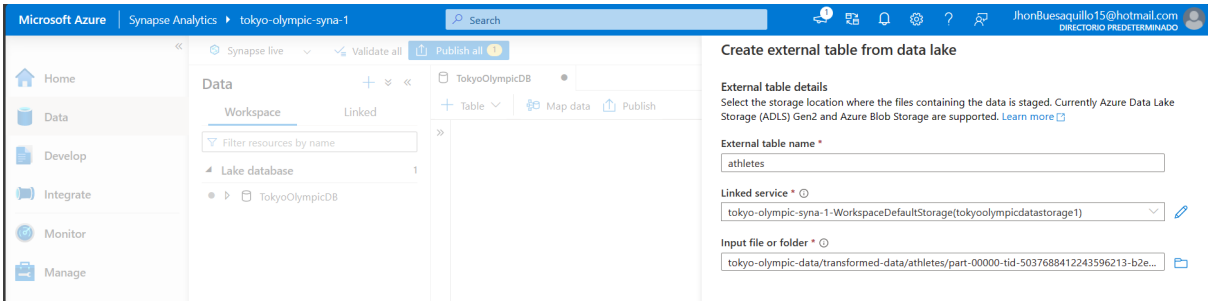
Como siguiente paso, usaremos los datos transformados en **Azure Synapse Analytics**.

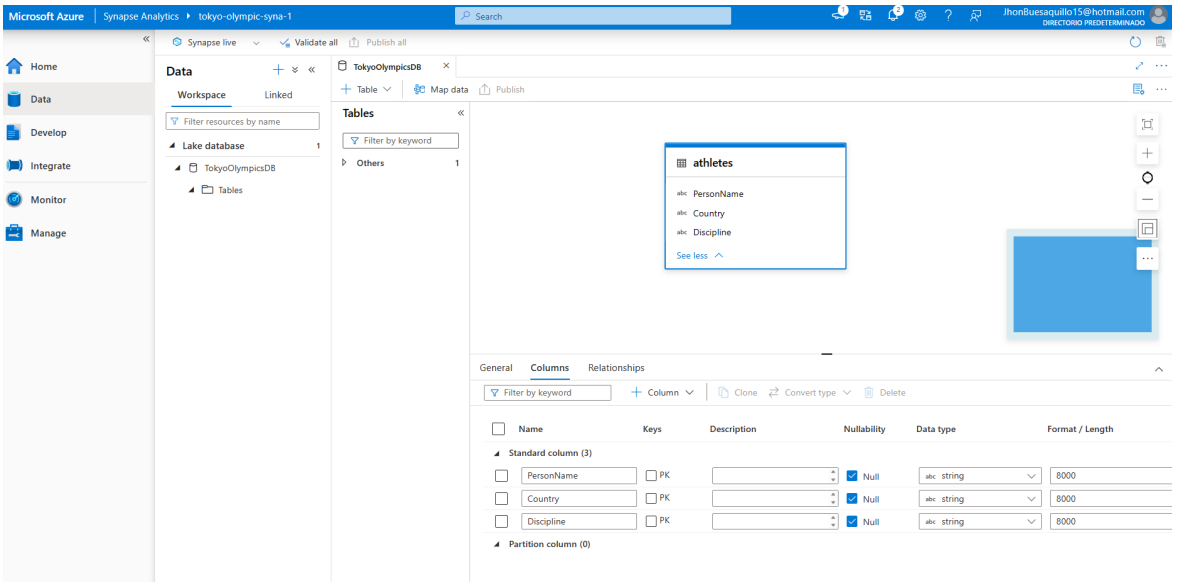


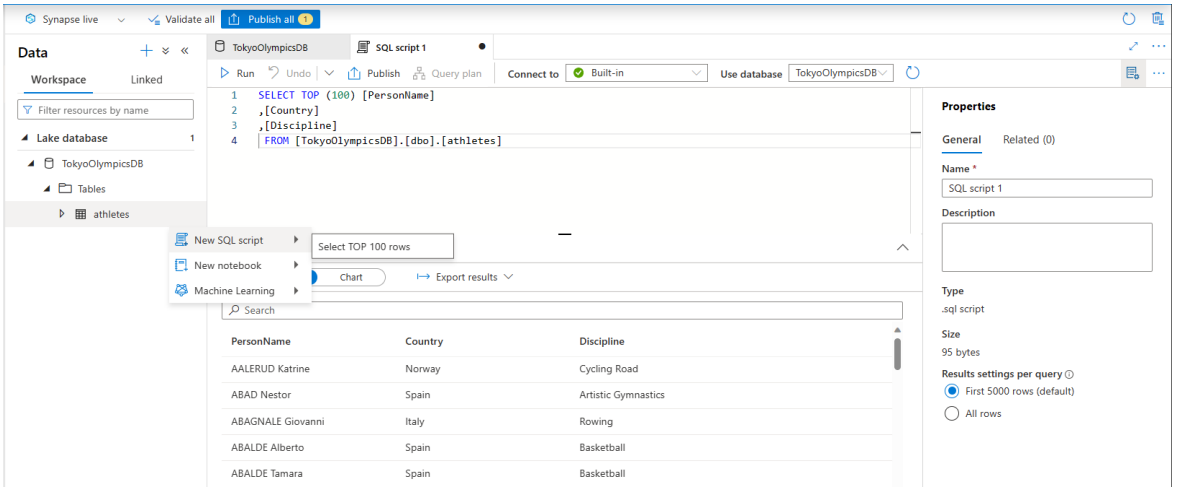
Nos damos cuenta que el servicio Synapse nos ofrece todo lo que hicimos anteriormente. En un caso real se podría usar solo este servicio.

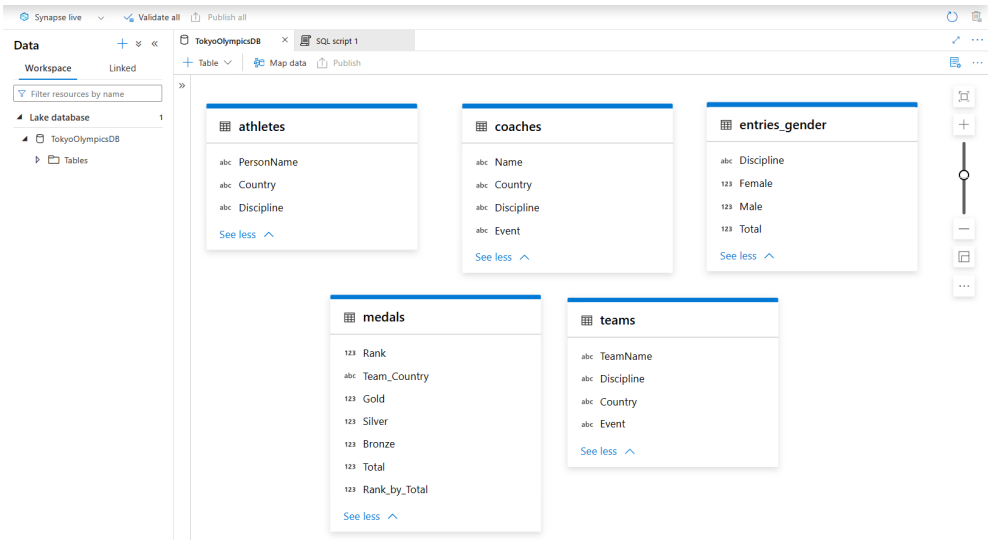


Creamos un nuevo workspace de Data (lake database). Vamos agregando las tablas provenientes de transformed-data




Microsoft Azure
Synapse Analytics
tokyo-olympic-syna-1
Search
Home
Data
Develop
Integrate
Monitor
Manage
Synapse live
Validate all
Publish all
Workspace
Linked
Filter resources by name
Lake database
TokyoOlympicsDB
Tables
Table
Map data
Publish
Filter by keyword
Others
athletes
PersonName
Country
Discipline
See less
General
Columns
Relationships
Column
Clone
Convert type
Delete
Name
Keys
Description
Nullability
Data type
Format / Length
Standard column (3)
PersonName
Country
Discipline
PK
Null
string
8000
Partition column (0)


Synapse live
Validate all
Publish all
Data
Workspace
Linked
Filter resources by name
Lake database
TokyoOlympicsDB
Tables
athletes
New SQL script
New notebook
Machine Learning
SQL script 1
Run
Undo
Publish
Query plan
Connect to
Built-in
Use database
TokyoOlympicsDB
SELECT TOP (100) [PersonName]
,[Country]
,[Discipline]
FROM [TokyoOlympicsDB].[dbo].[athletes]
Select TOP 100 rows
Chart
Export results
Search
PersonName
Country
Discipline
AALERUD Katrine
Norway
Cycling Road
ABAD Nestor
Spain
Artistic Gymnastics
ABAGNALE Giovanni
Italy
Rowing
ABALDE Alberto
Spain
Basketball
ABALDE Tamara
Spain
Basketball
Properties
General
Related (0)
Name *
SQL script 1
Description
Type
.sql script
Size
95 bytes
Results settings per query
First 5000 rows (default)
All rows


Synapse live
Validate all
Publish all
Data
Workspace
Linked
Filter resources by name
Lake database
TokyoOlympicsDB
Tables
SQL script 1
Table
Map data
Publish
athletes
PersonName
Country
Discipline
See less
coaches
Name
Country
Discipline
Event
See less
entries_gender
Discipline
Female
Male
Total
See less
medals
Rank
Team_Country
Gold
Silver
Bronze
Total
Rank_by_Total
See less
teams
TeamName
Discipline
Country
Event
See less

Usamos el script para probar algunos queries.

TokyoOlympicsDB | SQL script 1

Run | Undo | Publish | Query plan | Connect to: Built-in | Use database: TokyoOlympicsDB

```sql
SELECT
    Discipline,
    AVG(Female) Avg_Female,
    AVG(Male) Avg_Male
FROM [TokyoOlympicsDB].[dbo].[entries_gender]
GROUP BY Discipline;
```

Results | Messages

View: Table | Chart | Export results

Search

Discipline	Avg_Female	Avg_Male
3x3 Basketball	32	32
Archery	64	64
Artistic Gymnastics	98	98
Artistic Swimming	105	0
Athletics	969	1072
Badminton	86	87

También nos ofrece gráficas para visualizar mejor los datos.