

# Rozpoznanie narzędzi raportowych w Azure

Arkadiusz Chmura, Bartosz Ciesielski, Bartosz Przybył

## 1 Wprowadzenie

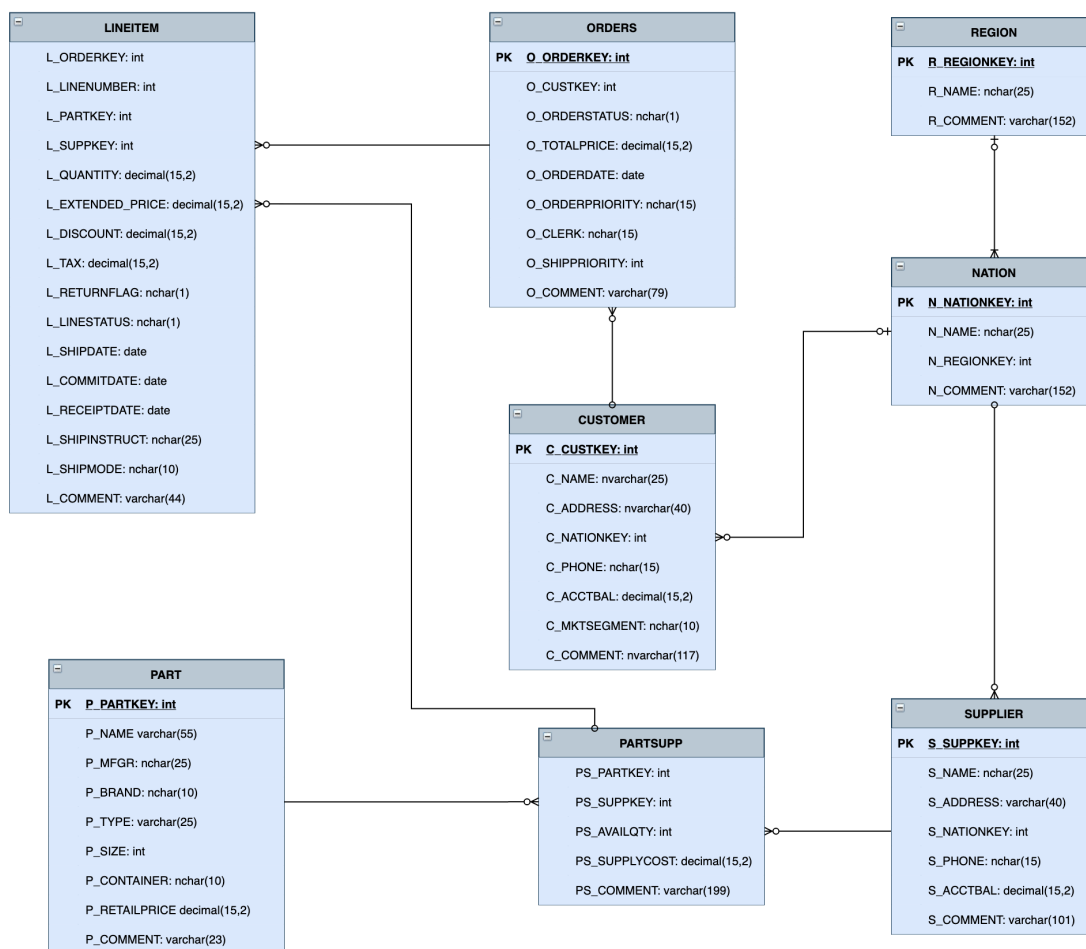
Poniższy dokument stanowi udokumentowanie pracy nad projektem wykonywanym w ramach przedmiotu *Hurtownie danych i przetwarzanie analityczne* we współpracy z firmą *Santander*.

## 2 Cel projektu

Celem projektu jest weryfikacja natywnych narzędzi raportowych dostępnych w chmurze *Microsoft Azure* i porównanie ich pod względem funkcjonalnym z narzędziami wykorzystywanymi w obecnym procesie raportowym.

Efektami końcowymi projektu będzie propozycja architektury nowego środowiska raportowego w chmurze oraz prezentacja wymaganych funkcjonalności na przykładzie.

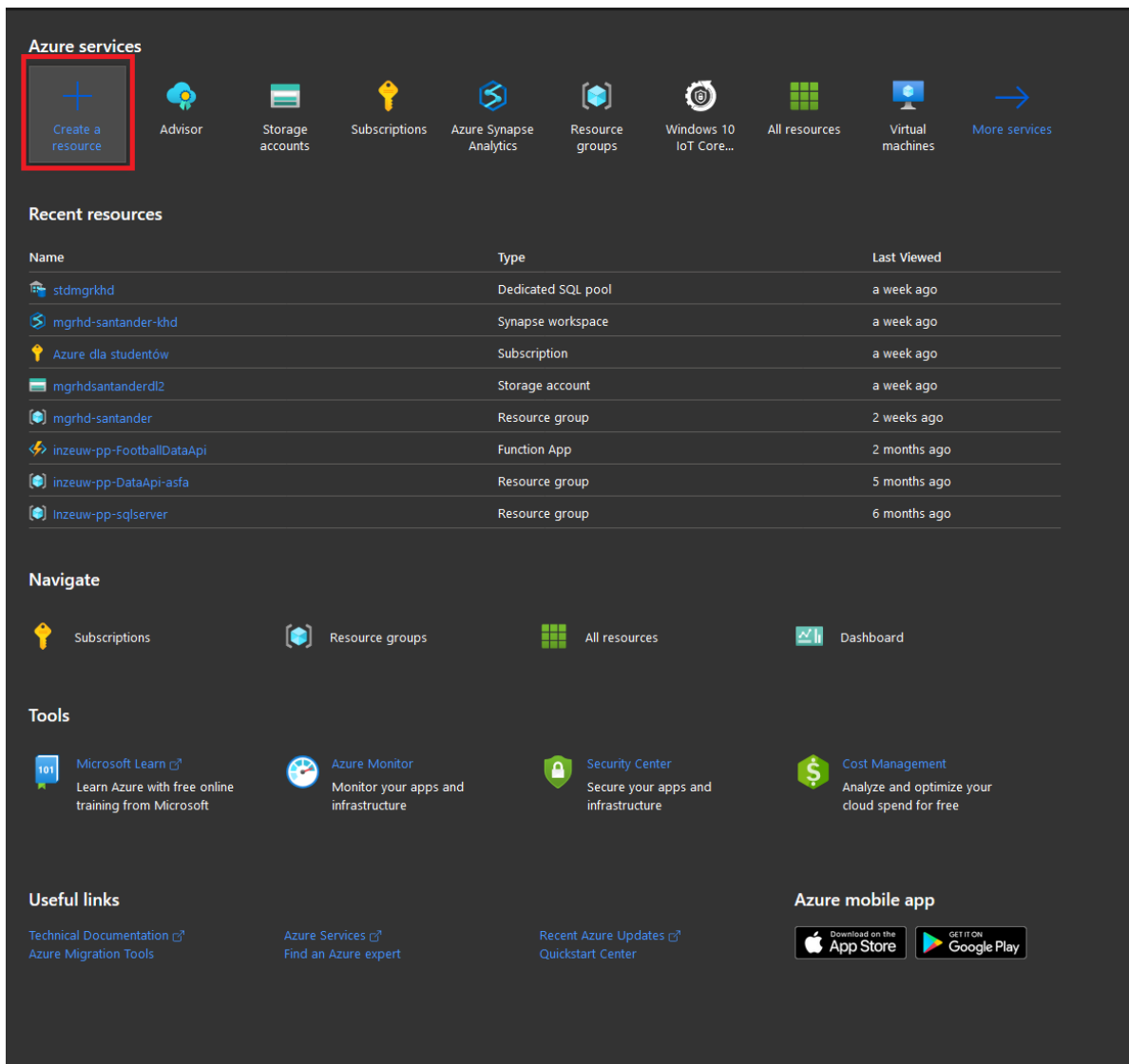
## 3 Schemat danych



## 4 Konfiguracja środowiska pracy na platformie Microsoft Azure

Konfigurację rozpoczynamy od zalogowania się do portalu na konto, na którym zostaną umieszczone wszystkie utworzone zasoby. Przed rozpoczęciem zalecamy zmianę języka klikając na zębatkę widoczną w prawym górnym rogu strony. Po otwarciu menu podręcznego wybieramy zakładkę dotyczącą języka i zmieniamy główny język na angielski.

1. Będąc zalogowanym na platformie *Microsoft Azure* kliknij w zaznaczoną na obrazie ikonę plusa.



2. Wyszukaj zasób „Resource group” i kliknij przycisk „Create”. W polu „Subscription” powinna zostać wybrana subskrypcja, z której pobierane będą środki pieniężne. Kolejno wybierz nazwę grupy.

Home > New > Resource group >

## Create a resource group

Basics Tags Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

**Project details**

Subscription \* ⓘ Azure dla studentów

Resource group \* ⓘ your-name **1**

**Resource details**

Region \* ⓘ (Europe) West Europe

3. Na ekranie wyszukiwania zasobów kolejno znajdź „Storage account” i kliknij „Create”.

Home > New >

## Storage account

Microsoft

**Storage account** Add to Favorites

Microsoft

★★★★☆ 4.2 (1734 ratings)

Create

Overview Plans Usage Information + Support Reviews

4. W celu utworzenia nowego „Storage account”:

4.1. Wybierz poprzednio utworzoną grupę zasobów

4.2. Wybierz nazwę dla swojego konta magazynu, który będzie Twoim *Data Lake’iem*. Kolejno wybierz lokalizację - najlepiej najbliższą, czyli West Europe

4.3. Dla oszczędności zmień opcję „Replication” na *Locally-redundant storage (LRS)*

**Create storage account** ...

Basics Networking Data protection Advanced Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more about Azure storage accounts](#)

**Project details**

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* Azure dla studentów

Resource group \* (New) your-resource-group

[Create new](#)

**Instance details**

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

Storage account name \* ① yourdatalatename ✓

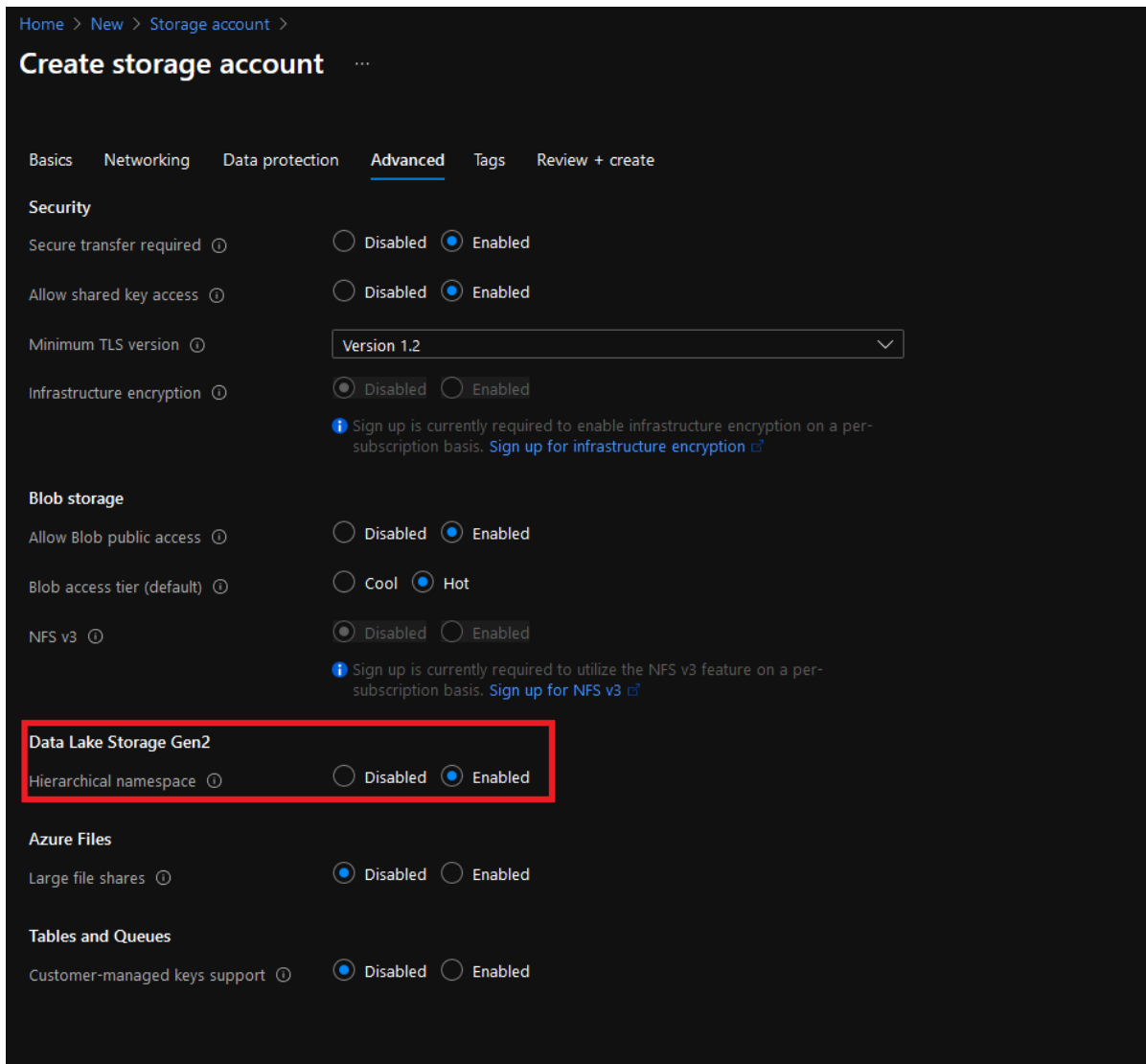
Location \* (Europe) West Europe

Performance ①  Standard  Premium

Account kind ① StorageV2 (general purpose v2)

Replication ① Locally-redundant storage (LRS)

5. Przejdź do zakładki „Advanced” i zmień opcję „Hierarchical namespace” na „Enabled”. Zmiana tej opcji sprawi, że nasz storage account będzie typu data lake. Kolejno możesz przejść do zakładki „Review + create” i utworzyć storage account.



Home > New > Storage account >

## Create storage account

Basics Networking Data protection **Advanced** Tags Review + create

### Security

Secure transfer required  Disabled  Enabled

Allow shared key access  Disabled  Enabled

Minimum TLS version  Version 1.2

Infrastructure encryption  Disabled  Enabled

**Blob storage**

Allow Blob public access  Disabled  Enabled

Blob access tier (default)  Cool  Hot

NFS v3  Disabled  Enabled

**Data Lake Storage Gen2**

Hierarchical namespace  Disabled  Enabled

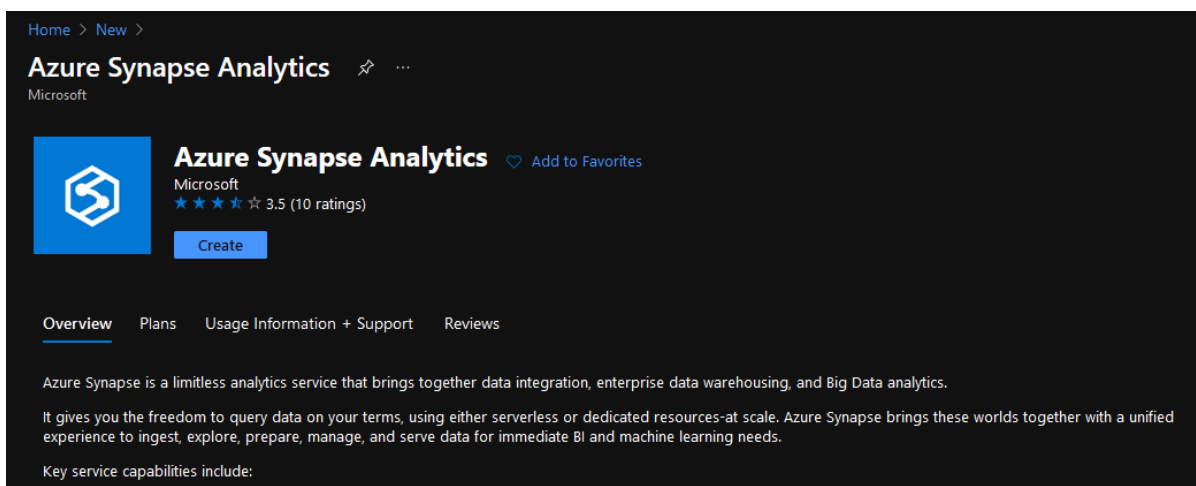
### Azure Files

Large file shares  Disabled  Enabled

### Tables and Queues

Customer-managed keys support  Disabled  Enabled

6. Następnie powróć do ekranu dodawania zasobów i znajdź *Azure Synapse Analytics* i również kliknij „Create” jak na obrazku.



Home > New >

## Azure Synapse Analytics

Microsoft

**Azure Synapse Analytics**  Add to Favorites

Microsoft

★★★★☆ 3.5 (10 ratings)

Create

Overview Plans Usage Information + Support Reviews

Azure Synapse is a limitless analytics service that brings together data integration, enterprise data warehousing, and Big Data analytics. It gives you the freedom to query data on your terms, using either serverless or dedicated resources-at scale. Azure Synapse brings these worlds together with a unified experience to ingest, explore, prepare, manage, and serve data for immediate BI and machine learning needs.

Key service capabilities include:

## 7. W celu utworzenia *Synapse workspace*:

- 7.1. Wybierz utworzony wcześniej „Resource group”
- 7.2. „Workspace name” to nazwa naszego zasobu *Synapse*
- 7.3. „Account name” należy wybrać z listy wcześniej utworzony *DataLake*
- 7.4. „File system name” to nazwa kontenera znajdującego się na *DataLake*, jeżeli żadnego nie ma należy utworzyć nowy poprzez przycisk „Create new” i nadać mu nazwę

### Create Synapse workspace

Basics Security Networking Tags Review + create

Create a Synapse workspace to develop an enterprise analytics solution in just a few clicks.

#### Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all of your resources.

Subscription \*  **1**

Resource group \*  **1**

[Create new](#)

Managed resource group

#### Workspace details

Name your workspace, select a location, and choose a primary Data Lake Storage Gen2 file system to serve as the default location for logs and job output.

Workspace name \*  **2**

Region \*

Select Data Lake Storage Gen2 \*  From subscription  Manually via URL

Account name \*  **3**

[Create new](#)

File system name \*  **4**

[Create new](#)

**i** We will automatically grant the workspace identity data access to the specified Data Lake Storage Gen2 account, using the [Storage Blob Data Contributor](#) role. To enable other users to use this storage account after you create your workspace, perform these tasks:

- Assign other users to the **Contributor** role on workspace
- Assign other users the appropriate **Synapse RBAC roles** using Synapse Studio
- Assign yourself and other users to the **Storage Blob Data Contributor** role on the storage account

[Learn more](#)

[Review + create](#) [< Previous](#) [Next: Security >](#)

8. Należy wypełnić informacje dotyczące administratora SQL, login i hasło.

Home > New > Azure Synapse Analytics >

## Create Synapse workspace

\* Basics \* **Security** Networking Tags Review + create


Configure security options for your workspace.

### SQL administrator credentials

Provide credentials that can be used for administrator access to the workspace's SQL pools. If you don't provide a password, one will be automatically generated. You can change the password later.

Admin username *	<input type="text" value="sqladminuser"/>
Password	<input type="password" value="Enter server password"/> ✓
Confirm password	<input type="password" value="Confirm the above password"/> ✓

### Workspace encryption

 Double encryption configuration cannot be changed after opting into using a customer-managed key at the time of workspace creation.

Choose to encrypt all data at rest in the workspace with a key managed by you (customer-managed key). This will provide double encryption with encryption at the infrastructure layer that uses platform-managed keys. [Learn more](#)


Enable double encryption using a customer-managed key

### System assigned managed identity

Choose the permissions that you would like to assign to the workspace's system-assigned identity. [Learn more](#)

Allow pipelines (running as workspace's system assigned identity) to access SQL pools. ⓘ

Allow network access to Data Lake Storage Gen2 account. ⓘ

 The selected Data Lake Storage Gen2 account does not restrict network access using any network access rules, or you selected a storage account manually via URL under Basics tab. [Learn more](#)

9. Przechodząc do zakładki „Review + Create” tworzymy nasz *Synapse Analytics*.

Home > New > Azure Synapse Analytics >

## Create Synapse workspace ...

✓ Validation succeeded

\* Basics \* Security Networking Tags Review + create

### Product Details

Azure Synapse Analytics workspace by Microsoft  
Serverless SQL est. cost/TB ⓘ  
**4.22 EUR**  
[Terms of use](#) | [Privacy policy](#)

### Terms

By clicking Create, I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. For additional details see [Azure Marketplace Terms](#).

### Basics

Subscription	Azure dla studentów
Resource group	your-resource-group
Region	West Europe
Workspace name	(new) your-synapse-name
Data Lake Storage Gen2 account	https://mgrhdsantanderdl2.dfs.core.windows.net
Data Lake Storage Gen2 file system	khd-datalake
Managed resource group	None
Role assignments	The Storage Blob Data Contributor role will be assigned on the specified Data Lake Storage Gen2 account to the workspace managed identity.

### Security

Admin username	sqladminuser
Password	Auto-generated
Allow pipelines to access SQL pools	Yes
Allow network access to storage account	No
Double encryption	No

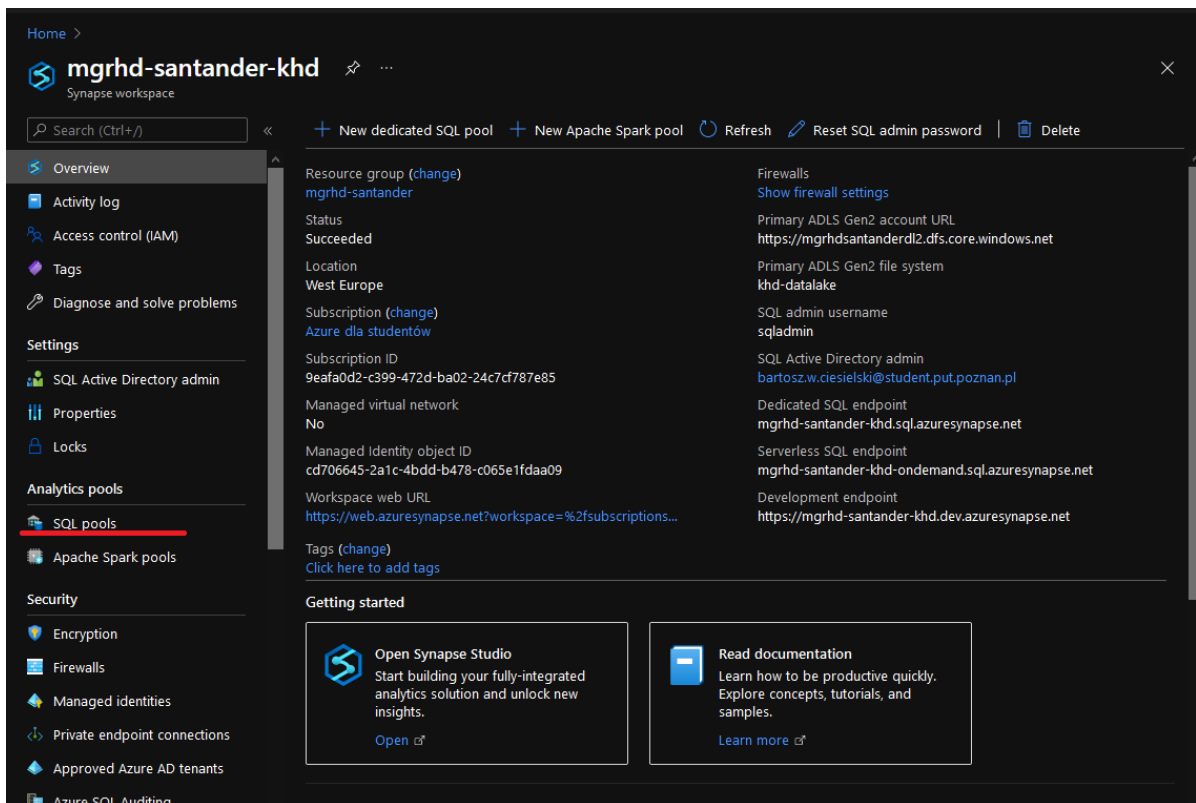
### Networking

Managed virtual network	No
Allow connections from all IP addresses	Yes

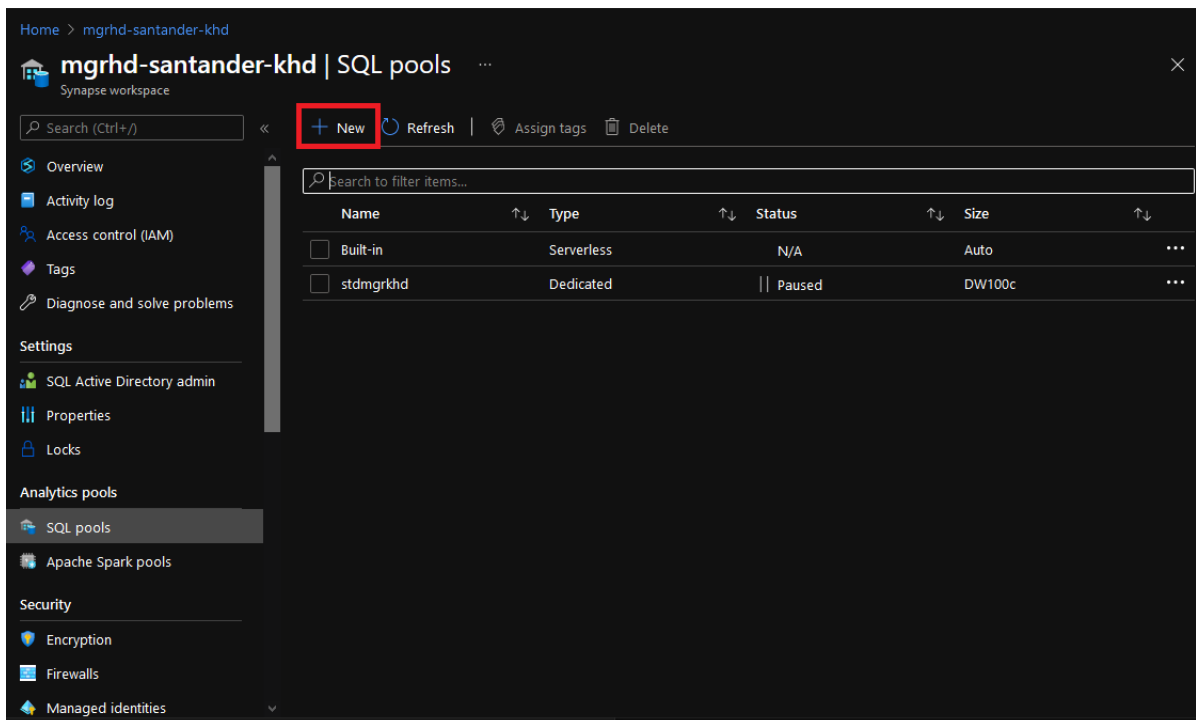
[Create](#) [< Previous](#) [Next >](#) [Download a template for automation](#)



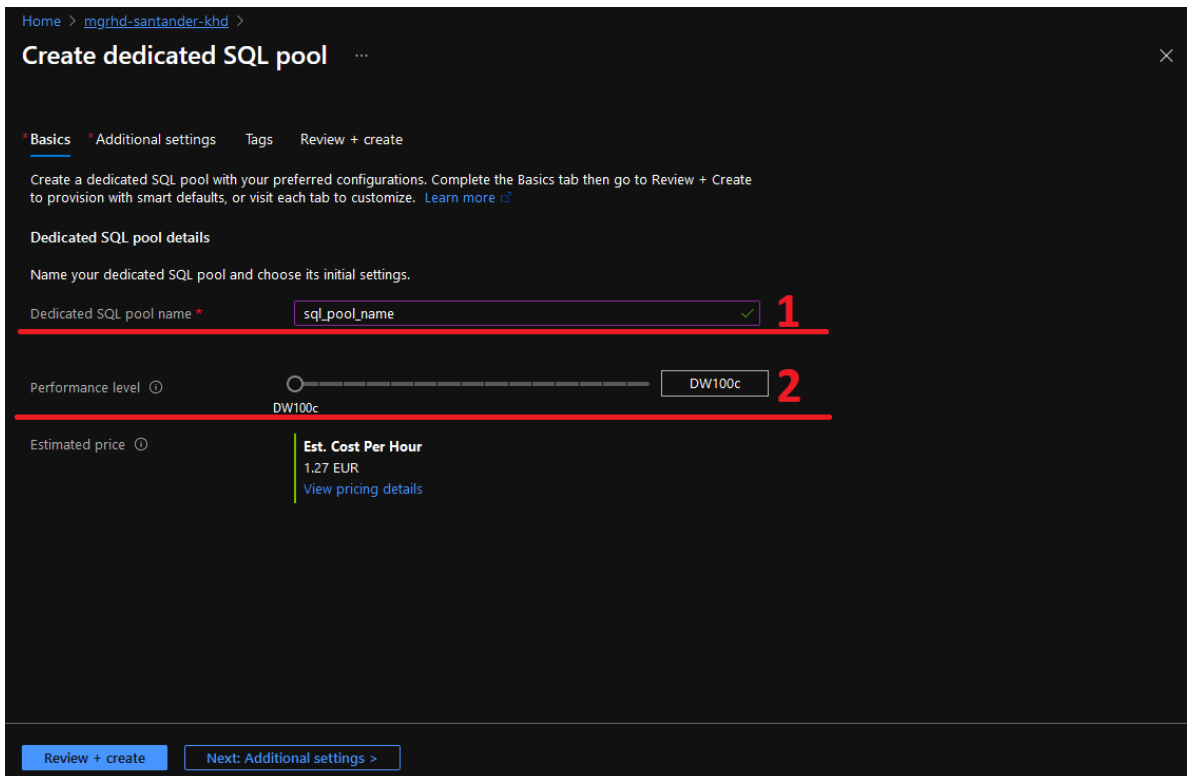
10. Wyszukujemy zasób na portalu wpisując nazwę naszego nowo utworzonego *Synapse* i po wejściu widzimy załączony na obrazku widok. Wchodzimy w podkreśloną po lewej opcję *SQL pools*.



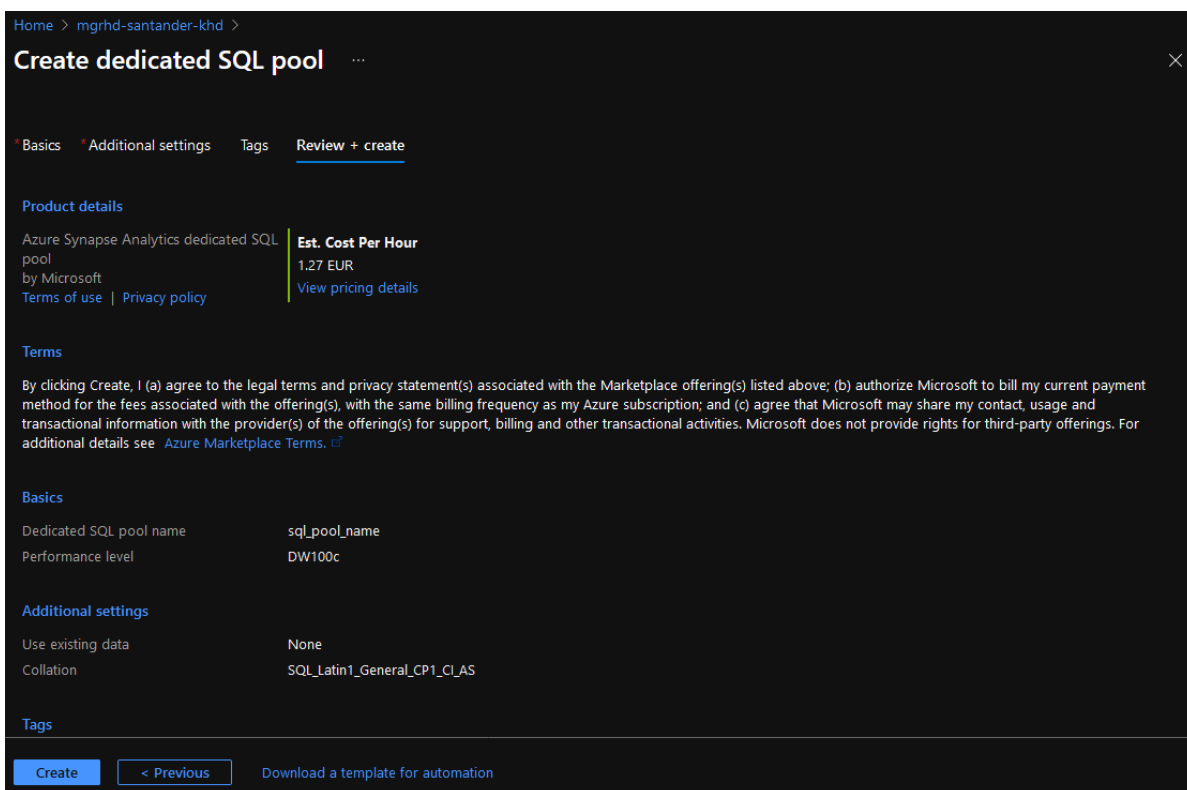
11. Klikamy „New”.



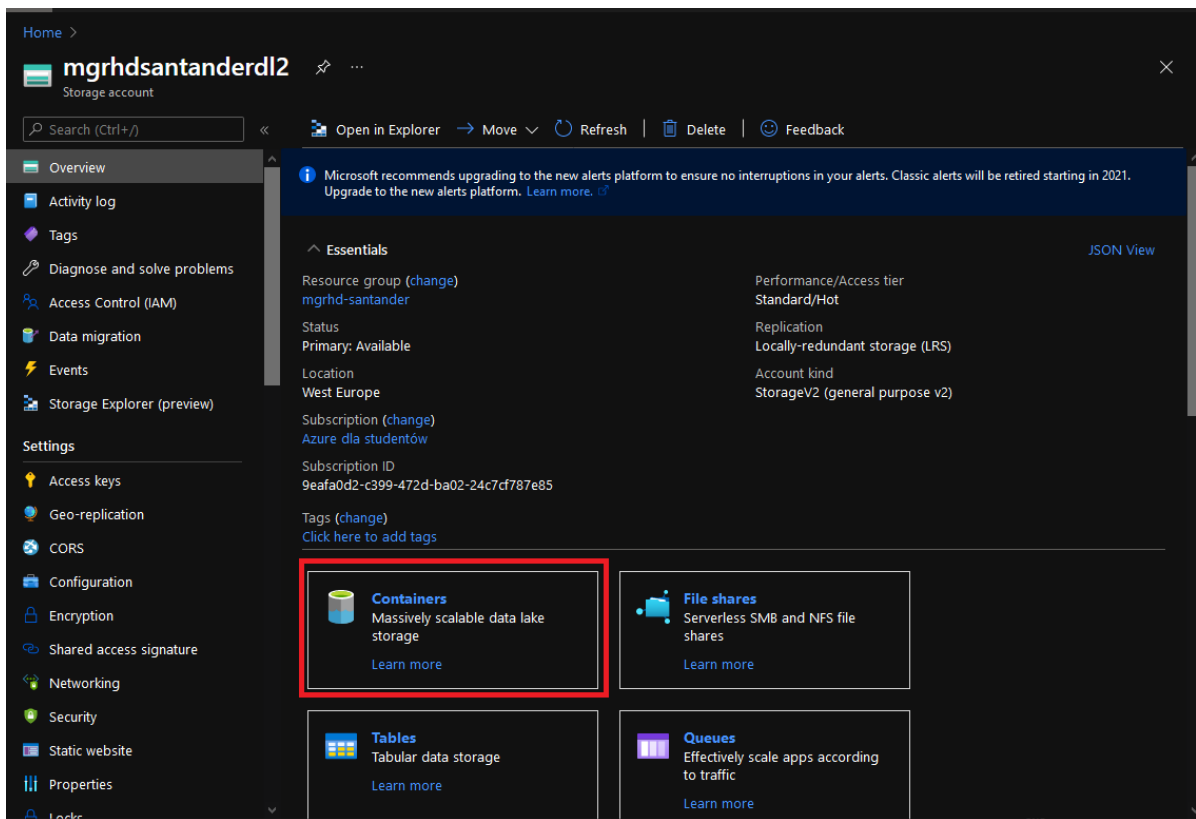
12. Nadajemy nazwę naszej bazie danych i wybieramy najtańszą opcję w celu zaoszczędzenia pieniędzy.



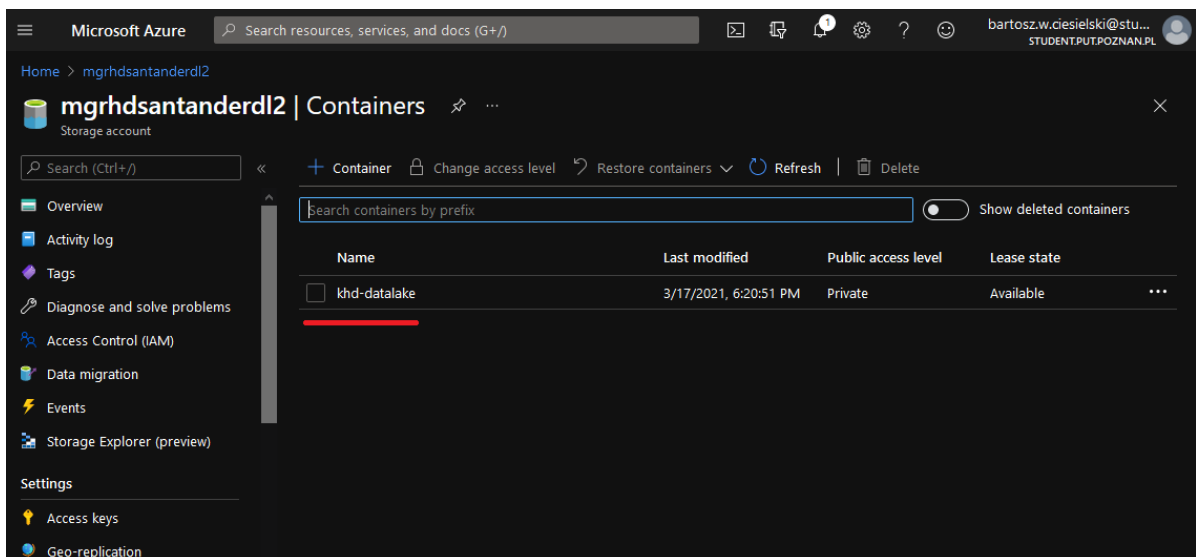
13. Reszta może pozostać bez zmian, tworzymy zasób.



14. Wchodzimy na wcześniej utworzony datalake wpisując jego nazwę w wyszukiwarce na portalu. Kolejno wybieramy zakładkę „Containers”.



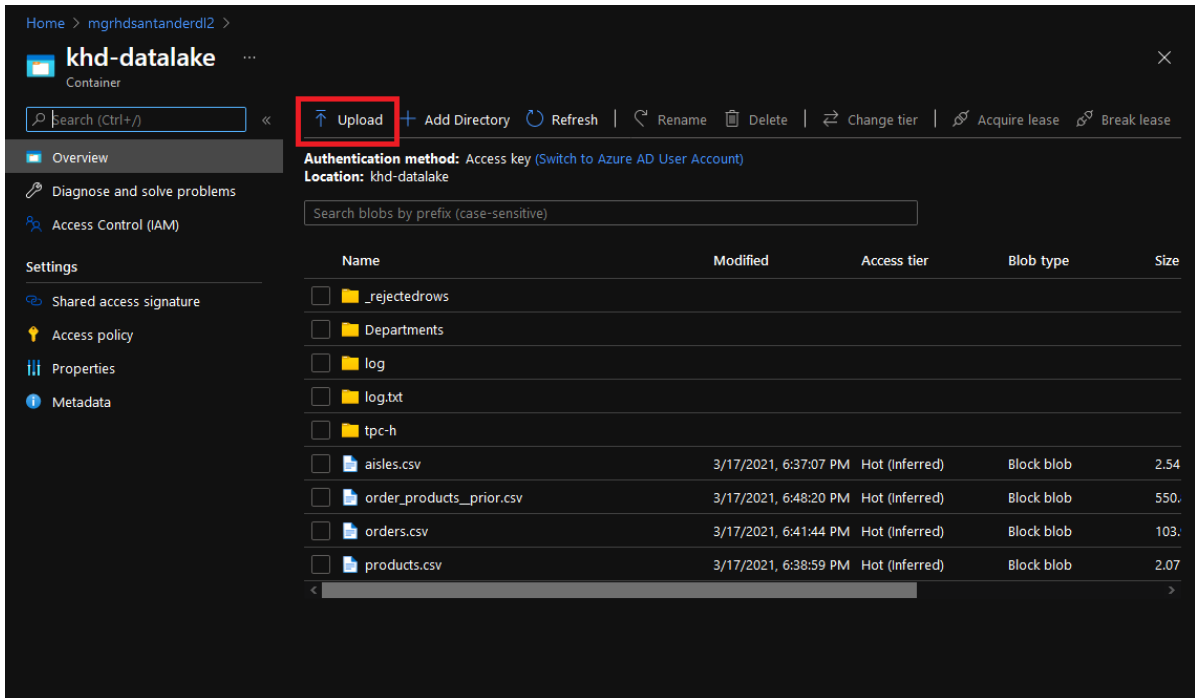
15. Wchodzimy w podpięty do *Synapse* kontener (wybrany jako „File system name” w punkcie 7).



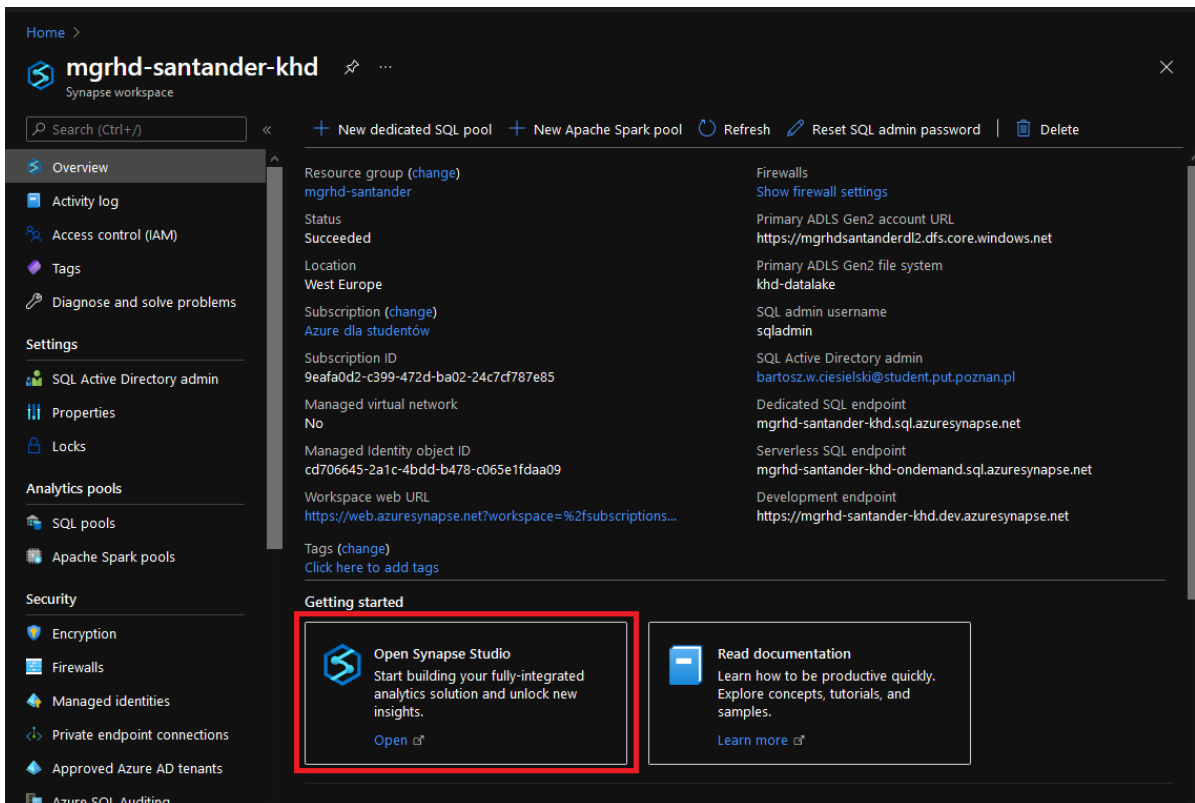
16. Wgrywamy przekazane pliki do datalake klikając „Upload”.

Wykorzystywane w projekcie pliki można znaleźć na githubie:

<https://github.com/Put-EngineeringThesis-soccer-prediction/SantanderAzure/tree/master/data>



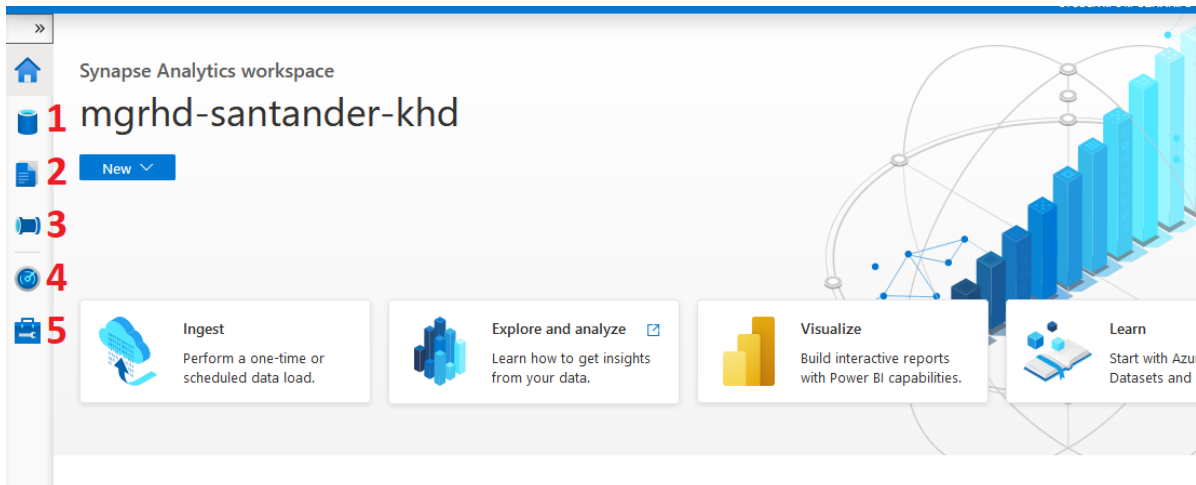
17. Wracamy do *Synapse Analytics* i wybieramy zaznaczone okno. Otworzy ono interfejs, w którym zarządza się Synapsem.



18. Ekran powitalny *Synapse workspace*, na którym widzimy:

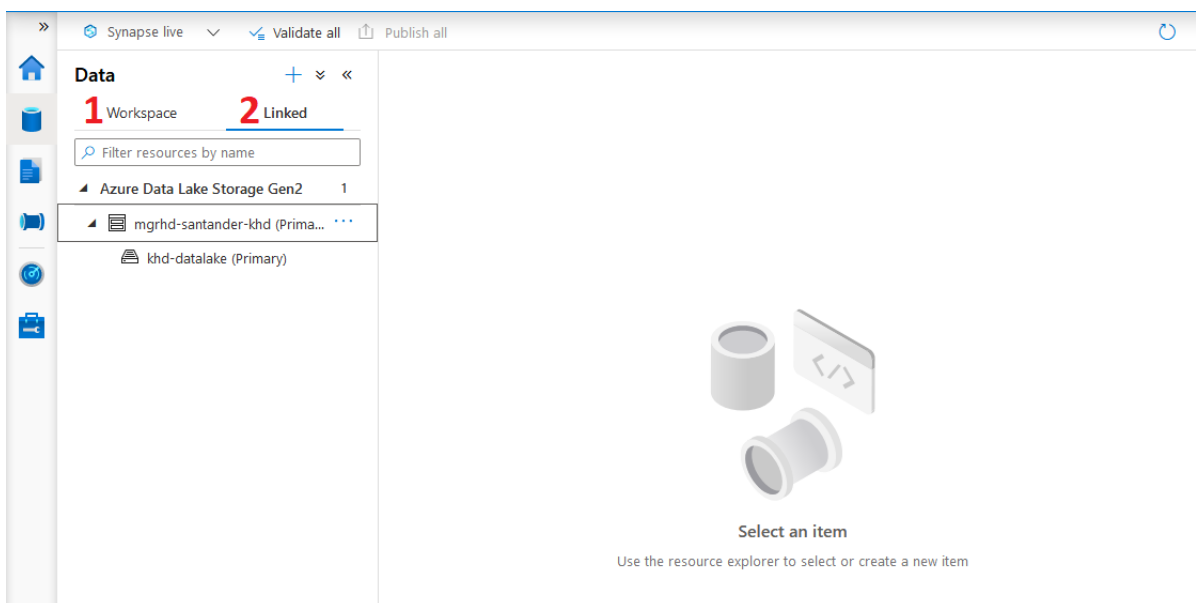
- 18.1. Bazy danych oraz datalake
- 18.2. Skrypt SQL, notatniki Spark i inne
- 18.3. Okno służące do tworzenia pipeline'ów
- 18.4. Monitorowanie wykonywanych zadań w Synapsie
- 18.5. Zarządzanie *SQL Pools* oraz *Spark pools* jak i połączeniami z zewnętrznymi zasobami

Wybieramy zakładkę oznaczoną numerem 1

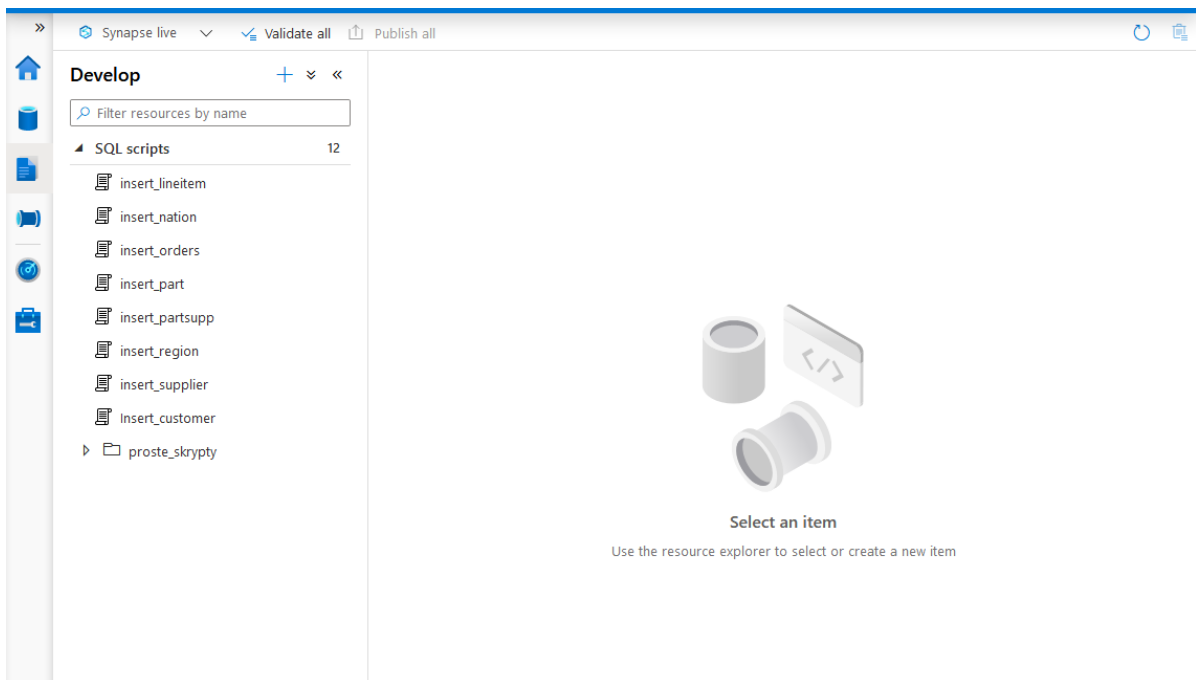


19. Tutaj mamy dostęp do naszych baz danych jak i plików załadowanych do datalake:

- 19.1. Workspace zawierający bazy danych i tabele zewnętrzne
- 19.2. Połączone obiekty typu datalake

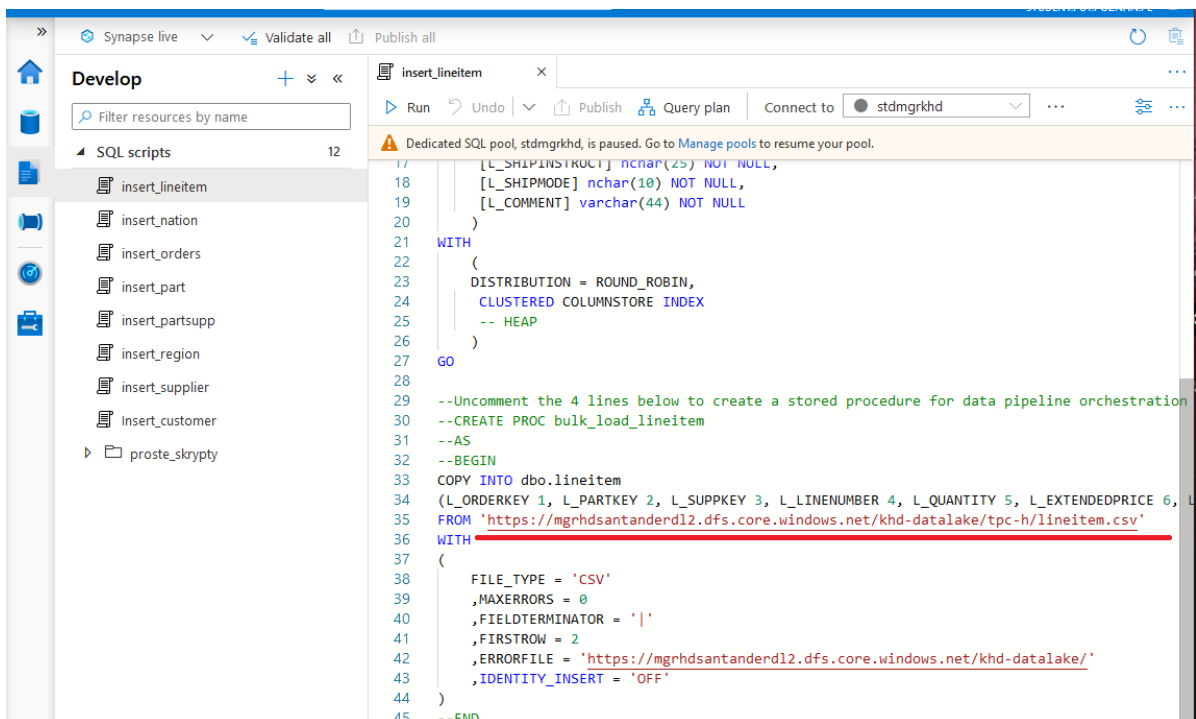


20. Zapisywane są tutaj wszelkie skrypty SQL do późniejszego użytku. Będą tu również udostępnione skrypty do ładowania danych do bazy danych.



21. W przekazanych skryptach należy zmienić podkreśloną liniijkę (w każdym skrypcie), aby wskazywała na odpowiedni plik w Twoim środowisku.

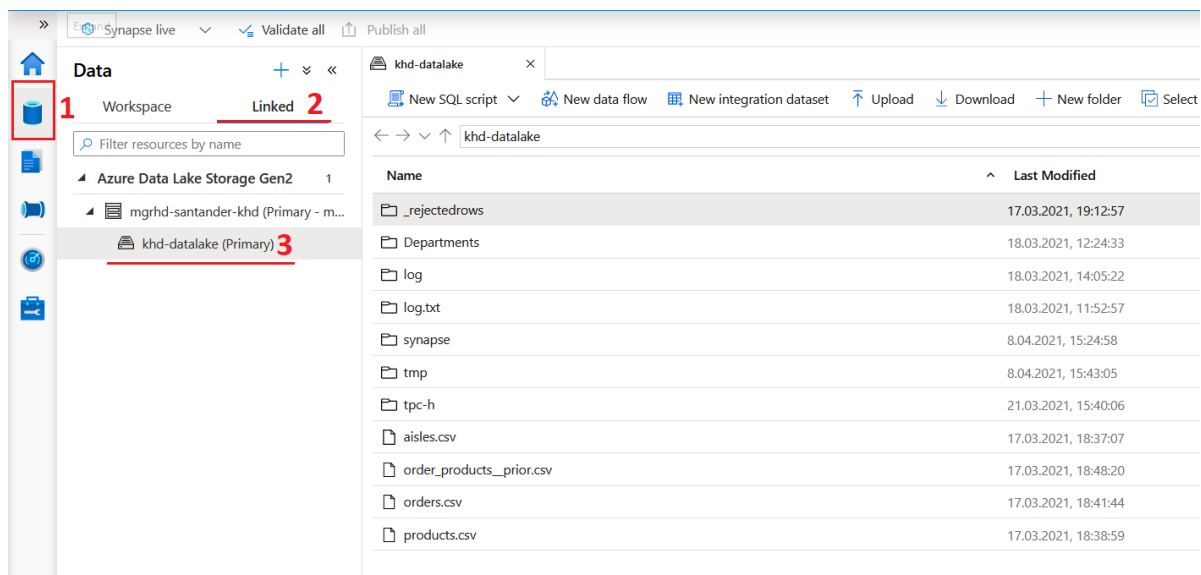
Skrypty można znaleźć na githubie: <https://github.com/Put-EngineeringThesis-soccer-prediction/SantanderAzure/tree/master/InsertScripts>



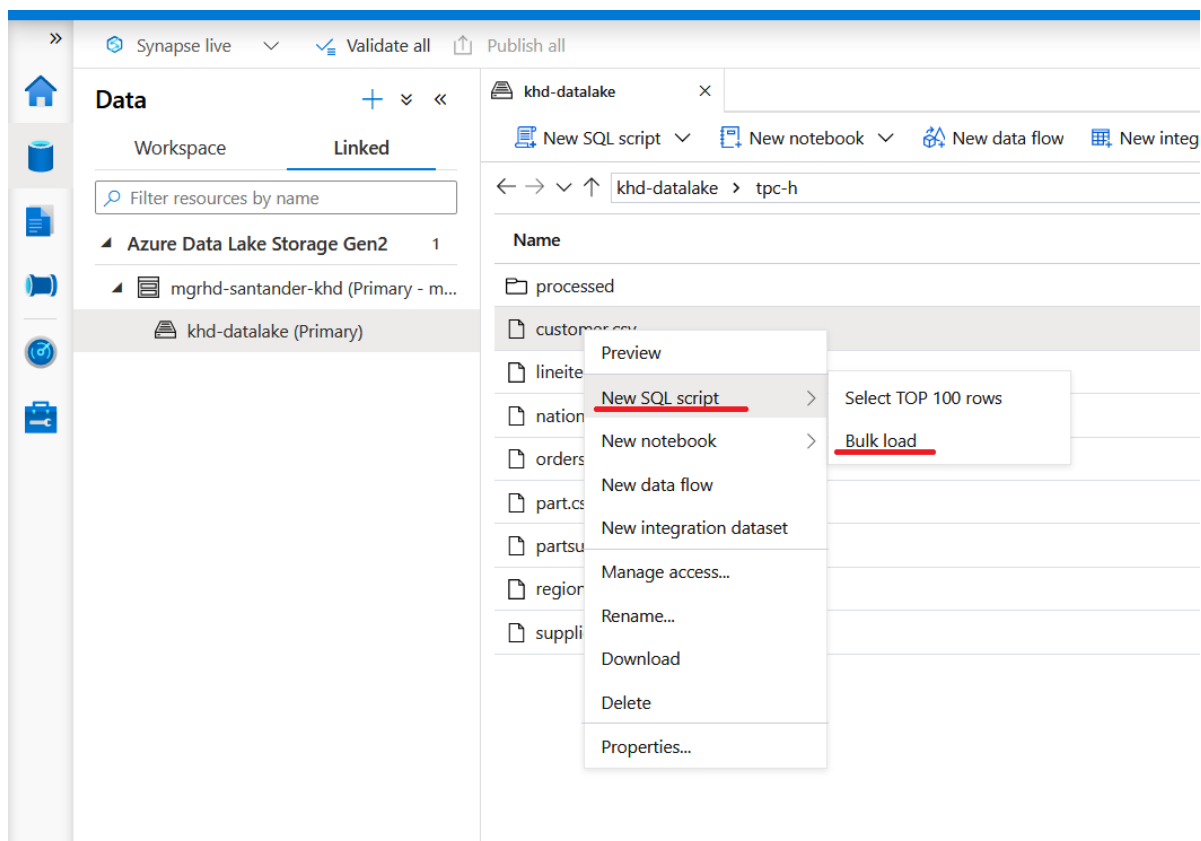
## 5 Ładowanie danych do bazy danych przez Azure Synapse Analytics Workbench

W celu załadowania danych do bazy danych z plików, która ma służyć jak hurtownia danych możemy użyć Azure Synapse Workbench do wygenerowania odpowiedniego kodu.

1. Aby załadować dane poprzez portal trzeba wrócić do Synapse Workbench, wejść do zakładki „data” (1) i znaleźć dane. Dane odnajdziemy w zakładce (2) Linked w podłączonym datalake’u (3).




2. Wybieramy przygotowany plik csv, klikamy prawym przyciskiem na wybrany plik, wchodzimy w „New SQL script” i dalej „Bulk load”.



3. Po prawej stronie pojawi się widoczne okienko. Dla naszych danych wybieramy (1) „Pipe(|)”, „First row” (2) ustawiamy na 1 i zaznaczamy „Infer column names” (3). Następnie klikamy dalej.

## Bulk load

### Source data file format settings

Specify the format and layout of your data. [Learn more](#) 

Source storage  
tpc-h/orders.csv

[Preview data](#)

#### File type

Text format 

#### Field terminator

Pipe (|)  **1**

Edit

#### Row terminator

Default (\n, or \r\n) 

Edit

#### First row

1  **2**

Infer column names  **3**

#### Field quote

Default (double quote ") 

Edit

#### Date format

Session default 

#### Compression

none 


#### Encoding

Auto detect (UTF8, UTF16) 

#### Max string length

4000 

### Error settings

Specify your error tolerance and where rejected rows and the corresponding error file will be written. [Learn more](#) 

#### Max errors

0



4. Wybieramy naszą dedykowaną bazę SQL, która jest naszą hurtownią danych (1), nadajemy nazwę tabeli (2) (najlepiej zgodnie z nazwą pliku), upewniamy się że jest wybrane „Using SQL script” (4), a następnie klikamy „configure column mapping” (3)

## Bulk load

### Select target SQL pool

Specify the target location for your load including the SQL pool, table, and column mapping.

[Learn more](#)

Select SQL pool ^ ⓘ

stdmgrkhd **1**

### Select target table

Existing table  Create new

New target table ⓘ

[schema].[tableName] **2** Clustered columnstore index

[Configure column mapping](#) **3**




Automatically  Using SQL script **4**




5. Tutaj możemy określić nazwy kolumn oraz ich typy w tabeli. Dla naszych danych należy również usunąć kolumn empty, która wynikała z sposobu generowania danych i posiadała zbędną pustą kolumnę. Typy danych dla każdej z tabel są określone na schemacie. Po ustawieniu wszystkich wartości wracamy do poprzedniego okna i tworzymy skrypt.

## Bulk load

### Configure column mapping

Map source data fields in the input file to target table columns. You can choose to omit and reorder fields for the load.

 New mapping  Clear  Delete

Source (Column number)	Destination	Type
1	O_ORDERKEY	<sup>123</sup> bigint
2	O_CUSTKEY	<sup>123</sup> bigint
3	O_ORDERST...	nvarchar(4000)
4	O_TOTALPRI...	<sup>1.2f</sup> float
5	O_ORDERD ...	 date
6	O_ORDERPR...	nvarchar(4000)
7	O_CLERK	nvarchar(4000)
8	O_SHIPPRIO...	<sup>123</sup> bigint
9	O_COMMENT	nvarchar(4000)
<input type="checkbox"/> 10	empty	nvarchar(4000)  

## 6. Utworzony skrypt służący do ładowania danych z pliku do tabeli SQL.

```
khd-datalake SQL script 1
Run Undo Publish Query plan Connect to stdmgrkhd Use database stdmgrkhd
1 IF NOT EXISTS (SELECT * FROM sys.objects O JOIN sys.schemas S ON O.schema_id = S.schema_id WHERE O.NAME = 'tablename' AND O.TYPE = 'U' AND S.NAME = 'dbo')
2 CREATE TABLE dbo.tablename
3 (
4     [O_ORDERKEY] bigint,
5     [O_CUSTKEY] bigint,
6     [O_ORDERSTATUS] nvarchar(4000),
7     [O_TOTALPRICE] float,
8     [O_ORDERDATE] date,
9     [O_ORDERPRIORITY] nvarchar(4000),
10    [O_CLERK] nvarchar(4000),
11    [O_SHIPPRIORITY] bigint,
12    [O_COMMENT] nvarchar(4000)
13 )
14 WITH
15 (
16     DISTRIBUTION = ROUND_ROBIN,
17     CLUSTERED COLUMNSTORE INDEX
18     -- HEAP
19 )
20 GO
21
22 --Uncomment the 4 lines below to create a stored procedure for data pipeline orchestration
23 --CREATE PROC bulk_load_tablename
24 --AS
25 --BEGIN
26 COPY INTO dbo.tablename
27 (O_ORDERKEY 1, O_CUSTKEY 2, O_ORDERSTATUS 3, O_TOTALPRICE 4, O_ORDERDATE 5, O_ORDERPRIORITY 6, O_CLERK 7, O_SHIPPRIORITY 8, O_COMMENT 9)
28 FROM 'https://mgrhdsantanderd12.dfs.core.windows.net/khd-datalake/tpc-h/orders.csv'
29 WITH
30 (
31     FILE_TYPE = 'CSV'
32     ,MAXERRORS = 0
33     ,FIELDTERMINATOR = '|'
34     ,FIRSTROW = 2
35     ,ERRORFILE = 'https://mgrhdsantanderd12.dfs.core.windows.net/khd-datalake/'
36     ,IDENTITY_INSERT = 'OFF'
37 )
38 --END
39 GO
40
41 SELECT TOP 100 * FROM dbo.tablename
42 GO
```

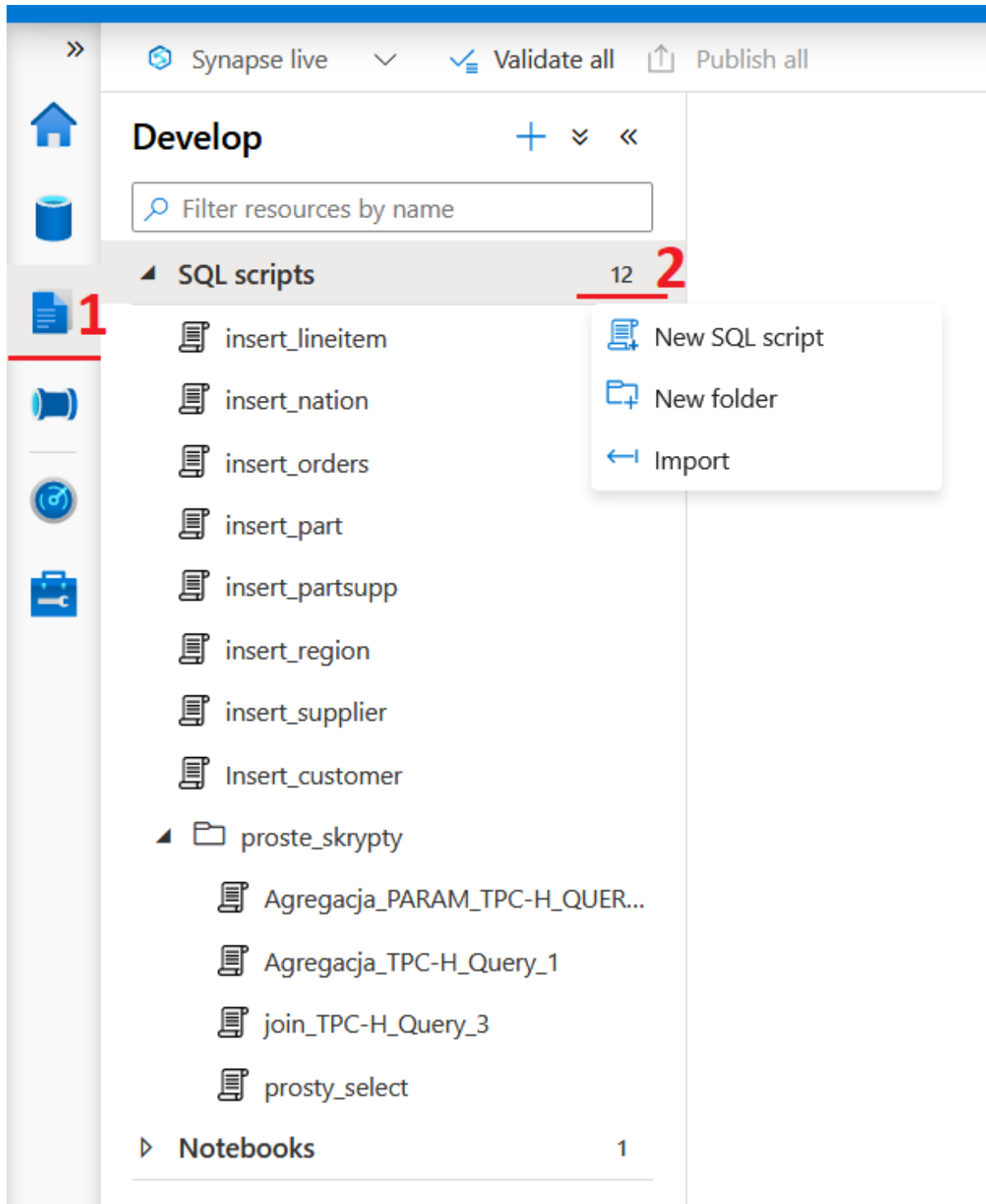
## 6 Przykładowe zapytania

Przygotowane dane w bazie danych można odpytywać językiem T-SQL.

1. W Azure Synapse Analytics Workbench przechodzimy do zakładki „Develop” (1) i w tej zakładce możemy utworzyć nowy skrypt lub załadować przygotowany plik SQL (2 - można również kliknąć w symbol plus powyżej)

Przykładowe zapytania można znaleźć na githubie:

[https://github.com/Put-EngineeringThesis-soccer-prediction/SantanderAzure/tree/master/Sample\\_Queries](https://github.com/Put-EngineeringThesis-soccer-prediction/SantanderAzure/tree/master/Sample_Queries)



2. W celu uruchomienia przykładowego skryptu należy wybrać bazę danych, na której zostanie on uruchomiony (1) i kliknąć przycisk „Run” (2)

The screenshot shows the SQL Server Enterprise Developer interface. On the left, the 'SQL scripts' folder is expanded, showing a file named 'Agregacja\_PARAM\_TPC-H\_QUERY...'. The main window displays a SQL script with the following content:

```

1  -- 'This query reports the amount of business that was billed, shipped, and returned.'
2
3  DECLARE @DELTA int = -90;
4
5  SELECT L_RETURNFLAG, L_LINestatus,
6         SUM(L_QUANTITY) AS SUM_QTY,
7         SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
8         SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS SUM_DISC_PRICE,
9         SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1+L_TAX)) AS SUM_CHARGE,
10        AVG(L_QUANTITY) AS AVG_QTY,
11        AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
12        AVG(L_DISCOUNT) AS AVG_DISC,
13        COUNT(*) AS COUNT_ORDER
14 FROM LINEITEM
15 WHERE L_SHIPDATE <= dateadd(dd, @DELTA, cast('1998-12-01' as datetime))
16 GROUP BY L_RETURNFLAG, L_LINestatus
17 ORDER BY L_RETURNFLAG, L_LINestatus
18 GO;

```

The 'Connect to' dropdown is set to 'stdmgrkhd' and 'Use database' is also set to 'stdmgrkhd'. The 'Run' button is highlighted with a red '2'.

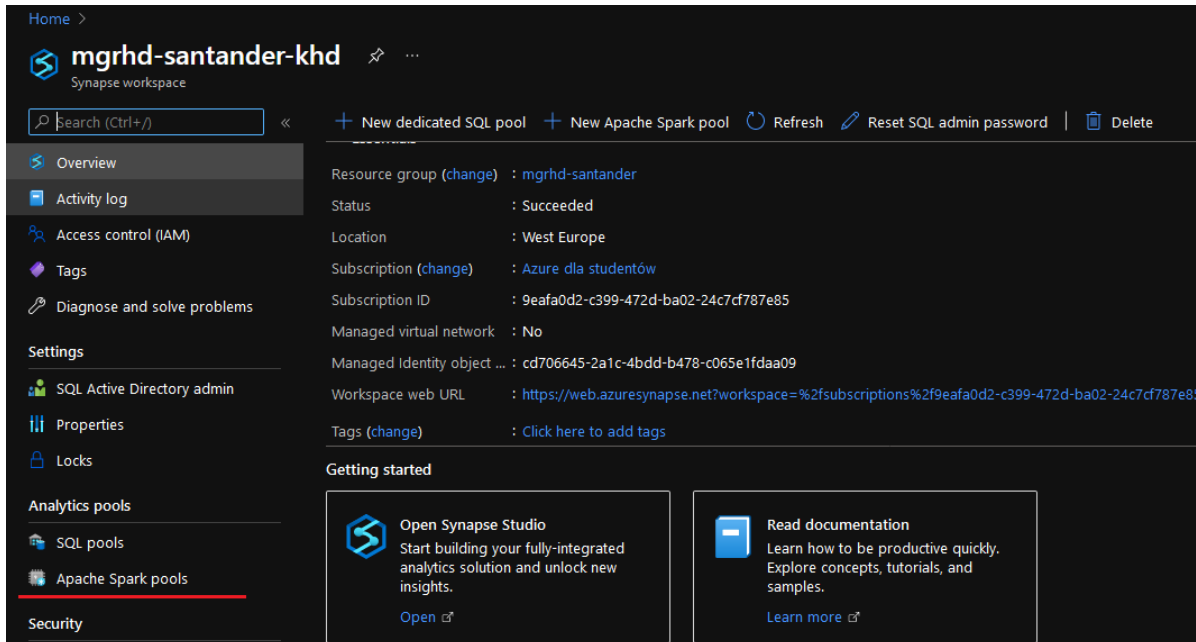
3. Poniżej widoczna jest odpowiedź na zapytanie utworzone w poprzednim punkcie.

L_RETURNFLAG	L_LINestatus	SUM_QTY	SUM_BASE_PRI...	SUM_DISC_PRI...	SUM_CHARGE	AVG_QTY	AVG_PRICE	AVG_DISC	COUNT_ORDER
A	F	37734107.00	56586554400.73	53758257134.8...	55909065222.8...	25.522005	38273.129734	0.049985	1478493
N	F	991417.00	1487504710.38	1413082168.05...	1469649223.19...	25.516471	38284.467760	0.050093	38854
N	O	74476040.00	111701729697...	106118230307....	110367043872....	25.502226	38249.117988	0.049996	2920374
R	F	37719753.00	56568041380.90	53741292684.6...	55889619119.8...	25.505793	38250.854626	0.050009	1478870

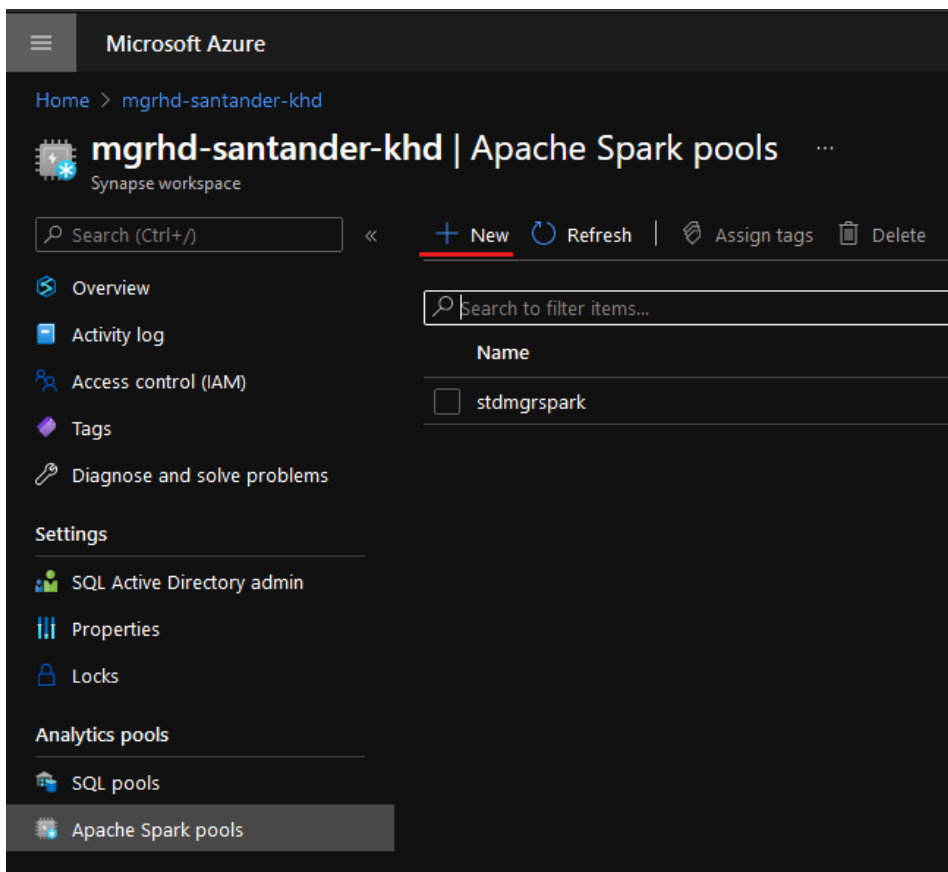
## 7 Przetwarzanie danych przy użyciu Apache Spark

Azure Synapse Analytics udostępnia możliwość tworzenia skryptów przy użyciu pythona w notebookach sparkowych. Wykorzystywana jest w tym celu biblioteka PySpark. Takie skrypty dają możliwość przetwarzania danych przechowywanych w hurtowni danych zarówno Synapse, jak i w Data Lake. Można je ze sobą łączyć i wynik zapisywać ponownie w wybranym miejscu w Data Lake'u.

1. Aby umożliwić przetwarzanie Spark, trzeba utworzyć tzw. „Apache Spark pool”. W tym celu, mając otwarty zasób Azure Synapse Analytics, przechodzimy do zakładki „Apache Spark pools”.



2. Klikamy przycisk „New”.



3. Tworząc Apache Spark pool należy podać jego dowolną nazwę (1), wielkość węzła klastra (2) oraz liczbę węzłów wraz z wyborem autoskalowania (3) (dynamiczna zmiana liczby węzłów w zależności od obciążenia). Dla zminimalizowania kosztów wybieramy najmniejszy węzeł oraz minimalną liczbę węzłów. Wyłączamy również opcje autoskalowania.

Home > mgrhd-santander-khd >

## Create Apache Spark pool ...

[\\* Basics](#) [\\* Additional settings](#) [Tags](#) [Review + create](#)

Create a Synapse Analytics Apache Spark pool with your preferred configurations. Complete the Basics tab then go to Review + create to provision with smart defaults, or visit each tab to customize.

### Apache Spark pool details

Name your Apache Spark pool and choose its initial settings.

Apache Spark pool name \*  **1**

Node size family **MemoryOptimized**

Node size \*  **2**

Autoscale \*  Enabled  Disabled

Number of nodes \*    **3**

Estimated price ⓘ

**Est. cost per hour**  
3.09 to 10.30 EUR  
[View pricing details](#)

**⚠** Contact an **Owner** of the storage account, and verify that the following role assignments have been made:

- Assign the workspace MSI to the **Storage Blob Data Contributor** role on the storage account
- Assign you and other users to the **Storage Blob Data Contributor** role on the storage account

Once those assignments are made, the following Spark features can be used: (1) Spark Library Management, (2) Read and Write data to SQL pool databases via the Spark SQL connector, and (3) Create Spark databases and tables

[Learn more](#)

4. W opcjach dodatkowych możemy określić po ilu minutach uruchomiony klaster zostanie wyłączony w przypadku bezczynności. Służy to temu, aby nieużywany klaster nie generował niepotrzebnych kosztów.

Home > mgrhd-santander-khd >

## Create Apache Spark pool

Basics **Additional settings** Tags Review + create

Customize additional configuration parameters including automatic pausing and component versions.

### Automatic pausing

Configure automatic pausing. If enabled, the Apache Spark pool will automatically pause after the selected idle time.

Automatic pausing  Enabled  Disabled

Number of minutes idle \*


### Component versions

Select the Apache Spark version for your Apache Spark pool.

Apache Spark *	<input type="text" value="2.4"/>
Python	3.6
Scala	2.11.12
Java	1.8.0_272
.NET Core	3.1
.NET for Apache Spark	1.0
Delta Lake	0.6

### Spark configuration

Upload a Spark configuration file to specify additional properties on the Spark pool. This will be referenced to configure Spark applications upon job submission. [Learn more](#)

File upload  

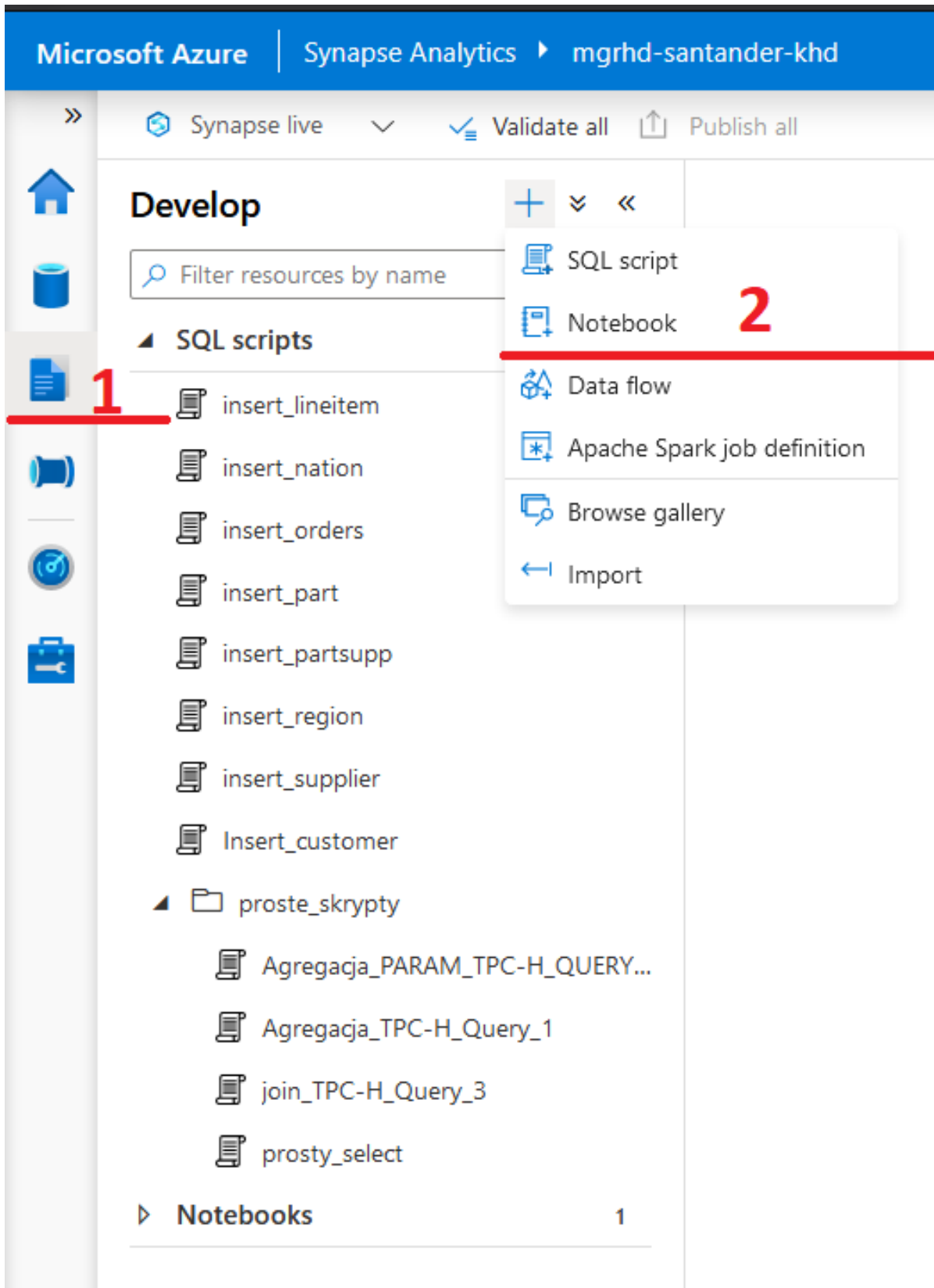
### Packages

Configure settings related to packages for this pool.

Allow session level packages  Enabled  Disabled



5. Po utworzeniu klastra Apache Spark, przechodzimy do Azure Synapse Analytics Workbench do zakładki „Develop” (1). Wybieramy plus (2) w celu utworzenia Notebook’a, w którym umieszczony zostanie kod przetwarzający dane przy użyciu sparka.



6. Poniżej widoczny jest kod ładujący dane z pliku znajdującego się na podpiętym DataLake.

Pierwszy parametr to ścieżka do pliku w postaci „ABFSS Path”. Jest to referencja do DataLake’a, a na końcu ścieżka przypominająca strukturę folderów. W przykładzie wczytujemy plik `tpc-h/region.csv`. (Ścieżka ABFSS może być znaleziona w parametrach pliku, tzn. w oknie data, klikając prawym przyciskiem myszy na plik -> properties.)

Kolejny parametr to format pliku ustawiony na `csv`, czy zawiera nagłówek oraz jaki separator został użyty w przypadku pliku `csv`. Dostępne są również inne formaty plików, np. `json` czy `parquet`. Następnie usuwamy kolumnę „empty” przy użyciu metody „drop”. W ten sposób tworzymy Spark Dataframe, który wyświetlamy w następnej linii kodu.

```
region_df = spark.read.load(
'abfss://khd-datalake@mgrhdsantanderdl2.dfs.core.windows.net/tpc-h/region.csv',
format='csv',
header=True,
sep='|').drop('empty')

region_df.show()
```

```
region_df = spark.read.load(
'abfss://khd-datalake@mgrhdsantanderdl2.dfs.core.windows.net/tpc-h/region.csv',
format='csv',
header=True,
sep='|').drop('empty')

region_df.show()
```

7. W osobnej komórce używamy `%%Spark` w celu zaznaczenia, że będziemy korzystać z języka Scala. Przez ograniczenia środowiska dostęp do bazy danych najprościej uzyskać poprzez użycie kodu zaimplementowanego w Scali. Zaimportowane biblioteki umożliwiają bezpośredni dostęp do tabel w bazach danych znajdujących się w obrębie Azure Synapse Analytics. Zaraz po zaimportowaniu bibliotek odczytujemy tabelę `nation` i zapisujemy jej zawartość do tabeli tymczasowej, aby uzyskać do niej dostęp w następnej części kodu.

```
%%spark

import com.microsoft.spark.sqlanalytics.utils.Constants
import org.apache.spark.sql.SqlAnalyticsConnector._
val df = spark.read.sqlanalytics("stdmgrkhd.dbo.nation")
df.write.mode("overwrite").saveAsTable("tempTable_Nation")
```

8. W ostatniej komórce kodu ładujemy najpierw do Spark Dataframe zawartość utworzonej wcześniej tabeli tymczasowej (spark stosuje tzw. „Lazy Loading”, więc dopóki nie zostaną wykonane na danych konkretne akcje, dane nie są pobierane). Łączymy wczytane wcześniej dane z pliku `csv` z danymi z tabeli pobranymi z hurtowni danych. Wybieramy interesujące nas kolumny, zmieniamy ich nazwy i sortujemy po nazwie regionu. Tak uzyskane wyniki wyświetlamy przy pomocy funkcji „display”, która uruchomi przetwarzanie, po czym zapisujemy zawartość do pliku `parquet` na tym samym `datalake`’u, z którego pobraliśmy plik `region.csv`.

```
nations_df = spark.sql("select_*_from_tempTable_Nation")

result_df = region_df.join(nations_df,
    region_df.R_REGIONKEY == nations_df.N_REGIONKEY) \
    .select(nations_df.N_NAME, region_df.R_NAME) \
    .withColumnRenamed("N_NAME", "nation_name") \
    .withColumnRenamed("R_NAME", "region_name") \
    .orderBy("region_name")

display(result_df)

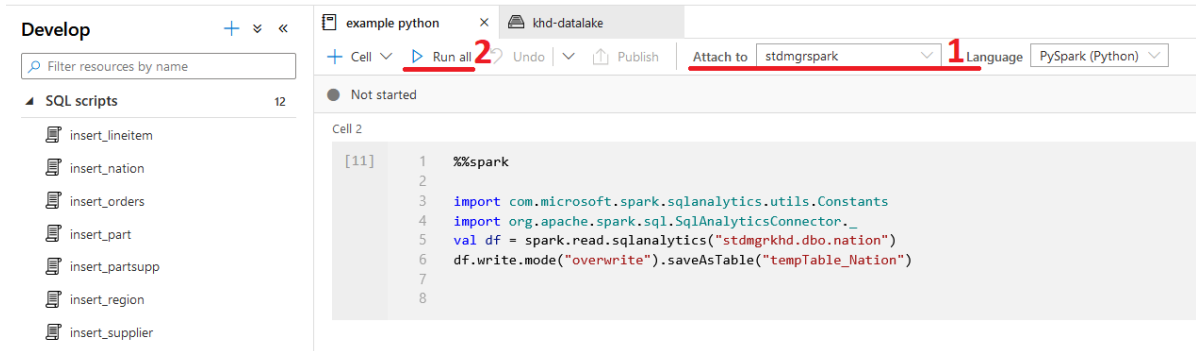
result_df.write.parquet(
```

```
"DesiredPathToParquetOnDataLake.parquet",  
partitionBy="region_name")
```

9. Tak napisany kod można uruchomić w Azure Synapse Analytics Workbench w utworzonym wcześniej notebooku. Należy wybrać stworzony Apache Spark pool (1) i kliknąć "Run all" (2) w celu uruchomienia każdej komórki zaczynając od góry.

Przygotowany notebook z omówionym kodem można znaleźć na githubie:

<https://github.com/Put-EngineeringThesis-soccer-prediction/SantanderAzure/tree/master/Spark>



The screenshot displays the Azure Synapse Analytics Workbench interface. On the left, there is a sidebar with a search bar and a list of SQL scripts under the heading "SQL scripts" (12 items). The main area shows a notebook cell titled "Cell 2" with the following code:

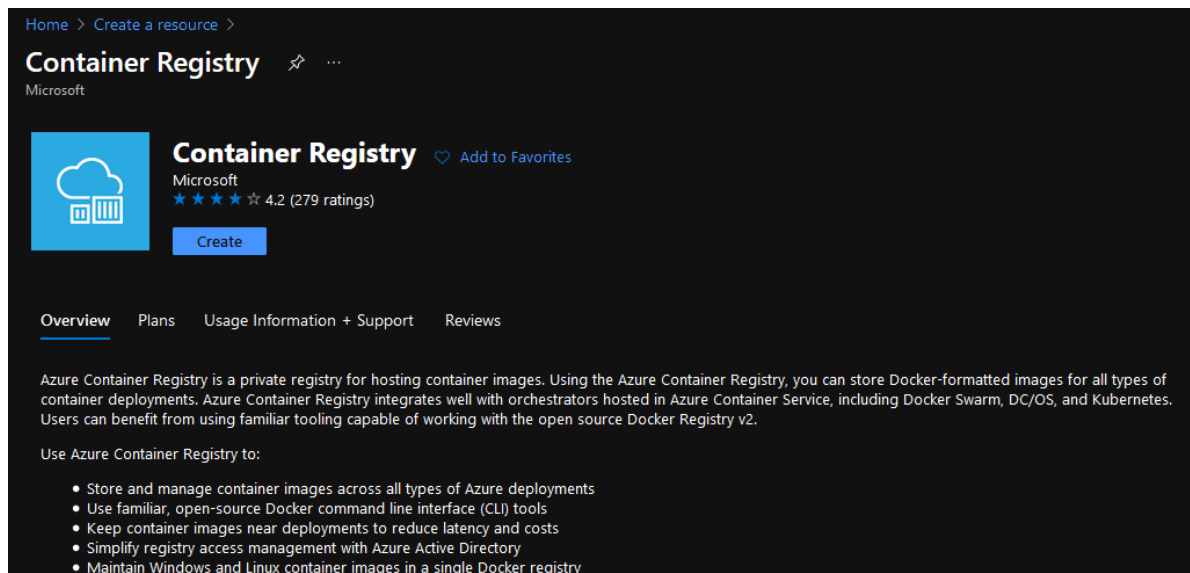
```
[11] 1  %%spark  
      2  
      3  import com.microsoft.spark.sqlanalytics.utils.Constants  
      4  import org.apache.spark.sql.SqlAnalyticsConnector._  
      5  val df = spark.read.sqlanalytics("stdmgrkhd.dbo.nation")  
      6  df.write.mode("overwrite").saveAsTable("tempTable_Nation")  
      7  
      8
```

The interface also shows a toolbar with a "Run all" button (2) and a dropdown menu for "Attach to" (1) set to "stdmgrspark". The language is set to "PySpark (Python)".

## 8 Wgrywanie obrazów dockera na Azure

Platforma Azure udostępnia narzędzia pozwalające na przechowywanie obrazów docker'a oraz ich uruchamianie w ramach instancji.

1. W celu utworzenia repozytorium dockera wyszukujemy na portalu „Container Registry” i klikamy „Create”



Home > Create a resource >

### Container Registry

Microsoft

**Container Registry** [Add to Favorites](#)

Microsoft  
★★★★☆ 4.2 (279 ratings)

[Create](#)

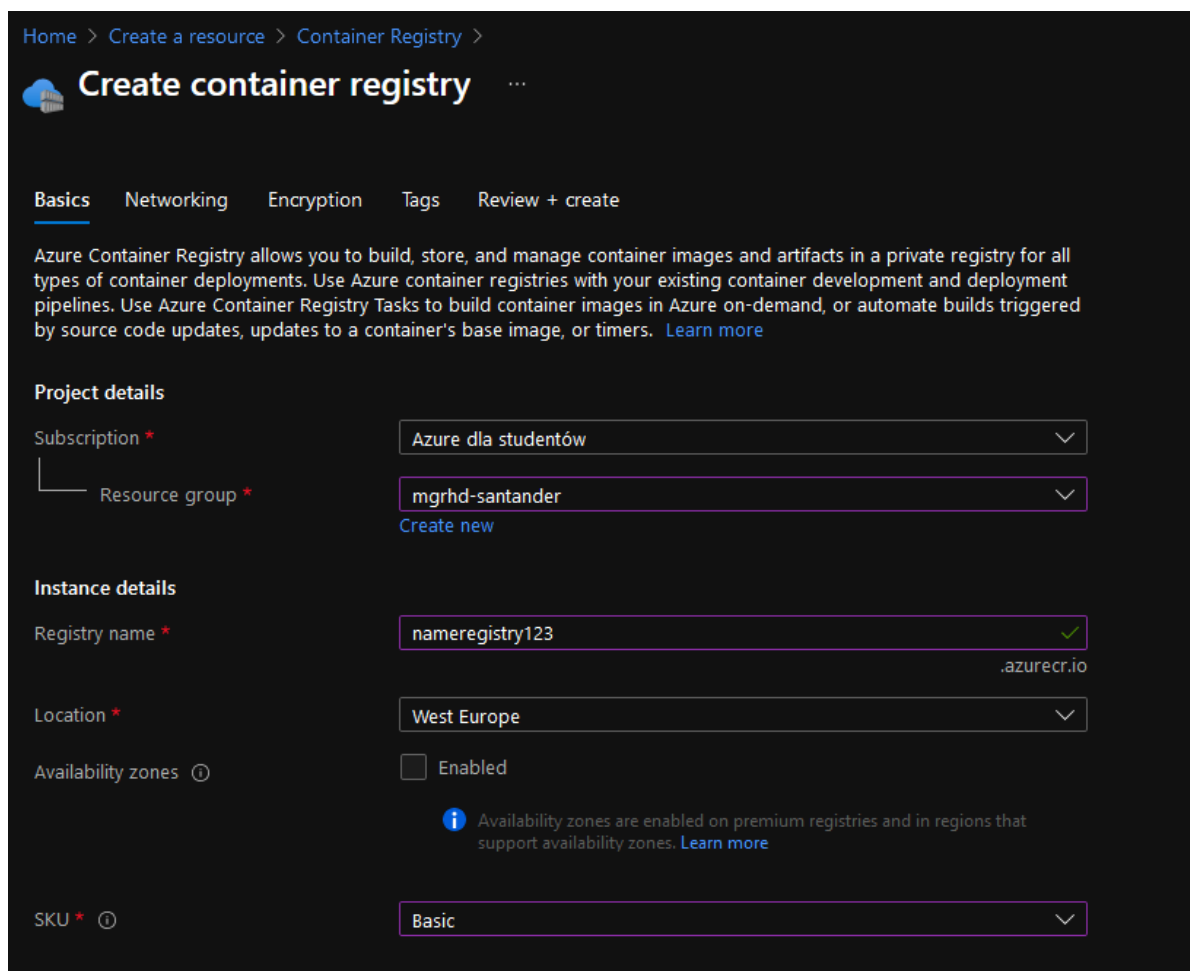
[Overview](#) [Plans](#) [Usage Information + Support](#) [Reviews](#)

Azure Container Registry is a private registry for hosting container images. Using the Azure Container Registry, you can store Docker-formatted images for all types of container deployments. Azure Container Registry integrates well with orchestrators hosted in Azure Container Service, including Docker Swarm, DC/OS, and Kubernetes. Users can benefit from using familiar tooling capable of working with the open source Docker Registry v2.

Use Azure Container Registry to:

- Store and manage container images across all types of Azure deployments
- Use familiar, open-source Docker command line interface (CLI) tools
- Keep container images near deployments to reduce latency and costs
- Simplify registry access management with Azure Active Directory
- Maintain Windows and Linux container images in a single Docker registry

2. Wybieramy subskrypcje oraz grupę zasobów (może być utworzona nowa). Następnie nazywamy nasze repozytorium oraz możemy wybrać SKU (jakość repozytorium).



Home > Create a resource > Container Registry >

### Create container registry

[Basics](#) [Networking](#) [Encryption](#) [Tags](#) [Review + create](#)

Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments. Use Azure container registries with your existing container development and deployment pipelines. Use Azure Container Registry Tasks to build container images in Azure on-demand, or automate builds triggered by source code updates, updates to a container's base image, or timers. [Learn more](#)

**Project details**

Subscription \*

Resource group \*  [Create new](#)

**Instance details**

Registry name \*  [.azurecr.io](#)

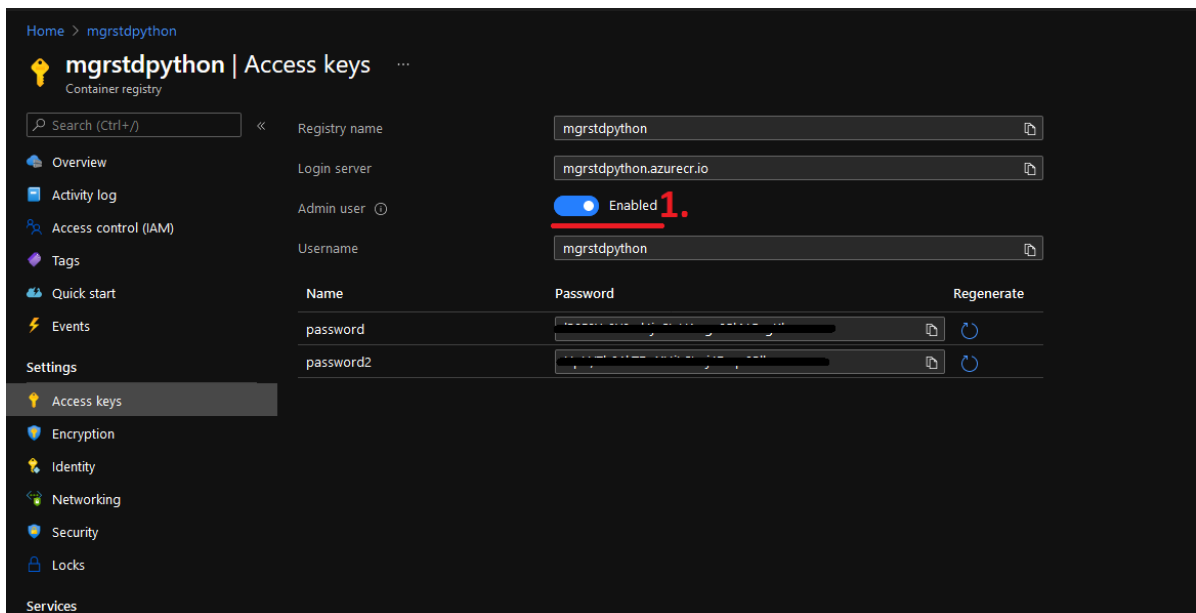
Location \*

Availability zones ⓘ  Enabled

**SKU** \* ⓘ

Availability zones are enabled on premium registries and in regions that support availability zones. [Learn more](#)

3. Przechodzimy do utworzonego przez nas repozytorium na Azure. Wybieramy w menu po lewej stronie „Access Keys” i ustawiamy „Admin user” na „Enabled” w celu umożliwienia logowania się do repozytorium. Zachowujemy jedno z hasłem oraz nazwę użytkownika na przyszłość (zawsze można tu wrócić i je skopiować).



4. W menu po lewej wybieramy opcje „Quick start”. Możemy znaleźć tutaj sposób na dodanie gotowego obrazu dockerowego do repozytorium.

## Instructions for Getting Started

- 1** Install Docker  
Before you can try out the Azure Container Registry, you should install Docker.  
Refer to the [Mac](#) or [Windows](#) or [Linux](#) getting started instructions for Docker.
- 2** Run the "hello-world" base image  
The following command will get you a running container straight off [Docker Hub](#).  
This should display "Hello from Docker!".

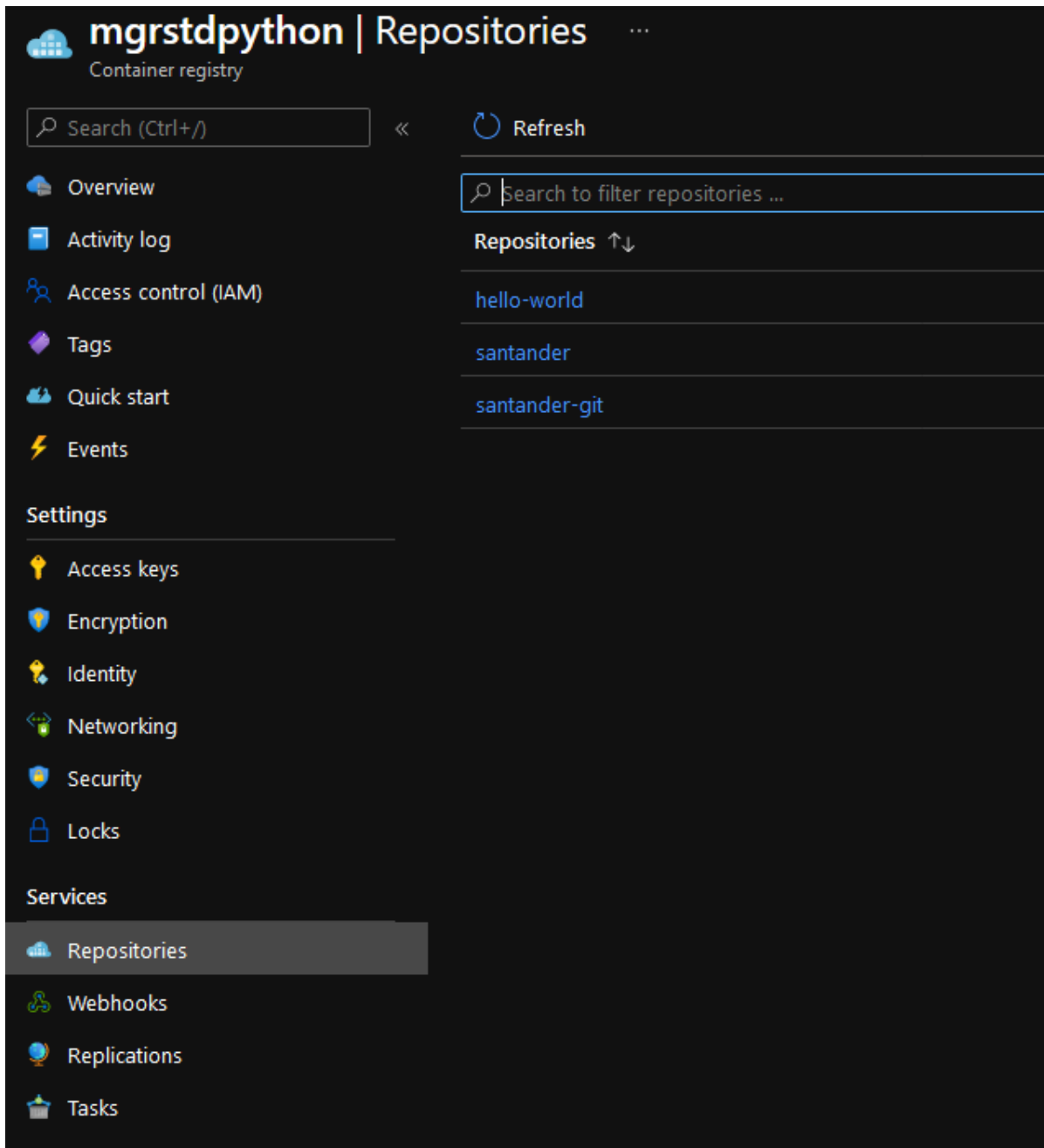
```
docker run -it hello-world
```
- 3** Login to your container registry  
Let's login to your container registry. You can get the login URI and credentials from Access keys blade.

```
docker login mgrstdpython.azurecr.io
```
- 4** Push to your registry  
Let's first prefix the image with your registry login URI so that it can be pushed to your private registry.  
The first command tags the image for uploading to your registry and the second pushes the image.

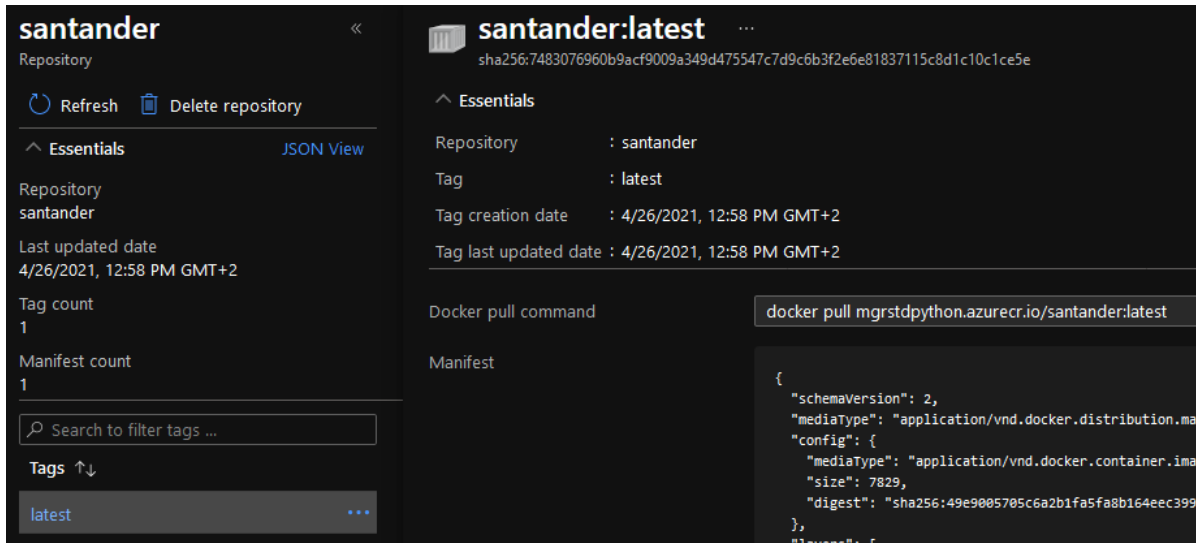
```
docker tag hello-world mgrstdpython.azurecr.io/hello-world  
docker push mgrstdpython.azurecr.io/hello-world
```
- 5** Pull from your registry  
Let's try to pull the image from your registry. You should find that your image is up to date.

```
docker pull mgrstdpython.azurecr.io/hello-world
```
- ✓** And you're ready!  
You now have a working Azure Container Registry for your applications!  
To learn more about Azure Container Registry visit [Azure Container Registry Documentation](#).

5. Po wgraniu obrazu możemy go obejrzeć w zakładce „Repositories”.



6. Klikając na tak dodany obraz możemy znaleźć jego tagi i sposób na ich pobranie.



The screenshot shows the Docker Hub interface for the 'santander:latest' image. The left sidebar displays repository information: 'santander' repository, last updated on 4/26/2021, 12:58 PM GMT+2, with 1 tag and 1 manifest. The main area shows 'Essentials' for the 'santander:latest' tag, including the repository name, tag name, creation date, and last updated date. Below this, the Docker pull command is shown as 'docker pull mgrstdpython.azurecr.io/santander:latest'. The manifest section displays a JSON snippet with fields like 'schemaVersion', 'mediaType', 'config', 'size', and 'digest'.

## 9 Przykładowy obraz dockerowy

Aby utworzyć obraz dockerowy, który można umieścić w Azurowym repozytorium, należy lokalnie stworzyć plik Dockerfile, który stanowi szablon dla tworzonego obrazu, a następnie na jego postawie zbudować go przy pomocy komendy `docker build`. Na poniższym listingu przedstawiony został przykładowy plik tego typu, wykorzystywany przy realizacji tego zadania.

```
FROM python:3.7

COPY requirements.txt .
COPY my.azureauth .

RUN apt-get update && \
    apt-get upgrade -y

# Install git
RUN apt-get install -y git

# Needed for pyodbc
RUN apt-get install -y unixodbc-dev

# Install libraries from requirements file
RUN pip install -r requirements.txt

# Install libraries for interacting with Azure
RUN pip install xlswriter azure-storage-file-datalake azure-mgmt azure-common

# Install ODBC 17
RUN apt-get install sudo
RUN curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -
RUN curl https://packages.microsoft.com/config/debian/10/prod.list >
    /etc/apt/sources.list.d/mssql-release.list
RUN sudo apt-get update
RUN sudo ACCEPT_EULA=Y apt-get install -y msodbcsql17
RUN sudo ACCEPT_EULA=Y apt-get install -y mssql-tools
RUN echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc

# Clone the repository with scripts
RUN git clone
    https://github.com/Put-EngineeringThesis-soccer-prediction/SantanderAzure
```



Na początku wybieramy obraz macierzysty, który będzie podstawą naszego obrazu. W naszym przypadku naszym wzorcem będzie obraz z zainstalowanym Pythonem w wersji 3.7. Jest to system bazujący na Debianie w wersji 10. Następnie kopiowane są pliki z lokalnego systemu plików, które znajdują się w utworzonym obrazie - pliki *requirements.txt* oraz *my.azureauth*. Są to odpowiednio pliki zawierające zależności do Pythona oraz plik zawierający informacje potrzebne do uwierzytelnienia w Azure.

W następnej kolejności instalowane są potrzebne komponenty - m.in. *git*, *ODBC 17* oraz biblioteki z pliku *requirements.txt* i te potrzebne do interakcji z różnymi komponentami chmurowymi.

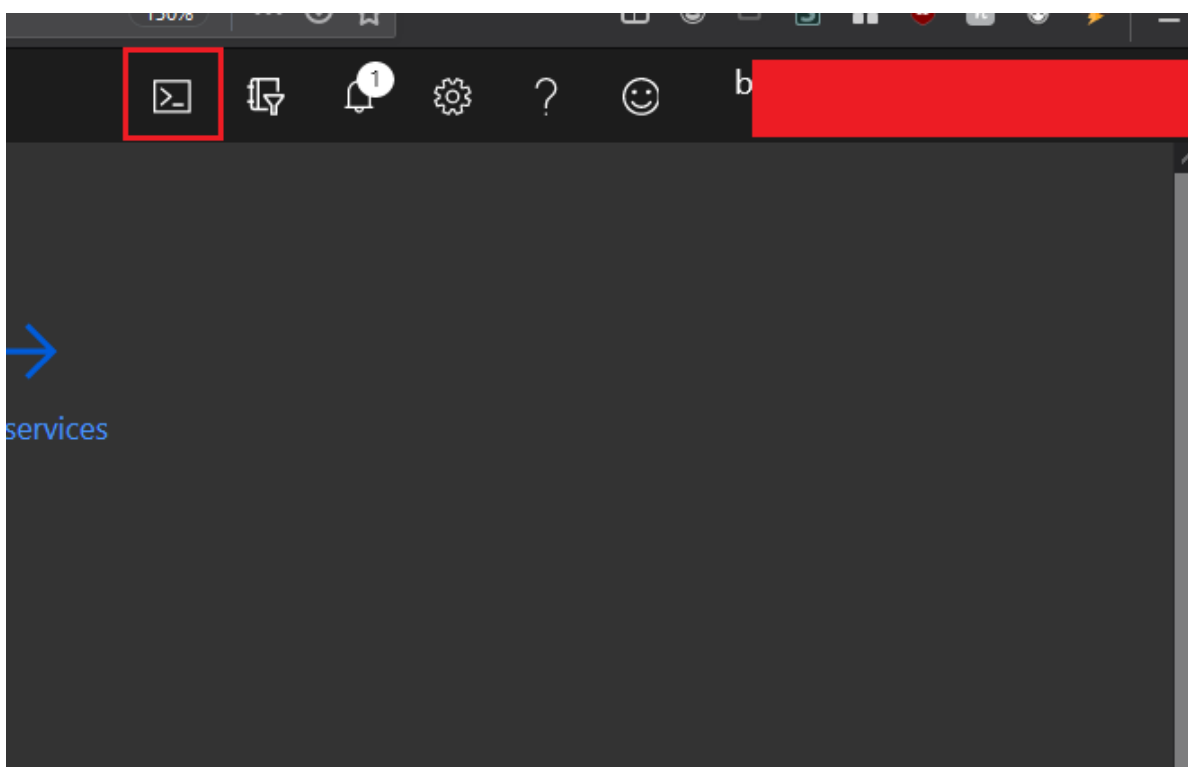
W ostatnim kroku pobierane jest repozytorium ze skryptami, które będzie wykorzystane w przypadku automatyzacji. Odpowiednie narzędzie będzie uruchamiało kontener dla tego obrazu i uruchamiało odpowiednie skrypty zgodnie z harmonogramem i zadanymi parametrami.

Zbudowany w ten sposób obraz można umieścić w repozytorium w sposób, który został omówiony w poprzednim podrozdziale przy pomocy komend *docker tag* oraz *docker push*.

## 9.1 Generowanie pliku *my.azureauth*

Przy budowie obrazu dockera został użyty plik *my.azureauth*. Jest to plik, który umożliwia skryptowi napisanemu w Pythonie autoryzację w Azure i manipulowanie zasobami. W naszym przypadku jest to potrzebne aby skrypt, który zarządza przetwarzaniem mógł usunąć instancje kontenera po zakończonej pracy. Aby go wygenerować należy:

1. Przejść do portalu Azure i kliknąć zaznaczoną ikonę. Uruchomi to Azure Cli, który umożliwia wykonanie instrukcji w powershellu. Podczas pierwszego uruchomienia może poprosić o utworzenie zasobów dla tego narzędzia. Wystarczy się zgodzić na wszystko.



## 2. Wpisać instrukcje `az ad sp create-for-rbac --sdk-auth`

Wygeneruje ona plik json zawierający dane autoryzujące. Wygenerowany json należy przekopiować do pliku, w naszym przypadku jest to plik `my.azureauth`. Ten plik należy wgrać na obraz dockera, aby był dostępny dla skryptu zarządzającego przetwarzaniem.

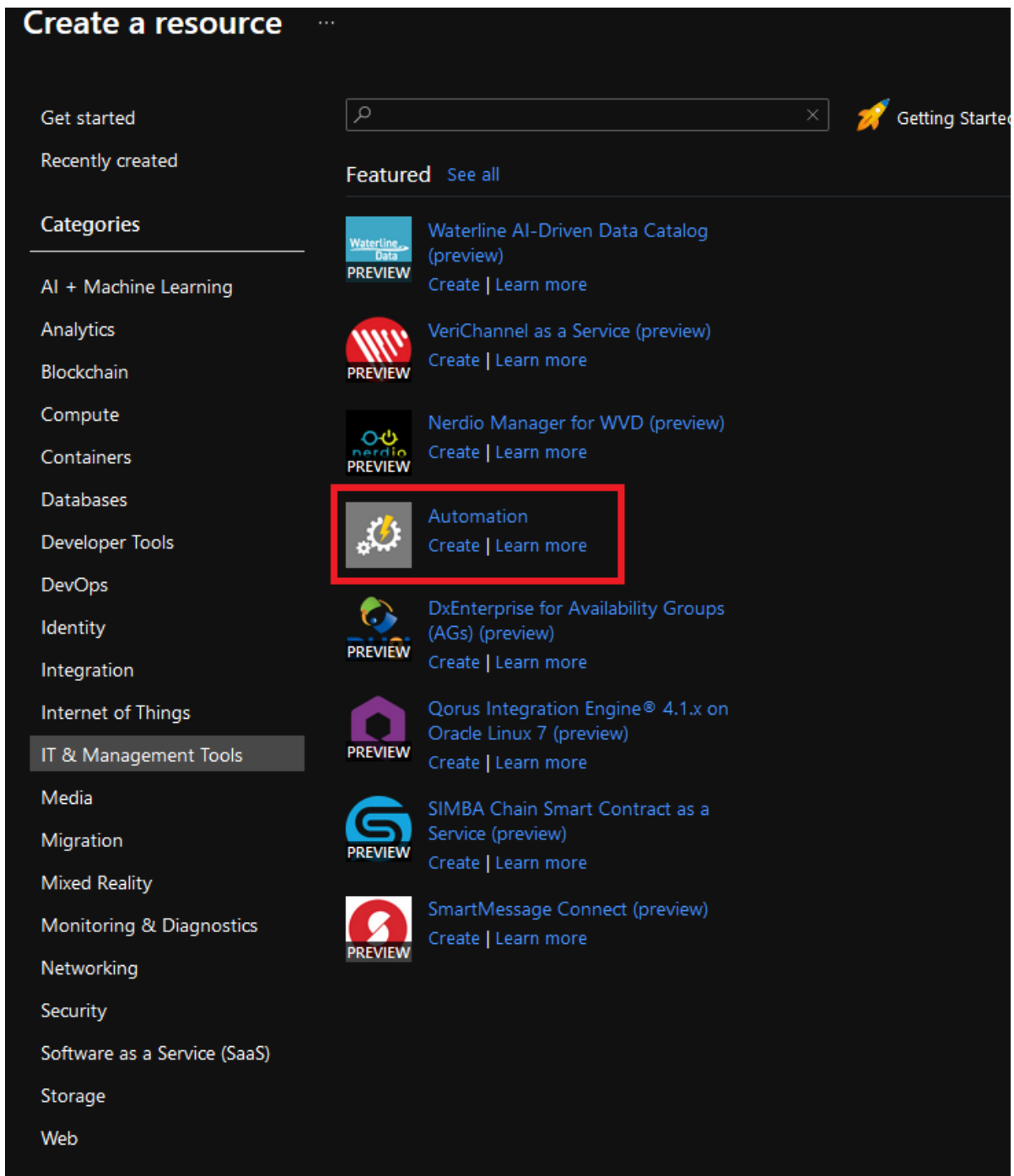
```
PowerShell | ? ? ? ? ? ? ? ?
MOTD: Manage Azure Active Directory: Get-Command -Module AzureAD*
VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS /home/ciesielski> az ad sp create-for-rbac --sdk-auth
In a future release, this command will NOT create a 'Contributor' role assignment
Creating 'Contributor' role assignment under scope '/subscriptions/afb7883d-d
The output includes credentials that you must protect. Be sure that you do not
zadsp-cli
{
  "clientId": "c83e...",
  "clientSecret": "vEkCM...",
  "subscriptionId": "afb...",
  "tenantId": "14cb4ab4-...",
  "activeDirectoryEndpointUrl": "https://login.microsoftonline.com",
  "resourceManagerEndpointUrl": "https://management.azure.com/",
  "activeDirectoryGraphResourceId": "https://graph.windows.net/",
  "sqlManagementEndpointUrl": "https://management.core.windows.net:8443/",
  "galleryEndpointUrl": "https://gallery.azure.com/",
  "managementEndpointUrl": "https://management.core.windows.net/"
}
PS /home/ciesielski> █
```

## 10 Automatyzowanie obrazów dockera

Jednym z natywnych narzędzi Azure jest tzw. „Automation Account”, który umożliwia tworzenie runbook'ów, które można sparametryzować i uruchamiać w nich kod powershell lub python. Taki runbook może mieć ustawiany harmonogram z parametrami.

W kodzie powershell'owym jest możliwe utworzenie instancji docker'a na podstawie obrazu wgranego na wcześniej utworzone repozytorium obrazów. Wykorzystując te wszystkie funkcje możemy zautomatyzować uruchomienie się kodów pythonowych w dockerze.

1. W celu utworzenia „Automation account” dodajemy nowy zasób i w zakładce „IT & Management Tools” wybieramy „Automation”.



2. Tworzymy zasób wybierając nazwę, subskrypcje, grupę zasobów oraz lokalizację.

Home > Create a resource >

# Add Automation Account

Name \* ⓘ

automate-santander ✓

Subscription \*

Visual Studio Enterprise – MPN

Resource group \*

test-docker

Create new

Location \*

West Europe

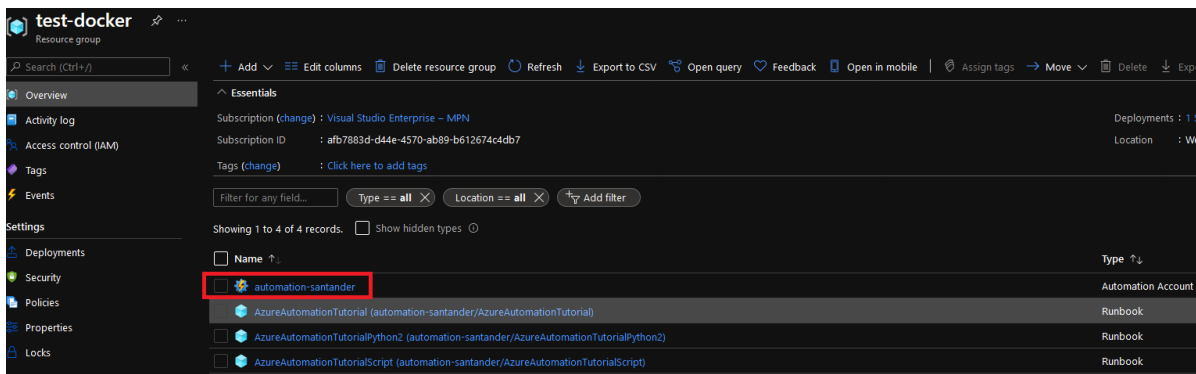
Create Azure Run As account \* ⓘ

Yes No

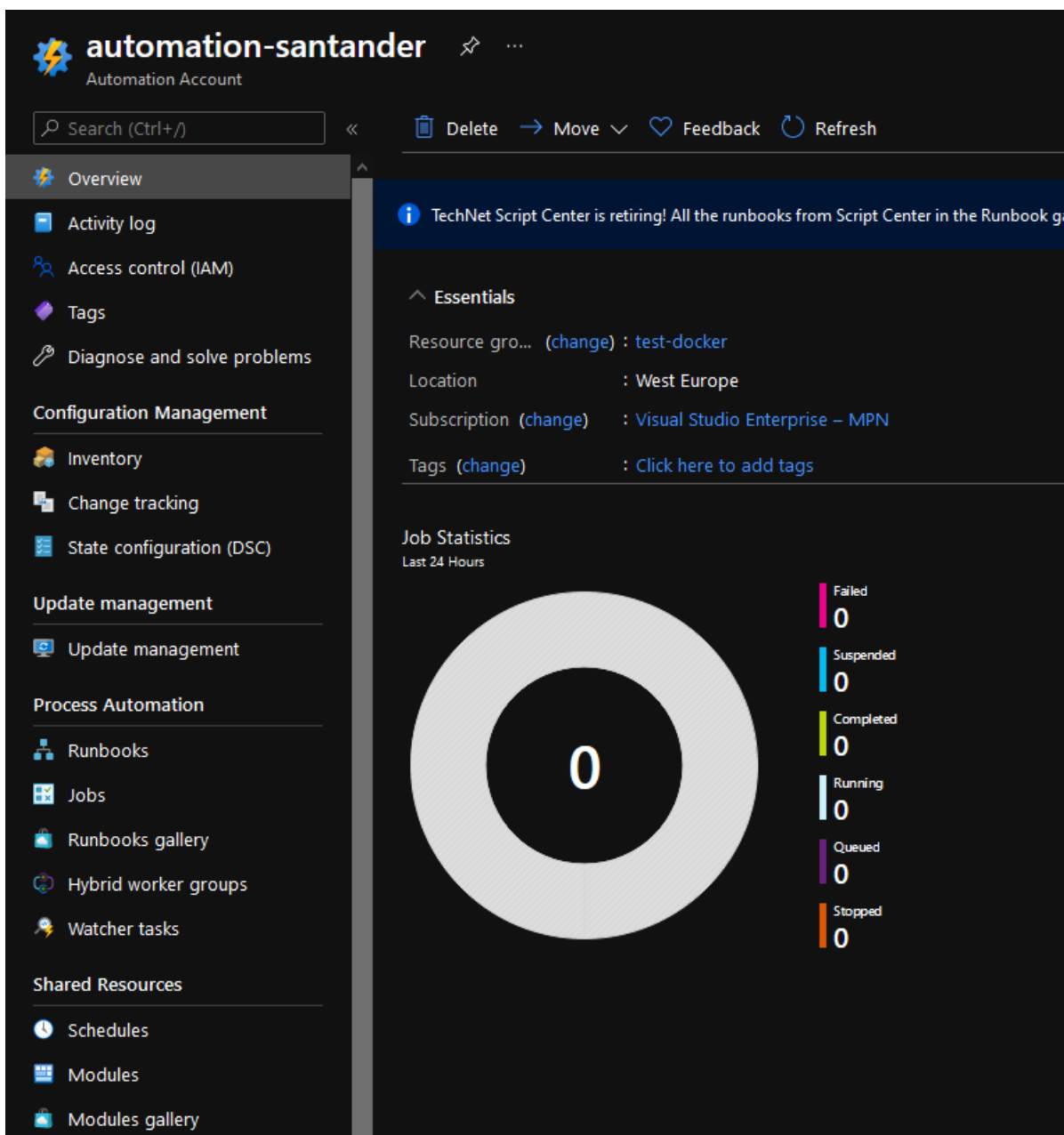
**i** This will create Azure Run As account in the Automation account which are useful for authenticating with Azure to manage Azure resources from Automation runbooks. Note that the creation of Azure Run As account may affect the security of the subscription.[Learn more](#)

**i** Learn more about Automation pricing.

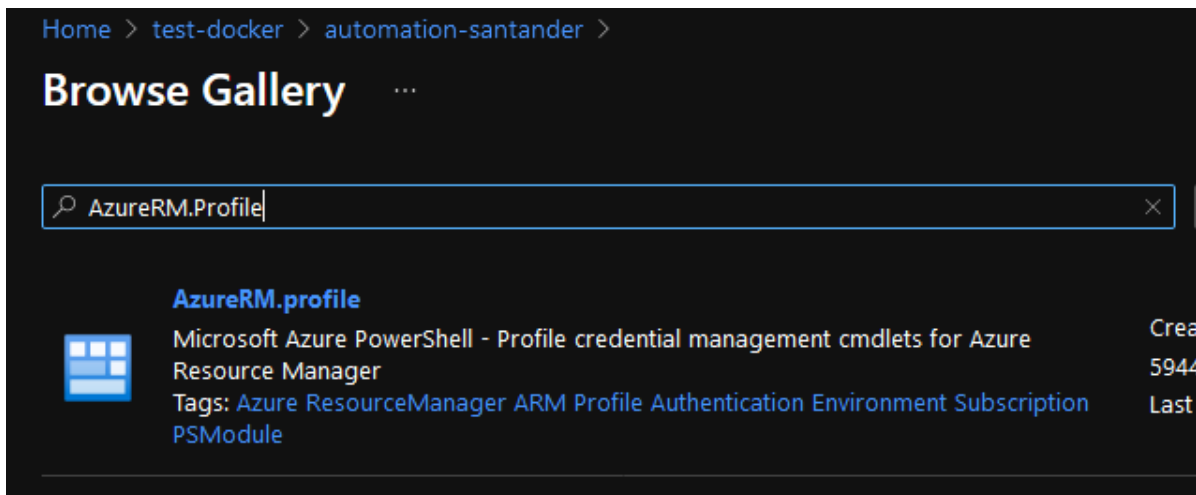
3. Wybieramy utworzony przez nas zasób. Dodatkowo zostaną utworzone przykładowe runbooki, można je usunąć albo zostawić i zignorować.



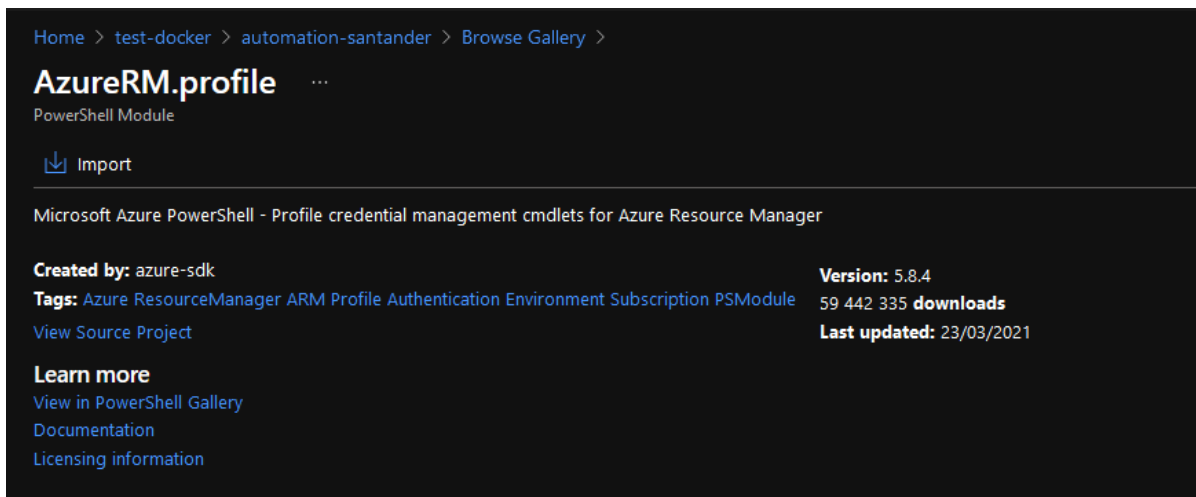
4. W menu po lewej stronie wybieramy „Modules gallery” w celu zainstalowania wymaganych modułów do manipulacji kontenerami na Azure.



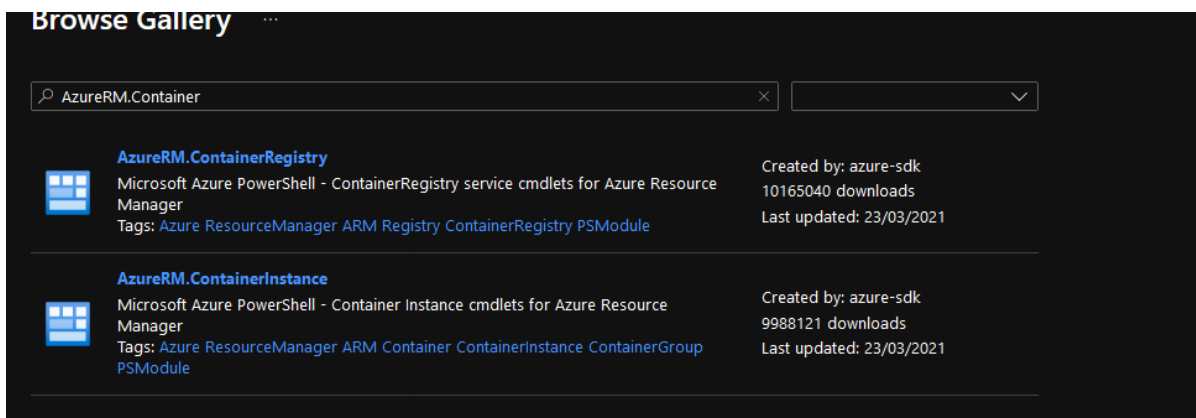
5. Pierwszym modulem, który musimy dograć jest „AzureRm.Profile”, wyszukujemy go i wybieramy z listy.



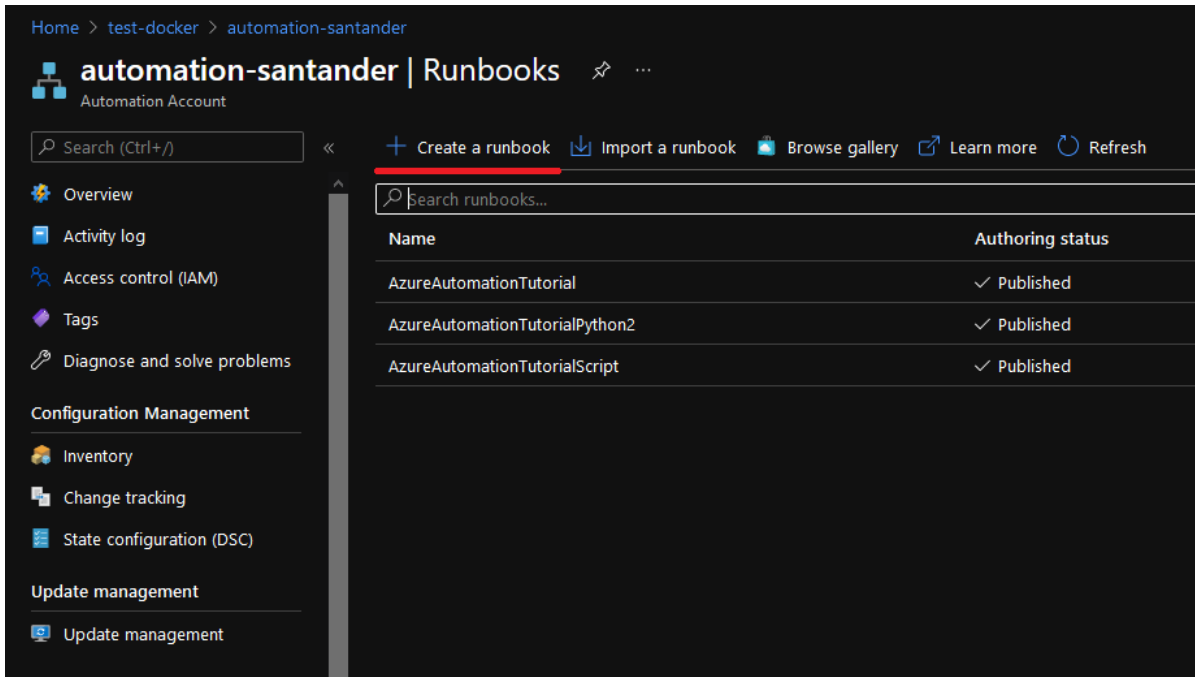
6. Klikamy guzik „Import” i importujemy moduł. Może to zająć kilka minut i dopóki się nie skończy nic nowego nie zainstalujemy.



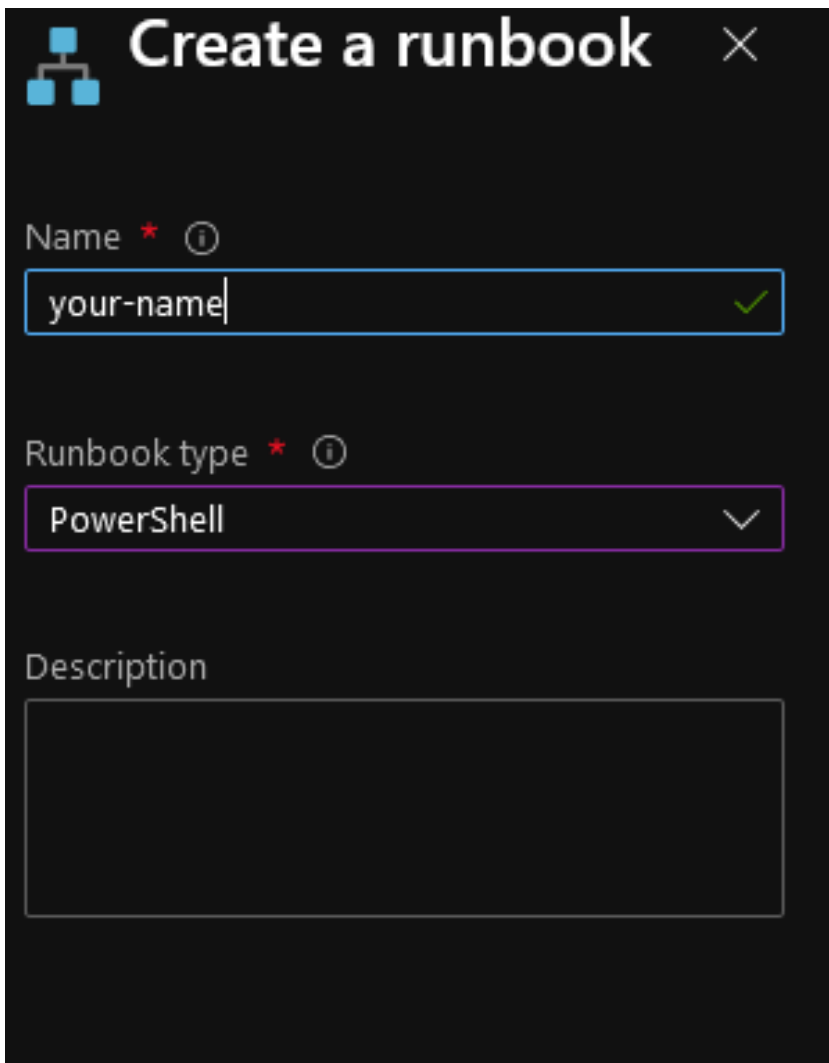
7. Po zainstalowaniu modułu „AzureRm.Profile” wyszukujemy w galerii następny moduł „AzureRM.Container” i postępujemy tak samo jak z poprzednim modulem.



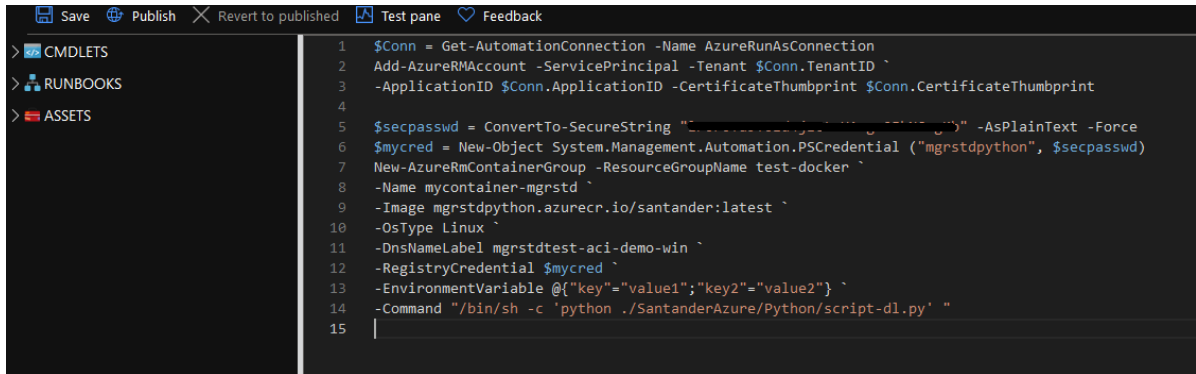
8. Kiedy wszystko jest gotowe przechodzimy do zakładki „Runbooks” i klikamy „Create a runbook”.



9. Nadajemy mu nazwę i wybieramy „Runbook type” jako Powershell.



10. W tak utworzonym runbook'u możemy wpisać kod tworzący naszą instancję obrazu kontenera.



```
1 $Conn = Get-AutomationConnection -Name AzureRunAsConnection
2 Add-AzureRMAccount -ServicePrincipal -Tenant $Conn.TenantID `
3 -ApplicationID $Conn.ApplicationID -CertificateThumbprint $Conn.CertificateThumbprint
4
5 $secpasswd = ConvertTo-SecureString "(...)" -AsPlainText -Force
6 $mycred = New-Object System.Management.Automation.PSCredential ("mgrstdpython", $secpasswd)
7 New-AzureRmContainerGroup -ResourceGroupName test-docker `
8 -Name mycontainer-mgrstd `
9 -Image mgrstdpython.azurecr.io/santander:latest `
10 -OsType Linux `
11 -DnsNameLabel mgrstdtest-aci-demo-win `
12 -RegistryCredential $mycred `
13 -EnvironmentVariable @{"key"="value1";"key2"="value2"} `
14 -Command "/bin/sh -c 'python ./SantanderAzure/Python/script-dl.py' "
15 |
```

Linie 1-3 służą do autoryzacji naszego runbook'a w Azure, bez tego nie możemy zarządzać zasobami. Zmienna \$secpasswd jest hasłem do naszego konta na repozytorium dockerowym, następnie tworzymy zmienną \$mycred w której mamy parę 'login hasło'.

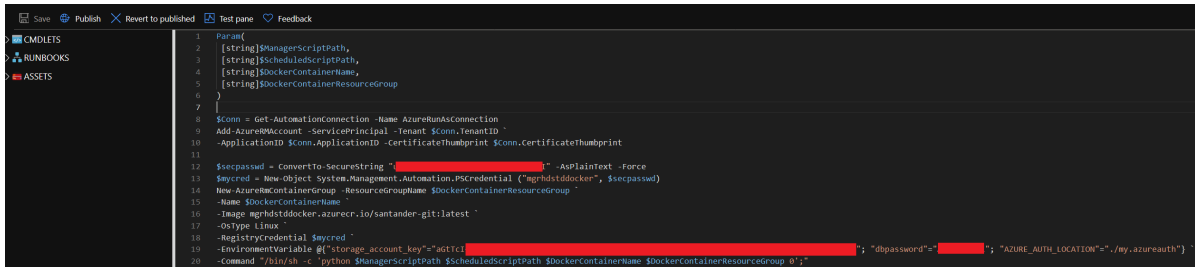
Zaczynając w linijce 7 - komenda 'New-AzureRmContainerGroup' służy do tworzenia instancji naszego obrazu dockerowego. Link do dokumentacji: [https://docs.microsoft.com/en-us/powershell/module/azurermscontainerinstance/new-azurermscontainerinstance?view=azurermps-6.13.0](https://docs.microsoft.com/en-us/powershell/module/azurermscontainerinstance/new-azurermscontainerinstance). W skrócie wybieramy nazwę grupy zasobów, gdzie będzie utworzona instancja, nazwę samej instancji a następnie wskazujemy, jaki obraz chcemy utworzyć. Musimy określić również typ systemu w obrazie, nazwa Dns nie jest wymagana. „RegistryCredential” wykorzystuje nasze dane do zalogowania się w repozytorium. Następnie możemy dodać zmienne środowiskowe oraz jaką komenda bashowa ma zostać wykonana po utworzeniu tego kontenera (np. uruchomienie odpowiedniego skryptu).



## 11 Uruchamianie skryptów Pythonowych z harmonogramem

Po utworzeniu podstawy do automatyzacji można utworzyć docelowy runbook z parametrami i ustawić harmonogram dla konkretnych skryptów. Zaczniemy od edycji skryptu powershell'a tak, aby przyjmował parametry, po czym ustawimy harmonogram dla przygotowanego skryptu, który będzie przyjmował te parametry.

1. Przejdź do przygotowanego runbook'a i kliknij *Edit*. Poniżej kod, który będzie używany przy harmonogramowanych uruchomieniach.



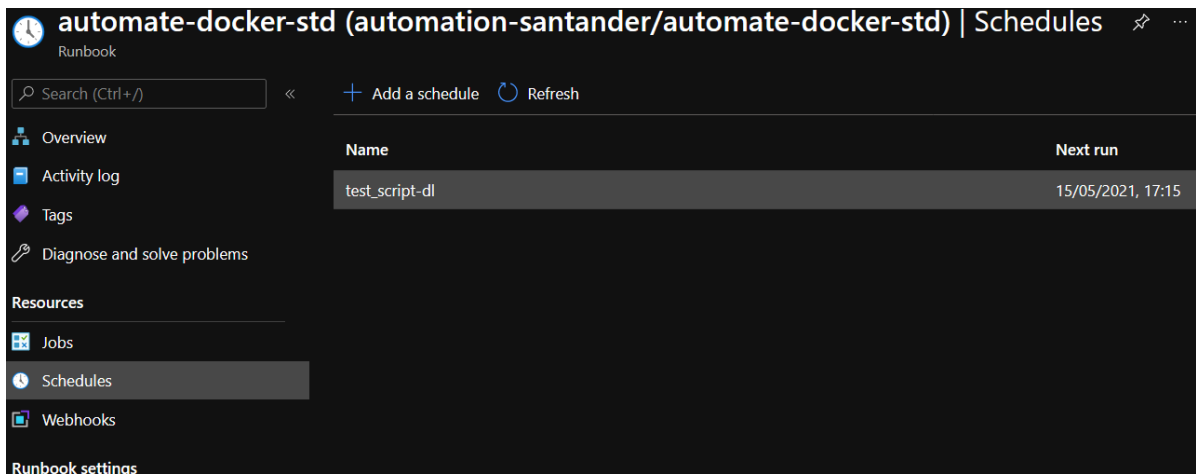
```
1 Param(
2     [string]$ManagerScriptPath,
3     [string]$ScheduledScriptPath,
4     [string]$DockerContainerName,
5     [string]$DockerContainerResourceGroup
6 )
7
8 $Conn = Get-AutomationConnection -Name AzureRunAsConnection
9 Add-AzureRmAccount -ServicePrincipal -Tenant $Conn.TenantID
10 -ApplicationID $Conn.ApplicationID -CertificateThumbprint $Conn.CertificateThumbprint
11
12 $Secpasswd = ConvertTo-SecureString " " -AsPlainText -Force
13 $Mycred = New-Object System.Management.Automation.PSCredential ("mghdstdocker", $Secpasswd)
14 New-AzureRmContainerGroup -ResourceGroupName $DockerContainerResourceGroup
15 -Name $DockerContainerName
16 -Image mghdstdocker.azurecr.io/santander-git:latest
17 -OSType Linux
18 -RegistryCredential $Mycred
19 -EnvironmentVariable @{"storage_account_key"=""; "dbpassword"=""; "AZURE_AUTH_LOCATION"="./my.azureauth"}
20 -Command "bin/sh -c 'python $ManagerScriptPath $ScheduledScriptPath $DockerContainerName $DockerContainerResourceGroup 0';"
```

Na samej górze kodu pojawiły się parametry, określamy ich typ i nazwę. Można jeszcze jeszcze dodać takie dekoratory jak *[Parameter (Mandatory= \$true/\$false)]*, domyślnie parametry są opcjonalne.

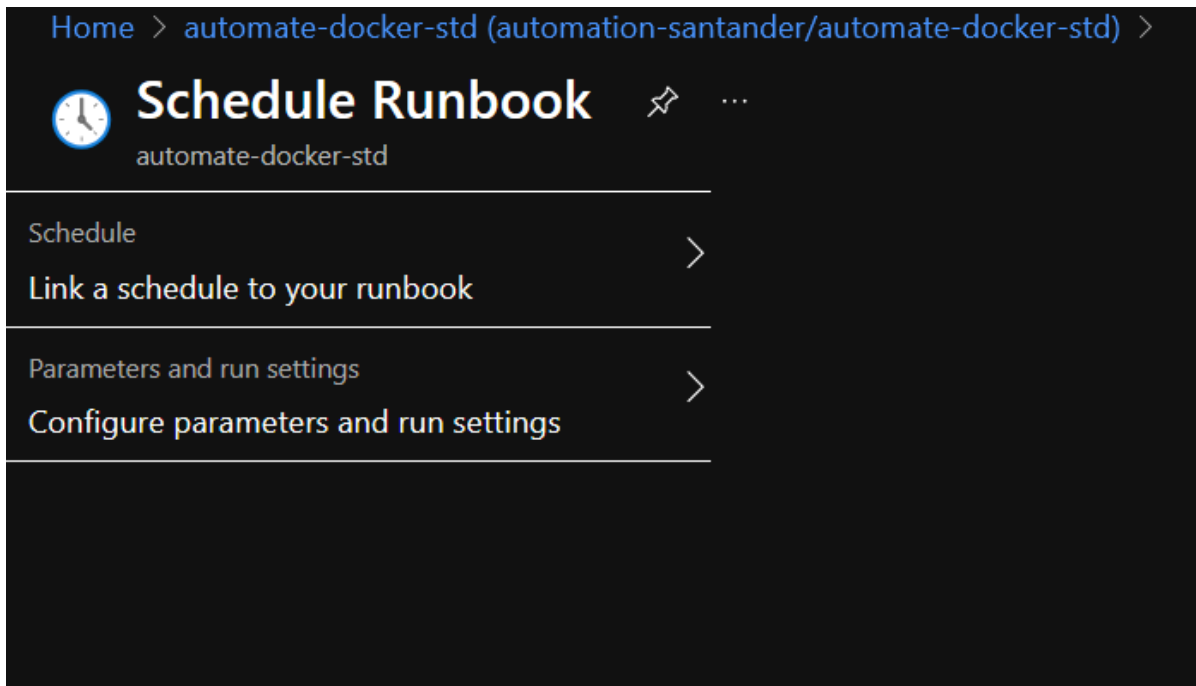
W linijce 19 określamy zmienne środowiskowe dla parametrów, które powinny być chronione. Klucz dostępu do data lake'u, hasło do bazy danych oraz lokalizacja wcześniej przygotowanego pliku my.azureauth. Te zmienne w przypadku kodu przygotowanego w ramach projektu są wymagane dla poprawnego działania skryptów.

Na końcu określamy komendę wykonaną na początku działania obrazu. Uruchamia ona skrypt przykazany w parametrze ManagerScriptPath (czyli skrypt menadżera, który później usunie kontener), następnie ścieżka do skryptu docelowego, który chcemy wykonać. Dodatkowo 2 parametry, nazwa instancji kontenera oraz grupa zasobów, na której się on znajduje. Tak przygotowany skrypt możemy zapisać i kliknąć *Publish*.

2. Po opublikowaniu skryptu wróćmy do zasobu runbook'a i z menu po lewej wybierzmy *Schedules*, a następnie kliknij przycisk *Add a schedule* w celu dodania nowego harmonogramu.



3. W środku zobaczymy poniższe menu. Klikamy *Schedule* i w środku dodajemy nowy harmonogram lub wybieramy istniejący, jeżeli już jakiś istnieje o wymaganych parametrach.



4. Po wyborze harmonogramu wypełniamy parametry wybierając *Parameters*, z jakimi danych scheduler będzie uruchamiać runbook. Poniżej widać przykładowe parametry dla skryptu script-dl.py. Uwaga, *Resource group* musi być istniejącym zasobem, utworzonym ręcznie.

The screenshot shows the 'Parameters' configuration page for a resource named 'automate-docker-std'. It features a dark background with white text. At the top left is an information icon (i) and the title 'Parameters'. Below the title are four parameter settings, each with a name, a help icon (i), a text input field, and a description. The parameters are: 1. MANAGERSRIPTPATH with value './SantanderAzure/Python/test-python-api.py' and description 'Optional, String'. 2. SCHEDULEDSCRIPTPATH with value './SantanderAzure/Python/script-dl.py' and description 'Optional, String'. 3. DOCKERCONTAINERNAME with value 'yourinstancename' and description 'Optional, String'. 4. DOCKERCONTAINERRESOURCEGROUP with value 'scheduled-docker' and description 'Optional, String'. At the bottom, there is a 'Run Settings' section with a 'Run on Azure' option and a help icon.

**Parameters**  
automate-docker-std

**MANAGERSRIPTPATH** ⓘ  
./SantanderAzure/Python/test-python-api.py  
*Optional, String*

**SCHEDULEDSCRIPTPATH** ⓘ  
./SantanderAzure/Python/script-dl.py  
*Optional, String*

**DOCKERCONTAINERNAME** ⓘ  
yourinstancename  
*Optional, String*

**DOCKERCONTAINERRESOURCEGROUP** ⓘ  
scheduled-docker  
*Optional, String*

**Run Settings**  
Run on Azure ⓘ

W ten sposób dodaliśmy harmonogram, który powinien uruchomić podany skrypt o zadanej godzinie, po czym usunie kontener który go wykonywał.

## 12 Skrypt Menadżera

W poprzednim rozdziale użyty został skrypt, który zarządza uruchamianiem innych skryptów na instancjach i usuwaniem kontenera. Jest to bardzo prosty skrypt, którego treść widać poniżej.

```
import os
import sys
import subprocess
from azure.common.client_factory import get_client_from_auth_file
from azure.mgmt.resource import ResourceManagementClient
from azure.mgmt.containerinstance import ContainerInstanceManagementClient

# Retrieve the IDs and secret to use with ClientSecretCredential
arguments = sys.argv[1:]
script_to_run = arguments[0]
container_name = arguments[1]
resource_group_name = arguments[2]

subprocess.run(["python", str(script_to_run)])

auth_file_path = os.getenv('AZURE_AUTH_LOCATION', None)
print(auth_file_path)
if auth_file_path is not None:
    print("Authenticating with Azure using credentials in file at {}".format(auth_file_path))

    aci_client = get_client_from_auth_file(
        ContainerInstanceManagementClient)
    res_client = get_client_from_auth_file(ResourceManagementClient)
else:
    print("\nFailed to authenticate to Azure. Have you set the")

aci_client.container_groups.delete(resource_group_name, container_name)
```

Najpierw importujemy potrzebne biblioteki, po czym czytamy argumenty z wejścia. Uruchamiamy skrypt docelowy i po zakończeniu jego działania autoryzujemy się korzystając z wcześniej przygotowanego pliku o nazwie *my.azureauth*, do którego ścieżka jest zapisana pod zmienną środowiskową **AZURE\_AUTH\_LOCATION**. Jeżeli autoryzacja się powiodła, korzystając z biblioteki Microsoftu, usuwamy podany w parametrach kontener. Operacja ta jest błyskawiczna, po wykonaniu skryptu nie ma dostępu do wcześniej uruchomionego kontenera.

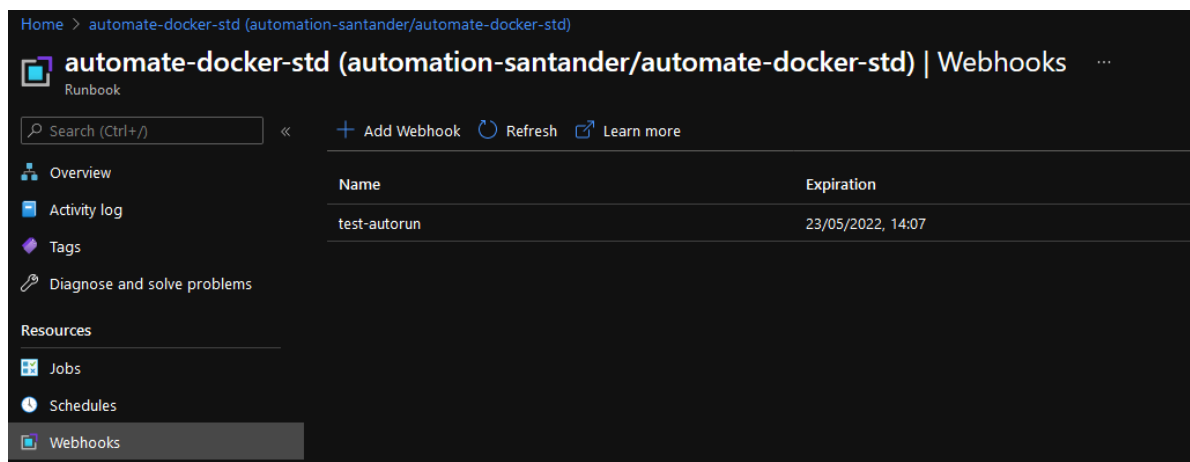
## 13 Uruchamianie skryptów Python w ETL

### 13.1 Generowanie Webhook dla runbook'a

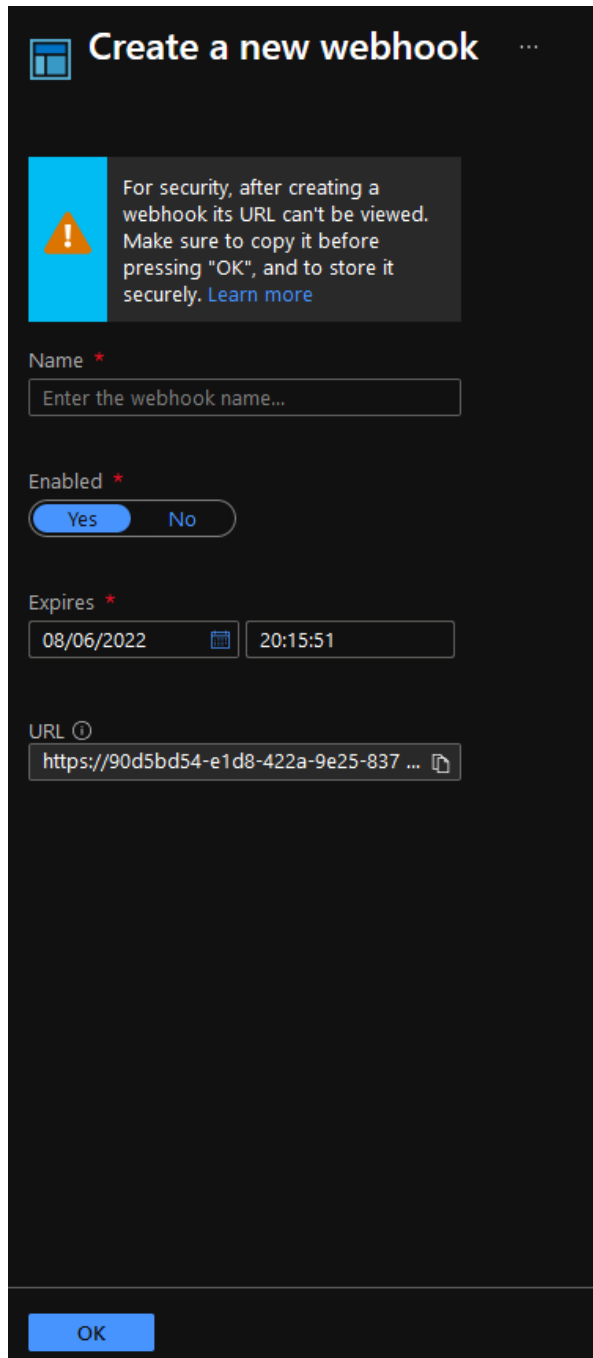
W celu uruchomienia runbook'a, który odpowiada za uruchomienie przetwarzania raportów spoza portalu Azure, możemy skorzystać z kilku metod. Szczegółowo są one opisane w dokumentacji Microsoftu <https://docs.microsoft.com/en-us/azure/automation/start-runbooks>.

Wykorzystana została metoda „Webhooks”, w której generujemy link służący do uruchamiania przetwarzania.


1. W przygotowanym wcześniej runbooku przechodzimy do zakładki „WebHooks” i w niej klikamy guzik „Add Webhook” w celu utworzenia sparametryzowanego adresu URL.



2. Następnie w pierwszej opcji wybieramy nazwę naszego webhook'a, określamy czy jest aktywny i wybieramy dzień w którym wygaśnie automatycznie. **Uwaga, trzeba zapisać URL w tym kroku. Zgodnie z tym, co jest napisane nie ma możliwości skopiowania go później.**



**Create a new webhook** ...

 For security, after creating a webhook its URL can't be viewed. Make sure to copy it before pressing "OK", and to store it securely. [Learn more](#)

Name \*

Enter the webhook name...

Enabled \*

Yes No

Expires \*

08/06/2022 20:15:51

URL ⓘ

https://90d5bd54-e1d8-422a-9e25-837 ...

OK

- Ostatnim zadaniem podczas tworzenia webhook'a jest określenie parametrów, z jakim zostanie uruchomiony runbook. W przypadku naszych skryptów, trzeba określić ścieżki do skryptu menadżera, skryptu do uruchomienia, nazwę kontenera i grupę zasobów dla tego kontenera.

**Parameters**  
automate-docker-std

**Parameters**

MANAGERSCRIPTPATH ⓘ  
No value  
*Optional, String*

SCHEDULEDSCRIPTPATH ⓘ  
No value  
*Optional, String*

DOCKERCONTAINERNAME ⓘ  
No value  
*Optional, String*

DOCKERCONTAINERRESOURCEGROUP ⓘ  
No value  
*Optional, String*

**Run Settings**

Run on Azure ⓘ

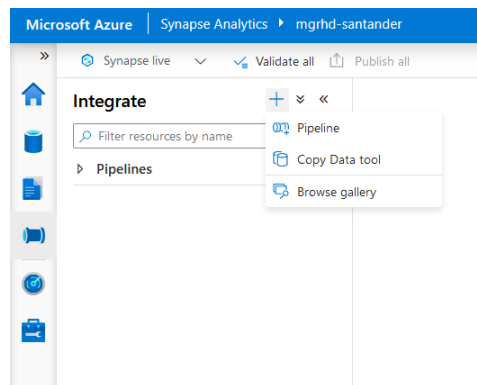
OK

Z tak przygotowanym webhook'iem możemy przejść do następnego kroku, czyli zintegrowanie go do procesu ETL.

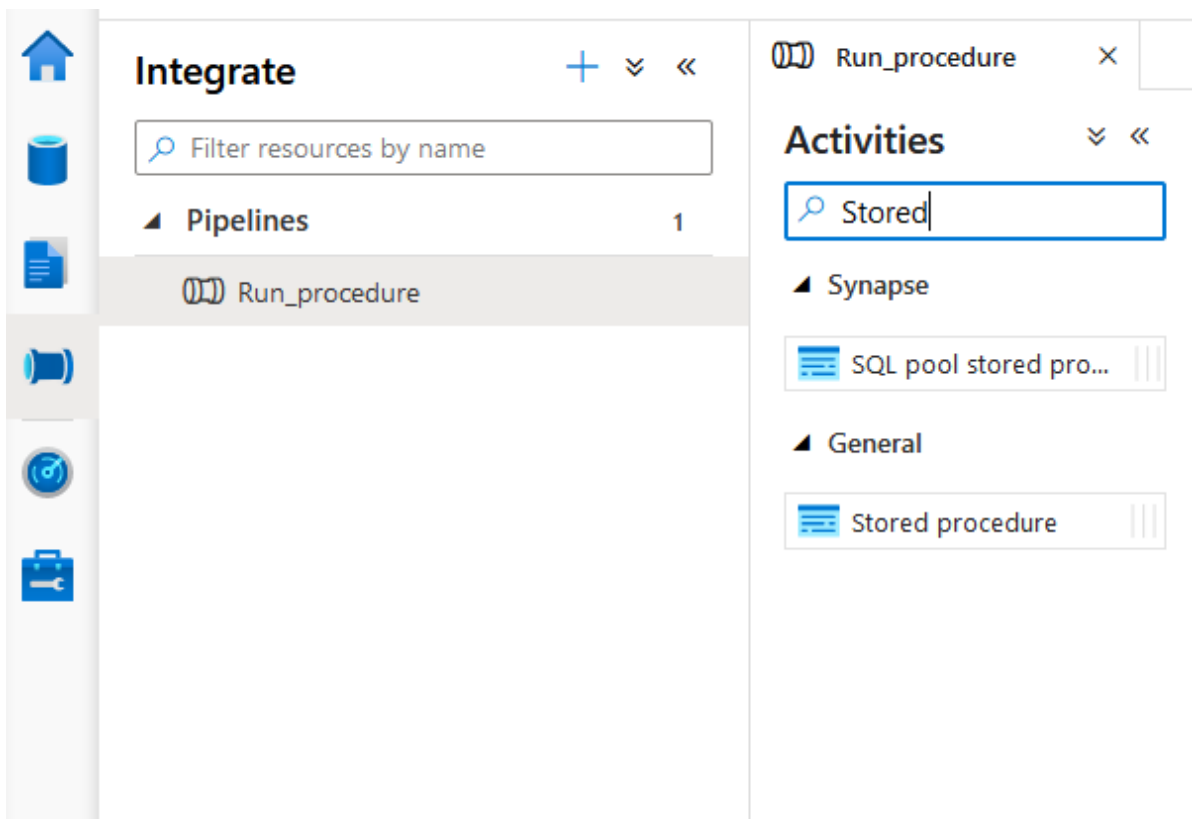
## 13.2 Przygotowanie prostego procesu ETL

Proces ETL można przygotować bezpośrednio w portalu Synapse Analytics Workbench, więc dla prostoty został tam utworzony jasny pipeline. Należy jednak pamiętać, że webhook można wykorzystać w każdym systemie, który ma możliwość wykonywania żądań HTTP.

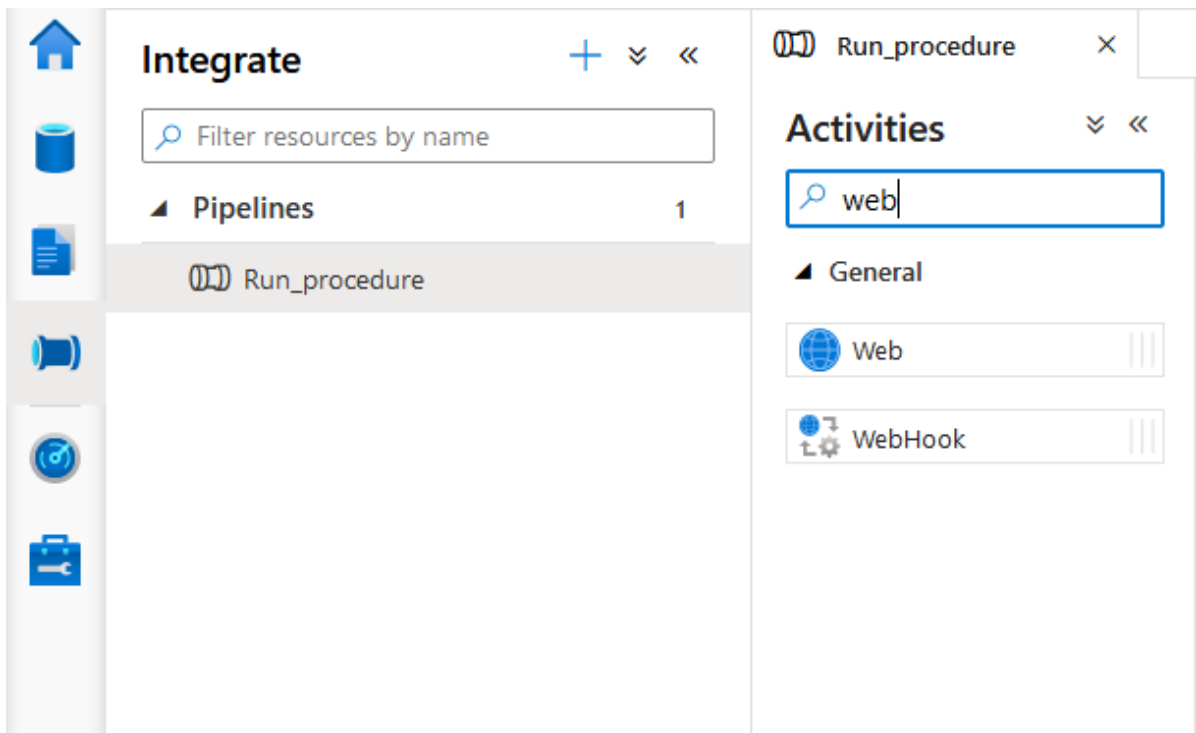
1. W utworzonym Synapse Analytics przechodzimy do zakładki Integrate i klikamy na plusik. Tam wybieramy „Pipeline” w celu utworzenia procesu ETL.



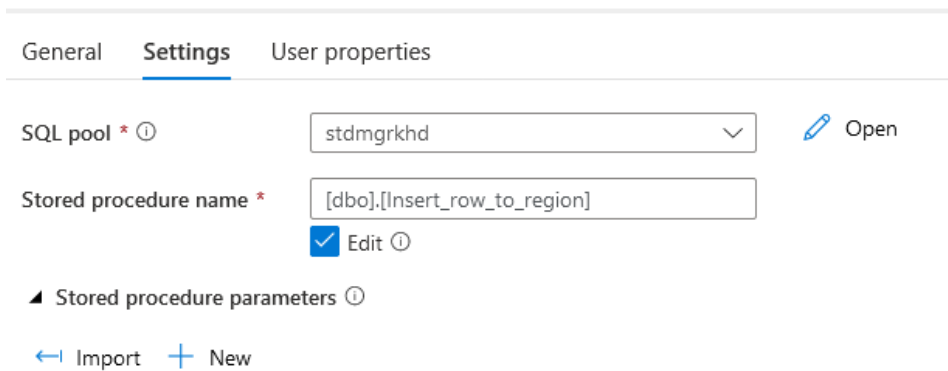
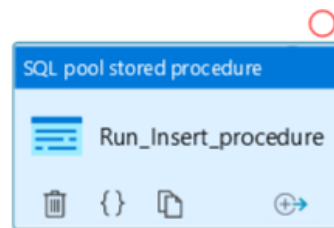
2. W wyszukiwarce „Activities” szukamy „SQL pool stored procedure” i dodajemy do pipeline’a. Tak samo szukamy „Web” i dodajemy do pipeline’a.



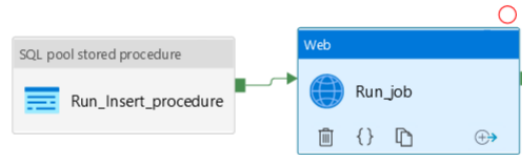




3. Wybierając wcześniej dodany SQL pool stored procedure w zakładce „Settings” określamy naszą składowaną procedurę w bazie danych, która ma symulować naszą hurtownię.



4. Następnie konfigurujemy aktywność „Web”, która będzie uruchomiona po składowanej procedurze. Również w zakładce „Settings” określamy URL (tutaj wklejamy nasz wcześniej przygotowany URL do webhook’a). Metode ustawiamy na POST i w ciele żądania dodajemy '{}’.



General	Settings	User properties
URL *	<input type="text" value="https://90d5bd54-e1d8-422a-9e25-837d5"/>	
Method *	<input type="text" value="POST"/>	
Headers	<a href="#">+ New</a>	
Body	<input type="text" value="{}"/>	
Datasets	<a href="#">+ Add dataset reference</a>	

W ten sposób można uruchomić przetwarzanie w bazie danych, po wykonaniu się procesów ETL.