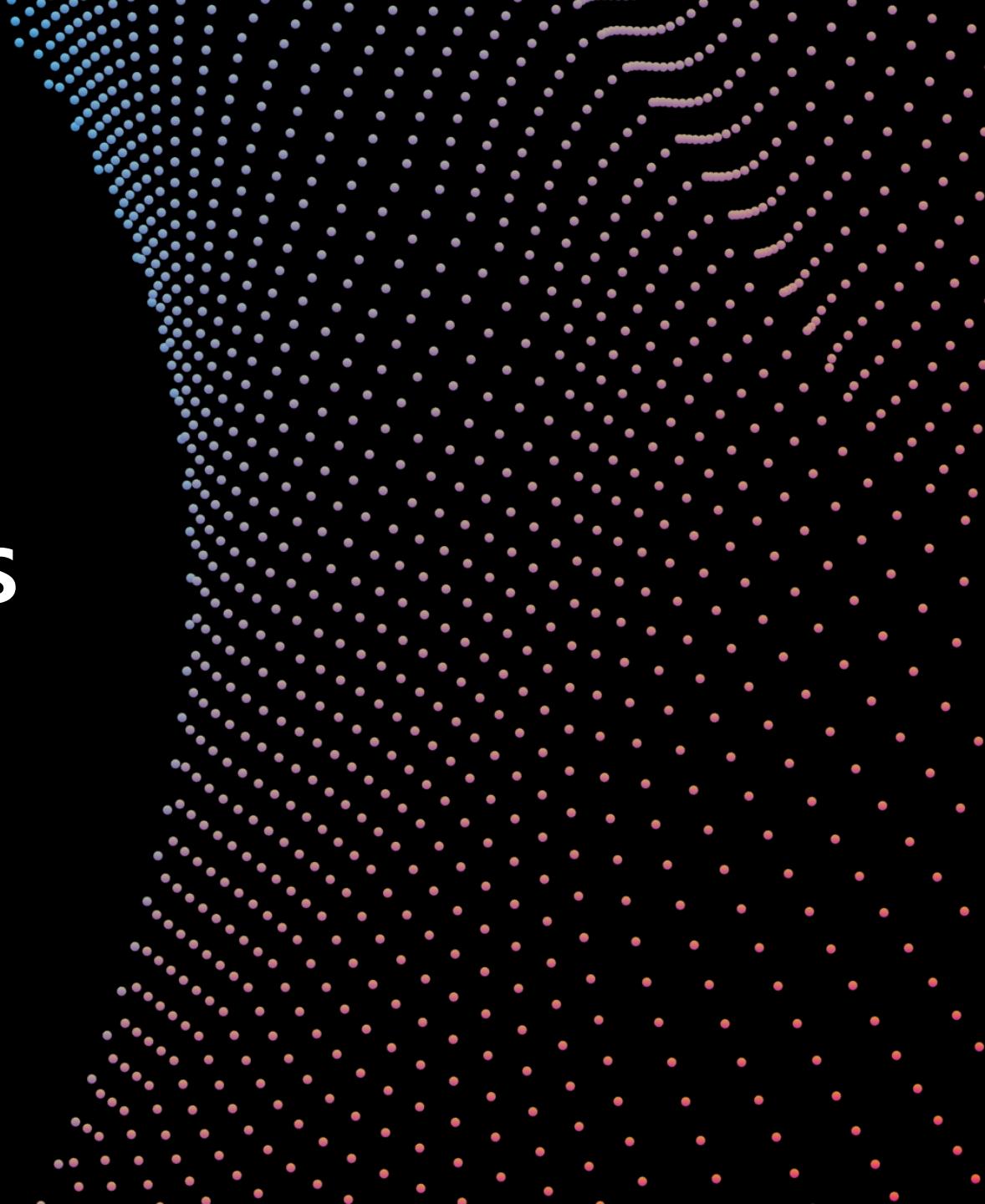


Azure Synapse Analytics

Technical Boot Camp

Day 3



A look into Day 3



Model	6:30-6:35	Welcome	Main Call
	6:35-7:30	Synapse Spark	
	7:30-8:30	Build Hands-on: Machine Learning	
	8:30-8:45	Break	Table Group Call
	8:45-9:30	Build Hands-on: Spark	
	9:30-10:15	Activity: CI/CD with Synapse Workspaces	
Monitor	10:15-11:15	Break	Main Call
	11:15-11:55	Monitor & Manage	
	11:55-12:15	Activity: Monitor & Manage	
	12:15-12:30	Break	
	12:30-1:15	Build Hands-on: Monitoring	Table Group Call
	1:15-1:30	Closing and Q&A	Main Call

Today we will be learning and collaborating across three spaces:

- Main call (this meeting)
- Table Group channel within the event Team
- CloudLabs Learner Portal & Synapse environment

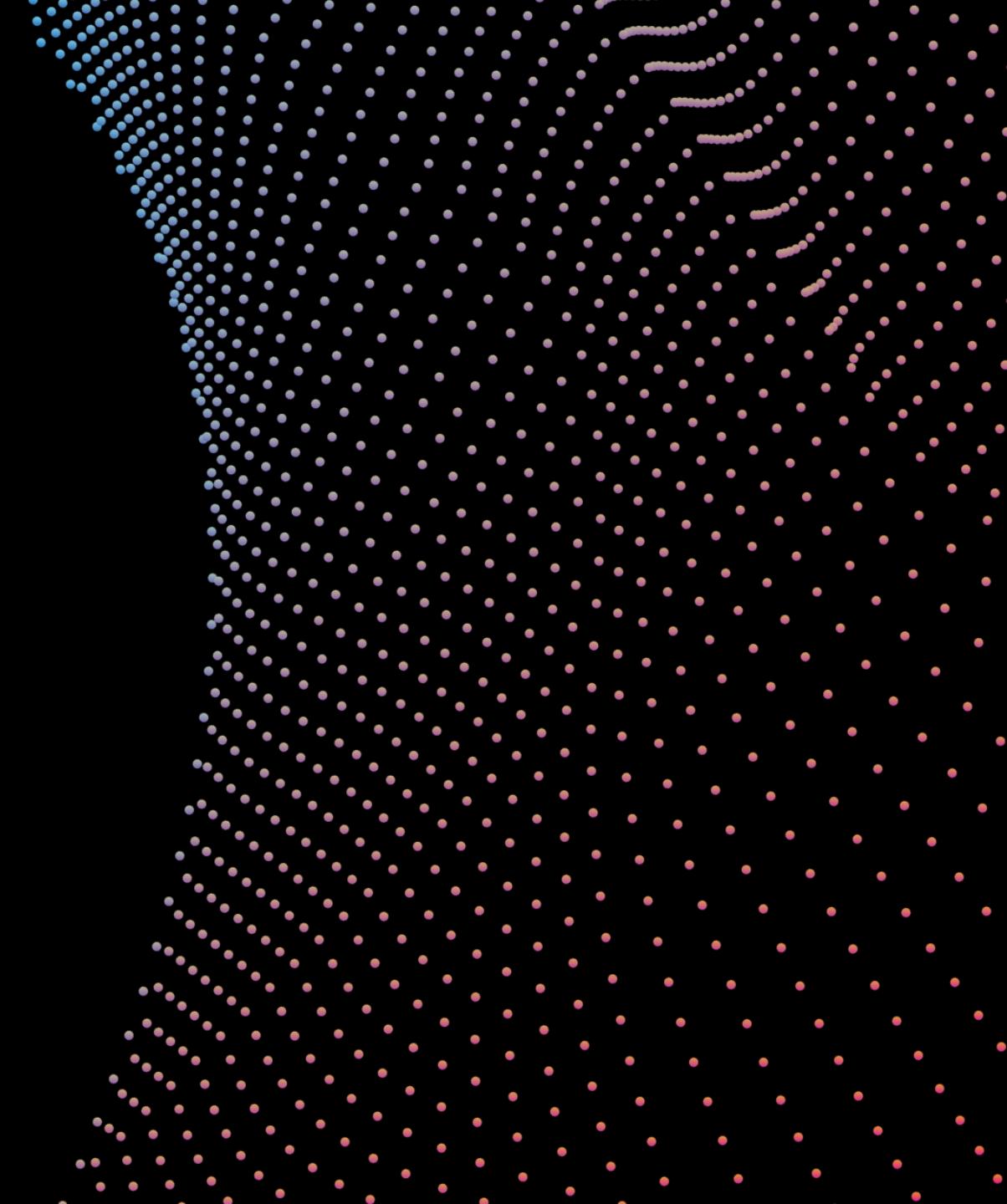
■ Presentation/
Whole Group

■ Lab

■ Activity/ Discussion/
Group Work

■ Announcements

Synapse Spark

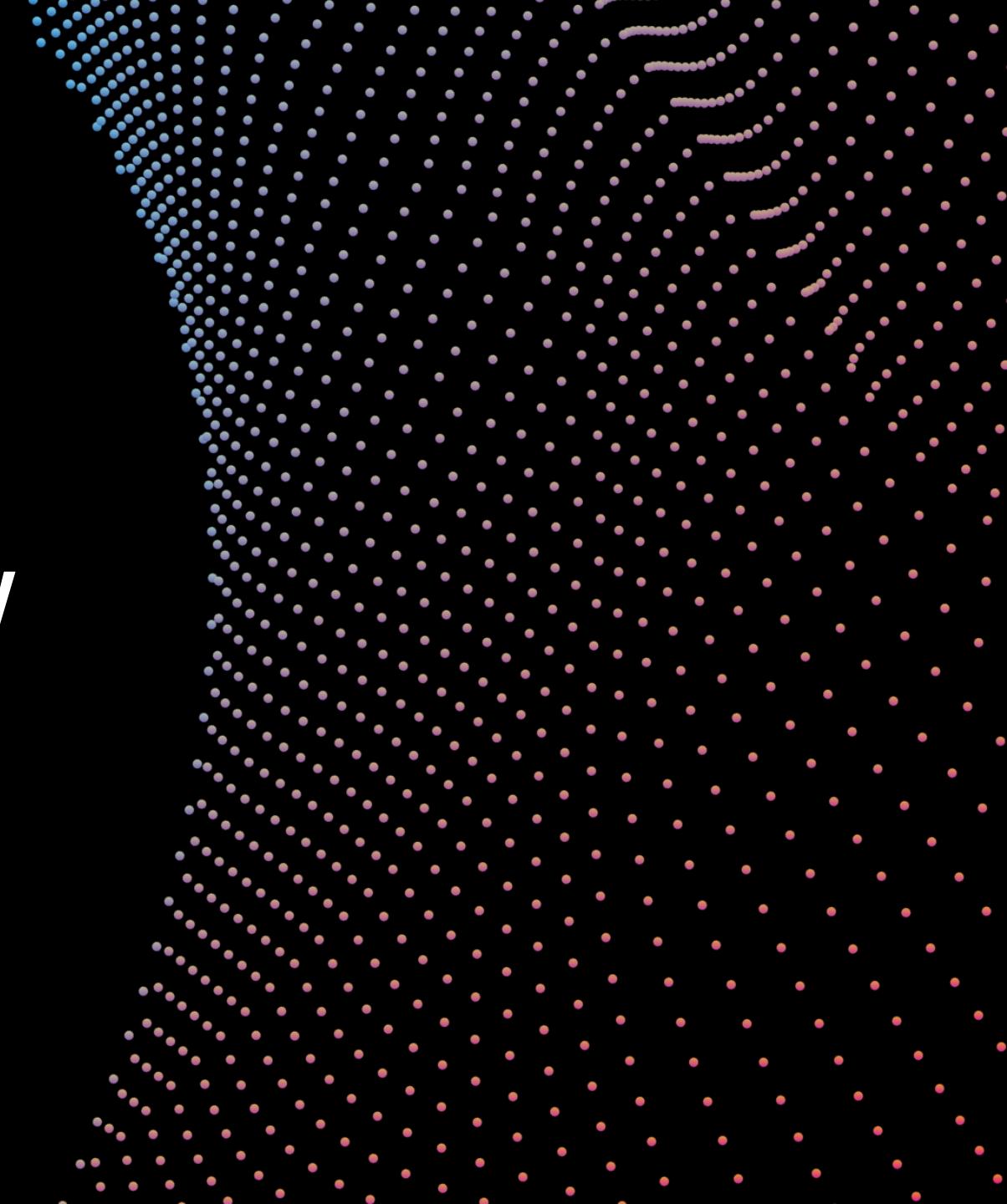


Agenda

- 1) **Apache Spark** - Overview of Apache Spark.
- 2) **Apache Spark in Synapse** - How Spark is utilized within Synapse.
- 3) **Notebooks** - Understand the notebook paradigm and how to use them in Synapse.
- 4) **Machine Learning** - How to train and deploy models with Synapse.



Apache Spark Overview



Question... ?

How have you seen Spark used in your customer engagements?

What use-cases does it excel in?

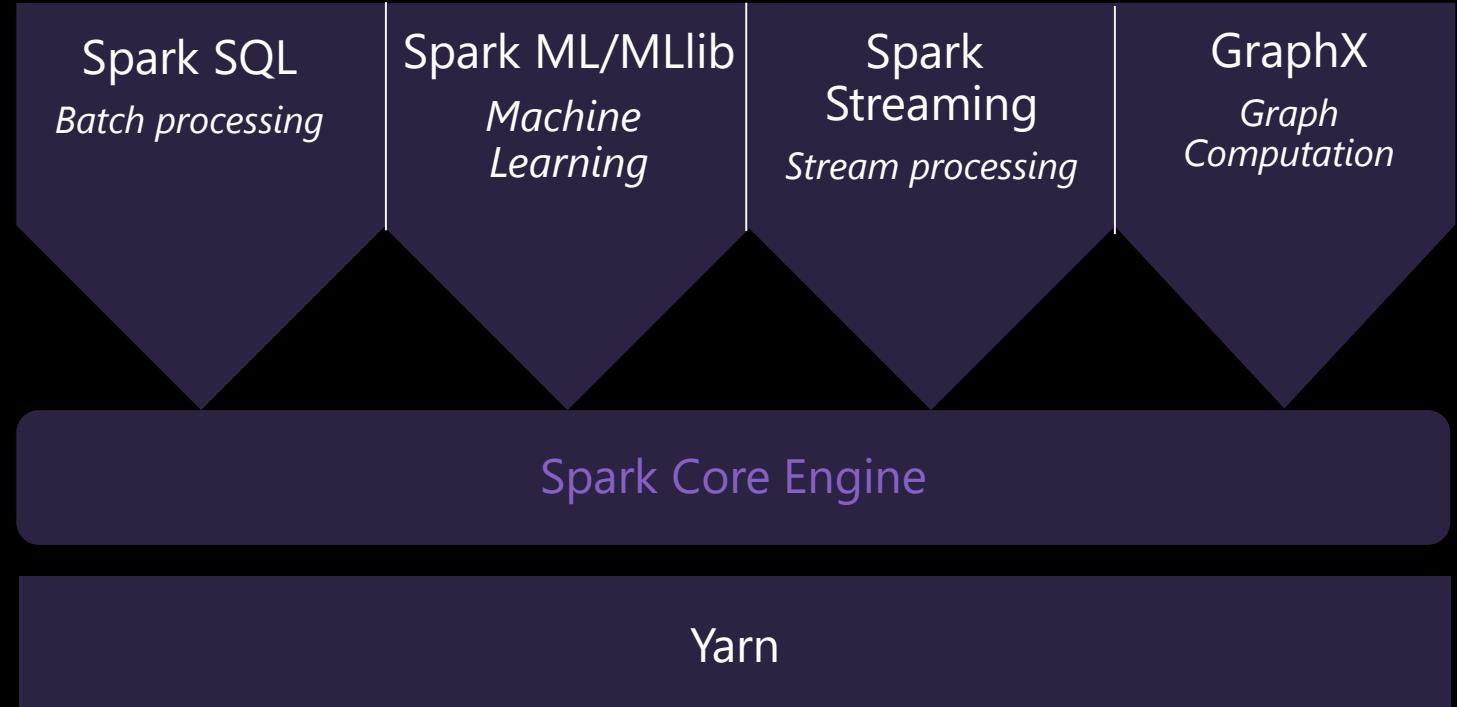


Apache Spark

A unified, open source, parallel, data processing framework for Big Data Analytics

Spark Unifies:

- Batch Processing
- Interactive SQL
- Real-time processing
- Machine Learning
- Deep Learning
- Graph Processing



<http://spark.apache.org>

Motivation for Apache Spark

Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of (**slow**) disk I/O

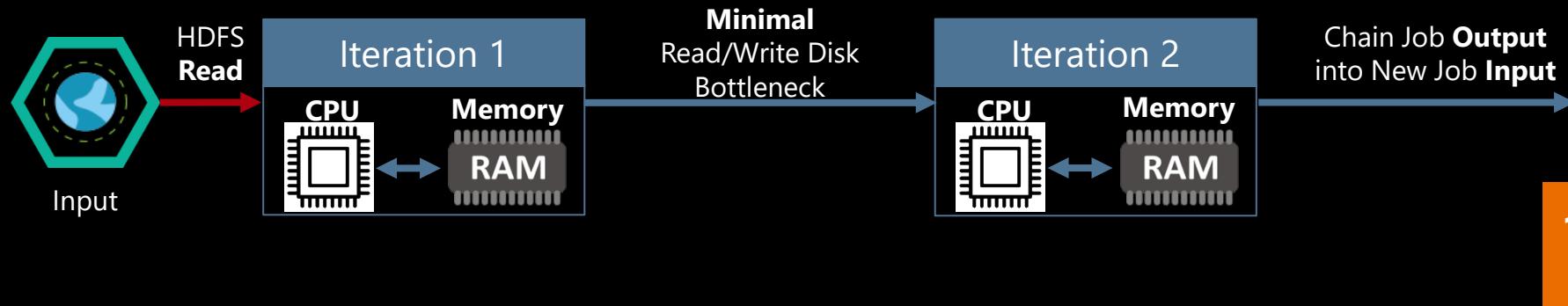


Motivation for Apache Spark

Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of **(slow) disk I/O**

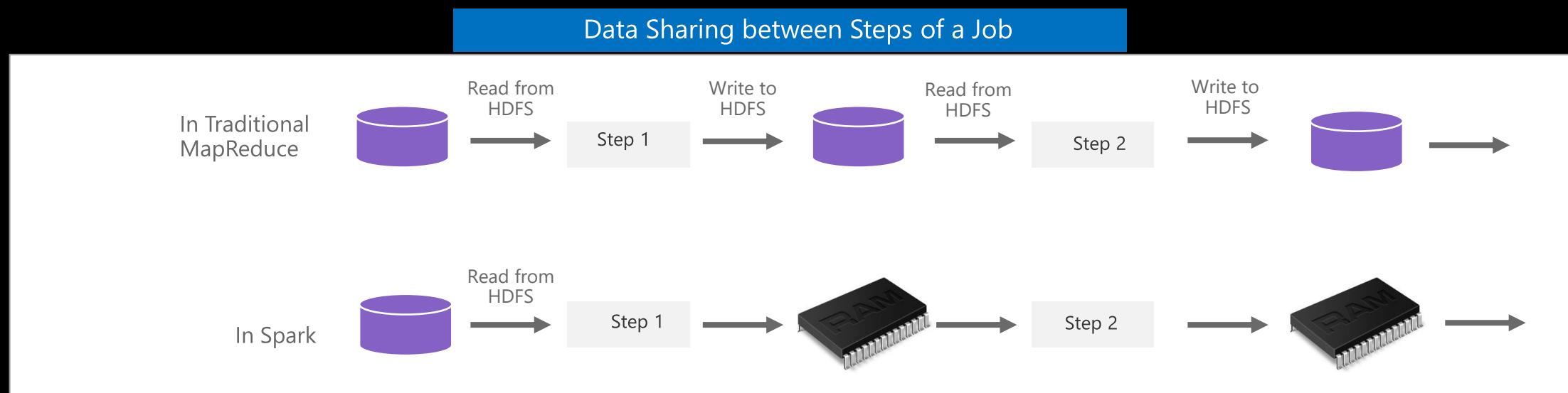


Solution: Keep data in-memory with a new distributed execution engine



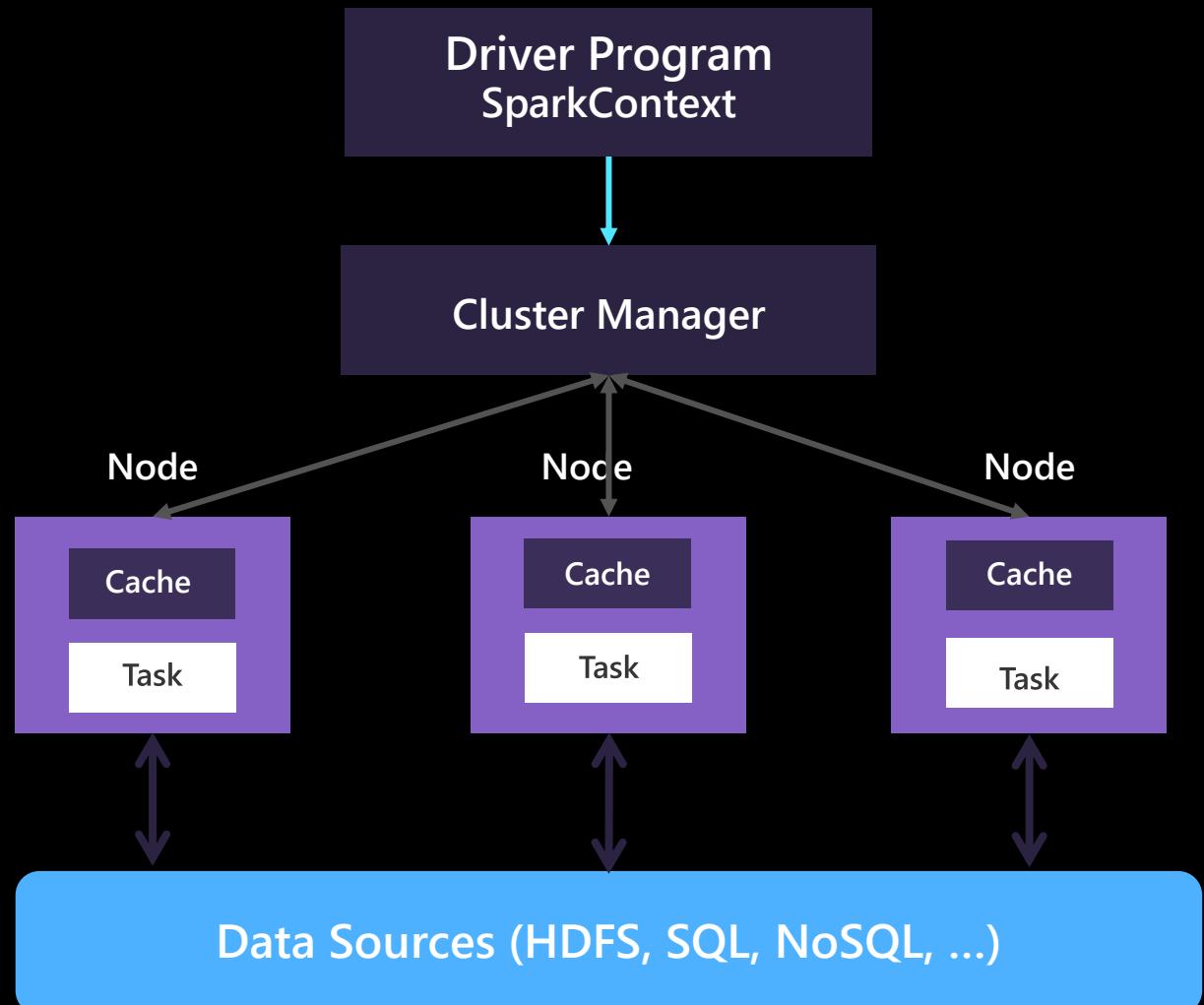
What makes Spark fast?

- **In-memory cluster computing:** Spark provides primitives for *in-memory* cluster computing. A Spark job can *load and cache* data into memory and query it repeatedly (iteratively) much quicker than disk-based systems.
- **Scala Integration:** Spark integrates into the Scala programming language, letting you manipulate distributed datasets like local collections. No need to structure everything as map and reduce operations
- **Faster Data-sharing:** Data-sharing between operations is faster as data is in-memory:
 - In (traditional) Hadoop data is shared through HDFS which is expensive. HDFS maintains three replicas.
 - Spark stores data in-memory *without any replication*.



General Spark Cluster Architecture

- 'Driver' runs the user's 'main' function and executes the various parallel operations on the worker nodes.
- The results of the operations are collected by the driver
- The worker nodes read and write data from/to Data Sources including HDFS.
- Worker node also cache transformed data in memory as RDDs (Resilient Distributed Datasets).
- Worker nodes and the Driver Node execute as VMs in public clouds (AWS, Google and Azure).



Apache Spark in Azure Synapse Analytics





Azure Synapse Apache Spark - Summary

Apache Spark 3.1 derivation

- Linux Foundation Delta Lake 1.0 support
- .Net Core 3.1 support
- Python 3.8 + Anaconda support
- Scala 2.12 support

Tightly coupled to other Azure Synapse services

- Integrated security and sign on
- Integrated Metadata
- Integrated and simplified provisioning
- Integrated UX including interact based notebooks
- Fast load of SQL Analytics pools

Core scenarios

- Data Prep/Data Engineering/ETL
- Machine Learning via Spark ML and Azure ML integration
- Extensible through library management

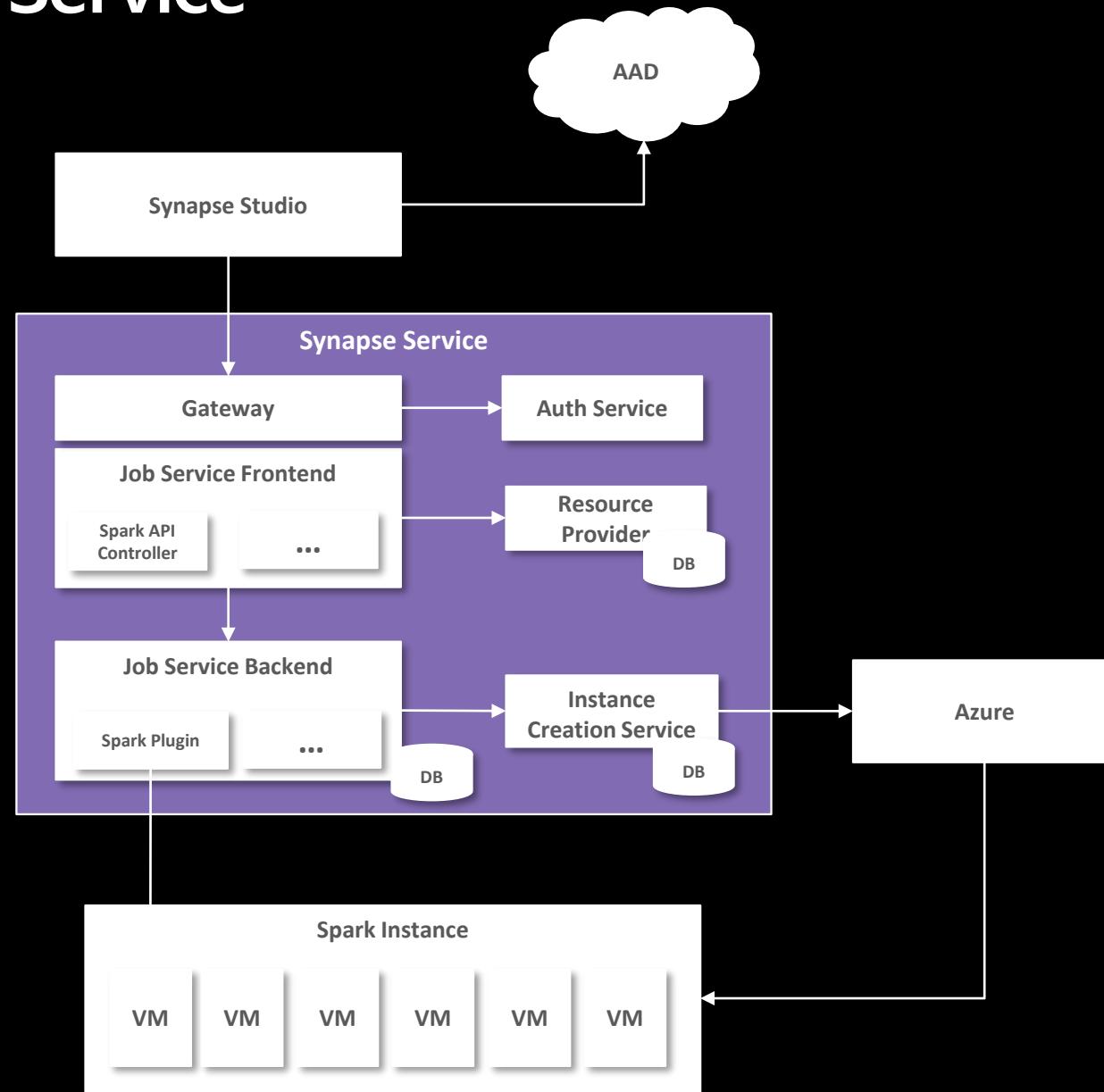
Efficient resource utilization

- Fast Start
- Auto scale (up and down)
- Auto pause
- Min cluster size of 3 nodes

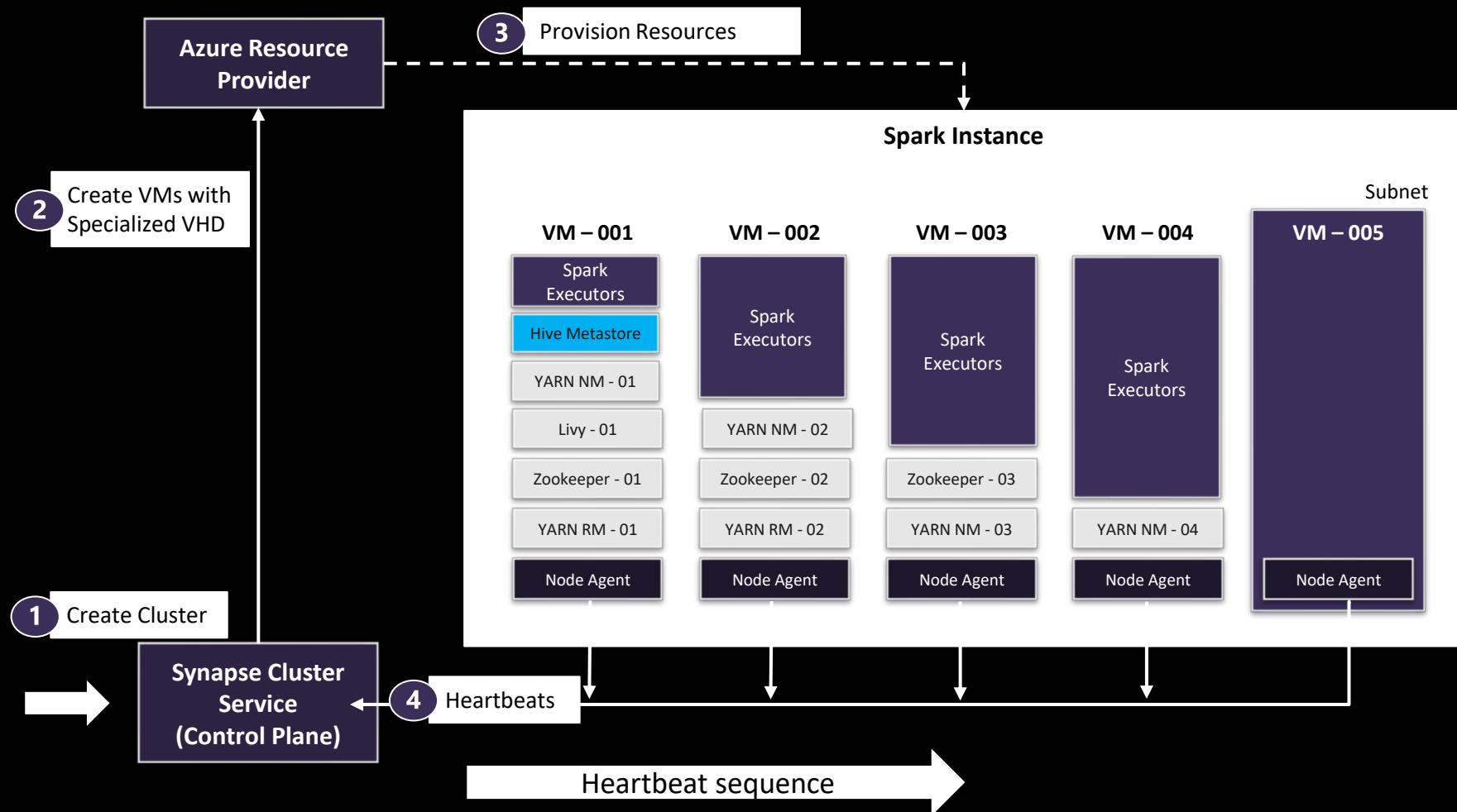
Multi Language Support

- .Net (C#), PySpark, Scala, Spark SQL, Java

Synapse Job Service



Spark Instance Creation



Creating a Spark pool (1 of 3)

Provision Spark Pool through Azure Portal with default settings or per requirements

Basic Settings – Minimum details required from user

Home > Synapse workspaces > euang-synapse-nov-ws - Apache Spark pools > Create Apache Spark pool

Create Apache Spark pool

Basics * Additional settings * Tags Summary

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 *

Enter Apache Spark pool name

Node size family

MemoryOptimized

Node size *

Medium (8 vCPU / 64 GB)

Autoscale * ⓘ

Enabled Disabled

Number of nodes *

3 40

Creating a Spark pool (2 of 3)

Autoscaling - Dynamic Allocation of Executors (optional)

- Allows a job to scale the number of executors within the min and max executor threshold
- More efficiently allocates resources reflective of the job being run
 - When requirements are vastly different across stages of a spark job
 - The volume of data being processed fluctuates with time
- Configuration via User Interface or code*

```
%%configure -f
{
  "conf": {
    "spark.dynamicAllocation.maxExecutors": 6,
    "spark.dynamicAllocation.enable": "true",
    "spark.dynamicAllocation.minExecutors": "2"
  }
}
```

* The defaults specified through code will override the values set through the user interface

Creating a Spark pool (3 of 3) - optional

Additional Settings offer optional settings to customize Spark pool

Customize component versions, auto-pause

Import libraries by providing text file containing library name and version

Home > Synapse workspaces > euang-synapse-nov-ws - Apache Spark pools > Create Apache Spark pool

Create Apache Spark pool

Basics * Additional settings * Tags Summary

Customize additional configuration parameters including autoscale and component versions.

Auto-pause

Enter required settings for this Apache Spark pool, including setting auto-pause and picking versions.

Auto-pause * ⓘ Enabled Disabled

Number of minutes idle * 15

Component versions

Select the Apache Spark version for your Apache Spark pool.

Apache Spark *	2.4
Python	3.6.1
Scala	2.11.12
Java	1.8.0_222
.NET Core	3.0
.NET for Apache Spark	0.6.0
Delta Lake	0.4.0

Packages

Upload environment configuration file ("PIP freeze" output).

File upload Select a file Upload

Library Management - Python

Overview

Customers can add new python libraries at Spark pool level

Benefits

Input requirements.txt in simple pip freeze format

Add new libraries to your cluster

Update versions of existing libraries on your cluster

Ability to specify different requirements file for different pools within the same workspace

Support for custom Wheel files when uploaded to the workspace default file system.

Constraints

The library version must exist on PyPI repository

Version downgrade of an existing library not allowed

Wheel files must be installed at a specific path:

abfss://<file_system>@<account_name>.dfs.core.windows.net/synapse/workspaces/<workspace_name>/sparkpools/<pool_name>/libraries/python/

Manage packages

SparkPool03

Apache Spark pools can be customized with additional libraries by assigning workspace packages or by providing a library requirements file. [Learn more](#)

Allow session level packages ⓘ

Enabled Disabled

Requirements files ⓘ

[Upload](#) [Refresh](#)

NAME	SIZE	DATE
No requirement files have been specified.		

Workspace packages ⓘ

[Select from workspace packages](#) [Refresh](#)

NAME	DATE
No workspaces packages have been selected.	

Python Library Requirements Format

Overview

The requirements text file has specific format requirements. These files are traditionally created in Python environments by running the `pip freeze` command but are also easy to create by hand.

File must be named **requirements.txt**.

Each line contains a requirement specifier.

libraryname ==1.3.0

This must contain at least the package name. You can also specify optional extras, an operator and version numbers according to the [PIP requirements file standard](#).

The basic operators used by version specifiers are:

An exact version (using ==).

A minimum version (using > or >=).

A maximum version (using < or <=).

The extras listed in square brackets ([]]) allow you to install additional components or variants of the library.

Spark Pools - Configuring

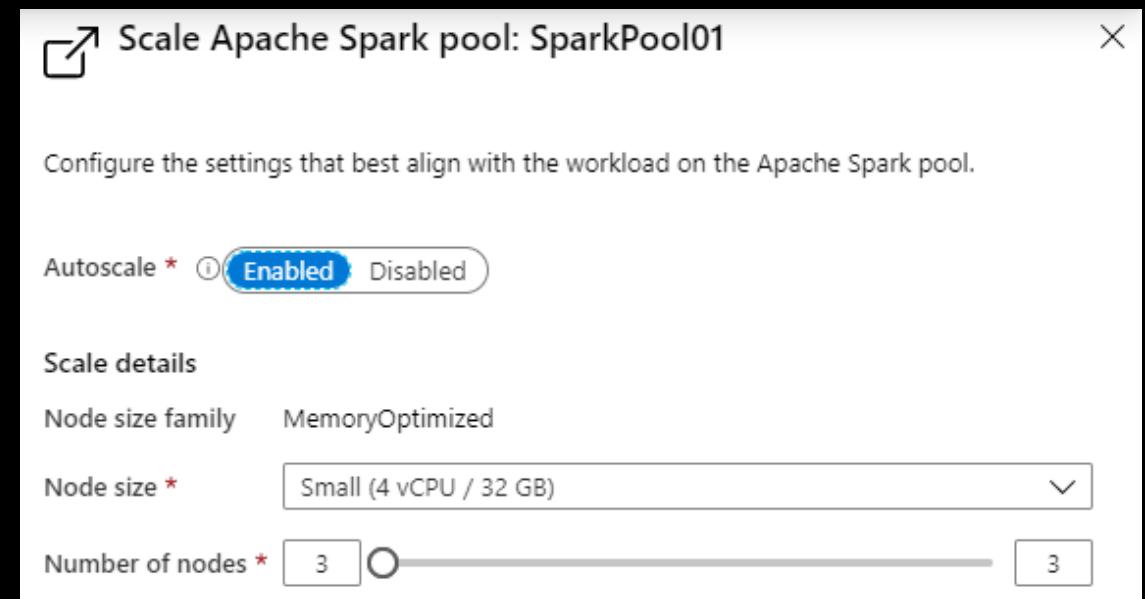
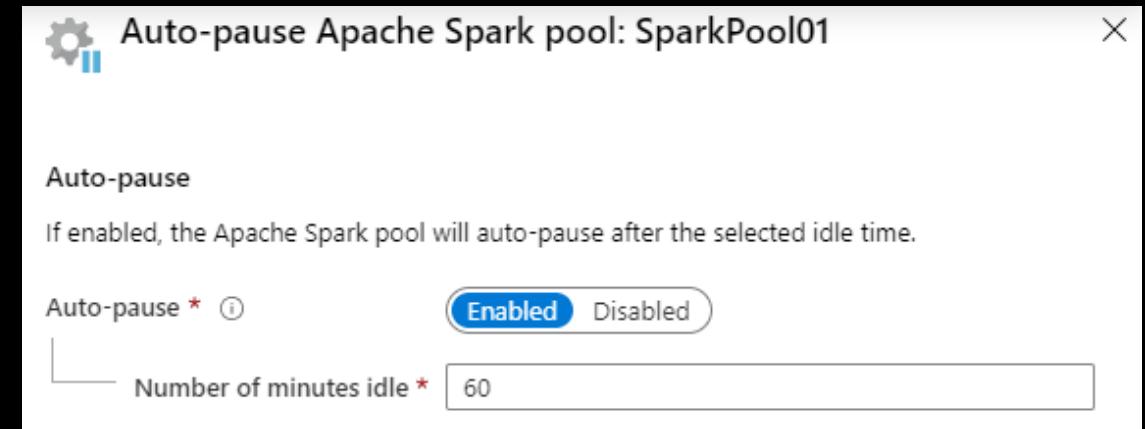
Auto-pause

Enables the pool to automatically stop running when no Spark applications (e.g., Notebooks) are running.

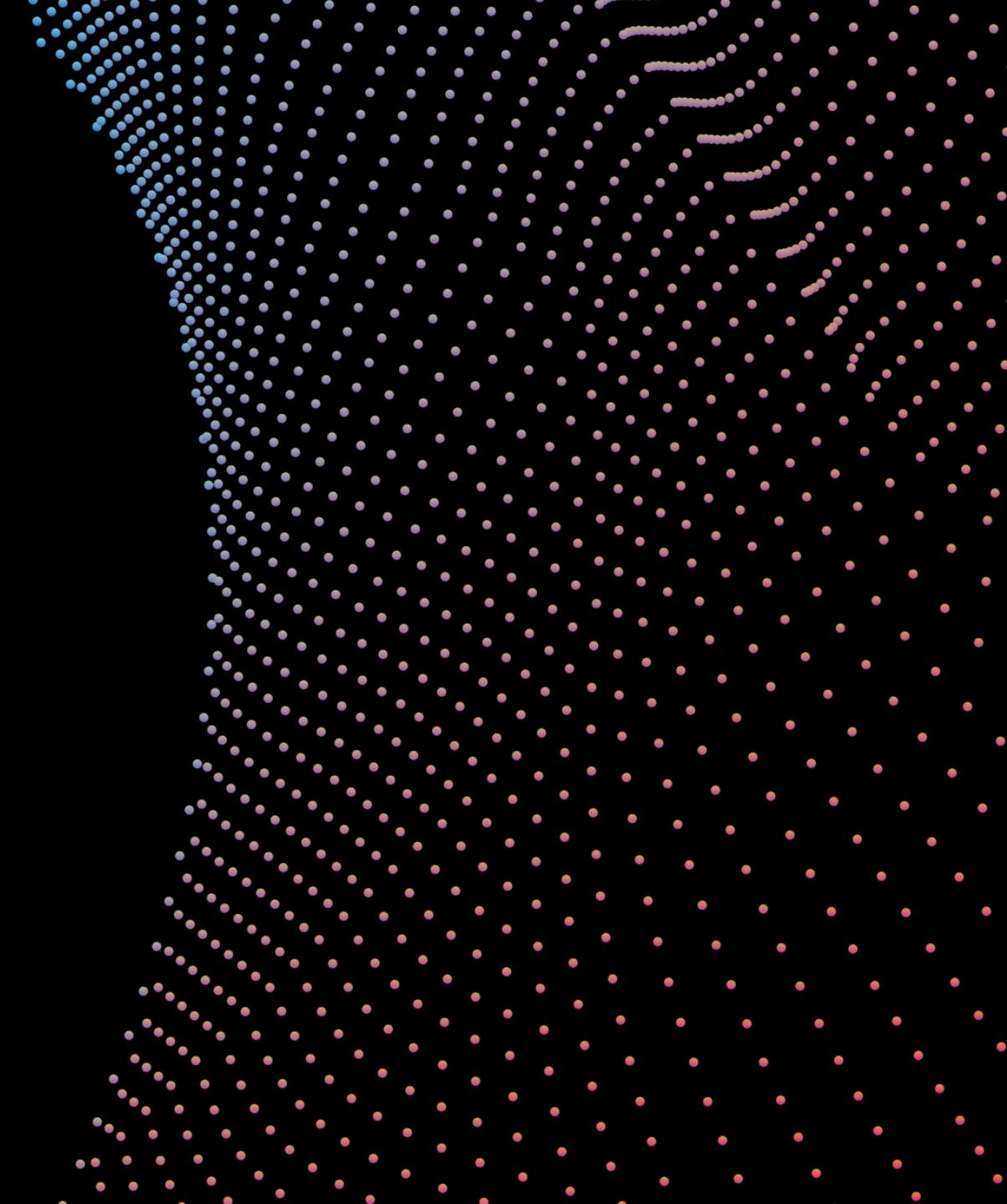
Autoscale

Enables the pool to automatically adjust the number of running nodes (VM's) supporting the Spark pool.

Note: Just because a pool is running, does not mean all the nodes allowed to run, according to the scale configuration, are actually running. They will be allocated on demand as new Notebooks are executed, up to the limit configured in autoscale.



Optimizing Spark Performance



Use the right data abstraction

DataFrames

- Best choice in most cases
- Catalyst optimization
- Whole-stage code generation
- Direct memory access
- Low GC overhead
- Less dev friendly than DataSets

DataSets

- Good in complex ETL pipelines
- Considerable performance hit in aggregations
- Catalyst optimization
- Dev friendly
- Serialization/deserialization overhead
- High GC overhead
- Breaks whole-stage code generation

RDDs

- Except for very special cases (new custom RDD), they are not needed
- No Catalyst optimization
- No whole-stage code generation
- High GC overhead
- Require Spark 1.x legacy APIs

Data formats, cache, and memory

Data formats

- Standard formats (CSV, JSON, XML, PARQUET, ORC, AVRO)
- Extensible via external data sources
- Best format is PARQUET with SNAPPY compression
- SNAPPY can be larger than GZIP but decompresses faster

Cache

- Native mechanisms (.persist(), .cache(), CACHE TABLE)
- Effective with small data sets and ETL pipelines to cache intermediate results
- Native caching does not work well with partitioning

Memory

- Prefer smaller partitions
- With partitioning, keep in mind data size, types, and distribution
- Consider Kryo data serialization instead of Java
- Monitor and tune Spark configuration settings

Other considerations

- **Optimize data serialization (Kryo vs. Java)**
- **Use bucketing – works well for partitioning on large numbers of values (e.g., product ids)**
 - Buckets are determined by hashing the bucket key of each row
 - Bucketed tables store metadata about how they are bucketed and sorted
- **Optimize joins and shuffles**
 - Look for data skews
 - Check join types (SortMerge is default, but Broadcast might work better for smaller data sets or asymmetric joins)
 - Merge join is available for bucketed tables
 - The order of joins matters
- **Select the correct executor size**
- **Optimize job execution**
 - Cache as necessary
 - Broadcast variables to executors
 - Use the thread pool on the driver

Notebooks in Azure Synapse Analytics

Creating a Notebook

Notebook creating

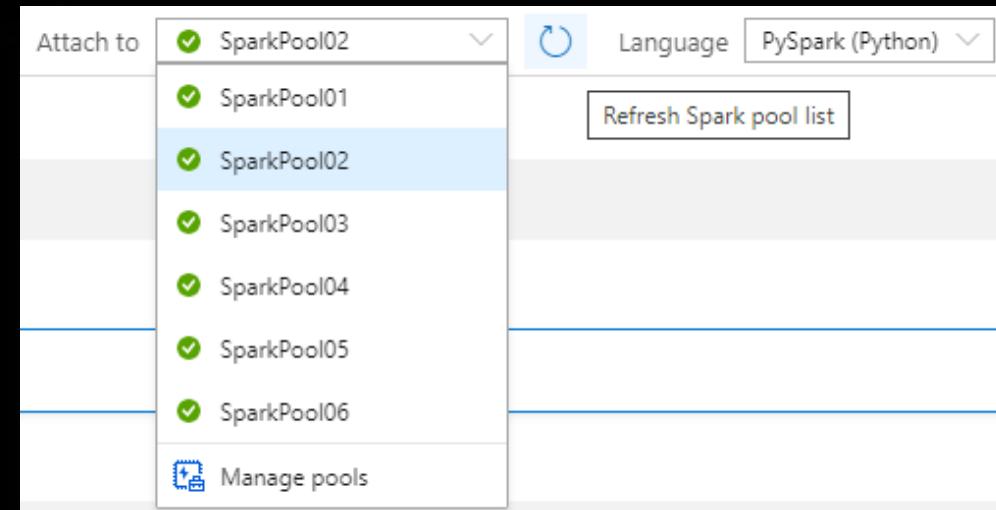
Create a new notebook within the Develop hub of Synapse Studio.

The screenshot shows the Microsoft Synapse Studio interface. At the top, there's a navigation bar with icons for back, forward, and search. Below it is a sidebar titled "Develop" with a search bar and a list of resource types: SQL script (selected), Notebook, Data flow, Spark job definition, Power BI report, and Import. To the right of the sidebar is a main area with a toolbar containing icons for file operations. A red arrow points from the "Notebook" icon in the toolbar down to a properties dialog box. This dialog has tabs for "General" and "Advanced". In the "General" tab, there's a note: "Choose a name for your Notebook. This name can be updated at any time until it is published." Below this is a "Name" field containing the value "Exercise 1 - Read with Spark_Copy1".

Notebook setup

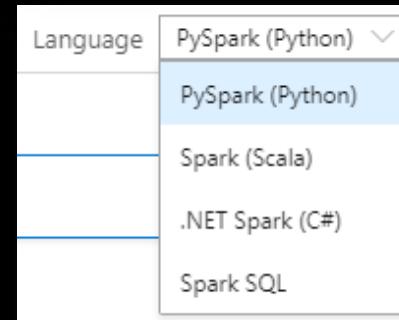
Selecting the Spark pool to attach to

In the notebook toolbar, select the Spark pool to attach to before running any cells in the notebook.



Notebook default language

The default language used for all cells in the notebook is selected in the Language drop down.



You can override this for any particular cell to use a different language by using the appropriate cell magic (covered later).

Notebook setup – Session configuration

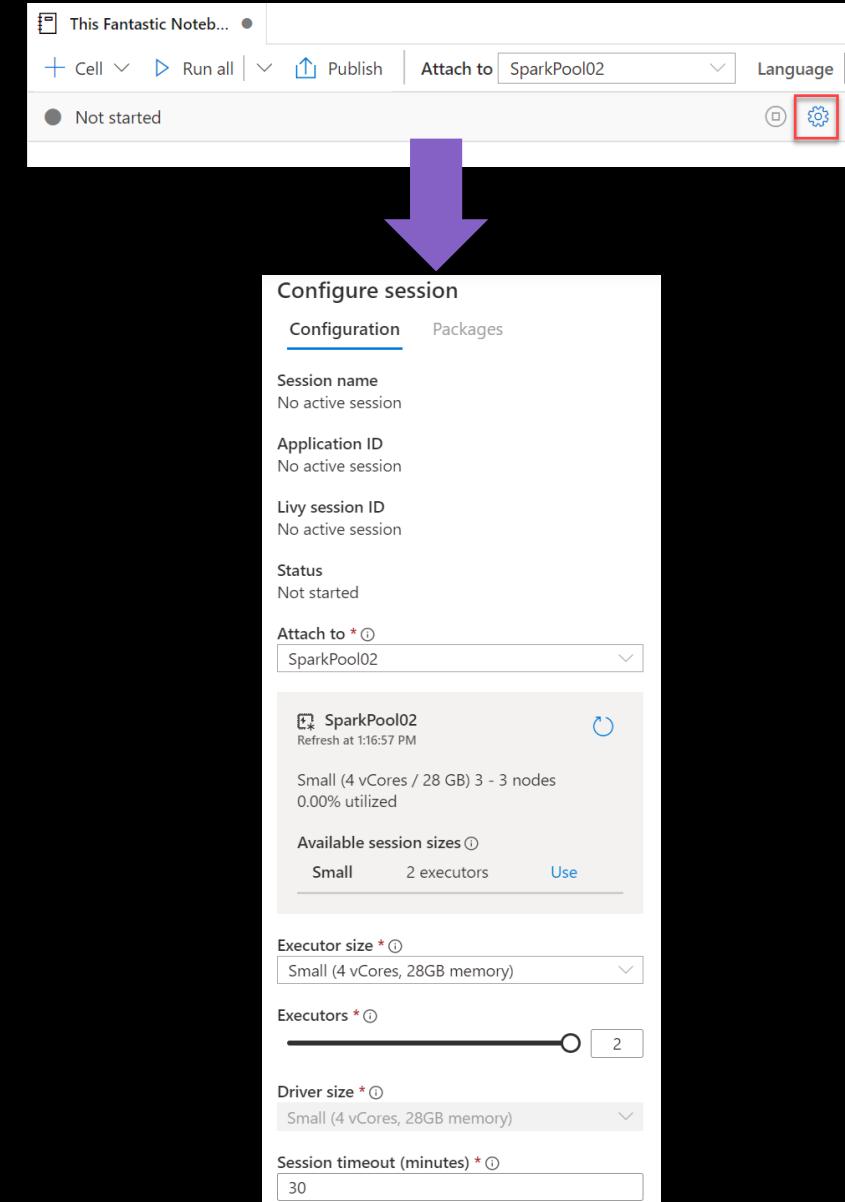
A Spark session is created upon first run of any cell in a notebook. The configuration for a Spark session can only be set *before* it is created.

The Spark session allocates one Spark application.

This Spark application consumes a fixed set of resources, as defined in the session configuration. These resources are reserved from the total resources available in the Spark pool upon session start.

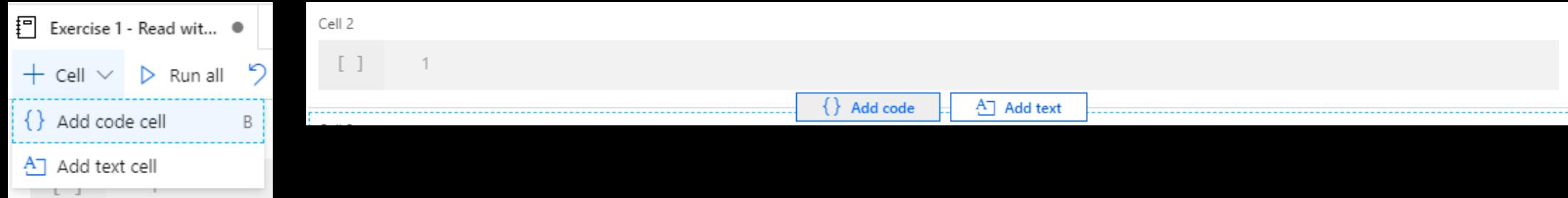
Each session reserves resources for at least one driver process and two executor processes. You can configure the size of executors, as well as the number of executors. The size of the driver is always the same as the executors.

A session ends when the notebook is closed or when the session timeout is reached without any activity in the notebook. You can configure the session timeout.

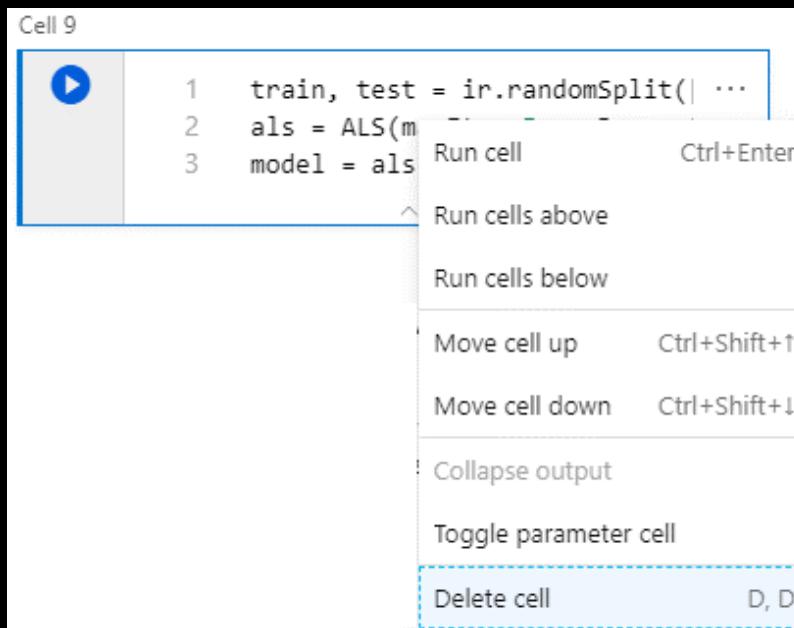


Notebook capabilities – Cell management

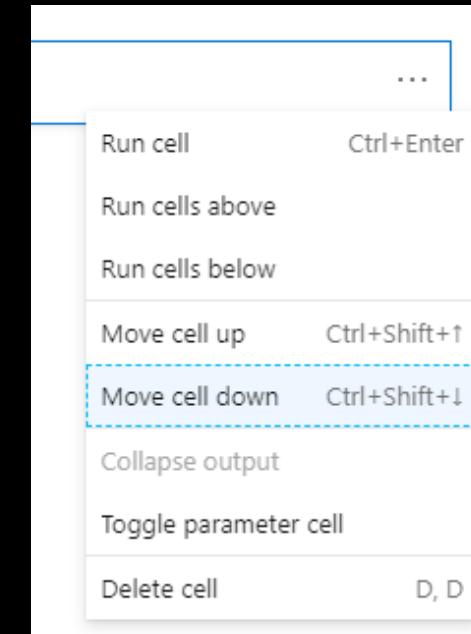
Add code or text cells



Remove code or text cells



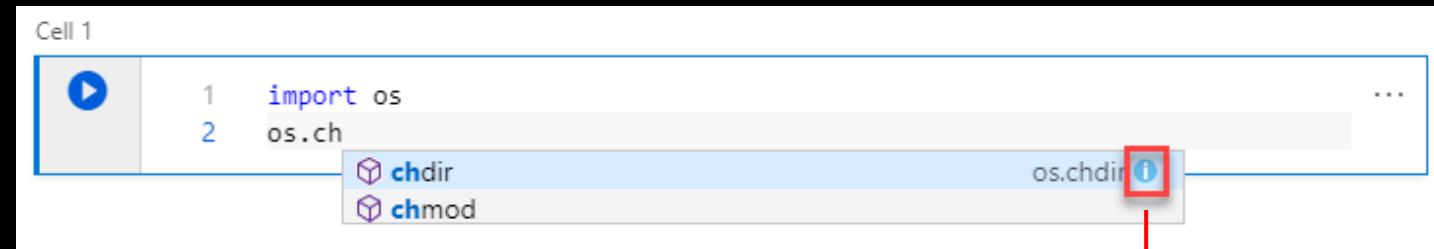
Adjust cell order



Notebook capabilities – Editing code

Statement completion

Use Ctrl + Space to display

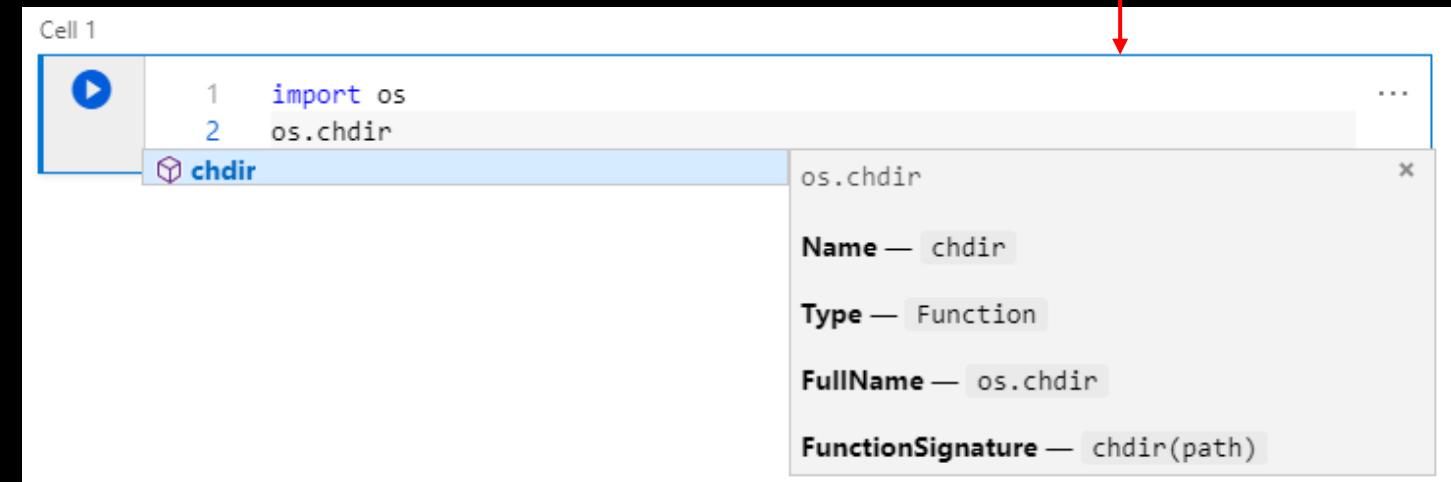


```
Cell 1
1 import os
2 os.ch
```

os.chdir
os.chmod

Inline documentation

Select the information tooltip to display



```
Cell 1
1 import os
2 os.chdir
```

os.chdir

Name — chdir
Type — Function
FullName — os.chdir
FunctionSignature — chdir(path)

Notebook capabilities – Editing code

Cell magics

Use cell magic to change the language used when interpreting the cell.

These must be the first line in the cell.

Magic	Description
%%pyspark	Execute Python code against Spark Context.
%%spark	Execute Scala code against Spark Context.
%%sql	Execute Spark SQL query against Spark Context.
%%csharp	Execute Spark.NET C# code against Spark Context.

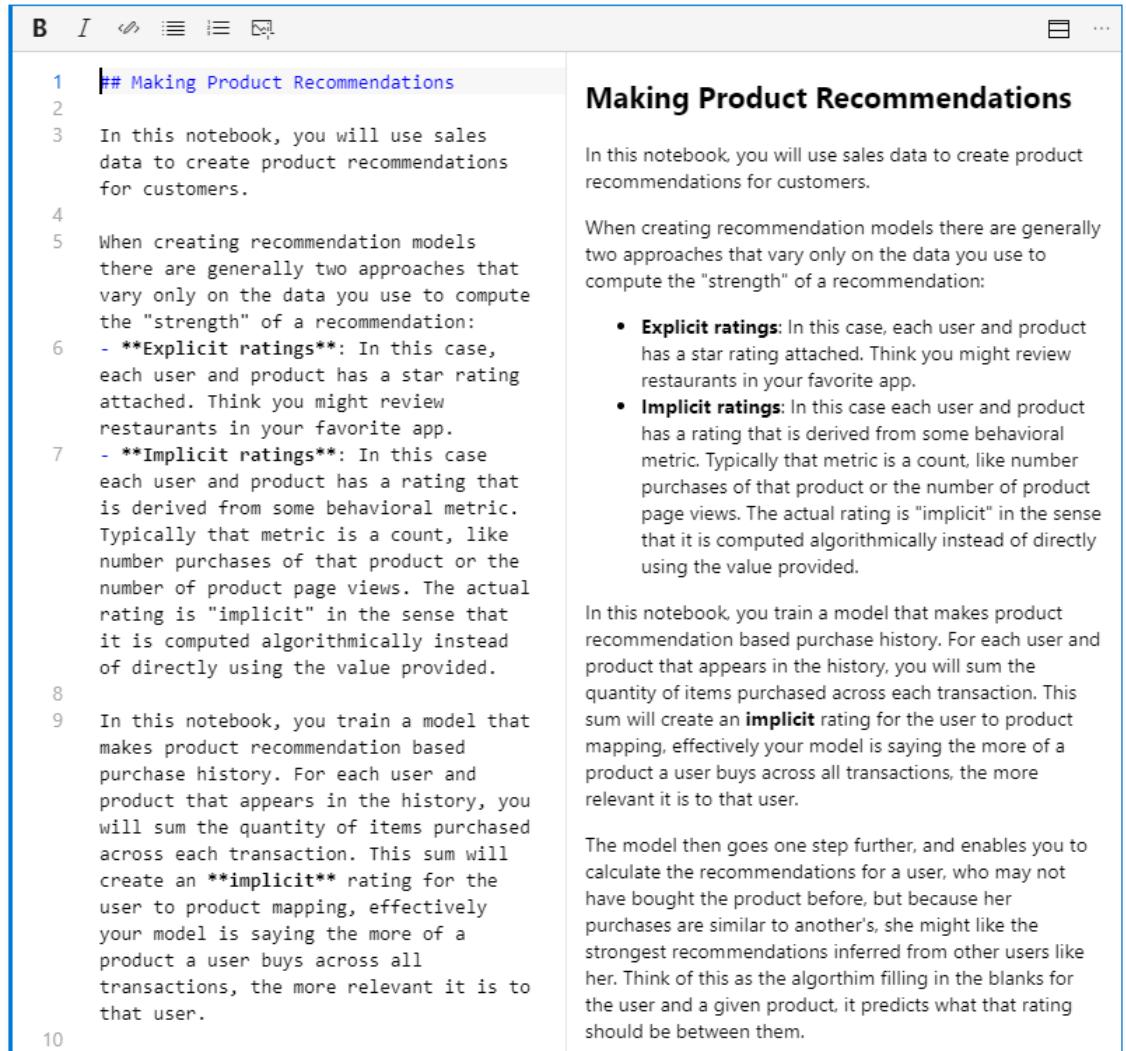
Cell 1

```
1 %%pyspark
2 data_path = spark.read.load([
3     'abfss://staging@asadatalake01.dfs.core.windows.net',
4     'abfss://staging@asadatalake01.dfs.core.windows.net',
5     'abfss://staging@asadatalake01.dfs.core.windows.net'],
6     format='parquet')
7 data_path.show(100)
```

Notebook capabilities – Editing text cells

Support for GitHub flavored markdown

Notebooks support GitHub flavored markdown, enabling limited text formatting, links, lists, code blocks and images.



The screenshot shows a Jupyter Notebook cell with the following content:

```
B I ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋
```

1 ## Making Product Recommendations
2
3 In this notebook, you will use sales data to create product recommendations for customers.
4
5 When creating recommendation models there are generally two approaches that vary only on the data you use to compute the "strength" of a recommendation:
6 - **Explicit ratings**: In this case, each user and product has a star rating attached. Think you might review restaurants in your favorite app.
7 - **Implicit ratings**: In this case each user and product has a rating that is derived from some behavioral metric. Typically that metric is a count, like number purchases of that product or the number of product page views. The actual rating is "implicit" in the sense that it is computed algorithmically instead of directly using the value provided.
8
9 In this notebook, you train a model that makes product recommendation based purchase history. For each user and product that appears in the history, you will sum the quantity of items purchased across each transaction. This sum will create an **implicit** rating for the user to product mapping, effectively your model is saying the more of a product a user buys across all transactions, the more relevant it is to that user.

10

Making Product Recommendations

In this notebook, you will use sales data to create product recommendations for customers.

When creating recommendation models there are generally two approaches that vary only on the data you use to compute the "strength" of a recommendation:

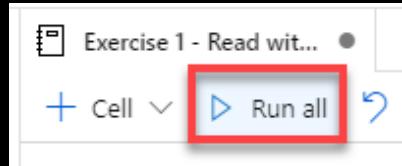
- **Explicit ratings:** In this case, each user and product has a star rating attached. Think you might review restaurants in your favorite app.
- **Implicit ratings:** In this case each user and product has a rating that is derived from some behavioral metric. Typically that metric is a count, like number purchases of that product or the number of product page views. The actual rating is "implicit" in the sense that it is computed algorithmically instead of directly using the value provided.

In this notebook, you train a model that makes product recommendation based purchase history. For each user and product that appears in the history, you will sum the quantity of items purchased across each transaction. This sum will create an **implicit** rating for the user to product mapping, effectively your model is saying the more of a product a user buys across all transactions, the more relevant it is to that user.

The model then goes one step further, and enables you to calculate the recommendations for a user, who may not have bought the product before, but because her purchases are similar to another's, she might like the strongest recommendations inferred from other users like her. Think of this as the algorithm filling in the blanks for the user and a given product, it predicts what that rating should be between them.

Notebook capabilities – Running cells

Run all cells from toolbar



Run current cell from toolbar from cell itself



Notebook capabilities – Cell output

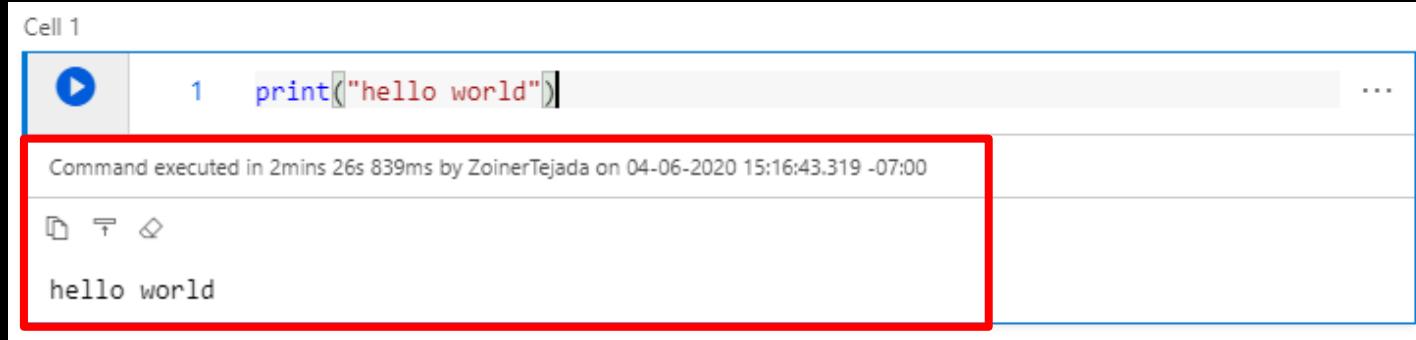
Cell output

Cell 1

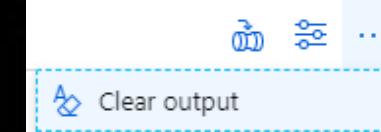
```
1 print("hello world")
```

Command executed in 2mins 26s 839ms by ZoinerTejada on 04-06-2020 15:16:43.319 -07:00

hello world



Clear all cells output from toolbar



Spark Pools – Notebook capabilities

View DataFrame content as table or chart

Use the `display` function and pass in your dataframe.

Cell 1

```
1 display(data_frame)
```

View **Table** **Chart**

SaleKey	CityKey
2003570	76101
2003571	76101
2003572	76101
2003573	106153
2003574	106153
2003575	106153
2003576	106153
2003577	106153
2003578	81839
2003579	81839
2003580	81839
2003581	81839

Filter by keyword
Please enter text here

Column
All

Apply **Reset**

Cell 1

```
1 display(data_frame)
```

View **Table** **Chart**

Chart type
bar chart

X axis column
CustomerKey

Y axis columns
CityKey

Aggregation
COUNT

Y axis label
CityKey

X axis label
CustomerKey

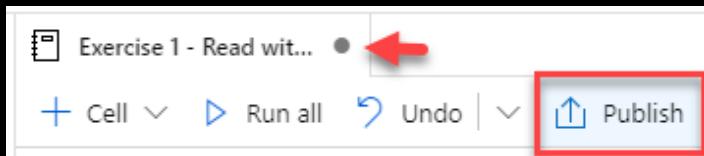
Stacked

Apply **Cancel**

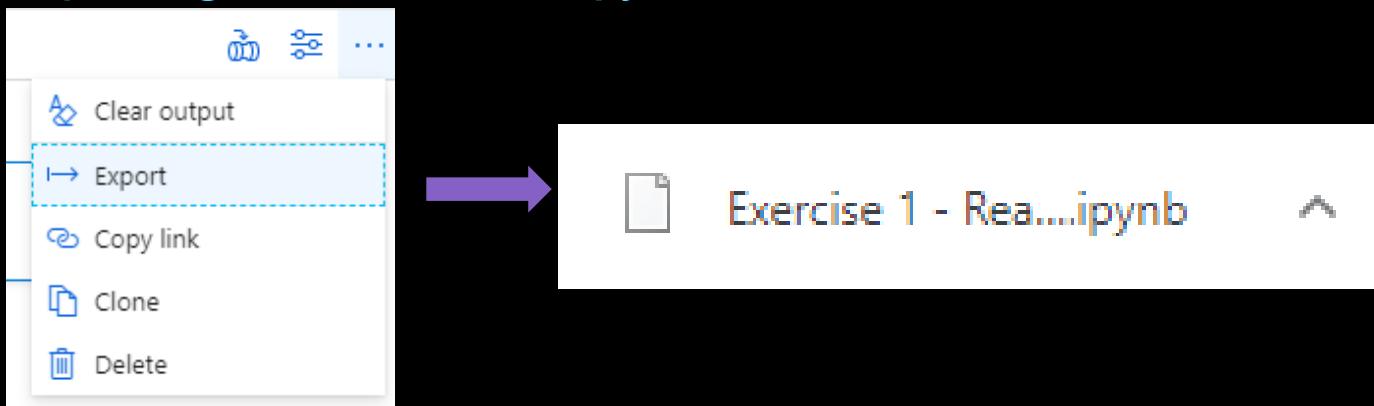
CustomerKey	CityKey
166	4
0	28
91	2
315	2
38	4
121	3
118	3
270	3

Spark Pools – Notebook capabilities

Saving (publishing)

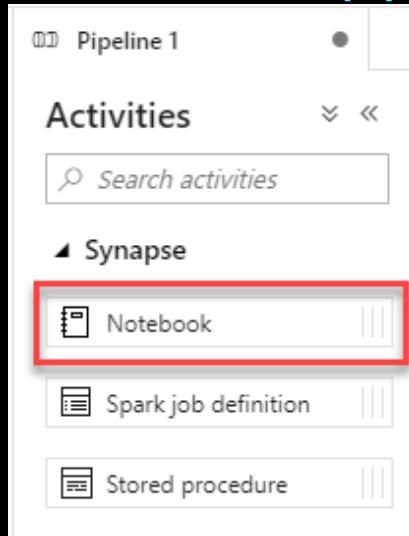


Exporting (download the .ipynb)

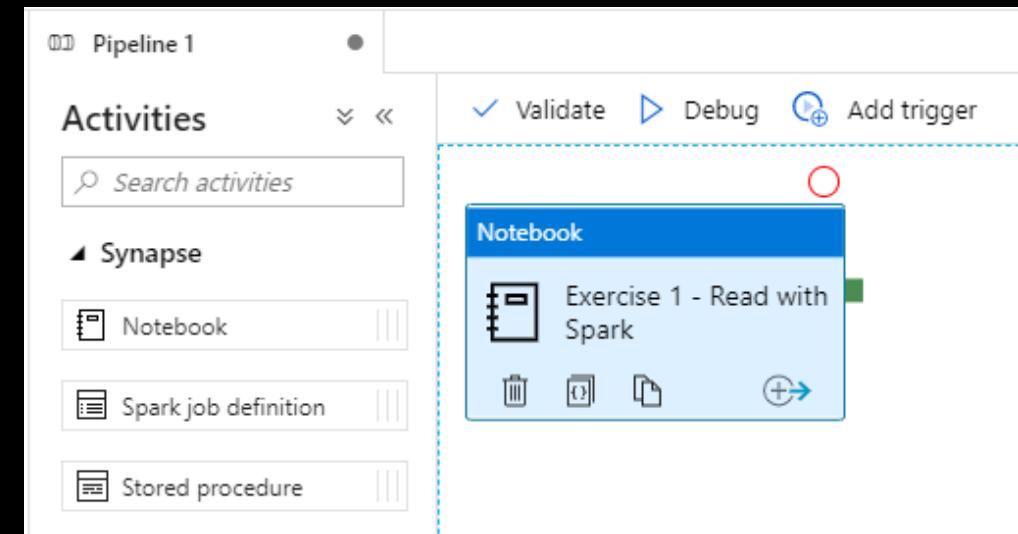
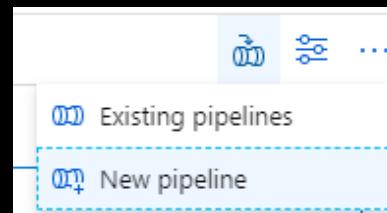


Notebook capabilities – Use in pipelines

Notebooks are pipeline activities

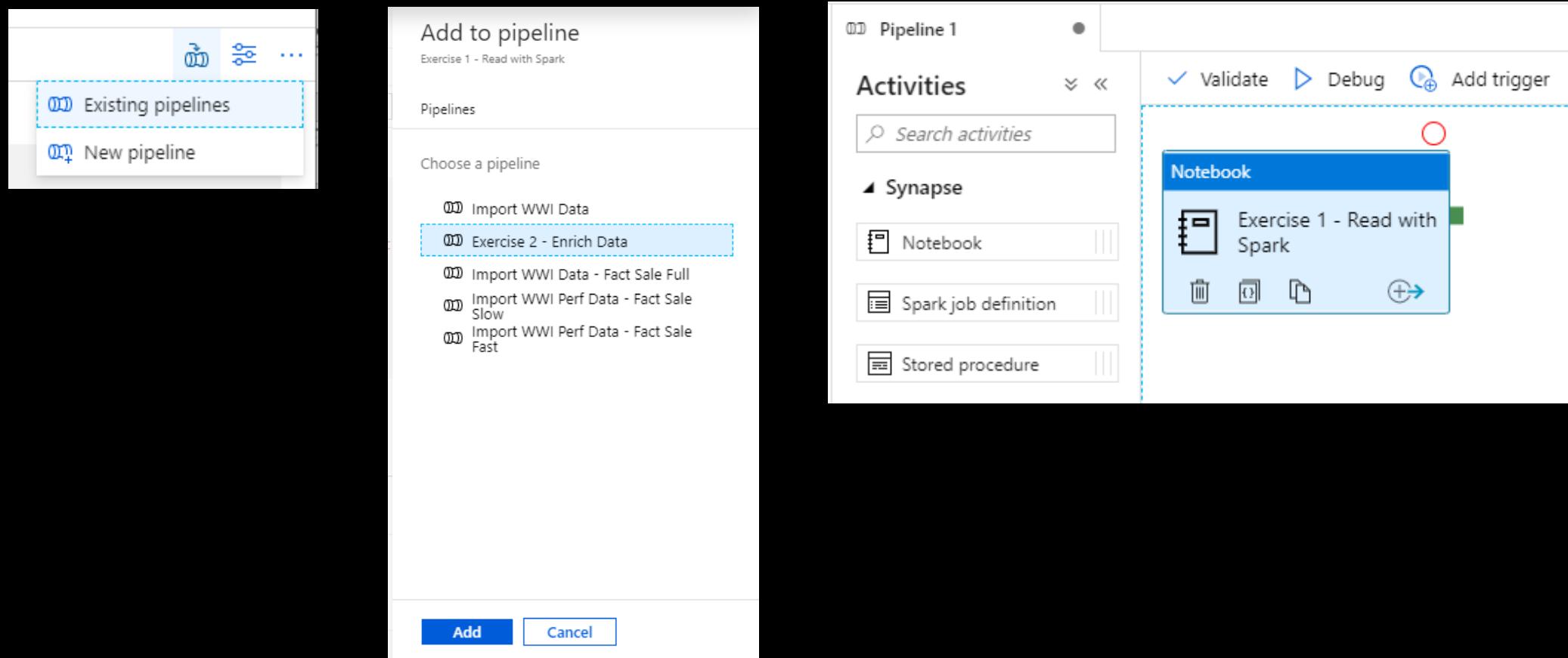


Execute a notebook as a step in a new pipeline



Notebook capabilities – Use in pipelines (2)

Execute a notebook as a step in an existing pipeline



Notebook capabilities – Querying SQL Pools

Loading a SQL Pool table to a DataFrame

Data + ⌂ <>

- ▶ Storage accounts 1
- ◀ Databases 3
- ◀ SQLPool01 (SQL pool)
 - ◀ Tables
 - ▶ wwi_staging.Sale_B23
 - ▶ wwi_staging.Sale_B24
 - ▶ wwi_staging.Sale_B25
 - ▶ wwi.DimCity
 - ▶ wwi.DimCustomer
 - ▶ wwi.DimDate
 - ▶ wwi.DimEmployee
 - New SQL script >
 - New notebook >
 - Load to DataFrame
- Refresh

Notebook 1

+ Cell ▶ Run all Undo Publish Attach to Select Spark pool

Cell 1

```
[ ] 1 val df = spark.read.sqldialect("SQLPool01.wwi.DimCity")
2 // df.show(10)
```

Notebook capabilities – Querying Parquet Files

Loading Parquet files to a DataFrame

The screenshot illustrates the process of loading Parquet files from a storage account into a DataFrame using a notebook in the Azure Data Explorer interface.

Left Panel (Data Explorer): Shows the storage account structure under the "Data" tab. The "staging" folder is selected. Inside "staging", there are three Parquet files: "enrichedcustomer.parquet", "enrichedcustomer_18fc8cab.parquet", and "enrichedcustomer_a6eb5555.parquet".

Middle Panel (File Explorer): Shows the contents of the "staging" folder. A context menu is open over the "enrichedcustomer.parquet" file, with the "New notebook" option highlighted by a red arrow.

Bottom Panel (Notebook View): Shows a notebook titled "Notebook 1" with one cell named "Cell 1". The cell contains the following PySpark code:

```
%%pyspark
data_path = spark.read.load([
    'abfss://staging@asadatalake01.dfs.core.windows.net/customer/enrichedcustomer.parquet',
    'abfss://staging@asadatalake01.dfs.core.windows.net/customer/enrichedcustomer_18fc8ca',
    'abfss://staging@asadatalake01.dfs.core.windows.net/customer/enrichedcustomer_a6eb555'],
    format='parquet')
data_path.show(100)
```

Pop Quiz 1

What types of library sources are supported in Apache Spark in Azure Synapse Analytics?

Maven

Spark

PyPi



Pop Quiz 1

What types of library sources are supported in Apache Spark in Azure Synapse Analytics?

Maven

Spark

PyPi



Machine Learning in Azure Synapse Analytics



Microsoft Spark Utilities

Overview

It provides utilities for working with file systems, including ADLS Gen2 and Azure Blob Storage.

Benefits

It supports multiple methods for file systems such as List, Copy, Move, Write, Append, Delete file or directory, View file properties, Create new directory, Preview file content.

It supports environment utilities to get username, user id, job id, workspace name, pool name, cluster id.

It supports to get the access tokens of linked services and manage secrets in Azure Key Vault.

The screenshot shows a Jupyter Notebook interface with three code cells. Cell 1 shows the import of mssparkutils and its help function. Cell 2 shows the import of mssparkutils.credentials and its help function. Cell 3 shows the import of mssparkutils.env and its help function. Each cell includes the command, execution time, and the resulting help documentation for the respective module.

Cell 1:

```
[2]: 1 from notebookutils import mssparkutils  
2 mssparkutils.fs.help()
```

Command executed in 437ms by prlangad on 11-24-2020 18:32:02.018 -08:00

mssparkutils.fs provides utilities for working with various FileSystems.

Below is overview about the available methods:

```
cp(from: String, to: String, recurse: Boolean = false): Boolean -> Copies a file or directory, possibly across FileSystems  
mv(from: String, to: String, recurse: Boolean = false): Boolean -> Moves a file or directory, possibly across FileSystems  
ls(dir: String): Array -> Lists the contents of a directory  
mkdirs(dir: String): Boolean -> Creates the given directory if it does not exist, also creating any necessary parent directories  
put(file: String, contents: String, overwrite: Boolean = false): Boolean -> Writes the given String out to a file, encoded in UTF-8  
head(file: String, maxBytes: int = 1024 * 100): String -> Returns up to the first 'maxBytes' bytes of the given file as a String encoded in UTF-8  
append(file: String, content: String, createFileIfNotExists: Boolean): Boolean -> Append the content to a file  
rm(dir: String, recurse: Boolean = false): Boolean -> Removes a file or directory
```

Use mssparkutils.fs.help("methodName") for more info about a method.

Cell 2:

```
[3]: 1 mssparkutils.credentials.help()
```

Command executed in 355ms by prlangad on 11-24-2020 18:32:47.633 -08:00

```
getToken(audience, name): returns AAD token for a given audience, name (optional)  
isValidToken(token): returns true if token hasn't expired  
getConnectionStringOrCreds(linkedService): returns connection string or credentials for linked service  
getSecret(akvName, secret, linkedService): returns AKV secret for a given AKV linked service, akvName, secret key  
getSecret(akvName, secret): returns AKV secret for a given akvName, secret key  
putSecret(akvName, secretName, secretValue, linkedService): puts AKV secret for a given akvName, secretName  
putSecret(akvName, secretName, secretValue): puts AKV secret for a given akvName, secretName
```

Cell 3:

```
[4]: 1 mssparkutils.env.help()
```

Command executed in 472ms by prlangad on 11-24-2020 18:33:14.526 -08:00

```
getUser(): returns user name  
getUser(): returns unique user id  
getJobId(): returns job id  
getWorkspaceName(): returns workspace name  
getPoolName(): returns Spark pool name  
getClusterId(): returns cluster id
```

Hyperspace

Overview

Hyperspace introduces the ability for Apache Spark users to create indexes on their data

Benefits

It helps accelerate your workloads or queries containing filters on predicates with high selectivity or a join that requires heavy shuffles.

Maintain the indexes through a multi-user concurrency model.

Leverage these indexes automatically, within your Spark workloads, without any changes to your application code for query/workload acceleration.

It supports index operations as create index, list index, restore index, delete index, vacuum index

Languages supported: Scala, Python, .NET

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** Cell ▾, Run all, Undo, Publish, Attach to analytics1, Language PySpark (Python).
- Status Bar:** Not started.
- Create indexes:** A section with Cell 4 containing Python code to create indexes from configurations:

```
[ ] 1 # Create indexes from configurations
2
3 hyperspace.createIndex(emp_DF, emp_IndexConfig)
4 hyperspace.createIndex(dept_DF, dept_IndexConfig1)
5 hyperspace.createIndex(dept DF, dept IndexConfig2)
```
- List indexes:** A section with Cell 6 containing Python code to list indexes:

```
[ ] 1 hyperspace.indexes().show()
```
- Index usage:** A section with Cell 8 containing Python code to enable Hyperspace and demonstrate its usage with filters and explainability:

```
[ ] 1 # Enable Hyperspace
2 Hyperspace.enable(spark)
3
4 emp_DF = spark.read.parquet(emp_Location)
5 dept_DF = spark.read.parquet(dept_Location)
6
7 emp_DF.show(5)
8 dept_DF.show(5)
9
10 # Filter with equality predicate
11 eqFilter = dept_DF.filter("deptId = 20").select("deptName")
12 eqFilter.show()
13
14 hyperspace.explain(eqFilter, True, displayHTML)
```

Spark CDM connector

Overview

It offers Spark dataframes to read and write entities in a CDM folder.

Benefits

It supports use of Managed Identities for Azure resources to mediate access to the Azure datalake storage.

Writes from a Spark dataframe to an entity in a CDM folder based on dataframe schema or CDM entity definition.

Supports data in Apache Parquet format and CSV format.

The CDM connector is pre-installed and supports languages: Python, Scala

```
1 # Explicit write, creating an entity in a CDM folder based on a pre-defined model
2
3 # Case 1: Using an entity definition defined in the CDM Github repo
4
5 data = [
6     ["1", "2", "3", 4],
7     ["4", "5", "6", 8],
8     ["7", "8", "9", 4],
9     ["10", "11", "12", 8],
10    ["13", "14", "15", 4]
11 ]
12
13 schema = (StructType()
14     .add(StructField("teamMembershipId", StringType(), True))
15     .add(StructField("systemUserId", StringType(), True))
16     .add(StructField("teamId", StringType(), True))
17     .add(StructField("versionNumber", LongType(), True))
18 )
19
20 df = spark.createDataFrame(spark.sparkContext.parallelize(data,1), schema)
21
22 (df.write.format("com.microsoft.cdm")
23     .option("storage", storageAccountName)
24     .option("manifestPath", container + "/explicitTest/root.manifest.cdm.json")
25     .option("entity", "TeamMembership")
26     .option("entityDefinitionPath", "core/applicationCommon/TeamMembership.cdm.json/TeamMembership")
27     .option("useCdmStandardModelRoot", True) # sets the model root to the CDM CDN schema documents folder
28     .option("useSubManifest", True)
29     .mode("overwrite")
30     .save())
31
32 readDf = (spark.read.format("com.microsoft.cdm")
33     .option("storage", storageAccountName)
34     .option("manifestPath", container + "/explicitTest/root.manifest.cdm.json")
35     .option("entity", "TeamMembership")
36     .load())
37
38 readDf.select("*").show()
```

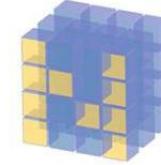
Examples of tools for data science used in Notebooks

Core frameworks and tools



pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



NumPy



Machine Learning & Deep Learning



APACHE Spark™



Visualization



plotly

seaborn

matplotlib



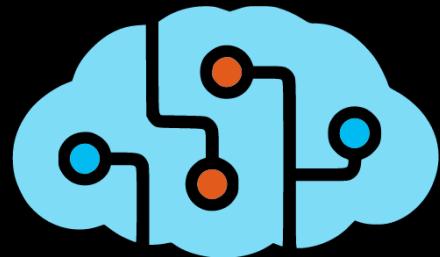
Spark ML Algorithms

Spark ML
Algorithms

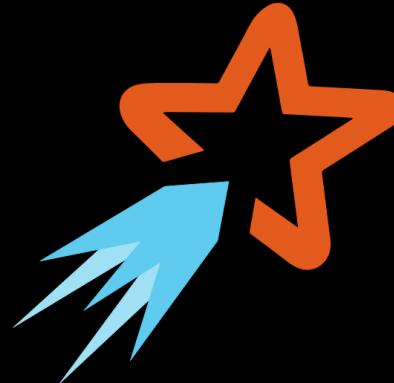
Classification and Regression	<ul style="list-style-type: none">• Linear Models (SVMs, logistic regression, linear regression)• Naïve Bayes• Decision Trees• Ensembles of trees (Random Forest, Gradient-Boosted Trees)• Isotonic regression
Clustering	<ul style="list-style-type: none">• k-means and streaming k-means• Gaussian mixture• Power iteration clustering (PIC)• Latent Dirichlet allocation (LDA)
Collaborative Filtering	<ul style="list-style-type: none">• Alternating least squares (ALS)
Dimensionality Reduction	<ul style="list-style-type: none">• SVD• PCA
Frequent Pattern Mining	<ul style="list-style-type: none">• FP-growth• Association rules
Basic Statistics	<ul style="list-style-type: none">• Summary statistics• Correlations• Stratified sampling• Hypothesis testing• Random data generation



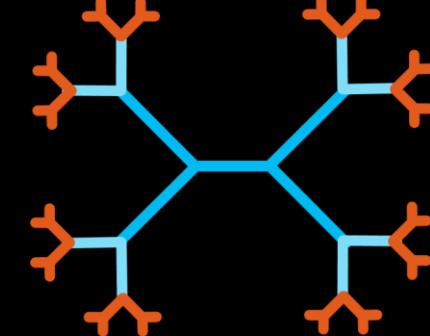
SynapseML



Cognitive Services on Spark



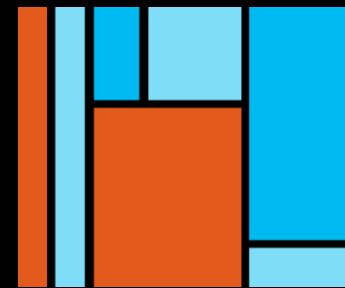
Stress Free Serving



Lightning Fast Gradient Boosting



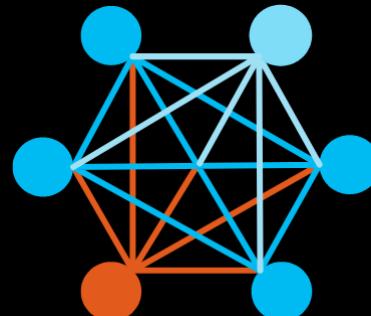
Fast & Sparse Text Analytics



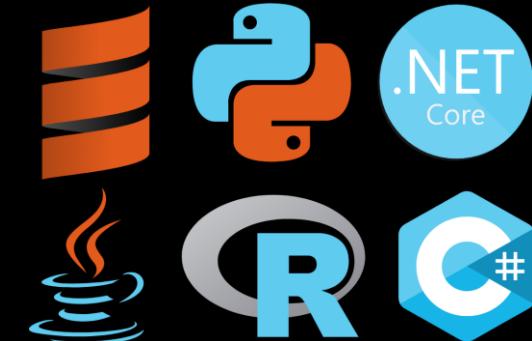
Distributed Microservices



Large Scale Model Interpretability

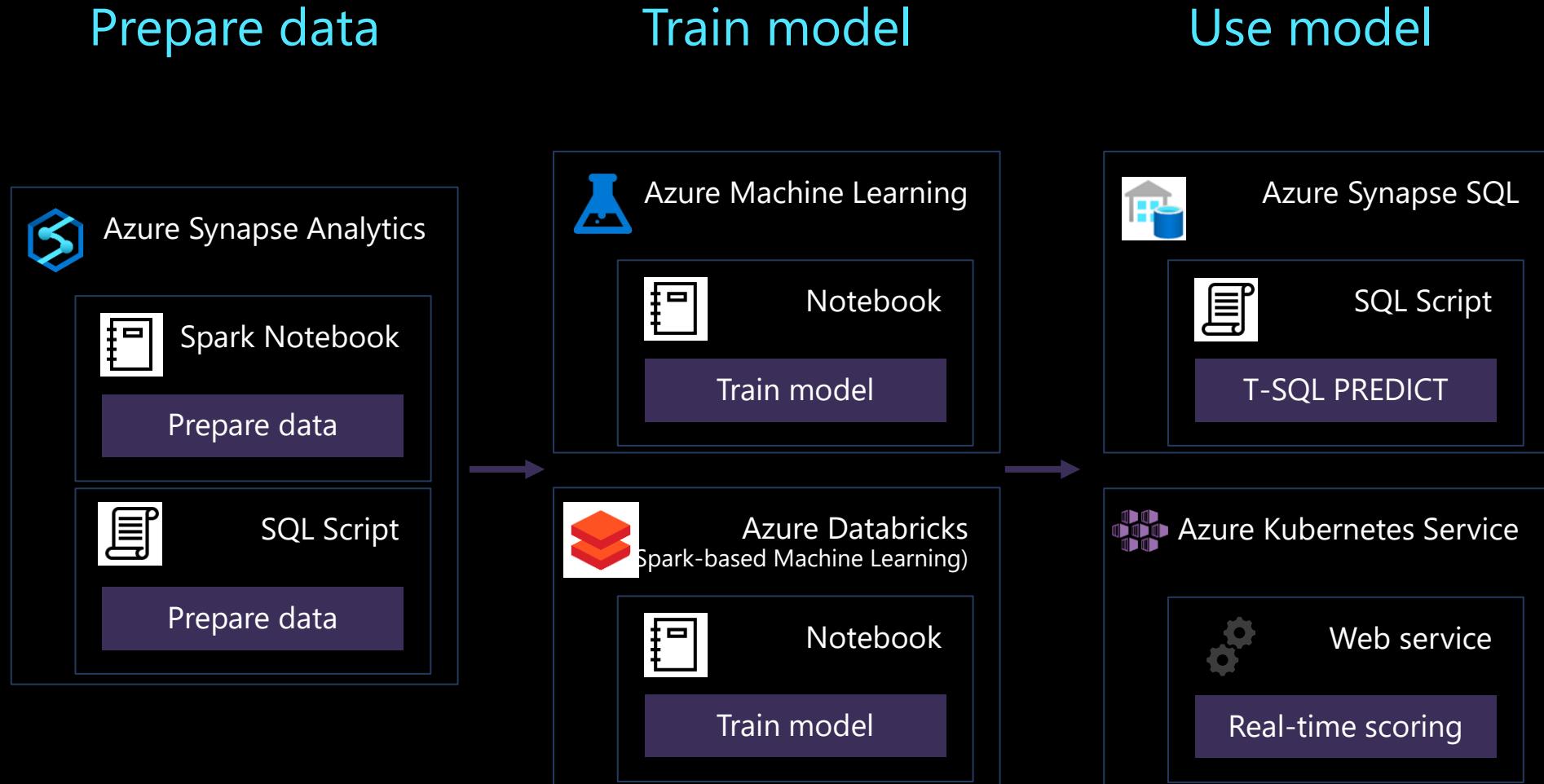


Scalable Deep Learning



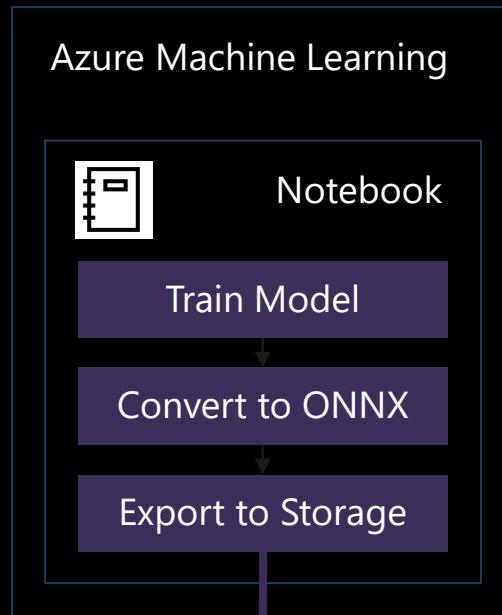
Broad Language Support

Synapse Analytics and the Machine Learning Process

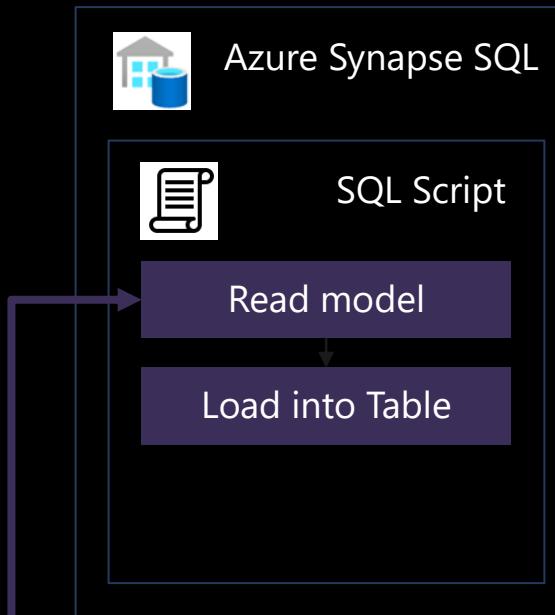


Making predictions with T-SQL

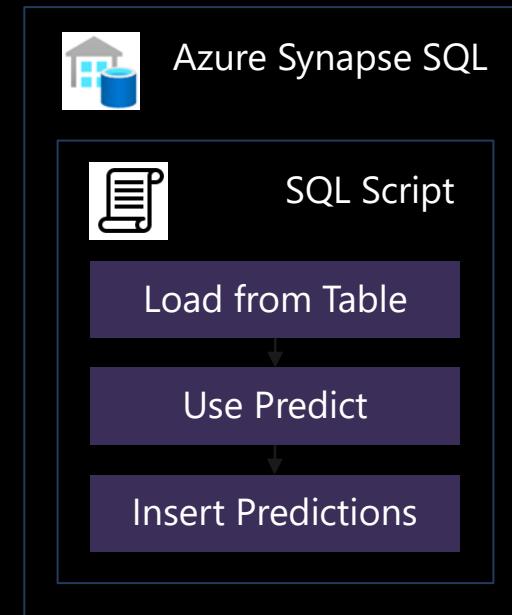
Create the model



Register the model



Use the model



Predict

Overview

It provides ability to import existing machine learning models and score them within provisioned SQL. It takes ONNX (Open Neural Network Exchange) and data as inputs and generates prediction based on model.

Benefits

1. It empowers data engineers to successfully deploy machine learning models with the familiar T-SQL interface
2. It offers seamless collaboration with data scientists
3. It generates new columns, but the number of columns and their data types depends on the type of model that was used for prediction.

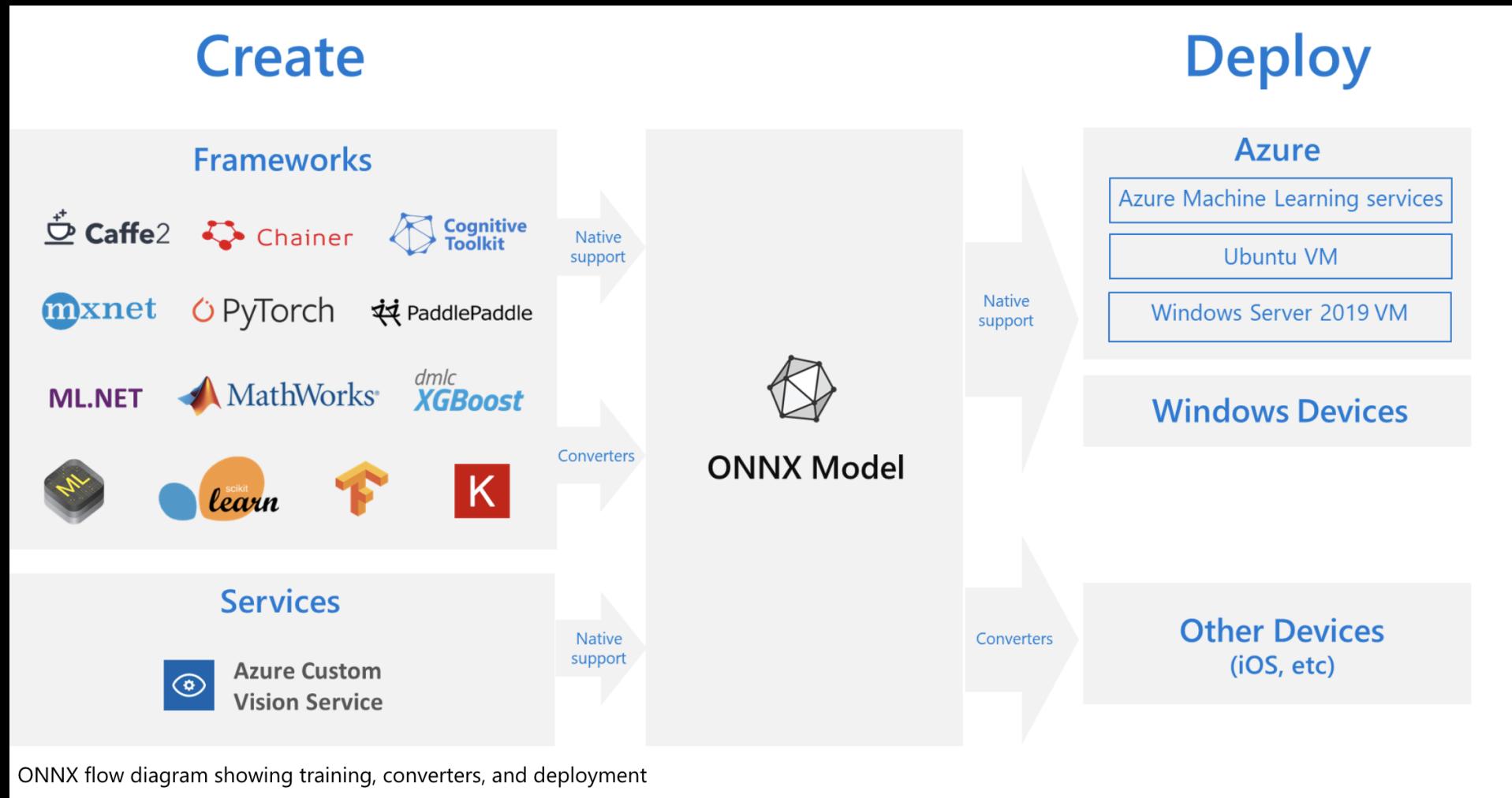
Syntax:

```
PREDICT
(
    MODEL = @model | model_literal,
    DATA = object AS <table_alias>
)
WITH ( <result_set_definition> )
<result_set_definition> ::= 
{
    { column_name
        data_type
    }
    [,...n]
}
MODEL = @model | model_literal
```

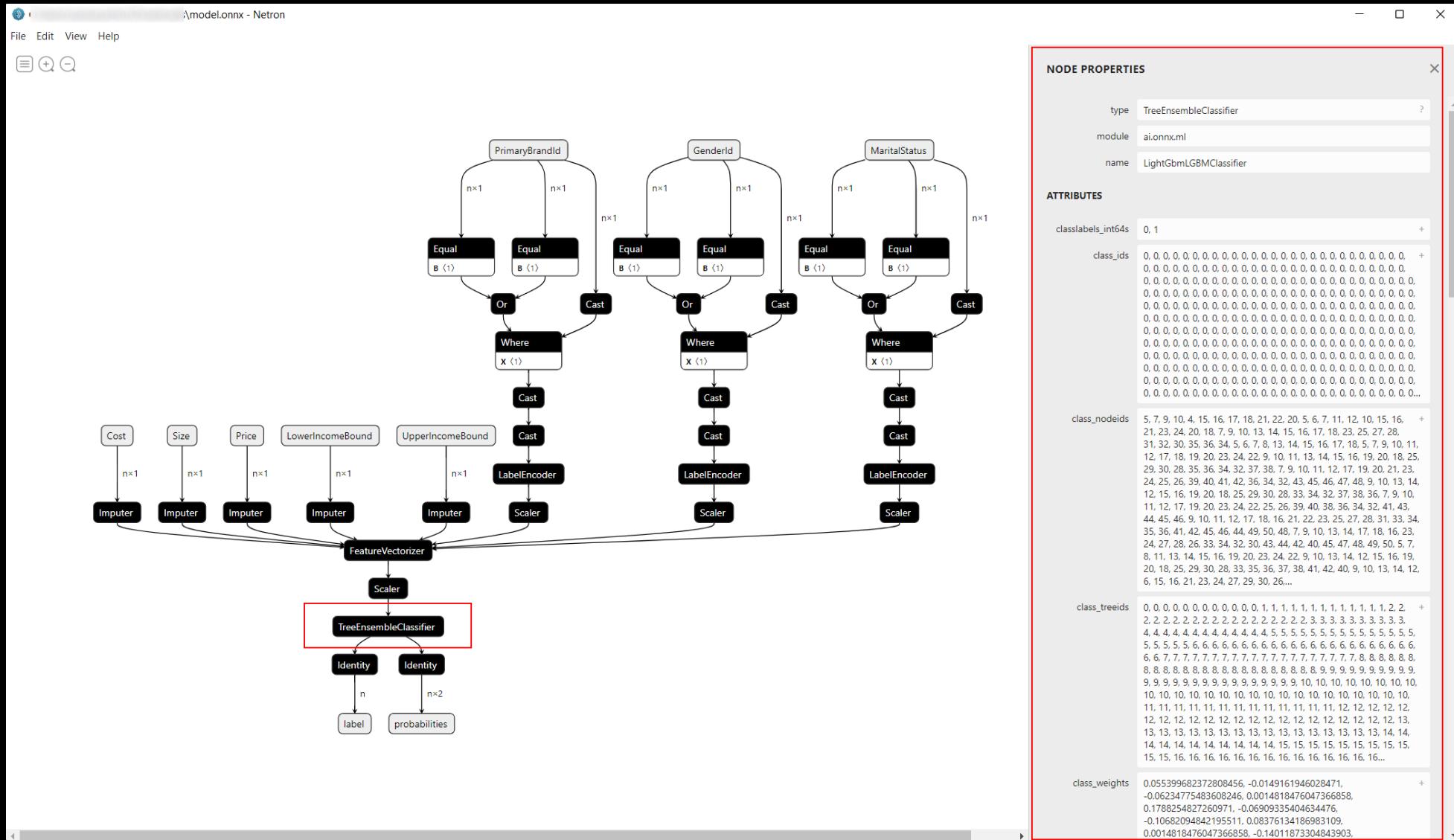
Example:

```
DECLARE @model varbinary(max) = (SELECT Model FROM Models WHERE Id = <>);
SELECT d.*, p.Score
FROM PREDICT(MODEL = @model,
    DATA = dbo.mytable AS d) WITH (Score float) AS p;
```

What is ONNX?



What is ONNX?



Synapse Notebook: Connect to AML workspace

The screenshot shows the Azure Synapse Analytics notebook interface. The left sidebar lists resources under 'Develop': SQL scripts, Notebooks (selected), Data flows, Spark job definitions, and Power BI. The main area displays a notebook cell titled 'Check the Azure ML Core SDK Version to Validate Your Installation'. The code in Cell 3 is:

```
[5] 1 import azureml.core  
2 print("SDK Version:", azureml.core.VERSION)
```

The output shows the command was executed in 1s 258ms by balapv on 11-12-2019 14:41:52.805 -08:00, and the result is 'SDK Version: 1.0.69'.

Below this, a section titled 'Connect to Azure Workspace' is shown. Cell 5 contains the following code:

```
[6] 1 ## Import the Workspace class and check the Azure ML SDK version.  
2 from azureml.core import Workspace  
3  
4 ws = Workspace(subscription_id = "6560575d-fa06-4e7d-95fb-f962e74efd7a",  
5 | | | | | resource_group = "balapv-synapse-rg", workspace_name = "AML-WS-synapse")  
6  
7 print(ws.name, ws.location, ws.resource_group, sep='\t')
```

The output shows the command was executed in 3s 909ms by balapv on 11-12-2019 14:41:55.491 -08:00, and the result is 'AML-WS-synapse westus2 balapv-synapse-rg'.

Cell 6 contains the following code:

```
[7] 1 # import modules  
2 import azureml.core  
3 import pandas as pd  
4 from azureml.core.authentication import ServicePrincipalAuthentication  
5 from azureml.core.workspace import Workspace  
6 from azureml.core.experiment import Experiment
```

At the bottom, there are buttons for 'Running', 'Stop session', 'Spark history server', and 'Configure session'.

A red arrow points from the text 'Simple code to connect workspace' to the code in Cell 5.

Simple code to connect workspace

Synapse Notebook: Configure AML job to run on Synapse

The screenshot shows the Azure Synapse Analytics notebook interface. On the left, the sidebar lists notebooks, including one named "automl_synapse_local_regr...". The main area displays the "Train" section of the notebook, which includes a table of configuration parameters and a code cell.

Configuration parameters:

Property	Description
task	classification or regression
primary_metric	This is the metric that you want to optimize.
iteration_timeout_minutes	Time limit in minutes for each iteration.
iterations	Number of iterations. In each iteration AutoML trains a specific pipeline with the data.
X	(sparse) array-like, shape = <code>n_samples, n_features</code>
y	(sparse) array-like, shape = <code>n_samples,</code> Multi-class targets.
enable_onnx_compatible_models	Enable the ONNX compatible models in the experiment.
path	Relative path to the project folder. AutoML stores configuration files for the experiment under this folder. You can specify a new empty folder.

Code Cell 13:

```
1 automl_config = AutoMLConfig(task = 'regression',
2                               debug_log = 'automl_errors.log',
3                               primary_metric = 'normalized_root_mean_squared_error',
4                               iteration_timeout_minutes = 10,
5                               iterations = 20,
6                               preprocess = True,
7                               n_cross_validations = 2,
8                               max_concurrent_iterations = 2, #spark compute size
9                               verbosity = logging.INFO,
10                              spark_context=sc, #spark related
11                              enable_onnx_compatible_models=True, # This will generate ONNX compatible models.
12                              cache_store=True,
13                              X = X_train,
14                              y = y_train)
```

Command executed in 630ms by balapv on 11-12-2019 15:03:57.443 -08:00

Call the `submit` method on the experiment object and pass the run configuration. Execution of local runs is synchronous. Depending on the data and the number of iterations this can run for a while. In this example, we specify `show_output = True` to print currently running iterations to the console.

Ready | Stop session | Spark history server | Configure session

Synapse Notebook: Run AML job

The screenshot shows a Microsoft Azure Synapse Analytics notebook titled "automl_synapse_local_regression". The notebook interface includes a left sidebar with categories like "Develop", "SQL scripts", "Notebooks", "Data flows", "Spark job definitions", and "Power BI". The main workspace displays a cell titled "Run AutoML job" containing Python code to submit an experiment. The output of the cell shows the execution command, logs indicating the experiment is running on a spark cluster, and a detailed table of iteration results. A red arrow points from the text "ML job execution result" to the table of results.

Run AutoML job

Cell 15

```
1 local_run = experiment.submit(automl_config, show_output = True)
```

Command executed in 12mins 34s 972ms by balapv on 11-12-2019 15:17:53.089 -08:00

Running an experiment on spark cluster: automl-local-regression-Synapse.
Parent Run ID: AutoML_ad8600ab-a1ab-4b6b-b233-059d969e0a0e

```
*****  
ITERATION: The iteration being evaluated.  
PIPELINE: A summary description of the pipeline being evaluated.  
DURATION: Time taken for the current iteration.  
METRIC: The result of computing score on the fitted pipeline.  
BEST: The best observed score thus far.  
*****
```

ITERATION	PIPELINE	DURATION	METRIC	BEST
1	StandardScalerWrapper ElasticNet	0:00:38	0.0021	0.0021
2	StandardScalerWrapper ElasticNet	0:00:32	0.0054	0.0021
0	StandardScalerWrapper ElasticNet	0:01:20	0.0004	0.0004
4	StandardScalerWrapper RandomForest	0:00:33	0.0179	0.0004
3	StandardScalerWrapper ElasticNet	0:00:36	0.0036	0.0004
5	StandardScalerWrapper LightGBM	0:00:28	0.0109	0.0004
6	MaxAbsScaler DecisionTree	0:00:34	0.0168	0.0004
7	MaxAbsScaler RandomForest	0:00:41	0.0104	0.0004
8	MaxAbsScaler DecisionTree	0:01:05	0.0077	0.0004
9	MaxAbsScaler DecisionTree	0:00:48	0.0086	0.0004
10	StandardScalerWrapper DecisionTree	0:00:39	0.0058	0.0004
11	MaxAbsScaler DecisionTree	0:00:45	0.0096	0.0004
13	MaxAbsScaler ExtremeRandomTrees	0:00:47	0.0147	0.0004
12	MaxAbsScaler ExtremeRandomTrees	0:01:54	0.0096	0.0004
14	StandardScalerWrapper ElasticNet	0:00:39	0.0027	0.0004
15	StandardScalerWrapper ElasticNet	0:00:54	0.0010	0.0004
16	StandardScalerWrapper ElasticNet	0:00:48	0.0023	0.0004
17	MaxAbsScaler ElasticNet	0:00:31	0.0239	0.0004
18	StandardScalerWrapper ElasticNet	0:00:53	0.0014	0.0004
19	VotingEnsemble	0:01:59	0.0004	0.0004

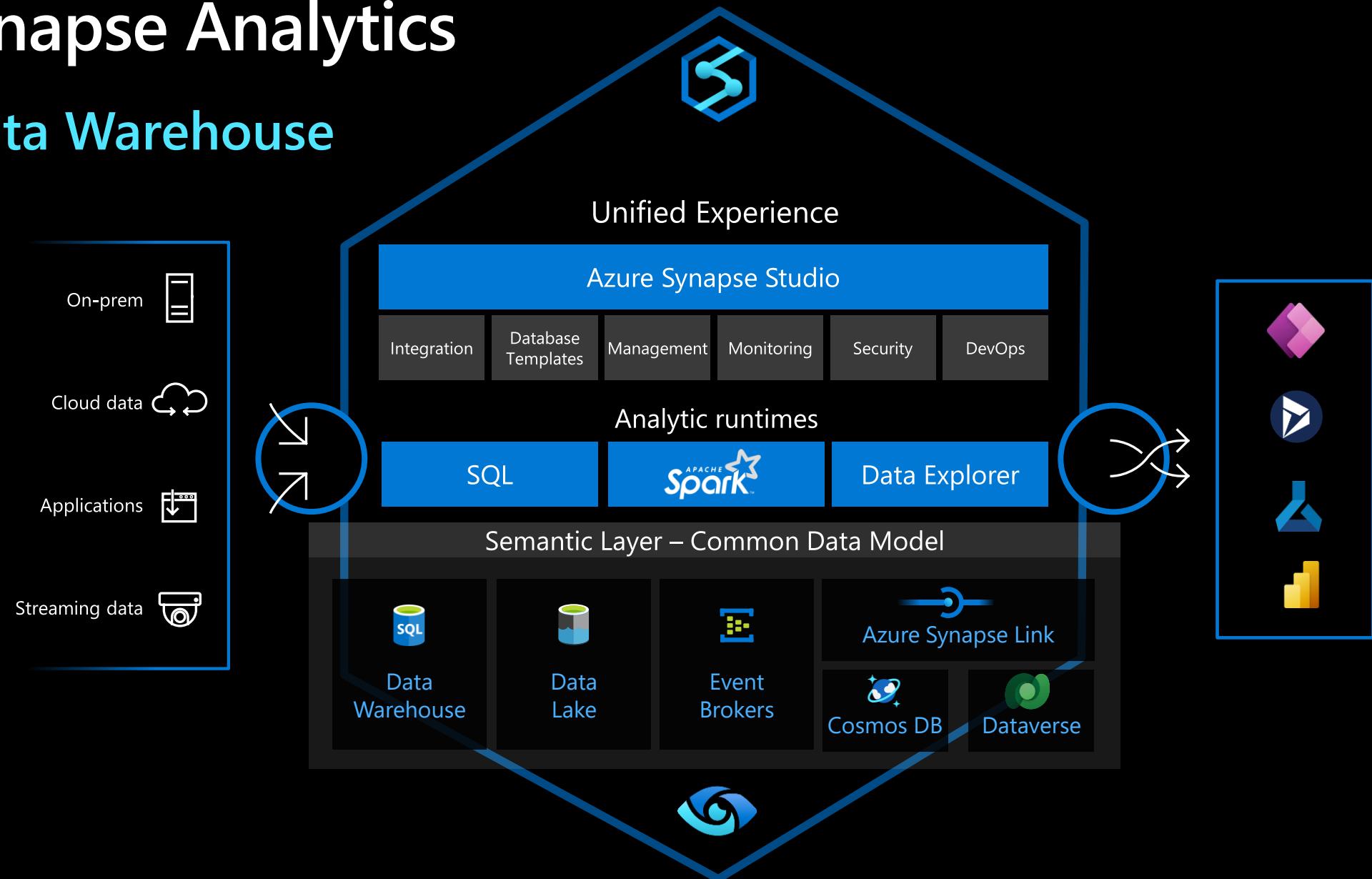
ML job execution result

Get Azure Portal URL for Monitoring Runs

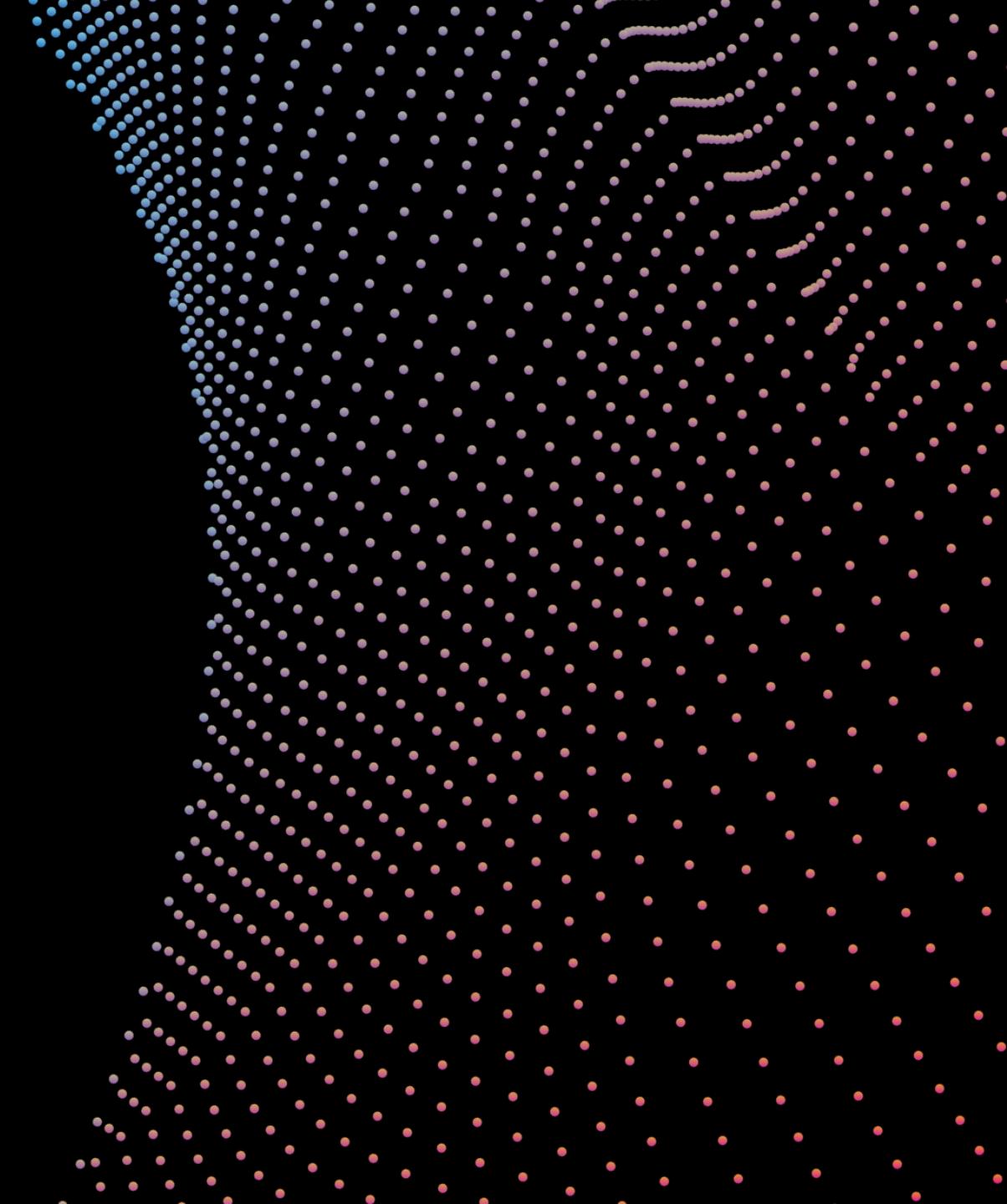
Running | Stop session | Spark history server | Configure session

Azure Synapse Analytics

Modern Data Warehouse

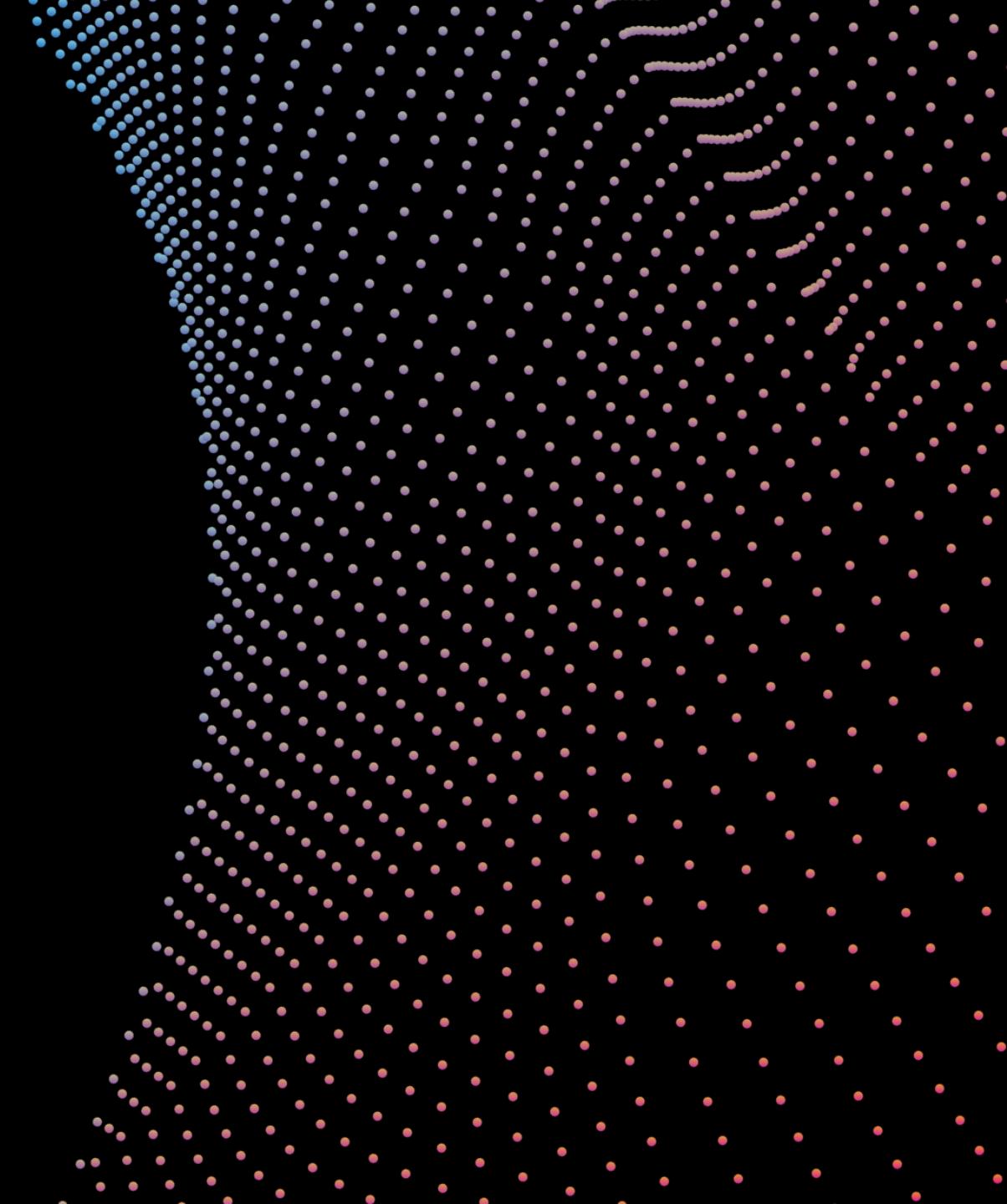


Thank You





Monitor & Manage



Agenda

- 1) **Monitoring** - Monitor resources within Synapse Studio and the Azure Portal.
- 2) **Manage** - Manage resources for optimal utilization and performance.



Discussion...?

How has the role of a DBA changed
in the era of managed Data-
Warehouses-as-a-Service?



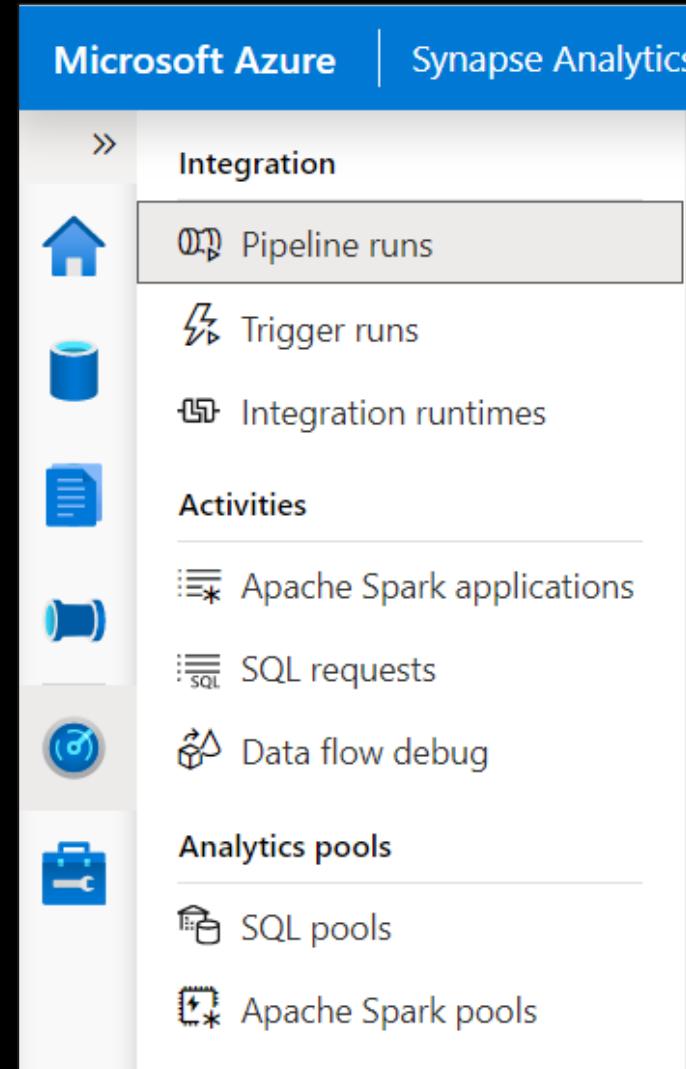
Monitor Hub

Overview

This feature provides single pane of glass to monitor orchestration, activities for Apache Spark Application and SQL requests.

Benefits

Offers additional filters to monitor specific activities or orchestration



Monitor Hub - Integration

Overview

Monitor orchestration in the Synapse workspace for the progress and status of pipeline

Pipeline runs				
All status		Rerun	Cancel	Refresh
Pipeline Name	Run Start	DURATION	Triggered By	Status
Load Data to SQLDW	10/25/2019, 3:49:42 PM	00:10:55	Manual trigger	✓ Succeeded
Copy Open Dataset	10/25/2019, 2:17:54 PM	00:14:12	Manual trigger	✓ Succeeded
Pipeline 1	10/24/2019, 1:23:43 PM	00:00:08	Manual trigger	✓ Succeeded

Benefits

Track all/specific pipelines

Monitor pipeline run and activity run details

Find the root cause of pipeline failure or activity failure

Trigger runs			
All status		Refresh	Edit columns
Trigger Name	Trigger Type	Trigger Time	Status
TriggerCopy_csvdata100	ScheduleTrigger	4/10/20, 12:14:00 AM	✓ Succeeded

Monitor Hub - Spark applications

Overview

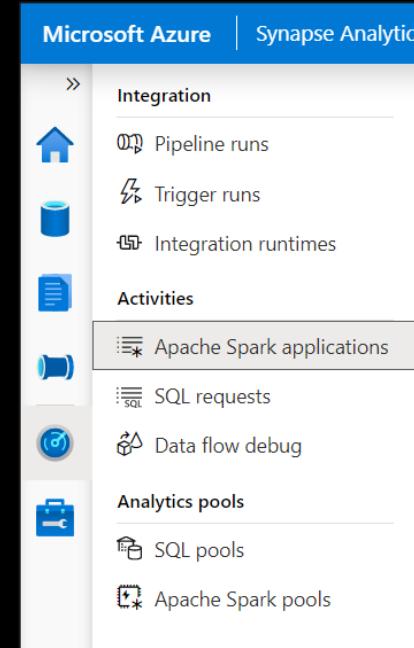
Monitor Spark pools, Spark applications for the status of activities

Benefits

Apply filter for pool to get Apache Spark Applications per pool

Additional available filters include:

1. Application Name
2. Livy ID
3. Status
4. End time



The screenshot shows the main content area of the Monitor Hub for Apache Spark applications. The sidebar on the left is identical to the one above. The main area has a header 'Apache Spark applications' with tabs for All, Spark session, Batch job, Refresh, Edit columns, and a date range of Last 30 days. Below this is a search bar with filters for Status (All) and Add filter. The main table lists 88 items, showing columns for Application name, Submitter, Submit time, Status, and Pool. The data includes entries for various notebooks and pools like automlpool, analyticspool, and hbpool, all marked as Stopped.

Application name	Submitter	Submit time	Status	Pool
Synapse_automlpool_1...	negust@microsoft.com	11/30/20, 7:23:26 PM	Stopped	automlpool
Synapse_automlpool_1...	negust@microsoft.com	11/30/20, 7:14:00 PM	Stopped	automlpool
Synapse_automlpool_1...	negust@microsoft.com	11/30/20, 5:09:00 PM	Stopped (session timed ou...	automlpool
Notebook 3_analyticscp...	negust@microsoft.com	11/30/20, 12:52:51 PM	Stopped	analyticspool
Notebook 4_analytics1...	prlangad@microsoft.com	11/24/20, 6:29:41 PM	Stopped	analytics1
Synapse_hbpool_16059...	charlesf@microsoft.com	11/20/20, 2:30:44 PM	Stopped	hbpool
Synapse_analyticspool_...	negust@microsoft.com	11/20/20, 11:20:30 AM	Stopped	analyticspool

Monitor Hub – SQL requests

Overview

Monitor SQL requests for the progress and status of activities

Benefits

Apply filter for pool to get SQL requests per compute pool

Validate query text

Additional available filters include:

1. Start time
2. End time
3. Request ID
4. Session ID
5. Submitter
6. Workload group

SQL requests						
Pacific Time (US & C... : Last 30 days)		Status : All	Pool : Predict_Pool	Add filter		
Showing 1 - 100 of 248 items						
Request ID ↑↓	Request content ↑↓	Submit time ↑↓	Duration	Submitter ↑↓	Status ↑↓	Queued duration
QID125878	USE [DWShellDb]	12/1/20, 12:27:53 AM	0s	System	✓ Completed	0s
QID125879	--Backing up Logical Azure Data	12/1/20, 12:27:53 AM	20s	System	✓ Completed	0s
QID125637	USE [DWShellDb]	11/30/20, 8:27:53 PM	0s	System	✓ Completed	0s
QID125638	--Backing up Logical Azure Data	11/30/20, 8:27:53 PM	15s	System	✓ Completed	0s
QID125529	USE [Predict_Pool]	11/30/20, 6:41:15 PM	0s	anrampal@microsoft.com	✓ Completed	0s
QID125530	SELECT s.NAME AS SchemaName	11/30/20, 6:41:15 PM	0s	anrampal@microsoft.com	✓ Completed	0s
QID125421	USE [Predict_Pool]	11/30/20, 4:54:56 PM	0s	negust@microsoft.com	✓ Completed	0s

Microsoft Azure | Synapse Analytics > wsazuresynapseanalytics

SQL requests				
Pacific Time (US & C... : Last 30 days)		Status : All	Pool : Built-in	Add filter
Showing 1 - 100 of 279 items				
Request ID ↑↓	Request content ↑↓	Submit time ↑↓	Duration	Data processed
12162198	SELECT TOP 100* FROM OPENROWSET (11/30/20, 8:57:12 PM	1s	1 MiB
11573006	SELECT TOP 100* FROM OPENROWSET (11/30/20, 6:23:32 PM	6s	1 MiB
9079654	SELECT product = ISNULL(p.pro	11/30/20, 7:28:38 AM	6s	1 MiB
9066730	SELECT product = ISNULL(p.pro	11/30/20, 7:26:17 AM	5s	1 MiB
9065769	SELECT * FROM OPENROWSET (11/30/20, 7:25:47 AM	2s	1 MiB
9062482	SELECT * FROM OPENROWSET (11/30/20, 7:25:17 AM	18s	18 MiB

Monitor Hub – Analytics pools

Integration

Home Pipeline runs Trigger runs Integration runtimes Activities Apache Spark applications SQL requests Data flow debug Analytics pools SQL pools Apache Spark pools

SQL pools

Refresh Edit columns

Pool : All

Showing 1 - 6 of 6 items

Pool name ↑↓	Type ↑↓	Status ↑↓	Size ↑↓	CPU utilizati... ↑↓	Memory utili... ↑↓	Created on ↑↓
Built-in	Serverless	✓ Online	Auto	N/A	N/A	N/A
newpool	Dedicated	● Paused	DW200c	-	-	11/5/2020 5:42:52 AM
NYCTaxi_Pool	Dedicated	✓ Online	DW100c	1.1	23	8/27/2020 6:03:32 AM
Predict_Pool	Dedicated	✓ Online	DW1000c	-	5.33	9/9/2020 1:13:26 AM
Streaming_Pool	Dedicated	● Paused	DW2000c	-	-	8/27/2020 4:04:18 AM
WWI_Pool	Dedicated	✓ Online	DW100c	0.12	20.22	8/26/2020 7:18:23 PM

Integration

Home Pipeline runs Trigger runs Integration runtimes Activities Apache Spark applications SQL requests Data flow debug Analytics pools SQL pools Apache Spark pools

Apache Spark pools

Refresh Edit columns

Pool : analytics

Showing 1 - 4 of 4 items

Pool name ↑↓	Size	Active users	Allocated vCores	Allocated memory (GB)	Created on
analytics2	Medium (8 vCores / 64 GB) - 10 nodes	0	0	0	10/30/20, 2:47:16 PM
analytics1	Medium (8 vCores / 64 GB) - 3 to 10 nodes	0	0	0	10/28/20, 3:36:18 AM
AnalyticsPool99	Medium (8 vCores / 64 GB) - 3 to 10 nodes	0	0	0	9/10/20, 7:17:53 AM
analyticspool	Medium (8 vCores / 64 GB) - 3 to 15 nodes	0	0	0	8/26/20, 7:13:44 PM

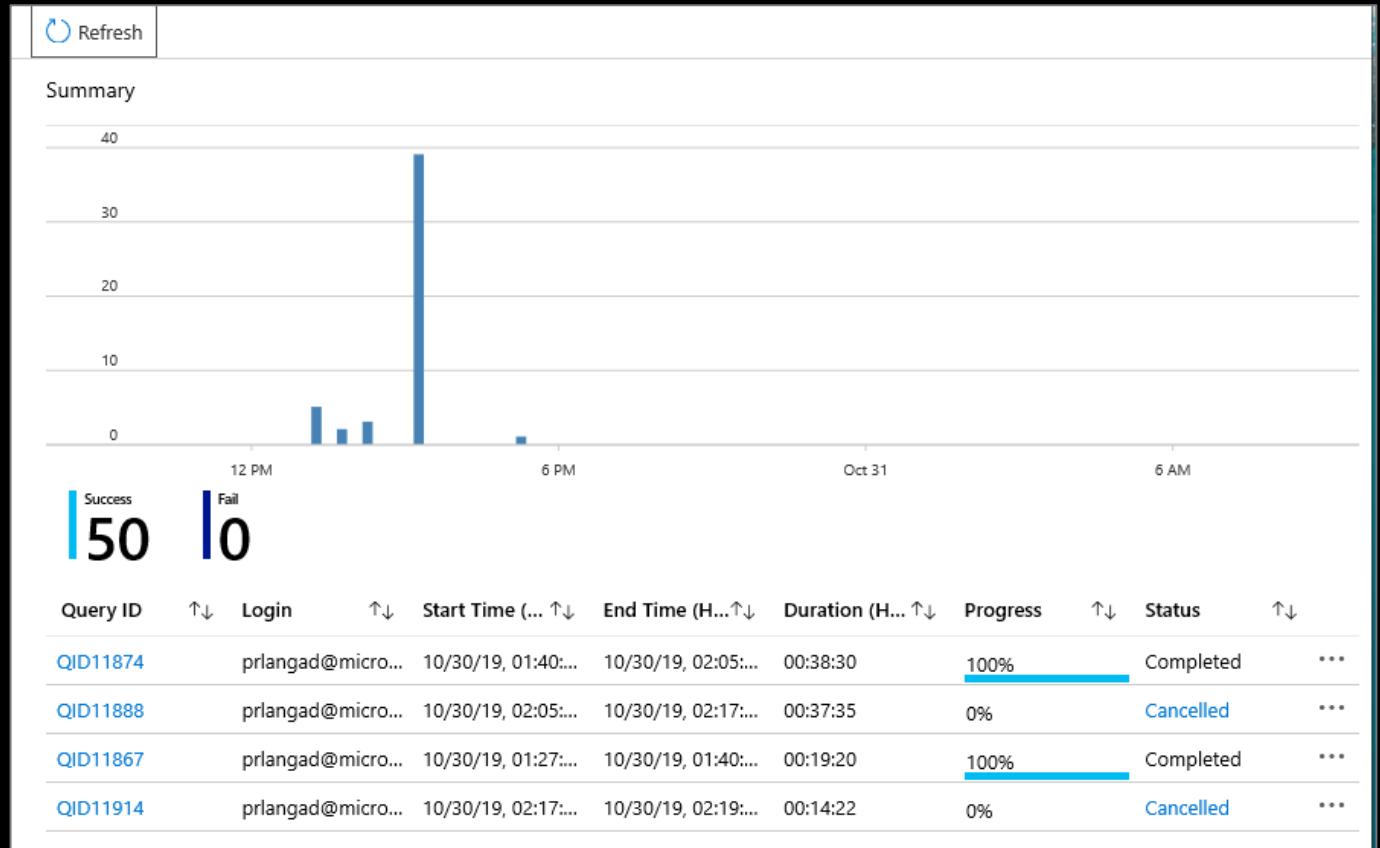
Monitor SQL Pools

Overview

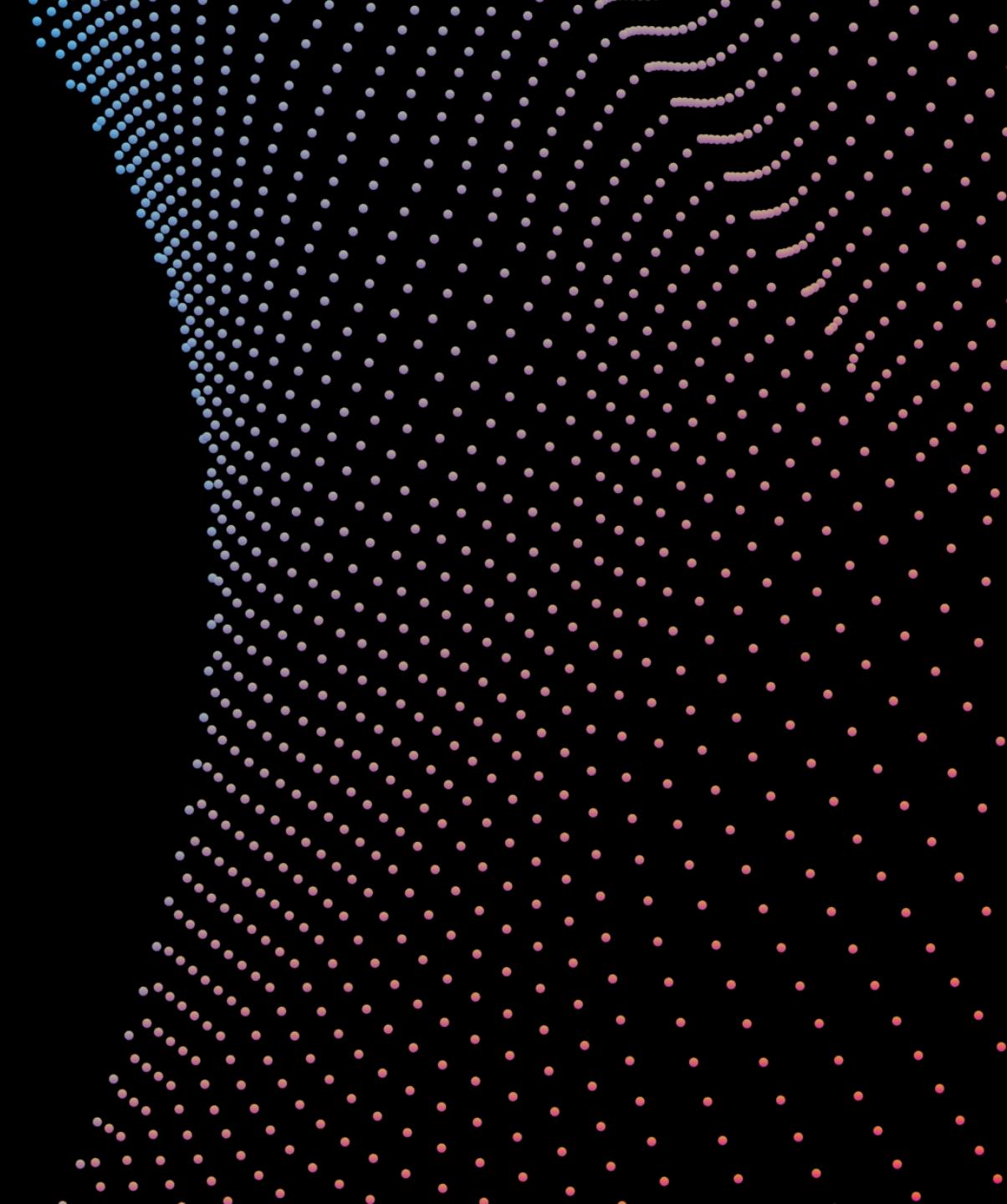
Monitor SQL Pool in Azure Portal for overall usage and query activities.

Benefits

- Access SQL Audit Logs for my SQL computes
- Monitor status and progress of all/specific activities
- Dashboard view to monitor performance
- Get to know scale of SQL compute resource



Manage



Workload Management

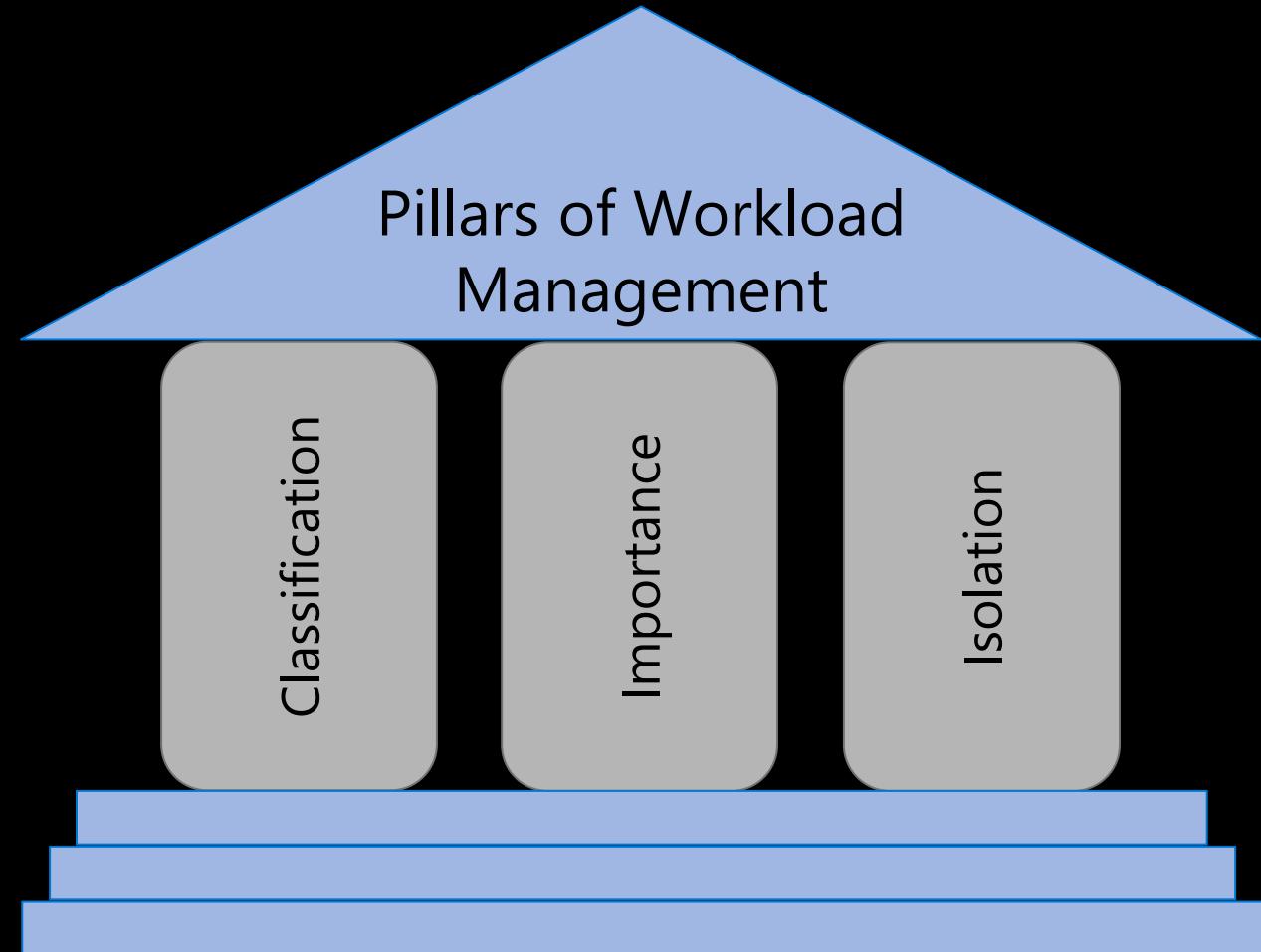
Overview

It manages resources, ensures highly efficient resource utilization, and maximizes return on investment (ROI).

Synapse is moving away from Resource Class and Concurrency Slots to Workload Management.

The three pillars of workload management are

- Workload Classification – To assign a request to a workload group and setting importance levels.
- Workload Importance – To influence the order in which a request gets access to resources.
- Workload Isolation – To reserve resources for a workload group.



Workload classification

Overview

Map queries to allocations of resources via pre-determined rules.

Use with workload importance to effectively share resources across different workload types.

If a query request is not matched to a classifier, it is assigned to the default workload group.

Benefits

Map queries to both Resource Management and Workload Isolation concepts.

Monitoring DMVs

`sys.workload_management_workload_classifiers`

`sys.workload_management_workload_classifier_details`

Query DMVs to view details about all active workload classifiers.

```
CREATE WORKLOAD CLASSIFIER classifier_name
WITH
(
    WORKLOAD_GROUP = 'name'
    , MEMBERNAME   = 'security_account'
    [[,] IMPORTANCE  = {LOW|BELOW_NORMAL|NORMAL|ABOVE_NORMAL|HIGH} ]
    [[,] WLM_LABEL   = 'label' ]
    [[,] WLM_CONTEXT = 'name' ]
    [[,] START_TIME  = 'start_time' ]
    [[,] END_TIME    = 'end_time' ]
)[;]
```

WORKLOAD_GROUP: maps to an existing resource class

IMPORTANCE: specifies relative importance of request

MEMBERNAME: database user, role, AAD login or AAD group

Workload importance

Overview

Queries past the concurrency limit enter a FiFo queue

By default, queries are released from the queue on a first-in, first-out basis as resources become available

Workload importance allows higher priority queries to receive resources immediately regardless of queue

Example Video

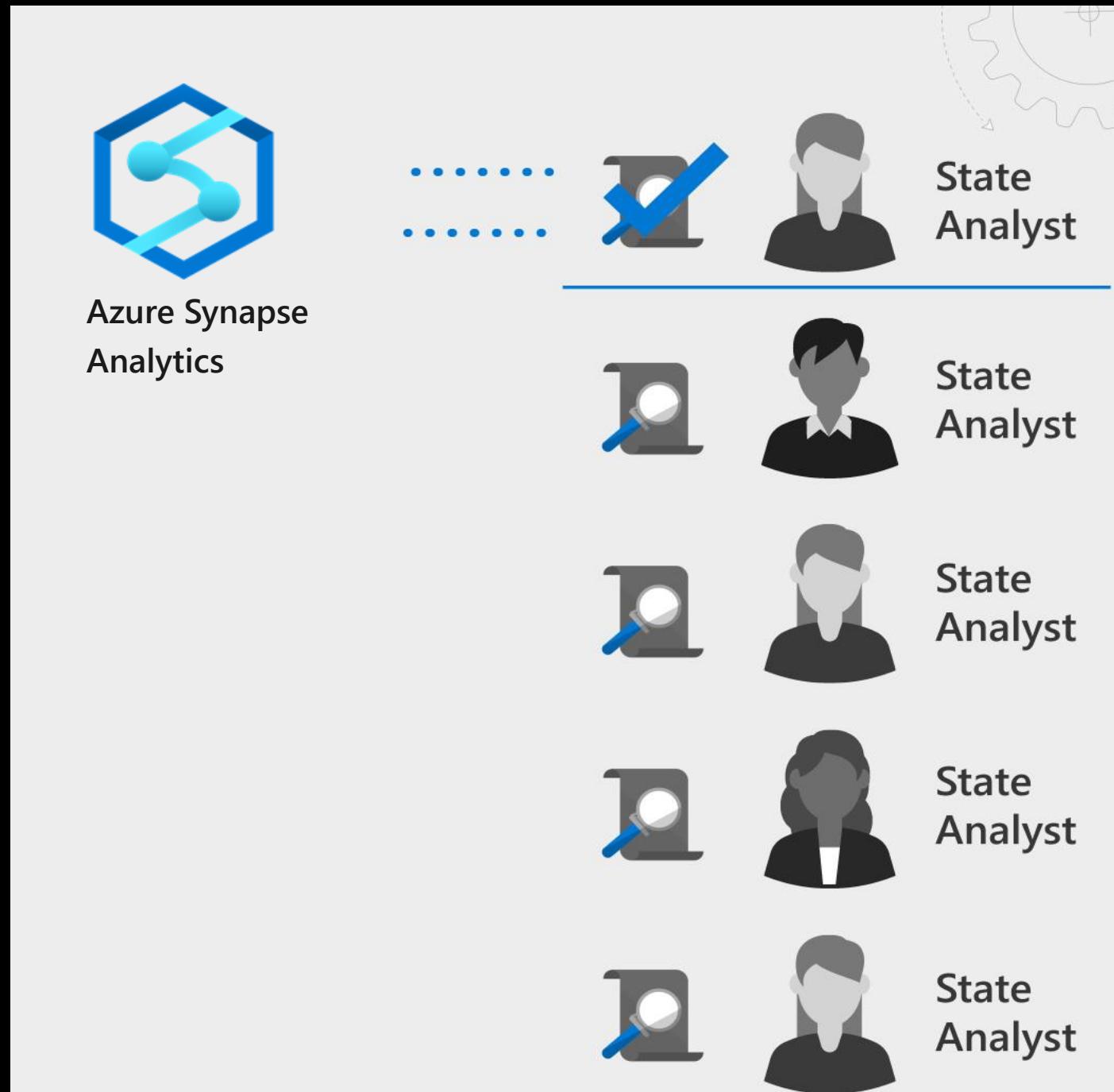
State analysts have normal importance.

National analyst is assigned high importance.

State analyst queries execute in order of arrival

When the national analyst's query arrives, it jumps to the top of the queue

```
CREATE WORKLOAD CLASSIFIER National_Analyst  
WITH  
(  
    WORKLOAD_GROUP = 'analyst'  
, IMPORTANCE = HIGH  
, MEMBERNAME = 'National Analyst Login')
```



Workload Isolation

Overview

Allocate fixed resources to workload group.

Assign maximum and minimum usage for varying resources under load. These adjustments can be done live without having to take SQL Analytics offline.

Benefits

Reserve resources for a group of requests

Limit the amount of resources a group of requests can consume

Shared resources accessed based on importance level

Set Query timeout value. Get DBAs out of the business of killing runaway queries

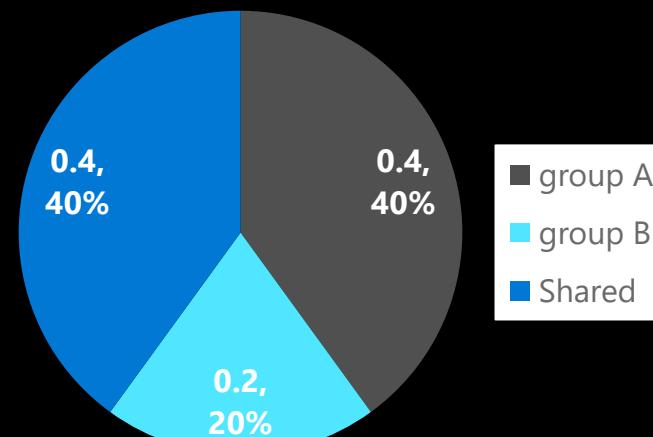
Monitoring DMVs

`sys.workload_management_workload_groups`

Query to view configured workload group.

```
CREATE WORKLOAD GROUP group_name
WITH
(
    MIN_PERCENTAGE_RESOURCE = value
    , CAP_PERCENTAGE_RESOURCE = value
    , REQUEST_MIN_RESOURCE_GRANT_PERCENT = value
    [[,] REQUEST_MAX_RESOURCE_GRANT_PERCENT = value ]
    [[,] IMPORTANCE = {LOW | BELOW_NORMAL | NORMAL | ABOVE_NORMAL | HIGH} ]
    [[,] QUERY_EXECUTION_TIMEOUT_SEC = value ]
)[;]
```

RESOURCE ALLOCATION



Discussion...?

What are some of the advantages and disadvantages of workload isolation versus SQL Pool scaling?



Continuous integration and delivery (CI/CD)

Overview

Database project support in SQL Server Data Tools (SSDT) allows teams of developers to collaborate over a version-controlled Azure Synapse Analytics, and track, deploy and test schema changes.

Benefits

Database project support includes first-class integration with Azure DevOps. This adds support for:

- **Azure Pipelines** to run CI/CD workflows for any platform (Linux, macOS, and Windows)
- **Azure Repos** to store project files in source control
- **Azure Test Plans** to run automated check-in tests to verify schema updates and modifications
- Growing ecosystem of third-party integrations that can be used to complement existing workflows (Timetracker, Microsoft Teams, Slack, Jenkins, etc.)

The screenshot displays the Azure DevOps interface, specifically the CI/CD pipeline configuration and marketplace integration sections.

Top Section (Pipeline Configuration): Shows the "Source type" section with options for "Build", "Azure Repos", "GitHub" (selected), and "TFVC". It includes fields for "Service" (github connection), "Source (repository)" (Devworkspacegit), "Default branch" (workspace_publish), "Default version" (Latest from the default branch), and "Source alias" (Artifacts_source). A blue "Add" button is at the bottom right.

Middle Section (Marketplace Integration): Shows the "Add tasks" section with a search bar containing "synap" and a red box highlighting it. Below is a card for the "Synapse-Deployment-Task" by Microsoft Corporation, which is a "Build and release CI/CD task for synapse workspace". A red box highlights the "Add" button next to the card.

Bottom Section (Pipeline Editor): Shows the "All pipelines > New release pipeline" editor. It displays a pipeline named "Staging-env-Artifacts" with an "Agent job" step. A "Synapse deployment task for workspace: workspacedeploy" step is being configured. The configuration pane shows fields for "Display name" (Synapse deployment task for workspace: workspacedeploy), "Template" (\$System.DefaultWorkingDirectory/Template), "Template parameters" (\$System.DefaultWorkingDirectory/parameters), "Synapse workspace connection type" (Manage), "Subscription of target workspace" (Scoped to subscription), "Synapse workspace resource group" (Resource group), "Synapse workspace name" (workspacedeploy), "OverrideArmParameters" (None), and "Advanced", "Control Options", and "Output Variables" sections.

Azure Advisor recommendations

Suboptimal Table Distribution

Reduce data movement by replicating tables

Data Skew

Choose new hash-distribution key

Slowest distribution limits performance

Cache Misses

Provision additional capacity

Tempdb Contention

Scale or update user resource class

Suboptimal Plan Selection

Create or update table statistics

The screenshot shows the Azure Advisor recommendations interface. At the top, a message says "You have free Azure Advisor recommendations!". Below it, a brief description of Azure Advisor is provided, along with a link to "Learn more". The interface is divided into four main sections: "High Availability", "Security", "Performance", and "Cost". Each section contains a summary of recommendations and impacted resources. A large green box on the right indicates "3,323 USD savings/mo *". At the bottom, there is a blue button labeled "View my free recommendations".

Category	Recommendations	Impact	Resources
High Availability	5 Recommendations	1 High impact, 4 Medium impact, 0 Low impact	36 impacted resources
Security	20 Recommendations	20 High impact, 0 Medium impact, 0 Low impact	133 impacted resources
Performance	1 Recommendation	1 High impact, 0 Medium impact, 0 Low impact	5 impacted resources
Cost	3 Recommendations	2 High impact, 1 Medium impact, 0 Low impact	29 impacted resources

Maintenance windows

Overview

Choose a time window for your upgrades.

Select a primary and secondary window within a seven-day period.

Windows can be from 3 to 8 hours.

24-hour advance notification for maintenance events.

Benefits

- Ensure upgrades happen on your schedule.
- Predictable planning for long-running jobs.
- Stay informed of start and end of maintenance.

The screenshot shows the 'Maintenance Schedule (preview)' blade in the Azure portal. At the top, there's a navigation bar with 'Home > maintenanceexamples > Maintenance Schedule (preview)'. Below it is a header with 'Maintenance Schedule (preview)', a 'Save' button, a 'Discard' button, and a 'Feedback' link. A sidebar on the left contains various icons for different Azure services. The main content area has an informational box with an info icon: 'Maintenance on your data warehouse could occur once a week within one of two maintenance windows. Choose the primary and secondary windows that best suit your operational needs. If you would like to use the maintenance windows already defined, no action is required.' It also states 'Maintenance will not take place outside these windows unless we notify you in advance.' Below this is a section titled 'Choose primary window' with two radio buttons: 'Saturday - Sunday' (selected) and 'Tuesday - Thursday'. The 'Primary maintenance window' section shows 'Day' set to 'Saturday', 'Start time' at '03:00 UTC', and 'Time window' at '8 hours'. The 'Secondary maintenance window' section shows 'Day' set to 'Tuesday', 'Start time' at '13:00 UTC', and 'Time window' at '8 hours'. At the bottom, there's a 'Schedule summary' section with the text 'Primary maintenance window Saturday 03:00 UTC (8 hours)' and 'Secondary maintenance window Tuesday 13:00 UTC (8 hours)'.

Manage Summary

- Resource contention can be mitigated by applying Workload Management in Azure Synapse Analytics.
 - **Workload Classification:** Assign a request to a workload group and set importance levels.
 - **Workload Importance:** Influence the order in which a request gets access to resources.
 - **Workload Isolation:** Reserve resources for a workload group.
- Use Azure Advisor recommendations to identify suboptimal table distribution, data skew, cache misses, tempdb contention, and suboptimal plan selection.
- Avoid disruptive system downtime due to system upgrades by configuring maintenance windows and notifications in Azure Synapse Analytics.

Monitoring Activity Mentimeter Poll

Scan QR Code
or

Go to www.menti.com and use code

7894 1936

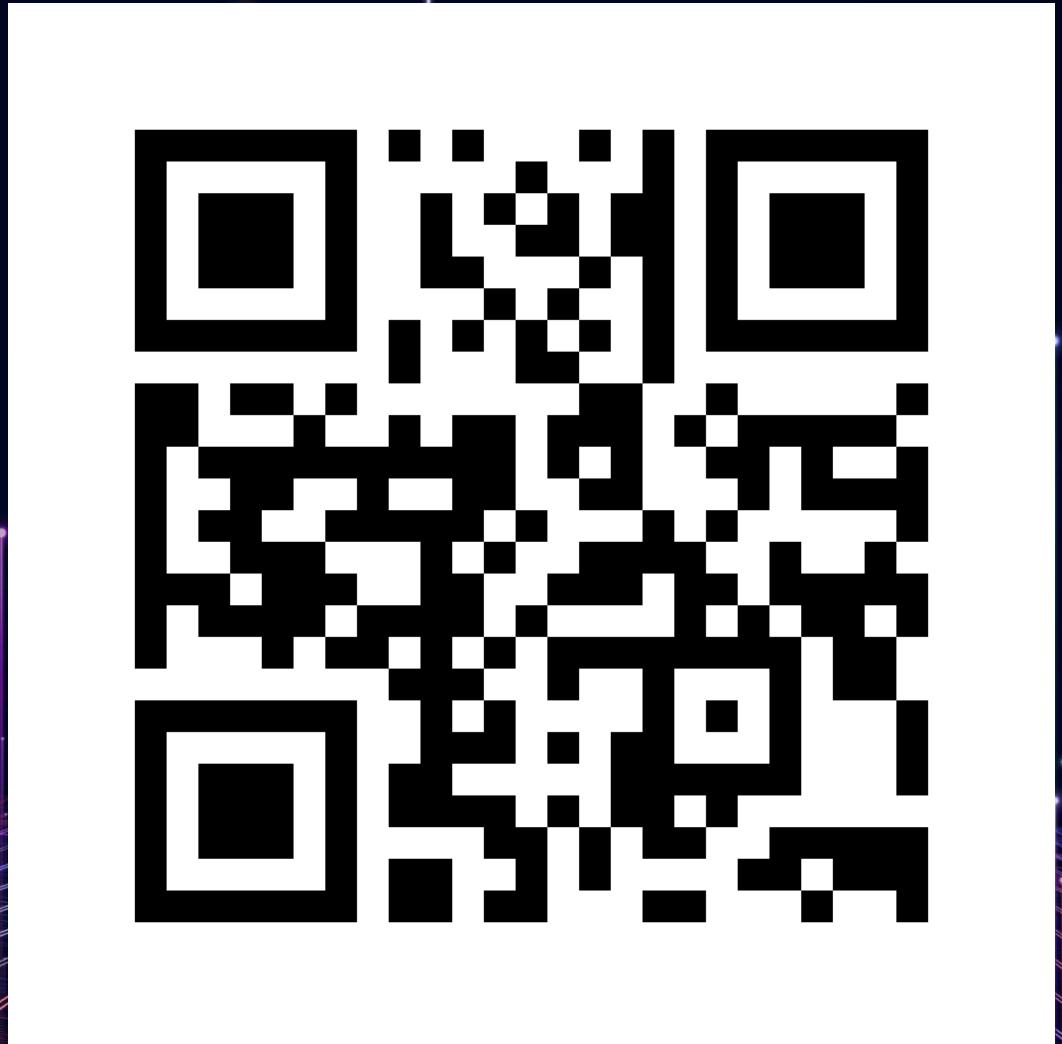


Overall Training Mentimeter Poll

Scan QR Code
or

Go [to www.menti.com](http://www.menti.com) and use code

4743 6920



Thank You

