

Creating Spark Pool

Azure Synapse Analytics offers Spark pools as part of its analytics service. These Spark pools are managed by Apache Spark clusters provided by Microsoft Azure. They allow you to run big data analytics and machine learning workloads on large datasets.

Here's how they typically work:

1. **Scalability:** Azure Spark pools can scale up or down based on workload demands, allowing you to allocate resources as needed.
2. **Integration:** They integrate with other Azure services, such as Azure Data Lake Storage, Azure Blob Storage, and Azure SQL Data Warehouse (now part of Azure Synapse Analytics).
3. **Managed Service:** Azure manages the infrastructure, so you don't have to worry about provisioning or managing the underlying virtual machines.
4. **Cost Optimization:** You pay for the resources you use, and you can optimize costs by scaling down or pausing the Spark pool when it's not in use.
5. **Compatibility:** They support Apache Spark APIs, libraries, and tools, making it easier to migrate existing Spark workloads to Azure.

Use cases of Spark Pool:

Azure Spark pools can be used in various use cases across industries for big data processing, analytics, and machine learning. Here are some common examples:

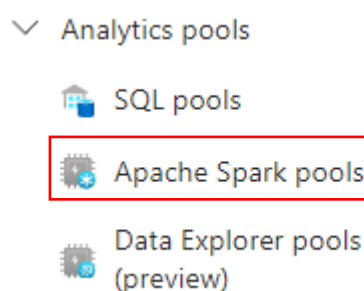
1. **Data Exploration and Analysis:** Analysts and data scientists can use Spark pools to explore and analyze large datasets efficiently. They can run SQL queries, perform data transformations, and visualize results using tools like Apache Zeppelin or Jupyter notebooks.
2. **ETL (Extract, Transform, Load) Pipelines:** Spark pools are ideal for building ETL pipelines to ingest data from multiple sources, transform it into the desired format, and load it into data warehouses or data lakes for further analysis.
3. **Real-time Stream Processing:** Organizations can use Spark pools to process and analyze streaming data in real-time. This could include tasks like event detection, anomaly detection, and aggregations on data streams from IoT devices, social media platforms, or financial transactions.
4. **Machine Learning and Predictive Analytics:** Data scientists can leverage Spark pools to build and train machine learning models on large datasets. Spark's MLlib library provides scalable machine learning algorithms for tasks such as classification, regression, clustering, and collaborative filtering.
5. **Data Warehousing:** Azure Spark pools can complement data warehousing solutions like Azure Synapse Analytics by offloading complex analytical workloads to Spark for faster processing. This allows organizations to perform advanced analytics on their data warehouse data without impacting performance.

6. **Customer Analytics and Personalization:** Retailers and e-commerce companies can use Spark pools to analyze customer behavior, preferences, and purchase history to personalize marketing campaigns, recommend products, and optimize pricing strategies.
7. **Fraud Detection and Security Analytics:** Financial institutions and cybersecurity firms can utilize Spark pools to analyze large volumes of transaction data and network logs for detecting fraudulent activities, identifying security threats, and implementing proactive measures to mitigate risks.
8. **Healthcare Analytics:** Healthcare organizations can leverage Spark pools to analyze electronic health records (EHRs), medical imaging data, and genomic data for clinical research, disease prediction, and personalized medicine initiatives.

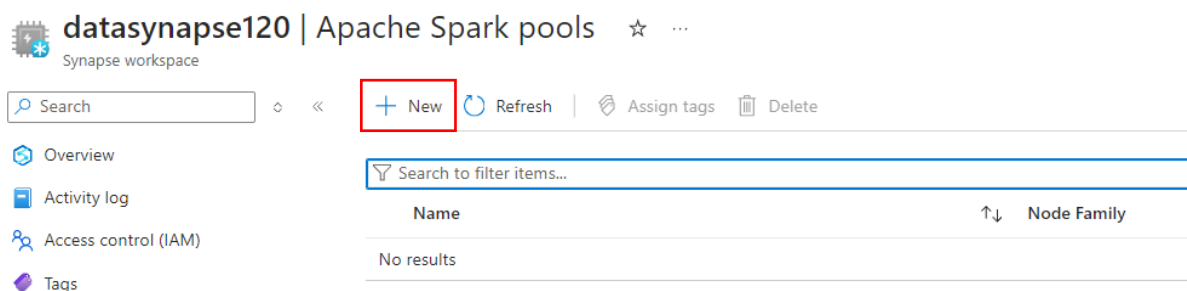
In this guide, we're setting up and utilizing Spark pools in Azure Synapse Analytics for big data processing and analytics. The end goal is to enable users to efficiently analyze large datasets, perform ETL tasks, run real-time stream processing, conduct machine learning and predictive analytics, and more, using the scalability and integration capabilities provided by Spark pools. By following the steps outlined, users can create a Spark pool, set up a notebook in Synapse Studio, write and execute code, and ultimately derive insights from their data to make data-driven decisions.

😊 To begin with the Lab:

1. There are some prerequisites for this lab, and they are you should have Azure Synapse Analytics in place along with a separate storage account in which you should have your CSV and Parquet file in place too.
2. Now to begin with, in your Synapse Analytics, from the left pane, you need to choose Apache Spark Pools.



3. Then you need to click on new to create one.



- Now you need to give it a name and then choose the node size and number of nodes then keep it to a minimum. Also, you need to disable auto-scale.

New 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 *	<input type="text" value="sparkpool120"/> ✓
Node size family	MemoryOptimized
Node size *	<input type="text" value="Small (4 vCores / 32 GB)"/> ▼
Autoscale * ⓘ	<input type="radio"/> Enabled <input checked="" type="radio"/> Disabled
Number of nodes *	<input type="range" value="3"/> 3
Dynamically allocate executors ⓘ	<input type="radio"/> Enabled <input checked="" type="radio"/> Disabled
Estimated price ⓘ	<div>Est. cost per hour 144.66 INR View pricing details</div>

- Now you need to keep the additional settings page to default just read out the settings it provides you.

* 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="3.4"/> ▼
Python	3.10
Scala	2.12.17
Java	11
Delta Lake	2.4
R	4.2.2

Spark configuration

To specify additional properties on the Apache Spark pool, select a configuration. This will be used as the default configuration for all jobs on this pool. Spark configurations can only be created and managed in Synapse Studio.

[Learn more](#) ↗

Apache Spark configuration ⓘ ▼

[View configurations](#)

Packages

Configure settings related to packages for this pool.

Allow session level packages ⓘ

☒ Enabled ☐ Disabled

Intelligent cache

Reserve space for Synapse storage cache. [Learn more](#) ↗

Intelligent cache size ⓘ

6. Then move to the review page and create your spark pool. Once the deployment is complete you can click on go to resources.

Microsoft.Azure.SynapseAnalytics.SparkPool-20240513124310 | Overview

Deployment

Search x << Delete Cancel Redeploy Download Refresh

Overview

Inputs

Outputs

Template

✓ Your deployment is complete

Deployment name : Microsoft.Azure.SynapseAnalytics.SparkPool-20240513124310 Start time : 13/5/2024, 12:52:08 pm

Subscription : Azure Pass - Sponsorship Correlation ID : e8a62ff3-b126-4750-9d2b-2ce918d31003

Resource group : new-grp

> Deployment details

< Next steps


Go to resource

Give feedback

Tell us about your experience with deployment

7. But to start working with Spark Pool you need to go back to synapse and open your synapse studio.


Getting started



Open Synapse Studio

Start building your fully-integrated analytics solution and unlock new insights.

[Open](#)



Read documentation

Learn how to be productive quickly. Explore concepts, tutorials, and samples.

[Learn more](#)

8. In your Synapse studio you need to go to the develop section and open a new notebook.

Microsoft Azure | Synapse Analytics ▶ datasynapse120

We use optional cookies to provide a better experience. [Learn more](#)

Synapse live Validate all Publish all

Develop

Filter resources by name

Home

Data

Develop

Integrate

Monitor

Manage

+ <<

SQL script

KQL script

Notebook

Data flow

Apache Spark job definition

Browse gallery

Import

9. The first thing you need to do is give your notebook a name.

Properties

General

Related (0)

Name *

01_basics

Description

10. Now you need to check your notebook is attached to the spark pool and the language should be scala.

Attach to sparkpool120 Language Spark (Scala)



11. Now you are going to write the code here in the notebook cells. When you are done with the code you need to click on the highlighted option to configure and run your code.

Run all | Undo | Publish | Outline | Attach to sparkpool120 Language Spark (Scala)

Not started

```
1 val dataValues = Array(20, 34, 44, 23, 34)
2 val dataSet = sc.parallelize(dataValues)
3
4 val maxValue=dataSet.max
5 maxValue
```

12. Then you need to set the number of executors to 1 and click on save.

 **sparkpool120**
Refresh at 1:03:44 PM 

Small (4 vCores / 28 GB) 3 nodes
0.00% utilized

Available session sizes ⓘ
12 vCores available in the Workspace

Small

2 executors

Use

Executor size * ⓘ

Small (4 vCores, 28GB memory) ▼



Dynamically allocate executors * ⓘ

☐ Enabled ☒ Disabled

Executors * ⓘ

1

13. Then click on the run button to run your code.
14. Now it might take around 3-4 minutes because it has to start an Apache spark session.
Once we have this session in place then we can run the notebooks that we have.

```
1  val dataValues = Array(20, 34, 44, 23, 34)
2  val dataSet = sc.parallelize(dataValues)
3
4  val maxValue=dataSet.max
5  maxValue
```

15. Below you can see that we got the output also the command executed in 6 seconds but the spark session took almost 3 minutes to start.

M ↴ ↻ ⌵ ⌵



```
1  val dataValues = Array(20, 34, 44, 23, 34)
2  val dataSet = sc.parallelize(dataValues)
3
4  val maxValue=dataSet.max
5  maxValue
```

[1] ✓ 2 min 47 sec park session started in 2 min 40 sec 590 ms. Command executed in 6 sec 838 ms by pulkitkumar2711 on 1:12:57 F

> **Job execution** Succeeded **Spark** 1 executors 4 cores

[View in monitoring](#) [Open Spark UI](#)

...
dataValues: Array[Int] = Array(20, 34, 44, 23, 34)
dataSet: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:30
maxValue: Int = 44
res4: Int = 44