Base-line Apache Spark performance with Apache Spark history server in Azure Synapse Analytics

Note:

You are not required to complete the processes, tasks, activities, or steps presented in this example. The various samples provided are for illustrative purposes only and it's likely that if you try this out you will encounter issues in your system.

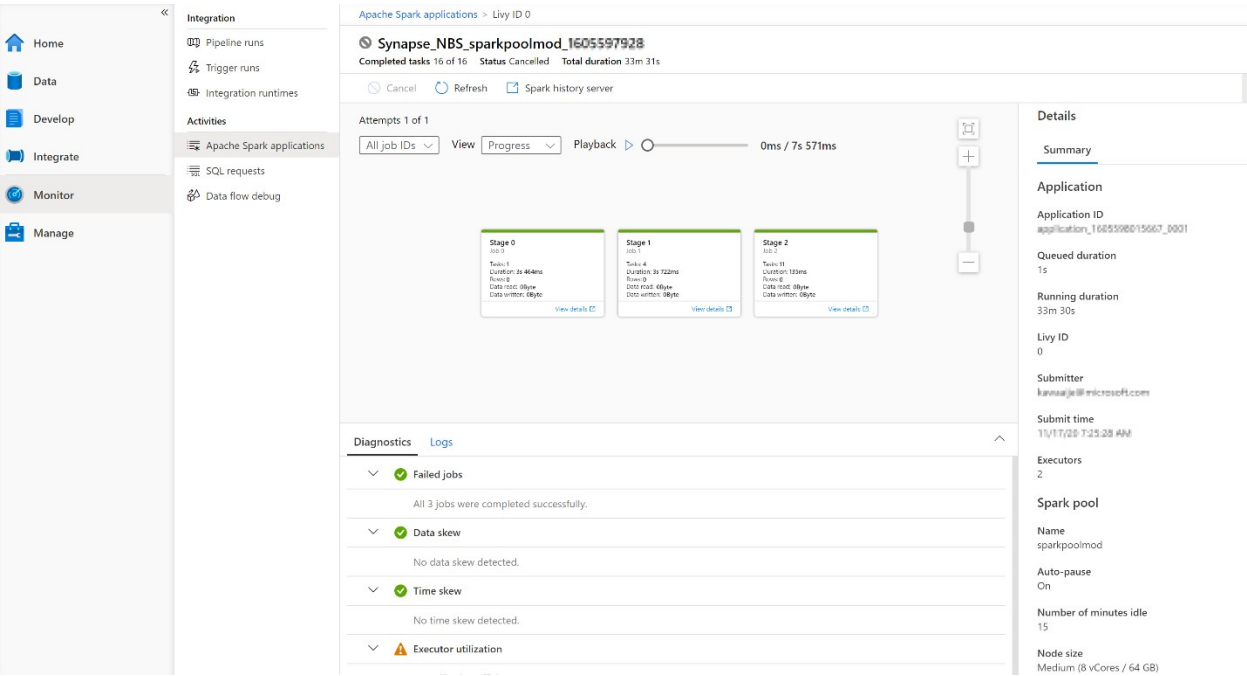
The Apache Spark history server can be used to debug and diagnose completed and running Apache Spark applications. You can use the Apache Spark history server web UI from the Azure Synapse Studio environment.

Once you launch it, there are several tabs that you can use in order to monitor the Apache Spark application:

- Jobs
- Stages
- Storage
- Environment
- Executors
- SQL

The Apache Spark history server is the web user interface known as the *Spark UI*, and is used to view completed and running Apache Spark applications.

If you want to navigate to the Apache Spark History server, you can navigate to the Azure Synapse Analytics Studio environment and go to the **Monitor** tab. In the Monitor tab, you can select **Apache Spark Applications**.

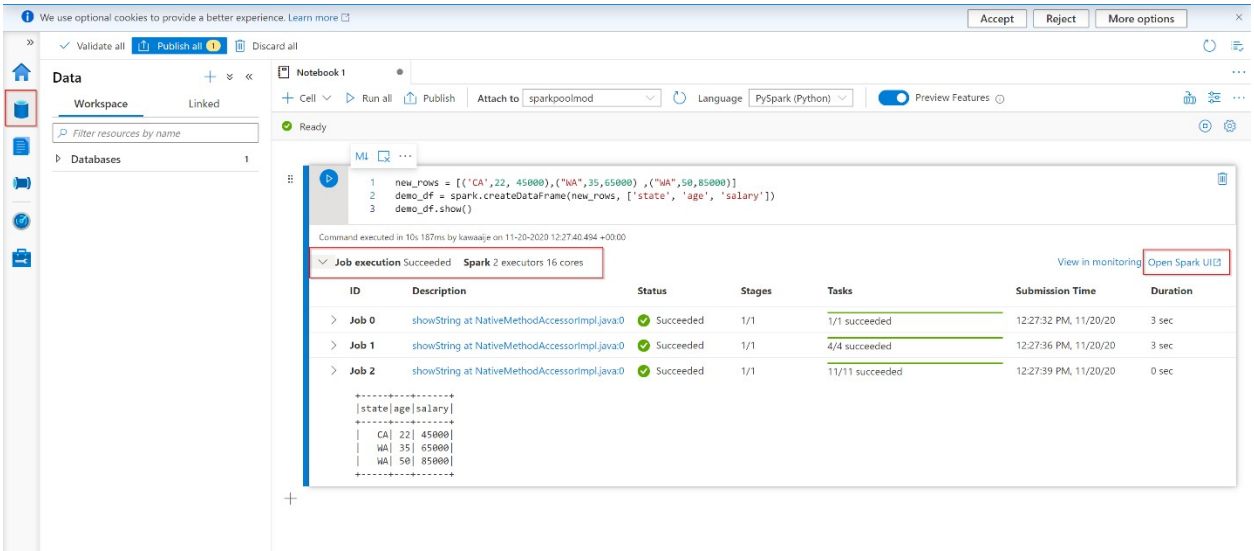


Monitor Apache Spark Application Overview

If you are familiar with Apache Spark, you can find the standard Apache Spark history server UI by selecting **Open Spark UI**.

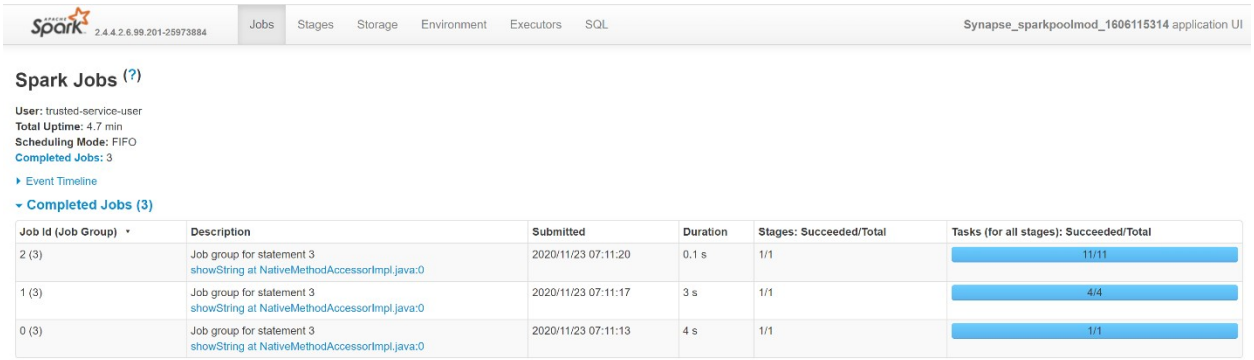
Another way to open the Apache Spark History server is to navigate to the **Data** tab. If you create a notebook and read a DataFrame then you can go to the bottom of the page and find the Spark History Server known as the Spark UI.

1. From your Azure Synapse Studio notebook, select **Open Spark UI** from the job execution output cell or from the status panel at the bottom of the notebook document.



Open Spark UI through Data Tab

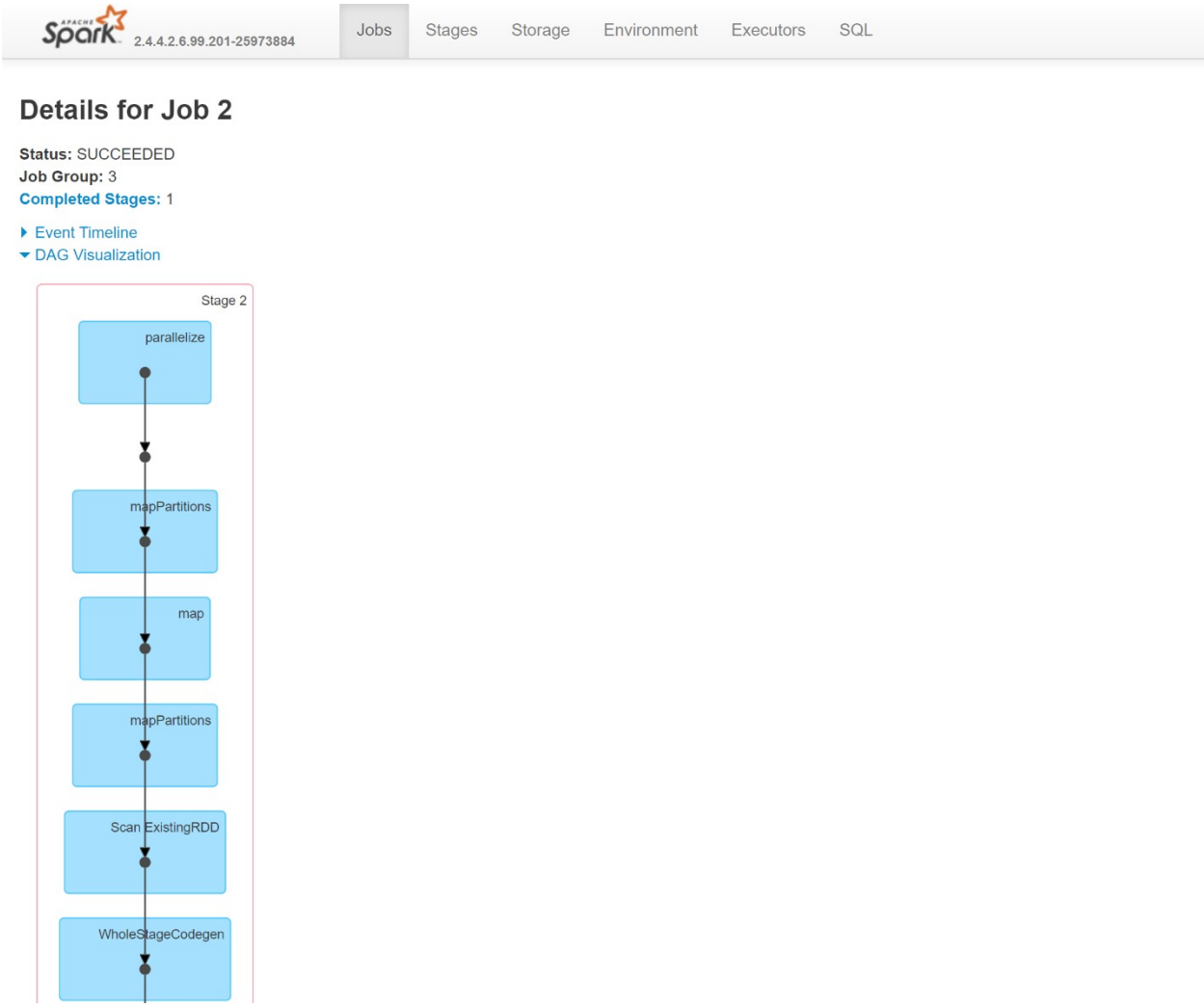
2. Select **Spark UI** from the slide out panel to be redirected to the Spark monitoring tab where you will land in the **Jobs** tab.



Open Spark UI
3. Within the Jobs tab of the Spark History server, you can view:

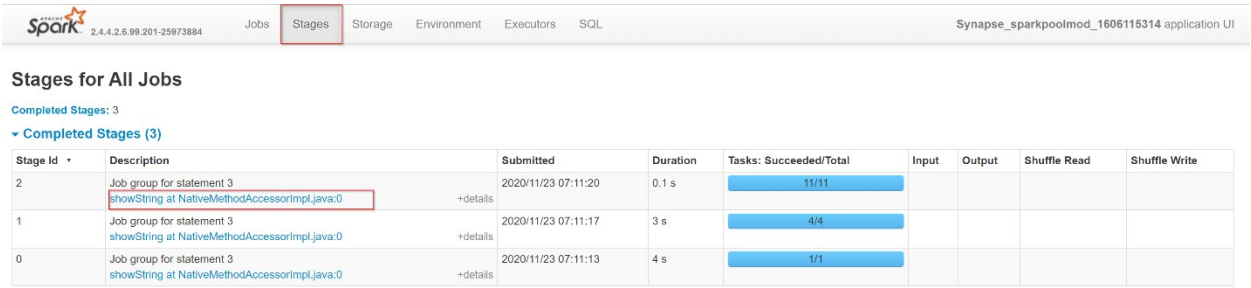
- Job ID
- Job description
- the time when the job was submitted
- the duration
- the stages
- and the task (for all stages).

If you select a job ID, go to the **description** and select the **url** to be redirected to the following screen:



DAG visualization
It will give you a DAG Visualization of the stage.

4. If you select the **Stages** tab, you'll find all the completed stages. If you want to dig deeper into the stages, you can select the **description url** as shown below:



Stage Tab in Spark UI

5. In the **Storage** tab, your Resilient Distributed Datasets (RDDs) will only be cached once you have evaluated it. A common way of forcing evaluation and populating a cache is, for example, to call the 'count' command: **`demo_df.cache() demo_df.count()`**

6. The image shows what you would see in the Storage tab of the Apache Spark UI:

Storage tab Cache

7. If you want to access more details, you can go into the RDD Name url where you can view the following details:

Spark 2.4.4.2.6.99.201-25973884 Jobs Stages Storage Environment Executors SQL Synapse_sparkpoolmod_1606115314 application UI

RDD Storage Info for Scan ExistingRDD[state#0,age#1L,salary#2L]

Storage Level: Memory Deserialized 1x Replicated
Cached Partitions: 16
Total Partitions: 16
Memory Size: 2.4 KB
Disk Size: 0.0 B

Data Distribution on 2 Executors

Host	On Heap Memory Usage	Off Heap Memory Usage	Disk Usage
bf5e6d37de2247e7847bed207a7b5e8f00337162244:36557	888.0 B (29.7 GB Remaining)	0.0 B (0.0 B Remaining)	0.0 B
bf5e6d37de2247e7847bed207a7b5e8f004f3689053:41025	1584.0 B (29.7 GB Remaining)	0.0 B (0.0 B Remaining)	0.0 B

16 Partitions

Block Name	Storage Level	Size in Memory	Size on Disk	Executors
rdd_22_0	Memory Deserialized 1x Replicated	24.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f00337162244:36557
rdd_22_1	Memory Deserialized 1x Replicated	24.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f004f3689053:41025
rdd_22_10	Memory Deserialized 1x Replicated	720.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f00337162244:36557
rdd_22_11	Memory Deserialized 1x Replicated	24.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f004f3689053:41025
rdd_22_12	Memory Deserialized 1x Replicated	24.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f00337162244:36557
rdd_22_13	Memory Deserialized 1x Replicated	24.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f004f3689053:41025
rdd_22_14	Memory Deserialized 1x Replicated	24.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f00337162244:36557
rdd_22_15	Memory Deserialized 1x Replicated	720.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f004f3689053:41025
rdd_22_2	Memory Deserialized 1x Replicated	24.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f00337162244:36557
rdd_22_3	Memory Deserialized 1x Replicated	24.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f004f3689053:41025
rdd_22_4	Memory Deserialized 1x Replicated	24.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f00337162244:36557
rdd_22_5	Memory Deserialized 1x Replicated	720.0 B	0.0 B	bf5e6d37de2247e7847bed207a7b5e8f004f3689053:41025

RDD Storage Details

8. If we move on to the **Environment** tab, you'll get the:

- Runtime Information
- the Spark Properties
- System properties
- and the class path entries.

9. The next thing we will look at is the **Executors** tab. Here you can find all the information about the executors:

Spark 2.4.4.2.6.99.201-25973884 Jobs Stages Storage Environment Executors SQL Synapse_sparkpoolmod_1606115314 application UI

Executors

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Blacklisted
Active(3)	16	97.7 KB / 95.6 GB	0.0 B	16	0	0	67	67	22 s (0.2 s)	0.0 B	2.7 KB	2.7 KB	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	0
Total(3)	16	97.7 KB / 95.6 GB	0.0 B	16	0	0	67	67	22 s (0.2 s)	0.0 B	2.7 KB	2.7 KB	0

Executors

Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
driver	bf5e6d37de2247e7847bed207a7b5e8f000a8796545:38719	Active	0	35.7 KB / 31.9 GB	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
1	bf5e6d37de2247e7847bed207a7b5e8f004f3689053:41025	Active	8	29.3 KB / 31.9 GB	0.0 B	8	0	0	33	33	12 s (0.1 s)	0.0 B	905 B	1.4 KB		Thread Dump
2	bf5e6d37de2247e7847bed207a7b5e8f00337162244:36557	Active	8	32.6 KB / 31.9 GB	0.0 B	8	0	0	34	34	10 s (61 ms)	0.0 B	1.8 KB	1.4 KB		Thread Dump

Showing 1 to 3 of 3 entries Previous 1 Next

Executors tab Spark UI

10. Finally, there is the **SQL** tab. Here you can find the completed queries and details within the IDs for the queries.

Spark 2.4.4.2.6.99.201-25973884 Jobs Stages Storage Environment Executors SQL Synapse_sparkpoolmod_1606115314 application UI

SQL

Completed Queries: 8

Completed Queries (8)

ID	Description	Submitted	Duration	Job IDs
-4	PeregrinePlanLogEvent +details Spark SQL Plans with annotations.	2020/11/23 07:31:45	21 ms	
3	count at NativeMethodAccessorImpl.java:0 +details org.apache.spark.sql.Dataset.count(Dataset.scala:2843) sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) java.lang.reflect.Method.invoke(Method.java:498) py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244) py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357) py4j.Gateway.invoke(Gateway.java:282) py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132) py4j.commands.CallCommand.execute(CallCommand.java:79) py4j.GatewayConnection.run(GatewayConnection.java:238) java.lang.Thread.run(Thread.java:748)	2020/11/23 07:31:44	0.6 s	[5]
-3	PeregrinePlanLogEvent +details	2020/11/23 07:30:58	12 ms	
2	count at NativeMethodAccessorImpl.java:0 +details	2020/11/23 07:30:58	0.3 s	[4]
-2	PeregrinePlanLogEvent +details	2020/11/23 07:28:43	26 ms	
1	count at NativeMethodAccessorImpl.java:0 +details	2020/11/23 07:28:42	0.6 s	[3]
-1	PeregrinePlanLogEvent +details	2020/11/23 07:11:20	0.1 s	
0	showString at NativeMethodAccessorImpl.java:0 +details	2020/11/23 07:11:12	7 s	[0][1][2]

SQL Tab Spark UI

Now we have been through all the different tabs using the Spark UI, enabling you to optimize and baseline through the Apache Spark performance and settings.