

Install Apache Spark 3.5.1 with Python API & Integrate with Jupyter Notebook on Windows 10

Author: Sri Adilakshmi M

Table of Contents

1. Overview:	4
2. Prerequisites:	5
3. Install Standalone Spark:	6
3.1. Download Spark Binaries:	6
3.2. Setup Environment Variables:	9
3.3. Verify Spark Installation:	14
3.4. Start Spark Services:	15
3.4.1. Start Spark Master:	15
3.4.2. Start Spark Worker:	16
3.5. Spark Examples:	19
3.5.1. Run Java SparkPI Program:	19
3.5.2. Run Python wordcount Program:	27
4. Install Spark on Hadoop:	31
4.1. Verify Hadoop Installation:	31
4.2. Download Spark Binaries:	31
4.3. Set up Environment Variables:	34
4.4. Configure Spark Environment:	38
4.5. Verify Spark Installation:	39
4.6. Start Hadoop Services:	40
4.6.1. Start Hadoop Nodes:	40
4.6.2. Start Hadoop YARN:	42
4.7. Set Spark YARN Directory:	43
4.8. Spark Examples:	45
4.8.1. Run Java SparkPI Program:	45
4.8.2. Run Python wordcount Program:	52
5. Spark CLIs:	58
5.1. Pyspark:	58
5.2. Spark Shell:	59
5.3. Spark SQL:	60

6.	Spark Web UI:	62
6.1.	Master Web UI:	62
6.2.	Worker Web UI:	63
6.3.	Application Web UI:	64
7.	Spark History Server:	65
8.	Write Sample PySpark Code:	69
8.1.	Read/Write Locally:	69
8.2.	Read/Write HDFS:	71
9.	Spark SQL:	75
10.	Spark Beeline:	80
10.1.1.	Start Spark Master:	80
10.1.2.	Start Spark Worker:	81
10.1.3.	Start Spark Thrift Server:	82
10.1.4.	Start Beeline:	83
11.	Spark with Jupiter Notebook:	86

This document outlines the steps needed to install two modes – **Standalone cluster** and **Hadoop cluster** - of **Apache Spark 3.x** with **Python** application programming interface on Windows operating system and integrate with **Jupyter Notebook**.

1. Overview:

Apache Spark is a powerful open-source analytical and computing engine to process large scale data loads. It is heavily used in Data Engineering, Data Science and Machine learning projects since it offers easy programming interface in various languages such as **Python**, **SQL**, **Java**, **Scala** and **R**.

Spark provides heavy processing tools such as **Spark SQL** for structured data processing, **MLib** for machine learning, **Graphx** for graph processing, **Structured Streaming** for real time data processing and **pandas** API to handle pandas workloads.

The important features of Spark are:

- It unifies data processing with batch and or real-time streaming.
- Provides advanced data analytics even with ANSI SQL offering fast, distributed queries for dashboarding and reporting.
- Powerful engine that runs 100 times faster than the traditional Hadoop MapReduce.
- Spark can train machine learning algorithms on a single computer and deploy it on cluster with thousands of machines offering parallel and distributed computation.

The key components of Spark include **Spark Driver**, **Spark Executor**, **Cluster Manager** and **Spark History Server** (Web UI). Spark also offers various Command Line Interfaces such as `spark-shell` (for Scala), `pyspark` (for Python), `sparkr` (for R) and `spark-sql` (for SQL) to execute the Spark code in different programming languages.

Spark supports 4 cluster managers:

1. **Standalone Cluster:** It is the simple cluster manager where Spark manages its cluster without any external dependencies. It is easy to setup and suitable for small-scale deployments.
2. **Apache Mesos:** It is a generic cluster manager that provides efficient resource allocation and scheduling, supporting dynamic resource sharing between applications.
Note: This cluster manager support has been deprecated from Spark 3.2.0 version.
3. **Hadoop YARN:** A resource manager integrated with the Hadoop ecosystem. It provides fine grained resource allocation and scheduling, supporting multiple concurrent applications on the same cluster.

4. **Kubernetes:** An open source system for automating deployment, scaling and management of containerized applications.

The Spark applications can be run in three different modes:

- **Local Mode** – In local mode, Spark runs all processes within a single JVM of the machine. It is a simple mode of deployment which is mostly used for testing purposes. This mode is best suited for users who are just learning spark or testing small-scale data processing workflows.
- **Client Mode** – This is the **default** deployment mode in which the Spark driver component runs on the client machine from where the job is submitted and connects to cluster manager for resource allocation. With this mode, if the machine or user session running the spark application terminates, the entire Spark application gets terminated with status fail. This mode is not suitable for Production deployments.
- **Cluster Mode** – In this mode, the Spark driver component gets launched on any of the available nodes in the Spark cluster, instead of launching on the client machine from where the job is submitted. Here, even if the local machine or user session is terminated, the spark application keeps running on the cluster. This mode is mostly used for processing large data sets and best suited for Production deployments.

Note:

Apache Spark downloads are pre-packaged with Hadoop libraries for users to install standalone Spark cluster. Spark also has “Hadoop free” download that allows to install Spark on top of existing Hadoop cluster.

2. Prerequisites:

The following prerequisites need to be installed before installing Spark.

1. **File Archiver:** Any file archiver such as **7zip** or **WinRAR** is needed to unzip the downloaded Spark binaries. 7zip can be downloaded from the [7zip Downloads](#) website and WinRAR can be downloaded from the [RAR lab Downloads](#) website.
2. **JRE 8:** Spark 3.x requires Java 8 runtime environment.
We can either download just JRE 8 (Java Runtime Environment) for Windows offline installation from the official [Java Download for Windows Offline](#) website or download the whole JDK 8 (Java Development Kit) directly from [Oracle Java Downloads](#) website. For the complete JDK installation steps, look at [here](#).
3. **Python:** To run Apache Spark with Python API, we should have Python software running in the system. If Python is not already available, you can install it from [Python](#)

[Downloads](#) website but it is recommended to install [Anaconda Distribution](#) which is an open source software built for **Python and R** programming languages that are heavily used in Data Science, Data Engineer and Data Analytics fields. Refer to [Official Anaconda Installation Guide](#) on how to install it.

3. Install Standalone Spark:

In this section, we will see how to setup Standalone Spark cluster with one master server and one worker node running locally.

3.1. Download Spark Binaries:

Download the latest **Spark 3.5.1** version from the [Apache Spark Downloads](#) page. In this page, choose the **package type** as **Pre-built for Apache Hadoop 3.3** and click on `spark-3.5.1-bin-hadoop3.tgz` binary file.

The screenshot shows the Apache Spark Downloads page. The URL is <https://spark.apache.org/downloads.html>. The main content area has four numbered steps for downloading:

- Choose a Spark release: 3.5.1 (Feb 23 2024)
- Choose a package type: Pre-built for Apache Hadoop 3.3 and later
- Download Spark: spark-3.5.1-bin-hadoop3.tgz
- Verify this release using the 3.5.1 signatures, checksums and project release KEYS by following these [procedures](#).

Below these steps, a note states: "Note that Spark 3 is pre-built with Scala 2.12 in general and Spark 3.2+ provides additional pre-built distribution with Scala 2.13."

On the right side, there is a "Latest News" sidebar with links to recent releases:

- Spark 3.4.3 released (Apr 18, 2024)
- Spark 3.5.1 released (Feb 23, 2024)
- Spark 3.3.4 released (Dec 16, 2023)
- Spark 3.4.2 released (Nov 30, 2023)

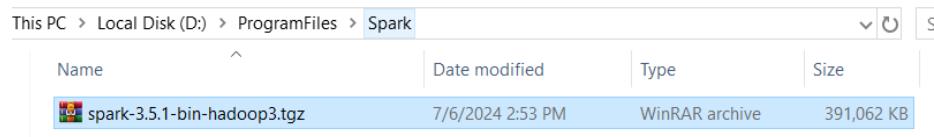
At the bottom right, there is a "COMMUNITY CODE" button and a "DOWNLOAD SPARK" button.

You will be navigated to [spark-3.5.1](#) mirror website where click on the suggested location for [spark-3.5.1-bin-hadoop3.tgz](#) file that gets downloaded to your **Downloads** folder in your machine.

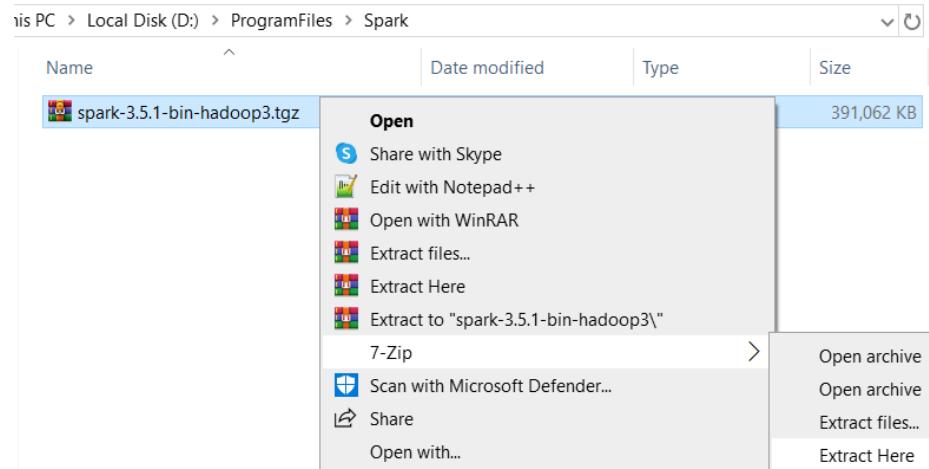
The screenshot shows a web browser displaying the Apache Software Foundation's download page for Apache Spark 3.5.1. The URL in the address bar is <https://www.apache.org/dyn/closer.lua/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz>. The page features the Apache logo and navigation links for Community, Projects, Downloads, Learn, Resources & Tools, and About. A banner at the top says "Sponsor the ASF". Below the banner, it says "We suggest the following location for your download:" followed by a link to <https://dlcdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz>. It also lists alternate download locations and instructions for verifying file integrity using PGP signatures or hashes.

After the binary file is downloaded, unpack it using any file archiver (**7zip** or **WinRAR**) utility as below:

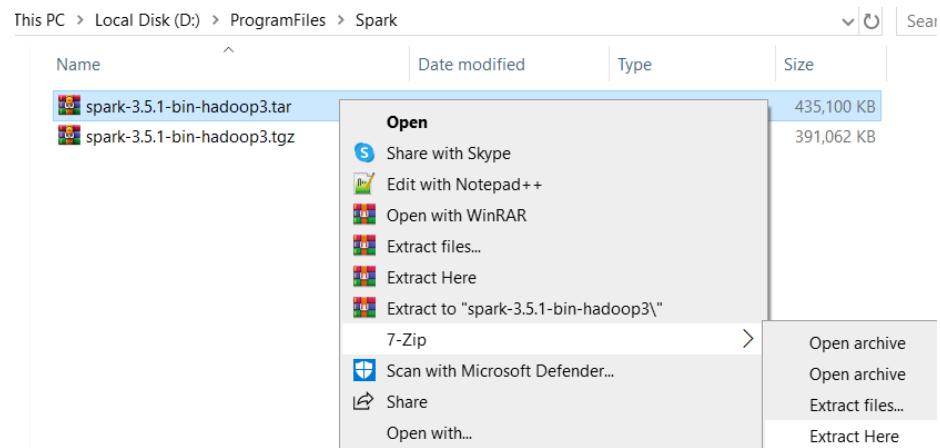
- Choose the installation directory in your machine and copy `spark-3.5.1-bin-hadoop3.tgz` file to that directory. Here, we are choosing Spark installation directory as `D:\ProgramFiles\Spark`.



- Right click on `spark-3.5.1-bin-hadoop3.tgz` and choose **7-Zip -> Extract Here** option which extracts a new packed file `spark-3.5.1-bin-hadoop3.tar`.



- Next, unpack `spark-3.5.1-bin-hadoop3.tar` file using 7zip utility.



- The tar file extraction may take few minutes to finish. After finishing, you see a folder named `spark-3.5.1-bin-hadoop3` which consists of Spark binaries and libraries.

Name	Date modified	Type	Size
bin	2/15/2024 5:06 PM	File folder	
conf	2/15/2024 5:06 PM	File folder	
data	2/15/2024 5:06 PM	File folder	
examples	2/15/2024 5:06 PM	File folder	
jars	2/15/2024 5:06 PM	File folder	
kubernetes	2/15/2024 5:06 PM	File folder	
licenses	2/15/2024 5:06 PM	File folder	
python	2/15/2024 5:06 PM	File folder	
R	2/15/2024 5:06 PM	File folder	
sbin	2/15/2024 5:06 PM	File folder	
yarn	2/15/2024 5:06 PM	File folder	
LICENSE	2/15/2024 5:06 PM	File	23 KB
NOTICE	2/15/2024 5:06 PM	File	57 KB
README.md	2/15/2024 5:06 PM	MD File	5 KB
RELEASE	2/15/2024 5:06 PM	File	1 KB

Note:

Although Spark has provided Hadoop libraries, they are not compatible to run on Windows systems by default. So, we need to get Hadoop's native IO utilities for Windows from [cdarlint GitHub repository](#) which are available for different versions of Hadoop.

- Since we installed the latest Spark 3.5.1 version, download utilities such as `winutils.exe` and `hadoop.dll` of the latest Hadoop 3.3.6 version from [here](#).
- Create a folder named `hadoop-winutils` in **D drive** and create a sub-folder named `bin` inside it.
- Move `winutils.exe` and `hadoop.dll` files from your downloaded location to `D:\hadoop-winutils\bin` directory.

This PC > Local Disk (D:) > hadoop-winutils > bin			
Name	Date modified	Type	Size
winutils.exe	7/6/2024 2:57 PM	Application	117 KB
hadoop.dll	7/6/2024 2:57 PM	Application extens...	77 KB

3.2. Setup Environment Variables:

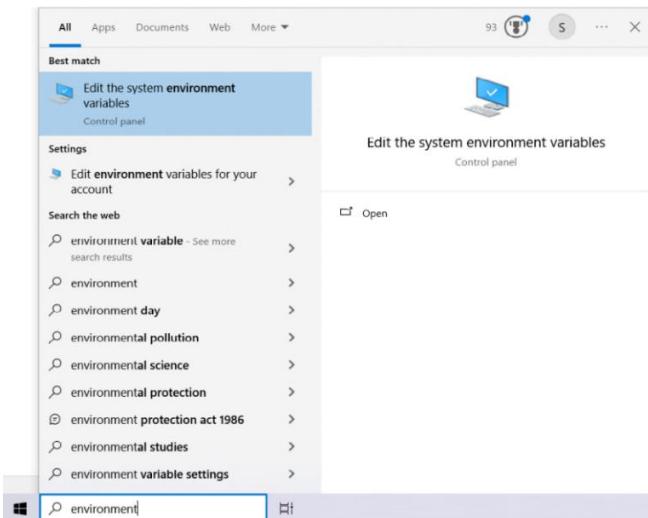
After installing pre-requisites, Spark binaries and Hadoop utilities for Windows, we should configure the below environment variables defining Java, Hadoop and Spark default paths.

- **JAVA_HOME:** This is the JDK installation directory path in the machine (*in my machine, it is D:\ProgramFiles\Java\jdk-1.8*). Ignore it if this is already done.
- **HADOOP_HOME:** This is the Hadoop's WinUtils path in the machine (*in our case, it is D:\hadoop-winutils*)
- **SPARK_HOME:** This is the Spark installation directory path in the machine (*in our case, it is D:\ProgramFiles\Spark\spark-3.5.1-bin-hadoop3*)
- **PYTHONPATH:** This is the Spark python directory path which should be `%SPARK_HOME%\python`
- **PYSPARK_PYTHON:** Set this variable to the location where `python.exe` file is available (*in my machine, it is D:\ProgramFiles\Anaconda\anaconda3\python*) which is needed for Spark to execute the Python code.

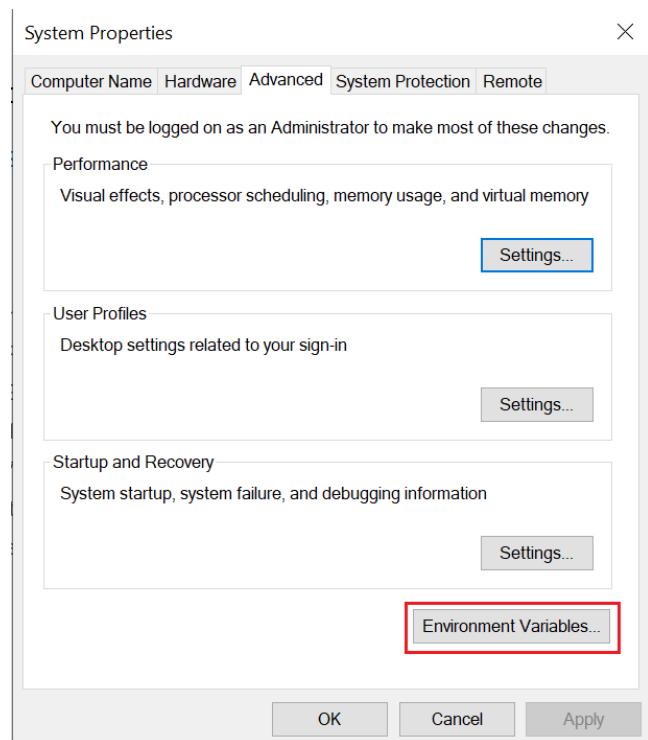
These variables need to be added to either **User environment variables** or **System environment variables** depending on Spark configuration needed **for a single user** or **for multiple users**.

In this tutorial, we will add User environment variables since we are configuring Spark for a single user. If you would like to configure Spark for multiple users, then define System environment variables.

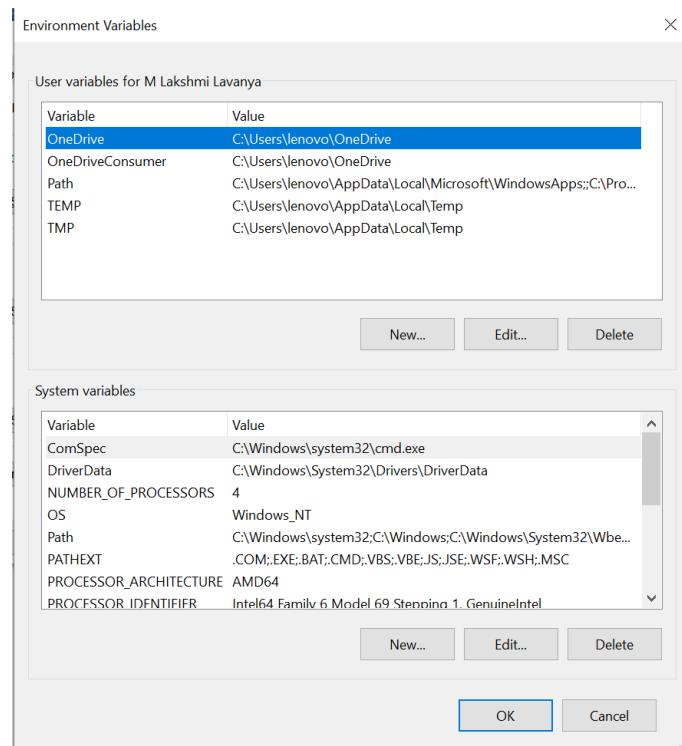
In the Windows search bar, start typing “environment variables” and select the first match which opens up **System Properties** dialog.



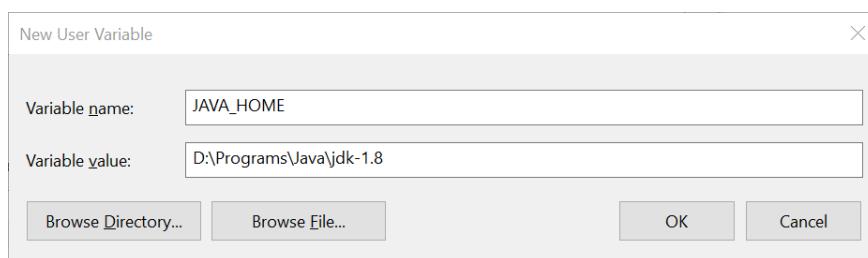
On the **System Properties** window, press **Environment Variables** button.



In the **Environment Variables** dialog, click on **New** under **User variables** section.



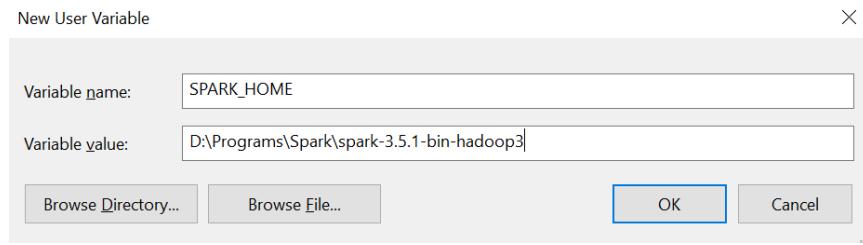
Add JAVA_HOME variable and press OK.



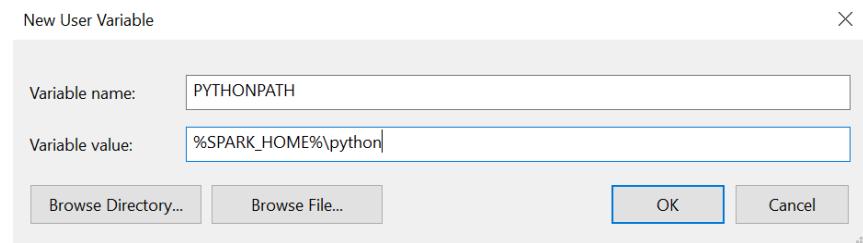
Click on **New** again and add HADOOP_HOME variable and press OK.



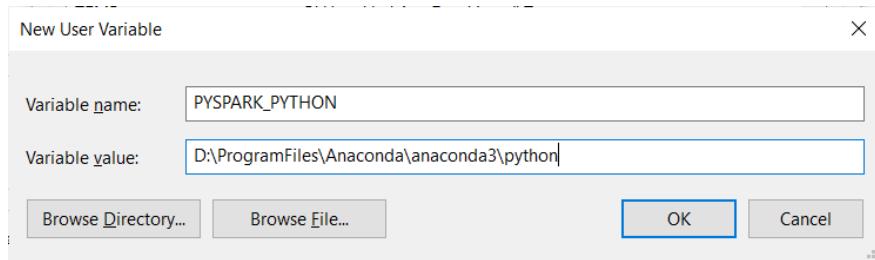
Click on **New** again and add **SPARK_HOME** variable and press OK.



Click on **New** again and add **PYTHONPATH** variable and press OK.

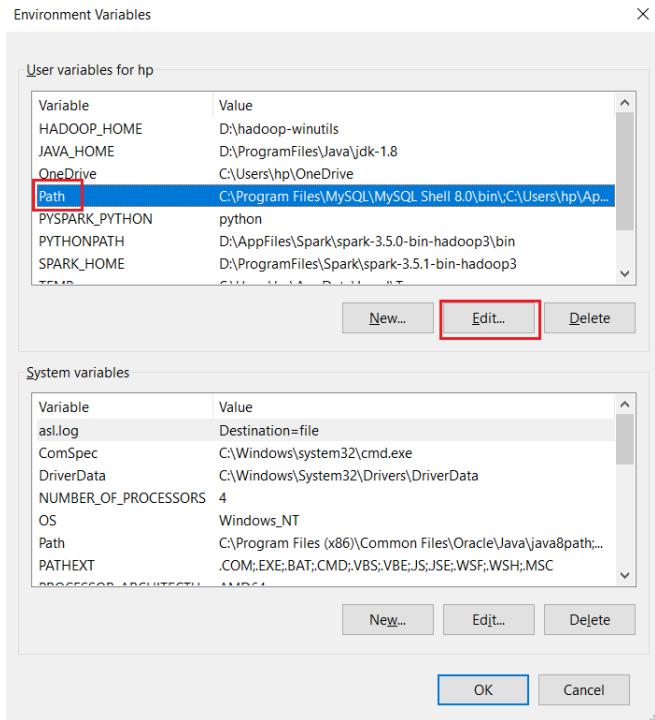


Click on **New** again and add **PYSPARK_PYTHON** variable and press OK.



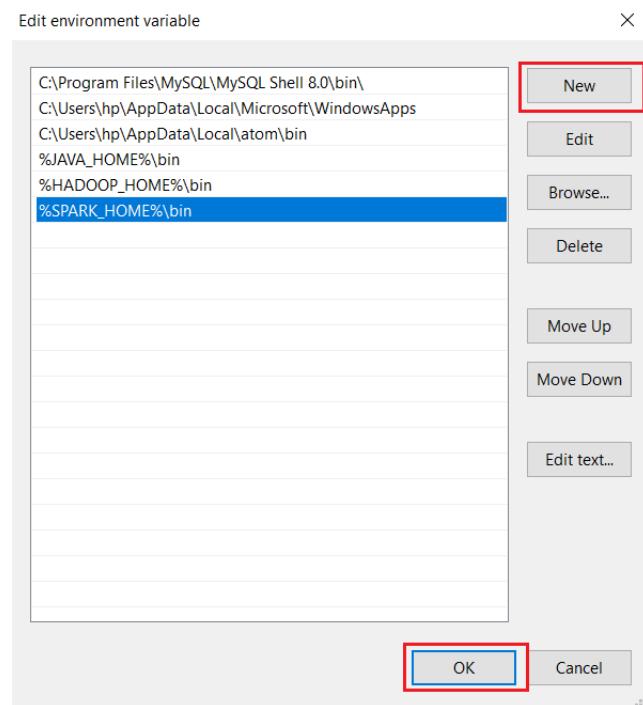
Now, we will update **PATH** variable to add Java, Hadoop and Spark binary paths.

Select **PATH** variable under **User Variables** and press **Edit** button.

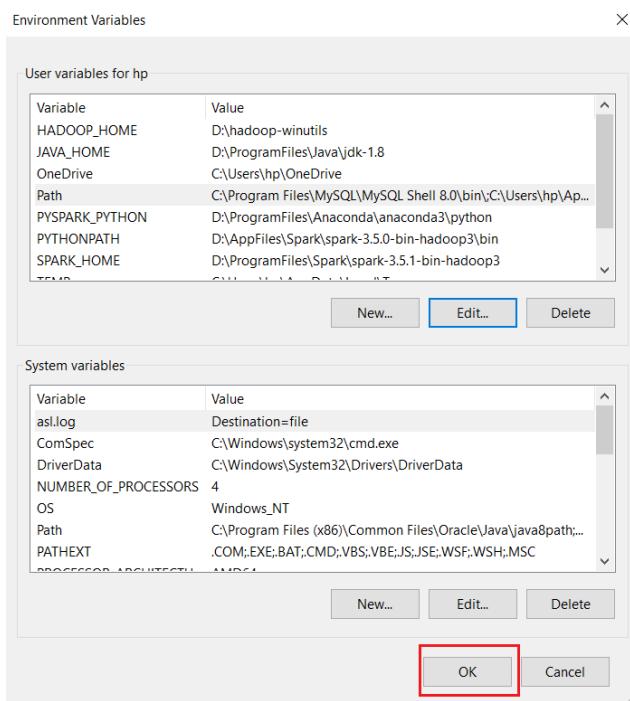


Press **New** and add the following values and press **OK**.

```
%JAVA_HOME%\bin
%HADOOP_HOME%\bin
%SPARK_HOME%\bin
```



Press OK again to apply environment variable changes and close window.



3.3. Verify Spark Installation:

Open either **Command Prompt** or **Windows PowerShell** prompt and run the following command:

```
pyspark --version
```

A screenshot of a Command Prompt window. The window title is 'Command Prompt'. The text inside the window shows the output of the 'pyspark --version' command. The output includes the Scala version (2.12.18), Java HotSpot(TM) 64-Bit Server VM version (1.8.0_411), and the Spark version (3.5.1). The word 'version 3.5.1' is highlighted with a red box.

After the above command is executed, you can see the installed **Spark version 3.5.1**.

3.4. Start Spark Services:

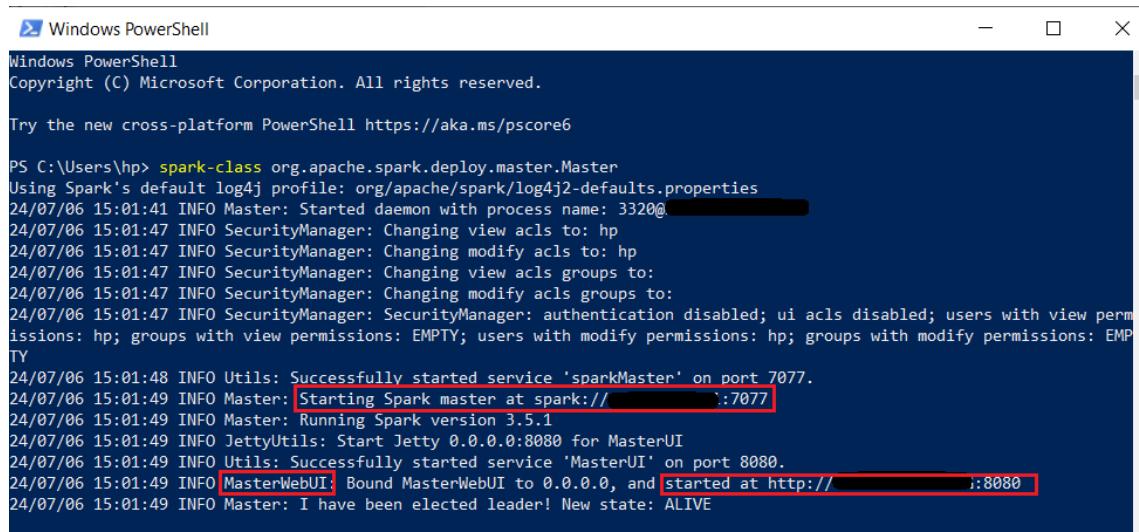
Now, we will start the Spark Master and Worker nodes in the local system.

3.4.1. Start Spark Master:

The master node is responsible for coordinating the work among the worker nodes in the Spark cluster.

Open a new **Command Prompt** or **Windows PowerShell** and run the following command to start the Spark Master service:

```
spark-class org.apache.spark.deploy.master.Master
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\hp> spark-class org.apache.spark.deploy.master.Master
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
24/07/06 15:01:41 INFO Master: Started daemon with process name: 3320@[REDACTED]
24/07/06 15:01:47 INFO SecurityManager: Changing view acls to: hp
24/07/06 15:01:47 INFO SecurityManager: Changing modify acls to: hp
24/07/06 15:01:47 INFO SecurityManager: Changing view acls groups to:
24/07/06 15:01:47 INFO SecurityManager: Changing modify acls groups to:
24/07/06 15:01:47 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: hp; groups with view permissions: EMPTY; users with modify permissions: hp; groups with modify permissions: EMPTY
24/07/06 15:01:48 INFO Utils: Successfully started service 'sparkMaster' on port 7077.
24/07/06 15:01:49 INFO Master: Starting Spark master at spark://[REDACTED]:7077
24/07/06 15:01:49 INFO Master: Running Spark version 3.5.1
24/07/06 15:01:49 INFO JettyUtils: Start Jetty 0.0.0.0:8080 for MasterUI
24/07/06 15:01:49 INFO Utils: Successfully started service 'MasterUI' on port 8080.
24/07/06 15:01:49 INFO MasterWebUI: Bound MasterWebUI to 0.0.0.0, and [REDACTED] started at http://[REDACTED]:8080
24/07/06 15:01:49 INFO Master: I have been elected leader! New state: ALIVE
```

Here, we can see that Spark master is started at <spark://<ipaddress>:7077> i.e listening on default port 7077 and **MasterWebUI** is started at <http://<ipaddress>:8080> by default.

Note that we can change the Master node to listen, port to listen and web UI port by specifying additional options while starting Master server.

Use the following command to get the additional options for starting master server

```
spark-class org.apache.spark.deploy.master.Master --help
```

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\hp> spark-class org.apache.spark.deploy.master.Master --help
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
24/07/06 15:04:24 INFO Master: Started daemon with process name: 14160@[REDACTED]
Usage: Master [options]

Options:
  -i HOST, --ip HOST      Hostname to listen on (deprecated, please use --host or -h)
  -h HOST, --host HOST    Hostname to listen on
  -p PORT, --port PORT    Port to listen on (default: 7077)
  --webui-port PORT       Port for web UI (default: 8080)
  --properties-file FILE   Path to a custom Spark properties file.
                           Default is conf/spark-defaults.conf.

PS C:\Users\hp>

```

Check that the master node is running correctly by accessing the Spark web UI at <http://localhost:8080/>

Worker ID	Address	State	Cores	Memory	Resources
Workers (0)					

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
Running Applications (0)								

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
Completed Applications (0)								

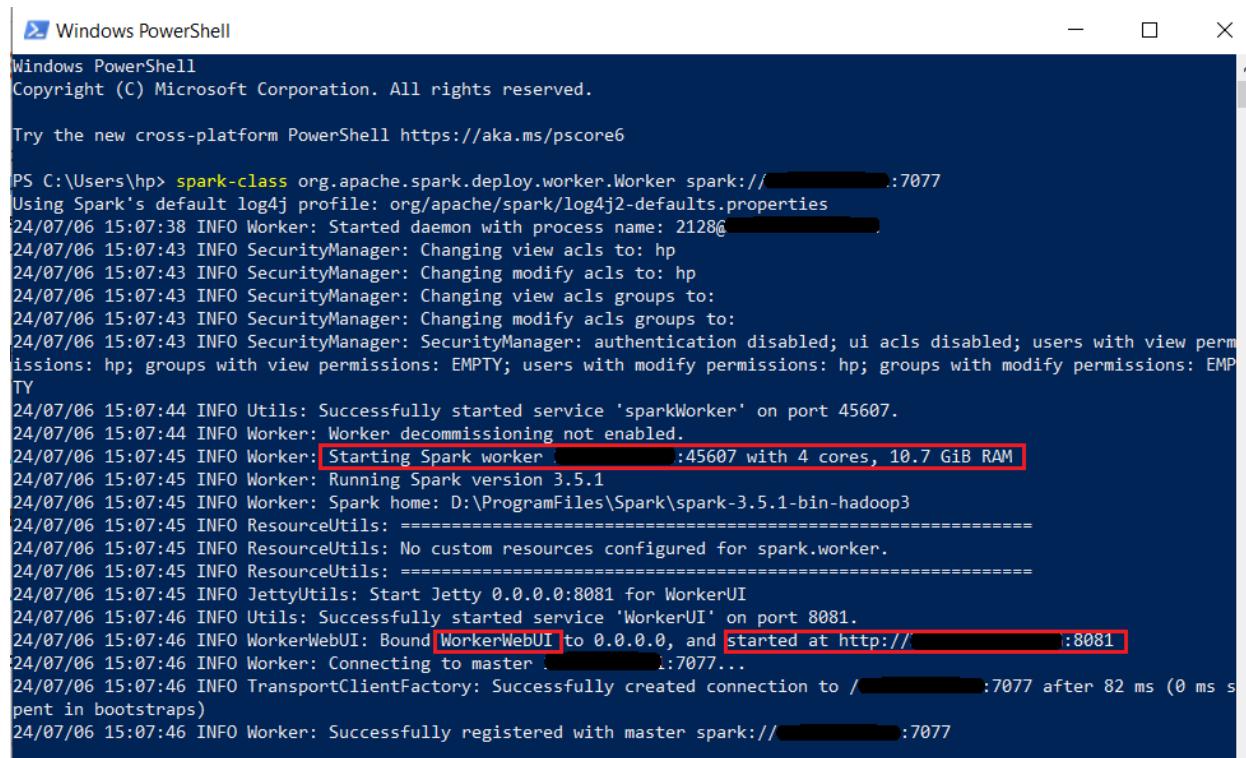
Here, we can see the Spark Master service is running at URL `spark://<ipaddress>:7077` and worker nodes as 0 since we have not started Spark worker service yet.

3.4.2. Start Spark Worker:

The worker nodes are responsible for executing the tasks assigned by the Spark master. We need to start the worker node by specifying the Spark master URL.

Open a new **Command Prompt** or **Windows PowerShell** and run the following command to start the Spark Worker service (*Make sure that you replace <ipaddress> with IP address mentioned in your Spark Master URL*)

```
spark-class org.apache.spark.deploy.worker.Worker  
spark://<ipaddress>:7077
```



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows command-line output. The command run was "spark-class org.apache.spark.deploy.worker.Worker spark://<ipaddress>:7077". The log output includes several INFO messages from the Spark SecurityManager and ResourceUtils, and a message indicating the worker is starting with 4 cores and 10.7 GiB RAM. It also shows the Worker connecting to a master and starting a WorkerWebUI at port 8081.

```
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Try the new cross-platform PowerShell https://aka.ms/pscore6  
  
PS C:\Users\hp> spark-class org.apache.spark.deploy.worker.Worker spark://<ipaddress>:7077  
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties  
24/07/06 15:07:38 INFO Worker: Started daemon with process name: 2128@  
24/07/06 15:07:43 INFO SecurityManager: Changing view acls to: hp  
24/07/06 15:07:43 INFO SecurityManager: Changing modify acls to: hp  
24/07/06 15:07:43 INFO SecurityManager: Changing view acls groups to:  
24/07/06 15:07:43 INFO SecurityManager: Changing modify acls groups to:  
24/07/06 15:07:43 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: hp; groups with view permissions: EMPTY; users with modify permissions: hp; groups with modify permissions: EMPTY  
24/07/06 15:07:44 INFO Utils: Successfully started service 'sparkWorker' on port 45607.  
24/07/06 15:07:44 INFO Worker: Worker decommissioning not enabled.  
24/07/06 15:07:45 INFO Worker: Starting Spark worker [REDACTED]:45607 with 4 cores, 10.7 GiB RAM  
24/07/06 15:07:45 INFO Worker: Running Spark version 3.5.1  
24/07/06 15:07:45 INFO Worker: Spark home: D:\ProgramFiles\Spark\spark-3.5.1-bin-hadoop3  
24/07/06 15:07:45 INFO ResourceUtils: =====  
24/07/06 15:07:45 INFO ResourceUtils: No custom resources configured for spark.worker.  
24/07/06 15:07:45 INFO ResourceUtils: =====  
24/07/06 15:07:45 INFO JettyUtils: Start Jetty 0.0.0.0:8081 for WorkerUI  
24/07/06 15:07:46 INFO Utils: Successfully started service 'WorkerUI' on port 8081.  
24/07/06 15:07:46 INFO WorkerWebUI: Bound WorkerWebUI to 0.0.0.0, and started at http://[REDACTED]:8081  
24/07/06 15:07:46 INFO Worker: Connecting to master [REDACTED]:7077...  
24/07/06 15:07:46 INFO TransportClientFactory: Successfully created connection to /[REDACTED]:7077 after 82 ms (0 ms spent in bootstraps)  
24/07/06 15:07:46 INFO Worker: Successfully registered with master spark://[REDACTED]:7077
```

Here, we can see that Spark worker is started at `<ipaddress>:45607` (on a random port) with available number of cores and RAM running on the machine. **WorkerWebUI** is started at <http://<ipaddress>:8081> by default.

Note that we can specify the number of cores to use and memory to use for Worker node, also change the Worker node to listen, port to listen and web UI port by specifying additional options while starting Worker server.

Use the following command to get the additional options for starting worker server

```
spark-class org.apache.spark.deploy.worker.Worker --help
```

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\hp> spark-class org.apache.spark.deploy.worker.Worker --help
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
24/07/06 15:09:58 INFO Worker: Started daemon with process name: 1272@[REDACTED]
Usage: Worker [options] <master>

Master must be a URL of the form spark://hostname:port

Options:
  -c CORES, --cores CORES  Number of cores to use
  -m MEM, --memory MEM    Amount of memory to use (e.g. 1000M, 2G)
  -d DIR, --work-dir DIR  Directory to run apps in (default: SPARK_HOME/work)
  -i HOST, --ip IP        Hostname to listen on (deprecated, please use --host or -h)
  -h HOST, --host HOST   Hostname to listen on
  -p PORT, --port PORT   Port to listen on (default: random)
  --webui-port PORT      Port for web UI (default: 8081)
  --properties-file FILE  Path to a custom Spark properties file.
                          Default is conf/spark-defaults.conf.

PS C:\Users\hp>

```

For example, to start the Worker node with 2 cores and 8GB RAM, use this command:

```
spark-class org.apache.spark.deploy.worker.Worker -c 2 -m 8G
spark://<ipaddress>:7077
```

Check that the worker node is connected to the Spark master correctly by accessing the Spark Worker web UI at <http://localhost:8081/>

The screenshot shows a web browser window with the address bar set to `http://localhost:8081`. The main content area displays the "Spark Worker" interface. At the top, it shows the Spark logo and the text "Spark Worker at [REDACTED]:45607". Below this, there is a summary of resources: ID: worker-20240706150744-[REDACTED]-45607, Master URL: spark://[REDACTED]/077, Cores: 4 (0 Used), Memory: 10.7 GiB (0.0 B Used), and Resources: [REDACTED]. A red box highlights the "Back to Master" link. Below this, a section titled "Running Executors (0)" is shown, with a table header row containing columns for ExecutorID, State, Cores, Memory, Resources, Job Details, and Logs.

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
------------	-------	-------	--------	-----------	-------------	------

Click on **Back to Master** link to open the Master UI where we can see one Alive Worker is available.

The screenshot shows the Apache Spark master interface at <http://desktop-kgh2e2g:8080>. The top navigation bar indicates it's not secure. The main title is "Spark Master at spark://[REDACTED]:7077". Below the title, it says "Alive Workers: 1". The "Workers" section lists one worker: "worker-20240706150744-[REDACTED] 45607" with an address of "[REDACTED]45607", state "ALIVE", cores "4 (0 Used)", and memory "10.7 GiB (0.0 B Used)". There are sections for "Running Applications (0)" and "Completed Applications (0)" both showing 0 entries.

3.5. Spark Examples:

Spark provides some in-built example programs such as `pi`, `wordcount`, `sort`, etc. available under `SPARK_HOME\examples` directory that can be execute on Spark cluster.

3.5.1. Run Java SparkPi Program:

Let us execute the example Java program `SparkPi` packaged under `spark-examples*.jar` file that is located in `SPARK_HOME\examples\jars` directory.

In Local Mode:

Open a new Command Prompt and run the below command which `SparkPi` application in local mode with all available cores:

```
spark-submit --class org.apache.spark.examples.SparkPi  
%SPARK_HOME%\examples\jars\spark-examples*.jar  
or  
spark-submit --master local[*] --class  
org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
```

```
Command Prompt
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-submit --master local[*] --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
24/07/06 15:15:52 INFO SparkContext: Running Spark version 3.5.1
24/07/06 15:15:52 INFO SparkContext: OS info Windows 10, 10.0, amd64
24/07/06 15:15:52 INFO SparkContext: Java version 1.8.0_411
24/07/06 15:15:52 INFO ResourceUtils: =====
24/07/06 15:15:52 INFO ResourceUtils: No custom resources configured for spark.driver.
24/07/06 15:15:52 INFO ResourceUtils: =====
24/07/06 15:15:52 INFO SparkContext: Submitted application: Spark Pi
24/07/06 15:15:52 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpu -> name: cpus, amount: 1.0)
24/07/06 15:15:52 INFO ResourceProfile: Limiting resource is cpu
24/07/06 15:15:52 INFO ResourceProfileManager: Added ResourceProfile id: 0
24/07/06 15:15:53 INFO SecurityManager: Changing view acls to: hp
24/07/06 15:15:53 INFO SecurityManager: Changing modify acls to: hp
24/07/06 15:15:53 INFO SecurityManager: Changing view acls groups to:
24/07/06 15:15:53 INFO SecurityManager: Changing modify acls groups to:
24/07/06 15:15:53 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: hp; groups with view permissions: EMPTY; users with modify permissions: hp; groups with modify permissions: EMPTY
24/07/06 15:15:54 INFO Utils: Successfully started service 'sparkDriver' on port 45722.
24/07/06 15:15:54 INFO SparkEnv: Registering MapOutputTracker
24/07/06 15:15:54 INFO SparkEnv: Registering BlockManagerMaster
24/07/06 15:15:54 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
24/07/06 15:15:54 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
24/07/06 15:15:54 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/07/06 15:15:54 INFO DiskBlockManager: Created local directory at C:\Users\hp\AppData\Local\Temp\blockmgr-f14f8c6c-9bb0-4c31-b4f9-58794331130a
24/07/06 15:15:54 INFO MemoryStore: MemoryStore started with capacity 366.3 MiB
24/07/06 15:15:54 INFO SparkEnv: Registering OutputCommitCoordinator
24/07/06 15:15:54 INFO JettyUtils: Start Jetty 0.0.0.0:4040 for SparkUI
24/07/06 15:15:54 INFO Utils: Successfully started service 'SparkUI' on port 4040.
24/07/06 15:15:54 INFO SparkContext: Added JAR file:/D:/ProgramFiles/Spark/spark-3.5.1-bin-hadoop3/examples/jars/spark-examples_2.12-3.5.1.jar at spark://--:45722/jars/spark-examples_2.12-3.5.1.jar with timestamp 1720259152522
24/07/06 15:15:55 INFO Executor: Starting executor ID driver on host
24/07/06 15:15:55 INFO Executor: OS info Windows 10, 10.0, amd64
24/07/06 15:15:55 INFO Executor: Java version 1.8.0_411
24/07/06 15:15:55 INFO Executor: Starting executor with user classpath (userClassPathFirst = false): ''
24/07/06 15:15:55 INFO Executor: Created or updated repl class loader org.apache.spark.util.MutableURLClassLoader@33b082c5 for default.
24/07/06 15:15:55 INFO Executor: Fetching spark://--:45722/jars/spark-examples_2.12-3.5.1.jar with timestamp 1720259152522
24/07/06 15:15:55 INFO TransportClientFactory: Successfully created connection to :/0.0.0.0:45722 after 74 ms (0 ms spent in bootstraps)
```

```

Command Prompt
)
24/07/06 15:15:57 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 2.3 KiB, free 366
.3 MiB)
24/07/06 15:15:57 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on [REDACTED]:45750 (size: 2.3 KiB, fre
e: 366.3 MiB)
24/07/06 15:15:57 INFO SparkContext: Created broadcast 0 from broadcast at DAGScheduler.scala:1585
24/07/06 15:15:57 INFO DAGScheduler: Submitting 2 missing tasks from ResultStage 0 (MapPartitionsRDD[1] at map at SparkP
i.scala:34) (first 15 tasks are for partitions Vector(0, 1))
24/07/06 15:15:57 INFO TaskSchedulerImpl: Adding task set 0.0 with 2 tasks resource profile 0
24/07/06 15:15:57 INFO TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0) ([REDACTED], executor driver, partiti
on 0, PROCESS_LOCAL, 7931 bytes)
24/07/06 15:15:57 INFO TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1) ([REDACTED], executor driver, partiti
on 1, PROCESS_LOCAL, 7931 bytes)
24/07/06 15:15:57 INFO Executor: Running task 0.0 in stage 0.0 (TID 1)
24/07/06 15:15:57 INFO Executor: Running task 0.0 in stage 0.0 (TID 0)
24/07/06 15:15:58 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 1055 bytes result sent to driver
24/07/06 15:15:58 INFO Executor: Finished task 1.0 in stage 0.0 (TID 1). 1055 bytes result sent to driver
24/07/06 15:15:58 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 887 ms on [REDACTED] (executor dri
ver) (1/2)
24/07/06 15:15:58 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 942 ms on [REDACTED] (executor dri
ver) (2/2)
24/07/06 15:15:58 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
24/07/06 15:15:58 INFO DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 1.589 s
24/07/06 15:15:59 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
24/07/06 15:15:58 INFO TaskSchedulerImpl: Killing all running tasks in stage 0: Stage finished
24/07/06 15:15:58 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 1.685101 s
Pi is roughly 3.139675698378492
24/07/06 15:15:58 INFO SparkContext: SparkContext is stopping with exitCode 0.
24/07/06 15:15:58 INFO SparkUI: Stopped Spark web UI at http://[REDACTED]:4040
24/07/06 15:15:58 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
24/07/06 15:15:59 INFO MemoryStore: MemoryStore cleared
24/07/06 15:15:59 INFO BlockManager: BlockManager stopped
24/07/06 15:15:59 INFO BlockManagerMaster: BlockManagerMaster stopped
24/07/06 15:15:59 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
24/07/06 15:15:59 INFO SparkContext: Successfully stopped SparkContext
24/07/06 15:15:59 INFO ShutdownHookManager: Shutdown hook called
24/07/06 15:15:59 INFO ShutdownHookManager: Deleting directory C:\Users\hp\AppData\Local\Temp\spark-79c18435-9bb6-4556-b
71a-789fa3ef3ff5
24/07/06 15:15:59 INFO ShutdownHookManager: Deleting directory C:\Users\hp\AppData\Local\Temp\spark-d6097c4c-2aa7-4009-b
5f3-9d49b7084c95
C:\Users\hp>

```

It displayed the output of approximate Pi value which is **3.1396**. In the console log, we can also see that Spark web UI was started and stopped at <http://localhost:4040>

Note:

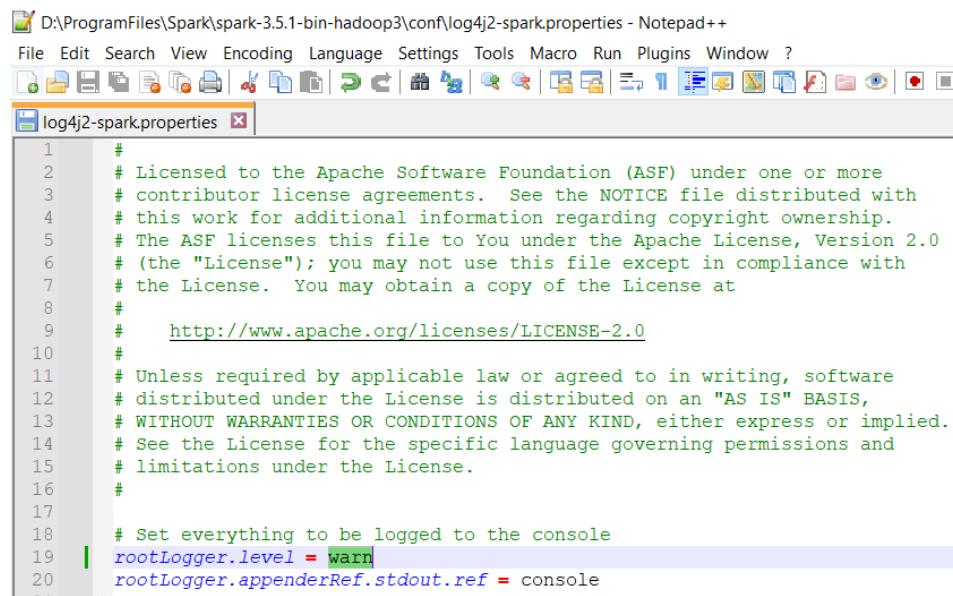
By default, Spark displays **INFO** logging on the console.

Follow the below steps to turn off the unnecessary INFO logging:

- Go to **SPARK_HOME\conf** folder, take the copy of **log4j2.properties.template** file and rename as **log4j2-spark.properties**.

This PC > Local Disk (D:) > ProgramFiles > Spark > spark-3.5.1-bin-hadoop3 > conf				
Name	^	Date modified	Type	Size
fairscheduler.xml.template		2/15/2024 5:06 PM	TEMPLATE File	2 KB
log4j2.properties.template		2/15/2024 5:06 PM	TEMPLATE File	4 KB
log4j2-spark.properties		2/15/2024 5:06 PM	PROPERTIES File	4 KB
metrics.properties.template		2/15/2024 5:06 PM	TEMPLATE File	9 KB
spark-defaults.conf.template		2/15/2024 5:06 PM	TEMPLATE File	2 KB
spark-env.sh.template		2/15/2024 5:06 PM	TEMPLATE File	5 KB
workers.template		2/15/2024 5:06 PM	TEMPLATE File	1 KB

- Open `log4j2-spark.properties` and change the `rootLogger.level` value from `info` to `warn` as shown below:



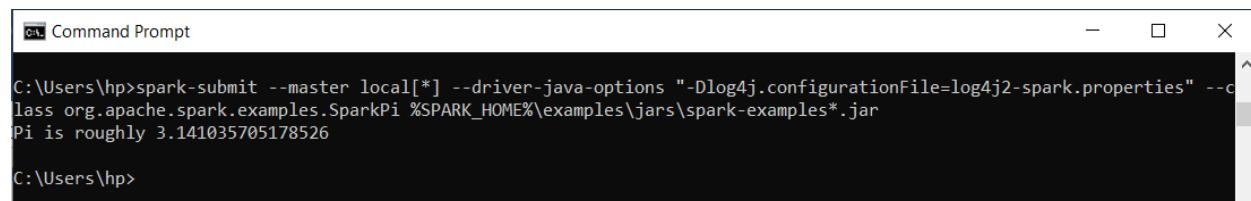
```

D:\ProgramFiles\Spark\spark-3.5.1-bin-hadoop3\conf\log4j2-spark.properties - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
log4j2-spark.properties x
1   #
2   # Licensed to the Apache Software Foundation (ASF) under one or more
3   # contributor license agreements. See the NOTICE file distributed with
4   # this work for additional information regarding copyright ownership.
5   # The ASF licenses this file to You under the Apache License, Version 2.0
6   # (the "License"); you may not use this file except in compliance with
7   # the License. You may obtain a copy of the License at
8   #
9   #     http://www.apache.org/licenses/LICENSE-2.0
10  #
11  # Unless required by applicable law or agreed to in writing, software
12  # distributed under the License is distributed on an "AS IS" BASIS,
13  # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14  # See the License for the specific language governing permissions and
15  # limitations under the License.
16  #
17  #
18  # Set everything to be logged to the console
19  rootLogger.level = warn
20  rootLogger.appenderRef.stdout.ref = console

```

Now, execute the below `spark-submit` command and we will not see any **INFO** level logging on the console.

```
spark-submit --master local[*] --driver-java-options "-Dlog4j.configurationFile=log4j2-spark.properties" --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
```



```
C:\Users\hp>spark-submit --master local[*] --driver-java-options "-Dlog4j.configurationFile=log4j2-spark.properties" --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
Pi is roughly 3.141035705178526
C:\Users\hp>
```

In Client Mode:

To run the SparkPi application in client mode, use the below command (*Make sure that you replace <ipaddress> with the IP address mentioned in your Spark Master URL*)

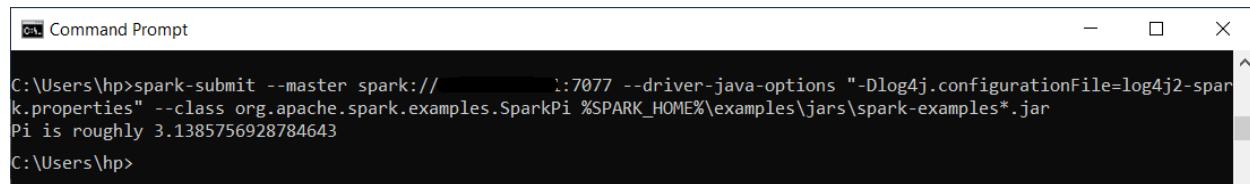
```
spark-submit --master spark://<ipaddress>:7077 --deploy-mode client --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
```

or

```
spark-submit --master spark://<ipaddress>:7077 --class  
org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-  
examples*.jar
```

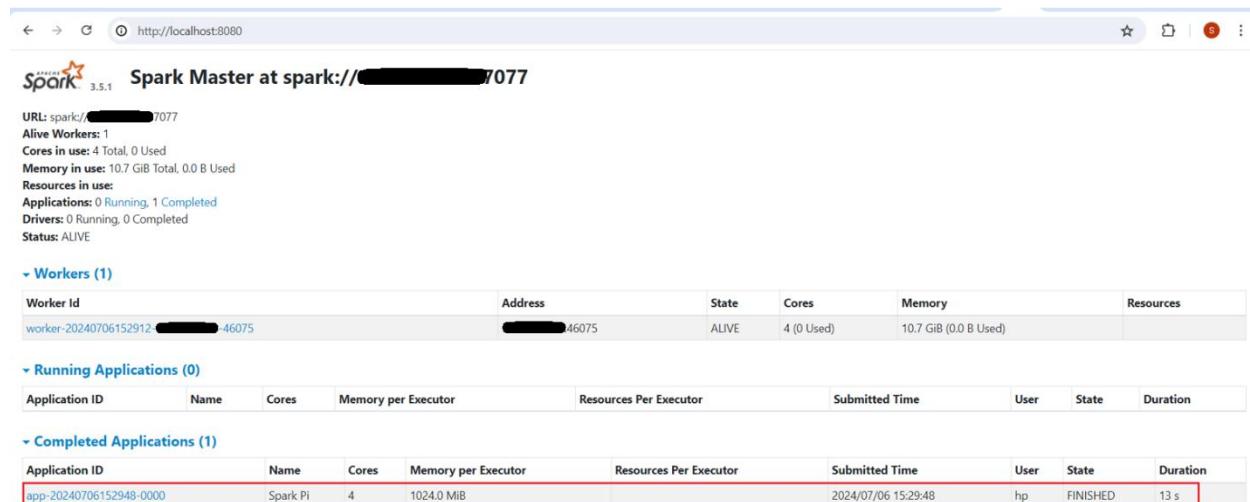
If you want to skip the INFO logging on console, use the extra property `--driver-java-options "-Dlog4j.configurationFile=log4j2-spark.properties"` in the `spark-submit` command as below:

```
spark-submit --master spark://<ipaddress>:7077 --driver-java-options  
"-Dlog4j.configurationFile=log4j2-spark.properties" --class  
org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-  
examples*.jar
```



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The command entered is `spark-submit --master spark://<ipaddress>:7077 --driver-java-options "-Dlog4j.configurationFile=log4j2-spark.properties" --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar`. The output shows the application has been submitted and is running.

On the Master Web UI (<http://localhost:8080>), we can see that **Spark Pi** application has been submitted and completed.



The screenshot shows the Spark Master Web UI at <http://localhost:8080>. The main page displays the following information:

- URL: `spark://<ipaddress>:7077`
- Alive Workers: 1
- Cores in use: 4 Total, 0 Used
- Memory in use: 10.7 GiB Total, 0.0 B Used
- Resources in use:
- Applications: 0 Running, 1 Completed
- Drivers: 0 Running, 0 Completed
- Status: ALIVE

Below this, there are sections for "Workers (1)", "Running Applications (0)", and "Completed Applications (1)". The "Completed Applications" section shows one entry:

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240706152948-0000	Spark Pi	4	1024.0 MiB		2024/07/06 15:29:48	hp	FINISHED	13 s

In Cluster Mode:

To run the SparkPi application in cluster mode, use the below command (*Make sure that you replace <ipaddress> with the IP address mentioned in your Spark Master URL*)

```
spark-submit --master spark://<ipaddress>:7077 --deploy-mode cluster
--class org.apache.spark.examples.SparkPi
%SPARK_HOME%\examples\jars\spark-examples*.jar
```

```
C:\Users\hp>spark-submit --master spark://:7077 --deploy-mode cluster --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
24/07/06 15:33:41 INFO SecurityManager: Changing view acls to: hp
24/07/06 15:33:41 INFO SecurityManager: Changing modify acls to: hp
24/07/06 15:33:41 INFO SecurityManager: Changing view acls groups to:
24/07/06 15:33:41 INFO SecurityManager: Changing modify acls groups to:
24/07/06 15:33:41 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: hp; groups with view permissions: EMPTY; users with modify permissions: hp; groups with modify permissions: EMPTY
24/07/06 15:33:42 INFO Utils: Successfully started service 'driverClient' on port 46230.
24/07/06 15:33:42 INFO TransportClientFactory: Successfully created connection to /:7077 after 88 ms (0 ms spent in bootstraps)
24/07/06 15:33:42 INFO ClientEndpoint: ... waiting before polling master for driver state
24/07/06 15:33:43 INFO ClientEndpoint: Driver successfully submitted as driver-20240706153342-0000
24/07/06 15:33:48 INFO ClientEndpoint: State of driver-20240706153342-0000 is RUNNING
24/07/06 15:33:48 INFO ClientEndpoint: Driver running on :46075 (worker-20240706152912-46075)
24/07/06 15:33:48 INFO ClientEndpoint: spark-submit not configured to wait for completion, exiting spark-submit JVM.
24/07/06 15:33:48 INFO ShutdownHookManager: Shutdown hook called
24/07/06 15:33:48 INFO ShutdownHookManager: Deleting directory C:\Users\hp\AppData\Local\Temp\spark-5df11884-7442-4bb7-9d4e-e4869d38b559
C:\Users\hp>
```

Since we used “cluster” deploy mode, we can see that the Spark application has been submitted as Driver and once the Driver started running, it exited the spark-submit JVM.

We can track the further application status on the Master Web UI at <http://localhost:8080>

Spark Master at spark://[REDACTED]:7077

URL: spark://[REDACTED]:7077
Alive Workers: 1
Cores in use: 4 Total, 0 Used
Memory in use: 10.7 GB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 2 Completed
Drivers: 0 Running, 1 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20240706152912-[REDACTED]-46075	[REDACTED]:46075	ALIVE	4 (0 Used)	10.7 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Running Drivers (0)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class	Duration

Completed Applications (2)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240706153342-0001	Spark Pi	3	1024.0 MiB		2024/07/06 15:33:54	hp	FINISHED	13 s
app-20240706152948-0000	Spark Pi	4	1024.0 MiB		2024/07/06 15:29:48	hp	FINISHED	13 s

Completed Drivers (1)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20240706153342-0000	2024/07/06 15:33:42	worker-20240706152912-[REDACTED]-46075	FINISHED	1	1024.0 MiB		org.apache.spark.examples.SparkPi

On the Master Web UI, we can see that a new application ID has been submitted and finished. We can also see the application submission details under **Completed Drivers** section.

To see the application output, go to the Worker Web UI <http://localhost:8081> and click on **stdout** link under **Finished Drivers** section.

The screenshot shows the Apache Spark 3.5.1 Master Web UI. At the top, it displays "Spark Worker at [REDACTED]:46075" with resource details: ID: worker-20240706152912-[REDACTED]-46075, Master URL: spark://[REDACTED]:7077, Cores: 4 (0 Used), Memory: 10.7 GiB (0.0 B Used), and Resources: Back to Master.

Below this, there are two sections: "Running Executors (0)" and "Finished Executors (2)". The "Finished Executors" table lists two entries:

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
0	KILLED	4	1024.0 MiB		ID: app-20240706152948-0000 Name: Spark Pi User: hp	stdout stderr
0	KILLED	3	1024.0 MiB		ID: app-20240706153354-0001 Name: Spark Pi User: hp	stdout stderr

Finally, the "Finished Drivers (1)" section is shown, with one entry:

DriverID	Main Class	State	Cores	Memory	Resources	Logs	Notes
driver-20240706153342-0000	org.apache.spark.examples.SparkPi	FINISHED	1	1024.0 MiB		stdout stderr	

We can see the standard output "*Pi is roughly 3.135*" on the log page for the respective Driver ID.

The screenshot shows the Apache Spark 3.5.1 Worker Web UI log page for driver-20240706153342-0000. It displays the log output: "Pi is roughly 3.141592653589793".

Note:

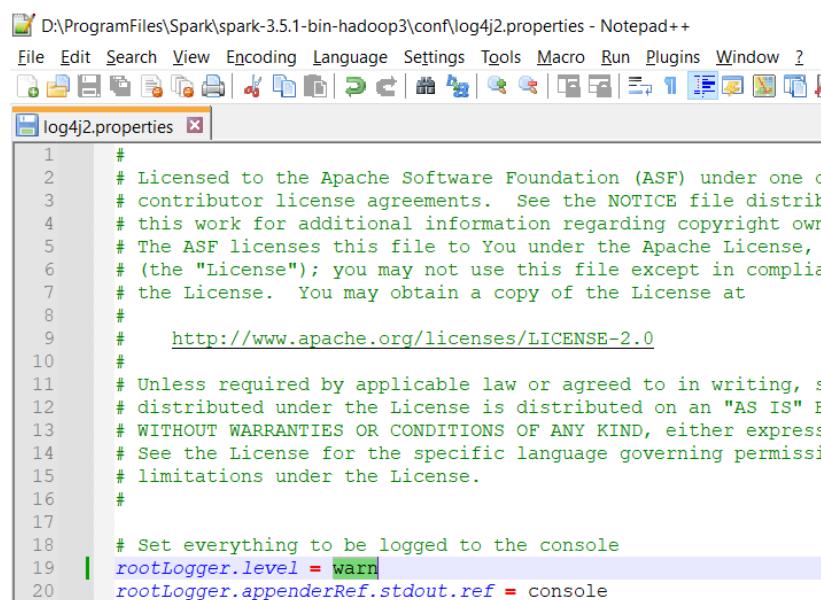
Follow the below steps to skip **INFO** logging on console while executing Spark in cluster mode:

- Go to `SPARK_HOME\conf` folder, take the copy of `log4j2.properties.template` file and rename as `log4j2.properties`.

File PC Local Disk (D:) ProgramFiles Spark spark-3.5.1-bin-hadoop3 conf

Name	Date modified	Type	Size
fairscheduler.xml.template	2/15/2024 5:06 PM	TEMPLATE File	2 KB
log4j2.properties	2/15/2024 5:06 PM	PROPERTIES File	4 KB
log4j2.properties.template	2/15/2024 5:06 PM	TEMPLATE File	4 KB
log4j2-spark.properties	7/6/2024 3:25 PM	PROPERTIES File	4 KB
metrics.properties.template	2/15/2024 5:06 PM	TEMPLATE File	9 KB
spark-defaults.conf.template	2/15/2024 5:06 PM	TEMPLATE File	2 KB
spark-env.sh.template	2/15/2024 5:06 PM	TEMPLATE File	5 KB
workers.template	2/15/2024 5:06 PM	TEMPLATE File	1 KB

- Open `log4j2.properties` and change the `rootLogger.level` value from `info` to `warn` as shown below:



```

1  #
2  # Licensed to the Apache Software Foundation (ASF) under one or more
3  # contributor license agreements. See the NOTICE file distributed
4  # with this work for additional information regarding copyright ownership.
5  # The ASF licenses this file to You under the Apache License,
6  # (the "License"); you may not use this file except in compliance
7  # with the License. You may obtain a copy of the License at
8  #
9  #     http://www.apache.org/licenses/LICENSE-2.0
10 #
11 # Unless required by applicable law or agreed to in writing, software
12 # distributed under the License is distributed on an "AS IS" BASIS,
13 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
14 # or implied. See the License for the specific language governing
15 # permissions and limitations under the License.
16 #
17 #
18 # Set everything to be logged to the console
19 rootLogger.level = warn
20 rootLogger.appenderRef.stdout.ref = console

```

Now, execute the `spark-submit` command with cluster deploy mode and we will not see any INFO messages on the console.

```

spark-submit --master spark://<ipaddress>:7077 --deploy-mode cluster
--class org.apache.spark.examples.SparkPi
%SPARK_HOME%\examples\jars\spark-examples*.jar

```



```

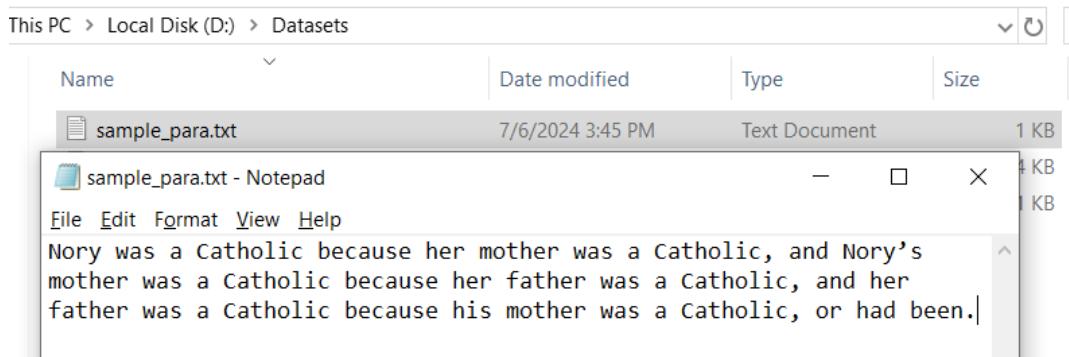
C:\Users\hp>spark-submit --master spark://192.168.56.1:7077 --deploy-mode cluster --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
C:\Users\hp>

```

3.5.2. Run Python wordcount Program:

Let us execute the Python program `wordcount` which counts the occurrences of each word in the given input file. This program is located in `SPARK_HOME\examples\src\main\python` directory.

First, create an input file named `sample_para.txt` with some random text and place it in `D:\Datasets` folder.



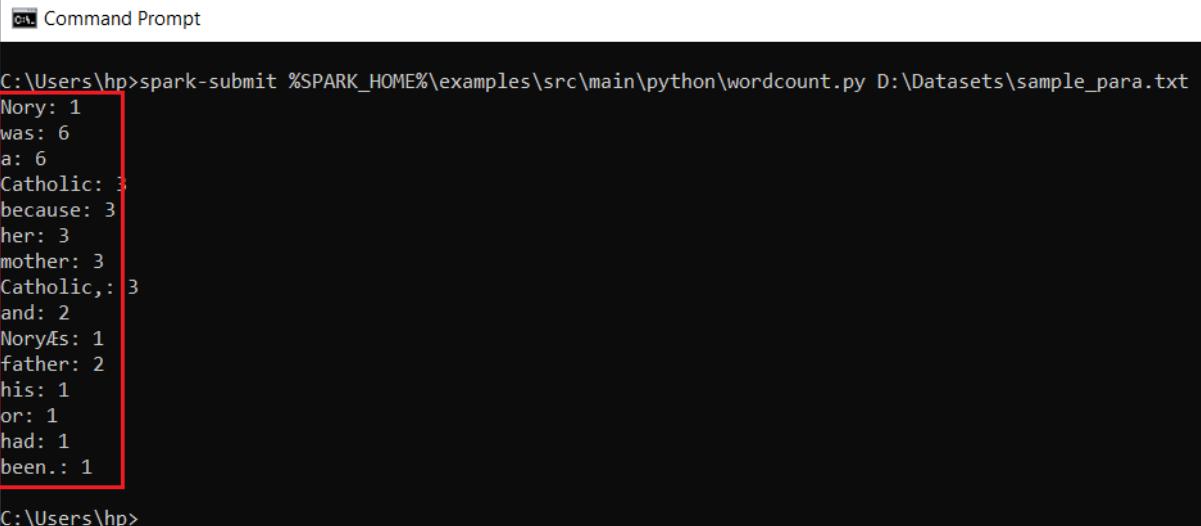
Next, we will run the `wordcount.py` program in local mode, client mode and cluster mode.

In Local Mode:

Use this command to execute the **PythonWordCount** application in local mode with all available cores:

```
spark-submit --master local[*]
%SPARK_HOME%\examples\src\main\python\wordcount.py
D:\Datasets\sample_para.txt
or
spark-submit %SPARK_HOME%\examples\src\main\python\wordcount.py
D:\Datasets\sample_para.txt
```

After the above command is executed, it displays the count of each word available in the input file on the console.

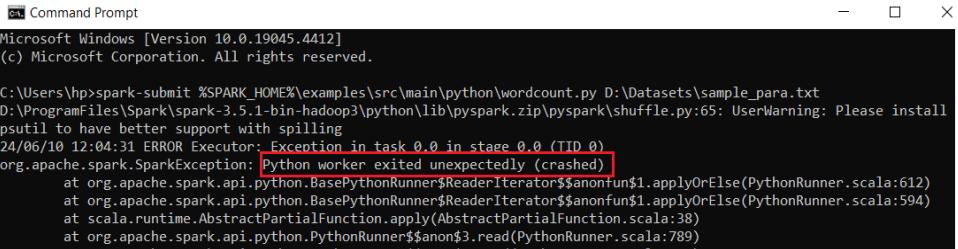


```
C:\Users\hp>spark-submit %SPARK_HOME%\examples\src\main\python\wordcount.py D:\Datasets\sample_para.txt
Nory: 1
was: 6
a: 6
Catholic: 3
because: 3
her: 3
mother: 3
Catholic,: 3
and: 2
Nory&s: 1
father: 2
his: 1
or: 1
had: 1
been.: 1
C:\Users\hp>
```

Here, we are not seeing any **INFO** messages on the console here because we have set `rootLogger.level to warn` in `SPARK_HOME\conf\log4j2.properties` file.

Note:

While executing Python programs in Spark, you may receive error ***SparkException: Python worker exited unexpectedly (crashed)*** as below:



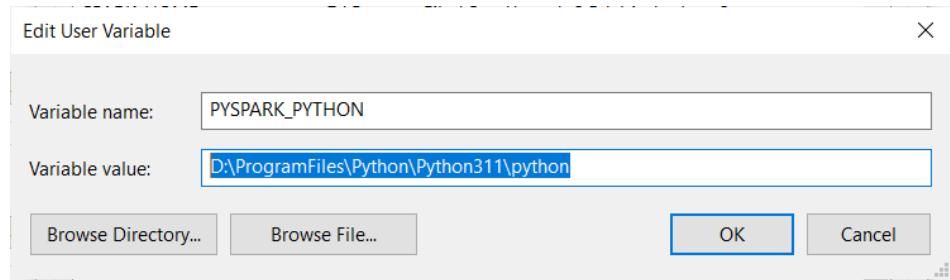
```
Microsoft Windows [Version 10.0.19045.4412]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-submit %SPARK_HOME%\examples\src\main\python\wordcount.py D:\Datasets\sample_para.txt
D:\Programfiles\Spark\spark-3.5.1-bin-hadoop3\python\lib\pyspark.zip\pyspark\shuffle.py:65: UserWarning: Please install
psutil to have better support with spilling
24/06/10 12:04:31 ERROR Executor: Exception in task 0.0 in stage 0.0 (TID 0)
org.apache.spark.SparkException: Python worker exited unexpectedly (crashed)
    at org.apache.spark.api.python.BasePythonRunner$ReaderIterator$$anonfun$1.applyOrElse(PythonRunner.scala:612)
    at org.apache.spark.api.python.BasePythonRunner$ReaderIterator$$anonfun$1.applyOrElse(PythonRunner.scala:594)
    at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)
    at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:789)
```

This error mostly occurs due to the latest Python version 3.12.

Follow the below steps to resolve the above error:

- Install the lower version of Python (say 3.11.9) from the [official python downloads website](#). I installed Python 3.11.9 version in `D:\ProgramFiles\Python\Python311` folder.
- Open **Environment Variables** dialog, select `PYSPARK_PYTHON` variable under **User variables** tab and press **Edit** button
- Update the value to the new location where the older version of `python.exe` file is available (*in my case, it is D:\ProgramFiles\Python\Python311\python*). Press **OK** and then **OK** to apply changes



- Open a new command prompt and start executing PySpark programs.

In Client Mode:

Use this command to execute the **PythonWordCount** application in client mode (*Make sure that you replace <ipaddress> with the IP address mentioned in your Spark Master URL*)

```
spark-submit --master spark://<ipaddress>:7077 --deploy-mode client  
%SPARK_HOME%\examples\src\main\python\wordcount.py  
D:\Datasets\sample_para.txt
```

or

```
spark-submit --master spark://<ipaddress>:7077  
%SPARK_HOME%\examples\src\main\python\wordcount.py  
D:\Datasets\sample_para.txt
```

After the above command is executed, it displays the count of each word available in the input file on the console.

```
C:\Users\hp>spark-submit --master spark://<ipaddress>:7077 %SPARK_HOME%\examples\src\main\python\wordcount.py D:\Datasets\sample_para.txt  
Nory: 1  
was: 6  
a: 6  
Catholic: 3  
because: 3  
her: 3  
mother: 3  
Catholic,: 3  
and: 2  
Nory&s: 1  
father: 2  
his: 1  
or: 1  
had: 1  
been.: 1  
C:\Users\hp>
```

On the Master Web UI (<http://localhost:8080>), we can see that **PythonWordCount** application has been submitted and completed.

The screenshot shows the Apache Spark 3.5.1 Master Web UI at <http://localhost:8080>. The main page displays the following information:

- URL:** spark://[REDACTED]:7077
- Alive Workers:** 1
- Cores in use:** 4 Total, 0 Used
- Memory in use:** 10.7 Gib Total, 0.0 B Used
- Resources in use:**
- Applications:** 0 Running, 4 Completed
- Drivers:** 0 Running, 2 Completed
- Status:** ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20240706152912-[REDACTED] 46075	[REDACTED] 46075	ALIVE	4 (0 Used)	10.7 Gib (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Running Drivers (0)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class	Duration

Completed Applications (4)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240706155116-0003	PythonWordCount	4	1024.0 MiB		2024/07/06 15:51:16	hp	FINISHED	20 s
app-20240706154337-0002	Spark Pi	3	1024.0 MiB		2024/07/06 15:43:37	hp	FINISHED	15 s
app-20240706153354-0001	Spark Pi	3	1024.0 MiB		2024/07/06 15:33:54	hp	FINISHED	13 s
app-20240706152948-0000	Spark Pi	4	1024.0 MiB		2024/07/06 15:29:48	hp	FINISHED	13 s

In Cluster Mode:

Use this command to execute the **PythonWordCount** application in cluster mode (*Make sure that you replace <ipaddress> with the IP address mentioned in your Spark Master URL*)

```
spark-submit --master spark://<ipaddress>:7077 --deploy-mode cluster
%SPARK_HOME%\examples\src\main\python\wordcount.py
D:\Datasets\sample_para.txt
```

The screenshot shows a Command Prompt window with the following output:

```
C:\Users\hp>spark-submit --master spark://192.168.56.1:7077 --deploy-mode cluster %SPARK_HOME%\examples\src\main\python\wordcount.py D:\Datasets\sample_para.txt
Exception in thread "main" org.apache.spark.SparkException: Cluster deploy mode is currently not supported for python applications on standalone clusters.
    at org.apache.spark.deploy.SparkSubmit.error(SparkSubmit.scala:1047)
    at org.apache.spark.deploy.SparkSubmit.prepareSubmitEnvironment(SparkSubmit.scala:293)
    at org.apache.spark.deploy.SparkSubmit.org$apache$spark$deploy$SparkSubmit$$runMain(SparkSubmit.scala:964)
    at org.apache.spark.deploy.SparkSubmit.doRunMain$1(SparkSubmit.scala:194)
    at org.apache.spark.deploy.SparkSubmit.submit(SparkSubmit.scala:217)
    at org.apache.spark.deploy.SparkSubmit.org$apache$spark$deploy$SparkSubmit$$doSubmit(SparkSubmit.scala:91)
    at org.apache.spark.deploy.SparkSubmit$$anon$2.org$apache$spark$deploy$SparkSubmit$$doSubmit(SparkSubmit.scala:1120)
    at org.apache.spark.deploy.SparkSubmit.org$main(SparkSubmit.scala:1129)
    at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

C:\Users\hp>

However, it throws **SparkException** because Spark does not support Cluster deploy mode of Python applications on Standalone clusters.

4. Install Spark on Hadoop:

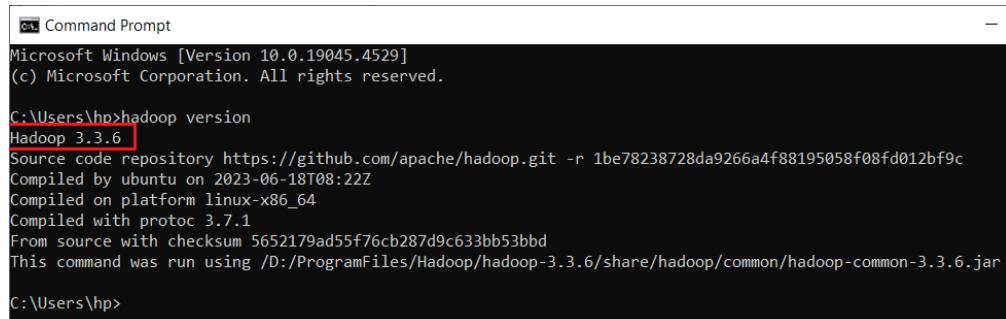
Now, we will see how to install Spark on top of existing Hadoop cluster.

4.1. Verify Hadoop Installation:

If you do not have Hadoop running, refer [here](#) to install Hadoop on Windows operating system.

Open **Command Prompt or Windows PowerShell** and run the following command:

```
hadoop version
```



```
Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

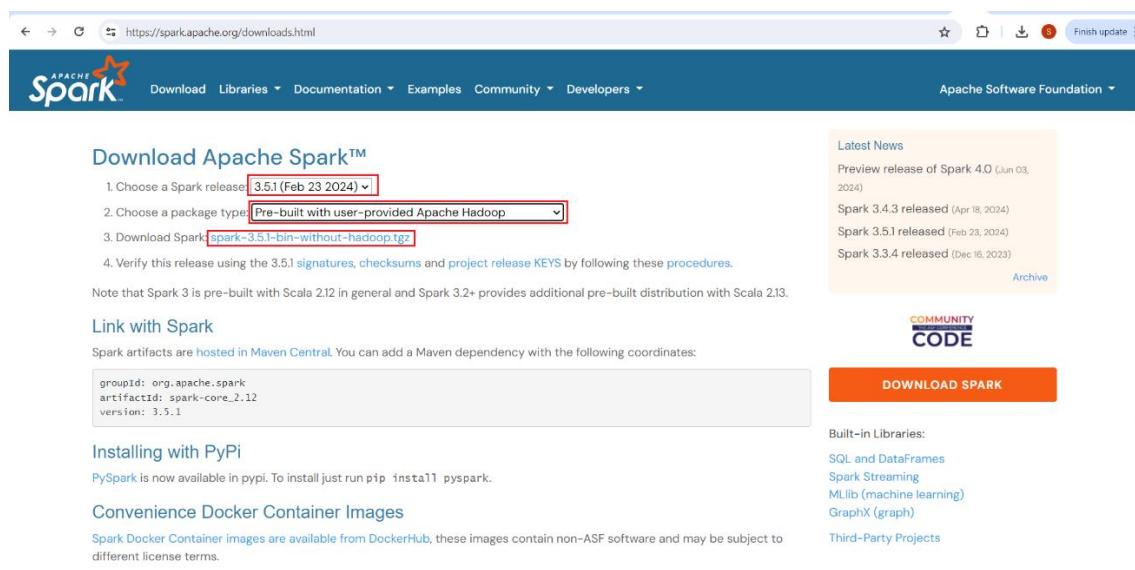
C:\Users\hp>hadoop version
Hadoop 3.3.6
Source code repository https://github.com/apache/hadoop.git -r 1be78238728da9266a4f88195058f08fd012bf9c
Compiled by ubuntu on 2023-06-18T08:22Z
Compiled on platform linux-x86_64
Compiled with protoc 3.7.1
From source with checksum 5652179ad55f76cb287d9c633bb53bbd
This command was run using /D:/ProgramFiles/Hadoop/hadoop-3.3.6/share/hadoop/common/hadoop-common-3.3.6.jar

C:\Users\hp>
```

It shows that **Hadoop 3.3.6** version has been installed.

4.2. Download Spark Binaries:

Download the latest **Spark 3.5.1** version from the [Apache Spark Downloads](#) page. In this page, choose the **package type** as **Pre-built with user-provided Apache Hadoop** and click on `spark-3.5.1-bin-without-hadoop.tgz` binary file.



The screenshot shows the Apache Spark Downloads page. The URL is <https://spark.apache.org/downloads.html>. The page features a navigation bar with links for Download, Libraries, Documentation, Examples, Community, Developers, and Apache Software Foundation. The main content area is titled "Download Apache Spark™". It contains a form with four steps: 1. Choose a Spark release (3.5.1 (Feb 23 2024)), 2. Choose a package type (Pre-built with user-provided Apache Hadoop), 3. Download Spark (spark-3.5.1-bin-without-hadoop.tgz), and 4. Verify this release using the 3.5.1 signatures, checksums and project release KEYS by following these procedures. Below the form, it notes that Spark 3 is pre-built with Scala 2.12 in general and Spark 3.2+ provides additional pre-built distribution with Scala 2.13. A "Link with Spark" section provides Maven dependency coordinates: groupId: org.apache.spark, artifactId: spark-core_2.12, version: 3.5.1. To the right, there's a "DOWNLOAD SPARK" button, a "COMMUNITY CODE" section, and a "Latest News" sidebar with links to preview releases of Spark 4.0, 3.4.3, 3.5.1, and 3.3.4, along with an "Archive" link.

You will be navigated to [spark-3.5.1](#) mirror website where click on the suggested location for [spark-3.5.1-bin-wihtout-hadoop.tgz](#) file which gets downloaded to your **Downloads** folder.

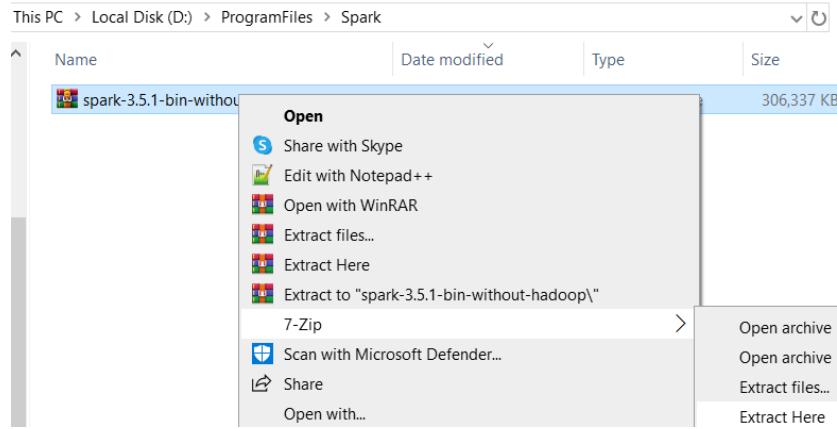
The screenshot shows a web browser displaying the Apache Software Foundation's download page for Apache Spark 3.5.1. The URL in the address bar is <https://www.apache.org/dyn/closer.lua/spark/spark-3.5.1/spark-3.5.1-bin-without-hadoop.tgz>. The page features the Apache logo and navigation links for Community, Projects, Downloads, Learn, Resources & Tools, and About. A prominent link to the download file is shown: <https://dlcdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-without-hadoop.tgz>. Below this, there are sections for HTTP download (<https://dlcdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-without-hadoop.tgz>) and Backup Sites (<https://dlcdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-without-hadoop.tgz>). A section titled "VERIFY THE INTEGRITY OF THE FILES" provides instructions on verifying the integrity of the download using PGP signatures or hashes.

After the binary file is downloaded, unpack it using any file archiver (**7zip** or **WinRAR**) utility as below:

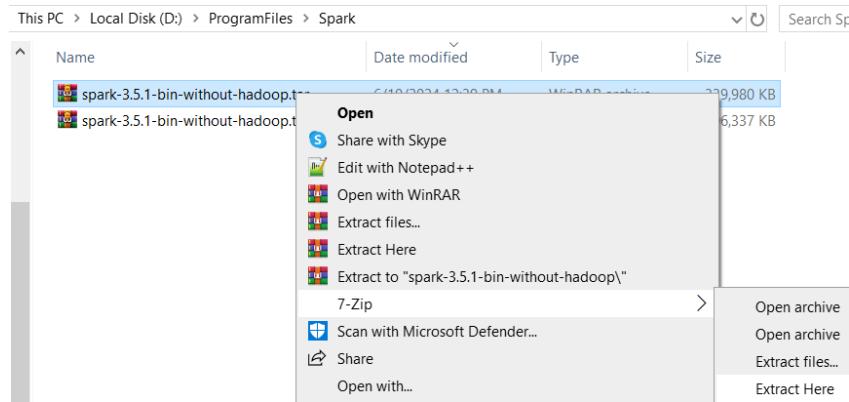
- Choose the installation directory in your machine and copy `spark-3.5.1-bin-without-hadoop.tgz` file to that directory. Here, we are choosing Spark installation directory as `D:\ProgramFiles\Spark`.

This PC > Local Disk (D:) > ProgramFiles > Spark			
Name	Date modified	Type	Size
spark-3.5.1-bin-without-hadoop.tgz	7/6/2024 5:15 PM	WinRAR archive	306,337 KB

- Right click on `spark-3.5.1-bin-without-hadoop.tgz` and choose **7-Zip -> Extract Here** option which extracts a new packed file `spark-3.5.1-bin-without-hadoop.tar`.



- Next, unpack `spark-3.5.1-without-hadoop.tar` file using 7zip utility.



- The tar file extraction may take few minutes to finish. After finishing, you see a folder named `spark-3.5.1-bin-without-hadoop` which consists of Spark binaries and libraries.

Name	Date modified	Type	Size
bin	2/15/2024 4:54 PM	File folder	
conf	2/15/2024 4:54 PM	File folder	
data	2/15/2024 4:54 PM	File folder	
examples	2/15/2024 4:54 PM	File folder	
jars	2/15/2024 4:54 PM	File folder	
kubernetes	2/15/2024 4:54 PM	File folder	
licenses	2/15/2024 4:54 PM	File folder	
python	2/15/2024 4:54 PM	File folder	
R	2/15/2024 4:54 PM	File folder	
sbin	2/15/2024 4:54 PM	File folder	
yarn	2/15/2024 4:54 PM	File folder	
LICENSE	2/15/2024 4:54 PM	File	23 KB
NOTICE	2/15/2024 4:54 PM	File	57 KB
README.md	2/15/2024 4:54 PM	MD File	5 KB
RELEASE	2/15/2024 4:54 PM	File	1 KB

4.3. Set up Environment Variables:

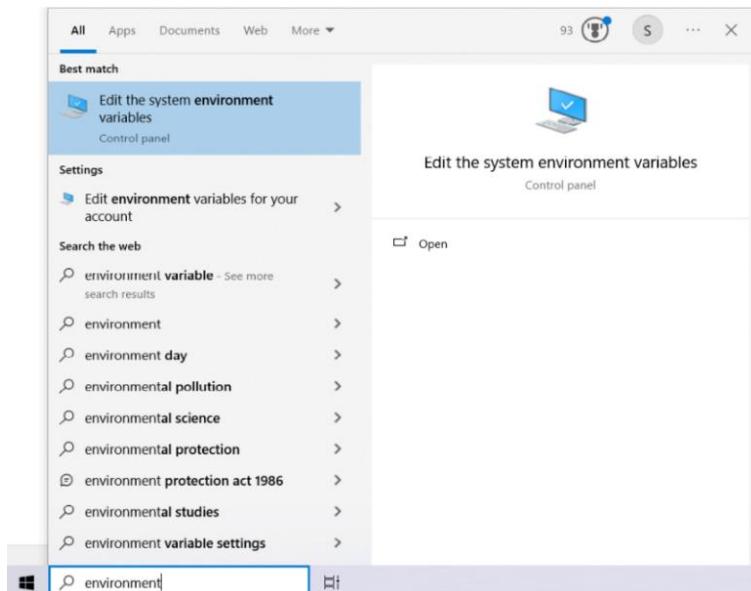
After installing Spark binaries, we should configure the below environment variables defining the Spark default paths.

- **SPARK_HOME**: This is the Spark installation directory path in the machine (*in our case, it is D:\ProgramFiles\Spark\spark-3.5.1-bin-without-hadoop*)
- **PYTHONPATH**: This is the Spark python directory path which should be.
`%SPARK_HOME%\python`
- **PYSpark_Python**: Set this variable to the location where `python.exe` file is available (*in my machine, it is D:\ProgramFiles\Anaconda\anaconda3\python*) which is needed for Spark to execute the Python code.

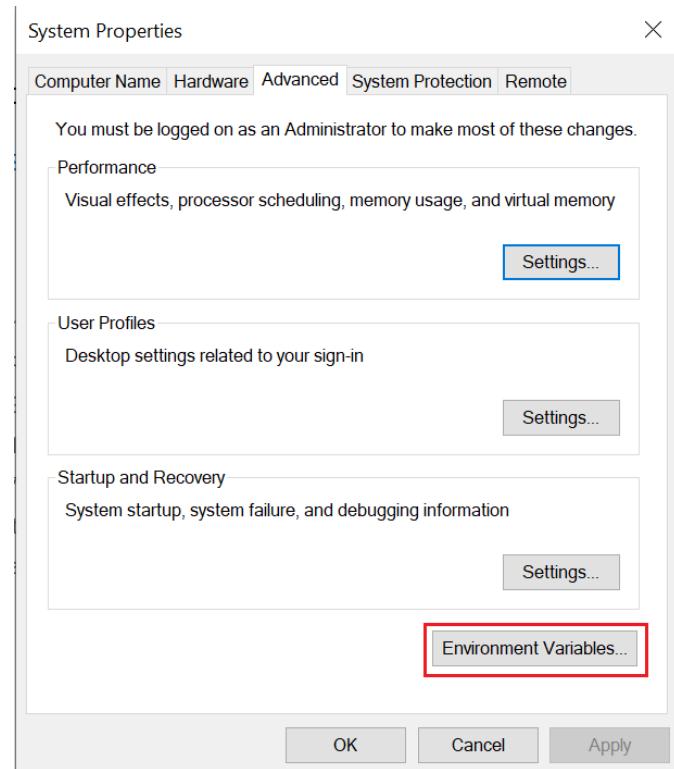
These variables need to be added to either **User environment variables** or **System environment variables** depending on Spark configuration needed **for a single user** or **for multiple users**.

In this tutorial, we will add User environment variables since we are configuring Spark for a single user. If you would like to configure Spark for multiple users, then define System environment variables.

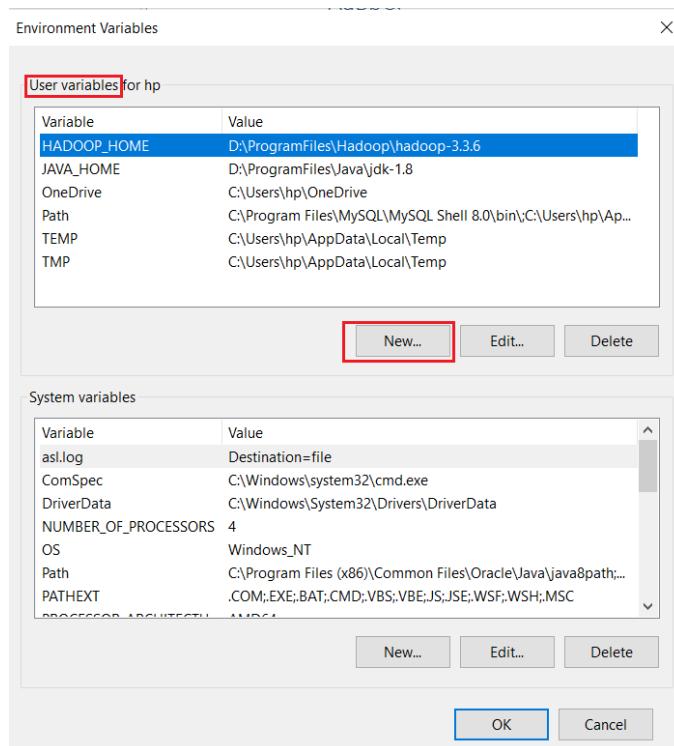
In the Windows search bar, start typing “environment variables” and select the first match which opens up **System Properties** dialog.



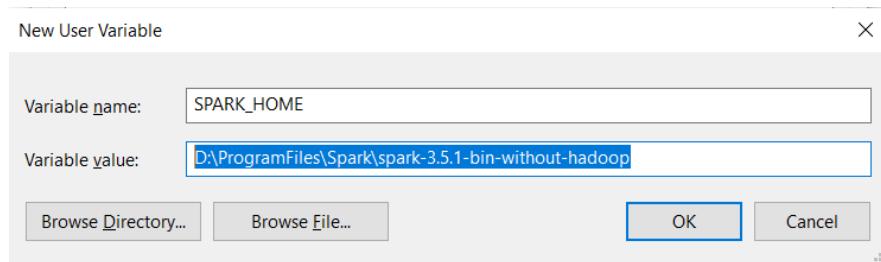
On the **System Properties** window, press **Environment Variables** button.



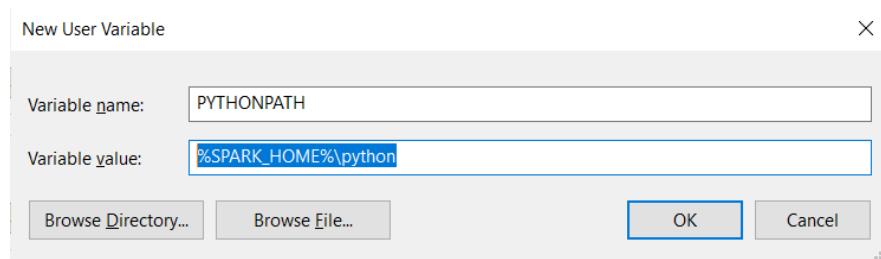
In the **Environment Variables** dialog, click on **New** under **User variables** section.



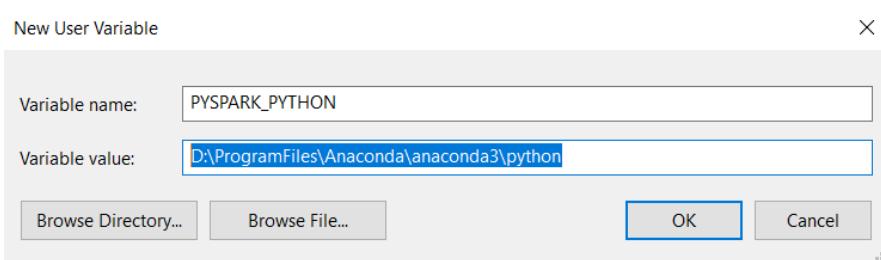
Add **SPARK_HOME** variable and press OK.



Click on **New** again and add **PYTHONPATH** variable and press OK.

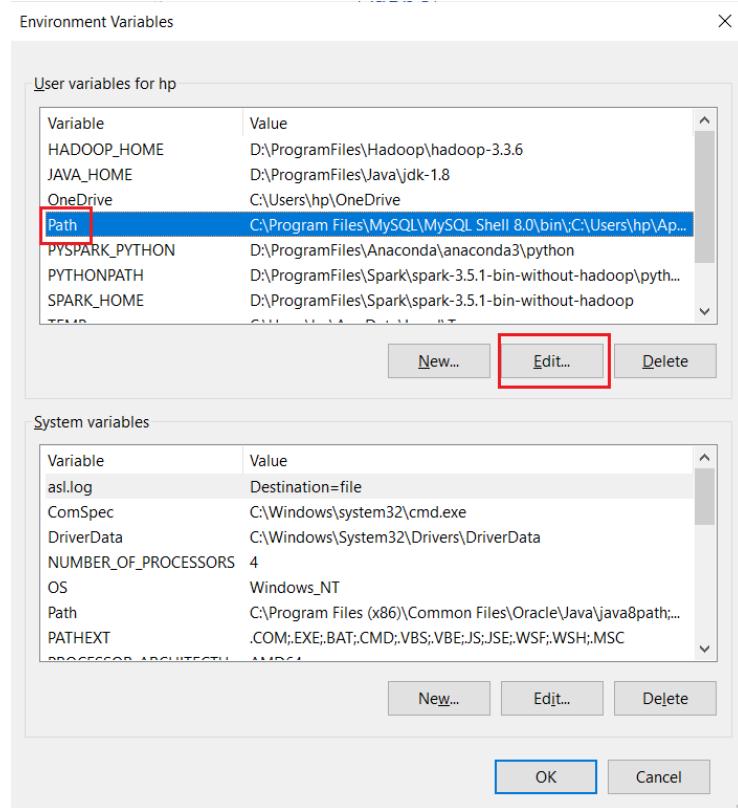


Click on **New** again and add **PYSPARK_PYTHON** variable and press OK.

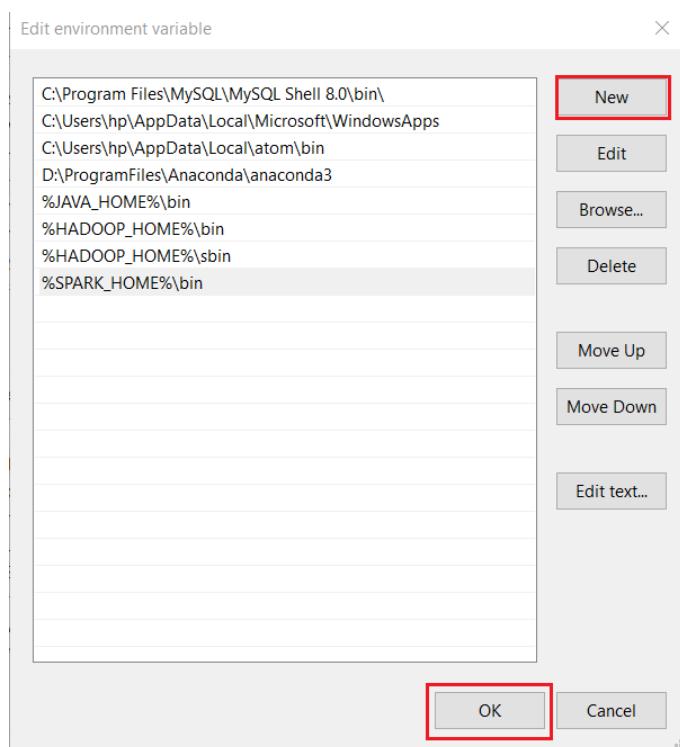


Now, we will update **PATH** variable to add Spark binary paths.

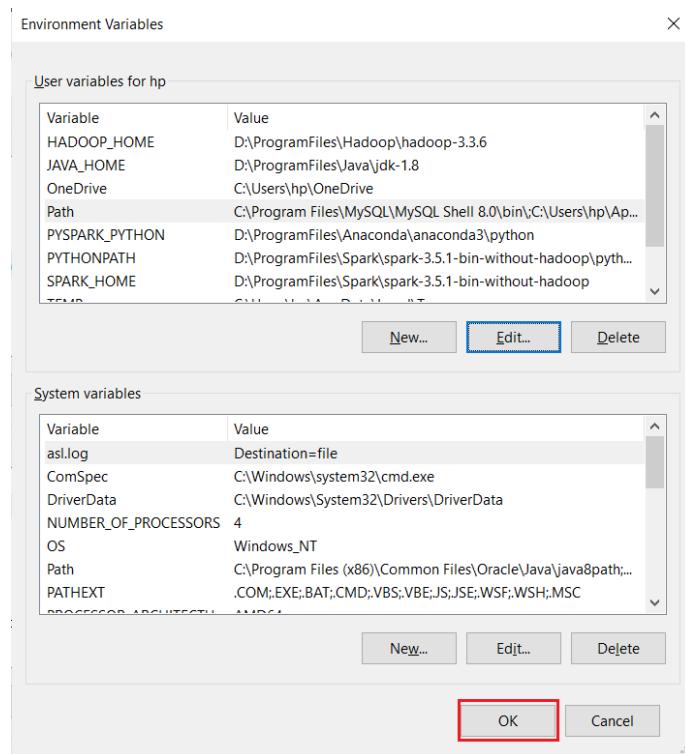
Select **PATH** variable under **User Variables** and press **Edit** button.



Press **New** and add `%SPARK_HOME%\bin` value and press **OK**.



Press OK again to apply environment variable changes and close window.



4.4. Configure Spark Environment:

To use Spark's "Hadoop Free" build i.e to use Spark on top of existing Hadoop cluster, we need to set SPARK_DIST_CLASSPATH environment variable in SPARK_HOME\conf\spark-env.cmd file.

Go to SPARK_HOME\conf location, create spark-env.cmd file and add the following lines

```
@echo off

@rem verify HADOOP_HOME env variable as a prereq
if not defined HADOOP_HOME (
    echo "HADOOP_HOME needs to be defined to point at the hadoop
installation"
    exit /b 1
)

@rem set Hadoop config directory
set HADOOP_CONF_DIR=%HADOOP_HOME%\etc\hadoop

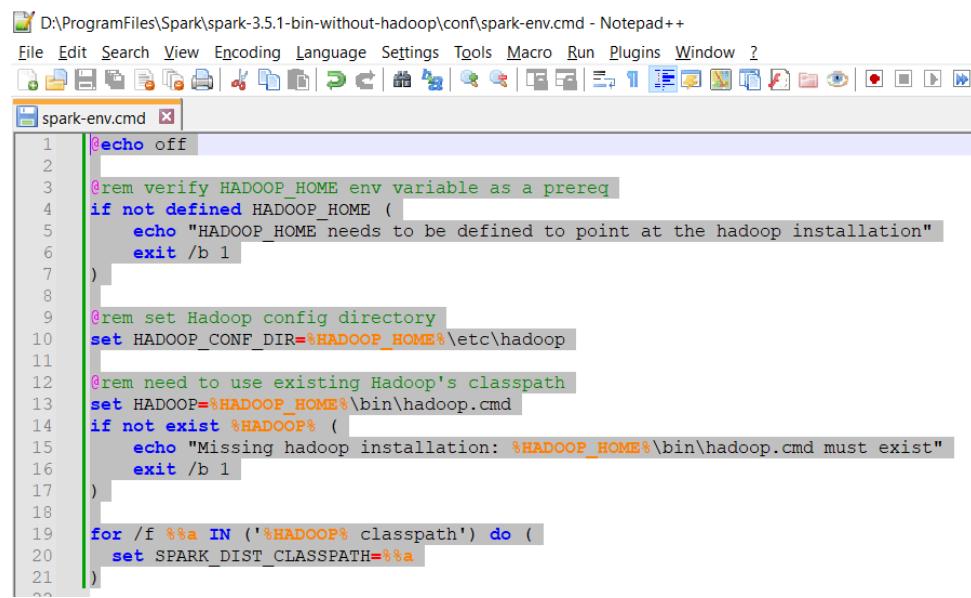
@rem need to use existing Hadoop's classpath
set HADOOP=%HADOOP_HOME%\bin\hadoop.cmd
if not exist %HADOOP% (
```

```

        echo "Missing hadoop installation: %HADOOP_HOME%\bin\hadoop.cmd must
exist"
        exit /b 1
    )

for /f %%a IN ('%HADOOP% classpath') do (
    set SPARK_DIST_CLASSPATH=%%a
)

```



The screenshot shows a Notepad++ window titled "spark-env.cmd". The code inside the file is as follows:

```

1  echo off
2
3  @rem verify HADOOP_HOME env variable as a prereq
4  if not defined HADOOP_HOME (
5      echo "HADOOP_HOME needs to be defined to point at the hadoop installation"
6      exit /b 1
7 )
8
9  @rem set Hadoop config directory
10 set HADOOP_CONF_DIR=%HADOOP_HOME%\etc\hadoop
11
12 @rem need to use existing Hadoop's classpath
13 set HADOOP=%HADOOP_HOME%\bin\hadoop.cmd
14 if not exist %HADOOP% (
15     echo "Missing hadoop installation: %HADOOP_HOME%\bin\hadoop.cmd must exist"
16     exit /b 1
17 )
18
19 for /f %%a IN ('%HADOOP% classpath') do (
20     set SPARK_DIST_CLASSPATH=%%a
21 )

```

4.5. Verify Spark Installation:

Open Windows Command prompt or Windows PowerShell prompt and run the following command:

```
pyspark --version
```

After the above command is executed, you can see the installed **Spark version 3.5.1**.

4.6. Start Hadoop Services:

Open Windows PowerShell or Command Prompt as Administrator and start services.

4.6.1. Start Hadoop Nodes:

Run the following command to start the Hadoop nodes.

start-dfs.cmd

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> start-dfs.cmd
PS C:\Windows\system32>
```

After executing the above command, it opens up two command prompt windows - one for the **namenode** and other for the **datanode** as below. Wait until namenode service says “*Quota initialization completed*” and datanode service says “*Successfully sent block report to namenode: localhost/127.0.0.1:9820*”.

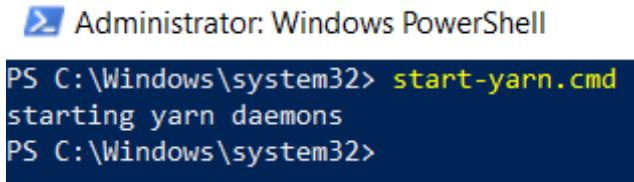
```
Apache Hadoop Distribution hadoop namenode
e been reached.
2024-07-06 17:24:06,831 INFO ipc.Server: IPC Server Responder: starting
2024-07-06 17:24:06,835 INFO ipc.Server: IPC Server listener on 9820: starting
2024-07-06 17:24:06,835 INFO namenode.NameNode: NameNode RPC up at: localhost/127.0.0.1:9820
2024-07-06 17:24:06,841 INFO namenode.FSNamesystem: Starting services required for active state
2024-07-06 17:24:06,842 INFO namenode.FSDirectory: Initializing quota with 12 thread(s)
2024-07-06 17:24:06,870 INFO namenode.FSDirectory: Quota initialization completed in 22 milliseconds
name space=437
storage space=307692594
storage types=RAM_DISK=0, SSD=0, DISK=0, ARCHIVE=0, PROVIDED=0
2024-07-06 17:24:06,899 INFO blockmanagement.CacheReplicationMonitor: Starting CacheReplicationMonitor with interval 300
00 milliseconds
2024-07-06 17:24:08,825 INFO hdfs.StateChange: BLOCK* registerDatanode: from DatanodeRegistration(127.0.0.1:9866, datanodeUuid=123a143e-536d-495e-a2ae-789991cf74d4, infoPort=9864, infoSecurePort=0, ipcPort=9867, storageInfo=lv=-57;cid=CID-0c83702c-4b94-4f10-beca-5173e24efdbc;nsid=195384883;c=1716620109937) storage 123a143e-536d-495e-a2ae-789991cf74d4
2024-07-06 17:24:08,831 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866
2024-07-06 17:24:08,832 INFO blockmanagement.BlockReportLeaseManager: Registered DN 123a143e-536d-495e-a2ae-789991cf74d4 (127.0.0.1:9866).
2024-07-06 17:24:08,998 INFO blockmanagement.DatanodeDescriptor: Adding new storage ID DS-51797c33-6f1d-4c19-80d1-e4f46981927d for DN 127.0.0.1:9866
2024-07-06 17:24:09,105 INFO BlockStateChange: BLOCK* processReport 0xc04a883d337738b6 with lease ID 0xd94992e2f8fa93da: Processing first storage report for DS-51797c33-6f1d-4c19-80d1-e4f46981927d from datanode DatanodeRegistration(127.0.0.1:9866, datanodeUuid=123a143e-536d-495e-a2ae-789991cf74d4, infoPort=9864, infoSecurePort=0, ipcPort=9867, storageInfo=lv=-57;cid=CID-0c83702c-4b94-4f10-beca-5173e24efdbc;nsid=195384883;c=1716620109937)
2024-07-06 17:24:09,117 INFO blockmanagement.BlockManager: initializing replication queues
2024-07-06 17:24:09,118 INFO hdfs.StateChange: STATE* Safe mode extension entered.
The reported blocks 299 has reached the threshold 0.9990 of total blocks 300. The minimum number of live datanodes is not required. In safe mode extension. Safe mode will be turned off automatically in 29 seconds.
2024-07-06 17:24:09,120 INFO BlockStateChange: BLOCK* processReport 0xc04a883d337738b6 with lease ID 0xd94992e2f8fa93da: from storage DS-51797c33-6f1d-4c19-80d1-e4f46981927d node DatanodeRegistration(127.0.0.1:9866, datanodeUuid=123a143e-536d-495e-a2ae-789991cf74d4, infoPort=9864, infoSecurePort=0, ipcPort=9867, storageInfo=lv=-57;cid=CID-0c83702c-4b94-4f10-beca-5173e24efdbc;nsid=195384883;c=1716620109937), blocks: 300, hasStaleStorage: false, processing time: 15 msecs, invalidatedBlocks: 0
2024-07-06 17:24:09,129 INFO blockmanagement.BlockManager: Total number of blocks = 300
2024-07-06 17:24:09,129 INFO blockmanagement.BlockManager: Number of invalid blocks = 0
2024-07-06 17:24:09,129 INFO blockmanagement.BlockManager: Number of under-replicated blocks = 0
2024-07-06 17:24:09,130 INFO blockmanagement.BlockManager: Number of over-replicated blocks = 0
2024-07-06 17:24:09,130 INFO blockmanagement.BlockManager: Number of blocks being written = 0
2024-07-06 17:24:09,131 INFO hdfs.StateChange: STATE* Replication Queue initialization scan for invalid, over- and under-replicated blocks completed in 13 msec
2024-07-06 17:24:29,244 INFO hdfs.StateChange: STATE* Safe mode ON, in safe mode extension.
The reported blocks 300 has reached the threshold 0.9990 of total blocks 300. The minimum number of live datanodes is not required. In safe mode extension. Safe mode will be turned off automatically in 9 seconds.
2024-07-06 17:24:39,354 INFO hdfs.StateChange: STATE* Safe mode is OFF
2024-07-06 17:24:39,355 INFO hdfs.StateChange: STATE* Leaving safe mode after 32 secs
2024-07-06 17:24:39,355 INFO hdfs.StateChange: STATE* Network topology has 1 racks and 1 datanodes
2024-07-06 17:24:39,356 INFO hdfs.StateChange: STATE* UnderReplicatedBlocks has 0 blocks
```

```
Apache Hadoop Distribution -hadoop datanode
2024-07-06 17:24:08,512 INFO impl.FsDatasetImpl: Time to add replicas to map for block pool BP-1739737951-16620109937 on volume D:\ProgramFiles\Hadoop\hadoop-3.3.6\data\dfs\datanode: 75ms
2024-07-06 17:24:08,519 INFO impl.FsDatasetImpl: Total time to add all replicas to map for block pool BP-1739737951-192.-1716620109937: 85ms
2024-07-06 17:24:08,525 INFO checker.ThrottledAsyncChecker: Scheduling a check for D:\ProgramFiles\Hadoop\hadoop-3.3.6\data\dfs\datanode
2024-07-06 17:24:08,545 INFO checker.DatasetVolumeChecker: Scheduled health check for volume D:\ProgramFiles\Hadoop\hadoop-3.3.6\data\dfs\datanode
2024-07-06 17:24:08,585 INFO datanode.VolumeScanner: VolumeScanner(D:\ProgramFiles\Hadoop\hadoop-3.3.6\data\dfs\datanode, DS-51797c33-6f1d-4c19-80d1-e4f46981927d): no suitable block pools found to scan. Waiting 187476211 ms.
2024-07-06 17:24:08,606 WARN datanode.DirectoryScanner: dfs.datanode.directoryscan.throttle.limit.ms.per.sec set to value above 1000 ms/sec. Assuming default value of -1
2024-07-06 17:24:08,607 INFO datanode.DirectoryScanner: Periodic Directory Tree Verification scan starting in 9389465ms with interval of 2160000ms and throttle limit of -1ms/s
2024-07-06 17:24:08,628 INFO datanode.DataNode: Block pool BP-1739737951-15-1716620109937 (Datanode Uuid 123a143e-536d-495e-a2ae-789991cf74d4) service to localhost/127.0.0.1:9820 beginning handshake with NN
2024-07-06 17:24:08,865 INFO datanode.DataNode: Block pool BP-1739737951-15-1716620109937 (Datanode Uuid 123a143e-536d-495e-a2ae-789991cf74d4) service to localhost/127.0.0.1:9820 successfully registered with NN
2024-07-06 17:24:08,867 INFO datanode.DataNode: For namenode localhost/127.0.0.1:9820 using BLOCKREPORT_INTERVAL of 2160000mses CACHEREPORT_INTERVAL of 10000mses Initial delay: 0msecs; heartBeatInterval=3000
2024-07-06 17:24:08,867 INFO datanode.DataNode: Starting IBR Task Handler.
2024-07-06 17:24:09,046 INFO datanode.DataNode: After receiving heartbeat response, updating state of namenode localhost:9820 to active
2024-07-06 17:24:09,184 INFO datanode.DataNode: Successfully sent block report 0xc04a883d337738b6 with lease ID 0xd94992e2f8fa93da to namenode: localhost/127.0.0.1:9820 containing 1 storage report(s), of which we sent 1. The reports had 300 total blocks and used 1 RPC(s). This took 21 msecs to generate and 115 msecs for RPC and NN processing. Got back one command: FinalizeCommand/5.
2024-07-06 17:24:09,185 INFO datanode.DataNode: Got finalize command for block pool BP-1739737951-15-1716620109937
```

4.6.2. Start Hadoop YARN:

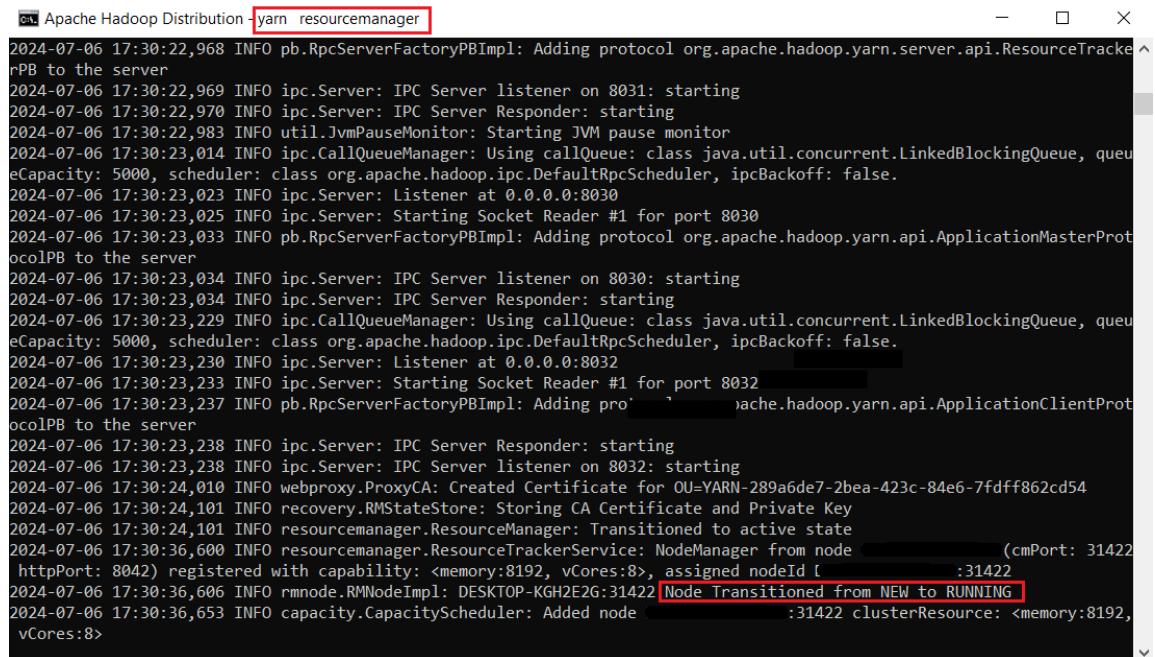
Next, start the Hadoop YARN services using the following command

```
start-yarn.cmd
```



```
Administrator: Windows PowerShell
PS C:\Windows\system32> start-yarn.cmd
starting yarn daemons
PS C:\Windows\system32>
```

After executing the above command, it opens up two command prompt windows - one for the **resourcemanager** and other for the **nodemanager** as below. Wait until **resourcemanager** service says "*Transitioned to active state*" and **nodemanager** service says "*Registered with ResourceManager*"



```
Apache Hadoop Distribution - yarn resourcemanager
2024-07-06 17:30:22,968 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.server.api.ResourceTrackerPB to the server
2024-07-06 17:30:22,969 INFO ipc.Server: IPC Server listener on 8031: starting
2024-07-06 17:30:22,970 INFO ipc.Server: IPC Server Responder: starting
2024-07-06 17:30:22,983 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2024-07-06 17:30:23,014 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlockingQueue, queueCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2024-07-06 17:30:23,023 INFO ipc.Server: Listener at 0.0.0.0:8030
2024-07-06 17:30:23,025 INFO ipc.Server: Starting Socket Reader #1 for port 8030
2024-07-06 17:30:23,033 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.api.ApplicationMasterProtocolPB to the server
2024-07-06 17:30:23,034 INFO ipc.Server: IPC Server listener on 8030: starting
2024-07-06 17:30:23,034 INFO ipc.Server: IPC Server Responder: starting
2024-07-06 17:30:23,229 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlockingQueue, queueCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2024-07-06 17:30:23,230 INFO ipc.Server: Listener at 0.0.0.0:8032
2024-07-06 17:30:23,233 INFO ipc.Server: Starting Socket Reader #1 for port 8032
2024-07-06 17:30:23,237 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.api.ApplicationClientProtocolPB to the server
2024-07-06 17:30:23,238 INFO ipc.Server: IPC Server Responder: starting
2024-07-06 17:30:23,238 INFO ipc.Server: IPC Server listener on 8032: starting
2024-07-06 17:30:24,010 INFO webproxy.ProxyCA: Created Certificate for OU=YARN-289a6de7-2bea-423c-84e6-7fdff862cd54
2024-07-06 17:30:24,101 INFO recovery.RMStateStore: Storing CA Certificate and Private Key
2024-07-06 17:30:24,101 INFO resourcemanager.ResourceManager: Transitioned to active state
2024-07-06 17:30:36,600 INFO resourcemanager.ResourceTrackerService: NodeManager from node [cmPort: 31422 httpPort: 8042] registered with capability: <memory:8192, vCores:8>, assigned nodeId [cmPort: 31422 httpPort: 8042]
2024-07-06 17:30:36,606 INFO rmnode.RMNodeImpl: DESKTOP-KGH2E2G:31422 Node Transitioned from NEW to RUNNING
2024-07-06 17:30:36,653 INFO capacity.CapacityScheduler: Added node [cmPort: 31422 httpPort: 8042] clusterResource: <memory:8192, vCores:8>
```

```
Apache Hadoop Distribution yarn nodemanager
Jul 06, 2024 5:30:30 PM com.sun.jersey.spi.container.GuiceComponentProviderFactory register
INFO: Registering org.apache.hadoop.yarn.server.nodemanager.webapp.JAXBContextResolver as a provider class
Jul 06, 2024 5:30:30 PM com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
INFO: Initiating Jersey application, version 'Jersey: 1.19.4 05/24/2017 03:20 PM'
Jul 06, 2024 5:30:30 PM com.sun.jersey.spi.container.GuiceComponentProviderFactory getComponentProvider
INFO: Binding org.apache.hadoop.yarn.server.nodemanager.webapp.JAXBContextResolver to GuiceManagedComponentProvider with the scope "Singleton"
Jul 06, 2024 5:30:30 PM com.sun.jersey.spi.container.GuiceComponentProviderFactory getComponentProvider
INFO: Binding org.apache.hadoop.yarn.webapp.GenericExceptionHandler to GuiceManagedComponentProvider with the scope "Singleton"
Jul 06, 2024 5:30:31 PM com.sun.jersey.spi.container.GuiceComponentProviderFactory getComponentProvider
INFO: Binding org.apache.hadoop.yarn.server.nodemanager.webapp.NMWebServices to GuiceManagedComponentProvider with the scope "Singleton"
2024-07-06 17:30:31,366 INFO handler.ContextHandler: Started o.e.j.w.WebAppContext@2a4f5433{node/,file:///C:/Users/hp/AppData/Local/Temp/jetty-0_0_0-8042-hadoop-yarn-common-3.3.6_jar-_any-1375801996928875865/webapp/,AVAILABLE}{jar:file:/D:/ProgramFiles/Hadoop/hadoop-3.3.6/share/hadoop/yarn/hadoop-yarn-common-3.3.6.jar!/webapps/node}
2024-07-06 17:30:31,379 INFO server.AbstractConnector: Started ServerConnector@263f04ca{HTTP/1.1, (http/1.1)}{0.0.0.0:8042}
2024-07-06 17:30:31,380 INFO server.Server: Started @25952ms
2024-07-06 17:30:31,380 INFO webapp.WebApps: Web app node started at 8042
2024-07-06 17:30:31,382 INFO nodemanager.NodeStatusUpdaterImpl: Node ID assigned is : :31422
2024-07-06 17:30:31,385 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2024-07-06 17:30:31,395 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8031
2024-07-06 17:30:36,644 INFO security.NMContainerTokenSecretManager: Rolling master-key for container-tokens, got key with id -266123525
2024-07-06 17:30:36,646 INFO security.NMTokenSecretManagerInNM: Rolling master-key for container-tokens, got key with id -620049725
2024-07-06 17:30:36,647 INFO nodemanager.NodeStatusUpdaterImpl: Registered with ResourceManager as :31422 with total resource of <memory:8192, vCores:8>
```

After Hadoop services are started, verify if NameNode, DataNode, ResourceManager and NodeManager services are running by executing the following command

```
jps
```

```
Command Prompt
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>jps
11904 Worker
16640 Jps
3540 Master
6628 ResourceManager
11976 NodeManager
7608 DataNode
14476 NameNode

C:\Users\hp>
```

4.7. Set Spark YARN Directory:

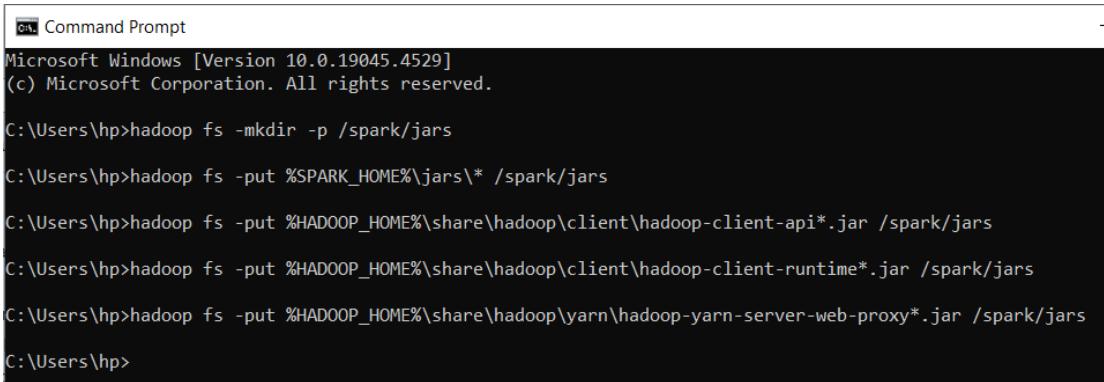
To run Spark applications on YARN cluster in Windows, we need to copy Spark jar files into HDFS and set `spark.yarn.jars` property in `spark-defaults.conf` file.

First, run the following commands to create a HDFS directory `/spark/jars` and copy Spark jar files and the necessary Hadoop jar files into HDFS directory.

```

hadoop fs -mkdir -p /spark/jars
hadoop fs -put %SPARK_HOME%\jars\* /spark/jars
hadoop fs -put %HADOOP_HOME%\share\hadoop\client\hadoop-client-
api*.jar /spark/jars
hadoop fs -put %HADOOP_HOME%\share\hadoop\client\hadoop-client-
runtime*.jar /spark/jars
hadoop fs -put %HADOOP_HOME%\share\hadoop\yarn\hadoop-yarn-server-
web-proxy*.jar /spark/jars

```



```

C:\ Command Prompt
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

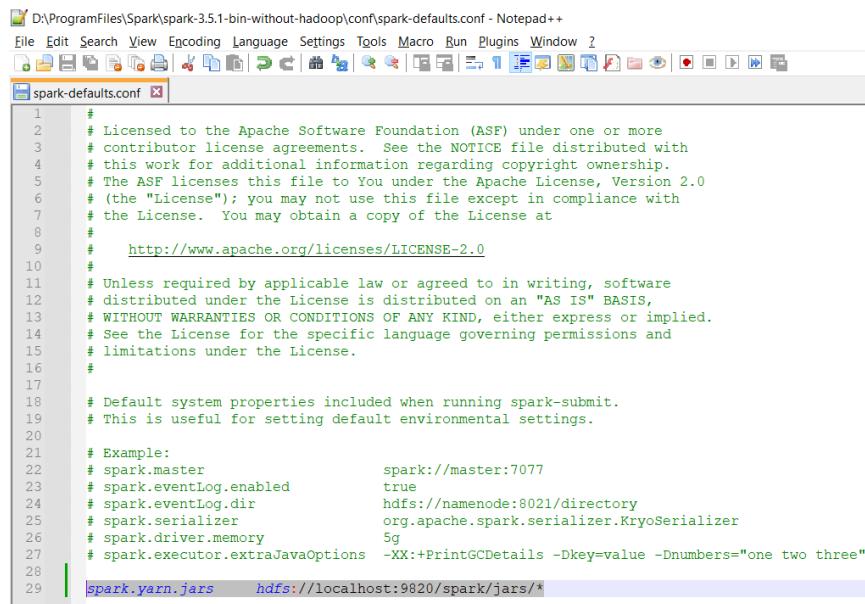
C:\Users\hp>hadoop fs -mkdir -p /spark/jars
C:\Users\hp>hadoop fs -put %SPARK_HOME%\jars\* /spark/jars
C:\Users\hp>hadoop fs -put %HADOOP_HOME%\share\hadoop\client\hadoop-client-api*.jar /spark/jars
C:\Users\hp>hadoop fs -put %HADOOP_HOME%\share\hadoop\client\hadoop-client-runtime*.jar /spark/jars
C:\Users\hp>hadoop fs -put %HADOOP_HOME%\share\hadoop\yarn\hadoop-yarn-server-web-proxy*.jar /spark/jars
C:\Users\hp>

```

Go to SPARK_HOME\conf directory, take the copy of spark-defaults.conf.template and rename it as spark-defaults.conf.

Add the following line in spark-defaults.conf file:

```
spark.yarn.jars          hdfs://localhost:9820/spark/jars/*
```



```

D:\ProgramFiles\Spark\spark-3.5.1-bin-without-hadoop\conf\spark-defaults.conf - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
spark-defaults.conf

1   #
2   # Licensed to the Apache Software Foundation (ASF) under one or more
3   # contributor license agreements. See the NOTICE file distributed with
4   # this work for additional information regarding copyright ownership.
5   # The ASF licenses this file to You under the Apache License, Version 2.0
6   # (the "License"); you may not use this file except in compliance with
7   # the License. You may obtain a copy of the License at
8   #
9   #     http://www.apache.org/licenses/LICENSE-2.0
10  #
11  # Unless required by applicable law or agreed to in writing, software
12  # distributed under the License is distributed on an "AS IS" BASIS,
13  # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14  # See the License for the specific language governing permissions and
15  # limitations under the License.
16  #
17  #
18  # Default system properties included when running spark-submit.
19  # This is useful for setting default environmental settings.
20  #
21  #
22  # Example:
23  # spark.master          spark://master:7077
24  # spark.eventLog.enabled true
25  # spark.eventLog.dir    hdfs://namenode:8021/directory
26  # spark.serializer      org.apache.spark.serializer.KryoSerializer
27  # spark.driver.memory   5g
28  # spark.executor.extraJavaOptions -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"
29  spark.yarn.jars        hdfs://localhost:9820/spark/jars/*

```

4.8. Spark Examples:

Spark provides some in-built example programs such as pi, wordcount, sort, etc. available under SPARK_HOME\examples directory that can be execute on Spark cluster.

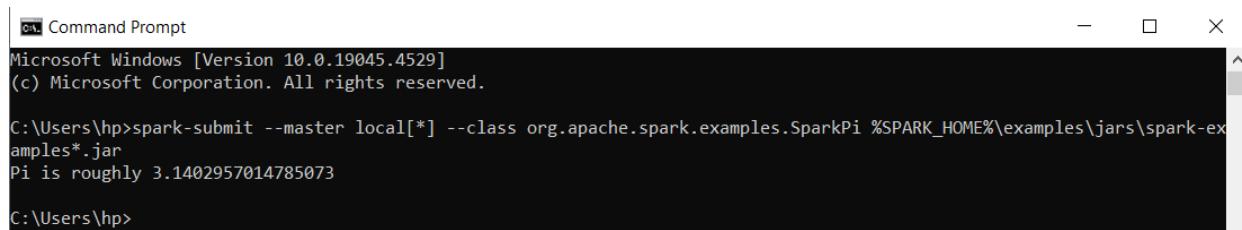
4.8.1. Run Java SparkPi Program:

Let us execute the example Java program SparkPi packaged under spark-examples*.jar file that is located in SPARK_HOME\examples\jars directory.

In Local Mode:

Open a new **Command Prompt** and run the below command which executes SparkPi application in local mode with all available cores:

```
spark-submit --class org.apache.spark.examples.SparkPi  
%SPARK_HOME%\examples\jars\spark-examples*.jar  
or  
spark-submit --master local[*] --class  
org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-  
examples*.jar
```



```
Command Prompt  
Microsoft Windows [Version 10.0.19045.4529]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\hp>spark-submit --master local[*] --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar  
Pi is roughly 3.1402957014785073  
  
C:\Users\hp>
```

It displayed the output of approximate Pi value which is **3.140**.

In YARN Client Mode:

Use the following command to run the SparkPi application in client mode on YARN cluster:

```
spark-submit --master yarn --deploy-mode client --class  
org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-  
examples*.jar  
or  
spark-submit --master yarn --class org.apache.spark.examples.SparkPi  
%SPARK_HOME%\examples\jars\spark-examples*.jar
```

```

Command Prompt
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-submit --master yarn --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
Pi is roughly 3.1409357046785233

C:\Users\hp>

```

On the YARN UI Web UI (<http://localhost:8088/cluster>), we can see that **Spark Pi** application has been submitted and completed.

The screenshot shows the Hadoop YARN UI at <http://localhost:8088/cluster>. The main page displays cluster metrics, node metrics, and scheduler metrics. On the right, there is a table titled "All Applications" showing the status of submitted applications. A single row is highlighted with a red border, representing the "Spark Pi" application. The table columns include ID, User, Name, Application Type, Application Tags, Queue, Application Priority, Start Time, Launch Time, Finish Time, State, and Final Status. The application listed is "application_1720267222503_0001" submitted by "hp" and named "Spark Pi". It was run with "SPARK" tags, placed in the "default" queue, and had priority 0. It started on Saturday, July 6, 2024, at 17:45:02 and finished at 17:46:33, both on +0550 (2024). The state is "FINISHED" and the final status is "SUCCEEDED".

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	Start Time	Launch Time	Finish Time	State	Final Status
application_1720267222503_0001	hp	Spark Pi	SPARK		default	0	Sat Jul 6 17:45:02 +0550 2024	Sat Jul 6 17:45:09 +0550 2024	Sat Jul 6 17:46:33 +0550 2024	FINISHED	SUCCEEDED

In YARN Cluster Mode:

To run the SparkPi application in cluster mode on YARN, use the below command:

```
spark-submit --master yarn --deploy-mode cluster --class
org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-
examples*.jar
```

```

Command Prompt
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-submit --master yarn --deploy-mode cluster --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
17:49:33.007 [main] ERROR org.apache.spark.deploy.yarn.Client - Application diagnostics message: User class threw exception: java.lang.ExceptionInInitializerError
        at org.apache.spark.SparkContext.withScope(SparkContext.scala:924)
        at org.apache.spark.SparkContext.parallelize(SparkContext.scala:941)
        at org.apache.spark.examples.SparkPi$.main(SparkPi.scala:34)
        at org.apache.spark.examples.SparkPi.main(SparkPi.scala)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.spark.deploy.yarn.ApplicationMaster$$anon$2.run(ApplicationMaster.scala:738)
Caused by: com.fasterxml.jackson.databind.JsonMappingException: Scala module 2.15.2 requires Jackson Databind version >= 2.15.0 and < 2.16.0 - Found jackson-databind version 2.12.7-1
        at com.fasterxml.jackson.module.scala.JacksonModule.setupModule(JacksonModule.scala:61)
        at com.fasterxml.jackson.module.scala.JacksonModule.setupModule$(JacksonModule.scala:46)
        at com.fasterxml.jackson.module.scala.DefaultScalaModule.setupModule(DefaultScalaModule.scala:17)
        at com.fasterxml.jackson.module.scala.ObjectMapper.registerModule(ObjectMapper.java:835)
        at org.apache.spark.rdd.RDDOperationScope$.<init>(RDDOperationScope.scala:82)
        at org.apache.spark.rdd.RDDOperationScope$.<clinit>(RDDOperationScope.scala)
        ... 9 more

Exception in thread "main" org.apache.spark.SparkException: Application application_1720267222503_0002 finished with failed status
        at org.apache.spark.deploy.yarn.Client.run(Client.scala:1309)
        at org.apache.spark.deploy.yarn.YarnClusterApplication.start(Client.scala:1742)
        at org.apache.spark.deploy.SparkSubmit.org$apache$spark$deploy$SparkSubmit$$runMain(SparkSubmit.scala:1029)
        at org.apache.spark.deploy.SparkSubmit.doRunMain$1(SparkSubmit.scala:194)
        at org.apache.spark.deploy.SparkSubmit.submit(SparkSubmit.scala:217)
        at org.apache.spark.deploy.SparkSubmit.doSubmit(SparkSubmit.scala:91)
        at org.apache.spark.deploy.SparkSubmit$$anon$2.doSubmit(SparkSubmit.scala:1120)
        at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:1129)
        at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)

C:\Users\hp>

```

Here, we can see an error "**Scala module 2.15.2 requires Jackson Databind version >= 2.15.0 and < 2.16.0 - Found jackson-databind version 2.12.7-1**" due to the conflict between Hadoop JARs and Spark JARs. Note that this error occurs only in Windows operating system.

Follow the below steps to resolve the above error:

- Download the latest `slf4j-api-2.0.7.jar` file from <https://mvnrepository.com/artifact/org.slf4j/slf4j-api/2.0.7>
- Place `slf4j-api-2.0.7.jar` file in `%SPARK_HOME%\jars` directory.

This PC > Local Disk (D:) > ProgramFiles > Spark > spark-3.5.1-bin-without-hadoop > jars				
Name	Date modified	Type	Size	
scala-library_2.12.18.jar		Executable Jar File	1,021 KB	
scala-parser-combinators_2.12-2.3.0.jar	2/15/2024 4:54 PM	Executable Jar File	183 KB	
scala-reflect-2.12.18.jar	2/15/2024 4:54 PM	Executable Jar File	3,585 KB	
scala-xml_2.12-2.1.0.jar	2/15/2024 4:54 PM	Executable Jar File	456 KB	
shims-0.9.45.jar	2/15/2024 4:54 PM	Executable Jar File	3 KB	
slf4j-api-2.0.7.jar	7/6/2024 5:53 PM	Executable Jar File	63 KB	
snakeyaml-2.0.jar	2/15/2024 4:54 PM	Executable Jar File	327 KB	
snakeyaml-engine-2.6.jar	2/15/2024 4:54 PM	Executable Jar File	286 KB	
snappy-java-1.1.10.3.jar	2/15/2024 4:54 PM	Executable Jar File	2,011 KB	

- Copy `slf4j-api-2.0.7.jar` file from `%SPARK_HOME%\jars` to HDFS location where Spark jars are available using the following command:

```
hadoop fs -put %SPARK_HOME%\jars\slf4j-api-2.0.7.jar /spark/jars
```

Command Prompt

```
C:\Users\hp>hadoop fs -put %SPARK_HOME%\jars\slf4j-api-2.0.7.jar /spark/jars
C:\Users\hp>
```

- Copy `hadoop-client-api-*.jar` and `hadoop-client-runtime-*/*.jar` files from `%HADOOP_HOME%\share\hadoop\client` directory to `%SPARK_HOME%\jars` directory in your local system:

```
copy %HADOOP_HOME%\share\hadoop\client\hadoop-client-api-*.* %SPARK_HOME%\jars
copy %HADOOP_HOME%\share\hadoop\client\hadoop-client-runtime-*.* %SPARK_HOME%\jars
```

Command Prompt

```
C:\Users\hp>copy %HADOOP_HOME%\share\hadoop\client\hadoop-client-api-3.3.6.jar %SPARK_HOME%\jars
1 file(s) copied.

C:\Users\hp>copy %HADOOP_HOME%\share\hadoop\client\hadoop-client-runtime-3.3.6.jar %SPARK_HOME%\jars
1 file(s) copied.

C:\Users\hp>
```

- Remove the following lines in `spark-env.cmd` file.

```
for /f %%a IN ('%HADOOP% classpath') do (
    set SPARK_DIST_CLASSPATH=%%a
)
```

```

D:\ProgramFiles\Spark\spark-3.5.1-bin-without-hadoop\conf\spark-env.cmd - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
spark-env.cmd
1 @echo off
2
3 @rem verify HADOOP_HOME env variable as a prereq
4 if not defined HADOOP_HOME (
5     echo "HADOOP_HOME needs to be defined to point at the hadoop installation"
6     exit /b 1
7 )
8
9 @rem set Hadoop config directory
10 set HADOOP_CONF_DIR=%HADOOP_HOME%\etc\hadoop
11
12 @rem need to use existing Hadoop's classpath
13 set HADOOP=%HADOOP_HOME%\bin\hadoop.cmd
14 if not exist %HADOOP% (
15     echo "Missing hadoop installation: %HADOOP_HOME%\bin\hadoop.cmd must exist"
16     exit /b 1
17 )
18
19
20

```

Now, run the SparkPi application in cluster mode on YARN using the below command:

```

spark-submit --master yarn --deploy-mode cluster --class
org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-
examples*.jar

```

```

Command Prompt
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-submit --master yarn --deploy-mode cluster --class org.apache.spark.examples.SparkPi %SPARK_HOME%\examples\jars\spark-examples*.jar
24/07/06 17:58:42 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
24/07/06 17:58:45 INFO Configuration: resource-types.xml not found
24/07/06 17:58:45 INFO ResourceUtils: Unable to find 'resource-types.xml'.
24/07/06 17:58:45 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (8192 MB per container)
24/07/06 17:58:45 INFO Client: Will allocate AM container, with 1408 MB memory including 384 MB overhead
24/07/06 17:58:45 INFO Client: Setting up container launch context for our AM
24/07/06 17:58:45 INFO Client: Setting up the launch environment for our AM container
24/07/06 17:58:45 INFO Client: Preparing resources for our AM container
24/07/06 17:58:46 INFO Client: Source and destination file systems are the same. Not copying hdfs://localhost:9820/spark/jars/JLargeArrays-1.5.jar
24/07/06 17:58:46 INFO Client: Source and destination file systems are the same. Not copying hdfs://localhost:9820/spark/jars/JTransforms-3.1.jar
24/07/06 17:58:46 INFO Client: Source and destination file systems are the same. Not copying hdfs://localhost:9820/spark/jars/RoaringBitmap-0.9.45.jar
24/07/06 17:58:46 INFO Client: Source and destination file systems are the same. Not copying hdfs://localhost:9820/spark/jars/aircompressor-0.26.jar
24/07/06 17:58:46 INFO Client: Source and destination file systems are the same. Not copying hdfs://localhost:9820/spark/jars/algebra_2.12-2.0.1.jar
24/07/06 17:58:46 INFO Client: Source and destination file systems are the same. Not copying hdfs://localhost:9820/spark/jars/annotations-17.0.0.jar
24/07/06 17:58:46 INFO Client: Source and destination file systems are the same. Not copying hdfs://localhost:9820/spark/jars/antlr4-runtime-4.9.3.jar
24/07/06 17:58:46 INFO Client: Source and destination file systems are the same. Not copying hdfs://localhost:9820/spark/jars/aopalliance-repackaged-2.6.1.jar

```

```

24/07/06 17:59:03 INFO Client: Application report for application_1720267222503_0003 (state: RUNNING)
24/07/06 17:59:03 INFO Client:
    client token: N/A
    diagnostics: N/A
    ApplicationMaster host: DESKTOP-KGH2E2G
    ApplicationMaster RPC port: 26347
    queue: default
    start time: 1720268928686
    final status: UNDEFINED
    tracking URL: http://DESKTOP-KGH2E2G:8088/proxy/application_1720267222503_0003/
    user: hp
24/07/06 17:59:33 INFO Client: Application report for application_1720267222503_0003 (state: RUNNING)
24/07/06 17:59:34 INFO Client: Application report for application_1720267222503_0003 (state: FINISHED)
24/07/06 17:59:34 INFO Client:
    client token: N/A
    diagnostics: N/A
    ApplicationMaster host: DESKTOP-KGH2E2G
    ApplicationMaster RPC port: 26347
    queue: default
    start time: 1720268928686
    final status: SUCCEEDED
    tracking URL: http://DESKTOP-KGH2E2G:8088/proxy/application_1720267222503_0003/
    user: hp
24/07/06 17:59:34 INFO ShutdownHookManager: Shutdown hook called
24/07/06 17:59:34 INFO ShutdownHookManager: Deleting directory C:\Users\hp\AppData\Local\Temp\spark-6ed7031b-4749-4ee7-b8f2-a1278d7923ce
24/07/06 17:59:34 INFO ShutdownHookManager: Deleting directory C:\Users\hp\AppData\Local\Temp\spark-536cd9ef-ad37-4409-a589-c949acdbccbb
C:\Users\hp>

```

Here, we can see that application has been submitted to YARN and FINISHED successfully. Since we executed the Spark application in cluster mode on YARN, the output of application can be seen in YARN application itself.

Open YARN UI: <http://localhost:8088/cluster> and click on the latest **Application ID** value.

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime
application_1720267222503_0003	hp	org.apache.spark.examples.SparkPi	SPARK		default	0	Sat Jul 6 17:58:48 +0550 2024	Sat Jul 6 17:58:49 +0550 2024	Sat Jul 6 17:59:34 +0550 2024
application_1720267222503_0002	hp	org.apache.spark.examples.SparkPi	SPARK		default	0	Sat Jul 6 17:48:27 +0550 2024	Sat Jul 6 17:48:27 +0550 2024	Sat Jul 6 17:49:32 +0550 2024
application_1720267222503_0001	hp	Spark Pi	SPARK		default	0	Sat Jul 6 17:45:02 +0550 2024	Sat Jul 6 17:45:09 +0550 2024	Sat Jul 6 17:46:33 +0550 2024

On the respective application details, click on **Logs** link which opens container logs.

The screenshot shows the Hadoop Application Overview page for application_1720267222503_0003. The left sidebar has sections for Cluster (About, Nodes, Node Labels, Applications, Scheduler) and Tools. The main content area displays application details:

User:	bp
Name:	org.apache.spark.examples.SparkPi
Application Type:	SPARK
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Sat Jul 06 17:58:48 +0530 2024
Launched:	Sat Jul 06 17:58:49 +0530 2024
Finished:	Sat Jul 06 17:59:34 +0530 2024
Elapsed:	45sec
Tracking URL:	History
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

Below this is the Application Metrics section:

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	214974 MB-seconds, 104 vcore-seconds
Aggregate Preempted Resource Allocation:	0 MB-seconds, 0 vcore-seconds

At the bottom is a table showing attempt details:

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1720267222503_0003_000001	Sat Jul 6 17:58:48 +0550 2024	http://DESKTOP-KGH2E2G:8042	Logs	0	0

Showing 1 to 1 of 1 entries

On the container logs, click on **stdout** link to see the output

The screenshot shows the Hadoop Local Logs page for container_1720267222503_0003_01_000001. The left sidebar has sections for ResourceManager (RM Home), NodeManager, and Tools. The main content area displays local logs:

```
directory.info : Total file length is 1231 bytes.  
launch_container.cmd : Total file length is 71137 bytes.  
stderr : Total file length is 62015 bytes.  
stdout : Total file length is 34 bytes.
```

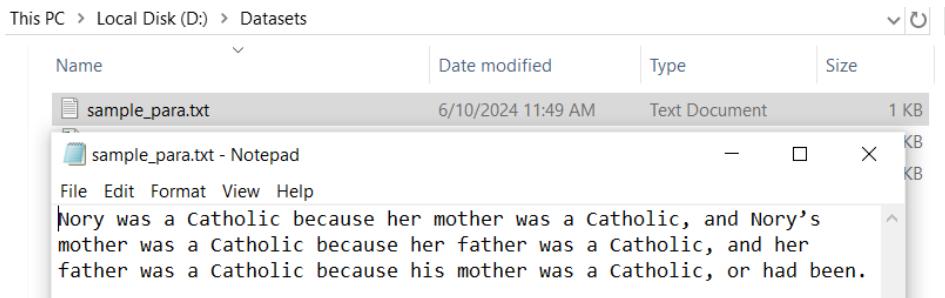
Below this is another Hadoop Local Logs page for the same container, showing the stdout log output:

```
Pi is roughly 3.1494957474787375
```

4.8.2. Run Python wordcount Program:

Let us execute the Python program `wordcount` which counts the occurrences of each word in the given input file. This program is located in `SPARK_HOME\examples\src\main\python` directory.

First, create an input file named `sample_para.txt` with some random text and place it in `D:\Datasets` folder.



Since our Spark is running on top of Hadoop cluster, we need to move the above file into HDFS using the following commands:

```
hadoop fs -mkdir /spark-input
hadoop fs -put D:\Datasets\sample_para.txt /spark-input
hadoop fs -ls /spark-input
```

```
C:\Users\hp>hadoop fs -mkdir /spark-input
C:\Users\hp>hadoop fs -put D:\Datasets\sample_para.txt /spark-input
C:\Users\hp>hadoop fs -ls /spark-input
Found 1 items
-rw-r--r-- 1 hp supergroup          202 2024-07-06 18:13 /spark-input/sample_para.txt
C:\Users\hp>
```

Note: To avoid INFO logging on the console while executing the `spark-submit` command, go to `%SPARK_HOME%\conf` location, copy `log4j2.properties.template` file as `log4j2.properties` and change `rootLogger.level` value from `info` to `warn` in `log4j2.properties` file as shown below:

```

D:\ProgramFiles\Spark\spark-3.5.1-bin-without-hadoop\conf\log4j2.properties - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
log4j2.properties x
1  #
2  # Licensed to the Apache Software Foundation (ASF) under one or more
3  # contributor license agreements. See the NOTICE file distributed with
4  # this work for additional information regarding copyright ownership.
5  # The ASF licenses this file to You under the Apache License, Version 2.0
6  # (the "License"); you may not use this file except in compliance with
7  # the License. You may obtain a copy of the License at
8  #
9  #     http://www.apache.org/licenses/LICENSE-2.0
10 #
11 # Unless required by applicable law or agreed to in writing, software
12 # distributed under the License is distributed on an "AS IS" BASIS,
13 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14 # See the License for the specific language governing permissions and
15 # limitations under the License.
16 #
17 #
18 # Set everything to be logged to the console
19 rootLogger.level = warn
20 rootLogger.appenderRef.stdout.ref = console
21

```

In Local Mode:

Open a new **Command Prompt** and run the below command which executes wordcount.py application in local mode with all available cores:

```
spark-submit --master local[*]
%SPARK_HOME%\examples\src\main\python\wordcount.py /spark-
input/sample_para.txt
```

or

```
spark-submit %SPARK_HOME%\examples\src\main\python\wordcount.py
/spark-input/sample_para.txt
```

```

C:\> Command Prompt
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-submit %SPARK_HOME%\examples\src\main\python\wordcount.py /spark-input/sample_para.txt
Nory: 1
was: 6
a: 6
Catholic: 3
because: 3
her: 3
mother: 3
Catholic,: 3
and: 2
Nory&s: 1
father: 2
his: 1
or: 1
had: 1
been.: 1

C:\Users\hp>

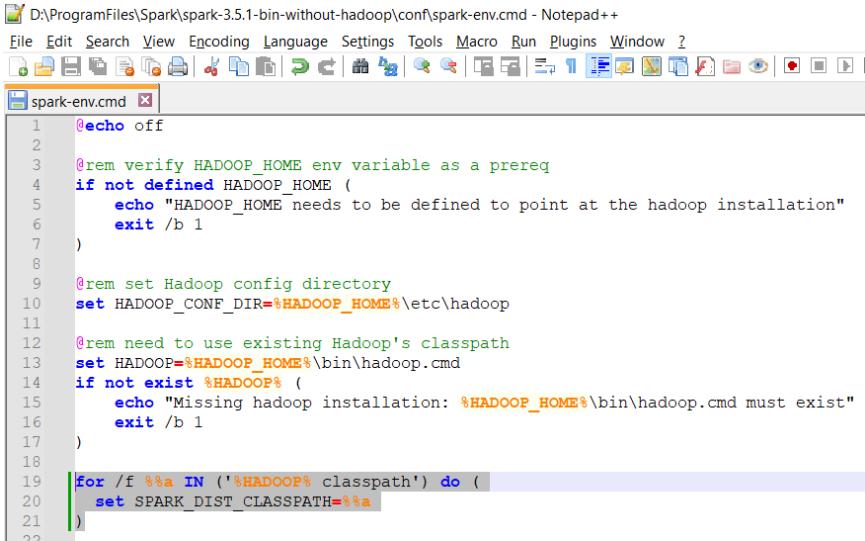
```

It displayed the count of each word in sample_para.txt file.

In YARN Client Mode:

Before executing in “YARN client” mode, make sure to add the following lines in `spark-env.cmd` file (*since we removed earlier to run in “YARN cluster” mode*)

```
for /f %%a IN ('%HADOOP% classpath') do (
    set SPARK_DIST_CLASSPATH=%%a
)
```



The screenshot shows a Notepad++ window titled "spark-env.cmd". The code inside the file is as follows:

```
1  @echo off
2
3  @rem verify HADOOP_HOME env variable as a prereq
4  if not defined HADOOP_HOME (
5      echo "HADOOP_HOME needs to be defined to point at the hadoop installation"
6      exit /b 1
7  )
8
9  @rem set Hadoop config directory
10 set HADOOP_CONF_DIR=%HADOOP_HOME%\etc\hadoop
11
12 @rem need to use existing Hadoop's classpath
13 set HADOOP=%HADOOP_HOME%\bin\hadoop.cmd
14 if not exist %HADOOP% (
15     echo "Missing hadoop installation: %HADOOP_HOME%\bin\hadoop.cmd must exist"
16     exit /b 1
17 )
18
19 for /f %%a IN ('%HADOOP% classpath') do (
20     set SPARK_DIST_CLASSPATH=%%a
21 )
22
```

Now run the following command to submit the `wordcount.py` application in client mode on YARN cluster:

```
spark-submit --master yarn --deploy-mode client
%SPARK_HOME%\examples\src\main\python\wordcount.py /spark-
input/sample_para.txt
```

or

```
spark-submit --master yarn
%SPARK_HOME%\examples\src\main\python\wordcount.py /spark-
input/sample_para.txt
```

```

C:\Users\hp>spark-submit --master yarn %SPARK_HOME%\examples\src\main\python\wordcount.py /spark-input/sample_para.txt
Nory: 1
was: 6
a: 6
Catholic: 3
because: 3
her: 3
mother: 3
Catholic,: 3
and: 2
Nory&s: 1
father: 2
his: 1
or: 1
had: 1
been.: 1

C:\Users\hp>

```

On the YARN UI Web UI (<http://localhost:8088/cluster>), we can see that **PythonWordCount** application has been submitted and completed.

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime
application_1720267222503_0005	hp	PythonWordCount	SPARK		default	0	Sat Jul 6 19:51:41 +0550 2024	Sat Jul 6 19:51:41 +0550 2024	Sat Jul 6 19:52:36 +0550 2024
application_1720267222503_0004	hp	PythonWordCount	SPARK		default	0	Sat Jul 6 18:44:40 +0550 2024	Sat Jul 6 18:44:40 +0550 2024	Sat Jul 6 18:45:59 +0550 2024
application_1720267222503_0003	hp	org.apache.spark.examples.SparkPi	SPARK		default	0	Sat Jul 6 17:58:48 +0550 2024	Sat Jul 6 17:58:49 +0550 2024	Sat Jul 6 17:59:34 +0550 2024
application_1720267222503_0002	hp	org.apache.spark.examples.SparkPi	SPARK		default	0	Sat Jul 6 17:48:27 +0550 2024	Sat Jul 6 17:48:27 +0550 2024	Sat Jul 6 17:49:32 +0550 2024
application_1720267222503_0001	hp	Spark Pi	SPARK		default	0	Sat Jul 6 17:45:02 +0550 2024	Sat Jul 6 17:45:09 +0550 2024	Sat Jul 6 17:46:33 +0550 2024

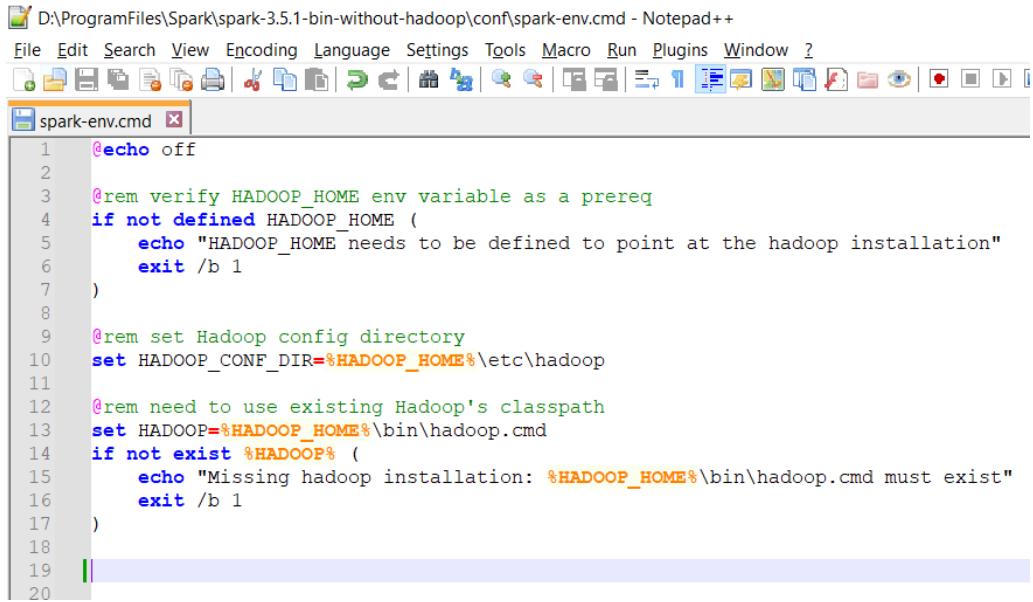
In YARN Cluster Mode:

Before executing in “YARN cluster” mode, make sure to remove the following lines in `spark-env.cmd` file.

```

for /f %%a IN ('%HADOOP% classpath') do (
    set SPARK_DIST_CLASSPATH=%%a
)

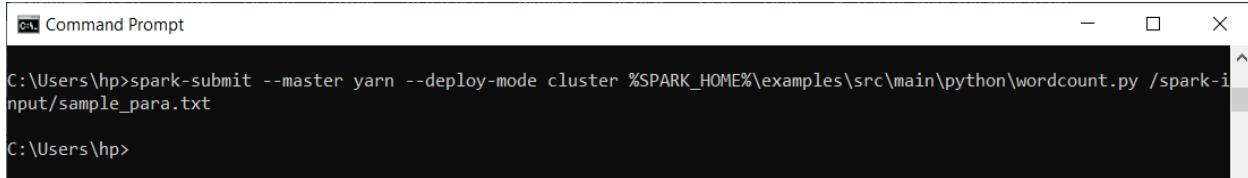
```



```
D:\ProgramFiles\Spark\spark-3.5.1-bin-without-hadoop\conf\spark-env.cmd - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
spark-env.cmd x
1 @echo off
2
3 @rem verify HADOOP_HOME env variable as a prereq
4 if not defined HADOOP_HOME (
5     echo "HADOOP_HOME needs to be defined to point at the hadoop installation"
6     exit /b 1
7 )
8
9 @rem set Hadoop config directory
10 set HADOOP_CONF_DIR=%HADOOP_HOME%\etc\hadoop
11
12 @rem need to use existing Hadoop's classpath
13 set HADOOP=%HADOOP_HOME%\bin\hadoop.cmd
14 if not exist %HADOOP% (
15     echo "Missing hadoop installation: %HADOOP_HOME%\bin\hadoop.cmd must exist"
16     exit /b 1
17 )
18
19
20
```

Now, run the below command which executes wordcount.py application in cluster mode on YARN:

```
spark-submit --master yarn --deploy-mode cluster
%SPARK_HOME%\examples\src\main\python\wordcount.py /spark-
input/sample_para.txt
```



```
Command Prompt
C:\Users\hp>spark-submit --master yarn --deploy-mode cluster %SPARK_HOME%\examples\src\main\python\wordcount.py /spark-
input/sample_para.txt
C:\Users\hp>
```

Since we executed the Spark application in cluster mode on YARN, the output of application can be seen in YARN application itself.

Open YARN UI: <http://localhost:8088/cluster> and click on the latest **Application ID** value.

The screenshot shows the Apache Hadoop Cluster Overview page at <http://localhost:8088/cluster>. The left sidebar includes links for Cluster Metrics, Cluster Nodes Metrics, Scheduler Metrics, and Tools. The main content displays Cluster Metrics with values: Apps Submitted (6), Apps Pending (0), Apps Running (6), Apps Completed (0), Containers Running (<memory:0 B, vCores:0), and Used Resources (<memory:0 B, vCores:0). Below this are sections for Cluster Nodes Metrics, Scheduler Metrics, and a table of applications. The table has columns: ID, User, Name, Application Type, Application Tags, Queue, Application Priority, Start Time, Launch Time, and Finish Time. One row, application_1720267222503_0006, is highlighted with a red border.

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	Start Time	Launch Time	Finish Time
application_1720267222503_0006	hp	wordcount.py	SPARK		default	0	Sat Jul 6 19:59:46 +0530 2024	Sat Jul 6 19:59:47 +0530 2024	Sat Jul 6 20:00:44 +0530 2024
application_1720267222503_0005	hp	PythonWordCount	SPARK		default	0	Sat Jul 6 19:51:41 +0530 2024	Sat Jul 6 19:51:41 +0530 2024	Sat Jul 6 19:52:36 +0530 2024
application_1720267222503_0004	hp	PythonWordCount	SPARK		default	0	Sat Jul 6 18:44:40 +0530 2024	Sat Jul 6 18:44:40 +0530 2024	Sat Jul 6 18:45:59 +0530 2024
application_1720267222503_0003	hp	org.apache.spark.examples.SparkPi	SPARK		default	0	Sat Jul 6 17:58:48 +0530 2024	Sat Jul 6 17:58:49 +0530 2024	Sat Jul 6 17:59:34 +0530 2024
application_1720267222503_0002	hp	org.apache.spark.examples.SparkPi	SPARK		default	0	Sat Jul 6 17:48:27 +0530 2024	Sat Jul 6 17:48:27 +0530 2024	Sat Jul 6 17:49:32 +0530 2024

On the respective application details, click on **Logs** link which opens container logs.

The screenshot shows the Application Overview page for application_1720267222503_0006 at http://localhost:8088/cluster/app/application_1720267222503_0006. The left sidebar includes links for Cluster Metrics, Cluster Nodes Metrics, Scheduler Metrics, and Tools. The main content displays Application Overview details and Application Metrics. The Application Overview section includes fields like User (hp), Name (wordcount.py), Application Type (SPARK), Application Tags, Application Priority (0), YarnApplicationState (FINISHED), Queue (default), FinalStatus Reported by AM (SUCCEEDED), Started (Sat Jul 06 19:59:46 +0530 2024), Launched (Sat Jul 06 19:59:47 +0530 2024), Finished (Sat Jul 06 20:00:44 +0530 2024), Elapsed (58sec), Tracking URL (History), Log Aggregation Status (DISABLED), Application Timeout (Remaining Time) (Unlimited), Diagnostics, Unmanaged Application (false), Application Node Label expression (<Not set>), and AM container Node Label expression (<DEFAULT_PARTITION>). The Application Metrics section shows metrics like Total Resource Preempted (<memory:0, vCores:0>), Total Number of Non-AM Containers Preempted (0), Total Number of AM Containers Preempted (0), Resource Preempted from Current Attempt (<memory:0, vCores:0>), Number of Non-AM Containers Preempted from Current Attempt (0), Aggregate Resource Allocation (297673 MB-seconds, 143 vcore-seconds), and Aggregate Preempted Resource Allocation (0 MB-seconds, 0 vcore-seconds). A table at the bottom lists container attempts with columns: Attempt ID, Started, Node, Logs, Nodes blacklisted by the app, and Nodes blacklisted by the system. The Logs column for the first attempt is highlighted with a red border.

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1720267222503_0006_000001	Sat Jul 6 19:59:46 +0530 2024	http://DESKTOP-KGH2E2G:8042	Logs	0	0

On the container logs, click on **stdout** link to see the output

Not secure http://desktop-kgh2e2g:8042/node/containerlogs/container_1720267222503_0006_01_000001/hp

hadoop

Logs for container_1720267222503_0006_01_000001

Local Logs:

- directory.info : Total file length is 1475 bytes.
- launch_container.cmd : Total file length is 71650 bytes.
- stderr : Total file length is 3164 bytes.
- stdout : Total file length is 144 bytes.

ResourceManager
RM Home

NodeManager
Tools

Not secure http://desktop-kgh2e2g:8042/node/containerlogs/container_1720267222503_0006_01_000001/hp/stdout/?start=-4096

hadoop

Logs for container_1720267222503_0006_01_000001

Navy: 1
was: 6
Catholic: 3
because: 3
her: 3
mother: 3
Catholic.: 3
and: 2
Nor: 2
she: 1
father: 2
his: 1
or: 1
had: 1
been: 1

ResourceManager
RM Home

NodeManager
Tools

We can see that it displayed the count of each word available in `sample_para.txt` file.

5. Spark CLIs:

Apache spark provides various command line interfaces such as PySpark Shell, Spark Shell, Spark SQL, Spark R, etc. to execute the spark code in various programming languages.

When any spark client such as `pyspark`, `spark-shell`, `spark-sql`, `spark-r`, etc is launched or `spark-submit` command is executed, it creates a **Spark Context Web UI** which runs on **4040** port by default. If the default port 4040 is occupied by other service, Spark tries to create the web interface on the next available port incremented by 1 (i.e. 4041 port). Even if 4041 port is also occupied by any other service, then Spark tries to open Web UI on the next available port incremented by 1.

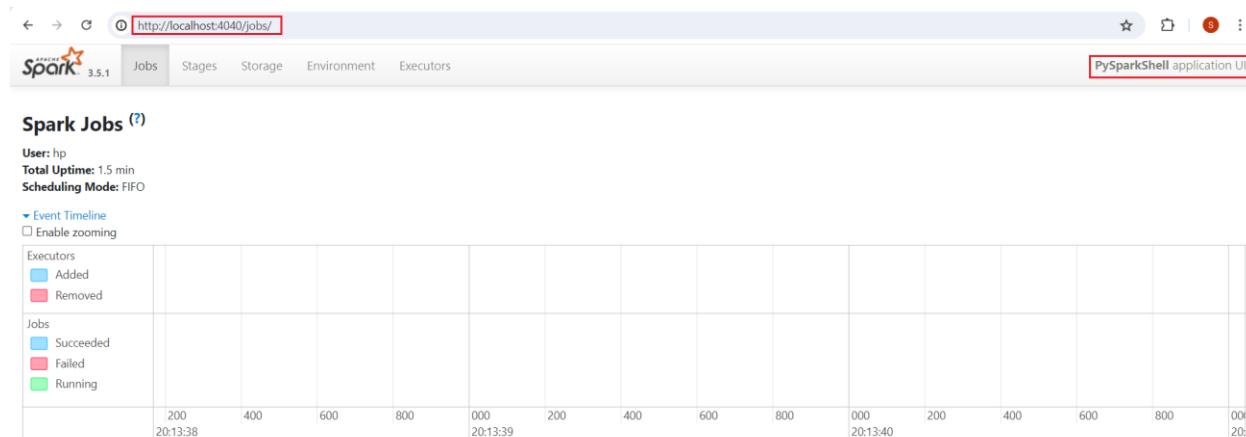
Note: Spark Web UI is opened only when a spark client is active and will be closed when exited from the spark client.

5.1. Pyspark:

Open either **Windows PowerShell** or **Command Prompt** and simply run the following command to start PySpark shell.

```
pyspark
```

PySparkShell application UI available at <http://localhost:4040/> (*I opened pyspark shell first and is active*)

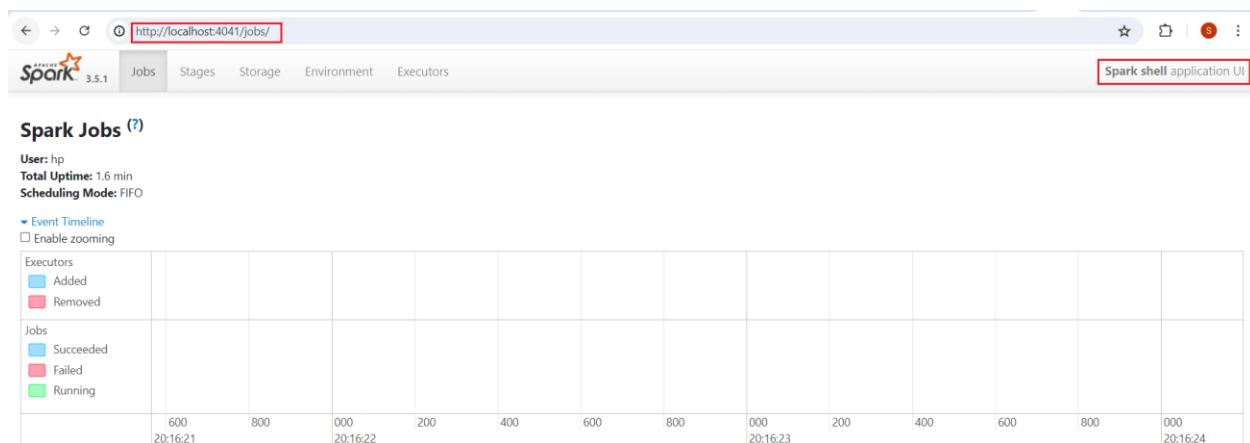


5.2. Spark Shell:

Open another **Windows PowerShell** or **Command Prompt** and run the following command to start Spark shell which opens `scala>` terminal by default.

spark-shell

Spark Shell application UI available at <http://localhost:4041/> (*I opened spark-shell while pyspark session is still active*)



5.3. Spark SQL:

Open another **Windows PowerShell** or **Command Prompt** and run the following command to start Spark SQL.

spark-sql

```
C:\ Command Prompt
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-sql
Error: Failed to load class org.apache.spark.sql.hive.thriftserver.SparkSQLCLIDriver.
Failed to load main class org.apache.spark.sql.hive.thriftserver.SparkSQLCLIDriver.
You need to build Spark with -Phive and -Phive-thriftserver.
24/07/06 20:23:55 INFO ShutdownHookManager: Shutdown hook called
24/07/06 20:23:55 INFO ShutdownHookManager: Deleting directory C:\Users\hp\AppData\Local\Temp\spark-c692a91b-b5ea-4a91-b
2e3-adccbbe7f1c6

C:\Users\hp>
```

Here, we are encountering "*Error: Failed to load class org.apache.spark.sql.hive.thriftserver.SparkSQLCLIDriver*" which occurs when we are running Spark version without built-in Hadoop libraries which do not include **Spark Hive** and **Spark Hive Thrift Server** and other dependent packages.

To resolve this issue, we need to either copy all files from jars folder in spark-3.5.1-bin-hadoop3 location to the current %SPARK_HOME%\jars location or set the SPARK_HOME environment variable to the location where spark-3.5.1-bin-hadoop3 version was installed.

Here, I am setting SPARK_HOME environment variable to the location where my spark-3.5.1-bin-hadoop3 version was installed and then launching spark-sql using below commands:

```
set SPARK_HOME=D:\ProgramFiles\Spark\spark-3.5.1-bin-hadoop3
```

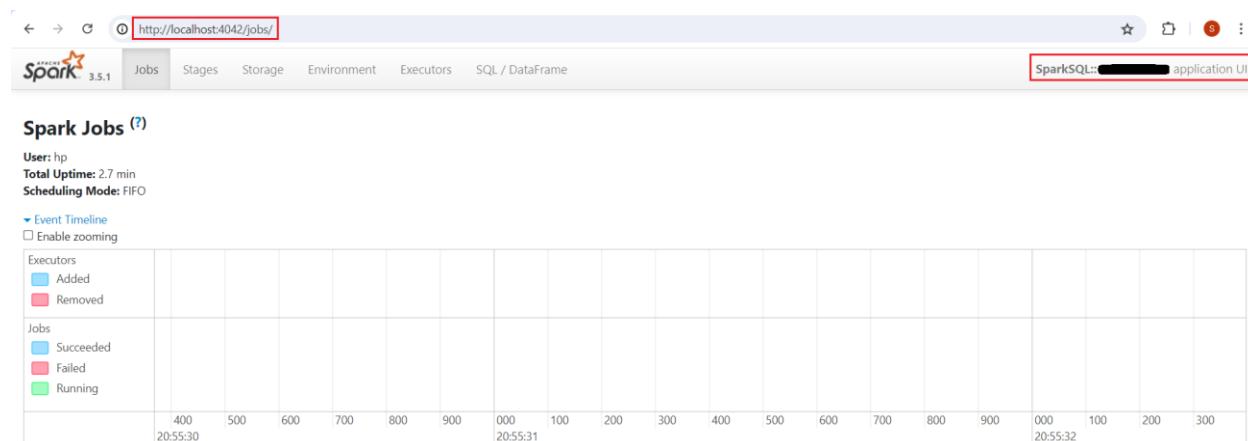
```
spark-sql
```

```
C:\ Command Prompt - spark-sql
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>set SPARK_HOME=D:\ProgramFiles\Spark\spark-3.5.1-bin-hadoop3

C:\Users\hp>spark-sql
24/07/06 20:55:32 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
24/07/06 20:55:32 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
24/07/06 20:55:34 WARN HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist
24/07/06 20:55:34 WARN HiveConf: HiveConf of name hive.stats.retries.wait does not exist
Spark Web UI available at http://*:4042
Spark master: local[*], Application Id: local-1720279532308
spark-sql (default)>
```

Spark SQL application UI available at <http://localhost:4042/> (I opened spark-sql while pyspark and spark-shell sessions are still active)



6. Spark Web UI:

Apache Spark provides the following UIs:

1. Master web UI
2. Worker web UI
3. Application web UI

6.1. Master Web UI:

The master web UI is accessible at <http://localhost:8080/> by default and will be available when Spark Master is running.

Start the Master service using the below command if not already running.

```
spark-class org.apache.spark.deploy.master.Master
```

The master web UI provides an overview of the Spark cluster and displays the following information:

- Master URL and REST URL.
- CPUs and memory allocated to the Spark cluster and to each application.
- Worker status and its allotted resources
- Information about the active and completed applications, such as their status, allotted resources, and duration.
- Information about the active and completed drivers, such as their status and allotted resources.

The screenshot shows the Apache Spark 3.5.1 Master UI at <http://localhost:8080>. The top navigation bar includes links for Home, Help, and Logout. The main header is "Spark Master at spark://[REDACTED]:7077". Below the header, several key metrics are displayed in red boxes: URL (spark://[REDACTED]:7077), Alive Workers (1), Cores in use (4 Total, 0 Used), Memory in use (10.7 GiB Total, 0.0 B Used), Resources in use, Applications (0 Running, 4 Completed), Drivers (0 Running, 2 Completed), and Status (ALIVE). A section titled "Workers (1)" lists one worker with ID worker-20240706152912-[REDACTED]-46075, State ALIVE, Cores 4 (0 Used), and Memory 10.7 GiB (0.0 B Used). Below this are sections for "Running Applications (0)", "Running Drivers (0)", and "Completed Applications (4)". The "Completed Applications" table lists four applications: PythonWordCount, Spark Pi, Spark Pi, and Spark Pi, all completed successfully. Finally, there is a section for "Completed Drivers (2)".

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240706155116-0003	PythonWordCount	4	1024.0 MiB		2024/07/06 15:51:16	hp	FINISHED	20 s
app-20240706154337-0002	Spark Pi	3	1024.0 MiB		2024/07/06 15:43:37	hp	FINISHED	15 s
app-20240706153354-0001	Spark Pi	3	1024.0 MiB		2024/07/06 15:33:54	hp	FINISHED	13 s
app-20240706152948-0000	Spark Pi	4	1024.0 MiB		2024/07/06 15:29:48	hp	FINISHED	13 s

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class	Duration
driver-20240706154325-0001	2024/07/06 15:43:25	worker-20240706152912-[REDACTED]-46075	FINISHED	1	1024.0 MiB		org.apache.spark.examples.SparkPi	
driver-20240706153342-0000	2024/07/06 15:33:42	worker-20240706152912-[REDACTED]-46075	FINISHED	1	1024.0 MiB		org.apache.spark.examples.SparkPi	

6.2. Worker Web UI:

The worker web UI is accessible at <http://localhost:8081> by default and will be available when Spark Worker is running.

Start the Worker service using the below command if not already running (*Make sure that you replace <ipaddress> with IP address mentioned in your Spark Master URL*)

```
spark-class org.apache.spark.deploy.worker.Worker
spark://<ipaddress>:7077
```

The worker web UI provides an overview of the executors and drivers that are spawned by the worker process. It displays information about the allotted resources, status, and provides links to log files for each of these child processes. We can use this web UI to see how many executors are currently running and the amount of resources allotted to each.

Spark Worker at [REDACTED]:46075

ID: worker-20240706152912-[REDACTED] 46075
Master URL: spark://[REDACTED]:7077
Cores: 4 (0 Used)
Memory: 10.7 GiB (0.0 B Used)
Resources:

Back to Master

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
- Running Executors (0)						
+ Finished Executors (3)						
0	KILLED	4	1024.0 MiB		ID: app-20240706152948-0000 Name: Spark Pi User: hp	stdout stderr
0	KILLED	3	1024.0 MiB		ID: app-20240706153354-0001 Name: Spark Pi User: hp	stdout stderr
0	KILLED	3	1024.0 MiB		ID: app-20240706154337-0002 Name: Spark Pi User: hp	stdout stderr

DriverID	Main Class	State	Cores	Memory	Resources	Logs	Notes
driver-20240706154325-0001	org.apache.spark.examples.SparkPi	FINISHED	1	1024.0 MiB		stdout stderr	
driver-20240706153342-0000	org.apache.spark.examples.SparkPi	FINISHED	1	1024.0 MiB		stdout stderr	
+ Finished Drivers (2)							

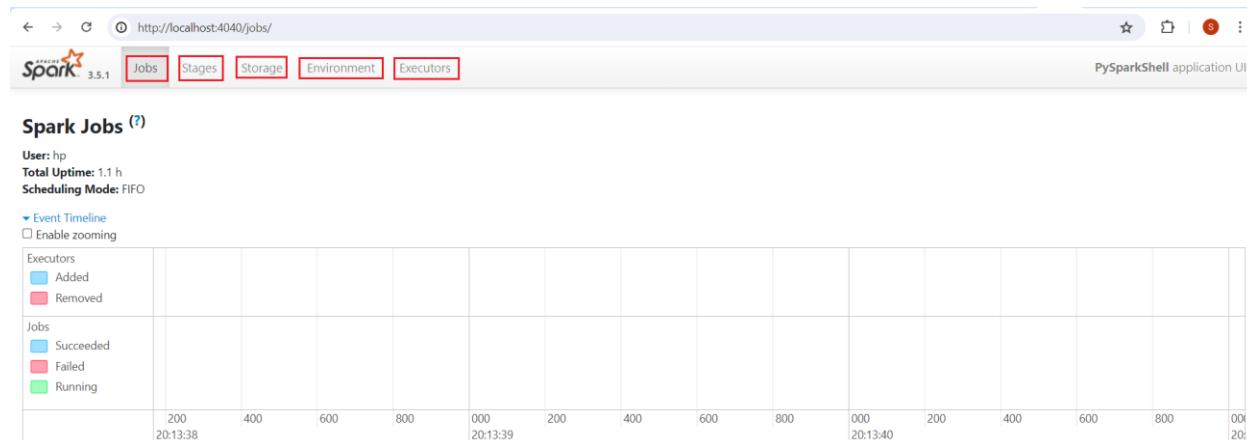
6.3. Application Web UI:

Each Spark application launches its own instance of the web UI. The application web UI is accessible at <http://localhost:4040/jobs/> by default and will be available only when a Spark application is running.

The application web UI provides a wealth of information about the Spark application and can be a useful tool to debug the application. This web UI has the following tabs:

- Jobs:** It displays a summary page of all jobs in the Spark application and a detailed page for each job. The summary page shows high-level information, such as the status, duration, and progress of all jobs and the overall event timeline. When you click on a job on the summary page, you see the detailed page for that job. The detailed page further shows the event timeline, DAG visualization, and all stages of the job.
- Stages:** It displays a summary page that shows the current state of all stages of all jobs in the Spark application, and, when you click on a stage, a detailed page for that stage. The details page shows the event timeline, DAG visualization, and all tasks for the stage.
- Storage:** It displays information about the RDDs and DataFrames, if any, in the application. The summary page shows the storage levels, sizes and partitions of all RDDs, and the detailed page shows the sizes and using executors for all partitions in an RDD.

- **Environment:** It displays the values for the different environment and configuration variables, including JVM, Spark, and system properties.
- **Executors:** It displays summary information about the executors that were created for the application, including memory and disk usage and task and shuffle information. The Storage Memory column shows the amount of memory used and reserved for caching data.
- **SQL:** If the application executes Spark SQL queries, the SQL tab displays information, such as the duration, jobs, and physical and logical plans for the queries.
- **Structured Streaming:** When running Structured Streaming jobs in micro-batch mode, a Structured Streaming tab will be available on the Web UI. It displays scheduling delay and processing time for each micro-batch in the data stream, which can be useful for troubleshooting the streaming application.
- **JDBC/ODBC Server:** This tab is visible when Spark is running as a distributed SQL engine. It shows information about sessions and submitted SQL operations.



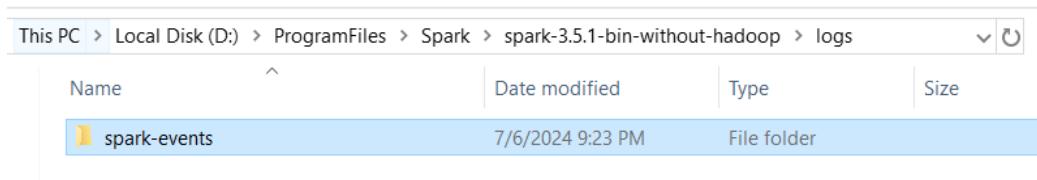
7. Spark History Server:

The Spark History Server is a web UI that displays the logging data from Spark jobs that were run with event logging enabled. It works only when files were not flushed before the Spark Master attempted to build a history user interface.

Before starting the Spark History server, we need to define Spark Event Log directory and Spark History Log directory in `spark-defaults.conf` file.

To store Spark logs on local file system:

Go to SPARK_HOME location and create logs folder and create another folder named spark-events inside logs folder.



To store Spark logs on HDFS:

Create /spark/logs folder and create another folder named spark-events inside /spark/logs folder using these commands:

```
hadoop fs -ls /spark
hadoop fs -mkdir /spark/logs
hadoop fs -mkdir /spark/logs/spark-events
hadoop fs -ls /spark/logs/spark-events
```

A screenshot of a Windows Command Prompt window titled 'Command Prompt'. The window shows the following command-line session:

```
C:\Users\hp>hadoop fs -ls /spark
Found 1 items
drwxr-xr-x  - hp supergroup          0 2024-07-06 17:55 /spark/jars

C:\Users\hp>hadoop fs -mkdir /spark/logs

C:\Users\hp>hadoop fs -mkdir /spark/logs/spark-events

C:\Users\hp>
```

The output shows the creation of the /spark/logs directory and its subdirectory /spark/logs/spark-events.

Now, go to SPARK_HOME/conf location and edit the spark-defaults.conf file (*if it is not available, rename spark-defaults.conf.template to spark-defaults.conf*) to enable event logging.

Depending on where you want to enable event logging, add the following lines in the spark-defaults.conf file and ensure that the paths mentioned in spark.history.fs.logDirectory and spark.eventLog.dir are valid and accessible. Otherwise, Spark History server would fail to start.

To enable event logging on local file system:

```
#Sets the logging directory for the history server
spark.history.fs.logDirectory file:///D:/ProgramFiles/Spark/spark-3.5.1-bin-
without-hadoop/logs

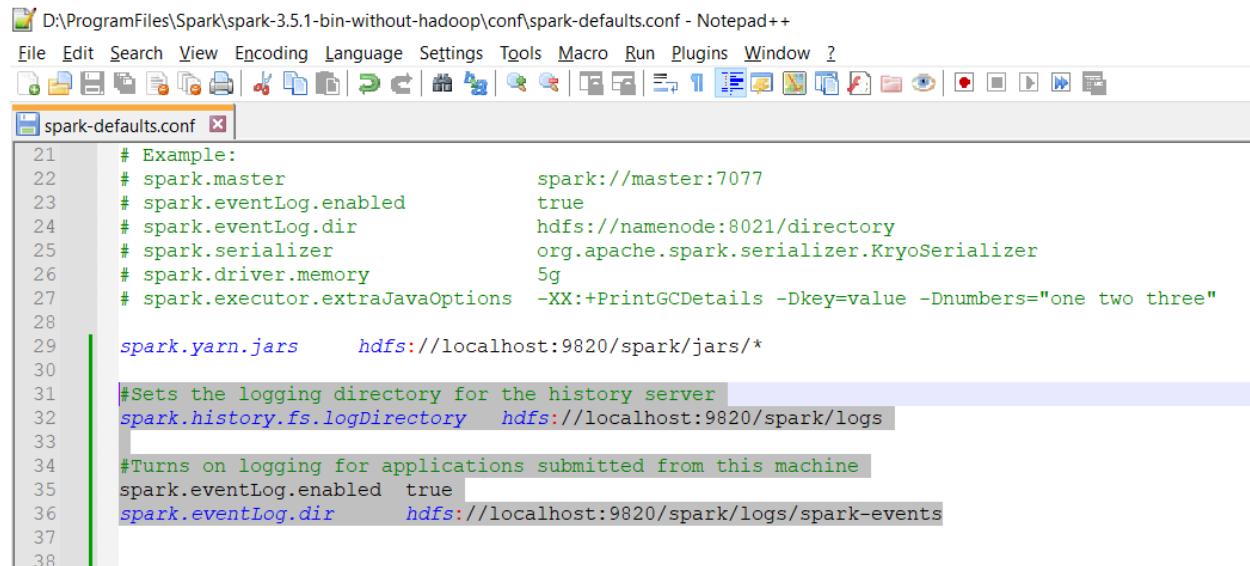
#Turns on logging for applications submitted from this machine
spark.eventLog.enabled true
spark.eventLog.dir           file:///D:/ProgramFiles/Spark/spark-3.5.1-bin-
without-hadoop/logs/spark-events
```

To enable event logging on HDFS:

```
#Sets the logging directory for the history server
spark.history.fs.logDirectory hdfs://localhost:9820/spark/logs

#Turns on logging for applications submitted from this machine
spark.eventLog.enabled true
spark.eventLog.dir           hdfs://localhost:9820/spark/logs/spark-events
```

Since I want to enable event logging on HDFS, I added the above lines of code in `spark-defaults.conf` file as shown below:



The screenshot shows the `spark-defaults.conf` file open in Notepad++. The file contains configuration settings for Spark. The relevant lines for event logging are highlighted in blue:

```
# Example:
# spark.master          spark://master:7077
# spark.eventLog.enabled true
# spark.eventLog.dir    hdfs://namenode:8021/directory
# spark.serializer       org.apache.spark.serializer.KryoSerializer
# spark.driver.memory   5g
# spark.executor.extraJavaOptions -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"
#
spark.yarn.jars      hdfs://localhost:9820/spark/jars/*
#
#Sets the logging directory for the history server
spark.history.fs.logDirectory hdfs://localhost:9820/spark/logs
#
#Turns on logging for applications submitted from this machine
spark.eventLog.enabled true
spark.eventLog.dir       hdfs://localhost:9820/spark/logs/spark-events
```

Now, start the Spark history server in Windows using the following command

```
spark-class org.apache.spark.deploy.history.HistoryServer
```

```
Command Prompt - spark-class org.apache.spark.deploy.history.HistoryServer
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-class org.apache.spark.deploy.history.HistoryServer
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
24/07/06 21:42:33 INFO HistoryServer: Started daemon with process name: 18256@[REDACTED]
24/07/06 21:42:34 INFO SecurityManager: Changing view acls to: hp
24/07/06 21:42:34 INFO SecurityManager: Changing modify acls to: hp
24/07/06 21:42:34 INFO SecurityManager: Changing view acls groups to:
24/07/06 21:42:34 INFO SecurityManager: Changing modify acls groups to:
24/07/06 21:42:34 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: hp; groups with view permissions: EMPTY; users with modify permissions: hp; groups with modify permissions: EMPTY
24/07/06 21:42:34 INFO FsHistoryProvider: History server ui acls disabled; users with admin permissions: ; groups with admin permissions:
24/07/06 21:42:36 INFO JettyUtils: Start Jetty 0.0.0.0:18080 for HistoryServerUI
24/07/06 21:42:36 INFO Utils: Successfully started service 'HistoryServerUI' on port 18080.
24/07/06 21:42:36 INFO HistoryServer: Bound HistoryServer to 0.0.0.0, and started at http://[REDACTED]:18080
```

Once the history server is started, its UI is accessible under default port 18080 at <http://localhost:18080/>

The screenshot shows a web browser window with the URL <http://localhost:18080/?showIncomplete=false>. The page is titled "History Server 3.5.1". It displays event logs from HDFS, last updated on 2024-07-06 21:44:47, in Asia/Calcutta time zone. A message states "No completed applications found!" and provides instructions for specifying a logging directory. A link "Show incomplete applications" is visible.

This History server is useful to track the running and completed Spark applications even if the spark application web UI is not available.

Note:

When event logging is enabled, the default behavior is for all logs to be saved, which causes the storage to grow over time.

To enable automated cleanup, edit `spark-defaults.conf` file and add the following lines to perform cleanup daily and delete logs older than 7 days.

```
spark.history.fs.cleaner.enabled true
spark.history.fs.cleaner.interval 1d
spark.history.fs.cleaner.maxAge 7d
```

8. Write Sample PySpark Code:

Let us write some sample Pyspark code to read a CSV file and write the data into JSON format.

We will write the code to read/write CSV file in local file system as well as in HDFS.

8.1. Read/Write Locally:

Since our SPARK_HOME is currently set to spark-3.5.1-bin-without-hadoop version, all files are expected to be read or written to HDFS.

To read or write files in local file system, we would need additional jar files that doesnot come with spark-3.5.1-bin-without-hadoop version. So, let us temporarily set the SPARK_HOME environment variable to the location where spark-3.5.1-bin-hadoop3 version was installed and then launch pyspark CLI.

Open a new command prompt and run the following commands (*here, my spark-3.5.1-bin-hadoop3 version is installed under D:\ProgramFiles\Spark location*)

```
set SPARK_HOME=D:\ProgramFiles\Spark\spark-3.5.1-bin-hadoop3
```

pyspark

Note that my pyspark interface has opened the Spark context web UI at <http://localhost:4043>

At the `pyspark` prompt, write the below lines of code to read a file into dataframe and displays output.

```
df=spark.read.option("header",True).csv("D:\\Datasets\\departments.csv")
df.printSchema()
df.show(5)
df.count()
df.createOrReplaceTempView("DEPARTMENTS")
df2=spark.sql("select DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID from
DEPARTMENTS where LOCATION_ID <> 1700")
df2.show()
df2.write.format("json").save("D:\\Datasets\\spark-output\\department-locations.json")
```

Here, I am using the input file `departments.csv` available under `D:\\Datasets` folder and writing into output location `D:\\Datasets\\spark-output\\department-locations.json`.

Note: You can copy `departments.csv` file from [here](#) and place it under `D:\\Datasets` folder.

```
>>> df=spark.read.option("header",True).csv("D:\\Datasets\\departments.csv")
>>> df.printSchema()
root
 |-- DEPARTMENT_ID: string (nullable = true)
 |-- DEPARTMENT_NAME: string (nullable = true)
 |-- MANAGER_ID: string (nullable = true)
 |-- LOCATION_ID: string (nullable = true)

>>> df.show(5)
+-----+-----+-----+
|DEPARTMENT_ID|DEPARTMENT_NAME|MANAGER_ID|LOCATION_ID|
+-----+-----+-----+
|          10| Administration|      200|       1700|
|          20|      Marketing|      201|       1800|
|          30| Purchasing|      114|       1700|
|          40|Human Resources|      203|       2400|
|          50|      Shipping|      121|       1500|
+-----+-----+-----+
only showing top 5 rows

>>> df.count()
27
>>> df2 = df.select("*").filter(df.LOCATION_ID != 1700)
>>> df2.show()
+-----+-----+-----+
|DEPARTMENT_ID| DEPARTMENT_NAME|MANAGER_ID|LOCATION_ID|
+-----+-----+-----+
|          20|      Marketing|      201|       1800|
|          40|Human Resources|      203|       2400|
|          50|      Shipping|      121|       1500|
|          60|          IT|      103|       1400|
|          70|Public Relations|      204|       2700|
|          80|          Sales|      145|       2500|
+-----+-----+-----+
>>> df2.write.format("json").save("D:\\Datasets\\spark-output\\department-locations.json")
>>>
```

After the above code is executed, we can see that the output has been generated in D:\Datasets\spark-output location.

The screenshot shows a Windows File Explorer window and a Notepad++ window. The File Explorer window displays files in the D:\Datasets\spark-output\department-locations.json directory, including '_SUCCESS.crc', '.part-00000-44daee05-4175-4f5a-b853-a109c8e7f7e6-c000.json.crc', '_SUCCESS', and 'part-00000-44daee05-4175-4f5a-b853-a109c8e7f7e6-c000.json'. The Notepad++ window shows the JSON content of 'part-00000-44daee05-4175-4f5a-b853-a109c8e7f7e6-c000.json'.

```

D:\Datasets\spark-output\department-locations.json\part-00000-44daee05-4175-4f5a-b853-a109c8e7f7e6-c000.json - Notepad++ - 100% - Encoding: UTF-8 - Language: JSON - Plugins - Help
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
part-00000-44daee05-4175-4f5a-b853-a109c8e7f7e6-c000.json
1 [{"DEPARTMENT_ID": "20", "DEPARTMENT_NAME": "Marketing", "MANAGER_ID": "201", "LOCATION_ID": "1800"}, 2 {"DEPARTMENT_ID": "40", "DEPARTMENT_NAME": "Human Resources", "MANAGER_ID": "203", "LOCATION_ID": "2400"}, 3 {"DEPARTMENT_ID": "50", "DEPARTMENT_NAME": "Shipping", "MANAGER_ID": "121", "LOCATION_ID": "1500"}, 4 {"DEPARTMENT_ID": "60", "DEPARTMENT_NAME": "IT", "MANAGER_ID": "103", "LOCATION_ID": "1400"}, 5 {"DEPARTMENT_ID": "70", "DEPARTMENT_NAME": "Public Relations", "MANAGER_ID": "204", "LOCATION_ID": "2700"}, 6 {"DEPARTMENT_ID": "80", "DEPARTMENT_NAME": "Sales", "MANAGER_ID": "145", "LOCATION_ID": "2500"}]

```

Open Spark Web UI at <http://localhost:4043> where we can notice some jobs submitted for the code that was executed from the Pyspark shell.

The screenshot shows the PySparkShell application UI with the 'Jobs' tab selected. It displays the 'Completed Jobs' section, which lists six completed jobs. Each job entry includes the Job ID, Description, Submitted time, Duration, Stages: Succeeded/Total, and Tasks (for all stages): Succeeded/Total.

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
5	save at NativeMethodAccessorImpl.java:0 save at NativeMethodAccessorImpl.java:0	2024/07/06 22:43:47	0.2 s	1/1	1/1
4	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/07/06 22:43:42	94 ms	1/1	1/1
3	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2024/07/06 22:43:20	0.1 s	1/1 (1 skipped)	1/1 (1 skipped)
2	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2024/07/06 22:43:20	0.2 s	1/1	1/1
1	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/07/06 22:43:14	0.1 s	1/1	1/1
0	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2024/07/06 22:43:06	1.0 s	1/1	1/1

8.2. Read/Write HDFS:

First, let us copy the input file departments.csv into HDFS

```
hadoop fs -put "D:\Datasets\departments.csv" /spark-input
```

```
hadoop fs -ls /spark-input
```

```
C:\Users\hp>hadoop fs -mkdir /spark-input  
C:\Users\hp>hadoop fs -put "D:\Datasets\departments.csv" /spark-input  
C:\Users\hp>hadoop fs -ls /spark-input  
Found 1 items  
-rw-r--r-- 1 hp supergroup      709 2024-07-07 00:34 /spark-input/departments.csv  
C:\Users\hp>
```

Next, open pyspark shell using below command

pyspark

```
C:\Users\hp>pyspark
Python 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
24/07/07 00:35:48 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
24/07/07 00:35:48 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
24/07/07 00:35:48 WARN Utils: Service 'SparkUI' could not bind on port 4042. Attempting port 4043.
Welcome to

    / \
   / \_ _\ \_ / \_ \
  /_ / .\_ \_,/_ / / \_ \
 /_/
version 3.5.1

Using Python version 3.11.7 (main, Dec 15 2023 18:05:47)
Spark context Web UI available at http://192.168.56.1:4043
Spark context available as 'sc' (master = local[*], app id = local-1720292748177).
SparkSession available as 'spark'.
>>>
```

On the pyspark shell, write the below lines of code to read the above input file into dataframe and displays output

```
df=spark.read.options(header=True, inferSchema=True).csv("/spark-
input/departments.csv")
df.printSchema()
df.show(5, truncate=False)
df.count()
df.createOrReplaceTempView("DEPARTMENTS")
df2=spark.sql("select DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID from
DEPARTMENTS where LOCATION_ID <> 1700")
df2.show()
df2.write.format("json").save("/spark-output/department-locations.json")
```

```

>>> df=spark.read.options(header=True, inferSchema=True).csv("/spark-input/departments.csv")
>>> df.printSchema()
root
 |-- DEPARTMENT_ID: integer (nullable = true)
 |-- DEPARTMENT_NAME: string (nullable = true)
 |-- MANAGER_ID: string (nullable = true)
 |-- LOCATION_ID: integer (nullable = true)

>>> df.show(5, truncate=False)
+-----+-----+-----+
|DEPARTMENT_ID|DEPARTMENT_NAME|MANAGER_ID|LOCATION_ID|
+-----+-----+-----+
|10      |Administration| 200      |1700      |
|20      |Marketing     | 201      |1800      |
|30      |Purchasing    | 114      |1700      |
|40      |Human Resources| 203      |2400      |
|50      |Shipping       | 121      |1500      |
+-----+-----+-----+
only showing top 5 rows

>>> df.count()
27
>>> df.createOrReplaceTempView("DEPARTMENTS")
>>> df2=spark.sql("select DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID from DEPARTMENTS where LOCATION_ID <> 1700")
>>> df2.show()
+-----+-----+-----+
|DEPARTMENT_ID| DEPARTMENT_NAME|MANAGER_ID|LOCATION_ID|
+-----+-----+-----+
| 20| Marketing| 201| 1800|
| 40| Human Resources| 203| 2400|
| 50| Shipping| 121| 1500|
| 60| IT| 103| 1400|
| 70| Public Relations| 204| 2700|
| 80| Sales| 145| 2500|
+-----+-----+-----+

>>> df2.write.format("json").save("/spark-output/department-locations.json")
>>>

```

Open another command prompt and run the following commands to see the output file created in HDFS.

```

hadoop fs -ls /spark-output/department-locations.json
hadoop fs -cat /spark-output/department-locations.json/part-00000-
*.json

```

```

C:\Users\hp>hadoop fs -ls /spark-output/department-locations.json
Found 2 items
-rw-r--r-- 1 hp supergroup          0 2024-07-07 00:40 /spark-output/department-locations.json/_SUCCESS
-rw-r--r-- 1 hp supergroup  535 2024-07-07 00:40 /spark-output/department-locations.json/part-00000-bf090c75-1f9
1-4d42-86be-8cccaa030828-c000.json

C:\Users\hp>hadoop fs -cat /spark-output/department-locations.json/part-00000-* .json
{"DEPARTMENT_ID":20,"DEPARTMENT_NAME":"Marketing","MANAGER_ID":"201","LOCATION_ID":1800}
{"DEPARTMENT_ID":40,"DEPARTMENT_NAME":"Human Resources","MANAGER_ID":"203","LOCATION_ID":2400}
{"DEPARTMENT_ID":50,"DEPARTMENT_NAME":"Shipping","MANAGER_ID":"121","LOCATION_ID":1500}
{"DEPARTMENT_ID":60,"DEPARTMENT_NAME":"IT","MANAGER_ID":"103","LOCATION_ID":1400}
 {"DEPARTMENT_ID":70,"DEPARTMENT_NAME":"Public Relations","MANAGER_ID":"204","LOCATION_ID":2700}
 {"DEPARTMENT_ID":80,"DEPARTMENT_NAME":"Sales","MANAGER_ID":"145","LOCATION_ID":2500}

C:\Users\hp>

```

The same is visible in NameNode UI: <http://localhost:9870/dfshealth.html>

In NameNode UI, go to **Utilities** tab and select **Browse the file system** option. Search for `/spark-output/department-locations.json` directory where we can see `part-00000-* .json` file is available.

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

Browse Directory

/spark-output/department-locations.json

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hp	supergroup	0 B	Jul 07 00:40	1	128 MB	_SUCCESS
-rw-r--r--	hp	supergroup	535 B	Jul 07 00:40	1	128 MB	part-00000-bf090c75-1f91-4d42-86be-8ccaa030828-c000.json

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2023.

Open Spark Web UI at <http://localhost:4043> where we can notice some jobs submitted for the code that we executed from the Pyspark shell.

Spark 3.5.1 Jobs Stages Storage Environment Executors SQL / DataFrame PySparkShell application UI

Spark Jobs (7)

User: hp
Total Uptime: 13 min
Scheduling Mode: FIFO
Completed Jobs: 7

Event Timeline

Completed Jobs (7)

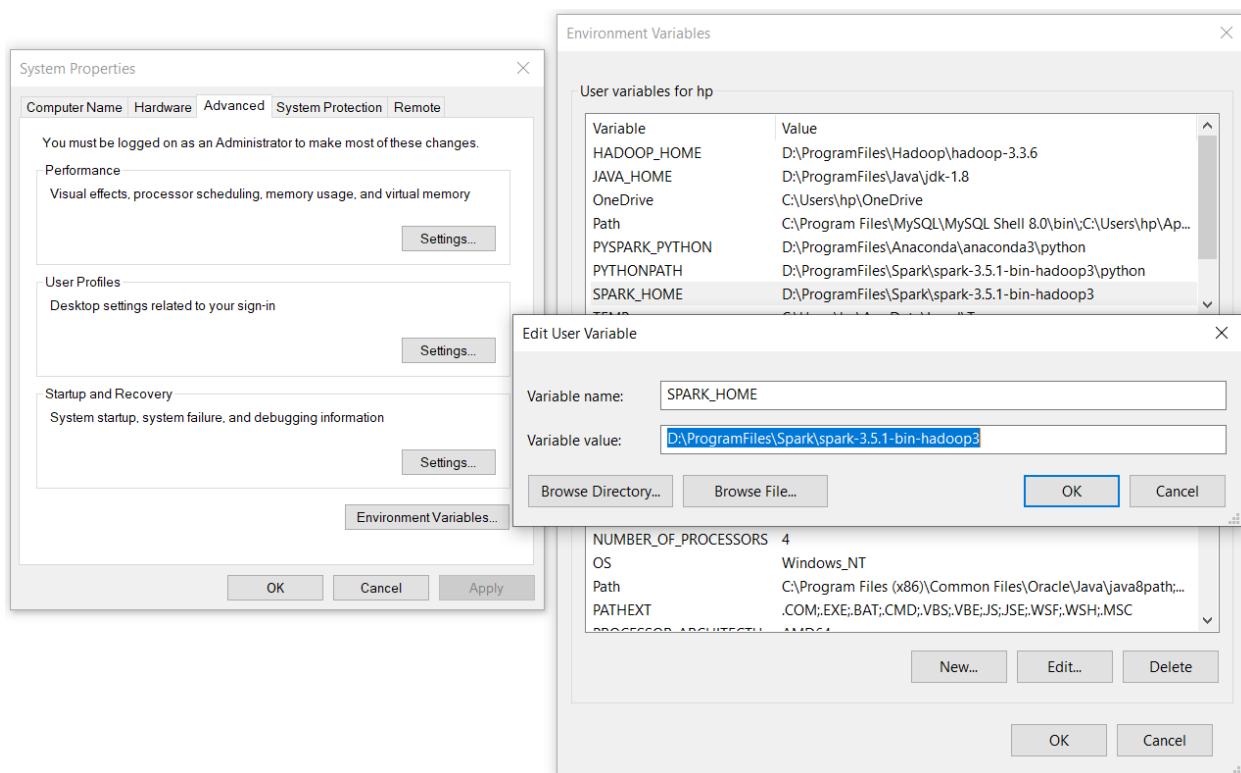
Page: 1

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
6	save at NativeMethodAccessorImpl.java:0 save at NativeMethodAccessorImpl.java:0	2024/07/07 00:40:18	0.5 s	1/1	1/1
5	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/07/07 00:40:11	0.1 s	1/1	1/1
4	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2024/07/07 00:39:47	0.1 s	1/1 (skipped)	1/1 (1 skipped)
3	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2024/07/07 00:39:46	0.8 s	1/1	1/1
2	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/07/07 00:39:40	0.2 s	1/1	1/1
1	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2024/07/07 00:39:17	0.3 s	1/1	1/1
0	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2024/07/07 00:39:16	1 s	1/1	1/1

9. Spark SQL:

The Spark SQL CLI is a conventional tool to run the Hive metastore service in local mode and execute queries from command line. It creates an embedded `metastore_db` directory (*a relational database to manage the metadata of relational entities such as databases, tables, columns partitions*) in the location from where ever `spark-sql` is started.

`spark-sql` command line interface requires `hive` libraries that comes with `spark-3.5.1-bin-hadoop3` version, so set your `SPARK_HOME` environment variable to the location where `spark-3.5.1-bin-hadoop3` version was installed.



Open a new **Command Prompt** in **Administrator** mode and run the following command:

```
spark-sql --master local[2] --executor-memory 1G --driver-memory 1G --conf spark.sql.warehouse.dir="file:///d:/tmp/spark-warehouse"
```

Note that we have set `spark.sql.warehouse.dir` property to `D:\tmp\spark-warehouse` where Spark SQL stores all relational entities such as databases, tables, columns, etc.

```

Administrator: Command Prompt - spark-sql --master local[2] --executor-memory 1G --driver-memory 1G --conf spark.sql.warehouse...
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

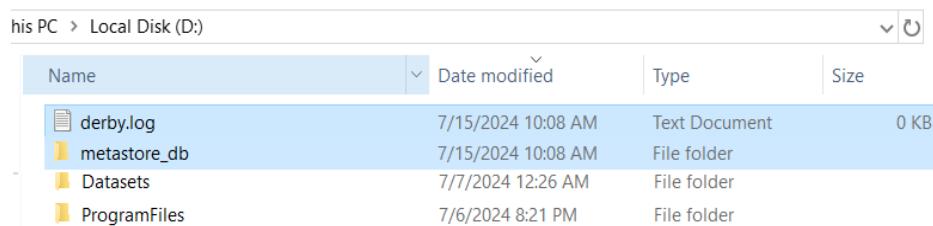
C:\Windows\system32>d:

D:>set SPARK_HOME=D:\ProgramFiles\Spark\spark-3.5.1-bin-hadoop3

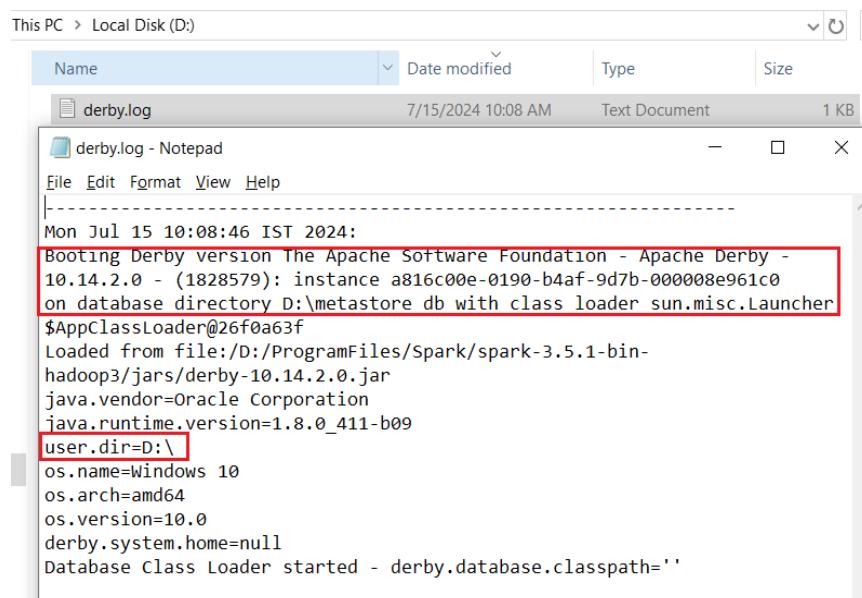
D:>spark-sql --master local[2] --executor-memory 1G --driver-memory 1G --conf spark.sql.warehouse.dir="file:///d:/tmp/spark-warehouse"
24/07/15 10:08:45 WARN HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist
24/07/15 10:08:45 WARN HiveConf: HiveConf of name hive.stats.retries.wait does not exist
24/07/15 10:09:01 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 2.3.0
24/07/15 10:09:01 WARN ObjectStore: setMetaStoreSchemaVersion called but recording version is disabled: version = 2.3.0, comment = Set by MetaStore UNKNOWN
24/07/15 10:09:01 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
Spark Web UI available at http://<redacted>:4040
Spark master: local[2], Application Id: local-1721018322776
spark-sql (default)>

```

As soon as `spark-sql` is launched, we can see that `metastore_db` directory and `derby_log` file is created in the location from where it was launched.



`derby.log` file tells us it has booted the Apache Derby database on the specific database directory.



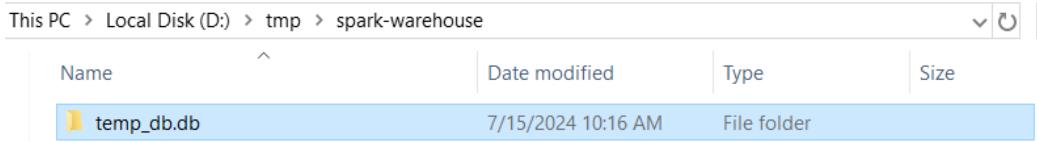
On the `spark-sql>` prompt, let us run some Hive queries to create a database, table and load data into it.

- Create a database named `temp_db`

```
create database temp_db;
show databases;
```

```
spark-sql (default)> show databases;
default
Time taken: 4.516 seconds, Fetched 1 row(s)
spark-sql (default)> create database temp_db;
24/07/15 10:16:35 WARN ObjectStore: Failed to get database temp_db, returning NoSuchObjectException
24/07/15 10:16:35 WARN ObjectStore: Failed to get database temp_db, returning NoSuchObjectException
24/07/15 10:16:35 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
24/07/15 10:16:35 WARN ObjectStore: Failed to get database temp_db, returning NoSuchObjectException
Time taken: 1.371 seconds
spark-sql (default)> show databases;
default
temp_db
Time taken: 0.053 seconds, Fetched 2 row(s)
spark-sql (default)>
```

As soon as the above queries are executed, we can see that `spark-warehouse` directory is created at `D:\tmp` location and `temp_db.db` folder is created under `spark-warehouse` directory.



- Create a table named `employees` in `temp_db` database. This table is created based on the columns data in the CSV file that we are going to load.

```
use temp_db;

create table employees(employee_id int, first_name string, last_name string,
email string, phone_number string, hire_date string, job_id string, salary
int, commission_pct int, manager_id int, department_id int) row format
delimited fields terminated by ','tblproperties
('skip.header.line.count'='1');
```

```

spark-sql (default)> use temp_db;
Time taken: 0.043 seconds
spark-sql (temp_db)> create table employees(employee_id int, first_name string, last_name string, email string, phone_number string, hire_date string, job_id string, salary int, commission_pct int, manager_id int, department_id int) row format delimited fields terminated by ','tblproperties ('skip.header.line.count'=1');
24/07/15 10:42:29 WARN SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFactory.
24/07/15 10:42:29 WARN HiveConf: HiveConf of name hive.internal.ss.authz.settings.applied.marker does not exist
24/07/15 10:42:29 WARN HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist
24/07/15 10:42:29 WARN HiveConf: HiveConf of name hive.stats.retries.wait does not exist
24/07/15 10:42:29 WARN HiveMetaStore: Location: file:/d:/tmp/spark-warehouse/temp_db.db/employees specified for non-external table:employees
Time taken: 2.068 seconds
spark-sql (temp_db)>

```

On the local system, employees folder is created in D:\tmp\temp_db.db location.

This PC > Local Disk (D:) > tmp > spark-warehouse > temp_db.db		
Name	Date modified	Type
employees	7/15/2024 10:42 AM	File folder

- Verify the format of the employees table created.

```
describe formatted employees;
```

```

spark-sql (temp_db)> describe formatted employees;
employee_id          int
first_name           string
last_name            string
email                string
phone_number         string
hire_date             string
job_id               string
salary               int
commission_pct       int
manager_id           int
department_id        int

# Detailed Table Information
Catalog              spark_catalog
Database             temp_db
Table                employees
Owner                hp
Created Time         Mon Jul 15 10:42:29 IST 2024
Last Access          UNKNOWN
Created By           Spark 3.5.1
Type                MANAGED
Provider             hive
Table Properties    [skip.header.line.count=1, transient_lastDdlTime=1721020805]
Location             file:/d:/tmp/spark-warehouse/temp_db.db/employees
Serde Library        org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat          org.apache.hadoop.mapred.TextInputFormat
OutputFormat         org.apache.hadoop.hive.io.HiveIgnoreKeyTextOutputFormat
Storage Properties   [serialization.format=,, field.delim=,]
Partition Provider   Catalog
Time taken: 0.1 seconds, Fetched 29 row(s)
spark-sql (temp_db)>

```

- Load data from `employees.csv` file located in the local directory `D:\Datasets` into the hive table `employees` (*You can download `employees.csv` file from [this location](#) and copy to `D:\Datasets` folder in your machine*).

```
load data local inpath 'file:///D:/Datasets/employees.csv' into table
employees;
```

```
spark-sql (temp_db)> load data local inpath 'file:///D:/Datasets/employees.csv' into table employees;
[LOAD_DATA_PATH_NOT_EXISTS] LOAD DATA input path does not exist: file:///D:/Datasetsemployees.csv.
spark-sql (temp_db)> load data local inpath 'file:///D:/Datasets/employees.csv' into table employees;
Time taken: 0.913 seconds
spark-sql (temp_db)>
```

In local system, there is a `employees.csv` file created in `D:\tmp\temp_db.db\employees` location.

Name	Date modified	Type	Size
.employees.csvcrc	7/15/2024 10:45 AM	CRC File	1 KB
employees.csv	7/15/2024 10:45 AM	Microsoft Excel Co...	4 KB

- Select top 5 records in `employees` table from Hive CLI.

```
set spark.hadoop.hive.cli.print.header=true;
select * from employees limit 5;
```

```
spark-sql (temp_db)> set spark.hadoop.hive.cli.print.header=true;
spark.hadoop.hive.cli.print.header      true
Time taken: 0.02 seconds, Fetched 1 row(s)
spark-sql (temp_db)> select * from employees limit 5;
NULL    FIRST_NAME      LAST_NAME      EMAIL      PHONE_NUMBER    HIRE_DATE      JOB_ID      NULL      NULL      NULL      NULL
198    Donald OConnell DOCONNEL     650.507.9833  21-JUN-07      SH_CLERK      2600      NULL      124      50
      50
199    Douglas Grant   DGRANT       650.507.9844  13-JAN-08      SH_CLERK      2600      NULL      124      50
200    Jennifer Whalen JWALEN      515.123.4444  17-SEP-03      AD_ASST      4400      NULL      101      10
201    Michael Hartstein MHARTSTE   515.123.5555  17-FEB-04      MK_MAN      13000     NULL      100      20
Time taken: 0.177 seconds, Fetched 5 row(s)
spark-sql (temp_db)>
```

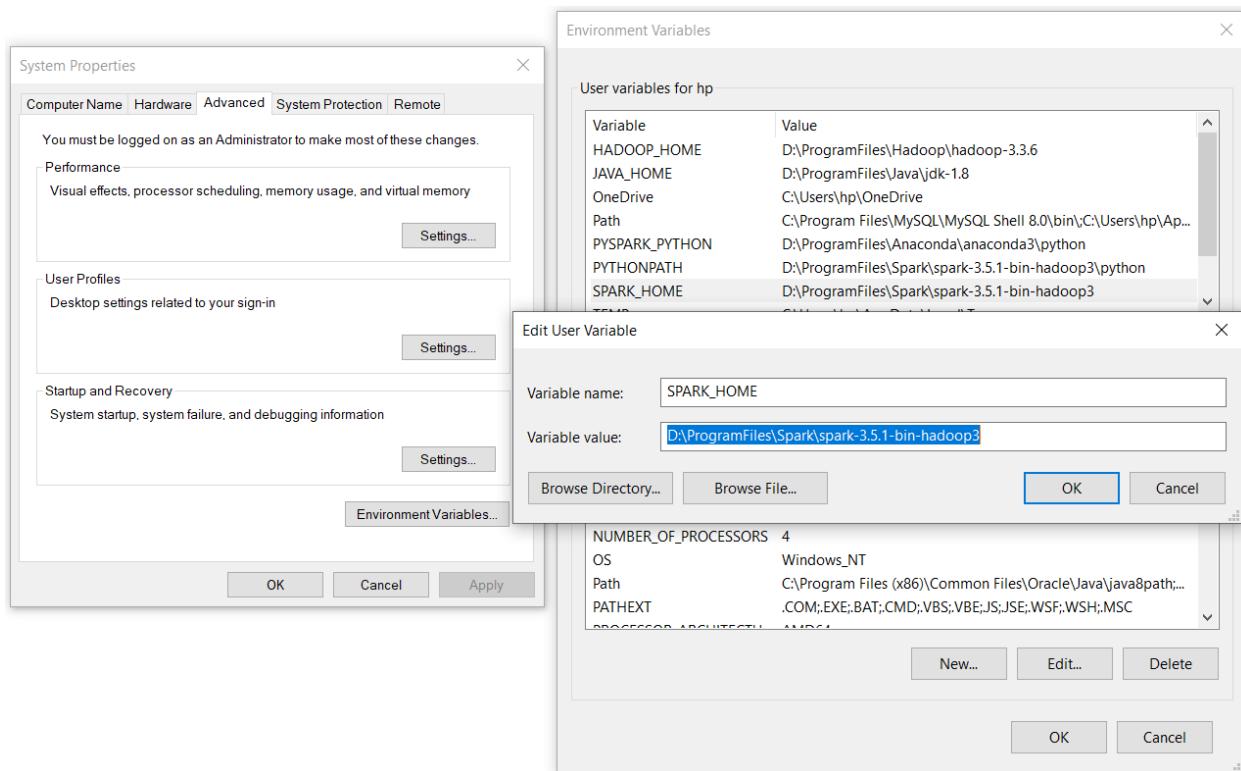
Note: By default, Spark SQL will not skip headers although we set the table property `tblproperties ('skip.header.line.count'='1')`.

Similarly, we can run any queries from `spark-sql` CLI that connects to Hive metastore database running locally.

10. Spark Beeline:

Spark provides `beeline` CLI that runs on Thrift JDBC/ODBC server (*corresponds to hiveserver2 service in built-in Hive*). Spark SQL can also act as a distributed query engine using its JDBC/ODBC server.

Since `beeline` command line interface requires hive libraries that comes with `spark-3.5.1-bin-hadoop3` version, make sure that your `SPARK_HOME` environment variable is set to the location where `spark-3.5.1-bin-hadoop3` version was installed.



To start using Spark provided `beeline`, we need to first start the Spark Master, Worker and Thrift JDBC/ODBC services.

10.1.1. Start Spark Master:

Open a new **Command Prompt** or **Windows PowerShell** and run the following command to start the Spark Master service:

```
spark-class org.apache.spark.deploy.master.Master
```

```

C:\Users\hp>spark-class org.apache.spark.deploy.master.Master
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-class org.apache.spark.deploy.master.Master
24/07/19 03:37:49 INFO Master: Started daemon with process name: 17860@HP-Opti...[REDACTED]
24/07/19 03:37:50 INFO SecurityManager: Changing view acls to: hp
24/07/19 03:37:50 INFO SecurityManager: Changing modify acls to: hp
24/07/19 03:37:50 INFO SecurityManager: Changing view acls groups to:
24/07/19 03:37:50 INFO SecurityManager: Changing modify acls groups to:
24/07/19 03:37:50 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: hp; groups with view permissions: EMPTY; users with modify permissions: hp; groups with modify permissions: EMP
TY
24/07/19 03:37:51 INFO Utils: Successfully started service 'sparkMaster' on port 7077.
24/07/19 03:37:51 INFO Master: Starting Spark master at spark://[REDACTED]:7077
24/07/19 03:37:51 INFO Master: Running Spark version 3.5.1
24/07/19 03:37:51 INFO JettyUtils: Start Jetty 0.0.0.0:8080 for MasterUI
24/07/19 03:37:52 INFO Utils: Successfully started service 'MasterUI' on port 8080.
24/07/19 03:37:52 INFO MasterWebUI: Bound MasterWebUI to 0.0.0.0, and started at http://[REDACTED]:8080
24/07/19 03:37:52 INFO Master: I have been elected leader! New state: ALIVE

```

Here, we can see that Spark master is started at <spark://<ipaddress>:7077> i.e listening on default port 7077 and **MasterWebUI** is started at <http://<ipaddress>:8080> by default.

Note:

If you are unable to see any logging on the console, then open log4j2.properties file in SPARK_HOME\conf location and change the rootLogger.level property to info value.

```

D:\ProgramFiles\Spark\spark-3.5.1-bin-hadoop3\conf\log4j2.properties - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window 2
File Edit View Insert Encoding Language Settings Tools Macro Run Plugins Window 2
log4j2.properties
1  #
2  # Licensed to the Apache Software Foundation (ASF) under one or more
3  # contributor license agreements. See the NOTICE file distributed with
4  # this work for additional information regarding copyright ownership.
5  # The ASF licenses this file to You under the Apache License, Version 2.0
6  # (the "License"); you may not use this file except in compliance with
7  # the License. You may obtain a copy of the License at
8  #
9  #     http://www.apache.org/licenses/LICENSE-2.0
10 #
11 # Unless required by applicable law or agreed to in writing, software
12 # distributed under the License is distributed on an "AS IS" BASIS,
13 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14 # See the License for the specific language governing permissions and
15 # limitations under the License.
16 #
17 #
18 # Set everything to be logged to the console
19 | rootLogger.level = info
20 | rootLogger.appenderRef.stdout.ref = console

```

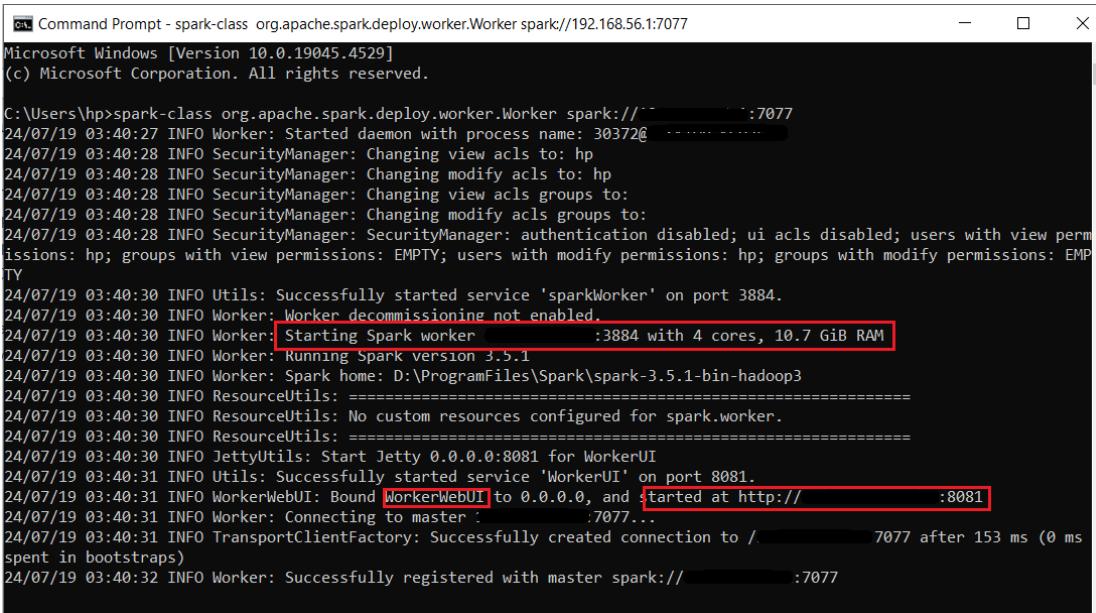
10.1.2. Start Spark Worker:

Open a new **Command Prompt** or **Windows PowerShell** and run the following command to start the Spark Worker service (*Make sure that you replace <ipaddress> with IP address mentioned in your Spark Master URL*)

```

spark-class org.apache.spark.deploy.worker.Worker
spark://<ipaddress>:7077

```



```
C:\Users\hp>spark-class org.apache.spark.deploy.worker.Worker spark://192.168.56.1:7077
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>spark-class org.apache.spark.deploy.worker.Worker spark://192.168.56.1:7077
24/07/19 03:40:27 INFO Worker: Started daemon with process name: 30372@192.168.56.1
24/07/19 03:40:28 INFO SecurityManager: Changing view acls to: hp
24/07/19 03:40:28 INFO SecurityManager: Changing modify acls to: hp
24/07/19 03:40:28 INFO SecurityManager: Changing view acls groups to:
24/07/19 03:40:28 INFO SecurityManager: Changing modify acls groups to:
24/07/19 03:40:28 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: hp; groups with view permissions: EMPTY; users with modify permissions: hp; groups with modify permissions: EMPTY
24/07/19 03:40:30 INFO Utils: Successfully started service 'sparkWorker' on port 3884.
24/07/19 03:40:30 INFO Worker: Worker decommissioning not enabled.
24/07/19 03:40:30 INFO Worker: Starting Spark worker :3884 with 4 cores, 10.7 GiB RAM
24/07/19 03:40:30 INFO Worker: Running Spark version 3.5.1
24/07/19 03:40:30 INFO Worker: Spark home: D:\ProgramFiles\Spark\spark-3.5.1-bin-hadoop3
24/07/19 03:40:30 INFO ResourceUtils: =====
24/07/19 03:40:30 INFO ResourceUtils: No custom resources configured for spark.worker.
24/07/19 03:40:30 INFO ResourceUtils: =====
24/07/19 03:40:30 INFO JettyUtils: Start Jetty 0.0.0.0:8081 for WorkerUI
24/07/19 03:40:31 INFO Utils: Successfully started service 'WorkerUI' on port 8081.
24/07/19 03:40:31 INFO WorkerWebUI: Bound WorkerWebUI to 0.0.0.0, and started at http://:8081
24/07/19 03:40:31 INFO Worker: Connecting to master :7077...
24/07/19 03:40:31 INFO TransportClientFactory: Successfully created connection to /:7077 after 153 ms (0 ms spent in bootstraps)
24/07/19 03:40:32 INFO Worker: Successfully registered with master spark://:7077
```

Here, we can see that Spark worker is started at <ipaddress>:3884 (on a random port) with available number of cores and RAM running on the machine. **WorkerWebUI** is started at <http://<ipaddress>:8081> by default.

10.1.3. Start Spark Thrift Server:

Spark Thrift services tries to access (or create if not available) an embedded `metastore_db` directory (*a relational database to manage the metadata of relational entities such as databases, tables, columns partitions*) in the location from where ever it is started.

Open a new **Command Prompt or Windows PowerShell** and run the following command to start the Spark Thrift service (*Make sure that you replace <ipaddress> with IP address mentioned in your Spark Master URL*)

Note: To connect to the existing `metastore_db` (*that was already created by Spark SQL*), make sure to navigate to the location where `metastore_db` is available and execute this command:

```
spark-submit --class
org.apache.spark.sql.hive.thriftserver.HiveThriftServer2 --master
spark://<ipaddress>:7077
```

```
cmd Command Prompt - spark-submit --class org.apache.spark.sql.hive.thriftserver.HiveThriftServer2 --master spark://[REDACTED]:7077
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>d:

D:\>spark-submit --class org.apache.spark.sql.hive.thriftserver.HiveThriftServer2 --master spark://[REDACTED]:7077
24/07/19 03:48:04 INFO HiveThriftServer2: Started daemon with process name: 27700@[REDACTED]
24/07/19 03:48:04 INFO HiveThriftServer2: Starting SparkContext
24/07/19 03:48:04 INFO HiveConf: Found configuration file null
24/07/19 03:48:04 INFO SparkContext: Running Spark version 3.5.1
24/07/19 03:48:04 INFO SparkContext: OS info Windows 10, 10.0, amd64
24/07/19 03:48:04 INFO SparkContext: Java version 1.8.0_411
24/07/19 03:48:05 INFO ResourceUtils: =====
24/07/19 03:48:05 INFO ResourceUtils: No custom resources configured for spark.driver.
24/07/19 03:48:05 INFO ResourceUtils: =====
24/07/19 03:48:05 INFO SparkContext: Submitted application: SparkSQL:[REDACTED].1
24/07/19 03:48:05 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
24/07/19 03:48:05 INFO ResourceProfile: Limiting resource is cpu
24/07/19 03:48:05 INFO ResourceProfileManager: Added ResourceProfile id: 0
24/07/19 03:48:05 INFO SecurityManager: Changing view acls to: hp
24/07/19 03:48:05 INFO SecurityManager: Changing modify acls to: hp
24/07/19 03:48:05 INFO SecurityManager: Changing view acls groups to:
24/07/19 03:48:05 INFO SecurityManager: Changing modify acls groups to:
24/07/19 03:48:05 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: hp; groups with view permissions: EMPTY; users with modify permissions: hp; groups with modify permissions: EMPTY
24/07/19 03:48:06 INFO Utils: Successfully started service 'sparkDriver' on port 3962.

24/07/19 03:48:21 INFO HiveMetaStore: No user is added in admin role, since config is empty
24/07/19 03:48:21 INFO HiveMetaStore: 0: get_database: default
24/07/19 03:48:21 INFO audit: ugi=hp ip=unknown-ip-addr cmd=get_database: default
24/07/19 03:48:21 INFO HiveUtils: Initializing execution hive, version 2.3.9
24/07/19 03:48:21 INFO HiveClientImpl: Warehouse location for Hive client (version 2.3.9) is file:/D:/spark-warehouse
24/07/19 03:48:21 INFO SessionManager: Operation log root directory is created: C:\Users\hp\AppData\Local\Temp\hp\operation_logs
24/07/19 03:48:21 INFO SessionManager: HiveServer2: Background operation thread pool size: 100
24/07/19 03:48:21 INFO SessionManager: HiveServer2: Background operation thread wait queue size: 100
24/07/19 03:48:21 INFO SessionManager: HiveServer2: Background operation thread keepalive time: 10 seconds
24/07/19 03:48:21 INFO AbstractService: Service:OperationManager is initied.
24/07/19 03:48:21 INFO AbstractService: Service:SessionManager is initied.
24/07/19 03:48:21 INFO AbstractService: Service:CLIService is initied.
24/07/19 03:48:21 INFO AbstractService: Service:ThriftBinaryCLIService is initied.
24/07/19 03:48:21 INFO AbstractService: Service:HiveServer2 is initied.
24/07/19 03:48:21 INFO AbstractService: Service:OperationManager is started.
24/07/19 03:48:21 INFO AbstractService: Service:SessionManager is started.
24/07/19 03:48:21 INFO AbstractService: Service:CLIService is started.
24/07/19 03:48:21 INFO AbstractService: Service:ThriftBinaryCLIService is started.
24/07/19 03:48:22 INFO ThriftCLIService: Starting ThriftBinaryCLIService on port 10000 with 5...500 worker threads
24/07/19 03:48:22 INFO AbstractService: Service:HiveServer2 is started
24/07/19 03:48:22 INFO HiveThriftServer2: HiveThriftServer2 started
```

Here, we can see that Spark **ThriftBinaryCLIService** is started on port **10000** and **HiveServer2** and **HiveThriftServer2** services are started.

10.1.4. Start Beeline:

Open a new **Command Prompt** or **Windows PowerShell** and run the following command to start the beeline CLI:

```
beeline
```

```
Command Prompt - beeline
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>beeline
Beeline version 2.3.9 by Apache Hive
beeline>
```

Here, we can see the Beeline version **2.3.9**.

Now, connect to the JDBC/ODBC thrift server in beeline using the following command. When Beeline asks for a username and password, simply enter the username on your machine and a blank password.

```
!connect jdbc:hive2://localhost:10000
```

```
beeline> !connect jdbc:hive2://localhost:10000
Connecting to jdbc:hive2://localhost:10000
Enter username for jdbc:hive2://localhost:10000: hp
Enter password for jdbc:hive2://localhost:10000:
24/07/24 22:50:51 INFO Utils: Supplied authorities: localhost:10000
24/07/24 22:50:51 INFO Utils: Resolved authority: localhost:10000
Connected to: Spark SQL (version 3.5.1)
Driver: Hive JDBC (version 2.3.9)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://localhost:10000>
```

Our Beeline has been successfully connected to Thrift server running on 10000 port.

Now, we can run queries similar to Spark SQL

```
show databases;
use temp_db;
show tables;
select * from employees limit 5;
```

```

0: jdbc:hive2://localhost:10000> show databases;
+-----+
| namespace |
+-----+
| default   |
| temp_db   |
+-----+
2 rows selected (0.262 seconds)
0: jdbc:hive2://localhost:10000> use temp_db;
+-----+
| Result  |
+-----+
+-----+
No rows selected (0.083 seconds)
0: jdbc:hive2://localhost:10000> show tables;
+-----+-----+-----+
| namespace | tableName | isTemporary |
+-----+-----+-----+
| temp_db   | employees | false      |
+-----+-----+-----+
1 row selected (0.149 seconds)
0: jdbc:hive2://localhost:10000> select * from employees limit 5;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| employee_id | first_name | last_name | email    | phone_number | hire_date   | job_id    | salary    | commission_pc
t | manager_id | department_id |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| NULL        | FIRST_NAME | LAST_NAME | EMAIL     | PHONE_NUMBER | HIRE_DATE  | JOB_ID    | NULL      | NULL
| NULL        | NULL       |           |           |             |           |           |           |           |
| 198         | Donald     | OConnell | DOCONNEL | 650.507.9833 | 21-JUN-07  | SH_CLERK  | 2600      | NULL
| 124         |           | 50          |           |             |           |           |           |           |
| 199         | Douglas    | Grant      | DGRANT    | 650.507.9844 | 13-JAN-08  | SH_CLERK  | 2600      | NULL
| 124         |           | 50          |           |             |           |           |           |           |
| 200         | Jennifer  | Whalen    | JWHALEN  | 515.123.4444 | 17-SEP-03  | AD_ASST   | 4400      | NULL
| 101         |           | 10          |           |             |           |           |           |           |
| 201         | Michael   | Hartstein | MHARTSTE | 515.123.5555 | 17-FEB-04  | MK_MAN   | 13000     | NULL
| 100         |           | 20          |           |             |           |           |           |           |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows selected (0.536 seconds)
0: jdbc:hive2://localhost:10000>

```

Use this command to exit out of beeline shell:

```
!quit
```

```

0: jdbc:hive2://localhost:10000> !quit
Closing: 0: jdbc:hive2://localhost:10000

C:\Users\hp>

```

11. Spark with Jupiter Notebook:

Now, we will see how to integrate **Pyspark** with **Jupiter Notebook** available in Anaconda installation so that we can execute Spark code on Jupiter Notebook directly.

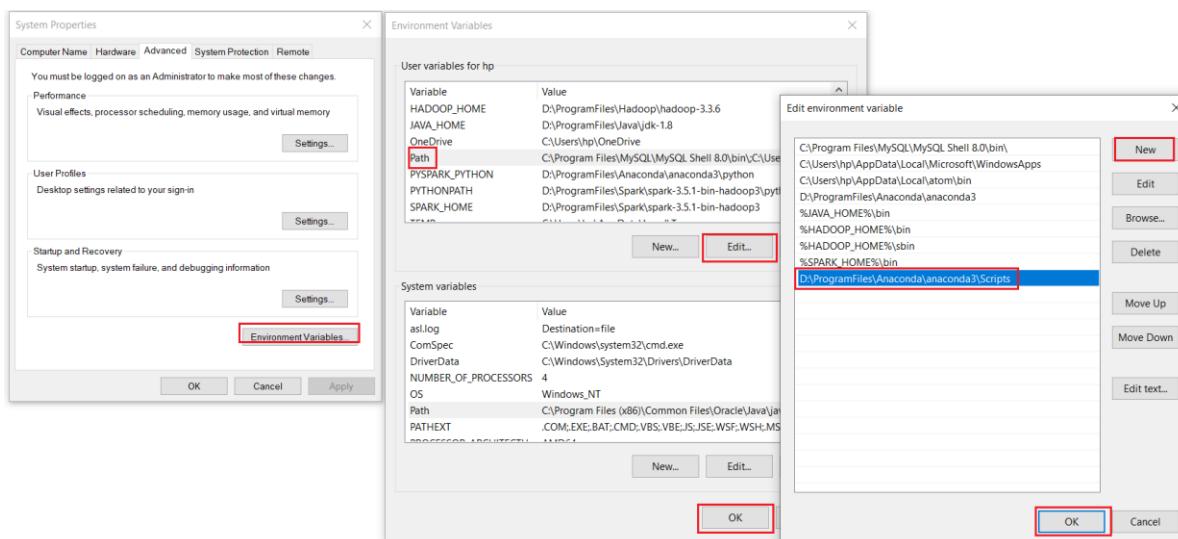
First, make sure that Anaconda scripts location is added to **PATH** environment variable.

In the Windows search bar, start typing “environment variables” and select the first match which opens up **System Properties** dialog where press **Environment Variables** button.

In the **Environment Variables** dialog, select **PATH** variable under **User Variables** and press **Edit** button. Then press **New** and add the value

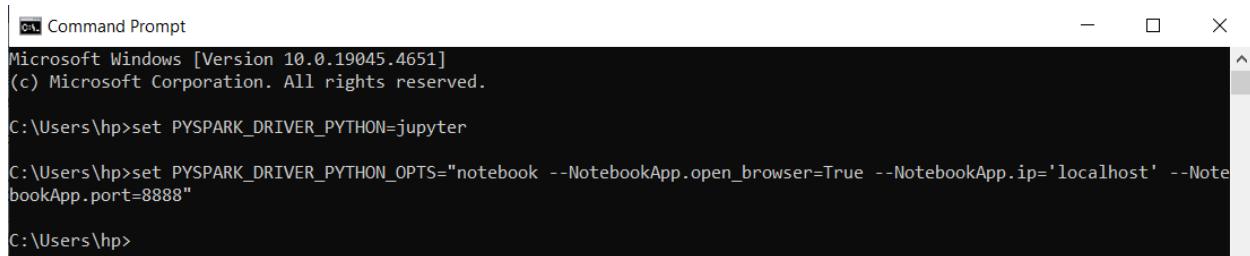
D:\ProgramFiles\Anaconda\anaconda3\Scripts (*the path where Anaconda distribution was installed*) and press **OK**. Then press **OK** on **Environment Variables** dialog

Note: My Anaconda is installed under **D:\ProgramFiles**. If you have installed in a different location, update the above path appropriately.



Now, open a new command prompt and set the following environment variables for pyspark to execute Jupyter.

```
set PYSPARK_DRIVER_PYTHON=jupyter
set PYSPARK_DRIVER_PYTHON_OPTS="notebook --"
NotebookApp.open_browser=True --NotebookApp.ip='localhost' --
NotebookApp.port=8888"
```



```
Command Prompt
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>set PYSPARK_DRIVER_PYTHON=jupyter
C:\Users\hp>set PYSPARK_DRIVER_PYTHON_OPTS="notebook --NotebookApp.open_browser=True --NotebookApp.ip='localhost' --NotebookApp.port=8888"
C:\Users\hp>
```

Note: If you are installing Spark on a Virtual Machine and would like to access Jupyter from your host browser, you should set the `NotebookApp.ip` flag to `--NotebookApp.ip='0.0.0.0'` in the above command so that your VM's Jupyter server will accept external connections. You can then access Jupyter notebook from the host machine on port 8888.

Execute pyspark with the following command:

```
pyspark --master local[*] --executor-memory 1G --driver-memory 1G --
conf spark.sql.warehouse.dir="file:///D:/tmp/spark-warehouse" --
packages com.amazonaws:aws-java-sdk-pom:1.10.34 --packages
org.apache.hadoop:hadoop-aws:2.8.0
```

The above `pyspark` command performs the following duties:

- Execute `pyspark` locally (local master) with all cores of the computer
- Set the Spark Executor memory to 1GB
- Set the Spark Driver memory to 1GB
- Set the Spark SQL Warehouse directory to `D:\tmp\spark-warehouse` in local system to store Spark SQL data
- Set specific packages such as `amazonaws:aws-java-sdk-pom` and `org.apache.hadoop:hadoop-aws` for `pyspark` to load (*These packages are necessary to access AWS S3 repositories from Spark*)

Note: After executing the above command, you may encounter an error `"Jupyter command `jupyter-notebook --NotebookApp.open_browser=True --NotebookApp.ip='localhost' --NotebookApp.port=8888` not found"` as below.

```
C:\Users\hp>pyspark --master local[*] --executor-memory 1G --driver-memory 1G --conf spark.sql.warehouse.dir="file:///D:/tmp/spark-warehouse" --packages com.amazonaws:aws-java-sdk-pom:1.10.34 --packages org.apache.hadoop:hadoop-aws:2.8.0
usage: jupyter [-h] [--version] [--config-dir] [--data-dir] [--runtime-dir] [--paths] [--json] [--debug] [subcommand]

Jupyter: Interactive Computing

positional arguments:
  subcommand      the subcommand to launch

options:
  -h, --help      show this help message and exit
  --version       show the versions of core jupyter packages and exit
  --config-dir   show Jupyter config dir
  --data-dir     show Jupyter data dir
  --runtime-dir  show Jupyter runtime dir
  --paths         show all Jupyter paths. Add --json for machine-readable format.
  --json          output paths as machine-readable json
  --debug         output debug information about paths

Available subcommands: console dejavu events execute kernel kernelspec lab labextension labhub migrate nbconvert
notebook qtconsole run script server troubleshoot trust

Jupyter command `jupyter-notebook --NotebookApp.open_browser=True --NotebookApp.ip='localhost' --NotebookApp.port=8888` not found.

C:\Users\hp>
```

To resolve the above error, set `PYSPARK_DRIVER_PYTHON_OPTS` to "notebook" and launch pyspark with these commands:

```
set PYSPARK_DRIVER_PYTHON_OPTS="notebook"

pyspark --master local[*] --executor-memory 1G --driver-memory 1G --conf spark.sql.warehouse.dir="file:///D:/tmp/spark-warehouse" --packages com.amazonaws:aws-java-sdk-pom:1.10.34 --packages org.apache.hadoop:hadoop-aws:2.8.0
```

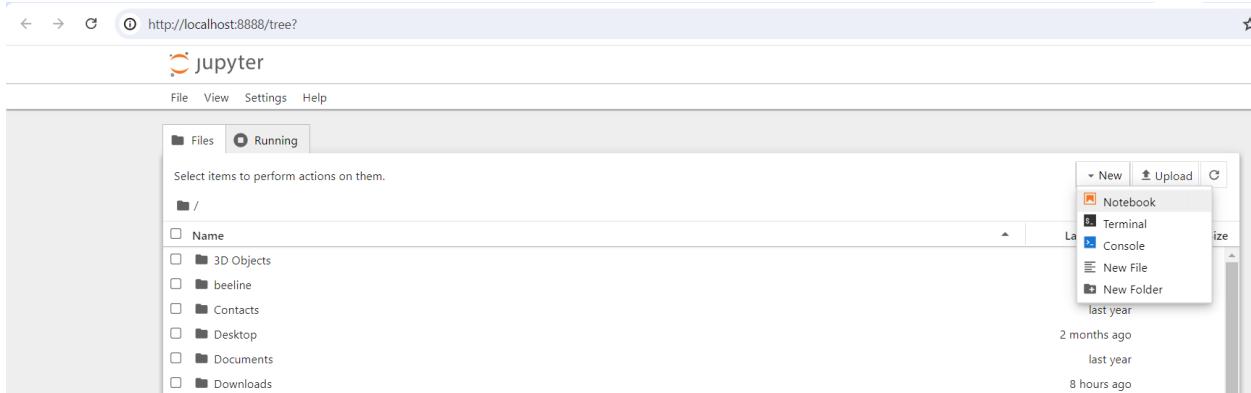
```
C:\ Command Prompt - pyspark --master local[*] --executor-memory 1G --driver-memory 1G --conf spark.sql.warehouse.dir="file:///D:/..." — □ ×

C:\Users\hp>set PYSPARK_DRIVER_PYTHON_OPTS="notebook"

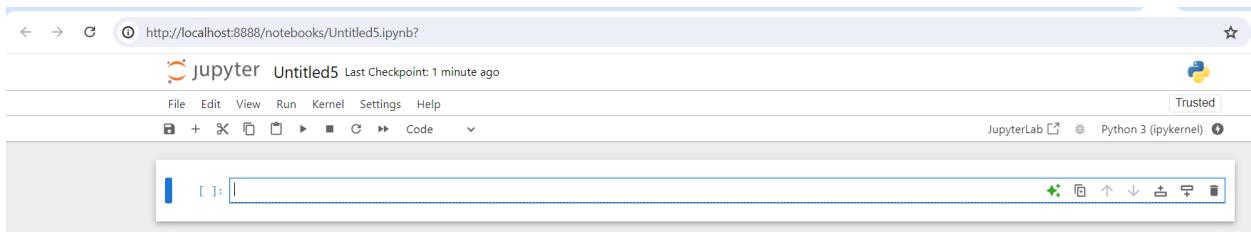
C:\Users\hp>pyspark --master local[*] --executor-memory 1G --driver-memory 1G --conf spark.sql.warehouse.dir="file:///D:/tmp/spark-warehouse" --packages com.amazonaws:aws-java-sdk-pom:1.10.34 --packages org.apache.hadoop:hadoop-aws:2.8.0
[I 2024-07-24 23:36:44.876 ServerApp] Package notebook took 0.0000s to import
[I 2024-07-24 23:36:45.369 ServerApp] Package aext_assistant took 0.4916s to import
[I 2024-07-24 23:36:45.378 ServerApp] Package aext_core took 0.0074s to import
[I 2024-07-24 23:36:45.411 ServerApp] **** ENVIRONMENT Environment.PRODUCTION ****
[I 2024-07-24 23:36:45.418 ServerApp] **** ENVIRONMENT Environment.PRODUCTION ****
[I 2024-07-24 23:36:45.424 ServerApp] Package aext_panels took 0.0452s to import
[I 2024-07-24 23:36:45.430 ServerApp] Package aext_share_notebook took 0.0063s to import
[I 2024-07-24 23:36:45.485 ServerApp] Package jupyter_lsp took 0.0532s to import
[W 2024-07-24 23:36:45.486 ServerApp] A '_jupyter_server_extension_points` function was not found in jupyter_lsp. Instead, a '_jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2024-07-24 23:36:45.514 ServerApp] Package jupyter_server_terminals took 0.0265s to import
[I 2024-07-24 23:36:45.515 ServerApp] Package jupyterlab took 0.0000s to import
[I 2024-07-24 23:36:45.667 ServerApp] Package notebook_shim took 0.0000s to import
[W 2024-07-24 23:36:45.667 ServerApp] A '_jupyter_server_extension_points` function was not found in notebook_shim. Instead, a '_jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2024-07-24 23:36:47.214 ServerApp] Package panel.io.jupyter_server_extension took 1.5426s to import
[I 2024-07-24 23:36:47.215 ServerApp] aext_assistant | extension was successfully linked.
[I 2024-07-24 23:36:47.216 ServerApp] aext_core | extension was successfully linked.
[I 2024-07-24 23:36:47.219 ServerApp] aext_panels | extension was successfully linked.
[I 2024-07-24 23:36:47.220 ServerApp] aext_share_notebook | extension was successfully linked.
[I 2024-07-24 23:36:47.221 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2024-07-24 23:36:47.230 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2024-07-24 23:36:47.244 ServerApp] jupyterlab | extension was successfully linked.
```

```
[I 2024-07-24 23:36:47.244 ServerApp] jupyterlab | extension was successfully linked.
[I 2024-07-24 23:36:47.256 ServerApp] notebook | extension was successfully linked.
[I 2024-07-24 23:36:47.913 ServerApp] notebook_shim | extension was successfully linked.
[I 2024-07-24 23:36:47.914 ServerApp] panel.io.jupyter_server_extension | extension was successfully linked.
[I 2024-07-24 23:36:47.968 ServerApp] notebook_shim | extension was successfully loaded.
[I 2024-07-24 23:36:47.970 ServerApp] Registered aext_assistant server extension
[I 2024-07-24 23:36:47.970 ServerApp] aext_assistant | extension was successfully loaded.
[I 2024-07-24 23:36:47.973 ServerApp] Registered aext_core server extension
[I 2024-07-24 23:36:47.974 ServerApp] aext_core | extension was successfully loaded.
[I 2024-07-24 23:36:47.978 ServerApp] Registered aext_panels server extension
[I 2024-07-24 23:36:47.978 ServerApp] aext_panels | extension was successfully loaded.
[I 2024-07-24 23:36:47.980 ServerApp] Registered aext_share_notebook_server server extension
[I 2024-07-24 23:36:47.980 ServerApp] aext_share_notebook | extension was successfully loaded.
[I 2024-07-24 23:36:47.986 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2024-07-24 23:36:47.987 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2024-07-24 23:36:47.997 LabApp] JupyterLab extension loaded from D:\ProgramFiles\Anaconda\anaconda3\Lib\site-packages\jupyterlab
[I 2024-07-24 23:36:47.997 LabApp] JupyterLab application directory is D:\ProgramFiles\Anaconda\anaconda3\share\jupyter\lab
[I 2024-07-24 23:36:47.999 LabApp] Extension Manager is 'pypi'.
[I 2024-07-24 23:36:48.003 ServerApp] jupyterlab | extension was successfully loaded.
[I 2024-07-24 23:36:48.015 ServerApp] notebook | extension was successfully loaded.
[I 2024-07-24 23:36:48.016 ServerApp] panel.io.jupyter_server_extension | extension was successfully loaded.
[I 2024-07-24 23:36:48.018 ServerApp] Serving notebooks from local directory: C:\Users\hp
[I 2024-07-24 23:36:48.018 ServerApp] Jupyter Server 2.10.0 is running at:
[I 2024-07-24 23:36:48.019 ServerApp] http://localhost:8888/tree?token=1c42b1e14b12732ad71f7b4fe3e034a3252c6b7983fcba
[I 2024-07-24 23:36:48.019 ServerApp] http://127.0.0.1:8888/tree?token=1c42b1e14b12732ad71f7b4fe3e034a3252c6b7983fcba
3ca
[I 2024-07-24 23:36:48.022 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

It will open Jupyter application URL <http://localhost:8888/tree>? where press **New** button and select **Notebook** option.



It opens an **Untitled.ipynb** application where we can execute the python code that runs on Spark engine.



Execute the following lines of code on the Jupyter notebook to create a Spark Session, read a CSV file D:\Datasets\departments.csv into a DataFrame and display the schema of DataFrame and its data.

```
import pyspark as ps
spark = ps.sql.SparkSession.builder.getOrCreate()
df=spark.read.option("header",True).csv("D:\\Datasets\\departments.csv")
df.printSchema()
df.show()
```

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

- Header:** http://localhost:8888/notebooks/Untitled5.ipynb? Last Checkpoint: 4 minutes ago
- Toolbar:** File, Edit, View, Run, Kernel, Settings, Help, Trusted, JupyterLab, Python 3 (pykernel)
- Code Cells:** Five cells numbered [1] through [5].
 - [1]: import pyspark as ps
 - [2]: spark = ps.sql.SparkSession.builder.getOrCreate()
 - [3]: df=spark.read.option("header",True).csv("D:\\Datasets\\departments.csv")
 - [4]: df.printSchema()
 - [5]: df.show()
- Data Output:** Cell [5] displays the schema and data of the DataFrame.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury	-	1700
130	Corporate Tax	-	1700
140	Control And Credit	-	1700
150	Shareholder Services	-	1700

Congratulations!! You have installed Spark 3.5.1 version as a Standalone as well as on Hadoop cluster with Python API implementation successfully and executed Python Spark code and queries on Spark SQL and Beeline. You also got an exposure on how to integrate the Spark engine with Jupiter Notebook provided by Anaconda distribution.