

Assisted Practice 15.1: RDD Partitions Using Coalesce Transformation

Problem Scenario: Create an RDD to check the number of partitions after applying the coalesce transformation

Objective: In this demonstration, you will create an RDD and perform coalesce transformation.

Tasks to Perform:

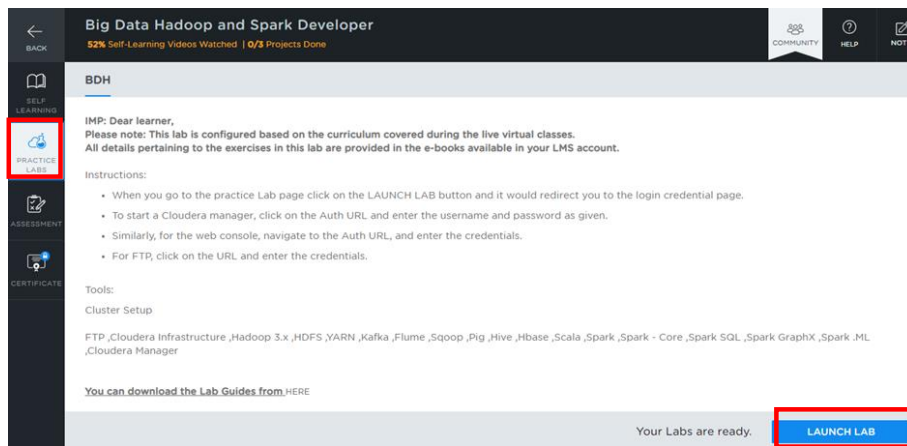
1. Login into the “**webconsole**” and open the PySpark shell
2. Import the required libraries and create a Spark Session
3. Create an RDD using the parallelize method and define eight different values
4. Check the number of partitions using the getNumpartitions function and then use the coalesce function over the given RDD

Steps to Perform:

Step 1: Log in to your LMS account

Step 2: Open the course “**Big Data Hadoop and Spark developer**”

Step 3: On the left side, click on the **“PRACTICE LABS”** tab and click on the **“LAUNCH LAB”** button



Big Data Hadoop and Spark Developer
52% Self-Learning Videos Watched | 0/3 Projects Done

BDH

IMP: Dear learner,
Please note: This lab is configured based on the curriculum covered during the live virtual classes.
All details pertaining to the exercises in this lab are provided in the e-books available in your LMS account.

Instructions:

- When you go to the practice Lab page click on the LAUNCH LAB button and it would redirect you to the login credential page.
- To start a Cloudera manager, click on the Auth URL and enter the username and password as given.
- Similarly, for the web console, navigate to the Auth URL, and enter the credentials.
- For FTP, click on the URL and enter the credentials.

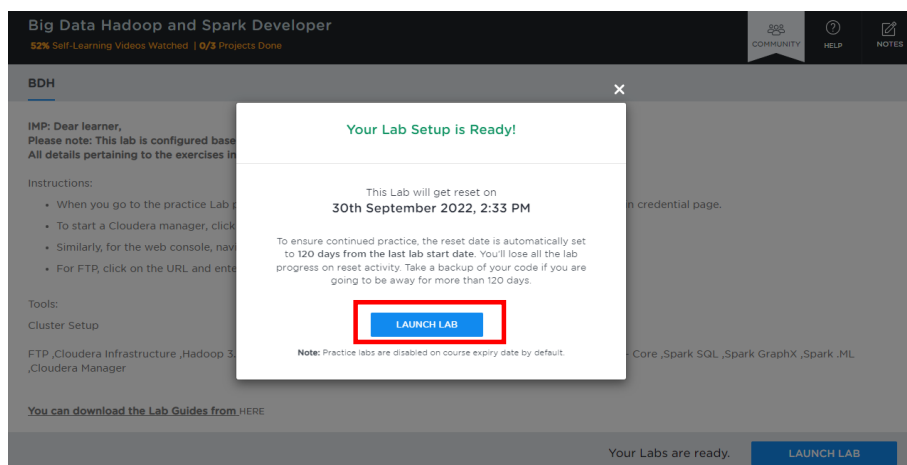
Tools:
Cluster Setup

FTP ,Cloudera Infrastructure ,Hadoop 3.x ,HDFS ,YARN ,Kafka ,Flume ,Sqoop ,Pig ,Hive ,Hbase ,Scala ,Spark ,Spark - Core ,Spark SQL ,Spark GraphX ,Spark .ML ,Cloudera Manager

You can download the Lab Guides from [HERE](#)

Your Labs are ready. **LAUNCH LAB**

Step 4: Again, click on the **“LAUNCH LAB”** button



Big Data Hadoop and Spark Developer
52% Self-Learning Videos Watched | 0/3 Projects Done

BDH

IMP: Dear learner,
Please note: This lab is configured based on the curriculum covered during the live virtual classes.
All details pertaining to the exercises in this lab are provided in the e-books available in your LMS account.

Instructions:

- When you go to the practice Lab page click on the LAUNCH LAB button and it would redirect you to the login credential page.
- To start a Cloudera manager, click on the Auth URL and enter the username and password as given.
- Similarly, for the web console, navigate to the Auth URL, and enter the credentials.
- For FTP, click on the URL and enter the credentials.

Tools:
Cluster Setup

FTP ,Cloudera Infrastructure ,Hadoop 3.x ,HDFS ,YARN ,Kafka ,Flume ,Sqoop ,Pig ,Hive ,Hbase ,Scala ,Spark ,Spark - Core ,Spark SQL ,Spark GraphX ,Spark .ML ,Cloudera Manager

You can download the Lab Guides from [HERE](#)

Your Labs are ready. **LAUNCH LAB**

Your Lab Setup is Ready!

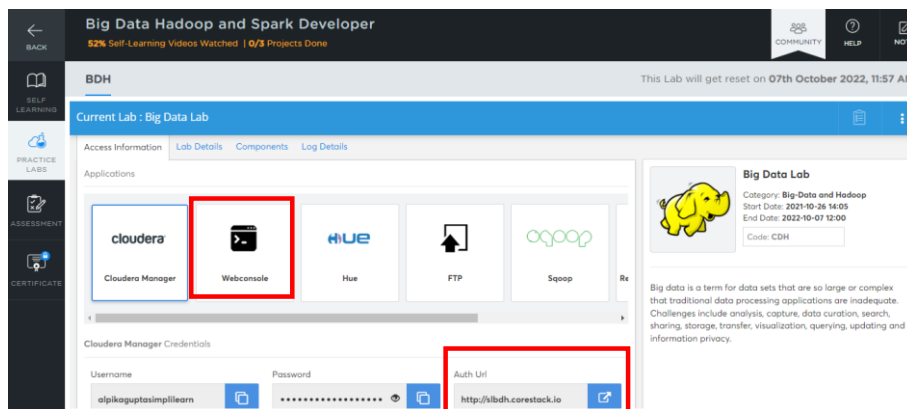
This Lab will get reset on
30th September 2022, 2:33 PM

To ensure continued practice, the reset date is automatically set to 120 days from the last lab start date. You'll lose all the lab progress on reset activity. Take a backup of your code if you are going to be away for more than 120 days.

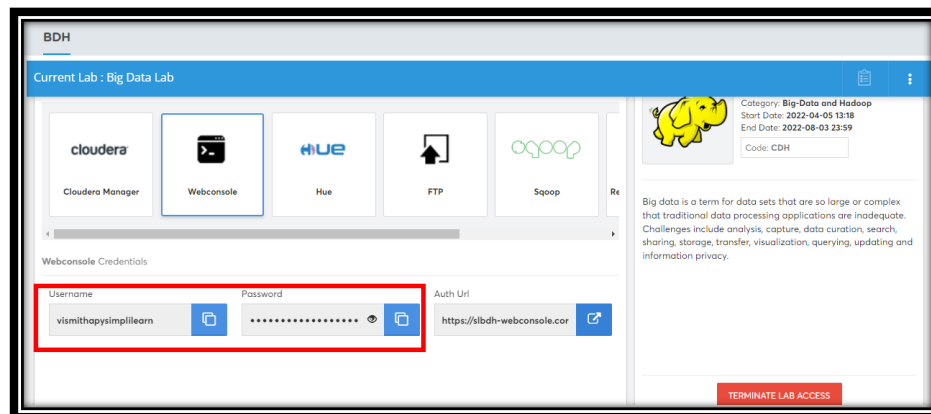
LAUNCH LAB

Note: Practice labs are disabled on course expiry date by default.

Step 5: Click on the lab window and click on **“Webconsole”** and click on the **“Auth Url”**



Step 6: Copy the **“Username”** and the **“Password”** provided to log in to the **“Webconsole”**



Step 7: Paste the **“Username”** and the **“Password”** on the console and click on enter

Note: The password will not be visible when pasted on the console.

Step 8: Enter the “PySpark” shell by running the below command.

Command:

`pyspark3`

```

Password for testdemomay1301mailinator@BDH-ENV.GNE4-RUTX.CLOUDERA.SITE:
[testdemomay1301mailinator@bdh-cluster2-edgenode10 ~]$ pyspark3
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/05/25 01:09:47 WARN util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
22/05/25 01:09:47 WARN util.Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
22/05/25 01:09:47 WARN util.Utils: Service 'SparkUI' could not bind on port 4042. Attempting port 4043.
22/05/25 01:09:47 WARN util.Utils: Service 'SparkUI' could not bind on port 4043. Attempting port 4044.
22/05/25 01:09:47 WARN util.Utils: Service 'SparkUI' could not bind on port 4044. Attempting port 4045.
Welcome to

  _ _ _ _ _
 _/   \ _/   \ _/   \
/_ _ _/_/ _/_/_/_/ _/_/_/_/
 version 3.1.2.7.2.12.4-1

Using Python version 3.7.3 (default, Mar 27 2019 22:11:17)
Spark context Web UI available at http://bdh-cluster2-edgenode10.bdh-env.gne4-rutx.cloudera.site:4045
Spark context available as 'sc' (master = local[*], app id = local-1653440987724).
SparkSession available as 'spark'.
>>>
  
```

Step 9: Import the required libraries and create a Spark Session as shown below

```

  _ _ _ _ _
 _/   \ _/   \ _/   \
/_ _ _/_/ _/_/_/_/ _/_/_/_/
 version 3.1.2.7.2.12.4-1

Using Python version 3.7.3 (default, Mar 27 2019 22:11:17)
Spark context Web UI available at http://bdh-cluster2-edgenode10.bdh-env.gne4-rutx.cloudera.site:4045
Spark context available as 'sc' (master = local[*], app id = local-1653883814277).
SparkSession available as 'spark'.
>>>
>>> from pyspark import SparkContext
>>> from pyspark.sql import SparkSession
>>>
>>> # Create Spark Session.
... sc = SparkContext = SparkSession \
...     .builder \
...     .appName("Simplilearn Examples") \
...     .getOrCreate() \
...     .sparkContext
>>>
  
```

Step 10: Create an RDD using parallelize method and define eight different values

Command:

```
rdd = sc.parallelize((0,1,2,3,4,5,6,7))
```

```
rdd.collect()
```

```
>>> rdd = sc.parallelize((0,1,2,3,4,5,6,7))
>>> rdd.collect()
[0, 1, 2, 3, 4, 5, 6, 7]
>>>
```

Step 11: Check the number of partitions created while creating RDD using the below command:

Command:

```
rdd.getNumPartitions()
```

```
>>> rdd.getNumPartitions()
8
>>>
```

Step 12: Next, use coalesce function over the given RDD

Command:

```
rdd1 = rdd.coalesce(4)
```

```
>>> rdd1 = rdd.coalesce(4)
>>> rdd1.getNumPartitions()
4
>>>
```

Step 13: Now, as you can see that RDD is partitioned and decreased to 4.

Commented [SB1]: Should this be 'Now, as we can see that'?

Commented [AG2R1]: done

