Big Data Hadoop and Spark Developer

Lesson-End Project Solution

simplilearn

Get Certified. Get Ahead.

Flipkart Analysis

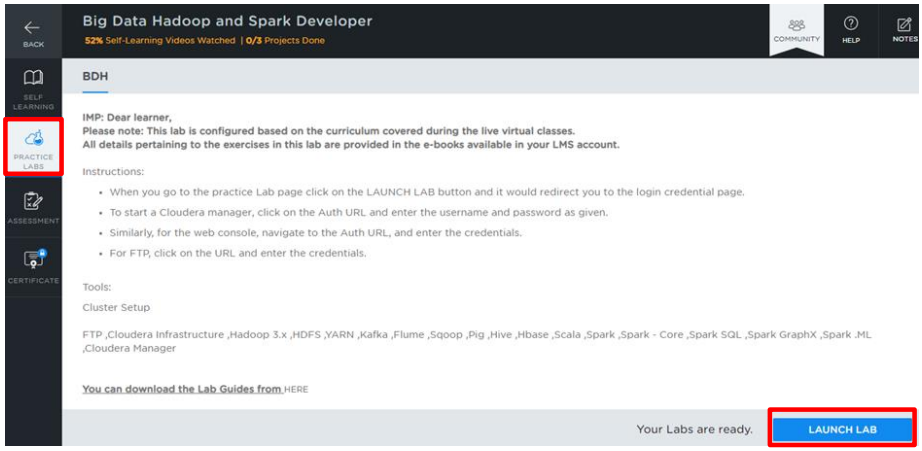
Steps to Perform:

Step 1: Log in to your LMS account

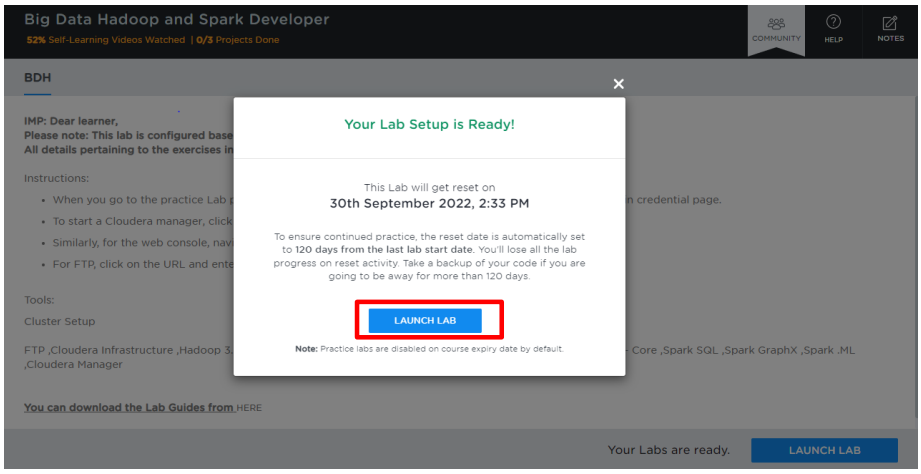
Step 2: Open the course "**Big Data Hadoop and Spark Developer**"

Step 3: Download the datasets from the "**Course Resources**" section

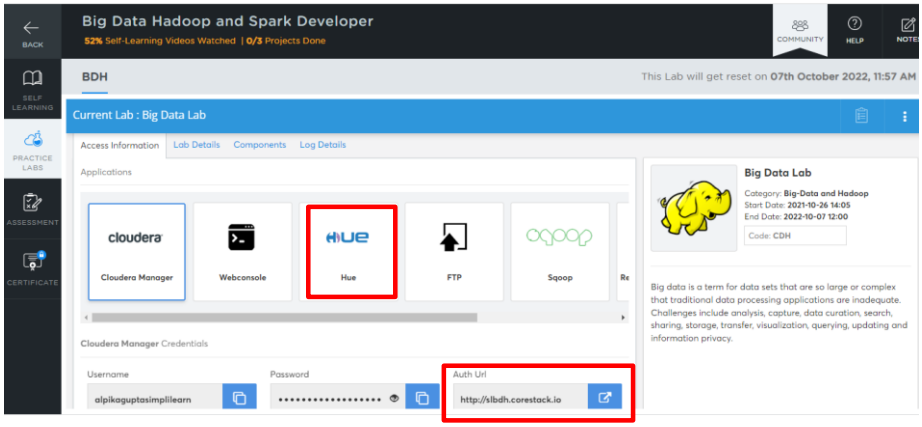
Step 4: Click on the "**PRACTICE LABS**" tab on the left side and select "**LAUNCH LAB**"



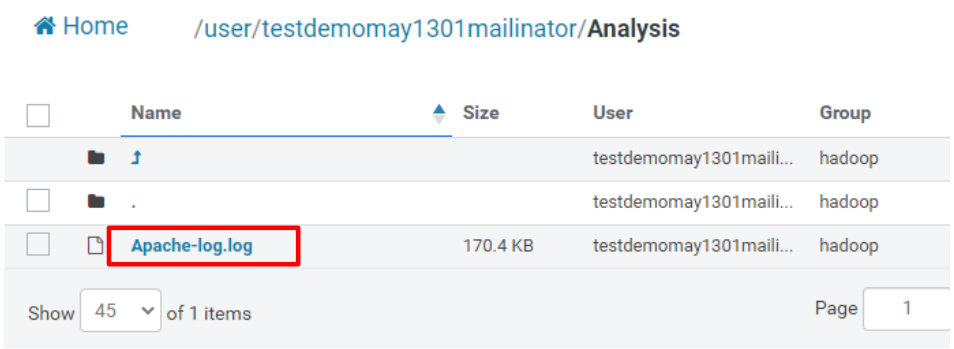
Step 5: Click on the "**LAUNCH LAB**" button



Step 6: Click on **"HUE"** to upload the datasets



Step 7: Log in to "**HUE"** and click on create a directory named **"Analysis"** and upload the **"Apache-log.log"** file into it



Step 8: Write the logic for the Mapper and Reducer files to process the log files

According to the problem statement, you need to extract the hour of the day. So, the Mapper code is shown below:

```
public class LogMapper extends Mapper<LongWritable, Text, IntWritable, IntWritable> {

   private IntWritable hour = new IntWritable();
   private final static IntWritable one = new IntWritable(1);
   private static Pattern logPattern = Pattern
                .compile("([^ ]*) ([^ ]*) ([^ ]*) \\[([^]]*)\\]" + " \"([^\"]*)\"" + " ([^ ]*) ([^ ]*).*");
   private static SimpleDateFormat sdf = new SimpleDateFormat("dd/MMM/yyyy:HH:mm:ss");

   public void map(LongWritable key, Text value, Context context) throws InterruptedException, IOException {

         Date date = null;
         String line = ((Text) value).toString();
         Matcher matcher = logPattern.matcher(line);
         if (matcher.matches()) {
               String timestamp = matcher.group(4);
               try {
```

```
                date = (Date) sdf.parse(timestamp);
            } catch (ParseException ex) {
                ex.printStackTrace();
            }
            Calendar cal = Calendar.getInstance();
            cal.setTime(date);
            hour.set(cal.get(Calendar.HOUR_OF_DAY));
            context.write(hour, one);
        }
    }
```

The output of the mapper will be the key-value pair as shown below:
(16,1).(1,1),(2,1).(3,1)
Where key is an hour of the day and 1 represents the occurrence of that hour.

Step 9: Create a Reducer file that will take Mapper as input and predict results to save Flipkart from a disaster

Reducer code:

```
public class LogReducer extends Reducer <IntWritable, IntWritable, IntWritable,
IntWritable> {

    private static Logger logger = LoggerFactory.getLogger(LogReducer.class);

    public void reduce(IntWritable key, Iterable<IntWritable> values,
    Context context) throws IOException, InterruptedException {

    logger.info("Reducer started");
    int sum = 0;
    for (IntWritable value : values) {
    sum = sum + value.get();
    }
    context.write(key, new IntWritable(sum));
```

```
        logger.info("Reducer completed");

        }
}
```

This will take all inputs from the Mapper and perform aggregation to provide the final result.

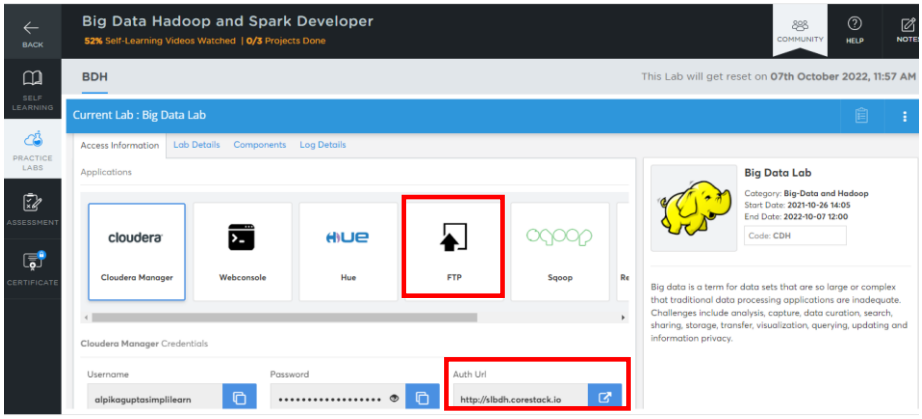
Step 10: Create a job file to identify a Mapper and a Reducer in a MapReduce framework and how they can work in tandem.

```
public class LogDriver {

        private static Logger logger = LoggerFactory.getLogger(LogDriver.class);

        public static void main(String[] args) throws Exception {
        logger.info("Code started");

        @SuppressWarnings("deprecation")
   Job job = new Job();
        job.setJarByClass(LogDriver.class);
        job.setJobName("Log Analyzer");

        job.setMapperClass(LogMapper.class);
        job.setReducerClass(LogReducer.class);

        job.setNumReduceTasks(1);

        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
```

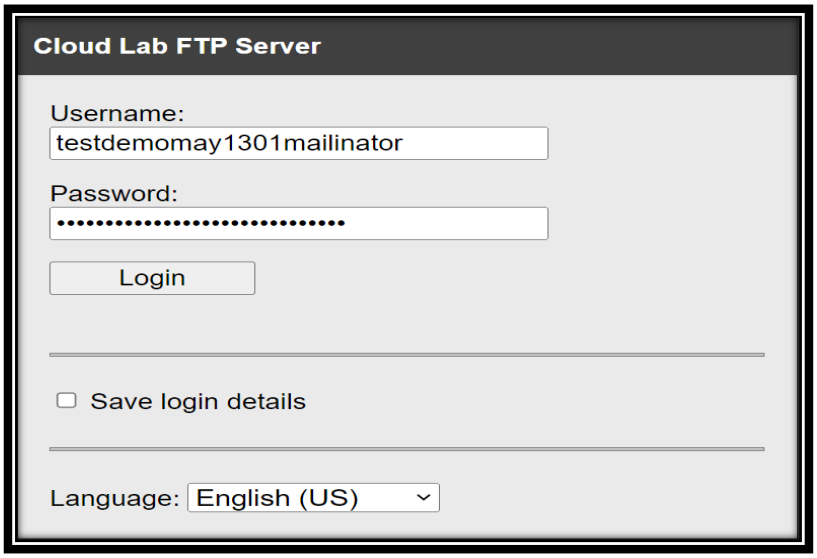
```
logger.info("Code ended");
}
```

Step 11: Create a JAR file to execute the same on the cluster
Note: All JAVA files, config, and JAR files are kept on the drive.

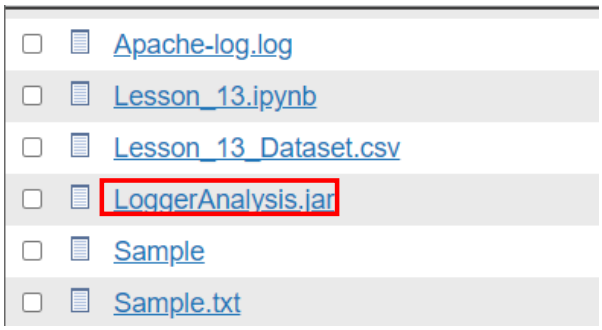
Step 12: Click on **"FTP"** and click on the **"Auth Url"** to upload the dataset. Copy the "**Username**" and "**Password**" provided to log in to **"FTP"**



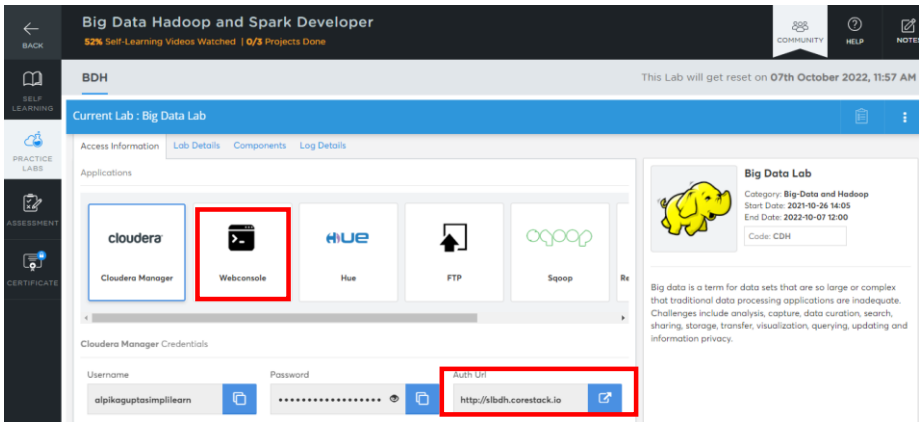
Step 13: Paste the **"Username"** and **"Password"** on the login window and click on **"Login"**



Step 14: Click on the **"Upload Files"** icon and upload the "LoggerAnalysis.jar**"** file in FTP



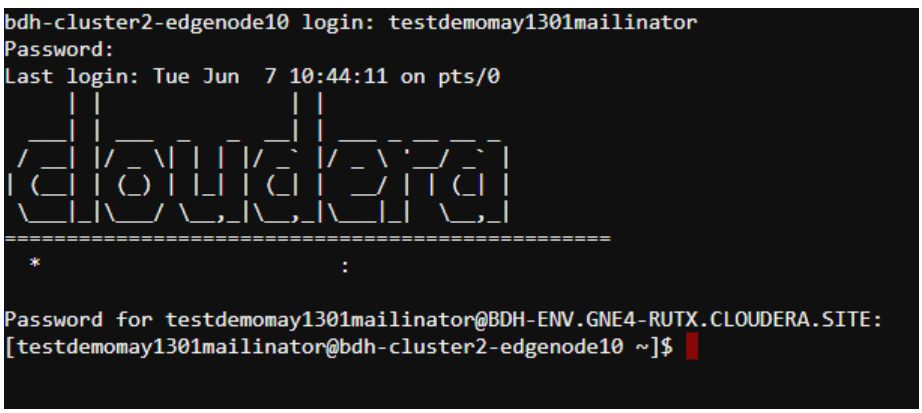
Step 15: Go back to the lab window and click on "**Webconsole**". Select **"Auth Url"**



Step 16: Copy the "**Username**" and "**Password**" provided to log in to the **"Webconsole"**

Step 17: Paste the "**Username**" and "**Password**" on the console and click on enter

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



Step 18: Make sure you have the **"LoggerAnalysis.jar"** file present in FTP using the ls command

```
[testdemomay1301mailinator@bdh-cluster2-edgenode10 ~]$ ls
13_4AP.py                    convert.py       encapsulation.py              insta-cart
13_flights_graph_case_study  data.csv         example1.py                   kmeans.py
abc                          data_files       example.py                    Lesson_13_Dataset.csv
abstractAP.py                data_files_CEP   flume.conf                    Lesson_13.ipynb
abstract.py                  demo11           hadoop-custom-demo.jar        lin_reg.py
Apache-log.log               derby.log        hadoop-mapreduce-example.jar  LoggerAnalysis.jar
classDemo.py                 dictionary.py    inheritance.py                log_reg.py
[testdemomay1301mailinator@bdh-cluster2-edgenode10 ~]$
```

Step 19: Execute the below command and see if your job gets executed successfully

Note: Change the username of the Hadoop directory to "testdemomay1301mailinator" as assigned in your lab

Command:

yarn jar LoggerAnalysis.jar /user/testdemomay1301mailinator/Analysis/Apache-log.log /user/testdemomay1301mailinator/Analysis/Output

```
[testdemomay1301mailinator@bdh-cluster2-edgenode10 ~]$ yarn jar LoggerAnalysis.jar /user/testdemomay1301mailinator/Analysis/Apache-log.log  /user/testdemomay1
301mailinator/Analysis/Output
WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN_OPTS.
```

Step 20: Log in to **HUE** and open the **"Analysis"** directory. You will find a folder named **"Output"** where you will be able to see the part files with aggregate count

Home /user/testdemomay1301mailinator/Analysis/**Output**

Name	Size	User	Group	Permissions	Date
↑		testdemomay1301maili...	hadoop	drwxr-xr-x	June 12, 2022 0
.		testdemomay1301maili...	hadoop	drwxr-xr-x	June 12, 2022 0
_SUCCESS	0 bytes	testdemomay1301maili...	hadoop	-rw-r--r--	June 12, 2022 0
part-r-00000	138 bytes	testdemomay1301maili...	hadoop	-rw-r--r--	June 12, 2022 0

Show 45 of 2 items Page 1 of 1

Edit file

Refresh

View as binary

Download

Last modified
06/13/2022 1:58 AM +05:30

User
testdemomay1301mailir

Group

/ user/ testdemomay1301mailinator/ Analysis/ Output/ **part-r-00000**

0	33
1	35
2	50
3	47
4	31
5	73
6	65
7	46
8	118
9	70
10	59
11	113
12	195