Lab 2

Divya K Raman, EE15B085

Answer 1

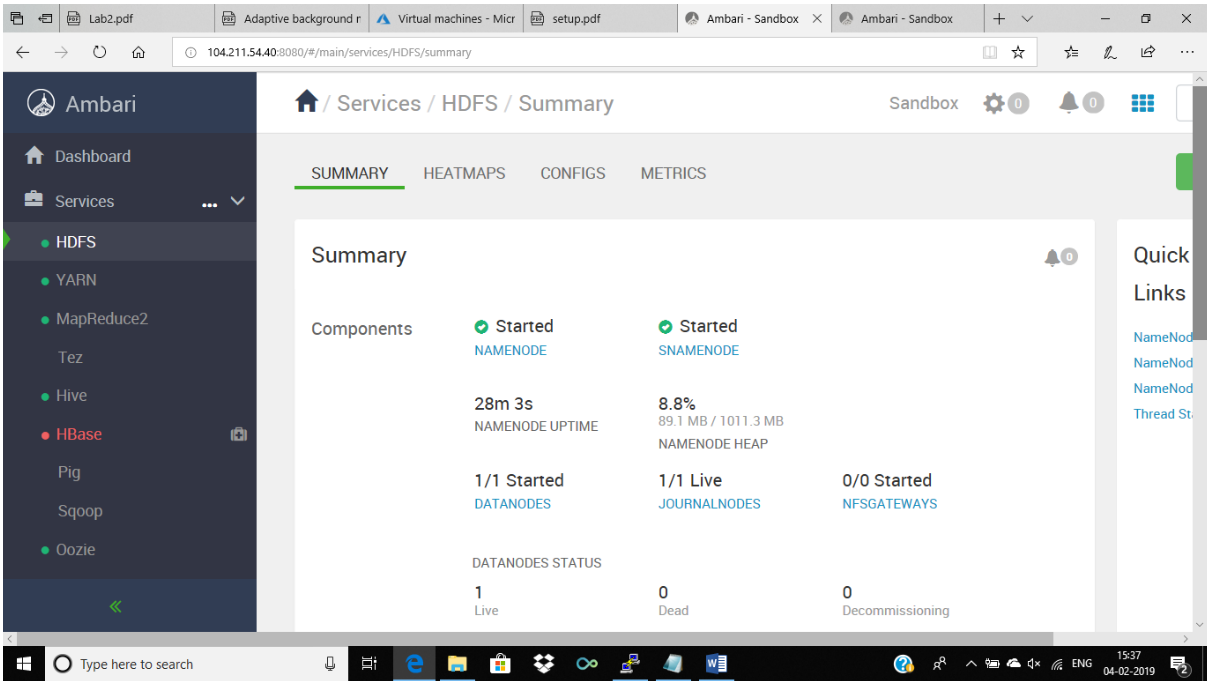
CAP Theorem is comprised of three technical terms. C stands for Consistency where all nodes see the data in homogeneous form i.e. every node has the same knowledge of data at any instant of time. A stands for availability i.e. A guarantee that every request receives a response which may be processed or failed. P stands for Partition Tolerance i.e. the system continues to operate even if a message is lost or part of the system fails Any system which satisfies two out of these three properties is a distributed system. Hadoop supports the Availability and Partition Tolerance property. The Consistency property is not supported because only namenode has the information of where the replicas are placed. This information is not available with each and every node of the cluster.

Answer 2

Apache Hive provides a database query interface which resembles SQL to data stored in databases and file systems that integrate with Hadoop. Pig enables data workers to write complex data transformations and manipulations without knowing Java. Both Pig and Hive are feature complete. Either of them can be used to do a particular task. Depending on their primary use case, data scientists generally prefer one over the other. Hive is easier for users proficient with SQL to query data whereas Pig has data flow strengths. Pig can bring data into Hadoop and get it into the form suitable for querying.

Answer 3

Before uploading any of the 4 files:



Components

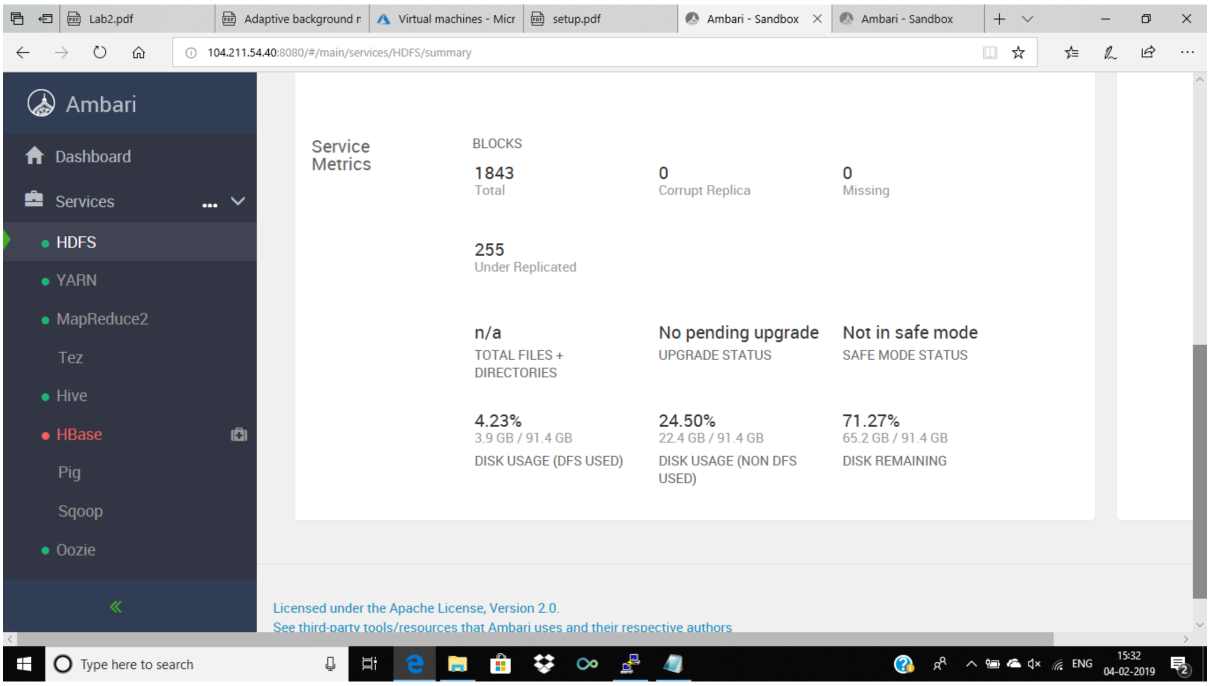
Started NAMENODE — Started SNAMENODE

28m 3s NAMENODE UPTIME — 8.8% 89.1 MB / 1011.3 MB NAMENODE HEAP

1/1 Started DATANODES — 1/1 Live JOURNALNODES — 0/0 Started NFSGATEWAYS

DATANODES STATUS

1 Live — 0 Dead — 0 Decommissioning



Service Metrics

BLOCKS

1843 Total | 0 Corrupt Replica | 0 Missing

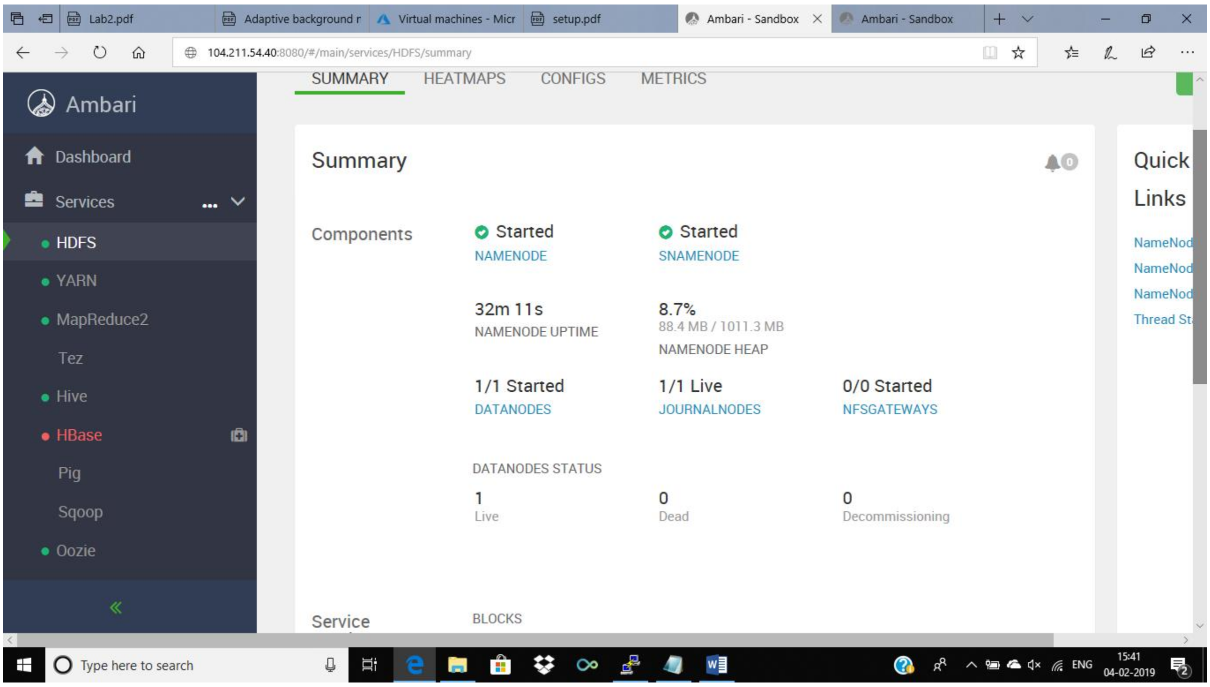
255 Under Replicated

n/a TOTAL FILES + DIRECTORIES | No pending upgrade UPGRADE STATUS | Not in safe mode SAFE MODE STATUS

4.23% 3.9 GB / 91.4 GB DISK USAGE (DFS USED) | 24.50% 22.4 GB / 91.4 GB DISK USAGE (NON DFS USED) | 71.27% 65.2 GB / 91.4 GB DISK REMAINING

Licensed under the Apache License, Version 2.0.
See third-party tools/resources that Ambari uses and their respective authors

After uploading trucks.csv



Summary

Components

Started NAMENODE | Started SNAMENODE

32m 11s NAMENODE UPTIME | 8.7% 88.4 MB / 1011.3 MB NAMENODE HEAP

1/1 Started DATANODES | 1/1 Live JOURNALNODES | 0/0 Started NFSGATEWAYS

DATANODES STATUS

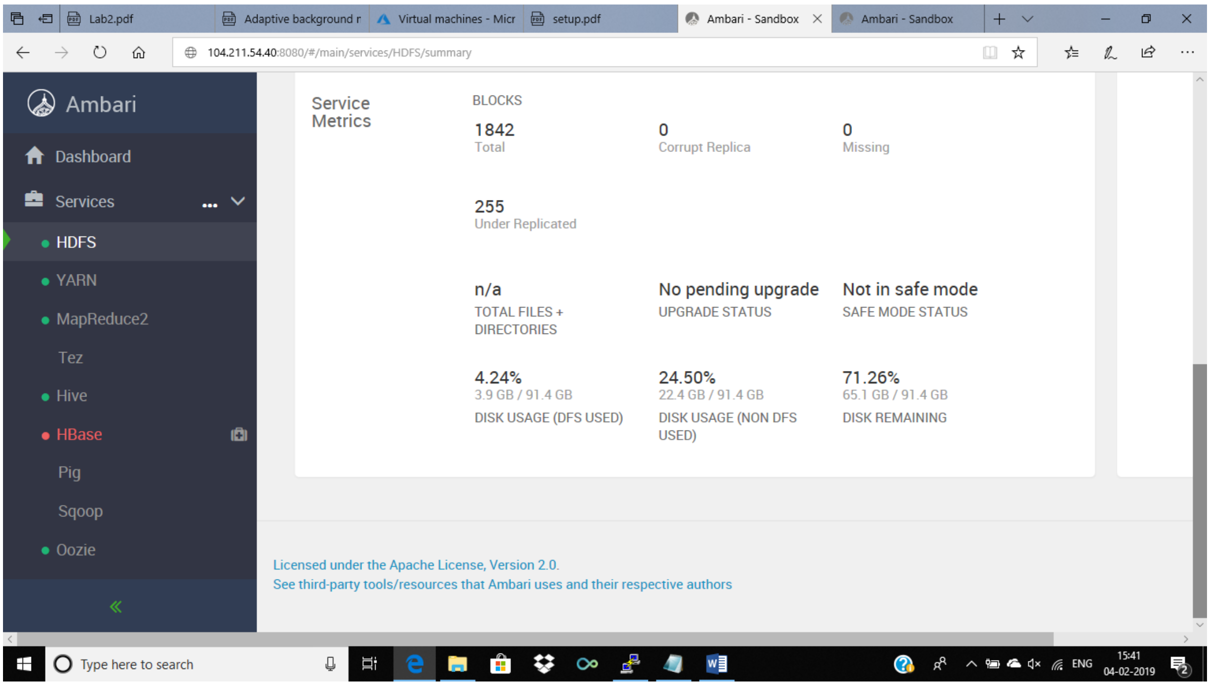
1 Live | 0 Dead | 0 Decommissioning

Service

BLOCKS

Quick Links

NameNod
NameNod
NameNod
Thread St



Service Metrics

BLOCKS

1842 Total

0 Corrupt Replica

0 Missing

255 Under Replicated

n/a TOTAL FILES + DIRECTORIES

No pending upgrade UPGRADE STATUS

Not in safe mode SAFE MODE STATUS

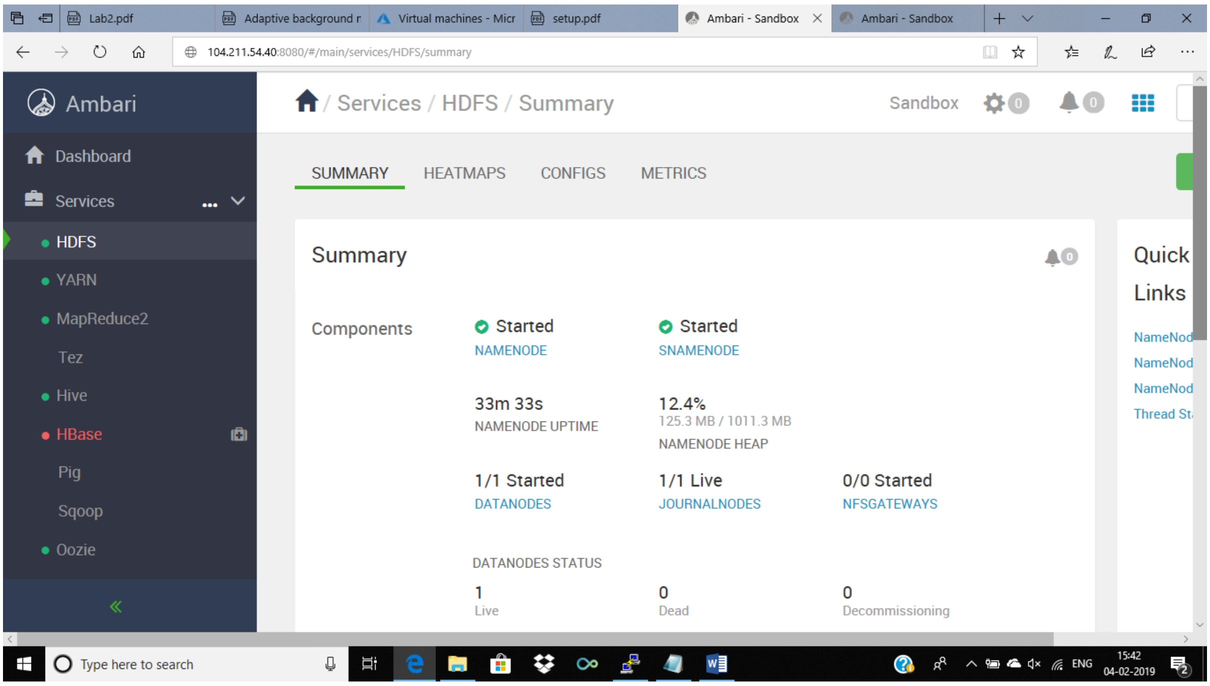
4.24% 3.9 GB / 91.4 GB DISK USAGE (DFS USED)

24.50% 22.4 GB / 91.4 GB DISK USAGE (NON DFS USED)

71.26% 65.1 GB / 91.4 GB DISK REMAINING

Licensed under the Apache License, Version 2.0.
See third-party tools/resources that Ambari uses and their respective authors

After uploading seasons.csv



Summary

Components

Started NAMENODE

Started SNAMENODE

33m 33s NAMENODE UPTIME

12.4% 125.3 MB / 1011.3 MB NAMENODE HEAP

1/1 Started DATANODES

1/1 Live JOURNALNODES

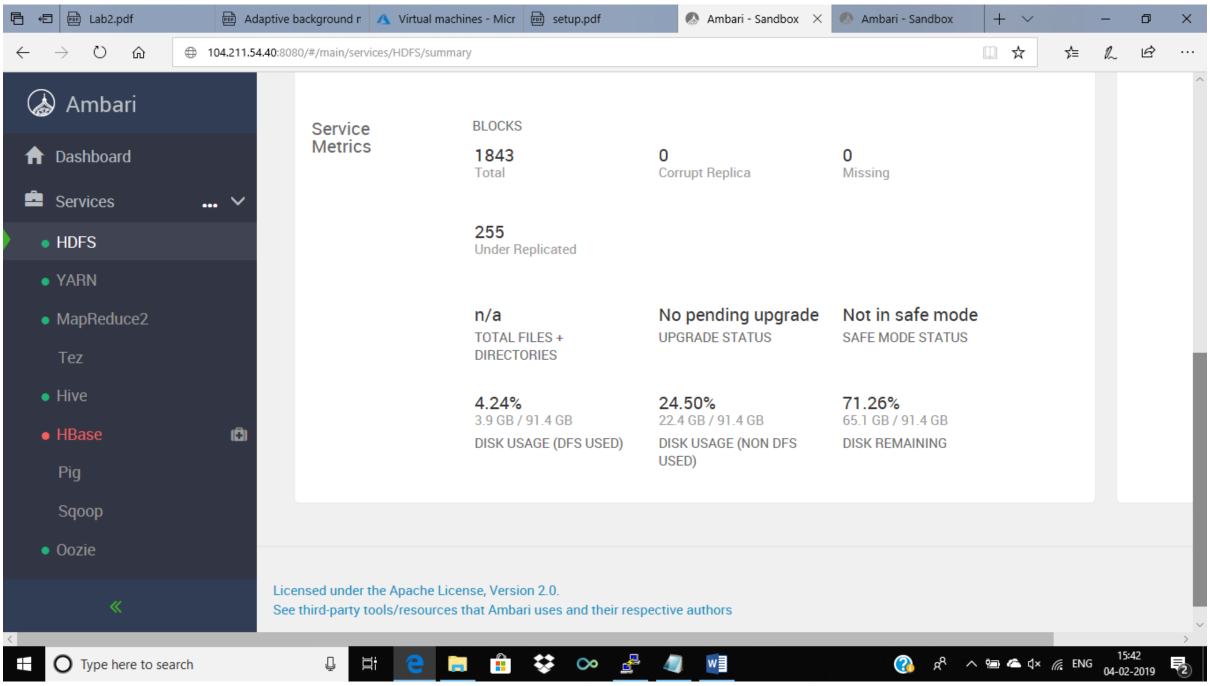
0/0 Started NFSGATEWAYS

DATANODES STATUS

1 Live

0 Dead

0 Decommissioning



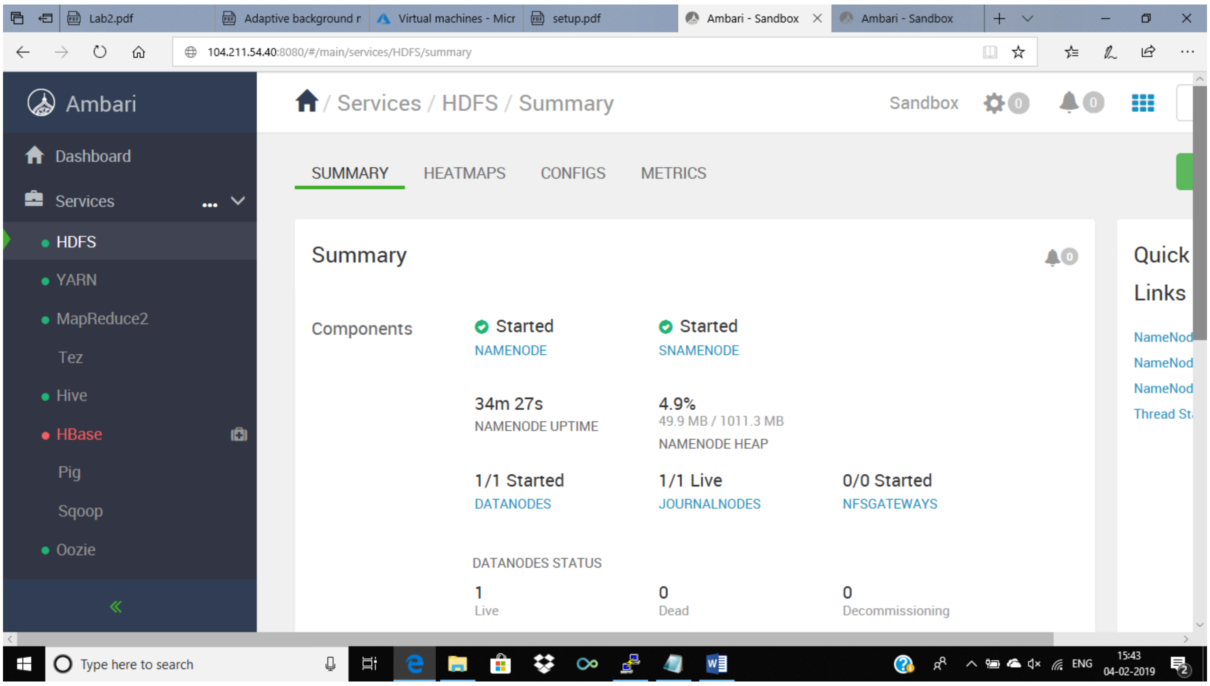
Service Metrics

BLOCKS

1843 Total	0 Corrupt Replica	0 Missing
255 Under Replicated		
n/a TOTAL FILES + DIRECTORIES	No pending upgrade UPGRADE STATUS	Not in safe mode SAFE MODE STATUS
4.24% 3.9 GB / 91.4 GB DISK USAGE (DFS USED)	24.50% 22.4 GB / 91.4 GB DISK USAGE (NON DFS USED)	71.26% 65.1 GB / 91.4 GB DISK REMAINING

Licensed under the Apache License, Version 2.0.
See third-party tools/resources that Ambari uses and their respective authors

After products.tsv



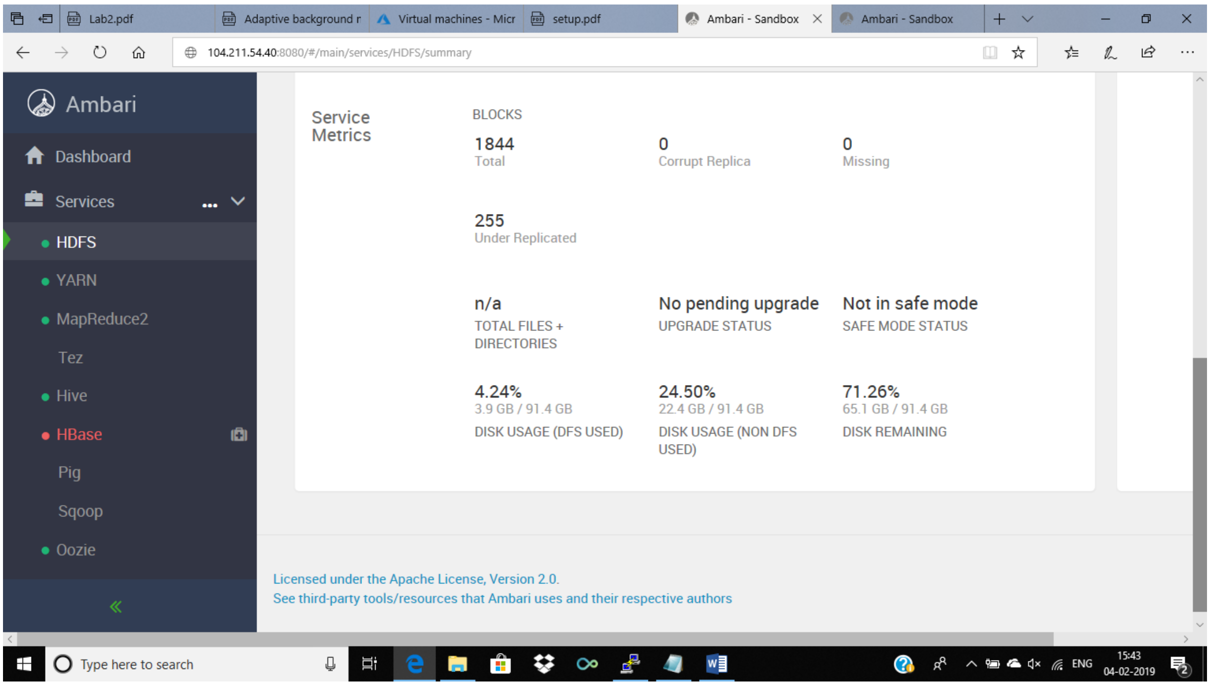
Summary

Components

Started NAMENODE	Started SNAMENODE	
34m 27s NAMENODE UPTIME	4.9% 49.9 MB / 1011.3 MB NAMENODE HEAP	
1/1 Started DATANODES	1/1 Live JOURNALNODES	0/0 Started NFSGATEWAYS

DATANODES STATUS

1 Live	0 Dead	0 Decommissioning



Service Metrics

BLOCKS

1844 Total | 0 Corrupt Replica | 0 Missing

255 Under Replicated

n/a TOTAL FILES + DIRECTORIES | No pending upgrade UPGRADE STATUS | Not in safe mode SAFE MODE STATUS

4.24% 3.9 GB / 91.4 GB DISK USAGE (DFS USED) | 24.50% 22.4 GB / 91.4 GB DISK USAGE (NON DFS USED) | 71.26% 65.1 GB / 91.4 GB DISK REMAINING

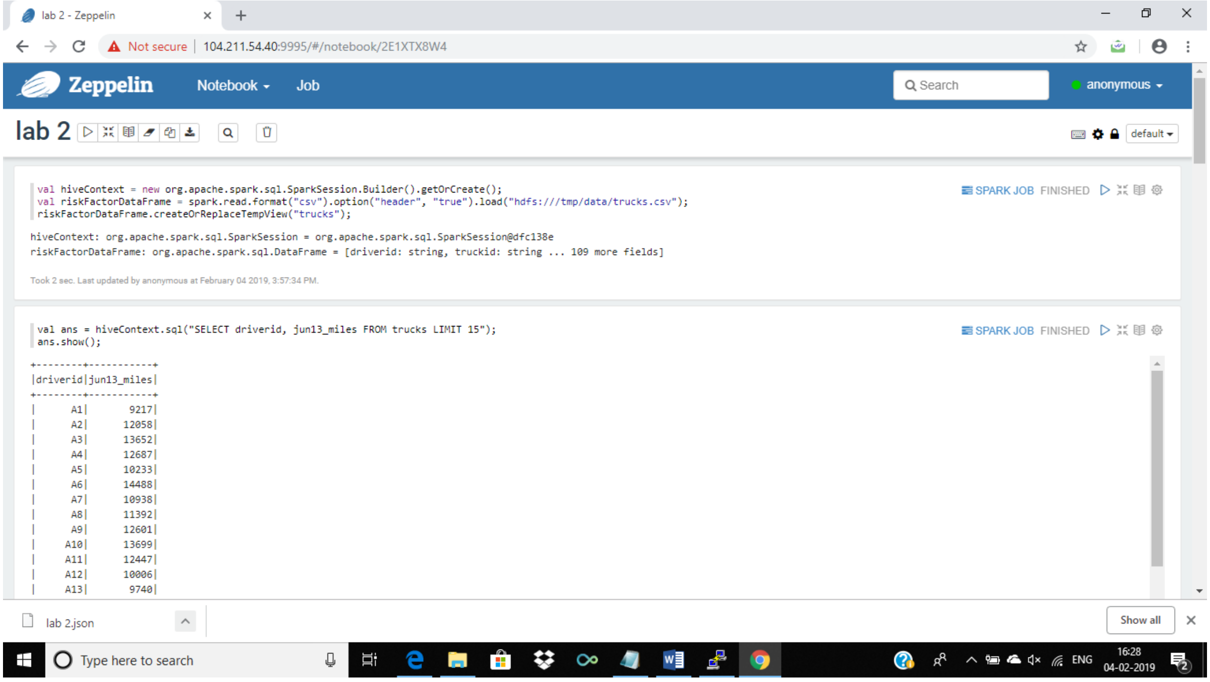
Licensed under the Apache License, Version 2.0.
See third-party tools/resources that Ambari uses and their respective authors

As files are added to the server, number of blocks increases which is understandable as new files are being added to the server. Under replicated remains the same. Under replicated blocks are blocks that do not meet their target replication for the file they belong to. This number indicates the number of copies of blocks missing from the environment. While uploading files, there were probably no copies of blocks which went missing and hence this number remains unchanged. Heap size first increases and then decreases. In Java, heap memory is an area of memory reserved for data that is created at runtime that is, when the program actually executes. NameNode heap size depends on many factors, such as the number of files, the number of blocks, and the load on the system.

Answer 4

In map, each worker node applies the map function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of redundant input data is processed. In Reduce, worker nodes now process each group of output data, per key, in parallel. In the given problem, each row describes a match officiated by a referee. In the map stage, we map the matches to the corresponding referees. In the reduce stage, we combine the matches officiated by each referee and take a total count for each referee.

Answer 5



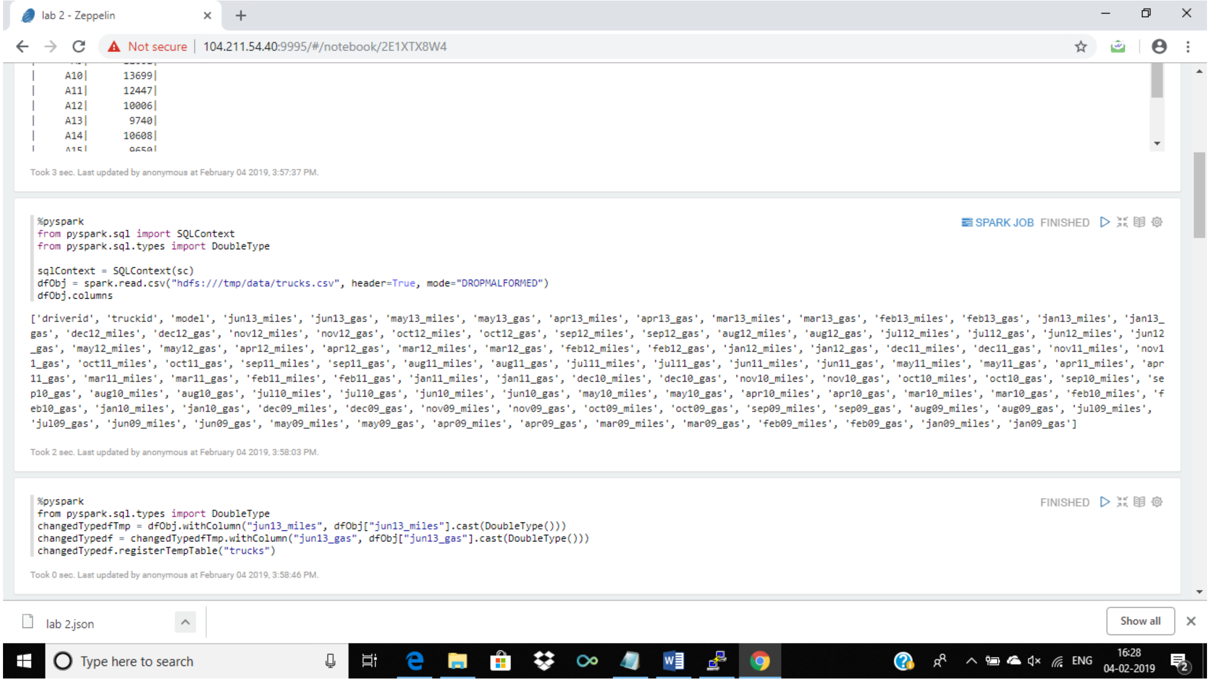
```
val hiveContext = new org.apache.spark.sql.SparkSession.Builder().getOrCreate();
val riskFactorDataFrame = spark.read.format("csv").option("header", "true").load("hdfs:///tmp/data/trucks.csv");
riskFactorDataFrame.createOrReplaceTempView("trucks");
```

```
hiveContext: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@dfc138e
riskFactorDataFrame: org.apache.spark.sql.DataFrame = [driverid: string, truckid: string ... 109 more fields]
```

Took 2 sec. Last updated by anonymous at February 04 2019, 3:57:34 PM.

```
val ans = hiveContext.sql("SELECT driverid, jun13_miles FROM trucks LIMIT 15");
ans.show();
```

```
+--------+-----------+
|driverid|jun13_miles|
+--------+-----------+
|      A1|       9217|
|      A2|      12058|
|      A3|      13652|
|      A4|      12687|
|      A5|      10233|
|      A6|      14488|
|      A7|      10938|
|      A8|      11392|
|      A9|      12601|
|     A10|      13699|
|     A11|      12447|
|     A12|      10006|
|     A13|       9740|
```



```
|     A10|      13699|
|     A11|      12447|
|     A12|      10006|
|     A13|       9740|
|     A14|      10608|
|     A15|       9650|
```

Took 3 sec. Last updated by anonymous at February 04 2019, 3:57:37 PM.

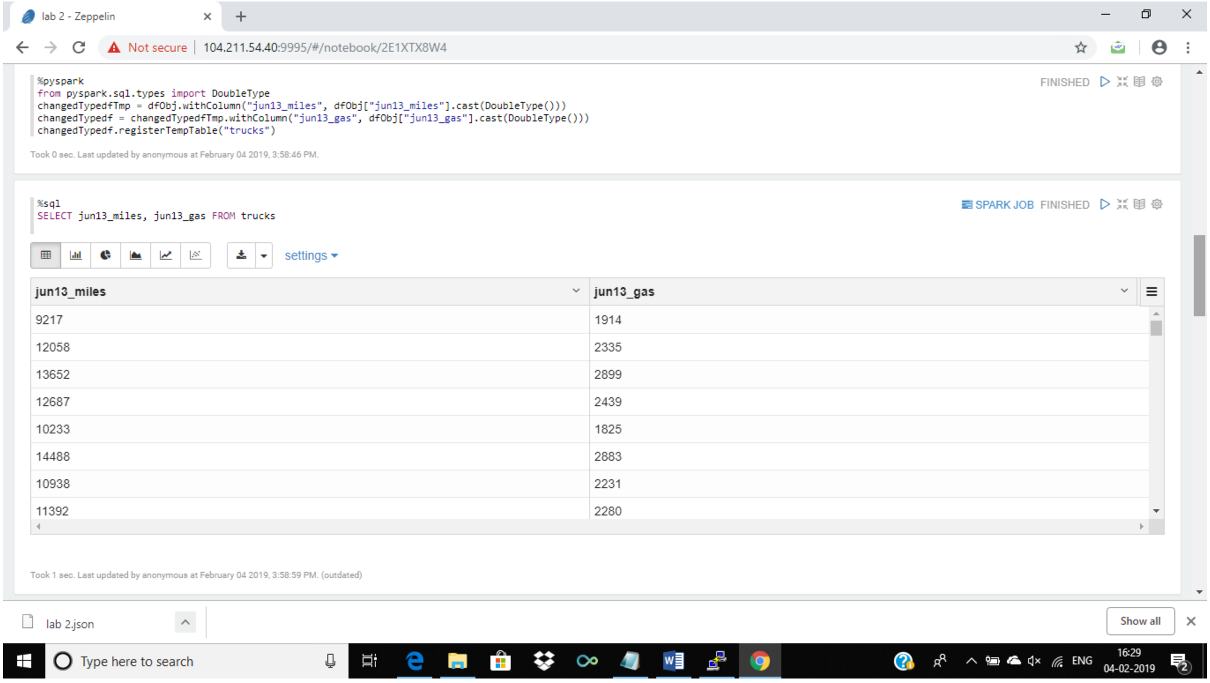
```
%pyspark
from pyspark.sql import SQLContext
from pyspark.sql.types import DoubleType

sqlContext = SQLContext(sc)
dfObj = spark.read.csv("hdfs:///tmp/data/trucks.csv", header=True, mode="DROPMALFORMED")
dfObj.columns
```

```
['driverid', 'truckid', 'model', 'jun13_miles', 'jun13_gas', 'may13_miles', 'may13_gas', 'apr13_miles', 'apr13_gas', 'mar13_miles', 'mar13_gas', 'feb13_miles', 'feb13_gas', 'jan13_miles', 'jan13_gas', 'dec12_miles', 'dec12_gas', 'nov12_miles', 'nov12_gas', 'oct12_miles', 'oct12_gas', 'sep12_miles', 'sep12_gas', 'aug12_miles', 'aug12_gas', 'jul12_miles', 'jul12_gas', 'jun12_miles', 'jun12_gas', 'may12_miles', 'may12_gas', 'apr12_miles', 'apr12_gas', 'mar12_miles', 'mar12_gas', 'feb12_miles', 'feb12_gas', 'jan12_miles', 'jan12_gas', 'dec11_miles', 'dec11_gas', 'nov11_miles', 'nov11_gas', 'oct11_miles', 'oct11_gas', 'sep11_miles', 'sep11_gas', 'aug11_miles', 'aug11_gas', 'jul11_miles', 'jul11_gas', 'jun11_miles', 'jun11_gas', 'may11_miles', 'may11_gas', 'apr11_miles', 'apr11_gas', 'mar11_miles', 'mar11_gas', 'feb11_miles', 'feb11_gas', 'jan11_miles', 'jan11_gas', 'dec10_miles', 'dec10_gas', 'nov10_miles', 'nov10_gas', 'oct10_miles', 'oct10_gas', 'sep10_miles', 'sep10_gas', 'aug10_miles', 'aug10_gas', 'jul10_miles', 'jul10_gas', 'jun10_miles', 'jun10_gas', 'may10_miles', 'may10_gas', 'apr10_miles', 'apr10_gas', 'mar10_miles', 'mar10_gas', 'feb10_miles', 'feb10_gas', 'jan10_miles', 'jan10_gas', 'dec09_miles', 'dec09_gas', 'nov09_miles', 'nov09_gas', 'oct09_miles', 'oct09_gas', 'sep09_miles', 'sep09_gas', 'aug09_miles', 'aug09_gas', 'jul09_miles', 'jul09_gas', 'jun09_miles', 'jun09_gas', 'may09_miles', 'may09_gas', 'apr09_miles', 'apr09_gas', 'mar09_miles', 'mar09_gas', 'feb09_miles', 'feb09_gas', 'jan09_miles', 'jan09_gas']
```

Took 2 sec. Last updated by anonymous at February 04 2019, 3:58:03 PM.

```
%pyspark
from pyspark.sql.types import DoubleType
changedTypedfTmp = dfObj.withColumn("jun13_miles", dfObj["jun13_miles"].cast(DoubleType()))
changedTypedf = changedTypedfTmp.withColumn("jun13_gas", dfObj["jun13_gas"].cast(DoubleType()))
changedTypedf.registerTempTable("trucks")
```

Took 0 sec. Last updated by anonymous at February 04 2019, 3:58:46 PM.



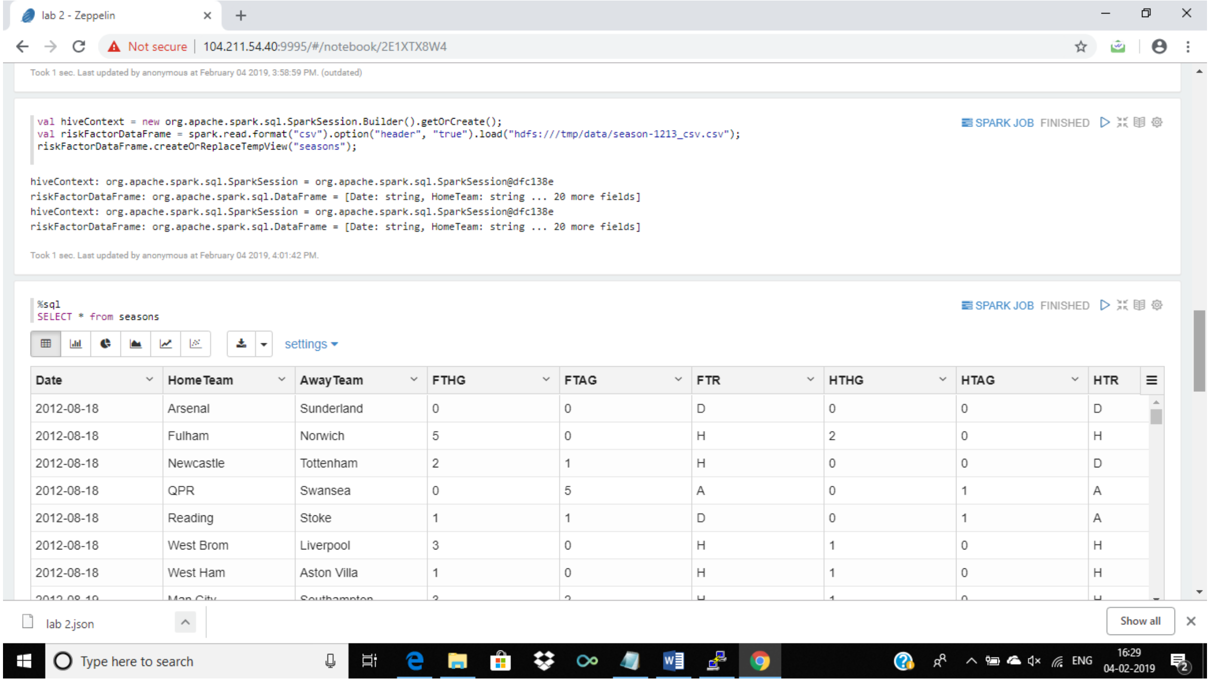
```
%pyspark
from pyspark.sql.types import DoubleType
changedTypedfTmp = dfObj.withColumn("jun13_miles", dfObj["jun13_miles"].cast(DoubleType()))
changedTypedf = changedTypedfTmp.withColumn("jun13_gas", dfObj["jun13_gas"].cast(DoubleType()))
changedTypedf.registerTempTable("trucks")
```

Took 0 sec. Last updated by anonymous at February 04 2019, 3:58:46 PM.

```
%sql
SELECT jun13_miles, jun13_gas FROM trucks
```

jun13_miles	jun13_gas
9217	1914
12058	2335
13652	2899
12687	2439
10233	1825
14488	2883
10938	2231
11392	2280

Took 1 sec. Last updated by anonymous at February 04 2019, 3:58:59 PM. (outdated)



Took 1 sec. Last updated by anonymous at February 04 2019, 3:58:59 PM. (outdated)

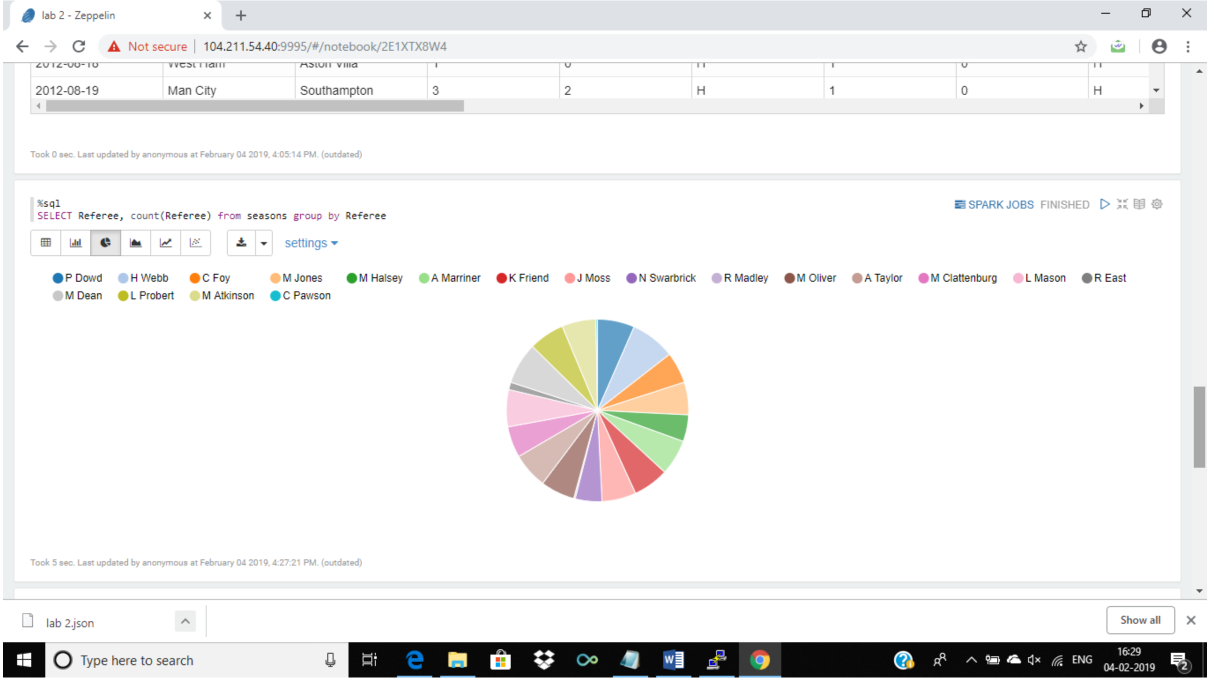
```
val hiveContext = new org.apache.spark.sql.SparkSession.Builder().getOrCreate();
val riskFactorDataFrame = spark.read.format("csv").option("header", "true").load("hdfs:///tmp/data/season-1213_csv.csv");
riskFactorDataFrame.createOrReplaceTempView("seasons");
```

```
hiveContext: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@dfc138e
riskFactorDataFrame: org.apache.spark.sql.DataFrame = [Date: string, HomeTeam: string ... 20 more fields]
hiveContext: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@dfc138e
riskFactorDataFrame: org.apache.spark.sql.DataFrame = [Date: string, HomeTeam: string ... 20 more fields]
```

Took 1 sec. Last updated by anonymous at February 04 2019, 4:01:42 PM.

```
%sql
SELECT * from seasons
```

Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR
2012-08-18	Arsenal	Sunderland	0	0	D	0	0	D
2012-08-18	Fulham	Norwich	5	0	H	2	0	H
2012-08-18	Newcastle	Tottenham	2	1	H	0	0	D
2012-08-18	QPR	Swansea	0	5	A	0	1	A
2012-08-18	Reading	Stoke	1	1	D	0	1	A
2012-08-18	West Brom	Liverpool	3	0	H	1	0	H
2012-08-18	West Ham	Aston Villa	1	0	H	1	0	H



lab 2 - Zeppelin

Not secure | 104.211.54.40:9995/#/notebook/2E1XTX8W4

2012-08-18	West Ham	Aston Villa	1	0	H	1	0	H
2012-08-19	Man City	Southampton	3	2	H	1	0	H

Took 0 sec. Last updated by anonymous at February 04 2019, 4:05:14 PM. (outdated)

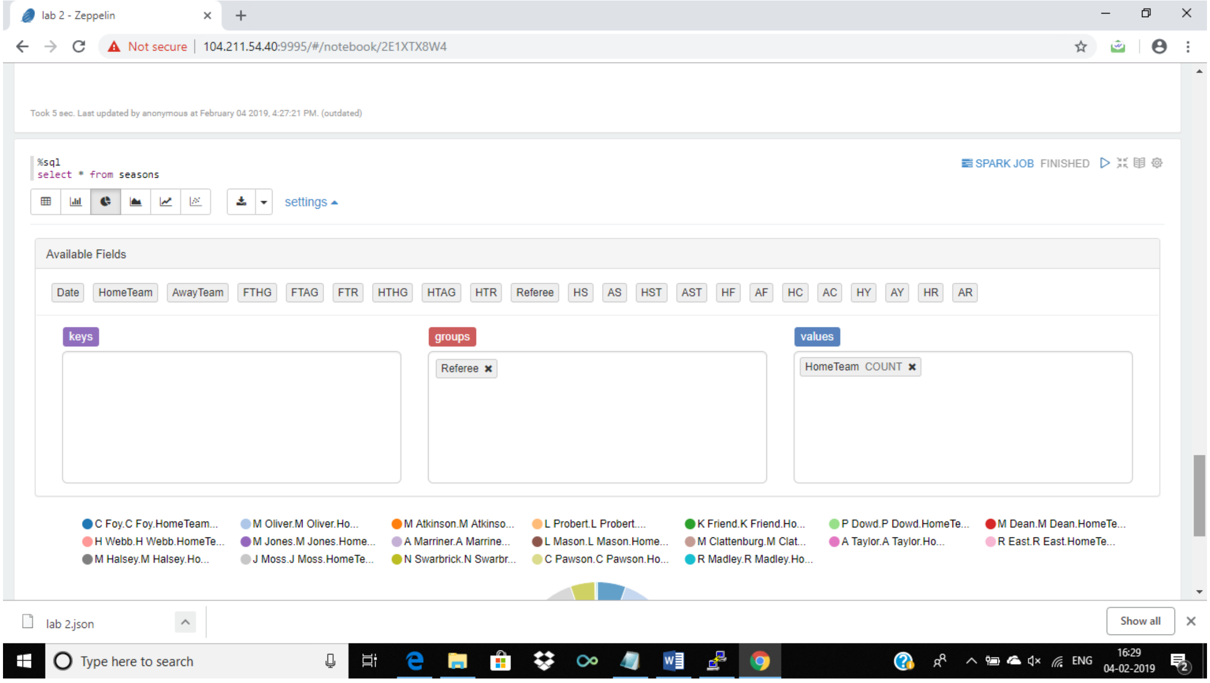
```
%sql
SELECT Referee, count(Referee) from seasons group by Referee
```

SPARK JOBS FINISHED

settings
● P Dowd ● H Webb ● C Foy ● M Jones ● M Halsey ● A Marriner ● K Friend ● J Moss ● N Swarbrick ● R Madley ● M Oliver ● A Taylor ● M Clattenburg ● L Mason ● R East
● M Dean ● L Probert ● M Atkinson ● C Pawson

Took 5 sec. Last updated by anonymous at February 04 2019, 4:27:21 PM. (outdated)

lab 2.json
Show all
Type here to search
ENG
16:29
04-02-2019



lab 2 - Zeppelin

Not secure | 104.211.54.40:9995/#/notebook/2E1XTX8W4

Took 5 sec. Last updated by anonymous at February 04 2019, 4:27:21 PM. (outdated)

```
%sql
select * from seasons
```

SPARK JOB FINISHED

settings

Available Fields

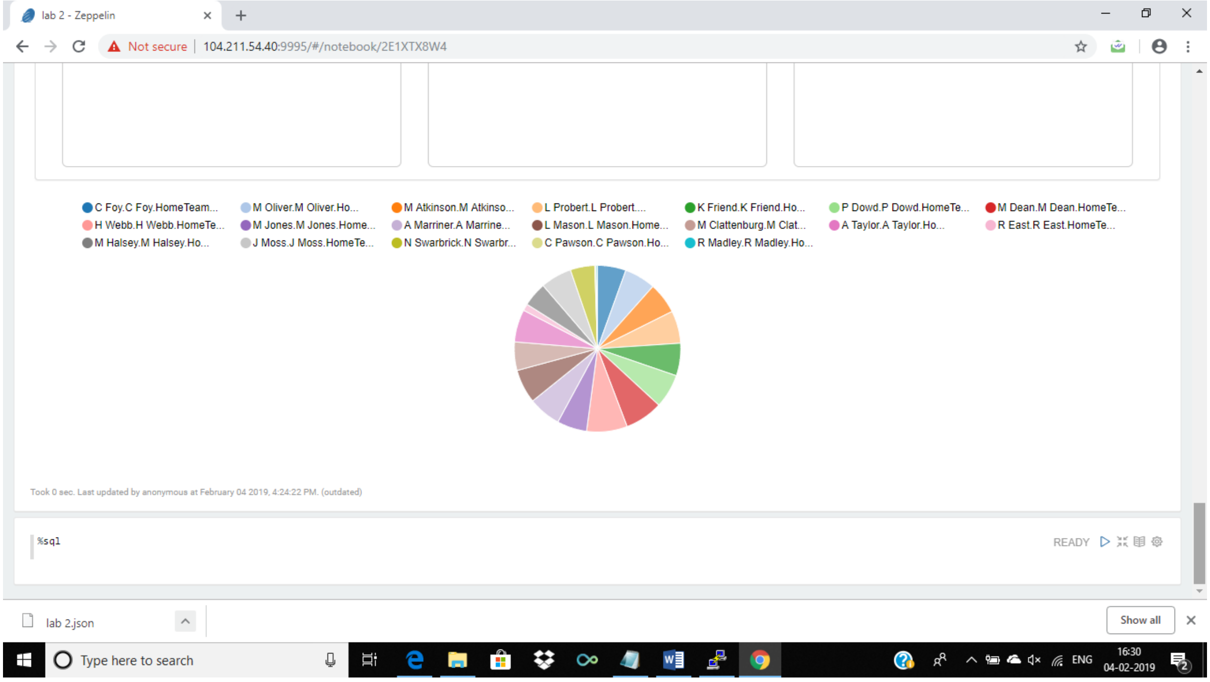
Date HomeTeam AwayTeam FTHG FTAG FTR HTHG HTAG HTR Referee HS AS HST AST HF AF HC AC HY AY HR AR

keys

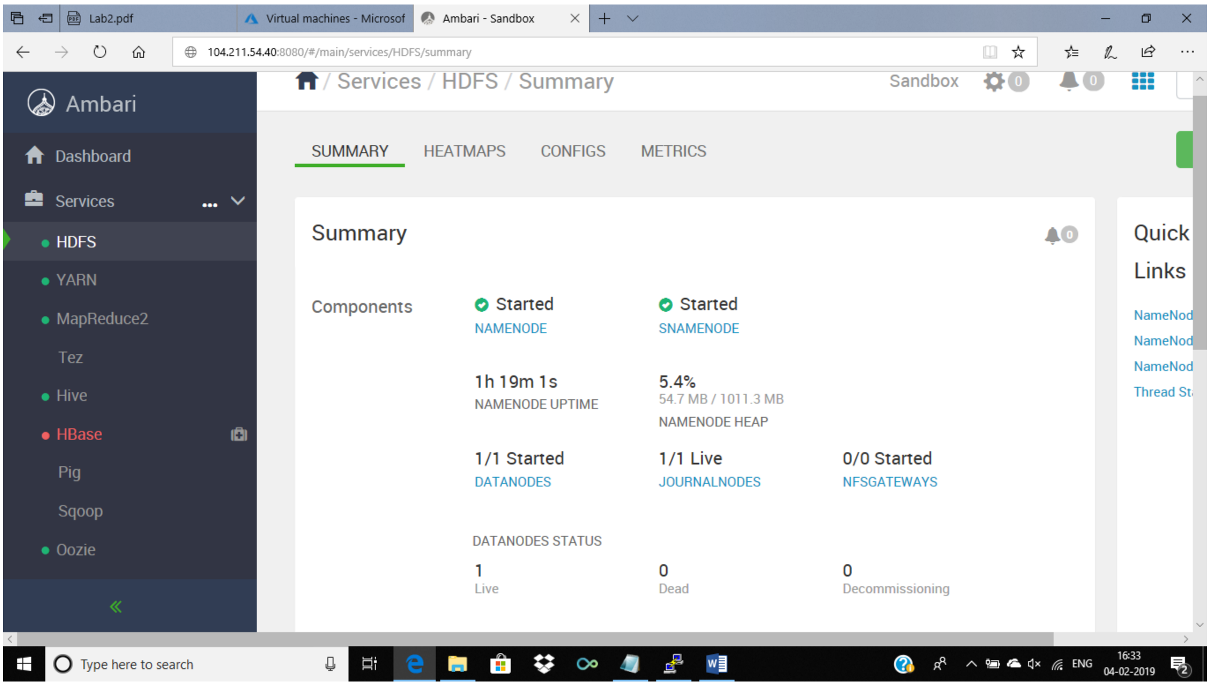
groups: Referee ✖

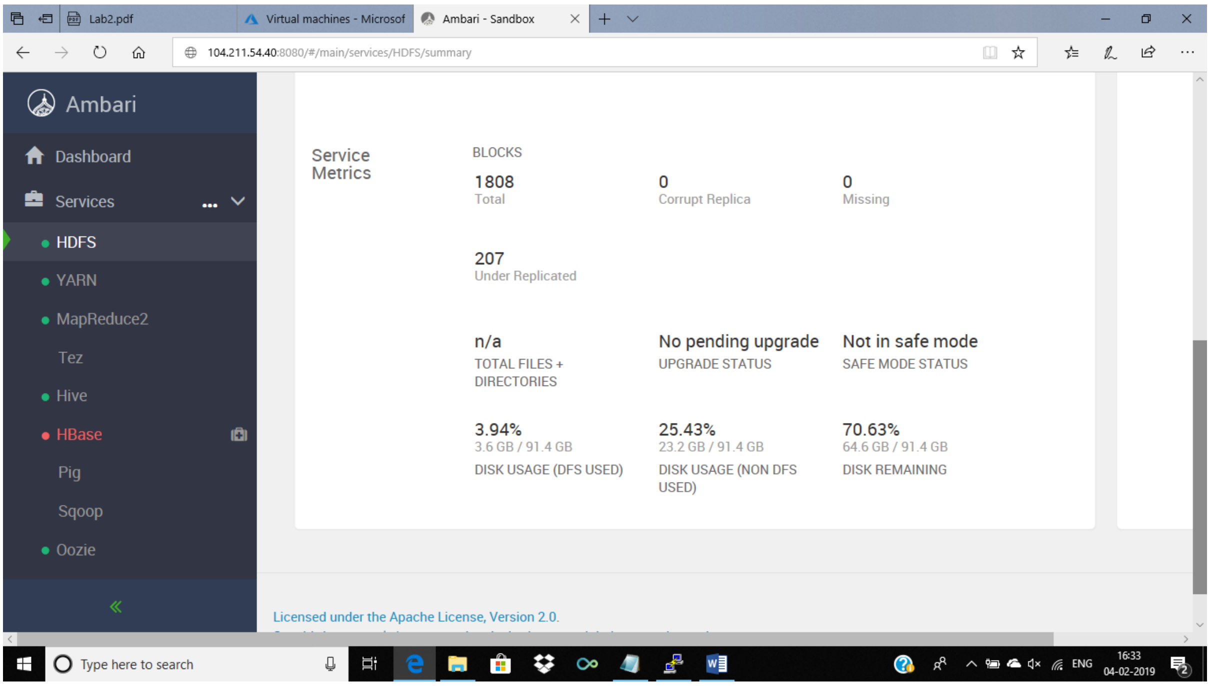
values: HomeTeam COUNT ✖

● C Foy.C Foy.HomeTeam... ● M Oliver.M Oliver.Ho... ● M Atkinson.M Atkinso... ● L Probert.L Probert.... ● K Friend.K Friend.Ho... ● P Dowd.P Dowd.HomeTe... ● M Dean.M Dean.HomeTe...
● H Webb.H Webb.HomeTe... ● M Jones.M Jones.Home... ● A Marriner.A Marrine... ● L Mason.L Mason.Home... ● M Clattenburg.M Clat... ● A Taylor.A Taylor.Ho... ● R East.R East.HomeTe...
● M Halsey.M Halsey.Ho... ● J Moss.J Moss.HomeTe... ● N Swarbrick.N Swarbr... ● C Pawson.C Pawson.Ho... ● R Madley.R Madley.Ho...
lab 2.json
Show all
Type here to search
ENG
16:29
04-02-2019


lab 2 - Zeppelin
Not secure | 104.211.54.40:9995/#/notebook/2E1XTX8W4
● C Foy.C Foy.HomeTeam...
● H Webb.H Webb.HomeTe...
● M Halsey.M Halsey.Ho...
● M Oliver.M Oliver.Ho...
● M Jones.M Jones.Home...
● J Moss.J Moss.HomeTe...
● M Atkinson.M Atkinso...
● A Marriner.A Marrine...
● N Swarbrick.N Swarbr...
● L Probert.L Probert....
● L Mason.L Mason.Home...
● C Pawson.C Pawson.Ho...
● K Friend.K Friend.Ho...
● M Clattenburg.M Clat...
● R Madley.R Madley.Ho...
● P Dowd.P Dowd.HomeTe...
● A Taylor.A Taylor.Ho...
● M Dean.M Dean.HomeTe...
● R East.R East.HomeTe...
Took 0 sec. Last updated by anonymous at February 04 2019, 4:24:22 PM. (outdated)
%sql
READY
lab 2.json
Show all
Type here to search
ENG
16:30
04-02-2019

Number of blocks after running the notebooks:





Service Metrics

BLOCKS

1808 Total

0 Corrupt Replica

0 Missing

207 Under Replicated

n/a
TOTAL FILES + DIRECTORIES

No pending upgrade
UPGRADE STATUS

Not in safe mode
SAFE MODE STATUS

3.94%
3.6 GB / 91.4 GB
DISK USAGE (DFS USED)

25.43%
23.2 GB / 91.4 GB
DISK USAGE (NON DFS USED)

70.63%
64.6 GB / 91.4 GB
DISK REMAINING

Licensed under the Apache License, Version 2.0.

References:

https://data-flair.training/forums/topic/what-is-cap-theorem-what-aspects-hadoop-supports-from-this-theorem/

https://stackoverflow.com/questions/19923196/cap-with-distributed-system

https://hortonworks.com/tutorial/how-to-process-data-with-apache-hive/

https://hortonworks.com/tutorial/beginners-guide-to-apache-pig/

https://stackoverflow.com/questions/36977746/in-hadoop-whats-under-replication-and-over-replication-mean-and-how-does-it-work

https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_command-line-installation/content/configuring-namenode-heap-size.html