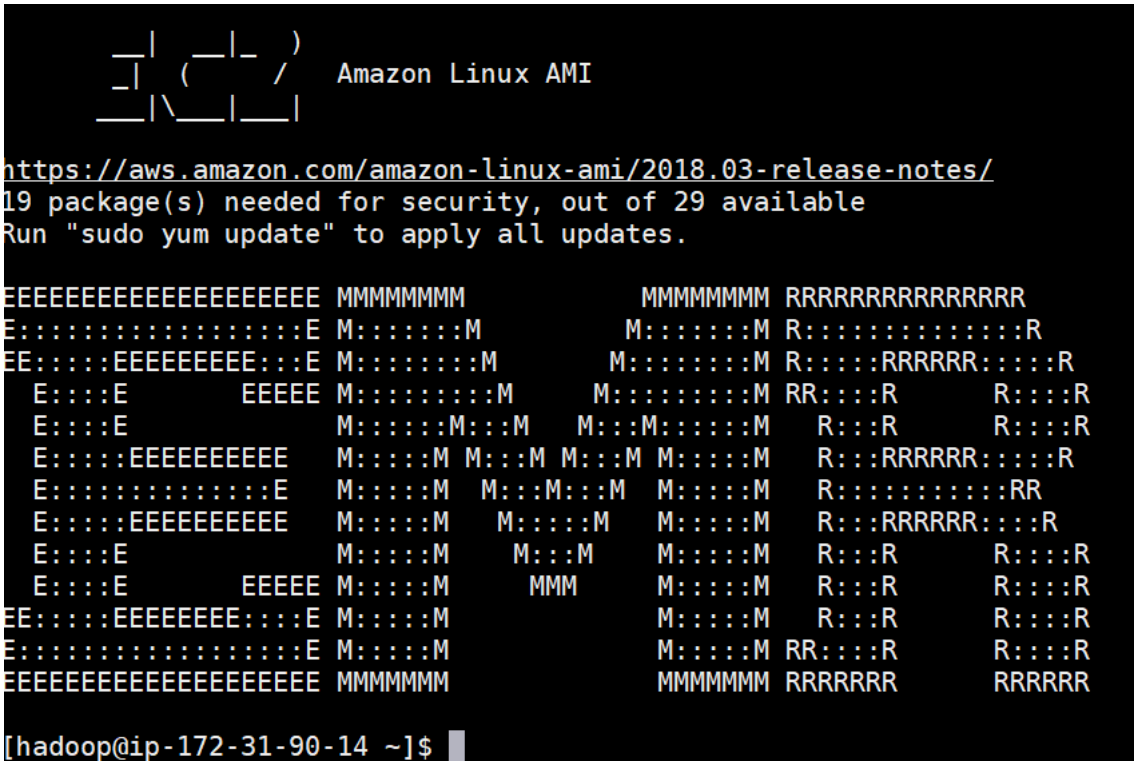
Hands-on Lab

- SSH login to our Hadoop cluster
  - Access info will be posted



Basic HDFS Commands

Command	Description
-ls <args>	List directory
-mkdir <paths>	Create a directory
-put <localsrc> <HDFS_dest_Path>	Upload files
-get <hdfs_src> <localdst>	Download file
-cat <path[filename]>	See content of files
-cp <source> <dest>	Copy files in HDFS
-rm <arg>	Remove files or directories
-tail <path[filename]>	Display last few lines of a file
-getmerge [hdfs_src_dir] [hdfs_dst_file]	Merge files (from reducers)

- $hadoop fs [command]
- Ref: https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

Steps to Prepare Input Files

- Copy files to your home directory & switch to the dir

```
$ cp -R /home/hadoop/lab .
$ cd lab
```

- Create your own folder on HDFS:

```
$ hadoop fs -mkdir /user/hadoop/[username]
```

- Copy input test files from local file system to your HDFS

```
$ hadoop fs -put input /user/hadoop/[username]/
```

- List the input files on HDFS

```
$ hadoop fs -ls /user/hadoop/[username]/input/
```

Steps to Prepare Input Files

```
-rw-r--r--   1 jchou hadoop     144860 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-first-51.txt
-rw-r--r--   1 jchou hadoop     182399 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-hamlet-25.txt
-rw-r--r--   1 jchou hadoop     117902 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-julius-26.txt
-rw-r--r--   1 jchou hadoop     157094 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-king-45.txt
-rw-r--r--   1 jchou hadoop     154933 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-life-54.txt
-rw-r--r--   1 jchou hadoop     148351 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-life-55.txt
-rw-r--r--   1 jchou hadoop     122448 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-life-56.txt
-rw-r--r--   1 jchou hadoop      14364 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-lovers-62.txt
-rw-r--r--   1 jchou hadoop     129916 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-loves-8.txt
-rw-r--r--   1 jchou hadoop     105202 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-macbeth-46.txt
-rw-r--r--   1 jchou hadoop     130363 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-measure-13.txt
-rw-r--r--   1 jchou hadoop     122508 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-merchant-5.txt
-rw-r--r--   1 jchou hadoop     131401 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-merry-15.txt
-rw-r--r--   1 jchou hadoop      96439 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-midsummer-16.txt
-rw-r--r--   1 jchou hadoop     123284 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-much-3.txt
-rw-r--r--   1 jchou hadoop     156338 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-othello-47.txt
-rw-r--r--   1 jchou hadoop     111421 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-pericles-21.txt
-rw-r--r--   1 jchou hadoop      84687 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-rape-61.txt
-rw-r--r--   1 jchou hadoop     144138 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-romeo-48.txt
-rw-r--r--   1 jchou hadoop     157146 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-second-52.txt
-rw-r--r--   1 jchou hadoop      95659 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-sonnets-59.txt
-rw-r--r--   1 jchou hadoop     124128 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-taming-2.txt
-rw-r--r--   1 jchou hadoop      99303 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-tempest-4.txt
-rw-r--r--   1 jchou hadoop     148008 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-third-53.txt
-rw-r--r--   1 jchou hadoop     113037 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-timon-49.txt
-rw-r--r--   1 jchou hadoop     123897 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-titus-50.txt
-rw-r--r--   1 jchou hadoop     134743 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-tragedy-57.txt
-rw-r--r--   1 jchou hadoop     180293 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-tragedy-58.txt
-rw-r--r--   1 jchou hadoop     158763 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-troilus-22.txt
-rw-r--r--   1 jchou hadoop     116626 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-twelfth-20.txt
-rw-r--r--   1 jchou hadoop     101862 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-two-18.txt
-rw-r--r--   1 jchou hadoop      54386 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-venus-60.txt
-rw-r--r--   1 jchou hadoop     145677 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-winters-19.txt
```

Steps to Compile & Run Jobs

- Compile the Hadoop java files

```
$ javac -classpath `hadoop classpath` WordCount.java -d bin
```

- Create a jar file for the executable

```
$ jar -cvf WordCount.jar -C bin .
```

It is normal to see the error: "No such file or directory" for your first time to run

- Remove the output folder on HDFS

```
$ hadoop fs -rm -R /user/hadoop/[username]/output
```

- Run the Hadoop job

```
$ hadoop jar WordCount.jar org.myorg.WordCount
/user/hadoop/[username]/input
/user/hadoop/[username]/output
```

Steps to Compile & Run Jobs

```
19/12/06 17:26:46 INFO mapreduce.Job:  map 70% reduce 7%
19/12/06 17:26:49 INFO mapreduce.Job:  map 73% reduce 7%
19/12/06 17:26:50 INFO mapreduce.Job:  map 73% reduce 8%
19/12/06 17:26:51 INFO mapreduce.Job:  map 75% reduce 8%
19/12/06 17:26:52 INFO mapreduce.Job:  map 77% reduce 8%
19/12/06 17:26:53 INFO mapreduce.Job:  map 80% reduce 8%
19/12/06 17:26:56 INFO mapreduce.Job:  map 82% reduce 9%
19/12/06 17:26:58 INFO mapreduce.Job:  map 85% reduce 9%
19/12/06 17:26:59 INFO mapreduce.Job:  map 88% reduce 9%
19/12/06 17:27:00 INFO mapreduce.Job:  map 90% reduce 9%
19/12/06 17:27:02 INFO mapreduce.Job:  map 90% reduce 10%
19/12/06 17:27:03 INFO mapreduce.Job:  map 93% reduce 10%
19/12/06 17:27:05 INFO mapreduce.Job:  map 95% reduce 10%
19/12/06 17:27:06 INFO mapreduce.Job:  map 100% reduce 10%
19/12/06 17:27:08 INFO mapreduce.Job:  map 100% reduce 33%
19/12/06 17:27:10 INFO mapreduce.Job:  map 100% reduce 67%
19/12/06 17:27:11 INFO mapreduce.Job:  map 100% reduce 100%
19/12/06 17:27:12 INFO mapreduce.Job: Job job_1575649265022_0005 completed successfully
19/12/06 17:27:12 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=1009048
                FILE: Number of bytes written=10539504
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=4978889
```

Steps to Check Output Results

- Show the content of the output files on HDFS

```
$ hadoop fs -ls /user/hadoop/[username]/output
$ hadoop fs -cat /user/hadoop/[username]/output/part-r-00002
```

- Merge & get the output files to local file system

```
$ hadoop fs -getmerge /user/hadoop/[username]/output output.txt
```

- Show the content of the output files to TA

```
$ tail output.txt
```

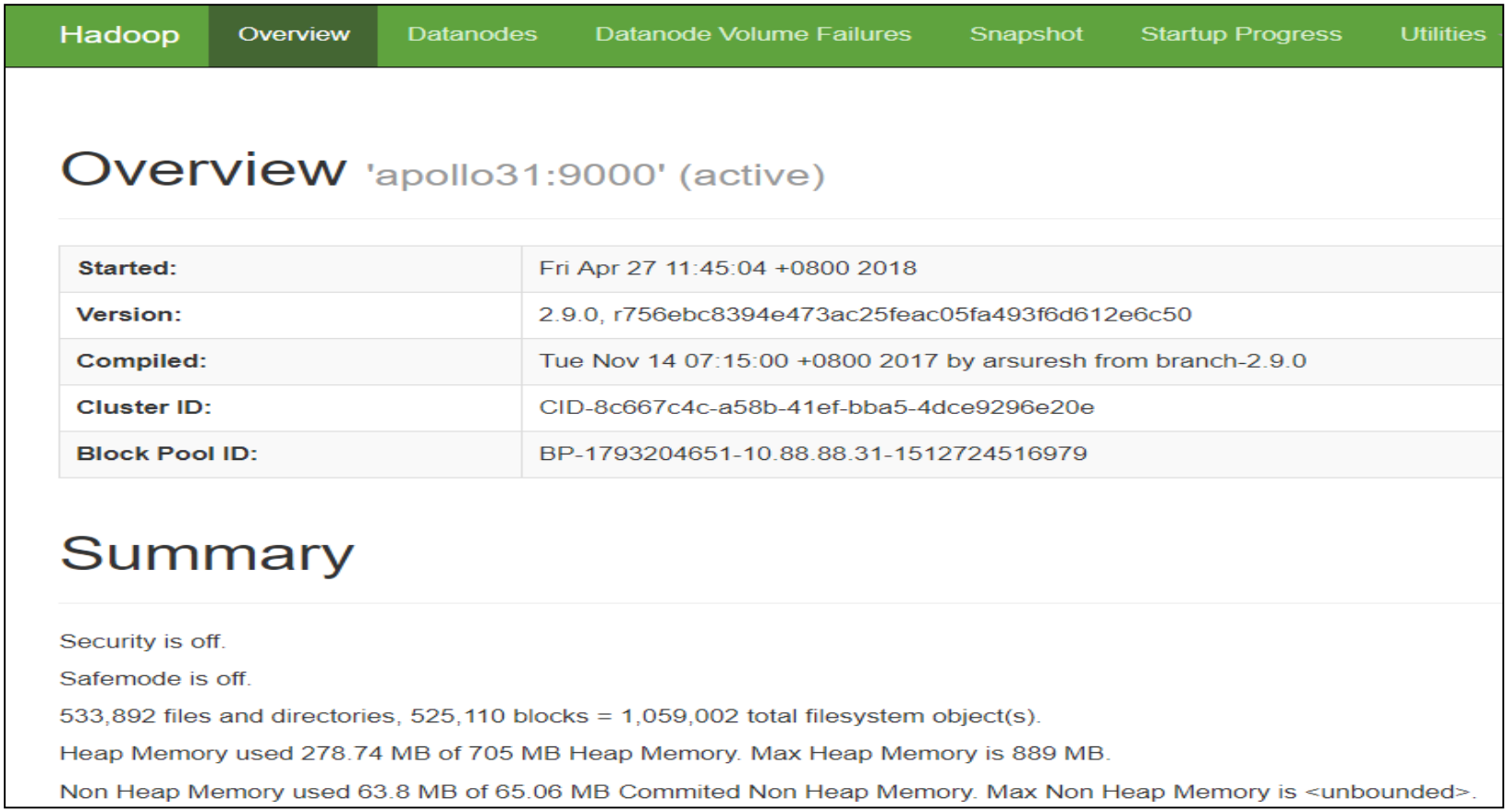
Hadoop Web UI

- Cluster Hadoop Status
  - http://[Hadoop Master IP]:9870
- MapReduce Job Tracker
  - http://[Hadoop Master IP]:8088
- Job History Server
  - http://[Hadoop Master IP]:19888

You must follow the instructions below to access the web portal on your EMR master
https://docs.aws.amazon.com/zh_tw/emr/latest/ManagementGuide/emr-web-interfaces.html

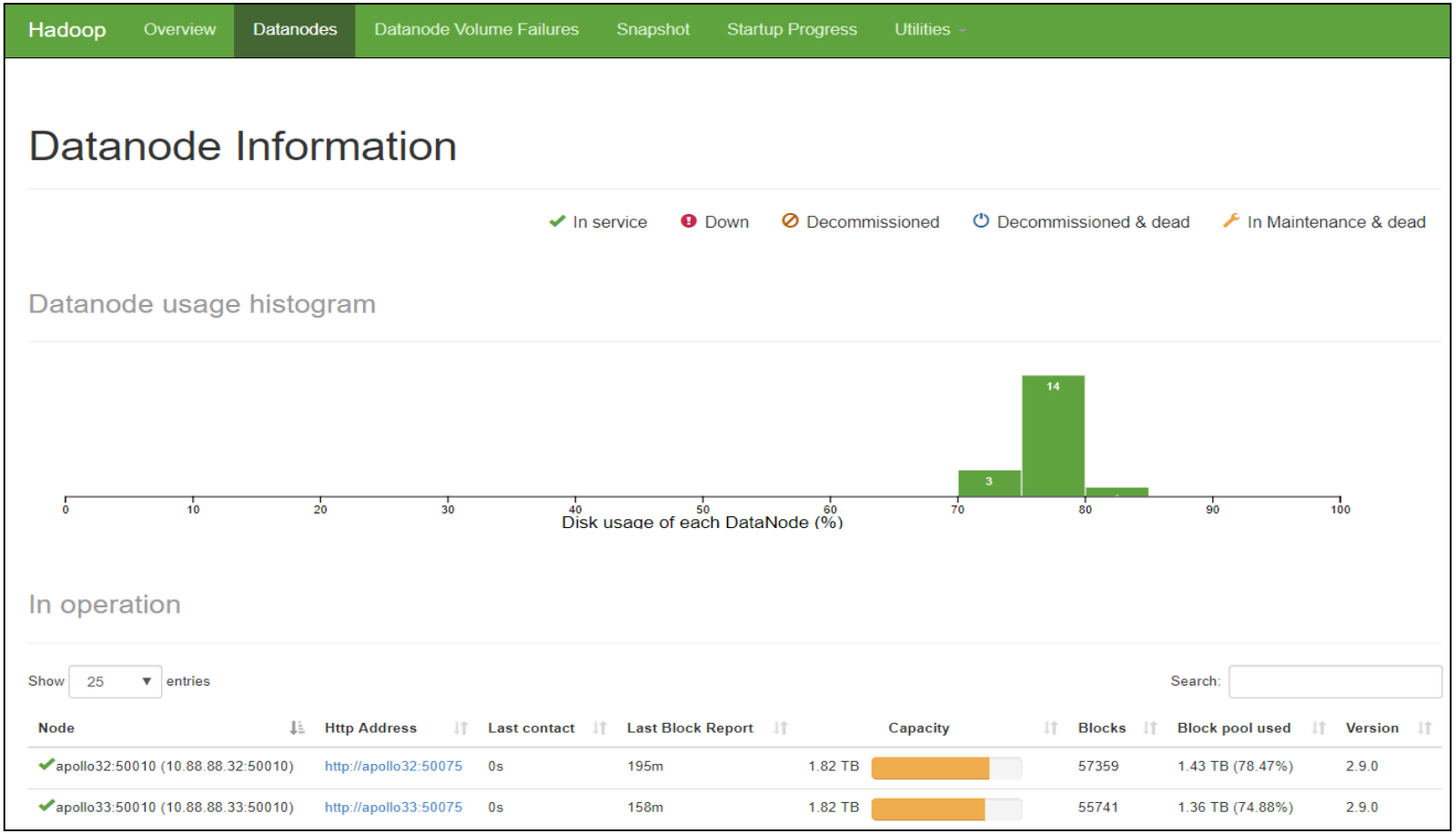
Cluster Hadoop Status

- Browser HDFS on web console (port:9870)



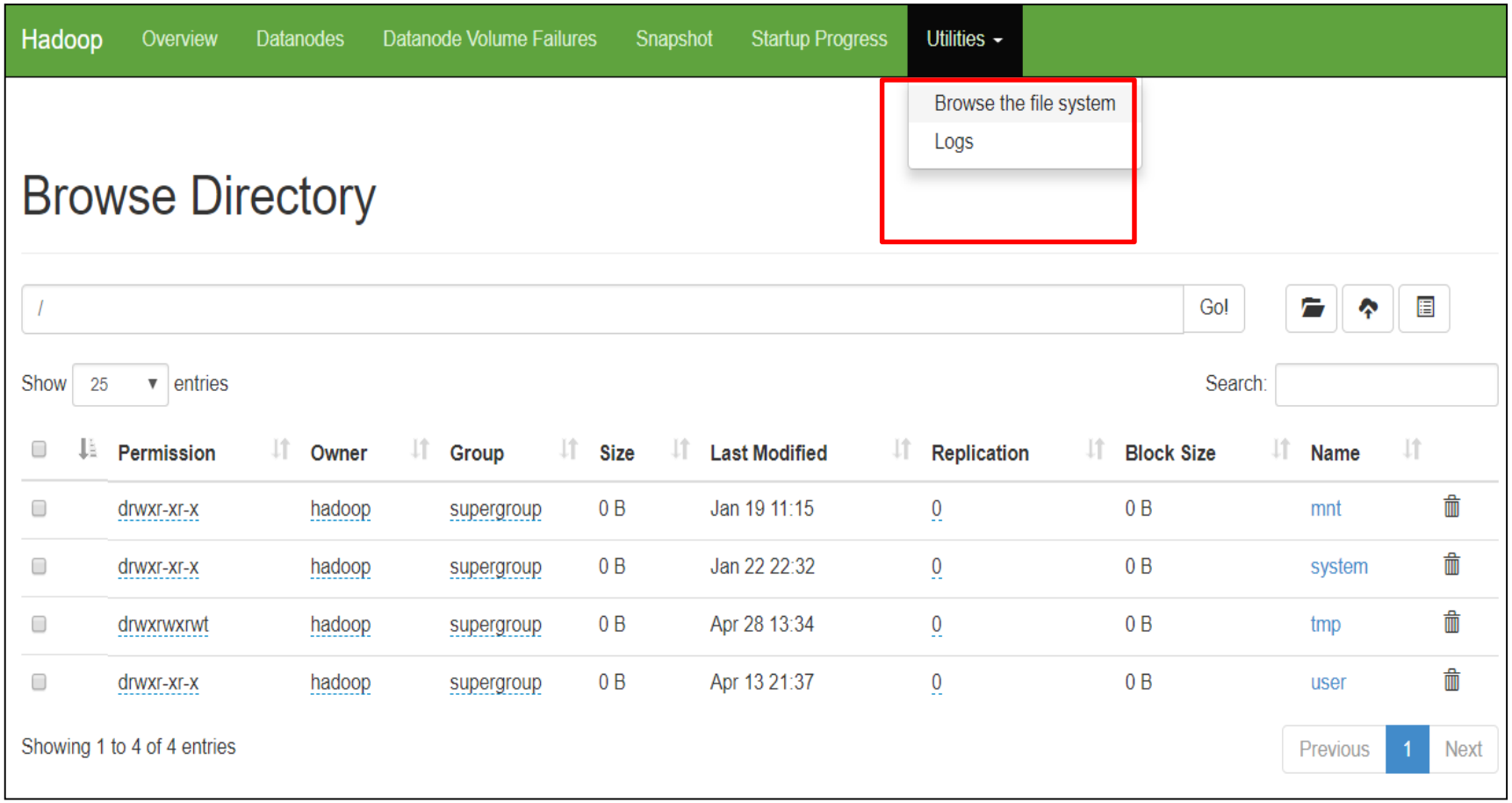
Cluster Hadoop Status

- Browser HDFS on web console (port:9870)



Cluster Hadoop Status

- Browser HDFS on web console (port:9870)



Job Tracker

- Browser all the job execution status (port:8088)

All Applications

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Mem
3	0	0	3	0	0 B	24 GB	0 B

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy N
2	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:32, vCores:1>	<memory:12288, vCores:8>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers
application_1575713584863_0003	pp19s50	wordcount	MAPREDUCE	default	0	Sat Dec 7 18:30:33 +0800 2019	Sat Dec 7 18:31:37 +0800 2019	FINISHED	SUCCEEDED	N/A
application_1575713584863_0002	hadoop	Spark shell	SPARK	default	0	Sat Dec 7 18:21:31 +0800 2019	Sat Dec 7 18:21:54 +0800 2019	FINISHED	SUCCEEDED	N/A
application_1575713584863_0001	hadoop	HIVE-3fbbadbc-46c7-425a-	TEZ	default	0	Sat Dec 7 18:21:07	Sat Dec 7 18:26:18 +0800	FINISHED	SUCCEEDED	N/A

Job History Server

- Browser job history (port:19888)

Job History Server

- Check the log file in cluster mode
  - Access Job History Server

Cheat Sheet Hive for SQL Users

- Complete reference: http://tw.gitbook.net/hive/index.html
- http://hortonworks.com/wp-content/uploads/2016/05/Hortonworks.CheatSheet.SQLtoHive.pdf

Function	MySQL	HiveQL
Retrieving information	`SELECT from_columns FROM table WHERE conditions;`	`SELECT from_columns FROM table WHERE conditions;`
All values	`SELECT * FROM table;`	`SELECT * FROM table;`
Some values	`SELECT * FROM table WHERE rec_name = "value";`	`SELECT * FROM table WHERE rec_name = "value";`
Multiple criteria	`SELECT * FROM table WHERE rec1="value1" AND rec2="value2";`	`SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";`
Selecting specific columns	`SELECT column_name FROM table;`	`SELECT column_name FROM table;`
Retrieving unique output records	`SELECT DISTINCT column_name FROM table;`	`SELECT DISTINCT column_name FROM table;`
Sorting	`SELECT col1, col2 FROM table ORDER BY col2;`	`SELECT col1, col2 FROM table ORDER BY col2;`
Sorting backward	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`
Counting rows	`SELECT COUNT(*) FROM table;`	`SELECT COUNT(*) FROM table;`
Grouping with counting	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`
Maximum value	`SELECT MAX(col_name) AS label FROM table;`	`SELECT MAX(col_name) AS label FROM table;`
Selecting from multiple tables (Join same table using alias w/"AS")	`SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;`	`SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);`

Hands-on Lab: Hive

- Run in Hive SHELL

```
$ hive
```

- Create a database named by your studentID

```
hive> CREATE DATABASE test_[username];
```

- Create a table named “exam” under your DB with the following schema: id(int), name(string), score(int)

```
hive> CREATE TABLE test_[username].exam (id int, name string, score int)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘\t’;
```

- Load data to the table from filepath ‘/home/hive/sample.txt’

```
hive> LOAD DATA LOCAL INPATH '/home/[username]/lab/hive/sample.txt'
OVERWRITE INTO TABLE test_[username].exam;
```

- Show all the content in table

```
hive> SELECT * from test_[username].exam;
```

Hands-on Lab: Hive

- Compute the average score from the table

```
hive> SELECT avg(score) from test_[username].exam;
```

```
Query ID = jchou_20191206171640_e0a851e3-8854-4197-89b2-c723f3eabfce
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1575649265022_0004)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 5.05 s
--------------------------------------------------------------------------------
OK
81.66666666666667
Time taken: 11.068 seconds, Fetched: 1 row(s)
```

Hands-on Lab: Spark

- spark-shell

```
[hadoop@ip-172-31-90-14 WordCount]$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
19/12/06 15:48:20 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling
 SPARK_HOME.
Spark context Web UI available at http://ip-172-31-90-14.ec2.internal:4040
Spark context available as 'sc' (master = yarn, app id = application_1575647061394_0002).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.4
      /_/

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_222)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

- Show your output result
- Spark can also be written in Python and run on PySpark
- Reference: https://spark.apache.org/examples.html

Hands-on Lab: Spark

- Type WordCount scala code line-by-line in Spark Shell

```
val textFile = sc.textFile("/user/hadoop/[username]/input/hello.dat")
val counts = textFile.flatMap(line => line.split(" ")) .map(word =>
(word, 1)) .reduceByKey(_ + _)
counts.saveAsTextFile("/user/spark/[username]")
```

- Show your output result

$ hadoop fs -cat /user/spark/[username]/part-00001

```
[hadoop@ip-172-31-90-14 WordCount]$ hadoop fs -ls /user/spark/123456/
Found 3 items
-rw-r--r--   1 hadoop spark          0 2019-12-06 16:09 /user/spark/123456/_SUCCESS
-rw-r--r--   1 hadoop spark         20 2019-12-06 16:09 /user/spark/123456/part-00000
-rw-r--r--   1 hadoop spark          9 2019-12-06 16:09 /user/spark/123456/part-00001
[hadoop@ip-172-31-90-14 WordCount]$ hadoop fs -cat /user/spark/123456/part-00000
(hello,2)
(world,1)
[hadoop@ip-172-31-90-14 WordCount]$ hadoop fs -cat /user/spark/123456/part-00001
(back,1)
```

Spark Examples

- WordCount

HDFS location

```
val textFile = sc.textFile("<input_directory_path>")
val counts = textFile.flatMap(line => line.split(" ")) .map(word
=> (word, 1)) .reduceByKey(_ + _)
counts.saveAsTextFile("<input_directory_path>")
```

- Pi Estimation

```
val count = sc.parallelize(1 to NUM_SAMPLES).filter { _ =>
      val x = math.random
      val y = math.random x*x + y*y < 1 }.count()
println(s"Pi is roughly ${4.0 * count / NUM_SAMPLES}")
```

Spark Examples

- Prediction with Logistic Regression

```
// Every record of this DataFrame contains the label and
// features represented by a vector.
val df = sqlContext.createDataFrame(data).toDF("label", "features")
// Set parameters for the algorithm.
// Here, we limit the number of iterations to 10.
val lr = new LogisticRegression().setMaxIter(10)
// Fit the model to the data.
val model = lr.fit(df)
// Inspect the model: get the feature weights.
val weights = model.weights
// Given a dataset, predict each point's label, and show the results.
model.transform(df).show()
```

Reference

- Source code for the lab
  - https://drive.google.com/file/d/1UltWVCYoGvDkVJe4-pXop0evJO9naX3Y/view?usp=sharing
- Write Hadoop in Python with Hadoop Streaming
  - https://hadoop.apache.org/docs/r1.2.1/streaming.html
  - https://www.tutorialspoint.com/hadoop/hadoop_streaming.htm
- Practice Hello World program in Hadoop using Hortonworks Sandbox
  - https://blogs.sap.com/2019/06/24/hello-world-program-in-hadoop-using-hortonworks-sandbox/