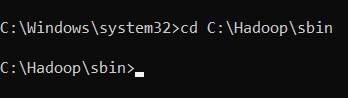
**Implement a MapReduce program to process a weather dataset**  **Steps:**

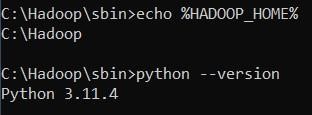
1. Open command prompt and run as administrator

Go to hadoop sbin directory

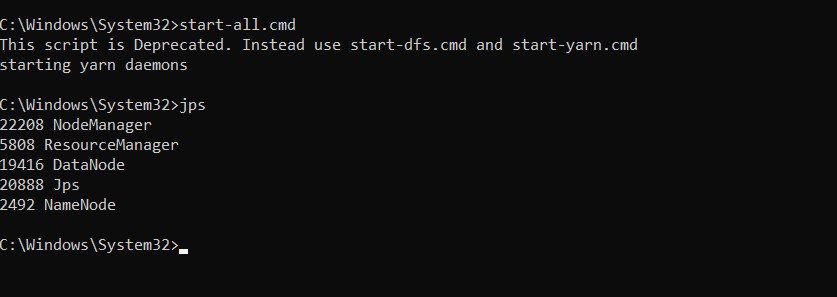


Note:

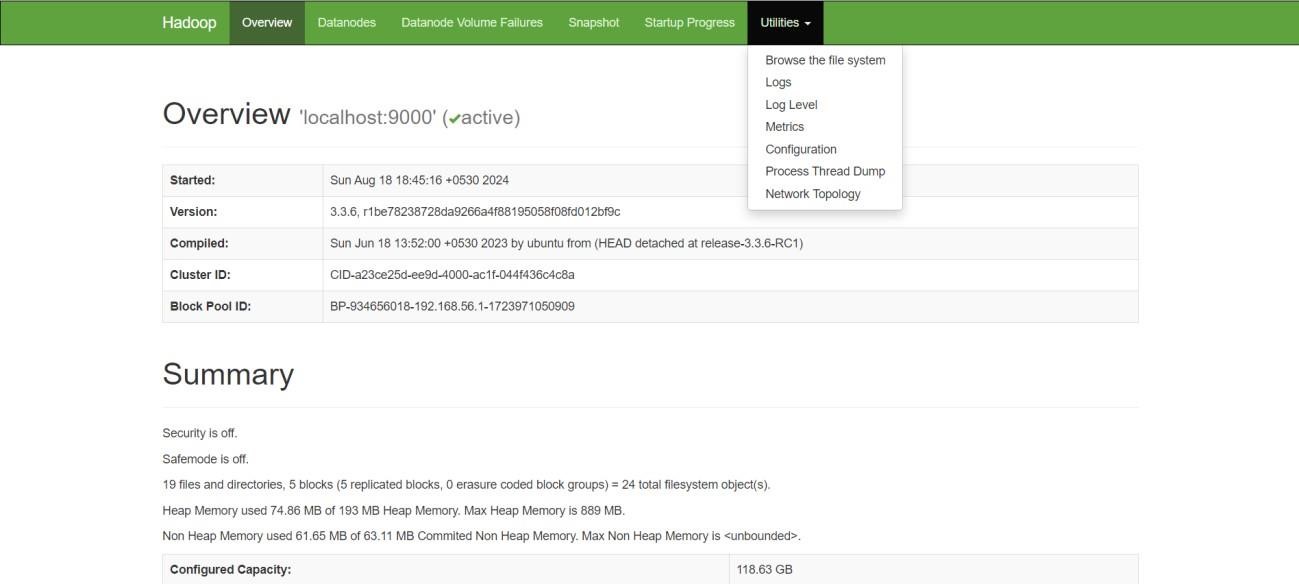
* 1. Check hadoop/data/datanode and hadoop/data/namenode and if both folders are empty, type “hdfs namenode -format”.
  2. Check python version with “python --version”.
  3. Check “C:\Python39\” is added in Environment variables > System variables > Path, if not add your python path.
  4. Check Environment variables > System variables > HADOOP\_HOME is set as “C:\Hadoop”.



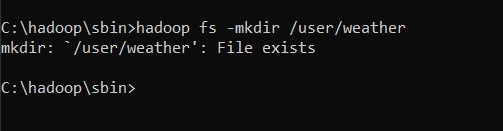
1. Start Hadoop Services start-dfs.cmd start-yarn.cmd



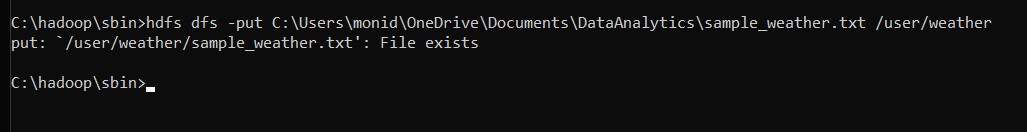
1. Open the browser and go to the URL “localhost:9870”



1. Create a Directory in HDFS hadoop fs -mkdir /user/weather



1. Copy the Input File to HDFS hdfs dfs -put C:\Users\monid\OneDrive\Documents\DataAnalytics\sample\_weather.txt /user/weather



**Note: mapper.py:**

#! /usr/bin/env python import sys

def map1(): for line in sys.stdin:

tokens = line.strip().split()

if len(tokens) < 13: continue

station = tokens[0]

if "STN" in station: continue

date\_hour = tokens[2] temp = tokens[3] dew = tokens[4] wind = tokens[12]

if temp == "9999.9" or dew == "9999.9" or wind == "999.9":

continue hour = int(date\_hour.split("\_")[-1]) date = date\_hour[:date\_hour.rfind("\_")-2] if 4 < hour <= 10: section = "section1" elif 10 < hour <= 16: section =

"section2" elif 16 < hour <= 22: section = "section3"

else:

section = "section4"

key\_out = f"{station}\_{date}\_{section}"

value\_out = f"{temp} {dew} {wind}" print(f"{key\_out}\t{value\_out}")

if \_\_name\_\_ == "\_\_main\_\_":

map1()

**reducer.py:** #!

/usr/bin/env python import sys

def reduce1(): current\_key = None sum\_temp, sum\_dew, sum\_wind = 0, 0, 0

count = 0

for line in sys.stdin:

key, value = line.strip().split("\t") temp, dew, wind = map(float, value.split()) if current\_key is None:

current\_key = key

if key == current\_key:

sum\_temp += temp sum\_dew += dew sum\_wind += wind

count += 1 else:

avg\_temp = sum\_temp / count avg\_dew = sum\_dew / count avg\_wind = sum\_wind / count print(f"{current\_key}\t{avg\_temp} {avg\_dew} {avg\_wind}")

current\_key = key sum\_temp, sum\_dew, sum\_wind = temp, dew, wind

count = 1

if current\_key is not None: avg\_temp = sum\_temp / count avg\_dew = sum\_dew / count avg\_wind = sum\_wind / count print(f"{current\_key}\t{avg\_temp} {avg\_dew} {avg\_wind}") if \_\_name\_\_ == "\_\_main\_\_":

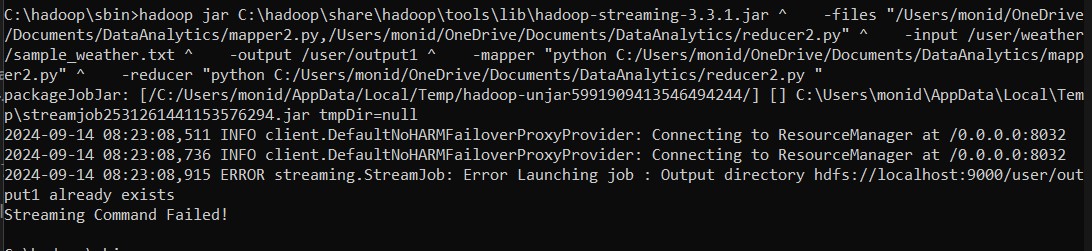
reduce1()

1. Run the Hadoop Streaming Job

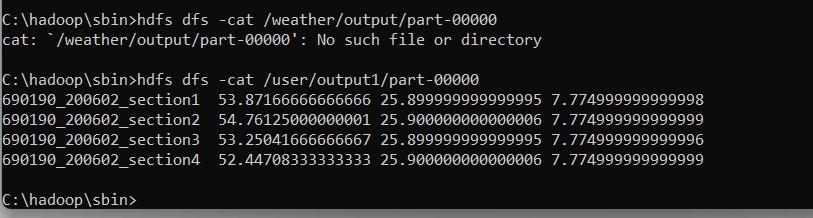
hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.3.1.jar ^ -files "/Users/monid/OneDrive/Documents/DataAnalytics/mapper2.py,/Users/monid/One Drive/Documents/DataAnalytics/reducer2.py" ^ -input

/user/weather/sample\_weather.txt ^ -output /user/output1 ^ -mapper "python

C:/Users/monid/OneDrive/Documents/DataAnalytics/mapper2.py" ^ -reducer "python C:/Users/monid/OneDrive/Documents/DataAnalytics/reducer2.py "

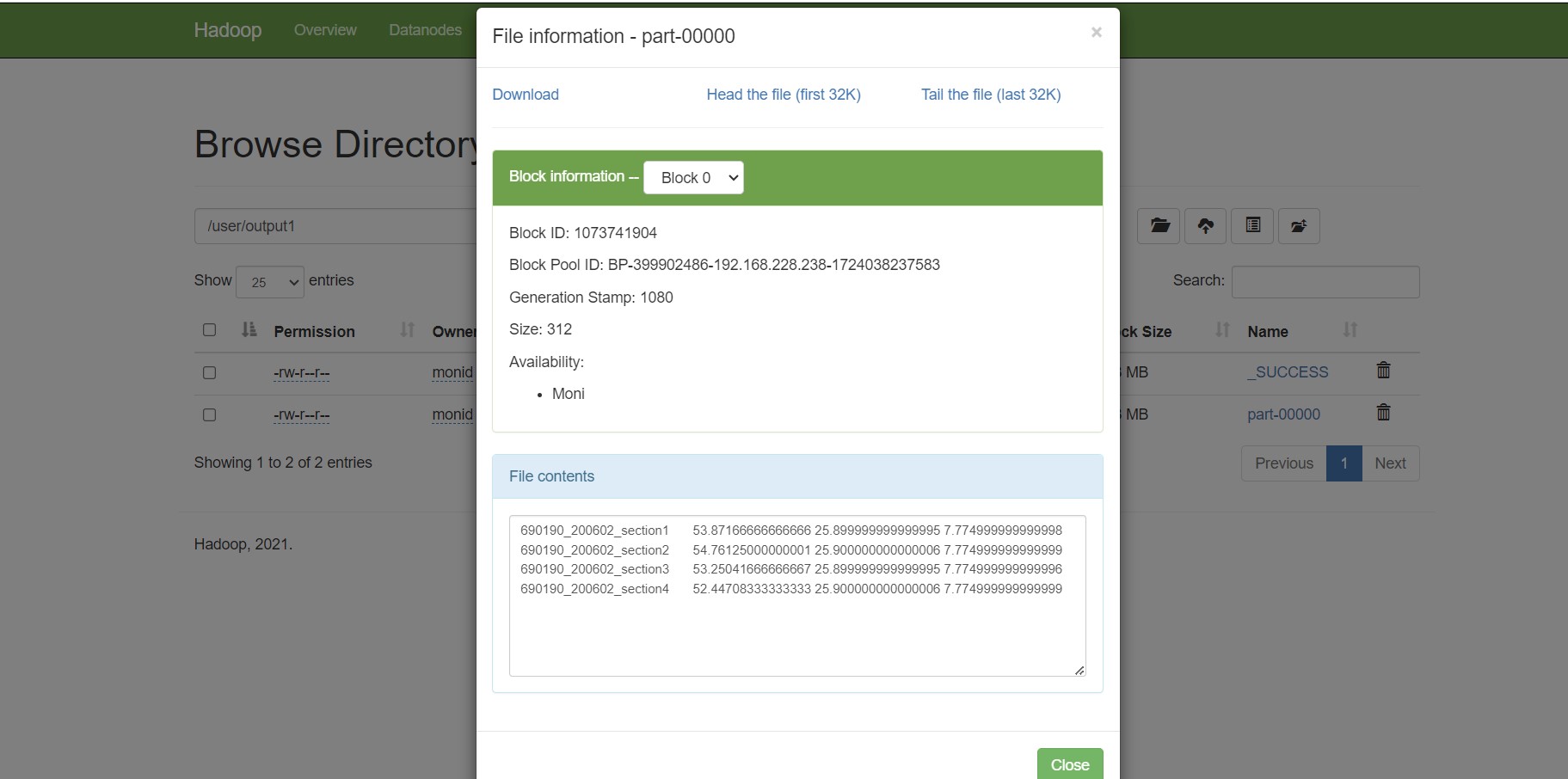


1. View the Output hdfs dfs -cat /user/output1/part-00000

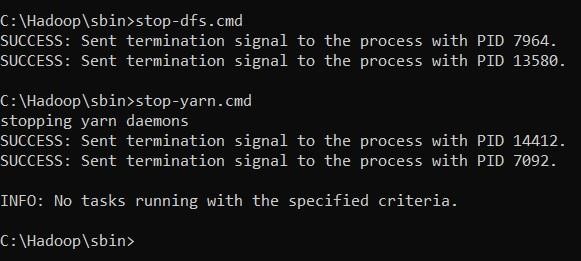


1. Once the map reduce operations are performed successfully, the output will be present in the specified directory.

“/user/output1/part-00000”



1. Stop Hadoop Services stop-dfs.cmd stop-yarn.cmd



**RESULT:**

Thus the implementation of the MapReduce python program a weather dataset in Hadoop is executed successfully.