

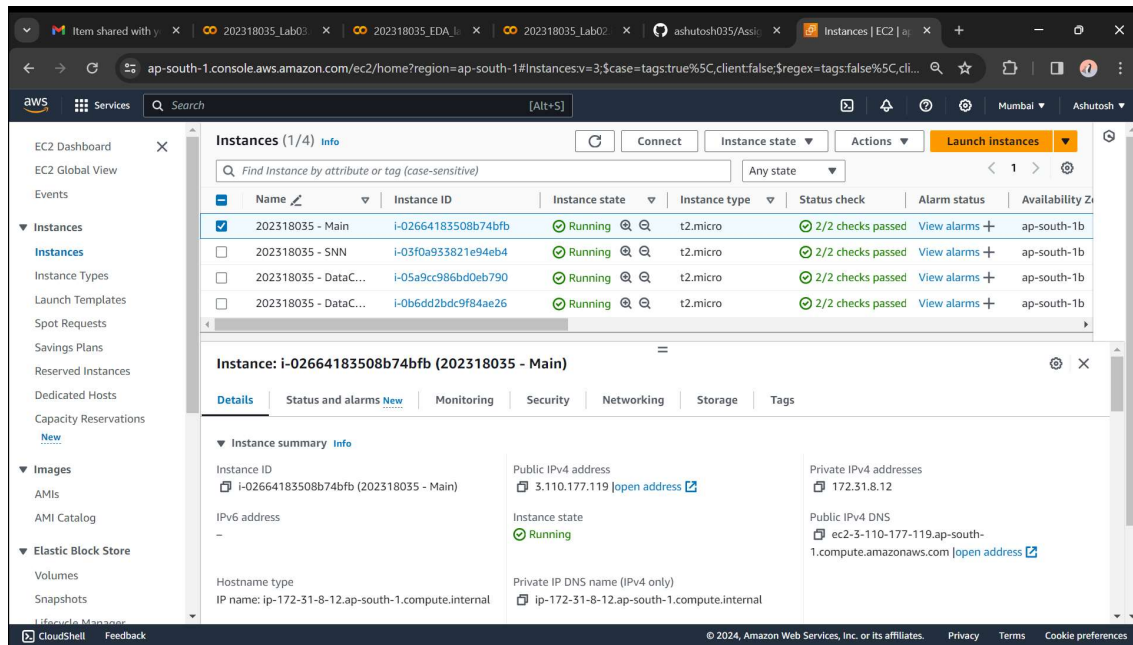
# IT 609 - Big Data Processing Lab 3

Ashutosh Anand

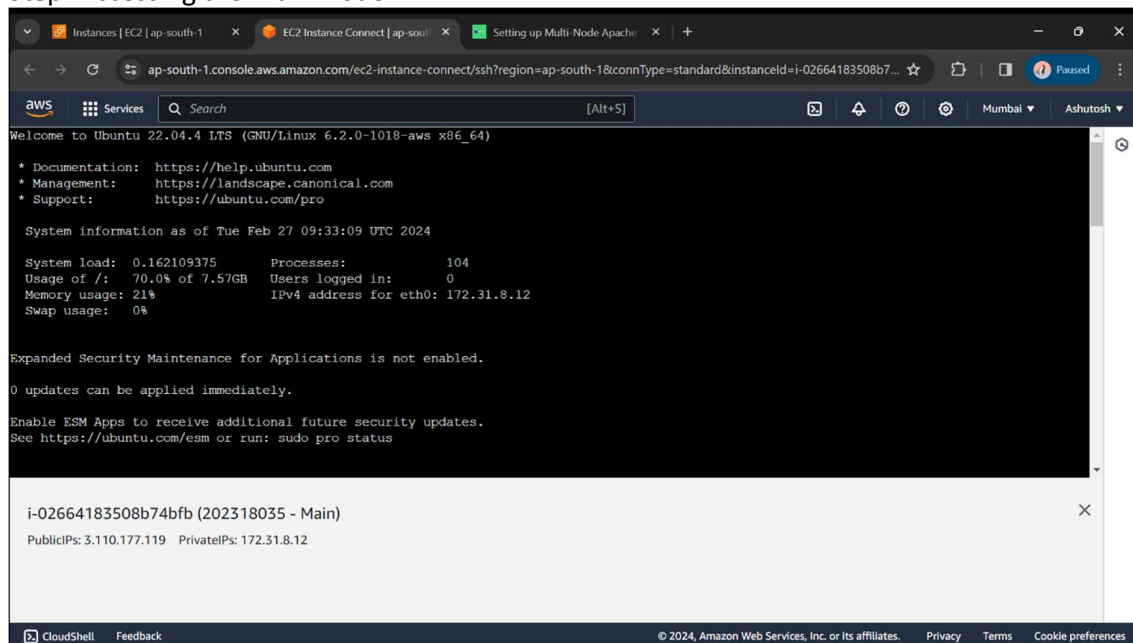
202318035

Implement the Map-Reduce algorithm in single Node cluster and Multi Node cluster using Hadoop-streaming Utility.

The Instances Created on the EC2, containing the Name, Secondary Name Node, Data Cluster1 & 2.

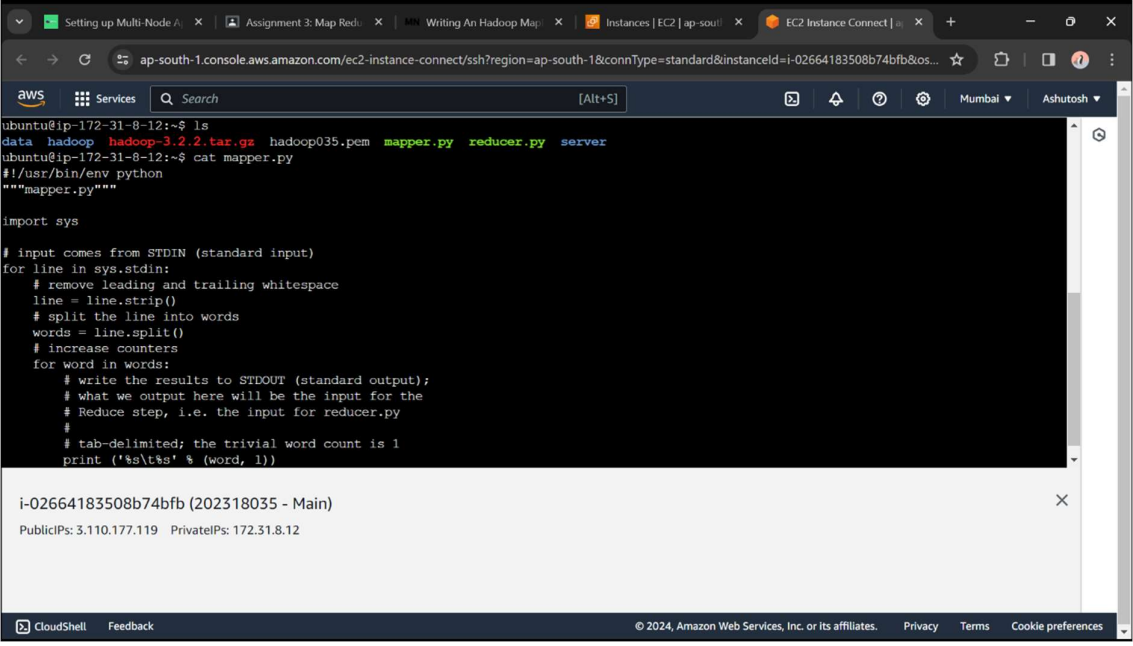


Step: Accessing the Main Node



Step: Map Step: mapper.py

It will read data, split it into words and output a list of lines mapping words to their (intermediate) counts to. The Map script will not compute an (intermediate) sum of a word's occurrences though. Instead, it will output `<word> 1` tuples immediately – even though a specific word might occur multiple times in the input.



The screenshot shows an AWS CloudShell terminal window. The terminal displays the following commands and output:

```
ubuntu@ip-172-31-8-12:~$ ls
data  hadoop  hadoop-3.2.2.tar.gz  hadoop035.pem  mapper.py  reducer.py  server
ubuntu@ip-172-31-8-12:~$ cat mapper.py
#!/usr/bin/env python
"""mapper.py"""

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print('%s\t%s' % (word, 1))
```

Below the terminal output, the instance information is displayed:

```
i-02664183508b74bfb (202318035 - Main)
PublicIPs: 3.110.177.119  PrivateIPs: 172.31.8.12
```

## Step: Reduce Step: reducer.py

It will read the results of `mapper.py` and sum the occurrences of each word to a final count, and then output its results.

The image displays two screenshots of an AWS CloudShell terminal window, showing the execution of a Hadoop reducer script. The terminal is connected to an EC2 instance via EC2 Instance Connect.

**Top Screenshot:** The terminal shows the command `cat reducer.py` being executed. The output displays the content of the `reducer.py` file, which is a Python script for a Hadoop reducer. The script processes input from STDIN, parses the input into words and counts, and writes the results to STDOUT. The script includes error handling for non-integer counts and ensures the last word is output.

```
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
            current_count = count
            current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s' % (current_word, current_count)

ubuntu@ip-172-31-8-12:~$
```

**Bottom Screenshot:** The terminal shows the same script being executed, but now the output is visible. The script has processed the input and written the results to STDOUT. The output shows the word and count for each line of input.

```
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
            current_count = count
            current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s' % (current_word, current_count)

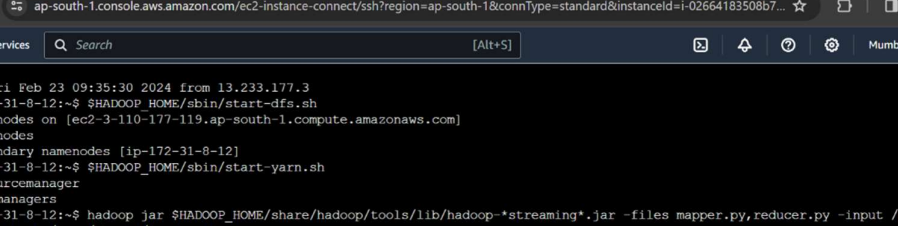
ubuntu@ip-172-31-8-12:~$
```

Step: Checking the mapper.py and reducer.py file with a short txt file to check if the code is working properly or not.

The image displays two screenshots of an AWS CloudShell terminal session. The top screenshot shows the initial setup of a Hadoop environment. The user runs `hdfs-site.xml` and `mapred-env.cmd`, then sets the `ssl-client.xml.example` file. They create a `test1.txt` file and run a Python script `mapper.py` and `reducer.py`. The output shows a list of words and their counts, such as `even 1`, `though 1`, `specific 1`, `word 1`, `might 1`, `occur 1`, `multiple 1`, `times 1`, `in 1`, `the 1`, `input. 1`, `in 1`, `our 1`, `case 1`, `we 1`, `let 1`, `the 1`, `subsequent 1`, `Reduce 1`, and `step 1`. The bottom screenshot shows the execution of the Hadoop MapReduce job. The user runs `hadoop jar hdfs-streaming.jar` and the output shows a list of words and their counts, such as `the 1`, `final 1`, `sum 1`, `count. 1`, `of 1`, `course, 1`, `you 1`, `can 1`, `change 1`, `this 1`, `behavior 1`, `in 1`, `your 1`, `own 1`, `scripts 1`, `as 1`, `you 1`, `please, 1`, `but 1`, `we 1`, `will 1`, `keep 1`, `it 1`, `like 1`, `that 1`, `in 1`, `this 1`, `tutorial 1`, `because 1`, `of 1`, `didactic 1`, `reasons. 1`, and `:-) 1`.

Step: Here we are starting the `dfs.sh` and `yarn.sh` and leaving the safe mode to run the MapReduce Job.

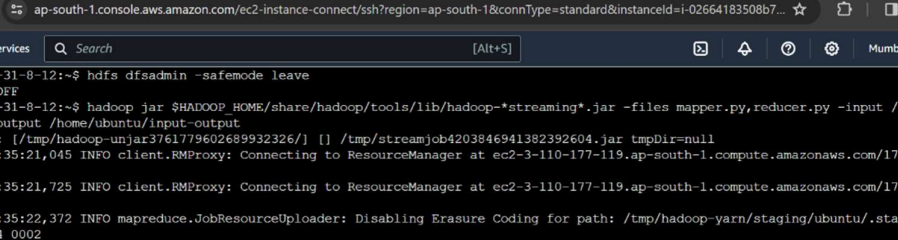
We leverage the Hadoop Streaming API for helping us passing data between our Map and Reduce code.



```
Last login: Fri Feb 23 09:35:30 2024 from 13.233.177.3
ubuntu@ip-172-31-8-12:~$ $HADOOP_HOME/sbin/start-dfs.sh
Starting namenodes on [ec2-3-110-177-119.ap-south-1.compute.amazonaws.com]
Starting datanodes
Starting secondary namenodes [ip-172-31-8-12]
ubuntu@ip-172-31-8-12:~$ $HADOOP_HOME/sbin/start-yarn.sh
Starting resourcemanager
Starting nodemanagers
ubuntu@ip-172-31-8-12:~$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar -files mapper.py, reducer.py -input /home/ubun
tu/input/* -output /home/ubuntu/input-output
packageJobJar: [/tmp/hadoop-unjar5300392927469417489/] [] /tmp/streamjob1837634361821133894.jar tmpDir=null
2024-02-27 09:34:43,760 INFO client.RMProxy: Connecting to ResourceManager at ec2-3-110-177-119.ap-south-1.compute.amazonaws.com/172.31.8.1
2:8032
2024-02-27 09:34:44,286 INFO client.RMProxy: Connecting to ResourceManager at ec2-3-110-177-119.ap-south-1.compute.amazonaws.com/172.31.8.1
2:8032
2024-02-27 09:34:45,012 INFO mapreduce.JobSubmitter: Cleaning up the staging area /tmp/hadoop-yarn/staging/ubuntu/.staging/job_170902644864
4_0001
2024-02-27 09:34:45,031 ERROR streaming.StreamJob: Error Launching job : Cannot delete /tmp/hadoop-yarn/staging/ubuntu/.staging/job_1709026
448644_0001. Name node is in safe mode.
The reported blocks 0 needs additional 6 blocks to reach the threshold 0.990 of total blocks 7.
The minimum number of live datanodes is not required. Safe mode will be turned off automatically once the thresholds have been reached. Nam
```

The job will read all the files in the HDFS directory `/home/ubuntu/input/*` , process it, and store the results in the HDFS directory `/home/ubuntu/input-output` . In general Hadoop will create one output file per reducer.

As per the Screenshot the MapReduce Job is running.



```
ubuntu@ip-172-31-8-12:~$ hdfs dfsadmin -safemode leave
Safe mode is OFF
ubuntu@ip-172-31-8-12:~$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-*streaming*.jar -files mapper.py, reducer.py -input /home/ub
ntu/input/* -output /home/ubuntu/output
packageJobJar: [/tmp/hadoop-unjar3761779602689932326/] [] /tmp/streamjob4203846941382392604.jar tmpDir=null
2024-02-27 09:35:21,045 INFO client.RMProxy: Connecting to ResourceManager at ec2-3-110-177-119.ap-south-1.compute.amazonaws.com/172.31.8.1
2:8032
2024-02-27 09:35:21,725 INFO client.RMProxy: Connecting to ResourceManager at ec2-3-110-177-119.ap-south-1.compute.amazonaws.com/172.31.8.1
2:8032
2024-02-27 09:35:22,372 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/ubuntu/.staging/job
_1709026448644_0002
2024-02-27 09:35:23,721 INFO mapred.FileInputFormat: Total input files to process : 1
2024-02-27 09:35:23,855 INFO mapreduce.JobSubmitter: number of splits:2
2024-02-27 09:35:24,723 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1709026448644_0002
2024-02-27 09:35:24,726 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-02-27 09:35:25,123 INFO conf.Configuration: resource-types.xml not found
2024-02-27 09:35:25,123 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-02-27 09:35:25,809 INFO impl.YarnClientImpl: Submitted application application_1709026448644_0002
2024-02-27 09:35:26,112 INFO mapreduce.Job: The url to track the job: http://ec2-3-110-177-119.ap-south-1.compute.amazonaws.com:8088/proxy/
application_1709026448644_0002/
2024-02-27 09:35:26,120 INFO mapreduce.Job: Running job: job_1709026448644_0002
```