
CSCI 5408

***Data Management, Warehousing, And
Analytics***

Assignment 3 - Problem 1

Perform research on NoSQL and data processing.

Prepared By

Bhavisha Oza (B00935827)

Problem-1:

Problem 1A: Reuter News Data Reading & Transformation and storing in MogoDb

1. Summary:

The task is to create a MongoDB database named "ReuterDb" using a Java program called "ReutRead.java" that processes two given news files (reut2-009.sgm and reut2-014.sgm). Each document in the database will represent a news article and must contain the extracted text between <REUTERS></REUTERS>, <TEXT></TEXT>, and <TITLE></TITLE> tags.

The program first reads the content of the two input files and merges them into a single string. It then uses regular expressions to extract data between <REUTERS> tags and further extracts text between <TITLE> and <TEXT> tags for each Reuters article. For each article, if both title and text content are present, it creates a MongoDB document and inserts it into the "ReuterDb" collection.

To visualize the program's logic, a flowchart and algorithm is included in the PDF file. The flowchart illustrates the sequence of steps, decision points, and data processing, while the algorithm provides a high-level outline of the program's functioning.

Requirements & Tasks fulfilled:

1. Create an account of MongoDB Atlas [3, 7]

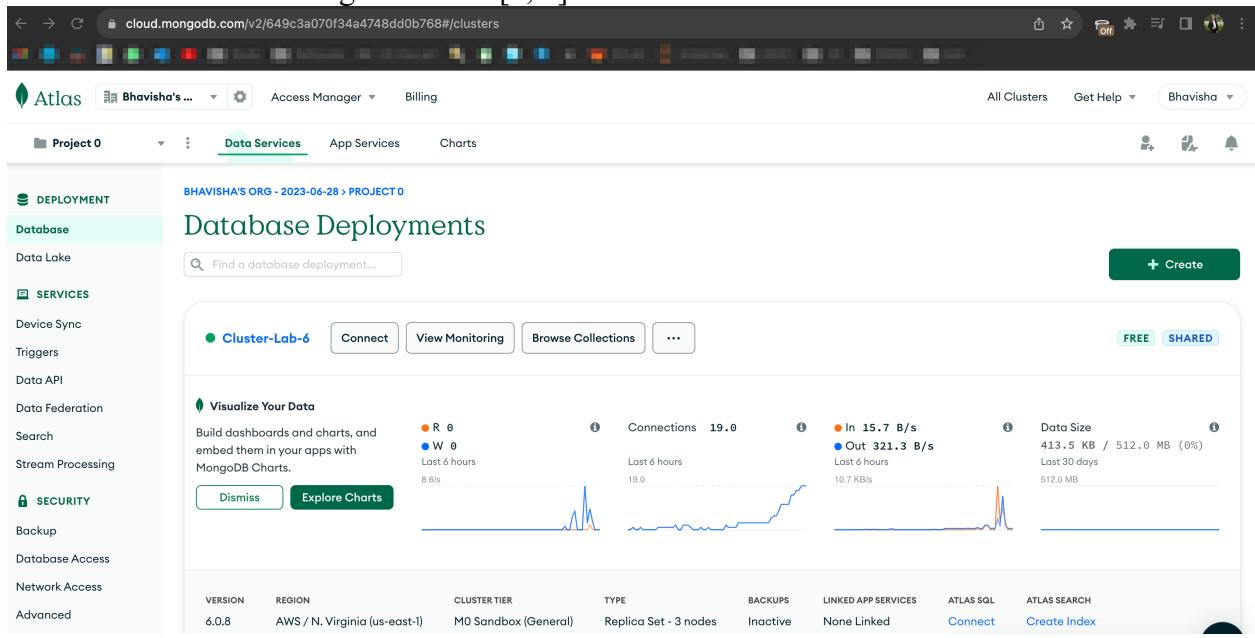


Figure 1: MongoDB atlas account creation.

2. Create a new cluster [7].

3. Cloud Provider & Region.

Enter the name which you want to give to your cluster [7].

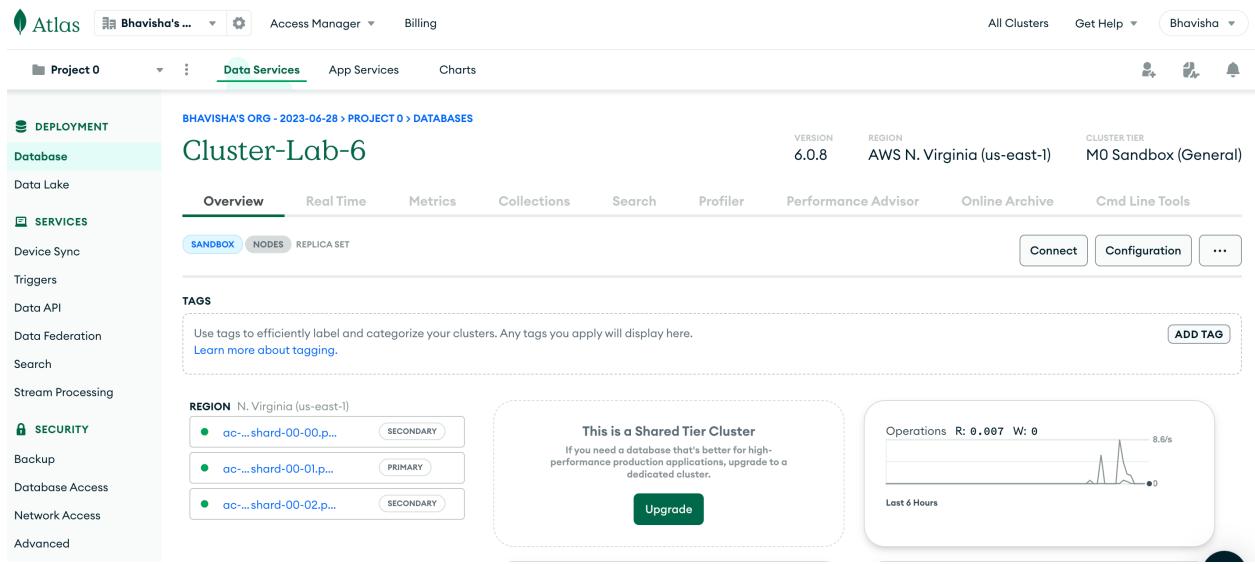


Figure 2: Cluster creation.

4. Create a java program, which creates MongoDB Database – ReuterDb from the two given news files (reut2-009.sgm, and reut2-014.sgm), where each Document contains a news article [1].

```

import com.mongodb.client.*;
import org.bson.Document;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

2 usages ▾ boza
public class ReutRead {

    1 usage ▾ boza
    public void processData() {
        String fileName1 = "/Users/bhavishaaze/IdeaProjects/DataMgmt/A3/src/reut2-009.sgm";
        String fileName2 = "/Users/bhavishaaze/IdeaProjects/DataMgmt/A3/src/reut2-014.sgm";

        // Step 1: Create MongoDB connection and database
        String uri = "mongodb+srv://erbhavisha:admin@cluster-lab-6.pqo1uvf.mongodb.net/";
        try (MongoClient mongoClient = MongoClients.create(uri)) {
            MongoDatabase database = mongoClient.getDatabase("A3DB");
            MongoCollection<Document> collection = database.getCollection("News");

            // Step 2: Read the content
            String part1 = readFromFile(fileName1);
            String part2 = readFromFile(fileName2);
            String mergedPart = part1 + "\n" + part2;

            // Step 3: Extract the text between <REUTERS> tags
            Pattern reutersPattern = Pattern.compile("(?s)<REUTERS[^>]*>.*?</REUTERS>", Pattern.DOTALL);
            Matcher reutersMatcher = reutersPattern.matcher(mergedPart);
    }
}

```

Figure 3: Java program for processes data from .sgm files and inserts article titles and texts into a MongoDB collection.

Program Output:

- IntelliJ

```

public class Main {
    public static void main(String[] args) {
        // Problem 1A call - ReutRead
        ReutRead reutRead = new ReutRead();
        reutRead.processData();
    }
}

INFO: opened connection [connectionId{localValue:3, serverValue:0} to ac-var4ffk-shard-00-00.pqo1uvf.mongodb.net:27017]
Jul. 30, 2023 3:09:12 A.M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:63934}] to ac-var4ffk-shard-00-02.pqo1uvf.mongodb.net:27017
Jul. 30, 2023 3:09:12 A.M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:67969}] to ac-var4ffk-shard-00-01.pqo1uvf.mongodb.net:27017
Jul. 30, 2023 3:09:13 A.M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=ac-var4ffk-shard-00-02.pqo1uvf.mongodb.net:27017, type=REPLICA_SET}
Jul. 30, 2023 3:09:13 A.M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=ac-var4ffk-shard-00-00.pqo1uvf.mongodb.net:27017, type=REPLICA_SET}
Jul. 30, 2023 3:09:13 A.M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=ac-var4ffk-shard-00-01.pqo1uvf.mongodb.net:27017, type=REPLICA_SET}
Jul. 30, 2023 3:09:13 A.M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Setting max election id to 7fffffff0000000000000000 from replica set primary ac-var4ffk-shard-00-01.pqo1uvf.mongodb.net:27017
Jul. 30, 2023 3:09:13 A.M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Setting max set version to 7 from replica set primary ac-var4ffk-shard-00-01.pqo1uvf.mongodb.net:27017
Jul. 30, 2023 3:09:13 A.M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Discovered replica set primary ac-var4ffk-shard-00-01.pqo1uvf.mongodb.net:27017
Jul. 30, 2023 3:09:13 A.M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:4, serverValue:67969}] to ac-var4ffk-shard-00-01.pqo1uvf.mongodb.net:27017

```

Figure 4: IntelliJ output

- MongoDB Compass

The screenshot shows the MongoDB Compass interface. The top bar indicates "MongoDB Compass - dwma/A3DB.News". The left sidebar lists databases (dwma, A3DB, Cluster-Lab-6, University, admin, local) and collections (News). The main area is titled "A3DB.News" and shows a list of 21 documents. A search bar at the top says "Type a query: { field: 'value' }". Below it are buttons for "ADD DATA" and "EXPORT DATA". The document list contains four entries, each with a preview of its content:

- Document 1:** `_id: ObjectId('64c5ff4f3bc5e01bf9b86b1c')`
`title: "<NORTH STAR URANIUM INC> CHANGES NAME"`
`text: "`
`<TITLE><NORTH STAR URANIUM INC> CHANGES NAME</TITLE>`
`<DATELINE>..."`
- Document 2:** `_id: ObjectId('64c5ff4f3bc5e01bf9b86b1d')`
`title: "GREAT LAKES CHEMICAL <GLK> TO BUILD NEW PLANT"`
`text: "`
`<TITLE>GREAT LAKES CHEMICAL <GLK> TO BUILD NEW PLANT</TITLE>`
`<...>"`
- Document 3:** `_id: ObjectId('64c5ff4f3bc5e01bf9b86b1e')`
`title: "SWEDISH UNEMPLOYMENT STEADY IN MARCH"`
`text: "`
`<TITLE>SWEDISH UNEMPLOYMENT STEADY IN MARCH</TITLE>`
`<DATELINE>..."`
- Document 4:** `_id: ObjectId('64c5ff4f3bc5e01bf9b86b1f')`
`title: "COMPAQ <CPO> EXPECTS HIGHER FIRST QUARTER NET"`

Figure 5: MongoDB Compass output

5. An algorithm of Reuters Data cleaning/transformation program

- Start the program execution.
- Define the 'ReutRead' class with the following methods:
 - `processData()`: This method will handle the data cleaning and transformation process.
- Inside the `processData()` method:
 - Define two variables `fileName1` and `fileName2` to store the paths of the input .sgm files.
 - Create a MongoDB connection using the provided connection URI.
 - Get a reference to the database named "A3DB" and the collection named "News" in MongoDB.
 - Read the content of both input .sgm files (`fileName1` and `fileName2`) using the `readFromFile()` method and store the content in strings `part1` and `part2`.
 - Merge `part1` and `part2` to form `mergedPart`, which will contain the entire data from both files as a single string.
- Use regular expressions to extract data between the `<REUTERS>` tags in `mergedPart`.
 - Define a regex pattern `reutersPattern` to match content between `<REUTERS>` and `</REUTERS>` tags using `Pattern.compile()`.
 - Create a matcher `reutersMatcher` using `reutersPattern.matcher(mergedPart)`.
- Use regular expressions to extract text between `<TITLE>` and `<TEXT>` tags for each Reuters article in `reutersMatcher`.
 - Define a regex pattern `titlePattern` to match content between `<TITLE>` and `</TITLE>` tags using `Pattern.compile()`.
 - Define a regex pattern `textPattern` to match content between `<TEXT>` and `</TEXT>` tags using `Pattern.compile()`.

- Iterate through `reutersMatcher` and for each Reuters article:
 - Extract the article's content into a `reutersArticle` string.
 - Create a matcher `titleMatcher` using `titlePattern.matcher(reutersArticle)`.
 - Create a matcher `textMatcher` using `textPattern.matcher(reutersArticle)`.
 - If both `titleMatcher` and `textMatcher` find a match (i.e., if there are both title and text content present), proceed to the next steps. Otherwise, skip the current article.
 - Extract the `title` and `text` content using `titleMatcher.group(1)` and `textMatcher.group(1)` respectively.
 - Create a MongoDB `Document` and add the extracted `title` and `text` as key-value pairs.
 - Insert the created `Document` into the MongoDB collection "News" using `collection.insertOne(document)`.
 - Repeat steps 6 to 10 until there are no more Reuters articles in `reutersMatcher`.
 - Close the MongoDB connection.
 - End the program execution.
6. The flowchart of Reuters Data cleaning/transformation program created using darw.io tool [8].

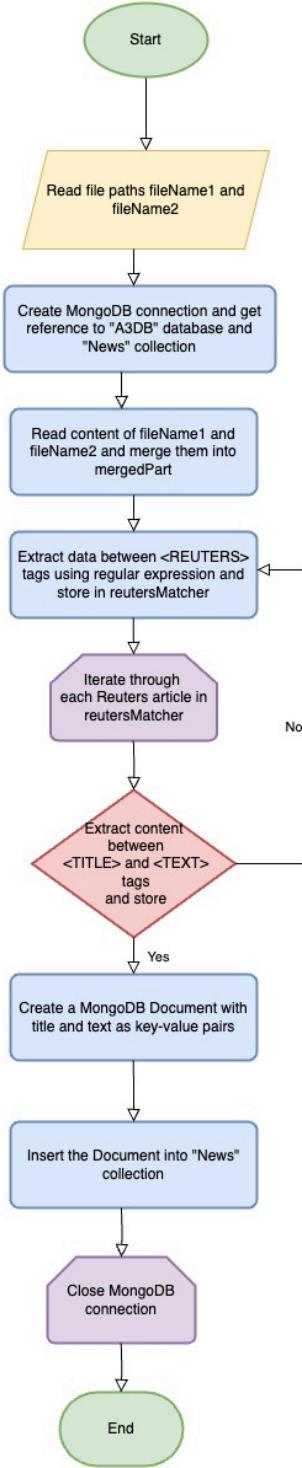


Figure 6: Flowchart of Reuters Data cleaning/transformation program

Problem 1B: Reuter News Data Processing using Spark.

1. Summary:

In this project, the main objective is to configure and initialize an Apache Spark cluster using the GCP (Google Cloud Platform) cloud account. The process involves following the provided tutorials in the lab session to set up the cluster successfully. The written explanation of the steps taken to complete the task are included in the description.

Once the cluster is set up, a MapReduce program is implemented using Java (WordCounter.java Engine). The program's purpose is to count the unique words and their frequencies in the file "reut2-009.sgm" and represent the words with the highest and lowest frequencies as an outcome of the MapReduce program.

TASK-1 Configure and initialize Apache Spark cluster using GCP cloud account [2].

Requirements & Tasks fulfilled to configure & initialize Apache Spark cluster using GCP:

1. Create an account on GCP, search for dataproc [2, 6]:

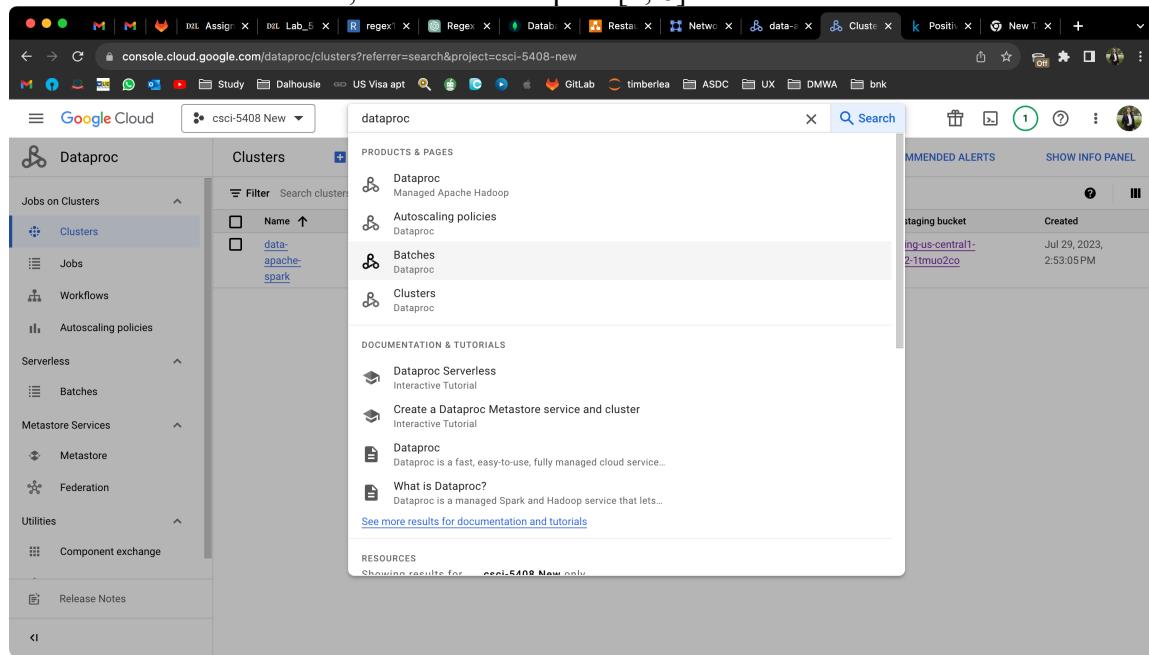


Figure 7: Searching dataproc.

2. Click on create cluster, select cluster on Compute Engine [6].

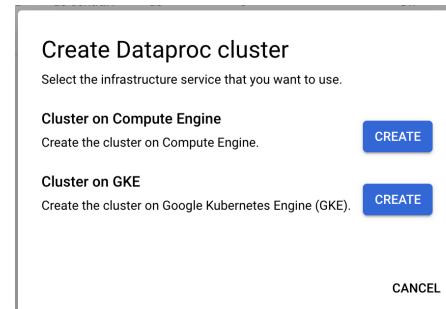


Figure 8: Creating cluster on Compute Engine

3. Create cluster by selecting following options:
Enter cluster name as data-apache-spark
In Location, select us-central1-c from zone dropdown
select cluster type as Single node [6].

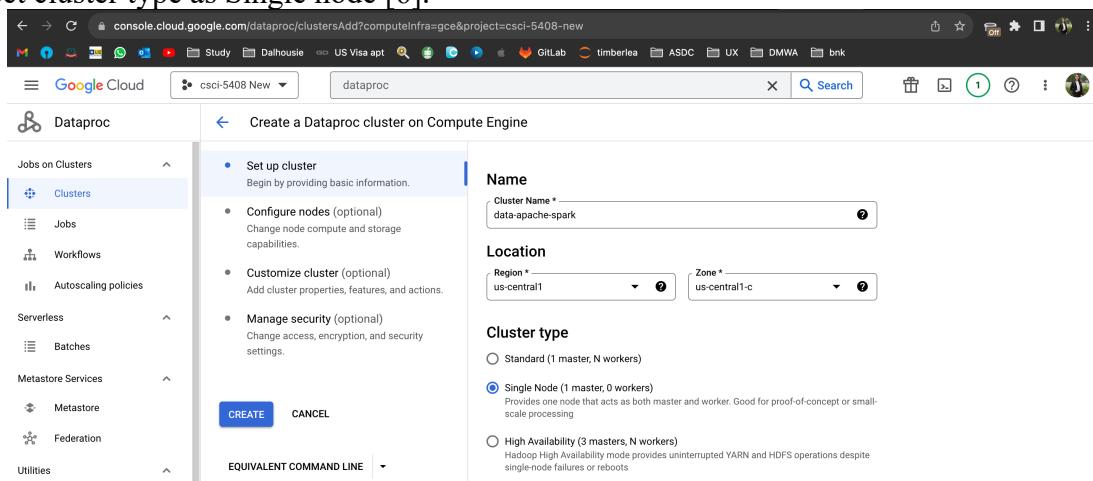


Figure 9: Setup selection for cluster

4. Start the cluster.

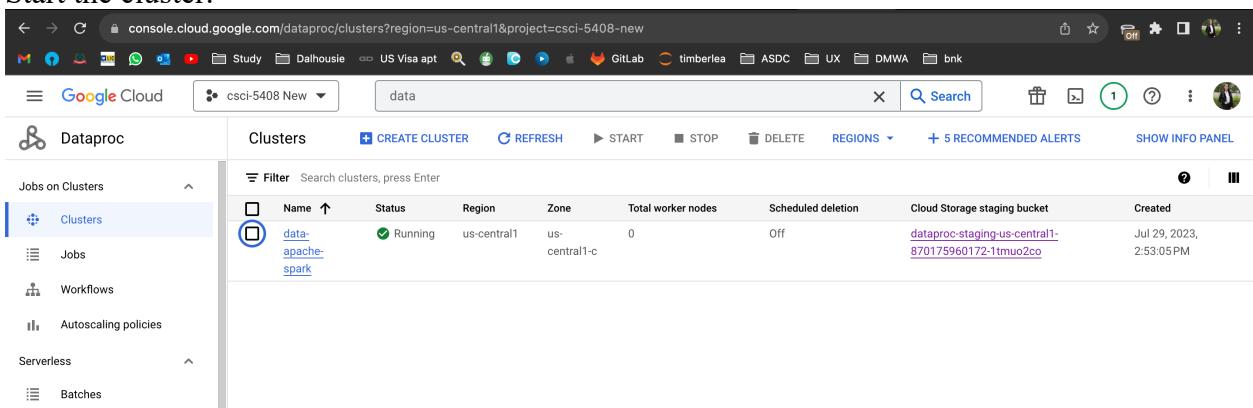


Figure 10: Apache spark cluster running.

5. Open the cluster and Go to VM instances and click on created instance.

The screenshot shows the Google Cloud Dataproc Cluster details page. On the left, there's a sidebar with sections like 'Jobs on Clusters' (Clusters, Jobs, Workflows, Autoscaling policies), 'Serverless' (Batches), and 'Metastore Services' (Metastore, Federation). The main area is titled 'Cluster details' with tabs for 'SUBMIT JOB', 'REFRESH', 'START', 'STOP', 'DELETE', and 'VIEW LOGS'. Below this, it says 'Consider using Auto Zone rather than selecting a zone manually. See https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/auto-zone'. It shows cluster details: Name (data-apache-spark), Cluster UUID (908165fc-cd73-4498-a304-67ee5916e1b3), Type (Dataproc Cluster), and Status (Running). The 'VM INSTANCES' tab is selected, showing a table with one row: 'data-apache-spark-m' (Role: Master). At the bottom, there's an 'EQUIVALENT REST' link.

Figure 11: Checking created instance.

6. From the details page, click on edit icon, go to the Firewalls section, and check the checkbox of Allow HTTP traffic and Allow HTTPS traffic and save [6].

The screenshot shows the Google Cloud Compute Engine VM instances details page for instance 'data-apache-spark-m'. The left sidebar includes sections for Virtual machines (VM instances, Instance templates, Sole-tenant nodes, Machine images, TPUs, Committed use discounts, Reservations, Migrate to Virtual Machine...), Storage (Disks, Marketplace), and Release Notes. The main page has tabs for 'DETAILS', 'OBSERVABILITY', 'OS INFO', and 'SCREENSHOT'. Under 'Networking', it shows Public DNS PTR Record (None), Total egress bandwidth tier (—), and NIC type (—). There's a link to 'VIEW IN NETWORK TOPOLOGY'. The 'Firewalls' section contains three rows: 'HTTP traffic' (On), 'HTTPS traffic' (On), and 'Allow Load Balancer Health checks' (Off). The 'Network tags' section lists 'http-server' and 'https-server'. The 'Network interfaces' section shows a table with one row: 'nic0' (Network: default, Subnetwork: default, Primary internal IP address: 10.128.0.2, Alias IP ranges: —, IP stack type: IPv4, External IP address: 35.224.21.92 (Ephemeral), Network: Premium).

Figure 12: Configuring the instance.

➔ **Figure 11** displays the created virtual machine (VM) instance `data-apache-spark` under the project `csci-5408 New` [6].

The screenshot shows the Google Cloud Network interface details page. On the left, there's a sidebar with options like VPC networks, IP addresses, Firewall, Routes, VPC network peering, Shared VPC, Serverless VPC access, and Packet mirroring. The main area displays 'Network interface details' for 'Selected network interface: nic0'. It includes tables for 'VM instance details' (listing data-apache-spark-m) and 'Firewall and routes details' (listing a VPC firewall rule named 'vpc-firewall-rules').

Figure 13: data-apache-spark VM hosted on GCP.

- Click on SSH from the VM instance menu to connect to the shell [6].

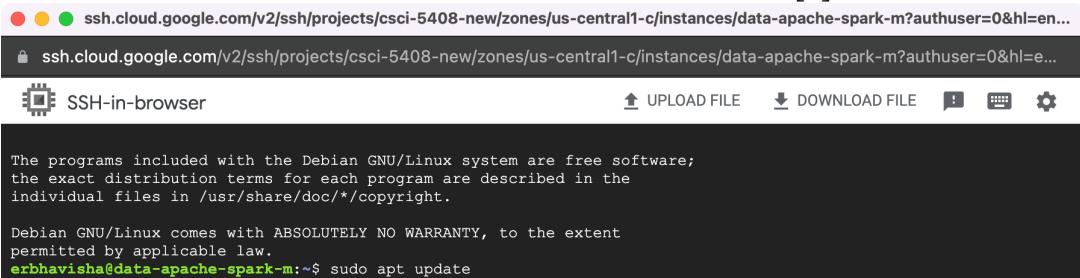


Figure 14: data-apache-spark VM connected to shell.

- Update OS using command sudo apt update as it is new instance.

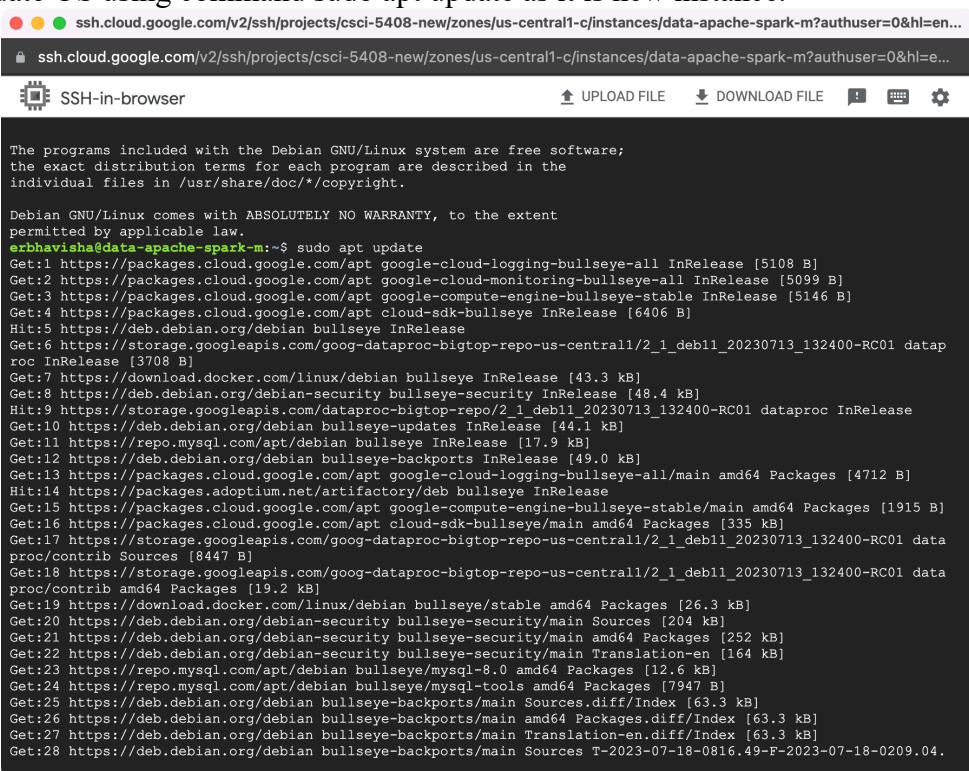
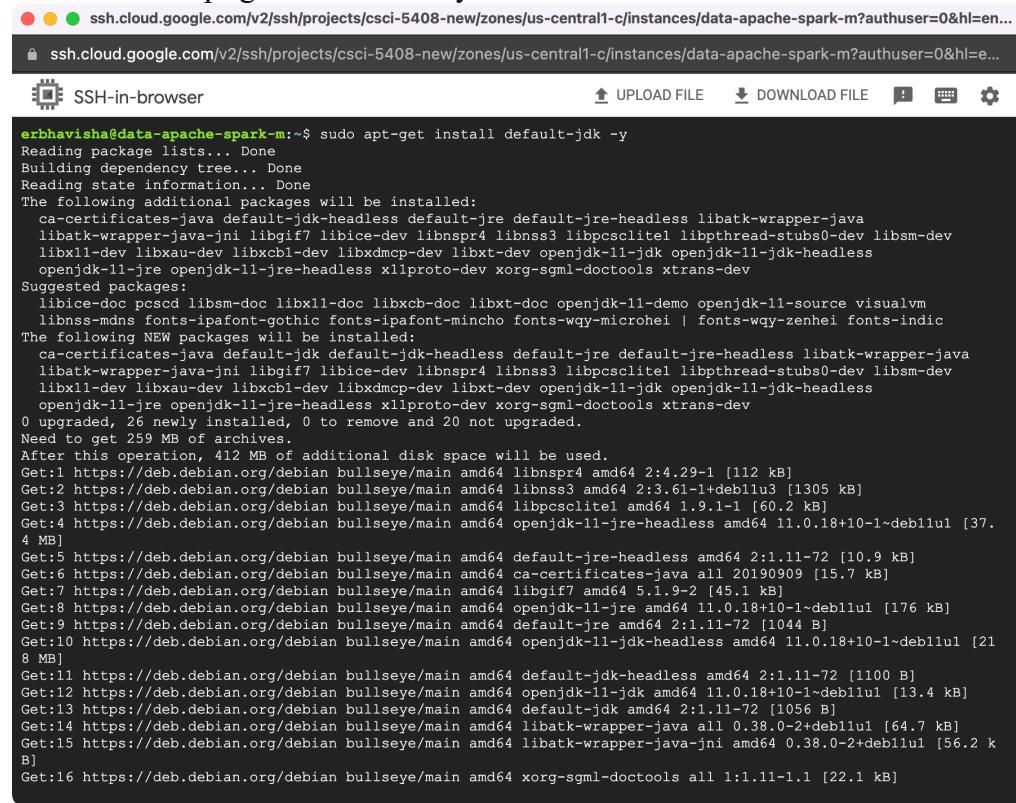


Figure 15: OS update for new instance

9. Install java and scala for the Apache Spark requirement [4]

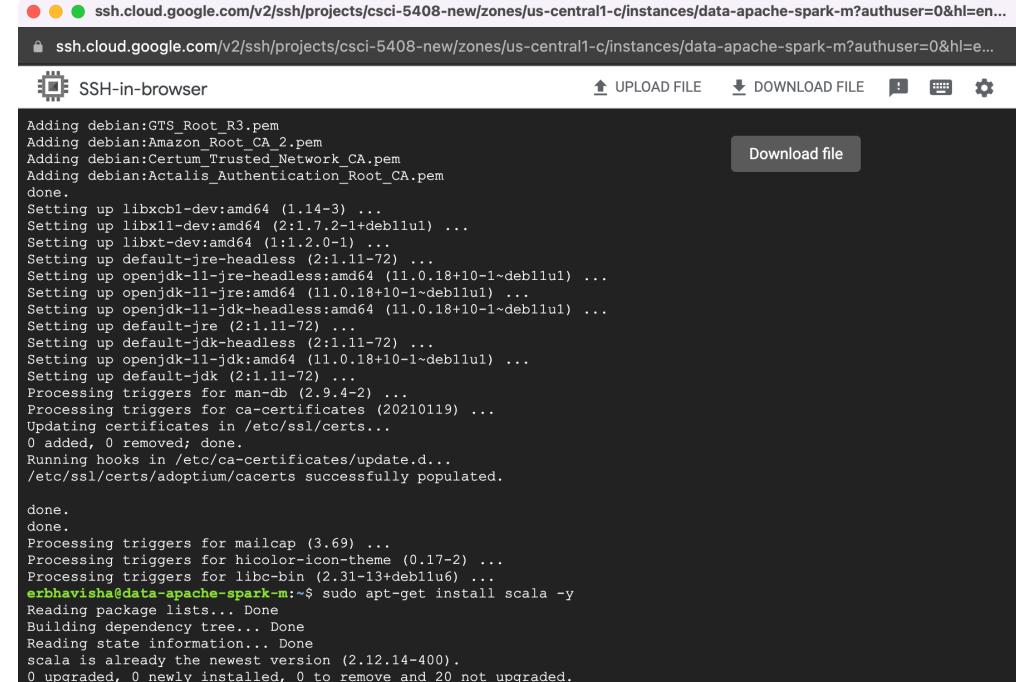
To install Java: sudo apt-get install default-jdk -y

To install Scala: sudo apt-get install scala -y



```
erbhavisha@data-apache-spark-m:~$ sudo apt-get install default-jdk -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java default-jdk-headless default-jre-headless libatk-wrapper-java
  libatk-wrapper-java-jni libgif7 libice-dev libnsspr4 libnss3 libpcsc-lite1 libpthread-stubs0-dev libsm-dev
  libx11-dev libxau-dev libxcb-dev libxdmcp-dev libxt-dev openjdk-11-jdk openjdk-11-jdk-headless
  openjdk-11-jre openjdk-11-jre-headless x11proto-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
  libice-doc pcscd libsm-doc libxcb-doc libxt-doc openjdk-11-demo openjdk-11-source visualvm
  libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  ca-certificates-java default-jdk default-jdk-headless default-jre default-jre-headless libatk-wrapper-java
  libatk-wrapper-java-jni libgif7 libice-dev libnsspr4 libnss3 libpcsc-lite1 libpthread-stubs0-dev libsm-dev
  libx11-dev libxau-dev libxcb-dev libxdmcp-dev libxt-dev openjdk-11-jdk openjdk-11-jdk-headless
  openjdk-11-jre openjdk-11-jre-headless x11proto-dev xorg-sgml-doctools xtrans-dev
0 upgraded, 26 newly installed, 0 to remove and 20 not upgraded.
Need to get 259 MB of archives.
After this operation, 412 MB of additional disk space will be used.
Get:1 https://deb.debian.org/debian bullseye/main amd64 libnsspr4 amd64 2:4.29-1 [112 kB]
Get:2 https://deb.debian.org/debian bullseye/main amd64 libnss3 amd64 2:3.61-1+deb11u3 [1305 kB]
Get:3 https://deb.debian.org/debian bullseye/main amd64 libpcsc-lite1 amd64 1.9.1-1 [60.2 kB]
Get:4 https://deb.debian.org/debian bullseye/main amd64 openjdk-11-jre-headless amd64 11.0.18+10-1~deb11u1 [37.4 MB]
Get:5 https://deb.debian.org/debian bullseye/main amd64 default-jre-headless amd64 2:1.11-72 [10.9 kB]
Get:6 https://deb.debian.org/debian bullseye/main amd64 ca-certificates-java all 20190909 [15.7 kB]
Get:7 https://deb.debian.org/debian bullseye/main amd64 libgif7 amd64 5.1.9-2 [45.1 kB]
Get:8 https://deb.debian.org/debian bullseye/main amd64 openjdk-11-jre amd64 11.0.18+10-1~deb11u1 [176 kB]
Get:9 https://deb.debian.org/debian bullseye/main amd64 default-jre amd64 2:1.11-72 [1044 B]
Get:10 https://deb.debian.org/debian bullseye/main amd64 openjdk-11-jdk-headless amd64 11.0.18+10-1~deb11u1 [218 kB]
Get:11 https://deb.debian.org/debian bullseye/main amd64 default-jdk-headless amd64 2:1.11-72 [1100 B]
Get:12 https://deb.debian.org/debian bullseye/main amd64 openjdk-11-jdk amd64 11.0.18+10-1~deb11u1 [13.4 kB]
Get:13 https://deb.debian.org/debian bullseye/main amd64 default-jdk amd64 2:1.11-72 [1056 B]
Get:14 https://deb.debian.org/debian bullseye/main amd64 libatk-wrapper-java all 0.38.0-2+deb11u1 [64.7 kB]
Get:15 https://deb.debian.org/debian bullseye/main amd64 libatk-wrapper-java-jni amd64 0.38.0-2+deb11u1 [56.2 kB]
Get:16 https://deb.debian.org/debian bullseye/main amd64 xorg-sgml-doctools all 1:1.11-1.1 [22.1 kB]
```

Figure 16: Installing Java



```
Adding debian:GTS_Root_R3.pem
Adding debian:Amazon_Root_CA_2.pem
Adding debian:Certum Trusted Network CA.pem
Adding debian:Actalis_Authentication_Root_CA.pem
done.
Setting up libxcb1-dev:amd64 (1.14-3) ...
Setting up libx11-dev:amd64 (2:1.7.2-1+deb11u1) ...
Setting up libxt-dev:amd64 (1:1.2.0-1) ...
Setting up default-jre-headless (2:1.11-72) ...
Setting up openjdk-11-jre-headless:amd64 (11.0.18+10-1~deb11u1) ...
Setting up openjdk-11-jre:amd64 (11.0.18+10-1~deb11u1) ...
Setting up openjdk-11-jdk-headless:amd64 (11.0.18+10-1~deb11u1) ...
Setting up default-jre (2:1.11-72) ...
Setting up default-jdk-headless (2:1.11-72) ...
Setting up openjdk-11-jdk:amd64 (11.0.18+10-1~deb11u1) ...
Setting up default-jdk (2:1.11-72) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for ca-certificates (20210119) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
/etc/ssl/certs/adoptium/cacerts successfully populated.

done.
done.
Processing triggers for mailcap (3.69) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for libc-bin (2.31-13+deb11u6) ...
erbhavisha@data-apache-spark-m:~$ sudo apt-get install scala -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
scala is already the newest version (2.12.14-400).
0 upgraded, 0 newly installed, 0 to remove and 20 not upgraded.
```

Figure 17: Installing Scala

10. After installation, check the version using below commands [4]

```
java -version  
scala -version.
```

```
erbhavisha@data-apache-spark-m:~$ java -version  
openjdk version "11.0.19" 2023-04-18  
OpenJDK Runtime Environment Temurin-11.0.19+7 (build 11.0.19+7)  
OpenJDK 64-Bit Server VM Temurin-11.0.19+7 (build 11.0.19+7, mixed mode)  
erbhavisha@data-apache-spark-m:~$ scala -version  
Scala code runner version 2.12.14 -- Copyright 2002-2021, LAMP/EPFL and Lightbend, Inc.  
erbhavisha@data-apache-spark-m:~$
```

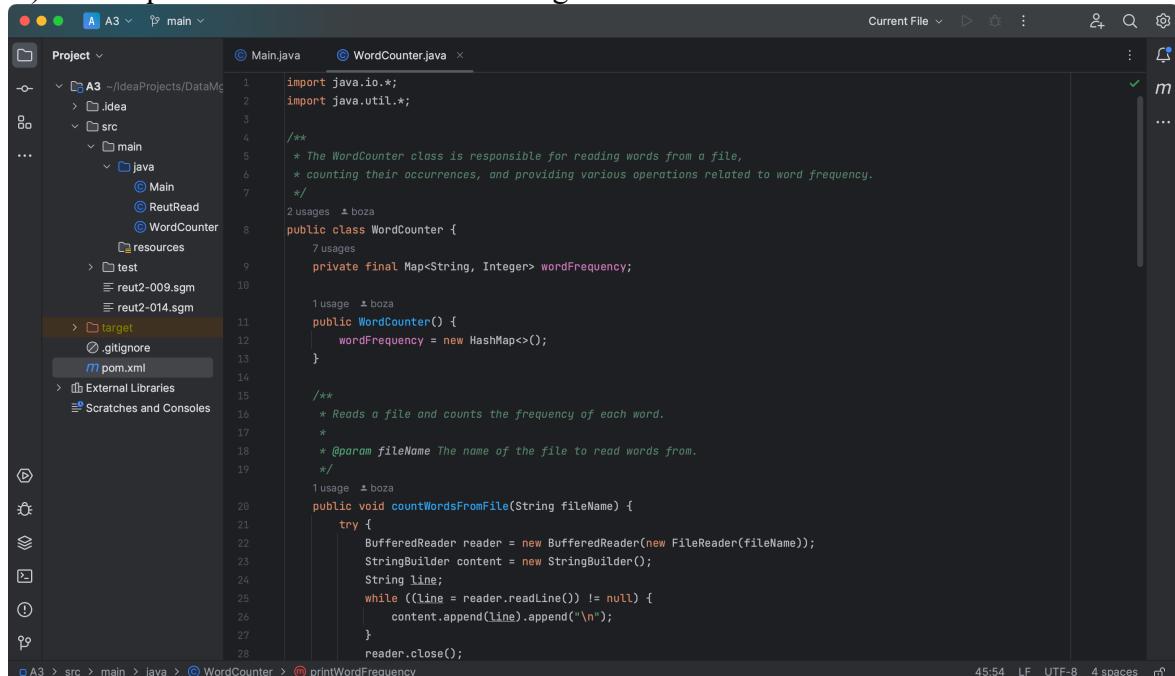
Figure 18: Java and Scala version check

11. Test Spark Shell [4]

```
erbhavisha@data-apache-spark-m:~$ sudo /opt/spark/bin/spark-shell  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For Spark, use setLogLevel(newLevel).  
23/07/30 04:46:17 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in  
n-java classes where applicable  
Spark context Web UI available at http://data-apache-spark-m.us-central1-c.c.cscl-5408-new.internal:4040  
Spark context available as 'sc' (master = local[*], app id = local-1690692378575).  
Spark session available as 'spark'.  
Welcome to  
  
version 3.4.1  
  
Using Scala version 2.12.17 (OpenJDK 64-Bit Server VM, Java 11.0.19)  
Type in expressions to have them evaluated.  
Type :help for more information.  
  
scala>
```

Figure 19: Spark shell check

12. Write a MapReduce program using Java (WordCounter.java Engine) to count (frequency count) the unique words found in “reut2-009.sgm”.



The screenshot shows the IntelliJ IDEA interface with the project 'A3' open. The 'src' directory contains a 'main' package with a 'java' subdirectory. Inside 'java', there are three files: 'Main.java', 'ReutRead.java', and 'WordCounter.java'. The 'WordCounter.java' file is currently selected and shown in the editor. The code implements a WordCounter class that reads words from a file and counts their occurrences using a HashMap. It includes methods for reading the file and counting words.

```
import java.io.*;  
import java.util.*;  
  
/**  
 * The WordCounter class is responsible for reading words from a file,  
 * counting their occurrences, and providing various operations related to word frequency.  
 */  
public class WordCounter {  
    private final Map<String, Integer> wordFrequency;  
  
    public WordCounter() {  
        wordFrequency = new HashMap<>();  
    }  
  
    /**  
     * Reads a file and counts the frequency of each word.  
     *  
     * @param fileName The name of the file to read words from.  
     */  
    public void countWordsFromFile(String fileName) {  
        try {  
            BufferedReader reader = new BufferedReader(new FileReader(fileName));  
            StringBuilder content = new StringBuilder();  
            String line;  
            while ((line = reader.readLine()) != null) {  
                content.append(line).append("\n");  
            }  
            reader.close();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
        wordFrequency.putAll(countWords(content.toString()));  
    }  
  
    private Map<String, Integer> countWords(String content) {  
        Map<String, Integer> wordCountMap = new HashMap<>();  
        String[] words = content.toLowerCase().split("\\s+");  
        for (String word : words) {  
            if (!word.isEmpty()) {  
                wordCountMap.put(word, wordCountMap.getOrDefault(word, 0) + 1);  
            }  
        }  
        return wordCountMap;  
    }  
}
```

Figure 20: Java program for reading words from a file, counting their occurrences, and providing various operations related to word frequency.

13. Program output in IntelliJ

```
...  
*****  
Word count:  
*****  
cancel: 1  
undermining: 1  
ccc: 5  
datelinebodycolombia: 1  
nysetitledeadline: 1  
japanesemoney: 2  
unavailable: 3  
stressed: 1  
saidproceeds: 1  
costing: 1  
salary: 4  
pollution: 1  
child: 1  
commercialcompanies: 1  
wouldve: 1  
datetopicstopicsplacesdusadplacespeoplepeopleorgsorgsexchangesexchangescompaniescompaniesunknown: 197  
theintermediate: 2  
pick: 5  
aspokesman: 2  
denominationsof: 1  
goldschmidt: 2  
energy: 44  
reinsure: 1  
buggies: 1  
antonio: 7  
Process finished with exit code 0
```

Figure 21: IntelliJ output -1.

```
...  
*****  
graham  
retentions  
billwhich  
cal  
cap  
itwanted  
datetopicscrudedtopicsplacesdmadagascarplacespeoplepeopleorgsorgsexchangesexchangescompaniescompaniesunknown  
ofpolystyrene  
alarm  
datetopicsdacqdtopicsplacesdcanaxaddukdplacespeoplepeopleorgsorgsexchangesexchangescompaniescompaniesunknown  
stamps  
datetopictopisplacesdcanaxaddsingaporeddphilippinesddindonesiasiaddthailandddbruneiidplacespeoplepeopleorgsorgsexchangesexchangescompaniescompaniesunknown  
writtenbefore  
bcmztradepsurplus  
efficacy  
detailedreports  
whichmontison  
interestand  
reprogrammings  
wouldbenefit  
misinterpreted  
*****  
Most common words:  
*****  
the  
Process finished with exit code 0
```

Figure 22: IntelliJ output -2

14. Program output in GCP

```

ssh.cloud.google.com/v2/ssh/projects/csci-5408-new/zones/us-central1-c/instances/data-apache-spark-m?authuser=0&hl=en...
ssh.cloud.google.com/v2/ssh/projects/csci-5408-new/zones/us-central1-c/instances/data-apache-spark-m?authuser=0&hl=en...
SSH-in-browser
Linux data-apache-spark-m 5.10.0-23-cloud-amd64 #1 SMP Debian 5.10.179-1 (2023-05-12) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul 30 04:30:26 2023 from 35.235.245.129
erbhavisha@data-apache-spark-m:~$ javac WordCounter.java Main.java
erbhavisha@data-apache-spark-m:~$ java Main
java.io.FileNotFoundException: /Users/bhavishaoza/IdeaProjects/DataMgmt/A3/src/reut2-009.sgm (No such file or d
irectory)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:112)
        at java.base/java.io.FileReader.<init>(FileReader.java:60)
        at WordCounter.countWordsFromFile(WordCounter.java:22)
        at Main.main(Main.java:11)

Word count:
Least common words:
Most common words:
erbhavisha@data-apache-spark-m:~$ 
```

Transferred 5 items

pom.xml	✓
reut2-009.sgm	✓
reut2-014.sgm	✓
WordCounter.java	✓
Main.java	✓

Figure 23: GCP output

15. An algorithm of MapReduce program:

1. Start the WordCounter class.
2. Declare a private instance variable `wordFrequency` of type `Map<String, Integer>` to store the frequency count of each word.
3. Implement the constructor `WordCounter()` to initialize the `wordFrequency` map.
4. Implement the method `countWordsFromFile(String fileName)` to read the content of the file and count the frequency of each word. The method takes the file name as input.
 - Open the file using a BufferedReader.
 - Read the content line by line, appending it to a StringBuilder.
 - Close the BufferedReader.
 - Remove all non-alphabetic characters and convert the content to lowercase.
 - Split the content into individual words using whitespace as the delimiter.
 - For each word, update its frequency in the `wordFrequency` map.
5. Implement the method `printWordFrequency()` to print the word count.
 - Iterate through the `wordFrequency` map and print each word along with its frequency.
6. Implement the method `getLeastCommonWords()` to find the least common words.
 - Initialize a variable `minCount` to Integer.MAX_VALUE.
 - Iterate through the values of the `wordFrequency` map and update `minCount` with the minimum frequency found.
 - Return an array of words that have a frequency equal to `minCount`.
7. Implement the method `getMostCommonWords()` to find the most common words.
 - Initialize a variable `maxCount` to Integer.MIN_VALUE.

- Iterate through the values of the `wordFrequency` map and update `maxCount` with the maximum frequency found.
 - Return an array of words that have a frequency equal to `maxCount`.
8. Implement the helper method `getWordsWithFrequency(int frequency)` to retrieve words with a specific frequency.
- Use stream operations to filter the `wordFrequency` map and collect words with the given `frequency`.
9. In the `main` method:
- Create an instance of the WordCounter class.
 - Call the `countWordsFromFile(fileName)` method to read the file and count words.
 - Call the `printWordFrequency()` method to print the word count.
 - Call the `getLeastCommonWords()` method to find the least common words and print them.
 - Call the `getMostCommonWords()` method to find the most common words and print them.

16. The words that have highest frequencies:

+++++

Most common words:

+++++

the: 6294

17. The words that have lowest frequencies:

+++++

Least common words:

+++++

cancel: 1

undermining: 1

datelinebodycolombia: 1

nysetitledateline: 1

stessed: 1

saidproceeds: 1

costing: 1

pollution: 1

childs: 1

commercialcompanies: 1

wouldveto: 1

denominationsof: 1

reinsure: 1

buggies: 1

spreading: 1

astudebaker: 1

carriers: 1

bcpanteraslpanttob: 1

packagetitleauthor: 1

averagedaily: 1

newemployment: 1

orcombination: 1
nonrecurring: 1
roll: 1
unknowntexttitlepenobscot: 1
wilsonis: 1
bbbminus: 1
athreefortwo: 1
fromtermination: 1
postings: 1
planpreviously: 1
thedepartments: 1
bcnmspharmaceuticalltn: 1
boostdemand: 1
broadcastingcorp: 1
ormillion: 1
posture: 1
pastdue: 1
throughlloyds: 1
morningcut: 1
o: 1
r: 1
bcmalaysiancentralban: 1
givetime: 1
datelinebodycalifornia: 1
ltchina: 1
x: 1
butbrokers: 1
beturned: 1
reportof: 1
raisedits: 1
empty: 1
amounted: 1
toaplus: 1
divergence: 1
datelinebodywaste: 1
categorically: 1
cie: 1
graincommodity: 1
bcgartnergroupltgart: 1
sevenyears: 1
asprimary: 1
scanners: 1
datelinebodyreal: 1
isoglucone: 1
rolein: 1
visitgeorgetown: 1

cit: 1
offs: 1
bcrsicorpltrsicndq: 1
datetopicsdcpidtopicsplacesdluxembourggdplacespeoplepeopleorgsdecorgsexchangesexchan
gescompaniescompaniesunknown: 1
silvercopper: 1
bctwasoldfourm: 1
olson: 1
itsintroduction: 1
productsld: 1
telephonebased: 1
congressmen: 1
wascommitted: 1
incanada: 1
unknowntexttitlecray: 1
corpccc: 1
allegations: 1
cancer: 1
isdetermined: 1
telephoniques: 1
bycommercial: 1
bcusshoeinc: 1
denshindenwa: 1
asturias: 1
pill: 1
investigators: 1
oregonutah: 1
whenit: 1
reformer: 1
protecting: 1

Full code can be found in the given repository: <https://git.cs.dal.ca/boza/csci-5408/-tree/main/A3>.

References:

- [1] "Download intelliJ idea – the leading Java and Kotlin IDE," *JetBrains* [Online]. Available: <https://www.jetbrains.com/idea/download/?section=mac> [Accessed: 01 July 2023].
- [2] Google, "Google Cloud Platform," Google, [Online]. Available: <https://console.cloud.google.com/> [Accessed: 25 July 2023].
- [3] "The most popular database for modern apps", MongoDB, 2020. [Online]. Available: <https://www.mongodb.com/> [Accessed: 20 July 2023].
- [4] D. Tucakov, "How to Install Spark on Ubuntu," phoenixNAP, [Online]. Available: <https://phoenixnap.com/kb/install-spark-on-ubuntu>. [Accessed: 23 July 2023].
- [5] "Build, test, and debug regex," *regex101* [Online]. Available: <https://regex101.com/> [Accessed: 20 July 2023].
- [6] "Lab-5," *Brightspace Dalhousie University* [Online]. Available: <https://dal.brightspace.com/d2l/le/content/271677/viewContent/3661458/View> [Accessed: July 28, 2023].
- [7] "Lab-7," *Brightspace Dalhousie University* [Online]. Available: <https://dal.brightspace.com/d2l/le/content/271677/viewContent/3681262/View> [Accessed: July 28, 2023].
- [8] "Flowchart Maker & Online Diagram Software," *Draw.io* [Online]. Available: <https://app.diagrams.net> [Accessed: 29 July 2023].