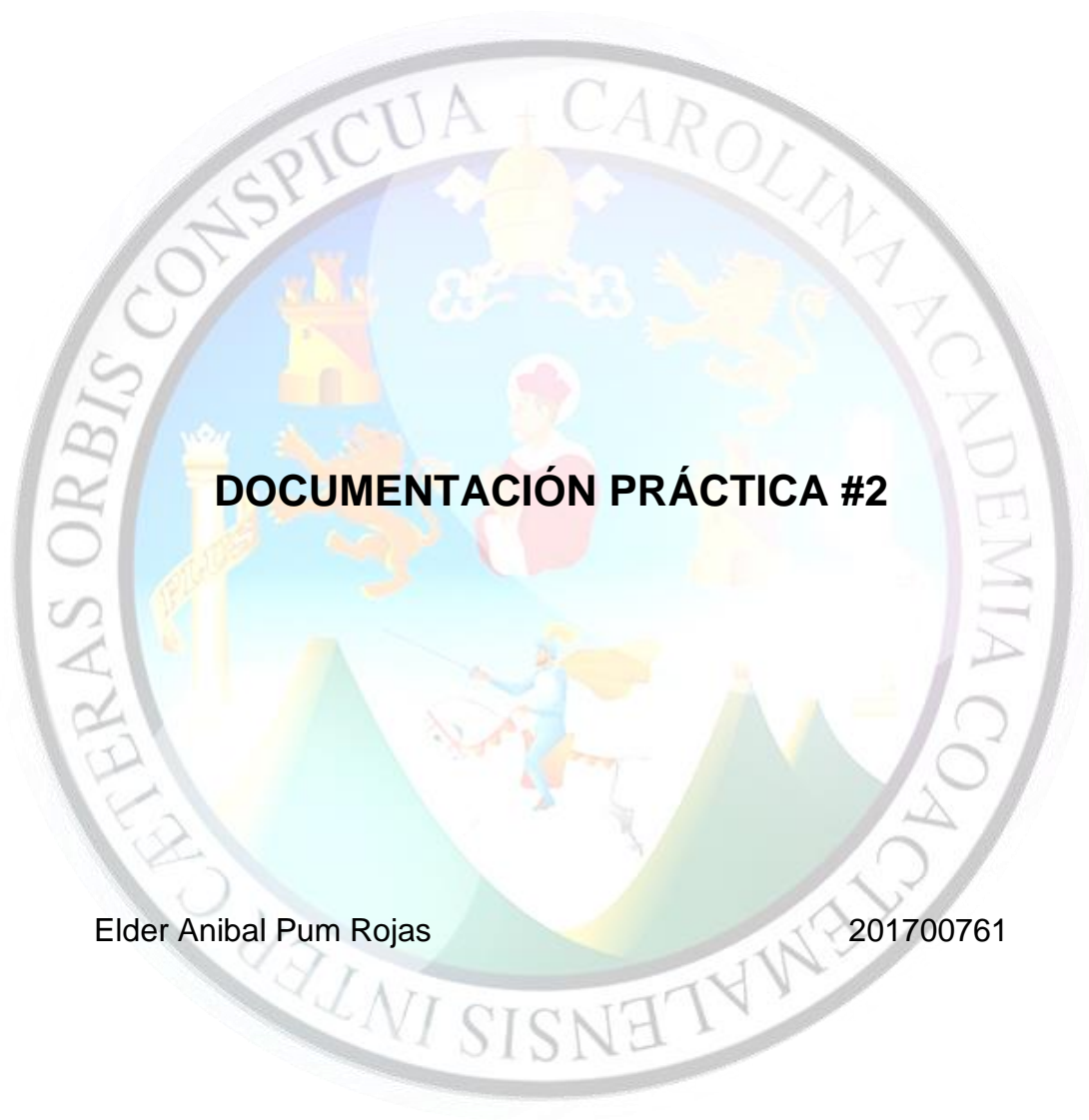


Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Seminario de Sistemas 2



DOCUMENTACIÓN PRÁCTICA #2

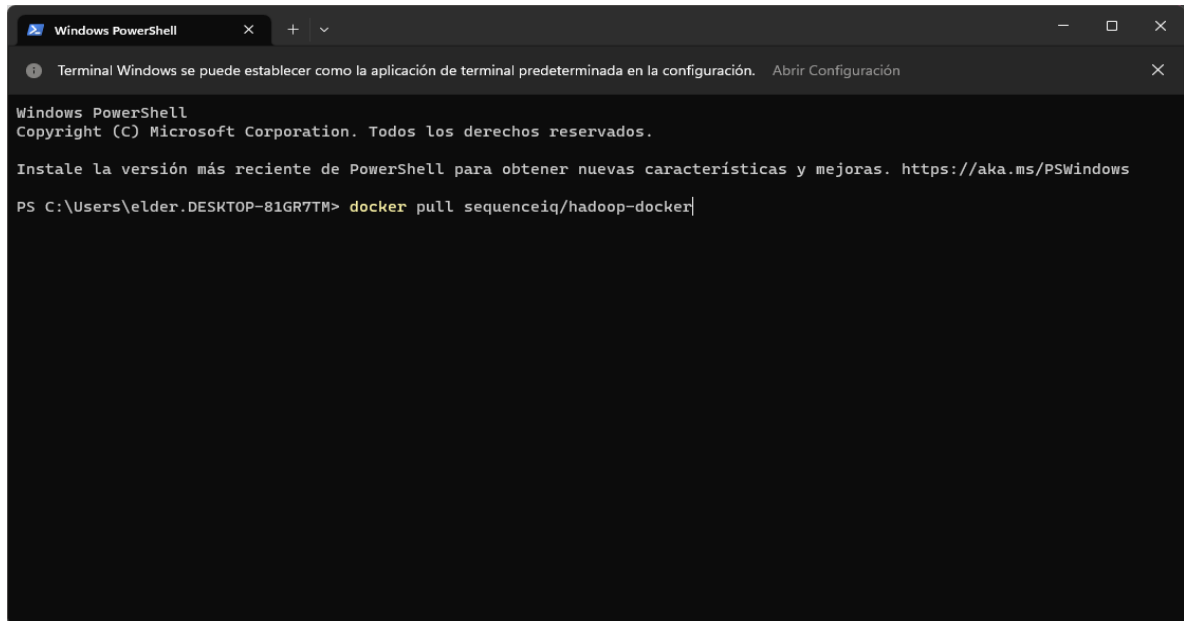
Elder Anibal Pum Rojas

201700761

Guatemala, 2 de Abril del 2023

Pasos a seguir para completar la práctica

1. Primeramente, necesitamos descargar la imagen de Hadoop, para esto utilizaremos Docker (ya sea en Windows o Linux) y correremos el siguiente comando en una terminal:

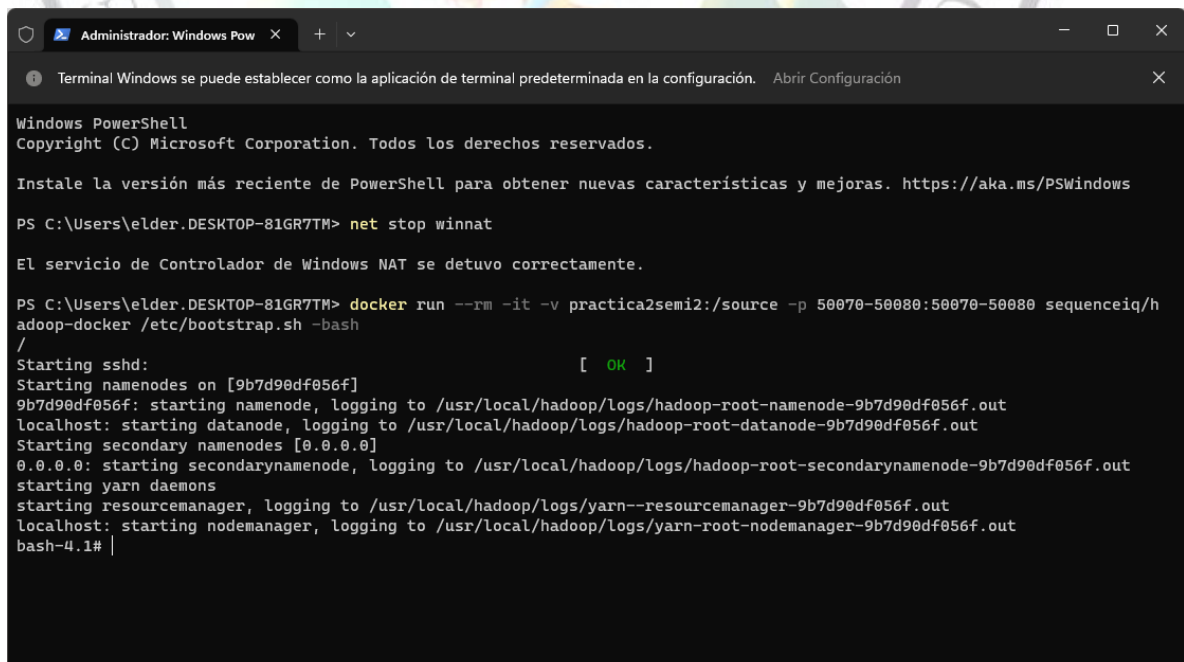


```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\elder.DESKTOP-81GR7TH> docker pull sequenceiq/hadoop-docker
```

2. Después de que se descargue la imagen de Hadoop, vamos a correrla utilizando el siguiente comando (tener en cuenta que cada vez que se apaga la imagen, esta se elimina y por ende, hay que volver a correr el comando. Además, hay que matar el proceso “winnat” porque usa los puertos que utilizaremos para Hadoop):



```
Administrador: Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

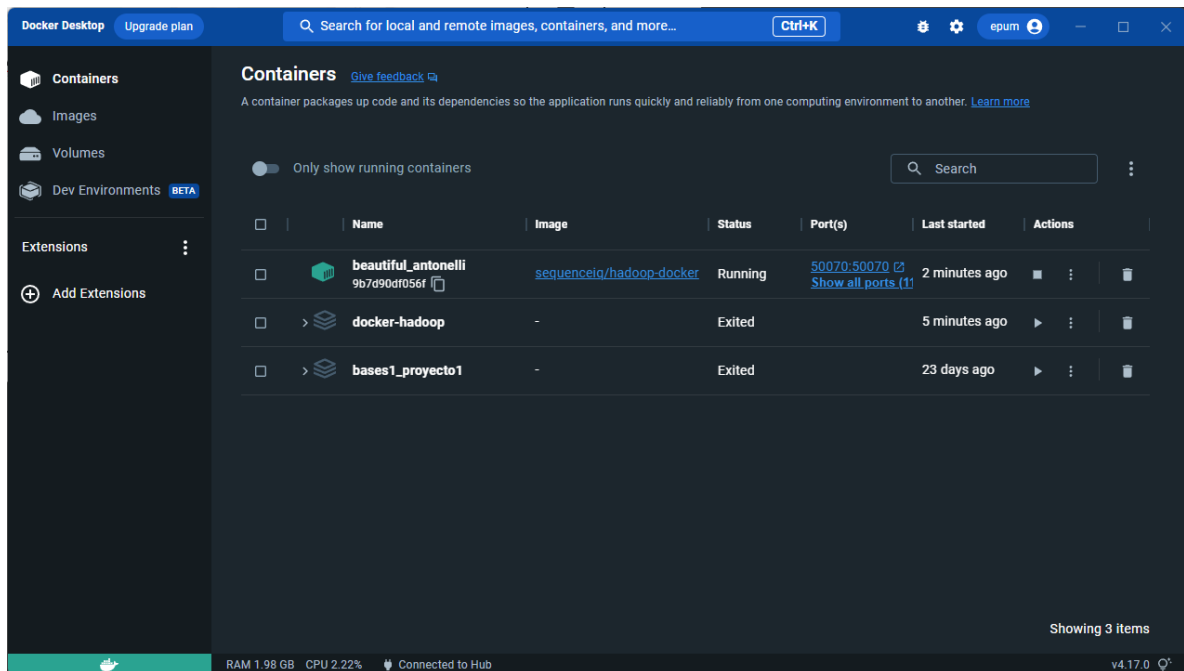
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\elder.DESKTOP-81GR7TH> net stop winnat

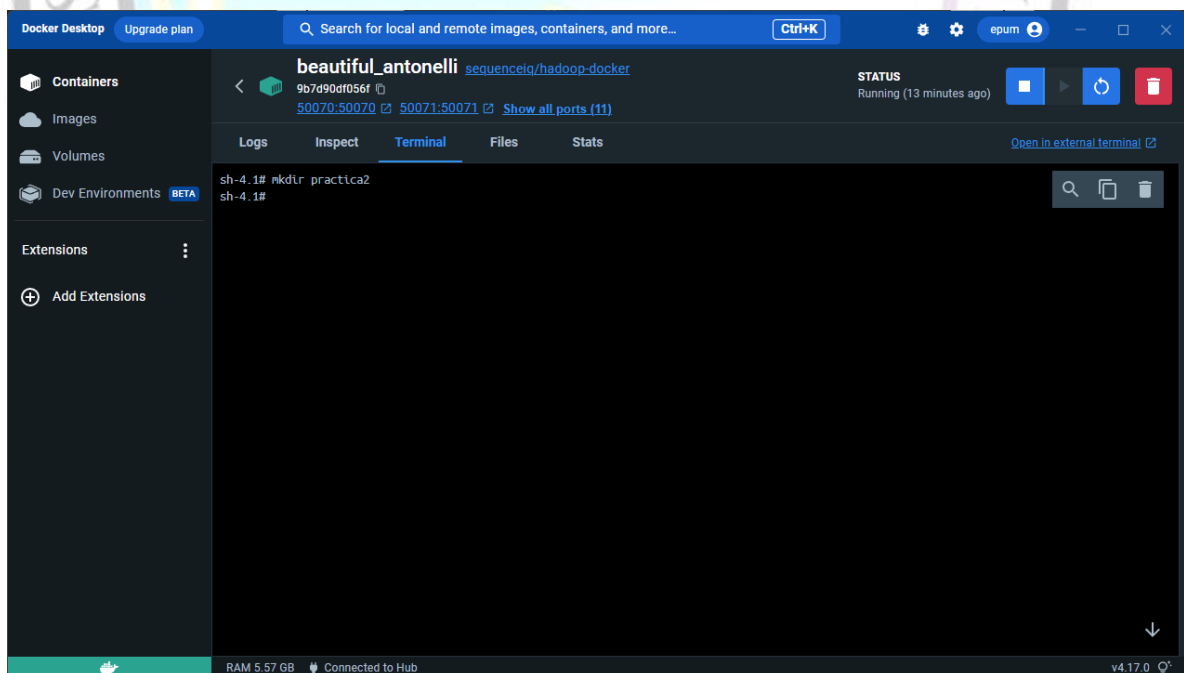
El servicio de Controlador de Windows NAT se detuvo correctamente.

PS C:\Users\elder.DESKTOP-81GR7TH> docker run --rm -it -v practica2semi2:/source -p 50070-50080:50070-50080 sequenceiq/hadoop-docker /etc/bootstrap.sh -bash
/
Starting sshd: [ OK ]
Starting namenodes on [9b7d90df056f]
9b7d90df056f: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-9b7d90df056f.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-9b7d90df056f.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-root-secondarynamenode-9b7d90df056f.out
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcemanager-9b7d90df056f.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-9b7d90df056f.out
bash-4.1#
```

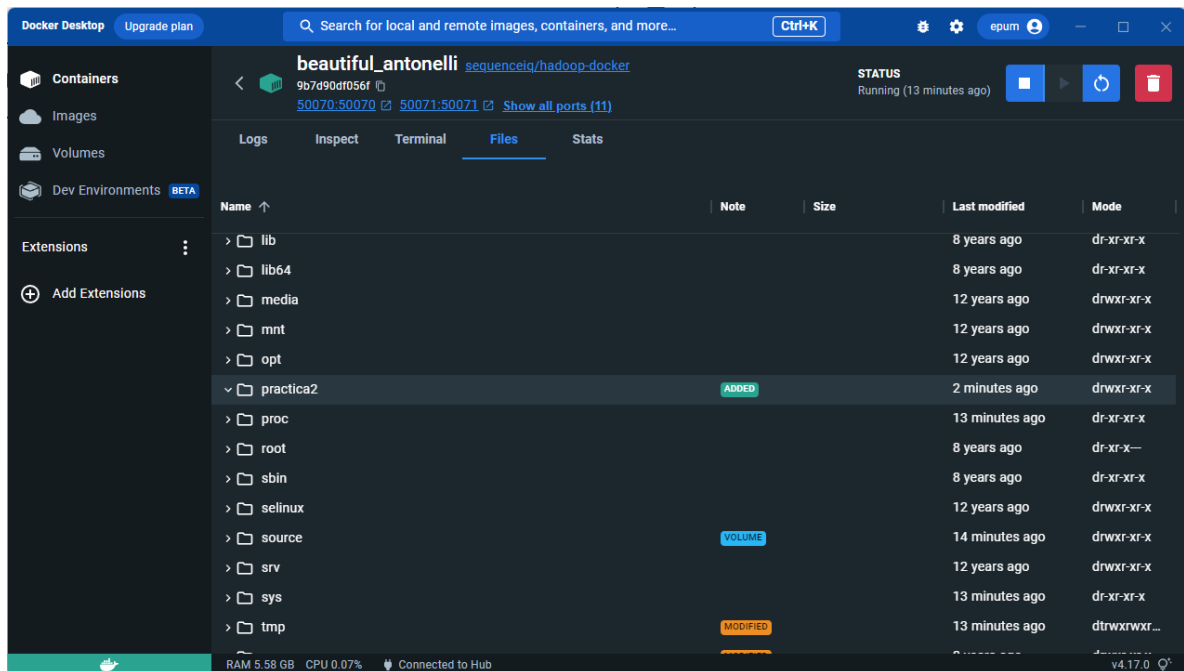
3. Revisando en Docker Desktop tenemos que el contenedor efectivamente está corriendo:



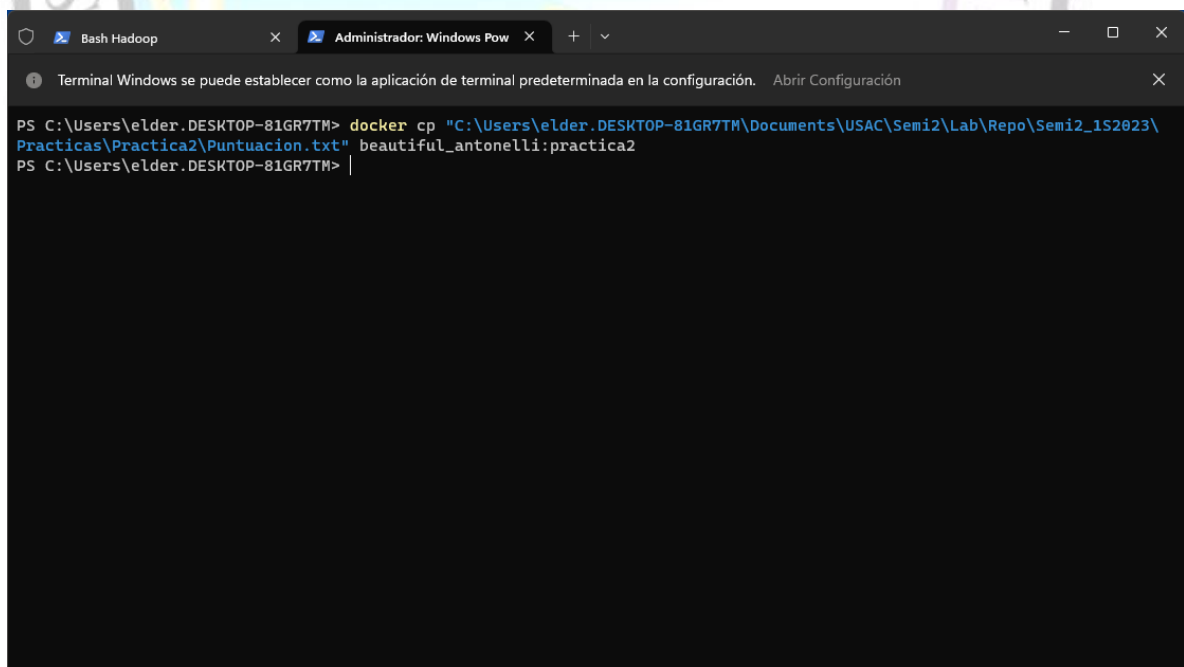
Ahora lo que vamos a hacer es crear una carpeta llamada “practica2” dentro del contenedor, para lo cual vamos a utilizar la terminal del contenedor en Docker Desktop y meter el siguiente comando:



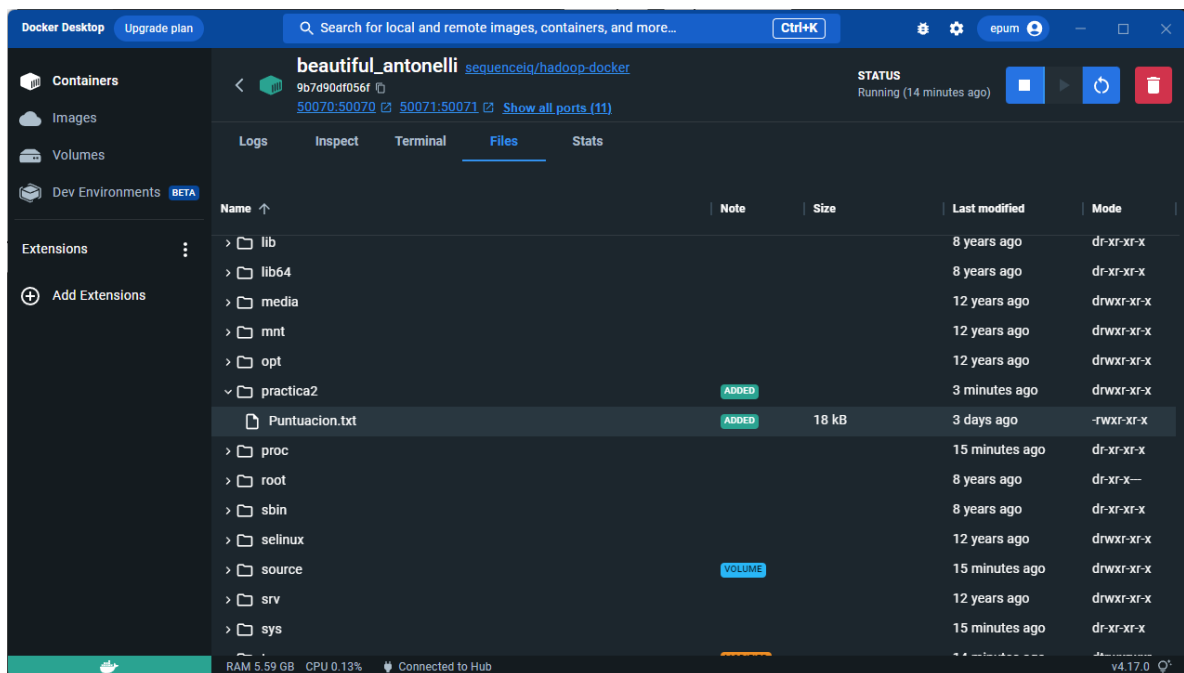
Y revisando en Docker Desktop en la parte de Files veremos que se ha creado la carpeta correctamente:



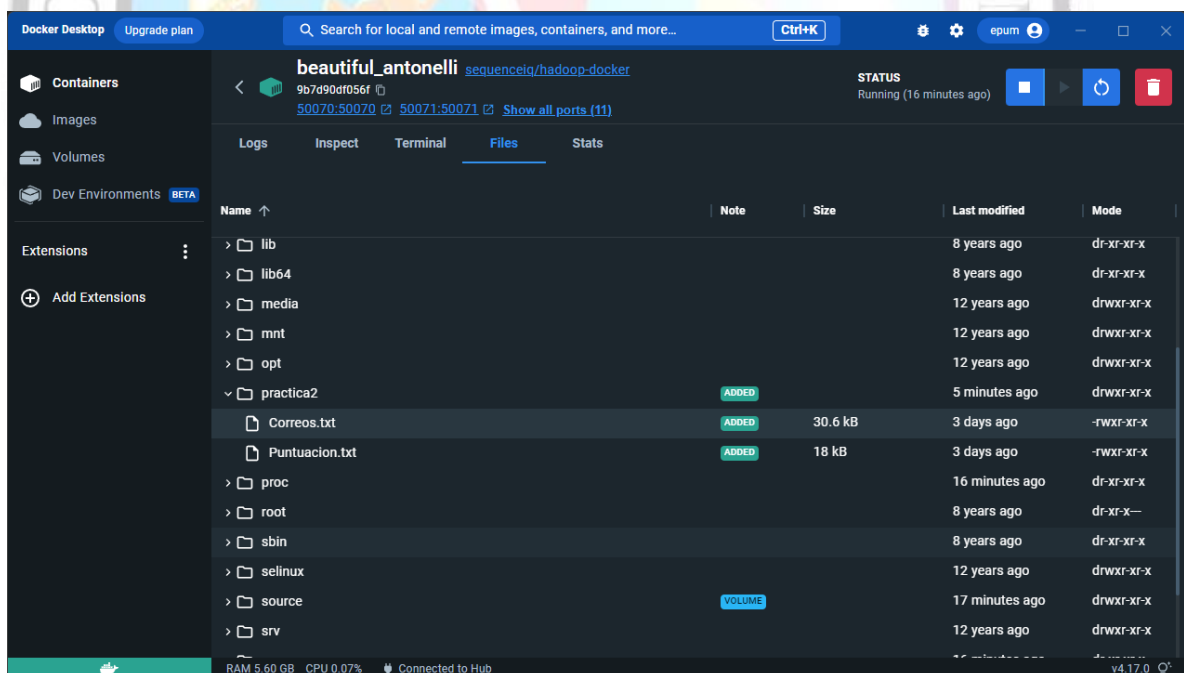
Ahora lo que vamos a hacer es copiar el archivo "Puntuacion.txt" dentro de la carpeta "practica" que acabamos de crear, para ello, vamos a utilizar una nueva terminal de Windows y usar el siguiente comando:



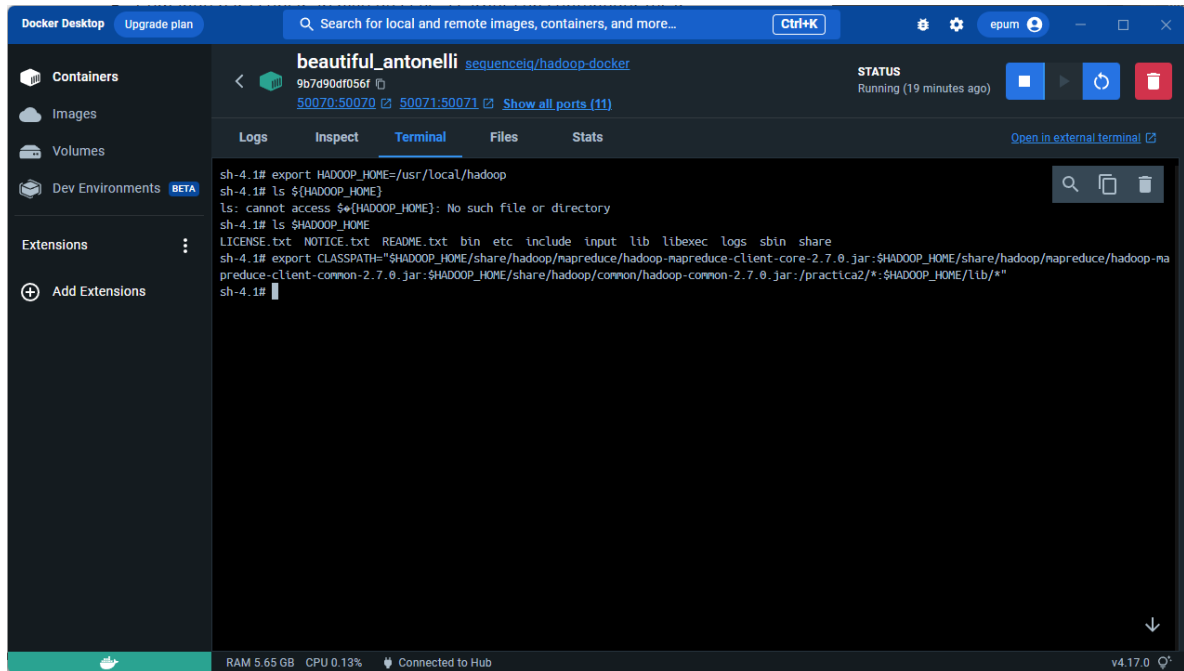
Y nuevamente verificamos que el archivo se ha agregado:



4. Vamos a hacer lo mismo para el archivo “Correos.txt” y verificaremos que el archivo esté dentro de la misma carpeta:



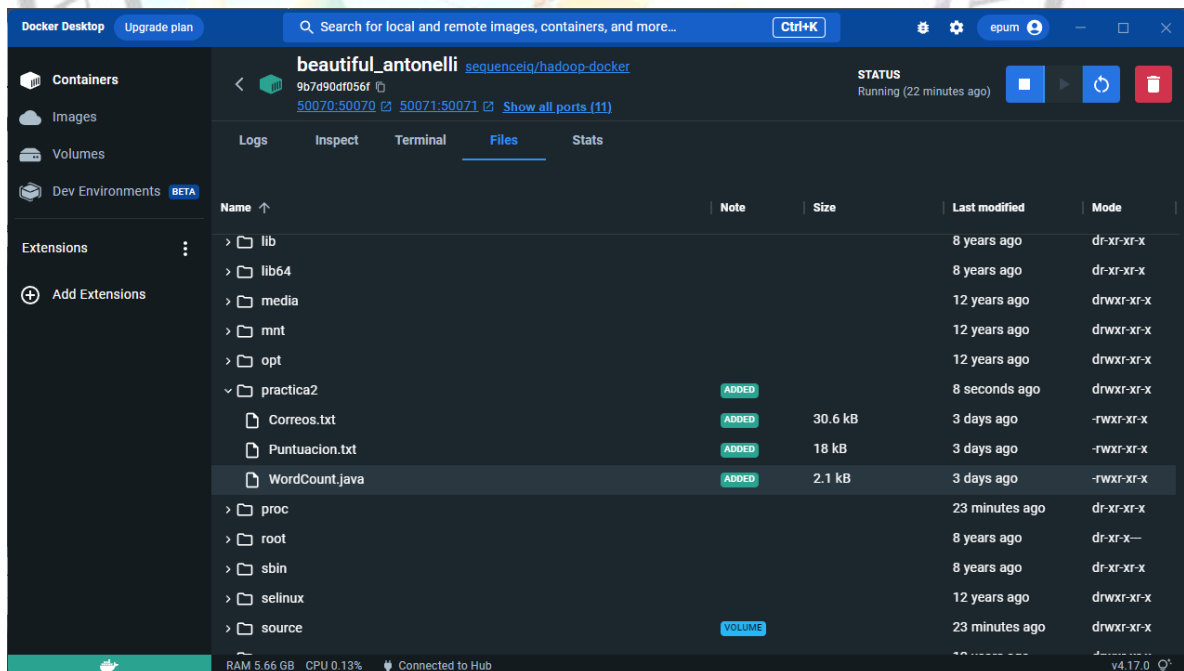
5. Ahora dentro de la terminal del contenedor vamos a iniciar las variables HADOOP_HOME y CLASSPATH:



The screenshot shows the Docker Desktop interface with a terminal window open for a container named 'beautiful_antonelli'. The terminal output shows the following commands and results:

```
sh-4.1# export HADOOP_HOME=/usr/local/hadoop
sh-4.1# ls ${HADOOP_HOME}
ls: cannot access ${HADOOP_HOME}: No such file or directory
sh-4.1# ls $HADOOP_HOME
LICENSE.txt NOTICE.txt README.txt bin etc include input lib libexec logs sbin share
sh-4.1# export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.7.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.7.0.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-2.7.0.jar:/practica2/*:$HADOOP_HOME/lib/*"
sh-4.1#
```

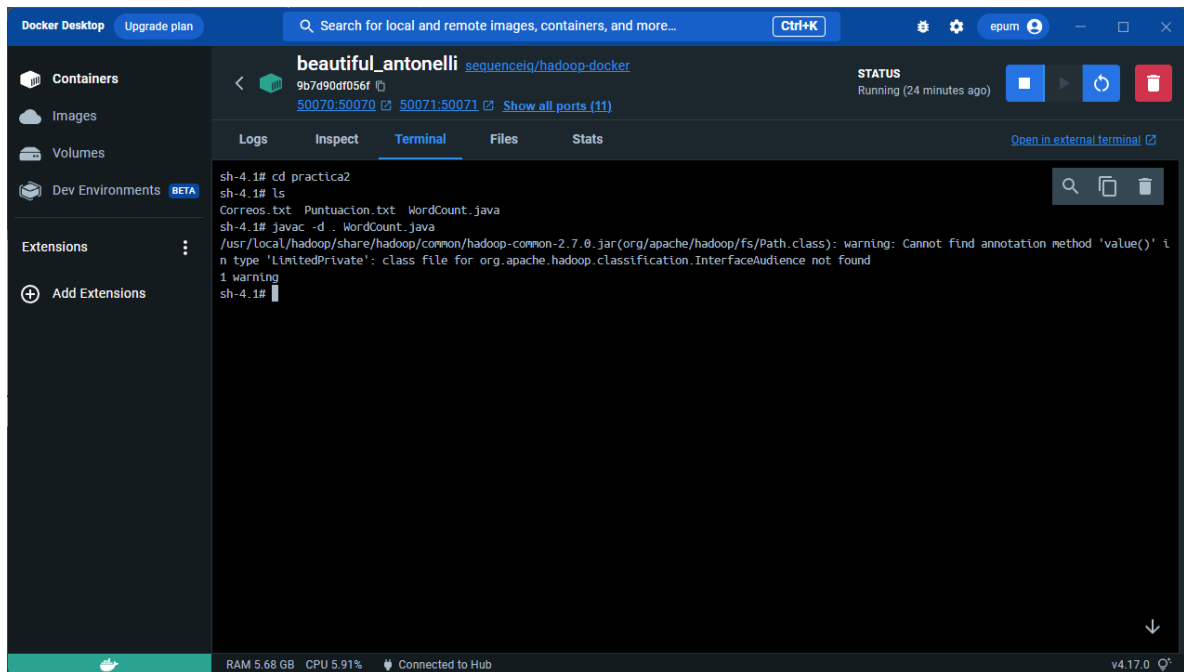
6. Antes de poder compilar, necesitamos subir igualmente al contenedor el archivo “WordCount.java” en la carpeta “practica2” utilizando el mismo método que utilizamos para subir los 2 archivos de texto anteriormente, luego de eso verificaremos que se encuentre el archivo dentro del contenedor:



The screenshot shows the Docker Desktop interface with the 'Files' tab selected for the container 'beautiful_antonelli'. The file explorer shows the following files and folders:

Name	Note	Size	Last modified	Mode
lib			8 years ago	dr-xr-xr-x
lib64			8 years ago	dr-xr-xr-x
media			12 years ago	drwxr-xr-x
mnt			12 years ago	drwxr-xr-x
opt			12 years ago	drwxr-xr-x
practica2	ADDED		8 seconds ago	drwxr-xr-x
Correos.txt	ADDED	30.6 kB	3 days ago	-rw-r--r--
Puntuacion.txt	ADDED	18 kB	3 days ago	-rw-r--r--
WordCount.java	ADDED	2.1 kB	3 days ago	-rw-r--r--
proc			23 minutes ago	dr-xr-xr-x
root			8 years ago	dr-xr-xr-x
sbin			8 years ago	dr-xr-xr-x
selinux			12 years ago	drwxr-xr-x
source	VOLUME		23 minutes ago	drwxr-xr-x

7. Vamos a compilar el archivo java con el siguiente comando:

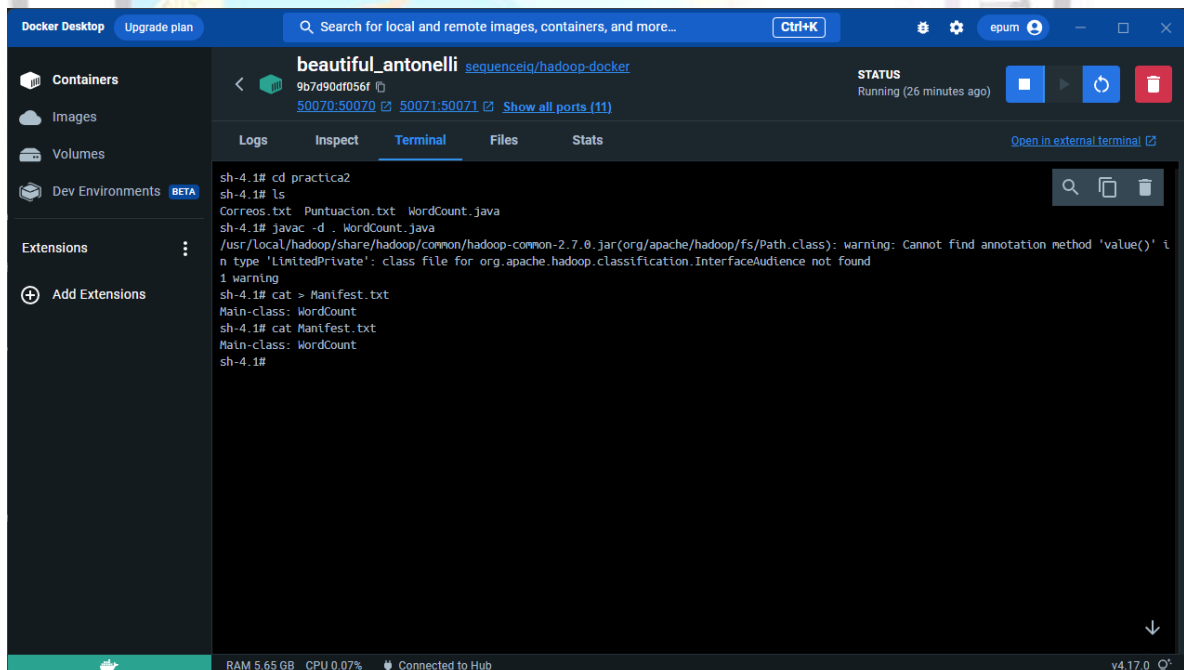


The screenshot shows the Docker Desktop interface with a terminal window open for a container named 'beautiful_antonelli'. The terminal shows the following commands and output:

```
sh-4.1# cd practica2
sh-4.1# ls
Correos.txt  Puntuacion.txt  WordCount.java
sh-4.1# javac -d . WordCount.java
/usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.0.jar(org/apache/hadoop/fs/Path.class): warning: Cannot find annotation method 'value()' i
n type 'LimitedPrivate': class file for org.apache.hadoop.classification.InterfaceAudience not found
1 warning
sh-4.1#
```

The status bar at the bottom indicates RAM 5.68 GB, CPU 5.91%, and Connected to Hub.

8. Ahora creamos un archivo Manifest.txt para indicar cual es el archivo que contiene el método Main del archivo java, para esto, utilizamos los siguientes comandos:

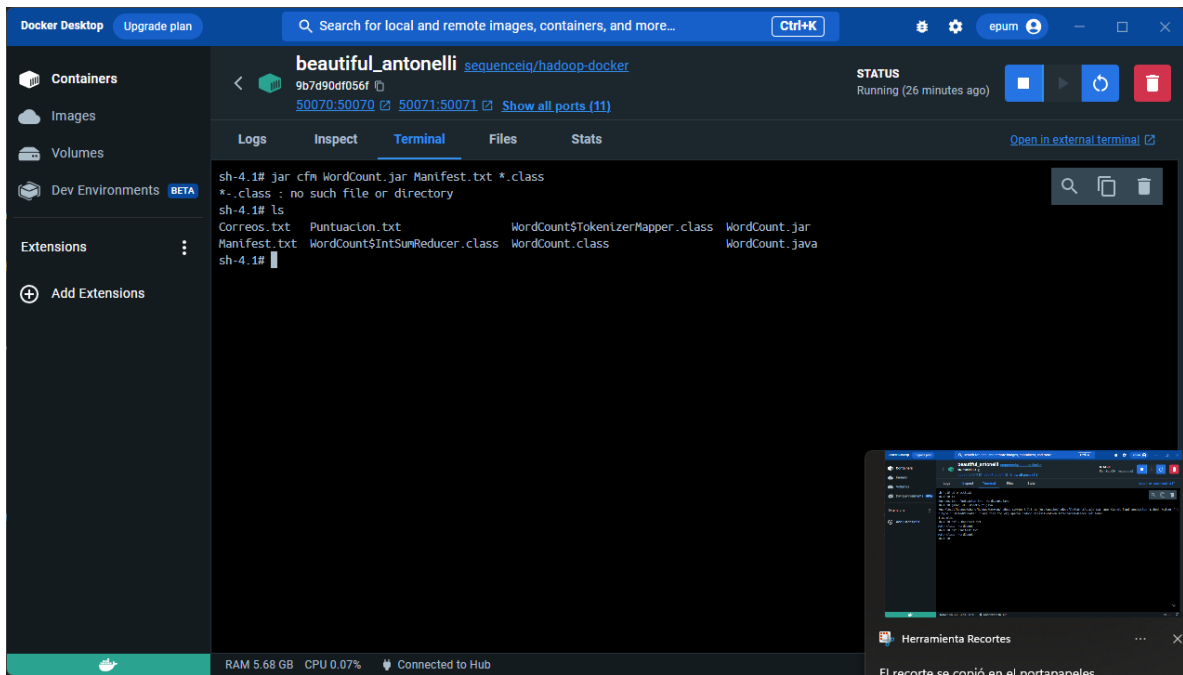


The screenshot shows the Docker Desktop interface with a terminal window open for the same container. The terminal shows the following commands and output:

```
sh-4.1# cd > Manifest.txt
Main-class: WordCount
sh-4.1# cat Manifest.txt
Main-class: WordCount
sh-4.1#
```

The status bar at the bottom indicates RAM 5.65 GB, CPU 0.07%, and Connected to Hub.

9. Ahora creamos un archivo .jar utilizando los siguientes comandos:

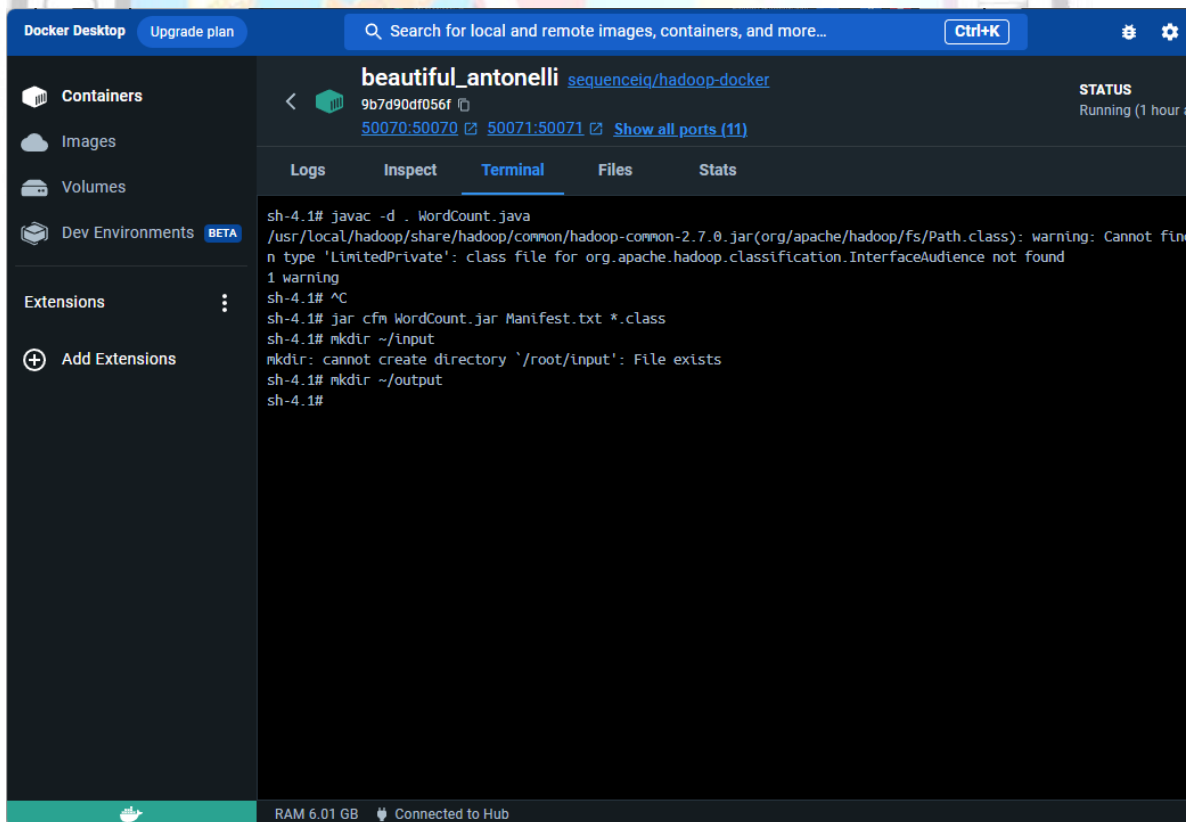


The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with 'Containers', 'Images', 'Volumes', 'Dev Environments', and 'Extensions'. The main panel shows a container named 'beautiful_antonelli' (image: sequenceiq/hadoop-docker) with ID 9b7d90df056f. The 'Terminal' tab is active, displaying the following commands and output:

```
sh-4.1# jar cfm WordCount.jar Manifest.txt *.class
*.class : no such file or directory
sh-4.1# ls
Correos.txt      Puntuacion.txt      WordCount$TokenizerMapper.class  WordCount.jar
Manifest.txt     WordCount$IntSumReducer.class  WordCount.class                  WordCount.java
sh-4.1#
```

At the bottom, system stats show RAM 5.68 GB and CPU 0.07%. A small inset window in the bottom right shows a file explorer view.

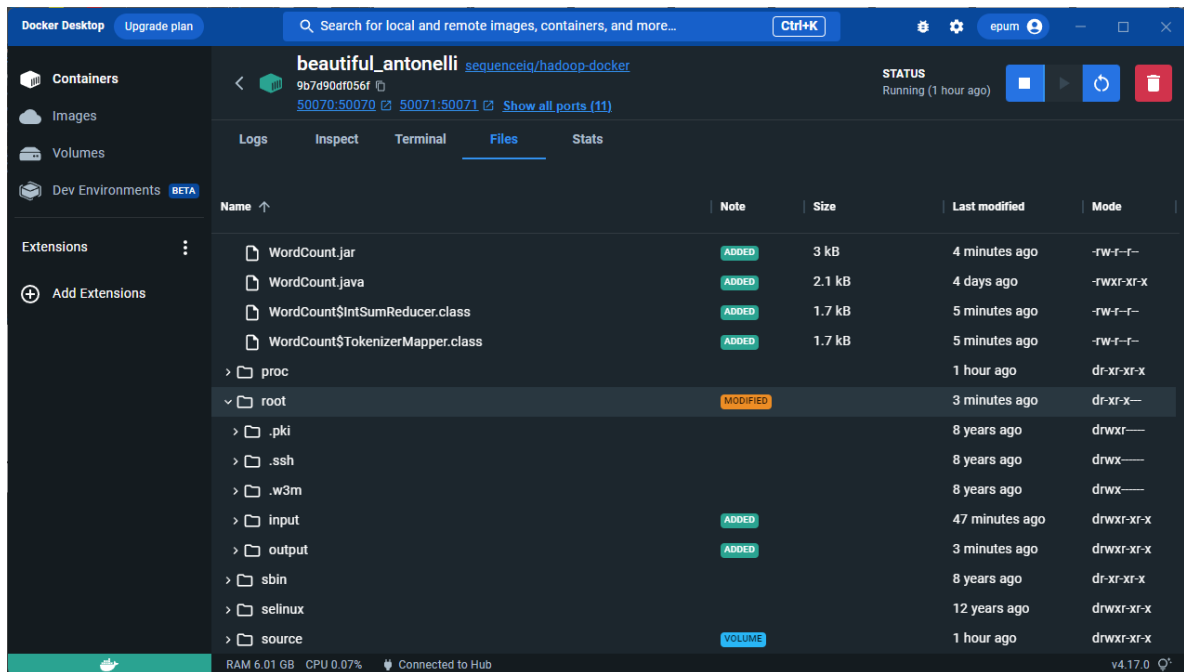
10. Ahora vamos a crear dentro de la carpeta “root” del contenedor las carpetas “input” y “output” utilizando los siguientes comandos:



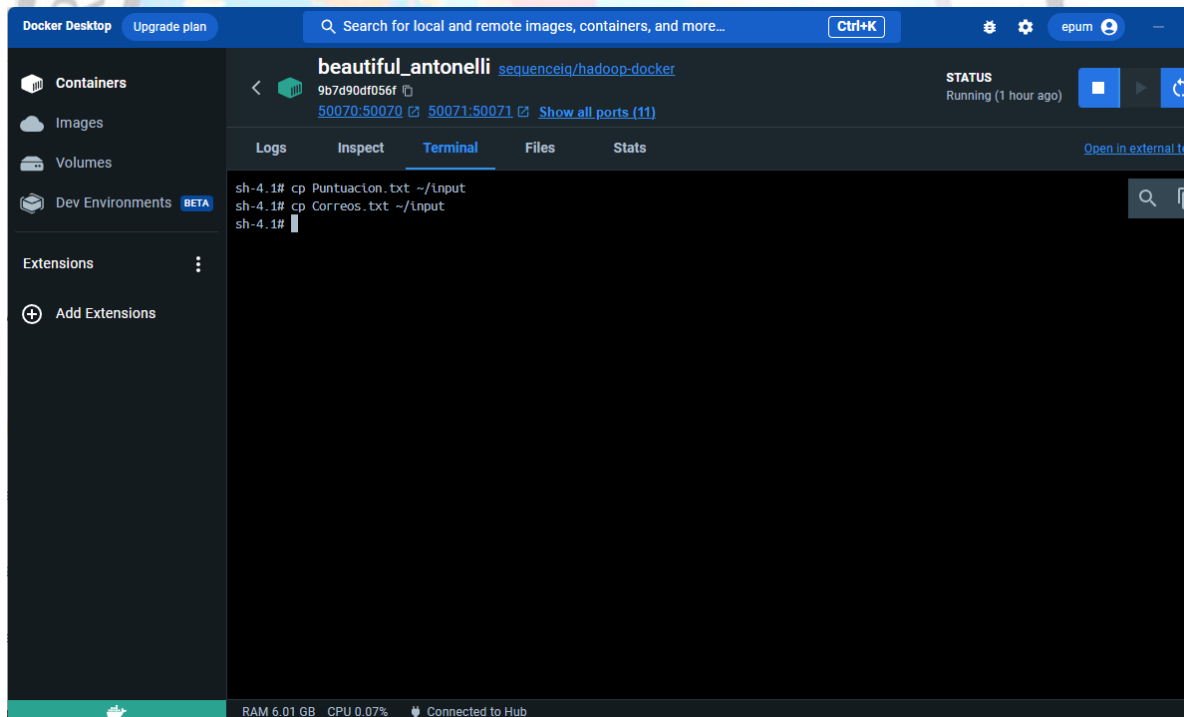
The screenshot shows the Docker Desktop interface with the same container 'beautiful_antonelli'. The 'Terminal' tab displays the following commands and output:

```
sh-4.1# javac -d . WordCount.java
/usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.0.jar(org/apache/hadoop/fs/Path.class): warning: Cannot find
n type 'LimitedPrivate': class file for org.apache.hadoop.classification.InterfaceAudience not found
1 warning
sh-4.1# ^C
sh-4.1# jar cfm WordCount.jar Manifest.txt *.class
sh-4.1# mkdir ~/input
mkdir: cannot create directory '/root/input': File exists
sh-4.1# mkdir ~/output
sh-4.1#
```

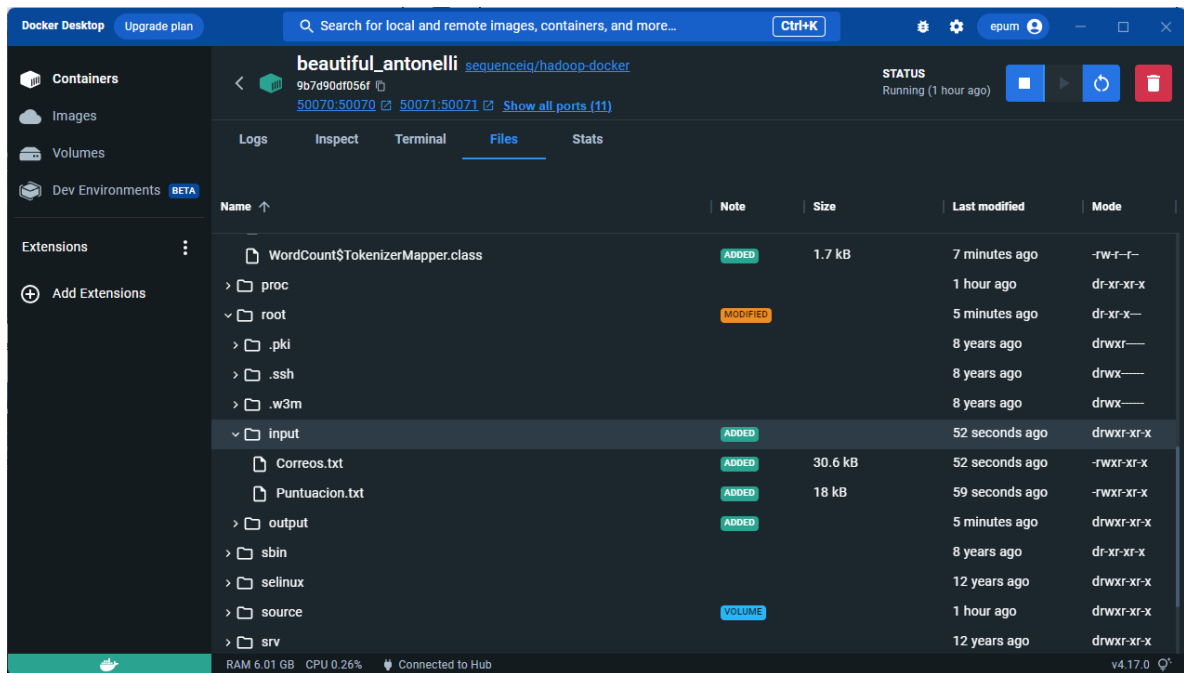
At the bottom, system stats show RAM 6.01 GB and 'Connected to Hub'.



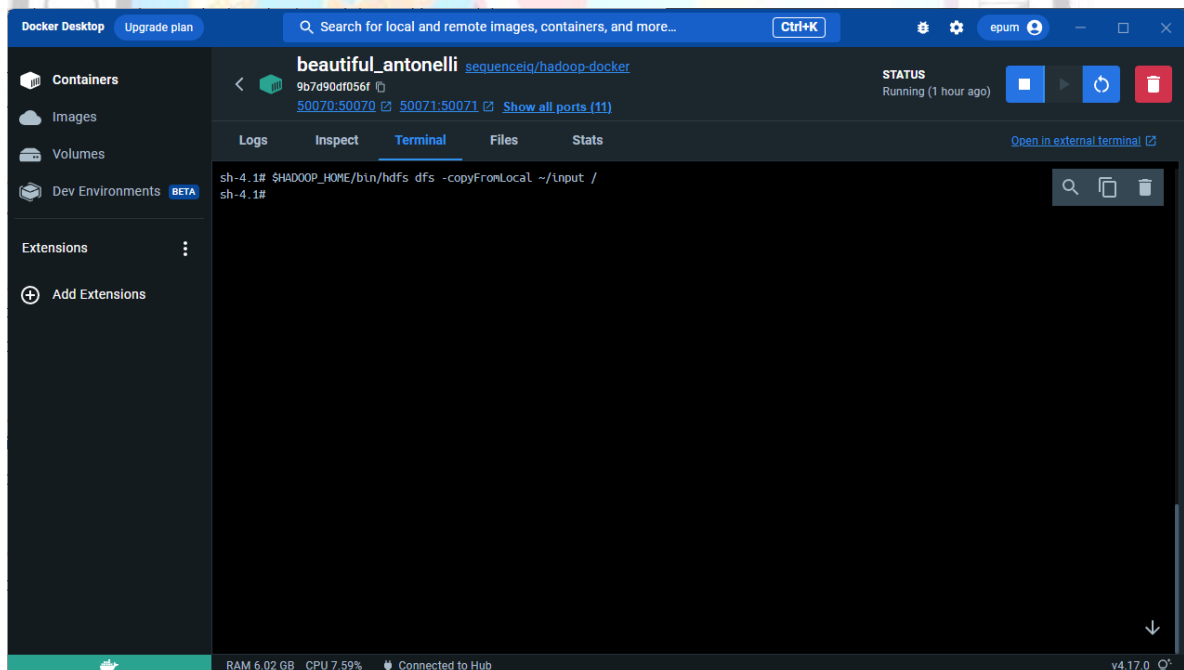
11. Copiamos los archivos de entrada dentro de la carpeta “input” utilizando los siguientes comandos:



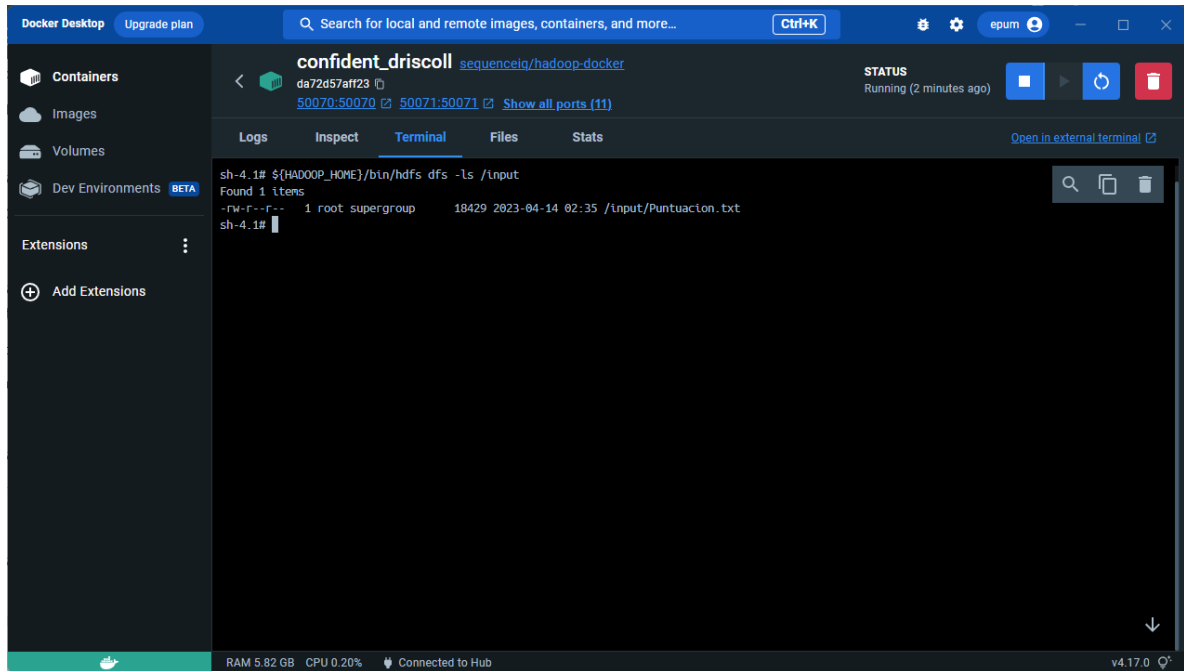
Verificamos que correctamente se hayan copiado los archivos:



12. Ahora vamos a copiar los archivos de entrada en el sistema de archivos de Hadoop utilizando el siguiente comando:

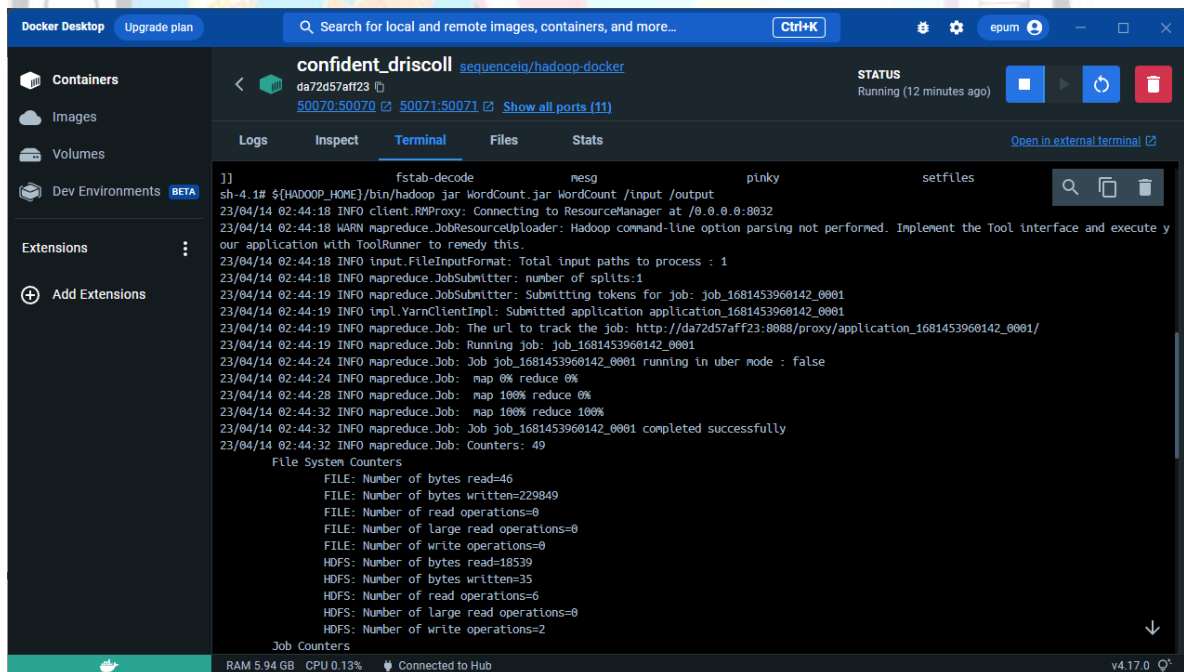


13. Verificamos que el comando anterior funcionó correctamente utilizando el siguiente comando en la terminal del contenedor:



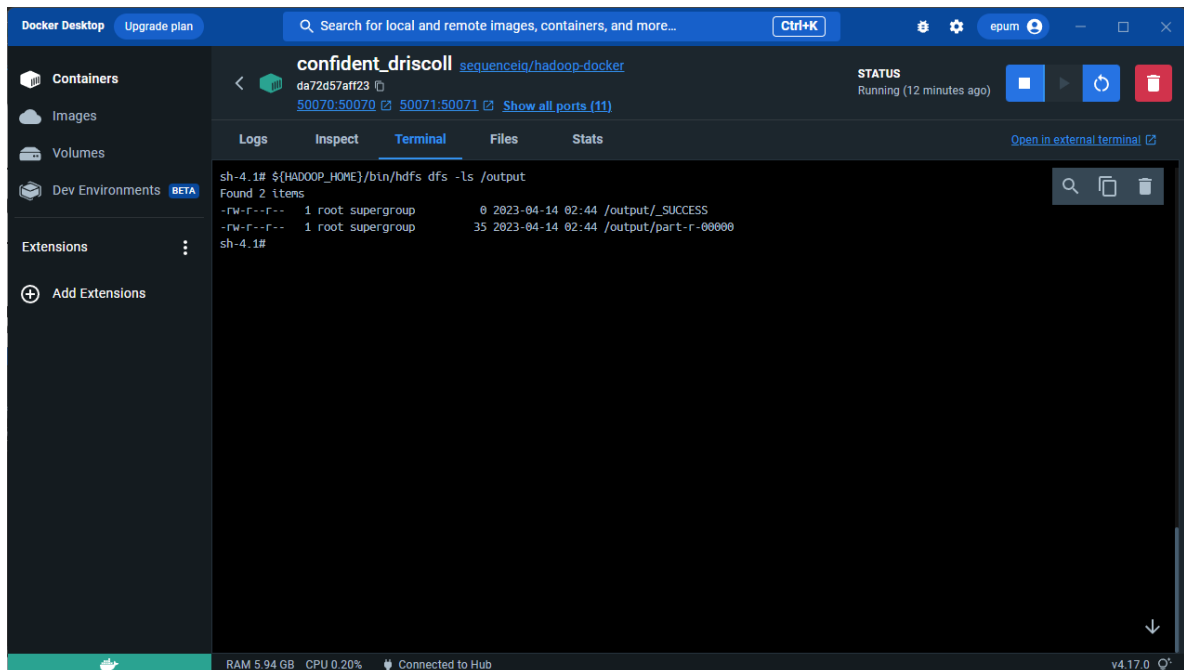
```
sh-4.1# ${HADOOP_HOME}/bin/hdfs dfs -ls /input
Found 1 items
-rw-r--r-- 1 root supergroup 18429 2023-04-14 02:35 /input/Puntuacion.txt
sh-4.1#
```

14. Ahora toca hacer el conteo de palabras utilizando el siguiente comando:



```
sh-4.1# ${HADOOP_HOME}/bin/hadoop jar WordCount.jar WordCount /input /output
23/04/14 02:44:18 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
23/04/14 02:44:18 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute y
our application with ToolRunner to remedy this.
23/04/14 02:44:18 INFO input.FileInputFormat: Total input paths to process : 1
23/04/14 02:44:18 INFO mapreduce.JobSubmitter: number of splits:1
23/04/14 02:44:19 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1681453960142_0001
23/04/14 02:44:19 INFO impl.YarnClientImpl: Submitted application application_1681453960142_0001
23/04/14 02:44:19 INFO mapreduce.Job: The url to track the job: http://da72d57aff23:8088/proxy/application_1681453960142_0001/
23/04/14 02:44:19 INFO mapreduce.Job: Running job: job_1681453960142_0001
23/04/14 02:44:24 INFO mapreduce.Job: Job job_1681453960142_0001 running in uber mode : false
23/04/14 02:44:24 INFO mapreduce.Job: map 0% reduce 0%
23/04/14 02:44:28 INFO mapreduce.Job: map 100% reduce 0%
23/04/14 02:44:32 INFO mapreduce.Job: map 100% reduce 100%
23/04/14 02:44:32 INFO mapreduce.Job: Job job_1681453960142_0001 completed successfully
23/04/14 02:44:32 INFO mapreduce.Job: Counters: 49
File System Counters
  FILE: Number of bytes read=46
  FILE: Number of bytes written=229849
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=18539
  HDFS: Number of bytes written=35
  HDFS: Number of read operations=6
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
Job Counters
```

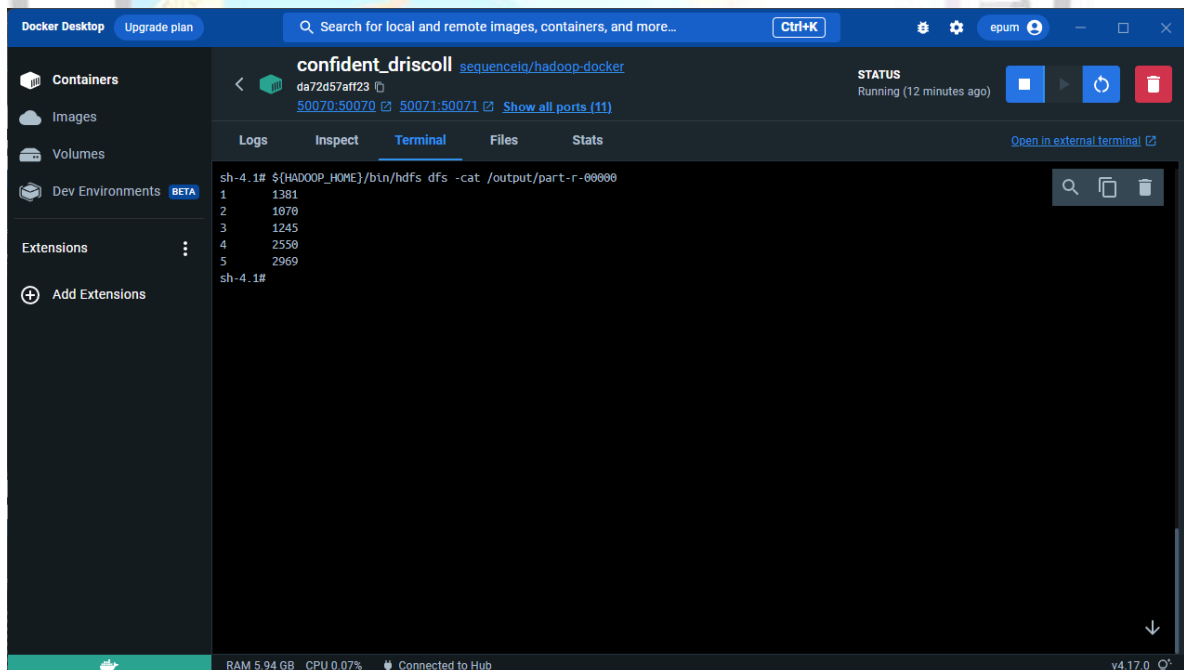
15. Para verificar que el comando anterior funcionó, ingresamos el siguiente comando:



The screenshot shows the Docker Desktop interface with a terminal window open for a container named 'confident_driscoll'. The terminal displays the output of the command 'ls /output', showing two files: 'SUCCESS' and 'part-r-00000'. The container is running and connected to the Hub.

```
sh-4.1# ${HADOOP_HOME}/bin/hdfs dfs -ls /output
Found 2 items
-rw-r--r-- 1 root supergroup 0 2023-04-14 02:44 /output/_SUCCESS
-rw-r--r-- 1 root supergroup 35 2023-04-14 02:44 /output/part-r-00000
sh-4.1#
```

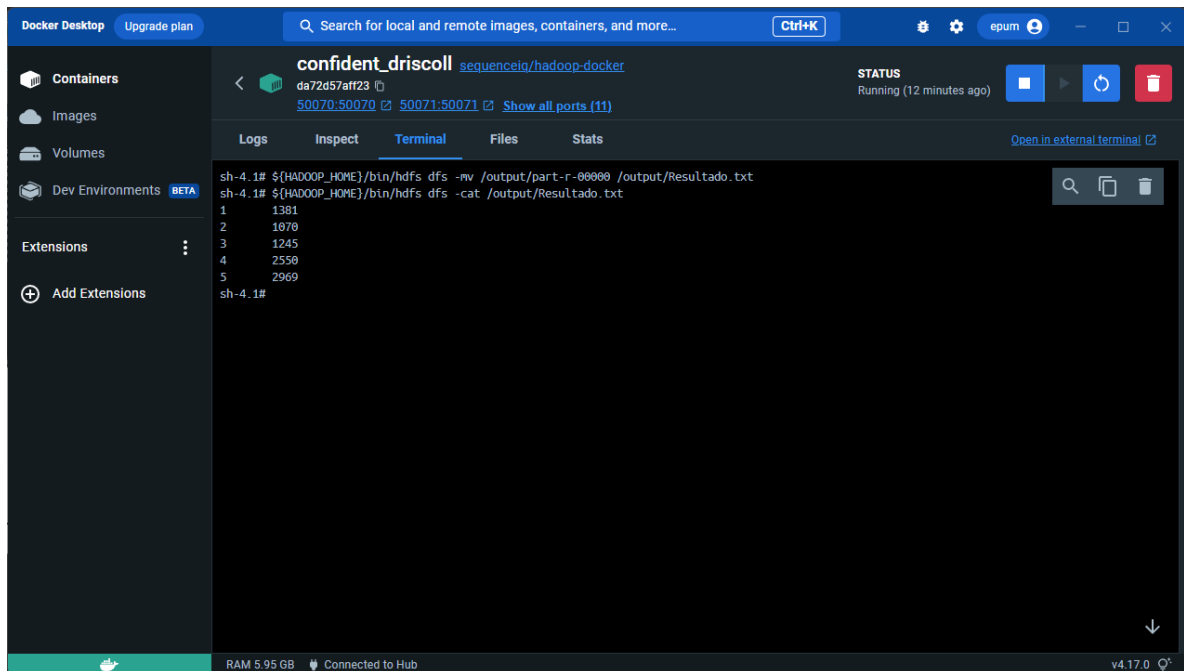
16. Para ver el resultado del conteo del archivo “Puntuacion.txt” que es con el que se está probando de primero, ingresamos el siguiente comando:



The screenshot shows the Docker Desktop interface with a terminal window open for the same container. The terminal displays the output of the command 'cat /output/part-r-00000', showing a list of numbers: 1381, 1070, 1245, 2550, and 2969. The container is running and connected to the Hub.

```
sh-4.1# ${HADOOP_HOME}/bin/hdfs dfs -cat /output/part-r-00000
1 1381
2 1070
3 1245
4 2550
5 2969
sh-4.1#
```

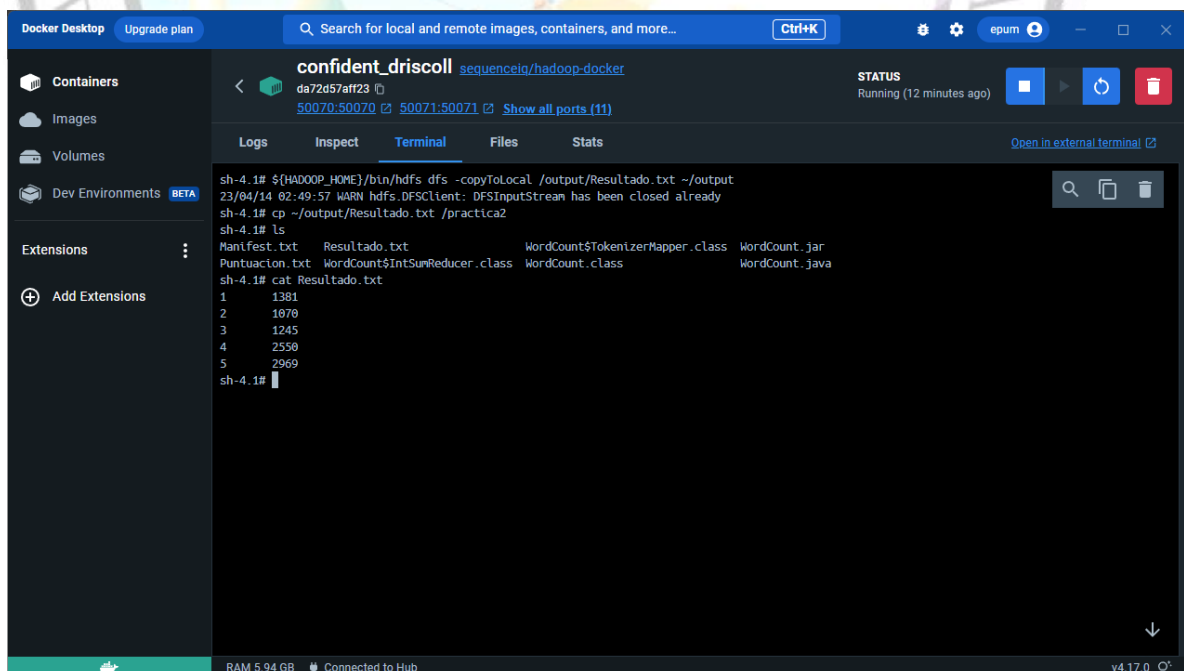
17. Ahora lo que vamos a hacer es renombrar el archivo de salida a “Resultado.txt” y además, vamos a ver que la información sea la correcta, para eso, ingresamos los siguientes comandos:



The screenshot shows the Docker Desktop application. On the left sidebar, there are options for Containers, Images, Volumes, Dev Environments (marked BETA), and Extensions. The main panel displays the details of a container named 'confident_driscoll' (image: sequenceiq/hadoop-docker). The 'Terminal' tab is active, showing a shell prompt 'sh-4.1#'. The terminal output shows the execution of 'hdfs dfs -mv /output/part-r-00000 /output/Resultado.txt' and 'hdfs dfs -cat /output/Resultado.txt', which results in a list of five numbers: 1381, 1070, 1245, 2550, and 2969. The status bar at the bottom indicates 'RAM 5.95 GB' and 'Connected to Hub'.

```
sh-4.1# ${HADOOP_HOME}/bin/hdfs dfs -mv /output/part-r-00000 /output/Resultado.txt
sh-4.1# ${HADOOP_HOME}/bin/hdfs dfs -cat /output/Resultado.txt
1      1381
2      1070
3      1245
4      2550
5      2969
sh-4.1#
```

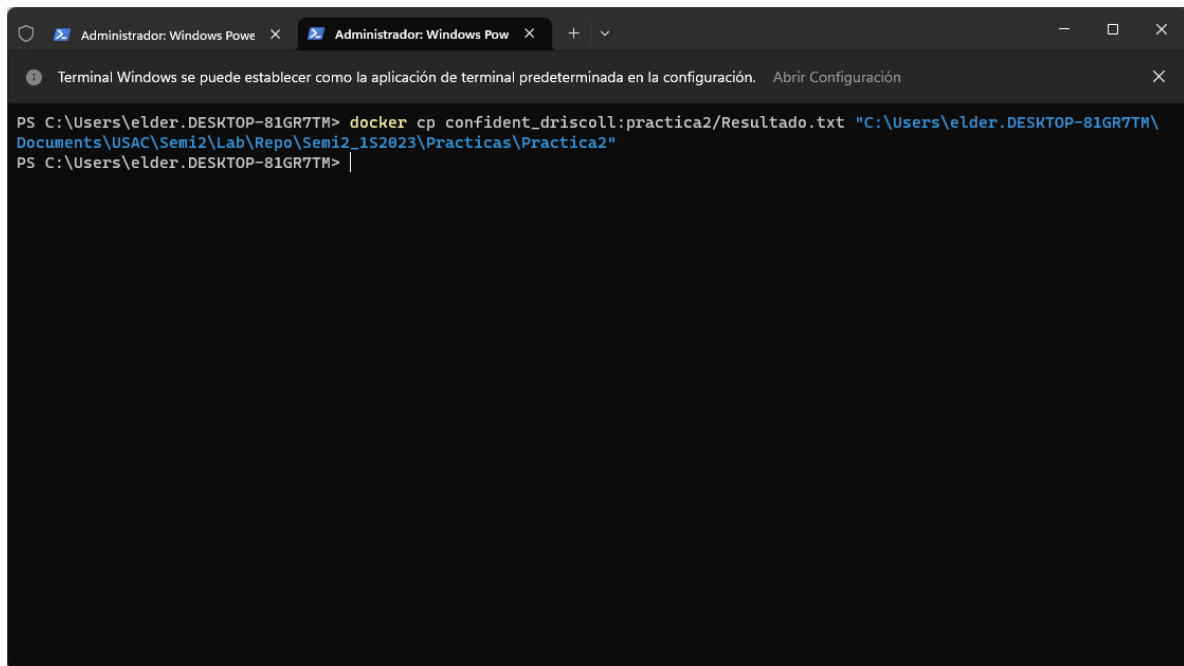
18. Ahora, vamos a copiar el archivo “Resultado.txt” del sistema de archivos de Hadoop a la carpeta que creamos al inicio llamada “practica2” utilizando los siguientes comandos:



The screenshot shows the Docker Desktop application with the same container 'confident_driscoll'. The 'Terminal' tab is active, showing the execution of 'hdfs dfs -copyToLocal /output/Resultado.txt ~/output', which produces a warning: 'WARN hdfs.DFSClient: DFSInputStream has been closed already'. This is followed by 'cp ~/output/Resultado.txt /practica2'. Then, 'ls' is run, showing a directory listing of files including 'Manifest.txt', 'Resultado.txt', 'Puntuacion.txt', 'WordCount\$IntSumReducer.class', 'WordCount.class', 'WordCount\$TokenizerMapper.class', 'WordCount.jar', and 'WordCount.java'. Finally, 'cat Resultado.txt' is executed, displaying the same five numbers as in the previous screenshot. The status bar at the bottom indicates 'RAM 5.94 GB' and 'Connected to Hub'.

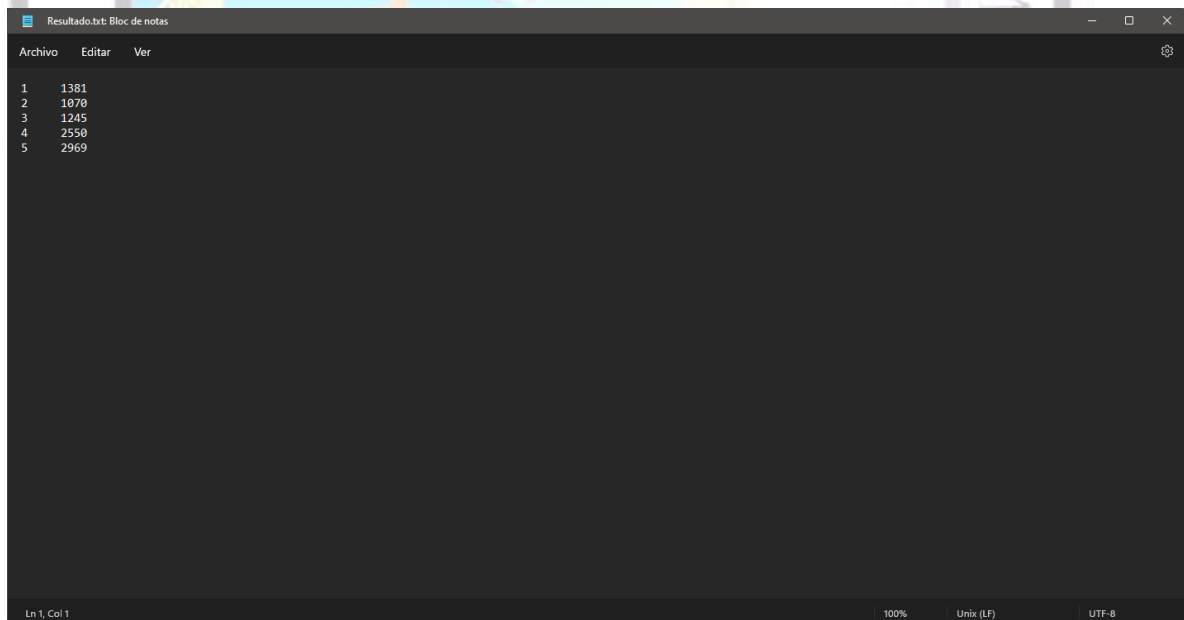
```
sh-4.1# ${HADOOP_HOME}/bin/hdfs dfs -copyToLocal /output/Resultado.txt ~/output
23/04/14 02:49:57 WARN hdfs.DFSClient: DFSInputStream has been closed already
sh-4.1# cp ~/output/Resultado.txt /practica2
sh-4.1# ls
Manifest.txt      Resultado.txt      WordCount$TokenizerMapper.class  WordCount.jar
Puntuacion.txt   WordCount$IntSumReducer.class  WordCount.class                  WordCount.java
sh-4.1# cat Resultado.txt
1      1381
2      1070
3      1245
4      2550
5      2969
sh-4.1#
```

19. Vamos a extraer el archivo “Resultado.txt” del contenedor y lo vamos a copiar a nuestra computadora utilizando la terminal de Windows/Linux, para esto, vamos a ingresar el siguiente comando:



```
PS C:\Users\elder.DESKTOP-81GR7TM> docker cp confident_driscoll:practica2/Resultado.txt "C:\Users\elder.DESKTOP-81GR7TM\Documents\USAC\Semi2\Lab\Repo\Semi2_1S2023\Practicas\Practica2"
PS C:\Users\elder.DESKTOP-81GR7TM> |
```

20. Verificamos en la ruta donde guardamos el archivo del Resultado:



```
Resultado.txt: Bloc de notas
Archivo  Editar  Ver
1      1381
2      1070
3      1245
4      2550
5      2969
Ln 1, Col 1
100%  Unix (LF)  UTF-8
```

Cabe resaltar que para que todo esto funcione, se tiene que hacer 1 vez por archivo para que no se mezclen resultados y el WordCount.jar pueda funcionar correctamente y sin complicaciones, además de que se tiene que crear un nuevo contenedor por archivo a revisar.

Resultados de Puntuacion.txt

Browse Directory

localhost:50070/explore.html/

Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	14/4/2023, 0:35:21	0	0 B	input
drwxr-xr-x	root	supergroup	0 B	14/4/2023, 0:48:49	0	0 B	output
drwx-----	root	supergroup	0 B	14/4/2023, 0:44:18	0	0 B	tmp
drwxr-xr-x	root	supergroup	0 B	22/7/2015, 9:17:26	0	0 B	user

Hadoop, 2014.

Browse Directory

localhost:50070/explore.html/#input

Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	18 KB	14/4/2023, 0:35:21	1	128 MB	Puntuacion.txt

Hadoop, 2014.

Docker Desktop

Upgrade plan

Search for local and remote images, containers, and more...

Ctrl+K

epum

Containers

Images

Volumes

Dev Environments BETA

Extensions

Add Extensions

confident_driscoll sequenceio/hadoop-docker

da72d57aff23

50070:50070

50071:50071

Show all ports (11)

STATUS

Running (12 minutes ago)

Logs

Inspect

Terminal

Files

Stats

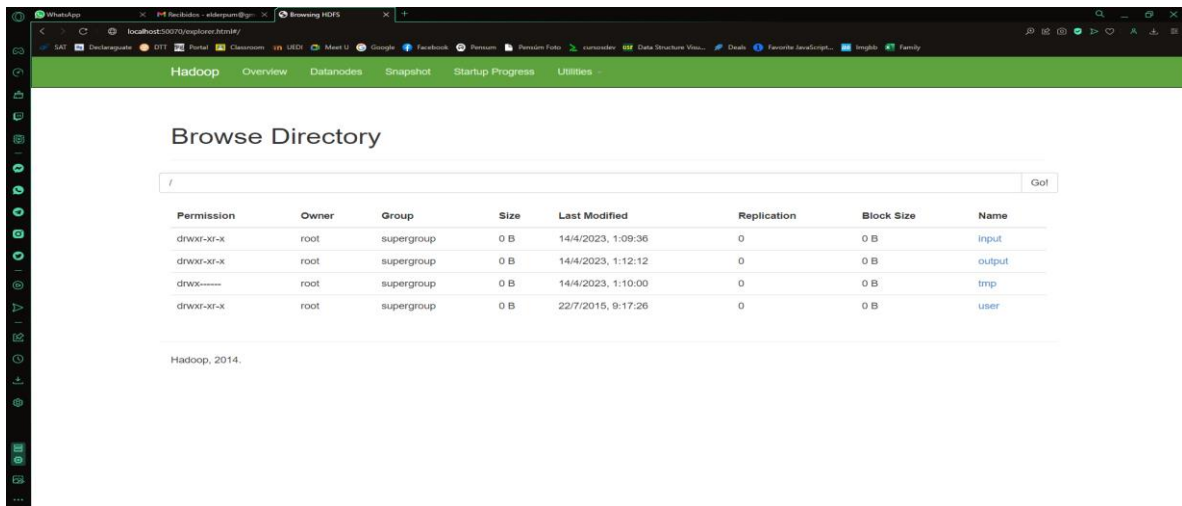
Open in external terminal

```
sh-4.1# ${HADOOP_HOME}/bin/hdfs dfs -cat /output/part-r-00000
1 1381
2 1870
3 1245
4 2558
5 2969
sh-4.1#
```

RAM 5.94 GB CPU 0.07% Connected to Hub

v4.17.0

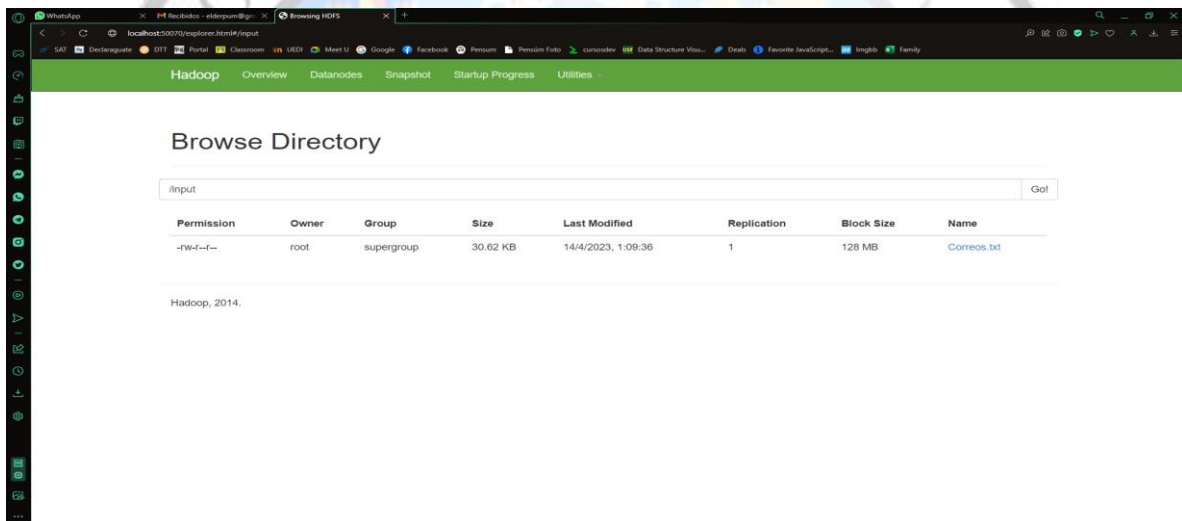
Resultados de Correos.txt



The screenshot shows the Hadoop File Explorer interface. The breadcrumb path is `localhost:50070/`. The search bar contains `/`. The table below lists the root directory contents:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	14/4/2023, 1:09:36	0	0 B	input
drwxr-xr-x	root	supergroup	0 B	14/4/2023, 1:12:12	0	0 B	output
drwxr-xr-x	root	supergroup	0 B	14/4/2023, 1:10:00	0	0 B	tmp
drwxr-xr-x	root	supergroup	0 B	22/7/2015, 9:17:26	0	0 B	user

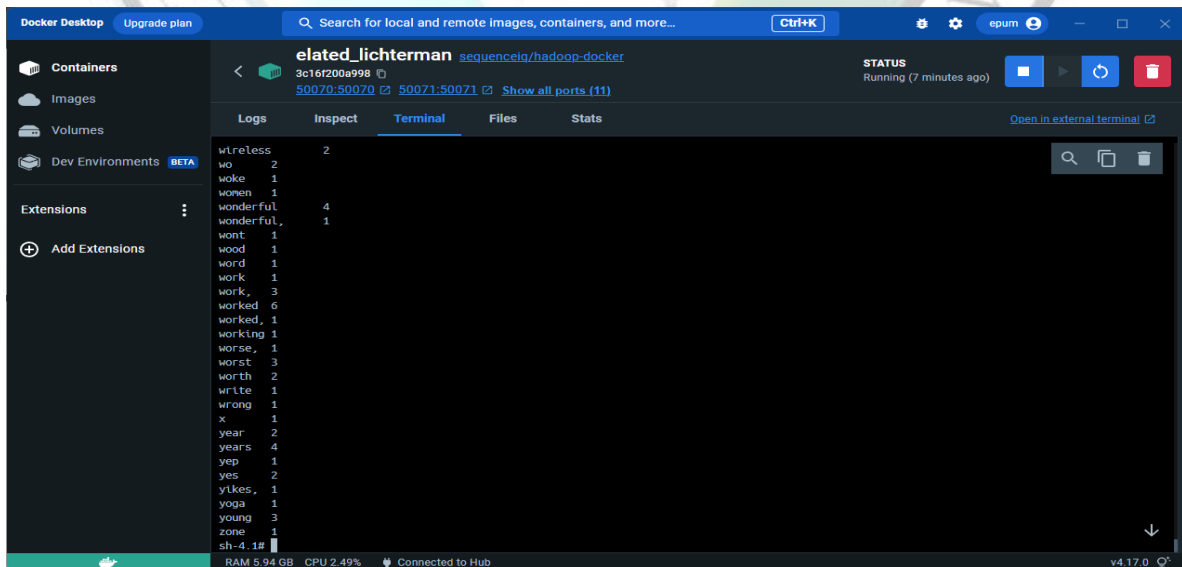
Hadoop, 2014.



The screenshot shows the Hadoop File Explorer interface with the breadcrumb path `localhost:50070/` and the search bar containing `/input`. The table below lists the contents of the `/input` directory:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	30.62 KB	14/4/2023, 1:09:36	1	128 MB	Correos.txt

Hadoop, 2014.



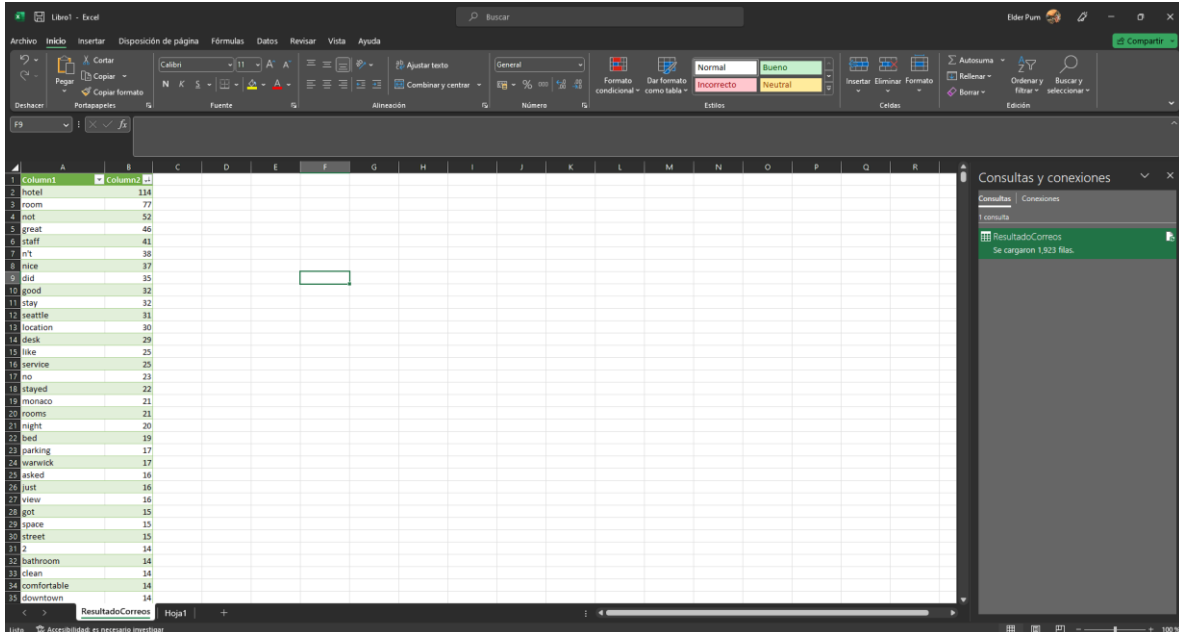
The screenshot shows the Docker Desktop interface. The container `elated_lichterman` (image `sequenceiq/hadoop-docker`) is running. The terminal shows the output of the `cat` command on `Correos.txt`:

```
wireless 2
wo 2
woke 1
women 1
wonderful 4
wonderful, 1
want 1
wood 1
word 1
work 1
work, 3
worked 6
worked, 1
working 1
worse, 1
worst 3
worth 2
write 1
wrong 1
x 1
year 2
years 4
yep 1
yes 2
yikes, 1
yoga 1
young 3
zone 1
sh-4.18
```

RAM 5.94 GB CPU 2.49% Connected to Hub v4.17.0

Resultados de los datos obtenidos del archivo Correos.txt

Después de obtener el archivo de texto en nuestra computadora, pasamos los datos a una tabla de datos en Excel y aplicándole un filtro de mayor a menor, nos queda los datos de la siguiente forma:



Column1	Column2
hotel	114
room	77
not	52
great	46
staff	41
n't	38
nice	37
did	35
good	32
stay	32
seattle	31
location	30
desk	29
like	25
service	25
no	23
stayed	22
monaco	21
rooms	21
night	20
bed	19
parking	17
warwick	17
asked	16
just	16
view	16
got	15
space	13
street	13
2	14
bathroom	14
clean	14
comfortable	14
downtown	14

Básicamente las palabras claves que se repiten son:

Hotel	114
Room	77
Not	52
Great	46
Staff	41
N't	38
Nice	37
Did	35
Good	32
Stay	32
Seattle	31
Location	30
Desk	29
Like	25
Service	25
Rooms	21

Si analizamos las palabras claves obtenidas del filtro de información, podemos deducir lo siguiente:

1. Principalmente que lo que más buscan las personas son hoteles. Se pueden deducir varias cosas, pero por lo general la gente lo que busca son hoteles fiables, con buen servicio en general y habitaciones cómodas para descansar después de un viaje agotador. Además, que el personal esté bien capacitado y sepa brindar un buen servicio.
2. Después podemos deducir que buscan buenas localizaciones para hospedarse, quizá buenas vistas desde el hotel.
3. Por último, las búsquedas indican que la gente busca servicios sanitarios dentro de los hoteles de óptimas condiciones, higiénicos y con una muy buena presentación.

Si ampliamos nuestras palabras claves usando Excel podemos saber que en este archivo toda la información trata sobre referencias de hoteles que cuentan con buena ubicación, parqueos óptimos, que el precio en la noche sea bueno, así como también habitaciones cómodas, aunque de precios elevados estos sean de excelente calidad, así como también el personal sea capacitado y por último, que la atención del lugar sea del agrado del cliente.



Resultados de los datos obtenidos del archivo Puntuacion.txt

El archivo Puntuacion.txt contiene calificaciones de distintos usuarios que varían entre las puntuaciones 1-5, donde 1 es la puntuación más baja y 5 la más alta.

Teniendo eso en cuenta y habiendo analizado el archivo de Correos, podremos darnos cuenta de que algunos factores principales que los usuarios toman en cuenta para emitir su calificación son los siguientes:

- Precio
- Disponibilidad
- Servicio
- Ubicación
- Presentación

Obviamente hay usuarios que tienen otros factores a tomar en cuenta, pero sumando esos factores junto a los principales descritos anteriormente, tenemos los siguientes resultados obtenidos del análisis del archivo de Puntuación:

1	1381
2	1070
3	1245
4	2550
5	2969

Es cierto que el puntaje más alto es también el más otorgado por los clientes, pero hay que darse cuenta que entre el punteo 1-3 juntos forman más que el punteo 4 o 5 individualmente. ¿Eso qué significa? Que hay bastantes clientes que consideran que el servicio es malo y no llenan sus expectativas, cosa alarmante para el hotel.

¿Cómo se arregla esto? Invirtiendo más en los factores principales que se detallaron anteriormente. Por ejemplo, regular el precio para lo que ofrecen, ofrecer disponibilidad 24/7 para abarcar mayor clientela, así como también capacitar al personal del hotel para ofrecer un excelente servicio de manera general e invertir en la presentación tanto externa como interna del hotel.