# EJERCICIOS

### EJERCICIO 1

Utilizando una de las tareas de ejemplo de MapReduce (sudoku) resuelve el siguiente sudoku:

puzzle1.dta

8 5 ? 3 9 ? ? ? ?

? ? 2 ? ? ? ? ? ?

? ? 6 ? 1 ? ? ? 2

? ? 4 ? ? 3 ? 5 9

? ? 8 9 ? 1 4 ? ?

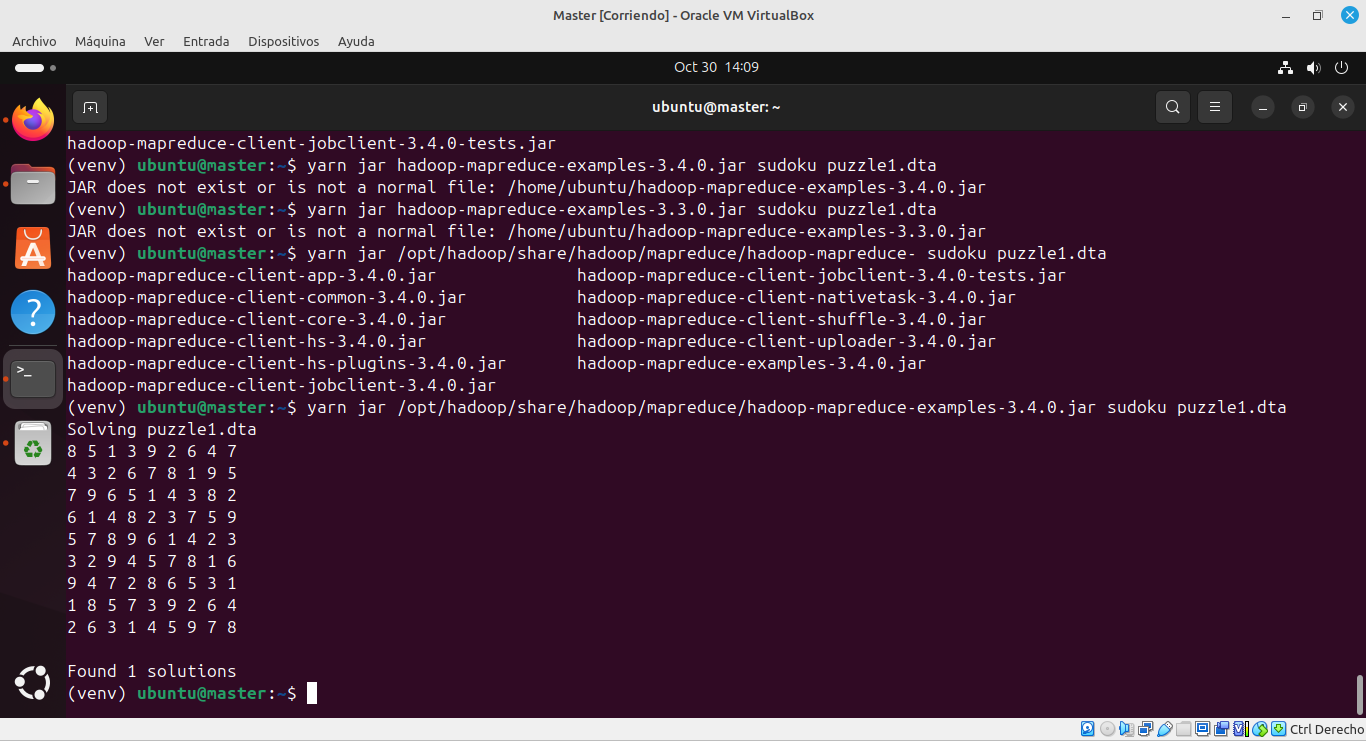
3 2 ? 4 ? ? 8 ? ?

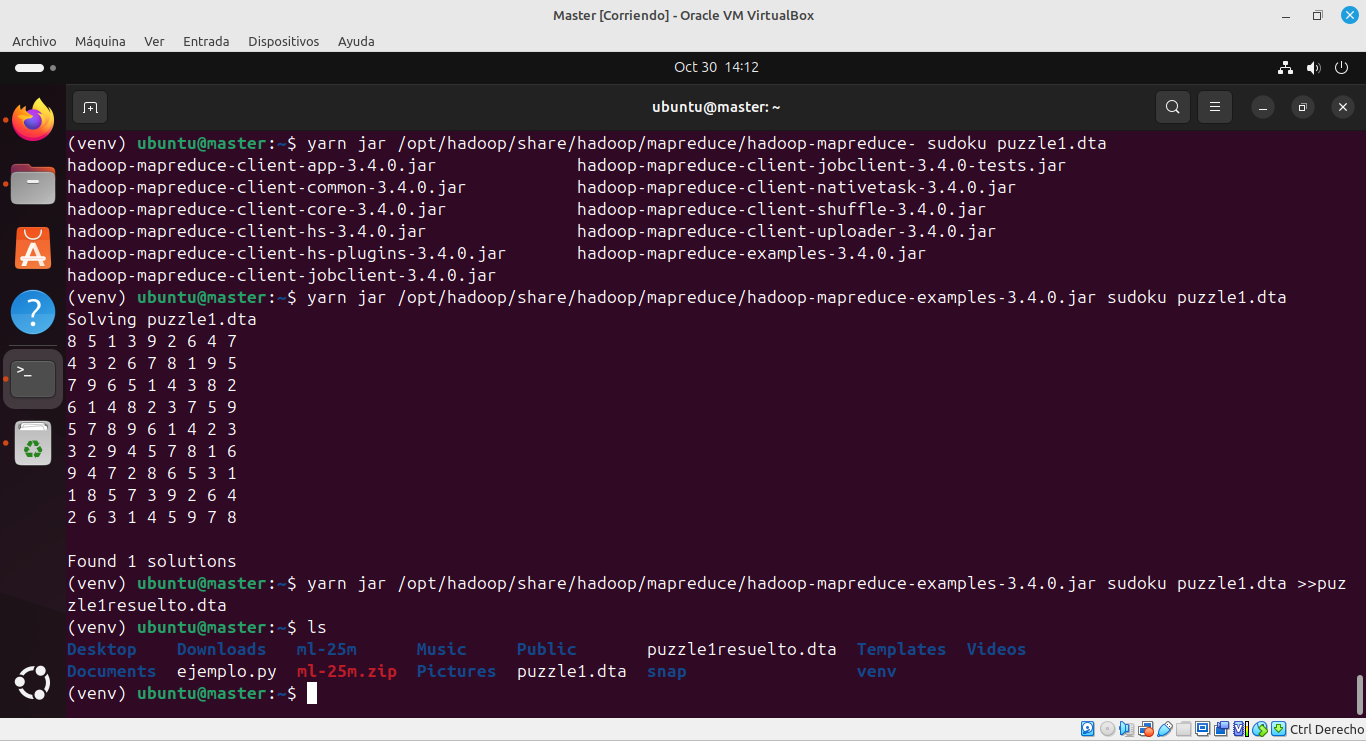
9 ? ? ? 8 ? 5 ? ?

? ? ? ? ? ? 2 ? ?

? ? ? ? 4 5 ? 7 8

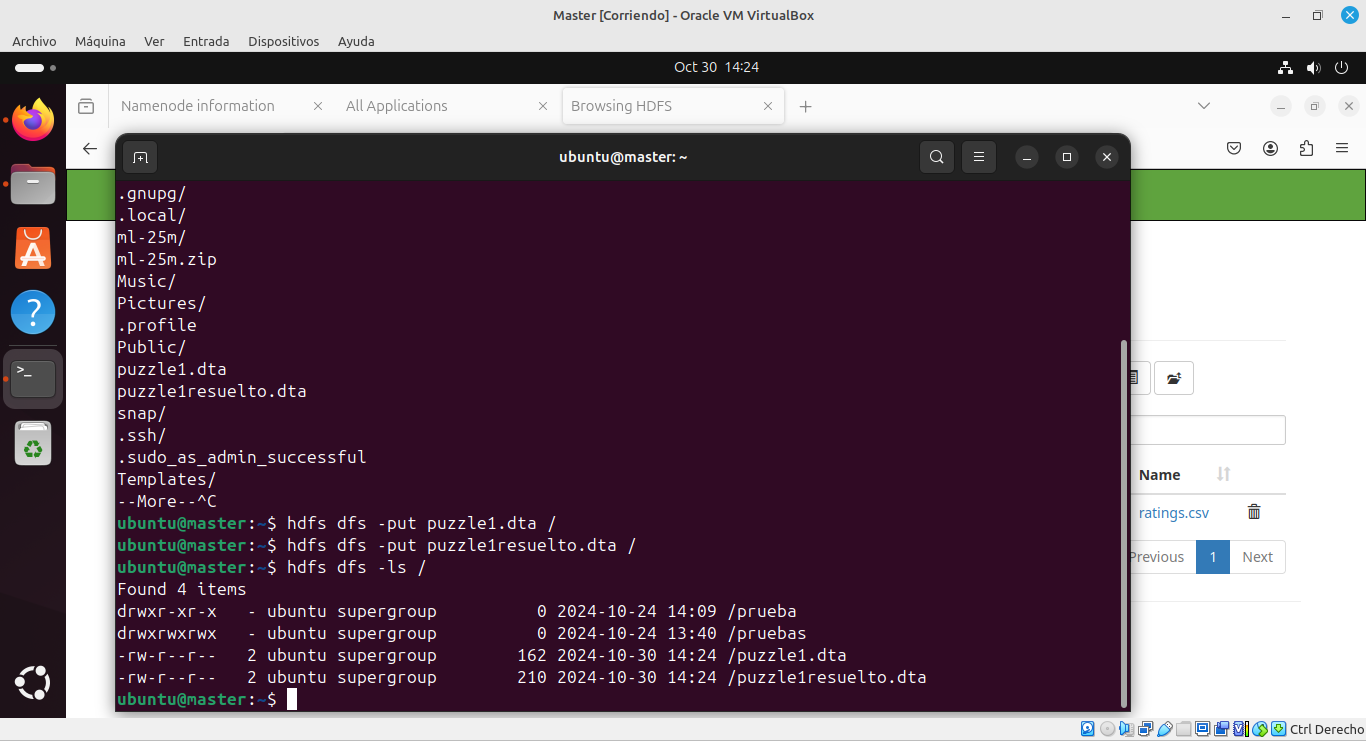
Primero muéstralo por pantalla y después vuélvelo a ejecutar, pero esta vez guardando la solución directamente en el fichero local de salida puzzle1resuelto.dta (recuerda las redirecciones en Linux).

yarn jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar sudoku puzzle1.dta





Para acabar sube el fichero inicial y la solución a hadoop.

hdfs dfs -put puzzle1.dta /

<https://learn.microsoft.com/es-es/azure/hdinsight/hadoop/apache-hadoop-run-samples-linux>

### EJERCICIO 2 (opcional- sólo si te sobra el tiempo)

Monta un clúster Hadoop sobre centOS sin entorno gráfico (instalación mínima con herramientas de desarrollo y herramientas de administración del sistema) usando 3 máquinas: 1 master y 2 trabajadores.

### EJERCICIO 3 (opcional sólo si has hecho el ejercicio 2)

Accede a las aplicaciones gráficas de hdfs y yarn del clúster anterior desde otro equipo que tenga entorno gráfico

### EJERCICIO 4

En el clúster anterior vamos a lanzar el programa de wordcount pero en Python.

* Crea el fichero pymap.py:

*#!/usr/bin/python3*

*import sys*

*for line in sys.stdin:*

*line = line.strip()*

*words = line.split()*

*for word in words:*

*print("{}\t1".format(word))*

* Crea el fichero pyreduce.py

*#!/usr/bin/python3*

*from operator import itemgetter*

*import sys*

*lastword = None*

*lastcount = 0*

*curword = None*

*i=1*

*for line in sys.stdin:*

*line = line.strip()*

*curword, count = line.split('\t', 1)*

*count = int(count)*

*#print("\*\*",curword,str(count),"\*\*")*

*if lastword == curword:*

*lastcount += count*

*else:*

*if lastword:*

*print("{}\t{}".format(lastword, lastcount))*

*lastcount = count*

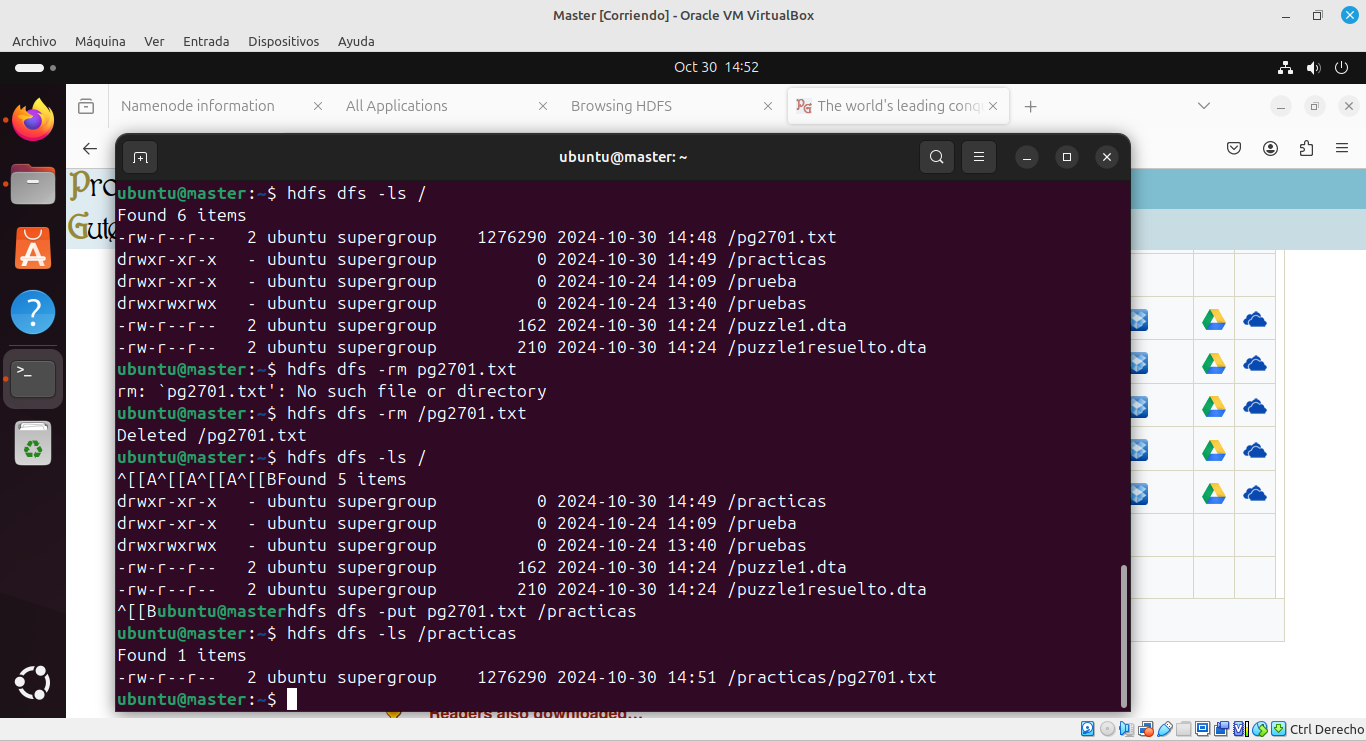
*lastword = curword*

*if lastword == curword:*

*print("{}\t{}".format(lastword, lastcount))*

* Descarga un libro del [proyecto Gutenberg](https://www.gutenberg.org/) y súbelo a hadoop al directorio /practicas

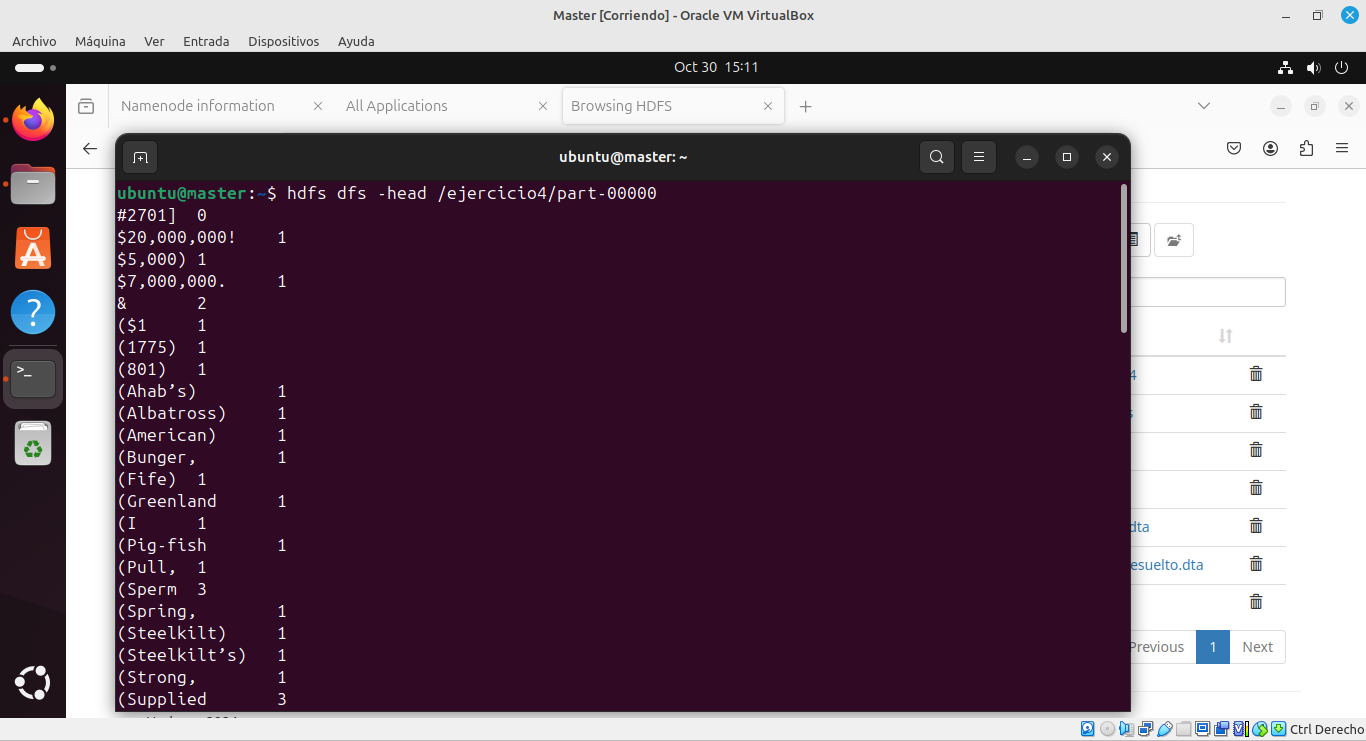
*wget https://www.gutenberg.org/cache/epub/2701/pg2701.txt*



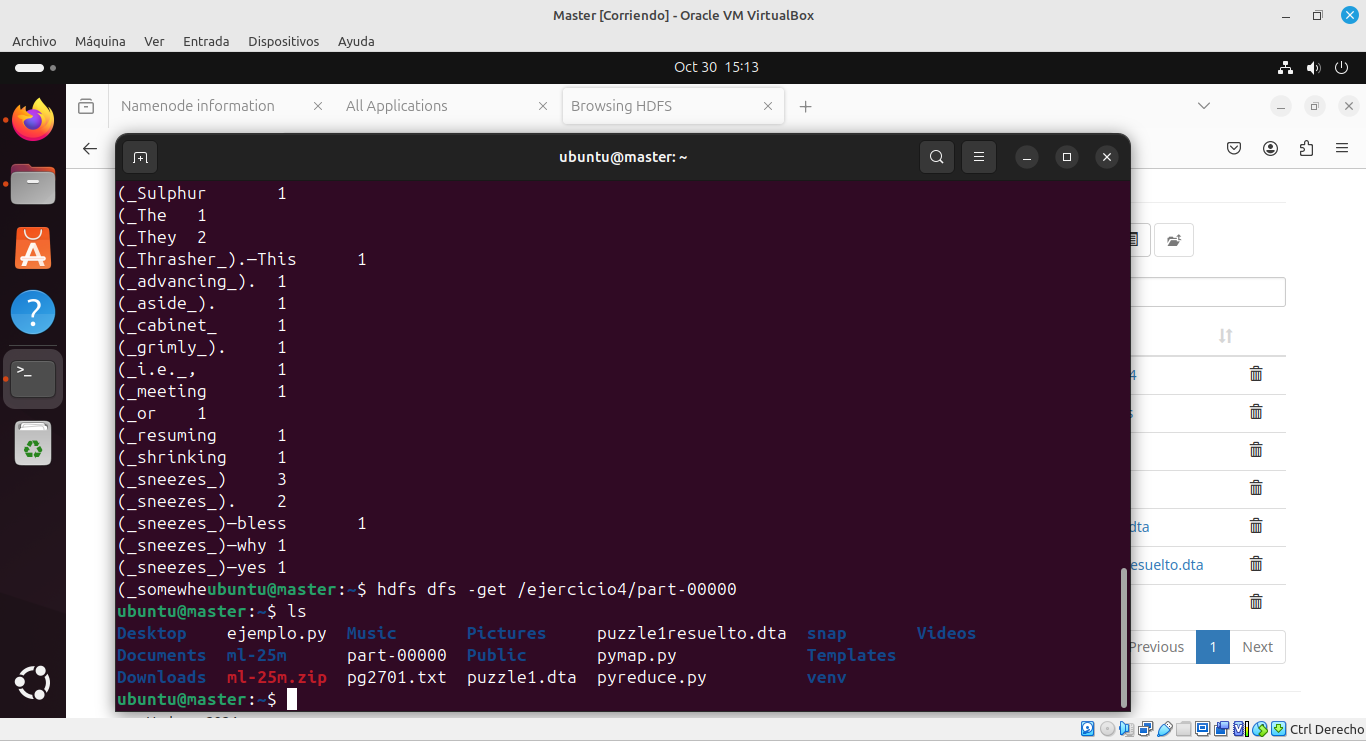
* Lanza el proceso y guarda la salida en el directorio /ejercicio4

*hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar -files pymap.py,pyreduce.py -mapper pymap.py -reducer pyreduce.py -input /practicas/pg2701.txt -output /ejercicio4*

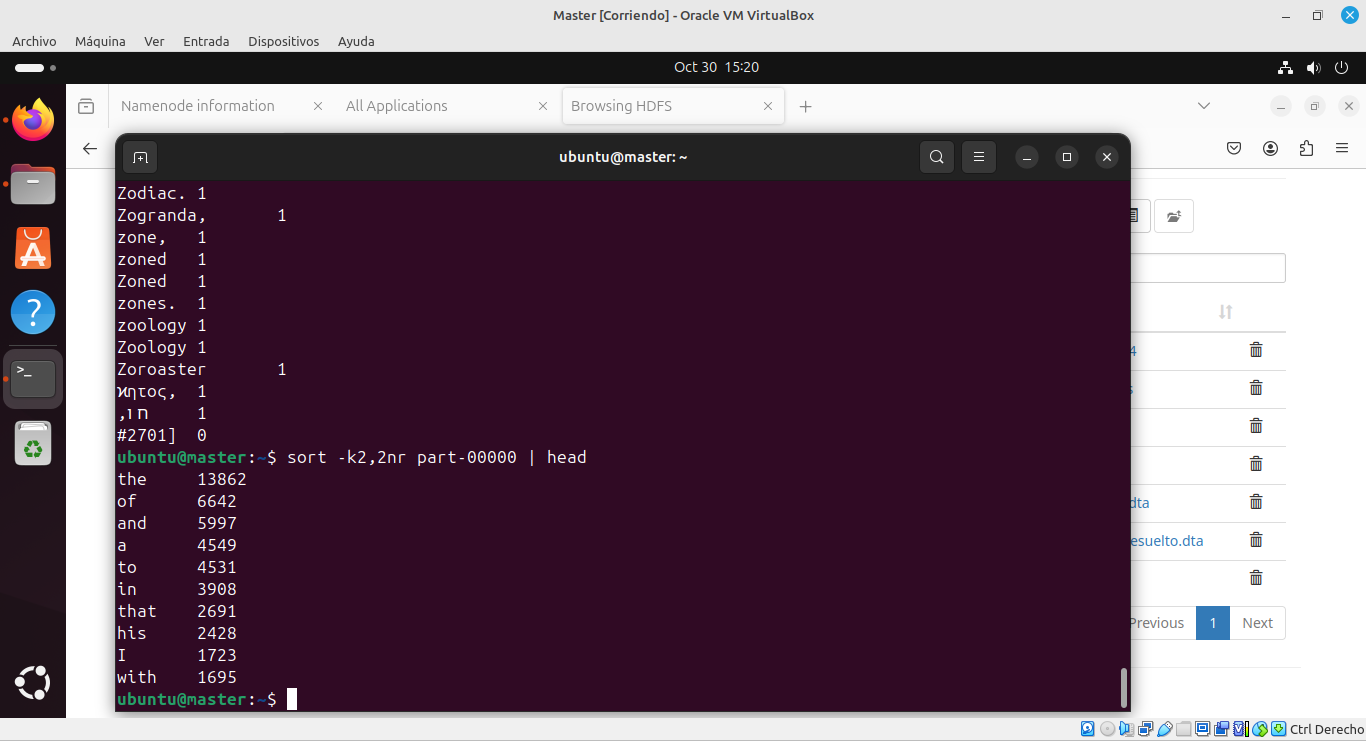
* Visualiza la salida



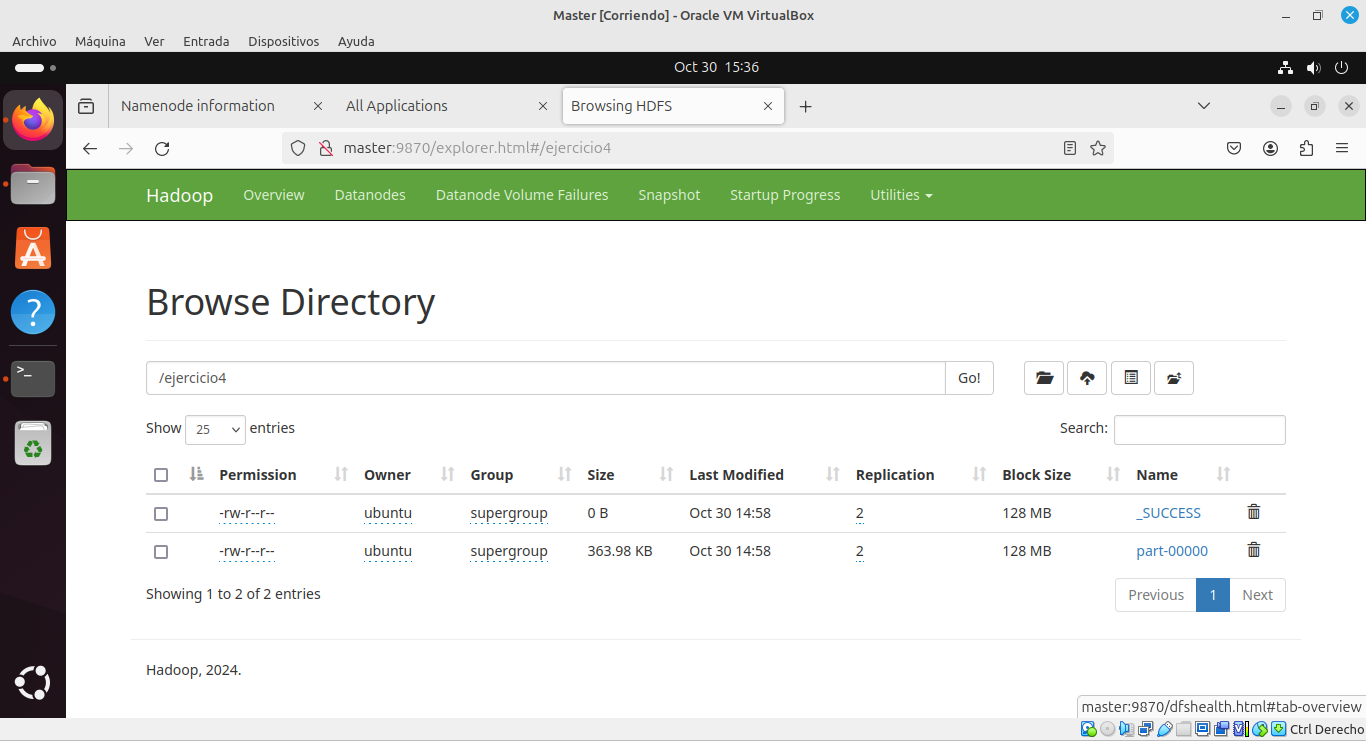
* Descarga a local el fichero de salida

hdfs dfs -get /ejercicio4/part-00000

* Muestra la salida ordenada de mayor a menor y de forma paginada según las palabras más frecuentes

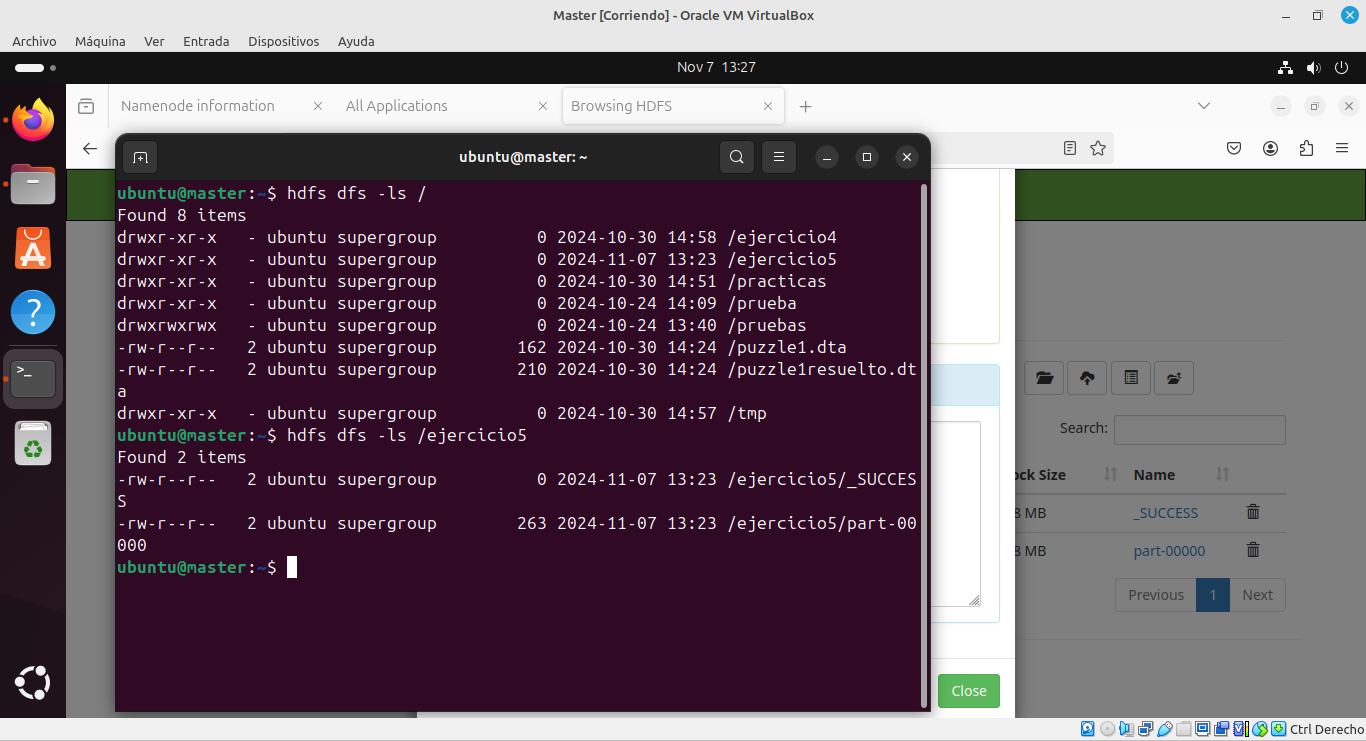


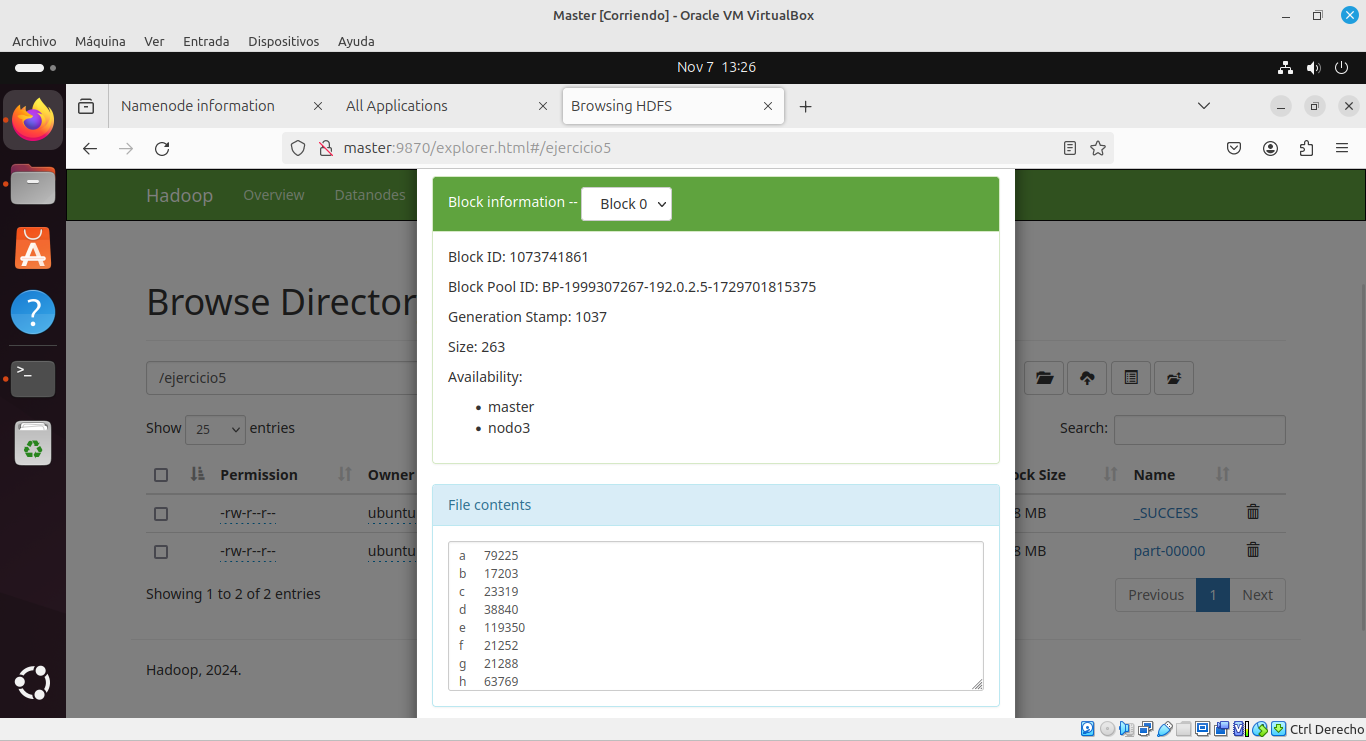
* Desde la aplicación web mira cómo ha ido la tarea



### EJERCICIO 5

Repite el ejercicio anterior pero ahora crea tus propias funciones map y reduce para contar el número de veces que aparece cada letra, pero ahora guarda la salida en /ejercicio5.

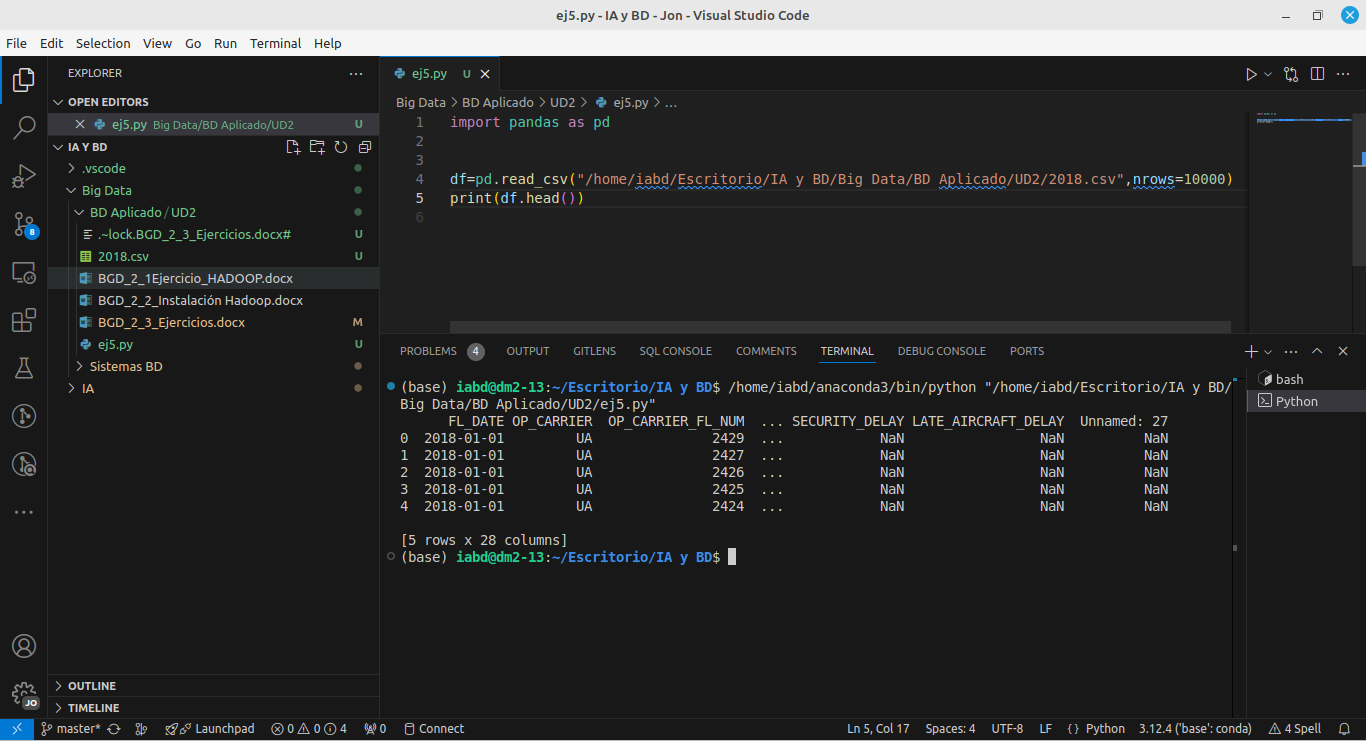




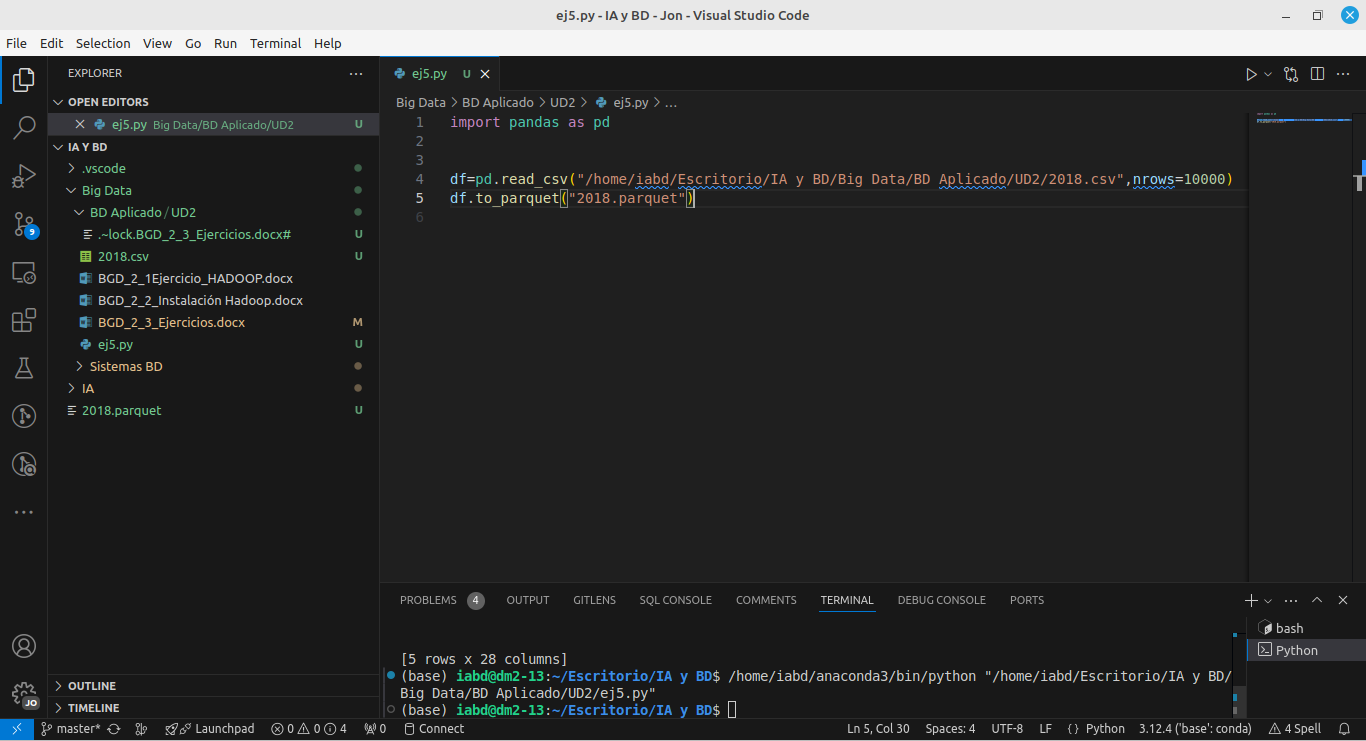
### EJERCICIO 6

Descarga los datos en formato csv correspondientes al año 2018 de la siguiente página [retrasos en los vuelos](https://www.kaggle.com/datasets/yuanyuwendymu/airline-delay-and-cancellation-data-2009-2018) y desde Python:

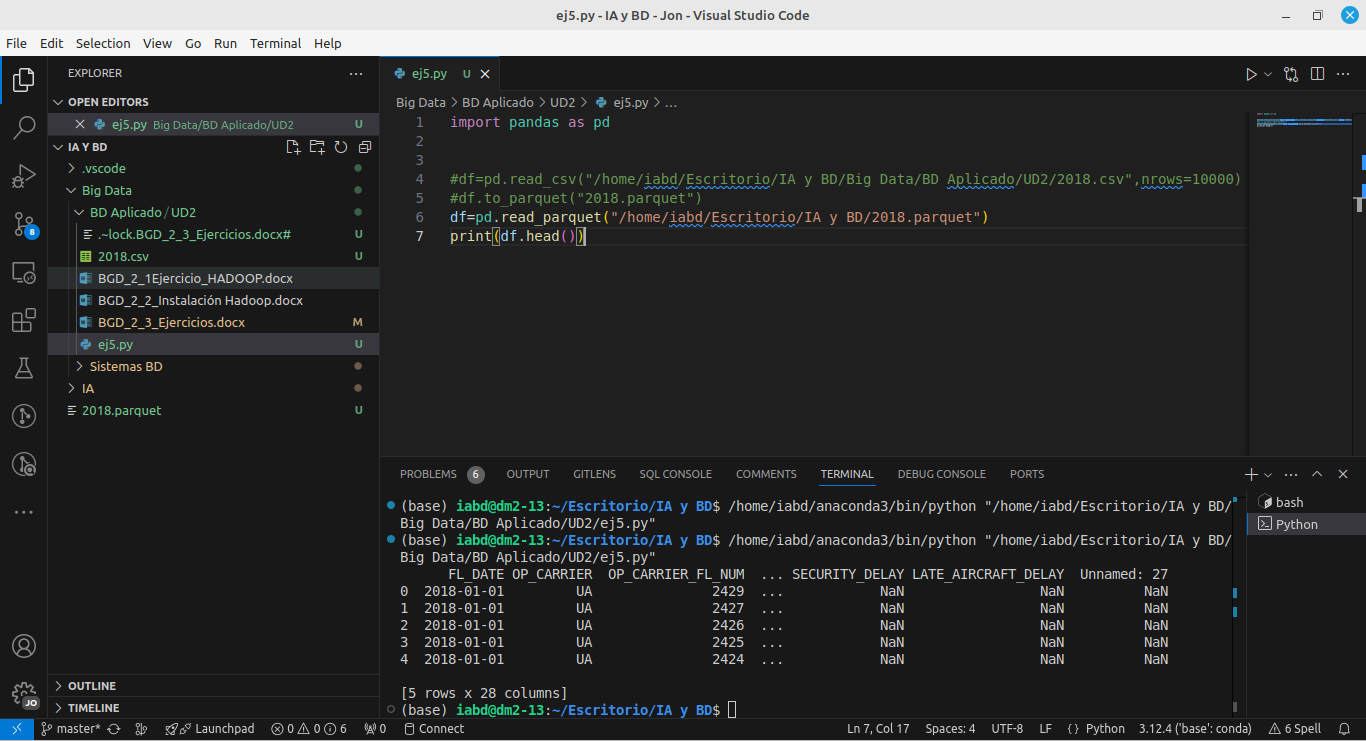
* Carga la información del fichero en un dataframe de pandas
* Muestra las primeras líneas del dataframe



* Guarda la información en un fichero parquet



* Carga la información del fichero parquet en un dataframe de pandas
* Muestra las primeras líneas del dataframe del punto anterior



* Muestra el tamaño del archivo parquet y el tiempo que se ha tardado en crear el archivo

