|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| logo berri CIFP  CIFP Ciudad Jardín LHII | | | **Curso de Especialización:** Inteligencia artificial y Big Data | | | |
| **Modulo:**  Sistemas de Big Data  Big Data Aplicado | | | **Cód. Modulo:**  SBIG  BIGD |
| **Curso:** IABD | | | |
| **1ª Evaluación** | | | **Fecha**: 11/12/2024 | | | |
| **Cód. Prof:** PTI8 | | **Profesora:** Saioa Uribeetxebarria | |  | | |
|  | | | | | | |
| **Datos del Alumno** | | | | | | |
| **Nº** | **Nombre:** | | | | **Firma** | |
| **Apellido 1:** | | | | |
| **Apellido 2:** | | | | |

**Calificación:**

Para realizar este examen se aconseja utilizar los documentos de teoría y ejercicios realizados durante el trimestre actual, que están en Moodle. Está prohibido utilizar conexión a internet durante la prueba. Este examen abarca los módulos BGD y SBGD y para aprobar se necesita:

|  |  |  |
| --- | --- | --- |
| **Módulo** | **Ejercicios** | **PUNTUACIÓN**  **mínima para aprobar** |
| **BIGD** | Hadoop: Ejercicio 1 | 5 |
| **SBIG** | Análisis de datos: Ejercicio: 2, 3  Almacenamiento de datos MongoDB: Ejercicios: 4, 5, 6 | 5 |
|  |  |  |

Todos los pasos del examen tienen que estar documentados en este documento mostrando el comando realizado y su salida.

## HADOOP

### EJERCICIO 1 - trabajando con hadoop

Con el clúster Hadoop proporcionado que consta de 1 nodo máster, 2 nodos workers y una conexión RED NAT, con las siguientes direcciones IP:

master IP:192.0.2.100

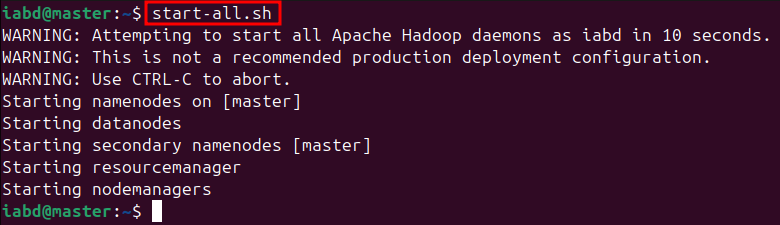
nodo1 IP:192.0.2.110

nodo2 IP:192.0.2.120

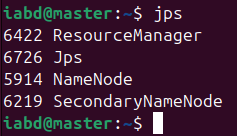
Lleva a cabo las siguientes acciones:

1.- Levanta los servicios, verifica que los servicios están en marcha tanto en el master como en los workers y verifica el acceso vía web a HDFS y YARN.

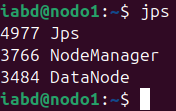
Start-all.sh



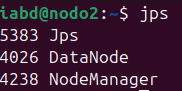
jps (Master)



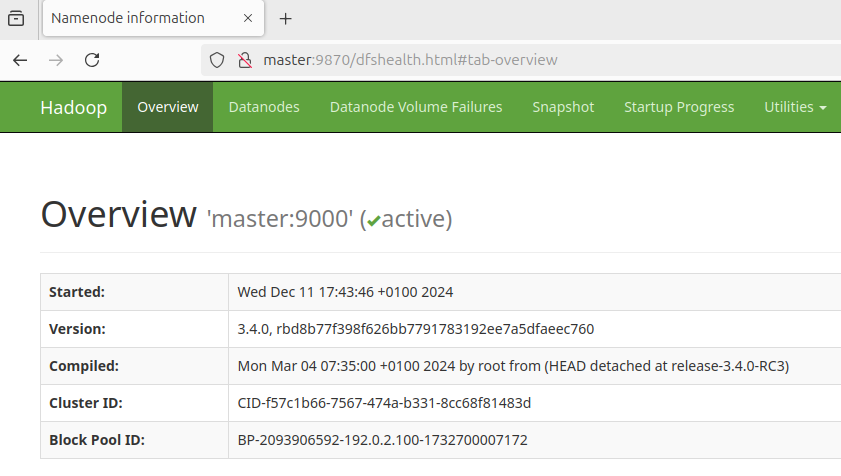
jps (nodo1)



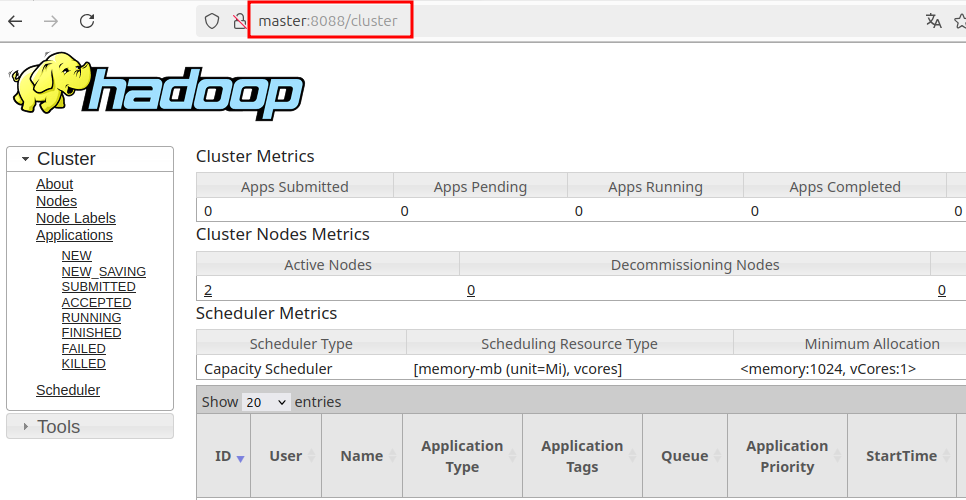
jps (nodo2)



https://master:9870



http://master:8088



2.- En el sistema HDFS crea una carpeta que se llame “Examen” y sube un archivo. (usar fichero “Subtitulos\_GameOfThrone.json” proporcionado con el enunciado del examen)

hdfs dfs -mkdir -p examen



Nos dirijimos a la carpeta donde esta el archivo en nuestro caso en descargas

cd Descargas/Examen Trimestre 1 Recursos-20241211/Ejercicio1

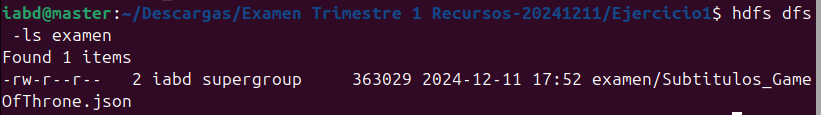


Subimos el archivo  
hdfs dfs -put Subtitulos\_GameOfThrone.json examen



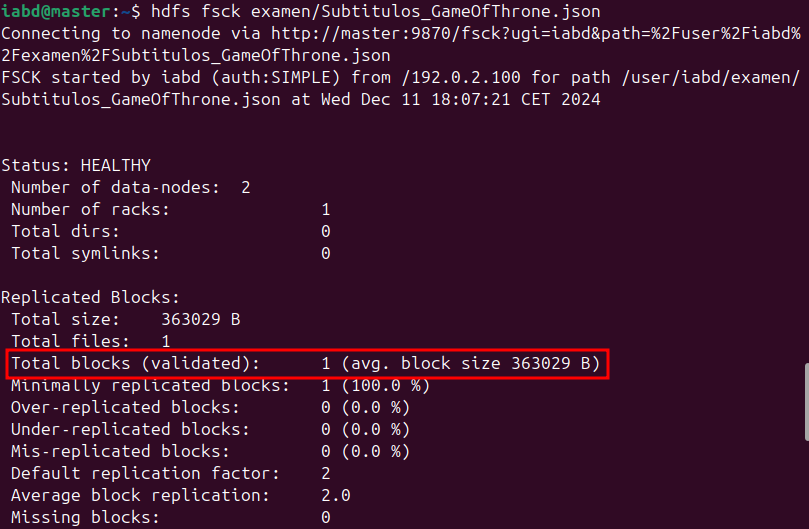
Comprobamos que se ha subido correctamente

hdfs dfs -ls examen



3.- ¿Cuántos bloques ocupa el archivo: “Subtitulos\_GameOfThrone.json” en el sistema de archivos HDFS)?

hdfs fsck examen/Subtitulos\_GameOfThrone.json

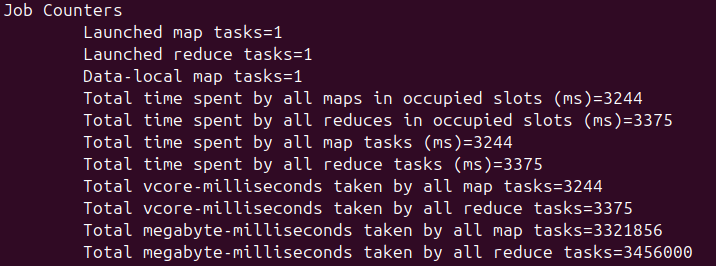


4.- Encarga un trabajo wordcount utilizando el ejemplo de map-reduce wordcount que tiene Hadoop, para el archivo: “Subtitulos\_GameOfThrone.json”

hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar wordcount examen/Subtitulos\_GameOfThrone.json /examen/wordcount

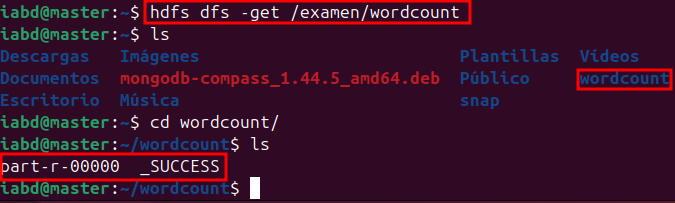


5.- ¿Cuánto tiempo le ha llevado realizar la tarea del punto 4?

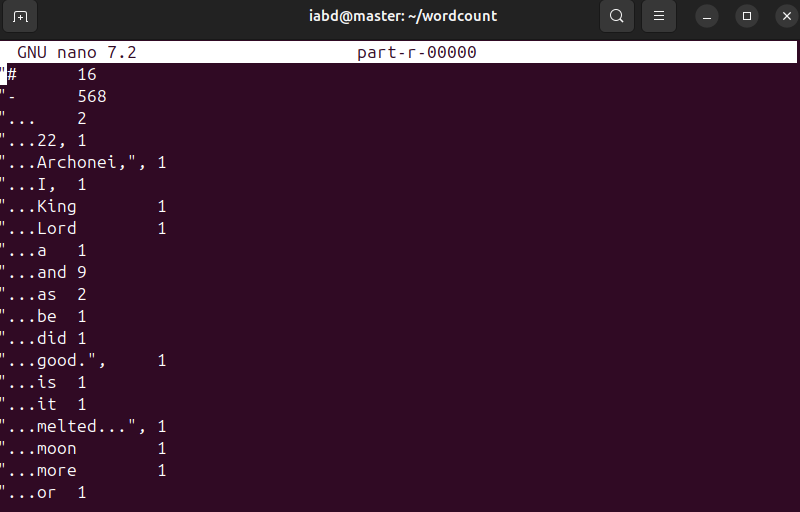


6.- Descarga el archivo con el resultado al sistema de archivos local y muestra su contenido.

hdfs dfs -get /examen/wordcount



Abrimos el archivo sudo nano part-r-00000



7.- Lleva a cabo el wordCount del mismo archivo que en el punto 4 pero aportando tus propios archivos mapper y reducer.

Creamos los archivos pymap

*#!/usr/bin/python3*

*import sys*

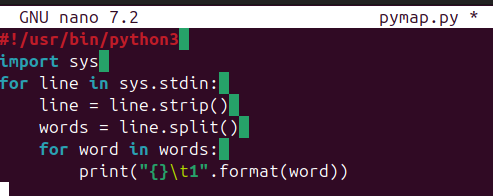
*for line in sys.stdin:*

*line = line.strip()*

*words = line.split()*

*for word in words:*

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



Creamos el archivo 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:*

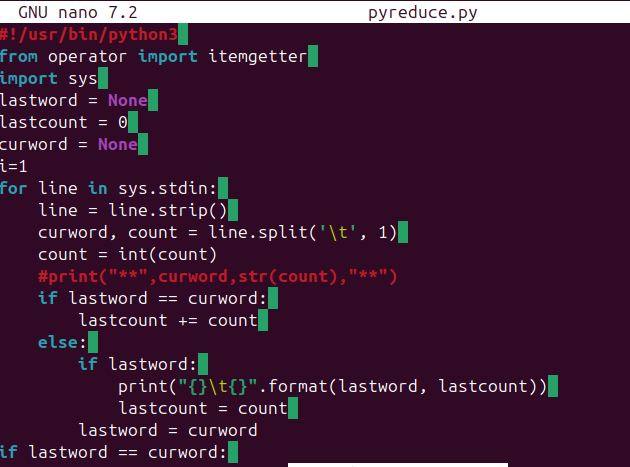
*print("{}\t{}".format(lastword, lastcount))*

*lastcount = count*

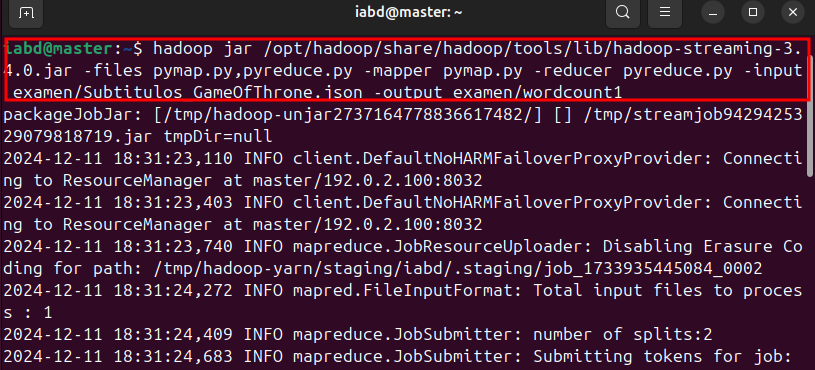
*lastword = curword*

*if lastword == curword:*

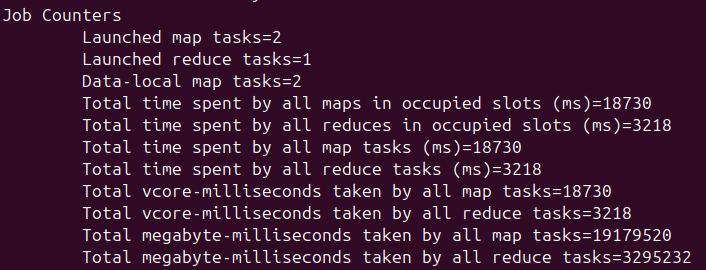
*print("{}\t{}".format(lastword, lastcount))*



*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 examen/Subtitulos\_GameOfThrone.json -output examen/wordcount1*



8.- ¿Cuánto tiempo le ha llevado realizar la tarea?



¿Es esta forma de ejecutar wordcount más lenta o más rápida que si ejecutamos wordcount con uno de los ejemplos que trae Hadoop?

*Es mas lenta que si la ejecutamos con los ejemplos que trae Hadoop.*

9.- Vamos a trabajar con snapShots. Para ello llevamos a cabo los siguientes pasos:

9.1.- Habilitar los snapShots sobre la carpeta “Examen” y crea el snapShot1 de la carpeta “Examen”

*Habilitamos los snapshot*

*hdfs dfsadmin -allowSnapshot /examen*

**

*Creamos el snapshot*

*hdfs dfs -createSnapshot /examen snapshot2*

**

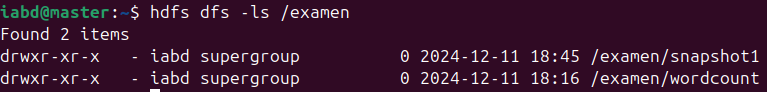
9.2.- Borrar el archivo “Subtitulos\_GameOfThrone.json” de la carpeta “Examen”

*hdfs dfs -rm /examen/Subtitulos\_GameOfThrone.json*

**

9.3.- Comprobar que el archivo ya no existe

*hdfs dfs -ls /examen*



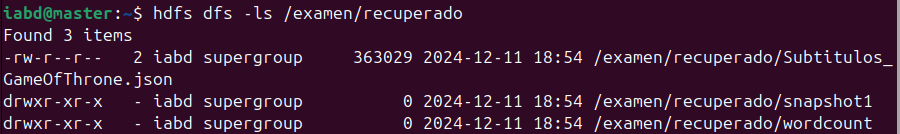
9.4.- Recuperar el archivo: “Subtitulos\_GameOfThrone.json” desde el snapShot1

*hdfs dfs -cp /examen/.snapshot/snapshot2 /examen/recuperado*

**

9.5.- Comprobar que el archivo se ha restaurado

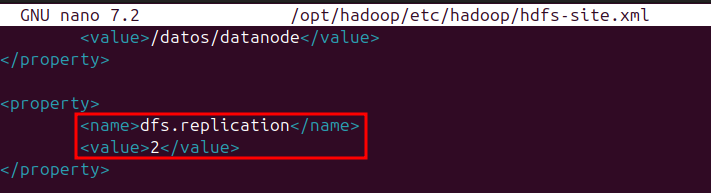
*hdfs dfs -ls /examen/recuperado*

**

10.- Comprobar en la configuración cuántas veces se replican los datos en el clúster.

*Entramos al archivo hdfs-site.xml y comprobamos que se replican 2 veces.*

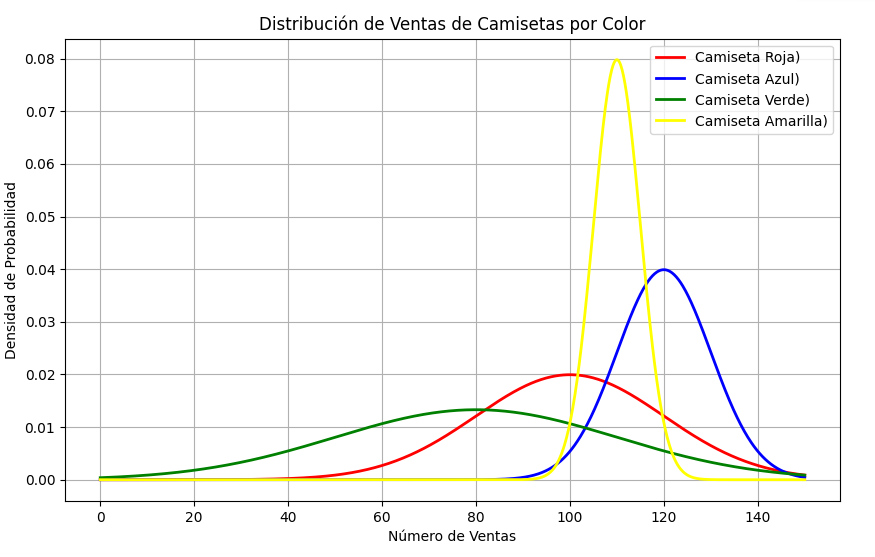
*nano /opt/hadoop/etc/hadoop/hdfs-site.xml*



## análisis de datos

### ejercicio 2 - gráfica 1

Responde a las preguntas:

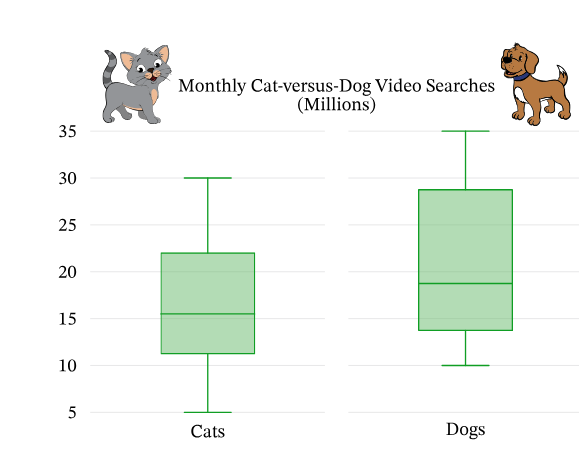


1. ¿Qué gráfica tiene la media mayor? **La grafica amarilla**
2. ¿Qué gráfica tiene la media menor? **La grafica verde**
3. ¿Qué gráfica tiene el menor rango de valores? **La grafica amarilla**
4. ¿Qué gráfica tiene el mayor rango de valores? **La grafica verde**
5. ¿Cuál es el color más estable en cuanto a sus ventas? **El Amarillo**
6. ¿Cuál es el color más inestable en cuanto a sus ventas? **El verde**
7. De media ¿cuál es el color menos vendido? **El verde**
8. De media ¿cuál es el color más vendido? **El amarillo**

### ejercicio 3 - gráfica 2

Durante mucho tiempo se ha pensado que los gatos son la mascota más popular en Internet. Para probar esta teoría, un conocido motor de búsqueda recopiló datos sobre el número de búsquedas de vídeos relacionados con gatos y vídeos relacionados con perros cada mes durante un período de cinco años. Los resultados se representan en los siguientes diagramas de caja.

Observando la gráfica responde a las siguientes preguntas:



1.- ¿Qué gráfica tiene el mayor rango intercuartilico Q1-Q3? **El grafico de los gatos**

2.- ¿Qué gráfica tiene el mayor rango de valores “normales”? **El grafico de los perros**

¿Cuál es ese rango? **Entre 14 y 29**

3.- ¿En qué gráficas hay outliers? **Ninguno tiene outliers**

4.- ¿Qué gráfica tiene la mediana mayor? **El grafico de los perros**

## ALMACENAMIENTO DE DATOS MONGODB

### Pasos previos, arranque de un servidor MongoDB

Para los ejercicios 4,5 y 6 usaremos una imagen de Docker como servidor MongoDB:

* sudo docker run -p 192.0.2.100:27017:27017 --name iabd-mongo -d mongo (Para levantar el servidor)
* sudo docker exec -it iabd-mongo bash (ejecutar un bash dentro del servidor)
* mongosh (una vez dentro del servidor, entrar en mongodb)

Algunos comandos de ayuda para los ejercicios:

* **sudo docker cp prueba.js iabd-mongo:/tmp**: copia un archivo llamado "prueba.js" desde tu sistema host al contenedor Docker "iabd-mongo" en la ubicación "/tmp" dentro del contenedor.
* **mongosh < /tmp/prueba.js**: ejecuta en mongosh las instrucciones del fichero. Para hacerlo desde host o desde el virtual, instalar en mongosh o mongodb.
* **sudo docker stop/restart/start iabd-mongo**: para/reinicia/inicia el contenedor iabd-mongo

### EJERCICIO 4 - DISEÑO DE COLECCIONES

Crea una base de datos MongoDB para una clínica veterinaria que permita gestionar la información de los clientes, sus mascotas y las citas médicas de las mascotas. Damos por hecho que una mascota tiene un único dueño.

Colecciones principales:

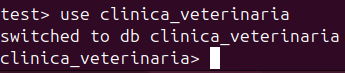
1. **Clientes:** Id, nombre, apellido, teléfono
2. **Mascotas:** Id, nombre, especie, raza, IdCliente (referencia al cliente, que es el dueño de la mascota)
3. **Citas:** Id, fecha, motivo, idMascota (referencia a la mascota)

Nota: Se da por bueno el definir el campo fecha como un string.

Asegúrate de incluir las referencias adecuadas en cada colección para establecer las relaciones. Utiliza identificadores únicos para enlazar los documentos entre colecciones.

1. Crea una BBDD conectándote al contenedor servidor de mongoDB, crea ejemplos de documentos para cada colección. (3 clientes, 3 mascotas y 2 citas).

*Use clinica\_veterinaria*



*db.createCollection("clientes")*



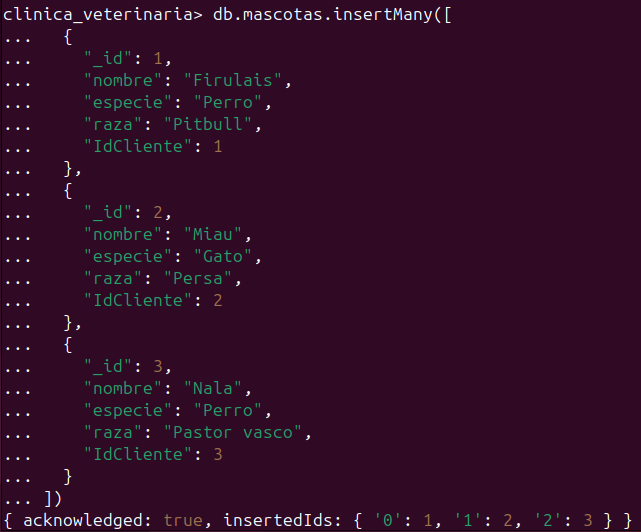
*db.createCollection("mascotas")*

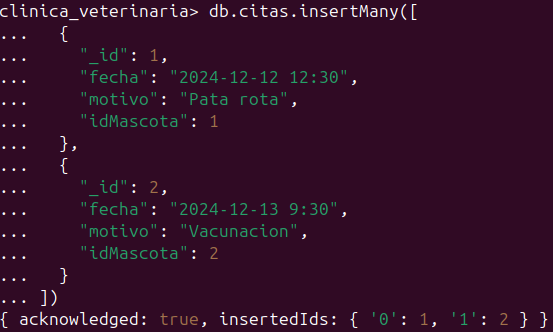


*db.createCollection("citas")*

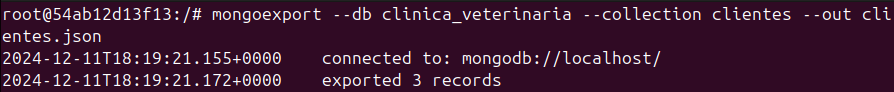


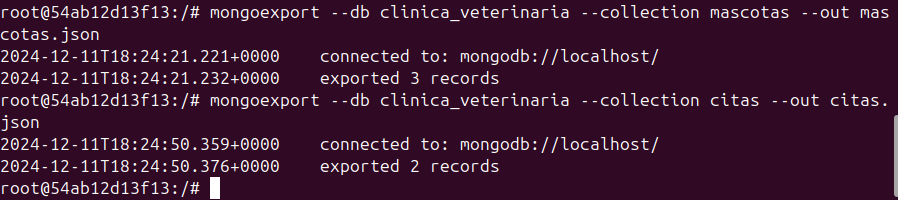


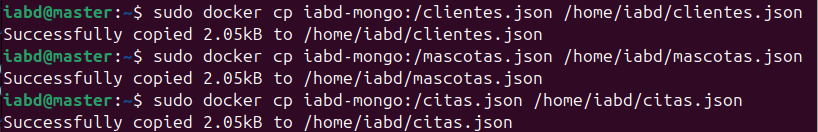




1. Exportar las colecciones a ficheros “JSON”. Sube estos ficheros a moodle junto con el archivo del examen.







### EJERCICIO 5 - CRUD Y AGREGACIONES

Realiza los siguientes ejercicios de CRUD y agregaciones utilizando las colecciones que encontrarás en la carpeta de “E*jercicio5*” junto al enunciado del examen (clientes.json, pedidos.json y productos.json).

La base de datos a la que hay que importar las colecciones se llama “tienda” y no está creada.

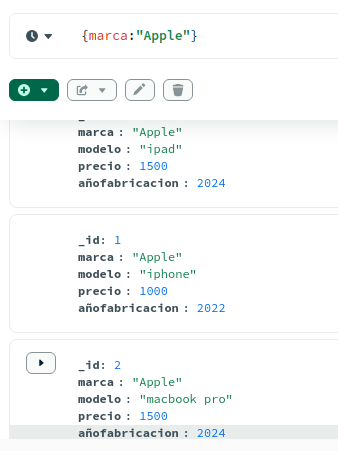
Para este ejercicio usar el **contenedor Docker del ejercicio anterior** o el software **mongoCompass** que está instalado en la máquina virtual Master.

**CRUD**

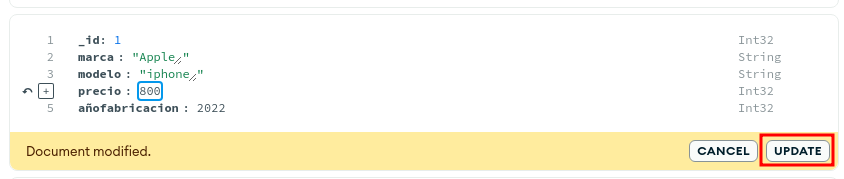
1. Insertar un nuevo cliente.



1. Leer los productos de la marca Apple.

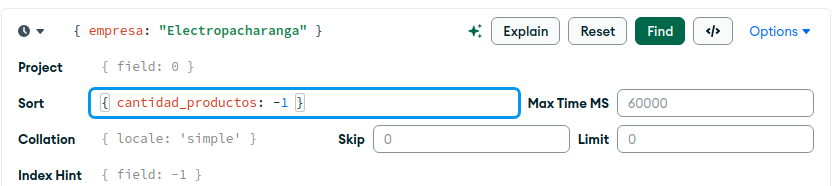


1. Actualizar precio del iphone.





4.- Leer todos los pedidos de la empresa electropacharanga y mostrarlos ordenados en orden decreciente según la cantidad de productos del pedido.





**Agregaciones**

1. **Obtener el total de productos pedidos por cada empresa.**
2. **Obtener el número total de productos vendidos por marca.**
3. **Obtener el número de pedidos por cada marca de producto.**
4. **Obtener el total de ingresos por modelo de producto.**

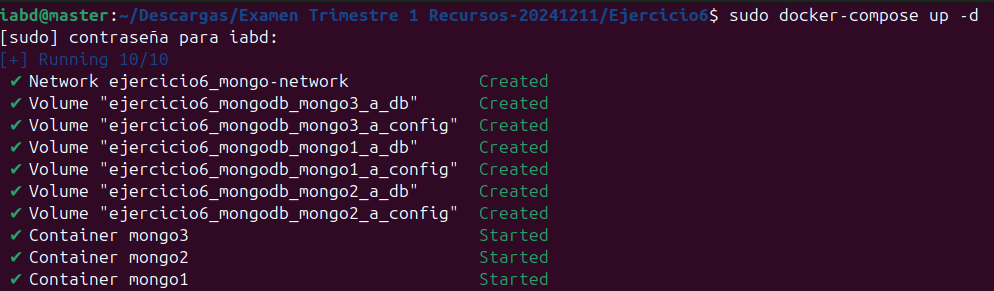
### EJERCICIO 6 - MONGODB RÉPLICAS

Crear un conjunto de réplicas en mongoDB de 3 nodos en 3 contenedores con Docker Compose (nodo 1 principal y nodo 2 y 3 secundarios). Usar el fichero “*Docker-compose.yml*” y el fichero “*init.js*” que están en la carpeta “Ejercicio6” junto al enunciado del examen.

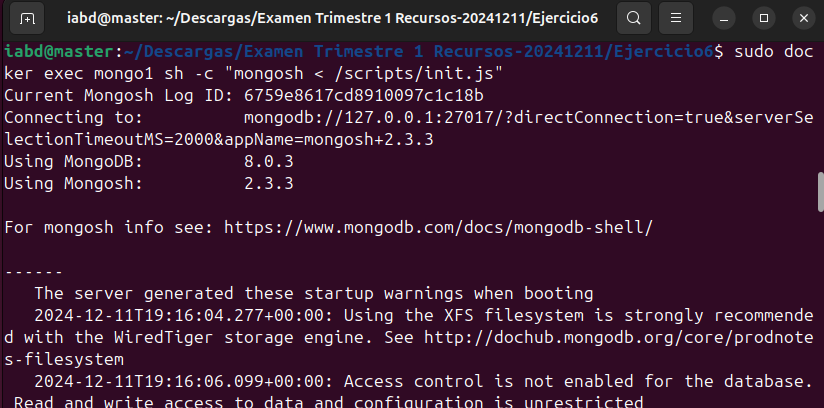
Para el ejercicio, en la carpeta “Ejercicio6” tenéis un fichero llamado images.json:

1. Levantar e inicializar el servicio en los contenedores del conjunto de réplicas.

*Sudo docker-compose up -d*

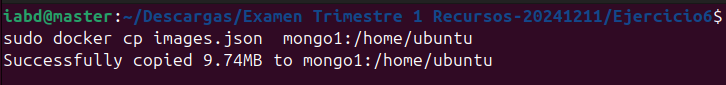


*sudo docker exec mongo1 sh -c "mongosh < /scripts/init.js"*



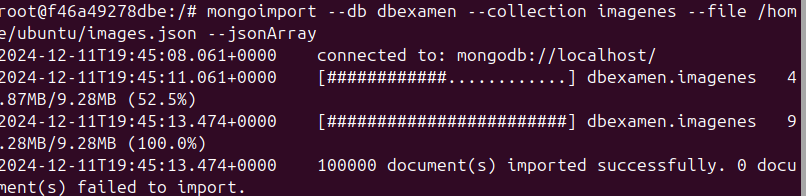
1. Importar el fichero images.json en el nodo primario y obtener el estado del conjunto de réplicas.

*sudo docker cp images.json mongo1:/home/ubuntu*

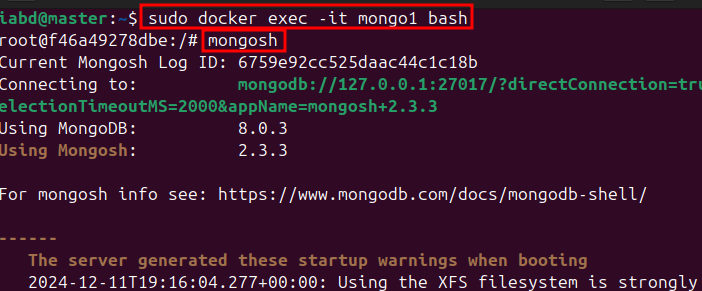


*sudo docker exec -it mongo1 bash*

*mongoimport --db dbexamen --collection imagenes --file /home/ubuntu/images.json –jsonArray*



*mongosh*

**

*rs.status()*



1. Conectarse a un nodo secundario y consultar los datos de una imagen.

*sudo docker exec -it mongo2 mongosh*



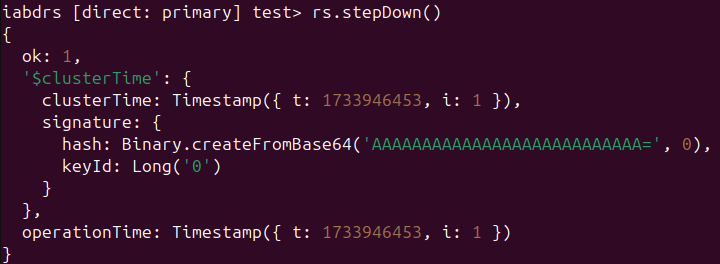
1. Desde el nodo secundario, intenta insertar un documento en la base de datos. ¿Es posible? ¿Sabrías decir por qué?

*No no es posible, supongo que es porque solo tiene permisos de lectura y que el nodo primario es el unico que tiene todos los permisos.*

1. Degradar el nodo primario.

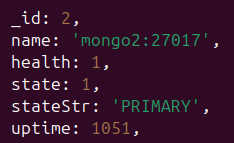
*sudo docker exec -it mongo1 mongosh*

*rs.stepDown()*

**

1. Averiguar cuál es el nuevo nodo primario.

*rs.status*



## CRITERIOS DE EVALUACIÓN

|  |  |
| --- | --- |
| **EJERCICIO1** | **PUNTOS** |
| **HADOOP** | 10 |
| **1.- Levanta los servicios y verifica los servidores web de HDFS y YARN.** | 0.5 |
| **2.- Crea una carpeta y sube un archivo. (usar fichero “Subtitulos\_GameOfThrone.json”)** | 0.5 |
| **3.- ¿Cuántos bloques ocupa el archivo: “Subtitulos\_GameOfThrone.json”)?** | 0.5 |
| **4.- Encarga un trabajo wordCount utilizando el archivo proporcionado.** | 2.0 |
| **5.- ¿Cuánto tiempo le ha llevado realizar la tarea del punto 4?** | 0.5 |
| **6.- Lleva a cabo el wordCount del mismo archivo que en el punto 4 pero aportando tus propios archivos mapper y reducer.** | 2.0 |
| **7.- Descarga el archivo con el resultado al sistema de archivos local y muestra su contenido.** | 0.5 |
| **8.- ¿Cuánto tiempo le ha llevado realizar la tarea?** | 0.5 |
| **9.- Habilita los snapshots, y borra el archivo “Subtitulos\_GameOfThrone.json” después recupera el snapShot y comprueba que el archivo se ha restaurado.** | 2.0 |
| **10.- Comprobar en la configuración cuántas veces se replican los datos en el clúster.** | 1.0 |

|  |  |
| --- | --- |
| **EJERCICIOS 2 y 3** | **PUNTUACIÓN(2p)** |
| **GRÁFICA 1 Cada pregunta: 0.125p** | 1 |
| **GRÁFICA 2 Cada pregunta: 0.25p** | 1 |

|  |  |
| --- | --- |
| **EJERCICIO 4** | **PUNTUACIÓN(2p)** |
| **Diseño de la base de datos.** | 1.5 |
| **Exportación de las BBDD desde MongoCompass** | 0.5 |

|  |  |
| --- | --- |
| **EJERCICIO 5** | **PUNTUACIÓN(3p)** |
| **Importar la BBDD en formato json** | 0.5 |
| **Implementación correcta de las consultas CRUD** | 1.0 |
| **Implementación correcta de las agregaciones** | 1.0 |
| **Uso de diferentes medios para consultas o agregaciones** | 0.5 |

|  |  |
| --- | --- |
| **EJERCICIO 6** | **PUNTUACIÓN(3p)** |
| **Levantar, arrancar y obtener el estado del conjunto de replicas** | 0.6 |
| **Conectarse a un nodo primario e importar BBDD** | 0.6 |
| **Conectarse a un nodo secundario y leer los datos de una imagen** | 0.6 |
| **En el nodo secundario escribir datos** | 0.6 |
| **Degradar nodo primario y averiguar cuál pasa a ser el primario** | 0.6 |