Cuaderno de prácticas

Repositorio

Este curso dispone de un repositorio en github donde puede encontrarse material adicional de apoyo.
https://github.com/evalero/RedHat-Mongodb.git

Tema 2 Instalación Básica de MongoDB

En la máquina virtual ofrecida **mongodb se encuentra instalado en su versión más reciente** para evitar posibles problemas con conexiones a internet

Aún así vamos a describir los pasos a seguir para llevar a cabo dicha tarea

Instalación mediante binarios precompilados

Accedemos a la web de MongoDB y descargamos el paquete correspondiente a nuestro SO (en este caso Red Hat 7):
https://www.mongodb.org/downloads#production
O directamente hacemos wget desde la máquina a la url del paquete precompilado:

```
wget
https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-3.2.4
.tgz
```

Una vez descargado el paquete, podemos descomprimirlo en un directorio :

```
sudo tar -xzvf mongodb-linux-x86_64-rhel70-3.2.4.tgz -C /opt
```

Actualizamos las variables de entorno en el bash_profile del usuario

```
vi ~/.bash_profile
MONGO_PATH=/opt/mongodb/bin
PATH=$PATH:$HOME/.local/bin:$HOME/bin:$MONGO_PATH

export PATH
```

Instalación usando repositorio

Importamos la public key de MongoDB

```
sudo rpm --import
https://www.mongodb.org/static/pgp/server-3.2.asc
```

Movemos el fichero ~/RedHat-Mongodb/Tema2/mongodb-org-3.2.repo a la ruta /etc/yum.repos.d/

```
sudo mv ~/RedHat-Mongodb/Tema2 /etc/yum.repos.d/
```

Instalamos el paquete

```
sudo yum install -y mongodb-org
```

Para las prácticas, SELINUX está deshabilitado. Para entornos productivos, se recomienda habilitarlo y añadir la siguiente regla para utilizar el puerto por defecto:

```
semanage port -a -t mongod_port_t -p tcp 27017
```

Instalando por paquetería mongoDB incluyen los scripts de arranque y parada de MongoDB (incluidos en el paquete mongodb-org-server o mongodb-org)
https://docs.mongodb.org/manual/administration/install-community/

Arrancar instancia de MongoDB

Por línea de comandos

Primeramente vamos a probar a arrancar una instancia por línea de comandos. Para ello, podemos utilizar el siguiente comando:

```
mongod --dbpath /data/db/ --port 27017 --fork --logpath
/var/log/mongod/mongod.log
```

Este comando arrancará el servidor de BBDD para que utilice la ruta /data/db como lugar para almacenar los datafiles, utilizará el puerto 27017 (puerto por defecto de MongoDB) como puerto de escucha, hará fork del proceso para devolvernos el prompt y utilizará la ruta /var/log/mongod/mongod.log para almacenar los logs

Usando fichero de configuración

Para entornos productivos o estables, es altamente recomendable utilizar ficheros de configuración. En el repositorio hemos incluido un fichero de configuración que realiza la misma instanciación que el ejemplo anterior.

Si queremos arrancar con fichero de configuración:
Matamos el proceso de mongo que podamos tener levantado (si lo hemos realizado el ejemplo anterior)

```
killall mongod
```

Y ejecutamos:
Echamos un vistazo al contenido del fichero:

```
cat   ~/RedHat-Mongodb/Tema2/ficheroConfiguracion/mongod.conf
```

Una vez revisado el contenido, ejecutamos :

```
mongod -f ~/RedHat-Mongodb/Tema2/ficheroConfiguracion/mongod.conf
```

Tema 3 Shell de MongoDB

Primeros pasos

Levantamos una instancia de MongoDB si no hemos dejado una previamente levantada como se ha descrito en los ejercicios anteriores.
Ahora nos conectamos a la Shell de MongoDB ejecutando el comando:

```
mongo
```

Después vamos a visualizar las BBDD que hay en la instancia

```
show dbs
```

Vamos a crear nuestra primera BBDD. No hace falta crearla previamente, podemos usar directamente este comando para posicionarnos.

```
use miBD
```

La base de datos como tal quedará creada en el momento en el que exista un solo documento en una colección en la BBDD. Vamos a crear nuestro primer documento.

```
db.holaMundo.insert({“Nombre”: “<tu nombre>” , “Apellido” :”<tu apellido>” , edad : <tu_edad> })
```

Por ejemplo:

```
db.holaMundo.insert({"Nombre" : "Ernesto", "Apellido" : "Valero", "edad":30})
```

Ahora vamos a ver las colecciones existentes en nuestra BBDD:

```
show collections
```

Ahora vamos a buscar el documento que hemos generado en nuestra colección:

```
db.holaMundo.find({})
```

Comandos administrativos

Vamos a analizar algunos comandos administrativos que suelen ser utilizados para hacer troubleshooting de algunos problemas en MongoDB
Ejecutamos los siguientes comandos

```
use miBD
db.stats()
```

Observaremos algunos datos interesantes tales como el tamaño usado en disco por la BBDD, la cantidad de índices creados y espacio utilizados por los mismos.

```
Vamos a ejecutar una query sobre la BBDD:
```

db.muchosDatos.find({x:{$gte:1040}},{y:1})

Si queremos saber cuál ha sido el plan ganador que la BBDD ha ejecutado, podemos ejecutar al final de la query la operación explain()

```
db.muchosDatos.find({x:{$gte:1040}},{y:1}).explain()
```

Vamos a crear ahora un índice para ver el impacto para ver el aumento de espacio generado por el nuevo índice:

```
db.muchosDatos.createIndex({x:1,y:1})
db.stats()
```

Ahora volvemos a ejecutar la query y ejecutamos el explain para ver como se ha modificado el query planner al ejecutar la query

```
db.muchosDatos.find({x:{$gte:1040}},{y:1})
db.muchosDatos.find({x:{$gte:1040}},{y:1}).explain()
```

Ahora vamos a habilitar el profiling en la base de datos al máximo nivel para que todas las querys sean guardadas en la colección system-profile de la BBDD miBD

```
db.setProfilingLevel(2)
db.muchosDatos.insert({“mensaje”:”Este mensaje va a ser traceado”})
db.system-profile.find()
```

Carga de datos y querys

Vamos a cargar algunos datos adicionales. Salimos de la consola de mongo ejecutando el comando:
exit
Nos posicionamos en el directorio donde está el script de carga

```
cd ~/RedHat-Mongodb/Tema3/scripts
```

Después cargamos los datos ejecutando:

```
mongo generaDatos.js
```

Nos metemos en la instancia de Mongo y comprobamos que colecciones tienen datos

```
#mongo Script
show collections
db.coleccionX.find()
```

Donde x es un número de 0 a 9
Salimos de la estancia de Mongo y ejecutamos el script

```
#mongo borradoColecciones.js
```

Repetimos el paso anterior y comprobamos cuál ha sido el resultado de la ejecución del script

Tema 5 Replica set y Sharding

Creación de Replica Set

Vamos a crear un entorno en replica set usando un único nodo simulando un entorno de 3 instancias con replica factor 3

Paramos todas las instancias de mongodb existentes

```
killall mongod
```

Creamos 3 directorios para almacenar los datos:

```
mkdir -p /data/RS1_1
mkdir -p /data/RS1_2
mkdir -p /data/RS1_3
```

Levantamos 3 instancias

```
mongod --replSet RS1 --dbpath /data/RS1_1 --port 27017
--noprealloc --nojournal --logpath /data/RS1_1/mongod.log &
mongod --replSet RS1 --dbpath /data/RS1_2 --port 27018
--noprealloc --nojournal --logpath /data/RS1_2/mongod.log &
mongod --replSet RS1 --dbpath /data/RS1_3 --port 27019
--noprealloc --nojournal --logpath /data/RS1_3/mongod.log &
```

Nos conectamos a una de las instancias usando el comando **mongo**
Una vez ejecutamos los siguientes comandos:

```
rs.initiate()
rs.add("mongodb:27018")
rs.add("mongodb:27019")
```

Una vez creado el replica set podemos alimentarlo con datos usando el siguiente script:

```
mongo populateDB.js
```

Creación de Sharding

Paramos todas las instancias de mongodb en ejecución
killall mongod

Config servers

Creamos los directorios:

```
mkdir -p /data/configdb1
mkdir -p /data/configdb2
mkdir -p /data/configdb3
```

Y levantamos 3 instancias:

```
mongod --configsvr --replSet config --dbpath /data/configdb1
--port 27019 --logpath /data/configdb1/mongod.log &

mongod --configsvr --replSet config --dbpath /data/configdb2
--port 27029 --logpath /data/configdb2/mongod.log &

mongod --configsvr --replSet config --dbpath /data/configdb3 --port
27039 --logpath /data/configdb3/mongod.log &
```

Nos conectamos a una de ellas e inicializamos el replica set
mongo --port 27019

```
rs.initiate()
rs.add("mongodb:27029")
rs.add("mongodb:27039")
```

Mongos
Inicializamos el proceso mongos

```
mongos --configdb config/mongodb:27019,mongodb:27029,mongodb:27039
&
```

Shards
Creamos los directorios

```
mkdir -p /data/shard1
mkdir -p /data/shard2
```

Inicializamos los procesos para cada shard

```
mongod --shardsvr --replSet RS1 --dbpath /data/shard1 --port 27018
--noprealloc --nojournal --logpath /data/shard1/mongod.log &
mongod --shardsvr --replSet RS2 --dbpath /data/shard2 --port 27028
--noprealloc --nojournal --logpath /data/shard2/mongod.log &
```

Inicializamos el replica set en cada nodo

```
mongo --port 27018
rs.initiate()
exit
mongo --port 27028
rs.initiate()
exit
```

Inicialización del shard

Nos conectamos a la instancia de Mongos y añadimos los shards

```
sh.addShard("RS1/mongodb:27018")
sh.addShard("RS2/mongodb:27028")
```

Creamos una base de datos llamada mishard e inicializamos el shard

```
use mishard
db.particion.createIndex({x:1})
sh.enableSharding("mishard")
sh.shardCollection("mishard.particion",{x:1})
sh.status()
```

Alimentamos la BBDD

```
for (i = 0 ; i < 1000000 ; i ++ ){
   db.particion.insert({x:i,y:i*2,z:i-100});
}
```

y ejecutamos nuevamente

```
sh.status()
```

Tema 6 Recomendaciones generales

Instancia con autenticación básica habilitada

Vamos a levantar una instancia con autenticación.
Paramos todas las instancias de mongodb que podamos tener levantadas
killall mongod

Nos posicionamos en el directorio ~/RedHat-Mongodb/Tema6/autenticación
```
cd ~/RedHat-Mongodb/Tema6/autenticación
```

Visualizamos el contenido del fichero de configuración observando de manera más particular el apartado del bloque security:
```
cat mongod.conf
```

Levantamos la instancia usando el fichero de configuración proporcionado

mongod -f mongod.conf

Nos conectamos a la base de datos y creamos un usuario con rol root
```
#mongo
use admin
db.createUser({user:"root",pwd:"pass",roles:
[{role:"root",db:"admin"}]})
```
exit
Nos volvemos autenticar en la instancia con el usuario administrador
mongo admin -uroot -p
pass

Y vamos a crear un usuario de solo lectura en la BBDD mites
```
use miTest
```
db.createUser({user:"readonly",pwd:"read",roles: [{role:"read",db:"miTest"}]})
```
exit
```

Ahora nos autenticamos con el usuario de solo lectura
mongo miTest -ureadonly -p
read

E intentamos escribir un documento
db.noPermitido.insert({"documento":"No introducido"})

Tema 9 : CRUD

Querys básicas:

Vamos a introducir algunos documentos en una base de datos y hacer unas querys básicas
use crud

```
db.name.insert({a:"1"})
db.name.insert({a:"2", b : null})
db.name.insert({a:1, b:"1"})
db.name.insert({a:2, b:"1", texto:"hola"})
db.name.insert({a:"1", c:true})
db.name.insert({d:{f:1}})
```

Y ahora vamos a hacer algunas querys para observar su resultados:

```
db.name.find({a:”1”})
db.name.find({a:”1”,c:true})
db.name.find({a:”1”,b:"1"})
db.name.find({”d.f”:”1”})
db.name.find({d : {f:1}})
```

Querys sobre juego de datos

Vamos a cargar el juego de datos que existe en el directorio del laboratorio. Para ello nos posicionamos en el directorio donde está el dump:

```
cd ~/RedHat-Mongodb/Anexo
```

Y ejecutamos el comando:

```
mongorestore
```

Nos conectamos a la instancia usando el comando:

```
mongo
```

Nos posicionamos en la base de datos students y después observamos la estructura de los documentos:

```
use students
db.grades.findOne()
```

Vamos a buscar el usuario con identificador 180

```
db.student.find({student_id : 180})
```

Ahora vamos a buscar todos los resultados que sean de tipo quiz:
db.grades.find({type:quiz})

Queremos obtener los ejercicios de tipo quiz del estudiante 180
```
db.grades.find({type : "quiz",student_id : 180})
```

Si queremos buscar todos los ejercicios de tipo quiz o exam:
```
db.grades.find({type : { $in:["quiz","exam"]}})
```

Ahora vamos a contar todos los ejercicios que son de tipo quiz pero no de tipo exam
```
 db.grades.count({type : {$ne : "exam",$eq : "quiz"}})
```

Si queremos obtener todos los resultados entre 90 y 95 podríamos pensar en usar esta query:
```
 db.grades.find( { score : { $gt : 90 }, score : { $lt : 95 } } )
```
Sin embargo la segunda parte de la query nos mostrarán resultados que están por debajo de 90. Para obtenerlo por el intervalo cerrado la query correcta sería:
```
 db.grades.find({score : {$gt : 90,$lt : 95}})
```

Si queremos mostrar los resultados de tipo test o exam:
```
db.grades.find({$or : [{type:"quiz"},{type:"exam"}]})
```

Obtener todos los documentos de tipo quiz pero no de tipo exam

```
db.grades.find({$or : [{type:"quiz"},{type:{$ne:"exam"}}]})
```

En cambio si queremos obtener todos los documentos que sean de tipo quiz pero no de tipo exam
```
db.grades.find({$and : [{type:"quiz"},{type:{$ne:"exam"}}]})
```

Operaciones sobre arrays

Vamos a hacer querys sobre una colección con documentos en los que vienen todos los resultados de cada una de las pruebas por cada estudiante.
Primero observamos la estructura de los documentos:
```
db.gradeArray.findOne()
```

Vamos a buscar el estudiante 180
```
db.gradeArray.find({student_id : 180})
```

Buscar los estudiantes más de un 80 en los ejercicios de tipo exam
```
db.gradeArray.find({ test : {$all : [ { "$elemMatch" : { score :
{ $gt: 80.0},type:"exam" } } ]} })
```

Ejemplos que no aplicarían serían:
```
db.gradeArray.find({ 'test.score' :   { $gt: 80.0}  , 'test.type'
:"exam"  })
```
Este caso va a mostrar aquellos documentos que independientemente del tipo de ejercicio, la nota sea mayor que 80

Si queremos buscar los estudiantes que han sacado más de un 50 en cada uno de los tipos de prueba

db.gradeArray.find({ test : {$all : [{ "$elemMatch" : { score : { $gt: 50.0},type:"exam" } },
{ "$elemMatch" : { score : { $gt: 50.0},type:"homework" } },
{ "$elemMatch" : { score : { $gt: 50.0},type:"quiz" } }]} })

Proyecciones

Si queremos mostrar únicamente ciertos elementos del documento, podemos hacer proyecciones.

Mostrar solo las notas estudiante180

```
db.grades.find({student_id:180},{_id:0, score:1})
```

Para grade array:

```
db.gradeArray.find({_id:180},{_id:0, 'test.score':1}).pretty()
```

Actualizaciones

Si queremos sumar un 5 a la nota del quiz del estudiante 180

```
 db.grades.update ({student_id:180,type:'quiz'},{ $inc : { score : 5 }})
```

añade el campo test ok cuando el alumno haya sacado más de un 50 y divide las notas por 10

```
db.grades.update ({score:{$gte : 50.0}},{ $set : { test : 'ok' },$mul : {score: 0.1 }},{multi:true})
```

Eliminar el campo quiz del estudiante 180

db.grades.update ({student_id:180,type:'quiz'},{ $unset : { quiz : 1 }})