MongoDB es un sistema manejador de Base de Datos orientado a documentos

Instalación de la versión actual (3.0) para Ubuntu 14.04

La suite MongoDB contiene los siguientes paquetes:

- **mongodb-org:** metapaquete, el cual se encargará de listar los cuatro paquetes que se listan a continuación
- **mongodb-org-server:** contiene el demonio mongd y los archivos de configuración e inicio.
- **mongodb-org-mongos**: contiene los demonios mongo
- **mongodb-org-shell:** contiene la consola de mongo
- **mongodb-org-tools**: contiene las siguientes herramientas mongoimport bsondump, mongodump,mongoexport, mongofiles, mongooplog, mongoperf, mongorestore, mongostat, andmongotop.

Al momento de descargar el paquete mongodb-org, los script de inicio de configuran en la ruta: **/etc/init.d/mongod**. Estos script funciona para el iniciar, detener o reiniciar el demonio mongod.

El archivo de configuración de demonio (mongod.conf) se encuentra por defecto en **/etc/mongod.conf**

La ip que se configura por defecto al momento de su instalacion en 127.0.0.1

Pasos para su instalación:

1) Importar la llave pública que utiliza el manejador de paquete:

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10

2) Crear un list file para Mongo

echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list

3) Recargar la base de datos local de paquetes

sudo apt-get update

4) Instalar los paquetes de MongoDB para la versión más reciente estable

sudo apt-get install -y mongodb-org

Luego de MongoDB se encuentra instalado, procedemos a iniciar sus servicios, si estos no se inician por defecto al arrancar la máquina:

```
nosql@nosql-VirtualBox:~$ sudo service mongod start
mongod start/running, process 2744
```

Para ver verificar si en verdad el servidor está arriba:

```
nosql@nosql-VirtualBox:~$ sudo service mongod status
mongod start/running, process 2744
```

Iniciar mongo: escribir en la consola mongo

```
nosql@nosql-VirtualBox:~$ mongo
MongoDB shell version: 3.0.7
connecting to: test
Server has startup warnings:
2015-11-19T09:59:25.785-0430 I CONTROL  [initandlisten]
2015-11-19T09:59:25.785-0430 I CONTROL  [initandlisten]
mm/transparent_hugepage/enabled is 'always'.
2015-11-19T09:59:25.785-0430 I CONTROL  [initandlisten]
ting it to 'never'
2015-11-19T09:59:25.785-0430 I CONTROL  [initandlisten]
2015-11-19T09:59:25.785-0430 I CONTROL  [initandlisten]
mm/transparent_hugepage/defrag is 'always'.
2015-11-19T09:59:25.785-0430 I CONTROL  [initandlisten]
ting it to 'never'
2015-11-19T09:59:25.785-0430 I CONTROL  [initandlisten]
>
```

Nota: cuando se instala mongo, la autenticación se encuentra deshabilitada por lo tanto se puede crear base de datos, colecciones y consultas sin necesidad de autenticarse.

Comandos Básicos de MongoDB

- **Crear una base de datos: use <nombre_base_datos>**

```
> use prueba
switched to db prueba
>
```

- **Crear una colección: db.createCollection('<nombre>'):** en MongoDB no existe el concepto de tablas, sin embargo se puede agrupar documentos en colecciones, las cuales son homólogas a las tablas en los sistemas relaciones.

```
> db.createCollection('persona')
{ "ok" : 1 }
>
```

Los nombres de las BD y colecciones no pueden exceder de 120 bytes.

- **Crear un documento:** cada documento debe estar asociado a una colección, y esto es lo homologó a las filas en un sistema relacional. Los tipos de datos de estos documentos son: string, enteros, reales, fechas y arreglos.

  Los documentos deben tener un _id que sea único, si no se especifica al momento de crearlo, Mongo lo crea por defecto. Estos documentos deben seguir el siguiente formato:

  documento: { _id: 1, campo: “valor”, campo2:”valor”, campo3:[x, y, z]}

  Un documento puede tener documentos anidados o embebidos, sin embargo, para esta versión de mongo solo se soportan 100 niveles de documentos embebidos.

  documento: {campo: “valor”,
  campo2:”valor”,
  campo3:[x, y, z],
  documento1: {campo: “valor”, campo2:”valor”, } }

  Los arreglos no solo pueden ser de atributos primitivos, también puede ser un arreglo de objetos. Ejemplo de un documento:

```
db.persona.insert({_id:1,
                   nombre:'Adriana',
                   apellido:'Ponceleon' ,
                   materia:[{codigo:'01',descrip:'BD'},{codigo:'02',descrip:'ABD'}]})
```

  Los documentos en esta versión no puede exceder de 16MB

  No soporta nativamente aplicarle hash a un campo tipo password, esto se debe realizar a través del lenguaje de programación elegido

  Se puede tener un archivo .js donde se tendría todas las creaciones de las colecciones y sus inserciones. Para cargarlo en mongo se utiliza la siguiente instrucción**:**

  load(“ruta/<nombre.js”)

  Si la carga fue exitosa debería devolver “true”, si no te indica la línea del archivo que tiene errores.

- **Para consultar los documentos:**

```
> db.persona.find().pretty()
{
        "_id" : ObjectId("564de3a0d7193629d60cbfc8"),
        "nombre" : "Gabriela",
        "apellido" : "Ponceleon",
        "sueldo" : 12.5
}
{
        "_id" : 1,
        "nombre" : "Adriana",
        "apellido" : "Ponceleon",
        "materia" : [
                {
                        "codigo" : "01",
                        "descrip" : "BD"
                },
                {
                        "codigo" : "02",
                        "descrip" : "ABD"
                }
        ]
}
```

Habilitar la autenticación en MongoDB

La autenticación en MongoDB es basada en roles . Cada usuario que se crea se le debe asignar los roles para que pueda realizar las acciones (operaciones) sobre los recursos (BD, colecciones o cluster).

Mongo almacena toda la información de los usuarios en la colección **system.users** de la db **admin**

Roles:

- **read:** permite leer los datos de las colecciones que no son del sistema y las colecciones system.indexes, system.js y system.namespace.

- **readwrite:** permite leer y escribir las colecciones que no son del sistema y en la colección system.js.

- **dbAdmin**: no permite leer en colecciones que no son del sistema, permite realizar operaciones de administración, mas no concede privilegios.

- **dbOwner:** acciones administrativas de la BD, este rol combina los privilegios de readwrite, dbAdmin y userAdmin.

- **UserAdmin**: crea y modifica usuarios en la actual BD, el el rol para un super usuario.

El usuario root es el superusuario de MongoDB

- **backup:** permite realizar operaciones de respaldo de la BD.

- **restore**: permite realizar operaciones de recuperación de la BD.

Pasos para habilitar el Control de Accesos a los Clientes:

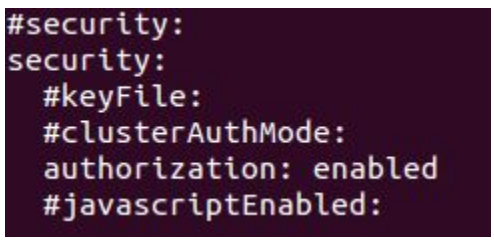
1) Crear el usuario **userAdmin** antes de habilitar la autenticación.
    - Conectarse a mongo
    - Conectarse a la **BD admin**: use admin
    - Crear el usuario:

      db.createUser({ user: “nombre_usuario”,
      pwd: password (debe ser string),
      roles: [{role: “userAdminAnyDatabase”, db:”admin”}]})

Siempre se debe indicar a que bd se puede conectar o es dueño

Nota: si se le asigna solo ese rol, se podrá crear usuarios pero no se podra obtener información de las BD existentes en mongo. Para poder hacer esto, se le debe dar el rol: **userAdminAnyDatabase**

2) Luego de haber creado el usuario, se modifica el archivo mongod.conf para habilitar la autenticación, se debe modificar el campo de security (el cual por defecto viene vacío, por lo tanto copiar y pegar lo siguiente)



3) Después de guardar los cambios reiniciar el servidor: sudo service mongod restart, luego ver el status de mongo.

   Nota: si luego de hacer esos cambios, mongo no funciona lo primero que se debe hacer es deshacer los cambios y reiniciar, más si esto no funciona seguir los siguientes pasos:

    - Remover el archivo mongo.lock

sudo rm /var/lib/mongodb/mongod.lock

- Reparar mongo

  mongod --dbpath /var/lib/mongodb -–repair

  La se debe colocar la ruta que indica el archivo de mongod.conf, sino ese comando se asumirá que debe ser la ruta por defecto db/data que podría no existir.

- Reiniciar mongo
- Iniciar el cliente

4) Luego de habilitar la autenticación, si se trata de iniciar el cliente, este se conectará por defecto a la bd test

```
nosql@nosql-VirtualBox:~$ mongo
MongoDB shell version: 3.0.7
connecting to: test
```

5) Para conectarte como el usuario admin:

```
nosql@nosql-VirtualBox:~$ mongo --port 27017 -u "gponce" -p 123456 --authenticat
ionDatabase "admin"
```

6) Cambiarse a la bd admin y crear un usuario dbOwner

```
> db.system.users.find().pretty()
{
        "_id" : "sudebip.sudebip",
        "user" : "sudebip",
        "db" : "sudebip",
        "credentials" : {
                "SCRAM-SHA-1" : {
                        "iterationCount" : 10000,
                        "salt" : "16B2nwOC2jaFugUN79PQMQ==",
                        "storedKey" : "osHONxrbNCB8H9WEpPOqP6lBaeM=",
                        "serverKey" : "kco8CVfnbEDjl4IXcrnb0wphzhY="
                }
        },
VBOXADDITIONS_4.3.24_98716
                        "role" : "dbOwner",
                        "db" : "sudebip"
                }
        ]
}
```

Asignación de privilegios sobre un objeto:

db.system.users.update({_id:“admin.gponce”},{$set:{privileges:[{resource:{db:”admin”,collection: “system.users”},actions:[“remove”,”update”]}]}}).
Ejemplo de usuario con privilegios sobre objetos:

En este ejemplo se les dio estos privilegios para poder editar o eliminar usuarios, debido que con los roles asignados anteriormente no se podía realizar estas operaciones.

Si se quiere conceder privilegios al usuario administrator, se debería deshabilitar la configuración (comentar security en el archivo mongod.conf y reiniciar), conectarse a la bd admin y concederse el privilegio.

Luego de conceder, deshacer el cambio en el archivo y reiniciar.

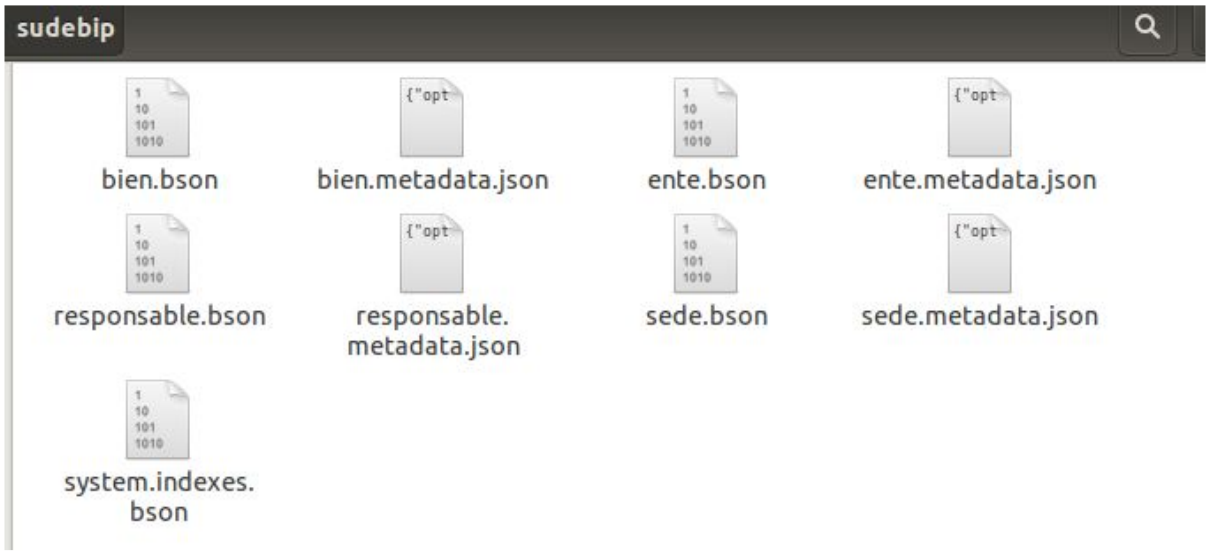
<u>Exportar una BD en MongoDB:</u>

1) Dirigirse a la ruta donde se encuentra instalado mongodump, para poder correr el ejecutable. Para este tutorial se encontraba en /usr/bin
2) Correr el siguiente comando:

mongodump -h <ip_servidor> -d <nombre_BD> -u <usuario> - p <password> -o <ruta_donde_va_el_archivo>

```
nosql@nosql-VirtualBox:/usr/bin$ mongodump -h localhost -d sudebip -o /home/nosq
l/Desktop
2015-11-26T09:50:23.700-0430    writing sudebip.bien to /home/nosql/Desktop/sude
bip/bien.bson
2015-11-26T09:50:23.714-0430    writing sudebip.bien metadata to /home/nosql/Des
ktop/sudebip/bien.metadata.json
2015-11-26T09:50:23.723-0430    done dumping sudebip.bien (4 documents)
```

3) Se creará una carpeta con el nombre de la BD, dentro de esta carpeta se encontrarán todas las colecciones en formato BSON



Replicación:

Mongo sigue un esquema de replicación donde se tiene un nodo primario, donde se escribe y lee en mongo, este nodo réplica a tantos nodos secundarios se tengan.

Este tutorial se trata de convertir una instancia standalone en un esquema de replicación de un nodo primario con dos nodos secundarios.

1) Modificar el archivo mongod.conf para activar la opción de replicación:

```
#replication:
replication:
   #oplogSizeMB:
   replSetName: "nosql-VirtualBox"
   #secondaryIndexPrefetch:
```

En el nombre replSetName se debe colocar el nombre del host. Guardar el cambio

2) Se debe crear los directorios donde los nodos almacenan los datos y apagar la instacia: sudo service mongod stop

```
root@nosql-VirtualBox:/srv/mongodb/db# ls
db0  db1  db2  journal  local.0  local.1  local.ns  storage.bson  _tmp
```

3) Ejecutar el siguiente comando:

sudo mongod --port 27017 --dbpath /var/lib/mongodb --replSet rs

Este comando levantará el servidor:

```
nosql@nosql-VirtualBox:~$ sudo mongod --port 27017 --dbpath /var/lib/mongodb --replSet rs
2015-11-26T10:37:39.480-0430 I JOURNAL  [initandlisten] journal dir=/var/lib/mongodb/journal
2015-11-26T10:37:39.480-0430 I JOURNAL  [initandlisten] recover : no journal files present, no recovery
eeded
```

4) Levantar 2 servidores mas, pero con puertos y dbpath diferentes:

```
nosql@nosql-VirtualBox:~$ sudo mongod --port 27004 --dbpath /srv/mongodb/db/db0 --replSet rs
[sudo] password for nosql:
2015-11-26T10:40:18.095-0430 I JOURNAL  [initandlisten] journal dir=/srv/mongodb/db/db0/journal
2015-11-26T10:40:18.096-0430 I JOURNAL  [initandlisten] recover : no journal files present, no recovery n
```

```
nosql@nosql-VirtualBox:~$ sudo mongod --port 27006 --dbpath /srv/mongodb/db/db1 --replSet rs
2015-11-26T10:46:46.998-0430 I JOURNAL  [initandlisten] journal dir=/srv/mongodb/db/db1/journal
2015-11-26T10:46:47.017-0430 I JOURNAL  [initandlisten] recover : no journal files present, no recovery n
eeded
```

5) Conectarse a través de otro terminar con el cliente de mongo

```
nosql@nosql-VirtualBox:~$ mongo --port 27017
MongoDB shell version: 3.0.7
connecting to: 127.0.0.1:27017/test
```

6) verificar el estado del nodo, con el comando rs.status()

```
> rs.status()
{
        "info" : "run rs.initiate(...) if not yet done for the set",
        "ok" : 0,
        "errmsg" : "no replset config has been received",
        "code" : 94
}
```

7) verificar la configuración del nodo rs.config()

```
> rs.config()
2015-11-26T10:55:24.880-0430 E QUERY    Error: Could not retrieve replica set
nfig: {
        "info" : "run rs.initiate(...) if not yet done for the set",
        "ok" : 0,
        "errmsg" : "no replset config has been received",
        "code" : 94
}
    at Function.rs.conf (src/mongo/shell/utils.js:1017:11)
    at (shell):1:4 at src/mongo/shell/utils.js:1017
```

8) Este nodo será el primario, por lo tanto se debe ejecutar el comando rs.initiate()

```
> rs.initiate()
{
        "info2" : "no configuration explicitly specified -- making one",
        "me" : "nosql-VirtualBox:27017",
        "ok" : 1
}
```

9) Luego volver a ejecutar rs.config(), para indicar que es el primario

```
rs:SECONDARY> rs.config()
{
        "_id" : "rs",
        "version" : 1,
        "members" : [
                {
                        "_id" : 0,
                        "host" : "nosql-VirtualBox:27017",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {

                        },
                        "slaveDelay" : 0,
                        "votes" : 1
```

```
rs:PRIMARY>
```

10) Añadir los nodos secundarios

```
rs:PRIMARY> rs.add('mongo:27001')
{ "ok" : 1 }
rs:PRIMARY> rs.add('mongo:27002')
{ "ok" : 1 }
```

11) Verificar si los otros nodos son secundarios

```
rs:SECONDARY> rs.status()
{
        "set" : "rs",
        "date" : ISODate("2015-11-26T18:33:01.391Z"),
        "myState" : 2,
        "members" : [
                {
```

12) Comprobar si las colecciones todavía existen

```
rs:PRIMARY> show collections
bien
ente
system.indexes
rs:PRIMARY> show collections
bien
ente
responsable
sede
system.indexes
```

13) En los secundarios por defecto no se puede leer por el cliente, pero existe una pequeña modificación para poder lograr esto:

```
rs:SECONDARY> db.getMongo().setSlaveOk()
```

```
rs:SECONDARY> db.getMongo().setSlaveOk()
rs:SECONDARY> use sudebip
switched to db sudebip
rs:SECONDARY> show collections
bien
ente
responsable
sede
system.indexes
rs:SECONDARY>
```

Nota: los nodos se deben llamar igual, y siempre deben estar arriba

Una vez configurado el replica Set para poder conectarse a Mongo se debe iniciar los servidores con la instrucción

sudo mongod --port <puerto> --dbpath <ruta> --replSet <nombre>

Entre los servidores elegirán cuál nodo es el primario

Importar una BD:

1) Ejecutar el siguiente comando debido que la exportación se realizó con mongodump
sudo mongorestore --host <host> --port <puerto> <ruta>/archivo.bson

```
root@mongo:/usr/bin# mongorestore --host mongo --port 27017  /media/sf_Gabriela/
sudebip/bien.bson
2015-11-26T14:17:52.008-0430    checking for collection data in /media/sf_Gabrie
la/sudebip/bien.bson
```

Crear un ReplicaSet en un ShardCluster

1) Crear el Replica Set, para esta prueba se creó en un mismo servidor.

Servidor: mongodb2
nombre del replica set: rs1
puertos: 27001,27002,27003

Comando:

mongod --port 27001 --logpath /srv/replset/log.rp0 --dbpath /srv/replset/rs2-0 --replSet rs1 --fork

Vista de los 3 mongod corriendo en background:

```
root     23700     1  1 ene19 ?        00:23:01 mongod --port 27001 -
-logpath /srv/mongodb/log.rs0 --dbpath /srv/mongodb/rs0-1 --replSet r
s1 --fork
root     23723     1  1 ene19 ?        00:22:59 mongod --port 27002 -
-logpath /srv/mongodb/log.rs1 --dbpath /srv/mongodb/rs0-2 --replSet r
s1 --fork
root     23746     1  1 ene19 ?        00:22:47 mongod --port 27003 -
-logpath /srv/mongodb/log.rs2 --dbpath /srv/mongodb/rs0-3 --replSet r
```

2) Conectarse con mongo a uno de esos nodos para iniciar el replicaSet.
3) Insertar data en el nodo primario
4) Levantar los servidores de configuración: son 3 servidores y su función es la de decirle a los mongos a qué Replica Set, o shard, tiene que enrutar las peticiones de los clientes

   Servidor: mongodb1
   puertos: 26050,26051,26053

   Comando:

   mongod --port 26050 --configsvr --replSet configReplSet --logpath /srv/mongodb/log.cfg0 --dbpath /srv/mongodb/cfg0 --fork

   Vista de los 3 servidores corriendo en Background

```
root     23576     1  1 ene19 ?        00:19:37 mongod --port 26050 --configsvr -
-replSet configReplSet --logpath /srv/mongodb/log.cfg0 --dbpath /srv/mongodb/cfg0
 --fork
root     23599     1  1 ene19 ?        00:19:06 mongod --port 26051 --configsvr -
-replSet configReplSet --logpath /srv/mongodb/log.cfg1 --dbpath /srv/mongodb/cfg1
 --fork
root     25389     1  1 09:51 ?        00:00:44 mongod --port 26052 --configsvr -
-replSet configReplSet --logpath /srv/mongodb/log.cfg2 --dbpath /srv/mongodb/cfg2
 --fork
```

5) Conectarse a una consola mongo a un servidor e iniciarlo

   mongo --port 26050
   rs.initiate()

6) Añadir los otros dos nodos

   rs.add(“mongodb1:26051”)
   rs.add(“mongodb1:26052”)

7) Iniciar los mongos: solicitan la información requerida por los clientes al shard que la contiene, por esos las app se conectan con los mongos, que son los encargados de enrutar las peticiones a los shards correspondientes.

   servidor:mongodb3
   puerto: 27017

   Comando:

mongos --configdb configReplSet/mongodb1:26050,mongodb1:26051,mongodb1:26052 --chunkSize 1

   Se debe indicar al momento de iniciar los servidores de configuración.

8) Conectarse a mongos

   mongo mongodb3:27017

9) Añadir el shard
   use admin

   sh.addShard("rs1/mongodb2:27001,mongodb2:27002, mongodb2:27003")
   { "shardAdded" : "rs1", "ok" : 1 }

   Se debe suministrar el nombre del replica set y sus nodos

10) Habilitar el shard a nivel de DB

   sh.enableSharding("test")

11) Crear el shard key
   use db
   crear un indice
   db.test_collection.createIndex({number:1})

12) Habilitar el balanceo
   use db
   sh.shardCollection("test.test_collection",{"number":1})

13) Verificar que este el balanceo:

   db.stats()
   db.printShardingStatus()

Nota: abrir los puertos para q los servidores puedan escuchar las peticiones

Comandos Centos 7

firewall-cmd --permanent --zone=public --add-port=27017/tcp
firewall-cmd --reload

mongo --port 27017 -u "gponceleon" -p "123456" --authenticationDatabase "admin"

Hacer conexion de Jmeter con MongoDB (con autenticación) :

1) Bajarse el driver mongo-java-driver-2.13.2.jar (o el más actual)
2) En el mongo source colocar el nombre de usuario, password y bd de autenticacion (admin)
3) Crear en el servidor el rol executeFunctions el cual servira para ejecutar script JavaScript en el servidor (servidores si es un cluster)

   use admin

   db.createRole({ role: "executeFunctions", privileges: [{ resource: { anyResource: true }, actions: ["anyAction"] }], roles: [] })

4) Agregarle el rol al usuario:

db.system.users.update({user:"test.gponce"},{$push:{"roles":{role:"executeFunctions","db":"admin"}}})

o

db.grantRolesToUser("gponceleon", [{ role: "executeFunctions", db: "admin" }])