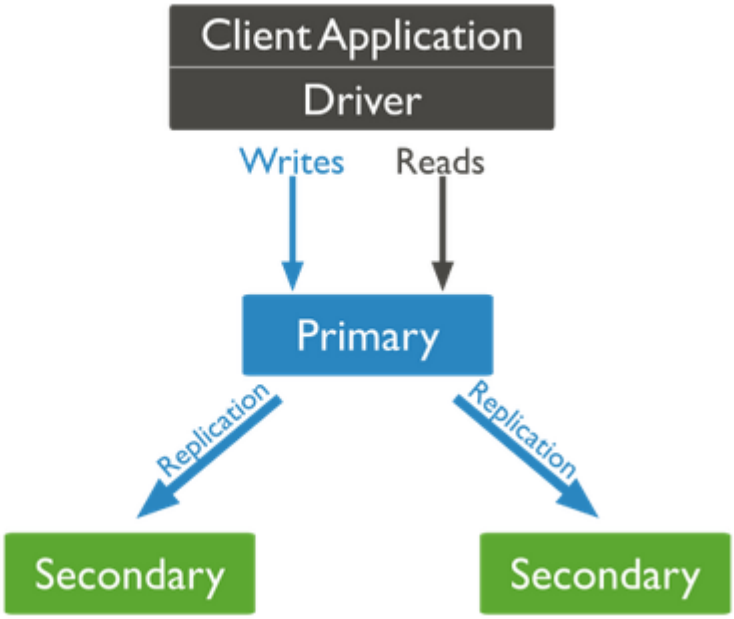
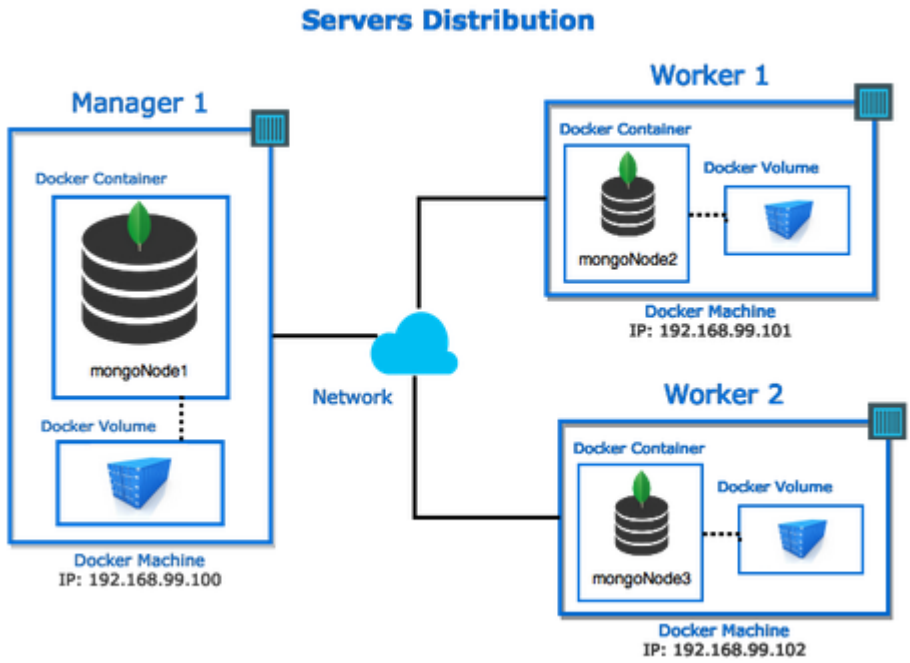
MongoDB distribuido

MongoDB es un sistema de base de datos no relacional (NoSQL) basado en documentos JSON, todo dato dentro de mongo es guardado como un documento JSON con un id único, los documentos pueden contener dentro sí datos atómicos (textos, números) u otros documentos.

MongoDB está diseñado para responder con gran velocidad y mantener una alta disponibilidad de los datos. La gráfica a continuación describe la organización arquitectónica de una aplicación que utiliza una base de datos MongoDB con un nodo primario y dos nodos secundarios que conforman un replica set que garantiza la alta disponibilidad de los datos para la operación de lectura.



A continuación se muestra la misma estructura de cluster pero implementada utilizando contenedores docker.



Instalando docker sin proxy

Ejecute los siguientes comandos para realziar la instalación de docker en su máquina.

1. Actualice el sistema de paquetes.

```
sudo apt-get update
```

2. Instale los certificados.

```
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

3. Descargue los certificados a la máquina local.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
-
```

4. Agregue el repositorio de mongo al sistema de paquetes con el siguiente código.

```
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

5. Actualice el sistema de paquetes nuevamente.

```
sudo apt-get update
```

6. Agregue docker a la política de actualización.

```
apt-cache policy docker-ce
```

7. Instale docker community edition.

```
sudo apt install docker-ce
```

8. Verifique que el servicio de docker esté corriendo.

```
sudo systemctl status docker
```

Deberá observar un punto de color verde al lado del nombre del servicio, para salir de este modo presione la tecla "q".

Configurando proxy Unal para Docker

Si usted se encuentra dentro de la red con proxy unal por favor realice los siguientes pasos para configurar el proxy para docker:

1. Ejecute el siguiente comando

```
sudo mkdir -p /etc/systemd/system/docker.service.d
```

2. Ejecute el siguiente comando para crear un archivo de configuración para el servicio

```
sudo nano /etc/systemd/system/docker.service.d/http-proxy.conf
```

3. Escriba las siguientes líneas dentro del archivo que acaba de crear

```
[Service]
Environment="HTTP_PROXY=http://Instei_bog:continuada@proxy4.unal.edu.co:8080/"
Environment="HTTPS_PROXY=http://Instei_bog:continuada@proxy4.unal.edu.co:8080/"
```

4. Guarde el archivo presionando las teclas CTRL+O.
5. Cierre el archivo presionando CTRL+x.
6. Aplique los cambios ejecutando el siguiente código

```
sudo systemctl daemon-reload
```

7. Reinicie el servicio de docker con el siguiente comando

```
sudo systemctl restart docker
```

Compruebe la configuración de docker con el siguiente comando

```
systemctl show --property=Environment docker
```

Evitar uso de la palabra sudo (opcional)

Para evitar el uso de la palabra sudo cada vez que se ejecuta un comando docker puede utilizar la siguiente instrucción:

```
sudo usermod -aG docker your-user
```

Donde **your-user** es su usuario de la máquina host. **Nota**: Si no realiza este paso necesitará anteponer la palabra sudo a todo comando docker.

Utilizando docker

Una vez instalado docker podremos traer las imagenes de mongo y utilizarlas para desplegar un cluster con los siguientes comandos:

1. Descargue la última imagen de mongo.

```
docker pull mongo
```

El comando *docker pull* traerá la primera y más reciente versión de la imagen solicitada dentro del repositorio de imagenes de contenedores dockerhub.

2. Configuramos una red para nuestro cluster, esto permitirá a los contenedores comunicarse entre sí.

```
docker network create my-mongo-cluster
```

3. Listamos las redes disponibles y verificamos que la red *my-mongo-cluster* se encuentre entre las redes disponibles

```
docker network ls
```

4. Iniciamos nuestro primer contenedor de mongo con el siguiente comando.

```
docker run \
-p 30001:27017 \
--name mongo1 \
--net my-mongo-cluster \
mongo mongod --replSet my-mongo-set
```

El comando **docker run** crea un nuevo contenedor a partir de una imagen. El paramétro **-p puerto_host:puerto_contendedor** permite mapear puertos del contenedor al host. El paramétro **--name nombre-contenedor** permite nombrar los contenedores para referenciarlos facilmente y administyrarlos en el futuro. El comando **--net network** permite conectar al contenedor a una red virtual entre contenedores. La palabra **mongo** corresponde a la imagen de la cual se creará el contenedor, **mongod** es un servicio que se iniciará con la imagen y **--replSet** es un paramétro pasado a dicho servicio y cuyo valor es **my-mongo-set**.

5. Iniciamos los otros dos contenedores en dos terminales separadas

```
docker run \
-p 30002:27017 \
--name mongo2 \
--net my-mongo-cluster \
mongo mongod --replSet my-mongo-set

docker run \
-p 30003:27017 \
--name mongo3 \
--net my-mongo-cluster \
mongo mongod --replSet my-mongo-set
```

Para que los contenedores puedan formar un cluster deben estar dentro del mismo replica-set esto se hace utilizando el mismo nombre de replica set para todos los contenedores, en este caso **my-mongo-set**.

6. Para configurar la replicación en mongo, conéctese a cualquiera de los contenedores que se encuentran corriendo la imagen de mongo, por ejemplo para conectarse al contenedor 1 utilice el siguiente comando:

```
docker exec -it mongo1 mongo
```

La orden **docker exec** se conecta a un contenedor que se encunetre en ejecución. El paramétro **-it** realiza una conexión interactiva con la teletype-terminal (TTY); **mongo1** es el nombre del contenedor al cual te deseas conectar y **mongo** es un comando que se ejecutará en el contenedor al momento de realziar la conexión, en este caso abrirá el shell de mongo.

7. Dentro de la sesión interactiva abierta previamente realizaremos la siguiente configuración:

```
db = (new Mongo('localhost:27017')).getDB('test')
config = {
      "_id" : "my-mongo-set",
      "members" : [
          {
              "_id" : 0,
              "host" : "mongo1:27017"
          },
          {
              "_id" : 1,
              "host" : "mongo2:27017"
          },
          {
              "_id" : 2,
              "host" : "mongo3:27017"
          }
      ]
  }
```

8. Inicie el cluster de replica set con el siguiente comando

```
rs.initiate(config)
```

9. Insertar un documento en la primera replica dataset

```
db.mycollection.insert({name : 'sample'})

db.mycollection.find()
```

10. Iniciamos una nueva conexión a una de las bases de datos secundarias, en este caso mongo2 usnado el comando descrito en el paso 6.

```
db2 = (new Mongo('mongo2:27017')).getDB('test')
db2.setSlaveOk()
db2.mycollection.find()
```

11. Detenga el servidor mongo1 para simular la caida del nodo maestro, con el siguiente comando:

```
docker stop mongo1
```

12. Conéctese al docker mongo3 y comprube que aún es posible realizar lecturas de datos a pesar de que el nodo maestro se encuentre fuera de servicio con el siguiente coamndo:

```
docker exec -it mongo3 mongo
```

```
db2 = (new Mongo('mongo3:27017')).getDB('test')
db3.setSlaveOK()
db3.mycollection.find()
```

13. Cree un nuevo contenedor de mongo utilizando el siguiente comando:

```
sudo docker run -d -p 30006:27017 -v mongo-data:/data/db --name mongo4
mongo mongod
```

```
docker exec -it mongo4 bash
```

14. Cree una carpeta dentro del contenedor para alojar los datos a ingresar con el siguiente comando:

```
mkdir import-data
```

15. Descargue el archivo data.json y copielo al contenedor utilizando el siguiente comando en otra terminal:

```
docker cp Descargas/data.json mongo4:/import-data
```

El comando **docker cp** copiará archivos desde y hacia los contenedores. **Descargas/data.json** es la ruta de origen del archivo y **mongo4:/import-data** es la ruta destino, tenga en cuenta que esto puede variar para su

caso.

16. Regrese a la terminal donde se encuentra conectado al contenedor y verifice que el archivo se encuentra dentro en la carpeta /import-data, con el siguiente comando.

```
cd import-data
ls
```

17. Una vez comprobado que el archivo se encuentra en la carpeta utilice el siguiente comando para cargar los datos:

```
mongoimport --db diplomado --collection restaurantes --drop --file
data.json
```

Referencias

configuración proxy

Configuración replica set