Universidad Nacional Autónoma de México
Instituto de Investigaciones en Matemáticas
Aplicadas y en Sistemas (IIMAS)
Licenciatura en Ciencia de Datos

BASES DE DATOS NO ESTRUCTURADAS
EJERCICIO PRÁCTICO 07

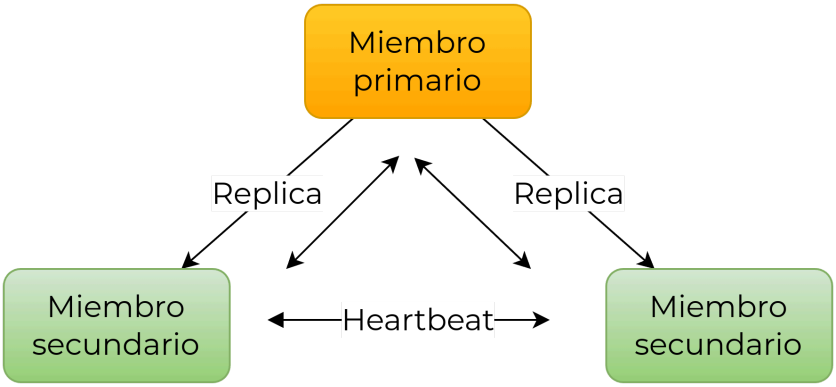
1. REPLICACIÓN DE DATOS EN MONGODB

1.1. Replica sets

MongoDB implementa replicación de datos a través del concepto de *Replica Sets*, formada por un conjunto de nodos o servidores llamados miembros. La réplica ***debe*** contar con un miembro *primario* y al menos un miembro *secundario*.

El miembro primario representa al punto de acceso principal para procesar transacciones en la réplica y es el ***único*** nodo que puede aceptar operaciones de escritura. Los cambios realizados por una transacción son escritos en el llamado ***primary oplog*** (operations log) el cual es copiado en los miembros secundarios.

Por default las aplicaciones leen y escriben únicamente en el nodo primario. Si el nodo primario falla, se aplica un algoritmo de selección de un miembro secundario para sustituir al miembro secundario. Cada *replica set* puede tener hasta de 50 miembros, máximo 7 de ellos pueden participar en la elección de un miembro principal en caso de falla.



1.2. Configuración de un Replica set

Los siguientes pasos muestran el proceso para configurar e iniciar una *replica set* de 3 miembros.

1.2.1. Configuración con Docker compose

A. Dentro del directorio `/unam-bd-ne` crear la siguiente estructura de directorios: `ejercicio-practico-07/mongodb-cluster/scripts`
B. Revisar el siguiente archivo composer.yaml. Generar uno similar en la carpeta `mongodb-cluster`. Observar las configuraciones: se crean 3 contenedores Docker con MongoDB instalado, puertos, etc. El nombre del replica set y de los miembros incluyen las iniciales del alumno (nombre y apellidos), sustituir el texto resaltado por los valores correspondientes.

```
services:
  jrc-mongo1:
    image: mongo:7
    hostname: jrc-mongo1
    container_name: jrc-mongo1
    ports:
      - 27017:27017
    entrypoint: ["mongod", "--replSet", "jrc-replica-set", "--bind_ip", "localhost,jrc-mongo1"]
  jrc-mongo2:
    image: mongo:7
    hostname: jrc-mongo2
    container_name: jrc-mongo2
    ports:
      - 27018:27017
    entrypoint: ["mongod", "--replSet", "jrc-replica-set", "--bind_ip", "localhost,jrc-mongo2"]
  jrc-mongo3:
    image: mongo:7
    hostname: jrc-mongo3
    container_name: jrc-mongo3
    ports:
      - 27019:27017
    entrypoint: ["mongod", "--replSet", "jrc-replica-set", "--bind_ip", "localhost,jrc-mongo3"]
  mongosetup:
    image: mongo:7
    depends_on:
      - jrc-mongo1
      - jrc-mongo2
      - jrc-mongo3
    volumes:
      - ./scripts:/scripts
    restart: "no"
    entrypoint: [ "bash", "/scripts/mongo-setup.sh"]
```

C. Observar que el código anterior hace referencia al archivo `mongo-setup.sh` el cual deberá crearse dentro de la carpeta `scripts` con el siguiente contenido. No olvidar sustituir las iniciales.

```
#!/bin/bash
sleep 10

mongosh --host jrc-mongo1:27017 <<EOF
  var cfg = {
    "_id": "jrc-replica-set",
    "version": 1,
    "members": [
      {
        "_id": 0,
        "host": "jrc-mongo1:27017",
        "priority": 2
      },
      {
        "_id": 1,
        "host": "jrc-mongo2:27017",
        "priority": 0
      },
      {
        "_id": 2,
        "host": "jrc-mongo3:27017",
        "priority": 0
      }
    ]
  };
  rs.initiate(cfg);
EOF
```

Básicamente el script crea un documento JSON que contiene todas las configuraciones requeridas para crear el replica set. Posteriormente, en `mongosh` se invoca a la función `rs.initiate`

D. cambiarse al directorio `mongodb-cluster` y ejecutar el siguiente comando para crear e iniciar los contenedores, ***C1. Incluir en la entrega*** la salida de ejecución. Si los contenedores ya existen, solo los inicia.

```
docker compose up --wait
```

E. Ejecutar el siguiente comando para verificar su correcta creación y ejecución en cada uno de los contenedores. En el ejemplo se hace uso del contenedor 1 ***C2. Incluir en la entrega*** la salida de ejecución por cada contenedor.

```
docker exec -it jrc-mongo1 mongosh --eval "rs.status()"
```

1.2.2. Configurar el archivo mongod.conf

Configurar el archivo `/etc/mongod.conf` en cada contenedor. Esta configuración permitirá realizar la comunicación entre los nodos del *replica set*.

- Acceder manualmente ***a cada contenedor*** empleando el siguiente ejemplo que hace uso del nodo 1

```
docker exec -it  jrc-mongo1  bash
```

- Por default el archivo se llama `mongod.conf.orig`, se requiere renombrar el archivo. Ejecutar las siguientes instrucciones

```
cd /etc
mv mongod.conf.orig mongod.conf
```

- Para editar el archivo se requiere instalar algún editor. Ejecutar las siguientes instrucciones para instalar nano y algunas otras dependencias

```
apt-get update
apt-get upgrade
apt-get install nano
apt-get install systemctl

#abrir el archivo
nano /etc/mongod.conf
```

- Modificar el contenido del archivo en las siguientes 2 secciones, similar al siguiente ejemplo. Notar que al parámetro `bindIp` se le agrega el nombre del host del contenedor `<iniciales>-mongo<N>`. No olvidar actualizar el valor de `N` en cada contenedor. Es importante cuidar la indentación de 2 espacios, notar que se requieren 2 espacios después del parámetro `replSetName`.

```
# network interfaces
net:
  port: 27017
  bindIp: 127.0.0.1,jrc-mongo1

#replication:
  replSetName:  "jrc-replica-set"
```

- Reiniciar el servicio de mongo

```
systemctl restart mongod
```

1.3. Crear una BD con replicación

Para probar las configuraciones anteriores realizar el siguiente procedimiento.

A. Iniciar el shell de mongo en cada uno de los contenedores. Por ejemplo

```
docker exec -it  jrc-mongo1  bash
mongosh
```

B. Notar la forma del prompt en los 3 contenedores:

```
#contenedor 1
jrc-replica-set [direct: primary] test>
#contenedor 2
jrc-replica-set [direct: secondary] test>
#contenedor 3
jrc-replica-set [direct: secondary] test>
```

- Notar que uno de los 3 contenedores tiene el rol de miembro primario. En este miembro se deberá realizar la creación de una base de datos de prueba.

C. Identificar el miembro primario, crear una nueva base de datos, no olvidar especificar las iniciales.

```
use jrc-test-cluster-db
```

D. Crear una colección cualquiera e insertar un documento

```
db.libros.insertOne({nombre:'El principito'});
```

E. Conectarse manualmente a cada uno de los otros 2 nodos. Validar que exista tanto la base de datos, colección y documento creado. ***C3. Incluir en la entrega*** las pantallas que muestran la existencia del documento en los 3 miembros.
F. Intentar modificar el documento en un miembro secundario. ***C4. Incluir en la entrega*** el resultado obtenido, explicar.

1.4. Contenido de la entrega

A. Elementos comunes de los ejercicios prácticos.
B. C1. Salida de ejecución del comando `docker compose` que permite la creación del cluster con mongodb

C. C2. Salida de ejecución del comando `docker exec` que permite validar la correcta creación del cluster
D. C3. Pantallas que muestran el resultado de replicación
E. C4. Pantalla que muestra el resultado de la actualización así como su explicación