Actividad 2 – Casos de pruebas

Reybert Andrés Peñuela Sepúlveda

Corporación universitaria Iberoamericana

Facultad de Ingeniería de software

Bases de datos avanzadas

Profesor William Ruiz

Cali, Colombia

03 de abril del 2023

Casos de pruebas relacionados con la disponibilidad y redundancia requeridas

Caso de prueba 1: Comprobar la sincronización de datos entre los miembros del conjunto de réplicas

Paso 1: Inserta un equipo en el conjunto de réplicas primario

Conéctate al shell de MongoDB en el nodo primario y selecciona la base de datos del torneo de fútbol (reemplaza "mongo1" si tu nodo primario tiene un nombre diferente):

```
docker exec -it mongo1 mongo -u admin -p adminpass --authenticationDatabase admin
```

Usa la base de datos e inserta un equipo:

```
use torneoFutbol;
db.equipos.insert({
  nombre: "Equipo A",
  pais: "País A",
  entrenador: {
    nombre: "Entrenador",
    apellido: "Apellido",
    edad: 40
  },
  jugadores: [
    {
      nombre: "Jugador",
      apellido: "Uno",
      edad: 25,
      posicion: "Delantero"
    },
    {
      nombre: "Jugador",
      apellido: "Dos",
      edad: 27,
      posicion: "Defensa"
    }
  ]
});
```

Paso 2: Verifica que el equipo se haya replicado en un nodo secundario

Conéctate al shell de MongoDB en un nodo secundario (reemplaza "mongo2" si tu nodo secundario tiene un nombre diferente):

```
docker exec -it mongo2 mongo -u admin -p adminpass --authenticationDatabase admin
```

Ejecuta el siguiente comando para permitir lecturas en el nodo secundario:

```
rs.slaveOk();
```

Verifica que el equipo se haya replicado correctamente:

```
use torneoFutbol;
db.equipos.find();
```

Caso de prueba 2: Comprobar la replicación de actualizaciones de datos

Paso 1: Actualiza un registro en el conjunto de réplicas primario

Conéctate al shell de MongoDB en el nodo primario y selecciona la base de datos del torneo de fútbol. Luego, actualiza el país del "Equipo A":

```
use torneoFutbol;
db.equipos.updateOne({nombre: "Equipo A"}, {$set: {pais: "Nuevo País A"}});
```

Paso 2: Verifica que la actualización se haya replicado en un nodo secundario

Conéctate al shell de MongoDB en un nodo secundario y ejecuta el siguiente comando para permitir lecturas:

```
rs.slaveOk();
```

Verifica que la actualización se haya replicado correctamente:

```
use torneoFutbol;
db.equipos.find({nombre: "Equipo A"});
```

Caso de prueba 3: Comprobar la conmutación por error (failover) en caso de fallo del nodo primario

Paso 1: Detén el contenedor del nodo primario

Identifica el nombre del contenedor del nodo primario (por ejemplo, "mongo1") y detén el contenedor usando Docker:

```
docker stop mongo1
```

Paso 2: Verifica que un nodo secundario asuma el rol de primario

Ejecuta el siguiente comando en la terminal (no en el shell de MongoDB) para ver el estado de los nodos:

```
docker exec -it mongo2 mongo -u admin -p adminpass --authenticationDatabase admin
--eval "rs.status()"
```

Revisa la salida y confirma que uno de los nodos secundarios ahora tiene el rol de primario (el campo "stateStr" debería mostrar "PRIMARY").

Paso 3: Conéctate al nuevo nodo primario y verifica la disponibilidad de los datos

Conéctate al shell de MongoDB en el nuevo nodo primario y verifica que los datos estén disponibles:

```
docker exec -it mongo2 mongo -u admin -p adminpass --authenticationDatabase admin
```

Selecciona la base de datos del torneo de fútbol y consulta los equipos:

```
use torneoFutbol;
db.equipos.find();
```

Confirma que los datos estén disponibles en el nuevo nodo primario.