

Materia Bases de datos avanzadas

PROFESOR: WILLIAM RUIZ

Universidad Iberoamericana

Ingeniería en Ciencia de Datos Actividad 2 - Conceptos y
comandos básicos de la replicación en bases de datos
NoSQL

Nelson Fernando Nopssa Castro

ID 100169351

Documento de Casos de Pruebas: Verificación del Mecanismo de Replicación

Objetivo

Verificar que el mecanismo de replicación implementado cumple con los requerimientos de redundancia y disponibilidad 24x7 para el caso planteado en la primera actividad.

Caso de Prueba 1: Verificación de la Creación del Clúster de Réplica

Descripción: Confirmar que el clúster de réplica se ha creado correctamente con al menos tres nodos.

Pasos:

Ejecutar los comandos de configuración del clúster de réplica:

F1MiejReplicaSet=new ReplSetTest({name:"MireplicaSet",nodes:3})

```
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c339f49f-b20c-4dbd-abb0-02f99e2013c1") }
MongoDB server version: 4.2.25
Server has startup warnings:
2024-05-23T17:40:16.860-0500 I CONTROL [initandlisten]
2024-05-23T17:40:16.861-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2024-05-23T17:40:16.861-0500 I CONTROL [initandlisten] **          Read and write access to data and configuration is unrestricted.
2024-05-23T17:40:16.861-0500 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> F1ReplicaSet=new ReplSetTest({name:"MireplicaSet",nodes:3})
```

F1MiejReplicaSet.startSet()

```
    return master;
  },
  "name" : "MireplicaSet",
  "useHostName" : true,
  "host" : "DESKTOP-D14HD7H",
  "oplogSize" : 40,
  "useSeedList" : false,
  "keyFile" : undefined,
  "protocolVersion" : undefined,
  "waitForKeys" : undefined,
  "nodeOptions" : {
    "n0" : undefined,
    "n1" : undefined,
    "n2" : undefined
  },
  "nodes" : [ ],
  "ports" : [
    20000,
    20001,
    20002
  ]
}
F1ReplicaSet.startSet()
```

F1MieJReplicaSet.initiate()

```
2024-05-24T18:45:32.252-0500 I REPL [initandlisten] initialized the rollback id to 1
2024-05-24T18:45:32.252-0500 I REPL [initandlisten] Did not find local replica set configuration document at startup; NoMatchingDocument: Did not find replica set configuration document in local .system.replset
2024-05-24T18:45:32.254-0500 I NETWORK [listener] Listening on 0.0.0.0
2024-05-24T18:45:32.254-0500 I NETWORK [listener] waiting for connections on port 20002
2024-05-24T18:45:32.666-0500 I NETWORK [listener] connection accepted from 127.0.0.1:55866 #1 (1 connection now open)
2024-05-24T18:45:32.677-0500 I NETWORK [conn1] received client metadata from 127.0.0.1:55866 c
conn1: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "4.2.25" }, os: { type: "Windows", name: "Microsoft Windows 10", architecture: "x86_64", version: "10.0 (build 22000)" } }
[
  connection to DESKTOP-D14HD7H:20000,
  connection to DESKTOP-D14HD7H:20001,
  connection to DESKTOP-D14HD7H:20002
]
[
  connection to DESKTOP-D14HD7H:20000,
  connection to DESKTOP-D14HD7H:20001,
  connection to DESKTOP-D14HD7H:20002
]
> F1ReplicaSet.initiate()
```

conn=new Mongo("DESKTOP-D14HD7H:20000")

```
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("aed5eade-0558-4132-b1eb-81545a357a5e") }
MongoDB server version: 4.2.25
Server has startup warnings:
2024-05-23T17:40:16.860-0500 I CONTROL [initandlisten]
2024-05-23T17:40:16.861-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2024-05-23T17:40:16.861-0500 I CONTROL [initandlisten] **           Read and write access to data and configuration is unrestricted.
2024-05-23T17:40:16.861-0500 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> conn=new Mongo("DESKTOP-D14HD7H:20000")
```

```
testDB=conn.getDB("Formula1")
```

```
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> conn=new Mongo("DESKTOP-D14HD7H:28000")
connection to DESKTOP-D14HD7H:28000
> testDB=conn.getDB("Formula1")
Formula1
>
```

```
testDB.isMaster()
```

```
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> conn=new Mongo("DESKTOP-D14HD7H:28000")
connection to DESKTOP-D14HD7H:28000
> testDB=conn.getDB("Formula1")
Formula1
> testDB.isMaster()
{
  "msg": "ok",
  "code": 0,
  "isMaster": true,
  "primary": "DESKTOP-D14HD7H",
  "secondary": null,
  "isWritable": true,
  "readOnly": false,
  "writeConcern": {
    "w": 1,
    "wtimeout": 0,
    "fsync": false,
    "journal": true
  },
  "localTime": "2020-07-14T14:00:00Z",
  "maxBsonSize": 16777216,
  "maxMessageSizeBytes": 48000000,
  "compression": {
    "compressor": "snappy"
  },
  "readOnlyReplicaSet": false,
  "replicaSet": "rs0"
}
```

```
testDB.Circuitos.insert({ "nombre": "Interlagos",
  "pais": "Brasil",
  "ciudad": "São Paulo",
  "vueltas": 71
})
```

```
> testDB.Circuitos.insert({
...   "nombre": "Suzuka Circuit",
...   "pais": "Japón",
...   "ciudad": "Suzuka",
...   "vueltas": 53
... })
```

```
testDB.Circuitos.count()
```

```
> testDB.Circuitos.insert({
...   "nombre": "Suzuka Circuit",
...   "pais": "Japón",
...   "ciudad": "Suzuka",
...   "vueltas": 53
... })
WriteResult({ "nInserted" : 1 })
> testDB.Circuitos.count()
1
>
```

```
testDB.Circuitos.find().pretty()
```

```
> testDB.Circuitos.find().pretty()
{
  "_id" : ObjectId("665129d930e82aed367d4a27"),
  "nombre" : "Suzuka Circuit",
  "pais" : "Japón",
  "ciudad" : "Suzuka",
  "vueltas" : 53
}
{
  "_id" : ObjectId("66512a5330e82aed367d4a28"),
  "nombre" : "Circuit of the Americas",
  "pais" : "Estados Unidos",
  "ciudad" : "Austin",
  "vueltas" : 56
}
```

```
connSecondary= new Mongo("DESKTOP-D14HD7H:20001")
```

```
}
> connSecondary= new Mongo("DESKTOP-D14HD7H:20001")
connection to DESKTOP-D14HD7H:20001
>
=
```

```
secondaryTestDB=connSecondary.getDB("Formula1")
```

```
/
> connSecondary= new Mongo("DESKTOP-D14HD7H:20001")
connection to DESKTOP-D14HD7H:20001
> secondaryTestDB=connSecondary.getDB("Formula1")
Formula1
>
```

```
secondaryTestDB.isMaster()
```

```
connection to DESKTOP-D14HD7H:20001
> secondaryTestDB=connSecondary.getDB("Formula1")
Formula1
> secondaryTestDB.isMaster()
{
```

```
  "setName" : "MircplicaSet",
  "setVersion" : 2,
  "ismaster" : false,
  "secondary" : true,
  "primary" : "DESKTOP-D14HD7H:20000",
  "me" : "DESKTOP-D14HD7H:20001",
  "lastWrite" : {
```

```
secondaryTestDB.Circuitos.count() ojo si no puede
```

```
_getErrorWithCode@src/mongo/shell/utils.js:25:13
DBQuery.prototype.count@src/mongo/shell/query.js:376:11
DBCollection.prototype.count@src/mongo/shell/collection.js:1401:12
@(shell):1:1
> connSecondary.setSecondaryOk()
> secondaryTestDB.nombreCollection.count()
1
```

connSecondary.setSecondaryOk()

```
_getErrorWithCode@src/mongo/shell/utils.js:25:13
DBQuery.prototype.count@src/mongo/shell/query.js:376:11
DBCollection.prototype.count@src/mongo/shell/collection.js:1401:12
@(shell):1:1
> connSecondary.setSecondaryOk()
>
```

secondaryTestDB.nombrecollection.count()

```
0
> secondaryTestDB.Circuitos.count()
2
> secondaryTestDB.Circuitos.find().pretty()
{
  "_id" : ObjectId("665129d930e82aed367d4a27"),
  "nombre" : "Suzuka Circuit",
  "pais" : "Japón",
  "ciudad" : "Suzuka",
  "vueltas" : 53
}
{
  "_id" : ObjectId("66512a5330e82aed367d4a28"),
  "nombre" : "Circuit of the Americas",
  "pais" : "Estados Unidos",
  "ciudad" : "Austin",
  "vueltas" : 56
}
```

secondaryTestDB.Circuitos.find().pretty()

```
0
> secondaryTestDB.Circuitos.count()
2
> secondaryTestDB.Circuitos.find().pretty()
{
  "_id" : ObjectId("665129d930e82aed367d4a27"),
  "nombre" : "Suzuka Circuit",
  "pais" : "Japón",
  "ciudad" : "Suzuka",
  "vueltas" : 53
}
{
  "_id" : ObjectId("66512a5330e82aed367d4a28"),
  "nombre" : "Circuit of the Americas",
  "pais" : "Estados Unidos",
  "ciudad" : "Austin",
  "vueltas" : 56
}
```



```
connPrimary= new Mongo("localhost:20000")
```

```
> connPrimary= new Mongo("localhost:20000")
connection to localhost:20000
>
```

```
primaryDB=connPrimary.getDB("Formula1")
```

```
> connPrimary= new Mongo("localhost:20000")
connection to localhost:20000
> primaryDB=connPrimary.getDB("Formula1")
Formula1
> _
```

```
primaryDB.isMaster()
```

```
{
  "setName" : "MireplicaSet",
  "setVersion" : 2,
  "ismaster" : true,
  "secondary" : false,
  "primary" : "DESKTOP-D14HD7H:20000",
  "me" : "DESKTOP-D14HD7H:20000",
  "electionId" : ObjectId("7fffffff0000000000000001"),
  "lastWrite" : {
    "opTime" : {
      "ts" : Timestamp(1716595283, 1),
      "t" : NumberLong(1)
    },
    "lastWriteDate" : ISODate("2024-05-25T00:01:23Z"),
    "majorityOpTime" : {
      "ts" : Timestamp(1716595283, 1),
      "t" : NumberLong(1)
    },
    "majorityWriteDate" : ISODate("2024-05-25T00:01:23Z")
  }
}
```

```
primaryDB.adminCommand({shutdown:1})
```

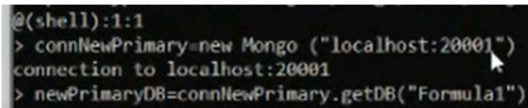
```
{
  "operationTime" : Timestamp(1716595283, 1)
}
> primaryDB.adminCommand({shutdown:1})
```

```
connNewPrimary=new Mongo ("localhost:20001")
```

```
> primaryDB.adminCommand({shutdown:1})
2024-05-24T19:53:14.285-0500 I NETWORK [js] DBClientConnection failed to receive message from localhost:20000 - HostUnreachable: Connection reset by peer
2024-05-24T19:53:14.288-0500 E QUERY [js] uncaught exception: Error: error doing query: failed: network error while attempting to run command 'shutdown' on host 'localhost:20000' :
DB.prototype.runCommand@src/mongo/shell/db.js:169:19
DB.prototype.adminCommand@src/mongo/shell/db.js:187:12
@(shell):1:1
> connNewPrimary=new Mongo ("localhost:20001")
```

```
@(shell):1:1
> connNewPrimary=new Mongo ("localhost:20001")
connection to localhost:20001
>
```

```
newPrimaryDB=connNewPrimary.getDB("Formula1")
```



```
@(<shell>):1:1
> connNewPrimary=new Mongo ("localhost:20001")
connection to localhost:20001
> newPrimaryDB=connNewPrimary.getDB("Formula1")
```

```
newPrimaryDB.isMaster()
```

Ejecutar el comando `F1ReplicaSet.status()` para verificar el estado del clúster.

Resultado Esperado: El comando debe mostrar tres nodos en el clúster de réplica.

Caso de Prueba 2: Verificación de la Replicación de Datos

Descripción: Comprobar que los datos se están replicando correctamente entre los nodos del clúster.

Pasos:

Insertar un nuevo registro en el nodo primario utilizando los siguientes comandos:

```
conn=new Mongo("DESKTOP-D14HD7H:20000")
```

```
testDB=conn.getDB("TorneoF1")
```

```
testDB.Circuitos.insert({ "nombre": "Interlagos", "pais": "Brasil", "ciudad": "São Paulo", "vueltas":
71 })
```

Conectar a cada nodo secundario y verificar la presencia del nuevo registro utilizando los siguientes comandos:

```
connSecondary= new Mongo("DESKTOP-D14HD7H:20001")
```

```
secondaryTestDB=connSecondary.getDB("TorneoF1")
```

```
secondaryTestDB.Circuitos.find().pretty()
```

Resultado Esperado: El nuevo registro debe estar presente en todos los nodos secundarios.

Caso de Prueba 3: Verificación de la Alta Disponibilidad

Descripción: Validar que el sistema mantiene la disponibilidad incluso en caso de fallo del nodo primario.

Pasos:

Simular un fallo del nodo primario mediante los siguientes comandos:

```
connPrimary= new Mongo("localhost:20000")
```

```
primaryDB=connPrimary.getDB("Formula1")
```

```
primaryDB.adminCommand({shutdown:1})
```


Verificar que uno de los nodos secundarios sea promovido automáticamente a nodo primario mediante los siguientes comandos:

```
connNewPrimary=new Mongo ("localhost:20001")  
newPrimaryDB=connNewPrimary.getDB("Formula1")  
newPrimaryDB.isMaster()
```

Resultado Esperado: El sistema debe seguir estando disponible y aceptando operaciones incluso después del fallo del nodo primario.

Caso de Prueba 4: Verificación del Monitoreo de Estado

Descripción: Confirmar que el monitoreo continuo del estado de los nodos está activo.

Pasos:

Verificar que el mecanismo de heartbeat de MongoDB está funcionando correctamente.

Monitorear el estado de los nodos durante un período de tiempo.

Resultado Esperado: El monitoreo debe detectar y manejar cualquier fallo rápidamente, manteniendo la disponibilidad del sistema.

Caso de Prueba 5: Verificación de Operaciones Sin Contratiempos

Descripción: Asegurar que las operaciones de ingreso, consulta y modificación de datos se realizan sin interrupciones.

Pasos:

Realizar operaciones de inserción, consulta y modificación en la base de datos.

Verificar que todas las operaciones se completan con éxito.

Resultado Esperado: Todas las operaciones deben completarse sin errores y con un tiempo de respuesta adecuado.

