Documento de Requerimientos No Funcionales y Pruebas para la Actividad de Replicación

Objetivo

Garantizar la redundancia y disponibilidad 24x7 de la base de datos del torneo de fútbol Copa América, asegurando que los datos estén siempre accesibles y consistentes incluso en caso de fallos de servidor.

Requerimientos No Funcionales

1. **Disponibilidad**: La base de datos debe estar disponible las 24 horas del día, los 7 días de la semana.
2. **Redundancia**: Debe existir una copia exacta de los datos en al menos tres nodos para garantizar la redundancia.
3. **Consistencia**: Los datos replicados en todos los nodos deben estar siempre consistentes con los datos del nodo primario.
4. **Tolerancia a Fallos**: En caso de fallo del nodo primario, uno de los nodos secundarios debe asumir automáticamente el rol de primario sin interrumpir el servicio.
5. **Escalabilidad**: La configuración debe permitir la fácil adición de más nodos en el futuro sin afectar la disponibilidad y consistencia de los datos.

Estrategia de Replicación

Definición de la Estrategia

La estrategia de replicación seleccionada es la configuración de un conjunto de réplicas (Replica Set) de MongoDB con tres nodos: un nodo primario y dos nodos secundarios. El nodo primario manejará todas las operaciones de escritura, mientras que los nodos secundarios replicarán los datos de forma asíncrona.

Comandos para Crear el Entorno de Replicación

1. Configurar Mongo en Windows para usar Replicación:
    - Se debe parar el servicio de MongoDB en el sistema de Windows

        net stop MongoDB

    - Se debe editar el archivo mongod.cfg en la ruta C:\Program Files\MongoDB\Server\<version>\bin\mongod.cfg y agregar las siguientes líneas:

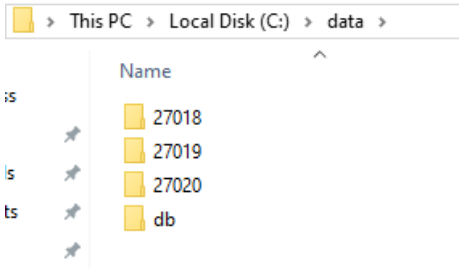
replication:

replSetName: "rs0"

- Iniciar de nuevo el servicio de MongoDB

  net start MongoDB

- Crear las carpetas para los datos de cada nodo



- Inicializar los servidores de Mongod para poder conectarse con cada nodo creado en este documento

  mongod --port 27018 --dbpath "C:\data\27018" --replSet rs0

  mongod --port 27019 --dbpath "C:\data\27019" --replSet rs0

  mongod --port 27020 --dbpath "C:\data\27020" --replSet rs0

- Iniciar Mongosh para poder escribir comandos a la base de datos

  cd C:\Users\Programacion\AppData\Local\Programs\mongosh

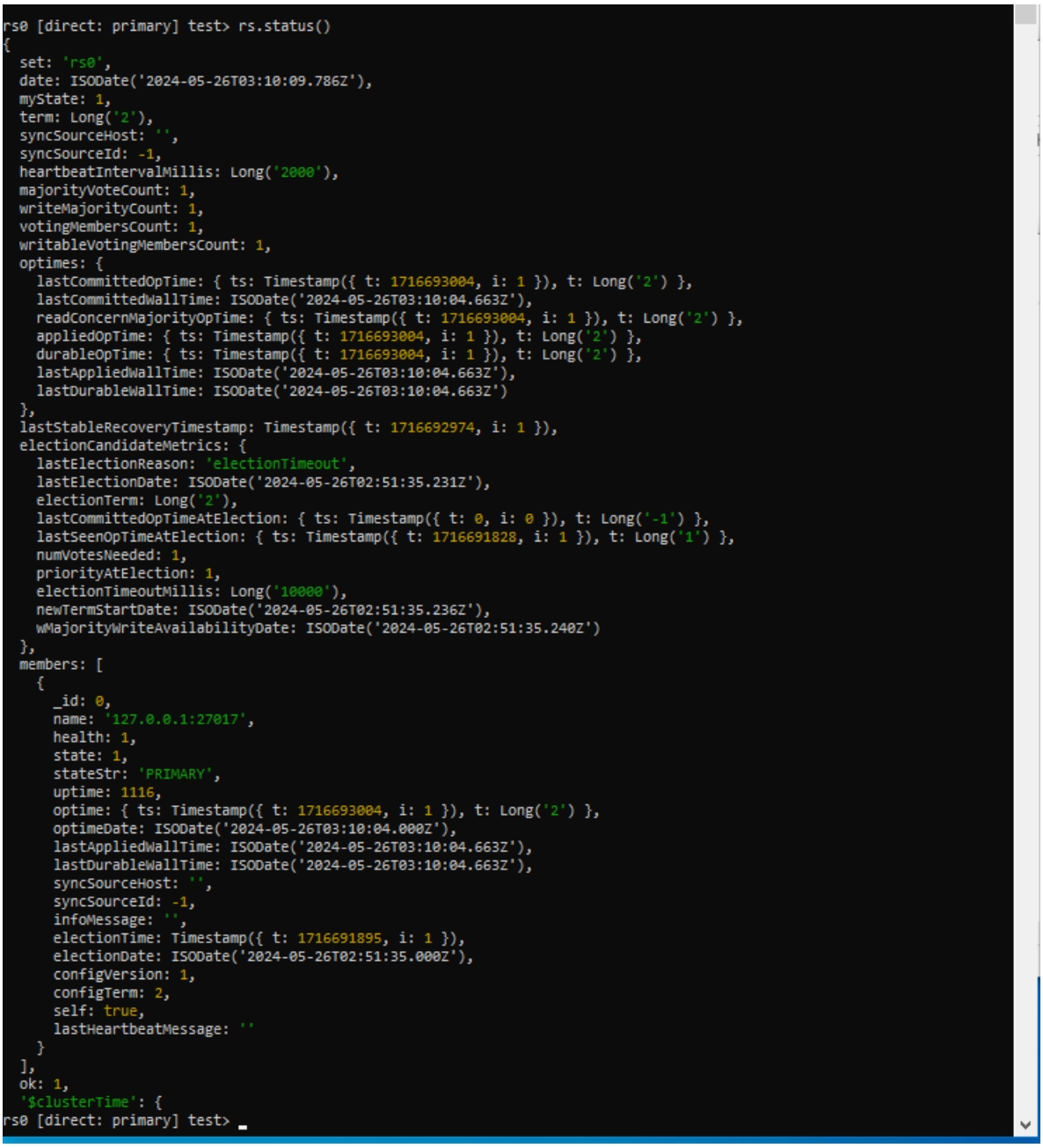
  .\mongosh.exe

2. **Comprobar si se está ejecutando Mongos en replSet**

El comando

rs.status()

Muestra lo siguiente por lo que podemos concluir que ya se tiene el sistema de replicación disponible para ser utilizado:



3. **Configurar los Nodos Secundarios**

Con los comandos

rs.add("localhost:27018")

rs.add("localhost:27019")

Se generan 2 nuevos nodos secundarios.

```
rs0 [direct: primary] test> rs.add("localhost:27018")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1716693317, i: 2 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1716693317, i: 2 })
}
rs0 [direct: primary] test> rs.add("localhost:27019")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1716693342, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1716693342, i: 1 })
}
rs0 [direct: primary] test>
```

4. **Verificar el status del sistema**

rs.status()

```
rs0 [direct: primary] test> rs.status()
{
  set: 'rs0',
  date: ISODate('2024-05-26T03:27:57.309Z'),
  myState: 1,
  term: Long('3'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1716694072, i: 1 }), t: Long('3') },
    lastCommittedWallTime: ISODate('2024-05-26T03:27:52.912Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1716694072, i: 1 }), t: Long('3') },
    appliedOpTime: { ts: Timestamp({ t: 1716694072, i: 1 }), t: Long('3') },
    durableOpTime: { ts: Timestamp({ t: 1716694072, i: 1 }), t: Long('3') },
    lastAppliedWallTime: ISODate('2024-05-26T03:27:52.912Z'),
    lastDurableWallTime: ISODate('2024-05-26T03:27:52.912Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1716693793, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-05-26T03:26:52.251Z'),
    electionTerm: Long('3'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1716693793, i: 1 }), t: Long('2') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1716693805, i: 1 }), t: Long('2') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2024-05-26T03:26:58.898Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-05-26T03:26:58.928Z')
  },
```

```
  members: [
    {
      _id: 0,
      name: '127.0.0.1:27017',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 2184,
      optime: { ts: Timestamp({ t: 1716694072, i: 1 }), t: Long('3') },
      optimeDate: ISODate('2024-05-26T03:27:52.000Z'),
      lastAppliedWallTime: ISODate('2024-05-26T03:27:52.912Z'),
      lastDurableWallTime: ISODate('2024-05-26T03:27:52.912Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: 'Could not find member to sync from',
      electionTime: Timestamp({ t: 1716694012, i: 1 }),
      electionDate: ISODate('2024-05-26T03:26:52.000Z'),
      configVersion: 5,
      configTerm: 3,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: 'localhost:27018',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 69,
      optime: { ts: Timestamp({ t: 1716694072, i: 1 }), t: Long('3') },
      optimeDurable: { ts: Timestamp({ t: 1716694072, i: 1 }), t: Long('3') },
      optimeDate: ISODate('2024-05-26T03:27:52.000Z'),
      optimeDurableDate: ISODate('2024-05-26T03:27:52.000Z'),
      lastAppliedWallTime: ISODate('2024-05-26T03:27:52.912Z'),
      lastDurableWallTime: ISODate('2024-05-26T03:27:52.912Z'),
      lastHeartbeat: ISODate('2024-05-26T03:27:56.130Z'),
      lastHeartbeatRecv: ISODate('2024-05-26T03:27:56.170Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: '127.0.0.1:27017',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 5,
      configTerm: 3
    },
```

```
    {
      _id: 2,
      name: 'localhost:27019',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 21,
      optime: { ts: Timestamp({ t: 1716694072, i: 1 }), t: Long('3') },
      optimeDurable: { ts: Timestamp({ t: 1716694072, i: 1 }), t: Long('3') },
      optimeDate: ISODate('2024-05-26T03:27:52.000Z'),
      optimeDurableDate: ISODate('2024-05-26T03:27:52.000Z'),
      lastAppliedWallTime: ISODate('2024-05-26T03:27:52.912Z'),
      lastDurableWallTime: ISODate('2024-05-26T03:27:52.912Z'),
      lastHeartbeat: ISODate('2024-05-26T03:27:56.130Z'),
      lastHeartbeatRecv: ISODate('2024-05-26T03:27:56.661Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'localhost:27018',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 5,
      configTerm: 3
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1716694072, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1716694072, i: 1 })
}
rs0 [direct: primary] test>
```

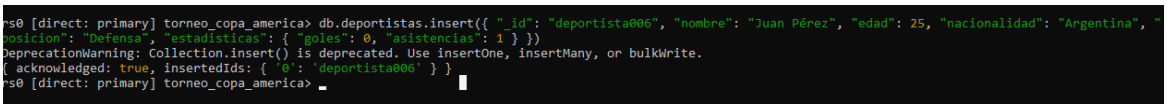
5. **Insertar Datos Iniciales en el Nodo Primario**

```
use torneo_copa_america


db.createCollection("deportistas")

db.createCollection("equipos")

db.createCollection("entrenadores")

db.createCollection("arbitros")

db.createCollection("encuentros_deportivos")

db.createCollection("resultados")

db.createCollection("tabla_de_posiciones")

# usando mongo import

mongoimport --host rs0/localhost:27017 --db torneo_copa_america --collection deportistas --file "C:\Users\Programacion\Documents\MongoDB\deportistas.json" --jsonArray

mongoimport --host rs0/localhost:27017 --db torneo_copa_america --collection equipos --file "C:\Users\Programacion\Documents\MongoDB\equipos.json" --jsonArray

mongoimport --host rs0/localhost:27017 --db torneo_copa_america --collection entrenadores --file "C:\Users\Programacion\Documents\MongoDB\entrenadores.json" --jsonArray

mongoimport --host rs0/localhost:27017 --db torneo_copa_america --collection arbitros --file "C:\Users\Programacion\Documents\MongoDB\arbitros.json" --jsonArray

mongoimport --host rs0/localhost:27017 --db torneo_copa_america --collection encuentros_deportivos --file "C:\Users\Programacion\Documents\MongoDB\encuentros_deportivos.json" --jsonArray

mongoimport --host rs0/localhost:27017 --db torneo_copa_america --collection resultados --file "C:\Users\Programacion\Documents\MongoDB\resultados.json" --jsonArray

mongoimport --host rs0/localhost:27017 --db torneo_copa_america --collection tabla_de_posiciones --file "C:\Users\Programacion\Documents\MongoDB\tabla_de_posiciones.json" –jsonArray
```

Verificación de la Replicación

1. **Insertar Datos en el Nodo Primario**

db.deportistas.insert({ "_id": "deportista006", "nombre": "Juan Pérez", "edad": 25, "nacionalidad": "Argentina", "posicion": "Defensa", "estadisticas": { "goles": 0, "asistencias": 1 } })



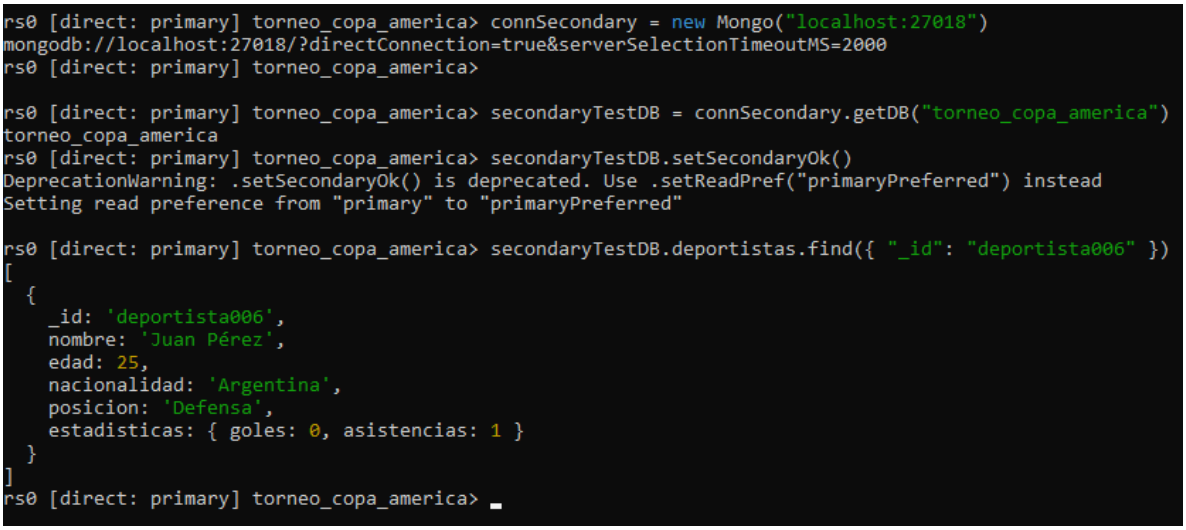
2. **Verificar la Replicación en un Nodo Secundario** Nos conectaremos al nodo en local host 27018 para verificar que los datos insertados en el nodo primario se han replicado correctamente:

connSecondary = new Mongo("localhost:27018")

secondaryTestDB = connSecondary.getDB("torneo_copa_america")

secondaryTestDB.setSecondaryOk()

secondaryTestDB.deportistas.find({ "_id": "deportista006" })



Casos de Prueba

Prueba de Disponibilidad y Redundancia

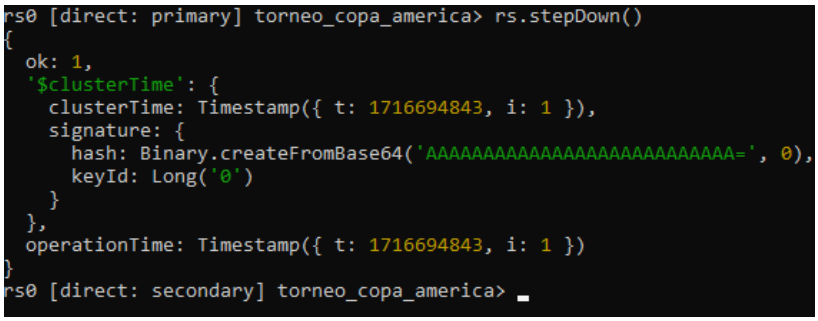
1. **Prueba de Replicación Inicial**
    - Insertar datos en el nodo primario.
    - Verificar que los datos se replican en los nodos secundarios.

2. **Prueba de Tolerancia a Fallos**
    - Apagar el nodo primario.
    - Verificar que uno de los nodos secundarios se promueve a primario.
    - Insertar nuevos datos en el nuevo nodo primario.
    - Verificar que los datos se replican en los nodos restantes.
3. **Prueba de Escalabilidad**
    - Agregar un nuevo nodo al Replica Set.
    - Verificar que el nuevo nodo recibe todos los datos replicados y se mantiene sincronizado con el nodo primario.

Ejecución de los Casos de Prueba

1. **Apagar el Nodo Primario**

rs.stepDown()



2. **Verificar el Nuevo Nodo Primario**

Aqui podemos evidenciar que el nodo en local host 27018 se ha convertido en el nodo primario.

rs.status()

```
rs0 [direct: secondary] torneo_copa_america> rs.status()
{
  set: 'rs0',
  date: ISODate('2024-05-26T03:41:20.746Z'),
  myState: 2,
  term: Long('4'),
  syncSourceHost: 'localhost:27019',
  syncSourceId: 2,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1716694873, i: 1 }), t: Long('4') },
    lastCommittedWallTime: ISODate('2024-05-26T03:41:13.515Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1716694873, i: 1 }), t: Long('4') },
    appliedOpTime: { ts: Timestamp({ t: 1716694873, i: 1 }), t: Long('4') },
    durableOpTime: { ts: Timestamp({ t: 1716694873, i: 1 }), t: Long('4') },
    lastAppliedWallTime: ISODate('2024-05-26T03:41:13.515Z'),
    lastDurableWallTime: ISODate('2024-05-26T03:41:13.515Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1716694853, i: 2 }),
  electionParticipantMetrics: {
    votedForCandidate: true,
    electionTerm: Long('4'),
    lastVoteDate: ISODate('2024-05-26T03:40:53.367Z'),
    electionCandidateMemberId: 1,
    voteReason: '',
    lastAppliedOpTimeAtElection: { ts: Timestamp({ t: 1716694843, i: 1 }), t: Long('3') },
    maxAppliedOpTimeInSet: { ts: Timestamp({ t: 1716694843, i: 1 }), t: Long('3') },
    priorityAtElection: 1,
    newTermStartDate: ISODate('2024-05-26T03:40:53.381Z'),
    newTermAppliedDate: ISODate('2024-05-26T03:40:53.886Z')
  },
```

```
  members: [
    {
      _id: 0,
      name: '127.0.0.1:27017',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 2987,
      optime: { ts: Timestamp({ t: 1716694873, i: 1 }), t: Long('4') },
      optimeDate: ISODate('2024-05-26T03:41:13.000Z'),
      lastAppliedWallTime: ISODate('2024-05-26T03:41:13.515Z'),
      lastDurableWallTime: ISODate('2024-05-26T03:41:13.515Z'),
      syncSourceHost: 'localhost:27019',
      syncSourceId: 2,
      infoMessage: '',
      configVersion: 81643,
      configTerm: -1,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: 'localhost:27018',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 873,
      optime: { ts: Timestamp({ t: 1716694873, i: 1 }), t: Long('4') },
      optimeDurable: { ts: Timestamp({ t: 1716694873, i: 1 }), t: Long('4') },
      optimeDate: ISODate('2024-05-26T03:41:13.000Z'),
      optimeDurableDate: ISODate('2024-05-26T03:41:13.000Z'),
      lastAppliedWallTime: ISODate('2024-05-26T03:41:13.515Z'),
      lastDurableWallTime: ISODate('2024-05-26T03:41:13.515Z'),
      lastHeartbeat: ISODate('2024-05-26T03:41:20.134Z'),
      lastHeartbeatRecv: ISODate('2024-05-26T03:41:19.528Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1716694853, i: 1 }),
      electionDate: ISODate('2024-05-26T03:40:53.000Z'),
      configVersion: 81643,
      configTerm: -1
    },
```

```
    {
      _id: 2,
      name: 'localhost:27019',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 824,
      optime: { ts: Timestamp({ t: 1716694873, i: 1 }), t: Long('4') },
      optimeDurable: { ts: Timestamp({ t: 1716694873, i: 1 }), t: Long('4') },
      optimeDate: ISODate('2024-05-26T03:41:13.000Z'),
      optimeDurableDate: ISODate('2024-05-26T03:41:13.000Z'),
      lastAppliedWallTime: ISODate('2024-05-26T03:41:13.515Z'),
      lastDurableWallTime: ISODate('2024-05-26T03:41:13.515Z'),
      lastHeartbeat: ISODate('2024-05-26T03:41:20.558Z'),
      lastHeartbeatRecv: ISODate('2024-05-26T03:41:19.597Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'localhost:27018',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 81643,
      configTerm: -1
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1716694873, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1716694873, i: 1 })
}
```

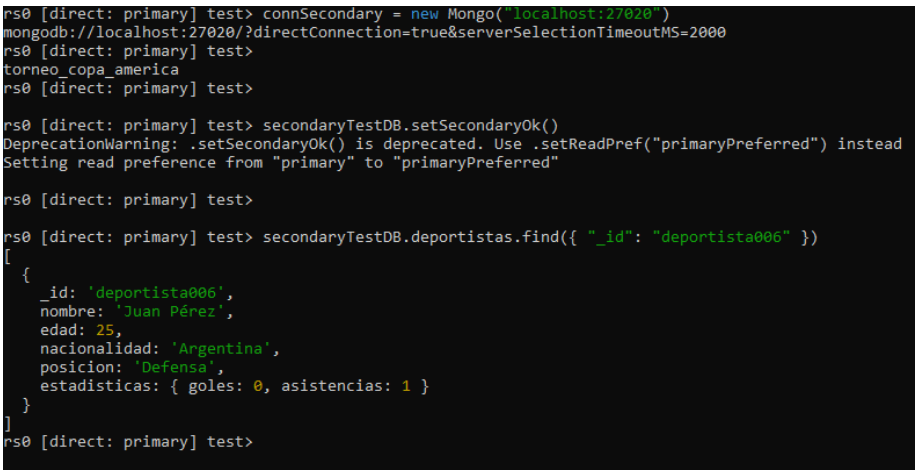

3. **Agregar un Nuevo Nodo**

rs.add("localhost:27020")


```
    {
      _id: 3,
      name: 'localhost:27020',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 18,
      optime: { ts: Timestamp({ t: 1716695400, i: 1 }), t: Long('4') },
      optimeDurable: { ts: Timestamp({ t: 1716695400, i: 1 }), t: Long('4') },
      optimeDate: ISODate('2024-05-26T03:50:00.000Z'),
      optimeDurableDate: ISODate('2024-05-26T03:50:00.000Z'),
      lastAppliedWallTime: ISODate('2024-05-26T03:50:00.158Z'),
      lastDurableWallTime: ISODate('2024-05-26T03:50:00.158Z'),
      lastHeartbeat: ISODate('2024-05-26T03:50:12.794Z'),
      lastHeartbeatRecv: ISODate('2024-05-26T03:50:13.273Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'localhost:27018',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 81645,
      configTerm: 4
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1716695400, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1716695400, i: 1 })
}
```

4. **Comprobar que los datos del nodo primario ya estén en el nuevo nodo creado**

```
connSecondary = new Mongo("localhost:27020")
secondaryTestDB = connSecondary.getDB("torneo_copa_america")
secondaryTestDB.setSecondaryOk()
secondaryTestDB.deportistas.find({ "_id": "deportista006" })
```



Reporte de Resultados y Análisis

1. **Resultados de la Replicación Inicial**
    - Los datos se replicaron correctamente en los nodos secundarios.
2. **Resultados de la Prueba de Tolerancia a Fallos**
    - Un nodo secundario se promovió a primario correctamente.
    - Los datos insertados en el nuevo nodo primario se replicaron en los nodos restantes.
3. **Resultados de la Prueba de Escalabilidad**
    - El nuevo nodo se agregó y se sincronizó correctamente con los datos existentes.