Troubleshooting y Operación de MongoDB

{ paradigma

Índice

1. Configuración de MongoDB
2. Replica Set Internals
3. Operación de MongoDB
4. Troubleshooting

Autor Ernesto Valero Marcelo
MongoDB Certified DBA
evalero@paradigmadigital.com

1 Configuración de MongoDB

Índice

1. Componentes de paquete de MongoDB
2. Arquitecturas de despliegue
3. Configuración en StandAlone
4. Configuración en Replica Set
5. Configuración en Sharding

El paquete de MongoDB contiene los siguientes binarios. Los binarios para arrancar el **servidor de BBDD y acceso a la shell** de MongoDB son:

- **mongod**: Es el servicio principal de MongoDB. Maneja los accesos a los datos, las peticiones de datos y ejecuta tareas de mantenimiento en background. Su fichero de configuración es **mongod.conf**.
- **mongo:** Es la shell interactiva de MongoDB. Aporta un entorno funcional completo para ser usado con la BBDD.
- **mongos**: Es un servicio propio del modo de despliegue Shard. Su función es la de enrutar las peticiones de la capa de aplicación y determinar la ubicación de los datos en los diferentes shards del despliegue.

Herramientas para **backup y restore** de datos :

- **mongodump**: Es una utilidad para crear un export binario del contenido una base de datos. Podemos considerar MongoDB como una herramienta más para realizar copias de seguridad. Podremos usar esta herramienta contra “mongod” o “mongos” , teniendo en cuenta que “mongod” podrá estar arrancado o parado indistintamente.
- **mongorestore**: En conjunción con mongodump, mongorestore se utiliza para restaurar los respaldos realizados con mongodump.

Herramientas para **exportación e importación** de datos :

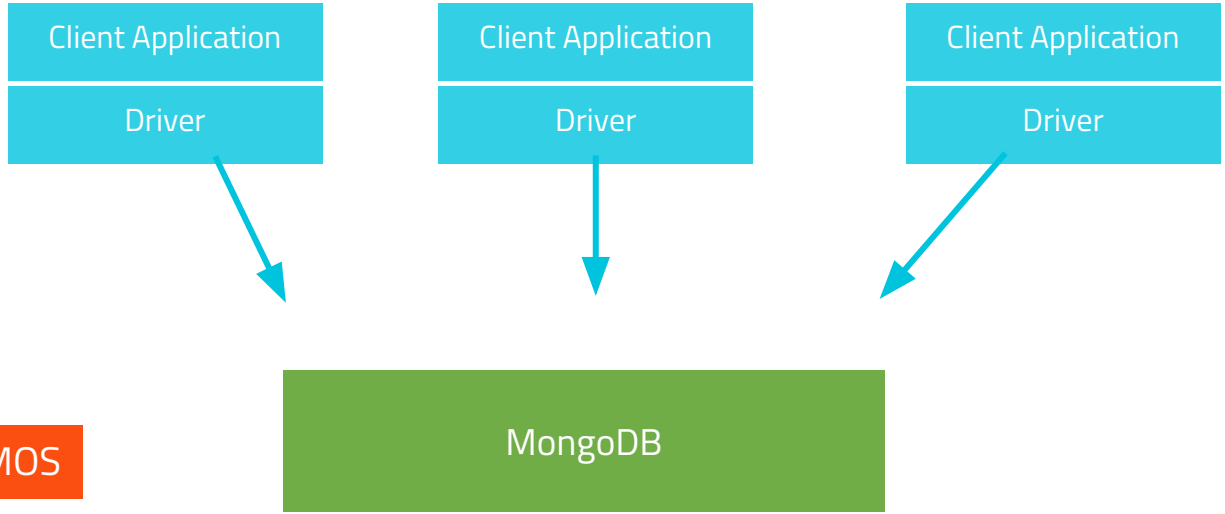
- **bsondump**: Convierte ficheros BSON a algún formato legible por humanos, incluido a JSON. Se trata de una herramienta de análisis, en ningún caso debe ser utilizada para otro tipo de actividades.
- **mongoexport**: Utilidad que permite exportar los datos de una instancia de MongoDB en formato JSON o CSV. En conjunción con mongoimport son útiles para hacer backup de una parte bien definida de los datos de la BBDD MongoDB o para casos concretos de inserción de datos.
- **mongoimport**: Utilidad que permite importar los datos de una instancia de MongoDB desde ficheros con formato JSON o CSV.
- **mongofiles**: Utilidad que permite manejar ficheros en una instancia de MongoDB con objetos GridFS desde la línea de comandos. En Replica Set sólo podrá leer desde el primario.

Herramientas para **análisis de performance y actividad**:

- **mongoperf:** Utilidad para comprobar el performance I/O de forma independiente a MongoDB.
- **mongostat:** Utilidad que proporciona una rápida visión del estado actual de los servicios mongod y mongos. Es similar a la utilidad vmstat.
- **mongotop:** Proporciona un método para trazar el tiempo que una instancia de MongoDB gasta en las operaciones de Lectura/Escritura de datos. Proporciona estadísticas a nivel de colección, por defecto cada segundo.
- **mongosniff:** Es una utilidad que proporciona una visión a bajo nivel de la actividad de la BBDD a través de la actividad de red. Es equivalente a un analizador de tráfico de red TCP/IP

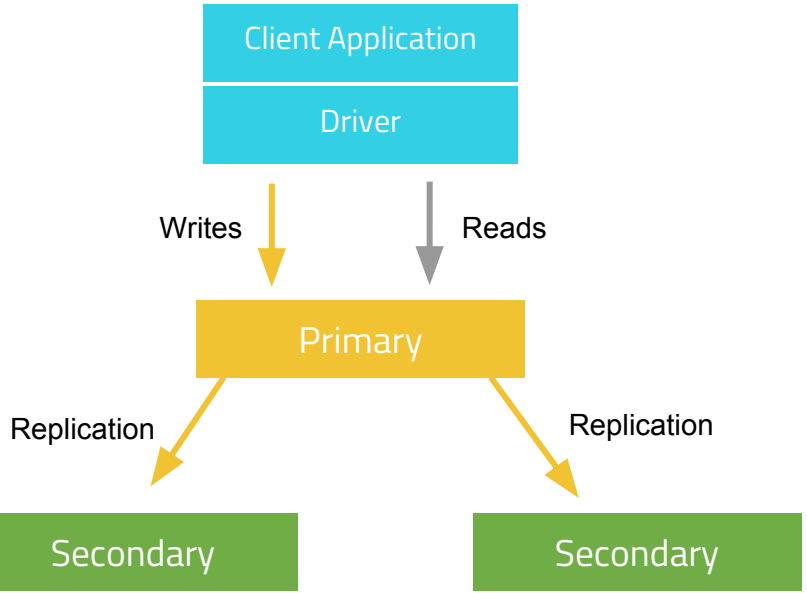
7 Arquitecturas de despliegue

- Standalone



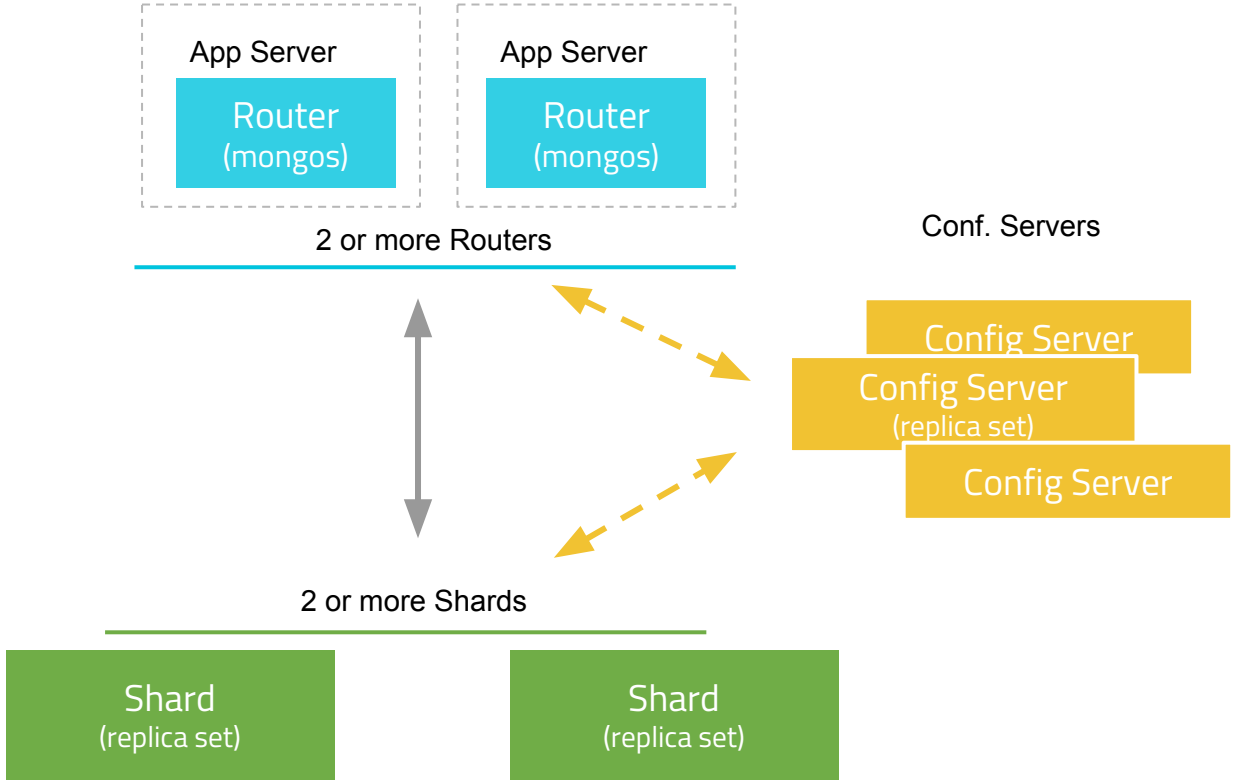
PRACTIQUEMOS

- Replica set



PRACTIQUEMOS

- Sharding


App Server
Router
(mongos)
App Server
Router
(mongos)
2 or more Routers
Conf. Servers
Config Server
Config Server
(replica set)
Config Server
2 or more Shards
Shard
(replica set)
Shard
(replica set)

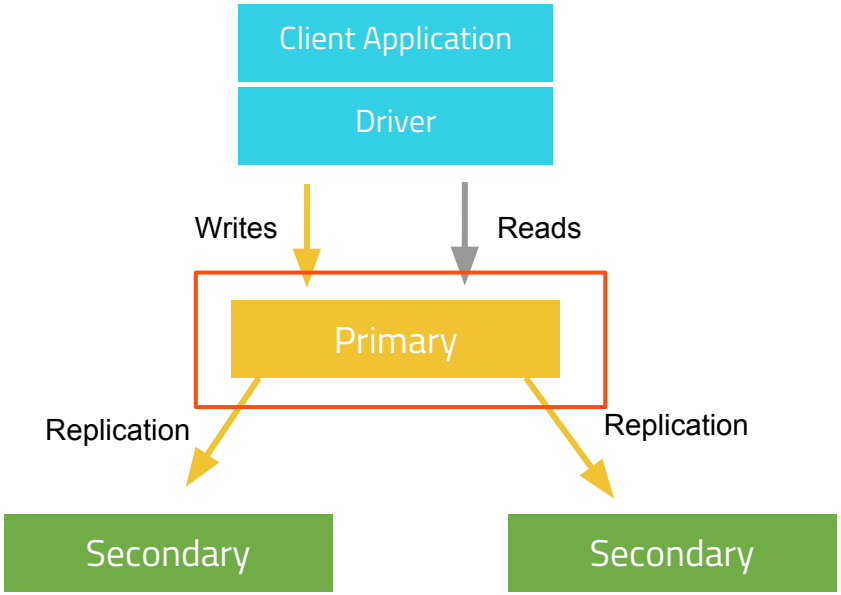
2 Replica Set Internals

Índice

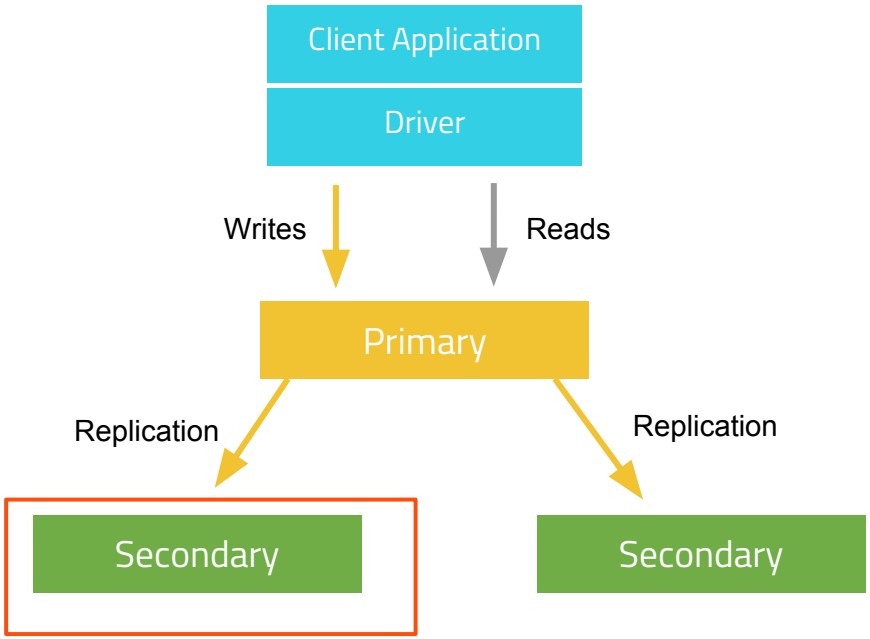
1. Componentes
2. Planificar arquitectura de replica set (# de nodos, replicación, hidden nodes ...)
3. Diagrama de comunicación, replicación
4. Funcionamiento de replicación y elección de nuevo primario
5. Diagrama de comunicación de lecturas y escrituras cliente
6. Operaciones sobre Replica sets (Añadir / quitar nodos, cambiar prioridades, cambiar tamaño de Oplog, compactación de tamaño en un replica set, aplicar índices no bloqueante)

Un replica set está compuesto por al menos:
Primario : Nodo que recibe todas las peticiones de escritura
Secundario: Nodo de réplica de datos

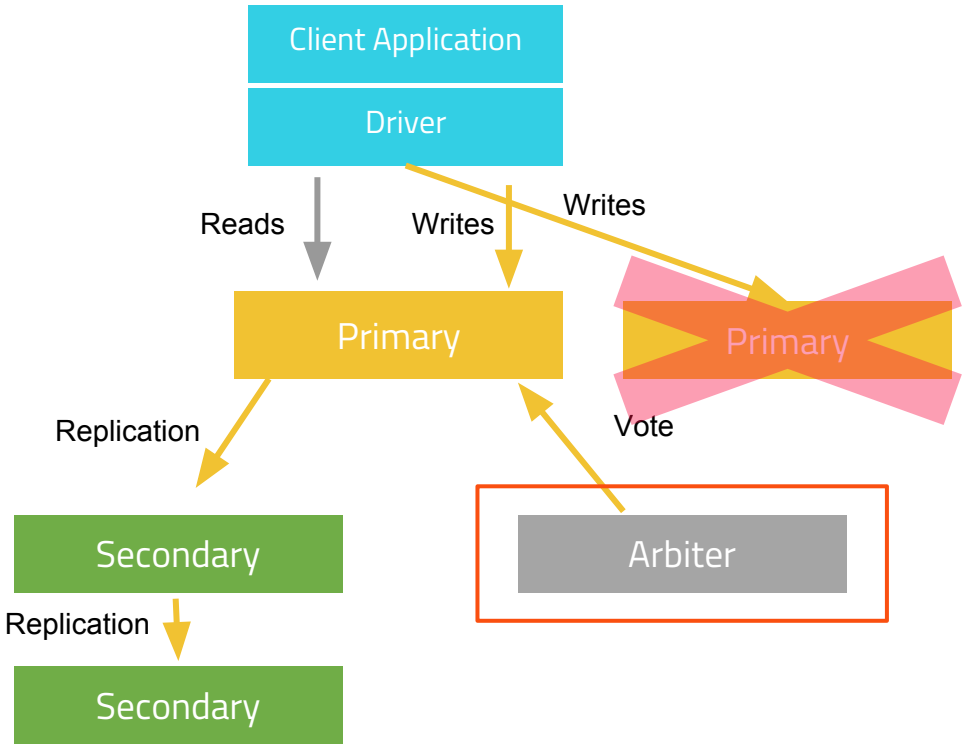
Y opcionalmente
Árbitro: sirve para el proceso de votación que promociona a primario alguno de los nodos secundarios en caso de empate. **Un nodo árbitro deberá ser utilizado siempre que tengamos un número par de nodos que almacenan datos**



- Es el único nodo que es capaz de recibir las escrituras
- Por defecto, también puede servir las lecturas
- Almacena la información a replicar entre el resto de nodos en la colección oplog



- Los nodos secundarios reciben la replicación del oplog de manera asíncrona
- Normalmente se replica del nodo primario.
- Por defecto, cualquier nodo con rol secundario podría promocionar a nodo primario en caso de ser necesario



- Los nodos árbitros únicamente se usan para desempatar en un proceso de elección de un nuevo primario
- Se añaden cuando el número de nodos del replica set es par y no se desea añadir más nodos de replicación

Factores para arquitectura de Replica Set

- Redundancia de datos necesaria
- Capacidad de failover
- Disponibilidad geográfica
- Operaciones de Business Intelligence
- Backup
- Costes

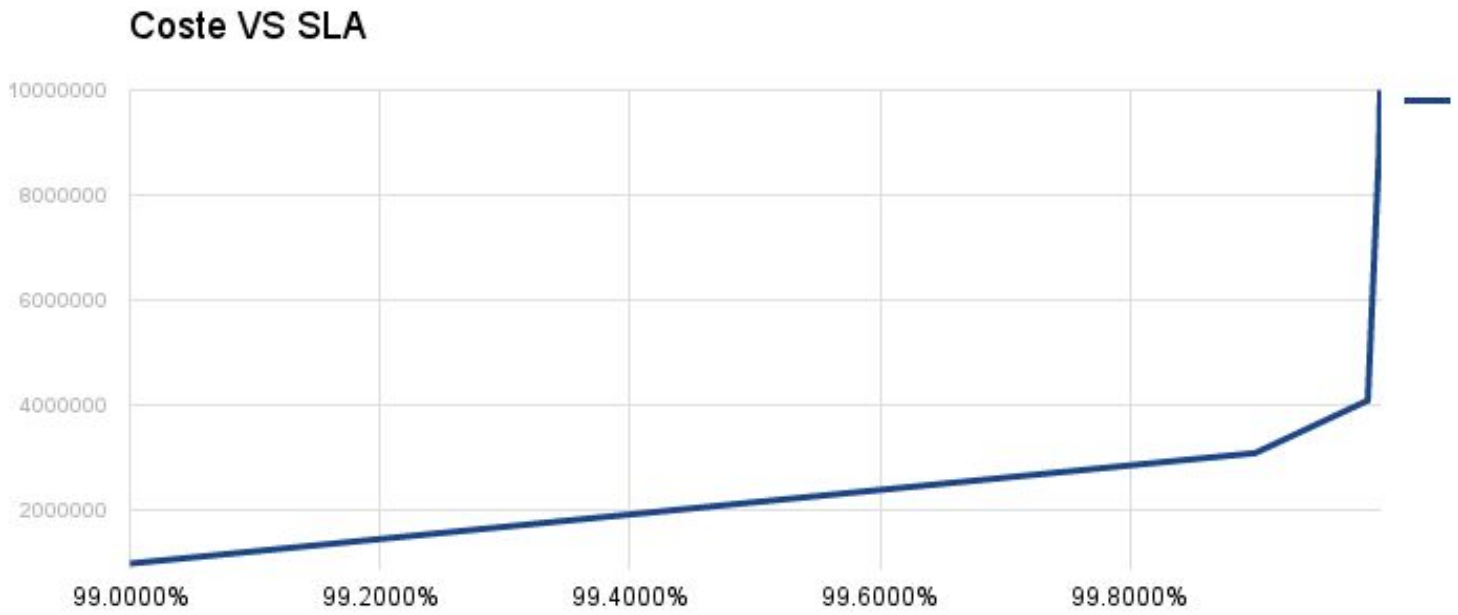
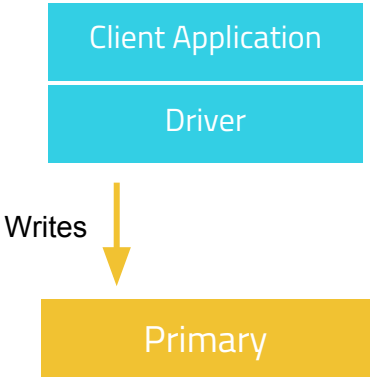
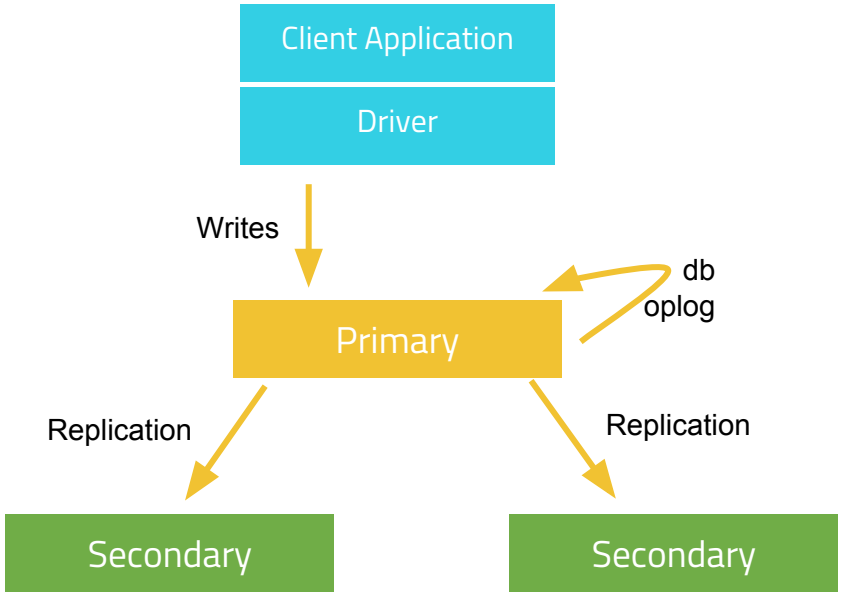

Coste VS SLA
10000000
8000000
6000000
4000000
2000000
99.0000%
99.2000%
99.4000%
99.6000%
99.8000%

Diagrama de replicación de escrituras



El nodo primario recibe la(s) escritura(s)

Diagrama de replicación de escrituras



- El Primario escribe la operación en memoria en la bbdd afectada por la escritura
- Registra en la colección del oplog la operación ejecutada
- Los Secundarios obtienen la operación del oplog del primario y lo escriben en sus respectivas copias

Diagrama de comunicación (heartbeat)

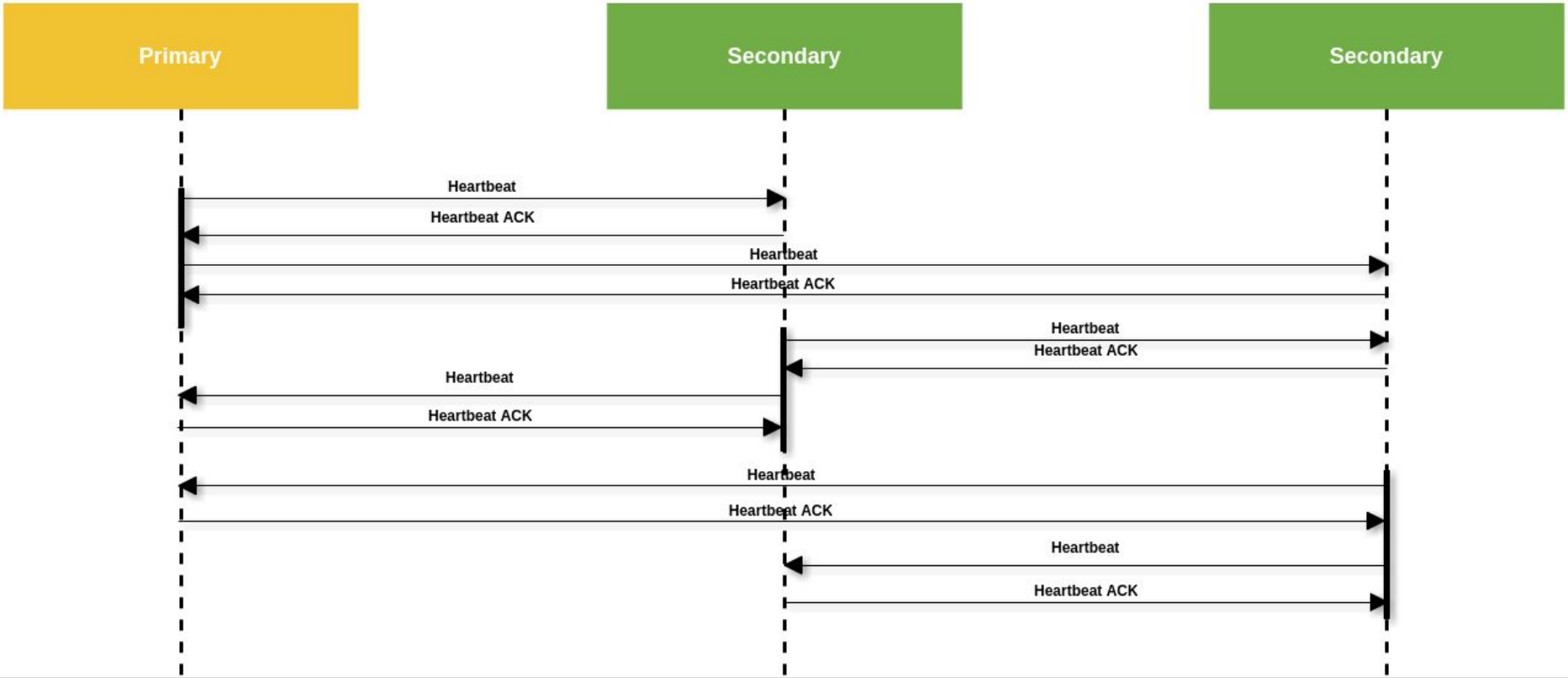


Diagrama de elección de nuevo primario

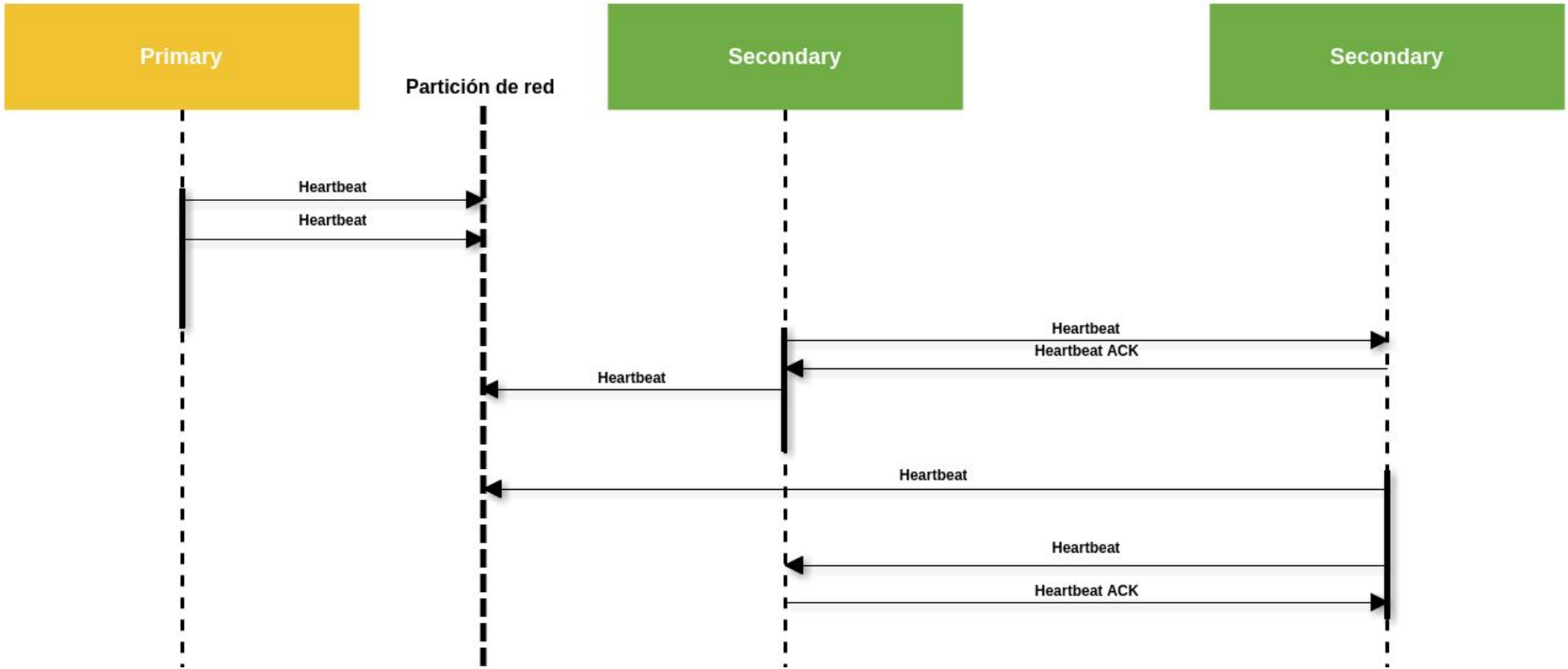


Diagrama de elección de nuevo primario

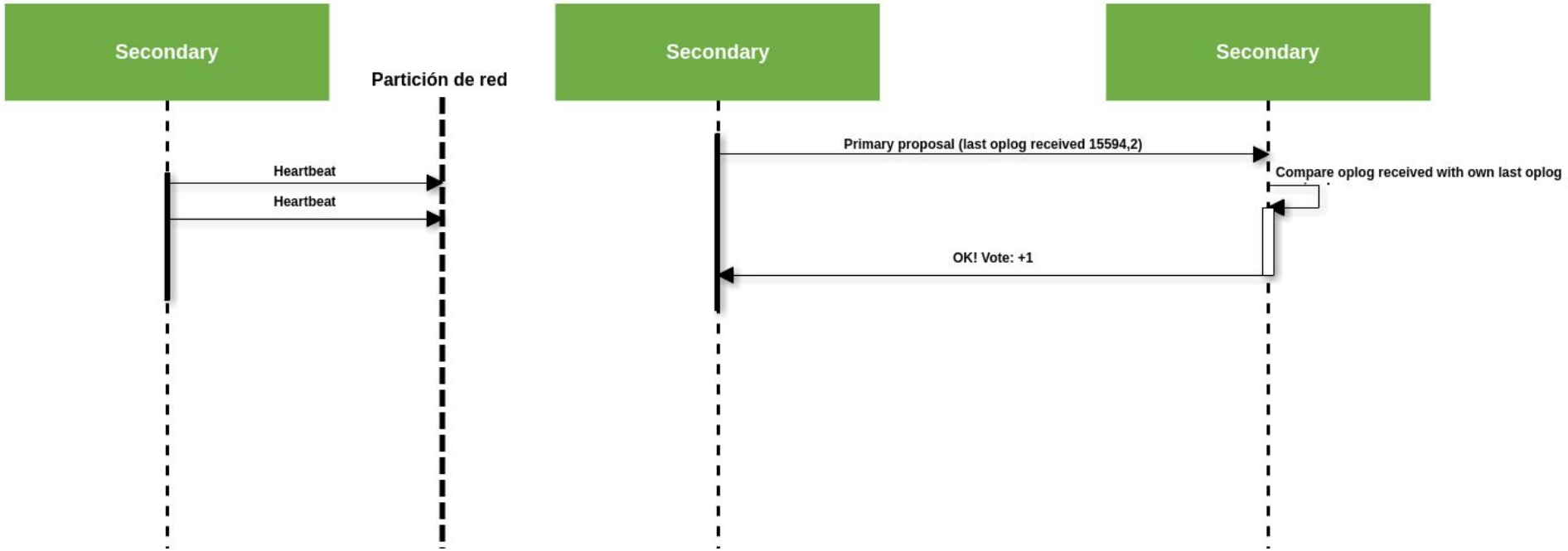
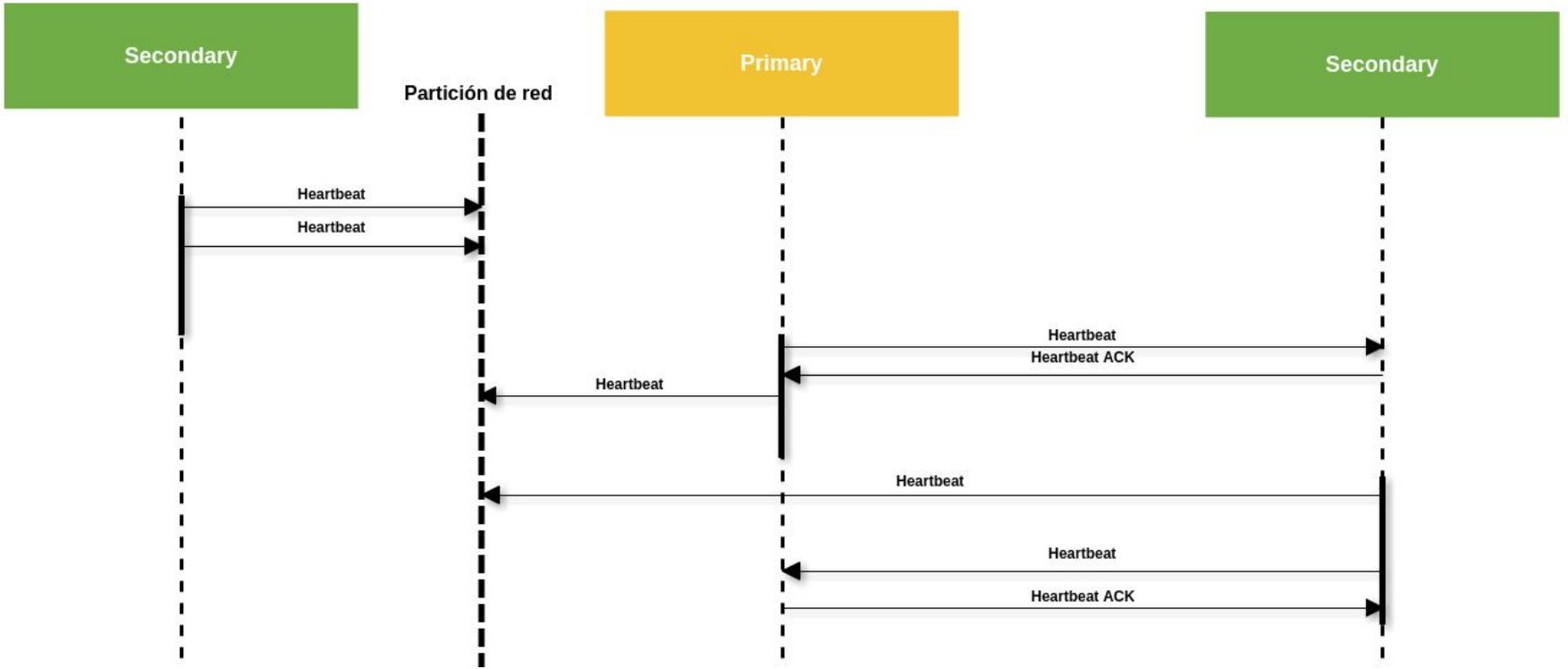
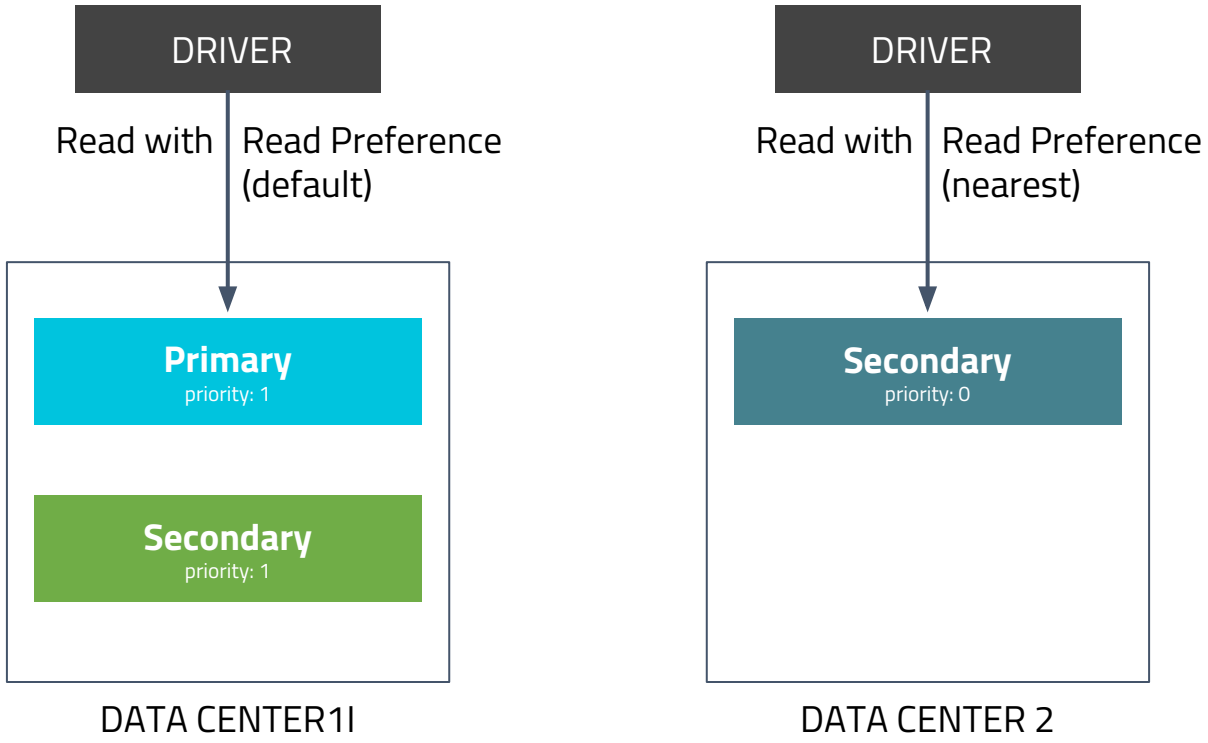


Diagrama de elección de nuevo primario



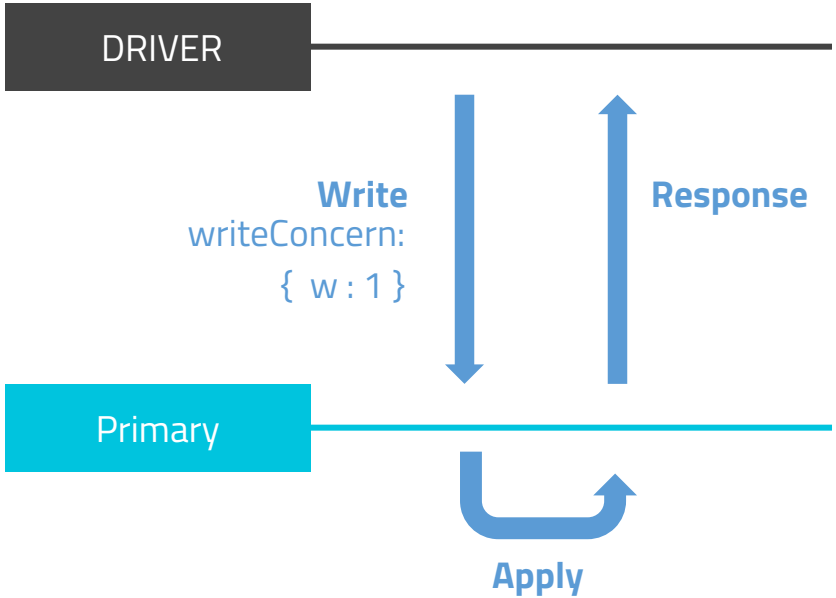
Los datos en un replica set son eventualmente consistentes y nuestra aplicación puede ser más o menos permisiva con la lectura de datos no frescos.
Para ello podemos configurar nuestra aplicación para indicarle sobre que nodos del replica set leer datos (read preference):

- **Primary :** (por defecto) Los datos se leen siempre de un nodo primario
- **Primary preferred :** Los datos preferentemente serán leídos de un primario, y en caso de no ser alcanzable o disponible, se leerá de un nodo secundario.
- **Secondary :** Los datos se leen siempre de un nodo secundario
- **Secondary preferred** : Los datos preferentemente serán leídos de un secundario, y en caso de no ser alcanzable o disponible, se leerá de un nodo primario.
- **Nearest :** Los datos se leen del nodo del réplica set con menor latencia sin importar el rol dentro del replica set

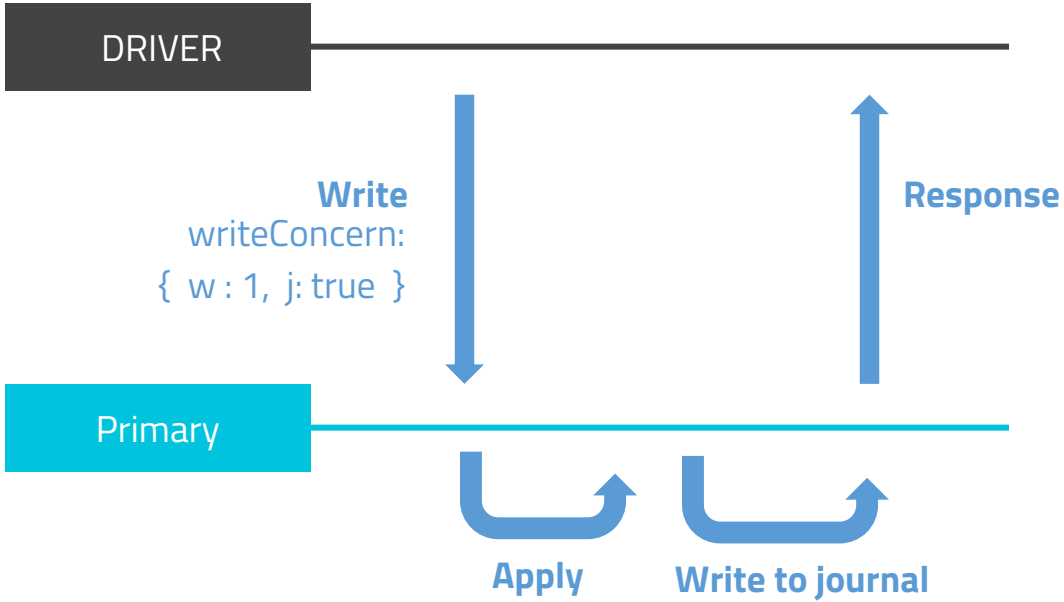

DRIVER
Read with
Read Preference
(default)
Primary
priority: 1
Secondary
priority: 1
DATA CENTER1I
DRIVER
Read with
Read Preference
(nearest)
Secondary
priority: 0
DATA CENTER 2

Al igual que con la lectura, dependiendo de la seguridad que queremos dar de que el dato ha sido escrito, podemos configurar diferentes niveles de acknowledge de escritura del dato.

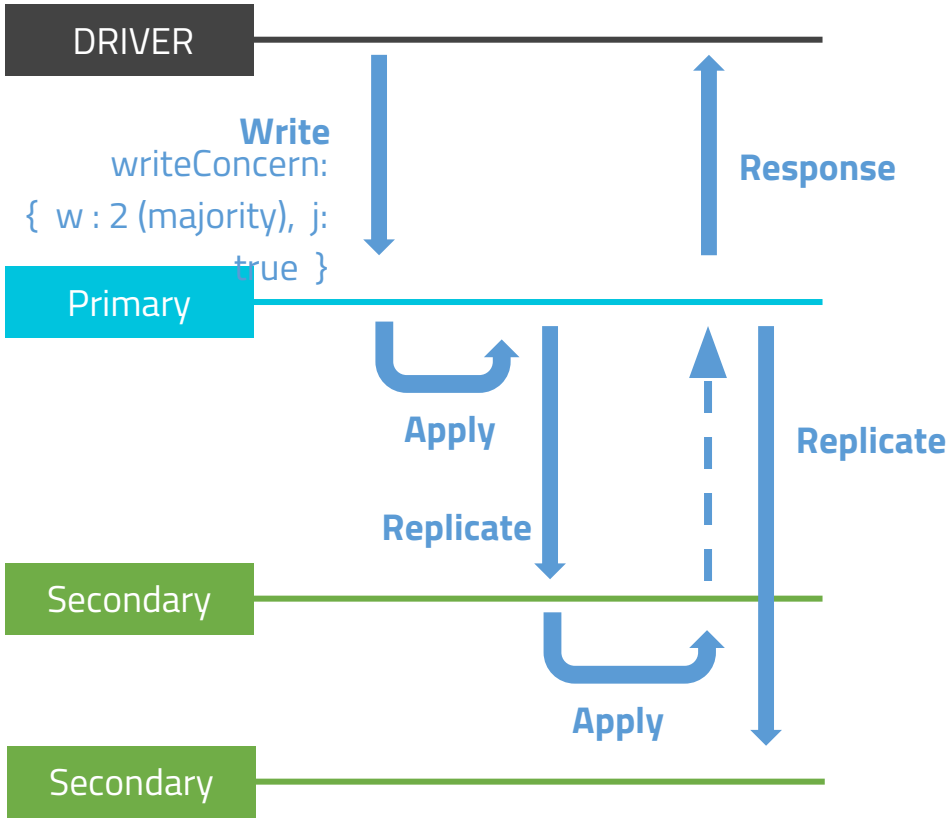
Por defecto, el dato se da por escrito cuando se encuentra aplicado sobre el nodo primario



El modo journaled, el dato se da por escrito cuando se encuentra escrito en el journal del nodo primario



Por último, podemos especificar el número de nodos en los que debe estar replicado el dato para considerarlo como escrito en el replica set



En algunas ocasiones, se requiere hacer algunas operativas sobre un replica set. Entre las más típicas

- **Añadir / quitar nodos a un replica set existente**
- **Cambio de configuración de replica set prioridad de nodos**
- **Convertir un nodo del replica set a hidden**
- **Convertir un nodo del replica set a delayed**
- **Aplicación de índices no bloqueantes**
- **Cambio de tamaño de oplog**
- **Compactación de datafiles**
- **Reconfiguración de la cadena de replicación**
- **Reconfiguración del cluster sin mayoría**

Modificación de miembros de un replica set

Para modificar el número de miembros de un replica set debemos tener en cuenta:

- Que exista mayoría en el replica set antes de modificar la configuración (salvo casos especiales)

Para añadir nuevos nodos (hasta 50 en total) se ejecuta desde el primario el comando:

```
rs.add("hostname:puerto")
```

Para eliminar nodos del replica set se ejecuta desde el nodo primario el comando:

```
rs.remove("hostname:puerto")
```

PRACTIQUEMOS

Cambio de configuración de prioridad de nodos

El fichero de configuración nos permite indicar afinidad contra un nodo para que se convierta en primario. Para ello en el json de la configuración del replica set debemos añadir el parámetro dentro `members[n].priority`

```
"members" : [{      "_id" : 1,
                    "host" : "mongodb:27018",
                    "arbiterOnly" : false,
                    "buildIndexes" : true,
                    "hidden" : false,
                    "priority" : 2,
                    "tags" : {
                    },
                    "slaveDelay" : NumberLong(0),
                    "votes" : 1
            },
                   ...]
```

PRACTIQUEMOS

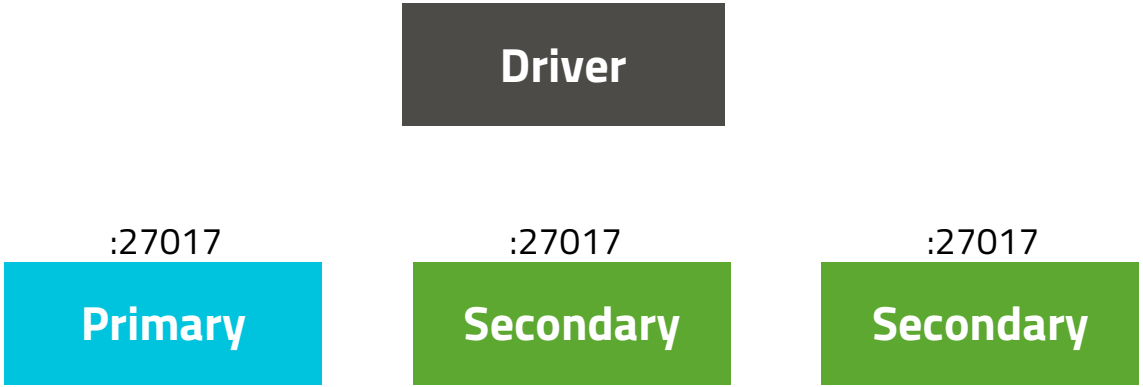
Cambio de configuración de nodo hidden

Los nodos hidden nunca pueden ser primarios y de cara al cliente “no existen” en el replica set.
Por ello este nodo **nunca deberían poder promocionar a primario.**
Para configurarlo debemos cambiar las siguientes propiedades:
members[n].hidden=true
members[n].priority=0

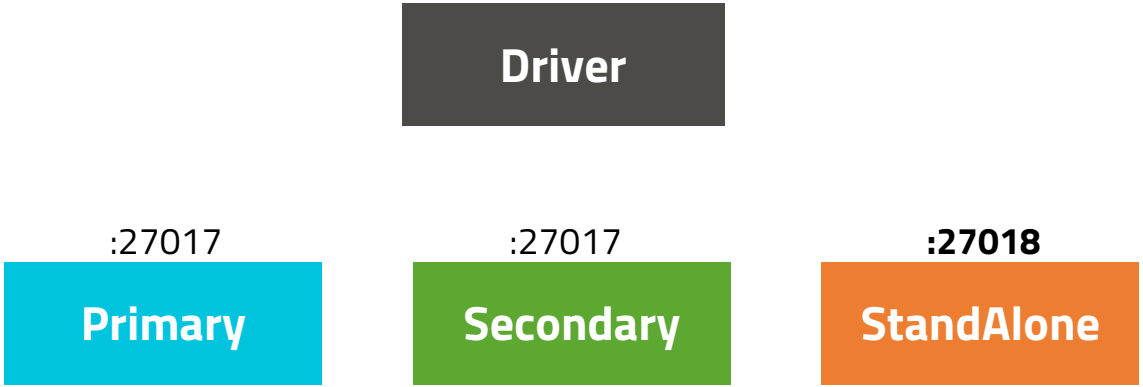
```
"members" : [{      "_id" : 1,
                    "host" : "mongodb:27018",
                    "arbiterOnly" : false,
                    "buildIndexes" : true,
                    "hidden" : true,
                    "priority" : 0,
                    "tags" : {
                    },
                    "slaveDelay" : NumberLong(0),
                    "votes" : 1
            },
```

PRACTIQUEMOS

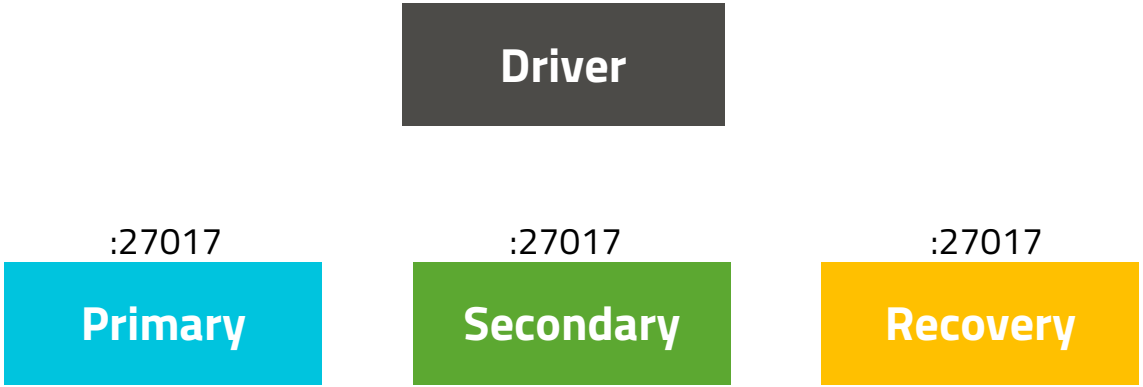
Rolling upgrade



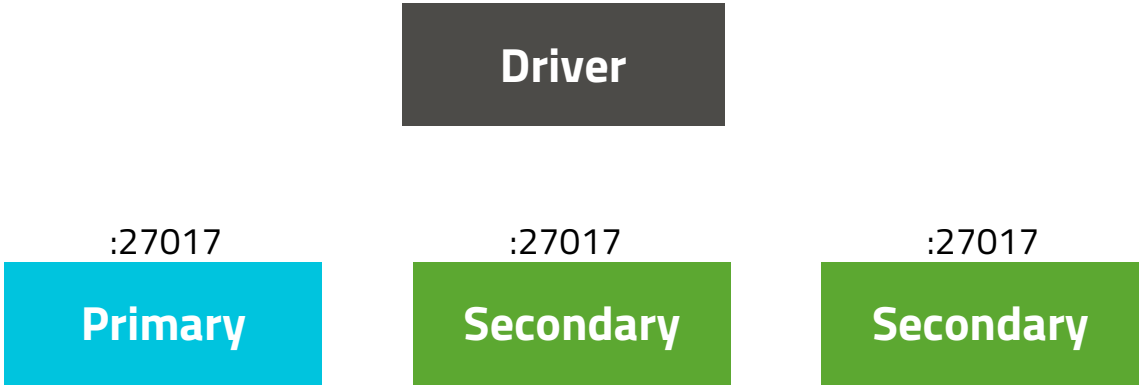
Rolling upgrade



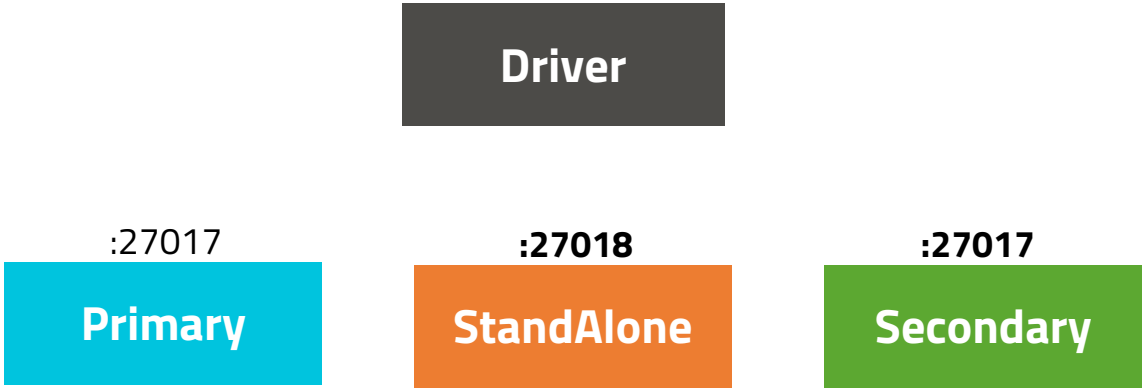
Rolling upgrade



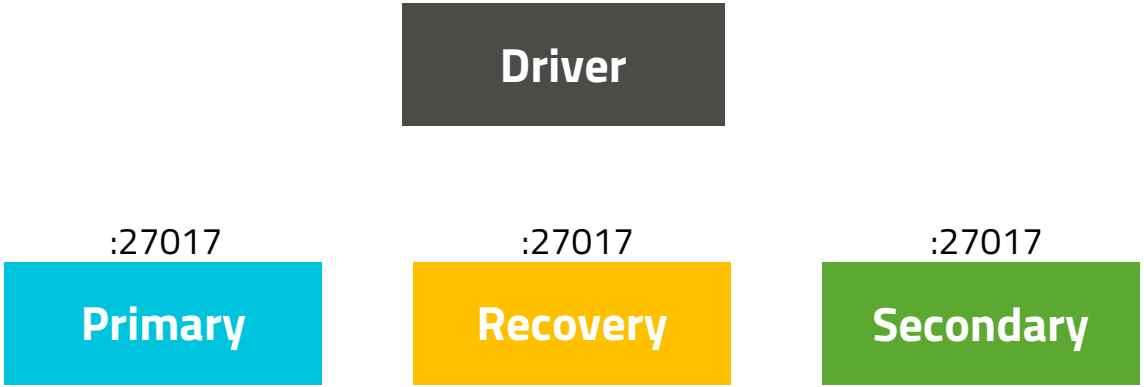
Rolling upgrade



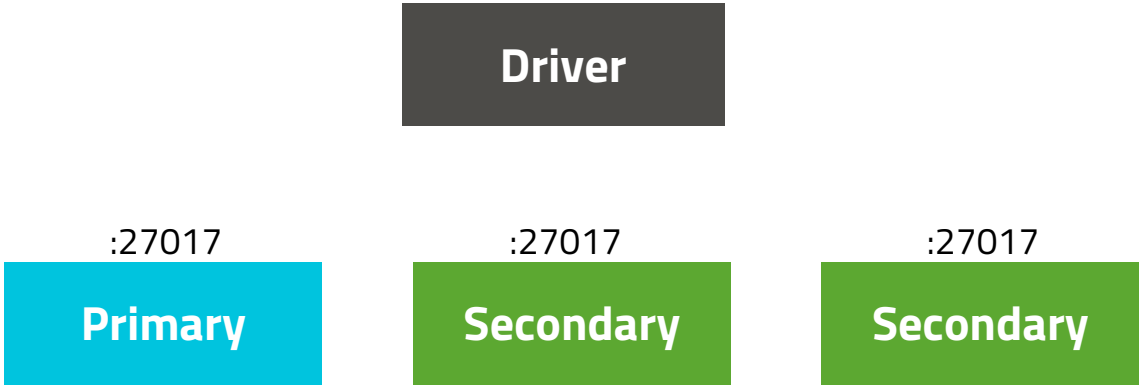
Rolling upgrade



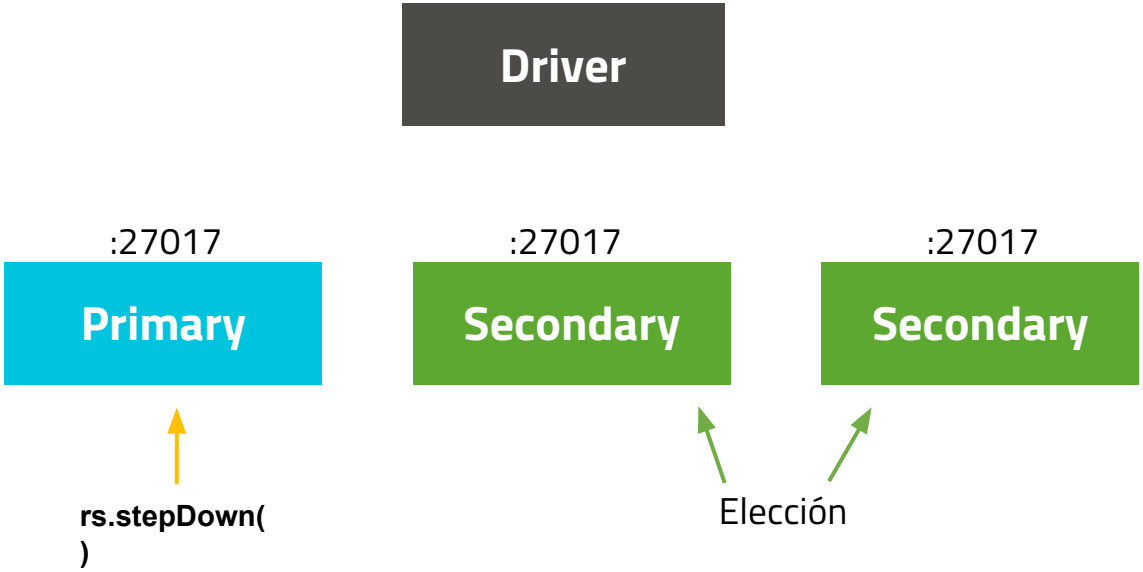
Rolling upgrade



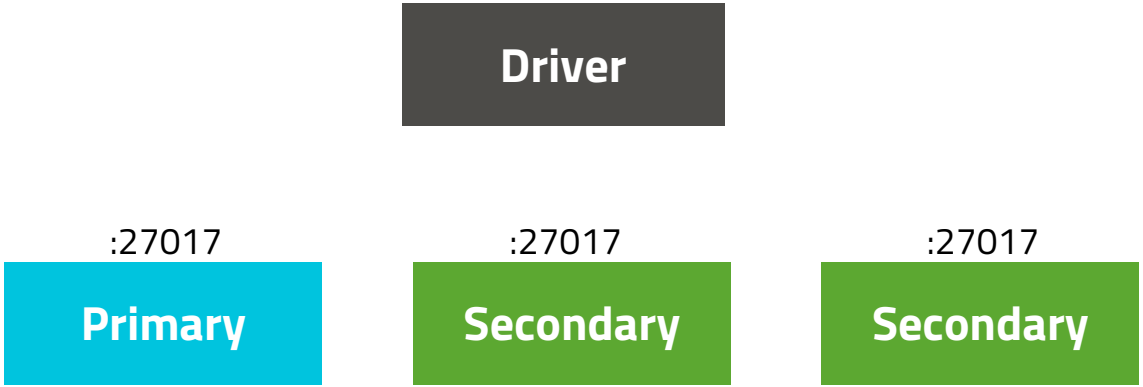
Rolling upgrade



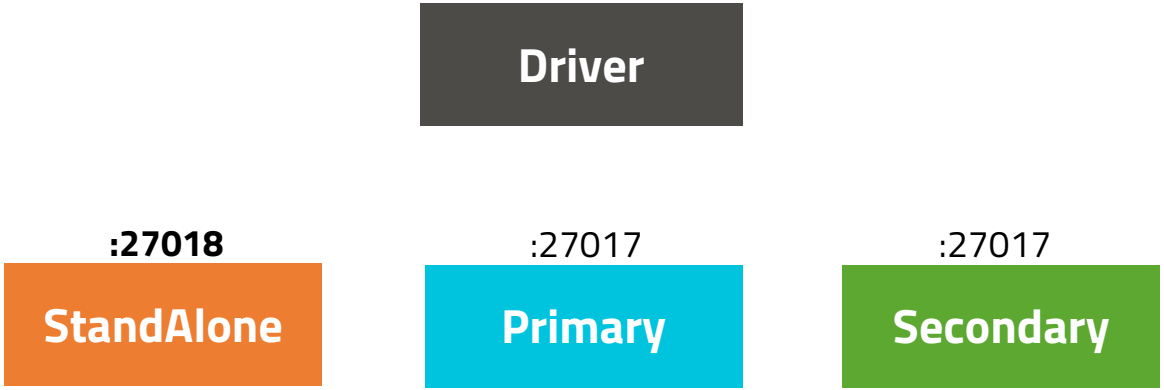
Rolling upgrade



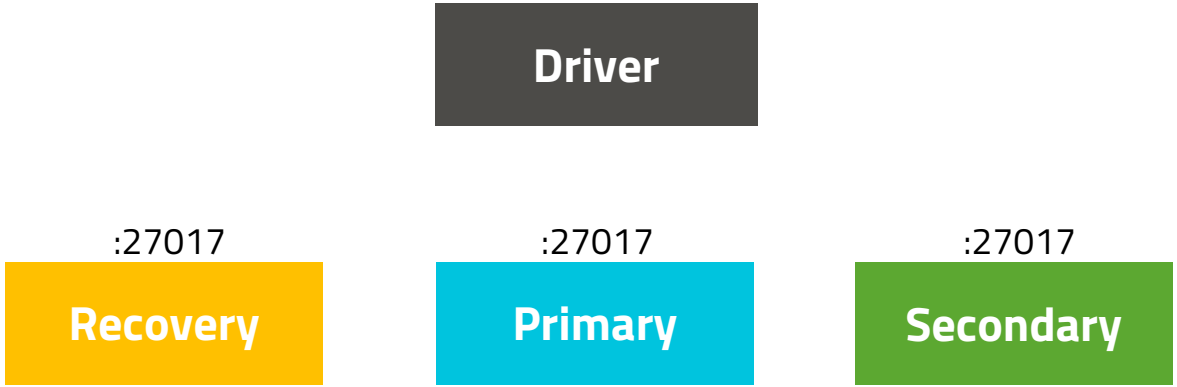
Rolling upgrade



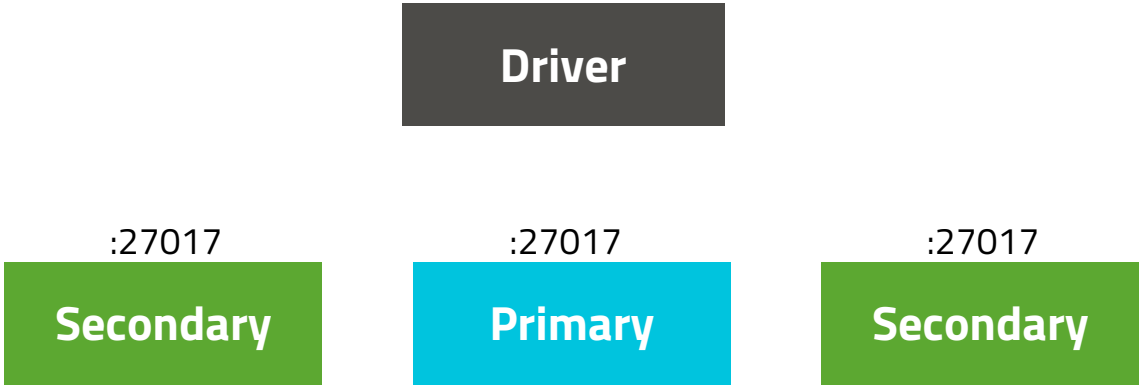
Rolling upgrade



Rolling upgrade



Rolling upgrade



Aplicación de índices en Replica Set

La aplicación de un índice es bloqueante a nivel de Base de Datos. Para aplicar un índice sobre una base de datos y no bloquear el servicio se aplica el siguiente procedimiento:

1. Paramos un nodo secundario
2. Levantamos el nodo **fuera del replica set y en un puerto distinto**
3. Aplicamos el índice
4. Volvemos a levantar el nodo dentro del **replica set**
5. Aplicamos el procedimiento al resto de nodos
6. En el nodo primario, ejecutar antes el comando rs.stepDown() antes de pararlo

PRACTIQUEMOS

Resize de Oplog

En algunos casos nos puede interesar aumentar el tamaño de Oplog para poder tener mayor ventana de replicación y capacidad de recuperación de un nodo secundario que haya quedado atrás durante la réplica.

El procedimiento resumido consta de los siguientes pasos:

1. Parar un nodo y sacarlo del replica set
2. Creamos una colección temporal en la BBDD local donde almacenamos la última entrada de la colección oplog.rs
3. Borramos la colección oplog.rs y la recreamos con el tamaño que deseemos
4. Introducimos el documento de la última entrada del oplog almacenado en la colección temporal
5. Volvemos a añadir el nodo al replica set
6. Repetimos en cada nodo

PRACTIQUEMOS

Compactación de datafiles

Cuando borramos documentos de una BBDD, el hueco que ha dejado el documento intentará ser reutilizado por MongoDB, pero podría llegar a producirse una fragmentación de los datafiles.
Para compactar y reclamar el espacio en disco debemos realizar el siguiente procedimiento:

1. Paramos un nodo secundario
2. Borramos el contenido del directorio del dbpath
3. Levantamos nuevamente el nodo para que se repliquen los datos por completo
4. Repetimos el proceso con todos los nodos del replica set (no hacer todos los nodos a la vez y esperar que un nodo haya acabado su resincronización antes de empezar uno nuevo)

PRACTIQUEMOS

Reconfiguración de la cadena de replicación

En ocasiones, como por ejemplo la sincronización inicial de un nuevo nodo de un replica set, queremos apuntar un nodo secundario a un nodo distinto del nodo primario para aliviar la carga.
Para realizarlo:

1. Nos conectamos al nodo secundario correspondiente
2. ejecutamos el comando rs.syncFrom("hostname:puerto")

Para comprobar que el cambio se ha realizado con éxito, ejecutamos el comando rs.status()

PRACTIQUEMOS

Reconfiguración de la cadena de replicación

En ocasiones, como por ejemplo la sincronización inicial de un nuevo nodo de un replica set, queremos apuntar un nodo secundario a un nodo distinto del nodo primario para aliviar la carga.
Para realizarlo:

1. Nos conectamos al nodo secundario correspondiente
2. ejecutamos el comando rs.syncFrom("hostname:puerto")

Para comprobar que el cambio se ha realizado con éxito, ejecutamos el comando rs.status()

PRACTIQUEMOS

Reconfiguración de un clúster sin mayoría

¡¡ADVERTENCIA!!

Usar con muchísima precaución

Reconfiguración de un clúster sin mayoría

Este comando debería usarse como último recurso si todos los miembros de un replica set entero no salvo uno

1. Nos conectamos al miembro superviviente y recuperamos la configuración del replica set
2. Modificamos la configuración dejando únicamente el miembro superviviente como miembro del replica set
3. Ejecutamos el comando rs.reconfig(cfg,{force:true})

PRACTIQUEMOS

3 Operación de MongoDB

Índice

1. Parada y arranque de MongoDB. Localización de fichero de log
2. Obtención de índices y evaluación de su uso en querys
3. Profiling

Para arrancar MongoDB puede realizarse mediante dos métodos:

- Arranque directamente por comando
  - **`mongod --port 27017 --dbpath /data --fork --logpath /var/log/mongod.log`**
- Arranque mediante systemd o init
  - systemctl start mongod
  - service mongod start

Para averiguar la localización del fichero de logs de MongoDB podemos:

- Consultar el fichero de configuración y buscar el parámetro **systemLog.path**
- Ejecutar el comando ps -efa | grep mongod si mongo ha sido instanciado directamente por comando
    - **`mongod --port 27017 --dbpath /data --fork --logpath /logs/mongod.log`**

Por defecto la ubicación de los logs de MongoDB se encuentra en :

`/var/log/mongod.log`

Para obtener el listado de índices creados en una colección ejecutamos el comando:

db.nombreColeccion.getIndexes()

```
[       {       "v" : 1,
                "key" : {
                        "_id" : 1},
                "name" : "_id_",
                "ns" : "test.companies"
        },
        {       "v" : 1,
                "key" : {
                        "category_code" : 1,
                        "name" : 1},
                "name" : "category_code_1_name_1",
                "ns" : "test.companies"
        },
        {       "v" : 1,
                "key" : {
                        "number_of_employees" : 1
                },
                "name" : "number_of_employees_1",
                "ns" : "test.companies"}
]
```

Si queremos evaluar si una query está usando alguno de los índices que tenemos creados ejecutamos el comando explain() al final de la query. Este comando nos muestra la salida del plan de ejecución del query planner de MongoDB para ejecutar la query

```
db.Companies.find({<query>}).explain()
```

PRACTIQUEMOS

```
[       {       "v" : 1,
                "key" : {
                        "_id" : 1},
                "name" : "_id_",
                "ns" : "test.companies"
        },
        {       "v" : 1,
                "key" : {
                        "category_code" : 1,
                        "name" : 1},
                "name" : "category_code_1_name_1",
                "ns" : "test.companies"
        },
        {       "v" : 1,
                "key" : {
                        "number_of_employees" : 1
                },
                "name" : "number_of_employees_1",
                "ns" : "test.companies"}
]
```

```
db.companies.find({name:"Skype"})

db.companies.find({category_code:"news"}

db.companies.find({name:"Skype"},{_id:0,category
_code:1,name:1})

db.companies.find({category_code:"news"},{catego
ry_code:1,name:1})

db.companies.find({category_code:"news"},{_id:0,
category_code:1,name:1})

db.companies.find({category_code:"news"},{_id:0,
category_code:1,name:1}).sort({name:-1,category_
code:-1})
```

https://docs.mongodb.com/manual/reference/explain-results/#queryplanner

En ocasiones hay que evaluar el rendimiento de las querys que se están ejecutando en una instancia. Para ello podemos habilitar el profiling en una base de datos y observar aquellas querys que están tardando más de un threshold específico o para todas las querys

```
db.setProfilingLevel(0) → Deshabilitado
db.setProfilingLevel(1,<threshold_ms>) → Habilitado para un thershold
db.setProfilingLevel(2) → Habilitado para todas las querys
```

En la base de datos se crea una colección capped llamada system.profile donde se almacenan las querys

PRACTIQUEMOS

4 Troubleshooting

Índice

1. Códigos de error en logs
2. Troubleshooting de instancias que no arrancan
3. Troubleshooting en Replica Set

Exit code	Significado
2	Opciones especificadas en configuración son incompatibles entre sí.
4	Si se levanta una instancia con una versión inferior de MongoDB que los datafiles han sido escritos, podría saltar este error
14	Error irrecuperable o excepción no manejada dentro de MongoDB. Se ejecuta una parada controlada de la instancia
45	No se puede crear el fichero mongod.lock o bloquear leer los datafiles
48	La instancia se ha parado porque el socket del puerto se ha caído
100	Excepción no recuperable, la instancia se para de forma abrupta

https://docs.mongodb.com/manual/reference/exit-codes/

Troubleshooting básico de instancias que no arrancan

- Existe fichero mongod.lock que no ha sido borrado
- El puerto está en uso
- No tiene permisos suficientes para escribir en el directorio de dbpath o logs
- Si se hace fork del proceso, no se puede escribir el fichero de pid
- El storage engine configurado no es el correcto

PRACTIQUEMOS

Checklist para troubleshooting de Replica Sets con comportamientos extraños

- Comprobar el estado del replica set (rs.status())
- Comprobar el lag de la replicación con los secundarios(rs.printSlaveReplicationInfo())
- Latencia entre los nodos que componen el replica set
- Comprobación del tamaño del Oplog para tener suficiente ventana de failover
- Sincronización de relojes entre las instancias (siempre usar NTP)
- Operativa de emergencia para forzar la reconfiguración con un nodo que no está disponible

GRACIAS

{ paradigma