

mongoDB®

# REPLICATION (複寫)



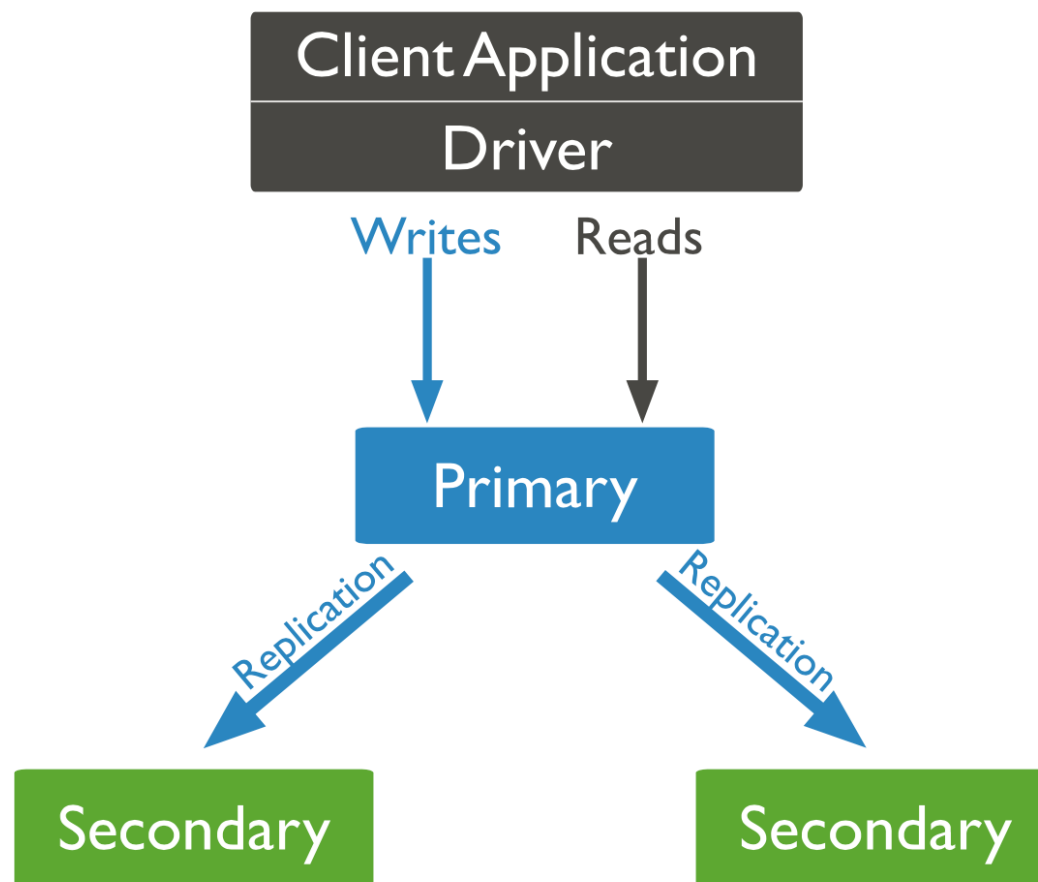
朱克剛

# 基本架構

一台 Primary、兩台 Secondary

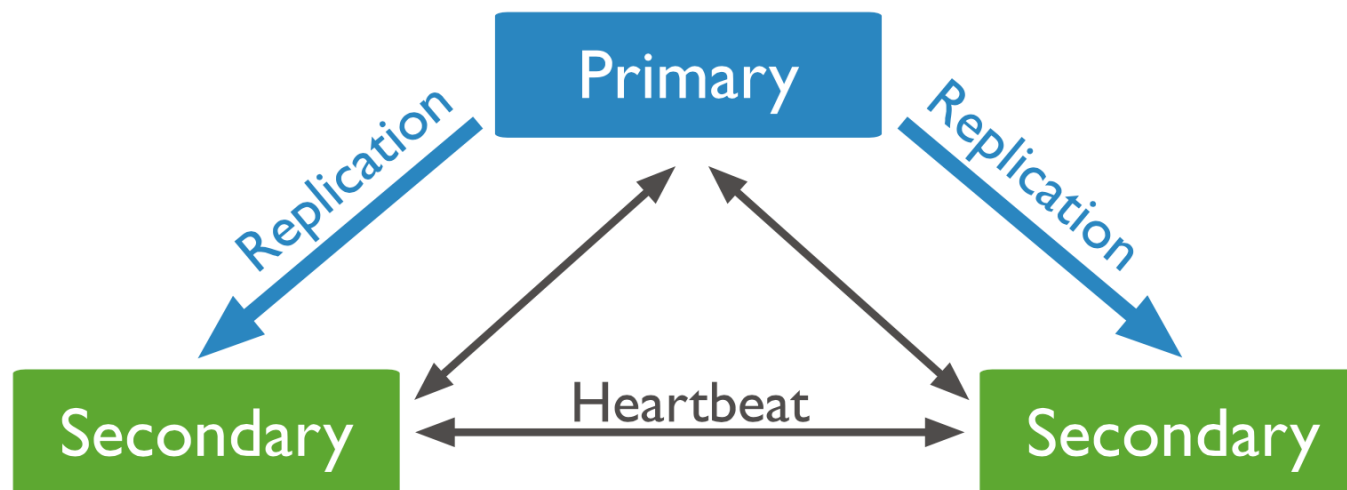
讀寫操作都在 Primary

一個複寫集最多可以有50個成員，  
但只有7個可以投票

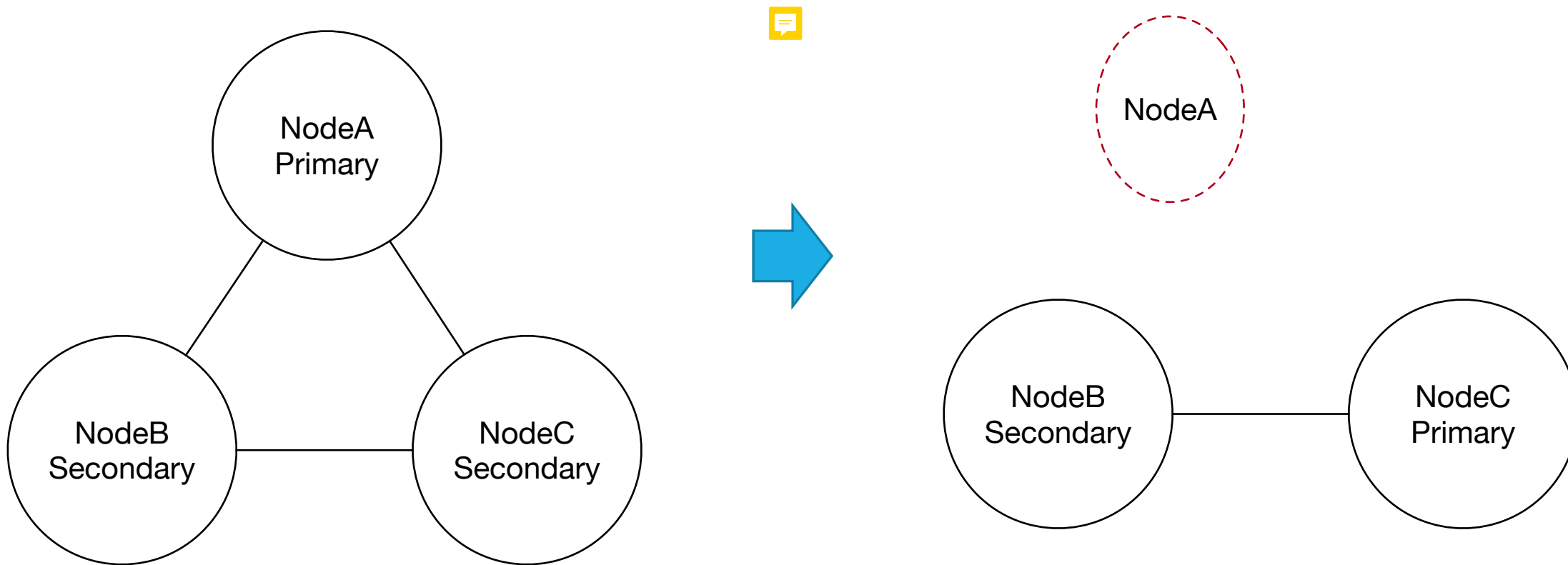


# 心跳

同一複寫集內的電腦彼此間透過 **heartbeat** 來確認對方是否正常運作  
預設心跳時間 **2 秒**，且需在 **10 秒**內回應



# 奇數節點 – 死掉一台沒事



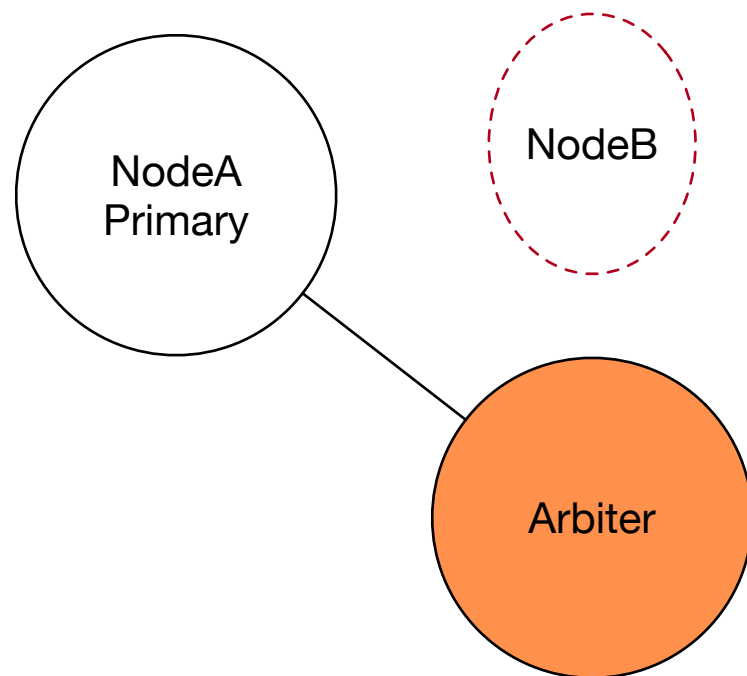
# 偶數節點 – 死掉一台



# 仲裁

## 作用

- 維持複寫集成員為奇數
- 節省複寫集成本
  - Arbiter 隨使用個樹莓派就搞定



NodeA 給自己一票，Arbiter 也給 NodeA 一票，共得到 2 票，得票率  $2/3$  超過一半，NodeA 轉成 Primary

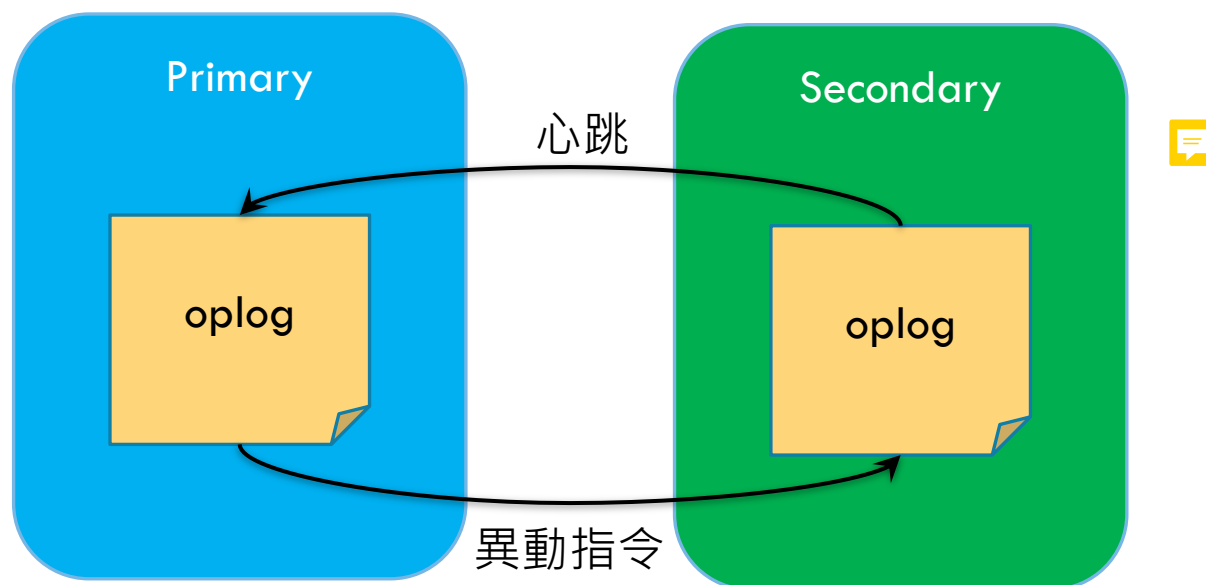
# 容錯能力

成員數	Primary 最小得票數	容忍損壞
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3

複寫集成員不應  
該是偶數

# OPLOG

資料異動時，Primary將異動指令寫入oplog資料表，Secondary 在心跳時將異動資料從 oplog 資料表拉到自己的 oplog，然後再執行異動指令





# 本機模擬 - PSS

先用 `mkdir` 建立資料庫位置

- NodeA 的資料庫位於 `data/dbA`
- NodeB 的資料庫位於 `data/dbB`
- NodeC 的資料庫位於 `data/dbC`

啟動 server

- `$ mongod --port 20000 --dbpath data/dbA --replSet rs0`
- `$ mongod --port 20001 --dbpath data/dbB --replSet rs0`
- `$ mongod --port 20002 --dbpath data/dbC --replSet rs0`

真正部署在不同機器上

- `$ sudo mongod --bind_ip_all --replSet rs0`

Primary  
NodeA:20000

Secondary  
NodeB:20001

Secondary  
NodeC:20002

# 連線 PRIMARY

## 連線

- `$ mongo --host localhost --port 20000`
- 或是 `$ mongo localhost:20000`

## 第一次執行

- `rs.initiate()`

## 增加節點

- `rs.add("localhost:20001")`
- `rs.add("localhost:20002")`



增加仲裁者指令  
`rs.addArb("ip:port")`

## 察看目前狀況

- `rs.status()`



# 連線SECONDARY

要察看在 secondary 的資料必須先下以下指令

```
rs.secondaryOk()
```

k 小寫

# ARBITER



唯一目的：投票

複寫集需要要有四個成員（一個Primary + 三個Secondary）後才能加入Arbiter


PSA架構可能會出現資料遺失、嚴重延遲或是永久延遲的狀況，因此預設不允許 PSA 架構


```
if [ (#arbiters > 0) AND (#non-arbiters <= majority(#voting-nodes)) ]  
    defaultWriteConcern = { w: 1 }  
else  
    defaultWriteConcern = { w: "majority" }
```



# PSA



> use admin 

```
db.adminCommand({  
  'setDefaultRWConcern': 1,  
  'defaultWriteConcern': {  
    'w': 1  
  }  
}) 
```

```
db.adminCommand({  
  'setDefaultRWConcern': 1,  
  'defaultWriteConcern': {  
    'w': 'majority'  
  }  
}) 
```

```
db.adminCommand({  
  'setDefaultRWConcern': 1,  
  'defaultWriteConcern': {  
    'w': 'majority',   
    'wtimeout': 2000  
  }   
})
```

Primary  
NodeA:20000

Secondary  
NodeB:20001

Arbiter  
NodeC:20003

Primary與Secondary都正常運作時，使用第二種  
Primary或Secondary有任何一個故障，使用第一種  
Arbiter不論是否正常運作，都不用修改

# 尋找並連線到PRIMARY

先連線到複寫集中的任何主機，然後下以下指令就會自動連到 PRIMARY 主機

```
db=connect(rs.isMaster().primary)
```

isMaster 這個字遲早  
會改成 isPrimary

# 移除節點

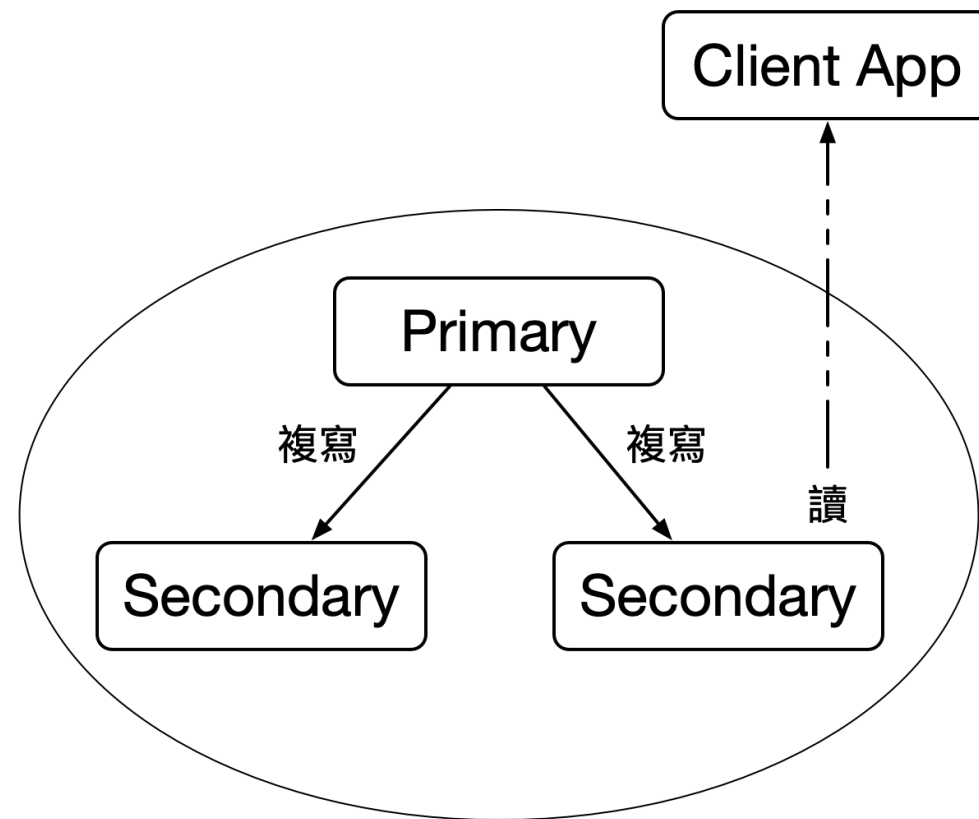
```
rs.remove("hostname:port")
```

# 讀取偏好

雖然預設是從 Primary 讀取資料，但必要時也可以從 Secondary 讀取資料

```
> db.getMongo().setReadPref('secondary')
```

```
import pymongo  
hosts = ['localhost:20000', 'localhost:20001', 'localhost:20002']  
client = pymongo.MongoClient(hosts, readPreference='secondary')
```





# 參數種類

偏好參數	說明
primary	只能從Primary讀取資料。
primaryPreferred	Primary 優先，若複寫集中沒有Primary存在，則改由Secondary讀取。
secondary	只能從Secondary讀取。
secondaryPreferred	Secondary 優先，若複寫集中Secondary不存在，則改由Primary讀取。
nearest	從評分項目最好的成員中隨機挑選一個作為資料讀取對象，不論該成員是Primary還是Secondary。評分項目包含網路速度、硬碟I/O速度、CPU效能等。

# 大型複寫集部署

九個成員的複寫集包含一個Arbiter的建置方式

複寫集最多只有七個成員具有投票資格，若目前已有八個成員且要加入Arbiter時，必須取消其中一位的投票資格

```
cfg = rs.conf()  
cfg.members[6].votes = 0  
cfg.members[6].priority = 0  
rs.reconfig(cfg)
```