

## 2) Продемонструвати запис даних на *primary node* з різними Write Concern Levels

```
rs0:PRIMARY> // unacknowledged
rs0:PRIMARY> db.values.insertOne({value: "unack"}, {writeConcern: { w: 0 }});
{ "acknowledged" : false }
rs0:PRIMARY> // acknowledged 2 replicas
rs0:PRIMARY> db.values.insertOne({value: "ack2"}, {writeConcern: { w: 2 }});
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60b63c71b84960bfca812e0c")
}
rs0:PRIMARY> // majority of voting members (here votes are unspecified - all are voting)
rs0:PRIMARY> db.values.insertOne({value: "ack_majority"}, {writeConcern: { w: "majority" }});
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60b63c71b84960bfca812e0d")
}
rs0:PRIMARY> // journaled - no rollbacks, write on disk, not in memory
rs0:PRIMARY> db.values.insertOne({value: "ack_majority_j"}, {writeConcern: { w: "majority", j: true }});
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60b63c78b84960bfca812e0e")
}
rs0:PRIMARY> █
```

## 3) Продемонструвати Read Preference Modes: читання з *primary* і *secondary* node

```
rs0:PRIMARY> // Продемонструвати Read Preference Modes: читання з primary і secondary node
rs0:PRIMARY> db.values.find({value: "unack"}).readPref("primary");
{ "_id" : ObjectId("60b63c71b84960bfca812e0b"), "value" : "unack" }
rs0:PRIMARY> db.values.find({value: "unack"}).readPref("secondary");
{ "_id" : ObjectId("60b63c71b84960bfca812e0b"), "value" : "unack" }
rs0:PRIMARY> █
```

## 4) Спробувати зробити запис з однією відключеною ногою та *write concern* рівнім 3 та нескінченим таймаутом. Спробувати під час таймаута включити відключену ноду

```
rs0:PRIMARY> db.values.insertOne({value: "ack2"}, {writeConcern: { w: 2 }});
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60b63da6529effaec5dbe8aa")
}
rs0:PRIMARY> db.values.insertOne({value: "ack3"}, {writeConcern: { w: 3 }});
^C
do you want to kill the current op(s) on the server? (y/n): █
```

5) Продемонстрував перевибори primary node в відключивши поточною primary (Replica Set Elections)

```
olekthunder@mellon ~ → mongo --host localhost:8001,localhost:8002,localhost:8003 --eval "JSON.stringify(rs.status())" --quiet | jq -r ".members[]|{stateStr,name}"
{
  "stateStr": "PRIMARY",
  "name": "m1:8001"
}
{
  "stateStr": "SECONDARY",
  "name": "m2:8002"
}
{
  "stateStr": "SECONDARY",
  "name": "m3:8003"
}
```

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] → docker-compose rm -s -v mong
o1 && date
Stopping m1 ... done
Going to remove m1
Are you sure? [yN] y
Removing m1 ... done
Вівторок, 1 червня 2021 17:40:13 +0300
olekthunder@mellon dist_systems_labs/labs/lab4 [master] → date
Вівторок, 1 червня 2021 17:40:21 +0300
olekthunder@mellon dist_systems_labs/labs/lab4 [master] → date
Вівторок, 1 червня 2021 17:40:23 +0300
olekthunder@mellon dist_systems_labs/labs/lab4 [master] → mongo --host localhost:8001,
localhost:8002,localhost:8003 --eval "JSON.stringify(rs.status())" --quiet | jq -r ".m
embers[]|{stateStr,name}"
{
  "stateStr": "(not reachable/healthy)",
  "name": "m1:8001"
}
{
  "stateStr": "PRIMARY",
  "name": "m2:8002"
}
{
  "stateStr": "SECONDARY",
  "name": "m3:8003"
}
olekthunder@mellon dist_systems_labs/labs/lab4 [master] → 10 seconds electionTimeoutMS
[]
```

```
s0:PRIMARY> db.values.insertOne({value: "after_m1_killed"});
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60b6482c590a5f85c807748d")
}
s0:PRIMARY> db.values.find({value: "after_m1_killed"});
{ "_id" : ObjectId("60b6482c590a5f85c807748d"), "value" : "after_m1_killed" }
s0:PRIMARY> []
```

6) Привести кластер до неконсистентного стану користуючись моментом часу коли *primary node* не відразу помічає відсутність *secondary node*

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] → mongo --host localhost:8001,localhost:8002,localhost:8003 --eval "JSON.stringify(rs.status())" --quiet | jq -r ".members[]|{stateStr,name}"
{
  "stateStr": "SECONDARY",
  "name": "m1:8001"
}
{
  "stateStr": "PRIMARY",
  "name": "m2:8002"
}
{
  "stateStr": "SECONDARY",
  "name": "m3:8003"
}

olekthunder@mellon dist_systems_labs/labs/lab4 [master] → mongo localhost:8001 --eval "rs.slaveOk();db.values.find({value: 'after_m1_killed'})" --quiet
{ "_id" : ObjectId("60b6482c590a5f85c807748d"), "value" : "after_m1_killed" }
olekthunder@mellon dist_systems_labs/labs/lab4 [master] →
```

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] → docker-compose rm -s -v mong
o2 mongo3
Stopping m3 ... done
Stopping m2 ... done
Going to remove m3, m2
Are you sure? [yN] y
Removing m3 ... done
Removing m2 ... done
olekthunder@mellon dist_systems_labs/labs/lab4 [master] →

rs0:PRIMARY>
rs0:PRIMARY> db.values.insertOne({value: "aftersecondaryded"});
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60b64dc7ed4809bbeb5acf0")
}
rs0:PRIMARY>
```

```
rs0:SECONDARY> db.values.find({value: "aftersecondaryded"}).readConcern("local")
{ "_id" : ObjectId("60b64dc7ed4809bbeb5acf0"), "value" : "aftersecondaryded" }
rs0:SECONDARY> db.values.find({value: "aftersecondaryded"}).readConcern("majority")

```

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] → mongo --host localhost:8001,localhost:8002,localhost:8003 --eval "JSON.stringify(rs.status())" --quiet | jq -r ".members[]|{stateStr,name}"
{
  "stateStr": "(not reachable/healthy)",
  "name": "m1:8001"
}
{
  "stateStr": "PRIMARY",
  "name": "m2:8002"
}
{
  "stateStr": "SECONDARY",
  "name": "m3:8003"
}
```

```
rs0:SECONDARY> rs.slaveOk();
rs0:SECONDARY> db.values.find();
{ "_id" : ObjectId("60b64dc7ed4809bbeb5acf0"), "value" : "aftersecondaryded" }
rs0:SECONDARY> // guess it should dissapear but I didn't understand the task fully
```

7) Показати відмінності в поведінці між рівнями `readConcern: {level: <"majority"|"local"|"linearizable">}`

```
rs0:SECONDARY> db.values.find({value: "aftersecondaryded"}).readConcern("local")
{ "_id" : ObjectId("60b64dc7ed4809bbeb5acf0"), "value" : "aftersecondaryded" }
rs0:SECONDARY> db.values.find({value: "aftersecondaryded"}).readConcern("majority")
[]
```

Linearizable works only on primary node

8)

```
rs0:PRIMARY> cfg.members[0].priority = 0
0
rs0:PRIMARY> cfg.members[0].hidden = true
true
rs0:PRIMARY> cfg.members[0].slaveDelay = 3600
3600
rs0:PRIMARY> rs.reconfig(cfg)
{
  "ok" : 1,
  "$clusterTime" : {
    "clusterTime" : Timestamp(1622561218, 1),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    },
    "operationTime" : Timestamp(1622561218, 1)
  }
}
rs0:PRIMARY> []
```

```
rs0:PRIMARY> db.values.insertOne({value: "cfg"});
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60b651e98e045357dd97a59a")
}
rs0:PRIMARY> db.values.find();
{ "_id" : ObjectId("60b64dc7ed4809bbeb5acf0"), "value" : "aftersecondaryded" }
{ "_id" : ObjectId("60b651e98e045357dd97a59a"), "value" : "cfg" }
```

```
rs0:SECONDARY> db.values.find().readPref("secondary");
{ "_id" : ObjectId("60b64dc7ed4809bbeb5acf0"), "value" : "aftersecondaryded" }
rs0:SECONDARY> []
```