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 нодою та *write concern* рівнім 3 та нескінченім таймаутом. Спробувати під час таймаута включити відключену ноду

- чи данні записались якщо запис було перевано?
- якщо під час очикування підключити 3-ю ноду чи завершиться запис?

Виключаємо одну ноду

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜ docker-compose stop mongo2
Stopping m2 ... done
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜
```

Запис с writeConcern = 3 (працюючих ноди 2 - запис висить)

```
rs0:PRIMARY> db.values.insertOne({value: 1}, {writeConcern: {w: 3}})
```

Включаємо ноду знову

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜ docker-compose start mongo2
Starting mongo2 ... done
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜
```

Запис відвисає, проходить

```
rs0:PRIMARY> db.values.insertOne({value: 1}, {writeConcern: {w: 3}})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("60bf37d8dbbaad5991ac8b66")
}
rs0:PRIMARY>
```

Виключаємо знову одну ноду

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜ docker-compose stop mongo2
Stopping m2 ... done
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜
```

Виконуємо аналогічну операцію

```
rs0:PRIMARY> db.values.insertOne({value: 2}, {writeConcern: {w: 3}})
```

Зупиняємо її

```
rs0:PRIMARY> db.values.insertOne({value: 2}, {writeConcern: {w: 3}})
^C
do you want to kill the current op(s) on the server? (y/n): y
2021-06-08T12:28:56.754+0300 I  CONTROL  [main] shutting down with code:0
```

Включаємо знову

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜ docker-compose start mongo2
Starting mongo2 ... done
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜
```

Перевіряємо чи записалось значення - записалось.

```
rs0:PRIMARY> db.values.findOne({value: 2})
{ "_id" : ObjectId("60bf382bdbbaad5991ac8b67"), "value" : 2 }
```

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

```
olekthunder@mellon ~ ➜ mongo --host localhost:8001,localhost:8002,localhost:8003 --eva
l "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
```

```
rs0:PRIMARY> db.values.insertOne({value: "after_m1_killed"});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("60b6482c590a5f85c807748d")
}
rs0:PRIMARY> db.values.find({value: "after_m1_killed"});
{ "_id" : ObjectId("60b6482c590a5f85c807748d"), "value" : "after_m1_killed" }
rs0:PRIMARY>
```

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

Виключаємо m2,m3, записуємо на m1. Записано.

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ✗ docker network disconnect mongo-rs m2
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜ docker network disconnect mongo-rs m3
```

```
rs0:PRIMARY> db.values.find()
rs0:PRIMARY> db.values.insertOne({foo: "bar"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("60bf41d0cfcb13dbb7ee0549")
}
```

Виключаємо m1, включаємо m2, m3

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜ docker network disconnect mongo-rs m1
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜ docker network connect mongo-rs m3
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜ docker network connect mongo-rs m2
```

Чекаємо результатів виборів - m3 primary

Включаємо m1

```
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜ docker network connect mongo-rs m1
olekthunder@mellon dist_systems_labs/labs/lab4 [master] ➜
```

Перевіряємо чи залишився запис на m1:

```
rs0:SECONDARY> db.values.find()
rs0:SECONDARY>
```

Запис зник

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

```
rs0:SECONDARY> db.values.find({value: "aftersecondaryded"}).readConcern("local")
{ "_id" : ObjectId("60b64dc7ed4809bbecb5acf0"), "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,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "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("60b64dc7ed4809bbecb5acf0"), "value" : "aftersecondaryded" }
{ "_id" : ObjectId("60b651e98e045357dd97a59a"), "value" : "cfg" }
```

```
rs0:SECONDARY> db.values.find().readPref("secondary");
{ "_id" : ObjectId("60b64dc7ed4809bbecb5acf0"), "value" : "aftersecondaryded" }
rs0:SECONDARY>
```