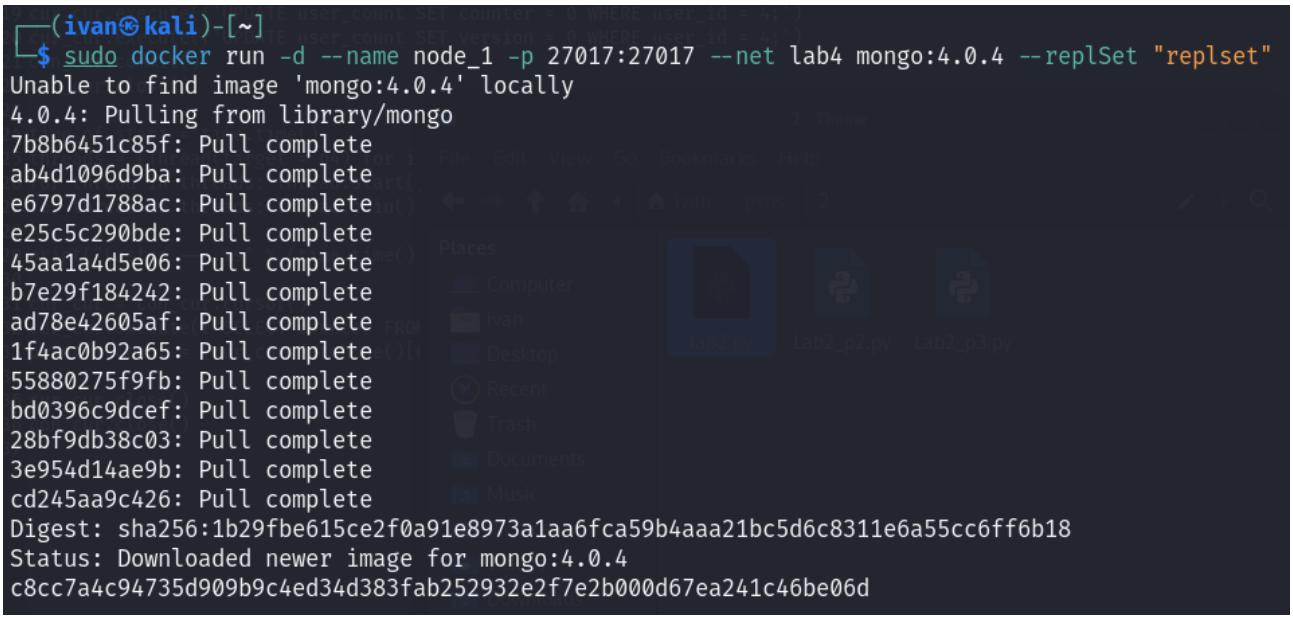
Завдання:

І Налаштування реплікації

1. Налаштувати реплікацію в конфігурації: Primary with Two Secondary Members (P-S-S) (всі ноди можуть бути запущені як окремі процеси або у Docker контейнерах) - http://docs.mongodb.org/manual/core/replica-set-architecture-three-members/

    - Deploy a Replica Set for Testing and Development- http://docs.mongodb.org/manual/tutorial/deploy-replica-set-for-testing/
    - http://www.tugberkugurlu.com/archive/setting-up-a-mongodb-replica-set-with-docker-and-connecting-to-it-with-a--net-core-app



```
> rs.initiate({ _id: "replset", members: [ {_id: 0, host: "node_1"}, {_id: 1, host: "node_2"}, {_id: 2, host: "node_3"} ] })
{
        "ok" : 1,
        "operationTime" : Timestamp(1734086150, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1734086150, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
replset:SECONDARY>
replset:PRIMARY>
```

```
"members" : [
        {
                "_id" : 0,
                "name" : "node_1:27017",
                "health" : 1,
                "state" : 1,
                "stateStr" : "PRIMARY",
                "uptime" : 301
```

```
{
        "_id" : 1,
        "name" : "node_2:27017",
        "health" : 1,
        "state" : 2,
        "stateStr" : "SECONDARY",
        "uptime" : 45,
```

```
{
        "_id" : 2,
        "name" : "node_3:27017",
        "health" : 1,
        "state" : 2,
        "stateStr" : "SECONDARY",
```

2. Спробувати зробити запис з однією відключеною нодою та *write concern* рівнім 3 та нескінченім таймаутом. Спробувати під час таймаута включити відключену ноду

```
┌──(ivan㉿kali)-[~]
└─$ sudo docker ps -a
[sudo] password for ivan:
CONTAINER ID   IMAGE                               COMMAND                  CREATED          STATUS                     PORTS                                                                      NAMES
dc861d67b4bd   mongo:4.0.4                         "docker-entrypoint.s…"   6 minutes ago    Up 6 minutes               0.0.0.0:27019→27017/tcp, :::27019→27017/tcp                                node_3
38bcd5b0113a   mongo:4.0.4                         "docker-entrypoint.s…"   7 minutes ago    Up 7 minutes               0.0.0.0:27018→27017/tcp, :::27018→27017/tcp                                node_2
c8cc7a4c9473   mongo:4.0.4                         "docker-entrypoint.s…"   7 minutes ago    Up 7 minutes               0.0.0.0:27017→27017/tcp, :::27017→27017/tcp                                node_1
f545c29686e6   postgres                            "docker-entrypoint.s…"   17 hours ago     Up 17 hours                0.0.0.0:5432→5432/tcp, :::5432→5432/tcp                                    postgres
7189dc25bcdf   hazelcast/management-center:5.4.0   "bash ./bin/mc-start…"   21 hours ago     Exited (130) 18 hours ago                                                                             main_node
6a19c3dfaae6   nginx                               "/docker-entrypoint.…"   4 days ago       Exited (255) 27 hours ago  0.0.0.0:8000→80/tcp, :::8000→80/tcp, 0.0.0.0:8443→443/tcp, :::8443→443/tcp   nginx
f2eea4df5603   traefik:v2.10                       "/entrypoint.sh --ap…"   4 days ago       Exited (255) 27 hours ago  0.0.0.0:80→80/tcp, :::80→80/tcp, 0.0.0.0:443→443/tcp, :::443→443/tcp           traefik

┌──(ivan㉿kali)-[~]
└─$ sudo docker pause node_3
node_3

┌──(ivan㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE                               COMMAND                  CREATED          STATUS                     PORTS                                                                      NAMES
dc861d67b4bd   mongo:4.0.4                         "docker-entrypoint.s…"   7 minutes ago    Up 7 minutes (Paused)      0.0.0.0:27019→27017/tcp, :::27019→27017/tcp                                node_3
38bcd5b0113a   mongo:4.0.4                         "docker-entrypoint.s…"   7 minutes ago    Up 7 minutes               0.0.0.0:27018→27017/tcp, :::27018→27017/tcp                                node_2
c8cc7a4c9473   mongo:4.0.4                         "docker-entrypoint.s…"   8 minutes ago    Up 8 minutes               0.0.0.0:27017→27017/tcp, :::27017→27017/tcp                                node_1
f545c29686e6   postgres                            "docker-entrypoint.s…"   17 hours ago     Up 17 hours                0.0.0.0:5432→5432/tcp, :::5432→5432/tcp                                    postgres
7189dc25bcdf   hazelcast/management-center:5.4.0   "bash ./bin/mc-start…"   21 hours ago     Exited (130) 18 hours ago                                                                             main_node
6a19c3dfaae6   nginx                               "/docker-entrypoint.…"   4 days ago       Exited (255) 27 hours ago  0.0.0.0:8000→80/tcp, :::8000→80/tcp, 0.0.0.0:8443→443/tcp, :::8443→443/tcp   nginx
f2eea4df5603   traefik:v2.10                       "/entrypoint.sh --ap…"   4 days ago       Exited (255) 27 hours ago  0.0.0.0:80→80/tcp, :::80→80/tcp, 0.0.0.0:443→443/tcp, :::443→443/tcp           traefik

┌──(ivan㉿kali)-[~]
```

```
        }
}
replset:PRIMARY> db.lab4.insertOne({name: "Ivan Y"}, {writeConcern: {w: 3}})
```

Після включення:

```
┌──(ivan㉿kali)-[~]
└─$ sudo docker unpause node_3
node_3

┌──(ivan㉿kali)-[~]
└─$
```

```
        }
}
replset:PRIMARY> db.lab4.insertOne({name: "Ivan Y"}, {writeConcern: {w: 3}})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("675c0fc3524aed9586a0b626")
}
replset:PRIMARY>
```

3. Аналогічно попередньому пункту, але задати скінченний таймаут та дочекатись його закінчення. Перевірити чи данні записались і чи доступні на читання з рівнем *readConcern: “majority”*

```
┌──(ivan㉿kali)-[~]
└─$ sudo docker pause node_3
node_3

┌──(ivan㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                     PORTS                                           NAMES
dc861d67b4bd   mongo:4.0.4    "docker-entrypoint.s…"   30 minutes ago   Up 30 minutes (Paused)     0.0.0.0:27019->27017/tcp, :::27019->27017/tcp   node_3
38bcd5b0113a   mongo:4.0.4    "docker-entrypoint.s…"   30 minutes ago   Up 30 minutes              0.0.0.0:27018->27017/tcp, :::27018->27017/tcp   node_2
c8cc7a4c9473   mongo:4.0.4    "docker-entrypoint.s…"   31 minutes ago   Up 30 minutes              0.0.0.0:27017->27017/tcp, :::27017->27017/tcp   node_1
```

```
}
replset:PRIMARY> db.lab4.insertOne({name: "Ivan Y - P2"}, {writeConcern: {w: 3}, wtimeout: 200})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("675c149d524aed9586a0b627")
}
replset:PRIMARY> db.getCollection('lab4').find().readConcern('majority')
{ "_id" : ObjectId("675c0fc3524aed9586a0b626"), "name" : "Ivan Y" }
{ "_id" : ObjectId("675c149d524aed9586a0b627"), "name" : "Ivan Y - P2" }
replset:PRIMARY>
```

4. Продемонстрував перевибори primary node відключивши поточний primary (Replica Set Elections) - http://docs.mongodb.org/manual/core/replica-set-elections/

   - і що після відновлення роботи старої primary на неї реплікуються нові дані, які з'явилися під час її простою

```
┌──(ivan㉿kali)-[~]
└─$ sudo docker unpause node_3
node_3

┌──(ivan㉿kali)-[~]
└─$ sudo docker pause node_1
node_1

┌──(ivan㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                     PORTS                                           NAMES
dc861d67b4bd   mongo:4.0.4    "docker-entrypoint.s…"   33 minutes ago   Up 33 minutes              0.0.0.0:27019->27017/tcp, :::27019->27017/tcp   node_3
38bcd5b0113a   mongo:4.0.4    "docker-entrypoint.s…"   33 minutes ago   Up 33 minutes              0.0.0.0:27018->27017/tcp, :::27018->27017/tcp   node_2
c8cc7a4c9473   mongo:4.0.4    "docker-entrypoint.s…"   34 minutes ago   Up 34 minutes (Paused)     0.0.0.0:27017->27017/tcp, :::27017->27017/tcp   node_1
```

```
┌──(ivan㉿kali)-[~]
└─$ sudo docker exec -it node_3 mongo
MongoDB shell version v4.0.4
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("be20a18d-33ad-46aa-9598-618602b02b37") }
MongoDB server version: 4.0.4
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2024-12-13T10:32:26.104+0000 I STORAGE  [initandlisten]
2024-12-13T10:32:26.104+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2024-12-13T10:32:26.104+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2024-12-13T10:32:26.598+0000 I CONTROL  [initandlisten]
2024-12-13T10:32:26.598+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2024-12-13T10:32:26.598+0000 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2024-12-13T10:32:26.598+0000 I CONTROL  [initandlisten]
2024-12-13T10:32:26.598+0000 I CONTROL  [initandlisten]
2024-12-13T10:32:26.598+0000 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2024-12-13T10:32:26.598+0000 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2024-12-13T10:32:26.598+0000 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

replset:PRIMARY>
```

```
replset:PRIMARY> db.lab4.insertOne({name: "Ivan Y - P3"}, {writeConcern: {w: 3}, wtimeout: 200})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("675c156d4c140a85421e1409")
}
replset:PRIMARY> db.getCollection('lab4').find().readConcern('majority')
{ "_id" : ObjectId("675c0fc3524aed9586a0b626"), "name" : "Ivan Y" }
{ "_id" : ObjectId("675c149d524aed9586a0b627"), "name" : "Ivan Y - P2" }
{ "_id" : ObjectId("675c156d4c140a85421e1409"), "name" : "Ivan Y - P3" }
replset:PRIMARY>
```

```
┌──(ivan㉿kali)-[~]
└─$ sudo docker unpause node_1
node_1

┌──(ivan㉿kali)-[~]
└─$
```

II Аналіз продуктивності та перевірка цілісності

Аналогічно попереднім завданням, необхідно буде створити колекцію (таблицю) з каунтером лайків. Далі з 10 окремих клієнтів одночасно запустити інкрементацію каунтеру лайків по 10_000 на кожного клієнта з різними опціями взаємодії з MongoDB.

Для того, щоб не було lost updates, для оновлення каунтера необхідно використовувати функцію findOneAndUpdate()

Приклад використання:

db.grades.findOneAndUpdate(

{ "name" : "R. Stiles" },

{ $inc: { "points" : 5 } }

)

https://www.mongodb.com/docs/manual/reference/method/db.collection.findOneAndUpdate/#update-a-document

5. Вказавши у парметрах *findOneAndUpdate writeConcern = 1* (це буде означати, що запис іде тільки на Primary ноду і не чекає відповіді від Secondary), запустіть 10 клієнтів з інкрементом по 10_000 на кожному з них. Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - 100К

```
┌──(ivan㉿kali)-[~/pvns/4/1]
└─$ python concern_with_primary.py
Increment completed with writeConcern=1 and Primary node up
Lasted —— 39.18419003486633
```

```
replset:PRIMARY> db.getCollection('like_counters').find().readConcern('majority')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 100000 }
replset:PRIMARY>
```

```
replset:SECONDARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 100000 }
replset:SECONDARY>
```

6. Вказавши у парметрах *findOneAndUpdate writeConcern = majority* (це буде означати, що Primary чекає поки значення запишется на більшість нод), запустіть 10 клієнтів з інкрементом по 10_000 на кожному з них. Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - 100К

```
┌──(ivan㉿kali)-[~/pvns/4/1]
└─$ python majority_with_primary.py
Increment completed with writeConcern=majority and Primary node up
Lasted —— 42.61364817619324

┌──(ivan㉿kali)-[~/pvns/4/1]
└─$
```

```
replset:PRIMARY> db.getCollection('like_counters').find().readConcern('majority')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 100000 }
replset:PRIMARY>
```

```
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 0 }
replset:SECONDARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 100000 }
replset:SECONDARY>
```

7. Повторно запустить код при *writeConcern = 1*, але тепер під час роботи відключіть Primary ноду і подивитись що буде обрана інша Primary нода, яка продовжить обробку запитів, і чи кінцевий результат буде коректним.

Попередньо зменшив electionTimeoutMillis на всіх нодах:

```
replset:SECONDARY> rs.conf().settings.electionTimeoutMillis = 2000
2000
replset:SECONDARY> rs.reconfig(rs.conf(), {force: true})
{
        "ok" : 1,
        "operationTime" : Timestamp(1734128365, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1734128365, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

```
┌──(root㉿kali)-[/home/ivan/pvns/4/1]
└─# python concern_without_primary.py
node_3
Increment completed with writeConcern=1 and Primary node switched
Lasted ── 53.72220849990845 seconds
```

Нода, що світчнулась:

```
replset:SECONDARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 79756 }
replset:PRIMARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 82246 }
replset:PRIMARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 100000 }
replset:PRIMARY>
```

Інша жива нода:

```
replset:SECONDARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 53752 }
replset:SECONDARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 59107 }
replset:SECONDARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 100000 }
replset:SECONDARY>
```

8. Повторно запустить код при *writeConcern = majority*, але тепер під час роботи відключіть Primary ноду і подивитись що буде обрана інша Primary нода, яка продовжить обробку запитів, і чи кінцевий результат буде коректним.

При *writeConcern = 1* деякі записи можуть губитись під час раптового відключення. При *writeConcern = majority* має виходити очікуваний результат.

```
┌──(root㉿kali)-[/home/ivan/pvns/4/1]
└─# python majority_without_primary.py
node_3
Increment completed with writeConcern=majority and Primary node switched
Lasted ── 52.708094120025635 seconds
```

Нода, яка свічнулась:

```
replset:SECONDARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 0 }
replset:SECONDARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 100000 }
replset:PRIMARY> db.getCollection('like_counters').find().readPref('secondary')
{ "_id" : ObjectId("675c2eebbe7e08d46fa8d6b2"), "user_id" : 1, "likes" : 100000 }
```

В моєму випадку (принаймні для одного запуску), і при *writeConcern = 1*, і при *writeConcern = majority* кінцеві значення каунтерів були коректними.