Homework 6. Replication in MongoDB

1) Set up replication in the configuration: Primary with Two Secondary Members.

Commands were taken from [this tutorial](#).

Create a network and run docker containers. Expose ports to connect to the mongo nodes via web-ui.

```
$ docker network create my-mongo-cluster
$ docker run --name mongo-node1 -d --net my-mongo-cluster -p 27017:27017 mongo
--replSet "rs0"
$ docker run --name mongo-node2 -d --net my-mongo-cluster -p 27027:27017 mongo
--replSet "rs0"
$ docker run --name mongo-node3 -d --net my-mongo-cluster -p 27037:27017 mongo
--replSet "rs0"
```

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Databases
/HW6_MongoDB_Replication$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS       NAMES
dcbe3bf63786   mongo     "docker-entrypoint.s…"   3 seconds ago    Up 2 seconds    27017/tcp   mongo-node3
d9601e7d79b4   mongo     "docker-entrypoint.s…"   8 seconds ago    Up 7 seconds    27017/tcp   mongo-node2
6bf579664f7a   mongo     "docker-entrypoint.s…"   23 seconds ago   Up 22 seconds   27017/tcp   mongo-node1
```

Connect to mongo-node1 to configure a replica set.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Databases
/HW6_MongoDB_Replication$ docker exec -it mongo-node1 mongosh
Current Mongosh Log ID: 647c635394b0f76b4cba2993
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongos
h+1.8.0
Using MongoDB:          6.0.5
Using Mongosh:          1.8.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb
.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.
```

Replica set config:

```
------
   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()
------

test> config = {
...       "_id" : "rs0",
...       "members" : [
...           {
...               "_id" : 0,
...               "host" : "mongo-node1:27017"
...           },
...           {
...               "_id" : 1,
...               "host" : "mongo-node2:27017"
...           },
...           {
...               "_id" : 2,
...               "host" : "mongo-node3:27017"
...           }
...       ]
...   }
{
  _id: 'rs0',
  members: [
    { _id: 0, host: 'mongo-node1:27017' },
    { _id: 1, host: 'mongo-node2:27017' },
    { _id: 2, host: 'mongo-node3:27017' }
  ]
}
```

```
{
  _id: 'rs0',
  members: [
    { _id: 0, host: 'mongo-node1:27017' },
    { _id: 1, host: 'mongo-node2:27017' },
    { _id: 2, host: 'mongo-node3:27017' }
  ]
}
test> rs.initiate(config)
{ ok: 1 }
rs0 [direct: other] test>

rs0 [direct: secondary] test> rs.status()
```

mongo-node 1 is a primary (from the rs.status() output):

```
members: [
  {
    _id: 0,
    name: 'mongo-node1:27017',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 221,
```

```
{
  _id: 1,
  name: 'mongo-node2:27017',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 20,
```

```
{
  _id: 2,
  name: 'mongo-node3:27017',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 20,
```

2) Demonstrate writing data to the primary node with different Write Concern Levels.

- Unacknowledged
- Acknowledged
- Journaled
- AcknowledgedReplica

Run a script as a docker container.

```
$ docker build -t dd_hw6 .
$ docker run --net my-mongo-cluster --rm dd_hw6 --write_concern <CONCERN_LEVEL>
```

```
denys_herasymuk@EPUALVIW07D6: ~   denys_herasymuk@EPUALVIW07D6: ~/U...   mongosh mongodb://127.0.0.1:27017/?d...
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --write_concern unacknowledged
Connected to the DB
Everything is written. Execution time: 0.5874872207641602.
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --write_concern acknowledged
Connected to the DB
Everything is written. Execution time: 0.8506736755371094.
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --write_concern journaled
Connected to the DB
Everything is written. Execution time: 7.335397720336914.
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --write_concern acknowledged_repl
ica
Connected to the DB
Everything is written. Execution time: 6.928675174713135.
```

As we can see from the execution time, journaled and acknowledged_replica write concerns take a lot of time, compared to acknowledged and unacknowledged levels.

3) Demonstrate Read Preference Modes: reading from primary and secondary nodes.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --read_preference primary
Connected to the DB
Records that contain the "Poland" country
{'name': 'Kevin White', 'address': '2713 Carter Cape Suite 183\nJanetville, MD 32660', 'country': 'Poland'}
{'name': 'Justin Quinn', 'address': 'Unit 0370 Box 7131\nDPO AP 17375', 'country': 'Poland'}
{'name': 'Mark Hawkins', 'address': '7395 Brent Underpass\nStevenbury, NY 61410', 'country': 'Poland'}
{'name': 'Keith Burnett', 'address': '223 Blankenship Run\nAlexanderhaven, NY 88058', 'country': 'Poland'}
{'name': 'Bridget Henry', 'address': '2424 Pittman Camp Apt. 302\nNew Tiffany, VI 29761', 'country': 'Poland'}
{'name': 'Adam King', 'address': '9587 Michael Place Apt. 459\nAmandaville, AZ 89729', 'country': 'Poland'}
Everything is written. Execution time: 0.0036580562591552734.
```

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --read_preference secondary
Connected to the DB
Records that contain the "Poland" country
{'name': 'Kevin White', 'address': '2713 Carter Cape Suite 183\nJanetville, MD 32660', 'country': 'Poland'}
{'name': 'Justin Quinn', 'address': 'Unit 0370 Box 7131\nDPO AP 17375', 'country': 'Poland'}
{'name': 'Mark Hawkins', 'address': '7395 Brent Underpass\nStevenbury, NY 61410', 'country': 'Poland'}
{'name': 'Keith Burnett', 'address': '223 Blankenship Run\nAlexanderhaven, NY 88058', 'country': 'Poland'}
{'name': 'Bridget Henry', 'address': '2424 Pittman Camp Apt. 302\nNew Tiffany, VI 29761', 'country': 'Poland'}
{'name': 'Adam King', 'address': '9587 Michael Place Apt. 459\nAmandaville, AZ 89729', 'country': 'Poland'}
Everything is written. Execution time: 0.003996610641479492.
```

** Ignore “Everything is written” log since it is a typo and it should not be there. The script in the above setting only reads from the replica set.

4) Try to write with one disabled node and write concern level 3 and infinite timeout. Try to turn on the disabled node during the timeout.

Disable one node.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker stop mongo-node3
mongo-node3
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
                    NAMES
12d8c12670a8   mongo     "docker-entrypoint.s…"   57 minutes ago   Up 57 minutes   0.0.0.0:27027->27017/tcp, ::
:27027->27017/tcp   mongo-node2
1ec5f7c44328   mongo     "docker-entrypoint.s…"   57 minutes ago   Up 57 minutes   0.0.0.0:27017->27017/tcp, ::
:27017->27017/tcp   mongo-node1
```

Write with write concern = 3.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --write_concern 3
Connected to the DB
Everything is written. Execution time: 40.44516706466675.
```

The write operation was waiting until I added the third node (execution time is big).

5) Similar to the previous task, but set a finite timeout and wait for it to end. Check whether the data has been written and is available for reading with readConcern level: “majority”.

Disable node 3.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED             STATUS              PORTS
                          NAMES
58af8ccc240e   mongo     "docker-entrypoint.s…"   25 minutes ago      Up 25 minutes       0.0.0.0:27037->27017/t
cp, :::27037->27017/tcp   mongo-node3
12d8c12670a8   mongo     "docker-entrypoint.s…"   About an hour ago   Up About an hour    0.0.0.0:27027->27017/t
cp, :::27027->27017/tcp   mongo-node2
1ec5f7c44328   mongo     "docker-entrypoint.s…"   About an hour ago   Up About an hour    0.0.0.0:27017->27017/t
cp, :::27017->27017/tcp   mongo-node1
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker stop mongo-node3
mongo-node3
```

Run the container with write concern = 3 and timeout = 10 seconds.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Dat
abases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --write_concern 3 --write_timeout
_ms 10000
Connected to the DB
Traceback (most recent call last):
  File "/app/mongodb_replication.py", line 97, in <module>
    write_to_db(write_concern, journaled, write_timeout_ms)
  File "/app/mongodb_replication.py", line 28, in write_to_db
    collection.insert_one(new_record)
  File "/usr/local/lib/python3.10/site-packages/pymongo/collection.py", line 628, in insert_one
    self._insert_one(
  File "/usr/local/lib/python3.10/site-packages/pymongo/collection.py", line 569, in _insert_one
    self.__database.client._retryable_write(acknowledged, _insert_command, session)
  File "/usr/local/lib/python3.10/site-packages/pymongo/mongo_client.py", line 1476, in _retryable_write
    return self._retry_with_session(retryable, func, s, None)
  File "/usr/local/lib/python3.10/site-packages/pymongo/mongo_client.py", line 1349, in _retry_with_session
    return self._retry_internal(retryable, func, session, bulk)
  File "/usr/local/lib/python3.10/site-packages/pymongo/_csot.py", line 105, in csot_wrapper
    return func(self, *args, **kwargs)
  File "/usr/local/lib/python3.10/site-packages/pymongo/mongo_client.py", line 1390, in _retry_internal
    return func(session, sock_info, retryable)
  File "/usr/local/lib/python3.10/site-packages/pymongo/collection.py", line 567, in _insert_command
    _check_write_command_response(result)
  File "/usr/local/lib/python3.10/site-packages/pymongo/helpers.py", line 221, in _check_write_command_response
    _raise_write_concern_error(wce)
  File "/usr/local/lib/python3.10/site-packages/pymongo/helpers.py", line 196, in _raise_write_concern_error
    raise WTimeoutError(error.get("errmsg"), error.get("code"), error)
pymongo.errors.WTimeoutError: waiting for replication timed out, full error: {'code': 64, 'codeName': 'WriteCon
cernFailed', 'errmsg': 'waiting for replication timed out', 'errInfo': {'wtimeout': True, 'writeConcern': {'w':
 3, 'j': False, 'wtimeout': 10000, 'provenance': 'clientSupplied'}}}
```

The above behavior of MongoClient is the same as described in the documentation (link).

Write Concern options:
(Only set if passed. No default values.)

- *w*: (integer or string) If this is a replica set, write operations will block until they have been replicated to the specified number or tagged set of servers. *w=<int>* always includes the replica set primary (e.g. w=3 means write to the primary and wait until replicated to **two** secondaries). Passing w=0 **disables write acknowledgement** and all other write concern options.
- *wTimeoutMS*: (integer) Used in conjunction with *w*. Specify a value in milliseconds to control how long to wait for write propagation to complete. If replication does not complete in the given timeframe, a timeout exception is raised. Passing wTimeoutMS=0 will cause **write operations to wait indefinitely**.

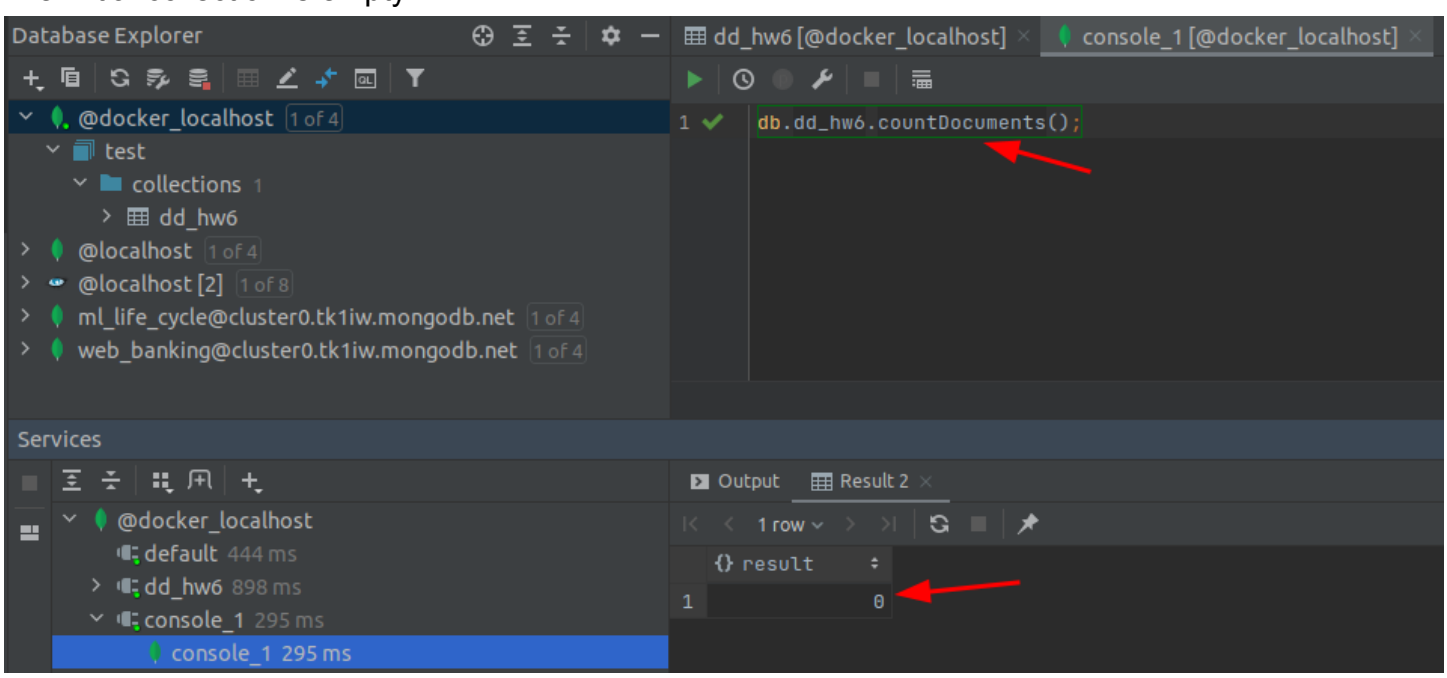
6) Demonstrated primary node re-elections by disabling the current primary (Replica Set Elections)

- and that after the old primary is restored, new data that appeared during its downtime is replicated to it

Initial nodes.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Databases/HW6_MongoDB_Replication$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                       NAMES
ad0dbe5ecedc   mongo     "docker-entrypoint.s…"   4 seconds ago   Up 4 seconds   0.0.0.0:27037->27017/tcp, :::27037->27017/tcp   mongo-node3
12d8c12670a8   mongo     "docker-entrypoint.s…"   2 hours ago     Up 2 hours     0.0.0.0:27027->27017/tcp, :::27027->27017/tcp   mongo-node2
1ec5f7c44328   mongo     "docker-entrypoint.s…"   2 hours ago     Up 2 hours     0.0.0.0:27017->27017/tcp, :::27017->27017/tcp   mongo-node1
```

The initial collection is empty.



Initial primary.

```
members: [
  {
    _id: 0,
    name: 'mongo-node1:27017',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 6116,
```

Let's disable mongo-node1.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Databases/HW6_MongoDB_Replication$ docker stop mongo-node1
mongo-node1
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Databases/HW6_MongoDB_Replication$
```

After the re-election, mongo-node2 is a *primary*.

```
{
  _id: 1,
  name: 'mongo-node2:27017',
  health: 1,
  state: 1,
  stateStr: 'PRIMARY',
  uptime: 6441,
```

mongo-node1 is marked as unhealthy.

```
members: [
  {
    _id: 0,
    name: 'mongo-node1:27017',
    health: 0,
    state: 8,
    stateStr: '(not reachable/healthy)',
    uptime: 0,
    optime: { ts: Timestamp({ t: 0, i: 0 }), t: Long("-1") },
    optimeDurable: { ts: Timestamp({ t: 0, i: 0 }), t: Long("-1") },
    optimeDate: ISODate("1970-01-01T00:00:00.000Z"),
    optimeDurableDate: ISODate("1970-01-01T00:00:00.000Z"),
```

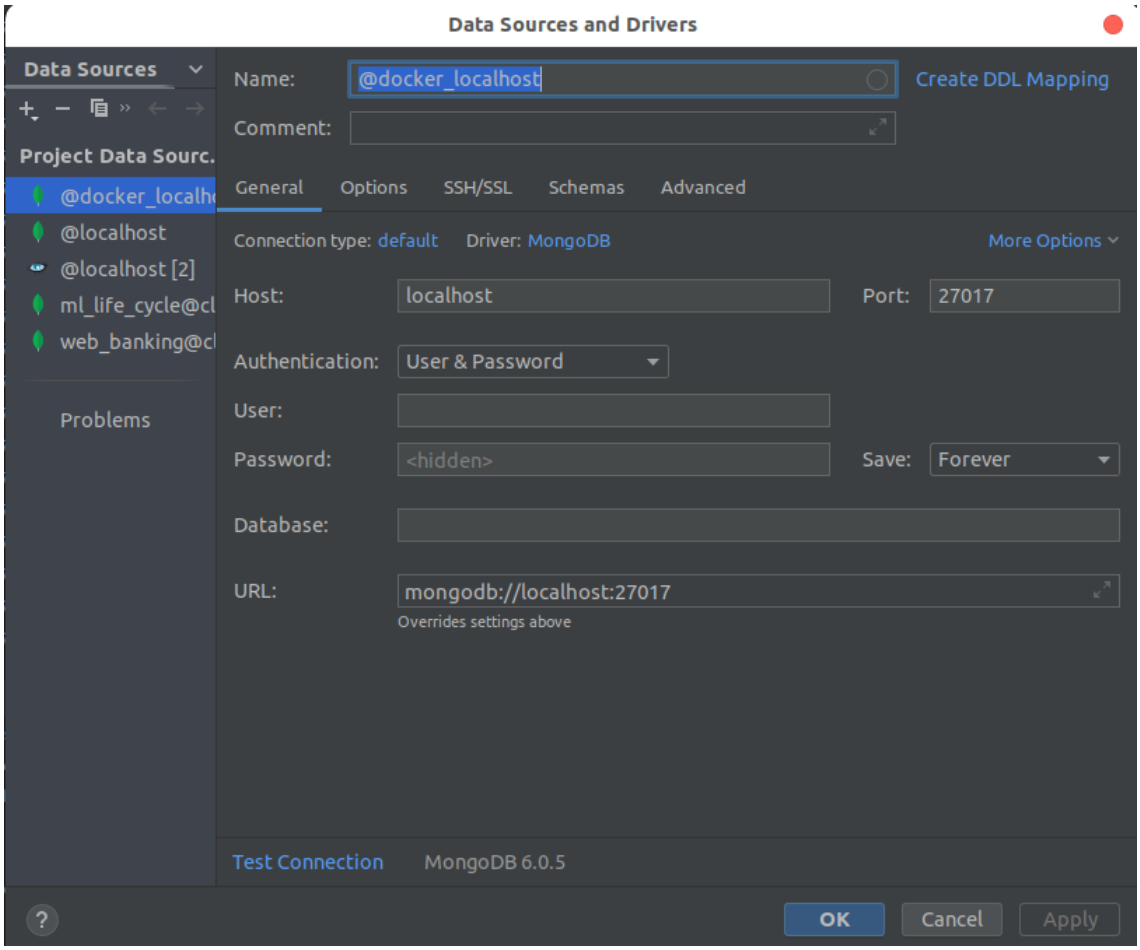
Write to the replica set without mongo-node1.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Databases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --write_concern acknowledged
Connected to the DB
Everything is written. Execution time: 0.8973305225372314.
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Databases/HW6_MongoDB_Replication$
```

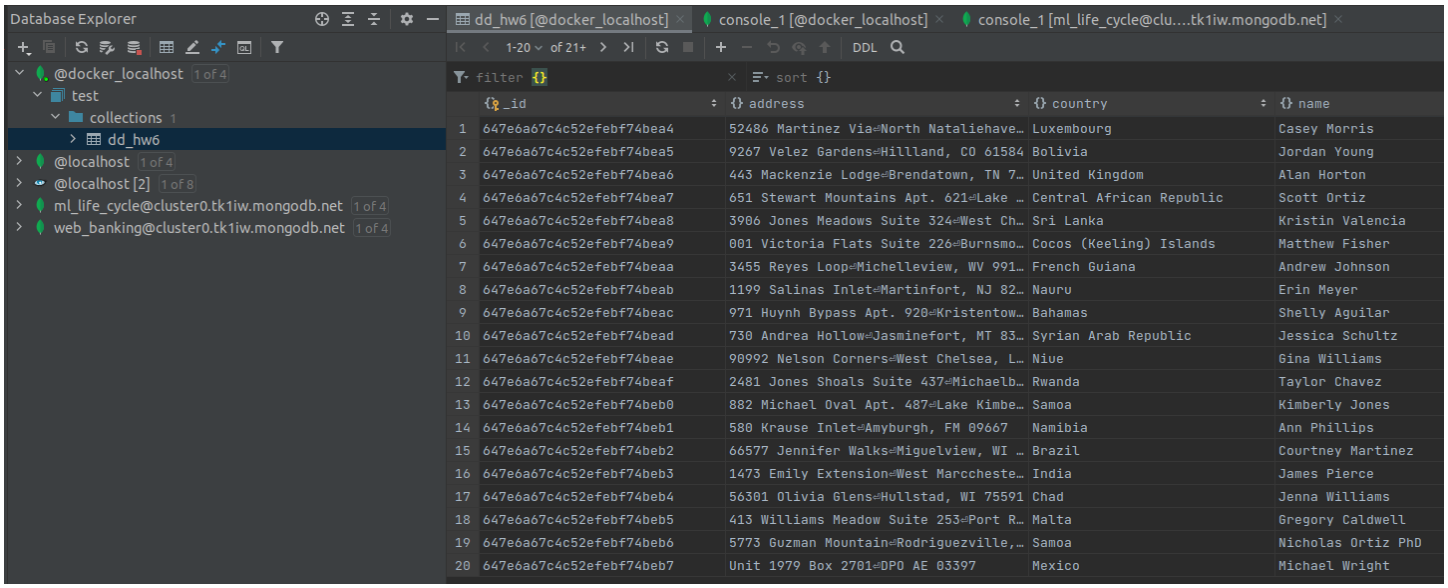
After enabling mongo-node1 back, we can see that this node is a secondary now.

```
members: [
  {
    _id: 0,
    name: 'mongo-node1:27017',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 55,
    optime: { ts: Timestamp({ t: 1686006505, i: 1 }), t: Long("2") },
    optimeDurable: { ts: Timestamp({ t: 1686006505, i: 1 }), t: Long("2") },
```

Since mongo-node1 has an exposed port to a local port 27017, we can connect to it via web-ui to check the data replication.



As we can see, despite being disabled for some time, mongo-node1 replicated the lost data (initial state of node 1 before disabling was no data at all).



	_id	address	country	name
1	647e6a67c4c52efebf74bea4	52486 Martinez Via⏎North Natalieheve…	Luxembourg	Casey Morris
2	647e6a67c4c52efebf74bea5	9267 Velez Gardens⏎Hillland, CO 61584	Bolivia	Jordan Young
3	647e6a67c4c52efebf74bea6	443 Mackenzie Lodge⏎Brendatown, TN 7…	United Kingdom	Alan Horton
4	647e6a67c4c52efebf74bea7	651 Stewart Mountains Apt. 621⏎Lake …	Central African Republic	Scott Ortiz
5	647e6a67c4c52efebf74bea8	3906 Jones Meadows Suite 324⏎West Ch…	Sri Lanka	Kristin Valencia
6	647e6a67c4c52efebf74bea9	001 Victoria Flats Suite 226⏎Burnsmo…	Cocos (Keeling) Islands	Matthew Fisher
7	647e6a67c4c52efebf74beaa	3455 Reyes Loop⏎Michelleview, WV 991…	French Guiana	Andrew Johnson
8	647e6a67c4c52efebf74beab	1199 Salinas Inlet⏎Martinfort, NJ 82…	Nauru	Erin Meyer
9	647e6a67c4c52efebf74beac	971 Huynh Bypass Apt. 920⏎Kristentow…	Bahamas	Shelly Aguilar
10	647e6a67c4c52efebf74bead	730 Andrea Hollow⏎Jasminefort, MT 83…	Syrian Arab Republic	Jessica Schultz
11	647e6a67c4c52efebf74beae	90992 Nelson Corners⏎West Chelsea, L…	Niue	Gina Williams
12	647e6a67c4c52efebf74beaf	2481 Jones Shoals Suite 437⏎Michaelb…	Rwanda	Taylor Chavez
13	647e6a67c4c52efebf74beb0	882 Michael Oval Apt. 487⏎Lake Kimbe…	Samoa	Kimberly Jones
14	647e6a67c4c52efebf74beb1	580 Krause Inlet⏎Amyburgh, FM 09667	Namibia	Ann Phillips
15	647e6a67c4c52efebf74beb2	66577 Jennifer Walks⏎Miguelview, WI …	Brazil	Courtney Martinez
16	647e6a67c4c52efebf74beb3	1473 Emily Extension⏎West Marccheste…	India	James Pierce
17	647e6a67c4c52efebf74beb4	56301 Olivia Glens⏎Hullstad, WI 75591	Chad	Jenna Williams
18	647e6a67c4c52efebf74beb5	413 Williams Meadow Suite 253⏎Port R…	Malta	Gregory Caldwell
19	647e6a67c4c52efebf74beb6	5773 Guzman Mountain⏎Rodriguezville,…	Samoa	Nicholas Ortiz PhD
20	647e6a67c4c52efebf74beb7	Unit 1979 Box 2701⏎DPO AE 03397	Mexico	Michael Wright

7) Bring the cluster to an inconsistent state using the moment when the *primary* node does not immediately notice the absence of the *secondary* node

- Disconnect two *secondary* nodes and write several records on *primary* (with w:1) within 5 seconds. Check that the records have been written.

Disable node 2 and 3

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distri
buted_Databases$ docker stop mongo-node2
mongo-node2
```

```
(base) denys_herasymuk@EPUALVIW07D6:~$ docker stop mongo-node3
mongo-node3
```

Write several records only on *primary*

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distri
buted_Databases$ docker run --net my-mongo-cluster --rm dd_hw6 --write_concern acknowledged
Connected to the DB
Everything is written. Execution time: 0.7968792915344238.
```

Records were successfully written

console_1 [@docker_localhost] | dd_hw6 [@docker_localhost] | console_1 [ml_life_cycle@clu....tk1iw.mongodb.net]

0 rows | DDL

filter {} | sort {}

	_id	address	country	name
1	6480f67084c50b5fa071bbb6	8921 Cook Drive Apt. 248⏎Powellmo…	Saint Kitts and Nevis	Melissa Wade
2	6480f67084c50b5fa071bbb7	30732 Spencer Oval⏎North Briannas…	Palestinian Territory	Brittany Davis
3	6480f67084c50b5fa071bbb8	08830 David Shoal⏎Lawrenceside, N…	Saint Lucia	Mark Mann
4	6480f67084c50b5fa071bbb9	35749 Janet Valley⏎Smithhaven, OH…	Heard Island and McDonald Islands	John Padilla
5	6480f67084c50b5fa071bbba	598 Koch Road Suite 139⏎Brianfurt…	Liberia	Autumn Bailey
6	6480f67084c50b5fa071bbbb	212 Gutierrez Drive⏎Port Davidfur…	Sierra Leone	Brenda Holmes
7	6480f67084c50b5fa071bbbc	30846 David Greens⏎Lake Lisafurt,…	Ireland	Michael Robinson
8	6480f67084c50b5fa071bbbd	7680 Newman Garden Suite 694⏎Thom…	Trinidad and Tobago	Angelica Pugh
9	6480f67084c50b5fa071bbbe	37311 Daniel Fall Suite 621⏎Shann…	Iran	Adriana Lee
10	6480f67084c50b5fa071bbbf	PSC 7914, Box 0849⏎APO AA 86203	Liberia	Maria Johnson
11	6480f67084c50b5fa071bbc0	4927 Michael Summit⏎Kristenberg, …	Christmas Island	Ashlee Davis
12	6480f67084c50b5fa071bbc1	851 Philip Island⏎Peterborough, R…	Malawi	Laura Turner
13	6480f67084c50b5fa071bbc2	78017 Joel Spurs Suite 784⏎New Ka…	Costa Rica	Wendy Gates
14	6480f67084c50b5fa071bbc3	48518 Donald Track Suite 149⏎Nort…	Taiwan	Jacqueline Armstrong
15	6480f67084c50b5fa071bbc4	81345 James Grove Apt. 695⏎Aaronl…	United States Minor Outlying Islands	Kimberly Knox
16	6480f67084c50b5fa071bbc5	0504 Jon Plains⏎Petersburgh, SD 1…	Cayman Islands	Erica Goodwin

- read these records with different levels of *read concern* - `readConcern: {level: <"majority"|"local"| "linearizable">}`

readConcern = "majority"

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Data
bases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --read_concern majority
```

Even after 10 minutes, I was still waiting. So we have an infinite wait until other nodes will be enabled.

readConcern = "local"

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Data
bases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --read_concern local
Connected to the DB
Top 10 records
{'name': 'Michael Finley', 'address': '35687 Curtis Summit\nGonzalezfurt, TX 20584', 'country': 'Bulgaria'}
{'name': 'Jennifer Gregory', 'address': '94956 Gerald Plaza\nLake Nicolasshire, MT 09812', 'country': 'Dominican
 Republic'}
{'name': 'Catherine Gibson', 'address': '381 Williams Harbor\nLake Shelby, MI 23901', 'country': 'Iran'}
{'name': 'Christopher Thompson', 'address': '324 Harding Forge Suite 901\nPort Ashley, CO 08339', 'country': 'Pa
lau'}
{'name': 'Richard Salazar', 'address': '2009 Michael Fall Suite 320\nSouth Stephenton, AL 96321', 'country': 'Be
nin'}
{'name': 'Brittany Nelson', 'address': '559 Lowe Burgs\nNorth Allenfort, CT 66775', 'country': 'Guinea'}
{'name': 'Bradley Rivera', 'address': '1024 Peters Crescent\nStephanietown, NC 56938', 'country': 'Comoros'}
{'name': 'Robert Lee', 'address': '390 Cindy Locks\nSouth Heidi, WI 84843', 'country': 'Tunisia'}
{'name': 'Mary Taylor', 'address': '23689 Jimenez Lodge\nSpencerstad, OR 51716', 'country': 'Romania'}
{'name': 'Joe Mckinney', 'address': '3619 Dorothy Pine Apt. 802\nNorth Bobby, KS 29684', 'country': 'Martinique'
}
```

As we can see, we could read the data from the primary that was alone in the replica set.

readConcern = “linearizable”

After 5 seconds the *primary* becomes a secondary when it is alone in the replica set. Using the ‘linearizable’ level, we can get an appropriate error in a few seconds without a long wait, as it was in case of the “majority” level.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Data
bases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --read_concern linearizable
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/site-packages/pymongo/mongo_client.py", line 1441, in _retryable_read
Connected to the DB
Top 10 records
    server = self._select_server(read_pref, session, address=address)
  File "/usr/local/lib/python3.10/site-packages/pymongo/mongo_client.py", line 1257, in _select_server
    server = topology.select_server(server_selector)
  File "/usr/local/lib/python3.10/site-packages/pymongo/topology.py", line 272, in select_server
    server = self._select_server(selector, server_selection_timeout, address)
  File "/usr/local/lib/python3.10/site-packages/pymongo/topology.py", line 261, in _select_server
    servers = self.select_servers(selector, server_selection_timeout, address)
  File "/usr/local/lib/python3.10/site-packages/pymongo/topology.py", line 223, in select_servers
    server_descriptions = self._select_servers_loop(selector, server_timeout, address)
  File "/usr/local/lib/python3.10/site-packages/pymongo/topology.py", line 238, in _select_servers_loop
    raise ServerSelectionTimeoutError(
pymongo.errors.ServerSelectionTimeoutError: No replica set members match selector "Primary()", Timeout: 30s, Top
ology Description: <TopologyDescription id: 64810072e17ccf7cab387a40, topology_type: ReplicaSetNoPrimary, server
s: [<ServerDescription ('mongo-node1', 27017) server_type: RSSecondary, rtt: 0.0009013634109403 94>, <ServerDescr
iption ('mongo-node2', 27017) server_type: Unknown, rtt: None, error=AutoReconnect('mongo-node2:27017: [Errno -3
] Temporary failure in name resolution')>, <ServerDescription ('mongo-node3', 27017) server_type: Unknown, rtt:
None, error=AutoReconnect('mongo-node3:27017: [Errno -3] Temporary failure in name resolution')>]>
```

```
During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/app/mongodb_replication.py", line 103, in <module>
    select_with_read_concern(args.read_concern)
  File "/app/mongodb_replication.py", line 58, in select_with_read_concern
    for x in collection.find({}, {"_id": 0, "name": 1, "address": 1, "country": 1}).limit(10):
  File "/usr/local/lib/python3.10/site-packages/pymongo/cursor.py", line 1248, in next
    if len(self.__data) or self._refresh():
  File "/usr/local/lib/python3.10/site-packages/pymongo/cursor.py", line 1165, in _refresh
    self.__send_message(q)
  File "/usr/local/lib/python3.10/site-packages/pymongo/cursor.py", line 1052, in __send_message
    response = client._run_operation(
  File "/usr/local/lib/python3.10/site-packages/pymongo/_csot.py", line 105, in csot_wrapper
    return func(self, *args, **kwargs)
  File "/usr/local/lib/python3.10/site-packages/pymongo/mongo_client.py", line 1330, in _run_operation
    return self._retryable_read(
  File "/usr/local/lib/python3.10/site-packages/pymongo/_csot.py", line 105, in csot_wrapper
    return func(self, *args, **kwargs)
  File "/usr/local/lib/python3.10/site-packages/pymongo/mongo_client.py", line 1455, in _retryable_read
    raise last_error
  File "/usr/local/lib/python3.10/site-packages/pymongo/mongo_client.py", line 1448, in _retryable_read
    return func(session, server, sock_info, read_pref)
  File "/usr/local/lib/python3.10/site-packages/pymongo/mongo_client.py", line 1326, in _cmd
    return server.run_operation(
  File "/usr/local/lib/python3.10/site-packages/pymongo/server.py", line 134, in run_operation
    _check_command_response(first, sock_info.max_wire_version)
  File "/usr/local/lib/python3.10/site-packages/pymongo/helpers.py", line 168, in _check_command_response
    raise NotPrimaryError(errmsg, response)
pymongo.errors.NotPrimaryError: operation was interrupted, full error: {'topologyVersion': {'processId': ObjectI
d('6480ffcdfbdbe972160adb01'), 'counter': 7}, 'ok': 0.0, 'errmsg': 'operation was interrupted', 'code': 11602, '
codeName': 'InterruptedDueToReplStateChange', '$clusterTime': {'clusterTime': Timestamp(1686175858, 1), 'signatu
re': {'hash': b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00', 'keyId': 0}},
 'operationTime': Timestamp(1686175858, 1)}
```

- Disable *primary* and turn on two *secondary* nodes. Wait until they elect a new *primary*.

Disable the *primary*

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Data
bases$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
                    NAMES
ee0629ee86b8   mongo     "docker-entrypoint.s…"   15 minutes ago   Up 15 minutes   0.0.0.0:27017->27017/tcp, :::
27017->27017/tcp   mongo-node1
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Data
bases$ docker stop ee0629ee86b8
ee0629ee86b8
```

Enable two *secondary* nodes

```
(base) denys_herasymuk@EPUALVIW07D6:~$ docker start mongo-node2
mongo-node2
```

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/I
bases$ docker start mongo-node3
mongo-node3
```

The current state of the nodes is the following:

```
members: [
  {
    _id: 0,
    name: 'mongo-node1:27017',
    health: 0,
    state: 8,
    stateStr: '(not reachable/healthy)',
    uptime: 0,
    optime: { ts: Timestamp({ t: 0, i: 0 }), t: Long("-1") },
```

```
  {
    _id: 1,
    name: 'mongo-node2:27017',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 39,
    optime: { ts: Timestamp({ t: 1686177128, i: 1 }), t: Long("2") },
```

```
  {
    _id: 2,
    name: 'mongo-node3:27017',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 32,
    optime: { ts: Timestamp({ t: 1686177128, i: 1 }), t: Long("2") },
```

- Enable the previous *primary* and check the records that were written on it before

Enable the previous *primary.*

```
(base) denys_herasymuk@EPUALVIW07D6:~$ docker start mongo-node1
mongo-node1
```

Now it is a *secondary.*

```
members: [
  {
    _id: 0,
    name: 'mongo-node1:27017',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 3,
    optime: { ts: Timestamp({ t: 1686177358, i: 1 }), t: Long("2") },
```

Now we can see that the records written only on *primary* are gone. The reason is that the records were not replicated to other replicas, and when the previous *primary* connected to the replica set, it became a secondary. Thus, it could not replicate these write operations to other replicas, but it replicated the state of the current *primary* that has no those records.

```
dd_hw6 [@docker_localhost]   console_1 [@docker_localhost]

1  db.dd_hw6.countDocuments();

Output   Result 2
1 row
{} result
1   0
```

8) Simulate eventual consistency by setting the replication delay for the replica

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Data
bases/HW6_MongoDB_Replication$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
                 NAMES
071da34370cc   mongo     "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:27037->27017/tcp, :::27
037->27017/tcp   mongo-node3
35357a561b5d   mongo     "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:27027->27017/tcp, :::27
027->27017/tcp   mongo-node2
b8a7f02c28bf   mongo     "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:27017->27017/tcp, :::27
017->27017/tcp   mongo-node1
```

Set a replication delay time to 90 seconds.

```
rs0 [direct: primary] test> cfg.members[1].priority = 0
0
rs0 [direct: primary] test> cfg.members[1].hidden = true
true
rs0 [direct: primary] test> cfg.members[1].secondaryDelaySecs = 90
90
rs0 [direct: primary] test> rs.reconfig(cfg)
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1686178142, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1686178142, i: 1 })
}
```

9) Disable one secondary without the replication delay. Write several records. Read these records with readConcern: {level: "linearizable"} . There should be a delay until the records are replicated to most nodes.

Stop node 3 that has no replication delay.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Data
bases/HW6_MongoDB_Replication$ docker stop mongo-node3
mongo-node3
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Data
bases/HW6_MongoDB_Replication$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
                 NAMES
35357a561b5d   mongo     "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:27027->27017/tcp, :::27
027->27017/tcp   mongo-node2
b8a7f02c28bf   mongo     "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:27017->27017/tcp, :::27
017->27017/tcp   mongo-node1
```

Read with the “linearizable” level.

```
(base) denys_herasymuk@EPUALVIW07D6:~/UCU/UCU_DE_Program_2022_2023/Distributed_Databases/UCU_DE_Distributed_Data
bases/HW6_MongoDB_Replication$ docker run --net my-mongo-cluster --rm dd_hw6 --read_concern linearizable
Connected to the DB
Top 10 records
{'name': 'Jared Osborne', 'address': '452 Pacheco Port Suite 762\nWest Joannburgh, VA 58515', 'country': 'Norway
'}
{'name': 'April Perez', 'address': '51743 Angela Turnpike\nSouth John, MS 52651', 'country': 'Bahamas'}
{'name': 'Nichole Wright', 'address': '694 Robert Spring\nSouth Christopherburgh, MT 08914', 'country': 'Sri Lan
ka'}
{'name': 'Patrick Farrell', 'address': '079 Cohen Meadows\nMichaeltown, WI 28157', 'country': 'Grenada'}
{'name': 'George Shannon', 'address': '2822 Tyler Plains\nWest Suzanneshire, GU 09885', 'country': 'Malawi'}
{'name': 'Andre Cortez', 'address': '27276 Diane Mountains\nValerieview, GA 47600', 'country': 'Slovakia (Slovak
 Republic)'}
{'name': 'Jeffrey Wilson', 'address': '88357 Ashley Branch Suite 739\nSouth Ericmouth, MD 56936', 'country': 'Eg
ypt'}
{'name': 'Angela Friedman', 'address': '7749 Hart Ranch Apt. 845\nWest Brandon, MT 64873', 'country': 'Angola'}
{'name': 'Caitlyn Yang', 'address': '56810 Burns Pine Suite 314\nCharleschester, CT 50230', 'country': 'Pitcairn
 Islands'}
{'name': 'Terry Silva', 'address': '41521 Robert Port\nLake Brenda, KS 14501', 'country': 'Barbados'}
Everything is written. Execution time: 90.70086431503296.
```

As we can see, the execution time is greater than 90 seconds, that is, actually, the replication delay time.