MongoDB 4.x

Eduardo Legatti
SYDLE

Agenda

- Architecture
- Replica Sets Configuration
- Essential Commands
- Backup / Restore
- Graphical Interface Clients
- Sharding

MongoDB Architecture

MongoDB - Architecture

History of MongoDB

- MongoDB (from hu**mongo**us) is a cross-platform document-oriented database
- **2007** - 10gen began developing MongoDB as a component of PaaS.
- **2009** - the company shifted to an open source development model, offering commercial support & other services.
- **2013** - 10gen changes its name to MongoDB Inc.

MongoDB - Architecture

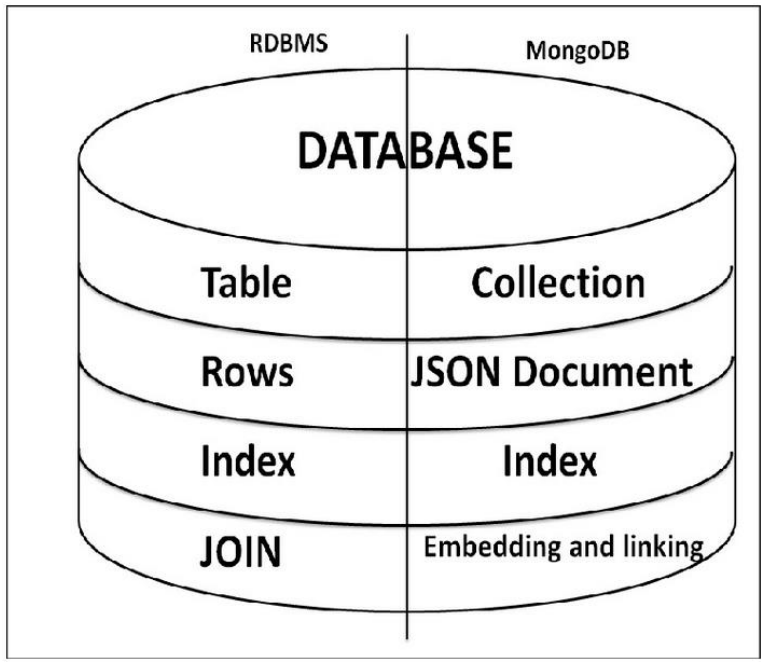
MySQL	MongoDB
Table	Collection
Row	Document
Column	Field
Joins	Embedded documents, linkin

MySQL	MongoDB

`INSERT INTO users (user_id, age, status)`
`VALUES ('bcd001', 45, 'A')`
	`db.users.insert({`
	`  user_id: 'bcd001',`
	`  age: 45,`
	`  status: 'A'`
	`})`

`SELECT * FROM users`	`db.users.find()`

`UPDATE users SET status = 'C'`
`WHERE age > 25`
	`db.users.update(`
	`  { age: { $gt: 25 } },`
	`  { $set: { status: 'C' } },`
	`  { multi: true }`
	`)`



MongoDB - Architecture

MMAPv1

- based on memory-mapped file
- MMAPv0 / v2.2 -> global lock
- MMAPv0 / v2.6 -> database level lock
- MMAPv1 / v3.0 -> collection level lock
- default storage engine
  - changed in v3.2 : MMAP is no longer the default

Deprecate MMAPv1

Starting in version 4.0, MongoDB deprecates the MMAPv1 storage Engine and will remove MMAPv1 in a future release.

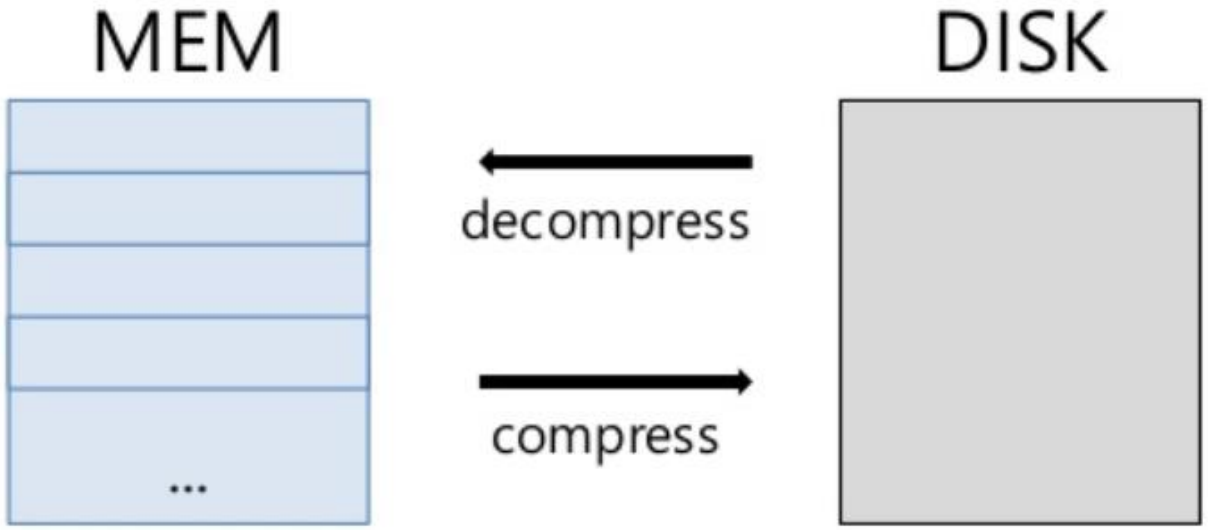
Removed MMAPv1 Storage Engine

MongoDB 4.2 removes the deprecated MMAPv1 storage engine.

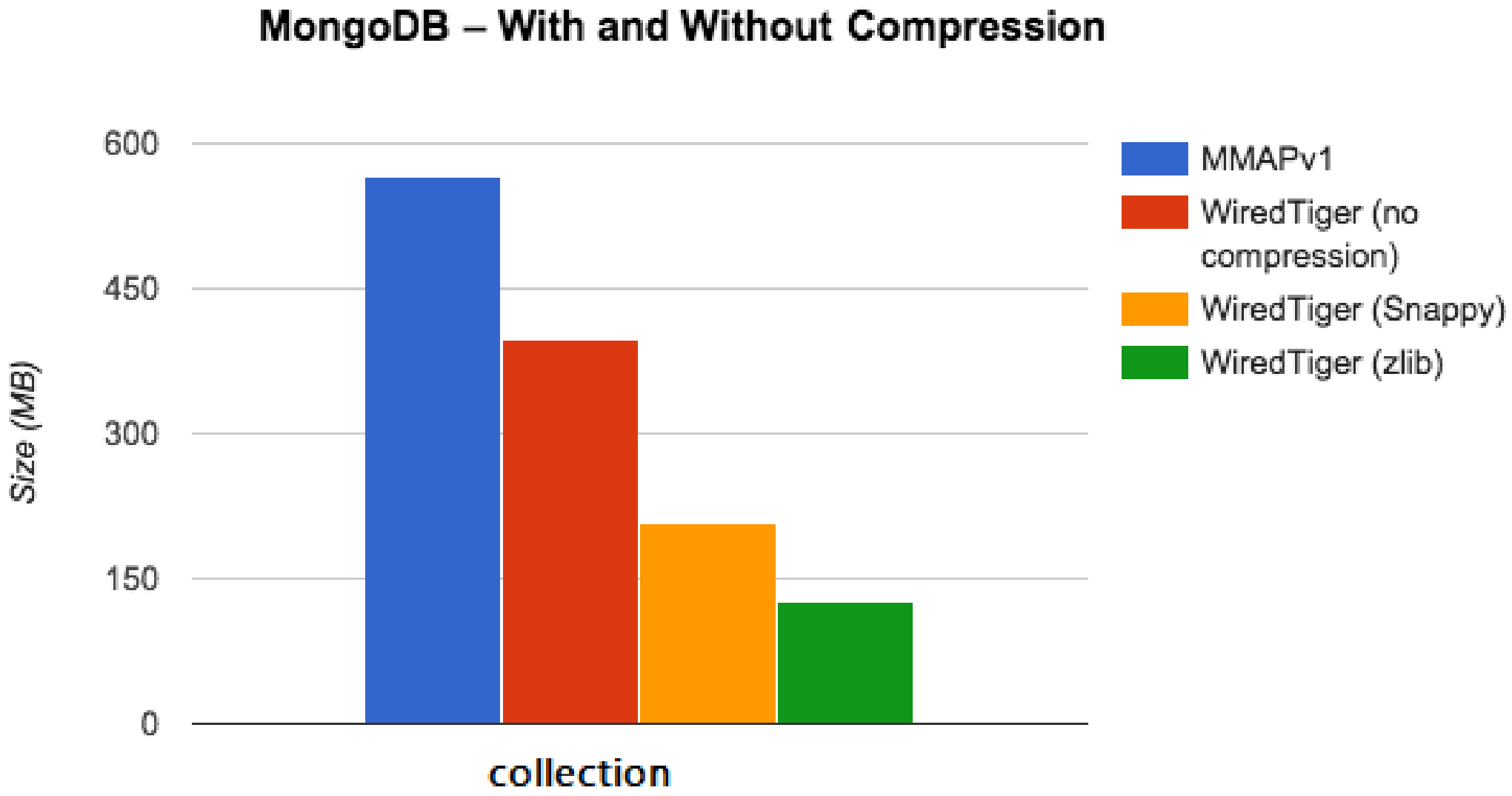
WiredTiger

. Document level concurrency

. Compression
  – all collections, indexes
  – snappy, zlib..



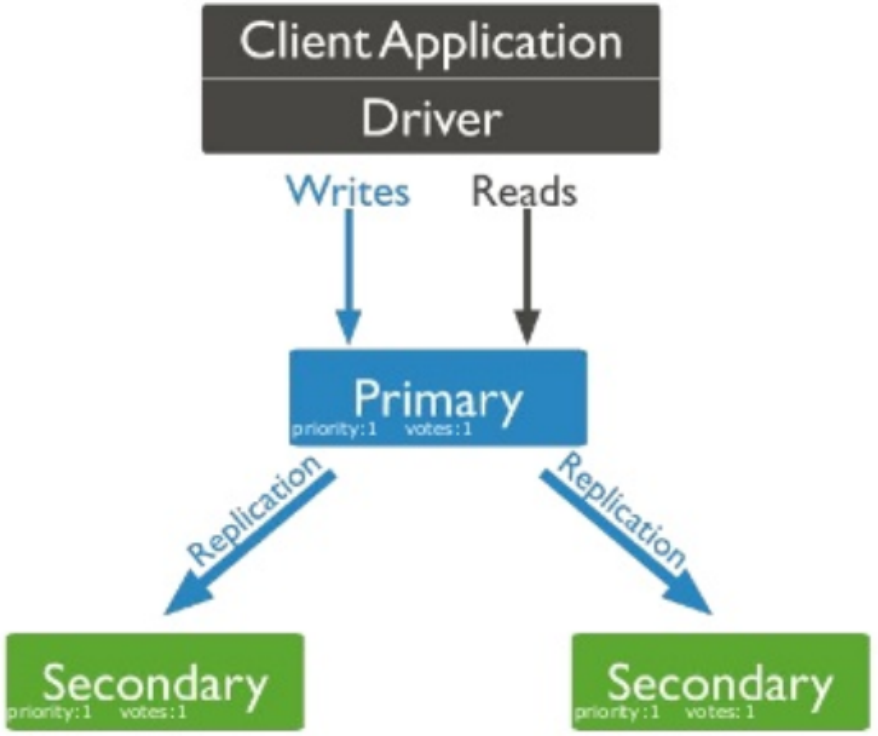
MongoDB - Architecture



Replica Set

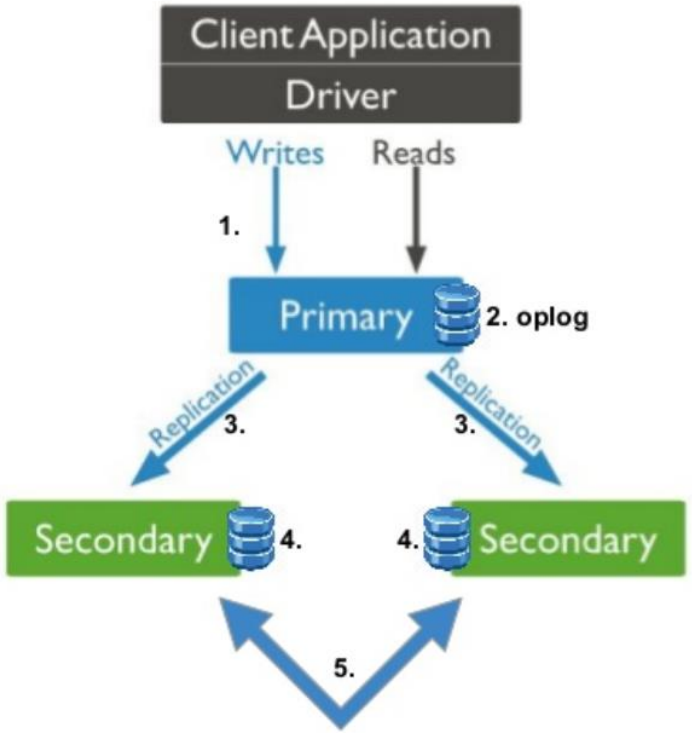
- Group of mongod processes that maintain the same data set
- Redundancy and high availability
- Increased read capacity (scaling reads)
- Automatic failover

# Members	# Nodes Required to Elect New Primary	Fault Tolerance
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3



Replication Concept

1. Write operations go to the Primary node
2. All changes are recorded into operations log
3. Asynchronous replication to Secondary
4. Secondaries copy the Primary oplog
5. Secondary can use sync source Secondary*



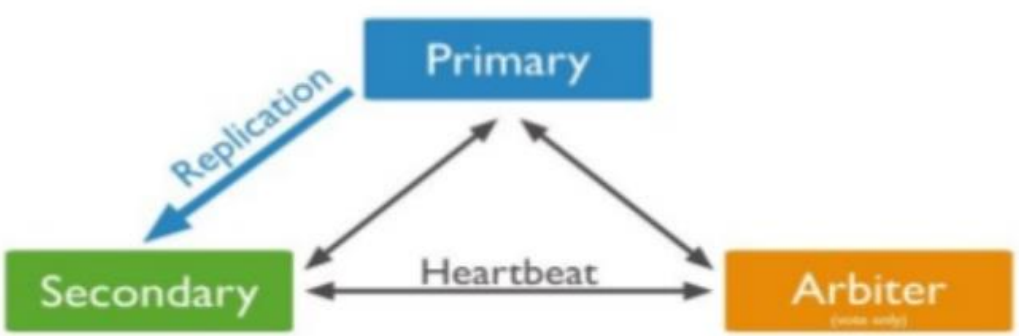
Replica Set Oplog

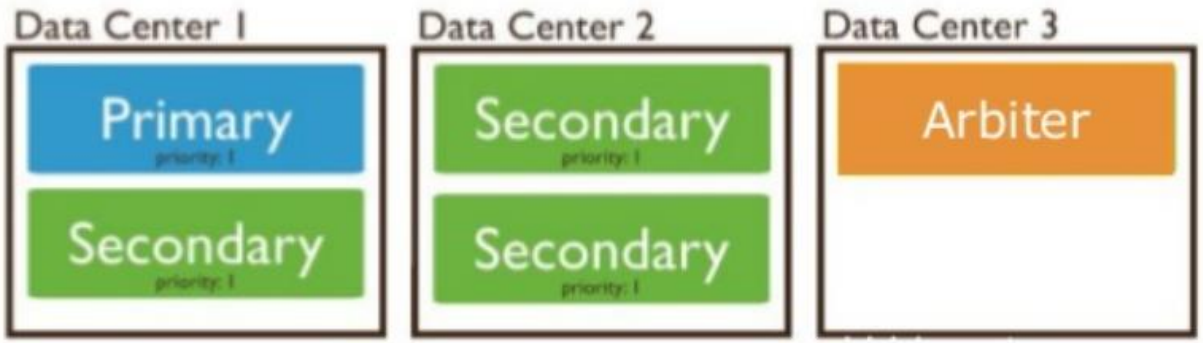
- Special capped collection that keeps a rolling record of all operations that modify the data stored in the databases
- Idempotent
- Default oplog size

Arbiter Node

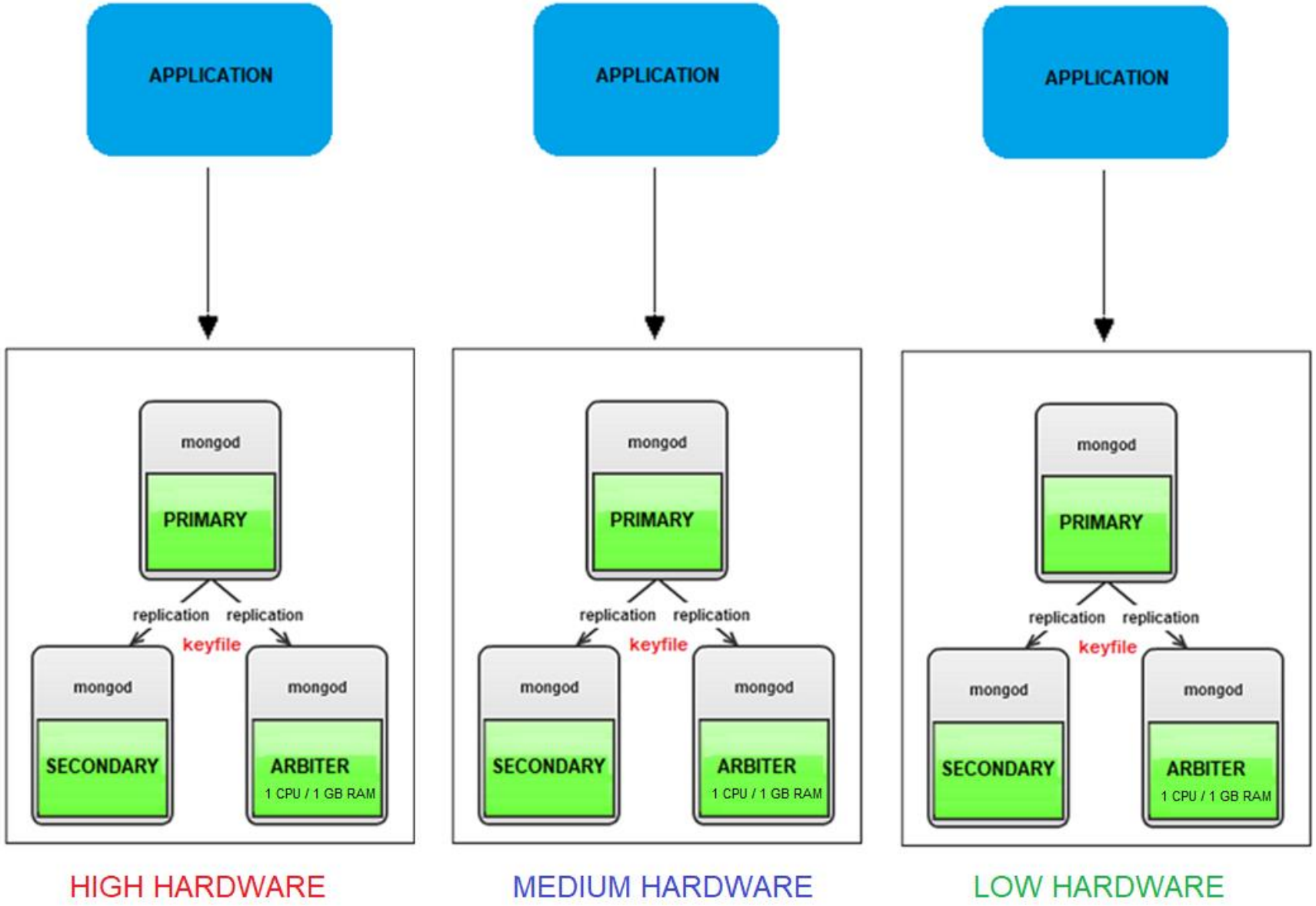
- Does not hold copy of data
- Votes in elections

```
rs.addArb("arbiter.net:27017")
```





MongoDB - Architecture

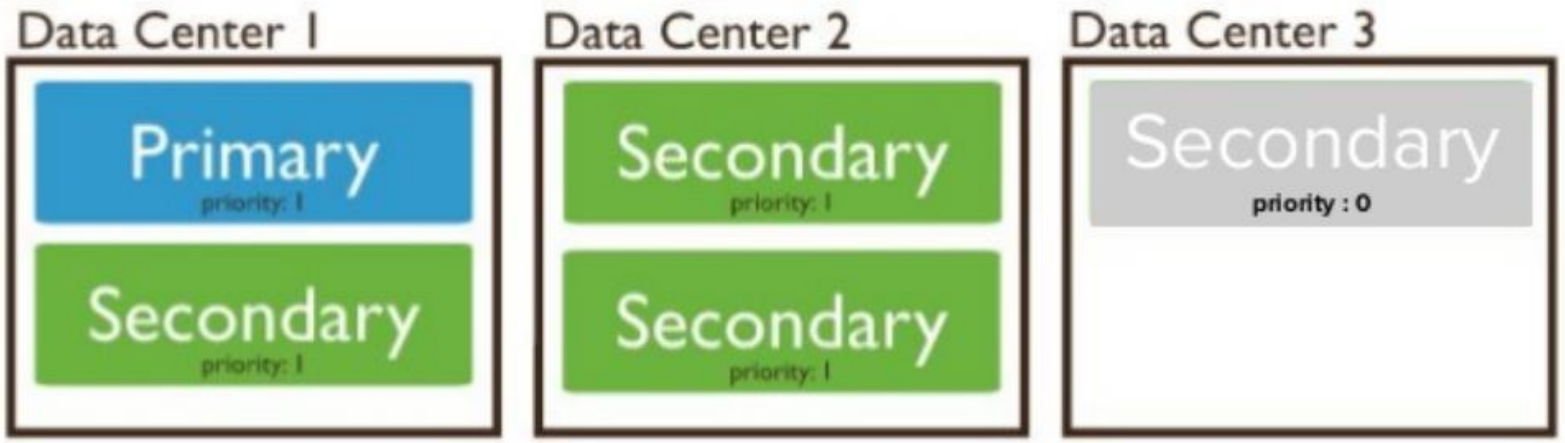


Priority 0 Node

Priority - floating point (i.e. decimal) number between 0 and 1000

- Cannot become primary, cannot trigger election
- Visible to application (accepts reads/writes)
- Votes in elections

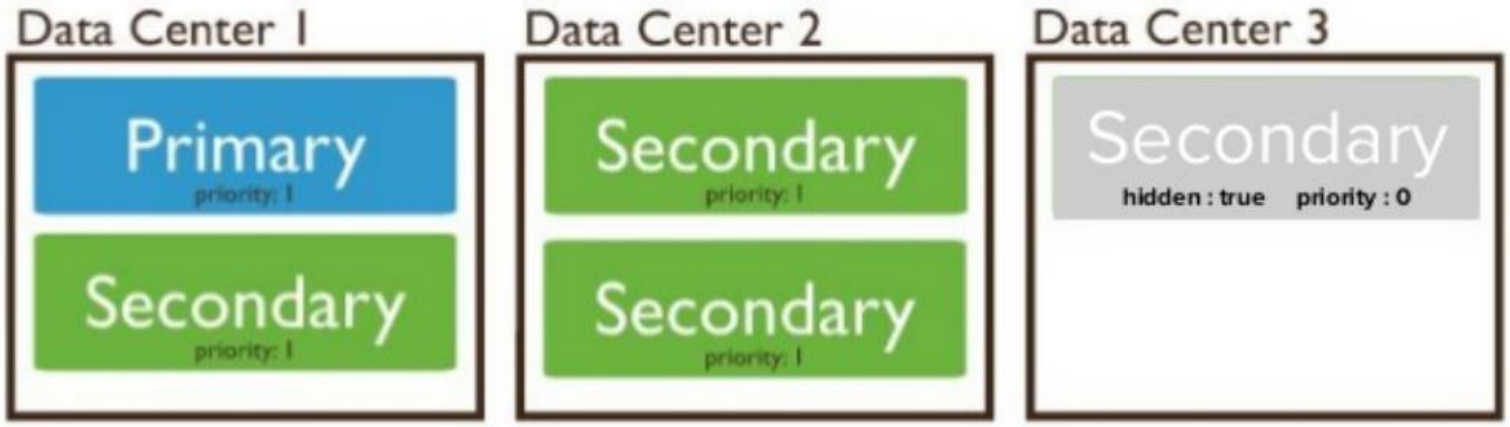
```
rs.add( { host: "mongodb5.net:27017", priority: 0 } )
```



Hidden Node

- Not visible to application
- Never becomes primary, but can vote in elections
- Use cases (reporting, backups)

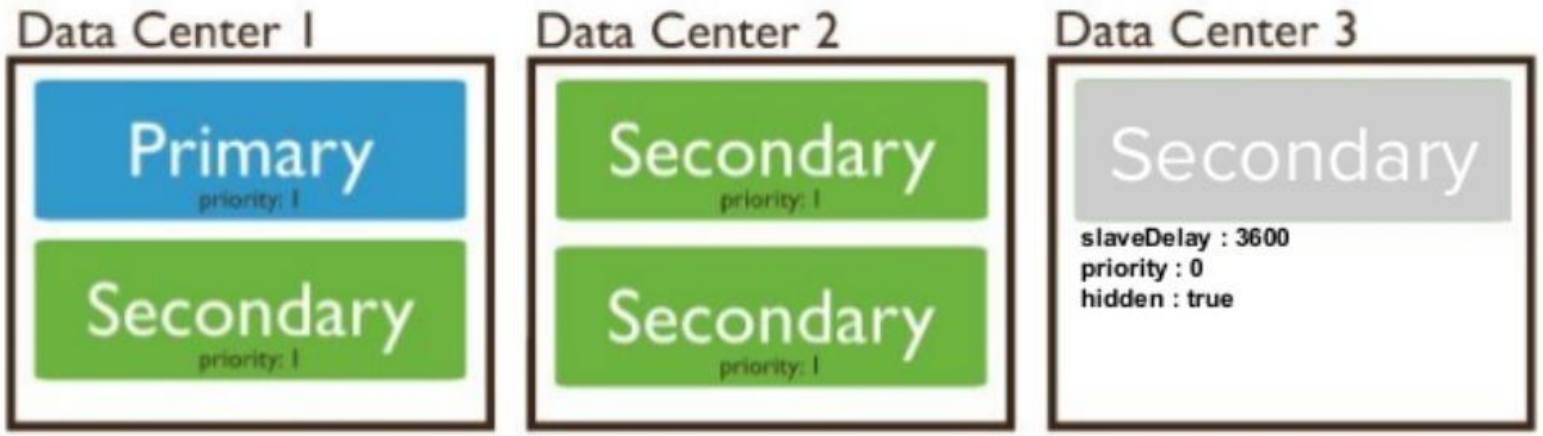
```
rs.add( { host: "mongodb5.net:27017", priority: 0, hidden: true } )
```



Delayed Node

- Must be priority 0 member
- Should be hidden member (not mandatory)
- Mainly used for backups (historical snapshot of data)
- Recovery in case of human error

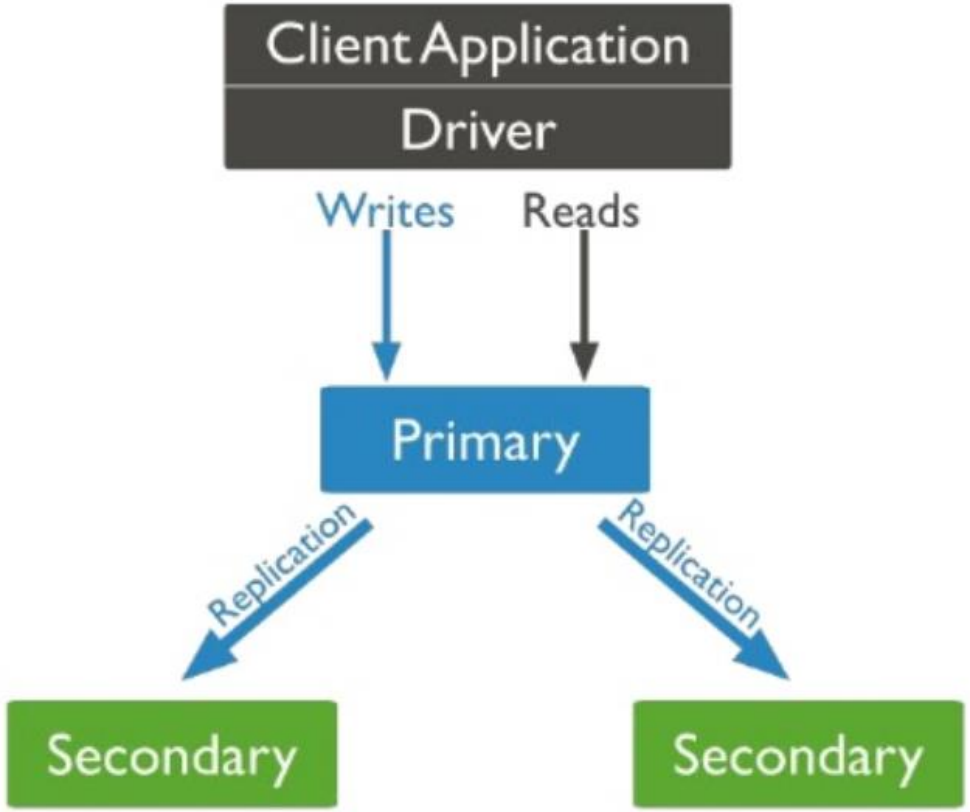
```
rs.add( { host: "mongodb5.net:27017", priority: 0, hidden: true, slaveDelay=3600 } )
```



Replica maintenance

- Change the necessary settings for the desired nodes

```
> cfg.members[0].priority = 0.5
> cfg.members[1].priority = 2
> cfg.members[2].priority = 0
> cfg.members[2].hidden = true
> cfg.members[2].slaveDelay = 3600
> cfg.settings.electionTimeoutMillis = 12000
```

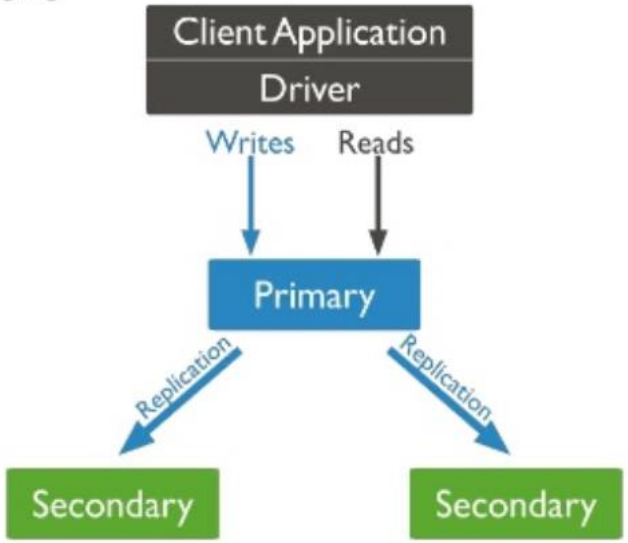


MongoDB - Architecture

Read preference

How read operations are routed to replica set members

1. primary (by default)
2. primaryPreferred
3. secondary
4. secondaryPreferred
5. nearest (least network latency)



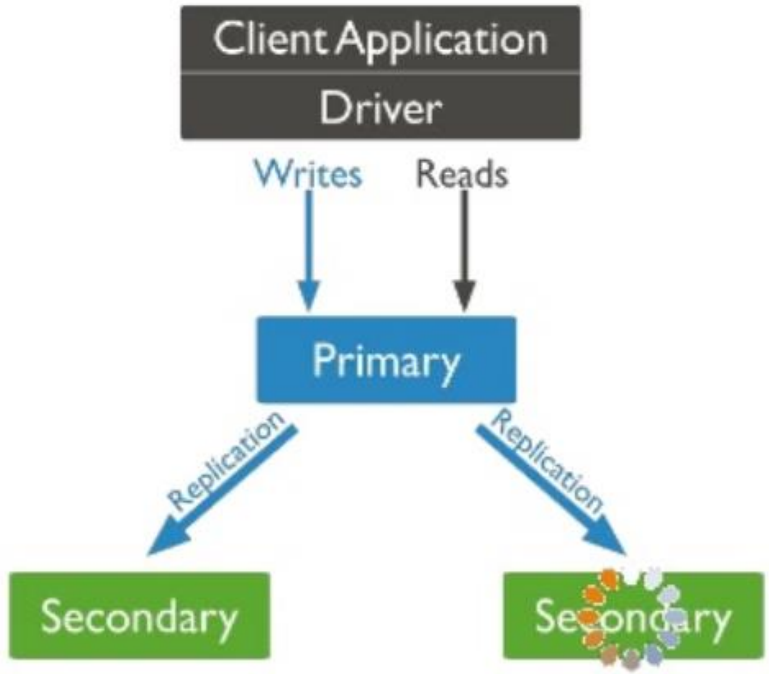
- **primary** - Default mode. All operations read from the current replica set primary.
- **primaryPreferred** - In most situations, operations read from the primary but if it is unavailable, operations read from secondary members.
- **secondary** - All operations read from the secondary members of the replica set.
- **secondaryPreferred** - In most situations, operations read from secondary members but if no secondary members are available, operations read from the primary.
- **nearest** - Operations read from member of the replica set with the least network latency, irrespective of the member's type.

Resync Replica member

- Stop mongod process on the stale node
- Remove everything from the `--dbPath` directory
- Start mongod process
- Wait initial sync to finish automatically

Alternative

- Copy the data files from a Secondary that is locked for writes `--db.fsyncLock()`
- Sync the stale node



MongoDB - Architecture

Replica Set Member States

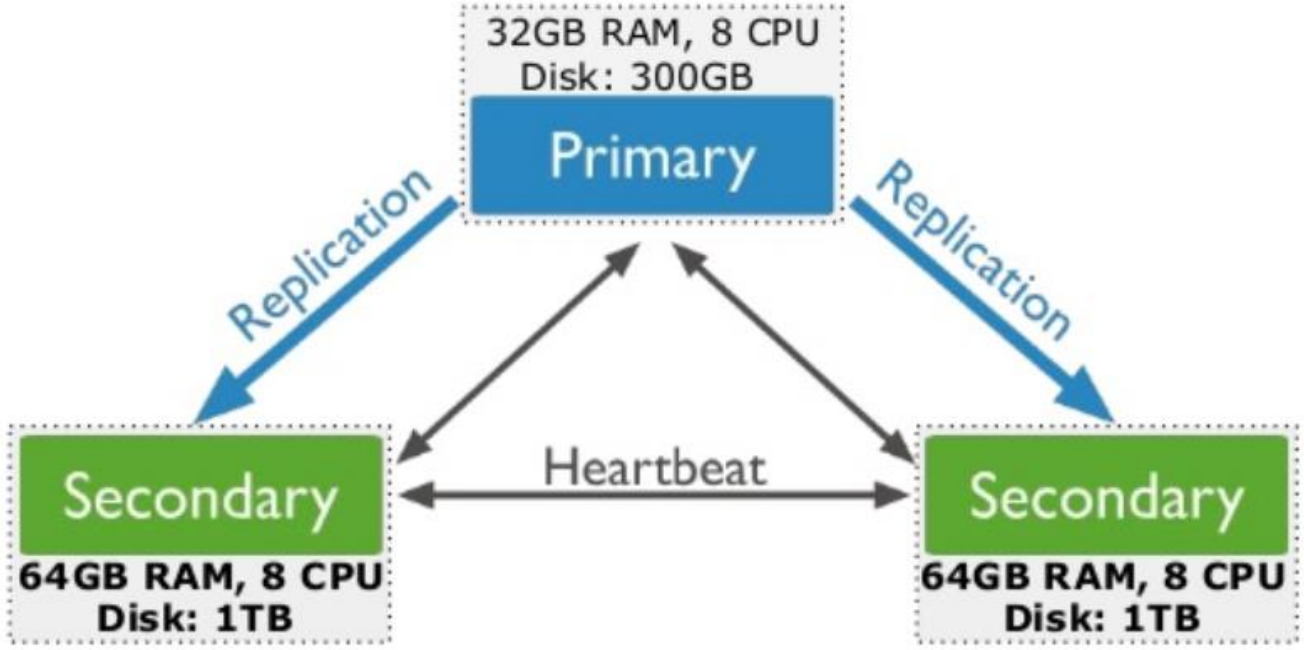
Number	Name	State Description
0	STARTUP	Not yet an active member of any set. All members start up in this state. The mongod parses the replica set configuration document while in STARTUP.
1	PRIMARY	The member in state primary is the only member that can accept write operations. Eligible to vote.
2	SECONDARY	A member in state secondary is replicating the data store. Eligible to vote.
3	RECOVERING	Members either perform startup self-checks, or transition from completing a rollback or resync. Eligible to vote.
5	STARTUP2	The member has joined the set and is running an initial sync. Eligible to vote.
6	UNKNOWN	The member's state, as seen from another member of the set, is not yet known.
7	ARBITER	Arbiters do not replicate data and exist solely to participate in elections. Eligible to vote.
8	DOWN	The member, as seen from another member of the set, is unreachable.
9	ROLLBACK	This member is actively performing a rollback. Eligible to vote. Data is not available for reads from this member.
10	REMOVED	This member was once in a replica set but was subsequently removed.

Hardware/OS upgrades

- Step down the Primary

```
> rs.stepDown(60)
```

- Confirm new Primary has been elected
- Do the same changes on the former Primary



MongoDB Replica Sets Configuration

MongoDB – Replica Sets Configuration

https://www.mongodb.com/try/download/community

MongoDB Community Server

MongoDB offers both an Enterprise and Community version of its powerful distributed document database. MongoDB Enterprise is available as part of the MongoDB Enterprise Advanced subscription, which features the most comprehensive support and the best SLA when you run MongoDB on your own infrastructure. MongoDB Enterprise Advanced also gives you comprehensive operational tooling, advanced analytics and data visualization, platform integrations and certification, along with on-demand training for your teams.

- **In-memory Storage Engine**
  Deliver high throughput and predictable low latency
- **Advanced Security**
  Secure your data with LDAP and Kerberos access controls and comprehensive auditing
- **Encrypted Storage Engine**
  Encrypt your data at rest

Available Downloads

Version: 4.2.8

Platform: RedHat / CentOS 7.0

Package: tgz

Download | Copy Link

Current releases & packages
Development releases
Archived releases
Changelog
Release Notes

MongoDB – Replica Sets Configuration

```
$ openssl rand -base64 756 > /mongodb/cluster/keyfile.conf
$ chown mongodb.mongodb /mongodb/cluster/keyfile.conf
$ chmod 400 /mongodb/cluster/keyfile.conf
```

```
/mongodb
|-- cluster
|   |-- keyfile.conf
|   |-- mongodb01
|   |   |-- data
|   |   `-- log
|   |-- mongodb02
|   |   |-- data
|   |   `-- log
|   `-- mongodb03
|       |-- data
|       `-- log
`-- mongodb-linux-x86_64-rhel70-4.2.8
    |-- bin
    |   |-- mongo
    |   |-- mongod
    |   |-- mongodump
    |   |-- mongorestore
    |   |-- mongostat
    |   `-- mongotop
.
.
```

MongoDB – Replica Sets Configuration

```
$ mongod --keyFile /mongodb/cluster/keyfile.conf --bind_ip_all --enableMajorityReadConcern false \
> --fork --oplogSize 1024 --journal --replSet RepSet --port 27017 \
> --dbpath /mongodb/cluster/mongodb01/data/ \
> --logpath /mongodb/cluster/mongodb01/log/mongo.log
about to fork child process, waiting until server is ready for connections.
forked process: 6296
child process started successfully, parent exiting
```

```
$ mongod --keyFile /mongodb/cluster/keyfile.conf --bind_ip_all --enableMajorityReadConcern false \
> --fork --oplogSize 1024 --journal --replSet RepSet --port 27018 \
> --dbpath /mongodb/cluster/mongodb02/data/ \
> --logpath /mongodb/cluster/mongodb02/log/mongo.log
about to fork child process, waiting until server is ready for connections.
forked process: 6408
child process started successfully, parent exiting
```

```
$ mongod --keyFile /mongodb/cluster/keyfile.conf --bind_ip_all --enableMajorityReadConcern false \
> --fork --oplogSize 1024 --journal --replSet RepSet --port 27019 \
> --dbpath /mongodb/cluster/mongodb03/data/ \
> --logpath /mongodb/cluster/mongodb03/log/mongo.log
about to fork child process, waiting until server is ready for connections.
forked process: 6464
child process started successfully, parent exiting
```

MongoDB – Replica Sets Configuration

```
$ ps aux | grep mongo
mongodb   6296  1.7  1.4 1570556 84556 ? SLl  15:29   0:01 mongod --keyFile /mongodb/cluster/keyfile.conf
mongodb   6408  2.8  1.3 1570560 81976 ? SLl  15:29   0:01 mongod --keyFile /mongodb/cluster/keyfile.conf
mongodb   6464  4.1  1.4 1576528 86920 ? SLl  15:30   0:01 mongod --keyFile /mongodb/cluster/keyfile.conf
```

```
$ netstat -nalp | grep 270
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:27017           0.0.0.0:*               LISTEN      6296/mongod
tcp        0      0 0.0.0.0:27018           0.0.0.0:*               LISTEN      6408/mongod
tcp        0      0 0.0.0.0:27019           0.0.0.0:*               LISTEN      6464/mongod
unix  2      [ ACC ]     STREAM     LISTENING     30677    6296/mongod         /tmp/mongodb-27017.sock
unix  2      [ ACC ]     STREAM     LISTENING     33479    6408/mongod         /tmp/mongodb-27018.sock
unix  2      [ ACC ]     STREAM     LISTENING     33507    6464/mongod         /tmp/mongodb-27019.sock
```

MongoDB – Replica Sets Configuration

```
$ mongo
MongoDB shell version v4.2.8
Implicit session: session { "id" : UUID("c32eb2ad-f2d3-46f1-9a34-978517614302") }

> config = {_id: 'RepSet', members: [
...                 {_id: 0, host: 'mongodb01:27017'},
...                 {_id: 1, host: 'mongodb02:27018'},
...                 {_id: 2, host: 'mongodb03:27019', arbiterOnly: true}]
...         }
{
        "_id" : "RepSet",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "mongodb01:27017"
                },
                {
                        "_id" : 1,
                        "host" : "mongodb02:27018"
                },
                {
                        "_id" : 2,
                        "host" : "mongodb03:27019",
                        "arbiterOnly" : true
                }
        ]
}

> rs.initiate(config);
{ "ok" : 1 }
```

MongoDB – Replica Sets Configuration

```
RepSet:PRIMARY> use admin
switched to db admin
RepSet:PRIMARY> db.createUser(
...     {
...         user: "admin",
...         pwd: "admin",
...         roles:["root"]
...     });
Successfully added user: { "user" : "admin", "roles" : [ "root" ] }
```

MongoDB – Replica Sets Configuration

```
$ mongod --auth --bind_ip_all --fork --oplogSize 1024 --journal --port 27017 \
> --dbpath /mongodb/cluster/mongodb01/data/ \
> --logpath /mongodb/cluster/mongodb01/log/mongo.log
about to fork child process, waiting until server is ready for connections.
forked process: 6074
child process started successfully, parent exiting

$ mongo --port 27017 --username "admin" --password admin --authenticationDatabase "admin"
MongoDB shell version v4.2.8
connecting to:
mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("bcd408a4-bd5a-4d19-81ad-086c098ffd57") }
MongoDB server version: 4.2.8
>
```

```
# mongod --auth --keyFile /mongodb/cluster/keyfile.conf --bind_ip_all
--enableMajorityReadConcern false --fork
--oplogSize 1024 --journal
--replSet RepSet --port 27017
--dbpath /mongodb/cluster/mongodb01/data/
--logpath /mongodb/cluster/mongodb01/log/mongo.log
```

MongoDB
Essential Commands

MongoDB – Essential Commands

https://docs.mongodb.com/manual/reference/sql-comparison/

SQL Schema Statements

```
CREATE TABLE people (
    id MEDIUMINT NOT NULL
        AUTO_INCREMENT,
    user_id Varchar(30),
    age Number,
    status char(1),
    PRIMARY KEY (id)
)
```

MongoDB Schema Statements

Implicitly created on first `insertOne()` or `insertMany()` operation. The primary key `_id` is automatically added if `_id` field is not specified.

```
db.people.insertOne( {
    user_id: "abc123",
    age: 55,
    status: "A"
 } )
```

However, you can also explicitly create a collection:

```
db.createCollection("people")
```

MongoDB – Essential Commands

```
ALTER TABLE people
ADD join_date DATETIME
```

Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.

However, at the document level, `updateMany()` operations can add fields to existing documents using the `$set` operator.

```
db.people.updateMany(
    { },
    { $set: { join_date: new Date() } }
)
```

MongoDB – Essential Commands

```
ALTER TABLE people
DROP COLUMN join_date
```

Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.

However, at the document level, `updateMany()` operations can remove fields from documents using the `$unset` operator.

```
db.people.updateMany(
    { },
    { $unset: { "join_date": "" } }
)
```

MongoDB – Essential Commands

`CREATE INDEX idx_user_id_asc` `ON people(user_id)`	`db.people.createIndex( { user_id: 1 } )`

`CREATE INDEX`
`idx_user_id_asc_age_desc`
`ON people(user_id, age DESC)`
	`db.people.createIndex( { user_id: 1, age: -1 } )`

`DROP TABLE people`	`db.people.drop()`

MongoDB – Essential Commands

SQL INSERT Statements

```
INSERT INTO people(user_id,
                   age,
                   status)
VALUES ("bcd001",
        45,
        "A")
```

MongoDB insertOne() Statements

```
db.people.insertOne(
   { user_id: "bcd001", age: 45, status: "A" }
)
```

MongoDB – Essential Commands

SQL SELECT Statements	MongoDB find() Statements
`SELECT *` `FROM people`	`db.people.find()`

`SELECT id,`
`  user_id,`
`  status`
`FROM people`
	`db.people.find(`
	`  { },`
	`  { user_id: 1, status: 1 }`
	`)`

`SELECT user_id, status`
`FROM people`
	`db.people.find(`
	`  { },`
	`  { user_id: 1, status: 1, _id: 0 }`
	`)`

`SELECT *`
`FROM people`
`WHERE status = "A"`
	`db.people.find(`
	`  { status: "A" }`
	`)`

MongoDB – Essential Commands

```sql
SELECT user_id, status
FROM people
WHERE status = "A"
```

```
db.people.find(
    { status: "A" },
    { user_id: 1, status: 1, _id: 0 }
)
```

```sql
SELECT *
FROM people
WHERE status != "A"
```

```
db.people.find(
    { status: { $ne: "A" } }
)
```

```sql
SELECT *
FROM people
WHERE status = "A"
AND age = 50
```

```
db.people.find(
    { status: "A",
      age: 50 }
)
```

```sql
SELECT *
FROM people
WHERE status = "A"
OR age = 50
```

```
db.people.find(
    { $or: [ { status: "A" } , { age: 50 } ] }
)
```

MongoDB – Essential Commands

`SELECT *`
`FROM people`
`WHERE age > 25`
	`db.people.find(`
	`  { age: { $gt: 25 } }`
	`)`

`SELECT *`
`FROM people`
`WHERE age < 25`
	`db.people.find(`
	`  { age: { $lt: 25 } }`
	`)`

`SELECT *`
`FROM people`
`WHERE age > 25`
`AND age <= 50`
	`db.people.find(`
	`  { age: { $gt: 25, $lte: 50 } }`
	`)`

`SELECT *`
`FROM people`
`WHERE user_id like "%bc%"`
	`db.people.find( { user_id: /bc/ } )`
	
	-or-
	
	`db.people.find( { user_id: { $regex: /bc/ } } )`

MongoDB – Essential Commands

`SELECT * FROM people WHERE user_id like "bc%"`	`db.people.find( { user_id: /^bc/ } )`
	-or-
	`db.people.find( { user_id: { $regex: /^bc/ } } )`

`SELECT * FROM people WHERE status = "A" ORDER BY user_id ASC`	`db.people.find( { status: "A" } ).sort( { user_id: 1 } )`
`SELECT * FROM people WHERE status = "A" ORDER BY user_id DESC`	`db.people.find( { status: "A" } ).sort( { user_id: -1 } )`

`SELECT COUNT(*) FROM people`	`db.people.count()`
	or
	`db.people.find().count()`

MongoDB – Essential Commands

Update Records

The following table presents the various SQL statements related to updating existing records in tables and the corresponding MongoDB statements.

SQL Update Statements	MongoDB updateMany() Statements

`UPDATE people`
`SET status = "C"`
`WHERE age > 25`
	`db.people.updateMany(`
	`{ age: { $gt: 25 } },`
	`{ $set: { status: "C" } }`
	`)`

`UPDATE people`
`SET age = age + 3`
`WHERE status = "A"`
	`db.people.updateMany(`
	`{ status: "A" } ,`
	`{ $inc: { age: 3 } }`
	`)`

MongoDB – Essential Commands

Delete Records

The following table presents the various SQL statements related to deleting records from tables and the corresponding MongoDB statements.

SQL Delete Statements	MongoDB deleteMany() Statements
`DELETE FROM people` `WHERE status = "D"`	`db.people.deleteMany( { status: "D" } )`
`DELETE FROM people`	`db.people.deleteMany({})`

MongoDB – Essential Commands

```
$ mongo --port 27017 --username "admin" --password admin --authenticationDatabase "admin"

RepSet:PRIMARY> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB

RepSet:PRIMARY> use admin
switched to db admin

RepSet:PRIMARY> show collections
system.keys
system.users
system.version
```

MongoDB – Essential Commands

```
RepSet:PRIMARY> rs.conf()
{
        "_id" : "RepSet",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "mongodb01:27017",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1
                },
                {
                        "_id" : 1,
                        "host" : "mongodb02:27018",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1
                },
                {
                        "_id" : 2,
                        "host" : "mongodb03:27019",
                        "arbiterOnly" : true,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 0
                }
        ]
}}
```

MongoDB – Essential Commands

```
RepSet:PRIMARY> cfg = rs.conf();
RepSet:PRIMARY> cfg.members[1].priority = 0.5;
0.5

RepSet:PRIMARY> rs.reconfig(cfg);
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1595265020, 1),
                "signature" : {
                        "hash" : BinData(0,"jpzyk7E0wNxfE347bAiVC7idRHw="),
                        "keyId" : NumberLong("6851570656530661379")
                }
        },
        "operationTime" : Timestamp(1595265020, 1)
}
```

MongoDB – Essential Commands

```
RepSet:PRIMARY> rs.conf()
{
        "_id" : "RepSet",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "mongodb01:27017",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1
                },
                {
                        "_id" : 1,
                        "host" : "mongodb02:27018",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 0.5
                },
                {
                        "_id" : 2,
                        "host" : "mongodb03:27019",
                        "arbiterOnly" : true,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 0
                }
        ]
}}
```

MongoDB – Essential Commands

```
RepSet:PRIMARY> rs.status()
{
        "set" : "RepSet",
        "members" : [
                {
                        "_id" : 0,
                        "name" : "mongodb01:27017",
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "syncingTo" : ""
                },
                {
                        "_id" : 1,
                        "name" : "mongodb02:27018",
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "pingMs" : NumberLong(0),
                        "syncingTo" : "mongodb01:27017",
                        "syncSourceHost" : "mongodb01:27017"
                },
                {
                        "_id" : 2,
                        "name" : "mongodb03:27019",
                        "state" : 7,
                        "stateStr" : "ARBITER",
                        "syncingTo" : "",
                        "syncSourceHost" : ""
                }
        ],
        "ok" : 1}
}
```

MongoDB – Essential Commands

```
$ mongod --dbpath /mongodb/cluster/mongodb02/data -shutdown

RepSet:PRIMARY> rs.status()
{
        "set" : "RepSet",
        "members" : [
                {
                        "_id" : 0,
                        "name" : "mongodb01:27017",
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "syncingTo" : ""
                },
                {
                        "_id" : 1,
                        "name" : "mongodb02:27018",
                        "state" : 2,
                        "stateStr" : "(not reachable/healthy)",
                        "pingMs" : NumberLong(0),
                        "lastHeartbeatMessage" : "Error connecting to mongodb02:27018 :: caused by :: Connection refused",
                        "syncingTo" : ""
                },
                {
                        "_id" : 2,
                        "name" : "mongodb03:27019",
                        "state" : 7,
                        "stateStr" : "ARBITER",
                        "syncingTo" : ""
                }
        ],
        "ok" : 1
}
```

MongoDB – Essential Commands

```
$ mongod --dbpath /mongodb/cluster/mongodb03/data -shutdown

RepSet:SECONDARY> rs.status()
{
        "set" : "RepSet",
        "members" : [
                {
                        "_id" : 0,
                        "name" : "mongodb01:27017",
                        "state" : 1,
                        "stateStr" : "SECONDARY",
                        "infoMessage" : "could not find member to sync from",
                        "syncingTo" : ""
                },
                {
                        "_id" : 1,
                        "name" : "mongodb02:27018",
                        "state" : 2,
                        "stateStr" : "(not reachable/healthy)",
                        "pingMs" : NumberLong(0),
                        "lastHeartbeatMessage" : "Error connecting to mongodb02:27018 :: caused by :: Connection refused",
                },
                {
                        "_id" : 2,
                        "name" : "mongodb03:27019",
                        "state" : 7,
                        "stateStr" : "(not reachable/healthy)",
                        "lastHeartbeatMessage" : "Error connecting to mongodb02:27019 :: caused by :: Connection refused",
                        "syncingTo" : ""
                }
        ],
        "ok" : 1
}
```

MongoDB – Essential Commands

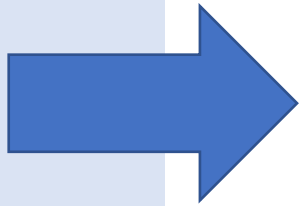
```
RepSet:PRIMARY> db.printReplicationInfo()
configured oplog size:   1024MB
log length start to end: 18547secs (5.15hrs)
oplog first event time:  Mon Jul 20 2020 11:33:14 GMT-0300 (-03)
oplog last event time:   Mon Jul 20 2020 16:42:21 GMT-0300 (-03)
now:                     Mon Jul 20 2020 16:42:22 GMT-0300 (-03)

RepSet:PRIMARY> db.printSlaveReplicationInfo()
source: mongodb02:27018
        syncedTo: Mon Jul 20 2020 16:42:21 GMT-0300 (-03)
        0 secs (0 hrs) behind the primary
```

MongoDB – Essential Commands

```
$ mongod --dbpath /mongodb/cluster/mongodb02/data --shutdown
2020-07-21T15:19:30.764-0300 I  CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0
specify --sslDisabledProtocols 'none'
2020-07-21T15:19:30.767-0300 W  ASIO     [main] No TransportLayer configured during NetworkInterface startup
killing process with pid: 4523

$ rm -rf /mongodb/cluster/mongodb02/data/*

$ mongod --keyFile /mongodb/cluster/keyfile.conf --bind_ip_all --enableMajorityReadConcern false \
> --fork --oplogSize 1024 --journal --replSet RepSet --port 27018 \
> --dbpath /mongodb/cluster/mongodb02/data/ \
> --logpath /mongodb/cluster/mongodb02/log/mongo.log
about to fork child process, waiting until server is ready for connections.
forked process: 25410
child process started successfully, parent exiting

$ mongo --port 27017 --username "admin" --password admin --authenticationDatabase "admin"
RepSet:PRIMARY> rs.status()
{
        "set" : "RepSet",
        },
        "members" : [
                {
                        "_id" : 1,
                        "name" : "mongodb02:27018",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "STARTUP2",
                        "uptime" : 10,
}
```

MongoDB – Essential Commands

```
RepSet:PRIMARY> print(Date())
Mon Jul 20 2020 16:51:30 GMT-0300 (-03)
RepSet:PRIMARY> rs.stepDown(300)
{
        "ok" : 1,
        "operationTime" : Timestamp(1595274690, 1)
}

RepSet:SECONDARY> rs.status()
{
        "set" : "RepSet",
        "members" : [
                {
                        "_id" : 0,
                        "name" : "mongodb01:27017",
                        "stateStr" : "SECONDARY",
                        "syncingTo" : "mongodb02:27018",
                        "syncSourceHost" : "mongodb02:27018"
                },
                {
                        "_id" : 1,
                        "name" : "mongodb02:27018",
                        "stateStr" : "PRIMARY"
                },
                {
                        "_id" : 2,
                        "name" : "mongodb03:27019",
                        "stateStr" : "ARBITER"
                }
        ],
        "ok" : 1
}
```



```
RepSet:SECONDARY> print(Date())
Mon Jul 20 2020 16:52:24 GMT-0300 (-03)
RepSet:SECONDARY> print(Date())
Mon Jul 20 2020 16:53:35 GMT-0300 (-03)
RepSet:SECONDARY> print(Date())
Mon Jul 20 2020 16:54:43 GMT-0300 (-03)
RepSet:SECONDARY> print(Date())
Mon Jul 20 2020 16:55:56 GMT-0300 (-03)
RepSet:SECONDARY> print(Date())
Mon Jul 20 2020 16:56:32 GMT-0300 (-03)
RepSet:PRIMARY>
```

MongoDB – Essential Commands

```
RepSet:PRIMARY> db.currentOp({"active" : true,"secs_running" : { "$gt" : 0 },"ns" : {$ne : "local.oplog.rs"}})
{
        "inprog" : [
                {
                        "desc" : "conn1371708",
                        "threadId" : "140655653025536",
                        "connectionId" : 1371708,
                        "client" : "172.31.80.121:51282",
                        "active" : true,
                        "opid" : 2036637847,
                        "secs_running" : 1,
                        "microsecs_running" : NumberLong(1082772),
                        "op" : "query",
                        "ns" : "bd01.product",
                        "query" : {
                                "$msg" : "query not recording (too large)"
                        },
                        "numYields" : 58,
                        "locks" : {
                                "Global" : "r",
                                "Database" : "r"
                        }
}
```

MongoDB – Essential Commands

```
$ mongotop --port 27017 --username admin --password admin --authenticationDatabase admin

2020-07-22T15:41:07.762-0300    connected to: mongodb://localhost:27017/

                    ns    total     read    write    2020-07-22T15:41:08-03:00
        local.oplog.rs    263ms    263ms      0ms
     admin.system.keys      0ms      0ms      0ms
    admin.system.roles      0ms      0ms      0ms
    admin.system.users      0ms      0ms      0ms
  admin.system.version      0ms      0ms      0ms
config.system.sessions      0ms      0ms      0ms
   config.transactions      0ms      0ms      0ms
local.replset.election      0ms      0ms      0ms
  local.system.replset      0ms      0ms      0ms
```

MongoDB – Essential Commands

```
$ mongostat --port 27017 --username admin --password admin --authenticationDatabase admin

insert query update delete getmore command dirty used flushes vsize  res qrw arw net_in net_out conn    set repl                time
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0 1.82G 101M 0|0 1|0   620b   38.2k    6 RepSet  PRI Jul 22 15:41:42.967
    *0    *0     *0     *0       0     1|0  0.0% 0.0%       0 1.82G 101M 0|0 1|0   518b   37.5k    6 RepSet  PRI Jul 22 15:41:43.982
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0 1.82G 101M 0|0 1|0   430b   38.7k    6 RepSet  PRI Jul 22 15:41:44.967
    *0    *0     *0     *0       1     3|0  0.0% 0.0%       0 1.82G 101M 0|0 1|0  2.76k   35.3k    6 RepSet  PRI Jul 22 15:41:46.109
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0 1.82G 101M 0|0 1|0   492b   44.2k    6 RepSet  PRI Jul 22 15:41:46.969
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0 1.82G 101M 0|0 1|0   527b   38.2k    6 RepSet  PRI Jul 22 15:41:47.966
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0 1.82G 101M 0|0 1|0   424b   38.1k    6 RepSet  PRI Jul 22 15:41:48.964
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0 1.82G 101M 0|0 1|0   526b   38.1k    6 RepSet  PRI Jul 22 15:41:49.963
    *0    *0     *0     *0       1     3|0  0.0% 0.0%       0 1.82G 101M 0|0 1|0  1.75k   39.5k    6 RepSet  PRI Jul 22 15:41:50.953
```

MongoDB Backup/Recovery

MongoDB – Backup / Recovery

```
$ mongo --port 27017 --username "admin" --password admin --authenticationDatabase "admin"
Implicit session: session { "id" : UUID("494d4e23-2018-4b22-91e2-b79faa4ab6cd") }
MongoDB server version: 4.2.8

RepSet:PRIMARY> use bd01
switched to db bd01

RepSet:PRIMARY> for (i=0; i<5000; i++) {db.customer.insert({code: i})}
WriteResult({ "nInserted" : 1 })

RepSet:PRIMARY> for (i=0; i<10000; i++) {db.product.insert({code: i})}
WriteResult({ "nInserted" : 1 })

RepSet:PRIMARY> db.getCollection("product").count();
10000

RepSet:PRIMARY> db.getCollection("product").stats();
{
        "ns" : "bd01.product",
        "size" : 360000,
        "count" : 10000,
        "avgObjSize" : 36,
        "storageSize" : 135168,
        "capped" : false,
        "nindexes" : 1,
        "totalIndexSize" : 102400,
        "indexSizes" : {
                "_id_" : 102400
        }
}
```

MongoDB – Backup / Recovery

```
$ mongodump --host localhost --port 27017 \
> --username admin --password admin --authenticationDatabase admin \
> --gzip --out /tmp/dump
2020-07-21T11:07:25.972-0300    writing admin.system.users to /tmp/dump/admin/system.users.bson.gz
2020-07-21T11:07:25.973-0300    done dumping admin.system.users (1 document)
2020-07-21T11:07:25.974-0300    writing admin.system.version to /tmp/dump/admin/system.version.bson.gz
2020-07-21T11:07:25.975-0300    done dumping admin.system.version (3 documents)
2020-07-21T11:07:25.976-0300    writing bd01.product to /tmp/dump/bd01/product.bson.gz
2020-07-21T11:07:26.031-0300    writing bd01.customer to /tmp/dump/bd01/customer.bson.gz
2020-07-21T11:07:26.037-0300    done dumping bd01.product (10000 documents)
2020-07-21T11:07:26.052-0300    done dumping bd01.customer (5000 documents)

$ mongodump --host localhost --port 27017 \
> --username admin --password admin --authenticationDatabase admin \
> --db bd01 \
> --gzip --out /tmp/dump
2020-07-21T11:07:33.650-0300    writing bd01.product to /tmp/dump/bd01/product.bson.gz
2020-07-21T11:07:33.685-0300    done dumping bd01.product (10000 documents)
2020-07-21T11:07:33.686-0300    writing bd01.customer to /tmp/dump/bd01/customer.bson.gz
2020-07-21T11:07:33.710-0300    done dumping bd01.customer (5000 documents)

$ mongodump --host localhost --port 27017 \
> --username admin --password admin --authenticationDatabase admin \
> --db bd01 \
> --collection product \
> --gzip --out /tmp/dump
2020-07-21T11:07:38.745-0300    writing bd01.product to /tmp/dump/bd01/product.bson.gz
2020-07-21T11:07:38.794-0300    done dumping bd01.product (10000 documents)
```

MongoDB – Backup / Recovery

```
$ tree /tmp/dump --charset=ascii
/tmp/dump
`-- bd01
    |-- customer.bson.gz
    |-- customer.metadata.json.gz
    |-- product.bson.gz
    `-- product.metadata.json.gz

1 directory, 4 files

$ mongo --port 27017 --username "admin" --password admin --authenticationDatabase "admin"
connecting to:
mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
MongoDB server version: 4.2.8

RepSet:PRIMARY> use bd01
switched to db bd01

RepSet:PRIMARY> db.dropDatabase()
{
        "dropped" : "bd01",
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1595343402, 3),
        },
        "operationTime" : Timestamp(1595343402, 3)
}
```

MongoDB – Backup / Recovery

```
$ mongorestore --host localhost --port 27017 \
> --username admin --password admin --authenticationDatabase admin \
> --gzip /tmp/dump
2020-07-21T12:03:33.052-0300    preparing collections to restore from
2020-07-21T12:03:33.053-0300    reading metadata for bd01.product from /tmp/dump/bd01/product.metadata.json.gz
2020-07-21T12:03:33.054-0300    reading metadata for bd01.customer from /tmp/dump/bd01/customer.metadata.json.gz
2020-07-21T12:03:33.121-0300    restoring bd01.customer from /tmp/dump/bd01/customer.bson.gz
2020-07-21T12:03:33.155-0300    restoring bd01.product from /tmp/dump/bd01/product.bson.gz
2020-07-21T12:03:33.391-0300    no indexes to restore
2020-07-21T12:03:33.391-0300    finished restoring bd01.customer (5000 documents, 0 failures)
2020-07-21T12:03:33.579-0300    no indexes to restore
2020-07-21T12:03:33.579-0300    finished restoring bd01.product (10000 documents, 0 failures)
2020-07-21T12:03:33.579-0300    15000 document(s) restored successfully. 0 document(s) failed to restore.

$ mongorestore --host localhost --port 27017 \
> --username admin --password admin --authenticationDatabase admin \
> --gzip --drop /tmp/dump
2020-07-21T12:04:13.811-0300    preparing collections to restore from
2020-07-21T12:04:13.825-0300    reading metadata for bd01.product from /tmp/dump/bd01/product.metadata.json.gz
2020-07-21T12:04:13.848-0300    restoring bd01.product from /tmp/dump/bd01/product.bson.gz
2020-07-21T12:04:13.891-0300    reading metadata for bd01.customer from /tmp/dump/bd01/customer.metadata.json.gz
2020-07-21T12:04:13.919-0300    restoring bd01.customer from /tmp/dump/bd01/customer.bson.gz
2020-07-21T12:04:14.217-0300    no indexes to restore
2020-07-21T12:04:14.217-0300    finished restoring bd01.customer (5000 documents, 0 failures)
2020-07-21T12:04:14.371-0300    no indexes to restore
2020-07-21T12:04:14.371-0300    finished restoring bd01.product (10000 documents, 0 failures)
2020-07-21T12:04:14.371-0300    15000 document(s) restored successfully. 0 document(s) failed to restore.
```

MongoDB – Backup / Recovery

```
$ mongodump --gzip --numParallelCollections=10 --verbose=3 \
> --port 27017 --username admin --password admin --authenticationDatabase admin \
> --out /tmp/dump \
> --oplog

2020-07-22T09:51:00.795-0300    initializing mongodump object
2020-07-22T09:51:00.796-0300    will listen for SIGTERM, SIGINT, and SIGKILL
2020-07-22T09:51:00.902-0300    starting Dump()
2020-07-22T09:51:00.902-0300    found databases: admin, bd01, config, local
2020-07-22T09:51:00.903-0300    enqueued collection 'admin.system.version'
2020-07-22T09:51:00.903-0300    enqueued collection 'admin.system.users'
2020-07-22T09:51:00.903-0300    will not dump system collection 'admin.system.keys'
2020-07-22T09:51:00.904-0300    enqueued collection 'bd01.product'
2020-07-22T09:51:00.904-0300    enqueued collection 'bd01.customer'
2020-07-22T09:51:00.905-0300    will not dump system collection 'config.system.sessions'
2020-07-22T09:51:00.905-0300    will not dump system collection 'config.transactions'
2020-07-22T09:51:00.905-0300    determined cluster to be a replica set
2020-07-22T09:51:00.905-0300    oplog located in local.oplog.rs
2020-07-22T09:51:00.905-0300    enqueued collection '.oplog'
2020-07-22T09:51:00.905-0300    dump phase I: metadata, indexes, users, roles, version
2020-07-22T09:51:00.905-0300            reading indexes for `bd01.product`
2020-07-22T09:51:00.908-0300    writing admin.system.users to /tmp/dump/admin/system.users.bson.gz
2020-07-22T09:51:00.989-0300    checking if oplog entry {1595422255 1} still exists
2020-07-22T09:51:00.990-0300    oldest oplog entry has timestamp {1595255594 1}
2020-07-22T09:51:00.990-0300    oplog entry {1595422255 1} still exists
2020-07-22T09:51:00.990-0300    writing captured oplog to
2020-07-22T09:51:00.990-0300    not counting query on .oplog
2020-07-22T09:51:00.991-0300            dumped 1 oplog entry
2020-07-22T09:51:00.991-0300    checking again if oplog entry {1595422255 1} still exists
2020-07-22T09:51:00.992-0300    oldest oplog entry has timestamp {1595255594 1}
2020-07-22T09:51:00.992-0300    oplog entry {1595422255 1} still exists
2020-07-22T09:51:00.992-0300    finishing dump
```

MongoDB – Backup / Recovery

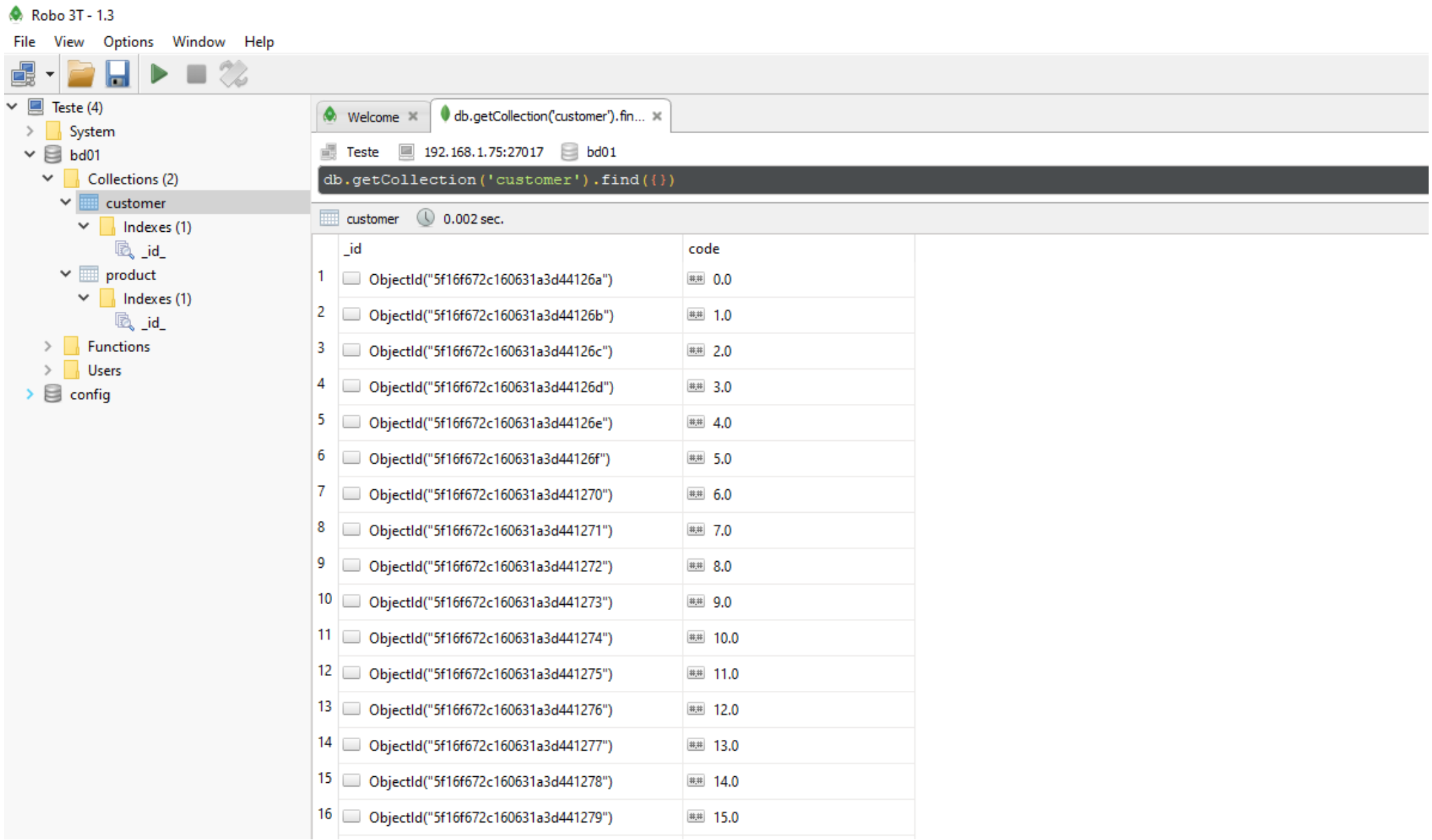
```
$ mongorestore --gzip --verbose=3 \
> --port 27017 --username admin --password admin --authenticationDatabase admin \
> --drop --oplogReplay /tmp/dump

2020-07-22T09:54:21.743-0300    using write concern: &{majority false 0}
2020-07-22T09:54:21.794-0300    checking options
2020-07-22T09:54:21.794-0300            dumping with object check disabled
2020-07-22T09:54:21.794-0300    will listen for SIGTERM, SIGINT, and SIGKILL
2020-07-22T09:54:21.817-0300    connected to node type: replset
2020-07-22T09:54:21.817-0300    mongorestore target is a directory, not a file
2020-07-22T09:54:21.817-0300    preparing collections to restore from
2020-07-22T09:54:21.817-0300    using /tmp/dump as dump root directory
2020-07-22T09:54:21.817-0300    reading collections for database bd01 in bd01
2020-07-22T09:54:21.817-0300    found collection bd01.customer bson to restore to bd01.customer
2020-07-22T09:54:21.817-0300    found collection metadata from bd01.customer to restore to bd01.customer
2020-07-22T09:54:21.817-0300    found collection bd01.product bson to restore to bd01.product
2020-07-22T09:54:21.817-0300    found collection metadata from bd01.product to restore to bd01.product
2020-07-22T09:54:21.817-0300    found oplog.bson file to replay
2020-07-22T09:54:21.817-0300    enqueued collection '.oplog'
2020-07-22T09:54:21.841-0300    ending restore routine with id=1, no more work to do
2020-07-22T09:54:21.841-0300    starting restore routine with id=2
2020-07-22T09:54:23.160-0300    restoring users to temporary collection
2020-07-22T09:54:23.160-0300    using 1 insertion workers
2020-07-22T09:54:23.218-0300    merging users/roles from temp collections
2020-07-22T09:54:23.219-0300    dropping temporary collection admin.tempusers
2020-07-22T09:54:23.251-0300    replaying oplog
2020-07-22T09:54:23.251-0300    applied 2 oplog entries
2020-07-22T09:54:23.251-0300    15000 document(s) restored successfully. 0 document(s) failed to restore.
```

MongoDB – Backup / Recovery

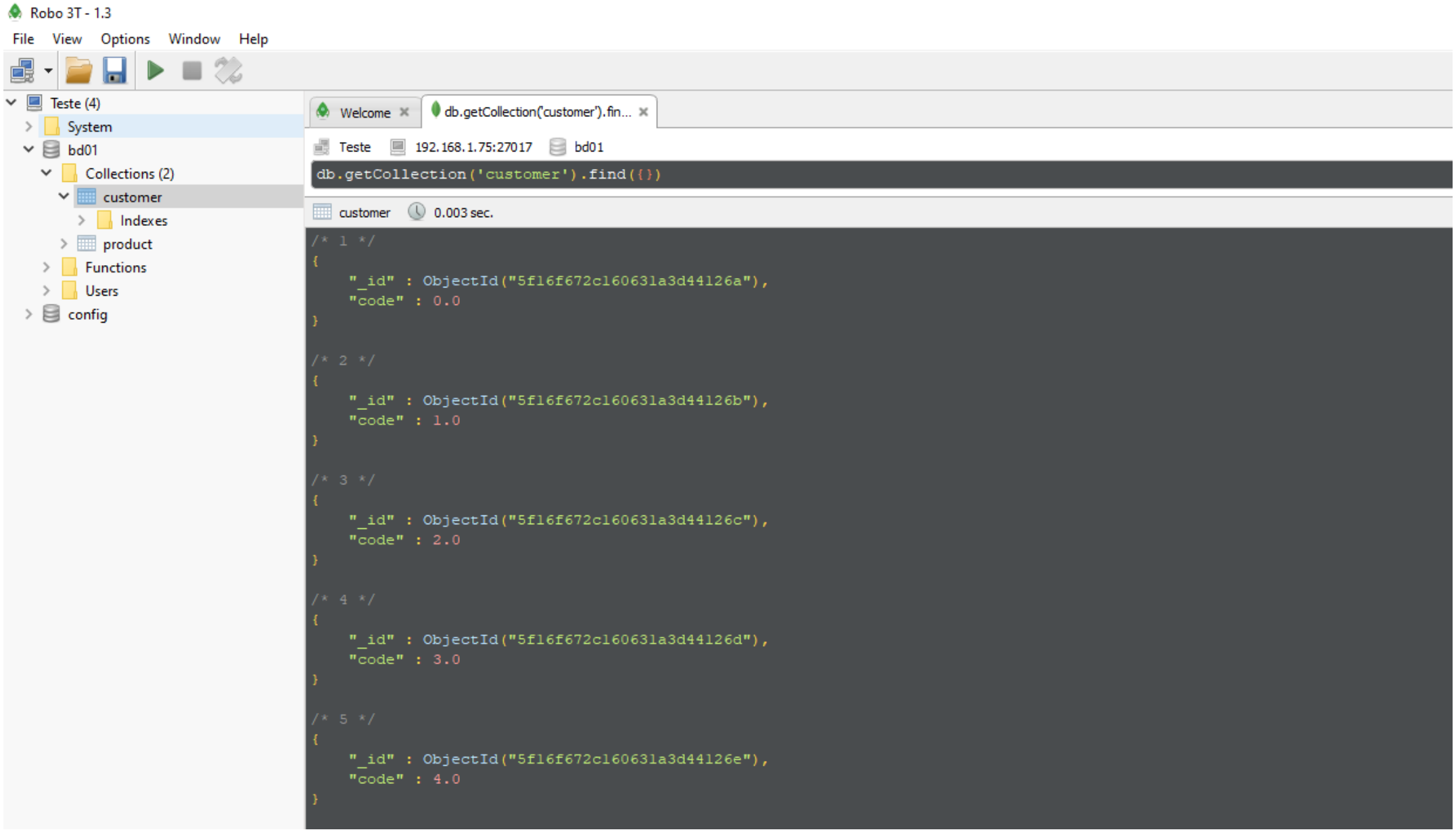
```
-- Export
$ mongoexport --host localhost --port 27017
--username admin --password admin --authenticationDatabase admin
--db bd01 --collection product
--out /tmp/dump/product.json

-- Import
$ mongoimport --upsert --host --port 27017
--username admin --password admin --authenticationDatabase admin
--db bd01 --collection product
--file /tmp/dump/product.json
```

MongoDB
Graphical Interface Clients

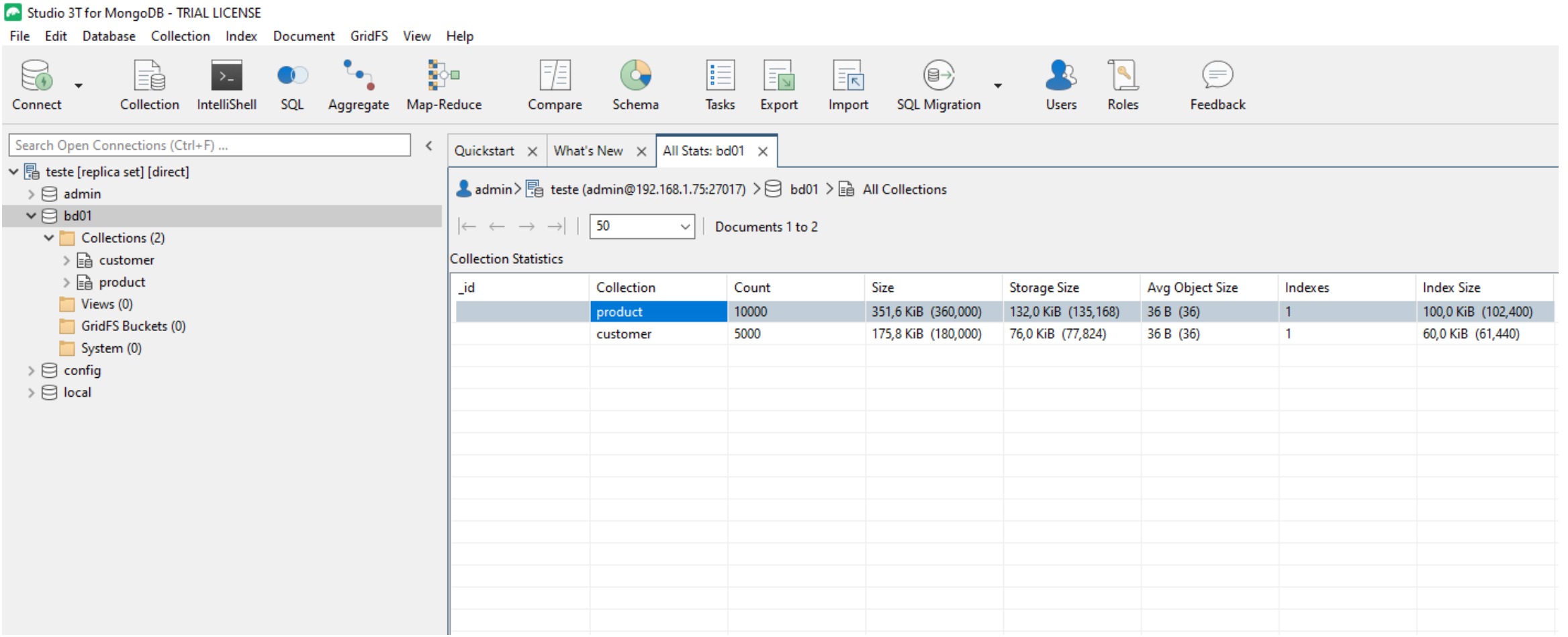
MongoDB – Graphical Interface Clients



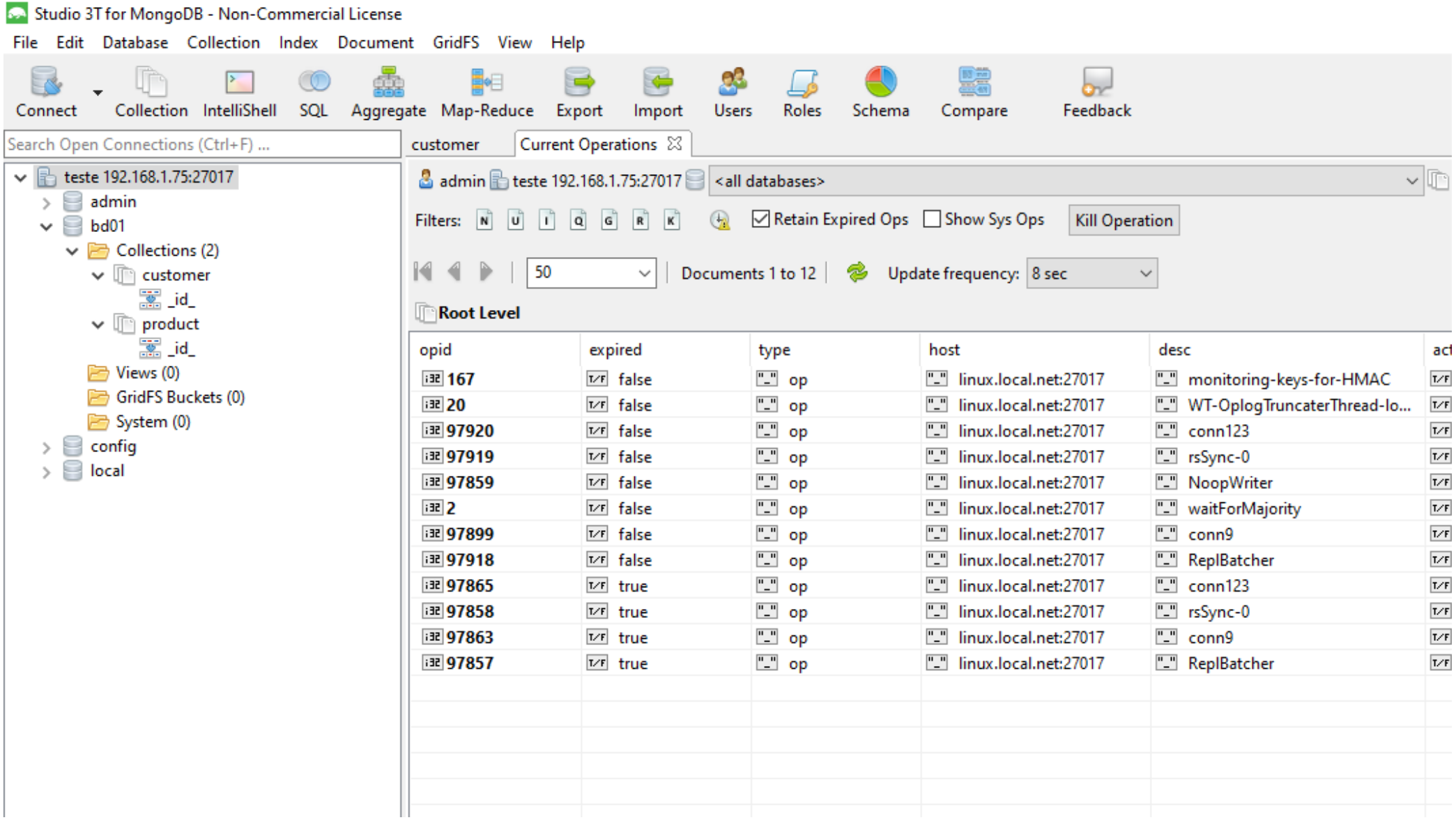
MongoDB – Graphical Interface Clients



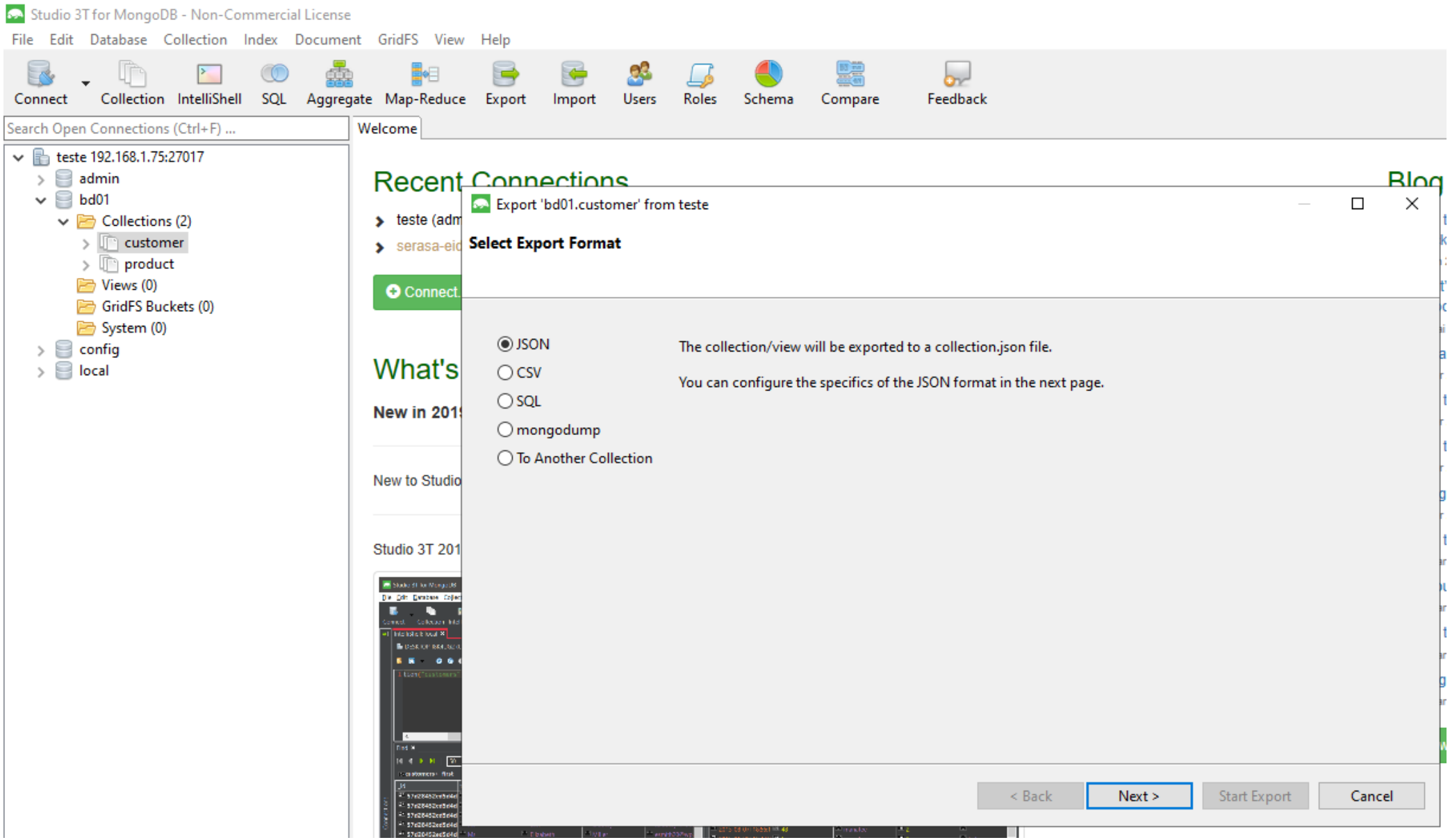
MongoDB – Graphical Interface Clients



MongoDB – Graphical Interface Clients



MongoDB – Graphical Interface Clients



MongoDB – Graphical Interface Clients

Studio 3T for MongoDB - Non-Commercial License

File Edit Database Collection Index Document GridFS View Help

Connect Collection IntelliShell SQL Aggregate Map-Reduce Export Import Users Roles Schema Compare Feedback

Search Open Connections (Ctrl+F) ...

Welcome

teste 192.168.1.75:27017
admin
bd01
Collections (2)
custo
produ
Views (0)
GridFS Bu
System (0
config
local

Recent Connections

teste (admin@192.168.1.75:27017)

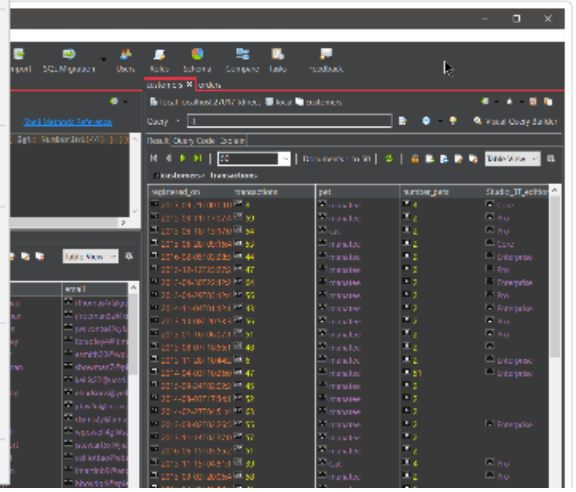
d on 20/05/2018 (54.233.170.145:27017)

Knowledge Base and get up to speed.

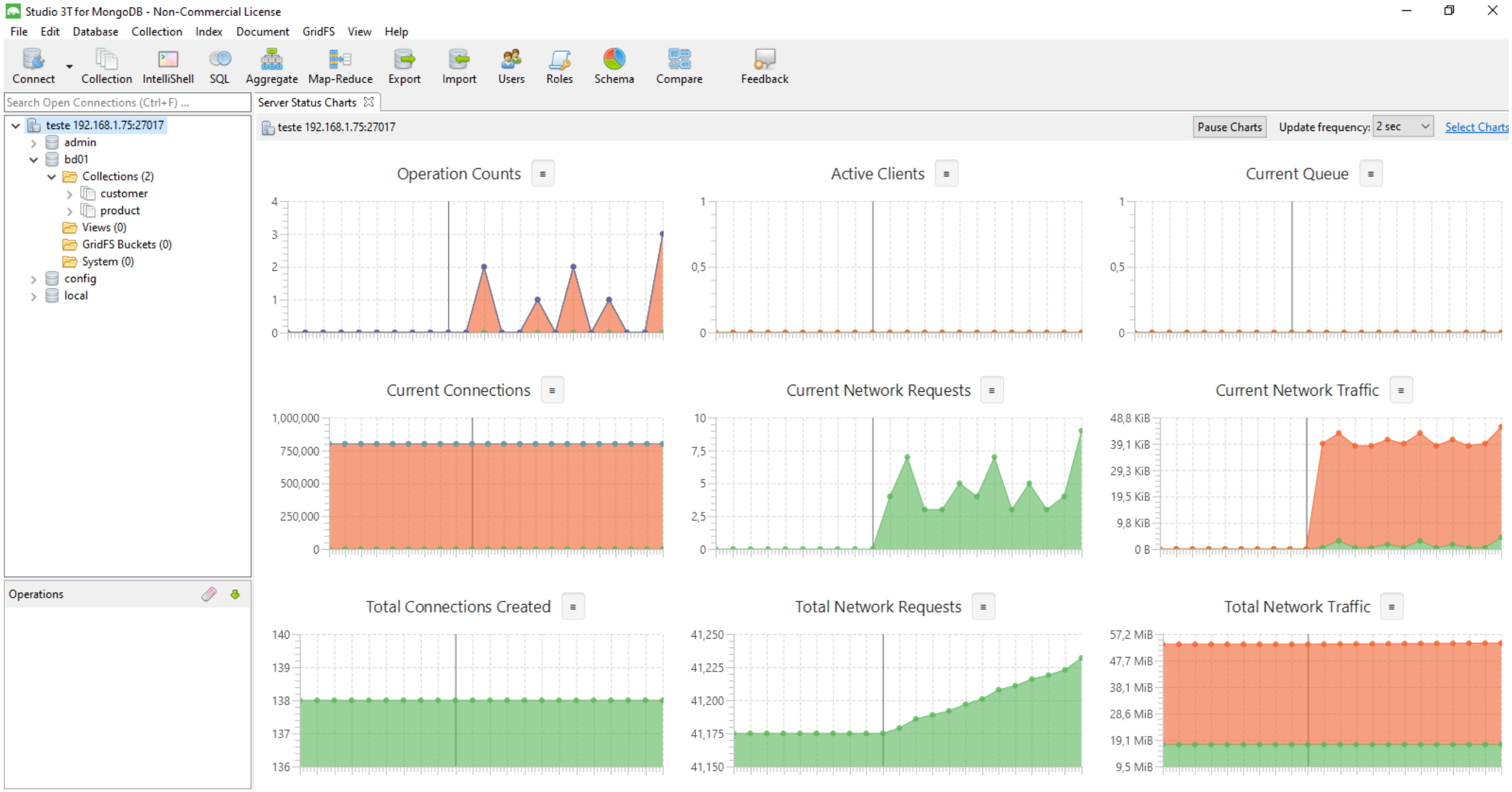
ark Theme, as well as a number of small bug fixes and improvements.

Operations

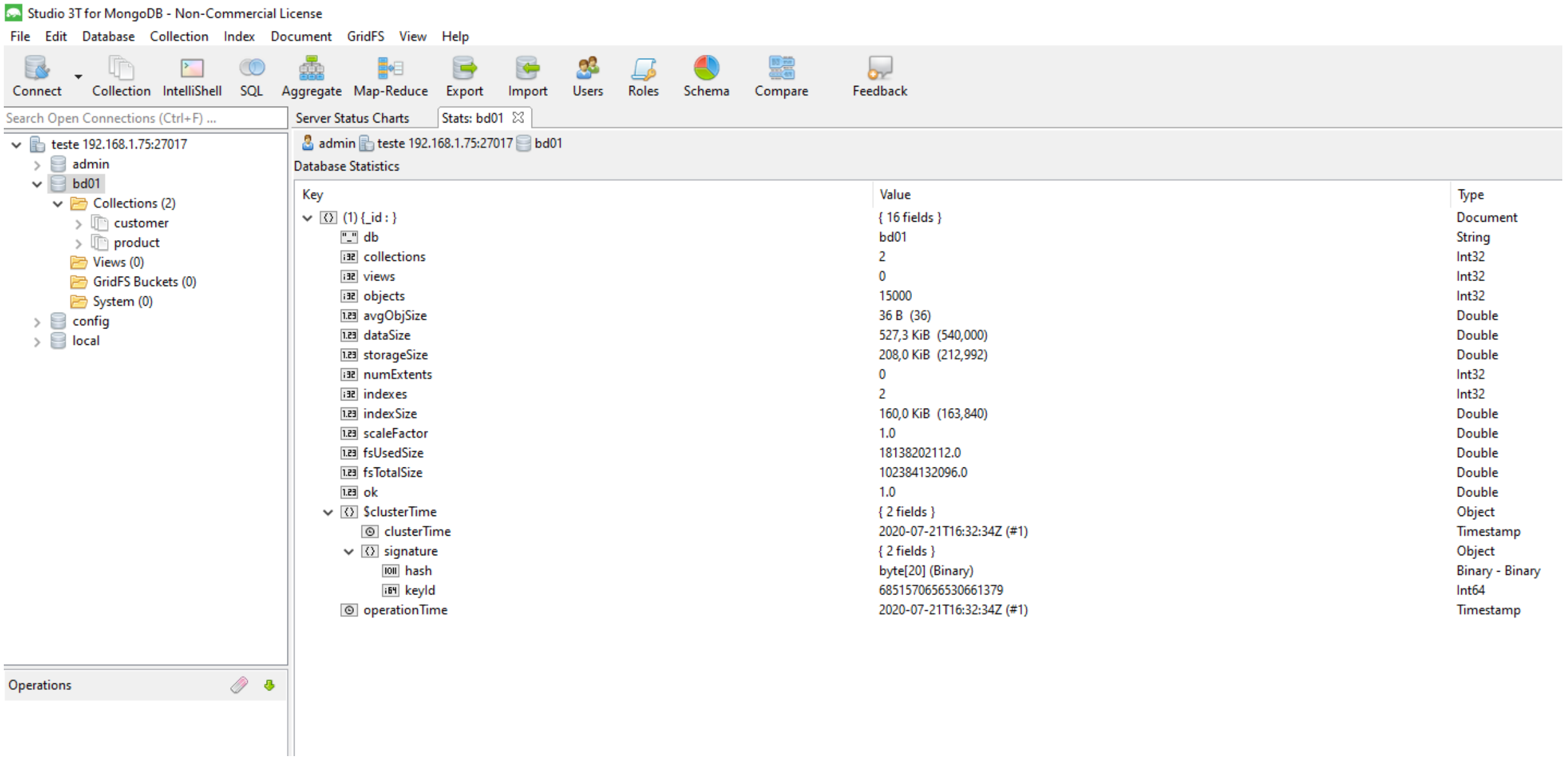
- Open Collection Tab — Enter
- Open Collection With Custom Page Size...
- Open IntelliShell — Ctrl+L
- Open SQL — Ctrl+Shift+L
- Open Aggregation Screen — F4
- Open Map-Reduce — Ctrl+M
- Export Collection...
- Import Data...
- Copy Collection — Ctrl+C
- Rename Collection...
- Clear Collection
- Drop Collection — Del
- Add Index...
- Add/Edit Validator...
- Add View Here...
- Analyze Schema...
- Compare To...
- Current Operations
- Collection Statistics
- Server Info
- Refresh Selected Item — Ctrl+R
- Refresh All — Ctrl+Shift+R
- Choose Color
- Disconnect — Ctrl+Alt+D



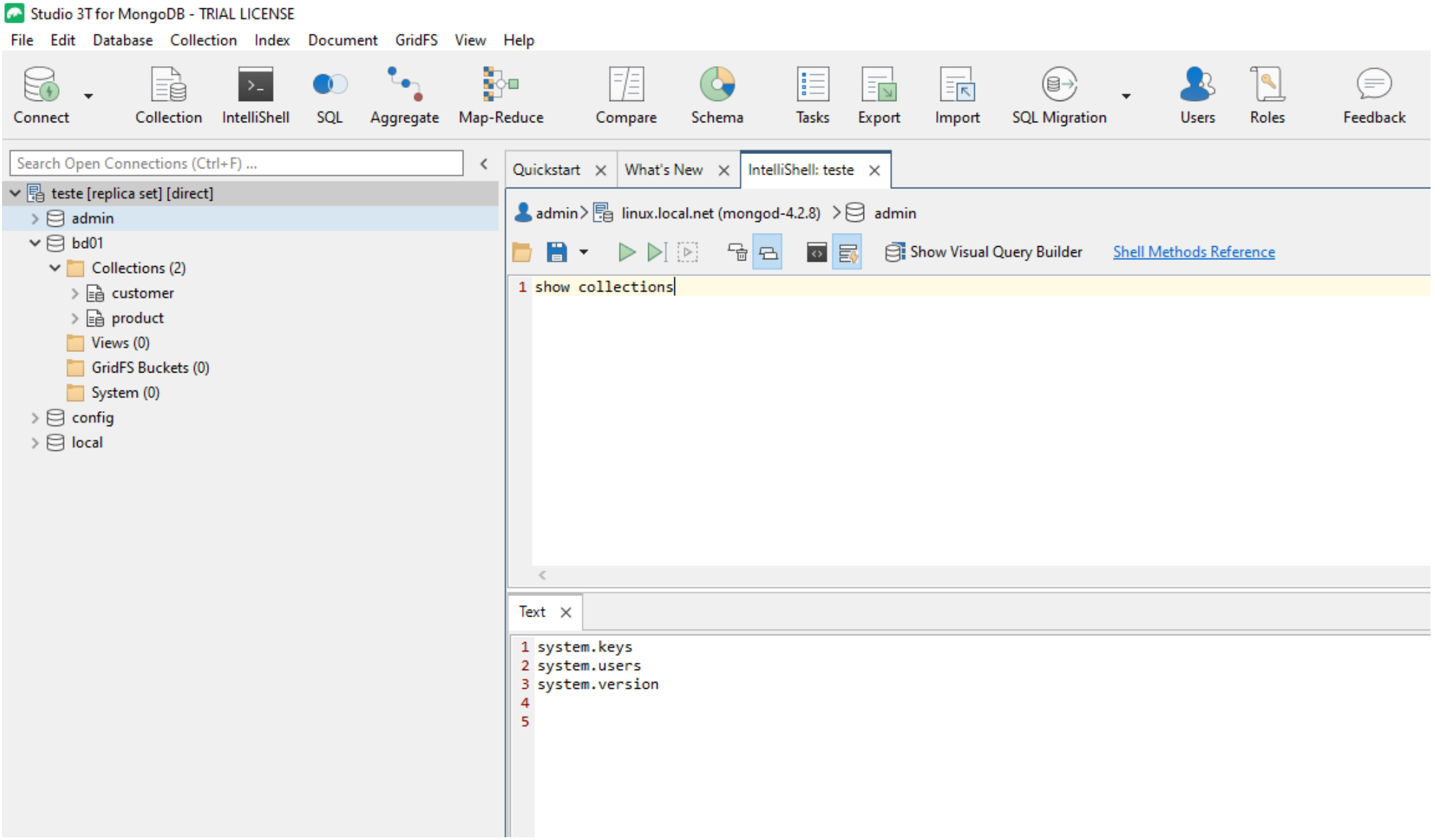
MongoDB – Graphical Interface Clients



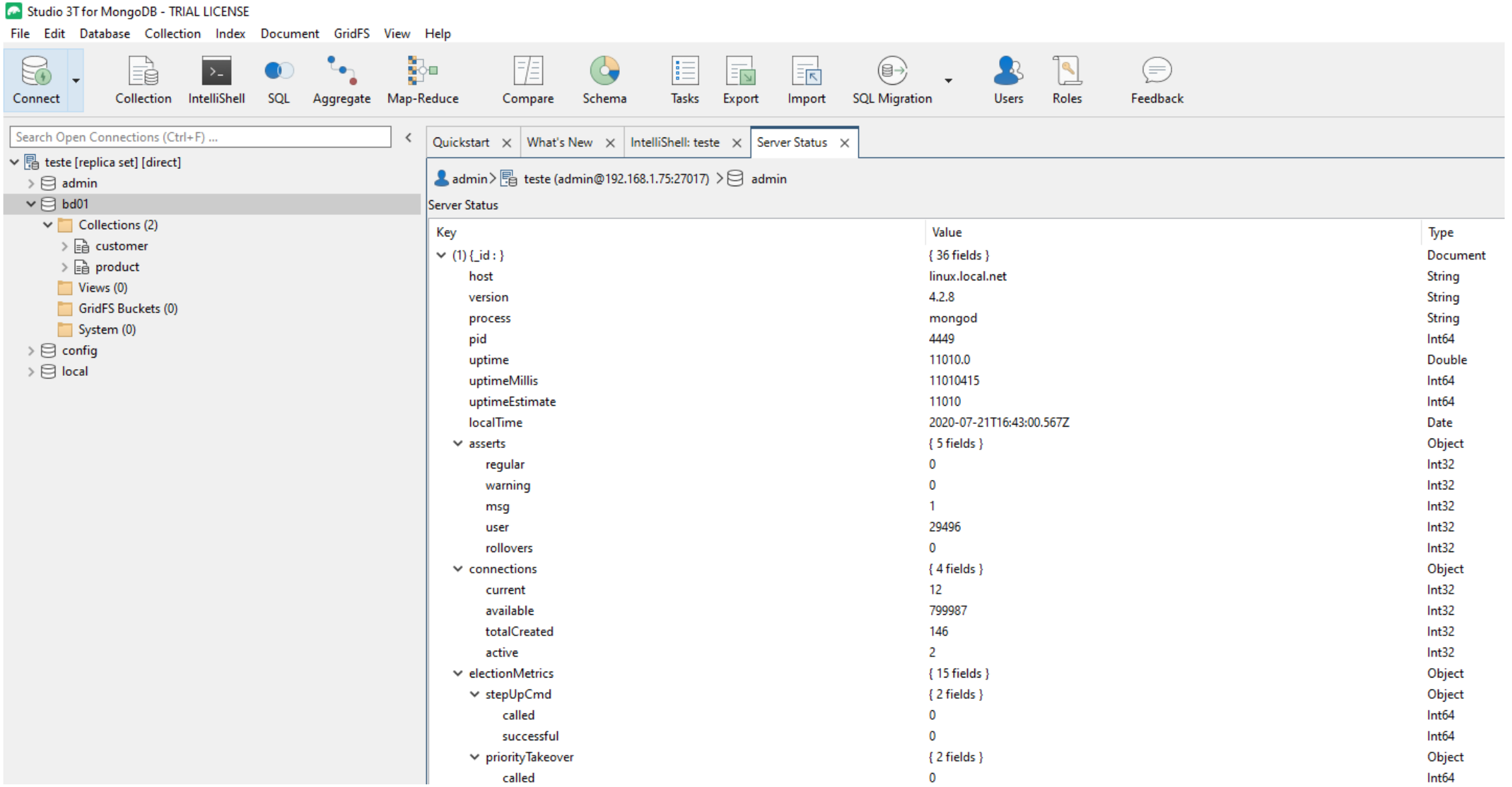
MongoDB – Graphical Interface Clients



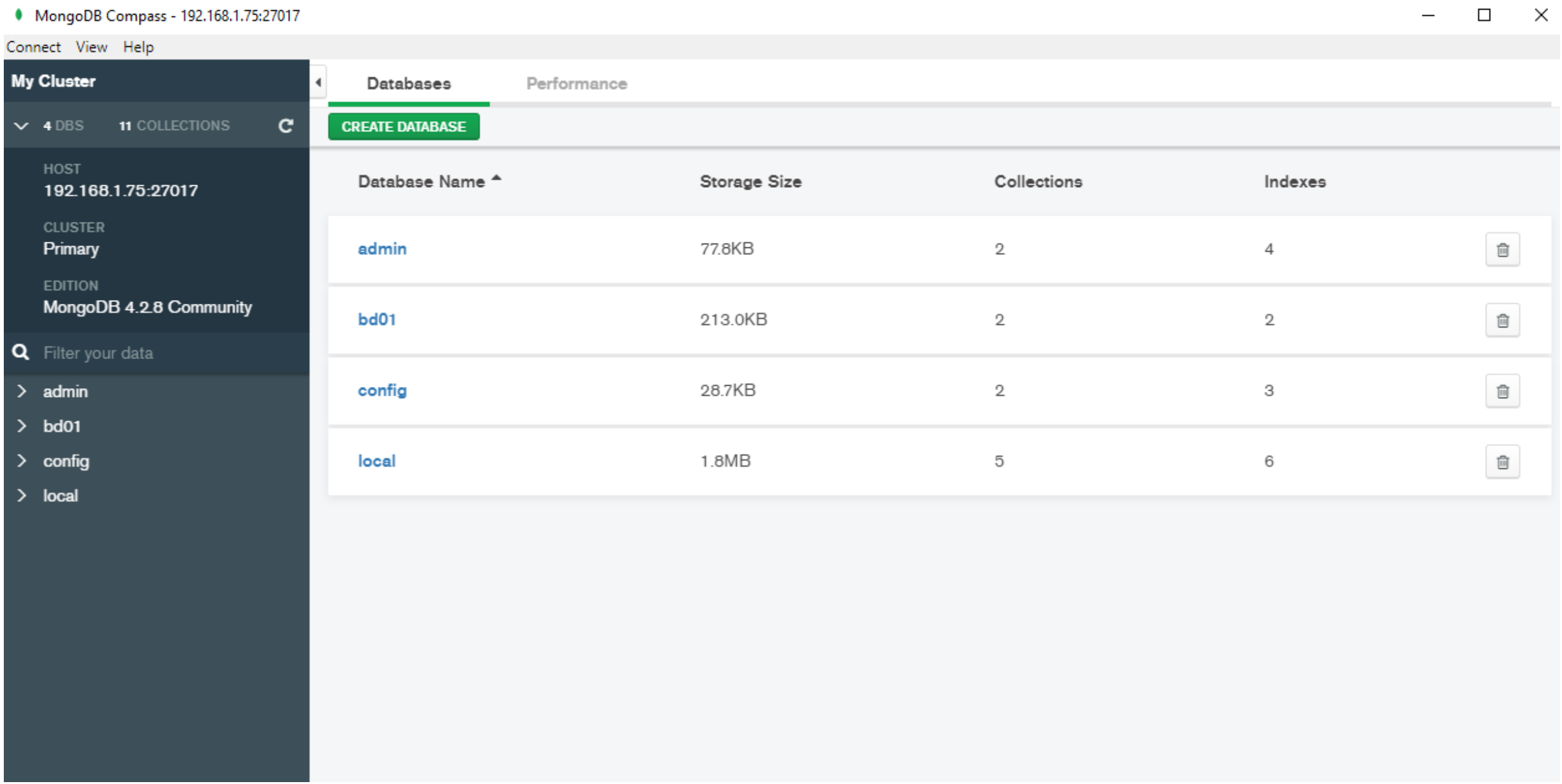
MongoDB – Graphical Interface Clients



MongoDB – Graphical Interface Clients

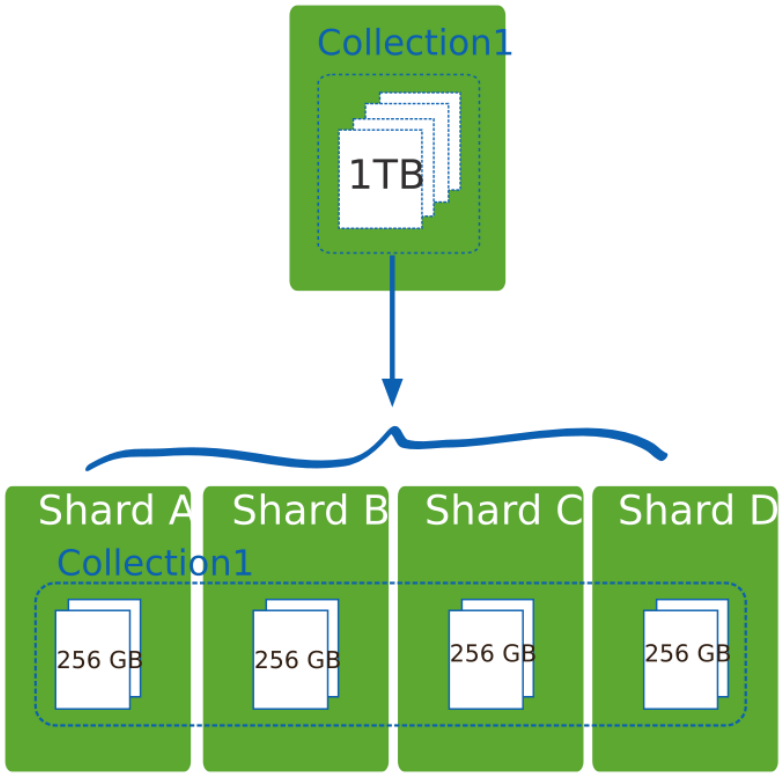


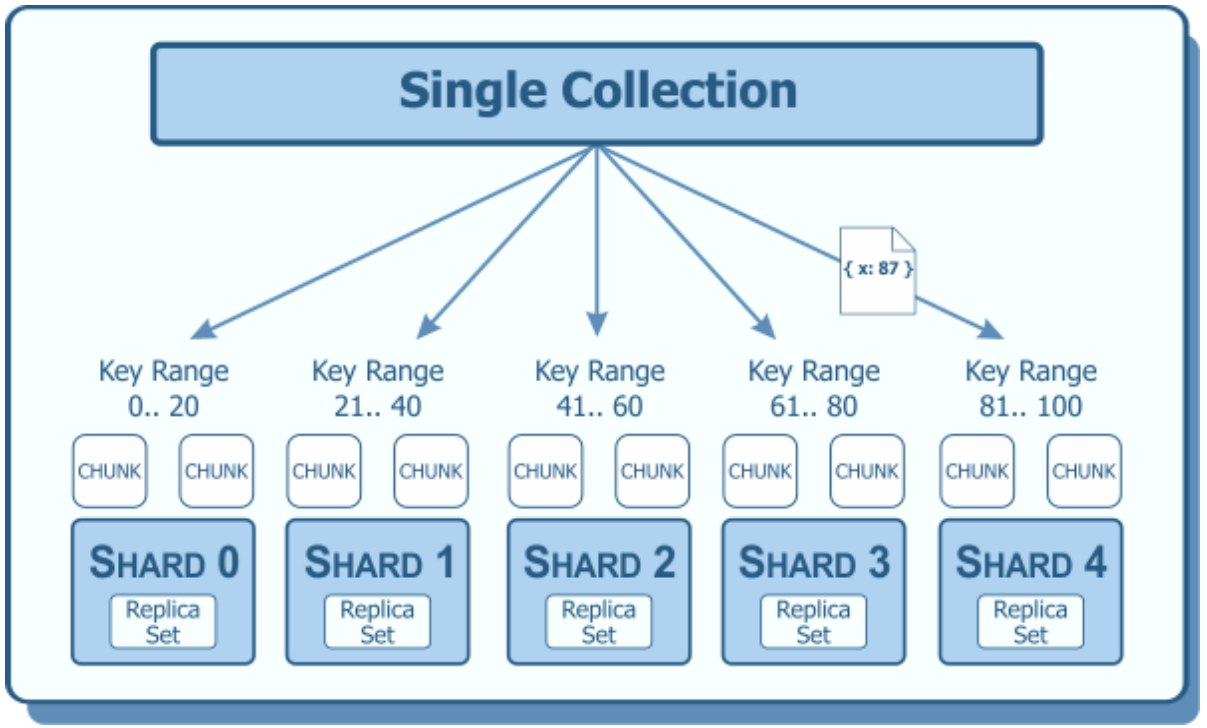
MongoDB – Graphical Interface Clients



MongoDB Sharding

MongoDB – Sharding





RDBMS	MongoDB
Database	Database
Table, view	Collection
Row	Documents(JSON,BSON)
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference
Partition Key	Shard Key

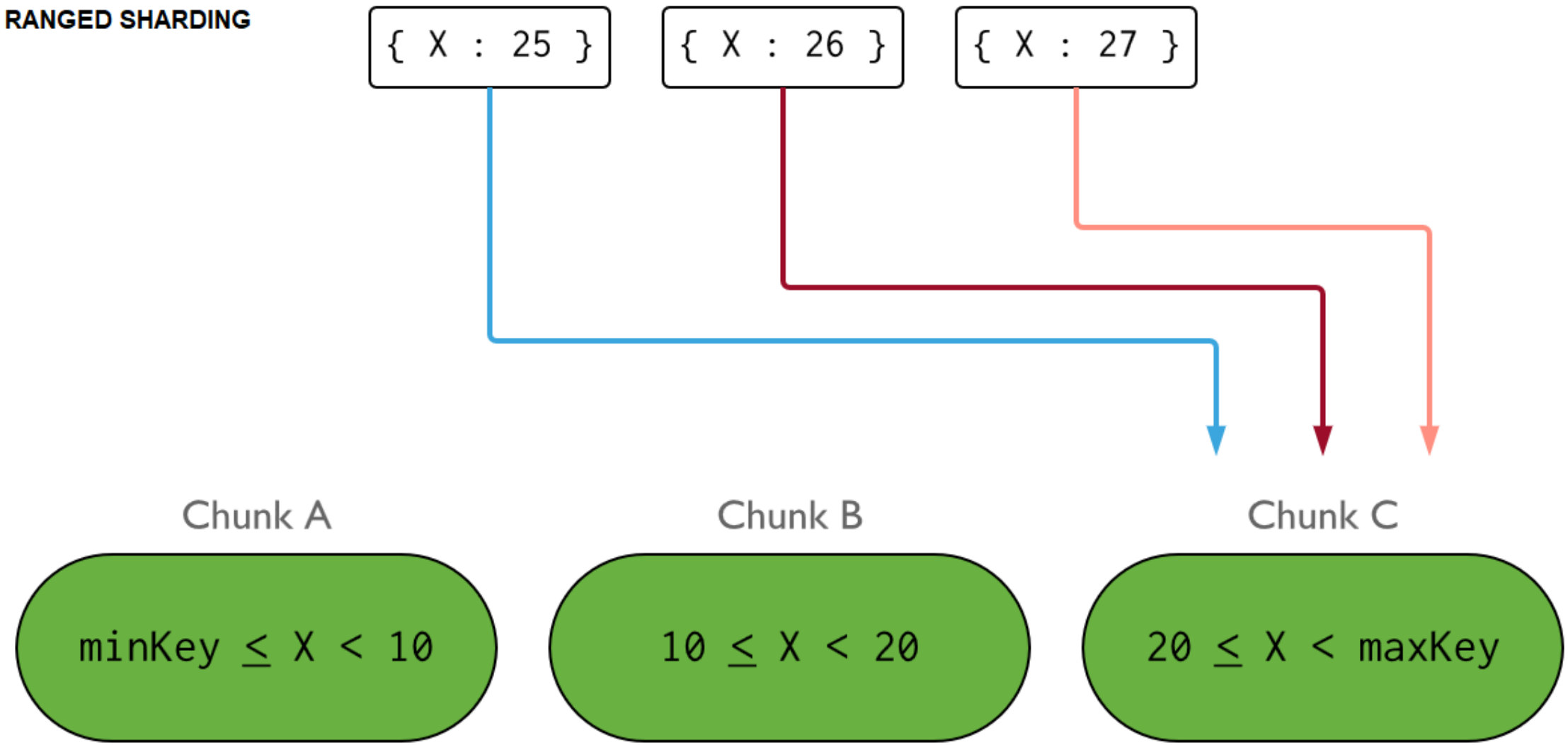
The collection can be splitted up into a dozen chunks, where each chunk is a subset of your data. These are listed by shard key range (the {minValue} -->>{maxValue} denotes the range of each chunk). Sharding splits the collection into many chunks based on shard key ranges.

IMPORTANT:

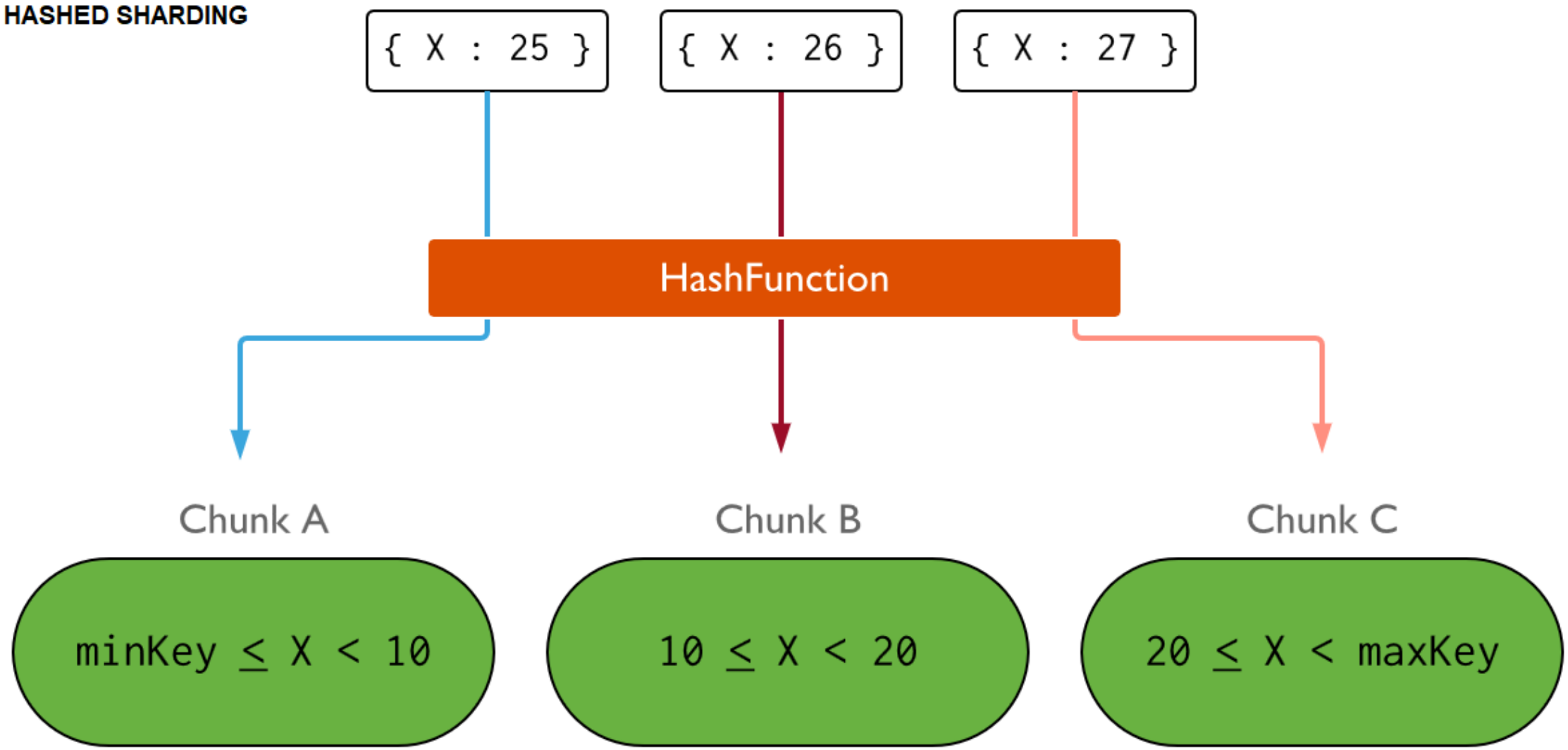
Once you shard a collection, the shard key and the shard key values are immutable; i.e.

- You cannot select a different shard key for that collection.
- You cannot update the values of the shard key fields.

MongoDB – Sharding

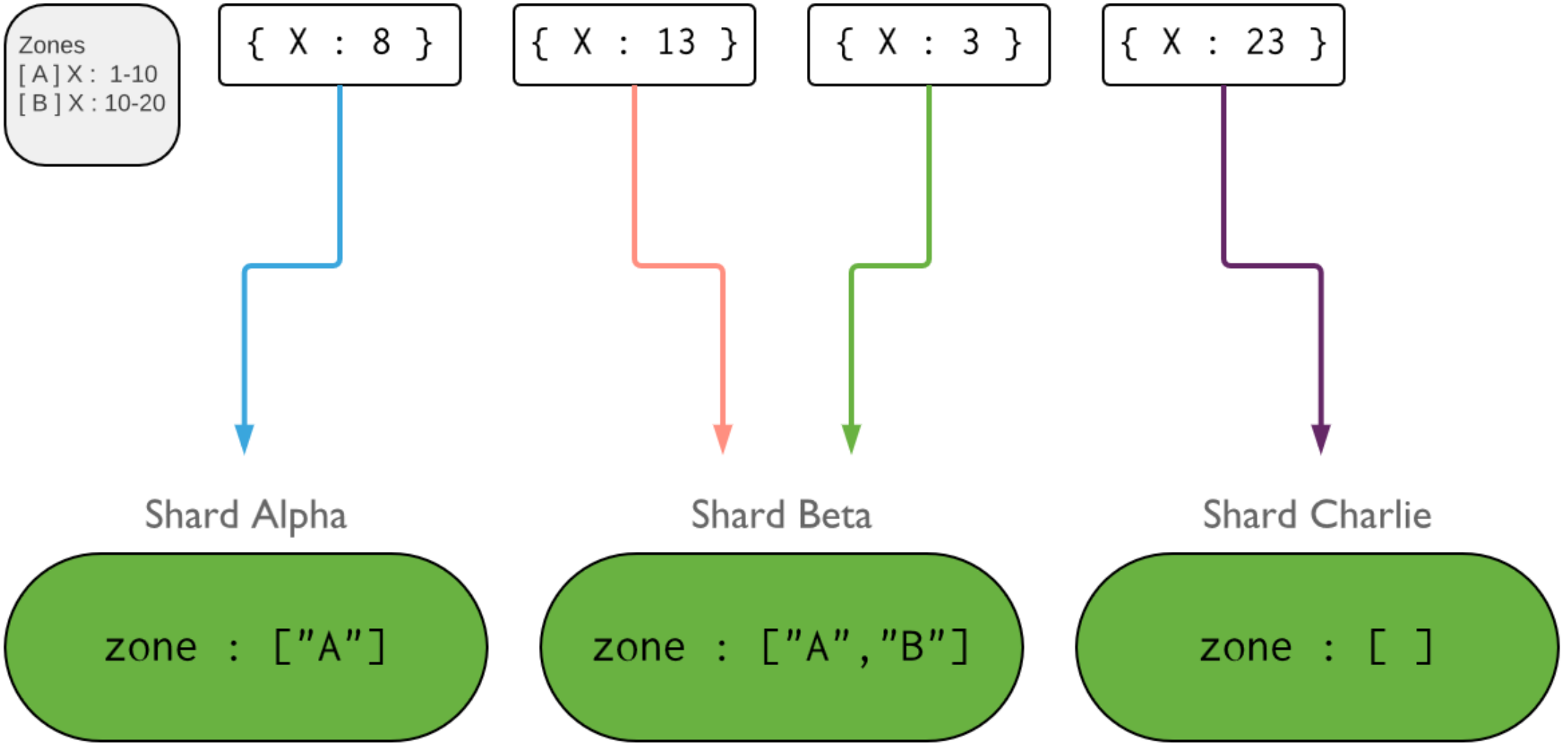


MongoDB – Sharding

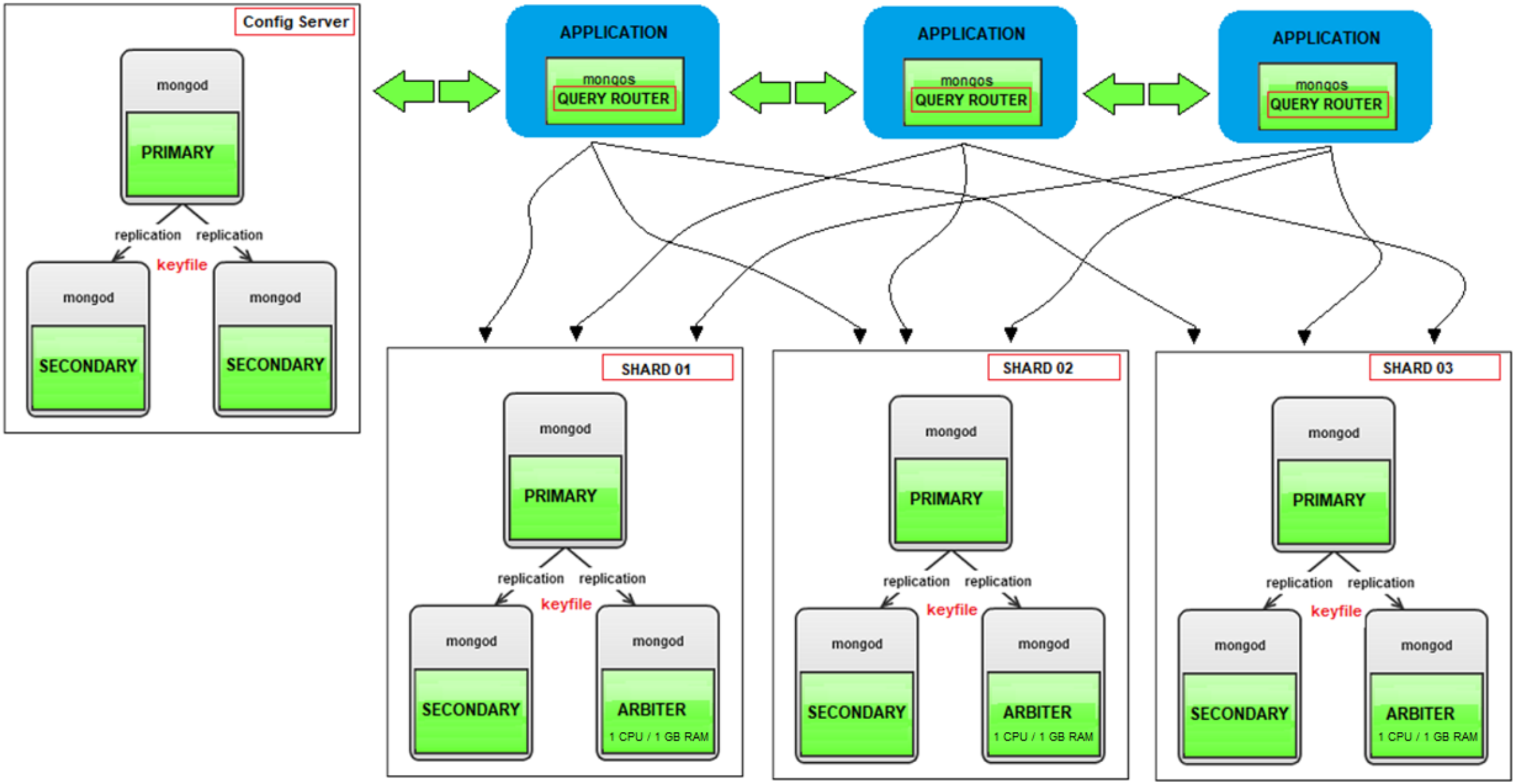


MongoDB – Sharding

TAG-AWARE SHARDING



MongoDB – Sharding



END