


Structure de projet recommandée

```
atelier-mongodb/  
├── docs/  
│   └── rapport.md  
├── mongo/  
│   ├── standalone/  
│   ├── replicaset/  
│   └── sharding/          # Bonus  
├── integration/  
│   └── <langage>/  
│       └── tests/  
└── README.md
```

Parties 1 – MongoDB Standalone

```
MongoDB >  docker-compose.yml > ...  
1  # Use root/example as user/password credentials  
2  version: '3.1'  
3  
4  >Run All Services  
5  services:  
6    >Run Service  
7    mongo:  
8      image: mongo  
9      restart: always  
10     environment:  
11       MONGO_INITDB_ROOT_USERNAME: root  
12       MONGO_INITDB_ROOT_PASSWORD: example  
13  
14    >Run Service  
15    mongo-express:  
16      image: mongo-express  
17      restart: always  
18      ports:  
19        - 8081:8081  
20      environment:  
21        ME_CONFIG_MONGODB_ADMINUSERNAME: root  
22        ME_CONFIG_MONGODB_ADMINPASSWORD: example  
23        ME_CONFIG_MONGODB_URL: mongodb://root:example@mongo:27017/  
24        ME_CONFIG_BASICAUTH: false
```

9826e539c051

mongo-express:latest

8081:8081

STATUS

Running (7 minutes ago)

Logs

Inspect

Bind mounts

Exec

Files

Stats

2025-04-22 12:56:31 /docker-entrypoint.sh: connect: Connection refused

2025-04-22 12:56:31 /docker-entrypoint.sh: line 15: /dev/tcp/mongo/27017: Connection refused

2025-04-22 12:56:32 /docker-entrypoint.sh: connect: Connection refused

2025-04-22 12:56:32 /docker-entrypoint.sh: line 15: /dev/tcp/mongo/27017: Connection refused

2025-04-22 12:56:33 /docker-entrypoint.sh: connect: Connection refused

2025-04-22 12:56:33 /docker-entrypoint.sh: line 15: /dev/tcp/mongo/27017: Connection refused

2025-04-22 12:56:34 /docker-entrypoint.sh: connect: Connection refused

2025-04-22 12:56:34 /docker-entrypoint.sh: line 15: /dev/tcp/mongo/27017: Connection refused

2025-04-22 12:56:35 Server is open to allow connections from anyone (0.0.0.0)

2025-04-22 12:56:35 Basic authentication is disabled. It is recommended to set the useBasicAuth to true in the config.js.

2025-04-22 14:23:34 /docker-entrypoint.sh: connect: Connection refused

2025-04-22 14:23:35 /docker-entrypoint.sh: connect: Connection refused

2025-04-22 14:23:35 /docker-entrypoint.sh: line 15: /dev/tcp/mongo/27017: Connection refused

2025-04-22 14:23:37 Server is open to allow connections from anyone (0.0.0.0)

2025-04-22 14:23:37 Basic authentication is disabled. It is recommended to set the useBasicAuth to true in the config.js.

2025-04-22 14:24:01 GET /public/vendor-93f5fc3ae20e0dfd68cb.min.js 304 2.823 ms - -

2025-04-22 14:24:01 GET /public/index-56afe067afbbbde795be.min.js 304 3.447 ms - -

2025-04-22 14:24:01 GET /public/img/gears.gif 304 1.941 ms - -

2025-04-22 14:24:01 GET /public/index-56afe067afbbbde795be.min.js 304 1.468 ms - -

2025-04-22 14:24:01 GET /public/vendor-93f5fc3ae20e0dfd68cb.min.js 304 3.982 ms - -

2025-04-22 14:24:01 GET /public/fonts/glyphicons-halflings-regular.woff2 304 4.539 ms - -

http://localhost:8081

Mongo Express

Database

Databases

Database Name

+ Create Database

View

admin

Del

View

config

Del

View

local

Del

Server Status

Hostname	5eb6290c50d5	MongoDB Version	8.0.8
Uptime	42 seconds	Node Version	18.20.3
Server Time	Tue, 22 Apr 2025 10:48:19 GMT	V8 Version	10.2.154.26-node.37
Current Connections	3	Available Connections	838857
Active Clients	0	Queued Operations	0
Clients Reading	0	Clients Writing	0
Read Lock Queue	0	Write Lock Queue	0
Disk Flushes		Last Flush	

```

MongoDB git:(main) docker-compose up
WARN [0000] /Users/clementgarcia/Desktop/Ynov24_25/NoSQL/MongoDB/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potent
ial confusion
[*] Running 6/18
[*] mongo-express [ ] Pulling
  : c6b39de5b339 Waiting 9.4s
  : 41c65f6bdf40 Waiting 9.4s
  : 23ee6393f192 Waiting 9.4s
  : 8f5ec9da9bc0 Waiting 9.4s
  : f541ecl4067 Waiting 9.4s
  : 4a17d49bb7c9 Waiting 9.4s
  : 2aa624262f6c Pulling fs layer 9.4s
  : f2d515a8401d Pulling 9.4s
[*] mongo [=====] Pulling
  : 49b96e96358d Extracting [=====] 26.84MB/28.85MB 9.4s
  : 647de49b3f54 Download complete 0.8s
  : 268841dcd611 Download complete 1.3s
  : 638631464f16 Download complete 8.7s
  : 7e63d955562b Download complete 6.9s
  : c2096b08e1e2 Download complete 9.1s
  : 9d88392cb05a Waiting 9.4s
  : d04becf8fed0 Download complete 9.3s

```

```

└─ MongoDB git:(main) docker exec -it mongodb-mongo-1 mongosh -u root -p example --authenticationDatabase admin

Current Mongosh Log ID: 6807878819fc600e1465d0fa
Connecting to:      mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&auth
source=admin&appName=mongosh+2.5.0
Using MongoDB:      8.0.8
Using Mongosh:      2.5.0

For mongosh info see: https://www.mongodb.com/docs/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.

=====
The server generated these startup warnings when booting
2025-04-22T10:52:45.157+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See
http://dochub.mongodb.org/core/prodnotes-filesystem
2025-04-22T10:52:45.999+00:00: For customers running the current memory allocator, we suggest changing the contents of
the following sysfsFile
2025-04-22T10:52:45.999+00:00: We suggest setting the contents of sysfsFile to 0.
2025-04-22T10:52:45.999+00:00: vm.max_map_count is too low
2025-04-22T10:52:45.999+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
=====

test>

```

Dans la base de données “testdb”

```
test> use testdb
...
switched to db testdb
testdb> 
```

Entrer des utilisateur :

```

test> use testdb
...
switched to db testdb
testdb> db.testcoll.insertMany([
...   { name: "Alice", age: 25 },
...   { name: "Bob", age: 30 },
...   { name: "Charlie", age: 28 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6807882119fc600e1465d0fb'),
    '1': ObjectId('6807882119fc600e1465d0fc'),
    '2': ObjectId('6807882119fc600e1465d0fd')
  }
}
testdb>

```

3. Lire tous les documents

```

testdb> db.testcoll.find().pretty()
...
[
  { _id: ObjectId('6807882119fc600e1465d0fb'), name: 'Alice', age: 25 },
  { _id: ObjectId('6807882119fc600e1465d0fc'), name: 'Bob', age: 30 },
  {
    _id: ObjectId('6807882119fc600e1465d0fd'),
    name: 'Charlie',
    age: 28
  }
]
testdb> 4

```

4. Rechercher ceux qui ont plus de 26 ans

```

testdb> db.testcoll.find({ age: { $gt: 26 } })
...
[
  { _id: ObjectId('6807882119fc600e1465d0fc'), name: 'Bob', age: 30 },
  {
    _id: ObjectId('6807882119fc600e1465d0fd'),
    name: 'Charlie',
    age: 28
  }
]
testdb>

```

5. Vérifier le fonctionnement avec une interface CLI ou graphique (mongosh, Compass...)

Viewing Collection: testcoll

[New Document](#) [New Index](#)







Simple

Advanced

String

Find

Delete all 3 documents retrieved

	_id	name	age
 	6807882119fc600e1465d0fb	Alice	25
 	6807882119fc600e1465d0fc	Bob	30
 	6807882119fc600e1465d0fd	Charlie	28

```
testdb> db.testcoll.deleteOne({ name: "Charlie" })
{ acknowledged: true, deletedCount: 1 }
testdb>
```

Commande pour se connecter :

docker exec -it mongodb-mongo-1 mongosh -u root -p example
--authenticationDatabase admin

Entrer dans la BDD :

use testdb

Ajouter : db.testcoll.insertMany([{ name: "Alice", age: 25 }, { name: "Bob", age: 30 }, { name: "Charlie", age: 28 }])

POUR LANCER LES INSTANCES :

docker exec -it mongo1 mongosh -u root -p le_bras_sur_la_chaise238
--authenticationDatabase admin

Parties 2 – MongoDB Replica Set

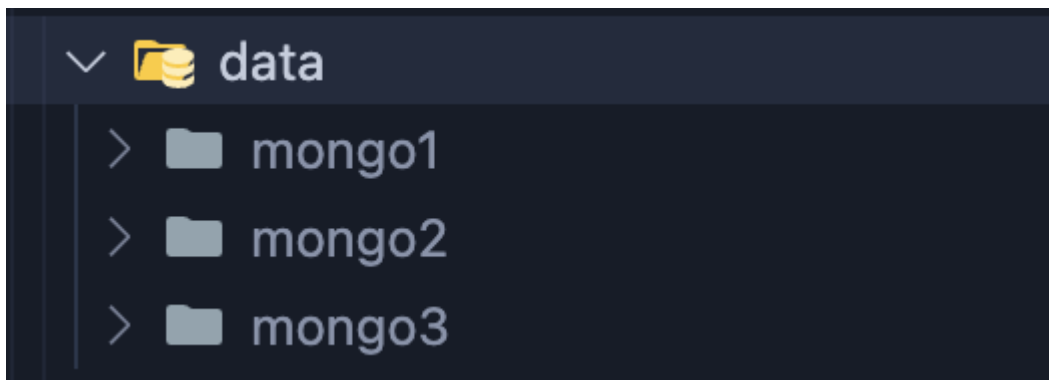
Quand l'authentification est activée (MONGO_INITDB_ROOT_USERNAME + PASSWORD) et que tu actives un replica set, Mongo exige un fichier de clé (keyFile) partagé entre les membres pour qu'ils puissent s'authentifier entre eux.

Afin que la clé fonctionne :

(dans le dossier replicaset)

```
chmod 600 mongo-keyfile
```

Création des répertoires de données pour MongoDB (Car soucis de lancement)



```
mkdir -p data/mongo1 data/mongo2 data/mongo3
```

Connexion à l'instance principale

```
bashdocker exec -it mongo1 mongosh
```

Vérification du lancement des Cluster avec la commande (rs.status)

```
mongo1 | {"t":{"$date":"2025-04-22T13:28:17.012+00:00"},"s":"w", "c":"QUERY",
  "id":23799, "ctx":{"ftdc","msg":"Aggregate command executor error","attr":{"er
  ror":{"code":26,"codeName":"NamespaceNotFound","errmsg":"Unable to retrieve stora
  geStats in $collStats stage :: caused by :: Collection [local.oplog.rs] not found
  ."},"stats":{"cmd":{"aggregate":"oplog.rs","cursor":{"pipeline":{"$collStats
  ":"$storageStats":{"waitForLock":false,"numericOnly":true}}},"$db":"local"}}}
  mongo2 | {"t":{"$date":"2025-04-22T13:28:18.005+00:00"},"s":"w", "c":"QUERY",
  "id":23799, "ctx":{"ftdc","msg":"Aggregate command executor error","attr":{"er
  ror":{"code":26,"codeName":"NamespaceNotFound","errmsg":"Unable to retrieve stora
  geStats in $collStats stage :: caused by :: Collection [local.oplog.rs] not found
  ."},"stats":{"cmd":{"aggregate":"oplog.rs","cursor":{"pipeline":{"$collStats
  ":"$storageStats":{"waitForLock":false,"numericOnly":true}}},"$db":"local"}}}
  mongo1 | {"t":{"$date":"2025-04-22T13:28:18.006+00:00"},"s":"w", "c":"QUERY",
  "id":23799, "ctx":{"ftdc","msg":"Aggregate command executor error","attr":{"er
  ror":{"code":26,"codeName":"NamespaceNotFound","errmsg":"Unable to retrieve stora
  geStats in $collStats stage :: caused by :: Collection [local.oplog.rs] not found
  ."},"stats":{"cmd":{"aggregate":"oplog.rs","cursor":{"pipeline":{"$collStats
  ":"$storageStats":{"waitForLock":false,"numericOnly":true}}},"$db":"local"}}}
  mongo3 | {"t":{"$date":"2025-04-22T13:28:18.006+00:00"},"s":"w", "c":"QUERY",
  "id":23799, "ctx":{"ftdc","msg":"Aggregate command executor error","attr":{"er
  ror":{"code":26,"codeName":"NamespaceNotFound","errmsg":"Unable to retrieve stora
  geStats in $collStats stage :: caused by :: Collection [local.oplog.rs] not found
  ."},"stats":{"cmd":{"aggregate":"oplog.rs","cursor":{"pipeline":{"$collStats
  ":"$storageStats":{"waitForLock":false,"numericOnly":true}}},"$db":"local"}}}
  }

test> rs.status
{Function: status} AsyncFunction {
  apiVersions: [ 0, 0 ],
  returnsPromise: true,
  serverVersions: [ '0.0.0', '999.999.999' ],
  topologies: [ 'RepSet', 'Sharded', 'LoadBalanced', 'Standalone' ],
  returnType: { type: 'unknown', attributes: {} },
  deprecated: false,
  platforms: [ 'Compass', 'Browser', 'CLI' ],
  isDirectShellCommand: false,
  acceptsRawInput: false,
  shellCommandCompleter: undefined,
  help: [Function (anonymous)] Help
}
test>
```

Lancement des instances avec Master et Slaves :

```
rs.initiate({
  _id: "rs0",
  members: [
    { _id: 0, host: "mongo1:27017" },
    { _id: 1, host: "mongo2:27017" },
    { _id: 2, host: "mongo3:27017" }
  ]
})
```

Vérification avec la commande rs.status :

```
{
  _id: 2,
  name: 'mongo3:27017',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 3,
  optime: { ts: Timestamp({ t: 1745328569, i: 2 }), t: Long('1') },
  optimeDurable: { ts: Timestamp({ t: 1745328569, i: 2 }), t: Long('1') },
  optimeDate: ISODate('2025-04-22T13:29:29.000Z'),
  optimeDurableDate: ISODate('2025-04-22T13:29:29.000Z'),
  lastAppliedWallTime: ISODate('2025-04-22T13:29:31.612Z'),
  lastDurableWallTime: ISODate('2025-04-22T13:29:31.612Z'),
  lastHeartbeat: ISODate('2025-04-22T13:29:31.622Z'),
  lastHeartbeatRecv: ISODate('2025-04-22T13:29:32.124Z'),
  pingMs: Long('0'),
  lastHeartbeatMessage: '',
  syncSourceHost: '',
  syncSourceId: -1,
  infoMessage: '',
  configVersion: 4,
  configTerm: 1
}
```

Afin d'insérer des données nous utiliserons la commande :

```
db.test.insertOne({name: "test document", value: 42})
```

Si le Master n'est pas disponible alors essayer :

```
db.getMongo().setReadPref("primaryPreferred")
```

Vérification de l'insertion (en lisant) :

```
rs0 [direct: secondary] testdb> db.test.find()
...
...
[
  {
    _id: ObjectId('6807a19c66cc956ee265d0fb'),
    name: 'test document',
    value: 42
  }
]
rs0 [direct: secondary] testdb> |
```

Connexion au slave 2 :

`docker exec -it mongo2 mongosh`

```
rs0 [direct: primary] testdb> db.test.insertOne({name: "test document", value:
42})
...
...
{
  acknowledged: true,
  insertedId: ObjectId('68079b52d4f4486b3565d0fc')
}
rs0 [direct: primary] testdb> |
```


Test de connexion pour test URI : (Un URI (Uniform Resource Identifier) est une chaîne de caractères qui identifie de manière unique une ressource, comme une page web, un fichier ou une adresse e-mail.)

```
docker exec -it mongo1 mongosh
```

```
"mongodb://mongo1:27017,mongo2:27017,mongo3:27017/?replicaSet=rs0&readPreference=secondary"
```

```
10055 ~ docker exec -it mongo2 mongosh
→ ~ docker exec -it mongo1 mongosh "mongodb://mongo1:27017,mongo2:27017,mongo3:27017/?replicaSet=rs0"

Current Mongosh Log ID: 6807a27f4e8df57c3465d0fa
Connecting to:      mongodb://mongo1:27017,mongo2:27017,mongo3:27017/?replicaSet=rs0&appName=mongosh+2.5.0
Using MongoDB:      7.0.19
Using Mongosh:      2.5.0

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

-----
  The server generated these startup warnings when booting
  2025-04-22T13:56:59.842+00:00: Access control is not enabled for the database
  . Read and write access to data and configuration is unrestricted
  2025-04-22T13:56:59.842+00:00: For customers running MongoDB 7.0, we suggest
  changing the contents of the following sysfsFile
  2025-04-22T13:56:59.842+00:00: vm.max_map_count is too low
  -----

rs0 [primary] test> |
```

Notabene :

Modes d'écriture PRIMARY uniquement: Toutes les opérations d'écriture (insert, update, delete) sont dirigées exclusivement vers le nœud PRIMARY du replica set. Lorsqu'une écriture est tentée sur un nœud SECONDARY, elle échoue avec l'erreur "NotWritablePrimary: not primary".

Cette restriction garantit la cohérence des données car un seul nœud est responsable des écritures.

Modes de lecture Par défaut: Toutes les lectures sont dirigées vers le PRIMARY.
Lectures sur SECONDARY: Nécessitent une configuration explicite via:
rs.secondaryOk() (obsolète) db.getMongo().setReadPref("secondaryPreferred")
(méthode recommandée)