Getting Started with MongoDB: GUI & Shell

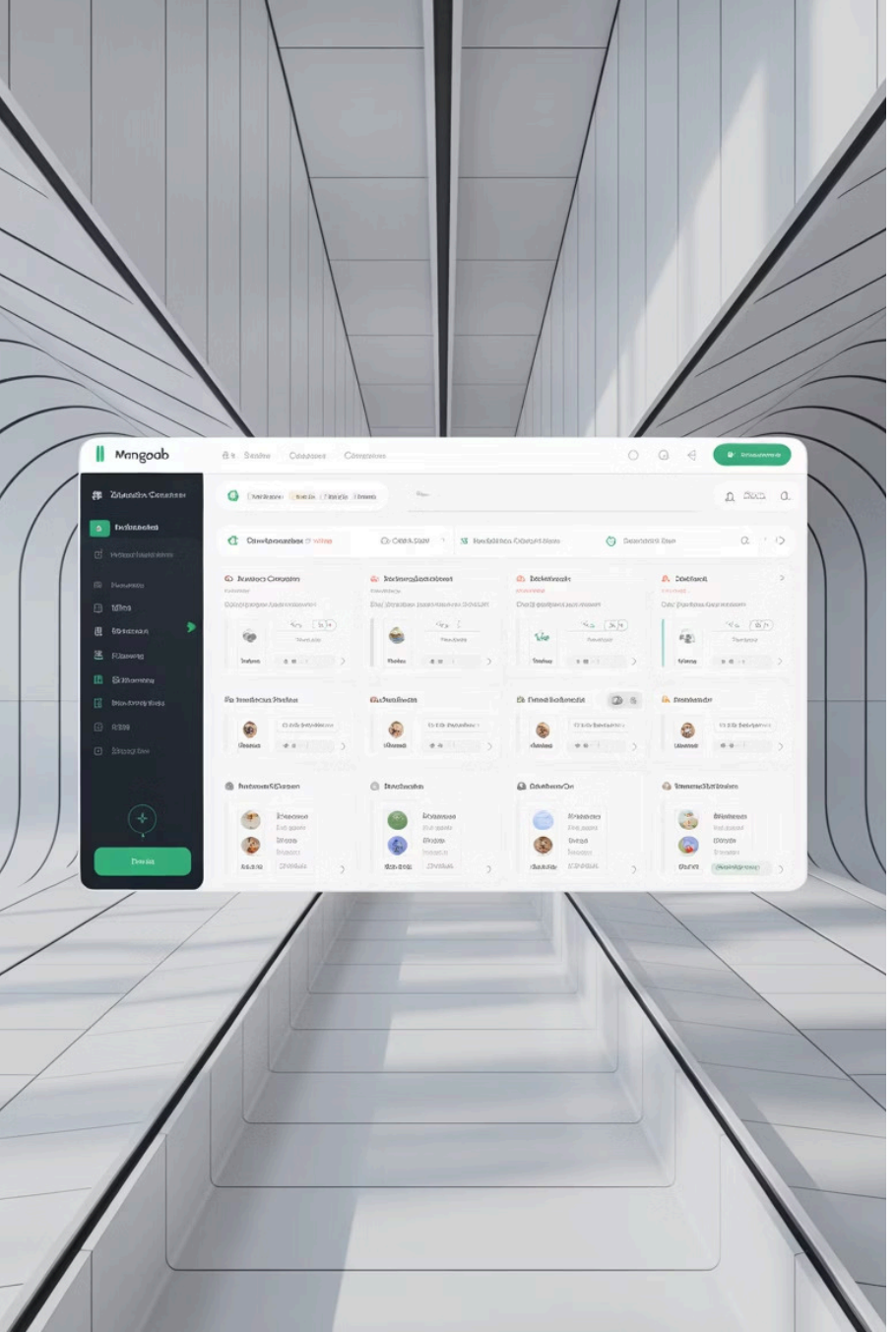
Understanding MongoDB Compass and MongoDB Shell (Mongosh)

What is MongoDB?

MongoDB is a **NoSQL, document-oriented database** that stores data in flexible JSON-like documents, allowing for dynamic schemas and efficient data management. Unlike traditional relational databases, MongoDB doesn't require predefined table structures, making it ideal for modern applications with evolving data requirements.

MongoDB provides two major interfaces for interaction, each designed to serve different user needs and workflows:

- **MongoDB Compass (GUI)** – Visual interface for beginners
- **MongoDB Shell (CLI)** – Command-line tool for developers



MongoDB Interfaces Overview

MongoDB Compass

Interface Type: GUI (Graphical User Interface)

Ideal For: Beginners, data visualization, quick exploration

An intuitive interface to explore, analyse, and modify MongoDB data using forms, charts, and visual schema representations.

MongoDB Shell (mongosh)

Interface Type: CLI (Command Line Interface)

Ideal For: Developers, scripting, automation

A powerful command-line tool for executing queries, managing databases, and automating operations through JavaScript-based commands.

Installing MongoDB Tools

MongoDB Compass Installation

Download Compass from the official MongoDB website at **mongodb.com/try/download/compass.** The installation process is straightforward and typically takes just a few minutes.

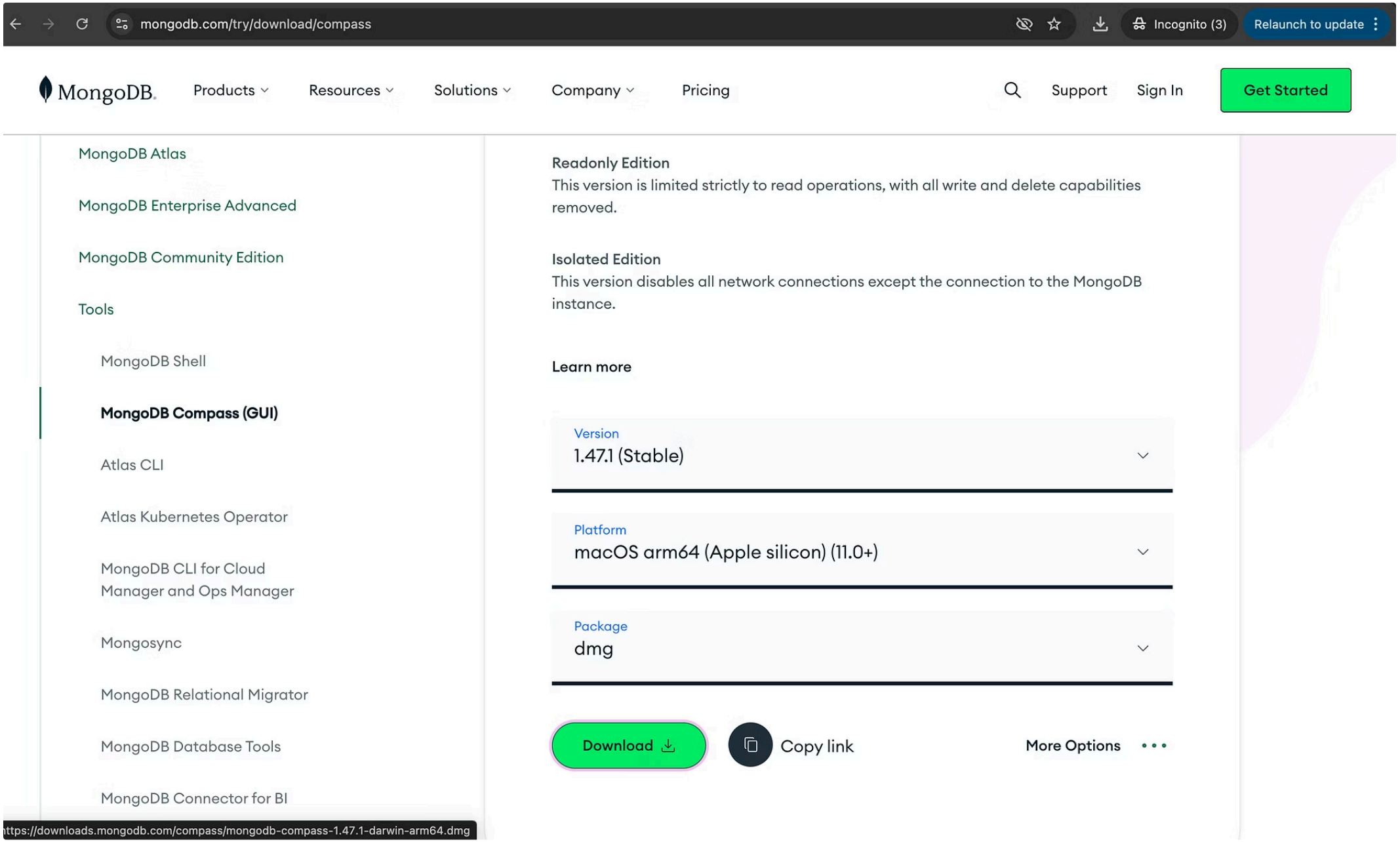
Choose from three editions:

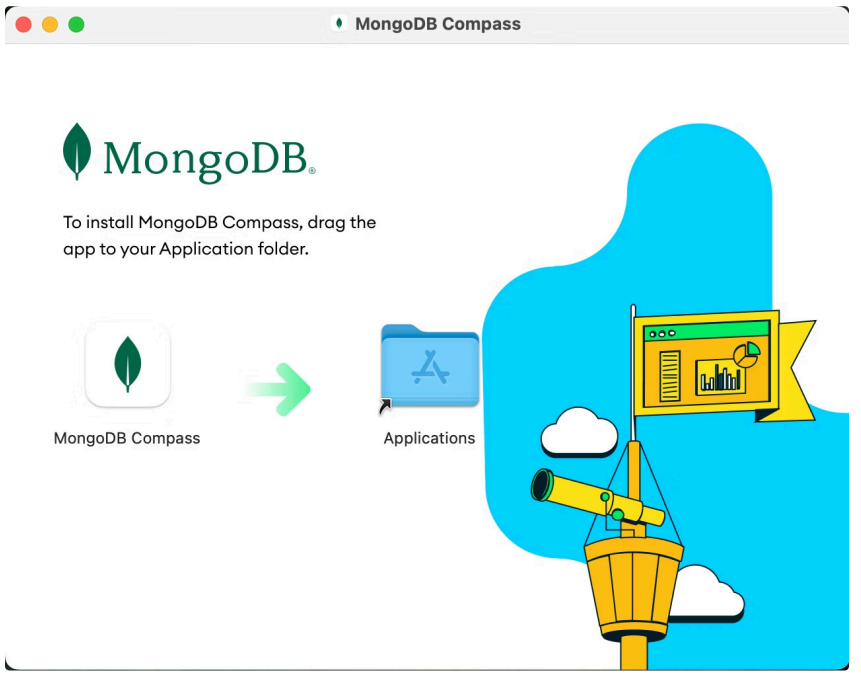
- **Full** – Complete feature set for development
- **Read-only** – Safe viewing without modification rights
- **Isolated** – Offline connections only for security

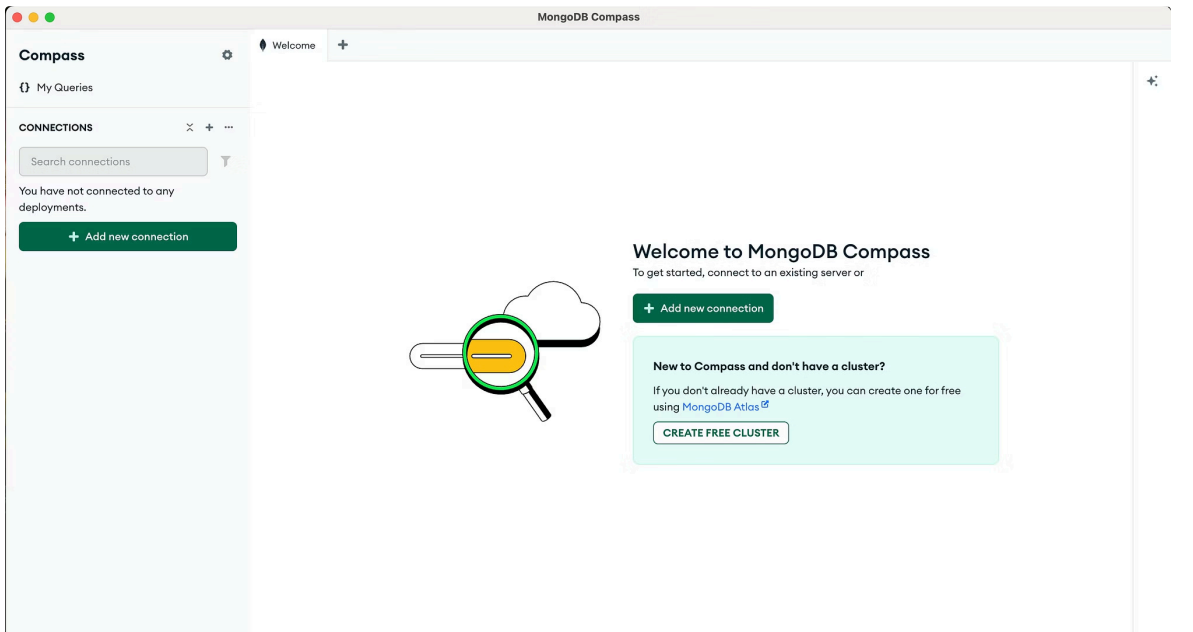
MongoDB Shell Installation

The MongoDB Shell (mongosh) is installed by default with MongoDB distributions, so if you've already installed MongoDB, you likely have it available.

Alternatively, you can download mongosh separately from the MongoDB website if you only need the shell interface without the full database installation.


mongodb.com/try/download/compass
Incognito (3)
Relaunch to update
MongoDB
Products
Resources
Solutions
Company
Pricing
Support
Sign In
Get Started
MongoDB Atlas
MongoDB Enterprise Advanced
MongoDB Community Edition
Tools
MongoDB Shell
MongoDB Compass (GUI)
Atlas CLI
Atlas Kubernetes Operator
MongoDB CLI for Cloud Manager and Ops Manager
Mongosync
MongoDB Relational Migrator
MongoDB Database Tools
MongoDB Connector for BI
Readonly Edition
This version is limited strictly to read operations, with all write and delete capabilities removed.
Isolated Edition
This version disables all network connections except the connection to the MongoDB instance.
Learn more
Version
1.47.1 (Stable)
Platform
macOS arm64 (Apple silicon) (11.0+)
Package
dmg
Download
Copy link
More Options
https://downloads.mongodb.com/compass/mongodb-compass-1.47.1-darwin-arm64.dmg


MongoDB Compass
MongoDB.
To install MongoDB Compass, drag the app to your Application folder.
MongoDB Compass
Applications


MongoDB Compass
Compass
My Queries
CONNECTIONS
Search connections
You have not connected to any deployments.
Add new connection
Welcome
Welcome to MongoDB Compass
To get started, connect to an existing server or
Add new connection
New to Compass and don't have a cluster?
If you don't already have a cluster, you can create one for free using MongoDB Atlas
CREATE FREE CLUSTER

Connecting to MongoDB

Connecting via Compass

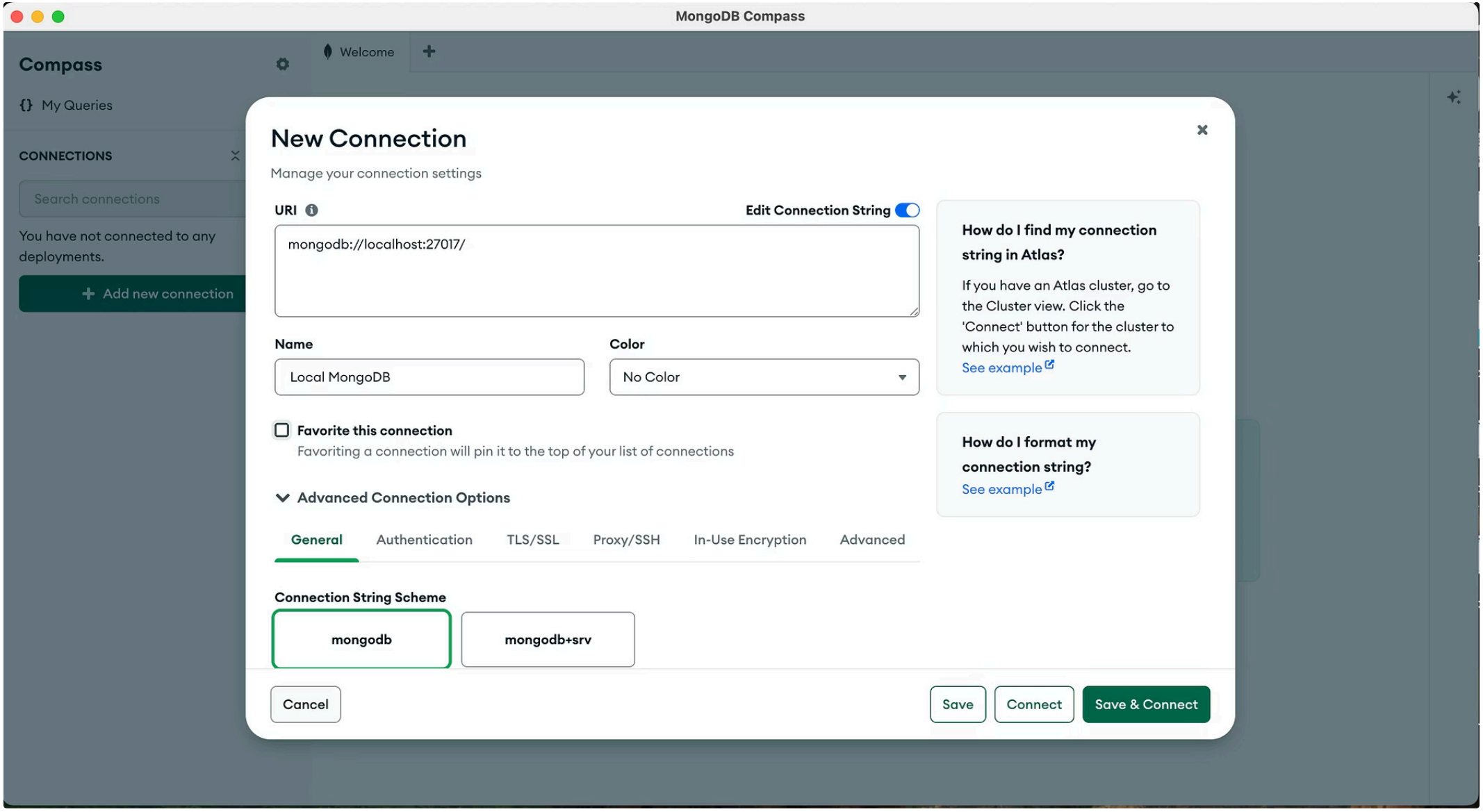
1. Click "New Connection" in the welcome screen
2. Enter your connection URI (e.g., mongodb://localhost:27017)
3. Optionally configure authentication credentials or TLS settings
4. Click "Connect" to establish the connection

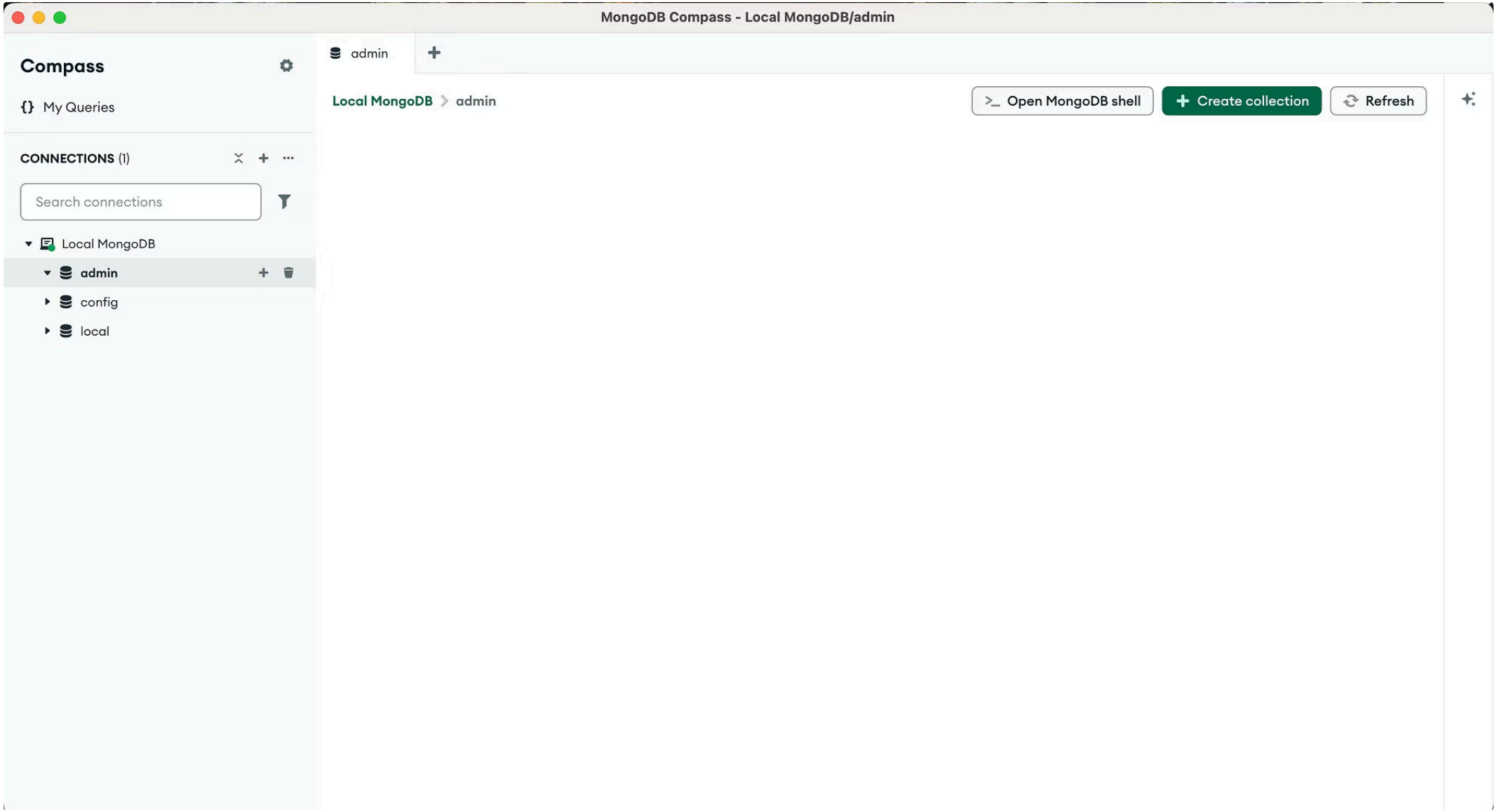
Connecting via Shell

Open your terminal and run the following command:

```
mongosh "mongodb://localhost:27017"
```

The shell will display version information, server build details, and connection status upon successful connection.


MongoDB Compass
Compass
My Queries
CONNECTIONS
Search connections
You have not connected to any deployments.
Add new connection
Welcome
New Connection
Manage your connection settings
URI
Edit Connection String
mongodb://localhost:27017/
Name
Local MongoDB
Color
No Color
Favorite this connection
Favoriting a connection will pin it to the top of your list of connections
Advanced Connection Options
General
Authentication
TLS/SSL
Proxy/SSH
In-Use Encryption
Advanced
Connection String Scheme
mongodb
mongodb+srv
How do I find my connection string in Atlas?
If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.
See example
How do I format my connection string?
See example
Cancel
Save
Connect
Save & Connect


MongoDB Compass - Local MongoDB/admin
Compass
My Queries
CONNECTIONS (1)
Search connections
Local MongoDB
admin
config
local
admin
Local MongoDB
admin
Open MongoDB shell
Create collection
Refresh

MongoDB Shell Fundamentals

The MongoDB Shell is a powerful JavaScript-based interactive environment that allows you to interact with your MongoDB databases through code. Understanding its foundations is essential for advanced database operations.

JavaScript Engine

Built on SpiderMonkey JS engine, providing robust JavaScript execution capabilities directly within the database shell.

Modern JavaScript Support

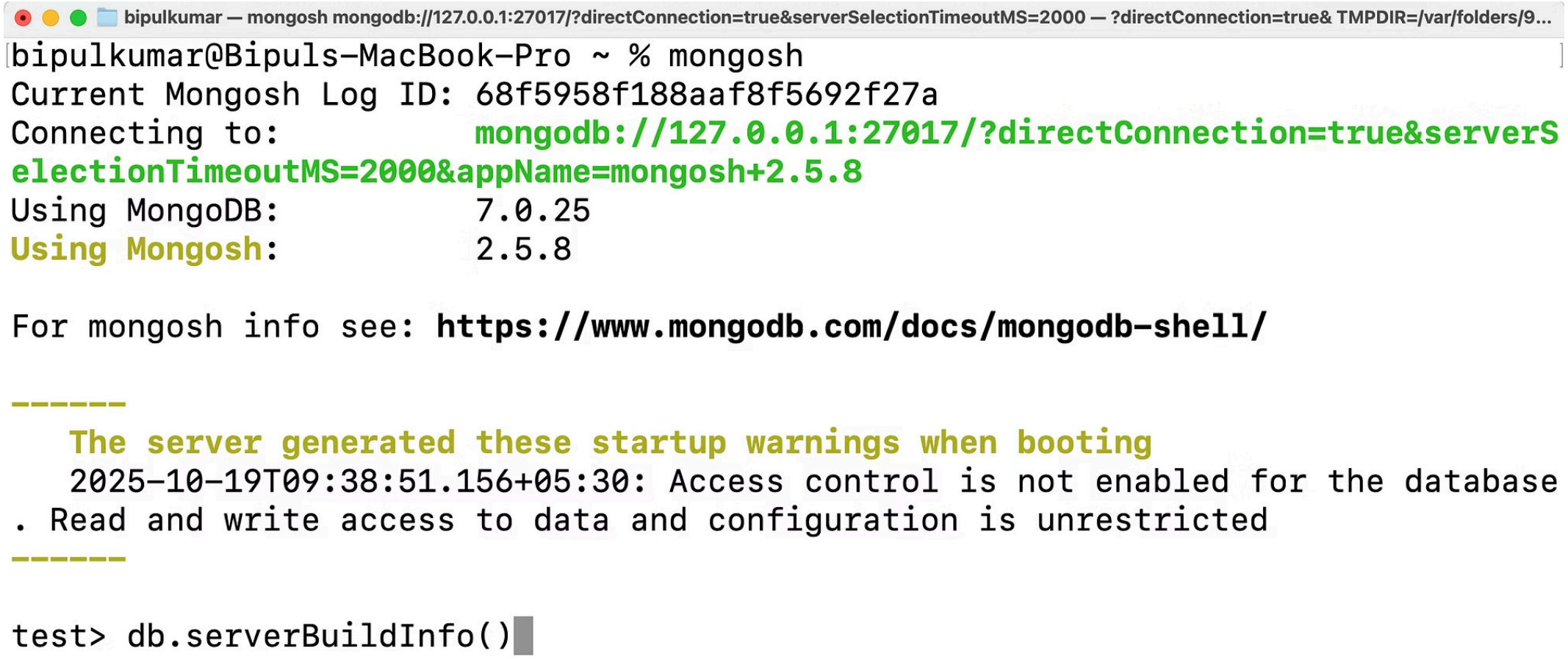
Supports ECMAScript 6 (ES6) features since MongoDB version 3.2, including arrow functions, destructuring, and template literals.

Dual Execution

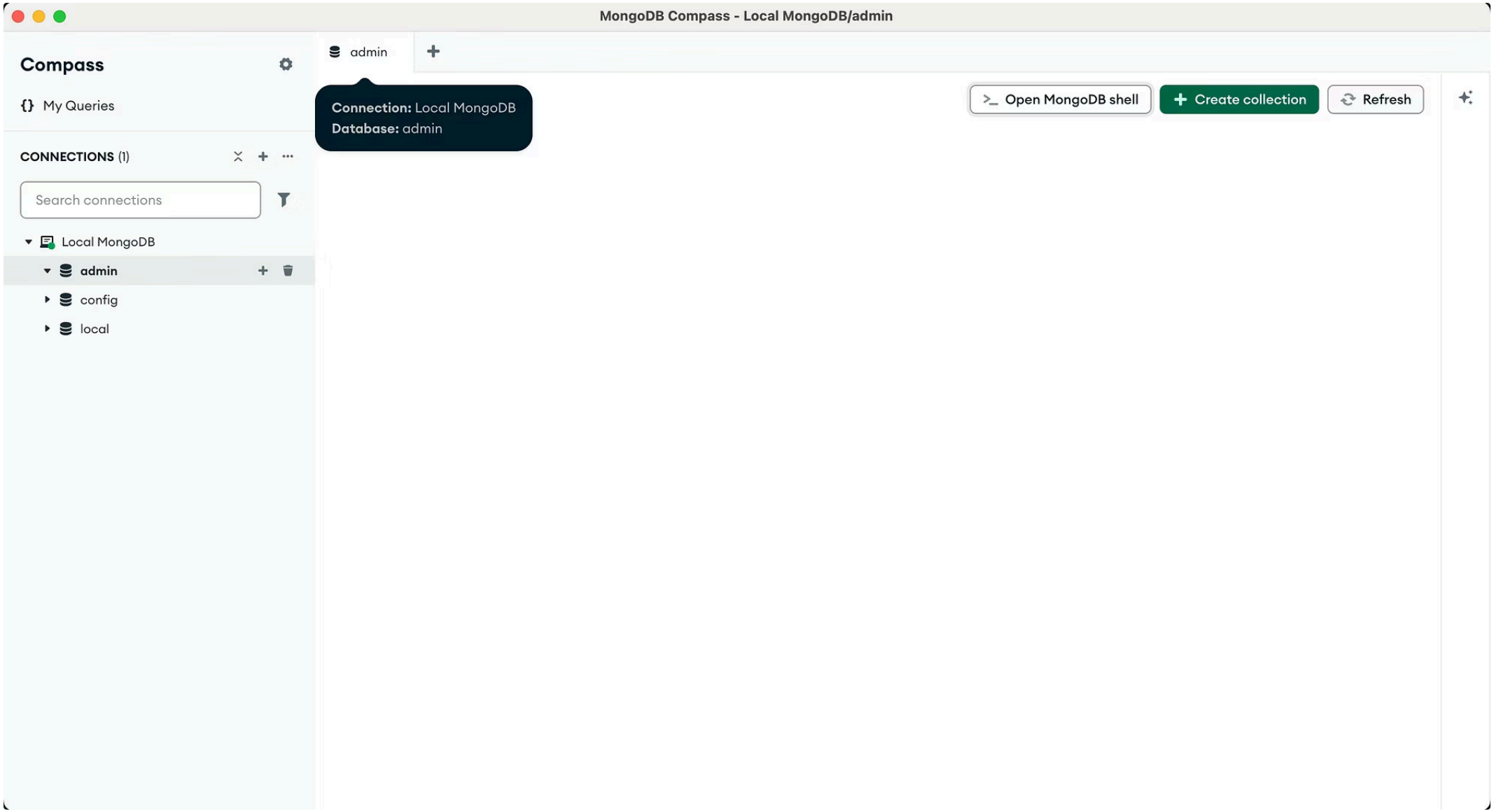
Allows both client-side and server-side JavaScript execution, enabling flexible data processing and manipulation.

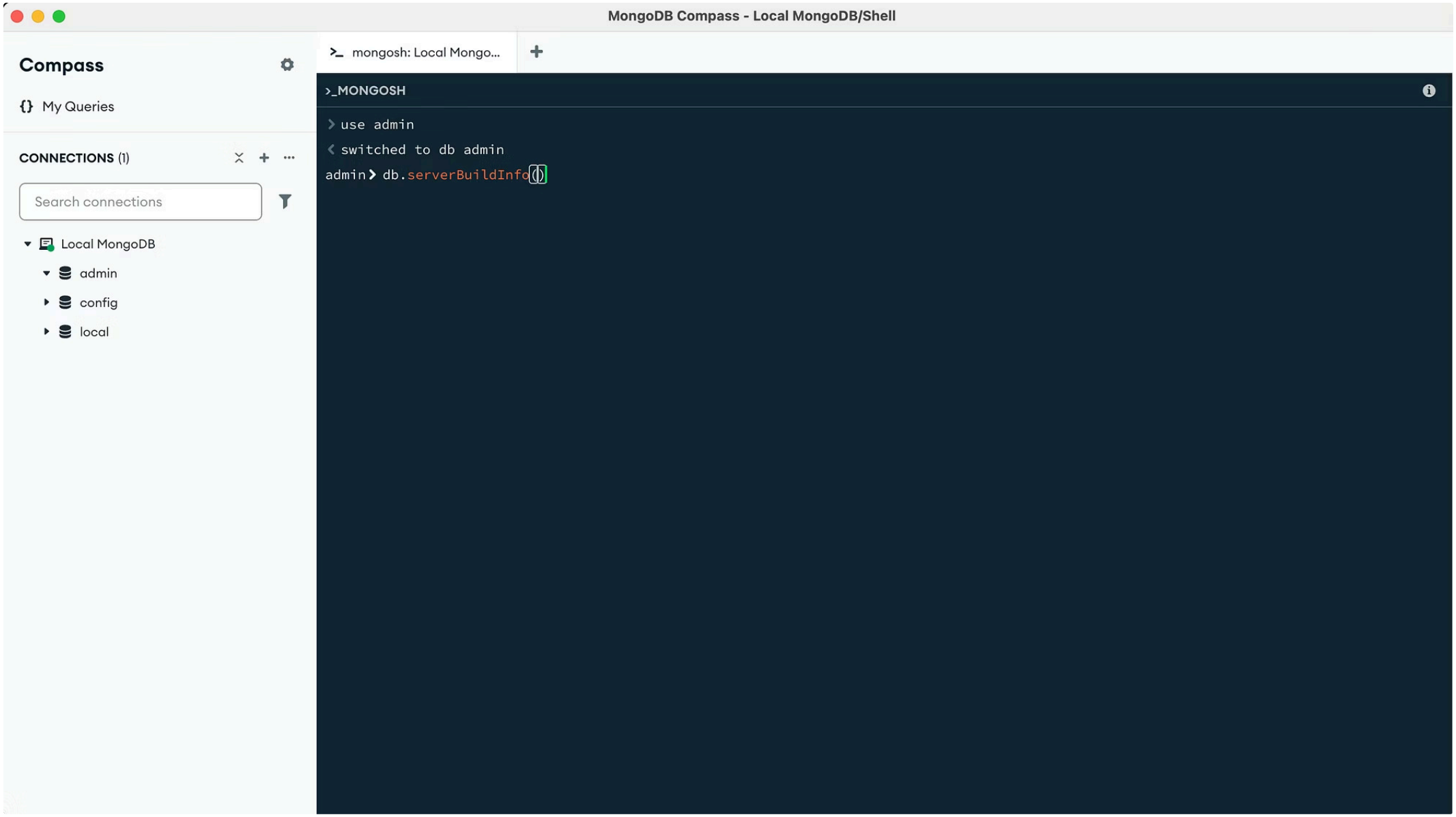
Essential Shell Commands

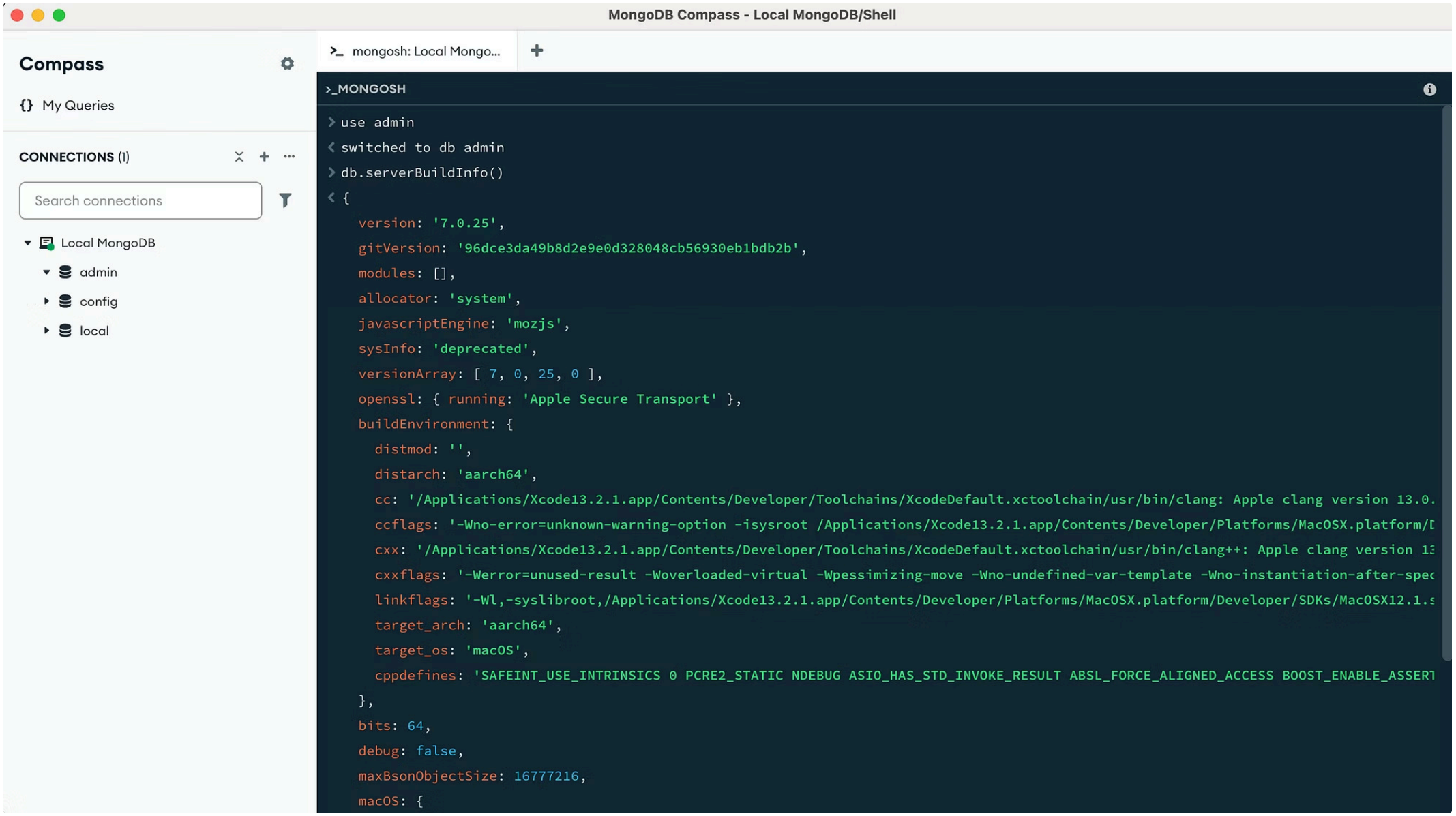
- `db.version()` – Get database version
- `db.serverBuildInfo()` – Get detailed server information
- `db.help()` – List all available commands


bipulkumar — mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — ?directConnection=true& TMPDIR=/var/folders/9...
bipulkumar@Bipuls-MacBook-Pro ~ % mongosh
Current Mongosh Log ID: 68f5958f188aaf8f5692f27a
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB: 7.0.25
Using Mongosh: 2.5.8
For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
The server generated these startup warnings when booting
2025-10-19T09:38:51.156+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
test> db.serverBuildInfo()

```
bipulkumar@Bipuls-MacBook-Pro ~ % mongosh
Current Mongosh Log ID: 68f5958f188aaf8f5692f27a
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB:          7.0.25
Using Mongosh:          2.5.8

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

------
   The server generated these startup warnings when booting
   2025-10-19T09:38:51.156+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> db.serverBuildInfo()
{
  version: '7.0.25',
  gitVersion: '96dce3da49b8d2e9e0d328048cb56930eb1bdb2b',
  modules: [],
  allocator: 'system',
  javascriptEngine: 'mozjs',
  sysInfo: 'deprecated',
  versionArray: [ 7, 0, 25, 0 ],
  openssl: { running: 'Apple Secure Transport' },
  buildEnvironment: {
    distmod: '',
    distarch: 'aarch64',
    cc: '/Applications/Xcode13.2.1.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang: Apple clang version 13.0.0 (clang-1300.0.29.30)',
    ccflags: '-Wno-error=unknown-warning-option -isysroot /Applications/Xcode13.2.1.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX12.1.sdk -mmacosx-version-min=11.
0 --target=darwin20.0.0 -arch arm64 -Werror -include mongo/platform/basic.h -ffp-contract=off -fasynchronous-unwind-tables -g2 -Wall -Wsign-compare -Wno-unknown-pragmas -Winvalid-pch -fno
-omit-frame-pointer -fno-strict-aliasing -O2 -Wno-unused-local-typedefs -Wno-unused-function -Wno-unused-private-field -Wno-deprecated-declarations -Wno-tautological-constant-out-of-range
-compare -Wno-tautological-constant-compare -Wno-tautological-unsigned-zero-compare -Wno-tautological-unsigned-enum-zero-compare -Wno-unused-const-variable -Wno-missing-braces -Wno-incons
istent-missing-override -Wno-potentially-evaluated-expression -Wno-unused-lambda-capture -Wunguarded-availability -fstack-protector-strong -fno-limit-debug-info -Wimplicit-fallthrough',
    cxx: '/Applications/Xcode13.2.1.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang++: Apple clang version 13.0.0 (clang-1300.0.29.30)',
    cxxflags: '-Werror=unused-result -Woverloaded-virtual -Wpessimizing-move -Wno-undefined-var-template -Wno-instantiation-after-specialization -fsized-deallocation -Wno-defaulted-functi
on-deleted -Wunused-exception-parameter -Wno-deprecated -stdlib=libc++ -std=c++20',
    linkflags: '-Wl,-syslibroot,/Applications/Xcode13.2.1.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX12.1.sdk -mmacosx-version-min=11.0 --target=darwin20.0.0 -a
rch arm64 -Wl,-fatal_warnings -Wl,-bind_at_load -fstack-protector-strong -stdlib=libc++',
    target_arch: 'aarch64',
    target_os: 'macOS',
    cppdefines: 'SAFEINT_USE_INTRINSICS 0 PCRE2_STATIC NDEBUG ASIO_HAS_STD_INVOKE_RESULT ABSL_FORCE_ALIGNED_ACCESS BOOST_ENABLE_ASSERT_DEBUG_HANDLER BOOST_FILESYSTEM_NO_CXX20_ATOMIC_REF B
OOST_LOG_NO_SHORTHAND_NAMES BOOST_LOG_USE_NATIVE_SYSLOG BOOST_LOG_WITHOUT_THREAD_ATTR BOOST_MATH_NO_LONG_DOUBLE_MATH_FUNCTIONS BOOST_SYSTEM_NO_DEPRECATED BOOST_THREAD_USES_DATETIME BOOST_
THREAD_VERSION 5'
  },
  bits: 64,
  debug: false,
  maxBsonObjectSize: 16777216,
  macOS: {
    osProductVersion: '15.6.1',
    osRelease: '24.6.0',
    version: 'Darwin Kernel Version 24.6.0: Mon Jul 14 11:30:40 PDT 2025; root:xnu-11417.140.69~1/RELEASE_ARM64_T8132'
  },
  storageEngines: [ 'devnull', 'wiredTiger' ],
  ok: 1
}
test>
```


MongoDB Compass - Local MongoDB/admin
Compass
My Queries
CONNECTIONS (1)
Search connections
Local MongoDB
admin
config
local
admin
Connection: Local MongoDB
Database: admin
Open MongoDB shell
Create collection
Refresh

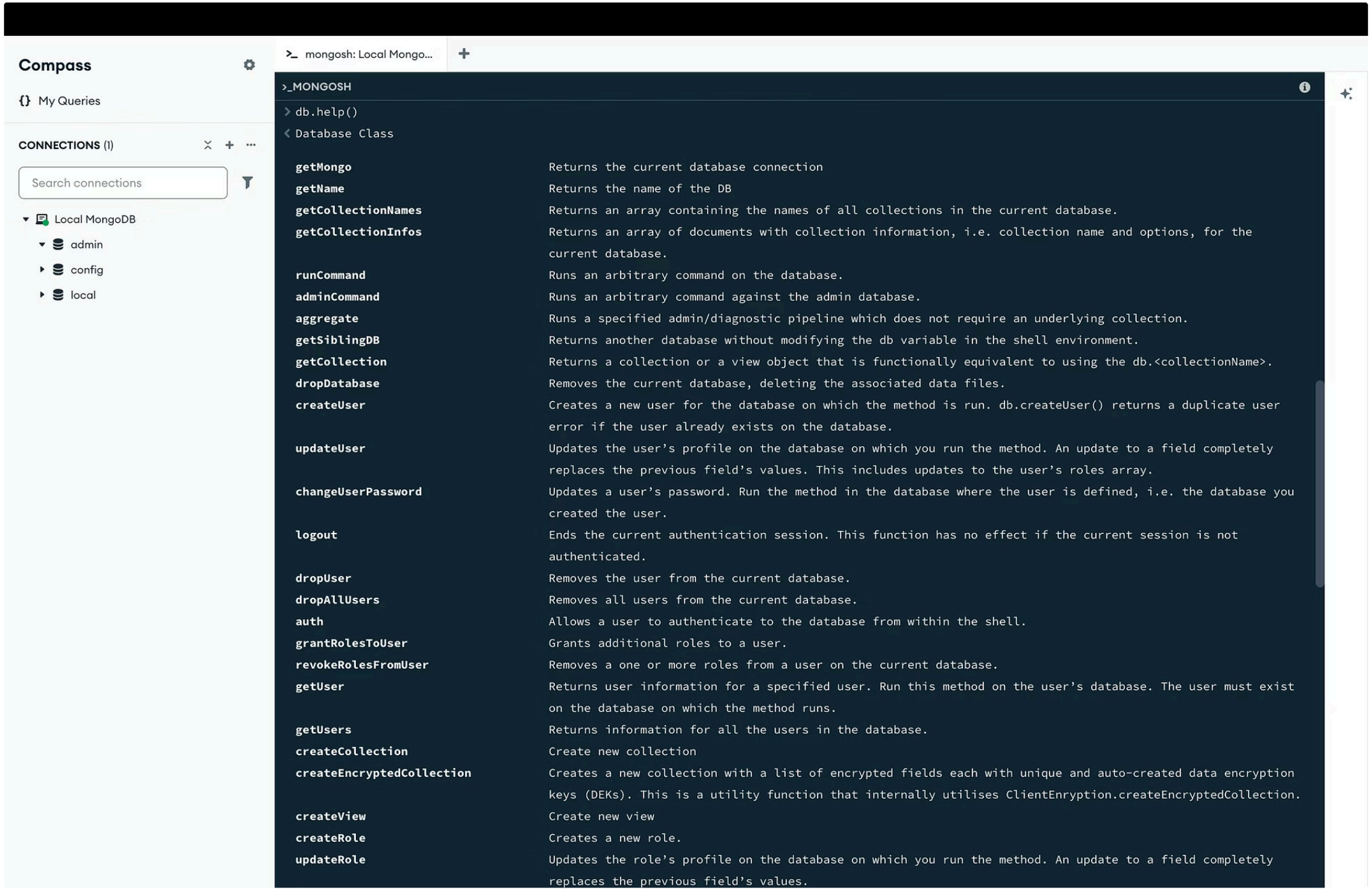

MongoDB Compass - Local MongoDB/Shell
Compass
My Queries
CONNECTIONS (1)
Search connections
Local MongoDB
admin
config
local
mongosh: Local Mongo...
>_MONGOSH
use admin
switched to db admin
admin> db.serverBuildInfo()

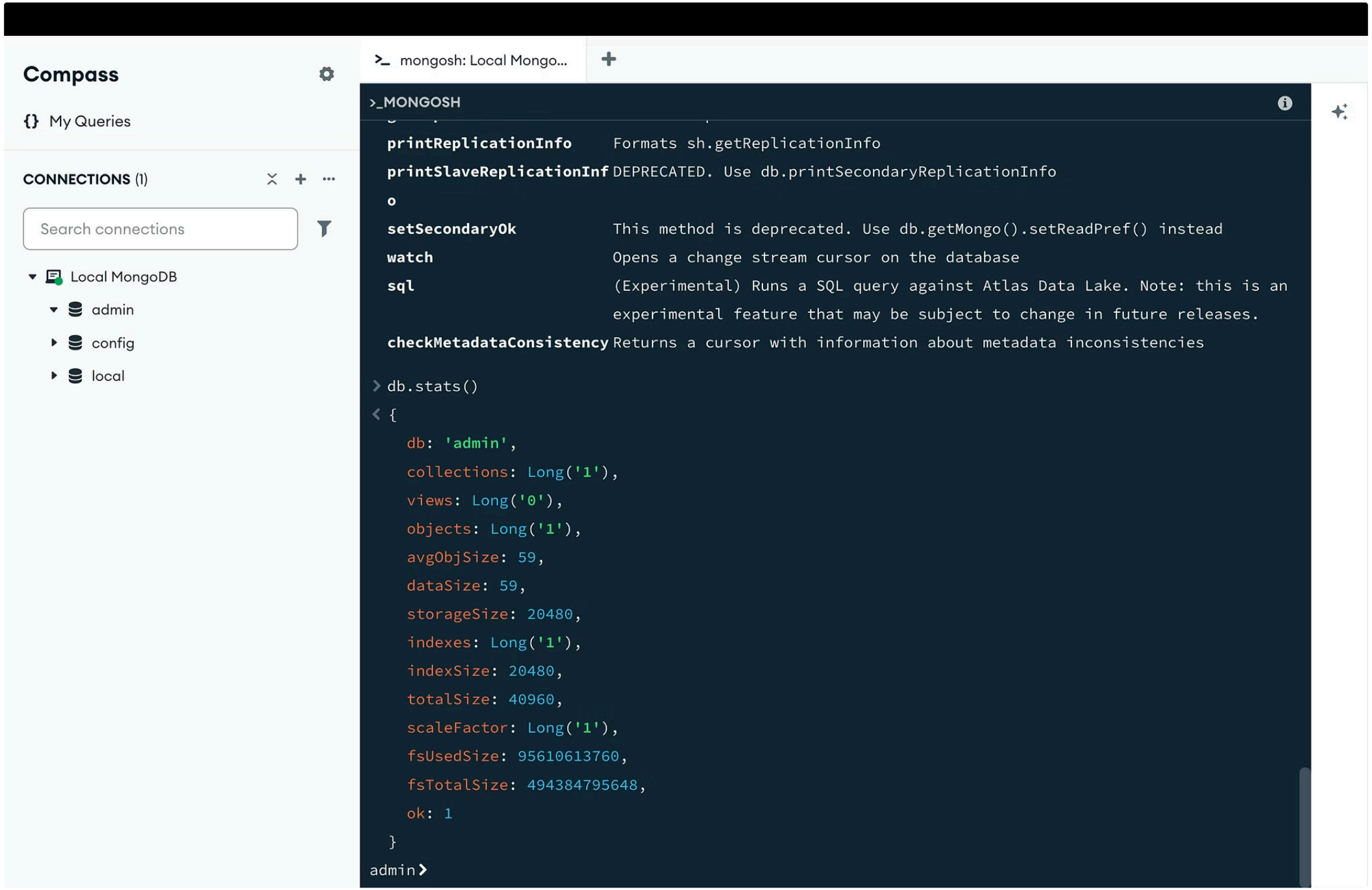

MongoDB Compass - Local MongoDB/Shell
Compass
My Queries
CONNECTIONS (1)
Search connections
Local MongoDB
admin
config
local
mongosh: Local Mongo...
>_MONGOSH
> use admin
< switched to db admin
> db.serverBuildInfo()
< {
version: '7.0.25',
gitVersion: '96dce3da49b8d2e9e0d328048cb56930eb1bdb2b',
modules: [],
allocator: 'system',
javascriptEngine: 'mozjs',
sysInfo: 'deprecated',
versionArray: [7, 0, 25, 0],
openssl: { running: 'Apple Secure Transport' },
buildEnvironment: {
distmod: '',
distarch: 'aarch64',
cc: '/Applications/Xcode13.2.1.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang: Apple clang version 13.0.
ccflags: '-Wno-error=unknown-warning-option -isysroot /Applications/Xcode13.2.1.app/Contents/Developer/Platforms/MacOSX.platform/D
cxx: '/Applications/Xcode13.2.1.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang++: Apple clang version 13
cxxflags: '-Werror=unused-result -Woverloaded-virtual -Wpessimizing-move -Wno-undefined-var-template -Wno-instantiation-after-spec
linkflags: '-Wl,-syslibroot,/Applications/Xcode13.2.1.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX12.1.s
target_arch: 'aarch64',
target_os: 'macOS',
cppdefines: 'SAFEINT_USE_INTRINSICS 0 PCRE2_STATIC NDEBUG ASIO_HAS_STD_INVOKE_RESULT ABSL_FORCE_ALIGNED_ACCESS BOOST_ENABLE_ASSERT
},
bits: 64,
debug: false,
maxBsonObjectSize: 16777216,
macOS: {

```
bipulkumar@Bipuls-MacBook-Pro ~ % mongosh
Current Mongosh Log ID: 68f599731236391191dad6e81
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&se
rverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB:          7.0.25
Using Mongosh:          2.5.8

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

------
   The server generated these startup warnings when booting
   2025-10-19T09:38:51.156+05:30: Access control is not enabled for the dat
abase. Read and write access to data and configuration is unrestricted
------

test> db.version()
7.0.25
test> version()
2.5.8
test>
```

```
test> help
[
  Shell Help:

    log                                 'log.info(<msg>)': Write a custom info/warn/error/fatal/debug message to the log file
                                        'log.getPath()': Gets a path to the current log file

    use                                 Set current database
    show                                'show databases'/'show dbs': Print a list of all available databases
                                        'show collections'/'show tables': Print a list of all collections for current database
                                        'show profile': Prints system.profile information
                                        'show users': Print a list of all users for current database
                                        'show roles': Print a list of all roles for current database
                                        'show log <name>': Display log for current connection, if name is not set uses 'global'
                                        'show logs': Print all logger names.
    exit                                Quit the MongoDB shell with exit/exit()/.exit
    quit                                Quit the MongoDB shell with quit/quit()
    Mongo                               Create a new connection and return the Mongo object. Usage: new Mongo(URI, options [optio
nal])
    connect                             Create a new connection and return the Database object. Usage: connect(URI, username [opt
ional], password [optional])
    it                                  result of the last line evaluated; use to further iterate
    version                             Shell version
    load                                Loads and runs a JavaScript file into the current shell environment
    enableTelemetry                     Enables collection of anonymous usage data to improve the mongosh CLI
    disableTelemetry                    Disables collection of anonymous usage data to improve the mongosh CLI
    passwordPrompt                      Prompts the user for a password
    sleep                               Sleep for the specified number of milliseconds
    print                               Prints the contents of an object to the output
    printjson                           Alias for print()
    convertShardKeyToHashed             Returns the hashed value for the input using the same hashing function as a hashed index.
    cls                                 Clears the screen like console.clear()
    isInteractive                       Returns whether the shell will enter or has entered interactive mode

  For more information on usage: https://mongodb.com/docs/manual/reference/method
]
test>
[
test>
]

test>
```

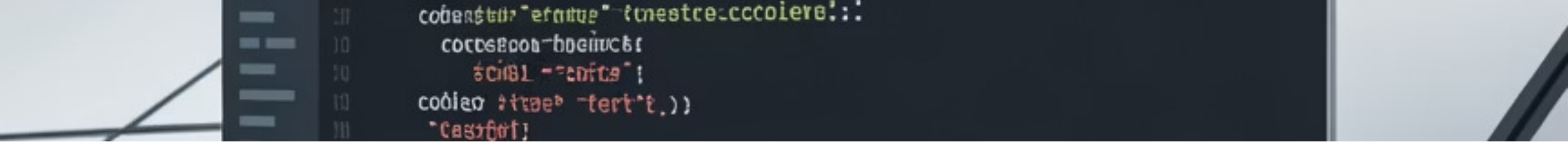
```
test>  db.help()

  Database Class:

    getMongo                                  Returns the current database connection
    getName                                   Returns the name of the DB
    getCollectionNames                        Returns an array containing the names of all collections in the current database.
    getCollectionInfos                        Returns an array of documents with collection information, i.e. collection name and optio
ns, for the current database.
    runCommand                                Runs an arbitrary command on the database.
    adminCommand                              Runs an arbitrary command against the admin database.
    aggregate                                 Runs a specified admin/diagnostic pipeline which does not require an underlying collectio
n.
    getSiblingDB                              Returns another database without modifying the db variable in the shell environment.
    getCollection                             Returns a collection or a view object that is functionally equivalent to using the db.<co
llectionName>.
    dropDatabase                              Removes the current database, deleting the associated data files.
    createUser                                Creates a new user for the database on which the method is run. db.createUser() returns a
 duplicate user error if the user already exists on the database.
    updateUser                                Updates the user's profile on the database on which you run the method. An update to a fi
eld completely replaces the previous field's values. This includes updates to the user's roles array.
    changeUserPassword                        Updates a user's password. Run the method in the database where the user is defined, i.e.
 the database you created the user.
    logout                                    Ends the current authentication session. This function has no effect if the current sessi
on is not authenticated.
    dropUser                                  Removes the user from the current database.
    dropAllUsers                              Removes all users from the current database.
    auth                                      Allows a user to authenticate to the database from within the shell.
    grantRolesToUser                          Grants additional roles to a user.
    revokeRolesFromUser                       Removes a one or more roles from a user on the current database.
    getUser                                   Returns user information for a specified user. Run this method on the user's database. Th
e user must exist on the database on which the method runs.
    getUsers                                  Returns information for all the users in the database.
    createCollection                          Create new collection
    createEncryptedCollection                 Creates a new collection with a list of encrypted fields each with unique and auto-create
d data encryption keys (DEKs). This is a utility function that internally utilises ClientEnryption.createEncryptedCollection.
    createView                                Create new view
    createRole                                Creates a new role.
    updateRole                                Updates the role's profile on the database on which you run the method. An update to a fi
eld completely replaces the previous field's values.
    dropRole                                  Removes the role from the current database.
    dropAllRoles                              Removes all roles from the current database.
```


Compass
My Queries
CONNECTIONS (1)
Search connections
Local MongoDB
admin
config
local
mongosh: Local Mongo...
>_MONGOSH
db.help()
Database Class
getMongo Returns the current database connection
getName Returns the name of the DB
getCollectionNames Returns an array containing the names of all collections in the current database.
getCollectionInfos Returns an array of documents with collection information, i.e. collection name and options, for the current database.
runCommand Runs an arbitrary command on the database.
adminCommand Runs an arbitrary command against the admin database.
aggregate Runs a specified admin/diagnostic pipeline which does not require an underlying collection.
getSiblingDB Returns another database without modifying the db variable in the shell environment.
getCollection Returns a collection or a view object that is functionally equivalent to using the db.<collectionName>.
dropDatabase Removes the current database, deleting the associated data files.
createUser Creates a new user for the database on which the method is run. db.createUser() returns a duplicate user error if the user already exists on the database.
updateUser Updates the user's profile on the database on which you run the method. An update to a field completely replaces the previous field's values. This includes updates to the user's roles array.
changeUserPassword Updates a user's password. Run the method in the database where the user is defined, i.e. the database you created the user.
logout Ends the current authentication session. This function has no effect if the current session is not authenticated.
dropUser Removes the user from the current database.
dropAllUsers Removes all users from the current database.
auth Allows a user to authenticate to the database from within the shell.
grantRolesToUser Grants additional roles to a user.
revokeRolesFromUser Removes a one or more roles from a user on the current database.
getUser Returns user information for a specified user. Run this method on the user's database. The user must exist on the database on which the method runs.
getUsers Returns information for all the users in the database.
createCollection Create new collection
createEncryptedCollection Creates a new collection with a list of encrypted fields each with unique and auto-created data encryption keys (DEKs). This is a utility function that internally utilises ClientEnryption.createEncryptedCollection.
createView Create new view
createRole Creates a new role.
updateRole Updates the role's profile on the database on which you run the method. An update to a field completely replaces the previous field's values.


Compass
My Queries
CONNECTIONS (1)
Search connections
Local MongoDB
admin
config
local
mongosh: Local Mongo...
>_MONGOSH
printReplicationInfo Formats sh.getReplicationInfo
printSlaveReplicationInf DEPRECATED. Use db.printSecondaryReplicationInfo
o
setSecondaryOk This method is deprecated. Use db.getMongo().setReadPref() instead
watch Opens a change stream cursor on the database
sql (Experimental) Runs a SQL query against Atlas Data Lake. Note: this is an
experimental feature that may be subject to change in future releases.
checkMetadataConsistency Returns a cursor with information about metadata inconsistencies
db.stats()
{
db: 'admin',
collections: Long('1'),
views: Long('0'),
objects: Long('1'),
avgObjSize: 59,
dataSize: 59,
storageSize: 20480,
indexes: Long('1'),
indexSize: 20480,
totalSize: 40960,
scaleFactor: Long('1'),
fsUsedSize: 95610613760,
fsTotalSize: 494384795648,
ok: 1
}
admin>

```
bipulkumar@Bipuls-MacBook-Pro ~ % mongosh
Current Mongosh Log ID: 68f59bc0839b44582bcb4ef1
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB:          7.0.25
Using Mongosh:          2.5.8

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

------
   The server generated these startup warnings when booting
   2025-10-19T09:38:51.156+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unr
estricted
------

test> typeof db
object
test>
```

```
bipulkumar@Bipuls-MacBook-Pro ~ % mongosh
Current Mongosh Log ID: 68f59bc0839b44582bcb4ef1
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB:          7.0.25
Using Mongosh:          2.5.8

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

------
   The server generated these startup warnings when booting
   2025-10-19T09:38:51.156+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unr
estricted
------

test> typeof db
object
test> db.
db.__proto__                db.constructor              db.hasOwnProperty             db.isPrototypeOf
db.propertyIsEnumerable     db.toLocaleString           db.toString                   db.valueOf
db.getMongo                 db.getName                  db.getCollectionNames         db.getCollectionInfos
db.runCommand               db.adminCommand             db.aggregate                  db.getSiblingDB
db.getCollection            db.dropDatabase             db.createUser                 db.updateUser
db.changeUserPassword       db.logout                   db.dropUser                   db.dropAllUsers
db.auth                     db.grantRolesToUser         db.revokeRolesFromUser        db.getUser
db.getUsers                 db.createCollection         db.createEncryptedCollection  db.createView
db.createRole               db.updateRole               db.dropRole                   db.dropAllRoles
db.grantRolesToRole         db.revokeRolesFromRole      db.grantPrivilegesToRole      db.revokePrivilegesFromRole
db.getRole                  db.getRoles                 db.currentOp                  db.killOp
db.shutdownServer           db.fsyncLock                db.fsyncUnlock                db.version
db.serverBits               db.isMaster                 db.hello                      db.serverBuildInfo
db.serverStatus             db.stats                    db.hostInfo                   db.serverCmdLineOpts
db.rotateCertificates       db.printCollectionStats     db.getProfilingStatus         db.setProfilingLevel
db.setLogLevel              db.getLogComponents         db.commandHelp                db.listCommands
db.sql

test> db.
```



JavaScript Syntax in Shell

MongoDB Shell expressions are JavaScript objects and methods, making it familiar for developers with JavaScript experience. Here's a practical example of inserting a document:

```
db.getCollection("test").insertOne({ a: 10, b: true })
```

01

Access Collection

`getCollection("test")` retrieves the "test" collection from the current database.

02

Execute Method

`insertOne()` is a method that inserts a single document into the collection.

03

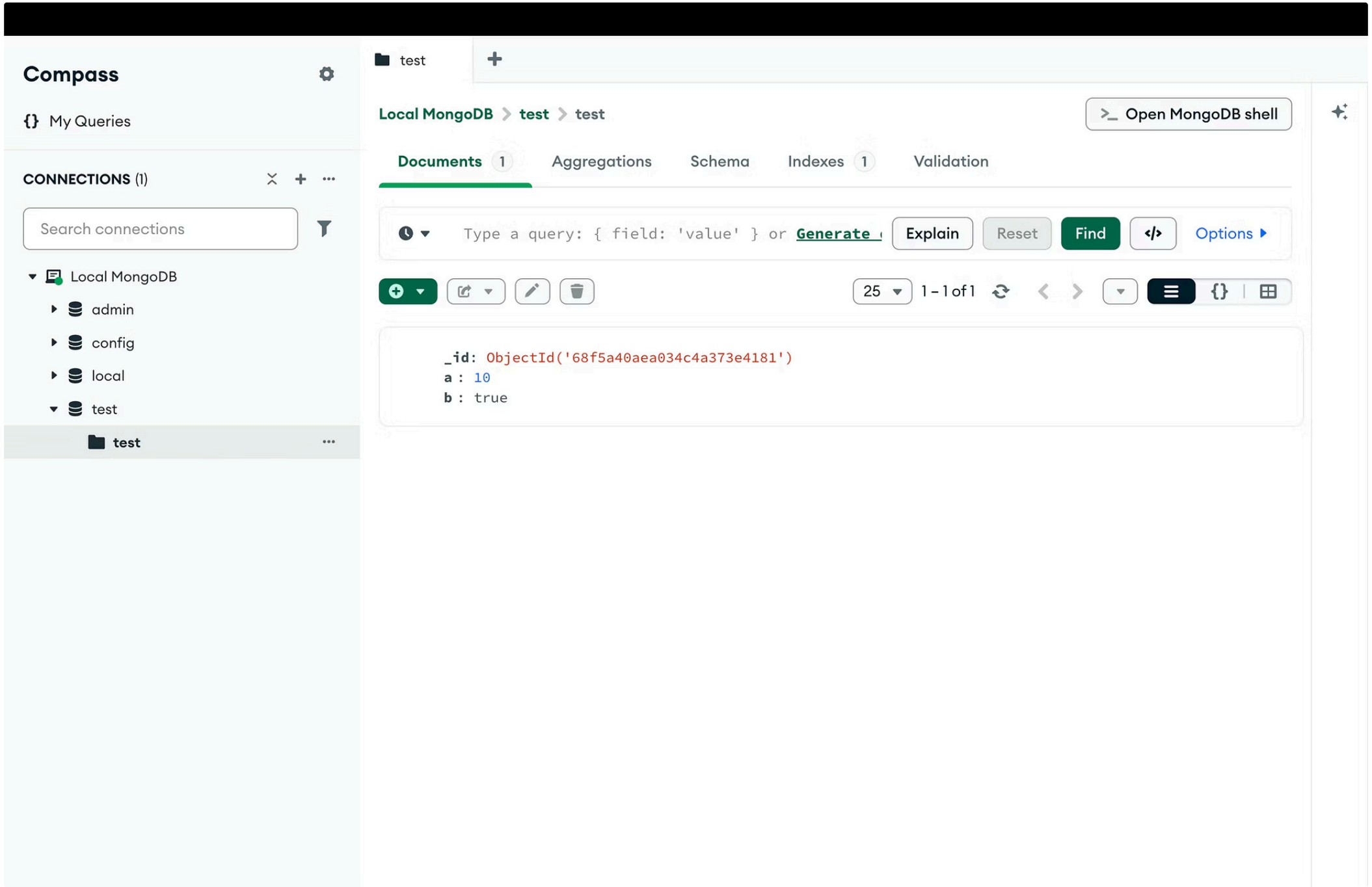
Pass Document

`{ a: 10, b: true }` is the JavaScript object representing the document to insert.

```
test> db.getCollection("test").insertOne({a:10,b:true})
{
  acknowledged: true,
  insertedId: ObjectId('68f5a40aea034c4a373e4181')
}
test> show dbs
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
test      8.00 KiB
test> use test
already on db test
test> show collections
test
```

```
   2025-10-19T09:38:51.156+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unr
estricted
------

test> db.
db.__proto__                  db.constructor                db.hasOwnProperty             db.isPrototypeOf
db.propertyIsEnumerable       db.toLocaleString             db.toString                   db.valueOf
db.getMongo                   db.getName                    db.getCollectionNames         db.getCollectionInfos
db.runCommand                 db.adminCommand               db.aggregate                  db.getSiblingDB
db.getCollection              db.dropDatabase               db.createUser                 db.updateUser
db.changeUserPassword         db.logout                     db.dropUser                   db.dropAllUsers
db.auth                       db.grantRolesToUser           db.revokeRolesFromUser        db.getUser
db.getUsers                   db.createCollection           db.createEncryptedCollection  db.createView
db.createRole                 db.updateRole                 db.dropRole                   db.dropAllRoles
db.grantRolesToRole           db.revokeRolesFromRole        db.grantPrivilegesToRole      db.revokePrivilegesFromRole
db.getRole                    db.getRoles                   db.currentOp                  db.killOp
db.shutdownServer             db.fsyncLock                  db.fsyncUnlock                db.version
db.serverBits                 db.isMaster                   db.hello                      db.serverBuildInfo
db.serverStatus               db.stats                      db.hostInfo                   db.serverCmdLineOpts
db.rotateCertificates         db.printCollectionStats       db.getProfilingStatus         db.setProfilingLevel
db.setLogLevel                db.getLogComponents           db.commandHelp                db.listCommands
db.sql

test> db.getCollection("test").insertOne({a:10,b:true})
{
  acknowledged: true,
  insertedId: ObjectId('68f5a40aea034c4a373e4181')
}
test> show dbs
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
test      8.00 KiB
test> use test
already on db test
test> show collections
test
test> db.collection.find()

test> db.test.find()
[ { _id: ObjectId('68f5a40aea034c4a373e4181'), a: 10, b: true } ]
test>
```

```
test> db.getCollection("test").insertOne({a:10,b:true})
{
  acknowledged: true,
  insertedId: ObjectId('68f5a40aea034c4a373e4181')
}
test> show dbs
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
test      8.00 KiB
test> use test
already on db test
test> show collections
test
test> db.collection.find()

test> db.test.find()
[ { _id: ObjectId('68f5a40aea034c4a373e4181'), a: 10, b: true } ]
```


Compass
My Queries
CONNECTIONS (1)
Search connections
Local MongoDB
admin
config
local
test
test
test
Local MongoDB > test > test
Open MongoDB shell
Documents 1
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate
Explain
Reset
Find
Options
25
1 – 1 of 1
_id: ObjectId('68f5a40aea034c4a373e4181')
a : 10
b : true

Common Shell Commands Reference

Mastering these fundamental commands will enable you to perform essential database operations efficiently. These commands form the foundation of MongoDB database management.

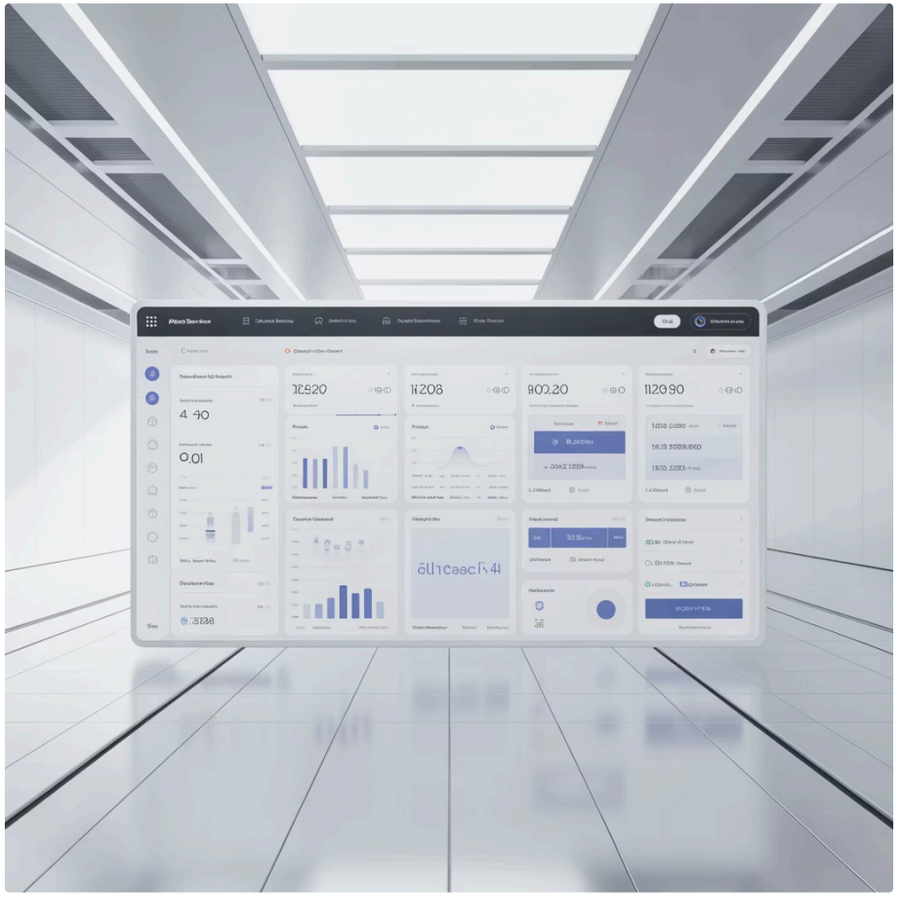
Command	Description
`show dbs`	List all databases in your MongoDB instance
`use <dbname>`	Switch to (or create) a specific database
`show collections`	List all collections in the current database
`db.createCollection("name")`	Create a new collection with the specified name
`db.collection.insertOne()`	Insert a single document into a collection
`db.collection.find()`	Query and retrieve documents from a collection
`db.collection.updateOne()`	Update a single document matching the criteria
`db.collection.deleteOne()`	Delete a single document from the collection
`exit` or `quit()`	Exit the MongoDB shell session

MongoDB Compass Overview

MongoDB Compass is a sophisticated graphical interface that allows you to explore and manipulate your data visually, making database management accessible to users of all skill levels.

Key Features

- **Schema Visualisation** – Understand your data structure at a glance
- **Real-time Performance Monitoring** – Track query performance and database health
- **Easy CRUD Operations** – Create, Read, Update, and Delete with intuitive forms
- **Query Builder** – Build complex queries without writing code



Available Editions

- **Full** – Complete feature set
- **Read-only** – Safe viewing access
- **Isolated** – Offline connections only

CRUD Operations in Compass

Compass makes database operations intuitive through its visual interface. Each operation can be performed with just a few clicks, making it perfect for beginners and rapid prototyping.

Create

Add documents easily via visual editors. Simply fill in forms or paste JSON directly. Compass validates your data and provides helpful syntax highlighting.

Read

View and filter documents in collections using the query builder. Apply filters, sort results, and paginate through large datasets with ease.

Update

Modify data using form inputs or the JSON editor. Changes are validated in real-time, preventing errors before they reach your database.

Delete

Remove selected documents with confirmation prompts. Compass ensures you don't accidentally delete important data with built-in safety checks.

GUI vs Shell Comparison

Feature	Compass (GUI)	Mongosh (Shell)
Skill Level	Beginner-friendly	Intermediate / Advanced
Data Visualisation	Available (Charts, Schema View)	Not available
Automation/Scripting	Limited capabilities	Fully supported
Ease of Use	Click-based interface	Command-based interface
Learning Curve	Gentle and gradual	Steeper, requires syntax knowledge
Example Action	Click "Add Document" button	`db.collection.insertOne()`

Both tools complement each other perfectly. Start with Compass to understand your data visually, then transition to Shell for automation and advanced operations.

When to Use Each Tool

Use Compass When...

- You're just starting with MongoDB
- You need to explore unfamiliar data structures
- You want to visualise schema patterns
- You're performing one-off data modifications
- You need to present data to non-technical stakeholders

Use Shell When...

- You need to automate repetitive tasks
- You're writing scripts for deployment
- You're performing complex aggregations
- You need precise control over operations
- You're working in production environments

Practical Demo: Creating a Database

In Compass

1. Click "Create Database" button
2. Enter database name (e.g., "learningDB")
3. Enter initial collection name (e.g., "users")
4. Click "Create Database"
5. Your new database appears in the sidebar

In Shell

1. Open mongosh terminal
2. Switch to new database:

```
use learningDB
```

1. Create collection and insert document:

```
db.users.insertOne({ name: "Alice", age: 28 })
```

Database is created automatically upon first document insertion.

Practical Demo: Inserting Documents

Let's explore how to add data using both interfaces. This side-by-side comparison illustrates the different approaches to accomplishing the same task.

GUI Approach

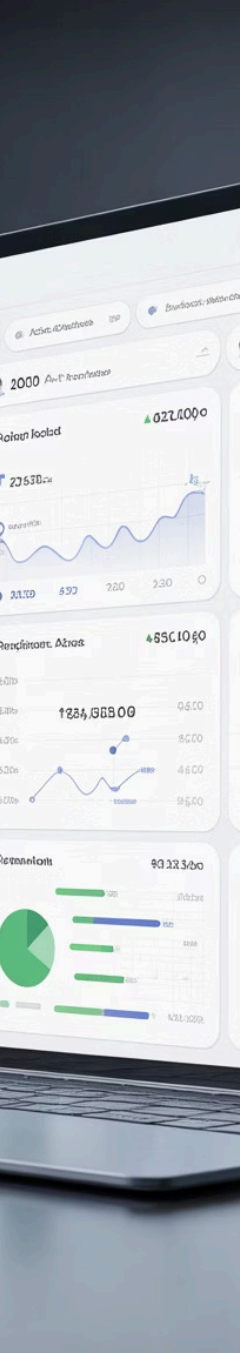
1. Navigate to your collection in Compass
2. Click the "Add Data" button
3. Select "Insert Document"
4. Fill in the document fields using the form
5. Click "Insert" to add the document

Compass validates your input and provides helpful error messages if something is incorrect.

Shell Approach

```
db.users.insertOne({
  name: "Bob Smith",
  email: "bob@example.com",
  age: 32,
  active: true
})
```

The shell returns an acknowledgement with the inserted document's ObjectId. You can insert multiple documents at once using `insertMany()`.



Monitoring Database Performance

Understanding your database's health and performance is crucial for maintaining efficient applications. MongoDB provides powerful tools for monitoring in both interfaces.

Using Compass

- View real-time performance metrics in the Performance tab
- Analyse slow queries and identify bottlenecks
- Monitor operation execution times
- Visual charts show trends over time

Using Shell Commands

```
db.stats()
```

Displays database statistics including size and collection count.

```
db.serverStatus()
```

Returns detailed server metrics including connections, operations, and memory usage.

Understanding Database Statistics

When you run `db.stats()` in the MongoDB Shell, you receive valuable insights about your database. Here's what the key metrics tell you:

Database Size

The total amount of storage space used by your database, measured in bytes. This helps you plan for storage capacity and optimise data structure.

Collections Count

The number of collections in your database. Each collection groups related documents together, similar to tables in relational databases.

Index Size

The storage space consumed by database indexes. Indexes improve query performance but require additional storage. Monitoring this helps balance speed and space.

Objects Count

The total number of documents across all collections. This metric helps you understand the scale of your data and plan for growth.

Best Practices for Beginners

Start with Compass

Begin your MongoDB journey using Compass to visualise data structures and understand document relationships. The visual interface helps build mental models of how NoSQL databases work.

Practise Shell Commands

Once comfortable with Compass, replicate operations in the Shell. This dual approach reinforces learning and prepares you for production environments where CLI access is common.

Keep a Command Reference

Maintain a personal reference document with commonly used commands and their syntax. This speeds up your workflow and helps you remember the patterns.

Common Beginner Mistakes to Avoid

Forgetting to Switch Databases

Always verify you're in the correct database before running commands. Use `db` to check your current database. Many errors occur from working in the wrong database context.

Not Using Proper JSON Syntax

MongoDB requires valid JSON format for documents. Common errors include missing quotes around keys, trailing commas, and incorrect data types. Compass validates syntax automatically, helping you learn correct formats.

Ignoring Index Optimisation

While it's tempting to focus solely on data storage, proper indexing dramatically improves query performance. Learn to create indexes on frequently queried fields early in your journey.

Overlooking Data Validation

MongoDB's flexible schema doesn't mean ignoring data structure. Establish validation rules to maintain data quality and prevent inconsistencies as your application grows.

Key Takeaways

As you continue your MongoDB journey, remember these essential points that will guide your learning and development:

Complementary Tools

MongoDB offers both GUI (Compass) and CLI (Shell) interfaces that work together, not in competition. Each has strengths for different scenarios.

Progressive Learning

Begin with Compass for visual understanding, then progress to Shell for scripting and automation as your comfort and skills develop.

Enhanced Productivity

Mastering both tools enhances productivity across user types, from beginners to advanced developers. The investment in learning pays dividends.

Continue Your Learning Journey

You've taken the first important steps in understanding MongoDB's interfaces. Here's where to go next to deepen your knowledge and practical skills:

Next Steps

- Explore **MongoDB Atlas** for cloud-based database deployments with free tier options
- Visit **MongoDB Documentation** at mongodb.com/docs for comprehensive guides
- Practise CRUD operations, schema exploration, and indexing with sample datasets
- Join the MongoDB Community forums to connect with other learners and experts

Pro Tip

Set aside 30 minutes daily to practise MongoDB operations. Consistency builds confidence and competence faster than intensive but infrequent study sessions.