

# How to setup a local MongoDB Connection

I always use MongoDB as a database when I work on an app. And I like to connect to a database on my computer because it speeds up dev and test-related work.

Today, I want to share how to create and connect to a local MongoDB Database.

## Installing MongoDB

You need to install MongoDB on your computer before you can connect to it. You can install MongoDB by following these instructions ([Mac](#) and [Windows](#)).

Once you have completed the installation process, try typing

```
mongo --  
version
```

into your command line. You should get a response similar to the following:

```
mongo --version
```

```
[~] mongo --version  
MongoDB shell version v4.0.3  
git version: 7ea530946fa7880364d88c8d8b6026bbc9ffa48c  
allocator: system  
modules: none  
build environment:  
  distarch: x86_64  
  target_arch: x86_64  
[~] █
```

## Starting MongoDB

You can start MongoDB on your computer with the `mongod` command.

```
mongod
```

```
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2019-05-29T11:26:42.297+0800 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/data/db/diagnostic.data'
2019-05-29T11:26:42.310+0800 I NETWORK [initandlisten] waiting for connections on port 27017
2019-05-29T11:26:43.008+0800 I FTDC [ftdc] Unclean full-time diagnostic data capture shutdown detected, found interim file, some metrics may have been lost. OK
```

Keep the `mongod` window running when you want to work with your local MongoDB. MongoDB stops when you close the window.

# Brief overview of how MongoDB works

MongoDB lets you store things (called *documents*) inside *databases*. Each database contains multiple *collections*.

To make it easier to understand, you can think of MongoDB as a building. It contains many rooms.

Each room is a database. Each database is responsible for storing information about one application. You can store as much information as you want.

You have an unlimited supply of boxes in each room. Each box is a collection. Each collection can only contain one type of data.

For example, one collection can be used for books, one collection for users, one collection for toys, and so on.

# Adding items to a database

One way to add items to a MongoDB database is through the Mongo Shell.

To open up the Mongo Shell, you open another command line window and run `mongo`.

```
mongo

Last login: Wed May 29 11:43:06 on tty001
[~] mongo
MongoDB shell version v4.0.3
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("80831262-598b-4a1e-9ef5-4cb5d4a6ba10") }
MongoDB server version: 4.0.3
Server has startup warnings:
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2019-05-29T11:26:42.212+0800 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
...
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

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

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

Note: Make sure you keep the `mongod` window open! You won't be able to interact with the Mongo Shell if you close the `mongod` window.

First, we need a database to work with. You can see the currently selected database with the `db` command. (By default, you should be on the `test` database).

```
> db
```

Note: The `>` in the code above signifies the Mongo Shell. You don't need to type `>`. It is not part of the command.

```
> db
test
> 
```

For this article, we'll create a database called `game-of-thrones`. You can use the `use <database>` command to create and switch to a new database.

```
> use game-of-thrones
```

```
> use game-of-thrones
switched to db game-of-thrones
> 
```

We're going to add a character into the `game-of-thrones`. Here, we need to put the character into a collection. We'll use `characters` as the name of the collection.

To add an item to a collection, you can pass a JavaScript object into

```
db.<collectionName>.insertOne()
```

```
db.characters.insertOne({ name: 'Jon Snow' })
```

```
> db.characters.insertOne({ name: 'Jon Snow' })
{
  "acknowledged" : true,
  "insertedId" : ObjectId("5cee31b688ffc30414a0303c")
}
> 
```

Let's add one character into the database before we continue.

```
db.characters.insertOne({ name: 'Arya Stark' })
```

```
> db.characters.insertOne({ name: 'Arya Stark' })
{
  "acknowledged" : true,
  "insertedId" : ObjectId("5cee31c288ffc30414a0303d")
}
> █
```

You can see the characters we've added by using the `find` command. (`db.<collectionName>.find()`).

```
db.characters.find()
```

```
> db.characters.find()
{ "_id" : ObjectId("5cee31b688ffc30414a0303c"), "name" : "Jon Snow" }
{ "_id" : ObjectId("5cee31c288ffc30414a0303d"), "name" : "Arya Stark" }
> █
```

This is all you need to know about the Mongo Shell for now.

## Accessing MongoDB with MongoDB Compass

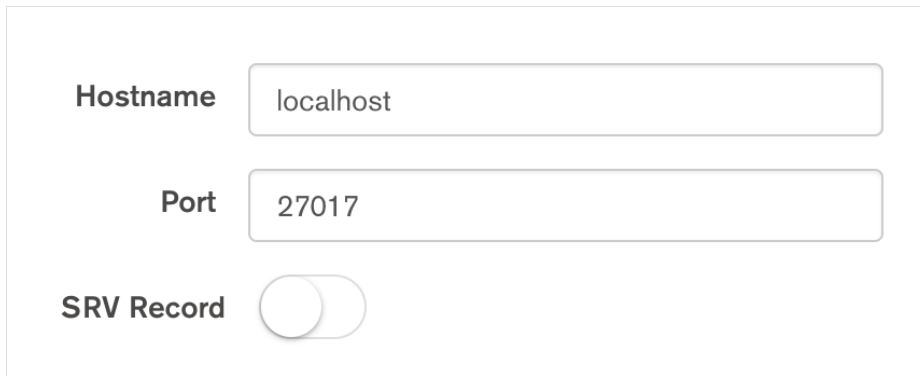
MongoDB Compass gives you another way to access MongoDB. It's an app that makes checking (and editing) databases easier if you're not a fan of the command line.

To use MongoDB Compass, you have to install it first. You can download and install MongoDB Compass from the [this page](#).

When you open MongoDB Compass, you'll see a screen that looks like this:

Connect to Host	
Hostname	<input type="text" value="localhost"/>
Port	<input type="text" value="27017"/>
SRV Record	<input type="checkbox"/>
Authentication	<div>None</div>
Replica Set Name	<input type="text"/>
Read Preference	<div>Primary</div>
SSL	<div>None</div>
SSH Tunnel	<div>None</div>
Favorite Name ⓘ	<input type="text" value="e.g. Shared Dev, QA Box, PRODUCTION"/>
<div>CONNECT</div>	

To connect to your local MongoDB, you set `Hostname` to `localhost` and `Port` to `27017`. These values are the default for all local MongoDB connections (unless you changed them).









Hostname: localhost

Port: 27017


SRV Record: ☐

Press connect, and you should see the databases in your local MongoDB. Here, you should be able to see `game-of-thrones` (the database we created for this tutorial).

Database Name	Storage Size	Collections	Indexes	
admin	16.0KB	0	1	
config	36.0KB	0	2	
game-of-thrones	16.0KB	0	1	
learnjavascript-api	32.0KB	2	3	
local	40.0KB	1	1	
students-dev	32.0KB	2	4	

 The database we created

If you click on `game-of-thrones`, you'll see a `characters` collection.

Collection Name ^	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	
characters	2	42.0 B	84.0 B	1	16.0 KB	

And if you click on `characters`, you'll see the two characters we created in the earlier section.

<a href="#">INSERT DOCUMENT</a>	<a href="#">VIEW</a>	<a href="#">LIST</a>	<a href="#">TABLE</a>	Displaying documents 1 - 2 of 2
<pre>{ "_id": ObjectId("5cee31b688ffc30414a0303c"),   "name": "Jon Snow" }</pre>				
<pre>{ "_id": ObjectId("5cee31c288ffc30414a0303d"),   "name": "Arya Stark" }</pre>				

This is how you can use MongoDB Compass to connect to a MongoDB that's running on your own computer.

## Connecting to MongoDB with a Node server

When we build applications, we connect to MongoDB through our applications (not through Mongo Shell nor MongoDB Compass).

To connect to MongoDB, we need to use the [mongodb](#) package.

Alternatively, you can also use [Mongoose](#).

(By the way, I prefer using Mongoose over the MongoDB native driver. I'll share why in a future article).

## Connecting with MongoDB native driver

First you have to install and require the mongodb package.

```
npm install mongodb --save
```

```
const MongoClient = require('mongodb').MongoClient
```

You can connect to your local MongoDB with this url:

```
const url = 'mongodb://127.0.0.1:27017'
```

With the Mongo Client, you need to specify the database you're using after you connect to MongoDB. Here's what it looks like:

```
const dbName = 'game-of-thrones'
let db

MongoClient.connect(url, { useNewUrlParser: true }, (err, client)
=> {
  if (err) return console.log(err)

  // Storing a reference to the database so you can use it later
  db = client.db(dbName)
  console.log(`Connected MongoDB: ${url}`)
  console.log(`Database: ${dbName}`)
})
```

```
[test] node server.js
Connected MongoDB: mongodb://127.0.0.1:27017
Database: game-of-thrones
```

## Connecting with Mongoose

To connect with Mongoose, you need to download and require

```
mongoose
```

```
npm install mongoose --save
```

```
const mongoose = require('mongoose')
```

When you use Mongoose, the connection `url` should include the database you're connecting to:

```
const url = 'mongodb://127.0.0.1:27017/game-of-thrones'
```

You can connect to MongoDB with the `connect` method:

```
mongoose.connect(url, { useNewUrlParser: true })
```

Here's how you can check whether the connection succeeds.

```
const db = mongoose.connection
db.once('open', _ => {
  console.log('Database connected:', url)
})

db.on('error', err => {
  console.error('connection error:', err)
})
```

```
[test] node server.js
Database connected: mongodb://127.0.0.1:27017/game-of-thrones
```