MongoDB is a **NoSQL database** that stores data in a flexible, **JSON-like format** called **BSON** (Binary JSON).

Unlike traditional relational databases like MySQL or PostgreSQL, which use tables and rows,

MongoDB uses collections and documents.

Key Concepts

Concept	Description
Database	A container for collections.
Collection	A group of documents, similar to a table in SQL.
Document	A single record in a collection, stored as a JSON-like object.

Example of a document:

```
{
  "name": "Alice",
  "age": 30,
  "email": "alice@example.com"
}
```

Download it from :-

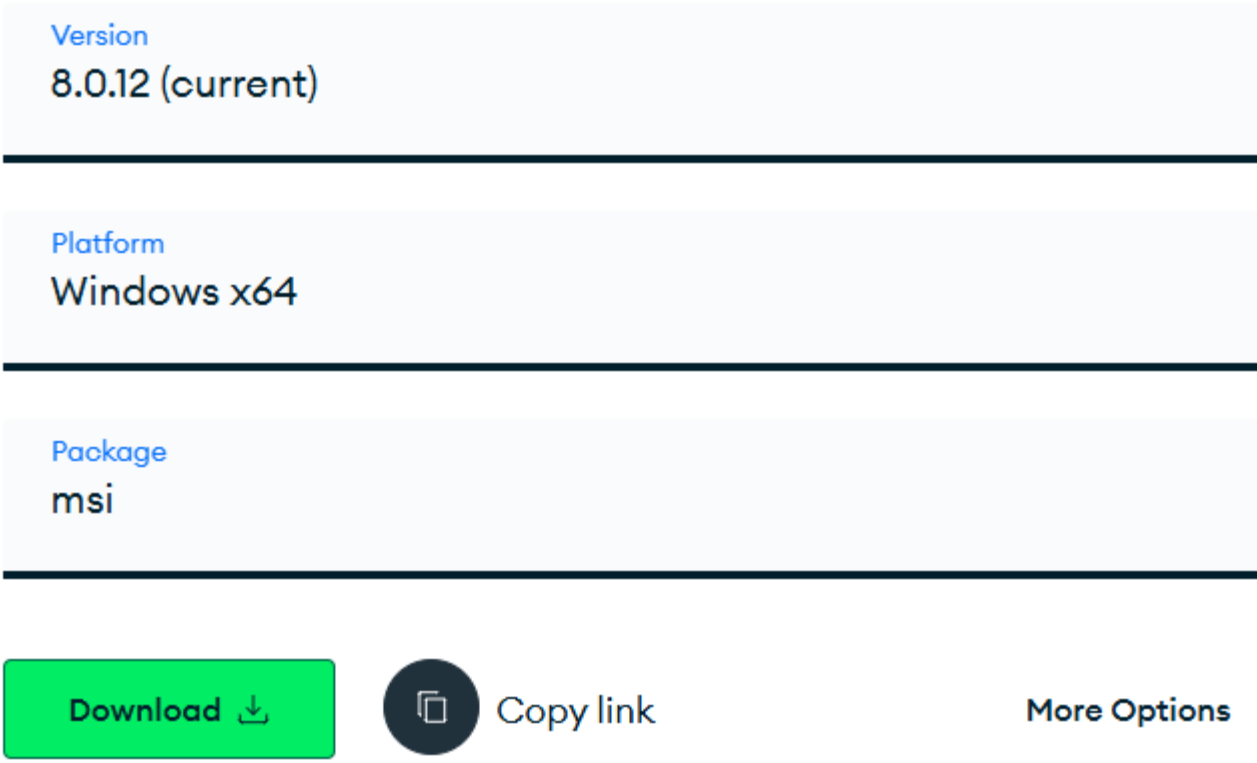
https://www.mongodb.com/try/download/community

for windows :-

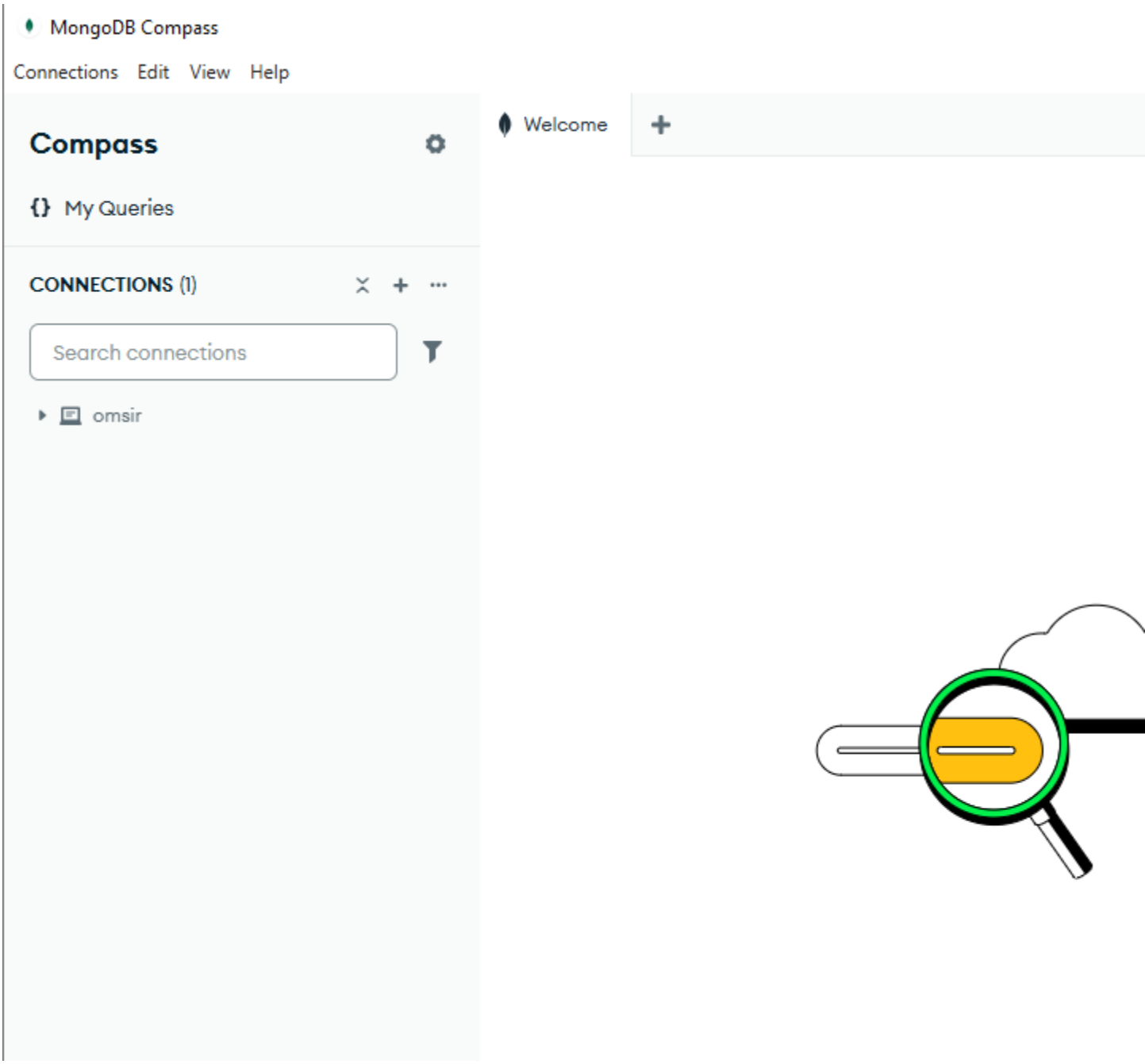
access to advanced functionality such as auto-scale, full-text search, and data distribution across regions and clouds. Deploy in minutes on AWS, Google Cloud, an Azure, with no downloads necessary.

Give it a try with a free, highly-available 512 MB cluster. or get started from your ter with the following two commands:

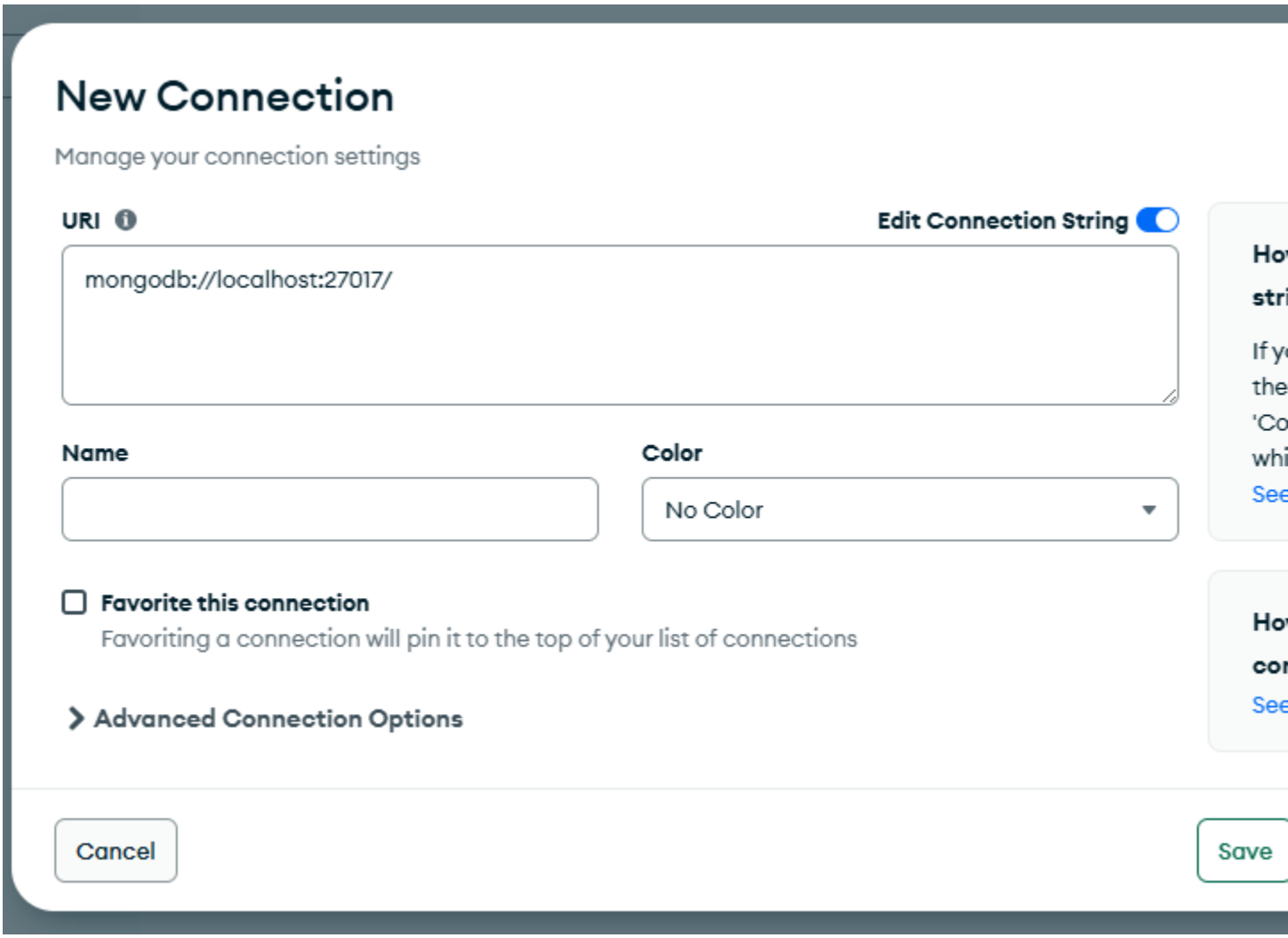
```
$ brew install mongodb-atlas
$ atlas setup
```



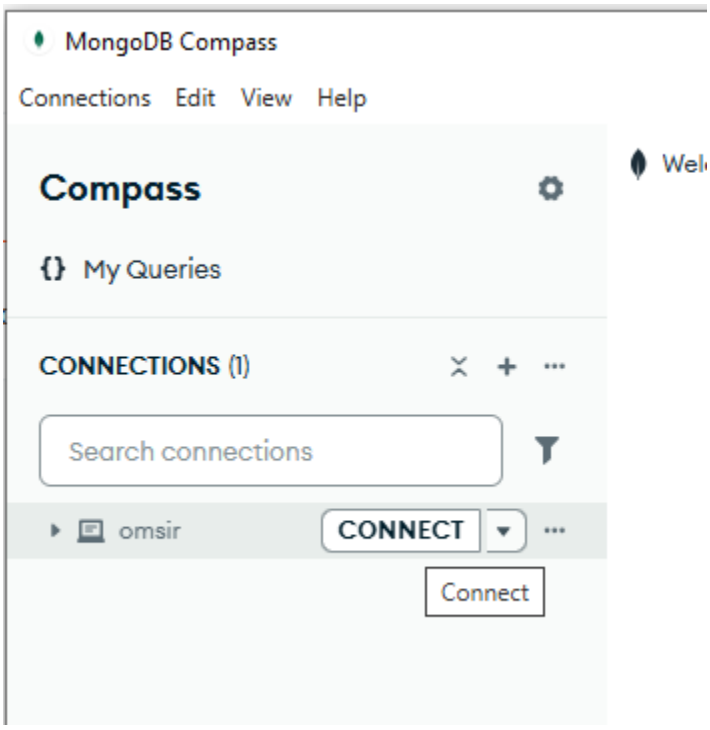
and after installation open it :-



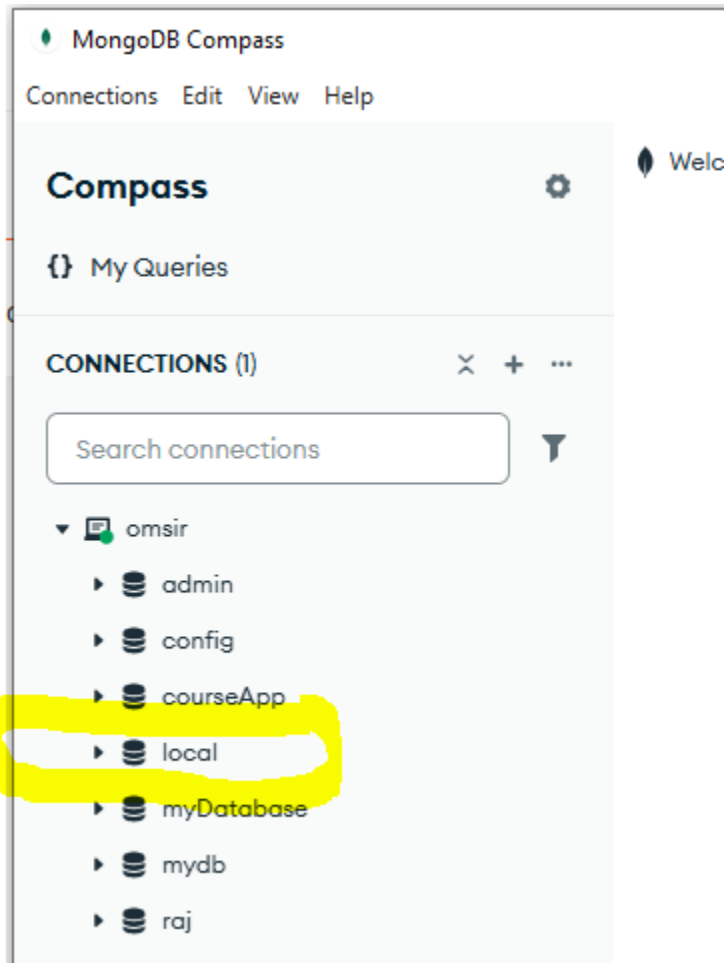
and create connection and connect it for example enter connection name as omsir :-



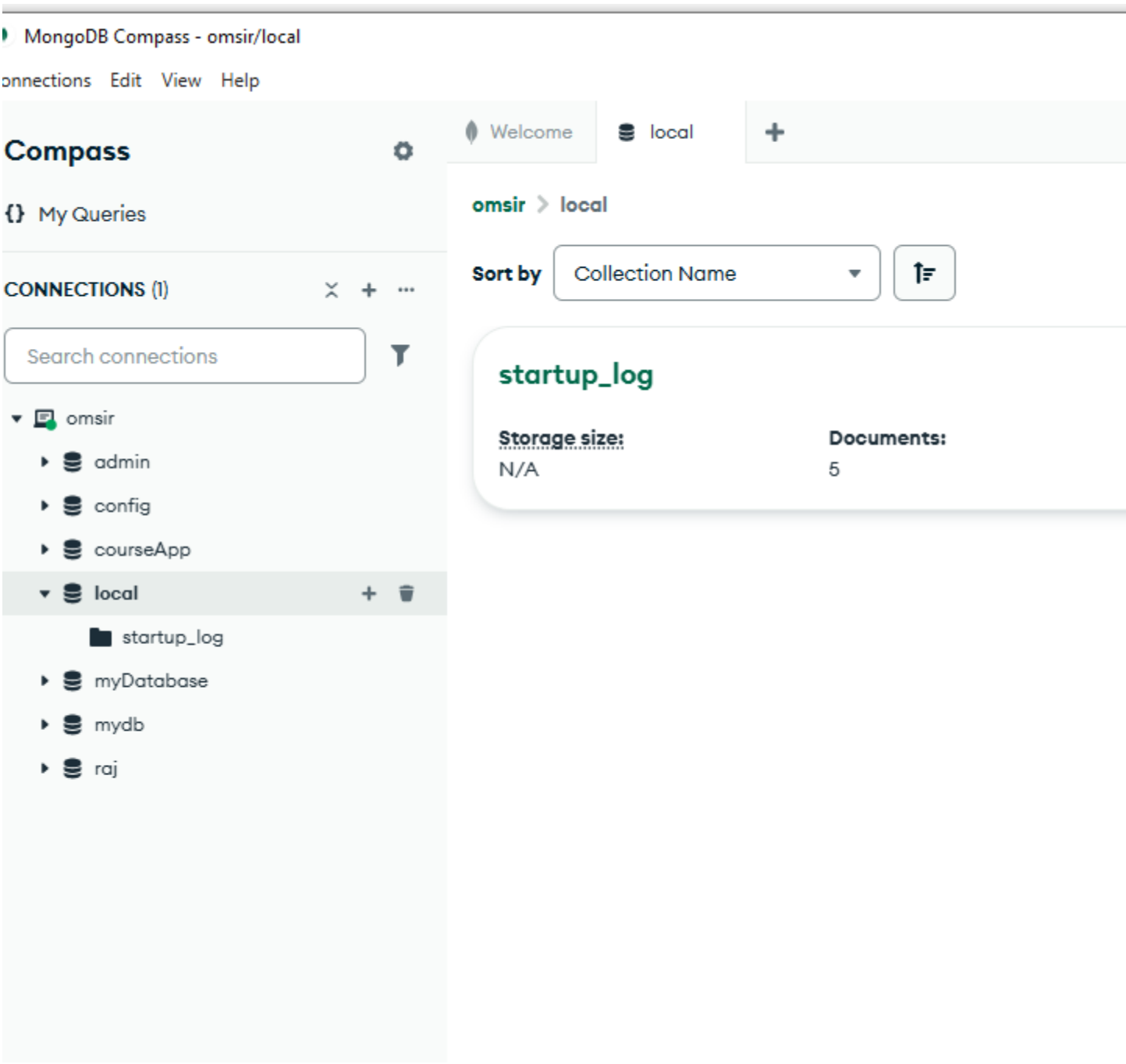
and then connect it :-



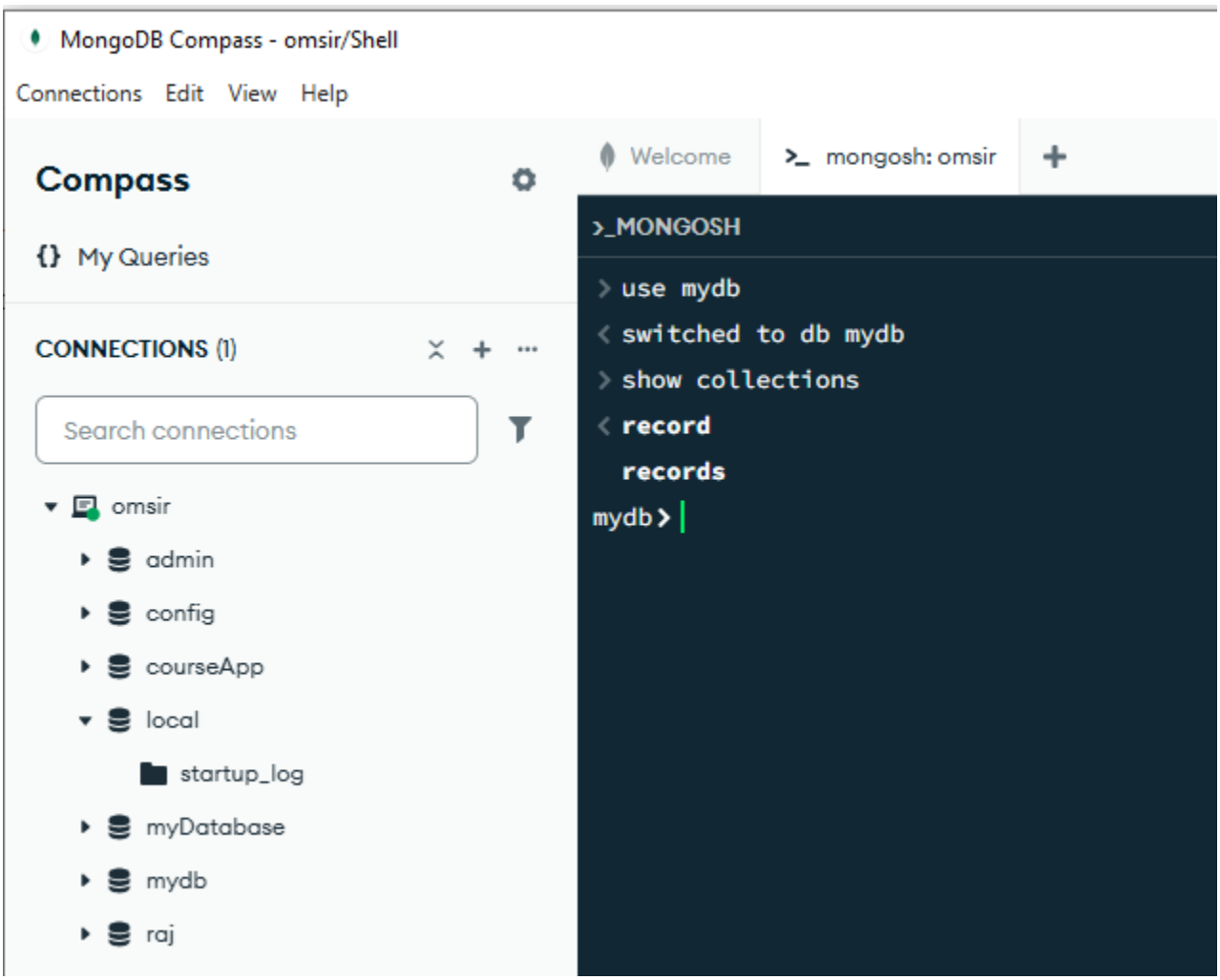
and then you will see following here just click on local you will see open mongodb shell options:-



and then you will see Open MongoDb shell and click on it and apply queries :-



and then use queries in it to show collections as shown below :-



and now check data of records collection use following queries :-

1. Insert Data into `records` Collection

```
db.records.insertMany([
  { name: "Alice", age: 30, city: "New York", status: "active" },
  { name: "Bob", age: 25, city: "Los Angeles", status: "inactive"
},
  { name: "Charlie", age: 35, city: "Chicago", status: "active" },
  { name: "David", age: 28, city: "New York", status: "pending" },
  { name: "Eva", age: 40, city: "Los Angeles", status: "active" }
])
```

✏ 2. Update a Document

Update one document (e.g., change Bob's status to "active"):

```
db.records.updateOne(
  { name: "Bob" },
  { $set: { status: "active" } }
)
```

3. Filter / Search the Collection

 Find all documents:

```
db.records.find()
```

 Find by city:

```
db.records.find({ city: "New York" })
```

 Find by status:

```
db.records.find({ status: "active" })
```

 Find age greater than 30:

```
db.records.find({ age: { $gt: 30 } })
```

 Find name starting with "A" (regex):

```
db.records.find({ name: /^A/ })
```

 Combine filters (e.g., active users in LA):

```
db.records.find({
  city: "Los Angeles",
  status: "active"
})
```

4. (Optional) Delete Documents

Delete one:

```
db.records.deleteOne({ name: "David" })
```

Delete all inactive users:

```
db.records.deleteMany({ status: "inactive" })
```

□ Summary of All Commands (In Order)

```
use mydb

db.records.insertMany([
  { name: "Alice", age: 30, city: "New York", status: "active" },
  { name: "Bob", age: 25, city: "Los Angeles", status: "inactive"
},
  { name: "Charlie", age: 35, city: "Chicago", status: "active" },
  { name: "David", age: 28, city: "New York", status: "pending" },
  { name: "Eva", age: 40, city: "Los Angeles", status: "active" }
])

db.records.updateOne(
  { name: "Bob" },
  { $set: { status: "active" } }
)

db.records.updateMany(
```

```
  { status: "pending" },
  { $set: { status: "inactive" } }
)

db.records.find()
db.records.find({ city: "New York" })
db.records.find({ status: "active" })
db.records.find({ age: { $gt: 30 } })
db.records.find({ name: /^A/ })
db.records.find({ city: "Los Angeles", status: "active" })
```