MongoDB

Contents

- What is NoSQL database?
- What is the difference between SQL and NoSQL
- What is MongoDB?
- Installation of MongoDB and Mongosh(shell)
- Database operations-CURD(create,Update,Read,Delete)
- MongoDB with Node.js
  - Adding mongodb driver to node.js
  - Connecting to mongodb using node.js

What is NoSQL?

- NoSQL stands for "not only SQL“.
- NoSQL databases are non-relational databases that store data in a flexible schema model.
- NoSQL databases store data in a non-tabular format
- NoSQL databases are designed to handle large amounts of unstructured data.
- NoSQL databases are well-suited to the large amounts of data generated by the cloud, mobile, and social media
- NoSQL databases are ideal for developing applications quickly and iteratively.

Examples of NoSQL databases

MongoDB, Cassandra, Redis, Elasticsearch, BigTable, Neo4j, HBase, and Amazon DynamoDB.

What is a MongoDB?

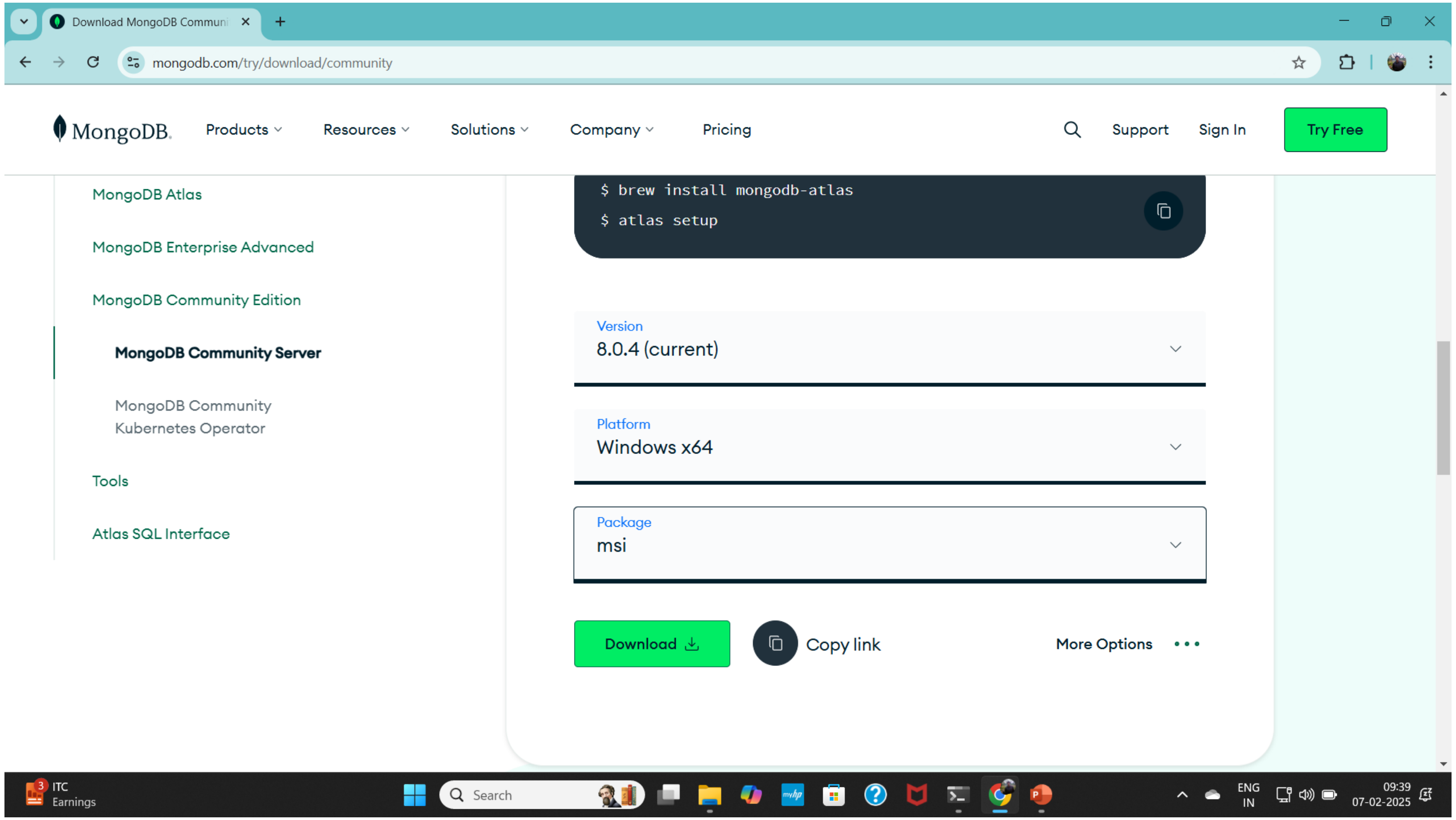
MongoDB is an open source, document-oriented database. It is designed to be highly scalable and offers high developer productivity. MongoDB stores data in JSON-like documents which have dynamic schema.

NoSQL vs SQL

NoSQL(MongoDB)	SQL(Oracle)
Stores data in collections	Stores data in tables
Unit of data storage is a document which is stored in a collection	Unit of data storage is a record(table row)
Collections have dynamic schema i.e., documents in collections have different fields	Tables have fixed schema i.e., attributes are pre defined before inserting data. Explicit NULL value has to be provided if data is missing for an attribute
CRUD operations are performed through insert , find, update, and remove operations on collection object	CRUD operations are performed through INSERT,SELECT,UPDATE and DELETE statements
PRIMARY KEY uniquely identifies a document in a collection. PRIMARY KEY field has a predefined name _id	PRIMARY KEY uniquely identifies a record in a table. You can choose any name for PRIMARY KEY
NOT NULL, UNIQUE, FOREIGN KEY and CHECK constraints are not supported	NOT NULL, UNIQUE, FOREIGN KEY and CHECK constraints are supported
Joins and Subquery are not supported	Joins and Subquery are supported

Installation of MongoDB

Step 1: Download the MongoDB Community Server installer from https://www.mongodb.com/try/download/community

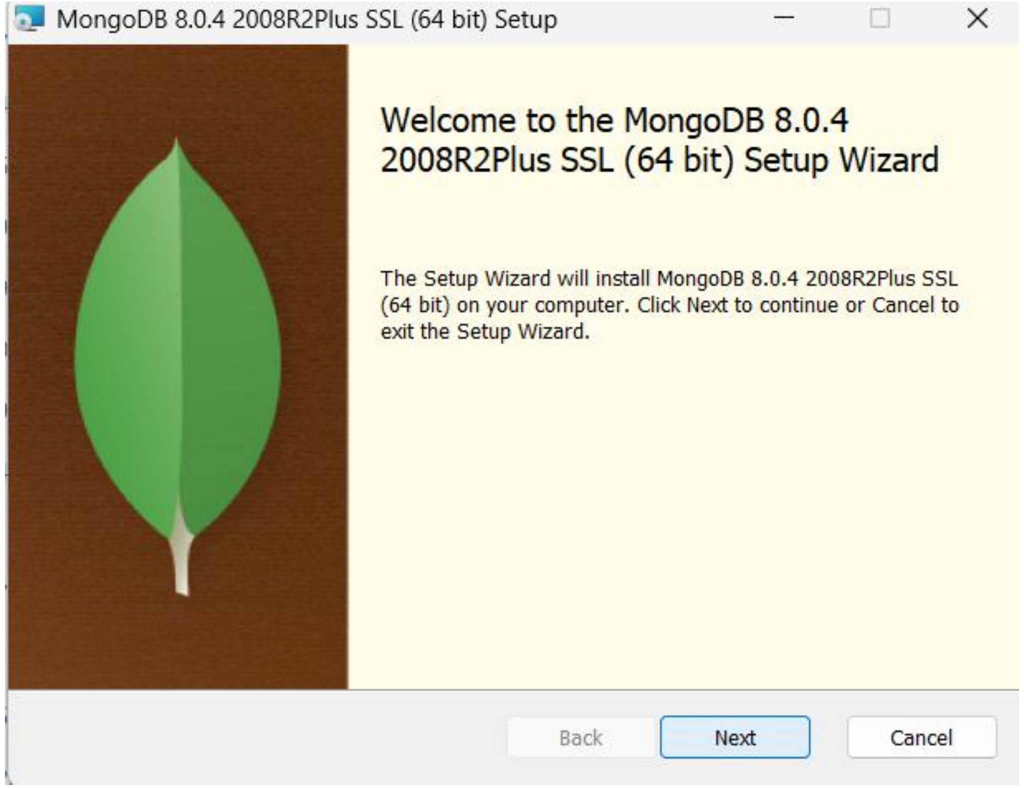


```
$ brew install mongodb-atlas
$ atlas setup
```

Version: 8.0.4 (current)

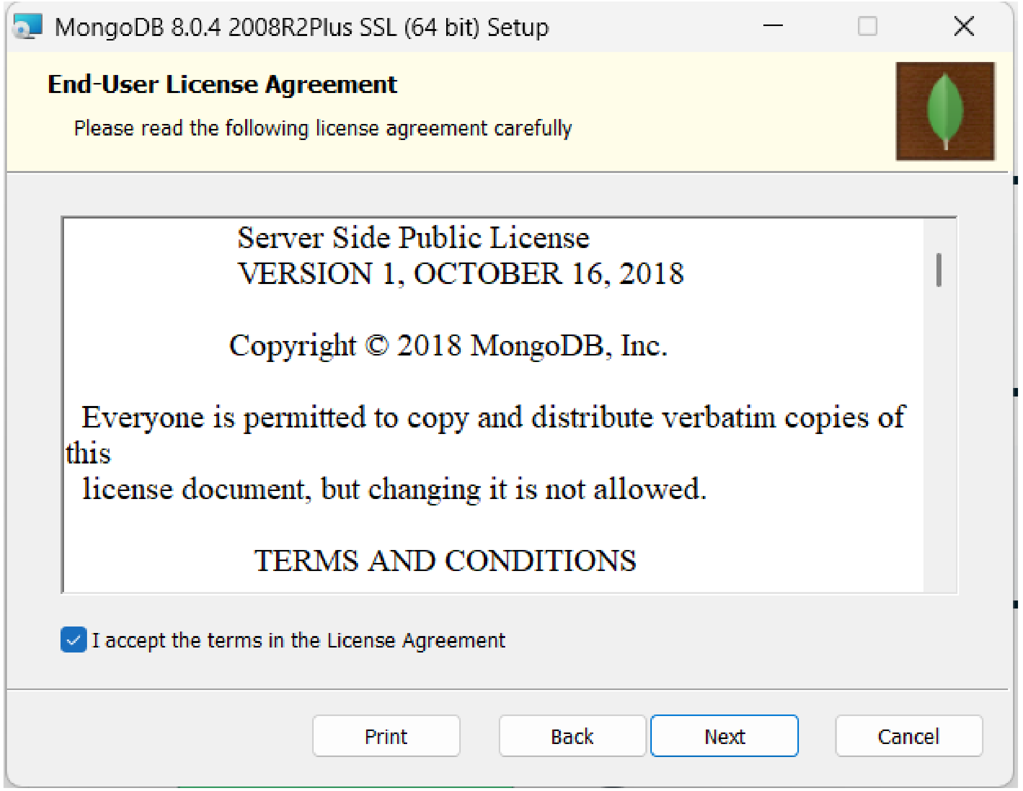
Platform: Windows x64

Package: msi


MongoDB 8.0.4 2008R2Plus SSL (64 bit) Setup

Welcome to the MongoDB 8.0.4 2008R2Plus SSL (64 bit) Setup Wizard

The Setup Wizard will install MongoDB 8.0.4 2008R2Plus SSL (64 bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.


MongoDB 8.0.4 2008R2Plus SSL (64 bit) Setup

End-User License Agreement

Please read the following license agreement carefully

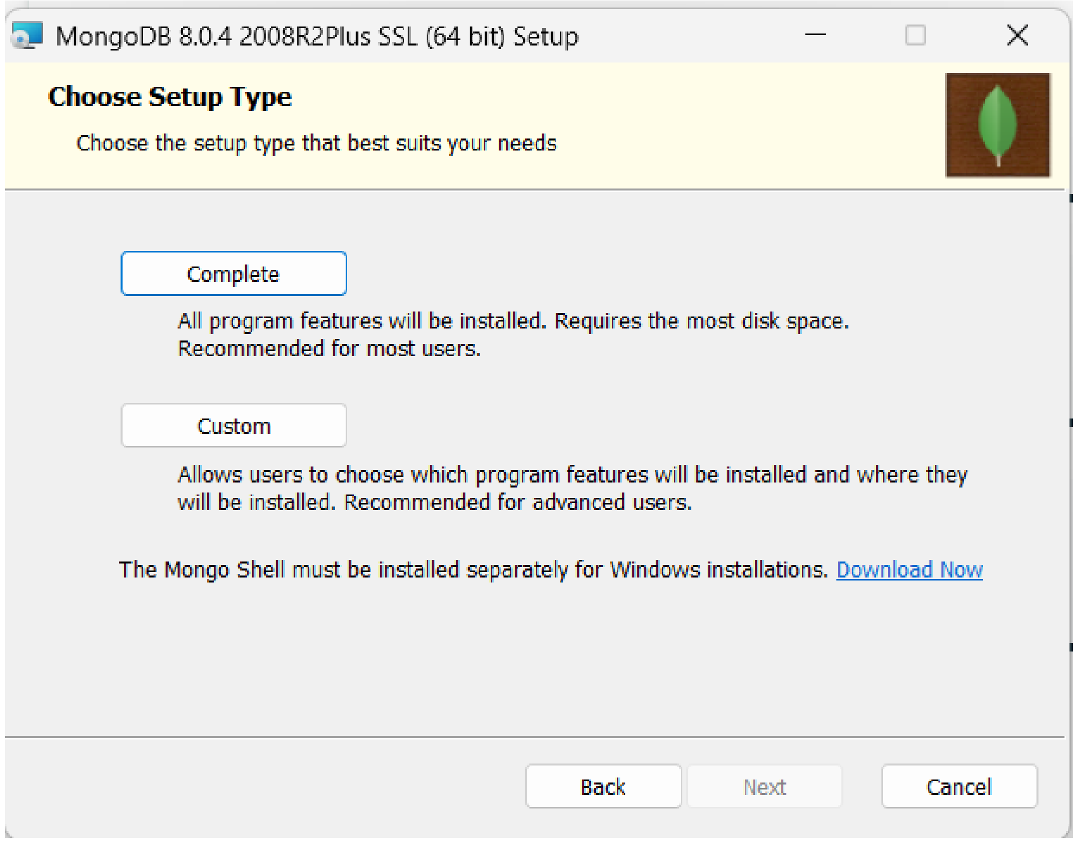
Server Side Public License
VERSION 1, OCTOBER 16, 2018

Copyright © 2018 MongoDB, Inc.

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

TERMS AND CONDITIONS

☑ I accept the terms in the License Agreement



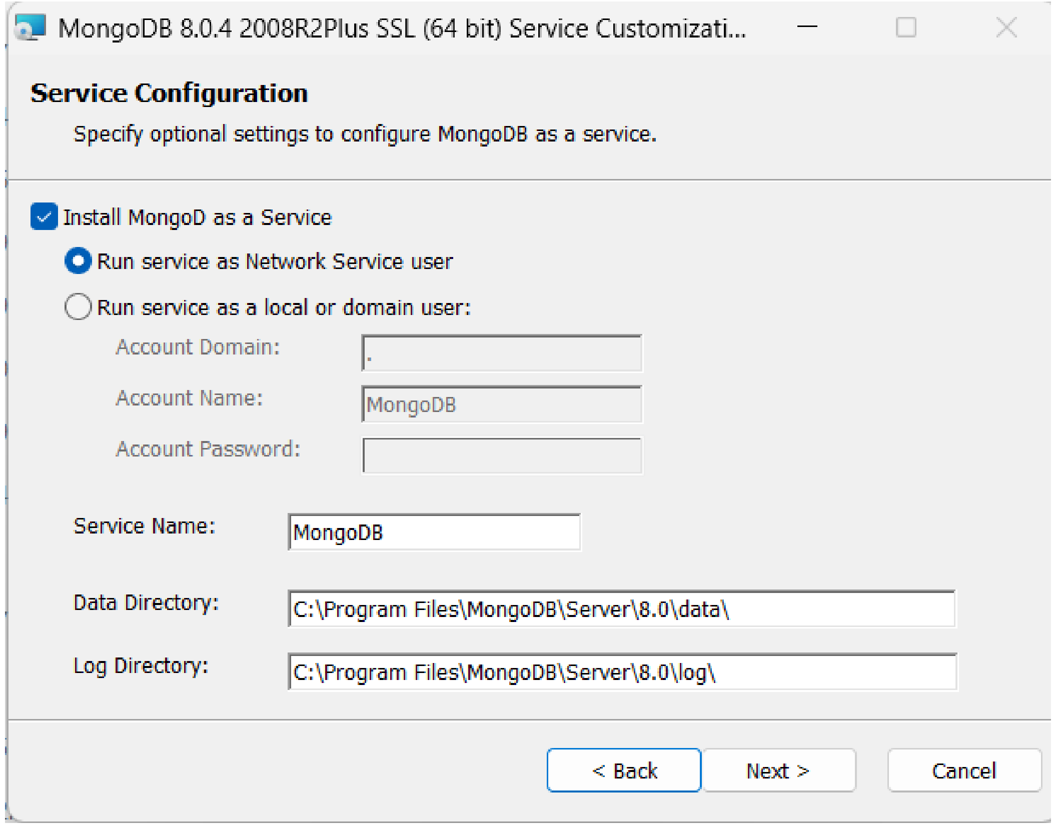
Choose Setup Type

Choose the setup type that best suits your needs

Complete: All program features will be installed. Requires the most disk space. Recommended for most users.

Custom: Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

The Mongo Shell must be installed separately for Windows installations. Download Now



MongoDB 8.0.4 2008R2Plus SSL (64 bit) Service Customizati...
< Back
Next >
Cancel

Service Configuration

Specify optional settings to configure MongoDB as a service.

☑ Install MongoD as a Service

● Run service as Network Service user

○ Run service as a local or domain user:

Account Domain: .

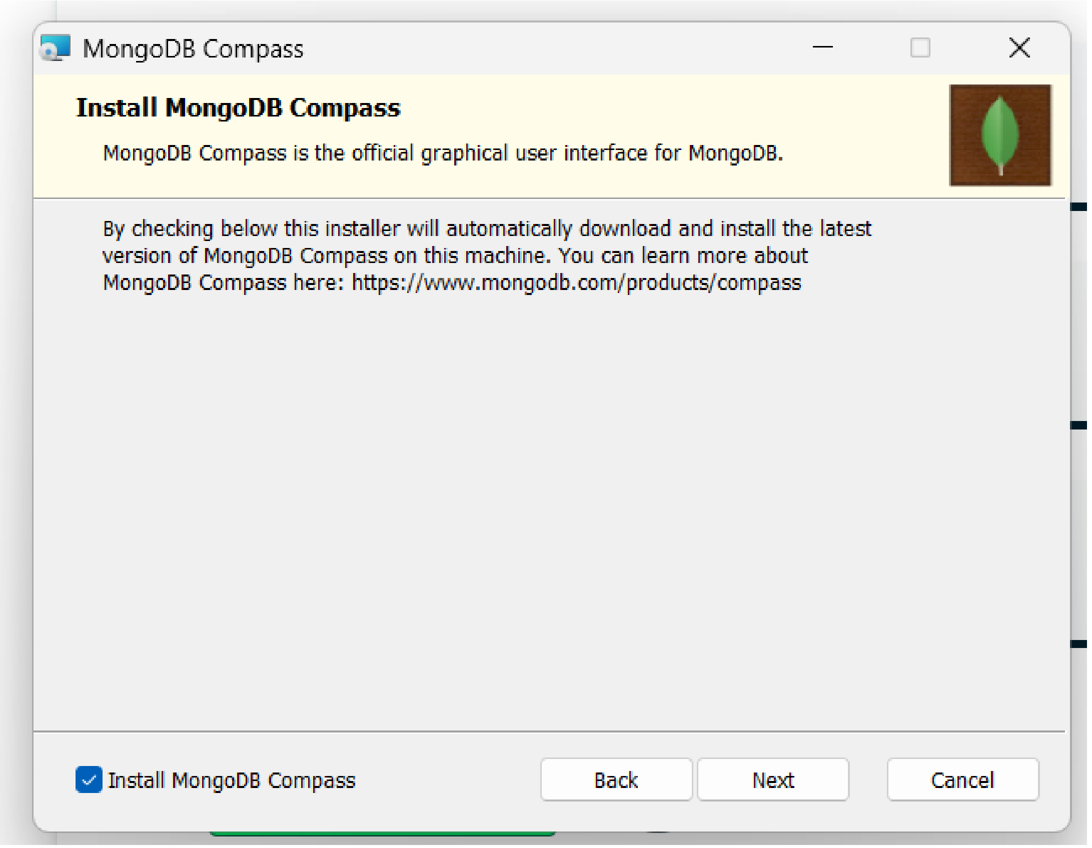
Account Name: MongoDB

Account Password:

Service Name: MongoDB

Data Directory: C:\Program Files\MongoDB\Server\8.0\data\

Log Directory: C:\Program Files\MongoDB\Server\8.0\log\

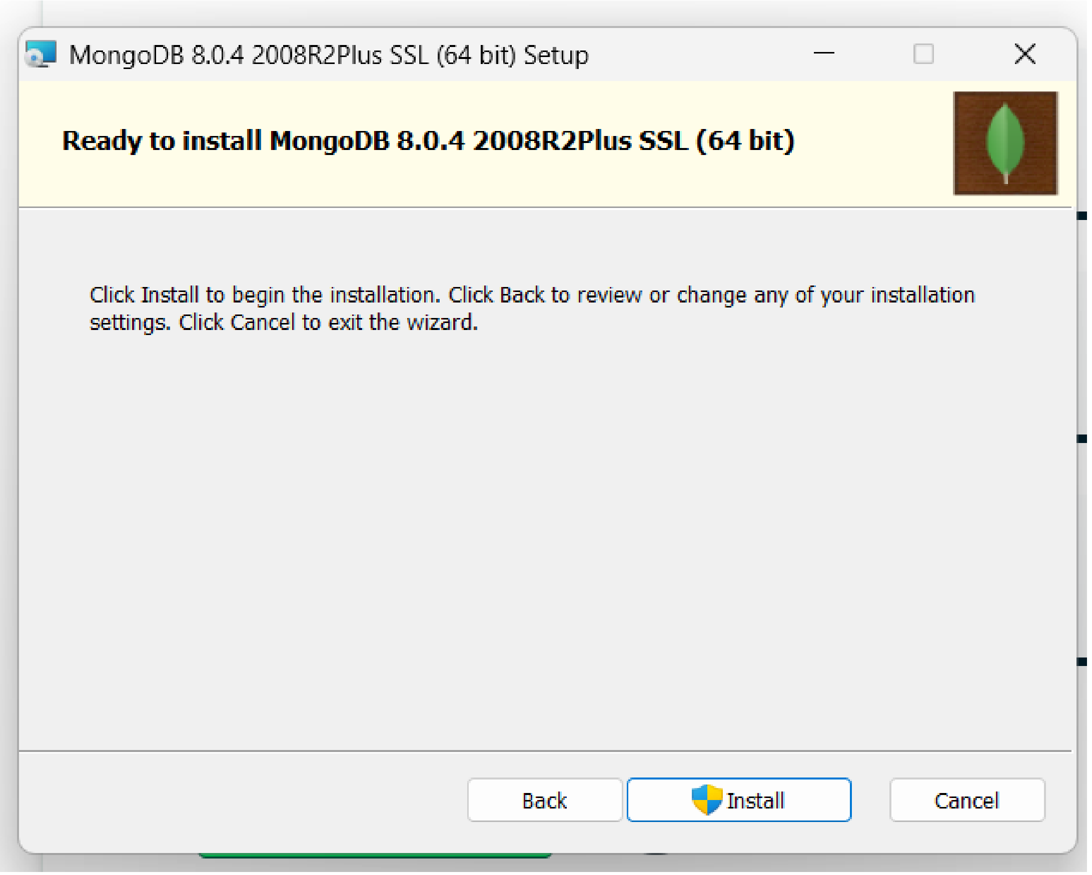


Install MongoDB Compass

MongoDB Compass is the official graphical user interface for MongoDB.

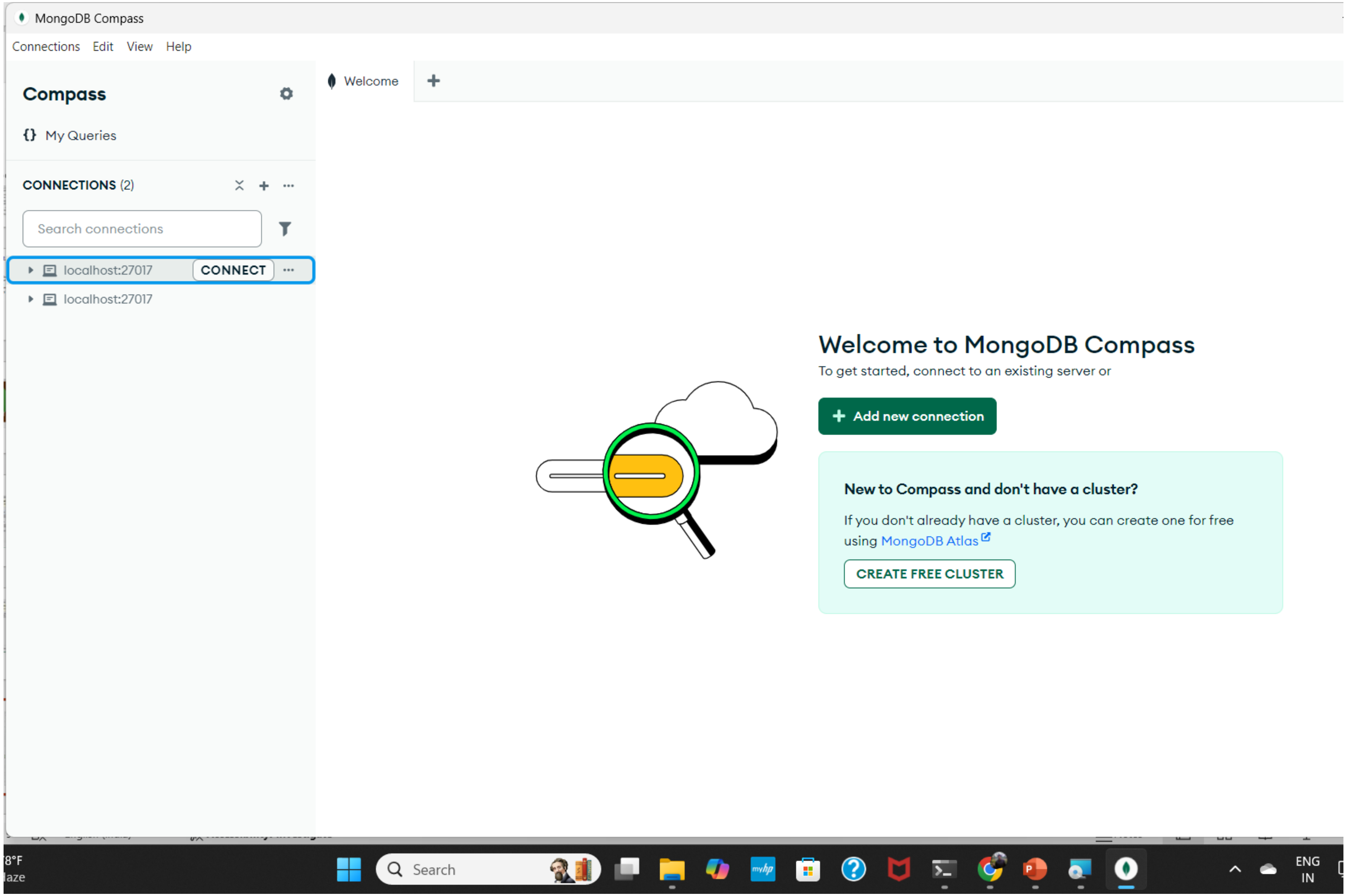
By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/compass

☑ Install MongoDB Compass



Ready to install MongoDB 8.0.4 2008R2Plus SSL (64 bit)

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.



Welcome to MongoDB Compass

To get started, connect to an existing server or

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

mongosh(MongoDB shell)

- The MongoDB Shell, mongosh, is a JavaScript and Node.js REPL(READ EVAL PRINT LOOP) environment for interacting with MongoDB deployments in Atlas, locally, or on another remote host.
- MongoDB Shell is used to test queries and interact with the data in your MongoDB database.

Download the mongosh from

https://www.mongodb.com/docs/mongodb-shell/

CRUD operations

The MongoDB shell provides the following methods to insert documents into a collection:

- To insert a single document, use db.collection.insertOne().
- To insert multiple documents, use db.collection.insertMany().

db.collection.insertOne() - Inserts a single document into a collection.
Return

- A document containing:A boolean acknowledged as true if the operation ran with write concern or false if write concern was disabled.
- A field insertedId with the _id value of the inserted document.

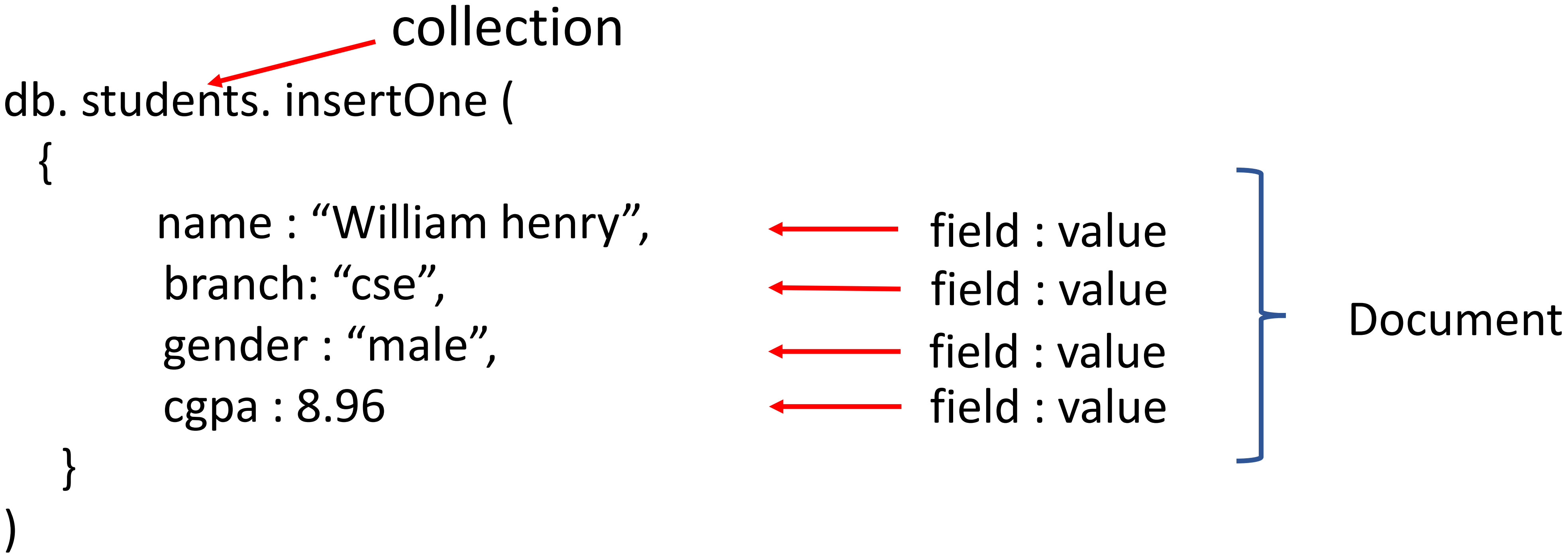
CRUD operations

Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

- db.collection.insertOne()
- db.collection.insertMany()

In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.



collection

```
db. students. insertOne (
  {
          name : “William henry”,
          branch: “cse”,
          gender : “male”,
          cgpa : 8.96
    }
)
```

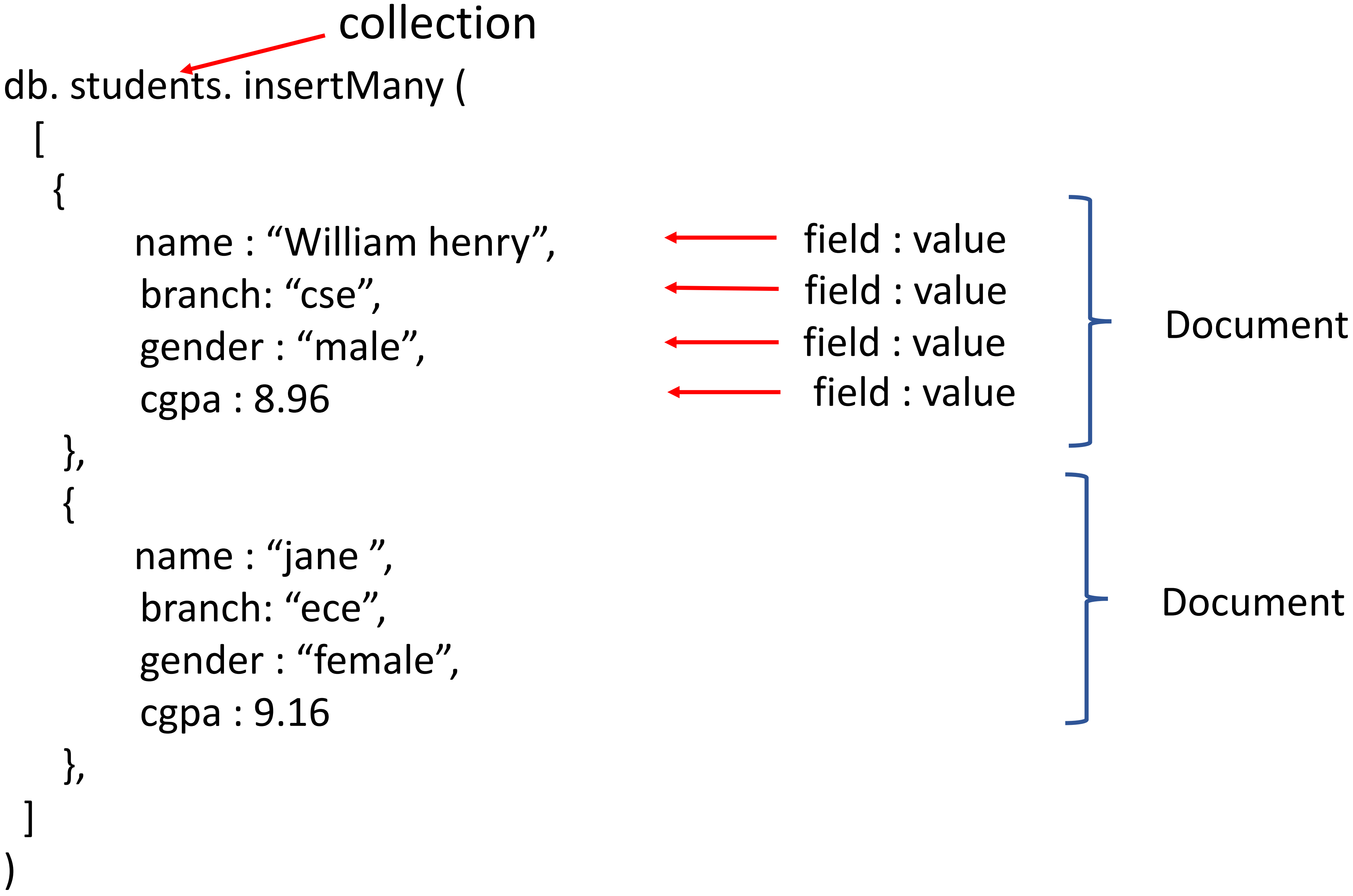
field : value

field : value

field : value

field : value

Document



collection

```
db. students. insertMany (
  [
   {
          name : “William henry”,
          branch: “cse”,
          gender : “male”,
          cgpa : 8.96
    },
    {
          name : “jane ”,
          branch: “ece”,
          gender : “female”,
          cgpa : 9.16
    },
  ]
)
```

field : value

field : value

field : value

field : value

Document

Document

Read Operation

To display all documents(records) in the collection(table)

MongoDB	SQL
db.students.find({})	SELECT * FROM students;
db.students.find({branch:”cse”})	SELECT * FROM students WHERE branch= “ cse ”

db.students.find(
{
branch:”cse”,
cgpa:{$gt:9.00}
}
)
	SELECT * FROM students WHERE branch= “ cse ” and cgpa > 9.00

db.students.find(
{
branch:”cse”,
cgpa:{$gt:9.00}
},
{
rollnumber:1, name: 1(true), cgpa:1
}
)
	SELECT rollnumber , name, cgpa FROM students
	WHERE branch= “ cse ” and cgpa > 9.00

updateOne(filter, update)

updateMany(filter, update) - returns **Promise**

The filter used to select the document to update

The update operations to be applied to the document

```
db.students.updateOne(
        {rollno:'20501A1225’},          ← filter
         {$set:{cgpa: 9.31}}            ← update
)
```

```
db.students.updateMany(
                {
                   gender:'female’,
                   discount:{$exists:false}     ← filter
                 },
                {$set:{discount:50}}            ← update
 )
```

```
const {MongoClient}=require('mongodb');
const uri='mongodb://localhost:27017/'

async function main(){
    // create an client instance(or connection object) using MongoClient
    const client= new MongoClient(uri);
    try{
        // connect the client to database
        await client.connect();
        console.log('mongodb connected successfully');
        await getdatabases(client);
    }catch(e){
        console.log(e)
    }finally{
        client.close()
    }
}
main().catch(console.error)
async function getdatabases(client){
    const databaseList= await client.db().admin().listDatabases();
    console.log("databases")
    databaseList.databases.forEach(db=>{
        console.log(`-${db.name}`);
    })
}
```

Returns the promise object

Specifies to wait for the event to happen