Introduction to MongoDB

A NoSQL, document-oriented database

What is MongoDB

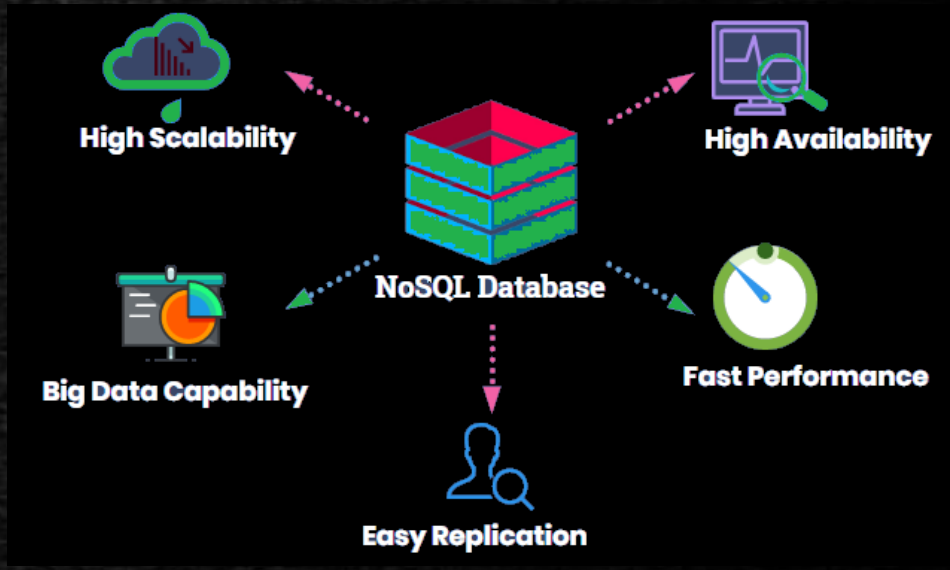
- MongoDB is an open-source document database.
- It provides high performance, high availability, and automatic scaling.

- <u>Features of MongoDB</u>

  - Light Weight and Easy to use
  - Very faster than typical RDBMS (about 100x faster)
  - Uses internal memory for storing working sets which makes it fast.
  - No complex joins required in MongoDB. It uses deep queries
  - Very easy to scale up or down based on requirement.

Usage Areas of MongoDB

- <u>MongoDB is best to be used in scenarios like :</u>
  - More customers are going online and DB needs to be scaled up easily.
  - Attributes can be added on demand without affecting other apps and services
  - If availability should be 24 hours a day, 7 days a week
  - Supporting continuous streams of real-time data
  - Dealing customer generated semi-structured/unstructured data

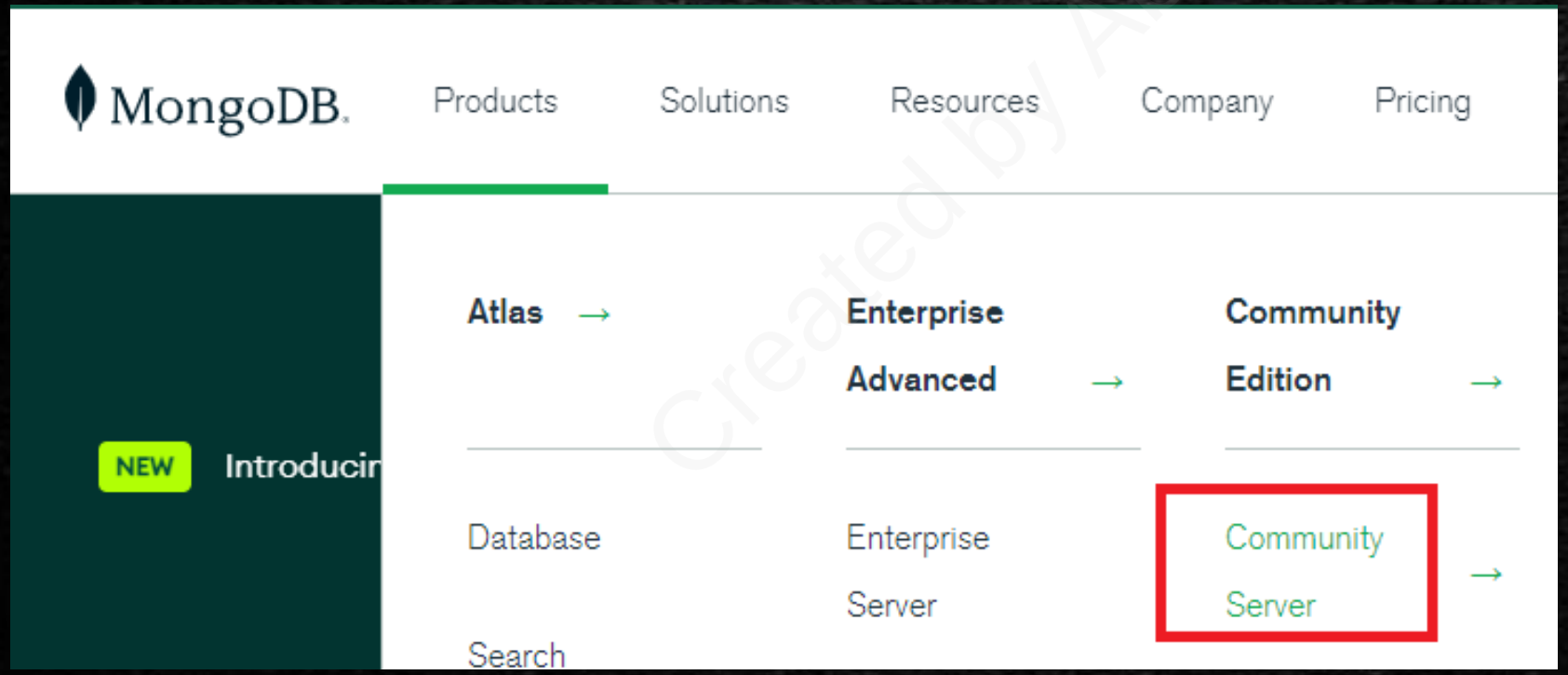


Installing MongoDB

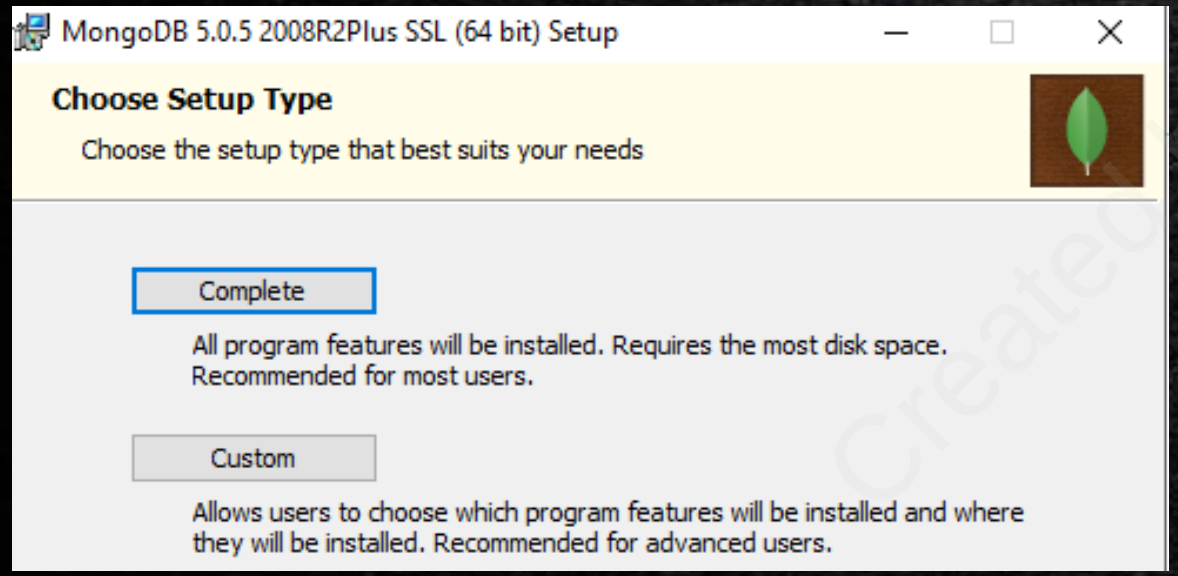
Install MongoDB Server, Compass and Mongo Shell

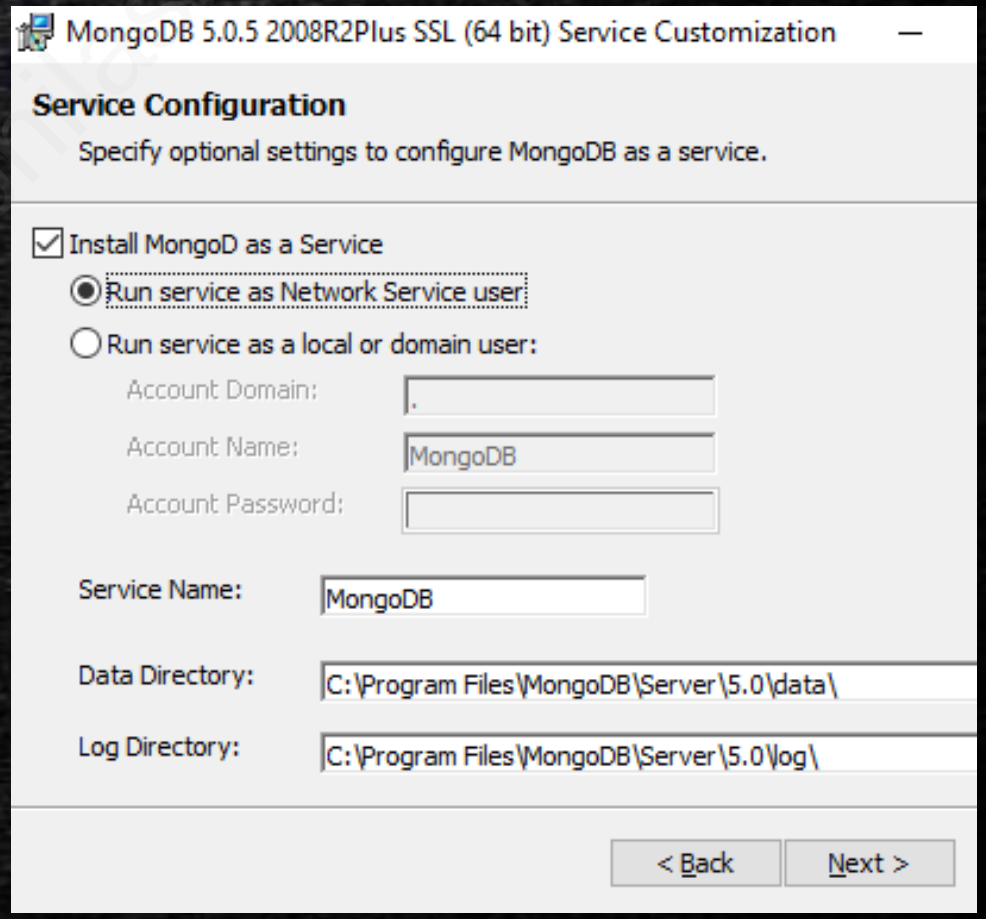
Installing MongoDB Community Server

- Go to www.mongodb.com
- From products select Community Server and download it.

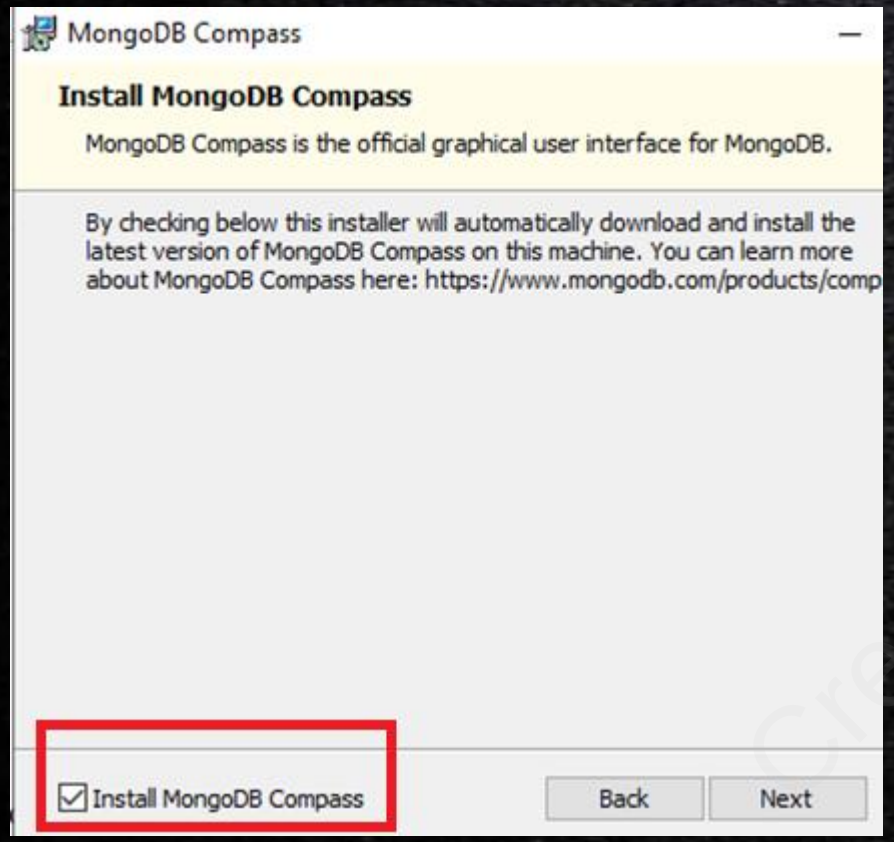


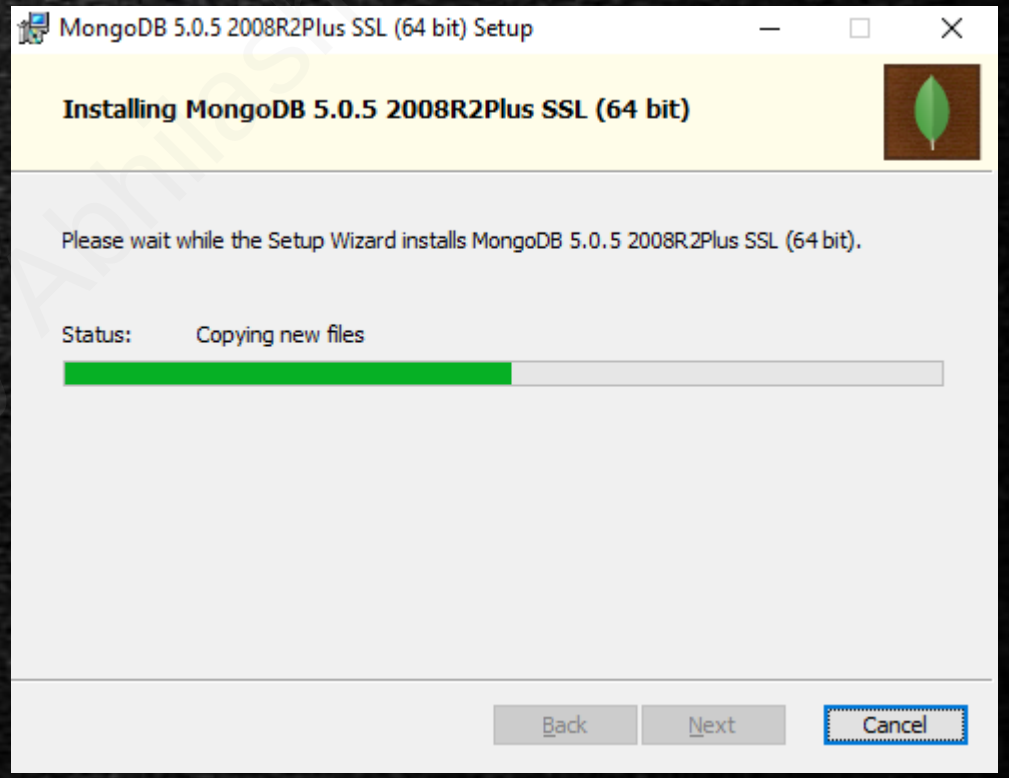
Installing MongoDB Community Server



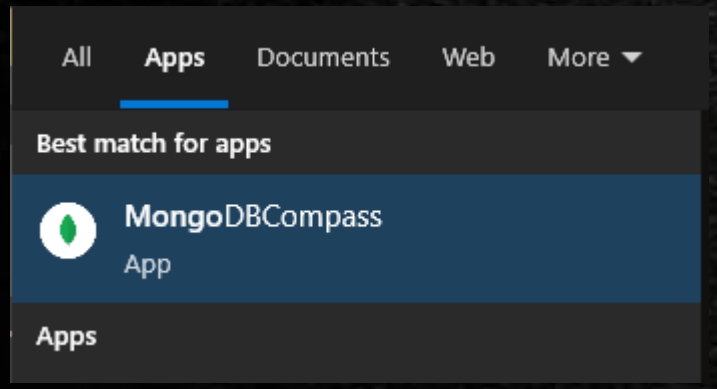


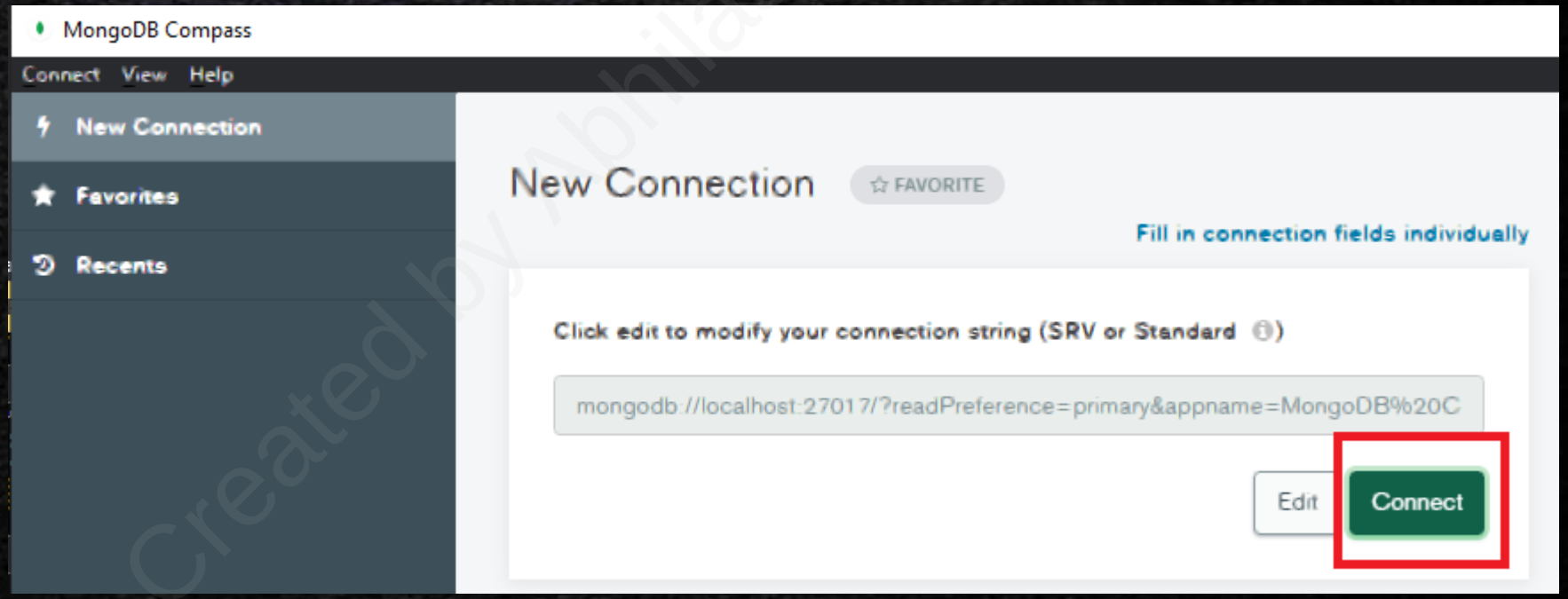
Installing MongoDB Community Server





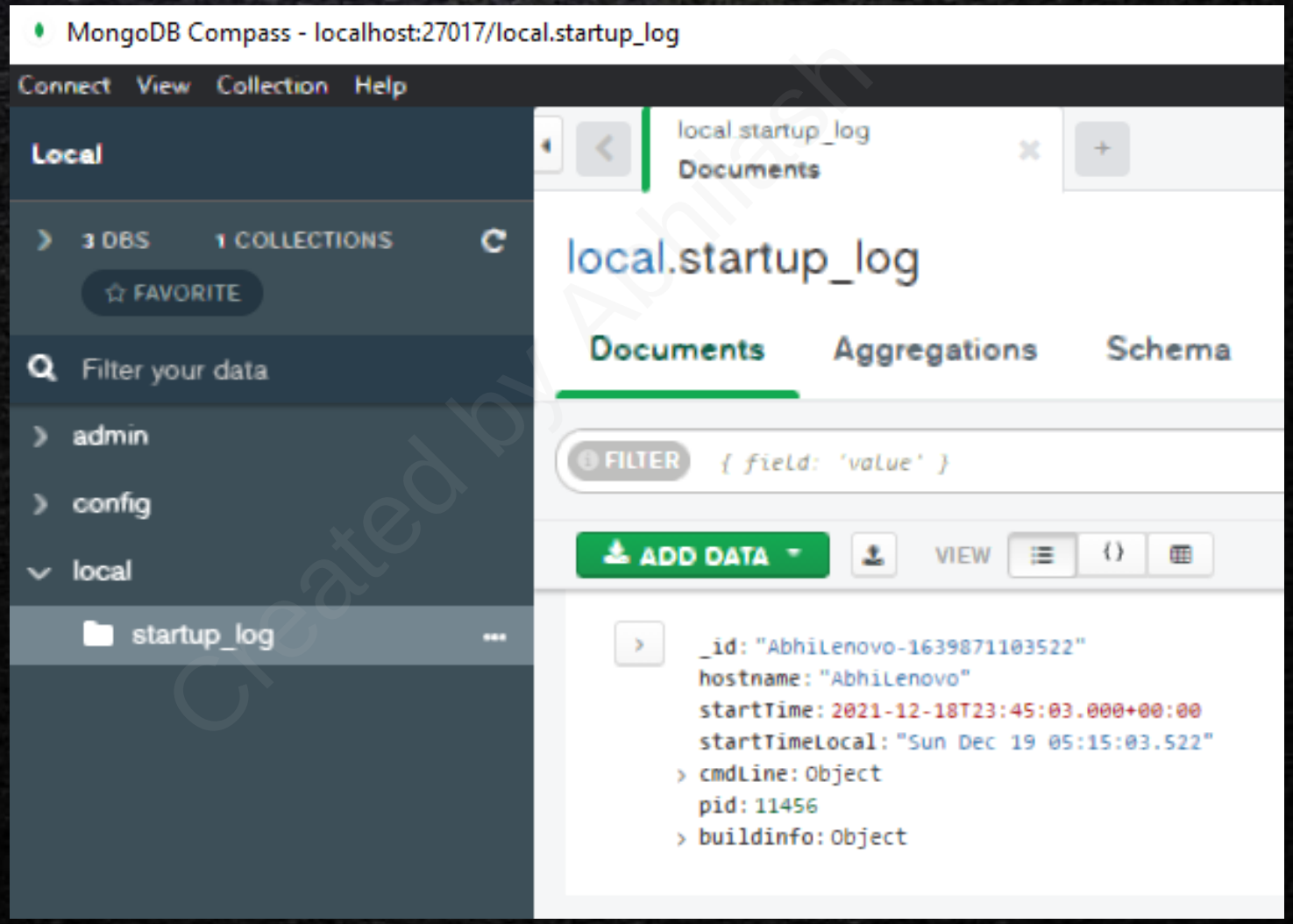
Connecting to MongoDB Server from Compass





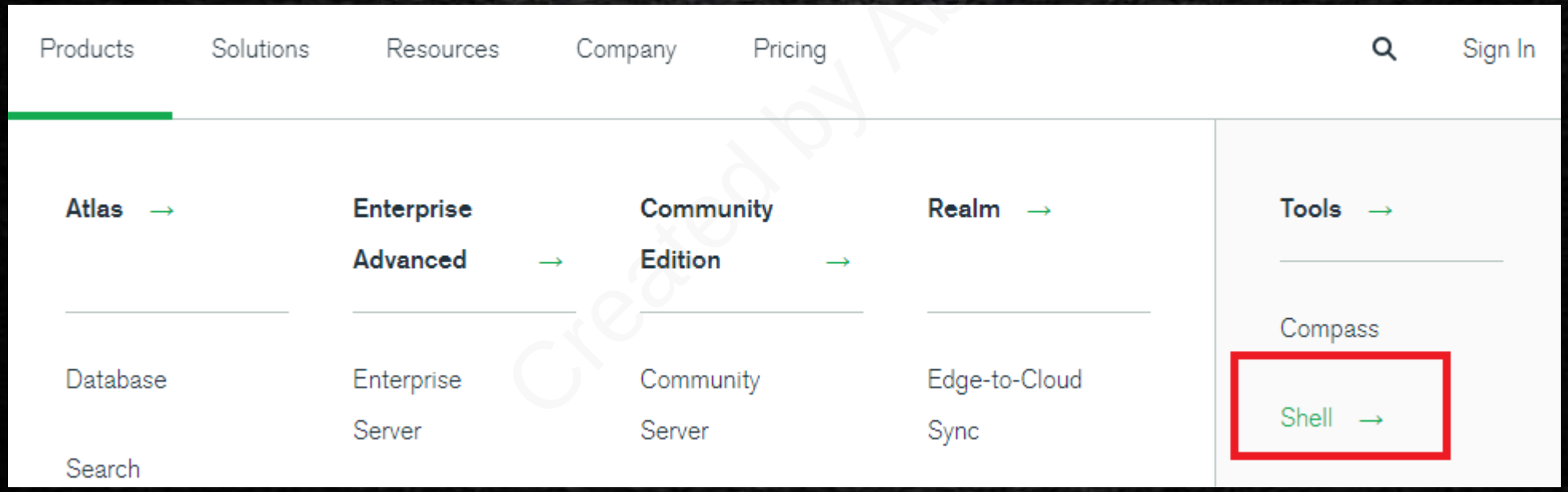
By Default MongoDB instance will be running on your localhost with default port 27017:

Connecting to MongoDB Server from Compass

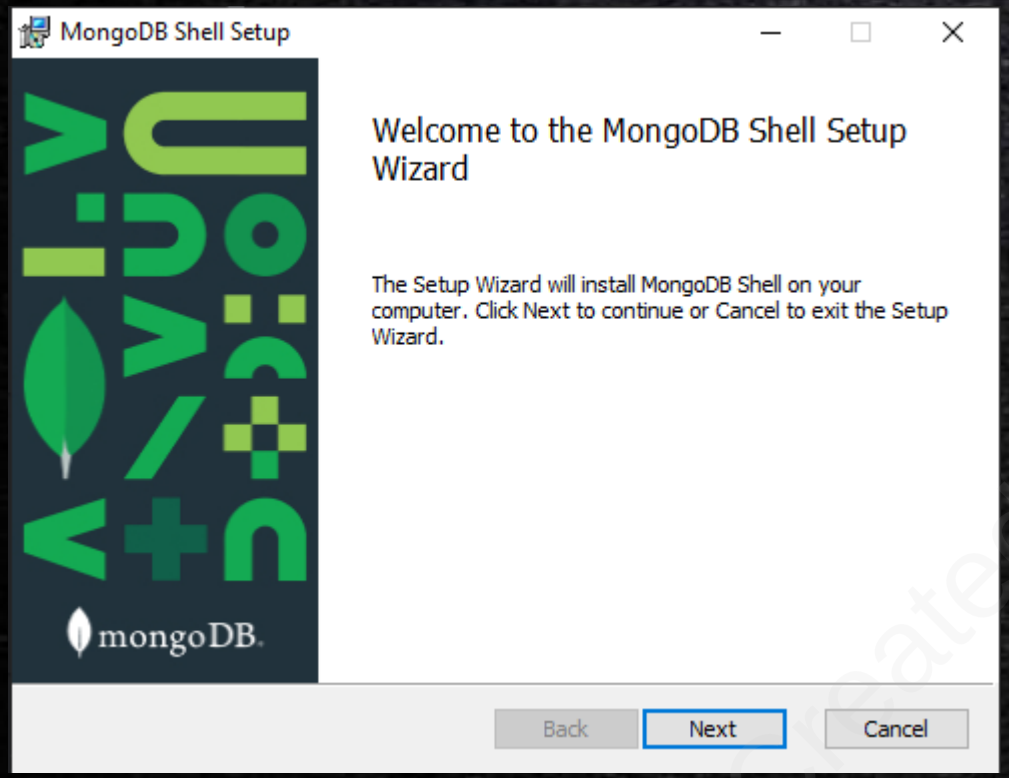


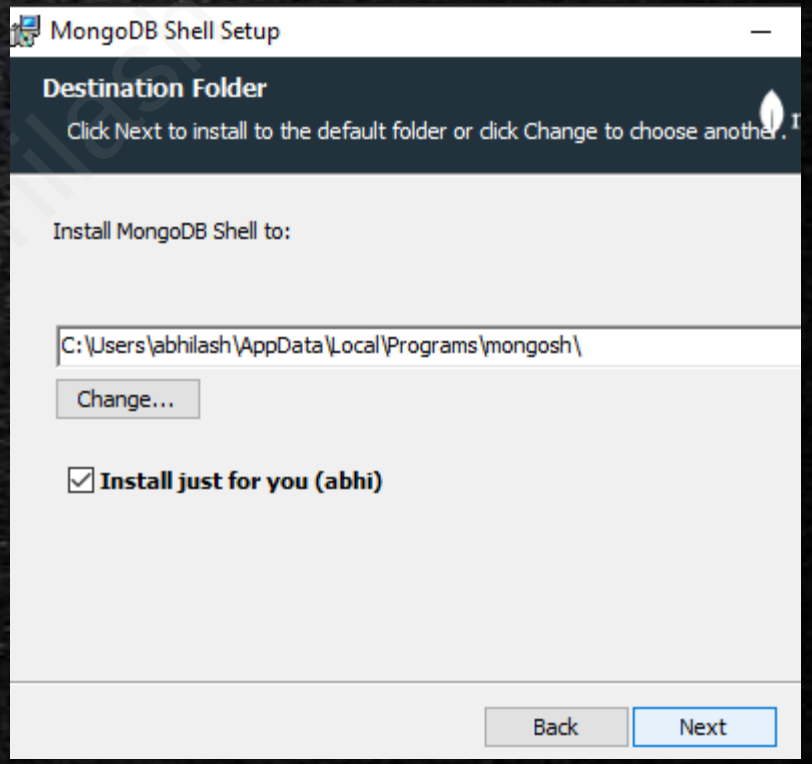
Installing MongoDB Shell

- Go to www.mongodb.com
- From tools select ‘Shell’ and download it.



Installing MongoDB Shell





Connecting to MongoDB Server from Shell

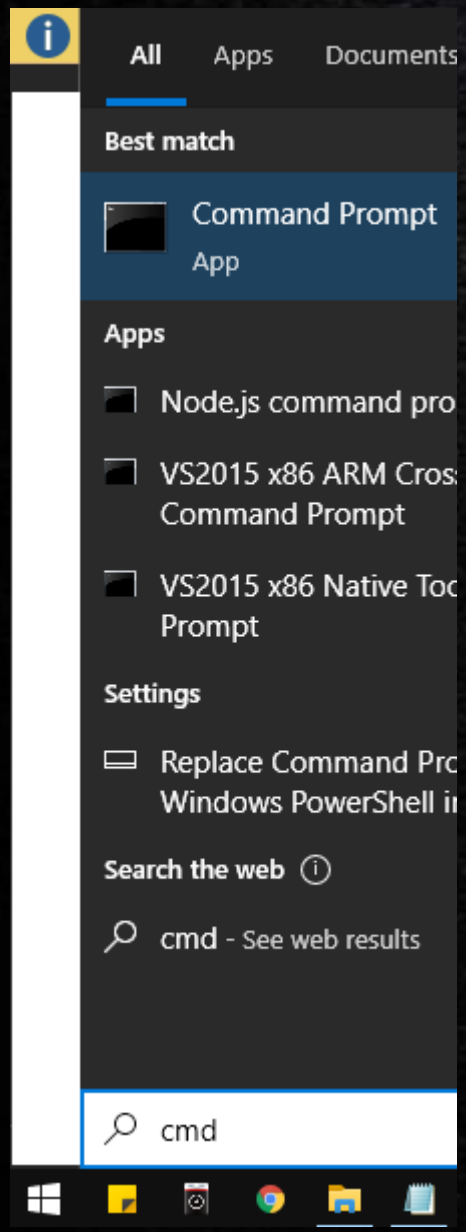
Run mongosh without any command-line options to connect to a MongoDB instance running on your localhost with default port 27017:

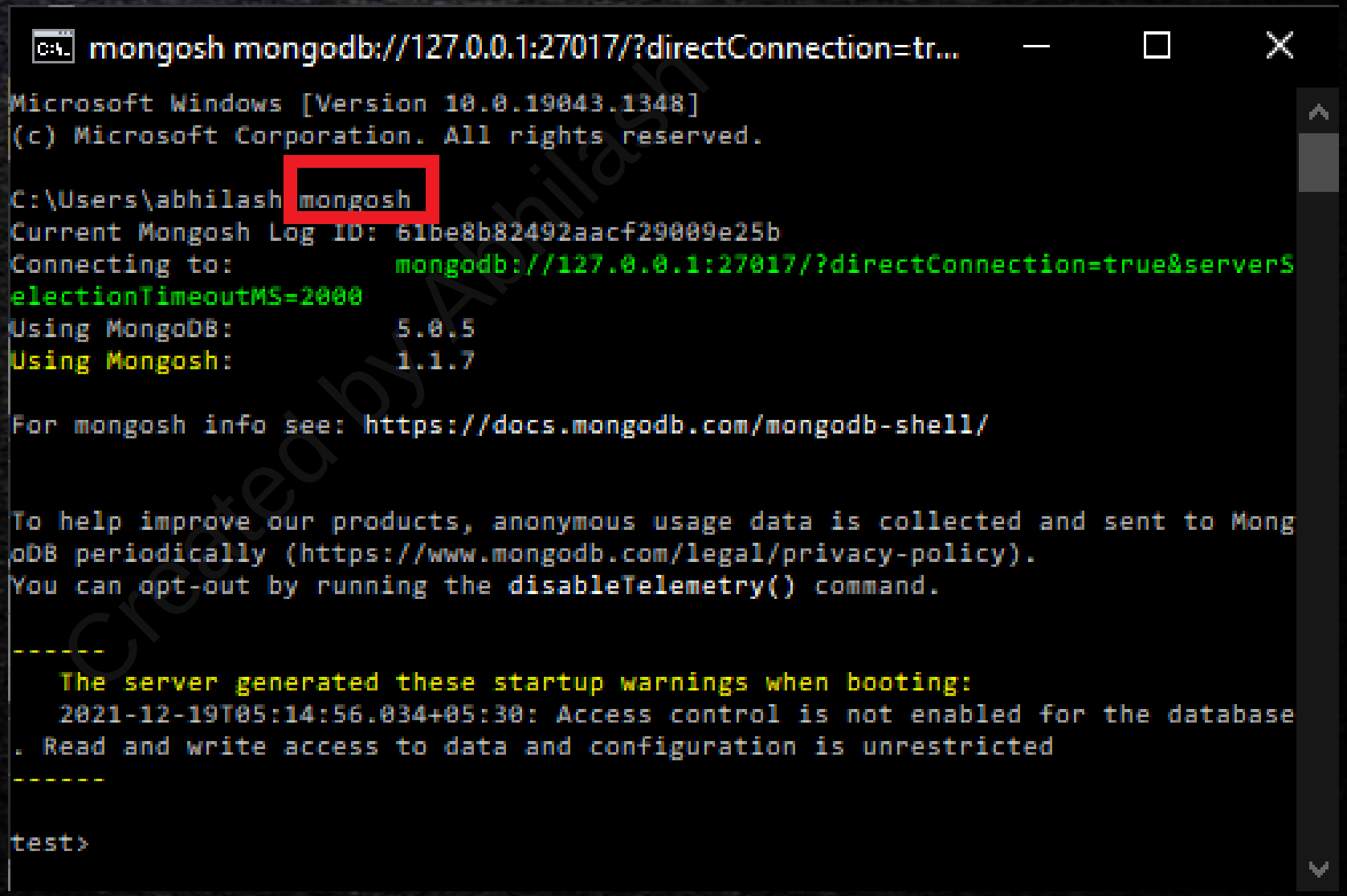
```
mongosh
```

This is equivalent to the following command:

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

Connecting to MongoDB Server from Shell





MongoDB : Database Operations

Create, List and Delete Databases in MongoDB

mongoDB®

Creating New Database in MongoDB Shell

- There no ‘create database’ command.
- Just issue `use DATABASE_NAME` , if exists will use it, else it will create.

```
test> use training
switched to db training
training>
```

Inserting data into DB

- At least one data document (like ‘row’ in rdb) should be there so that we can list the database

- The syntax to create a new collection is
  `db.dbname.insertOne({"key":"value"})`

```
training> db.training.insertOne({"name":"Tom"})
{
  acknowledged: true,
  insertedId: ObjectId("61bf00c7265f18646c7d9dbc")
}
training>
```

List Databases

- Use show dbs, to list the databases available

```
training> show dbs
admin       135 kB
config    73.7 kB
local     73.7 kB
training  8.19 kB
training>
```

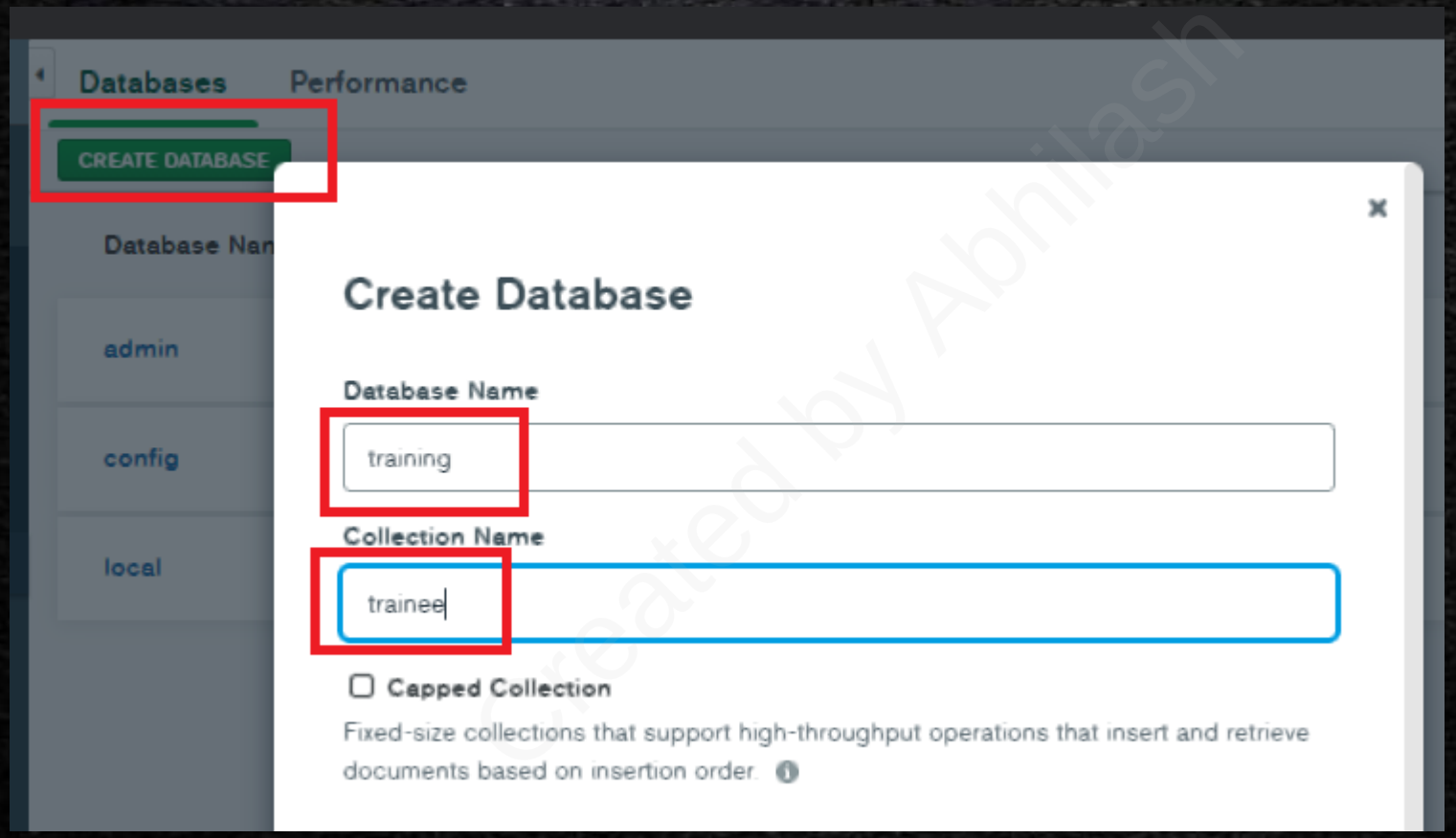
Drop Database

- Use db.dropDatabase(), to delete the current database in use

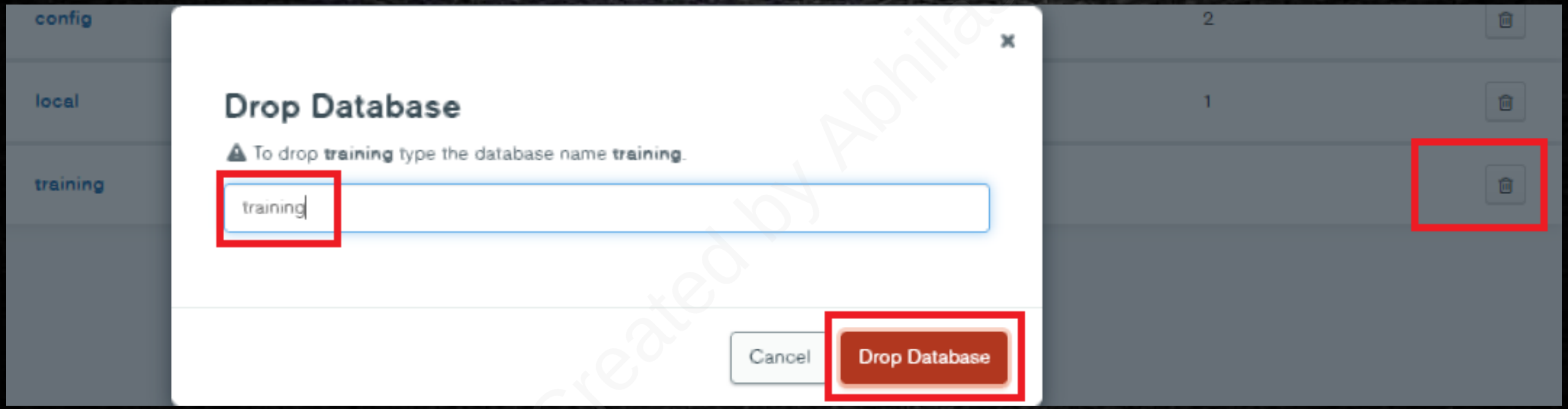
```
training> db.dropDatabase()
{ ok: 1, dropped: 'training' }
training> show dbs
admin     135 kB
config    111 kB
local    73.7 kB
training> exit

C:\WINDOWS\system32>
```

Create Database using Compass



Drop Database using Compass



MongoDB : Collections

Create, List and Remove Collections

Creating & Listing Collections in the DB

- The syntax to create a new collection is `db.createCollection(name, options)`

```
training> db.createCollection("trainee")
{ ok: 1 }
training> show collections
trainee
training>
```

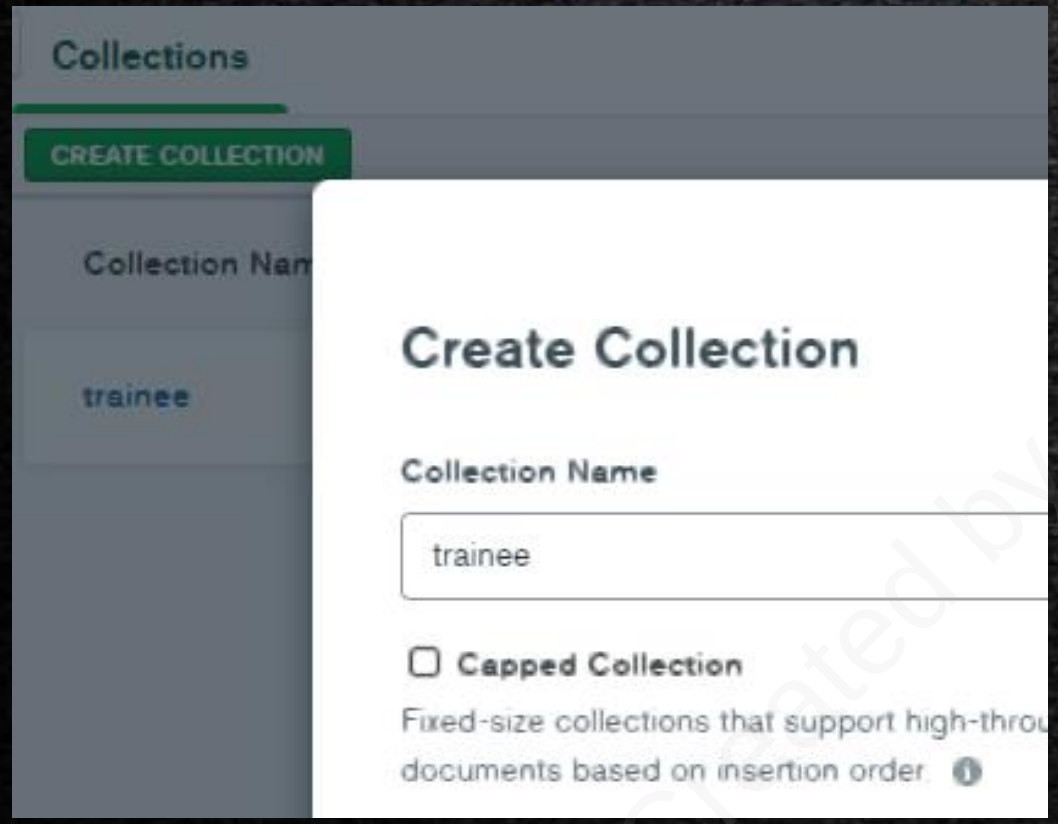
Removing a Collection from the DB

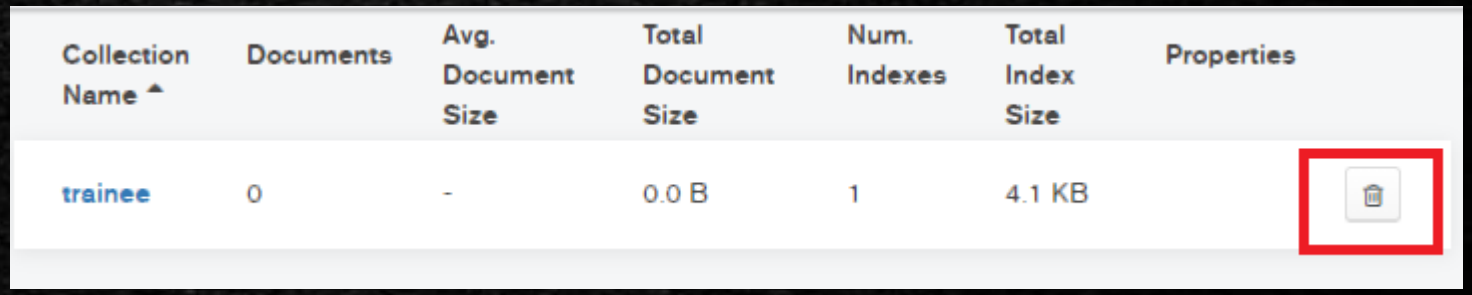
- The syntax to drop a collection is

```
db.COLLECTION_NAME.drop()
```

```
training> db.trainee.drop()
true
training>
```

Creating and Deleting Collection using compass





Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
trainee	0	-	0.0 B	1	4.1 KB	

Options When Creating a Collection

Field	Type	Description
Capped	Boolean	(Optional) If it is set to true, enables a capped collection. Capped collection is a fixed size collecction that automatically overwrites its oldest entries when it reaches its maximum size. If you specify true, you need to specify size parameter also.
AutoIndexID	Boolean	(Optional) If it is set to true, automatically create index on ID field. Its default value is false.
Size	Number	(Optional) It specifies a maximum size in bytes for a capped collection. Ifcapped is true, then you need to specify this field also.
Max	Number	(Optional) It specifies the maximum number of documents allowed in the capped collection.

Options When Creating a Collection

```
db.createCollection("trainee",
{ capped : true,
size : 6142800,
max : 10000
})
```

Collection gets automatically created when adding document

- The syntax to create a new document is
  `db.collectionName.insertOne({document})`

We can use insertOne, insertMany, or bulkWrite.

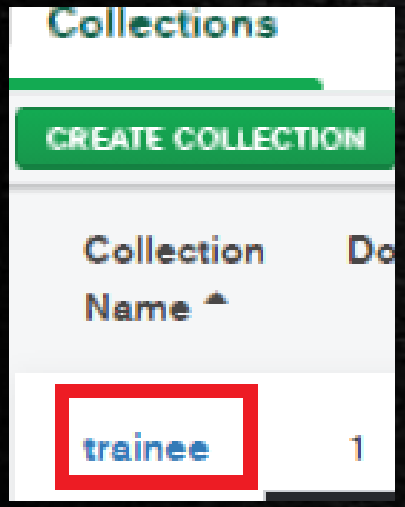
```
training> db.trainee.insertOne({ "admission_no": 5, "name": "Tom" })
{
  acknowledged: true,
  insertedId: ObjectId("61bf12fd9e8329e5e559f9b4")
}
```

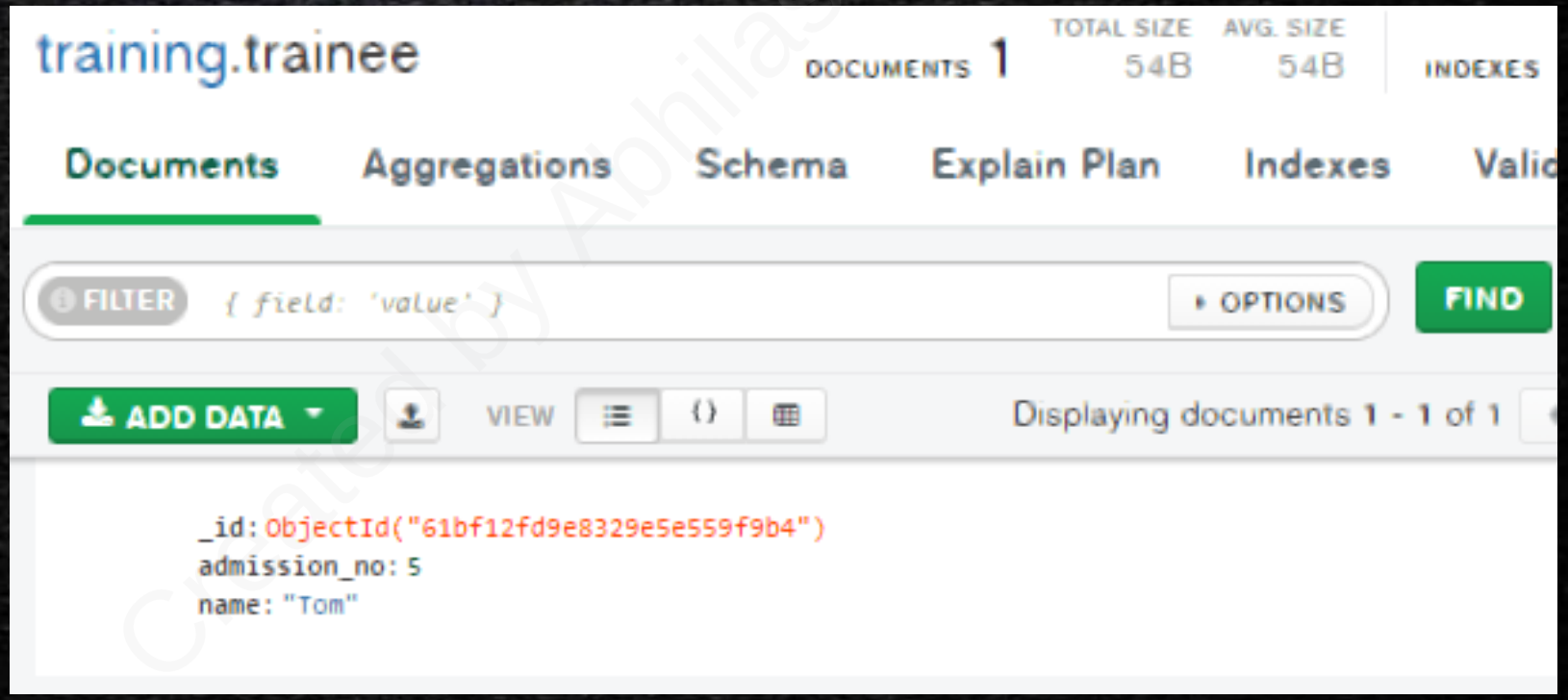
List all documents in collection

- The syntax to get the details of documents in collection is `db.collectionName.find()`

```
training> db.trainee.find()
[
  {
    _id: ObjectId("61bf12fd9e8329e5e559f9b4"),
    admission_no: 5,
    name: 'Tom'
  }
]
training>
```

List all documents in collection using compass





MongoDB : Basic CRUD Operations

Create, Read, Update, and Delete documents

JSON Format

- JSON, or JavaScript Object Notation, is the wildly popular standard for data interchange on the web.
- It was defined as part of the JavaScript language in the early 2000s by JavaScript creator Douglas Crockford.
- It was in 2013 that the format was officially specified.

JSON Objects Format

- JavaScript objects are a series of containers
- A string key is mapped to a value (which can be a number, string, function, or even another object).
- An Ideal JSON Object:
  - Starts with {} curly braces
  - Key and value in quotations
  - Key and Value pair separated by comma
  - Keys always surrounded by quotations
  - Sub documents also kept within {}
  - Arrays within []
  - Use https://jsonformatter.org/ to verify

```
{
    "admission_no" : 100,
    "first_name":"Tom",
    "last_name":"Hanks",
    "address":{
        "city":"Albany",
        "state" :"New York"
    },
    "Session":[
        {
            "name":"Morning",
            "duration":4
        },
        {
            "size":"Evenning",
            "duration":3
        }
    ],
    "course":"Neural Networks"
}
```

BSON Format

- BSON stands for “Binary JSON,”.
- BSON’s binary structure allows it to be parsed much more quickly.
- BSON has been extended to add some optional non-JSON-native data types, like dates and binary data
- Anything you represent in JSON, MongoDB stores that data in BSON format both internally, and over the network

Creating Documents using insertOne()

The syntax to create a new document is

```
db.collectionName.
insertOne({document})
```

We can use insertOne, insertMany, or bulkWrite.

```
test> use training
switched to db training
```

```
training> db.trainee.insertOne(
.....    "admission_no" : 100,
.....    "first_name":"Tom",
.....    "last_name":"Hanks",
.....    "address":{
.......        "city":"Albany",
.......        "state" :"New York"
.......    },
.....    "Session":[
.....        {
.......            "name":"Morning",
.......            "duration":4
.......        },
.....        {
.......            "size":"Evening",
.......            "duration":3
.......        }
.....    ],
.....    "course":"Neural Networks"
..... })
{
  acknowledged: true,
  insertedId: ObjectId("61bfc4eadf9d3c572c601e12")
}
training>
```

Read all documents in collection

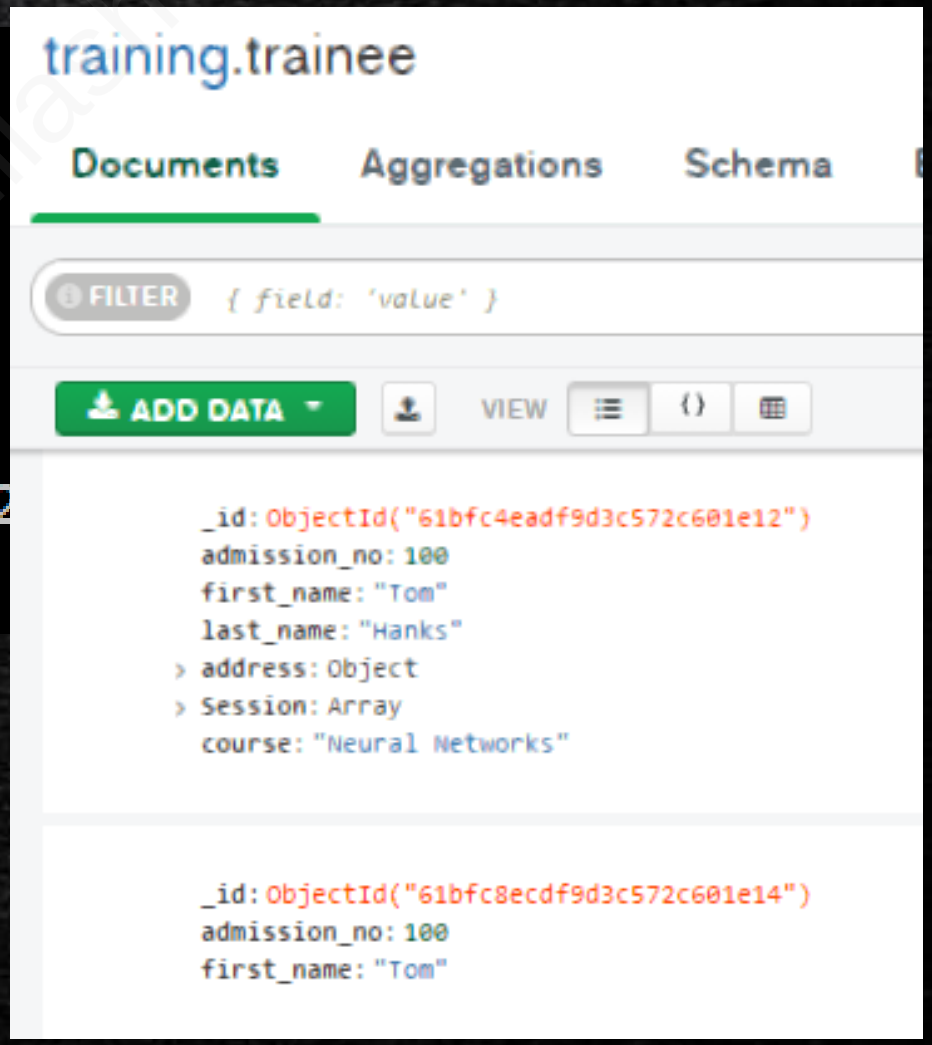
- The syntax to get the details of documents in collection is `db.collectionName.find()`

```
training> db.trainee.find()
[
  {
    _id: ObjectId("61bfc4eadf9d3c572c601e12"),
    admission_no: 100,
    first_name: 'Tom',
    last_name: 'Hanks',
    address: { city: 'Albany', state: 'New York' },
    Session: [
      { name: 'Morning', duration: 4 },
      { size: 'Evening', duration: 3 }
    ],
    course: 'Neural Networks'
  }
]
```

Insert a Document without an _id Field

mongodb creates and adds the _id field and assigns it a unique ObjectId

```
training> db.trainee.insertOne({
...     "admission_no" : 100,
...     "first_name":"Tom",
...   });
{
  acknowledged: true,
  insertedId: ObjectId("61bfc8ecdf9d3c572
}
```



Insert a Document with an _id Field

The value of _id must be unique within the collection to avoid duplicate key error.

```
training>    db.trainee.insertOne({
...     "_id" : 10,
...     "admission_no" : 100,
...     "first_name":"Tom",
...   });
{ acknowledged: true, insertedId: 10 }
```

```
training>    db.trainee.insertOne({
...     "_id" : 10,
...     "admission_no" : 100,
...     "first_name":"Tom",
...   });
MongoServerError: E11000 duplicate key error
 _id_ dup key: { _id: 10 }
```

Creating Documents using insertMany()

The insertMany() method is used to insert more than one document at once into a collection. It has the following syntax:

```
db.collection.insertMany(
    [ <document 1> , <document 2>, ... ],
    {
      ordered: <boolean>
    }
)
```

ordered is an optional boolean specifying whether should perform an ordered or unordered insert. Defaults to true.

Creating Documents using insertMany()

```
training>    db.trainee.insertMany([
...         {"admission_no" : 101, "first_name":"James"},
...         {"admission_no" : 102, "first_name":"Rock"},
...         {"admission_no" : 103, "first_name":"Tony"}
...     ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("61bfcdd9df9d3c572c601e18"),
    '1': ObjectId("61bfcdd9df9d3c572c601e19"),
    '2': ObjectId("61bfcdd9df9d3c572c601e1a")
  }
}
training>
```