

Introduction to MongoDB

A NoSQL, document-oriented database



mongoDB®

What is MongoDB

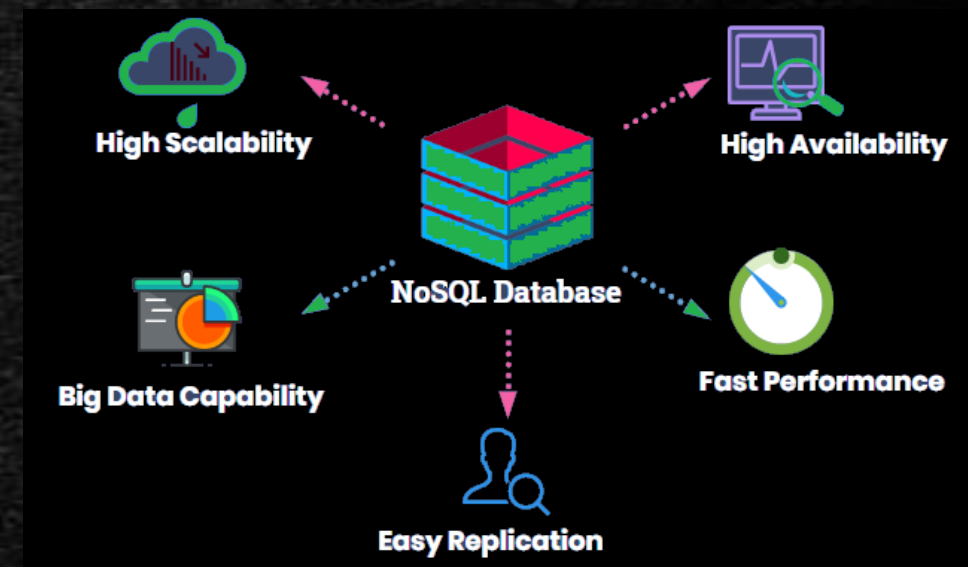


- MongoDB is an open-source document database.
- It provides high performance, high availability, and automatic scaling.
- Features of MongoDB
 - Light Weight and Easy to use
 - Very faster than typical RDBMS (about 100x faster)
 - Uses internal memory for storing working sets which makes it fast.
 - No complex joins required in MongoDB. It uses deep queries
 - Very easy to scale up or down based on requirement.

Usage Areas of MongoDB



- MongoDB is best to be used in scenarios like :
 - 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

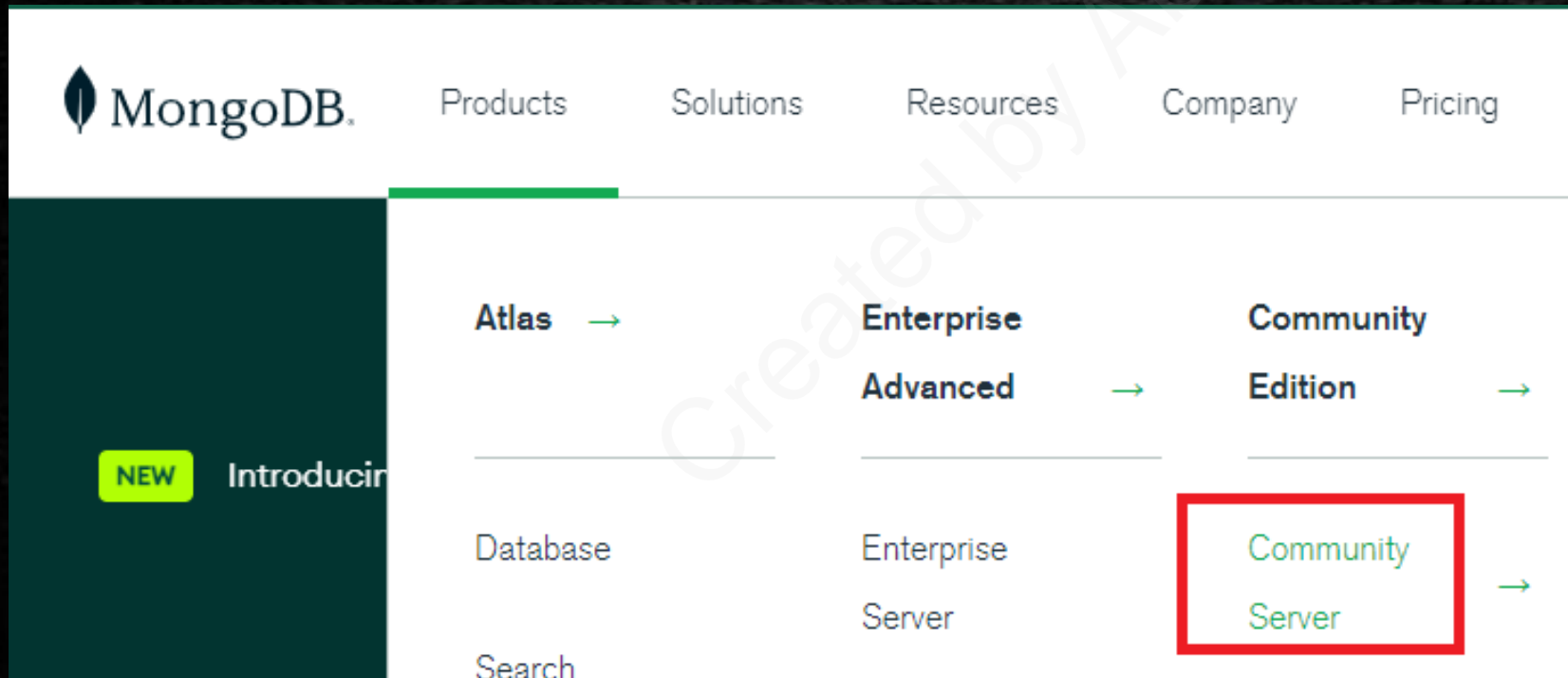


mongoDB®

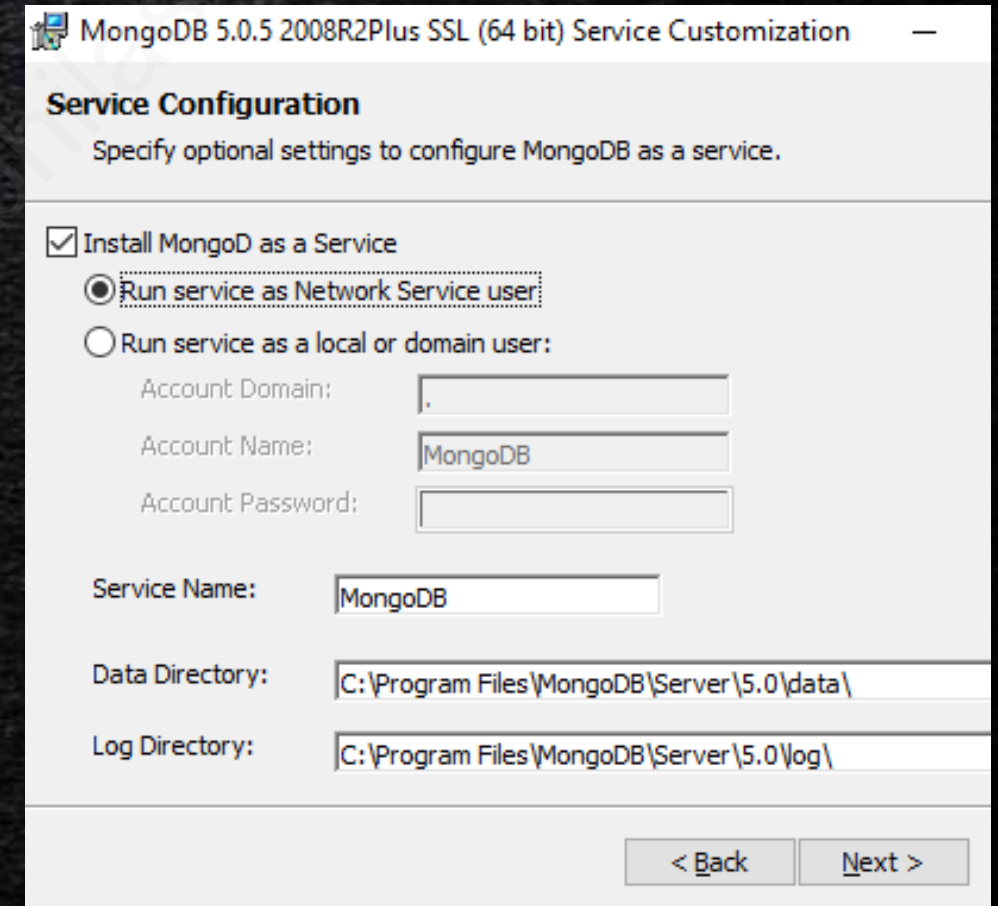
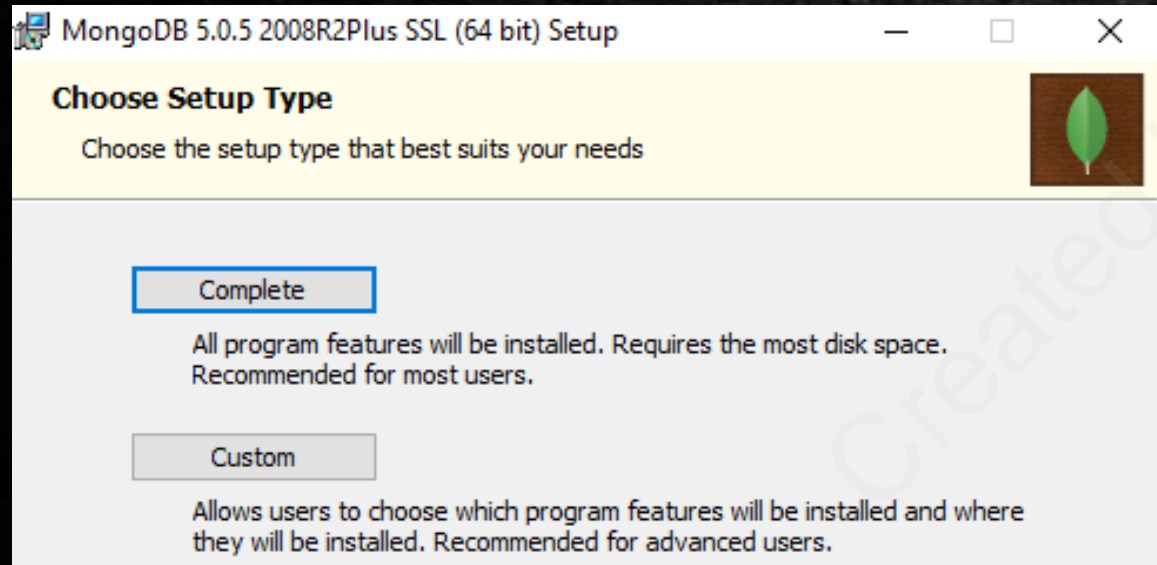
Installing MongoDB Community Server



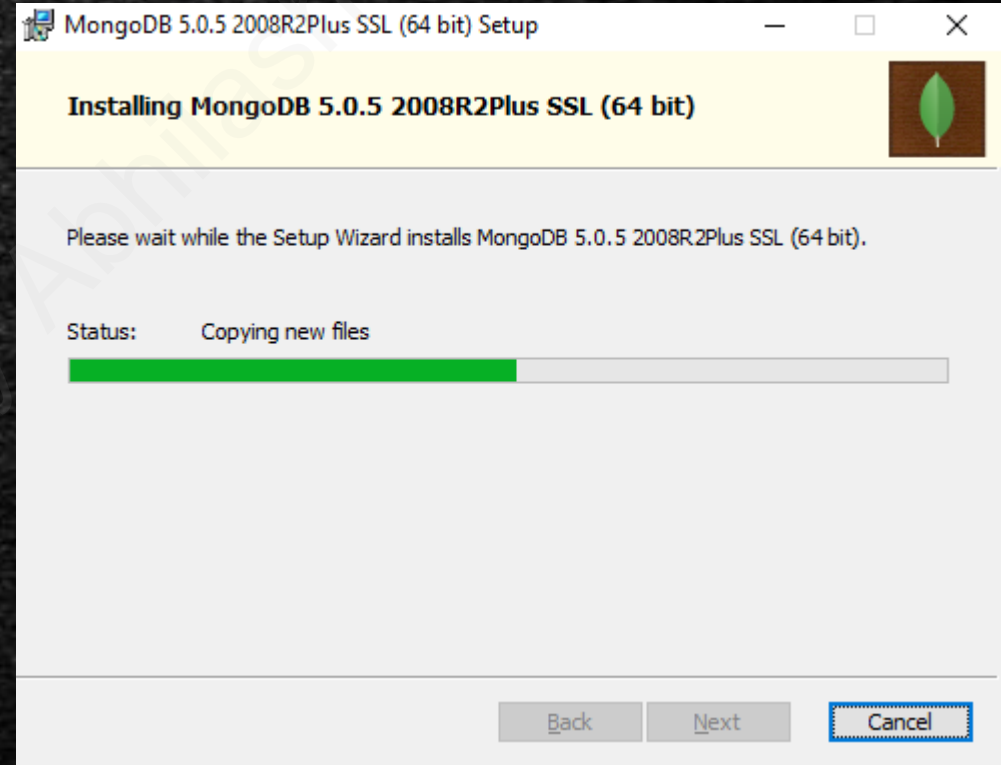
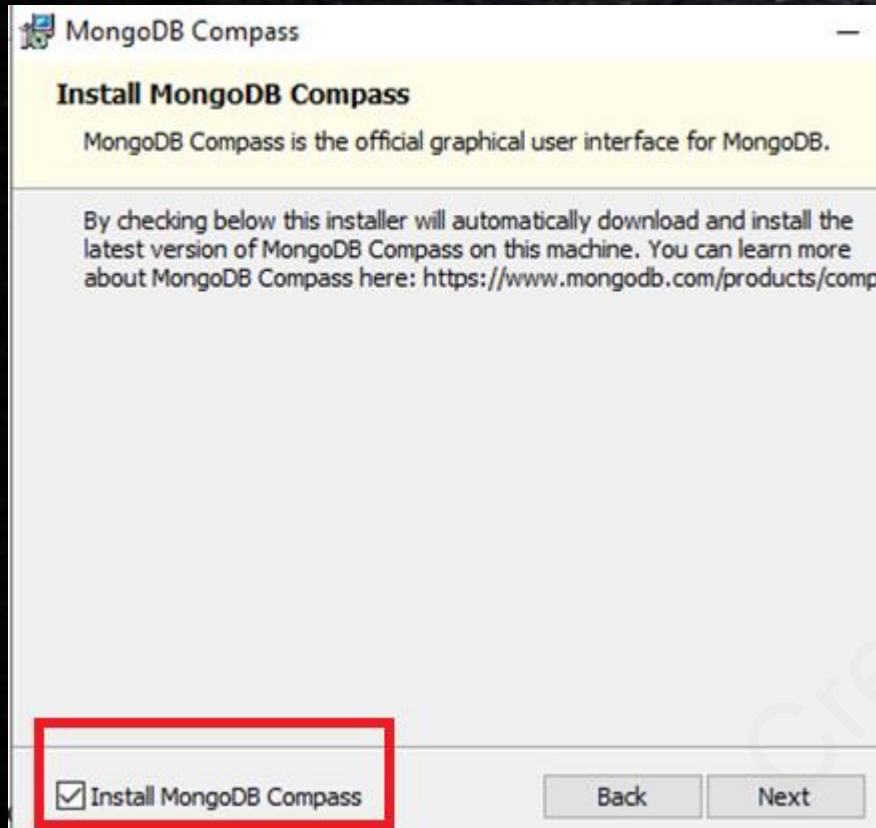
- Go to www.mongodb.com
- From products select Community Server and download it.



Installing MongoDB Community Server



Installing MongoDB Community Server



Connecting to MongoDB Server from Compass

The screenshot shows the MongoDB Compass application window. On the left, there's a sidebar with a menu: 'All', 'Apps' (selected), 'Documents', 'Web', and 'More'. Below the menu, it says 'Best match for apps' and lists 'MongoDBCompass App'. The main area is titled 'New Connection' and has a 'FAVORITE' button. It includes a text input field with the connection string 'mongodb://localhost:27017/?readPreference=primary&appName=MongoDB%20C'. At the bottom right, there are 'Edit' and 'Connect' buttons. The 'Connect' button is highlighted with a red rectangle. A watermark 'Created by Abhishan' is visible across the center of the interface.

MongoDB Compass

Connect View Help

New Connection

★ Favorites

🕒 Recents

New Connection ☆ FAVORITE

Fill in connection fields individually

Click edit to modify your connection string (SRV or Standard ⓘ)

mongodb://localhost:27017/?readPreference=primary&appName=MongoDB%20C

Edit Connect

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

Connecting to MongoDB Server from Compass



MongoDB Compass - localhost:27017/local.startup_log

Connect View Collection Help

Local

3 DBS 1 COLLECTIONS

☆ FAVORITE

Filter your data

admin

config

local

startup_log

local.startup_log Documents

local.startup_log

Documents Aggregations Schema

FILTER { field: 'value' }

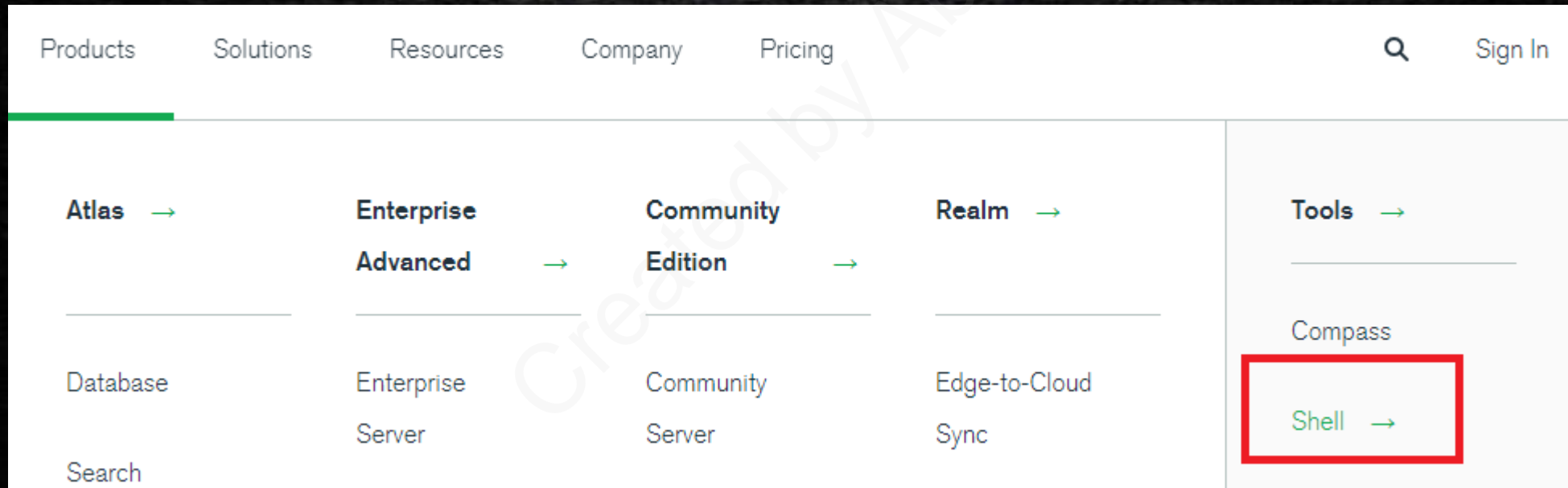
ADD DATA VIEW

```
> {
  "_id": "AbhiLenovo-1639871103522",
  "hostname": "AbhiLenovo",
  "startTime": "2021-12-18T23:45:03.000+00:00",
  "startTimeLocal": "Sun Dec 19 05:15:03.522",
  "cmdLine": Object,
  "pid": 11456,
  "buildInfo": Object
}
```

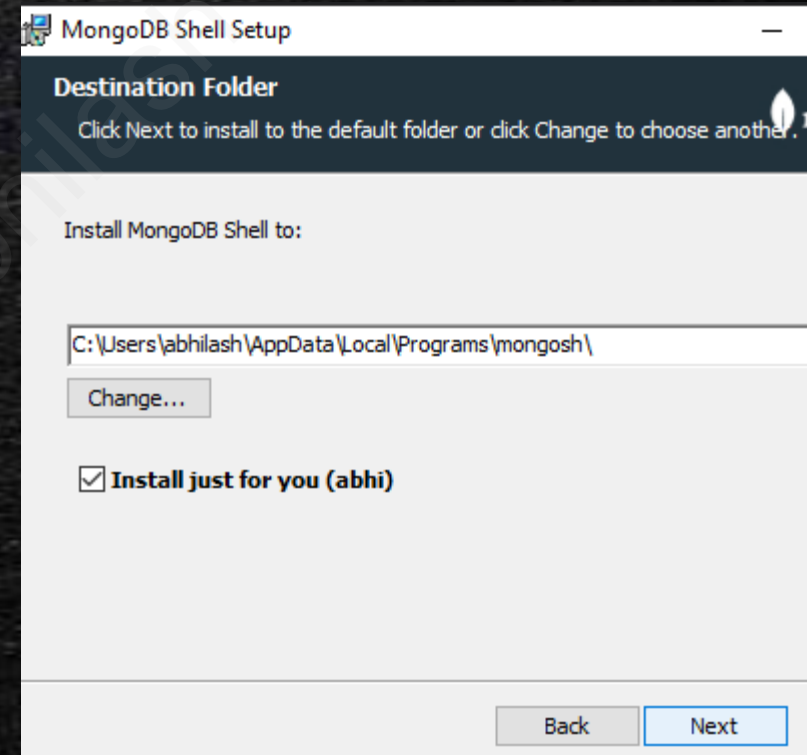
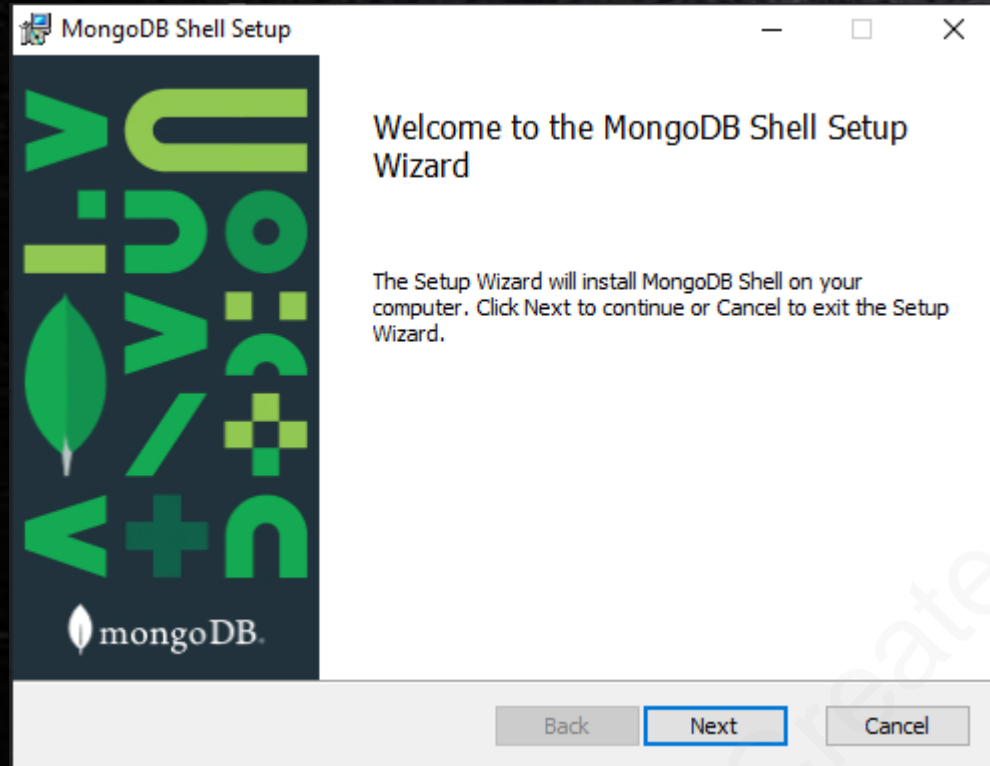
Installing MongoDB Shell



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



Installing MongoDB Shell



Connecting to MongoDB Server from Shell



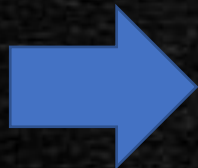
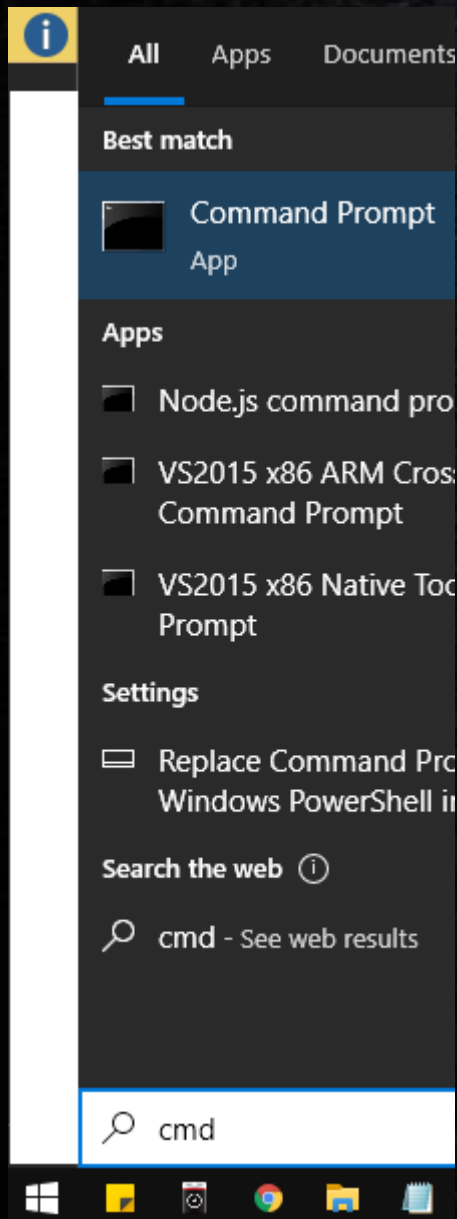
Run **mongosh** without any command-line options to connect to a MongoDB instance running on your localhost with default port 27017:

```
mongosh
```

This is equivalent to the following command:

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

Connecting to MongoDB Server from Shell



```
cmd mongosh mongodb://127.0.0.1:27017/?directConnection=tr...
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\abhilash mongosh
Current Mongosh Log ID: 61be8b82492aacf29009e25b
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverS
electionTimeoutMS=2000
Using MongoDB:      5.0.5
Using Mongosh:       1.1.7

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

To help improve our products, anonymous usage data is collected and sent to Mong
oDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting:
2021-12-19T05:14:56.034+05:30: Access control is not enabled for the database
. Read and write access to data and configuration is unrestricted
-----

test>
```

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



The screenshot shows the MongoDB Compass interface. The 'Databases' tab is selected, and the 'CREATE DATABASE' button is highlighted with a red box. A modal dialog titled 'Create Database' is open. Inside the dialog, the 'Database Name' field contains 'training' and the 'Collection Name' field contains 'trainee', both highlighted with red boxes. Below these fields, there is an unchecked checkbox labeled 'Capped Collection' with a description: 'Fixed-size collections that support high-throughput operations that insert and retrieve documents based on insertion order. ⓘ'.

Create Database

Database Name
training

Collection Name
trainee

☐ Capped Collection
Fixed-size collections that support high-throughput operations that insert and retrieve documents based on insertion order. ⓘ

Drop Database using Compass



config

local

training

Drop Database

⚠ To drop **training** type the database name **training**.

Cancel Drop Database

| | |
|---|--|
| 2 | |
| 1 | |
| | |

MongoDB : Collections

Create, List and Remove Collections



mongoDB®

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



Collections


CREATE COLLECTION

Collection Name

trainee

☐ **Capped Collection**

Fixed-size collections that support high-throughput documents based on insertion order. ⓘ

| 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 collection 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. If capped 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  
  })
```

Created by Abhilash

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")
}
```

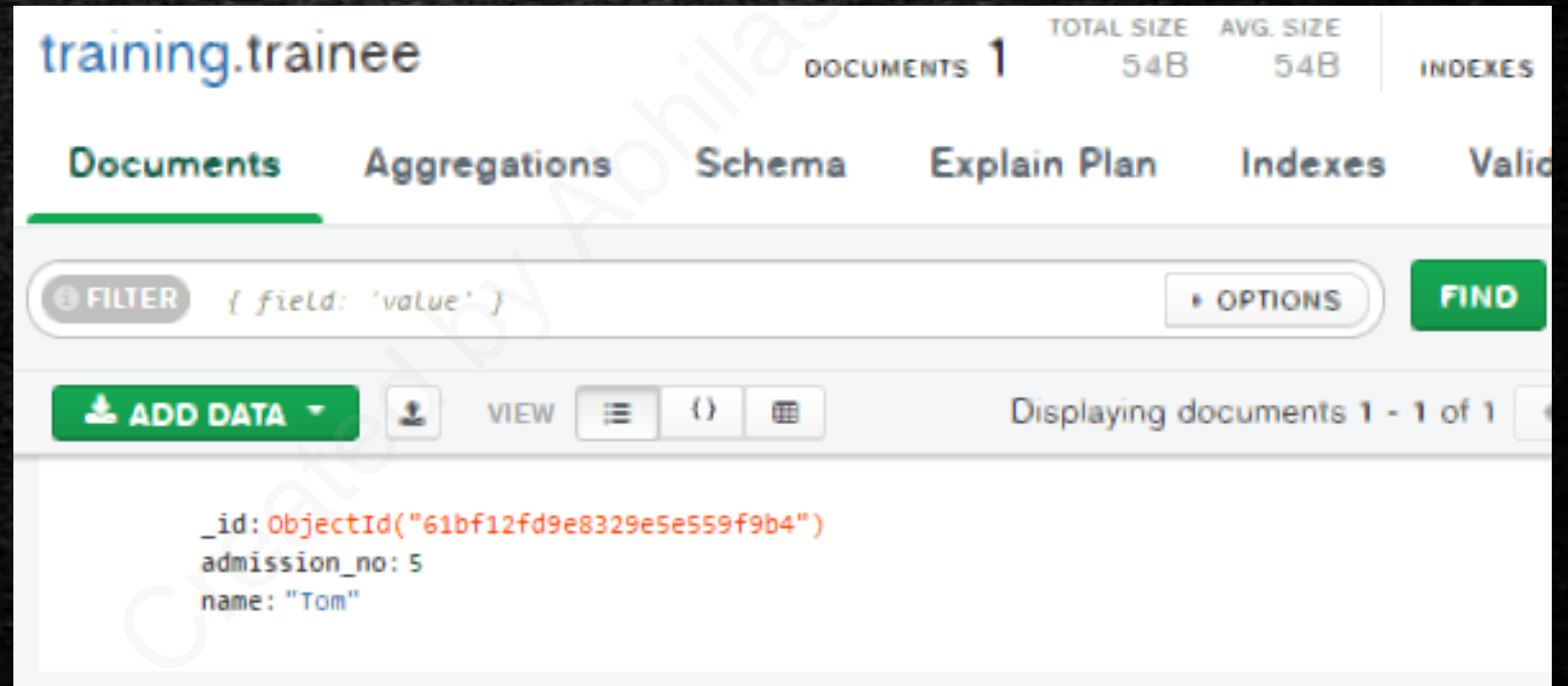
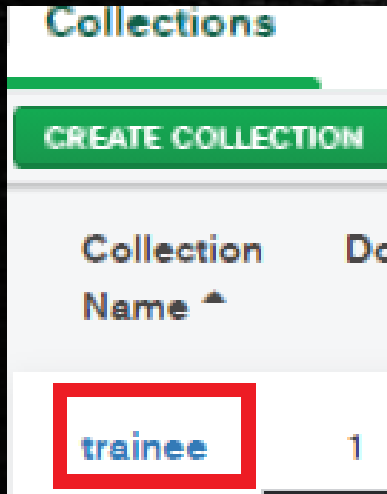
List all documents in collection



- 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



mongoDB®

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
    },
    {
      "name": "Evening",
      "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("61bfc8ecdf9d3c572c601e14")
}
```

training.trainee

Documents Aggregations Schema

FILTER { field: 'value' }

ADD DATA VIEW

```
_id: ObjectId("61bfc4eADF9d3c572c601e12")
admission_no: 100
first_name: "Tom"
last_name: "Hanks"
> address: Object
> Session: Array
  course: "Neural Networks"
```

```
_id: ObjectId("61bfc8ecdf9d3c572c601e14")
admission_no: 100
first_name: "Tom"
```

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> _
```