

# MongoDB Complete Guide

Develop Strong Understanding of Administering MongoDB, CRUD Operations, MongoDB Commands, MongoDB Compass, MongoDB Server, MongoDB Replication and MongoDB Sharding

MANU SHARMA



# MongoDB

# Complete

# Guide

Develop Strong Understanding of Administering MongoDB, CRUD Operations, MongoDB Commands, MongoDB Compass, MongoDB Server, MongoDB Replication and MongoDB Sharding

MANU SHARMA



# MongoDB Complete Guide

---

*Develop Strong Understanding of  
Administering  
MongoDB, CRUD Operations, MongoDB  
Commands, MongoDB Compass,  
MongoDB Server,  
MongoDB Replication and MongoDB  
Sharding*

---

**Manu Sharma**



[www.bpbonline.com](http://www.bpbonline.com)

**FIRST EDITION 2021**

**Copyright © BPB Publications, India**

**ISBN: 978-93-89898-866**

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

#### **LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY**

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

#### **Distributors:**

##### **BPB PUBLICATIONS**

20, Ansari Road, Darya Ganj

New Delhi-110002

Ph: 23254990/23254991

##### **MICRO MEDIA**

Shop No. 5, Mahendra Chambers,

150 DN Rd. Next to Capital Cinema,

V.T. (C.S.T.) Station, MUMBAI-400 001

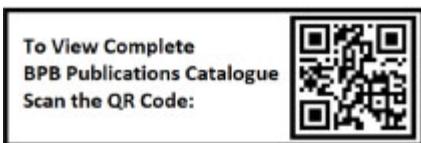
Ph: 22078296/22078297

##### **DECCAN AGENCIES**

4-3-329, Bank Street,

Hyderabad-500195  
Ph: 24756967/24756400

**BPB BOOK CENTRE**  
376 Old Lajpat Rai Market,  
Delhi-110006  
Ph: 23861747



Published by Manish Jain for BPB Publications, 20 Ansari Road, Darya Ganj, New Delhi-110002 and Printed by him at Repro India Ltd, Mumbai

[www.bpbonline.com](http://www.bpbonline.com)

# **Dedicated to**

*All My Family Members and Friends:*

*My Grandmother:  
Shrimati Pushpa Devi*

*My Parents:  
Shri Vijay Sharma  
Shrimati Neelam Sharma*

*My Wife:  
Anu Sharma*

*My Sister:  
Neha Sharma*

*And Specially to My Angel Daughter:*

*Siya Sharma*

# About the Author



**Manu Sharma** (MPhil) is having more than 17 years of industry experience in Software Development at Architect Level, Web Administration, Project Management and Execution, Product Development and Team Management.

He has worked in various Multinational Companies, Small to Mid-Sized Organizations, Universities as well as one of the Biggest Conglomerate of India. He is also the Founder, Architect and Developer of two Open Source Projects. In his free time, he loves to spend his time with his family and daughter. His other interests are singing and Arts, during some weekend you can find him Singing and Recording Music in Studios, Playing Flute and sometimes you can catch him with painting brushes in his hand while he paints.

# About the Reviewers

**Cindreen Clarence** is a Technical Lead in Database Technologies and she is having more than 16 Years of Experience in IT, Project Management and Delivery. She carries a lot of Experience in various Databases like Oracle, Microsoft SQL Server and Modern Databases like MongoDB. She is currently working in Wipro Technologies

**Dheeraj Chhabra** is Techno-Functional Project Manager with more than 16 years' experience in spearheading loosely coupled service oriented product and project development, architecture design and executing solutions for various initiatives – “Run the Business”, “Protect the Business” and “Change the Business”, having key focus on end-to-end project planning, execution and delivery management.

Dheeraj carries various certifications such as PSM, TOGAF, PMP, SAFe 4, PMI-ACP, SAFe 5 and He is currently working as Technical Project Manager in Ford Motor Pvt. Ltd.

**Harish Kumar Buttolia** is MSC (IT) from Punjab Technical University, Harish carries more than 18 years of experience in IT in different Private and Government organizations. He is having good knowledge in software and application development using open source technologies. He also has various articles/publications published in renewed India/International Journals. He always keen to learn new technologies. He is also interested in writing, reading music and travel.

Harish is currently working as a Scientist in ICMR (Indian Council of Medical Research), New Delhi where he is one of the core members of development team who develop and manage Covid19 patient's National data collection application, data analytics and Antimicrobial Resistance Surveillance System (AMR).

**Rohit Agarwal** is a Sr. Data Architect at Virtustream (A Dell Technologies Business). He is specialize in providing end to end

solutions to the various teams by creating data architecture, pipelines & managerial reporting system. This helps the Datacenter Platform & Operations folks to learn more about their data by taking better decisions, perform managerial reporting by using different tools, etc. He earned his master's degree in information system from Northeastern University, Boston, in the year 2017. He have a keen interest in Entrepreneurship & Innovation, and He is learning the same in Harvard School, Boston. In his free time He love to get indulge in the new activities like Photography, reading, fitness (Kickboxing), etc.

**Shailesh Soni** is technically accomplished IT professional with enrich experience of more than 16 years in service and as well as product based organizations with insightful experience in various aspects of software development using multiple technologies.

Shailesh carries vast experience in software development and currently working as Solution Architect at Table Space Technologies.

# Acknowledgement

First of all I am thankful to almighty God to provide me with the opportunity to write a Book, I am very thankful to Mr. Nrip Jain (Head, Business Development Group, BPB Publications) for believing in me and offering me to write this Book.

I would like to thank all Family Members and Friends for continuously encouraging me for writing the book — I could have never completed this book without their support.

I am also thankful to all my Gurus and Teachers in life for their teachings and blessings.

My Special Thanks to my Daughter “Siya” for supporting me during the Book Journey.

I feel great to have some college time friends most of them are still in touch with me including my college buddies from ET&T as well as few other friends Dr. Krishna Kabir, Sanjay Gupta, Amit Goel, Rajan Goel, Dharmender, Sunil Saini, S K Rai, Ron Buening, Amit Puri, Harsh Thukral – Thank You All

I am very thankful to Mr. Dheeraj Sareen who have been my life long mentor and Immediate Supervisor in 5 Organisations including the Present one. One of the Best Person whom I met in my Professional Life.

I am thankful to Prof. R.B. Solanki who has guide me when I was doing MPhil under his Guidance.

I am also very Thankful to David Hirschfeld (Founder, CEO and CTO of my ex-organisation, Tekyz Inc) and Yaseen Shaik (CTO of my Present Organisation, Spiralyze LLC) to be my Technical Mentors and always giving their best insights of Technology and Current Trends.

I would like to also thank few People from my Previous and Present Organisation(s): Amit Sood, Harish Buttolia, Sanjay Dagar (@ Webcom Systems) Hemant Malik, Kuldeep Singh Sidhu, Anurag

Sharma, Dheeraj Chhabra, Sailesh Soni, Mahesh Pal, Vineet Malhotra (@ Infopro) Prof. S.P. Narang, Dr. Pratul Sharma, Anupam Kaushik, Vipin Sharma, Sourav Basu, Jatinder Jeet Singh (@ Shri Ram New Horizons) Anupam Srivastava (@ Miracle Technologies) Raj Bawa, LK Gahlot, Prakash Dutta, Sunil Kumar (@JBi Digital) Sailesh Tyagi, Joby Jose, Rajesh Singh, Shailendra Dixit (@TSI India) Gayathri P Borker (@ Tekyz Inc) Prof. A.K Bakshi, Dr. Vimal Rarh, Dr. Arun Julka (@ University of Delhi), Gajan Retnasba (One of the best CEOs that I worked with), Mamoon Mustaque, Nikunjkumar Balar, Yuriy Kycha-Kolot, Bhavesh Vavadiya (@ Spiralyze LLC)

My gratitude also goes to the entire team at BPB Publications for being always supportive during the entire Book Journey and whenever I need their help they were always available to help me.

Last but not the least I am very thankful to all the Technical Reviewers of this Book, I really appreciate their hard work during Technical Review of this Book and thankful to them whenever they have corrected me in some places which requires changes.

# Preface

This book covers MongoDB in a manner keeping the view for the Beginners who wanted to learn MongoDB starting from the Introduction of MongoDB and why it is different from other traditional databases. The Chapters in this Book are divided into 3 Main Topics covering MongoDB from Its Basics to the Intermediate Level and then at last covering some very Advance topics.

So a Beginner who will start his journey to learn MongoDB can easily feel the progress from one Chapter to the next one. At last when reader will finish his journey he will have a great confidence of Mastering the MongoDB.

This book is written with a great thought process and covers almost every topic of MongoDB used in daily use cases. Every Chapter and Topics are explained in a detailed Step by Step Practical Manner with the use of Diagrams, Codes, Commands and Screenshots which will make this Book very interesting to read.

Whether it is the Introduction of MongoDB, Installation Part, Introduction to Storage Engines, MongoDB Shell, CRUD Methods, Indexes in MongoDB, Query Selectors, Projection, Aggregation, MongoDB Compass, MongoDB Administration and Management or some Advanced Topics like Replication or Sharding, each and every topic is well placed so that the flow of the reader will remain intact and reader will enjoy reading the entire Book by learning it Practically with the Code and Commands explained in the Book.

**The First Main Topics are Basic Level Topics and will consist of the following 9 chapters, in which you will learn the following:**

[\*\*Chapter 1\*\*](#) will focus the Introduction to MongoDB and its Architecture, it will cover the basics and key terms of MongoDB in a manner which is easier for a beginner to understand very quickly. This Chapter also gives the introduction about the concept of Document based Database and NoSQL Databases and how these are different from SQL based databases. We will be also covering

the MongoDB Architecture is a very easy manner with the help of Diagrams. In the last topic of this chapter we will also cover the Core Concepts and Vocabulary of MongoDB and will also compare these terms with respect to the SQL based Databases.

[Chapter 2](#) will focus the Installation and Steps to Setting up MongoDB on Windows 10 powered machines. This chapter will also cover the post installation checks and the overview of MongoDB editions. So in this chapter you will be learning about how to download MongoDB for Windows and then how to install it correctly on your Windows Machine, This chapter also covers the Enterprise Advanced Edition of MongoDB and advantages to use it. This chapter is full of Step by Step method explained with Screenshots so that you are able to understand the Installation and Setup of MongoDB very easily on Windows OS Platforms. This chapter also covers the Post Installation Steps so that you can easily verify that if MongoDB is correctly installed on your Windows System.

[Chapter 3](#) will focus the Installation and Steps to Setting up MongoDB on Machines powered by Linux Operating System (We have used Ubuntu OS the widely used variant of Linux Operating System). This chapter will also cover the post installation checks for MongoDB Installation for Linux Operating Systems. So in this chapter you will be learning about how to download MongoDB for Linux and then how to install it correctly on your Linux Machine. This chapter is full of Step by Step method explained with Screenshots so that you are able to understand the Installation and Setup of MongoDB very easily on Linux OS Platforms. This chapter also covers the Post Installation Steps so that you can easily verify that if MongoDB is correctly installed on your Linux System.

[Chapter 4](#) will focus the Installation and Steps to Setting up MongoDB on Machines powered by macOS (Mac Operating System). This chapter will also cover the post installation checks for MongoDB Installation for Mac Operating Systems. So in this chapter you will be learning about how to download MongoDB for macOS and then how to install it correctly on your macOS Machine. This chapter is full of Step by Step method explained with Screenshots so that you are able to understand the Installation and Setup of

MongoDB very easily on macOS Platforms. This chapter also covers the Post Installation Steps so that you can easily verify that if MongoDB has correctly installed on your macOS System.

**Chapter 5** will learn the basics of MongoDB, including the overview of MongoDB Databases, MongoDB Collections and MongoDB Documents. This Chapter will explain you the difference between the terminologies used in the MongoDB, which is different from the other types of Databases like RDBMS in more detailed manner. This Chapter also gives you an overview of MongoDB Shell and covers some Basic Shell Commands. In last topic, this chapter will also give you an introduction to MongoDB Clients which can be useful to connect to MongoDB Server and perform various operations that you can perform easily using these MongoDB Clients.

**Chapter 6** will learn the concept of Storage Engines in Database Management Systems and why they are used. This chapter also covers the concept of Storage Engines and explain the Storage Engine with the help of Diagram. This Chapter also covers the Storage Engines which are used in MongoDB such as WiredTiger Storage Engine as well as In-Memory Storage Engine. We will be also covering Encrypted Storage Engines and well as Third Party Pluggable Storage Engines in this chapter and also compare the Main Storage Engines with respect to their features. This Chapter will also cover the concept of Locks in the Database and also covers the overview of Locks in MongoDB.

**Chapter 7** will learn the basic Commands and Methods which are used for Managing and Administrating MongoDB. We would be learning these Commands and Methods with the help of MongoDB Shell. We will learn how we can create, update and delete Databases, Collections and Documents using MongoDB Shell with the Help of MongoDB Shell Commands and Methods. We will also learn how we can create, update and delete documents using MongoDB Query and Write Operations Shell Commands and Methods. Later in this Chapter we will be learning MongoDB Authentication and Role Based Access Methods and how we can use them in MongoDB using MongoDB Shell Commands and Methods.

[\*\*Chapter 8\*\*](#) will learn more about the MongoDB Shell Methods which are used to Connect to the MongoDB Server. We will start with the JavaScript in MongoDB and also covers and overview of list of various other Languages which are officially supported by MongoDB. We will also cover the commands related to the Database in which we will cover various methods related to Database Management, we will be also covering various methods related to Collections and how we can Manipulate MongoDB Collections using these Methods. At last we will be covering Cursor in MongoDB and what are various Cursor related Methods in MongoDB that we can use. These methods are very useful and we can use them in various scenarios while working with MongoDB.

[\*\*Chapter 9\*\*](#) will learn about the Data Types which are can be used in MongoDB. We will start with the introduction of Data Type what exactly it is and then we will also cover an overview of the BSON Data Types. There are different types of Data Types which are used in MongoDB and each one of them are having different properties and structure and each of them are used in different scenarios some of them are widely used and some of them are not very frequently used.

**The Second Main Topics are Intermediate Level Topics and will consist of the next following 7 chapters from [Chapter 10](#) to [Chapter 16](#), in which you will learn the following:**

[\*\*Chapter 10\*\*](#) will learn about the MongoDB CRUD Operations, we will be covering the operations which are helpful in Creating Documents, Reading Documents, Updating Documents and Deleting Documents in MongoDB. We will also be covering the Bulk Write Operation in MongoDB. All of these examples will be explained in Step by Step Manner. We will be covering all these by giving the Practical Example and also covers various Methods to perform all these Operations. There are multiple Methods for each Operation that we will be covering in this Chapter. We will be also learning about the various Options such as ordered, multi and justOne which we will be using in various MongoDB CRUD Methods and we will be also covering the Field Update Operators which are used in MongoDB Update Methods. These CRUD Operations are very important in day

to day working with MongoDB and there are used by Application Developers quite frequently.

**Chapter 11** will learn about the MongoDB Intermediate Concepts. In this Chapter we will be covering the topics like Atomicity and Consistency in MongoDB, we will also learn about Consistency and Consistency in MongoDB, This Chapter also gives the Basic Introduction to Replication and Sharding and how they are useful. Later in this Chapter we will cover the MongoDB specific Distributed Operations and Queries in which we will look how the Read and Write Operations are performed when we use Replication and Sharding in MongoDB.

**Chapter 12** will learn about the concept of Indexing, in this chapter will start from the introduction to Indexes and their benefits where we will learn that Indexes are special type of data structures in Data Base Management Systems like MongoDB which stores the data in easy to traverse form we will be also giving Introduction to MongoDB Default \_id Index and learning more about its properties, we will then learn how we can create an Index in MongoDB. Later in this chapter we will study the different Types of Index which we can create in MongoDB with the step by step Practical Examples. We will also study some of the different Index Properties which we can use in MongoDB while we create an Index in MongoDB. We will also study how we can use Indexes with other MongoDB Methods, we will also learn about the Collation and how we can use it with MongoDB Index. In the Last part of this Chapter we will be covering about how we can view the existing Indexes and how we can delete the existing Indexes from the MongoDB Collection.

**Chapter 13** will learn about the MongoDB Query Selectors, we will start with the basic Introduction to the Query Selectors and why they are useful, we will be covering different types of Query Selectors that are available in MongoDB and then we will be also covering these Query Selectors by Step by Step Practical Examples.

**Chapter 14** will learn about the Projection in MongoDB by giving its introduction and what are the benefits of using Projection. In the Later Part of this Chapter we will be covering the Projection Operators by giving their Introduction and we will be covering the

various Types of Projection Operators which are available in MongoDB. In the Last Section of this Chapter we will be covering some Projection Operators with Step by Step Practical Examples.

[\*\*Chapter 15\*\*](#) we will get introduced to Aggregation in MongoDB and Benefits of using MongoDB Aggregation. We will be learning about Aggregation Expression Types and we will also covering Aggregation Expression Types with Step by Step Practical Examples. Later in this Chapter we will be covering the Map-Reduce in MongoDB, what are the benefits of using Map-Reduce and we will be learning Map-Reduce with some Practical Examples. In the last Section of this Book we will be covering the Aggregation Pipeline and its benefits, we will be covering how Aggregation Pipeline works in MongoDB in detailed manner and at last we will be using some Step by Step practical Examples to understand the Aggregation Pipeline in better way.+

[\*\*Chapter 16\*\*](#) we will learn about MongoDB Compass which is the Official GUI Tool for MongoDB. We will be learning about the Benefits of using MongoDB Compass and how we can install MongoDB Compass on our Machine using Step by Step Method. Later in this Chapter we will be learning about how we can use MongoDB Compass to connect to MongoDB Server. In the Last Part of this Chapter we will be covering MongoDB Compass with some Step by Step Practical Examples which will give us more idea on what we can do with MongoDB Compass.

**The Third and the Last Topics are Advanced Level Topics and will consist of the next following 3 chapters from [Chapter 17](#) to [Chapter 19](#), in which you will learn the following:**

[\*\*Chapter 17\*\*](#) we will learn the Advanced Administration Topics of MongoDB which is very helpful to the people who will perform various MongoDB Administrative tasks. In this Chapter we will learn about mongod Process and how to manage mongod process. We will learn about how to monitor and diagnose MongoDB. We will cover the step by step method to how to install MongoDB Tools on our Machine and how to use these MongoDB Tools and various other MongoDB commands to monitor and diagnose MongoDB.

Later in this Chapter we will be learning about how we can take MongoDB backups and how we can restore these backups. In later sections of this Book we will cover how we can perform the export and import of MongoDB Data. In the Last Part of this Chapter we will be covering the various important points related to MongoDB Security which we should take care so that our MongoDB Database and its data will get secured.

**Chapter 18** we will learn the Replication Part of MongoDB. In this Chapter we will learn about the Replication and Replica Set in a Quick Recap, We will be also learning about the MongoDB Heartbeats and how Heartbeats plays an important role in the replicated environment, We will be also learning that how the election of the new Primary Member takes Place. Later in this Chapter we will cover the Pre Configuration Steps before we start with the practical Step by Step method to create the Replicated Environment with MongoDB Primary and Secondary Instances, We will then also learn how to Setup the Replicated MongoDB Environment with the Step by Step method. In the last Part of this Chapter we will be learning on how we can verify the Replication Setup if it has been configured correctly with the help of the data.

**Chapter 19** we will learn the Sharding Part of MongoDB. In this Chapter we will learn about the Sharding and Shaded Clusters in a Quick Recap, We will be also learning Importance of Config Database in the Sharded Environment, We will be also learning about the Shard Keys. Later in this Chapter we will cover the Pre Configuration Steps before we start with the practical Step by Step method to create the Sharded Environment with MongoDB Replica Sets, We will then also learn how to Setup the MongoDB Sharded Environment. We will learning the complete Step by Step Process of Sharding in easy to understand 8 Steps

# Downloading the code bundle and coloured images:

Please follow the link to download the **Code Bundle** and the **Coloured Images** of the book:

**<https://rebrand.ly/dfca4a>**

## Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

**[errata@bpbonline.com](mailto:errata@bpbonline.com)**

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at **[www.bpbonline.com](http://www.bpbonline.com)** and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at **[business@bpbonline.com](mailto:business@bpbonline.com)** for more details.

At [www.bpbonline.com](http://www.bpbonline.com), you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

## **BPB is searching for authors like you**

If you're interested in becoming an author for BPB, please visit [www.bpbonline.com](http://www.bpbonline.com) and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

The code bundle for the book is also hosted on GitHub at <https://github.com/bpbpublications/MongoDB-Complete-Guide>. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at <https://github.com/bpbpublications>. Check them out!

## **PIRACY**

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at [business@bpbonline.com](mailto:business@bpbonline.com) with a link to the material.

## **If you are interested in becoming an author**

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit [www.bpbonline.com](http://www.bpbonline.com).

## **REVIEWS**

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased

opinion to make purchase decisions, we at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit  
[www.bpbonline.com](http://www.bpbonline.com).

# Table of Contents

## 1. Introduction to MongoDB

Structure

Objectives

Introduction

The definition of MongoDB

What is a document database?

What is JSON? How does it look?

Cross-platform

Scalable

Flexible

Classified as NoSQL database

What is a NoSQL database?

An overview of the MongoDB architecture

A quick look into the NoSQL database architecture

A quick look into the MongoDB architecture

MongoDB data platform

Difference from other databases

The concept of NoSQL databases

Types of NoSQL database management systems

Key-value paired databases

Column-oriented databases

Document databases

Graph databases

Introduction to MongoDB basics, core concepts, and vocabulary.

Document database

Collections

Support for rich query language

Support multiple storage engines

Some basic MongoDB terminology

MongoDB terminology comparison with SQL databases

Conclusion

Questions

## 2. MongoDB Installation Setup on Windows

Structure

Objectives

Overview of MongoDB editions

Features of MongoDB Enterprise Advanced Edition

Comparison between MongoDB Community Edition and MongoDB Enterprise Advanced Edition

MongoDB setup on Windows

Installing MongoDB Community Edition on Windows operating system

Installation steps

Conclusion

Questions

## 3. MongoDB Installation and Setup on Linux (Ubuntu)

Structure

Objectives

MongoDB Setup on Linux

Installing MongoDB Community Edition on Linux Operating System

Installation steps

Method one (Browser method).

Step 1 – Download MongoDB Community Edition

Step 2 – Install MongoDB Community Edition on your Linux machine

Installation Steps

Method two (Shell method).

Steps for installing MongoDB clients (mongo-clients) on Linux based systems (Ubuntu).

Steps for installing MongoDB on Linux based systems (Ubuntu) using the Shell commands

Step 3 – Starting MongoDB on Linux (Ubuntu).

Step 4 – Connecting to MongoDB on Linux (Ubuntu).

Conclusion

Questions

## 4. MongoDB Installation and Setup on macOS

[Structure](#)

[Objectives](#)

[MongoDB setup on macOS](#)

[Installing MongoDB Community Edition on macOS](#)

[Installation steps](#)

[Conclusion](#)

[Questions](#)

## **5. Getting Started with MongoDB**

[Structure](#)

[Objectives](#)

[MongoDB databases](#)

[What is a database?](#)

[What is a relational database?](#)

[What is a NoSQL database?](#)

[What is a MongoDB database?](#)

[MongoDB collections](#)

[What is a table in RDBMS?](#)

[What is a collection in MongoDB?](#)

[MongoDB documents](#)

[Row and column in RDBMS](#)

[What is a document in MongoDB?](#)

[Introduction to MongoDB Shell](#)

[What is a MongoDB Shell?](#)

[Connecting to MongoDB Shell](#)

[Step 1 – Connecting to MongoDB Shell](#)

[Basic Shell commands](#)

[MongoDB Shell basic command helpers](#)

[MongoDB Shell command history](#)

[Introduction to MongoDB clients](#)

[Conclusion](#)

[Questions](#)

## **6. Storage Engines in MongoDB**

[Structure](#)

[Objectives](#)

[What are storage engines?](#)

[Types of storage engines in MongoDB](#)  
[Introduction to the WiredTiger storage engine](#)  
[Introduction to the in-memory storage engine](#)  
[Encrypted storage engine](#)  
[Third-party pluggable storage engines](#)  
[MongoDB storage engines comparison](#)  
[MongoDB locks](#)  
[What is a database lock?](#)  
[Database lock operations types](#)  
[Database locks operations in MongoDB](#)  
[Conclusion](#)  
[Questions](#)

## **7. Managing and Administering MongoDB**

[Structure](#)  
[Objectives](#)  
[MongoDB administration commands and methods](#)  
[Create database command](#)  
[Create collection command](#)  
[Drop database command](#)  
[Drop collection command](#)  
[MongoDB query and write operation commands and methods](#)  
[Insert document command](#)  
[Read document command](#)  
[Delete document command](#)  
[MongoDB user authentication and role based commands and methods](#)  
[What is database authentication?](#)  
[What is role-based access control?](#)  
[Role-based authentication in MongoDB](#)  
[Conclusion](#)  
[Questions](#)

## **8. MongoDB Shell Methods**

[Structure](#)  
[Objectives](#)  
[JavaScript in MongoDB](#)

[Server Side JavaScript in MongoDB](#)

[What is map-reduce in MongoDB?](#)

[What is \\$where operator in MongoDB?](#)

[List of officially supported languages in MongoDB](#)

[MongoDB methods](#)

[Step 1 – Connecting to MongoDB Shell](#)

[MongoDB connection methods](#)

[connect\(url,username,password\)](#)

[Mongo\(host, clientSideOptions\)](#)

[Mongo.getDB\(database\)](#)

[MongoDB database methods](#)

[db.getMongo\(\)](#)

[db.stats\(\)](#)

[db.serverStatus\(\)](#)

[MongoDB Collection methods](#)

[db.collection.count\(\)](#)

[db.collection.stats\(\)](#)

[db.collection.totalSize\(\)](#)

[db.collection.validate\(\)](#)

[db.collection.drop\(\)](#)

[MongoDB cursor methods](#)

[What is a cursor in MongoDB?](#)

[cursor.count\(\)](#)

[cursor.pretty\(\)](#)

[cursor.sort\(\)](#)

[Conclusion](#)

[Questions](#)

## [9. Data Types in MongoDB](#)

[Structure](#)

[Objectives](#)

[What are data types?](#)

[Introduction to BSON data types](#)

[Data types in MongoDB](#)

[Integer data types](#)

[String data types](#)

[Double data types](#)

[Array data types](#)  
[Object data types](#)  
[Binary data types](#)  
[ObjectId data types](#)  
[Date data types](#)  
[Null data types](#)  
[Regular expression data types](#)  
[JavaScript data types \(without scope\)](#).  
[Javascript data types \(with scope\)](#).  
[Timestamp data types](#)  
[Boolean data types](#)  
[Min and max key](#).  
[Decimal128](#)  
[Comparison and sort order](#)  
[Conclusion](#)  
[Questions](#)

## **[10. Introduction to MongoDB CRUD Operations](#)**

[Structure](#)  
[Objectives](#)  
[MongoDB create operations](#)  
[db.collection.insert\(\) method](#)  
[Method definition](#)  
[Example 1 – Creating a single document in MongoDB collection](#)  
[Example 2 – Creating multiple documents in MongoDB collection](#)  
[db.collection.insertOne\(\) method](#)  
[Method definition](#)  
[Example – Creating a single document in MongoDB Collection using insertOne\(\) method](#)  
[db.collection.insertMany\(\) method](#)  
[Method definition](#)  
[Example – Creating multiple documents in MongoDB collection using insertMany\(\) method](#)  
[The \\_id Field](#)  
[Example - Creating a new document by specifying \\_id key](#)

The ordered option

MongoDB read operations

`db.collection.find()` Method

Method Definition

Example 1 – Reading documents in MongoDB collection without Query

Example 2 – Reading documents in MongoDB collection with Query

Using Pretty method with `find()`.

Example 3 – Reading documents in MongoDB collection with Query and Pretty method

MongoDB update operations

The `$set` operator

The `$unset` operator

`db.collection.update()` method

Method definition

Example 1 – Updating a single document in MongoDB collection using `update()` method

Example 2 – Updating multiple documents in MongoDB collection using `update()` method

`db.collection.updateOne()` method

Method definition

Example – Updating a single document in MongoDB collection using `updateOne()` method

`db.collection.updateMany()` method

Method definition

Example – Updating multiple documents in MongoDB collection using `updateMany()` method

The `upsert` option

The `multi` option

MongoDB delete operations

`db.collection.remove()` method

Method definition

The `justOne` option

Example 1 – Deleting a single document in MongoDB collection using `remove()` method

[Example 2 – Deleting multiple documents in MongoDB collection using remove\(\) method](#)

[db.collection.deleteOne\(\) method](#)

[Method definition](#)

[Example – Deleting a single document in MongoDB collection using deleteOne\(\) method](#)

[db.collection.deleteMany\(\) method](#)

[Method definition](#)

[Example – Deleting multiple documents in MongoDB collection using deleteMany\(\) method](#)

[MongoDB bulk write operations](#)

[db.collection.bulkWrite\(\) method](#)

[Method definition](#)

[Example – Bulk write in MongoDB collection using bulkwrite\(\) method](#)

[Conclusion](#)

[Questions](#)

## [11. MongoDB Intermediate Concepts](#)

[Structure](#)

[Objectives](#)

[Atomicity](#)

[What is atomicity?](#)

[Atomicity in MongoDB](#)

[MongoDB atomicity and multiple document transactions](#)

[Consistency](#)

[What is consistency?](#)

[Consistency in MongoDB](#)

[MongoDB and eventual consistency](#)

[Basic introduction to replication](#)

[Replica sets](#)

[Basic introduction to sharding](#)

[Sharded clusters](#)

[Distributed operations and queries](#)

[Read operations on replica sets](#)

[Write operations on replica sets](#)

[Read operations on sharded clusters](#)

## Write operations on sharded clusters

Conclusion

Questions

## 12. Introduction to MongoDB Indexes

Structure

Objectives

What are indexes?

[Indexing and MongoDB](#)

[Benefits of indexing](#)

Default \_id index

[The \\_id properties](#)

[Code 1](#)

[Code 2](#)

Creating an index

[db.collection.createIndex\(\) method](#)

[Method definition](#)

[Example – Creating an index in MongoDB collection](#)

[Code 1](#)

Index types in MongoDB

[Single field index](#)

[Example – Creating a single field index in MongoDB collection](#)

[Code 1](#)

[Compound Index](#)

[Example – Creating a compound index in MongoDB collection](#)

[Code 1](#)

[Multikey index](#)

[Example – Creating a multikey index in a MongoDB collection](#)

[Code 1](#)

[Text index](#)

[Example – Creating a text index in a MongoDB collection](#)

[Code 1](#)

Special types of index

[Geospatial index](#)

## Hashed index

### Index properties

#### Unique index

Example – Creating a unique index in a MongoDB Collection

Code 1

#### Partial index

Example – Creating a partial index in a MongoDB Collection

Code 1

#### Sparse index

Example – Creating a sparse index in a MongoDB collection

Code 1

#### TTL index

Example – Creating a TTL index in a MongoDB collection

Code 1

### Using an index

Example – Creating and using an index in a MongoDB collection

Code 1

Code 2

### Indexes and collation

Example – Creating an index with collation in a MongoDB Collection

Code 1

### View index information

#### `db.collection.getIndexes()` method

Example – Viewing all the Indexes in a MongoDB Collection

Code 1

### Deleting an index

#### `db.collection.dropIndex()` method

Method definition

Example 1 – Deleting an index in a MongoDB collection

Code 1

#### `db.collection.dropIndexes()` method

[Method definition](#)

[Example 2 – Deleting multiple index in a MongoDB collection](#)

[Code 1](#)

[Example 3 – Deleting multiple index in a MongoDB collection using the array\\_type values as parameter](#)

[Code 1](#)

[Example 4 – Deleting all the indexes in a MongoDB collection](#)

[Code 1](#)

[Some restrictions in MongoDB index](#)

[Conclusion](#)

[Points to Remember](#)

[Multiple choice questions](#)

[Answer](#)

[Questions](#)

[Key terms](#)

## [13. MongoDB Query Selectors](#)

[Structure](#)

[Objectives](#)

[Introduction to query selectors](#)

[Comparison Selectors](#)

[Logical Selectors](#)

[Element Selectors](#)

[Evaluation Selectors](#)

[Geospatial Selectors](#)

[Bitwise Selectors](#)

[Comment Selector](#)

[Examples and use of query selectors](#)

[Code 1](#)

[Code 2](#)

[Examples of comparison selectors](#)

[Example 1 - \\$gt Comparison Selector](#)

[Code 1](#)

[Selector Details](#)

[Example 2 - \\$lte comparison selector](#)

Code 1

Selector Details

Example 3 - \$in comparison selector

Code 1

Selector Details

Examples of logical selectors

Example 1 - \$and logical selector

Code 1

Selector Details

Example 2 - \$not logical selector

Code 1

Selector details

Examples of element selectors

Example 1 - \$exists element selector

Code 1

Selector details

Examples of array selectors

Example 1 - \$all array selector

Code 1

Selector Details

Examples of evaluation selectors

Example 1 - \$regex evaluation selector

Code 1

Selector details

Conclusion

Questions

## 14. Projection in MongoDB and Projection Operators

Structure

Objectives

Introduction to projection

How to use projection in MongoDB?

Examples of projection

Example 1 - Show only specific fields

Example 2 - Hide specific fields

Example 1 - Show only specific fields and hide the \_id field

Introduction to projection operators

[Examples of projection operators](#)

[Example 1 - \\$projection operator](#)

[Operator details](#)

[Example 2 - \\$elemMatch projection operator](#)

[Operator details](#)

[Conclusion](#)

[Questions](#)

## [15. Aggregation in MongoDB](#)

[Structure](#)

[Objectives](#)

[Introduction to MongoDB aggregation](#)

[Aggregation method syntax and use](#)

[Examples and use of aggregation method](#)

[Examples of aggregation](#)

[The MongoDB \\$group operator](#)

[Example 1 - \\$sum aggregation expression type](#)

[Expression type details](#)

[Example 2 - \\$sum aggregation expression type with operation in group output](#)

[Example 3 - \\$sum aggregation expression type with some other field](#)

[Example 4 - \\$avg aggregation expression type](#)

[Expression type details](#)

[Example 5 - \\$max aggregation expression type](#)

[Expression type details](#)

[Example 6 - \\$push aggregation expression type](#)

[Expression type details](#)

[Example 7 - \\$last aggregation expression type](#)

[Expression type details](#)

[Introduction to map-reduce](#)

[The mapReduce\(\) method](#)

[Example 1 – mapReduce\(\)](#)

[Introduction to aggregation pipeline](#)

[What is pipeline?](#)

[MongoDB aggregation pipeline](#)

[Example 1 – aggregate\(\)](#)

[Example 2 – aggregate\(\) with \\$out](#)

[Conclusion](#)

[Questions](#)

## [16. MongoDB Data Manipulations Using MongoDB Compass](#)

[Structure](#)

[Objectives](#)

[Introduction to MongoDB Compass](#)

[Installing MongoDB Compass](#)

[Installing MongoDB Compass on Windows operating system](#)

[Connecting MongoDB Server with MongoDB Compass](#)

[Practical examples with MongoDB Compass](#)

[Example 1 –Browsing the collections in the database](#)

[Example 2–Creating new collection in the database](#)

[Example 3–Browsing documents in the database](#)

[Example 4–Performing CRUD operations in documents](#)

[Example 5–Editing a document](#)

[Conclusion](#)

[Questions](#)

[Points to remember](#)

[Multiple choice questions](#)

[Answer](#)

[Key terms](#)

## [17. Managing and Administering MongoDB \(Advanced Level\)](#)

[Structure](#)

[Objectives](#)

[About mongod process](#)

[Managing mongod process](#)

[MongoDB service in Windows](#)

[Running mongod from command prompt](#)

[Stopping MongoDB services from Windows service manager](#)

[Stopping MongoDB Services from command line – MongoDB Shell method](#)

[Monitoring and diagnosing MongoDB](#)

[Installing MongoDB tools and utilities](#)

[Verifying the installation of MongoDB tools and utilities](#)

## Working with MongoDB tools and utilities

[mongostat](#)

[mongotop](#)

[serverStatus](#)

[dbStats](#)

[collStats](#)

[buildInfo](#)

[hostInfo](#)

[listCommands](#)

[ping](#)

[getLog](#)

## Backup and restore with MongoDB

[Taking a MongoDB backup using mongodump](#)

[Restoring a MongoDB database using mongorestore](#)

## Import and export with MongoDB

[Exporting a MongoDB data using mongoexport](#)

[Importing a MongoDB data using mongoimport](#)

## MongoDB security

[Enable authentication](#)

[Use role-based authorization \(role-based access control\)](#)

[Encrypt the communication channels and connections](#)

[Encrypt your MongoDB data](#)

[Use firewalls and restrict all the incoming and outgoing traffic](#)

[Regularly perform security audits](#)

[Conclusion](#)

## Questions

## 18. Replication in MongoDB

[Structure](#)

[Objectives](#)

[Basic introduction to replication – quick recap](#)

[Replica sets](#)

[MongoDB heartbeats](#)

[Automatic election of the new primary member](#)

[Pre-configuration steps](#)

[Starting with the MongoDB replication on Windows machine](#)

[Verifying the MongoDB replication using data](#)

Conclusion  
Questions

## 19. Sharding in MongoDB

Structure

Objectives

Basic introduction to sharding – quick recap

Sharded clusters

*Read and write operations and importance of config database*

Shard key

Pre-configuration steps

Starting with MongoDB sharding on Windows machine

*Step 1 – Stop the existing MongoDB services on Windows machine*

*Step 2 – Create the first replica set (our first shard)*

*Step 3 – Create the second replica set (our second shard)*

*Step 4 – Create the third replica set (our third shard)*

*Step 5 – Create the replica set of config servers in the sharded environment*

*Step 6 - Starting MongoDB in the sharded environment as "mongos"*

*Step 7 – Adding MongoDB in the shard keys*

*Step 8 - Verifying the MongoDB sharding using data*

Conclusion  
Questions

Index

# CHAPTER 1

## Introduction to MongoDB

This chapter covers the introduction to MongoDB and its architecture. It will cover the basics and key terms of MongoDB in a manner that is easier for a beginner to understand. This chapter also gives an introduction to the concepts of document-based databases and NoSQL databases and how these are different from SQL-based databases. We will also cover the MongoDB architecture in a very simple manner with the help of easy-to-understand diagrams. In the last topic, we will cover the core concepts and vocabulary of MongoDB and compare these terms with respect to SQL-based databases.

### Structure

In this chapter, we will discuss the following topics:

- Introduction to MongoDB and its architecture
- Basics and key terms of MongoDB
- Introduction to the concept of document-based databases and NoSQL databases
- Difference between the NoSQL databases and SQL based databases
- The MongoDB architecture overview
- Core concepts and vocabulary of the MongoDB

### Objectives

After studying this unit, you should be able to understand what exactly is MongoDB and also understand the basics of MongoDB.

- Understand what are NoSQL databases

- Compare between MongoDB and other databases

## **Introduction**

MongoDB is one of the top most popular modern databases which is now widely used for building modern day applications. It is different from traditional RDBMS (Relational Database Management Systems) and it is categorized as NoSQL database because of the flexibility of schema which it provides. MongoDB is easy to use and the data is stored in the JSON (JavaScript Object Notation) like format which stores data in key-value pair and it is very easy for developers while they work on database related queries.

Simple to use, Flexible, Powerful, Scalable and Development Friendly are some of key reasons behind the popularity of MongoDB.

## **The definition of MongoDB**

The best way to define MongoDB is from two main sources. The first is the official website of the creators of MongoDB, *MongoDB Inc.*, and the other is Wikipedia.

Let's see how MongoDB is defined by the creators of MongoDB (*MongoDB Inc.*) and Wikipedia and let us go through these two definitions one-by-one.

According to the official website, *MongoDB Inc.* MongoDB is a general-purpose, document-based, distributed database built for modern application developers and for the cloud era.

Wikipedia defines MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with an optional schema. MongoDB is developed by *MongoDB Inc.*.

From the above two definitions, let's see the key pointers that define MongoDB:

- Document database
- Cross-platform
- Scalable

- Flexible
- Classified as a NoSQL database

Now, let's dive a little deeper into these concepts and see how these separate MongoDB from other database engines available.

## What is a document database?

The document database is a type of non-relational database that is designed to store documents in JSON-like format.

## What is JSON? How does it look?

JSON Stands for JavaScript Object Notation and it is one of the widely used open standard file formats which uses human-readable text to store and transmit data. It is a lightweight format for data transportation and is often used between the client and server architecture in software development.

JSON is very easy to understand and usually, it is "self-describing".

### **Simple JSON example:**

```
#Example 1.1
{
  "Students": [
    {"firstName": "Siya", "lastName": "Sharma",
     "Location": "India"},

    {"firstName": "Ron", "lastName": "Smith", "Location": "USA"},

    {"firstName": "Bash",
     "lastName": "Tao", "Location": "Philippines"}
  ],
  "Teachers": [
    {"firstName": "Dheeraj", "lastName": "Sareen",
     "Location": "India"},

    {"firstName": "David", "lastName": "Baker",
     "Location": "Canada"}
  ]
}
```

In the preceding example:

- JSON data is specified in the name and value pairs
- Each data is separated by commas in JSON
- The square brackets in JSON are used to hold an array of object(s)
- The curly braces in JSON are used to hold object(s)
- It is highly flexible and easy to understand as you write JSON in a human-understandable format.

The document-based databases make the lives of developers easier as they allow them to store data in the same document-model format as developers use JSON while coding. It is easy for them to relate the name and value pairs while referring to the data entered in these databases.

## **Cross-platform**

MongoDB is a cross-platform database, which means it can run on various operating systems and on various computer architectures. You can run MongoDB on Windows, Linux, and even Mac Operating Systems. It also supports various computer architectures that are machine-based and hardware-specific and depends on the processors and hardware plus operating systems that run that machine.

Officially *MongoDB Inc.* recommends these platforms for production use (as of now):

- Amazon Linux 2
- Debian 9 and 10
- RHEL/CentOS 6, 7, and 8
- SLES 12 and 15
- Ubuntu LTS 16.04 and 18.04
- Windows Server 2016 and 2019

We can see that there are a number of platforms supported by MongoDB for the production use. This also means that we can install

and run MongoDB on a variety of platforms like Windows, Linux, etc., and their variants.

**The 32-bit MongoDB processes are limited to about 2 GB of data. So if you have large data then please consider using 64-Bit architecture.**

**Note that 32 Bit processes are related to the 32-bit computer architecture, these kinds of architecture are present in the old processors. New Processors are capable of handling 64-bits of data per clock cycle.**

You can refer to the official MongoDB documentation here for the latest updates:  
<https://docs.mongodb.com/manual/administration/productivity-notes/>

## Scalable

MongoDB is extremely scalable, suppose you are developing an application that is presently not widely used and has limited usage but you believe that in the coming years, your application will have a huge increase in new users and there will be a multi-fold increase in the traffic. So the choice you make now for selecting a MongoDB database for your application is worth it. MongoDB is the first choice of Fortune 100 enterprises as well as startups. These organizations rely on MongoDB for their operations. The industry has seen a dramatic increase in the deployments of MongoDB, ranging from a single server to multi servers and clusters of MongoDB database.

MongoDB offers scalability for three different metrics:

- In terms of cluster
- In terms of performance
- *Scalability* in terms of data

We will cover all these topics related to Scalability in the coming chapters.

## Flexible

MongoDB doesn't heavily enforce schema but uses dynamic schema, and hence, is also referred to as a schema-less database. This makes it a highly flexible database.

**So first just a quick look at what is schema.**

Schema is the database structure in **Relational Database Management System (RDBMS)** where you define it before using the database in the application. This means that you are creating a blueprint of your application data structure and defining the tables, fields, and relationships.

**The advantages of using schema-less databases are as follows:**

So this gives MongoDB an edge over the schema-related databases where we first define the schema and then add records or data. The problem with the schema-related databases is that each time we need to add one extra field, we have to update our schema as we cannot add a new record without adding an extra field.

MongoDB is flexible in terms of its data structure. As we have discussed earlier that MongoDB relies on the JSON-like data structure which is very flexible in itself. So when we need to add any new data in MongoDB which has or more fields and where each data is dynamic, we can easily add these records to our database in MongoDB without any issues as it won't restrict us and we are not bound to a specific data structure to be added in the database.

Many organizations throughout the world rely on MongoDB because it enables their development teams to build applications faster using different data types and also allows them to manage their applications more effectively and with high flexibility.

## Classified as NoSQL database

We have already seen that MongoDB is a schema-less (or a dynamic schema-based) database. Let us now learn why it is called a NoSQL database.

But before that, let us understand what the term NoSQL is and what are NoSQL databases.

## **What is a NoSQL database?**

The NoSQL databases, sometimes also called non-SQL, non-relational, or not only SQL databases are those that have a different mechanism to store and retrieve data other than the tabular relations which are used in relational databases. These types of databases have existed long ago but the term NoSQL is new, coined because of the requirements of today's companies that work on big, real-time data, and AI applications.

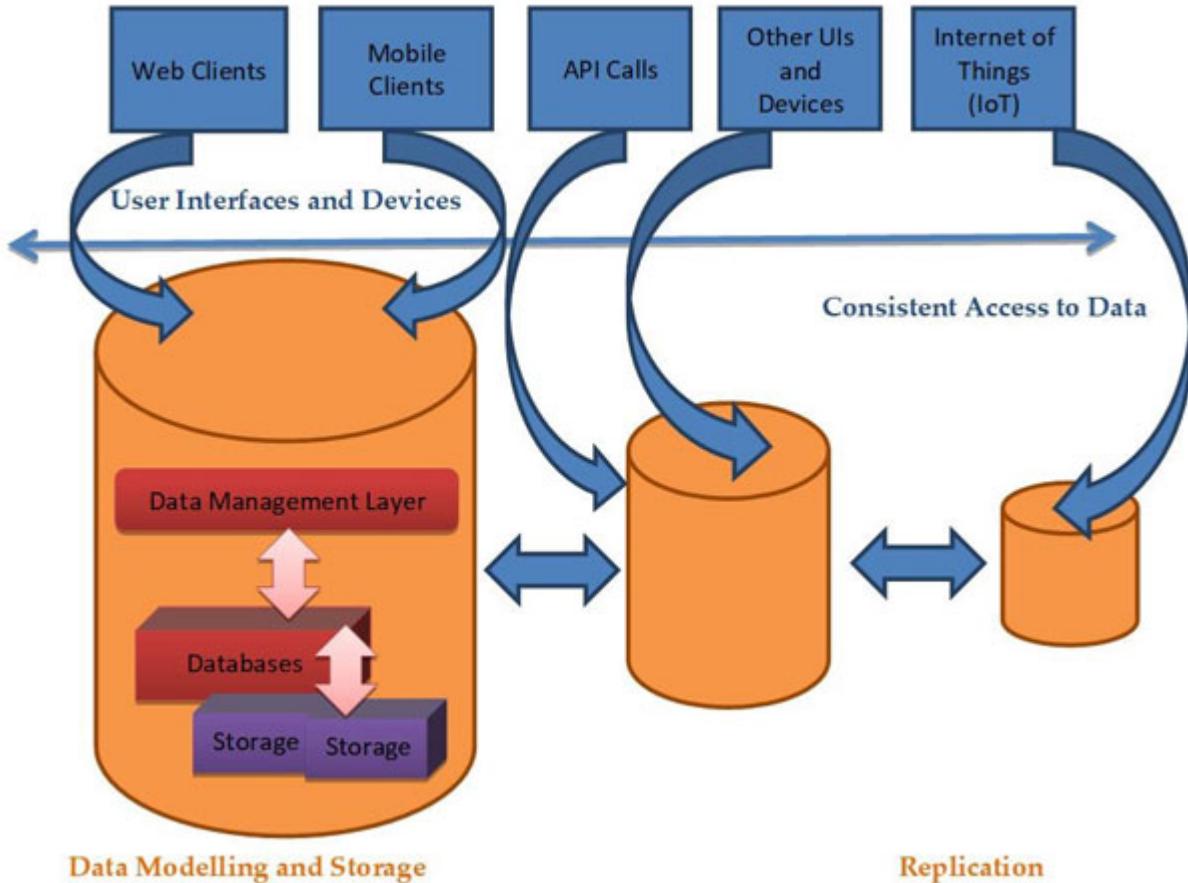
There are many kinds of NoSQL databases that we will cover in the next topic.

As we know, MongoDB uses JSON-like data structures that are different from what we use in RDBMS, so it is also called a NoSQL database. We will cover other NoSQL concepts later in this chapter.

## **An overview of the MongoDB architecture**

We will now look into how the MongoDB architecture looks like. It would be easier if we explain this with the help of some diagrams. Below is a diagram of the NoSQL database. As MongoDB is also a type of NoSQL database it would be better if we cover this first.

## **A quick look into the NoSQL database architecture**

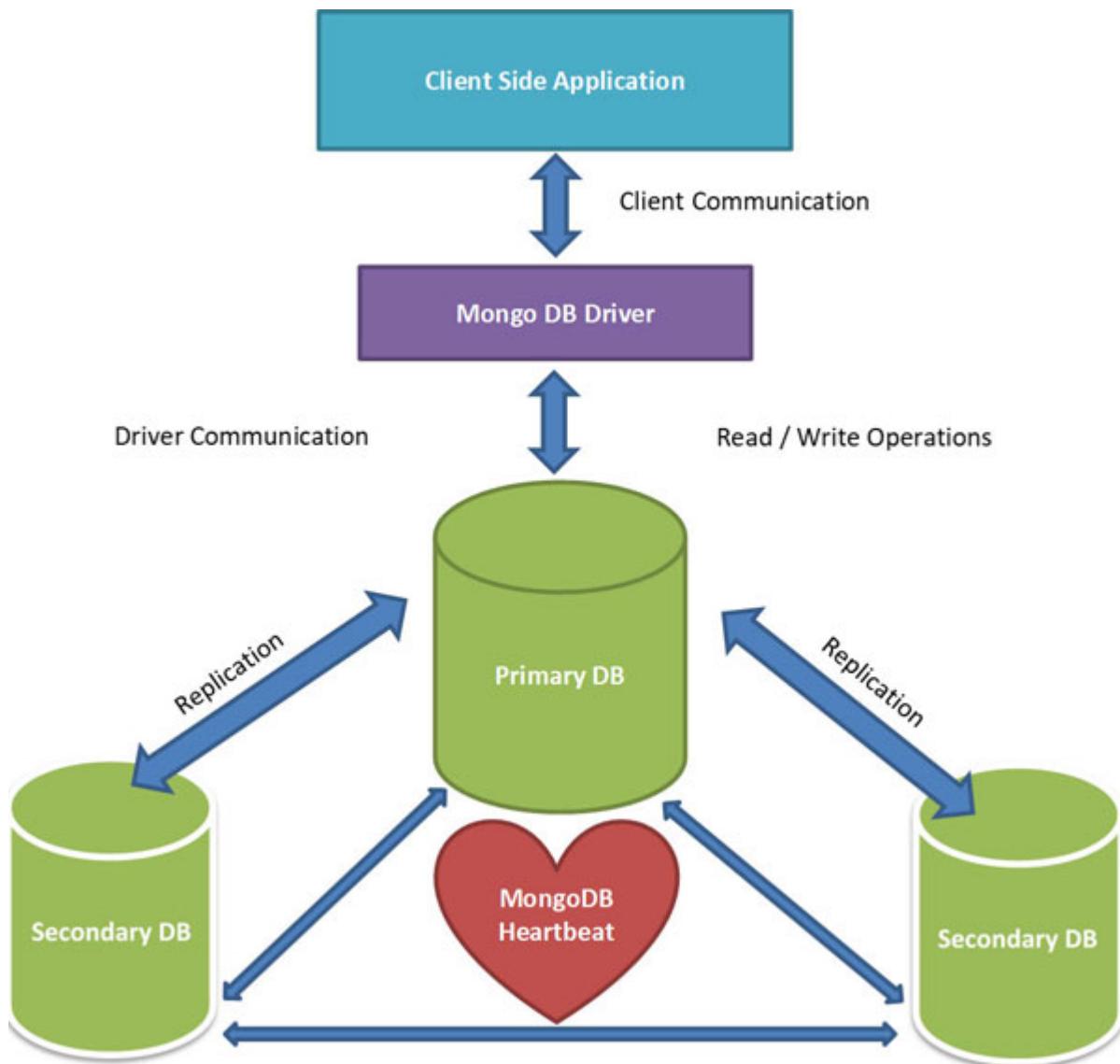


*Figure 1.1: NoSQL database architecture*

The important feature of NoSQL databases with respect to their architecture is that it provides nested, hierarchical structures in data entities. These hierarchical data structures can be easily described with the JSON and other formats used by the NoSQL databases. These structures also closely match with the data structures used in the programming languages.

## A quick look into the MongoDB architecture

Now, we will cover the MongoDB architecture with the help of a diagram. If you see the diagram below, you will easily get a quick idea of how the MongoDB architecture looks like:

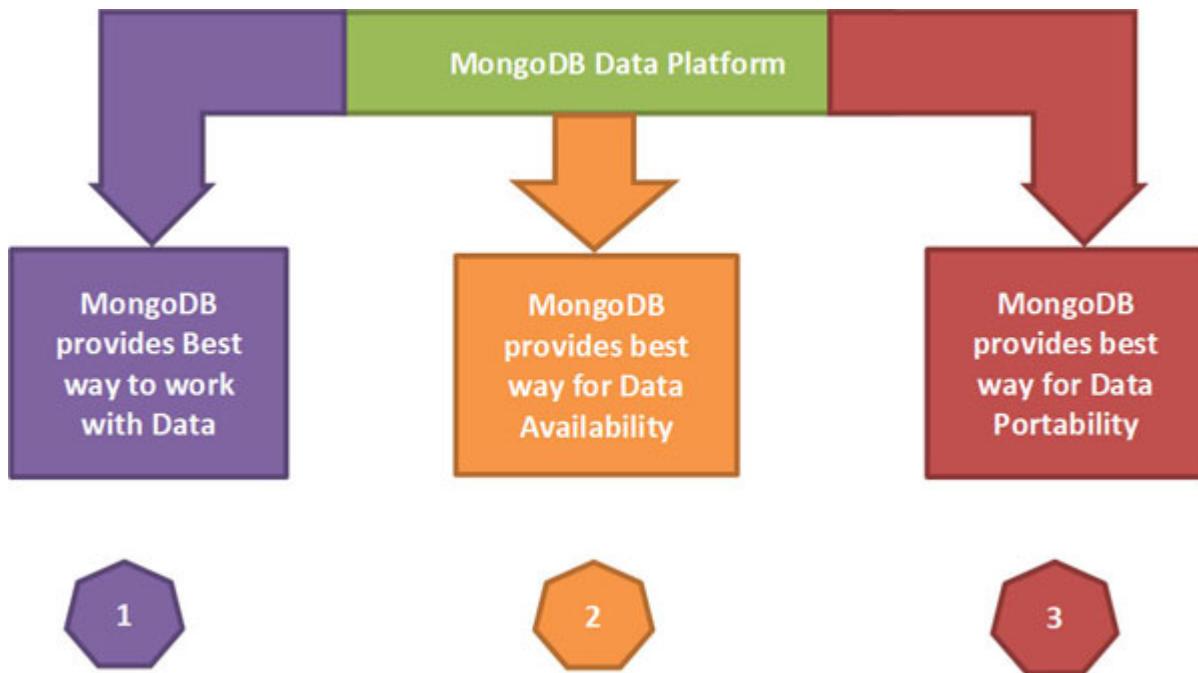


**Figure 1.2: MongoDB architecture**

In the preceding diagram, you will see that the client-side application communicates with the MongoDB database with the help of MongoDB drivers. For any read and write operations, the MongoDB drivers play an important role to communicate with the MongoDB database. The MongoDB drivers depend on the programming language and help applications for various CRUD (Create, Read, Update, and Delete) and other operations with respect to the MongoDB database. The primary and secondary DBs ensure the high availability of data and the frequent synchronization provides eventually consistent data records.

## MongoDB data platform

Let us have a quick look into the MongoDB database platform. As you can see in the following figure, the MongoDB database platform gives a lot of advantages as discussed in the following section:



*Figure 1.3: MongoDB Database Platform*

So, MongoDB provides:

- **Best way:** MongoDB provides an easy, fast, and flexible way to work with data. It also provides a wide range of data types and expressive queries that are easy to understand.
- **Best availability:** MongoDB provides the best way for data availability. With MongoDB, you can place and scale your data on any device as well as at any geographic location whenever you need it.
- **Best portability:** MongoDB runs the same everywhere, whether it is a local server or cloud. MongoDB is also available as a service throughout the world with major cloud platforms supporting it.

We will cover more on the MongoDB architecture in the advanced chapters of this book, where we will also explain the MongoDB

storage engines.

We will cover some more things like the MongoDB core concepts and the basics in the last topic of this chapter.

## Difference from other databases

It is a well-known fact the SQL databases have ruled the world in data technologies for many decades. The SQL databases are used to access mainly relational databases.

With databases like Microsoft SQL server, Oracle, MySQL, and Postgres, whether they are commercial databases or open-source databases, all of them are relational databases, which rely heavily on the schema, and thus, are also schema-related databases. We have already gone through these topics in the previous section of this chapter.

The surprising thing about the NoSQL databases is that they have existed since the 1960s but are getting popular now because today's applications require a high level of scalability and availability of the data. Another factor that makes the NoSQL databases popular is the flexibility to use dynamic data on the fly without worrying about the data structure so much.

The major difference between the NoSQL databases and the SQL databases is that the traditional SQL databases use the row-column structure while the MongoDB, which is the NoSQL database, uses a rich data document model to store the data, thus allows any type of data for storage.

MongoDB removes the complex **Object-Relation Mapping (ORM)** layer and uses a flexible data model which helps in evolving the database with the business changing requirements. So, MongoDB adapts faster to the business requirements than the SQL databases.

We will cover more about the MongoDB vocabulary in the last topic of this chapter, the following table shows the basic comparison between the NoSQL database and the SQL database.

Basic comparison between the NoSQL database and the SQL database

SQL database	NoSQL database (MongoDB)
Relational database	Non-relational database
Supports SQL query language	Supports JSON query language
Table based	A collection based and key-value pair
Row-based	Document-based
Column-based	Field-based
Support for foreign key	No support for foreign key
Support for triggers	No Support for triggers
Contains a predefined schema	Contains a dynamic schema
Not fit for hierarchical data storage	Best fit for hierarchical data storage
Vertically scalable - increases RAM	Horizontally scalable - adds more servers
Emphasizes ACID properties (Atomicity, Consistency, Isolation, and Durability)	Emphasizes the CAP theorem (Consistency, Availability, and Partition tolerance)

**Table 1.1: Basic Comparison between the NoSQL and SQL Databases**

Now, we can easily compare and differentiate between the core terms and concepts which are used in the SQL databases and the NoSQL databases.

## **The concept of NoSQL databases**

The NoSQL databases comprise a wide variety of database-related technologies to meet the dynamic demands of modern applications.

Today's developers work on various data types which can be structured, semi-structured, unstructured, or polymorphic. Relational databases are not designed to meet the demands of these applications and development environments. So, the developers feel stuck at some point in the development cycle, whether during the decision on the architecture of the application or during the scaling of the application at a later stage of the deployment cycle.

NoSQL databases, like MongoDB, play a very important role to allow the developers and the development teams to focus on the application programming part without worrying a lot about the data complexities.

Therefore, the developers feel a lot of flexibility and scalability. There are a lot of benefits that we can achieve when we use NoSQL databases as these are more scalable and give better performances.

So, for the contemporary development environment, which is rapid and uses agile methodologies, quick iterations, and frequent code pushes, the developers feel a lot easier to use NoSQL databases, like MongoDB in order to meet their demands.

## **Types of NoSQL database management systems**

There are four major types of NoSQL database management systems. These are classified as follows.

- Key-value paired databases
- Column-oriented databases
- Document-oriented databases
- Graph databases

Let us look at them one by one.

### **Key-value paired databases**

These are the simplest NoSQL databases. Here, the data is stored as a name (or key) with its value. Some key-value paired databases also have the feature to give the data type such as `int` or `float`.

Examples of key-value paired databases are:

- Berkeley DB
- Redis

### **Column-oriented databases**

These types of NoSQL databases allow you to store the columns of data instead of rows. These are very effective in handling the queries

of large data sets.

Examples of column-oriented databases are:

- Cassandra
- HBase

## **Document databases**

These databases pair each key with a data structure called to document and these documents then contain key-value pairs, key-array pairs, and nested documents.

Examples of document databases are:

- MongoDB
- Couchbase

## **Graph databases**

These databases are useful in storing data that are inter-connected as nodes just like a graph. These databases add an extra layer of highlighting the relationship among the documents.

Examples of document databases are:

- OrientDB
- Neo4j

Till now, we have studied different types of NoSQL databases, their advantages, and which databases come under this category.

## **Introduction to MongoDB basics, core concepts, and vocabulary**

As we have acquired a good knowledge about the NoSQL database and MongoDB, now let us start with some basic concepts and terminology used specifically in MongoDB.

## **Document database**

We have learned in the previous topic that MongoDB is a NoSQL database. MongoDB documents are similar to JSON objects and these documents can contain arrays, other documents, or an array of documents.

## Collections

The MongoDB documents are stored in collections. So you can consider this as a table with respect to the relational database.

## Support for rich query language

MongoDB uses rich query language to support the CRUD (Create, Read, Update, and Delete) operations. It also supports the queries related to data aggregation, search operations, as well as geospatial queries.

## Support multiple storage engines

MongoDB supports multiple storage engines like:

- WiredTiger storage engine
- In-memory storage engine

MongoDB provides a pluggable storage engine, API, that lets the third parties develop their own engine type for MongoDB.

We will cover MongoDB storage engines in detail in the advanced chapters of this book.

## Some basic MongoDB terminology

The following are the basic MongoDB specific terms we will use consistently in this book throughout all the chapters:

- **MongoDB Database:** A single MongoDB server consists of various databases where each database is a physical container of collections.
- **MongoDB Collection:** A collection in MongoDB is equivalent to a database table of the SQL-based databases and it exists

within a single database. It includes a group of MongoDB documents.

- **MongoDB Document:** A document can be defined as an instance of a MongoDB collection. It includes a set of key-value pairs. All the documents include a dynamic schema which means the documents that comprise of the same collection do not need to have the same set of fields and structure.

We have now covered the basic concepts, vocabulary, and terminology of MongoDB and understood the MongoDB specific terms. In the next topic, we will compare these terms to the SQL databases so that you will easily understand them.

## [\*\*MongoDB terminology comparison with SQL databases\*\*](#)

The following table compares the MongoDB terms and concepts with the SQL based databases:

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON (Binary JSON) document
column	field
index	index
table joins	embedded documents
primary key	primary key
Specify any unique column or column combination as the primary key.	The primary key is automatically set to the <code>_id</code> field.

*Table 1.2: Basic MongoDB terms*

We have compared the MongoDB terms to the SQL databases and so you should now get a clear idea about the core terms and

concepts used in MongoDB. It will be helpful for readers as they will progress to the next chapter of this book.

## **Conclusion**

In this chapter, we covered the basics of MongoDB including its definition. We covered the concept of NoSQL databases and how these are different from SQL databases.

We also covered the concepts of document-based databases. We also covered the architecture of MongoDB with the help of diagrams. We also covered different types of NoSQL databases and the specific terminologies of MongoDB, the vocabulary used, and also compared the MongoDB specific terms with the SQL databases.

In the next chapter, we will cover how to download and install MongoDB on Windows OS and how to verify that MongoDB is correctly installed on your Windows-based machine. We will also cover how to connect to MongoDB.

## **Questions**

1. What is MongoDB?
2. How is MongoDB different from SQL-based databases?
3. Explain the MongoDB architecture.
4. What is a document database?
5. What is a NoSQL database?
6. Compare the MongoDB collections and documents with respect to the SQL based databases?

## CHAPTER 2

# MongoDB Installation Setup on Windows

This chapter covers the installation and steps to set up MongoDB on Windows powered machines. It will also cover the post-installation checks and the overview of MongoDB editions. You will learn how to download MongoDB for Windows and how to install it correctly on your Windows machine. This chapter also covers the Enterprise Advanced Edition of MongoDB and advantages of using it. You will get step-by-step explanation with screenshots to make you understand the installation and setup of MongoDB very easily on Windows operated machines and also covers the post-installation steps to easily verify if MongoDB is correctly installed on your Windows system.

### Structure

In this chapter, we will discuss the following topics:

- Overview of MongoDB editions
- MongoDB setup on Windows
- Checking the installation
- Connecting to MongoDB

### Objectives

After studying this unit, you should be able to understand the different editions of MongoDB and also learn the steps to install MongoDB on your Windows operated machine. Later in this chapter you will be able to learn how to check if MongoDB has been installed correctly on your Windows operated machine. In the last section of this chapter you will learn how to connect to MongoDB and post-installation verification checks for MongoDB installation on your Windows operated machine.

## Overview of MongoDB editions

MongoDB comes with two editions. The first one is the MongoDB Community Edition, which is an open source edition and is free. The second one is an Enterprise Advanced Edition, which is a paid version and comes with more advanced features that are not available in the Community Edition.

MongoDB Community License is available in 2 variants:

- **Server side public license:** This license is applicable to all the MongoDB Community Edition versions released after October 16, 2018 (including patch fixes for prior versions).
- **Free software foundation's GNU AGPL v3.0:** This license is applicable to all the MongoDB Community Edition versions released prior to October 16, 2018.

Therefore, we can see that all the MongoDB Community Edition versions which are released after October 16, 2018 come with a new Server Side Public License which is applicable to the current version of MongoDB.

## Features of MongoDB Enterprise Advanced Edition

MongoDB Enterprise Advanced Edition has lot of functionalities which are very beneficial, especially if you or your organization is developing any hi-end application that needs some advanced requirements and support. In this scenario, it is recommended to take the benefits of the MongoDB Enterprise Advanced edition to save your development, **R&D (research and development)** and production costs.

Let us now read about the features of the Enterprise Advance Edition of MongoDB:

- **Proactive support:** The MongoDB Enterprise Advanced Edition comes with a proactive support from MongoDB Inc. The support is available 24\*7\*365 globally. It includes the emergency fixes, guidance on upgradations, deployments, scalability, and optimizations.
- **Better management:** The MongoDB Enterprise Advanced Edition comes with the tool called Ops Manager, manages MongoDB in a

better way. This tool helps to monitor, automate, and backup processes related to MongoDB.

- **Kubernetes integration:** The MongoDB Enterprise Advanced Edition comes with the support and integration for Kubernetes, which is an open-source solution for automating application deployment.
- **Better security:** The MongoDB Enterprise Advanced Edition is built on the Advanced Security framework and comes with many high level security features such as Role-Based Access Controls, **Public Key Infrastructure (PKI) certificates** which (a technology used to authenticate users and devices), **Transport Layer Security (TLS)**, and Client-Side Field Level Encryption. This edition also comes with MongoDB encrypted storage engine which helps in keeping the data secure without using the third-party solutions.
- **Better analytics and visualization:** The MongoDB Enterprise Advanced Edition comes with the feature of the MongoDB charts which is one of the best ways to visualize the MongoDB data.
- **Availability of other storage engines:** The MongoDB Enterprise Advanced Edition comes with in-built support for the in-memory storage engine which helps us to increase the speed and response time.
- **Enterprise software integration:** The MongoDB Enterprise Advanced Edition supports the integration with the other enterprise monitoring and management tools specific to an organization, and thus, provides more flexibility to use MongoDB according to their processes and IT infrastructure.
- **Commercial license:** The MongoDB Enterprise Advanced Edition offers a commercial license to meet the development and distribution needs of an organization that have specific policy requirements.

We can see that there are many enhanced features available in the MongoDB Enterprise Advanced Edition and their benefits. In the next topic, we will compare the MongoDB Enterprise Advanced Edition with the MongoDB Community Edition.

## Comparison between MongoDB Community Edition and MongoDB Enterprise Advanced Edition

Followed is a detailed comparison of the MongoDB Community Edition with its Enterprise Advanced Version:

Security	MongoDB Community Edition	MongoDB Enterprise Advanced Edition
LDAP Authentication	No	Yes
LDAP Authorization	No	Yes
Kerberos	No	Yes
Auditing	No	Yes
Log Redaction	No	Yes
X509 Authentication	Yes	Yes
SNMP	No	Yes
Engines	MongoDB Community Edition	MongoDB Enterprise Advanced Edition
WiredTiger Engine	Yes	Yes
WiredTiger Encryption	No	Yes
In-Memory Engine	No	Yes
Features	MongoDB Community Edition	MongoDB Enterprise Advanced Edition
Multi Document ACID Transactions	Yes	Yes
Sharding Zones	Yes	Yes
Faster Initial Sync	Yes	Yes
Auto-Balancing	Yes	Yes
Tunable Consistency	Yes	Yes
x64 Support	Yes	Yes

ARM support	Yes	Yes
Views	Yes	Yes
mongoreplay Tool	Yes	Yes
Collation	Yes	Yes
Decimal Type	Yes	Yes
Balancer on Config Primary	Yes	Yes
Parallel Balancing	Yes	Yes
Writes Consistent by Default	Yes	Yes
Linear Read Concern	Yes	Yes
Graph DB Functions	Yes	Yes
Faceted Search	Yes	Yes
Bucketed Aggregation	Yes	Yes
Sort by Bucket Count	Yes	Yes
Counts in Aggregation	Yes	Yes
Improved Aggregation Arrays	Yes	Yes
Improved Aggregation Strings	Yes	Yes
Add Aggregation Flow controls	Yes	Yes
Improved Aggregation Dates	Yes	Yes
collStats in Aggregation	Yes	Yes
Add Aggregation Type operator	Yes	Yes
Improved profiler and currentOp	Yes	Yes
Aggregation Operators for Type Conversion	Yes	Yes
Aggregation String Operators	Yes	Yes
Enhancements to Change Streams	Yes	Yes

**Table 2.1:** MongoDB Community Edition VS MongoDB Enterprise Advanced Edition

Now, you can easily compare what exactly are the differences between the features available in MongoDB Enterprise Advanced Edition and those not available in the Community Edition of MongoDB.

## MongoDB setup on Windows

Let us explore how we can download, install, and setup the MongoDB Community Edition on machines that run on Windows OS.

### Installing MongoDB Community Edition on Windows operating system

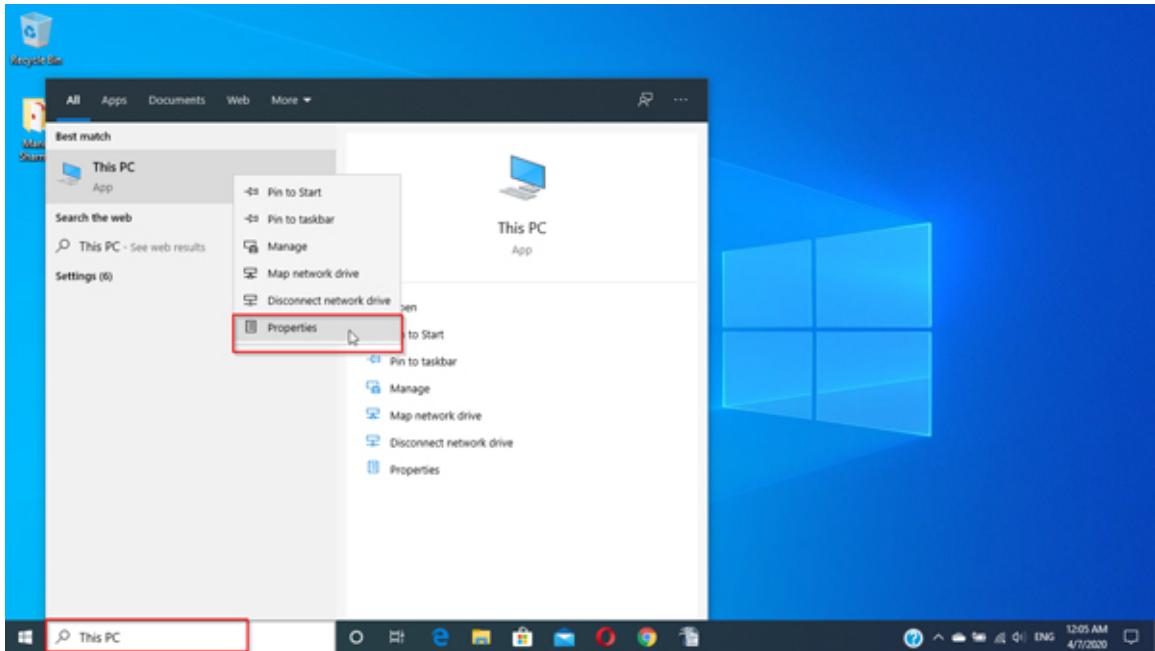
MongoDB installation is available for Windows, Linux, and MacOS. This chapter focuses on installing MongoDB on Windows 10. The installation process on Linux and macOS will be discussed in detail in the later chapters.

We will show you how you can install the MongoDB Community Edition (version 4.4) on Windows operating system. We will use the default installation method to install the MongoDB Community Edition on the machines that run on the Windows operating system.

Here, we will cover the MongoDB Community Edition (version 4.4) for 64-bit versions of Windows on x86\_64 architecture. This includes the following version of the Windows operating system.

- Windows 10 / Windows Server 2016
- Windows Server 2019

In our example, we will use Windows 10 to install and setup MongoDB. To check the version of your operating system, right click on **Properties** of your computer, as shown in the following screenshot:



**Figure 2.1: Selecting Properties of your Computer**

You can also check your Windows version by clicking on **Properties**, as shown in the following screenshot:

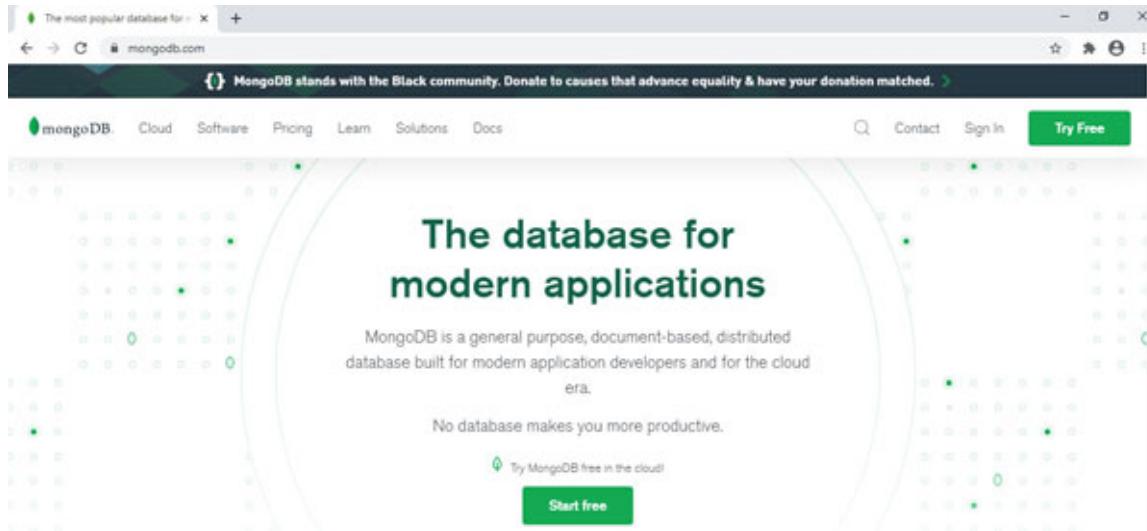


**Figure 2.2: Checking your Windows OS Version**

## Installation steps

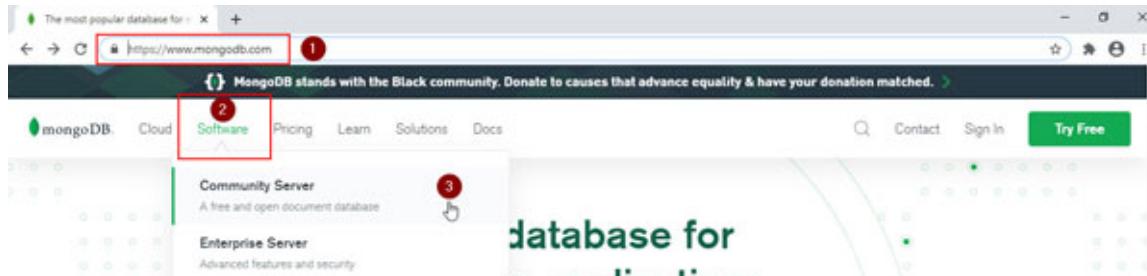
### **Step 1 – Download the MongoDB Community Edition**

1. Open MongoDB Inc. official website – <https://www.mongodb.com/> in your favorite browser, as shown in the following screenshot:



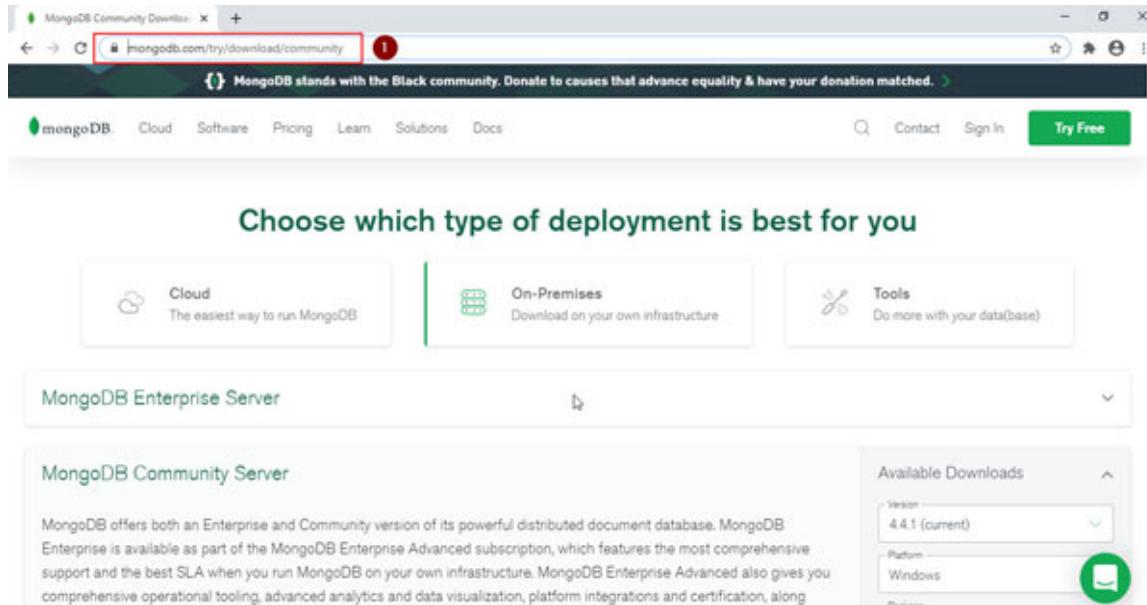
**Figure 2.3:** MongoDB Inc. Official Website Home Page

2. Click on the **Software** link on the top navigation bar of the website and then click on the **Community Server** link from the **Software** dropdown, as shown in the following screenshot:



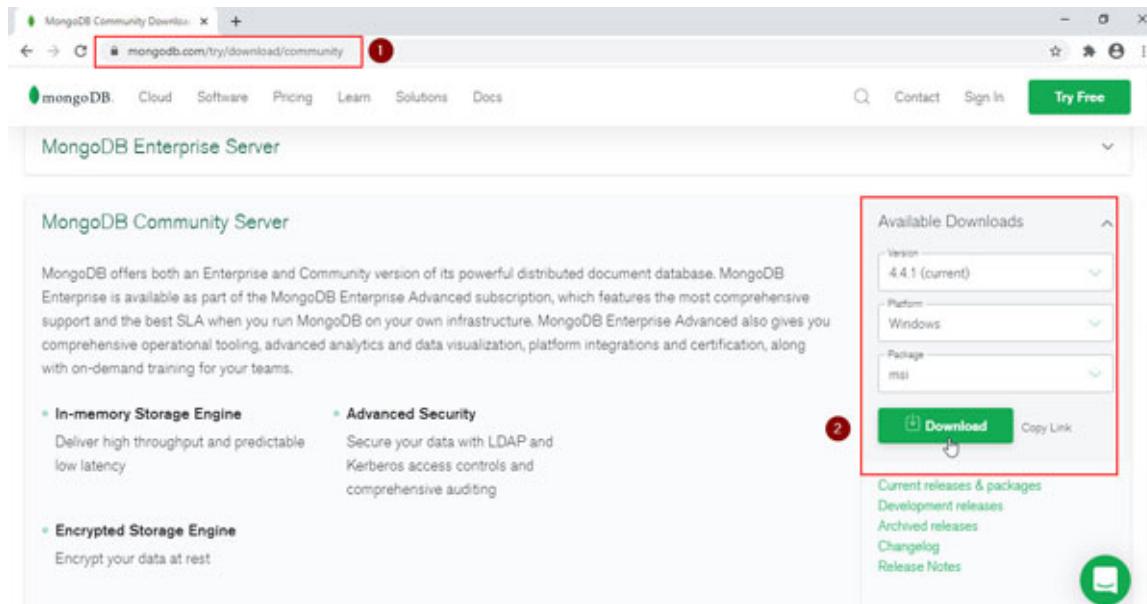
**Figure 2.4:** MongoDB Inc. Official Website Home Page – Community Server

3. This will open a **Download** page from where you can download the MongoDB server, as shown in the following screenshot:



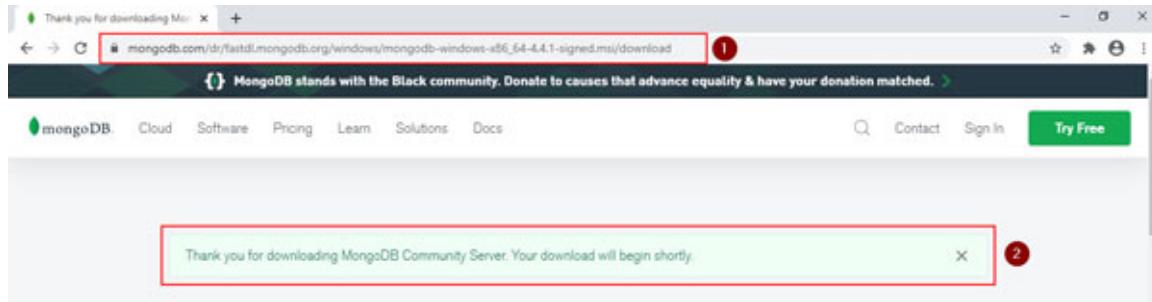
**Figure 2.5:** MongoDB Inc. Official Download Page for Community Edition

4. In this page, you will see a **Download** section where you will see the **Available Downloads** section. This section will auto detect your OS type, as shown in the following screenshot:



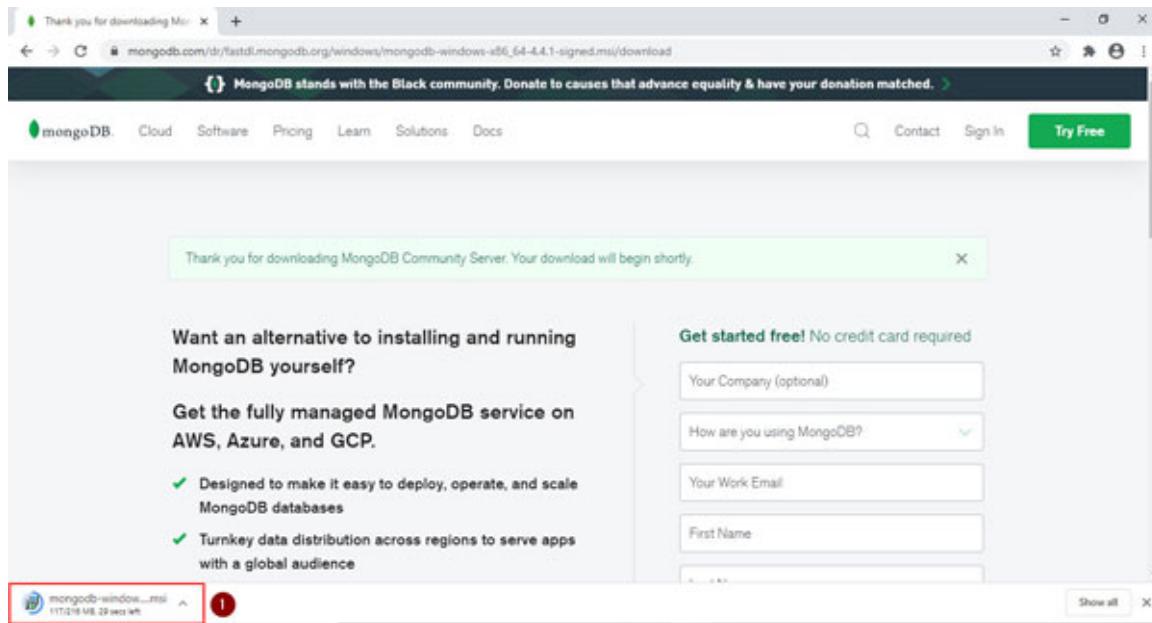
**Figure 2.6:** MongoDB Inc. Official Download Center – MongoDB Server Download Screen

5. Now, click the **Download** button to download the MongoDB Community Edition installer. The download will start automatically, as shown in the following screenshot:



**Figure 2.7: MongoDB Inc. Official Download Center – MongoDB Server Download – Thank You Page**

- Once the download starts, you can easily see the process with the download icon and progress on your browser (this progress is displayed differently in different browsers). Wait until it is 100% complete, as shown in the following screenshot:



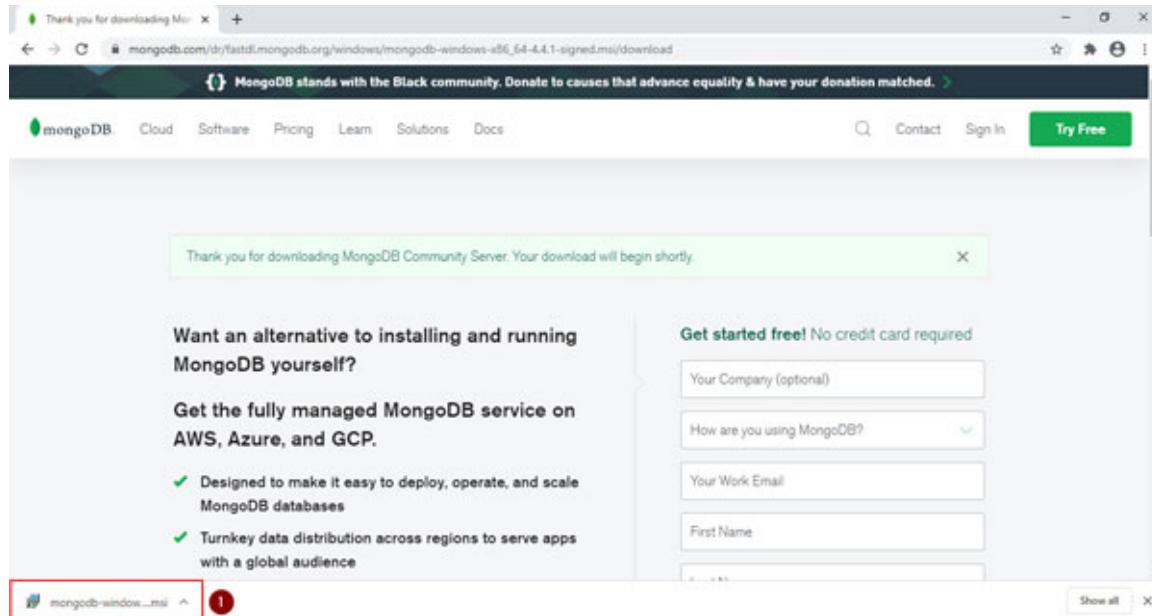
**Figure 2.8: MongoDB Download Process – MongoDB Server Download Started**

- After the installer is downloaded, follow installation process as defined in step 2 – *Installing the MongoDB Community Edition on Windows 2010* as follows, as shown in the following screenshot:

Here, we have covered how to download MongoDB from the official website. Next step is to install the MongoDB Community Server on Windows 2010.

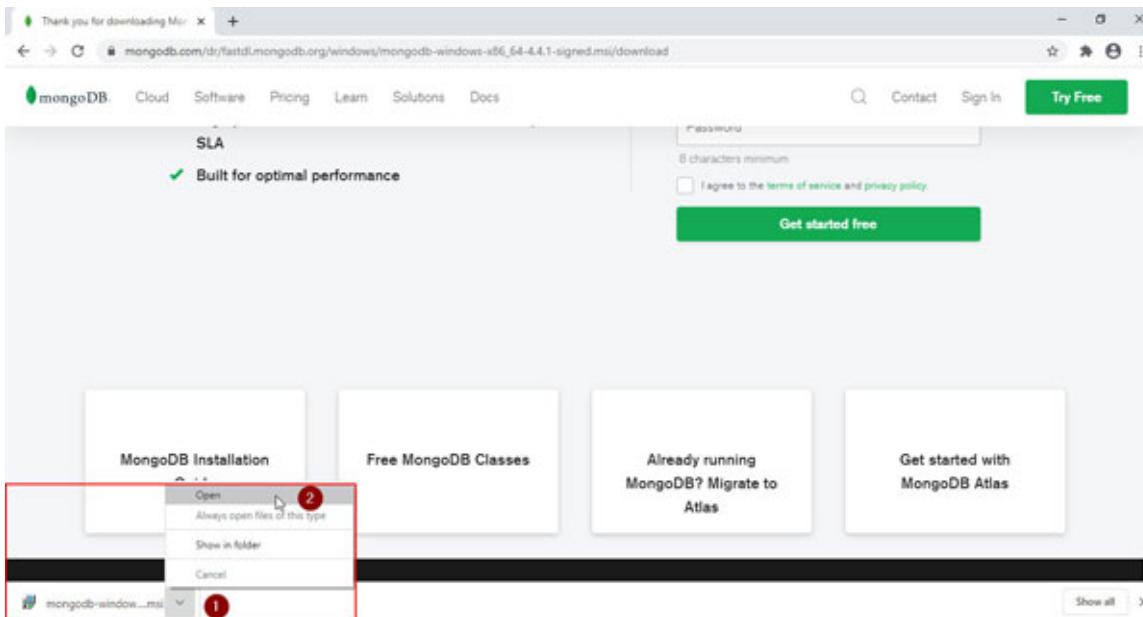
## **Step 2 – Installing MongoDB Community Edition on Windows 2010**

Once the download is complete and the installer file (MSI file) is fully downloaded, it will show a download complete icon, as shown in the following screenshot and you can proceed further:



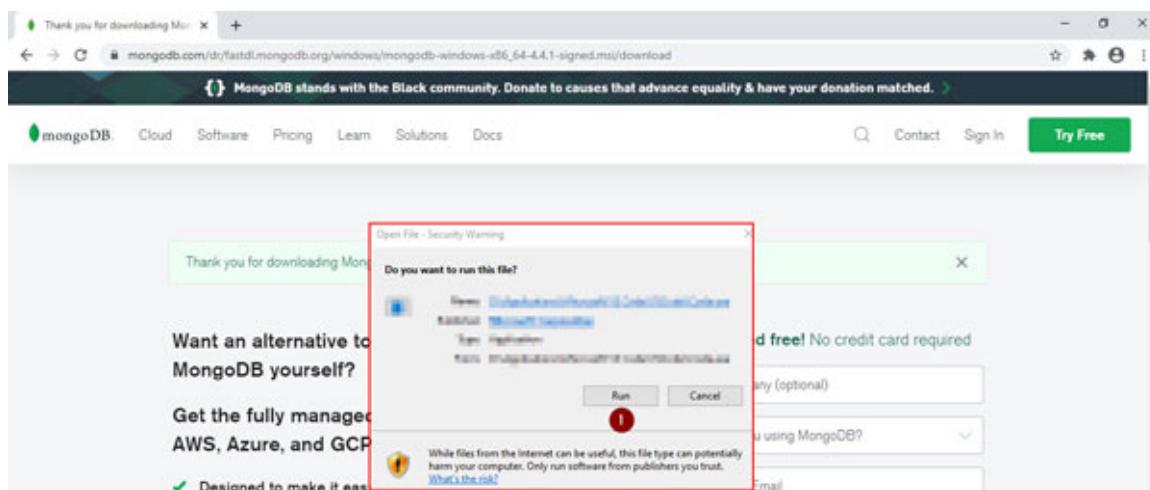
**Figure 2.9: MongoDB Download Process – MongoDB Server Download Complete**

1. Now, open this MSI file. It is a Windows installer file, previously known as Microsoft installer, and it will start the MongoDB setup wizard which will guide you to complete the installation of MongoDB in your machine, as shown in the following screenshot:



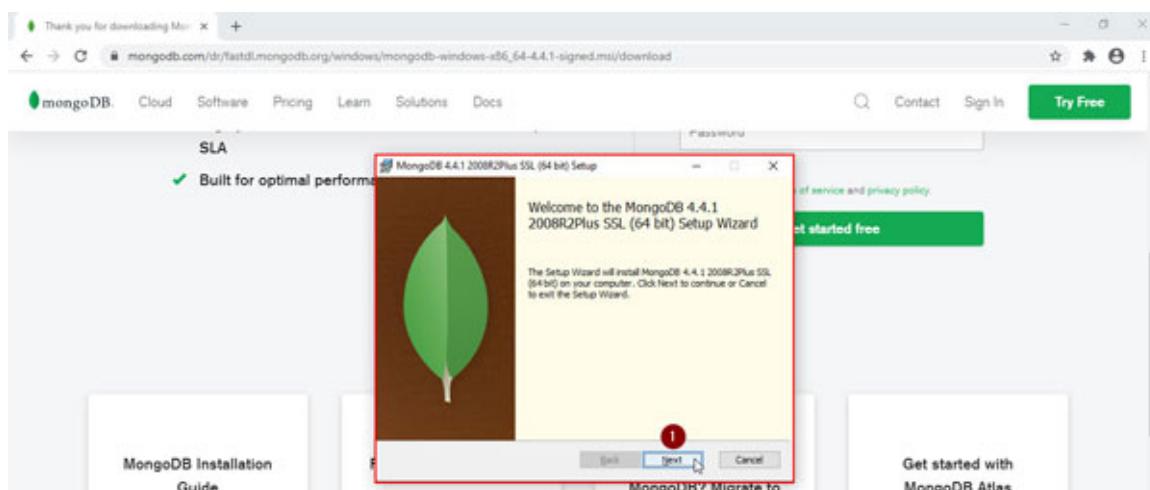
**Figure 2.10: MongoDB Download Process – Open MongoDB MSI File.**

2. MongoDB installation requires administrative privileges. Usually, if you are using single user machine, it comes with you by default. But, during installation of any software on Windows, the prompt, as shown in the following screenshot, appears and asks you to **Run** this setup application or **Cancel** the application setup wizard. To Install MongoDB on your system, click **Run**:



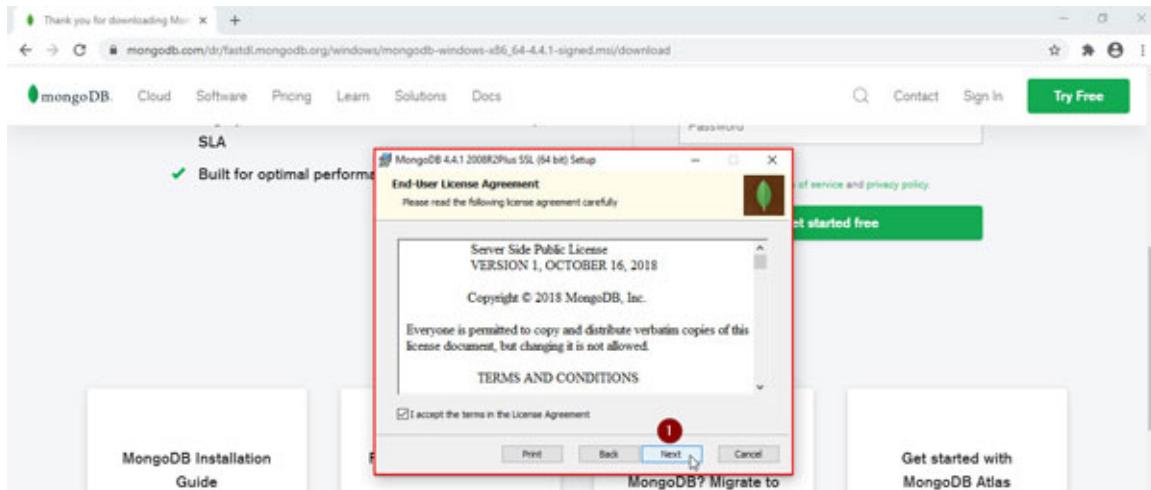
**Figure 2.11: MongoDB Installation Wizard – Windows Security Prompt**

3. Once you click on **Run** you will see the startup screen of the installation wizard. You will also see some buttons that are easy to understand. You may click the **Cancel** button at any point of time to stop the installation or you may click the **Back** button to go back to the previous steps. In order to install MongoDB on your system, press the **Next** button, as shown in the following screenshot:



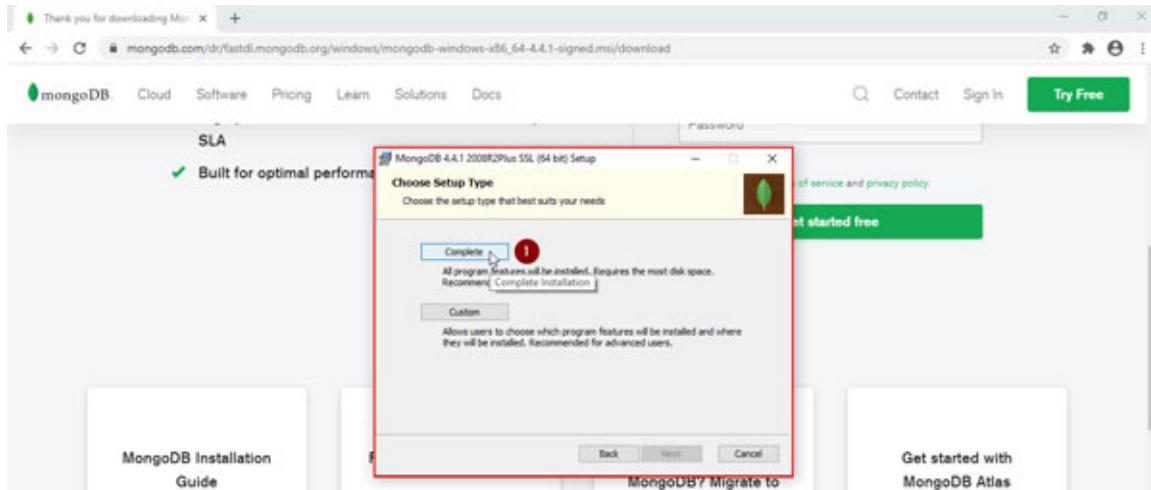
**Figure 2.12: MongoDB Installation Wizard – Startup Screen**

4. Once you click the **Next** button, you will see the next screen of the installation wizard which will display the **MongoDB Server Side Public License and Terms and Conditions**. It is recommended to read these **Terms and Conditions** before you follow the next steps. After reading this, if you agree with MongoDB Inc. license and its terms and conditions, click on the checkbox where it says **I accept the terms of License Agreement**, so that it will enable the **Next** button of this screen and you can move further with the installation process of MongoDB. Now, click the **Next** button to continue with the installation process, as shown in the following screenshot:



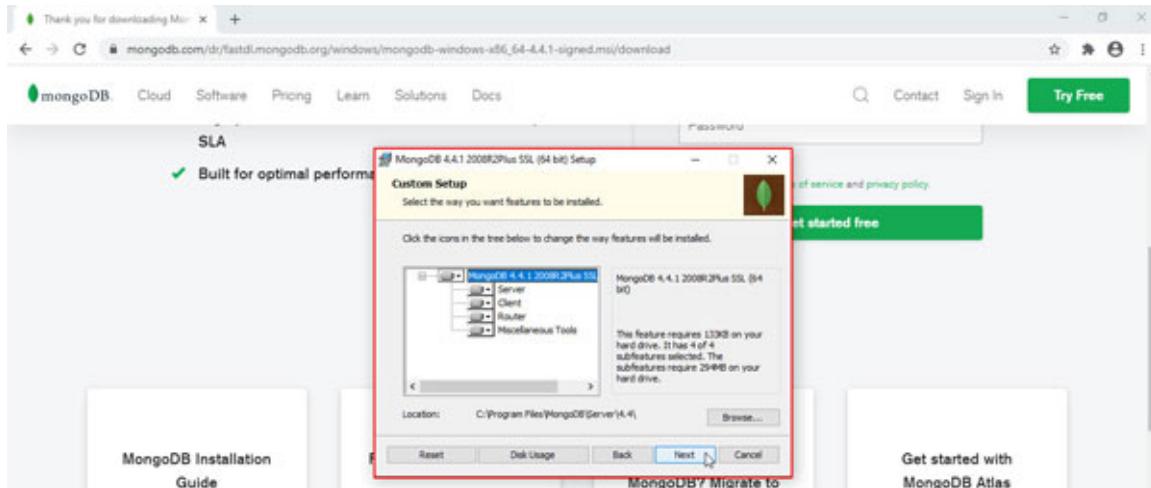
**Figure 2.13: MongoDB Installation Wizard –MongoDB License and Terms and Agreement Screen**

5. Once you click on the **Next** button, you will see the next screen of the installation wizard. Here, you will have the following two options:
  - Complete install, as shown in [Figure 2.14](#)
  - Custom install, which has more options to choose, as shown in [Figure 2.15](#):



**Figure 2.14:** MongoDB Installation Wizard – MongoDB Complete Install

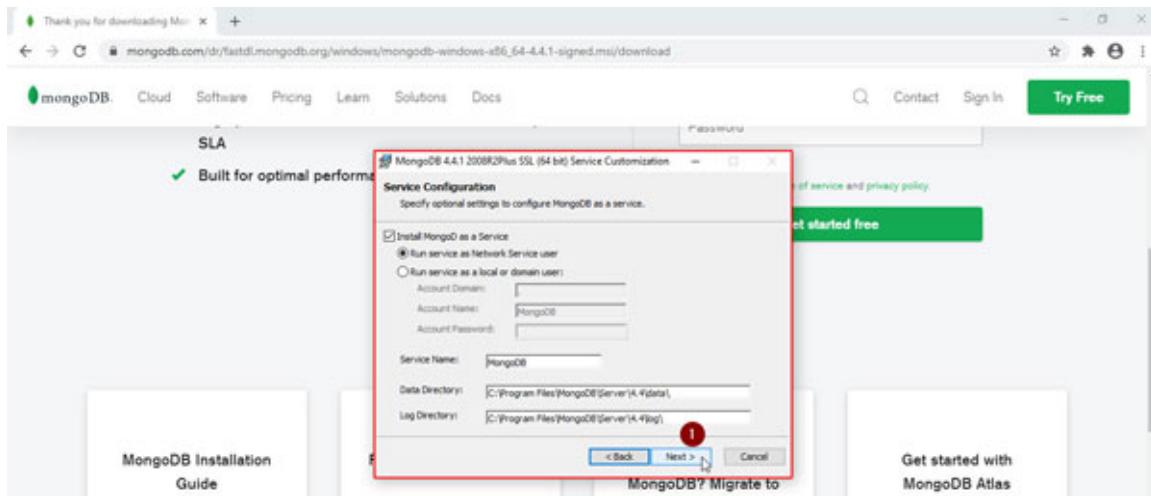
- Once you click the `Custom Install` button, you will see the screen, as shown in the following screenshot. In this screen, you will have multiple options to choose the Program related to MongoDB, like client, server, monitoring tools, etc. You can also select the default directory of your choice where these files will be copied in your Windows machine. Click on the `Back` button to go to the previous screen, as we will do to complete the installation of MongoDB with its default settings:



**Figure 2.15:** MongoDB Installation Wizard – MongoDB Custom Install

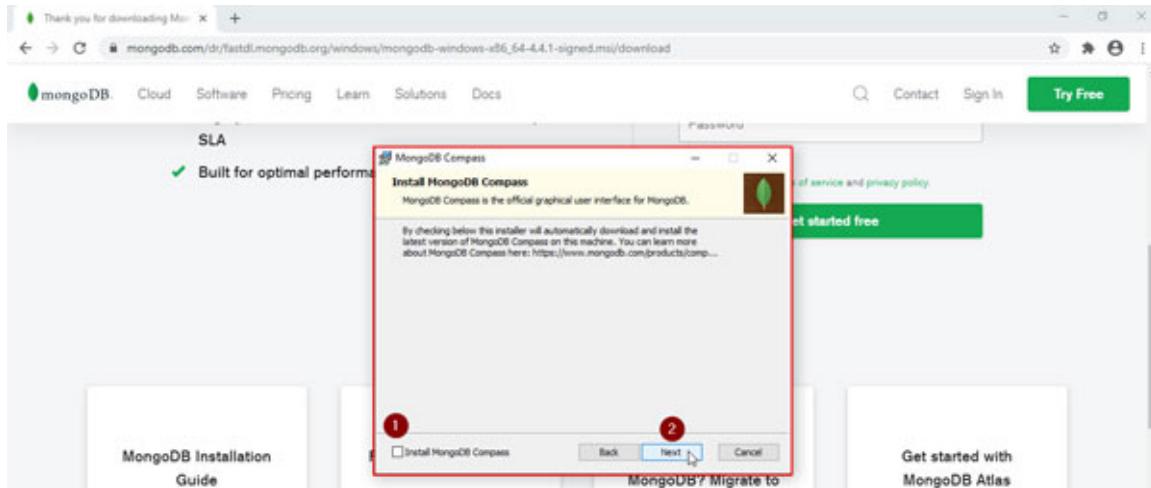
- Once you click the `Back` button, it will open the same screen, as shown in the preceding screenshot). Click the `complete` button and you will be taken to the next screen, as shown in the following screenshot. In this screen, you will see few options related to the

MongoDB service and its related `option` on how we need to start MongoDB on our machine. Another setting is related to the `Data and Log Directory Paths` which you can change if you want these to be stored in some other locations on your machine. In our example, we will keep these where they are and not change anything. You can now click the **Next** button and the next screen will appear for this MongoDB installation wizard:



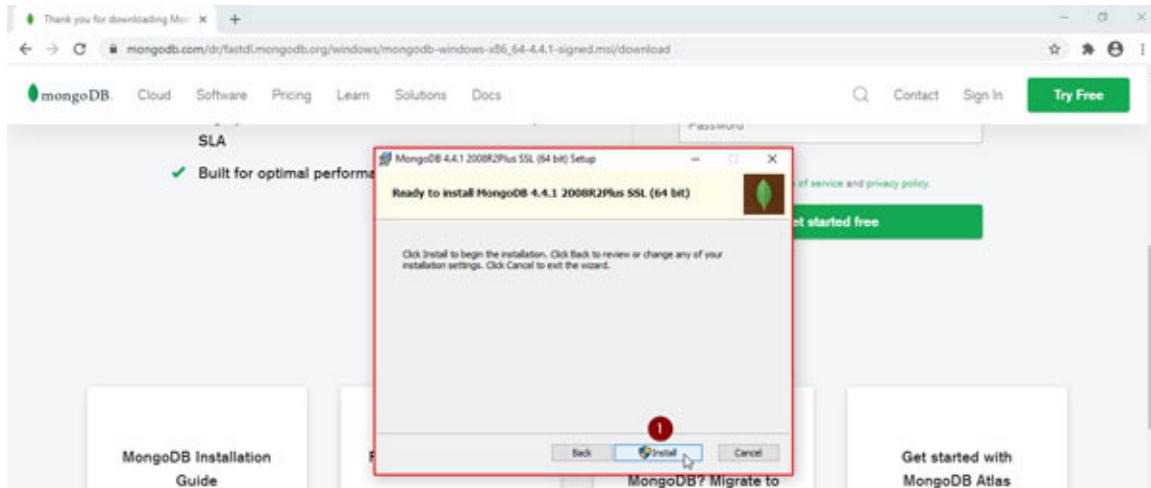
**Figure 2.16: MongoDB Installation Wizard – MongoDB Complete Install (Service Configuration Settings)**

Once you click **Next** button, it will open the new screen, as shown in the following screenshot. There, you will see a checkbox which will be checked by default and it is related to the installation of MongoDB compass. If that remains checked, this MongoDB setup wizard will also download and install the MongoDB Compass along with this wizard. We don't want to install the MongoDB Compass in this chapter, since we will cover the MongoDB Compass separately in the intermediate chapter ([Chapter 16: MongoDB Data Manipulations using MongoDB Compass](#)) of this book. Therefore, you need to uncheck this for now and press the **Next** button:



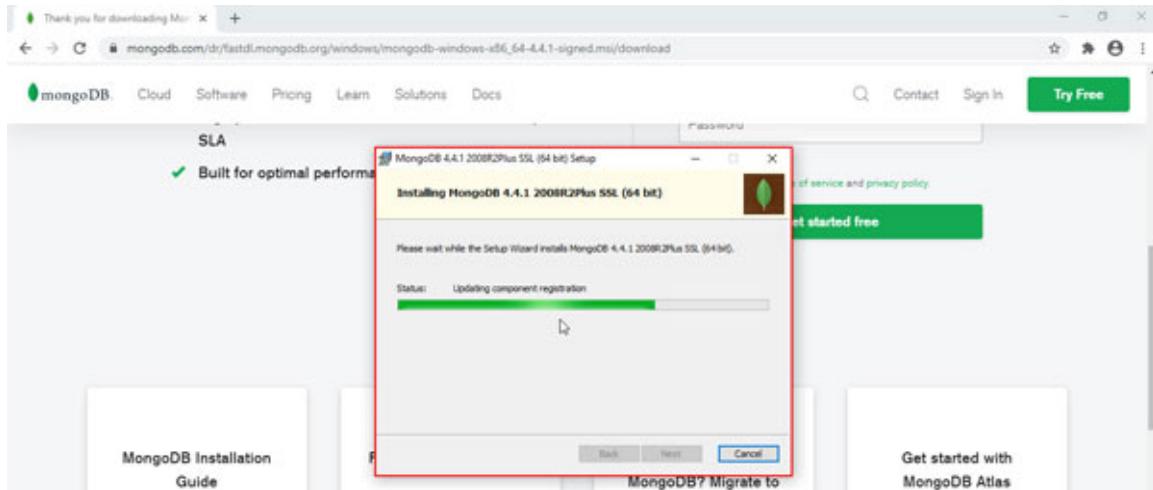
**Figure 2.17:** MongoDB Installation Wizard – MongoDB Complete Install (Install Compass Checkbox)

- Once you click the **Next** button, it will open a new screen. Here, you will see the **Install** button. To start the MongoDB installation, click the **Install** button, as shown in the following screenshot:



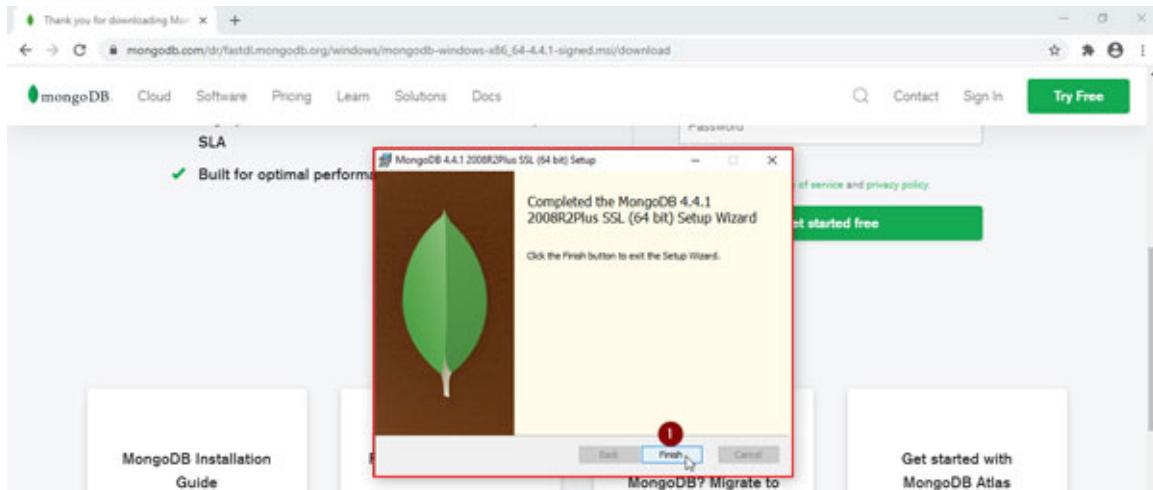
**Figure 2.18:** MongoDB Installation Wizard – MongoDB Complete Install (Install Button)

- Once you click the **Install** button, it will open the new screen and in this screen, MongoDB will copy the files in your machine and configure it on your Windows machine. You can also see the progress of the installation and setup, as shown in the following screenshot:



**Figure 2.19:** MongoDB Installation Wizard – MongoDB Complete Install (Install Button)

10. Once this installation is complete, it will display the next screen and a `Finish` button will appear, as shown in the following screenshot. Once you click the `Finish` button, the installation will be finished:

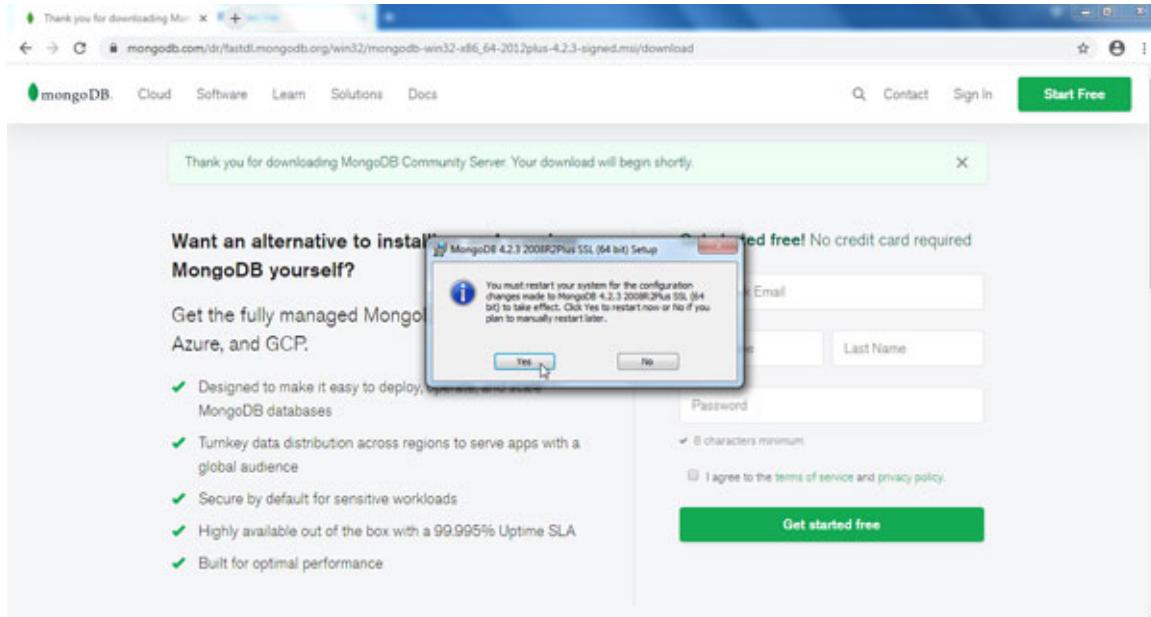


**Figure 2.20:** MongoDB Installation Wizard – MongoDB Complete Install (Finish Button)

11. Once you click the `Finish` button, sometimes, it asks you to restart your machine to fully complete the installation. You will get a prompt to restart your computer (you might get this message if you are using previous version of Windows such as Windows 7 and if you are installing MongoDB versions below 4.4), as shown in the

following screenshot. You should always restart your system in order to use MongoDB on your machine without any issues.

12. If you are installing MongoDB version 4.4 on Windows 10, as we are doing in our case, you might not get this message and you can ignore this step:

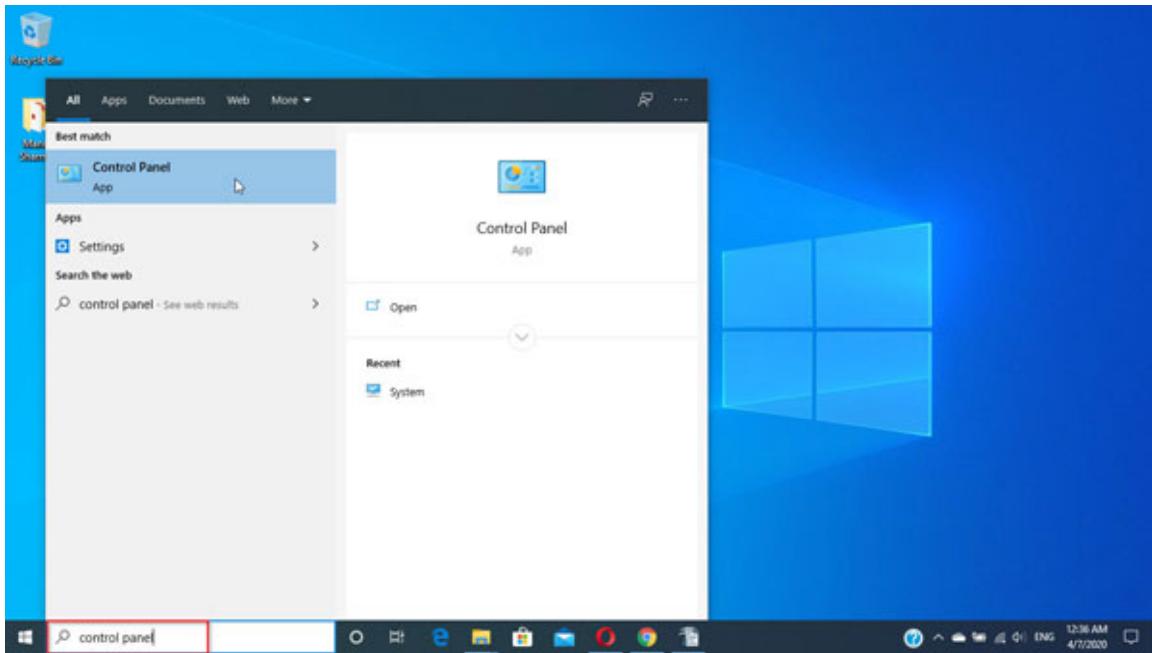


**Figure 2.21: MongoDB Installation Wizard – MongoDB Complete Install (Restart your System Prompt)**

### Step 3 – Post Installation Checks

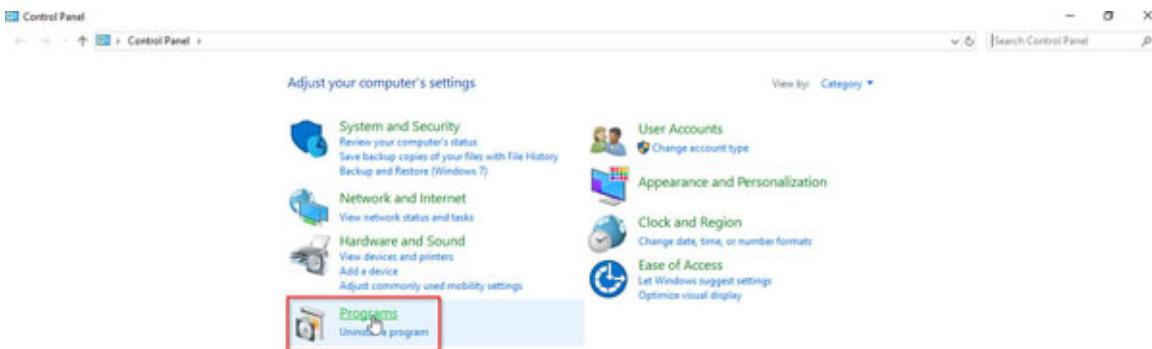
To ensure that MongoDB has been correctly installed in your Windows machine, follow these steps:

1. Once your machine restarts, you can now verify if the installation of MongoDB is correctly done on your machine. To check this, click the start button of your Windows machine and open Windows Control Panel, as shown in the following screenshot:



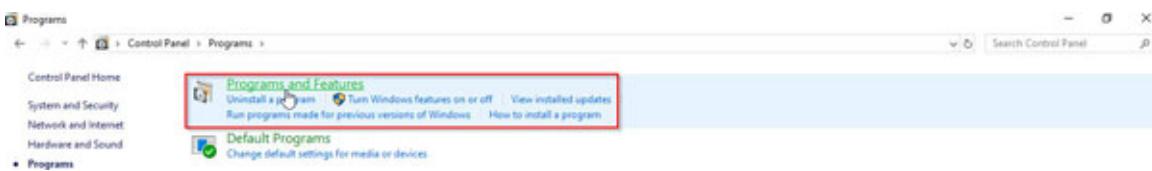
**Figure 2.22:** Click Start Button and then Open Windows Control Panel.

2. Once you are in the Windows Control Panel click on **Programs**, as shown in the following screenshot:



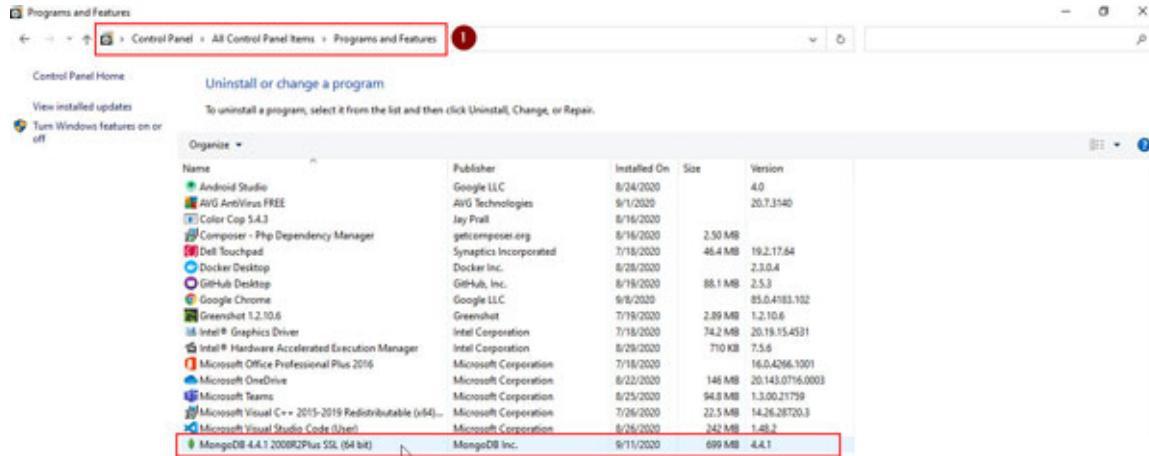
**Figure 2.23:** Windows Control Panel – Programs

3. After you click on the **Programs** icon, another window will open with two **options**. Here, you need to click on the Program and Features icon, as shown in the following screenshot:



**Figure 2.24:** Windows Control Panel – Programs and Features

4. After clicking on **Programs and Features**, another window will open with the list of all the programs installed in your system. If your MongoDB has been installed successfully, you will be easily able to see it here, as shown in the following screenshot:



**Figure 2.25: Windows Control Panel – Installed Programs List**

Till now, we have covered how to verify if MongoDB is installed correctly in your Windows machine in a step-by-step manner. As we have done the default installation of MongoDB here, the installer creates an automatic MongoDB service in Windows and now we can connect to MongoDB using MongoDB Shell.

## Step 4 – Connecting to MongoDB on Windows

Let us now try to connect MongoDB from command line from your Windows machine. To connect to MongoDB from your Windows machine, follow these steps:

1. Open a command window and navigate to the `bin` directory of MongoDB.

The path could be as follows:

```
C:\Program Files\MongoDB\Server\4.4\bin
```

It is shown in the following screenshot:



**Figure 2.26: From Command Line – Navigate to MongoDB "bin" Directory.**

2. Now give the following command, as shown in the following screenshot: `mongo` and press *Enter*.

```
C:\Program Files\MongoDB\Server\4.4\bin>mongo
```

**Figure 2.27:** From Command Line –Type "mongo" command and Press enter

3. Once you run the preceding command and press *Enter*, it will open Mongo Shell and we can type Mongo related commands, as shown in the following screenshot:

```
C:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d739a88c-5443-4232-b535-358eb49da7f5") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-09-11T13:58:48.644+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
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()
```

**Figure 2.28:** MongoDB Shell

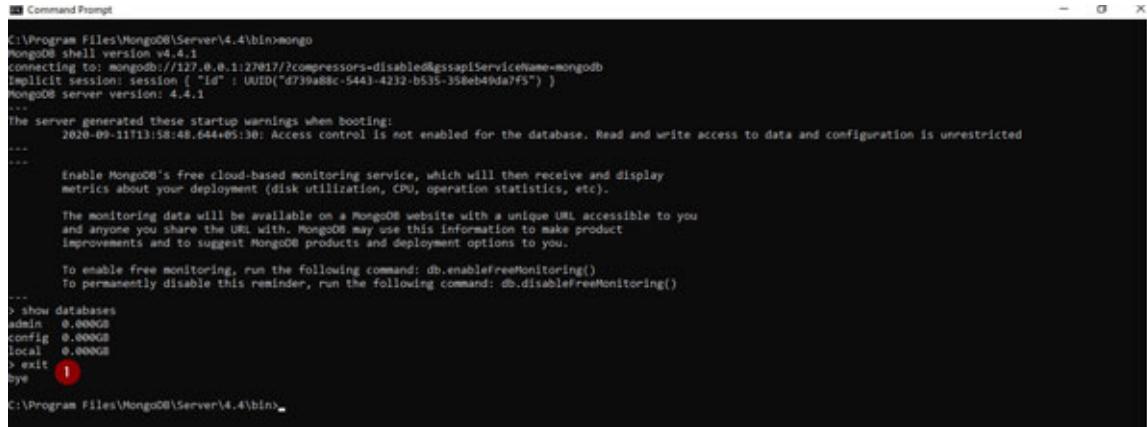
4. To test further, just issue the `show databases` Mongo command and press *Enter*, as shown in the following screenshot:

```
C:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d739a88c-5443-4232-b535-358eb49da7f5") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-09-11T13:58:48.644+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
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()
...
> show databases
admin 0.0000GB
config 0.0000GB
local 0.0000GB
v
```

**Figure 2.29:** Mongo Shell – "show databases" MongoDB Command

5. To exit from MongoDB Shell type, click on the `exit` command and press the *Enter* key:

This will take you out from the MongoDB Shell, as shown in the following screenshot:



```
C:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d739a88c-5443-4232-b535-350eb49da7f5") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-09-11T13:58:48.644+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
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()
...
> show databases
admin 0.000GB
config 0.000GB
local 0.000GB
> exit
bye
```

*Figure 2.30: Mongo Shell – "exit" MongoDB Command – To Exit from MongoDB*

We now learned how to install MongoDB on Windows 10 and verify MongoDB installation and at last we learned how to connect to the MongoDB Shell.

## **Conclusion**

MongoDB can be installed by downloading it from the MongoDB Inc. official website and choosing the platform as Windows. We need to run the MSI File (Windows installer file) to run the setup. Once the setup is complete, we can then verify the Installation by the steps mentioned in the post-installation section of this chapter.

You have also gained the knowledge on how to start and use Mongo shell.

In the next chapter, we will cover the installation of MongoDB on Linux-based systems. This would be helpful for the readers who use Linux-based machines.

## **Questions**

1. What is MSI File?
2. From which source should you download the MongoDB setup file?
3. What are the two editions of MongoDB server?
4. How can you login to the MongoDB Shell? Explain the process.
5. List a few MongoDB Shell commands you learned in this chapter.

## CHAPTER 3

# MongoDB Installation and Setup on Linux (Ubuntu)

This chapter covers the installation and steps to set up MongoDB on machines powered by Linux (we have used Ubuntu, the widely used variant of Linux). This chapter will also cover the post installation checks for MongoDB installation for Linux. So, in this chapter, you will learn how to download MongoDB for Linux and how to install it correctly on your Linux operated machine. This chapter covers step-by-step methods explained with screenshots to make you understand the installation and setup of MongoDB very easily on Linux operate machines. This chapter also covers the post installation steps to easily verify if MongoDB is correctly installed on your Linux system.

### Structure

In this chapter, we will discuss the following topics:

- MongoDB setup on Linux operating system (Ubuntu)
- Checking the installation on Linux operating system
- Connecting to MongoDB on Linux operating system

### Objectives

After studying this unit, you should be able to learn the steps to install MongoDB on your Linux operating system and also how to check if MongoDB has been installed correctly on your Linux operating system. Later in the Chapter you will learn that how you can connect to MongoDB for post-installation verification and checks on your Linux operating system.

### MongoDB Setup on Linux

Let us explore how we can download, install, and setup MongoDB Community Edition on the machines running on Linux OS (Ubuntu).

## [Installing MongoDB Community Edition on Linux Operating System](#)

We will show you how to install the MongoDB Community Edition (version 4.2) on Linux operating system (Ubuntu). We will use the default installation method to install the MongoDB Community Edition on the machines that run on Linux operating system.

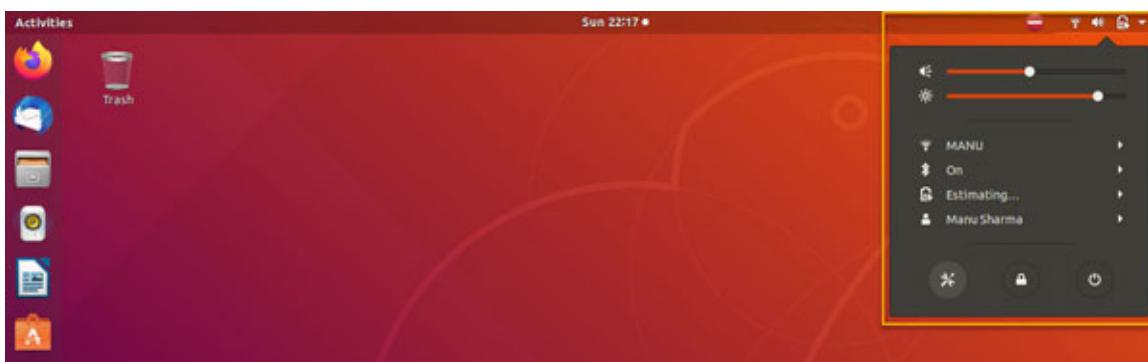
Here, we will cover the MongoDB Community Edition (version 4.2) for 64-bit versions of Ubuntu on x86\_64 architecture. This includes the following version of Linux operating system (Ubuntu):

- 18.04 LTS ("Bionic")
- 16.04 LTS ("Xenial")

Only 64-bit versions of these platforms are supported by MongoDB. In our example, we will use Ubuntu 18.04 to install and setup MongoDB. To check the version of your Linux operating system (Ubuntu), open your system settings by clicking the settings button on the top left corner of your Ubuntu OS (this varies from one Linux variant to another), as shown in the following screenshot:.

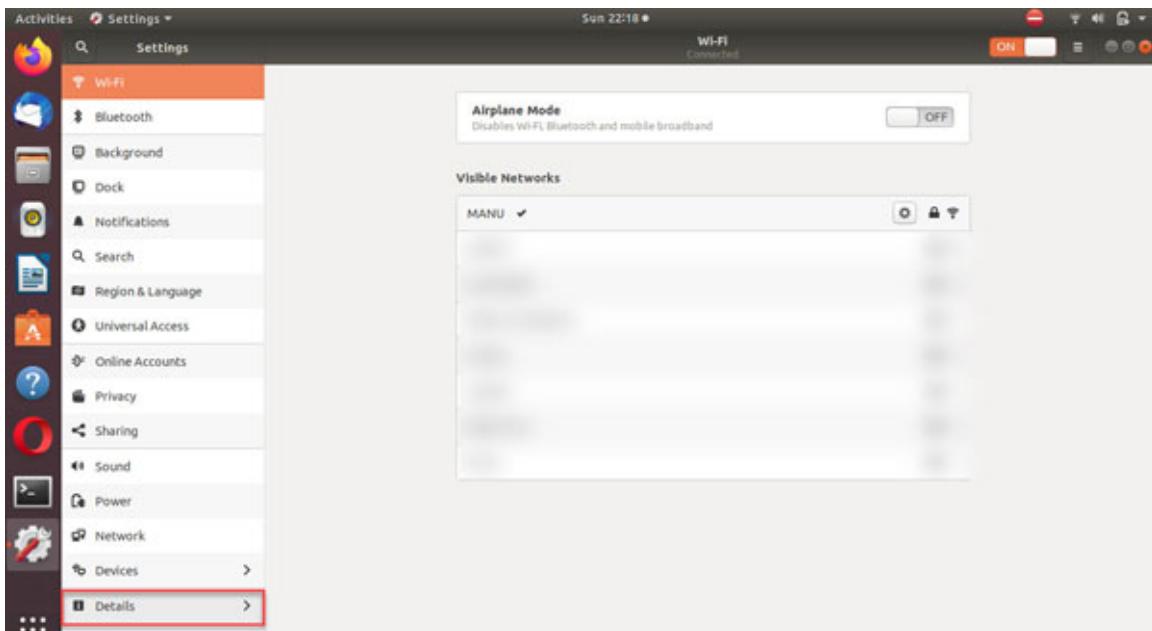
An alternate way is by using the following command on the command line terminal. Follow these steps:

1. Open terminal using the keys [CTRL] + [ALT] + [T]
2. Type the command - `lsb_release-a` and press *Enter*
3. The terminal will show the Ubuntu version.



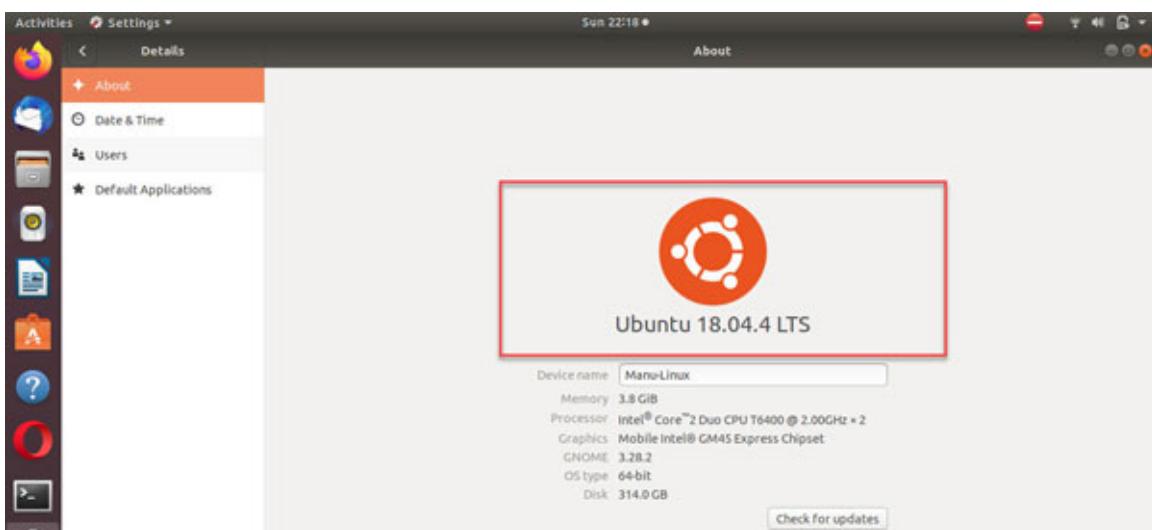
**Figure 3.1:** Selecting system settings of your Ubuntu machine

It will then open another Window with many options. Click on the **Details** option at the bottom left corner of the screen, as shown in the following screenshot:



**Figure 3.2:** Checking your Ubuntu OS version

After this, click on the **Menu** item. You will see the Ubuntu version, as shown in the following screenshot:



**Figure 3.3:** Ubuntu OS version

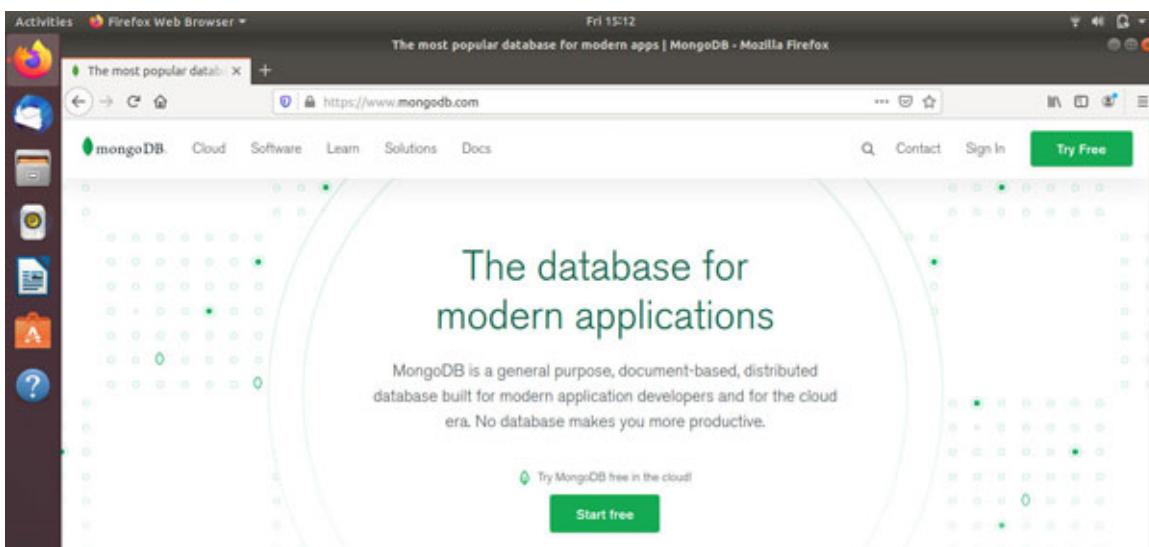
Now, since we have checked our Linux operating system version, we can continue with the installation of MongoDB on our Linux machine.

## Installation steps

### Method one (Browser method)

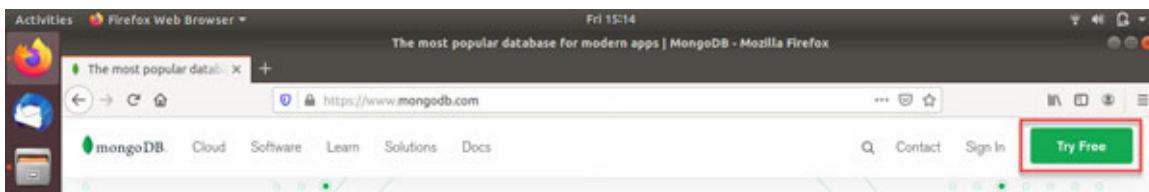
#### Step 1 – Download MongoDB Community Edition

1. Open the official website of MongoDB Inc.– <https://www.mongodb.com/> in your favorite browser, as shown in the following screenshot:



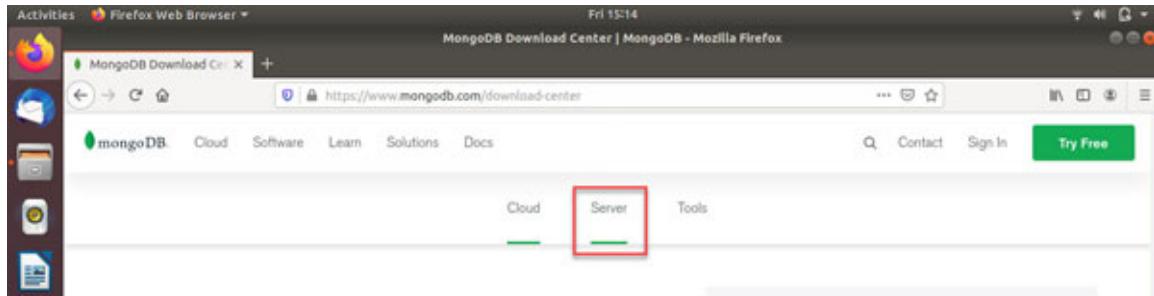
*Figure 3.4: MongoDB Inc. official website home page*

2. Click on the `try free` green color button on the top right corner of the website, as shown in the following screenshot:



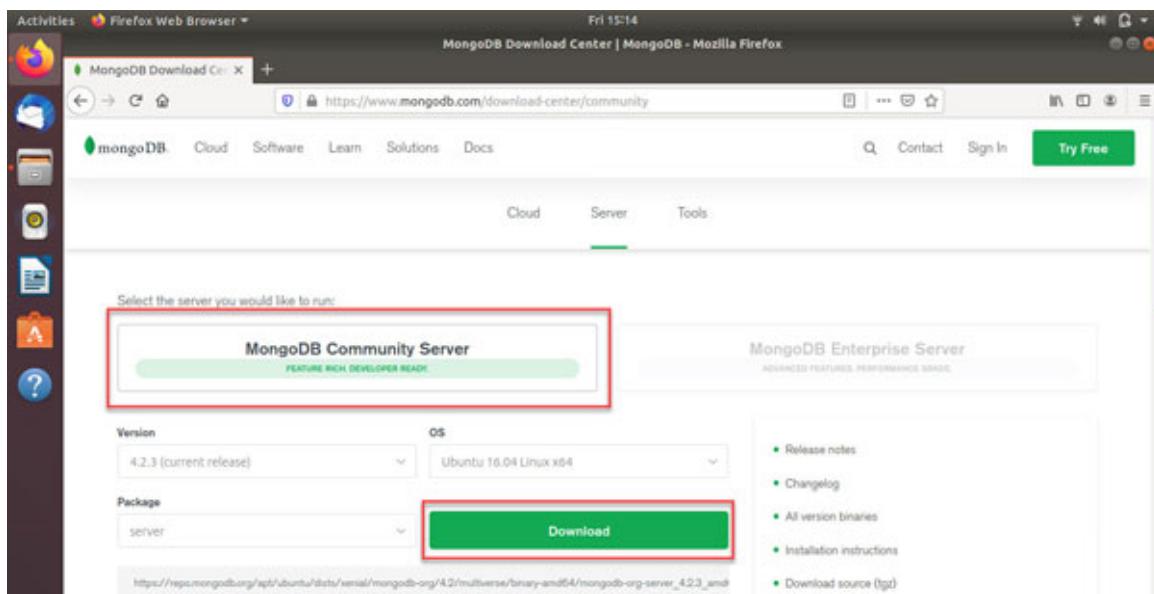
*Figure 3.5: MongoDB Inc. official website home page – Try Free Button*

3. This will open the **MongoDB Download Center** page where you will see 3 main tabs on the top section of the page. Click on the `server` tab link, as shown in the following screenshot:



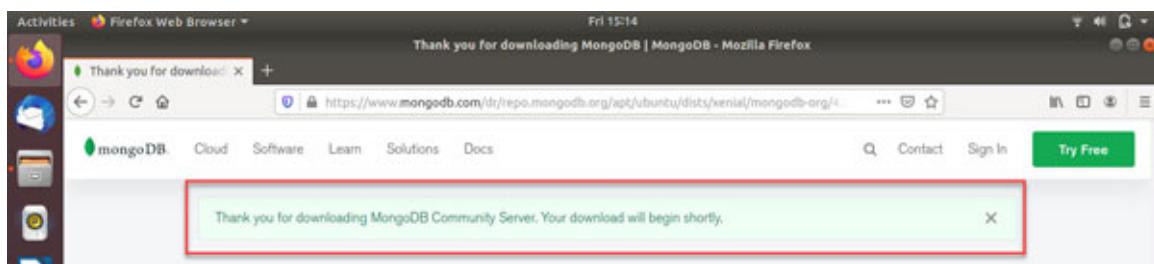
**Figure 3.6:** MongoDB Inc. official download center page

4. This will open a **MongoDB Server Download** screen. This screen will auto detect your OS type. You can change the OS type to correct version if not detected automatically, as shown in the following screenshot:



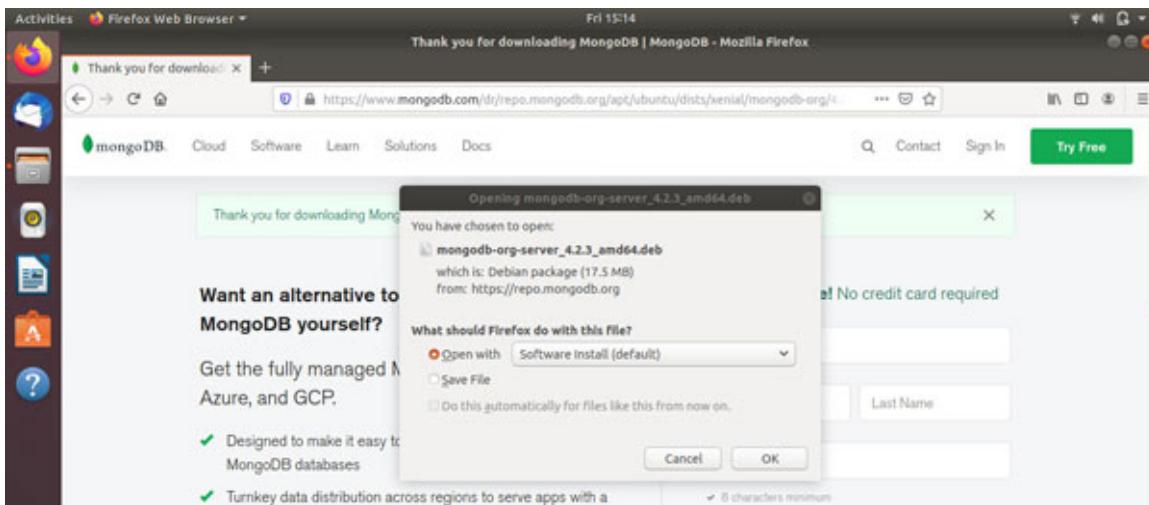
**Figure 3.7:** MongoDB Inc. official download center – MongoDB server download screen

5. Now, click on the **Download** button and MongoDB download will start automatically, as shown in the following screenshot:



**Figure 3.8: MongoDB Inc. Official Download Center – MongoDB Server Download – Thank You Page**

- Once the download starts, you will see a prompt window related to download which will ask you to either open this file with software install or to download this file. It is recommended to use the default settings, as shown in the following screenshot:



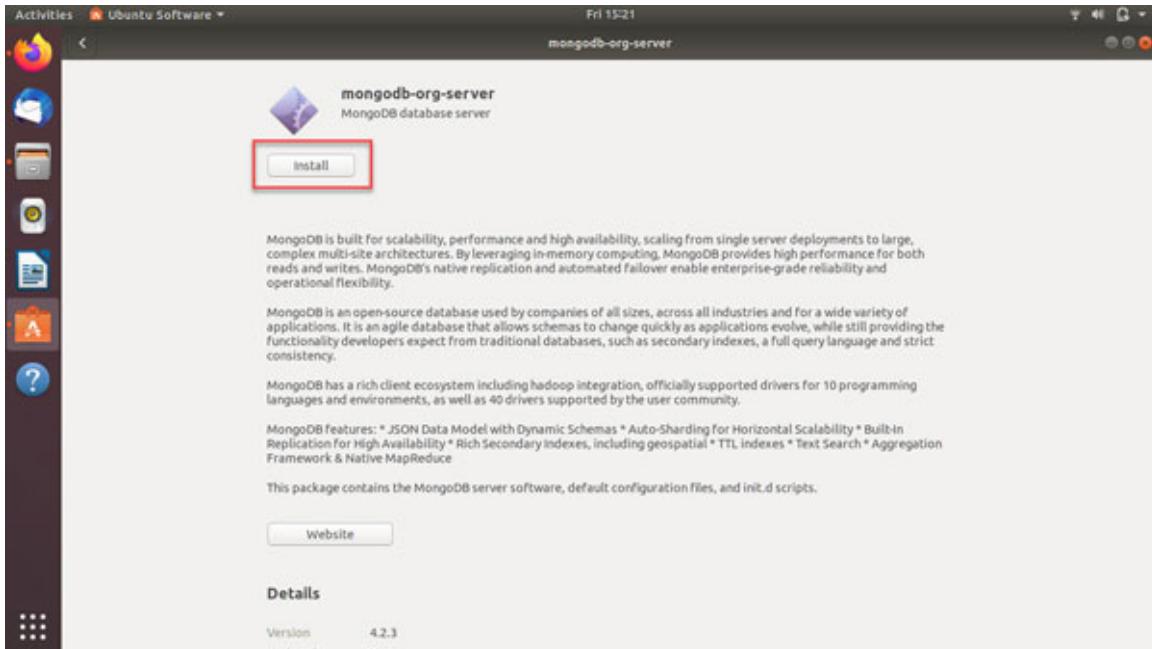
**Figure 3.9: MongoDB Download Process – MongoDB Server Download Started**

As we have now downloaded the MongoDB Community Edition from MongoDB Inc. download page. We can now continue with the installation mentioned in the step 2 of this installation process.

## **Step 2 – Install MongoDB Community Edition on your Linux machine**

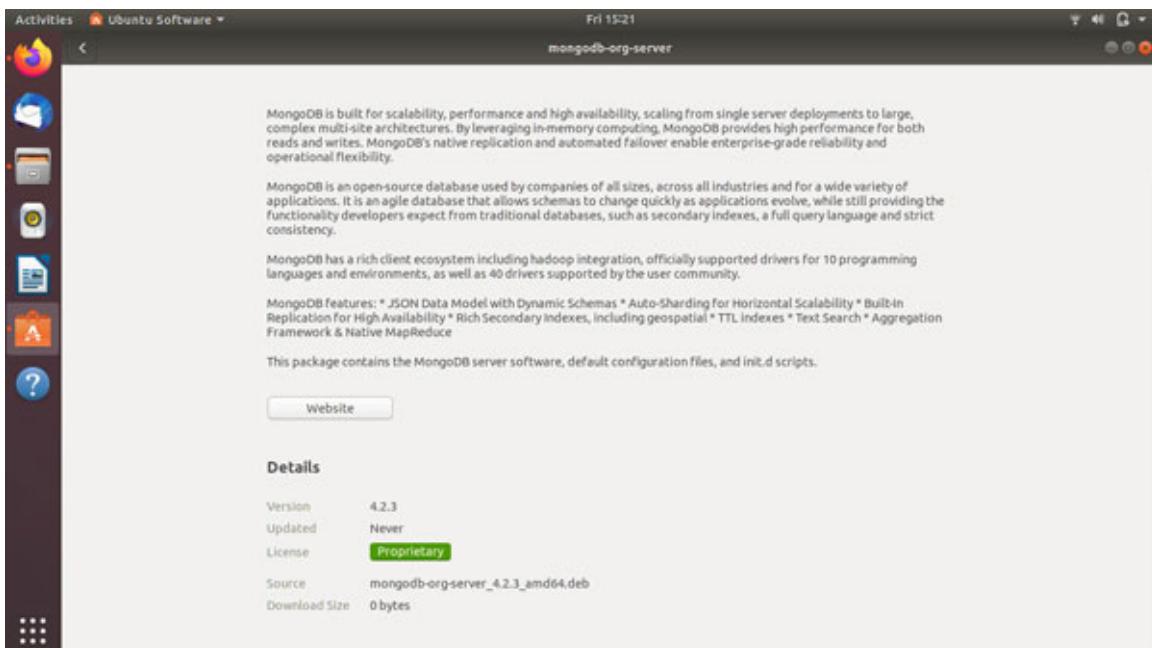
In this step, we will cover how to install MongoDB with the help of Linux installer using the browser-based method as we have already downloaded it from MongoDB Inc. website.

- After the download is 100% complete, a new screen will open where you will find the install the Mongo DB server, as shown in the following screenshot:



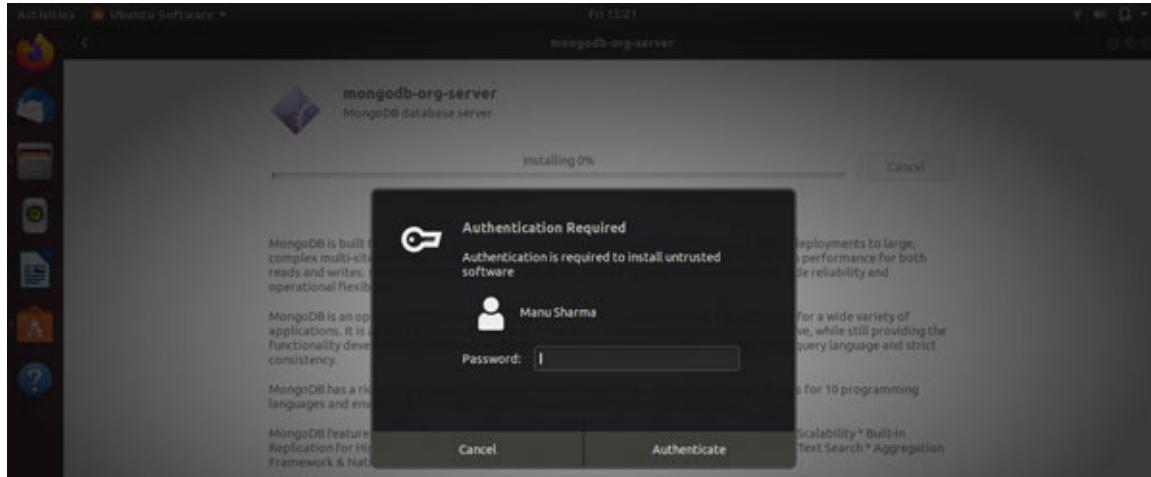
**Figure 3.10:** MongoDB Installation Process on Linux (Ubuntu)

2. You can also see the details of the MongoDB package, its version, and license in the installation screen, as shown in the following screenshot:



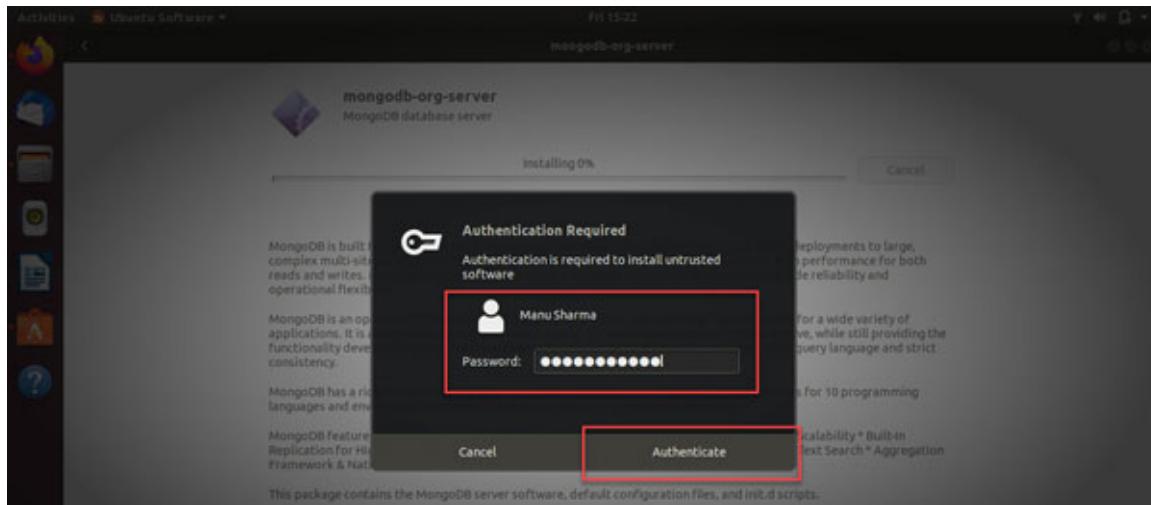
**Figure 3.11:** MongoDB Installation Process – MongoDB Package Details

3. Once you click on **Install**, you will get a prompt window, which will ask you to enter your password since it requires root access to install it on a Linux machine (in our case, Ubuntu):



*Figure 3.12: MongoDB Installation– Ubuntu Authentication Prompt*

4. So you need to type your password, and click the **Authenticate** button, as shown in the following screenshot:



*Figure 3.13: MongoDB Installation– Ubuntu Authentication Prompt*

5. Once you type your password and the MongoDB installation on Linux (Ubuntu) will **start** and the screen will show you the progress bar, as shown in the following screenshot:



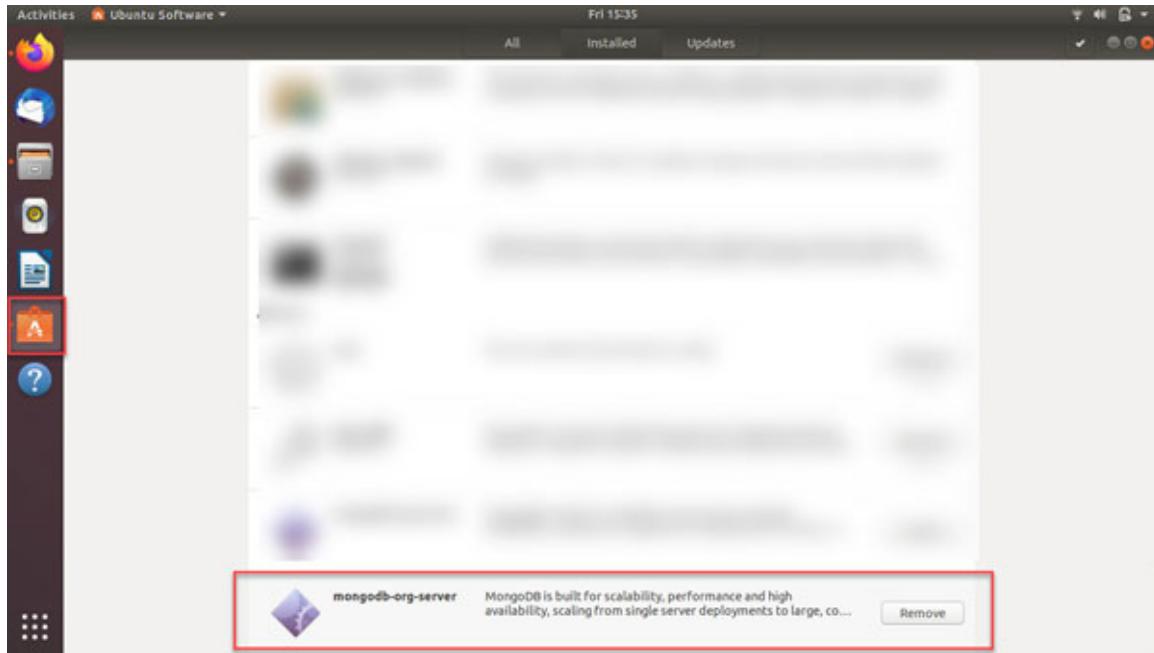
**Figure 3.14:** MongoDB Installation on Linux—Installation Progress

6. Once the installation is complete, you will see a screen, as shown in the following screenshot, which has a **Remove** button if you want to remove MongoDB from your Linux machine (Ubuntu, in our case).



**Figure 3.15:** MongoDB Installation on Linux – Installation Complete Screen

7. You can also In order to verify if MongoDB is installed in your Linux (Ubuntu) machine, Click on the **Ubuntu Software** and **Browsing**, and then checking the list of installed software, as shown in the following screenshot:



**Figure 3.16:** MongoDB Installation on Linux – Ubuntu Software Verification for MongoDB Installation

Now, we have covered how to install MongoDB using the browser-based method. In the next section, we will cover how to install MongoDB using the shell method.

## Installation Steps

### Method two (Shell method)

Sometimes, you might face some issues while installing MongoDB using the browser-based method. Sometimes, some Linux machines need some extra software, like MongoDB clients, to be installed properly before you can run MongoDB on your Linux machine. One example is explained in the following screenshot:

A screenshot of a terminal window. The title bar says "Activities" and "Terminal". The top right shows the date "Fri 16:18". Below the title bar is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The main area of the terminal shows a command being run: "manu@Manu-Linux:~\$ mongo". The terminal then outputs an error message: "Command 'mongo' not found, but can be installed with: sudo apt install mongodb-clients". The entire error message is highlighted with a red box.

**Figure 3.17:** MongoDB Installation on Linux – Post Installation Issues

Here in this example, when we try to run the `mongo` command from shell, it gives the issue that it is not found and we need some additional software or libraries, like `mongo-clients` in our example, to be installed before we run MongoDB on our Linux operated system.

So in order to resolve this, we can run the following command and get those software or libraries:

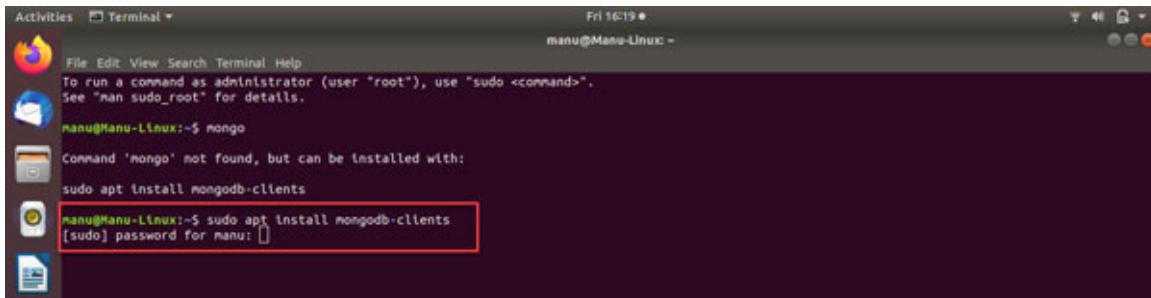
## **Steps for installing MongoDB clients (`mongo-clients`) on Linux based systems (Ubuntu)**

Mongo clients libraries (`mongo-clients`) are used to connect to the MongoDB server. This step will show you how to install these on your Linux operated machine.

1. You need to type the following command from your Linux Shell:

```
sudo apt install mongodb-clients
```

This will prompt you to enter your password to proceed ahead, as shown in the following screenshot:



**Figure 3.18: MongoDB Clients Installation on Linux – Authentication Prompt by Shell**

2. After you type your password and hit the *Enter* key, it will start downloading and installing the MongoDB client for Linux (Ubuntu). At times, there is an additional prompt where it will ask you to press the *Y* key and press *Enter* for your confirmation, as shown in the following screenshot:

```

Activities Terminal Fri 16:19 manu@Manu-Linux
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

manu@Manu-Linux:~$ mongo
Command 'mongo' not found, but can be installed with:

sudo apt install mongodb-clients

[sudo] password for manu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  eftbootmgr libfwupl libwayland-egl1-mesa
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libboost-program-options1.65.1 libgoogle-perfetto4 libpcrecpp0v5 libsnappy1v5 libtcmalloc-minimal4 libyaml-cpp0.5v5 mongo-tools
The following packages will be REMOVED:
  mongodb-org-server
The following NEW packages will be installed:
  libboost-program-options1.65.1 libgoogle-perfetto4 libpcrecpp0v5 libsnappy1v5 libtcmalloc-minimal4 libyaml-cpp0.5v5 mongo-tools
  mongodb-clients
0 upgraded, 8 newly installed, 1 to remove and 8 not upgraded.
Need to get 33.1 MB of archives.
After this operation, 71.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

```

**Figure 3.19: MongoDB Clients Installation on Linux – Confirmation Prompt by Shell**

3. You can see the progress for installation of the MongoDB clients, as shown in the following screenshot:

```

Activities Terminal Fri 16:20 manu@Manu-Linux
File Edit View Search Terminal Help
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 libboost-program-options1.65.1 amd64 1.65.1+dfsg-0ubuntu5 [137 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 libtcmalloc-minimal4 amd64 2.5.2-2ubuntu3 [91.6 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 libgoogle-perfetto4 amd64 2.5-2.2ubuntu3 [190 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 libpcrecpp0v5 amd64 2:8.39-9 [15.3 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 libyaml-cpp0.5v5 amd64 0.5.2-4ubuntu1 [150 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 mongo-tools amd64 3.6.3-0ubuntu1 [12.3 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 libsnappy1v5 amd64 1.1.7-1 [16.0 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 mongodb-clients amd64 1:3.6.3-0ubuntu1.1 [20.2 kB]
Fetched 33.1 MB in 14s (2,345 kB/s)
(Reading database ... 108782 files and directories currently installed.)
Removing mongodb-org-server (4.2.3) ...
Selecting previously unselected package libboost-program-options1.65.1:amd64.
(Reading database ... 108771 files and directories currently installed.)
Preparing to unpack .../0-libboost-program-options1.65.1-1.65.1+dfsg-0ubuntu5_amd64.deb ...
Unpacking libboost-program-options1.65.1:amd64 (1.65.1+dfsg-0ubuntu5) ...
Selecting previously unselected package libtcmalloc-minimal4.
Preparing to unpack .../1-libtcmalloc-minimal4_2.5-2.2ubuntu3_amd64.deb ...
Unpacking libtcmalloc-minimal4 (2.5-2.2ubuntu3) ...
Selecting previously unselected package libgoogle-perfetto4.
Preparing to unpack .../2-libgoogle-perfetto4_2.5-2.2ubuntu3_amd64.deb ...
Unpacking libgoogle-perfetto4 (2.5-2.2ubuntu3) ...
Selecting previously unselected package libpcrecpp0v5:amd64.
Preparing to unpack .../3-libpcrecpp0v5_283a8.39-9_amd64.deb ...
Unpacking libpcrecpp0v5:amd64 (2:8.39-9) ...
Selecting previously unselected package libyaml-cpp0.5v5:amd64.
Preparing to unpack .../4-libyaml-cpp0.5v5_0.5.2-4ubuntu1_amd64.deb ...
Unpacking libyaml-cpp0.5v5:amd64 (0.5.2-4ubuntu1) ...
Selecting previously unselected package mongo-tools.
Preparing to unpack .../5-mongo-tools_3.6.3-0ubuntu1_amd64.deb ...
Unpacking mongo-tools (3.6.3-0ubuntu1) ...
Selecting previously unselected package libsnappy1v5:amd64.
Preparing to unpack .../6-libsnappy1v5_1.1.7-1_amd64.deb ...
Unpacking libsnappy1v5:amd64 (1.1.7-1) ...
Selecting previously unselected package mongodb-clients.
Preparing to unpack .../7-mongodb-clients_13a3.6.3-0ubuntu1.1_amd64.deb ...
Unpacking mongodb-clients (1:3.6.3-0ubuntu1.1) ...

```

**Figure 3.20: MongoDB Clients Installation on Linux – Progress**

4. Once this installation of the MongoDB clients is complete, it will display the **command prompt** again, as shown in the following screenshot:

```
Activities Terminal Fri 16:20 • manu@Manu-Linux:~  
File Edit View Search Terminal Help  
Selecting previously unselected package libboost-program-options1.65.1:amd64.  
(Reading database ... 108771 files and directories currently installed.)  
Preparing to unpack .../0-libboost-program-options1.65.1.1.65.1+dfsg-0ubuntu5_amd64.deb ...  
Unpacking libboost-program-options1.65.1:amd64 (1.65.1+dfsg-0ubuntu5) ...  
Selecting previously unselected package libtcmalloc-minimal4.  
Preparing to unpack .../1-libtcmalloc-minimal4_2.5.2-2.2ubuntu3_amd64.deb ...  
Unpacking libtcmalloc-minimal4 (2.5.2-2.2ubuntu3) ...  
Selecting previously unselected package libgoogle-perftools4.  
Preparing to unpack .../2-libgoogle-perftools4_2.5-2.2ubuntu3_amd64.deb ...  
Unpacking libgoogle-perftools4 (2.5-2.2ubuntu3) ...  
Selecting previously unselected package libpcrecpp0v5:amd64.  
Preparing to unpack .../3-libpcrecpp0v5_2.3.0.39-9_amd64.deb ...  
Unpacking libpcrecpp0v5:amd64 (2:8.39-9) ...  
Selecting previously unselected package libyaml-cpp0.5v5:amd64.  
Preparing to unpack .../4-libyaml-cpp0.5v5_0.5.2-4ubuntu1_amd64.deb ...  
Unpacking libyaml-cpp0.5v5:amd64 (0.5.2-4ubuntu1) ...  
Selecting previously unselected package mongo-tools.  
Preparing to unpack .../5-mongo-tools_3.6.3-0ubuntu1_amd64.deb ...  
Unpacking mongo-tools (3.6.3-0ubuntu1) ...  
Selecting previously unselected package libsnappy1v5:amd64.  
Preparing to unpack .../6-libsnappy1v5_1.1.7-1_amd64.deb ...  
Unpacking libsnappy1v5:amd64 (1.1.7-1) ...  
Selecting previously unselected package mongodb-clients.  
Preparing to unpack .../7-mongodb-clients_1.3.6.3-0ubuntu1.1_amd64.deb ...  
Unpacking mongodb-clients (1:3.6.3-0ubuntu1.1) ...  
Setting up libtcmalloc-minimal4 (2.5-2.2ubuntu3) ...  
Setting up libgoogle-perftools4 (2.5-2.2ubuntu3) ...  
Setting up libpcrecpp0v5:amd64 (2:8.39-9) ...  
Setting up libyaml-cpp0.5v5:amd64 (0.5.2-4ubuntu1) ...  
Setting up libboost-program-options1.65.1:amd64 (1.65.1+dfsg-0ubuntu5) ...  
Setting up mongo-tools (3.6.3-0ubuntu1) ...  
Setting up mongodb-clients (1:3.6.3-0ubuntu1.1) ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
Processing triggers for libc-bin (2.27-3ubuntu1) ...  
manu@Manu-Linux:~$
```

**Figure 3.21: MongoDB Clients Installation on Linux – Command Prompt**  
available to type any command after MongoDB Clients installation is complete

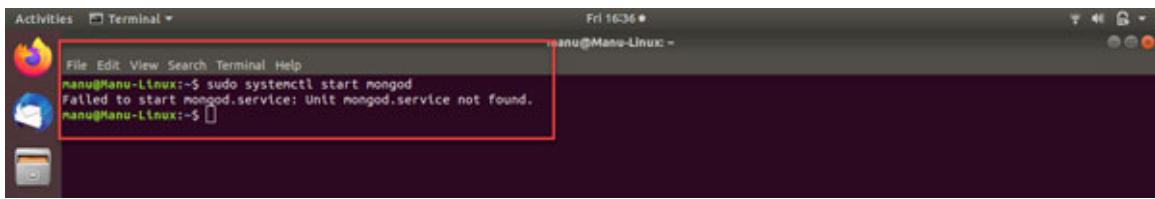
Now, in the preceding step, we covered how to install `mongo-clients` library on the machines running on Linux. In the next step, we will cover the installation of MongoDB on Linux machine (Ubuntu, in our case) using the Shell commands.

## Steps for installing MongoDB on Linux based systems (Ubuntu) using the Shell commands

In some scenarios, you might face an issue in starting MongoDB even when you have already installed it using the browser-based method. If you are unable to start the MongoDB services, as shown in the following screenshot, try to start it using the Shell command. For example, try the following command:

```
sudo systemctl start mongodb
```

And if you're still unable to start the MongoDB services, follow the next steps to install MongoDB using the Shell method, which we will cover in the following steps:



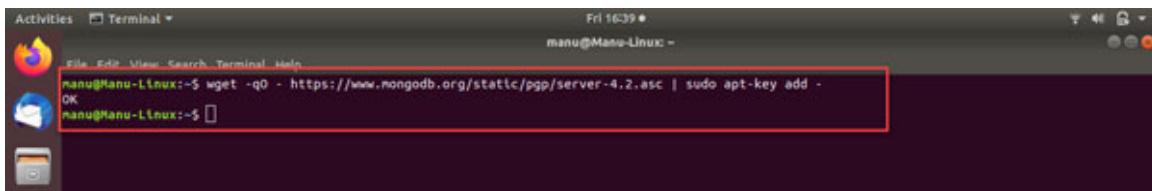
**Figure 3.22:** MongoDB Service – Unable to Start (MongoDB Service was not found)

Now, we will cover the step-by-step method to install MongoDB from the command line method using the Linux (Ubuntu) Shell.

1. Type the following command from your Linux Shell:

```
 wget -qO - https://www.mongodb.org/static/pgp/server-4.3.asc  
 | sudo apt-key add -
```

This will import the public key (also known as the Public GPG Key) used by the package management system. Note that this command will return you an `OK` message, as shown in the following screenshot:



**Figure 3.23:** MongoDB Installation (Command Line) – Importing Public GPG Key

If you receive any message other than `OK`, it is quite possible that the `gnupg` is not yet installed on your system. In this case, you need to install the `gnupg` on your Linux (Ubuntu) system, as shown in the following screenshot: `sudo apt-get install gnupg`

The above command requires `gnupg` to run and function to return the `OK` Message. If you receive any message other than `OK`, then it is quite possible that `gnupg` is not yet installed on your system. In this case you need to install the `gnupg` on your Linux (Ubuntu) System:

```

Activities Terminal Fri 16:39 manu@Manu-Linux -
File Edit View Search Terminal Help
manu@Manu-Linux:~$ sudo apt-get install gnupg
Reading package lists... Done
Building dependency tree
Reading state information... Done
gnupg is already the newest version (2.2.4-1ubuntu1.2).
The following packages were automatically installed and are no longer required:
  efibootmgr libfwupl libwaylandegl-mesa
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
manu@Manu-Linux:~$ 

```

**Figure 3.24:** MongoDB Installation (Command Line) – Install "gnupg"

After installing the `gnupg`, you should now follow Step 1.

2. After completing step 1, follow the next step where you create a list file for MongoDB using the following commands, as shown in the following screenshot:

```

echo "deb [arch=amd64,arm64]
https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-
4.3.list

```

```

Activities Terminal Fri 16:44 manu@Manu-Linux -
File Edit View Search Terminal Help
manu@Manu-Linux:~$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse
manu@Manu-Linux:~$ 

```

**Figure 3.25:** MongoDB Installation (Command Line) – creating a list file for MongoDB

3. Now, reload the local package database using the following command, as shown in the following screenshot:

```
sudo apt-get update
```

```

Activities Terminal Fri 16:45 manu@Manu-Linux -
File Edit View Search Terminal Help
manu@Manu-Linux:~$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse
manu@Manu-Linux:~$ sudo apt-get update
Get:1 http://ln.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://ln.archive.ubuntu.com/ubuntu bionic-updates InRelease
Get:3 http://ln.archive.ubuntu.com/ubuntu bionic-backports InRelease
Ign:4 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 InRelease
Get:5 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 Release [3,953 B]
Get:6 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 Release.gpg [801 B]
Get:7 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:8 https://debs.opera.com/opera-stable stable InRelease [2,591 B]
Get:9 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2/multiverse arm64 Packages [4,063 B]
Get:10 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2/multiverse arm64 DEP-11 Metadata [4,877 B]
Get:11 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [38.5 kB]
49% [Waiting for headers] [Waiting for headers]

```

**Figure 3.26:** MongoDB Installation (Command Line) – Reloading the local Package Database

4. Now, follow the final step which will install MongoDB packages on your Linux (Ubuntu) system using the following command, as shown in the following screenshot:

```
sudo apt-get install -y mongodb-org
```

```
manu@Manu-Linux:~$ sudo apt-get install -y mongodb-org
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  elfbootmgr libboost-program-options1.65.1 libfwupl libgoogle-perfetto4 libpcrecppv5 libsnappyv5 libtcmalloc-minimal4
  libwayland-egl1-mesa libyaml-cpp0.5v5 mongo-tools
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libcurl4 mongodb-org-mongos mongodb-org-server mongodb-org-shell mongodb-org-tools
The following packages will be REMOVED:
  libcurl3 mongodb-clients
The following NEW packages will be installed:
  libcurl4 mongodb-org mongodb-org-mongos mongodb-org-server mongodb-org-shell mongodb-org-tools
0 upgraded, 6 newly installed, 2 to remove and 8 not upgraded.
Need to get 97.6 MB of archives.
After this operation, 222 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcurl4 amd64 7.58.0-2ubuntu3.8 [214 kB]
Get:2 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2/multiverse amd64 mongodb-org-shell amd64 4.2.3 [12.0 MB]
Get:3 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2/multiverse amd64 mongodb-org-server amd64 4.2.3 [18.4 MB]
Get:4 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2/multiverse amd64 mongodb-org-mongos amd64 4.2.3 [10.1 MB]
Get:5 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2/multiverse amd64 mongodb-org-tools amd64 4.2.3 [57.0 MB]
70% [5 mongodb-org-tools 29.1 MB/57.0 MB 51N] 2,225 kB/s 12s
```

**Figure 3.27: MongoDB Installation (Command Line) – Installing MongoDB Packages**

5. Once MongoDB is installed on your Linux (Ubuntu) system, you will be taken back to the command prompt, as shown in the following screenshot:

```
manu@Manu-Linux:~$ Fri 16:47 ~
File Edit View Search Terminal Help
Get:6 https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2/multiverse amd64 mongodb-org and64 4.2.3 [3,532 B]
Fetched 97.6 MB in 43s (2,280 kB/s)
dpkg: libcurl3:amd64: dependency problems, but removing anyway as you requested:
  opera-stable depends on libcurl3 (>= 7.16.2) | libcurl4 (>= 7.58.0); however:
    Package libcurl3:amd64 is to be removed.
    Package libcurl4 is not installed.

(Reading database ... 168841 files and directories currently installed.)
Removing libcurl3:amd64 (7.58.0-2ubuntu2) ...
Selecting previously unselected package libcurl4:amd64.
(Reading database ... 168835 files and directories currently installed.)
Preparing to unpack .../libcurl4_7.58.0-2ubuntu3.8_amd64.deb ...
Unpacking libcurl4:amd64 (7.58.0-2ubuntu3.8) ...
(Reading database ... 168840 files and directories currently installed.)
Removing mongodb-clients (1:13.0.3-0ubuntu1.1) ...
Selecting previously unselected package mongodb-org-shell.
(Reading database ... 168833 files and directories currently installed.)
Preparing to unpack .../mongodb-org-shell_4.2.3_amd64.deb ...
Unpacking mongodb-org-shell (4.2.3) ...
Selecting previously unselected package mongodb-org-server.
Preparing to unpack .../mongodb-org-server_4.2.3_amd64.deb ...
Unpacking mongodb-org-server (4.2.3) ...
Selecting previously unselected package mongo-tools.
Preparing to unpack .../mongo-tools_4.2.3_amd64.deb ...
Unpacking mongo-tools (4.2.3) ...
dpkg: error processing archive '/var/cache/apt/archives/mongo-tools_4.2.3_amd64.deb (--unpack):
  trying to overwrite '/usr/bin/bondone', which is also in package mongo-tools 3.6.3-0ubuntu1
dpkg-deb: error: paste subprocess was killed by signal (Broken pipe)
Selecting previously unselected package mongodb-org.
Preparing to unpack .../mongodb-org_4.2.3_amd64.deb ...
Unpacking mongodb-org (4.2.3) ...
Errors were encountered while processing:
 /var/cache/apt/archives/mongo-tools_4.2.3_amd64.deb
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

**Figure 3.28: MongoDB Installation (Command Line) – MongoDB Packages Installation Completed**

In step 2, we covered how to install MongoDB using the Shell method. In step 3, we will cover how to start MongoDB on your Linux machine (Ubuntu, in our case).

## **Step 3 –Starting MongoDB on Linux (Ubuntu)**

To make it sure that MongoDB has been correctly installed in your Linux (Ubuntu) machine, follow these steps:

1. Start the MongoDB service using the following command:

```
sudo systemctl start mongod
```

2. This will start the MongoDB service on your Linux machine (Ubuntu, in our case), as shown in the following screenshot:

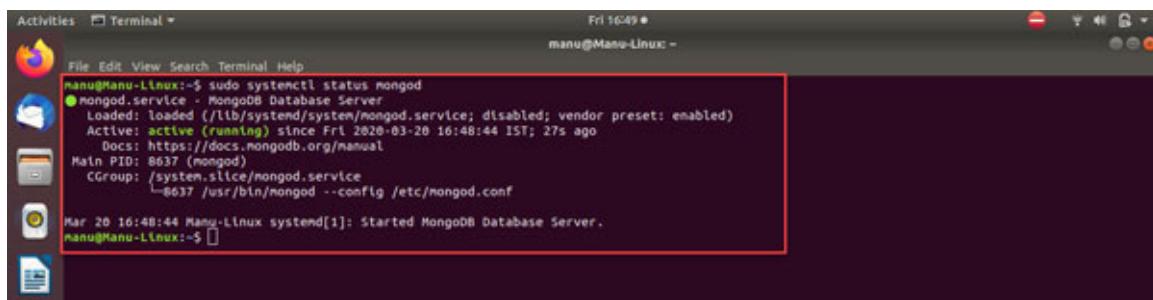


*Figure 3.29: Starting MongoDB Service on Linux (Ubuntu)*

3. You should check if the service has been started by checking MongoDB service status using the following command:

```
sudo systemctl status mongod
```

If the service has begun correctly, it will show you the status of the MongoDB service, as shown in the following screenshot:



*Figure 3.30: Checking the MongoDB Service Status*

In step 3, we covered how to start MongoDB on your Linux machine (Ubuntu, in our case).; In step 4, we will cover how to connect to MongoDB Shell from the Linux machine.

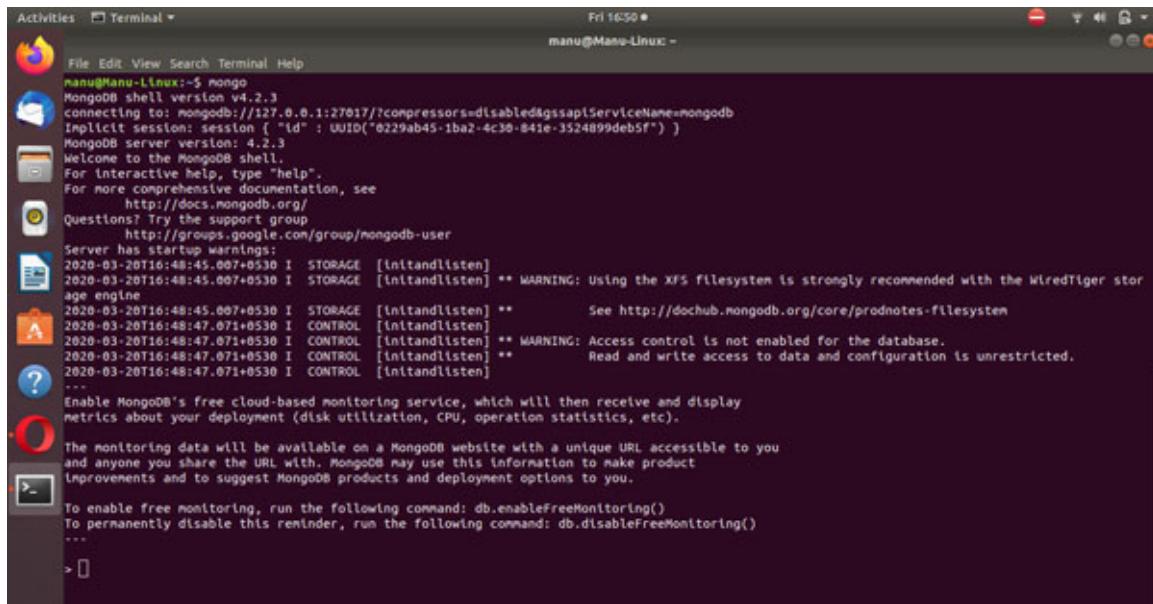
## Step 4 – Connecting to MongoDB on Linux (Ubuntu)

Let us now try to connect MongoDB command line from your Linux (Ubuntu) machine using the Shell command. To connect to MongoDB from your Linux machine, follow these steps:

1. Open a separate Shell command window and run the following command:

```
mongo
```

This will open the MongoDB Shell interface to run various MongoDB commands, as shown in the following screenshot:



The screenshot shows a terminal window titled "Terminal" with the command "mongo" entered. The output shows the MongoDB shell version 4.2.3 connecting to the local host. It provides welcome messages, documentation links, and server startup warnings. It also mentions the availability of free monitoring and provides commands to enable or disable it.

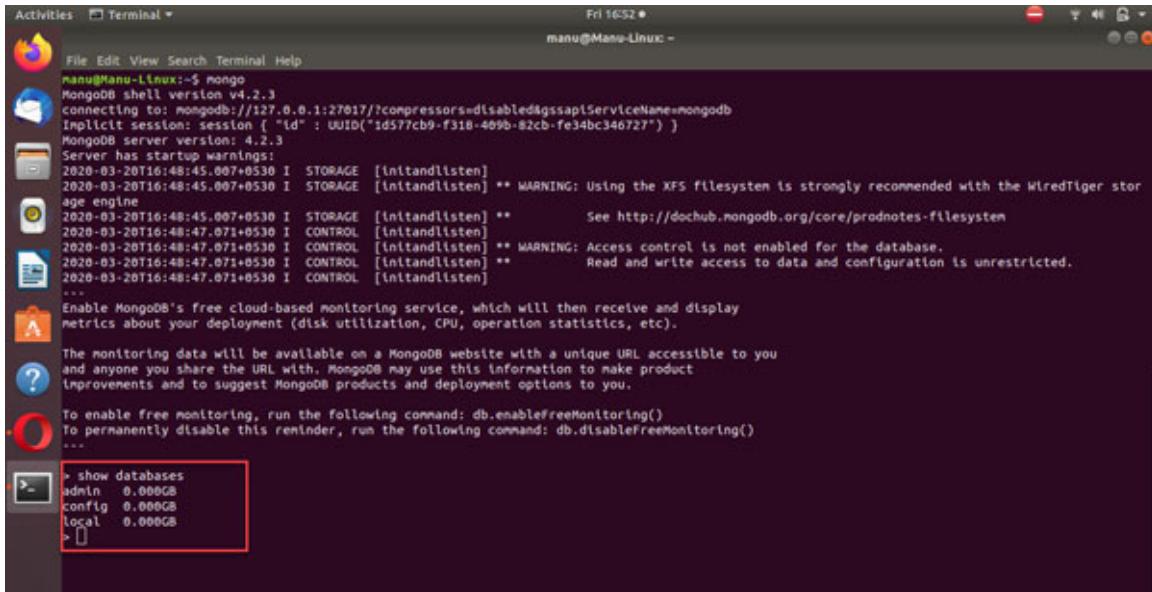
```
File Edit View Search Terminal Help
manu@Manu-Linux:~$ mongo
MongoDB shell version v4.2.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "_id" : UUID("0229ab45-1ba2-4c30-841e-3524899deb5f") }
MongoDB server version: 4.2.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
    http://docs.mongodb.org/
Questions? Try the support group
    http://groups.google.com/group/mongodb-user
Server has startup warnings:
2020-03-20T16:48:45.007+0530 I STORAGE  [initandlisten]
2020-03-20T16:48:45.007+0530 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2020-03-20T16:48:45.007+0530 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-03-20T16:48:47.071+0530 I CONTROL  [initandlisten]
2020-03-20T16:48:47.071+0530 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-03-20T16:48:47.071+0530 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2020-03-20T16:48:47.071+0530 I CONTROL  [initandlisten]
...
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()
...
> 
```

**Figure 3.31: From Command Line –Running "mongo" command and accessing MongoDB Shell.**

2. To test further, issue the following Mongo command and press *Enter*:

```
show databases
```

This will give you the list of MongoDB databases that are present in your system, as shown in the following screenshot:



```
File Edit View Search Terminal Help
manu@Manu-Linux:~$ mongo
MongoDB shell version v4.2.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("1d577cb9-f318-409b-82cb-fe34bc346727") }
MongoDB server version: 4.2.3
Server has startup warnings:
2020-03-20T16:48:45.007+0530 I STORAGE [initandlisten]
2020-03-20T16:48:45.007+0530 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2020-03-20T16:48:47.071+0530 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-03-20T16:48:47.071+0530 I CONTROL [initandlisten]
2020-03-20T16:48:47.071+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-03-20T16:48:47.071+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-03-20T16:48:47.071+0530 I CONTROL [initandlisten]
...
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.

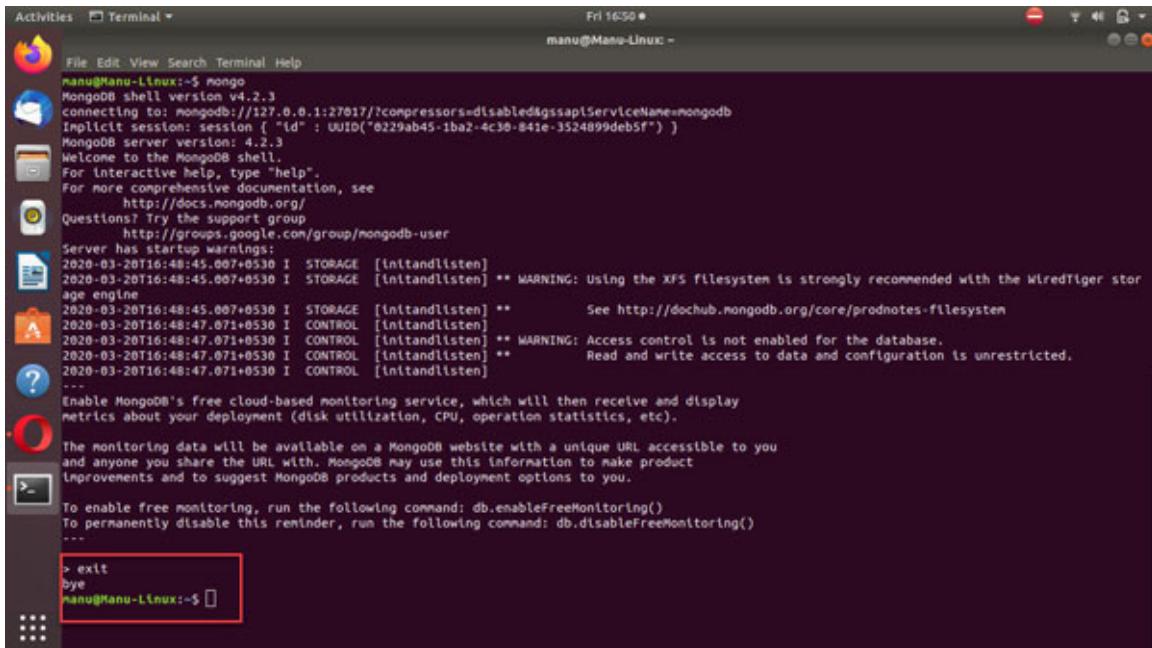
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
...
> show databases
admin 0.00008
config 0.00008
local 0.00008
> 
```

**Figure 3.32: Mongo Shell – "show databases" MongoDB Command**

3. To exit from the MongoDB Shell, type the following command and press the *Enter* key:

```
exit
```

This will take you out of the MongoDB Shell, as shown in the following screenshot:



```
File Edit View Search Terminal Help
manu@Manu-Linux:~$ mongo
MongoDB shell version v4.2.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("0229ab45-1ba2-4c38-841e-3524899deb5f") }
MongoDB server version: 4.2.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
    http://docs.mongodb.org/
Questions? Try the support group
    http://groups.google.com/group/mongodb-user
Server has startup warnings:
2020-03-20T16:48:45.007+0530 I STORAGE [initandlisten]
2020-03-20T16:48:45.007+0530 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2020-03-20T16:48:47.071+0530 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-03-20T16:48:47.071+0530 I CONTROL [initandlisten]
2020-03-20T16:48:47.071+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-03-20T16:48:47.071+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-03-20T16:48:47.071+0530 I CONTROL [initandlisten]
...
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()
...
> exit
bye
manu@Manu-Linux:~$ 
```

**Figure 3.33: Mongo Shell – "exit" MongoDB Command – To Exit from MongoDB**

So, we learned how to install and setup MongoDB on Linux machines (in our case, Ubuntu), and verify the MongoDB installation. We also learned to start MongoDB using the Shell commands. At last, we learned how to connect to MongoDB Shell.

## **Conclusion**

In this chapter, we learned to install MongoDB on our Linux based machine, using both the browser-based method, as well as the Linux Shell method.

MongoDB can be installed by downloading it from the MongoDB Inc. official website and choosing the platform as Linux. We have to run the installer file (Linux installer file) to run the setup. Once the setup is complete, we can then verify the installation by the post-installation verification steps. The reader also learned how to start the MongoDB service on Linux, and how to start and use the MongoDB Shell.

In the next chapter, we will cover the installation of MongoDB on macOS based systems. This would be helpful for the readers who use macOS based machines.

So far, we learned how to install and setup MongoDB on the Linux OS (Ubuntu) based machines and verify the MongoDB installation. We also learned to start MongoDB using the Shell commands, and at last we learned how to connect to the MongoDB Shell.

## **Questions**

1. Where should you download the MongoDB setup file from?
2. Explain how to start the MongoDB service on a Linux machine.
3. How can you install the MongoDB client libraries on your Linux machine?
4. How can you verify if MongoDB has been successfully installed in your Linux based system?
5. Explain the process of installing MongoDB using the Linux Shell method.

6. How can you login to the MongoDB Shell on your Linux based machine? Explain the process.

## CHAPTER 4

# MongoDB Installation and Setup on macOS

This chapter covers the installation and steps to set up MongoDB on machines powered by macOS (Mac operating system). This chapter will also cover the post-installation checks for the MongoDB installation for Mac operating systems. So, in this chapter, you will learn how to download MongoDB for macOS and how to install it correctly on your macOS operated machine. This chapter is covers step-by-step method with screenshots to make you understand the installation and setup of MongoDB very easily on macOS platforms. This chapter also covers the post-installation steps to easily verify if MongoDB has been correctly installed on your macOS system.

### Structure

In this chapter, we will discuss the following topics:

- MongoDB setup on macOS
- Checking the installation on macOS
- Connecting to MongoDB on macOS

### Objectives

After studying this unit, you should be able to learn the steps required to install MongoDB on your macOS operated machine and check if MongoDB is correctly installed on your MacOS. Later in this chapter, you will learn to connect to MongoDB and post-installation and verification checks on MacOS.

### MongoDB setup on macOS

Let us explore how we can install and setup the MongoDB Community Edition on machines running on macOS.

## Installing MongoDB Community Edition on macOS

We will show you how you can install the MongoDB Community Edition on macOS. We will use the official installation method to install the MongoDB Community Edition on the machines that run on Mac operating system.

Here, we will cover the MongoDB Community Edition (version 4.4) for 64-bit versions of Mac on x86\_64 architecture. This includes the following version of macOS.

macOS 10.13 or Later Ver0

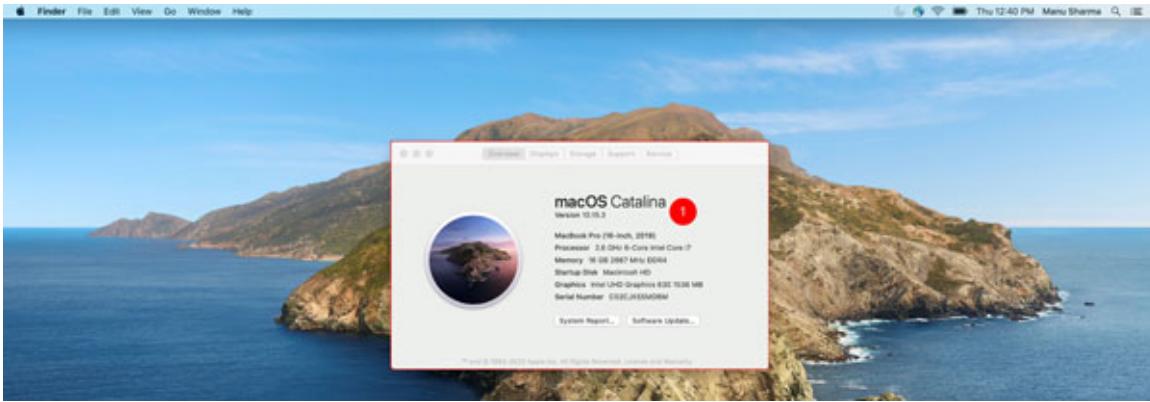
In our example, we will use macOS 10.13 or the later version to install and setup MongoDB.

1. To check the version of your macOS, click the Apple menu on your macOS and click the option `About this Mac`, as shown in the following screenshot:



*Figure 4.1: macOS Apple Menu > About this Mac*

- Once you do this, you will see the details of the version of the MacOS, as shown in the following screenshot:



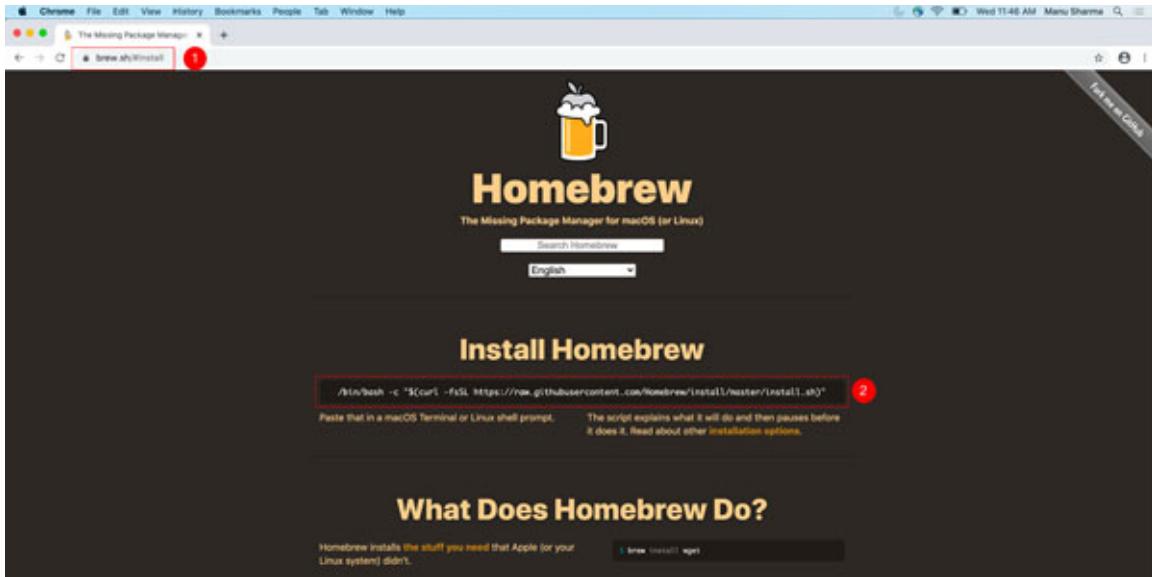
*Figure 4.2: Checking your macOS version*

## Installation steps

### **Step 1 – Install Homebrew**

In this step, we will use Brew, a package manager for macOS to install MongoDB later in this chapter. By default, OSX does not have a Homebrew package manager pre-installed. So, we need to install Brew in our macOS. If you already have Homebrew Brew package manager installed in your macOS, you can skip this step and follow the next step to install the MongoDB community server with the Brew package manager.

- Open the Homebrew official website, <https://brew.sh>, in your favorite browser, as shown in the following screenshot:



**Figure 4.3:** Homebrew Official Website Home Page

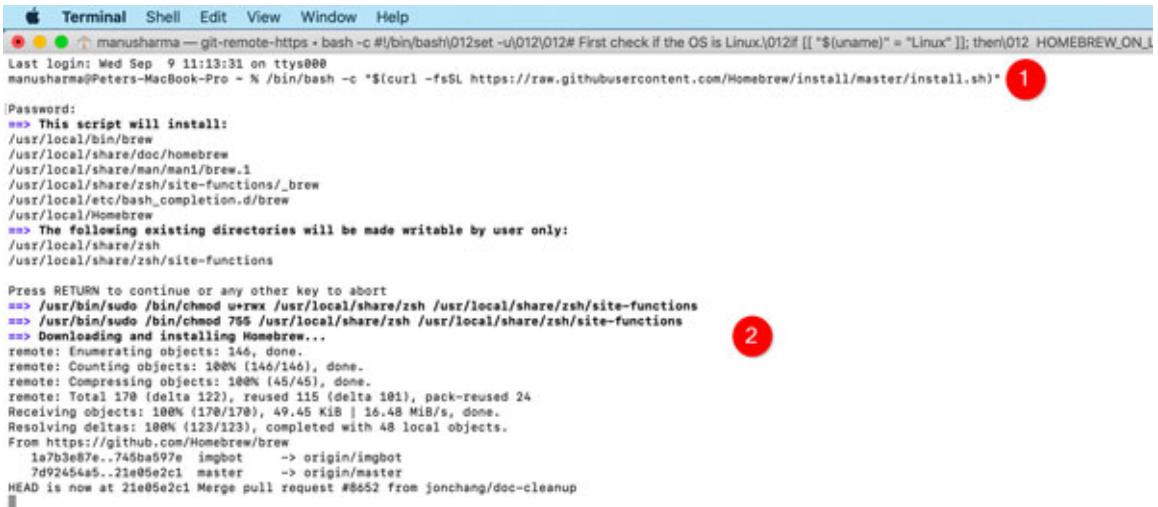
2. You will see an installation script mentioned on the top section of the home page. Copy this script and open your terminal. Paste this script and press *Enter* to run the following command, as shown in the following screenshot:

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/ins  
tall.sh)"
```



**Figure 4.4:** macOS Terminal Screen - Script to install Homebrew in your MacOS

3. This will install the Brew package manager for macOS and you can see the installation progress while it is being installed on your MacOS. as shown in the following screenshot:



```
Terminal Shell Edit View Window Help
manusharma — git-remote-https • bash -c #!/bin/bash$012set -u$012# First check if the OS is Linux:$012if [[ "$uname" = "Linux" ]]; then$012 HOMEBREW_ON_LINUX=1$012else$012 HOMEBREW_ON_LINUX=0$012fi$012$012Last login: Wed Sep 9 11:13:31 on ttys000
manusharma@Peters-MacBook-Pro ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)" 1

>Password:
=> This script will install:
/usr/local/bin/brew
/usr/local/share/doc/homebrew
/usr/local/share/man/man1/brew.1
/usr/local/share/zsh/site-functions/_brew
/usr/local/etc/bash_completion.d/brew
/usr/local/Homebrew
=> The following existing directories will be made writable by user only:
/usr/local/share/zsh
/usr/local/share/zsh/site-functions

Press RETURN to continue or any other key to abort
=> /usr/bin/sudo /bin/chmod u+rwx /usr/local/share/zsh /usr/local/share/zsh/site-functions
=> /usr/bin/sudo /bin/chmod 755 /usr/local/share/zsh /usr/local/share/zsh/site-functions
=> Downloading and installing Homebrew...
remote: Enumerating objects: 146, done.
remote: Counting objects: 100% (146/146), done.
remote: Compressing objects: 100% (45/45), done.
remote: Total 170 (delta 122), reused 115 (delta 101), pack-reused 24
Receiving objects: 100% (170/170), 49.45 KiB | 16.48 MiB/s, done.
Resolving deltas: 100% (123/123), completed with 48 local objects.
From https://github.com/Homebrew/brew
  1a7b3e87e..745ba597e  imgbot      -> origin/imgbot
    7d92e54a5..21e05e2c1  master      -> origin/master
HEAD is now at 21e05e2c1 Merge pull request #8652 from jonchang/doc-cleanup
```

**Figure 4.5:** Homebrew is getting installed in MacOS

- Once the installation gets completed, you will see the `Installation Successful` message, as shown in the following screenshot:

```

Terminal Shell Edit View Window Help
manusharma — git-remote-https + bash -c #!/bin/bash|012set -u|012|012# First check if the OS is Linux.|012if [[ "$uname" = "Linux" ]]

* [new tag]      2.2.17          -> 2.2.17
* [new tag]      2.2.2           -> 2.2.2
* [new tag]      2.2.3           -> 2.2.3
* [new tag]      2.2.4           -> 2.2.4
* [new tag]      2.2.5           -> 2.2.5
* [new tag]      2.2.6           -> 2.2.6
* [new tag]      2.2.7           -> 2.2.7
* [new tag]      2.2.8           -> 2.2.8
* [new tag]      2.2.9           -> 2.2.9
* [new tag]      2.3.0           -> 2.3.0
* [new tag]      2.4.0           -> 2.4.0
* [new tag]      2.4.1           -> 2.4.1
* [new tag]      2.4.10          -> 2.4.10
* [new tag]      2.4.11          -> 2.4.11
* [new tag]      2.4.12          -> 2.4.12
* [new tag]      2.4.13          -> 2.4.13
* [new tag]      2.4.14          -> 2.4.14
* [new tag]      2.4.15          -> 2.4.15
* [new tag]      2.4.16          -> 2.4.16
* [new tag]      2.4.2           -> 2.4.2
* [new tag]      2.4.3           -> 2.4.3
* [new tag]      2.4.4           -> 2.4.4
* [new tag]      2.4.5           -> 2.4.5
* [new tag]      2.4.6           -> 2.4.6
* [new tag]      2.4.7           -> 2.4.7
* [new tag]      2.4.8           -> 2.4.8
* [new tag]      2.4.9           -> 2.4.9
* [new tag]      2.5.0           -> 2.5.0
HEAD is now at 7d9245a5 Merge pull request #8635 from Rylan12/deprecate_disable_info
=> Homebrew is run entirely by unpaid volunteers. Please consider donating:
https://github.com/Homebrew/brew#donations
=> Tapping homebrew/core
Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-core'...
remote: Enumerating objects: 94, done.
remote: Counting objects: 100% (94/94), done.
remote: Compressing objects: 100% (68/68), done.
remote: Total 798374 (delta 53), reused 49 (delta 26), pack-reused 798288
Receiving objects: 100% (798374/798374), 314.29 MiB | 11.17 MiB/s, done.
Resolving deltas: 100% (528583/528583), done.
Tapped 2 commands and 5222 formulae (5,493 files, 344.9MB).
Already up-to-date.
=> Installation successful!

=> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
https://docs.brew.sh/Analytics
No analytics data has been sent yet (or will be during this 'install' run).

=> Homebrew is run entirely by unpaid volunteers. Please consider donating:
https://github.com/Homebrew/brew#donations

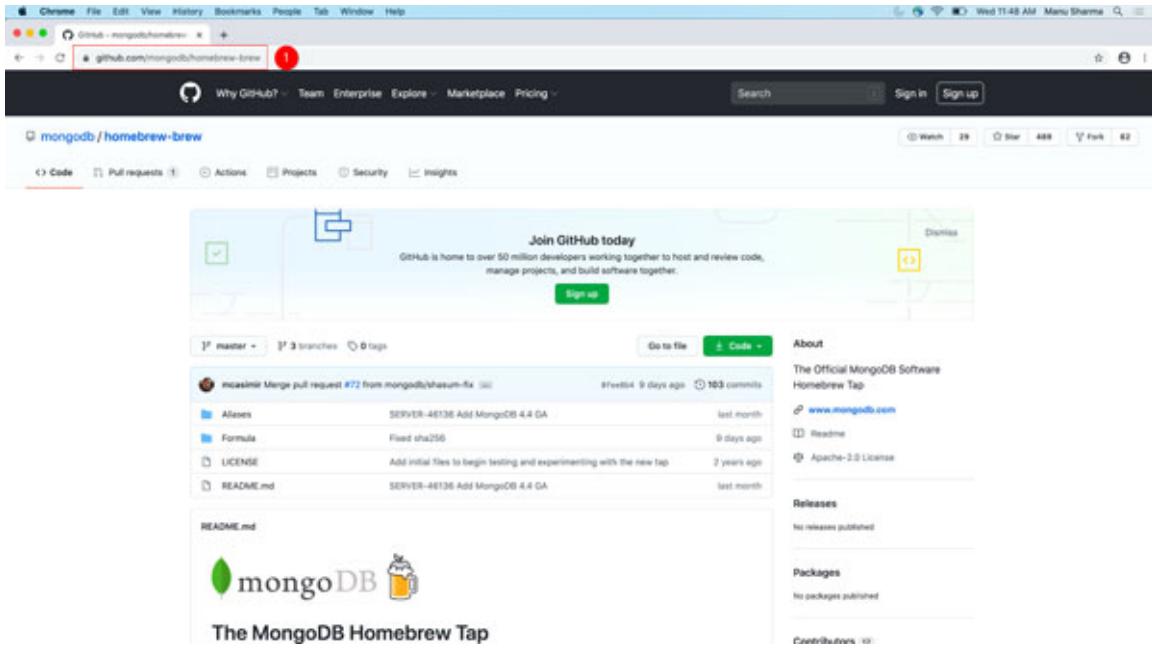
=> Next steps:
- Run 'brew help' to get started
- Further documentation:
https://docs.brew.sh
manusharma@Peters-MacBook-Pro ~ %

```

*Figure 4.6: Homebrew Installation Successful on MacOS*

## Step 2 – Install MongoDB Community Edition on your macOS machine.

1. After you have successfully installed the Homebrew Brew package manager for macOS, you can now start installing the MongoDB server Community Edition using Brew. To do this, open the URL: <https://github.com/mongodb/homebrew-brew>, which is the MongoDB Homebrew. Tap and follow the instructions, as shown in the following screenshot:



**Figure 4.7:** Homebrew MongoDB Homebrew Tap – GitHub Official Home Page

2. Here, we will follow the same instructions to install MongoDB on MacOS. Open the terminal in your macOS and run the following command, as shown in the following screenshot:

```
brew tap mongodb/brew
```

The preceding command will add the custom tap in your macOS terminal session:



**Figure 4.8:** Adding MongoDB Tap using Brew in MacOS

3. Once you have added the custom tap, run the following command to install the MongoDB Community Server, as shown in the following screenshot:

```
brew install mongodb-community
```

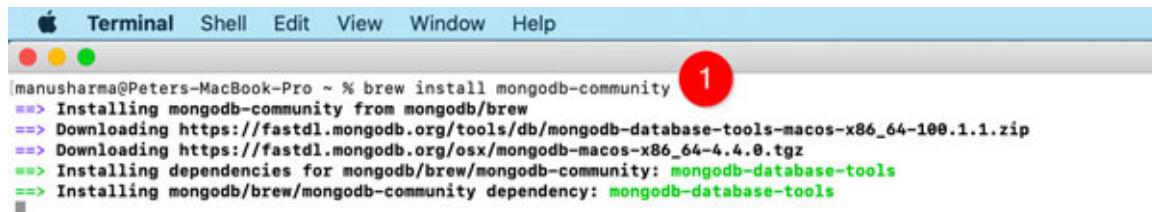
OR

```
brew install mongodb-community@<latest-version>
```



**Figure 4.9:** MongoDB Installation using Brew on MacOS

4. You will see the progress while the Homebrew Brew package manager is installing the MongoDB Community Server, as shown in the following screenshot:



**Figure 4.10:** MongoDB Installation on macOS – Brew Installing MongoDB Community Edition

5. Once the Homebrew Brew package manager has finished installing the MongoDB Community Server in your MacOS, you will see the message that the installation has successfully completed on your terminal screen, as shown in the following screenshot:

```

manusharma@Peters-MacBook-Pro ~ % brew install mongodb-community
==> Installing mongodb-community from mongodb/brew
==> Downloading https://fastdl.mongodb.org/tools/db/mongodb-database-tools-macos-x86_64-100.1.1.zip
==> Downloading https://fastdl.mongodb.org/osx/mongodb-macos-x86_64-4.4.0.tgz
==> Installing dependencies for mongodb/brew/mongodb-community: mongodb-database-tools
==> Installing mongodb/brew/mongodb-community dependency: mongodb-database-tools
[!] /usr/local/Cellar/mongodb-database-tools/100.1.1: 13 files, 172.9MB, built in 4 seconds
==> Installing mongodb/brew/mongodb-community
==> Caveats
To have launchd start mongodb/brew/mongodb-community now and restart at login:
  brew services start mongodb/brew/mongodb-community
Or, if you don't want/need a background service you can just run:
  mongod --config /usr/local/etc/mongod.conf
==> Summary
[!] /usr/local/Cellar/mongodb-community/4.4.0: 11 files, 136.7MB, built in 3 seconds
==> Caveats
==> mongodb-community
To have launchd start mongodb/brew/mongodb-community now and restart at login:
  brew services start mongodb/brew/mongodb-community
Or, if you don't want/need a background service you can just run: 3
  mongod --config /usr/local/etc/mongod.conf
manusharma@Peters-MacBook-Pro ~ %

```

**Figure 4.11: MongoDB Community Edition Installation on macOS – Installation Complete**

### Step 3 – Starting MongoDB on macOS

To make sure that MongoDB has been correctly installed on your macOS machine, follow these steps:

1. Start the MongoDB service using the following command:

```
brew services start mongodb-community
```

This will start the MongoDB service on your macOS based machine, as shown in the following screenshot:

```

manusharma@Peters-MacBook-Pro ~ % brew services start mongodb-community
==> Tapping homebrew/services
Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-services'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 916 (delta 0), reused 0 (delta 0), pack-reused 912
Receiving objects: 100% (916/916), 256.46 KiB | 1.94 MiB/s, done.
Resolving deltas: 100% (374/374), done.
Tapped 1 command (40 files, 337.3KB).
==> Successfully started `mongodb-community` (label: homebrew.mxcl.mongodb-community)
manusharma@Peters-MacBook-Pro ~ %

```

**Figure 4.12: Starting MongoDB service on MacOS**

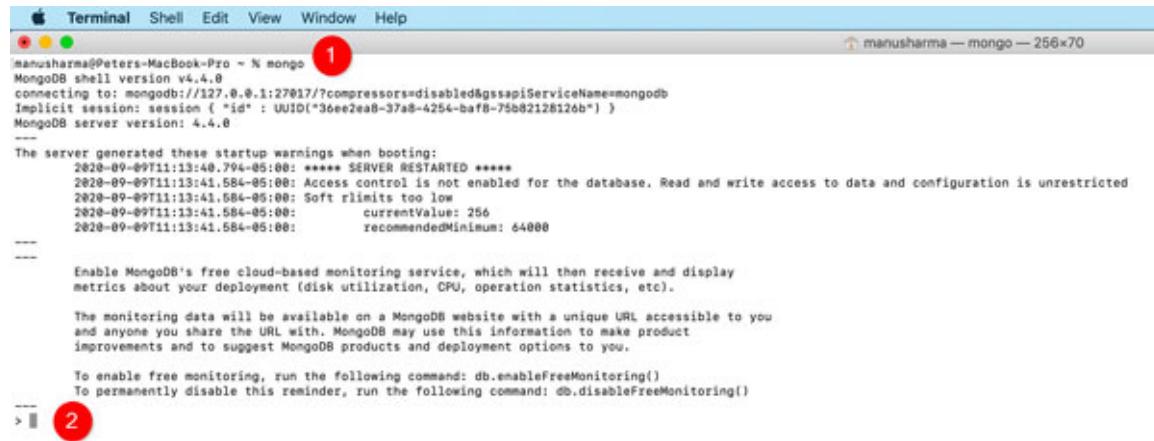
### Step 4 – Connecting to MongoDB on Mac OS

Let us now try to connect MongoDB from the command line from your macOS machine by using terminal. To connect to MongoDB from your macOS machine, follow these steps:

1. Open a separate terminal and run the following command:

```
mongo
```

This will open the MongoDB Shell interface to run various MongoDB commands, as shown in the following screenshot:



The screenshot shows a macOS Terminal window with the title bar "Terminal" and the path "manusharma@Peters-MacBook-Pro ~ % mongo". A red circle labeled "1" highlights the title bar. The main pane displays the MongoDB shell version 4.4.8 connecting to the local host. It shows startup warnings about server restarts, access control, soft limits, and monitoring. A red circle labeled "2" highlights the command prompt ">>".

```
manusharma@Peters-MacBook-Pro ~ % mongo
MongoDB shell version v4.4.8
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("36ee2ea8-37a8-4254-baf8-75b82128126b") }
MongoDB server version: 4.4.8
-- 
The server generated these startup warnings when booting:
2020-09-09T11:13:40.794+05:00: ***** SERVER RESTARTED *****
2020-09-09T11:13:41.584+05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-09-09T11:13:41.584+05:00: Soft limits too low
2020-09-09T11:13:41.584+05:00:    currentValue: 256
2020-09-09T11:13:41.584+05:00:    recommendedMinimum: 64000
-- 
-- 
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()
> 2
```

**Figure 4.13:** From macOS terminal – Running "mongo" command and accessing MongoDB Shell

2. To test further, issue the following MongoDB Shell command and press *Enter*:

```
show databases
```

This will give you the list of MongoDB databases that are present in your system, as shown in the following screenshot:

```

manusharma@Peters-MacBook-Pro ~ % mongo
MongoDB shell version v4.4.0
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("36ee2ea8-37a8-4254-baf8-75b82128126b") }
MongoDB server version: 4.4.0

The server generated these startup warnings when booting:
2020-09-09T11:13:41.794+05:00: ***** SERVER RESTARTED *****
2020-09-09T11:13:41.584+05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-09-09T11:13:41.584+05:00: Soft rlimits too low
2020-09-09T11:13:41.584+05:00:   currentValue: 256
2020-09-09T11:13:41.584+05:00:   recommendedMinimum: 64000

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()

> show dbs
admin 0.0000B
config 0.0000B
local 0.0000B
> 

```

**Figure 4.14:** Mongo Shell – "show databases" MongoDB command

- To exit from the MongoDB Shell, type the following command and press *Enter*:

```
exit
```

This will take you out from the MongoDB Shell, as shown in the following screenshot:

```

manusharma@Peters-MacBook-Pro ~ % mongo
MongoDB shell version v4.4.0
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("36ee2ea8-37a8-4254-baf8-75b82128126b") }
MongoDB server version: 4.4.0

The server generated these startup warnings when booting:
2020-09-09T11:13:41.794+05:00: ***** SERVER RESTARTED *****
2020-09-09T11:13:41.584+05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-09-09T11:13:41.584+05:00: Soft rlimits too low
2020-09-09T11:13:41.584+05:00:   currentValue: 256
2020-09-09T11:13:41.584+05:00:   recommendedMinimum: 64000

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()

> show dbs
admin 0.0000B
config 0.0000B
local 0.0000B
> exit
bye
manusharma@Peters-MacBook-Pro ~ 

```

**Figure 4.15:** Mongo Shell – "exit" MongoDB Command – To Exit from MongoDB

We learned how to install and setup MongoDB on macOS powered machines and how to verify MongoDB installation. We also learned to

start MongoDB using some commands, and at last, we learned how to connect to MongoDB Shell.

## **Conclusion**

In this chapter, we learned that MongoDB can be installed in macOS by first installing Homebrew, and then, by using the Homebrew tap, we can install the MongoDB Community Edition on our macOS operated machine. We have to run the Shell script in our terminal to first install the Homebrew on our macOS machine. Once Homebrew is installed, we can then use it to install the MongoDB Community Version on our macOS.

We also learned how to start the MongoDB service on macOS and how to start and use the Mongo Shell in macOS.

In next chapter, we will start with the basics of MongoDB. We will get the overview of the MongoDB database, MongoDB collections, and MongoDB documents. We will also learn about the difference between the terminologies used in MongoDB that are different from the other types of databases, like RDBMS in a detailed manner.

## **Questions**

1. What is Homebrew?
2. How can we check the version of our Mac OS?
3. What is the process of installing the MongoDB Community Server on Mac OS?
4. How can we start the MongoDB server on Mac OS?
5. How can you connect to the MongoDB Shell?

# CHAPTER 5

## Getting Started with MongoDB

This chapter covers the basics of MongoDB including the overview of MongoDB databases, MongoDB collections and MongoDB documents. This chapter will explain you the difference between the terminologies used in MongoDB and how it is different from the other databases like RDBMS in a more detailed manner. This chapter also gives you an overview of MongoDB Shell and covers some basic Shell commands. In the last topic, this chapter will give you an introduction to MongoDB clients which can be useful to connect to MongoDB server and perform various operations that you can perform easily using these MongoDB clients.

### Structure

In this chapter, we will discuss the following topics:

- MongoDB databases
- MongoDB collections
- MongoDB documents
- Introduction to MongoDB Shell
- Basic Shell commands
- Introduction to MongoDB clients

### Objectives

After studying this unit, you should be able to understand MongoDB database in a more detailed manner. You will also able to understand about MongoDB Collections and Documents in a more detailed manner. In the Later section of this Chapter you will able to learn about MongoDB Shell and how to connect and exit properly from MongoDB Shell. In the last section of this Chapter you will learn about MongoDB clients and how to use them.

## MongoDB databases

Let us explore what are MongoDB database in a detailed manner. Before we start with MongoDB databases, let's put some light on what exactly are databases, relational databases, and NoSQL databases.

### What is a database?

A database is a collection of data stored in the Database Management Systems (DBMS). Data is a small unit which, after processing, is converted into information. So, this data stays in the database in a structured manner and is then processed by the help of a DBMS to be converted into some information which is helpful for analyzing and decision making process.

So overall, the database allows us to store data in some structured manner like table, rows, and columns with the help of database management systems.

The main function of any DBMS is to:

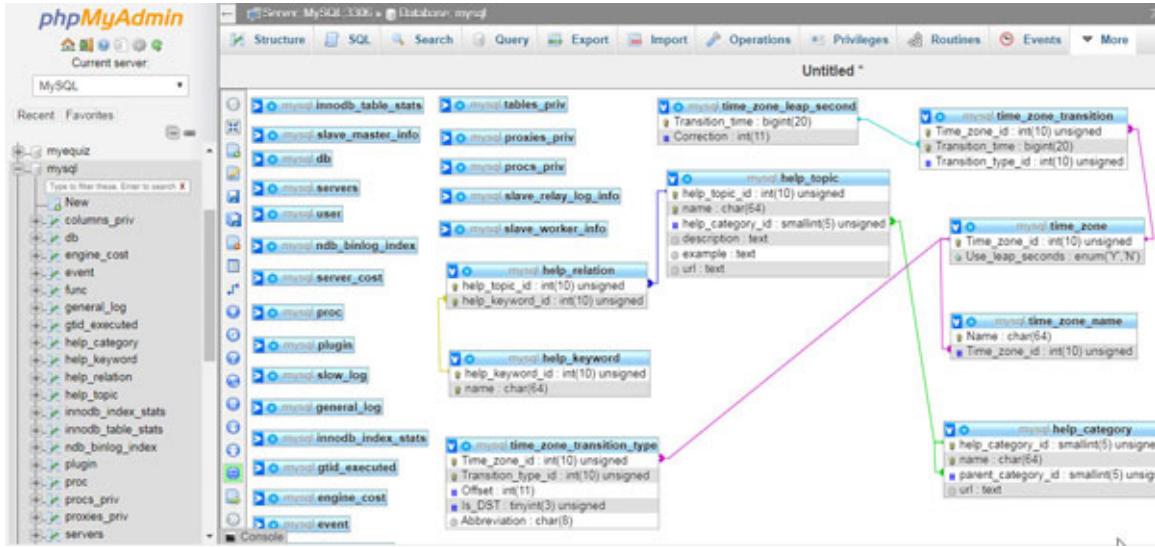
- Store the data
- Retrieve the data
- Manipulate the data
- Process the data

The preceding functions are performed using **Data Query Languages, like Structured Query Language (SQL)** or in our case, **MongoDB Query Language.(MQL)**

### What is a relational database?

A relational database contains data having some sort of relationships among two or more data records. These kinds of databases usually have tables interlinked with key-value or field-value pairs. For example, if there is a table which has a unique primary key column in table A then it is linked with table B which has a column that stores the primary key of table A, (also known as foreign key) in table B in a manner such that these two are interlinked. These types of interrelated or interlinked data tables contribute to form relational databases, as shown in the following figure. To manage these types of databases, we have

relational database management systems which helps us to store, retrieve or manipulate data from these types of databases:



**Figure 5.1: Relational Databases – Interconnected or Interlinked with Keys**

So far, we have studied what are relational databases and how tables in relational databases are interlinked with each other. Now, we will study about the NoSQL databases in the next section.

## What is a NoSQL database?

We have covered this in [Chapter 1: Introduction to MongoDB](#) also, but let's refresh it again. A NoSQL database is a kind of database that provides the mechanism of storage, retrieval and manipulations of data in a different approach than what the **Relational Database Management System** (an SQL based database) provides. It is sometimes called as a **NoSQL** database and sometimes "Not Only SQL".

MongoDB is one of the NoSQL databases which we will cover in this book.

It is a myth that NoSQL data cannot store relationship data while in reality NoSQL databases like MongoDB, stores relationship data in a different manner than SQL based databases. In case of NoSQL database like MongoDB, the relationship data is not stored and linked in a simpler manner because here, the relationship data doesn't split between the tables as it does in the SQL based databases.

NoSQL database provides us more flexibility in terms of storing the data and scaling, which is more developer-friendly.

## What is a MongoDB database?

MongoDB database is a document oriented database which stores data in JSON like documents with the dynamic schema. The concept of dynamic schema is to store the records without worrying too much on the structural part of the database. MongoDB database is a collection of data, or documents that have JSON like structure and are not very rigid in terms of schema. One document can be different from another one in the same collection.

MongoDB database architecture is designed on collections and documents. Here, the basic unit (which is data) consists sets of key-value or field-value pairs and allows documents to have different fields and structures according to its flexibility and dynamic schema.

## MongoDB collections

Let us explore what are MongoDB collections in a detailed manner. Before we start with MongoDB collections, let's have an overview of tables in RDBMS.

## What is a table in RDBMS?

A table is a set of related data stored in RDBMS in a form of tabular fashion as columns and rows. Here, the columns provide the structure and it is common for all the rows that are inserted in the table.

The table rows can have data according to the column type. For example, string, integer, float, BLOB, etc. In RDBMS, as defined by *E. F. Codd*, the inventor and father of the RMDM, the tables are termed as relations and rows as tuples.

RDBMS is a collection of set of tables organized in a manner that they are related to each other. In a RDBMS, is a collection of data elements organized as rows and columns, as shown in the following table:

Student ID	Student Name	Student Class	Student Roll No
1	Siya Sharma	4A	007

2	Harry Dsouza	7A	009
3	John Mathew	5E	015
4	Md. Hussain	10B	030
5	Preet Kaur	6C	041

**Table 5.1: A Typical table in RDBMS**

We can easily get an idea from the preceding table about how a typical table in RDBMS looks like.

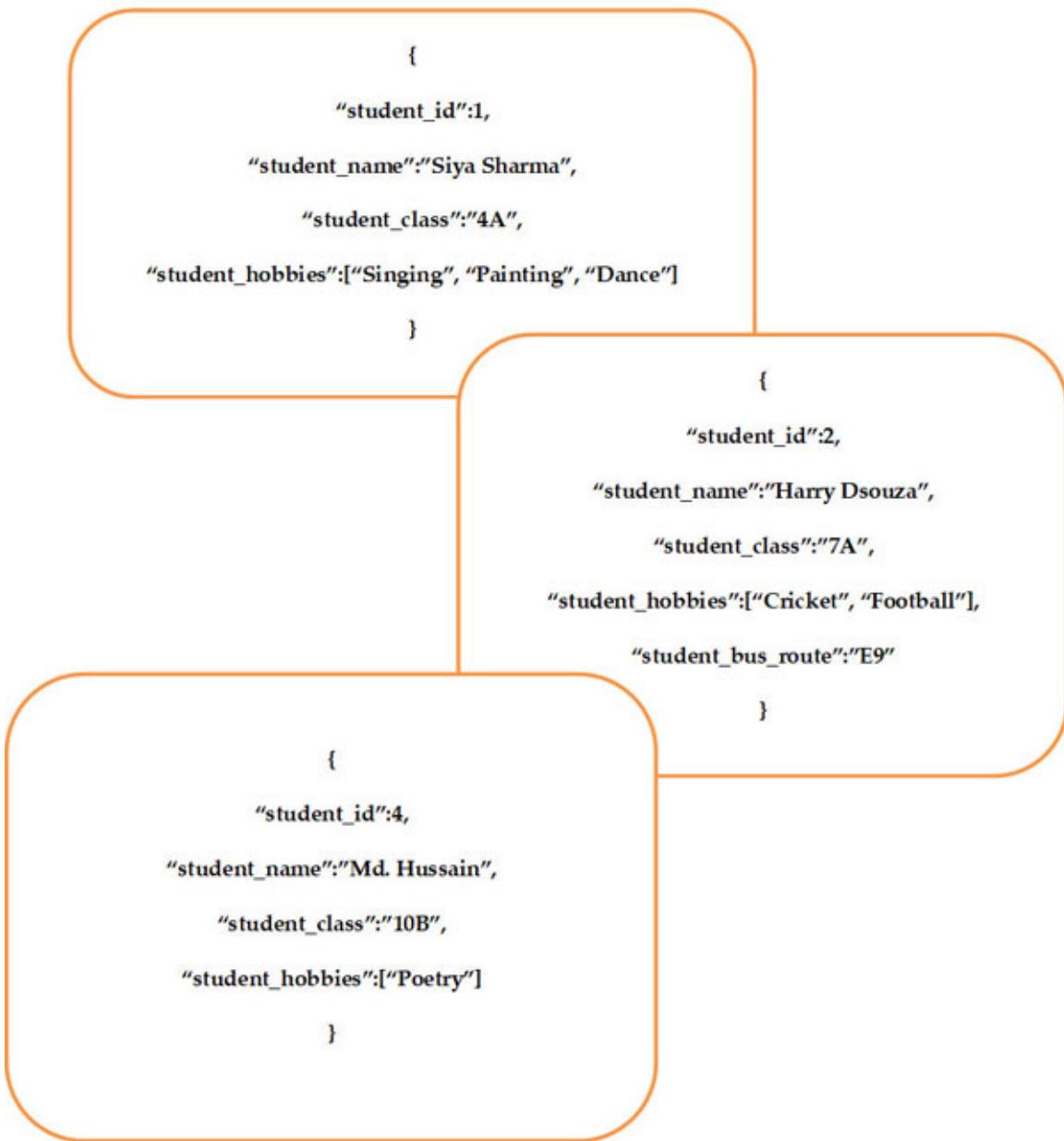
## What is a collection in MongoDB?

A collection in MongoDB is a grouping of MongoDB documents that hold data, usually dynamic in nature, because collection does not enforce a schema.

A collection is equivalent to a table in RDBMS. Here, the important aspect of collection is that it allows dynamic schema, which means that a collection can hold documents that could be different in terms of their structure.

In RDBMS, we know that schema (or structure) is enforced by the table and thus, the data in the rows should be consistent in terms of the number of columns and column type as defined in a table while creating an it. But, in MongoDB, collection can hold multiple documents which can have different documents with different data in terms of key-value or field-value pairs and data types, as shown in the following figure:

A Collection of Documents in MongoDB



**Figure 5.2:** Example of a Collection in MongoDB

We saw in the preceding figure how the collection binds different documents together in MongoDB.

## MongoDB documents

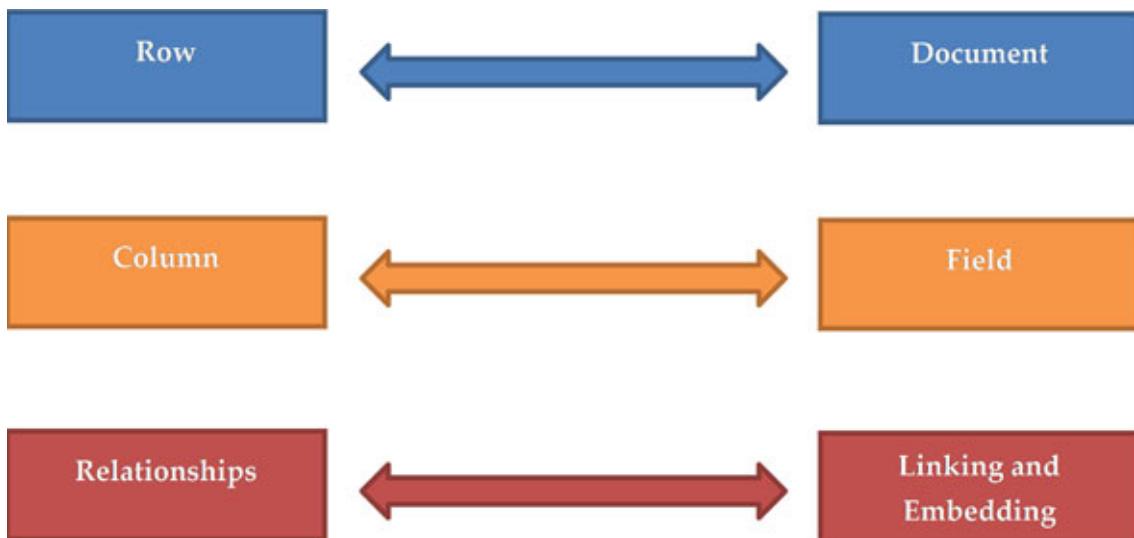
Let us explore what are MongoDB documents in a detailed manner. Before we start with MongoDB documents, we need to go through

some of the terms used in RDBMS and their equivalent in MongoDB.

## Row and column in RDBMS

We have studied in previous topics that row and columns constitute together to form a table in RDBMS. Same happens in MongoDB where the documents are bind by collections. The only difference in MongoDB is that the collection can store documents which are different in structure and have different data types and values.

To understand it in a better way, take a look at the following figure:

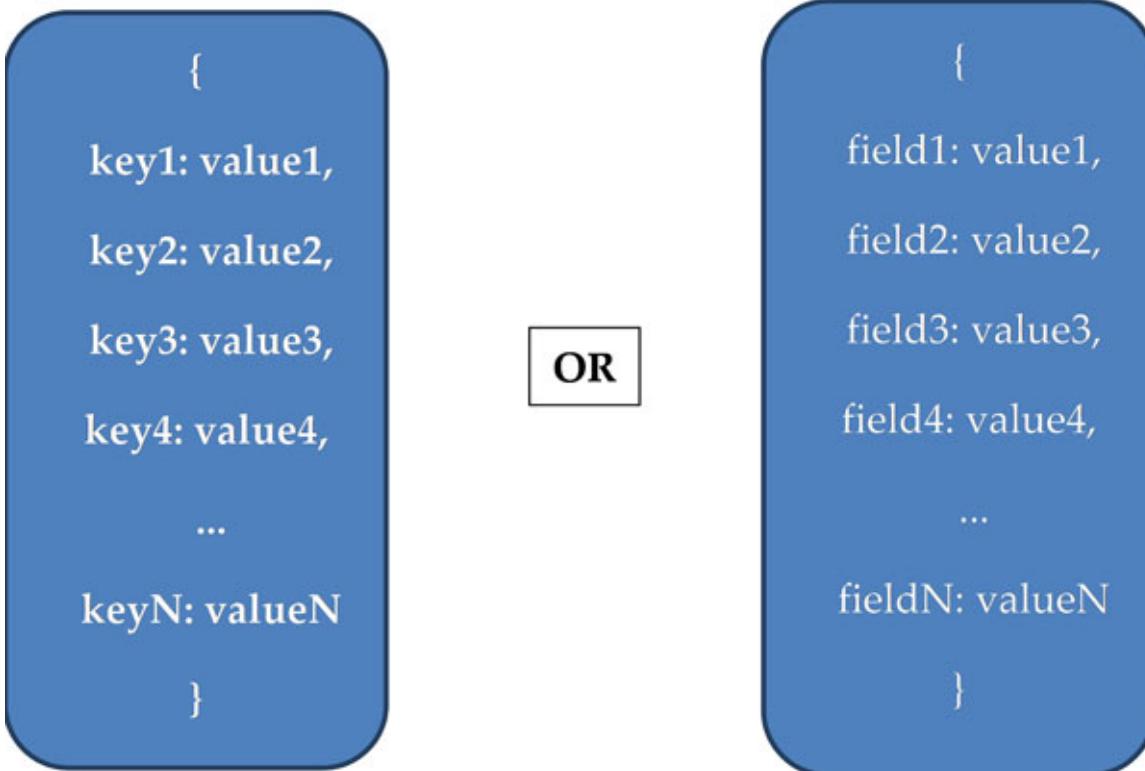


*Figure 5.3: RDBMS vs MongoDB Key Terms*

We can have a collection which can have two different documents, one with 5 fields and another one with 25 fields with different key-value or field-value pairs (refer to [Figure 5.4](#)).

## What is a document in MongoDB?

We know that MongoDB is a document based database and each record in MongoDB is termed as document. These documents are made up of key-value or field-value pairs which are just like JSON, as discussed in [Chapter 1: Introduction to MongoDB](#) of this book. These documents are stored in BSON (Binary JSON) format in MongoDB. A document in MongoDB is a binary format and is similar to JSON, as shown in the following figure:



**Figure 5.4:** MongoDB Document

Let us see the practical example of how Mongo DB document looks with some real data, as shown in the following figure:

```
[  
  {  
    "employee_id": "001",  
    "fname": "Manish",  
    "lname": "Sharma",  
    "department": "IT"  
  },  
  {  
    "employee_id": "002",  
    "fname": "Pooja",  
    "lname": "Kaushik",  
    "department": "HR"  
  },  
  {  
    "employee_id": "003",  
    "fname": "Shahid",  
    "lname": "Reza",  
  }
```

*Figure 5.5: MongoDB Document – Real Data Example*

## Introduction to MongoDB Shell

Let us explore what is a MongoDB Shell and how we can use it. We will give examples using the Windows platform as the Shell commands are the same for other platforms.

## What is a MongoDB Shell?

MongoDB Shell is an interface used in MongoDB which allows users to interact with MongoDB database for database-related queries to perform various **CRUD (Create, Read, Update, and Delete)** operations as well as administration of MongoDB.

When you install MongoDB server on your machine, MongoDB Shell is installed automatically by MongoDB installer program. It is also available as a standalone program and you can install it separately from MongoDB Inc. official website.

In order to connect to MongoDB Shell and work with it, you need to have MongoDB running on your machine. Otherwise, you will get an error that MongoDB Shell cannot be connected to the MongoDB server.

## Connecting to MongoDB Shell

To connect to MongoDB Shell, you need to follow these steps. These points are also mentioned in [Chapter 2: MongoDB Installation and Setup on Windows](#) of this book, but, we will cover it here also so that you can refresh them again. You may skip if you know these points

### Step 1 – Connecting to MongoDB Shell

Let us now try to connect MongoDB from command line from your Windows machine. To connect to MongoDB from your Windows machine, follow these steps:

1. Open a separate Shell command window and navigate to the `bin` directory of MongoDB.

The path could be as follows:

```
C:\Program Files\MongoDB\Server\4.4\bin
```



**Figure 5.6:** From Command Line – Navigate to MongoDB "bin" Directory.

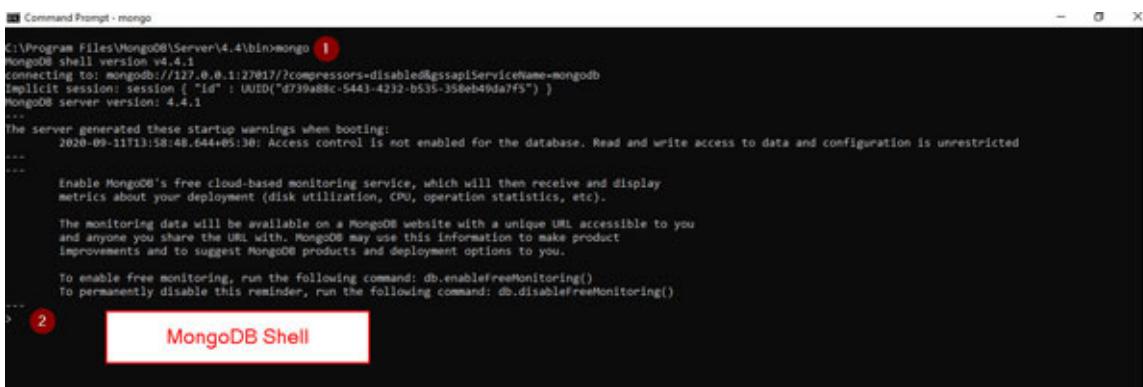
2. Now, give the following command and press *Enter*, as shown in the following screenshot:

```
"mongo"
```



**Figure 5.7:** From Command Line – Type "mongo" command and Press enter

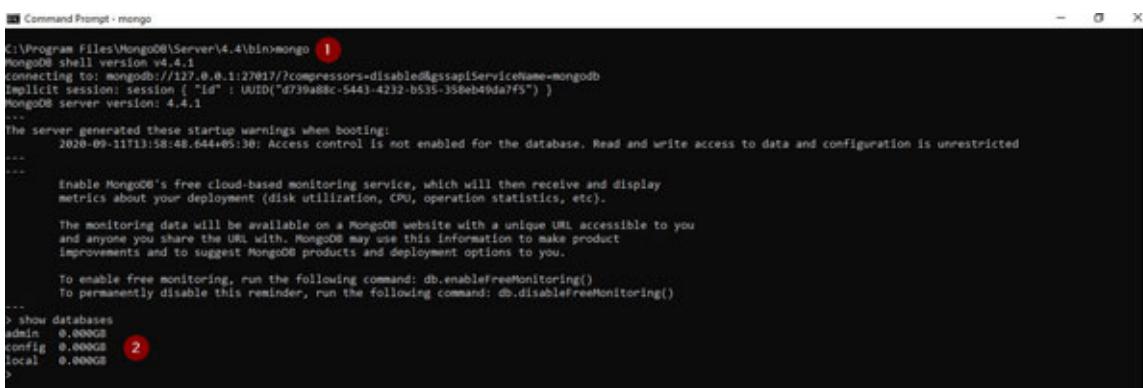
3. Once you give this command and press enter, it will open MongoDB Shell and you can type the MongoDB related commands, as shown in the following screenshot:



**Figure 5.8:** From Command Line – Type "mongo" command and Press enter

4. To test further, issue the following MongoDB command and press *Enter*, as shown in the following screenshot:

```
"show databases"
```

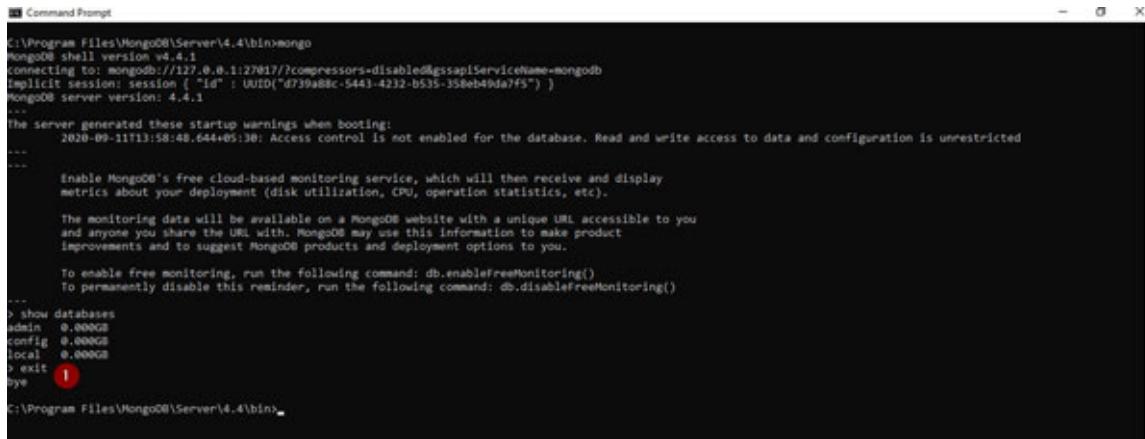


**Figure 5.9:** Mongo Shell – "show databases" MongoDB Command

5. To exit from MongoDB Shell, type the following command and press *Enter*:

```
"exit"
```

This will take you out from MongoDB Shell, as shown in the following screenshot:



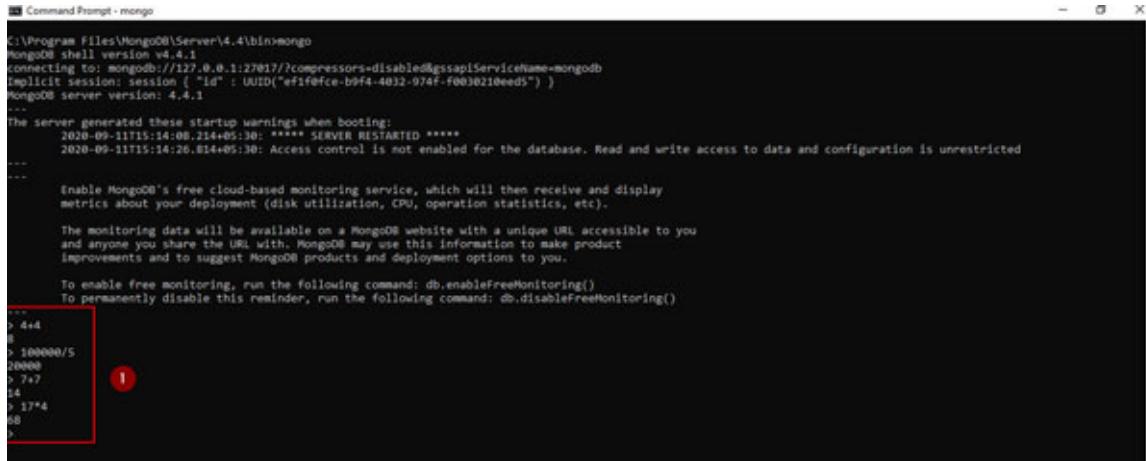
```
C:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d739a88c-5443-4232-b535-358eb49da7f5") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-09-11T13:58:48.644+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
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()
...
> show databases
admin 0.0000GB
config 0.0000GB
local 0.0000GB
> exit !  
C:\Program Files\MongoDB\Server\4.4\bin>
```

**Figure 5.10: Mongo Shell – "exit" MongoDB Command – To Exit from MongoDB**

In Step 1, we studied how to connect to MongoDB Shell. In the next topic, we will cover some basic Shell commands so that you will learn using and working with MongoDB Shell.

## Basic Shell commands

MongoDB Shell is a JavaScript based interface which allows you to run various CRUD (Create, Read, Update, and Delete) and administrative operations. As it is a JavaScript based interface, it has the ability to interpret JavaScript commands as well, other than MongoDB specific operations. A simple example is arithmetic operations like addition or multiplication, as shown in the following screenshot:



A screenshot of a Windows Command Prompt window titled "Command Prompt - mongo". The window shows the MongoDB shell version 4.4.1 connecting to a local host at port 27017. It displays several startup warnings and informational messages about monitoring and access control. In the bottom left corner, there is a red rectangular box highlighting a series of arithmetic operations entered by the user:

```
C:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "_id" : UUID("ef1fe0ce-b9f4-4032-974f-f0e30210eed5") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-09-11T15:14:08.214+05:30: ***** SERVER RESTARTED *****
2020-09-11T15:14:26.814+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
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()

> 4*4
8
> 1000000/5
200000
> 7*7
49
> 17*4
68
>
```

*Figure 5.11: Mongo Shell – Arithmetic Operations*

By using MongoDB Shell, we can also run few other JavaScript methods, as shown in the following screenshot:



A screenshot of a Windows Command Prompt window titled "Command Prompt - mongo". The window shows the MongoDB shell version 4.4.1 connecting to a local host at port 27017. In the bottom left corner, there is a red rectangular box highlighting the use of the JavaScript replace() method:

```
C:\Program Files\MongoDB\Server\4.4\bin>mongo
"Hello World!".replace("World", "MongoDB")
Hello MongoDB!
```

*Figure 5.12: Mongo Shell – JavaScript Replace method*

## MongoDB Shell basic command helpers

There are many helpers available from MongoDB command line. Let us explore few important ones one-by-one.

- **General Help Command**

Type "help" in your MongoDB Shell

This command is a general purpose help command which will give us the details of all the basic help commands available from MongoDB Shell, as shown in the following screenshot:

```

Command Prompt - mongo
> Help
1 db.help()          help on db methods
sh.help()           help on collection methods
rs.help()           sharding helpers
help admin          administrative help
help connect        connecting to a db help
help keys           key shortcuts
help misc           misc things to know
help mr             mapreduce

show dbs            show database names
show collections   show collections in current database
show users          show users in current database
show profile         show most recent system.profile entries with time >= 1ms
show logs           show the accessible logger names
show log [name]     prints out the last segment of log in memory, 'global' is default
use <db_name>       set current database
db.mycoll.find()    list objects in collection mycoll
db.mycoll.find( { a : 1 } ) list objects in mycoll where a == 1
it                 result of the last line evaluated; use to further iterate
DBQuery.shellBatchSize = x set default number of items to display on shell
exit               quit the mongo shell

```

**Figure 5.13: Mongo Shell – General Help Command**

- **DB related help command**

Type `db.help()` in your MongoDB Shell

This command will print the details of all the database methods available from MongoDB Shell, as shown in the following screenshot:

```

Command Prompt - mongo
> db.help()
1
96 methods:
db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]
db.aggregate([pipeline], {options}) - performs a collectionless aggregation on this database; returns a cursor
db.auth(username, password)
db.cloneDatabase(fromhost) - will only function with MongoDB 4.0 and below
db.commandHelp(name) returns the help for the command
db.copyDatabase(fromdb, todb, fromhost) - will only function with MongoDB 4.0 and below
db.createCollection(name, {size: ..., capped: ..., max: ...})
db.createUser(userDocument)
db.createView(viewName, viewOn, [{$operator: {...}}, ...], {viewOptions})
db.currentOp() displays currently executing operations in the db
db.dropDatabase(writeConcern)
db.dropUser(username)
db.eval() - deprecated
db.fsyncLock() flush data to disk and lock server for backups
db.fsyncUnlock() unlocks server following a db.fsyncLock()
db.getCollection(cname) same as db['cname'] or db.cname
db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
db.getCollectionNames()
db.getLastError() - just returns the err msg string
db.getLastErrorObj() - return full status object
db.getComponents()
db.getMongo() - get the server connection object
db.getMongo().setSecondaryOk() allow queries on a replication secondary server
db.getNs() - deprecated
db.getProfilingLevel() - deprecated
db.getProfilingStatus() - returns if profiling is on and slow threshold
db.getReplicationInfo()
db.getSiblingDB(name) get the db at the same server as this one
db.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server object if set
db.hostInfo() get details about the server's host
db.isMaster() check replica primary status
db.killOp(opid) kills the current operation in the db
db.listCommands() lists all the db commands
db.loadServerScripts() loads all the scripts in db.system.js
db.logout()
db.printCollectionStats()
db.printReplicationInfo()
db.printShardingStatus()
db.printSecondaryReplicationInfo()
db.resetError()
db.runCommand(cmdObj) run a database command. If cmdObj is a string, turns it into {cmdObj: 1}
db.serverStatus()

```

**Figure 5.14: Mongo Shell – "db.help()" - DB Help Command Showsvarious DB Methods which we can use**

- **Show databases command**

Type `show databases` in your MongoDB Shell.

This command will print the details of all the database methods available from the MongoDB Shell, as shown in the following screenshot:

```
Command Prompt - mongo
> show databases
BPBOnlineDB 0.000GB
admin 0.000GB
config 0.000GB
local 0.000GB
```

**Figure 5.15:** Mongo Shell – "show databases" - To Show Databases Command.

- **Use <DB> Command**

Type "`use <db>`" in your MongoDB Shell. Here, `<db>` is the name of your database.

Example: If the name of your database is `BPBOnlineDB`, then you will type: `use BPBOnlineDB` in the Shell prompt.

This command will now let us use or access `BPBOnlineDB` database so that we can perform some actions on this database, as shown in the following screenshot:

```
Command Prompt - mongo
> show databases
BPBOnlineDB 0.000GB
admin 0.000GB
config 0.000GB
local 0.000GB
> use BPBOnlineDB
switched to db BPBOnlineDB
```

**Figure 5.16:** Mongo Shell – "use <db>" - To access and Use Specific Database Command.

- **Show collections command**

Type `show collections` in your MongoDB Shell. Note that this command will only work if you have selected the database using `use <db>` command first.

This command will show you the list of collections that exists in the database you are using by `use <db>` command, as shown in the following screenshot:

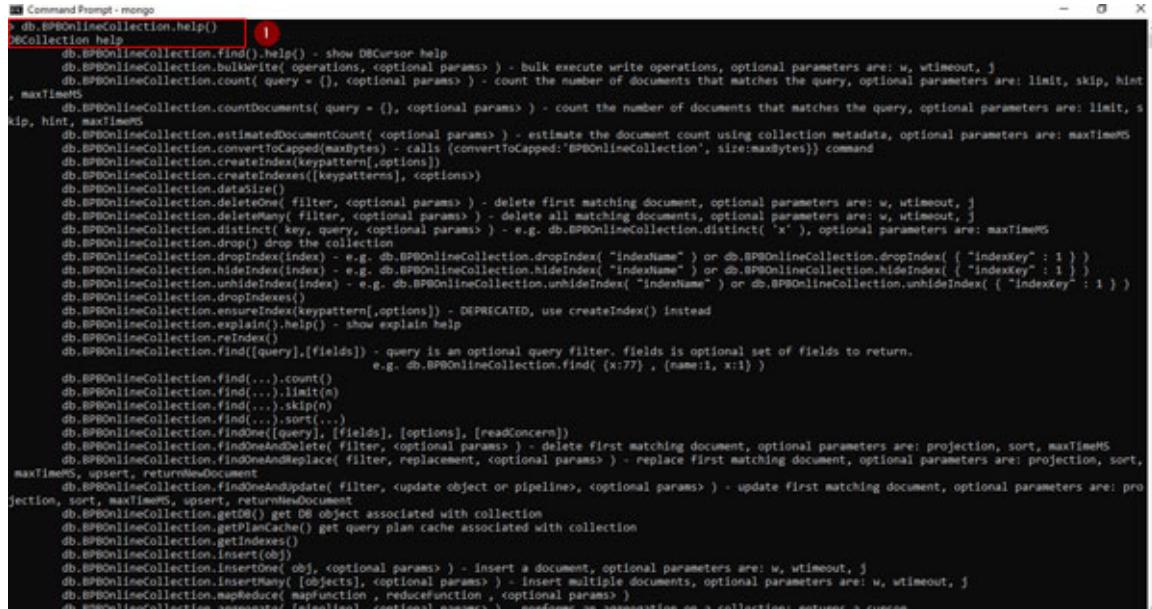
```
Command Prompt - mongo
> use BPBOnlineDB
switched to db BPBOnlineDB
> show collections
BPBOnlineCollection
```

**Figure 5.17:** Mongo Shell – "show collections" Command - To show the list of Collections exists in the particular database.

- **Collection related help commands**

Type `db.<collection-name>.help()` in your MongoDB Shell, where `<collection-name>` is the name of your collection.

This command will show you the list of collection-related methods that you can use, as shown in the following screenshot:



```
Command Prompt - mongo
> db.BPBOlineCollection.help()
db.BPBOlineCollection.find() - show DBcursor help
db.BPBOlineCollection.bulkWrite( operations, <optional params> ) - bulk execute write operations, optional parameters are: w, wtimeout, j
db.BPBOlineCollection.count( query = {}, <optional params> ) - count the number of documents that matches the query, optional parameters are: limit, skip, hint
, maxTimeMS
db.BPBOlineCollection.countDocuments( query = {}, <optional params> ) - count the number of documents that matches the query, optional parameters are: limit, skip, hint, maxTimeMS
db.BPBOlineCollection.estimatedDocumentCount( <optional params> ) - estimate the document count using collection metadata, optional parameters are: maxTimeMS
db.BPBOlineCollection.convertToCapped(maxBytes) - calls {convertToCapped:'BPBOlineCollection', size:maxBytes} command
db.BPBOlineCollection.createIndex(keypattern[,options])
db.BPBOlineCollection.createIndexes(keypatterns[,options])
db.BPBOlineCollection.dataSize()
db.BPBOlineCollection.deleteOne( filter, <optional params> ) - delete first matching document, optional parameters are: w, wtimeout, j
db.BPBOlineCollection.deleteMany( filter, <optional params> ) - delete all matching documents, optional parameters are: w, wtimeout, j
db.BPBOlineCollection.distinct( key, query, <optional params> ) - e.g. db.BPBOlineCollection.distinct( 'x' ), optional parameters are: maxTimeMS
db.BPBOlineCollection.drop() - drop the collection
db.BPBOlineCollection.dropIndex(index) - e.g. db.BPBOlineCollection.dropIndex( "indexName" ) or db.BPBOlineCollection.dropIndex( { "indexKey" : 1 } )
db.BPBOlineCollection.hideIndex(index) - e.g. db.BPBOlineCollection.hideIndex( "indexName" ) or db.BPBOlineCollection.hideIndex( { "indexKey" : 1 } )
db.BPBOlineCollection.unhideIndex(index) - e.g. db.BPBOlineCollection.unhideIndex( "indexName" ) or db.BPBOlineCollection.unhideIndex( { "indexKey" : 1 } )
db.BPBOlineCollection.dropIndexes()
db.BPBOlineCollection.ensureIndex(keypattern[,options]) - DEPRECATED, use createIndex() instead
db.BPBOlineCollection.explain([query],<optional params>) - show explain help
db.BPBOlineCollection.reIndex()
db.BPBOlineCollection.find(query,[fields]) - query is an optional query filter, fields is optional set of fields to return.
e.g. db.BPBOlineCollection.find( {name:1, x:1} )
db.BPBOlineCollection.find(...).count()
db.BPBOlineCollection.find(...).limit(n)
db.BPBOlineCollection.find(...).skip(n)
db.BPBOlineCollection.find(...).sort(...)
db.BPBOlineCollection.findOne([query],[fields],[options],[readConcern])
db.BPBOlineCollection.findOneAndDelete( filter, <optional params> ) - delete first matching document, optional parameters are: projection, sort, maxTimeMS
db.BPBOlineCollection.findOneAndReplace( filter, replacement, <optional params> ) - replace first matching document, optional parameters are: projection, sort, maxTimeMS, upsert, returnNewDocument
db.BPBOlineCollection.findOneAndUpdate( filter, <update object or pipeline>, <optional params> ) - update first matching document, optional parameters are: projection, sort, maxTimeMS, upsert, returnNewDocument
db.BPBOlineCollection.getDB() - get DB object associated with collection
db.BPBOlineCollection.getPlanCache() - get query plan cache associated with collection
db.BPBOlineCollection.getIndexes()
db.BPBOlineCollection.insert(obj)
db.BPBOlineCollection.insertOne( obj, <optional params> ) - Insert a document, optional parameters are: w, wtimeout, j
db.BPBOlineCollection.insertMany( [objects], <optional params> ) - Insert multiple documents, optional parameters are: w, wtimeout, j
db.BPBOlineCollection.mapReduce( mapfunction , reducefunction , <optional params> )
```

**Figure 5.18: Mongo Shell –"db.<collection-name>.help()" Command - To show the list of Collections methods that are available to be used with particular collection.**

In this section, we have covered many basic MongoDB commands that are helpful to work with MongoDB. MongoDB Shell also stores the history of the commands which we cover in the next section.

## MongoDB Shell command history

MongoDB stores the history of the commands that you run in a session. You can easily retrieve the previous commands you have used by the up and down arrow keys.

- **The up-arrow key** is used to retrieve the previous command you have used from your current position.
- **The down -arrow key** is used to retrieve the next command you have used from your current position.

We have studied that by using the Up and Down arrow keys we can easily access the commands from the MongoDB Shell history. These commands are those commands which we have already used while

working on MongoDB Shell. In the next section, we will get introduced to MongoDB clients.

## **Introduction to MongoDB clients**

In MongoDB or database world, client is a program or application which provides user the interface, be it command line or GUI (Graphical User Interface) to connect, use, manage and administer the database.

There are various MongoDB clients available in the market to use, right from the simplest MongoDB client, a MongoDB command line Shell program to the MongoDB Inc. compass, a GUI (Graphical User Interface) system to manage and administer MongoDB. We also have various other tools available in the market. Few of them are as follows:

- MongoDB Inc Compass - <https://www.mongodb.com/products/compass>
- Studio 3T - <https://studio3t.com/>
- RoboMongo - <https://robomongo.org>
- NoSQL Manager - <https://www.mongodbmanager.com>
- NoSQL Booster - <https://nosqlbooster.com>

We will cover the official client, MongoDB Inc. compass, in [Chapter 16](#) of this book in more detail.

## **Conclusion**

In this chapter, we learned what is MongoDB database as well as the difference between the SQL Databases with NoSQL databases. We have also studied about MongoDB collections and MongoDB documents. We have also covered MongoDB Shell and how we can use it to run various commands related to MongoDB database.

In the last topic, we also gave a small overview of various MongoDB clients. We will cover more advanced topics in more detail in the upcoming chapters.

## **Questions**

1. What are NoSQL databases?

2. How are NoSQL databases different from SQL-based databases?
3. Explain MongoDB collection and MongoDB document in detail.
4. How to create a database in MongoDB using MongoDB shell?
5. What are database clients?
6. Can you name some MongoDB clients?

# CHAPTER 6

## Storage Engines in MongoDB

In this chapter, we will cover the concept of storage engines in database management systems and why they are used. We will discuss the storage engine with the help of diagrams. We will also cover the storage engines that are used in MongoDB such as the WiredTiger storage engine as well as the in-memory storage engine. We will also cover encrypted storage engines and third-party pluggable storage engines in this chapter and also compare the main storage engines with their features. In this chapter, we will also cover the concept of locks in the database and the overview of locks in MongoDB.

### Structure

In this chapter, we will discuss the following topics:

- What are storage engines?
- Types of storage engines in MongoDB
- Introduction to the WiredTiger storage engine
- Introduction to the in-memory storage engine
- Encrypted storage engine
- Third-party pluggable storage engines
- MongoDB locks

### Objectives

After studying this unit, you should be able to understand the concept of storage engines in the database management systems and learn about the storage engines used in MongoDB, including the MongoDB's WiredTiger storage engine, in-memory storage engine, and encrypted storage engine. Later in this chapter, you will also

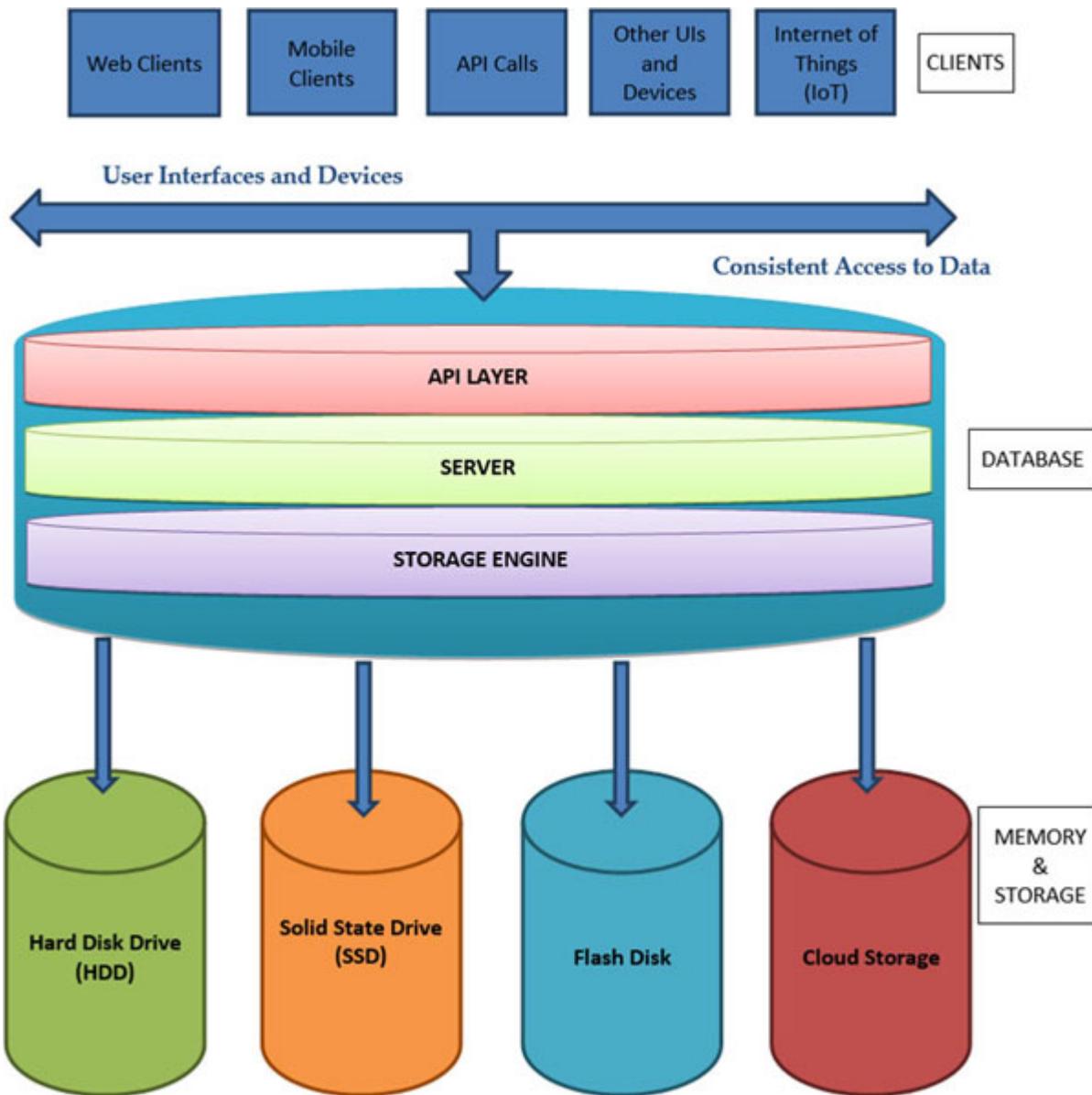
learn about the third-party pluggable storage engines that you can also use with MongoDB.

## What are storage engines?

Storage engine is a software component that works in the **Database Management System (DBMS)** to provide the user **CRUD (Create, Read, Update, and Delete)** functionality. The database engine defines a way to store the data into the database and also the manner in which the data is stored in both the memory and the disk.

The database uses its database engine to store and retrieve data to and from the memory and disk. So, this means that with the help of storage engines, the user will be able to perform various operations like CRUD, etc.

Various databases use their own **API (Application Programming Interface)** that works in between the database engine and user to provide the medium for the user interaction with the database and database engine as shown in the following figure:



*Figure 6.1: Database Storage Engine*

So, it is very important to select the right storage engine for your application from the beginning itself.

## Types of storage engines in MongoDB

We know that storage engines are very vital components of the database management systems and they are used to manage and store data in the memory and disk. MongoDB supports multiple

storage engines; each of which has some unique features. Let's take a look at the major storage engines supported by MongoDB.

The following storage engines are supported by MongoDB:

- WiredTiger
- In-memory
- Encrypted
- MMAPv1 (from version 4.2 and above, this storage engine is not supported by MongoDB and it has been removed from the later versions)

Each application has different needs and different workloads. Some of the applications are write-intensive, some are read-intensive, and some applications may require data encryption. So, the development team chooses various storage engines according to the needs of the application.

## Introduction to the WiredTiger storage engine

The WiredTiger storage engine is the default storage engine for MongoDB starting from MongoDB version 3.2 and above. This is best suited for most of the workloads and thus, it is recommended for most of the applications.

The WiredTiger storage engine uses document-level concurrency control for write operations, which means that it can handle multiple requests without conflicting with each other. In case, there is a conflict between two operations, then the WiredTiger storage engine will retry that operation with ease. While using WiredTiger as a storage engine, MongoDB utilizes the WiredTiger internal cache as well as the file system cache. WiredTiger is helpful in the efficient use of the CPU and RAM and is also helpful when it comes to tuning the database storage engine. It can be tuned more than the MMAP storage engine.

WiredTiger allows up to 7 to 10 times more write performance than other storage engines. It also reduces the storage and achieves up to 80% less storage with compression. Compression also reduces the CPU load and overhead.

## Introduction to the in-memory storage engine

The in-memory storage engine is available in the MongoDB Enterprise Edition only starting from version 3.2.6 and above. These provide high output with low latency and high availability. The in-memory storage engines support high-level infrastructure based on zonal sharding.

These also come with MongoDB rich query capability and indexing support.

## Encrypted storage engine

MongoDB Enterprise provides encryption support for WiredTiger starting from the MongoDB Enterprise version 3.2 and above. This feature allows MongoDB to encrypt data and decrypt whenever required.

Many a times, data encryption is forced by the government bodies or some industry standards like HIPAA, PCI-DSS, and FERPA. These are some security standards and guidelines which help in ensuring compliance with security and privacy policies of the industry or organization.

## Third-party pluggable storage engines

MongoDB also supports the third-part storage engines. These storage engines can be plugged in like modules and can also be independently updated. An example of the third-party storage engine is RocksDB developed by Facebook Inc. and designed to handle write-intensive workloads. The RocksDB storage engine is the first one to use the module system for the MongoDB storage integration layer.

## MongoDB storage engines comparison

The comparison of storage engines in MongoDB is shown in the following table:

	Wired Tiger	In-memory	Encrypted

Versions	From Community	3.0 +	Enterprise	Enterprise	
Concurrency Level	Document	Document	Document	Document	
Write Performance	Excellent	Excellent	Good	Good	
Read Performance	Excellent	Excellent	Good	Good	
Disk Compression	Good	Excellent	Good	Good	
Query Language	Yes	Yes	Yes	Yes	
Secondary Index Support	Yes	Yes	Yes	Yes	
Replication Support	Yes	Yes	Yes	Yes	
Sharding Support	Yes	Yes	Yes	Yes	
Ops & Cloud Manager Support	Yes	Yes	Yes	Yes	
Native Encryption (REST)	No	N/A	Yes	Yes	
Read Concern	Yes	Yes	Yes	Yes	
Security Controls	Yes	Yes	Yes	Yes	
Larger than RAM Datasets	Yes	No	Yes	Yes	
Platform Availability	Windows, Linux, OSX	Mac	Windows, Linux, OSX	Mac	Windows, Linux, OSX

**Table 6.1: Database Storage Engine in MongoDB – Comparison of Features**

So far, we have learned what storage engines are and how they play an important role in the database management systems. We have also covered the available storage engines in MongoDB and their features and comparison.

## MongoDB locks

While inserting data into the database, MongoDB creates locks with the help of its storage engine, like WiredTiger. Earlier, we learned how this is done in MongoDB. Now, let's study what exactly a database lock is and why it is required.

## What is a database lock?

A database lock is a mechanism used by the database to prevent a scenario where two users or two sessions modify the same data at the same time. What this means is that there could be a scenario where there are two or more database users who are working on the same set of data and want to update the same data or record. In this case, there would be lot of issues that would arise. The mostly aroused situation could be related to which data is latest and up-to-date.

In order to prevent this, a database creates a *lock* when any update is done to a particular set of data and after the update is done, it releases the lock so that other users or sessions can work on that particular data.

Therefore, in the database world, in order to achieve the concurrency, multiple users are concurrently working in the database and its records. This is then achieved by the database concurrency control mechanisms which are done by the help of database engines.

## Database lock operations types

There are mainly three different types of locking operation that are mostly done in databases:

- Read lock operation
- Write lock operation
- Unlock operation

Lock operations can further be classified as shared lock or exclusive lock. Let us now learn about how the locks are used in MongoDB.

## Database locks operations in MongoDB

MongoDB uses multiple granularity locking. This type of locking ensures that the database locking can be done at the child level (record level). MongoDB allows multiple clients to read and write

data at the same time and thus, it uses locking at different levels and other concurrency control methods to achieve this.

MongoDB's WiredTiger engine uses intent locks as well as optimistic concurrency control and 1) Global 2) Database and 3) Collection Level, so it always prevents the conflicts. MongoDB also allows the database engines to implement their own concurrency and locking mechanisms at the document level.

In case WiredTiger detects any write conflict with any client, then that client will transparently retry its operation.

## **Conclusion**

In this chapter, we studied about database storage engines with the help of diagrams. We also studied the storage engines available in MongoDB and their features. The concept of database locking was also covered and we learned the different types of locking available in MongoDB and how to achieve those.

In the next chapter, we will cover how to manage and administer MongoDB with the help of MongoDB Shell commands and this will be the intermediate level of MongoDB management.

## **Questions**

1. What are storage engines?
2. Why are storage engines so useful in databases?
3. What are the two major storage engines of MongoDB?
4. Explain the WiredTiger storage engine with the help of some features.
5. What are database locks?
6. Why are locks used in databases?

## CHAPTER 7

# Managing and Administering MongoDB

This chapter will cover the basic commands and methods used to manage and administer MongoDB. We would learn these commands and methods with the help of MongoDB Shell. We will learn how to create, update, and delete databases, collections and documents using the MongoDB Shell with the help of MongoDB Shell commands and methods. We will also learn how to create, update, and delete documents using the MongoDB query and write operations Shell commands and methods. Later in this chapter, we will learn about the MongoDB authentication and role based access methods and how to use them in MongoDB using the MongoDB Shell commands and methods.

### Structure

In this chapter, we will discuss the following topics:

- MongoDB administration commands and methods
- MongoDB query and write operation commands and methods
- Query and write operation commands MongoDB user authentication and role based commands and methods

### Objectives

After studying this unit, you should be able to learn about MongoDB administrative command and methods and how to use them in MongoDB later in the chapter we will also learn query and write operation commands and methods in MongoDB and in the last section of this chapter we will learn about the user authentication and role based commands and methods in MongoDB.

## MongoDB administration commands and methods

We will start this chapter with some administration commands and methods which are used very frequently and are used before any other command related to MongoDB. These commands and methods are used to create databases and collections and to insert a document. Let us check them one by one. To run these commands and methods, you need to run the MongoDB Shell as explained in the previous chapters.

### Create database command

This command is useful to create a database. Type the following command in your MongoDB Shell prompt to create a database in MongoDB:

```
"use <database-name>"
```

Here, `<database-name>` is the name of your database you want to create. In our example, let us create a database with the name `BPBOnlineBooksDB`, as shown in the following screenshot:

```
"use BPBOnlineBooksDB"
```



**Figure 7.1: MongoDB Administration Commands and Methods – Creating a Database "use <database-name>"**

Note that this command `use <database-name>` can also be used to switch to any database. When you use this command to create a database, you must create a collection and insert at least one record so that the database gets created. This is done using the following method:

```
"db.<collection-name>.insert()  
db.BPBOnlineBooksDBCollection.insert({ "book-title": "Mastering  
MongoDB" })
```

Once you run this method, it will create a collection in the database with a single document based on the key-pair values of the JSON string that you provide. This method will automatically create a collection if it

doesn't exists. In our case, we have created a collection named BPBOnlineBooksDBCollection and a simple document with book-title as Mastering MongoDB, as shown in the following screenshot:



```
Administrator: Command Prompt - mongo
> use BPBOnlineBooksDB
switched to db BPBOnlineBooksDB
> db.BPBOnlineBooksDBCollection.insert({"book-title": "Mastering MongoDB"})
writeResult({ "nInserted" : 1 })
```

**Figure 7.2: MongoDB Administration Commands and Methods – Creating a Database, Collection and Document using "db.<collection-name>.insert()"**

So, we have learnt about the MongoDB administration commands and methods and how to create database, collection, and documents using these administration commands and methods. Let us now learn some more MongoDB Shell commands and methods.

## Create collection command

This command is used to create a collection in database in MongoDB. Before you run this command, you need to switch to a database with `use <database-name>` command we studied earlier.

Type the following command in your MongoDB Shell prompt to switch to your database:

```
"use <database-name>"
```

After this, type the following command to create a collection:

```
db.createCollection("<collection-name>")
db.createCollection("BPBOnlineBooksDBCollection-V2")
```

Once you run this command, it will create a collection in a database. In our case, we have created a new collection named BPBOnlineBooksDBCollection-V2, as shown in the following screenshot:



```
Administrator: Command Prompt - mongo
> use BPBOnlineBooksDB
switched to db BPBOnlineBooksDB
> db.createCollection("BPBOnlineBooksDBCollection-V2")
{ "ok" : 1 }
```

**Figure 7.3: MongoDB Administration Commands and Methods – Creating a Collection using db.createCollection(<collection-name>)"**

MongoDB can create a collection automatically if a collection doesn't exist. So, type the following command to insert a document in a

MongoDB selected database:

```
"db.<collection-name>.insert()"  
db.BPBOnlineBooksDBCollection-V3.insert({"book-title": "Mastering  
MongoDB with JavaScript"})
```

- In the 1st step, we switched to our database named BPBOnlineBooksDB
- In the 2nd step, we printed the list of collections which are in the current database
- In the 3rd step, we inserted the document
- In the 4th step, we again printed the list of collections in the current database

It will automatically create a collection, as shown in the following screenshot:

The screenshot shows a Windows Command Prompt window titled 'Administrator: Command Prompt - mongo'. The mongo shell is running. Step 1: The user switches to the database 'BPBOnlineBooksDB' with the command 'use BPBOnlineBooksDB'. Step 2: The user runs 'show collections' to see existing collections: 'BPBOnlineBooksDBCollection' and 'BPBOnlineBooksDBCollection-V2'. Step 3: The user inserts a document into the collection 'BPBOnlineBooksDBCollectionV3' with the command 'db.BPBOnlineBooksDBCollectionV3.insert({"book-title": "Mastering MongoDB with JavaScript"})'. Step 4: The user runs 'show collections' again to see the updated list: 'BPBOnlineBooksDBCollection', 'BPBOnlineBooksDBCollection-V2', and 'BPBOnlineBooksDBCollectionV3'.

**Figure 7.4: MongoDB Administration Commands and Methods – Creating a Collection using "db.<collection-name>.insert()" - Collection is automatically created even if doesn't exists**

So far, we have learnt to create a collection using the MongoDB Shell commands and methods. Let us now learn some more interesting MongoDB Shell commands and methods with step-by-step practical examples.

## Drop database command

This command is used to delete or remove a database in MongoDB. Before you run this command, you need to switch to a database with `use <database-name>` command that we have studied earlier.

Type the following command in your MongoDB Shell prompt to switch to your database:

```
"use <database-name>"
```

After this, type the following command to create a collection:

```
db.dropDatabase()
```

In our case, let's try to delete the database `BPBOnlineBooksDBV2Collection`. To do so, we have to run the following commands and methods in the Shell prompt, as shown in the following screenshot:

```
use BPBOnlineBooksDBV2
db.BPBOnlineBooksDBV2Collection.insert({"book-title": "The
introduction to Cobol"})
show dbs
db.dropDatabase()
show dbs
```

The screenshot shows a Windows Command Prompt window titled 'Administrator: Command Prompt - mongo'. It displays the following MongoDB shell session:

```
> use BPBOnlineBooksDBV2
switched to db BPBOnlineBooksDBV2
> db.BPBOnlineBooksDBV2Collection.insert({"book-title": "The introduction to Cobol"})
2
> show dbs
BPBOnlineBooksDB 0.000GB
BPBOnlineBooksDBV2 0.000GB
Employees 0.000GB
admin 0.000GB
config 0.000GB
local 0.000GB
> db.dropDatabase()
{"dropped": "BPBOnlineBooksDBV2", "ok": 1}
4
> show dbs
BPBOnlineBooksDB 0.000GB
Employees 0.000GB
admin 0.000GB
config 0.000GB
local 0.000GB
5
```

Annotations with red circles numbered 1 through 5 point to specific lines of the command history:

- Line 1: The command `use BPBOnlineBooksDBV2`.
- Line 2: The command `db.BPBOnlineBooksDBV2Collection.insert({ "book-title": "The introduction to Cobol" })`.
- Line 3: The output of the `show dbs` command.
- Line 4: The command `db.dropDatabase()`.
- Line 5: The output of the `show dbs` command after the database has been dropped.

**Figure 7.5: MongoDB Administration Commands and Methods – Dropping a Database in MongoDB**

So far, we learned how to drop or delete a database using the MongoDB Shell commands and methods. Let us now learn some more interesting MongoDB Shell commands and methods with step-by-step practical examples.

## Drop collection command

This command is used to delete or remove collections in MongoDB. Before you run this command, you need to switch to a database with `use <database-name>` command that we studied earlier.

Type the following command in your MongoDB Shell prompt to switch to your database:

```
"use <database-name>"
```

After this, type the following command to create a collection:

```
db.<collection-name>.drop()
```

In our case, let us try to delete a collection named `BPBOnlineBooksDBV3Collection`. To do so, we have to run the following commands and methods in the Shell prompt, as shown in the following screenshot:

```
use BPBOnlineBooksDB
db.BPBOnlineBooksDBV3Collection.insert({"book-title": "The
introduction to Pascal"})
show collections
db.BPBOnlineBooksDBV3Collection.drop()
show collections
```



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - mongo". The shell session is as follows:

```
Administrator: Command Prompt - mongo
> use BPBOnlineBooksDB
switched to db BPBOnlineBooksDB
> db.BPBOnlineBooksDBV3Collection.insert({"book-title": "The introduction to Pascal"}) ②
> show collections
BPBOnlineBooksDBCollection
BPBOnlineBooksDBCollection-V2 ③
BPBOnlineBooksDBCollectionV3
BPBOnlineBooksDBV3Collection
> db.BPBOnlineBooksDBV3Collection.drop() ④
true
> show collections
BPBOnlineBooksDBCollection
BPBOnlineBooksDBCollection-V2 ⑤
BPBOnlineBooksDBCollectionV3
```

Annotations with red circles numbered 1 through 5 point to specific lines of the command history:

- 1: The command `use BPBOnlineBooksDB`.
- 2: The command `db.BPBOnlineBooksDBV3Collection.insert({ "book-title": "The introduction to Pascal" })`.
- 3: The collection `BPBOnlineBooksDBCollection-V2`.
- 4: The command `db.BPBOnlineBooksDBV3Collection.drop()`.
- 5: The collection `BPBOnlineBooksDBCollection-V2` again, likely a typo or a different state.

**Figure 7.6: MongoDB Administration Commands and Methods – Dropping a Collection in MongoDB**

So far, we learned how to drop or delete a collection using the MongoDB Shell commands and methods. Let us now learn some more interesting MongoDB Shell commands and methods with step-by-step practical examples.

## MongoDB query and write operation commands and methods

There are certain Shell commands and methods which are very helpful for read and write operations. These commands and methods are used to perform the CRUD operations at document or collection level in MongoDB. Let us go through them one by one.

### Insert document command

This command is used to insert a document in collection. Type the following command to insert a document in MongoDB. Before you run this command, you need to switch to a database with `use <database-name>` command.

Type the following command in your MongoDB Shell prompt to switch to your database:

```
"use <database-name>"
```

After this, type the following command to insert or create a document in a collection:

```
db.<collection-name>.insert()
```

In our case, let's try to insert a document in a collection named `BPBOnlineBooksDBV4Collection`. To do so, we have to run the following commands and methods in the Shell prompt, as shown in the following screenshot:

```
use BPBOnlineBooksDB
db.BPBOnlineBooksDBV4Collection.insert(
{
  "book-isbn-number": "1234567890",
  "book-title": "The introduction to Qbasic",
  "book-price": "INR 500"
}
)
show collections
```



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - mongo". The mongo shell is running. The user has switched to the database "BPBOnlineBooksDB" (marked with a red circle 1). Then, they run the command `db.BPBOnlineBooksDBV4Collection.insert({ ... })` (marked with a red circle 2). Finally, they run `show collections` (marked with a red circle 3) to see the available collections: BPBOnlineBooksDBCollection, BPBOnlineBooksDBCollection-V2, BPBOnlineBooksDBCollection-V3, and BPBOnlineBooksDBV4Collection.

**Figure 7.7: MongoDB Administration Commands and Methods – Inserting a Document "db.<collection-name>.insert()"**

So far, we learned how to create a document using the MongoDB Shell commands and methods. Let us now learn some more interesting

MongoDB Shell commands and methods with step-by-step practical examples.

## Read document command

This command is used to read documents or a specific document in a collection. Before you run this command, you need to switch to a database with `use <database-name>` command.

Type the following command in your MongoDB Shell prompt to switch to your database:

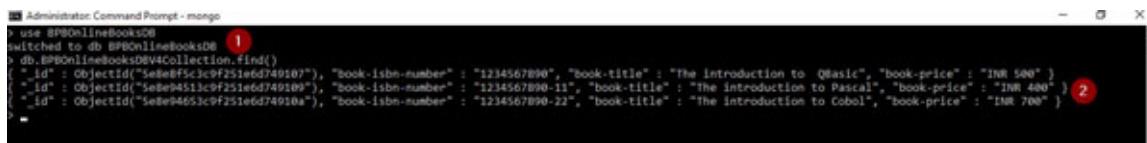
```
"use <database-name>"
```

After this, type the following command to read a document in a collection:

```
db.<collection-name>.find()
```

In our case, let us try to read all documents in the collection named `BPBOnlineBooksDBV4Collection`. So, we need to run the following commands and methods in the Shell prompt, as shown in the following screenshot:

```
use BPBOnlineBooksDB
db.BPBOnlineBooksDBV4Collection.find()
```



```
Administrator Command Prompt - mongo
> use BPBOnlineBooksDB
switched to db BPBOnlineBooksDB
> db.BPBOnlineBooksDBV4Collection.find()
[{"_id": ObjectId("5ebeff5c3c9f251e6d749107"), "book-isbn-number": "1234567890", "book-title": "The introduction to QBasic", "book-price": "INR 500"}, {"_id": ObjectId("5ebeff5c3c9f251e6d749109"), "book-isbn-number": "1234567890-11", "book-title": "The introduction to Pascal", "book-price": "INR 400"}, {"_id": ObjectId("5ebeff5c3c9f251e6d74910a"), "book-isbn-number": "1234567890-22", "book-title": "The introduction to Cobol", "book-price": "INR 700"}]
```

**Figure 7.8: MongoDB Administration Commands and Methods – Reading Documents**  
`"db.<collection-name>.find()"`

You can also format the results in a formatted manner to make it prettier to read. To do this, just add `pretty()` after `find()`. This will show you the documents in a more readable format, as shown in the following screenshot:

```

Administrator Command Prompt - mongo
> use BPBOnlineBooksDB
switched to db BPBOnlineBooksDB
> db.BPBOnlineBooksDBV4Collection.find()
[{"_id": ObjectId("5ebe94513c9f251e6d749107"), "book-isbn-number": "1234567890", "book-title": "The introduction to QBasic", "book-price": "INR 500"}, {"_id": ObjectId("5ebe94513c9f251e6d749108"), "book-isbn-number": "1234567890-11", "book-title": "The introduction to Pascal", "book-price": "INR 400"}, {"_id": ObjectId("5ebe94513c9f251e6d74910a"), "book-isbn-number": "1234567890-22", "book-title": "The introduction to Cobol", "book-price": "INR 700"}]
> db.BPBOnlineBooksDBV4Collection.find().pretty()
[{"_id": ObjectId("5ebe94513c9f251e6d749107"), "book-isbn-number": "1234567890", "book-title": "The introduction to QBasic", "book-price": "INR 500"}, {"_id": ObjectId("5ebe94513c9f251e6d749108"), "book-isbn-number": "1234567890-11", "book-title": "The introduction to Pascal", "book-price": "INR 400"}, {"_id": ObjectId("5ebe94513c9f251e6d74910a"), "book-isbn-number": "1234567890-22", "book-title": "The introduction to Cobol", "book-price": "INR 700"}]
>

```

pretty() Method to display documents in more readable format

**Figure 7.9: MongoDB Administration Commands and Methods – Reading Documents in Formatted Manner"db.<collection-name>.find().pretty()"**

So far, we have learnt how to read documents using the MongoDB Shell commands and methods. Let us now learn some more interesting MongoDB Shell commands and methods with step-by-step practical examples.

## Delete document command

This command is useful to delete or remove documents or a specific document in collection. Before you run this command, you need to switch to a database with `use <database-name>` command.

Type the following command in your MongoDB Shell prompt to switch to your database:

```
"use <database-name>"
```

After this, type the following command to read a document in a collection:

```
db.<collection-name>.remove()
```

In our case, let's try to delete or remove a document in a collection based on a specific `_id` from a collection named `BPBOnlineBooksDBV4Collection`. To do so, we need to run the following commands and methods in the Shell prompt, as shown in the following screenshot:

```
use BPBOnlineBooksDB
db.BPBOnlineBooksDBV4Collection.find().pretty()
```

```

db.BPBOnlineBooksDBV4Collection.remove({ "_id" : 
ObjectId("5e8e94513c9f251e6d749109") })
db.BPBOnlineBooksDBV4Collection.find().pretty()

```

Note that in step 3 we have used the `_id` to delete a specific document which has this `_id`. By this, we have deleted only a specific document based on a key-value pair.

The screenshot shows the MongoDB Administration Commands and Methods – Deleting a Document "db.<collection-name>.remove()" process. It consists of four numbered steps:

- `> db.BPBOnlineBooksDBV4Collection.find().pretty()`
- `> db.BPBOnlineBooksDBV4Collection.remove({ "_id" : ObjectId("5e8e94513c9f251e6d749109") })`
- `> db.BPBOnlineBooksDBV4Collection.find().pretty()`
- `> db.BPBOnlineBooksDBV4Collection.remove({ "_id" : ObjectId("5e8e94513c9f251e6d749109") })`

The collection contains three documents, each with an \_id, book-isbn-number, book-title, and book-price. Step 1 shows the initial state. Step 2 shows the removal of the first document. Step 3 shows the state after one document is removed. Step 4 shows the final state where the second document is removed.

```

Administrator Command Prompt - mongo
> use BPBOnlineBooksDB
switched to db BPBOnlineBooksDB
> db.BPBOnlineBooksDBV4Collection.find().pretty()
{
    "_id" : ObjectId("5e8e94513c9f251e6d749109"),
    "book-isbn-number" : "1234567890",
    "book-title" : "The introduction to QBasic",
    "book-price" : "INR 500"
}

{
    "_id" : ObjectId("5e8e94513c9f251e6d749109"),
    "book-isbn-number" : "1234567890-11",
    "book-title" : "The introduction to Pascal",
    "book-price" : "INR 400"
}

{
    "_id" : ObjectId("5e8e94513c9f251e6d749109"),
    "book-isbn-number" : "1234567890-22",
    "book-title" : "The introduction to Cobol",
    "book-price" : "INR 700"
}
> db.BPBOnlineBooksDBV4Collection.remove({ "_id" : ObjectId("5e8e94513c9f251e6d749109") }) 1
> db.BPBOnlineBooksDBV4Collection.find().pretty()
{
    "_id" : ObjectId("5e8e94513c9f251e6d749109"),
    "book-isbn-number" : "1234567890",
    "book-title" : "The introduction to QBasic",
    "book-price" : "INR 500"
}

{
    "_id" : ObjectId("5e8e94513c9f251e6d749109"),
    "book-isbn-number" : "1234567890-22",
    "book-title" : "The introduction to Cobol",
    "book-price" : "INR 700"
}
>

```

*Figure 7.10: MongoDB Administration Commands and Methods – Deleting a Document "db.<collection-name>.remove()"*

So far, we have learnt how to delete documents using the MongoDB Shell commands and methods. Let us now learn some more interesting MongoDB Shell commands and methods with step-by-step practical examples.

## MongoDB user authentication and role based commands and methods

Now, we will look into some of the MongoDB's authentication commands and methods. We will cover the basic authentication methods within the build roles and we will also cover advanced topics related to authentication in the following advanced chapters.

To start with, let us understand some basics.

## What is database authentication?

Database authentication is a process of allowing access to the right user who has the right credentials to access the database. The credentials are username and password.

Some databases, like MongoDB, allow us to create a **role based access control (RBAC)**.

## What is role-based access control?

In a role-based access control, the users accessing a database are allowed only to access the database based on their roles in the organization. Not everyone needs full privileges or rights to access the database as super admin. Some people only require the read only access while some people, like system administrators, may require the full access to the database.

This allows the user to access the data (or information) which is only related to their jobs in the organization.

## Role-based authentication in MongoDB

MongoDB comes with in-built roles. In our example, we will use `userAdmin` role. This role has the privileges of an administrator.

1. Type the following command in your MongoDB Shell prompt:

```
"use <database-name>"
```

Here, `<database-name>` is the name of your database you want to give role based access to. In our example, let us create a database with the name `BPBOnlineBooksDBWithAuth`, as shown in the following screenshot:

```
"use BPBOnlineBooksDBWithAuth"
```

2. Now, we will use the following command and settings to setup the authentication to this database:

```
db.createUser(  
  {  
    user: "manusharma",  
    pwd: "admin1234",  
    roles:  
      [  
        {  
          role: "userAdmin",  
          db: "BPBOnlineBooksDBWithAuth"  
        }  
      ]  
  })
```

```

    {
      role: "userAdmin",
      db: "BPBOnlineBooksDBWithAuth"
    }
  ]
}

Administrator: Command Prompt - mongo -u "manusharma" -p "admin123" --authenticationDatabase "BPBOnlineBooksDBWithAuth"
> use BPBOnlineBooksDBWithAuth
switched to db BPBOnlineBooksDBWithAuth ①
> db.createUser(
...   {
...     user: "manusharma",
...     pwd: "admin1234",
...     roles: [
...       {
...         role: "userAdmin",
...         db: "BPBOnlineBooksDBWithAuth"
...       }
...     ]
...   }
... )
successfully added user: {
  "user": "manusharma",
  "roles": [
    {
      "role": "userAdmin",
      "db": "BPBOnlineBooksDBWithAuth"
    }
  ]
}

```

**Figure 7.11:** MongoDB Administration Commands and Methods – Creating a User with Role

In the previous step, we learned how to create a user with a role using the MongoDB Shell commands and methods. Let us now move on to the next step:

3. Restart your `mongod` service with `--auth` parameter, as shown in the following screenshot:

```
mongod --auth --port 27017 --dbpath "c:\Program Files\MongoDB\Server\4.2\data\db"
```



**Figure 7.12:** MongoDB Administration Commands and Methods – Restarting mongod Service with --auth parameter

4. Authenticate with username, password and database using the `mongo` shell. In our example, we have used username, password and database name to authenticate, as shown in the following screenshot:

```
mongo -u "manusharma" -p "admin1234" --authenticationDatabase
"BPBOnlineBooksDBWithAuth"
```

```
c:\Program Files\MongoDB\Server\4.2\bin>mongo -u "manusharma" -p "admin1234" --authenticationDatabase "BPBOnlineBooksDBWithAuth"
MongoDB shell version v4.2.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "_id" : UUID("8668cdcd-491a-4194-aF7e-940de2c2e564") }
MongoDB server version: 4.2.5
Server has startup warnings:
2020-04-07T00:14:00.162+0530 I CONTROL [initandlisten]
2020-04-07T00:14:00.163+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-04-07T00:14:00.164+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-04-07T00:14:00.166+0530 I CONTROL [initandlisten]
...
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()
...
```

**Figure 7.13: MongoDB Administration Commands and Methods – Restarting mongod Shell with parameters**

We can also authenticate using `db.auth("<username>", "<password>")` method. Here, `<username>` is your username and `<password>` is your password which you have used while creating this database user. In this case, we simply run the Mongo command to access the mongo Shell without any parameters and then use the `db.auth()` method, as shown in the following screenshot.

It will show "1" if the authentication is successful, else, it will show an error as shown in the following screenshot:

```
c:\Program Files\MongoDB\Server\4.2\bin>mongo
MongoDB shell version v4.2.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "_id" : UUID("40a401bd-acde-4b8e-9e5a-a0ff95c48dbe") }
MongoDB server version: 4.2.5
Server has startup warnings:
2020-04-07T00:14:00.162+0530 I CONTROL [initandlisten]
2020-04-07T00:14:00.163+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-04-07T00:14:00.164+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-04-07T00:14:00.166+0530 I CONTROL [initandlisten]
...
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()
...
> use BPBOnlineBooksDBWithAuth
switched to db BPBOnlineBooksDBWithAuth
> db.auth('manusharma', 'admin1234')
1
>
```

**Figure 7.14: MongoDB Administration Commands and Methods – Authentication using `db.auth("<username>", "<password>")` method**

## Conclusion

In this chapter, we learned the basic commands and methods used for managing and administrating MongoDB. We learned these commands and methods with the help of MongoDB Shell. We also learned how to create, update, and delete databases, collections and documents using the MongoDB Shell with the help of MongoDB Shell commands and methods. We also learned how to create, read and delete documents using the MongoDB query and write operations Shell commands and methods.

Later in this chapter, we learned about the MongoDB authentication and role-based access methods and how we can use these in MongoDB using the MongoDB Shell commands and methods.

In the next chapter, we will learn about the MongoDB Shell methods in a more detailed manner in which we will learn about the JavaScript in MongoDB, list of official supported languages in MongoDB, the MongoDB connection methods, MongoDB database methods, MongoDB collection methods, and MongoDB cursor methods.

## **Questions**

How can we create database in MongoDB? Explain this by using the MongoDB Shell.

1. How can we create a collection in MongoDB? Explain this by using the MongoDB Shell.
2. How can we create a document in MongoDB with the help of MongoDB Shell?
3. Is it possible to delete database with the MongoDB Shell? Explain the steps.
4. How can we delete documents by using the MongoDB Shell?
5. What do you understand by authentication?
6. What is role based access?

# CHAPTER 8

## MongoDB Shell Methods

In the previous chapter, we learned about the MongoDB Shell commands and some methods. In this chapter, we will learn more about the MongoDB Shell methods used to connect to the MongoDB server. We will start with JavaScript in MongoDB and will cover an overview of list of various other languages which are officially supported by MongoDB. We will also cover the commands related to the database, various methods related to the database management and collections and how we can manipulate the MongoDB collections using these methods. Towards the end, we will cover cursor in MongoDB and various cursor related methods that we can use. These methods are very useful and we can use them in various scenarios while working with MongoDB and later when we will learn some advanced topics and application development using MongoDB.

### Structure

In this chapter, we will discuss the following topics:

- MongoDB database methods
- MongoDB collection methods
- MongoDB cursor methods

### Objectives

After studying this unit, you should be able to:

- Learn about JavaScript in MongoDB
- Learn about the other languages officially supported by MongoDB
- Learn about MongoDB connection using Shell methods
- Learn about MongoDB database Shell methods
- Learn about MongoDB collection Shell methods

- Learn about MongoDB cursor Shell methods

## JavaScript in MongoDB

You have learnt in the previous chapter that the MongoDB Shell is a JavaScript based interface that allows you to run various commands. The MongoDB Shell has an ability to interpret JavaScript commands too other than the MongoDB specific operations which we have covered in our previous chapter.

## Server Side JavaScript in MongoDB

MongoDB has various methods and operators which use server side execution of JavaScript. We will cover these in more details in our next chapters. But to give an overview of these, we will now cover the overview of two main topics:

- MapReduce
- \$where

## What is map-reduce in MongoDB?

Map-reduce is a process or method in MongoDB in which large volume of data is processed, filtered and then reduced to a set of small number of cluster of data. This is a group of data which is combined to form a set which contains some information. So, the map-reduce method is generally used to process a large set of data.

## What is \$where operator in MongoDB?

\$where is an operator in JavaScript which is used to match documents in MongoDB which matches certain criteria given in a JavaScript expression. Normally, we pass the JavaScript expression or function in \$where operator.

We will cover these topics in a more detailed manner in the advanced chapters.

## List of officially supported languages in MongoDB

We learned in [\*Chapter 1: Introduction to MongoDB\*](#) of this book that MongoDB drivers depend on programming languages and help applications for various CRUD and other operations with respect to the MongoDB database. So, MongoDB supports many other languages other than JavaScript. So, while communicating with the apps, MongoDB communicates with the diversity of these languages.

Following is the list of languages officially supported by MongoDB:

- C
- C++
- C#
- Go
- Java
- Node.js
- Perl
- PHP
- Python
- Ruby
- Scala
- Swift

Other than these languages, there are many other languages supported by the MongoDB community, and their drivers have been built by the MongoDB community.

## **MongoDB methods**

We will cover various MongoDB methods in this chapter. But before we do that, let us open up a MongoDB Shell.

## **Step 1 – Connecting to MongoDB Shell**

Let us now try to connect MongoDB from the command line from your Windows machine. To connect to MongoDB from your Windows machine, follow these steps:

1. Open a command window and navigate to the `bin` directory of MongoDB.

The path could be as shown in the following figure:

```
C:\Program Files\MongoDB\Server\4.2\bin
```



**Figure 8.1: From Command Line – Navigate to MongoDB "bin" Directory.**

2. Now, give the following command and press *Enter*, as shown in the following figure:

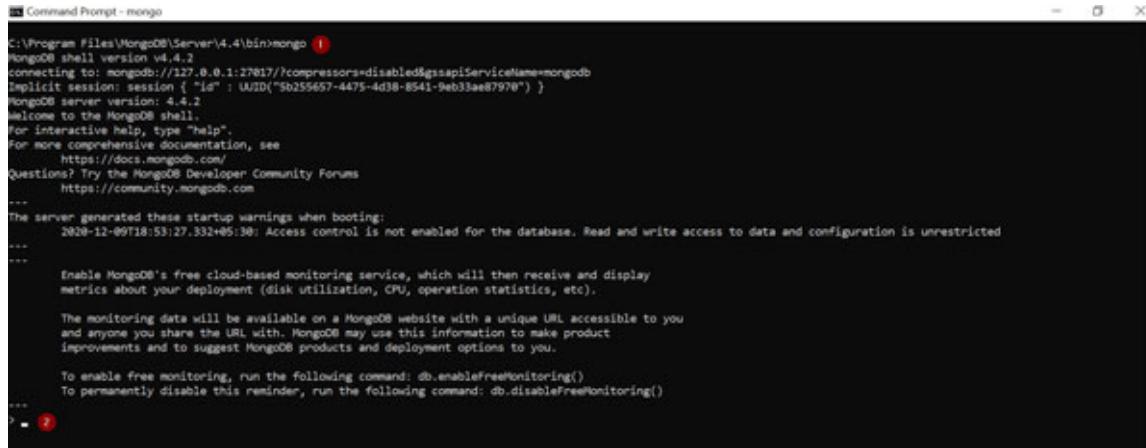
```
"mongo"
```

Press *Enter*



**Figure 8.2: From Command Line – Type "mongo" command and Press enter**

3. Once you give this command and press *Enter*, it will open the Mongo Shell and we can type the Mongo related commands, as shown in the following figure:



**Figure 8.3: The MongoDB Shell**

Now, we have entered into the MongoDB Shell. We can now use various MongoDB methods using the MongoDB Shell. Let us now learn

about these methods.

## MongoDB connection methods

We will cover some most common and important methods related to connection which are helpful in connecting to the MongoDB server.

Some of these methods are as follows:

- `connect()`
- `Mongo()`
- `Mongo.getDB()`

Let us study them one-by-one.

### [connect\(url,username,password\)](#)

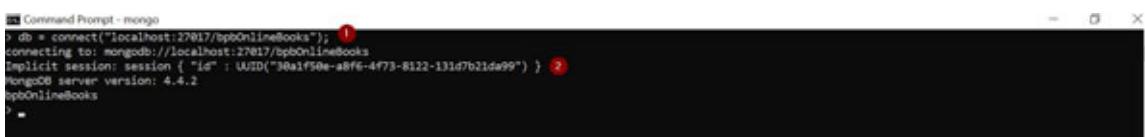
- This method creates the connection to the MongoDB instance.
- This method returns the reference to the MongoDB database.
- It takes up to 3 parameters.
- The first parameter is mandatory and it takes the URL with host name, port and database.
- The last two parameters, which are username and password, are optional.

The first parameter value can be as follows:

- `<hostname>:<port-number>/<database>`
- `<hostname>/<database>`
- `<database>`

We will use the following example, as shown in the following figure:

```
db = connect("localhost:27017/bpbOnlineBooks");
```



A screenshot of a Windows Command Prompt window titled "Command Prompt - mongo". The command entered is `db = connect("localhost:27017/bpbOnlineBooks");`. The output shows the connection process: "connecting to: mongodb://localhost:27017/bpbOnlineBooks", "Implicit session: session { "id" : UUID("30a1f50e-a8f6-4f73-8122-131d7b21da99") }", and "MongoDB server version: 4.4.2 bpbOnlineBooks". The prompt then changes to a single greater than sign (>).

**Figure 8.4: The MongoDB `connect()` Method**

We can now see that after we run this method, it returns the reference to the MongoDB database. We can also use the `Mongo()` and `Mongo.getDB()` methods which are recommended methods to use.

## [Mongo\(host, clientSideOptions\)](#)

- This method initiates the connection to the MongoDB database either from the MongoDB Shell or the JavaScript file.
- It takes up to 2 parameters.
- The first parameter is optional and it takes the host name or host name with port. If these values are omitted, it will initiate the connection to local host with the default port number, which is 27017, in case of MongoDB.
- The last parameter is also optional and it contains the parameters for client side field level encryption.

The first parameter value can be as follows:

- `<hostname>`
- `<hostname>:<port-number>`

We will use the example as shown in the following figure:

```
MongoDBConnection = Mongo("mongodb://localhost:27017/");
```



*Figure 8.5: The MongoDB Mongo() Method*

After we run this method, we have the connection available to run our next methods. Let us now try to run the `Mongo.getDB()` method.

## [Mongo.getDB\(database\)](#)

The following are the details of this method, explained with some points:

- This method provides the access to MongoDB database objects from the MongoDB Shell or the JavaScript file.

- It takes only 1 parameter.
- The parameter is mandatory and it takes the database name which we want the access to.
- In the previous command, we already have a MongoDB connection available with us. So, we will use that object to access the MongoDB database.

The first parameter value is as follows:

- <database>

We will use the example as shown in the following figure:

```
db = MongoDBConnection.getDB("BPBOnlineBooksDB");
```

We can also run the following command:

```
db = new Mongo().getDB("BPBOnlineBooksDB");
```

Either way, it will give us the reference to the MongoDB database:

**Figure 8.6: The MongoDB Mongo.getDB() Method**

So far, we have learnt about `Mongo.getDB()` method. We will learn about some MongoDB database related methods in the next section.

## MongoDB database methods

We will cover some most common and important methods related to the MongoDB database which are helpful in connecting to the MongoDB server.

Some of these methods are as follows:

- `db.getMongo()`
- `db.hostInfo()`
- `db.stats()`
- `db.serverStatus()`

Let us study this one-by-one.

## db.getMongo()

Following are the details of this method, explained with some points:

- This method is used to test if the MongoDB Shell has a proper connection to the database instance.
- This method returns the database connection, as shown in the following figure:



```
Command Prompt - mongo
> db.getMongo();
connection to 127.0.0.1:27017
```

**Figure 8.7: The db.getMongo() method**

We can now see that after we run this method, it returns the database connection.

## db.hostInfo()

This method returns the information about the host, which means that it will return the information about system in which MongoDB is running, as shown in the following figure:



```
Command Prompt - mongo
> db.hostInfo();
{
  "system" : {
    "currentTime" : ISODate("2020-12-09T15:26:20.363Z"),
    "hostname" : "MSI",
    "cpuAddSize" : 64,
    "memSizeMB" : NumberLong(16216),
    "memLimitMB" : NumberLong(16216),
    "numCores" : 8,
    "cpuArch" : "x86_64",
    "numEnabled" : false
  },
  "os" : {
    "type" : "Windows",
    "name" : "Microsoft Windows 10",
    "version" : "10.0 (build 18363)"
  },
  "extra" : {
    "pageSize" : NumberLong(4096),
    "physicalCores" : 4
  },
  "ok" : 1
}
```

**Figure 8.8: The db.hostInfo() Method**

We can now see that after we run this method, it returns the system information.

## db.stats()

Following are the details of this method, explained with some points:

- This method returns the statistics of the single database, as shown in the following figure.
- It takes only 1 parameter.
- The parameter is optional which is a scale number in which we would like our output in terms of bytes or kilobytes. For example, if you would like the results to be displayed in kilobytes, you should pass 1024 in the parameter.



```
Command Prompt - mongo
> db.stats();
{
  "db" : "BPPOnlineBooksDB",
  "collections" : 0,
  "views" : 0,
  "objects" : 0,
  "avgObjSize" : 0,
  "dataSize" : 0,
  "storageSize" : 0,
  "totalSize" : 0,
  "indexes" : 0,
  "indexSize" : 0,
  "scaleFactor" : 1,
  "fileSize" : 0,
  "fslIndexSize" : 0,
  "fsTotalSize" : 0,
  "ok" : 1
}
>
```

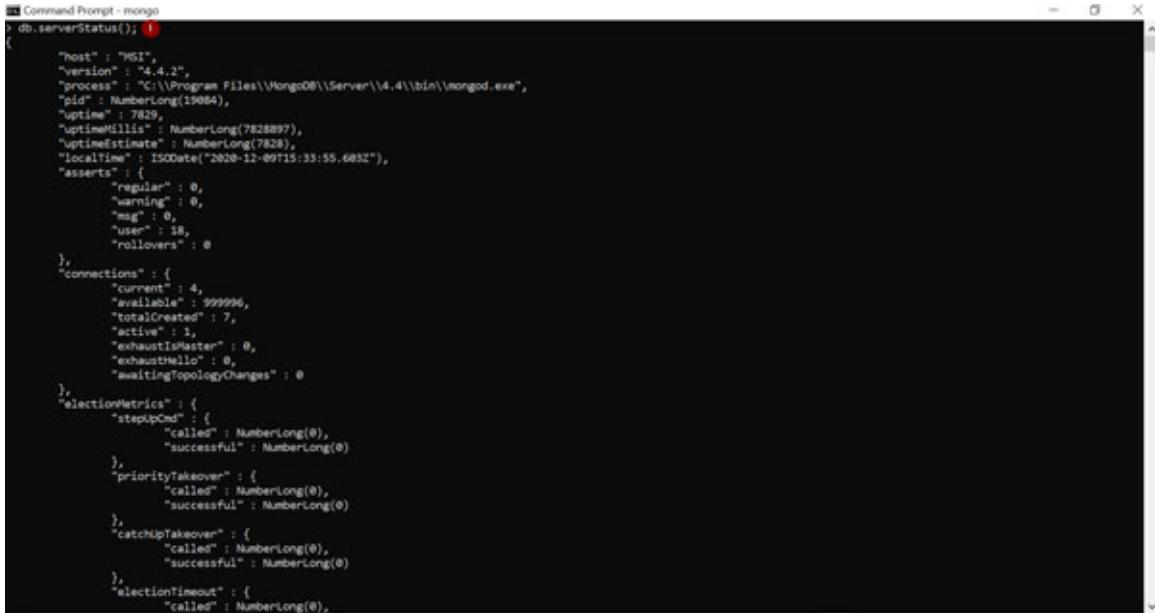
**Figure 8.9: The db.stats() Method**

We can now see that after we run this method, it returns the database statistics information.

## [db.serverStatus\(\)](#)

Following are the details of this method, explained with some points:

- This method returns the document which provides the complete overview of the database status and other process related information, as shown in the following figure.



```
Command Prompt - mongo
> db.serverStatus(); ⚡
{
  "Host" : "MSI",
  "version" : "4.4.2",
  "process" : "C:\Program Files\MongoDB\Server\4.4\bin\mongod.exe",
  "pid" : NumberLong(19684),
  "uptime" : 7829,
  "uptimeMillis" : NumberLong(7828897),
  "uptimeEstimate" : NumberLong(7828),
  "localtime" : ISODate("2020-12-09T15:33:55.683Z"),
  "asserts" : {
    "regular" : 0,
    "warning" : 0,
    "msg" : 0,
    "user" : 18,
    "rollovers" : 0
  },
  "connections" : {
    "current" : 4,
    "available" : 999996,
    "totalCreated" : 7,
    "active" : 1,
    "exhaustIsMaster" : 0,
    "exhaustHello" : 0,
    "waitingTopologyChanges" : 0
  },
  "electionMetrics" : {
    "stepUpCmd" : {
      "called" : NumberLong(0),
      "successful" : NumberLong(0)
    },
    "priorityTakeover" : {
      "called" : NumberLong(0),
      "successful" : NumberLong(0)
    },
    "catchUpTakeover" : {
      "called" : NumberLong(0),
      "successful" : NumberLong(0)
    },
    "electionTimeout" : {
      "called" : NumberLong(0),
      "successful" : NumberLong(0)
    }
  }
}
```

**Figure 8.10: The db.serverStatus() Method**

We can now see that after we run this method, it returns the document of database process related information.

## **MongoDB Collection methods**

We will cover some most common and important methods related to the collections which are helpful in performing the collections-related tasks in the MongoDB server.

Some of these methods are as follows:

- db.collection.count()
- db.collection.stats()
- db.collection.totalSize()
- db.collection.validate()
- db.collection.drop()

Let us study these one-by-one.

### **db.collection.count()**

Following are the details of this method, explained with some points:

- This method returns the count number of the documents in the collection, as shown in the following figure.
- It takes the query and options parameters, which we will study in the advanced chapters, where we will cover the CRUD operations.



A screenshot of the MongoDB Command Prompt window. The command entered is `> db.BPBOnlineBooksCollection.count();`. The output shows the result of the count method: `5`. A red box highlights this result, and a callout bubble says "Count() method has returned 5 Documents".

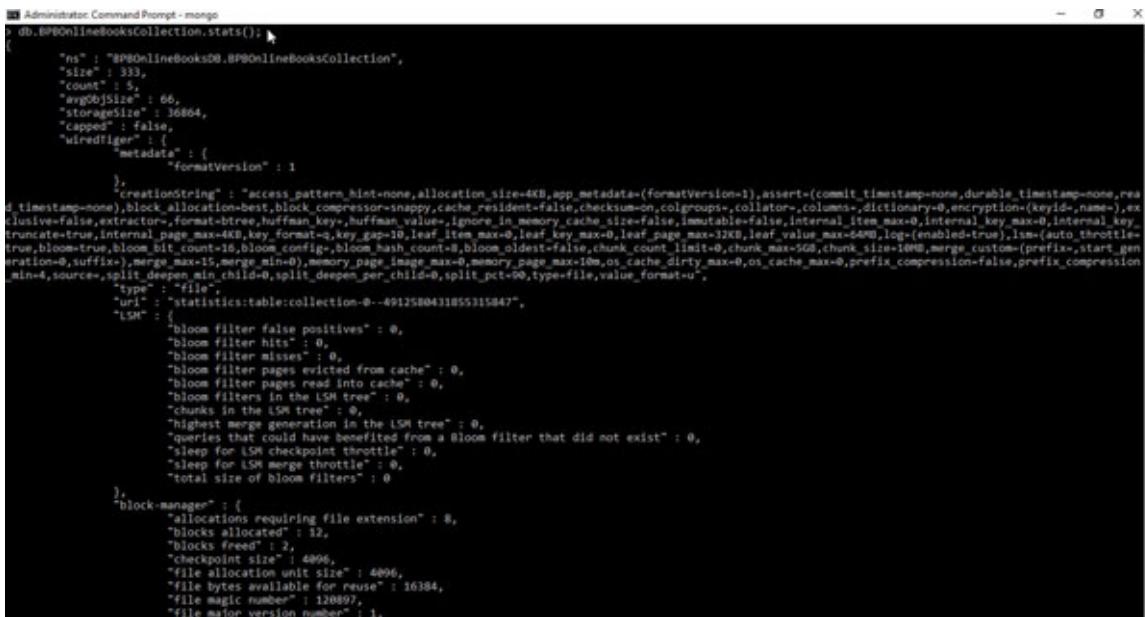
**Figure 8.11:** The MongoDB Collection `count()` Method

We can now see that after we run this method, it returns the total number of documents in the collection.

## [db.collection.stats\(\)](#)

Following are the details of this method, explained with some points:

- This method returns the statistics of the collection, as shown in the following figure.
- It takes only 1 parameter.
- The parameter is optional. It is a scale number in which we would like our



A screenshot of the MongoDB Command Prompt window. The command entered is `> db.BPBOnlineBooksCollection.stats();`. The output is a detailed JSON object containing various statistics about the collection, such as document count, storage size, and various counters related to the LSM tree and block manager.

```

{
  "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollection",
  "size": 333,
  "count": 5,
  "avgObjSize": 66,
  "storageSize": 36864,
  "capped": false,
  "wiredTiger": {
    "metadata": {
      "formatVersion": 1
    },
    "creationString": "access patterns:hint:none,allocation_size=4KB,app_metadata=(formatVersion=1),assert=(commit_timestamp:none,durable_timestamp:none,read_time_ms=0),block_allocation_best,block_compressor=nappy,cache_resident=false,checksum_on_colgroups,columns,dictionary=0,encryption=(keyId=,name=),exclusive=false,extractor=format-btree,huffman_key,huffman_value,ignore_in_memory_cache_size=false,immutable=false,internal_item_max=0,internal_key_max=0,internal_key_truncate=true,internal_page_max=4KB,key_format=<,key_gap=10,leaf_item_max=0,leaf_key_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,bloom=true,bloom_bit_count=16,bloom_config=bloom_hash_count=8,bloom_oldest=false,chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_custom=(prefix,start_gen=generation@,suffix@),merge_max=15,merge_min=0),memory_page_image_max=0,memory_page_max=10MB,os_cache_dirty_max=0,os_cache_max=0,prefix_compression=false,prefix_compression_min=4,source=_split,deepen_min_child=0,split_deepen_per_child=0,split_pct=90,type=file,value_format=u",
      "type": "file",
      "uri": "statistics:table:collection=0--4912580431855315847",
      "LSM": {
        "bloom_filter_false_positives": 0,
        "bloom_filter_hits": 0,
        "bloom_filter_misses": 0,
        "bloom_filter_pages_evicted_from_cache": 0,
        "bloom_filter_pages_read_into_cache": 0,
        "bloom_filters_in_the_LSM_tree": 0,
        "chunks_in_the_LSM_tree": 0,
        "highest_merge_generation_in_the_LSM_tree": 0,
        "queries_that_could_have_benefited_from_a_Bloom_filter_that_did_not_exist": 0,
        "sleep_for_LSM_checkpoint_throttle": 0,
        "sleep_for_LSM_merge_throttle": 0,
        "total_size_of_bloom_filters": 0
      }
    },
    "block_manager": {
      "allocations_requiring_file_extension": 8,
      "blocks_allocated": 132,
      "blocks_freed": 3,
      "checkpoint_size": 4896,
      "file_allocation_unit_size": 4896,
      "file_bytes_available_for_reuse": 16384,
      "file_magic_number": 128897,
      "file_main_version_number": 1
    }
  }
}

```

**Figure 8.12:** The MongoDB Collection `stats()` Method

We can now see that after we run this method, it returns the statistical data of the collection.

## [\*\*db.collection.totalSize\(\)\*\*](#)

Following are the details of this method, explained with some points:

This method returns the total number of size of data and size of indexes in the collection in bytes, as shown in the following figure:



```
Administrator: Command Prompt - mongo
> db.EPBOOnlineBooksCollection.totalSize();
73728
```

Total number of size of the Collection in Bytes

*Figure 8.13: The MongoDB Collection totalSize() Method*

We can now see that after we run this method, it returns us the total size of the collection in bytes.

## [\*\*db.collection.validate\(\)\*\*](#)

Following are the details of this method, explained with some points:

- This method scans the collection data and its indices for its correctness and returns the validated output, as shown in the following figure.
- It takes only 1 parameter.
- The parameter is optional which is a Boolean value. If this value is true, this method will do the full scan of the collection data and index. Else, if the value is omitted or is set to false, it will do the normal scan, which is faster than the full scan.
- Note that this method is resource intensive and may impact the performance of the MongoDB instance while the scanning is under progress.



```
Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollection.validate();
{
  "ns" : "BPBOnlineBooksDB.BPBOnlineBooksCollection",
  "nInvalidDocuments" : NumberLong(0),
  "nRecords" : 5,
  "nIndexes" : 1,
  "keysPerIndex" : {
    "_id" : 5
  },
  "indexDetails" : {
    "_id" : {
      "valid" : true
    }
  },
  "valid" : true,
  "warnings" : [],
  "errors" : [],
  "extraIndexEntries" : [],
  "missingIndexEntries" : [],
  "ok" : 1
}
>
```

**Figure 8.14:** The MongoDB Collection validate() Method

We can now see that after we run this method, it returns us the validated output of the collection data and index.

## [\*\*db.collection.drop\(\)\*\*](#)

Following are the details of this method, explained with some points:

- This method drops the collection and also removes any associated index with it, as shown in the following figure.
- This method will return true if the drop operation is successful.



```
Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollection.drop();
true
>
```

**Figure 8.15:** The MongoDB Collection drop() Method

We can now see that after we run this method, it returns `True`, which means that the drop operation was successful.

## [\*\*MongoDB cursor methods\*\*](#)

We will cover some common and important methods related to the MongoDB cursors. But before we do that, let us understand the concept of cursors in MongoDB.

## [\*\*What is a cursor in MongoDB?\*\*](#)

Whenever we use the `db.collection.find()` method in the MongoDB database to search the documents in a collection, it returns the pointer to these documents. This pointer, which is returned, is termed as

cursor. We will read more about cursor in the advanced chapters. In this chapter, we will just cover few cursor related methods.

Some of these methods are as follows:

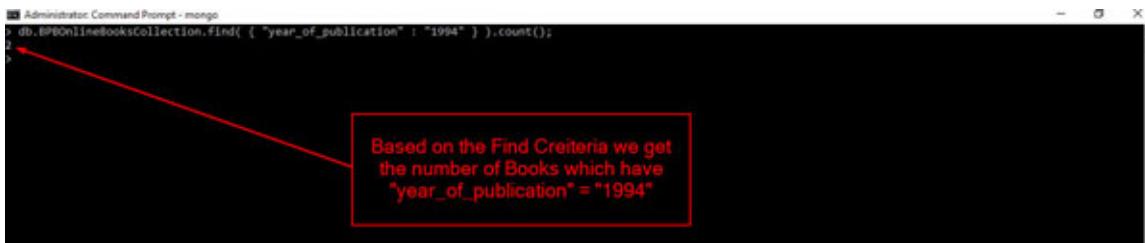
- `cursor.count()`
- `cursor.pretty()`
- `cursor.sort()`

Let us study them one-by-one.

## [cursor.count\(\)](#)

Following are the details of this method, explained with some points:

- This method is used to count the number of documents which are referenced by the cursor.
- In this method, we use `find()` to query and return the number of documents based on the criteria, as shown in the following figure:



A screenshot of the MongoDB shell interface. The command entered is `db.BooksCollection.find( { "year_of_publication": "1994" } ).count();`. The output returned is the number 2. A red arrow points from a callout box to the output line, with the text: "Based on the Find Criteria we get the number of Books which have "year\_of\_publication" = "1994"".

*Figure 8.16: The `cursor.count()` Method*

We can now see that after we run this method, it returns the count of number of documents based on the criteria given in the find method.

## [cursor.pretty\(\)](#)

This method is used to print results in easy-to-read format, as shown in the following figure:



A screenshot of the MongoDB shell interface. The command entered is `db.BooksCollection.find( { "year_of_publication": "1994" } ).pretty();`. The output is a pretty-printed JSON representation of two documents. Each document has fields: \_id, title, isbn, and year\_of\_publication, all set to specific values for books published in 1994.

```
{  
  "_id": ObjectId("5ea4a0246023fd968be0cb8512"),  
  "title": "Learn C",  
  "isbn": "12355433322344",  
  "year_of_publication": "1994"  
},  
{  
  "_id": ObjectId("5ea4a024f023fd968be0cb8513"),  
  "title": "Learn C++",  
  "isbn": "12355433322344",  
  "year_of_publication": "1994"  
}
```

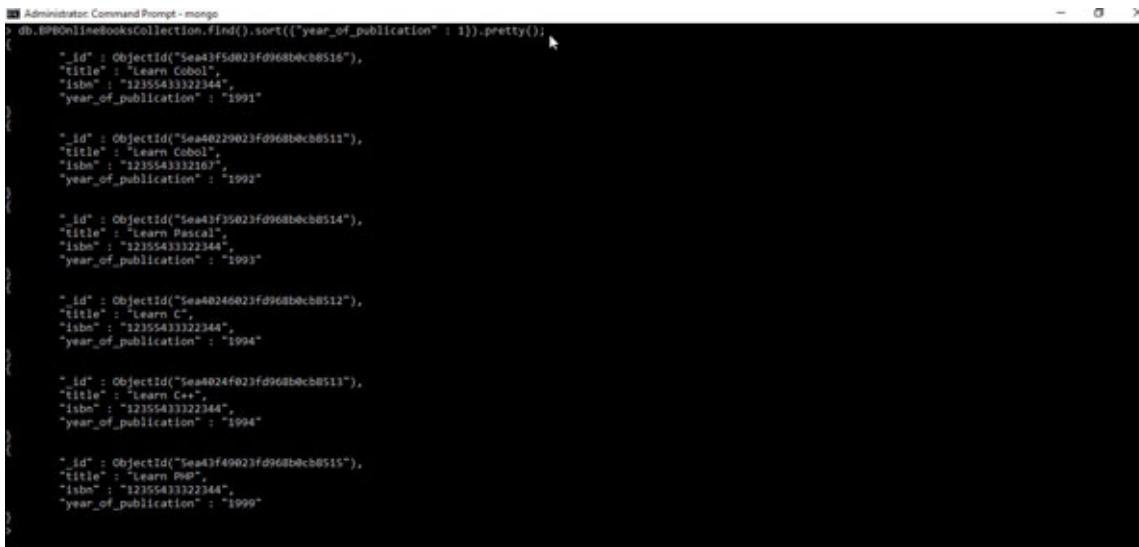
**Figure 8.17: The cursor.pretty() Method**

We can now see that after we run this method, it returns the results in easy-to-read format.

## cursor.sort()

Following are the details of this method, explained with some points:

- This method is used to sort the results either in ascending or in descending order, as shown in the following figure.
- If 1 is specified for a field, it means ascending order.
- If -1 is specified for a field, it means descending order.



```
Administrator Command Prompt - mongo
> db.EasyOnlineBooksCollection.find().sort({"year_of_publication": 1}).pretty();
{
  "_id" : ObjectId("5ea4f5d023fd968b0cb8516"),
  "title" : "Learn Cobol",
  "isbn" : "12355433322344",
  "year_of_publication" : "1991"
}

{
  "_id" : ObjectId("5ea4f229023fd968b0cb8511"),
  "title" : "Learn Cobol",
  "isbn" : "12355433322344",
  "year_of_publication" : "1992"
}

{
  "_id" : ObjectId("5ea4f35023fd968b0cb8514"),
  "title" : "Learn Pascal",
  "isbn" : "12355433322344",
  "year_of_publication" : "1993"
}

{
  "_id" : ObjectId("5ea4f246023fd968b0cb8512"),
  "title" : "Learn C",
  "isbn" : "12355433322344",
  "year_of_publication" : "1994"
}

{
  "_id" : ObjectId("5ea4f24f023fd968b0cb8511"),
  "title" : "Learn C++",
  "isbn" : "12355433322344",
  "year_of_publication" : "1994"
}

{
  "_id" : ObjectId("5ea4f49023fd968b0cb8515"),
  "title" : "Learn PHP",
  "isbn" : "12355433322344",
  "year_of_publication" : "1999"
}
```

**Figure 8.18: The cursor.sort() Method**

We can now see that after we run this method, it returns the results in ascending manner.

## Conclusion

In this chapter, we learned about the MongoDB Shell methods, use of JavaScript in MongoDB and that we can use other languages also that are officially supported by MongoDB.

We covered various Shell methods related to the MongoDB connections, MongoDB databases, and MongoDB collections. In the

last topic of this chapter, we also covered the few methods related to cursors.

These methods are very useful and we can use them in various scenarios while working with MongoDB and later when we will learn some advanced topics and application development using MongoDB.

In the next chapter, we will study the data types in MongoDB where we will cover various data types available in MongoDB with examples.

## **Questions**

1. What is the default Shell language used in MongoDB?
2. Does MongoDB support different languages other than JavaScript?
3. Can you name few languages supported by MongoDB?
4. Can you explain the MongoDB `connect()` method and the ways to connect to the MongoDB server, as explained in the connection methods topic?
5. What is the method used to calculate the size of the collection?
6. What is a cursor in MongoDB?

# CHAPTER 9

## Data Types in MongoDB

This chapter covers the data types used in MongoDB. We will start with the introduction of data type - what exactly it is, and then we will move on to the overview of the BSON data types. There are different data types used in MongoDB and each one of them have different properties and structure and are used in different scenarios. Some of them are widely used and some of them are not very frequently used.

### Structure

In this chapter, we will discuss the following topics:

- What are data types?
- Introduction to BSON data types
- Integer
- Double
- String
- Object
- Array
- Binary data
- Object Id
- Boolean
- Date
- Null
- Regular expression
- JavaScript
- JavaScript with scope

- Timestamp
- Min key
- Max key
- Decimal128
- Comparison and Sort Order

## Objectives

After studying this unit, you should be able to learn about the data types and BSON data types. Later, in this chapter, you will learn about the data types supported by MongoDB and understand those using practical methods.

## What are data types?

Data types are the types of data used to store data in various formats that is understood by the programming language, or, in our case, the database. Normally, the data types can be represented by `string type`, `integer type`, `float type`, etc. The data types give the meaning to the data as well as types of operations that can be performed with these data types.

## Introduction to BSON data types

The BSON data types are just like the JSON data types, but in the binary format. That's why, they are called Binary JSON. BSON is the binary format and can have simple to complex data types, including the name-value pairs, such as associative arrays.

MongoDB stores the data types in a unique manner. It has an alias as well as associated number for identification.

## Data types in MongoDB

Following is the list of data types available in MongoDB which includes data type, data type number, and data type alias:

Data Type	Data Type Number	Data Type Alias

Double	1	"double"
String	2	"string"
Object	3	"object"
Array	4	"array"
Binary data	5	"binData"
ObjectId	7	"objectId"
Boolean	8	"bool"
Date	9	"date"
Null	10	"null"
Regular Expression	11	"regex"
JavaScript	13	"javascript"
JavaScript (with scope)	15	"javascriptWithScope"
32-bit integer	16	"int"
Timestamp	17	"timestamp"
64-bit integer	18	"long"
Decimal128	19	"decimal"
Min key	-1	"minKey"
Max key	127	"maxKey"

**Table 9.1: MongoDB Data Types**

Let us now study the different data types one-by-one. We will cover all the different data types used in MongoDB, along with their practical examples using the MongoDB Shell.

## **Integer data types**

Integer data types are used to store numeric values. MongoDB supports 32-bit or 64-bit integers which depend on the architecture of the machine on which MongoDB is running. 64-bit integers are also

called as `long int` and have alias value as `long` and number value as 18 in the MongoDB data types.

In our example, we have inserted the integer data type value in our MongoDB collection and the code for the same is shown in the following screenshot:

## Code 1

```
db.BPBOnlineBooksDataTypesCollection.insert({  
    "Data Type Alias": "int",  
    "Data Type Number": "16",  
    "Data Type Value": 7777  
});
```

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.find({  
    "Data Type Alias": "int"  
}).pretty();
```

The screenshot shows a terminal window titled 'Administrator Command Prompt - mongo'. It displays the following MongoDB shell session:

```
> db.BPBOnlineBooksDataTypesCollection.insert({  
...     "Data Type Alias": "int",  
...     "Data Type Number": "16",  
...     "Data Type Value": 7777  
... });  
WriteResult({ "nInserted" : 1 })  
> db.BPBOnlineBooksDataTypesCollection.find({  
...     "Data Type Alias": "int"  
... }).pretty();  
{  
    "_id": ObjectId("5e8ade4c000f9357bcd469eb"),  
    "Data Type Alias": "int",  
    "Data Type Number": "16",  
    "Data Type Value": 7777  
}  
>
```

A red box highlights the line 'Data Type Value': 7777, and a red callout bubble points to it with the text 'Data Type Value is Integer'.

**Figure 9.1: Creating a MongoDB Document with Integer Data Type**

In the preceding example, we saw how to create a new document using data type having integer type value.

## String data types

String data types are one of the most common data types used in MongoDB. You might be aware that many of the international languages use some special characters. In order to save these characters, the MongoDB drivers convert these characters to UTF-8 format during the serialization and de-serialization process, thus, making it possible to store most of these international characters in

the BSON strings and validating the international characters to be stored in the database.

In our example, we have inserted the string data type value in our MongoDB collection and the code for the same is shown in the following screenshot:

## Code 1

```
db.BPBOnlineBooksDataTypesCollection.insert({  
    "Data Type Alias": "string",  
    "Data Type Number": "2",  
    "Data Type Value": "BPB Publications – The Largest Online  
    Resource for IT Books"  
});
```

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.find({  
    "Data Type Alias": "string"  
}).pretty();
```

The screenshot shows a terminal window titled 'Administrator Command Prompt - mongo'. It displays two commands: an insert operation and a find operation. The insert operation creates a document with fields 'Data Type Alias' (value 'string'), 'Data Type Number' (value '2'), and 'Data Type Value' (value 'BPB Publications – The Largest Online Resource for IT Books'). The find operation retrieves all documents where 'Data Type Alias' is 'string'. A red box highlights the 'Data Type Value' field in the inserted document. A red arrow points from this highlighted field to a callout box containing the text 'Data Type Value is String'.

**Figure 9.2: Creating a MongoDB Document with String Data Type**

In the preceding example, we saw how to create a new document using data type having string type value.

## Double data types

Double data types are used to store the floating point values. Floating point values have decimal points, which are somewhat similar to integer values but they have decimal values with them. For example, 777.27 or 12.10 are decimal type values and they are called double types in MongoDB.

In our example, we have inserted the double data type value in our MongoDB collection and the code for the same is shown in the following screenshot:

## Code 1

```
db.BPBOnlineBooksDataTypesCollection.insert({  
    "Data Type Alias": "double",  
    "Data Type Number": "1",  
    "Data Type Value": 777.27  
});
```

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.find({  
    "Data Type Alias": "double"  
}).pretty();
```

The screenshot shows the MongoDB shell interface. A red box labeled '1' highlights the command to insert a new document. Another red box labeled '2' highlights the command to find the inserted document and its pretty-printed output. A red callout box points to the 'Data Type Value' field in the result, which contains the value '777.27'. The output also includes the '\_id' field.

```
db.BPBOnlineBooksDataTypesCollection.insert({  
    "Data Type Alias": "double",  
    "Data Type Number": "1",  
    "Data Type Value": 777.27  
});  
writeResult({ "nInserted": 1 })  
db.BPBOnlineBooksDataTypesCollection.find({  
    "Data Type Alias": "double"  
}).pretty();  
{  
    "_id": ObjectId("5ebe12100bf9357bcd469ed"),  
    "Data Type Alias": "double",  
    "Data Type Number": "1",  
    "Data Type Value": 777.27  
}
```

**Figure 9.3: Creating a MongoDB Document with Double Data Type**

In the preceding example, we saw how to create a new document using data type having double type value.

## Array data types

The array data types are used to store the group of similar data type values which are linked or associated in the form of key and value.

In our example, we have defined 2 arrays and then we will insert these arrays as a value and the code for the same is as follows:

## Code 1

```
var BPBBookStoresinIndia = ['New Delhi', 'Mumbai', 'Kolkata',  
    'Chennai'];
```

```
var BPBBookStoresinUSA = ['New York', 'Atlanta', 'Arizona',
'New Jersey'];
```

And then, we have inserted these arrays in our MongoDB collection and below are the code for the same, are shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.insert({
  "Data Type Alias": "array",
  "Data Type Number": "4",
  "Data Type Value 1": BPBBookStoresinIndia,
  "Data Type Value 2": BPBBookStoresinUSA
});
```

## Code 3

```
db.BPBOnlineBooksDataTypesCollection.find({
  "Data Type Alias": "array"
}).pretty();
```

The screenshot shows a MongoDB shell window with the following session:

```
Administrator Command Prompt - mongo
> var BPBBookStoresinIndia = ['New Delhi', 'Mumbai', 'Kolkata', 'Chennai'];
> var BPBBookStoresinUSA = ['New York', 'Atlanta', 'Arizona', 'New Jersey'];
> db.BPBOnlineBooksDataTypesCollection.insert({
...   "Data Type Alias": "array",
...   "Data Type Number": "4",
...   "Data Type Value 1": BPBBookStoresinIndia,
...   "Data Type Value 2": BPBBookStoresinUSA
... });
writeResult({ "nInserted": 1 })
> db.BPBOnlineBooksDataTypesCollection.find({
...   "Data Type Alias": "array"
... }).pretty();
{
  "_id": ObjectId("5ebe27c080f9357bcd460ee"),
  "Data Type Alias": "array",
  "Data Type Number": "4",
  "Data Type Value 1": [
    "New Delhi",
    "Mumbai",
    "Kolkata",
    "Chennai"
  ],
  "Data Type Value 2": [
    "New York",
    "Atlanta",
    "Arizona",
    "New Jersey"
  ]
}
```

A red box highlights the value of "Data Type Value 1" with the annotation "Data Type Value is an Array". Three numbered circles point to the following lines of code:

1. `BPBBookStoresinIndia = ['New Delhi', 'Mumbai', 'Kolkata', 'Chennai'];`
2. `BPBBookStoresinUSA = ['New York', 'Atlanta', 'Arizona', 'New Jersey'];`
3. `db.BPBOnlineBooksDataTypesCollection.insert({ ... })`

**Figure 9.4: Creating a MongoDB Document with Array Data Type**

In the preceding example, we saw how to create a new document using data type having array type value.

## Object data types

The object data types are used to store the embedded documents or the JSON data type which is a kind of JSON object or document and are sometimes called embedded documents.

In our example, we have created an object as follows:

## Code 1

```
var BPBBooksLatestEditions = [ {  
    'Title': 'Instant Approach to Software Testing: Principles,  
    Applications, Techniques, and Practices',  
    'Year': '2019',  
    'ISBN': '9789388511162',  
    'Pages': 368,  
    'Weight': '677gm',  
    'Dimension': '24x18x2cm'  
}, {  
    'Title': 'IOT and Smart Cities: Your Smart City Planning  
    Guide',  
    'Year': '2019',  
    'ISBN': '9789388511322',  
    'Pages': 242,  
    'Weight': '357gm',  
    'Dimension': '22.5x15x1.5gm'  
} ];
```

And then, we have inserted this object type in our MongoDB collection and the code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.insert({  
    "Data Type Alias": "object",  
    "Data Type Number": "3",  
    "Data Type Value": BPBBooksLatestEditions  
});
```

## Code 3

```
db.BPBOnlineBooksDataTypesCollection.find({
  "Data Type Alias": "object"
}).pretty();
```

The screenshot shows a MongoDB shell window with the following content:

```
Administrator: Command Prompt - mongo
> var BPBBooksLatestEditions = [
...   {
...     "title": "Instant Approach to Software Testing: Principles, Applications, Techniques, and Practices",
...     "Year": "2019",
...     "ISBN": "9789388511162",
...     "Pages": 368,
...     "Weight": "677gm",
...     "Dimension": "24x18x2cm"
...   },
...   {
...     "title": "IoT and Smart Cities: Your Smart City Planning Guide",
...     "Year": "2019",
...     "ISBN": "9789388511322",
...     "Pages": 242,
...     "Weight": "357gm",
...     "Dimension": "22.5x15x1.5cm"
... };
> db.BPBOnlineBooksDataTypesCollection.insert({
...   "Data Type Alias": "object",
...   "Data Type Number": "3",
...   "Data Type Value": BPBBooksLatestEditions
... });
> writeResult({ "nInserted": 1 })
> db.BPBOnlineBooksDataTypesCollection.find({
...   "Data Type Alias": "object"
... }).pretty();
{
  "_id": ObjectId("5ebe36a000f9357bcd460ef"),
  "Data Type Alias": "object",
  "Data Type Number": "3",
  "Data Type Value": [
    {
      "Title": "Instant Approach to Software Testing: Principles, Applications, Techniques, and Practices",
      "Year": "2019",
      "ISBN": "9789388511162",
      "Pages": 368,
      "Weight": "677gm",
      "Dimension": "24x18x2cm"
    },
    {
      "Title": "IoT and Smart Cities: Your Smart City Planning Guide", ←
      "Year": "2019",
      "ISBN": "9789388511322",
      "Pages": 242,
      "Weight": "357gm",
      "Dimension": "22.5x15x1.5cm"
    }
  ]
}
```

A red box highlights the "Data Type Value" field, and a red arrow points from it to a callout box containing the text "Data Type Value is an Object".

**Figure 9.5: Creating a MongoDB Document with Object Data Type**

In the preceding example, we saw how to create a new document using data type having array type value.

## Binary data types

The binary data types are used to store the values in binary form such as Base64. Sometimes, there are scenarios where we store some objects like images (in a binary form) in database.

In our example, we have created a variable which contains a binary data as follows:

### Code 1

```
var BPBBooksBinaryData = BinData(1,
"SGVsbtG8gV29ybGQgRnJvbSBCUEIgUHVibGljYXRpb25z");
```

And then, we have inserted this binary data type in our MongoDB collection and the code for the same is shown in the following

screenshot:

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.insert({  
    "Data Type Alias": "binData",  
    "Data Type Number": "5",  
    "Data Type Value": BPBBooksBinaryData  
});
```

## Code 3

```
db.BPBOnlineBooksDataTypesCollection.find({  
    "Data Type Alias": "binData"  
}).pretty();
```

The screenshot shows a terminal window titled 'Administrator Command Prompt - mongo'. The user has run the following commands:

```
> var BPBBooksBinaryData = BinData(1, "SGVsbG8gV29ybGQgRnJvbSBzUEIgUHv1bG1jYKRpZz");  
> db.BPBOnlineBooksDataTypesCollection.insert({  
...     "Data Type Alias": "binData",  
...     "Data Type Number": "5",  
...     "Data Type Value": BPBBooksBinaryData  
... });  
> writeResult({ "nInserted": 1 })  
> db.BPBOnlineBooksDataTypesCollection.find({  
...     "Data Type Alias": "binData"  
... }).pretty();
```

Three numbers (1, 2, 3) are circled in red above the command line, corresponding to numbered annotations in the image:

- Annotation 1: Points to the variable assignment line.
- Annotation 2: Points to the 'Data Type Value' field in the insert command.
- Annotation 3: Points to the 'Data Type Value' field in the find command.

A red callout box on the right side of the terminal window contains the text 'Data Type Value is Binary Data'.

**Figure 9.6:** Creating a MongoDB Document with Binary Data Type

In the preceding example, we saw how to create a new document using data type having the binary type value.

## ObjectId data types

This is a special MongoDB specific data type which stores the unique key ID. In MongoDB, every document in a collection has unique `ObjectId` and every document has the `_id` field.

The size of the `ObjectId` is 12 bytes and it is divided into 4 parts, as shown in the following table:

Part name	Size(bytes)
Timestamp	4
Machine Id	3

Process Id	2
Counter	3

**Table 9.2: Size Division of ObjectId**

In our example, we have created a variable which contains an `ObjectId` value.

## Code 1

```
var BPBBooksObjectId = ObjectId();
```

And then, we have inserted this `ObjectId` data type in our MongoDB collection and the code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.insert({
  "Data Type Alias": "objectid",
  "Data Type Number": "7",
  "Data Type Value": BPBBooksObjectId
});
```

## Code 3

```
db.BPBOnlineBooksDataTypesCollection.find({
  "Data Type Alias": "objectid"
}).pretty();
```

```
Administrator Command Prompt - mongo
1> var BPBBooksObjectId = ObjectId();
2> db.BPBOnlineBooksDataTypesCollection.insert({
3>   ...
4>   "Data Type Alias": "objectid",
5>   "Data Type Number": "7",
6>   "Data Type Value": BPBBooksObjectId
7> });
8> writeResult({ "nInserted": 1 })
9> db.BPBOnlineBooksDataTypesCollection.find({
10>   ...
11>   "Data Type Alias": "objectid"
12> }).pretty();
{
  "_id": ObjectId("Sebae70000f9357bcd460f2"),
  "Data Type Alias": "objectid",
  "Data Type Number": "7",
  "Data Type Value": ObjectId("Sebae70000f9357bcd460f1")
```

Data Type Value is ObjectId

**Figure 9.7: Creating a MongoDB Document with ObjectId Data Type**

In the preceding example, we saw how to create a new document using the data type having the `ObjectId` type value.

## Date data types

Date data type stores the date and time. These can be date or date and time combined. There are various methods in MongoDB to generate date and time, as shown in the following table:

Date method	Description
Date()	This method returns the current date in the string format.
New Date()	This method returns a date object. This method uses the <code>ISODate()</code> wrapper.

**Table 9.3: Date Methods**

In our example, we have created 2 variables which contain a string and a date object respectively.

### **Code 1**

```
var BPBBooksDate1 = Date();  
var BPBBooksDate2 = new Date();
```

And then, we have inserted these date data type variables in our MongoDB collection and the code for the same is shown in the following screenshot:

### **Code 2**

```
db.BPBOnlineBooksDataTypesCollection.insert({  
    "Data Type Alias": "date",  
    "Data Type Number": "9",  
    "Data Type Value 1": BPBBooksDate1,  
    "Date Type Value 2": BPBBooksDate2  
});
```

### **Code 3**

```
db.BPBOnlineBooksDataTypesCollection.find({  
    "Data Type Alias": "date"  
}).pretty();
```

```

Administrator Command Prompt - mongo
> var BPBBooksDate1 = new Date();
> var BPBBooksDate2 = new Date();
> db.BPBOnlineBooksDataTypesCollection.insert({ ②
...   "Data Type Alias": "date",
...   "Data Type Number": "9",
...   "Data Type Value 1": BPBBooksDate1,
...   "Data Type Value 2": BPBBooksDate2
... });
> writeResult({ "nInserted": 1 })
> db.BPBOnlineBooksDataTypesCollection.find({ ③
...   "Data Type Alias": "date"
... }).pretty();
{
  "_id": ObjectId("Sebae023000f9357bcd460f3"),
  "Data Type Alias": "date",
  "Data Type Number": "9",
  "Data Type Value 1": "Tue May 12 2020 23:46:50 GMT+0530 (India Standard Time)",
  "Data Type Value 2": "ISODate(\"2020-05-12T18:17:00.443Z\")"
}

```

Data Type Value is Date

**Figure 9.8: Creating a MongoDB Document with Date Data Type**

In the preceding example, we saw how to create a new document using data type having date type value.

## Null data types

Null data type stores the null values. Null are the data types which has no type or value, so it is called as null. Sometimes, there are scenarios where we have to store something which exists but is yet to be defined and we are not sure what type and value it would have. In this case, we use null.

In our example, we have created a variable which contains a null type value.

### Code 1

```
var BPBBooksNull = null;
```

And then, we have inserted this null data type variable in our MongoDB collection and the code for the same is shown in the following screenshot:

### Code 2

```
db.BPBOnlineBooksDataTypesCollection.insert({
  "Data Type Alias": "null",
  "Data Type Number": "10",
  "Data Type Value": BPBBooksNull
});
```

### Code 3

```
db.BPBOnlineBooksDataTypesCollection.find({
  "Data Type Alias": "null"
}) .pretty();
```

```
Administrator: Command Prompt - mongo
> var BPBBookNull = null;
> db.BPBOnlineBooksDataTypesCollection.insert({ ②
...   "Data Type Alias": "null",
...   "Data Type Number": "10",
...   "Data Type Value": BPBBookNull
... });
writeResult({ "nInserted" : 1 })
> db.BPBOnlineBooksDataTypesCollection.find({ ③
...   "Data Type Alias": "null"
... }).pretty();
{
  "_id" : ObjectId("5ebae8d1e09ff9357bcd460f4"),
  "Data Type Alias" : "null",
  "Data Type Number" : "10",
  "Data Type Value" : null
}
```

**Figure 9.9:** Creating a MongoDB Document with Null Data Type

In the preceding example, we saw how to create a new document using data type having null type value.

## Regular expression data types

Regular expression data type stores the regular expression values. Regular expression, or regex, is the sequence of characters which defines the search patterns and is used for find, replace, and validation operations.

In our example, we have created a variable which contains a regular expression value.

### **Code 1**

```
var BPBBooksRegEx = RegExp ("%BPB");
```

And then, we have inserted these regular expression data type variables in our MongoDB collection and the code for the same is shown in the following screenshot:

### **Code 2**

```
db.BPBOnlineBooksDataTypesCollection.insert({
  "Data Type Alias": "regex",
  "Data Type Number": "11",
  "Data Type Value": BPBBooksRegEx
});
```

## Code 3

```
db.BPBOnlineBooksDataTypesCollection.find({  
    "Data Type Alias": "regex"  
}).pretty();
```

The screenshot shows a Windows Command Prompt window titled 'Administrator: Command Prompt - mongo'. The command entered is:

```
> var BPBBooksRegEx = RegExp('1010');  
> db.BPBOnlineBooksDataTypesCollection.insert({  
    ...  
    "Data Type Alias": "regex",  
    ...  
    "Data Type Number": "11",  
    ...  
    "Data Type Value": BPBBooksRegEx  
    ...});  
> writeResult({ "nInserted": 1 });  
> db.BPBOnlineBooksDataTypesCollection.find({  
    ...  
    "Data Type Alias": "regex"  
    ... }).pretty();
```

A red box highlights the line 'Data Type Value': BPBBooksRegEx. A red arrow points from this box to a callout box containing the text 'Data Type Value is Regular Expression'.

**Figure 9.10:** Creating a MongoDB Document with Regular Expression Data Type

In the preceding example, we saw how to create a new document using data type having regular expression type value.

## JavaScript data types (without scope)

We can store JavaScript functions in MongoDB documents and use these functions in our scenarios. In our example, we have created 2 variables where one contains a function definition and the other is a scope variable, but it will be empty and thus, it will be without scope.

## Code 1

```
var BPBBooksFunction = "function() {var bpb; bpb=100;}";  
var BPBBooksFunctionScope = {};
```

And then, we have inserted these variables in our MongoDB collection and the code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.insert({  
    "Data Type Alias": "javascript",  
    "Data Type Number": "13",  
    "Data Type Value Function": BPBBooksFunction,  
    "Data Type Value Function Scope": BPBBooksFunctionScope
```

```
});
```

### Code 3

```
db.BPBOnlineBooksDataTypesCollection.find({  
    "Data Type Alias": "javascript"  
}).pretty();
```

The screenshot shows a terminal window titled "Administrator Command Prompt - mongo". Inside, a script is run to insert a document into the "BPBOnlineBooksDataTypesCollection". The document has fields: "Data Type Alias" (set to "javascript"), "Data Type Number" (set to "13"), "Data Type Value Function" (set to "BPBBooksFunction"), and "Data Type Value Function Scope" (set to "BPBBooksFunctionScope"). After insertion, a find operation is performed with the same filter, returning the inserted document. A red box highlights the "Data Type Value Function" field's value, which is "function(){var bp; bp=100;}", with a callout "Data Type Value is JavaScript".

```
var BPBBooksFunction = "function(){var bp; bp=100;}"  
var BPBBooksFunctionScope = []  
db.BPBOnlineBooksDataTypesCollection.insert({  
    ...  
    "Data Type Alias": "javascript",  
    ...  
    "Data Type Number": "13",  
    ...  
    "Data Type Value Function": BPBBooksFunction,  
    ...  
    "Data Type Value Function Scope": BPBBooksFunctionScope  
    ...});  
writeResult({ "nInserted": 1 })  
> db.BPBOnlineBooksDataTypesCollection.find({  
    ...  
    "Data Type Alias": "javascript"  
    ... }).pretty();  
{  
    "_id": ObjectId("5ebeaea64aa80f9357bcd46af6"),  
    "Data Type Alias": "javascript",  
    "Data Type Number": "13",  
    "Data Type Value Function": "function(){var bp; bp=100;}",  
    "Data Type Value Function Scope": {  
        ...  
    }  
}
```

*Figure 9.11: Creating a MongoDB Document with JavaScript Data Type*

In the preceding example, we saw how to create a new document using data type having JavaScript type value.

### Javascript data types (with scope)

In our example, we have created two variables where one contains a function definition and the other one is a scope variable, and now we will define it.

### Code 1

```
var BPBBooksFunction = "function() {var bp; bp=2000;}";  
var BPBBooksFunctionScope = ["object"];
```

And then, we have inserted these variables in our MongoDB collection and the code for the same is shown in the following screenshot:

### Code 2

```
db.BPBOnlineBooksDataTypesCollection.insert({  
    "Data Type Alias": "javascriptWithScope",  
    "Data Type Number": "15",  
    ...});
```

```

    "Data Type Value Function": BPBBooksFunction,
    "Data Type Value Function Scope": BPBBooksFunctionScope
} );

```

## Code 3

```

db.BPBOnlineBooksDataTypesCollection.find({
    "Data Type Alias": "javascriptWithScope"
}).pretty();

```

The screenshot shows the MongoDB shell interface. A command is being run to insert a document into the `BPBOnlineBooksDataTypesCollection`. The document has the following structure:

```

{
  "Data Type Function": "function()(var bpb; bpb+2000);",
  "Data Type Scope": ["object"],
  "Data Type Alias": "javascriptWithScope",
  "Data Type Number": "15",
  "Data Type Value Function": BPBBooksFunction,
  "Data Type Value Function Scope": BPBBooksFunctionScope
}

```

Three specific parts of the command are highlighted with red circles and numbered 1, 2, and 3:

- ① `Data Type Function`: `"function()(var bpb; bpb+2000);"`
- ② `Data Type Scope`: `["object"]`
- ③ `Data Type Alias`: `"javascriptWithScope"`

A red box highlights the value of `Data Type Value Function`, which is `"function()(var bpb; bpb+2000);"`. An arrow points from this box to the text **Data Type Value is JavaScript (With Scope)**.

**Figure 9.12: Creating a MongoDB Document with JavaScript (With Scope) Data Type**

In the preceding example, we saw how to create a new document using data type having JavaScript (with scope) type value.

## Timestamp data types

Timestamp is the current time of event which is recorded by the computer and it is accurate to milliseconds. We can store timestamps in the MongoDB documents.

In our example, we have created a variable which contains a timestamp.

## Code 1

```
var BPBBooksTimestamp = new Timestamp();
```

And then, we have inserted this variable in our MongoDB collection and the code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.insert({
    "Data Type Alias": "timestamp",
    "Data Type Number": "17",
    "Data Type Value": BPBBooksTimestamp
});
```

## Code 3

```
db.BPBOnlineBooksDataTypesCollection.find({
    "Data Type Alias": "timestamp"
}).pretty();
```

The screenshot shows a terminal window titled "Administrator Command Prompt - mongo". Inside, a JavaScript code block is being run to insert a document into the "BPBOnlineBooksDataTypesCollection". The code uses a timestamp variable and includes a find operation to verify the insertion. A red box highlights the timestamp value in the inserted document, which is then annotated with a callout "Data Type Value is Timestamp".

```
var BPBBooksTimestamp = new Timestamp();
db.BPBOnlineBooksDataTypesCollection.insert({
...   "Data type Alias": "timestamp",
...   "Data type Number": "17",
...   "Data type Value": BPBBooksTimestamp
... });
writeResult({ "nInserted": 1 })
db.BPBOnlineBooksDataTypesCollection.find({
...   "Data type Alias": "timestamp"
... }).pretty();
{
  "_id": ObjectId("5e8aecd600bf9357bcd460fc"),
  "Data type Alias": "timestamp",
  "Data type Number": "17",
  "Data type Value": timestamp(1589300000, 1) ← Data Type Value is Timestamp
}
```

*Figure 9.13: Creating a MongoDB Document with Timestamp Data Type*

In the preceding example, we saw how to create a new document using data type having timestamp type value.

## Boolean data types

Boolean data type stores the boolean data which is either true or false. In our example, we have created two variables which contains a "true value" and a "false value" respectively.

## Code 1

```
var BPBBooksBoolean1 = true;
var BPBBooksBoolean2 = false;
```

And then, we have inserted these variables in our MongoDB collection and the code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksDataTypesCollection.insert({
```

```

    "Data Type Alias": "bool",
    "Data Type Number": "8",
    "Data Type Value 1": BPBBooksBoolean1,
    "Data Type Value 2": BPBBooksBoolean2
  ) );

```

## Code 3

```

db.BPBOnlineBooksDataTypesCollection.find({
  "Data Type Alias": "bool"
}).pretty();

```

The screenshot shows a MongoDB command prompt window. The user has run the following code:

```

var BPBBooksBoolean1 = true;
var BPBBooksBoolean2 = False; ①
db.BPBOnlineBooksDataTypesCollection.insert({ ②
  ...
  "Data Type Alias": "bool",
  ...
  "Data Type Number": "8",
  ...
  "Data Type Value 1": BPBBooksBoolean1,
  ...
  "Data Type Value 2": BPBBooksBoolean2
  ...
});
writeResult({ "nInserted": 1 });
db.BPBOnlineBooksDataTypesCollection.find({ ③
  ...
  "Data Type Alias": "bool"
  ...
}).pretty();

```

A red box highlights the line `BPBBooksBoolean1 = true;` with the annotation **1**. Another red box highlights the line `db.BPBOnlineBooksDataTypesCollection.insert({` with the annotation **2**. A third red box highlights the line `db.BPBOnlineBooksDataTypesCollection.find({` with the annotation **3**. A red arrow points from the text "Data Type Value is Boolean" to the value `true` in the inserted document.

**Figure 9.14: Creating a MongoDB Document with Boolean Data Type**

In the preceding example, we saw how to create a new document using data type having Boolean type value.

## Min and max key

In MongoDB, the min and max keys compare a value against the lowest and the highest BSON elements. The min key compares the values of the lowest BSON element, whereas the max key compares the values of the highest BSON element.

The Min and max keys are both internal data types in MongoDB.

## Decimal128

The Decimal128 data type has been introduced in the MongoDB version 3.4. There are some scenarios where the double data type has limited capacity to store the decimal values which are either very large or very small, and if we store the large decimal value, then the precision will be lost. In this case, MongoDB provides the

Decimal128 data type so that we can store the decimal values with a lot more precision.

Decimal128 allows us to store the values up to 34 decimal digits of precision, which means we can store maximum and minimum values in the order of  $10^{6144}$  and  $10^{-6143}$ , respectively. This is a lot of precision in terms of decimal values and this also prevents any rounding off of the values for the data which sometimes requires a lot of precision like in mathematical or scientific calculations.

## Comparison and sort order

While comparing the values of different BSON types, MongoDB uses the following comparison order from the lowest to the highest:

- MinKey
- Null
- Numbers (ints, longs, doubles)
- Symbol, String
- Object
- Array
- BinData
- ObjectId
- Boolean
- Date
- Timestamp
- Regular expression
- MaxKey

In our example, we have created few variables which contain different data types values.

### **Code 1**

```
var BPBBooksVar1 = 100000;  
var BPBBooksVar2 = "BPB Publications";  
var BPBBooksVar3 = 77.07;
```

```
var BPBBooksVar4 = true;
var BPBBooksVar5 = null;
var BPBBooksVar6 = MinKey;
var BPBBooksVar7 = MaxKey;
```

And then, we have inserted these variables in our MongoDB collection and the code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksDataTypesCollectionV2.insert([
    {
        "Data Type Alias": "int",
        "Data Type Value": BPBBooksVar1
    },
    {
        "Data Type Alias": "string",
        "Data Type Value": BPBBooksVar2
    },
    {
        "Data Type Alias": "double",
        "Data Type Value": BPBBooksVar3
    },
    {
        "Data Type Alias": "bool",
        "Data Type Value": BPBBooksVar4
    },
    {
        "Data Type Alias": "null",
        "Data Type Value": BPBBooksVar5
    },
    {
        "Data Type Alias": "minKey",
        "Data Type Value": BPBBooksVar6
    },
    {
        "Data Type Alias": "maxKey",
        "Data Type Value": BPBBooksVar7
    }
]);
```

```

Administrator Command Prompt - mongo
> var BPBBooksVar1 = 100000;
> var BPBBooksVar2 = "BPB Publications";
> var BPBBooksVar3 = 77.07;
> var BPBBooksVar4 = true;
> var BPBBooksVar5 = null;
> var BPBBooksVar6 = MinKey;
> var BPBBooksVar7 = MaxKey;
> db.BPBOnlineBooksDataTypesCollectionV2.insert([
...   {
...     "Data Type Alias": "int",
...     "Data Type Value": BPBBooksVar1
...   },
...   {
...     "Data Type Alias": "string",
...     "Data Type Value": BPBBooksVar2
...   },
...   {
...     "Data Type Alias": "double",
...     "Data Type Value": BPBBooksVar3
...   },
...   {
...     "Data Type Alias": "bool",
...     "Data Type Value": BPBBooksVar4
...   },
...   {
...     "Data Type Alias": "null",
...     "Data Type Value": BPBBooksVar5
...   },
...   {
...     "Data Type Alias": "minKey",
...     "Data Type Value": BPBBooksVar6
...   },
...   {
...     "Data Type Alias": "maxKey",
...     "Data Type Value": BPBBooksVar7
...   }
... ]);
> bulkWriteResult = [
...   {
...     "writeErrors": [ ],
...     "writeConcernErrors": [ ],
...     "nInserted": 7,
...     "nUpserted": 0,
...     "nMatched": 0,
...     "nModified": 0,
...     "nRemoved": 0,
...     "upserted": [ ]
... }
... ];

```

**Figure 9.15: MongoDB Data Types – Comparison and Sort Order – Inserting Values**

After the values are inserted, we can then use the sort method to sort and the values will be sorted according to the comparison and sort order. The code for the same is shown in the following screenshot.

### Code 3

```

db.BPBOnlineBooksDataTypesCollectionV2.find().sort({
  "Data Type Value": 1
});

```

```

Administrator Command Prompt - mongo
> db.BPBOnlineBooksDataTypesCollectionV2.find().sort({
...   "Data Type Value": 1
... });
> [
...   {
...     "_id": ObjectId("Seba#296000f9357bcd46112"),
...     "Data Type Alias": "minKey",
...     "Data Type Value": { "$minKey": 1 }
...   },
...   {
...     "_id": ObjectId("Seba#296000f9357bcd46113"),
...     "Data Type Alias": "null",
...     "Data Type Value": null
...   },
...   {
...     "_id": ObjectId("Seba#296000f9357bcd4610f"),
...     "Data Type Alias": "double",
...     "Data Type Value": 77.07
...   },
...   {
...     "_id": ObjectId("Seba#296000f9357bcd4610d"),
...     "Data Type Alias": "int",
...     "Data Type Value": 100000
...   },
...   {
...     "_id": ObjectId("Seba#296000f9357bcd4610e"),
...     "Data Type Alias": "string",
...     "Data Type Value": "BPB Publications"
...   },
...   {
...     "_id": ObjectId("Seba#296000f9357bcd46110"),
...     "Data Type Alias": "bool",
...     "Data Type Value": true
...   },
...   {
...     "_id": ObjectId("Seba#296000f9357bcd46113"),
...     "Data Type Alias": "maxKey",
...     "Data Type Value": { "$maxKey": 1 }
...   }
... ];

```

**Figure 9.16: MongoDB Data Types – Comparison and Sort Order – Sort Results**

In the preceding example, we saw how to create a new document using the data type having boolean type value.

## Conclusion

In this chapter, we studied about the data types and their uses. We also learned about the BSON data types in MongoDB. We learned the different data types available in MongoDB and how we can use these data types in MongoDB to create any document. We also learned the uses of data types in various scenarios from our practical examples.

We learned about the comparison and sort order of the data types in MongoDB and learned about their sorting order with the practical examples.

In the next part of this book, we will cover the intermediate level of MongoDB and will start to learn the basics of programming with MongoDB. In the next chapter, will introduce the concept of CRUD and will cover the MongoDB CRUD operations.

## **Questions**

1. What are data types?
2. Why data types are important?
3. What are the BSON data types in MongoDB?
4. Explain the three MongoDB data types with example.
5. What is min key and max key in MongoDB?

# CHAPTER 10

## Introduction to MongoDB CRUD Operations

This chapter covers the MongoDB CRUD operations. We will cover the operations helpful in creating, reading, updating, and deleting in MongoDB. We will also be covering the bulk write operation in MongoDB. All of these examples will be explained in step-by-step manner. We will be covering all these by giving the practical example and also covers various methods to perform all these operations. There are multiple methods for each operation that we will cover in this chapter. We will also learn about the various options, such as ordered, multi, and just one which we will use in the various MongoDB CRUD methods and we will also cover the field update operators used in the MongoDB update methods. These CRUD operations are very important in day-to-day working with MongoDB and are used by the application developers quite frequently. These methods will also be used in the advanced chapters of this book where we will learn about the application development with other programming languages, such as PHP, JavaScript, and Python.

### Structure

In this chapter, we will discuss the following topics:

- MongoDB create operations
- MongoDB read operations
- MongoDB update operations
- MongoDB delete operations
- MongoDB bulk write

### Objectives

After studying this unit, you should be able to create documents using the MongoDB create operations, Read documents using the MongoDB read operations, Update documents using the MongoDB update operations, Delete documents using the MongoDB delete operations and Later in this Chapter you will learn how to perform various MongoDB CRUD operations using the MongoDB bulk write method.

## **MongoDB create operations**

There are different ways in MongoDB by which we can create a document in the MongoDB collection. Few of the methods we can use are as follows:

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

As the name suggests, these methods are used to insert or create document(s) in the MongoDB collection. Let us study them one-by-one.

### **db.collection.insert() method**

This method inserts one or more than one document in the MongoDB collection.

### **Method definition**

```
db.collection.insert(  
  <single document or multiple documents in an array>,  
  {  
    ordered: <Boolean (true or false)>  
  }  
)
```

In our example, we learned how we can use the MongoDB method to create one or multiple documents.

## Example 1 – Creating a single document in MongoDB collection

We have created a variable `BPBBooksBestSellingEditions` which contains a JSON data to be inserted into the collection and the code for the same is as follows:

### Code 1

```
var BPBBooksBestSellingEditions = {  
    'Title': 'Cloud Computing',  
    'Year': '2019',  
    'ISBN': '9789388511407',  
    'Pages': 330,  
    'Weight': '570gm',  
    'Dimension': '23x19x1.5cm'  
};
```

And then, we have used the MongoDB `insert()` method to create a new document in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

### Code 2

```
db.BPBOnlineBooksCollection.insert(BPBBooksBestSellingEditions)  
;
```



```
Administrator: Command Prompt - mongo  
> var BPBBooksBestSellingEditions = {  
...     "Title": "Cloud Computing",  
...     "Year": "2019",  
...     "ISBN": "9789388511407",  
...     "Pages": 330,  
...     "Weight": "570gm",  
...     "Dimension": "23x19x1.5cm"  
... };  
> db.BPBOnlineBooksCollection.insert(BPBBooksBestSellingEditions); ②  
writeResult({ "nInserted": 1 })  
  
MongoDB insert() Method
```

**Figure 10.1:** Creating a single document in MongoDB Collection using `insert()` method

In the preceding example, we saw how to create a single document using the MongoDB `insert()` method.

## Example 2 – Creating multiple documents in MongoDB collection

In our example, we have created a variable `BPBBooksBestSellingEditions` which contains a JSON array data to be inserted into the collection and the code for the same is as follows:

### **Code 1**

```
var BPBBooksBestSellingEditions = [ {  
    'Title': 'Introduction to Digital Marketing 101 : Easy to  
    Learn and Implement Hands-on Guide for Digital Marketing',  
    'Year': '2019',  
    'ISBN': '9789389328189',  
    'Pages': 464  
}, {  
    'Title': 'IOT and Smart Cities: Your Smart City Planning  
    Guide',  
    'Year': '2019',  
    'ISBN': '9789388511322',  
    'Pages': 242,  
    'Weight': '357gm',  
    'Dimension': '22.5x15x1.5gm'  
} ];
```

And then, we have used the MongoDB `insert()` method to create two new documents in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

### **Code 2**

```
db.BPBOnlineBooksCollection.insert(BPBBooksBestSellingEditions)  
;
```

```

Administrator Command Prompt - mongo
> var BPBBooksBestSellingEditions = [
...   {
...     'Title': 'Introduction to Digital Marketing 101 : Easy to Learn and Implement Hands-on Guide for Digital Marketing',
...     'Year': '2019',
...     'ISBN': '9789389328189',
...     'Pages': 464
...   },
...   {
...     'Title': 'IOT and Smart Cities: Your Smart City Planning Guide',
...     'Year': '2019',
...     'ISBN': '9789388511322',
...     'Pages': 242,
...     'Weight': '357gm',
...     'Dimension': '22.5x15x1.5gm'
...   }
... ];
> db.BPBOnlineBooksCollection.insert(BPBBooksBestSellingEditions); ②
BulkWriteResult({
  "writeErrors": [ ],
  "writeConcernErrors": [ ],
  "nInserted": 2,
  "nUpserted": 0,
  "nMatched": 0,
  "nModified": 0,
  "nRemoved": 0,
  "nUpgraded": 0
})

```

**Figure 10.2:** Creating Multiple Documents in MongoDB Collection using `insert()` method

In the preceding example, we saw how to create multiple documents using the MongoDB `insert()` method.

## [db.collection.insertOne\(\) method](#)

This method inserts a single document in the MongoDB collection.

### Method definition

```
db.collection.insertOne(
  <single document>
)
```

## [Example – Creating a single document in MongoDB Collection using insertOne\(\) method](#)

In our example, we have created a variable `BPBBooksBestSellingEditions` which contains a JSON data to be inserted into the collection and the code for the same is as follows:

### **Code 1**

```
var BPBBooksBestSellingEditions = {
  'Title': 'Machine Learning with Python',
  'Year': '2018',
  'ISBN': '9789386551931',
  'Pages': 267
```

```
};
```

And then, we have used the MongoDB `insertOne()` method to create a new document in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksCollection.insertOne(BPBBooksBestSellingEditions);
```



```
Administrator: Command Prompt - mongo
> var BPBBooksBestSellingEditions = {
...   "title": "Machine Learning with Python",
...   "Year": "2018",
...   "ISBN": "9789386551931",
...   "Pages": 267
... };
> db.BPBOnlineBooksCollection.insertOne(BPBBooksBestSellingEditions);
{
  "acknowledged": true,
  "insertedId": ObjectId("5ec92dab7128f637c5804da4")
}

MongoDB insertOne() Method - Creating a Single Document
```

*Figure 10.3: Creating a Single Document in MongoDB Collection using `insertOne()` Method*

In the preceding example, we saw how to create a single document using the MongoDB `insertOne()` method.

## [db.collection.insertMany\(\) method](#)

This method inserts multiple documents in the MongoDB collection. Note that here, the documents can have different key values pairs and one document can be different from the others in terms of number of fields that it contains.

## [Method definition](#)

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

## Example – Creating multiple documents in MongoDB collection using insertMany() method

In our example, we have created a variable `BPBBooksBestSellingEditions` which contains a JSON array data to be inserted into the collection and the code for the same is as follows:

### **Code 1**

```
var BPBBooksBestSellingEditions = [ {  
    'Title': 'Artificial Intelligence Ethics and International  
    Law: An Introduction',  
    'Year': '2019',  
    'ISBN': '9789388511629',  
    'Pages': 188,  
    'Weight': '268gm'  
}, {  
    'Title': 'A Practical Approach for Machine Learning and Deep  
    Learning Algorithms',  
    'Year': '2019',  
    'ISBN': '9789388511131',  
    'Pages': 280,  
    'Weight': '424gm'  
} ];
```

And then, we have used the MongoDB `insertMany()` method to create two new documents in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

### **Code 2**

```
db.BPBOnlineBooksCollection.insertMany(BPBBooksBestSellingEditi  
ons);
```

```

Administrator Command Prompt - mongo
> var BPBBooksBestSellingEditions = [
...   {
...     "title": "Artificial Intelligence Ethics and International Law: An Introduction",
...     "Year": "2019",
...     "ISBN": "9789388511629",
...     "Pages": 188,
...     "Weight": "260gms"
...   },
...   {
...     "title": "A Practical Approach for Machine Learning and Deep Learning Algorithms", ①
...     "Year": "2019",
...     "ISBN": "9789388511131",
...     "Pages": 280,
...     "Weight": "424gms"
... };
> db.BPBOnlineBooksCollection.insertMany(BPBBooksBestSellingEditions); ②
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("5ec92eb67128f637c5804da5"),
    ObjectId("5ec92eb67128f637c5804da6")
  ]
}

```

MongoDB insertMany() Method - Creating Multiple Documents

**Figure 10.4:** Creating Multiple Documents in MongoDB Collection using insertMany() Method

In the preceding example, we saw how to create a single document using the MongoDB insertOne() method.

## The `_id` Field

In MongoDB, every document must have an `_id` field and it should be unique. It acts as a primary key in the MongoDB collection.

MongoDB automatically creates the `_id` field if not specified during the creation of a new document and its value is assigned as the `ObjectId` type by default.

We can also specify the `_id` value and it can be any BSON data type other than array. But it must be unique in a collection, otherwise, the document will not be created and you will get an error if you try to insert a document having a non-unique `_id` value.

## Example - Creating a new document by specifying `_id` key

In our example, we have created a variable `BPBBooksBestSellingEditions` which contains a JSON data to be inserted into the collection. This data also contains an `_id` key with value as `20021111` and the code for the same is as follows:

### Code 1

```
var BPBBooksBestSellingEditions = {
  '_id': '20021111',
```

```

    'Title': 'Introduction to Database Management',
    'Year': '2002',
    'ISBN': ' 9788176566384',
    'Pages': 342
};

}

```

And then, we have used the MongoDB `insert()` method to create a new document in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksCollection.insert(BPBBooksBestSellingEditions)
;
```

The screenshot shows the MongoDB shell interface. A variable `BPBBooksBestSellingEditions` is defined with a document containing an `_id` field (marked with a red circle 1). This document is then inserted into the `BPBOnlineBooksCollection` (marked with a red circle 2). The output shows the insertion result with `nInserted: 1`. A callout box highlights the process: "MongoDB insert() Method - Creating a Document by specifying the \_id".

**Figure 10.5:** Creating a Document in MongoDB Collection using `insert()` Method by specifying the `_id`

In the preceding example, we saw how to create a single document using the MongoDB `insert()` method by specifying the `_id` for the document.

## The ordered option

The `ordered` option is by default set to true in the MongoDB `insert()` and `insertMany()` methods, and is used while we insert multiple documents in a collection.

If `ordered` option is set to true and if any error occurs while inserting any document while we are inserting multiple document, in that case, MongoDB will not process the remaining documents and will return back with an error.

If `ordered` option is set to false and if any error occurs while inserting any document while we are inserting multiple document, in that case, MongoDB will continue to process the remaining documents.

## MongoDB read operations

In a MongoDB collection, we can read documents by using the `find()` method.

- `db.collection.find()`

As the name suggests, this method is used to find or read document(s) in the MongoDB collection.

### [db.collection.find\(\) Method](#)

This method finds or read one or more documents in the MongoDB collection.

### Method Definition

```
db.collection.find(query, projection)
```

We will cover about projection in the advanced chapters, which are an optional value. So, we will only cover the Query part.

### [Example 1 – Reading documents in MongoDB collection without Query](#)

In our example, we have used the `find()` method to read all the documents in a collection named `BPBBooksBestSellingEditions` and the code for the same is shown in the following screenshot:

#### **Code 1**

```
db.BPBOnlineBooksCollection.find();
```

```

db.BPBOnlineBooksCollection.find()
[{"_id": ObjectId("Sec92ca17128f637c5884da6"), "Title": "Cloud Computing", "Year": "2019", "ISBN": "9789388511407", "Pages": 330, "Weight": "570gm", "Dimension": "23x19x1.5cm"}, {"_id": ObjectId("Sec92ca17128f637c5884da1"), "Title": "Introduction to Digital Marketing 101 : Easy to Learn and Implement Hands-on Guide for Digital Marketing", "Year": "2019", "ISBN": "9789389328189", "Pages": 464}, {"_id": ObjectId("Sec92ca17128f637c5884da2"), "Title": "IoT and Smart Cities: Your Smart City Planning Guide", "Year": "2019", "ISBN": "9789388511322", "Pages": 220, "Weight": "357gm", "Dimension": "22.5x15x1.5cm"}, {"_id": ObjectId("Sec92ca17128f637c5884da4"), "Title": "Machine Learning with Python", "Year": "2018", "ISBN": "9789386551931", "Pages": 267}, {"_id": ObjectId("Sec92ca17128f637c5884da5"), "Title": "Artificial Intelligence Ethics and International Law: An Introduction", "Year": "2019", "ISBN": "9789388511629", "Pages": 188, "Weight": "260gm"}, {"_id": ObjectId("Sec92da67128f637c5884da6"), "Title": "A Practical Approach for Machine Learning and Deep Learning Algorithms", "Year": "2019", "ISBN": "9789388511311", "Pages": 280, "Weight": "424gm"}, {"_id": ObjectId("28021111"), "Title": "Introduction to Database Management", "Year": "2002", "ISBN": "9788176566384", "Pages": 342}]

```

MongoDB find() Method

**Figure 10.6:** Reading MongoDB Documents using `find()` method

In the preceding example, we saw how to read documents using the MongoDB `find()` method.

## Example 2 – Reading documents in MongoDB collection with Query

In our example, we have used the `find()` method to read all the documents in a collection named `BPBBooksBestSellingEditions` which has a key equal to `Year` and value equal to `2002`, and the code for the same is shown in the following screenshot:

### Code 1

```
db.BPBOnlineBooksCollection.find({ 'Year': '2002' });
```

```

db.BPBOnlineBooksCollection.find({ 'Year': '2002' })
[{"_id": "28021111", "Title": "Introduction to Database Management", "Year": "2002", "ISBN": "9788176566384", "Pages": 342}]

```

MongoDB find() Method - with Selection Query

**Figure 10.7:** Reading MongoDB Documents using `find()` Method with Selection Query

In our example, we learned how we can use the MongoDB `find()` method to find and read documents with the help of selection query.

## Using Pretty method with find()

In MongoDB, we can use the `pretty()` method along with the `find()` method to improve the readability of the results generated by the

`find()` method.

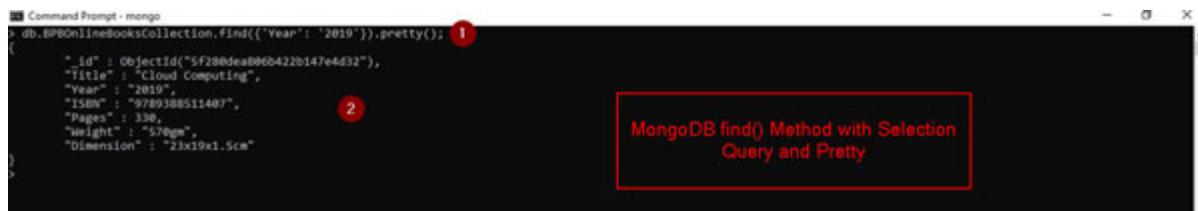
By using the `pretty()` method, we will get the results in easy to read attractive format.

## Example 3 – Reading documents in MongoDB collection with Query and Pretty method

In our example, we have used the `find()` method to read all the documents in the collection named `BPBBooksBestSellingEditions` which has a key equal to `Year` and value equal to `2019`, and the code for the same is shown in the following screenshot:

### **Code 1**

```
db.BPBOnlineBooksCollection.find({'Year': '2019'}).pretty();
```



The screenshot shows a Command Prompt window titled "Command Prompt - mongo". Inside, a MongoDB shell command is run: `db.BPBOnlineBooksCollection.find({'Year': '2019'}).pretty();`. The output is a single document represented in a pretty-printed JSON format. A red box labeled "1" highlights the first character of the command. A red box labeled "2" highlights the first character of the document's content. A red-bordered callout box in the bottom right corner contains the text "MongoDB find() Method with Selection Query and pretty() Method".

```
{ "_id" : ObjectId("5f280dea8860422b147e4d32"),  
  "Title" : "Cloud Computing",  
  "Year" : "2019",  
  "ISBN" : "9789388511407",  
  "Pages" : 330,  
  "Weight" : "570gm",  
  "Dimension" : "23x19x1.5cm"}
```

**Figure 10.8:** Reading MongoDB Documents using `find()` Method with Selection Query and `pretty()` Method

In our example, we learned how we can use the MongoDB `find()` method to find and read documents with the help of selection query and the `pretty()` method to display the documents in more readable form.

## MongoDB update operations

There are different ways in MongoDB to update a document in the MongoDB collection.

Following are the few methods we can use:

- `db.collection.update()`
- `db.collection.updateOne()`
- `db.collection.updateMany()`

As the name suggests, these methods are used to update or change document(s) values in the MongoDB collection.

Let us study them one-by-one. But before we move on to our examples, let us understand the field update operators that will be used in these methods.

## The \$set operator

The `$set` operator sets the value of the field in a document. It is used to update the values of one or more fields in a document in conjunction with the selection query given in the update methods.

## The \$unset operator

The `$unset` operator removes the field in a document. It is used to delete one or more fields in a document in conjunction with the selection query given in the update methods.

## db.collection.update() method

This method updates one or more documents in the MongoDB collection.

## Method definition

The first part of this method is a selection query which is used to select our documents based on some criteria, and in the second part, we use the data to be updated in our documents. In both of these parts, we use the JSON format.

```
db.collection.update(  
  <selection query>,  
  <data to update>,  
  {  
    upsert: <boolean>,  
    multi: <boolean>  
  }  
)
```

Let us now use this method with some practical examples.

## Example 1 – Updating a single document in MongoDB collection using update() method

In our example, we have created two variables `BPBBooksSelectionQuery` and `BPBBooksDatatoUpdate`. The first one has a JSON data for selection query and the second one has a JSON data for updating the document. The code for the same is as follows:

### Code 1

```
var BPBBooksSelectionQuery = {  
    'Title': 'Introduction to Database Management'  
};  
  
var BPBBooksDatatoUpdate = {  
    'Title': 'Introduction to Database Management (The Complete  
    Text Book for Computer Science Students)'  
};
```

And then, we have used the MongoDB `update()` method to update a document based on our selection query in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

### Code 2

```
db.BPBOnlineBooksCollection.update(BPBBooksSelectionQuery, {  
    $set: BPBBooksDatatoUpdate  
});
```



The screenshot shows a terminal window titled "Administrator Command Prompt - mongo". It displays the following MongoDB shell script:

```
var BPBBooksSelectionQuery = [  
...     "Title": "Introduction to Database Management"  
...];  
var BPBBooksDatatoUpdate = [  
...     "Title": "Introduction to Database Management (The Complete Text Book for Computer Science Students)"  
...];  
db.BPBOnlineBooksCollection.update(BPBBooksSelectionQuery, {  
...     $set: BPBBooksDatatoUpdate  
...});  
writeResult({ "nMatched": 1, "nInserted": 0, "nModified": 1 })
```

Three red circles numbered 1, 2, and 3 point to specific lines in the code:

- Circle 1 points to the line `var BPBBooksSelectionQuery = [`.
- Circle 2 points to the line `var BPBBooksDatatoUpdate = [`.
- Circle 3 points to the line `$set: BPBBooksDatatoUpdate`.

A red box at the bottom right of the terminal window contains the text: "MongoDB update() Method - Updating a Single Document".

**Figure 10.9: Updating a Single Document in MongoDB Collection using update() Method**

In the preceding example, we saw how to update a single document using the MongoDB `update()` method.

## **Example 2 – Updating multiple documents in MongoDB collection using update() method**

In our example, we have created two variables “BPBBooksSelectionQuery” and “BPBBooksDatatoUpdate”. The first one has a JSON data for selection query and the second one has a JSON data for updating the document. The code for the same is as follows:

### **Code 1**

```
var BPBBooksSelectionQuery = {  
    'Year': '2019'  
};  
  
var BPBBooksDatatoUpdate = {  
    'Publisher': 'BPB Publications'  
};
```

And then, we have used the MongoDB `update()` method to update all the documents based on our selection Query in the MongoDB collection, `BPBOnlineBooksCollection`, with multi option set to true. The code for the same is shown in the following screenshot:

### **Code 2**

```
db.BPBOnlineBooksCollection.update(BPBBooksSelectionQuery, {  
    $set: BPBBooksDatatoUpdate  
}, {  
    multi: true  
});
```



MongoDB update() Method - Updating Multiple Documents

**Figure 10.10:** Updating Multiple Documents in MongoDB Collection using update() Method

In the preceding example, we saw how to update multiple documents using the MongoDB `update()` method.

## db.collection.updateOne() method

This method updates a single document in the MongoDB collection.

### Method definition

```
db.collection.updateOne(
  <filter>,
  <update>,
  {
    upsert: <boolean>
  }
)
```

## Example – Updating a single document in MongoDB collection using updateOne() method

In our example, we have created two variables `BPBBooksSelectionQuery` and `BPBBooksDataToUpdate`. The first one has a JSON data for selection query and the second one has a JSON data for updating the document. The code for the same is as follows:

### **Code 1**

```
var BPBBooksSelectionQuery = {
```

```

    'Title': 'IOT and Smart Cities: Your Smart City Planning
    Guide'
};

var BPBBooksDatatoUpdate = {
    'Title': 'IOT and Smart Cities',
    'Year': '2020'
};

```

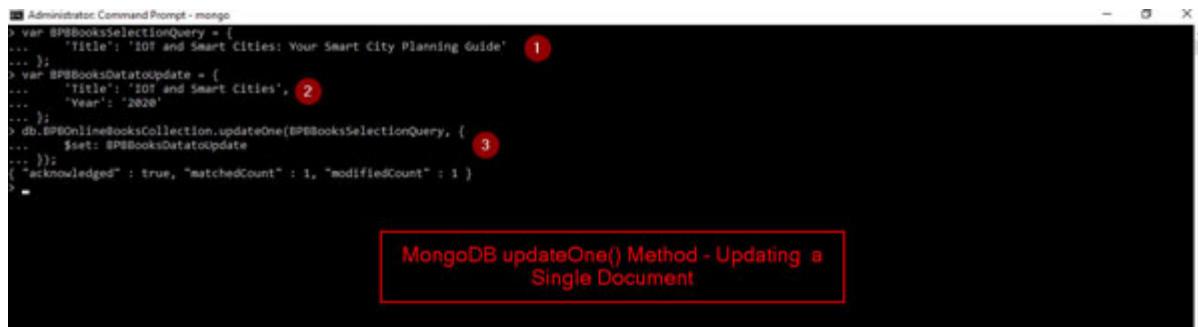
And then, we have used the MongoDB `updateOne()` method to update a document based on our selection query in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

## Code 2

```

db.BPBOnlineBooksCollection.updateOne(BPBBooksSelectionQuery, {
    $set: BPBBooksDatatoUpdate
}) ;

```



The screenshot shows a terminal window titled "Administrator Command Prompt - mongo". It contains the following MongoDB shell code:

```

> var BPBBooksSelectionQuery = [
...   'Title': 'IOT and Smart Cities: Your Smart City Planning Guide' ①
... ];
> var BPBBooksDatatoUpdate = [
...   'Title': 'IOT and Smart Cities', ②
...   'Year': '2020'
... ];
> db.BPBOnlineBooksCollection.updateOne(BPBBooksSelectionQuery, {
...   $set: BPBBooksDatatoUpdate
... });
> {"acknowledged": true, "matchedCount": 1, "modifiedCount": 1 }

```

A red box highlights the text "MongoDB updateOne() Method - Updating a Single Document" at the bottom of the terminal window.

**Figure 10.11:** Updating a single document in MongoDB Collection using `updateOne()` method

In the preceding example, we saw how to update a single document using the MongoDB `updateOne()` method.

## [db.collection.updateMany\(\) method](#)

This method updates multiple documents in the MongoDB collection.

## [Method definition](#)

```
db.collection.updateMany(
```

```
<filter>,
<update>,
{
    upsert: <boolean>
}
)
```

## [Example – Updating multiple documents in MongoDB collection using updateMany\(\) method](#)

In our example, we have created two variables `BPBBooksSelectionQuery` and `BPBBooksDatatoUpdate`. The first one has a JSON data for the selection query and the second one has a JSON data for updating the document. The code for the same is as follows:

### **Code 1**

```
var BPBBooksSelectionQuery = {
    'Year': '2019'
};

var BPBBooksDatatoUpdate = {
    'Discount': '10%'
};
```

And then, we have used the MongoDB `updateMany()` method to update a document based on our selection query in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

### **Code 2**

```
db.BPBOnlineBooksCollection.updateMany(BPBBooksSelectionQuery,
{
    $set: BPBBooksDatatoUpdate
});
```

```

Administrator Command Prompt - mongo
> var BPBooksSelectionQuery = [
...   { "Year": "2019" } ①
... ];
> var BPBooksDataToUpdate = {
...   "Discount": "10%" ②
... };
> db.BPBOnlineBooksCollection.updateMany(BPBooksSelectionQuery, {
...   $set: BPBooksDataToUpdate ③
... });
> {"acknowledged": true, "matchedCount": 4, "modifiedCount": 4}

```

**MongoDB updateMany() Method - Updating Multiple Documents**

**Figure 10.12:** Updating multiple documents in MongoDB Collection using `updateMany()` method

In the preceding example, we saw how to update a single document using the MongoDB `updateOne()` method.

## The upsert option

The upsert option is by default set to false in the MongoDB `update()`, `updateOne()`, and `updateMany()` methods, and can be used while we are updating single or multiple documents in a collection.

If the upsert option is set to true and if there is no document in a collection that matches the selection query, a new document will automatically be created by the MongoDB update methods.

If the upsert option is set to false and if there is no document in a collection that matches the selection query, no new document will be created by the MongoDB update methods.

## The multi option

The multi option is by default set to false in the MongoDB `update()` method, and can be used while we update documents in a collection.

If multi option is set to true, it will update multiple documents according to the selection query.

If multi option is set to false, it will update single document according to the selection query.

## MongoDB delete operations

There are different ways in MongoDB by which we can delete a document in the MongoDB collection.

Following are the few methods which we can use:

- db.collection.remove()
- db.collection.deleteOne()
- db.collection.deleteMany()

As the name suggests, these methods are used to delete or remove the document(s) in the MongoDB collection.

Let us study them one-by-one.

## [db.collection.remove\(\) method](#)

This method deletes one or more documents in the MongoDB collection.

`remove()` method is a depreciated function and it is going to be unavailable in the future versions of MongoDB. So, it is better to use the `deleteOne()` and `deleteMany()` methods instead of the `remove()` method.

## [Method definition](#)

```
db.collection.remove(  
<selection query>,  
{  
  justOne: <boolean>  
}  
)
```

## [The justOne option](#)

The `justOne` option is by default set to false in the MongoDB `remove()` method and can be used while we are removing the documents in a collection.

If `justOne` option is set to true, it will remove only one document according to the selection query.

If `justOne` option is set to false, then it will remove multiple documents according to the selection query.

## **Example 1 – Deleting a single document in MongoDB collection using `remove()` method**

In our example, we have created a variable `BPBBooksSelectionQuery` which is having a JSON data for selection query and the code for the same is as follows:

### **Code 1**

```
var BPBBooksSelectionQuery = {  
    'Year': '2002'  
};
```

And then, we have used the MongoDB `remove()` method to delete a document based on our selection query in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

### **Code 2**

```
db.BPBOnlineBooksCollection.remove(  
    BPBBooksSelectionQuery,  
    {  
        justOne: true  
    }  
);
```



The image shows a terminal window titled "Administrator: Command Prompt - mongo". It displays the following command:

```
var BPBBooksSelectionQuery = {  
    'Year': '2002'  
};  
db.BPBOnlineBooksCollection.remove(  
    BPBBooksSelectionQuery,  
    {  
        justOne: true  
    }  
);  
writeResult({ "nRemoved" : 1 })
```

Two annotations are present: a red circle labeled "1" points to the line where the selection query is defined; another red circle labeled "2" points to the line where the `justOne: true` option is specified. A red box highlights the text "MongoDB remove() Method - Deleting a Single Document".

**Figure 10.13: Deleting a Single Document in MongoDB Collection using `remove()` Method**

In the preceding example, we saw how to delete a single document using MongoDB `remove()` method.

## **Example 2 – Deleting multiple documents in MongoDB collection using remove() method**

In our example, we have created a variable `BPBBooksSelectionQuery` which is having a JSON data for selection query and the code for the same is as follows:

### **Code 1**

```
var BPBBooksSelectionQuery = {  
    'Year': '2019'  
};
```

And then, we have used the MongoDB `remove()` method to delete multiple documents based on our selection query in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

### **Code 2**

```
db.BPBOnlineBooksCollection.remove(BPBBooksSelectionQuery);
```

MongoDB remove() Method - Deleting Multiple Documents

**Figure 10.14:** Deleting multiple documents in MongoDB Collection using `remove()` method

In the preceding example, we saw how to delete multiple documents using the MongoDB `remove()` method.

## **db.collection.deleteOne() method**

This method deletes one document in the MongoDB collection.

## Method definition

```
db.collection.deleteOne(  
  <filter>  
)
```

## Example – Deleting a single document in MongoDB collection using deleteOne() method

In our example, we have created a variable `BPBBooksSelectionQuery` having a JSON data for selection query, and the code for the same is as follows:

### **Code 1**

```
var BPBBooksSelectionQuery = {  
  'Year': '2020'  
};
```

And then, we have used the MongoDB `deleteOne()` method to delete a single document based on our selection query in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

### **Code 2**

```
db.BPBOnlineBooksCollection.deleteOne(BPBBooksSelectionQuery);
```



The screenshot shows a terminal window titled "Administrator: Command Prompt - mongo". Inside, a command is being run to delete a document from the "BPBOnlineBooksCollection" collection using a selection query defined in the variable "BPBBooksSelectionQuery". The output indicates that the deletion was successful, with "acknowledged": true and "deletedCount": 1.

MongoDB deleteOne() Method - Deleting a Single Document

**Figure 10.15: Deleting a single document in MongoDB Collection using deleteOne() method**

In the preceding example, we saw how to delete a single document using the MongoDB `deleteOne()` method.

## db.collection.deleteMany() method

This method deletes more than one documents in the MongoDB collection.

### Method definition

```
db.collection.deleteMany(  
  <filter>  
)
```

### Example – Deleting multiple documents in MongoDB collection using deleteMany() method

In our example, we have created a variable `BPBBooksSelectionQuery` which is having a JSON data for selection query and the code for the same is as follows:

#### **Code 1**

```
var BPBBooksSelectionQuery = {  
  'Year': '2019'  
};
```

And then, we have used the MongoDB `deleteMany()` method to delete multiple documents based on our selection query in the MongoDB collection, `BPBOnlineBooksCollection`, and the code for the same is shown in the following screenshot:

#### **Code 2**

```
db.BPBOnlineBooksCollection.deleteMany(BPBBooksSelectionQuery);
```



The screenshot shows a terminal window titled "Administrator: Command Prompt - mongo". The command entered is `db.BPBOnlineBooksCollection.deleteMany(BPBBooksSelectionQuery);`. The output shows the command at line 1 and the result at line 2. A red box highlights the command line, and another red box highlights the output line.

```
Administrator: Command Prompt - mongo  
1 var BPBBooksSelectionQuery = {  
2   'Year': '2019'  
3 };  
4 db.BPBOnlineBooksCollection.deleteMany(BPBBooksSelectionQuery); 2  
5 { "acknowledged" : true, "deletedCount" : 2 }
```

MongoDB deleteMany() Method - Deleting Multiple Documents

**Figure 10.16:** Deleting multiple Documents in MongoDB Collection using `deleteMany()` Method

In the preceding example, we saw how to delete multiple documents using the MongoDB `deleteMany()` method.

## **MongoDB bulk write operations**

MongoDB provides a way to perform multiple writing operations. By using the MongoDB bulk write, we can perform bulk insert, bulk update and bulk remove operations in one go.

These operations can be ordered or unordered.

If the `ordered` option is set to true and if any error occurs while performing bulk write, in that case, MongoDB will not process the remaining write operations and will return an error.

If the `ordered` option is set to false and if any error occurs while performing bulk write, in that case, MongoDB will still process the remaining write operations.

## **db.collection.bulkWrite() method**

This method performs the multiple write operations in the MongoDB collection.

## **Method definition**

```
db.collection.bulkWrite(  
  [<operation 1>, <operation 2>, ... <operation N>],  
  {  
    ordered : <boolean>  
  }  
)
```

`bulkWrite()` supports the following write operations:

- `insertOne`
- `updateOne`
- `updateMany`
- `replaceOne`

- deleteOne
- deleteMany

**Let us go through the definitions of these `Write` operations.**

```
db.collection.bulkWrite([
  {insertOne : {"document" : <document>}}
])
db.collection.bulkWrite([
  {updateOne :
    {
      "filter": <document>,
      "update": <document or pipeline>,
      "upsert": <boolean>
    }
  }
])
db.collection.bulkWrite([
  {updateMany :
    {
      "filter": <document>,
      "update": <document or pipeline>,
      "upsert": <boolean>
    }
  }
])
db.collection.bulkWrite([
  {replaceOne :
    {
      "filter" : <document>,
      "replacement" : <document>,
      "upsert" : <boolean>
    }
  }
])
db.collection.bulkWrite([
  {deleteOne : {
    "filter" : <document>
  }}
])
```

```

        } }
    ])
db.collection.bulkWrite([
  {deleteMany : {
    "filter" : <document>
  }}
])

```

## Example – Bulk write in MongoDB collection using bulkwrite() method

The following code will perform bulk write with six operations:

```

db.collection.bulkWrite(
  [
    {insertOne : <document>} ,
    {updateOne : <document>} ,
    {updateMany : <document>} ,
    {replaceOne : <document>} ,
    {deleteOne : <document>} ,
    {deleteMany : <document>}
  ]
)

```

In the preceding code, bulk write will be executed in an ordered way as it is, by default, set to true. So, the first operation which will be executed is `insertOne` and last operation which will be executed is `deleteMany`.

Let us assume we have 3 documents already existing in our collection `BPBOnlineBooksCollection`.

```

{"_id" : 1, "Title" : "Introduction to Python", "Year" :
"2017", "Price" : 500},
{"_id" : 2, "Title" : "Mastering MySQL", "Year" : "2010",
"Price" : 600},
{"_id" : 3, "Title" : "Learn JavaScript in 24 Hrs", "Year" :
"2015", "Price" : 400}

```

Now, we will perform bulk write in our MongoDB collection and will see the effect. The code for the same is shown in the following screenshot:

## Code 1

```
db.BPBOnlineBooksCollection.bulkWrite([
  {insertOne: {"document": {"_id": 4, "Title": "Learn C++",
    "Year": "2000", "Price": 450}}},
  {insertOne: {"document": {"_id": 5, "Title": "Mastering Java",
    "Year": "2005", "Price": 700}}},
  {updateOne : {
    "filter" : {"Title" : "Learn C++"},
    "update" : {"$set" : {"Publisher" : "BPB Publication"} }
  },
  {deleteOne : {"filter" : {"Year" : "2010"} }},
  {replaceOne : {
    "filter" : {"Title" : "Introduction to Python"},
    "replacement" : {"Title" : "Mastering Python", "Year" :
      "2020", "Price": 800}
  }
} );
]
```

The screenshot shows the MongoDB command prompt window titled "Administrator Command Prompt - mongo". The command entered is a bulk write operation on the "BPBOnlineBooksCollection" database. The operation consists of several stages: inserting two documents (with IDs 4 and 5), updating one document (ID 4), deleting one document (Year 2010), and replacing one document (Title "Introduction to Python"). A red box highlights the "MongoDB bulkWrite() Method - Bulk Write Operation" text.

**Figure 10.17: Bulk Write Operation in MongoDB Collection using bulkWrite() Method**

In the preceding example, we saw how to perform bulk write in the MongoDB collection using the MongoDB `bulkWrite()` method.

## **Conclusion**

In this chapter, we studied the MongoDB CRUD operations, and learned how to create, read, update, and delete using the various MongoDB methods, specifically used to perform these CRUD Operations.

We also learned about the MongoDB bulk write method in which we covered how to perform bulk write in order to achieve multiple operations in a single go. We also learned about the various options such, as ordered, multi, and justOne which are used in the various MongoDB CRUD methods and learned about the field update operators used in MongoDB update methods.

In the next chapter of this book, we will cover MongoDB CRUD concepts wherein we will cover the topics like atomicity, consistency, distributed operations, query plan, and performance and analysis.

## **Questions**

1. Give examples of two create document methods used in MongoDB?
2. What is multi option and how can we use it? Give example.
3. Give examples of two document update methods used in MongoDB?
4. Explain the use of field update operators and where are they useful?
5. Give example of bulk write operation in MongoDB

# CHAPTER 11

## MongoDB Intermediate Concepts

This chapter covers the MongoDB intermediate concepts. In this chapter, we will cover the topics like atomicity and consistency in MongoDB. We will also learn about replication and sharding in MongoDB. This chapter also gives the basic introduction to replication and why it is useful. We will also learn about sharding and why it is useful. Later in this chapter, we will cover the MongoDB specific distributed operations and queries in which we will look how the read and write operations are performed when we use replication and sharding in MongoDB.

### Structure

In this chapter, we will discuss the following topics:

- Atomicity
- Consistency
- Basic introduction to replication
- Basic introduction to sharding
- Distributed operations and queries

### Objectives

After studying this unit, you should be able to understand the atomicity and consistency in MongoDB. Later in this chapter, you will get a brief introduction to replication and sharding in MongoDB, which are also covered in the advanced chapters in this book. In the last section of this chapter, you will learn about the distributed operations in MongoDB.

### Atomicity

Before we learn about the atomicity in MongoDB, let us first understand what exactly atomicity is in terms of database management systems.

## What is atomicity?

Atomicity is the property of the database management systems in which all the operations, like insert, update, and delete will happen for a transaction or nothing will happen at all. In this, either all the operations will complete or they will roll back.

## Atomicity in MongoDB

MongoDB supports atomicity at the single document level. In MongoDB, if there is a document which contains hundreds of fields, then an `update` statement will update all the fields or none. So, MongoDB maintains the atomicity at the single document level.

We know that in MongoDB, we can have a single document which can have multiple documents. So, in MongoDB, the write operation for a single document is atomic in nature, even if it updates multiple embedded documents inside a single document.

## MongoDB atomicity and multiple document transactions

MongoDB does not support the atomicity for multiple document transactions. For example, in some cases, we use single write operation which can update multiple documents, like, when we use the method such as the `db.collection.updateMany()` method which can modify multiple documents. Even in this case, the modification in a single document level is atomic but the overall operation is not atomic.

## Consistency

Before we learn about the consistency in MongoDB, let us first understand what exactly is consistency in terms of database management systems.

## What is consistency?

Consistency is the property of the database management systems in which a consistent state is maintained in the database. This means whatsoever happens during that transaction, it will never leave the database in a half-consistent state or a non-consistent state. For example:

- If the transaction happened successfully, all the changes will be applied to the database
- If the transaction does not happen successfully due to any error or system failure, in this case, the changes will be rolled back and the database will be maintained to its original state where it was before the transaction began.

## Consistency in MongoDB

MongoDB supports consistency and in it, the data is consistent by default. The applications can read and write to MongoDB replica sets, so:

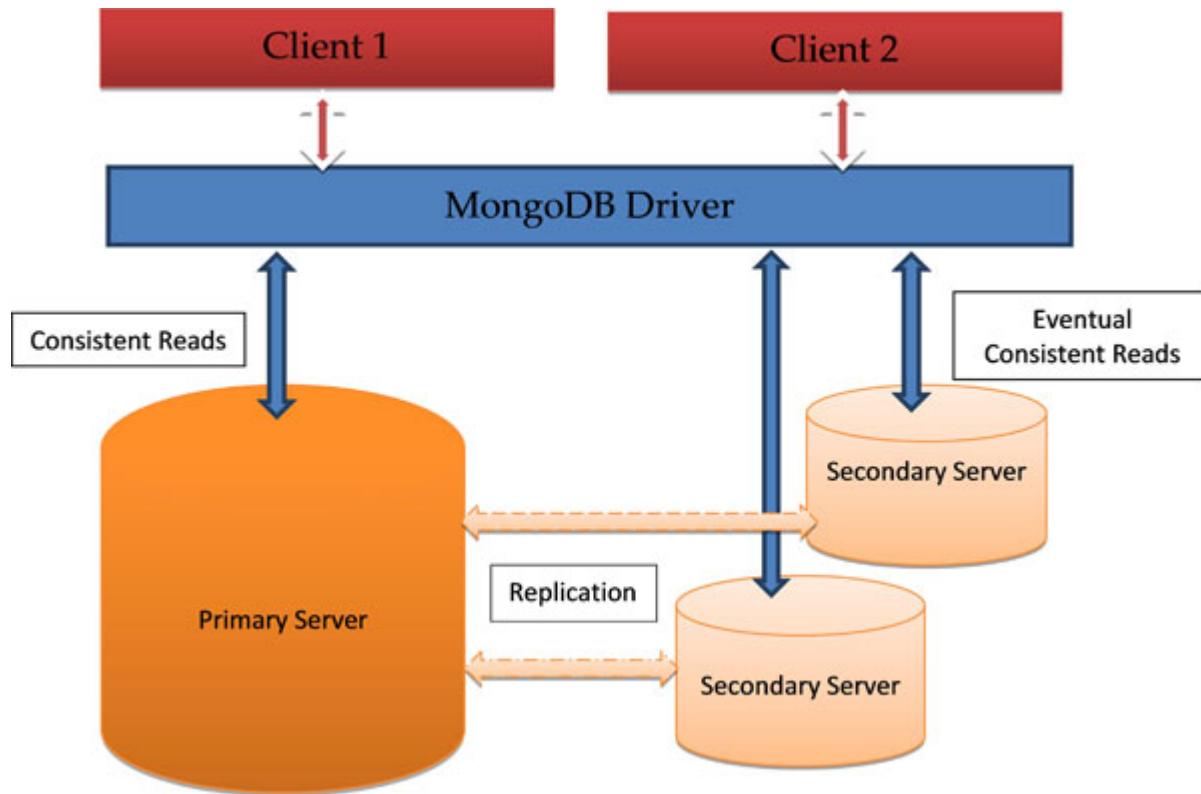
- For MongoDB, the primary members of replica set, the read and write are consistent.
- Sometimes, the applications also read from the secondary replica sets. In this case, for the MongoDB secondary members of the replica set, the data is eventually consistent by default.

## MongoDB and eventual consistency

Eventual consistency is the modal in MongoDB which guarantees that if there are no new updates made in a given object (let's assume a document), then all the accesses to that object will be returned with the last updated value only.

Let us explain this model with an example. Let us assume there are 2 clients whose requirements are reading and writing on a document with the current value (`Example booktitle = "Mysql"`), and then, at some point of time, the 1st client wants to update the value to a new value (`Example booktitle = "MongoDB"`). In this case, the 2nd client

needs to record its new value (`Example bookpublisher = "BPB Publications"`) just before the 1st client makes the changes to the current document or wait till the 1st client has updated the document and made the changes and the database write lock gets released.



*Figure 11.1: Consistency in MongoDB*

## Basic introduction to replication

Replication is a process of duplicating the same set of databases so that we have multiple copies of the same data available to us on different servers because of which we have redundancy and high availability of data.

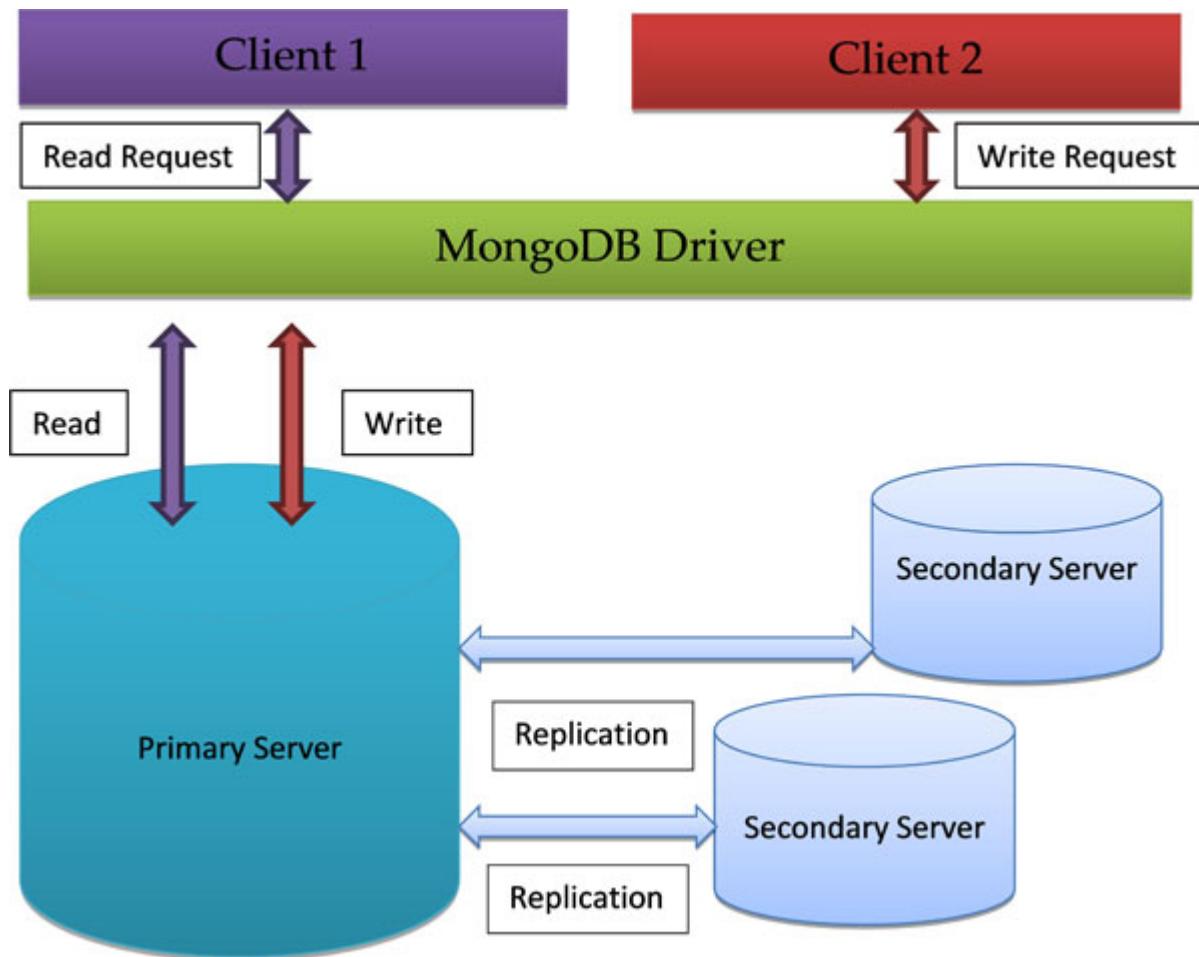
Following are some of the benefits of replication:

- Replication helps in the fail-over recovery, such as, in case one server fails, or in case of hardware failure, or any other service interruptions.
- Replication also helps in the increased performance because clients can send their requests to different servers.

- Replication helps in delivering the data to the distributed applications from various data center locations across the globe.

## Replica sets

Replica sets are the groups of MongoDB processes or instances that host the same data set. There is one primary and many secondary nodes that form a group to create replica sets, as shown in the following figure:



*Figure 11.2: Replication in MongoDB*

We will cover more about replication in the advanced chapters of this book.

## Basic introduction to sharding

Replication is a process of distributing large data across multiple machines or servers. There are cases where the applications have large sets of data that cannot be served by a single machine or server due to their hardware limits. These limits can be due to the CPU capacities or sometimes memory capacities like the RAM or disk drives.

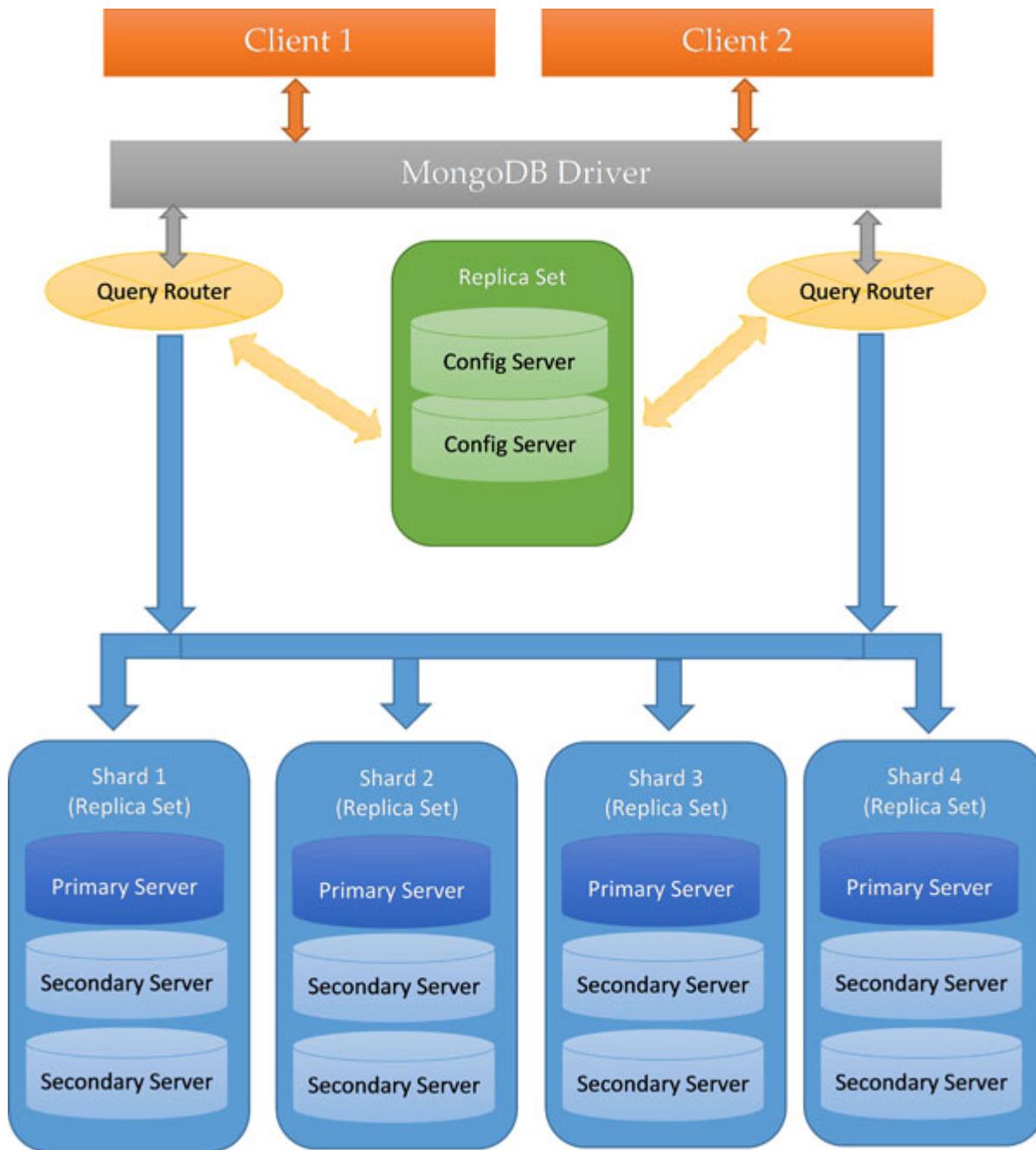
In this case, MongoDB provides the feature of sharding where we can distribute our large data across different machines or servers. By doing this, we can serve this large data easily with the increased computational power which we gained by using more machines or servers instead of a single machine or server.

Following are some benefits of sharding:

- Increase in computational power due to the use of multiple machines or servers.
- Lower costs because instead of increasing the computational capacities of one machine, which has a limitation too, we increase the computational capacities by involving multiple machines which together create a lot of computational power and is a cheaper option too.
- MongoDB supports horizontal scaling by using sharding which is a more effective solution. Many a times, it is not practically possible to increase the CPU and memory capacities of a single machine as it has a maximum limit and is expensive. Increasing a computational capacity of a single machine is also termed as vertical scaling. So, it is always a better and cheaper option to increase the computational power by using multiple machines. This is also called horizontal scaling.

## **Sharded clusters**

In simple terms, sharded clusters are groups of MongoDB instances used to serve large data. Here, the data is served to the applications by splitting the large data into multiple sharded clusters, as shown in the following figure:



**Figure 11.3: Sharding in MongoDB**

We will cover more about sharding in the advanced chapters of this book.

## Distributed operations and queries

As we have studied, in MongoDB, we can have replication as well as sharding, and we are also aware of their usefulness and advantages.

Let us now understand how the read and write operations are performed in these distributed replica sets or sharded clusters.

## **Read operations on replica sets**

Normally, in MongoDB, the clients usually read from the primary replica set, but they may also opt to use the secondary replica set or the nearest member of the replica set to avoid various factors, like the reduction in the latency due to multiple data center locations or sometimes to improve the read performance.

## **Write operations on replica sets**

In MongoDB, all the write operations are done on the primary replica set. Whenever the clients send the write requests to MongoDB, the requests go to the primary replica set. Here, the primary replica set then maintains the operation log, or `Oplog`. This operation log has a set of sequence of the operations that are performed by the primary replica set. This operation log is then used by the secondary replica sets to reproduce the same sequence of operations in order to sync themselves so that they have the replicated (or duplicate) data of the primary replica set.

## **Read operations on sharded clusters**

As we studied earlier in this chapter, in MongoDB, we can distribute the large data into small data using sharding. So, when we divide the large data in smaller subsets using sharding, it is totally transparent to an application.

Whenever we perform sharding, we specify a sharding key for the MongoDB collections so that the data will be handled in an effective manner. So, whenever the clients send any requests, they should include this sharded key so that the data will be fetched directly from the particular shard (subset of the sharded data). If the clients do not send their query with the shard key, then the read operation will take more time since in this operation every shard has to be involved in the query.

In the sharded clusters, the config database is maintained and is very helpful in routing the queries to the shards. So, if any client sends the query with the sharded key, then, by using the metadata from the config database, the queries are redirected or routed to the particular shard.

## **Write operations on sharded clusters**

Whenever there is a write request from the client for the sharded collections in the sharded cluster, the request goes to the particular shard (subset of the sharded data) which is responsible for that particular data set.

Here, the write operation is performed using the metadata which is in config database and based on this metadata, the redirection or routing is done for the write operation to the particular shard in the sharded cluster.

## **Conclusion**

In this chapter, we studied the intermediate concepts like atomicity and atomicity in MongoDB. We also learned about the consistency and consistency in MongoDB. Later in this chapter, we learned about the concept of replication in MongoDB and we its advantages. We also learned about the concept of sharding in MongoDB and its advantages.

In the last part of this chapter, we learned about the distributed operations and queries and how the read and write operations are handled when we use replication and sharding in MongoDB.

In the next chapter of this book, we will cover the concept of index in MongoDB and its usage. We will also cover the default `_id` index and how we can create an index in MongoDB. We will also discuss the various types of indexes in MongoDB and will cover the index properties and how we can use the index in MongoDB.

## **Questions**

1. What is atomicity in database management systems?

2. Explain how MongoDB supports atomicity.
3. What is consistency in database management systems?
4. Explain how MongoDB supports consistency.
5. Explain the concept of replication in MongoDB.
6. Explain the concept of sharding in MongoDB.

# CHAPTER 12

## Introduction to MongoDB Indexes

In the previous chapter, we covered intermediate topics like atomicity and consistency in MongoDB. We also learned about consistency and consistency in MongoDB. We also covered the basic introduction to replication and why it is useful. And finally, we learned about sharding and why it is useful. This chapter covers the concept of indexing. In this chapter, we will start from the introduction to indexes and their benefits wherein we will learn that indexes are special types of data structures in database management systems, like MongoDB, that store the data in an easy-to-traverse form. We will also give an introduction to the MongoDB default `_id` index and learn more about its properties. We will then learn how to create an index in MongoDB. Later in this chapter, we will study the different types of indexes that we can create in MongoDB with step-by-step practical examples. We will also study some of the different index properties that we can use in MongoDB while we create an index in MongoDB. We will also study how we can use indexes with other MongoDB methods. We will also learn about the collation and how we can use it with the MongoDB index. In the last part of this chapter, we will cover how we can view the existing indexes and how we can delete the existing indexes from the MongoDB collection. Additionally, we will also learn about some restrictions in MongoDB indexing.

### Structure

In this chapter, we will discuss the following topics:

- What are indexes?
- Default `_id` index
- Creating an index

- Index types in MongoDB
- Index properties
- Using an index
- Indexes and collation
- Viewing index information
- Deleting an index
- Some restrictions in MongoDB index

## **Objectives**

After studying this unit, you should be able to understand about indexing in MongoDB and also about the default `_id` index in MongoDB. Later in this chapter will learn how to create an index in MongoDB, Understand different types of indexes used in MongoDB, Understand about MongoDB index properties and learn how to use indexes along with some MongoDB methods. We will also learn how to create an index with collation and view existing index information in a MongoDB collection. In the Last Section of this chapter we will learn how to delete the indexes using different MongoDB methods and also understand about some restrictions in MongoDB index.

## **What are indexes?**

Indexes are special types of data structures in database management systems, like MongoDB, that store data in an easy-to-traverse form. These data are related to the MongoDB document's field. In MongoDB, we can create indexes for a single field or on a set of fields. These indexes store the values of these fields in an orderly manner.

## **Indexing and MongoDB**

The following points are specific to the indexing in MongoDB:

- In MongoDB, indexes are defined at the collection level
- MongoDB supports indexing on any field or sub-field of MongoDB collection

We will cover the practical examples in the next sections of this chapter.

## **Benefits of indexing**

Indexing plays an important role in terms of the efficient execution of queries in MongoDB. It improves the performance of the application. If no indexing is done for a collection, then the MongoDB query has to scan the entire documents in a collection to match a query to any document in a collection.

When we define index in the MongoDB collection, the query will limit the scanning of the documents based on the index, and this way, the MongoDB query will not scan the entire documents in the MongoDB collection.

Proper indexing improves the performance and speed of the application running the MongoDB database significantly. We should take care while adding a new index. We should properly plan and check our application queries and analyze whether we really need a new index or not since too many indexes might lead to performance issues in the create, update and delete operations because of the additional data space used by these indexes as well as the additional write operations required by each of these operations due to too many indexes.

## **Default \_id index**

Whenever we create a document in a MongoDB collection, the `_id` index is created automatically with the type `ObjectId`. `ObjectId` is a special MongoDB-specific data type that stores the unique key ID. In MongoDB, every document in a collection has a unique `ObjectId` and every document has the `_id` field.

The size of the `ObjectId` is 12 bytes and it is divided into 4 parts, as shown in the following table:

Part name	Size (bytes)
Timestamp	4

Machine Id	3
Process Id	2
Counter	3

**Table 12.1: Size Division of ObjectId**

## The \_id properties

- In MongoDB, an automatic `_id` index is created in a collection whenever we create a collection.
- In MongoDB, the `_id` field is the first field in MongoDB documents.
- In MongoDB, while creating a document, we can specify the `_id` field and its value.
- In MongoDB, we can have the `_id` field of any BSON data type except the array. The array data type is not a valid `_id` type and is not supported by MongoDB.
- In MongoDB, whenever we define the `_id` type by ourselves and if MongoDB receives any document in which the `_id` is not defined in the beginning, then MongoDB automatically pushes this `_id` at the beginning of the document.

Thus, the `_id` index prevents the insertion of two documents having the same `_id` value. Also, in MongoDB, we cannot drop the `_id` index. Before we start creating an index in our collection, let's populate our new collection with the new data.

In our example, we have created a variable named "`BPBBooksBestSelling EditionsWithIndex`" and the code for the same is as follows:

## Code 1

```
var BPBBooksBestSellingEditionsWithIndex = [ {
    'Title': 'Cloud Computing',
    'Year': '2019',
    'ISBN': '9789388511407',
```

```
'Pages': 330,
'Weight': '570gm',
'Dimension': '23x19x1.5cm',
'Tags': ['Cloud Computing', 'Cloud Computing Concepts', 'Non
Programming'],
'InStock': [{
    'Type': 'Paperback',
    'Quantity': 3000
},
{
    'Type': 'Hardcover',
    'Quantity': 1500
}
],
'SpecialOfferDiscount': '100'
}, {
    'Title': 'Introduction to Digital Marketing 101 : Easy to
Learn and Implement Hands-on Guide for Digital Marketing',
    'Year': '2019',
    'ISBN': '9789389328189',
    'Pages': 464,
    'Tags': ['Digital Marketing', 'Digital Marketing Tips', 'Non
Programming'],
    'InStock': [{
        'Type': 'Paperback',
        'Quantity': 4000
},
{
        'Type': 'Hardcover',
        'Quantity': 2300
}
]
}, {
    'Title': 'IOT and Smart Cities: Your Smart City Planning
Guide',
    'Year': '2019',
    'ISBN': '9789388511322',
```

```
'Pages': 242,
'Weight': '357gm',
'Dimension': '22.5x15x1.5gm',
'Tags': ['Internet of Things', 'IoT', 'Smart City', 'Planning Guide', 'Non Programming'],
'InStock': [{  
    'Type': 'Paperback',  
    'Quantity': 2000  
},  
{  
    'Type': 'Hardcover',  
    'Quantity': 1000  
}  
],  
'SpecialOfferDiscount': '200'  
, {  
    'Title': 'Machine Learning with Python',  
    'Year': '2018',  
    'ISBN': '9789386551931',  
    'Pages': 267,  
    'Tags': ['Python', 'Machine Learning', 'Python Programming',  
    'Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 4500  
    },  
{  
        'Type': 'Hardcover',  
        'Quantity': 1300  
    }  
]  
, {  
    'Title': 'Artificial Intelligence Ethics and International Law: An Introduction',  
    'Year': '2019',  
    'ISBN': ' 9789388511629',  
    'Pages': 188,
```

```
'Weight': '268gm',
'Tags': ['Artificial Intelligence', 'International Law', 'AI',
'Artificial Intelligence Ethics', 'Non Programming'],
'InStock': [{  
    'Type': 'Paperback',  
    'Quantity': 5200  
,  
{  
    'Type': 'Hardcover',  
    'Quantity': 3300  
}  
]  
}, {  
    'Title': 'A Practical Approach for Machine Learning and Deep  
Learning Algorithms',  
    'Year': '2019',  
    'ISBN': '9789388511131',  
    'Pages': 280,  
    'Weight': '424gm',  
    'Tags': ['Machine Learning', 'Deep Learning', 'Algorithms',  
'Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 2800  
,  
{  
        'Type': 'Hardcover',  
        'Quantity': 1250  
}  
,  
    ],  
    'SpecialOfferDiscount': '150'  
}, {  
    '_id': '20021111',  
    'Title': 'Introduction to Database Management',  
    'Year': '2002',  
    'ISBN': ' 9788176566384',  
    'Pages': 342,
```

```

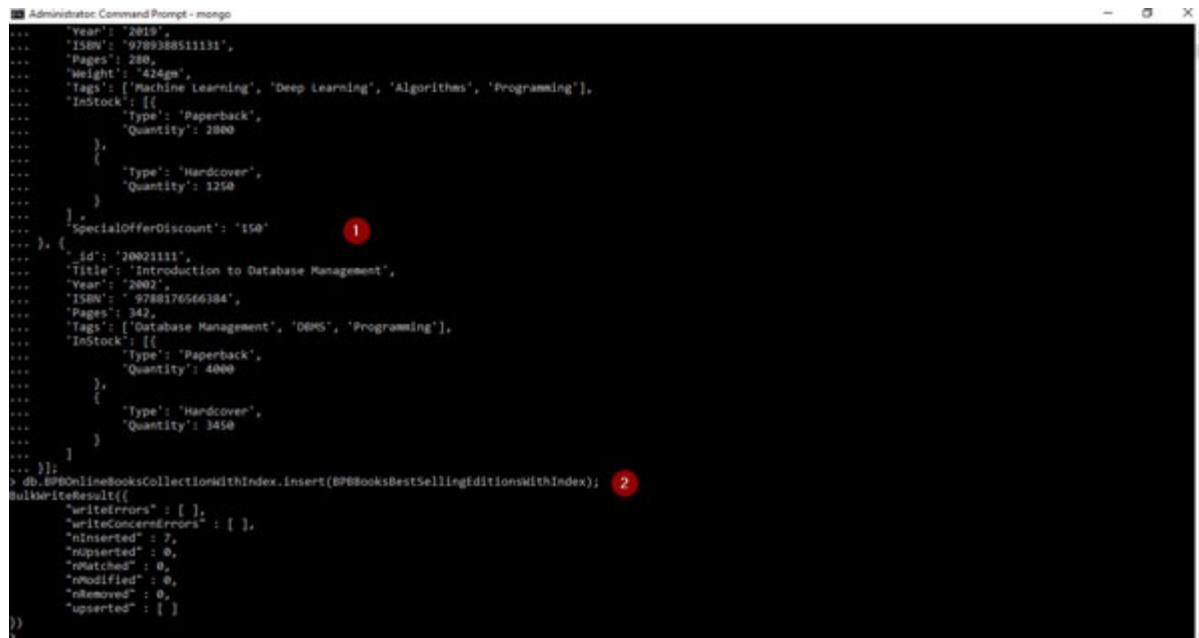
'Tags': ['Database Management', 'DBMS', 'Programming'],
'InStock': [
    {
        'Type': 'Paperback',
        'Quantity': 4000
    },
    {
        'Type': 'Hardcover',
        'Quantity': 3450
    }
]
} ];

```

And then, we have used the MongoDB `insert()` method to create a new documents in the MongoDB collection, `BPBOnlineBooksCollectionWithIndex`. The code for the same is shown in the following figure:

## Code 2

```
db.BPBOnlineBooksCollectionWithIndex.insert(BPBBooksBestSelling
EditionsWithIndex);
```



The screenshot shows a terminal window titled "Administrator: Command Prompt - mongo". It displays a MongoDB shell session. The user has defined a document with two books and then inserted it into the collection. The document structure is as follows:

```

{
    "Year": "2019",
    "ISBN": "9789388511131",
    "Pages": 288,
    "Weight": "424gm",
    "Tags": ["Machine Learning", "Deep Learning", "Algorithms", "Programming"],
    "InStock": [
        {
            "Type": "Paperback",
            "Quantity": 2880
        },
        {
            "Type": "Hardcover",
            "Quantity": 1250
        }
    ],
    "SpecialOfferDiscount": "150"
},
{
    "_id": "20021111",
    "Title": "Introduction to Database Management",
    "Year": "2002",
    "ISBN": "9788176566384",
    "Pages": 342,
    "Tags": ["Database Management", "DBMS", "Programming"],
    "InStock": [
        {
            "Type": "Paperback",
            "Quantity": 4000
        },
        {
            "Type": "Hardcover",
            "Quantity": 3450
        }
    ]
}
]);

```

After defining the document, the user runs the command `db.BPBOnlineBooksCollectionWithIndex.insert(BPBBooksBestSellingEditionsWithIndex)`. The response shows the result of the insert operation, which includes metrics such as write errors, write concern errors, and the number of inserted documents.

```

> db.BPBOnlineBooksCollectionWithIndex.insert(BPBBooksBestSellingEditionsWithIndex);
{
    "writeErrors": [ ],
    "writeConcernErrors": [ ],
    "nInserted": 2,
    "nUpserted": 0,
    "nMatched": 0,
    "nModified": 0,
    "nRemoved": 0,
    "upserted": [ ]
}

```

**Figure 12.1: Creating a new Collection and Inserting a new Data**

Since we are done creating a new collection and inserting new documents in it, let us now create an index to our new collection.

## Creating an index

To create an index in MongoDB, we use the following method:

- db.collection.createIndex()

Let us study this method in detail.

### db.collection.createIndex() method

This method creates one or more indexes in the MongoDB collection based on the key and value.

#### Method definition

```
db.collection.createIndex(<key and index type>, <options>)
```

In this method:

- Key is the name of the field in MongoDB document
- Index type is the value of the index which is generally 1 or -1. 1 specifies the ascending index in the field while -1 specifies the descending index in the field.
- Options are optional and these are related to the index properties which we will cover later in this chapter.

Therefore, in order to create an index for MongoDB collection, we use the `createIndex()` method in the following manner:

```
db.collection.createIndex({key: value})
```

### Example – Creating an index in MongoDB collection

In our example, we have created an index in our collection "BPBOnline BooksCollectionWithIndex" in the field "Title" in the ascending order. The code for the same is shown in the following figure:

## Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex({Title : 1});
```

A screenshot of the MongoDB shell interface. The command `db.BPBOnlineBooksCollectionWithIndex.createIndex({Title : 1});` is entered. The response shows the index details: "createdCollectionAutomatically": false, "nIndexesBefore": 1, "nIndexesAfter": 2, and "ok": 1. A red circle highlights the number 1 in the "nIndexesAfter" field.

**Figure 12.2:** Creating an Index in a MongoDB Collection using `createIndex()` Method

In the preceding example, we saw how to create an index using the MongoDB `createIndex()` method.

## Index types in MongoDB

MongoDB supports many types of indexes. These are as follows:

### Single field index

These are used to create an index on a single field and can be user-defined.

## Example – Creating a single field index in MongoDB collection

In our example, we have created an index in our collection "BPBOnlineBooksCollectionWithIndex" in the field "Year" in the ascending order. The code for the same is shown in the following figure:

## Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex({Year : 1});
```

A screenshot of the MongoDB shell interface. The command `db.BPBOnlineBooksCollectionWithIndex.createIndex({Year : 1});` is entered. The response shows the index details: "createdCollectionAutomatically": false, "nIndexesBefore": 2, "nIndexesAfter": 3, and "ok": 1. A red circle highlights the number 1 in the "nIndexesAfter" field.

**Figure 12.3: Creating a Single Field Index in a MongoDB Collection using `createIndex()` Method**

In the preceding example, we saw how to create a single field index using the MongoDB `createIndex()` method.

## Compound Index

These are user-defined indexes on multiple fields.

## Example – Creating a compound index in MongoDB collection

In our example, we have created a compound index in our collection "BPBOnlineBooksCollectionWithIndex" in the field "Title" in the ascending order and "Year" in the descending order. The code for the same is shown in the following figure:

## Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex({Title : 1,  
Year : -1});
```



**Figure 12.4: Creating a Compound Index in a MongoDB Collection using `createIndex()` Method**

In the preceding example, we saw how to create a compound index using the MongoDB `createIndex()` method.

## Multikey index

MongoDB creates multikey indexes to store arrays. Here, MongoDB creates a separate index for each element of an array and automatically identifies where to create a multikey index if an index contains an array.

## Example – Creating a multikey index in a MongoDB collection

In our example, we have created a multikey index in our collection `BPBOnlineBooksCollectionWithIndex` in the field `InStock.Type` in the ascending order and `InStock.Quantity`, also in the ascending order. The code for the same is shown in the following figure:

### Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex({ "InStock.Type"
" : 1, "InStock.Quantity" : 1});
```



**Figure 12.5: Creating a Multi Key Index in a MongoDB Collection using `createIndex()` Method**

In the preceding example, we saw how to create a multikey index using the MongoDB `createIndex()` method.

### Text index

Text indexes are useful where we want to search and query the text related data in the collection. Text Indexes prevents some language-specific stop words. Following are some of the stop words in English:

- a
- an
- the
- or
- and

## Example – Creating a text index in a MongoDB collection

In our example, we have created a text key index in our collection "BPBOnlineBooksCollectionWithIndex" in the field "Title" with the index type as text. The code for the same is shown in the following figure:

## Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex({Title:  
"text"});
```



**Figure 12.6:** Creating a Text Index in a MongoDB Collection using `createIndex()` Method

In the preceding example, we saw how to create a text index using the MongoDB `createIndex()` method.

## Special types of index

MongoDB supports few special types of indexes which are as follows:

### Geospatial index

These types of indexes are used to support the queries related to the data that are geospatial in nature and are usually in the form of coordinates.

### Hashed index

Hashed indexes are used in sharding. Hashed indexes support sharding using the hashed shard keys. We will learn more about sharding in the advanced chapters of this book.

### Index properties

In MongoDB, indexes have some properties which we can use while creating one.

## Unique index

As the name suggests, if we apply this unique property of the index while creating one for a document field, it will only allow to have unique value for that field and discard any duplicate value.

## Example – Creating a unique index in a MongoDB Collection

In our example, we have created a unique index in our collection "BPBOnlineBooksCollectionWithIndex" in the field "ISBN" in the ascending order with "unique" values set to true. The code for the same is as follows:

### Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex(  
  {ISBN: 1},  
  {unique: true}  
) ;
```

## Partial index

The partial index property is very useful when we want to index documents based on some criteria. So, the partial index is applied to a document if it meets the specific filter criteria. If we create an index with some conditions, it is a partial index. The partial index takes few resources in terms of space and performance costs.

## Example – Creating a partial index in a MongoDB Collection

In our example, we have created a partial index in our collection "BPBOnlineBooksCollectionWithIndex" in the field "Title" in the

descending order by giving 2019 as the partial index for "Year". The code for the same is shown in the following figure:

## Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex(  
  {Title: -1},  
  {partialFilterExpression: {Year: "2019"} }  
) ;
```

A screenshot of a terminal window titled "Administrator Command Prompt - mongo". The command entered is "db.BPBOnlineBooksCollectionWithIndex.createIndex({Title: -1}, {partialFilterExpression: {Year: "2019"} }) ;". A red circle with the number "1" is drawn around the command line. The response shows the index creation details: "createdCollectionAutomatically": false, "numIndexesBefore": 6, "numIndexesAfter": 7, "ok": 1.

**Figure 12.7: Creating a Partial Index in a MongoDB Collection using `createIndex()` Method**

In the preceding example, we saw how to create a partial index using the MongoDB `createIndex()` method.

## Sparse index

This is a very interesting property of a MongoDB index. This property ensures that any new document being inserted into the MongoDB collection gets indexed only when it contains an indexed field, otherwise, not.

## Example – Creating a sparse index in a MongoDB collection

In our example, we have created a sparse index in our collection "BPBOnlineBooksCollectionWithIndex" in the field "ISBN" in the ascending order along with "sparse" value set to true. The code for the same is shown in the following figure:

## Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex(
```

```
{ISBN: 1},  
{sparse: true}  
);
```



A screenshot of the MongoDB Command Prompt window titled "Administrator: Command Prompt - mongo". The command entered is:

```
db.BPBOnlineBooksCollectionWithIndex.createIndex(  
  {ISBN: 1},  
  {sparse: true}  
)
```

The response shows the index was created successfully:

```
"createdCollectionAutomatically": false,  
"numIndexesBefore": 7,  
"numIndexesAfter": 8,  
"ok": 1
```

**Figure 12.8: Creating a Sparse Index in a MongoDB Collection using `createIndex()` Method**

In the preceding example, we saw how to create a sparse index using the MongoDB `createIndex()` method.

## TTL index

This is another very interesting property of a MongoDB index and is very useful. **TTL** means "**time to live**", which is in seconds. If we apply the TTL index to any document, these documents will be removed from the MongoDB collection after the defined specific time while applying this index property.

This type of index is very useful in the data management of certain type, like machine logs, events data, session based information, etc.

## Example – Creating a TTL index in a MongoDB collection

In our example, we have created a TTL index in our collection `BPBOnlineBooksCollectionWithIndex` in the `SpecialOfferDiscount` field in the ascending order along with the TTL parameter `expireAfterSeconds` value set to 604800 (equals to 7 days). The code for the same is shown in the following figure:

## Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex(  
  {SpecialOfferDiscount: 1},
```

```
{expireAfterSeconds: 604800}  
};
```



```
Administrator: Command Prompt - mongo  
db.BPBOnlineBooksCollectionWithIndex.createIndex(  
... {SpecialOfferDiscount: 1},  
... {expireAfterSeconds: 604800} !  
... );  
{  
    "createdCollectionAutomatically" : false,  
    "numIndexesBefore" : 8,  
    "numIndexesAfter" : 9,  
    "ok" : 1  
}
```

**Figure 12.9:** Creating a TTL Index in a MongoDB Collection using `createIndex()` Method

In the preceding example, we saw how to create a TTL index using the MongoDB `createIndex()` method.

## Using an index

We can use an index after we create it on any field. The easiest way to use an index is to use the MongoDB `find()` method and to query the database with the indexed fields.

## Example – Creating and using an index in a MongoDB collection

In our example, we have created an index in our collection "BPBOnlineBooksCollectionWithIndex" in the field "Title" in the ascending order and "Year", also in the ascending order. The codes for the same are shown in the following figure:

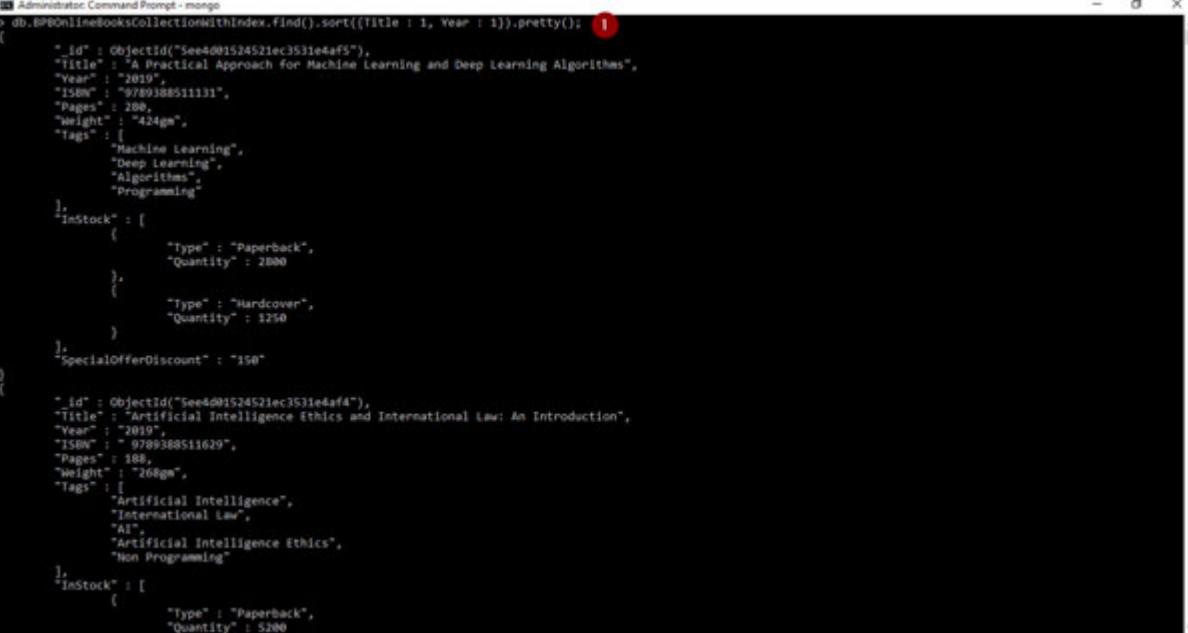
### Code 1

```
db.BPBOnlineBooksCollectionWithIndex.createIndex({Title : 1,  
Year : 1});
```

Then using the `sort()` method, we can easily sort the indexed fields in the MongoDB collection.

### Code 2

```
db.BPBOnlineBooksCollectionWithIndex.find().sort({Title : 1,  
Year : 1}).pretty();
```



```
Administrator Command Prompt - mongo
> db.BPBooksCollectionWithIndex.find().sort({title: 1, Year: 1}).pretty();
{
  "_id": ObjectId("5e4d01524521ec3531e4af5"),
  "title": "A Practical Approach for Machine Learning and Deep Learning Algorithms",
  "Year": 2019,
  "ISBN": "9789388511131",
  "Pages": 288,
  "Weight": "424gm",
  "Tags": [
    "Machine Learning",
    "Deep Learning",
    "Algorithms",
    "Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 2800
    },
    {
      "Type": "Hardcover",
      "Quantity": 1250
    }
  ],
  "SpecialOfferDiscount": "150"
},
{
  "_id": ObjectId("5e4d01524521ec3531e4af4"),
  "title": "Artificial Intelligence Ethics and International Law: An Introduction",
  "Year": 2019,
  "ISBN": "9789388511629",
  "Pages": 188,
  "Weight": "208gm",
  "Tags": [
    "Artificial Intelligence",
    "International Law",
    "AI",
    "Artificial Intelligence Ethics",
    "Non Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 5200
    }
  ]
}
```

*Figure 12.10: Using an Index in a MongoDB Collection*

In the preceding example, we saw how we can use an index in the MongoDB `sort()` method.

## Indexes and collation

Sometimes it is useful to create an index that is helpful to perform the language-specific tasks. Some language contains some special characters and during the search queries, they play significant roles in terms of searching the right set of data strings. So, whenever we have a language-specific need, we can create an index with the collation property of the index.

Following are the fields of the collation property:

```
collation: {
  locale: <string>,
  caseLevel: <boolean>,
  caseFirst: <string>,
  strength: <int>,
  numericOrdering: <boolean>,
  alternate: <string>,
  maxVariable: <string>,
  backwards: <boolean>
```

```
}
```

In MongoDB, if you create an index with the collation property, you must specify the "locale" field as it is mandatory. Rest all the other collation fields are optional.

## [Example – Creating an index with collation in a MongoDB Collection](#)

In our example, we have created an index in our collection "BPBOnlineBooksCollectionWithIndex" in the field "Title" in the ascending order and "Tags" too in the ascending order along with the collation property of the index and the locale set to "en\_US", which is for English (United States). The code for the same is shown in the following figure:

### [Code 1](#)

```
db.BPBOnlineBooksCollectionWithIndex.createIndex({Title : 1,  
Tags: 1}, {collation: {locale: "en_US"}});
```

A screenshot of an Administrator Command Prompt window titled "Administrator: Command Prompt - mongo". The window contains the following command and its output:

```
db.BPBOnlineBooksCollectionWithIndex.createIndex({Title : 1, Tags: 1}, {collation: {locale: "en_US"}});
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 10,
  "numIndexesAfter" : 11,
  "ok" : 1
}
```

The output is highlighted with a red oval around the "ok" field.

**Figure 12.11: Creating an Index with Collation in a MongoDB Collection using `createIndex()` Method**

In the preceding example, we saw how to create an index with the collation using the MongoDB `createIndex()` method.

### [View index information](#)

Sometimes we need information about the indexes that are already created in the collection. In this case, we can get the list of the index created in the collection by using the following method:

- `db.collection.getIndexes()`

## db.collection.getIndexes() method

This method returns the array of all the indexes with their details that exist in the collection.

## Example – Viewing all the Indexes in a MongoDB Collection

In our example, we used the `getIndexes()` method in our collection "BPBOnlineBooksCollectionWithIndex" to view all the indexes that exist in our collection. The code for the same is shown in the following figure:

### Code 1

```
db.BPBOnlineBooksCollectionWithIndex.getIndexes();
```

```
Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollectionWithIndex.getIndexes();
[{"v": 2, "key": {"_id": 1}, "name": "_id", "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollectionWithIndex"}, {"v": 2, "key": {"title": 1}, "name": "title_1", "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollectionWithIndex"}, {"v": 2, "key": {"Year": 1}, "name": "Year_1", "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollectionWithIndex"}, {"v": 2, "key": {"title": 1, "Year": -1}, "name": "Title_1_Year_-1", "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollectionWithIndex"}, {"v": 2, "key": {"InStock.type": 1, "InStock.quantity": 1}, "name": "InStock_Type_1_InStock_Quantity_1", "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollectionWithIndex"}]
```

**Figure 12.12: Viewing all the Indexes in a MongoDB Collection using `getIndexes()` Method – Screen 1**



```
Administrator: Command Prompt - mongo
{
  "name": "ISBN_1",
  "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollectionWithIndex",
  "sparse": true
},
{
  "v": 2,
  "key": {
    "SpecialOfferDiscount": 1
  },
  "name": "SpecialOfferDiscount_1",
  "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollectionWithIndex",
  "expireAfterSeconds": 604800
},
{
  "v": 2,
  "key": {
    "Title": 1,
    "Year": 1
  },
  "name": "Title_1_Year_1",
  "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollectionWithIndex"
},
{
  "v": 2,
  "key": {
    "Title": 1,
    "Tags": 1
  },
  "name": "Title_1_Tags_1",
  "ns": "BPBOnlineBooksDB.BPBOnlineBooksCollectionWithIndex",
  "collation": {
    "locale": "en_US",
    "caseLevel": false,
    "caseFirst": "off",
    "strength": 3,
    "numericOrdering": false,
    "alternate": "non-ignorable",
    "maxVariable": "punct",
    "normalization": false,
    "backwards": false,
    "version": "57.1"
  }
}
```

**Figure 12.13:** Viewing all the Indexes in a MongoDB Collection using `getIndexes()` Method – Screen 2

In the preceding example, we saw how to view all the indexes in the MongoDB collection using the MongoDB `getIndexes()` method.

## Deleting an index

Sometimes we need to remove one or more indexes from the MongoDB collection, or, in some cases, we need to delete all the indexes from the MongoDB collection. In these cases, we can use the following methods:

- `db.collection.dropIndex()`
- `db.collection.dropIndexes()`

Note that we cannot remove the default `_id` index with any of these methods.

## [db.collection.dropIndex\(\) method](#)

This method deletes the index from the collection.

## Method definition

```
db.collection.dropIndex(index)
```

Here, the index can be one of the following:

- Name of the index
- Index specification document

## **Example 1 – Deleting an index in a MongoDB collection**

In the earlier section, we used the `getIndexes()` method in our collection "BPBOnlineBooksCollectionWithIndex" to view all the indexes that exist in our collection, as earlier shown in [figure 12.12](#) and [figure 12.13](#). If there is a name and key listed, these two values can be used to remove an index from the MongoDB collection.

Now, let us remove the index having the "Index Specification Document" value equal to `{Title:1}`. The code for the same is shown in the following figure:

### **Code 1**

```
db.BPBOnlineBooksCollectionWithIndex.dropIndex({Title:1});
```



*Figure 12.14: Deleting an Index in a MongoDB Collection using `dropIndex()` Method*

In the preceding example, we saw how to delete an index in a MongoDB collection using the MongoDB `dropIndex()` method.

### **db.collection.dropIndexes() method**

This method deletes the index from the collection.

### **Method definition**

```
db.collection.dropIndex(indexes)
```

Here, the indexes' parameter is optional but it has to be one of the following:

- Name of the index
- Index specification document
- Names of the indexes in an array

If there is no index value specified in the parameter of the method, it will remove all the indexes except the default `_id` index from the MongoDB collection.

## [Example 2 – Deleting multiple index in a MongoDB collection](#)

Now, let us remove multiple indexes having "Index Specification Document" values equal to `{Title:1, Year:1}`. The code for the same is shown in the following figure:

### [Code 1](#)

```
db.BPBOnlineBooksCollectionWithIndex.dropIndexes({Title:1,  
Year:1});
```



*Figure 12.15: Deleting Multiple Index in a MongoDB Collection using dropIndexes() Method*

In the preceding example, we saw how to delete multiple indexes in a MongoDB collection using the MongoDB `dropIndexes()` method.

## [Example 3 – Deleting multiple index in a MongoDB collection using the array type values as parameter](#)

Now, let us remove multiple indexes having "Name" values given in an array `["Title_1_Tags_1", "ISBN_1", "SpecialOfferDiscount_1"]`.

The code for the same is shown in the following figure:

## **Code 1**

```
db.BPBOnlineBooksCollectionWithIndex.dropIndexes(["Title_1_Tags  
_1", "ISBN_1", "SpecialOfferDiscount_1"]);
```



**Figure 12.16: Deleting Multiple Index in a MongoDB Collection using Array TypeValues as Parameter and dropIndexes() Method**

In the preceding example, we saw how to delete multiple indexes in a MongoDB collection using the array type values as parameter and the `dropIndexes()` method.

## **Example 4 – Deleting all the indexes in a MongoDB collection**

To delete all the indexes from the collection, we can use the `dropIndexes()` method without passing any parameter value. So, let us now remove all the indexes from the MongoDB collection. The code for the same is shown in the following figure:

## **Code 1**

```
db.BPBOnlineBooksCollectionWithIndex.dropIndexes();
```



**Figure 12.17: Deleting All Indexes in a MongoDB Collection using dropIndexes() Method**

In the preceding example, we saw how to delete all the indexes in a MongoDB collection using the `dropIndexes()` method.

## Some restrictions in MongoDB index

When we do indexing, it takes up the memory space during certain operations, such as insert, update, etc. Following are some of the restrictions in MongoDB indexing:

- In case you don't need to read the collections frequently, try to avoid indexing
- As the MongoDB index uses memory space (RAM), it is recommended that indexing does not exceed the memory limits
- If the index exceeds the memory limits, then MongoDB can remove some of the indexes that can lead to performance issues
- The index name should be less than or equal to 164 characters
- A MongoDB collection cannot have index more than 64
- A compound index cannot have more than 31 fields

## Conclusion

In this chapter, we studied the concept of indexing in a detailed manner. We also learned about the indexes and their benefits. We also studied about the MongoDB default `_id` index and its properties. We learned how to create an index in MongoDB and about the different types of indexes that can be created with the step-by-step practical examples. Further in this chapter, we studied some of the index properties used in MongoDB to create an index. We also studied how we can use indexes with other MongoDB methods. Additionally, we also learned about the collation and how we can use it with the MongoDB index. In the last part of this chapter, we learned how to view and delete the existing indexes from the MongoDB collection. Additionally, we also learned about some of the restrictions in the MongoDB indexing.

In the next chapter, we will cover the data modeling in MongoDB where we will cover the topics like introduction to data modeling, data modeling concepts and design, data model examples, data model patterns, and model relationships between documents.

## Points to Remember

- Proper indexing improves the performance and speed of the application running the MongoDB database significantly.
- Whenever we create a document in a MongoDB collection, the `_id` Index is created automatically with the type `ObjectId`.
- To create an index in MongoDB, we use the method:  
`db.collection.createIndex()`
- If we apply TTL index to the documents, these documents will be removed from the MongoDB collection after the defined specific time while applying the index property. This type of index is very useful in the data management of certain type like machine logs, events data, session based information, etc.
- Whenever we have language-specific needs, we can create an index with the collation property of the index.
- Sometimes we need to remove one or more indexes from the MongoDB collection, or, sometimes, in some cases, we need to delete all the indexes from the MongoDB collection. In these cases, the few methods which we can use are  
`db.collection.dropIndex()` and `db.collection.dropIndexes()`

## Multiple choice questions

1. **The default index `_id` is of the type:**
  - a. String
  - b. Integer
  - c. ObjectId
  - d. None of these
2. **To create an index in MongoDB, which of the following methods is used:**
  - a. `db.collection.addIndex()`
  - b. `db.collection.createIndex()`
  - c. `db.collection.Index()`

- d.** `db.collection.newIndex()`
3. To create an index with the collation property in MongoDB, which of the following methods is used:
- `db.collection.Index({Field1 : 1, Field2: 1}, {collation: {locale: "en_US"}});`
  - `db.collection.createIndex({Field1 : 1, Field2: 1}, {collation: 1});`
  - `db.collection.createIndex({Field1 : 1, Field2: 1});`
  - `db.collection.createIndex({Field1 : 1, Field2: 1}, {collation: {locale: "en_US"}});`
4. To delete all the indexes from the collection, we can use:
- `db.collection.dropIndexes();`
  - `db.collection.deleteIndex();`
  - `db.collection.createIndex({Field1 : -1, Field2: -1});`
  - `db.collection.noIndex();`

## Answer

1. **c**
2. **b**
3. **d**
4. **a**

## Questions

1. What is indexing and why is it useful?
2. Explain the MongoDB default `_id` index with some of its properties?
3. How can we create an index in MongoDB?
4. List three different types of MongoDB index.
5. List two MongoDB index properties.

6. How can we view the existing indexes in a MongoDB collection?
7. List the various methods to delete an index in a MongoDB collection.

## Key terms

- **Indexes:** These are the special types of data structures in database management systems, like MongoDB, that store the data in an easy-to-traverse form.
- **Default \_id index:** Whenever we create a document in a MongoDB collection, the `_id` index is created automatically with the type `ObjectId`.
- **TTL:** It means "time to live", which is in seconds.

# CHAPTER 13

## MongoDB Query Selectors

This chapter covers the MongoDB query selectors. We will start with the basic introduction to the query selectors and why they are useful. Further, we will cover different types of query selectors available in MongoDB and then we will move on to cover these query selectors with step-by-step practical examples..

### Structure

In this chapter, we will discuss the following topics:

- Introduction to query selectors
- Examples and use of query selectors

### Objectives

After studying this unit, you should be able to:

- Understand query selectors in MongoDB
- Understand different types of query selectors
- Use query selectors with practical examples
- Use projection operators with practical examples

### Introduction to query selectors

Whenever we query a MongoDB database, it fetches some data stored in the MongoDB documents from various MongoDB collections. There are different requirements of data throughout the application. Sometimes, we need to use some small portion of data for a specific action in the application, sometimes, we need to update this data based on some criteria, and sometimes, we need to delete some old documents based on some condition. In these situations,

we need something which will help us to work on only those documents which are needed for CRUD operations.

So, the query selectors are helpful for the following:

- Fetch the MongoDB documents from the collections based on some conditions
- Modify the multiple documents based on some conditions
- Delete the multiple documents based on some conditions

There are various types of query selectors we can use based on our requirements. In MongoDB, these query selectors are divided into the following types:

- Comparison Selectors
- Logical Selectors
- Element Selectors
- Array Selectors
- Evaluation Selectors
- Geospatial Selectors
- Bitwise Selectors
- Comment Selector

## Comparison Selectors

These types of selectors are helpful to perform the CRUD operations based on the comparison. The following table has the list of comparison selectors:

Selector	Selector Description	Selector Use and Syntax
\$eq	This selector matches the documents that have values equal to the specified value	{<document field> : {\$eq : <value to match>}}
\$gt	This selector matches the documents that have values greater than the specified value	{<document field> : {\$gt : <value to match>}}

\$gte	This selector matches the documents that have values greater than or equal to the specified value	{<document field> : {\$gte : <value to match>} }
\$in	This selector matches the documents that have any of the values specified in an array	{<document field> : {\$in : [<array value1>, <array value2>, ... <array valueN>]} }
\$lt	This selector matches the documents that have values less than the specified value	{<document field> : {\$lt : <value to match>} }
\$lte	This selector matches the documents that have values less than or equal to the specified value	{<document field> : {\$lte : <value to match>} }
\$ne	This selector matches the documents that have values not equal to the specified value	{<document field> : {\$ne : <value to match>} }
\$nin	This selector matches the documents that have none of the values specified in an array	{<document field> : {\$nin : [<array value1>, <array value2>, ... <array valueN>]} }

**Table 13.1: Comparison Selectors**

## Logical Selectors

These types of selectors are helpful to perform the CRUD operations based on the logical conditions. The following table has the list of logical selectors:

Selector	Description	Selector Use and Syntax
\$and	This selector performs the logical AND operation on different expressions and joins both the queries to deliver the combined result by returning all the documents based on the Join	se{\$and : [{<expression1>}, {<expression2>}, ..., {<expressionN>}]} }
\$not	This selector performs the	{<document field> : {\$not :

	logical NOT operation and returns the documents that do not match the expression	{<expression>} }
\$or	This selector performs the logical OR operation on different expressions and joins both the queries to deliver the combined result by returning all the documents based on the Join	{\$or : [{<expression1>}, {<expression2>}, ..., {<expressionN>}]} }
\$nor	This selector performs the logical NOR operation on different expressions and selects all the documents that fail all the query expressions	{\$nor: [{<expression1>}, {<expression2>}, ..., {<expressionN>}]} }

**Table 13.2: Logical Selectors**

## Element Selectors

These types of selectors are helpful to perform the CRUD operations based on the element-based conditions. The following table has the list of element selectors:

Selector	Description	Selector Use and Syntax
\$exists	This selector matches the documents that have the specified field in the documents. It accepts the Boolean value. If the value is set to "true", it returns all the documents that have the specified field. If the value is set to "false", it returns all the documents that do not contain the specified field.	{<document field> : {\$exists: <boolean>}} }
\$type	This selector matches the documents that have the specified field with a BSON type in the documents. We can	{< document field> : {\$type: <BSON type>}} }

	query with BSON type to get the results according to the BSON-type field in the documents.
--	--

**Table 13.3: Element Selectors**

## Array Selectors

These types of selectors are helpful to perform the CRUD operations based on the array fields conditions. The following table has the list of array selectors:

Selector	Description	Selector Use and Syntax
\$all	This selector matches the documents that contain all the elements in an array type field of a document that contain all the elements specified in the query.	{<document field> : {\$all : [<array value1>, <array value2> ... <array valueN>]} }
\$elemMatch	This selector matches all the documents based on the array type field in the document. It selects the documents if at least one element in the document's array field matches all the specified conditions given in the query criteria to match the elements.	{<document field> : {\$elemMatch : {<query1>, <query2>, ... <queryN>} } }
\$size	This selector selects all the documents if the array field in the document is of a specific size given in the query.	{<document field> : {\$size: <number>} }

**Table 13.4: Array Selectors**

## Evaluation Selectors

These types of selectors are helpful to perform the evaluation operations based on the expressions. The following table has the list

of evaluation selectors:

<b>Selector</b>	<b>Description</b>	<b>Selector Use and Syntax</b>
\$expr	This selector allows the use of the aggregation expressions. We will learn more about aggregation in the next chapter of this book.	<code>{\$expr : &lt;expression&gt;}}</code>
\$jsonSchema	This selector helps to validate the documents with the given JSON schema. The schema will match all the documents that satisfy the valid schema	<code>{\$jsonSchema : &lt;JSON Schema object&gt;}</code>
\$mod	This selector performs the modulo operation on the value of a field and selects the documents with a specified result.	<code>{&lt;document field&gt; : {\$mod: [divisor, remainder]}}</code>
\$regex	This selector selects all the documents where values are matched with the specified regular expression. We can use various ways (as shown in the selector use and syntax) to use this selector.	<code>{&lt;document field&gt; : {\$regex: /pattern/, \$options: '&lt;options&gt;'}}</code> <code>{&lt;document field&gt; : {\$regex: 'pattern', \$options: '&lt;options&gt;'}}</code> <code>{&lt;document field&gt; : {\$regex: /pattern/&lt;options&gt;}}</code>
\$text	This selector performs the text search and selects all the documents with the searched terms. Here, in the searched terms, we can search it using various ways like "single word search", "multiple word search", "complete phrase", "excludes documents based on the search term"	<code>\$text : {\$search : "&lt;search terms&gt;"}</code>
\$where	This selector performs the matches on the documents that satisfy a JavaScript	<code>\$where : function() { }</code>

expression. We can pass the JavaScript expression or the full JavaScript function to query the system. This selector gives us lot of flexibility in terms of querying the database, but, at the same time, it requires the database to process each document with the JavaScript expression or function.

*Table 13.5: Evaluation Selectors*

## Geospatial Selectors

These types of selectors are used to perform operations based on the data which is geographical in nature while we select the documents from the database, data like coordinates, address, city, or ZIP code. GIS data are some examples of the data formats which are geospatial in nature.

## Bitwise Selectors

These types of selectors are used to perform bitwise operations while we select the documents from the database. The bitwise operations are done at a bit-level, which is the smallest unit of data in the computer system. While we compare the bitwise data, the document field against which we are comparing the data should be numeric type or `BinData` Type (binary data type).

## Comment Selector

The comment selector has only one type of query operator, `$comment`. It helps to associate comments to the expression while querying. These comments are propagated to the MongoDB profile log.

You might know that profiling is the process to measure and collect the information about the code, program, or, in our case, the database-related queries and operations. Generally, profiling is done

to capture various information, such as the query execution time, errors, etc. All this information is stored in the form of logs. These profile logs analyze the application and make it more optimized by performing the required changes after analyzing these profile logs.

In MongoDB, these profile logs are generated by the MongoDB database profiler that collects a detailed information on all the database commands, such as the CRUD operations, configuration commands, as well as the administrative commands. The advantage of using the comment selector is that it makes our profile data easier to understand and therefore, helps in the better tracing of the profile log data.

## **Selector Use and Syntax**

```
{<query>, $comment : <comment>}
```

## **Examples and use of query selectors**

In the previous section, we studied about the different types of query selectors. Now, let us see how we can use them with some practical examples. Before we start using the query selectors, let's populate our new collection with the new data.

In our example, we have created a variable named "BPBBooksBestSellingEditionsQuerySelectors". The code for the same is as follows:

## **Code 1**

```
var BPBBooksBestSellingEditionsQuerySelectors = [ {  
    'Title': 'Cloud Computing',  
    'Year': '2019',  
    'ISBN': '9789388511407',  
    'Pages': 330,  
    'Weight': '570gm',  
    'Dimension': '23x19x1.5cm',  
    'Tags': ['Cloud Computing', 'Cloud Computing Concepts', 'Non  
    Programming'],  
    'InStock': [{
```

```
'Type': 'Paperback',
'Quantity': 3000
},
{
'Type': 'Hardcover',
'Quantity': 1500
},
],
'SpecialOfferDiscount': '100'
}, {
'Title': 'Introduction to Digital Marketing 101 : Easy to Learn and Implement Hands-on Guide for Digital Marketing',
'Year': '2019',
'ISBN': '9789389328189',
'Pages': 464,
'Tags': ['Digital Marketing', 'Digital Marketing Tips', 'Non Programming'],
'InStock': [
{
'Type': 'Paperback',
'Quantity': 4000
},
{
'Type': 'Hardcover',
'Quantity': 2300
}
]
},
{
'Title': 'IOT and Smart Cities: Your Smart City Planning Guide',
'Year': '2019',
'ISBN': '9789388511322',
'Pages': 242,
'Weight': '357gm',
'Dimension': '22.5x15x1.5gm',
'Tags': ['Internet of Things', 'IoT', 'Smart City', 'Planning Guide', 'Non Programming'],
'InStock': [

```

```
'Type': 'Paperback',
'Quantity': 2000
},
{
'Type': 'Hardcover',
'Quantity': 1000
}
],
'SpecialOfferDiscount': '200'
}, {
'Title': 'Machine Learning with Python',
'Year': '2018',
'ISBN': '9789386551931',
'Pages': 267,
'Tags': ['Python', 'Machine Learning', 'Python Programming',
'Programming'],
'InStock': [
{
'Type': 'Paperback',
'Quantity': 4500
},
{
'Type': 'Hardcover',
'Quantity': 1300
}
]
},
{
'Title': 'Artificial Intelligence Ethics and International
Law: An Introduction',
'Year': '2019',
'ISBN': ' 9789388511629',
'Pages': 188,
'Weight': '268gm',
'Tags': ['Artificial Intelligence', 'International Law', 'AI',
'Artificial Intelligence Ethics', 'Non Programming'],
'InStock': [
{
'Type': 'Paperback',
'Quantity': 5200
}
]
```

```
        },
        {
            'Type': 'Hardcover',
            'Quantity': 3300
        }
    ]
},
{
    'Title': 'A Practical Approach for Machine Learning and Deep
Learning Algorithms',
    'Year': '2019',
    'ISBN': '9789388511131',
    'Pages': 280,
    'Weight': '424gm',
    'Tags': ['Machine Learning', 'Deep Learning', 'Algorithms',
    'Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 2800
        },
        {
            'Type': 'Hardcover',
            'Quantity': 1250
        }
    ],
    'SpecialOfferDiscount': '150'
},
{
    '_id': '20021111',
    'Title': 'Introduction to Database Management',
    'Year': '2002',
    'ISBN': ' 9788176566384',
    'Pages': 342,
    'Tags': ['Database Management', 'DBMS', 'Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 4000
        },
        {

```

```

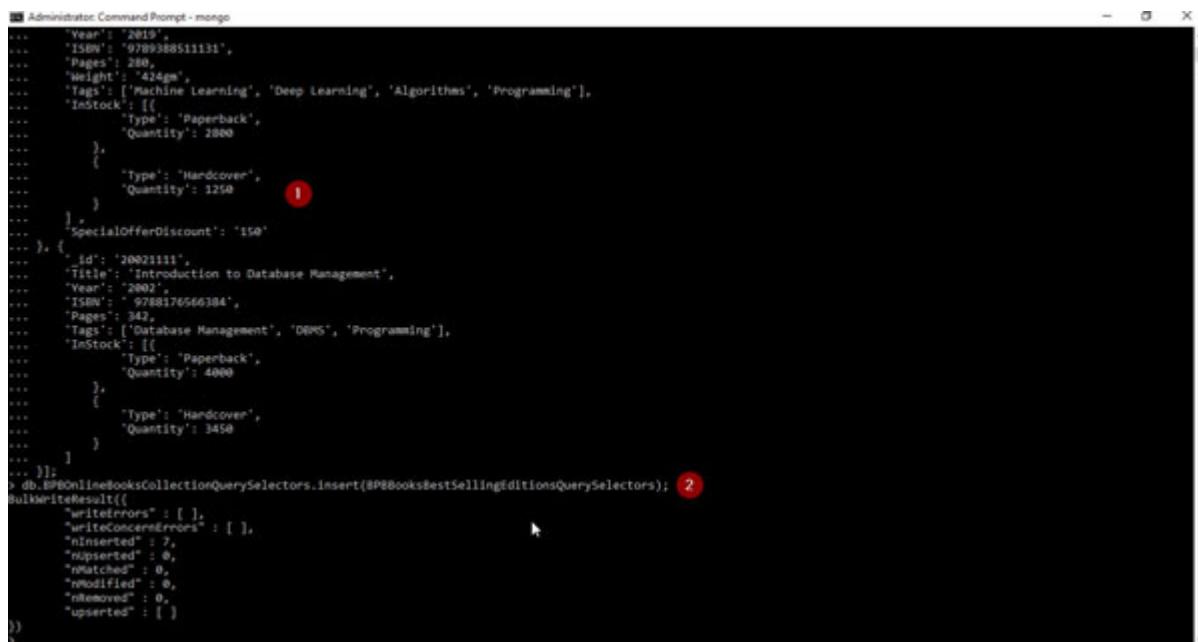
        'Type': 'Hardcover',
        'Quantity': 3450
    }
]
} ];

```

Then, we have used the MongoDB `insert()` method to create a new document in the MongoDB collection "BPBOnlineBooksCollectionQuerySelectors". The code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksCollectionQuerySelectors.insert(BPBBooksBestSellingEditionsQuerySelectors);
```



The screenshot shows the MongoDB shell interface with the command prompt "Administrator Command Prompt - mongo". The command being run is `db.BPBOnlineBooksCollectionQuerySelectors.insert(BPBBooksBestSellingEditionsQuerySelectors);`. The output shows two documents being inserted. The first document has an ID of 20021111, a title of "Introduction to Database Management", and a quantity of 3450. The second document has an ID of 20021112, a title of "Machine Learning", and a quantity of 1250. Both documents have a "SpecialOfferDiscount" field set to 15%. The shell also displays the results of the insert operation, including the number of inserted documents (2), and other metrics like writeConcernErrors and upserted.

```

{
  "_id": "20021111",
  "Title": "Introduction to Database Management",
  "Year": 2002,
  "ISBN": "9788176566384",
  "Pages": 842,
  "Tags": ["Database Management", "DBMS", "Programming"],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 4000
    },
    {
      "Type": "Hardcover",
      "Quantity": 3450
    }
  ]
},
{
  "_id": "20021112",
  "Title": "Machine Learning",
  "Year": 2019,
  "ISBN": "9789388511131",
  "Pages": 288,
  "Weight": "424gms",
  "Tags": ["Machine Learning", "Deep Learning", "Algorithems", "Programming"],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 2800
    },
    {
      "Type": "Hardcover",
      "Quantity": 1250
    }
  ],
  "SpecialOfferDiscount": "15"
},
{
  "_id": "20021113",
  "Title": "Introduction to Database Management",
  "Year": 2002,
  "ISBN": "9788176566384",
  "Pages": 842,
  "Tags": ["Database Management", "DBMS", "Programming"],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 4000
    },
    {
      "Type": "Hardcover",
      "Quantity": 3450
    }
  ]
}
];
db.BPBOnlineBooksCollectionQuerySelectors.insert(BPBBooksBestSellingEditionsQuerySelectors);
{
  "writeErrors": [],
  "writeConcernErrors": [],
  "nInserted": 2,
  "nUpserted": 0,
  "nMatched": 0,
  "nModified": 0,
  "nRemoved": 0,
  "nUpgraded": 0
}

```

*Figure 13.1: Creating a New Collection and Inserting a New Data*

We are done with creating a new collection and inserting new documents in it. Let us now create an index to our new collection.

## Examples of comparison selectors

The following are some examples of the comparison selectors. You may run different examples by changing the comparison selectors,

fields, and values.

## **Example 1 - \$gt Comparison Selector**

In our example, we have used the "\$gt" comparison selector in our collection "BPBOnlineBooksCollectionQuerySelectors" in the field "InStock.Quantity" to find all the documents that have their "InStock.Quantity" greater than 2500. The code and the selector details for the same are shown in the following screenshot:

### **Code 1**

```
db.BPBOnlineBooksCollectionQuerySelectors.find({  
    "InStock.Quantity": {  
        $gt: 2500  
    }  
}).pretty();
```

### **Selector Details**

Selector	Selector Description	Selector Use and Syntax
\$gt	This selector matches the documents that have values greater than the specified value	{<document field> : {\$gt : <value to match>}}

***Table 13.6: Using \$gt Comparison Selector - Details***

```

Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollectionQuerySelectors.find({
...   "InStock.Quantity": {
...     $gt: 2500
...   }
... }).pretty();
{
  "_id": ObjectId("Seef06e5956f78e07fd71ddc"),
  "title": "Cloud Computing",
  "Year": "2019",
  "ISBN": "9789388511407",
  "Pages": 330,
  "Weight": "570gm",
  "Dimension": "23x19x1.5cm",
  "Tags": [
    "Cloud Computing",
    "Cloud Computing Concepts",
    "Non Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 3000
    },
    {
      "Type": "Hardcover",
      "Quantity": 1500
    }
  ],
  "SpecialOfferDiscount": "100"
}

{
  "_id": ObjectId("Seef06e5956f78e07fd71ddc"),
  "title": "Introduction to Digital Marketing 301 : Easy to Learn and Implement Hands-on Guide for Digital Marketing",
  "Year": "2019",
  "ISBN": "9789389328189",
  "Pages": 464,
  "Tags": [
    "Digital Marketing",
    "Digital Marketing Tips",
    "Non Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback"
    }
  ]
}

```

*Figure 13.2: Using \$gt Comparison Selector*

## Example 2 - \$lte comparison selector

In our example, we have used the "\$lte" comparison selector in our collection "BPBOnlineBooksCollectionQuerySelectors" in the field "InStock.Quantity" to find all the documents that have their "InStock.Quantity" less than or equal to 1300. The code and the selector details for the same are shown in the following screenshot:

### Code 1

```

db.BPBOnlineBooksCollectionQuerySelectors.find({
  "InStock.Quantity": {
    $lte: 1300
  }
}).pretty();

```

### Selector Details

Selector	Selector Description	Selector Use and Syntax
\$lte	This selector matches the documents that have values	{<document field> : {\$lte : <value to match>}}

less than or equal to the specified value.

**Table 13.7: Using \$lte Comparison Selector - Details**



```
Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollectionQuerySelectors.find({
...   "inStock.Quantity": {
...     "$lte": 1500
...   }
... }).pretty();
{
  "_id": ObjectId("5ef05e5956f78e07fd71ddd"),
  "title": "IOT and Smart Cities: Your Smart City Planning Guide",
  "Year": "2010",
  "ISBN": "9789388511322",
  "Pages": 242,
  "Weight": "357gm",
  "Dimension": "22.5x15x1.5gm",
  "Tags": [
    "Internet of Things",
    "IOT",
    "Smart City",
    "Planning Guide",
    "Non Programming"
  ],
  "inStock": [
    {
      "Type": "Paperback",
      "Quantity": 2000
    },
    {
      "Type": "Hardcover",
      "Quantity": 1000
    }
  ],
  "SpecialOfferDiscount": "200"
}
{
  "_id": ObjectId("5ef05e5956f78e07fd71dde"),
  "title": "Machine Learning with Python",
  "Year": "2018",
  "ISBN": "9789386551931",
  "Pages": 207,
  "Tags": [
    "Python",
    "Machine Learning",
    "Python Programming",
    "Programming"
  ]
}
```

**Figure 13.3: Using \$lte Comparison Selector**

## Example 3 - \$in comparison selector

In our example, we have used the "\$in" comparison selector in our collection "BPBOnlineBooksCollectionQuerySelectors" in the field "Tags" to find all the documents that have the "Tags" containing the values "Machine Learning" or "DBMS" or "AI". The code and the selector details for the same are shown in the following screenshot:

### Code 1

```
db.BPBOnlineBooksCollectionQuerySelectors.find({
  "Tags": {
    "$in": ["Machine Learning", "DBMS", "AI"]
  }
}).pretty();
```

### Selector Details

Selector	Selector Description	Selector Use and Syntax
\$in	This selector matches the documents that have any of the values specified in an array.	{<document field> : {\$in : [<array value1>, <array value2>, ... <array valueN>]}}

**Table 13.8: Using \$in Comparison Selector - Details**



```
Administrator: Command Prompt - mongo
> db.BPBOnlineBooksCollectionQuerySelectors.find({
...   "Tags": {
...     "$in": ["Machine Learning", "DEMS", "AI"]
...   }
... }).pretty();
{
  "_id": ObjectId("5e05956f78e07fd71ddc"),
  "Title": "Machine Learning with Python",
  "Year": "2018",
  "ISBN": "9789386551931",
  "Pages": 267,
  "Tags": [
    "Python",
    "Machine Learning",
    "Python Programming",
    "Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 4500
    },
    {
      "Type": "Hardcover",
      "Quantity": 1300
    }
  ]
}
{
  "_id": ObjectId("5e05956f78e07fd71ddc"),
  "Title": "Artificial Intelligence Ethics and International Law: An Introduction",
  "Year": "2019",
  "ISBN": "9789388511629",
  "Pages": 188,
  "Weight": "260gm",
  "Tags": [
    "Artificial Intelligence",
    "International Law",
    "AI",
    "Artificial Intelligence Ethics",
    "Non Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 4500
    }
  ]
}
```

**Figure 13.4: Using \$in Comparison Selector**

## Examples of logical selectors

The following are some examples of the logical selectors. You may run different examples by changing the logical selectors, fields, and values.

### Example 1 - \$and logical selector

In our example, we have used the "\$and" logical selector in our collection "BPBOnlineBooksCollectionQuerySelectors" in the fields "SpecialOfferDiscount" and "Year" to find all the documents that have the "SpecialOfferDiscount" greater than or equal to 150 and "Year" equal to 2019. The code and the selector details for the same are shown in the following screenshot:

## Code 1

```
db.BPBOnlineBooksCollectionQuerySelectors.find({  
    $and: [{  
        SpecialOfferDiscount: {  
            $gte: 150  
        }  
    }, {  
        Year: {  
            $eq: "2019"  
        }  
    }]  
}).pretty();
```

## Selector Details

Selector	Selector Description	Selector Use and Syntax
\$and	This selector performs the Logical AND operation on different expressions and joins both the queries to deliver the combined result by returning all the documents based on the join.	<pre>{\$and : [ {&lt;expression1&gt;} , &lt;expression2&gt; , ... , &lt;expressionN&gt; ] }</pre>

**Table 13.9: Using \$and Logical Selector - Details**

```

Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollectionQuerySelectors.find({
...   $and: [
...     {SpecialOfferDiscount: {
...       $gte: "150"
...     }},
...     {Year: {
...       $eq: "2019"
...     }}
...   ]
... }).pretty();
{
  "_id": ObjectId("Seef06e5956f78e07fd71ddd"),
  "Title": "IoT and Smart Cities: Your Smart City Planning Guide",
  "Year": "2019",
  "ISBN": "9789388511322",
  "Pages": 242,
  "Weight": "357gm",
  "Dimension": "22.5x15x1.5ga",
  "Tags": [
    "Internet of Things",
    "IoT",
    "Smart City",
    "Planning Guide",
    "Non Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 2000
    },
    {
      "Type": "Hardcover",
      "Quantity": 1000
    }
  ],
  "SpecialOfferDiscount": "200"
}
{
  "_id": ObjectId("Seef06e5956f78e07fd71de0"),
  "Title": "A Practical Approach for Machine Learning and Deep Learning Algorithms",
  "Year": "2019",
  "ISBN": "9789388511131",
  "Pages": 288
}

```

**Figure 13.5: Using \$and Logical Selector**

## Example 2 - \$not logical selector

In our example, we have used the "\$not" logical selector in our collection "BPBOnlineBooksCollectionQuerySelectors" in the field "Year" to find all the documents that have the "Year" not equal to 2019. The code and the selector details for the same are shown in the following screenshot:

### Code 1

```

db.BPBOnlineBooksCollectionQuerySelectors.find({
  Year: {
    $not: {
      $eq: "2019"
    }
  }
}).pretty();

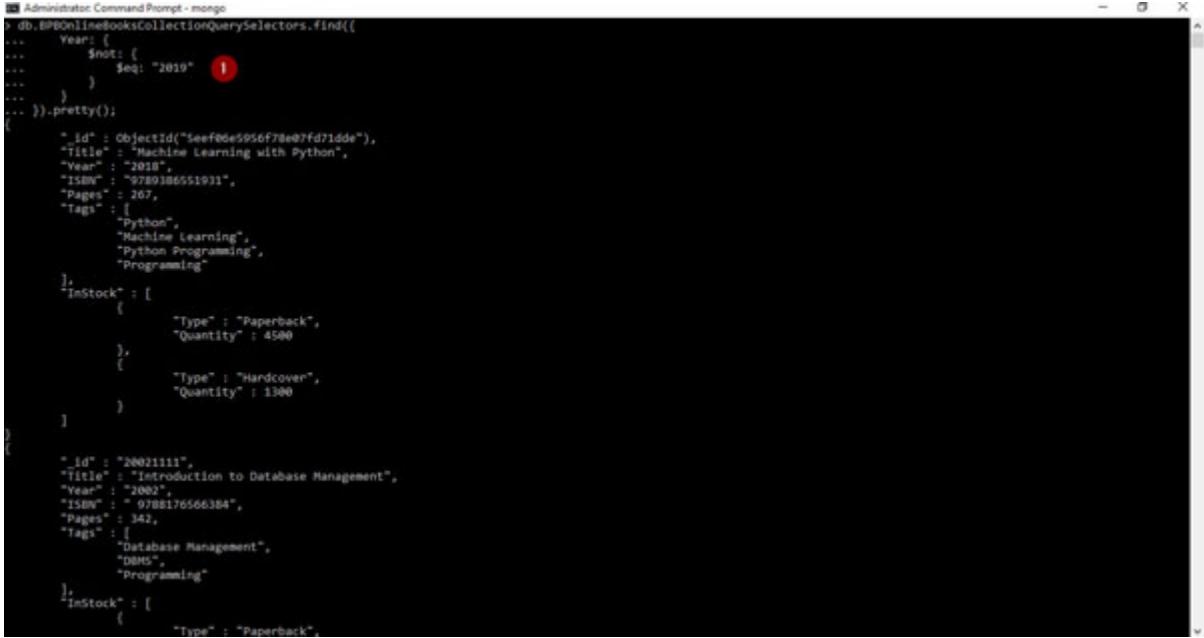
```

### Selector details

Selector	Selector Description	Selector Use and Syntax

\$not	This selector performs the logical NOT operation and returns the documents that do not match with the expression.	{<document field> : {\$not : {<expression>}}}
-------	---	---

**Table 13.10: Using \$not Logical Selector - Details**



```
Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollectionQuerySelectors.find({
...   Year: {
...     $not: {
...       $eq: "2019"
...     }
...   }
... }),pretty();
{
  "_id": ObjectId("5eef0de5956f78e07fd71ddc"),
  "Title": "Machine Learning with Python",
  "Year": "2018",
  "ISBN": "97809386551931",
  "Pages": 267,
  "Tags": [
    "Python",
    "Machine Learning",
    "Python Programming",
    "Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 4500
    },
    {
      "Type": "Hardcover",
      "Quantity": 1300
    }
  ]
},
{
  "_id": "2009211111",
  "Title": "Introduction to Database Management",
  "Year": "2002",
  "ISBN": "97881765660384",
  "Pages": 342,
  "Tags": [
    "Database Management",
    "DBMS",
    "Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback"
    }
  ]
}
```

**Figure 13.6: Using \$not Logical Selector**

## Examples of element selectors

The following are some examples of the element selectors. You may run different examples by changing the element selectors, fields, and values.

### Example 1 - \$exists element selector

In our example, we have used the "\$and" element selector in our collection "BPBOnlineBooksCollectionQuerySelectors" in the field "SpecialOffer Discount" to find all the documents that have the "SpecialOfferDiscount" field present. The query will not select the documents that do not have "Special OfferDiscount" field. The code and the selector details for the same are shown in the following screenshot:

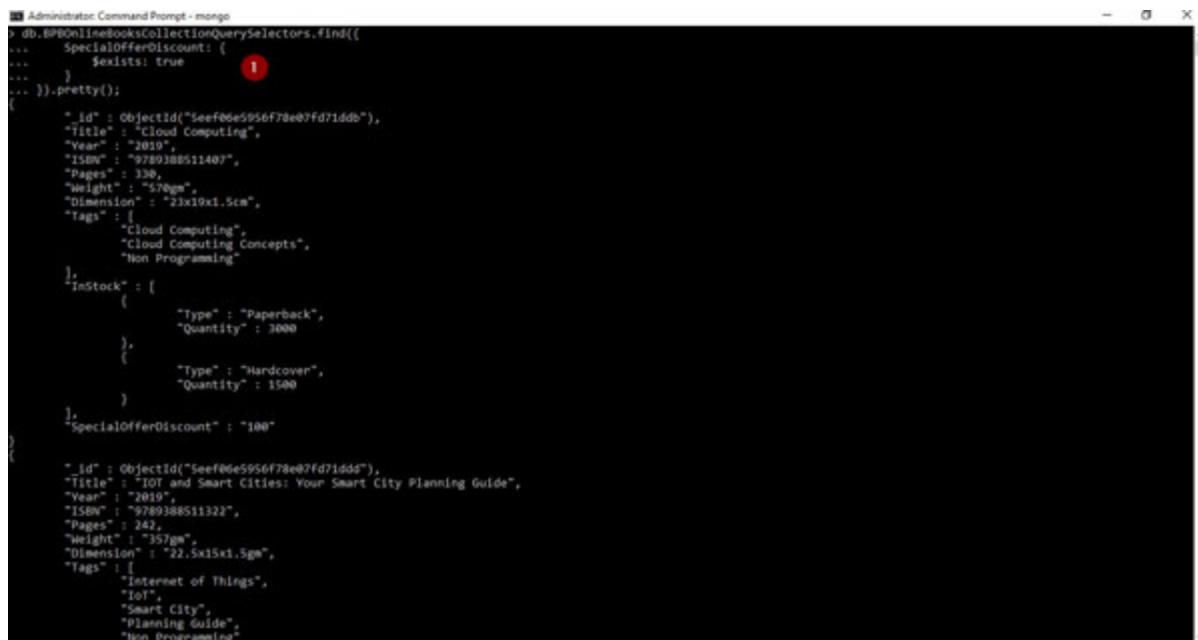
## Code 1

```
db.BPBOnlineBooksCollectionQuerySelectors.find({
  SpecialOfferDiscount: {
    $exists: true
  }
}).pretty();
```

## Selector Details

Selector	Selector Description	Selector Use and Syntax
\$exists	This selector matches the documents that have the specified field in the documents. It accepts the Boolean value. If the value is set to "true", it returns all the documents that have the specified field. If the value is set to "false", it returns all the documents that do not contain the specified field.	{<document field> : {\$exists: <boolean>}}

**Table 13.11: Using \$exists Element Selector - Details**



The screenshot shows the MongoDB shell interface. A command is being run in the 'Administrator Command Prompt - mongo' window:

```
db.BPBOnlineBooksCollectionQuerySelectors.find({
  ...
  SpecialOfferDiscount: {
    ...
    $exists: true
  }
}).pretty();
```

The '\$exists' selector is highlighted with a red circle and a question mark. The output of the query is displayed below, showing two documents from the collection. The first document has a 'SpecialOfferDiscount' field set to '100'. The second document has a 'SpecialOfferDiscount' field set to null.

```
...
{
  "_id": ObjectId("5eef06e5956f78e07fd71dd5"),
  "title": "Cloud Computing",
  "Year": "2019",
  "ISBN": "9789388511487",
  "Pages": 330,
  "Weight": "570gm",
  "Dimension": "23x19x1.5cm",
  "Tags": [
    "Cloud Computing",
    "Cloud Computing Concepts",
    "Non Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 3000
    },
    {
      "Type": "Hardcover",
      "Quantity": 1500
    }
  ],
  "SpecialOfferDiscount": "100"
}

{
  "_id": ObjectId("5eef06e5956f78e07fd71ddd"),
  "title": "IOT and Smart Cities: Your Smart City Planning Guide",
  "Year": "2019",
  "ISBN": "9789388511322",
  "Pages": 242,
  "Weight": "357gm",
  "Dimension": "22.5x15x1.5cm",
  "Tags": [
    "Internet of Things",
    "Iot",
    "Smart City",
    "Planning Guide",
    "Non Programming"
  ]
}
```

*Figure 13.7: Using \$exists Element Selector*

## Examples of array selectors

The following are some examples of the array selectors. You may run different examples by changing the array selectors, fields, and values.

### Example 1 - \$all array selector

In our example, we have used the "\$all" array selector in our collection "BPBOnlineBooksCollectionQuerySelectors" in the field "Tags" to find all the documents that have the "Tags" having the values "Machine Learning" as well as "Programming" present in them. The code and the selector details for the same are shown in the following screenshot:

### Code 1

```
db.BPBOnlineBooksCollectionQuerySelectors.find({  
    Tags: {  
        $all: ["Machine Learning", "Programming"]  
    }  
}).pretty();
```

### Selector Details

Selector	Selector Description	Selector Use and Syntax
\$all	This selector matches the documents that contain all the elements in an array type field of a document that contains all the elements specified in the query.	{<document field> : {\$all : [<array value1>, <array value2> ... <array valueN>]}}}

*Table 13.12: Using \$all Array Selector - Details*

The screenshot shows a terminal window titled "Administrator Command Prompt - mongo". The command entered is:

```
db.BPBOnlineBooksCollectionQuerySelectors.find({  
...   $all: [  
...     "Machine Learning", "Programming"  
...   ]  
... }).pretty();
```

The output displays two documents from the collection. Both documents have an array named "Tags" containing the values "Machine Learning" and "Programming". The first document also has an array named "InStock" containing two objects: one for "Paperback" type with quantity 4500, and one for "Hardcover" type with quantity 1300.

Figure 13.8: Using \$all Array Selector

## Examples of evaluation selectors

The following are some examples of the evaluation selectors. You may run different examples by changing the evaluation selectors, fields, and values.

### Example 1 - \$regex evaluation selector

In our example, we have used the "\$regex" evaluation selector in our collection "BPBOnlineBooksCollectionQuerySelectors" in the field "Title" to find all the documents that have the word "Machine" present in the "Title". The code and the selector details for the same are shown in the following screenshot:

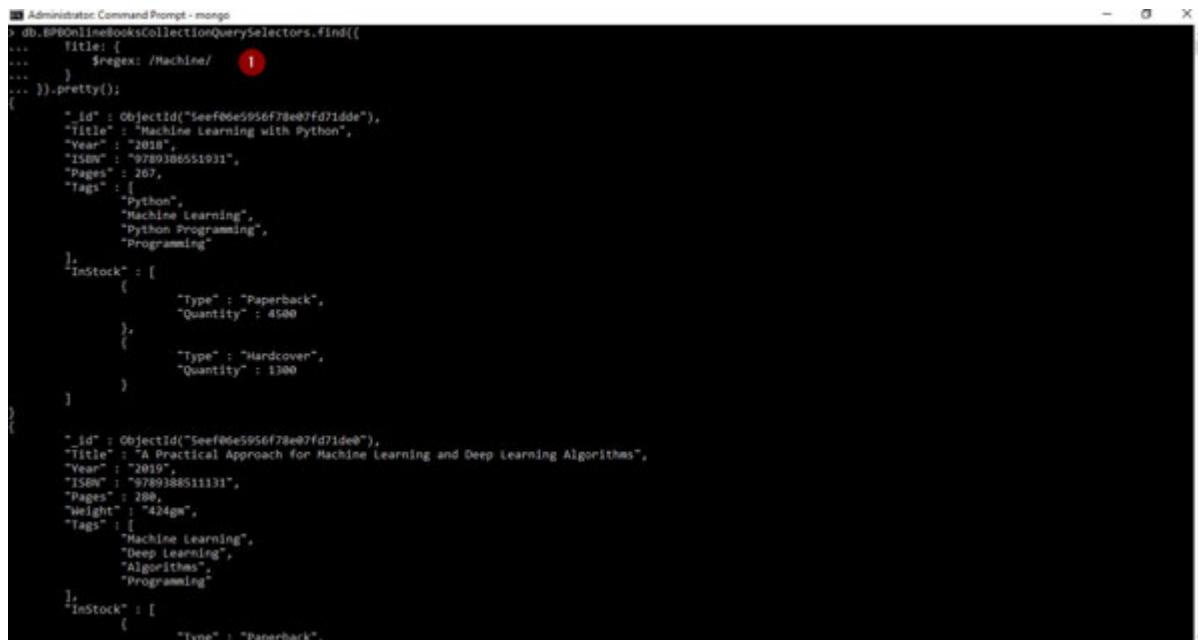
### Code 1

```
db.BPBOnlineBooksCollectionQuerySelectors.find({  
  Title: {  
    $regex: /Machine/  
  }  
}).pretty();
```

## Selector details

Selector	Selector Description	Selector Use and Syntax
\$regex	This selector selects all the documents where values are matched with the specified regular expression. We can use various ways (as shown in selector use and syntax) to use this selector.	{<document field> : {\$regex: /pattern/, \$options: '<options>'}} {<document field> : {\$regex: 'pattern', \$options: '<options>'}} {<document field> : {\$regex: /pattern/<options>}}

**Table 13.13: Using \$regex Evaluation Selector - Details**



```
Administrator Command Prompt - mongo
> db.BPOnlineBooksCollectionQuerySelectors.find({
...   title: {
...     $regex: /Machine/ !!
...   }
... }).pretty();
{
  "_id": ObjectId("5eeff05e5956f78e07fd71dde"),
  "Title": "Machine Learning with Python",
  "Year": "2018",
  "ISBN": "9789386551031",
  "Pages": 267,
  "Tags": [
    "Python",
    "Machine Learning",
    "Python Programming",
    "Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 4500
    },
    {
      "Type": "Hardcover",
      "Quantity": 1300
    }
  ]
}

{
  "_id": ObjectId("5eeff05e5956f78e07fd71deab"),
  "Title": "A Practical Approach for Machine Learning and Deep Learning Algorithms",
  "Year": "2019",
  "ISBN": "9789388511131",
  "Pages": 200,
  "Weight": "424gm",
  "Tags": [
    "Machine Learning",
    "Deep Learning",
    "Algorithms",
    "Programming"
  ],
  "InStock": [
    {
      "Type": "Paperback",
      ...
    }
  ]
}
```

**Figure 13.9: Using \$regex Evaluation Selector**

## Conclusion

In this chapter, we studied about the query selectors and their benefits. We studied the various types of query selectors available in MongoDB. We also covered the query selectors with the practical examples.

In the next chapter of this book, we will study about the projection in MongoDB. We will learn about the benefits of using the projection

and then, we will learn about the various types of projection operators available in MongoDB. We will also cover the projection operators with step-by-step practical examples.

## **Questions**

1. What are the uses of query selectors?
2. List the four different types of query selectors.
3. What are the logical query selectors? Explain them with some examples.
4. Give one example of the element selector.
5. Give one example of the array selector.

# CHAPTER 14

## Projection in MongoDB and Projection Operators

This chapter covers the projection in MongoDB. We will begin with an introduction of the projection and the benefits of using the projection. In this chapter, we will also cover the various types of projection operators available in MongoDB. The last section of this chapter has some step-by-step practical examples of the projection operators.

### Structure

In this chapter, we will discuss the following topics:

- Introduction to projection
- Introduction to the projection operators
- Examples and use of the projection operators

### Objectives

After studying this unit, you should be able to understand projection in MongoDB and also the benefits of using projection. Later in this chapter you will learn different types of projection operators and how to use projection operators with practical examples.

### Introduction to projection

Whenever we select some documents from the collections in MongoDB using queries, it gives us the list of all the fields present in those documents. Most of the times, we do not require all the fields, but only some specific fields. In these cases, projection comes into

picture. In MongoDB, projection means to select only the required data rather than selecting the entire data.

Suppose, there is a collection that has many documents of varied sizes in terms of field structure. Some documents have 10 fields, some have 15, etc. However, during our query; we want to fetch only 2 fields. In this case, if we use projection, we will get lesser data from MongoDB which makes good sense in many scenarios during the application development, where projection helps in the query optimization with respect to time, speed, and even memory.

Before we start using projection and the projection operators, let's populate our new collection with new data. In our example, we have created a variable named `BPBBooksBestSellingEditionsProjection`. The code for the same is as follows:

## Code 1

```
var BPBBooksBestSellingEditionsProjection = [{}  
    'Title': 'Cloud Computing',  
    'Year': '2019',  
    'ISBN': '9789388511407',  
    'Pages': 330,  
    'Weight': '570gm',  
    'Dimension': '23x19x1.5cm',  
    'Tags': ['Cloud Computing', 'Cloud Computing Concepts', 'Non  
    Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 3000  
    },  
    {  
        'Type': 'Hardcover',  
        'Quantity': 1500  
    }  
],  
    'SpecialOfferDiscount': '100'  
}, {
```

```
'Title': 'Introduction to Digital Marketing 101 : Easy to Learn and Implement Hands-on Guide for Digital Marketing',
'Year': '2019',
'ISBN': '9789389328189',
'Pages': 464,
'Tags': ['Digital Marketing', 'Digital Marketing Tips', 'Non Programming'],
'InStock': [{  
    'Type': 'Paperback',  
    'Quantity': 4000  
},  
{  
    'Type': 'Hardcover',  
    'Quantity': 2300  
}  
]  
, {  
    'Title': 'IOT and Smart Cities: Your Smart City Planning Guide',  
    'Year': '2019',  
    'ISBN': '9789388511322',  
    'Pages': 242,  
    'Weight': '357gm',  
    'Dimension': '22.5x15x1.5gm',  
    'Tags': ['Internet of Things', 'IoT', 'Smart City', 'Planning Guide', 'Non Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 2000  
},  
{  
        'Type': 'Hardcover',  
        'Quantity': 1000  
}  
],  
    'SpecialOfferDiscount': '200'  
, {
```

```
'Title': 'Machine Learning with Python',
'Year': '2018',
'ISBN': '9789386551931',
'Pages': 267,
'Tags': ['Python', 'Machine Learning', 'Python Programming',
'Programming'],
'InStock': [
    {
        'Type': 'Paperback',
        'Quantity': 4500
    },
    {
        'Type': 'Hardcover',
        'Quantity': 1300
    }
]
}, {
    'Title': 'Artificial Intelligence Ethics and International
Law: An Introduction',
    'Year': '2019',
    'ISBN': ' 9789388511629',
    'Pages': 188,
    'Weight': '268gm',
    'Tags': ['Artificial Intelligence', 'International Law',
'AI', 'Artificial Intelligence Ethics', 'Non Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 5200
        },
        {
            'Type': 'Hardcover',
            'Quantity': 3300
        }
    ]
}, {
    'Title': 'A Practical Approach for Machine Learning and Deep
Learning Algorithms',
    'Year': '2019',
```

```

'ISBN': '9789388511131',
'Pages': 280,
'Weight': '424gm',
'Tags': ['Machine Learning', 'Deep Learning', 'Algorithms',
'Programming'],
'InStock': [
    {
        'Type': 'Paperback',
        'Quantity': 2800
    },
    {
        'Type': 'Hardcover',
        'Quantity': 1250
    }
],
'SpecialOfferDiscount': '150'
}, {
    '_id': '20021111',
    'Title': 'Introduction to Database Management',
    'Year': '2002',
    'ISBN': ' 9788176566384',
    'Pages': 342,
    'Tags': ['Database Management', 'DBMS', 'Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 4000
        },
        {
            'Type': 'Hardcover',
            'Quantity': 3450
        }
    ]
}];

```

Then, we have used the MongoDB `insert()` method to create new documents in the MongoDB Collection `BPBOnlineBooksCollectionProjection`. The code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksCollectionProjection.insert(BPBBooksBestSellin  
gEditionsProjection);
```

The screenshot shows the MongoDB Command Prompt window titled "Administrator: Command Prompt - mongo". The command entered is:

```
db.BPBOnlineBooksCollectionProjection.insert(BPBBooksBestSellin  
gEditionsProjection);
```

Two red circles are overlaid on the screen:

- Circle 1 is positioned over the first document being inserted, which contains details about a book with ISBN 9789388511131.
- Circle 2 is positioned over the command itself, specifically the part where the projection is specified.

**Figure 14.1: Creating a New Collection and Inserting a New Data**

Since we are done with creating a new collection and inserting new documents in it, let us now create an index to our new collection.

## How to use projection in MongoDB?

It is very simple to use projection with the MongoDB queries. There is the following two ways to use projection:

- **Show Only Specific Fields** – In this method, we can specify the fields we would like to display by the query. The fields can be set with a value of 1 or 0, and in this way, after the query, we can specify the field with 1 as an option. Here, 1 means to display those fields only among all the other fields in the document. In this case, only the fields that have 1 as an option will be displayed and the rest of the fields will remain hidden.
- **Hide Specific Fields** – In this method, we can specify which fields we would like to hide by the query, and in this way, after the query, we can specify the field with 0 as an option. Here, 0

means to hide those fields among all the other fields in the document. In this case, only the fields that have 0 as an option will be hidden and the rest of the fields will be displayed.

Note that we cannot combine and use both of these methods in a single query as this is not permitted in MongoDB and will result in an error. Mixing of 1 and 0 is only allowed to hide the `_id` field which is always displayed. We will cover all this with the practical examples.

## Examples of projection

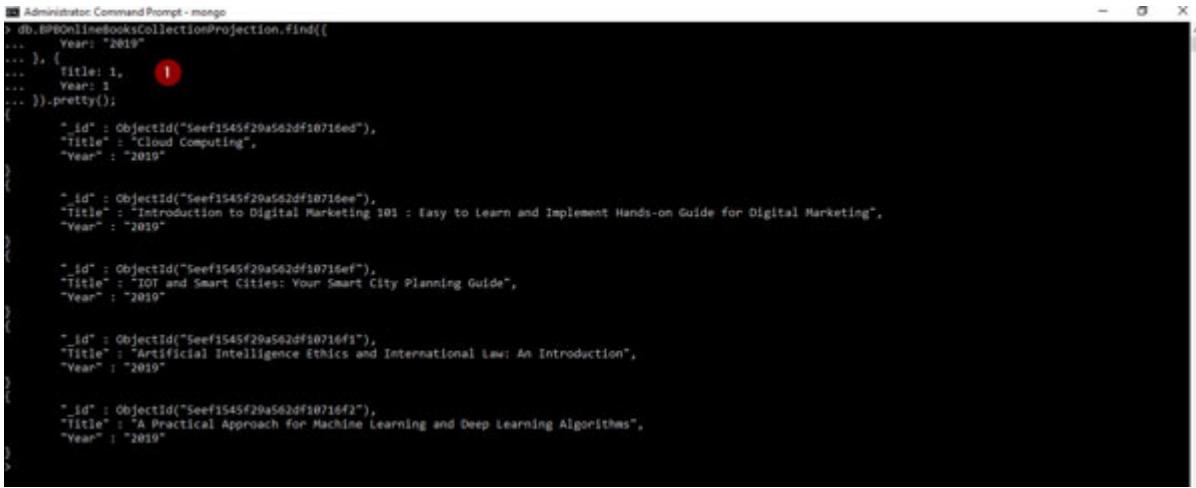
The following are some examples of how to use projection in MongoDB. You may run different examples by changing the query and the fields.

### Example 1 - Show only specific fields

In our example, we have used projection in our collection `BPBOnlineBooksCollectionProjection`. This query will show all the documents that have their `Year` as 2019 but will only show 2 Fields, `Title` and `Year`, in addition to the `_id` field which is displayed by default. The code for the same is shown in the following screenshot:

#### **Code 1**

```
db.BPBOnlineBooksCollectionProjection.find({  
    Year: "2019"  
, {  
    Title: 1,  
    Year: 1  
}).pretty();
```

A screenshot of a Windows Command Prompt window titled "Administrator Command Prompt - mongo". Inside, a MongoDB query is run against a collection named "BPBOnlineBooksCollectionProjection". The query filters documents where the "Year" field is "2019" and then uses a projection object to specify which fields to return: "\_id", "title", and "Year". A red circle with a question mark is drawn around the projection object to draw attention to it.

```
db.BPBOnlineBooksCollectionProjection.find({  
...  
  Year: "2019"  
... }, {  
...   title: 1,  
...   Year: 1  
... }).pretty();  
  
{  
  "_id": ObjectId("5eef1545f29a562df10716ed"),  
  "title": "Cloud Computing",  
  "Year": "2019"  
}  
  
{  
  "_id": ObjectId("5eef1545f29a562df10716ee"),  
  "title": "Introduction to Digital Marketing 101 : Easy to learn and Implement Hands-on Guide for Digital Marketing",  
  "Year": "2019"  
}  
  
{  
  "_id": ObjectId("5eef1545f29a562df10716ef"),  
  "title": "IOT and Smart Cities: Your Smart City Planning Guide",  
  "Year": "2019"  
}  
  
{  
  "_id": ObjectId("5eef1545f29a562df10716f1"),  
  "title": "Artificial Intelligence Ethics and International Law: An Introduction",  
  "Year": "2019"  
}  
  
{  
  "_id": ObjectId("5eef1545f29a562df10716f2"),  
  "title": "A Practical Approach for Machine Learning and Deep Learning Algorithms",  
  "Year": "2019"  
}
```

**Figure 14.2: Projection - Show Only Specific Fields**

## Example 2 - Hide specific fields

In our example, we have used projection in our collection "BPBOnlineBooksCollectionProjection". This query will display all the documents that have their Year as 2019, but will not show 2 fields, Tags and InStock. The code for the same is shown in the following screenshot:

### Code 1

```
db.BPBOnlineBooksCollectionProjection.find({  
  Year: "2019"  
, {  
  Tags: 0,  
  InStock: 0  
}).pretty();
```

```

Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollectionProjection.find({
...   Year: "2019"
... }, {
...   Tags: 0,
...   InStock: 0
... }).pretty();
{
  "_id": ObjectId("5eef1545f29a562df10716ef"),
  "Title": "Cloud Computing",
  "Year": "2019",
  "ISBN": "9789388511407",
  "Pages": 330,
  "Weight": "570gm",
  "Dimension": "23x19x1.5cm",
  "SpecialOfferDiscount": "100"
}

{
  "_id": ObjectId("5eef1545f29a562df10716ef"),
  "Title": "Introduction to Digital Marketing 101 : Easy to Learn and Implement Hands-on Guide for Digital Marketing",
  "Year": "2019",
  "ISBN": "9789389328189",
  "Pages": 464
}

{
  "_id": ObjectId("5eef1545f29a562df10716ef"),
  "Title": "IoT and Smart Cities: Your Smart City Planning Guide",
  "Year": "2019",
  "ISBN": "9789388511322",
  "Pages": 242,
  "Weight": "357gm",
  "Dimension": "22.5x15x1.5gm",
  "SpecialOfferDiscount": "200"
}

{
  "_id": ObjectId("5eef1545f29a562df10716ef"),
  "Title": "Artificial Intelligence Ethics and International Law: An Introduction",
  "Year": "2019",
  "ISBN": "9789388511629",
  "Pages": 188,
  "Weight": "260gm"
}

{
  "_id": ObjectId("5eef1545f29a562df10716ef"),
  "Title": "A Practical Approach for Machine Learning and Deep Learning algorithms"
}

```

*Figure 14.3: Projection - Hide Specific Fields*

## Example 1 - Show only specific fields and hide the \_id field

In our example, we have used projection in our collection `BPBOnlineBooksCollectionProjection`. This query will display all the documents that have their `Year` as 2019, but will only show 2 fields `Title` and `Year`. Here, the `_id` field will not be displayed, since we will set it to 0. The code for the same is shown in the following screenshot:

### Code 1

```

db.BPBOnlineBooksCollectionProjection.find({
  Year: "2019"
}, {
  _id: 0,
  Title: 1,
  Year: 1
}) ;

```

```

Administrator Command Prompt - mongo
> DB.BooksCollectionProjection.find({
...   Year: "2019"
... }, {
...   _id: 0,
...   title: 1,
...   Year: 1
... }).pretty();
{
  "title": "Cloud Computing", "Year": "2019"
}
{
  "title": "Introduction to Digital Marketing 201 : Easy to Learn and Implement Hands-on Guide for Digital Marketing",
  "Year": "2019"
}
{
  "title": "IOT and Smart Cities: Your Smart City Planning Guide",
  "Year": "2019"
}
{
  "title": "Artificial Intelligence Ethics and International Law: An Introduction",
  "Year": "2019"
}
{
  "title": "A Practical Approach for Machine Learning and Deep Learning Algorithms",
  "Year": "2019"
}

```

**Figure 14.4:** Show Only Specific Fields and Hide `_id` Field

## Introduction to projection operators

The projection operators help perform the projection operations and implement various logics required in some scenarios. The following table has the list of the projection operators:

Operator	Operator Description	Operator Use and Syntax
\$	This operator projects the first element in the array type field that matches the query condition.	{<query>}, {<array type field>.\$: 1}
\$elemMatch	This operator projects the first element in the array type field that matches the condition given in the \$elemMatch	{<array type field>: {\$elemMatch: {<query>}}}
\$slice	This operator limits the number of elements projected from an array. We can also use skip and limit in this operator.	{<array type field>: {\$slice: <number of elements>}}

**Table 14.1:** Projection Operators

## Examples of projection operators

The following are some examples of the projection operators. You may run different examples by changing the query, fields, and values.

## Example 1 - \$ projection operator

In our example, we have used the \$ projection operator in our collection `BPBOnlineBooksCollectionProjection` in the field `Tags` to find all the documents that have `Tags` with both the values, Programming and Machine Learning, present in them. Then, we have used the \$ projection operator on `Tags` array type field to project the first element. The code and the operator details for the same as shown in the following screenshot:

### Code 1

```
db.BPBOnlineBooksCollectionProjection.find({  
    Tags: {  
        $all: ["Programming", "Machine Learning"]  
    }  
}, {  
    "Tags.$": 1  
}).pretty();
```

## Operator details

Operator	Operator Description	Operator Use and Syntax
\$	This operator projects the first element in the array type field that matches the query condition.	{<query>} , {<array field>.\$: 1}

*Table 14.2: \$ Projection Operator - Details*



The screenshot shows a terminal window titled 'Administrator: Command Prompt - mongo'. The command entered is:

```
db.BPBOnlineBooksCollectionProjection.find({  
    Tags: {  
        $all: ["Programming", "Machine Learning"]  
    }  
}, {  
    "Tags.$": 1  
}).pretty();
```

The output shows two documents. The first document has an '\_id' field and a 'Tags' field containing a single element 'Machine Learning'. The second document also has an '\_id' field and a 'Tags' field containing a single element 'Machine Learning'. A red circle with a question mark is placed over the '\$' symbol in the query's projection stage.

*Figure 14.5: Using \$ Projection Operator*

## Example 2 - \$elemMatch projection operator

In our example, we have used the `$elemMatch` projection operator in our collection `BPBOnlineBooksCollectionProjection`. We have first used the query to find all the documents that have `Year` equal to 2019 and then we have used the `$elemMatch` projection operator on `InStock` array type field to project the element `Quantity` having the value equal to 4000. The code and the operator details for the same are shown in the following screenshot:

### **Code 1**

```
db.BPBOnlineBooksCollectionProjection.find({  
    Year: "2019"  
, {  
    InStock: {  
        $elemMatch: {  
            Quantity: 4000  
        }  
    }  
}) ;
```

## Operator details

Operator	Operator Description	Operator Use and Syntax
<code>\$elemMatch</code>	This operator projects the first element in the array type field that matches the condition given in the <code>\$elemMatch</code> .	<code>{&lt;array type field&gt;: {\$elemMatch: {&lt;query&gt;}}}</code>

**Table 14.3: \$elemMatch Projection Operator - Details**



```
Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollectionProjection.find({
...   Year: "2019"
... }, {
...   InStock: [
...     { $elemMatch: {
...       Quantity: 4000
...     }
...   }
... ]).pretty();
{
  "_id": ObjectId("Seef1545f29a562df10716ed"),
  "_id": ObjectId("Seef1545f29a562df10716ee"),
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 4000
    }
  ]
}
{
  "_id": ObjectId("Seef1545f29a562df10716ef")
}
{
  "_id": ObjectId("Seef1545f29a562df10716f1")
}
{
  "_id": ObjectId("Seef1545f29a562df10716f2")
}
```

**Figure 14.6: Using \$elemMatch Projection Operator**

## Conclusion

In this chapter, we studied about projection in MongoDB. We also learned the benefits of using projection and the various types of projection operators available in MongoDB. In the last part of this chapter, we covered projection and the projection operators with practical examples.

In the next chapter of this book, we will cover aggregation in MongoDB. We will learn about aggregation and the benefits of using it. We will also cover the map-reduce method of aggregation. Finally, we will also cover the aggregation methods and how to use them with some practical examples.

## Questions

1. What is projection in MongoDB?
2. Explain some benefits of using projection.
3. Give two examples of the projection operators.
4. How can we use the `$` projection operator?
5. Give an example of the slice projection operator.

# CHAPTER 15

## Aggregation in MongoDB

This chapter covers the MongoDB aggregation. We will begin with an introduction to the aggregation in MongoDB and will learn the benefits of using MongoDB aggregation. We will learning about aggregation expression types and will cover it with step-by-step practical examples. Later in this chapter, we will learn Map-reduce in MongoDB and the benefits of using it, with some practical examples. In the last section of this chapter, we will cover the aggregation pipeline and its benefits and the working of aggregation pipeline in MongoDB in a detailed manner. At last, we will use some step-by-step practical examples to understand the aggregation pipeline in an easy-to-understand way.

### Structure

In this chapter, we will discuss the following topics:

- Introduction to MongoDB aggregation
- Introduction to aggregation expression types
- Step-by-step practical examples of aggregation expression types
- Introduction to map-reduce
- Step-by-step examples of map-reduce
- Introduction to aggregation pipeline
- Aggregation pipeline in MongoDB
- Step-by-step practical examples of MongoDB aggregation pipeline

### Objectives

After studying this unit, you should be able to understand the MongoDB aggregation and the MongoDB aggregation expression types. You will also learn the aggregation expression types. Later in this chapter, you

will learn about MongoDB map-reduce and aggregation pipeline and how it works in MongoDB using the step-by-step practical examples.

## Introduction to MongoDB aggregation

Aggregation means the gathering of things together. In Computer Science, data aggregation means the grouping of data to prepare combined data sets helpful in generating better information. Aggregation in MongoDB groups the data from various collections and then performs various operations to generate one combined result.

The aggregation method in MongoDB groups the data from various collections, then, in turn, performs some operations, like the total number (sum), average, minimum, maximum, etc. out of the selected groups.

## Aggregation method syntax and use

```
db.collection.aggregate(<AGGREGATE OPERATION>)
```

We can use the MongoDB aggregate method to perform the aggregate operation on the documents in MongoDB. We will understand this method better with the help of some practical examples. First, let us take a look at various expressions used in the aggregate operation. Following are the expression types used in the MongoDB aggregate operation.

Expression type	Expression type description	Expression type use and syntax
\$sum	Sums up the defined value from all the documents in a collection.	<pre>db.collection.aggregate([{\$group : {_id : \$&lt;Field Name&gt;, &lt;Field Label&gt; : {\$sum : &lt;Field Name, Number or Operation&gt;}}}])</pre>
\$avg	Calculates the average values from all the documents in a collection.	<pre>db.collection.aggregate([{\$group : {_id : \$&lt;Field Name&gt;, &lt;Field Label&gt; : {\$avg : &lt;Field Name, Number or Operation&gt;}}}])</pre>
\$min	Gives the minimum of all the values of	<pre>db.collection.aggregate([{\$group : {_id : \$&lt;Field Name&gt;, &lt;Field Label&gt; : {}}}]</pre>

	documents in a collection.	<code>[\$min : &lt;Field Name, Number or Operation&gt;} } ])</code>
\$max	Gives the maximum of all the values of documents in a collection.	<code>db.collection.aggregate([{\$group : {_id : \$&lt;Field Name&gt;, &lt;Field Label&gt; : {\$max : &lt;Field Name, Number or Operation&gt;} }}])</code>
\$push	Inserts values to an array of the resulting document.	<code>db.collection.aggregate([{\$group : {_id : \$&lt;Field Name&gt;, &lt;Field Label&gt; : {\$push : &lt;Field Name&gt;} }}])</code>
\$addToSet	Inserts values to an array of the resulting document, but does not create duplicates in the resulting document.	<code>db.collection.aggregate([{\$group : {_id : \$&lt;Field Name&gt;, &lt;Field Label&gt; : {\$addToSet : &lt;Field Name&gt;} }}])</code>
\$first	Gives the first document from the source document.	<code>db.collection.aggregate([{\$group : {_id : \$&lt;Field Name&gt;, &lt;Field Label&gt; : {\$first : &lt;Field Name&gt;} }}])</code>
\$last	Gives the last document from the source document.	<code>db.collection.aggregate([{\$group : {_id : \$&lt;Field Name&gt;, &lt;Field Label&gt; : {\$last : &lt;Field Name&gt;} }}])</code>

**Table 15.1: MongoDB Aggregate Operation – Expression Types**

## Examples and use of aggregation method

Before we start using aggregation, let's populate our new collection with the new data.

In our example, we have created a variable named BPBBooksBestSellingEditionsAggregation. The code for the same is as follows:

### Code 1

```
var BPBBooksBestSellingEditionsAggregation = [
    'Title': 'Cloud Computing',
    'Year': '2019',
    'ISBN': '9789388511407',
    'Pages': 330,
    'Weight': '570gm',
```

```
'Dimension': '23x19x1.5cm',
'Tags': ['Cloud Computing', 'Cloud Computing Concepts', 'Non
Programming'],
'InStock': [{{
    'Type': 'Paperback',
    'Quantity': 3000
},
{
    'Type': 'Hardcover',
    'Quantity': 1500
}
],
'SpecialOfferDiscount': '100'
}, {
    'Title': 'Introduction to Digital Marketing 101 : Easy to Learn
and Implement Hands-on Guide for Digital Marketing',
    'Year': '2019',
    'ISBN': '9789389328189',
    'Pages': 464,
    'Tags': ['Digital Marketing', 'Digital Marketing Tips', 'Non
Programming'],
    'InStock': [{{
        'Type': 'Paperback',
        'Quantity': 4000
},
{
        'Type': 'Hardcover',
        'Quantity': 2300
}
]
}, {
    'Title': 'IOT and Smart Cities: Your Smart City Planning
Guide',
    'Year': '2019',
    'ISBN': '9789388511322',
    'Pages': 242,
    'Weight': '357gm',
    'Dimension': '22.5x15x1.5gm',
```

```
'Tags': ['Internet of Things', 'IoT', 'Smart City', 'Planning Guide', 'Non Programming'],
'InStock': [{  
    'Type': 'Paperback',  
    'Quantity': 2000  
},  
{  
    'Type': 'Hardcover',  
    'Quantity': 1000  
}  
],  
'SpecialOfferDiscount': '200'  
, {  
    'Title': 'Machine Learning with Python',  
    'Year': '2018',  
    'ISBN': '9789386551931',  
    'Pages': 267,  
    'Tags': ['Python', 'Machine Learning', 'Python Programming',  
    'Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 4500  
},  
{  
    'Type': 'Hardcover',  
    'Quantity': 1300  
}  
]  
, {  
    'Title': 'Programming In Python',  
    'Year': '2018',  
    'ISBN': '9789386551276',  
    'Pages': 267,  
    'Tags': ['Python', 'Machine Learning', 'Python Programming',  
    'Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 4000  
},
```

```
{  
    'Type': 'Hardcover',  
    'Quantity': 1500  
}  
]  
, {  
    'Title': 'Artificial Intelligence Ethics and International Law:  
An Introduction',  
    'Year': '2019',  
    'ISBN': ' 9789388511629',  
    'Pages': 188,  
    'Weight': '268gm',  
    'Tags': ['Artificial Intelligence', 'International Law', 'AI',  
    'Artificial Intelligence Ethics', 'Non Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 5200  
    },  
    {  
        'Type': 'Hardcover',  
        'Quantity': 3300  
    }  
]  
, {  
    'Title': 'A Practical Approach for Machine Learning and Deep  
Learning Algorithms',  
    'Year': '2019',  
    'ISBN': '9789388511131',  
    'Pages': 280,  
    'Weight': '424gm',  
    'Tags': ['Machine Learning', 'Deep Learning', 'Algorithms',  
    'Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 2800  
    },  
    {  
        'Type': 'Hardcover',  
        'Quantity': 1250
```

```

        }
    ] ,
    'SpecialOfferDiscount': '150'
}, {
    '_id': '20021111',
    'Title': 'Introduction to Database Management',
    'Year': '2002',
    'ISBN': ' 9788176566384',
    'Pages': 342,
    'Tags': ['Database Management', 'DBMS', 'Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 4000
        },
        {
            'Type': 'Hardcover',
            'Quantity': 3450
        }
    ]
}];

```

And then, we have used the MongoDB `insert()` method to create a new document in the MongoDB collection `BPBOnlineBooksCollectionAggregation`. The code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksCollectionAggregation.insert(BPBBooksBestSellingEditionsAggregation);
```

```

Administrator Command Prompt - mongo
...
    "Year": 2019,
    "ISBN": "9789385111131",
    "Pages": 450,
    "Weight": "454gms",
    "Tags": ["Machine Learning", "Deep Learning", "Algorithms", "Programming"],
    "InStock": [
        {
            "Type": "Paperback",
            "Quantity": 2800
        },
        {
            "Type": "Hardcover",
            "Quantity": 1250
        }
    ],
    "SpecialOfferDiscount": "150"
},
{
    "_id": "20021111",
    "Title": "Introduction to Database Management",
    "Year": 2002,
    "ISBN": "9788176566384",
    "Pages": 342,
    "Tags": ["Database Management", "DBMS", "Programming"],
    "InStock": [
        {
            "Type": "Paperback",
            "Quantity": 4000
        },
        {
            "Type": "Hardcover",
            "Quantity": 3450
        }
    ]
];
> db.BPBOnlineBooksCollectionAggregation.insert(BPBBooksBestSellingEditionsAggregation); ①
bulkWriteResult: [
    {
        "writeErrors": [ ],
        "writeConcernErrors": [ ],
        "nInserted": 2,
        "nUpserted": 0,
        "nMatched": 0,
        "nModified": 0,
        "nRemoved": 0,
        "upserted": [ ]
    }
]
}

```

**Figure 15.1: Creating a new Collection and Inserting a new Data**

Since we are done with creating a new collection and inserting new documents in it, let us now use aggregation with our new collection.

## Examples of aggregation

The following are some examples of aggregation. You may run different examples by changing the expression type, fields, and values.

## The MongoDB \$group operator

What we are doing by using the aggregation operation is grouping the documents in the collections based on some fields and then using the expression types to output the result performed in the documents by grouping.

We use the MongoDB `$group` operator in the aggregate method to perform the aggregation operation.

## Example 1 - \$sum aggregation expression type

In our example, we have used the `$sum` aggregation expression type in our collection `BPBOnlineBooksCollectionAggregation` with the help of `$group` operator in the field `Year` to group all the documents together by the field `Year`, and then, output their total sum. The code and the

expression type details for the same are shown in the following screenshot:

## Code 1

```
db.BPBOnlineBooksCollectionAggregation.aggregate([ {
    $group: {
        _id: $Year,
        Number of Published Books: {
            $sum: 1
        }
    }
}]);
```

## Expression type details

Expression type	Expression type description	Expression type use and syntax
\$sum	Sums up the defined value from all the documents in a collection.	db.collection.aggregate([{\$group : {_id : \$<Field Name>, <Field Label> : {\$sum : <Field Name, Number or Operation>}}}] )

*Table 15.2: \$sum aggregation expression type - Details*

The screenshot shows a terminal window titled "Administrator Command Prompt - mongo". Inside, a MongoDB aggregation pipeline is run against the "BPBOnlineBooksCollectionAggregation" database. The pipeline consists of two stages: a "\$group" stage where documents are grouped by year and the count of published books is summed, and an output stage showing the results. A red box highlights the "\$sum: 1" part of the second stage. A red callout box at the bottom right of the terminal window reads "Number of Published Books Grouped by their respective Year of Publication".

```
> db.BPBOnlineBooksCollectionAggregation.aggregate([
    { $group: {
        _id: "$Year",
        "Number of Published Books": {
            $sum: 1
        }
    }
}]);
{
    "_id": "2019",
    "Number of Published Books": 5
}
{
    "_id": "2018",
    "Number of Published Books": 2
}
{
    "_id": "2002",
    "Number of Published Books": 1
}
```

*Figure 15.2: \$sum Aggregation Expression Type*

In our example, we saw how we can use the `$sum` aggregate expression type in the MongoDB `aggregate()` method.

## Example 2 - \$sum aggregation expression type with operation in group output

In our example, we have used the `$sum` aggregation expression type in our collection `BPBOnlineBooksCollectionAggregation` with the help of

`$group` operator in the field `Year` to group all the documents together by the field `Year`, and then, output their total sum multiplied by 10. The code and the expression type details for the same is shown in the following screenshot:

## Code 1

```
db.BPBOnlineBooksCollectionAggregation.aggregate([ {  
    $group: {  
        _id: "$Year",  
        "Number of Published Books Multiplied by 10": {  
            $sum: 1*10  
        }  
    }  
} ]);
```

_id	Number of Published Books Multiplied by 10
"2019"	50
"2002"	10
"2018"	20

**Figure 15.3: \$sum Aggregation Expression Types with Operation in Group Output**

In our example, we saw how we can use the `$sum` aggregate expression type with the operation in group output using the MongoDB `aggregate()` method.

## Example 3 - \$sum aggregation expression type with some other field

In our example, we have used the `$sum` aggregation expression type in our collection `BPBOnlineBooksCollectionAggregation` with the help of `$group` operator in the field `Year` to group all the documents together by the field `Year` and then using `$Pages` as the option to the `$sum` expression type to output the total number of pages from all the books according to their respective years. The code and the expression type details for the same is as shown in the following screenshot:

## Code 1

```

db.BPBOnlineBooksCollectionAggregation.aggregate([ {
    $group: {
        _id: $Year,
        Total Number of Pages Published in all the Books: {
            $sum: $Pages
        }
    }
} ]);

```

The screenshot shows the MongoDB command prompt with the following command:

```

db.BPBOnlineBooksCollectionAggregation.aggregate([
    {
        $group: {
            _id: "$Year",
            "Total Number of Pages Published in all the Books": {
                $sum: "$Pages"
            }
        }
    }
])

```

The output of the command is:

```

[{"_id": "2019", "Total Number of Pages Published in all the Books": 1564}, {"_id": "2018", "Total Number of Pages Published in all the Books": 534}, {"_id": "2002", "Total Number of Pages Published in all the Books": 342}]

```

A red box highlights the annotation: "Total Number of Pages in all the Books Grouped by their respective Year of Publication by using another Field as the Option".

**Figure 15.4: \$sum Aggregation Expression Type with Some Other Field**

In our example, we saw how we can use the `$sum` aggregate expression type with another field using the MongoDB `aggregate()` method.

## **Example 4 - \$avg aggregation expression type**

In our example, we have used the `$avg` aggregation expression type in our collection `BPBOnlineBooksCollectionAggregation` with the help of `$group` operator in the field `Year` to group all the documents together by the field `Year`, and then, using `$Pages` as the option to the `$avg` expression type to output the average number of pages from all the books according to their respective years. The code and the expression type details for the same as shown in the following screenshot:

### **Code 1**

```

db.BPBOnlineBooksCollectionAggregation.aggregate([ {
    $group: {
        _id: $Year,
        Average Number of Pages Published in all the Books: {
            $sum: $Pages
        }
    }
} ]);

```

## Expression type details

Expression type	Expression type description	Expression type use and syntax
\$avg	Calculates the average values from all the documents in a collection.	db.collection.aggregate([{\$group : {_id : \$<Field Name>, <Field Label> : {\$avg : <Field Name, Number or Operation>}}}])

**Table 15.3: \$avg aggregation expression type - Details**

The screenshot shows a terminal window titled "Administrator Command Prompt - mongo". The command entered is:

```
> db.BPBOnlineBooksCollectionAggregation.aggregate([{$group : {_id : "$Year", <Field Label> : {$avg : <Field Name, Number or Operation>}}}])
```

The output shows three documents grouped by year, each with an average page count:

```
{ "_id" : "2019", "Average Number of Pages Published in all the Books" : 1504 }, { "_id" : "2001", "Average Number of Pages Published in all the Books" : 342 }, { "_id" : "2018", "Average Number of Pages Published in all the Books" : 534 }
```

A red callout box highlights the output area with the text: "Average Number of Pages in all the Books Grouped by their respective Year of Publication by using another Field as the Option".

**Figure 15.5: \$avg Aggregation Expression Type with Some Other Field**

In our example, we saw how we can use the `$avg` aggregate expression type with some another field using the MongoDB `aggregate()` method.

## Example 5 - \$max aggregation expression type

In our example, we have used the `$max` aggregation expression type in our collection `BPBOnlineBooksCollectionAggregation` with the help of `$group` operator in the field `Year` to group all the documents together by the field `Year`, and then, using `$Pages` as the option to the `$max` expression type to output the maximum pages of a book in the collection with respect to the year. The code and the expression type details for the same is shown in the following screenshot:

### **Code 1**

```
db.BPBOnlineBooksCollectionAggregation.aggregate([{$group : {_id : "$Year", Maximum Pages Published in a Book: {$max: "$Pages"} }}])
```

```
});
```

## Expression type details

Expression type	Expression type description	Expression type use and syntax
\$max	Gives the maximum of all the values of documents in a collection.	db.collection.aggregate([{\$group : {_id : \$<Field Name>, <Field Label> : {\$max : <Field Name, Number or Operation>}}}] )

*Table 15.4: \$max aggregation expression type - Details*

The screenshot shows a MongoDB shell window with the command:

```
> db.BPBOnlineBooksCollectionAggregation.aggregate([
...   {
...     $group: {
...       _id: "$Year",
...       "Maximum Pages Published in a Book": {
...         $max: "$Pages"
...       }
...     }
...   }
...]);
```

The output is:

```
{ "_id" : "2019", "Maximum Pages Published in a Book" : 464 },
{ "_id" : "2002", "Maximum Pages Published in a Book" : 342 },
{ "_id" : "2018", "Maximum Pages Published in a Book" : 267 }
```

A red callout box highlights the \$max expression in the aggregation stage, and another box contains the following text:

Maximum Number of Pages in all the Books  
Grouped by their respective Year of Publication by  
using another Field as the Option

*Figure 15.6: \$max aggregation expression type with some other field*

In our example, we saw how we can use the `$max` aggregate expression type with another field using the MongoDB `aggregate()` method.

## Example 6 - \$push aggregation expression type

In our example, we have used the `$push` aggregation expression type in our collection `BPBOnlineBooksCollectionAggregation` with the help of `$group` operator in the field `Year` to group all the documents together by the field `Year`, and then, using `$Tags` as option to the `$push` expression type to output all the tags used in the books with respect to the year. The code and the expression type details for the same is shown in the following screenshots respectively. We have used the MongoDB `pretty()` method to make our output look better.

### **Code 1**

```
db.BPBOnlineBooksCollectionAggregation.aggregate([
  {
    $group: {
      _id: "$Year",
      All the Tags used in the Books: {
        $push: "$Tags"
      }
    }
  }
]);
```

```

        $push: $Tags
    }
}
}]).pretty();

```

## Expression type details

Expression type	Expression type description	Expression type use and syntax
\$push	Inserts the values to an array of the resulting document.	db.collection.aggregate([{\$group : {_id : \$<Field Name>, <Field Label> : {\$push : <Field Name>}}}] )

**Table 15.5: \$push aggregation expression type - Details**



```

Administrator: Command Prompt - mongo
> db.BPBOnlinedooksCollectionAggregation.aggregate([
...   {$group: {
...     _id: "$year",
...     "All the Tags used in the Books": {
...       $push: "$Tags"
...     }
...   }}).pretty();
{
  "_id": "2019",
  "All the Tags used in the Books": [
    {
      "Cloud Computing",
      "Cloud Computing Concepts",
      "Non Programming"
    },
    {
      "Digital Marketing",
      "Digital Marketing Tips",
      "Non Programming"
    },
    {
      "Internet of Things",
      "IoT",
      "Smart City",
      "Planning Guide",
      "Non Programming"
    },
    {
      "Artificial Intelligence",
      "International Law",
      "AI",
      "Artificial Intelligence Ethics",
      "Non Programming"
    },
    {
      "Machine Learning",
      "Deep Learning",
      "Algorithms",
      "Programming"
    }
  ]
}

```

All the Tags which are used in all the Books  
Grouped by their respective Year of Publication by  
using another Field as the Option

**Figure 15.7: \$push Aggregation Expression Type with Some Other Field – 1st Screen**

```

Administrator: Command Prompt - mongo
{
    "Year": "2017",
    "Tags": [
        "Artificial Intelligence",
        "International Law",
        "AI",
        "Artificial Intelligence Ethics",
        "Non Programming"
    ],
    "Books": [
        {
            "Title": "Machine Learning",
            "Author": "Deep Learning",
            "Category": "Algorithms",
            "Tags": [
                "Programming"
            ]
        }
    ]
},
{
    "Year": "2018",
    "Tags": [
        "Database Management",
        "DBMS",
        "Programming"
    ],
    "Books": [
        {
            "Title": "Python",
            "Author": "Machine Learning",
            "Category": "Python Programming",
            "Tags": [
                "Programming"
            ]
        },
        {
            "Title": "Python",
            "Author": "Machine Learning",
            "Category": "Python Programming",
            "Tags": [
                "Programming"
            ]
        }
    ]
}
]
}

```

All the Tags which are used in all the Books  
Grouped by their respective Year of Publication by  
using another Field as the Option

**Figure 15.8:** \$push Aggregation Expression Type with Some Other Field – 2nd Screen

In our example, we saw how we can use the \$push aggregate expression type with some another field using the MongoDB aggregate() method.

## Example 7 - \$last aggregation expression type

In our example, we have used the \$last aggregation expression type in our collection BPBOnlineBooksCollectionAggregation with the help of \$group operator in the field Year to group all the documents together by the field Year, and then, using \$Title as the option to the \$last expression type to output the title of the last or the latest published books with respect to the year. The code and the expression type details for the same is shown in the following screenshot:

### Code 1

```

db.BPBOnlineBooksCollectionAggregation.aggregate([
    {
        $group: {
            _id: "$Year",
            Latest Title the Books Published: {
                $last: "$Title"
            }
        }
    }
])

```

```
 } ] ) ;
```

## Expression type details

Expression type	Expression type description	Expression type use and syntax
\$last	Gives the last document from the source document.	db.collection.aggregate([{\$group : {_id : \$<Field Name>, <Field Label> : {\$last : <Field Name>}}}] )

**Table 15.6: \$last aggregation expression type - Details**

The screenshot shows a Windows Command Prompt window titled "Administrator Command Prompt - mongo". Inside, a MongoDB aggregation query is run against a collection named "BPMOnlineBooksCollection". The query groups documents by year and then uses the \$last operator to select the latest title for each year. The output shows three groups: one for 2019 (title: "A Practical Approach for Machine Learning and Deep Learning Algorithms"), one for 2018 (title: "Programming In Python"), and one for 2002 (title: "Introduction to Database Management"). A red callout box highlights the output with the text: "Latest Titles of the Books Grouped by their respective Year of Publication by using another Field as the Option".

```
> db.BPMOnlineBooksCollection.aggregate([
... $group : {
...   _id : "$Year",
...   "Latest Title the Books Published": {
...     $last: "$title"
...   }
... }
... ])
[{"_id" : "2019", "Latest Title the Books Published" : "A Practical Approach for Machine Learning and Deep Learning Algorithms"}, {"_id" : "2018", "Latest Title the Books Published" : "Programming In Python"}, {"_id" : "2002", "Latest Title the Books Published" : "Introduction to Database Management"}]
```

**Figure 15.9: \$last Aggregation Expression Type**

In our example, we saw how we can use the \$push aggregate expression type with some another field using the MongoDB aggregate() method.

## Introduction to map-reduce

In the previous section, we learned that the aggregation method can group documents and perform various operations that are helpful when we need to filter some information out of the large set of documents and collections.

The basic concept of map-reduce is to first map each document, then reduce or filter these documents by logic or condition. These are usually done using the JavaScript functions.

MongoDB provides the `mapReduce()` method to use map-reduce.

## The mapReduce() method

```

db.collection.mapReduce(
  function() {
    emit(key, value);
  }, /* Map : Function to Map */
  function(key, values) {
    return reduceFunctionResult
  }, {
    /* Reduce : Function to Reduce */
    out: < collection or screen > ,
    query: < document > ,
    sort: < document > ,
    limit: < number >
  }
);

```

The following table is for your reference to help you to understand this function better:

Property	Property Description
Map	This is the JavaScript function to map a value with a key and also emit a key-value pair with respect to the documents.
Reduce	This JavaScript function reduces or groups all the documents having the same key with respect to the documents.
out	This option is used to locate the map-reduce query result. It can create a new collection or print results on the screen.
query	This option is optional and can be used as selection criteria to select the documents based on some query.
sort	This option is optional and can be used to sort the documents based on some query.
limit	This option is optional and can be used to limit the number of documents to be returned based on some query.

**Table 15.7: Map-Reduce Method Property Details**

How map-reduce function works?

1. Step 1 – The map-reduce method first queries the collection
2. Step 2 – It then maps the documents with the key-value pairs
3. Step 3 – In the last step, the reduction is done based on the keys that have multiple values which group them into the aggregated result.

Let us now use map-reduce with some practical examples. In our example, we have created a variable named BPBBooksBestSellingEditionsMapReduce. The code for the same is as follows:

## **Code 1**

```
var BPBBooksBestSellingEditionsMapReduce = [ {
    'Title': 'Cloud Computing',
    'Year': '2019',
    'ISBN': '9789388511407',
    'Pages': 330,
    'Weight': '570gm',
    'Dimension': '23x19x1.5cm',
    'Tags': ['Cloud Computing', 'Cloud Computing Concepts', 'Non Programming'],
    'InStock': [ {
        'Type': 'Paperback',
        'Quantity': 3000
    },
    {
        'Type': 'Hardcover',
        'Quantity': 1500
    }
],
    'SpecialOfferDiscount': '100'
}, {
    'Title': 'Introduction to Digital Marketing 101 : Easy to Learn and Implement Hands-on Guide for Digital Marketing',
    'Year': '2019',
    'ISBN': '9789389328189',
    'Pages': 464,
```

```
'Tags': ['Digital Marketing', 'Digital Marketing Tips', 'Non Programming'],
'InStock': [
    {
        'Type': 'Paperback',
        'Quantity': 4000
    },
    {
        'Type': 'Hardcover',
        'Quantity': 2300
    }
],
{
    'Title': 'IOT and Smart Cities: Your Smart City Planning Guide',
    'Year': '2019',
    'ISBN': '9789388511322',
    'Pages': 242,
    'Weight': '357gm',
    'Dimension': '22.5x15x1.5gm',
    'Tags': ['Internet of Things', 'IoT', 'Smart City', 'Planning Guide', 'Non Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 2000
        },
        {
            'Type': 'Hardcover',
            'Quantity': 1000
        }
    ],
    'SpecialOfferDiscount': '200'
},
{
    'Title': 'Machine Learning with Python',
    'Year': '2018',
    'ISBN': '9789386551931',
    'Pages': 267,
    'Tags': ['Python', 'Machine Learning', 'Python Programming', 'Programming'],
    'InStock': [

```

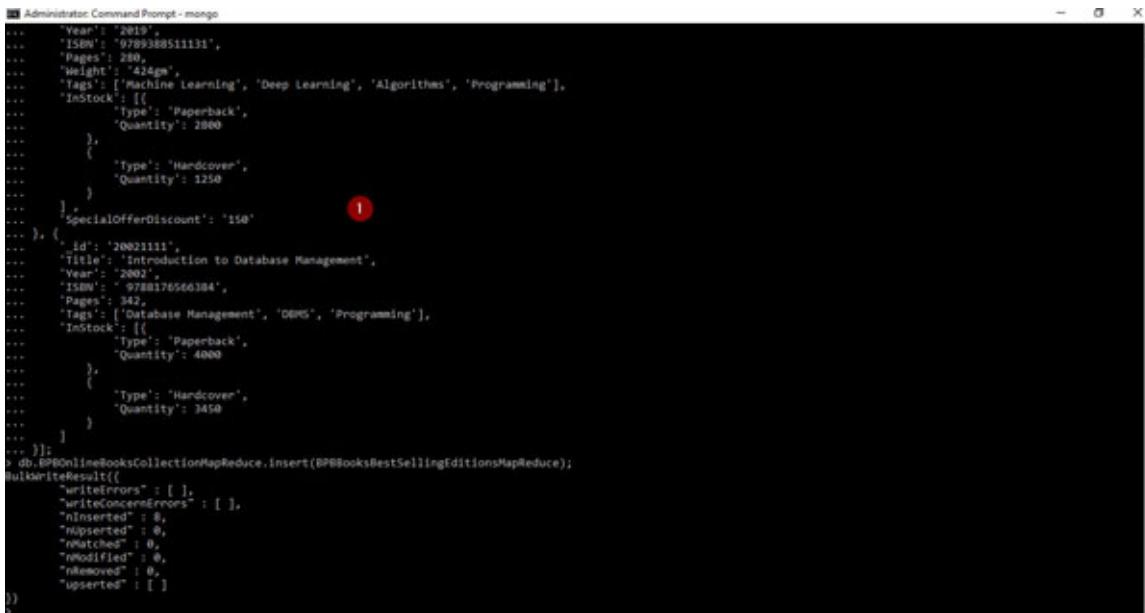
```
        'Type': 'Paperback',
        'Quantity': 4500
    },
    {
        'Type': 'Hardcover',
        'Quantity': 1300
    }
]
}, {
    'Title': 'Programming In Python',
    'Year': '2018',
    'ISBN': '9789386551276',
    'Pages': 267,
    'Tags': ['Python', 'Machine Learning', 'Python Programming',
    'Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 4000
        },
        {
            'Type': 'Hardcover',
            'Quantity': 1500
        }
    ]
}, {
    'Title': 'Artificial Intelligence Ethics and International Law:
An Introduction',
    'Year': '2019',
    'ISBN': ' 9789388511629',
    'Pages': 188,
    'Weight': '268gm',
    'Tags': ['Artificial Intelligence', 'International Law', 'AI',
    'Artificial Intelligence Ethics', 'Non Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 5200
        },
        {
            'Type': 'Hardcover',
            'Quantity': 1800
        }
    ]
}
```

```
        'Quantity': 3300
    }
]
}, {
    'Title': 'A Practical Approach for Machine Learning and Deep
Learning Algorithms',
    'Year': '2019',
    'ISBN': '9789388511131',
    'Pages': 280,
    'Weight': '424gm',
    'Tags': ['Machine Learning', 'Deep Learning', 'Algorithms',
'Programming'],
    'InStock': [{
        'Type': 'Paperback',
        'Quantity': 2800
    },
    {
        'Type': 'Hardcover',
        'Quantity': 1250
    }
],
    'SpecialOfferDiscount': '150'
}, {
    '_id': '20021111',
    'Title': 'Introduction to Database Management',
    'Year': '2002',
    'ISBN': ' 9788176566384',
    'Pages': 342,
    'Tags': ['Database Management', 'DBMS', 'Programming'],
    'InStock': [{
        'Type': 'Paperback',
        'Quantity': 4000
    },
    {
        'Type': 'Hardcover',
        'Quantity': 3450
    }
]
}
];
```

And then, we have used the MongoDB `insert()` method to create a new document in the MongoDB collection `BPBOnlineBooksCollectionMapReduce`. The code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksCollectionMapReduce.insert(BPBBooksBestSellingEditionsMapReduce);
```



```
Administrator Command Prompt - mongo
> db.BPBOnlineBooksCollectionMapReduce.insert(BPBBooksBestSellingEditionsMapReduce);
{
  "_id": "20021111",
  "Title": "Introduction to Database Management",
  "Year": "2007",
  "ISBN": "9788176566384",
  "Pages": 342,
  "Tags": ["Database Management", "DBMS", "Programming"],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 4000
    },
    {
      "Type": "Hardcover",
      "Quantity": 3450
    }
  ],
  "SpecialOfferDiscount": "15"
},
{
  "_id": "20021111",
  "Title": "Introduction to Database Management",
  "Year": "2007",
  "ISBN": "9788176566384",
  "Pages": 342,
  "Tags": ["Database Management", "DBMS", "Programming"],
  "InStock": [
    {
      "Type": "Paperback",
      "Quantity": 4000
    },
    {
      "Type": "Hardcover",
      "Quantity": 3450
    }
  ],
  "SpecialOfferDiscount": "15"
},
...
}
> db.BPBOnlineBooksCollectionMapReduce.insert(BPBBooksBestSellingEditionsMapReduce);
BulkWriteResult({
  "writeErrors": [],
  "writeConcernErrors": [],
  "nInserted": 2,
  "nUpserted": 0,
  "nMatched": 0,
  "nModified": 0,
  "nRemoved": 0,
  "nUpgraded": []
})
```

**Figure 15.10: Creating a new Collection and Inserting a new Data**

Since we are done with creating a new collection and inserting new documents in it, let us now use map-reduce with our new collection.

## Example 1 – mapReduce()

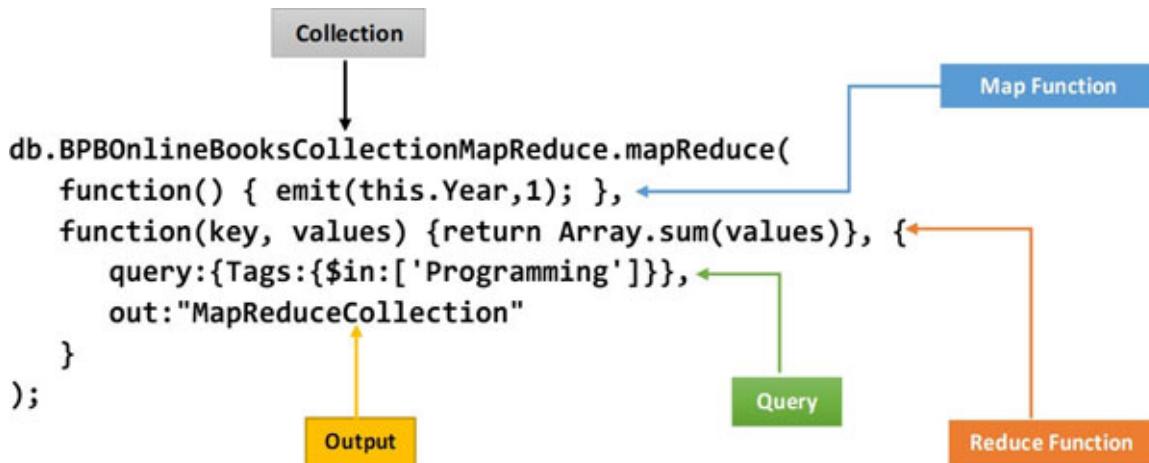
In our example, we have used the `mapReduce()` method in our collection `BPBOnlineBooksCollectionMapReduce` with the help of the map function to emit the field `Year`. Then, we have used the reduce function so that the deduction based on the query `{Tags:{$in:['Programming']}}}` reduces the documents having the field named `Tags` having `Programming` in them. The final output is the sum of all the documents grouped by year. With the `out` parameter, the output is saved in a new collection named `MapReduceCollection`.

The code and details for the same is shown in the following screenshots:

## Code 1

```
db.BPBOnlineBooksCollectionMapReduce.mapReduce(  
    function() {emit(this.Year,1); },  
    function(key, values) {return Array.sum(values)}, {  
        query:{Tags:{$in:['Programming']}},  
        out:MapReduceCollection  
    };
```

Let us understand this code with the help of a diagram:



**Figure 15.11: Map Reduce Example 1**

The output of the `mapReduce()` method is as follows:

```
Administrator: Command Prompt - mongo  
> db.BPBOnlineBooksCollectionMapReduce.mapReduce(  
...     function() { emit(this.Year,1); },  
...     function(key, values) {return Array.sum(values)}, {  
...         query:{Tags:{$in:['Programming']}},  
...         out:"MapReduceCollection"  
...     }  
... );  
({"result": "MapReduceCollection",  
 "timeMillis": 5161,  
 "counts": {  
     "input": 4,  
     "emit": 4,  
     "reduce": 1,  
     "output": 1  
 },  
 "ok": 1}
```

Total Number of Documents which has Tag as "Programming" are 4  
There are 2 records which has same Year so they are Grouped and Reduced to 1

**Figure 15.12: Map Reduce Example 1 – Output**

In our example, we saw how we can use map-reduce with the help of MongoDB `mapReduce()` method.

In the preceding code, the output result is stored in the new collection. If we want to display these results, we can simply use the `find()` method along with the `mapReduce()` method. The code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksCollectionMapReduce.mapReduce (
  function() {emit(this.Year,1)},
  function(key, values) {return Array.sum(values)}, {
    query:{Tags:{$in:['Programming']}},
    out:MapReduceCollection
  }
).find();
```

```
> db.BPBOnlineBooksCollectionMapReduce.mapReduce(
...   function() {emit(this.Year,1)};
...   function(key, values) {return Array.sum(values)}, {
...     query:{Tags:{$in:['Programming']}},
...     out:"MapReduceCollection"
...   }
... ).find();
{ "_id" : "2002", "value" : 1 }
{ "_id" : "2018", "value" : 2 }
{ "_id" : "2019", "value" : 1 }
```

Total Number of Documents which has Tag as "Programming" are 4

There are 2 records which has same Year (2018) so they are Grouped and Reduced to 1 and the sum of them are 2

**Figure 15.13: `mapReduce()` Method – Example 2 with `find()`**

In our example, we saw how we can use map-reduce with the help of MongoDB `mapReduce()` method in conjugation with the `find()` method. Here, if you check your collections using `show collections`, you will find that a new collection has been created by the `mapReduce()` method in your database, as shown in the following screenshot:

```
> db
> show collections
1 BPBOnlineBooksDB
2 BPBOnlineBooksCollection
3 BPBOnlineBooksCollectionAggregation
4 BPBOnlineBooksCollectionMapReduce
5 BPBOnlineBooksCollectionProjection
6 BPBOnlineBooksCollectionQuerySelectors
7 BPBOnlineBooksCollectionWithIndex
8 BPBOnlineBooksDBCollection
9 BPBOnlineBooksDBVCollection
10 BPBOnlineBooksDatatypesCollection
11 MapReduceCollection
```

"MapReduceCollection" created by the "out" Option of "mapReduce()" Method

**Figure 15.14: New Collection Created - `MapReduceCollection` created by the `out` Option of `mapReduce()` Method**

We have verified how a new collection is created using the `mapReduce()` method. If we don't want the `mapReduce()` method to create any new collection, there is a way to prevent that. We can use `{inline: 1}` as a

parameter to the `out` option. The code for the same is shown in the following screenshots:

### Code 3

```
db.BPBOnlineBooksCollectionMapReduce.mapReduce(  
    function() {emit(this.Year,1);},  
    function(key, values) {return Array.sum(values)}, {  
        query:{Tags:{$in:['Programming']}},  
        out: {inline: 1}  
    }  
) .find();
```

We are running this in steps:

1. Show DB
2. Show collections in DB
3. Drop (or delete) the `MapReduceCollection` collection
4. Run the `mapReduce()` method – *Code 3*

The screenshot shows a terminal window titled "Admin:Ubuntu Command Prompt - mongo". The command history is as follows:

```
> db ①  
BPBOnlineBooksDB  
> show collections ②  
BPBOnlineBooksCollection  
BPBOnlineBooksCollectionAggregation  
BPBOnlineBooksCollectionMapReduce  
BPBOnlineBooksCollectionProjection  
BPBOnlineBooksCollectionQuerySelectors  
BPBOnlineBooksCollectionWithIndex  
BPBOnlineBooksDBCollection  
BPBOnlineBooksDWCollection  
BPBOnlineBooksdatatypeCollection  
MapReduceCollection  
> db.MapReduceCollection.drop(); ③  
true  
> show collections  
BPBOnlineBooksCollection  
BPBOnlineBooksCollectionAggregation  
BPBOnlineBooksCollectionMapReduce  
BPBOnlineBooksCollectionProjection  
BPBOnlineBooksCollectionQuerySelectors  
BPBOnlineBooksCollectionWithIndex  
BPBOnlineBooksDBCollection  
BPBOnlineBooksDWCollection  
BPBOnlineBooksdatatypeCollection  
> db.BPBOnlineBooksCollectionMapReduce.mapReduce(  
...    function() { emit(this.Year,1); },  
...    function(key, values) {return Array.sum(values)}, { ④  
...        query:{Tags:{$in:['Programming']}},  
...        out: {inline: 1}  
...    }  
... ),Find();  
{  
    {  
        "_id": "2002",  
        "Value": 1  
    },  
    {  
        "_id": "2018",  
        "Value": 2  
    },  
    {  
        "_id": "2019",  
        "Value": 3  
    }  
}
```

A red box highlights the steps:

- 1) Show DB
- 2) Show Collections in DB
- 3) Drop (or Delete) "MapReduceCollection" Collection
- 4) Run `mapReduce()` Method

**Figure 15.15:** Running `mapReduce()` Method without creating additional collection

5. Check the output as printed on the screen
6. Verify that no additional collection has been created by the `mapReduce()` method

```

Administrator: Command Prompt - mongo
> db.MapReduceCollection.drop();
true
> show collections
SPBOnlineBooksCollection
SPBOnlineBooksCollectionAggregation
SPBOnlineBooksCollectionMapReduce
SPBOnlineBooksCollectionProjection
SPBOnlineBooksCollectionQuerySelectors
SPBOnlineBooksCollectionWithIndex
SPBOnlineBooksDDCCollection
SPBOnlineBooksDWVCollection
SPBOnlineBooksDataTypesCollection
> db.SPBOnlineBooksCollectionMapReduce.mapReduce(
...   function() { emit(this.Year,1); },
...   function(key, values) {return Array.sum(values);},
...   query:{Tags:[{in:["Programming"]}],},
...   out:{$_inline: 1}
... ),{find()};
[{"_id": "2002", "value": 3}, {"_id": "2016", "value": 2}, {"_id": "2019", "value": 1}]
5
6
> show collections
SPBOnlineBooksCollection
SPBOnlineBooksCollectionAggregation
SPBOnlineBooksCollectionMapReduce
SPBOnlineBooksCollectionProjection
SPBOnlineBooksCollectionQuerySelectors
SPBOnlineBooksCollectionWithIndex
SPBOnlineBooksDDCCollection
SPBOnlineBooksDWVCollection
SPBOnlineBooksDataTypesCollection

```

5) Check the Output as Printed on Screen  
 6) Verify that no additional collection is created by mapReduce() Method

**Figure 15.16:** Running `mapReduce()` Method without creating additional collection

We have verified how we can prevent the creation of a new collection using the `mapReduce()` method by using `{inline: 1}` as a parameter to the `out` option.

Let us explore the aggregation pipeline in MongoDB in the next section.

## Introduction to aggregation pipeline

MongoDB provides us a better way to filter the complex data, perform some operations on the data, and then, give us the final result out of the data. This process is termed as aggregation pipeline.

## What is pipeline?

In computer terminology, the pipeline is a process to execute a sequence of steps where the output of one step is input to the next step. All the steps are executed in the same sequence as defined to produce the final result as output.

The concept of pipeline is used in many ways in IT, like the execution of UNIX commands, wherein the commands are piped and executed in steps, taking the output of the previous steps as input, and giving their outputs to the next steps. We can also see the concept of pipeline in the server deployments, wherein during deployments on the servers or

on cloud, the application code is deployed on the servers using the pipeline methods and follows the same concept.

## MongoDB aggregation pipeline

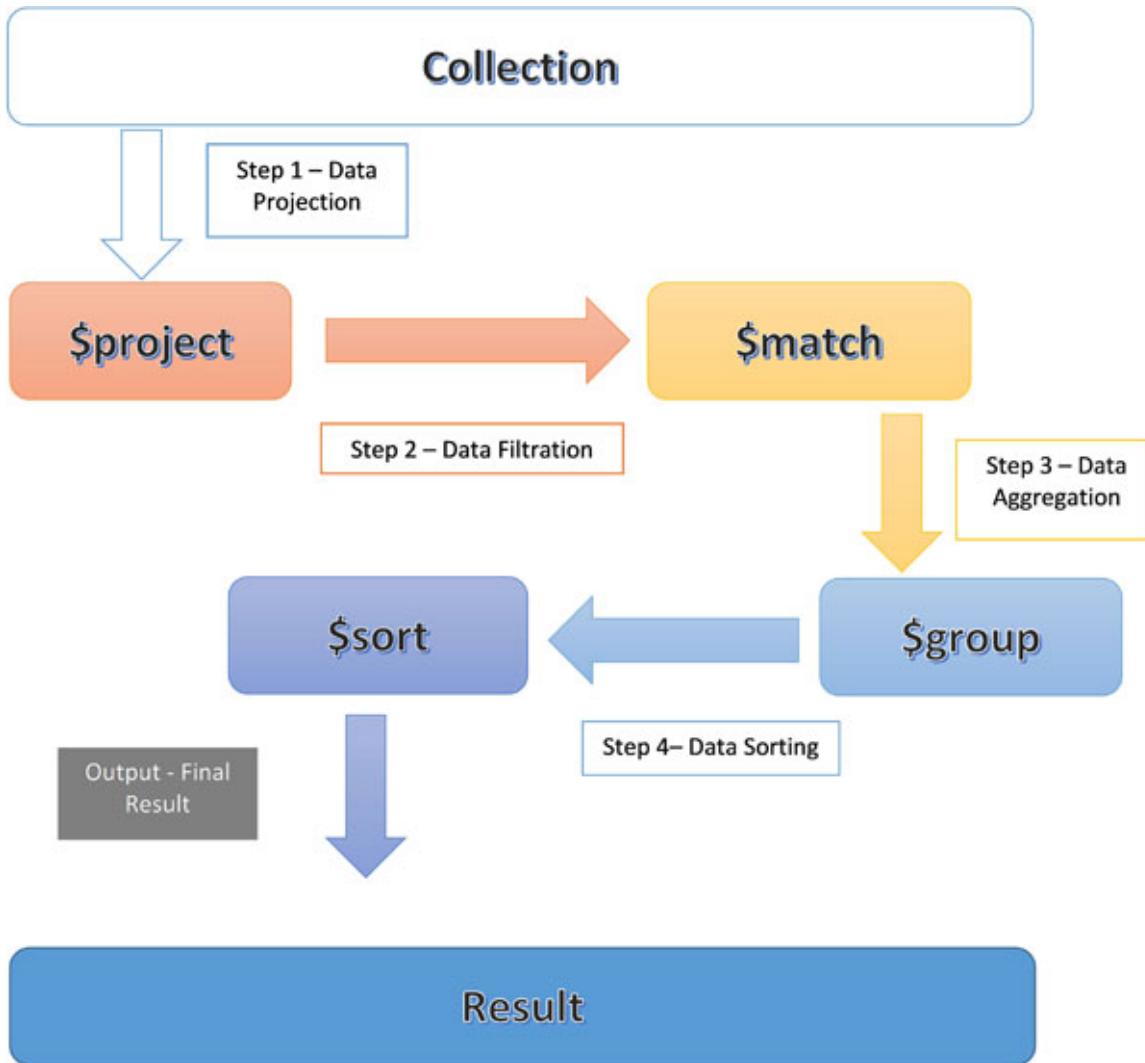
In MongoDB, the aggregation pipeline is better and flexible than the MongoDB `map-reduce()` method. The MongoDB aggregation pipeline is divided in the stages in which the data is projected, matched, manipulated, segregated, and grouped, before the final result is delivered.

The MongoDB aggregation pipeline framework has the following steps or stages, as shown in the following table:

Aggregation pipeline operator (step or stage)	Description
<code>\$project</code>	This stage is used to select the specific fields from a collection.
<code>\$match</code>	In this stage, a filtering operation is conducted on the documents, which then results in the reduction of the documents.
<code>\$group</code>	In this stage, the aggregation happens and the documents are grouped.
<code>\$sort</code>	This stage sorts the documents.
<code>\$skip</code>	By using this step, it is possible to skip and forward the documents.
<code>\$limit</code>	By using this step, we can limit the number of documents.
<code>\$unwind</code>	We can use this stage to have some documents added again for the next stage. In this stage, we can increase the number of documents for the next stage that are required sometimes to justify some logic.

**Table 15.8: MongoDB Aggregate Framework – Step or Stage**

Let us understand the MongoDB aggregation pipeline with the help of the following diagram:



**Figure 15.17: MongoDB Aggregation Pipeline Stages**

Let us now use the aggregation pipeline with some practical examples.

In our example, we have created a variable named `BPBBooksBestSellingEditionsAggregationPipeline`. The code for the same is as follows:

### Code 1

```
var BPBBooksBestSellingEditionsAggregationPipeline = [ {  
    'Title': 'Cloud Computing',  
    'Year': '2019',  
}
```

```
'ISBN': '9789388511407',
'Pages': 330,
'Weight': '570gm',
'Dimension': '23x19x1.5cm',
'Tags': ['Cloud Computing', 'Cloud Computing Concepts', 'Non Programming'],
'InStock': [
    {'Type': 'Paperback',
     'Quantity': 3000
    },
    {
        'Type': 'Hardcover',
        'Quantity': 1500
    }
],
'SpecialOfferDiscount': '100'
}, {
    'Title': 'Introduction to Digital Marketing 101 : Easy to Learn and Implement Hands-on Guide for Digital Marketing',
    'Year': '2019',
    'ISBN': '9789389328189',
    'Pages': 464,
    'Tags': ['Digital Marketing', 'Digital Marketing Tips', 'Non Programming'],
    'InStock': [
        {'Type': 'Paperback',
         'Quantity': 4000
        },
        {
            'Type': 'Hardcover',
            'Quantity': 2300
        }
    ]
}, {
    'Title': 'IOT and Smart Cities: Your Smart City Planning Guide',
    'Year': '2019',
    'ISBN': '9789388511322',
    'Pages': 242,
    'Weight': '357gm',
```

```
'Dimension': '22.5x15x1.5gm',
'Tags': ['Internet of Things', 'IoT', 'Smart City', 'Planning Guide', 'Non Programming'],
'InStock': [{  
    'Type': 'Paperback',  
    'Quantity': 2000  
,  
{  
    'Type': 'Hardcover',  
    'Quantity': 1000  
}  
],  
'SpecialOfferDiscount': '200'  
, {  
    'Title': 'Machine Learning with Python',  
    'Year': '2018',  
    'ISBN': '9789386551931',  
    'Pages': 267,  
    'Tags': ['Python', 'Machine Learning', 'Python Programming',  
    'Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 4500  
,  
{  
        'Type': 'Hardcover',  
        'Quantity': 1300  
}  
]  
}, {  
    'Title': 'Programming In Python',  
    'Year': '2018',  
    'ISBN': '9789386551276',  
    'Pages': 267,  
    'Tags': ['Python', 'Machine Learning', 'Python Programming',  
    'Programming'],  
    'InStock': [{  
        'Type': 'Paperback',  
        'Quantity': 4000
```

```
        },
        {
            'Type': 'Hardcover',
            'Quantity': 1500
        }
    ]
},
{
    'Title': 'Artificial Intelligence Ethics and International Law: An Introduction',
    'Year': '2019',
    'ISBN': ' 9789388511629',
    'Pages': 188,
    'Weight': '268gm',
    'Tags': ['Artificial Intelligence', 'International Law', 'AI', 'Artificial Intelligence Ethics', 'Non Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 5200
        },
        {
            'Type': 'Hardcover',
            'Quantity': 3300
        }
    ]
},
{
    'Title': 'A Practical Approach for Machine Learning and Deep Learning Algorithms',
    'Year': '2019',
    'ISBN': '9789388511131',
    'Pages': 280,
    'Weight': '424gm',
    'Tags': ['Machine Learning', 'Deep Learning', 'Algorithms', 'Programming'],
    'InStock': [
        {
            'Type': 'Paperback',
            'Quantity': 2800
        },
        {
            'Type': 'Hardcover',
            'Quantity': 1800
        }
    ]
}
```

```

        'Quantity': 1250
    }
],
'SpecialOfferDiscount': '150'
}, {
'_id': '20021111',
'Title': 'Introduction to Database Management',
'Year': '2002',
'ISBN': ' 9788176566384',
'Pages': 342,
'Tags': ['Database Management', 'DBMS', 'Programming'],
'InStock': [
    {
        'Type': 'Paperback',
        'Quantity': 4000
    },
    {
        'Type': 'Hardcover',
        'Quantity': 3450
    }
]
}];
}

```

And then, we have used the MongoDB `insert()` method to create new documents in the MongoDB collection `BPBBooksBestSellingEditionsAggregationPipeline`. The code for the same is shown in the following screenshot:

## Code 2

```
db.BPBOnlineBooksCollectionAggregationPipeline.insert(BPBBooksBest
SellingEditionsAggregationPipeline);
```

```

> db.BPBOnlineBooksCollection.create();
> db.BPBOnlineBooksCollection.insert([
  {
    "Title": "Machine Learning for Dummies",
    "Year": "2018",
    "ISBN": "9788120350001",
    "Pages": 300,
    "Tags": ["Machine Learning", "Deep Learning", "Algorithms", "Programming"],
    "InStock": [
      {
        "Type": "Paperback",
        "Quantity": 2000
      },
      {
        "Type": "Hardcover",
        "Quantity": 1250
      }
    ],
    "SpecialOfferDiscount": "15%"
  },
  {
    "_id": "20021111",
    "Title": "Introduction to Database Management",
    "Year": "2002",
    "ISBN": "9788176566384",
    "Pages": 342,
    "Tags": ["Database Management", "DBMS", "Programming"],
    "InStock": [
      {
        "Type": "Paperback",
        "Quantity": 4000
      },
      {
        "Type": "Hardcover",
        "Quantity": 3450
      }
    ]
  }
]);
> db.BPBOnlineBooksCollectionAggregationPipeline.insert(BPBBooksBestSellingEditionsAggregationPipeline);

```

**Figure 15.18: Creating a new Collection and Inserting a new Data**

Since we are done with creating a new collection and inserting new documents in it, let us now use the aggregation pipeline with our new collection.

## **Example 1 – aggregate()**

In our example, we have used the `aggregate()` method in our collection `BPBOnlineBooksCollectionAggregationPipeline`. With the help of the `$project` operator, we will project the `Year` and `Tags` fields. Then, with the help of the `$match` operator, we will match all the documents that have `tags` field having `programming` as a value `{Tags: {$in: ['Programming']}}` to reduce the documents. Then, with the help of the `$group` operator, we will group the documents. Here, with the help of the `$sum` operator, we will sum up all documents grouped by `Year`. Then, with the help of the `$sort`, operator the results will be sorted in descending order. All the results will then be printed on the screen using the `pretty()` method. So, by running these aggregation pipeline operations, we will print the sum of all the documents that have `Programming` as value in the `tags` field grouped by year.

### **Code 1**

```

db.BPBOnlineBooksCollectionAggregationPipeline.aggregate([
  {$project : {Year: 1, Tags: 1}},

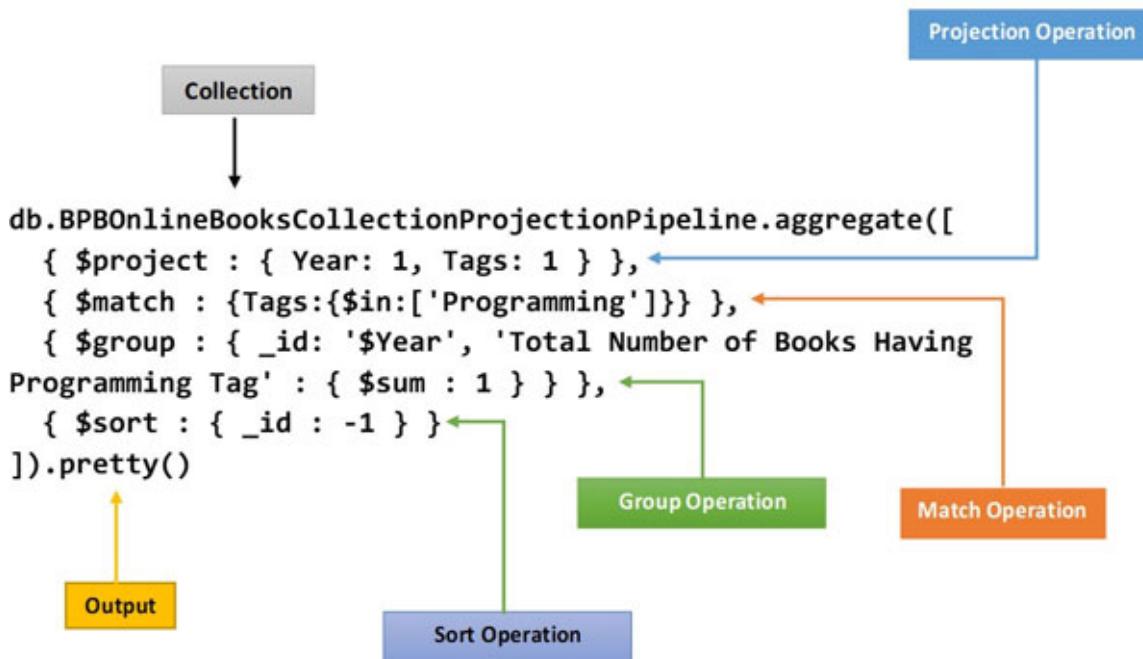
```

```

{$match : {Tags:{$in:['Programming']} }},
{$group : {_id: '$Year', 'Total Number of Books Having
Programming Tag' : {$sum : 1}}},
{$sort : {_id : -1}}
]).pretty()

```

Let us understand this code with the help of the following diagram:



**Figure 15.19: MongoDB Aggregation Pipeline - Stages**

This method will be executed by the following steps:

1. Projection stage
2. Matching or filtering stage
3. Grouping stage
4. Sorting stage
5. Printing the result or output

The output of the aggregate pipeline method is shown in the following screenshot:

```

db.BPBOnlineBooksCollectionAggregationPipeline.aggregate([
  { $project : { Year: 1, Tags: 1 } },
  { $match : {Tags:$in:['Programming']} } , ②
  { $group : { _id: '$Year', 'Total Number of Books Having Programming Tag' : { $sum : 1 } } } , ③
  { $sort : { _id : -1 } } ④
]).pretty()
[{"_id": "2019", "Total Number of Books Having Programming Tag": 1},
 {"_id": "2018", "Total Number of Books Having Programming Tag": 2},
 {"_id": "2002", "Total Number of Books Having Programming Tag": 1}]

```

1) Projection Stage  
 2) Matching or Filtering Stage  
 3) Grouping Stage  
 4) Sorting Stage  
 5) Printing the Result or Output

**Figure 15.20:** Aggregation Pipeline Example 1 – Output

In our example, we saw how we can use the MongoDB aggregation pipeline with the help of the MongoDB `aggregate()` method.

In the preceding code, the output result is printed instead of creating any new collection. If we want this operation to create a new collection, we need to add the `$out` operator in the `aggregate()` method. The code for the same is shown in the following screenshot:

## Example 2 – aggregate() with \$out

In our example, we have used the `aggregate()` method in our collection `BPBOnlineBooksCollectionAggregationPipeline`. With the help of the `$project` operator, we will project the `Year` and `Tags` fields. Then, with the help of the `$match` operator, we will match all the documents that have the `tags` field having `programming` as a value `{Tags:{$in: ['Programming'] }}` to reduce the documents. Then, with the help of the `$group` operator, we will group the documents. Here, with the help of the `$sum` operator, we will sum up all the documents grouped by year. Then, with the help of the `$sort` operator, the results will be sorted in descending order. All the results are stored into the new collection `AggregationPipelineCollection`.

By running this aggregation pipeline operation, we will print the sum of all the documents that have `Programming` as value in the `tags` field grouped by year.

The code and details for the same is shown in the following screenshot:

### Code 1

```
db.BPBOnlineBooksCollectionAggregationPipeline.aggregate([
  {$project : {Year: 1, Tags: 1}},
  {$match : {Tags:{$in:['Programming']}}},
```

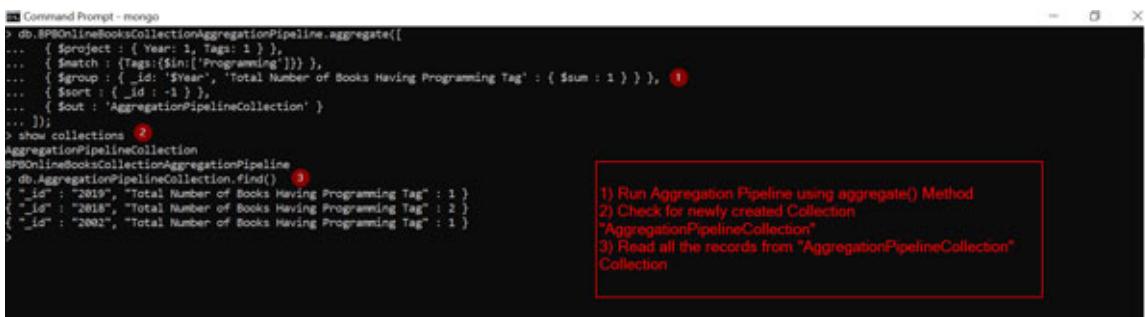
```

    {$group : { _id: '$Year', 'Total Number of Books Having
Programming Tag' : { $sum : 1 } } },
    {$sort : { _id : -1 } },
    { $out : 'AggregationPipelineCollection' }
  );

```

We will do the following steps here:

1. Run the aggregation pipeline using the `aggregate()` method (use the preceding code 1).
2. Check for the newly created collection `AggregationPipelineCollection` using the `show collections` command.
3. Read all the records from the `AggregationPipelineCollection` collection using the `db.AggregationPipelineCollection.find()` method.



The screenshot shows a MongoDB command prompt window. The user has run an aggregation pipeline to create a new collection named `AggregationPipelineCollection`. The pipeline stages are:

- `$project`: Projects Year and Tags.
- `$match`: Matches documents where Tags contain 'Programming'.
- `$group`: Groups by Year and calculates the sum of books having the 'Programming' tag.
- `$sort`: Sorts by Year in descending order.
- `$out`: Outputs the result to the `AggregationPipelineCollection`.

After running the pipeline, the user checks the collections with `show collections`, which lists `AggregationPipelineCollection`. Finally, the user runs `db.AggregationPipelineCollection.find()` to see the results, which are:

```

[{"_id": "2019", "Total Number of Books Having Programming Tag": 1},
 {"_id": "2018", "Total Number of Books Having Programming Tag": 2},
 {"_id": "2002", "Total Number of Books Having Programming Tag": 1}
]

```

A red box highlights the last three lines of the command history, which correspond to the three steps listed in the text above.

**Figure 15.21: aggregate() Method with \$out Option**

We saw in the preceding code that the output result has created a new collection `AggregationPipelineCollection` with the help of the `$out` operator in the `aggregate()` method.

## Conclusion

In this chapter, we learned about the MongoDB aggregation and the benefits of using it. We also learned about the aggregation expression types and how to use those using the step-by-step practical examples. We also covered the map-reduce method in MongoDB and the benefits of using map-reduce with the help of practical examples to understand it better. Further in this chapter, we covered the aggregation pipeline and its benefits and learned the working of the aggregation pipeline in MongoDB in a very detailed manner. In the last part of this chapter, we

did some step-by-step practical examples to understand the aggregation pipeline in a better way.

In the next chapter of this book, we will learn about the MongoDB compass which is the official GUI client for MongoDB. We will also learn to install the MongoDB compass using step-by-step method and to use the MongoDB compass GUI application to connect to the MongoDB server and perform various database related operations.

## **Questions**

1. What is aggregation and why is it useful?
2. List any three aggregation expression types in MongoDB.
3. How can we use aggregation with the \$sum aggregation expression type operator in MongoDB?
4. How can we use map-reduce in MongoDB?
5. What are the benefits of using map-reduce in MongoDB?
6. What are the benefits of the aggregation pipeline?
7. How does the aggregation pipeline work in MongoDB? Give an example with the help of the stages in the aggregation pipeline.

## CHAPTER 16

# MongoDB Data Manipulations Using MongoDB Compass

This chapter covers the MongoDB Compass. We will learn about the MongoDB Compass and the benefits of using it. We will also learn how to install it on our system using step-by-step method. Later in this chapter, we will learn about the usage of MongoDB Compass to connect to the MongoDB server. In the last part of this chapter, we will cover the MongoDB Compass with some step-by-step practical examples to give us more idea on what we can do with the MongoDB Compass.

### Structure

In this chapter, we will discuss the following topics:

- Introduction to MongoDB Compass
- Installing MongoDB Compass
- Connecting MongoDB server with MongoDB Compass
- Practical examples with MongoDB Compass

### Objectives

After studying this unit, you will should be able to understand about the MongoDB Compass and benefits of MongoDB Compass. We will also learn how to install MongoDB Compass. Later in this chapter we will learn how we can connect MongoDB server using MongoDB Compass. In the last section of this chapter we will learn MongoDB Compass using practical examples.

### Introduction to MongoDB Compass

The MongoDB Compass is the **GUI (Graphical User Interface)** tool to connect to the MongoDB server very easily and do a lot of things using the GUI which are time consuming with the commands or queries. The MongoDB Compass also provides many features to visualize and manipulate the data in the collections. The MongoDB Compass is more than just a visual GUI client or data manipulation tool.

Some of the features that the MongoDB Compass provides are as follows:

- Rich visual GUI interface to connect to the MongoDB server
- Rich visual GUI interface to visualize the MongoDB databases, collections, and documents
- Rich visual GUI interface to manipulate the MongoDB databases, collections, and documents
- Easy interface to create, update as well as delete databases, collections, and documents
- We can easily perform the CRUD operations on the documents with the help of the MongoDB Compass visual editing tools
- With the help of MongoDB Compass, we can optimize our database and application performance using visual graphs to understand how our queries perform
- The MongoDB Compass also helps in the management of indexes. We can easily view the existing indexes and their properties and we can easily add a new index or remove existing index from the collection using the MongoDB Compass
- With the help of MongoDB Compass, we can also validate our data with the help of JSON schema validation rules. For this purpose, MongoDB provides us the editor which auto suggests the field names, BSON data types and more.
- MongoDB Compass is a very good tool for the aggregation pipeline. By using the drag and drop features to add the stages to the MongoDB aggregation pipeline, we can easily check if the aggregation pipeline is working as per the expected behavior and then we can export this code to our application
- The MongoDB Compass has a lot of plugins available so that you can add more features to it and it is extensible as per our needs

## Installing MongoDB Compass

MongoDB Compass is available for all the major operating systems, such as Windows, Linux, and MacOS. For this chapter, we will use Windows OS to install and explore MongoDB Compass. If you would like to use the MongoDB Compass for other operating systems like Linux or MacOS, you can follow the same steps. You just need to select the operating system of your choice. The basic interface of the MongoDB Compass is almost similar for every OS.

Let us explore how to download, install, and setup the MongoDB Compass on machines running on the Windows OS.

## Installing MongoDB Compass on Windows operating system

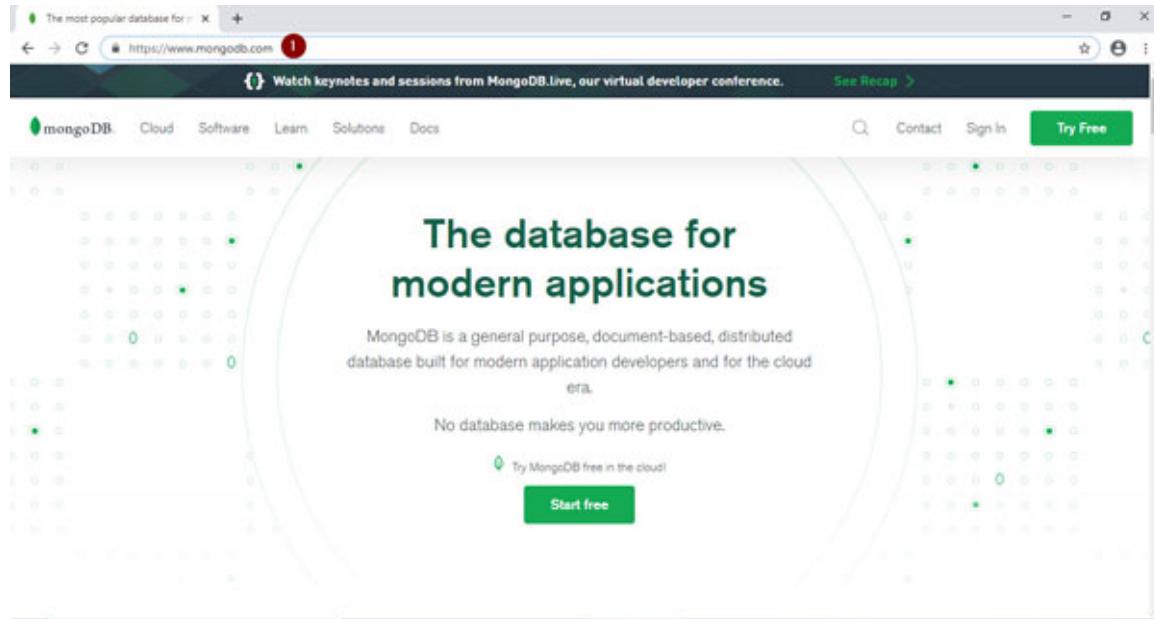
We will show you how you can install MongoDB Compass on the Windows operating system. We will use the default installation method to install MongoDB Compass on the machines that run on Windows operating system.

Installation steps

Let us start with the installation of MongoDB on our machine. The steps required to install MongoDB Compass are as follows:

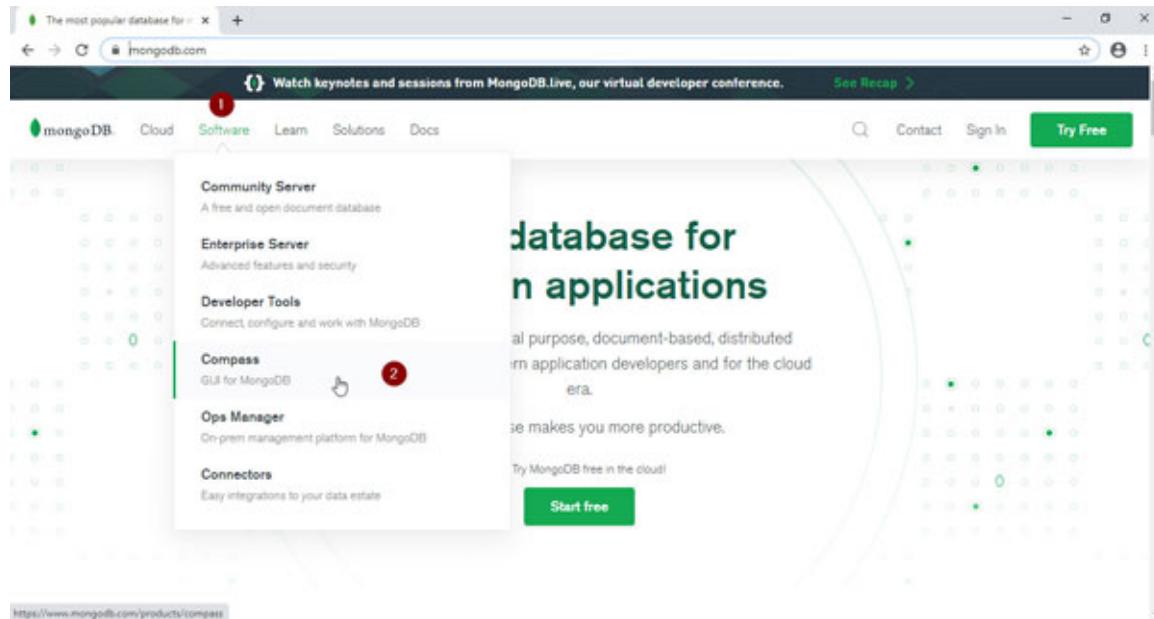
### **Step 1 – Download MongoDB Compass**

1. Open the MongoDB Inc. Official Website – <https://www.mongodb.com/> in your favorite browser, as shown in the following screenshot:



**Figure 16.1:** MongoDB Inc. Official Website Home Page

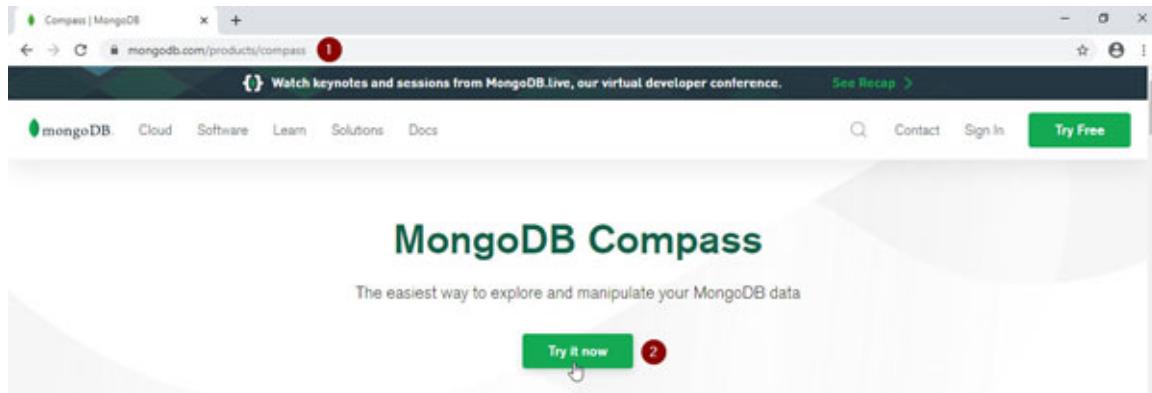
2. Click the **Software** link on the top navigation of the home page and then click **Compass** from the drop down menu, as shown in the following screenshot:



**Figure 16.2:** MongoDB Inc. Official Website Home Page – Software Menu > Compass

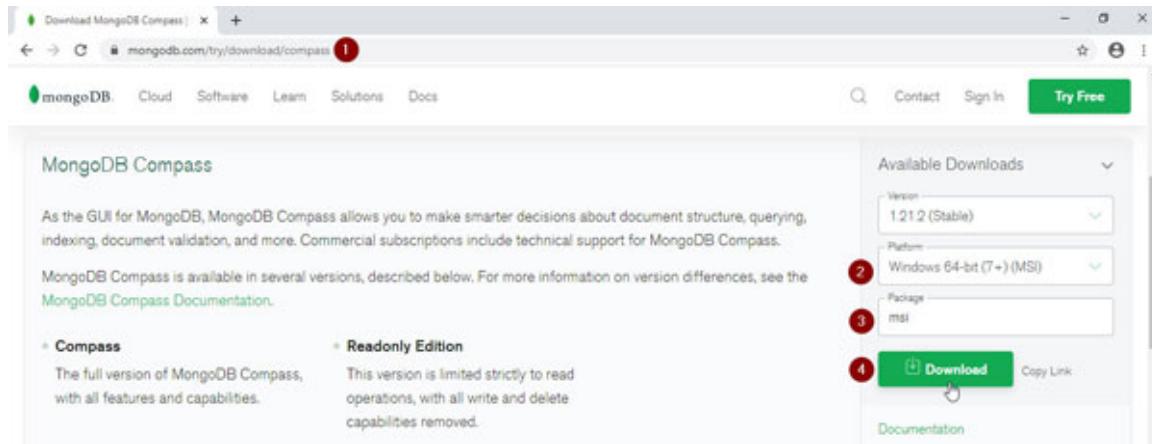
3. This will open the MongoDB Compass page. Click the green button which says **try it now**, as shown in the following

screenshot:



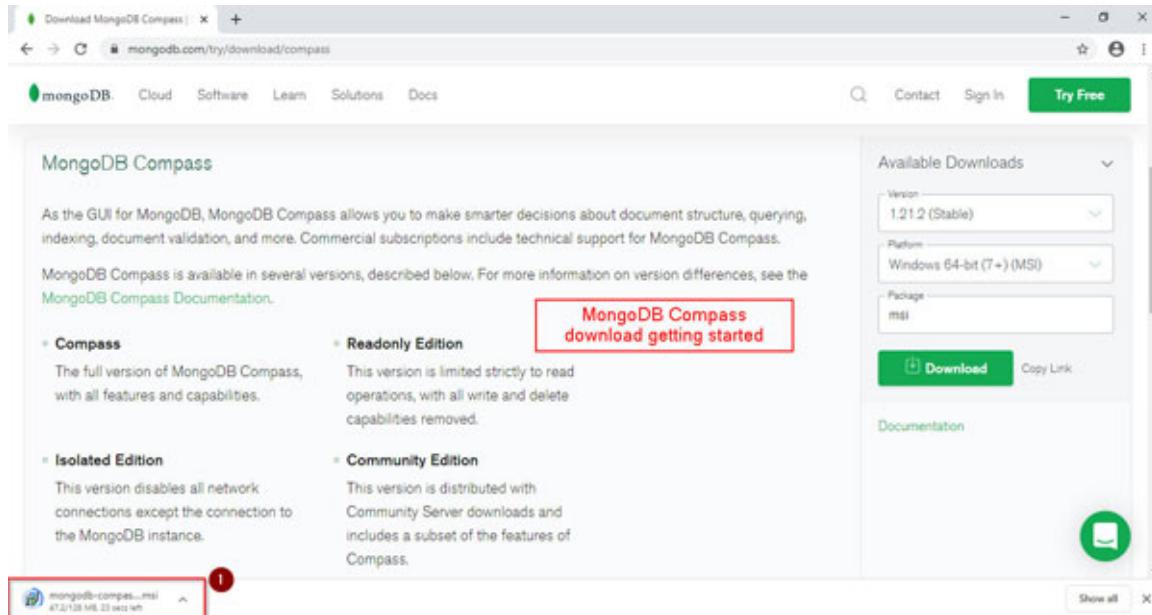
**Figure 16.3: MongoDB Inc. – MongoDB Compass Page**

4. This will open the MongoDB Compass download page. This page will auto detect your OS type. You can select any platform according to your OS. For our example, we have chosen Windows 64-bit MSI method, as shown in the following screenshot:



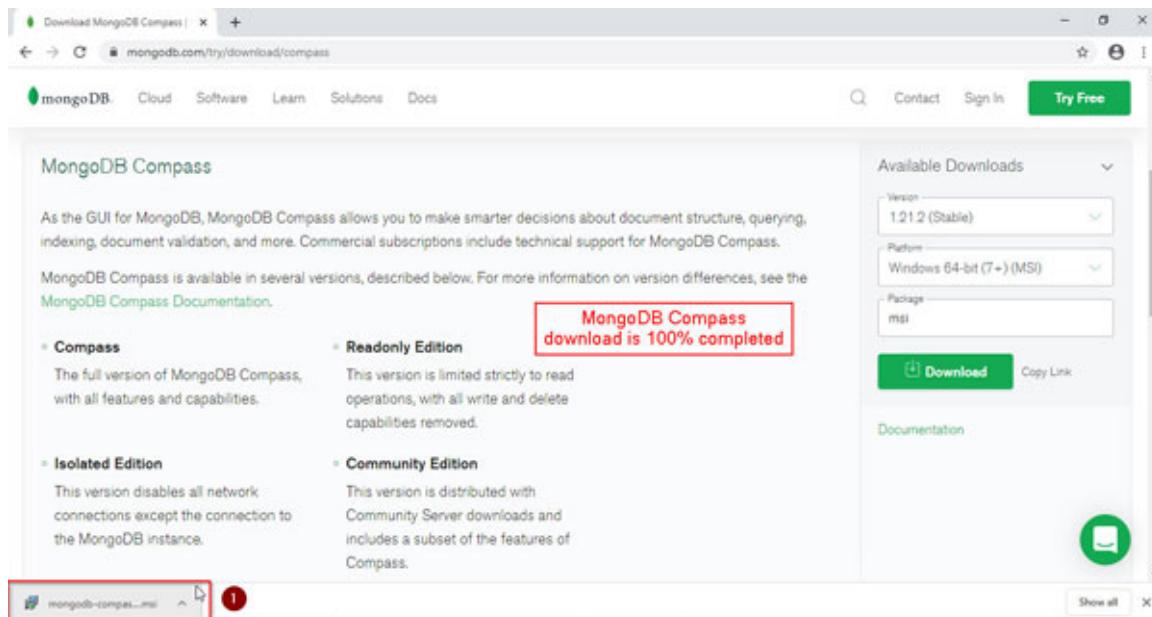
**Figure 16.4: MongoDB Inc. – MongoDB Compass Download Page**

5. Now, click on the **Download** button and MongoDB Compass download will start automatically, as shown in the following screenshot:



**Figure 16.5: MongoDB Compass Download Page – Download Started**

- Once the download starts, you can easily see the download process with the download icon and progress on your browser (this progress shows different in every browser. The screenshot is of Google Chrome. Every browser shows this in different manner). Wait till the download is 100% complete, as shown in the following screenshot:



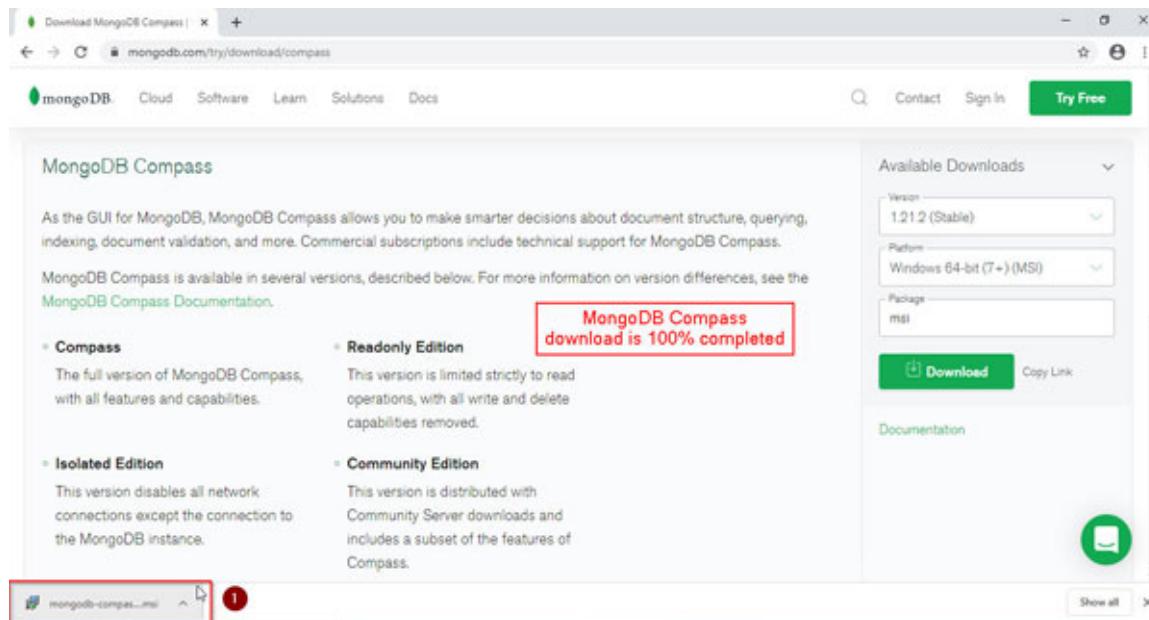
**Figure 16.6: MongoDB Compass Download Page – Download 100% Complete**

- Once the download is 100% complete, follow the next steps, as shown in the following screenshot:

In step 1, we covered how to download MongoDB Compass from the official website. The next steps are related to the installation process covered in step 2 of the MongoDB Compass installation process.

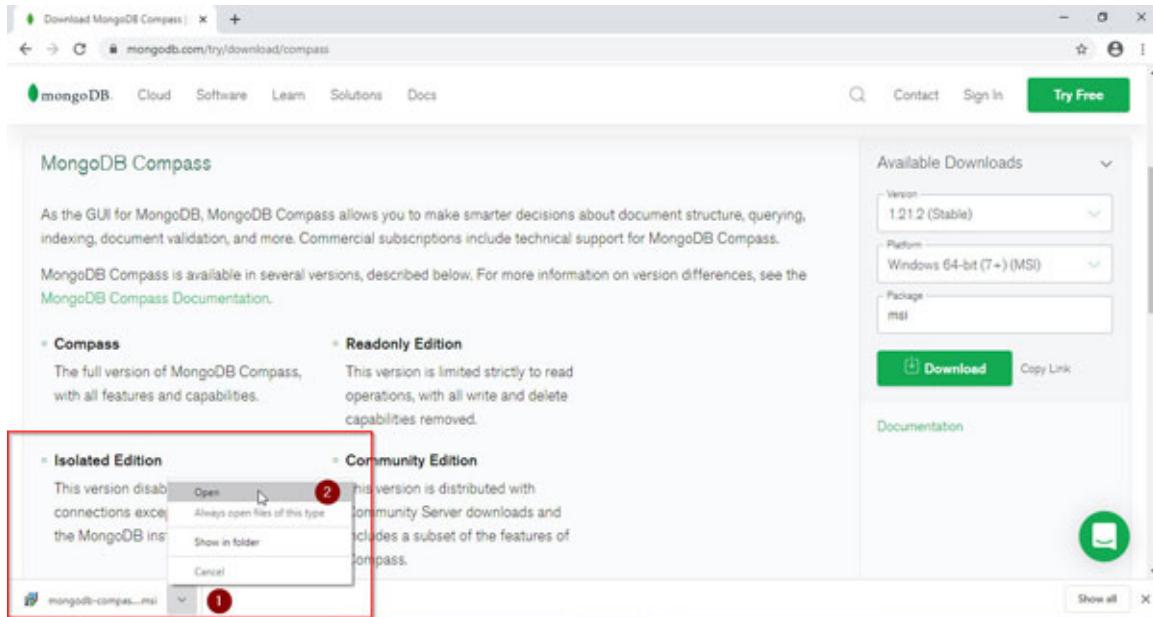
## Step 2 – Install MongoDB Compass on your Windows machine

Once the download is complete and the installer file (MSI File) is fully downloaded, it will show a download complete icon, as shown in the following screenshot, and you can proceed further.



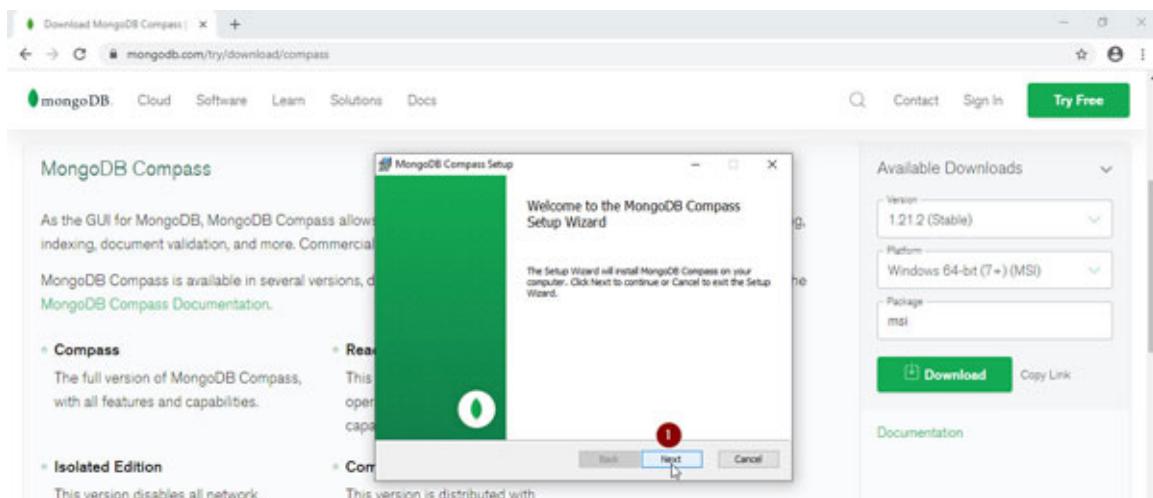
**Figure 16.7: MongoDB Compass Download Page – Download 100% Complete – Next Steps**

- Now, open this MSI file, which is a Windows installer file (previously known as Microsoft Installer) and it will start MongoDB setup wizard which will guide you to complete the installation of MongoDB Compass in your machine, as shown in the following screenshot:



**Figure 16.8: MongoDB Compass Installation Process – Open MongoDB MSI File.**

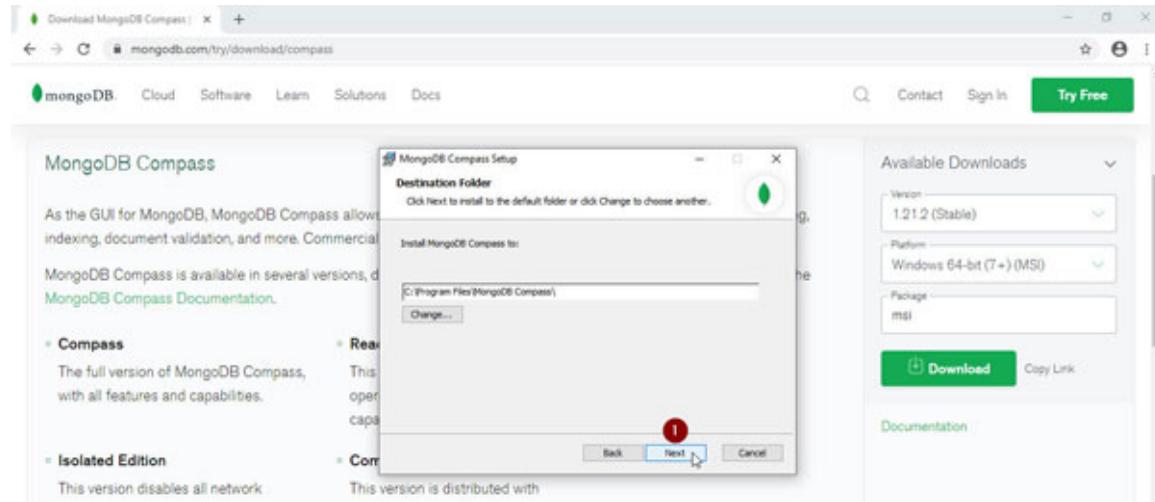
- Once you click on Open, you will see the startup screen of the installation wizard. You will see a few buttons which are easy to understand and you may cancel or go back to previous steps at any point of time. In order to install MongoDB Compass on your system, press the Next button, as shown in the following screenshot:



**Figure 16.9: MongoDB Compass Installation Wizard – Startup Screen**

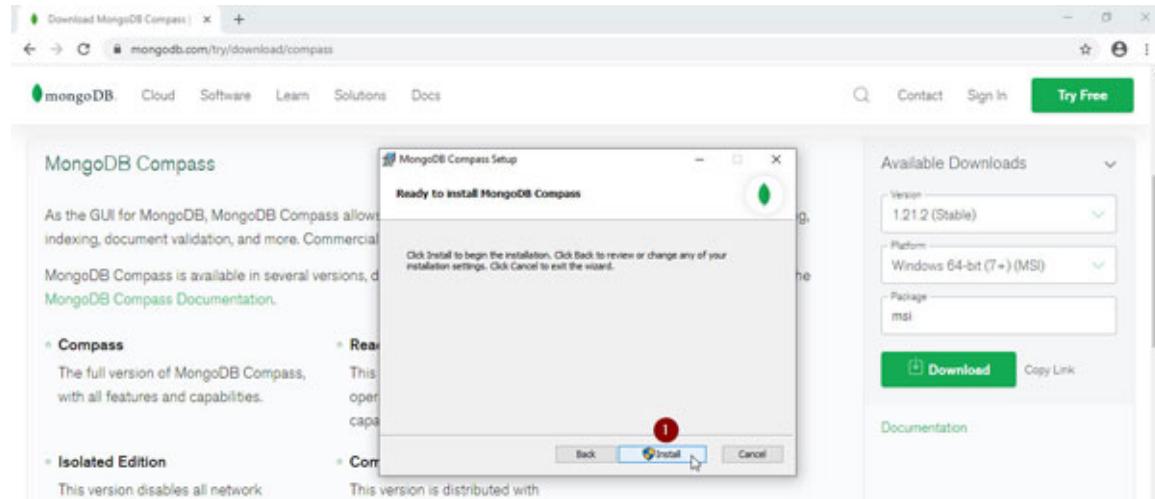
- Once you click on the Next button, you will see the next screen of the installation wizard which will display the current destination

folder where the MongoDB Compass will get installed. You may change the location if you want to or keep this as it is and press the **Next** button, as shown in the following screenshot:



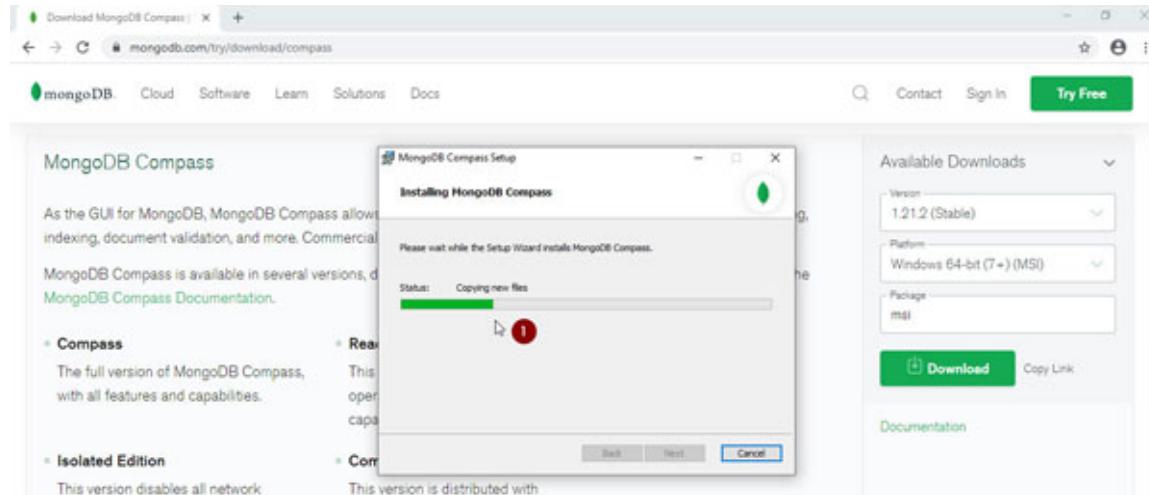
**Figure 16.10: MongoDB Compass Installation Wizard – Choose Destination Folder**

4. Once you click on the **Next** button, you will see the next screen of the installation wizard. Click on the **Install** button to install MongoDB Compass, as shown in the following screenshot:



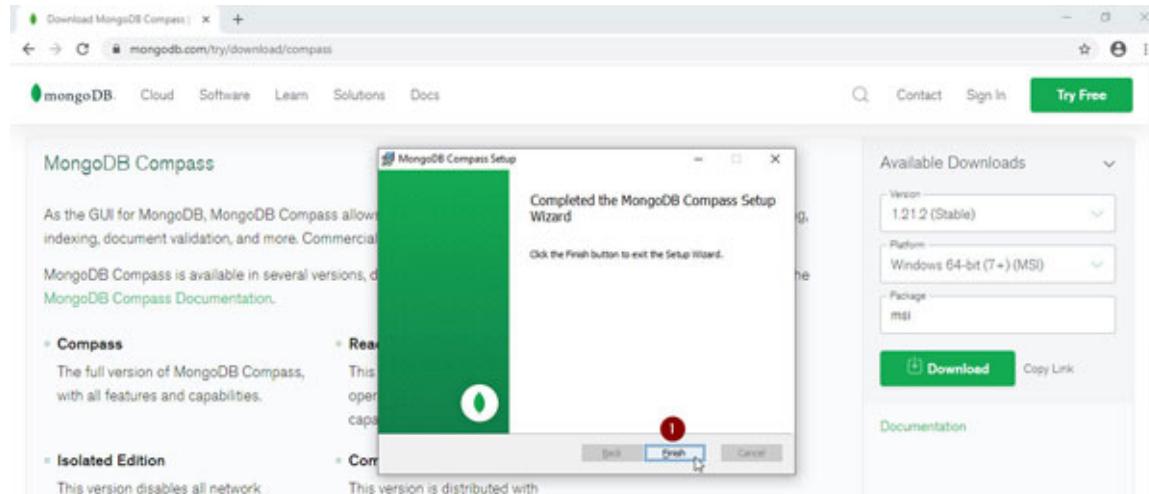
**Figure 16.11: MongoDB Compass Installation Wizard – Install**

5. Once you click on the **Install** button, you will see a setup wizard which will install the MongoDB Compass related files, as shown in the following screenshot:



**Figure 16.12:** MongoDB Compass Installation Wizard – Setup Wizard Installing Files

- Once this installation is 100% complete, it will show you the next screen with a `Finish` button, as shown in the following screenshot. Once you click on the `Finish` button, the installation will be finished. Click the `Finish` button to complete the setup process. Now, MongoDB Compass is installed on your computer.

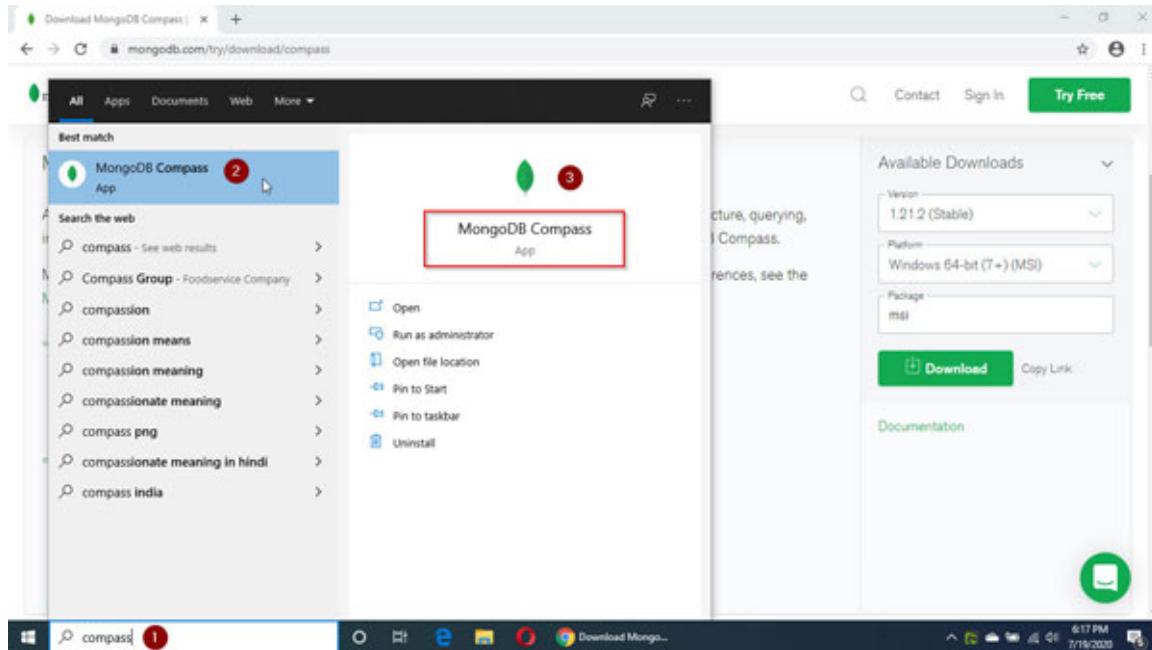


**Figure 16.13:** MongoDB Compass Installation Wizard – Complete Install (Finish Button)

### Step 3 – Post installation checks

To make it sure that MongoDB Compass has been correctly installed in your Windows machine, follow these steps:

1. Click the search area of your taskbar and type **Compass**. You will see that compass app will appear along with the details, as shown in the following screenshot:



**Figure 16.14: MongoDB Compass – Post Installation Checks**

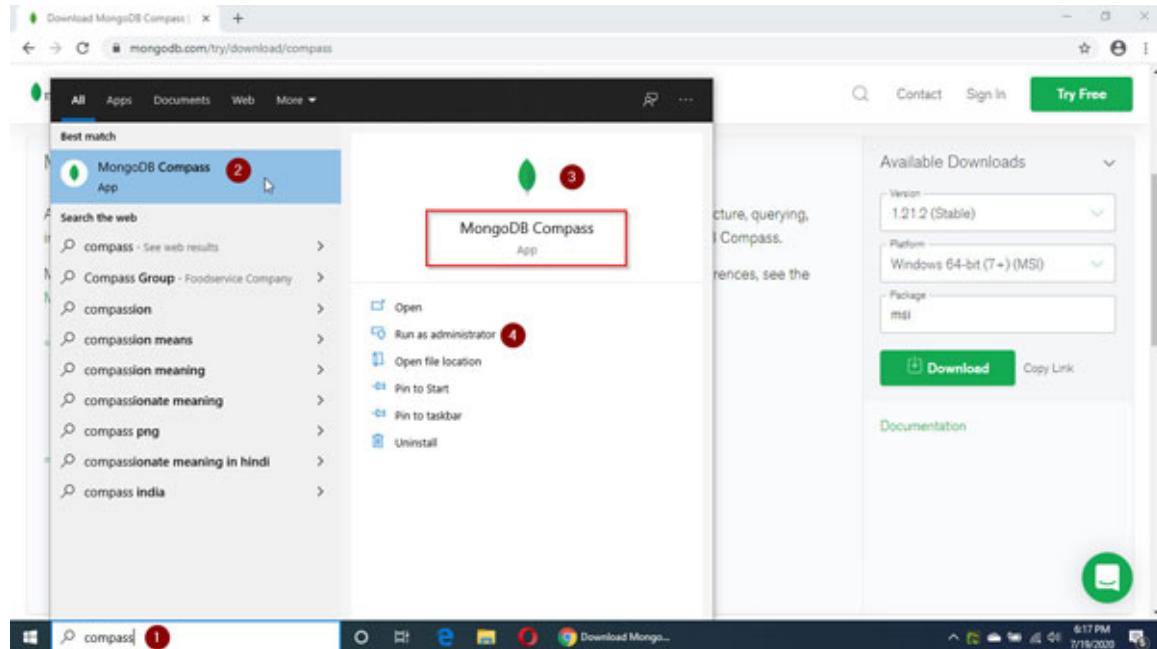
## Connecting MongoDB Server with MongoDB Compass

We have installed MongoDB Compass on our Machine. Now, we can start using the MongoDB Compass. In step 4, we will connect to the MongoDB server using the MongoDB Compass.

### **Step 4 – Connecting to the MongoDB Server on Windows using MongoDB Compass**

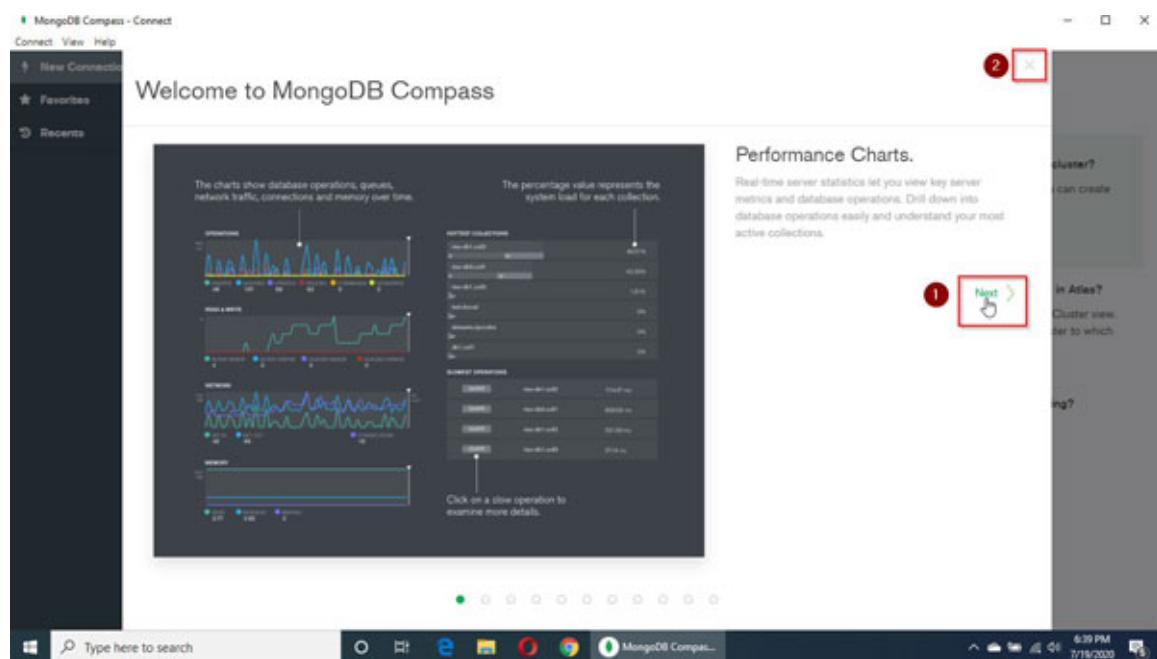
Let us now try to connect the MongoDB server using MongoDB Compass from your Windows machine. To connect to the MongoDB server from your Windows machine, follow these steps:

1. Click the search area of your taskbar and type **Compass**. You will see that compass app will appear along with the details. Click **Run as administrator**. This will open MongoDB Compass with the administrative privileges, as shown in the following screenshot:



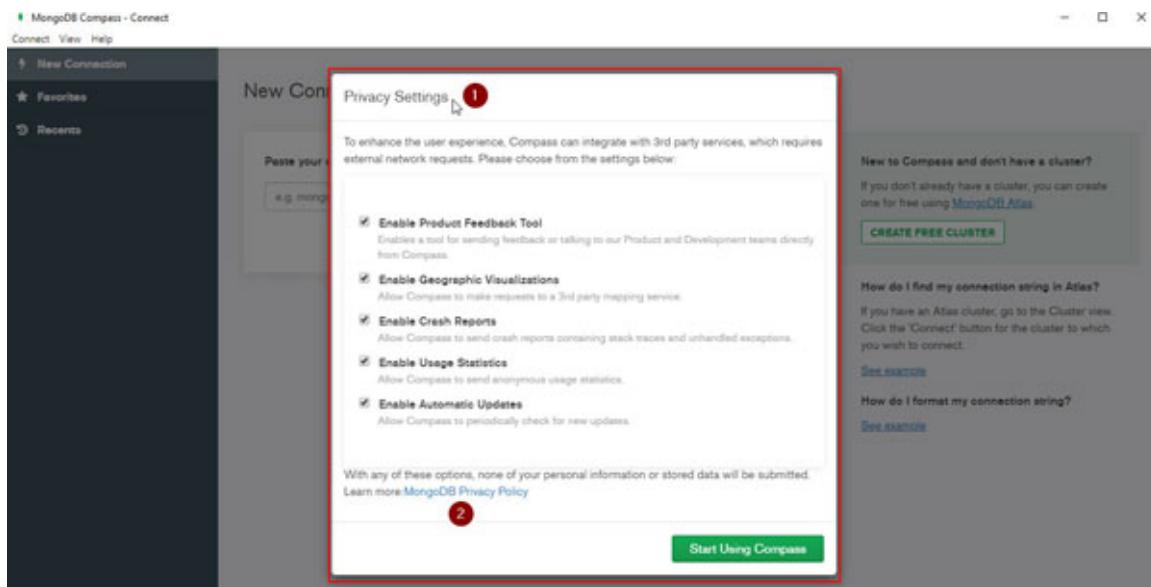
**Figure 16.15: MongoDB Compass – Open MongoDB Compass**

2. After you click on **Run as administrator**, it will open the MongoDB Compass application. Since we are using this application for the first time, it will show up the slider displaying the features of MongoDB Compass. At this point, you may read these features or may close the slider Window, as shown in the following screenshot:



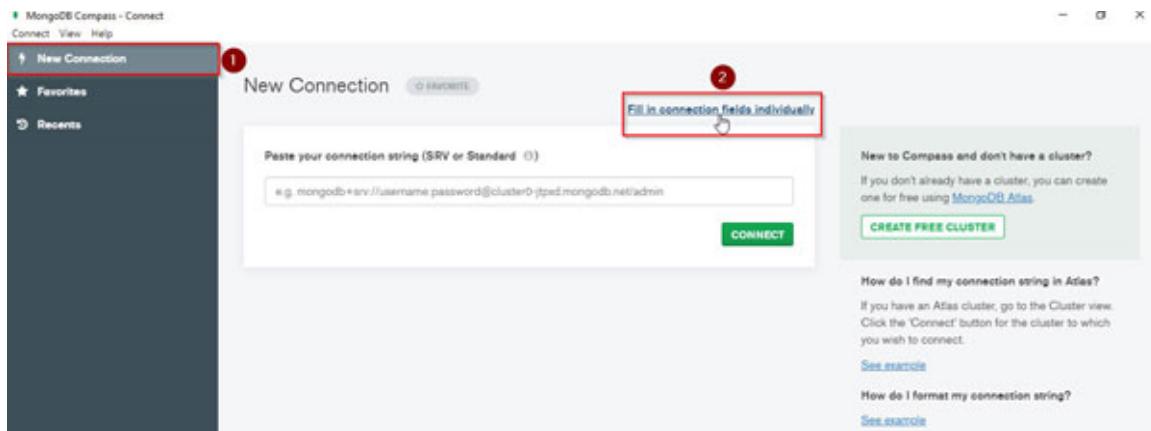
**Figure 16.16: MongoDB Compass – Application Main Screen**

3. Before you can start using MongoDB Compass, the application will ask you to decide on the **Privacy Settings**. These privacy settings are out of the scope of this book and it is up to the user to decide, based on his/her personal or organizational preferences, and it is advisable to read the MongoDB privacy policy by clicking **MongoDB Privacy Policy** link to get more information on this part. You may check or uncheck all of them, as shown in the following screenshot:



**Figure 16.17: MongoDB Compass – Privacy Settings.**

4. There are two ways to connect to the MongoDB Server with the MongoDB Compass. The first method is by using the connection string and the second one is by filling the connection fields individually. Let us explore the second method. Click on the link which says, **Fill in connection fields individually**, as shown in the following screenshot:



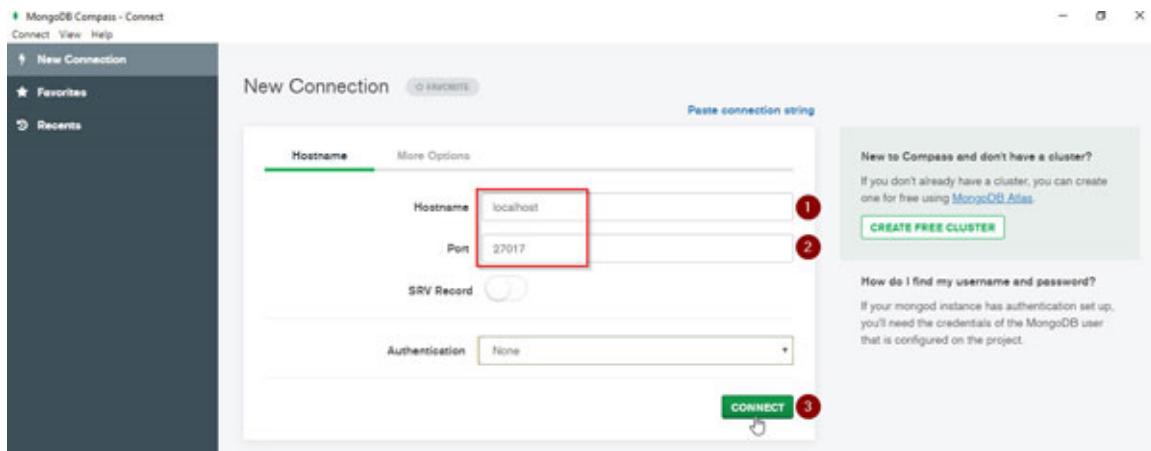
**Figure 16.18: MongoDB Compass – Connecting to MongoDB Server - Fill in connection fields individual**

5. In our example, we have entered the following values:

**Hostname:** localhost

**Port:** 27017

6. Now click on the **CONNECT** button to connect the MongoDB Server, which is already running on our local machine on port 27017 (this is the default port for MongoDB Server), as shown in the following screenshot:



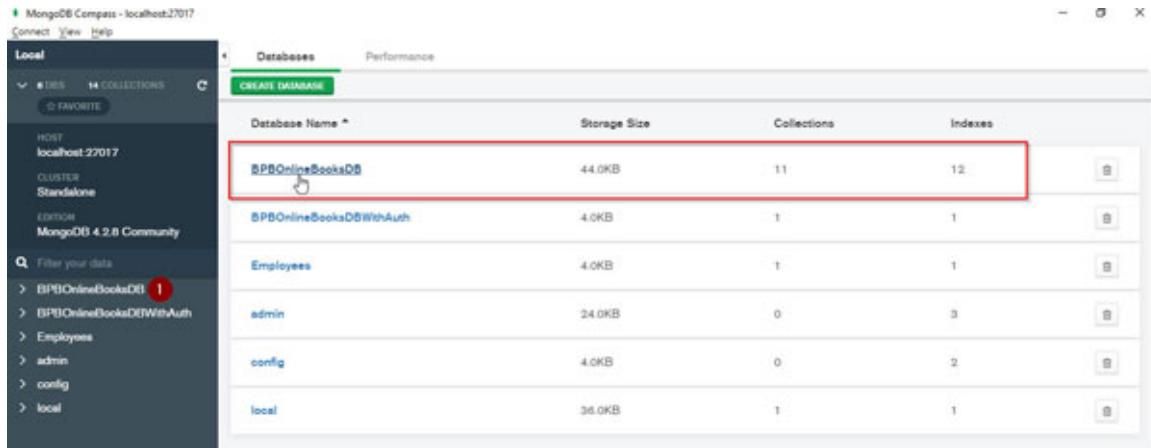
**Figure 16.19: MongoDB Compass – Connecting to MongoDB Server – Connecting to Local MongoDB Server**

7. After we click to connect, we can see the list of databases. Few of the Databases we might see are related to the MongoDB system and these are created by MongoDB by default as listed below:

**admin**

```
config  
local
```

8. We will work on the database `BPBOnlineBooksDB` which we have created in the previous chapters, as shown in the following screenshot:



The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'HOST' set to 'localhost:27017', 'CLUSTER' set to 'Standalone', and 'EDITION' set to 'MongoDB 4.2.8 Community'. Below that is a 'Filter your data' section with several dropdowns and a red-highlighted 'BPBOnlineBooksDB' dropdown. The main area is titled 'Databases' and shows a table of databases. The table has columns: Database Name, Storage Size, Collections, and Indexes. A red box highlights the first row for 'BPBOnlineBooksDB', which has a storage size of 44.0KB, 11 collections, and 12 indexes. Other databases listed are 'BPBOnlineBooksDBWithAuth', 'Employees', 'admin', 'config', and 'local'.

Database Name	Storage Size	Collections	Indexes
BPBOnlineBooksDB	44.0KB	11	12
BPBOnlineBooksDBWithAuth	4.0KB	1	1
Employees	4.0KB	1	1
admin	24.0KB	0	3
config	4.0KB	0	2
local	36.0KB	1	1

**Figure 16.20: MongoDB Compass – List of Databases**

So far, we have learnt how to connect to the MongoDB Server using MongoDB Compass. In the next section, we will understand MongoDB Compass with some more practical examples.

## Practical examples with MongoDB Compass

The following are some practical examples of MongoDB Compass to understand it in a better way:

### Example 1 –Browsing the collections in the database

To browse the collections in the database, follow the steps, as shown in the following screenshot:

1. Connect to the MongoDB server, as explained in the previous section of this chapter.
2. Click on the database of your choice.
3. You will get the list of collections present in that particular database.

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'HOST' set to 'localhost:27017', 'CLUSTER' set to 'Standalone', and 'EDITION' set to 'MongoDB 4.2.8 Community'. Below these are sections for 'BPBOnlineBooksDB' and its collections: AggregationPipelineCollection, BPBOnlineBooksCollection, BPBOnlineBooksCollectionAggregation, BPBOnlineBooksCollectionMapReduce, BPBOnlineBooksCollectionProjection, BPBOnlineBooksCollectionProjectionPipeline, BPBOnlineBooksCollectionQuerySelectors, BPBOnlineBooksCollectionWithIndex, BPBOnlineBooksCollection, and BPBOnlineBooksDBV4Collection. A red box highlights the 'CREATE COLLECTION' button at the top left of the main panel. The main panel displays a table of collections with columns: Collection Name, Documents, Avg. Document Size, Total Document Size, Num. Indexes, Total Index Size, and Properties.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
AggregationPipelineCollection	3	73.0 B	219.0 B	1	20.0 KB	
BPBOnlineBooksCollection	5	90.4 B	452.0 B	1	20.0 KB	
BPBOnlineBooksCollectionAggregation	8	368.0 B	2.9 KB	1	20.0 KB	
BPBOnlineBooksCollectionMapReduce	8	368.0 B	2.9 KB	1	20.0 KB	
BPBOnlineBooksCollectionProjection	7	376.4 B	2.6 KB	1	20.0 KB	
BPBOnlineBooksCollectionProjectionPipeline	8	368.0 B	2.9 KB	1	20.0 KB	
BPBOnlineBooksCollectionQuerySelectors	7	376.4 B	2.6 KB	1	20.0 KB	
BPBOnlineBooksCollectionWithIndex	7	376.4 B	2.6 KB	2	40.0 KB	
BPBOnlineBooksCollection	2	56.0 B	112.0 B	1	20.0 KB	
BPBOnlineBooksDBV4Collection	2	123.5 B	247.0 B	1	20.0 KB	

**Figure 16.21: Example 1 – Browsing the Collections in the Database**

In our example, we saw how we can browse the list of MongoDB Collection in the database.

## Example 2—Creating new collection in the database

To create a new collection in the database, follow these steps:

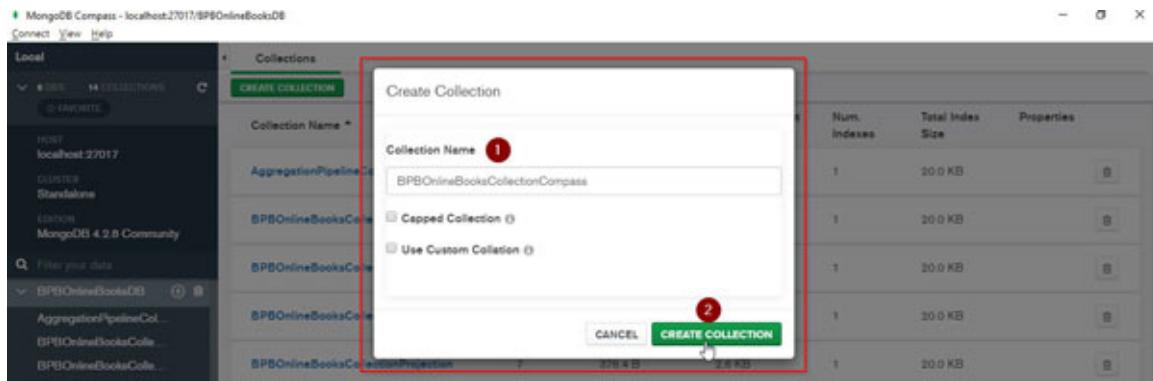
1. Connect to the MongoDB server, as explained in the previous section of this chapter.
2. Click on the database of your choice.
3. You will get the list of collections present in that particular database.
4. To create a new collection in this database, click the **CREATE COLLECTION** button on top left of the right part of the screen in which you are browsing your collections, as shown in the following screenshot:

The screenshot shows the MongoDB Compass interface, similar to Figure 16.21. The left sidebar shows 'HOST' as 'localhost:27017', 'CLUSTER' as 'Standalone', and 'EDITION' as 'MongoDB 4.2.8 Community'. The main panel displays a table of collections with columns: Collection Name, Documents, Avg. Document Size, Total Document Size, Num. Indexes, Total Index Size, and Properties. A red box highlights the 'CREATE COLLECTION' button at the top left of the main panel. The table data is identical to Figure 16.21.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
AggregationPipelineCollection	3	73.0 B	219.0 B	1	20.0 KB	
BPBOnlineBooksCollection	5	90.4 B	452.0 B	1	20.0 KB	

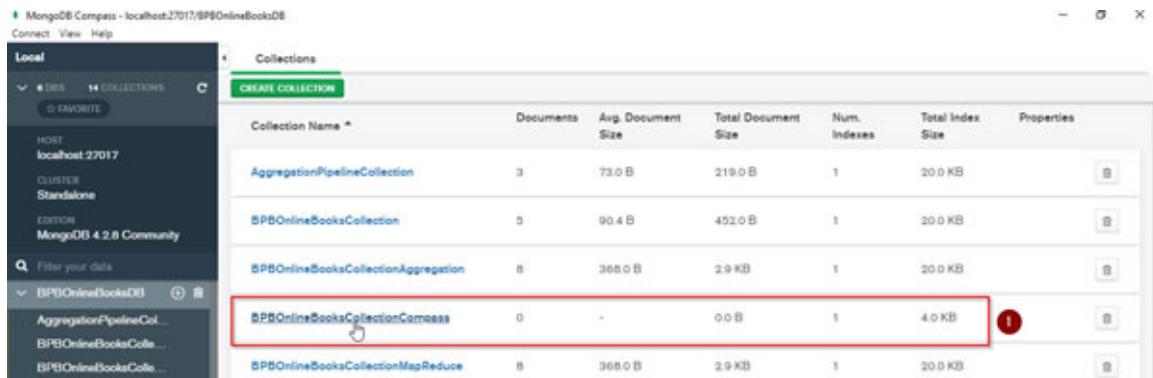
**Figure 16.22: Example 2 – Create New Collection – Screen 1**

5. This will open up the new popup window in which you can enter your Collection's name. After you have entered your collection's name, click on the **CREATE COLLECTION** button. This will create a new collection in the database, as shown in the following screenshot:



**Figure 16.23: Example 2 – Create New Collection – Screen 2**

6. You can browse the list of collections in the database to verify that this collection has been created by the MongoDB Compass, as shown in the following screenshot:



**Figure 16.24: Example 2 – Create New Collection – Screen 3**

So, in our example, we saw how to create a new MongoDB collection in the database by using MongoDB Compass.

## **Example 3–Browsing documents in the database**

To browse documents in the database, follow these steps, as shown in the following screenshot:

1. Connect to the MongoDB server, as explained in the previous section of this chapter.
2. Click on the database of your choice.
3. You will get the list of collections that are present in that particular database.
4. Click on the collection of your choice.
5. Once you select the collection of your choice, you will get the list of documents present in that particular collection.
6. Click the expand icon on the left side on the document to see all the data inside the document.

The screenshot shows the MongoDB Compass interface. The left sidebar lists databases and collections. The main area shows the 'BPBOnlineBooksDB.BPBOnlineBooksCollectionProjectionPipeline' collection. A red box highlights the pipeline name at the top. Another red box highlights a specific document in the list, showing its detailed structure:

```

_id: ObjectId("5f8eecd6d68812007012412acff")
Title: "IoT and Smart Cities: Your Smart City Planning Guide"
Year: "2018"
ISBN: "9789386511322"
Pages: 242
Weight: "3.5kg"
Dimension: "22.5x15x1.5cm"
Tags: Array
  0: "Internet of Things"
  1: "IoT"
  2: "Smart City"
  3: "Planning Guide"
  4: "Non Programming"
  5: "Technology"
  6: "Project"
  7: "Object"
  8: "Book"
  9: "Paperback"
  10: Object
    Type: "Hardcover"
    Quantity: 2000
  11: Object
    Type: "Hardcover"
    Quantity: 3000
    SpecialOfferDiscount: "100"
  
```

**Figure 16.25: Example 2 – Browsing the Documents in the Database**

In our example, we saw how we can browse the list of MongoDB documents in the collection.

## Example 4—Performing CRUD operations in documents

There is lot of features available in MongoDB Compass to manipulate the data, as shown in the following screenshot:

1. Connect to the MongoDB server, as explained in the previous section of this chapter.

2. Click on the database of your choice.
3. You will get the list of collections that are present in that particular database.
4. Click on the collection of your choice.
5. Once you select the collection of your choice you, will get the list of documents that are present in that particular collection.
6. On the right side of the document you will see lot of CRUD options, like editing, copying, cloning, and deleting. You can also add a new document by clicking the **ADD DATA** button.

The screenshot shows the MongoDB Compass interface connected to the 'localhost:27017' cluster. The left sidebar lists databases and collections, with 'BPBOnlineBooksDB' selected. The main area displays the 'BPBOnlineBooksDB.BPBOnlineBooksCollectionProjectionPipeline' collection. The 'Documents' tab is active, showing 8 documents. A red box highlights the green 'ADD DATA' button. Below it, a red box highlights the 'Edit Document' icon (pencil) in the toolbar of a document preview. The documents listed include details like title, year, ISBN, pages, tags, and dimensions.

**Figure 16.26: Example 4 – Performing CRUD Operations in Documents**

In our example, we saw how we can perform the CRUD operations in the documents.

## Example 5–Editing a document

To edit a document using MongoDB Compass, follow these steps, as shown in the following screenshot:

1. Connect to the MongoDB server, as explained in the previous section of this chapter.
2. Click on the database of your choice.

3. You will get the list of collections that are present in that particular database.
4. Click on the collection of your choice.
5. Once you select the collection of your choice you, will get the list of documents that are present in that particular collection.
6. On the right side of the document you will see lot of CRUD options, like editing, copying, cloning, and deleting. Click on Edit and it will open a new window where you can very easily manipulate or edit your data.
7. Click on the **UPDATE** button to make the changes or on the **CANCEL** button to ignore.

The screenshot shows the MongoDB Compass interface. The left sidebar lists databases and collections, with 'BPBOnlineBooksDB' selected. The main pane displays the 'BPBOnlineBooksDB.BPBOnlineBooksCollectionProjectionPipeline' collection. A document is being edited, showing fields like '\_id', 'Title', 'Year', 'ISBN', 'Pages', 'Tags' (which contains 'Digital Marketing'), and 'Instock'. A red box highlights the 'Tags' field, with a callout 'New Tag Added in Tags Field'. At the bottom right of the document editor, a red box highlights the 'UPDATE' button.

**Figure 16.27: Example 5 – Editing a Document**

In our example, we saw how we can edit or manipulate the documents using MongoDB Compass. We will use MongoDB Compass in our next chapters also where we will learn how to develop applications with MongoDB.

## Conclusion

In this chapter, we covered the MongoDB Compass and learned about the benefits of using it. We also learned how to install MongoDB Compass on our machine using step-by-step method. We also learned

how we can use MongoDB Compass to connect to the MongoDB server. In the last part of this chapter, we learned more about the MongoDB Compass with some step-by-step practical examples to give us more insights about MongoDB Compass.

In the next chapter of this book, we will cover the managing and administering MongoDB at advanced level which is very helpful to the people who will perform various MongoDB administrative tasks. We will learn about `mongod` process and how to manage `mongod` process. We will learn how to monitor and diagnose MongoDB. We will also cover the step-by-step methods to install MongoDB tools on our machine and how to use these tools and various other MongoDB commands to monitor and diagnose MongoDB as well as backup and restore with MongoDB and MongoDB security.

## **Questions**

1. What is MongoDB Compass?
2. List any four features of MongoDB Compass.
3. Can we use MongoDB Compass on Linux or MacOS machines?
4. List the steps by which you can connect MongoDB Compass to the MongoDB server.
5. How can you create a new collection in the MongoDB database using MongoDB Compass?
6. Can we perform CRUD operations with MongoDB Compass?

## **Points to remember**

- MongoDB Compass is the GUI (Graphical User Interface) tool which helps us to connect to the MongoDB server very easily and do a lot of things using GUI which takes lot of time if we do them via commands or queries.
- MongoDB Compass is available for all the major operating systems, such as Windows, Linux, or MacOS.
- Before using MongoDB Compass, make sure that the MongoDB server is running on your machine.

## Multiple choice questions

**1. What is MongoDB Compass?**

- a. Command Line Interface
- b. Programming Language
- c. GUI (Graphical User Interface) Tool
- d. None of these

**2. MongoDB Compass is available for:**

- a. Windows
- b. Linux
- c. Mac OS
- d. All of the Above

**3. With MongoDB Compass you can edit and delete:**

- a. Documents
- b. Collections
- c. All of the Above
- d. None of the Above

**4. Before you connect the MongoDB server using MongoDB Compass, is it required that MongoDB server should be running?**

- a. Yes
- b. No
- c. Not required when you run MongoDB Compass as an administrator
- d. Sometimes Required

## Answer

- 1. c
- 2. d
- 3. c

4. a

## Key terms

- **GUI:** Graphical User Interface.
- **CRUD:** In computer programming, the acronym CRUD stands for create, read, update, and delete.
- **MSI:** This is the extension used by the installer file for the Windows operating systems and it is called Microsoft installer or later as Windows installer.

## CHAPTER 17

# Managing and Administering MongoDB (Advanced Level)

This chapter covers the advanced administration topics of MongoDB which is very helpful to the people who will perform various MongoDB administrative tasks. In this chapter, we will learn about mongod process and how to manage mongod process. We will learn how to monitor and diagnose MongoDB. We will cover the step-by-step method to install MongoDB tools on our machine and how to use these MongoDB tools and various other MongoDB commands to monitor and diagnose MongoDB. Later in this chapter, we will learn how to take MongoDB backups and how to restore these backups. In the later sections of this chapter, we will cover how to perform the export and import of the MongoDB data. In the last part of this chapter, we will cover various important points related to MongoDB security which we should take care of so that our MongoDB database and its data will get secured.

### Structure

In this chapter, we will discuss the following topics:

- Managing mongod process
- Monitoring and diagnosing MongoDB
  - Installing MongoDB tools
  - Using MongoDB tools
- Backup and restore with MongoDB
- Export and import with MongoDB
- MongoDB security

### Objectives

After studying this unit, you should be able to understand the `mongod` process and how to manage the `mongod` process. You should also be able to monitor and diagnose MongoDB. You will learn how to install MongoDB tools and also how to use these tools and commands to monitor and diagnose MongoDB. Later in this chapter, you will learn how to take backups and how to restore backups with MongoDB and perform export and import with MongoDB. In the last section of this chapter, you will understand about the MongoDB security and various important points to implement the MongoDB security.

## About mongod process

The `mongod` process is the "*primary process*" as well as the "*daemon process*" for MongoDB. The `mongod` process runs in the background and that is why it is called daemon process. This MongoDB process is responsible for the following things:

- Handling data requests
- Managing data access
- Performing background management process

`mongod` is a program which can be run from the command line. In Windows, `mongod` also runs as the service and by default, when we install MongoDB in Windows, the MongoDB installer automatically creates the service named "**MongoDB Server (MongoDB)**", which uses `mongod` for this service.

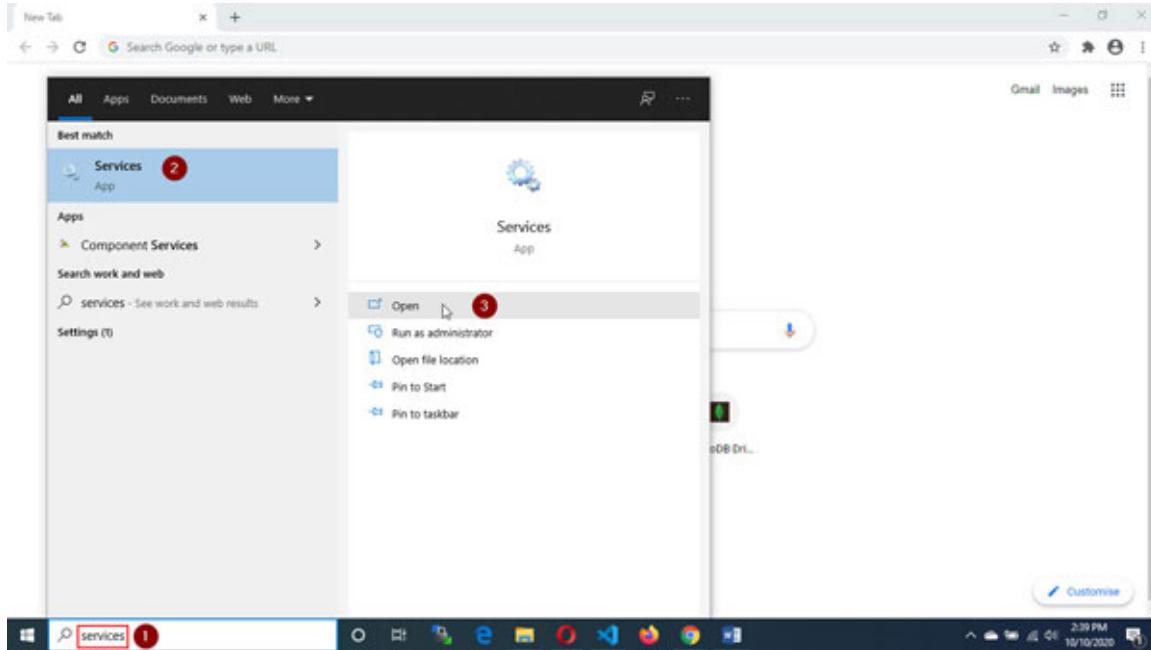
## Managing mongod process

We know when we install MongoDB on Windows; it automatically creates MongoDB service, which is responsible to start the `mongod` process in the background. Let us now understand how we can check the MongoDB service in Windows.

## MongoDB service in Windows

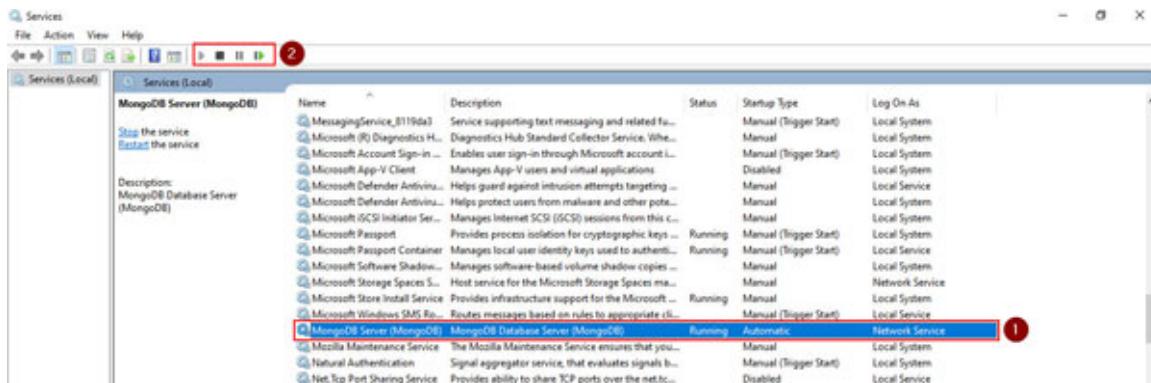
If you want to see the MongoDB service and related settings in Windows, follow these steps:

1. Type **services** in the search box of your Windows. You will see that the Windows Services Manager application icon will appear. Click on **open** to open this, as shown in the following screenshot:



**Figure 17.1: Windows Search Box – Opening Windows Services Manager**

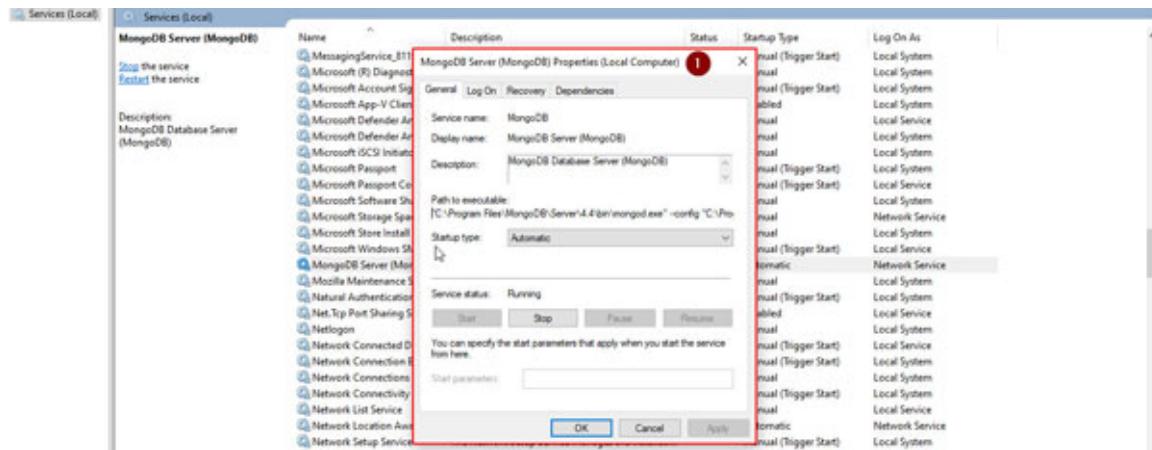
2. This will open the Windows Services Manager. You can now navigate to the MongoDB server (MongoDB) and can see the status of these services. There is an option available to start, stop, or restart these services, as shown in the following screenshot:



**Figure 17.2: Windows Services Manager – MongoDB Server (MongoDB) Service**

3. Now, to view some more information, right click the MongoDB server (MongoDB) from the Windows services list and then click

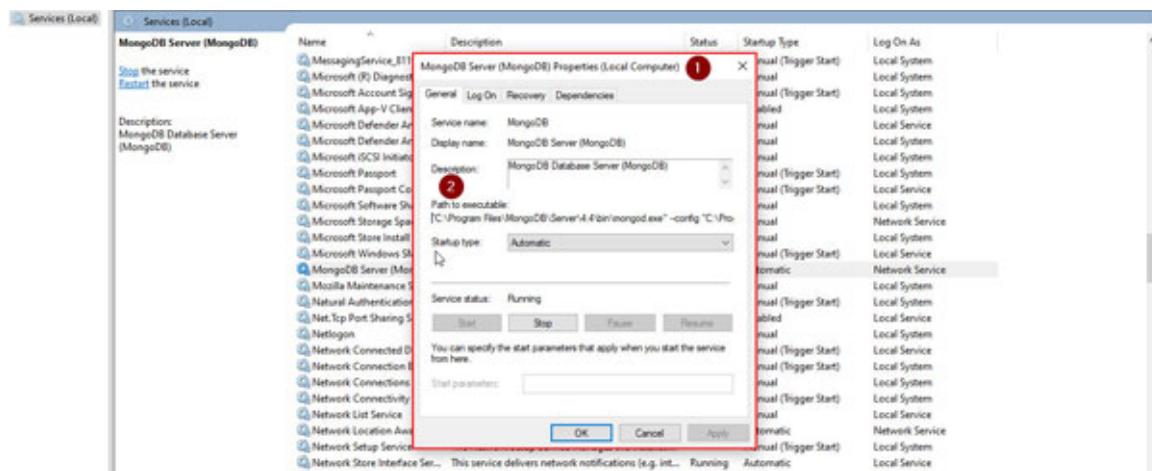
on the **Properties** option from the menu, as shown in the following screenshot:



**Figure 17.3: MongoDB Server (MongoDB) Service - Properties**

- The previous step will open the service properties window where you will get all the details about the MongoDB service. If you see properly, you will also get the information about the `mongod` under **Path to executable** and you will see something similar as the following path, as shown in the following screenshot:

**Path to executable:** "C:\Program  
Files\MongoDB\Server\4.4\bin\mongod.exe" --config "C:\Program  
Files\MongoDB\Server\4.4\bin\mongod.cfg" -service

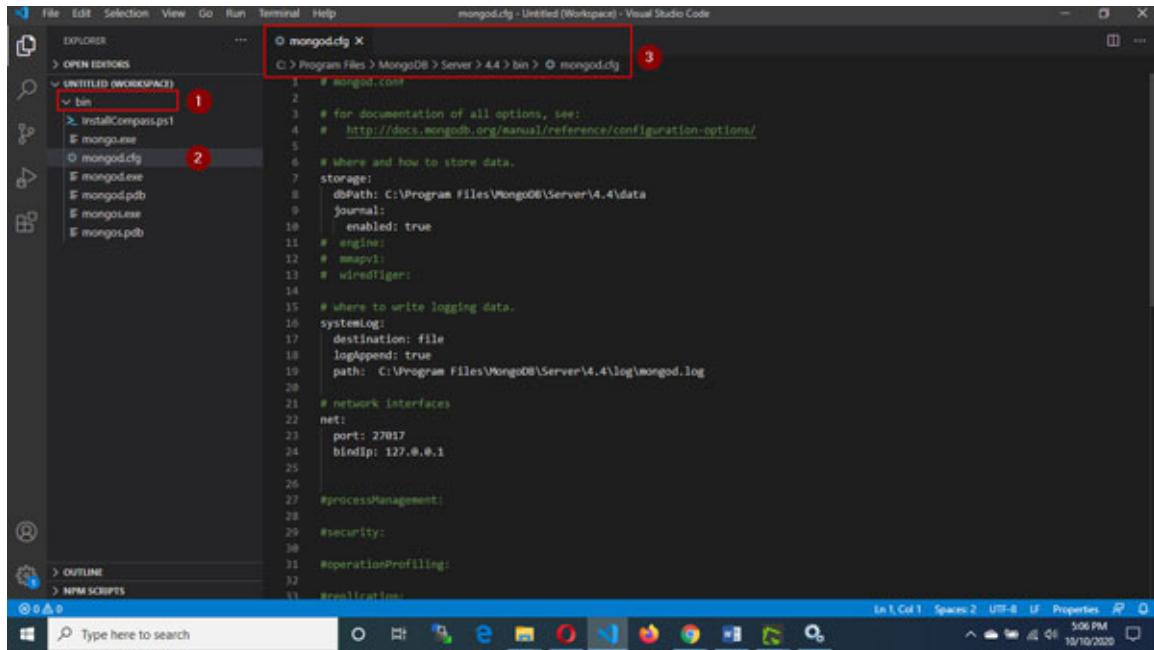


**Figure 17.4: MongoDB Server (MongoDB) Service – Properties Details**

Here, under the "Path to executable", you will see the location of "mongod" and its related configuration file, "mongod.cfg". This file contains the various default configuration options that come pre-

configured when you install the MongoDB server on your Windows machine.

5. You may open this file in any text editor or code editor, like *Visual Studio Code*, to view its contents, as shown in the following screenshot:



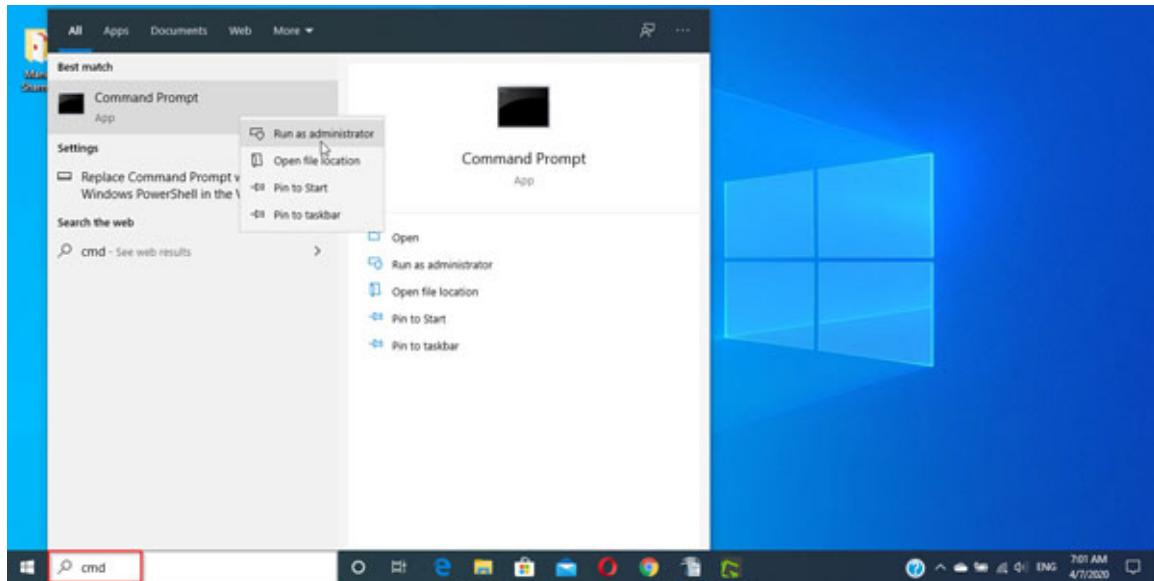
*Figure 17.5: "mongod.cfg" – Configuration File*

## Running mongod from command prompt

We can also run "mongod" from the command prompt and we can use multiple options while running `mongod` from the command line or shell.

The simple way to run MongoDB is to navigate to the `bin` directory of your MongoDB server installation path and type "mongod". To start `mongod` from the command prompt, follow these steps:

1. Click the Windows start button and type "`cmd`" in the search box. You will see the Windows Command Shell program. Open this program as an administrator, as shown in the following screenshot:



**Figure 17.6:** Opening Command Prompt as an Administrator.

2. This will open a command prompt. Here you need to navigate to your MongoDB installation path, as shown in the following screenshot, which could be something like this: C:\Program Files\MongoDB\.



**Figure 17.7:** From Command Line – Navigate to MongoDB Installation Directory.

3. Now, we need to further navigate to the child directories so that we reach to the "bin" directory under \Server\4.4\bin. The complete path would be something like this: C:\Program Files\MongoDB\Server\4.4\bin, as shown in the following screenshot:



**Figure 17.8:** From Command Line – Navigate to MongoDB "bin" Directory.

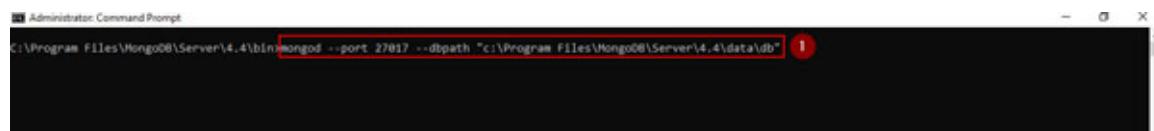
4. Now, we need to run MongoDB using the following command, as shown in the following screenshot:

```
mongod --port 27017 --dbpath "c:\Program  
Files\MongoDB\Server\4.4\data\db"
```

Note that if you haven't created the "db" directory under this path: "c:\Program Files\MongoDB\Server\4.4\data\", you should first create the "db" directory.

This command will try to run MongoDB on the port: 27017 with a DB path:

```
"c:\Program Files\MongoDB\Server\4.4\data\db"
```



**Figure 17.9:** From Command Line – Navigate to MongoDB "bin" Directory.

5. After this command successfully gets completed, MongoDB will start, as shown in the following screenshot:

```
mongod --port 27017 --dbpath "c:\Program  
Files\MongoDB\Server\4.4\data\db"
```

This command will try to run MongoDB on port: 27017 with a DB path: "c:\Program Files\MongoDB\Server\4.4\data\db"

Do not close this window. You can minimize it so that MongoDB will work and the listen to the requests. In case you want to stop it, use **Ctrl + C** keys to stop it on your Windows machine.

```

Administrator: Command Prompt - mongod --port 27017 --dbpath "c:\Program Files\MongoDB\Server\4.4\data\db"
getInfoOS {"os": "Windows 7/Windows Server 2008 R2"}
{"t":{"$date":2020-10-10T17:41:15.878+05:30"}, "s":1, "c":"CONTROL", "id":23403, "ctx":"initandlisten", "msg":"Build info", "attr":{"buildInfo":{"version": "4.4.1", "gitVersion": "ad91e93a5a175f5c6bf8c69561e780bc5ce1", "modules": {}, "allocator": "tcmalloc", "environment": {"distmod": "windows", "distarch": "x86_64", "target_arch": "x86_64"}}}}
{"t":{"$date":2020-10-10T17:41:15.878+05:30}, "s":1, "c":"CONTROL", "id":51765, "ctx":"initandlisten", "msg":"Operating System", "attr":{"os":{"name": "Microsoft Windows 20", "version": "10.0 (build 19041)"}}, "attr":{}}
{"t":{"$date":2020-10-10T17:41:15.878+05:30}, "s":1, "c":"CONTROL", "id":22430, "ctx":"initandlisten", "msg":"Options set by command line", "attr":{"options": {"net": {"port": 27017}, "storage": {"dbPath": "c:\\Program Files\\MongoDB\\Server\\4.4\\data\\db"}}, "attr":{}}}
{"t":{"$date":2020-10-10T17:41:15.881+05:30}, "s":1, "c":"STORAGE", "id":22270, "ctx":"initandlisten", "msg":"Storage engine to use detected by data files", "attr": {"dbpath": "c:\\Program Files\\MongoDB\\Server\\4.4\\data\\db", "storageEngine": "wiredTiger"}, "attr":{}}
{"t":{"$date":2020-10-10T17:41:15.881+05:30}, "s":1, "c":"STORAGE", "id":22315, "ctx":"initandlisten", "msg":"Opening WiredTiger", "attr": {"config": {"create,cache_size": 35360, "session_max": 33000, "eviction": {"threads_min": 4, "threads_max": 4}, "config_base": false, "statistics": {fast}, "log": {"enabled": true, "archive": true, "path": "journal", "compressor": "snappy"}, "file_manager": {"close_idle_time": 1000000, "close_scan_interval": 10, "close_handle_minimum": 250}, "statistics_log": {"wait": 0}, "verbose": {"recovery_progress", "checkpoint_progress", "compact_progress": 0}}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:15.878+05:30}, "s":1, "c":"CONTROL", "id":51765, "ctx":"initandlisten", "msg":"Opening WiredTiger message", "attr": {"message": "[1602331876:0117][2736:146717795857648] txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 3 through 4"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:16.189+05:30}, "s":1, "c":"CONTROL", "id":22430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr": {"message": "[1602331876:00089][12736:140717795857648] txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 4 through 4"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:16.341+05:30}, "s":1, "c":"STORAGE", "id":22430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr": {"message": "[1602331876:01086][12736:140717795857648] txn-recover: [WT_VERB_RECOVERY | WT_VERB_RECOVERY_PROGRESS] Main recovery loop: starting at 3/4096 to 4/256"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:16.587+05:30}, "s":1, "c":"STORAGE", "id":22430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr": {"message": "[1602331876:06997][12736:140717795857648] txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 3 through 4"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:16.812+05:30}, "s":1, "c":"STORAGE", "id":22430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr": {"message": "[1602331876:01462][12736:140717795857648] txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 4 through 4"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:16.956+05:30}, "s":1, "c":"STORAGE", "id":22430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr": {"message": "[1602331876:06131][12736:140717795857648] txn-recover: [WT_VERB_RECOVERY | WT_VERB_RECOVERY_PROGRESS] Set global recovery timestamp: (0, 0)"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:16.956+05:30}, "s":1, "c":"STORAGE", "id":22430, "ctx":"initandlisten", "msg": "WiredTiger message", "attr": {"message": "[1602331876:06131][12736:140717795857648] txn-recover: [WT_VERB_RECOVERY | WT_VERB_RECOVERY_PROGRESS] Set global oldest timestamp: (0, 0)"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:17.208+05:30}, "s":1, "c":"RECOVERY", "id":23987, "ctx":"initandlisten", "msg": "WiredTiger recoveryTimestamp", "attr": {"recoveryTimestamp": "(1, 1)"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:17.238+05:30}, "s":1, "c":"STORAGE", "id":22262, "ctx":"initandlisten", "msg": "Timestamp monitor starting"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:17.306+05:30}, "s":1, "c":"CONTROL", "id":22120, "ctx":"initandlisten", "msg": "Access control is not enabled for the database. Read and write access to data and configuration is unrestricted.", "tags": ["startupWarnings"]}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:17.330+05:30}, "s":1, "c":"CONTROL", "id":22140, "ctx":"initandlisten", "msg": "This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 0.0.0.1 to disable this warning.", "tags": ["startupWarnings"]}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:17.330+05:30}, "s":1, "c":"STORAGE", "id":20536, "ctx":"initandlisten", "msg": "Flow Control is enabled on this deployment"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:18.042+05:30}, "s":1, "c": "FDDC", "id":20625, "ctx": "initandlisten", "msg": "Initializing full-time diagnostic data capture", "attr": {"dataDirectory": "c:\\Program Files\\MongoDB\\Server\\4.4\\data\\db\\diagnostic_data"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:18.058+05:30}, "s":1, "c": "NETWORK", "id":23015, "ctx": "listener", "msg": "Listening on", "attr": {"address": "127.0.0.1"}, "attr": {}}
{"t":{"$date":2020-10-10T17:41:18.058+05:30}, "s":1, "c": "NETWORK", "id":23016, "ctx": "listener", "msg": "Waiting for connections", "attr": {"port": 27017, "ssl": "off"}, "attr": {}}

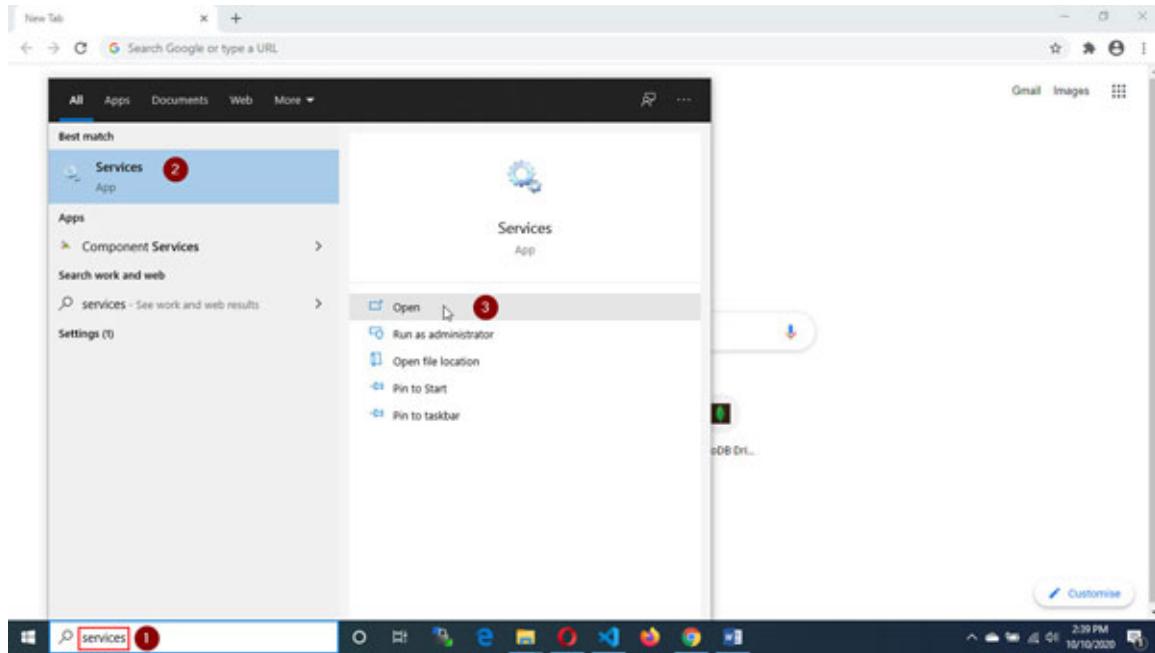
```

**Figure 17.10: From Command Line – MongoDB Service gets started**

## Stopping MongoDB services from Windows service manager

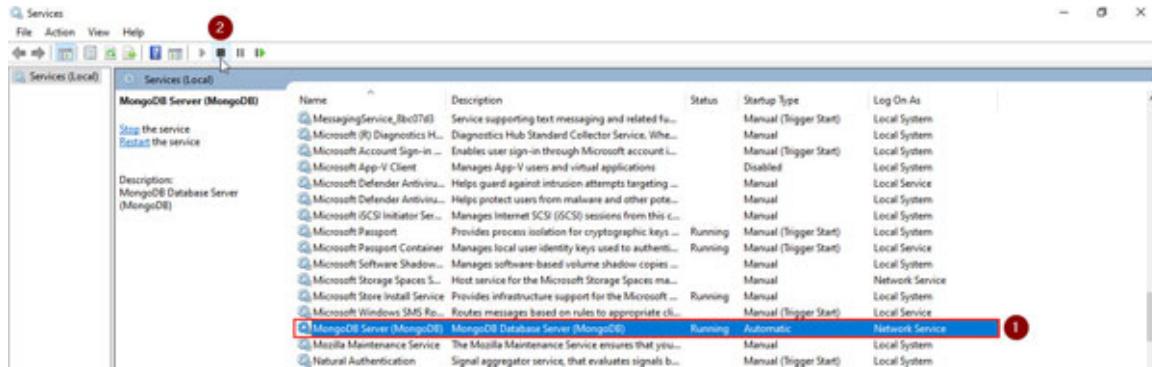
If you want to see the MongoDB service and related settings in Windows, follow these steps:

1. Type `services` in the search box of your Windows. You will see that `services` Windows Services Manager application icon will appear. Click on `open` to open this, as shown in the following screenshot:



**Figure 17.11: MongoDB Inc. Official Website Home Page**

2. This will open the Windows Services Manager. You can then navigate to the MongoDB server (MongoDB) and see the status of these services. There is an option available to start, stop, or restart these services. Click on the **stop** button to stop the MongoDB services, as shown in the following screenshot:

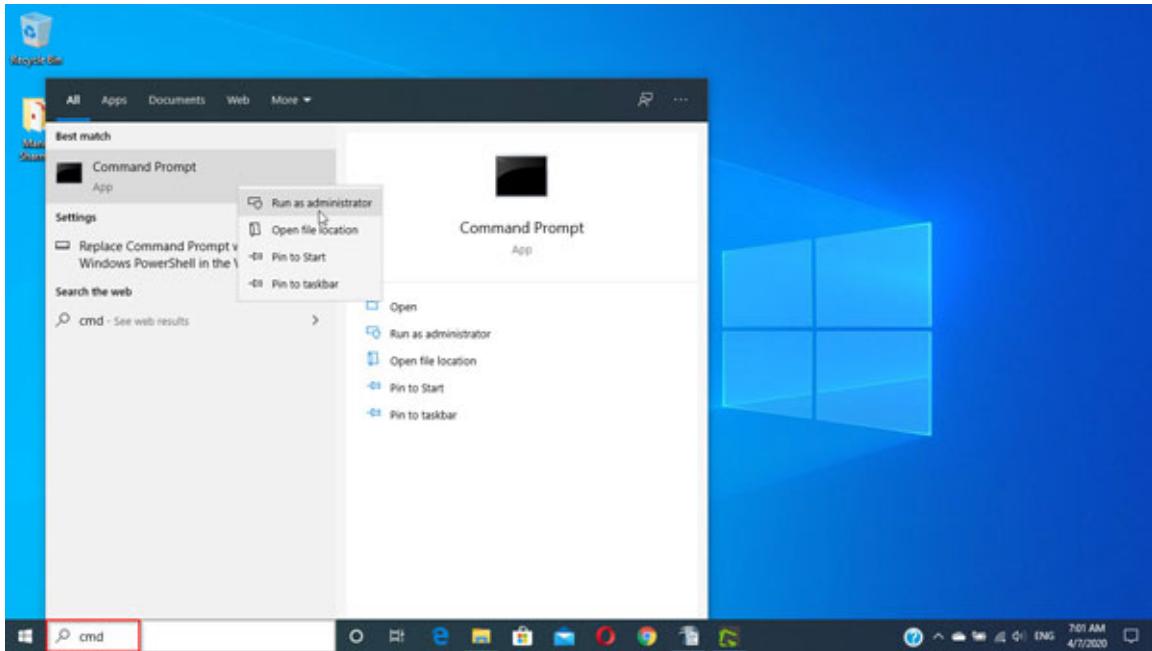


**Figure 17.12: Stopping the MongoDB Service from Windows Service Manager**

## Stopping MongoDB Services from command line – MongoDB Shell method

1. Click the Windows start button and type `cmd` in the search box. You will see the windows command shell program. Open this program

as an administrator, as shown in the following screenshot:



**Figure 17.13:** Opening Command Prompt as an Administrator.

2. This will open a command prompt. Here, you need to navigate to your MongoDB bin directory path, as shown in the following screenshot, which could be something like this: C:\Program Files\MongoDB\Server\4.4\bin.



**Figure 17.14:** From Command Line – Navigate to MongoDB Installation Directory.

3. Now, login to MongoDB Shell by typing "mongo". It will start the MongoDB Shell prompt, as shown in the following screenshot:

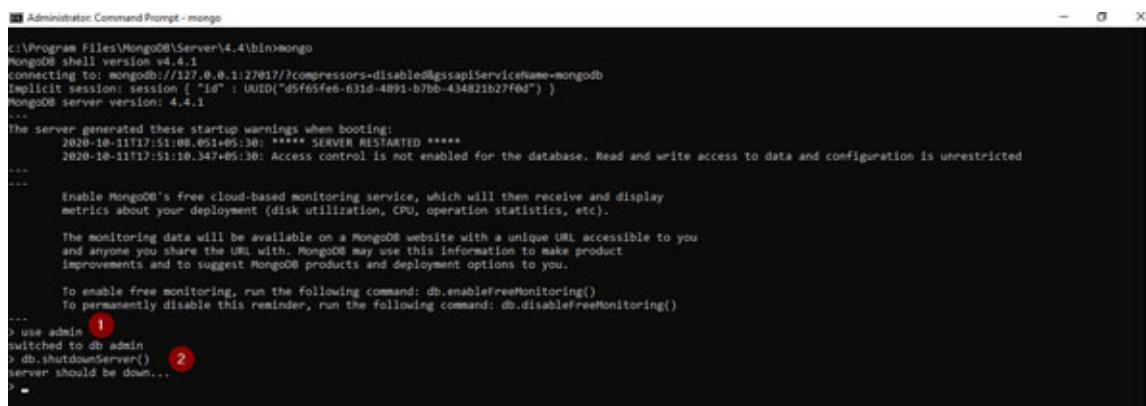
```
c:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gsapiServiceName=mongodb
Implicit session: session { "_id" : UUID("d5f65fe6-631d-4891-b7bb-434821b27f6d") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-10-11T17:51:08.051+05:30: ***** SERVER RESTARTED *****
2020-10-11T17:51:10.347+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
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()
```

A screenshot of a command prompt window titled 'Administrator: Command Prompt - mongo'. The title bar has a red border around the text. The command 'mongo' is typed into the prompt. The output shows the MongoDB shell version (v4.4.1), the connection to the local host at port 27017, the implicit session ID, and the MongoDB server version (4.4.1). It also displays startup warnings about access control and a reminder to enable free monitoring.

**Figure 17.15:** From Command Line – Navigate to MongoDB "bin" directory.

- Now, first select the database admin by issuing the command, `use admin`, and then, issue the following method to stop the MongoDB server and to gracefully shutdown it, as shown in the following screenshot:

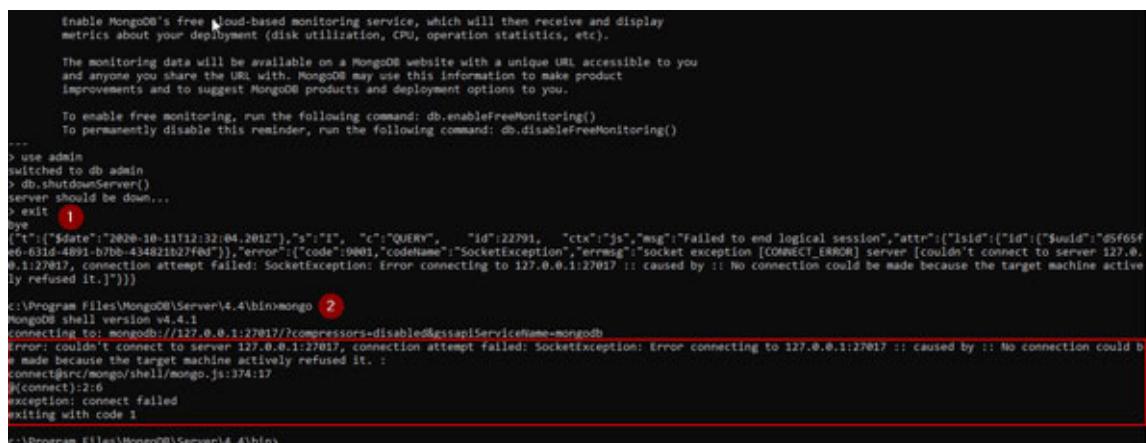
```
db.shutdownServer()
```



```
c:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d5f05fe0-631d-4891-b7bb-434821b27fed") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings:
2020-10-11T17:51:00.051405Z: **** SERVER RESTARTED ****
2020-10-11T17:51:10.347405Z: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
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()
> use admin
switched to db admin
1
> db.shutdownServer()
2
server should be down...
>
```

**Figure 17.16:** MongoDB Shell - `db.shutdownServer()` Method to shut down the MongoDB Server

- Once you do this, MongoDB will try to stop MongoDB gracefully by properly closing all the data files and flushing the data files and then stopping MongoDB. You can exit from the MongoDB Shell and then try to issue the `mongo` command again. You will see that the MongoDB Shell could not start as the MongoDB server is not running now, as shown in the following screenshot:



```
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()
> use admin
switched to db admin
> db.shutdownServer()
server should be down...
> exit
1
2
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
error: couldn't connect to server 127.0.0.1:27017, connection attempt failed: SocketException: Error connecting to 127.0.0.1:27017 :: caused by :: No connection could be made because the target machine actively refused it.
connect@src/mongo/shell/mongo.js:374:17
@connect():2:6
exception: connect failed
existing with code 1
c:\Program Files\MongoDB\Server\4.4\bin>
```

**Figure 17.17:** `mongo` Command could connect to MongoDB Server

## Monitoring and diagnosing MongoDB

Monitoring and diagnosing MongoDB are one of the advanced level administration tasks for MongoDB. In this, we first try to monitor our MongoDB databases for their proper working by analyzing the logs and other metrics provided by MongoDB when there are query executions and during the input and output of the data. Monitoring gives us the picture of what is happening in our MongoDB instance and helps us to prevent any failure of the MongoDB instance or any other issue related to various factors, like slow query responses, traffics, etc.

We can monitor our MongoDB server at various levels, such as at the MongoDB instance level, the database level, or the collection level, etc. Sometimes we might experience some issue with our MongoDB server. Then, we need to diagnose and quickly understand the issue and take the required action to resolve it, so that it does not get escalated or result in the down time or system failure.

So, analyzing and diagnosing the issue with the help of proper tools and commands help MongoDB administrators to tackle the issues as they come. With the help of proper monitoring and preventive diagnosing, we can have our MongoDB server and related services up and running without any major issues.

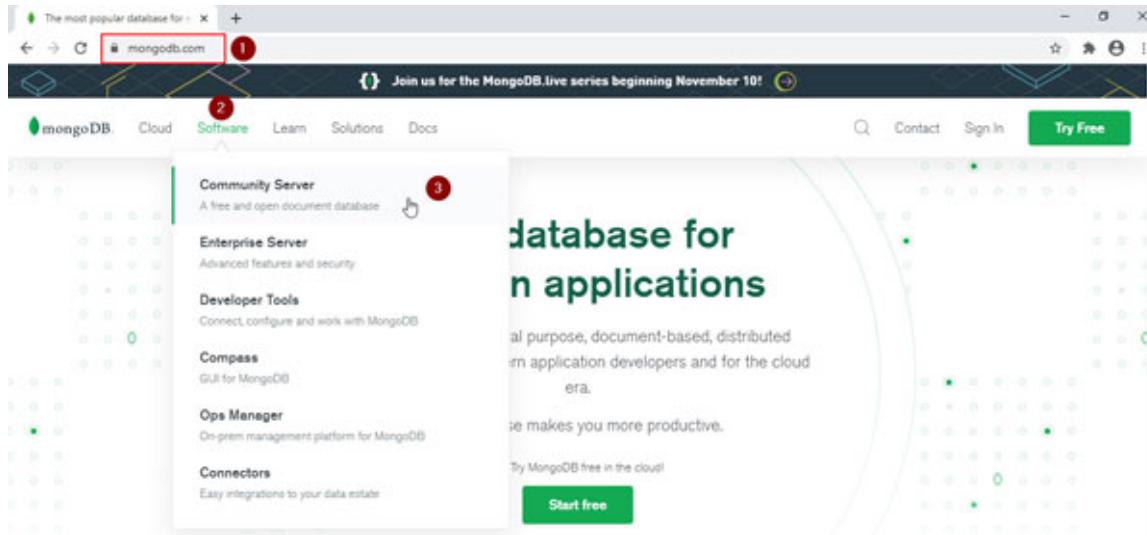
In order to monitor and diagnose MongoDB, we need to have some tools that will show us the related data as well as help in diagnosing and troubleshooting the issues. For this, we need to install the *MongoDB Reporting Tools* from the MongoDB Inc. official website.

Let us first install the *MongoDB Reporting Tools and Utilities* and then we can further look on how to use these tools.

## Installing MongoDB tools and utilities

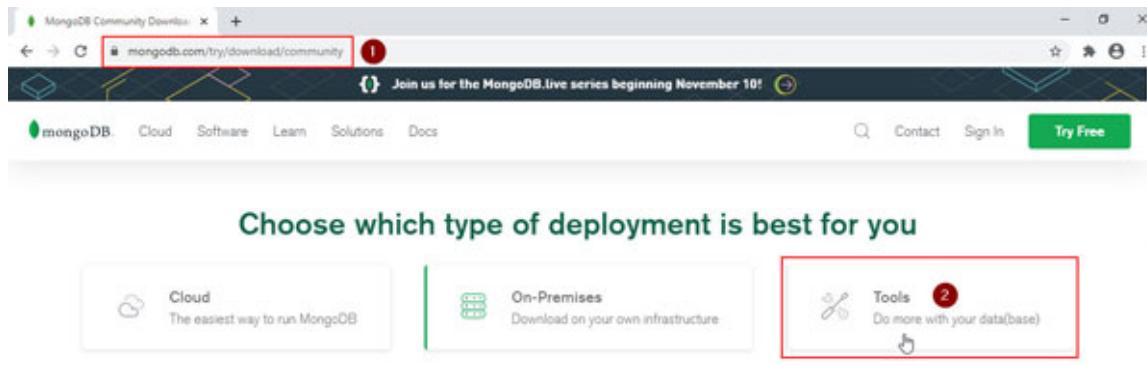
In order to install the *MongoDB Tools*, follow these steps:

1. Visit the MongoDB Inc. official website home page <https://www.mongodb.com/>. On the top navigation section, move your mouse pointer to the `software` link. This will open the `software` menu. Under this menu, click the `community Server`, which will open the new page related to the Community Edition of the MongoDB Server, as shown in the following screenshot:



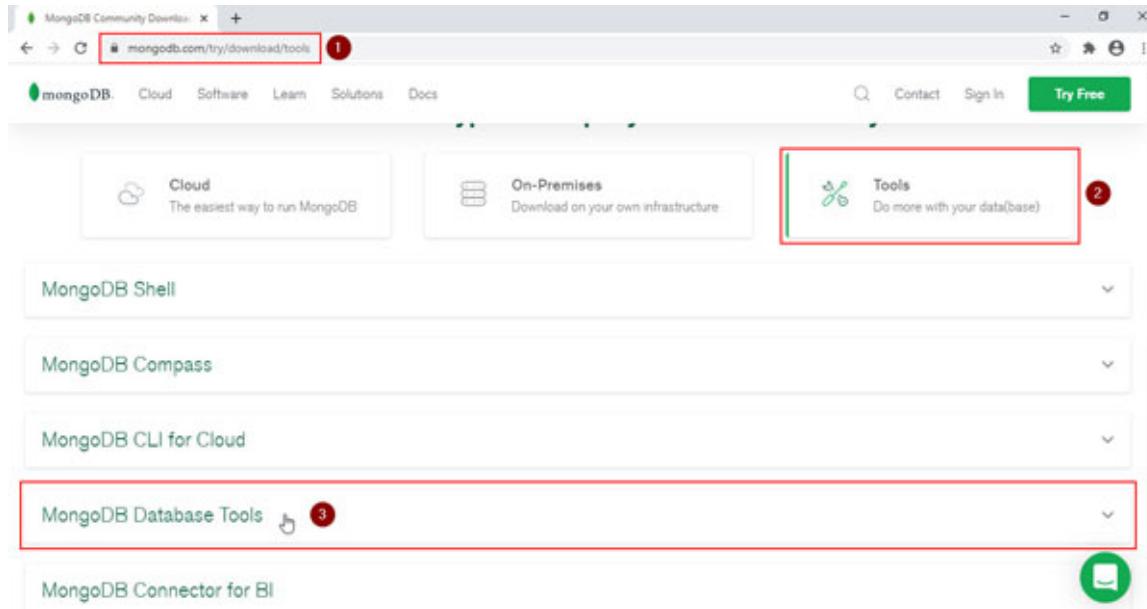
**Figure 17.18:** MongoDB Inc. Official Website Home Page

2. In the **MongoDB Community Edition** page, you will see the 3 sections (or cards) on the top. Click the section (or card) which has a title **Tools**, as shown in the following screenshot:



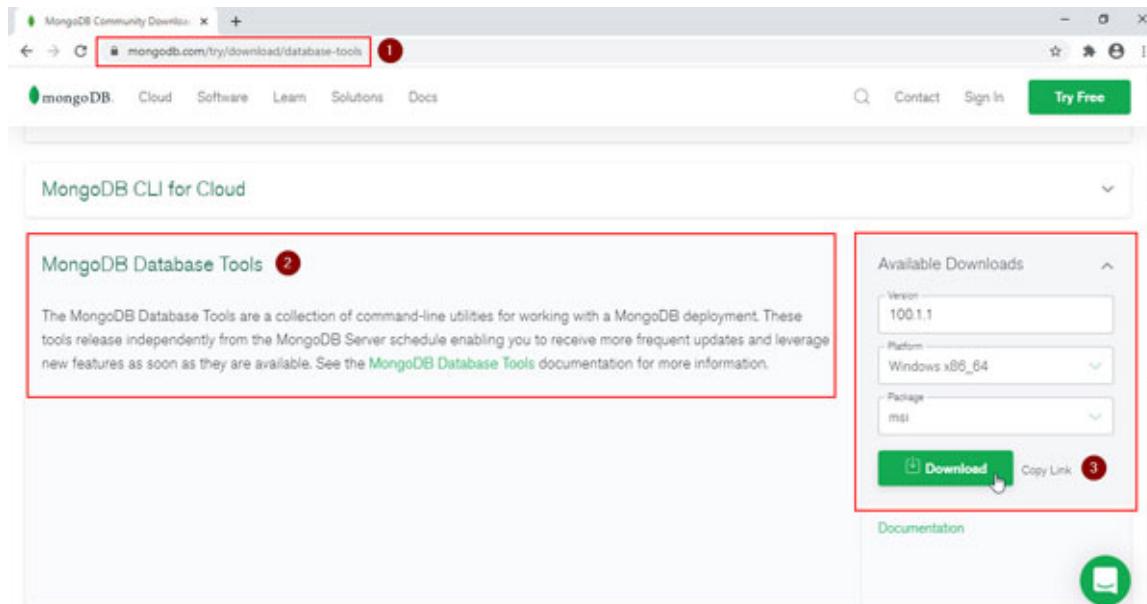
**Figure 17.19:** MongoDB Inc. Official Website – Community Page – Tools Section

3. Once you click on the **Tools**, this will open the **MongoDB Tools** screen which has the MongoDB tools related links at the bottom of the page, as shown in the following screenshot:



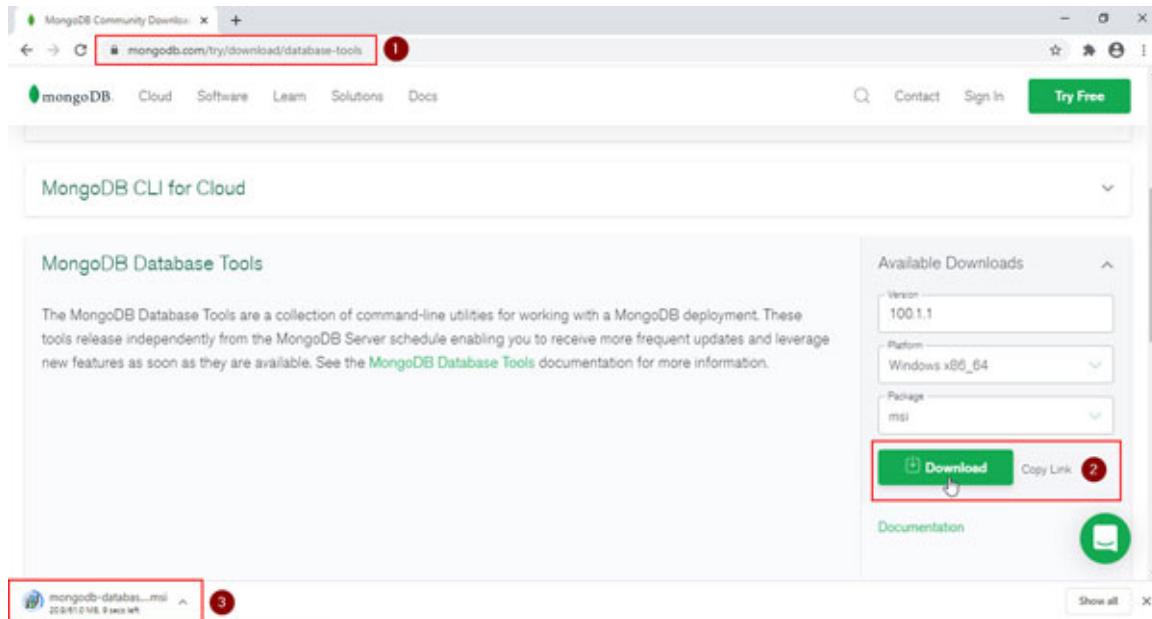
**Figure 17.20:** MongoDB Inc. Official Website – Tools Screen - MongoDB Database Tools Link

- Click on the [MongoDB Database Tools](#). This will further expand the **MongoDB Database Tools** screen. Here, you will see the **Available Download** section at the right side of the page. Select the **Package** option as MSI (which is a Windows installer file) and click on the **Download** button, as shown in the following screenshot:



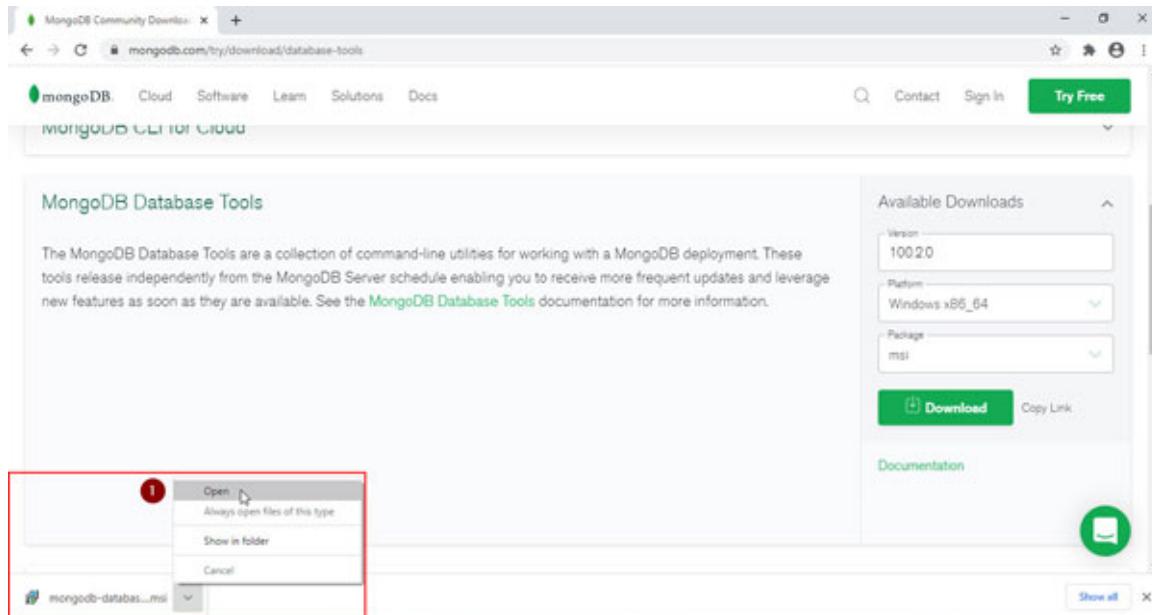
**Figure 17.21:** MongoDB Inc. Official Website – MongoDB Database Tools Download Section

- Once you click the **Download** button, the download process will start, as shown in the following screenshot:



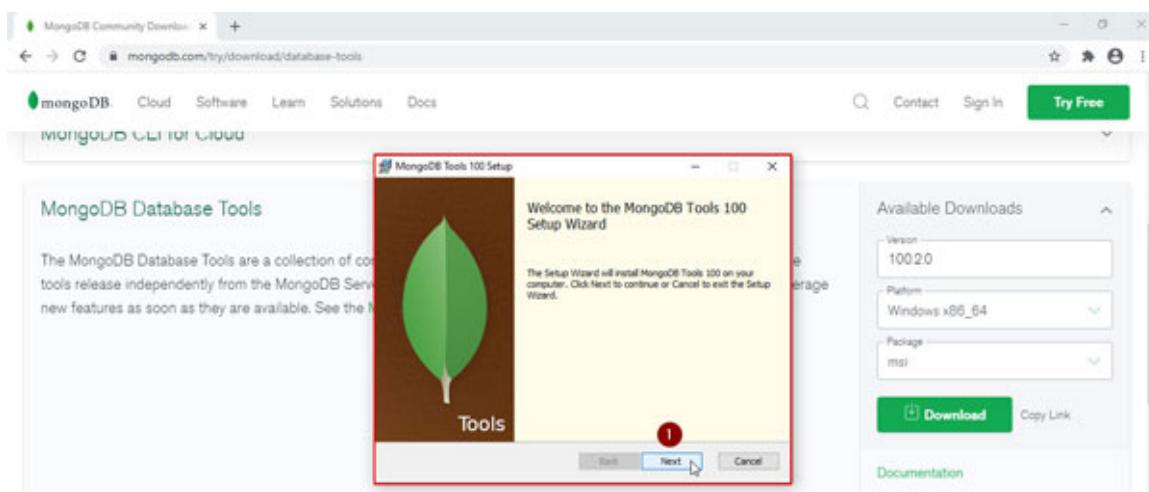
**Figure 17.22: MongoDB Inc. Official Website – MongoDB Database Tools - Download**

- After the download is 100% complete, we can now open this Windows installer file so that the installer can install the MongoDB tools, as shown in the following screenshot:



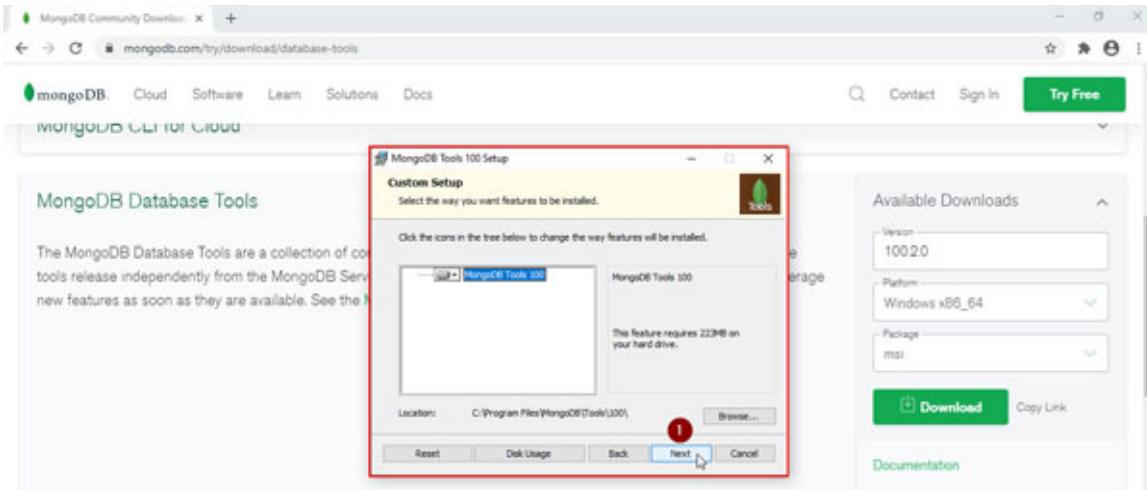
**Figure 17.23: MongoDB Inc. Official Website – MongoDB Database Tools – Download Complete**

7. After you click on the **open** button, the MongoDB tools setup wizard will launch and it will guide you to install the MongoDB tools on your machine. You can click on the **Next** button to continue. During the MongoDB tools installation process, the setup wizard will ask you to accept the **License Agreement** or other terms and conditions. It is suggested to go through them, as shown in the following screenshot:



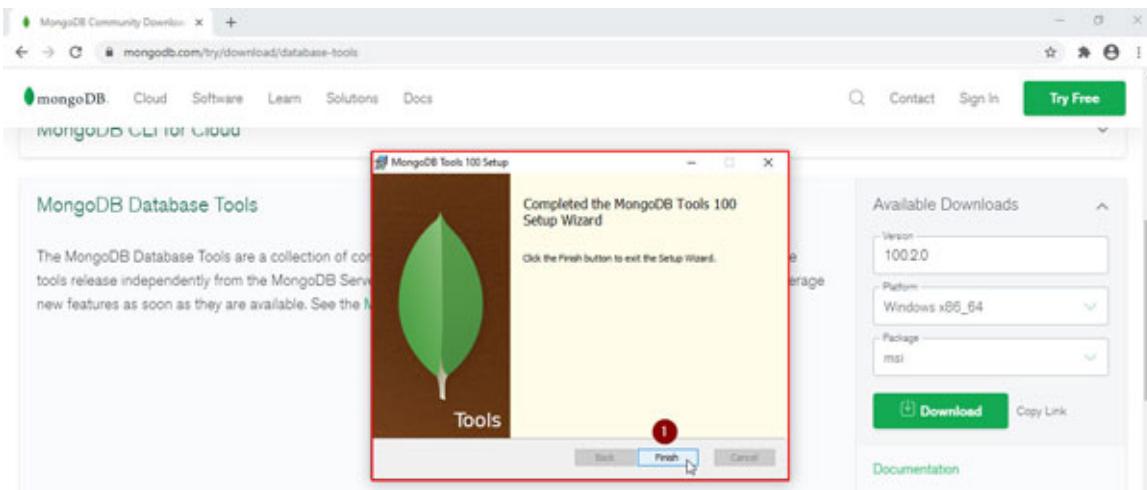
**Figure 17.24: MongoDB Database Tools - Setup**

8. The MongoDB tools setup wizard will ask you to choose the location to install the MongoDB tools in your local machine. It is recommended to keep the default settings as it is, unless you have some specific reason to choose another location, as shown in the following screenshot:



**Figure 17.25: MongoDB Database Tools – Setup Process**

- Once the setup wizard completes the installation of the MongoDB tools, click on the `Finish` button to complete the installation of MongoDB tools, as shown in the following screenshot:



**Figure 17.26: MongoDB Database Tools – Setup Process Complete**

## Verifying the installation of MongoDB tools and utilities

In order to verify the installation of the *MongoDB Tools*, follow these steps:

- Open the command prompt and navigate to the directory where you have installed MongoDB, which is, `C:\Program`

Files\MongoDB\Tools, in our case, as shown in the following screenshot:



```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\manus>cd "c:\Program Files\MongoDB\Tools" ①
c:\Program Files\MongoDB\Tools> ②
```

**Figure 17.27: Command Prompt - Verifying the Installation of MongoDB Tools and Utilities**

2. Now, further navigate to the `bin` directory which is under the version directory of the MongoDB tools, which is, `C:\Program Files\MongoDB\Tools\100\bin`, in our case, as shown in the following screenshot:

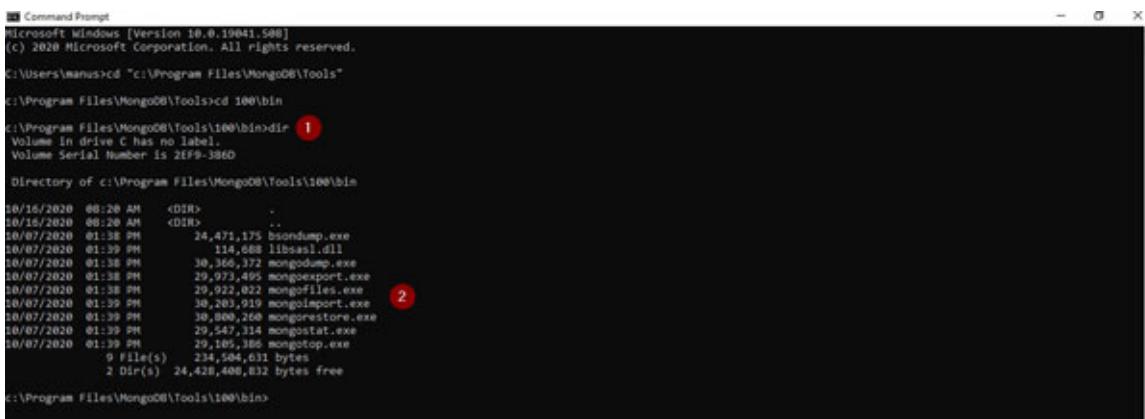


```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\manus>cd "c:\Program Files\MongoDB\Tools"
c:\Program Files\MongoDB\Tools>cd 100\bin ①
c:\Program Files\MongoDB\Tools\100\bin> ②
```

**Figure 17.28: Command Prompt - Verifying the Installation of MongoDB Tools and Utilities – Navigating to "bin" Directory**

3. Now, to list the contents of this directory, type `dir` command which will print the contents of this directory and you will see a lot of `.exe` (windows executable) files in the `bin` directory. Each of these files are helpful and together, these are a bundle of MongoDB tools and utilities, as shown in the following screenshot:



```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\manus>cd "c:\Program Files\MongoDB\Tools"
c:\Program Files\MongoDB\Tools>cd 100\bin
c:\Program Files\MongoDB\Tools\100\bin>dir ①
Volume in drive C has no label
Volume Serial Number is 2EFF-386D

Directory of c:\Program Files\MongoDB\Tools\100\bin

28/07/2020 01:39 AM <DIR> .
28/07/2020 01:39 AM <DIR> ..
28/07/2020 01:39 PM 24,471,175 bsondump.exe ②
28/07/2020 01:39 PM 114,688 libssl.dll
28/07/2020 01:39 PM 30,366,372 mongodump.exe
28/07/2020 01:39 PM 29,073,495 mongoexport.exe
28/07/2020 01:39 PM 29,922,032 mongofiles.exe
28/07/2020 01:39 PM 30,283,919 mongoimport.exe
28/07/2020 01:39 PM 30,080,260 mongorestore.exe
28/07/2020 01:39 PM 29,547,314 mongostat.exe
28/07/2020 01:39 PM 29,185,386 mongotop.exe
28/07/2020 01:39 PM 234,584,631 bytes
2 Dir(s) 24,428,408,832 bytes free

c:\Program Files\MongoDB\Tools\100\bin>
```

**Figure 17.29: Command Prompt - Verifying the Installation of MongoDB Tools and Utilities – Viewing the contents of "bin" Directory**

## Working with MongoDB tools and utilities

## [mongostat](#)

This MongoDB tool helps to view the database operations. It gives the statistics of the database operations, like insert, query, update, delete, etc. To use this tool, navigate to the `bin` directory under the MongoDB tools installation parent directory and type `mongostat` and press the enter key to execute this tool. Once this tool executes, it will continuously give you all the data related to the MongoDB database operations, as shown in the following screenshot:

```
c:\Program Files\MongoDB\Tools\100\bin>mongostat 1
insert query update delete getmore command dirty used flushes vsize  res qrw arw net_in net_out conn time
*0 *0 *0 *0 0 0|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 41.8k 3 Oct 17 14:08:17.248
*0 *0 *0 *0 0 0|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 41.5k 3 Oct 17 14:08:18.257
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 42.2k 3 Oct 17 14:08:19.250
*0 *0 *0 *0 0 0|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 41.6k 3 Oct 17 14:08:20.257
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 42.1k 3 Oct 17 14:08:21.253
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 42.3k 3 Oct 17 14:08:22.244
*0 *0 *0 *0 0 0|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 41.8k 3 Oct 17 14:08:23.247
*0 *0 *0 *0 0 0|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 41.6k 3 Oct 17 14:08:24.253
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 42.0k 3 Oct 17 14:08:25.249
*0 *0 *0 *0 0 2|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 30|b 42.3k 3 Oct 17 14:08:26.254
insert query update delete getmore command dirty used flushes vsize  res qrw arw net_in net_out conn time 2
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 41.9k 3 Oct 17 14:08:27.254
*0 *0 *0 *0 0 0|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 41.7k 3 Oct 17 14:08:28.258
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 42.1k 3 Oct 17 14:08:29.254
*0 *0 *0 *0 0 0|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 41.8k 3 Oct 17 14:08:30.257
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 41.9k 3 Oct 17 14:08:31.257
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 42.1k 3 Oct 17 14:08:32.251
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 42.0k 3 Oct 17 14:08:33.249
*0 *0 *0 *0 0 1|0 0.05 0.0% 0 5.38G 21.0M 0|0 1|0 11|b 42.3k 3 Oct 17 14:08:34.244
```

**Figure 17.30: Working with MongoDB Tools and Utilities - mongostat**

You can see that there is no activity related to any database operations as of now and all the operations counts can be seen with zero count (0). Now, let's open the MongoDB Compass and try to view some MongoDB collections and keep on running the `mongostat` in the command prompt. As we do this, we can see that there is an increase under the database operations `query` counts, as shown in the following screenshot:

```

c:\Program Files\WongoDB\Tools\bin>mongostat
insert query update delete getmore command dirty used flushes vsize    res qps arw net_in net_out conn      time
+0 +0 +0 +0 0 0|0.0% 0.0% 0 5.390 21.00 0|0 1|0 112b 41.6k 3 Oct 17 14:14:25.384
+0 +0 +0 +0 0 0|0.0% 0.0% 0 5.390 21.00 0|0 1|0 112b 41.6k 3 Oct 17 14:14:26.391
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 21.00 0|0 1|0 112b 42.2k 3 Oct 17 14:14:27.383
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 21.00 0|0 1|0 112b 41.7k 3 Oct 17 14:14:28.380
+0 +0 +0 +0 0 17|0 0.0% 0.0% 0 5.390 23.00 0|0 1|0 112b 42.1k 3 Oct 17 14:14:29.383
+0 +0 +0 +0 0 6|0 0.0% 0.0% 0 5.390 23.00 0|0 1|0 809b 45.4k 9 Oct 17 14:14:31.377
+0 +0 +0 +0 0 1|0 0.0% 0.0% 1 5.390 23.00 0|0 1|0 112b 41.9k 9 Oct 17 14:14:32.376
+0 +0 +0 +0 0 0|0 0.0% 0.0% 0 5.390 23.00 0|0 1|0 110b 41.5k 9 Oct 17 14:14:33.385
+0 +0 +0 +0 0 3|0 0.0% 0.0% 0 5.390 23.00 0|0 1|0 309b 42.7k 9 Oct 17 14:14:34.381
insert query update delete getmore command dirty used flushes vsize    res qps arw net_in net_out conn      time
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 23.00 0|0 1|0 112b 42.0k 9 Oct 17 14:14:35.378
+0 +0 +0 +0 0 2|0 0.0% 0.0% 0 5.390 23.00 0|0 1|0 245b 60.2k 9 Oct 17 14:14:36.375
+0 +0 +0 +0 0 5|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 1.03k 59.8k 9 Oct 17 14:14:37.383
+0 1 +0 +0 0 7|0 0.0% 0.0% 0 5.390 24.00 0|0 2|0 1.35k 79.4k 9 Oct 17 14:14:38.381
+0 +0 +0 +0 0 0|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 112b 58.2k 9 Oct 17 14:14:39.380
+0 +0 +0 +0 0 2|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 164b 42.3k 9 Oct 17 14:14:40.385
+0 +0 +0 +0 0 0|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 115b 41.9k 9 Oct 17 14:14:41.386
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 112b 41.9k 9 Oct 17 14:14:42.384
+0 +0 +0 +0 0 0|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 115b 41.9k 9 Oct 17 14:14:43.385
+0 +0 +0 +0 0 3|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 308b 42.5k 9 Oct 17 14:14:44.384
insert query update delete getmore command dirty used flushes vsize    res qps arw net_in net_out conn      time
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 112b 42.4k 9 Oct 17 14:14:45.380
+0 +0 +0 +0 0 2|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 245b 60.6k 9 Oct 17 14:14:46.374
+0 +0 +0 +0 0 0|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 2.24k 97.4k 9 Oct 17 14:14:47.374
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 310b 41.4k 9 Oct 17 14:14:48.374
+0 +0 +0 +0 0 0|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 112b 41.4k 9 Oct 17 14:14:49.374
+0 +0 +0 +0 0 2|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 166b 42.9k 9 Oct 17 14:14:50.374
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 242b 59.7k 9 Oct 17 14:14:51.382
+0 +0 +0 +0 0 0|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 2.26k 114k 9 Oct 17 14:14:52.386
+0 +0 +0 +0 0 3|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 111b 41.8k 9 Oct 17 14:14:53.387
+0 +0 +0 +0 0 0|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 310b 42.8k 9 Oct 17 14:14:54.379
insert query update delete getmore command dirty used flushes vsize    res qps arw net_in net_out conn      time
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 112b 42.2k 9 Oct 17 14:14:55.372
+0 2 +0 +0 0 34|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 5.76k 154k 9 Oct 17 14:14:56.382
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 112b 42.2k 9 Oct 17 14:14:57.373
+0 +0 +0 +0 0 0|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 110b 41.3k 9 Oct 17 14:14:58.388
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 112b 42.1k 9 Oct 17 14:14:59.383
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 162b 41.9k 9 Oct 17 14:15:00.390
+0 +0 +0 +0 0 1|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 112b 42.1k 9 Oct 17 14:15:01.385
+0 +0 +0 +0 0 11|0 0.0% 0.0% 0 5.390 24.00 0|0 1|0 512b 42.3k 9 Oct 17 14:15:02.378

```

**Figure 17.31:** Working with MongoDB Tools and Utilities – mongostat – Increase in "query" counts

If you do any other database operations, like insert, update, or delete, then `mongostat` will show this to you. We have seen that `mongostat` is a very helpful tool to monitor the MongoDB server database operations and we can use this tool in our day-to-day MongoDB monitoring purposes.

## [mongotop](#)

This MongoDB tool helps to view the read and write activities of the MongoDB instances at collection level. It gives the statistics of the collection level activity which are the total number of reads and writes per collection.

To use this tool, navigate to the `bin` directory under the MongoDB tools installation parent directory and type `mongotop` and press the enter key to execute this tool. Once this tool executes, it will continuously give you all the data related to the MongoDB collections read and write activities, as shown in the following screenshot:

```

c:\Program Files\MongoDB\Tools\100\bin>mongotop
2021-05-13T21:35:16.466+0530 connected to: mongodb://localhost/

```

ns	total	read	write	date
BPBCatalogDB.BPBCatalogCollection	0ms	0ms	0ms	
BPBOnlineDB.BPBOnlineCollection	0ms	0ms	0ms	
admin.system.roles	0ms	0ms	0ms	
admin.system.version	0ms	0ms	0ms	
config.transactions	0ms	0ms	0ms	
local.oplog.rs	0ms	0ms	0ms	
local.system.replset	0ms	0ms	0ms	
BPBCatalogDB.BPBCatalogCollection	0ms	0ms	0ms	2021-05-13T21:35:18+05:30
BPBOnlineDB.BPBOnlineCollection	0ms	0ms	0ms	
admin.system.roles	0ms	0ms	0ms	
admin.system.version	0ms	0ms	0ms	
config.transactions	0ms	0ms	0ms	
local.oplog.rs	0ms	0ms	0ms	
local.system.replset	0ms	0ms	0ms	
BPBCatalogDB.BPBCatalogCollection	0ms	0ms	0ms	2021-05-13T21:35:19+05:30
BPBOnlineDB.BPBOnlineCollection	0ms	0ms	0ms	
admin.system.roles	0ms	0ms	0ms	
admin.system.version	0ms	0ms	0ms	
config.transactions	0ms	0ms	0ms	
local.oplog.rs	0ms	0ms	0ms	
local.system.replset	0ms	0ms	0ms	

**Figure 17.32:** Working with MongoDB Tools and Utilities – mongotop

You can see that there is no read or write activity related to any collection as of now, and all the activities counts can be seen with the zero count (0). Now, let's open the MongoDB Compass and try to view some MongoDB collections and keep on running `mongotop` in the command prompt. As we do this, we can see that there is an increase under the collection activity `read` counts, as shown in the following screenshot:

```

c:\Program Files\MongoDB\Tools\100\bin>mongotop
2020-10-17T14:33:55.970+0530 connected to: mongodb://localhost/

```

ns	total	read	write	date
BPBCatalogDB.BPBCatalogCollection	0ms	0ms	0ms	
BPBOnlineDB.BPBOnlineCollection	0ms	0ms	0ms	
admin.system.roles	0ms	0ms	0ms	
admin.system.version	0ms	0ms	0ms	
config.transactions	0ms	0ms	0ms	
local.oplog.rs	0ms	0ms	0ms	
local.system.replset	0ms	0ms	0ms	
BPBCatalogDB.BPBCatalogCollection	20ms	20ms	0ms	2020-10-17T14:33:57+05:30
BPBOnlineDB.BPBOnlineCollection	2ms	7ms	0ms	
admin.system.roles	0ms	0ms	0ms	
admin.system.version	0ms	0ms	0ms	
config.transactions	0ms	0ms	0ms	
local.oplog.rs	0ms	0ms	0ms	
local.system.replset	0ms	0ms	0ms	
BPBCatalogDB.BPBCatalogCollection	10ms	10ms	0ms	2020-10-17T14:33:58+05:30
BPBOnlineDB.BPBOnlineCollection	7ms	7ms	0ms	
admin.system.roles	0ms	0ms	0ms	
admin.system.version	0ms	0ms	0ms	
config.transactions	0ms	0ms	0ms	
local.oplog.rs	0ms	0ms	0ms	
local.system.replset	0ms	0ms	0ms	
BPBCatalogDB.BPBCatalogCollection	20ms	20ms	0ms	2020-10-17T14:33:59+05:30
BPBOnlineDB.BPBOnlineCollection	0ms	0ms	0ms	
admin.system.roles	0ms	0ms	0ms	
admin.system.version	0ms	0ms	0ms	
config.transactions	0ms	0ms	0ms	
local.oplog.rs	0ms	0ms	0ms	
local.system.replset	0ms	0ms	0ms	

**Figure 17.33:** Working with MongoDB Tools and Utilities – mongotop – Increase in collection activity "read" counts

If you do any other database operations, like insert, update or delete, then `mongotop` will show the increase in `write` counts due to the write activities. So, we have seen that `mongotop` is a very helpful tool to monitor the MongoDB server collection related activities and we can use this tool in our day-to-day MongoDB collection monitoring purposes.

There are some other ways to monitor MongoDB by using the MongoDB Shell commands and methods. To use these commands, we first need to log into the MongoDB Shell, as shown in the following screenshot:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d5c5fbcd-c8fa-4c3d-94c5-d0dd5dcff55") }
MongoDB server version: 4.4.1

The server generated these startup warnings when booting:
2020-10-11T18:15:35.857+05:30: ***** SERVER RESTARTED *****
2020-10-11T18:15:38.063+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
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()

> 2
```

**Figure 17.34: Working with MongoDB Tools and Utilities – log into MongoDB Shell**

## serverStatus

This MongoDB Shell command returns the state of the MongoDB instance in a document form. When you run this command, it will return the state information of the MongoDB instance. To run the command, type the following command in your MongoDB Shell, as shown in the following screenshot:

```
db.runCommand({serverStatus: 1})
```

```

> db.runCommand( { serverStatus: 1 } )
{
  "host" : "DESKTOP-10665QW",
  "version" : "4.4.1",
  "process" : "C:\Program Files\MongoDB\Server\4.4\bin\mongod.exe",
  "pid" : NumberLong(12224),
  "uptime" : 50658,
  "uptimeMillis" : NumberLong(50658293),
  "uptimeEstimate" : NumberLong(506582),
  "localtime" : ISODate("2020-10-17T09:28:38.784Z"),
  "asserts" : {
    "regular" : 0,
    "warning" : 0,
    "msg" : 0,
    "user" : 10,
    "rollovers" : 0
  },
  "connections" : {
    "current" : 2,
    "available" : 999998,
    "totalCreated" : 39,
    "active" : 1,
    "exhaustingMaster" : 0,
    "awaitingTopologyChanges" : 0
  },
  "electionMetrics" : {
    "stepUpCmd" : {
      "called" : NumberLong(0),
      "successful" : NumberLong(0)
    },
    "priorityTakeover" : {
      "called" : NumberLong(0),
      "successful" : NumberLong(0)
    },
    "catchUpTakeover" : {
      "called" : NumberLong(0),
      "successful" : NumberLong(0)
    },
    "electionTimeout" : {
      "called" : NumberLong(0),
      "successful" : NumberLong(0)
    },
    "freezeTimeout" : {
      "called" : NumberLong(0),
      "successful" : NumberLong(0)
    }
  }
}

```

**Figure 17.35:** Working with MongoDB Tools and Utilities – MongoDB Shell - serverStatus

Thus, by using this command, you will be able to see all the related details which are very helpful in gathering the current state of the MongoDB instance.

## dbStats

This MongoDB Shell command returns the storage details of the MongoDB database. When you run this command, it will return the storage information of the given MongoDB database. To run the command, first select the database for which you would like to get the information and then type the following command in your MongoDB Shell, as shown in the following screenshot:

```
db.runCommand({dbStats: 1})
```

```

> use BPBCatalogDB
switched to db BPBCatalogDB
> db.runCommand( { dbStats: 1 } )
{
  "db" : "BPBCatalogDB",
  "collections" : 1,
  "views" : 0,
  "objects" : 8,
  "avgObjSize" : 1028.875,
  "dataSize" : 8231,
  "storageSize" : 45056,
  "indexes" : 1,
  "indexSize" : 36864,
  "totalSize" : 81928,
  "scaleFactor" : 1,
  "fsUsedSize" : 105684963328,
  "fsTotalSize" : 130389037856,
  "ok" : 1
}

```

**Figure 17.36:** Working with MongoDB Tools and Utilities – MongoDB Shell - dbStats

Thus, by using this command, you will be able to see all the related storage details of the MongoDB database.

## collStats

This MongoDB Shell command returns the storage details of the MongoDB collection. When you run this command, it will return the storage information of the given MongoDB collection. To run the command, first select the database for which you would like to get the information and then type the following command in your MongoDB Shell, as shown in the following screenshot:

```
db.runCommand({collStats: "<collection-name>"})
```

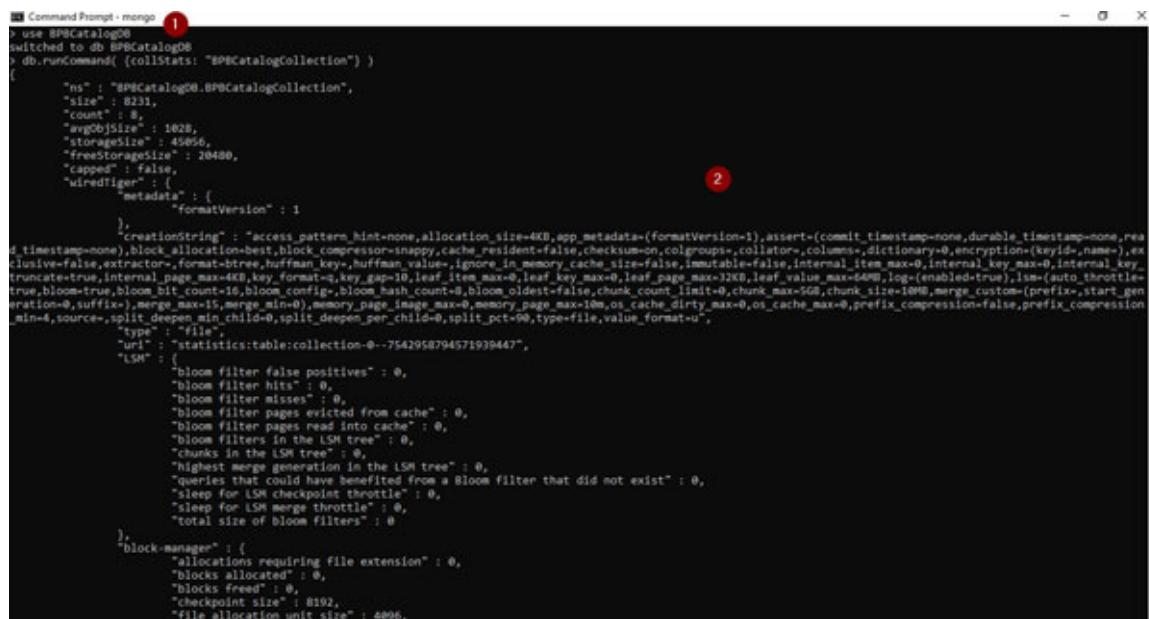
Here, the `<collection-name>` is the name of the collection.

In our example, we are using the Database as `BPBCatalogDB` and the collection as `BPBCatalogCollection`. First, we need to select the DB with the following command:

```
use BPBCatalogDB
```

Then, we need to run the following command:

```
db.runCommand({collStats: "BPBCatalogCollection"})
```



```
! Command Prompt - mongo
> use BPBCatalogDB
switched to db BPBCatalogDB
> db.runCommand({collStats: "BPBCatalogCollection"})
{
  "ns" : "BPBCatalogDB.BPBCatalogCollection",
  "size" : 8231,
  "count" : 8,
  "avgObjSize" : 1628,
  "storageSize" : 45856,
  "freeStorageSize" : 20480,
  "capped" : false,
  "wiredTiger" : {
    "metadata" : {
      "formatVersion" : 1
    },
    "creationString" : "access_pattern_hint:none,allocation_size=4KB,app_metadata=(formatVersion=1),assert=(commit_timestamp=None,durable_timestamp=None,read_timestamp=None),block_allocation=best,block_compressor=snappy,cache_resident=False,checksum_on,colgroups=,columns=,dictionary=0,encryption=(keyId=,name=),exclusive=False,extractor=,format=btree,huffman_key=huffman_value=,ignore_in_memory_cache_size=False,immutable=False,internal_item_max=0,internal_key_max=0,internal_key_truncate=False,internal_page_max=4KB,key_format=q,key_gap=10,leaf_item_max=0,leaf_key_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=True),lsm=(auto_throttle=true,bloom=true,bloom_bit_count=16,bloom_config=,bloom_hash_count=8,bloom.oldest=false,chunk_count_limit=0,chunk_max=50B,chunk_size=10MB,merge_custom=(prefix=,start_gen=4,suffix=),merge_max=15,merge_min=0),memory_page_image_max=0,memory_page_max=10m,os_cache_dirty_max=0,os_cache_max=0,prefix_compression=false,prefix_compression_min=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=98,type=file,value_format=u",
      "type" : "file",
      "url" : "statistics:table:collection-0--7542958794571939447",
      "LSM" : {
        "bloom filter false positives" : 0,
        "bloom filter hits" : 0,
        "bloom filter misses" : 0,
        "bloom filter pages evicted from cache" : 0,
        "bloom filter pages read into cache" : 0,
        "bloom filters in the LSM tree" : 0,
        "chunks in the LSM tree" : 0,
        "highest merge generation in the LSM tree" : 0,
        "queries that could have benefited from a Bloom filter that did not exist" : 0,
        "sleep for LSM checkpoint throttle" : 0,
        "sleep for LSM merge throttle" : 0,
        "total size of bloom filters" : 0
      }
    },
    "block-manager" : {
      "allocations requiring file extension" : 0,
      "blocks allocated" : 0,
      "blocks freed" : 0,
      "checkpoint size" : 8192,
      "file allocation unit size" : 4096
    }
  }
}
```

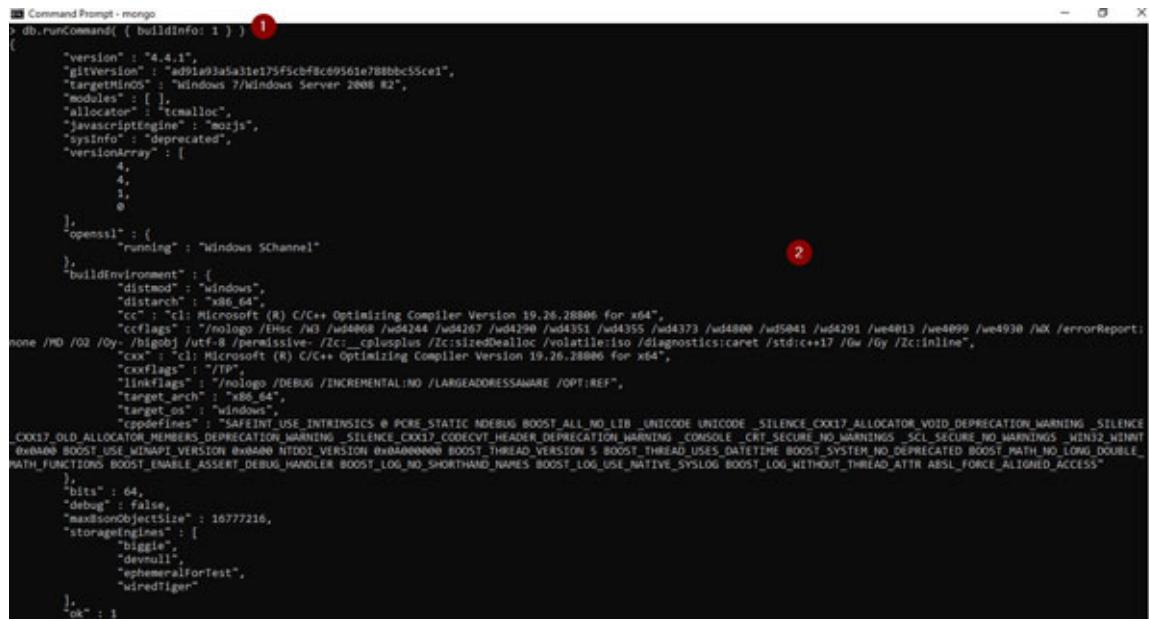
**Figure 17.37: Working with MongoDB Tools and Utilities – MongoDB Shell - collStats**

Thus, by using this command, you will be able to see all the storage related details of the MongoDB collections in a database.

## buildInfo

This MongoDB Shell command returns the details of the current build of mongod. You can run the following command to get the details about the mongod build, as shown in the following screenshot:

```
db.runCommand({buildInfo: 1})
```



```
1> db.runCommand( { buildInfo: 1 } ) ①
{
  "version": "4.4.1",
  "gitVersion": "ad91a93a5a31e175f5cbf8c69561e788bbc55ce1",
  "targetMnOS": "Windows ?/Windows Server 2008 R2",
  "modules": [],
  "allocator": "tcmalloc",
  "javascriptEngine": "mozjs",
  "sysInfo": "deprecated",
  "versionArray": [
    4,
    4,
    1,
    0
  ],
  "openssl": {
    "running": "Windows SChannel"
  },
  "buildEnvironment": {
    "distmod": "windows",
    "distarch": "x86_64",
    "cc": "cl Microsoft (R) C/C++ Optimizing Compiler Version 19.26.28806 for x64",
    "cclflags": "/nologo /O2c /W3 /wd4468 /wd4244 /wd4267 /wd4290 /wd4351 /wd4355 /wd4373 /wd4800 /wd5041 /wd4291 /we4013 /we4099 /we4938 /WX /errorReport:none /MD /O2 /Oy- /Dnogdb /utf8 /permissive- /Zc:_cpplus /Zc:sizedAlloc /volatile:iso /diagnostics:caret /std:c++17 /Ow /Oy /Zc:inline",
    "cxx": "cl Microsoft (R) C/C++ Optimizing Compiler Version 19.26.28806 for x64",
    "cxxflags": "/TP",
    "linkflags": "/nologo /DEBUG /INCREMENTAL:NO /LARGEADDRESSAWARE /OPT:REF",
    "target_arch": "x86_64",
    "target_os": "windows",
    "cppdefines": "#SAFEINT USE_INTRINSICS # PORE_STATIC NDEBUG BOOST_ALL_NO_LIB _UNICODE UNICODE _SILENCE_CXX17_ALLOCATOR_VOID_DEPRECATED_WARNING _SILENCE_CXX17_OLD_ALLOCATOR_MEMBERS_DEPRECATED_WARNING _SILENCE_CXX17_CODECVT_HEADER_DEPRECATED_WARNING _CONSOLE _CRT_SECURE_NO_WARNINGS _SCL_SECURE_NO_WARNINGS _WIN32 _WINNT _WIN64 BOOST_USE_WINAPI VERSION 0x8000 NTDDI VERSION 0x80000000 BOOST_THREAD_VERSION 5 BOOST_THREAD_USES_DATETIME BOOST_SYSTEM_NO_DEPRECATED BOOST_MATH_NO_LONG_DOUBLE_MATH_FUNCTIONS BOOST_ENABLE_ASSERT_DEBUG_HANDLER BOOST_LOG_NO_SHORTHAND_NAMES BOOST_LOG_USE_NATIVE_SYSLOG BOOST_LOG_WITHOUT_THREAD_ATTR ABSL_FORCE_ALIGNED_ACCESS",
    "bits": 64,
    "debug": false,
    "maxsonObjectSize": 16777216,
    "storageEngines": [
      "bigfile",
      "devnull",
      "ephemeralForTest",
      "wiredTiger"
    ],
    "ok": 1
  }
} ②
```

**Figure 17.38: Working with MongoDB Tools and Utilities – MongoDB Shell – buildInfo**

Thus, by using this command, you will be able to see all the build related details of "mongod".

## hostInfo

This MongoDB Shell command returns the system information of mongod or mongos (in case of sharding, where we have sharded cluster). You can run the following command to get the details about the system information, as shown in the following screenshot:

```
db.hostInfo()
```

or

```
db.adminCommand({ "hostInfo" : 1 })
```

```
> db.hostInfo() 1
{
  "system": {
    "currentTime": ISODate("2020-10-17T11:56:15.073Z"),
    "hostname": "DESKTOP-T8665QR",
    "cpuidleSize": 64,
    "memsizeMB": NumberLong(8096),
    "memlimitMB": NumberLong(8096),
    "numCores": 4,
    "cpuArch": "x86_64",
    "numaEnabled": false
  },
  "os": {
    "type": "Windows",
    "name": "Microsoft Windows 10",
    "version": "10.0 (build 19041)"
  },
  "extra": {
    "pageSize": NumberLong(4096)
  },
  "ok": 1
}
> db.adminCommand( { "hostInfo" : 1 } ) 3
{
  "system": {
    "currentTime": ISODate("2020-10-17T11:56:29.201Z"),
    "hostname": "DESKTOP-T8665QR",
    "cpuidleSize": 64,
    "memsizeMB": NumberLong(8096),
    "memlimitMB": NumberLong(8096),
    "numCores": 4,
    "cpuArch": "x86_64",
    "numaEnabled": false
  },
  "os": {
    "type": "Windows",
    "name": "Microsoft Windows 10",
    "version": "10.0 (build 19041)"
  },
  "extra": {
    "pageSize": NumberLong(4096)
  },
  "ok": 1
} 4
```

**Figure 17.39:** Working with MongoDB Tools and Utilities – MongoDB Shell – hostInfo

Thus, by using this command, you will be able to see all the system information of "mongod" or "mongos".

## [listCommands](#)

This MongoDB Shell command returns the list of all the database related commands that are supported by the current version of `mongod` or `mongos` (in case of sharding, where we have sharded clusters), as shown in the following screenshot:

```
db.runCommand({listCommands: 1})
```

```

> db.runCommand( { listCommands: 1 } ) ①
{
  "commands": [
    "_addShard": {
      "help": "Internal command, which is exported by shards. Do not call directly. Adds a new shard to a cluster.",
      "requiresAuth": true,
      "slaveOk": false,
      "adminOnly": true
    },
    "_cloneCollectionOptionsFromPrimaryShard": {
      "help": "Internal command, do not call directly. Creates a collection on a shard with UUID existing on primary.",
      "requiresAuth": true,
      "slaveOk": false,
      "adminOnly": false
    },
    "_configsvrAddShard": {
      "help": "Internal command, which is exported by the sharding config server. Do not call directly. Validates and adds a new shard to the cluster",
      "requiresAuth": true,
      "slaveOk": false,
      "adminOnly": true
    },
    "_configsvrAddShardToZone": {
      "help": "Internal command, which is exported by the sharding config server. Do not call directly. Validates and adds a new zone to the shard.",
      "requiresAuth": true,
      "slaveOk": false,
      "adminOnly": true
    },
    "_configsvrBalancerCollectionStatus": {
      "help": "Internal command, which is exported by the sharding config server. Do not call directly. Checks whether the chunks of a given collection are in a quiesced state or the ones may which need to be moved because of (1) draining shards, (2) zone violation or (3) imbalance between shards.",
      "requiresAuth": true,
      "slaveOk": false,
      "adminOnly": true
    },
    "_configsvrBalancerStart": {
      "help": "Internal command, which is exported by the sharding config server. Do not call directly. Controls the balancer state.",
      "requiresAuth": true,
      "slaveOk": false,
      "adminOnly": true
    },
    "_configsvrBalancerStatus": {
      "help": "Internal command, which is exported by the sharding config server. Do not call directly. Controls the balancer state."
    }
  ]
}

```

**Figure 17.40: Working with MongoDB Tools and Utilities – MongoDB Shell – listCommands**

Thus, by using this command, you will get a long list of all the database related commands with their details, as shown in the following screenshot:

```

{
  "serverStatus": {
    "help": "returns lots of administrative server statistics",
    "requiresAuth": true,
    "slaveOk": true,
    "adminOnly": false
  }
},
setDefaultRwConcern": {
  "help": "set the current read/write concern defaults (cluster-wide)"
}

```

We get all the information related to the specific command

**Figure 17.41: Working with MongoDB Tools and Utilities – MongoDB Shell – listCommands (Specific Command Information)**

## ping

This MongoDB Shell command is very helpful to diagnosing whether the server is responding to the commands or not. This command even works when the server is write-locked, as shown in the following screenshot:

```
db.runCommand({ping: 1})
```

```

> db.runCommand( { ping: 1 } ) ①
{ "ok": 1 } ②
>

```

**Figure 17.42:** Working with MongoDB Tools and Utilities – MongoDB Shell – ping

Thus, by using this command, you can diagnose the server whether it is responding to the other database commands or not.

## **getLog**

This MongoDB Shell command is very helpful in the administrative command which will print the list of the latest 1024 logs logged by `mongod` events. This command takes one of the three values described, as shown in the following screenshot:

```
db.adminCommand({getLog: "<value>"})
```

Here, `<value>` can have any one of the following three types:

- \*— All the available values which we can use with the `getLog` command, as shown in [figure 17.43](#).
- `global` – It will print all recent log entries, as shown in [figure 17.44](#)
- `startupWarnings` – It will print the log entries that contain the warnings or error occurred during the "`mongod`" startup, if any. In case of no warnings or errors, this will print an empty array, as shown in the following screenshot:



```
1 Command Prompt - mongo
> db.adminCommand({ getLog: "*" })
2 { "names": [ "global", "startupWarnings" ], "ok": 1 }
```

**Figure 17.43:** Working with MongoDB Tools and Utilities – MongoDB Shell – getLog - `{getLog: "*"}`

Thus, by using `db.adminCommand({getLog: "*"})`, we can get the details of the available values that we can use with the `getLog` command. Using the `db.adminCommand({getLog: "global"})`, we will get the list of all the latest log entries, as shown in the following screenshot:

```

> db.adminCommand( { getLog: "global" } )
{
  2 "totalLinesWritten" : 194,
  "log" : [
    {
      "t": {
        "$date": "2020-10-11T18:15:35.857+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 28698,
      "ctx": "\033[1;32mmain\033[0m",
      "msg": "\033[1;32m***** SERVER RESTARTED *****\033[0m",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:35.859+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23285,
      "ctx": "\033[1;32mmain\033[0m",
      "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.729+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mASIO\033[0m",
      "id": 22601,
      "ctx": "\033[1;32mmain\033[0m",
      "msg": "No TransportLayer configured during NetworkInterface startup",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.730+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mNETWORK\033[0m",
      "id": 4648802,
      "ctx": "\033[1;32mmain\033[0m",
      "msg": "Implicit TCP FastOpen in use.",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.730+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23318,
      "ctx": "\033[1;32mmain\033[0m",
      "msg": "Trying to start Windows service '{ouffString serviceName}'",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mSTORAGE\033[0m",
      "id": 4615611,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "MongoDB starting",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23398,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Target operating system minimum version",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23400,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Build Info",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23401,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "MongoDB starting",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23402,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Windows Server 2008 R2",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23403,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Build Info",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23404,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "MongoDB starting",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23405,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Operating System",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23406,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Microsoft Windows 10",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23407,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Version 10.0 (build 19041)",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23408,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Options set by command line",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23409,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "MongoDB options",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 23410,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Config file C:\Program Files\MongoDB\Server\4.4\log\mongod.log",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mSTORAGE\033[0m",
      "id": 22270,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Storage engine to use detected by data files",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mSTORAGE\033[0m",
      "id": 22271,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "StorageEngine\\lrediger",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mSTORAGE\033[0m",
      "id": 22272,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Opening Wire Digger",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.734+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mSTORAGE\033[0m",
      "id": 22273,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "StorageEngine\\lrediger\\create_cache_size=3536M,session_max=38000,eviction_threads_min=4,threads_max=4,config_base=false,statistics={fast},log_enabled=true,archive=true,path_compressor_snappy,FileLockManager\\close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250,statistics_log={wait=0},verbose=[recover_progress,checkpoint_progress,compact_progress],\"",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    }
  ]
}

```

**Figure 17.44:** Working with MongoDB Tools and Utilities – MongoDB Shell – `getLog - {getLog: "global"}`

Using the `db.adminCommand({getLog: "startupWarnings"})`, we will get the list of all the log entries which are logged during the startup of the mongod due to any warnings or errors, as shown in the following screenshot:

```

> db.adminCommand( { getLog: "startupWarnings" } )
{
  2 "totalLinesWritten" : 2,
  "log" : [
    {
      "t": {
        "$date": "2020-10-11T18:15:35.857+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 28698,
      "ctx": "\033[1;32mmain\033[0m",
      "msg": "\033[1;32m***** SERVER RESTARTED *****\033[0m",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    },
    {
      "t": {
        "$date": "2020-10-11T18:15:36.063+05:30"
      },
      "s": "\n",
      "c": "\033[1;32mCONTROL\033[0m",
      "id": 22120,
      "ctx": "\033[1;32minitandlisten\033[0m",
      "msg": "Access control is not enabled for the database. Read and write access to data and configuration is unrestricted",
      "tags": [
        "\033[1;32mstartupWarnings\033[0m"
      ]
    }
  ]
}

```

**Figure 17.45:** Working with MongoDB Tools and Utilities – MongoDB Shell – `getLog - {getLog: "startupWarnings"}`

## Backup and restore with MongoDB

The MongoDB tools provide ways to take the backup of your MongoDB databases and restore them whenever they are required to be restored. In general, every organization (or a database administrator) must have a solid database backup policy in place according to your requirements. Database backups play a very important role. In case of any disastrous situation, if we have a database backup, we can easily restore them using the MongoDB tools.

So, keeping this in mind, we must always create database backups from time-to-time so that we can restore it, in case there is a need to do that.

## Taking a MongoDB backup using mongodump

By using the `mongodump` tool (utility), we can very easily take backups of our MongoDB database. We can use the `mongodump` with the following options:

### Simple method

```
mongodump --host=<hostname> --port=<port-number> --db=<db-name> --  
collection=<collection-name>out=<output-path>
```

### With access control

```
mongodump--host=<hostname> --port=<port-number>--db=<db-name> --  
collection=<collection-name>--username=<user> --  
authenticationDatabase=admin --out=<output-path>
```

Where:

- <hostname>: Hostname of MongoDB, where the MongoDB instance is hosted. `localhost`, in our case.
- <port-number>: Port of the MongoDB server. Generally, it is, by default `27017`. `27017`, in our case.
- <db-name>: Name of the database for which you need to take the dump.
- <collection-name>: Name of the collection for which you need to take the dump. This is optional and if you don't give, `mongodump` will take the dump of all the collections in the database.
- <user>: MongoDB database user, in case of access control.
- <output-path>: The output location, where `mongodump` will dump or save the database backup.

**Starting from version 4.4, "mongodump" is released separately. Earlier, it used to bundle with the MongoDB server and in the old releases of MongoDB, it can be found in the MongoDB bin directory. But now, it is released with the MongoDB tools.**

To take the backup, follow these steps:

1. Navigate to the `bin` directory which is under the `version` directory of the MongoDB tools, which is, `C:\Program Files\MongoDB\Tools\100\bin`, in our case, as shown in the following screenshot:

```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All Rights Reserved.

C:\Users\manus>cd "c:\Program Files\MongoDB\Tools"
C:\Program Files\MongoDB\Tools>cd 100\bin
C:\Program Files\MongoDB\Tools\100\bin>
```

**Figure 17.46: Navigating to MongoDB Tools "bin" Directory**

2. Now, give the following command to take the database backup. Note that we are using the `BPBCatalogDB` database in our example, which is a sample database that contains the catalog of BPB publication house latest books. In our case, we can take this backup at any location. In our system, it is something like `D:\Manu\bpb-db-dump`. You can change the location as well as the database according to your system, as shown in the following screenshot:

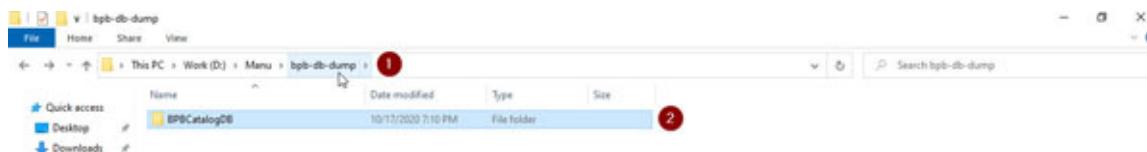
```
mongodump --host=localhost --port=27017 --db=BPBCatalogDB --
out="D:\Manu\bpb-db-dump"
```

As we haven't specified the collection name, in our case, it will take the dump of all the collections in the database.

```
c:\Program Files\MongoDB\Tools\100\bin>mongodump --host=localhost --port=27017 --db=BPBCatalogDB --out="D:\Manu\bpb-db-dump"
2020-10-17T19:10:35.844+0530 writing BPBCatalogDB.BPBCatalogCollection to D:\Manu\bpb-db-dump\BPBCatalogDB\BPBCatalogCollection.bson
2020-10-17T19:10:35.851+0530 done dumping BPBCatalogDB.BPBCatalogCollection (8 documents)
c:\Program Files\MongoDB\Tools\100\bin>
```

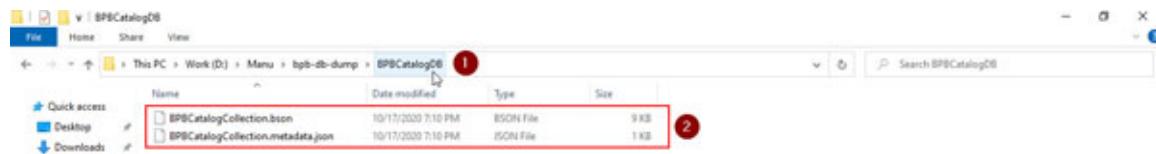
**Figure 17.47: Running "mongodump" Command**

3. We can verify if `mongodump` has successfully created the dump files by browsing the folder's content of our output directory that we have specified during the `mongodump` command with the `--out` option. It is `D:\Manu\bpb-db-dump` in our case. To do this, open this folder on our machine to view its content, as shown in the following screenshot:



**Figure 17.48:** Verifying the "mongodump" Command – Viewing the Output Directory

4. You can now see that `mongodump` has successfully created the folder named as `BPBCatalogDB` inside our output directory, which is the same as our database. We can further open this folder to see its content, in which we will see the files related to the dump (or backups) of all the collections in the database, as shown in the following screenshot:



**Figure 17.49:** Verifying the "mongodump" Command – Viewing the Output Backup Files

## Restoring a MongoDB database using mongorestore

By using the `mongorestore` tool (utility), we can very easily restore the backups of our MongoDB database. We can use `mongorestore` with the following options:

### Simple method

```
mongorestore --host=<hostname> --port=<port-number> --db=<db-name>
<db-dump-path>
```

### With access control

```
mongorestore --host=<hostname> --port=<port-number> --db=<db-name>
--username=<user> --authenticationDatabase=admin <db-dump-path>
```

Where:

- `<hostname>`: Hostname of MongoDB, where the MongoDB instance is hosted. `localhost`, in our case.
- `<port-number>`: Port of the MongoDB server. Generally, it is, by default, `27017`. `27017`, in our case.
- `<db-name>`: Name of the database for which you need to take the dump.

- <user>: MongoDB database user, in case of access control.
- <db-dump-path>: The Location where the MongoDB database dump is present.

Starting from version 4.4, `mongorestore` is released separately. Earlier, it used to bundle with the MongoDB server, and in the old releases of MongoDB, it can be found in the MongoDB `bin` directory. But now, it is released with the MongoDB tools.

To restore the backup, follow these steps:

1. Navigate to the `bin` directory which is under the `version` directory of the MongoDB tools, which is, `C:\Program Files\MongoDB\Tools\100\bin`, in our case, as shown in the following screenshot:



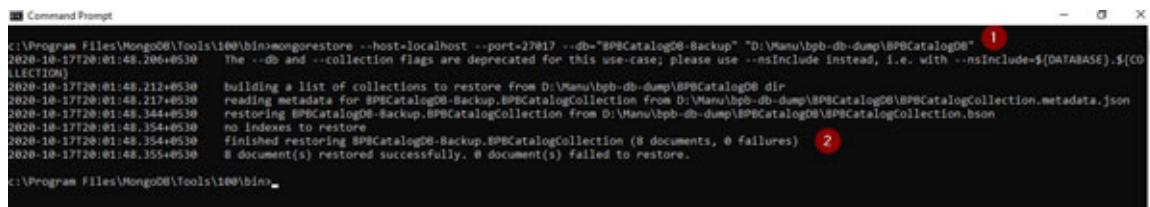
```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All Rights Reserved.

C:\Users\manus>cd "c:\Program Files\MongoDB\Tools"
c:\Program Files\MongoDB\Tools>cd 100\bin ①
c:\Program Files\MongoDB\Tools\100\bin> ②
```

**Figure 17.50: Navigating to MongoDB Tools "bin" Directory**

2. Now, give the following command to restore the database backup. Note that we are using the `BPBCatalogDB-Backup` database in our example, and in our case, we can take this backup from the location on our system, which is something like `D:\Manu\bpb-db-dump`. You can change the location as well as the database according to your system and requirements, as shown in the following screenshot:

```
mongorestore --host=localhost --port=27017 --db=BPBCatalogDB-
Backup "D:\Manu\bpb-db-dump"
```



```
mongorestore --host=localhost --port=27017 --db=BPBCatalogDB-
Backup "D:\Manu\bpb-db-dump" ①
2020-10-17T20:01:48.212+0530  The --db and --collection flags are deprecated for this use-case; please use --nsInclude instead, i.e. with --nsInclude=${DATABASE}.${COLLECTION}
2020-10-17T20:01:48.217+0530  building a list of collections to restore from D:\Manu\bpb-db-dump\BPBCatalogDB dir
2020-10-17T20:01:48.344+0530  reading metadata for BPBCatalogDB-Backup.BPBCatalogCollection from D:\Manu\bpb-db-dump\BPBCatalogDB\BPBCatalogCollection.metadata.json
2020-10-17T20:01:48.354+0530  restoring BPBCatalogDB-Backup.BPBCatalogCollection from D:\Manu\bpb-db-dump\BPBCatalogDB\BPBCatalogCollection.bson
2020-10-17T20:01:48.354+0530  no indexes to restore
2020-10-17T20:01:48.354+0530  finished restoring BPBCatalogDB-Backup.BPBCatalogCollection (8 documents, 0 failures) ②
2020-10-17T20:01:48.355+0530  8 document(s) restored successfully. 0 document(s) failed to restore.
```

**Figure 17.51: Running "mongorestore" Command**

3. To verify if the `mongorestore` command has successfully restored our database, open the MongoDB Compass and check for the

newly created database, which is `BPBCatalogDB-Backup` in our case, and also view the collections and documents inside this database, as shown in the following screenshot:

The screenshot shows the MongoDB Compass interface. On the left, the sidebar lists databases: BPBOnlineDB, admin, config, and local. The main area shows the `BPBCatalogDB-Backup.BPBCatalogCollection`. The collection has 8 documents and 1 index. Document 1 and Document 2 are highlighted with red circles numbered 1 and 2 respectively. Document 3 is highlighted with a red circle numbered 3. The documents are represented as JSON objects:

```

{
    "_id": "ObjectId(\"5f5cbe0fc2900000200f5a1e\")",
    "title": "DataScience For Dummies",
    "authorname": "Robert T. du Prez",
    "isbnnumber": "9780334236515",
    "publicationyear": "2009",
    "price": "220",
    "description": "From Start To Finish, This Book Will Cover All The Intricacies Of The ...",
    "coverimage": "5f5cbe0fc2900000200f5a1c.jpg",
    "thumbCount": 5
}

{
    "_id": "ObjectId(\"5f5cbe0fc2900000200f5a1f\")",
    "title": "Computer Science with Python",
    "authorname": "Anilish Once Gava, Gurpreet Kaur",
    "isbnnumber": "9789386355203",
    "publicationyear": "2020",
    "price": "250",
    "description": "This Book Covers Questions And Answers From The Entire Syllabus Of Com...",
    "coverimage": "5f5cbe0fc2900000200f5a1d.jpg"
}

```

*Figure 17.52: Verifying the "mongorestore" command - Vewing the restored Database and Collection data using MongoDB Compass*

## Import and export with MongoDB

Sometimes, we want to use some methods which allow us to export or import the data into various formats. The MongoDB tools also provide ways by which we can export and import our MongoDB data to and from various formats, like JSON and CSV.

## Exporting a MongoDB data using mongoexport

By using the `mongoexport` tool (utility), we can very easily export our MongoDB data to the formats, like JSON and CSV. We can use `mongoexport` with the following options:

### **Simple method**

```
mongoexport --host=<hostname> --port=<port-number> --db=<db-name>
--collection=<collection-name> --type=<output-type> out=<output-
path>
```

### **With access control**

```
mongoexport --host=<hostname> --port=<port-number> --db=<db-name>
--collection=<collection-name> --username=<user> --
```

```
authenticationDatabase=admin -type=<output-type> --out=<output-path>
```

Where:

- <hostname>: Hostname of MongoDB, where the MongoDB instance is hosted. `localhost`, in our case.
- <port-number>: Port of the MongoDB server. Generally, it is, by default, `27017`. `27017`, in our case.
- <db-name>: Name of the database for which you need to take the dump.
- <collection-name>: Name of the collection for which you need to take the dump. This is optional and if you don't give, then `mongoexport` will take the dump of all the collections in the database.
- <user>: MongoDB database user, in case of access control.
- <output-type>: The output type, JSON or CSV
- <output-path>: The output location where the `mongoexport` will dump or save the database export. You must also specify the filename with the location.

**Note:** that starting from version 4.4, `mongoexport` is released separately. Earlier, it used to bundle with the MongoDB server, and in the old releases of MongoDB, it can be found in the MongoDB bin directory. But now, it is released with the MongoDB tools.

Follow the below steps to learn how we can use `mongoexport`:

1. Navigate to the `bin` directory, which is under the `version` directory of the MongoDB tools, which is `C:\Program Files\MongoDB\Tools\100\bin`, in our case, as shown in the following screenshot:



```
Microsoft Windows [Version 10.0.19041.586]
(C) 2020 Microsoft Corporation. All Rights Reserved.

C:\Users\manus>cd "c:\Program Files\MongoDB\Tools"
C:\Program Files\MongoDB\Tools>cd 100\bin 1
C:\Program Files\MongoDB\Tools\100\bin> 2
```

The screenshot shows a Windows Command Prompt window. The command history at the bottom shows:  
1. `cd "c:\Program Files\MongoDB\Tools"`  
2. `cd 100\bin`  
A red circle labeled '1' is over the second command, and a red circle labeled '2' is over the third command.

**Figure 17.53:** Navigating to MongoDB Tools "bin" Directory

2. Now, give the following command to export the collection. Note that we are using the `BPBCatalogDB` database in our example, which is a sample database that contains the catalog of BPB Publication house latest books, and in our case, we can take the export of this database collection `BPBCatalogCollection` at any location. In our system, it is something like `D:\Manu\bpb-db-export`. You can change the location as well as the database and collection according to your system, as shown in the following screenshot:

```
mongoexport --host=localhost --port=27017 --db=BPBCatalogDB --
--collection=BPBCatalogCollection --type=json --
out="D:\Manu\bpb-db-export\BPBCatalogCollection.json"
```



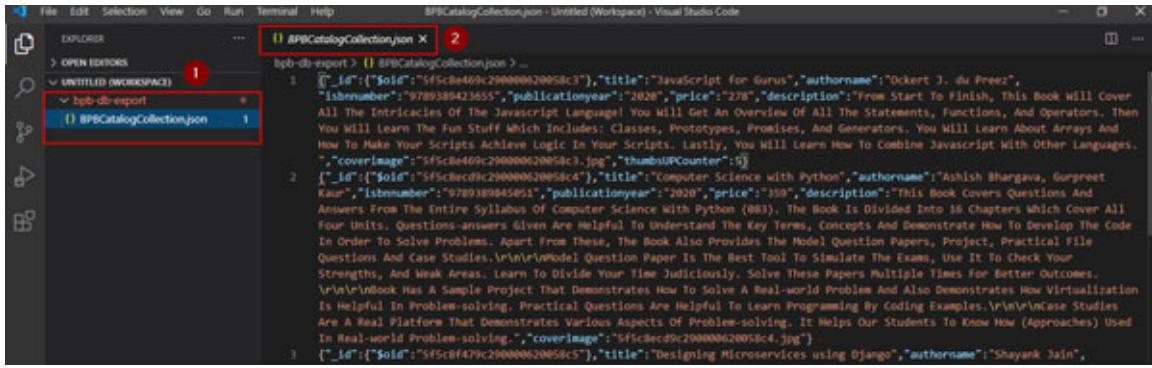
**Figure 17.54:** Running "mongoexport" Command

3. We can verify if `mongoexport` has successfully created the JSON file by browsing the folder's content of our `output` directory that we have specified during the `mongoexport` command with the `--out` option, and it is `D:\Manu\bpb-db-export` (excluding JSON file name), in our case. To do this, open this folder on our machine to view its content, as shown in the following screenshot:



**Figure 17.55:** Verifying the "mongoexport" Command – Viewing the Contents of the Output Folder

4. You can also view the contents of this file easily by opening this file in any text editor or IDE of your choice, such as Microsoft Visual Studio Code, as shown in the following screenshot:



**Figure 17.56:** Opening the JSON File in the Microsoft Visual Studio Code – Viewing the contents of the exported JSON file

## Importing a MongoDB data using mongoimport

By using the `mongoimport` tool (utility), we can very easily import our MongoDB data to the formats, like JSON and CSV. We can use `mongoimport` with the following options:

### Simple method

```
mongoimport --host=<hostname> --port=<port-number> --db=<db-name>
--collection=<collection-name><file-path>
```

### With access control

```
mongoimport --host=<hostname> --port=<port-number> --db=<db-name>
--collection=<collection-name> --username=<user> --
authenticationDatabase =admin -type=<file-type><file-path>
```

Where:

- <hostname>: Hostname of MongoDB, where the MongoDB instance is hosted. `localhost`, in our case.
- <port-number>: Port of the MongoDB server. Generally, it is, by default, `27017`. `27017`, in our case.
- <db-name>: Name of the database for which you need to take the dump.
- <collection-name>: Name of the collection for which you need to take the dump. This is optional and if you don't give, `mongoimport` will take the dump of all the collections in the database.

- <user>: MongoDB database user, in case of access control.
- <file-type>: File type, JSON or CSV.
- <file-path>: The location where `mongoimport` will get the file. You must also specify the filename with the location.

**Note:** that starting from version 4.4, `mongoimport` is released separately. Earlier, it used to bundle with the MongoDB server, and in the old releases of MongoDB, it can be found in the MongoDB bin directory. But now, it is released with the MongoDB tools.

To take the backup follow the following steps:

1. Navigate to the `bin` directory, which is under the `version` directory of the MongoDB tools, which is `C:\Program Files\MongoDB\Tools\100\bin`, in our case as shown in the following screenshot:



```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All Rights Reserved.

C:\Users\manus>cd "c:\Program Files\MongoDB\Tools"
c:\Program Files\MongoDB\Tools>cd 100\bin ①
c:\Program Files\MongoDB\Tools\100\bin> ②
```

**Figure 17.57:** Navigating to MongoDB Tools "bin" Directory

2. Now, give the following command to import the collection. Note that we are using `BPBCatalogDB-Import` database in our example and `BPBCatalogCollection`. Our import file path is `D:\Manu\bpb-db-export\BPBCatalogCollection.json`, which contains the JSON file that we have earlier exported using `mongoexport`. You can change the location as well as the database and collection according to your system, as shown in the following screenshot:

```
mongoimport --host=localhost --port=27017 --db=BPBCatalogDB-Import --collection=BPBCatalogCollection --type=json
"D:\Manu\bpb-db-export\BPBCatalogCollection.json"
```



```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All Rights Reserved.

c:\Program Files\MongoDB\Tools\100\bin>mongoimport --host=localhost --port=27017 --db=BPBCatalogDB-Import --collection=BPBCatalogCollection --type=json "D:\Manu\bpb-db-export\BPBCatalogCollection.json"
2020-10-18T13:58:39.438+05:30    connected to: mongodb://localhost:27017/
2020-10-18T13:58:39.722+05:30    0 document(s) imported successfully. 0 document(s) failed to import.

c:\Program Files\MongoDB\Tools\100\bin>
```

**Figure 17.58:** Running "mongoimport" Command

3. To verify if the `mongoimport` command has successfully imported our database and collection, we can open the MongoDB Compass and check for the newly created database and collection, which is `BPBCatalogDB-Import`, and we can see the collection and its documents, as shown in the following screenshot:

_id	title	authorname	isbnnumber	publicationyear	price	description	coverimage	thumbnailcounter
<code>5e03e5c1021f53d4ec2300006200f5a1c1</code>	"Starting From Scratch"	"Robert J. du Prez"	"9780303942365"	"2009"	"29"	"From Start To Finish, This Book Will Cover All The Intricacies Of The ..."	<code>5e03e5c1021f53d4ec2300006200f5a1c1.jpg</code>	5
<code>5e03e5c1021f53d4ec2300006200f5a1c4</code>	"Computer Science with Python"	"Ashish Waragave, Surpreet Kaur"	"9780303946665"	"2009"	"79"	"This Book Covers Questions And Answers From The Entire Syllabus Of Com..."	<code>5e03e5c1021f53d4ec2300006200f5a1c4.jpg</code>	5

**Figure 17.59:** Verifying the "mongoimport" command - Viewing the restored Collection data using MongoDB Compass

We have seen how we can use various MongoDB tools to take and restore backups with the help of `mongodump`, `mongorestore`, `mongoexport` and `mongoimport`.

## MongoDB security

Information and data security is very important for any organization and every organization must have good security policies and practices in place, which also include regular security related audits. There are many factors that we can consider to make our database secure. In this section, we will study some important points that any database administrator should be aware of and plan according to their corporate or company environments. This section doesn't include any practical demonstration as it is not possible to practically cover all these points in a single chapter and it is out of the scope of this book. But we will study some important points that we can take care of during the production level implementation of MongoDB in our organization.

## Enable authentication

MongoDB provides the authentication methods so that when any client tries to connect to the MongoDB database, it should authenticate itself using the valid username and password.

If the authentication is enabled in MongoDB, we have to use the mongo command-line authentication options, such as `--username`, `--password`, and `--authenticationDatabase`, while connecting to `mongod` or `mongos` (in case of sharding, where the sharded clusters are used).

## **Use role-based authorization ([role-based access control](#))**

It is always recommended to have a role-based access control in place so that each person has limited access to the data. Every organization must have a strong role-based policy in place through the departments and organizational structure so that a person can only be able to see the data which is intended to be seen or modified by him/her.

MongoDB provides *in-build* as well as user defined roles and we can use the MongoDB "*User Management*" and "*Role Management*" commands, such as `createUser` and `createRole`, to create the user and allocate the right type of roles to a particular user and group.

Take a note that "*not every user is a super user or administrator*". So, we should always take care of giving limited access to the users according to their job profiles. This will make sure that our data is always protected and the sensitive information of an organization is not available to anyone who is not authorized for that data or information.

## **Encrypt the communication channels and connections**

Encryption plays a very important role in protecting the data from the evil hands. Make sure your connections are secure and encrypted. It is recommended to configure your MongoDB to use TLS (Transport Layer Security)/SSL (Secure Sockets Layer).

If we have TLS/SSL in place, then any communication between MongoDB and the applications which are connected to it always flows in a secure and encrypted channel.

## Encrypt your MongoDB data

Like the network or communication channel encryption, data encryption is also very important. If your data is encrypted, it is very difficult to decode it, even if it reaches into the wrong hands. Starting from version 3.2, MongoDB provides the method to encrypt the MongoDB data with the **WiredTiger** storage engine's native "**Encryption at Rest**".

We should always make sure that all the MongoDB data, which includes data files, logs files, configuration files, audit files, or any other key files, are encrypted at the storage level.

## Use firewalls and restrict all the incoming and outgoing traffic

Make sure that your network, where your MongoDB server resides, is properly protected with firewalls, and only the applications that are allowed to use the MongoDB database or instance has the permission, and no other applications can access the MongoDB instances.

## Regularly perform security audits

As an organization or network administrator, you should regularly perform the network and security audits. At every regular interval, make sure your MongoDB database has the correct roles and users and correct application servers who has the right to access your MongoDB instance. If you find any breach, first restrict the access, and then take the appropriate organization level action.

## Conclusion

In this chapter, we covered the advanced administration topics of MongoDB which is very helpful to the people who will perform various MongoDB administrative tasks. In this chapter, we learned about the mongod process and how to manage it. We also learned how to monitor and diagnose MongoDB. We covered the step-by-step method to install the MongoDB tools in our machine and how to use these MongoDB tools and various other MongoDB commands to monitor and diagnose MongoDB. Later in this chapter, we learned how to take MongoDB backups and how to restore these backups. We also covered

how to perform the export and import of the MongoDB data. In the last part of this chapter, we covered various important points related to MongoDB security which we should take care of, so that our MongoDB database and its data are secured.

## **Questions**

1. What do you understand by `mongod`? Explain in detail.
2. What is the daemon or background process?
3. How can we start MongoDB from the command line?
4. Why should we monitor our MongoDB instance at regular intervals? What are the benefits of doing so?
5. List any seven MongoDB tools and commands helpful for monitoring and diagnosing MongoDB?
6. How can you take MongoDB backups?
7. What are the two main formats we can use during `mongoexport`?
8. List three ways to make your MongoDB instance secure?

## CHAPTER 18

# Replication in MongoDB

This chapter covers the replication part of MongoDB. In this chapter, we will learn about the replication and replica set in a quick recap. We will also learn about the MongoDB heartbeats and how the heartbeats play an important role in the replicated environment. We will also learn how the election of the new primary member takes place. Later in this chapter, we will cover the pre-configuration steps before we start with the practical step-by-step method to create the replicated environment with the MongoDB primary and secondary instances. We will then learn how to setup the replicated MongoDB environment with the step-by-step method. In the last part of this chapter, we will learn how to verify the replication setup if it has been configured correctly with the help of the data.

### Structure

In this chapter, we will discuss the following topics:

- Basic introduction to replication – quick recap
  - Basic introduction to the MongoDB replication
  - Replica sets
- MongoDB heartbeats
- Election of the new primary member
- Pre-configuration steps for replication
- Starting with the MongoDB replication on Windows machine
- Verifying the MongoDB replication using data

### Objectives

After studying this unit, you should be able to get an introduction to the MongoDB replication in a quick recap and also get introduced to the

replica sets. You should be able to understand MongoDB heartbeats and how the election of the new primary member takes place. You should be able to learn the pre-configuration steps before you start with the practical step-by-step configuration process. You should also be able to learn and understand how to perform the MongoDB replication on Windows machine and also understand how to verify the MongoDB replication using data.

## **Basic introduction to replication – quick recap**

Let us quickly recap the replication process. We have already covered the basic introduction in our previous chapter. In this chapter, we will learn how we can practically do the replication on our machine using MongoDB.

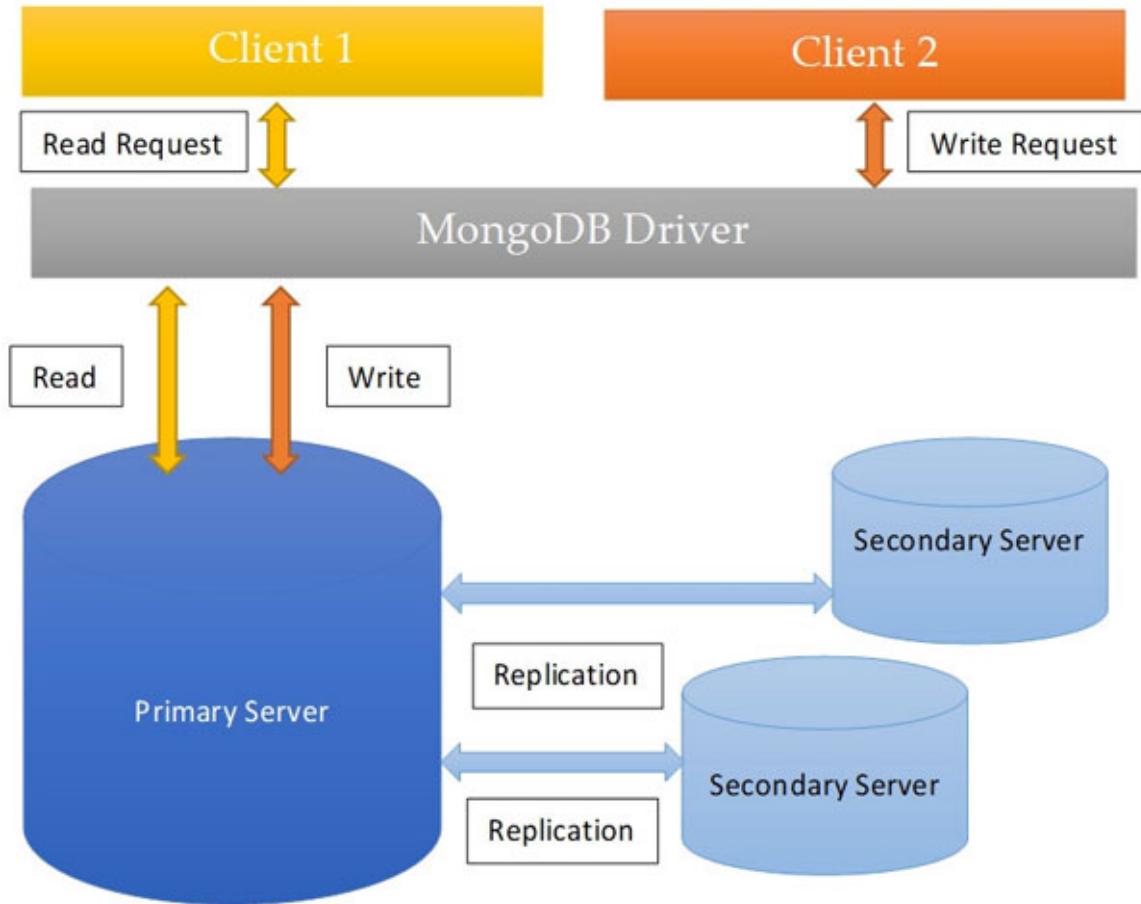
Replication is a process of duplicating the same set of databases so that we have multiple copies of the same data available to us on different servers because of which we have redundancy and high availability of data.

Following are some benefits of replication:

- Replication helps in the fail-over recovery, in case one server fails, or the hardware fails, or in case of any other service interruptions.
- Replication also helps in the increased performance because the clients are able to send their requests to different servers.
- Replication helps in delivering the data to the distributed applications from various data center locations across the globe.

## **Replica sets**

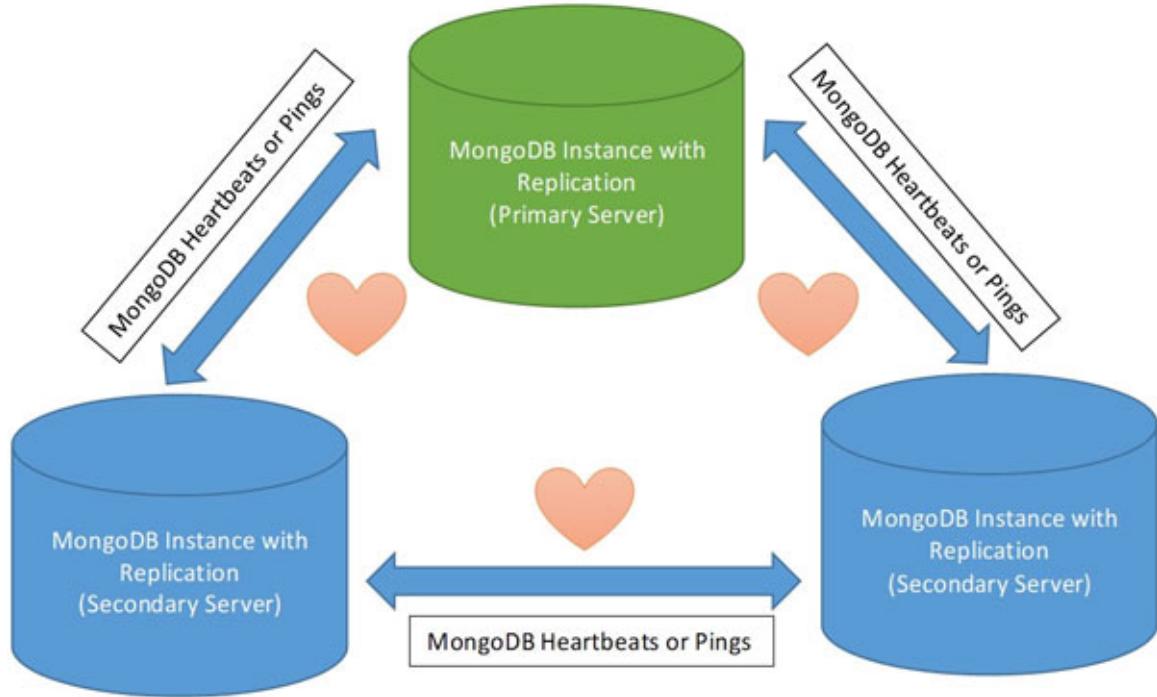
Replica sets are the groups of MongoDB processes or instances which host the same data set. There is one primary and many secondary nodes that form a group to create replica sets, as shown in the following figure:



*Figure 18.1: Replication in MongoDB*

## MongoDB heartbeats

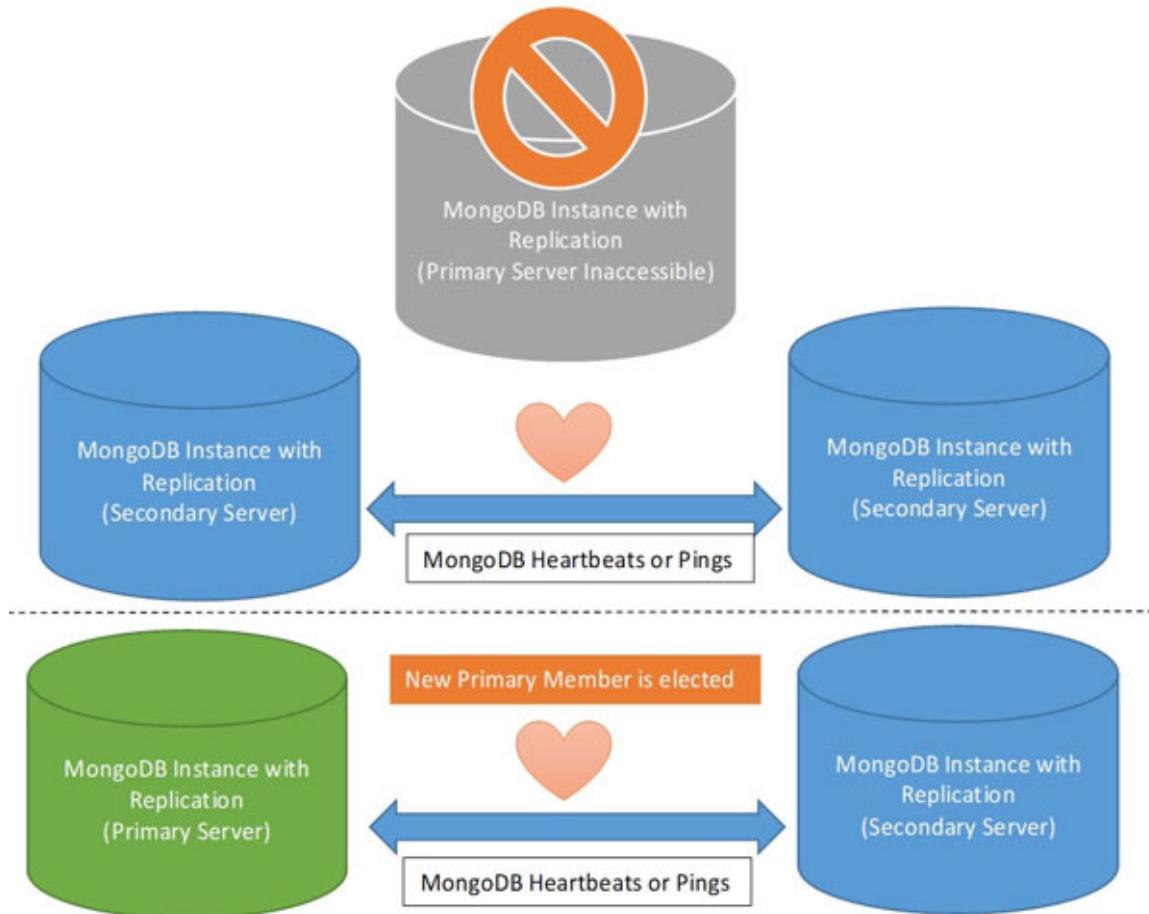
In the MongoDB replication environment, the replica group members send the "*Heartbeats*" or pings every 2 seconds to communicate with each other. This ensures that the replication is actually happening between the servers, as shown in the following figure: If any of the member does not respond to the others within 10 seconds, in that case, it is treated as inaccessible.



*Figure 18.2: MongoDB Heartbeats*

## Automatic election of the new primary member

In case there is no communication taking place between the primary server and the secondary server, and when the primary server is not able to send heartbeats or pings for more than 10 seconds to the secondary server in the replica sets, the election of a new primary automatically takes place between the secondary server and the new primary is elected among the secondary members of the replica sets.



*Figure 18.3: MongoDB Election of a new Primary*

## Pre-configuration steps

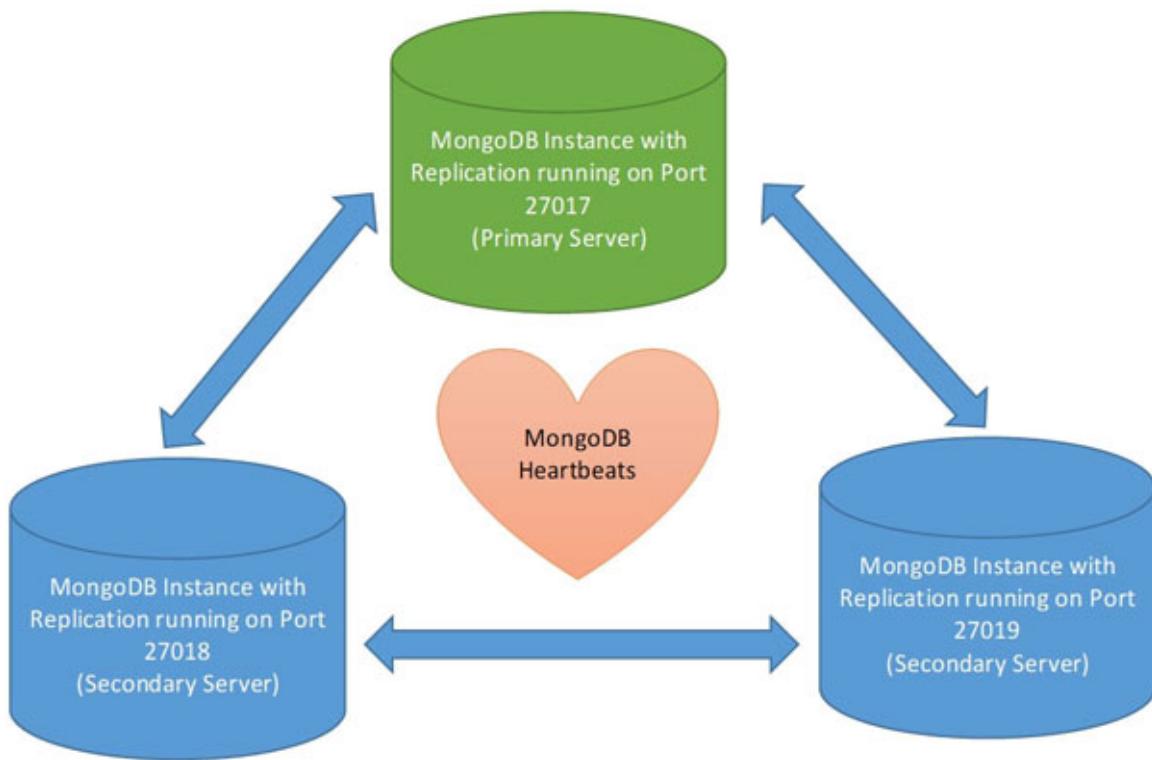
In order to perform the replication on our machine, let us first understand the steps we need to take to correctly setup a replication environment on our machine using MongoDB.

To start the replication process, we need to follow these steps:

1. We will use Windows Operating System that already has MongoDB installed in it
2. We will stop the existing MongoDB service on our Windows machine
3. We will create a replica set of 3 MongoDB nodes that will form a group of replica sets
4. We will create one MongoDB instance which will act like a primary server or a node of the replica set

5. We will create two MongoDB instances which will act like secondary servers or nodes of the replica set
6. So, basically, we will form a group of replica sets having one primary and two secondary servers
7. We will create a new database and collection and will then confirm that the replication is taking place on these two secondary servers and we can view this database and collection on our secondary servers too.

The diagram of our MongoDB replica sets group on our Windows machine is shown in the following figure:

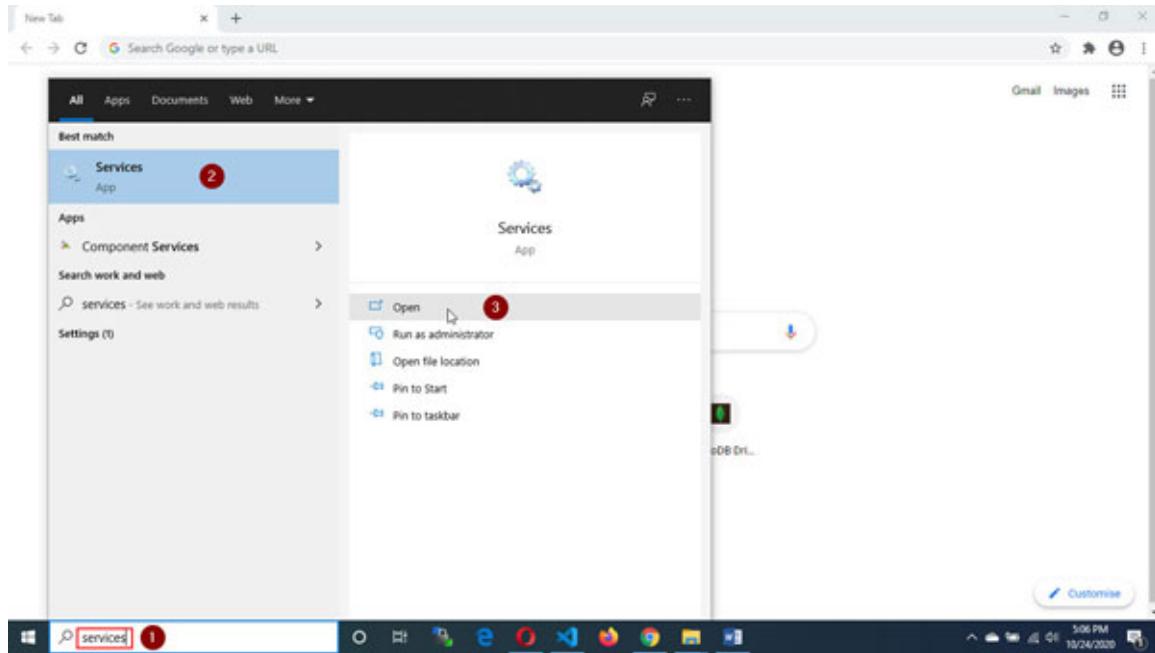


*Figure 18.4: MongoDB Replication on Windows*

## Starting with the MongoDB replication on Windows machine

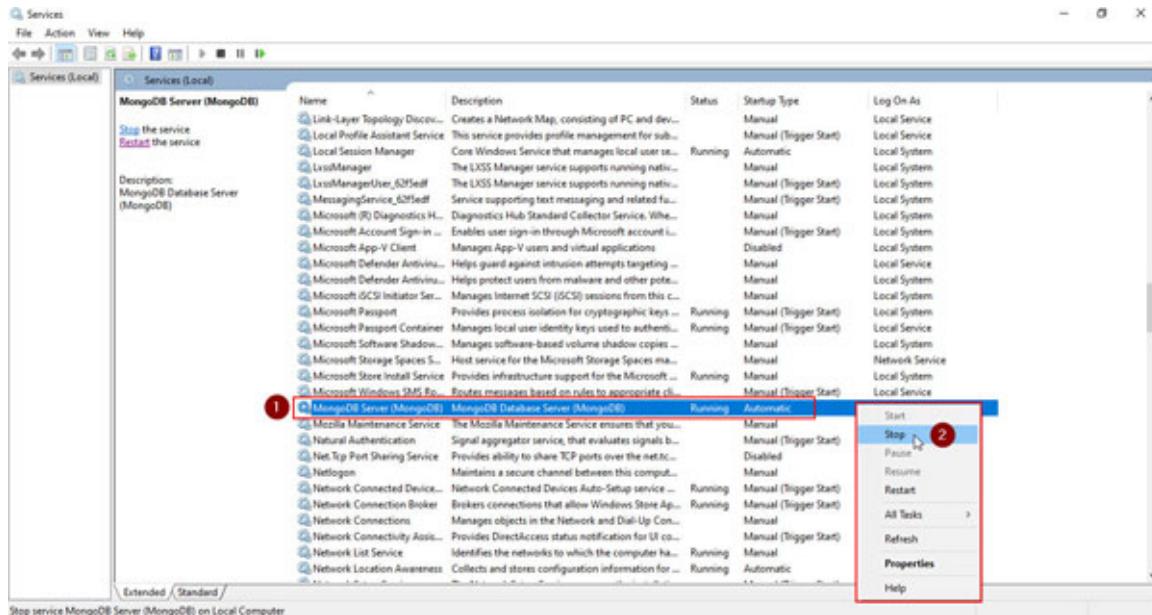
The step-by-step process to do the replication of MongoDB is as follows:

1. In the search box of your Windows, type `services` and open the Windows Service Manager, as shown in the the following figure:



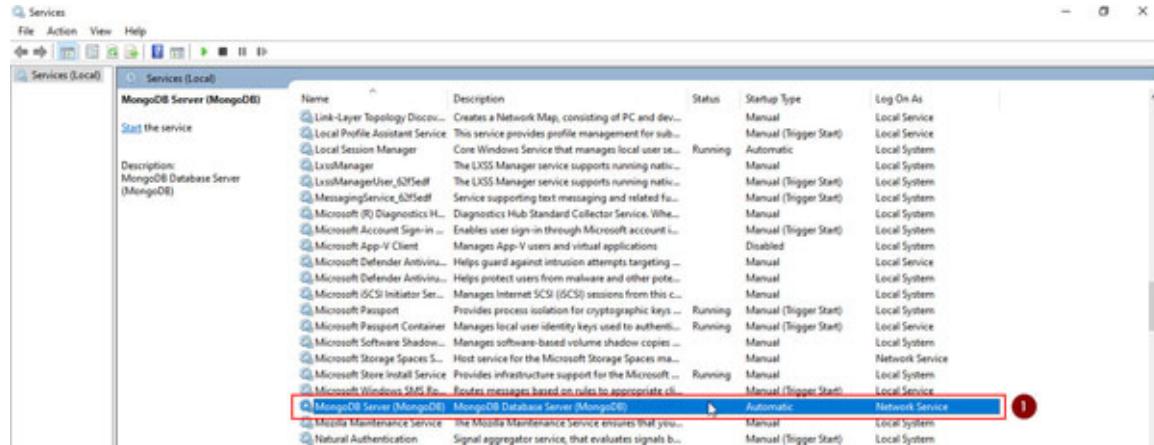
**Figure 18.5: Open Windows Services Manager**

2. Now, in the Windows Service Manager, navigate to the MongoDB service and right click on the service. This will open up a service-related menu which has the `stop` option. Click this option to stop the MongoDB service, as shown in the the following figure:



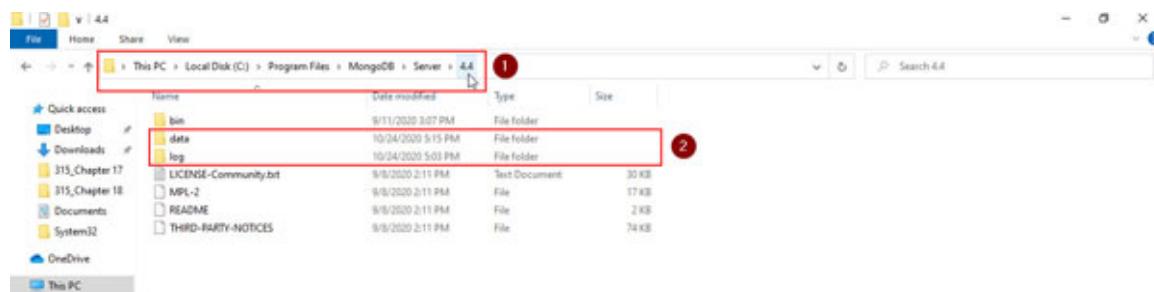
**Figure 18.6: Windows Service Manager – Stop MongoDB Service**

3. Now, as the MongoDB service will stop, you will see that the running status is blank. Now you can close the Windows Service Manager, as shown in the following figure:



**Figure 18.7: Windows Service Manager – MongoDB Service has been Stopped**

4. Now, open the location of the folder where you have installed MongoDB. In our case, it is, C:\Program Files\MongoDB\Server\4.4. You will see the data and log directories under this folder location, as shown in the following figure:



**Figure 18.8: MongoDB Installation Folder Location**

5. We need to create a few folders under both the data and the log directories as follows:

Under the data directory (or folder), we need to create three new folders which will contain the MongoDB data files:

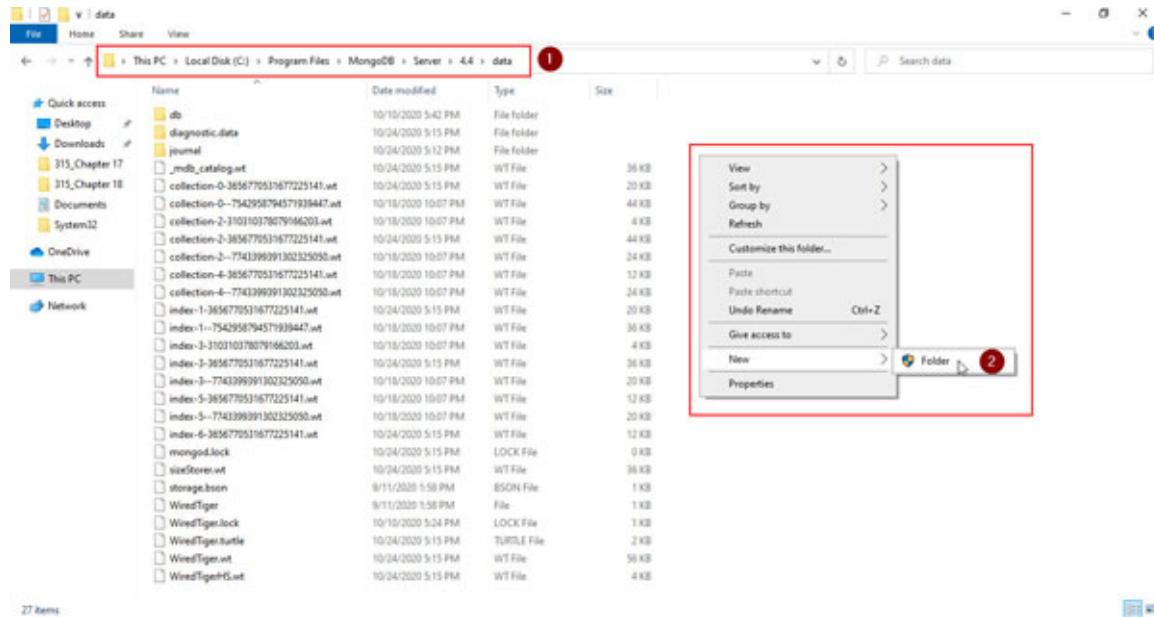
- C:\Program Files\MongoDB\Server\4.4\data\BPBOnlineDBPS  
(Data Directory for our Primary Server or Instance)
- C:\Program Files\MongoDB\Server\4.4\data\BPBOnlineDBSS1  
(Data Directory for our Secondary Server 1)

- C:\Program Files\MongoDB\Server\4.4\data\BPBOnlineDBSS2  
(Data Directory for our Secondary Server 2)

Under the log directory (or folder), we need to create three new folders which will contain the MongoDB log files:

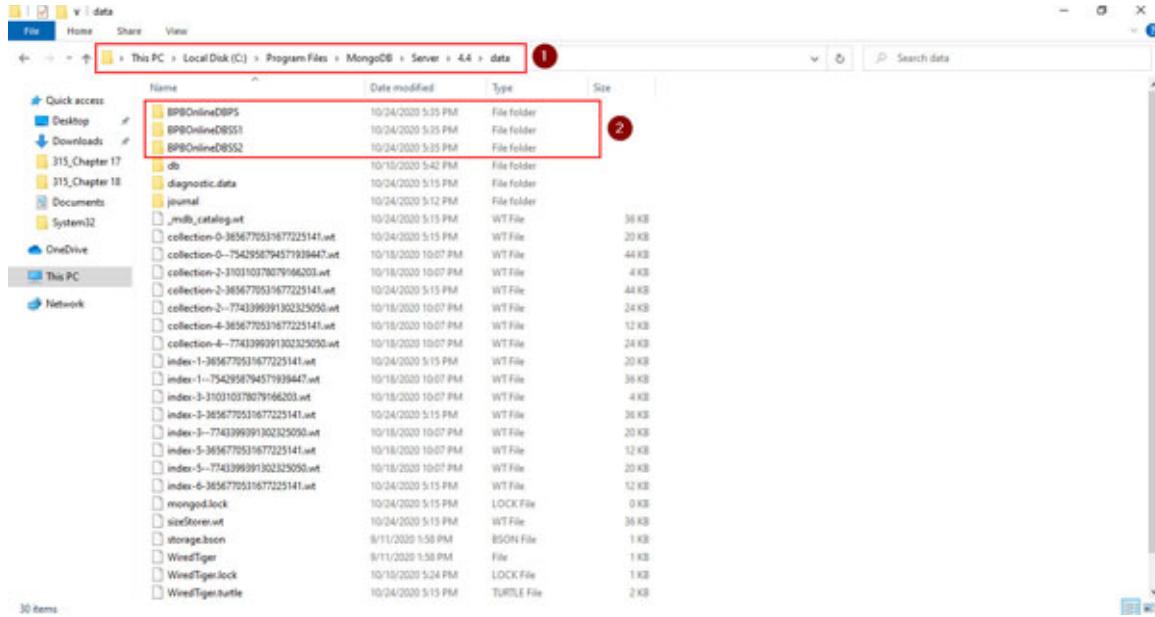
- C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDBPS  
(Log Directory for our Primary Server or Instance)
- C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDBSS1  
(Log Directory for our Secondary Server 1)
- C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDBSS2  
(Log Directory for our Secondary Server 2)

6. Let us first create new folders under the data directory (or folder), as shown in the the following figure:



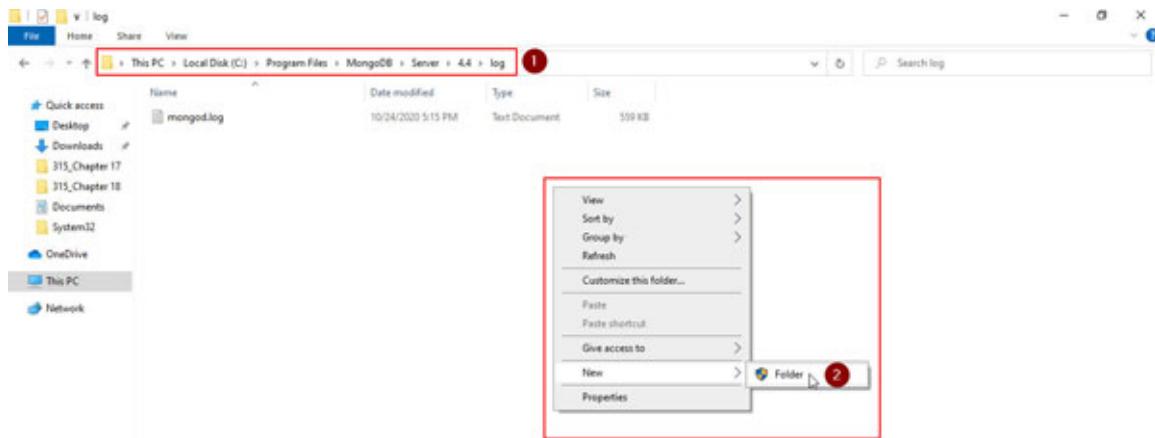
**Figure 18.9: MongoDB Installation "data" Directory Location – Creating new Folders**

7. After you have created new folders under the "data" directory (or folder), these should look similar to the screenshot, as shown in the the following figure:



**Figure 18.10: MongoDB Installation Folder Location**

- Let us now create new folders under the `log` directory (or folder), as shown in the the following figure:



**Figure 18.11: MongoDB Installation "log" Directory Location – Creating new Folders**

- After you create new folders under the `log` directory (or folder), these should look similar to the screenshot, as shown in the the following figure:



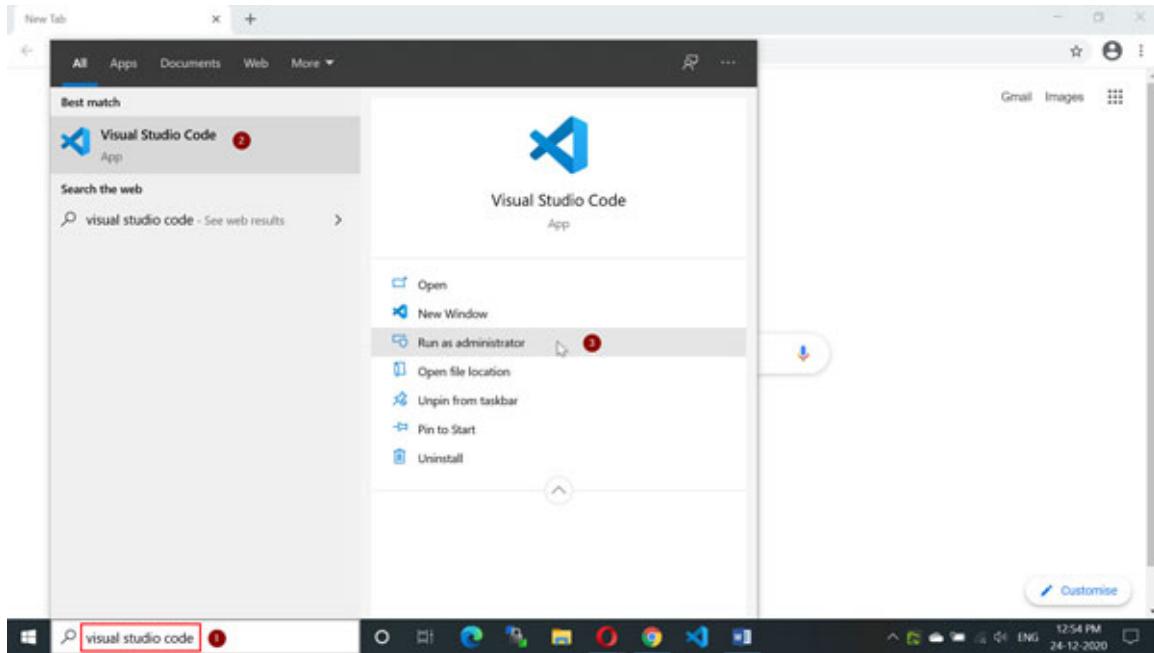
**Figure 18.12: MongoDB Installation Folder Location**

10. Now, we need to create few files with the name `mongod.log` under each of our newly created folders under the parent directory `log`, as follows:

- C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDBPS\mongod.log  
(Log file for our primary server or instance)
- C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDBSS1\mongod.log  
(Log file for our secondary server 1)
- C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDBSS2\mongod.log  
(Log file for our secondary server 2)

To do this, we need to open any text editor like windows Notepad in the administrative mode. Or we can also use IDE like Microsoft Visual Studio Code in the administrative mode. In this step, we will use Visual Studio Code in the administrative mode to create three blank files with the name `mongod.log` in the location mentioned earlier.

11. Open Visual Studio Code in the administrative mode, as shown in the following figure:



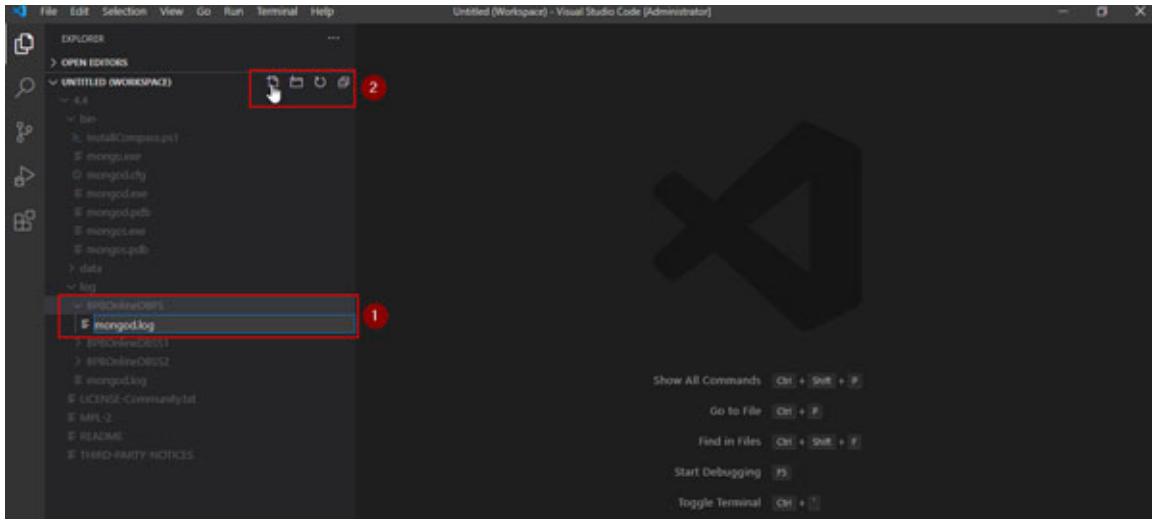
**Figure 18.13:** Opening Visual Studio Code with Administrative Mode

12. Open the location of your MongoDB installation path in the Visual Studio from the `Add Folder` button, as shown in the following figure:



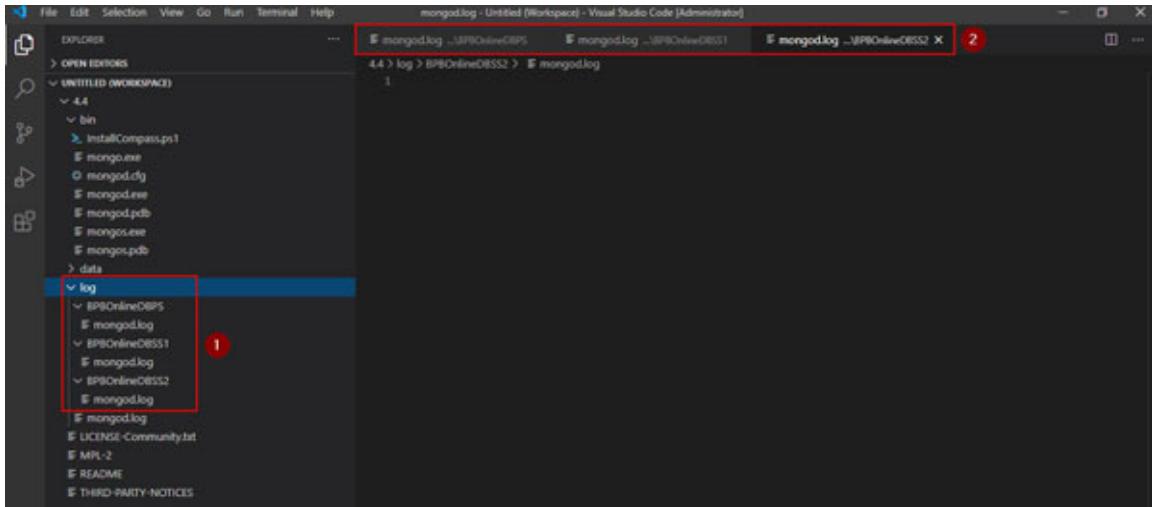
**Figure 18.14:** Opening the Folder Location of your MongoDB Installation in Visual Studio Code

13. Now you can create the new files under the sub-folders of `log` created in the previous steps. You need create a new blank file named `mongod.log` under each of these folders, as shown in the following figure:



**Figure 18.15:** Creating the "mongod.log" Files under Sub-Folders of "log"

14. After you have successfully created these three blank mongod.log files in each of these folders, you will see something similar the screenshot, as shown in the following figure. You should close the Visual Studio Code after creating the blank mongod.log files.



**Figure 18.16:** Creating the "mongod.log" Files under Sub-Folders of "log"

15. Now we will start our first server with the help of `mongod` command with the few parameters as follows:

```
mongod --dbpath "C:\Program  
Files\MongoDB\Server\4.4\data\BPBOnlineDBPS" --logpath  
"C:\Program  
Files\MongoDB\Server\4.4\log\BPBOnlineDBPS\mongod.log" --port
```

```
27017 --storageEngine wiredTiger --journal --replicaSet  
BPBOnlineReplicaSet
```

This command needs to be run from the command prompt opened in the administrator mode. Here, the parameters given in the `mongod` command are described as follows:

- `--dbpath`: Path of our DB directory location
- `--logpath`: Path of our DB log file location
- `--port`: For primary server, we will use 27017
- `--storageEngine`: We will use "wiredTiger"
- `--journal`: This will enable the MongoDB journaling
- `--replicaSet`: In our case, we will use `BPBOnlineReplicaSet`, which is common for all the three member nodes or instances in the replica set.

16. Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is, `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following figure:



**Figure 18.17: Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory**

17. Type the following command with all the parameters that will start our first server in the replica set as primary server as follows:

```
mongod --dbpath "C:\Program  
Files\MongoDB\Server\4.4\data\BPBOnlineDBPS" --logpath  
"C:\Program  
Files\MongoDB\Server\4.4\log\BPBOnlineDBPS\mongod.log" --port  
27017 --storageEngine wiredTiger --journal --replicaSet  
BPBOnlineReplicaSet
```

Once you enter this command, you will get a response something similar to the screenshot, as shown in the following figure:

The screenshot shows an Administrator Command Prompt window. The command entered is:

```
C:\Windows\system32>cd "C:\Program Files\MongoDB\Server\4.4\bin"
C:\Program Files\MongoDB\Server\4.4\bin>mongod --dbpath "C:\Program Files\MongoDB\Server\4.4\data\BPBOnlineDBPS" --logpath "C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDBPS\mongod.log" --port 27017 --storageEngine wiredTiger
```

Annotations:

- Annotation 1: A red circle highlights the log path parameter: `--logpath "C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDBPS\mongod.log"`.
- Annotation 2: A red circle highlights the journal parameter: `--journal --replSet BPBOnlineReplicaSet`.
- A red box highlights the text: **Response from the "mongod" command**.

**Figure 18.18: Starting the "mongod" with Replication parameters**

- Now open another command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is, `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, and then type the following command:

```
mongo --host localhost --port 27017
```

This command will connect us to the MongoDB Shell, as shown in the following figure.

Note that you should not close the another command prompt. Keep it open so that the `mongod` instance will keep on running on the port 27017:

The screenshot shows an Administrator Command Prompt window connected to the MongoDB Shell. The command entered is:

```
C:\Windows\system32>cd "C:\Program Files\MongoDB\Server\4.4\bin"
C:\Program Files\MongoDB\Server\4.4\bin>mongo --host localhost --port 27017
```

Annotations:

- Annotation 1: A red circle highlights the connection parameters: `--host localhost --port 27017`.
- Annotation 2: A red circle highlights the session ID: `{"id": "a6c2f21b-2991-4c1c-8e9e-a87415d4ef74"}`.

The MongoDB Shell output includes startup warnings and information about monitoring:

```
connecting to: mongodb://localhost:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("a6c2f21b-2991-4c1c-8e9e-a87415d4ef74") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-10-24T18:40:06.375+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-10-24T18:40:06.375+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the 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()
```

**Figure 18.19: Connected to the MongoDB Shell**

- After we get connected to the MongoDB Shell, try to run any MongoDB Shell command like the following:

```
show dbs
```

You will see that the MongoDB Shell will return the error, as shown in the following figure. At this point, we haven't initiated the replica

set and we need to give some MongoDB Shell commands to initiate the replication process:

**Figure 18.20:** "show dbs" returned error in MongoDB Shell

20. Now, we need to initiate the replication process and for this we need to add our first member which will act as the primary member in the replica set. To do this, first create a variable with the details of the members of the replica set. In our case, we will create a variable with the name `replicaSetConfig` having the `_id` same as of our replica set, which is `BPBOnlineReplicaSet` and the members details will have the `_id` as 0 (for the first member in the replica set) and their host value which is the host and port values and it is `localhost:2701`, in our case:

```
replicaSetConfig = {
  _id: "BPBOnlineReplicaSet",
  members: [
    {
      _id: 0,
      host: "localhost:27017"
    }
  ]
}
```

Enter the preceding code in the MongoDB Shell and you will get the result similar to the screenshot, as shown in the following figure:

```
Administrator Command Prompt - mongo --host localhost --port 27017
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
...
> show dbs
uncaught exception: Error: listDatabases failed:[
  "topologyVersion" : {
    "processId" : ObjectId("5f9427ac313c29434a1536fa"),
    "counter" : NumberLong(0)
  },
  "operationTime" : Timestamp(0, 0),
  "ok" : 0,
  "errmsg" : "not master and slaveOk=false",
  "code" : 13435,
  "codeName" : "NotMasterNoSlaveOk",
  "$clusterTime" : {
    "clusterTime" : Timestamp(0, 0),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
]
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:147:19
Mongo.prototype.getDB@src/mongo/shell/mongo.js:99:12
shellHelper.show@src/mongo/shell/utils.js:937:13
shellHelper@src/mongo/shell/utils.js:819:15
$([shellHelp2]):i;
> replicasetConfig = [
...   "_id": "BPPOnlineReplicaSet",
...   "members": [
...     {
...       "_id": 0,
...       "host": "localhost:27017"
...     }
...   ]
... ]
{
  "_id": "BPPOnlineReplicaSet",
  "members": [
    {
      "_id": 0,
      "host": "localhost:27017" ②
    }
  ]
}
```

**Figure 18.21:** "show dbs" returned error in MongoDB Shell

21. Now, we can use this variable `replicaSetConfig` during the initialization command for the replica set creation. Type the following command to initiate the replication process:

```
rs.initiate(replicaSetConfig)
```

Enter the preceding command in the MongoDB Shell and you will get the result similar to the screenshot, as shown in the following figure:

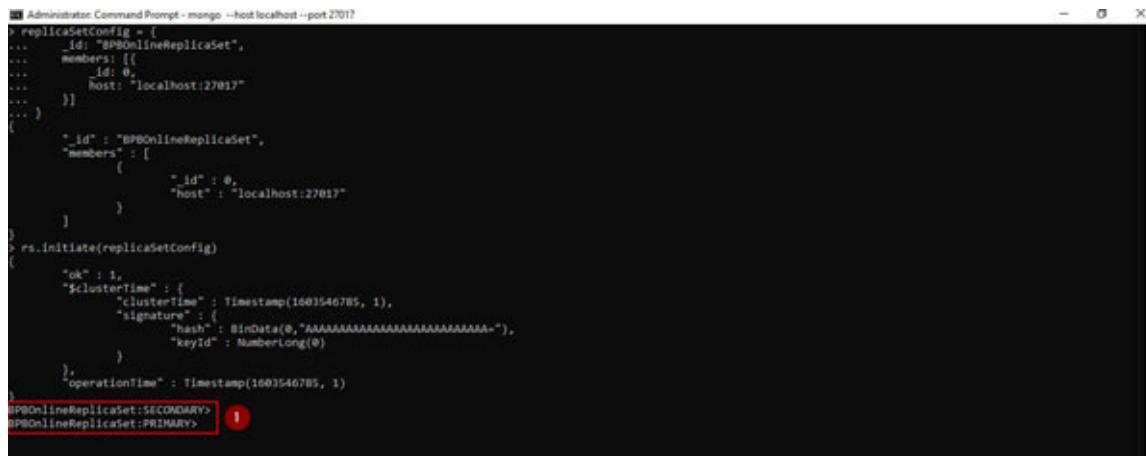
```
Administrator: Command Prompt - mongo --host localhost --port 27017
{
  "codeName": "NoMasterNoSlaveOk",
  "$clusterTime": {
    "clusterTime": Timestamp(0, 0),
    "signature": {
      "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAA"),
      "keyId": NumberLong(0)
    }
  }
}
getErrronWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs:@src/mongo/shell/mongo.js:147:19
Mongo.prototype.getDBs:@src/mongo/shell/mongo.js:99:12
shellHelper.show:@src/mongo/shell/utils.js:937:13
shellHelper:@src/mongo/shell/utils.js:819:15
$[shellHelp2]:i:1
> replicaSetConfig = {
...   "_id": "BPBOnlineReplicaSet",
...   "members": [
...     {
...       "_id": 0,
...       "host": "localhost:27017"
...     }
...   ]
...
  "_id": "BPBOnlineReplicaSet",
  "members": [
    {
      "_id": 0,
      "host": "localhost:27017"
    }
  ]
}
> rs.initiate(replicaSetConfig) ①
{
  "ok": 1, ②
  "$clusterTime": {
    "clusterTime": Timestamp(1603546785, 1),
    "signature": {
      "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAA"),
      "keyId": NumberLong(0)
    }
  },
  "operationTime": Timestamp(1603546785, 1)
}
BPBOnlineReplicaSet:SECONDARY
```

**Figure 18.22: Initializing the Replication with "rs.initiate()" Command in MongoDB Shell**

22. If you see carefully in our MongoDB Shell, you will find that our MongoDB prompt has changed to

BPBOnlineReplicaSet:SECONDARY>:

Just press the *Enter* key without typing anything and this will change the prompt to BPBOnlineReplicaSet:PRIMARY> and you will get the result similar to the screenshot, as shown in the following figure:



```
Administrator: Command Prompt - mongo --hostlocalhost --port27017
> rs.initiate(replicaSetConfig)
{
  "_id": "BPBOnlineReplicaSet",
  ...
  "members": [
    {
      "_id": 0,
      "host": "localhost:27017"
    }
  ]
}
> rs.initiate(replicaSetConfig)
{
  "ok": 1,
  "$clusterTime": {
    "clusterTime": Timestamp(1603546785, 1),
    "signature": {
      "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId": NumberLong(0)
    }
  },
  "operationTime": Timestamp(1603546785, 1)
}
BPBOnlineReplicaSet:SECONDARY>
BPBOnlineReplicaSet:PRIMARY>
```

**Figure 18.23: Changing the MongoDB Shell from SECONDARY to PRIMARY with a Enter key**

23. We can now verify the status of our replica set using the following command:

`rs.status()`

Just enter the aforementioned command in your MongoDB Shell of the primary instance and you will get the result similar to the following screenshot. You will get the information about the replica set members and the status, which is `OK` in our case, as shown in the following figure:

```

Administrator: Command Prompt - mongo --host localhost --port 27017
{
    "electionTerm" : NumberLong(1),
    "lastCommittedOpTime" : {
        "ts" : Timestamp(0, 0),
        "t" : NumberLong(-1)
    },
    "lastSeenOptimeElection" : {
        "ts" : Timestamp(1603546785, 1),
        "t" : NumberLong(-1)
    },
    "numVotesNeeded" : 1,
    "priorityAtElection" : 1,
    "electionTimeoutMillis" : NumberLong(10000),
    "newTermStartDate" : ISODate("2020-10-24T13:39:45.895Z"),
    "newMajorityWriteAvailabilityDate" : ISODate("2020-10-24T13:39:46.072Z")
},
"members" : [
    {
        "_id" : 0,
        "name" : "localhost:27017",
        "health" : 1,
        "state" : 1,
        "stateStr" : "PRIMARY",
        "uptime" : 2584,
        "optime" : {
            "ts" : Timestamp(1603547506, 1),
            "t" : NumberLong(1)
        },
        "optimeDate" : ISODate("2020-10-24T13:51:46Z"),
        "syncSourceHost" : "",
        "syncSourceId" : 1,
        "infoMessage" : "",
        "electionTime" : Timestamp(1603546785, 2),
        "electionDate" : ISODate("2020-10-24T13:39:45Z"),
        "configVersion" : 1,
        "configTerm" : 1,
        "self" : true,
        "lastHeartbeatMessage" : ""
    }
],
"ok" : 1,
"clusterTime" : {
    "clusterTime" : Timestamp(1603547506, 1),
    "signature" : {
        "hash" : BinData(0, "AAAAAAAAAAAAAAAAAAAAAA="),
        "keyId" : NumberLong(0)
    }
}
}

```

**Figure 18.24:** "rs.status()" – Checking the status of the Replica Set

24. Now we can add new members to this replica set. But before we do so, we need to start the secondary members (these members will run the MongoDB instances in different ports) and add them after these secondary instances will start. To do this, we should start our first secondary member with the help of the `mongod` command similar to what we have used at the beginning of our primary member of the replica set.

To do this, open another command prompt in the administrative mode without closing any other command prompts that are already open. Navigate to the `bin` directory of the MongoDB installation and type the following command, as shown in the following figure:

```

mongod --dbpath "C:\Program
Files\MongoDB\Server\4.4\data\BPBOnlineDBSS1" --logpath
"C:\Program
Files\MongoDB\Server\4.4\log\BPBOnlineDBSS1\mongod.log" --
port 27018 -storageEngine wiredTiger --journal --repSet
BPBOnlineReplicaSet

```

Here, the parameters given in the `mongod` command are described as follows:

- `--dbpath`: Path of our DB directory location

- **--logpath:** Path of our DB log file location
- **--port:** For our first secondary server, we will use 27018
- **--storageEngine:** We will use "wiredTiger"
- **--journal:** This will enable the MongoDB journaling
- **--replSet:** In our case, we will use BPBOnlineReplicaSet, which is common for all the three member nodes or instances in the replica set.

```

Administrator: Command Prompt - mongod --dbpath "C:\Program Files\MongoDB\Server\4.4\data\BPBOnlineDB551" --logpath "C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDB551\mongod.log" --port 27018 --store...
Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd "c:\Program Files\MongoDB\Server\4.4\bin"
c:\Program Files\MongoDB\Server\4.4\bin>mongod --dbpath "C:\Program Files\MongoDB\Server\4.4\data\BPBOnlineDB551" --logpath "C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDB551\mongod.log" --port 27018 --storageEngine wiredTiger --journal --replSet BPBOnlineReplicaSet
{"t": {"$date": "2020-10-24T14:04:33.958Z"}, "s": "I", "c": "CONTROL", "id": 20697, "ctx": "main", "msg": "Renamed existing log file", "attr": {"oldLogPath": "C:\\Program Files\\MongoDB\\Server\\4.4\\log\\BPBOnlineDB551\\mongod.log", "newLogPath": "C:\\Program Files\\MongoDB\\Server\\4.4\\log\\BPBOnlineDB551\\mongod.log.2020-10-24T14-04-33"}}
2

Message returned by "mongod" command

```

**Figure 18.25: Starting the Secondary Server Instance using "mongod"**

- Now, as the first secondary instance has started successfully on the port number 27018, we can add this instance to our replica set. To do this, open the existing shell window of the primary instance and type the following command in the Shell prompt of the primary instance:

```
rs.add("localhost:27018")
```

After you enter the preceding command in your MongoDB Shell of the primary instance, you will get the result similar to the following screenshot. You will see that in our case, the status is ok, as shown in the following figure:

```

Administrator: Command Prompt - mongo --host localhost --port 27017
{
    "_id" : ObjectId("5f8a2a0a0000000000000000"),
    "name" : "localhost:27017",
    "health" : 1,
    "state" : 1,
    "stateStr" : "PRIMARY",
    "uptime" : 2584,
    "optime" : {
        "ts" : Timestamp(1603547506, 1),
        "t" : NumberLong(1)
    },
    "optimeDate" : ISODate("2020-10-24T13:51:46Z"),
    "syncSourceHost" : "",
    "syncSourceId" : -1,
    "infoMessage" : "",
    "electionTime" : Timestamp(1603546785, 2),
    "electionDate" : ISODate("2020-10-24T13:39:45Z"),
    "configVersion" : 1,
    "configTerm" : 1,
    "self" : true,
    "lastHeartbeatMessage" : ""
},
{
    "ok" : 1,
    "$clusterTime" : {
        "clusterTime" : Timestamp(1603547506, 1),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId" : NumberLong(0)
        }
    },
    "operationTime" : Timestamp(1603547506, 1)
}
PBONlineReplicaSet> rs.add("localhost:27018") ①
{
    "ok" : 1,
    "$clusterTime" : {
        "clusterTime" : Timestamp(1603548704, 1),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId" : NumberLong(0)
        }
    },
    "operationTime" : Timestamp(1603548704, 1)
}
PBONlineReplicaSet>

```

**Figure 18.26:** "rs.add()" – Adding new Secondary Members to the Replica Set

26. Now we can add our third member to this replica set. But before we do so, we need to start the third instance on a different port. To do this, we need to start our second secondary member with the help of the `mongod` command which is similar to what we used during the starting of our first and second member of the replica set.

To do this, open another command prompt in the administrative mode without closing any other command prompt that are already open. Navigate to the `bin` directory of the MongoDB installation and type the following command, as shown in the following figure:

```

mongod --dbpath "C:\Program
Files\MongoDB\Server\4.4\data\BPBOnlineDBSS2" --logpath
"C:\Program
Files\MongoDB\Server\4.4\log\BPBOnlineDBSS2\mongod.log" --
port 27019 -storageEngine wiredTiger --journal --replSet
BPBOnlineReplicaSet

```

Here, the parameters that are given in the `mongod` command are described as follows:

- `--dbpath`: Path of our DB directory location
- `--logpath`: Path of our DB log file location

- **--port:** For our first secondary server, we will use 27019
- **--storageEngine:** We will use "wiredTiger"
- **--journal:** This will enable the MongoDB journaling
- **--replSet:** In our case, we will use BPBOnlineReplicaSet, which is common for all the three member nodes or instances in the replica set.



The screenshot shows an Administrator Command Prompt window on Windows 10. The command entered is:

```
Administrator: Command Prompt - mongod --dbpath "C:\Program Files\MongoDB\Server\4.4\data\BPBOnlineDB0552" --logpath "C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDB0552\mongod.log" --port 27019 --storeLogPath
```

Output from the command:

```
C:\Windows\system32>cd "c:\Program Files\MongoDB\Server\4.4\bin"
c:\Program Files\MongoDB\Server\4.4\bin>mongod --dbpath "C:\Program Files\MongoDB\Server\4.4\data\BPBOnlineDB0552" --logpath "C:\Program Files\MongoDB\Server\4.4\log\BPBOnlineDB0552\mongod.log" --port 27019 --storageEngine wiredTiger --journal --replSet BPBOnlineReplicaSet
{"t":("$date": "2020-10-24T14:19:53.585Z"), "s": "I", "c": "CONTROL", "id": 20697, "ctx": "main", "msg": "Renamed existing log file", "attr": {"oldLogPath": "C:\\Program Files\\MongoDB\\Server\\4.4\\log\\BPBOnlineDB0552\\mongod.log", "newLogPath": "C:\\Program Files\\MongoDB\\Server\\4.4\\log\\BPBOnlineDB0552\\mongod.log", "newFile": "mongod.log.2020-10-24T14-19-53"}}
```

A red box highlights the message: "Message returned by \"mongod\" command".

**Figure 18.27: Starting the Secondary Server Instance using "mongod"**

27. Now, as the second secondary instance has started successfully on the port number 27019, we can add this instance into our replica set. To do this, open the existing shell window of the primary instance and type the following command in the Shell prompt of the primary instance:

```
rs.add("localhost:27019")
```

Once you enter the preceding command in your MongoDB Shell of the primary instance, you will get the result similar to the following screenshot. You will see that in our case the status is **ok**, as shown in the following figure:

```

Administrator: Command Prompt - mongo --host localhost --port 27017
{
    "infoMessage": "",
    "electionTime": Timestamp(1603546785, 2),
    "electionDate": ISODate("2020-10-24T13:39:45Z"),
    "configVersion": 1,
    "configTerm": 1,
    "self": true,
    "lastHeartbeatMessage": ""
}
],
"ok": 1,
"$clusterTime": {
    "clusterTime": Timestamp(1603547506, 1),
    "signature": {
        "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
        "keyId": NumberLong(0)
    }
},
"operationTime": Timestamp(1603547506, 1)
}
BPOnlineReplicaSet:PRIMARY> rs.add("localhost:27018")
{
    "ok": 1,
    "$clusterTime": {
        "clusterTime": Timestamp(1603548704, 1),
        "signature": {
            "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId": NumberLong(0)
        }
    },
    "operationTime": Timestamp(1603548704, 1)
}
BPOnlineReplicaSet:PRIMARY> rs.add("localhost:27019") ①
{
    "ok": 1, ②
    "$clusterTime": {
        "clusterTime": Timestamp(1603549313, 1),
        "signature": {
            "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId": NumberLong(0)
        }
    },
    "operationTime": Timestamp(1603549313, 1)
}
BPOnlineReplicaSet:PRIMARY>

```

**Figure 18.28:** "rs.add()" – Adding new Secondary Members to the Replica Set

28. We can again verify the status of our replica set using the following command:

```
rs.status()
```

Just enter the preceding command in your MongoDB Shell of the primary instance and you will get the result similar to the following screenshot. You will get the information about the replica set members and the status for which is `ok`, in our case, as shown in the following figure:

```

Administrator: Command Prompt - mongo --host localhost --port 27017
{
    "syncSourceId": 0,
    "infoMessage": 1,
    "configVersion": 3,
    "configTerm": 1
},
{
    "id": 2,
    "name": "localhost:27018", ①
    "health": 1,
    "state": 2,
    "stateStr": "SECONDARY",
    "uptime": 288,
    "optime": {
        "ts": Timestamp(1603549596, 1),
        "t": NumberLong(1)
    },
    "optimeDurable": {
        "ts": Timestamp(1603549596, 1),
        "t": NumberLong(1)
    },
    "optimeDate": ISODate("2020-10-24T14:26:36Z"),
    "optimeDurableDate": ISODate("2020-10-24T14:26:36Z"),
    "lastHeartbeat": ISODate("2020-10-24T14:26:41.551Z"),
    "lastHeartbeatRecv": ISODate("2020-10-24T14:26:41.946Z"),
    "pingMs": NumberLong(0),
    "lastHeartbeatMessage": "",
    "syncSourceHost": "localhost:27018",
    "syncSourceId": 0,
    "infoMessage": 1,
    "configVersion": 3,
    "configTerm": 1
},
],
"ok": 1, ②
"$clusterTime": {
    "clusterTime": Timestamp(1603549596, 1),
    "signature": {
        "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
        "keyId": NumberLong(0)
    }
},
"operationTime": Timestamp(1603549596, 1)
}
PRIMARY>

```

**Figure 18.29: "rs.status()" – Checking the status of the Replica Set**

29. We can run one more command that will give the status of our primary replica set, Ether the following command in the MongoDB Shell of the primary instance:

```
rs.isMaster()
```

Just enter the preceding command in your MongoDB Shell of the primary instance. You will get the result similar to the following screenshot. You will get the information about the primary server and you will see that it has the value `true`, in our case, as shown in the following figure:

```

Administrator: Command Prompt - mongo --host localhost --port 27017
{
    "ok": 1,
    "clusterTime": {
        "clusterTime": Timestamp(1603549506, 1),
        "signature": {
            "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
            "keyId": NumberLong(0)
        }
    },
    "operationTime": Timestamp(1603549596, 1)
}
BPBOnlineReplicaSet:PRIMARY> rs.isMaster()
{
    "topologyVersion": {
        "processId": ObjectId("5f0427ac313c29434a1536fa"),
        "counter": NumberLong(8)
    },
    "hosts": [
        "localhost:27017", ②
        "localhost:27018",
        "localhost:27019"
    ],
    "setName": "BPBOnlineReplicaSet",
    "setVersion": 3,
    "isMaster": true, ③
    "secondary": false,
    "primary": "localhost:27017", ④
    "me": "localhost:27017",
    "electionId": ObjectId("7fffffff0000000000000001"),
    "lastWrite": {
        "optime": {
            "ts": Timestamp(1603549836, 1),
            "t": NumberLong(1)
        },
        "lastWriteDate": ISODate("2020-10-24T14:30:36Z"),
        "majorityOptime": {
            "ts": Timestamp(1603549836, 1),
            "t": NumberLong(1)
        },
        "majorityWriteDate": ISODate("2020-10-24T14:30:36Z")
    },
    "maxbsonobjsize": 16777216,
    "maxMessageSizeBytes": 48000000
}

```

**Figure 18.30:** "rs.isMaster()" – Checking the status of the Primary Member of Replica Set

Now, we have successfully created the replica set with three members. But to check whether the replication is happening correctly or not, we need to verify with the help of some process. Let us verify this with some MongoDB database and collection and create new documents under the MongoDB collection.

## Verifying the MongoDB replication using data

For the verification process, we need to follow these steps:

1. Go to the MongoDB Shell of the primary instance and create a new database and collection and insert some documents. We will run the following command in the MongoDB Shell of our primary server, as shown in the following figure:

```

use BPBOnlineRepliationDB
db. BPBOnlineRepliationCollection.insert({ "booktitle": "Docker
Demystified" })
db.
BPBOnlineRepliationCollection.insert({ "booktitle": "Hardware
Description Language Demystified" })
db.BPBOnlineRepliationCollection.find().pretty()

```

```

Administrator Command Prompt - mongo --host localhost --port 27017
> BPBOnlineReplicaSet:PRIMARY> use BPBOnlineRepliationDB
switched to db BPBOnlineRepliationDB
BPBOnlineReplicaSet:PRIMARY> db.BPBOnlineRepliationCollection.insert([{"booktitle": "Docker Demystified"}]) ②
WriteResult({ "nInserted" : 1 })
BPBOnlineReplicaSet:PRIMARY> db.BPBOnlineRepliationCollection.insert([{"booktitle": "Hardware Description Language Demystified"}]) ③
WriteResult({ "nInserted" : 1 })
BPBOnlineReplicaSet:PRIMARY> db.BPBOnlineRepliationCollection.find().pretty() ④
{
    "_id" : ObjectId("5f9441fca0fa91e1929c2eeff3"),
    "booktitle" : "Docker Demystified"
}
{
    "_id" : ObjectId("5f9441fca0fa91e1929c2eeff4"),
    "booktitle" : "Hardware Description Language Demystified"
}
BPBOnlineReplicaSet:PRIMARY>

```

**Figure 18.31:** Running few Data Related commands on the MongoDB Shell of Primary Server

- Now, open the MongoDB Shell of our secondary server in the replica set using the new command prompt in the administrative mode and run the following command. It will then open the MongoDB Shell prompt of our secondary server in the replica set. Please keep all the earlier command prompts open, as shown in the following figure:

```
mongo --host localhost --port 27018
```

```

Administrator Command Prompt - mongo --host localhost --port 27018
Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. All Rights Reserved.

C:\Windows\system32>cd "c:\Program Files\MongoDB\Server\4.4\bin" ①
c:\Program Files\MongoDB\Server\4.4\bin>mongo --host localhost --port 27018 ②
MongoDB shell version v4.4.1
connecting to: mongodb://localhost:27018/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("7aa07562-862d-4f35-ac46-a1075a2f546") }
MongoDB server version: 4.4.1

The server generated these startup warnings when booting:
2020-10-24T19:34:35.544+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted.
2020-10-24T19:34:35.545+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip address to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the 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()
```
BPBOnlineReplicaSet:SECONDARY ③

```

**Figure 18.32:** Opening MongoDB Shell of the Secondary Server

- In the same MongoDB Shell (secondary server), type the following command to verify if the replication has taken place and you will be able to see the documents under the collection created by us from our primary server Shell:

```
use BPBOnlineRepliationDB
db.BPBOnlineRepliationCollection.find().pretty()
```

Once you run the preceding commands from the secondary server first time, it will give you an error as some configuration needs to

take place from our secondary servers Shell, as shown in the following figure:

```

Administrator: Command Prompt - mongo --host localhost --port 27018
C:\Program Files\MongoDB\Server\4.4\bin>mongo --host localhost --port 27018
MongoDB shell version v4.4.1
connecting to: mongodb://localhost:27018/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session "id" : UUID("71a07562-062d-4f35-ac46-a1075a2f1546")
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-10-24T19:34:35.544+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-10-24T19:34:35.545+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <addr> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the 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()
...
#> db.$PBOnlineReplicaSet.find().pretty() !  

Error: error: {
  "topologyVersion": {
    "processId": ObjectId("5f043472a81149547a3bacbb"),
    "counter": NumberLong(4)
  },
  "operationTime": Timestamp(1603552466, 1),
  "ok": 0,
  "errmsg": "not master and slaveOk=false", 2
  "code": 13435,
  "codeName": "NotMasterNoSlaveOK",
  "$clusterTime": {
    "clusterTime": Timestamp(1603552466, 1),
    "signature": {
      "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
      "keyId": NumberLong(0)
    }
  }
}
#>

```

*Figure 18.33: Secondary Server giving error on db.collection.find()*

4. In order to resolve this issue, we need to run the following command on the MongoDB Shell of both of our secondary servers, as shown in the following figure:

```

rs.secondaryOk()  

rs.status()

```

```

Administrator: Command Prompt - mongo --host localhost --port 27018
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()
...
#> db.$PBOnlineReplicaSet.find().pretty()
Error: error: {
  "topologyVersion": {
    "processId": ObjectId("5f043472a81149547a3bacbb"),
    "counter": NumberLong(4)
  },
  "operationTime": Timestamp(1603553097, 1),
  "ok": 0,
  "errmsg": "not master and slaveOk=false",
  "code": 13435,
  "codeName": "NotMasterNoSlaveOK",
  "$clusterTime": {
    "clusterTime": Timestamp(1603553097, 1),
    "signature": {
      "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
      "keyId": NumberLong(0)
    }
  }
}
#> rs.secondaryOk() !  

#> rs.status()  

{
  "set": "SPBOnlineReplicaSet",
  "date": ISODate("2020-10-24T15:25:28.237Z"),
  "myState": 2,
  "term": NumberLong(1),
  "syncSourceHost": "localhost:27017",
  "syncSourceId": 0,
  "heartbeatIntervalMillis": NumberLong(2000),
  "majorityVoteCount": 2,
  "writeMajorityCount": 2,
  "votingMembersCount": 3,
  "writableVotingMembersCount": 3,
  "options": {
    "lastCommittedOpTime": {
      ...
    }
  }
}

```

**Figure 18.34: Secondary Server – Resolving the Issues**

- Now go back to the MongoDB Shell prompt of the primary server and add one more document in the same collection by giving the following command, as shown in the following figure:

```
db.BPBOnlineRepliationCollection.insert({ "booktitle" : "MongoDB
Replication" })
```

```
Administrator: Command Prompt - mongo --host localhost --port 27019
BPBOnlineReplicaSet:PRIMARY> use BPBOnlineRepliationDB
switched to db BPBOnlineRepliationDB
BPBOnlineReplicaSet:PRIMARY> db.BPBOnlineRepliationCollection.insert({ "booktitle" : "Docker Demystified" })
WriteResult({ "nInserted" : 1 })
BPBOnlineReplicaSet:PRIMARY> db.BPBOnlineRepliationCollection.insert({ "booktitle" : "Hardware Description Language Demystified" })
WriteResult({ "nInserted" : 1 })
BPBOnlineReplicaSet:PRIMARY> db.BPBOnlineRepliationCollection.find().pretty()
{
    "_id" : ObjectId("5f9441fec0fa91e1929c2eeef3"),
    "booktitle" : "Docker Demystified"
}

{
    "_id" : ObjectId("5f9441fec0fa91e1929c2eeef4"),
    "booktitle" : "Hardware Description Language Demystified"
}
BPBOnlineReplicaSet:PRIMARY> db.BPBOnlineRepliationCollection.find().pretty()
{
    "_id" : ObjectId("5f9441fec0fa91e1929c2eeef3"),
    "booktitle" : "Docker Demystified"
}

{
    "_id" : ObjectId("5f9441fec0fa91e1929c2eeef4"),
    "booktitle" : "Hardware Description Language Demystified"
}
BPBOnlineReplicaSet:PRIMARY> db.BPBOnlineRepliationCollection.insert({ "booktitle" : "MongoDB Replication" }) ①
WriteResult({ "nInserted" : 1 })
BPBOnlineReplicaSet:PRIMARY>
```

**Figure 18.35: Primary Server adding one more record to the document**

- Now, in the MongoDB Shell of any of the secondary servers, type the following command to verify if the replication has taken place. You will be able to see the documents including the newly created document under the collection created by us from our primary server Shell:

```
use BPBOnlineRepliationDB
db.BPBOnlineRepliationCollection.find().pretty()
```

Once you run the preceding command from the both of the secondary servers for the first time, it will show you all the documents present in the collection, and thus, we can verify now that the replication is taking place correctly on the replica set created by us, as shown in the following figure:

```
Administrator: Command Prompt - mongo --host localhost --port 27019
BPBOnlineReplicaSet:SECONDARY> use BPBOnlineRepliationDB
switched to db BPBOnlineRepliationDB
BPBOnlineReplicaSet:SECONDARY> db.BPBOnlineRepliationCollection.find().pretty() ②
{
    "_id" : ObjectId("5f9441fec0fa91e1929c2eeef3"),
    "booktitle" : "Docker Demystified"
}

{
    "_id" : ObjectId("5f9441fec0fa91e1929c2eeef4"),
    "booktitle" : "Hardware Description Language Demystified" ③
}

{
    "_id" : ObjectId("5f9448790fa91e1929c2eeef5"),
    "booktitle" : "MongoDB Replication"
}
BPBOnlineReplicaSet:SECONDARY>
```

**Figure 18.36:** Secondary Server displaying documents with db.collection.find()

## Conclusion

In this chapter, we covered the replication part of MongoDB. We learned about the replication and the replica sets in a quick recap. We also learned about the MongoDB heartbeats and how the heartbeats play an important role in the replicated environment. We also learned how the election of the new primary member takes place and when this election process starts. Later in this chapter, we covered the pre-configuration steps before we started with the practical step-by-step method to create the replicated environment with the MongoDB primary and secondary instances. We also learned how to setup the replicated MongoDB environment with the step-by-step method. In the last part of this chapter, we learned how to verify the replication setup if it has been configured correctly with the help of the data.

## Questions

1. What do you understand by replication?
2. What is a replica set?
3. What do you understand by the MongoDB heartbeats?
4. How does the election of the primary member take place and when does it happen usually between the other members of the replica set?
5. Explain the process of creating the replicated environment in your Windows machine?
6. Name the command by which we can check the status of the replica set.
7. Explain the process of verifying the replicated environment you created with the step-by-step method explained in this chapter.

# CHAPTER 19

## Sharding in MongoDB

This chapter covers the sharding part of MongoDB. In this chapter, we will learn about sharding and shaded clusters in a quick recap. We will also learn the importance of the config database in a sharded environment. We will also learn about the shard keys. Later in this chapter, we will cover the pre-configuration steps before we start with the practical step-by-step method to create a sharded environment with the MongoDB replica sets. We will then also learn how to setup the MongoDB sharded environment. We will learn the complete the step-by-step process of sharding in an easy-to-understand 8 steps.

### Structure

In this chapter, we will discuss the following topics:

- Basic introduction to sharding – quick recap
- Sharded clusters
- Read and write operations and the importance of config database
- Shard key
- Pre-configuration steps
- Starting with MongoDB sharding on Windows machine
  - The complete step-by-step process of sharding
  - Step 1 – Stop the existing MongoDB services on Windows machine
  - Step 2 – Create the first replica set (our first shard)
  - Step 3 – Create the second replica set (our second shard)
  - Step 4 – Create the third replica set (our third shard)
  - Step 5 – Create the replica set of the config servers in a sharded environment
  - Step 6 - Start MongoDB in sharded environment as "mongos"

- Step 7 - Add MongoDB in shard keys
- Step 8 - Verify MongoDB sharding using data

## **Objectives**

After studying this unit, you should be able to get a quick recap and introduction to the process of sharding and sharded clusters. You will also understand the read and write operations in sharding and the importance of config database. You will also learn about the shard keys. Later in this chapter, you will learn the pre-configuration steps to setup the sharded environment and also learn how we can setup MongoDB sharding on Windows machine. Major part of this chapter is focused on the step-by-step process of sharding in an easy-to-understand 8 steps to give you the practical knowledge on this topic.

## **Basic introduction to sharding – quick recap**

Sharding is a process of distributing the large data across multiple machines or servers. There are cases where the applications have large sets of data which cannot be served by a single machine or server due to its hardware limits. These limits can be due to the CPU capacities or sometimes due to the memory capacities like RAM or disk drives.

In this case, MongoDB provides the feature of sharding where we can distribute our large data across different machines or servers. By doing this, we can serve this large data easily by an increased computational power we gained by using more machines or servers instead of a single machine or server.

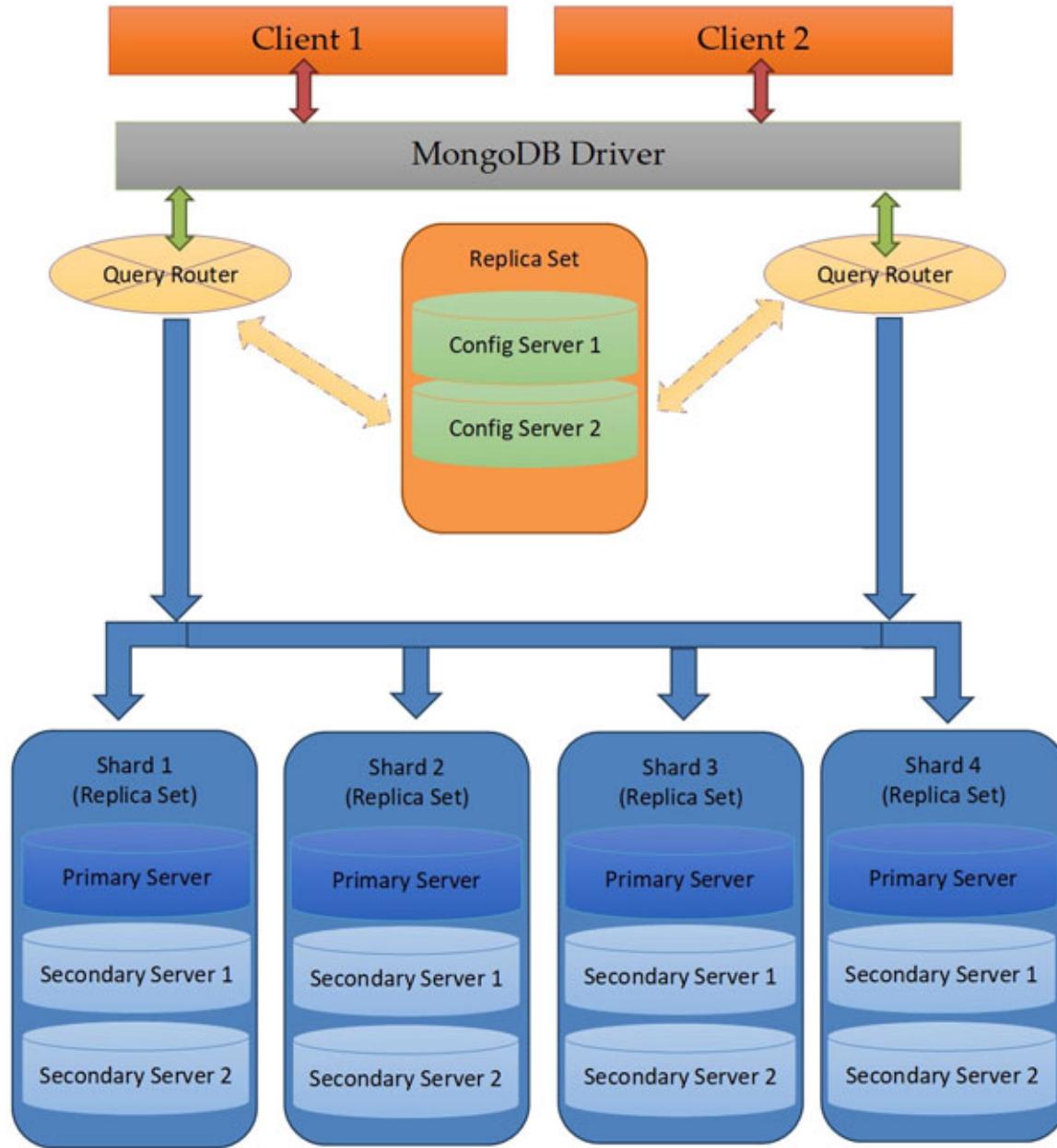
Some of the benefits of sharding are as follows:

- An increase in the computational power due to the use of multiple machines or servers.
- Lower costs because instead of increasing the computational capacities of one machine, which also has a limitation, we increase the computational capacities by involving multiple machines which can together create a lot of computational power and is a cheaper option.

- MongoDB supports horizontal scaling by using sharding which is a more effective solution. Many a times, it is not practically possible to increase the CPU and memory capacities of a single machine as it has a maximum limit and is expensive. Increasing the computational capacity of a single machine is also termed as vertical scaling. So, it is always a better and cheaper option to increase the computational power by using multiple machines. This is called horizontal scaling.

## **Sharded clusters**

In simple terms, sharded clusters are groups of MongoDB instances used to serve large data. Here, the data is served to the applications by splitting the large data into the multiple sharded clusters, as shown in the following figure:



*Figure 19.1: Sharding in MongoDB*

## Read and write operations and importance of config database

As we studied earlier in this chapter, in MongoDB, we can distribute the large data into small data using sharding. So, when we divide the larger data into smaller subsets using sharding, it is totally transparent to an application.

Whenever we perform sharding, we specify a sharding key for the MongoDB collections so that the data can be handled in an effective manner. So, whenever the clients send any requests, they should include this sharded key so that the data can be fetched directly from the particular shard (subset of the sharded data). If the client does not send its query with the shard key, then the read operation will take more time since every shard has to be involved in the query.

In the sharded clusters, the config database is maintained and is very helpful in routing the queries to the shards. So, if any client sends the query with the sharded key, then by using the metadata from the config database, the queries are redirected or routed to a particular shard.

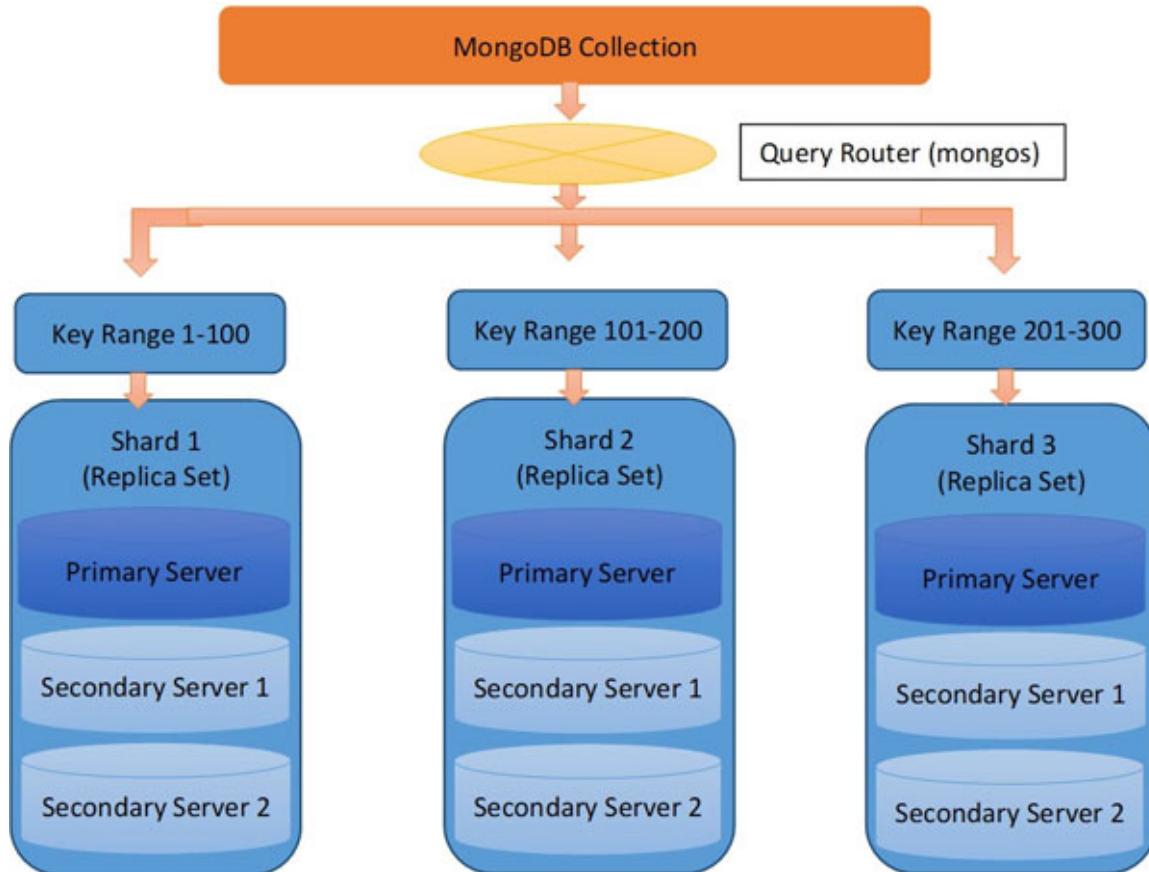
Whenever there is a write request from the client for the sharded collections in the sharded cluster, the request goes to the particular shard (subset of the sharded data) which is responsible for that particular data set.

Here, the write operation is performed using the metadata which is in the config database. Based on this metadata, the redirection or routing is done for the write operation to the particular shard in the sharded cluster. The write operations are done only on the primary nodes of the sharded cluster.

## **Shard key**

In a MongoDB sharded environment, the shard key plays an important role in distributing the MongoDB collection data to different shards. Shard key is an indexed field or indexed compound field of a collection. We need to specify the shard key during the implementation of the sharded environment.

During any read or write request from the client in a shaded environment, the query router redirects the requests to the related shard based on these shard keys, as shown in the following figure:



*Figure 19.2: Shard Key in MongoDB*

## Pre-configuration steps

In order to perform sharding on our machine, let us first understand the steps we need to take to correctly setup a replication environment on our machine using MongoDB.

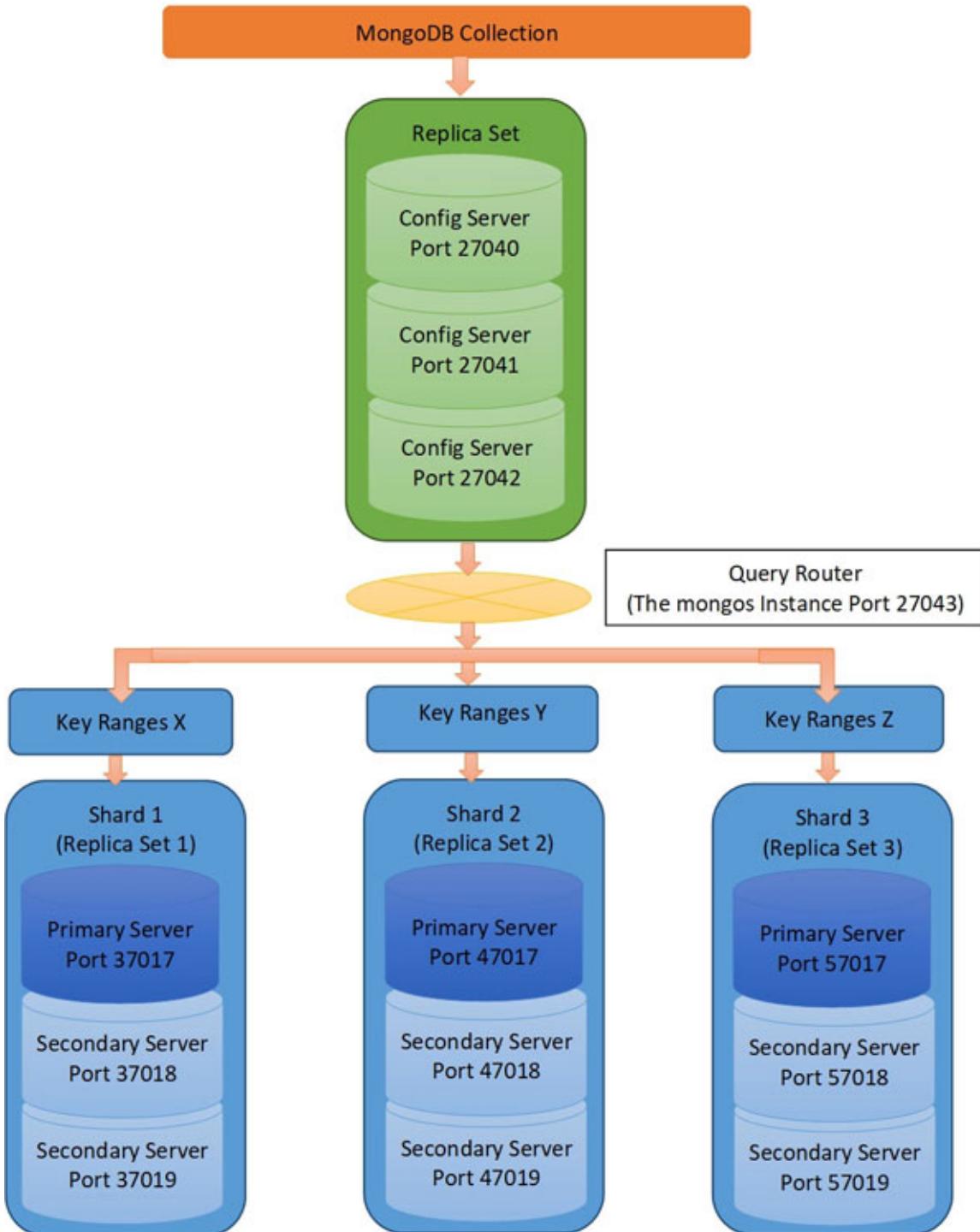
To start the sharding process, we need to follow these steps:

1. We will use Windows operating system having MongoDB installed
2. We will stop the existing MongoDB service on our Windows machine
3. We will create 3 replica sets which has 3 MongoDB nodes to form a group of replica sets
4. In each of the 3 replica sets, we will create one MongoDB instance which will act as the primary server or node of the replica set
5. In each of the 3 replica sets, we will create two MongoDB instances which will act like the secondary servers or nodes of the

replica set

6. So, basically we will form 3 groups of replica sets having one primary and two secondary servers
7. We will then create a set of 3 configuration servers
8. After that, we will create a sharding server with a sharded key on a collection.
9. We will now insert a few documents in the sharded collection to check if it is working properly.

The following diagram depicts our MongoDB replica sets grouped with sharded environment on our Windows machine:



*Figure 19.3: Sharding in MongoDB on Windows Machine*

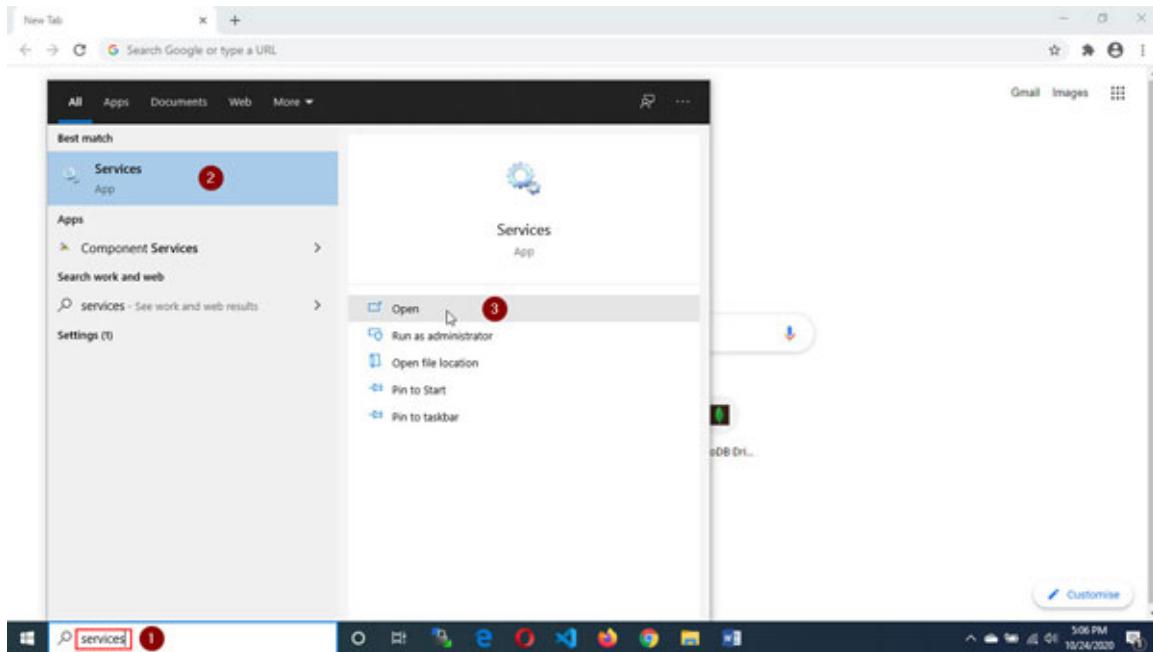
## Starting with MongoDB sharding on Windows machine

Note that here we will create MongoDB instances (cluster nodes) with the help of Windows service option of `mongod`. Instead of starting `mongod` from the command prompt and keep that running, we do this to prevent opening of so many command prompt windows so that our system won't get messy as we need to create 12 MongoDB instances on our Windows machine and then we need to start `mongos` (MongoDB sharing environment) from the Windows command prompt.

Following is the step-by-step process to do the sharding of MongoDB:

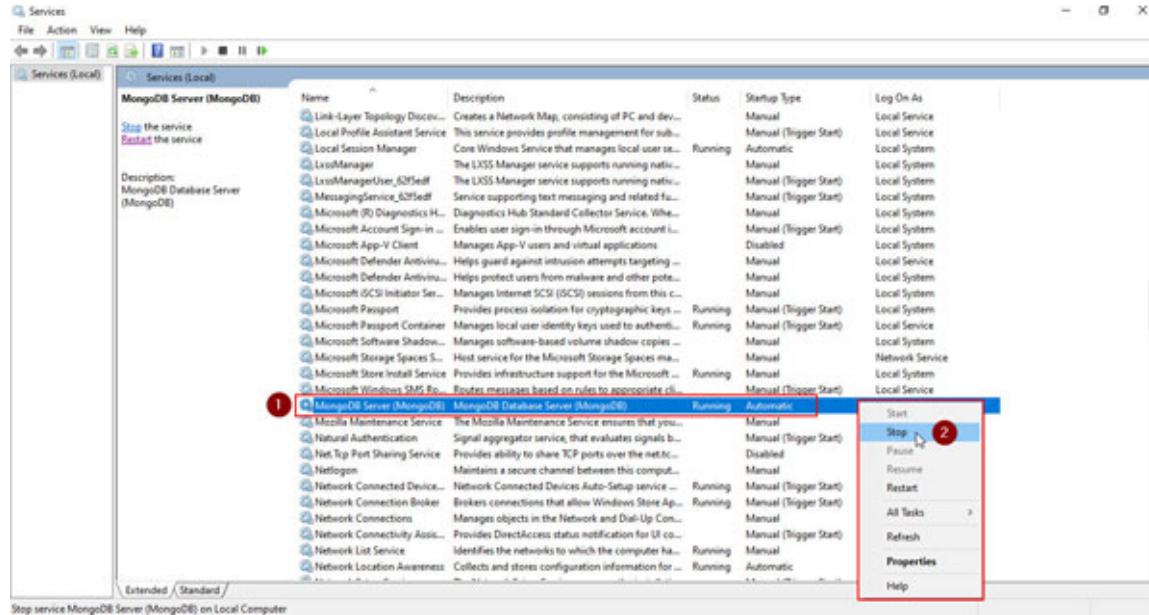
## Step 1 –Stop the existing MongoDB services on Windows machine

1. In the `search Box` of your Windows, type `services` and open Windows service manager, as shown in the following screenshot:



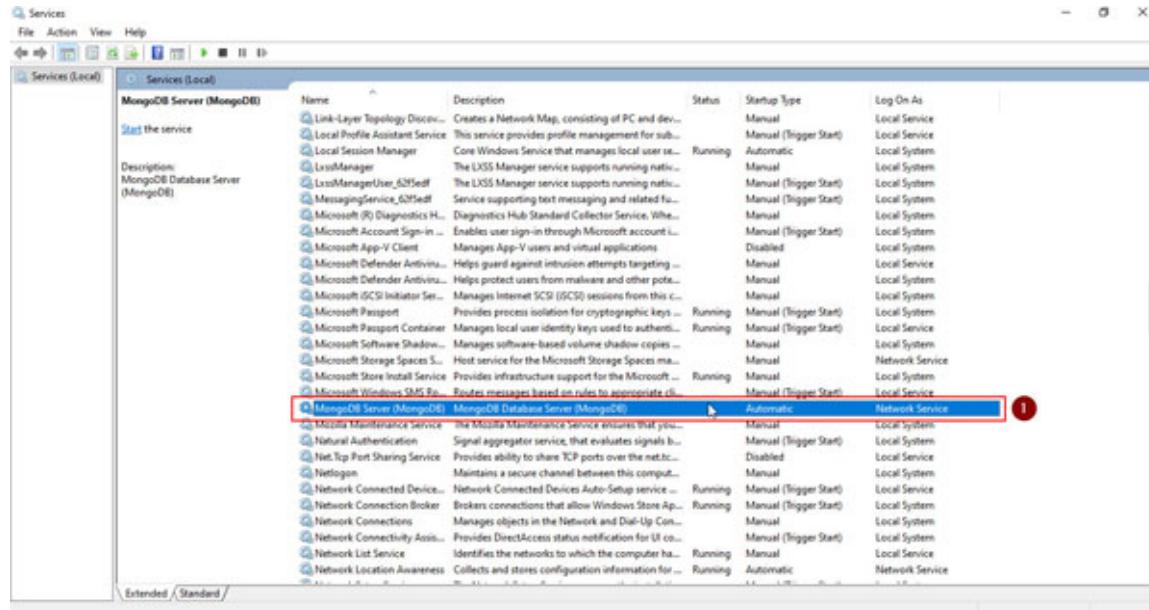
*Figure 19.4: Open Windows Services Manager*

2. Now, in the Windows service manager, navigate to MongoDB service, then right click on the `service`. This will open up a service related `Menu` which has the `stop` option. Click this option to stop the MongoDB service, as shown in the following screenshot:



**Figure 19.5: Windows Service Manager – Stop MongoDB Service**

3. Now, as the MongoDB service will stop, you will see that the running status is blank. Now, you can close the Windows service manager, as shown in the following screenshot:



**Figure 19.6: Windows Service Manager – MongoDB Service has been stopped**

## Step 2 – Create the first replica set (our first shard)

Let us create the first replica set for the MongoDB sharded environment. To do this, we need to perform the following steps:

1. We need to create few folders to keep the MongoDB instances data and log files. Now, instead of doing it manually one-by-one, we will create these folders with the help of the following commands which will automatically create these folders:

For MongoDB data files:

```
mkdir "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplica
Set1\BPBOnlineDBS1" "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineD
BS2" "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineD
BS3"
```

For MongoDB log files:

```
mkdir "c:\MongoDB-
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB
S1" "c:\MongoDB-
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB
S2" "c:\MongoDB-
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB
S3"
```

If you look this closely, you will find that our main directory is MongoDB-Sharding under the c: drive on your Windows machine (you may change the name and the directory according to your wish).

Then, under the "MongoDB-Sharding" directory, there will be 2 child directories, `data`, which will keep the MongoDB data files, and `log`, which will keep the MongoDB log files.

So, till now, the structure will be as follows:

- c:\MongoDB-Sharding\data
- c:\MongoDB-Sharding\log

Then, in both the `data` and `log` directories, we will have two common types of directories, `BPBOnlineShard1` (which is the name

of our first shard), and under this directory there would be another directory, BPBOnlineReplicaSet1 (which is the name of our first replica set).

At last, under the BPBOnlineReplicaSet1 directory, we will have 3 different directories which will cater to each one of the 3 MongoDB instances in the replica set for these directories as follows:

### For MongoDB data files (**complete path**):

- <Main Directory>\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS1
- <Main Directory>\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS2
- <Main Directory>\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS3

### For MongoDB log files (**complete path**):

- <Main Directory>\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS1
- <Main Directory>\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS2
- <Main Directory>\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS3

Here:

- <Main Directory> = c:\MongoDB-Sharding\
- "BPBOnlineDBS1", "BPBOnlineDBS2" and "BPBOnlineDBS3" are the names of the database server directories

If you closely look at the directory structure, it looks perfect to create the first replica set in the sharded environment.

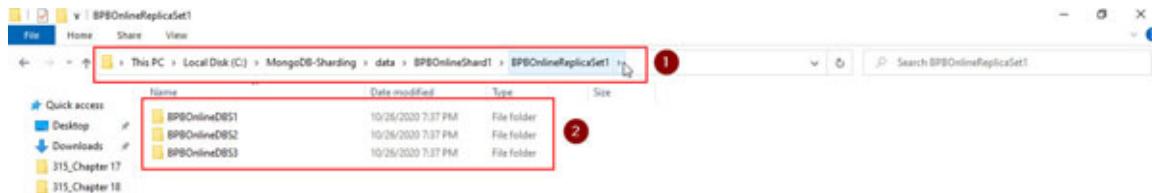
2. Let us first create new directories which will be the `data` directories for our MongoDB instances. To do this, open the Windows command prompt in the administrator mode and then enter the following command, as shown in the following screenshot:

```
mkdir "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineD
BS1" "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineD
BS2" "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineD
BS3"
```



**Figure 19.7: MongoDB Data Folders - Creating new Folders**

3. After you have created the new directories, you should see them under the `c:` drive, similar to what is shown in the following screenshot:



**Figure 19.8: MongoDB Data Folders – New Folders gets created**

4. Now, let's first create the new directories which would be the `log` directories for our MongoDB instances. To do this, open the Windows command prompt in the administrator mode and then enter the following command, as shown in the following screenshot:

```
mkdir "c:\MongoDB-
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB
S1" "c:\MongoDB-
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB
S2" "c:\MongoDB-
```

Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS3"



**Figure 19.9: MongoDB Log Folders - Creating new Folders**

5. After you have created new directories, you should see them under the c: drive, similar to what is shown in the following screenshot:



**Figure 19.10: MongoDB Log Folders – New Folders gets created**

6. Now, we will start our first server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```
mongod --replSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS1\mongod.log" --port 37017 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 37017" --serviceDisplayName "BPBOnlineSharding MongoDB 37017" --serviceDescription "MongoDB on Port 37017 BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS1"
```

This command needs to run in the command prompt in the administrator mode. Here, the parameters given in the `mongod` command are described as follows:

--dbpath= Path of our DB Directory Location  
--logpath= Path of our DB Log File Location  
--port = Port Number for MongoDB Instance, in our case we will use "37017"

```

--replSet= In our case we will be using
"BPBOnlineReplicaSet1", which is common for all 3 Member
Nodes or Instances in the Replica Set 1 under the MongoDB
Sharded Environment 1 "BPBOnlineShard1".
--shardsvr = MongoDB Option for Sharding (Sharded
Environment)
--install = To Install the MongoDB as Service
--serviceName= Name of the Windows Service, in our case we
will be using "BPBOnlineSharding MongoDB 37017"
--serviceDisplayName= Display Name of the Windows Service, in
our case we will be using "BPBOnlineSharding MongoDB 37017"
--serviceDescription= Description of the Windows Service, in
our case we will be using "MongoDB on Port 37017
BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS1"

```

7. Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following screenshot:



**Figure 19.11: Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory**

8. Now, type the following command with all the parameters that we will create Windows service for our first server in the replica set as follows:

```

mongod --replSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineD
BS1" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB
S1\mongod.log" --port 37017 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 37017" --
serviceDisplayName "BPBOnlineSharding MongoDB 37017" --
serviceDescription "MongoDB on Port 37017 BPBOnlineShard1-
BPBOnlineReplicaSet1-BPBOnlineDBS1"

```

Once you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:



The screenshot shows an Administrator Command Prompt window. The command entered is:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --replSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS1\mongod.log" --port 37017 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 37017" --serviceDisplayName "BPBOnlineSharding MongoDB 37017" --serviceDescription "MongoDB on Port 37017 BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS1"
```

Two red circles are overlaid on the window: circle 1 points to the command line, and circle 2 points to the status message at the bottom.

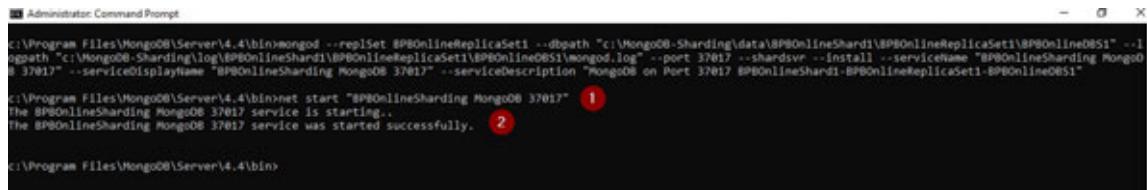
**Figure 19.12: Adding the new MongoDB Windows Service - "mongod" with Replication parameters**

- Now, type the following command to start this Windows service from your command prompt which is already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 37017"
```



The screenshot shows an Administrator Command Prompt window. The command entered is:

```
c:\Program Files\MongoDB\Server\4.4\bin>net start "BPBOnlineSharding MongoDB 37017"
```

Two red circles are overlaid on the window: circle 1 points to the status message "The BPBOnlineSharding MongoDB 37017 service is starting..", and circle 2 points to the final message "The BPBOnlineSharding MongoDB 37017 service was started successfully."

**Figure 19.13: Starting the new MongoDB Windows Service**

- Now, we will start our second server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```
mongod --replSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS2" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS2\mongod.log" --port 37018 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 37018" --serviceDisplayName "BPBOnlineSharding MongoDB 37018" --serviceDescription "MongoDB on Port 37018 BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS2"
```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the `mongod`

command are described as follows:

```
--dbpath= Path of our DB Directory Location  
--logpath= Path of our DB Log File Location  
--port = Port Number for MongoDB Instance, in our case we  
will use "37018"  
--replSet= In our case we will be using  
"BPBOnlineReplicaSet1", which is common for all 3 Member  
Nodes or Instances in the Replica Set 1 under the MongoDB  
Sharded Environment 1 "BPBOnlineShard1".  
--shardsvr = MongoDB Option for Sharding (Sharded  
Environment)  
--install = To Install the MongoDB as Service  
--serviceName= Name of the Windows Service, in our case we  
will be using "BPBOnlineSharding MongoDB 37018"  
--serviceDisplayName= Display Name of the Windows Service, in  
our case we will be using "BPBOnlineSharding MongoDB 37018"  
--serviceDescription= Description of the Windows Service, in  
our case we will be using "MongoDB on Port 37018  
BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS2"
```

11. Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following screenshot:



**Figure 19.14:** Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory

12. Now, type the following command with all the parameters that we will create Windows service for our second server in the replica set as follows:

```
mongod --replSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-  
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineD  
BS2" --logpath "c:\MongoDB-  
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB  
S2\mongod.log" --port 37018 --shardsvr --install --
```

```

serviceName "BPBOnlineSharding MongoDB 37018" --
serviceDisplayName "BPBOnlineSharding MongoDB 37018" --
serviceDescription "MongoDB on Port 37018 BPBOnlineShard1-
BPBOnlineReplicaSet1-BPBOnlineDBS2"

```

After you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:

The screenshot shows an Administrator Command Prompt window. The command entered is:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --replSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS2" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS2\mongod.log" --port 37018 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 37018" --serviceDisplayName "BPBOnlineSharding MongoDB 37018" --serviceDescription "MongoDB on Port 37018 BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS2"
```

**Figure 19.15: Adding the new MongoDB Windows Service - "mongod" with Replication parameters**

- Now, type the following command to start this Windows service from your command prompt already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 37018"
```

The screenshot shows an Administrator Command Prompt window. The command entered is:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --replSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS2" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS2\mongod.log" --port 37018 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 37018" --serviceDisplayName "BPBOnlineSharding MongoDB 37018" --serviceDescription "MongoDB on Port 37018 BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS2"
```

Then, the command to start the service is:

```
c:\Program Files\MongoDB\Server\4.4\bin>net start "BPBOnlineSharding MongoDB 37018"
```

The output shows the service starting successfully:

```
The BPBOnlineSharding MongoDB 37018 service is starting...  
The BPBOnlineSharding MongoDB 37018 service was started successfully.
```

**Figure 19.16: Starting the new MongoDB Windows Service**

- Now, we will start our third server with the help of the `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```

mongod --replSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB-
S3" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB-
S3\mongod.log" --port 37019 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 37019" --
serviceDisplayName "BPBOnlineSharding MongoDB 37019" --

```

```
serviceDescription "MongoDB on Port 37019 BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS3"
```

This command needs to run in the command prompt in the administrator mode. Here, the parameters given in the mongod command are described as follows:

```
--dbpath= Path of our DB Directory Location  
--logpath= Path of our DB Log File Location  
--port = Port Number for MongoDB Instance, in our case we  
will use "37019"  
--replSet= In our case we will be using  
"BPBOnlineReplicaSet1", which is common for all 3 Member  
Nodes or Instances in the Replica Set 1 under the MongoDB  
Sharded Environment 1 "BPBOnlineShard1".  
--shardsvr = MongoDB Option for Sharding (Sharded  
Environment)  
--install = To Install the MongoDB as Service  
--serviceName= Name of the Windows Service, in our case we  
will be using "BPBOnlineSharding MongoDB 37019"  
--serviceDisplayName= Display Name of the Windows Service, in  
our case we will be using "BPBOnlineSharding MongoDB 37019"  
--serviceDescription= Description of the Windows Service, in  
our case we will be using "MongoDB on Port 37019  
BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS3"
```

15. Now, open the command prompt in the administrator mode and navigate to the bin directory of your MongoDB installation path, which is C:\Program Files\MongoDB\Server\4.4\bin, in our case, as shown in the following screenshot:



**Figure 19.17: Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory**

16. Now, type the following command with all the parameters that we will create Windows service for our third server in the replica set as follows:

```
mongod --repSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineD
BS3" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB
S3\mongod.log" --port 37019 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 37019" --
serviceDisplayName "BPBOnlineSharding MongoDB 37019" --
serviceDescription "MongoDB on Port 37019 BPBOnlineShard1-
BPBOnlineReplicaSet1-BPBOnlineDBS3"
```

After you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:



```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --repSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS3" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS3\mongod.log" --port 37019 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 37019" --serviceDisplayName "BPBOnlineSharding MongoDB 37019" --serviceDescription "MongoDB on Port 37019 BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS3"
c:\Program Files\MongoDB\Server\4.4\bin>
```

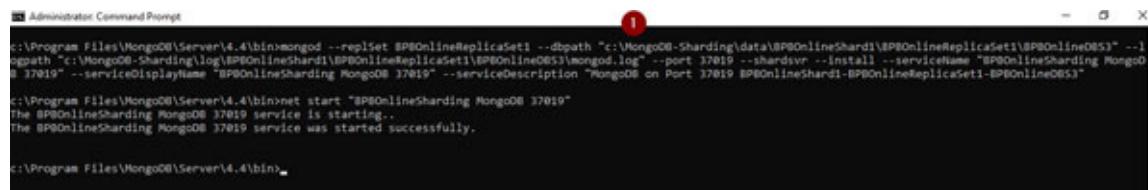
**Figure 19.18: Adding the new MongoDB Windows Service - "mongod" with Replication parameters**

- Now, type the following command to start this Windows service from your command prompt already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 37019"
```



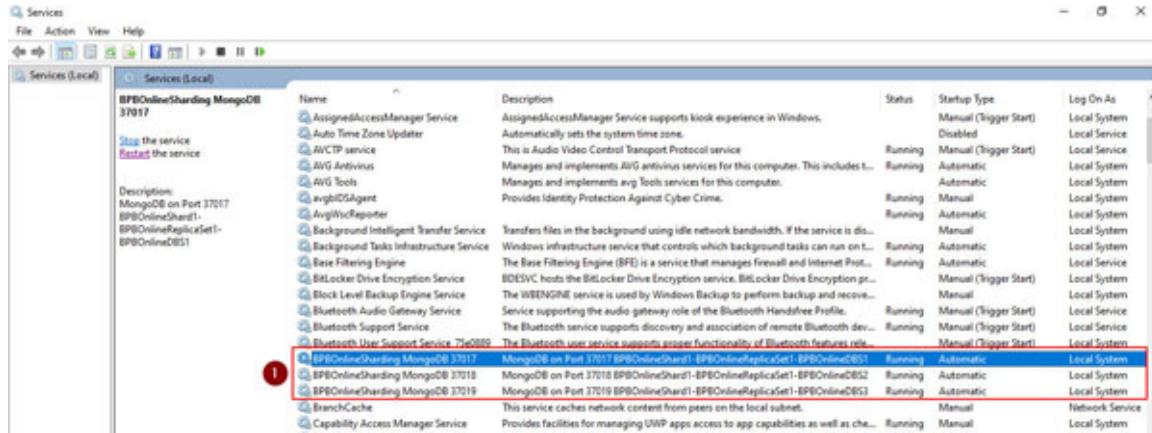
```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --repSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS3" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDBS3\mongod.log" --port 37019 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 37019" --serviceDisplayName "BPBOnlineSharding MongoDB 37019" --serviceDescription "MongoDB on Port 37019 BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS3"
c:\Program Files\MongoDB\Server\4.4\bin>net start "BPBOnlineSharding MongoDB 37019"
The BPBOnlineSharding MongoDB 37019 service is starting...
The BPBOnlineSharding MongoDB 37019 service was started successfully.

c:\Program Files\MongoDB\Server\4.4\bin>
```

**Figure 19.19: Starting the new MongoDB Windows Service**

- Now, as our all the three MongoDB server `mongod` instances have successfully started, we can open the `Windows Service Manager` (as explained in the previous step of this chapter, where we have explained the process of opening the Windows service manager using the Windows search box) and then navigate to the service named, `BPBOnlineSharding MongoDB<Port Number>`. You will see

that all the MongoDB instances are created in the background and their status should be **Running**, as shown in the following screenshot:



**Figure 19.20: Windows Service Manager – MongoDB Replica Set Instances (Running in Background)**

- Now, open the existing command prompt (which is already open in the administrator mode). If you are not already in the `bin` directory of your MongoDB installation path, first navigate to it, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, and then type the following command:

```
mongo --port 37017
```

This command will connect us to the MongoDB Shell, as shown in the following screenshot:

The screenshot shows an Administrator Command Prompt window. A red circle labeled '1' highlights the command `mongo --port 37017`. Another red circle labeled '2' highlights the output of the command, which shows the MongoDB shell version 4.4.1 connecting to the database at port 37017. The output also includes startup warnings about access control and monitoring.

```
c:\>Administrator: Command Prompt - mongo --port 37017
C:\Program Files\MongoDB\Server\4.4\bin>mongo --port 37017
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:37017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("03186ac2-7bd3-4300-aeb5-1437ce198d") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings:
2020-10-26T20:22:55.873+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-10-26T20:22:55.873+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the 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()
```

**Figure 19.21: Connected to the MongoDB Shell**

- Now, we need to initiate the replication process, and for this, we need to add all the member details with their host and port. In our

case, we will create a variable named `BPBOnlineReplicaSet1Config` with the `_id` same as our replica set, which is `BPBOnlineReplicaSet1`, and the member details have the `_id` 0, 1, and 2 (for the first, second, and third members in the replica set), and their host and port values, which are `localhost:37017`, `localhost:37018`, and `localhost:37019` (for the first, second, and third members in the replica set). In our case, it is as follows.

```
BPBOnlineReplicaSet1Config = {  
    _id: "BPBOnlineReplicaSet1",  
    members: [{  
        _id: 0,  
        host: "localhost:37017"  
    },  
    {  
        _id: 1,  
        host: "localhost:37018"  
    },  
    {  
        _id: 2,  
        host: "localhost:37019"  
    }  
};
```

Enter the preceding code in the MongoDB Shell and you will get the result similar to the screenshot shown in the following screenshot:

```

Administrator: Command Prompt - mongo --port 37017
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()

> BPBOnlineReplicaSet1Config = {
...   "_id": "BPBOnlineReplicaSet1",
...   "members": [
...     {
...       "_id": 0,
...       "host": "localhost:37017"
...     },
...     {
...       "_id": 1,
...       "host": "localhost:37018" 1
...     },
...     {
...       "_id": 2,
...       "host": "localhost:37019"
...     }
...   ],
...   "_id": "BPBOnlineReplicaSet1",
...   "members": [
...     {
...       "_id": 0,
...       "host": "localhost:37017"
...     },
...     {
...       "_id": 1,
...       "host": "localhost:37018" 2
...     },
...     {
...       "_id": 2,
...       "host": "localhost:37019"
...     }
...   ]
}

```

**Figure 19.22: MongoDB Shell – Replication Config Variable**

21. Now, we can use the variable `BPBOnlineReplicaSet1Config` during the initialization command for the replica set creation. Type the following command to initiate the replication process:

```
rs.initiate(BPBOnlineReplicaSet1Config)
```

Enter the preceding command in the MongoDB Shell and you will get the result similar to the screenshot shown in the following screenshot:

```

Administrator: Command Prompt - mongo --port 37017
members: [
...   {
...     "_id": 0,
...     "host": "localhost:37017"
...   },
...   {
...     "_id": 1,
...     "host": "localhost:37018"
...   },
...   {
...     "_id": 2,
...     "host": "localhost:37019"
...   }
... ],
... "_id": "BPBOnlineReplicaSet1",
... "members": [
...   {
...     "_id": 0,
...     "host": "localhost:37017"
...   },
...   {
...     "_id": 1,
...     "host": "localhost:37018"
...   },
...   {
...     "_id": 2,
...     "host": "localhost:37019"
...   }
... ]
1
> rs.initiate(BPBOnlineReplicaSet1Config) 1
2
{
  "ok": 1,
  "$clusterTime": {
    "clusterTime": Timestamp(1603726533, 1),
    "signature": {
      "hash": BinData(0, "AAAAAAAAAAAAAAAAAAAAAA-"),
      "keyId": NumberLong(0)
    }
  },
  "operationTime": Timestamp(1603726533, 1)
}

```

BPBOnlineReplicaSet1:SECONDARY>

**Figure 19.23: Initializing the Replication with "rs.initiate()" Command in MongoDB Shell**

22. If you see carefully in our MongoDB Shell, you will find that your MongoDB prompt has changed to `BPBOnlineReplicaSet1:SECONDARY>`.

Usually, it takes some time to decide for the primary and it can be any member chosen to be the primary member of the replica set. But usually, the first member in the configuration file becomes the **Primary Member**. This process can take a few seconds. So, if you just press the *Enter* key without typing anything, this could change the prompt to `BPBOnlineReplicaSet1:PRIMARY>` and you can get the result similar to the screenshot shown in the following screenshot:



The screenshot shows a terminal window titled "Administrator: Command Prompt - mongo --port 37019". The command entered is `rs.initiate(BPBOnlineReplicaSet1Config)`. The response indicates success with "ok": 1. It includes cluster metadata like `clusterTime`, `signature`, and `operationTime`. At the bottom of the terminal, the prompt changes from `BPBOnlineReplicaSet1:SECONDARY>` to `BPBOnlineReplicaSet1:PRIMARY>`, with a red circle highlighting the change.

**Figure 19.24: Changing the MongoDB Shell from SECONDARY to PRIMARY with an Enter key**

23. We can verify the status of our replica set using the following command:

```
rs.status()
```

Just enter the preceding command in your MongoDB Shell of the primary instance and you will get the result similar to the following screenshot. You will get the information about the replica set members and their status, which is "o", in our case, as shown in the following screenshot:

```

Administrator: Command Prompt - mongo --port 37017
{
    "pingMs" : NumberLong(0),
    "lastHeartbeatMessage" : "",
    "syncSourceHost" : "localhost:37017",
    "syncSourceId" : 0,
    "infoMessage" : "",
    "configVersion" : 1,
    "configTerm" : -1
},
{
    "_id" : 2,
    "name" : "localhost:37019",
    "health" : 1,
    "state" : 2,
    "stateStr" : "SECONDARY",
    "uptime" : 382,
    "optime" : {
        "ts" : Timestamp(1603726904, 1),
        "t" : NumberLong(1)
    },
    "optimeDurable" : {
        "ts" : Timestamp(1603726904, 1),
        "t" : NumberLong(1)
    },
    "optimeDate" : ISODate("2020-10-26T15:41:44Z"),
    "optimeDurableDate" : ISODate("2020-10-26T15:41:44Z"),
    "lastHeartbeat" : ISODate("2020-10-26T15:41:54.863Z"),
    "lastHeartbeatKey" : ISODate("2020-10-26T15:41:55.882Z"),
    "pingMs" : NumberLong(0),
    "lastHeartbeatMessage" : "",
    "syncSourceHost" : "localhost:37017",
    "syncSourceId" : 0,
    "infoMessage" : "",
    "configVersion" : 1,
    "configTerm" : -1
},
],
"ok" : 1,
"clusterTime" : {
    "clusterTime" : Timestamp(1603726914, 1),
    "signature" : {
        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA="),
        "keyId" : NumberLong(0)
    }
}
}

```

**Figure 19.25: "rs.status()" – Checking the status of the Replica Set**

So, we have successfully created our first shard with the replica set of three members. Now, let us repeat this entire process to create the other two shards having the replica set of three members each.

### **Step 3 – Create the second replica set (our second shard)**

Let us create the second replica set for the MongoDB sharded environment. To do this, we need to perform the following steps:

1. We need to create few folders to keep the MongoDB instances data and log files. Now, instead of doing it manually one-by-one, we will create these folders with the help of few commands which will automatically create these folders:

**For MongoDB data files:**

```

mkdir "c:\MongoDB-
Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineD
BS1" "c:\MongoDB-
Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineD
BS2" "c:\MongoDB-
Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineD
BS3"

```

## For MongoDB log files:

```
mkdir "c:\MongoDB-
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB
S1" "c:\MongoDB-
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB
S2" "c:\MongoDB-
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB
S3"
```

If you look this closely, you will find that here, our main directory is `MongoDB-Sharding` under the `c:` drive of your Windows machine (you may change the name and directory according to your wish).

Then, under the `MongoDB-Sharding` directory, there would be 2 child directories, `data`, which will keep the MongoDB data files, and `log`, which will keep the MongoDB log files.

So, till now, the structure will be as follows:

- `c:\MongoDB-Sharding\data`
- `c:\MongoDB-Sharding\log`

In both the `data` and `log` directories we will have two common types of directories, which would be `BPBOnlineShard2` (which is the name of our second shard), and under this directory there would be another directory, `BPBOnlineReplicaSet2` (which is the name of our second replica set). At last, under the `BPBOnlineReplicaSet2` directory, we will have 3 different directories which will cater to each one of the 3 MongoDB instances in the replica set in these directories as follows:

For MongoDB data files (complete path):

- <Main  
Directory>\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS1
- <Main  
Directory>\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS2
- <Main  
Directory>\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS3

```
nlineDBS3
```

### For MongoDB log files (complete path):

- <Main Directory>\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS1
- <Main Directory>\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS2
- <Main Directory>\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS3

Here:

- <Main Directory> = c:\MongoDB-Sharding\
- "BPBOnlineDBS1", "BPBOnlineDBS2" and "BPBOnlineDBS3" are the names of the Database Server Directories

If you closely look at the directory structure, it looks perfect to create a second replica set in the sharded environment.

2. Let us first create the new directories which will be the data directory for our MongoDB instances. To do this, first open the Windows command prompt in the administrator mode and then enter the following command, as shown in the following screenshot:

```
mkdir "c:\MongoDB-Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS1" "c:\MongoDB-Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS2" "c:\MongoDB-Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS3"
```



**Figure 19.26: MongoDB Data Folders - Creating new Folders**

3. After you have created new directories, you should see them under the "c:" drive, similar to the screenshot, as shown in the following screenshot:



**Figure 19.27: MongoDB Data Folders – New Folders gets created**

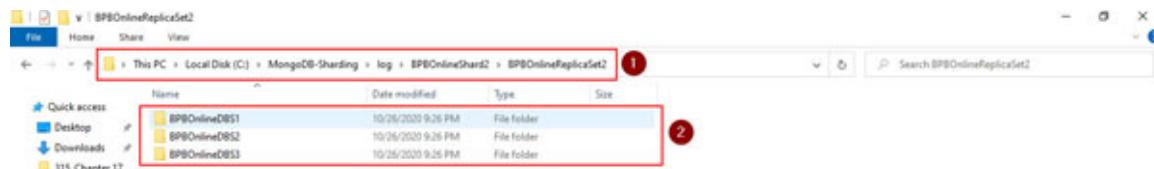
4. Now, let us first create the new directories which would be the log directories for our MongoDB instances. To do this, first open the Windows command prompt in the administrator mode and then enter the following command, as shown in the following screenshot:

```
mkdir "c:\MongoDB-
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB
S1" "c:\MongoDB-
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB
S2" "c:\MongoDB-
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB
S3"
```



**Figure 19.28: MongoDB Log Folders - Creating new Folders**

5. After you created new directories, you should see them under the c: drive, similar to the screenshot shown in the following screenshot:



**Figure 19.29: MongoDB Log Folders – New Folders gets created**

6. Now, we will start our first server with the help of `mongod` command by giving few parameters as follows (note that here, we will run

this as Windows service as explained in the previous section of this chapter):

```
mongod --replSet BPBOnlineReplicaSet2 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS1\mongod.log" --port 47017 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 47017" --serviceDisplayName "BPBOnlineSharding MongoDB 47017" --serviceDescription "MongoDB on Port 47017 BPBOnlineShard2-BPBOnlineReplicaSet2-BPBOnlineDBS1"
```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the `mongod` command are as follows:

```
--dbpath= Path of our DB Directory Location  
--logpath= Path of our DB Log File Location  
--port = Port Number for MongoDB Instance, in our case we will use "47017"  
--replSet= In our case we will be using "BPBOnlineReplicaSet2", which is common for all 3 Member Nodes or Instances in the Replica Set 2 under the MongoDB Sharded Environment 2 "BPBOnlineShard2".  
--shardsvr = MongoDB Option for Sharding (Sharded Environment)  
--install = To Install the MongoDB as Service  
--serviceName= Name of the Windows Service, in our case we will be using "BPBOnlineSharding MongoDB 47017"  
--serviceDisplayName= Display Name of the Windows Service, in our case we will be using "BPBOnlineSharding MongoDB 47017"  
--serviceDescription= Description of the Windows Service, in our case we will be using "MongoDB on Port 47017 BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS1"
```

7. Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following screenshot:



**Figure 19.30:** Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory

- Now type the following command with all the parameters that we will create Windows service for our first server in the replica set as follows:

```
mongod --replSet BPBOnlineReplicaSet1 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineD
BS1" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard1\BPBOnlineReplicaSet1\BPBOnlineDB
S1\mongod.log" --port 37017 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 37017" --
serviceDisplayName "BPBOnlineSharding MongoDB 37017" --
serviceDescription "MongoDB on Port 37017 BPBOnlineShard1-
BPBOnlineReplicaSet1-BPBOnlineDBS1"
```

Once you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:



**Figure 19.31:** Adding the new MongoDB Windows Service - "mongod" with Replication parameters

- Now, type the following command to start this Windows service from your command prompt which is already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 47017"
```

```

Administrator: Command Prompt
c:\Program Files\MongoDB\Server\4.4\bin>mongod --replSet BPBOnlineReplicaSet2 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDBS1\mongod.log" --port 47017 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 47017" --serviceDescription "MongoDB on Port 47017 BPBOnlineShard2-BPBOnlineReplicaSet2-BPBOnlineDBS1"
c:\Program Files\MongoDB\Server\4.4\bin>net start "BPBOnlineSharding MongoDB 47017"
the BPBOnlineSharding MongoDB 47017 service is starting...
the BPBOnlineSharding MongoDB 47017 service was started successfully.

c:\Program Files\MongoDB\Server\4.4\bin>

```

**Figure 19.32:** Starting the new MongoDB Windows Service

10. Now, we will start our second server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```

mongod --replSet BPBOnlineReplicaSet2 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB-
S2" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB-
S2\mongod.log" --port 47018 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 47018" --
serviceDisplayName "BPBOnlineSharding MongoDB 47018" --
serviceDescription "MongoDB on Port 47018 BPBOnlineShard2-
BPBOnlineReplicaSet2-BPBOnlineDBS2"

```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the `mongod` command are as follows:

```

--dbpath= Path of our DB Directory Location
--logpath= Path of our DB Log File Location
--port = Port Number for MongoDB Instance, in our case we
will use "47018"
--replSet= In our case we will be using
"BPBOnlineReplicaSet2", which is common for all 3 Member
Nodes or Instances in the Replica Set 2 under the MongoDB
Sharded Environment 2 "BPBOnlineShard2".
--shardsvr = MongoDB Option for Sharding (Sharded
Environment)
--install = To Install the MongoDB as Service
--serviceName= Name of the Windows Service, in our case we
will be using "BPBOnlineSharding MongoDB 47018"
--serviceDisplayName= Display Name of the Windows Service, in
our case we will be using "BPBOnlineSharding MongoDB 47018"

```

```
--serviceDescription= Description of the Windows Service, in  
our case we will be using "MongoDB on Port 47018  
BPBOnlineShard2-BPBOnlineReplicaSet2-BPBOnlineDBS2"
```

- Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following screenshot:



**Figure 19.33:** Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory

- Now, type the following command with all the parameters that we will create Windows service for our second server in the replica set as follows:

```
mongod --replSet BPBOnlineReplicaSet2 --dbpath "c:\MongoDB-  
Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineD  
BS2" --logpath "c:\MongoDB-  
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB  
S2\mongod.log" --port 47018 --shardsvr --install --  
serviceName "BPBOnlineSharding MongoDB 47018" --  
serviceDisplayName "BPBOnlineSharding MongoDB 47018" --  
serviceDescription "MongoDB on Port 47018 BPBOnlineShard2-  
BPBOnlineReplicaSet2-BPBOnlineDBS2"
```

Once you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:



**Figure 19.34:** Adding the new MongoDB Windows Service - "mongod" with Replication parameters

- Now, type the following command to start this Windows service from your command prompt which is already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 47018"
```

*Figure 19.35: Starting the new MongoDB Windows Service*

14. Now, we will start our third server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```
mongod --replSet BPBOnlineReplicaSet2 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineD
BS3" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB
S3\mongod.log" --port 47019 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 47019" --
serviceDisplayName "BPBOnlineSharding MongoDB 47019" --
serviceDescription "MongoDB on Port 47019 BPBOnlineShard2-
BPBOnlineReplicaSet2-BPBOnlineDBS3"
```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the `mongod` command are as follows:

```
--dbpath= Path of our DB Directory Location
--logpath= Path of our DB Log File Location
--port = Port Number for MongoDB Instance, in our case we
will use "47019"
--replSet= In our case we will be using
"BPBOnlineReplicaSet2", which is common for all 3 Member
Nodes or Instances in the Replica Set 2 under the MongoDB
Sharded Environment 2 "BPBOnlineShard2".
--shardsvr = MongoDB Option for Sharding (Sharded
Environment)
```

```
--install = To Install the MongoDB as Service
--serviceName= Name of the Windows Service, in our case we
will be using "BPBOnlineSharding MongoDB 47019"
--serviceDisplayName= Display Name of the Windows Service, in
our case we will be using "BPBOnlineSharding MongoDB 47019"
--serviceDescription= Description of the Windows Service, in
our case we will be using "MongoDB on Port 47019
BPBOnlineShard2-BPBOnlineReplicaSet2-BPBOnlineDBS3"
```

- Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following screenshot:



**Figure 19.36:** Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory

- Now, type the following command with all the parameters that we will create a Windows service for our third server in the replica set as follows:

```
mongod --replSet BPBOnlineReplicaSet2 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineD
BS3" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnlineDB
S3\mongod.log" --port 47019 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 47019" --
serviceDisplayName "BPBOnlineSharding MongoDB 47019" --
serviceDescription "MongoDB on Port 47019 BPBOnlineShard2-
BPBOnlineReplicaSet2-BPBOnlineDBS3"
```

Once you enter this command, you will get a response something similar to the screenshot, as shown in the following screenshot:



**Figure 19.37: Adding the new MongoDB Windows Service - "mongod" with Replication parameters**

- Now, type the following command to start this Windows service from your command prompt already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

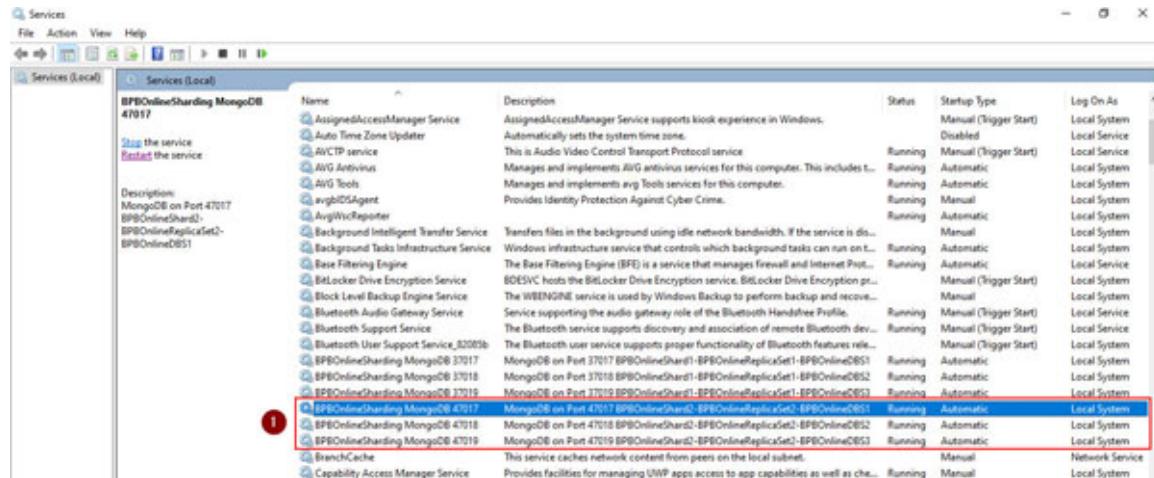
In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 47019"
```

```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --replicaSet BPBOnlineReplicaSet2 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnline0853" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard2\BPBOnlineReplicaSet2\BPBOnline0853\mongod.log" --port 47019 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 47019" --serviceDescription "MongoDB on Port 47019 BPBOnlineShard2-BPBOnlineReplicaSet2-BPBOnline0853"
c:\Program Files\MongoDB\Server\4.4\bin>net start "BPBOnlineSharding MongoDB 47019" !
The BPBOnlineSharding MongoDB 47019 service is starting...
The BPBOnlineSharding MongoDB 47019 service was started successfully. 2
c:\Program Files\MongoDB\Server\4.4\bin>
```

**Figure 19.38: Starting the new MongoDB Windows Service**

- Now, as all our three MongoDB server `mongod` instances have successfully started, we can open the `Windows Service Manager` (as explained in the previous step of this chapter where we have explained the process of opening the Windows service manager using the Windows search box), and then navigate to the services named `BPBOnlineSharding MongoDB<Port Number>`. You will see that all the MongoDB instances are created in background and their status should be `Running`, as shown in the following screenshot:

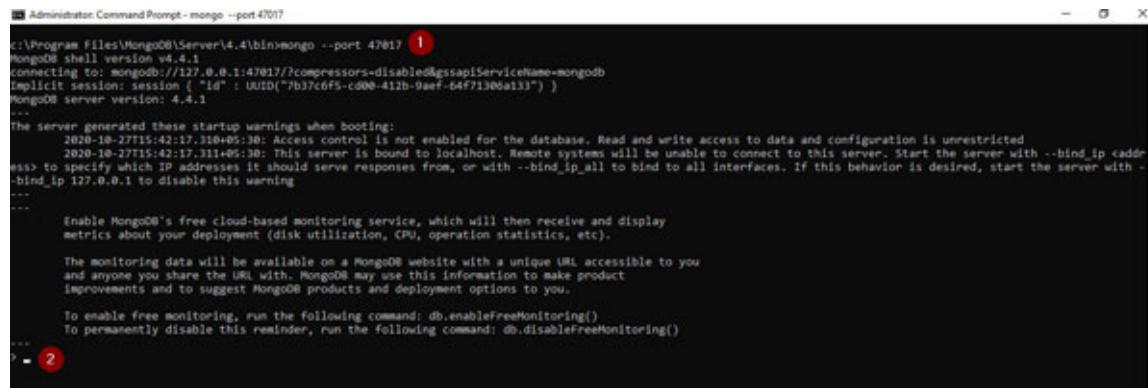


**Figure 19.39: Windows Service Manager – MongoDB Replica Set Instances (Running in Background)**

19. Now, open the existing command prompt (which is already open in the administrator mode). If you are not already in the `bin` directory of your MongoDB installation path, first navigate to it, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case and then type the following command:

```
mongo --port 47017
```

This command will connect us to the MongoDB Shell, as shown in the following screenshot:

A screenshot of an Administrator Command Prompt window titled "Administrator: Command Prompt - mongo --port 47017". The window shows the MongoDB shell version 4.4.1 starting up. It displays several startup warnings and informational messages, including: "Access control is not enabled for the database. Read and write access to data and configuration is unrestricted", "This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind\_ip <address> to specify which IP addresses it should serve responses from, or with -bind\_ip\_all to bind to all interfaces. If this behavior is desired, start the server with -bind\_ip 127.0.0.1 to disable this warning", and instructions for enabling free monitoring. A red circle labeled '1' is drawn around the title bar, and another red circle labeled '2' is drawn around the bottom-left corner of the window.

**Figure 19.40: Connected to the MongoDB Shell**

20. Now, we need to initiate the replication process. For this, we need to add all the member details with their host and port. In our case, we will create a variable named `BPBOnlineReplicaSet2Config` with the `_id` same as our replica set, which is `BPBOnlineReplicaSet2`, and the member details have the `_id` 0, 1, and 2 (for the first, second, and third members in the replica set) and their host and port values which are `localhost:47017`, `localhost:47018`, and `localhost:47019` (for the first, second, and third members in the replica set). In our case, it is as follows:

```
BPBOnlineReplicaSet2Config = {
  _id: "BPBOnlineReplicaSet2",
  members: [
    {
      _id: 0,
      host: "localhost:47017"
    },
    {
      _id: 1,
      host: "localhost:47018"
    }
  ]
}
```

```

    },
    {
        _id: 2,
        host: "localhost:47019"
    }
]
};

```

Enter the preceding code in the MongoDB Shell and you will get the result similar to the screenshot shown in the following screenshot:

```

Administrator: Command Prompt - mongo --port 47017
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()

> BPBOnlineReplicaSet2Config = {
...     _id: "BPBOnlineReplicaSet2",
...     members: [
...         {
...             _id: 0,
...             host: "localhost:47017"
...         },
...         {
...             _id: 1,          ①
...             host: "localhost:47018"
...         },
...         {
...             _id: 2,          ②
...             host: "localhost:47019"
...         }
...     ]
... }

{
    "_id": "BPBOnlineReplicaSet",
    "members": [
        {
            "_id": 0,
            "host": "localhost:47017"
        },
        {
            "_id": 1,          ②
            "host": "localhost:47018"
        },
        {
            "_id": 2,
            "host": "localhost:47019"
        }
    ]
}

```

**Figure 19.41: MongoDB Shell – Replication Config Variable**

21. Now, we can use the variable `BPBOnlineReplicaSet2Config` during the initialization command for the replica set creation. Type the following command to initiate the replication process:

```
rs.initiate(BPBOnlineReplicaSet2Config)
```

Enter the preceding command in the MongoDB Shell and you will get the result similar to the screenshot shown in the following screenshot:

```
Administrator: Command Prompt - mongo --port 47017
...
members: [
    {
        "_id": 0,
        "host": "localhost:47017"
    },
    {
        "_id": 1,
        "host": "localhost:47018"
    },
    {
        "_id": 2,
        "host": "localhost:47019"
    }
]
};

"_id": "BPBOnlineReplicaSet2",
"members": [
    {
        "_id": 0,
        "host": "localhost:47017"
    },
    {
        "_id": 1,
        "host": "localhost:47018"
    },
    {
        "_id": 2,
        "host": "localhost:47019"
    }
]
}

rs.initiate(BPBOnlineReplicaSet2Config) 1
{
    "ok": 1,
    "$clusterTime": {
        "clusterTime": Timestamp(1603793983, 1),
        "signature": {
            "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
            "keyId": NumberLong(0)
        }
    },
    "operationTime": Timestamp(1603793983, 1)
}
BPBOnlineReplicaSet2:SECONDARY>
```

**Figure 19.42:** Initializing the Replication with "rs.initiate()" Command in MongoDB Shell

22. If you see carefully in our MongoDB Shell, you will find that your MongoDB prompt has changed to BPBOnlineReplicaSet2:SECONDARY>.

Usually, it takes some time to decide for the primary and it could be any member chosen to be the primary member of the replica set. But usually, the first member in the configuration file becomes the **Primary Member**. This process can take a few seconds. So, if you just press the *Enter* key without typing anything, this could change the prompt to `BPBOnlineReplicaSet2:PRIMARY>` and you can get the result similar to the screenshot shown in the following screenshot:



The screenshot shows the MongoDB shell running on port 47017. The command `rs.initiate({})` is being run to set up a replica set named 'RPBOnlineReplicaSet2'. The configuration includes three members: one primary at 'localhost:47019' and two secondaries at 'localhost:47017' and 'localhost:47018'. After the initiation, the status is checked again, and the output shows the primary has changed to 'localhost:47019' (SECONDARY) and the original primary is now (PRIMARY). A red box highlights the status change.

```

Administrator: Command Prompt - mongo --port 47017
...
...
...
{
  "_id" : "RPBOnlineReplicaSet2",
  "members" : [
    {
      "_id" : 0,
      "host" : "localhost:47017"
    },
    {
      "_id" : 1,
      "host" : "localhost:47018"
    },
    {
      "_id" : 2,
      "host" : "localhost:47019"
    }
  ]
}
rs.initiate({})

"ok" : 1,
"$clusterTime" : {
  "clusterTime" : Timestamp(1603793903, 1),
  "signature" : {
    "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
    "keyId" : NumberLong(0)
  }
},
"operationTime" : Timestamp(1603793903, 1)

RPBOnlineReplicaSet2:SECONDARY>
RPBOnlineReplicaSet2:PRIMARY> 

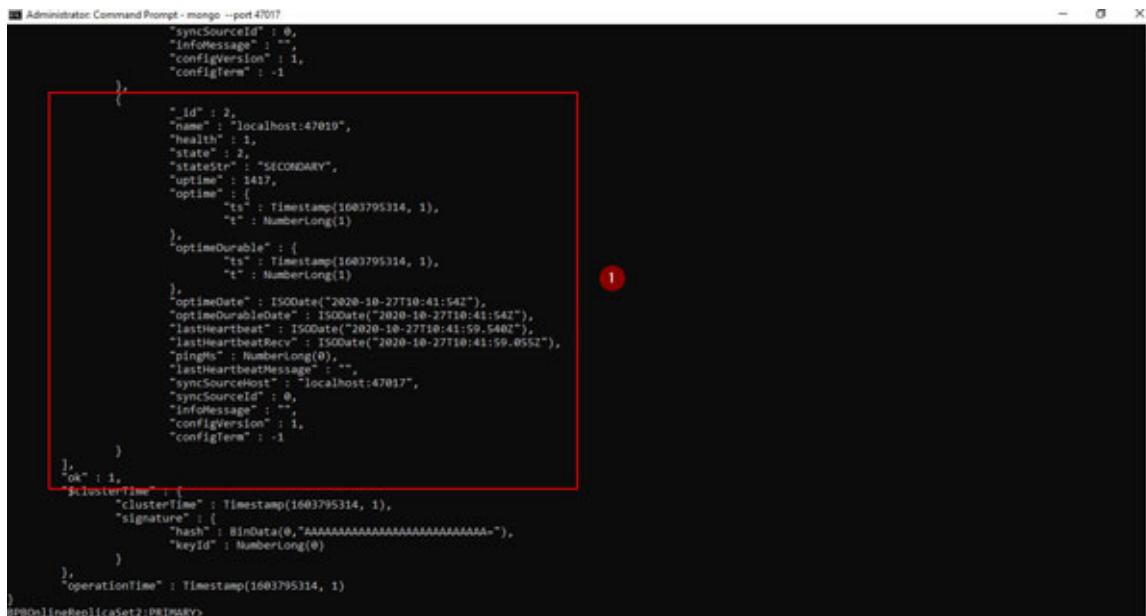
```

**Figure 19.43:** Changing the MongoDB Shell from SECONDARY to PRIMARY with an Enter key

23. We can verify the status of our replica set using the following command:

```
rs.status()
```

Just enter the preceding command in your MongoDB Shell of the primary instance and you will get the result similar to the following screenshot. You will get the information about the replica set members and their status, which is `ok`, in our case, as shown in the following screenshot:



The screenshot shows the MongoDB shell running on port 47017. The command `rs.status()` is run, and the output displays the status of the replica set members. The member at 'localhost:47019' is listed as 'SECONDARY' with a status of 'ok'. A red box highlights this secondary member. The member at 'localhost:47017' is the primary, indicated by the status 'PRIMARY'.

```

Administrator: Command Prompt - mongo --port 47017
...
...
{
  "syncSourceId" : 0,
  "infoMessage" : "",
  "configVersion" : 1,
  "configTerm" : -1
},
{
  "_id" : 2,
  "name" : "localhost:47019",
  "health" : 1,
  "state" : 2,
  "stateStr" : "SECONDARY",
  "uptime" : 1437,
  "optime" : {
    "ts" : Timestamp(1603795314, 1),
    "t" : NumberLong(1)
  },
  "optimeDurable" : {
    "ts" : Timestamp(1603795314, 1),
    "t" : NumberLong(1)
  },
  "optimeDate" : ISODate("2020-10-27T10:41:54Z"),
  "optimeDurableDate" : ISODate("2020-10-27T10:41:54Z"),
  "lastHeartbeat" : ISODate("2020-10-27T10:41:59.549Z"),
  "lastHeartbeatRecv" : ISODate("2020-10-27T10:41:59.655Z"),
  "pingMs" : NumberLong(0),
  "lastHeartbeatMessage" : "",
  "syncSourceHost" : "localhost:47017",
  "syncSourceId" : 0,
  "infoMessage" : "",
  "configVersion" : 1,
  "configTerm" : -1
}
],
"ok" : 1,
"$clusterTime" : {
  "clusterTime" : Timestamp(1603795314, 1),
  "signature" : {
    "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
    "keyId" : NumberLong(0)
  }
},
"operationTime" : Timestamp(1603795314, 1)

RPBOnlineReplicaSet2:PRIMARY>

```

**Figure 19.44:** "rs.status()" – Checking the status of the Replica Set

So, we have successfully created our second shard with the replica set of three members. Now, let us repeat this entire process to create the third shard having the replica set of three members.

## **Step 4 – Create the third replica set (our third shard)**

Let us create the third replica set for the MongoDB sharded environment. To do this, we need to perform the following steps:

1. We need to create few folders which will keep the MongoDB instances data and log files. Now, instead of doing it manually one-by-one, we will create these folders with the help of few commands which will automatically create these folders:

For MongoDB data files:

```
mkdir "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
BS1" "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
BS2" "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
BS3"
```

For MongoDB log files:

```
mkdir "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S1" "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S2" "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S3"
```

If you look this closely, you can find that here our main directory is MongoDB-Sharding under the c: drive of your Windows machine (you may change the name and directory according to your wish).

Then, under the MongoDB-Sharding directory, there would be 2 child directories, data, which will keep the MongoDB data files, and log

which will keep the MongoDB log files. So, till now, the structure will be as follows:

- c:\MongoDB-Sharding\data
- c:\MongoDB-Sharding\log

Then, in both the `data` and `log` directories, we will have two common types of directories which would be, `BPBOnlineShard3` (which is the name of our third shard), and under this directory, there would be another directory, `BPBOnlineReplicaSet1` (which is the name of our third replica set).

At last, under the `BPBOnlineReplicaSet3` directory, we will have 3 different directories which will cater to each one of the 3 MongoDB instances in the replica set in these directories as follows:

For MongoDB data files (complete path):

- <Main  
Directory>\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS1
- <Main  
Directory>\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS2
- <Main  
Directory>\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS3

For MongoDB log files (complete path):

- <Main  
Directory>\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS1
- <Main  
Directory>\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS2
- <Main  
Directory>\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS3

Here:

- <Main Directory> = c:\MongoDB-Sharding\
- "BPBOnlineDBS1", "BPBOnlineDBS2" and "BPBOnlineDBS3" are the names of the Database Server Directories

If you closely look at the directory structure, it looks perfect to create a third replica set in the sharded environment.

2. Let us first create the new directories which will be the data directory for our MongoDB instances. To do this, first open the Windows command prompt in the administrator mode and then enter the following command, as shown in the following screenshot:

```
mkdir "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplica
Set3\BPBOnlineDBS1" "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
BS2" "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
BS3"
```



**Figure 19.45: MongoDB Data Folders - Creating new Folders**

3. After you have created new directories, you should see them under the c: drive, similar to the screenshot shown in the following screenshot:



**Figure 19.46: MongoDB Data Folders – New Folders gets created**

4. Now, let us first create the new directories which would be the <sup>log</sup> directories for our MongoDB instances. To do this, first open the Windows command prompt in the administrator mode and then enter the following command, as shown in the following screenshot:

```

mkdir "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplica
Set3\BPBOnlineDBS1" "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S2" "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S3"

```



**Figure 19.47: MongoDB Log Folders - Creating new Folders**

- After you have created new directories, you should see them under the `c:` drive, similar to the screenshot shown in the following screenshot:



**Figure 19.48: MongoDB Log Folders – New Folders gets created**

- Now, we will start our first server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```

mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
BS1" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S1\mongod.log" --port 57017 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 57017" --
serviceDisplayName "BPBOnlineSharding MongoDB 57017" --
serviceDescription "MongoDB on Port 57017 BPBOnlineShard3-
BPBOnlineReplicaSet3-BPBOnlineDBS1"

```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the `mongod` command are as follows:

```

--dbpath= Path of our DB Directory Location
--logpath= Path of our DB Log File Location
--port = Port Number for MongoDB Instance, in our case we
will use "57017"
--replSet= In our case we will be using
"BPBOnlineReplicaSet3", which is common for all 3 Member
Nodes or Instances in the Replica Set 3 under the MongoDB
Sharded Environment 3 "BPBOnlineShard3".
--shardsvr = MongoDB Option for Sharding (Sharded
Environment)
--install = To Install the MongoDB as Service
--serviceName= Name of the Windows Service, in our case we
will be using "BPBOnlineSharding MongoDB 57017"
--serviceDisplayName= Display Name of the Windows Service, in
our case we will be using "BPBOnlineSharding MongoDB 57017"
--serviceDescription= Description of the Windows Service, in
our case we will be using "MongoDB on Port 57017
BPBOnlineShard1-BPBOnlineReplicaSet1-BPBOnlineDBS1"

```

- Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following screenshot:



**Figure 19.49:** Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory

- Now, type the following command with all the parameters that we will create a Windows service for our first server in the replica set as follows:

```

mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
BS1" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S1\mongod.log" --port 57017 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 57017" --

```

```

serviceDisplayName "BPBOnlineSharding MongoDB 57017" --
serviceDescription "MongoDB on Port 57017 BPBOnlineShard3-
BPBOnlineReplicaSet3-BPBOnlineDBS1"

```

Once you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:



The screenshot shows an 'Administrator: Command Prompt' window. The command entered is:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS1\mongod.log" --port 57017 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 57017" --serviceDescription "MongoDB on Port 57017 BPBOnlineShard3-BPBOnlineReplicaSet3-BPBOnlineDBS1"
```

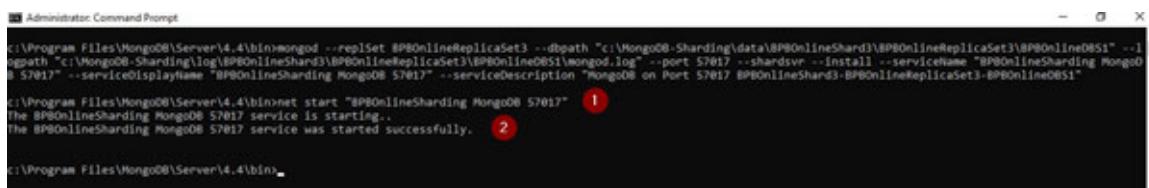
**Figure 19.50:** Adding the new MongoDB Windows Service - "mongod" with Replication parameters

- Now, type the following command to start this Windows service from your command prompt already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 57017"
```



The screenshot shows an 'Administrator: Command Prompt' window. The command entered is:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS1\mongod.log" --port 57017 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 57017" --serviceDescription "MongoDB on Port 57017 BPBOnlineShard3-BPBOnlineReplicaSet3-BPBOnlineDBS1"
```

Then, the command:

```
c:\Program Files\MongoDB\Server\4.4\bin>net start "BPBOnlineSharding MongoDB 57017"
```

is run, with a red circle labeled '1' above the command and another red circle labeled '2' above the output line:

```
! The BPBOnlineSharding MongoDB 57017 service is starting..  
The BPBOnlineSharding MongoDB 57017 service was started successfully.
```

**Figure 19.51:** Starting the new MongoDB Windows Service

- Now, we will start our second server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```

mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
BS2" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S2\mongod.log" --port 57018 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 57018" --
serviceDisplayName "BPBOnlineSharding MongoDB 57018" --

```

```
serviceDescription "MongoDB on Port 57018 BPBOnlineShard3-BPBOnlineReplicaSet3-BPBOnlineDBS2"
```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the `mongod` command are as follows:

```
--dbpath= Path of our DB Directory Location  
--logpath= Path of our DB Log File Location  
--port = Port Number for MongoDB Instance, in our case we  
will use "57018"  
--replSet= In our case we will be using  
"BPBOnlineReplicaSet3", which is common for all 3 Member  
Nodes or Instances in the Replica Set 3 under the MongoDB  
Sharded Environment 3 "BPBOnlineShard3".  
--shardsvr = MongoDB Option for Sharding (Sharded  
Environment)  
--install = To Install the MongoDB as Service  
--serviceName= Name of the Windows Service, in our case we  
will be using "BPBOnlineSharding MongoDB 57018"  
--serviceDisplayName= Display Name of the Windows Service, in  
our case we will be using "BPBOnlineSharding MongoDB 57018"  
--serviceDescription= Description of the Windows Service, in  
our case we will be using "MongoDB on Port 57018  
BPBOnlineShard3-BPBOnlineReplicaSet3-BPBOnlineDBS2"
```

11. Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following screenshot:



**Figure 19.52:** Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory

12. Now, type the following command with all the parameters that we will create a Windows service for our second server in the replica set as follows:

```

mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
BS2" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S2\mongod.log" --port 57018 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 57018" --
serviceDisplayName "BPBOnlineSharding MongoDB 57018" --
serviceDescription "MongoDB on Port 57018 BPBOnlineShard3-
BPBOnlineReplicaSet3-BPBOnlineDBS2"

```

After you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:

The screenshot shows an Administrator Command Prompt window. The command entered is:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS2" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS2\mongod.log" --port 57018 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 57018" --serviceDisplayName "BPBOnlineSharding MongoDB 57018" --serviceDescription "MongoDB on Port 57018 BPBOnlineShard3-BPBOnlineReplicaSet3-BPBOnlineDBS2"
```

**Figure 19.53: Adding the new MongoDB Windows Service - "mongod" with Replication parameters**

- Now, type the following command to start this Windows service from your command prompt already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 57018"
```

The screenshot shows an Administrator Command Prompt window. The command entered is:

```
c:\Program Files\MongoDB\Server\4.4\bin>net start "BPBOnlineSharding MongoDB 57018" ①
The BPBOnlineSharding MongoDB 57018 service is starting.. ②
The BPBOnlineSharding MongoDB 57018 service was started successfully.
```

**Figure 19.54: Starting the new MongoDB Windows Service**

- Now, we will start our third server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as a Windows service as explained in the previous section of this chapter):

```
mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-
Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineD
```

```
BS3" --logpath "c:\MongoDB-
Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDB
S3\mongod.log" --port 57019 --shardsvr --install --
serviceName "BPBOnlineSharding MongoDB 57019" --
serviceDisplayName "BPBOnlineSharding MongoDB 57019" --
serviceDescription "MongoDB on Port 57019 BPBOnlineShard3-
BPBOnlineReplicaSet3-BPBOnlineDBS3"
```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the mongod command are as follows:

```
--dbpath= Path of our DB Directory Location
--logpath= Path of our DB Log File Location
--port = Port Number for MongoDB Instance, in our case we
will use "57019"
--replSet= In our case we will be using
"BPBOnlineReplicaSet3", which is common for all 3 Member
Nodes or Instances in the Replica Set 3 under the MongoDB
Sharded Environment 3 "BPBOnlineShard3".
--shardsvr = MongoDB Option for Sharding (Sharded
Environment)
--install = To Install the MongoDB as Service
--serviceName= Name of the Windows Service, in our case we
will be using "BPBOnlineSharding MongoDB 57019"
--serviceDisplayName= Display Name of the Windows Service, in
our case we will be using "BPBOnlineSharding MongoDB 57019"
--serviceDescription= Description of the Windows Service, in
our case we will be using "MongoDB on Port 57019
BPBOnlineShard3-BPBOnlineReplicaSet3-BPBOnlineDBS3"
```

15. Now, open the command prompt in the administrator mode and navigate to the bin directory of your MongoDB installation path, which is C:\Program Files\MongoDB\Server\4.4\bin, in our case, as shown in the following screenshot:



**Figure 19.55:** Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory

- Now, type the following command with all the parameters that we will create Windows service for our third server in the replica set as follows:

```
mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS3" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS3\mongod.log" --port 57019 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 57019" --serviceDisplayName "BPBOnlineSharding MongoDB 57019" --serviceDescription "MongoDB on Port 57019 BPBOnlineShard3-BPBOnlineReplicaSet3-BPBOnlineDBS3"
```

Once you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:



The screenshot shows an Administrator Command Prompt window. The command entered is:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongod --replSet BPBOnlineReplicaSet3 --dbpath "c:\MongoDB-Sharding\data\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS3" --logpath "c:\MongoDB-Sharding\log\BPBOnlineShard3\BPBOnlineReplicaSet3\BPBOnlineDBS3\mongod.log" --port 57019 --shardsvr --install --serviceName "BPBOnlineSharding MongoDB 57019" --serviceDisplayName "BPBOnlineSharding MongoDB 57019" --serviceDescription "MongoDB on Port 57019 BPBOnlineShard3-BPBOnlineReplicaSet3-BPBOnlineDBS3"
```

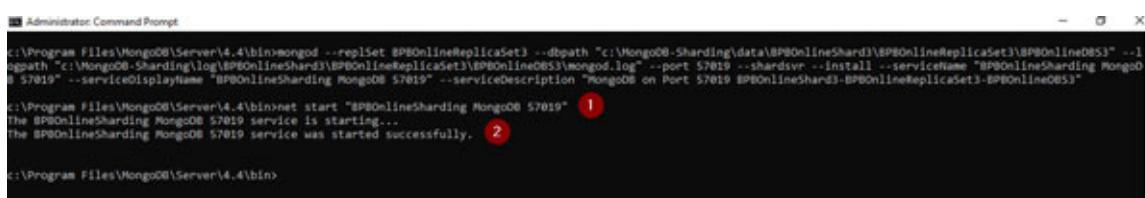
**Figure 19.56:** Adding the new MongoDB Windows Service - "mongod" with Replication parameters

- Now, type the following command to start this Windows service from your command prompt already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 57019"
```



The screenshot shows an Administrator Command Prompt window. The command entered is:

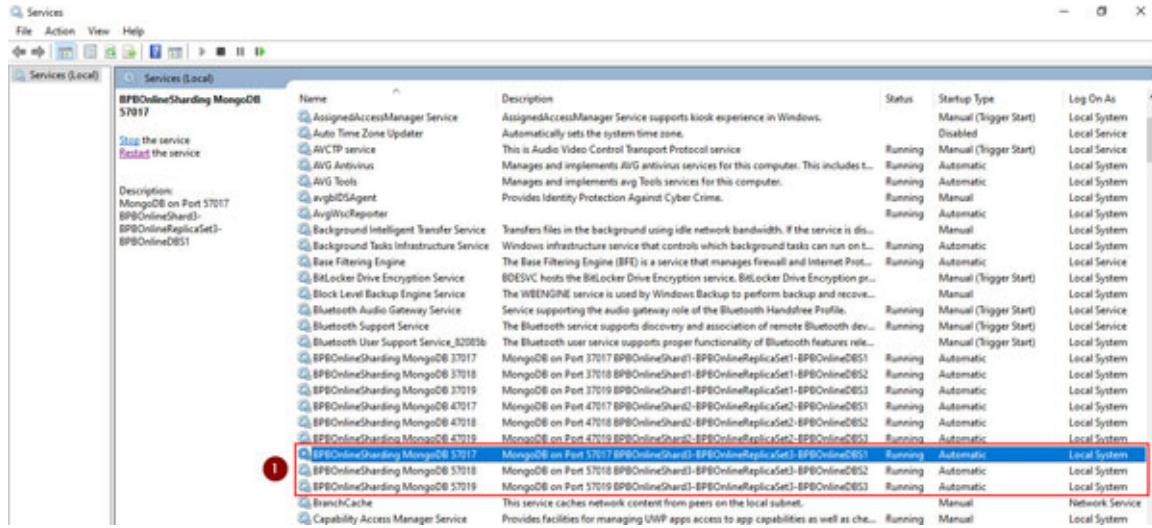
```
c:\Program Files\MongoDB\Server\4.4\bin>net start "BPBOnlineSharding MongoDB 57019" !
```

Output:

```
The BPBOnlineSharding MongoDB 57019 service is starting... . 2
The BPBOnlineSharding MongoDB 57019 service was started successfully.
```

**Figure 19.57:** Starting the new MongoDB Windows Service

18. Now, as all our three MongoDB server `mongod` instances have successfully started, we can open the "Windows Service Manager" (as explained in the previous step of this chapter where we have explained the process of opening the Windows service manager using the Windows search box), and then navigate to the service named `BPBOnlineSharding MongoDB<Port Number>`. You will see that all the MongoDB instances are created in the background and their status should be `Running`, as shown in the following screenshot:



**Figure 19.58: Windows Service Manager – MongoDB Replica Set Instances (Running in Background)**

19. Now, open the existing command prompt (which is already open in the administrator mode). If you are not already in the `bin` directory of your MongoDB installation path, first navigate to it, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case and then type the following command:

```
mongo --port 57017
```

This command will connect us to the MongoDB Shell, as shown in the following screenshot:

```

Administrator: Command Prompt - mongo --port 57017
c:\Program Files\MongoDB\Server\4.4\bin>mongo --port 57017 ①
mongodba shell version v4.4.1
connecting to: mongodb://127.0.0.1:57017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session "id": UUID("5fdseal-9892-4199-a8ee-cd5205d63fe")
mongodba server version: 4.4.1

The server generated these startup warnings when booting:
2020-10-27T19:36:41.554+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-10-27T19:36:41.555+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <addr> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the 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()

> ②

```

**Figure 19.59: Connected to the MongoDB Shell**

20. Now, we need to initiate the replication process. For this, we need to add all the member details with their host and port. In our case, we will create a variable named `BPBOnlineReplicaSet3Config` with the `_id` same as our replica set, which is `BPBOnlineReplicaSet3`, and the member details have the `_id` 0, 1, and 2 (for the first, second, and third members in the replica set), and their host and port values which are `localhost:57017`, `localhost:57018`, and `localhost:57019` (for the first, second, and third members in the replica set). In our case, it is as follows:

```

BPBOnlineReplicaSet3Config = {
    _id: "BPBOnlineReplicaSet3",
    members: [
        {
            _id: 0,
            host: "localhost:57017"
        },
        {
            _id: 1,
            host: "localhost:57018"
        },
        {
            _id: 2,
            host: "localhost:57019"
        }
    ]
};

```

Enter the preceding code in the MongoDB Shell and you will get the result similar to the screenshot shown in the following

## screenshot:

```
Administrator: Command Prompt - mongo --port 57017
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()

> BPBOnlineReplicaSet3Config = {
...   "_id": "BPBOnlineReplicaSet3",
...   "members": [
...     {
...       "_id": 0,
...       "host": "localhost:57017"
...     },
...     {
...       "_id": 1,
...       "host": "localhost:57018" ①
...     },
...     {
...       "_id": 2,
...       "host": "localhost:57019"
...     }
...   ]
... }

"_id": "BPBOnlineReplicaSet3",
"Members": [
  {
    "_id": 0,
    "Host": "localhost:57017"
  },
  {
    "_id": 1,
    "Host": "localhost:57018" ②
  },
  {
    "_id": 2,
    "Host": "localhost:57019"
  }
]
```

**Figure 19.60: MongoDB Shell – Replication Config Variable**

21. Now, we can use the variable `BPBOnlineReplicaSet3Config` during the initialization command for the replica set creation. Type the following command to initiate the replication process:

```
rs.initiate(BPBOnlineReplicaSet3Config)
```

Enter the preceding command in the MongoDB Shell and you will get the result similar to the screenshot shown in the following screenshot:

The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - mongo --port 57017". The shell output is as follows:

```

Administrator: Command Prompt - mongo --port 57017
...
{
  "_id": "BPBOnlineReplicaSet3",
  "members": [
    {
      "_id": 0,
      "host": "localhost:57017"
    },
    {
      "_id": 1,
      "host": "localhost:57018"
    },
    {
      "_id": 2,
      "host": "localhost:57019"
    }
  ]
};

rs.initiate({1})
2
{
  "ok": 1,
  "$clusterTime": {
    "clusterTime": Timestamp(1603000359, 1),
    "signature": {
      "hash": BinData(0,"AAAAAAAAAAAAAAAAAA-"),
      "keyId": NumberLong(0)
    }
  },
  "operationTime": Timestamp(1603000359, 1)
}
BPBOnlineReplicaSet3:SECONDARY>

```

A red box highlights the command `rs.initiate()`. A red circle with the number **1** is placed over the configuration object. A red circle with the number **2** is placed over the response object.

**Figure 19.61: Initializing the Replication with "rs.initiate()" Command in MongoDB Shell**

22. If you see carefully in our MongoDB Shell, you will find that your MongoDB prompt has changed to `BPBOnlineReplicaSet3:SECONDARY>`.

Usually, it takes some time to decide for the primary and it could be any member which is chosen to be the primary member of the replica set. But usually, the first member in the configuration file becomes the **Primary Member**. This process can take a few seconds. So, if you just press the *Enter* key without typing anything, this could change the prompt to `BPBOnlineReplicaSet3:PRIMARY>` and you can get the result similar to the screenshot shown in the following screenshot:

```
Administrator: Command Prompt - mongo --port 57017
...
    "host": "localhost:57019"
...
}]
...
{
  "_id": "BPBOnlineReplicaSet3",
  "members": [
    {
      "_id": 0,
      "host": "localhost:57017"
    },
    {
      "_id": 1,
      "host": "localhost:57018"
    },
    {
      "_id": 2,
      "host": "localhost:57019"
    }
  ]
}

rs.initiate(BPBOnlineReplicaSet3Config)

{
  "ok": 1,
  "$clusterTime": [
    {
      "clusterTime": Timestamp(1603888359, 1),
      "signature": [
        {
          "hash": BinData(0, "AAAAAAAAAAAAAAAAAAAAAAA="),
          "keyId": NumberLong(0)
        }
      ],
      "operationTime": Timestamp(1603888359, 1)
    }
  ],
  "operationTime": Timestamp(1603888359, 1)
}
BPBOnlineReplicaSet3:SECONDARY>
BPBOnlineReplicaSet3:PRIMARY>
```

**Figure 19.62:** Changing the MongoDB Shell from SECONDARY to PRIMARY with an Enter key

23. We can verify the status of our replica set using the following command:

```
rs.status()
```

Just enter the preceding command in your MongoDB Shell of the primary instance and you will get the result similar to the following screenshot. You will get the information about the replica set members and their status, which is `ok`, in our case, as shown in the following screenshot:

```
Administrator: Command Prompt - mongo --port 57017
{
  "syncSourceId": 0,
  "infoMessage": "",
  "configVersion": 1,
  "configItem": -1
},
{
  "_id": 2,
  "name": "localhost:57019",
  "health": 1,
  "state": 2,
  "stateStr": "SECONDARY",
  "uptime": 188,
  "optime": {
    "ts": Timestamp(1603800540, 1),
    "t": NumberLong(1)
  },
  "optimeDurable": {
    "ts": Timestamp(1603800540, 1),
    "t": NumberLong(1)
  },
  "optimeDate": ISODate("2020-10-27T14:22:20Z"),
  "optimeDurableDate": ISODate("2020-10-27T14:22:20Z"),
  "lastHeartbeat": ISODate("2020-10-27T14:22:26.293Z"),
  "lastHeartbeatRecv": ISODate("2020-10-27T14:22:27.671Z"),
  "pingMs": NumberLong(0),
  "lastHeartbeatMessage": "",
  "syncSourceHost": "localhost:57017",
  "syncSourceId": 0,
  "infoMessage": "",
  "configVersion": 1,
  "configItem": -1
}
],
"ok": 1,
"$clusterTime": {
  "clusterTime": Timestamp(1603800540, 1),
  "signature": {
    "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAA"),
    "keyId": NumberLong(0)
  }
},
"operationTime": Timestamp(1603800540, 1)
}
MongoDB [IP9OnlineReplicaSet1:PRIMARY]
```

**Figure 19.63: "rs.status()" – Checking the status of the Replica Set**

So, we have successfully created the third shard with the replica set of three members. Now, let us setup our replica set of the config servers with the three members.

## **Step 5 – Create the replica set of config servers in the sharded environment**

Let us create the replica set for the config servers in the MongoDB sharded environment. To do this, we need to perform the following steps:

1. We need to create few folders to keep the MongoDB instances data and log files. Now, instead of doing it manually one-by-one, we will create these folders with the help of a few commands which will automatically create these folders:

### **For MongoDB data files:**

```
mkdir "c:\MongoDB-Sharding\data\ BPBOnlineConfigServer\BPBOnlineDBS1" "c:\MongoDB-Sharding \data\BPBOnlineConfigServer\BPBOnlineDBS2" "c:\MongoDB-Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS3"
```

### **For MongoDB log files:**

```
mkdir "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS1" "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS2" "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS3"
```

If you look this closely, you can find that here our main directory would be `MongoDB-Sharding` under the `c:` drive of our Windows machine (you may change the name and directory according to your wish).

Then, under the `MongoDB-Sharding` directory, there would be 2 child directories, `data`, which will keep the MongoDB data files, and `log`, which will keep the MongoDB log files.

So, till now, the structure will be as follows:

- `c:\MongoDB-Sharding\data`

- c:\MongoDB-Sharding\log

Then, in both the `data` and `log` directories, we will have one common type of directory which would be `BPBOnlineConfigServer` (which is the name of our config server).

At last, under the `BPBOnlineConfigServer` directory, we will have 3 different directories which will cater to each one of the 3 MongoDB instances in the replica set in these directories as follows:

For MongoDB data files (complete path):

- <Main Directory>\data\BPBOnlineConfigServer\BPBOnlineDBS1
- <Main Directory>\data\BPBOnlineConfigServer\BPBOnlineDBS2
- <Main Directory>\data\BPBOnlineConfigServer\BPBOnlineDBS3

For MongoDB log files (complete path):

- <Main Directory>\log\BPBOnlineConfigServer\BPBOnlineDBS1
- <Main Directory>\log\BPBOnlineConfigServer\BPBOnlineDBS2
- <Main Directory>\log\BPBOnlineConfigServer\BPBOnlineDBS3

Here:

- <Main Directory> = c:\MongoDB-Sharding\
- "BPBOnlineDBS1", "BPBOnlineDBS2" and "BPBOnlineDBS3" are the names of the Database Server Directories

If you closely look at the directory structure, it looks perfect to create a config replica set in the sharded environment.

2. Let us first create the new directories which will be the `data` directories for our MongoDB config instances. To do this, first open the Windows command prompt in the administrator mode and then enter the following command, as shown in the following screenshot:

```
mkdir "c:\MongoDB-
Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS1"
"c:\MongoDB-
Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS2"
"c:\MongoDB-
Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS3"
```

```

Administrator: Command Prompt
c:\Program Files\MongoDB\Server\4.4\bin>mkdir "c:\MongoDB-Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS1" "c:\MongoDB-Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS2"
c:\Program Files\MongoDB\Server\4.4\bin>

```

**Figure 19.64:** MongoDB Data Folders - Creating new Folders

- After you have created new directories, you should see them under the `c:` drive, similar to the screenshot shown in the following screenshot:



**Figure 19.65:** MongoDB Data Folders – New Folders gets created

- Now, let us first create the new directories which would be the `log` directories for our MongoDB config instances. To do this, first open the Windows command prompt in the administrator mode and then enter the following command, as shown in the following screenshot:

```

mkdir "c:\MongoDB-
Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS1"
"c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS2"
"c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS3"

```

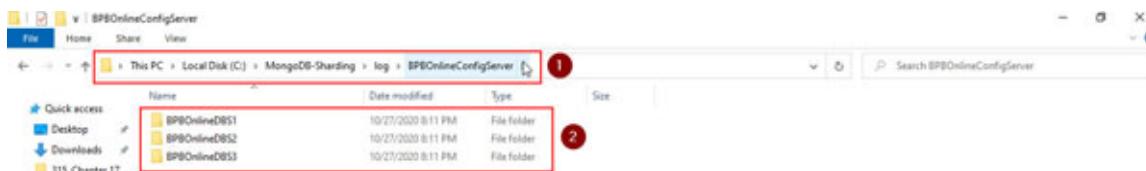
```

Administrator: Command Prompt
c:\Program Files\MongoDB\Server\4.4\bin>mkdir "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS1" "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS2"
c:\Program Files\MongoDB\Server\4.4\bin>

```

**Figure 19.66:** MongoDB Log Folders - Creating new Folders

- After you have created the new directories, you should see them under the `c:` drive, similar to the screenshot shown in the following screenshot:



**Figure 19.67:** MongoDB Log Folders – New Folders gets created

6. Now, we will start our first config server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```
mongod --replSet BPBOnlineConfReplicaSet --dbpath "c:\MongoDB-Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS1\config.log" --port 27040 --configsvr --install --serviceName "BPBOnlineSharding MongoDB 27040" --serviceDisplayName "BPBOnlineSharding MongoDB 27040" --serviceDescription "MongoDB on Port 27040 BPBOnlineConfigServer-BPBOnlineDBS1"
```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the `mongod` command are as follows:

```
--dbpath= Path of our DB Directory Location  
--logpath= Path of our DB Log File Location  
--port = Port Number for MongoDB Instance, in our case we will use "27040"  
--replSet= Replica Set ID for our Instances which is "BPBOnlineConfReplicaSet" in our case  
--configsvr= MongoDB Option for Config Server (Sharded Environment)  
--install = To Install the MongoDB as Service  
--serviceName= Name of the Windows Service, in our case we will be using "BPBOnlineSharding MongoDB 27040"  
--serviceDisplayName= Display Name of the Windows Service, in our case we will be using "BPBOnlineSharding MongoDB 27040"  
--serviceDescription= Description of the Windows Service, in our case we will be using "MongoDB on Port 27040 BPBOnlineConfigServer-BPBOnlineDBS1"
```

7. Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following screenshot:



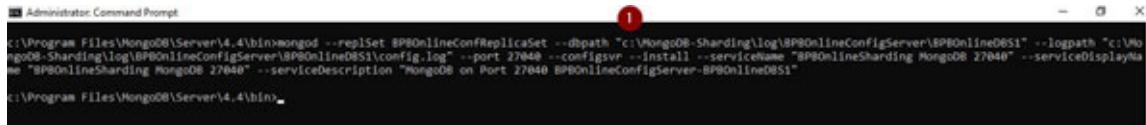
```
Administrator: Command Prompt  
Microsoft Windows [Version 10.0.19041.572]  
(C) 2020 Microsoft Corporation. All Rights Reserved.  
C:\Windows\system32>cd "c:\Program Files\MongoDB\Server\4.4\bin" ①  
c:\Program Files\MongoDB\Server\4.4\bin> ②
```

**Figure 19.68:** Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory

- Now, type the following command with all the parameters that we will create Windows service for our first server in the replica set as follows:

```
mongod --replicaSet BPBOnlineConfReplicaSet --dbpath "c:\MongoDB-Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS1\config.log" --port 27040 --configsvr --install --serviceName "BPBOnlineSharding MongoDB 27040" --serviceDisplayName "BPBOnlineSharding MongoDB 27040" --serviceDescription "MongoDB on Port 27040 BPBOnlineConfigServer-BPBOnlineDBS1"
```

After you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:



```
Administrator: Command Prompt  
C:\Program Files\MongoDB\Server\4.4\bin>mongod --replicaSet BPBOnlineConfReplicaSet --dbpath "c:\MongoDB-Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS1\config.log" --port 27040 --configsvr --install --serviceName "BPBOnlineSharding MongoDB 27040" --serviceDisplayName "BPBOnlineSharding MongoDB 27040" --serviceDescription "MongoDB on Port 27040 BPBOnlineConfigServer-BPBOnlineDBS1"  
C:\Program Files\MongoDB\Server\4.4\bin>
```

**Figure 19.69:** Adding the new MongoDB Windows Service - "mongod" with Replication parameters

- Now, type the following command to start this Windows service from your command prompt already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 27040"
```



```
Administrator: Command Prompt  
C:\Program Files\MongoDB\Server\4.4\bin>mongod --replicaSet BPBOnlineConfReplicaSet --dbpath "c:\MongoDB-Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS1" --logpath "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS1\config.log" --port 27040 --configsvr --install --serviceName "BPBOnlineSharding MongoDB 27040" --serviceDisplayName "BPBOnlineSharding MongoDB 27040" --serviceDescription "MongoDB on Port 27040 BPBOnlineConfigServer-BPBOnlineDBS1"  
C:\Program Files\MongoDB\Server\4.4\bin>net start "BPBOnlineSharding MongoDB 27040" ①  
The BPBOnlineSharding MongoDB 27040 service is starting...  
The BPBOnlineSharding MongoDB 27040 service was started successfully. ②  
C:\Program Files\MongoDB\Server\4.4\bin>
```

**Figure 19.70:** Starting the new MongoDB Windows Service

10. Now, we will start our second config server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```
mongod --replSet BPBOnlineConfReplicaSet --dbpath  
"c:\MongoDB-\  
Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS2" --logpath  
"c:\MongoDB-\  
Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS2\config.log"  
--port 27041 --configsvr --install --serviceName  
"BPBOnlineSharding MongoDB 27041" --serviceDisplayName  
"BPBOnlineSharding MongoDB 27041" --serviceDescription  
"MongoDB on Port 27041 BPBOnlineConfigServer-BPBOnlineDBS2"
```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the `mongod` command are as follows:

```
--dbpath= Path of our DB Directory Location  
--logpath= Path of our DB Log File Location  
--port = Port Number for MongoDB Instance, in our case we  
will use "27041"  
--replSet= Replica Set ID for our Config Servers, which is  
"BPBOnlineConfReplicaSet" in our case.  
--configsvr= MongoDB Option for Config Server (Sharded  
Environment)  
--install = To Install the MongoDB as Service  
--serviceName= Name of the Windows Service, in our case we  
will be using "BPBOnlineSharding MongoDB 27041"  
--serviceDisplayName= Display Name of the Windows Service, in  
our case we will be using "BPBOnlineSharding MongoDB 27041"  
--serviceDescription= Description of the Windows Service, in  
our case we will be using "MongoDB on Port 27041  
BPBOnlineConfigServer-BPBOnlineDBS2"
```

11. Now, open the Command Prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path,

which is C:\Program Files\MongoDB\Server\4.4\bin, in our case, as shown in the following screenshot:



A screenshot of an Administrator Command Prompt window. The title bar says "Administrator: Command Prompt". The command line shows: "Administrator: Command Prompt [Version 10.0.19041.572]" and "C:\Windows\system32>cd "c:\Program Files\MongoDB\Server\4.4\bin"" with a red circle labeled '1' over the command line. The prompt then changes to "c:\Program Files\MongoDB\Server\4.4\bin>" with a red circle labeled '2' over it.

**Figure 19.71: Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory**

12. Now, type the following command with all the parameters that we will create a Windows service for our second server in the replica set as follows:

```
mongod --replSet BPBOnlineConfReplicaSet --dbpath  
"c:\MongoDB-  
Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS2" --logpath  
"c:\MongoDB-  
Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS2\config.log"  
--port 27041 --configsvr --install --serviceName  
"BPBOnlineSharding MongoDB 27041" --serviceDisplayName  
"BPBOnlineSharding MongoDB 27041" --serviceDescription  
"MongoDB on Port 27041 BPBOnlineConfigServer-BPBOnlineDBS2"
```

After you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:



A screenshot of an Administrator Command Prompt window. The command line shows: "Administrator: Command Prompt [Version 10.0.19041.572]" and "c:\Program Files\MongoDB\Server\4.4\bin>mongod --replSet BPBOnlineConfReplicaSet --dbpath "c:\MongoDB-Sharding\data\BPBOnlineDBS2" --logpath "c:\MongoDB-Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS2\config.log" --port 27041 --configsvr --install --serviceName "BPBOnlineSharding MongoDB 27041" --serviceDisplayName "BPBOnlineSharding MongoDB 27041" --serviceDescription "MongoDB on Port 27041 BPBOnlineConfigServer-BPBOnlineDBS2" with a red circle labeled '1' over the command line. The prompt then changes to "c:\Program Files\MongoDB\Server\4.4\bin>".

**Figure 19.72: Adding the new MongoDB Windows Service - "mongod" with Replication parameter**

13. Now, type the following command to start this Windows service from your command prompt already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 27041"
```

**Figure 19.73: Connected to the MongoDB Shell**

14. Now, we will start our third config server with the help of `mongod` command by giving few parameters as follows (note that here, we will run this as Windows service as explained in the previous section of this chapter):

```
mongod --replSet BPBOnlineConfReplicaSet --dbpath
"c:\MongoDB-
Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS3" --logpath
"c:\MongoDB-
Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS3\config.log"
--port 27042 --configsvr --install --serviceName
"BPBOnlineSharding MongoDB 27042" --serviceDisplayName
"BPBOnlineSharding MongoDB 27042" --serviceDescription
"MongoDB on Port 27042 BPBOnlineConfigServer-BPBOnlineDBS3"
```

This command needs to run in the command prompt open in the administrator mode. Here, the parameters given in the `mongod` command are as follows:

```
--dbpath= Path of our DB Directory Location
--logpath= Path of our DB Log File Location
--port = Port Number for MongoDB Instance, in our case we
will use "27042"
--configsvr= MongoDB Option for Config Server (Sharded
Environment)
--replSet= Replica Set ID for our Config Servers, which is
"BPBOnlineConfReplicaSet" in our case.
--install = To Install the MongoDB as Service
--serviceName= Name of the Windows Service, in our case we
will be using "BPBOnlineSharding MongoDB 27042"
--serviceDisplayName= Display Name of the Windows Service, in
our case we will be using "BPBOnlineSharding MongoDB 27042"
--serviceDescription= Description of the Windows Service, in
our case we will be using "MongoDB on Port 27042"
```

```
BPBOnlineConfigServer-BPBOnlineDBS3"
```

- Now, open the command prompt in the administrator mode and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, as shown in the following screenshot:



**Figure 19.74:** Opening Command Prompt with Administrator Mode and navigating to MongoDB "bin" directory

- Now, type the following command with all the parameters that we will create a Windows service for our third server in the replica set as follows:

```
mongod --replSet BPBOnlineConfReplicaSet --dbpath  
"c:\MongoDB-\  
Sharding\data\BPBOnlineConfigServer\BPBOnlineDBS3" --logpath  
"c:\MongoDB-\  
Sharding\log\BPBOnlineConfigServer\BPBOnlineDBS3\config.log"  
--port 27042 --configsvr --install --serviceName  
"BPBOnlineSharding MongoDB 27042" --serviceDisplayName  
"BPBOnlineSharding MongoDB 27042" --serviceDescription  
"MongoDB on Port 27042 BPBOnlineConfigServer-BPBOnlineDBS3"
```

After you enter this command, you will get a response something similar to the screenshot shown in the following screenshot:



**Figure 19.75:** Adding the new MongoDB Windows Service - "mongod" with Replication parameters

- Now, type the following command to start this Windows service from your command prompt already open in the administrative mode, as shown in the following screenshot:

```
net start "<Name of the Windows Service>"
```

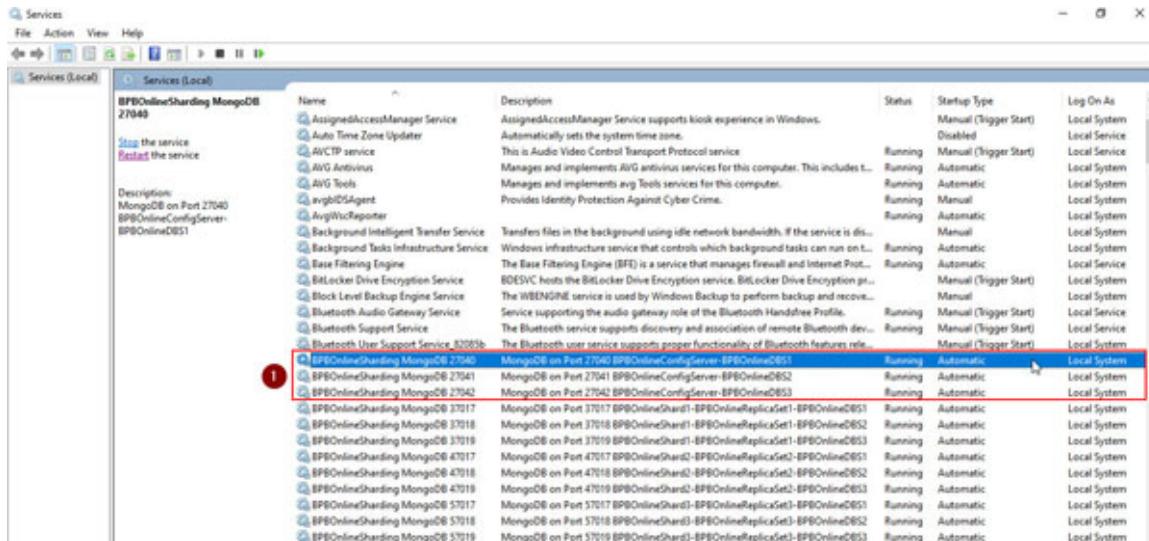
In our case, it is as follows:

```
net start "BPBOnlineSharding MongoDB 27042"
```

The screenshot shows a Windows Command Prompt window titled 'Administrator: Command Prompt'. The command entered is 'net start "BPBOnlineSharding MongoDB 27042"'. The output shows the service starting successfully. A red circle labeled '1' highlights the command line. A red circle labeled '2' highlights the status message 'The BPBOnlineSharding MongoDB 27042 service was started successfully.'

**Figure 19.76: Starting the new MongoDB Windows Service**

- Now, as our all three MongoDB server `mongod` instances have successfully started, we can open the **Windows Service Manager** (as explained in the previous step of this chapter where we have explained the process of opening the Windows service manager using the Windows search box), and then navigate to the service named `BPBOnlineSharding MongoDB<Port Number>`. You will see that all the MongoDB instances are created in the background and their status should be `Running`, as shown in the following screenshot:

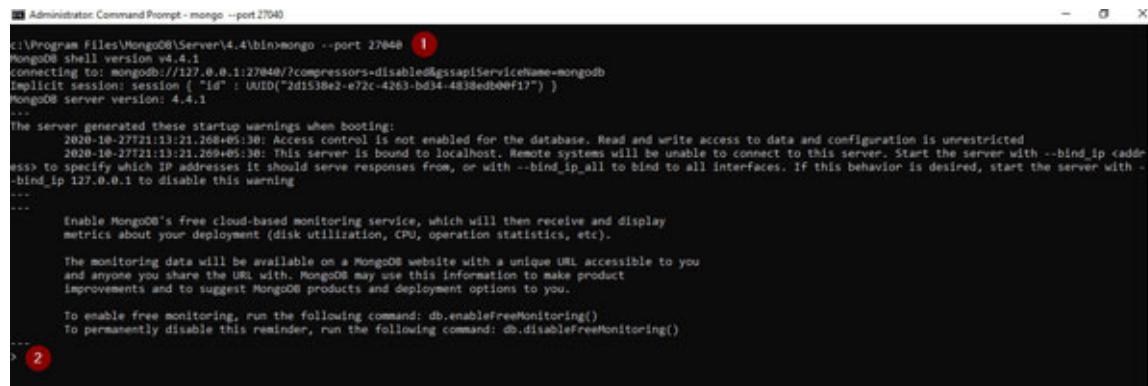


**Figure 19.77: Windows Service Manager – MongoDB Replica Set Instances (Running in Background)**

- Now, open the existing command prompt (which is already open in the administrator mode). If you are not already in the `bin` directory of your MongoDB installation path, first navigate to it, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, and then type the following command:

```
mongo --port 27040
```

This command will connect us to the MongoDB Shell, as shown in the following screenshot:



```
C:\Program Files\MongoDB\Server\4.4\bin>mongo --port 27040
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27040/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("2d1538e2-e72c-4263-bd34-4838edb0ef1") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-10-27T21:13:21.268+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-10-27T21:13:21.269+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the 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()
```

**Figure 19.78: Connected to the MongoDB Shell**

20. Now, we need to initiate the replication process and for this we need to add our all the member details with their host and port. In our case, we will create a variable named BPBOnlineConfReplicaSetConfig with the \_id same as our replica set, which is BPBOnlineConfReplicaSet and the member details have the \_id 0, 1, and 2 (for the first, second, and third members in the replica set) and their host and port values which are localhost:27040, localhost:27041, and localhost:27042 (for the first, second, and third members in the replica set). In our case, it is as follows:

```
BPBOnlineConfReplicaSetConfig = {
  _id: "BPBOnlineConfReplicaSet",
  members: [
    {
      _id: 0,
      host: "localhost:27040"
    },
    {
      _id: 1,
      host: "localhost:27041"
    },
    {
      _id: 2,
      host: "localhost:27042"
    }
]
```

```

        }
    ]
} ;

```

Enter the preceding code in the MongoDB Shell and you will get the result similar to the screenshot shown in the following screenshot:

```

Administrator: Command Prompt - mongo --port 27040
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()

> BPBOnlineConfReplicaSetConfig = {
...     "_id": "BPBOnlineConfReplicaSet",
...     "members": [
...         {
...             "_id": 0,
...             "host": "localhost:27040"
...         },
...         {
...             "_id": 1,
...             "host": "localhost:27041" ①
...         },
...         {
...             "_id": 2,
...             "host": "localhost:27042"
...         }
...     ],
...     "_id": "BPBOnlineConfReplicaSet",
...     "members": [
...         {
...             "_id": 0,
...             "host": "localhost:27040"
...         },
...         {
...             "_id": 1,
...             "host": "localhost:27041" ②
...         },
...         {
...             "_id": 2,
...             "host": "localhost:27042"
...         }
...     ]
}

```

**Figure 19.79: MongoDB Shell – Replication Config Variable**

21. Now, we can use the variable `BPBOnlineConfReplicaSetConfig` during the initialization command for the replica set creation. Type the following command to initiate the replication process:

```
s.initiate(BPBOnlineConfReplicaSetConfig)
```

Enter the preceding command in the MongoDB Shell and you will get the result similar to the screenshot shown in the following screenshot:

**Figure 19.80: Initializing the Replication with "rs.initiate()" Command in MongoDB Shell**

22. If you see carefully in your MongoDB Shell, you will find that your MongoDB prompt has changed to `BPBOnlineConfReplicaSet:SECONDARY>`.

Usually, it takes some time to decide for the primary and it could be any member chosen to be the primary member of the replica set. But usually, the first member in the configuration file becomes the **Primary Member**. This process can take a few seconds. So, if you just press the *Enter* key without typing anything, this could change the prompt to `BPBOnlineConfReplicaSet:PRIMARY>` and you can get the result similar to the screenshot shown in the following screenshot:

The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - mongo --port 27040". The command entered is "rs.initiate({})." The response shows the configuration of a replica set named "IPBOnlineConfReplicaSet" with three members. The status of the primary node is highlighted with a red box and a red circle containing the number 1.

```

Administrator: Command Prompt - mongo --port 27040
> rs.initiate({})
{
  "_id" : "IPBOnlineConfReplicaSet",
  "members" : [
    {
      "_id" : 0,
      "host" : "localhost:27040"
    },
    {
      "_id" : 1,
      "host" : "localhost:27041"
    },
    {
      "_id" : 2,
      "host" : "localhost:27042"
    }
  ]
}
> rs.initiate({$parallel: 1})
{
  "ok" : 1,
  "$gleStats" : {
    "lastOpTime" : Timestamp(1603814118, 1),
    "electionId" : ObjectId("000000000000000000000000")
  },
  "lastCommittedOpTime" : Timestamp(0, 0),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1603814118, 1),
    "signature" : BinData(0, "AAAAAAAAAAAAAAAAAAAAAA-"),
    "keyId" : NumberLong(0)
  },
  "operationTime" : Timestamp(1603814118, 1)
}
IPBOnlineConfReplicaSet:SECONDARY>
IPBOnlineConfReplicaSet:PRIMARY> 1

```

**Figure 19.81:** Changing the MongoDB Shell from SECONDARY to PRIMARY with an Enter key

23. We can verify the status of our replica set using the following command:

```
rs.status()
```

Just enter the preceding command in your MongoDB Shell of the primary instance and you will get the result similar to the following screenshot. You will get the information about the replica set members and their status, which is `ok`, in our case, as shown in the following screenshot:

The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - mongo --port 27040". The command entered is "rs.status()." The response displays the status of the replica set members, including their health, state, and last heartbeat times. The primary node's status is highlighted with a red box and a red circle containing the number 1.

```

Administrator: Command Prompt - mongo --port 27040
> rs.status()
{
  "ok" : 1,
  "configVersion" : 3,
  "configTerm" : 1
},
{
  "_id" : 2,
  "name" : "localhost:27042",
  "health" : 1,
  "state" : 2,
  "stateStr" : "SECONDARY",
  "uptime" : 344,
  "optime" : {
    "t" : Timestamp(1603814262, 1),
    "t" : NumberLong(1)
  },
  "optimeDurable" : {
    "t" : Timestamp(1603814262, 1),
    "t" : NumberLong(1)
  },
  "optimeDate" : ISODate("2020-10-27T15:57:42Z"),
  "optimeDurableDate" : ISODate("2020-10-27T15:57:42Z"),
  "lastHeartbeat" : ISODate("2020-10-27T15:57:42.676Z"),
  "lastHeartbeatRecv" : ISODate("2020-10-27T15:57:43.852Z"),
  "pingMs" : NumberLong(0),
  "lastHeartbeatMessage" : "",
  "syncSourceHost" : "localhost:27040",
  "syncSourceId" : 0,
  "infoMessage" : "",
  "configVersion" : 1,
  "configTerm" : 1
},
{
  "_id" : 1,
  "name" : "localhost:27041",
  "health" : 1,
  "state" : 1,
  "stateStr" : "PRIMARY",
  "uptime" : 344,
  "optime" : {
    "t" : Timestamp(1603814262, 1),
    "t" : NumberLong(1)
  },
  "optimeDurable" : {
    "t" : Timestamp(1603814262, 1),
    "t" : NumberLong(1)
  },
  "optimeDate" : ISODate("2020-10-27T15:57:42Z"),
  "optimeDurableDate" : ISODate("2020-10-27T15:57:42Z"),
  "lastHeartbeat" : ISODate("2020-10-27T15:57:42.676Z"),
  "lastHeartbeatRecv" : ISODate("2020-10-27T15:57:43.852Z"),
  "pingMs" : NumberLong(0),
  "lastHeartbeatMessage" : "",
  "syncSourceHost" : "localhost:27040",
  "syncSourceId" : 0,
  "infoMessage" : "",
  "configVersion" : 1,
  "configTerm" : 1
},
{
  "_id" : 0,
  "name" : "localhost:27040"
},
{
  "ok" : 1,
  "$gleStats" : {
    "lastOpTime" : Timestamp(1603814118, 1),
    "electionId" : ObjectId("000000000000000000000001")
  },
  "lastCommittedOpTime" : Timestamp(1603814263, 1),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1603814263, 1),
    "signature" : BinData(0, "AAAAAAAAAAAAAAAAAAAAAA-"),
    "keyId" : NumberLong(0)
  }
}

```

**Figure 19.82:** "rs.status()" – Checking the status of the Replica Set

So, we have successfully created the config servers with the replica set of three members. Now, let us start our MongoDB in the sharded environment as `mongos`.

## Step 6 - Starting MongoDB in the sharded environment as "mongos"

To start MongoDB in the sharded environment, we need to follow these steps:

1. Open the existing command prompt (which is already open in the administrator mode). If you are not already in the `bin` directory of your MongoDB installation path, first navigate to it, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, and then type the following command:

```
mongos --logpath "c:\MongoDB-Sharding\mongos.log" --configdb  
BPBOnlineConfReplicaSet/localhost:27040,localhost:27041,local  
host:27042 --port 27043
```

Note that in `--configdb`, you must specify the replica set ID for your config servers.

This will start MongoDB in the sharded environment `mongos` on the port `27043` and will take few seconds to start, as shown in the following screenshot:



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - mongos". The command entered is "mongos --logpath "c:\MongoDB-Sharding\mongos.log" --configdb BPBOnlineConfReplicaSet/localhost:27040,localhost:27041,localhost:27042 --port 27043". The output shows a log entry: {"t": {"\$date": "2020-10-29T12:25:07.776Z"}, "s": "I", "c": "CONTROL", "id": 20697, "ctx": "main", "msg": "Renamed existing log file", "attr": {"oldLogPath": "c:\MongoDB-Sharding\mongos.log", "newLogPath": "c:\MongoDB-Sharding\mongos.log.2020-10-29T12:25:07"}}. A red exclamation mark icon is visible in the bottom-left corner of the window.

**Figure 19.83:** Starting MongoDB in Sharded Environment "mongos"

2. Now, open the new command prompt (in the administrator mode) and navigate to the `bin` directory of your MongoDB installation path, which is `C:\Program Files\MongoDB\Server\4.4\bin`, in our case, and then type the following command:

```
mongo --port 27043
```

The preceding command will start the MongoDB Shell and you can now see that the Shell prompt will come as `mongos`, as shown

in the following screenshot:



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - mongo --port 27043". The command entered was "mongo --port 27043". The output shows the MongoDB shell version 4.4.1 connecting to the mongos instance at localhost:27043. It also displays a warning message about access control being disabled and the server being bound to localhost. The prompt "mongos>" is visible at the bottom.

*Figure 19.84: Connected to the MongoDB Shell "mongos"*

## Step 7 – Adding MongoDB in the shard keys

As in the preceding step, you have logged into the MongoDB Shell (sharding) `mongos` (MongoDB sharded environment running at `mongo --port 27043`), keep that open and you need to follow these instructions mentioned:

1. In this MongoDB Shell `mongos`, we need to add shards which are the primary members of our replica sets that we have created. To do this, type the following commands:

```
db.adminCommand({addshard :  
    "BPBOnlineReplicaSet1/"+"localhost:37017"});  
db.adminCommand({addshard :  
    "BPBOnlineReplicaSet2/"+"localhost:47017"});  
db.adminCommand({addshard :  
    "BPBOnlineReplicaSet3/"+"localhost:57017"});
```

Here, the `db.adminCommand()` method takes the parameter with the `addshard` key and the value is the string with the `Name Space` of the replica set ID that we have created earlier. Then it is concatenated with the host and the IP address of the primary member of the replica set. We can also write the preceding command without concatenation as follows:

```
db.adminCommand({addshard :  
    "BPBOnlineReplicaSet1/localhost:37017"});
```

The preceding command will add the primary members of each of the replica sets that we have created earlier in the sharded environment. Once each replica set is added, you will see the

success message with `ok` status, as shown in the following screenshot:

```

Administrator: Command Prompt - mongo --port 27041
C:\Program Files\MongoDB\Server\4.4\bin>mongo --port 27041
MongoDB shell version: 4.4.1
connecting to: mongodb://127.0.0.1:27041/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "_id" : UUID("77a955e1-f7f3-46ca-bd38-e995deaa6b19") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-10-28T14:57:19.651+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-10-28T14:57:19.651+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <addr> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
...
mongos> db.adminCommand( { addshard : "BPBOnlineReplicaSet1/" + "localhost:37017" } );
{
    "shardAdded" : "BPBOnlineReplicaSet1",
    "ok" : 1,
    "operationTime" : Timestamp(1603877301, 2),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1603877301, 2),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId" : NumberLong(0)
        }
    }
}
mongos>

```

**Figure 19.85: Adding Shards**

Once all the primary members of each of the replica sets that we have created earlier have been successfully added into the sharded environment, you will receive `ok` status, as shown in the following screenshot:

```

Administrator: Command Prompt - mongo --port 27041
2020-10-28T14:57:19.651+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <addr> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
...
mongos> db.adminCommand( { addshard : "BPBOnlineReplicaSet1/" + "localhost:37017" } );
{
    1 "shardAdded" : "BPBOnlineReplicaSet1",
    "ok" : 1,
    "operationTime" : Timestamp(1603877301, 2),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1603877301, 2),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId" : NumberLong(0)
        }
    }
}
mongos> db.adminCommand( { addshard : "BPBOnlineReplicaSet2/" + "localhost:47017" } );
{
    2 "shardAdded" : "BPBOnlineReplicaSet2",
    "ok" : 1,
    "operationTime" : Timestamp(1603877589, 2),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1603877589, 2),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId" : NumberLong(0)
        }
    }
}
mongos> db.adminCommand( { addshard : "BPBOnlineReplicaSet3/" + "localhost:57017" } );
{
    3 "shardAdded" : "BPBOnlineReplicaSet3",
    "ok" : 1,
    "operationTime" : Timestamp(1603877607, 4),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1603877607, 4),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId" : NumberLong(0)
        }
    }
}
mongos>

```

**Figure 19.86: Adding Shards**

- Now, we need to enable sharding for our database which we would like to use in the sharded environment. We can do this by using the following command:

```
db.adminCommand({ enableSharding: "BPBOnlineShardedDB" })
```

Here, the `db.adminCommand()` method takes the parameter with the `enableSharding` key and the value is the string which is the database name that we would like to use for sharding. The preceding command will enable sharding to the MongoDB database mentioned in the preceding command and you will see the status as `ok` once this command gets successfully completed, as shown in the following screenshot:

```

Administrator: Command Prompt - mongo --port 27043
{
    "hash" : BinData(0,"AAAAAAAAAAAAAAAAAA-"),
    "keyId" : NumberLong(0)
}
}
mongos> db.adminCommand( { addshard : "BPBOnlineReplicaSet2/"+"localhost:47017" } );
{
    "shardAdded" : "BPBOnlineReplicaSet2",
    "ok" : 1,
    "operationTime" : Timestamp(1603877589, 2),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1603877589, 2),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId" : NumberLong(0)
        }
    }
}
mongos> db.adminCommand( { addshard : "BPBOnlineReplicaSet3/"+"localhost:57017" } );
{
    "shardAdded" : "BPBOnlineReplicaSet3",
    "ok" : 1,
    "operationTime" : Timestamp(1603877607, 4),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1603877607, 4),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId" : NumberLong(0)
        }
    }
}
mongos> db.adminCommand( {enableSharding: "BPBOnlineShardedDB"} ) ①
{
    "ok" : 1, ②
    "operationTime" : Timestamp(1603882239, 3),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1603882239, 3),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAA-"),
            "keyId" : NumberLong(0)
        }
    }
}
mongos>

```

**Figure 19.87: Enabling Sharding on Database**

- Now, we will add the sharded collection and specify the shard key. We can do this by using the following command:

```

db.adminCommand({shardCollection:
  "BPBOnlineShardedDB.BPBOnlineShardedCollection", key:
  {_id:1}})

```

Here, the `db.adminCommand()` method takes the first parameter with the `shardCollection` key and the value is the string which is a combination of the database and collection name <DatabaseName.CollectionName> we would like to use for sharding.

The second parameter of this command is the key which is the Sharded Key and we can specify the value of our sharded key in this parameter.

The preceding command will enable the shard key to the MongoDB collection mentioned in the preceding command and

you will see the status as `ok` once this command gets successfully completed, as shown in the following screenshot:

The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - mongo --port 27043". It displays several MongoDB shell commands:

- `db.adminCommand({ addshard : "BPBOnlineReplicaSet3/localhost:57817" })`
- `db.adminCommand({ enableSharding: "BPBOnlineShardedDB" })`
- `db.adminCommand({ shardCollection: "BPBOnlineShardedDB.BPBOnlineShardedCollection", key: { _id: 1 } })` (Line 1, highlighted with a red box)
- `db.BPBOnlineShardedCollection.insert([ { "booktitle": ".Net Interview Questions - 7th Revised Edition" }, { "booktitle": "101 Challenges In C Programming" }, { "booktitle": "101 Challenges In C++ Programming" }, { "booktitle": "21 Internet Of Things (IOT) Experiments" } ])` (Line 2, highlighted with a red box)

**Figure 19.88: Adding Shard Collection with Shard Key**

## Step 8 - Verifying the MongoDB sharding using data

For the verification process, we need to follow these steps:

Go to the MongoDB Shell of the sharded environment where `mongos` is running (MongoDB sharded environment running at `mongo --port 27043`) and create a new database and collection and insert some documents. We will run the following command on the MongoDB Shell of the sharded environment where "`mongos`" is running, as shown in the following screenshot:

```
use BPBOnlineShardedDB
db.BPBOnlineShardedCollection.insert([
    "booktitle": ".Net Interview Questions - 7th Revised Edition",
    {
        "booktitle": "101 Challenges In C Programming"
    },
    {
        "booktitle": "101 Challenges In C++ Programming"
    },
    {
        "booktitle": "21 Internet Of Things (IOT) Experiments"
    }
])
```

```

}, {
  "booktitle": "3D Game Weapons (Modeling UV Mapping &
Texturing)"
}, {
  "booktitle": "3D Printing Made Simple"
}, {
  "booktitle": "A Practical Approach for Machine Learning and
Deep Learning Algorithms"
}, {
  "booktitle": "ARTIFICIAL INTELLIGENCE"
} ] );

```

```

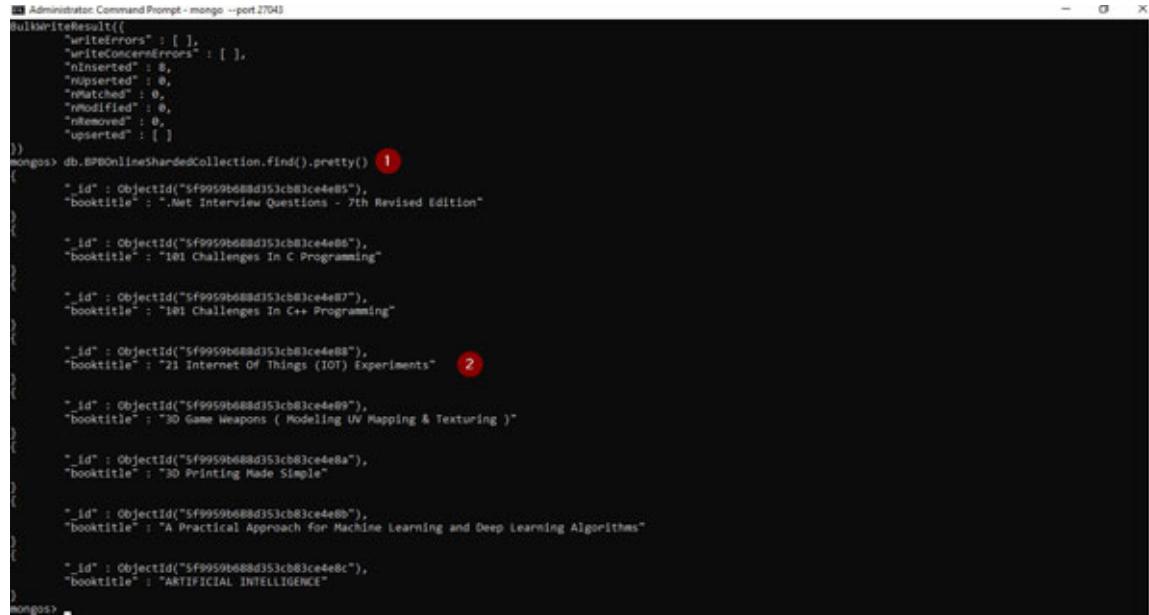
Administrator: Command Prompt - mongo --port27049
mongos> use BPBOnlineSharded08
switched to db BPBOnlineSharded08
mongos> db.BPBOnlineShardedCollection.insert([
...   {
...     "booktitle": ".Net Interview Questions - 7th Revised Edition"
...   },
...   {
...     "booktitle": "101 Challenges In C Programming"
...   },
...   {
...     "booktitle": "101 Challenges In C++ Programming"
...   },
...   {
...     "booktitle": "21 Internet Of Things (IoT) Experiments"
...   },
...   {
...     "booktitle": "3D Game Weapons ( Modeling UV Mapping & Texturing )"
...   },
...   {
...     "booktitle": "3D Printing Made Simple"
...   },
...   {
...     "booktitle": "A Practical Approach for Machine Learning and Deep Learning Algorithms"
...   },
...   {
...     "booktitle": "ARTIFICIAL INTELLIGENCE"
... });
BulkWriteResult({
  "writeErrors": [ ],
  "writeConcernErrors": [ ],
  "nInserted": 8,
  "nUpserted": 0,
  "nMatched": 0,
  "nModified": 0,
  "nRemoved": 0,
  "nUpgraded": 0
})
mongos>

```

**Figure 19.89:** Running few Database commands on the MongoDB Shell of the Sharded Environment where "mongos" is running

1. Now, run the following command to view the inserted documents, as shown in the following screenshot:

```
db.BPBOnlineShardedCollection.find().pretty()
```



```

Administrator: Command Prompt - mongo --port 27043
> bulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 8,
  "nModified" : 0,
  "nMatched" : 0,
  "nRemoved" : 0,
  "nUpserted" : 0
})
> db.BP8OnlineShardedCollection.find().pretty() ①
[{"_id": ObjectId("5f9959b688d353cb83ce4e85"), "booktitle": ".Net Interview Questions - 7th Revised Edition"}, {"_id": ObjectId("5f9959b688d353cb83ce4e86"), "booktitle": "101 Challenges In C Programming"}, {"_id": ObjectId("5f9959b688d353cb83ce4e87"), "booktitle": "101 Challenges In C++ Programming"}, {"_id": ObjectId("5f9959b688d353cb83ce4e88"), "booktitle": "71 Internet Of Things (IOT) Experiments" ②"}, {"_id": ObjectId("5f9959b688d353cb83ce4e89"), "booktitle": "3D Game Weapons ( Modeling UV Mapping & Texturing )"}, {"_id": ObjectId("5f9959b688d353cb83ce4e8a"), "booktitle": "3D Printing Made Simple"}, {"_id": ObjectId("5f9959b688d353cb83ce4e8b"), "booktitle": "A Practical Approach for Machine Learning and Deep Learning Algorithms"}, {"_id": ObjectId("5f9959b688d353cb83ce4e8c"), "booktitle": "ARTIFICIAL INTELLIGENCE"}]

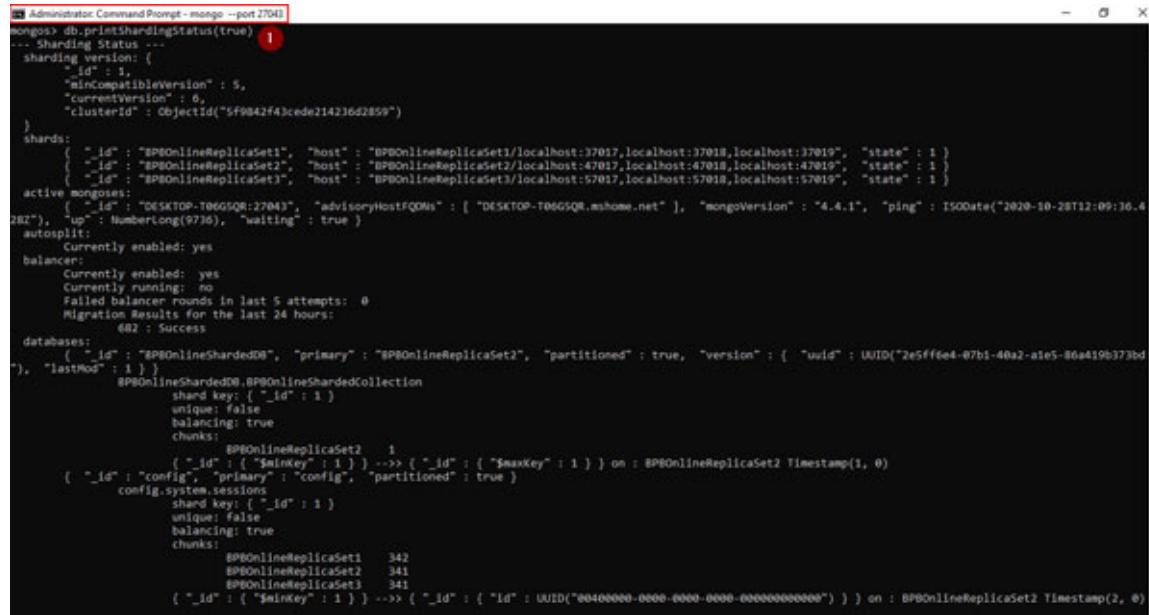
```

**Figure 19.90:** Viewing the Inserted Documents

2. In the same MongoDB Shell, type the following command to verify if the sharding has taken place and its status around the shards:

```
db.printShardingStatus(true)
```

After you run the preceding commands, you will see the result similar to the screenshot, as shown in the following screenshot:



```

Administrator: Command Prompt - mongo --port 27043
> db.printShardingStatus(true) ①
{
  "_id": 1,
  "minCompatibleVersion": 5,
  "currentVersion": 6,
  "clusterId": ObjectId("5f9842f43cede214236d2859")
}
shards:
  {
    "_id": "BP8OnlineReplicaSet1",
    "host": "BP8OnlineReplicaSet1/localhost:37017,localhost:37018,localhost:37019",
    "state": 1
  }
  {
    "_id": "BP8OnlineReplicaSet2",
    "host": "BP8OnlineReplicaSet2/localhost:47017,localhost:47018,localhost:47019",
    "state": 1
  }
  {
    "_id": "BP8OnlineReplicaSet3",
    "host": "BP8OnlineReplicaSet3/localhost:57017,localhost:57018,localhost:57019",
    "state": 1
  }
active mongoses:
  {
    "_id": "DESKTOP-T06G5QR:27043",
    "advisoryHostFQDNs": [ "DESKTOP-T06G5QR.mshome.net" ],
    "mongoVersion": "4.4.1",
    "ping": ISODate("2020-10-28T12:09:36.482Z"),
    "up": NumberLong(9736),
    "waiting": true
  }
  autopartition:
    Currently enabled: yes
  balancer:
    Currently enabled: yes
    Currently running: no
    Failed balancer rounds in last 5 attempts: 0
    Migration Results for the last 24 hours:
      602 : Success
  databases:
    {
      "_id": "BP8OnlineShardedDB",
      "primary": "BP8OnlineReplicaSet2",
      "partitioned": true,
      "version": {
        "uuid": UUID("2e5ff6e4-07b1-40a2-a1e5-86a419b373bd")
      },
      "lastMod": 1
    }
    BP8OnlineShardedDB.BP8OnlineShardedCollection
      shard key: { "_id": 1 }
      unique: false
      balancing: true
      chunks:
        BP8OnlineReplicaSet2 1
        { "_id": { "$minKey": 1 } } --> { "_id": { "$maxKey": 1 } } on : BP8OnlineReplicaSet2 Timestamp(1, 0)
    {
      "_id": "config",
      "primary": "config",
      "partitioned": true
    }
    config.system.sessions
      shard key: { "_id": 1 }
      unique: false
      balancing: true
      chunks:
        BP8OnlineReplicaSet1 342
        BP8OnlineReplicaSet2 341
        BP8OnlineReplicaSet3 341
    { "_id": { "$minKey": 1 } } --> { "_id": { "id": UUID("00400000-0000-0000-0000-000000000000") } } on : BP8OnlineReplicaSet2 Timestamp(2, 0)

```

**Figure 19.91:** Printing the Sharding Status

Note that here, `true` in the parameter value of the command is used to print all the output and you can see that it can print large output related to the sharding, as shown in the following screenshot:

```
Administrator: Command Prompt - mongo --port 27049
db.BPBOnlineReplicaSet1.find({}).pretty(true)
[{"_id": "a6400000-0000-0000-0000-000000000000"}, {"_id": "a6800000-0000-0000-0000-000000000000"}, {"_id": "a6c00000-0000-0000-0000-000000000000"}, {"_id": "a7000000-0000-0000-0000-000000000000"}, {"_id": "a7400000-0000-0000-0000-000000000000"}, {"_id": "a7800000-0000-0000-0000-000000000000"}, {"_id": "a7c00000-0000-0000-0000-000000000000"}, {"_id": "a8000000-0000-0000-0000-000000000000"}]
```

**Figure 19.92:** Printing the Sharding Status – with "true" value of the parameter

3. We can also run another command to verify if everything is working fine at the collection level in the sharded environment of MongoDB, as shown in the following screenshot:

```
db.BPBOnlineShardedCollection.getShardDistribution()
```

```
Administrator: Command Prompt - mongo --port 27049
songs> db.BPBOnlineShardedCollection.getShardDistribution()
Shard: BPBOnlineReplicaSet2 at BPBOnlineReplicaSet2/localhost:47017,localhost:47018,localhost:47019
data : 6200 docs : 8 chunks : 1
estimated data per chunk : 6200
estimated docs per chunk : 8
totals
data : 62000 docs : 1
Shard: BPBOnlineReplicaSet2 contains 100% data, 100% docs in cluster, avg obj size on shard : 778
```

**Figure 19.93:** View the Shard Distribution of the Collection's Documents

So, we can see that right now all the eight documents get stored in the replica set `BPBOnlineReplicaSet2`. Once we have more records, other replica sets will get distributed in the documents around all the three replica sets of the sharded environment. We can go to the primary server of our replica set 2, `BPBOnlineReplicaSet2`, which runs on port `47017` and run the following commands. We can see all the 8 records entered by us will be displayed as this replica

set contains these 8 records. First, login to your primary instance of the replica set 2 `BPBOnlineReplicaSet2` and then give the following commands in the MongoDB Shell of the primary server, as shown in the following screenshot:

To login to the MongoDB Shell of the primary server of the replica set 2 `BPBOnlineReplicaSet2`, type the following command:

```
mongo --port 47017
```

After you login successfully to the MongoDB Shell, type the following commands:

```
use BPBOnlineShardedDB
db.BPBOnlineShardedCollection.find().pretty()
```

```
c:\Program Files\MongoDB\Server\4.4\bin>mongo --port 47017
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:47017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("204faacc-c687-41ac-ab88-aab70000cdf9") }
MongoDB server version: 4.4.1
...
The server generated these startup warnings when booting:
2020-10-29T17:39:58.819+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-10-29T17:39:58.820+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the 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()
BPBOnlineReplicaSet2:PRIMARY> use BPBOnlineShardedDB
switched to db BPBOnlineShardedDB
BPBOnlineReplicaSet2:PRIMARY> db.BPBOnlineShardedCollection.find().pretty()
{
    "_id" : ObjectId("5f9959b688d353cb83ce4e88"),
    "booktitle" : "21 Internet Of Things (IOT) Experiments"
    ...
    "_id" : ObjectId("5f9959b688d353cb83ce4e89"),
    "booktitle" : "3D Game Weapons ( Modeling UV Mapping & Texturing )"
    ...
    "_id" : ObjectId("5f9959b688d353cb83ce4e8b"),
    "booktitle" : "A Practical Approach for Machine Learning and Deep Learning Algorithms"
    ...
    "_id" : ObjectId("5f9959b688d353cb83ce4e85"),
    "booktitle" : ".Net Interview Questions - 7th Revised Edition"
}
```

**Figure 19.94:** View the Documents in the Primary Instance of the Replica Set

Once your documents in the collection keep on adding in the MongoDB sharded environment, they will keep on distributing to the other replica sets too, and this is decided by `mongos`.

So, we are finished with the step-by-step implementation of MongoDB sharding and we learned the concept of MongoDB sharding in a practical manner.

1. We have showed the sharding and created the replica sets on the same machine only to make you understand sharding in a

practical manner so that it becomes easier for you to do it practically on your machine.

2. It is very important to note that in the real environment, it is done on separate servers and each replica member is created on separate server instance.
3. Live and production environments have separate servers for every instance in the replica sets and instead of using the `localhost`, we use the `IP Address` of the instances.

## Conclusion

So, in this chapter, we covered the sharding part of MongoDB. We also learned about the sharding and shaded clusters in a quick recap. We also learned the importance of the config database in the sharded environment and also learned about the shard keys. Later in this chapter, we covered the pre-configuration steps before covering the practical step-by-step method to create the sharded environment with MongoDB replica sets. We then learned how to setup the MongoDB sharded environment. We learned the complete step-by-step process of sharding in an easy-to-understand 8 steps.

## Questions

1. What do you understand by sharding?
2. What are sharded clusters?
3. What do you understand by config servers and how do they play an important role in sharding?
4. What is a shard key?
5. Explain the process of creating the sharded environment on a Windows machine?
6. Explain the process to start the `mongos`?
7. Name the command by which we can add the shards into our MongoDB sharded environment.
8. Name the command to add the sharded collection with a shard key in the MongoDB sharded environment.

9. Explain the process to verify the shaded environment that you have created with the step-by-step method explained in this chapter.

# Index

## Symbols

- 32-bit MongoDB processes [4](#)
- \$all array selector
  - about [219](#)
  - using [220](#)
- \$and logical selector
  - about [215](#)
  - using [216](#)
- \$avg aggregation expression type
  - about [245](#)
  - using [246](#)
- \$elemMatch projection operator
  - about [233](#)
  - using [233](#)
- \$exists element selector
  - about [218](#)
  - using [218](#)
- \$gt comparison selector
  - about [212](#)
  - using [212](#)
- \$in comparison selector
  - about [214](#)
  - using [214](#)
- \$last aggregation expression type
  - about [249](#)
  - using [249](#)
- \$lte comparison selector
  - about [213](#)
  - using [213](#)
- \$max aggregation expression type
  - about [246](#)
  - using [247](#)
- \$not logical selector
  - about [216](#)
  - using [217](#)
- \$ projection operator
  - about [232](#)

- using [232](#)
- \$push aggregation expression type
  - about [247](#)
  - using [248](#)
- \$regex evaluation selector
  - about [220](#)
  - using [221](#)
- \$set operator [151](#)
- \$sum aggregation expression type
  - about [243](#)
  - using [243](#)
  - with operation in group output [244](#)
  - with some other field [244, 245](#)
- \$unset operator [151](#)
- \$where operator, in MongoDB [104](#)
- \_id field
  - about [147](#)
  - used, for creating new document [147, 148](#)

## A

- administration commands and methods, MongoDB
  - collection command, creating [91, 92](#)
  - database command, creating [90, 91](#)
  - drop collection command [93, 94](#)
  - drop database command [92, 93](#)
- aggregation method
  - examples [237, 242](#)
  - syntax [236](#)
- aggregation pipeline [260](#)
- API (Application Programming Interface) [84](#)
- array data types
  - about [124](#)
  - example [124, 125](#)
- array selectors
  - \$all array selector [219](#)
  - about [204](#)
  - examples [219](#)
  - list [204](#)
- atomicity [168](#)
- atomicity, in MongoDB
  - about [168](#)
  - for multiple document transactions [168](#)

## B

backup and restore, with MongoDB  
  about [318](#)  
  backup, performing with mongodump [318-320](#)  
  database, restoring with mongorestore [320-322](#)

basic command helpers, MongoDB Shell  
  collection related help commands [79](#)  
  DB related help command [78](#)  
  General Help Command [77](#)  
  Show collections command [79](#)  
  Show databases command [78](#)  
  use <DB> Command [78](#)

basic concepts, MongoDB  
  collections [12](#)  
  document database [12](#)  
  support for multiple storage engines [12](#)  
  support for rich query language [12](#)

basic terminology, MongoDB  
  comparing, with SQL databases [13](#)  
  MongoDB collection [13](#)  
  MongoDB database [13](#)  
  MongoDB document [13](#)

binary data types  
  about [127](#)  
  example [127](#)

bitwise selectors [206](#)

boolean data types  
  about [135](#)  
  example [135, 136](#)

BSON data types [120](#)

buildInfo [314](#)

## C

collation property  
  used, for creating index [190, 191](#)

collection, MongoDB [71](#)

collection related help commands [79](#)

collStats [313, 314](#)

column-oriented databases  
  about [11](#)  
  examples [11](#)

comment selector [206](#), [207](#)  
comparison and sort order  
    example [137-139](#)  
    using [137](#)  
comparison selectors  
    \$gt comparison selector [212](#)  
    \$in comparison selector [214](#)  
    \$lte comparison selector [213](#)  
    about [202](#)  
    examples [212](#)  
    list [202](#), [203](#)  
compound index  
    about [184](#)  
    creating, in MongoDB collection [184](#)  
connect() method [107](#)  
consistency [169](#)  
consistency, in MongoDB  
    about [169](#)  
    eventual consistency [169](#)  
cross-platform [4](#)  
CRUD (Create, Read, Update, and Delete) operations [74](#)  
cursor.count() method [114](#), [115](#)  
cursor, in MongoDB [114](#)  
cursor.pretty() method [115](#)  
cursor.sort() method [115](#), [116](#)

## D

database [68](#)  
database authentication [97](#)  
database lock  
    about [87](#)  
    operations types [88](#)  
database locks operations, MongoDB [88](#)  
Database Management Systems (DBMS)  
    about [68](#)  
    main function [68](#)  
database storage engine [84](#)  
Data Query Languages  
    MongoDB Query Language (MQL) [68](#)  
    Structured Query Language (SQL) [68](#)  
data types [120](#)  
data types, MongoDB

about [120](#), [121](#)  
array data types [124](#), [125](#)  
binary data types [127](#)  
boolean data types [135](#)  
date data types [129](#), [130](#)  
Decimal128 data type [136](#)  
double data types [123](#), [124](#)  
integer data types [121](#), [122](#)  
JavaScript data types (without scope) [132](#)  
JavaScript data types (with scope) [133](#)  
null data types [130](#), [131](#)  
object data types [125](#), [126](#)  
ObjectId data types [127](#), [128](#)  
regular expression data types [131](#), [132](#)  
string data types [122](#), [123](#)  
timestamp data types [134](#)  
date data types  
  about [129](#)  
  example [129](#), [130](#)  
db.collection.bulkWrite() method  
  definition [162](#)  
    used, for performing bulk write in MongoDB collection [163-165](#)  
db.collection.count() method [111](#), [112](#)  
db.collection.createIndex() method  
  about [182](#)  
  definition [183](#)  
    used, for creating index in MongoDB collection [183](#)  
db.collection.deleteMany() method  
  definition [160](#)  
    used, for deleting multiple documents in MongoDB collection [161](#)  
db.collection.deleteOne() method  
  definition [159](#)  
    used, for deleting single document in MongoDB collection [160](#)  
db.collection.dropIndexes() method  
  definition [194](#)  
    used, for deleting all indexes in MongoDB collection [196](#)  
    used, for deleting multiple index in MongoDB collection [195](#)  
    used, for deleting multiple index in MongoDB collection with array type values as  
      parameter [195](#)  
db.collection.dropIndex() method  
  definition [193](#)  
    used, for deleting index in MongoDB collection [194](#)  
db.collection.drop() method [114](#)

db.collection.find() method  
definition [149](#)  
pretty() method, using with [150](#)  
used, for reading documents in MongoDB collection without Query [149](#)  
used, for reading documents in MongoDB collection with Query [149](#), [150](#)

db.collection.getIndexes() method  
about [192](#)  
used, for creating indexes in MongoDB collection [192](#), [193](#)

db.collection.insertMany() method  
definition [146](#)  
used, for creating multiple documents in MongoDB collection [146](#), [147](#)

db.collection.insert() method  
definition [142](#)  
used, for creating multiple documents in MongoDB collection [143](#), [144](#)  
used, for creating single document in MongoDB collection [142](#), [143](#)

db.collection.insertOne() method  
definition [144](#)  
used, for creating single document in MongoDB collection [145](#)

db.collection.remove() method  
definition [157](#)  
justOne option [158](#)  
used, for deleting multiple documents in MongoDB collection [159](#)  
used, for deleting single document in MongoDB collection [158](#)

db.collection.stats() method [112](#)

db.collection.totalSize() method [113](#)

db.collection.updateMany() method  
definition [155](#)  
used, for updating multiple documents in MongoDB collection [155](#), [156](#)

db.collection.update() method  
definition [151](#)  
used, for updating multiple documents in MongoDB collection [153](#), [154](#)  
used, for updating single document in MongoDB collection [152](#), [153](#)

db.collection.updateOne() method  
definition [154](#)  
used, for updating single document in MongoDB collection [154](#), [155](#)

db.collection.validate() method [113](#)

db.getMongo() method [109](#)

db.hostInfo() method [109](#)

DB related help command [78](#)

db.serverStatus() method [110](#), [111](#)

dbStats [313](#)

db.stats() method [110](#)

Decimal128 data type [136](#)

delete document command [96](#), [97](#)  
distributed operations and queries  
about [172](#)  
read operations, on replica sets [173](#)  
read operations on sharded clusters [173](#)  
write operations, on replica sets [173](#)  
write operations on sharded clusters [173](#)  
document databases  
about [3](#), [11](#)  
examples [11](#)  
double data types  
about [123](#)  
example [123](#), [124](#)

## E

element selectors  
\$exists element selector [218](#)  
about [204](#)  
examples [217](#)  
list [204](#)  
encrypted storage engine [86](#)  
evaluation selectors  
\$regex evaluation selector [220](#)  
about [205](#)  
examples [220](#)  
list [205](#)  
eventual consistency  
about [169](#)  
example [169](#)  
expression types, MongoDB aggregate operation  
\$avg aggregation expression type [245](#)  
\$last aggregation expression type [249](#)  
\$max aggregation expression type [246](#)  
\$push aggregation expression type [247](#)  
\$sum aggregation expression type [243](#)  
\$sum aggregation expression type, with operation in group output [244](#)  
\$sum aggregation expression type, with some other field [244](#), [245](#)

## F

flexibility [5](#)

## G

General Help Command [77](#)  
geospatial index [186](#)  
geospatial selectors [206](#)  
getLog [317, 318](#)  
graph databases  
    about [11](#)  
    examples [11](#)  
GUI (Graphical User Interface) tool [272](#)

## H

hashed index [186](#)  
hostInfo [315](#)

## I

indexes [176](#)  
indexing, in MongoDB  
    about [176](#)  
    benefits [177](#)  
    collation property [190](#)  
    default \_id index [177, 178](#)  
    index, creating [182, 189](#)  
    restrictions [196](#)  
index properties  
    about [186](#)  
    partial index property [187](#)  
    sparse index property [188](#)  
    TTL index [188](#)  
    unique index property [186](#)  
index types, in MongoDB  
    compound index [184](#)  
    multikey index [185](#)  
    single field index [183](#)  
    special types of index [186](#)  
    text index [185](#)  
in-memory storage engine [86](#)  
insert document command [94, 95](#)  
integer data types  
    example [121, 122](#)

## J

JavaScript data types (without scope)

about [132](#)

example [132](#), [133](#)

JavaScript data types (with scope)

example [133](#), [134](#)

JavaScript, in MongoDB

about [104](#)

Server Side JavaScript [104](#)

JavaScript Object Notation (JSON)

about [3](#)

example [3](#)

## K

key-value paired databases

about [11](#)

examples [11](#)

## L

Linux

MongoDB installation [36](#)

listCommands [315](#), [316](#)

logical selectors

\$and logical selector [215](#)

\$not logical selector [216](#)

examples [215](#)

list [203](#)

## M

macOS

MongoDB installation [55](#)

map-reduce in MongoDB [104](#)

mapReduce() method

about [250](#)

examples [251](#), [255](#)

output [257](#)

running [259](#)

using [256](#), [258](#)

working [251](#)

min and max keys [136](#)  
MongoDB  
    about [2](#)  
    administration commands and methods [90](#)  
    backup and restore [318](#)  
    basic concepts [12](#)  
    classified, as NoSQL database [5](#)  
    connecting to, on Windows [32](#), [33](#)  
    cross-platform [4](#)  
    cursor [114](#)  
    database locks operations [88](#)  
    data types [120](#)  
    definitions [2](#)  
    diagnosing [303](#)  
    document database [3](#)  
    flexibility [5](#)  
    indexing [176](#)  
    index properties [186](#)  
    index types [183](#)  
    JavaScript [104](#)  
    key pointers [2](#)  
    min and max keys [136](#)  
    monitoring [303](#)  
    officially supported languages [105](#)  
    projection [224](#)  
    query and write operation commands and methods [94](#)  
    query selectors [202](#)  
    role-based authentication [98](#)  
    scalable [4](#)  
    storage engine, types [85](#)  
    user authentication and role based commands and methods [97](#)  
MongoDB \$group operator  
    about [242](#)  
    using [242](#)  
MongoDB aggregation  
    about [236](#)  
    aggregation method [236](#)  
    examples [242](#)  
MongoDB aggregation pipeline  
    about [260](#)  
    aggregate() method [266-268](#)  
    aggregate() method, with \$out [268](#), [269](#)  
    framework [261](#)

using, with practical examples [261-266](#)

MongoDB architecture

- about [6, 7](#)
- MongoDB data platform [8](#)
- NoSQL database architecture [6, 7](#)

MongoDB backup

- performing, mongodump used [318-320](#)

MongoDB bulk write operations

- about [161](#)
- db.collection.bulkWrite() method [162](#)

MongoDB clients [80](#)

MongoDB collection methods

- db.collection.count() [111, 112](#)
- db.collection.drop() [114](#)
- db.collection.stats() [112](#)
- db.collection.totalSize() [113](#)
- db.collection.validate() [113](#)

MongoDB collections

- about [70, 71](#)
- example [71](#)
- table in RDBMS [70](#)

MongoDB Community Edition

- free software foundation's GNU AGPL v3.0 [16](#)
- installing, on Windows 2010 [19-30](#)
- post installation checks [31, 32](#)
- server side public license [16](#)
- versus, MongoDB Enterprise Advanced Edition [17-19](#)

MongoDB Community Edition installation, on Linux

- browser method [38-43](#)
- MongoDB clients, installing [44-46](#)
- performing [36-38](#)
- Shell commands, used [47-49](#)
- shell method [44](#)

MongoDB Community Edition installation, on macOS

- Homebrew, installing [57-59](#)
- MongoDB, connecting to [63, 64](#)
- MongoDB, starting [62](#)
- performing [56-62](#)

MongoDB Compass

- about [271, 272](#)
- downloading [273-276](#)
- features [272](#)
- installing [273](#)

installing, on Windows [273-279](#)  
MongoDB Server, connecting with [281-284](#)  
post installation checks [280](#)  
practical examples [284](#)

MongoDB connection methods  
about [106](#)  
`connect()` [107](#)  
`Mongo.getDB()` [108](#)  
`Mongo()` [107, 108](#)

MongoDB create operations  
about [142](#)  
`db.collection.insertMany()` method [146](#)  
`db.collection.insert()` method [142](#)  
`db.collection.insertOne()` method [144](#)  
`_id` field [147](#)  
ordered option [148](#)

MongoDB CRUD operations  
about [141](#)  
MongoDB bulk write operations [161](#)  
MongoDB create operations [142](#)  
MongoDB delete operations [157](#)  
MongoDB read operations [148](#)  
MongoDB update operations [151](#)

MongoDB cursor methods  
about [114](#)  
`cursor.count()` [114, 115](#)  
`cursor.pretty()` [115](#)  
`cursor.sort()` [115, 116](#)

MongoDB data  
exporting, mongoexport used [322-324](#)  
importing, mongoimport used [324-326](#)

MongoDB database  
about [68, 70](#)  
NoSQL database [69](#)  
restoring, mongorestore used [320-322](#)

MongoDB database methods  
about [109](#)  
`db.getMongo()` [109](#)  
`db.serverStatus()` [110, 111](#)  
`db.stats()` [110](#)

MongoDB database platform  
about [8](#)  
advantages [8](#)

MongoDB delete operations  
db.collection.deleteMany() method [160](#)  
db.collection.deleteOne() method [159](#)  
db.collection.remove() method [157](#)

MongoDB documents  
about [72](#)  
example [72, 73](#)

MongoDB editions  
MongoDB Community Edition [16](#)  
MongoDB Enterprise Advanced Edition [16](#)  
overview [16](#)

MongoDB Enterprise Advanced Edition  
features [16, 17](#)

MongoDB heartbeats  
about [333](#)  
new primary member, automatic election [334](#)

MongoDB Inc Compass  
URL [80](#)

MongoDB index  
creating [189](#)  
creating, with collation property [190](#)  
deleting [193](#)  
index information, viewing [191](#)  
restrictions [196](#)  
using [189, 190](#)

MongoDB installation, in macOS  
about [55](#)  
MongoDB Community Edition, installing [56, 57](#)

MongoDB installation setup, on Linux  
about [36](#)  
MongoDB Community Edition, installing [36](#)  
MongoDB, connecting to [50-52](#)  
MongoDB, starting on Linux [50](#)

MongoDB installation setup, on Windows  
about [15](#)  
MongoDB Community Edition, installing [19-30](#)

MongoDB intermediate concepts  
about [167](#)  
atomicity [168](#)  
consistency [169](#)  
distributed operations and queries [172](#)  
replication [170](#)  
sharding [171](#)

MongoDB locks [87](#)  
MongoDB methods [105](#)  
MongoDB read operations  
    about [148](#)  
    db.collection.find() method [149](#)  
MongoDB replication  
    verifying, data used [352-355](#)  
MongoDB replication, on Windows  
    step-by-step process [335-352](#)  
MongoDB security  
    about [326](#), [327](#)  
    authentication, enabling [327](#)  
    communication channel encryption [328](#)  
    data encryption [328](#)  
    firewalls, using [328](#)  
    role-based authorization, using [327](#)  
    security audits, performing regularly [328](#)  
MongoDB sharding, on Windows  
    existing MongoDB services on Windows, stopping [363](#), [364](#)  
    first replica set, creating [365-377](#)  
    MongoDB, adding in shard keys [414-417](#)  
    MongoDB, starting in sharded environment as mongos [413](#), [414](#)  
    replica set of config servers, creating in sharded environment [401-413](#)  
    second replica set, creating [377-389](#)  
    starting [362](#), [363](#)  
    third replica set, creating [389-401](#)  
    verifying, data used [417-421](#)  
MongoDB Shell  
    about [74](#)  
    basic command helpers [77](#)  
    commands [76](#), [77](#)  
    connecting to [33](#), [34](#), [74](#), [75](#), [105](#), [106](#)  
    exiting from [76](#)  
    history, of commands [80](#)  
MongoDB Shell methods  
    about [103](#)  
        MongoDB collection methods [111](#)  
        MongoDB connection methods [106](#)  
        MongoDB cursor methods [114](#)  
        MongoDB database methods [109](#)  
MongoDB tools and utilities  
    buildInfo [314](#)  
    collStats [313](#), [314](#)

dbStats [313](#)  
getLog [317](#)  
hostInfo [315](#)  
installation, verifying [308](#), [309](#)  
installing [304-308](#)  
listCommands [315](#), [316](#)  
mongostat [309](#), [310](#)  
mongotop [310-312](#)  
ping [316](#), [317](#)  
serverStatus [312](#), [313](#)  
MongoDB update operations  
  db.collection.updateMany() method [155](#)  
  db.collection.update() method [151](#)  
  db.collection.updateOne() method [154](#)  
  multi option [157](#)  
  upsert option [156](#)  
mongod process  
  about [294](#)  
  managing [294](#)  
MongoDB service, in Windows [294-297](#)  
MongoDB services, stopping from command line [301](#), [302](#)  
MongoDB services, stopping from Windows service manager [300](#)  
  running, from command prompt [297-299](#)  
Mongo.getDB() method [108](#)  
Mongo() method [107](#)  
mongostat [309](#), [310](#)  
mongotop [310-312](#)  
multikey index  
  about [185](#)  
  creating, in MongoDB collection [185](#)  
multi option [157](#)

## N

NoSQL Booster  
  URL [80](#)  
NoSQL database  
  about [5](#), [6](#), [10](#), [69](#)  
  comparing, with SQL database [9](#)  
  database management systems [10](#)  
NoSQL database management systems  
  about [10](#)  
  column-oriented databases [11](#)

document databases [11](#)  
graph databases [11](#)  
key-value paired databases [11](#)  
NoSQL Manager  
  URL [80](#)  
null data types  
  about [130](#), [131](#)  
  example [130](#)

## O

object data types  
  about [125](#)  
  example [125](#), [126](#)  
ObjectId data types  
  about [127](#)  
  example [128](#)  
Object-Relation Mapping (ORM) [9](#)  
officially supported languages, MongoDB [105](#)  
ordered option [148](#)

## P

partial index  
  about [187](#)  
  creating, in MongoDB collection [187](#), [188](#)  
ping [316](#)  
pipeline [260](#)  
practical examples, of MongoDB Compass  
  collections, browsing in database [284](#), [285](#)  
  CRUD operations, performing in documents [288](#)  
  document, editing [289](#), [290](#)  
  documents, browsing in database [287](#)  
  new collection, creating in database [285](#), [286](#)  
pretty() method  
  used, for reading documents in MongoDB collection [150](#)  
  using [150](#)  
projection, in MongoDB  
  about [224](#), [228](#)  
  examples [229](#), [230](#)  
  Hide Specific Fields [229](#), [230](#)  
  Show Only Specific Fields [228](#), [229](#)  
  Show Only Specific Fields and Hide \_id Field [230](#)

using [228](#)  
projection operators  
    \$elemMatch projection operator [233](#)  
    \$ projection operator [232](#)  
examples [232](#)  
list [231](#)  
Public Key Infrastructure (PKI) certificates [17](#)

## Q

query and write operation commands and methods, MongoDB  
    delete document command [96, 97](#)  
    insert document command [94, 95](#)  
    read document command [95, 96](#)  
query selectors  
    about [202](#)  
    array selectors [204](#)  
    bitwise selectors [206](#)  
    comment selector [206](#)  
    comparison selectors [202](#)  
    element selectors [204](#)  
    evaluation selectors [205](#)  
    examples [207](#)  
    geospatial selectors [206](#)  
    logical selectors [203](#)  
    using [207-211](#)

## R

read document command [95, 96](#)  
regular expression data types [131, 132](#)  
relational database [68, 69](#)  
Relational Database Management System (RDBMS)  
    about [5, 70](#)  
    row and column [72](#)  
    table [70](#)  
replica sets  
    about [170, 332](#)  
    creating [170](#)  
replication  
    about [170, 332](#)  
    benefits [170, 332](#)  
    pre-configuration steps [334, 335](#)

restrictions, MongoDB index [196](#)  
RoboMongo  
    URL [80](#)  
role-based access control [98](#)  
role-based authentication, in MongoDB [98-100](#)

## S

scalability [4, 5](#)  
schema [5](#)  
schema-less databases  
    advantages [5](#)  
Server Side JavaScript, in MongoDB  
    \$where operator [104](#)  
    about [104](#)  
    map-reduce [104](#)  
serverStatus [312](#)  
sharded clusters  
    about [172, 359](#)  
    config database [360](#)  
    read and write operations [360](#)  
sharding  
    about [171, 358](#)  
    benefits [171, 358, 359](#)  
    pre-configuration steps [361, 362](#)  
shard key [360](#)  
Show collections command [79](#)  
Show databases command [78](#)  
single field index  
    about [183](#)  
    creating, in MongoDB collection [184](#)  
sparse index  
    about [188](#)  
    creating, in MongoDB collection [188](#)  
special types of indexes  
    about [186](#)  
    geospatial index [186](#)  
    hashed index [186](#)  
storage engines [84](#)  
storage engines, MongoDB  
    comparison [86, 87](#)  
    encrypted storage engine [86](#)  
    in-memory storage engine [86](#)

third-party storage engines [86](#)  
types [85](#)  
WiredTiger storage engine [85](#)  
string data types  
about [122](#)  
example [122, 123](#)  
Studio 3T  
URL [80](#)

## T

table, in RDBMS [70](#)  
text index  
about [185](#)  
creating, in MongoDB collection [186](#)  
third-party pluggable storage engines [86](#)  
timestamp data types  
about [134](#)  
example [134, 135](#)  
Transport Layer Security (TLS) [17](#)  
TTL index  
about [188](#)  
creating, in MongoDB collection [189](#)

## U

Ubuntu. See Linux  
unique index  
about [186](#)  
creating, in MongoDB collection [187](#)  
upsert option [156, 157](#)  
use <DB> Command [78](#)  
user authentication and role based commands and methods  
database authentication [97](#)  
role-based access control [98](#)

## W

Windows  
MongoDB installation [15](#)  
WiredTiger storage engine [85](#)