

Node.js Section 11

Merhabalar bugün serimizde devam ediyoruz ve task manager projesi yapacağız.

Node developerlar genelde MongoDB kullanır. MySql veya Postgres de kullanabilirler.

MongoDB no sql bir veritabanıdır MySql ve Postgresden farklı olarak.

MongoDB ilişkisel veri tabanı değilken MySql ve Postgres ilişkisel veri tabanlarıdır.

MySQL ve Postgres de tablolar varken MongoDB de collectionlar vardır tablolar yerine.

Tablolarda row, record varken collectionlarda ise documents vardır.

<https://www.mongodb.com> sitesine gidelim ve kendimize bir hesap oluşturalım uzakta çalışan bir veri tabanı kullanacağımız şimdiden.

Hesabımız yoksa hesap oluşturalım.

The screenshot shows the MongoDB Atlas setup personalization interface. At the top, it says "Welcome to Atlas. Let's build something great." Below that, there are two main sections: "GETTING TO KNOW YOU" and "GETTING TO KNOW YOUR PROJECT".

GETTING TO KNOW YOU:

- What is your primary goal? Learn MongoDB
- How long have you been developing software with MongoDB? I've never developed software with MongoDB before

GETTING TO KNOW YOUR PROJECT:

- What programming language are you primarily building on MongoDB with? JavaScript / Node.js
- What type(s) of data will your project use? You can choose as many as you want. Other
- Please specify the types of data. [Optional] Input text
- Will your application include any of the following architectural models? You can choose as many as you want. Not sure... X

At the bottom right, there is a "Finish" button and a help icon.

Kayıt olduktan sonra önümüze bunlar geliyor kendimize uygun seçenekleri seçerek başlayalım.

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

M10 **\$0.09/hour**
For production applications with sophisticated workload requirements.

Serverless **\$0.11/1M reads**
For application development and testing, or workloads with variable traffic.

M0 **Free**
For learning and exploring MongoDB in a cloud environment.

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Name: ClusterLearnMongoDB

Automate security setup

Preload sample dataset

Provider: AWS

Region: I'll do this later

[Go to Advanced Configuration](#) [Create Deployment](#)

STORAGE 10 GB **RAM** 2 GB **vCPU** 2 vCPUs

STORAGE Up to 1TB **RAM** Auto-scale **vCPU** Auto-scale

STORAGE 512 MB **RAM** Shared **vCPU** Shared

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Name: ClusterLearnMongoDB

Automate security setup

Preload sample dataset

Provider: AWS

Region: Bahrain (me-south-1) ★

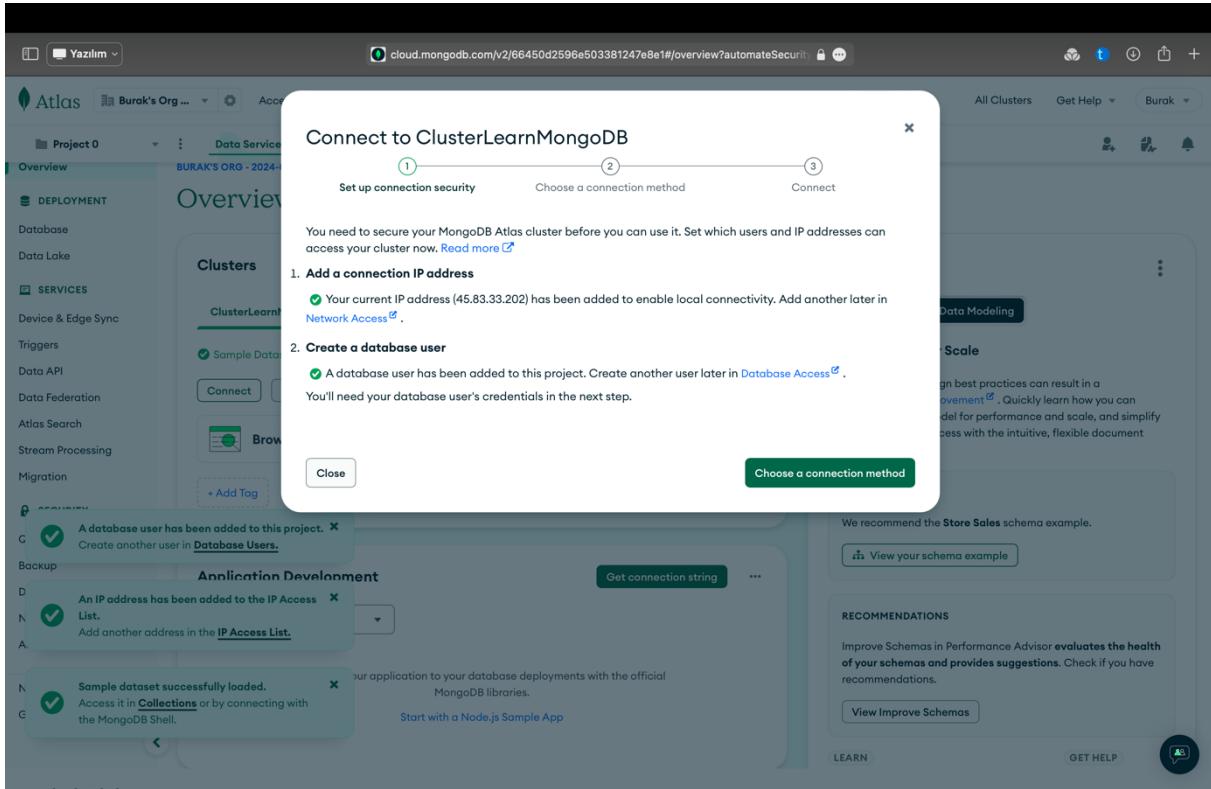
Tag (optional): Create your first tag to categorize and label your resources; more tags can be added later. [Learn more](#).

Select or enter key : Select or enter value

I'll do this later [Go to Advanced Configuration](#) [Create Deployment](#)

Bunları seçerek devam ediyoruz ücretsiz olanı seçelim ki ücret ödemeyelim 😊

Önümüzde database user kullanıcı adı ve şifresi çıkacak onları kaydedelim bize lazım olacaklar.



Connection methodu seçelim.

Şimdi ise mongodb yi indirmemiz kurmamız için yol gösteriyor.

Projemizi dizinimizde oluşturalım.

Section11 adında klasör oluşturduğum ve içinde npm init -y yazarak kuralım.

Daha sonra npm install mongodb yazarak mongodb yi kuralım zaten mongodb ekranında gösterecek bize.

The screenshot shows the MongoDB Atlas Cluster Overview page. On the left, there's a sidebar with various project and cluster management options. The main area is titled "Connecting with MongoDB Driver". It includes three steps: 1. Select your driver and version (Node.js, 5.5 or later), 2. Install your driver (npm install mongodb), and 3. Add your connection string into your application code. A red box highlights the "View full code sample" button. Below it is a code snippet for a Node.js script connecting to a MongoDB cluster. The "Done" button is at the bottom right.

Bu kısmı aktif edip çıkan kodu dosyamıza yapıştırılarım.

Mongodb.js adında dosya oluşturup içine yapıştırılarım.

Username var zaten oraya ise bize başta gelen şifreyi yapıştırılarım.

```
Section11 - mongodb.js

const { MongoClient, ServerApiVersion } = require("mongodb")
const uri =
  "mongodb+srv://developerburakgultool:IDH93vYwUNkiMID0@clusterlearnmongodb.wjuie0k.mongodb.net/?retryWrites=true&w=majority&appName=ClusterLearnMongoDB"

// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {
  serverApi: {
    version: ServerApiVersion.v1,
    strict: true,
    deprecationErrors: true
  }
})

async function run() {
  try {
    // Connect the client to the server (optional starting in v4.7)
    await client.connect()
    // Send a ping to confirm a successful connection
    await client.db("admin").command({ ping: 1 })
    console.log(
      "Pinged your deployment. You successfully connected to MongoDB!"
    )
  } finally {
    // Ensures that the client will close when you finish/error
    await client.close()
  }
}
run().catch(console.dir)
```

Bu kodu çalışıralım ve

Pinged your deployment. You successfully connected to MongoDB! Yazısını console da görelim.

Section11 - mongodb.js

```
const db = client.db(databaseName)
  const collection = db.collection("users")
  await db
    .collection("users")
    .insertOne({ name: "BURAK", age: 23 })
    .then((result) => {
      console.log(result)
    })
    .catch((error) => {
      console.log(error)
    })
```

Bu kodu run fonksiyonu içine yazarak veri ekleyelim collectionumuza ve çalışıralım.

Bu hali ile hata alırız çünkü database name olmadığı için ve mongodb de o name e sahip database oluşturmadığımızdan nul döner consoleda.

The screenshot shows the MongoDB Atlas web interface. On the left, there's a sidebar with various sections like Project Overview, Deployment, Services, Security, and Goto. The main area is titled 'ClusterLearnMongoDB' and shows the 'sample_mflix.comments' collection. It displays storage details (Storage Size: 5.69MB, Logical Data Size: 11.44MB, Total Documents: 4109, Indexes Total Size: 0.11MB) and a list of documents. A red box highlights the '+ Create Database' button in the sidebar.

Buradan “task-manager” adında database oluşturalım.

```
Section1 - mongodb.js

const databaseName = "task-manager"
const { MongoClient, ServerApiVersion } = require("mongodb")
const uri =
  "mongodb+srv://developerburakgultool:IDH93vYwUNkiMID@clusterlearnmongodb.wjuie0k.mongodb.net/?retryWrites=true&w=majority&appName=ClusterLearnMongoDB"

// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {
  serverApi: {
    version: ServerApiVersion.v1,
    strict: true,
    deprecationErrors: true
  }
})

async function run() {
  try {
    // Send a ping to confirm a successful connection
    await client.db("admin").command({ ping: 1 })

    const db = client.db(databaseName)
    const collection = db.collection("users")
    await db
      .collection("users")
      .insertOne({ name: "BURAK", age: 23 })
      .then((result) => {
        console.log(result)
      })
      .catch((error) => {
        console.log(error)
      })
  } finally {
    await client.close()
  }
}
run().catch(console.dir)
```

Tüm kodumuzu da bu hale getirip çalıştırıralım.

The screenshot shows the MongoDB Atlas Data Services interface. On the left, there's a sidebar with navigation links like Overview, Deployment, Database, Services, Security, and a Goto button. The main area is titled 'task-manager.users' and shows storage details: STORAGE SIZE: 20KB, LOGICAL DATA SIZE: 47B, TOTAL DOCUMENTS: 1, INDEXES TOTAL SIZE: 20KB. It includes tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A search bar at the top right says 'View your schema example(s)' and 'VISUALIZE YOUR DATA'. Below the search bar is an 'INSERT DOCUMENT' button. A query result table shows one document: '_id: ObjectId('664514afc66da3fb10476979'), name : "BURAK", age : 23'. There are 'Reset', 'Apply', and 'Options' buttons at the bottom of the query table.

Gördüğümüz üzere verimiz eklendi.

InsertOne yerine insertMany ile birden fazla veri ekleyebiliriz.

```
Section11 - mongodb.js

const db = client.db(databaseName)
  const collection = db.collection("users")
  await db
    .collection("users")
    .insertMany([
      { name: "MERYEM", age: 1 },
      { name: "BATUHAN", age: 20 }
    ])
    .then((result) => {
      console.log(result)
    })
    .catch((error) => {
      console.log(error)
    })
}
```

Bu kod ise birde fazla veri eklememizi sağlar.

The screenshot shows the MongoDB Atlas interface for a project named 'Project 0'. On the left, there's a sidebar with sections for Deployment, Services, and Security. Under 'Deployment', 'Database' is selected, showing 'sample_mflix' and 'task-manager' databases, with 'users' selected under 'task-manager'. The main panel displays the results of a query: 'Find' results for the 'users' collection. The results show three documents:

```
_id: ObjectId('664514afc66da3fb10476979')
name : "BURAK"
age : 23

_id: ObjectId('664515e7ab5959d21d00d8b1')
name : "MERYEM"
age : 1

_id: ObjectId('664515e7ab5959d21d00d8b2')
name : "BATUHAN"
age : 20
```

Below the results, there are buttons for 'Reset', 'Apply', and 'Options'.

Modal Title

Gördüğümüz üzere veriler eklendi.

Şimdi de başka bir collection'a birden fazla veri ekleyelim. task collectionu olsun.

Section11 - mongodb.js

```
const db = client.db(databaseName)
  const collection = db.collection("tasks")
  await db
    .collection("tasks")
    .insertMany([
      {
        description: "Task 1",
        completed: true
      },
      {
        description: "Task 2",
        completed: false
      },
      {
        description: "Task 3",
        completed: true
      }
    ])
    .then((result) => {
      console.log(result)
    })
    .catch((error) => {
      console.log(error)
    })
  }
```

Şimdi de bu kodla değiştirelim önceki kodu ve çalışıralım.

The screenshot shows the MongoDB Atlas Data Services interface. On the left, a sidebar lists various services and security options. The main area displays the 'task-manager.tasks' collection within the 'task-manager' database. The collection has 12 documents and a total size of 36KB. The results page shows three documents:

```
_id: ObjectId('664517cf65083d8597f09ae2')
description: "Task 1"
completed: true

_id: ObjectId('664517cf65083d8597f09ae3')
description: "Task 2"
completed: false

_id: ObjectId('664517cf65083d8597f09ae4')
description: "Task 3"
completed: true
```

Gördüğümüz üzere mongodb bizim için verilerimize id ekledi.

Bu id değeri benzersiz olmasını sağlayacaktır.

İd değerinin özelliklerine erşiebiliriz merak edenler id. Diyerek erişebilirler.

Section11 - mongodb.js

```
const db = client.db(databaseName)
  const collection = db.collection("users")
  await collection
    .findOne({ name: "BURAK" })
    .then((result) => {
      console.log(result)
    })
    .catch((error) => {
      console.log(error)
    })
} finally {
  await client.close()
}
```

Bu kod ise users collectionundan name değeri “BURAK” olanı çeker.Eğer birden fazla “BURAK” var ise ilk bulduğu değeri çeker.

Şimdi yaşı farklı olan bir “BURAK” ekleyelim ve çekelim ve görelim yaşı 23 olanı getireceğini.

Section11 - mongodb.js

```
const db = client.db(databaseName)
  const collection = db.collection("users")
  await collection.insertOne({
    name: "BURAK",
    age: 25
})
```

Yaşı 25 olan bir “BURAK” adlı user ekledik.(Eğer ssl hatası alırsanız MongoDB den kullandığınız ip adresini eklemelisiniz erişmek için.)

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with sections like Project 0, Overview, Deployment, Services, and Security. The main area is titled 'task-manager.users' and shows four documents. Each document has fields: _id, name, and age. The first document is for 'BURAK' (age 23), the second for 'MERYEM' (age 1), the third for 'BATUHAN' (age 20), and the fourth for another 'BURAK' (age 25). A search bar at the top has the query 'name : "BURAK"'.

Düzen verimizi de ekledik şimdi çekelim.

```
{  
  _id: new ObjectId('664514afc66da3fb10476979'),  
  name: 'BURAK',  
  age: 23  
}
```

Gördüğümüz üzere ilk bulduğu veriyi getirdi.

Yaşı 25 olan başka birini ekleyelim ve yaşı 25 olanları getirmesini isteyelim.

```
Section11 - mongodb.js

const db = client.db(databaseName)
  const collection = db.collection("users")
  await collection
    .insertOne({
      name: "Mehmet",
      age: 25
    })
    .then((result) => {
      console.log(result)
    })
    .catch((error) => {
      console.log(error)
    })
```

İle Mehmeti de ekledik.

```
Section11 - mongodb.js

const db = client.db(databaseName)
  const collection = db.collection("users")
  await collection
    .find({ age: 25 })
    .toArray()
    .then((result) => {
      console.log(result)
    })
    .catch((error) => {
      console.log(error)
    })
```

Bu kod ile yaşı 25 olanları çekebiliriz.

```
[base] tumburkoglu@BURAKS-MACBOOK-AIR: Section11 ~ node
[
  {
    _id: new ObjectId('6645fb043d4da934f4f81b94'),
    name: 'BURAK',
    age: 25
  },
  {
    _id: new ObjectId('6645fc887e121460426d08a2'),
    name: 'Mehmet',
    age: 25
  }
]
```

Peki taskların completed özelliği true olanları çekerek olsak nasıl yaparız.

```
Section11 - mongodb.js

const db = client.db(databaseName)
const collection = db.collection("tasks")
await collection
  .find({ completed: true })
  .toArray()
  .then((result) => {
    console.log(result)
  })
  .catch((error) => {
    console.log(error)
  })
```

```
[{
  {
    _id: new ObjectId('664517cf65083d8597f09ae2'),
    description: 'Task 1',
    completed: true
  },
  {
    _id: new ObjectId('664517cf65083d8597f09ae4'),
    description: 'Task 3',
    completed: true
  }
]
```

Gördüğümüz üzere bu şekilde de completed olanları çekiyoruz.