

Furkan Kazım ÇAM

21360859036

11.Hafta Rapor

MongoDB nedir?

MongoDB, açık kaynaklı bir NoSQL veritabanı yönetim sistemidir. İsmi "humongous" (çok büyük) ve "database" (veritabanı) kelimelerinin kısaltmasıdır. MongoDB, doküman odaklı bir veritabanıdır, yani ilişkisel veritabanlarından farklı olarak verileri JSON benzeri belgelere (dokümanlara) saklar. Her belge kendi alanlarını içerir ve bu alanlar farklı veri tiplerini (metin, sayı, dizi, vb.) içerebilir. MongoDB'nin esnek yapısı, veri modelini değiştirme ve uygulama gereksinimlerine daha iyi uyum sağlama yeteneği sağlar.

MongoDB'nin bazı ana özellikleri şunlardır:

- **Yüksek Performans:** Verilerin hızlı bir şekilde yazılması ve okunması için optimize edilmiştir.
- **Yüksek Esneklik:** Esnek bir veri modeli ile farklı türde verileri saklayabilir.
- **Yüksek Ölçeklenebilirlik:** Büyük ölçekte veri depolama ve işleme gereksinimlerini karşılamak için ölçeklenebilir bir yapı sunar.
- **Yüksek Erişilebilirlik:** Verilerin kesintisiz erişimini sağlamak için çeşitli yüksek erişilebilirlik özelliklerine sahiptir.
- **Belge Tabanlı Model:** Verileri JSON benzeri belgelere saklar ve ilişkisel veritabanlarından farklı bir veri modeli sunar.
- **Karmaşık Sorgular:** Zengin sorgu diline ve dizinleme özelliklerine sahiptir, bu da karmaşık sorguların yapılmasını sağlar.

MongoDB, web uygulamaları, mobil uygulamalar, IoT (nesnelerin interneti) projeleri ve büyük veri analitiği gibi çeşitli alanlarda kullanılmaktadır.

Bu veri tabanını kullanmak için bir exe indirmemize gerek yok. Bulut tabanlı olduğu için bu şekildedir. Peki nasıl çalıştıracağız?

<https://www.mongodb.com/> sitesine gidiniz ve ücretsiz deneme ile üyelik oluşturunuz. Bazı sorular soracaktır genel ne amaçlı kullanacağınıza dair.

Help us tailor your experience by taking a minute to answer the questions below.

GETTING TO KNOW YOU

What is your primary goal?

Learn MongoDB

How long have you been developing software with MongoDB?

1-6 months experience

☐ AI/ML Enriched

☐ Mobile

☐ IoT / edge computing

☐ Search engine

☒ **Microservices**

☐ Event-driven

☐ Serverless / function-as-a-service

Microserv... ✕ | ✕

Finish

For production applications with sophisticated workload requirements.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

For application development and testing, or workloads with variable traffic.

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

For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

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

Name
You cannot change the name once the cluster is created.

NodeJS

☒ Automate security setup ⓘ

☒ Preload sample dataset ⓘ

Provider

☐ AWS ☒ Google Cloud ☐ Azure

Region

Belgium (europe-west1) ★ 🌱

I'll do this later

Go to Advanced Configuration

Create Deployment

Bağlanacağımız serverı, providerı ve cluster name i veriyoruz ve create deployment'e basıyoruz.

Connect to NodeJS



You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

1. Add a connection IP address

✓ Your current IP address (95.2.10.76) has been added to enable local connectivity. Add another later in [Network Access](#).

2. Create a database user

This first user will have [atlasAdmin](#) permissions for this project.

We autogenerated a username and password. You can use this or create your own.

ⓘ You'll need your database user's credentials in the next step. Copy the database user password.

Username

furkankazimc

Password

MSmvB3r3fSVqV4fs

HIDE

Copy

Create Database User

Close

Choose a connection method

Burada bir kullanıcı ve şifre oluşturmamızı istiyor. Şifreyi bir kez daha görüntüleyemeceğiz o yüzden şifreyi kaydediniz.

İkinci adımda ise

Connect to NodeJS



Connect to your application



Drivers

Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)



Drivers seçeneğini tıklayarak devam edin.

Üçüncü adımda ise

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver	Version
Node.js ▼	5.5 or later ▼

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

[View MongoDB Node.js Driver installation instructions.](#)

3. Add your connection string into your application code

☐ View full code sample

```
mongodb+srv://furkan:<password>@nodejs.smtkeb3.mongodb.net/?  
retryWrites=true&w=majority&appName=NodeJS
```

Replace **<password>** with the password for the **furkan** user. Ensure any option params are [URL encoded](#).

RESOURCES

[Get started with the Node.js Driver](#)

[Node.js Starter Sample App](#)

[Access your Database Users](#)

[Troubleshoot Connections](#)

Connection stringimizi veriyor. Bu string sayesinde kodumuzu veri tabanına bağlayabiliyoruz.

Veri tabanını kurduk şimdi koda geçebiliriz. VS Code'u açıp Task manager adında bir klasör oluşturunuz ve npm init -y ile projeyi başlatınız. Ardından ise npm install mongodb ile modülü indirin.

```

TaskManager > JS mongodbjs > ...
  Click here to ask Blackbox to help you code faster
1  const { MongoClient, ServerApiVersion } = require("mongodb"); 724.5k (gzipped: 192.5k)
2
3  // Replace the placeholder with your Atlas connection string
4  const uri = "<connection string>";
5
6  // Create a MongoClient with a MongoClientOptions object to set the Stable API version
7  const client = new MongoClient(uri, {
8    serverApi: {
9      version: ServerApiVersion.v1,
10     strict: true,
11     deprecationErrors: true,
12   }
13 });
14
15
16 async function run() {
17   try {
18     // Connect the client to the server (optional starting in v4.7)
19     await client.connect();
20
21     // Send a ping to confirm a successful connection
22     await client.db("admin").command({ ping: 1 });
23     console.log("Pinged your deployment. You successfully connected to MongoDB!");
24   } finally {
25     // Ensures that the client will close when you finish/error
26     await client.close();
27   }
28 }
29 run().catch(console.dir);

```

Örnek kod blogunu yazınız. Bu kod blokları database'e bağlanıp bağlanmadığımızı test etmek içindir. Tabii ki uri değişkenine kendi connection stringimizi yapııştırıyoruz.

Bu çıktıyı alıyorsanız veri tabanına bağlanmışsınız demektir.

```

PS C:\Users\furka\OneDrive\Masaüstü\Bahar 23-24\Node.js\10.hafta\TaskManager> node .\mongodb.js
Pinged your deployment. You successfully connected to MongoDB!

```

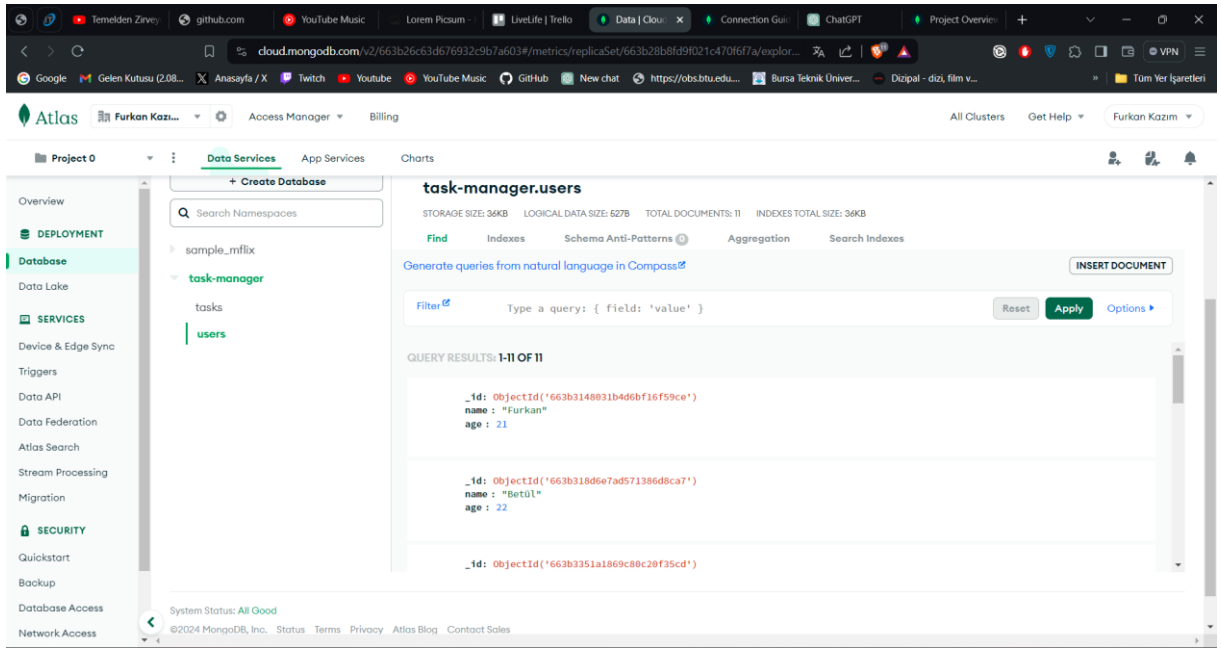
Tekli veri gönderimi için bu kod parçasını kullanmalısınız.

```

await db.collection("users").insertOne({
  _id: id,
  name: "Furkan",
  age: 21
}); // Tekli Gönderim

```

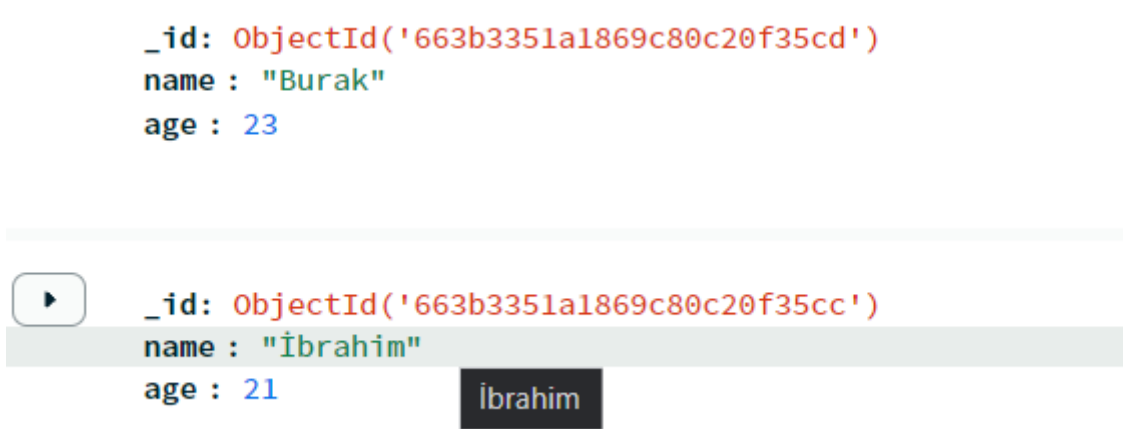
Kodu böyle çalıştırınız.



Bu şekilde kaydımızın veri tabanına gönderildiğini görebiliyoruz.

Çoklu veri gönderimi için ise

```
await db.collection("users").insertMany([
  { name: "İbrahim", age: 21 },
  { name: "Burak", age: 23 }
]); // Çoklu Gönderim
```



Veri tabanına aktarılmış.

Peki bu kullanıcı id'lerine ulaşabilir miyiz?

Bu sorunun cevabı tabii ki evet. Nasıl olduğunu hemen göstereyim.

```
const id = new ObjectId()
console.log(id)           // Kullanıcının Db'deki id'si
console.log(id.id)        // id'nin bölünmüş hali
console.log(id.id.length) // id'nin bölünmüş halinin uzunluğu
console.log(id.getTimestamp()) // kullanıcının eklendiği tarih
```

Bu kod satırları bir id tanımlar ver onların bazı özelliklerine erişmemize olanak tanır.

```
await db.collection('users').insertOne({
  _id: 1,
  name: 'Tansel',
  age: 26
  Complexity is 3 Everything is cool!
}, (error, result) => {
  if (error) {
    return console.log('Unable to insert user')
  }
  console.log(result.ops)
})
```

Burada ise id'yi kendimizin verebildiğini görebiliyoruz.

Veri tabanında da kontrol edelim.

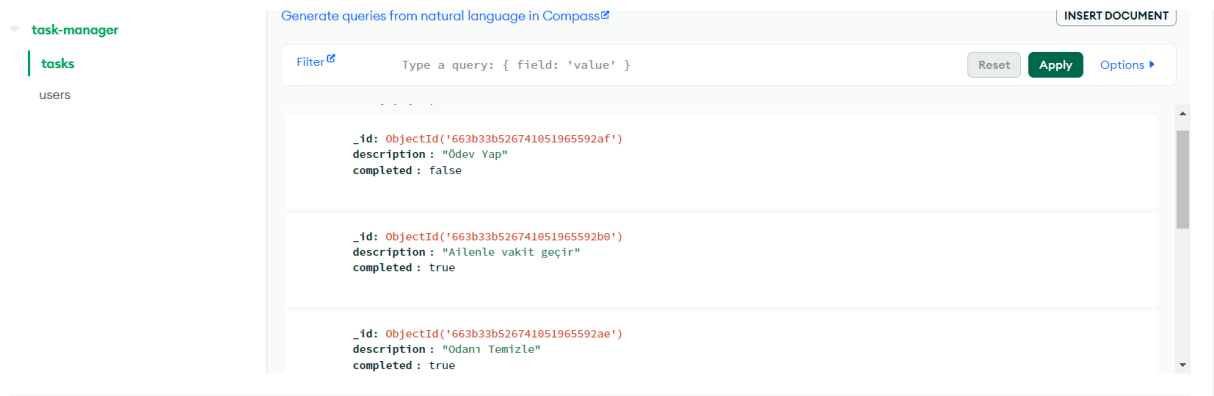
```
_id: 1
name : "Tansel"
age : 26
```

Proje ismine uygun olarak veri ekleyelim.

```
await db.collection("tasks").insertMany([
  { description: "Odanı Temizle", completed: true },
  { description: "Ödev Yap", completed: false },
  { description: "Ailenle vakit geçir", completed: true }
]); // Çoklu Gönderim

console.log("Data inserted successfully");
```

Bu sefer veri tabanını değiştiriyoruz. Artık verileri tasks adlı veritabanına gönderilmiş olması gerekiyor.



Görüldüğü gibi tasks ve users adlı 2 tane veri tabanımız var. Öncekiler users'a şimdikiler ise tasks veri tabanına gönderildi.

En son ise veri tabanında veri arama işlemi nasıl yapılır ona bakalım.

```
await db.collection("users").findOne({ //Birden fazla aynı isimde kullanıcı varsa ilk karşılaştığını getirir.
  name: "Furkan"
}, (error, user) => {
  if (error) {
    console.log("Unable to fetch")
  }
  console.log(user)
})
```

Bu kod bloğu users veri tabanında ismi Furkan olan ilk veriyi geri döndürür. Eğer yoksa Unable to fetch hata mesajını yazdırır. Ama eğer ki bulursa user'ı konsola yazdırır.