# MongoDB part - C

# Create a MongoDB Atlas cluster

Follow these steps to create and connect to a free cluster in MongoDB Atlas

- Create an Atlas account
- Deploy a Free Tier cluster
- Add your connection IP address to your IP access list
- Create a database user for your cluster
- Connect to your cluster

Be sure to save the connection string and database user's credentials for connecting to the database.

## 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

**Use this connection string in your application**

View full code sample  Show Password

```
mongodb+srv://kilarijyothsna_db_user:<db_password>@student.rr4mkyr.mongodb.net/?
retryWrites=true&w=majority&appName=Student
```

Replace **<db_password>** with the password for the **kilarijyothsna_db_user** database 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

Go Back

Done

**Step 1: Set up MongoDB Atlas**

1. Go to MongoDB Atlas and **create a free account**.
2. Create a **new cluster** (free tier is fine).
3. Go to **Database → Collections → Create Database**:
    1. Database name: `studentDB`
    2. Collection name: `students`
4. Go to **Database Access → Add New Database User**:
    1. Username: `myUser`
    2. Password: `myPassword`
    3. Assign **Read/Write to any database**
5. Go to **Network Access → Add IP Address**:
    1. Click **Allow Access from Anywhere (0.0.0.0/0)** for testing
    2. Or whitelist your current IP
6. Go to **Clusters → Connect → Connect Your Application**

mongodb+srv://kilarijyothsna_db_user:<db_password>@student.rr4mkyr.mongodb.net/?retryWrites=true&w=majority&appName=Student

https://www.mongodb.com/products/platform/atlas-database

Update

Import favorites | Booking.com | Enhanced Multimed... | Utomik Games | Booking.com | Express VPN | LastPass password... | r | Other favorites

NEW Modernize 2-3x faster with MongoDB's AI-powered Application Modernization Platform. Learn more >

MongoDB. Products Resources Solutions Company Pricing

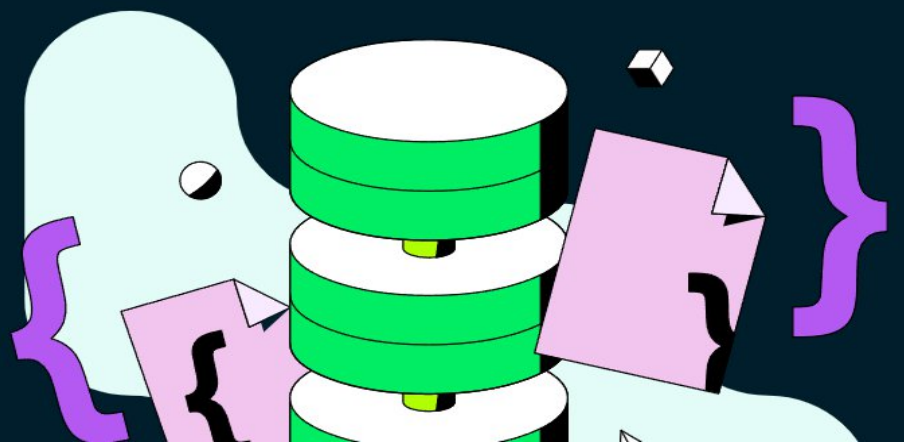
Get Started

Select this

Atlas Database Overview Features Security Getting Started Pricing Resources

# Atlas Database

The most effective way to deploy MongoDB. Enjoy unmatched resilience, scalability, and enterprise-grade security while eliminating operational



# Sign up

Sign up with Google

First Name*

Last Name*

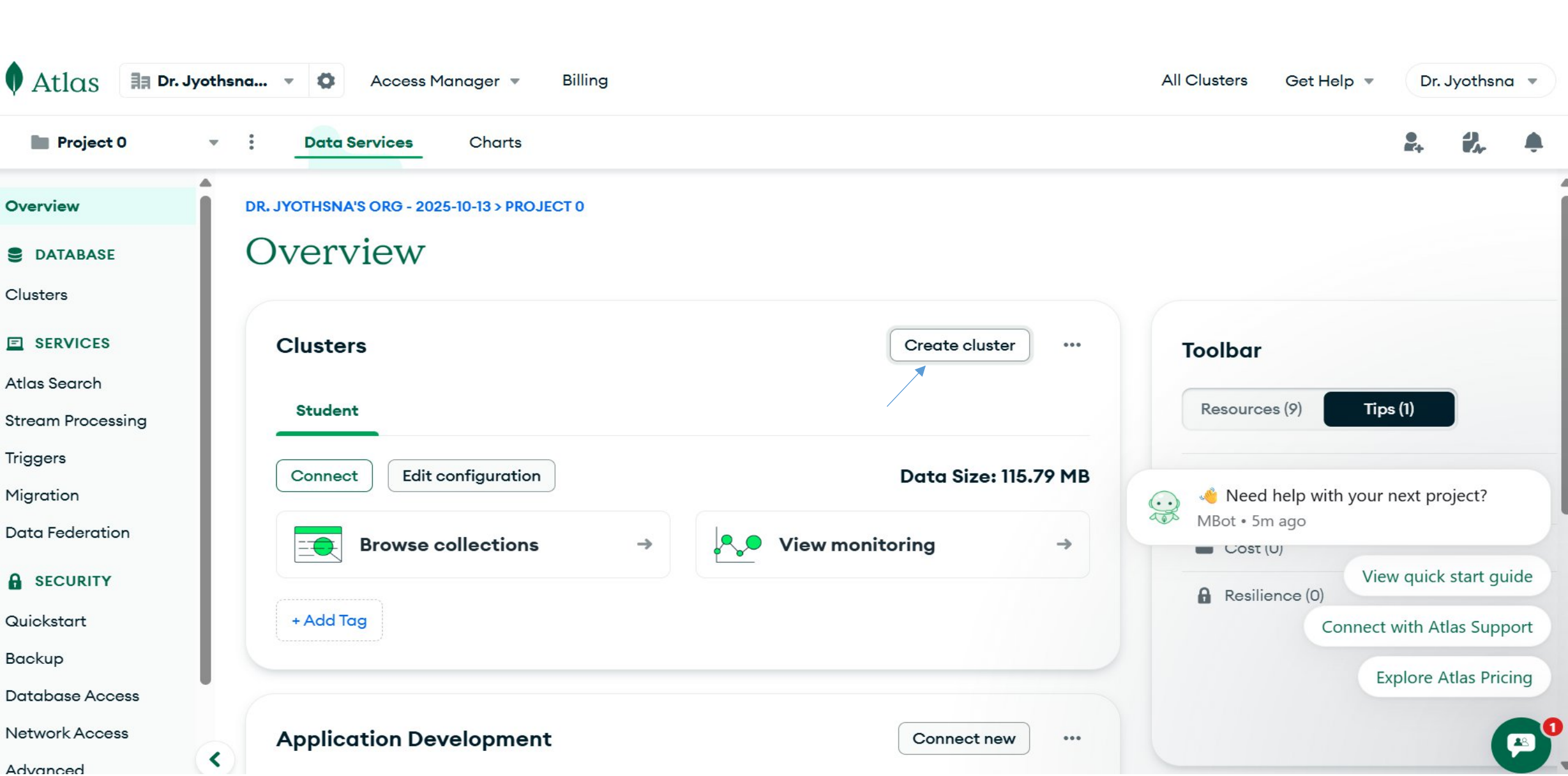
Company

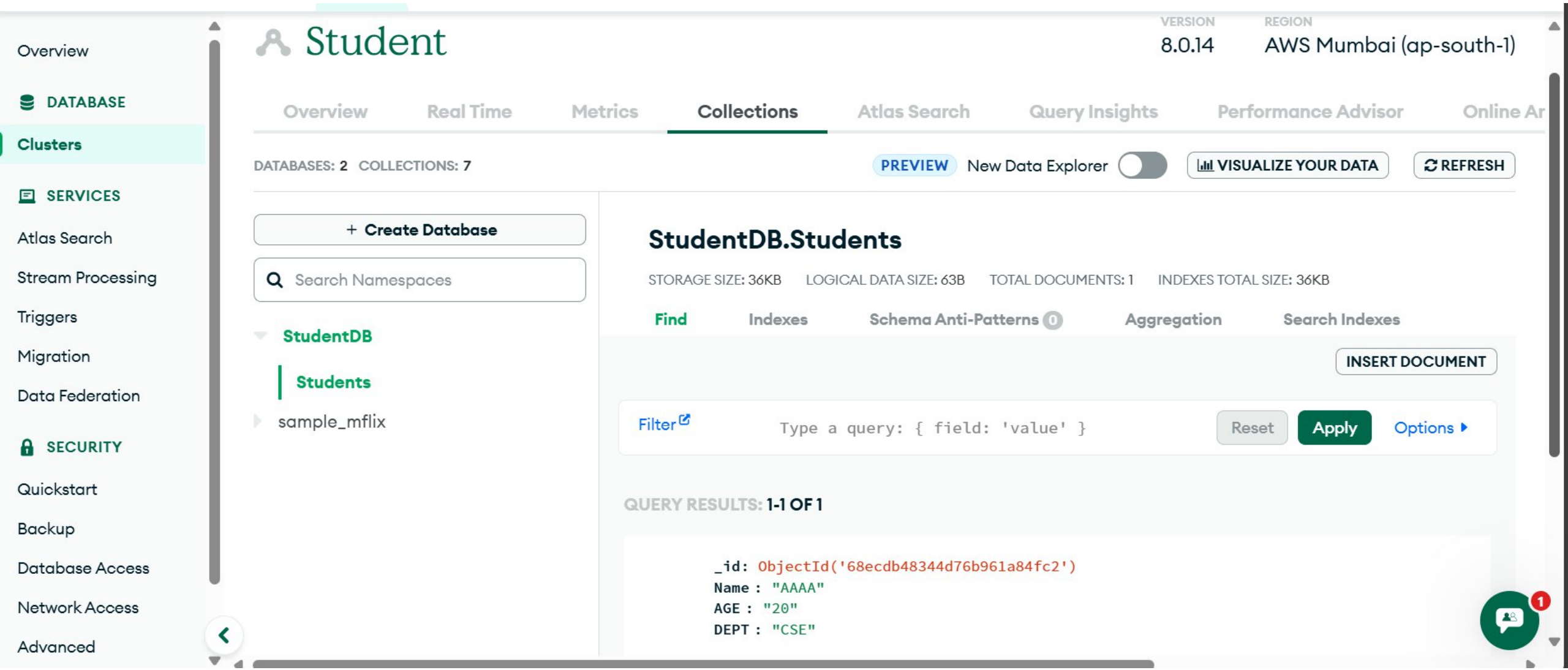
Email*

Password*

I agree to the Terms of Service and Privacy Policy.

Create your Atlas account


Atlas
Dr. Jyothsna...
Access Manager
Billing
All Clusters
Get Help
Dr. Jyothsna
Project 0
Data Services
Charts
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
DR. JYOTHSNA'S ORG - 2025-10-13 > PROJECT 0
Overview
Clusters
Create cluster
Student
Connect
Edit configuration
Data Size: 115.79 MB
Browse collections
View monitoring
+ Add Tag
Application Development
Connect new
Toolbar
Resources (9)
Tips (1)
Need help with your next project?
MBot • 5m ago
Cost (0)
Resilience (0)
View quick start guide
Connect with Atlas Support
Explore Atlas Pricing


Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
Student
VERSION
8.0.14
REGION
AWS Mumbai (ap-south-1)
Overview
Real Time
Metrics
Collections
Atlas Search
Query Insights
Performance Advisor
Online Ar
DATABASES: 2 COLLECTIONS: 7
PREVIEW
New Data Explorer
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
StudentDB
Students
sample_mflix
StudentDB.Students
STORAGE SIZE: 36KB
LOGICAL DATA SIZE: 63B
TOTAL DOCUMENTS: 1
INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
QUERY RESULTS: 1-1 OF 1
_id: ObjectId('68ecdb48344d76b961a84fc2')
Name : "AAAA"
AGE : "20"
DEPT : "CSE"
1

Overview

DATABASE

Clusters

SERVICES

Atlas Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Database Access

Network Access

DR. JYOTHSNA'S ORG - 2025-10-13 > PROJECT 0

# Database Access

Database Users | Custom Roles

+ ADD NEW DATABASE USER

| User | Description | Authentication Method | MongoDB Roles | Resources | Actions |
|---|---|---|---|---|---|
| kilarijyothsna_db_user | | SCRAM | atlasAdmin@admin | 1 Cluster, 0 Federated Database Instances, 0 Stream Processing Instances | EDIT DELETE |

# Edit User: kilarijyothsna_db_user@admin

Update a database user to grant an application or user access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding Access Manager

**Authentication Method**

| Password | Certificate | AWS IAM | Federated Auth (MongoDB 7.0 and up) |
|---|---|---|---|

MongoDB uses SCRAM as its default authentication method.

**Password Authentication**

kilarijyothsna_db_user

jyothsna_ HIDE

Autogenerate Secure Password | Copy

**User Description**

Add an optional description to your user.

Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access

We are deploying your changes (current action: configuring MongoDB)

DR. JYOTHSNA'S ORG - 2025-10-13 > PROJECT 0

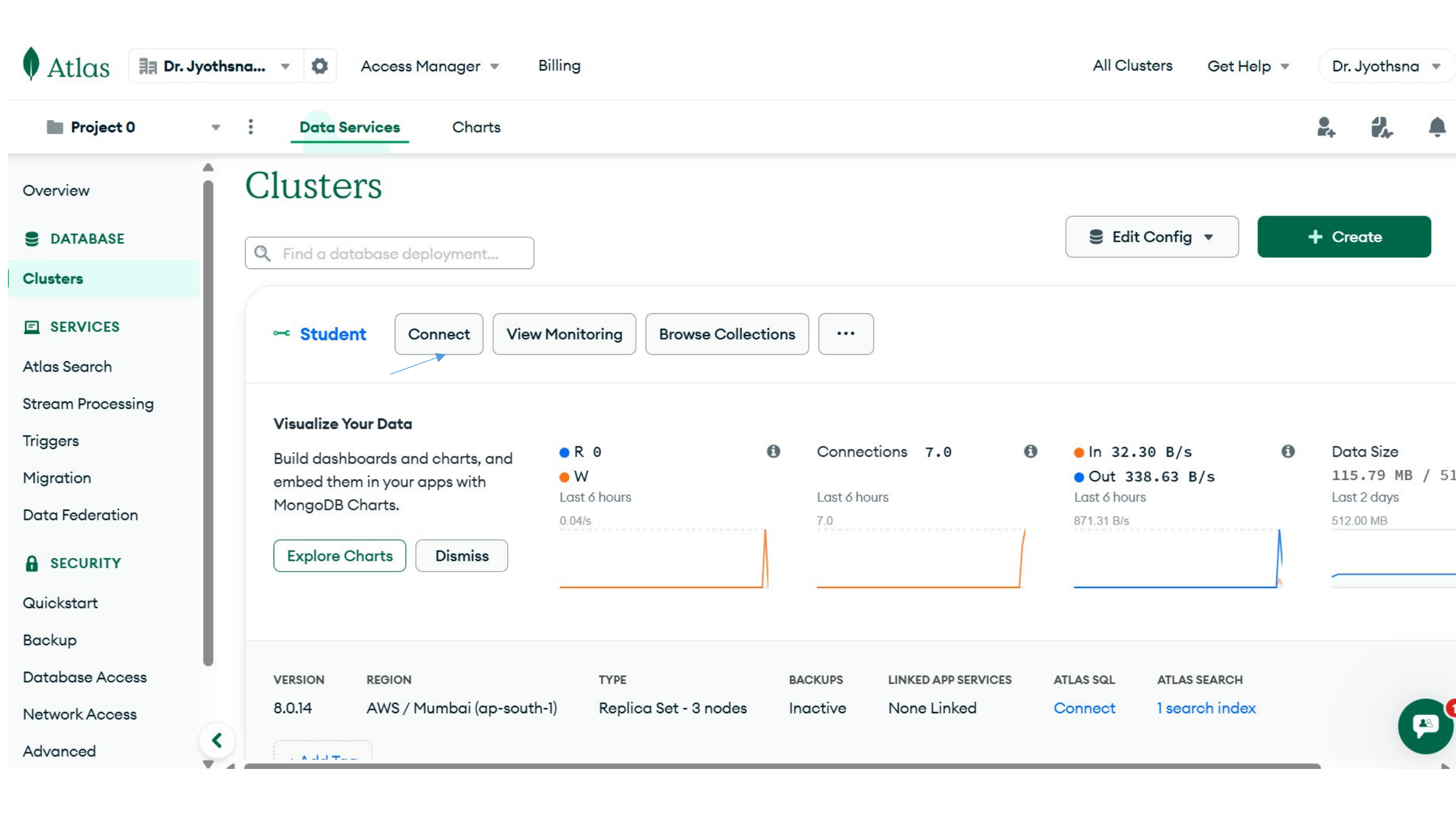
# Network Access

IP Access List | Peering | Private Endpoint

+ADD IP ADDRESS

You will only be able to connect to your cluster from the following list of IP Addresses:

| IP Address | Comment | Status | Actions |
|---|---|---|---|
| 0.0.0.0/0 (includes your current IP address) | | Pending | EDIT DELETE |
| 42.111.160.27/32 (includes your current IP address) | | Active | EDIT DELETE |
| 202.53.81.82/32 | Created as part of the Auto Setup process | Active | EDIT DELETE |


Atlas
Dr. Jyothsna...
Access Manager
Billing
All Clusters
Get Help
Dr. Jyothsna
Project 0
Data Services
Charts
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
Clusters
Find a database deployment...
Edit Config
Create
Student
Connect
View Monitoring
Browse Collections
Visualize Your Data
Build dashboards and charts, and embed them in your apps with MongoDB Charts.
Explore Charts
Dismiss
R 0
W
Last 6 hours
0.04/s
Connections 7.0
Last 6 hours
7.0
In 32.30 B/s
Out 338.63 B/s
Last 6 hours
871.31 B/s
Data Size
115.79 MB / 51
Last 2 days
512.00 MB
VERSION
REGION
TYPE
BACKUPS
LINKED APP SERVICES
ATLAS SQL
ATLAS SEARCH
8.0.14
AWS / Mumbai (ap-south-1)
Replica Set - 3 nodes
Inactive
None Linked
Connect
1 search index

# Connect to Student

1. Set up connection security
2. Choose a connection method
3. Connect

## Connect to your application

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

## Access your data through tools

**Compass**
Explore, modify, and visualize your data with MongoDB's GUI

**Shell**
Quickly add & update data using MongoDB's Javascript command-line interface

**MongoDB for VS Code**
Work with your data in MongoDB directly from your VS Code environment

**Atlas SQL**

## Connecting with MongoDB Driver

### 1. Select your driver and version

We recommend installing and using the latest driver version.

**Driver** Node.js

**Version** 6.7 or later

### 2. Install your driver

**Run the following on the command line**

```
npm install mongodb
```

View MongoDB Node.js Driver installation instructions.

npm install mongodb
npm install  dotenv

### 3. Add your connection string into your application code

**Use this connection string in your application**

View full code sample

```
mongodb+srv://kilarijyothsna_db_user:<db_password>@student.rr4mkyr.mongodb.net/?retryWrites=true&w=majority&appName=Student
```

mongodb+srv://kilarijyothsna_db_user:<db_password>@student.rr4mkyr.mongodb.net/?retryWrites=true&w=majority&appName=Student

Replace **<db_password>** with the password for the **kilarijyothsna_db_user** database user. Ensure any option params are URL encoded .

jyothsna_

**RESOURCES**

- Get started with the Node.js Driver
- Node.js Starter Sample App
- Access your Database Users
- Troubleshoot Connections

Go Back

Done

**Step 2: Update Backend (`server.js`)**
Replace local MongoDB URI with **Atlas URI**:

**Step 1 — Open Command Prompt properly**
Run as **Administrator** (important if previous errors showed “System error 5”).
Then navigate to your backend folder:

```
cd "C:\My project\pvp\Subjects\2025-26\Sem-1\MERN\React  Programs\Student-atlas\student-backend"
```

**Step 2 — Create `.env` file**
In this folder, create a file named `.env` with this content:

```
PORT=5000
MONGO_URI=mongodb+srv://kilarijyothsna_db_user:<YOUR_ENCODED_PASSWORD>@student.rr4mkyr.mongodb.net/StudentDB?retryWrites=true&w=majority
```

Replace `<YOUR_ENCODED_PASSWORD>` with your **MongoDB Atlas password**, URL-encoded if it includes special characters.
For example,
if your password is `My@Pass:123`, use `My%40Pass%3A123`.

**Step 3 — Create or update `server.js`**

Inside `student-backend`, create a file named **server.js** :

```
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();
app.use(cors());
app.use(express.json());

// ✅ Connect to MongoDB Atlas
mongoose.connect(process.env.MONGO_URI)
  .then(() => console.log('✅ Connected to MongoDB Atlas'))
  .catch(err => console.error('❌ MongoDB connection failed:',
err.message));
```

```
// Define schema & model
const studentSchema = new mongoose.Schema({}, { strict:
false });
const Student = mongoose.model('Student', studentSchema,
‘Students');

// Routes
app.get('/students', async (req, res) => {
  try {
    const students = await Student.find();
    res.json(students);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```

```
app.post('/students', async (req, res) => {
  try {
    const newStudent = new Student(req.body);
    const saved = await newStudent.save();
    res.status(201).json(saved);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`🚀 Server running on
http://localhost:${PORT}`));
```

Step 4 — Install dependencies

```
npm install express mongoose cors dotenv
```

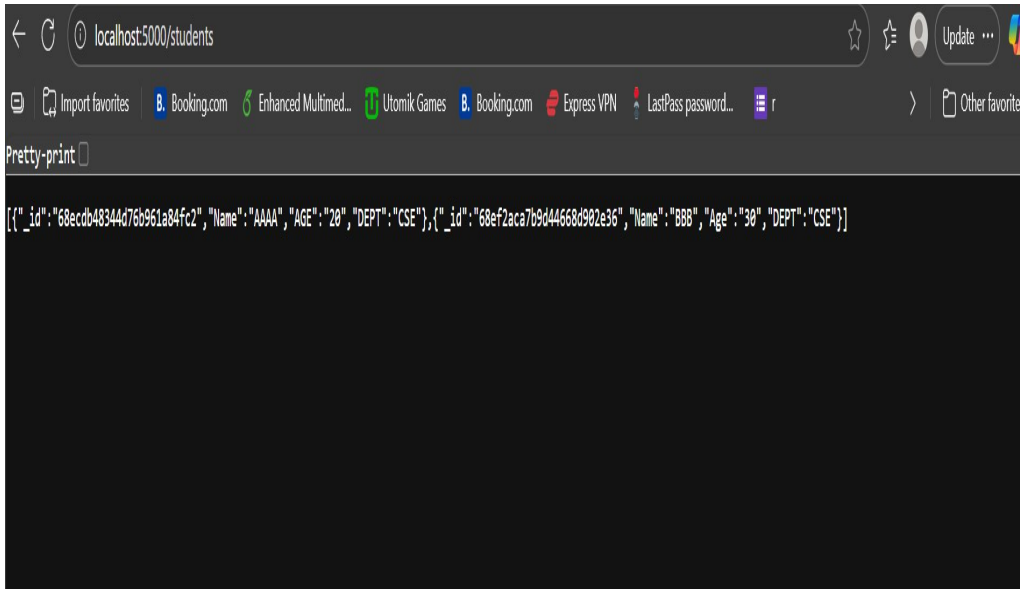
Step 5 — Start your backend server

```
node server.js
```

✔ Connected to MongoDB Atlas
🚀 Server running on http://localhost:5000

**Step 6 — Test it**
Open a browser or Postman:

```
http://localhost:5000/students
```



# React frontend Development

Open a **new terminal** and navigate to:

```
C:\My project\pvp\Subjects\2025-26\Sem-1\MERN\React  Programs\Student-atlas>npx create-react-app student-frontend

Creating a new React app in C:\My project\pvp\Subjects\2025-26\Sem-1\MERN\React  Programs\Student-atlas\student-frontend
.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

\
```

App.js

```
import React, { useEffect, useState } from 'react';

function App() {
  const [students, setStudents] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetch('http://localhost:5000/students')
      .then(res => res.json())
      .then(data => {
        setStudents(data);
        setLoading(false);
      })
      .catch(err => {
        console.error('✕ Fetch error:', err);
        setLoading(false);
      });
  }, []);
```

```
if (loading) return <p>Loading students...</p>;
if (!students || students.length === 0) return <p>No students found.</p>;

const keys = Object.keys(students[0]).filter(k => k !== '__v' && k !== '_id');

return (
  <div style={{ padding: '20px' }}>
    <h1>🎓 Student Information</h1>
    <table border="1" cellPadding="10">
      <thead>
        <tr>
          {keys.map(key => (
            <th key={key}>{key}</th>
          ))}
        </tr>
      </thead>
      <tbody>
        {students.map(student => (
          <tr key={student._id}>
            {keys.map(key => (
              <td key={key}>{student[key]}</td>
            ))}
          </tr>
        ))}
      </tbody>
    </table>
  </div>
  );
}

export default App;
```

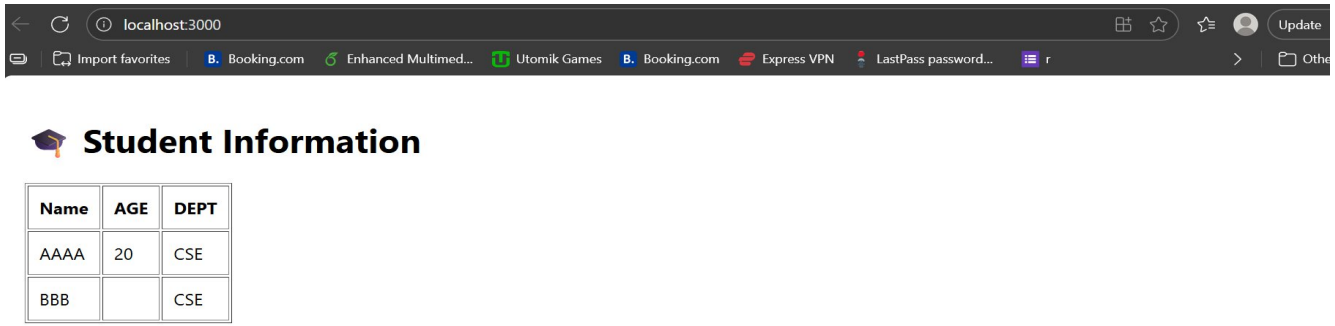
Step 4 — Run React

npm start

Now open your browser at:

http://localhost:3000/

You should see a **table listing all students** fetched from MongoDB Atlas



mongoDB. Atlas

## Get started free

No credit card required

Sign up with Google

or

Your Company (optional)

Your Work Email

arpanneupane192@gmail.com

First Name

Arpan

Last Name

Neupane

Password

8 characters minimum

I agree to the terms of service and privacy policy.

Get started free

Already have an account? Sign in.

MongoDB.

## Great, now verify your email



Check your inbox at **<arpanneupane192@gmail.com>** and click the verification link inside to complete your registration. This link will expire shortly, so verify soon!

**Don't see an email?** Check your spam folder.

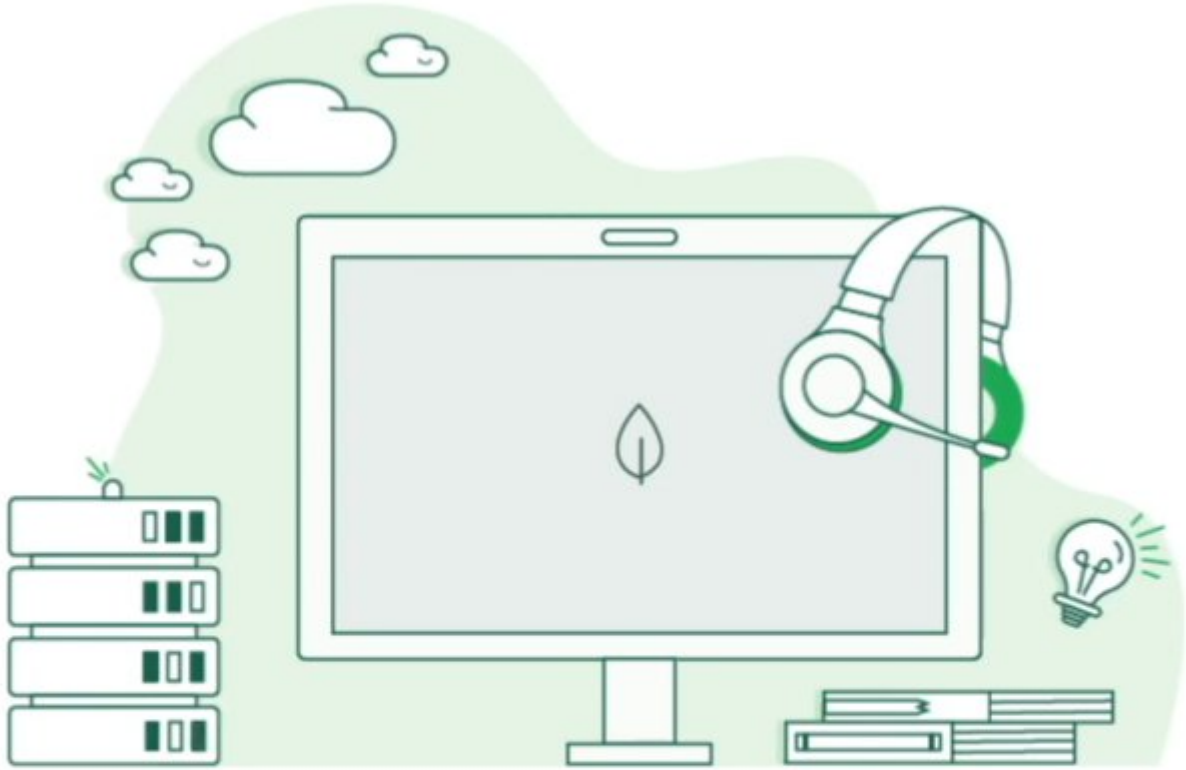
**Link expired?** Resend verification email

MongoDB.

## Email successfully verified!



Continue



## Welcome!

Use your account to deploy a **cloud database** with **MongoDB Atlas** and contact **Support.**

# Welcome to Atlas!

Tell us a few things about yourself and your project.



## What is your goal today?

Your answer will help us guide you to successfully getting started with MongoDB Atlas.

- ○ Migrate an existing application
- ○ Learn MongoDB
- ○ Explore what I can build
- ◉ Build a new application

## What type of application are you building?

Other

**Tell us what you're building**

Web Application

## What is your preferred language?

We'll use this to customize code samples and content we share with you. You can always change this later.

Select...

MongoDB.

MONGODB ATLAS

# Deploy a cloud database

Experience the best of MongoDB on AWS, Azure, and Google Cloud. Choose a deployment option to get started.

PREVIEW

## Serverless

For serverless applications that aren't critical with variable traffic. Minimal configuration required.

- Pay only for the operations you run
- Resources scale seamlessly to meet your workload
- Always-on security and backups

Create

Starting at

$0.30/1M reads

ADVANCED

## Dedicated

For production applications with sophisticated workload requirements. Advanced configuration controls.

- Network isolation and fine-grained access controls
- On-demand performance advice
- Multi-region and multi-cloud options available

Create

Starting at

$0.08/hr*

*estimated cost $56.94/month

FREE

## Shared

For learning and exploring MongoDB in a cloud environment. Basic configuration options.

- No credit card required to start
- Explore with sample datasets
- Upgrade to dedicated clusters for full functionality

Create

Starting at

FREE

I'll do this later

Advanced Configuration Options

**For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.**

No credit card required to start. Upgrade to dedicated clusters for full functionality.
Explore with sample datasets. Limit of one free cluster per project.

## Cloud Provider & Region

AWS, N. Virginia (us-east-1)

aws | Google Cloud | Azure

★ Recommended region  Dedicated tier region

**NORTH AMERICA**
- Oregon (us-west-2) ★
- N. Virginia (us-east-1) ★
- Ohio (us-east-2) ★
- N. California (us-west-1)
- Montreal (ca-central-1)

**SOUTH AMERICA**
- Sao Paulo (sa-east-1)

**EUROPE**
- Paris (eu-west-3) ★
- Stockholm (eu-north-1) ★
- Frankfurt (eu-central-1) ★
- Ireland (eu-west-1) ★
- London (eu-west-2) ★
- Milan (eu-south-1) ★

**MIDDLE EAST**
- Bahrain (me-south-1) ★

**AUSTRALIA**
- Sydney (ap-southeast-2) ★

**ASIA**
- Tokyo (ap-northeast-1)
- Seoul (ap-northeast-2)
- Singapore (ap-southeast-1) ★
- Hong Kong (ap-east-1) ★
- Mumbai (ap-south-1)
- Jakarta (ap-southeast-3) ★

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

Back | Create Cluster

## Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage)
Encrypted

Base hourly rate is for a MongoDB replica set with **3 data bearing servers**.

**Shared** Clusters for development environments and low-traffic applications

| Tier | RAM | Storage | vCPU | Base Price |
|---|---|---|---|---|
| M0 Sandbox | Shared | 512 MB | Shared | Free forever |
| M2 | Shared | 2 GB | Shared | $9 / MONTH |
| M5 | Shared | 5 GB | Shared | $25 / MONTH |

M0 clusters are best for getting started, and are not suitable for production environments.

500 max connections | Low network performance | 100 max databases | 500 max collections

## Additional Settings

MongoDB 5.0, No Backup

## Cluster Name

Cluster0

# Security Quickstart

To access data stored in Atlas, you'll need to create users and set up network security controls. Learn more about security setup

## 1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

**Username and Password** | **Certificate**

Create a database user using a username and password. Users will be given the *read and write to any database* privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

**Username**

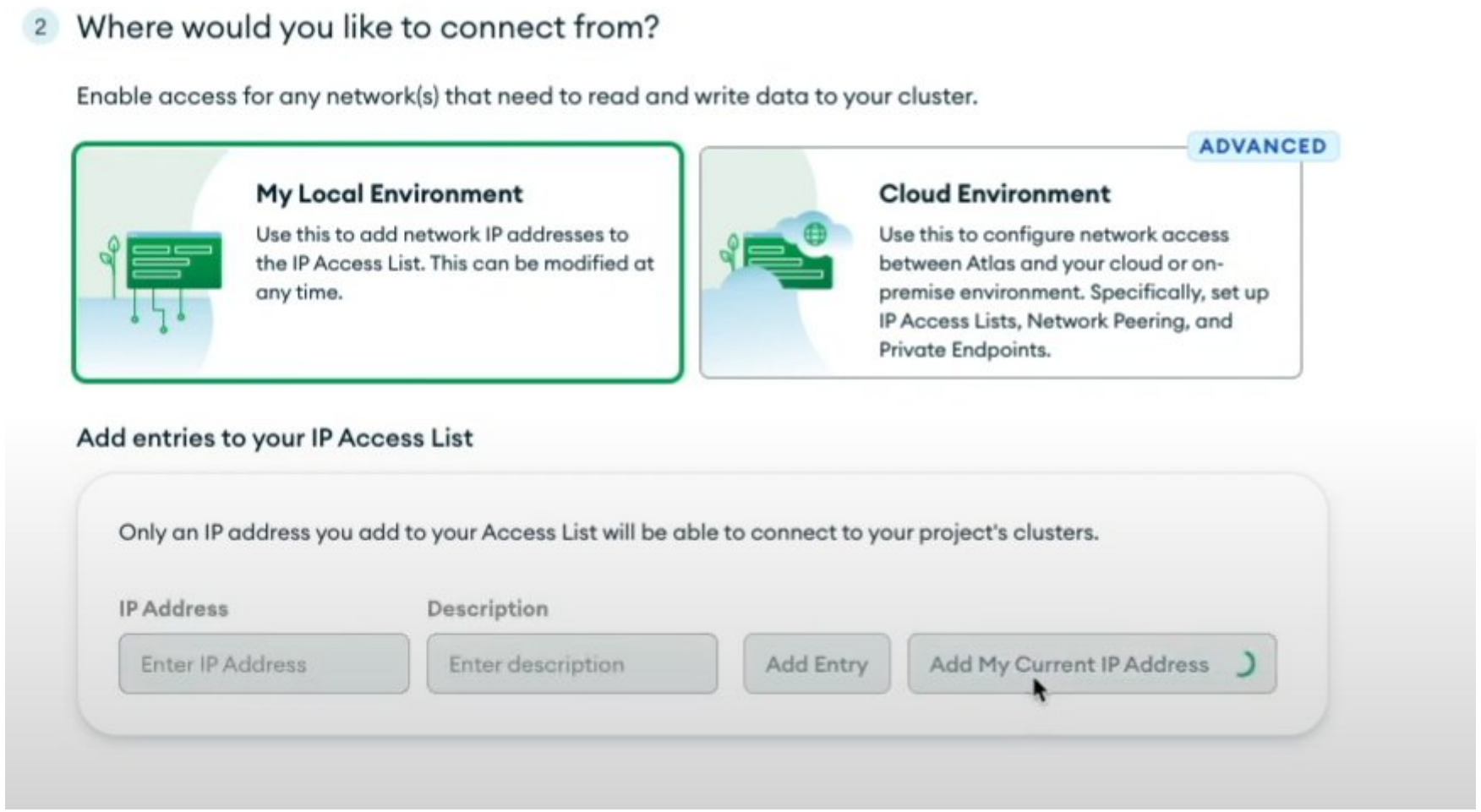
Enter username

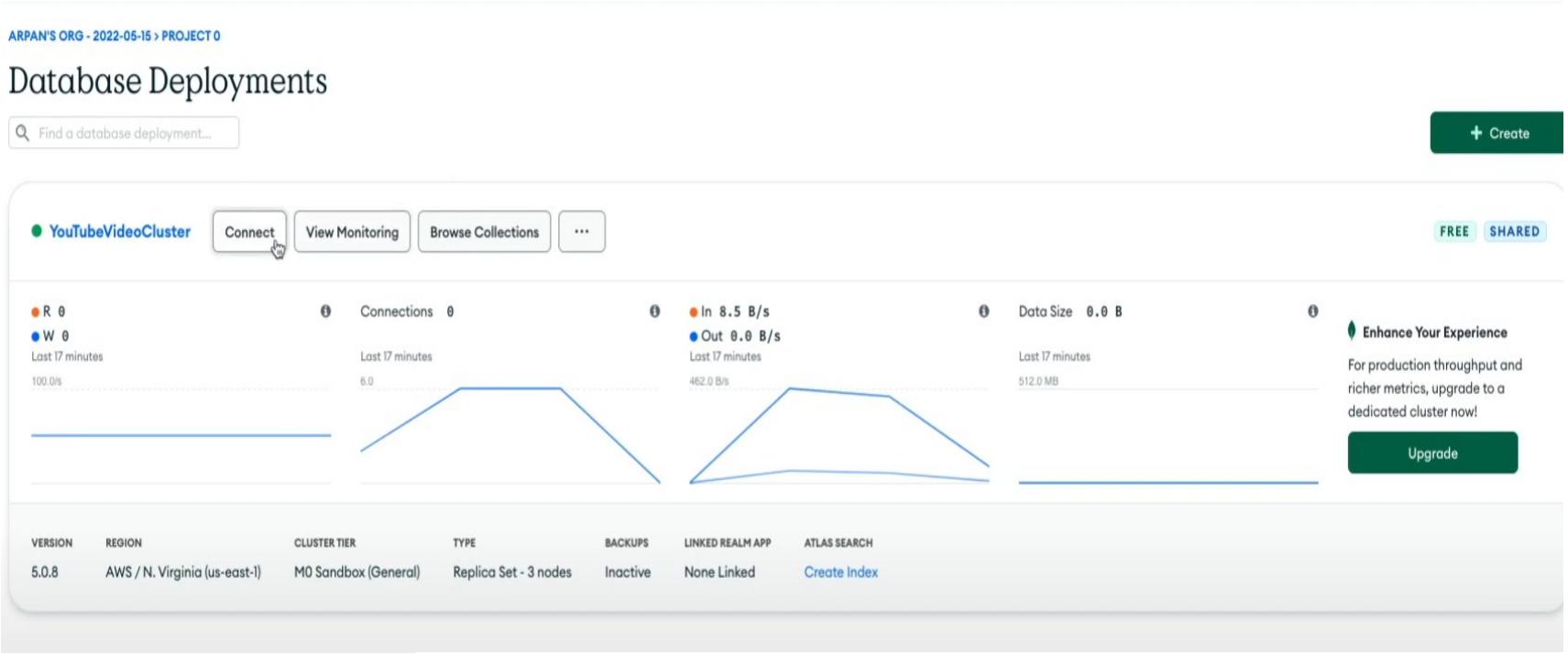
Any name

**Password**

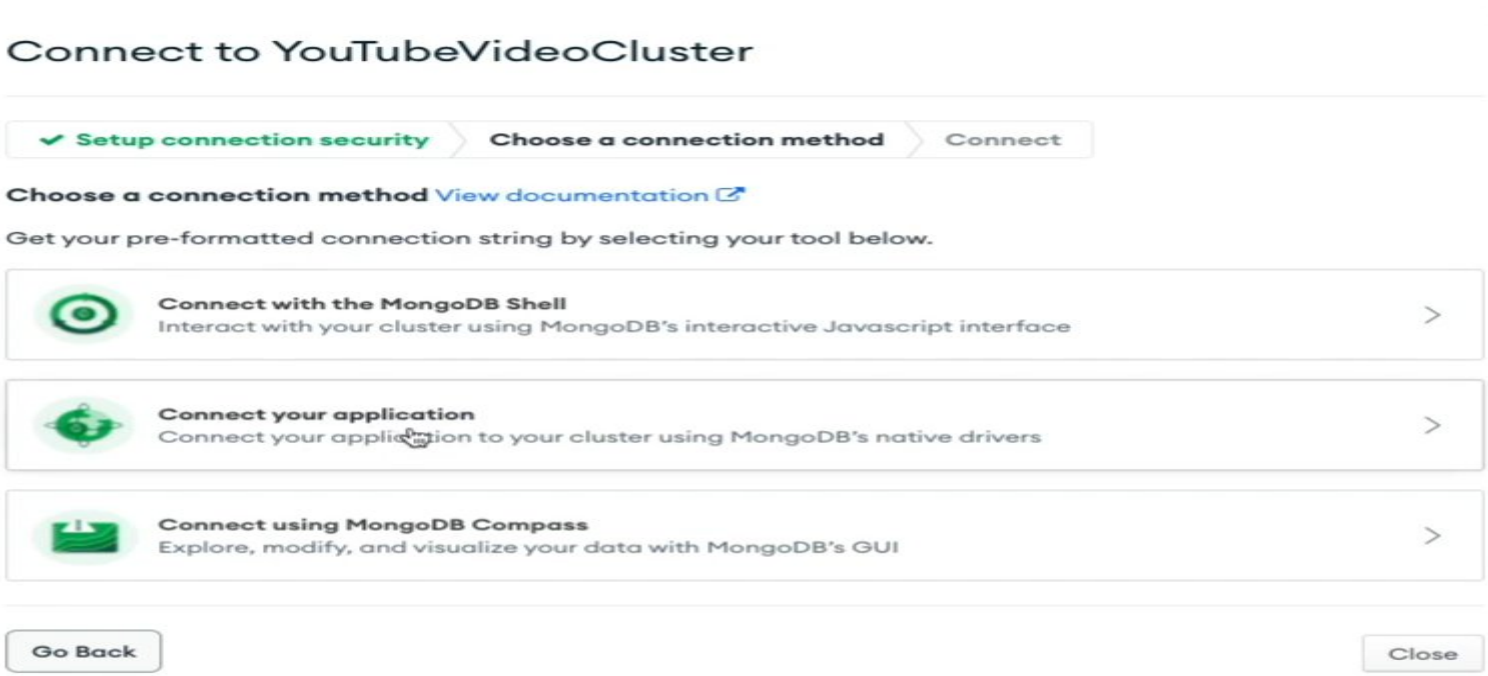
Enter password | Autogenerate Secure Password | Copy

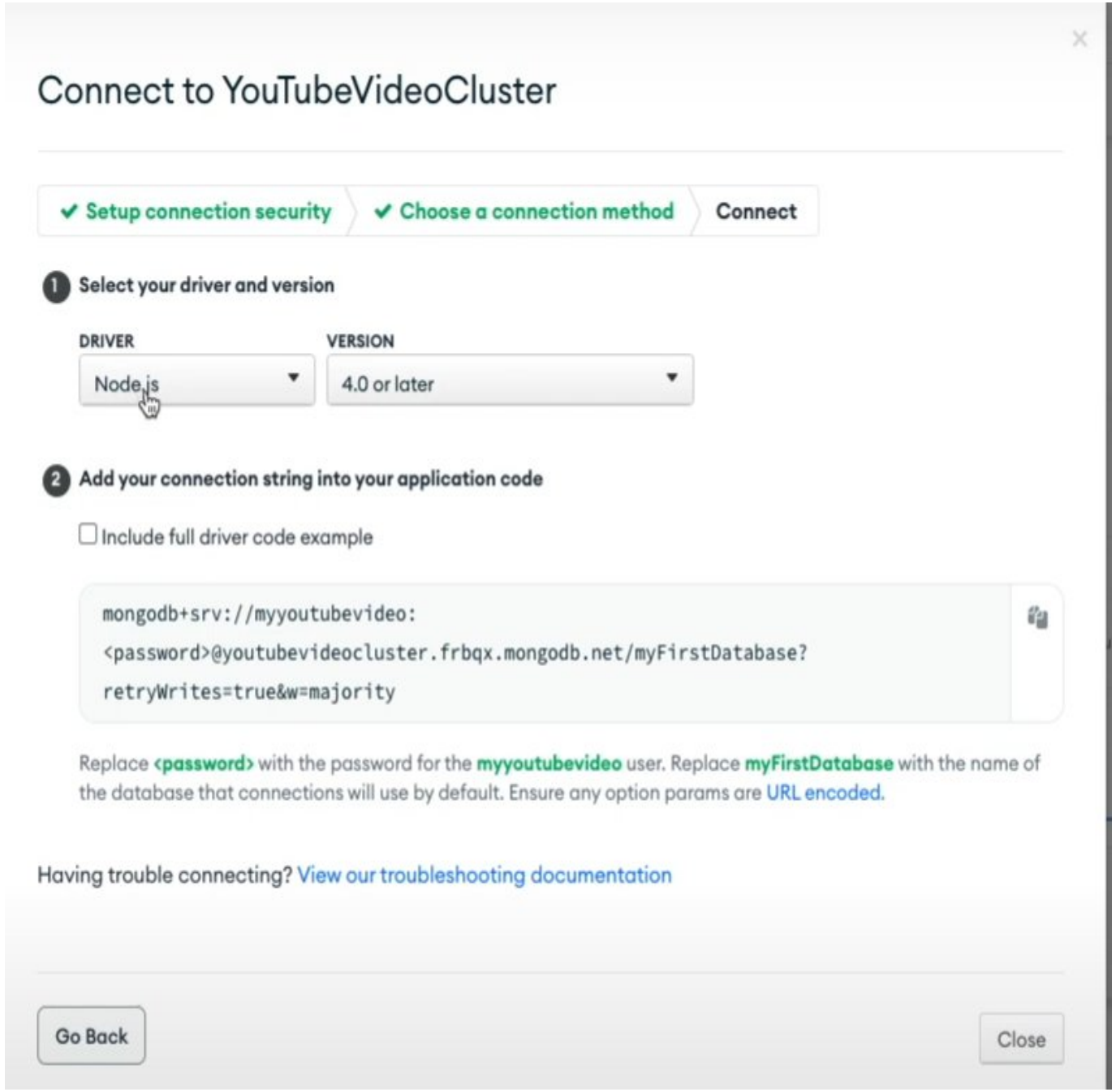
Create User

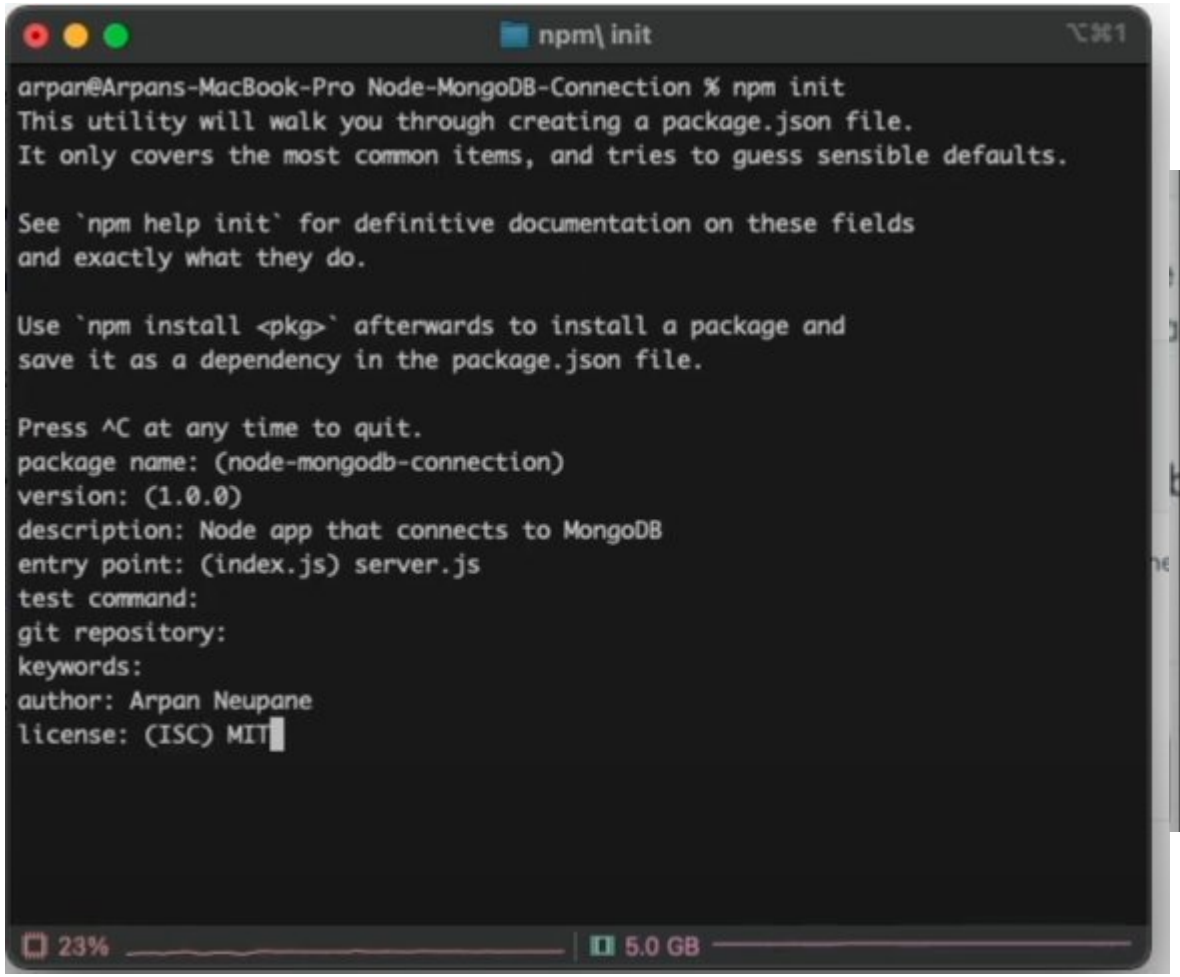

2 Where would you like to connect from?
Enable access for any network(s) that need to read and write data to your cluster.
My Local Environment
Use this to add network IP addresses to the IP Access List. This can be modified at any time.
ADVANCED
Cloud Environment
Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.
Add entries to your IP Access List
Only an IP address you add to your Access List will be able to connect to your project's clusters.
IP Address
Description
Enter IP Address
Enter description
Add Entry
Add My Current IP Address




3 different ways to connect MongoDb




Connect to YouTubeVideoCluster
✔ Setup connection security
✔ Choose a connection method
Connect
1 Select your driver and version
DRIVER
Node.js
VERSION
4.0 or later
2 Add your connection string into your application code
Include full driver code example
mongodb+srv://myyoutubevideo:
<password>@youtubevideocluster.frbqx.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority
Replace <password> with the password for the myyoutubevideo user. Replace myFirstDatabase with the name of the database that connections will use by default. Ensure any option params are URL encoded.
Having trouble connecting? View our troubleshooting documentation
Go Back
Close




```
Is this OK? (yes)
arpan@Arpans-MacBook-Pro Node-MongoDB-Connection % npm install express mongoose
--save

added 85 packages, and audited 86 packages in 4s

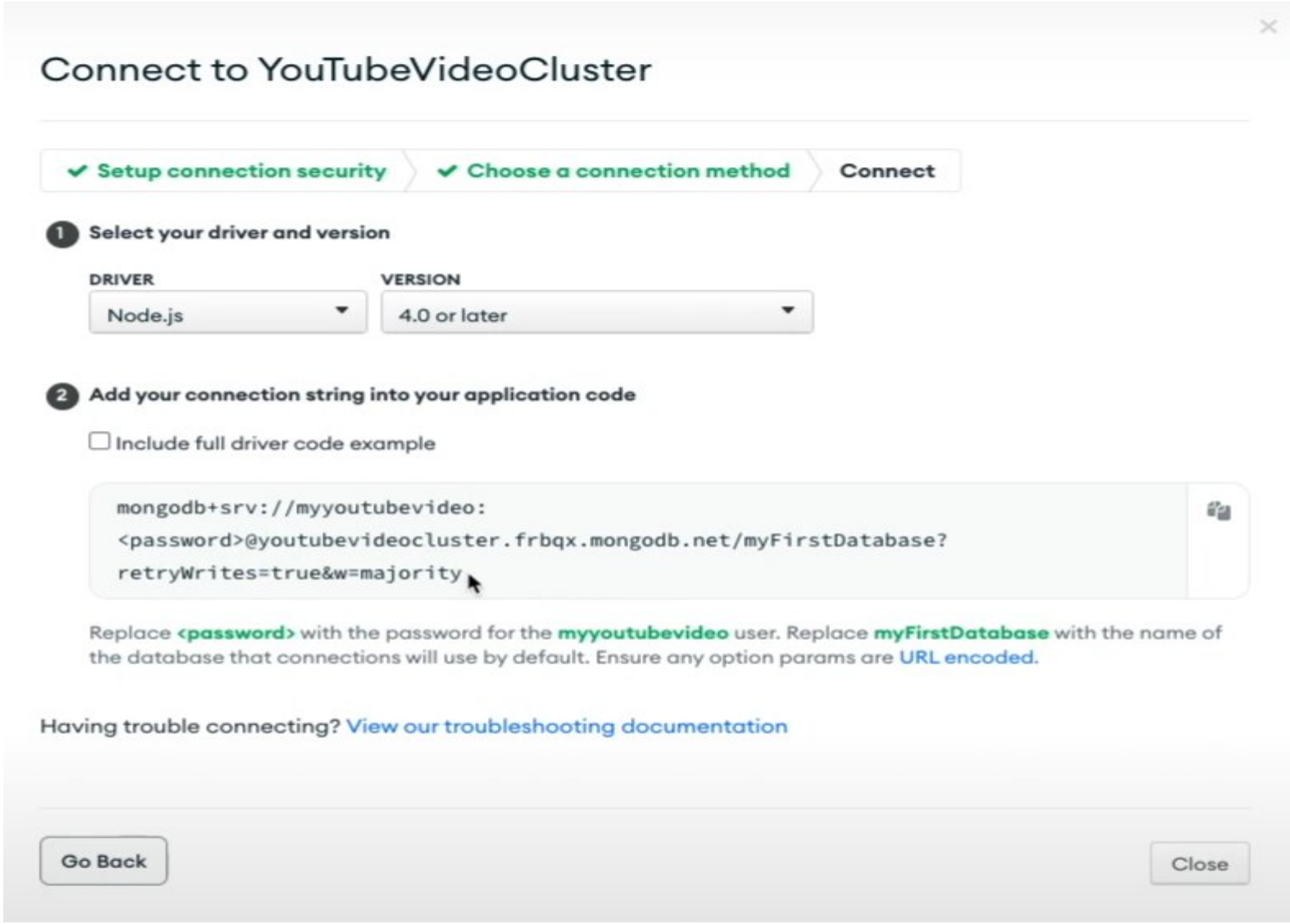
11 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
arpan@Arpans-MacBook-Pro Node-MongoDB-Connection %
```

```
r.js > app.listen() callback
const express = require("express");
const mongoose = require("mongoose");
const app = express();

app.listen(8000, () => {
  console.log("Server started on port 8000");
});
```

In password: replace password



```
const express = require("express");
const mongoose = require("mongoose");
const app = express();

const uri =
  "mongodb+srv://myyoutubevideo:myyoutubevideopassword@youtubevideocluster.frbqx.mongodb.net/myFirstDatabase
async function connect() {
  try {
    await mongoose.connect(uri);
    console.log("Connected to MongoDB");
  } catch (error) {
    console.error(error);
  }
}

connect();

app.listen(8000, () => {
  console.log("Server started on port 8000");
});
```

ARPAN'S ORG - 2022-05-15 > PROJECT 0

# Database Deployments

Find a database deployment...

+ Create

YouTubeVideoCluster | Connect | View Monitoring | Browse Collections | ···

FREE SHARED

R 0
W 0
Last 6 hours
100.0/s

Connections 0
Last 6 hours
100.0

In 6.4 B/s
Out 7.7 B/s
Last 6 hours
7.7 B/s

Data Size 0.0 B
Last 7 hours
512.0 MB

**Enhance Your Experience**

For production throughput and richer metrics, upgrade to a dedicated cluster now!

Upgrade

| VERSION | REGION | CLUSTER TIER | TYPE | BACKUPS | LINKED REALM APP | ATLAS SEARCH |
|---|---|---|---|---|---|---|
| 5.0.8 | AWS / N. Virginia (us-east-1) | M0 Sandbox (General) | Replica Set - 3 nodes | Inactive | None Linked | Create Index |

Connect to Your Cluster

Now that we have a cluster, we can connect to it. It's considered a good practice to keep your connection string and credentials separate from your code.

You can do that by creating a configuration file that's not checked into revision history.

Create a ".env" file at the root of your project and add your cluster connection string as a variable:

```
DB_URI=mongodb+srv://<username>:<password>@<clustername>.mongodb.net/
```

To load that configuration into our application, we'll use the dotenv package. Install it by executing the following npm command in your terminal:

```
npm install dotenv
```

we need to invoke the config() function to load the variables from the “.env” file into process.env.
Once we’ve done that, we can log the connection URI to make sure the configuration is loaded correctly.

```
import { config } from 'dotenv';

config();
console.log(process.env.DB_URI);
```

```
import { MongoClient } from 'mongodb';

export async function connectToCluster(uri) {
    let mongoClient;

    try {
        mongoClient = new MongoClient(uri);
        console.log('Connecting to MongoDB Atlas cluster...');
        await mongoClient.connect();
        console.log('Successfully connected to MongoDB Atlas!');

        return mongoClient;
    } catch (error) {
        console.error('Connection to MongoDB Atlas failed!', error);
        process.exit();
    }
}
```

```
export async function executeStudentCrudOperations() {
    const uri = process.env.DB_URI;
    let mongoClient;

    try {
        mongoClient = await connectToCluster(uri);
    } finally {
        await mongoClient.close();
    }
}
```

```
Connecting to MongoDB Atlas cluster...
Successfully connected to MongoDB Atlas!
```

Index,js

```
import { config } from 'dotenv';
import { executeStudentCrudOperations } from './studentsCrud.js';

config();
await executeStudentCrudOperations();
```

# CRUD Operations

```
export async function executeStudentCrudOperations() {
    const uri = process.env.DB_URI;
    let mongoClient;

    try {
        mongoClient = await connectToCluster(uri);
        const db = mongoClient.db('school');
        const collection = db.collection('students');
    } finally {
        await mongoClient.close();
    }
}
```

If the database doesn’t exist yet, it will be created.

```
export async function createStudentDocument(collection) {
    const studentDocument = {
        name: 'John Smith',
        birthdate: new Date(2000, 11, 20),
        address: { street: 'Pike Lane', city: 'Los Angeles', state: 'CA' },
    };

    await collection.insertOne(studentDocument);
}
```

```
export async function executeStudentCrudOperations() {
    const uri = process.env.DB_URI;
    let mongoClient;

    try {
        mongoClient = await connectToCluster(uri);
        const db = mongoClient.db('school');
        const collection = db.collection('students');

        console.log('CREATE Student');
        await createStudentDocument(collection);
    } finally {
        await mongoClient.close();
    }
}
```

You should see the message “CREATE Student” logged on your terminal. To verify that the document was created, navigate to your MongoDB Atlas instance, and select *Browse Collections*. Once selected, you should see your newly created database, collection, and document.

```
export async function updateStudentsByName(collection, name, updatedFields) {
    await collection.updateMany(
        { name },
        { $set: updatedFields }
    );
}
```

```
export async function executeStudentCrudOperations() {
    const uri = process.env.DB_URI;
    let mongoClient;

    try {
        mongoClient = await connectToCluster(uri);
        const db = mongoClient.db('school');
        const collection = db.collection('students');

        // console.log('CREATE Student');
        // await createStudentDocument(collection);
        console.log(await findStudentsByName(collection, 'John Smith'));

        console.log('UPDATE Student\'s Birthdate');
        await updateStudentsByName(collection, 'John Smith', { birthdate: new Date(2001, 5, 5
        console.log(await findStudentsByName(collection, 'John Smith'));
    } finally {
        await mongoClient.close();
    }
}
```

```
// APIs
app.get('/', (req, res) => {
    res.send('Hi');
})

// GET ALL
app.get('/students', async (req, res) => {
    const result = await connect();
    res.send(result);
})
```

```
app.get('/students/:id', async (req, res) => {
    let id = Number(req.params.id);
    const result = await connect({"id":id});
    res.send(result);
})
```

```
async app.get('/students/:id', async (req, res) => {
    t       let id = Number(req.params.id);
            const result = await connect({"id":id});
            res.send(result);
        })
            const result =  await coll.find(option).toArray();
            return result;
    } catch (err) { console.log(err) };
    return [];
}
```