

# STARTER-PACK HANDS-ON DATABASE MENGGUNAKAN MONGODB

Langkah-langkah dibawah ini perlu dilakukan oleh peserta sebelum webinar dimulai.

## Prasyarat

Sebelum memulai hands-on, pastikan peserta telah memenuhi prasyarat dengan install aplikasi dibawah ini:

1. **Database:** MongoDB. Bisa di download [disini](#)
2. **Database administrator:** MongoDB Compass. Bisa di download [disini](#)
3. **Akun atlasian:**

Setelah memastikan semua prasyarat di atas, peserta dapat melanjutkan ke bagian instalasi project starter-pack yang sudah dikirimkan:

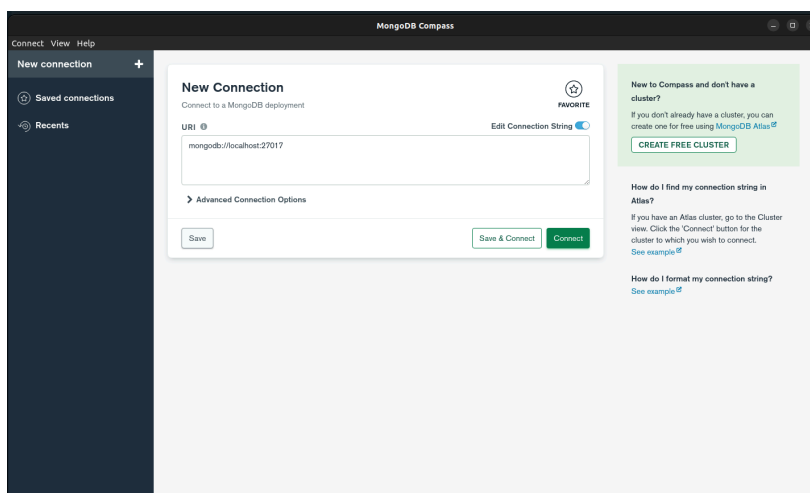
1. Buka terminal dan ketikan mongo agar dapat melakukan syntax mongo

```
# mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d979c2d7-8618-43ff-8111-297345b7364c") }
MongoDB server version: 4.4.6
...
The server generated these startup warnings when booting:
  2024-07-22T15:34:56.811+08:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/
  2024-07-22T15:34:57.749+08:00: 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).

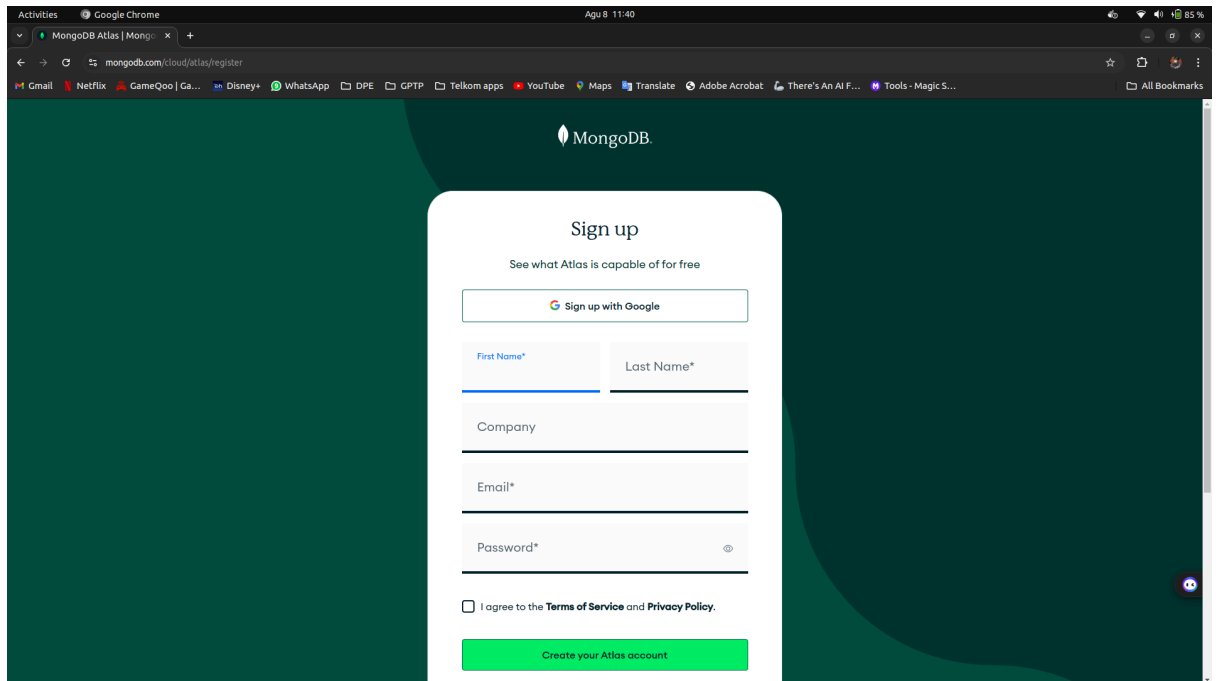
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
...
>
```

2. Buka mongoDB Compass untuk bisa melakukan Syntax mongo dengan menggunakan GUI.

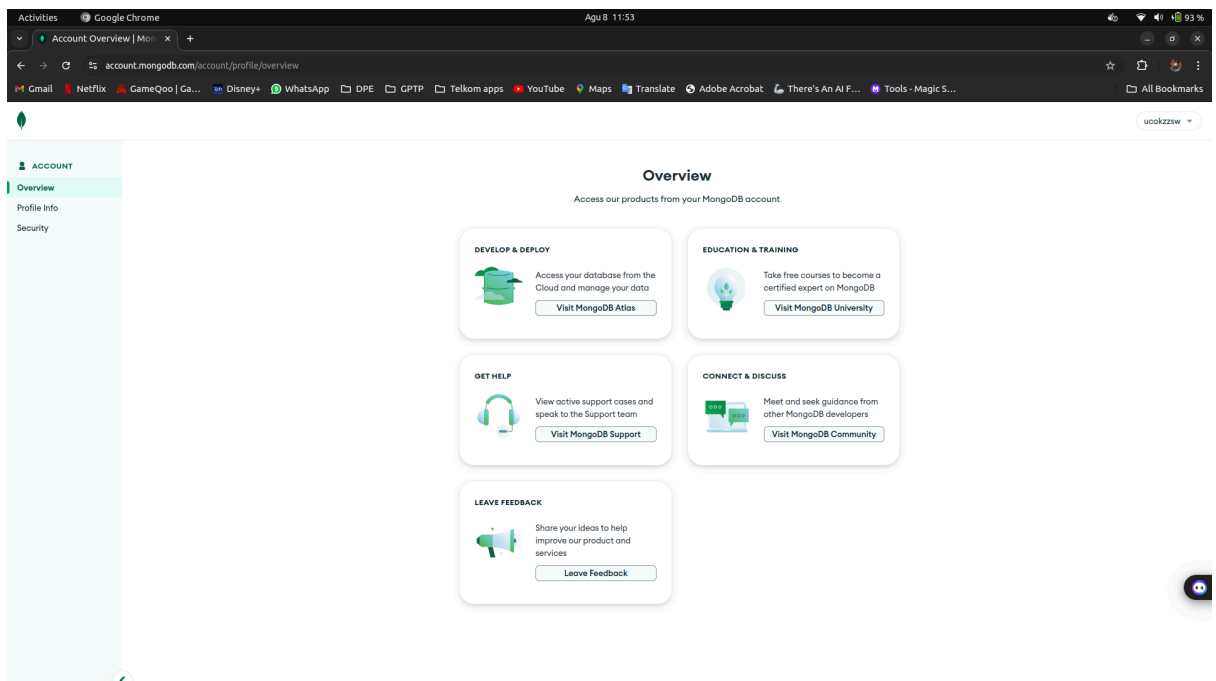


3. Jika kesulitan menggunakan mongoDB di local, bisa melakukan pendaftaran database free untuk mongoDB yang di sediakan di atlasian dengan cara membuat user di [mogodb.com](https://mongodb.com)

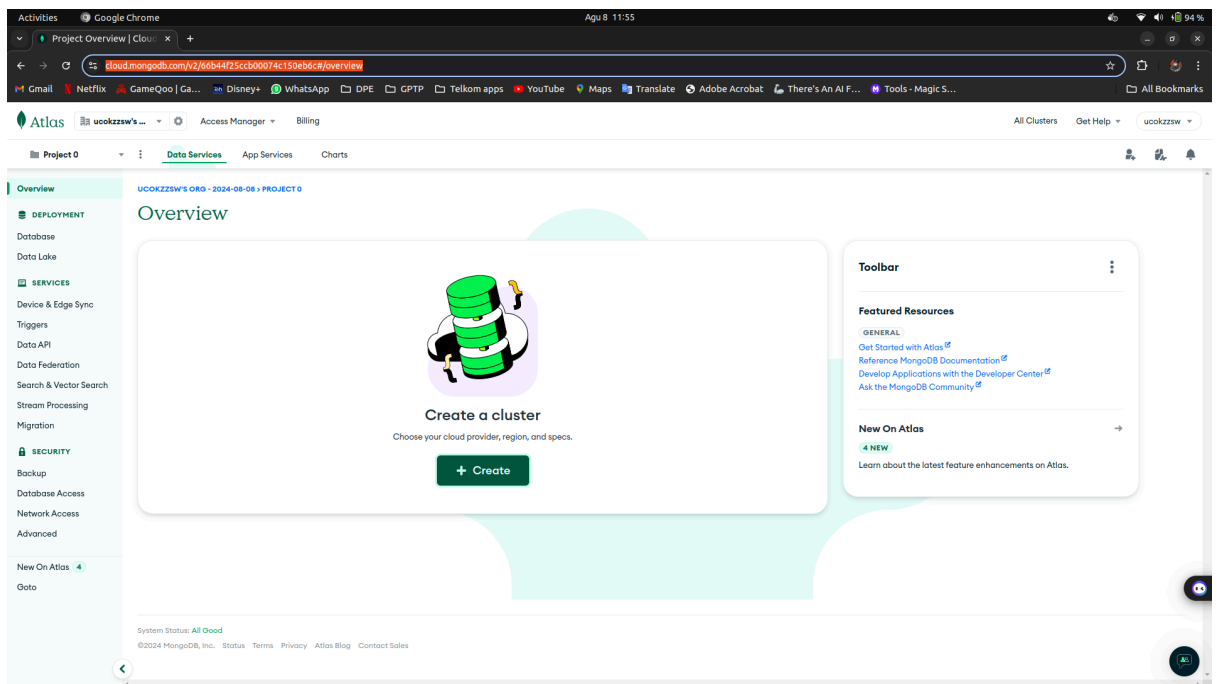


The screenshot shows the MongoDB Atlas registration page in a Google Chrome browser. The page has a dark green background with the MongoDB logo at the top. A white sign-up form is centered, titled "Sign up" with the subtitle "See what Atlas is capable of for free". The form includes a "Sign up with Google" button, input fields for "First Name\*", "Last Name\*", "Company", "Email\*", and "Password\*", and a checkbox for "I agree to the Terms of Service and Privacy Policy". A green "Create your Atlas account" button is at the bottom.

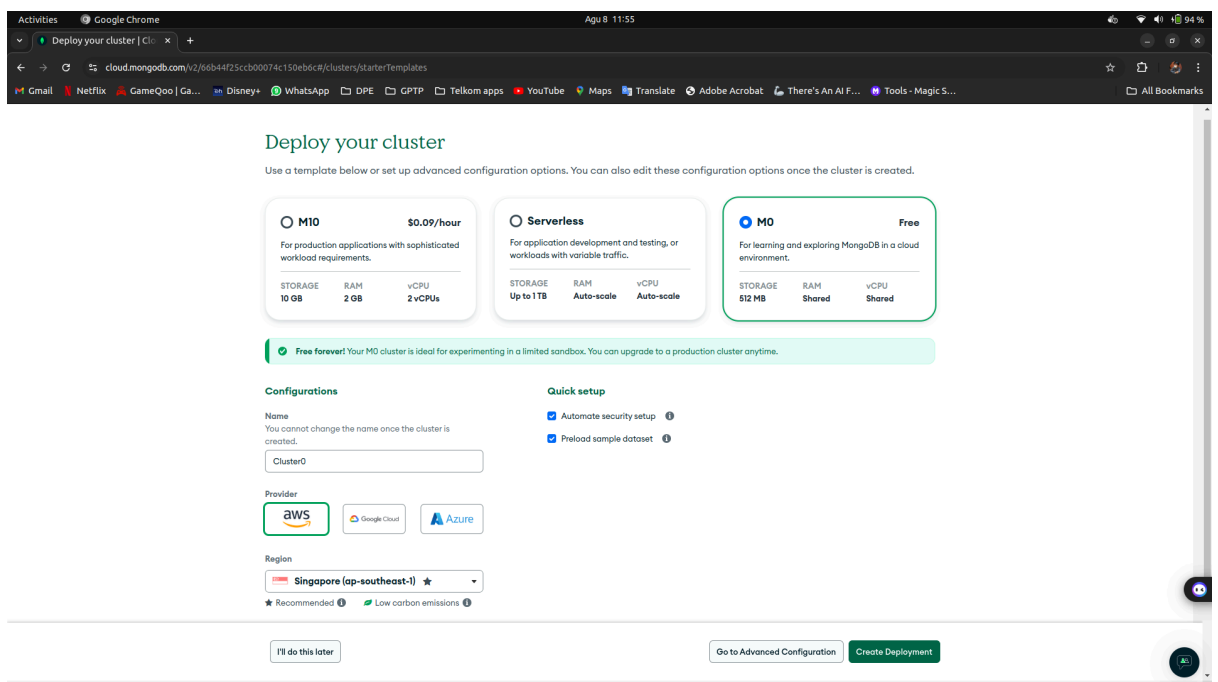
4. Ketika sudah berhasil masuk, kemudian pilih “[visiting mongoDB Atlas](#)”



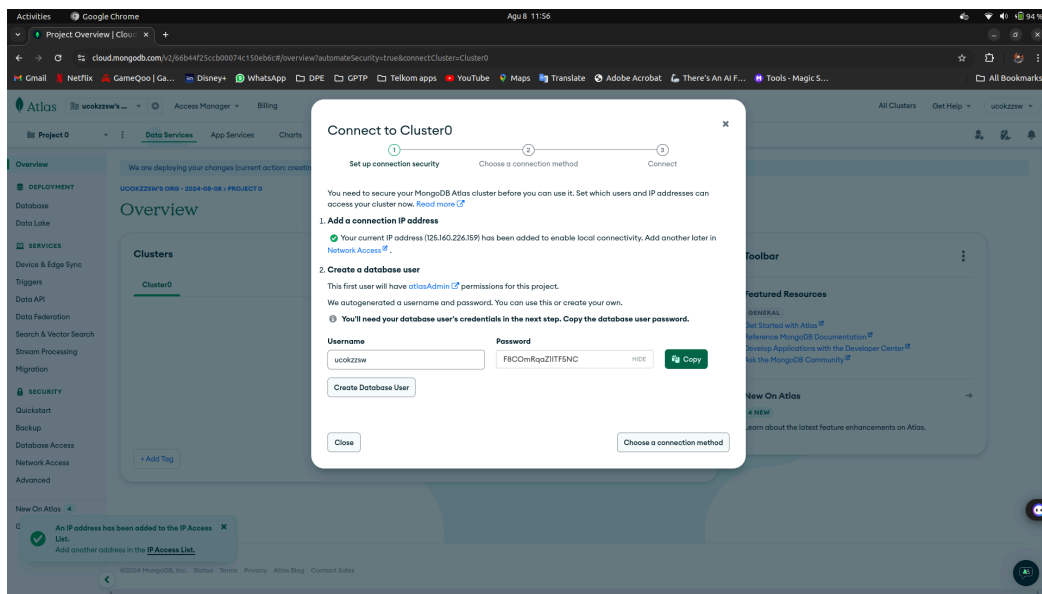
## 5. Create cluster



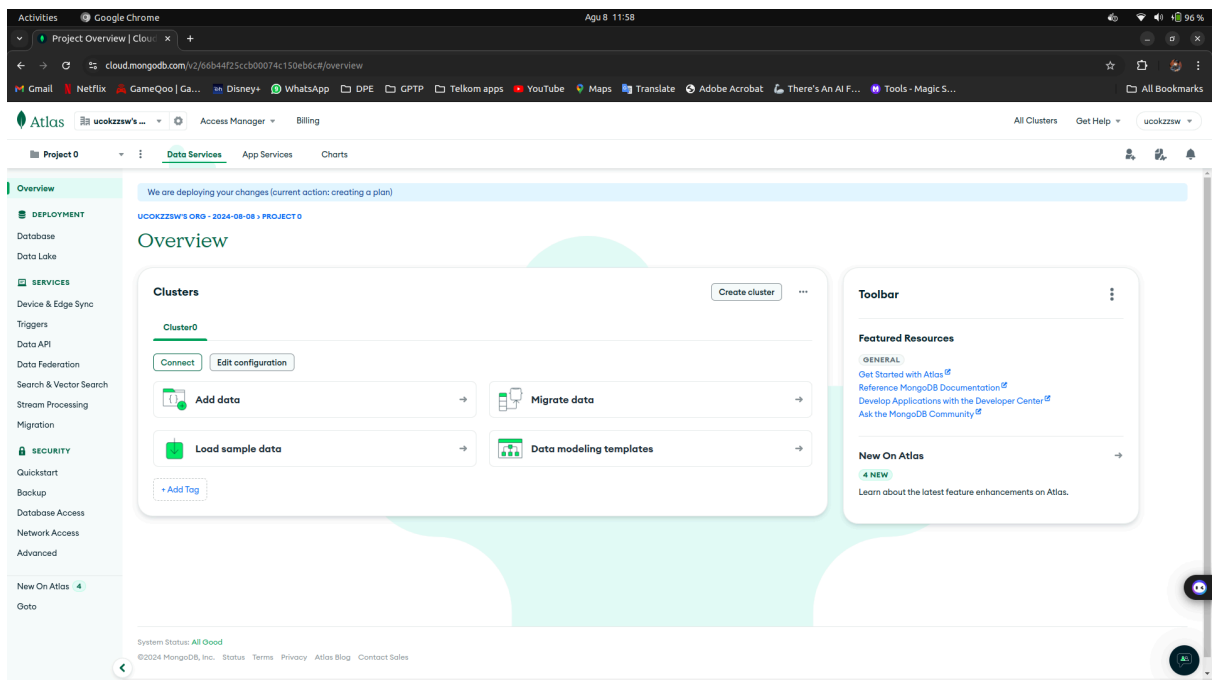
## 6. Pilih M0 free



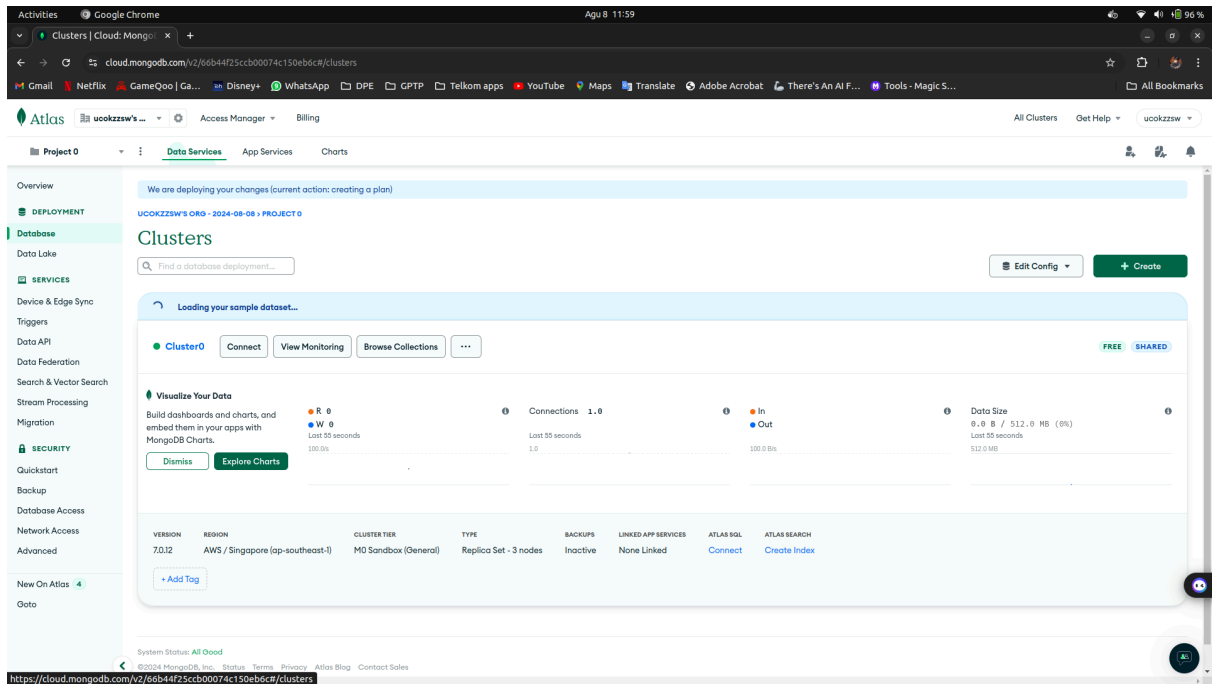
## 7. Create database user and close



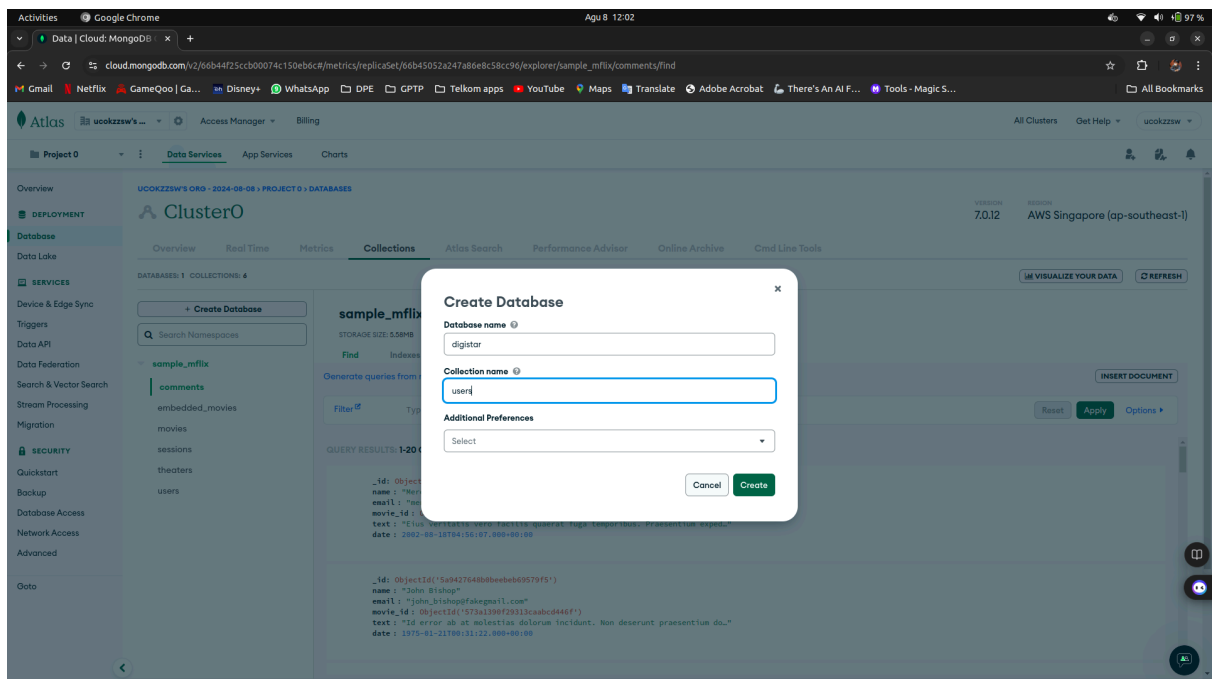
## 8. free cluster sudah siap digunakan



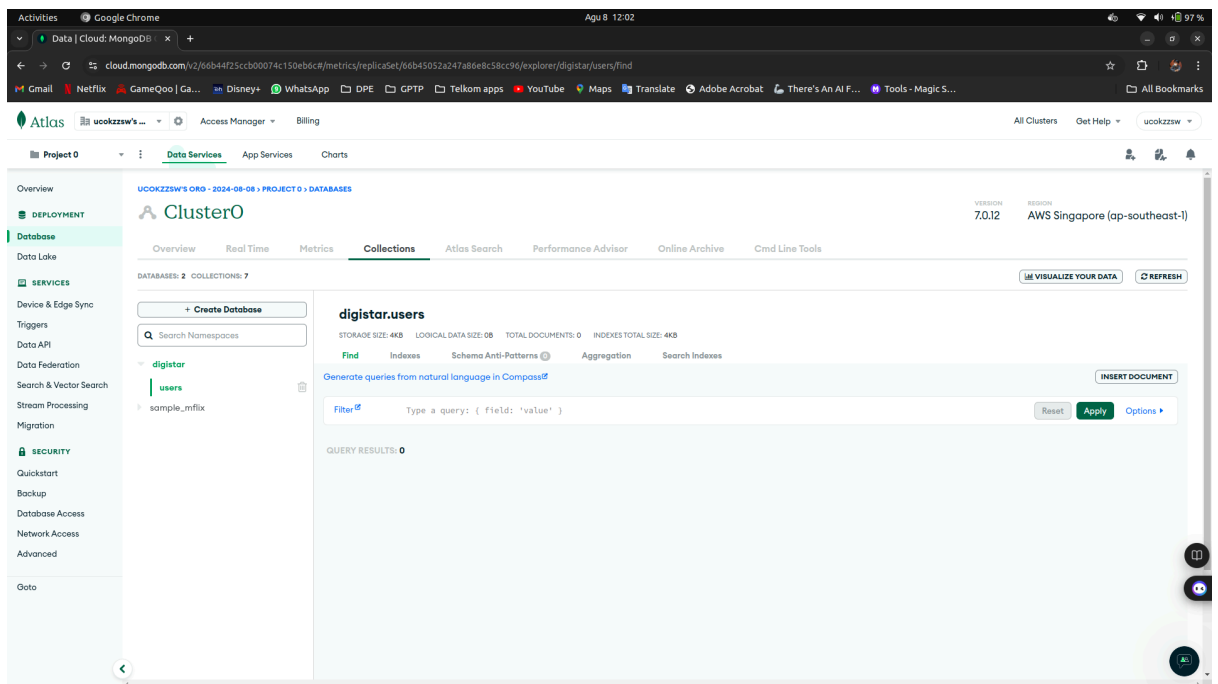
## 9. Pilih Database, kemudian Browse Collection



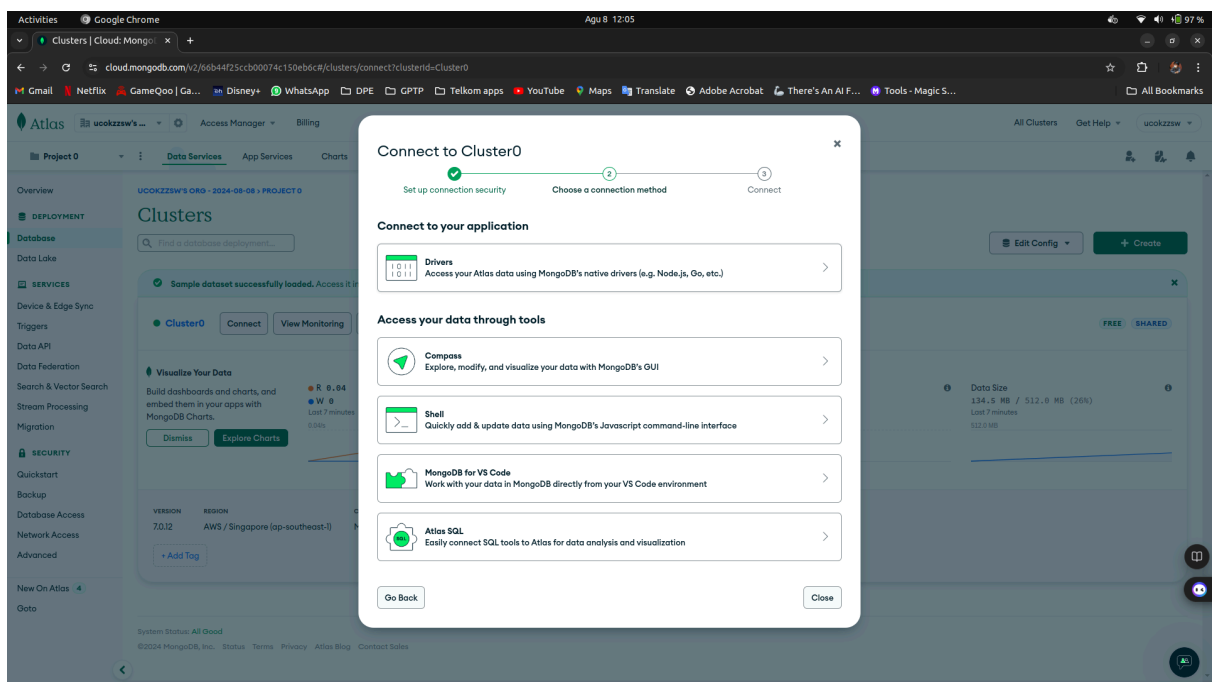
## 10. Telah di buatkan database sample, buatkan database baru dengan nama digistar dengan collection users



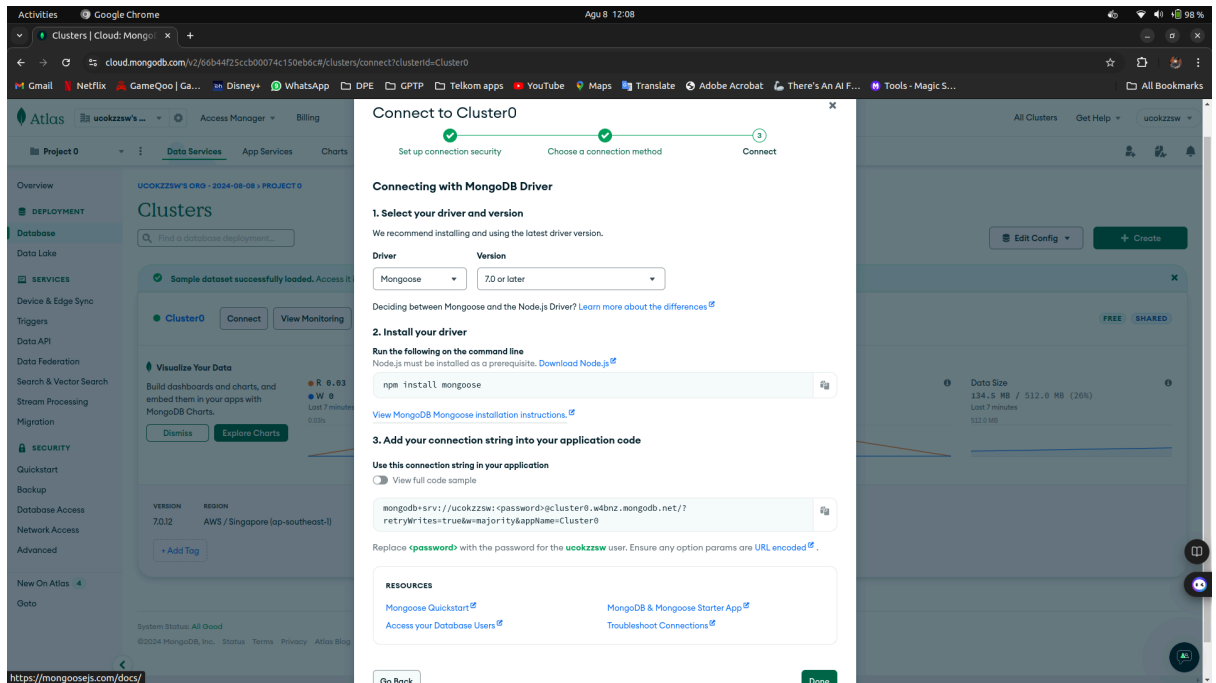
## 11. Database siap digunakan



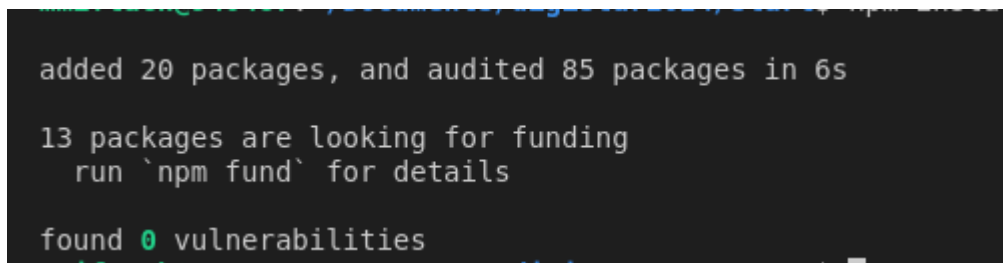
## 12. Jika ingin mengkoneksikan, database, bisa di dengan check connect



13. Pada modul kali ini kita menggunakan mongoose sebagai library yang digunakan untuk connect ke mongo db, pilih drivers, kemudian pilih mongoose, dan untuk code sample koneksi, bisa dengan mengaktifkan view full code sample(jangan lupa mengganti password dengan password user)



14. Pada petunjuk di anjurkan untukn npm install mongoose di local

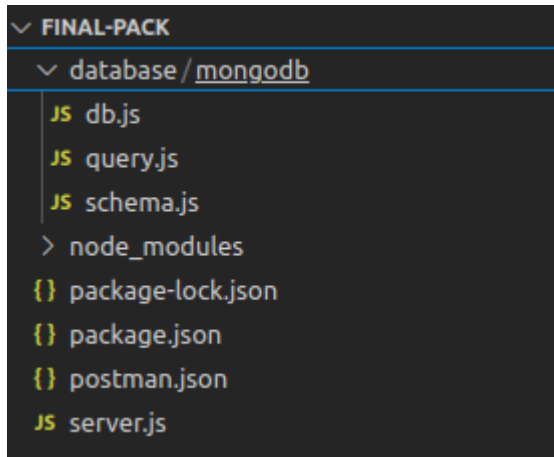


15. Untuk query pada mongoose bisa di cek [query](#)

Mongoose [models](#) provide several static helper functions for [CRUD operations](#). Each of these functions returns a [mongoose Query object](#).

- `Model.deleteMany()`
- `Model.deleteOne()`
- `Model.find()`
- `Model.findById()`
- `Model.findByIdAndDelete()`
- `Model.findByIdAndRemove()`
- `Model.findByIdAndUpdate()`
- `Model.findOne()`
- `Model.findOneAndDelete()`
- `Model.findOneAndReplace()`
- `Model.findOneAndUpdate()`
- `Model.replaceOne()`
- `Model.updateMany()`
- `Model.updateOne()`

16. Final dari aplikasi struktur folder akan seperti berikut.



17. Peruntukan file db.js sebagai fungsi untuk terhubung ke mongodb

```
1  const mongoose = require('mongoose');
2
3
4  const uri = "mongodb+srv://user:user@example.7izueqk.mongodb.net/digistar?retryWrites=true&w=majority&appName=example";
5  const clientOptions = { serverApi: { version: '1', strict: true, deprecationErrors: true } };
6
7  async function connectDB() {
8    try {
9      await mongoose.connect(uri, clientOptions);
10     await mongoose.connection.db.admin().command({ ping: 1 });
11     console.log('MongoDB connected successfully');
12   } catch (error) {
13     console.error('MongoDB connection error:', error);
14     process.exit(1); // Exit process with failure
15   }
16 }
17
18 async function disconnectDB() {
19   try {
20     await mongoose.disconnect();
21     console.log('MongoDB disconnected successfully');
22   } catch (error) {
23     console.error('MongoDB disconnection error:', error);
24     process.exit(1); // Exit process with failure
25   }
26 }
27
28
29
30 module.exports = {
31   connectDB,
32   disconnectDB
33 };
```



18. Peruntukan file schema.js sebagai struktur dari data yang ingin di simpan

```
1 const mongoose = require('mongoose');
2
3 // Define the user schema
4 const userSchema = new mongoose.Schema({
5   name: String,
6   email: String
7 });
8
9 module.exports = {
10   userSchema
11 };
12
13
14
```

19. Peruntukan query, sebagai query untuk pada mongoose

```
1  const mongoose = require('mongoose');
2  const schema = require('./schema');
3
4  const Users = mongoose.model('User', schema.userSchema);
5
6  async function getUsers() {
7    return Users.find();
8  }
9
10 async function createUser(user) {
11   return Users.create(user);
12 }
13
14 async function updateUser(id, user) {
15   return Users.findByIdAndUpdate(id, user, { new: true });
16 }
17
18 async function deleteUser(id) {
19   return Users.findByIdAndDelete(id);
20 }
21
22 async function findByName(name) {
23   return Users.find({ name: name });
24 }
25
26 module.exports = {
27   getUsers,
28   createUser,
29   updateUser,
30   deleteUser,
31   findByName
32 }
```

20. Pemanggilan inisiasi pada server.js

```
1 const mongoose = require('mongoose');
2 const userQuery = require('./database/mongoose/query');
3
4 mongoose.connectDB();
5
```

## 21. Pemanggilan pada server.js

```
30 // Route to GET all users - returns the users array as JSON
31 app.get('/users', (req, res) => {
32   userQuery.getUsers().then((users) => {
33     res.json(users);
34   });
35 });
36
37 // Route to POST a new user - adds a new user to the users array
38 app.post('/users', (req, res) => {
39   const user = req.body; // Extract the user from the request body
40   console.log(req);
41   userQuery.createUser(user).then((user) => {
42     res.status(201).json(user); // Respond with the created user and status code 201
43   });
44 });
45
46 // Route to PUT (update) a user by id
47 app.put('/users/:id', (req, res) => {
48   const { id } = req.params; // Extract the id from the request parameters
49   const user = req.body; // Extract the updated user from the request body
50   userQuery.updateUser(id, user).then((user) => {
51     res.status(200).json(user); // Respond with the updated user
52   });
53 });
54
55 // Route to DELETE a user by id
56 app.delete('/users/:id', (req, res) => {
57   const { id } = req.params; // Extract the id from the request parameters
58   userQuery.deleteUser(id).then(() => {
59     res.status(204).send(); // Respond with no content and status code 204
60   });
61 });
62
63 // Route to search users by name
64 app.get('/users/search', (req, res) => {
65   const { name } = req.query; // Extract the name query parameter
66
67   // Check if the name query parameter is provided
68   if (!name) {
69     return res.status(400).send({ message: "Name query parameter is required" });
70   }
71   userQuery.findByName(name).then((users) => {
72     res.status(200).json(users); // Respond with the filtered users
73   });
74 });
75 });
```

Semua persiapan sebelum hands-on telah selesai.