First MEAN partie 2

Objectif

Créer un projet avec la stack MEAN
Créer un serveur avec node et express
Créer un front end angular
gérer le cross origin resource sharing

Get et Post

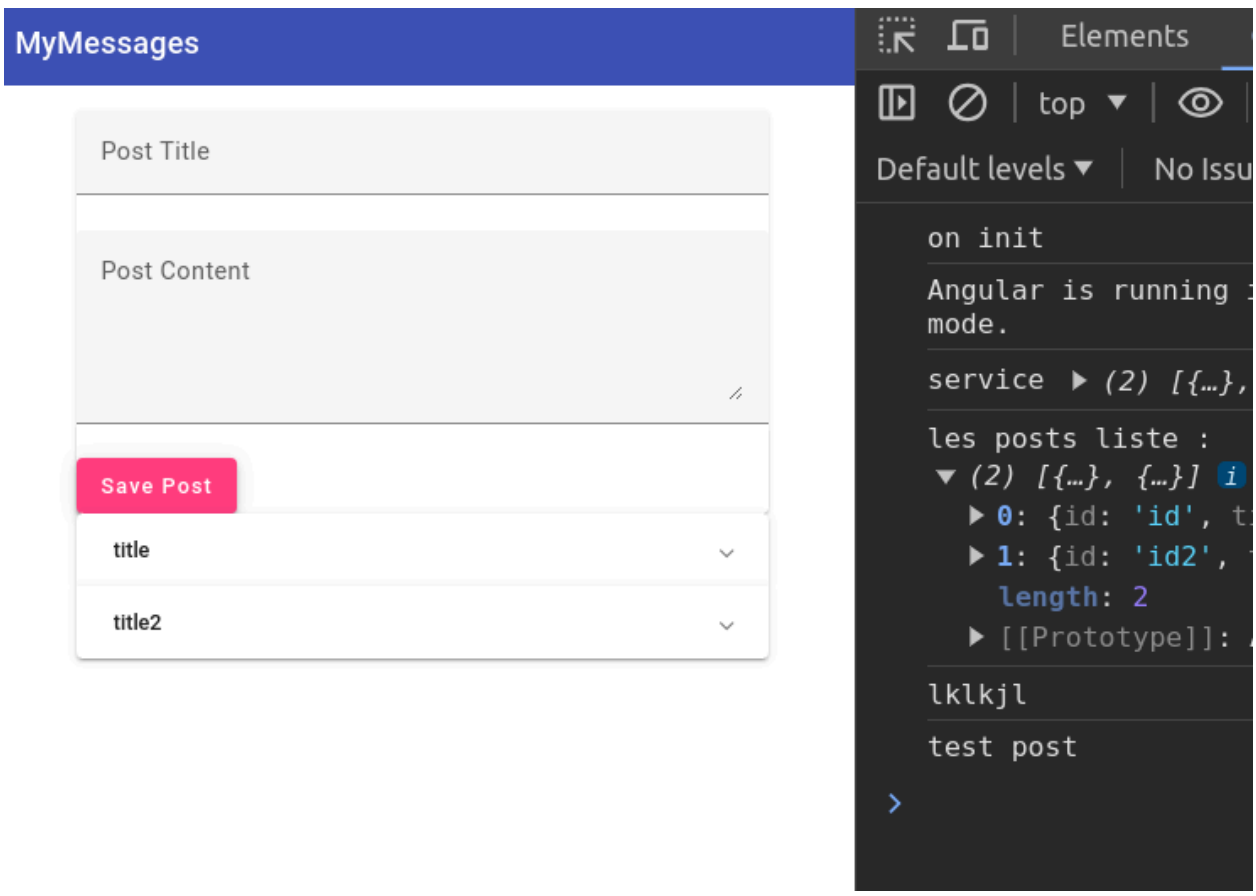
Modifions le middleware generique en get et post

```
18
19  app.post("/api/posts", (req, res, next) => {
20    const post = req.body;
21    console.log(post);
22    res.status(201).json({
23      message: "Post added",
24    });
25    next();
26  });
27
28  // middleware
29  app.get("/api/posts", (req, res, next) => {
30    const posts = [
31      { id: "id", title: "title", content: "form express" },
32      { id: "id2", title: "title2", content: "form express2" },
33    ];
34
35    res.status(200).json({
36      message: "Posts from server",
37      posts: posts,
38    });
39  });
40
41  module.exports = app;
```

Dans post.create.ts

```
5
6   @Component({
7     selector: "app-post-create",
8     templateUrl: "./post-create.component.html",
9     styleUrls: ["./post-create.component.css"]
10  })
11  export class PostCreateComponent {
12    enteredTitle = "";
13    enteredContent = "";
14
15    constructor(public postsService: PostsService) {}
16
17    onAddPost(form: NgForm) {
18      if (form.invalid) {
19        return;
20      }
21      console.log(form.value.title);
22
23      this.postsService.addPost(form.value.title, form.value.content);
24      form.resetForm();
25    }
26  }
```

tester



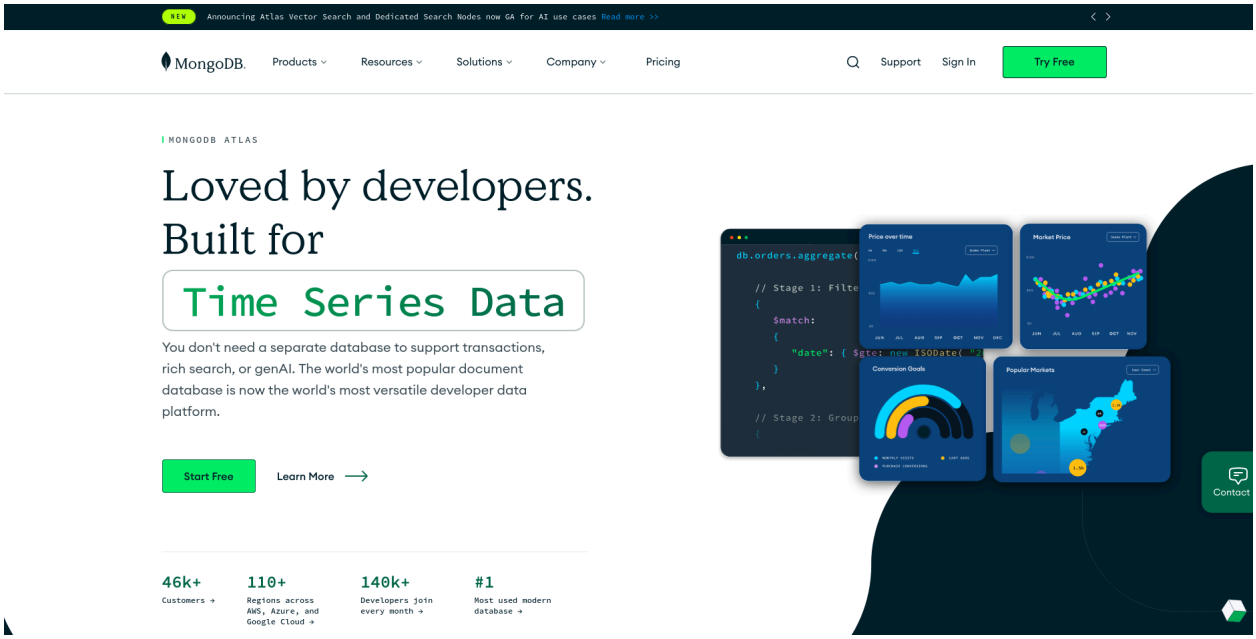
Il nous faut ajouter le modele
Pour extraire facilement la requête, il a un package nommé body-parser qui n’est plus nécessaire depuis la 4.16

```
backend > app.js > ...
const express = require("express");


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

```

MonogoDB et Mongoose

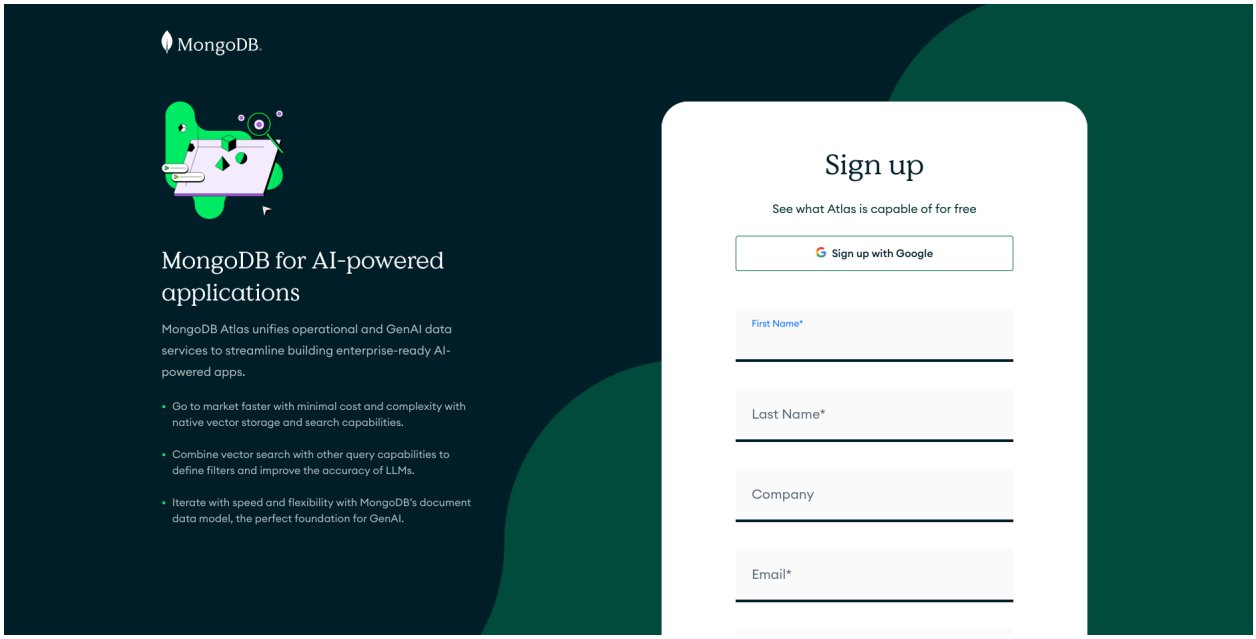
Ajoutons une base de données

Aller sur https://www.mongodb.com/



Try free !

Créer un compte ou se connecter si vous en avez déjà une



MongoDB.

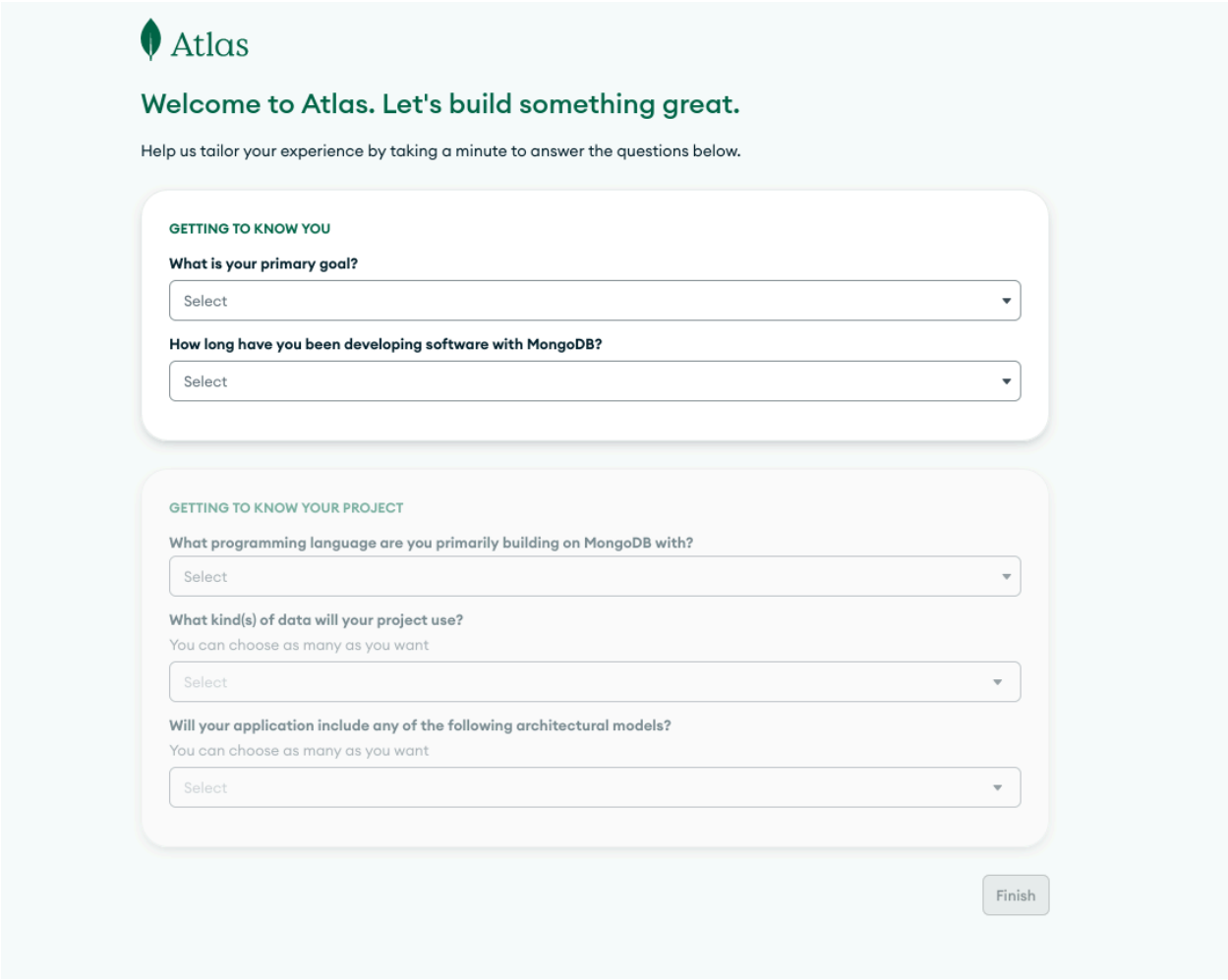
Accept Privacy Policy & Terms of Service

Please acknowledge the following terms and conditions to finish creating your account.

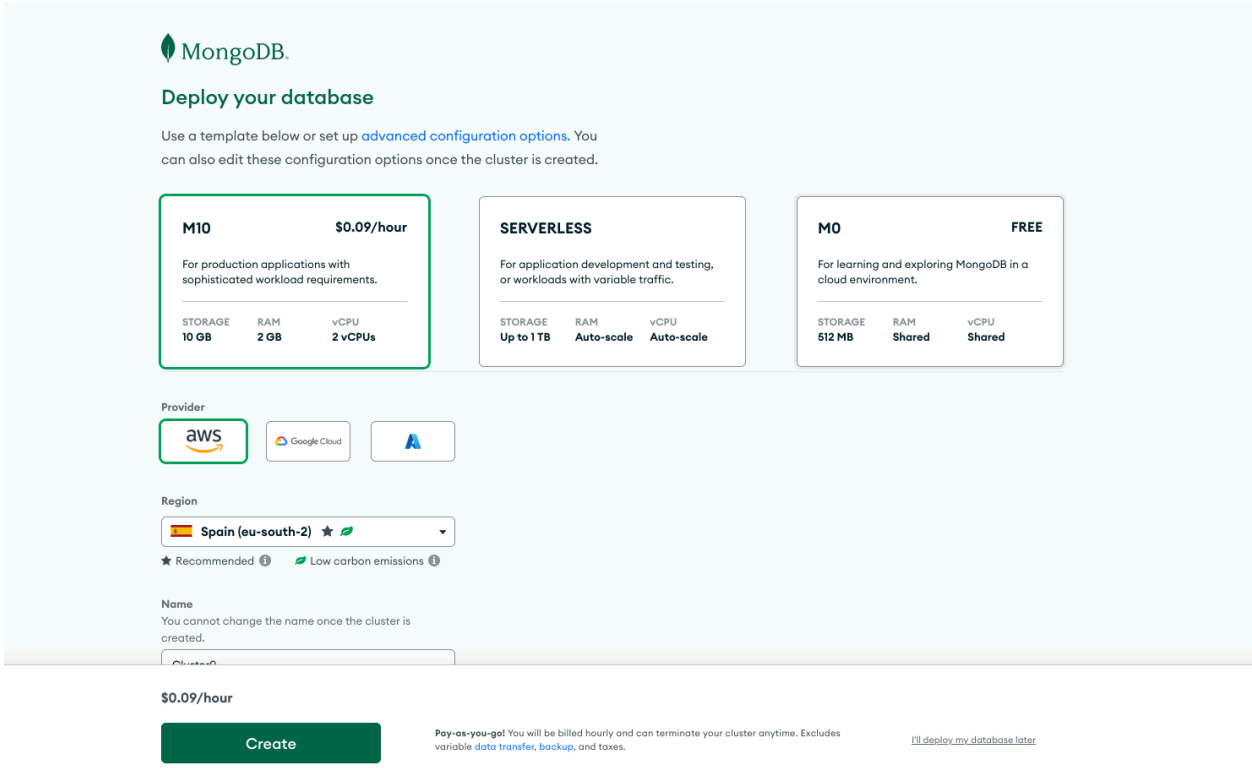
☐ I accept the Privacy Policy and the Terms of Service

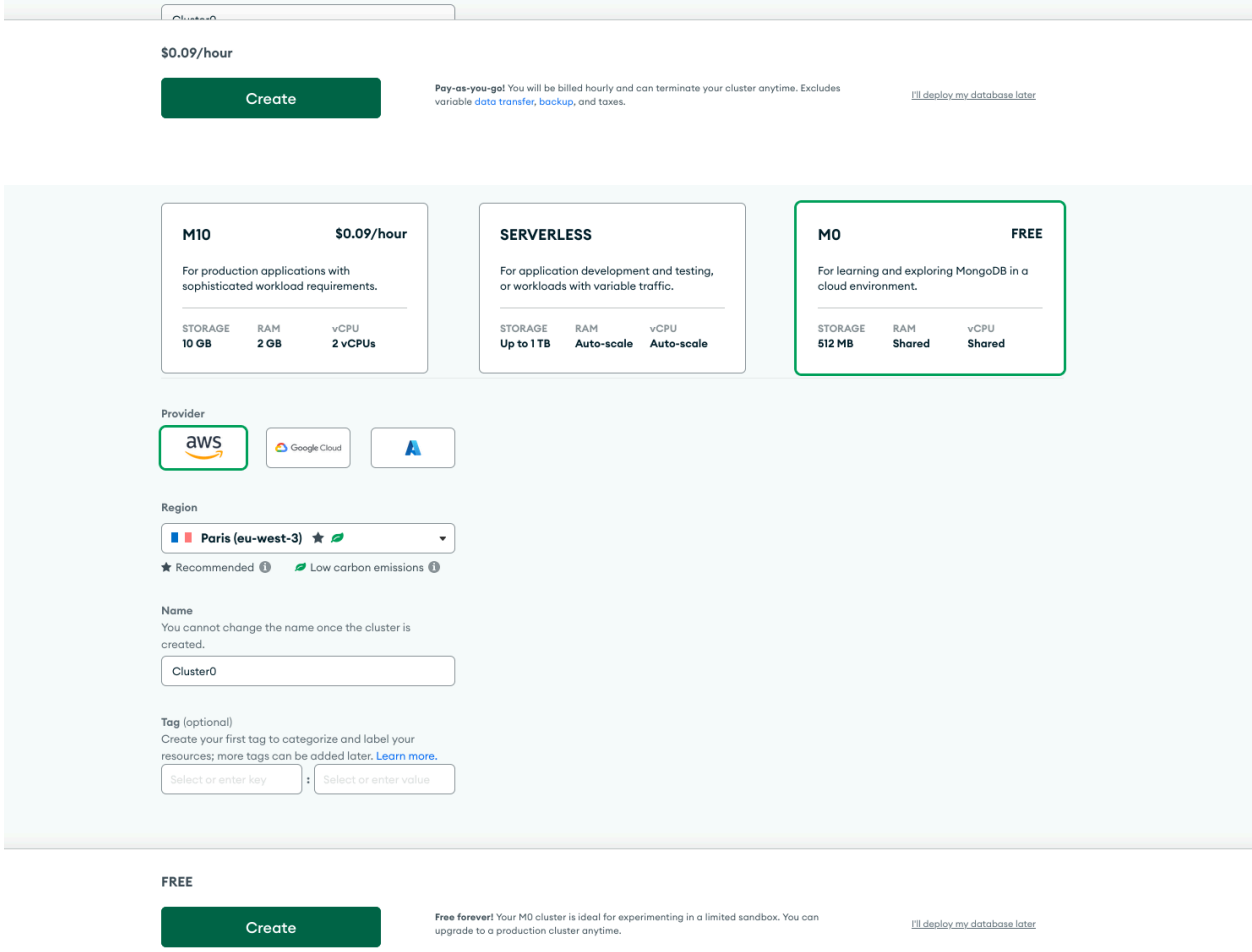
Cancel Signup | Submit

répondre au questionnaire



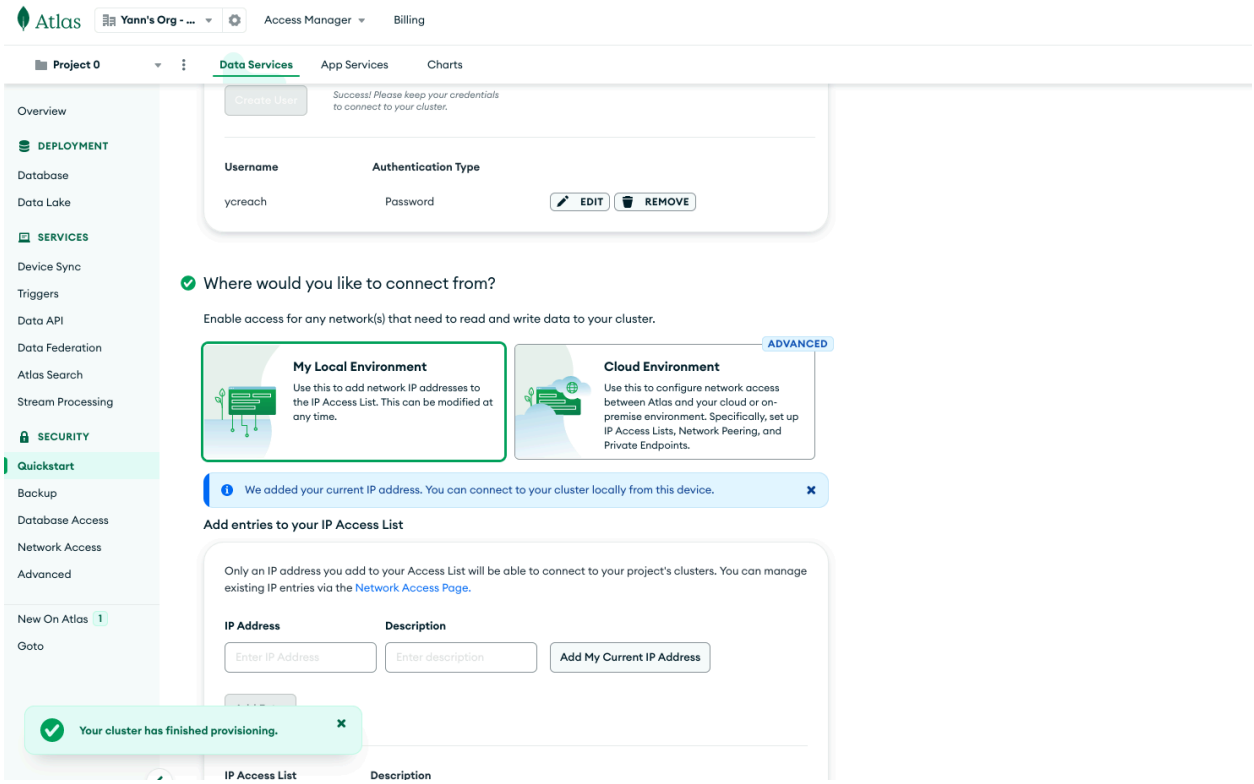
choisir free à gauche


MongoDB.
Deploy your database
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.
STORAGE 10 GB
RAM 2 GB
vCPU 2 vCPUs
SERVERLESS
For application development and testing, or workloads with variable traffic.
STORAGE Up to 1 TB
RAM Auto-scale
vCPU Auto-scale
M0
FREE
For learning and exploring MongoDB in a cloud environment.
STORAGE 512 MB
RAM Shared
vCPU Shared
Provider
aws
Google Cloud
Region
Spain (eu-south-2)
Recommended
Low carbon emissions
Name
You cannot change the name once the cluster is created.

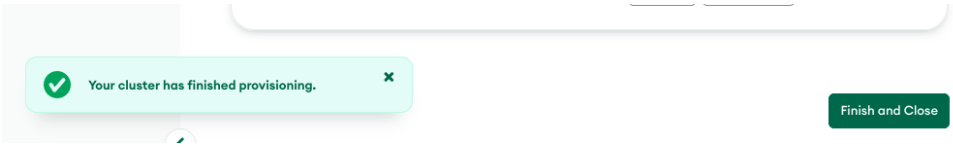

$0.09/hour
Create
Pay-as-you-go! You will be billed hourly and can terminate your cluster anytime. Excludes variable data transfer, backup, and taxes.
I'll deploy my database later
M10
$0.09/hour
For production applications with sophisticated workload requirements.
STORAGE 10 GB
RAM 2 GB
vCPU 2 vCPUs
SERVERLESS
For application development and testing, or workloads with variable traffic.
STORAGE Up to 1 TB
RAM Auto-scale
vCPU Auto-scale
M0
FREE
For learning and exploring MongoDB in a cloud environment.
STORAGE 512 MB
RAM Shared
vCPU Shared
Provider
aws
Google Cloud
Region
Paris (eu-west-3)
Recommended
Low carbon emissions
Name
You cannot change the name once the cluster is created.
Cluster0
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
FREE
Create
Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.
I'll deploy my database later

create

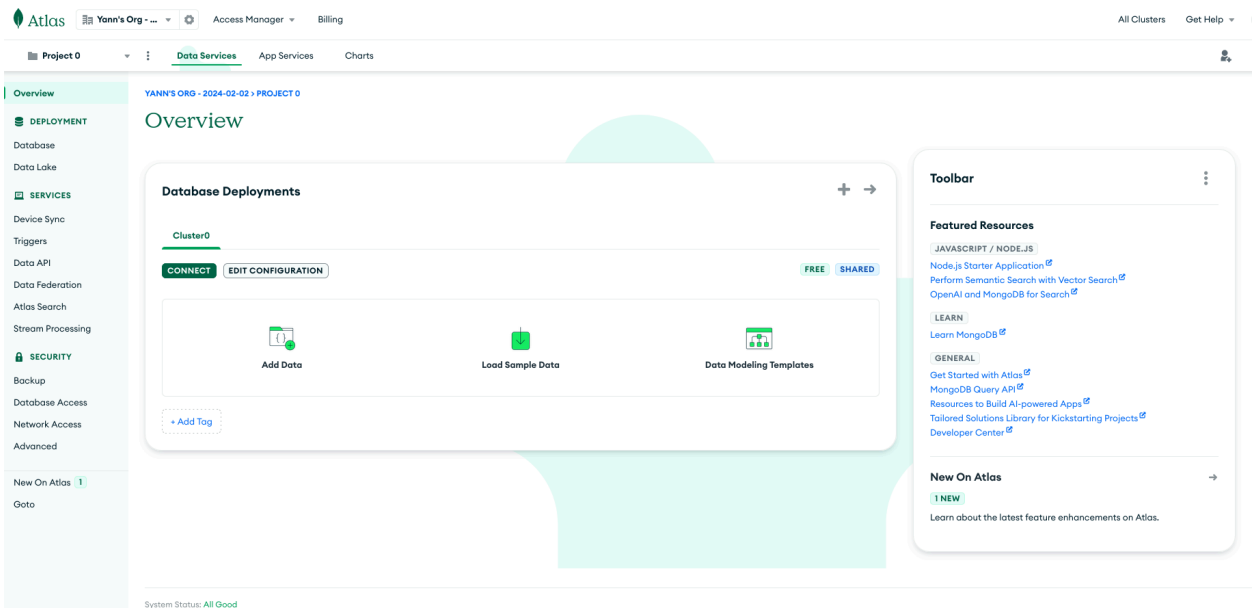
choisir son type de connection
password



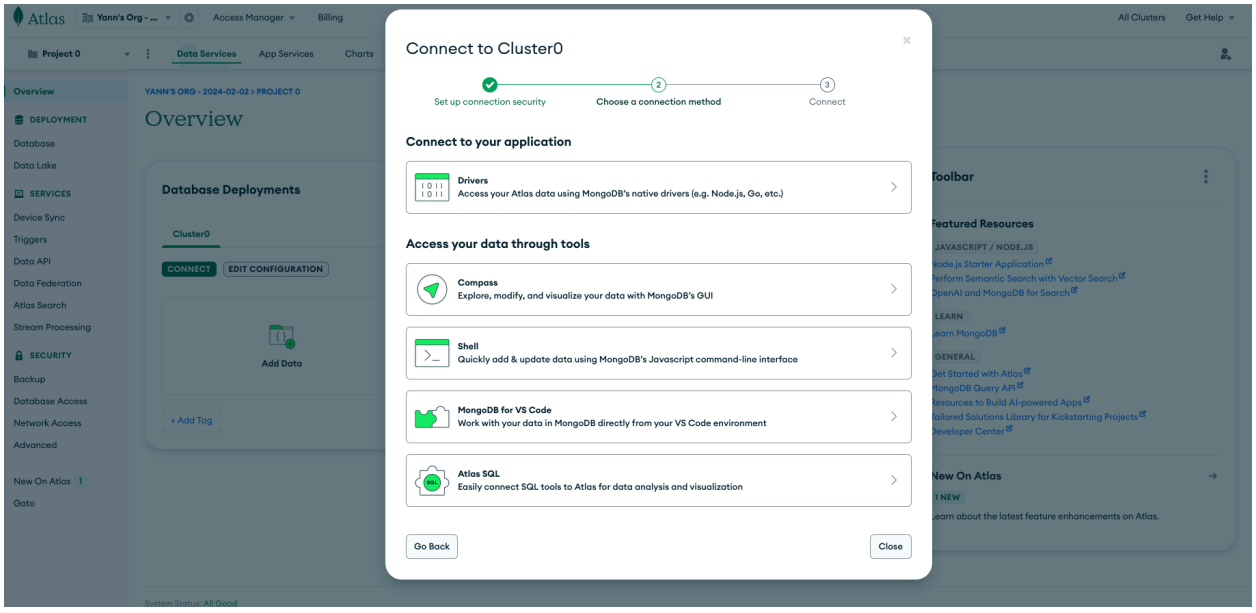
Local environment
et
add my current address



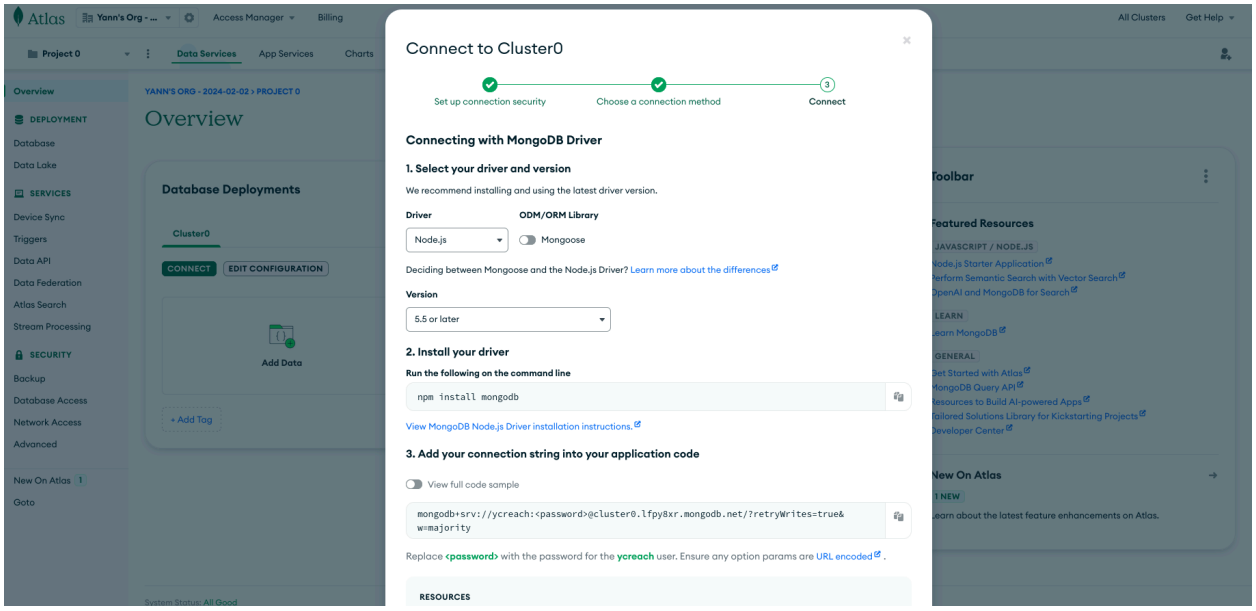
finish



connect



choisir driver



Conserver cette url

Mongoose : Un ‘Driver’ pour angular

Mongoose est une bibliothèque pour Node.js qui fournit une solution de modélisation des objets MongoDB (un ODM - Object Data Modeling) de manière schématique. Elle sert d'interface pour faciliter les interactions entre votre application Node.js et une base de données MongoDB

Mongoose inclut des mécanismes intégrés pour la validation, la requête, les hooks (middlewares), et plus encore, rendant le travail avec MongoDB plus intuitif et sécurisé comparé à l'utilisation du pilote MongoDB natif seul.

Une fois un schéma défini, Mongoose vous permet de créer des modèles basés sur ces schémas. Les modèles sont des constructeurs qui prennent des documents MongoDB et les enveloppent avec des fonctionnalités de modélisation.

Workspaces Applications Jan 24 6:47 PM en

Download and Install — No Mongoose ODM v8.1.0 Private browsing

https://mongoosejs.com

Fork me on GitHub

mongoose

elegant mongodb object modeling for node.js

read the docs discover plugins

Star 26,367 Version 8.1.0 Fork 3,826

Let's face it, **writing MongoDB validation, casting and business logic boilerplate is a drag**. That's why we wrote Mongoose.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://127.0.0.1:27017/test');

const Cat = mongoose.model('Cat', { name: String });

const kitty = new Cat({ name: 'Zildjian' });
kitty.save().then(() => console.log('meow'));
```

Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

Get Professionally Supported Mongoose

getting started

- quick start guide

support

```
yann@pop-os:~/Documents/FORMATIONS/ANGULAR/sources/simplemean$ npm install --save mongoose

added 17 packages, and audited 998 packages in 10s

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

found 0 vulnerabilities
```

Retour au code : Ajoutons un schéma pour le serveur

Dans backend/model/post.js

```
backend > models > JS post.js > <unknown>
const mongoose = require('mongoose')
const postSchema = mongoose.Schema({
    title: {type: String, required: true},
    content: {type: String, required: true}
})

module.exports = mongoose.model('Post', postSchema)
```

Dans app.js
on ajoute le modèle et mongoose

```
SIMPLEMEAN
> .angular
> .vscode
v backend
  v models
    JS post.js   U
  JS app.js      U
> node_modules
```

```
backend > JS app.js > mongoose
const express = require("express");
const Post = require('./models/post')
const mongoose = require('mongoose')

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

Modifions app.post

```
app.post("/api/posts", (req, res, next) => {

  console.log('==========================='+post)
  const post = new Post({
      title: req.body.title,
      content: req.body.content
  })
  res.status(201).json({
      message: 'Post added'
  })
  next()

})
```

Testons

On vois un post dans console serveur

```
============================{
  title: ';lk;lk',
  content: ';lk;lk',
  _id: new ObjectId('65bc96dec0e142971aa64b25')
}
```

Connectons nous à la base

Il faut donner les accès à express
dans app.js

```
mongoose.connect('mongodb+srv://creachyann:password@cluster0.sqgldfc.mongodb.net/?retryWrites=true&w=majority')
.then(() => console.log('Connected to atlas DB'))
.catch(()=> console.log('Connection failed'))
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
Connected to atlas DB
```

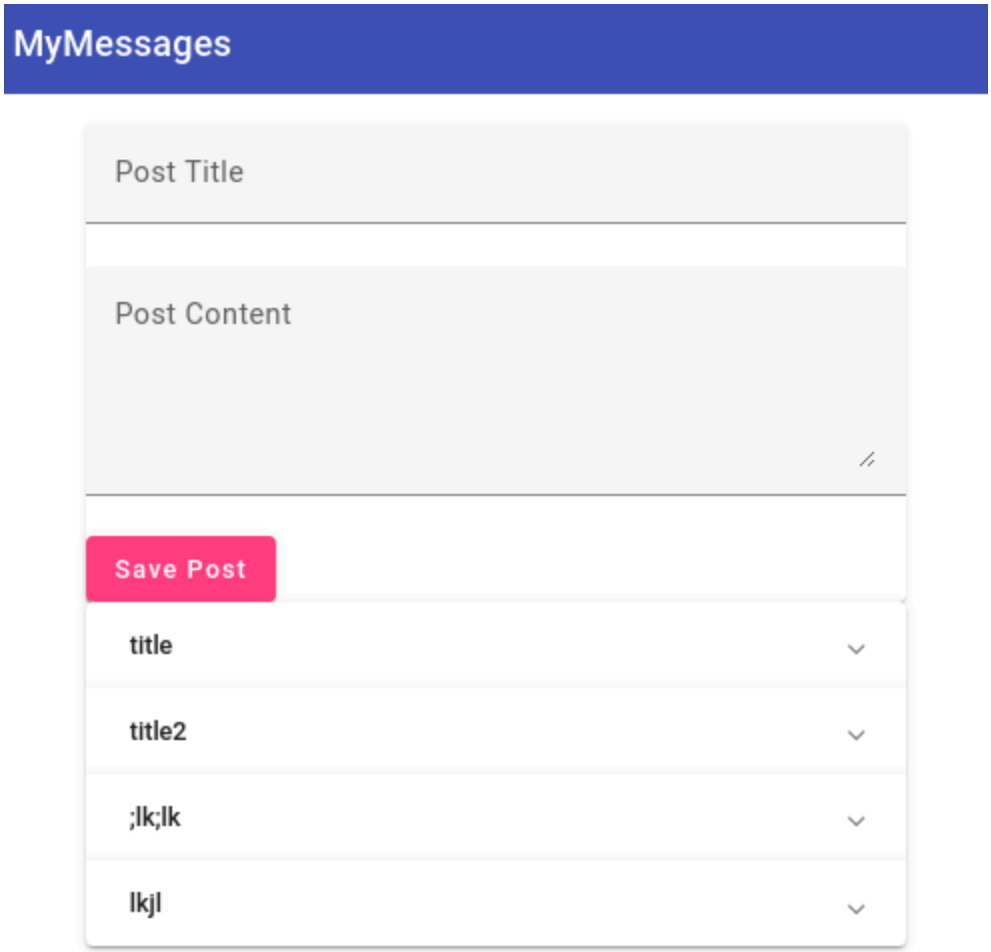
Bravo!

Essayons de sauvegarder un post

Toujours dans app.js

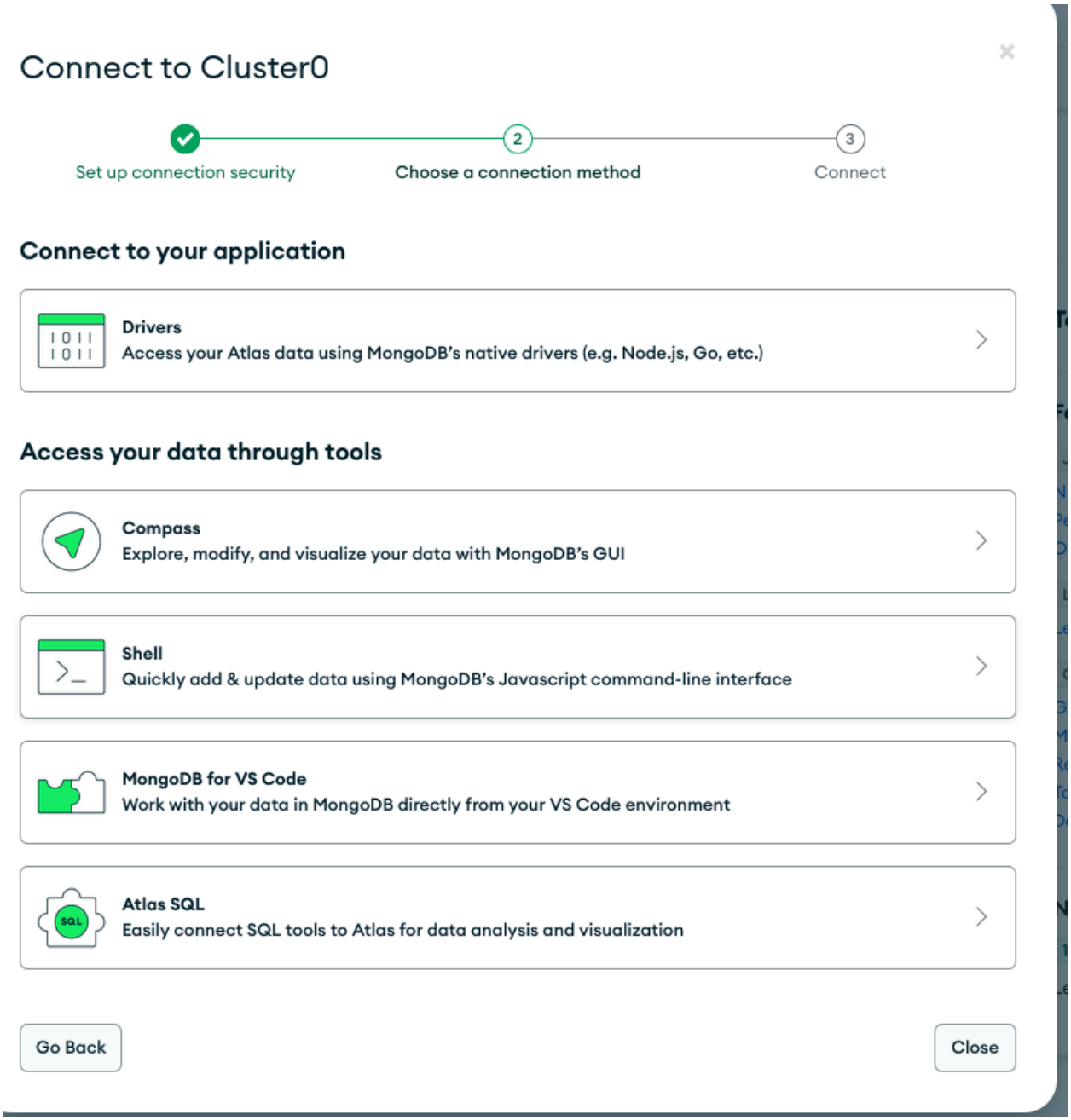
```
app.post("/api/posts", (req, res, next) => {
    //const post = req.body
    const post = new Post({
        title: req.body.title,
        content: req.body.content
    })
    console.log(post)
    post.save()
    res.status(201).json({
        message: 'Post added'
    })
    next()

})
```

Tester

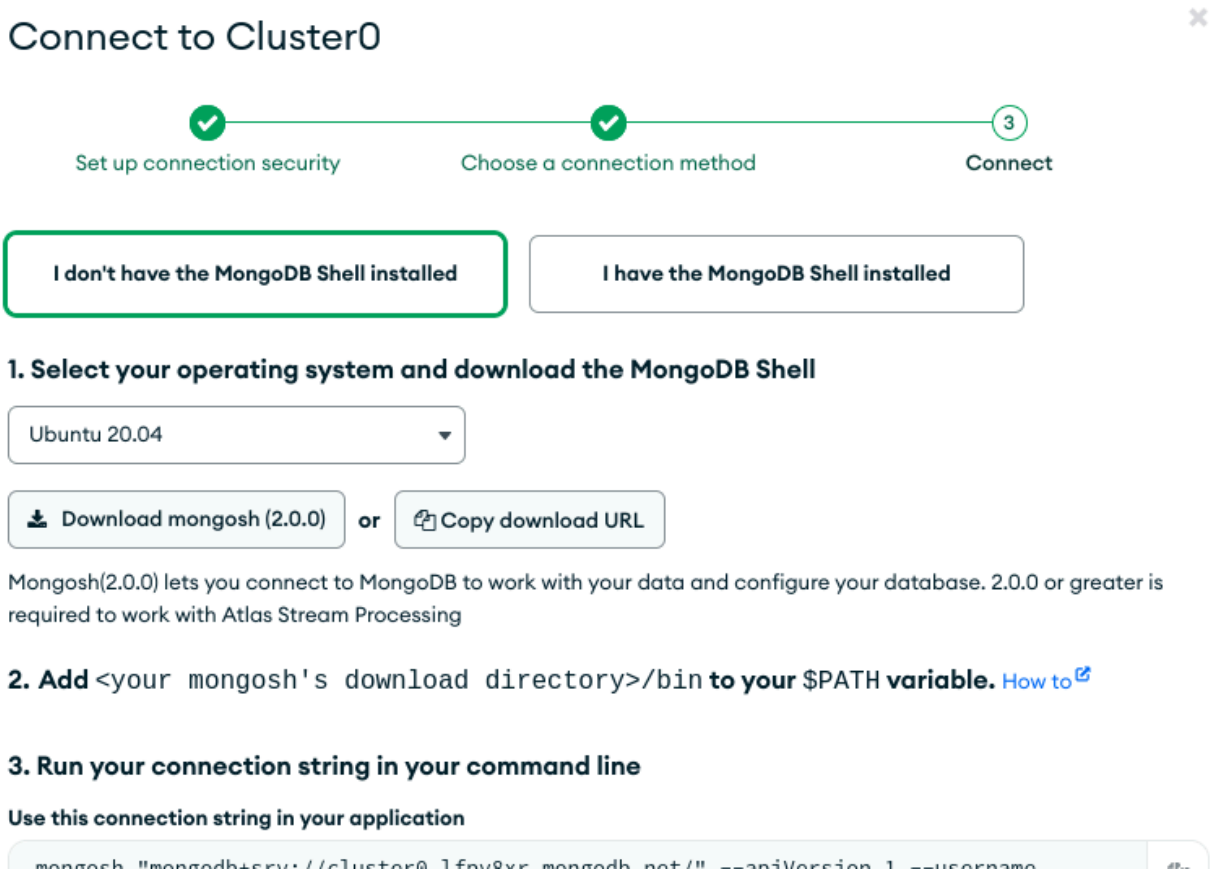



```
{
  title: 'lkjl',
  content: 'kkljlkj',
  _id: new ObjectId('65bc98d853f8bb7a9cb56b46')
}
```

Voir ses donnees

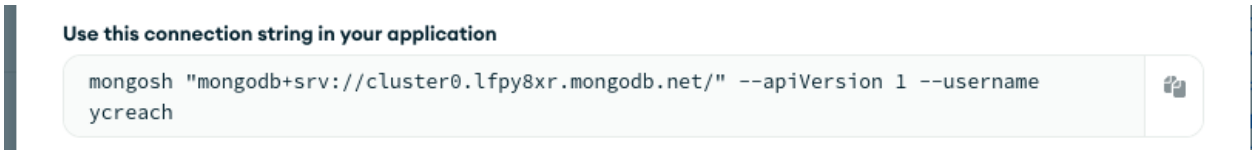


cliquer sur shell



Installer au besoin

Copier la commande



```
yann@pop-os:~$ mongosh "mongodb+srv://cluster0.sqgldfc.mongodb.net/" --apiVersion 1 --username
 creachyann
Enter password: ****************
Current Mongosh Log ID: 65b155c298ecd5ea6a08767a
Connecting to:          mongodb+srv://<credentials>@cluster0.sqgldfc.mongodb.net/?appName=mong
osh+2.1.1
Using MongoDB:          6.0.13 (API Version 1)
Using Mongosh:          2.1.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-af9okz-shard-0 [primary] test>
```

```
yann@pop-os:~$ mongosh "mongodb+srv://cluster0.sqgldfc.mongodb.net/" --apiVersion 1 --username
 creachyann
Enter password: ****************
Current Mongosh Log ID: 65b155c298ecd5ea6a08767a
Connecting to:          mongodb+srv://<credentials>@cluster0.sqgldfc.mongodb.net/?appName=mong
osh+2.1.1
Using MongoDB:          6.0.13 (API Version 1)
Using Mongosh:          2.1.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-af9okz-shard-0 [primary] test> use simplemeandb
switched to db simplemeandb
Atlas atlas-af9okz-shard-0 [primary] simplemeandb>
```

```
Atlas atlas-af9okz-shard-0 [primary] simplemeandb> show collections
posts
Atlas atlas-af9okz-shard-0 [primary] simplemeandb> db.p
db.propertyIsEnumerable           db.printCollectionStats
db.printSecondaryReplicationInfo  db.printReplicationInfo
db.posts

Atlas atlas-af9okz-shard-0 [primary] simplemeandb> db.p
db.propertyIsEnumerable           db.printCollectionStats
db.printSecondaryReplicationInfo  db.printReplicationInfo
db.posts

Atlas atlas-af9okz-shard-0 [primary] simplemeandb> db.posts.find()
[
  {
    _id: ObjectId('65b154eb2dc0a2d5921071fe'),
    title: 'Test atlas',
    content: 'atlas con',
    __v: 0
  }
]
Atlas atlas-af9okz-shard-0 [primary] simplemeandb>
```

Bravo!

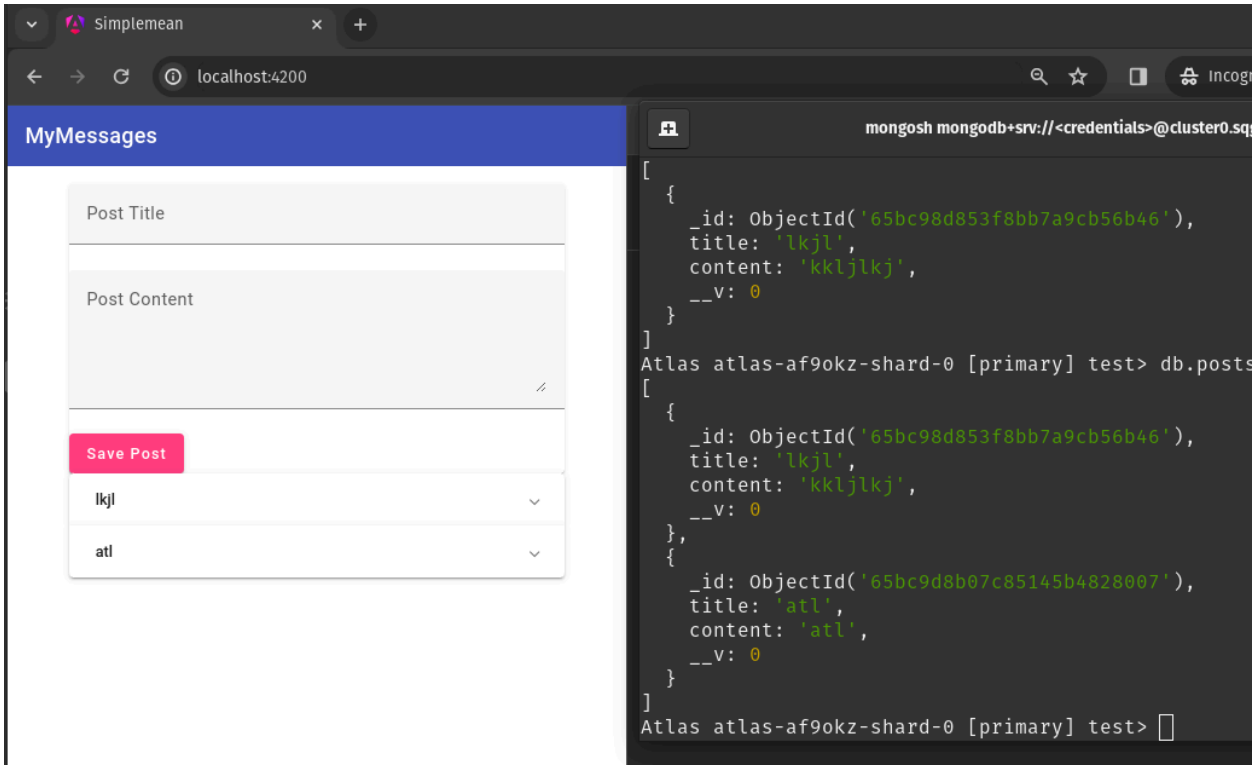
Le Get

Utilisons la méthode find

app.get devient:

```
// middleware
app.get("/api/posts", (req, res, next) => {
  Post.find().then((documents) => {
    res.status(200).json({
        message: "Posts from server",
        posts: documents,
      });
  });


});
```

Tester



Bravo !