Programmation Mongo

En javascript

JSON

```
/**
* Paste one or more documents here
*/
{
  "_id": {
    "$oid": "6570c1350b72abc0a367720f"
  },
  "nom": "Mésange à tête noire",
  "scientifique": "Poecile atricapillus",
  "description": "La mésange à tête noire est un petit oiseau au plumage noir et blanc, avec une calotte noire distinctive sur la tête. Elle est connue pour son comportement vif et curieux.",
  "couleur": "Noir et blanc",
  "poids": "Environ 12 grammes"
}
```

Node dans Ubuntu

```
sudo apt-get install nodejs
sudo apt-get install npm

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash
Relancer Recharger le shell
nvm install 18
nvm use 18

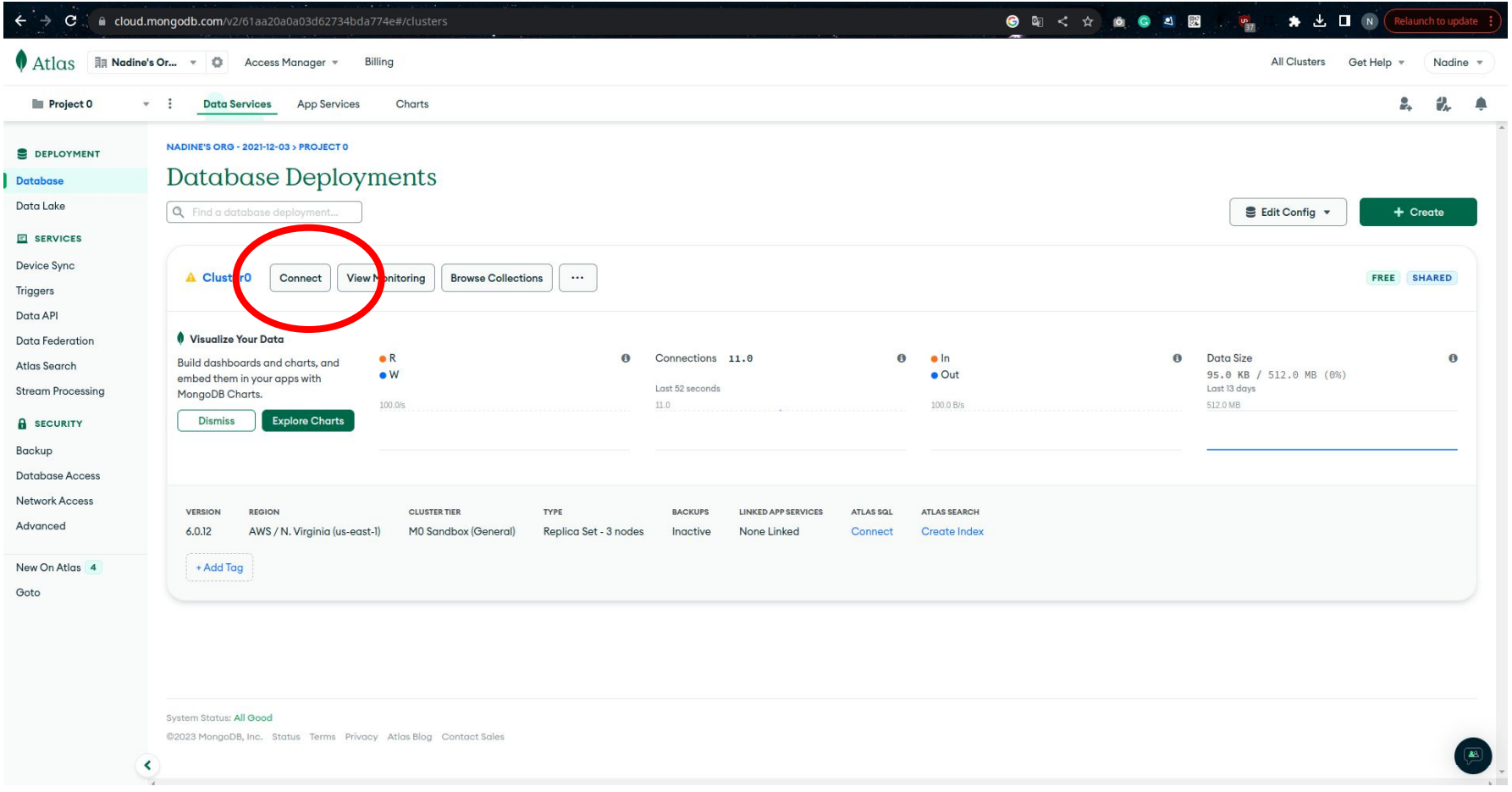
nvm install lts
nvm list
nvm install 20.18.5
```

Creation du projet

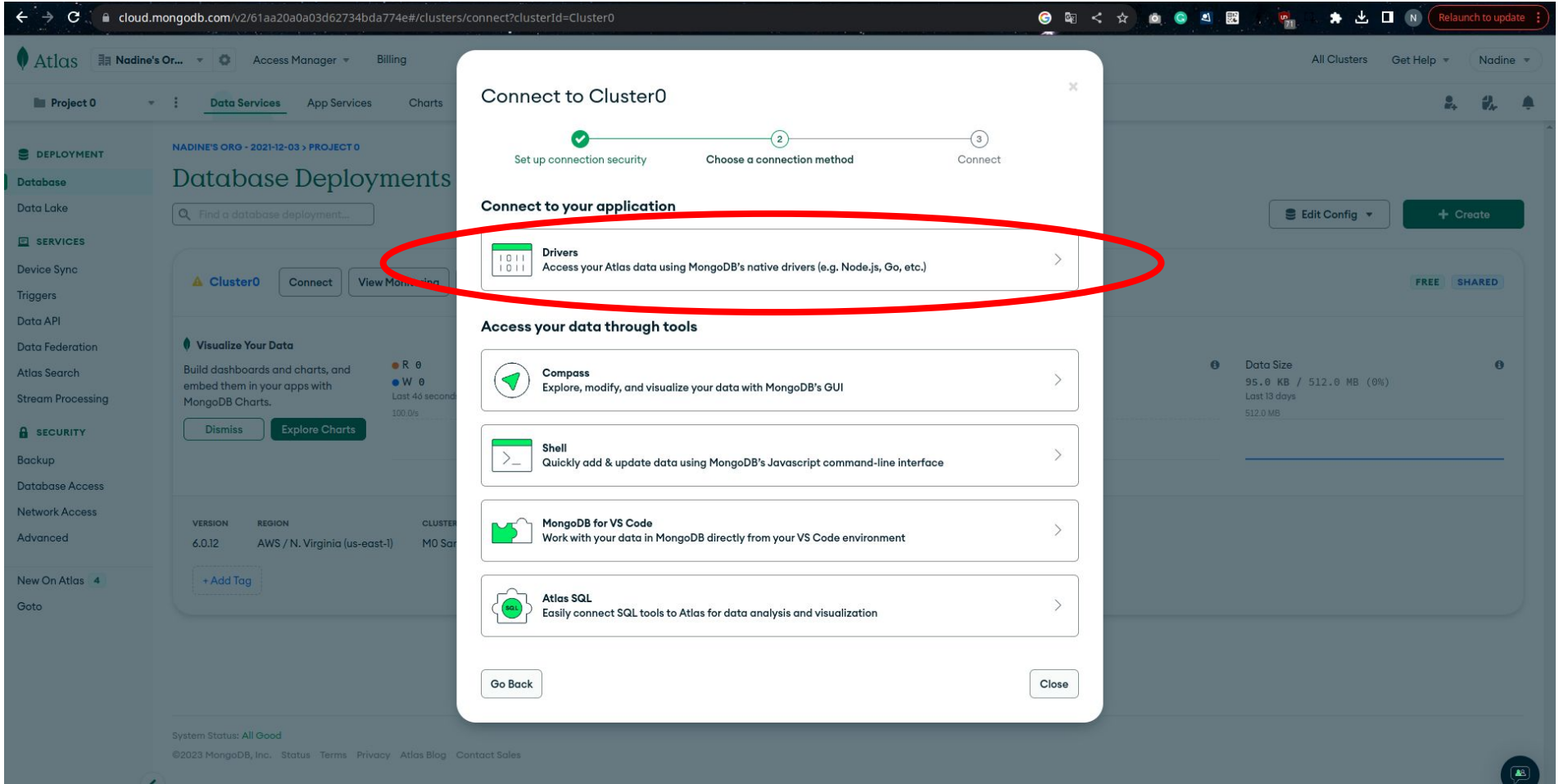
```
mkdir bestiaire
cd bestiaire

npm install mongodb
touch oiseau.js
node oiseau.js
```

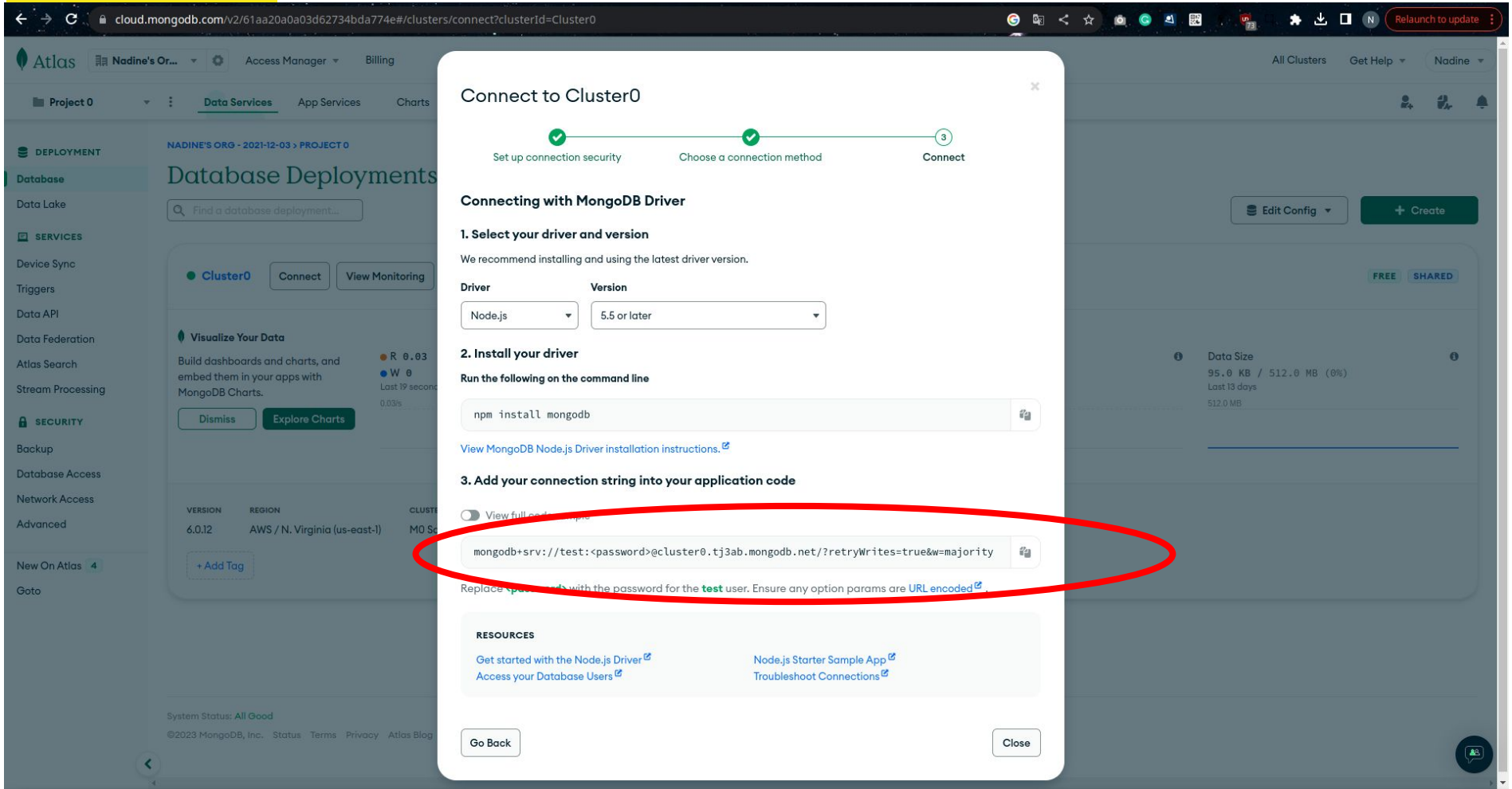
Se connecter



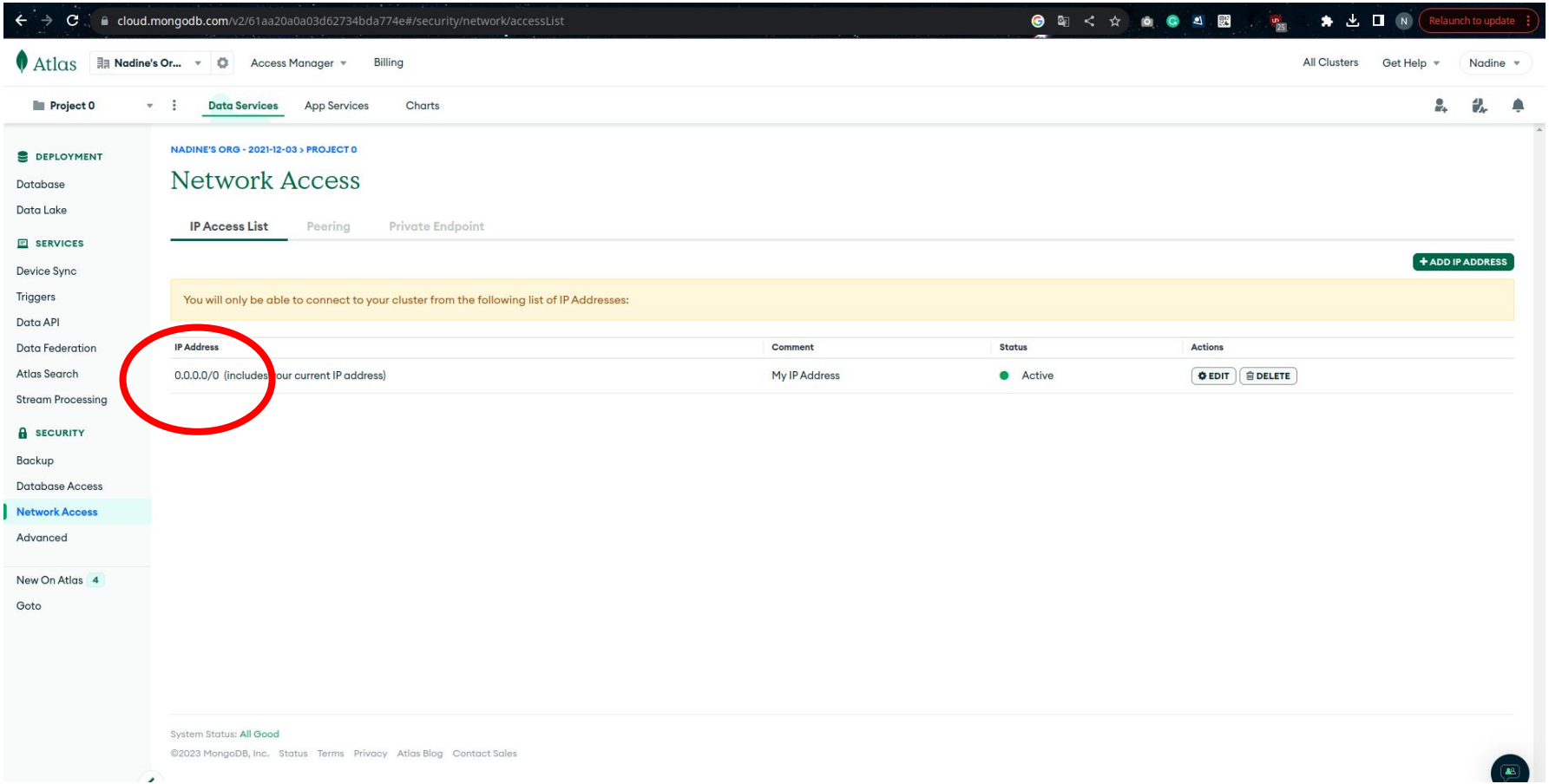
Se connecter



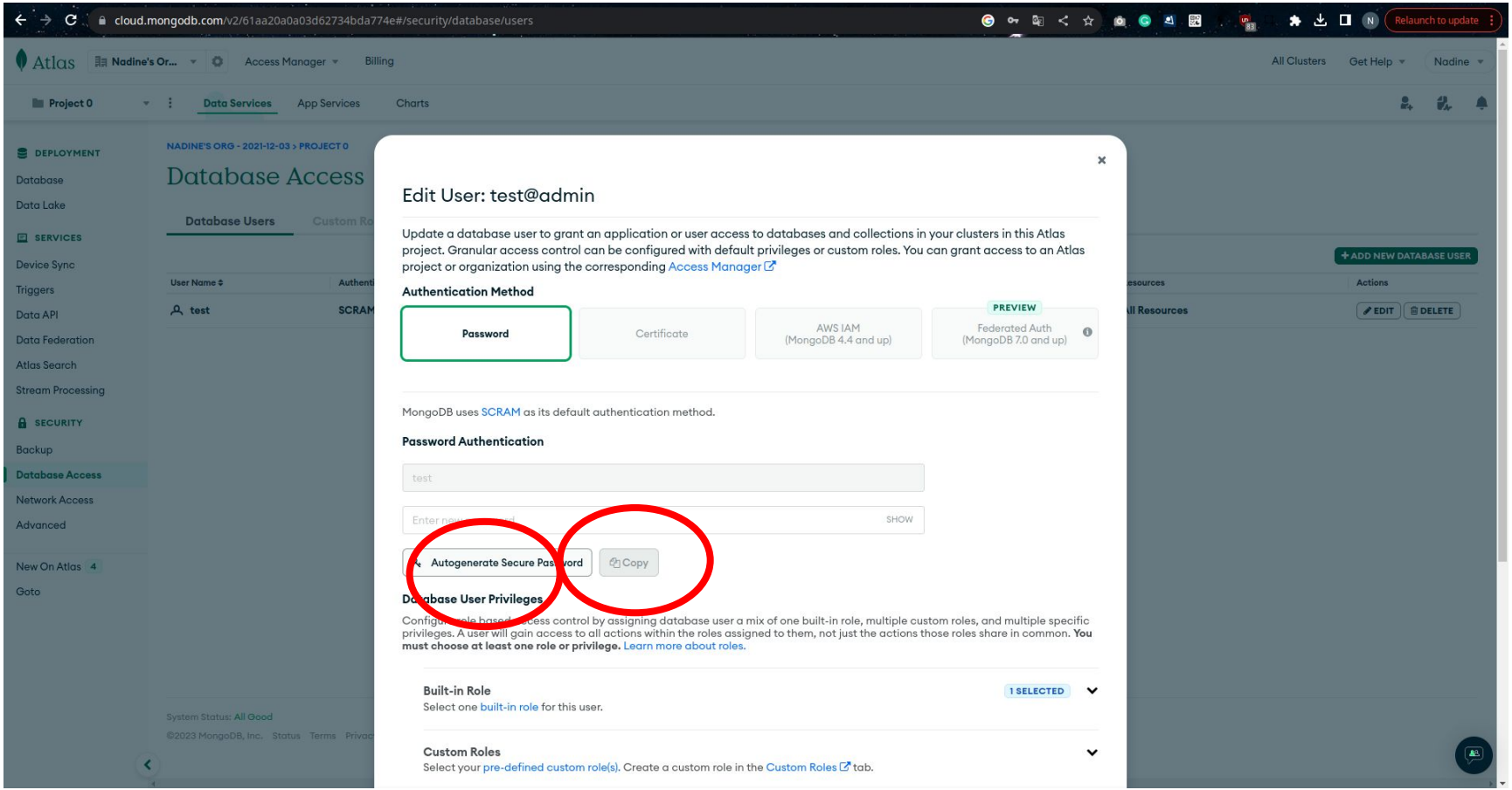
Se connecter



Se connecter



Se connecter



Se connecter

```
mongodb+srv://test:<db_password>@clusterdumusee.d0urnao.mongodb.net/?appName=ClusterDuMusee
```

```
mongodb+srv://test:testtest@clusterdumusee.d0urnao.mongodb.net/?appName=ClusterDuMusee
```

Se connecter

```
const {MongoClient} = require('mongodb');

async function main(){
    const uri = `mongodb+srv://test:testtest@cluster0.tj3ab.mongodb.net/bestiaire`;
    const client = new MongoClient(uri);

    try {
        await client.connect();
         console.log('j''ai trouve les fleurs');
    } catch (e) {
        console.error(e);
    } finally {
        await client.close();
    }
}

main().catch(console.error);
```

Programmation Mongo

En javascript

Lister les BDD

```
const {MongoClient} = require('mongodb');

async function listerBases(client){
    databasesList = await client.db().admin().listDatabases();
    console.log("Bases de donnees :");
    databasesList.databases.forEach(db => console.log(` - ${db.name}`));
};
```

```
try
{
      await client.connect();
      await listerBases(client);
}
```

Lister les objets

```
async function listerOiseaux(client){
    const oiseaux = await client.db("bestiaire").collection("oiseau").find({}).toArray();
    console.log("Oiseaux :");
    //console.log(oiseaux);
    oiseaux.forEach(o => console.log(` - ${o.nom}`));
}
```

```
try
{
      await client.connect();
      await listerOiseaux(client);
}
```

Rechercher un objet

```
async function chercherOiseauSelonNom(client, nom){
    const oiseau = await client.db("demo").collection("oiseau").findOne({ nom: nom});
    if(oiseau)
    {
        console.log(oiseau);
        return oiseau;
    }
}
```

```
try
{
      await client.connect();
      //await listerOiseaux(client);
      await chercherOiseauSelonNom(client, "Corbeau");
}
```

Ajouter un objet

```
async function ajouterOiseau(client, nouvelOiseau){
    const insertion = await client.db("demo").collection("oiseau").insertOne(nouvelOiseau);
    console.log(`Nouvel oiseau - id ${insertion.insertedId}`);
    return insertion.insertedId;
}
```

```
try
{
      await client.connect();
      //await listerOiseaux(client);
      //await chercherOiseauSelonNom(client, "Corbeau");
      let oiseauB = {"nom" : "OiseauEnPlus", "habitat":"ordinateur"};
      id = await ajouterOiseau(client, oiseauB);
}
```

Editer l'objet

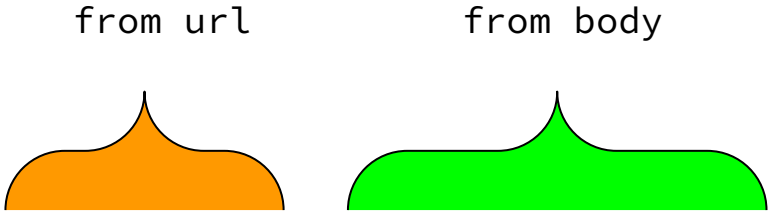
```
var ObjectId = require('mongodb').ObjectId;

exports.editerOiseau = async function (id, nouvelOiseau){
    id = ObjectId(id);
    let recherche = {_id:id};
    let changement = { $set: nouvelOiseau };
    delete nouvelOiseau._id;

    const insertion = await client.db("demo").collection("oiseau").updateOne(recherche, changement);
}
```

```
try
{
        await oiseauDAO.connecter();
        let oiseauF = {"nom" : "DroleDeOiseau", "habitat":"virtuel"};
        oiseauF._id = '6398df33bca3bb742b6dac14';
        await
oiseauDAO.editerOiseau('6398eceb8e74f28d61bd66d7',oiseauF);
}
```

UPDATEONE dans mongo




```
db.collection.updateOne(filter, object)
```

```
await db.collection("seats").updateOne({ _id : ‘2341’ },{ nom : "turlutte" });
```

```
var ObjectId = require('mongodb').ObjectId;
let id = '6398ecc971f22cad38b89671';
id = ObjectId(id);

let recherche = {_id : id};
```

```
let oiseauChange = { nom : "nouveauNom" };

let oiseauChange = oiseau;
delete oiseauChange._id;

let changement = { $set: oiseauChange };
```

```
await db("demo").collection("oiseau").updateOne(recherche,update);
```

Effacer un objet

```
var ObjectId = require('mongodb').ObjectId;

exports.effacerOiseau = async function (oiseau){
    id = ObjectId(oiseau._id);
    let recherche = {_id:id};

    const insertion = await client.db("demo").collection("oiseau").deleteOne(recherche);
}
```

```
try
{
        await oiseauDAO.connecter();
        let oiseauZ = {"nom" : "DroleDeOiseau", "habitat":"virtuel"};
        oiseauZ._id = '6398ed851b3f05a49f680924';
        await oiseauDAO.effacerOiseau(oiseauZ);
}
```

API

Pour mongo

reponse de updateOne()

https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/#returns

The method returns a document that contains:

- `matchedCount` containing the number of matched documents
- `modifiedCount` containing the number of modified documents
- `upsertedId` containing the `_id` for the upserted document.
- A boolean `acknowledged` as `true` if the operation ran with write concern or `false` if write concern was disabled

find()

- find()
  - rapporte tout
- find(criteria)
  - rapporte tout ce qui correspond au critere
  - db.collection.find({category = ‘musique’});
- find(criteria, fieldChooser)
  - rapporte seulement les champs a 1
  - db.collection.find({category = ‘musique’}, {name:1, price:1, img : 0});
  - db.collection.find({category = ‘musique’}, {name:1});
  - db.collection.find({category = ‘musique’}, {img:0});
  - db.collection.find({}, {name:1});

Find()

```
db.collection.find(filter)
```

```
await db.collection("greetings").findOne({ lang : ‘FR’ });
```

FILTER : SEARCH inside collection according to one field

lang	lang	hello
EN	english	hello
FR	french	alllo
ES	spanish	si

Findone()

```
db.collection.findOne(filter)
```

```
await db.collection("greetings").findOne({ lang : 'FR' });
```

FILTER : SEARCH inside collection according to one field

lang	lang	hello
EN	english	hello
FR	french	alllo
ES	spanish	si

Find()

```
db.collection.find(filter)
```

```
await db.collection("items").find({ category : ‘Fitness’ });
```

FILTER : SEARCH inside collection according to one field

lang	lang	hello
EN	english	hello
FR	french	alllo
ES	spanish	si

reponse de updateOne()

https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/#returns

The method returns a document that contains:

- found : matchedCount > 0
- updated : modifiedCount > 0

let **result** = await db.collection("greet").updateOne(criteria,update);

- let **found** = (result.matchedCount > 0)
- let **updated** = (result.modifiedCount > 0)

let answer = {found : **found**, updated : **updated**};

Types illustrés

Pour mongo

UPDATEONE dans mongo

```
db.collection.updateOne(filter, object)

db.collection("seats").updateOne({ _id : 'A-3' },{$set : { isBooked : true }});
```

FILTER : SEARCH inside collection according to one field (criteria)

_id	lang	hello
EN	english	hello
FR	french	alllo
ES	spanish	si

OBJECT : new data to be added or updated into the collection - called update
because only some fields.

_id	lang	hello
EN	english	hello
FR	french	salut
ES	spanish	si

FindOne dans mongo

```
db.collection.updateOne(filter)
```

```
const result = await db.collection("greetings").findOne({ _id : 'FR' });
```

FILTER : SEARCH inside collection according to one field

_id	lang	hello
EN	english	hello
FR	french	alllo
ES	spanish	si

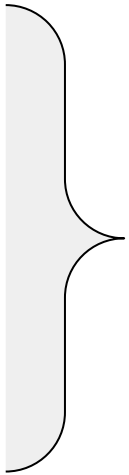
InsertOne dans mongo

```
db.collection.insertOne(object)
```

```
const result = await db.collection("birds").insertOne({ name : 'merle' });
```

_id	name	description
23fwe	raven	blabla
23dew	goeland	bla
23fff	merle	

23fff	merle	



OBJECT : new data to be added or updated into the collection

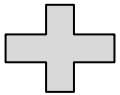
_id	name	description
23fwe	raven	blabla
23dew	goeland	bla
23fff	merle	

deleteOne dans mongo

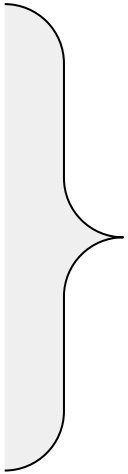
```
db.collection.deleteOne(object)
```

```
id = ObjectId(id);
const result = await db.collection("birds").deleteOne({ _id : id });
```

_id	name	description
23fwe	raven	blabla
23dew	goeland	bla



23fff	merle	



OBJECT : new data to be added or updated into the collection

_id	name	description
23fwe	raven	blabla
23dew	goeland	bla
23fff	merle	

UPDATEONE dans mongo

```
db.collection.updateOne(filter, object)
```

```
await db.collection("greet").updateOne({ _id : 'FR' },{ hello : 'salut' });
```

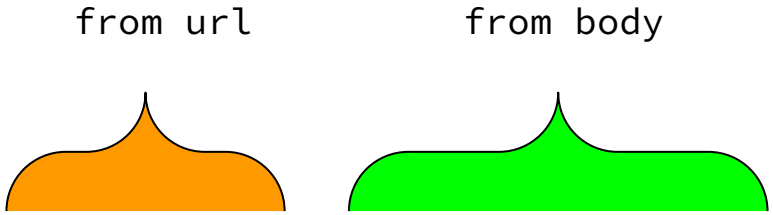
FILTER : SEARCH inside collection according to one field (criteria)

_id	lang	hello
EN	english	hello
FR	french	alllo
ES	spanish	si

OBJECT : new data to be added or updated into the collection - called update
because only some fields.

_id	lang	hello
EN	english	hello
FR	french	salut
ES	spanish	si

UPDATEONE dans mongo




```
db.collection.updateOne(filter, object)
```

```
await db.collection("seats").updateOne({ _id : 'A-3' },{ isBooked : true });
```

PATCH http://localhost:5678/api/book-seat/:id + { } in body

```
var ObjectId = require('mongodb').ObjectId;

let id = request.params.id;

id = ObjectId(id);

let criteria = {_id : id};
```

```
let data = { isBooked : true };

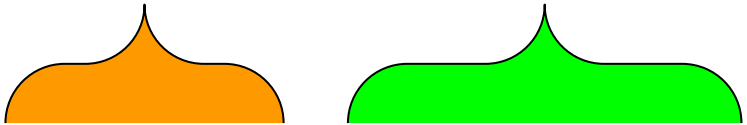
let update = { $set: data };
```

```
await db.collection("seats").updateOne(criteria,update);
```

UPDATEONE dans mongo

from url

from body



```
db.collection.updateOne(filter, object)
```

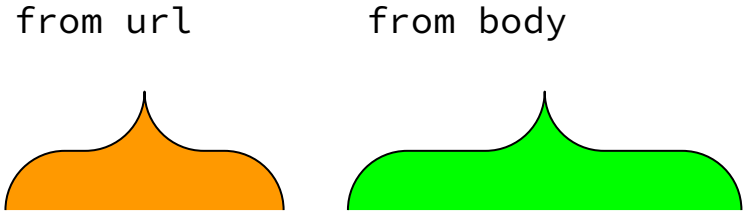
```
await db.collection("greet").updateOne({ _id : 'FR' },{ hello : 'salut' });
```

PATCH http://localhost:8000/api/greeting/:id + { hello : 'salut' } in body

```
let id = request.params.id;
```

```
let hello = request.body.hello;
```

UPDATEONE dans mongo




```
db.collection.updateOne(filter, object)
```

```
await db.collection("greet").updateOne({ _id : 'FR' },{ hello : 'salut' });
```

PATCH http://localhost:8000/api/greeting/:id + { hello : 'salut' } in body

```
let id = request.params.id;

let criteria = {_id : id};
```

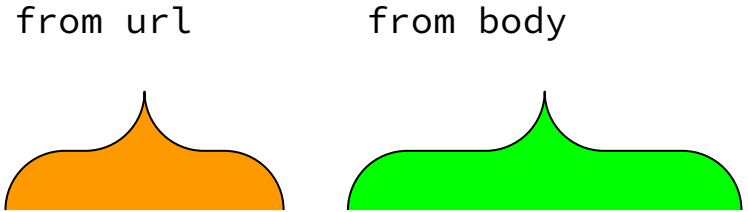
```
let newHello = request.body.hello;

let data = { hello : newHello };

let update = { $set: data };
```

```
await db.collection("greet").updateOne(criteria,update);
```

UPDATEONE dans mongo




```
db.collection.updateOne(filter, object)
```

```
await db.collection("greet").updateOne({ _id : 'FR' },{ hello : 'salut' });
```

PATCH http://localhost:8000/api/greeting/:id + { hello : 'salut' } in body

```
let id = request.params.id;

id = ObjectId(id);

let criteria = {_id : id};
```

```
let newHello = request.body.hello;

let bird = { hello : newHello };

let update = { $set: data };
```

```
await client.db("test").collection("birds").updateOne(criteria,update);
```

UPDATEONE dans mongo

```
router.post("/birds/:id/update", async(request,response) => {
    let bird = {};
    bird.color = request.body.color; // the value of the form is copied ito the object
    await bird_database.update_a_bird(bird);
});
```

```
async function update_a_bird(bird)
{
    let id = ObjectId(bird.id); // bird.id need to be populated
    let criteria = {_id : id };
    let update = { $set: bird };

    await client.db("test").collection("birds").updateOne(criteria,update);
}
```



Séparer des fichiers



require() dans quel ordre ?

Module Core

Node.js regarde si il existe un module 'core' portant ce nom. (Des includes paths à venir ?)

1

Module dans le répertoire

Node.js explore le répertoire de déploiement si un module contribué existe tel que lemodule/index.js

2

Module en fichier dans le répertoire

Nodejs vérifie si un fichier de module existe tel que lemodule.js

3



Programme avec une classe externe

Fichier main.js

```
var truc = require('./Truc');
truc.operer();
```

Fichier Truc.js

```
exports.operer = function()
{
  console.log('Allo !');
}
```

Invocation

cd .. (répété)

```
C:\>cd nodejs
C:\nodejs>node main
```

Fonction Callback

FICHIER service.js

```
var moutonDAO = require('./MoutonDAO');
var http = require('http');

var repondre = async function(requete, reponse)
{
        var listeMoutons = await moutonDAO.listerMoutons();
        reponse.end(JSON.stringify(listeMoutons));
}

var serveur = http.createServer(repondre);
serveur.listen(8080);
```

Séparer la classe DAO

FICHIER service.js

```
var moutonDAO = require('./MoutonDAO');
var http = require('http');
var serveur = http.createServer(
        // dans evenement de requete http
        function(requete, reponse)
        {
                moutonDAO.listerMoutons().then(console.log);
                reponse.end('Donnees listées');
        }
);
serveur.listen(8080);
```

FICHIER MoutonDAO.js

```
var postgresql = require('pg');
var chaineDeConnection = 'postgres://postgres:test@localhost:5432/bergerie';
var SQL_LISTE_MOUTON = 'SELECT * FROM mouton';
exports.listerMoutons = async function()
{
        var basededonnees = new postgresql.Client(chaineDeConnection);
        await basededonnees.connect();
        //console.log('base de donnees ' + basededonnees._connected);
        var curseurListeMouton = await basededonnees.query(SQL_LISTE_MOUTON);
        //console.log(curseurListeMouton);
        var listeMoutons = {}; var position = 0;
        curseurListeMouton.rows.forEach
        (
                mouton=>
                {
                        listeMoutons[position] = row; position++;
                }
        );
        console.log(JSON.stringify(listeMoutons));
}
```

Retourner la liste

FICHIER service.js

```
var moutonDAO = require('./MoutonDAO');
var http = require('http');
var serveur = http.createServer(
    // dans evenement de requete http
    function(requete, reponse)
    {
        var listeMoutons =
moutonDAO.listerMoutons().then(console.log);
        for(position in listeMoutons)
        {
            console.log(JSON.stringify(listeMoutons[position]
));
        }
        reponse.end('Donnees listées');
    }
);
serveur.listen(8080);
```

FICHIER MoutonDAO.js

```
var postgresql = require('pg');
var chaineDeConnection = 'postgres://postgres:test@localhost:5432/bergerie';
var SQL_LISTE_MOUTON = 'SELECT * FROM mouton';
exports.listerMoutons = async function()
{
    var basededonnees = new postgresql.Client(chaineDeConnection);
    await basededonnees.connect();
    //console.log('base de donnees ' + basededonnees._connected);
    var curseurListeMouton = await basededonnees.query(SQL_LISTE_MOUTON);
    var listeMoutons = {}; var position = 0;
    curseurListeMouton.rows.forEach
    (
        mouton=>
        {
            console.log("DAO:" + mouton.nom + " (" + mouton.couleur
                + "," + mouton.naissance + ")");
            listeMoutons[position] = row; position++;
        }
    );
    return listeMoutons;
}
```

Module technique A

Fichier OiseauDAO.js

```
exports.listerOiseau = function(){}
exports.chercherOiseauSelonId = function(id){}
exports.ajouterOiseau = function(oiseau){}
exports.effacerOiseau = function(oiseau){}
exports.editerOiseau = function(oiseau){}
```

Fichier App.js

```
const oiseauDAO = require('./OiseauDAO');
await oiseauDAO.listerOiseau( );
```

Module technique B

Fichier OiseauDAO.js

```
const listerOiseau = function(){}
const chercherOiseauSelonId = function(id){}
const ajouterOiseau = function(oiseau){}
const effacerOiseau = function(oiseau){}
const editerOiseau = function(oiseau){}

module.exports = { listerOiseau, chercherOiseauSelonId, ajouterOiseau, effacerOiseau, editerOiseau
};
```

Fichier App.js

```
const oiseauDAO = require('./OiseauDAO');
```

Types illustrés

Pour mongo

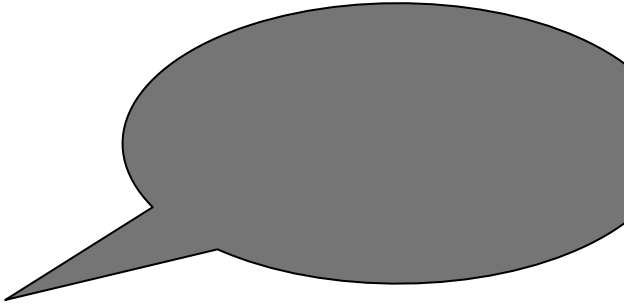
Template

Le diaporama est fini

Titre

Titre

bla



Lister les collections

```
async function listerCollections(client){
    collectionsList = await client.db("demo").listCollections({}, { nameOnly: true }).toArray();
    console.log("Collections :");
    collectionsList.databases.forEach(collection => console.log(` - ${collection.name}`));
};
```

Erreur de compatibilite

```
n@dust:~/bestiaire$ node mongo.js
/home/n/bestiaire/node_modules/mongodb/lib/admin.js:62
           session: options?.session,
                            ^
SyntaxError: Unexpected token '.'

    at wrapSafe (internal/modules/cjs/loader.js:915:16)
    at Module._compile (internal/modules/cjs/loader.js:963:27)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1027:10)
    at Module.load (internal/modules/cjs/loader.js:863:32)
    at Function.Module._load (internal/modules/cjs/loader.js:708:14)
    at Module.require (internal/modules/cjs/loader.js:887:19)
    at require (internal/modules/cjs/helpers.js:74:18)
    at Object.<anonymous> (/home/n/bestiaire/node_modules/ mongodb/lib/index.js:6:17)
    at Module._compile (internal/modules/cjs/loader.js:999:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1027:10)
```

Erreur de compatibilite

```
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -

sudo apt install -y nodejs

node -v

==================================OU=======================================

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash

Relancer Recharger le shell
nvm install 15
nvm use 15
```

Node dans Ubuntu

```
apt-get install nodejs
apt-get install npm

# upgrade a 15
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
Relancer Recharger le shell
nvm install 15
nvm use 15
```