

Addressing the Issues for Deploying Instructor Demo 03

Edward Apostol

January 24th, 2024

Deploying the MERN Site

Finalizing Local Changes and Locally Test the App

1. Ensure that the application was running without issue locally on your laptop (this is without adding automated testing, only manually testing at this stage - automation will be added next)
2. Review your root directory's package.json script settings, so that you will know which script commands to setup when you create a "web service":

The current package.json script block looks like this:

```
○   "scripts": {  
      "start": "npm run client:build && npm run server",  
      "start:dev": "concurrently \"npm run server:dev\" \"wait-on  
tcp:3001 && npm run client:dev\"",  
      "server": "cd server && npm start",  
      "server:dev": "cd server && npm run dev",  
      "server:build": "cd server && npm run build",  
      "install": "cd server && npm i && cd ../client && npm i",  
      "client:build": "cd client && npm run build",  
      "client:dev": "cd client && npm run dev",  
      "build": "npm run client:build && npm run server:build",  
      "seed": "cd server && npm run seed",  
      "render-build": "npm install && npm run build",  
      "cypress": "npx cypress open",
```

```

    "test": "npx cypress run"
  }

```

It is likely you will be performing the following commands to set up your site on your hosting provider's dashboard:

- `npm run install`
to execute the install command from the script block
`npm run render-build`
runs the npm install (root folder) then the build: script
`npm run seed`
don't forget to seed the db (will fail 1st time
#until mongoDB is set up on Atlas - your mongoDB Cloud provider)

Setting Up DB Hosting

4. Did you: set up your MongoDB on Atlas? This includes

1. Creating an account at <https://www.mongodb.com/> and logging in.
2. Creating a *cluster*. You get one free cluster on Atlas. This includes

. Who is the Cloud Provider and where is the Provider (Region?) i.e. Amazon Web Services in North Virginia

Cloud Provider & Region

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



Upgrade to change your Cloud Provider or Region

N. Virginia (us-east-1) ★

. *Cluster capacity* - Your server hosting your DB needs RAM and CPU. Choose the free option until you or your client are ready to pay for more.

Cluster Capacity

0 - 100 Ops/Sec, 512 MB Storage ^

Ops/Sec	Storage	RAM, vCPU	Connections, Databases, Collections	Price
0 - 100	512 MB	Shared	Up to 500 each	Free

. Additional settings include termination protection (enabled in the free tier by default) and optional meta tags should you wish to search or log some info about your DB cluster.

3. After creating your cluster, make sure you get the connection string that is required to connect to your MongoDB Server on Atlas. **Note you will need to get or set the password for the default user in the next step to fill in the part**

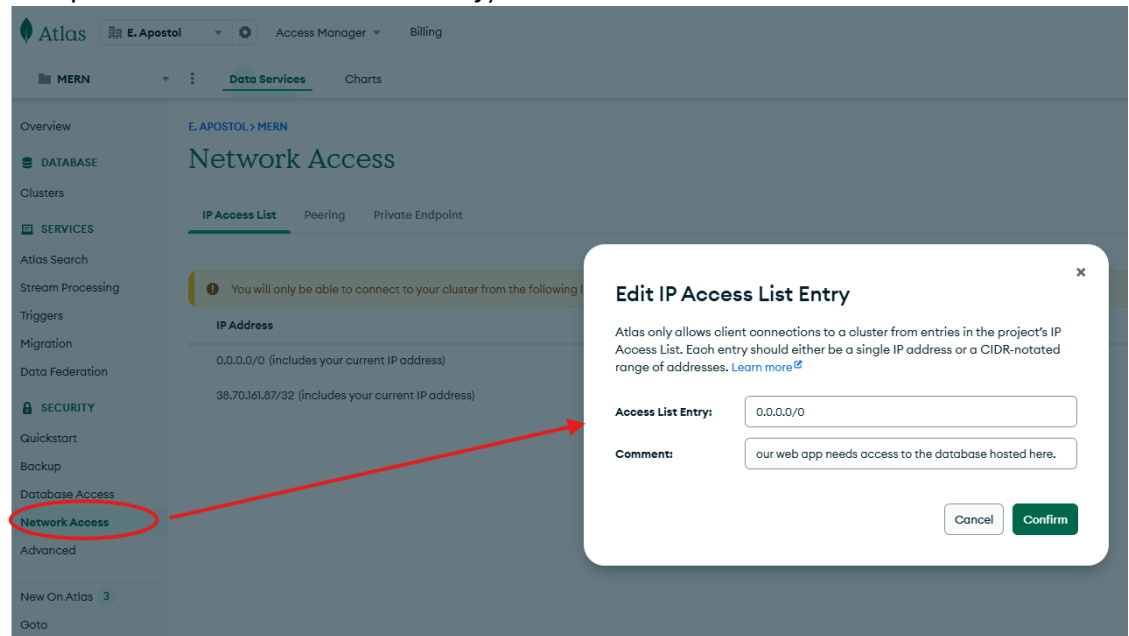
The image shows two panels from the MongoDB Atlas interface. The left panel, titled 'Connection Configuration', has a 'Select database user' dropdown set to 'dbadmin (SCRAM)'. Below it, 'Select connection method' shows four options: 'Drivers' (highlighted with a green box), 'Compass', 'Shell', and 'MongoDB for VS Code'. The right panel, titled 'Connection Instructions', shows a '1. Select driver' dropdown set to 'Node.js' (circled in red). Below it, '2. Install your driver' shows a terminal command: `npm install mongodb`. '3. Add connection string into your application code.' shows a 'String' tab with a connection string: `mongodb+srv://dbadmin:<db_password>@merncluster.y581z.mongodb.net/?retryWrites=true&w=majority&appName=merncluster`. A red arrow points from the 'String' tab to the connection string.

4. On the Atlas Dashboard, select Database Access and setup the password for the default user, DBADMIN. *Next, you will ensure your database is accessible for your Render hosted site.*

The image shows the MongoDB Atlas 'Database Access' configuration page for the 'dbadmin' user. The 'Authentication Method' is set to 'Password' (highlighted with a green box). The 'Password Authentication' section shows the username 'dbadmin' and a field for the password, with an 'Autogenerate Secure Password' button and a 'Copy' button. The 'User Description' field is set to 'default user'. A red arrow points from the 'Database Access' link in the left sidebar to the 'Password' authentication method.

5. On the Atlas Dashboard, select Network Access and ensure that you add the setting to make your DB accessible publically (instead of your own

computer's default IP address only)



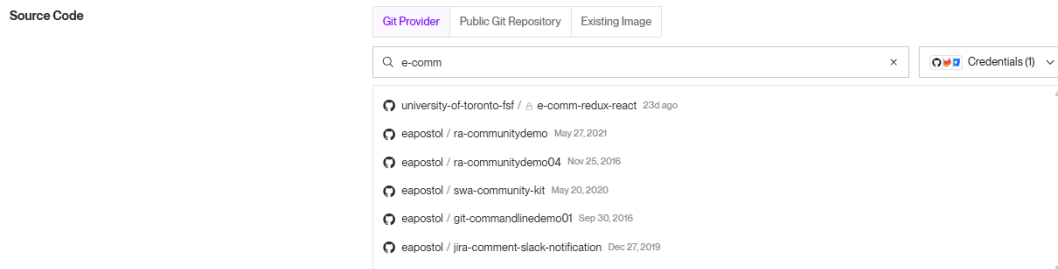
Now that the DB is setup, you will go to Render and proceed with creating your web service / site.

Setting Up Render Deployment

5. If you had not done so, **create an account on Render** (<https://www.render.com>) and login.
6. Did you create a **web service** with the following settings?
 - **Git Provider** Choose the repository where you are deploying from - You may also use *Public Git Repository* if your repository is hosted outside of Github (e.g. Gitlab or Bitbucket by Atlassian)

You are deploying a Web Service

You seem to be using **Node**, so we've autofilled some fields accordingly. Make sure the values look right to you!



- **Name-** provide a unique name
- **Region-** Choose a region close to where your users may access the site (e.g)

- **Environment**- Choose *Node* or the appropriate language / environment you are using
- **Build Command** - referring to step (3) in the **Finalizing Local Changes Section**

```
npm run install; npm run render-build; npm run seed
```
- **Start Command**: For now, `npm run start:dev` will suffice. Note even though we are using port 3001 to connect our client to our server for our requests, Render will work with it. See [this link](#) for additional details.

Port binding

Every Render web service must bind to a port on host **0.0.0.0** to serve HTTP requests. Render forwards inbound requests to your web service at this port (it is not *directly* reachable via the public internet).

We recommend binding your HTTP server to the port defined by the **PORT** environment variable. Here's a basic Express example:

```
const express = require('express')
const app = express()
const port = process.env.PORT || 4000;

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Adapted ever-so-slightly from [here](#)

The default value of **PORT** is **10000** for all Render web services. You can override this value by [setting the environment variable](#) for your service in the [Render Dashboard](#).

i If you bind your HTTP server to a different port, Render is *usually* able to detect and use it.
 If Render fails to detect a bound port, your web service's deploy fails and displays an error in your [logs](#).

- **Instance Type** : Choose Free while testing, Starter or higher if you want the site to load immediately and need more memory / CPU for your app.
7. **Environment Variables** -You will need to set up your environment variables which were likely from a `.env` file in your server folder in the current monorepo-based project. Don't forget to (a) reference the `MONGODB_URI` variable to store the connection string value that retrieved when setting up your connection in Atlas:

MONGODB_URI	👁
-------------	-------	---

The value again should look something like this:

```
mongodb+srv://dbadmin:<password>@merncluster.y58iz.mongodb.net/?retryWrites=true&w=majority&appName=merncluster
```

Where password was obtained from the Atlas dashboard as well

Since your client side react application is running on top of VITE, you need to prefix your variables with VITE_.

Key	Value	
VITE_OMDB_API_KEY	👁
VITE_OMDB_API_URL	👁

Notes: If your react front-end also has environmental variables stored in a .env file in the /client directory as well as a ./server directory (a *monorepo* setup), then you will have to ensure that (a) your files in the ./client and ./server folders point to the correct .env file when testing locally- use the **dotenv** package to set that up. OR use a *single*.env file in the project root directory (outside ./client and ./server) and ensure again your files and folders in the respective ./client and ./server directories properly reference the single .env file with the **dotenv** package.

8. Review and save your settings.

Deploy, Monitor and Fix

9. Monitor the event log to see if there are any errors, and troubleshoot / fix where appropriate. Once the service is set up, you can click on the URL link to view the app. Note when clicking on the link, when using the free tier, it could take about a minute or more for the server to spin up and start the site.

03-MERN-Sample-Deploy

Events

Settings

MONITOR

Logs

Metrics

MANAGE

Environment

Shell

Scaling

Previews

Disks

Jobs

Changelog

Invite a friend

Contact support

Render Status

03-MERN-Sample-Deploy

Node Free Upgrade your instance →

eaapostol / 03-ins_deploy_fsf · main
<https://zero3-mern-sample-deploy.onrender.com>

Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. [Upgrade now](#)

March 18, 2025 at 12:09 PM Live

[66bb1f2](#) fix: downgrade to Node 20 in order to prevent assert error with seeding

All logs Search Mar 18, 12:08 PM - 12:13 PM EDT

```
Mar 18 12:11:30 PM [0]
Mar 18 12:11:30 PM [0] > mern-server@1.0.0 dev
Mar 18 12:11:30 PM [0] > npm run dev
Mar 18 12:11:30 PM [0]
Mar 18 12:11:30 PM [0] [nodemon] 3.1.9
Mar 18 12:11:30 PM [0] [nodemon] to restart at any time, enter 'rs'
Mar 18 12:11:30 PM [0] [nodemon] watching path(s): src/**/*
Mar 18 12:11:30 PM [0] [nodemon] watching extensions: ts,json,js
Mar 18 12:11:30 PM [0] [nodemon] starting 'npm run dev'
Mar 18 12:11:30 PM [0] --> No open ports detected, continuing to scan...
```

<https://zero3-mern-sample-deploy.onrender.com/> for example.

zero3-mern-sample-deploy.onrender.com

Word Guess

Guess the movie title!

Guesses Remaining: 9

A	B	C	D	E	F	G	H	I
J	K	L	M	N	O	P	Q	R
S	T	U	V	W	X	Y	Z	