Addressing the Issues for Deploying Instructor Demo 03

Edward Apostol

January 24th, 2024

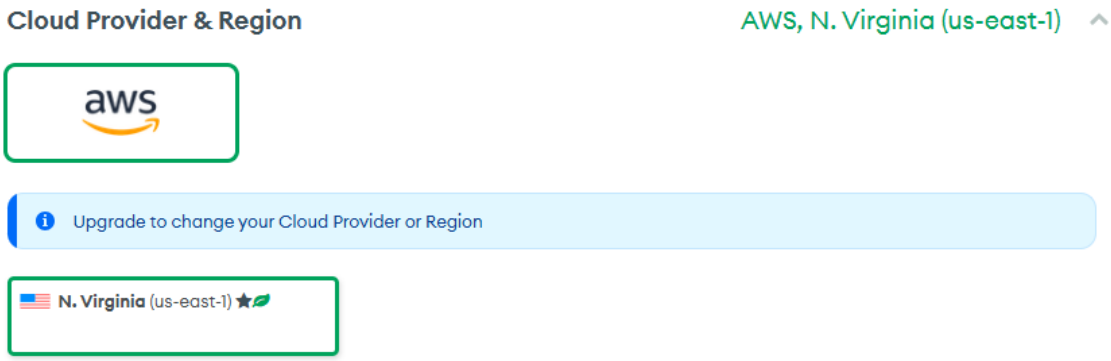
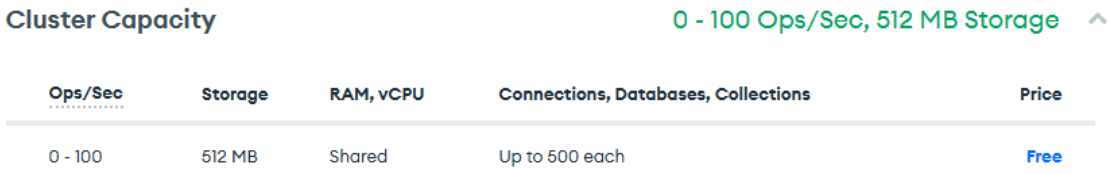
Deploying the MERN Site

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)

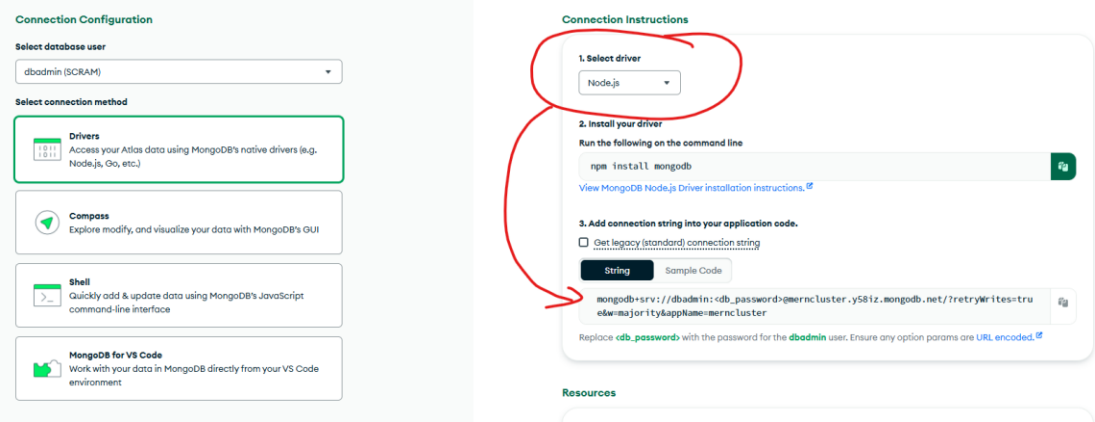
2. Review your root directory's package.json script settings, so that you will know which script commands to setup when you create a "web service":

   The current package.json `script` block looks like this:

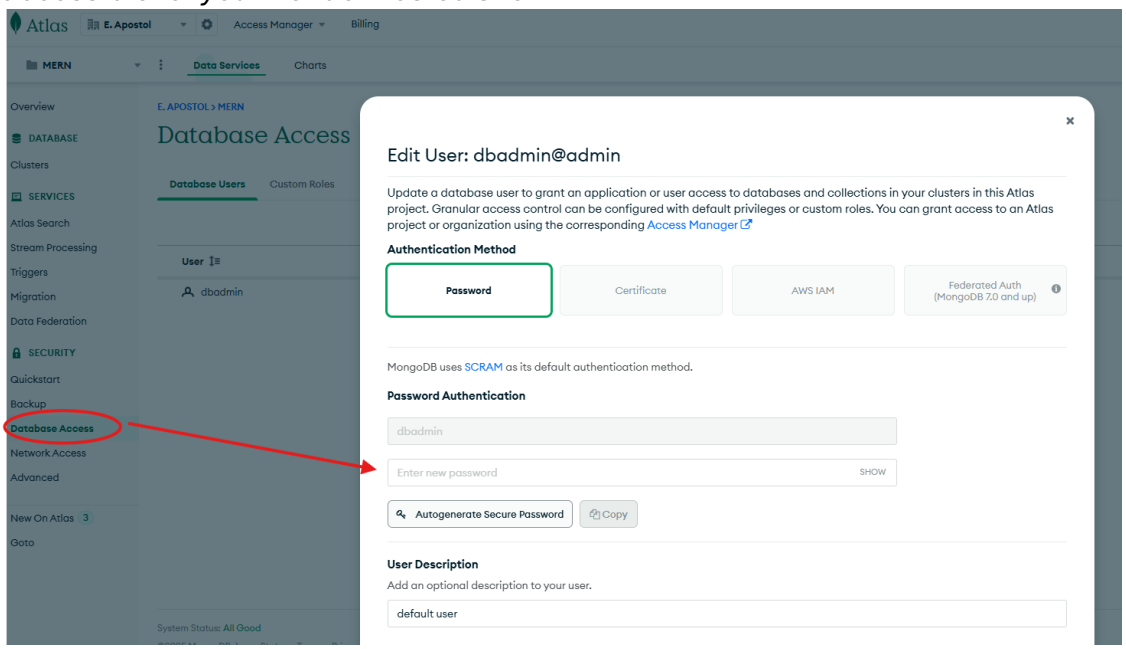
   - ```
     "scripts": {

       "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



    . *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.


  ```

. 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 theconnection 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**

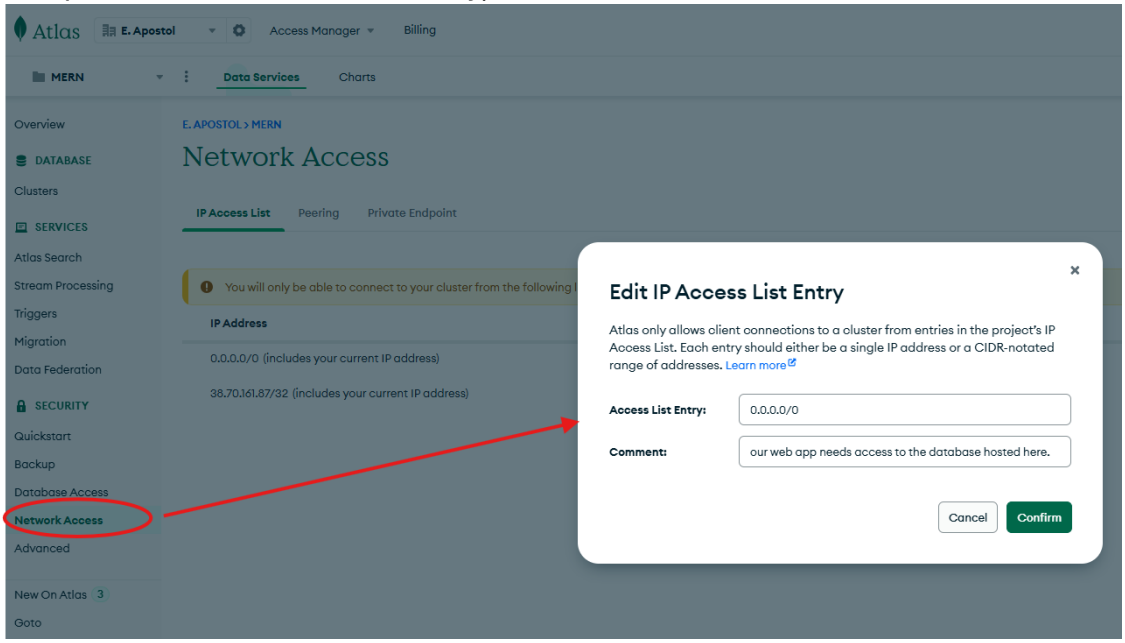


4. On the Atlas Dashboard, select `Database Access` and setup the password for the default user, DBADMIN. *Next, you will ensure your database is accessible for your Render hosted site.*



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)



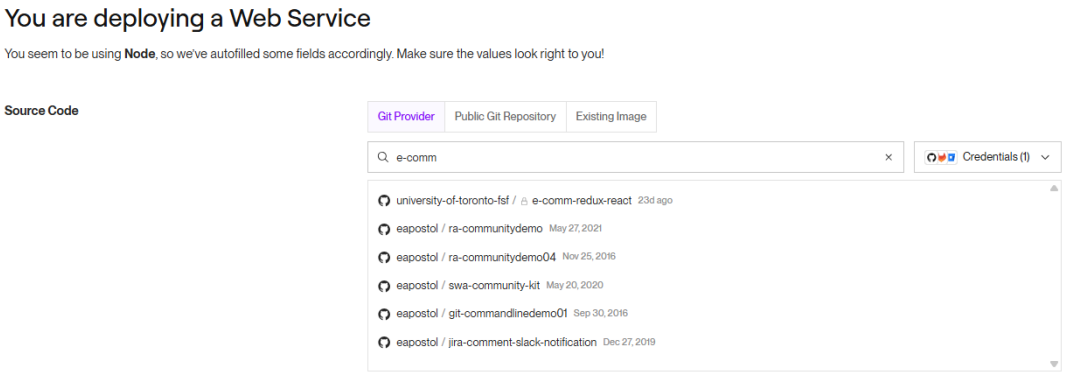
Now that the DB is setup, you will go to Render and proceed with creating your web service / site.

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)

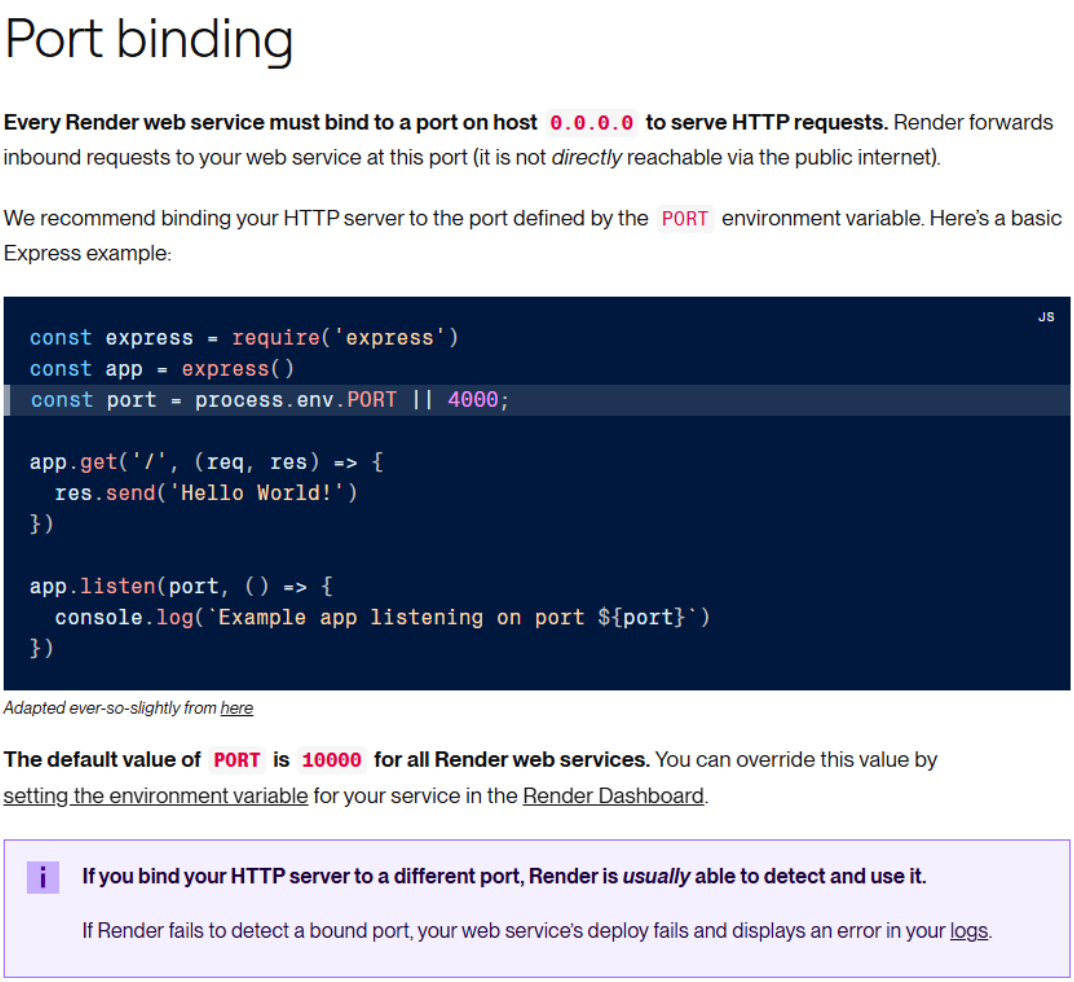


- **Name**- provide a unique name

- **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:

JS

```
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

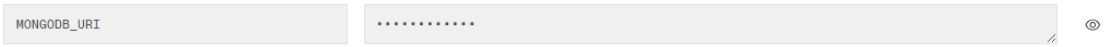
The default value of `PORT` is `10000` for all Render web services. You can override this value by setting the environment variable for your service in the Render Dashboard.

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:

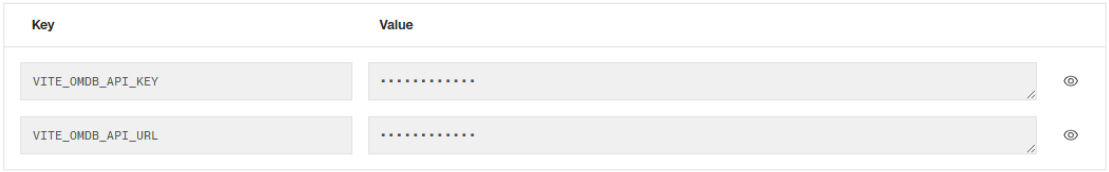


The value again should look something like this:

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_.

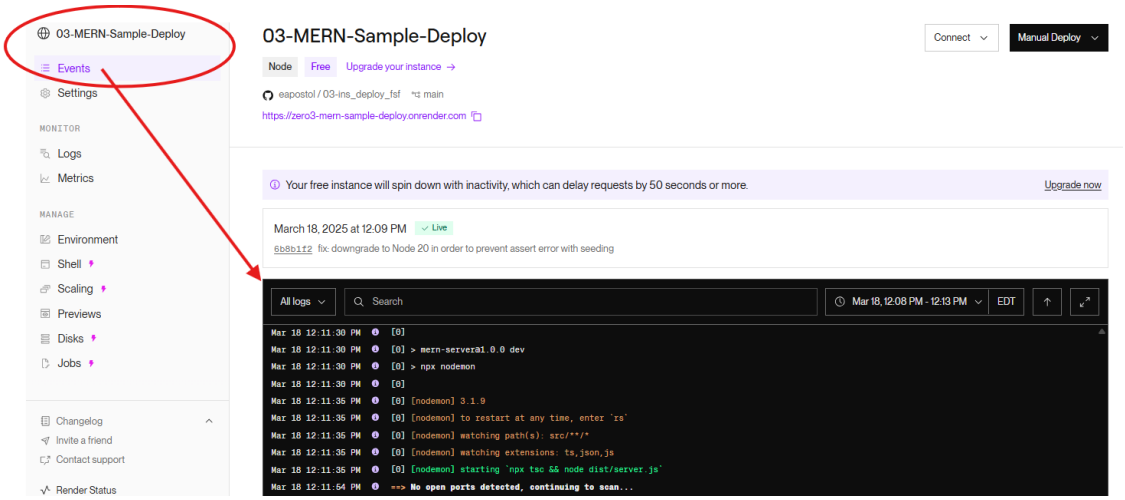


Notes: If your react front-end also has environmental variables stored in a .env file in the /client directory as well as a ./server directory (a *monorepo* setup), then you will have to ensure that (a) your files in the ./client and ./server folders point to the correct .env file when testing locally- 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.



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