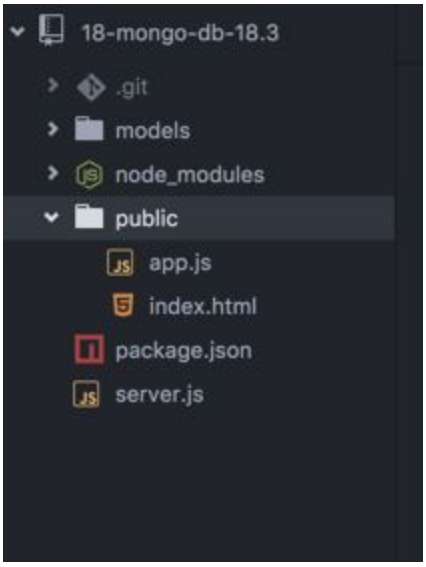
MongoDB mLab Heroku Deployment

1. Folder structure

a. For these instructions, we will be working with the folder structure as seen in **step b.** below.

b. 

2. Set up mLab

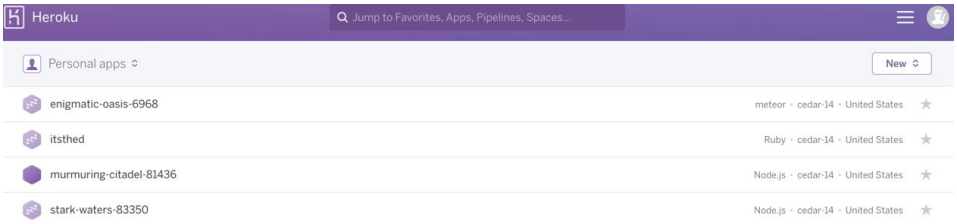
a. Starting at this point, you should have created your Heroku app and should be remote linked on your local repo on your machine. To check that you have your Heroku remotely linked, type in the command ***git remote -v*** in your local repo and press **Enter**. You should be able see your remote links as seen in the image **below**:

   i. 

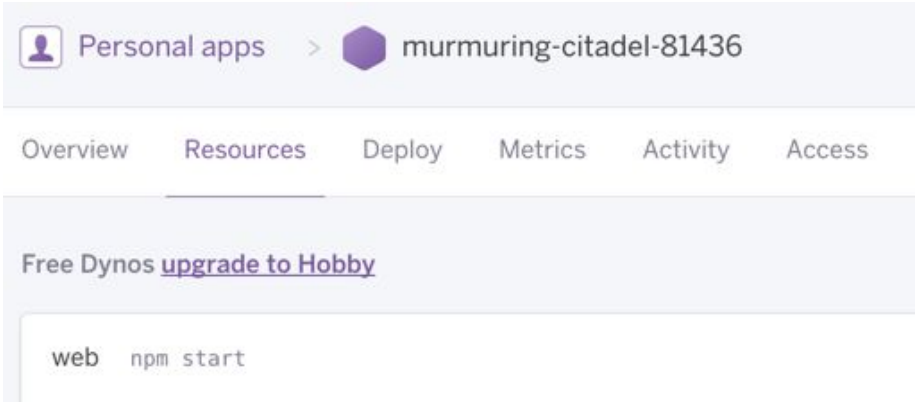
b. Login to your Heroku

c. Locate your app on your dashboard and click on it.

i. Dashboard:

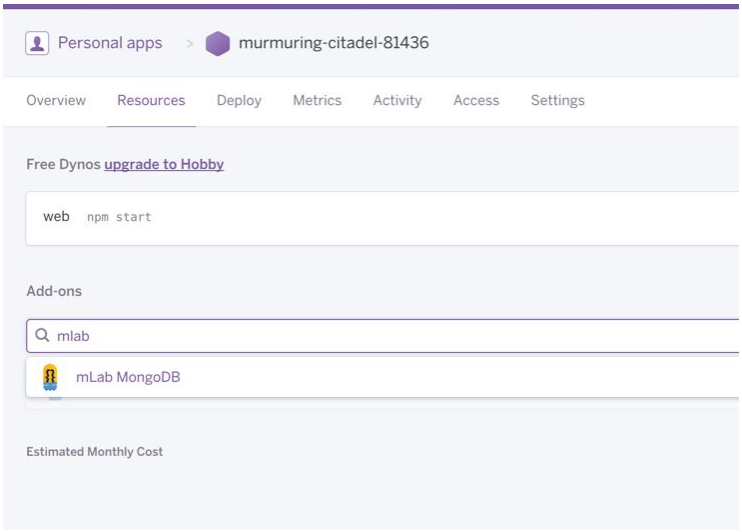


ii. After clicking your app, click on the **Resources** tab:



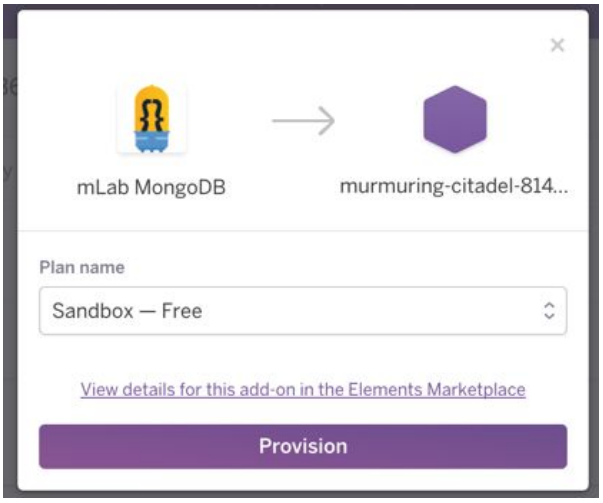
iii.

d. In the **Resources** tab, navigate to the **Add-ons** section:

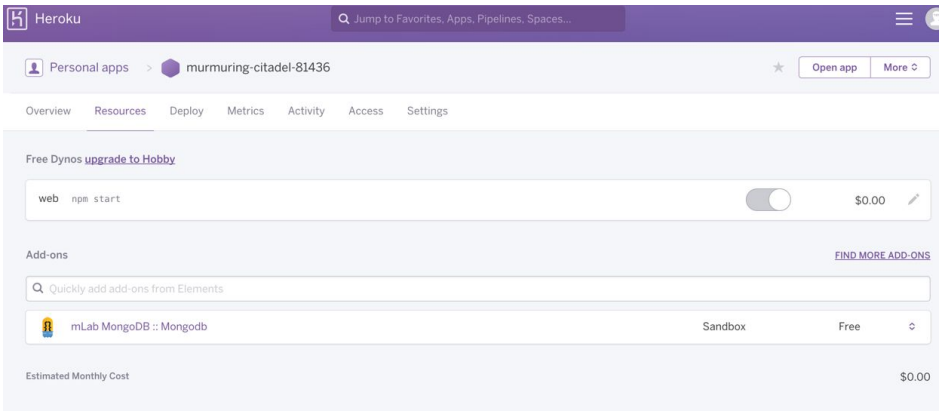


i.

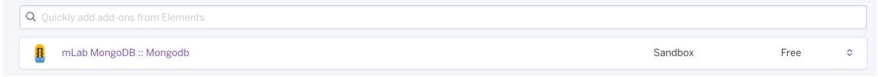
ii. Type in **mlab** in the input field and select the option **mLab MongoDB** as seen in the image above. This will bring a modal up as seen in the image below:

iii. 

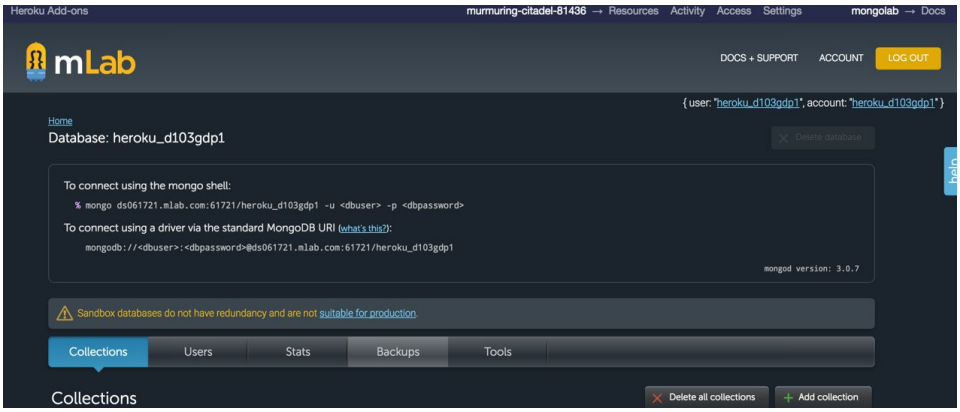
iv. Click on the **Provision** button. This will add the mLab MongoDB service to your app. To make sure that mLab has been added to your app, you should be able the service in the **Add-ons** section as seen in the image below:

v. 

vi. To navigate to your mLab add-on specifically for your Heroku app, click on the mLab option in the **Add-ons** section as seen in the image below:

vii. 

viii. This will bring you to your mLab service screen as seen in the image below:

ix. 

x. At this point, we **should not have** any collections yet, so there should be no collections in the **Collections** section as seen in the image above (**step ix)**.

3. Configure your code

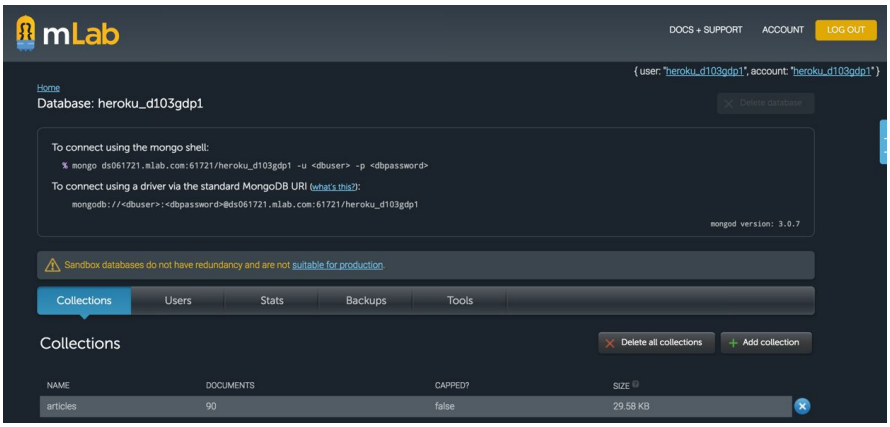
a. Navigate back to your Sublime text editor and go to your **server.js** file (or whichever file you use to start your server).

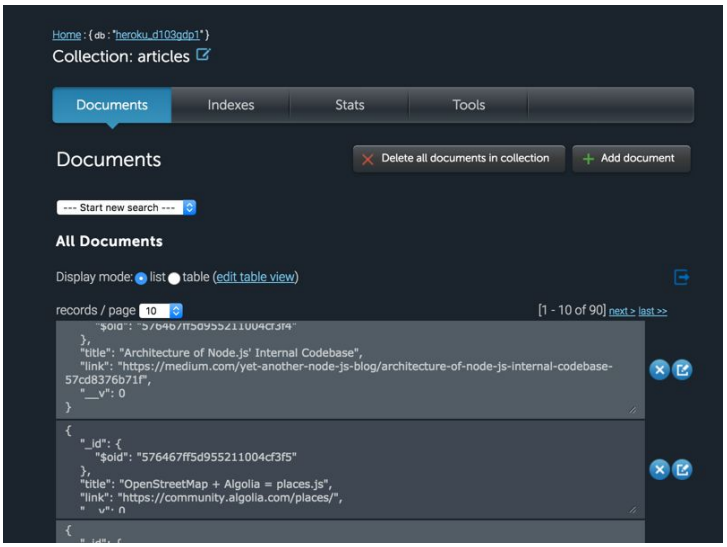
b. Using the **mongoose** ODM package, modify your code to connect to your mLab Heroku service like in the image below, taking important note of the **process.env.MONGODB_URI** property:

c.
```
var mongoose = require('mongoose');
// Notice: Our scraping tools are prepared, too
var request = require('request');
var cheerio = require('cheerio');

// use morgan and bodyparser with our app
app.use(logger('dev'));
app.use(bodyParser.urlencoded({
  extended: false
}));

// make public a static dir
app.use(express.static('public'));

// ----------Database configuration with Mongoose----------
// ---------Define local MongoDB URI----------
var databaseUri = 'mongodb://localhost/week18day3mongoose';
// ------------------------------------------
if (process.env.MONGODB_URI) {
  // THIS EXECUTES IF THIS IS BEING EXECUTED IN YOUR HEROKU APP
  mongoose.connect(process.env.MONGODB_URI);
} else {
  // THIS EXECUTES IF THIS IS BEING EXECUTED ON YOUR LOCAL MACHINE
  mongoose.connect(databaseUri);
}
// ----------End database configuration----------

var db = mongoose.connection;

// show any mongoose errors
db.on('error', function(err) {
  console.log('Mongoose Error: ', err);
});

// once logged in to the db through mongoose, log a success message
db.once('open', function() {
  console.log('Mongoose connection successful.');
});
```

d. This code has been written so that it'll connect to either your heroku mLab MongoDB service or your local machine's mongodb database, whichever is detected.

e. After configuring your code, first verify that your app connects to your local mongodb database by running the **server.js** (or equivalent JS file) in your command terminal / git bash. The command would be **node server.js OR nodemon server.js (if you have the nodemon package globally installed)**.

f. After verifying your local mongodb connection works, make the necessary commits and push your code up to your Github repo and specifically your **MASTER** branch.
g. After you've successfully pushed your code to your **MASTER** branch, deploy your **MASTER** branch to Heroku using the **git push heroku master** command.
h. After you've successfully deployed, check your heroku app take make sure you can view it in the browser. You can quickly do this from the command line by typing in **heroku open** and pressing the **Enter** button. This will open your app in your Google Chrome browser.
i. Make sure your heroku app is working by testing the functionality (i.e. creating a new user). Test as much as you want to keep adding documents to your collection. If there are no errors in your tests, navigate back to your mLab service in your browser:
j. 

k. To verify that you've successfully connected to your mLab service, check the **Collections** as seen in the image above.
    i. You should see the name if your collection along with the number of documents in the collection.
    ii. Click on the collection and you should see all your documents:
    iii. 

iv. Voila! You've connected to your mLab mongodb service for your Heroku app. Awesome job!