Web Development

Setting up the Development Environment

Description

This assignment describes setting up the development environment used throughout the semester. The development environment consists of a local Node.js HTTP server and a MongoDB database server where local development will be performed. You will commit and push your code to a remote Git code repositories. Configure the repository to auto deploy into a remote Node.js instance hosted at Heroku that will make your Web application publicly available on the Web. Invite all the instructors and TAs as collaborators with read/write permission. Instructors and TAs will visit the hosted site periodically to see your assignments and project running on the Web. The GitHub repository will make your source code available to the TAs and the instructor so they can see how you implemented the Web application. In this assignment you will setup:

1. GitHub
2. Node.js
3. MongoDB (community edition)
4. Heroku
5. Angular CLI
6. WebStorm
7. nodemon

Complete the requirements shown below to successfully complete this assignment.

Setting Up the Local Development Environment

Setting Up Node.js Locally

Node.js will be used to run and debug Web applications on local machines. When satisfied with local development, the code will be committed and push to Heroku and GitHub. The code will be deployed to Heroku and executed on a remote Node.js server. Download and install Node.js on the local development machine. Test that Node.js installed by typing the command **`node`** at the command line. A command prompt should appear waiting for additional commands. Type Ctrl+c twice to exit Node.js command line interpreter.
Node JS Download: https://nodejs.org/en/download/

Setting Up MongoDB Locally

MongoDB will be used to provide a local database to develop and debug database development. In a later assignment you will create a remote MongoDB instance on OpenShift where your data will be stored remotely. Your remote Node.js server will talk to your remote MongoDB instance. Your local Node.js server will talk to your local MongoDB instance. When you are content with your local development, you will push your code to Heroku for remote deployment, and to GitHub for us to grade. For now you will just download and install MongoDB locally. Later we will show you how to configure and integrate MongoDB with your Node.js Web application.

1. Download MongoDB and install on the local development machine as described in the MongoDB documentation. These instructions assume MongoDB was installed in **`/usr/local/mongo`** on a Mac

2. At the root of the file system, create a directory called **data** and in there create a directory called **db**, e.g., **/data/db**. This directory is where MongoDB will store all files related to the databases managed by MongoDB

   ```
   $ mkdir /data
   $ cd /data
   $ mkdir db
   ```

3. Add the MongoDB binaries to the system PATH environment variable so that the database can be executed from anywhere in the terminal command line.

   a. On macOS, create a file called .bash_profile in your home directory

      ```
      $ touch ~/.bash_profile
      ```

      i. Assuming MongoDB was installed in **/usr/local/mongo**, add the **bin** directory to the system path by adding the following line to the **~/.bash_property** file

         ```
         export PATH=/usr/local/mongo/bin:$PATH
         ```

      ii. For changes to take effect, either source the file or just restart your computer

         ```
         $ source ~/.bash_profile
         ```

   b. On Windows, add the path to MongoDB's bin directory to PATH environment variable

4. Test that MongoDB starts by typing **mongod** at the command line. The command line will print several lines. Some of the lines should say something similar to

   a. On Windows, start the mongo database by typing **mongod** at the command line
   b. On macOS, start the mongo database by typing **mongod** at the command line. You might need to start the database with root access, so you might need to use the **sudo** command

      ```
      $ sudo mongod
      ```

   c. The server will start and print the port it's listening to and the data directory it is using

      ```
      MongoDB starting : pid=56659 port=27017 dbpath=/data/db
      ...
      waiting for connections on port 27017
      ```

5. Make sure MongoDB is running before you start Node.js.

Setting Up Git Locally

Git is a distributed source control system that allows developers collaborate and share source code. All assignments and the final project will be submitted using git. Install and configure git on your machine.

1. On a windows machine, download, install, and configure a github client from any of the following sites. Be sure to install the command tools as well
    a. https://git-scm.com/download/win (preferred)
    b. https://desktop.github.com
    c. https://github.com/blog/1127-github-for-windows
2. On macOS git should already be installed. Verify that git is installed on your machine by typing git at a terminal window. You might be prompted to install additional Xcode command line components. Accept and follow installation instructions.

Setup Angular CLI

The Angular CLI is a tool to initialize, develop, scaffold and maintain Angular applications. Setting up angular CLI is simple. From a command line terminal use the following command to install Angular CLI tool globally

```
$ npm install -g @angular/cli
```

Setting up nodemon

We will be using Node.js to execute JavaScript from the command line. In particular we will be developing servers that listen for incoming HTTP requests and programmatically generate HTTP responses. When developing server side applications, you will need to restart Node.js every time you make a change to source files. To simplify server side development we will automate Node.js restarting with nodemon. Nodemon is a tool that wraps Node.js to watch for changes in your source file and automatically restart node when files change. Use the following command to install nodemon globally

```
$ npm install -g nodemon
```

Setting Up the Template Project Locally

A template project has been created that will be used as the starting point for all development in this course. Using git, clone the template to your local development environment. All development for the assignments and final project will be done in the cloned project. Optionally, you may choose to clone the template to two separate repositories, one for the assignments, and one for the project. To clone and test the template follow the steps below

1. Using the terminal change to the directory where all work will be done for this course. These instructions assume you will do all work in a folder called **neu** in your home directory, e.g., **~/neu**

    ```
    $ mkdir  ~/neu
    $ cd  ~/neu
    ```

2. Clone the template project by typing the following at the command line

    ```
    $ cd  ~/neu
    $ git  clone  https://github.com/jannunzi/angular-four-MEAN-seed
    ```

3. A directory called **angular-four-MEAN-seed** will be created. Check that it's there using **ls**

```
$ cd  ~/neu
$ ls
angular-four-MEAN-seed
```

4. Optionally rename the directory to something else, e.g., **`webdev`**

```
$ cd  ~/neu
$ mv  angular-four-MEAN-seed  webdev
```

5. Go into the directory and use **`npm`** to install the project. This will install all dependencies declared in **`package.json`**

```
$ cd  ~/neu/webdev
$ npm  install
```

6. Once all Node.js modules have been installed, run the project

    a. If not already running, make sure mongo is running

    ```
    $ sudo  mongod
    ```

    b. Run the front end using angular 4. This will start a local embedded server that will serve the Angular front end. You can stop the front end with Ctrl+c twice

    ```
    $ cd  ~/neu/webdev
    $ npm  start
    ```

    c. In a separate terminal, run the backend using Node.js. You can stop the back end with Ctrl+c

    ```
    $ cd  ~/neu/webdev
    $ nodemon  server.js
    ```

7. Test the application is running by navigating with the browser to the following URL

    http://localhost:4200

    NOTE: There is no server side for this assignment, but we added mongoDB services to test the database. The API will run on 3100. The port is specified in 2 locations. In **`server.js`** and **`environment.ts`**

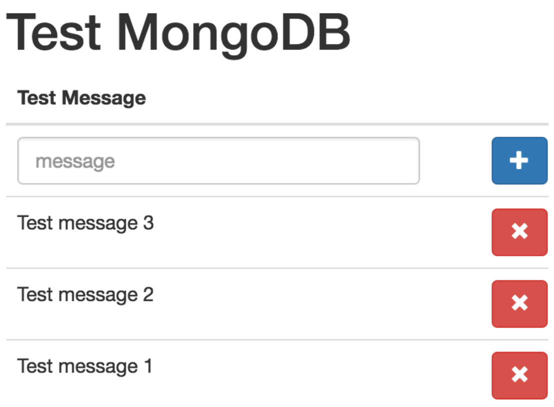
```
server.js
const port = '3100' ;
app.set('port', port);

environment.ts
export const environment = {
 production: false,
```

```
  baseUrl : 'http://localhost:3100'
 };
```

8. Click on Test MongoDB to test your database connection works. This navigates to

   http://localhost:4200/test/index.html

   Type several messages in the input field and press the blue plus button. The messages should be created on the local database and they should appear below the field. For instance



Test MongoDB

Test Message

Test message 3

Test message 2

Test message 1

9. Test deleting some of the messages. The messages should disappear as they are removed

Setting Up GitHub

GitHub will be used to keep track of all work. All code will be committed and pushed to a public GitHub repository. A private GitHub can be used, but it may not be free. The GitHub repository will allow TAs and instructors easy access to your work. When submitting an assignment, tag the latest commit with a particular tag for that assignment provided by the assignment. This will be used by the TAs and instructors to grade the correct version of your code.

1. Create an account on github.com. If you already have a GitHub account, feel free to use the same account
2. Create a repository called **webdev-LAST_NAME-FIRST_NAME**, where LAST_NAME and FIRST_NAME is **YOUR** last and first name. For **ME** it would be **webdev-annunziato-jose**
3. Commit and push your code to GitHub. In this example **jannunzi** is **MY** GitHub username, and the new repository is **webdev-annunziato-jose**. You would use **YOUR** GitHub username, and **YOUR** new repository name

```
$ cd ~/neu/webdev
$ git init
$ git commit -m "first commit"
$ git remote add github https://github.com/jannunzi/webdev-annunziato-jose.git
$ git push -u github master
```

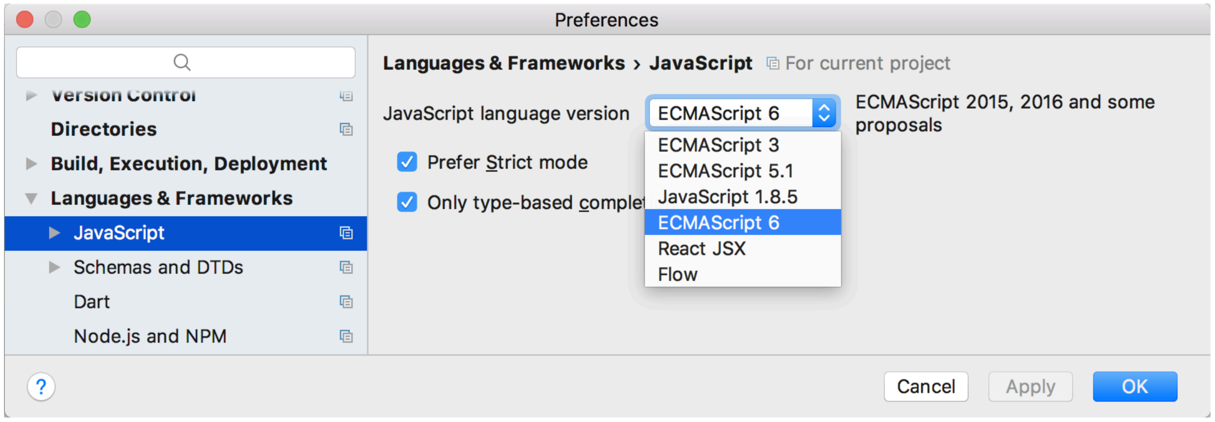
4. Click on the **Code** tab and verify that the source code is there
5. Invite the TAs and the instructors as administrators of the repository. This will allow them to help you debug your code if you run into trouble. It will also allow them to keep track of your progress

a. Go into **Settings**, then **Collaborators**. Search for the instructors and TAs by their username, and then click on **Add collaborator**
b. The instructor's GitHub username is **jannunzi**. The instructor's repository is located at https://github.com/jannunzi
c. The TAs will provide their GitHub usernames on **Piazza**

Setting Up WebStorm IDE

In this course we will be using the WebStorm IDE for developing MEAN stack Web applications. The IDE is free for users with an .edu email. Once you have downloaded and installed WebStorm, you will be able to clone and checkout the remote repositories from OpenShift. From within WebStorm you will be able to manage the repository by committing and pushing right from the IDE. The IDE will allow you to run and debug Node.js Web applications.

1. Download and install the WebStorm IDE on your local machine
2. Use your **.edu** email to register a free license
3. **Make sure you change the preferences in Webstorm.**
   Webstorm > Preferences> Languages > JavaScript > EcmaScript 6



Languages & Frameworks › JavaScript For current project

JavaScript language version: ECMAScript 6 — ECMAScript 2015, 2016 and some proposals

✓ Prefer Strict mode

✓ Only type-based complet

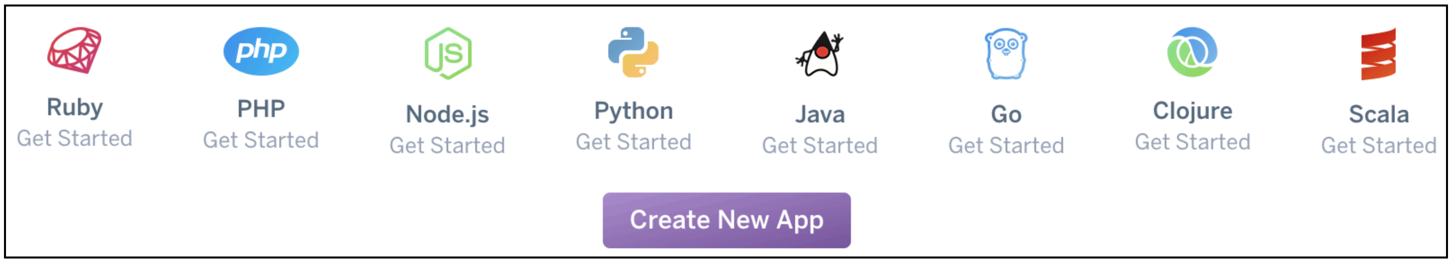
4. Start WebStorm and open the project created in earlier steps in `~/neu/webdev`
5. Create a run configuration so you can run and debug NodeJS applications from within WebStorm
   a. Select from the menu Run, Edit Configurations...
   b. In the Run/Debug Configurations window click on the plus button on the top left
   c. Select Node.js
   d. Name the configuration **webdev**
   e. Verify the **Node interpreter** field has the directory of where Node.js is installed
   f. Verify the **Working directory** field has the directory of your project
   g. In the **JavaScript file** field type **server.js**, the file you edited previously
6. Verify the application works locally
   a. From the top right of the IDE, select the dropdown and select webdev, the run time configuration you created earlier
   b. Click on the green play icon next to the dropdown to run your server
   c. Alternatively, you can start the local node.js server at the command line terminal at the bottom of the window typing the following:

```
$ node server.js
```

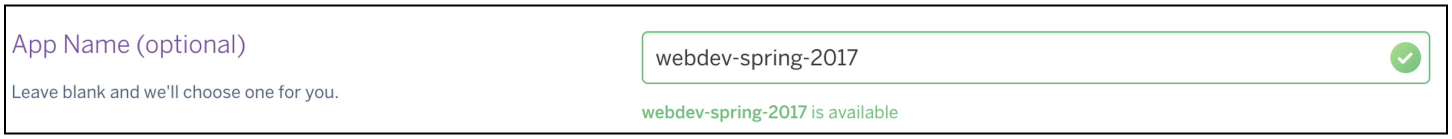
   d. With a browser, navigate to http://localhost:4200

Setting Up Heroku

Create an account at heroku.com and create a Node.js application



Name your application as suggested earlier, e.g., **`webdev-carpenter-john`**



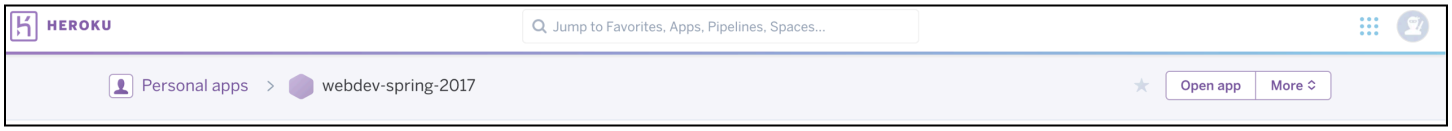
App Name (optional)

Leave blank and we'll choose one for you.

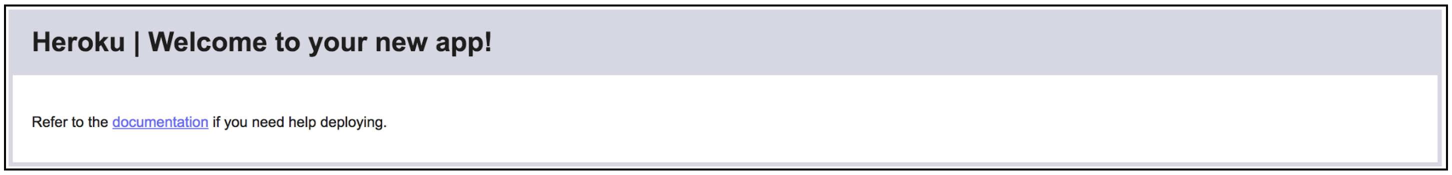
webdev-spring-2017

webdev-spring-2017 is available

Once the application has been created you can open the app by clicking on the ***Open app*** link



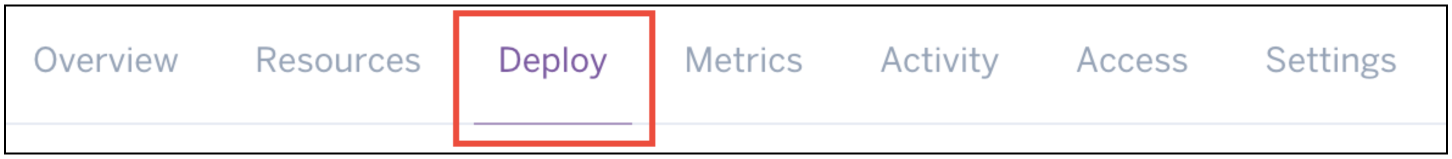
The default app is a simple placeholder we will be replacing shortly



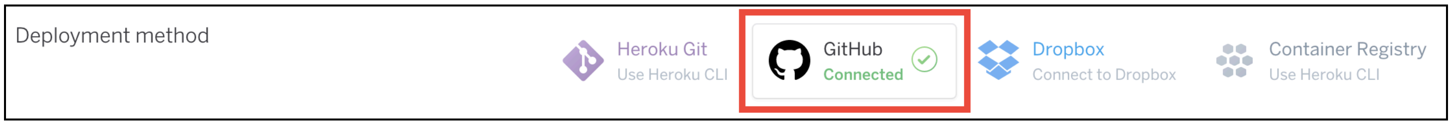
Heroku | Welcome to your new app!

Refer to the documentation if you need help deploying.

In the Heroku dashboard, select the deploy tab



Connect Heroku to auto deploy from your github repository, master branch. Now, when you commit and push to your repo, your remote application will auto deploy and restart



Back to the dashboard, navigate to the CLI page and install the heroku CLI which will allow administering and configuring the heroku application from your command line

Deploy using Heroku Git

Use git in the command line or a GUI tool to deploy

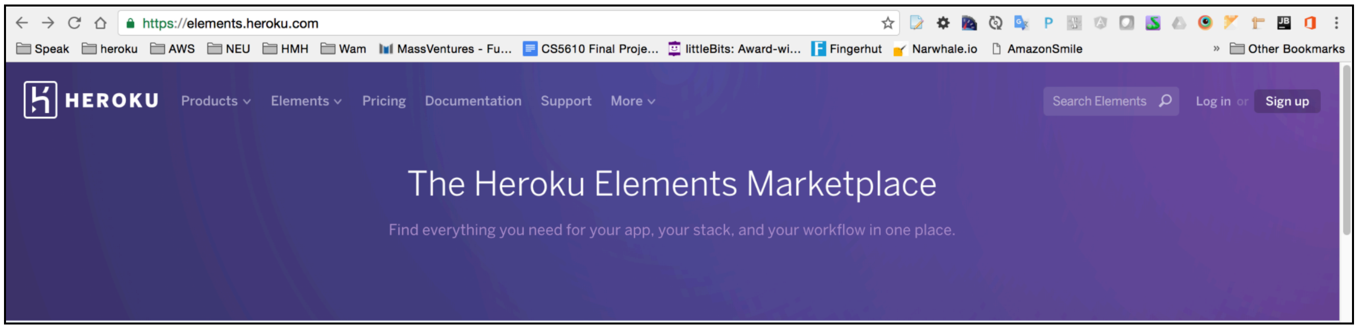
Install the Heroku CLI

Download and install the Heroku CLI.

Setting Up mLab

We will be using **mLab** as the remote mongo database. Follow the steps below to configure mLab from within Heroku

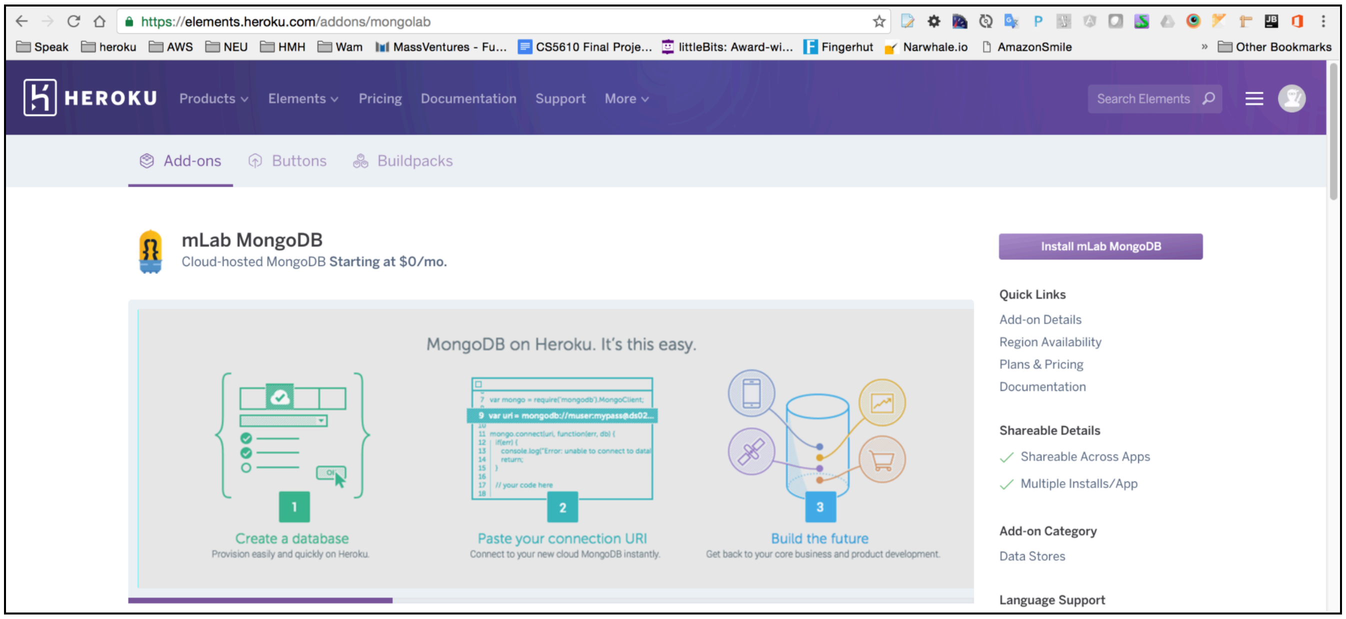
1. Login to Heroku
2. Navigate to Heroku's marketplace: https://elements.heroku.com



The Heroku Elements Marketplace

Find everything you need for your app, your stack, and your workflow in one place.

3. Click on mLab MongoDB which should take you to: https://elements.heroku.com/addons/mongolab



mLab MongoDB

Cloud-hosted MongoDB Starting at $0/mo.

MongoDB on Heroku. It's this easy.

Create a database
Provision easily and quickly on Heroku.

Paste your connection URI
Connect to your new cloud MongoDB instantly.

Build the future
Get back to your core business and product development.

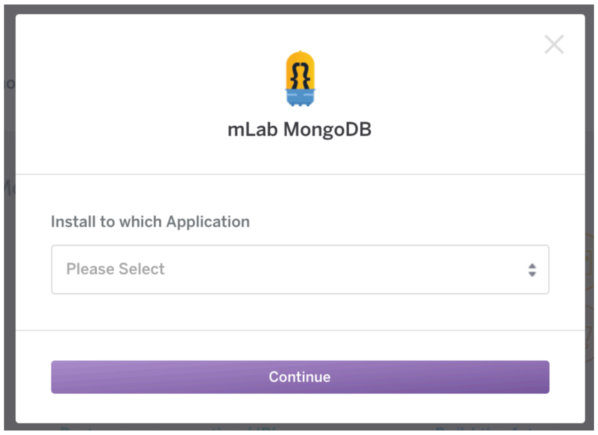
Quick Links
Add-on Details
Region Availability
Plans & Pricing
Documentation

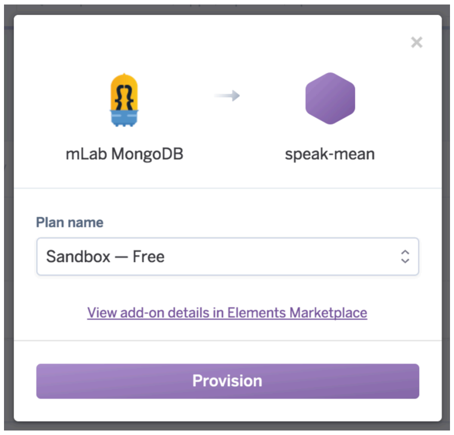
Shareable Details
✓ Shareable Across Apps
✓ Multiple Installs/App

Add-on Category
Data Stores

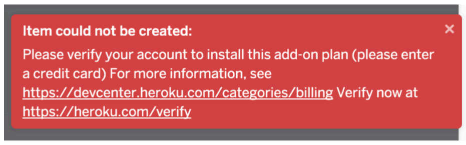
Language Support

4. Click on Install mLab MongoDB and then choose the Heroku application you want to associate MongoDB with and then choose the Sandbox - Free plan





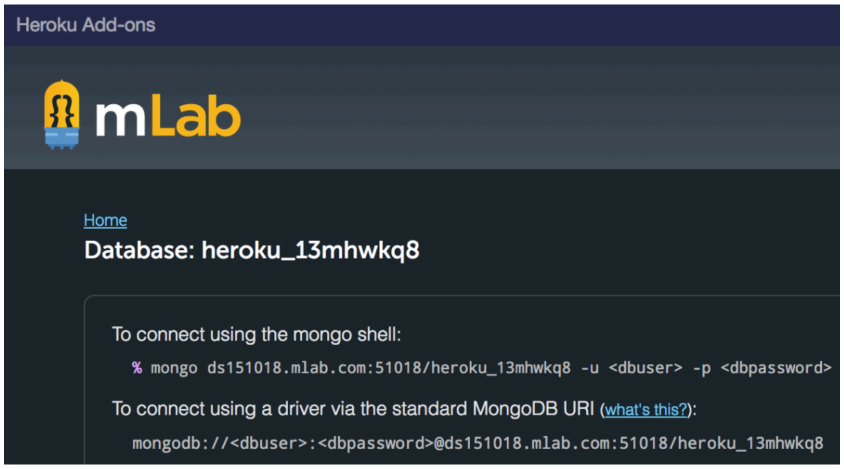
5. If you get an error, you might need to enter a credit card, but it will not be charged



Item could not be created:
Please verify your account to install this add-on plan (please enter a credit card) For more information, see https://devcenter.heroku.com/categories/billing Verify now at https://heroku.com/verify

6. mLab creates a free sandbox. Make note of the connection string since you will need it in an upcoming step. The connection string has the following pattern:

```
mongodb://<dbuser>:<dbpassword>@ds1234.mlab.com:12345/heroku_1234xyz
```



Database: heroku_13mhwkq8

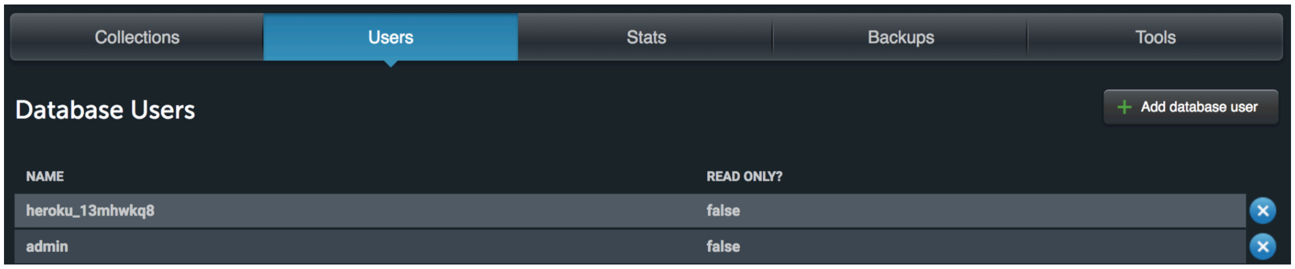
To connect using the mongo shell:

```
% mongo ds151018.mlab.com:51018/heroku_13mhwkq8 -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard MongoDB URI (what's this?):

```
mongodb://<dbuser>:<dbpassword>@ds151018.mlab.com:51018/heroku_13mhwkq8
```

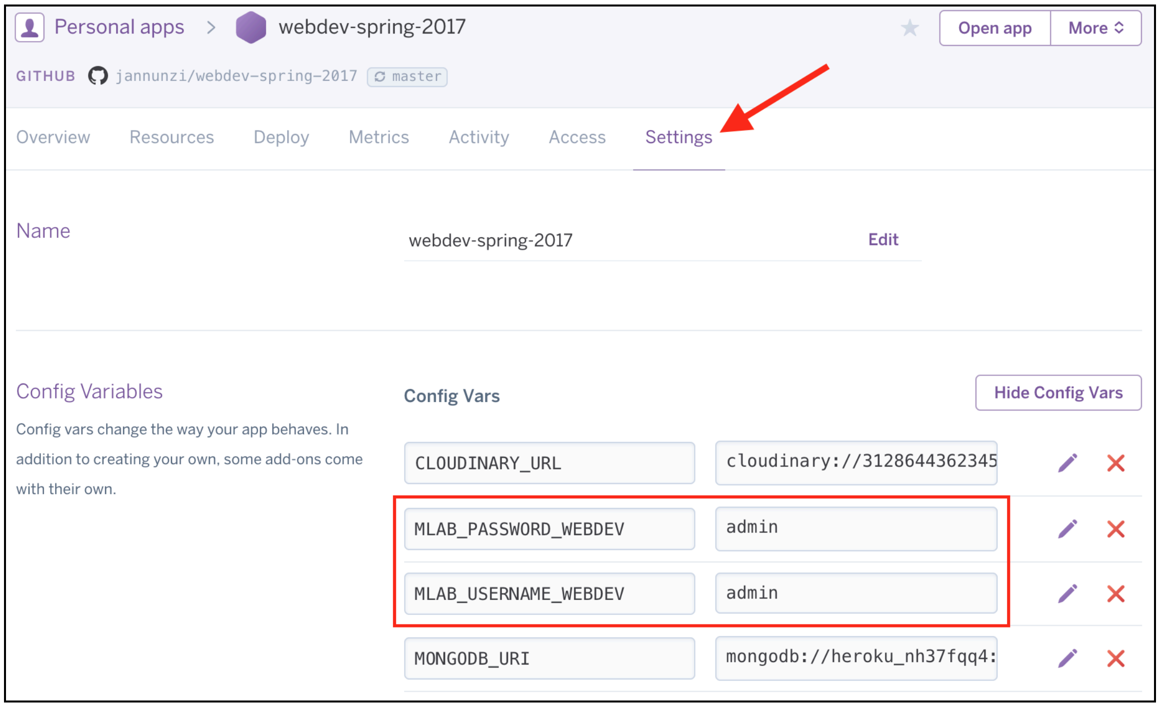
7. Create a username and password to fill in the <dbuser> and <dbpassword> in the connection string. Below I've created a username/password: admin/admin, but you can choose any username you please



NAME	READ ONLY?
heroku_13mhwkq8	false
admin	false

8. To test the connection to mLab from your remote server, we have provided JavaScript code that tests whether your server is running locally or remotely on heroku. This is implemented in

/server/test-mongodb/app.js. The script checks to see if the right environment variables are present and configures a connection string appropriately. If the script finds environment variables for the username and password for mLab, then it builds a connection string to the mLab MongoDB instance. Otherwise it uses a local connection string to connect to your local MongoDB instance. You could just replace the `<dbuser>` and `<dbpassword>` with the correct username and passwords from mLab, but that would be a bad idea since your source code would be publicly available on GitHub and everyone would be able to see your username and password for your database. Instead, store the username and password in environment variables in Heroku. Create environment variables in Heroku with the username and password. For instance, here are the username and password I'm using



Name: webdev-spring-2017

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

CLOUDINARY_URL	cloudinary://3128644362345
MLAB_PASSWORD_WEBDEV	admin
MLAB_USERNAME_WEBDEV	admin
MONGODB_URI	mongodb://heroku_nh37fqq4:

9. Then in your **/server/test-mongodb/app.js** file, the connection string can be set localhost when you run the server locally, or mLab URL when running on Heroku. Use the following code in **/server/test-mongodb/app.js** to check which environment you are running in and use the appropriate connection string

```
var connectionString = 'mongodb://127.0.0.1:27017/test'; // for local
if(process.env.MLAB_USERNAME_WEBDEV) { // check if running remotely
   var username = process.env.MLAB_USERNAME_WEBDEV; // get from environment
   var password = process.env.MLAB_PASSWORD_WEBDEV;
   connectionString = 'mongodb://' + username + ':' + password;
   connectionString += '@ds157268.mlab.com:57268/heroku_nh37fqq4'; // use yours
}
```

10. Replace "**@ds157268.mlab.com:57268/heroku_nh37fqq4**" above with your own URL given to you by mLab

11. Commit your changes and push. Heroku will auto deploy the remote application
12. Note: For your project/code to be able to be deployable on heroku. You need to run the following commands
ng build --env=prod

https://github.com/angular/angular-cli/wiki/build
dThis will create a dist folder in your project folder. Push this code to git/heroku to deploy on heroku.

Deliverables

For this course you will not be submitting any source code to Blackboard. All your source code should be available on GitHub. We only need to know the URL to the GitHub repository, and the URL to the hosted environment, e.g., Heroku, OpenShift, or AWS. You will submit the same two URLs as deliverables for all your assignments. Even though we will make note of these URLs, it will be convenient if you include them in all your submissions.

Blackboard Deliverables

For this (and all other) assignment please submit the following in Blackboard:

1. Submit GitHub URL
2. Submit the URL to the root of your hosted environment Web application, e.g., Heroku, AWS, or OpenShift

GitHub Deliverables

On GitHub

1. Invite the TAs and your instructor as a collaborator to your repository. TAs and instructors will provide their github usernames
2. Create a README.md file at the root of your github repository with the following information
    a. Say something about yourself, .e.g., you're a graduate student at Northeastern, past experience, if you are working, projects you've worked on, etc.
    b. Describe the purpose of repository, e.g., for webdev
    c. Create links to the following
        i. Root to the hosted environment on Heroku (or AWS, or OpenShift)
        ii. Link to your project (you might need to come back to this)
        iii. Link to your assignment (you might need to come back to this)
    d. Feel free to add more content
3. Create a Home page in your github wiki with the same information.
4. Upload an image of yourself to github (optional)
5. Update your bio on github, if you don't already have it