Heroku App Setup

1. Sign up for a free account at heroku.com
2. Install the Heroku Command Line Interface. Follow the instructions at https://devcenter.heroku.com/articles/heroku-cli. However, SKIP the step 'heroku create'. We'll do that later.
3. Configure your Node app.
    a. You should already have a package.json file in your project.
    b. Add a plain text file named 'Procfile' in your project folder. Inside that file write the following. But replace index.js with the name of your main JavaScript file. (Whatever run with node or nodemon. It might be server.js.)

    ```
    web: node index.js
    ```

    c. Add a file named 'app.json' to your project folder. In that file, put the following. Change the PROJECT NAME to whatever you call your app. It doesn't matter what you choose. Also fill in the description. But do not change the image.

    ```
    {
      "name": "PROJECT NAME",
      "description": "Teach the kids Heroku",
      "image": "heroku/nodejs"
    }
    ```

    d. Save and commit those changes.
    e. Now from the command line run 'heroku create' from your project folder. That will make this project into a new heroku app with a random name. Alternatively, you can pick a name: 'heroku create SOME_NAME`.
4. When your app is deployed, Heroku will tell you what port you have to use by setting a PORT environment variable. Put something like this in your JavaScript file to handle this. Use this port variable when you start express.

    ```
    var port = process.env.PORT || 5000;
    ```

5. After you've configured your project, anytime you want to deploy it...
    a. Commit the changes
    b. Run 'git push heroku master'
    c. For more info, https://devcenter.heroku.com/articles/deploying-nodejs#deploy-your-application-to-heroku
6. You can also test your Heroku config locally. Run the app using `heroku local web`.

Adding a Database

1. Add a PostgreSQL to your app. In your project folder, run `heroku addons:create heroku-postgresql:hobby-dev`.
2. You can now connect to your database with PG Admin
    a. On the Heroku website, pick "Data" from the menu at the top right.
    b. Select your database.
    c. Scroll down to "Database Credentials" and click on "View Credentials".
    d. Use this data to configure a new Server in PG Admin. When you create a new Server, there is a connection tab where you fill it in.
3. In order to connect your app to the database, note that Heroku will provide a DATABASE_URL environment variable that is the PG connection string. When running locally, I recommend you set this environment variable.

Configure Your Code to Connect to Database

1. Add the following file to your project **pg-connection-pool.js**

```
const pg = require('pg');
const url = require('url');
try {
    // When not running via Heroku, this will load the .env file.
    require('dotenv').config();
} catch (e) {
    // When running with Heroku, dotenv doesn't need to be available.
}

console.info("DATABASE_URL:", process.env.DATABASE_URL);
const params = url.parse(process.env.DATABASE_URL);
const auth = params.auth.split(':');

const config = {
  user: auth[0],
  password: auth[1],
  host: params.hostname,
  port: params.port,
  database: params.pathname.split('/')[1],
  ssl: params.hostname !== 'localhost'
};

module.exports = new pg.Pool(config);
```

2. In your other Node JavaScript files you can now...

```
var pool = require("./pg-connection-pool");
```

3. Add dotenv as an npm dev dependency.
   $ npm install dotenv --save-dev
4. Create a **.env** file to set your DATABASE_URL environment variable

```
DATABASE_URL=postgres://postgres:<password>@localhost:5432/<database>
```

5. Ignore the .env file in .gitignore. You want to keep this file secret. All of your team members will have to create the same file manually.