Deploying Node.js to Heroku

Review

- Up until now we've been building and running our applications locally on our computers.
- What's needed to run our web applications:
  - Need to have NodeJS installed.
  - Need to have postgres installed, configured and running.
  - Need to start your NodeJS application.

Background - Cloud Computing

- Cloud Computing is, as the name suggests, being able to have a constantly running computer in the cloud.
- All the things above we've been doing on our local computers we can do on these cloud computers.

Cloud Computing

- Distributed Servers: having multiple servers running in geographically different locations.
- Datacenters: The data warehouses that hold our application's code along with the database data.

(PS...In real life, your database and your application are running on different machines)

Cloud Computing

- Cloud Computing get complicated when you have thousands of computers running simultaneously.
- How do you manage memory?
- What about databases? I don't want you to have access to my db!
- There is a lot of configuration needed to have multiple users being able to run different servers/databases.
- This is what services like Heroku or Amazon Web Services (AWS) offer you:
  - A constantly running computer in the cloud without all the configuration.
  - Manage the overhead of maintaining all these servers simultaneously.

What is deployment?

- Deployment is the process of taking your application, uploading it to a server in the cloud and subsequently running it on that server.
- The cloud service you use will be responsible for running your application and making sure the environment is stable so that your users are happy!

Heroku

Heroku is a simple solution for hosting a web application. It allows you to focus on building your app and not setting up or maintaining infrastructure.

> Heroku is a cloud platform that lets companies build, deliver, monitor and scale apps — we're the fastest way to go from idea to URL, bypassing all those infrastructure headaches.
>
> *— Heroku*

Heroku

First you will need to make sure you have node, npm, and git installed locally

Run the following commands and make sure they output a version number

```
$ node -v
$ npm -v
$ git --version
```

Heroku

Sign up for a Heroku account

https://signup.heroku.com/dc

Heroku

Install the Heroku CLI

https://devcenter.heroku.com/articles/heroku-command-line

- The Heroku CLI (Command Line Interface) is simply a way for us to interact with the Heroku services via the command line instead of a website.

Heroku

Log in to Heroku over the command line

```
$ heroku login
```

Heroku

Clone the example blogger application node application

```
$ git clone https://github.com/amyhua/blogger
$ cd blogger
```

First Step

- We will need to make a few modifications to our code so it will work when we deploy it to Heroku.
- First, we will create what's called a 'Procfile'.

```
$ touch Procfile
```

Procfile

- The Procfile is the file that Heroku looks for to find instructions on how to run your application.
- This is where we will provide our instructions. Add the following to your Procfile

```
web: npm start
```

- What does that 'web: ' mean from our Procfile?
- It what specifies which type of 'dyno' we need to use.

Procfile Explanation - Dynos

- A Heroku dyno is the actual worker that runs your application.
- You can think of a dyno as a single command line window that runs exactly one command.
- There are different types of dynos, but what we want is a 'web' dyno that is responsible for running our web application
- This is the reason why we specify 'web: '

Second Step - Database URL

- We will need to update our code to use an environment variable to specify how to connect to our database.
- However, we still want to use the same configuration when running locally, so what we will have is:

```
// ./models/index.js

var sequelize;
if (process.env.DATABASE_URL) {
  sequelize = new Sequelize(process.env.DATABASE_URL);
} else {
  sequelize = new Sequelize(config.database, config.username, config.password, config);
}
```

Third Step - Provision a Database

- We need to tell Heroku that we want a postgres database for our application.
- To do this, we will use what are called Heroku add-on's.
- An add-on is a feature that is available inside the Heroku ecosystem that you can utilize in your application.
  - https://elements.heroku.com/addons
- To provision our postgress add-on run the command:

```
$ heroku addons:create heroku-postgresql:hobby-dev
```

- This command creates a database and a DATABASE_URL environment variable.

Heroku - Create

Next step is to create an app on Heroku that will host your source code (make sure you are inside the blogger directory)

```
$ heroku create
```

You can also specify the name of your application

```
$ heroku create my-app
```

Heroku

The heroku create command will generate a random app name for you on Heroku. It will also create a new git remote for you called heroku.

```
$ git remote -v

heroku  https://git.heroku.com/gd-blogger.git (fetch)
heroku  https://git.heroku.com/gd-blogger.git (push)
origin  git@github.com:amyhua/blogger.git (fetch)
origin  git@github.com:amyhua/blogger.git (push)
```

git remotes

- A git remote is simply a reference to where the code is being pushed.
- In our case, when we want to push our code up to 'github' we use the following command because as we can see the 'origin' remote represents github.

```
$ git push origin master
```

- The 'heroku create' command generated a new remote called 'heroku' which points to Heroku.

Heroku

Deploy your code

- So if we want to push our code to Heroku, we will use the 'heroku' remote.

```
$ git add .
$ git commit -m "heroku setup"
$ git push heroku master
```

Heroku

Open the app with:

```
$ heroku open
```

You just deployed an application to Heroku!

Heroku

- You will need to be able to view your application logs for successful debugging.

```
$ heroku logs --tail
```

- The --tail option tells heroku to display the last 10 lines of the output
- Refresh your browser with your terminal open to see a new GET request logged to your terminal window.

Heroku

Heroku recognizes an app as a node.js app by the existence of a package.json file in the root directory. Your package.json file will determine which version of node Heroku will use to run your app and which dependencies should be installed. Heroku will run `npm install` when your app is deployed.

Resources

- https://devcenter.heroku.com/articles/how-heroku-works#defining-an-application
- https://devcenter.heroku.com/articles/heroku-command-line
- https://devcenter.heroku.com/articles/getting-started-with-nodejs#introduction