Deployment

Contents

Introduction

We have reached the point where we want our projects set free into the world. We are deploying.

Signing up and getting ready

First things first, we need to set up a few accounts to get things ready:

- Get a Netlify account (free): https://www.netlify.com
- Get a Heroku account (free): https://www.heroku.com
- Set up the Heroku CLI for your OS: https://devcenter.heroku.com/articles/getting-started-with-nodejs#set-up

Setting up the local environment

We need to start our projects in a place where they aren't within a Git repo. This is true for any project being tracked by Git, but will be even more pertinent as we attempt to deploy.

You don't need to follow these steps, and can adapt them to the project you are working on, but just be careful that you understand which parts of this will apply, and which you can leave out.

1. Get to a workspace where you can create some new projects, and that isn't within a git repo (you should see the 'fatal' message if you do > git status.
2. Make two directories, one for front-end and one for the backend. We are going to deploy them separately. (Mine are called deploy-express and deploy-front-end.)
3. Add in the following code to a server.js file in the root of the backend directory:

```
const express = require('express')
const app = express()

const asia = {
    name: "Asia",
    countries: [
        {
            name: "Japan",
            cities: [
                {
                    name: "Tokyo"
                },
                {
                    name: "Osaka"
                }
            ]
        },
        {
            name: "Laos",
            cities: []
        }
    ]
}

app.get('/', function(req, res) {
    console.log("Success")
    res.send("Success connecting to API")
})

app.get('/continent', function(req, res) {
    res.send(asia)
})

app.get('/countries', function(req, res) {
    res.send(asia.countries)
})

app.get('/cities', function(req, res) {
    res.send(asia.countries[0].cities)
})

app.listen(3000, function() {
    console.log("express running")
})
```

A Note on copying from Slack

Use the view raw option if you want to be careful about not copying across special and invisible extra characters. It's an option under the 'three dots' menu.

Install your dependencies, so you'll need to `> yarn init`, and `> yarn add express`. When doing the yarn init please give the entry point as `server.js`.

Heroku Instructions

Please follow the instructions in here, although I will pull out the most pertinent ones:

https://devcenter.heroku.com/articles/deploying-nodejs

This will go into your `package.json` to tell heroku your Node version:

```
"engines": {
    "node": "10.x"
},
```

Change your server file to be able to run from a different port in production:

```
const port = process.env.PORT || 3000;
```

(also change the 3000 in the listen function to port)

Add git

Create a git repo: `> git init`

And then commit the project so far (`> git add .` and `git commit -m "..`)

Then we need to login to heroku from the CLI

`> heroku login`

Login via the website, and hopefully that works.

Now we need to create the connection to Heroku:

`> heroku create`

And push the changes that we have committed to git:

`> git push heroku master`

Now a whole lot of logs will roll up the screen. Have a look through these. They will be confusing at first, but they give you good information, and you will need to get comfortable with them should things not go smoothly.

You should be able to go to the link that heroku gives you at the end of the deploy logs.

Dotenv

Now we are adding dotenv package, to help us to hide certain things, and also differentiate the code when in dev and production. The idea here is that we don't want to be commiting our API keys and passwords and any other important and personal information, and having them available on GitHub or anywhere else. Instead, we are going to write these secrets into a file, and make sure that is being ignored by Git. Then we will set them manually on the deployment side.

The other thing that environment variables are useful for is keeping track of variables that will change from the development and production environments. For example, when in development, we will usually be hitting our API at `localhost`. But once it's all deployed, our server is going to be on the internet at a `heroku` URL. We want to be able to make that change smoothly, and set these variables to change depending on the environment that they are being used within.

`> yarn add dotenv`

Then in the root of your project make a file called `.env`

Make sure that you put `.env` into your `.gitignore` file:

```
.env
```

In that file put this:

```
MY_SECRET=This is meant to be secret from GitHub
```

Here is some more info: https://www.npmjs.com/package/dotenv

Dotenv helps to load environment variables from your `.env` file.

Add this to your Express server:

```
require('dotenv').config()
```

Then add these updates to your root route:

```
// updatedRootRoute.js
app.get('/', function(req, res) {
    console.log("Success")
    const mySecret = process.env.MY_SECRET
    res.send(`Success connecting to API, and the secret is ${mySecret}`)
})
```

Now we need to set `MY_SECRET` on the heroku side:

`> heroku config:set MY_SECRET="My heroku secret"`

Then you can hit the Heroku URL again (at the root), and hopefully you will see the secret displayed.

Netlify

Have a crack at deploying on Netlify. It's designed to be easy, and it's very much worth having a go at this and seeing if you can get it up and running.

Netlify: https://www.netlify.com/blog/2016/09/29/a-step-by-step-guide-deploying-on-netlify/