Easily deploy a Vue + Webpack App to Heroku in 5 Steps [tutorial]

Nick Manning Follow
Sep 3, 2017 · 4 min read

After some serious Google searches on this topic, I finally got fed up and wrote this tutorial as I couldn't believe there was just poor information out there on this topic!

What We'll Cover in This Quick Tutorial:

1. **Create a vue project**
2. **Create a Heroku app**
3. **Configure vue project so that Heroku can serve up our vue code.**
4. **Push and deploy!**

Even though I've been mostly on backend bot frameworks in Golang for drinkeasy.co (a chatbot that sells alcohol), Vue.js is still my top choice for things like one-off marketing or hobby sites.

Here we go…

1. Generate a Vue.js Project

First off, install Vue.js (instructions here). We'll also need Vue's CLI to easily generate our project. (As a bonus option, feel free to use the awesome Yarn dependency manager rather than npm for these examples. Just substitute the `npm` command with `yarn` instead throughout the rest of this tutorial):

```
npm install --global vue-cli
vue init webpack <YOUR-PROJECT-NAME-HERE>
cd <YOUR-PROJECT-NAME-HERE>
npm install
npm run dev
```

Make sure your browser window opens and displays the starter project.

2. Create Your Heroku App

Heroku is a platform that let's us easily deploy and host our Vue.js app. If you haven't already, sign up for a Heroku account here. Then, install Heroku's CLI tool via the instructions here. Then, let's create our Heroku app:

```
heroku create <YOUR-PROJECT-NAME-HERE>
```

When this is done, you'll get a fresh URL to your project, i.e. `https://<YOUR-PROJECT-NAME-HERE>.herokuapp.com`. Make sure you head over to the URL and see a temporary Heroku landing page there.

Lastly, in order to avoid having Heroku install needless development dependencies when deploying later, let's go ahead and set the `NODE_ENV` setting to `production` :

```
heroku config:set NODE_ENV=production --app <YOUR-PROJECT-NAME-HERE>
```

3. Create a server.js and Build Your Site

Since Vue is only a frontend library, the easiest way to host it and do things like serve up assets is to create a simple Express friendly script that Heroku can use to start a mini-web server. Read up quickly on Express if you haven't already. After that, add express:

```
npm install express --save
```

Now add a `server.js` file to your project's root directory:

```
// server.js

var express = require('express');
var path = require('path');
var serveStatic = require('serve-static');

app = express();
app.use(serveStatic(__dirname + "/dist"));

var port = process.env.PORT || 5000;
app.listen(port);

console.log('server started '+ port);
```

IMPORTANT: What you probably noticed is that this will serve up a `dist` directory. `dist` is a predefined directory that Vue.js builds which is a compressed, minified version of your site. We'll build this and then tell Heroku to run `server.js` so Heroku hosts up this `dist` directory:

```
npm run build
```

You should see an output `dist` directory now.

Let's test our `server.js` file by running it:

```
node server.js
```

Tiny highlight

Now go to http://localhost:5000 and make sure your app loads. This is the actual site Heroku will serve up.

Lastly, we'll have to edit our `start` script in `package.json` to start our node server, as Heroku will automatically look for this script when looking for how to run a node.js app.

```
// package.json
{
  "name": "<YOUR-PROJECT-NAME-HERE>",
  "description": "A Vue.js project",
  "author": "",
  "private": true,
  "scripts": {
    "dev": "node build/dev-server.js",
    "build": "node build/build.js",
    "start": "node server.js",  <--- EDIT THIS LINE HERE
...
```

4. Git Init and Add Your Heroku Remote Repository

Heroku allows us to push to a remote repository so we'll first need to create our own git repository:

```
git init
```

Now let's add our Heroku remote repository:

```
heroku git:remote --app <YOUR-PROJECT-NAME-HERE>
```

Let's keep our generated `dist` directory so that we can always keep a pristine copy of what we've deployed to Heroku by removing `dist/` from `.gitignore`

```
.DS_Store
node_modules/
dist/  <--- REMOVE THIS LINE
npm-debug.log*
yarn-debug.log*
yarn-error.log*
test/unit/coverage
test/e2e/reports
selenium-debug.log

# Editor directories and files
.idea
*.suo
*.ntvs*
*.njsproj
*.sln
```

Now, most importantly, let's add and commit our code files:

```
git add . && git commit -a -m "Adding files."
```

5. Push Your Code to Deploy!

Now all we need to deploy to Heroku is:

```
git push heroku master
```

This will take our committed code, push it to Heroku's remote repository, run our `start` command in `package.json` which will serve up our freshly built `dist` directory.

If you come across any issues, you can always run `heroku logs` to troubleshoot.

If deployment is successful, test out your project's URL `https://<YOUR-PROJECT-NAME-HERE>.herokuapp.com` and you're done!

. . .

I hope this tutorial was helpful to anyone else navigating the aggravating maze that is web development in 2017.

BONUS TIP: Heroku's free tier forces your app to go to sleep if there's no traffic hitting it after awhile, thus causing some serious "wake up" time if someone tries to check out your app. One thing I like to do is set a free health check (i.e. via pingdom.com) that hits my Heroku URL every few minutes to keep it awake.

Vuejs Heroku Nodejs JavaScript Web Development

1.6K claps

WRITTEN BY
Nick Manning Follow
Code-linguist, father, entrepreneur and a friend to all devs.
Creator of fluttercrashcourse.com

Netscape Follow
A community dedicated to those who use JavaScript every day.

See responses (56)

More From Medium

More from Nick Manning

What It Was Like to Write a Full Blown Flutter App
Nick Manning in… Jun 15, 2018 · 9 min read 6.9K

Also tagged Nodejs

Going BIG with JavaScript: Numbers
Casey Gibson Feb 2 · 4 min read 74

Related reads

How To Use Vuetify in Vue.js
John Au-Yeung in Better… Aug 24, 2018 · 11 min read 206