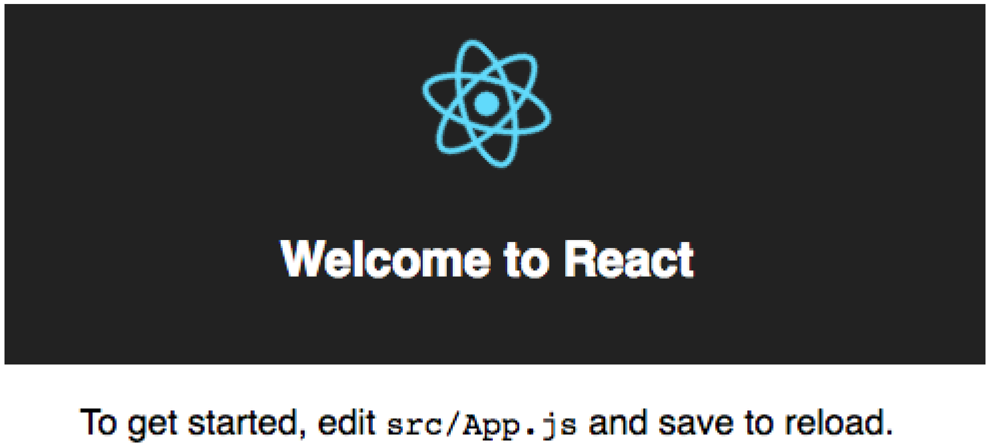
Tutorial: how to deploy a production React app to Heroku

Jeremy Gottfried Follow
Aug 3, 2018 · 4 min read



Create-react-app and Heroku are great tools for building highly performant apps, but the React and Heroku docs have very little information on how to deploy React production builds to heroku. I will guide you through deploying a simple production-ready React app to Heroku. These steps will work for any React app built with `create-react-app` .

Step 1: Create a React App

First, you’ll need to create a React app using the create-react-app generator. Make sure you have installed Node and npm first.

In the terminal, enter the following to generate a new react app (and replace `hello-world` with your app name) :

```
npx create-react-app hello-world

cd hello-world
```

Open the `hello-world` repository in your preferred text editor. If you’re using atom, simply type `atom .` in the terminal to open your repo. Here is what the repo will look like:

```
hello-world
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── registerServiceWorker.js
```

Step 2: Add your own app icon to the Public folder

You can convert any PNG into a `favicon.ico` file here: https://www.favicon-generator.org/

Then delete the existing favicon.ico file from the Public folder and add your own favicon.ico file to the Public folder. If you don’t add your own icon, the app will default to the React icon.

Step 3: Create an Express JS server to serve your production build

In your repository, create a file called server.js:

```
touch server.js
```

In server.js, copy/paste the following code:

```
//server.js


const express = require('express');
const favicon = require('express-favicon');
const path = require('path');
const port = process.env.PORT || 8080;
const app = express();
app.use(favicon(__dirname + '/build/favicon.ico'));
// the __dirname is the current directory from where the script is
running
app.use(express.static(__dirname));
app.use(express.static(path.join(__dirname, 'build')));
app.get('/ping', function (req, res) {
 return res.send('pong');
});
app.get('/*', function (req, res) {
  res.sendFile(path.join(__dirname, 'build', 'index.html'));
});
```

```
app.listen(port);
```

This code creates a special Node JS server that can serve minified/uglified JS. /ping is a meaningless path you can use to test that the server is working.

Make sure you add `express`, `express-favicon`, and `path` to your dependencies:

```
yarn add express express-favicon path
```

In your package.json file, change the `start` script to the following:

```
start: "node server.js"
```

Step 4: Create a React production build

Heroku now runs the build command automatically when you deploy, but it's a good idea to test the production build locally before deploying (especially your first time).

You can create a production build locally by running this command in your terminal:

```
yarn build
```

Then run `yarn start` to start the production server locally.

Step 5: Prevent source code from being deployed

In your repository, create a file called `.env` :

`touch .env`

Then add the following to the .env file to prevent source maps from being generated.

```
#.env
```

```
GENERATE_SOURCEMAP=false
```

Source maps allow you to access your source code in the production environment, which makes debugging easier. However, source maps also allow anyone from the public to view your source code.

Note: if you are having trouble debugging an issue in production but you want to keep your source code private, you can create a separate branch, remove that line from the `.env` file, and deploy that branch to a secret heroku url.

Step 6: Deploy to heroku

If you don't already have a heroku account, create one here: https://signup.heroku.com/

In your command line, run the following:

```
heroku login
```

You will need to type in your heroku credentials to the terminal. Once you have successfully entered your heroku credentials, run the following in your terminal to create a new deployed app:

```
heroku create sample-react-production-app
```

Replace sample-react-production-app with your own app name.

Then push your app build to heroku with the following git in your terminal:

```
yarn install
git add .
git commit -m "initial commit"
heroku git:remote -a sample-react-production-app
git push heroku master
```

These commands install your dependencies, initialize git, and connect your repo to the remote repository hosted by heroku.

Note: if you already initialized your git before running `heroku create [app-name]` , then you don’t need to run `heroku git:remote -a [app-name]` .

Congrats! Now you've completed all the necessary steps to deploy a React build. To view your app, run the following in the terminal:

```
heroku open
```

Enjoy! You can also see what the final repo looks like here: https://github.com/jeremygottfried/sample-react-production-app

JavaScript React Create React App Heroku Deployment