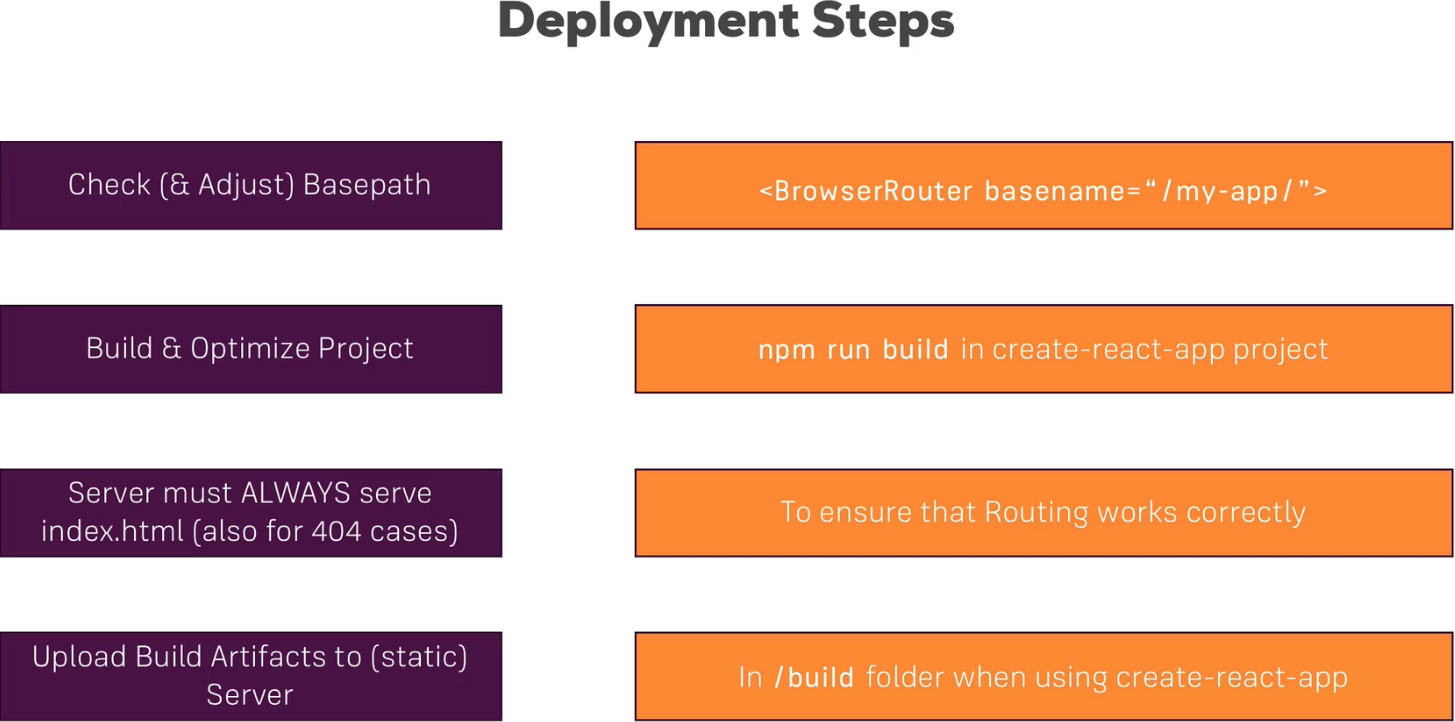
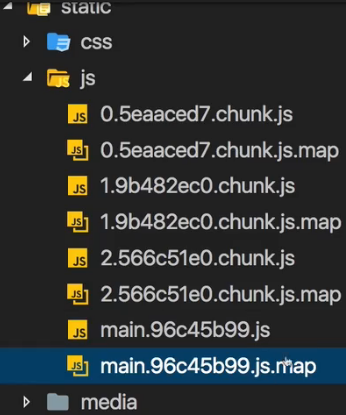
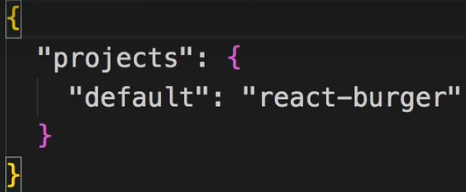
**Section 21 done: 6/6 Deploying the App to the Web**  
**Introduction**  
\* Thankfully it’s super easy.  
**Deployment Steps**  
**=> 1. =>** First of all we need to **check and possibly adjust the Basepath**. This is only important when we use the React Router.  
=> And there when wrapping the App in the BrowserRouter, you can set your **basename** property.  
=> That is always required when your app is served from something else than the root route, the root domain.  
=> If you’re serving your app from “example.com/my-app”, then you need to set the basename to “/my-app/” for the Router to work correctly.  
**=> 2. =>** The second important thing is that we need to **BUILD & OPTIMIZE the Project**.  
=> This is done with 1 convenient command in our application: **npm run build** in create-create-app projects.  
=> That will automatically build the project and optimize all our JavaScript bundles, to have a very small codebase we then actually upload on a server because we obviously want to ship as little code as possible since that’ll be as fast as possible.  
**=> 3. =>** Now another important thing to keep in mind is: **your server must always serve the index.html file, also and especially in 404 cases**.  
=> That’s required due to the way the Internet works, if your user visits my-app.com/users, then the server is the first to receive that route and chances are the server doesn’t know the “/users” path because you define it in your React app.  
=> Now for that to not throw an error, your React app needs to get a chance of parsing the path and it only does so if you return the index.html file for that unknown route, it is unknown to the server in the end.  
=> So always return index.html in 404 cases.  
=> Otherwise your Routing-dependent React applications won’t work.  
**=> 4. =>** Finally, **upload the Build Artifacts you get from the 2nd step to a (Static) Server**.  
=> You don’t need a server running PHP or Node.js or anything like that, in the end what you’ll deploy here is a SPA (Single Page Application) consisting of HTML and a little bit of CSS and a lot of JavaScript.  
=> None of that needs a server-side language.  
=> So a Static Server like AWS S3, GitHub Pages, or Firebase, or any other server will do.  
=> And you just upload the Build Artifacts, not your entire project folder.  
=> The Build Artifacts will be stored in a “./build” folder.  
  
\* And that’s actually all.  
**Building the Project**  
\* Now I want to deploy our application.  
**=> 1. =>** The Basepath.  
=> In index.js file there we could set the basename property but I plan on serving my app from myDomain.com/ (slash nothing) **so I don’t need to set this**.  
**=> 2. =>** Ensure that the server always returns the index.html file.  
=> That is something all Static HOSTS typically allow you to configure. And if you’re not using a Static HOST but you are using your Node.js server, you just need to write code - you need to set up a CATCH ALL Route and return index.html and the same logic of course applies for any other server-side language.   
=> **So we don’t have to do it now**.  
**=> 3. =>** Build the Project.  
**npm run build**  
=> This will now build our project, similar to `npm start` but now it will also spit out a folder and not just store the result in memory and that folder will contain our Optimized PRODUCTION Build.  
=> If we have a look at the folder, we see our single page - index.html, which also is optimized to consume as little space as possible, you see that some script imports were added there like to this:  
  
=> And that is what we can find in the `static` folder, there we got our `media` files - images like the burger logo, and the `js` folder with JavaScript files for our different chunks, for Lazy Loading and the main.js file.  
  
=> We also got source-maps here, now you don’t need to deploy these though, these are purely optional, you can focus on the .js files but you can also - and that is what I’ll do - deploy the content of the whole build folder.  
\* And that’s important: the content of the `build` folder.  
\* Ship that content, all the files there and the `static` folder to your Static HOST.  
**Example: Deploying on Firebase**  
\* You’re really free to choose whichever Static file HOST you want to choose.  
=> **Simply follow the instructions of your chosen HOST to upload the files in the `build` folder**.  
=> **And make sure you always return the index.html file**.  
\* I’ll use Firebase hosting.  
> Hosting > Get Started > for Firebase HOSTING we actually need to install an extra tool with:   
**npm install -g firebase-tools**   
globally on our machine, if you don’t want this, then simply follow along with another Static HOST of your choice, like AWS S3, check out their documentations to learn how to deploy to that specific page.  
\* Now run:  
**firebase login**  
**firebase init**  
=> Now we can navigate to HOSTING and press SPACE to check that option, we don’t need any of the other, then hit ENTER.  
=> Now you choose the project you want to choose.  
=> Now you’re asked what your public directory is and for us that’s not the default of `public`, but `build`. So just type “build”. That tells Firebase which files to upload and we want to upload all files in the `build` directory.  
=> Now we’re asked if we want to configure this is a SPA Single Page App. “y”. This will automatically set this up to always return index.html as it says here.  
=> Now it asks us if it should overwrite the index.html file it found. “n”. Because we obviously want to keep the one generated by our project and with that we’re done.  
\* Now in the case of Firebase, this gave us a special .firebaserc file and a firebase.json file to configure this project.  
\* **.firebaserc** simply contains information about this project, obviously you should replace this with your project ID if you ever download my project and want to use that.  
  
\* **firebase.json** this is where the HOSTING is configured, where a `public` folder is set up to be the `build` folder, where it ignores some files to not upload these, and where it also rewrites all the incoming Requests to return index.html.  
  
\* And for Firebase you can read more in their documentation, for your favorite Static Site HOST you pick their docs of course.  
**firebase deploy**  
\* For other HOSTS you might need to manually upload the files there, that also shouldn’t be too difficult though.  
\* And now we can visit the application here under this HOSTING URL:  
  
\* And there also are ways for assigning your own domain, just check out the Firebase docs.  
\* This is now our application running on a real server.  
**Wrap Up**  
\* So we deployed our application in this module, the last and of course very important step to really conclude this project.  
\* We now see it run on the Internet and that of course is a huge step.   
\* It is a step towards really building a realistic React apps because such apps of courssse should also be able to run on the Internet.  
\* As you saw, deployment is super easy, it’s 1 command and a couple of things to keep in mind and no matter which HOST you then choose, these steps never change.