Core Admin

A message from Gopher Labs:

Thank you for purchasing this product. We are hard at work here to make the experience of using Core Admin, customization and deployment of it easier for you. Feel free to send us a message on ThemeForest if you have any further questions!

Table of Contents:

1. Installation

1.1) In order to install and run this application on your local system, you would need to have **NodeJS** installed.

1.2) After doing so, launch your terminal environment (Terminal app in Mac, Command prompt in Windows), and browse into the directory of this application. Replace ***cd ~/Downloads/core-admin*** with the correct folder.

```
cd ~/Downloads/core-admin
```

To learn the basics of terminal for Mac read **this**.

1.3) Now we need to use the Node Package Manager (NPM) to install the dependencies of our application. You would install NPM automatically along with NodeJS.

To check if NPM is installed, run this command:

```
npm -v
```

If the output shown to you is anything but a version number, most likely your Node installation wasn't successful.

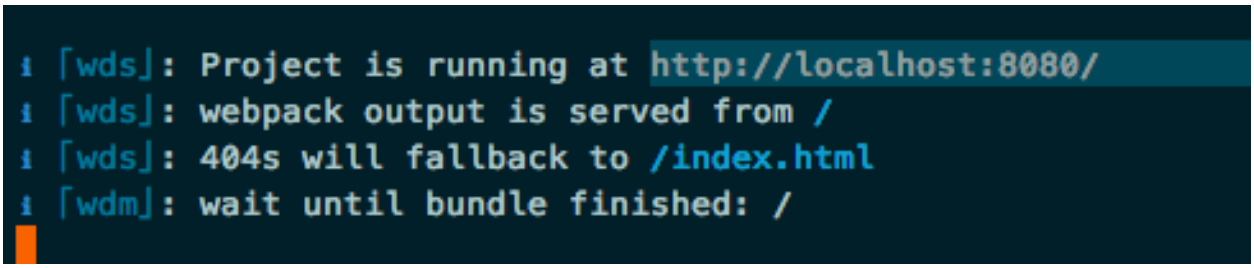
Once NPM is available in our system, we will run this command while being in the applications main folder, in order to install all the node modules (dependencies) that the application needs.

```
npm install
```

1.4) The installation of packages will take a while, depending on your internet connection. Once it is over, running this command you will be able to get the application running in a development environment:

```
npm start
```

The application should automatically open in your default browser, but if for any reason it doesn't, you can go to the URL that is outputted in your terminal after running the command above:



This is a **development** server that you will be running, and any changes you make to the application files will trigger a refresh, and you will see the updates as you make them.

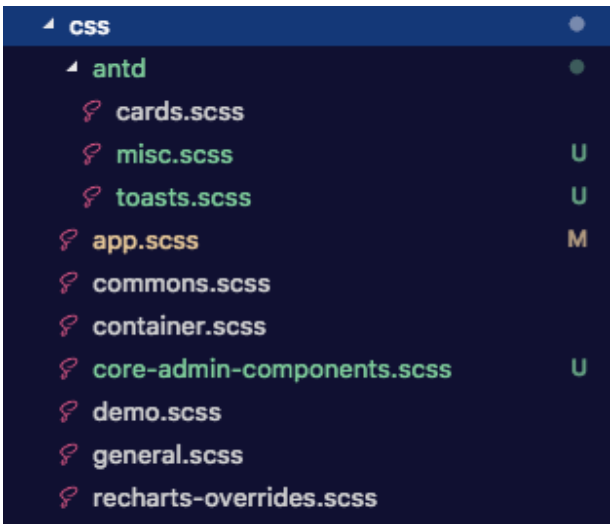
In order to deploy the application and run it in a **production** environment, take a look at the deployment section of this document.

2. Customizations

We have done our best to use a modular approach to develop this application, and that means that you should be able to isolate/distinguish certain functions/pages that you need to customize easily!

2.1 Styles

All the styles are inside the **“css”** directory:



The names of the files should be straightforward in helping you figure out that for a certain style modification which file you would need to look at.

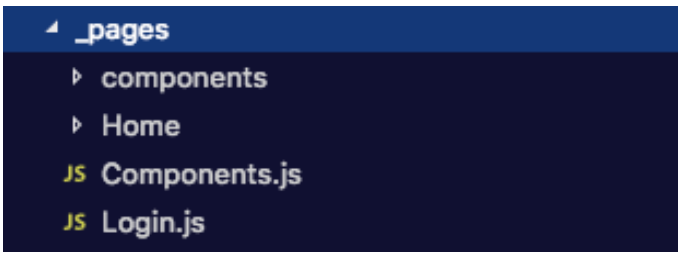
We would suggest using Inspect Element to figure out the classes of the elements you are modifying, and searching the “css” folder to find the appropriate file.

All the files in this directory are using Sass, which is an enhanced version of CSS that allows for things such as variables and functions, and would then compile back into regular CSS at build time.

If you’re not familiar with Sass, don’t worry! You can still write regular CSS code inside those files.

2.2 Pages

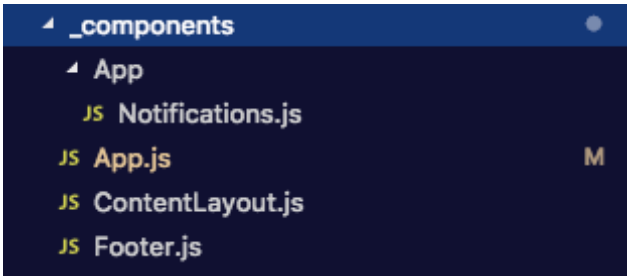
All of the pages in the application can be found inside the "_pages" directory. They are all React components and follow the same structure.



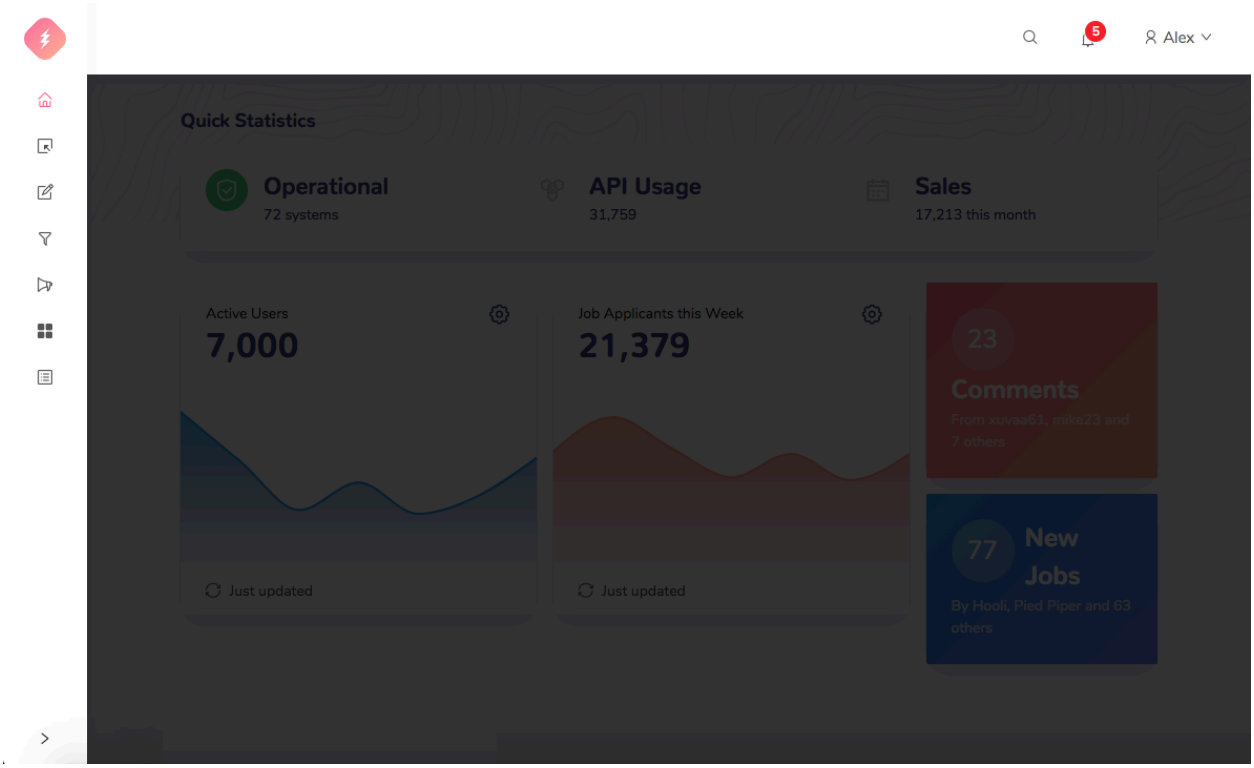
- At the top of every file, we have the import statements, that import any **dependencies/libraries/data** that are needed for that component.
- Right below the imports we would define any sort of data or functions outside of the component that the component would use.
- Then we are defining the React component, which usually include a state, the internal methods of the component, and the JSX
- At the end of each file we are exporting the component, and in some files we are applying the Redux content to the component, in order to be able to use/update the global state inside the component.

2.3 Components

Inside the "_components" directory, you will find components of the application that are not a page of their own, but can be included inside a page, and are reusable across multiple pages (e.g. footer, header).



Quick hint: The outer container of the application is defined in ***App.js***, and can easily be modified in there.



2.4 MISC

Redux-related logic is mostly inside the "__actions" and "__reducers" directory. To learn more about actions and reducers in redux, read **this**.

Bootstrapping of the application

The index.js file in the main directory of the application contains the bootstrapping/initialization logic for the app, as well some redux configurations.

"_data"

This directory contains the mock data files being used across the application. Usually in a production environment you would no longer include any mock data, and would replace those with real API requests, Database queries etc.

The ***Tools.js*** file will contain functions that are commonly used across multiple components. We would strongly recommend the continuous use of the modular approach for you!

3. Deployment

Now you are at a stage where you have developed your customizations into this application, removed the mock data, and have gotten your app to a state where you believe real users can make use of.

There are many ways you can deploy a React application; we would recommend using a tool called Netlify.

3.1) Push your app to a Github repository

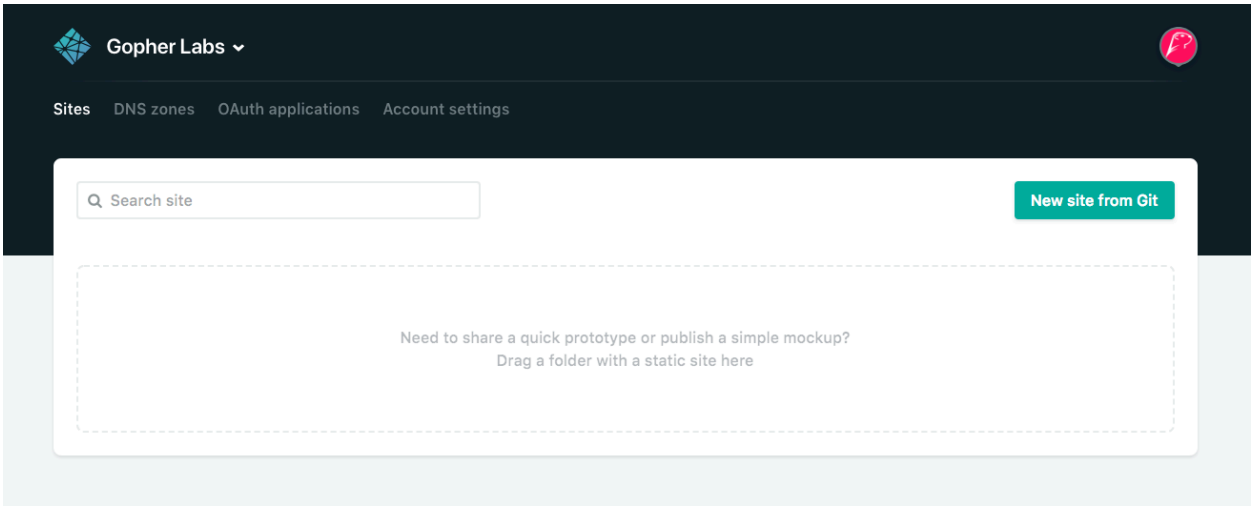
Learn here

We would recommend using Git Version Control, which can help you track your application's progress, as well as the ability to recover previous states of the application.

Not to mention that with the Github option, with any new updates to the repository, Netlify will rebuild the app and push it online automatically!

3.2) Create an account and login

This is the screen you would see after logging in.

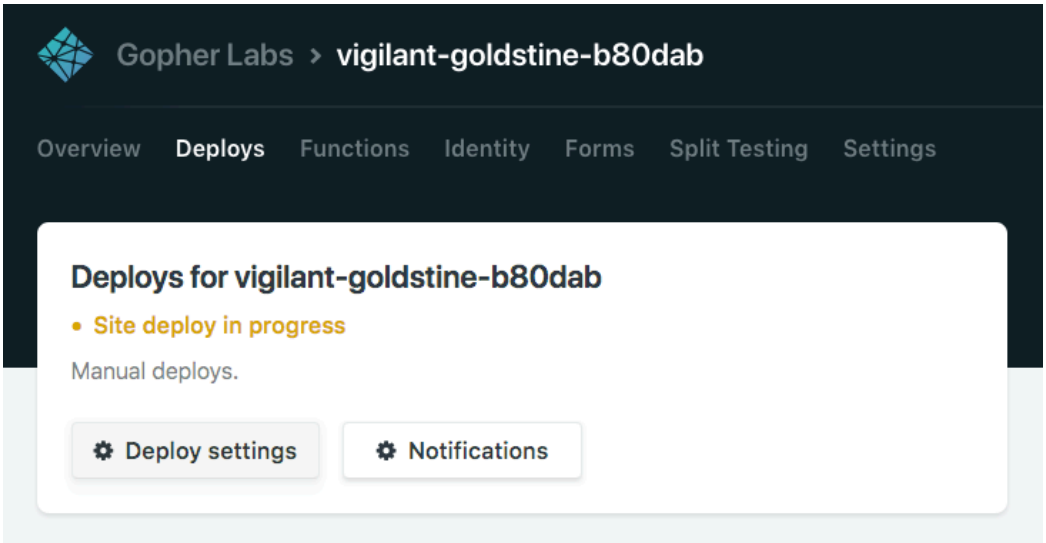


3.3) Upload the application

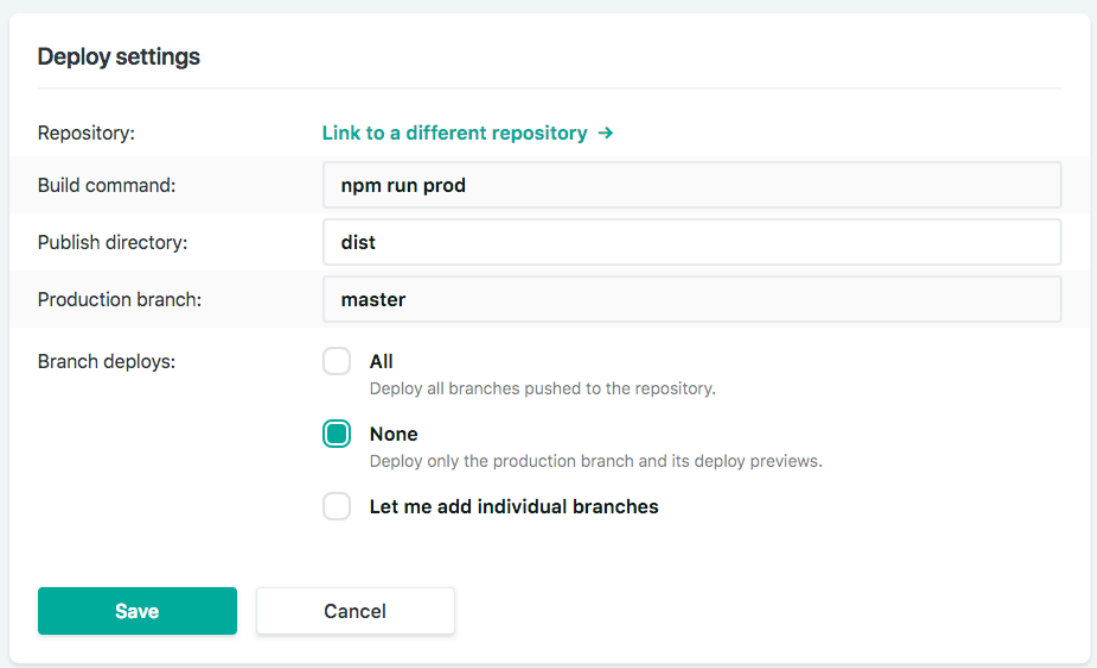
Click on “*New site from Git*”. Click on the Github option, login to your account, find and select your repo.

3.3) Apply deploy settings

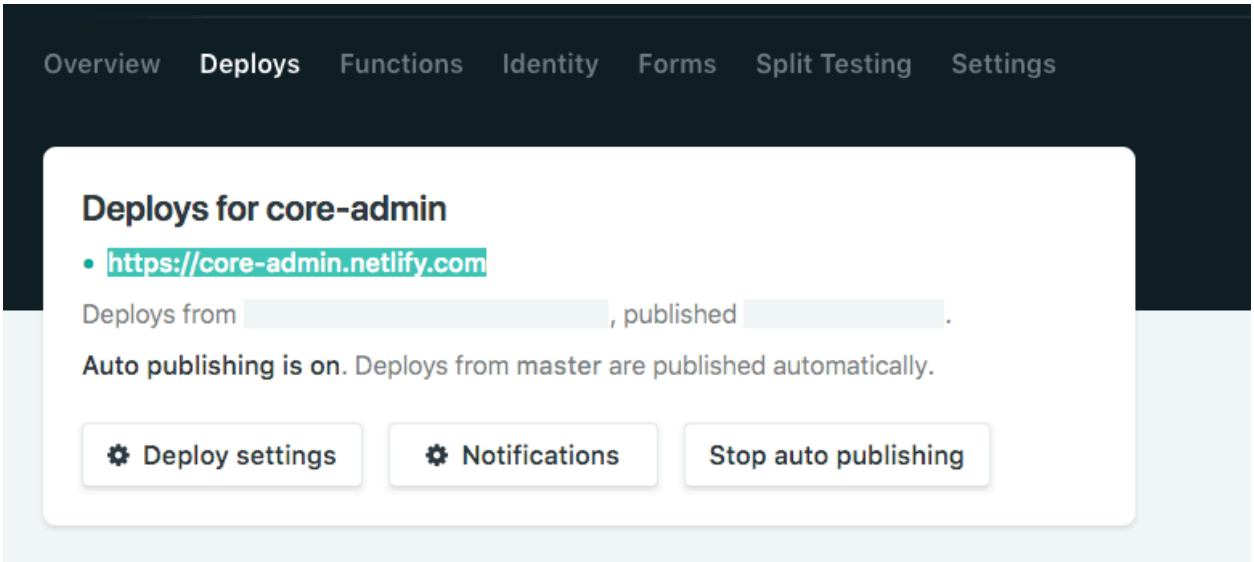
From the “Deploy” tab, click on “Deploy Settings”



Apply these settings and press “Save”

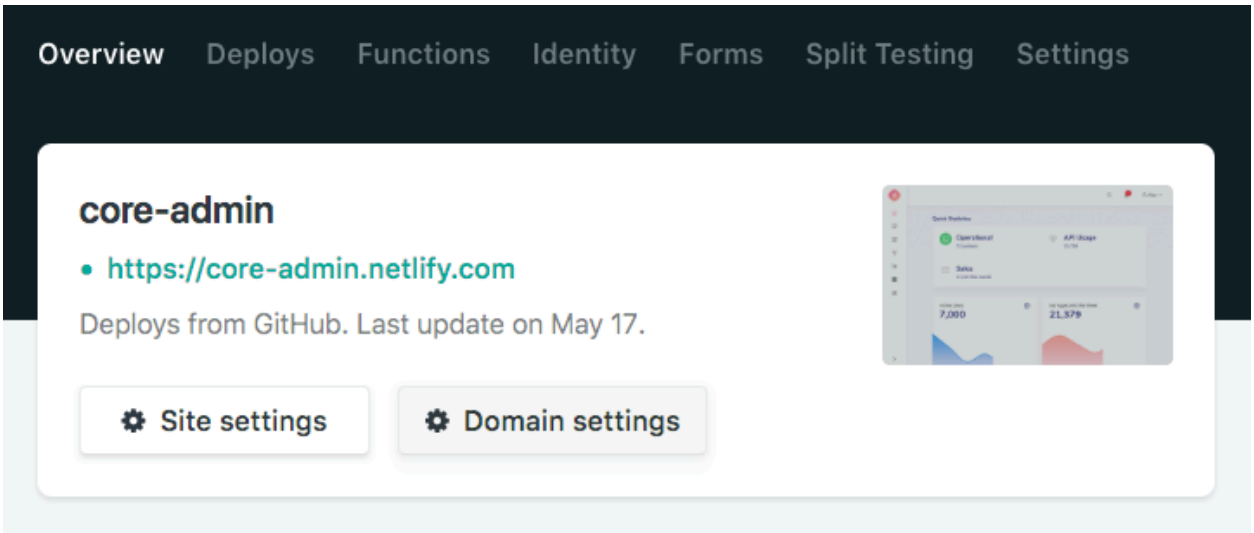


3.4) Successful Deployment



Once a deploy process is completed, you will see a URL, in which your application will permanently be available.

You also have the choice to connect the site to your own custom domain, and receive free SSL (Click on “Domain Settings”):



If your build is unsuccessful, you can take a look at the build log to find the error. Try to implement a fix, and make another push to your Github repo. Netlify will pick it up and rebuild: