# 10 ways to deploy a React app for free

August 17, 2020 · 14 min read

It's time that you took your applications out of development and into production. The process of deploying an application built on top of a framework such as React, Vue, or Angular is much different from that of deploying a site built with HTML, CSS, and JavaScript.

In this tutorial, we'll demonstrate how to deploy a React application in 10 different ways. All the services described in this post are completely free with no hidden credit card requirements.

Let's get started!

## 1. Vercel

Vercel, formerly known as ZEIT, is a revolutionary serverless deployment service designed for React, Angular, Vue, etc. You can easily import projects from GitLab or Bitbucket with Vercel. Automatic SSL is one of the many cool features it offers.

To deploy Vercel, create a new account. You can quickly login using OAuth.

After a successful login, the dashboard screen will appear. You can use either this dashboard or Vercel CLI and deploy it from the terminal. We'll discuss both methods in more detail.

### Dashboard

To deploy using the Vercel dashboard, integrate GitHub, GitLab, or Bitbucket, wherever your React application is stored. Click "Import Project" on your panel.

You can opt either to import the project from the Git repository or use a template, which is another excellent feature of Vercel.

Click "Continue" under "From Git Repository."

You may have integrated GitHub, GitLab, or Bitbucket; for the purpose of this tutorial, we'll assume you used GitHub. At any rate, you'll have the option to import projects from GitHub, GitLab, or Bitbucket.

Click "Import Project from GitHub." If this is your first time using Vercel, you'll see a screen like this.

Click "Install Now For GitHub." You'll be prompted to install Vercel for GitHub. Save the setting for GitHub, navigate back to the import Git page, and you'll see that your GitHub is now connected. Click "Import Project from GitHub."

You'll be prompted to choose which project you want to deploy. Choose the React project and click "Import." Next, you'll be prompted to write a project name. Leave it as the default and click "Continue." When asked for the root directory, choose accordingly and click "Continue."

This step is important. If you've initialized your React project using `create-react-app`, Vercel will autodetect it and choose a suitable configuration on its own. Leave the default configuration and click "Deploy."

If your React app was not initialized using `create-react-app`, you'l be asked for the configuration. Using the above configuration as an example, fill all the configuration fields and click "Deploy."

Your React application will now be deployed within a matter of seconds with two to three preview links.

# Vercel CLI

The first step is to install vercel globally.

```
npm i -g vercel
// or
yarn global add vercel
```

Once vercel is installed, run the following command.

```
vercel login
```

You'll be prompted to enter the email with which you registered on Vercel. After submitting that, you'll receive an email to verify your login.

Next, go to your project root directory and run the following command.

```
vercel
```

You'll be prompted to answer a few questions. First, confirm that this is the project you want to deploy.

```
? Set up and deploy "path to your project"? [Y/n] y
```

Next, you'll be asked in which account to deploy this app. It'll give you a default option; just hit enter.

After this, you'll be asked whether to link this to the existing project. Answer `N`.

```
? Which scope do you want to deploy to? Your Vercel Account
? Link to existing project? [y/N] n
```

The next step is to name your project and specify the path. Since we're already in the project directory, it'll be same as the default option, `./`.

```
? What's your project's name? project-name
? In which directory is your code located? ./
```

It will autodetect whether your project was initialized using `create-react-app` and configure the settings accordingly. Otherwise, it will ask you to set them. Answer `No` if asked to override the settings.

```
Auto-detected Project Settings (Create React App):
- Build Command: `npm run build` or `react-scripts build`
- Output Directory: build
- Development Command: react-scripts start
? Want to override the settings? [y/N] n
```

Your project will be deployed.

```
🔍  Deployed to production. Run `vercel --prod` to overwrite later
(https://vercel.link/2F).
```

# 2. Firebase

Firebase is an entire platform that you can use to develop and scale your application. Along with hosting, it offers myriad other services, including authentication, Cloud Firestore, cloud functions, and more.

If you haven't already, create an account on Firebase and then create a new project.

Install Firebase CLI globally. Having it installed globally makes it easier to use in different projects.

```
npm install -g firebase-tools
```

Login with your Firebase/Google account.

```
firebase login
```

You'll be prompted with a URL in the terminal that will open in the browser to verify. After giving the necessary permissions, you'll see a successful login message.

Next, go to your project root directory and run the following command to initialize a Firebase project.

```
firebase init
```

You'll be asked to confirm. Reply `Yes`.

```
 ? Are you ready to proceed? Yes
```

Choose the hosting option.

```
? Which Firebase CLI features do you want to setup for this folder? Press
Space to select features, then Enter to confirm your choices.
 ◯ Database: Deploy Firebase Realtime Database Rules
 ◯ Firestore: Deploy rules and create indexes for Firestore
 ◯ Functions: Configure and deploy Cloud Functions
❯◯ Hosting: Configure and deploy Firebase Hosting sites
 ◯ Storage: Deploy Cloud Storage security rules
```

When asked to choose the Firebase project associated with your application, since you already created a project in the first step, choose `Use an existing project`. Otherwise, you can select `Create a new project`.

```
=== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? Please select an option: (Use arrow keys)
> Use an existing project
  Create a new project
  Add Firebase to an existing Google Cloud Platform project
  Don't set up a default project
```

If you choose to `Create a new project`, you'll be asked to provide a unique project ID.

Lastly, change the default `public` folder to `build` for your `create-react-app` project. If you've not initialized the project with `create-react-app`, choose the appropriate build folder.

```
=== Hosting Setup

Your public directory is the folder (relative to your project directory)
that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? build
? Configure as a single-page app (rewrite all urls to /index.html)? Yes
```

If you've already built the project, you'll be asked whether to overwrite or not. Answer `No`.

```
? File public/index.html already exists. Overwrite? No
```

With this, `firebase init` is complete.

Before proceeding to the next step, build your React project.

```
npm run build
```

The next and final step is to deploy the project. Run the following command.

```
firebase deploy
```

Once the process completes, you'll see the deployed links in the terminal.

```
+  Deploy complete!

Project Console: https://console.firebase.google.com/project/react-project/overview
Hosting URL: https://react-project.web.app
```

# 3. Netlify

Netlify is one of the most popular services out there for web deployment. It easily imports projects from GitHub, GitLab, and Bitbucket and is widely used with JAMstack. One cool feature is that it creates a random name for every project, and the names are quite catchy.

To get started, create a Netlify account if you haven't already.

Like Vercel, you can choose to deploy your app either through Netlify Dashboard or Netlify CLI.

## Netlify drag-and-drop

One of the coolest features that Netlify offers is the ability to drag and drop your site folder and deploy your app like magic.

For your React app, you'll have to drag and drop the `build` folder on Netlify Dashboard. Run `npm run build` beforehand to deploy the latest build.

You can also connect GitHub, GitLab, or Bitbucket, depending on where your project is stored.

Choose the project repository that you need to deploy.

Once you've selected the project, the final step is the configuration, which Netlify will autodetect if the project is initialized with `create-react-app`.

Click "Deploy site" and your app will be deployed.

## Netlify CLI

If you prefer deploying apps through the terminal, here are the steps to do so with Netlify CLI.

To deploy the latest build, run `npm run build` beforehand.

Next, install `netlify-cli` globally.

```
npm install netlify-cli -g
```

In your project root directory, run the following command.

```
netlify deploy
```

You might be prompted to grant access to Netlify CLI. Click "Authorize."

After successful authorization, you'll see a successful login message in the terminal.

```
You are now logged into your Netlify account!
```

The following steps will guide you through the prompts that you'll encounter in the terminal.

First, you'll be asked to link this project to a site, since this is the first time you're deploying this app. Select `Create configure a new site`.

```
This folder isn't linked to a site yet
? What would you like to do?
```

```
Link this directory to an existing site
Create configure a new site
```

Then you'll be asked for the `Team`. Unless you're already using Netlify in your local machine, chances are you will see only one option with your name, select it.

```
? Team:
```

```
ASHUTOSH KUMAR SINGH's team
```

The next prompt is `Site name`. This is an optional field, as you can see below. If you already have a name in mind, type that and hit enter. If you leave this blank, Netlify will give this site a random name that you can change later.

```
Choose a unique site name (e.g. netlify-thinks-lelouchB-is-
great.netlify.app) or leave it blank for a random name. You can update the
site name later.
? Site name (optional):
```

After this step, your site will be created and you'll be asked for a `Publish directory`. Type `build` here.

```
Please provide a publish directory

? Publish directory build
```

With this, your site will be published and you will be provided a draft URL.

```
Deploying to draft URL...
✓ Finished hashing 19 files
✓ CDN requesting 10 files
✓ Finished uploading 10 assets
✓ Draft deploy is live!

Logs:https://app.netlify.com/sites/serene-fermi-
6d50a8/deploys/5f1194c3b903cadb238eabb4
Website Draft URL: https://5f1194c3b903cadb238eabb4--serene-fermi-
6d50a8.netlify.app

If everything looks good on your draft URL, deploy it to your main site URL
with the --prod flag.
netlify deploy --prod
```

Go to this draft URL. If everything checks out, you can deploy the website to the main site URL.

Run the following command to deploy to production.

```
netlify deploy --prod
```

It will ask for the `Publish directory` one final time. Type `build` and hit enter. You'll be provided with two URLs.

```
Unique Deploy URL: https://5f11977085ef8fafe7535ff9--serene-fermi-6d50a8.netlify.app
Website URL:       https://serene-fermi-6d50a8.netlify.app
```

The difference between these two URLs is that `Unique Deploy URL` points to a specific version of your application. For example, if you make a change in your application and deploy again, you'll get another `Unique Deploy URL` that is specific for that change. Your `Website URL` is the main URL, which corresponds to the latest version of your application.

You might encounter a 404 error if your application uses a router, such as React Router.

In your `build` folder, create a new file called `_redirects` and add the following to it.

```
/*    /index.html  200
```

You'llneed to redeploy your application. This question posted to the Netlify community might help.

# 4. GitHub Pages

GitHub Pages is one of the fastest and most widely used methods for beginners to deploy websites. It's easier to maintain than many other tools described in this guide. With GitHub actions, you can trigger automatic deployments, configure CI/CD, and much more.

Create a GitHub account if you haven't already, then create a repository for your application.

In your terminal, initialize the local directory as a Git repository, commit all the changes, and push it to remote by running the following command in the project root.

```
git init
git add .
git commit -m "initial commit"
git remote add origin .git
git push -u origin master
```

You'll get this repository URL when you create a new repository.

With this, your project will be pushed to GitHub.

In your project's `package.json` add a `homepage` field, like this:

```
"homepage": "https://myusername.github.io/my-app"
```

`myusername` is your GitHub username and `my-app` is your repository's name.

Next, install `gh-pages` in your project.

```
npm install --save gh-pages
```

In your `package.json`, add the following scripts.

```
"scripts": {
+   "predeploy": "npm run build",
+   "deploy": "gh-pages -d build",
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
}
```

`predeploy` and `deploy` are the only additions to the scrips. `predeploy` will automatically run before `deploy` and make sure the latest build of the application is deployed.

Run the following command to deploy the application.

```
npm run deploy
```

This command will create a new branch named `gh-pages` in your project's GitHub repository. You may need to set a source under the GitHub Pages option in your repository's settings to `gh-pages branch`.

# 5. Heroku

Heroku is a cloud application platform that has attracted a large number of developers since its launch in 2007. One reason for this is that it supports most programming languages, including the likes of Go, Node.js, Clojure, and more.

## Heroku CLI

Like most of the other services we've discussed thus far, start by creating a free account on Heroku.

Install `heroku-cli` globally by running the following command.

```
npm install -g heroku
```

You can read about other installation methods in the official docs.

Log into `heroku-cli`.

```
heroku login
```

You'll be prompted to log into your account in the browser. Click "Log In."

We'll use Heroku Buildpack for create-react-app for quick deployment. Below is all the code you'll need; you can copy/paste and deploy in one step.

```
git init
heroku create -b https://github.com/mars/create-react-app-buildpack.git
git add .
git commit -m "react-create-app on Heroku"
git push heroku master
heroku open
```

This might be the fastest way to deploy via terminal because yiuy don't have to answer any prompts as with other CLIs.

## Heroku dashboard

You can aso deploy with Heroku via the dashboard.

Make sure your project is stored in a GitHub repository.

Go to your Heroku dashboard, click "New," and then click "Create new app."

You'll be prompted to give your project a name. Type your application name and click "Create app."

After creating an app, sync your GitHub repository. You'll see something like this on your app dashboard.

Once you've successfully connected your GitHub to Heroku, you can search for the project repository and deploy it.

Select your project from the list of repositories.

You'll have two choices: manual deploy or automatic deploy. For the purpose of this tutorial, we'll go with manual deploy.

Click "Deploy Branch" under "Manual Deploy" and your application will be deployed once the build process completes.

# 6. Surge

Surge is one of the fastest ways to deploy frontend projects. Compared to other CLIs, it requires much less configuration, and you can create a Surge account directly from the terminal when using it for the first time.

To deploy the latest build of the project, run the following command in the project root directory.

```
npm run build
```

To install Surge CLI globally:

```
npm install -g surge
```

Run `surge` inside the `build` folder.

```
cd build
surge
```

Follow the prompts. First, you'll be asked for an email and password.

```
Welcome to surge! (surge.sh)
Login (or create surge account) by entering email & password.
```

```
      email: admin@ashusingh.me
      password:
```

Before filling in any other prompts, it's a good idea to confirm your Surge account.

In the terminal, you'll be asked to confirm your project directory. Hit enter.

You'll see the domain of your project. Again, hit enter.

```
project: D:\code\react-example-deploy\build\
domain: tremendous-person.surge.sh
```

This will publish your application.

```
Success! - Published to outstanding-scent.surge.sh
```

You might want to rename `index.html` to `200.html` in your `build` folder before surging to support the HTML5 `pushState API`.

If you run into an aborted error, try to run the `surge` command again.

# 7. Render

Render, the winner of Startup Battlefield at Disrupt SF 2019, founded in 2018, is a new and rising cloud platform.

While hosting sites on Render is free, you need to pay for other services such as databases, cron jobs, Docker containers, etc. Both Heroku and Render classify as platform-as-a-service.

Before you deploy, make sure your project is stored in a GitHub repository.

Create an account on Render.

Next, click "New Web Service" on the dashboard.

If this is your first time using Render, you'll need to connect your GitHub or GitLab, wherever your repository is stored.

Once connected, search for your project repository and select it. After this, you'll need to give your web service a unique name. Render will auto-detect if your application is bootstrapped with `create-react-app` and fill the configuration accordingly.

Click "Create Web Service," after which your project will be deployed.

## Client-side routing

If your project uses something like React router, you'll need to redirect all routing requests to `index.html`. Go to the "Redirects/Rewrites" tab for your service, add a rule, and save the changes.

# 8. Roast

Roast is another web hosting platform that you can use to deploy single-page applications (SPAs).

It's very simple to configure and use. The Roast docs are clear and concise.

Start by creating a Roast account.

Install Roast CLI globally.

```
npm install -g roast
```

To deploy the latest build, run the following command in your project directory.

```
npm run build
```

Run the following command to deploy with Roast.

```
roast deploy
```

After this, you'll be authenticated.

In the terminal, you'll be asked to create a new site. Answer `Yes`. Then, you'll be asked for `Path to deploy`, which will be the `build` folder.

```
? No site ID specified, create a new site Yes ? Path to deploy? build
```

Your application is now deployed.

```
🔥 Roasting deploy from folder D:\code\react-example-deploy\build

☕ Deploy roasted!

✨ https://witty-surprise-3068.roast.io
```

To return `index.html` instead of a 404, create a `_redirects` file inside `build` with the following code.

```
/* /index.html 200
```

Once you've added this file, you'll need to redeploy your app with `roast deploy`.

You can now see this redirect in your Roast dashboard.

# 9. Moovweb XDN

Moovweb XDN is an all-in-one platform to develop, deploy, preview, experiment on, monitor, and run your application.

Moovweb's free tier should be more than sufficient for your side projects.

Start by creating a Moovweb XDN account.

XDN only supports Node.js version 12 and higher.

Check your Node version by running the following command.

```
node --version
```

Install XDN CLI globally.

```
npm i -g @xdn/cli
```

After installing the XDN CLI, run the following command to authenticate yourself.

```
xdn login
```

You'll see the following output in the terminal.

```
Authenticating user!
```

After this, You will be redirected to the Moovweb page and see the following message.

To deploy the latest build, run the following command in your project directory.

```
npm run build
```

In your project's root directory, run the following command in the terminal.

```
xdn init
```

This command will install all the dependencies and files required by XDN in your project. After a few minutes, you will see the following message in your terminal.

```
✔ installing @xdn/core, @xdn/cli, @xdn/prefetch, @xdn/devtools... done.
> routes.js not found, creating...
> xdn.config.js not found, creating...

Added the following scripts:

xdn - Run xdn commands using the local version of the XDN CLI.
xdn:start - Simulate your app on the XDN locally.
xdn:build - Build your app for deployment on the XDN.
xdn:deploy - Build and deploy your app on the XDN.

🎉 Your app is now ready to deploy on the MOOVWEB XDN!
```

You'll notice some new files will be created after the `xdn init` command. Modify `routes.js` in the root like this to serve the `build` directory.

```
// routes.js
const { Router } = require('@xdn/core/router')
const ONE_HOUR = 60 * 60
const ONE_DAY = 24 * ONE_HOUR
const ONE_YEAR = 365 * ONE_DAY
const edgeOnly = {
  browser: false,
  edge: { maxAgeSeconds: ONE_YEAR },
}
const edgeAndBrowser = {
  browser: { maxAgeSeconds: ONE_YEAR },
  edge: { maxAgeSeconds: ONE_YEAR },
}
module.exports = new Router()
  .prerender([{ path: '/' }])
  // js and css assets are hashed and can be far-future cached in the
browser
  .get('/static/:path*', ({ cache, serveStatic }) => {
    cache(edgeAndBrowser)
    serveStatic('build/static/:path*')
```

To deploy your React app, run the following command.

```
xdn deploy
```

After a few minutes, your site will be deployed.

You can read more about deploying a React app with Moovweb XDN.

# 10. GitLab Pages

You can also use GitLab pages to publish static sites directly from the repository in GitLab. With GitLab pages, you can easily connect your custom domains and TLS certificates.

Create a GitLab account if you haven't already.

After creating an account, you will see the following options to add your React app.

If your project is already present on GitHub, then select **Import project** and import the project. Otherwise, select **Create blank project** and create an empty project.

Give your project a name and click **Create project**.

On the next page, you'll see various commands to add your React project, configured according to your credentials.

In your terminal, initialize the local directory as a Git repository, commit all the changes, and push it to remote by running the following command in the project root.

```
git init
git remote add origin git@gitlab.com:<username/project-name>.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Be sure to replace `<username/project-name>` with your username and project name. For example:

```
git remote add origin git@gitlab.com:lelouchB/react-material-ui.git
```

In your project's `package.json`, add a `homepage` field.

```
"homepage": "https://myusername.gitlab.io/my-app"
```

`myusername` is your username and `my-app` is the name of your project. For example:

```
"homepage":"https://lelouchB.gitlab.io/react-material-ui/"
```

Create a new file named `.gitlab-ci.yml` and add the following code to it.

```
image: node:10 # change to match your node version

cache:
  paths:
    - node_modules/

before_script:
  - rm -rf build
  - npm install

pages:
  stage: deploy
  script:
    - npm run build
    - rm -rf public
    - mv build public
  artifacts:
    paths:
      - public
  only:
```

After committing this file, a pipeline will start. This will deploy your React app to GitLab pages. You can see this pipeline under `CI/CD` tab.

After the pipeline has passed successfully, it may take up to 30 minutes before the site is available after the first deployment, so be patient.

You can manage your React app under the `Settings/Pages` tab.

Your app will be deployed to the same URL, as mentioned in the `homepage` field in `package.json`.

## Some other platforms worth checking out

- DigitalOcean App Platform — You can deploy three static sites on Starter or the free tier
- StormKit — Single app deployment is available on the free tier
- Platform.sh — Offers a 30-day free trial
- Bip — Offers a seven-day free trial

## Bonus freebie: once you deploy, monitor React apps for free with LogRocket.

Debugging React applications can be difficult, especially when users experience issues that are difficult to reproduce. If you're interested in monitoring and tracking Redux state, automatically surfacing JavaScript errors, and tracking slow network requests and component load time, try LogRocket.

LogRocket is like a DVR for web apps, recording literally everything that happens on your React app. Instead of guessing why problems happen, you can aggregate and report on what state your application was in when an issue occurred. LogRocket also monitors your app's performance, reporting with metrics like client CPU load, client memory usage, and more.

The LogRocket Redux middleware package adds an extra layer of visibility into your user sessions. LogRocket logs all actions and state from your Redux stores.

Modernize how you debug your React apps — start monitoring for free.

## Conclusion

Now that we've discussed ten different ways to deploy React, you should try out as many as you can to determine which best aligns with your deployment requirements. After all, they're free to use.

For a good next step, try to add custom domains to your deployed application. It's good to have a distinctive domain for projects, which have different ways to add a custom domain — some easy, some relatively complex.

Until next time, happy coding!

**Share this:**

Twitter Reddit LinkedIn Facebook

Ashutosh Singh Follow

Ashutosh is a JavaScript developer and a technical writer. He writes about the fundamentals of JavaScript, Node.js, React, Next, Vue, and tutorials on building projects.

#react

## 6 Replies to “10 ways to deploy a React app for free”

**Ajitfawade** Says:
August 25, 2020 at 4:39 pm

Reply

Thanks for such a great article!!!

Any reason why AWS S3 is not mentioned in the list? We can use the free tier for an year.

**Ashutosh K Singh** Says:
August 26, 2020 at 8:35 am

Reply

Thanks. Glad you like it. 🙂
I guess I didn’t include services like AWS and Azure because the complete step-by-step process of configuring and deploying a React app with AWS or Azure is too big to include in this article. Separate articles for them will be great to explain everything adequately.

**Rg1107** Says:
September 18, 2020 at 8:00 am

Reply

Thanks for sharing above article.

**Eric** Says:
September 20, 2020 at 2:15 am

Reply

Thanks for this, it has helped me beat my capstone deadline. So thankful!

**Ashutosh K Singh** Says:
September 21, 2020 at 8:30 am

Reply

Thanks! Glad you liked it 🙂

**Sneha Herle** Says:
November 1, 2020 at 1:59 am

Reply

Thanks a lot. I used Surge to host my app. Honestly I had never heard of Surge until I read this article and I was having trouble hosting my app everywhere from Netlify, GitHub Pages, and Heroku! I this really helped me.

## Leave a Reply

Enter your comment here...