



Task

React - Deployment

[Visit our website](#)

Introduction

WELCOME TO THE REACT - DEPLOYMENT TASK!

There are various deployment methods, such as Docker containerisation and GitHub Pages for static site hosting. Today we'll look at [Render](#) for deploying your React app. Render is a great platform that streamlines the deployment and scaling of your applications. In this session, we'll provide a simple and user-friendly outline of using Render, covering signing up for a Render account and seamlessly deploying your app. Let's explore this powerful tool that offers efficient hosting solutions for your projects with ease and speed.

First, we will touch on GitHub which makes deploying to Render seamless.

WHAT IS GITHUB?

Git is the foundation of many services that work on version control. The most popular and widely used of them all is GitHub. GitHub is an online Git repository hosting service. GitHub offers all of the functionality of Git and a lot more. While Git is a command-line tool, GitHub provides a Web-based graphical interface. It provides access control and many features that assist with collaboration, such as wikis and basic task management tools for all projects.

GitHub is not just a project-hosting service, it is also a large social networking site for developers and programmers. Each user on GitHub has a profile, showing their past work and contributions that they have made to other projects. GitHub allows users to follow each other, subscribe to updates for projects, like them by giving them a star rating, etc.

Each project hosted on GitHub will have its own repository. Anyone can sign up for an account on GitHub and create their own repositories. They can then invite other GitHub users to collaborate on their project. You can even host static websites for free directly from your repository using [GitHub Pages](#)! If you do not have one already please **create a free GitHub account** by visiting <https://github.com/join>.

GITHUB AND YOUR DEVELOPER PORTFOLIO

Your [developer portfolio](#) (a collection of online programs that you have developed) allows you to demonstrate your skills rather than just telling people about them.

GitHub provides one of the most industry-recognised ways of sharing your code with others, including peers, prospective employers, or clients. A well-organised and documented GitHub repository can serve as a core component of a developer portfolio.

Even before seeing your work, prospective employers may also be impressed with the fact that you have experience in working with Git and Github.

SYNCING YOUR LOCAL AND REMOTE REPOSITORIES

Previously you learned how to create and add files to a local Git repository. Now you will learn how to **push** your repository from the command line to a remote GitHub repository. Have a look at this [excellent video tutorial](#) that explains how you can do exactly that.

Here is a summary of the command line prompts when you "push an existing repository from the command line":

```
git remote add origin https://github.com/[REPO-OWNER]/[REPO-NAME]
git branch -M main
git push -u origin main
```

ACCESS TOKENS

In GitHub, an Access Token is an alternative to using passwords for authentication when using the GitHub API or, in our case, the command line. The purpose of these tokens are to provide added security and define which types of actions can be performed based on the "scope" of that token.

We recommend using access tokens rather than a password for authentication as it gives you more granular control of your security settings.

Generate an access token by following these steps:

1. Login to **GitHub**.
2. Click on the drop down arrow next to your profile picture and click on "**Settings**".
3. In your settings window, scroll down and click on "**Developer settings**".
4. Next, click on the "**Personal access token**" drop down selection.

5. Click on “**Tokens (classic)**”.

You’ll notice there’s also the option to select “**Fine Grained Tokens**”. Both types of personal access tokens differ in terms of their capabilities and the level of access they grant to users.

Tokens (classic) provide full access to the account or organisation associated with the token, including the ability to read, write, and delete all types of resources, such as repositories, issues, and pull requests. They are designed for scripts and other automated processes that require full access to the account or organisation.

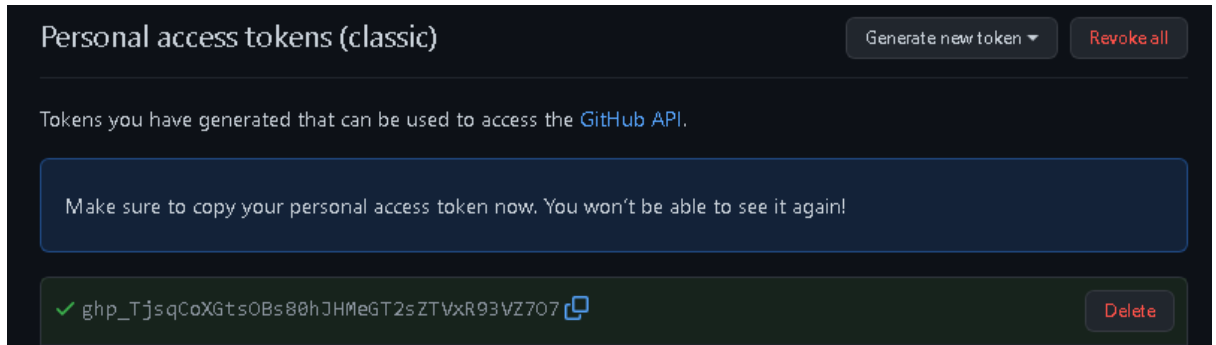
Fine Grained Tokens, on the other hand, are a newer type of personal access token that provides more control over the level of access granted to users, enabling access to be granted to specific resources without giving full access to the entire account or organisation.

Currently it’s simpler and faster to use a **Classic** token which is why we will be selecting this option instead of a Fine Grained Token.

6. Now click on the drop down selection that says “**Generate new token**”.
7. Click on “**Generate new token (classic)**”.
8. GitHub might ask you for your password or to verify your login with GitHub mobile depending on the security measures you’ve enabled.
9. Confirm your access by entering your password or using your preferred method.
10. You should now be able to start creating an **Access Token**. You can use the “Note” section to call the token anything you’d like. The expiration will set how long the token will be valid. Please note that after your token expires you will no longer be able to use it and will need to generate a new one!
11. You will need to select all the applicable scopes for your token. For example, if you’d like full control of the repository using this access key, simply ensure you check the “repo” checkbox.

12. Finally scroll down and click on the “Generate token” button at the bottom. The full access token will be displayed on the next screen.

Here is an example:



13. Please ensure you **copy** your access token! If you intend to reuse it multiple times while it's active, ensure you store it in a secure location such as a credential store or password manager!

Once you have your token copied, ensure you paste it as your “**password**” when prompted to back on your original Git request.

Entering your username and password in the command prompt

When it's time to push your local repository to your remote GitHub repository, you may occasionally be prompted for your **username** and **password** when performing specific actions. **Take note** that this is not always the case and that you may not be prompted to authenticate every time you push your repositories to a remote location.

Before moving ahead, the important thing to note here when pushing your repositories is that while your “**username**” in this context will need to match your actual “**username**” on GitHub, your “**password**” will need to be replaced with the **Access Token** that was generated.

There are two options available when entering your username and password / token credentials into the command prompt:

Option 1 - add the token directly into the URL:

```
git remote add origin
https://[TOKEN]@github.com/[REPO-OWNER]/[REPO-NAME]
git branch -M main
git push -u origin main
```

Option 2 - enter the username and password separately:

```
git push https://github.com/[REPO-OWNER]/[REPO-NAME]  
Username: <username>  
Password: <token>
```

You can also review alternative methods on the official GitHub documentation here:

<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>.

Now that you are familiar with GitHub let's move onto Render.

WHAT IS RENDER?

Much like GitHub Pages, Render is a Platform-as-a-Service (PaaS) provider that allows developers to host their applications online with minimal manual setup. One remarkable difference between the two is that Render is able to host much more than static sites. It can also be used to host Node, Python, Ruby, Go, Rust, and PHP applications, just to name a few.

In a nutshell, you can use Render to deploy just about any web-based app you want. You can even deploy a React application in less than a minute because the deployment is linked to your GitHub repo for the application!

A big advantage of Render is that static deployments are totally free of charge as long as you stay within the 100GB bandwidth a month.

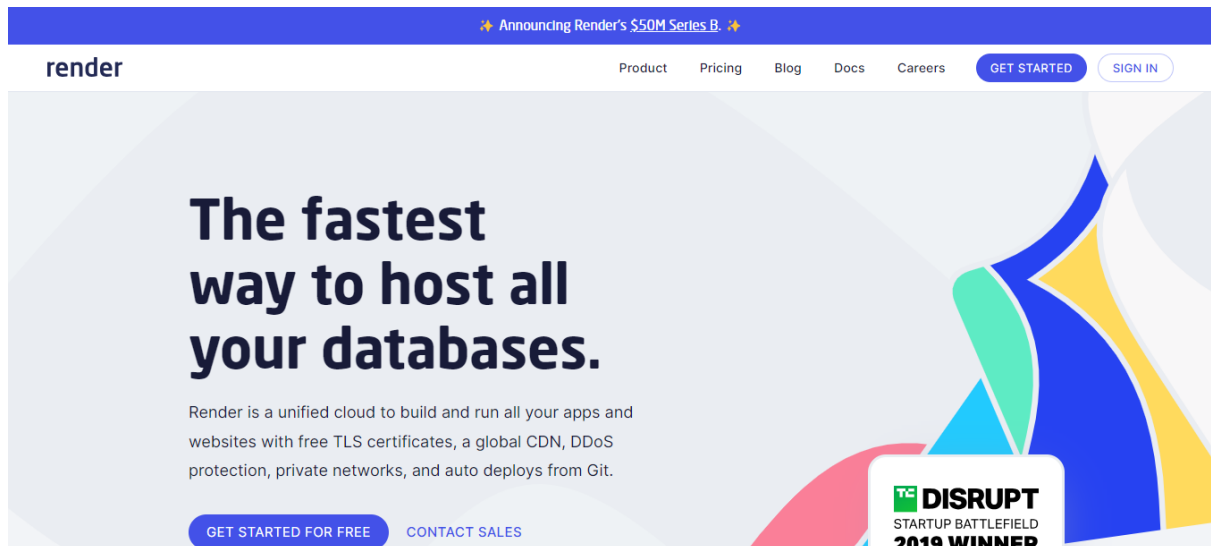
There are many PaaS alternatives to Render that work in much the same way, but there are pricing implications that must be considered. These include [Heroku](#), [Railway](#), [AWS](#), [Azure](#), and [Firebase](#). These providers typically allow a certain amount of free platform usage, and only start charging when you need complex features, or have many users. Some also require credit card details even for the free tiers. We **do not recommend** following the above route for this bootcamp.

SIGN UP FOR A RENDER ACCOUNT

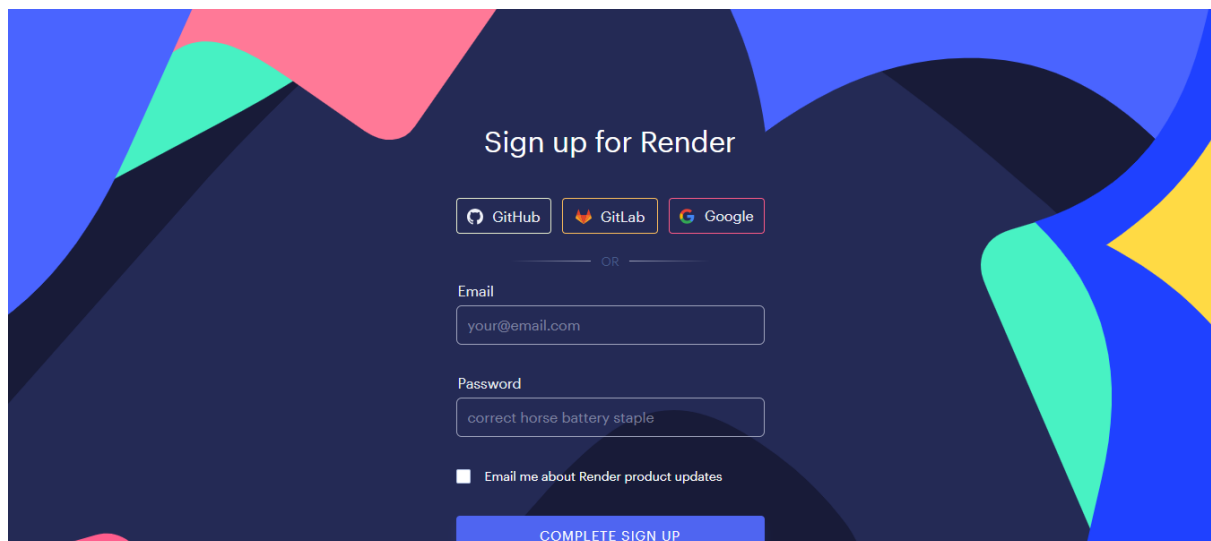
You are going to have to sign up for a Render account in order to be able to deploy your app via Render. Because deployment platforms can unexpectedly become subscription-based, the choice of whether to sign up for a Render account is

completely **optional** and not compulsory in any way. This only applies **if you wish to deploy** your application to the World Wide Web. If you would like to go ahead and do so, click on this [link](#) and follow the instructions below.


Click on the “Getting Started” button (top right-hand).



You will then be given options to log in with your GitHub or Google account. Please note that you should sign in with your **GitHub** account for seamless deployment and safety reasons.







Sign in to GitHub
to continue to Render

Username or email address

Password [Forgot password?](#)

Sign in

New to GitHub? [Create an account.](#)

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

BRIEF OUTLINE FOR USING RENDER

It is really very easy to deploy a React application with Render. It involves three steps:

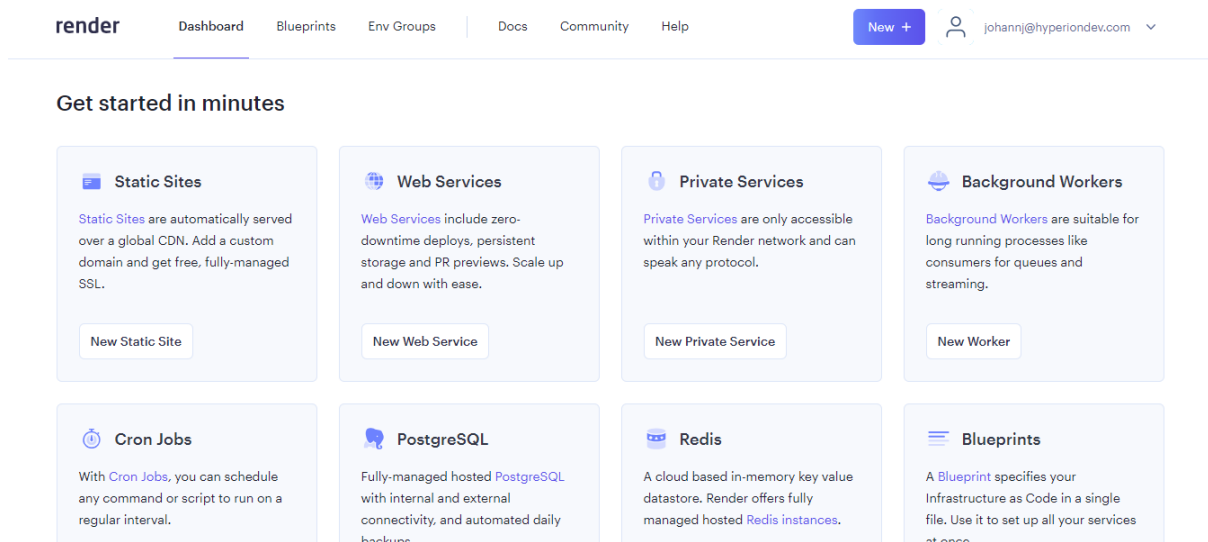
1. Make sure you are happy with your application's GitHub repository (i.e., it is finalised and up to date).
2. Create a new static site from the Render dashboard and give Render access to your GitHub repository.
3. Use the following commands during creation:
 - Build command: **yarn build**
 - Publish directory: **build**

And it is as easy as that! The application will be available on the Render URL as soon as the build finishes.

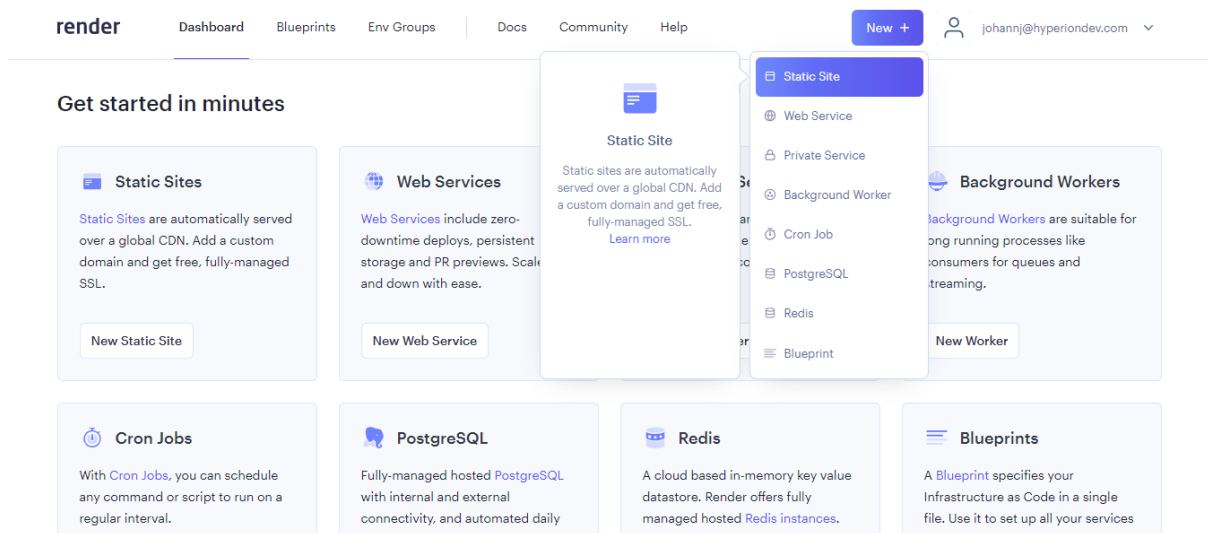
DEPLOYING YOUR APP TO RENDER

The detailed deployment process is as follows:

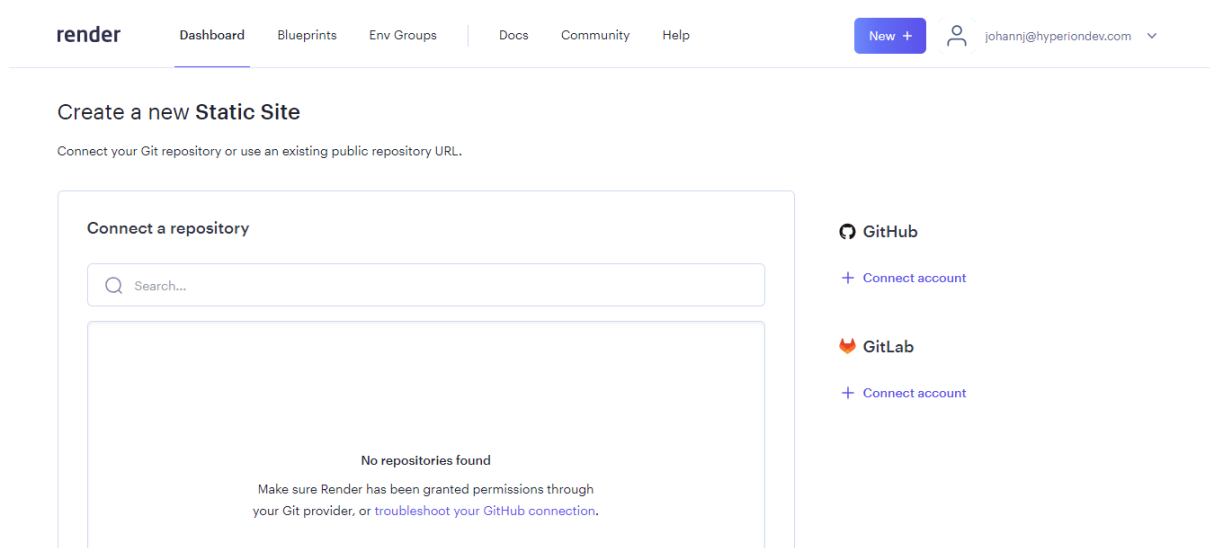
First, sign in to Render. Your Dashboard will appear.



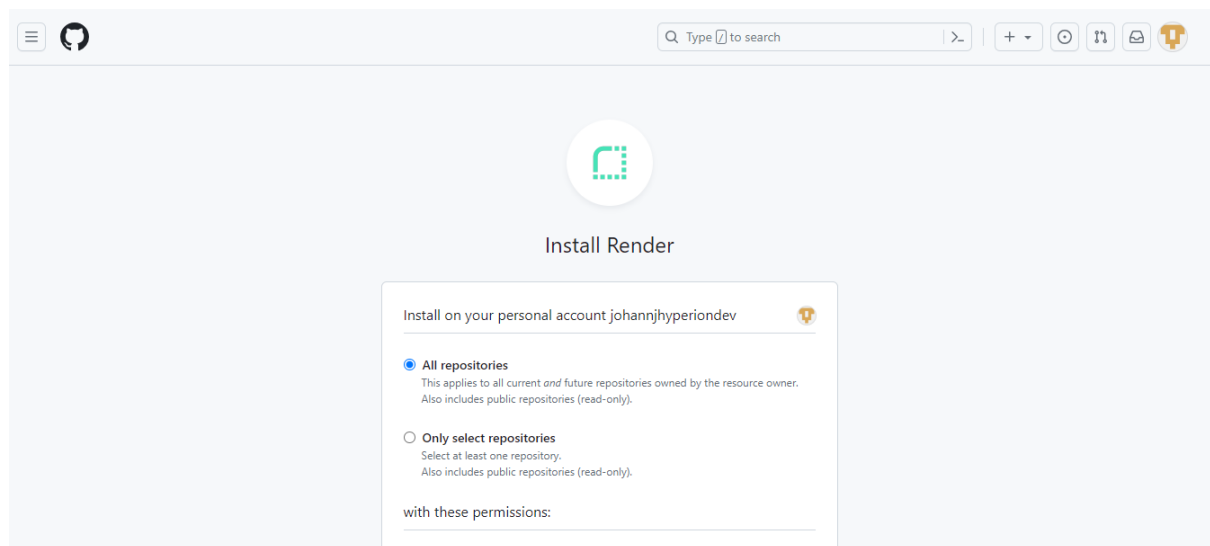
Click on the **New+** button (top right). Then click on the **Static Site** button:



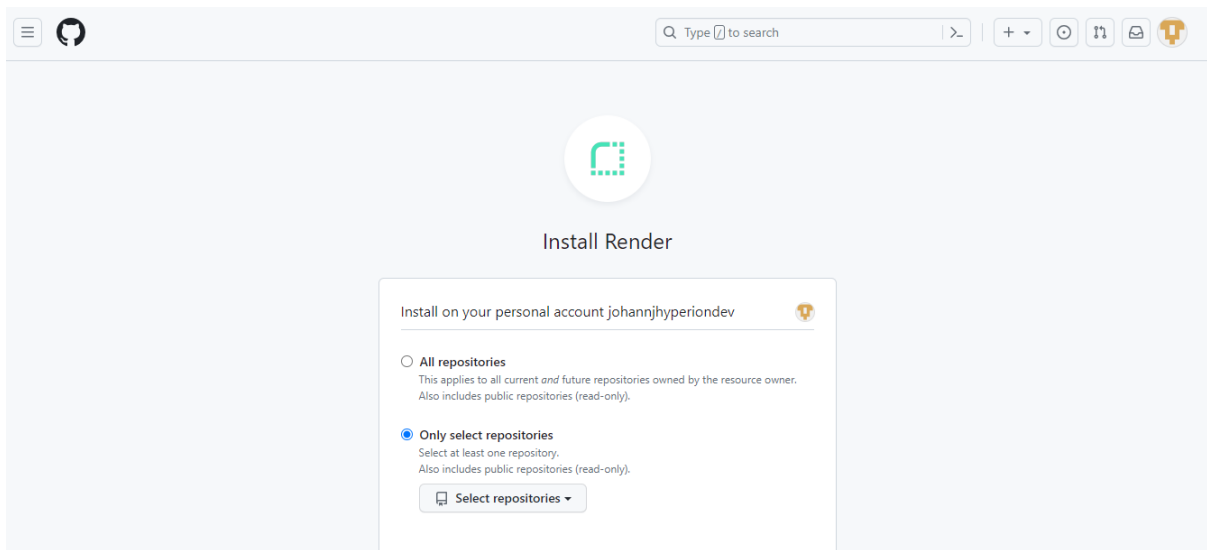
Under the GitHub icon/heading, click on **Configure account**:



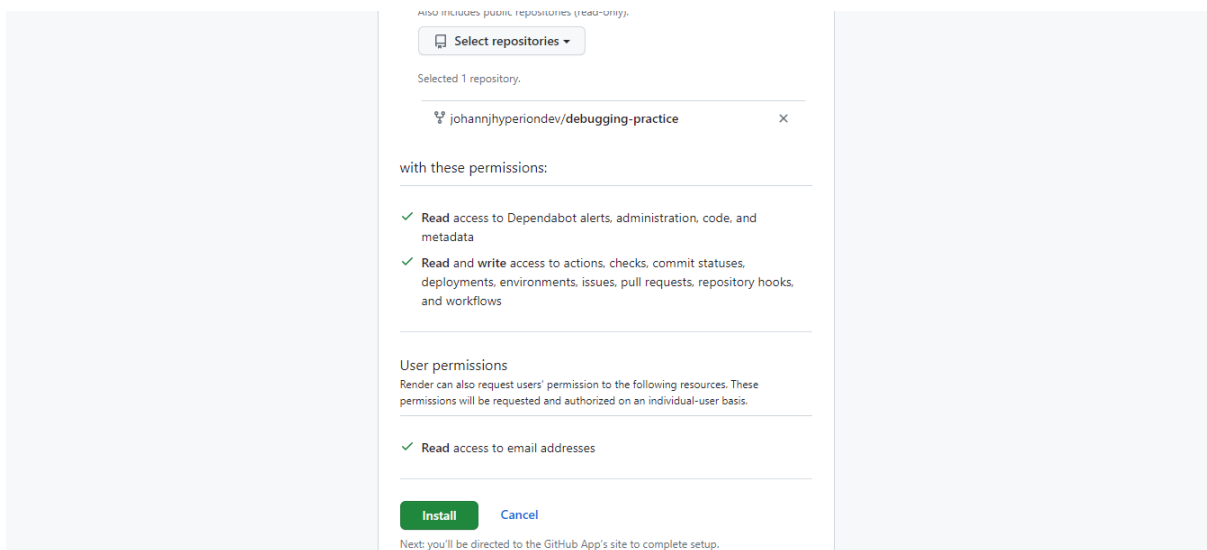
On the next page choose the **Only select repositories** radio button:



Then click on the **Select Repositories** button:



Then click on the green **install** button:



On the next page, add a name of your choice for your static site next to the “Name” label. Leave the “branch” label as it is. Enter **yarn build** next to the “Root Directory” label and **build** next to the “Publish directory” label:

render

DashboardBlueprintsEnv GroupsDocsCommunityHelp

New +

johannj@hyperiondev.com

You are deploying a static site for [johannjhyperiondev/demoproject](#).

Name
A unique name for your static site.

example-service-name

Branch
The repository branch used for your static site.

main

Root Directory Optional
Defaults to repository root. When you specify a [root directory](#) that is different from your repository root, Render runs all your commands in the [specified directory](#) and ignores changes outside the directory.

e.g. src

Build Command
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

\$

Publish directory
The [relative](#) path of the directory containing built assets to publish. Examples: [./](#), [./build](#), [dist](#) and [frontend/build](#).

e.g. build

Advanced

Next, click on the **Create Site** button:

render

DashboardBlueprintsEnv GroupsDocsCommunityHelp

New +

johannj@hyperiondev.com

Build Command
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

\$

Publish directory
The [relative](#) path of the directory containing built assets to publish. Examples: [./](#), [./build](#), [dist](#) and [frontend/build](#).

e.g. build

Advanced

Create Static Site

Feedback

Invite a Friend

Contact Support

It will now take some time for the build. When the build is finished, a **Live** button will appear and the URL will be displayed towards the top left of the screen:

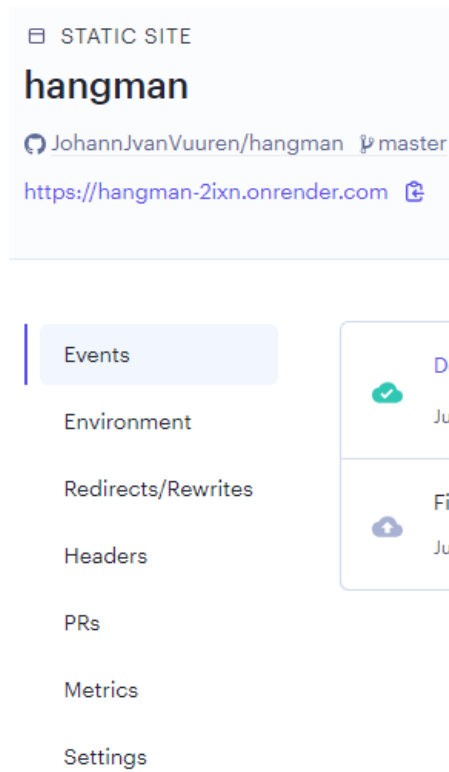
The screenshot shows the Render dashboard for a project named 'hangman'. The top navigation bar includes the 'render' logo, links to 'Dashboard', 'Blueprints', 'Env Groups', 'Docs', 'Community', and 'Help', a 'New +' button, and a user profile for 'Johann Jansen van Vuuren'. The main content area shows the project 'hangman' with a 'Connect' button and a 'Manual Deploy' button. Below this, a sidebar on the left lists 'Events', 'Environment', 'Redirects/Rewrites', 'Headers', 'PRs', 'Metrics', and 'Settings'. The 'Events' section is active, showing a deployment event from July 26, 2023, at 4:12 PM, marked as 'Live'. The event details show the commit '84fad0f 2023-02-20: Final commit. Change font family, sizes and w...'. Below the event details is a 'Search logs' input field with a 'Search' button. The logs themselves are displayed in a dark-themed terminal window, showing the following output:

```
Jul 26 04:14:07 PM You may serve it with a static server:
Jul 26 04:14:07 PM
Jul 26 04:14:07 PM   yarn global add serve
Jul 26 04:14:07 PM   serve -s build
Jul 26 04:14:07 PM
Jul 26 04:14:07 PM Find out more about deployment here:
Jul 26 04:14:07 PM   https://cra.link/deployment
Jul 26 04:14:07 PM
Jul 26 04:14:07 PM Done in 13.79s.
Jul 26 04:14:12 PM --> Your site is live 🎉
```

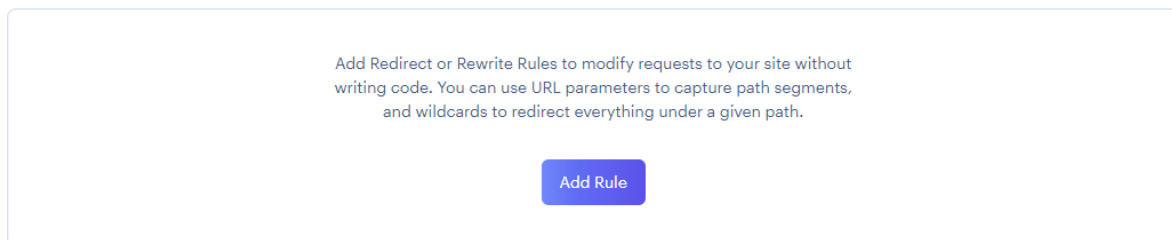
At the bottom of the logs, there is a 'Scroll to bottom' button. The footer of the dashboard includes links for 'Feedback', 'Invite a Friend', and 'Contact Support'.

Since you are using React Router for client-side routing in your application, you will need to direct routing requests to index.html.

Click on the **Redirects/Rewrites** tab on the left-hand side of the screen:



Below, click on **Add rule**:



Under Source add: `/*`

Under Destination add `/index.html`

Under action select: **Rewrite**

Below, click on: **Save Changes**.

Redirect and Rewrite Rules

Add Redirect or Rewrite Rules to modify requests to your site without writing code. You can use URL parameters to capture path segments, and wildcards to redirect everything under a given path.

Source	Destination	Action
<div><div>↑</div><div>↓</div><div>/*</div></div>	<div>/index.html</div>	<div>Rewrite</div> <div>⌵</div> <div></div>

Add Rule

Save Changes

With these simple steps, your static application will be deployed immediately and will be rebuilt every time there is a new pull request on GitHub!

In closing, Render is an excellent Platform-as-a-Service (PaaS) for the deployment of applications directly from your GitHub account without any command line interfacing or any other complicated methods.



Take note:

Using Render for dynamic websites that rely on state, for example, with databases and backends, is a little more complicated than this task. It will require extra configuration and dependencies in your projects. If you're interested, check Render's [website](#). In particular, the Web Services section under products.

Compulsory Task

Follow these steps:

In your previous project, you created a basic app using Create React App.

- Push this app to a remote GitHub repo using access tokens for authentication. Do this by following the instructions above.

- Ensure your GitHub repo is public and share your project URL in a text file called **github.txt**.
- **Optional:** After pushing your basic app from the previous capstone project to GitHub, you may deploy this app using Render. Do this by following the instructions above.
 - Create a text document called **link.txt** in which you provide the URL to your deployed app.



Rate us Share your thoughts

HyperionDev strives to provide internationally excellent course content that helps you achieve your learning outcomes.

Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

[Click here](#) to share your thoughts anonymously.

REFERENCE

- Render home page (2023) Retrieved 20 July 2023
<https://render.com/>