Hyperiondev

Task

React – Deployment

Visit our website

Introduction

There are various deployment methods for static site hosting, such as Docker containerisation and GitHub Pages. 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 for 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 that 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, but 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, and 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 projects. 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**.

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 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 is 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. Log in 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.

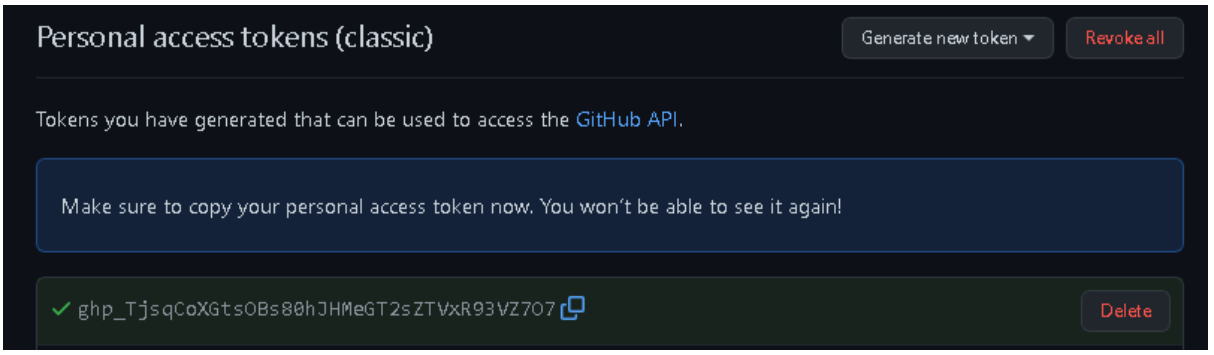
   Fine-grained tokens, on the other hand, are a newer type of personal access token that provides more control over the level of access granted to users, enabling access to be granted to specific resources without giving full access to the entire account or organisation.

   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 when you're 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 is to 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 is to 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**.

Now that you are familiar with GitHub, let's move on to 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 repository for the application.

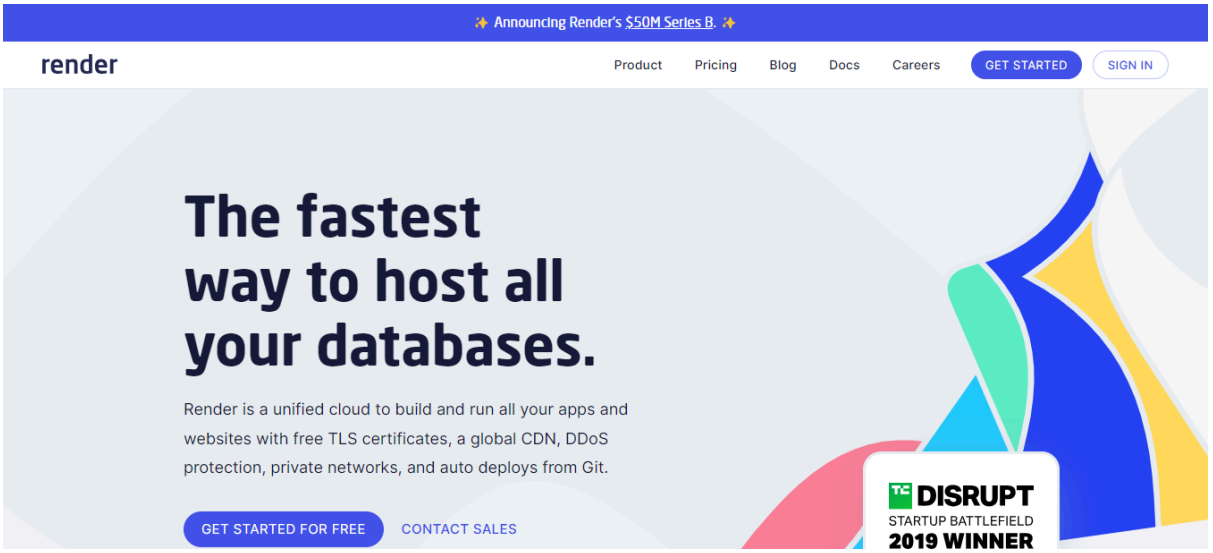
Another big advantage of Render is that static deployments are free of charge as long as you stay within 100 GB of bandwidth a month.

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.

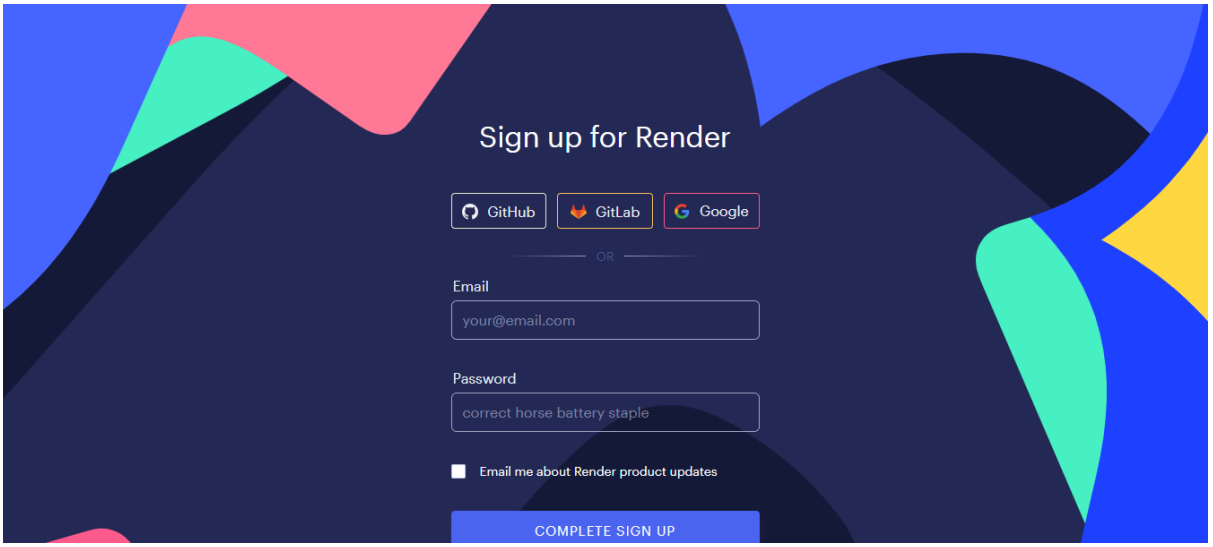
SIGN UP FOR A RENDER ACCOUNT

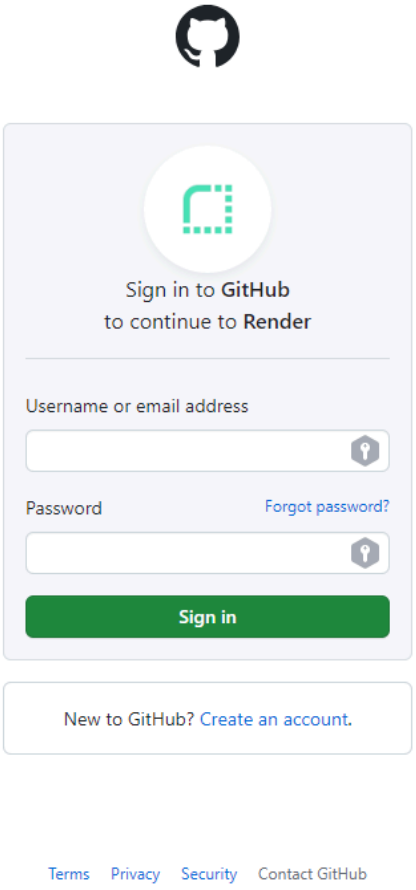
You are going to have to sign up for a Render account to be able to deploy your app via Render. Because deployment platforms can unexpectedly become subscription-based, the choice of whether to sign up for a Render account is completely **optional** and not compulsory in any way. This only applies **if you wish to deploy** your application to the World Wide Web. If you would like to go ahead and do so, go to the **Render website** and follow the instructions below.

Click on the "Get Started" button on the top right-hand corner:



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.





BRIEF OUTLINE FOR USING RENDER

It's very easy to deploy a React application with Render in three steps:

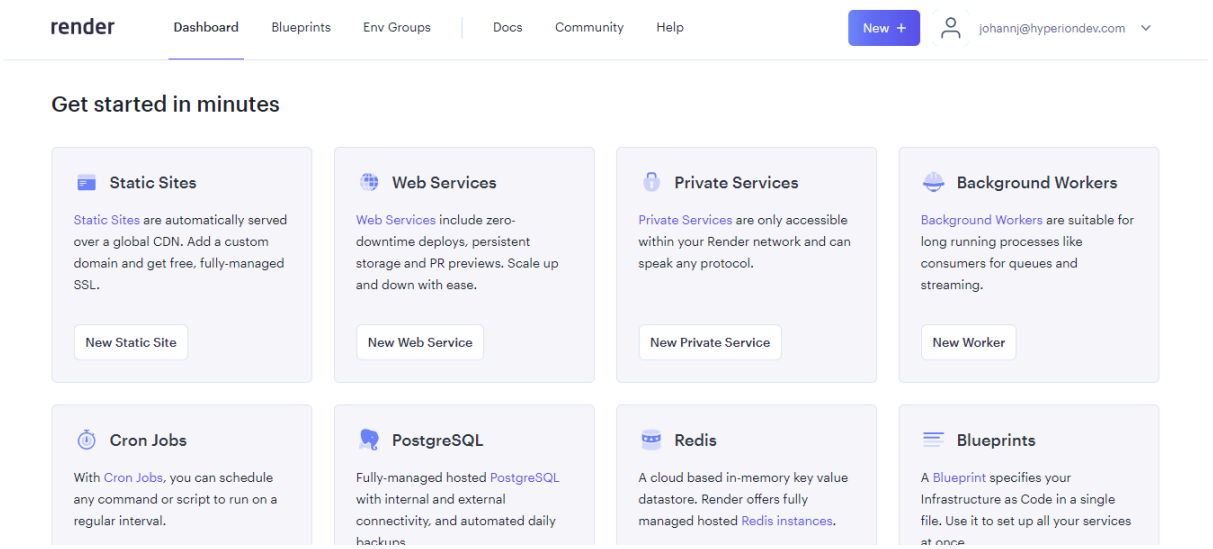
1. Make sure you are happy with your application's GitHub repository (i.e., it's finalised and up to date).
2. Create a new static site from the Render dashboard and give Render access to your GitHub repository.
3. Use the following commands during creation:
    - Build command: `yarn build`
    - Publish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

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

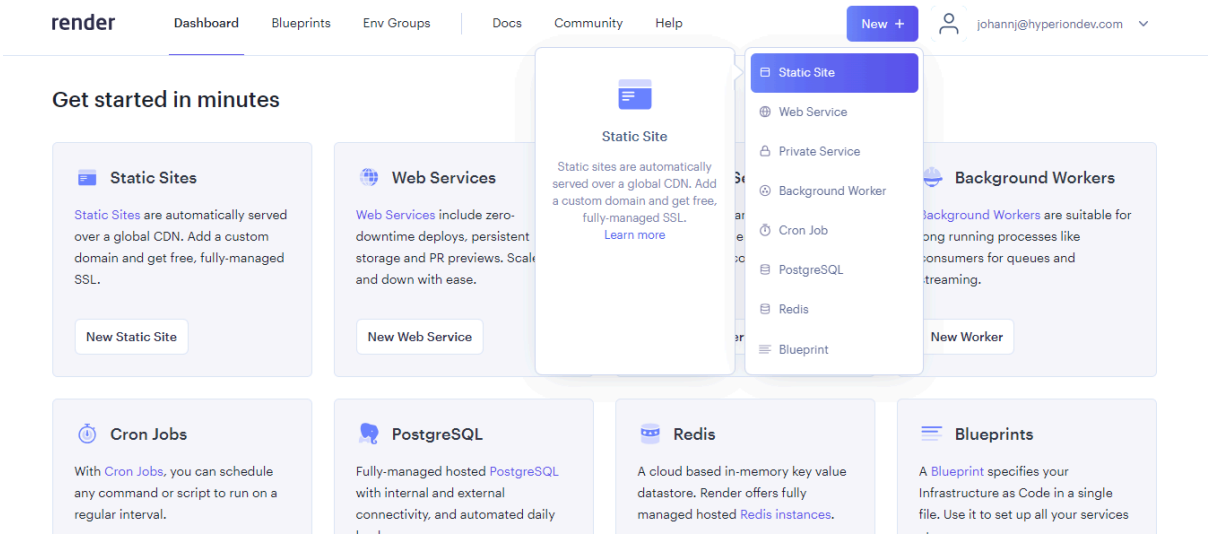
DEPLOYING YOUR APP TO RENDER

The detailed deployment process is outlined in the following steps.

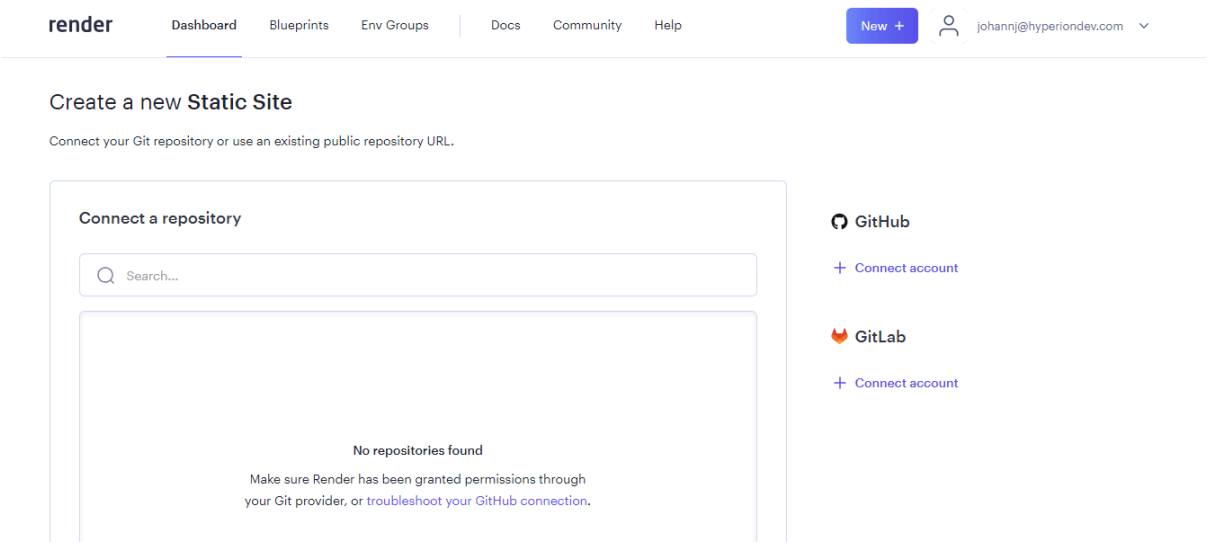
First, sign in to Render. Your dashboard will appear:



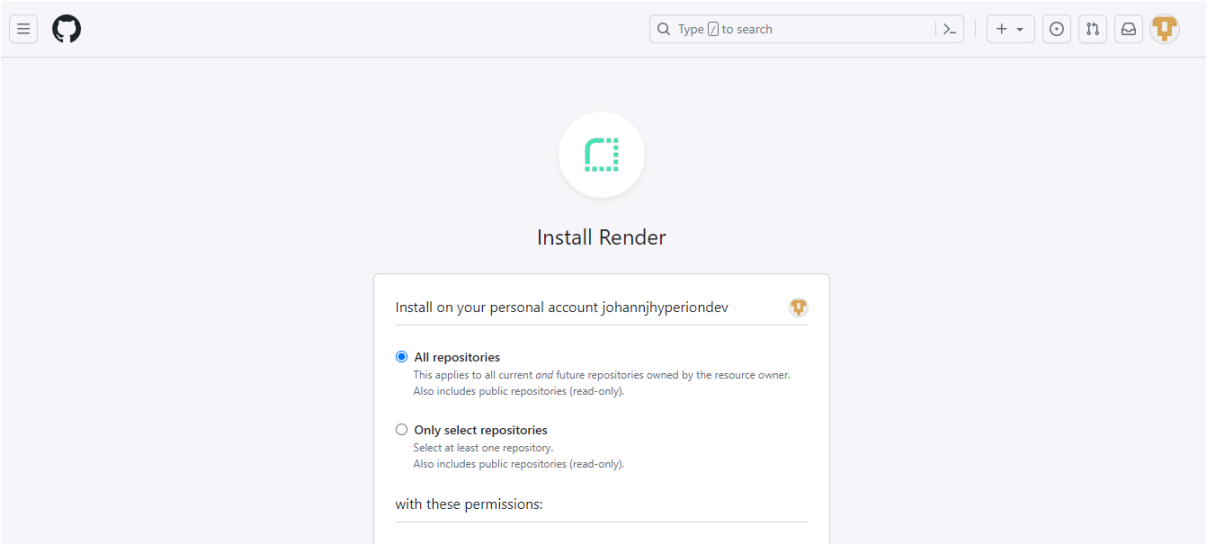
Click on the “New +” button (top right), then click on the “Static Site” button:



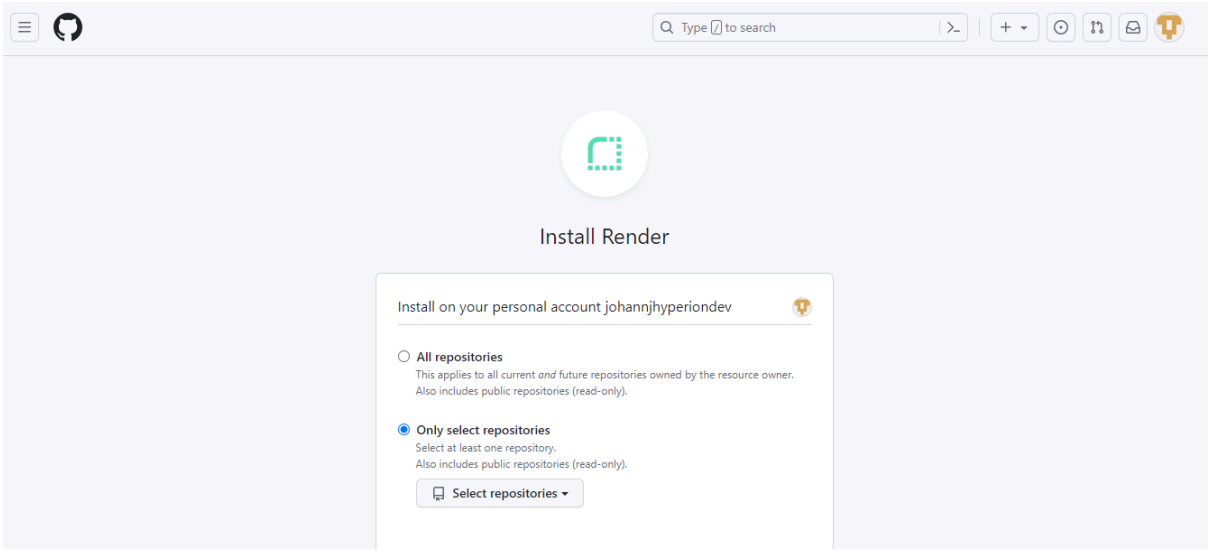
Under the GitHub icon/heading, click on “Connect account”:



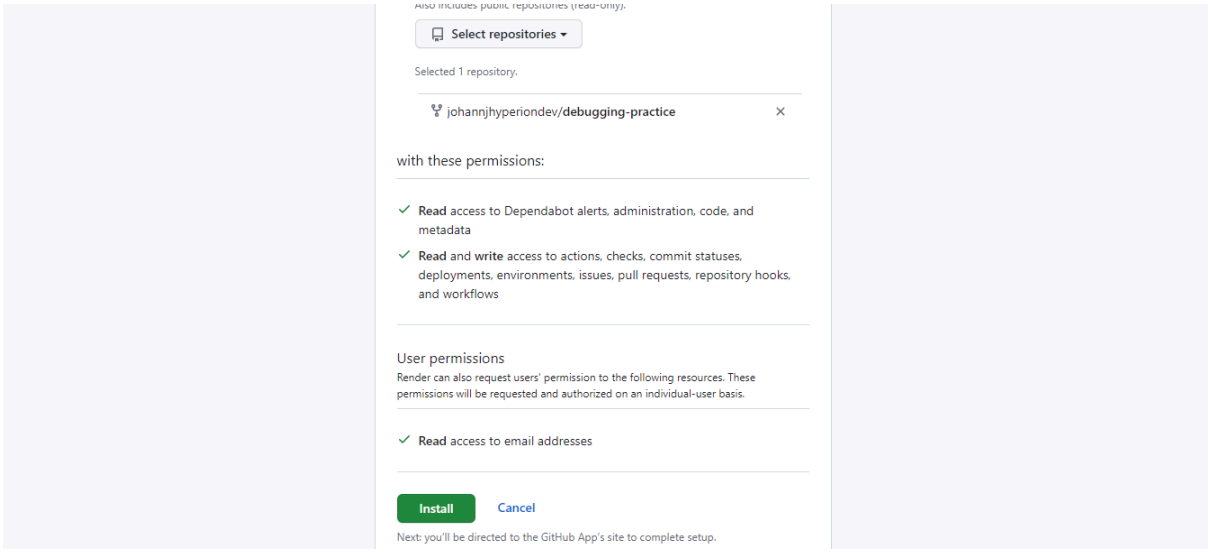
On the next page, choose the “Only select repositories” radio button:



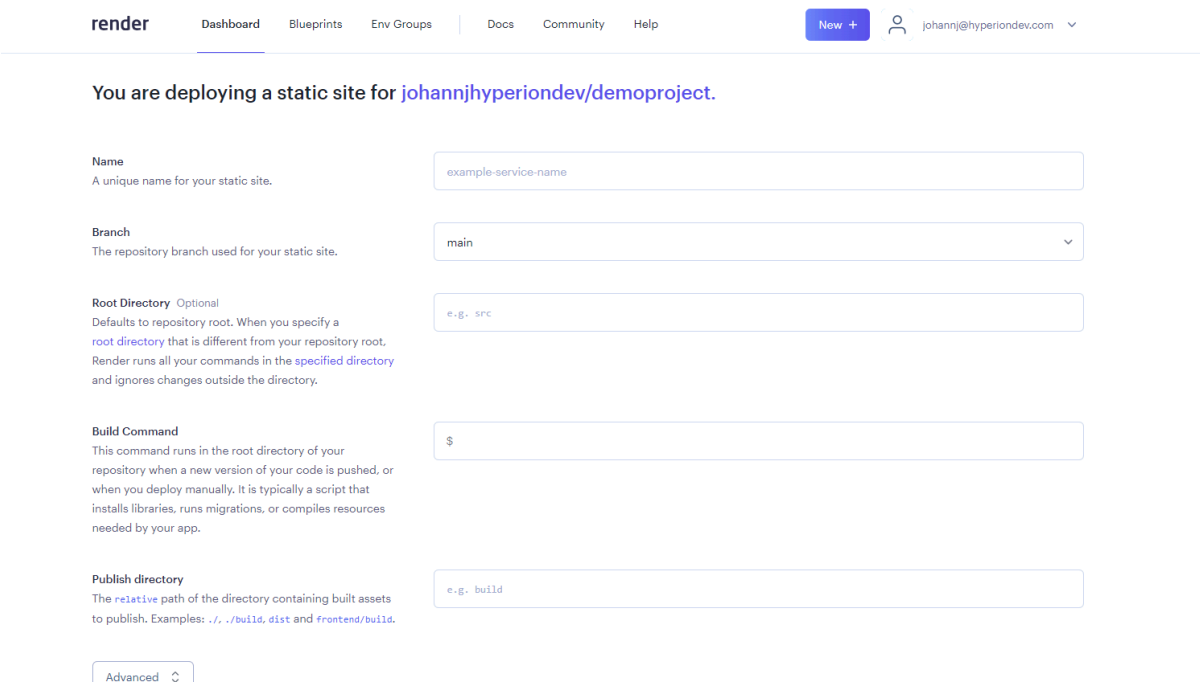
Then click on the “Select repositories” button:



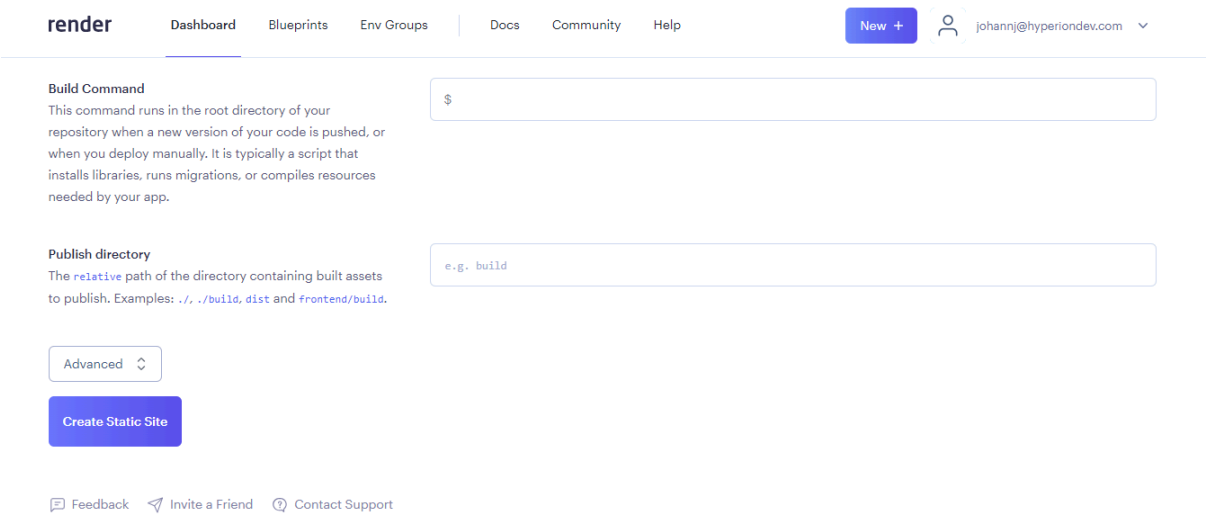
Then click on the green “Install” button:



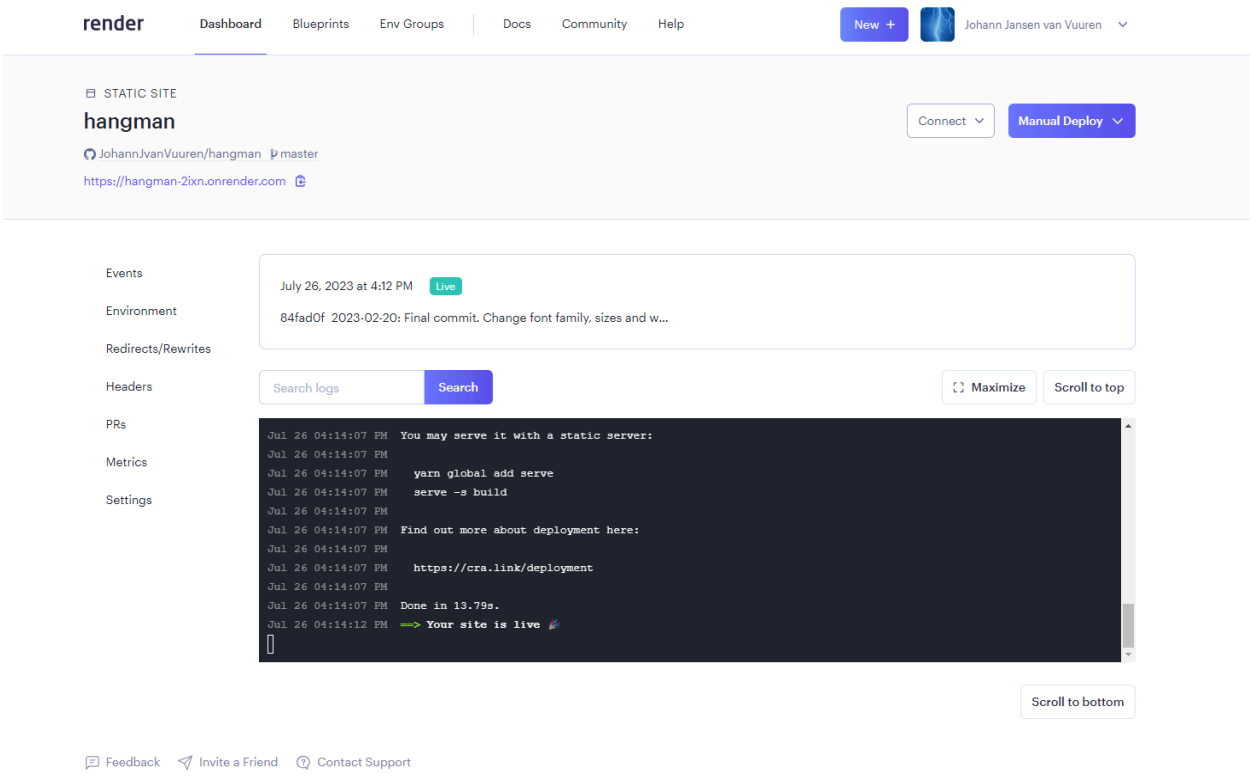
On the next page, add a name of your choice for your static site next to the “Name” label. Leave the “Branch” label as it is, and ensure the “Root Directory” field is left empty or set to its default value. Update the “Publish directory” field to **dist**, as this is the default output folder when using Vite with NPM:



Next, proceed to deploy your site by clicking the relevant button to create your site:

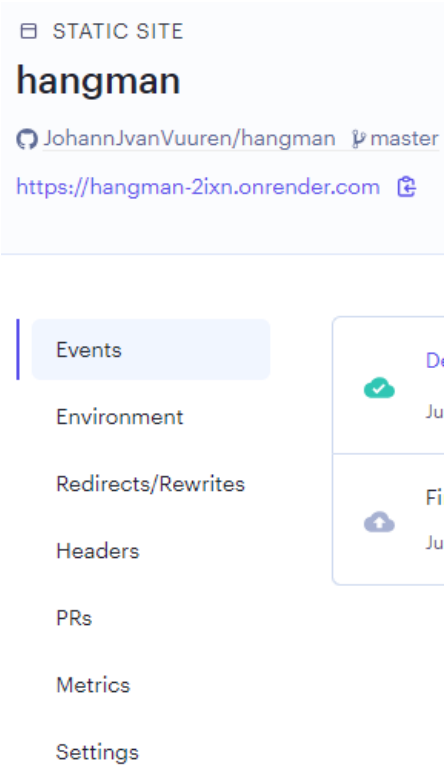


It will now take some time for the build to complete. When the build is finished, a “Live” icon will appear and the URL will be displayed towards the top left of the screen:

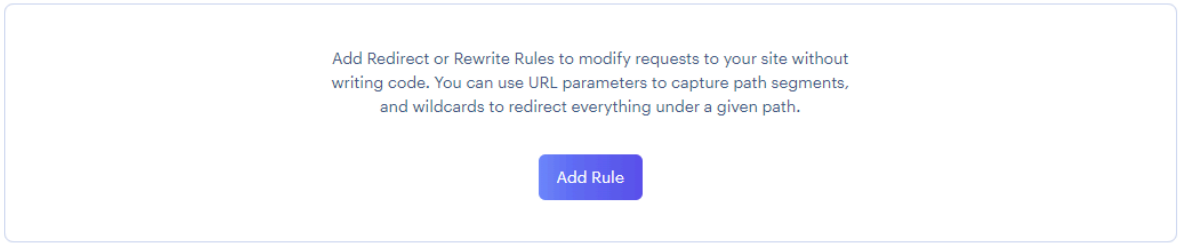


If your project includes client-side routing (e.g., React Router), you'll need to ensure that routing requests are directed to your application's entry point, such as **index.html**.

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



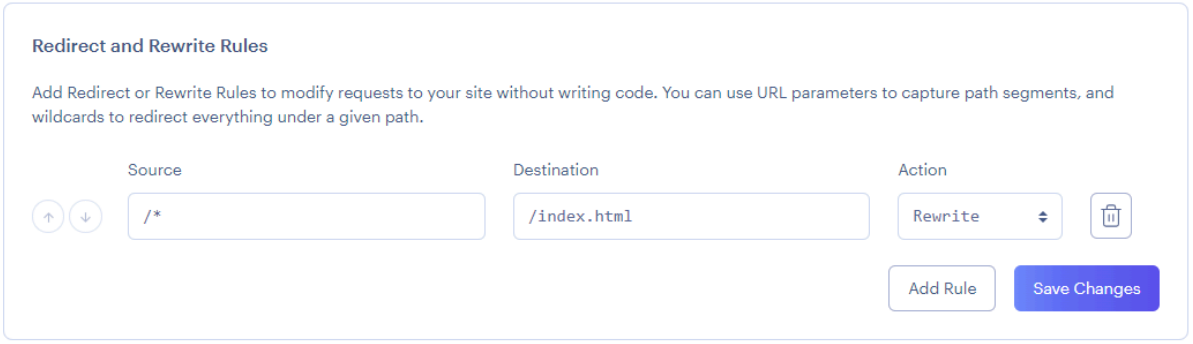
Then, click on “Add Rule”:



As seen in the following screenshot, fill in the fields with the following:

- Under “Source” add: /*
- Under “Destination” add: /index.html
- Under “Action” select: Rewrite

Then, click on “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 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 back ends, 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 “Product”.

Practical task / Auto-graded task

In your previous React capstone project, you created a React application. Ensure you use this React capstone project to proceed with the following steps:

- If you do not already have a repository for this project, create a public repository on GitHub.
- Push your React application to the GitHub repository. Ensure all necessary project files are included, and confirm that the repository is set to “Public”.
    - Share the URL of your GitHub repository in a text file named **github.txt**. This file should contain **only** the URL to your repository.
- **Optional:** After pushing your React application from the previous capstone project to GitHub, you may deploy this app using Render. Do this by following the instructions in this task.
    - Create a text document called **link.txt** in which you provide the URL to your deployed app.

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.