

Deploying a Simple Application with Render

Learn how to deploy your first web service application using Render!

Introduction

In today's fast-paced digital landscape, it's crucial to have reliable and efficient deployment processes for applications. With popular Platform as a Service (PaaS) products like [Render](#), we can set up and launch our applications to end-users quickly and efficiently.

In this tutorial, we will practice deploying a simple application using Render and go over:

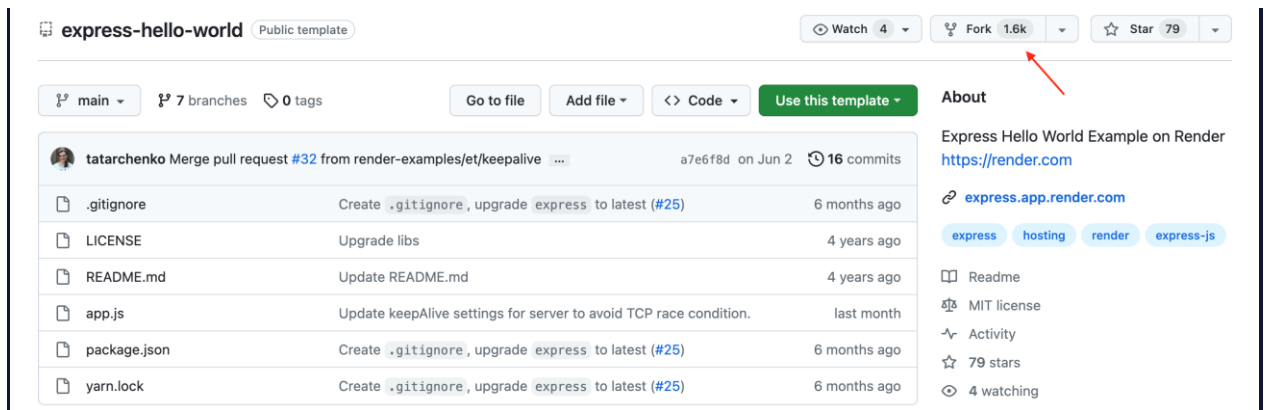
- [Forking a sample Render code repository on GitHub](#)
- [Connecting a GitHub repository to Render](#)
- [Configuring a web service deployment](#)
- [Deploying a new Render web service](#)

Note: This tutorial requires access to a GitHub account and assumes familiarity with using GitHub. If you are unfamiliar with GitHub or need help setting up an account, we recommend visiting the [Learn GitHub: Introduction](#) course first.
Let's get started!

Forking a Sample Render Application

To get started with deploying with Render, we will first need an application. We can use an existing application we have written previously or one provided by Render. Render provides a few different [sample applications](#) that span a variety of popular languages and frameworks. These applications are hosted on GitHub that can be used for quickly setting up, configuring, and testing the deployment process.

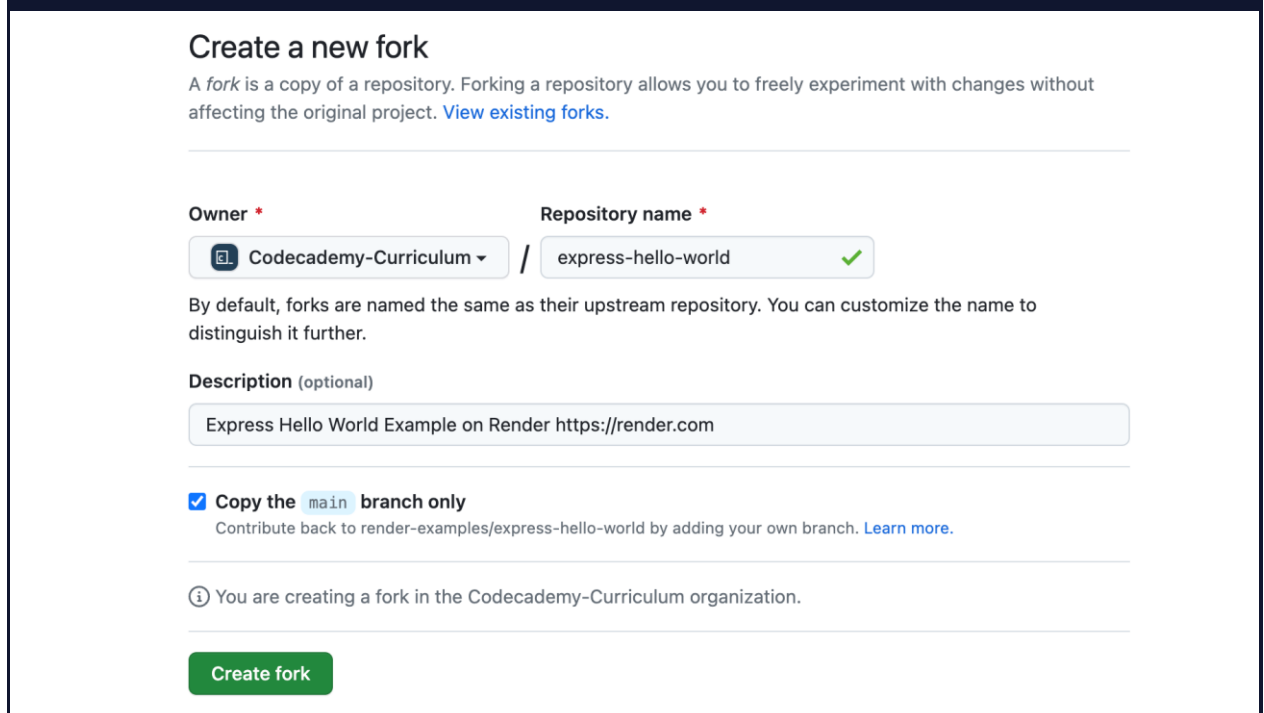
For the purposes of this tutorial, we will be using the [express-hello-world](#) sample application, which is a simple web application built with Node.js and Express.js. While knowing Node.js or Express.js may help and provide context, it is not necessary to have prior experience with either. The full application code is provided by the Render team and we will be forking the code repository. To fork the repository, click the "Fork" button on the top-right corner.



We will need to configure a few settings to fork the repository correctly for this tutorial:

- Ensure the GitHub username is correct.
- Select a name for the forked repository. The name can also be left as the default, original repository name.
- Add an optional description describing what this code repository does. This field can also be left as the default, original repository description.
- We are only interested in the code found in the `main` branch for our application, so leave that checkbox selected.

When completed, the settings should look something like this:



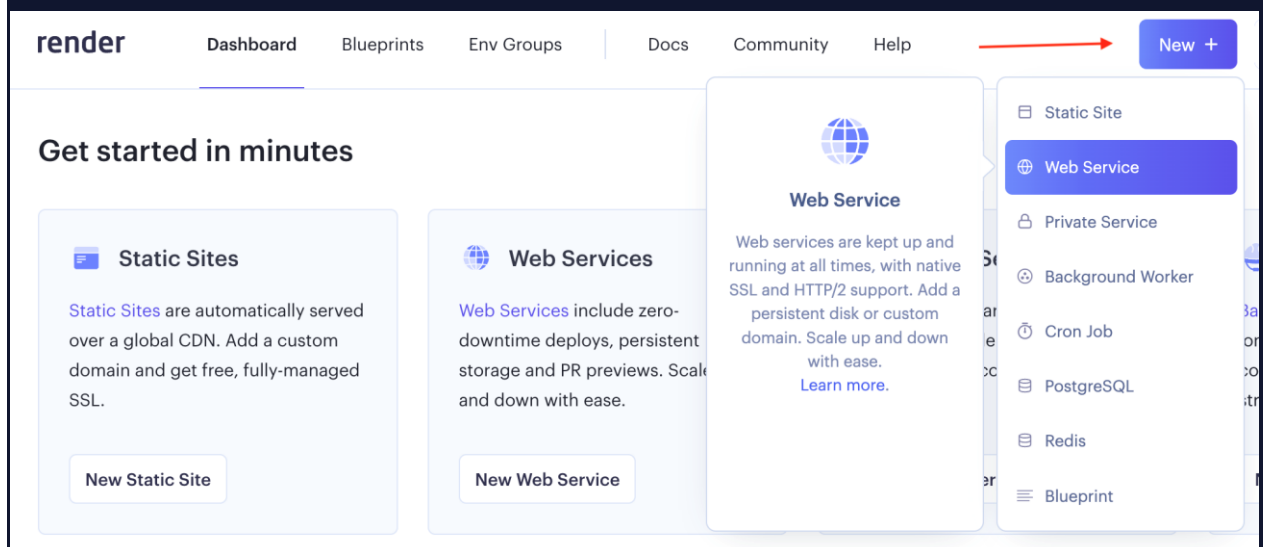
Once the settings are confirmed, click the “Create fork” button. Then, take a moment to browse the files in the repository. Notice that the repository has a file called `app.js`, which holds the main HTML and JavaScript code used to

display the web application content. Two more files, `yarn.lock` and `package.json` can also be found in the repository and are used to build and deploy the application. Don't worry too much about the files just yet, we will have a chance to explore them once we create more complicated deployments.

Now, with our forked repository, we can start learning how to deploy it using Render!

Deployment with Render

To start the process of deployment, we need to navigate to the [Render dashboard](#) (make sure to be logged in to see the dashboard). From the top-right of the menu, we can click the blue, "New +" button to configure our deployment. From the dropdown, select the "Web Service" option to deploy our application to a web server.



Once a service is selected, we will need to select our method to deploy the web service. In this case we'll select "Build and deploy from a Git repository":

[Screenshot showing the two options to "build and deploying from a git repository" or "deploy an existing image from a registry"](#)

Now we will need to connect a GitHub or GitLab account and repository. For this tutorial, we'll stick with GitHub. The page will look like:

Create a new Web Service

Connect your Git repository or use an existing public repository URL.

Connect a repository

Connect your Render account to GitHub or GitLab to begin using your existing repos for new services.

 Connect GitHub

 Connect GitLab

 GitHub

+ [Connect account](#)

 GitLab

+ [Connect account](#)

After choosing to connect a GitHub account, we should be able to install Render on our GitHub account for all repositories repository or select specific repositories we want to deploy as a web service. It should look like this:



Update Render's repository access

Render has access to the following repositories organization
Codecademy



Repository access

☒ **Only select repositories**


Select at least one repository.

Also includes public repositories (read-only).



Select repositories ▾

express

Pe  Codecademy-Curriculum/express-hello-world
Express Hello World Example on Render <https://render.com>



members, metadata, and organization hooks

✓ **Read and write access to actions, checks, commit statuses, deployments, environments, issues, pull requests, repository hooks, and workflows**



Install

With Render installed on our GitHub account, we'll be navigated back to the Render site to choose the repository we would like to connect to.


Create a new Web Service

Connect your Git repository or use an existing public repository URL.



Connect a repository

 Codecademy-Curriculum / Render-Deployment  • 2 days ago

Connect

 Codecademy-Curriculum / express-hello-world • 3 days ago

Connect




 Codecademy-Curriculum / Personal-Budget-Part-II-Solution-...  • 7 days ago


Connect

Public Git repository

Use a **public repository** by entering the URL below. Features like [PR Previews](#) and [Auto-Deploy](#) are not available if the repository has not been configured for Render.

Continue

 **GitHub**
 @Codecademy-Curriculum  • 3 repos
[Configure account](#)

 **GitLab**
[+ Connect account](#)

Select the “Connect” button to give Render permission to access the forked repository from earlier. Once it finishes connecting, we can start configuring the deployment.

Configuring a Web Service

To start, notice that Render has automatically detected the type of code we are using in our application to determine the environment needed for deployment.

You are deploying a web service for **Codecademy-Curriculum/express-hello-world**.

You seem to be using **Node**, so we've autofilled some fields accordingly. Make sure the values look right to you!

By detecting what type of application we are deploying, Render is able to pre-populate some of the configuration fields for us. While having these fields pre-populated saves us some time configuring our deployment, we can modify them as needed. Let's go through the main settings to be aware of:

- **Name:** This field represents the unique name we want for our web service. It is important to note that the name chosen here will also be

used to generate the Render-provided URL. So if we select `"my-hello-world-app"`, our application URL will be `"my-hello-world-app.onrender.com"`.

- **Region:** Since Render manages the infrastructure (e.g., memory, storage) and hosting of the server, the service can be hosted in a variety of regions. It is recommended to select a region that is closest to where a majority of the application's users will be located. For the purposes of the tutorial, we can leave the default region that Render selects for us. However, know that in a production application, the region field is important for determining the latency, i.e. the server's response time.
- **Branch:** This setting will point to the branch within our forked repository that we want to deploy the code from. Typically, this will be the "main" branch; however, we may want to deploy several instances of your application pointing to different branches. For example, we could have a "test" branch, a "pre-production" branch, and a "production" branch and have three different deployment instances for each.
- **Root Directory:** This optional setting tells Render the repository directory location to run all deployment commands. If this field is left empty, it will default to the root directory of the connected repository.
- **Runtime:** Earlier, we mentioned that Render was able to automatically pre-populate some configuration settings for us by scanning the files in the connected repository. For our sample application, Render has pre-selected **Node** for the runtime environment. Make sure to double-check that the correct runtime is selected when starting a new project.
- **Build Command:** This setting sets the command Render uses to install libraries or packages needed for the associated application to run. It will use it when it attempts to deploy the application (we will see an example in a bit). For our Node.js Render sample application, and per the instructions of the sample application `Readme.md` file, we need to use the build command `yarn` to install our dependencies. Since Render already detected the Node.js application, it has pre-populated the command into the field.
- **Start Command:** Similar to the "Build Command" option, Render also requires a "Start Command" that will run from the "Root Directory" location and starts up all the processes needed to run the application. Since we are creating a web service, this command is used to start the web server. This code is found in `app.js`. We can also dig into `package.json` and find the `script` property and see that `start` runs the `node app.js` command, i.e. use Node to execute the `app.js` file to boot up the server.

- **Instance Type:** This field selects which instance type we will use for the deployment. Render offers a “Free” instance tier which provides sufficient resources for deploying our simple application. Explore the features and limitations of a free instance type in the [Render documentation](#).

Whew! That was a whole lot of settings. Don’t worry, setting up service configurations will become a breeze once we have deployed a few more times. Now that the web service is configured, let’s deploy it!

Note: There is an additional “Advanced Settings” option on the service configuration page that is not covered above.

Multiple choice

Which of the following web service configuration settings tells Render which package dependencies to install in order for the application to build and deploy successfully?

Build Command

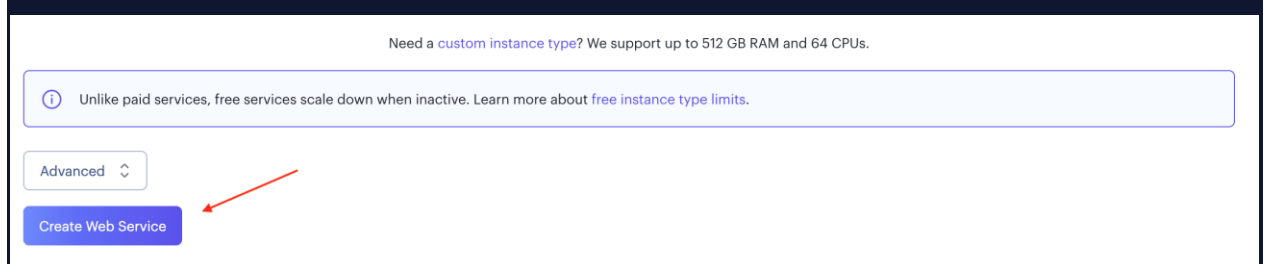
Runtime

Region

Start Command

Building and Deploying the Web Service

At the bottom of the web service configuration page, there will be a blue “Create Web Service” button. Click this to start building and deploying your application.



We will then be redirected to a different page, the web service dashboard page, that will have a console window that will show the initial build steps using the configuration settings we supplied. Take a look at the console and observe the steps it takes. It will look similar to this:


```
Jul 6 08:11:17 AM ==> Cloning from https://github.com/Codecademy-Curriculum/express-hello-world...
Jul 6 08:11:18 AM ==> Checking out commit a7e6f8df8777411038d0246629c18e5aaee9797 in branch main
Jul 6 08:11:20 AM ==> Downloading cache...
Jul 6 08:11:32 AM ==> Transferred 45MB in 8s. Extraction took 1s.
Jul 6 08:11:33 AM ==> Using Node version 20.4.0 via /opt/render/project/src/package.json
Jul 6 08:11:33 AM ==> Docs on specifying a Node version: https://render.com/docs/node-version
Jul 6 08:11:33 AM ==> Running build command 'yarn'...
Jul 6 08:11:34 AM yarn install v1.22.5
Jul 6 08:11:34 AM [1/5] Validating package.json...
Jul 6 08:11:34 AM [2/5] Resolving packages...
Jul 6 08:11:34 AM success Already up-to-date.
Jul 6 08:11:34 AM Done in 0.14s.
Jul 6 08:11:34 AM ==> Uploading build...
Jul 6 08:11:43 AM ==> Build uploaded in 8s
Jul 6 08:11:43 AM ==> Build successful 🎉
Jul 6 08:11:43 AM ==> Deploying...
Jul 6 08:12:05 AM ==> Using Node version 20.4.0 via /opt/render/project/src/package.json
Jul 6 08:12:05 AM ==> Docs on specifying a Node version: https://render.com/docs/node-version
Jul 6 08:12:05 AM ==> Starting service with 'node app.js'
Jul 6 08:12:06 AM Example app listening on port 10000!
```

□

To summarize, Render will attempt to deploy by performing the following operations:

1. First, Render will clone the GitHub repo and check out the branch specified in the configuration. In this case, notice it is the “main” branch.
2. Next, Render will build the application using the command specified in the configuration. In this case, notice it says it is building via the “yarn” command. This build step may do several operations like validate, fetch, and build packages.
3. Next, under the hood, Render uses containerization technology to spin up a cloud-based infrastructure for your web app.
4. Lastly, once Render is done deploying the application, it will start the web service. In this case, notice it runs the `node app.js` command.

We can confirm that the application is running by verifying the green “Live” state above the console window.

July 6, 2023 at 8:24 AM Live

a7e6f8d Merge pull request #32 from render-examples/et/keepalive ...

Search logs

Search

Maximize

Scroll to top

```
Jul 6 08:25:05 AM ==> Running build command 'yarn'...
Jul 6 08:25:05 AM yarn install v1.22.5
Jul 6 08:25:05 AM [1/5] Validating package.json...
Jul 6 08:25:05 AM [2/5] Resolving packages...
Jul 6 08:25:05 AM success Already up-to-date.
Jul 6 08:25:05 AM Done in 0.13s.
Jul 6 08:25:06 AM ==> Uploading build...
Jul 6 08:25:04 AM ==> Transferred 45MB in 8s. Extraction took 2s.
Jul 6 08:25:14 AM ==> Build uploaded in 8s
Jul 6 08:25:14 AM ==> Build successful 🎉
Jul 6 08:25:14 AM ==> Deploying...
Jul 6 08:25:51 AM ==> Using Node version 20.4.0 via /opt/render/project/src/package.json
Jul 6 08:25:51 AM ==> Docs on specifying a Node version: https://render.com/docs/node-version
Jul 6 08:25:51 AM ==> Starting service with 'node app.js'
Jul 6 08:25:53 AM Example app listening on port 10000!
Jul 6 08:25:58 AM Your service is live 🎉
[]
```

Scroll to bottom

With our application now deployed and running, we can try accessing the application Render-provided URL at the top left.

WEB SERVICE

Express Example Node Free

Connect

Manual Deploy

Codecademy-Curriculum/express-hello-world ↗ main

<https://express-example-i8i9.onrender.com>

Clicking this link should open a new browser window where our application will be running!



Congratulations! We've just successfully deployed our first application using Render!

Wrap Up

As we've seen, Render's intuitive interface and sample pre-made applications make it an excellent choice for quick application deployment. To recap, in this article, we covered how to do the following:

- Fork a sample Render code repository in GitHub
- Connect a code repository in Render to create a web service
- Configure the Render settings to deploy a web service
- Deploy a new Render web service

Now that we have completed a full web service deployment, continue exploring Render and take advantage of its powerful features to practice deploying more existing or new applications!