Rishi Kavikondala

Follow 3 Followers About

How to Deploy a Static Website to Heroku

Rishi Kavikondala May 24, 2020 · 6 min read

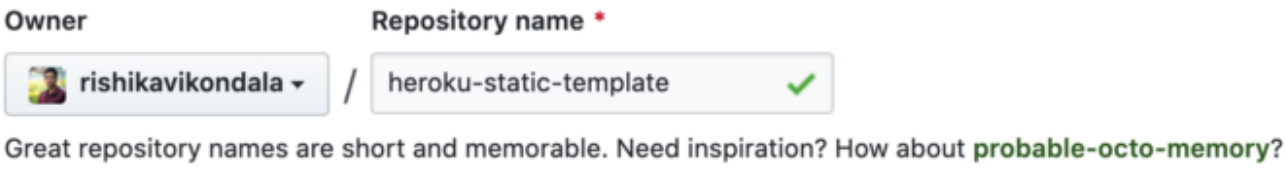
Prerequisites:
- Install a text editor (Visual Studio Code recommended)
- Set up and install git and GitHub
- Make a GitHub account
- Install Node.js
- Install Heroku Command-Line Interface (CLI)
- Make a Heroku account

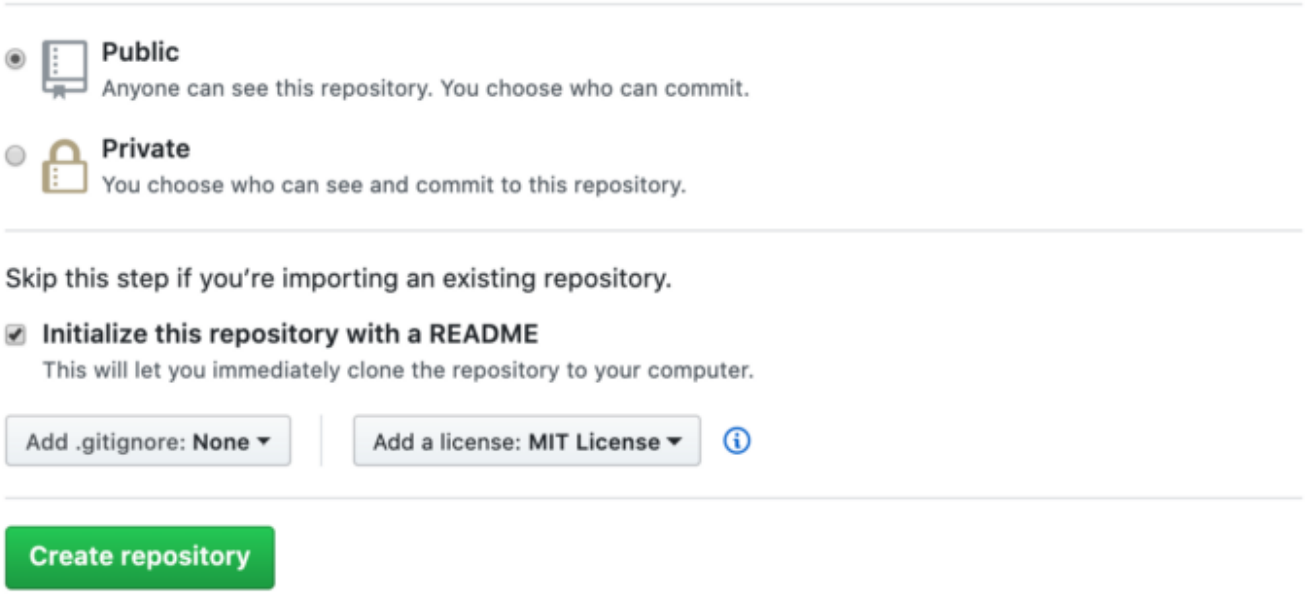
Difficulty level: Beginner
Time to complete: 15 minutes.

The purpose of this article is to teach the basics of building a Node.js server and using Heroku to easily deploy static websites. The HTML/CSS portion is intentionally made simple as it is not the main focus of the tutorial.

Part I: GitHub repository

1. Create a new GitHub repository. For reference, here is the setup that I used:

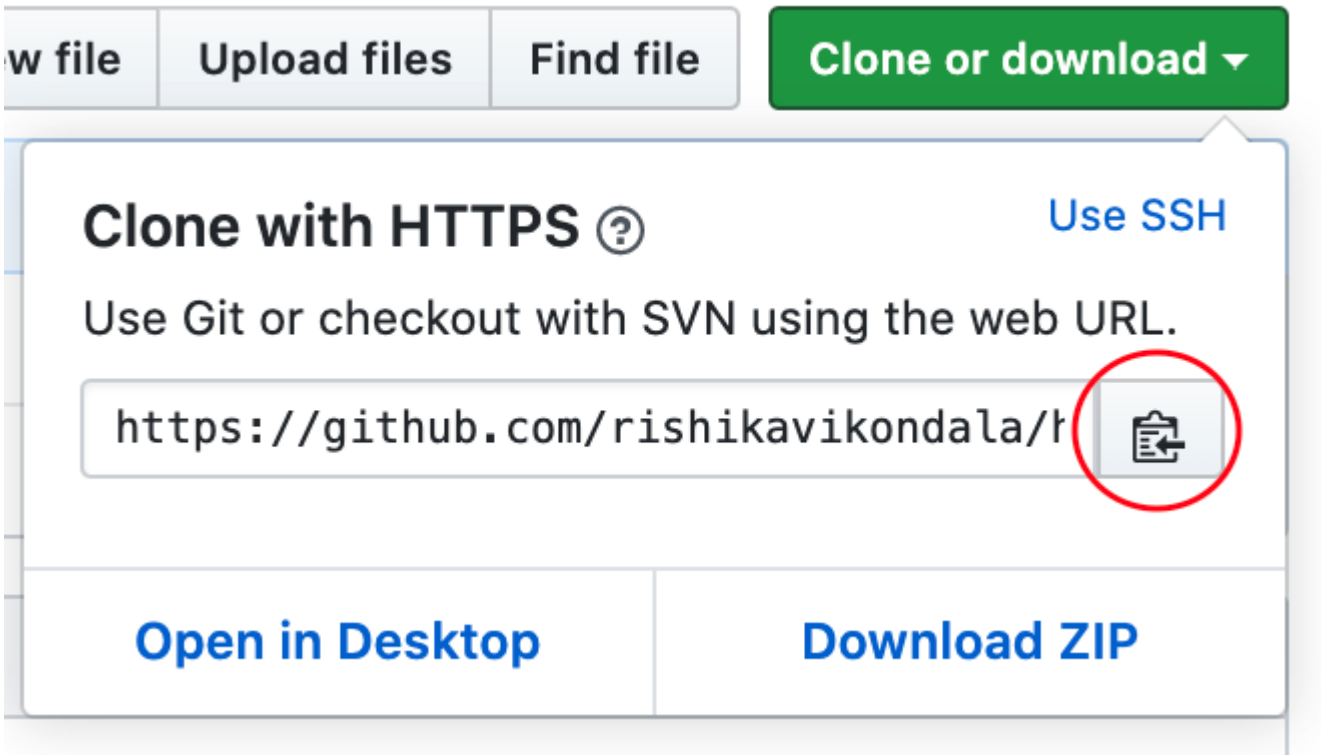




GitHub repository creation preferences

Part II: File structure setup

1. After creation, go to the homepage of your GitHub repository. Click the green "Clone or download" button. Copy the link using the copy button.



Cloning a GitHub repository

example, to navigate to a folder named Projects, use `cd Projects`. Continue using `cd` to get to your desired folder. Alternatively, use `cd` followed by a file path. For example, `cd Documents/Repositories`.

4. Once you are in your desired folder, use `git clone` to create a local clone of your GitHub repository.
5. Use `cd` to navigate into the Git repository. See the text below for an example of what this entire process (steps 1–5) should look like.

```
Last login: Sat May 23 12:00:11 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Rishis-MBP:~ rishikavikondala$ cd Projects
(base) Rishis-MBP:Projects rishikavikondala$ git clone https://github.com/rishikavikondala/heroku-static-template.git
Cloning into 'heroku-static-template'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
(base) Rishis-MBP:Projects rishikavikondala$ cd heroku-static-template/
(base) Rishis-MBP:heroku-static-template rishikavikondala$
```

Repository cloning process

Part III: File structure setup

In this step, we will be creating a very simple Node.js server. Node.js creates a runtime environment for JavaScript code to run outside of a web browser, which is unique because the language was initially designed to only execute within a browser (like Google Chrome). It is used by companies like Netflix, Uber, LinkedIn, and PayPal to build their applications.

To create this server, we will be using a simple Node.js framework called Express. Frameworks can be thought of as dialects of programming languages that make it easier to perform a particular task.

Steps 1–4 below require you to run commands in the command line.

1. Run the following command: `npm init`. npm stands for Node Package Manager, which allows us to install Node.js packages and set up configurations for our server code. This will create a file called `package.json`, which contains a set of instructions and information for our Node.js server. Hit Enter/Return to click through these fields. You

```
(base) Rishis-MBP:heroku-static-template rishikavikondala$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (heroku-static-template)
version: (1.0.0)
description: Template for deploying static HTML sites to Heroku
entry point: (index.js)
test command:
git repository: (https://github.com/rishikavikondala/heroku-static-template.git)
keywords: Heroku, Node.js, HTML, CSS, JavaScript
author: rishikavikondala
license: (ISC)
About to write to /Users/rishikavikondala/Projects/heroku-static-template/package.json:

{
  "name": "heroku-static-template",
  "version": "1.0.0",
  "description": "Template for deploying static HTML sites to Heroku",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/rishikavikondala/heroku-static-template.git"
  },
  "keywords": [
    "Heroku",
    "Node.js",
    "HTML",
    "CSS",
    "JavaScript"
  ],
  "author": "rishikavikondala",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/rishikavikondala/heroku-static-template/issues"
  },
  "homepage": "https://github.com/rishikavikondala/heroku-static-template#readme"
}


Is this OK? (yes) yes
```

Creation of package.json

4. Run `mkdir public` to create a folder to house the static HTML/CSS files.

5. Open your local repository folder in your text editor (I will be using Visual Studio Code). In VS Code, you can do this by clicking File → Add Folder to Workspace. This will add the folder to a navigation panel on the left side of the text editor.

Part IV: Express server

1. Open your `index.js` file.

2. Include the following code in the file.

```
const express = require('express');
const app = express();

app.use(express.static('public'));

app.get('/', (req, res) => {
     res.sendFile('index.html', {root: __dirname + '/public/'});
});

app.listen(process.env.PORT || 5000)
```

This code is performing five functions:

Line 1: Initializes the Express framework.

Line 2: Initializes the Node.js application.

Line 4: Grants the server access to any static files within the `public` folder.

Line 6–8: Sets the home page of the website to be `index.html` .

Line 10: Tells the server to run on port 5000. This is essential for hosting the website on Heroku.

That's it! You've created a very simple Node.js server using the Express framework.

Part IV: Static frontend pages

1. In the previous command line window, run `cd public` .

2. Run `touch index.html` . This will create a home page for the website.

3. Run `touch about.html` . This will create an about page. You can name this file anything

5. Run `cd ..` to exit out of the `public` folder in the command line and go back to the GitHub repository.
6. Open the index.html file in Visual Studio Code. Use this starter code in the file. Alternatively, feel free to use any HTML code you'd like on the home page.
7. Repeat step 6 with about.html and style.css.
8. To see what the site looks like right now, go to the command line and run `node index.js`. This will spin up the server; to see what the site looks like, visit http://localhost:5000/. You should see a very basic HTML and CSS site with minimal styling. You should also be able to switch between the Home page and About page.

Part V: Heroku setup

1. Open the package.json file in Visual Studio Code.
2. Add the following snippet of text to the bottom. The purpose of this text is to specify the version of Node.js being used by the server.

```
"engines": {
    "node": "12.11.1"
 }
```

3. In the `scripts` section of package.json (should be near the top of the file), specify the following command. This will tell Heroku that index.js is the file to run to spin up the Node.js server.

```
"start": "node index.js"
```

4. After completing these steps, your package.json should look something like this.
5. The code needed to deploy to Heroku is now complete, so push your code to GitHub. Run the following commands in your command-line window in order.

```
git add .
git commit -m "Finished Node server and static pages"
```

Part VI: Heroku deployment

Heroku is a cloud platform that makes it easy to deploy, host, and scale web applications. It is a great tool for prototyping websites and for projects that need to be quickly and/or frequently updated.

1. In your command-line window, run `heroku login`. Go through the specified steps to log into your Heroku account.
2. Once again in the command-line, run `heroku create`. You will see that Heroku generates a URL for our website.
3. Run `git push heroku master`. This will push all content for our website into a D*yno*, which is a container used by Heroku to hold all of application code and dependencies into one unit of storage. This command will take some time to complete, as Heroku will be installing Node.js and bundling all the code into a Dyno. Any time you update your website, simply push your code to GitHub, and then re-run this command to update the
4. Run `heroku open`. This will open our newly created and hosted static website.
5. Congratulations! You've just created and hosted a website hosted on a cloud platform. If you want to view more details about what you've built, visit your Heroku Dashboard.

To view a working demo of what was built in this tutorial, see this link. Additionally, the GitHub repository with all the code can be found here.

Part VII: Switching to a Heroku Hobby Dyno (optional)

By default, Heroku hosts in a free Dyno, which is their cost-free hosting plan. If you are a student, you are eligible to receive one free Hobby Dyno. This will allow your website to run faster, while enabling you to view performance metrics and use a custom domain. If you obtain a free Hobby Dyno, making the switch is as easy as going to the Resources tab in your Heroku Dashboard and clicking "Change Dyno Type." To view a comparison of Heroku Dynos, see this link.