

# manually deploying a NodeJS server application to Heroku

Steven Brawer  
June 8, 2017

This document shows how to manually deploy the simplest NodeJS server to Heroku, using GitHub as the repository, and how to invoke the app via a browser. The application is pure NodeJS. The deployment is done via the Heroku GUI and not using the command-line. Altogether, this is among the simplest combinations of server and deployment. This document assumes you are running on a Mac.

Note that *this is **not** a general procedure - just one for getting started.*

The procedure has the following steps. All these steps will be illustrated in detail, including the .js file, the package.json file, and local and Heroku execution.

- Create accounts on GitHub and Heroku.
- Decide on the names to use
- Create a GitHub repository dedicated to the Heroku server application.
- Create the app (a .js file) and package.json file.
- Run the app locally to make sure it works.
- Copy the server app and package.json to the master branch of the GitHub repository created as above.
- Deploy the app with the Heroku GUI.

In addition, it is shown how to enhance the app to read a file.

**Note that this is not a NodeJS commentary or instruction manual. It is assumed**

- you have NodeJS and npm installed globally on your computer,
- that you know the basics of "raw" NodeJS (exclusive of any modules or frameworks)
- that you know how to use GitHub at least in a rudimentary manner,
- and that you know how to execute commands in a terminal window (used only to locally execute the node application. I do not use the terminal window for Heroku.)

## create accounts on GitHub and Heroku

For Heroku, go to <https://signup.heroku.com>. Follow directions to sign up for a free account. Select *Node.js* as the primary development language.

For GitHub, I assume you have an account and you know how to create a repository.

## decide on the names to use

Heroku applications may be linked to GitHub applications, in the sense that Heroku will read all the files in a given GitHub repository, identify which is the entry point (ie, the server .js file) and execute the file, creating a server, which then can be invoked from a browser.

There are several names that are required in order that Heroku and GitHub be coordinated.

- The name of the GitHub repository.
- The name of the application.
- The name of the Node .js file.
- The local directory (on your computer) containing the server application and the package.json file. This is NOT used by heroku (in this example) but rather to run the application locally to ensure that it is correct.

In this application, I will use the following names, so that all examples will be specific. You can substitute your own names. Assume that all names are case sensitive.

name of GitHub repository: herokuTestDeploy  
name of application: first-hello-world-brwr  
name of Node .js file: hello1.js  
local computer directory: <home directory>/NodeJS/networking.

The **local computer directory** will be referred to as the **working directory** or **current directory**.

We will see below where these names fit in.

## create GitHub repository

Create a new GitHub repository dedicated to this Heroku application. As noted above, in this example the repository is named *herokuTestDeploy*.

We will use the **master branch** of this repository.

## create the application and package.json file in current directory (<home directory>/NodeJS/networking in this example.)

This application creates a server, which causes *Hello, world* to be written to the browser.

Create the `hello1.js` file in the working directory. The **hello1.js** file is:

```
const server = require('http').createServer();

server.on('request', (req, res) => {
  res.writeHead(200, { 'content-type': 'text/plain' });
  res.end("Hello world\n");
});

let port = process.env.PORT || 8000;
server.listen(port);
console.log('server listening on port:' + port);
```

A Node server listens on a port. Heroku creates its own port (whose value we will not need anywhere else), while if we run on the local computer, the port will be 8000 (or whatever number you like). The name of the port selected by Heroku, (when running in Heroku), will be returned by the expression `process.env.PORT`. This expression will return *false* when running locally, in which case the port will be 8000.

The **package.json** file should also be in the working directory, and is

```
{
  "name": "first-hello-world-brwr",
  "version": "1.0.0",
  "engines": {
    "node": "6.10.2"
  },
  "description": "first hello world in order to begin to learn heroku manual deployment",
  "main": "hello1.js",
  "scripts": {
    "start": "node hello1.js"
  },
  "dependencies": {},
  "author": "",
  "license": "ISC"
}
```

The following properties of the `package.json` object are crucial in order that Heroku runs the server from the files stored in the GitHub repository's master branch.

- The **name** property is the name of the application (in this example *first-hello-world-brwr*)
- The **main** property is *hello.js*
- The **start** property (in the *scripts* property) is *node hello1.js* - that is, the word *node* followed by space and then the name of the `.js` file.
- The **engines** sub-property tells Heroku the version of node. For consistency, this should be the same version that is on your computer. If you don't know it, bring up a terminal window and type *node -v*, which will return the version.
- The other properties can be whatever you want, or they don't have to be there at all.

## run the app locally (to make sure it works)

In a terminal window, cd to the *working directory* (see above). Type

`npm start`

which will execute *hello1.js* based on the contents of the *package.json* file. Since this is how (in our example) Heroku will start your application, you should execute it this way in testing.

You should see the following message in the terminal window:

`server listening on port: 8000.`

Next open a browser and in the address window type your *localhost* address and the desired port. On a mac this is (using port 8000)

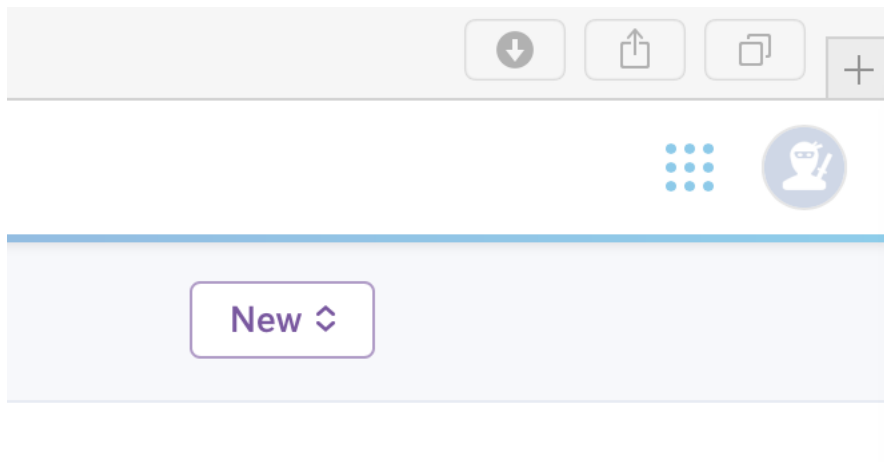
`127.0.0.1:8000`

You should see *Hello, world* in the browser.

## copy the hello1.js and package.json files to the GitHub repository, master branch (created as above)

## deploy the app to Heroku

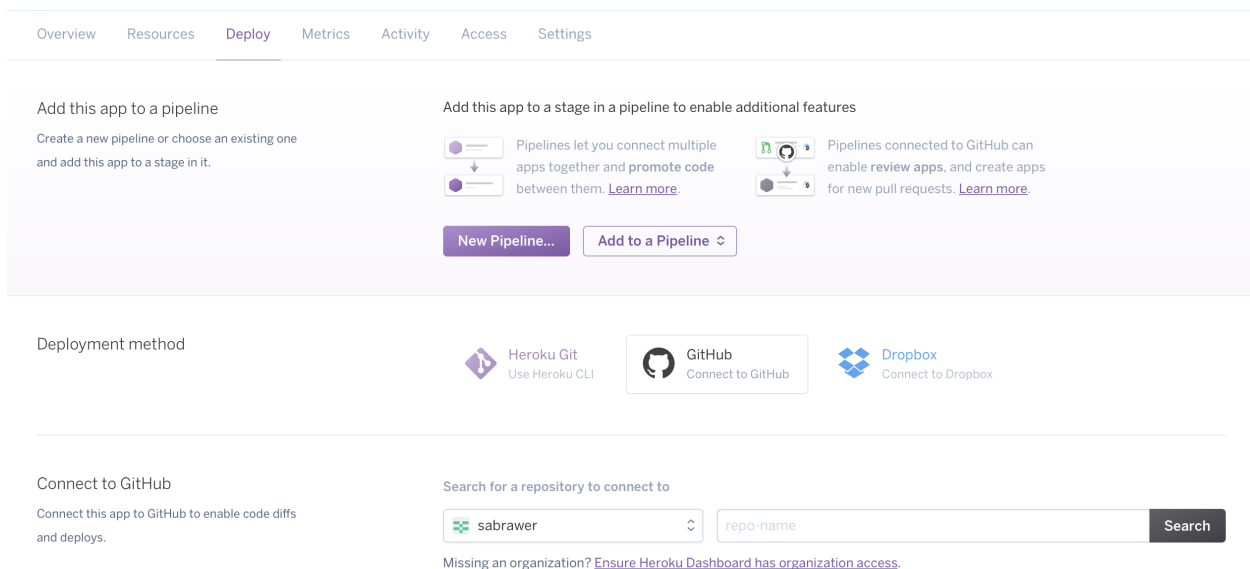
Go to Heroku.com. Login (if necessary). In the window which appears (see figure below), click on the square at right (composed of 9 dots), and select **dashboard**.



Then click the **New** button and select **create new app**.

In the screen which appears, enter the app name. In this example, it is *first-hello-world-brwr*, which is the same as the *name* property in the *package.json* file. Then click the **create app** button.


This is the screen which appears.



In **Deployment method**, select **GitHub**. You will need to authenticate yourself, which generally means entering your github password. It is only necessary to do this the first time.

Next select the github repository which has your application. Click the **search** button, seen above. In my configuration, the following comes up:




Search for a repository to connect to

 sabrawer

repo-name

Search


Missing an organization? [Ensure Heroku Dashboard has organization access.](#)

 sabrawer/Documentation	Connect
 sabrawer/Web-Pages-for-FCC	Connect
 sabrawer/herokuTestDeploy	Connect

Click on the **connect** button of **herokuTestDeploy** (or whatever your repository name is). The following addition to the dashboard page appears:

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to  [sabrawer/herokuTestDeploy](#)

✓ Releases in the [activity feed](#) link to GitHub to view commit diffs

Disconnect...


---

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. Deploys happen automatically; be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more.](#)

 master

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys


---

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)

 master

Deploy Branch

Use the **Manual deploy** section. We are deploying from the master branch, so click on **Deploy Branch**. Heroku reads the *package.json* file to determine the application and how to start it. When it is deployed (ie, executing as a server), the message is: **your app was successfully deployed**, then click on **View**, which runs the app in your browser. The url for your app is

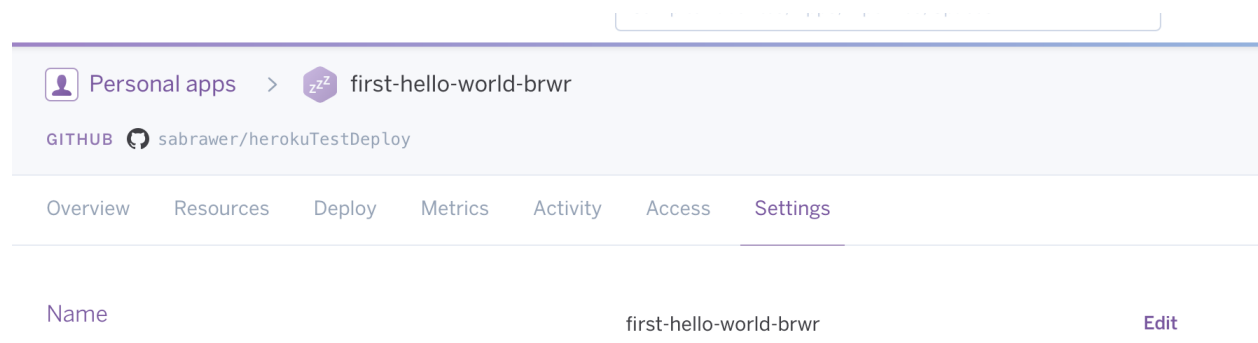
<https://first-hello-world-brwr.herokuapp.com>

You can use this address from any browser to connect to your server. This is the general form of the url:

<https://<application-name>.herokuapp.com>

If you want to change something in the github files, such as change the .js file and/or modify the package.json file, disconnect from github (see **app connected to github** section of the dashboard screen for the **disconnect** button, click that button), then reconnect and deploy, as described above.

To delete the application, go to the top of the dashboard window



select **settings**, scroll to the bottom of the settings page, and click on **delete app**.

## accessing a file

Your server can read the files from the repository. As an example, modify the *hello1.js* file as follows, to access a file **temp.txt** and output the stats:

```
const fs = require('fs');
const server = require('http').createServer();
let str= "not read";
fs.stat('./temp.txt', function(err,stats){
    if(err)
        str = JSON.stringify(err);
    else
        str = JSON.stringify(stats);

    console.log(str);
});

server.on('request',(req,res) => {
    res.writeHead(200,{ 'content-type': 'text/plain' });
    res.end(str + "\n"); // output to the browser
});
let port = process.env.PORT || 8000;
```

```
server.listen(port);  
console.log('server listening on port 8000');
```

Create a **temp.txt** file in the working directory, test the modified application locally, move the modified application and the txt file to the repository and deploy as before.