# Deploying a Node Web Server to Heroku

This guide walks through the necessary steps to deploy your full stack Node.js application to Heroku!

## Prerequisites

To begin with, you'll need a git repository initialized locally with your basic web server code working and committed.

1. There are a couple of ways to do this.
   - If you cloned from a remote repository and then wrote/committed your code to the local clone, you should be set and can skip these steps and go straight to deploying.
   - If you haven't set up a git repository for your files yet (or didn't clone), proceed to the next step.
2. Run `git init` locally in the folder with your web server files.
   - If you want to also push to GitHub in addition to hosting on Heroku (recommended), you can follow the Adding Existing Projects to GitHub through the command line Guide
3. Commit all changes (if you haven't already with the above steps) using `git add .` and `git commit -am "<message>"`. If you haven't run into any errors at this point, you should be able to proceed to the next section.
   - You may also want to create a `.gitignore` file if you don't have one already. This file will allow you to tell git to not track files such as those in the `node_modules` folder.
   - The GitHub gitignore Documentation and this node gitignore example are useful resources for this, though it is pretty much as simple as:
     1. Before you commit, create a file called `.gitignore`. Inside of that file, add `node_modules/` as the first line and save the file. Now, git should no longer track `node_modules` files.

## Steps to Deploy

1. Log in to Heroku.
   - If you are a windows user open the cmd.exe (NOT Git Bash) and type `heroku login`. Keep this command prompt open in the background. Then, open Git Bash and navigate to the folder with your code.

- If you are a mac open terminal and type the command `heroku login`. Enter your Heroku credentials and proceed with all the below steps in terminal. Navigate to the folder with your code.

1. Run the command: `git remote -v`.
   - This is to show you that right now, you do not have heroku listed as a remote repository.
2. Run the command `heroku create`.
   - This will create an app instance on the Heroku server and will add heroku as a remote for your local git repository.
3. Run `git remote -v` again.
   - This isn't necessary, but helps to confirm that Heroku is now in your list of remotes. This time you should see the `heroku` remote.
4. Ensure that your `package.json` file is set up correctly. It must have a `start` script and all dependencies defined. E.g.:`{`

```
  "name": "starwars",
  "version": "1.0.0",
  "description": "Helps you find the characters you are looking for",
  "main": "server.js",
  "dependencies": {
    "body-parser": "^1.15.0",
    "express": "^4.13.4"
  },
  "scripts": {
    "start": "node server.js"
  }
}
```

5. Ensure your web server is starting with a dynamic port.
   - For an express app, the code for this would look like:
6. ```
   var PORT = process.env.PORT || 3000;
   ...
   app.listen(PORT, function() {
   ```
   - This allows you to get the port from the bound environment variable (using `process.env.PORT`) if it exists, so that when your app starts on heroku's machine it will start listening on the appropriate port.
   - You app will still run on port 3000 locally if you haven't set an environment variable.
7. Commit any changes you've made up until this point using `git commit -am "<message>"`
8. Run the command `git push heroku master`. A series of processes will be initiated. Once the process is complete note the name of the app.
9. Log in to your Heroku account at www.heroku.com . You will see a list or a (single) app. Note the one that has the same funky name as you saw in bash. Click on it.
10. Click on settings. Then, scroll down until you see the part that says: "Domains". Note the URL listed under Heroku Domain.
11. Finally, go in your browser to the URL listed under the Heroku Domain. If all went well you should see your website!
   ```