

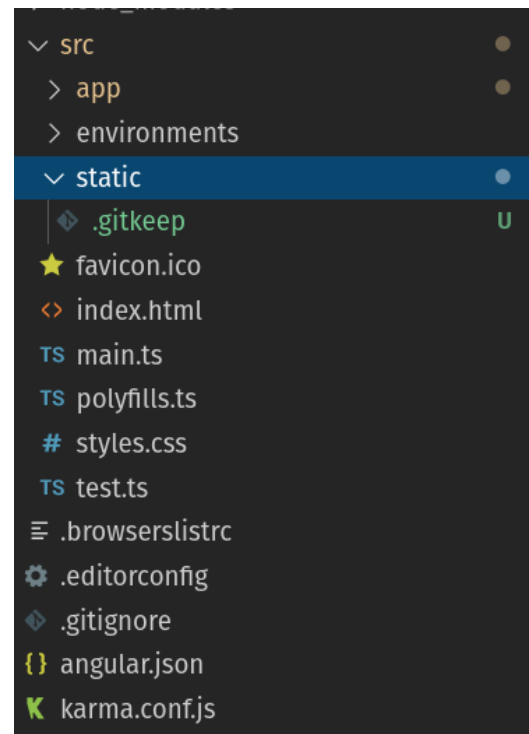
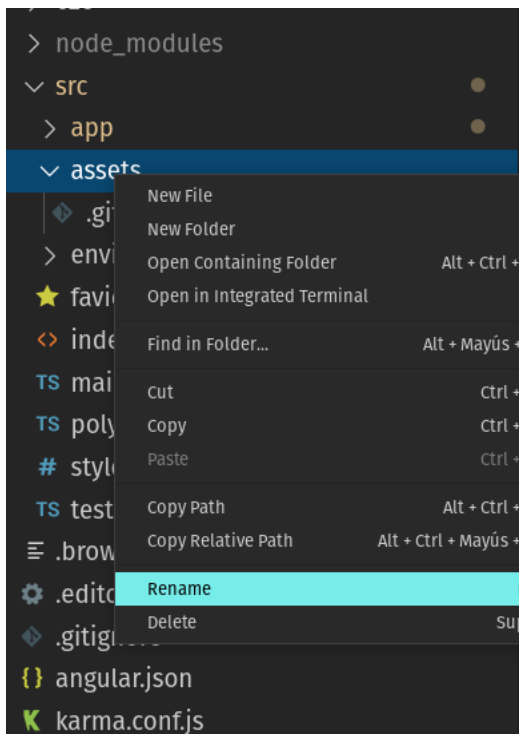


DESPLIEGUE DE UNA APP ANGULAR EN HEROKU USANDO FLASK COMO SERVIDOR

A continuación, desplegaremos una app Angular en la plataforma **Heroku**. Lo primero que debemos tener presente es que una aplicación Angular mientras se desarrolla involucra lenguajes como Typescript, múltiples componentes que creamos para dividir nuestra aplicación, archivos de configuración, entre otros, pero al final, cuando llevamos nuestra aplicación a producción, todo lo anterior no es más que un archivo `index.html` y archivos Javascript, sumando assets (íconos, imágenes). Todos son archivos estáticos, los cuales deben ser servidos por algún servidor web escrito en cualquier lenguaje (Php, Java, Python etc), nosotros usaremos Python y su framework **Flask** para servir nuestra App Angular.

Cuando construimos nuestra aplicación Angular para llevarla a producción, nuestras imágenes suelen estar ubicadas en la carpeta `assets`, ya que es la carpeta por defecto cuando creamos cualquier proyecto Angular. Lo que haremos a continuación será crear nuestra aplicación en Angular y hacer algunos ajustes antes de llevarla a producción con el objetivo de que se pueda servir desde **Flask**, luego, haremos el despliegue en **Heroku** para lo cual también usaremos **Git** y **GitHub**:

1. Pues bien, empecemos creando una aplicación Angular, la llamaremos **deploy**. Usamos el comando **ng new deploy**
2. A continuación nos ubicamos dentro de la carpeta de nuestra aplicación y crearemos tres componentes, **nav**, **body** y **footer**. Usamos los comandos **ng g c nav**, **ng g c body**, **ng g c footer** para crearlos.
3. Abrimos nuestra App en nuestro editor de código y cambiamos el nombre de la carpeta **assets** a **static**





A continuación, debemos editar el archivo **angular.json** de nuestro proyecto y “decirle” a Angular dónde estarán ubicados nuestros assets. Abrimos el archivo y cambiamos **assets** por **static** tal cual se muestra en la siguiente imagen:

```
"index": "src/index.html",
"main": "src/main.ts",
"polyfills": "src/polyfills.ts",
"tsConfig": "tsconfig.app.json",
"aot": true,
"assets": [
  "src/favicon.ico",
  "src/static"
],
"styles": [
  "src/styles.css"
],
"scripts": []
},
"configurations": {
  "production": {
    "fileReplacements": [
```

4. Ahora en el archivo **index.html** “linkeamos” bootstrap el cual podemos acceder desde este link : <https://getbootstrap.com/docs/4.5/getting-started/introduction/>



Programación de software.
Centro de Comercio y Turismo.
Regional Quindío. 2020
Instructor: Germán Alberto Angarita Henao



Quick start

Looking to quickly add Bootstrap to your project? Use jsDelivr, a free open source CDN. Using a package manager or need to download the source files? [Head to the downloads page](#).

CSS

Copy-paste the stylesheet `<link>` into your `<head>` before all other stylesheets to load our CSS.

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css" integrity="sha384-TX8t27EcRE3e/ihU7z" data-bbox="75 270 929 284">
```

JS

Many of our components require the use of JavaScript to function. Specifically, they require [jQuery](#), [Popper.js](#), and our own JavaScript plugins. We use [jQuery's slim build](#), but the full version is also supported.

Place **one of the following `<script>`s** near the end of your pages, right before the closing `</body>` tag, to enable them. jQuery must come first, then Popper.js, and then our JavaScript plugins.

Bundle

Include everything you need in one script with our bundle. Our `bootstrap.bundle.js` and `bootstrap.bundle.min.js` include [Popper](#), but not [jQuery](#). For more information about what's included in Bootstrap, please see our [contents](#) section.

```
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+0GpamoFVy38MVBnE+IbbVYUew+0r" data-bbox="75 493 929 505">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ho+j7jyWK8fNqE+A12Hb8AhRq26Lr" data-bbox="75 505 929 518">
```



```
<> index.html •
src > <> index.html > html > body > script
1  <!doctype html>
2  <html lang="en">
3  <head>
4    <meta charset="utf-8">
5    <title>Deploy</title>
6    <base href="/">
7    <meta name="viewport" content="width=device-width, initial-scale=1">
8    <link rel="icon" type="image/x-icon" href="favicon.ico">
9    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5
10 </head>
11 <body>
12   <app-root></app-root>
13   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity
14   <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/boots
15 </body>
16 </html>
17
```

5. Nos dirigimos al componente nav y eliminamos cualquier línea en él archivo **nav.component.html**, a continuación, pegamos el siguiente código:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Deploy</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Inicio<span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item active">
        <a class="nav-link" href="#">Despegue<span class="sr-only"></span></a>
      </li>
    </ul>
  </div>
</nav>
```



Repetimos el paso anterior análogamente para los componentes **footer** y **body**:

código del **footer**:

```
<div class="card text-center">
  <div class="card-body">
    <h5 class="card-title">Allá vamos!</h5>
  </div>
</div>
```

código del **body**:

```
<div class="jumbotron">
  <h1 class="display-4">Mi proyecto en marcha!</h1>
  
</div>
```

Acá en el body, la imagen **cohete.png** la puedes descargar desde:

<https://images.app.goo.gl/61QEEN9QsWpCRkKr6>

pero puedes usar cualquier imagen que quieras.

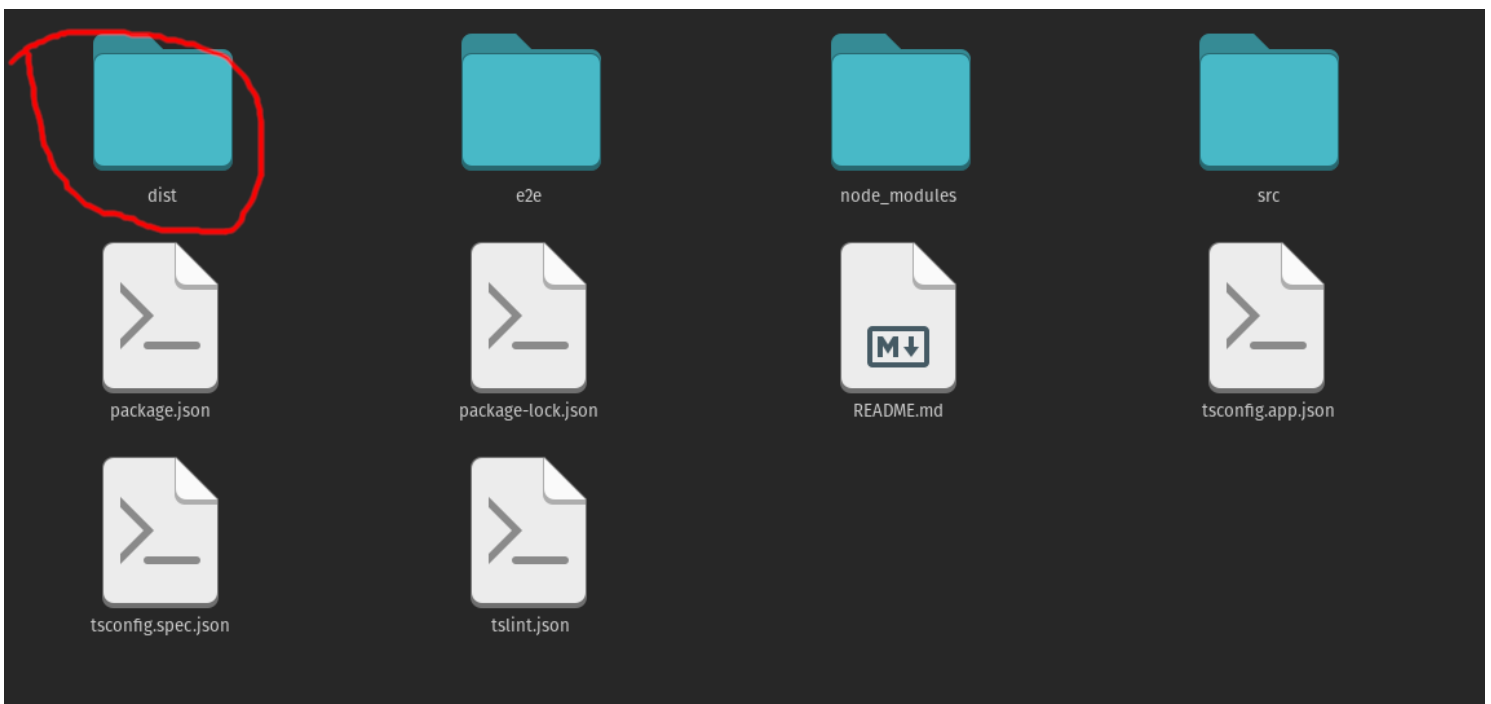
6. Vamos a **app.component.html** e insertamos el siguiente código para que nuestros componentes sean renderizados:

```
<app-nav></app-nav>
<app-body></app-body>
<app-footer></app-footer>
```

7. Lanzamos nuestra aplicación con el comando **ng serve** y nos dirigimos a: **http://localhost:4200/** debemos ver algo como lo siguiente:

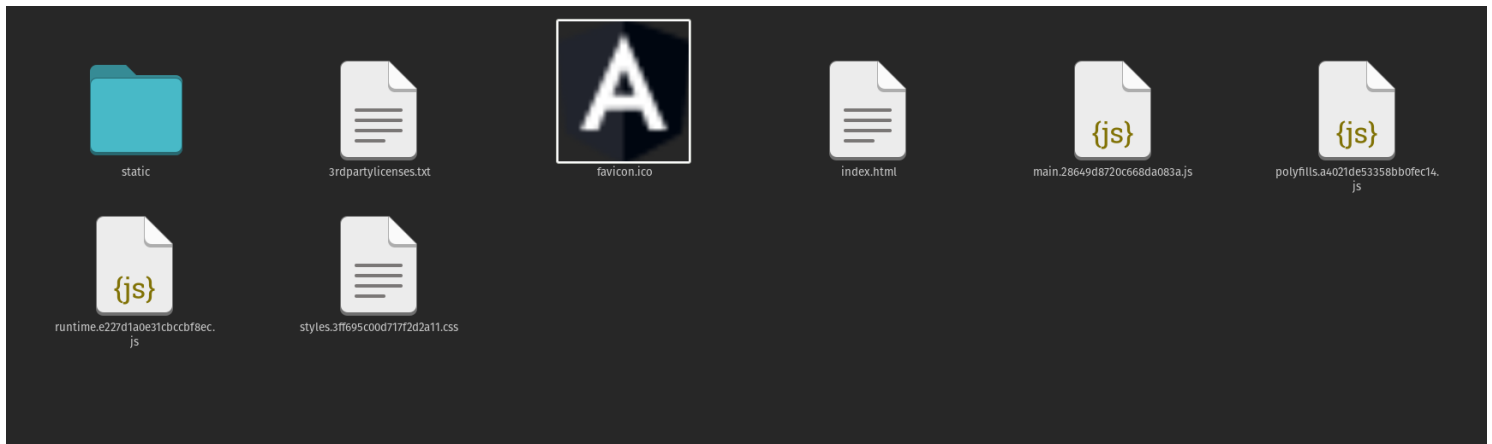


8. Ahora construimos nuestra aplicación para llevarla a producción, para ello usamos el comando **ng build --prod**. Lo anterior, nos creará la carpeta **dist**, dentro de ella veremos nuestro proyecto construido:

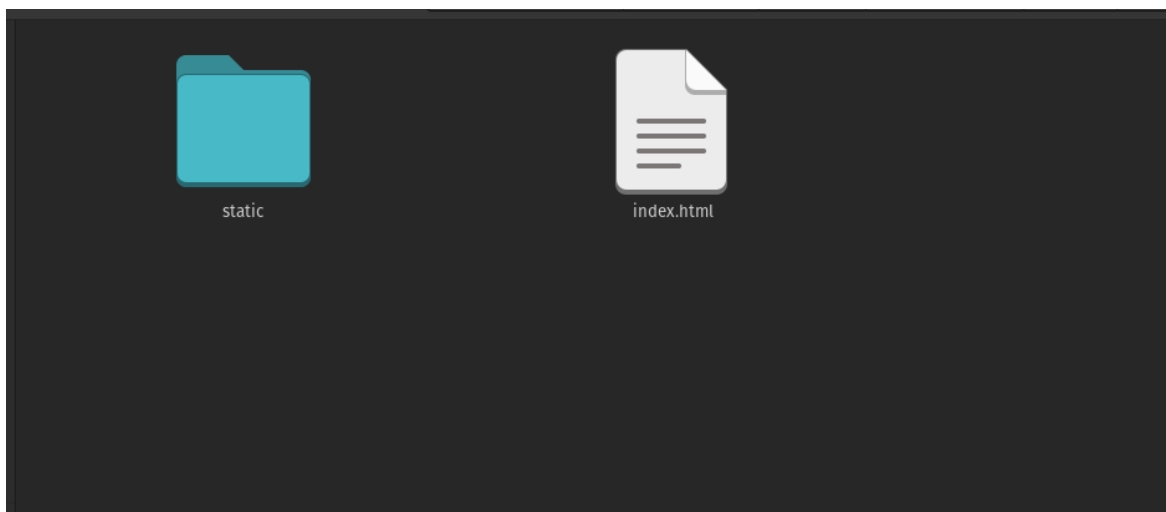




Si entramos a la carpeta dist, encontraremos la carpeta de nuestro proyecto construido, la cual debe verse con una estructura como la siguiente:



Pasamos todos los archivos con extensión **.js** y **.css** y el **favicon** a la carpeta **static**. Eliminamos el archivo de extensión **.txt** La estructura de nuestro proyecto debe quedarnos así:





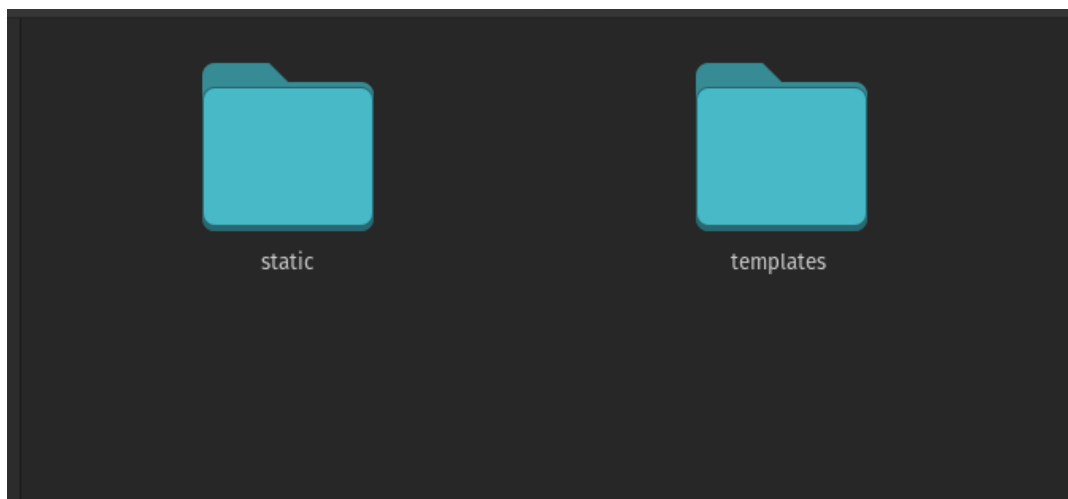
A continuación editamos el archivo **index.html** de la app **construida anteriormente** y modificamos las rutas de los archivos estáticos haciendo que estos se carguen desde la carpeta **static**. Nuestro **index.html** debe quedar de la siguiente manera:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Deploy</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="static/favicon.ico">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css" integrity="sha384-TX8t27E"
  <link rel="stylesheet" href="static/styles.3ff695c00d717f2d2a11.css"></head>
<body>
  <app-root></app-root>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+0GpamoFVy38MVBnE+
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ho+j7jyWK8fNQe+A12
  <script src="static/runtime.e227d1a0e31cbccbf8ec.js" defer></script><script src="static/polyfills.a4021de53358bb0fec14.js"
</html>
```

```
trap.min.css" integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous">

rtPH0lsSSs5nCTpuj/zy4C+0GpamoFVy38MVBnE+IbbVYUew+0rCXaRkfj" crossorigin="anonymous"></script>
s" integrity="sha384-ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGG0n+Y7RoweNrtn/E3MoK7ZeZDyx" crossorigin="anonymous"></script>
itic/polyfills.a4021de53358bb0fec14.js" defer></script><script src="static/main.28649d8720c668da083a.js" defer></script></body>
```

Ahora creamos la carpeta **templates** e incluimos en ella el archivo **index.html** Nuestro directorio de proyecto debe quedar de la siguiente manera:





9. Incluimos en el directorio de nuestra app construida el archivo **main.py** el cual es el servidor **Flask** que servirá nuestra aplicación Angular. Su código es el siguiente:

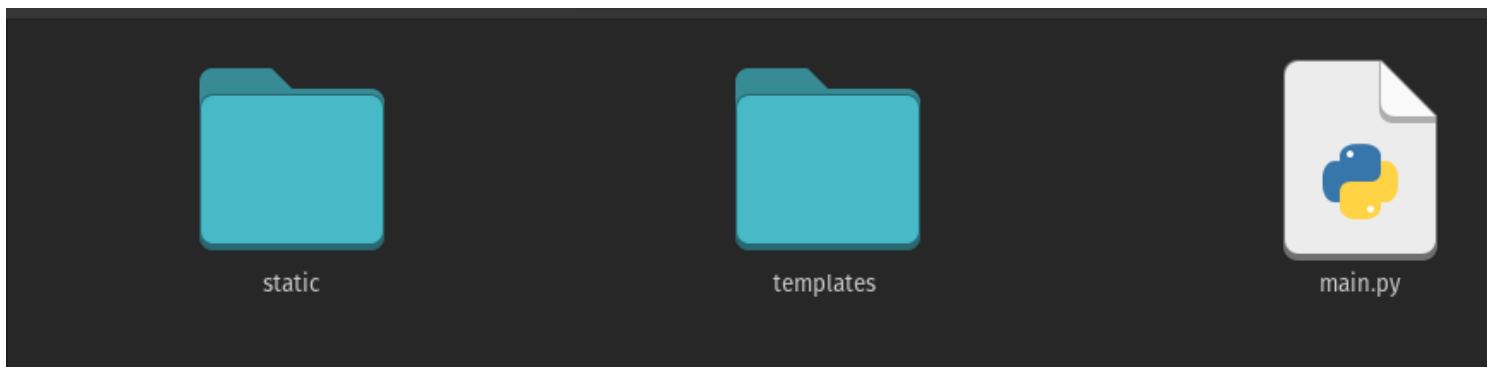
```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run()
```

Nuestro directorio debe quedar así:



ejecutamos nuestra aplicación **Flask** y nos dirigimos con el navegador a <http://localhost:5000> Si todo ha ido como esperamos, debemos ver nuestra App Angular servida por **Flask** así:



Ahora debemos agregar dos archivos a nuestro directorio de proyecto, son necesarios para el despliegue en la plataforma **Heroku**: **requirements.txt** y **Procfile**.

Creemos el archivo **requirements.txt** y agregamos lo siguiente:

```
Click==7.0
Flask==1.1.2
gunicorn==20.0.4
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==1.0.1
```

Lo anterior son los requerimientos que le pasaremos al sistema de despliegue de **Heroku** para que sepa qué dependencias debe instalar para poner en producción nuestra App.

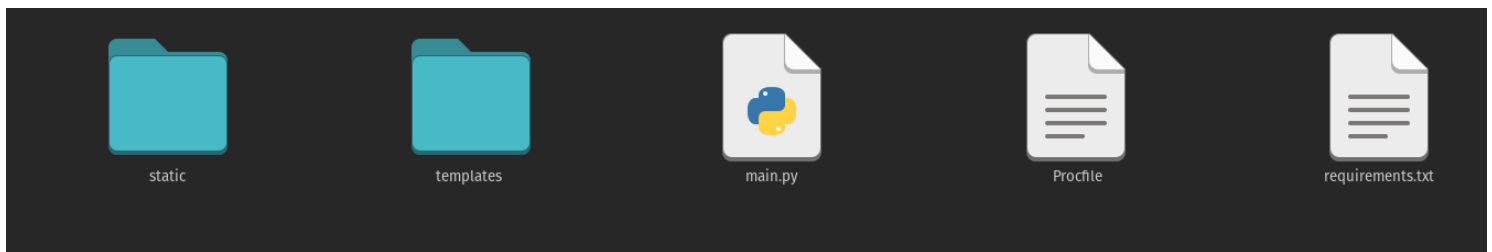
Creemos el archivo **Procfile** (sin ninguna extensión en especial) y agregamos la siguiente línea:

```
web: gunicorn main:app
```

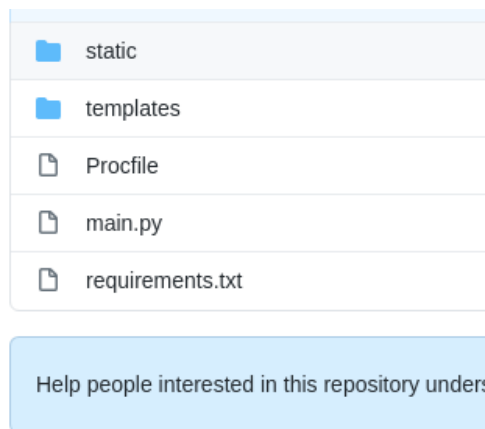


gunicorn es el servidor de producción que implementaremos en **Heroku** para servir nuestra App con **Flask**, ya que el servidor que **Flask** trae por defecto nunca se debe usar en entornos de producción, sólo en desarrollo.

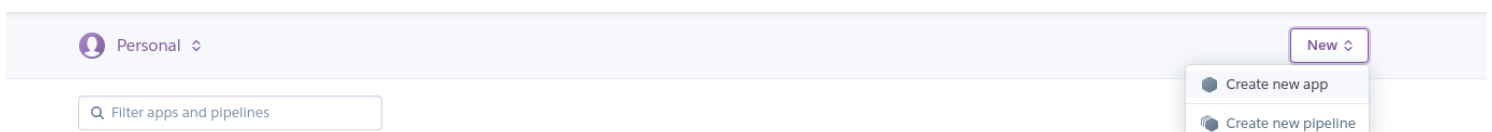
Luego de esto, nuestro directorio debe verse así:



10. A esta altura nuestra app está lista para ser desplegada, llevada a producción. Primero debemos crear un repositorio en **GitHub** de nuestra aplicación. Asegurémonos de que todos los archivos y carpetas han sido incluidos en el repositorio. En **GitHub** debemos ver algo así en nuestro repositorio:



11. Una vez cumplamos con los pasos anteriores, pasamos a registrarnos en **Heroku**: <https://www.heroku.com/>. Iniciamos sesión y damos click en **New** y luego en **Create new app**





Programación de software.
Centro de Comercio y Turismo.
Regional Quindío. 2020
Instructor: Germán Alberto Angarita Henao



Le asignamos un nombre que no esté en uso a nuestra App(en este caso se le ha puesto deploy-flask-angular) y damos click en **Create app**

Create New App

App name

deploy-flask-angular

✓

deploy-flask-angular is available


Choose a region


United States


Add to pipeline...

Create app

Conectamos **Heroku** con **GitHub**, otorgamos permisos:

 Heroku Git
Use Heroku CLI

 GitHub
Connect to GitHub

 Container Registry
Use Heroku CLI

View your code diffs on GitHub

Connect your app to a GitHub repository to see commit diffs in the activity log.

Deploy changes with GitHub

Connecting to a repository will allow you to deploy a branch to your app.

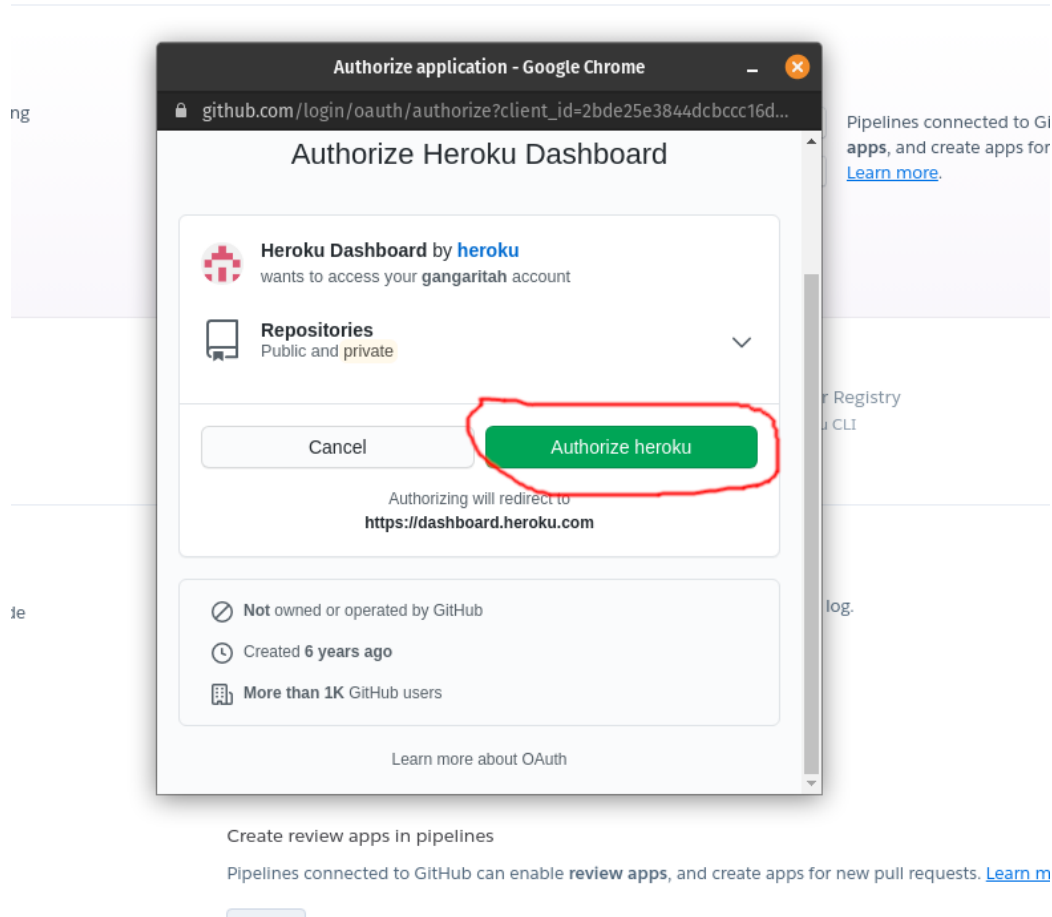
Automatic deploys from GitHub

Select a branch to deploy automatically whenever it is pushed to.

Create review apps in pipelines

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. [Learn more.](#)

Connect to GitHub



Luego de otorgar la autorización para que **Heroku** se conecte con nuestro repositorio en **GitHub**, seleccionamos **GitHub** como método de despliegue:



Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features



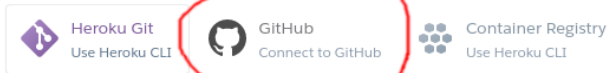
Pipelines let you connect multiple apps together and **promote code** between them. [Learn more.](#)



Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. [Learn more.](#)

Choose a pipeline

Deployment method



A continuación, en la barra de búsqueda ponemos el nombre(no confundir con la URL) de nuestro repositorio, el cual se encuentra ligado a nuestra cuenta de **GitHub**(la cual aparece a la derecha), una vez nuestro repositorio haya sido encontrado, damos click en **Connect**:



Search for a repository to connect to

gangeritah

deploy

Search

Missing a GitHub organization? [Ensure Heroku Dashboard has team access.](#)

gangeritah/deploy

Connect

Una vez hecho esto nos aparece lo siguiente:



Programación de software.
Centro de Comercio y Turismo.
Regional Quindío. 2020
Instructor: Germán Alberto Angarita Henao



Connected to [gangaritah/deploy](#) by [gangaritah](#) [Disconnect...](#)

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

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

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](#).

Choose a branch to deploy

☐ 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](#)

Deploy a GitHub branch

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

Choose a branch to deploy

[Deploy Branch](#)

damos click en **Enable Automatic Deploys** (esto hará que **Heroku** realice despliegues automáticos cada vez que ocurre una actualización en el repositorio de nuestra App) y luego click en **Deploy Branch**, esperamos un momento a que **Heroku** instale las dependencias necesarias y despliegue nuestra App.

Una vez hemos realizados los pasos anteriores, nuestra App se encuentra desplegada. Nos debe aparecer lo siguiente:

Deploy a GitHub branch

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

Choose a branch to deploy

[Deploy Branch](#)

Receive code from GitHub



Build **master** 04a0fea8



Release phase



Deploy to Heroku



Your app was successfully deployed.

[View](#)



Programación de software.
Centro de Comercio y Turismo.
Regional Quindío. 2020
Instructor: Germán Alberto Angarita Henao



Damos click en **View** y se nos debe abrir en una ventana del navegador nuestra App Angular servida por **Flask**:

