



Cómo hacer un deploy

Objetivo

Siguiendo estos pasos vamos a poder subir **nuestro proyecto de Mercado Libre** a **Heroku** para que todos lo puedan ver.

Antes de empezar...

Antes de hacer cualquier cosa, asegurémonos de tener listos estos **4 puntos**:

1. Un proyecto subido a un repo de [GitHub](#).
2. Una cuenta en [Heroku](#).
3. **Puertos** configurados.*
4. Archivo **procfile**.*

* Configuración del puerto

Si utilizamos **express-generator** para crear nuestro proyecto, es posible que no necesitemos seguir esta indicación. Si iniciamos nuestro proyecto instalando Express a través de npm y así las diferentes dependencias, probablemente tengamos que hacer un pequeño cambio dentro de la configuración del puerto donde va a correr nuestro proyecto.

Normalmente utilizamos el puerto **3000**, pero, a la hora de publicar el proyecto dentro de **Heroku**, esta plataforma establece otro diferente. En ese caso, vamos a cambiar el número de puerto dentro del **listen** a la siguiente línea:

// CAMBIAR ESTE BLOQUE:

```
app.listen(3000, function() {  
  console.log('Servidor corriendo en el puerto 3000');  
})
```

// POR ESTE BLOQUE

```
app.listen(process.env.PORT || 3000, function() {  
  console.log('Servidor corriendo en el puerto 3000');  
})
```

Heroku utiliza [variables de entorno](#) para guardar cierta información. Dentro de la variable llamada **PORT** estará guardado aquel dato que necesitamos. Entonces, ¿podríamos traducir esa línea que cambiamos al español? Claro que sí:

```
app.listen(**Si existe la variable PORT, usá ese dato** || **Sino directo  
3000**, function() {  
  console.log('Servidor corriendo en el puerto 3000');  
})
```

* Procfile

Otra cosa que Heroku necesita es un **Procfile**. Esto le dice a Heroku que comandos ejecutar para iniciar nuestro sitio web. Para ello, hace falta crear un archivo llamado **Procfile** (sin extensión) en la **raíz del proyecto** y guardar dentro la siguiente línea:

- web: node rutaAlArchivoPrincipal.js

```
web: node app.js  
web: node ./src/app.js
```

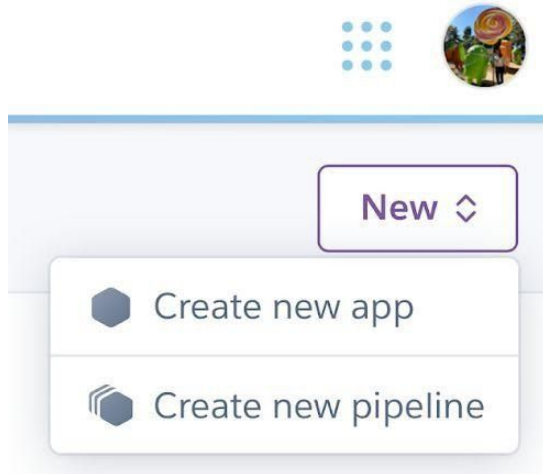
Esta línea significa que vamos a desplegar una aplicación **web** ejecutando el comando **gunicorn mysite.wsgi** (**Gunicorn** es un programa que es la versión más poderosa del comando de Django **runserver**).

Para finalizar, tan solo habrá que guardarlo y ¡listo!



Paso 1

Crear una nueva app en Heroku.





Paso 2

Ponerle un **nombre** a la **app**. El nombre que elijamos es el nombre que vamos a usar para acceder a la página. Ejemplo: *mi-app-de-node.herokuapp.com*.

En la **región**, dejar "United States" y hacer click en **Create App**.

Create New App

App name

app-name

Choose a region



United States



Add to pipeline...

Create app



Paso 3

En la sección **Deploy** hacer click en el botón de **GitHub** para conectar su repositorio.



Luego, tendremos que **conectar** nuestra **cuenta** y elegir qué repositorio queremos publicar.

Search for a repository to connect to

violetbaggins

repo-name

Search

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

violetbaggins/violetbaggins.github.io	Connect
violetbaggins/angular_suelos	Connect
violetbaggins/mundoparlantedashboard	Connect
violetbaggins/elefantecosmico	Connect
violetbaggins/practica-register-login	Connect

Una vez encontremos nuestro repositorio, presionamos **Connect** y debería aparecer esto:



Heroku Git
Use Heroku CLI



GitHub
Connected



Container Registry
Use Heroku CLI

Connected to [violetbaggins/MP_JSON](#) by [violetbaggins](#)

Disconnect...

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



Paso 4

Ahí mismo, en la sección **Deploy**, pero más abajo, está **Automatic Deploys**. Hacer click en **Enable Automatic Deploys** para que cada vez que hagamos un push a nuestro repositorio, nuestra página se actualice.



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



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



Paso 5

Ahora, en la sección **Settings**, debemos ir a **Buildpacks** y presionar **Add Buildpack**.

Add buildpack










En la ventana que se abre, elegir **Node.js** y guardar los cambios.

Add Buildpack ×

Enter Buildpack URL

heroku/nodejs

Or select from our officially supported buildpacks

 nodejs	 python	 php	 ruby	 java
 go	 gradle	 scala	 clojure	

Save changes



Paso 6

Por último, vamos de nuevo a la sección **Deploy** y le damos al botón **Deploy Branch**. Su aplicación va a comenzar el deploy y, cuando termine, veremos algo así:

Deploy a GitHub branch

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

Choose a branch to deploy

 master 

Deploy Branch

Receive code from GitHub



Build master c2c53e62



Release phase



Deploy to Heroku



Your app was successfully deployed.

 View



Paso 7

Haciendo click en el botón de **VIEW**, ¡vamos a poder ver nuestro sitio online!

¡Happy Deploying!