

# Proyecto colaborativo en servidor Parte I

<b>Proyecto colaborativo en servidor Parte I</b>	<b>1</b>
<b>Generación de proyecto base con un framework</b>	<b>1</b>
<b>Preparando el entorno de desarrollo Visual studio code</b>	<b>2</b>
Instalando extensiones recomendadas	2
Refrescado el proyecto (opcional)	3
<b>Probando nuestra aplicación en local</b>	<b>4</b>
<b>Probando que la aplicación funciona en local</b>	<b>4</b>
<b>Configurando la aplicación para poder subirla correctamente al servidor Heroku</b>	<b>5</b>
Solucionando problemas con el puerto	5
Solucionando problemas con JAVA_HOME	6
<b>Subiendo la aplicación a un repositorio de GitHub</b>	<b>6</b>
<b>Deployando nuestra aplicación a un servidor real (heroku)</b>	<b>8</b>
Subiendo el código fuente alojado en github a heroku.	9

NOTA: El proyecto **completo** está subido al repositorio siguiente de github:

<https://github.com/kant003/transformatoTu.git>

## Generación de proyecto base con un framework

Vamos a usar como framework de desarrollo Spring Boot

Desde la pagina <https://start.spring.io/> podemos generar una especie de hola mundo básico de un proyecto vacío de **Spring Boot**

Rellena los parámetros:

- Gestor de dependencias: **Maven**
- Lenguaje para programar: **Java**
- Group: **com.prueba** ← puedes elegir tu el nombre
- Artifact y Name: **demo** ← puedes elegir tu el nombre
- Packaging: **jar**
- Versión de java: **11**

Añade también la dependencia:

- Spring web

Project

☒ Maven Project
 ☐ Gradle Project

Language

☒ Java
 ☐ Kotlin
 ☐ Groovy

Spring Boot

☐ 2.5.0 (SNAPSHOT)
 ☐ 2.5.0 (M3)
 ☐ 2.4.5 (SNAPSHOT)
 ☒ 2.4.4
 ☐ 2.3.10 (SNAPSHOT)
 ☐ 2.3.9

Project Metadata

Group

com.prueba

Artifact

demo

Name

demo

Description

Demo project for Spring Boot

Package name

com.prueba.demo

Packaging

☒ Jar
 ☐ War

Java

☐ 16
 ☒ 11
 ☐ 8

Dependencies

ADD DEPENDENCIES... % + B

Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Esto nos generará un archivo comprimido .zip con todos los archivos necesarios.

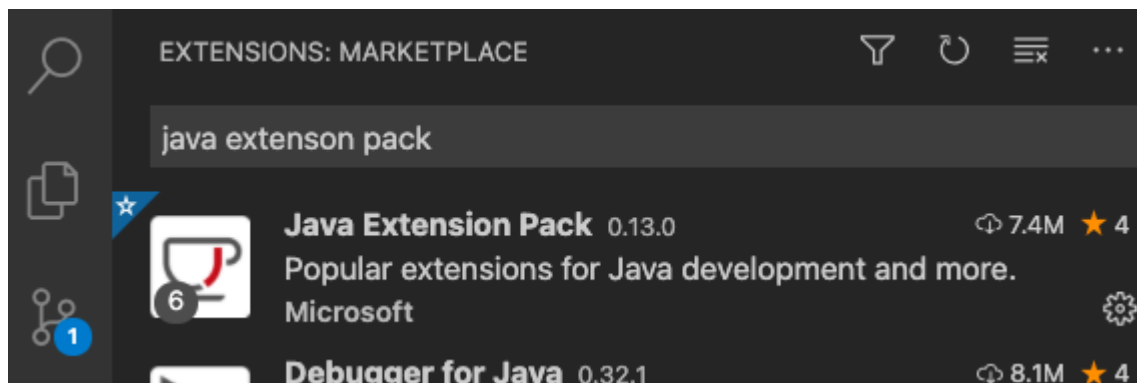
- Descomprimelo en una carpeta de tu disco duro.
- Edítalo con tu editor de código favorito

## Preparando el entorno de desarrollo Visual studio code

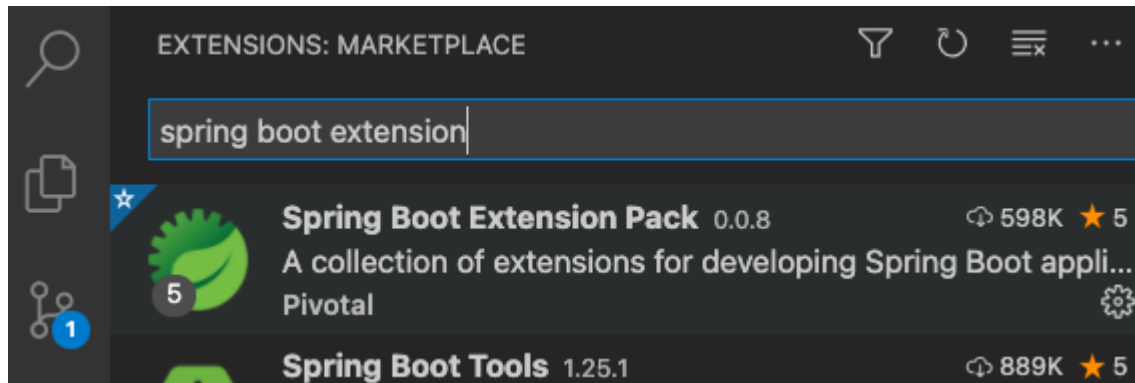
### Instalando extensiones recomendadas

Para trabajar correctamente con el editor de código **Visual Studio Code** tendrás que instalar las siguientes extensiones:

#### Java Extensión Pack



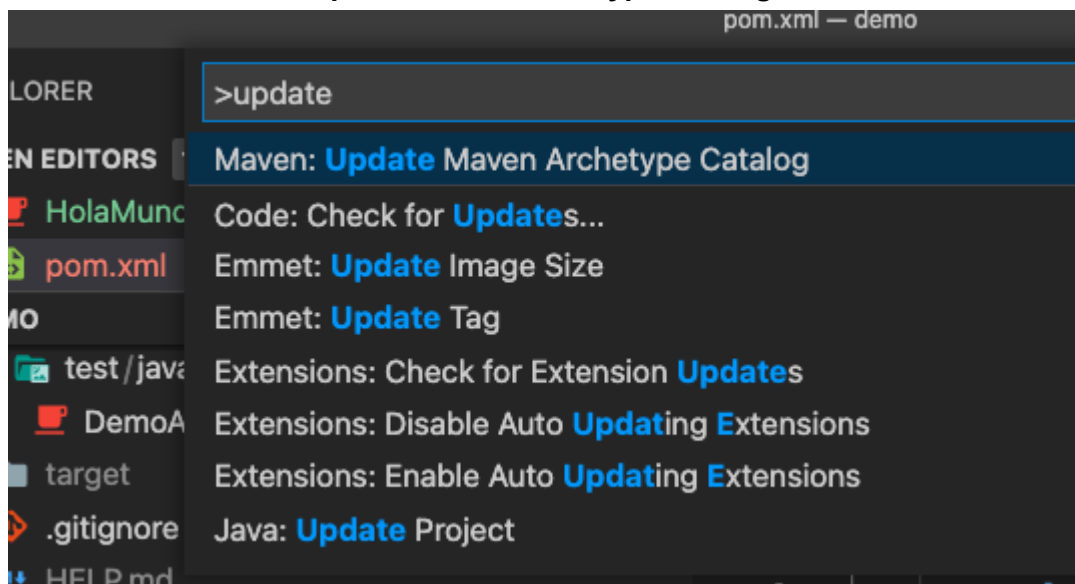
## Sprint Boot Extensión Pack



## Refrescado el proyecto (opcional)

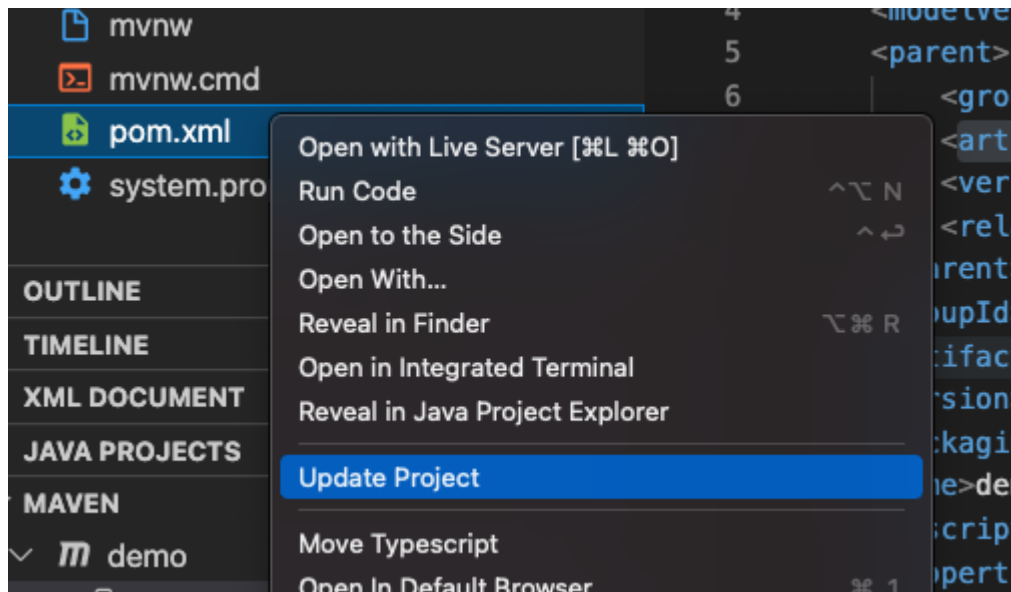
**Actualiza** el catálogo repositorios de maven:

- Pulsa F1
- Busca "**Maven: Update Maven Archetype Catalog**"



**Refresca** el proyecto:

- Pulsa con el botón derecho en el fichero ***pom.xml*** y selecciona "**Update project**"



## Probando nuestra aplicación en local

Sitúate en la carpeta de tu proyecto y ejecuta el siguiente comando:

Linux y mac:

```
./mvnw spring-boot:run
```

Windows:

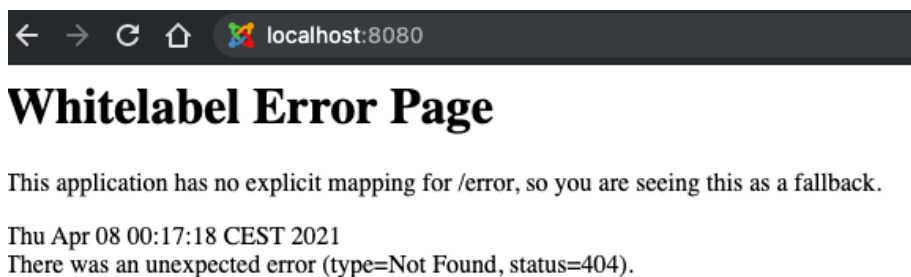
```
mvnw.cmd spring-boot:run
```

Si todo va bien, se ha iniciado un servidor web en tu máquina local en el puerto 8080

## Probando que la aplicación funciona en local

Accede desde tu navegador a la dirección <http://localhost:8080>

Si ves esta página web, tu aplicación está funcionando correctamente

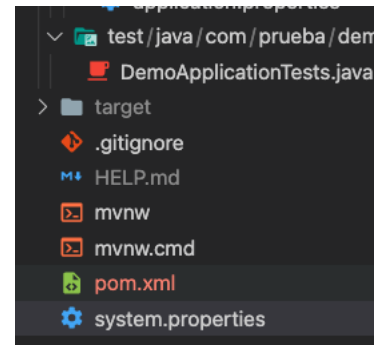


# Configurando la aplicación para poder subirla correctamente al servidor Heroku

Uno de los pasos que haremos más adelante es publicar (deploy) nuestra aplicación en un servidor real (en concreto Heroku).

Como nuestra aplicación usa la versión 11 de Java (jdk) para trabajar, debemos indicarle a heroku por medio de un fichero de configuración.

Para ello, crea un fichero llamado **system.properties** en la raíz de tu proyecto (*junto al fichero pom.xml*), con el siguiente contenido:



```
java.runtime.version=11
```

## Solucionando problemas con el puerto

Podría ocurrir que tengas otro servidor web ya arrancado en el puerto **8080** y nos de un error (conflicto).

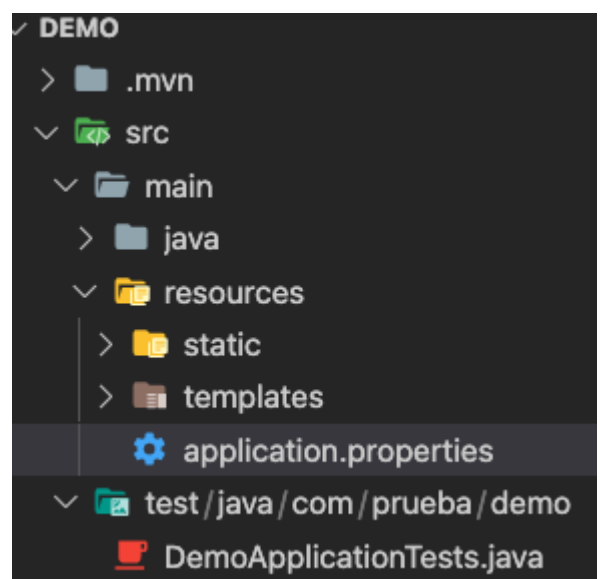
Para solucionarlo vamos a indicarle a spring boot que use otro puerto para funcionar.

Edita el fichero **application.properties**

Añade como contenido:

```
#Inicia la aplicación en el puerto 8888
server.port = 8888
```

donde **8888** será el puerto que quieres usar



## Solucionando problemas con JAVA\_HOME

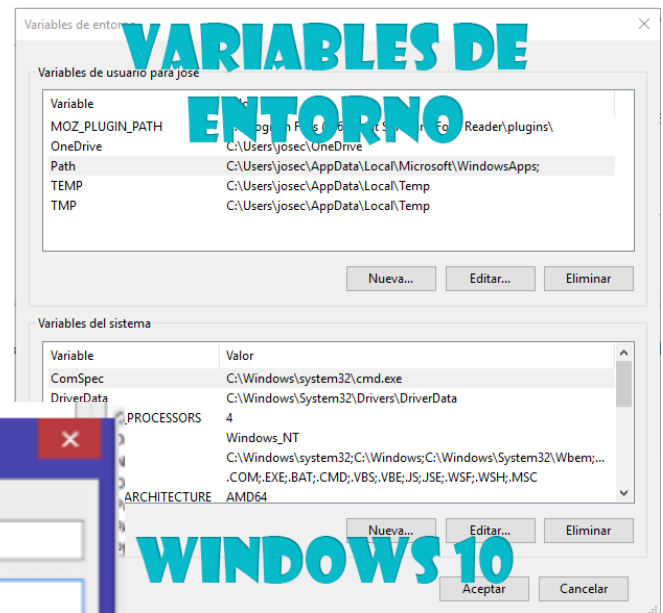
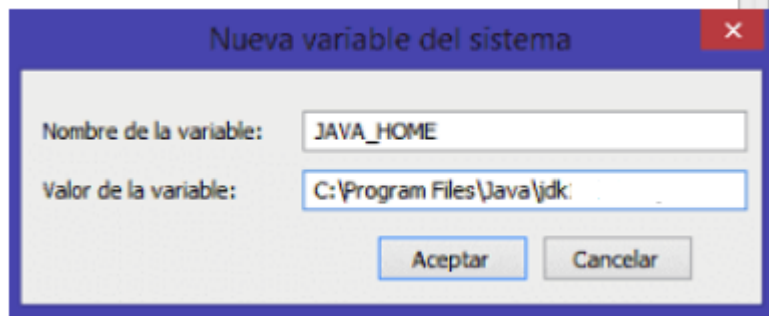
También puede ocurrir que la variable de entorno **JAVA\_HOME** no esté configurada correctamente.

En windows ve a la configuración de las variables de entorno y añade una nueva con:

clave = **JAVA\_HOME**

valor = ruta de tu instalación de java (jdk)

**C:\Program Files\Java\jdkXXXXXX**



## Subiendo la aplicación a un repositorio de GitHub

Vamos a subir la aplicación web al repositorio remoto de **GitHub**.

Para ello primero tenemos que crear un nuevo repositorio. Ponle el nombre que más te guste.

## Create a new repository

A repository contains all project files, including the revision history. Already have a repository elsewhere? [Import a repository](#).

Owner \*

 kant003 ▾

Repository name \*

/ tuAplicacion ✓

Great repository names are short and memorable. Need inspiration? How about [vel](#)

Description (optional)


☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

### Quick setup — if you've done this kind of thing before

 Set up in Desktop

or

HTTPS

SSH

<https://github.com/kant003/tuAplicacion>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every

### ...or create a new repository on the command line

```
echo "# temp" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/kant003/temp.git
git push -u origin main
```

Accede a la carpeta donde está guardada tu aplicación y ejecuta los siguientes comandos para subir tu app a github:

**# git init**

**# git add .**

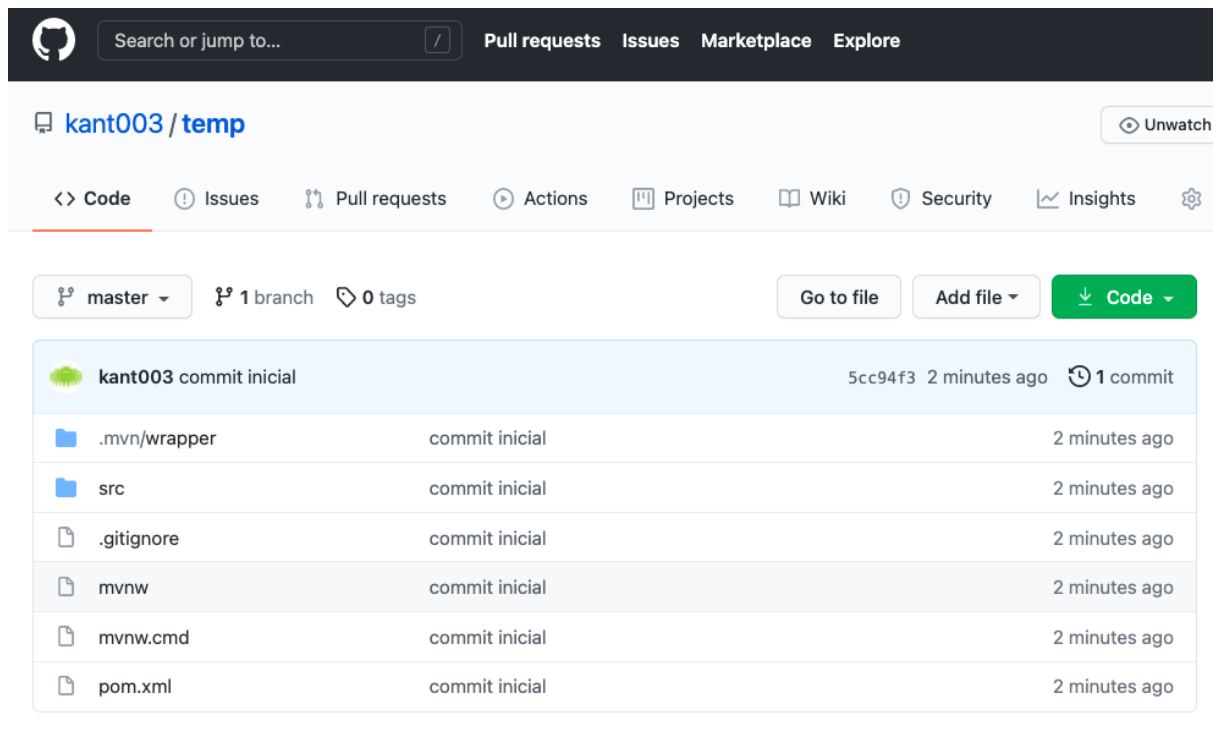
**# git commit -m "primer commit"**

**# git remote add origin <https://github.com/xxxx/tuAplicación.git>**

**# git push -u origin master**

Tendrás que cambiar la url <https://github.com/xxxx/tuAplicación.git> por la url de tu repositorio.

Comprueba en la página de github, que se ha subido el proyecto correctamente.



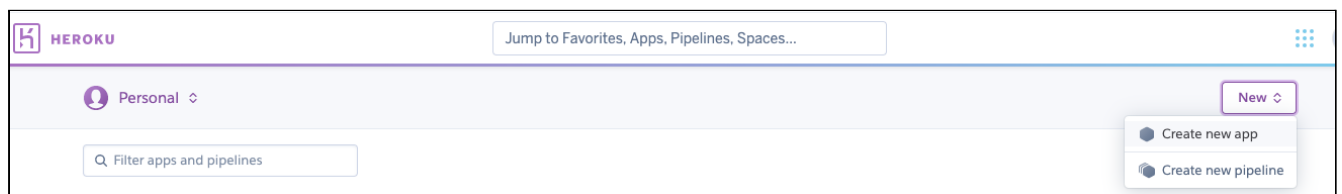
## Deployando nuestra aplicación a un servidor real (heroku)

Date de alta en la web de heroku

[www.heroku.com](https://www.heroku.com)

Crea una nueva aplicación

Pulsando en new -> Create new app



Asigne un **nombre** cualquiera a tu aplicación y selecciona como localización **Europa**



App name

tempangel

tempangel is available

Choose a region

Europe

Add to pipeline...

Create app

## Subiendo el código fuente alojado en github a heroku.

Tu código fuente está guardado en un repositorio remoto de GitHub, vamos a enviar la información desde GitHub a Heroku.

Para ello, conecta github a heroku pulsando el botón **"GitHub Connect to GitHub"**

HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

Personal > tempangel

Overview Resources **Deploy** Metrics Activity Access Settings

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 deployments.

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

Choose a pipeline

Deployment method

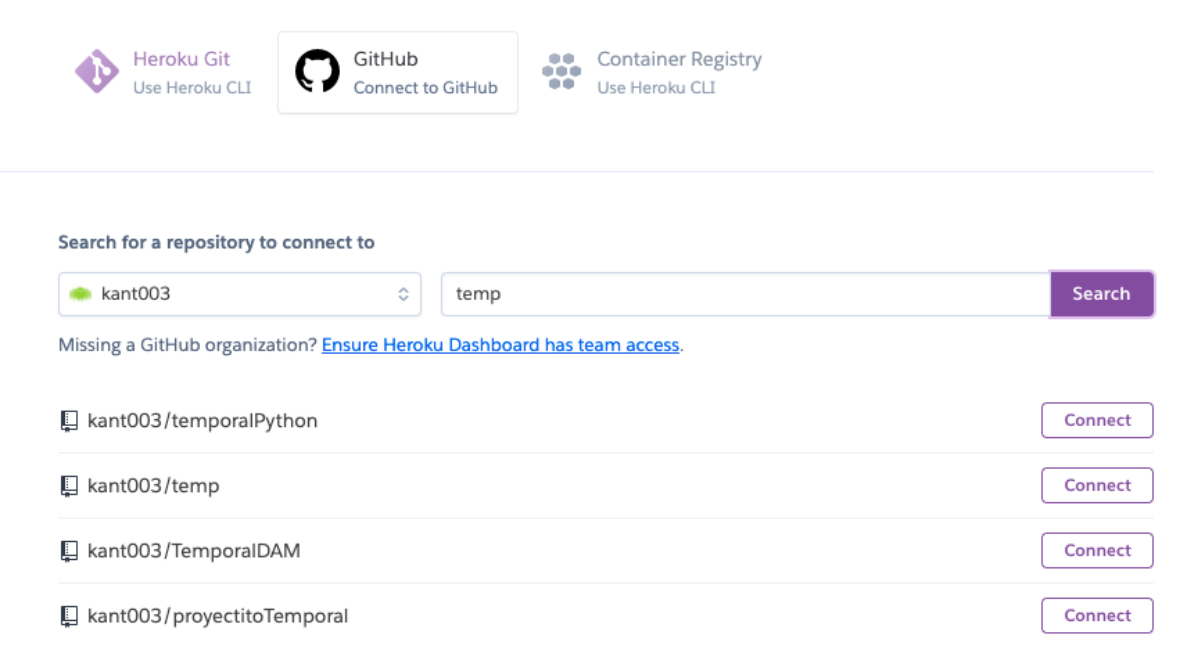
Heroku Git

Use Heroku CLI

GitHub

Connect to GitHub

Busca el nombre de tu repositorio y pulsa en **"Connect"**



The screenshot shows the Heroku Connect interface. At the top, there are three options: Heroku Git (Use Heroku CLI), GitHub (Connect to GitHub), and Container Registry (Use Heroku CLI). Below this is a search bar with the text "Search for a repository to connect to". The search results show four repositories: kant003/temporalPython, kant003/temp, kant003/TemporalDAM, and kant003/proyectitoTemporal. Each repository has a "Connect" button next to it.

Heroku Git Use Heroku CLI

GitHub Connect to GitHub

Container Registry Use Heroku CLI

Search for a repository to connect to

kant003 temp Search

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

kant003/temporalPython Connect

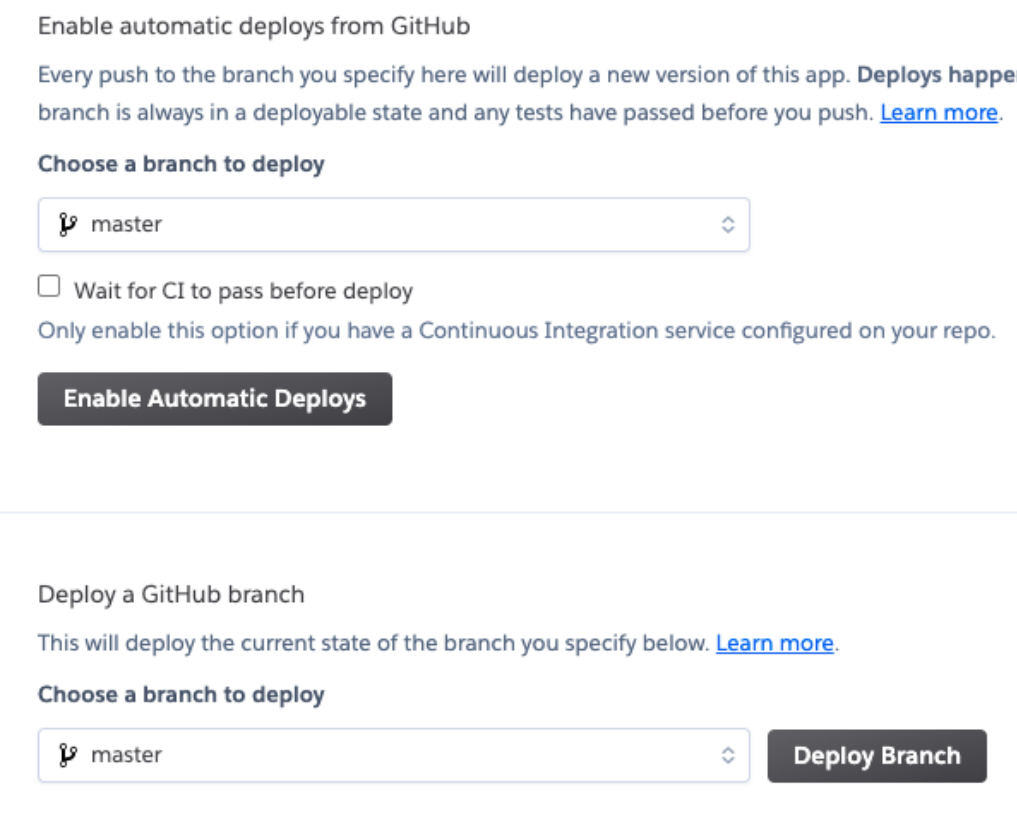
kant003/temp Connect

kant003/TemporalDAM Connect

kant003/proyectitoTemporal Connect

Para deployar el proyecto, basta con pulsar en el botón **"deploy branch"**

Además, si quiere que cada vez que se suban cambios a la rama master de github se publiquen automáticamente los cambios, pulsa el botón **"Enable automatics deploys"**



The screenshot shows the Heroku Deploy Branch interface. It has two sections: "Enable automatic deploys from GitHub" and "Deploy a GitHub branch". The first section has a dropdown menu for "Choose a branch to deploy" set to "master", a checkbox for "Wait for CI to pass before deploy", and a button "Enable Automatic Deploys". The second section has a dropdown menu for "Choose a branch to deploy" set to "master" and a button "Deploy Branch".

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. Deploys happen every time you push to the branch you specify here. The 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

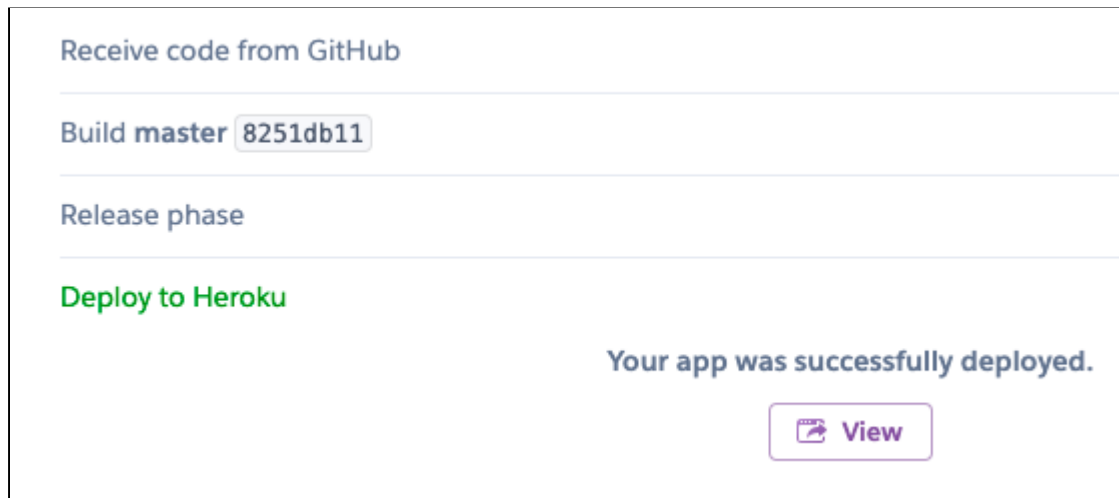
Deploy a GitHub branch

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

Choose a branch to deploy

master Deploy Branch

Si aparece el mensaje "**Your app was successfully deployed**" el proyecto se habrá subido correctamente al servidor.



Ya puedes ver el proyecto alojado en el servidor heroku, pulsando en el botón **View**