# **¡Desplegar!**

**Nota** El siguiente capítulo puede ser, a veces, un poco difícil de seguir. Se persistente y acábalo. El despliegue es una parte importante del proceso en el desarrollo web. Este capítulo está situado en el medio del tutorial para que tu guía pueda ayudarte a poner tu sitio web en línea, lo que puede ser un proceso algo más complicado.

Hasta ahora tu sitio web estaba disponible sólo en tu computadora, ¡ahora aprenderás cómo desplegarlo! El despliegue es el proceso de publicar tu aplicación en internet para que la gente pueda acceder y ver tu aplicación :).

Como ya has aprendido, un sitio web tiene que estar en un servidor. Hay muchos proveedores de servidores disponibles en Internet. Vamos a utilizar uno que tiene un proceso de despliegue relativamente sencillo: [PythonAnywhere](https://www.pythonanywhere.com/). PythonAnywhere es gratis para pequeñas aplicaciones que no tienen demasiados visitantes, lo que es definitivamente suficiente para este caso.

El otro servicio externo que vamos a utilizar es [GitHub](https://www.github.com/), un servicio de alojamiento de código. Hay otras opciones por ahí, pero hoy en día casi todos los programadores tienen una cuenta de GitHub, ¡y ahora tú también la vas a tener!

Estos tres lugares serán importantes. Tu computadora local será el lugar de desarrollo y pruebas. Cuando estés contento con los cambios, pondrás una copia de tu programa en Github. Tu sitio web estará en PythonAnywhere y lo actualizarás obteniendo una nueva copia del código desde GitHub.

# **Git**

**Nota** Si ya has hecho los pasos de instalación, no hace falta que hagas esto otra vez. Puedes avanzar a la siguiente sección y empezar a crear tu repositorio Git.

Git es un "sistema de control de versiones" que utilizan muchos programadores. Este software puede rastrear los cambios realizados en archivos a lo largo del tiempo de forma que puedas recuperar una versión específica más tarde. Es un poco parecido a la opción de "control de cambios" de Microsoft Word, pero mucho más potente.

## **Instalar Git**

**Instalar Git: Windows**

Puedes descargar Git de [git-scm.com](https://git-scm.com/). Puedes hacer clic en "Next" para todos los pasos excepto en uno; en el quinto paso titulado "Adjusting your PATH environment", elije "Run Git and associated Unix tools from the Windows command-line" (la última opción). Aparte de eso, los valores por defecto están bien. "Checkout Windows-style, commit Unix-style line endings" también está bien.

**Instalar Git: OS X**

Descarga Git de [git-scm.com](https://git-scm.com/) y solo sigue las instrucciones.

**Nota** Si estás en OS X 10.6, 10.7 o 10.8, necesitarás instalar la versión de Git desde aquí: [instalador Gitpara OS X Snow Leopard](https://sourceforge.net/projects/git-osx-installer/files/git-2.3.5-intel-universal-snow-leopard.dmg/download)

**Instalar Git: Debian o Ubuntu**

$ sudo apt-get install git

**Instalar Git: Fedora**

$ sudo dnf install git

**Instalar Git: openSUSE**

$ sudo zypper install git

## **Iniciar nuestro repositorio Git**

Git rastrea los cambios realizados a un grupo determinado de archivos en lo que llamamos un repositorio de código (abreviado "repo"). Iniciemos uno para nuestro proyecto. Abre la consola y ejecuta los siguientes comandos en el directorio de djangogirls:

**Nota** Comprueba el directorio de trabajo actual con el comando pwd (OSX/Linux) o cd(Windows) antes de inicializar el repositorio. Deberías estar en la carpeta djangogirls.

$ git init  
Initialized empty Git repository in ~/djangogirls/.git/  
$ git config --global user.name "Your Name"  
$ git config --global user.email you@example.com

Inicializar el repositorio git es algo que sólo necesitamos hacer una vez por proyecto (y no tendrás que volver a poner tu usuario y correo electrónico nunca más).

Git llevará un registro de los cambios realizados en todos los archivos y carpetas en este directorio, pero hay algunos archivos que queremos que ignore. Esto lo hacemos creando un archivo llamado .gitignore en el directorio base. Abre tu editor y crea un nuevo archivo con el siguiente contenido:

\*.pyc  
\*~  
\_\_pycache\_\_  
myvenv  
db.sqlite3  
/static  
.DS\_Store

Y guárdalo como .gitignore en la carpeta djangogirls.

**Nota** ¡El punto al principio del nombre del archivo .gitignore es importante! Si tienes dificultades para crearlo (a los Mac no les gusta que crees ficheros que empiezan por punto desde Finder, por ejemplo), usa la opción "Guardar como" en tu editor, eso no falla.

**Nota** Uno de los archivos especificados en tu .gitignore es db.sqlite3. Este archivo es tu base de datos local, donde todos los posts son guardados. No queremos agregarlo al repositorio ya que tu sitio web de PythonAnywhere usará una base de datos diferente. Esa base de datos puede ser SQLite, como en nuestra computadora de desarrollo, pero normalmente usarás algo llamado PostgreSQL o MySQL que permite manejar muchos más visitantes que SQLite. De cualquier forma, al ignorar tu base de datos SQLite para la copia de GitHub, significa que todos los post que crees van a estar disponibles de forma local únicamente, y tendrás que agregarlos manualmente de nuevo en producción. Debes pensar en tu base de datos local como un buen lugar donde jugar y probar diferentes cosas sin miedo de que puedas borrar los posts reales de tu blog.

Es una buena idea utilizar el comando git status antes de git add o en cualquier momento en que no estés seguro de lo que ha cambiado. Esto ayudará a evitar cualquier sorpresa, como agregar o hacer commit de archivos equivocados. El comando git status devuelve información sobre los archivos sin seguimiento ("untracked"), modificados, preparados ("staged"), el estado de la rama y mucho más. La salida debería ser similar a:

$ git status  
On branch master  
  
Initial commit  
  
Untracked files:  
 (use "git add <file>..." to include in what will be committed)  
  
 .gitignore  
 blog/  
 manage.py  
 mysite/  
  
nothing added to commit but untracked files present (use "git add" to track)

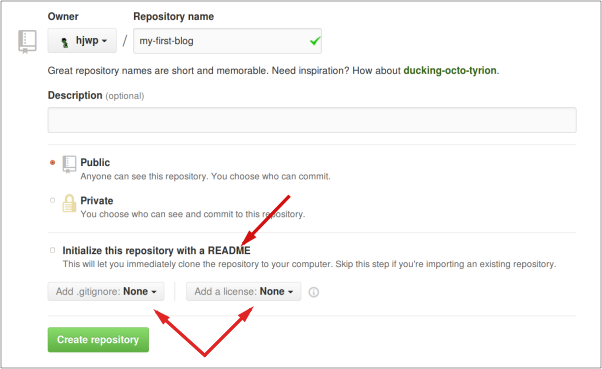
Y finalmente guardamos nuestros cambios. Ve a tu consola y ejecuta estos comandos:

$ git add --all .  
$ git commit -m "My Django Girls app, first commit"  
 [...]  
 13 files changed, 200 insertions(+)  
 create mode 100644 .gitignore  
 [...]  
 create mode 100644 mysite/wsgi.py

## **Enviar nuestro código a GitHub**

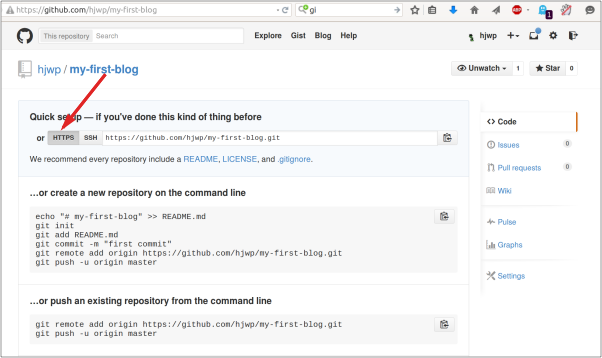
Ve a [GitHub.com](https://www.github.com/) y registra una nueva cuenta gratuita. (Si ya lo hiciste, ¡genial!)

Luego, crea un nuevo repositorio con el nombre "my-first-blog". Deja desmarcada la opción "Initialize with a README", deja la opción .gitignore en blanco (lo hemos hecho a mano) y deja la licencia como "None".



**Nota** El nombre my-first-blog es importante. Podrías elegir otra cosa, pero va a aparecer muchas veces en las instrucciones que siguen y tendrías que sustituirlo cada vez. Probablemente sea más sencillo quedarte con el nombre my-first-blog.

En la próxima pantalla verás la URL para clonar tu repositorio. Elige la versión "HTTPS", cópiala y en un momento la pegaremos en la consola:



Ahora tenemos que conectar el repositorio Git de tu computadora con el que está en GitHub.

Escribe lo siguiente en la consola (sustituye <tu-usuario-github> por el nombre de usuario que elegiste al crear tu cuenta de GitHub, pero sin los signos de mayor y menor):

$ git remote add origin https://github.com/<your-github-username>/my-first-blog.git  
$ git push -u origin master

Escribe tu nombre de usuario y contraseña de GitHub y deberías ver algo así:

Username for 'https://github.com': hjwp  
Password for 'https://hjwp@github.com':  
Counting objects: 6, done.  
Writing objects: 100% (6/6), 200 bytes | 0 bytes/s, done.  
Total 3 (delta 0), reused 0 (delta 0)  
To https://github.com/hjwp/my-first-blog.git  
 \* [new branch] master -> master  
Branch master set up to track remote branch master from origin.

Tu código está ahora en GitHub. ¡Ve y míralo! Verás que está en buena compañía; [Django](https://github.com/django/django), el [Tutorial de Django Girls](https://github.com/DjangoGirls/tutorial) y muchos otros grandes proyectos de código abierto también alojan su código en GitHub :)

# **Configurar nuestro blog en PythonAnywhere**

**Nota** Puede que ya hayas creado una cuenta en PythonAnywhere durante los paso de instalación. Si es así, no necesitas hacerlo de nuevo.

Es hora de registrar una cuenta gratuita de tipo "Beginner" en PythonAnywhere.

* [www.pythonanywhere.com](https://www.pythonanywhere.com/)/)

**Nota** Cuando elijas tu nombre de usuario ten en cuenta que la URL de tu blog tendrá la forma nombredeusuario.pythonanywhere.com, así que o bien elige tu propio apodo o bien un nombre que describa sobre qué trata tu blog.

## **Cargar nuestro código en PythonAnywhere**

Cuando te hayas registrado en PythonAnywhere serás redirigido a tu panel de control o página "Consoles". Elige la opción para iniciar una consola "Bash". Esta es la versión PythonAnywhere de una consola, como la que tienes en tu PC.



**Nota**: PythonAnywhere está basado en Linux, por lo que si estás en Windows la consola será un poco distinta a la que tienes en tu computadora.

Vamos a descagar nuestro código de GitHub a PythonAnywhere mediante la creación de un "clon" del repositorio. Escribe lo siguiente en la **consola de PythonAnywhere** (no te olvides de utilizar tu nombre de usuario de GitHub en lugar de <tu-usuario-github>):

$ git clone https://github.com/<your-github-username>/my-first-blog.git

Esto va a descargar una copia de tu código en PythonAnywhere. Compruébalo escribiendo tree my-first-blog:

$ tree my-first-blog  
my-first-blog/  
├── blog  
│ ├── \_\_init\_\_.py  
│ ├── admin.py  
│ ├── migrations  
│ │ ├── 0001\_initial.py  
│ │ └── \_\_init\_\_.py  
│ ├── models.py  
│ ├── tests.py  
│ └── views.py  
├── manage.py  
└── mysite  
 ├── \_\_init\_\_.py  
 ├── settings.py  
 ├── urls.py  
 └── wsgi.py

### **Crear un virtualenv (entorno virtual) en PythonAnywhere**

Tal y como hiciste en tu propia computadora, puedes crear un virtualenv en PythonAnywhere. En la consola Bash, escribe:

$ cd my-first-blog  
  
$ virtualenv --python=python3.6 myvenv  
Running virtualenv with interpreter /usr/bin/python3.6  
[...]  
Installing setuptools, pip...done.  
  
$ source myvenv/bin/activate  
  
(myvenv) $ pip install django~=1.11.0  
Collecting django  
[...]  
Successfully installed django-1.11.3

**Nota** El paso pip install puede llevar un par de minutos. ¡Paciencia, paciencia! Pero si tarda más de 5 minutos, algo va mal. Pregunta a tu tutor.

### **Crear la base de datos en PythonAnywhere**

Aquí hay otra cosa que es diferente entre tu computadora y el servidor: éste utiliza una base de datos diferente. Por lo tanto, las cuentas de usuario y las entradas pueden ser diferentes en el servidor y en tu computadora.

De la misma forma que hicimos en nuestra computadora, repetimos los pasos para inicializar la base de datos en el servidor, con migrate y createsuperuser:

(myvenv) $ python manage.py migrate  
Operations to perform:  
[...]  
 Applying sessions.0001\_initial... OK  
(myvenv) $ python manage.py createsuperuser

## **Publicar nuestro blog como una aplicación web**

Ahora nuestro código está en PythonAnywhere, el virtualenv está listo y la base de datos está inicializada. ¡Estamos listos para publicarla como una aplicación web!

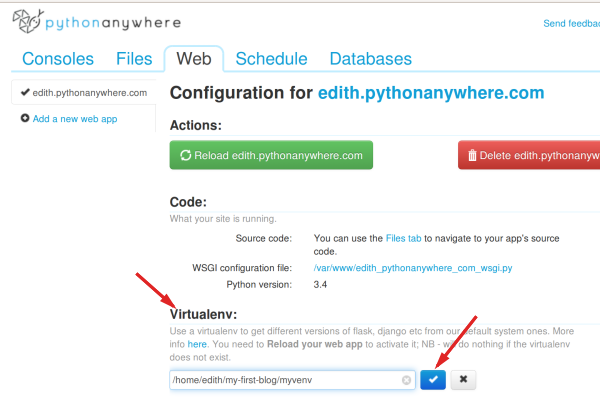
Haz clic en el logo de PythonAnywhere para volver al panel principal y haz clic en la pestaña **Web**. Por último, pincha en **Add a new web app**.

Después de confirmar tu nombre de dominio, elige **manual configuration** o "configuración manual" (N.B. – la opción "Django" *no*) en el diálogo. Luego elige **Python 3.6** y haz clic en "Next" para terminar con el asistente.

**Nota** Asegúrate de elegir la opción de "Manual configuration", no la de "Django". Somos demasiado buenos para la configuración por defecto de Django de PythonAnywhere ;-)

### **Configurar el virtualenv**

Serás redirigido a la pantalla de configuración de PythonAnywhere para tu aplicación web, a la que deberás acceder cada vez que quieras hacer cambios en la aplicación del servidor.



En la sección "Virtualenv", haz clic en el texto rojo que dice "Enter the path to a virtualenv" ("Introduce la ruta a un virtualenv") y escribe: /home/<tu-usuario>/my-first-blog/myvenv/. Haz clic en el cuadro azul seleccionado para guardar la ruta antes de continuar.

**Nota** Sustituye tu propio nombre de usuario como corresponda. Si cometes un error, PythonAnywhere te mostrará una pequeña advertencia.

### **Configurar el archivo WSGI**

Django funciona utilizando el "protocolo WSGI", un estándar para servir sitios web que usan Python, el cual es soportado por PythonAnywhere. La forma de configurar PythonAnywhere para que reconozca nuestro blog Django es editando un archivo de configuración WSGI.

Haz clic en el enlace "WSGI configuration file" (en la sección "Code" en la parte de arriba de la página; se llamará algo parecido a /var/www/<tu-usuario>\_pythonanywhere\_com\_wsgi.py) y te redirigirá al editor.

Borra todo el contenido y reemplázalo con algo como esto:

import os  
import sys  
  
path = os.path.expanduser('~/my-first-blog')  
if path not in sys.path:  
 sys.path.append(path)  
  
os.environ['DJANGO\_SETTINGS\_MODULE'] = 'mysite.settings'  
  
from django.core.wsgi import get\_wsgi\_application  
from django.contrib.staticfiles.handlers import StaticFilesHandler  
application = StaticFilesHandler(get\_wsgi\_application())

Este archivo se encarga de decirle a PythonAnywhere donde vive nuestra aplicación web y como se llama el archivo de configuración de Django.

El StaticFilesHandler es para manejar nuestro CSS. Esto lo hace automáticamente en nuestro desarrollo local el comando runserver. Encontraremos un poquito más sobre los archivos estáticos luego en este tutorial, cuando editemos los archivos CSS para nuestro sitio.

Haz click en **Save** y vuelve a la pestaña **Web**.

¡Todo listo! Dale al botón verde grande que dice **Reload** y podrás ver tu aplicación. Verás un enlace a ella en la parte de arriba de la página.

## **Consejos de depuración**

Si ves un error cuando intentas visitar tu página, el primer lugar donde buscar información de depuración es el **registro de errores**. Encontrarás un enlace a él en la [pestaña Web](https://www.pythonanywhere.com/web_app_setup/) de PythonAnywhere. Mira si hay algún mensaje de error ahí. Los más recientes están al final. Los problemas más comunes incluyen:

* Olvidar alguno de los pasos que hicimos en la consola: crear el virtualenv, activarlo, instalar Django en él, inicializar la base de datos.
* Cometer un error en la ruta del virtualenv en la pestaña Web; suele haber un mensajito de error de color rojo, si hay algún problema.
* Cometer un error en el fichero de configuración WSGI; ¿has puesto bien la ruta a la carpeta my-first-blog?
* ¿Has elegido la misma versión de Python para el virtualenv y para la aplicación web? Ambas deberían ser 3.6.

Hay algunos [consejos generales de depuración en el wiki de Pythonanywhere](https://www.pythonanywhere.com/wiki/DebuggingImportError)

Y recuerda, ¡tu tutor está ahí para ayudarte!

# **¡Estás en vivo!**

La página predeterminada para su sitio web debe decir "Bienvenido a Django", al igual que lo hace en su computadora. Intenta agregar /admin/ al final de la URL y te redirigirá al panel de administración. Ingresa con tu nombre de usuario y contraseña y verás que puedes agregar nuevas entradas en el servidor.

Una vez que hayas creado algunos posts en tu blog, puedes volver a tu instancia local (no la de PythonAnywhere). Desde ahí podrías trabajar en tu instancia local para hacer algunos cambios. Estos son los pasos comunes en el desarrollo web (hacer cambios locales, enviarlos a GitHub, descargarlos en el servidor web). Esto te permite trabajar y experimentar sin romper tu sitio web en vivo. Muy bueno, ¿no?

¡Date una *GRAN* palmada en la espalda! Los despliegues en el servidor son una de las partes más complejas del desarrollo web y muchas veces a la gente le cuesta varios días tenerlo funcionando. Pero tú tienes tu sitio en vivo, en Internet de verdad, ¡así como suena!