# **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