# E Solución de problemas de despliegue



Desplegar una aplicación es tremendamente satisfactorio cuando funciona, sobre todo si nunca lo has hecho antes. Sin embargo, hay muchos obstáculos que pueden surgir en el proceso de despliegue y, por desgracia, algunos de estos problemas pueden ser difíciles de identificar y solucionar. Este apéndice te ayudará a comprender los enfoques modernos de la implantación y te dará formas concretas de solucionar los problemas del proceso de implantación cuando las cosas no funcionen.

Si la información adicional de este apéndice no es suficiente para ayudarte a llevar a cabo con éxito el proceso de despliegue, consulta los recursos en línea en [https://ehmatthes.github.io/pcc\_3e;](https://ehmatthes.github.io/pcc_3e) las actualizaciones que hay allí te ayudarán casi con toda seguridad a llevar a cabo un despliegue con éxito.

## Comprender las implantaciones

Cuando trates de solucionar un intento de despliegue concreto, es útil tener una idea clara de cómo funciona un despliegue típico. *Deployment* se refiere al proceso de tomar un proyecto que funciona en tu sistema local, y copiar ese proyecto a un servidor remoto de forma que pueda responder a las peticiones de cualquier usuario de Internet. El entorno remoto difiere de un sistema local típico en varios aspectos importantes: probablemente no sea el mismo sistema operativo (SO) que el que estás utilizando, y lo más probable es que sea uno de los muchos servidores virtuales de un único servidor físico.

Cuando despliegas un proyecto, o *push* al servidor remoto, hay que seguir los siguientes pasos:

* Crear un servidor virtual en una máquina física en un centro de datos.
* Establecer una conexión entre el sistema local y el servidor remoto.
* Copia el código del proyecto en el servidor remoto.
* Identifica todas las dependencias del proyecto e instálalas en el servidor remoto.
* Configura una base de datos y ejecuta las migraciones existentes.
* Copia los archivos estáticos (CSS, archivos JavaScript y archivos multimedia) en un lugar donde puedan servirse con eficacia.
* Inicia un servidor para gestionar las solicitudes entrantes.
* Comienza a dirigir las solicitudes entrantes al proyecto, una vez que esté preparado para gestionarlas.

Si tienes en cuenta todo lo que conlleva un despliegue, no es de extrañar que los despliegues fracasen a menudo. Afortunadamente, una vez que comprendas lo que debería estar ocurriendo, tendrás más posibilidades de identificar lo que ha ido mal. Si puedes identificar lo que ha ido mal, tal vez puedas identificar una solución que haga que el siguiente intento de despliegue tenga éxito.

Puedes desarrollar localmente en un tipo de SO y enviar a un servidor que ejecute un SO diferente. Es importante que sepas a qué tipo de sistema estás enviando el software, porque eso puede ayudarte a solucionar algunos problemas. En el momento de escribir esto, un servidor remoto básico en Platform.sh ejecuta Debian Linux; la mayoría de los servidores remotos son sistemas basados en Linux.

## Solución de problemas básicos

Algunos pasos de la resolución de problemas son específicos de cada SO, pero llegaremos a eso dentro de un momento. En primer lugar, vamos a considerar los pasos que todo el mundo debería probar al solucionar los problemas de una implantación.

Tu mejor recurso es la salida generada durante el intento de inserción. Esta salida puede parecer intimidante; si eres nuevo en el despliegue de aplicaciones, puede parecer muy técnica, y suele haber mucha. La buena noticia es que no necesitas entenderlo todo. Deberías tener dos objetivos al hojear la salida del registro: identificar los pasos de despliegue que funcionaron e identificar los pasos que no funcionaron. Si lo consigues, en podrás averiguar qué cambiar en tu proyecto, o en tu proceso de despliegue, para que tu próximo empuje sea un éxito.

### Sigue las sugerencias en pantalla

A veces, la plataforma a la que estás empujando generará un mensaje con una sugerencia clara sobre cómo solucionar el problema. Por ejemplo, éste es el mensaje que verás si creas un proyecto Platform.sh antes de inicializar un repositorio Git, y luego intentas empujar el proyecto:

$ platform push  
❶ Enter a number to choose a project:  
 [0] ll\_project (votohz445ljyg)  
 > 0  
  
❷ [RootNotFoundException]  
 Project root not found. This can only be run from inside a project directory.  
  
❸ To set the project for this Git repository, run:  
 platform project:set-remote [id]

Estamos intentando empujar un proyecto, pero el proyecto local aún no se ha asociado a un proyecto remoto. Entonces, la CLI de Platform.sh nos pregunta a qué proyecto remoto queremos empujar ❶. Introducimos 0, para seleccionar el único proyecto de la lista. Pero a continuación, vemos un RootNotFoundException ❷. Esto ocurre porque Platform.sh busca un directorio *.git* cuando inspecciona el proyecto local, para averiguar cómo conectar el proyecto local con el remoto. En este caso, como no había ningún directorio *.git* cuando se creó el proyecto remoto, nunca se estableció esa conexión. La CLI sugiere una solución ❸; nos dice que podemos especificar el proyecto remoto que debe asociarse a este proyecto local, utilizando el comando project:set-remote.

Probemos esta sugerencia:

$ platform project:set-remote votohz445ljyg  
Setting the remote project for this repository to: ll\_project (votohz445ljyg)  
  
The remote project for this repository is  
 now set to: ll\_project (votohz445ljyg)

En la salida anterior, la CLI mostraba el ID de este proyecto remoto, votohz4451jyg. Así que ejecutamos el comando sugerido, utilizando este ID, y la CLI es capaz de establecer la conexión entre el proyecto local y el proyecto remoto.

Ahora vamos a intentar empujar el proyecto de nuevo:

$ platform push  
Are you sure you want to push to the main (production) branch? [Y/n] y  
Pushing HEAD to the existing environment main  
--snip--

Esto ha sido un push con éxito; seguir la sugerencia en pantalla ha funcionado.

Debes tener cuidado al ejecutar comandos que no entiendas completamente. Sin embargo, si tienes buenas razones para creer que un comando puede hacer poco daño, y si confías en la fuente de la recomendación, puede ser razonable probar las sugerencias que ofrecen las herramientas que estás utilizando.

## Nota

Ten en cuenta que hay personas que te dirán que ejecutes comandos que borrarán tu sistema o lo expondrán a la explotación remota. Seguir las sugerencias de una herramienta proporcionada por una empresa u organización en la que confías es diferente de seguir las sugerencias de personas al azar en Internet. Siempre que trates con conexiones remotas, procede con mucha precaución.

### Lee el registro de salida

Como ya se ha dicho, la salida del registro que ves cuando ejecutas un comando como platform push puede ser a la vez informativa e intimidatoria. Lee el siguiente fragmento de la salida del registro, tomado de un intento diferente de utilizar platform push, y comprueba si puedes detectar el problema:

--snip--  
Collecting soupsieve==2.3.2.post1  
 Using cached soupsieve-2.3.2.post1-py3-none-any.whl (37 kB)  
Collecting sqlparse==0.4.2  
 Using cached sqlparse-0.4.2-py3-none-any.whl (42 kB)  
Installing collected packages: platformshconfig, sqlparse,...  
Successfully installed Django-4.1 asgiref-3.5.2 beautifulsoup4-4.11.1...  
W: ERROR: Could not find a version that satisfies the requirement gunicorrn  
W: ERROR: No matching distribution found for gunicorrn  
  
130 static files copied to '/app/static'.  
  
Executing pre-flight checks...  
--snip--

Cuando falla un intento de despliegue, una buena estrategia consiste en examinar la salida del registro y ver si puedes detectar algo que parezca una advertencia o un error. Las advertencias son bastante comunes; a menudo son mensajes sobre los próximos cambios en las dependencias de un proyecto, para ayudar a los desarrolladores a abordar los problemas antes de que causen fallos reales.

Un push correcto puede tener advertencias, pero no debería tener errores. En este caso, Platform.sh no pudo encontrar la forma de instalar el requisito gunicorrn. Se trata de un error tipográfico en el archivo *requirements\_remote.txt*, que debía incluir gunicorn (con un *r*). No siempre es fácil detectar el problema de raíz en la salida del registro, especialmente cuando el problema causa un montón de errores y advertencias en cascada. Al igual que cuando lees un rastreo en tu sistema local, es una buena idea fijarse bien en los primeros errores que aparecen, y también en los últimos. La mayoría de los errores intermedios suelen ser paquetes internos que se quejan de que algo ha ido mal, y pasan mensajes sobre el error a otros paquetes internos. El error real que podemos solucionar suele ser uno de los primeros o últimos errores de la lista.

A veces, serás capaz de detectar el error, y otras veces, no tendrás ni idea de lo que significa la salida. Sin duda merece la pena intentarlo, y utilizar la salida del registro para diagnosticar con éxito un error es una sensación tremendamente satisfactoria. A medida que pases más tiempo examinando los resultados del registro, mejorarás en la identificación de la información más significativa para ti.

## Solución de problemas específicos del sistema operativo

Puedes desarrollar en el sistema operativo que quieras y enviar a cualquier host que desees. Las herramientas para enviar proyectos se han desarrollado lo suficiente como para modificar tu proyecto según sea necesario para que se ejecute correctamente en el sistema remoto. Sin embargo, pueden surgir algunos problemas específicos del sistema operativo.

En el proceso de despliegue de Platform.sh, una de las fuentes más probables de dificultades es la instalación de la CLI. Aquí tienes el comando para hacerlo:

$ curl -fsS https://platform.sh/cli/installer | php

El comando comienza con curl, una herramienta que te permite solicitar recursos remotos, a los que se accede a través de una URL, dentro de un terminal. En este caso, se utiliza para descargar el instalador de la CLI desde un servidor Platform.sh. La sección -fsS del comando es un conjunto de banderas que modifican cómo se ejecuta curl. La bandera f indica a curl que suprima la mayoría de los mensajes de error, para que el instalador de la CLI pueda gestionarlos en lugar de informarte de todos ellos. La bandera s indica a curl que se ejecute en silencio; deja que el instalador de la CLI decida qué información mostrar en el terminal. La bandera S indica a curl que muestre un mensaje de error si el comando general falla. El | php al final del comando indica a tu sistema que ejecute el archivo de instalación descargado utilizando un intérprete de PHP, porque la CLI Platform.sh está escrita en PHP.

Esto significa que tu sistema necesita curl y PHP para instalar la CLI Platform.sh. Para utilizar la CLI, también necesitarás Git y un terminal que pueda ejecutar comandos Bash. *Bash* es un lenguaje que está disponible en la mayoría de los entornos de servidor. La mayoría de los sistemas modernos tienen espacio suficiente para instalar varias herramientas como ésta.

Las siguientes secciones te ayudarán a abordar estos requisitos para tu sistema operativo. Si aún no tienes instalado Git, consulta las instrucciones para instalar Git en la página 484 del Apéndice D y luego ve a la sección de aquí que sea aplicable a tu SO.

## Nota

Una herramienta excelente para entender comandos de terminal como el que se muestra aquí es [https://explainshell.com.](https://explainshell.com) Introduce el comando que intentas comprender y el sitio te mostrará la documentación de todas las partes del comando. Pruébalo con el comando utilizado para instalar la CLI Platform.sh.

### Despliegue desde Windows

Windows ha experimentado un resurgimiento de su popularidad entre los programadores en los últimos años. Windows ha integrado muchos elementos diferentes de otros sistemas operativos, proporcionando a los usuarios una serie de opciones sobre cómo realizar el trabajo de desarrollo local e interactuar con sistemas remotos.

Una de las dificultades más significativas de la implantación desde Windows es que el sistema operativo base de Windows no es el mismo que utiliza un servidor remoto basado en Linux. Un sistema Windows base tiene un conjunto de herramientas y lenguajes distinto al de un sistema Linux base, por lo que, para llevar a cabo el trabajo de despliegue desde Windows, tendrás que elegir cómo integrar los conjuntos de herramientas basados en Linux en tu entorno local.

#### Subsistema Windows para Linux

Un enfoque popular es utilizar *Windows Subsystem for Linux* (*WSL*), un entorno que permite ejecutar Linux directamente en Windows. Si tienes configurado WSL, utilizar la CLI Platform.sh en Windows será tan fácil como utilizarla en Linux. La CLI no sabrá que se está ejecutando en Windows; sólo verá el entorno Linux en el que la estás utilizando.

Configurar WSL es un proceso de dos pasos: primero instalas WSL y luego eliges una distribución de Linux para instalarla en el entorno WSL. Configurar un entorno WSL es más de lo que se puede describir aquí; si te interesa este enfoque y aún no lo tienes configurado, consulta la documentación en [https://docs.microsoft.com/en-us/windows/wsl/about.](https://docs.microsoft.com/en-us/windows/wsl/about) Una vez que tengas configurado WSL, puedes seguir las instrucciones de la sección Linux de este apéndice para continuar tu trabajo de despliegue.

#### Git Bash

Otra forma de crear un entorno local desde el que puedas desplegar utiliza *Git Bash*, un entorno de terminal compatible con Bash pero que se ejecuta en Windows. Git Bash se instala junto con Git cuando utilizas el instalador de [https://git-scm.com.](https://git-scm.com) Este enfoque puede funcionar, pero no es tan ágil como WSL. En este enfoque, tendrás que utilizar un terminal Windows para algunos pasos y un terminal Git Bash para otros.

Primero tendrás que instalar PHP. Puedes hacerlo con *XAMPP*, un paquete que incluye PHP con algunas otras herramientas para desarrolladores. Ve a <https://apachefriends.org> y haz clic en el botón para descargar XAMPP para Windows. Abre el instalador y ejecútalo; si ves una advertencia sobre las restricciones del Control de Cuentas de Usuario (UAC), haz clic en **Aceptar**. Acepta todos los valores predeterminados del instalador.

Cuando termine de ejecutarse el instalador, tendrás que añadir PHP a la ruta de tu sistema; esto indicará a Windows dónde buscar cuando quieras ejecutar PHP. En el menú Inicio, escribe path y haz clic en **Editar las variables de entorno del** sistema; haz clic en el botón **Variables de entorno**. Deberías ver la variable Path resaltada; haz clic en **Editar** en este panel. Haz clic en **Nueva** para añadir una nueva ruta a la lista actual de rutas. Suponiendo que mantuviste la configuración predeterminada al ejecutar el instalador de XAMPP, añade C:\xampp\php en el cuadro que aparece y haz clic en **Aceptar**. Cuando hayas terminado, cierra todos los cuadros de diálogo del sistema que sigan abiertos.

Una vez cumplidos estos requisitos, puedes instalar la CLI Platform.sh. Necesitarás utilizar un terminal de Windows con privilegios de administrador; introduce command en el menú Inicio, y en la aplicación Símbolo del sistema, haz clic en **Ejecutar como administrador**. En el terminal que aparece, introduce el siguiente comando:

> curl -fsS https://platform.sh/cli/installer | php

Esto instalará la CLI Platform.sh, como se ha descrito anteriormente.

Por último, trabajarás en Git Bash. Para abrir un terminal Git Bash, ve al menú Inicio y busca git bash. Haz clic en la **aplicación G** it Bash que aparece; deberías ver que se abre una ventana de terminal. Puedes utilizar comandos tradicionales basados en Linux como ls en este terminal, así como comandos basados en Windows como dir. Para asegurarte de que la instalación se ha realizado correctamente, ejecuta el comando platform list. Deberías ver una lista de todos los comandos de la CLI Platform.sh. A partir de este momento, realiza todo tu trabajo de despliegue utilizando la CLI Platform.sh dentro de una ventana de terminal Git Bash.

### Despliegue desde macOS

El sistema operativo macOS no está basado en Linux, pero ambos se desarrollaron sobre principios similares. Lo que esto significa, en la práctica, es que muchos de los comandos y flujos de trabajo que utilizas en macOS funcionarán también en un entorno de servidor remoto. Puede que necesites instalar algunos recursos orientados al desarrollador para disponer de todas estas herramientas en tu entorno local de macOS. Si en algún momento de tu trabajo se te pide que instales la *command line developer tools*, haz clic en **Instalar** para aprobar la instalación.

La dificultad más probable al instalar la CLI Platform.sh es asegurarse de que PHP está instalado. Si aparece un mensaje que indica que no se encuentra el comando php, tendrás que instalar PHP. Una de las formas más sencillas de instalar PHP es utilizando el gestor de paquetes *Homebrew*, que facilita la instalación de una amplia variedad de paquetes de los que dependen los programadores. Si aún no tienes Homebrew instalado, visita <https://brew.sh> y sigue las instrucciones para instalarlo.

Una vez instalado Homebrew, utiliza el siguiente comando para instalar PHP:

$ brew install php

Esto tardará un rato en ejecutarse, pero una vez que se haya completado, deberías poder instalar correctamente la CLI Platform.sh.

### Despliegue desde Linux

Dado que la mayoría de los entornos de servidor están basados en Linux, no deberías tener muchas dificultades para instalar y utilizar la CLI Platform.sh. Si intentas instalar la CLI en un sistema con una instalación reciente de Ubuntu, te dirá exactamente qué paquetes necesitas:

$ curl -fsS https://platform.sh/cli/installer | php  
Command 'curl' not found, but can be installed with:  
sudo apt install curl  
Command 'php' not found, but can be installed with:  
sudo apt install php-cli

La salida real tendrá más información sobre algunos otros paquetes que podrían funcionar, además de alguna información sobre la versión. El siguiente comando instalará curl y PHP:

$ sudo apt install curl php-cli

Tras ejecutar este comando, el comando de instalación CLI Platform.sh debería ejecutarse correctamente. Dado que tu entorno local es bastante similar a la mayoría de los entornos de alojamiento basados en Linux, gran parte de lo que aprendas sobre cómo trabajar en tu terminal se trasladará también al trabajo en un entorno remoto.

## Otros métodos de instalación

Si Platform.sh no te funciona, o si quieres probar un enfoque diferente, hay muchas plataformas de alojamiento entre las que elegir. Algunas funcionan de forma similar al proceso descrito en el Capítulo 20, y otras tienen un enfoque muy diferente para llevar a cabo los pasos descritos al principio de este apéndice:

* Platform.sh te permite utilizar un navegador para llevar a cabo los pasos para los que utilizamos la CLI. Si te gustan más las interfaces basadas en navegador que los flujos de trabajo basados en terminal, puede que prefieras este enfoque.
* Hay otros proveedores de alojamiento que ofrecen enfoques basados tanto en la CLI como en el navegador. Algunos de estos proveedores ofrecen terminales dentro de su navegador, por lo que no tienes que instalar nada en tu sistema.
* Algunos proveedores te permiten enviar tu proyecto a un sitio de alojamiento de código remoto como GitHub, y luego conectar tu repositorio de GitHub al sitio de alojamiento. El servidor extrae entonces tu código de GitHub, en lugar de pedirte que envíes tu código desde tu sistema local directamente al servidor. Platform.sh también admite este tipo de flujo de trabajo.
* Algunos proveedores ofrecen una serie de servicios entre los que puedes elegir para crear una infraestructura que se adapte a tu proyecto. Esto normalmente requiere que tengas un conocimiento más profundo del proceso de despliegue, y de lo que necesita un servidor remoto para servir a un proyecto. Estos hosts incluyen Amazon Web Services (AWS) y la plataforma Azure de Microsoft. Puede ser mucho más difícil hacer un seguimiento de tus costes en este tipo de plataformas, porque cada servicio puede acumular cargos de forma independiente.
* Mucha gente aloja sus proyectos en un servidor virtual privado (VPS). En este enfoque, alquilas un servidor virtual que actúa como un ordenador remoto, te conectas al servidor, instalas el software necesario para ejecutar tu proyecto, copias tu código, estableces las conexiones adecuadas y permites que tu servidor empiece a aceptar solicitudes.

Regularmente aparecen nuevas plataformas y enfoques de alojamiento; encuentra una que te parezca atractiva, e invierte tiempo en aprender el proceso de despliegue de ese proveedor. Mantén tu proyecto el tiempo suficiente para que llegues a saber qué funciona bien con el enfoque de tu proveedor y qué no. Ninguna plataforma de alojamiento va a ser perfecta; tendrás que juzgar continuamente si el proveedor que utilizas actualmente es lo suficientemente bueno para tu caso de uso.

Voy a ofrecer una última advertencia sobre la elección de una plataforma de despliegue y un enfoque general del despliegue. Algunas personas te guiarán con entusiasmo hacia enfoques y servicios de despliegue excesivamente complejos, destinados a hacer que tu proyecto sea altamente fiable y capaz de servir a millones de usuarios simultáneamente. Muchos programadores invierten mucho tiempo, dinero y energía en crear una compleja estrategia de despliegue, sólo para descubrir que casi nadie utiliza su proyecto. La mayoría de los proyectos Django pueden configurarse en un pequeño plan de alojamiento y ajustarse para servir miles de peticiones por minuto. Si tu proyecto recibe algo menos que este nivel de tráfico, dedica tiempo a configurar tu despliegue para que funcione bien en una plataforma mínima antes de invertir en una infraestructura pensada para algunos de los sitios más grandes del mundo.

El despliegue es increíblemente desafiante a veces, pero igual de satisfactorio cuando tu proyecto en vivo funciona bien. Disfruta del reto, y pide ayuda cuando la necesites.