

# Integración continua utilizando Jenkins y una aplicación de Django alojada en un repositorio de Github.

Jenkins es una aplicación que permite hacer Integración Continua de tu aplicación. La forma en la que funciona es la siguiente: se especifica qué acciones o eventos son los que accionarían el despliegue de la aplicación en el servidor (como un push a una rama), y se crea un script que se ejecutará en el servidor antes de desplegar la aplicación. Generalmente este script contiene los comandos para probar la aplicación y para asegurarse que esta funciona correctamente en el servidor (como el *python manage.py test* para Django)

Requisitos: Servidor dedicado, aplicación en Django corriendo en servidor.

Una vez que tenemos nuestra aplicación alojada en el repositorio de Github, procederemos a instalar Jenkins en el servidor.

## Instalar Jenkins en servidor

Para instalar Jenkins en el servidor, entramos a la [página oficial del distribuidor](https://pkg.jenkins.io/debian-stable/jenkins.io.key) y buscamos las instrucciones específicas para el sistema operativo de el servidor. En este caso, instalaremos Jenkins en un servidor Ubuntu.

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key  
add -
```

```
deb https://pkg.jenkins.io/debian-stable binary/
```

```
sudo apt-get update
```

```
sudo apt-get install jenkins
```

Una vez instalado Jenkins, verificamos que el servicio está corriendo con el siguiente comando:

```
sudo service jenkins status
```

Si está corriendo con normalidad, veremos un mensaje como el siguiente:

```
Jenkins Continuous Integration Server is running with the pid XXX
```

Iniciamos el servicio con el siguiente comando:

```
sudo service jenkins start
```

Ahora Jenkins está funcionando y podemos acceder a él en el puerto 8080 del servidor: <host>:8080 (<host> es la IP del servidor)

# Conectar Jenkins y Github

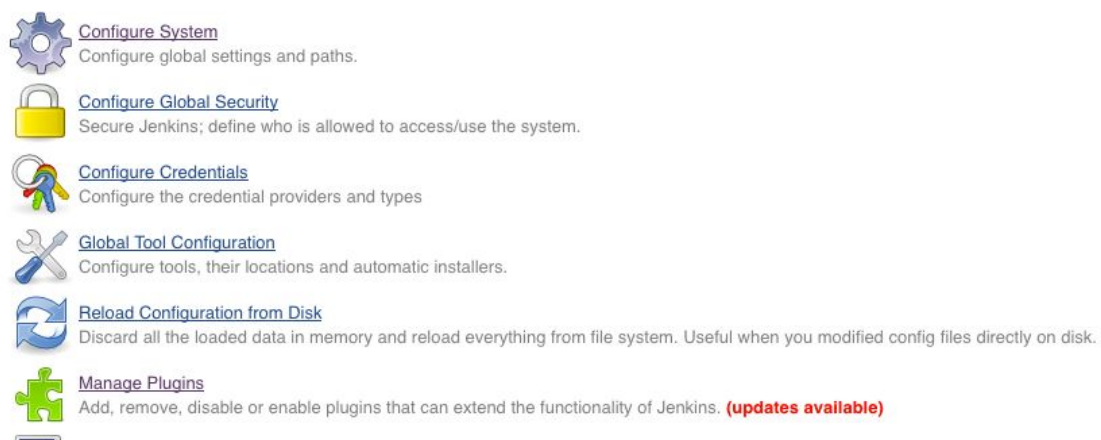
## Instalar dependencia en Jenkins

Para instalar la dependencia de Github en Jenkins, accedemos entrando a `<host>:8080` en el navegador y seleccionamos, del menú de la izquierda, la opción de **Manage Jenkins**



Después, entramos a **Manage Plugins** de la lista de opciones que nos aparece

### Manage Jenkins



Después, instalaremos el Plugin de Github: *Github plugin*. Verificamos que no se encuentre ya en los instalados

Filter:

Updates

Available

Installed

Advanced

Enabled	Name ↓	Version	Previously installed version	Uninstall
<input checked="" type="checkbox"/>	<a href="#">bouncycastle API Plugin</a> This plugin provides an stable API to Bouncy Castle related tasks.	<a href="#">2.16.2</a>		<input type="button" value="Uninstall"/>
<input checked="" type="checkbox"/>	<a href="#">Command Agent Launcher Plugin</a> Allows agents to be launched using a specified command.	<a href="#">1.2</a>		<input type="button" value="Uninstall"/>
<input checked="" type="checkbox"/>	<a href="#">Credentials Plugin</a> This plugin allows you to store credentials in Jenkins.	<a href="#">2.1.16</a>		<input type="button" value="Uninstall"/>
<input checked="" type="checkbox"/>	<a href="#">Display URL API</a> Provides the DisplayURLProvider extension point to provide alternate URLs for use in notifications	<a href="#">2.2.0</a>		<input type="button" value="Uninstall"/>
<input checked="" type="checkbox"/>	<a href="#">Git plugin</a> This plugin integrates <a href="#">Git</a> with Jenkins.	<a href="#">3.8.0</a>		<input type="button" value="Uninstall"/>
<input checked="" type="checkbox"/>	<a href="#">GitHub API Plugin</a> This plugin provides <a href="#">GitHub API</a> for other plugins.	<a href="#">1.90</a>		<input type="button" value="Uninstall"/>
<input checked="" type="checkbox"/>	<a href="#">GitHub Branch Source</a> Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc.	<a href="#">2.3.2</a>		<input type="button" value="Uninstall"/>
<input checked="" type="checkbox"/>	<a href="#">GitHub plugin</a> This plugin integrates <a href="#">GitHub</a> to Jenkins.	<a href="#">1.29.0</a>		<input type="button" value="Uninstall"/>

Si no está instalado, lo buscamos en la pestaña “Available” y lo instalamos.

## Crear trabajo

En este paso creamos el trabajo que hará la integración continua de nuestra aplicación.

En la página inicial de Jenkins, es decir, en <localhost>:8080 hacemos click en *New Item* del menú de la izquierda.




Ingresamos un nombre para el trabajo, seleccionamos “Freestyle Project” y hacemos click en el botón OK de hasta abajo.


**Enter an item name**

panayuda

» Required field

---

 **Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Pipeline**


Esto nos lleva a una pantalla donde detallaremos el trabajo, lo primero que haremos es seleccionar la casilla 'Discard Older Builds'

**General** Source Code Management Build Triggers Build Environment Build Post-build Actions

Project name panayuda

Description

[Plain text] [Preview](#)

☒ Discard old builds 

Strategy Log Rotation

Days to keep builds

if not empty, build records are only kept up to this number of days

Max # of builds to keep

if not empty, only up to this number of build records are kept

[Advanced...](#)

Debajo, seleccionamos Git para acceder al repositorio, el URL del repositorio y la rama de la cual se hará el build

### Source Code Management

☐ None  
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

Repository browser

Additional Behaviours

Después, seleccionamos los Triggers que harán el build. En este caso, el trigger será un push a la rama de master.

### Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☐ Build periodically

☒ GitHub hook trigger for GITScm polling

☐ Poll SCM

Por último, creamos el script que se ejecutará antes de hacer el build

### Build

Add build step ▾

- Invoke Ant
- Execute shell**
- Invoke top-level Maven targets
- Execute Windows batch command
- Invoke Gradle script
- Run with timeout
- Set build status to "pending" on GitHub commit

En el script, especificamos los comandos que haríamos manualmente para desplegar nuestra aplicación, como cambiarse de directorio, activar virtual environment, instalar requerimientos, correr pruebas, etcétera. Por default, Jenkins hace el deploy en la carpeta de Jenkins (/var/lib/jenkins/workspace), si quisiéramos que Jenkins hiciera el deploy en otra carpeta (la que tiene nuestro proyecto), tendremos que darle permisos de propietario al usuario jenkins a dicha carpeta, y en el script movernos a esa carpeta como primer comando.

Si el script llegara a tener un fallo, por ejemplo, en los tests, el despliegue no se completará.

Comando para darle permisos de propietario a jenkins a cierta carpeta:

```
sudo chown jenkins <path/carpeta/proyecto>
```

Script de ejemplo

```
#!/bin/bash
```

```
export WORKSPACE=`pwd`
```

```
# Create/Activate virtualenv
```

```
virtualenv venv
```

```
source venv/bin/activate
```

```
pip install django
```

```
pip install psycpg2
```

```
pip install psycpg2-binary
```

```
cd panqayuda
```

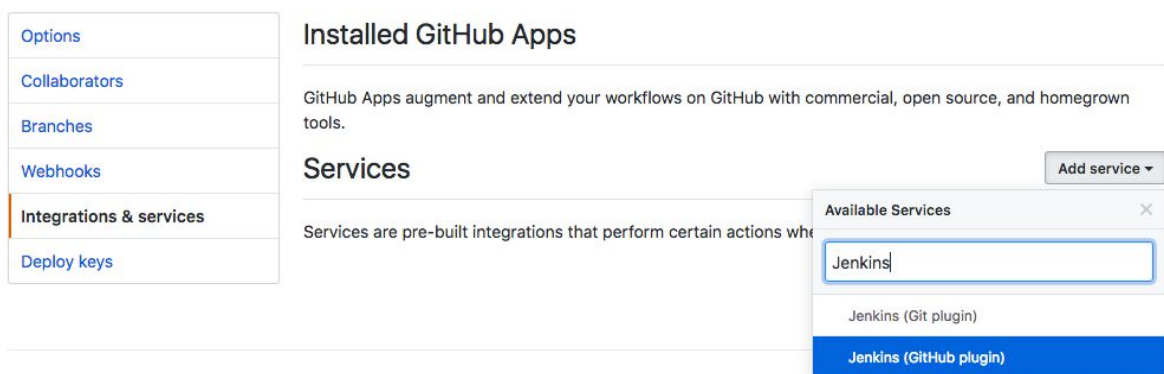
```
# Run tests
```

```
python manage.py test
```

Después, hacemos click en save y el trabajo queda guardado.

Lo siguiente que hay que hacer es configurar el repositorio para que accione Jenkins

En los settings de nuestro repositorio, entramos a Integration & services, y seleccionamos *Add service* y buscamos Jenkins Github plugin




Insertamos el URL que nos pide, el cual es el siguiente:


*http://<host>:8080/github-webhook/*


*Donde <host> es la dirección IP del servidor*

Y hacemos click en Add Service

Por último, hacemos un push de prueba y verificamos el resultado del build, viendo los detalles de nuestro trabajo en la página principal de Jenkins. En build history, podemos ver los builds y el estado de cada uno (fallido, exitoso). Si el build es exitoso, la nueva versión de la aplicación estará corriendo en el servidor.

 [Back to Dashboard](#)

 [Status](#)


 [Changes](#)

 [Workspace](#)

 [Build Now](#)

 [Delete Project](#)

 [Configure](#)

 [GitHub Hook Log](#)

 [GitHub](#)


## Project panqayuda

 [Workspace](#)

 [Recent Changes](#)

### Permalinks

- [Last build \(#53\), 9 hr 43 min ago](#)
- [Last stable build \(#53\), 9 hr 43 min ago](#)
- [Last successful build \(#53\), 9 hr 43 min ago](#)
- [Last failed build \(#33\), 2 days 3 hr ago](#)
- [Last unsuccessful build \(#33\), 2 days 3 hr ago](#)
- [Last completed build \(#53\), 9 hr 43 min ago](#)




## Build History


trend

find


x

 **#53**


Mar 14, 2018 7:38 PM

 **#52**

Mar 14, 2018 7:33 PM

 **#51**

Mar 14, 2018 7:25 PM

 **#50**

Mar 13, 2018 8:57 PM