

**Despliegue Automático de Aplicación Web con Jenkins y Netlify**

TRABAJo Fin de grado - tfg

Miguel Ángel Barea Ortega | ASIX

Índice

[Introducción de la idea del proyecto 2](#_Toc196301463)

[Palabras clave del proyecto 2](#_Toc196301464)

[Objeto / finalidad 3](#_Toc196301465)

[Metodología 3](#_Toc196301466)

[Contenido (paso a paso) 4](#_Toc196301467)

[Conclusiones 20](#_Toc196301468)

[Glosario 20](#_Toc196301469)

[Referencias bibliográficas utilizadas 20](#_Toc196301470)

[Anexos 20](#_Toc196301471)

# Introducción de la idea del proyecto

Este proyecto tiene como objetivo implementar un sistema de despliegue automático de una aplicación web sencilla usando herramientas modernas de integración continua (CI) y despliegue continuo (CD). Para ello, utilizaré Jenkins como servidor de automatización y Netlify como plataformas de despliegue en la nube.

# Palabras clave del proyecto

**CI/CD (Integración Continua / Entrega o Despliegue Continuo)**  
Es un conjunto de prácticas de desarrollo que permiten automatizar y mejorar el proceso de integración de código (CI) y su entrega o despliegue (CD).

* **CI (Continuous Integration)**: Integrar los cambios de código frecuentemente, asegurando que funcionen correctamente mediante pruebas automáticas.
* **CD (Continuous Delivery o Deployment)**: Automatizar la entrega o el despliegue del software a entornos de pruebas o producción.

Imagen que contiene Logotipo

El contenido generado por IA puede ser incorrecto.**Jenkins**  
Herramienta de automatización de código abierto utilizada para implementar procesos de CI/CD. Permite ejecutar tareas como compilaciones, pruebas y despliegues automáticamente cada vez que se actualiza el código.

**GitHub**  
Plataforma de alojamiento de código que utiliza Git como sistema de control de versiones. Permite colaborar, almacenar y gestionar versiones del código fuente, además de integrarse fácilmente con herramientas de CI/CD.

Netlify - Wikipedia**Netlify**  
Plataforma para desplegar sitios web y aplicaciones web estáticas o JAMstack. Automatiza procesos como el build, testeo y despliegue, integrándose con repositorios como GitHub.

**Automatización**  
Proceso de ejecutar tareas sin intervención manual, utilizando scripts o herramientas como Jenkins, Docker o GitHub Actions. Permite ahorrar tiempo y reducir errores humanos.

**Despliegue web**  
Proceso de mover una aplicación o sitio web desde un entorno de desarrollo local hasta un servidor o servicio donde los usuarios puedan acceder a ella a través de Internet.

**Docker**  
Plataforma que permite crear, ejecutar y administrar contenedores. Un contenedor es una unidad de software que incluye todo lo necesario para ejecutar una aplicación, asegurando que funcione igual en cualquier entorno.

**Pipeline**  
Secuencia automatizada de pasos que sigue una aplicación desde que se escribe el código hasta que se despliega. En un pipeline CI/CD, estos pasos pueden incluir compilación, pruebas, empaquetado y despliegue.

Icono

El contenido generado por IA puede ser incorrecto.**Cloudflare (Cloudflare Tunnel)**Cloudflare Tunnel permite establecer una conexión segura entre un servicio local (como un servidor Jenkins en desarrollo) y una URL accesible públicamente en Internet, sin necesidad de abrir puertos en el router ni modificar configuraciones de red. Esta solución expone el servicio local a través de un subdominio personalizado, protegido por la red de Cloudflare, lo que facilita la recepción de peticiones externas (como webhooks) en entornos de desarrollo o pruebas, manteniendo un alto nivel de seguridad y disponibilidad.

# Objeto / finalidad

* Automatizar el proceso de despliegue de una aplicación web.
* Configurar Jenkins para detectar cambios en un repositorio GitHub.
* Desplegar automáticamente la aplicación en Netlify.
* Aprender el flujo completo de CI/CD en un entorno real.

# Metodología

* Hipótesis: Es posible automatizar todo el ciclo de despliegue de una aplicación estática utilizando herramientas gratuitas y modernas como Jenkins y Netlify o Vercel.
* Herramientas utilizadas: GitHub, Jenkins, Docker, Netlify.
* Estrategia: Desarrollo por etapas siguiendo un enfoque práctico, validando cada paso con pruebas funcionales.

# Contenido (paso a paso)

**PASO 1: Crear la aplicación web**

Crear una carpeta e incluir un archivo index.html con contenido básico HTML. Aquí se puede ver la estructura de directorios y Web estática que he hecho y usado para incluir en el proyecto:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

**PASO 2: Subir el proyecto a GitHub**

Crear un repositorio en GitHub y clonar en Visual Studio Code:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

**PASO 3: Crear cuenta en Netlify**

Registrar una cuenta, importar el proyecto desde GitHub y desplegarlo automáticamente. Guardar la URL pública del sitio.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**PASO 4: Instalar y configurar Jenkins con Docker**

1. Instalar Docker si no está instalado.

Captura de pantalla de computadora

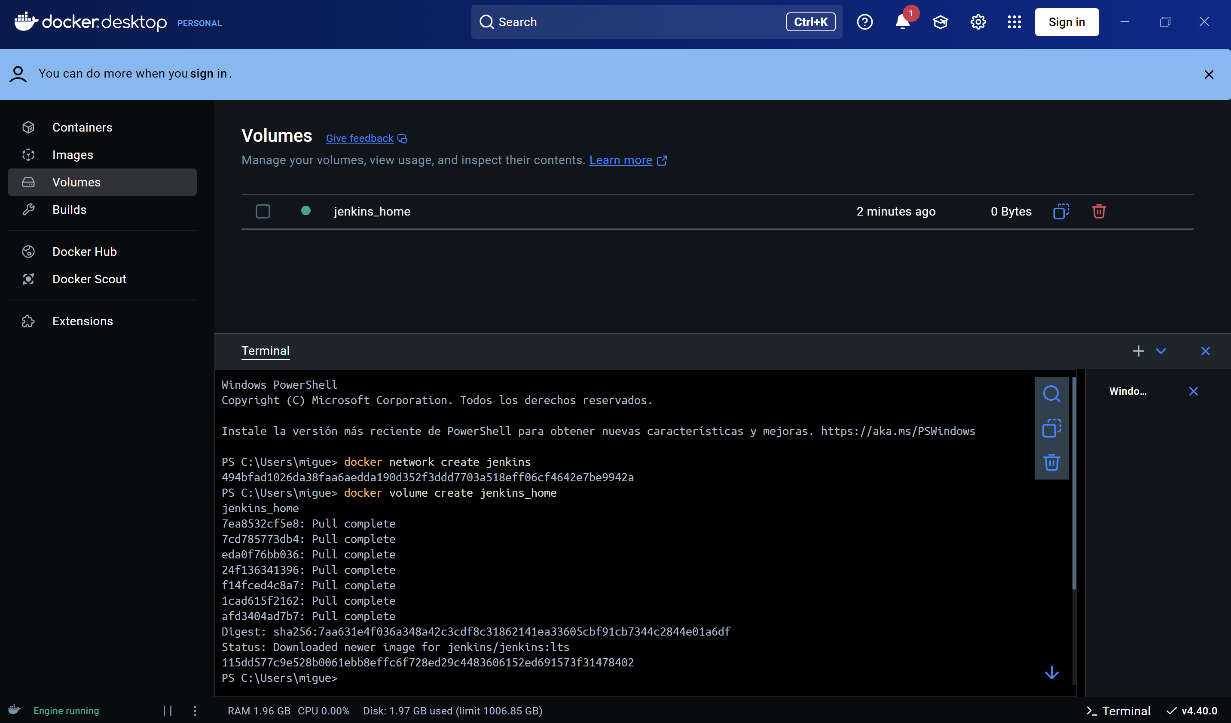
El contenido generado por IA puede ser incorrecto.

1. Crear una red Docker:

docker network create jenkins

1. Crear un volumen para Jenkins:

docker volume create jenkins\_home



1. Ejecutar Jenkins:

docker run -d --name jenkins -p 8080:8080 -p 50000:50000 --network jenkins -v jenkins\_home:/var/jenkins\_home jenkins/jenkins:lts

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

1. Acceder a Jenkins desde el navegador en ***http://localhost:8080*** y seguir las instrucciones para completar la configuración inicial.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

**PASO 5: Instalar plugins necesarios**

Instalar los siguientes plugins desde "Manage Plugins":

* GitHub Integration
* Pipeline

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**PASO 6: Configurar credenciales de GitHub y Netlify**

Este paso permite que Jenkins pueda autenticar y conectarse con los servicios externos de GitHub y Netlify para realizar acciones como clonar repositorios o desplegar sitios.

**1. Obtener credenciales**

* **GitHub**: Crear un token de acceso personal (PAT) desde <https://github.com/settings/tokens>, con acceso de lectura al repositorio.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

* **Netlify**: Crear un token desde <https://app.netlify.com/user/applications>.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

**2. Añadir a Jenkins**

Desde "Manage Jenkins → Credentials → Global → Add Credentials":

* **GitHub**:
  + Kind: Secret text
  + Secret: [token GitHub]
  + ID: github-token
  + Description: Token GitHub para Jenkins
* **Netlify**:
  + Kind: Secret text
  + Secret: [token Netlify]
  + ID: netlify-token
  + Description: Token Netlify para despliegue

En ambos casos encontraremos esto:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

**3. Uso en Jenkinsfile**

Dentro del pipeline, se puede usar:

withCredentials([string(credentialsId: 'netlify-token', variable: 'NETLIFY\_TOKEN')]) {

sh 'echo "Usando token: $NETLIFY\_TOKEN"'

}

**PASO 7: Crear archivo Jenkinsfile**

Crear un archivo Jenkinsfile en el repositorio con la siguiente estructura:

pipeline {

agent any

environment {

// Variables de entorno para los tokens

NETLIFY\_TOKEN = credentials('netlify-token') *// Se obtiene el token de Netlify desde las credenciales de Jenkins*

GITHUB\_TOKEN = credentials('github-token') // Se obtiene el token de GitHub desde las credenciales de Jenkins

}

stages {

stage('Checkout') {

steps {

// Clonamos el repositorio de GitHub usando las credenciales

git credentialsId: 'github-token', url: 'https://github.com/Mabarea/TFG\_MiguelBarea.git'

}

}

stage('Build') {

steps {

echo "No hay build porque es HTML estático"

}

}

stage('Deploy') {

steps {

// Desplegamos el código a Netlify usando su API

withCredentials([string(credentialsId: 'netlify-token', variable: 'NETLIFY\_TOKEN')]) {

sh '''

curl -H "Authorization: Bearer ${NETLIFY\_TOKEN}" \

-F "file=@index.html" \

https://api.netlify.com/api/v1/sites/TU\_SITE\_ID/deploys

'''

}

}

}

}

}

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**PASO 8: Crear Pipeline en Jenkins**

Crear un nuevo Pipeline en Jenkins que apunte al repositorio GitHub y utilice el Jenkinsfile para la configuración.

1. **Acceder a Jenkins**.

Entrar en la interfaz web de Jenkins en <http://localhost:8080>.

2. **Crear un nuevo proyecto de tipo "Pipeline":**

* Hacer clic en "Nuevo elemento" o "New Item" en la pantalla principal de Jenkins.
* Ingresar un nombre para el proyecto (por ejemplo, "Despliegue-Automático").
* Seleccionar "Pipeline" como tipo de proyecto.
* Hacer clic en "OK".

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

3. **Configurar el Pipeline:**

En la pantalla de configuración del nuevo Pipeline, se deben realizar las siguientes configuraciones:

* Descripción (es opcional): Se puede incluir una breve descripción del pipeline.
* Fuente del código: Se debe conectar el repositorio de GitHub.
  + En el apartado "Pipeline", en la sección "Definition", se debe seleccionar Pipeline script from SCM (esto indica que Jenkins obtendrá el Jenkinsfile desde el repositorio).
  + En "SCM", se debe seleccionar Git.
  + En "Repository URL", se debe ingresar la URL del repositorio de GitHub, en mi caso:

https://github.com/Mabarea/TFG\_MiguelBarea.git

* + En "Credentials", no seleccionar nada.
  + En “Avanzado”, añadimos en “Name” el ID *github-token*, que incluye el token de GitHub previamente configurado.
  + En " Refspec” dejar en blanco si no se necesita configurar ramas específicas. Si solo se necesita una rama (por ejemplo, master), se puede usar el formato:
    - * +refs/heads/master:refs/remotes/origin/master

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

* **Configuración del Jenkinsfile:**
  + En el campo "Script Path", se debe asegurar que esté configurado como Jenkinsfile, que es donde se encuentra el código del pipeline (el que se ha editado previamente).

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.**

**4. Guardar y ejecutar el Pipeline:**

* Después de guardar los cambios, se podrá ejecutar el Pipeline de inmediato haciendo clic en "Construir ahora" o "Build Now".

Jenkins debería clonar el repositorio, ejecutar el pipeline definido en el Jenkinsfile y desplegar la aplicación en Netlify.

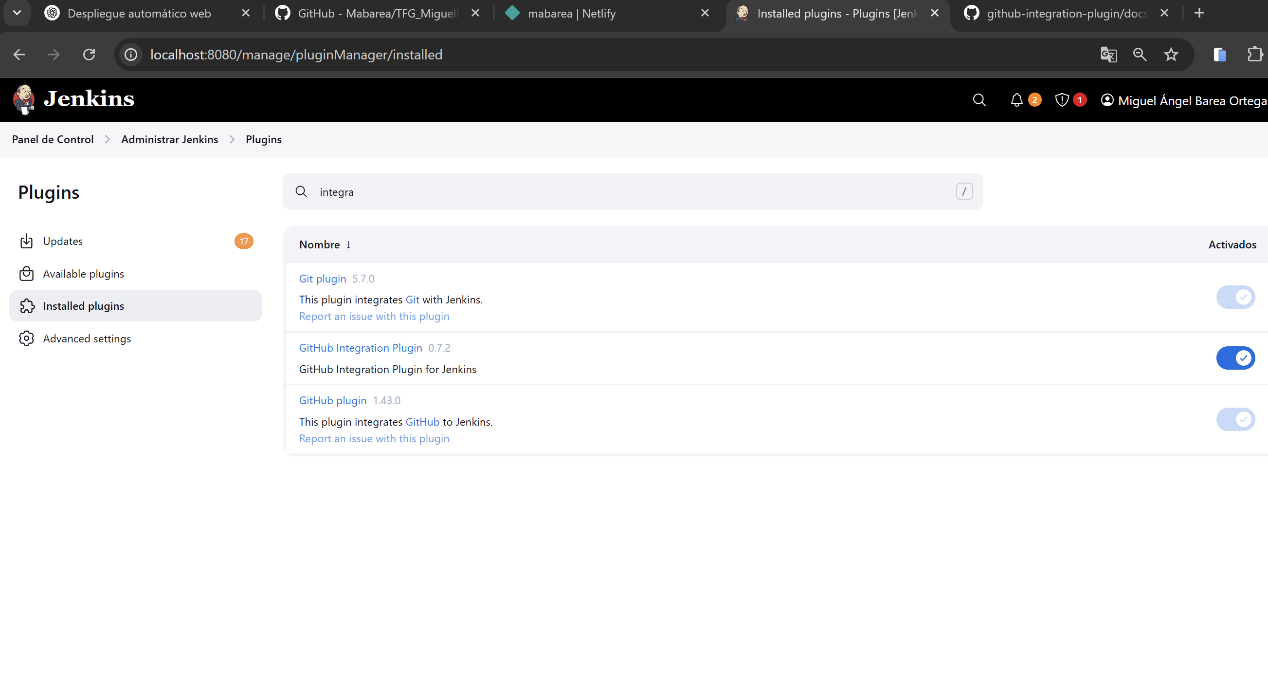
**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.**

**PASO 9: Automatizar con Webhooks (opcional pero recomendable)**

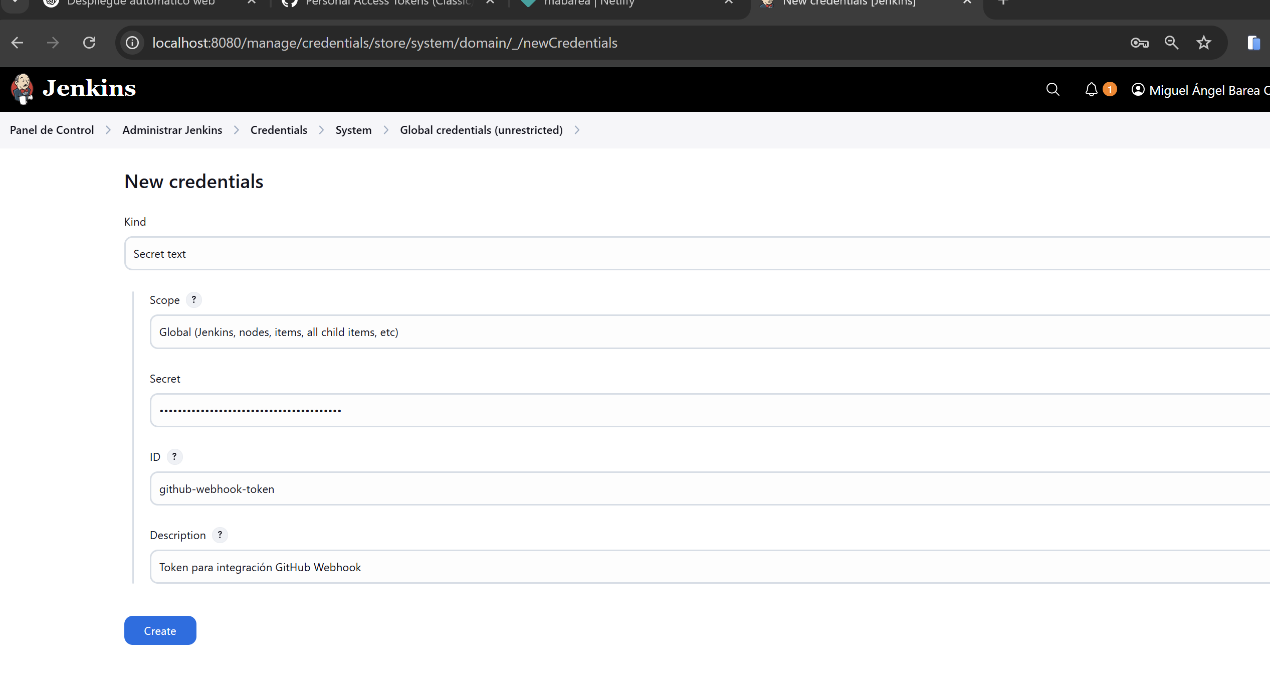
En GitHub, configurar un webhook que apunte a http://TU\_IP\_PUBLICA:8080/github-webhook/ para que Jenkins se ejecute al hacer push.

**1. Instalar el plugin necesario en Jenkins**

* Acceder a Jenkins con usuario administrador.
* Ir a **Manage Jenkins > Manage Plugins > Available**.
* Buscar e instalar **GitHub Integration Plugin** (o similar, que incluya webhook support).
* Reiniciar Jenkins si es necesario.

**2. Configurar las credenciales de GitHub en Jenkins**

* Ir a **Manage Jenkins > Manage Credentials > (global)**.
* Añadir nuevas credenciales:
  + Tipo: **Secret text** o **Username with password** (según token o user/pass).
  + Introducir el token personal de GitHub o credenciales.



* Guardar y anotar el ID de credencial para usar en el pipeline.

**3. Configurar el repositorio Git en Jenkinsfile o pipeline**

* En el Jenkinsfile, usar:

git credentialsId: 'github-token', url: 'https://github.com/usuario/repositorio.git', branch: 'main'

* Reemplazar github-token por el ID real de la credencial creada.
* Confirmar que el pipeline clona correctamente el repositorio.

**4. Configurar el Job en Jenkins para usar webhook**

* Crear un nuevo job o editar uno existente.
* En la sección **Build Triggers** activar **GitHub hook trigger for GITScm polling.**
* Guardar los cambios.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

**5. Configurar el Webhook en GitHub**

* Instalar **ngrok** en Windows.
* (Recomendado) Añadir ngrok al PATH:
  + Abrir Panel de Control → Sistema → Configuración avanzada del sistema → **Variables de entorno**.
  + En la sección "**Variables del sistema**", buscar la **variable Path**, hacer clic en Editar.
  + **Añadir** una **nueva entrada** con la ruta a la carpeta que contiene ngrok.exe (ejemplo: C:\Users\TuUsuario\ngrok).
* Crear cuenta y obtener Authtoken de ngrok
  + Ejecutar en la terminal (CMD o PowerShell):

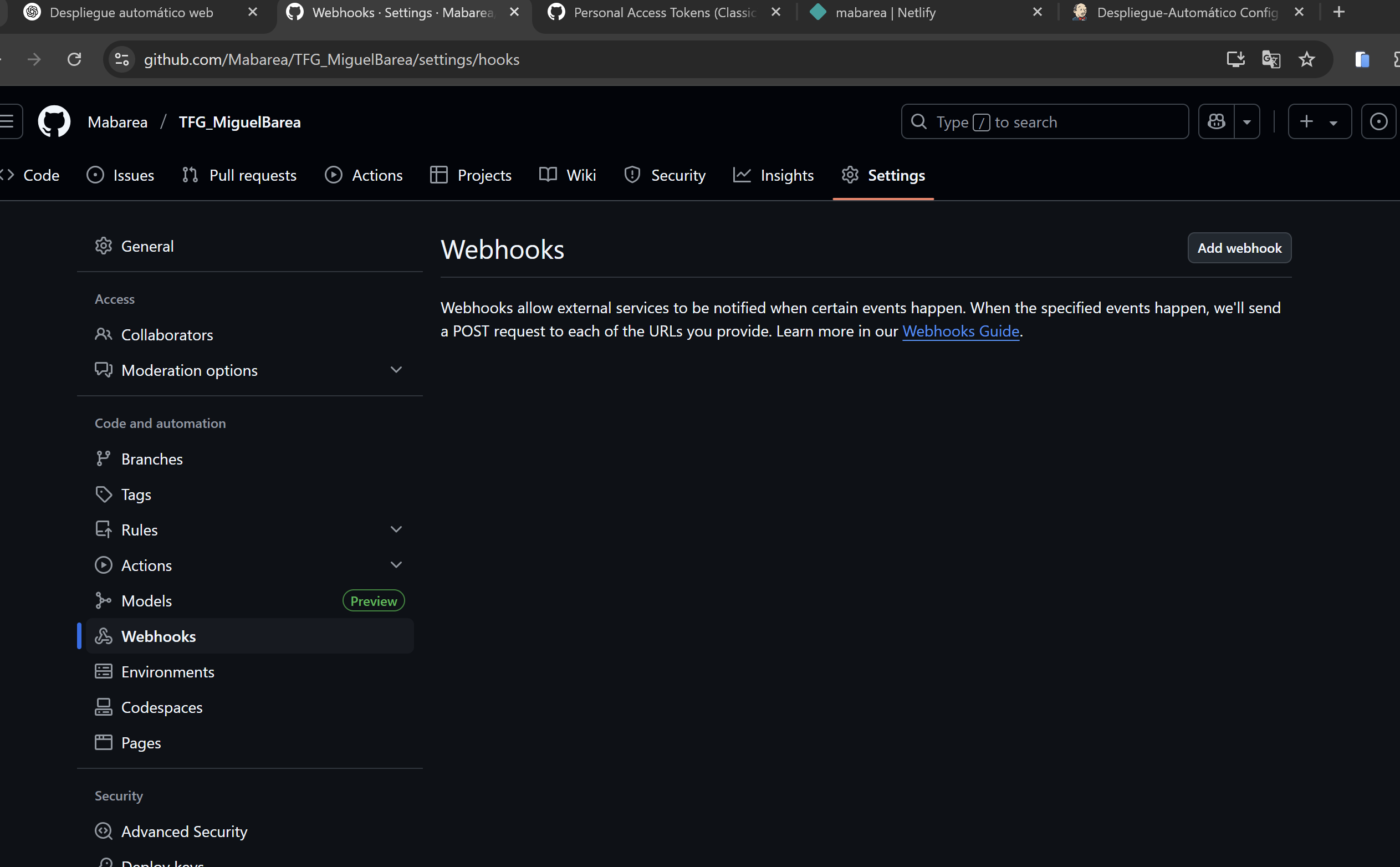
ngrok http 8080

* **Ejecutar ngrok para exponer el puerto de Jenkins** (que Docker mapea normalmente en el puerto 8080):

ngrok config add-authtoken $YOUR\_AUTHTOKEN

* Texto

  El contenido generado por IA puede ser incorrecto.Copiar la URL pública temporal, en este caso la indicada en “Forwarding”:
* Entrar en el repositorio GitHub.
* Ir a **Settings > Webhooks > Add webhook.**



* URL del webhook:

http://<JENKINS\_URL>/github-webhook/

* Content type: application/json.
* Elegir evento: **Just the push event.**
* Captura de pantalla de computadora

  El contenido generado por IA puede ser incorrecto.Guardar webhook.

**6. Probar la configuración**

* Hacer un push al repositorio GitHub.
* Verificar en Jenkins que el job se dispara automáticamente.
* Consultar logs para detectar posibles errores.

**7. (Opcional) Ajustar configuraciones de polling**

* En el job, desactivar polling si se usa solo webhook para evitar duplicidad.
* O bien, configurar polling con un intervalo alto para respaldo.

**PASO 10: Prueba final**

Modificar el index.html, hacer commit y push. Verificar que Jenkins ejecuta el pipeline y que la app se actualiza automáticamente.

# Conclusiones

* Se ha implementado un flujo CI/CD completo y funcional.
* Jenkins permite automatizar tareas de despliegue eficientemente.
* Netlify simplifica el proceso de publicación web.
* El sistema puede escalarse a proyectos más grandes y complejos.

# Glosario

**CI/CD**: Integración continua / Despliegue continuo.  
**Pipeline**: Flujo automatizado de tareas en un proceso de desarrollo.  
**Docker**: Plataforma para ejecutar aplicaciones en contenedores.  
**Webhook**: Notificación automática de eventos, como un push a un repositorio.

# Referencias bibliográficas utilizadas

* <https://www.jenkins.io/>
* <https://www.netlify.com/>
* Documentación oficial de Docker y GitHub

# Anexos

**9. Anexos**

*(Capturas de pantalla del proceso, configuraciones de Jenkins, salidas de consola, pruebas, etc.)*