



## DESAFIO 2

Kevin Damian Hidalgo – 99405

<b>1. Procedimiento para creación y configuración de un Pipeline.....</b>	<b>3</b>
1.1. Exponer una URL con Ngrok (opcional).....	3
1.2. Crear un Fork (opcional).....	3
1.3. Configuración de Webhook.....	5
1.4. Creación de Token en Github.....	6
1.5. Creación y configuración del pipeline .....	8
1.5.1. Instalación de NodeJS en la VM.....	8
1.5.2. Instalación de Plugins .....	9
1.5.3. Configuración de Tools .....	11
1.5.4. Crear una Credencial Jenkins .....	12
1.5.5. Creación de Pipeline .....	14
1.5.6. Configuración del Pipeline .....	15
<b>2. Glosario.....</b>	<b>18</b>
2.1. Branch: .....	18
2.2. Dashboard.....	18
2.3. Fork.....	18
2.4. Github.....	18
2.5. Jenkins .....	18
2.6. Log.....	18
2.7. Multipass .....	19
2.8. Ngrok.....	19
2.9. NVM.....	19
2.10. Pipeline.....	19
2.11. Plugins.....	19
2.12. Push.....	19
2.13. Pull Request .....	19

# 1. Procedimiento para creación y configuración de un Pipeline

Este instructivo detalla los pasos esenciales y las configuraciones opcionales necesarias para la creación y ejecución de un pipeline, tanto en su forma manual como automatizada, asegurando que se pueda implementar de manera eficiente y sin intervención del usuario.

## 1.1. Exponer una URL con Ngrok (opcional)

Para exponer un Jenkins a externos, se puede utilizar la aplicación gratuita Ngrok que crea una URL expuesta a internet, siguiendo los siguientes pasos:

- Acceder a la instancia (Maquina Virtual) ya creada en donde está instalado Jenkins (en este caso multipass), mediante el comando "multipass shell <Instancia>"
- Utilizar el siguiente comando de la aplicación Ngrok que está instalada en la misma instancia que Jenkins para exponer su servicio a externos

```
ngrok http http://localhost:8080
```

- Una vez ejecutado el comando se creará una URL expuesta a internet a la cual cualquier persona podrá acceder.

❖ **Observación:** Dicha URL expuesta tiene una duración en Horas, y si se cancela el comando de Ngrok ejecutado con "**Ctrl + c**" dejará de estar activa dicha URL.



```
ngrok (Ctrl+C to quit) ^
Found a bug? Let us know: https://github.com/ngrok/ngrok

Session Status      online
Account             kevin.d.hidalgo21@gmail.com (Plan: Free)
Version             3.14.1
Region              South America (sa)
Web Interface        http://127.0.0.1:4040
Forwarding           https://fc38-200-125-110-3.ngrok-free.app -> http://localhost:8080

Connections
t1l   opn   rt1   rt5   p50   p90
0     0     0.00  0.00  0.00  0.00
```

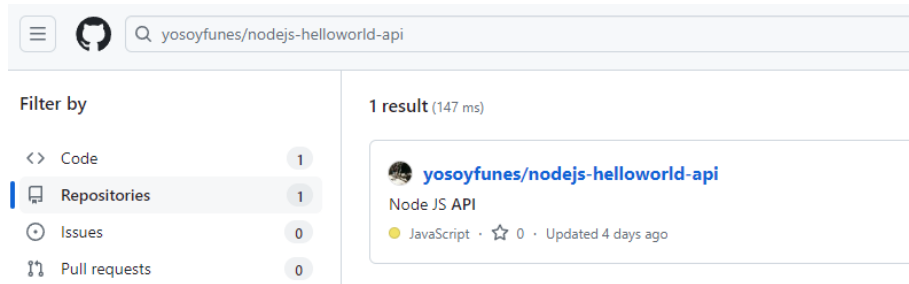
- Dicha URL expondrá el Jenkins, pero solo podrán acceder aquellos que posean credenciales en la misma.

## 1.2. Crear un Fork (opcional)

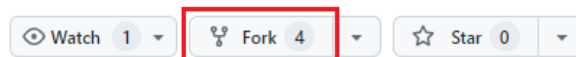
Con el fin de utilizar un repositorio externo (de otro usuario) se deberá crear un Fork del repositorio, se debe realizar los siguientes pasos:

- **Accede a GitHub:** Abre tu navegador web y dirígete a la página principal de GitHub en <https://github.com/>
- **Iniciar Sesión:** Ingresa tus credenciales de usuario y contraseña para acceder a tu cuenta de GitHub.

- **Buscar el repositorio:** En la barra de búsqueda de GitHub, ingresa **yosoyfunes/nodejs-helloworld-api** y presiona Enter.



- **Seleccionar resultado:** Selecciona el repositorio que aparece en los resultados de búsqueda para acceder a la página principal del repositorio.
- **Realizar el Fork:** En la esquina superior derecha de la página del repositorio, verás un botón que dice Fork al cual debes darle clic.



GitHub te llevará a una página donde podrás seleccionar tu cuenta o la organización donde deseas hacer el fork (Owner). Selecciona la opción deseada.

### Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (\*).

Owner \*      Repository name \*

/

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

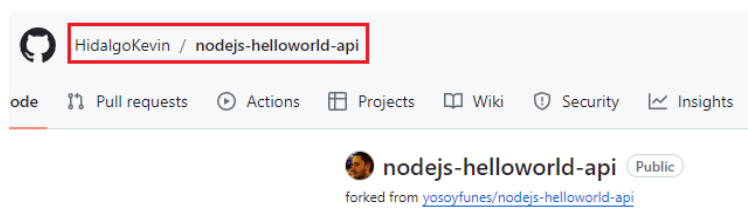
Description (optional)

☒ Copy the `main` branch only

Contribute back to yosoyfunes/nodejs-helloworld-api by adding your own branch. [Learn more.](#)

[Create fork](#)

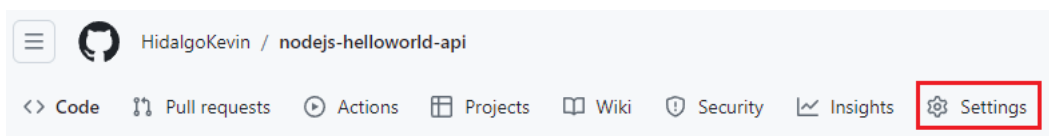
- **Verificar el Fork:** Después de hacer clic en **“Create Fork”**, serás redirigido a una copia del repositorio en tu propia cuenta de GitHub. El nombre del repositorio en este caso quedaría como **KevinHidalgo/nodejs-helloworld-api**.



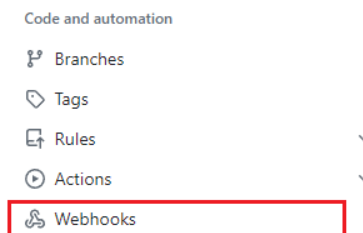
## 1.3. Configuración de Webhook

Para configurar un webhook en un repositorio de GitHub, se debe realizar los siguientes pasos:

- **Accede a GitHub:** Abre tu navegador web y dirígete a la página principal de GitHub en <https://github.com/>.
- **Iniciar Sesión:** Ingresa tus credenciales de usuario y contraseña para acceder a tu cuenta de GitHub.
- **Accede al Repositorio:** Ve al repositorio donde deseas configurar el webhook.
- **Ir a la Configuración del Repositorio:** En la página principal del repositorio, haz clic en la pestaña “Settings” ubicada en la parte superior derecha.



- **Acceder a Webhooks:** En el menú lateral izquierdo (**Code and automation**), desplázate hacia abajo y selecciona **Webhooks**.



- **Crear un Nuevo Webhook:** Haz clic en el botón **Add webhook**.

### Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

- **Configurar la URL del Payload:** En el campo Payload URL, ingresa la URL donde deseas que GitHub envíe las solicitudes POST cuando ocurra un evento en el repositorio (en este caso Jenkins).  
En el campo Payload URL, ingresa la URL donde deseas que GitHub envíe las solicitudes POST cuando ocurra un evento en el repositorio (en este caso Jenkins).
- El Payload URL quedaría como con el formato **URL\_Ngrok/github-webhook/**.  
Ejemplo:  
<https://ad27-200-125-110-3.ngrok-free.app/github-webhook/>
- **Elegir el Tipo de Contenido:** En Content type, selecciona application/json para que GitHub envíe los datos del webhook en formato JSON.
- **Configurar el Secreto (Opcional):** Si deseas agregar una capa adicional de seguridad, puedes ingresar un secreto en el campo Secret. Este valor se utiliza para generar una firma que puedes verificar en tu servidor.

- **Seleccionar Eventos:** Selecciona los eventos que deseas que activen el webhook. Puedes elegir que se envíen datos para todos los eventos (**Just the push event.**) o para eventos específicos como push, pull request, entre otros. En este caso seleccionar la opción **Just the push event** y marcar las opciones Pull Request y Pushes (push).

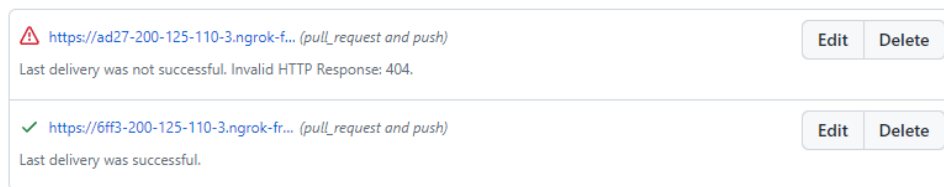
☒ **Pull requests**

Pull request assigned, auto merge disabled, auto merge enabled, closed, converted to draft, demilestoned, dequeued, edited, enqueued, labeled, locked, milestoned, opened, ready for review, reopened, review request removed, review requested, synchronized, unassigned, unlabeled, or unlocked.

☒ **Pushes**

Git push to a repository.

- **Finalizar la Configuración:** Revisa todas las configuraciones y luego haz clic en Add webhook para crear el webhook.
- **Verificación del Webhook:** Una vez creado el webhook, GitHub enviará una solicitud de prueba a la URL que has configurado (pull). Puedes verificar en la página de Webhooks si la solicitud fue exitosa. En esta screen se puede visualizar 2 casos en los cuales uno falló ya que la URL no existe.



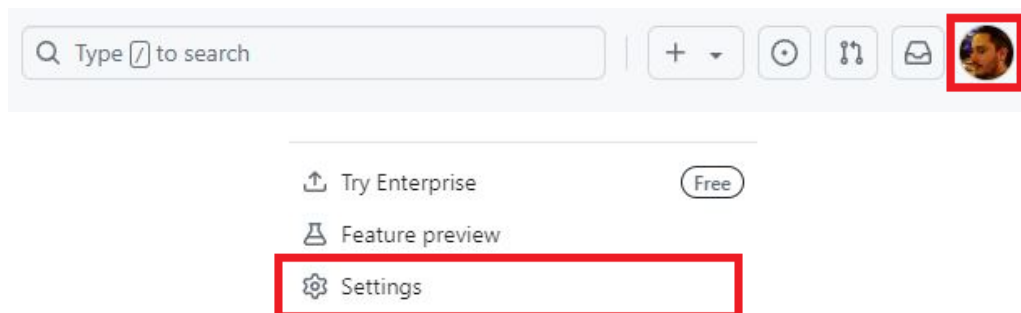
Siguiendo estos pasos, habrás configurado un webhook en tu repositorio de GitHub. Este webhook enviará datos a la URL especificada cada vez que ocurra uno de los eventos seleccionados.

## 1.4. Creación de Token en Github

Para que Jenkins pueda acceder a un repositorio de Github se debe configurar un token que es una cadena alfanumérica que se utiliza como una clave sensible de acceso la cual se utilizara al momento de crear una Credencial en Jenkins.

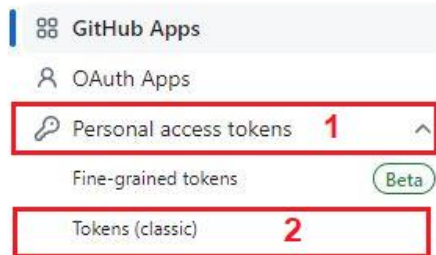
Los pasos para crear un token en Github son los siguientes:

- Se debe acceder a la opción Settings (configuración) de tu perfil de Github



- Se debe ir a la opción **Developer Settings** del menú de opciones a la izquierda.

- Se debe desplegar las opciones de **Personal Access token** y hacer clic en **Token (classic)**

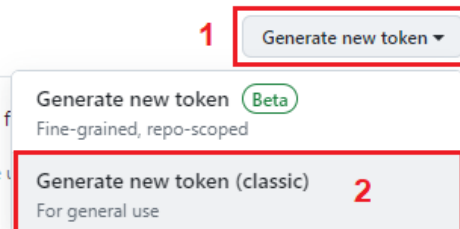


- Se debe hacer clic en **Generate new token** y utilizar la opción **classic**, utilizando de referencia la screen.

### Personal access tokens (classic)

Need an API token for scripts or testing? [Generate a personal access token](#) for your account.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used to [authenticate to the API over Basic Authentication](#).



- Se debe completar los campos
  - **Note:** Sería el nombre del token a crear
  - **Expiration:** Es el tiempo de vida del token a crear, en este caso se creó uno de 7 días.
  - **Select scopes:** Se debe seleccionar los tipos de permisos de tendrá el Token, en este caso, se seleccionó todos los permisos relacionados a **repo** (repository) ya que le permitirá tener acceso a su comportamiento.

## New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Note

Jenkins

What's this token for?

### Expiration \*

7 days

The token will expire on Sun, Sep 1 2024

### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events

- Una vez creada te generara un token alfanumérico el cual cumple la función de una password, por lo cual se debe resguardar el valor del token (el resaltado en verde) en un archivo temporal hasta que se utilice para crear una credencial en Jenkins

## Personal access tokens (classic)

Generate new token ▼

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp\_gzxpHzf0gKhBb2SEZkFcQxyb41q31c4F8vRM

Delete

## 1.5. Creación y configuración del pipeline

Se describirá en detalle el procedimiento para crear y configurar todos los elementos necesarios para asegurar el correcto funcionamiento del pipeline en un proyecto NodeJS.

### 1.5.1. Instalación de NodeJS en la VM

- Se debe realizar la instalación de NVM (Node Versión Management) utilizando el siguiente comando:



**`curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh | bash`**

- Una vez instalado la NVM, se debe instalar NodeJS especificando la versión a instalar.

**`nvm install 20`**

- **Observación:** Es posible que una vez instalado NVM (Node Version Management) debas reiniciar la terminal (salir de la VM e ingresar nuevamente).
- Una vez instalado NodeJS, se debe verificar la versión instalada en el entorno.

**`node -v`**

- Debería devolver **“v20.17.0”**
- Además se debe verificar la versión de NVM instalada en el entorno.

**`npm -v`**

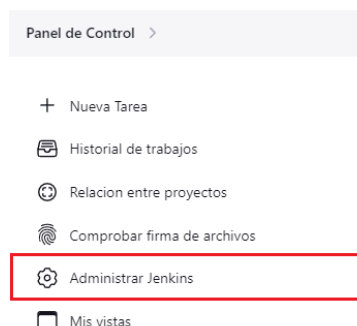
- Debería devolver **“10.8.2”**

---

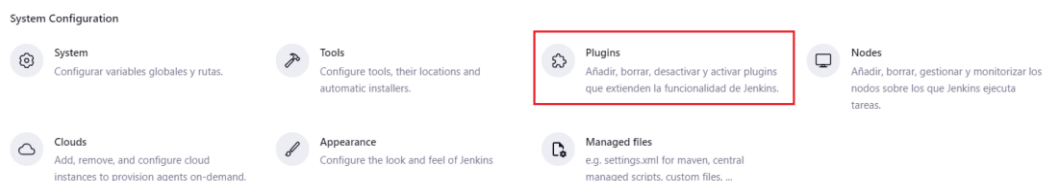
## 1.5.2. Instalación de Plugins

Al ser un proyecto de NodeJS, para evitar la ejecución del Pipeline de Jenkins falle se debe instalar el Plugin correspondiente a NodeJS. Los pasos son los siguientes:

- **Acceder a Jenkins:** Abre tu navegador web y accede a la URL de servidor Jenkins (en este caso la URL expuesta por Ngrok).
- **Iniciar sesión:** Ingresa tus credenciales de usuario y contraseña para acceder al dashboard de Jenkins.
- **Acceder a la Gestión de Plugins:**
  - En el tablero de Jenkins, haz clic en "Manage Jenkins" (Gestionar Jenkins) en el menú de la izquierda.

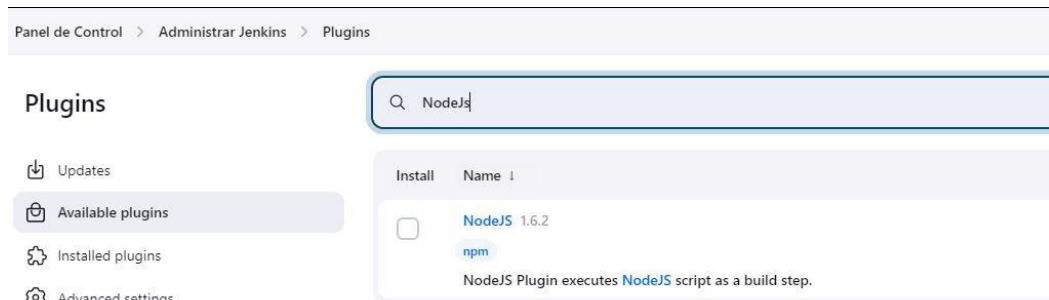


- Dentro de la página de gestión, selecciona la opción “Plugins” de System Configuration.



- **Buscar el Plugin de NodeJS:**

- Dentro de la pestaña **Available** (Disponible), utiliza la barra de búsqueda para encontrar el plugin escribiendo "NodeJS".
- Marca la casilla al lado del plugin de **NodeJS** y presiona el botón **Install** (Instalar).



➤ **Instalar el Plugin:**

- Haz clic en el botón **Install without restart** (Instalar sin reiniciar) para comenzar la instalación.

## Download progress

Preparación

- Checking internet connectivity
- Checking update center connectivity
- Success

Config File Provider ✓ Actualizado

NodeJS ✓ Actualizado

Loading plugin extensions ⋮ Running

→ [Volver al inicio de la página](#)  
(puedes empezar a usar los plugins instalados inmediatamente)

→ ☐ Reiniciar Jenkins cuando termine la instalación y no queden trabajos en ejecución

- Una vez haya finalizado la descarga y la instalación del Plugin, si no hay otros plugin por descargar, se debe marcar la casilla de **“Reiniciar Jenkins cuando termine la instalación y no queden trabajos en ejecución”**.
  - **Importante:** El reinicio puede demorar bastante tiempo, por lo cual se recomienda instalar todos los plugin necesarios y verificar que sea fuera de horario laboral antes de realizar el reinicio para evitar generar indisponibilidad del servicio.
- Procederá a reiniciar Jenkins para aplicar el Plugin instalado.



Por favor espera hasta que Jenkins acabe de reiniciarse. ...

Su navegador recargará esta página cuando Jenkins esté listo.

Safe Restart

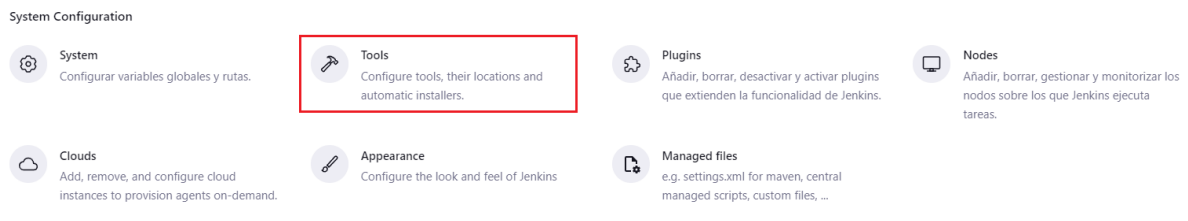
Builds on agents can usually continue.

- Una vez finalizado, te solicita ingresar con tus credenciales y ya estará habilitado el Plugin para su uso (finalizando el ciclo completo).

### 1.5.3. Configuración de Tools

Después de la instalación del plugin de NodeJS, se debe configurar el Tools correspondiente a dicho Plugin.

- Se debe ir a **Manage Jenkins**, se accede desde el Dashboard.
- Selecciona **Global Tool Configuration** (Configuración Global de Herramientas).



- Desplázate hacia abajo hasta encontrar la sección **NodeJS**.

instalaciones de NodeJS

instalaciones de NodeJS ▼

Edited

- Configura la versión de **NodeJS** que deseas usar en tu entorno de Jenkins. Asegúrate de marcar la opción **Install automatically** (Instalar automáticamente) si deseas que Jenkins maneje la instalación de NodeJS.

- En el campo Nombre se le puede setear un valor a gusto, pero en este caso como tomara el código del archivo Jenkinsfile del repositorio Github, el cual tiene un apartado tools, para que el pipeline lo reconozca se le debe colocar el mismo valor **nodejs**.

```
pipeline {
  agent any
  tools {
    nodejs 'nodejs'
  }
}
```

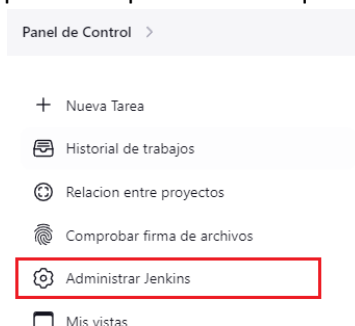
- En el campo Versión debe ser igual o similar a la versión instalada en la VM (**1.5.1 Instalación de NodeJS en la VM**), en este caso sería la versión **20.17.0**
- Después de configurar **nodejs**, haz clic en **Save** (Guardar) para aplicar los cambios.

Con estos pasos, tendrás el plugin de NodeJS instalado y configurado en Jenkins, listo para ser utilizado en tus pipelines.

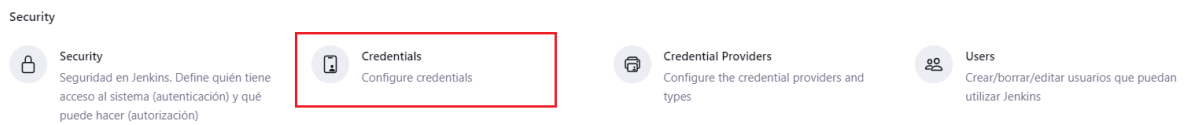
## 1.5.4. Crear una Credencial Jenkins

Una credencial Jenkins permite conectar, Jenkins con Github para acceder a un repositorio (entre otras opciones), los pasos para crear una credencial son:

- Acceder a Administrar Jenkins: Desde el Dashboard (Panel de Control) se debe acceder a la opción Administrar Jenkins del panel de opciones a la Izquierda.



- Debe acceder a la opción Credentials (Credenciales) en la lista de opciones de Security (Seguridad).



- Cuando se accede se podrá visualizar las credenciales existentes (si se crearon previamente) y para crear una nueva credencial, se debe acceder **System** de **Stores scoped to Jenkins**.

#### Credentials

T	P	Store	Domain	ID	Name
		System	(global)	Jenkins_Github	HidalgoKevin/***** (Conexion entre Jenkins y Github para el Desafio 2)


#### Stores scoped to Jenkins

P	Store	Domains
	System	(global)

- En **System**, acceder a **Global credentials (unrestricted)**

#### System

Domain	Description
	Global credentials (unrestricted)
	Credentials that should be available irrespective of domain specification to requirements matching.

- Una vez ingresado, se debe hacer clic en el botón  el cual desplegara una interfaz para completar los datos de la credencial.
  - Los campos **Kind** y **Scope** se dejan por defecto en este caso.
  - En el campo **Username** se le puede colocar el nombre de tu cuenta de Github (en mi caso KevinHidalgo)
  - En el campo **Password** debe ir valor del Token creado en Github (**1.4 Creación de Token en Github**)
  - El campo **ID** es un nombre con el cual podes identificarlo entre todas las credenciales que puedas crear.
  - El campo **Descripción** que es Opcional, puedes colocar en él un texto para transparentar el uso de dicha credencial.

Jenkins Credentials Provider: Jenkins

Global credentials (unrestricted) ▼

Kind

Username with password ▼

Scope ?

Global (Jenkins, nodes, items, all child items, etc) ▼

Username ?

HidalgoKevin

☐ Treat username as secret ?

Password ?

.....

ID ?

Jenkins\_Github

Description ?

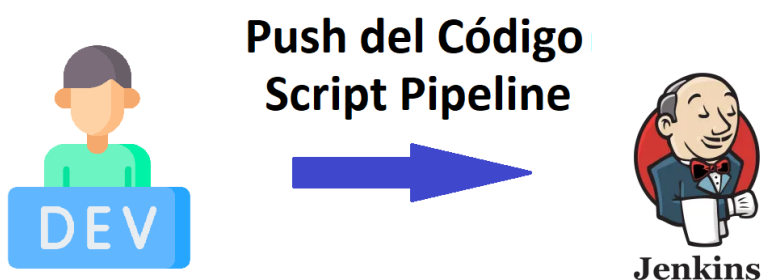
Conexion entre Jenkins y Github para el Desafio 2

- Una vez confirmada se creara la credencial la cual podrás utilizar al momento de configurar el pipeline.

### 1.5.5. Creación de Pipeline

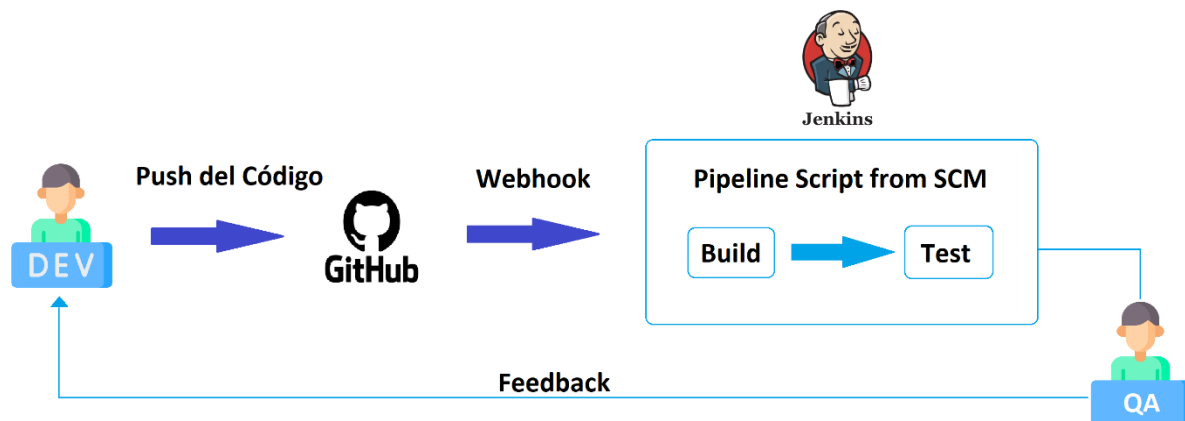
Este es un instructivo de la creación de un pipeline simple en Jenkins sin configuraciones:

- **Acceder a Jenkins:** Abre tu navegador web y accede a la URL de servidor Jenkins (en este caso la URL expuesta por Ngrok).
- **Iniciar sesión:** Ingresa tus credenciales de usuario y contraseña para acceder al dashboard de Jenkins.  
En la página principal de Jenkins, haz clic en **New Item** (Nuevo Elemento) en el menú de la izquierda.
- **Crear un Nuevo Pipeline:**
  - En la página de creación de un nuevo ítem, escribe un nombre para tu pipeline.
  - Selecciona la opción **Pipeline** y haz clic en **OK**.
- **Configurar el Pipeline:**
  - En la página de configuración del pipeline, ve a la sección **Pipeline**.
  - Aquí tienes dos opciones para definir el pipeline:
    - **Script Pipeline:** El código a ejecutar se define directamente en el Pipeline.



- **Pipeline Script from SCM:** Si tu código de pipeline está en un repositorio de control de versiones (como Git), selecciona esta opción y configura los detalles del repositorio.

Para esta opción seguir los pasos del punto **1.5.6) Configuración del Pipeline**



➤ **Guardar y Ejecutar el Pipeline:**

- Después de configurar el pipeline, haz clic en **Save** (Guardar).

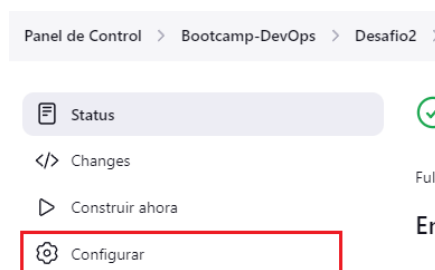
Este pipeline básico está configurado para ejecutarse manualmente, pero también puedes integrar webhooks de GitHub o configuraciones similares para automatizar su ejecución cuando se realicen cambios en el código.

## 1.5.6. Configuración del Pipeline

Para que un pipeline utilice el código de un repositorio en vez del código escrito en el pipeline, ya que es más óptimo porque reduce los cambios en el pipeline (cualquier cambio realizado se realiza en el repositorio de Github).

Se debe realizar los siguientes pasos para configurar el **Pipeline Script from SCM**.

- Posicionado en el Pipeline se debe acceder a la opción **Configurar**



- En la sección de **Build Triggers** se al activar la opción **GitHub hook trigger for GITScm polling** estaría habilitando que se dispare el pipeline de forma automática cuando se aplique un cambio sobre el repositorio de github configurado.

#### Build Triggers

- ☐ Construir tras otros proyectos ?
- ☐ Ejecutar periódicamente ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Consultar repositorio (SCM) ?
- ☐ Periodo de espera ?
- ☐ Lanzar ejecuciones remotas (ejem: desde 'scripts') ?

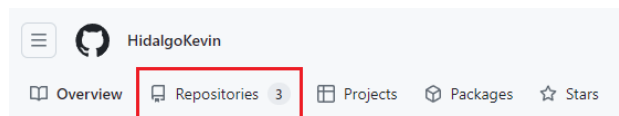
- En la sección **Pipeline** estará el campo Definición en donde se especifica qué tipo de Pipeline se quiere crear (**Script Pipeline** o un **Pipeline Script from SCM**).

En este caso se utilizara la opción **Pipeline Script from SCM**.

- En el campo **SCM** se especificara de que gestor de versionado se tomara el código del repositorio, en este caso se colocara la opción de **“Git”** que viene por defecto instalado en Jenkins.
- Al seleccionar la opción **“Git”**, se desplegaran unos campos a completar para que pueda llegar al repositorio en cuestión.
- Sección **Repositories**:
  - **Repository URL**: Se debe colocar la URL del repositorio en donde se encuentra el código.

Si no conoces la URL, puedes obtenerla siguiendo estos pasos:

- Ir a tu Github (con la sesión ya iniciada)
- Ir a la sección de Repositorios y acceder al que quieres vincular.



nodejs-helloworld-api Public

Forked from yosoyfunes/nodejs-helloworld-api

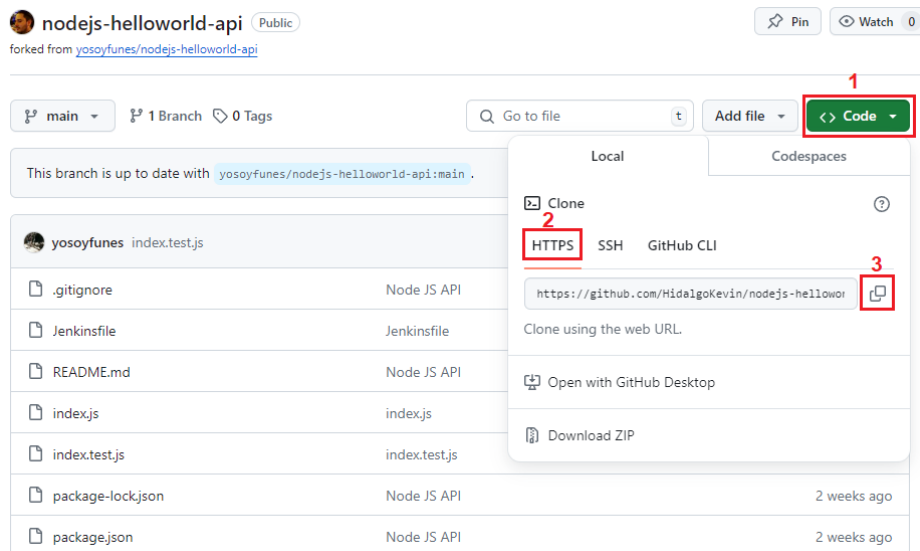
Node JS API

JavaScript Updated last week

Star

- Una vez que hayas accedido al repositorio, para obtener la URL se debe hacer clic en **<> Code**, hacer clic en la opción **HTTPS** y copiar la URL que retorna.

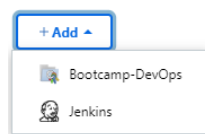




- Así podrás obtener la URL del repositorio a utilizar.
- **Credentials:** Se debe seleccionar la credencial que se utilizara para acceder al repositorio.

Si no posees una credencial creada, puedes generarla de 2 formas:

- **Opción 1:** Siguiendo los pasos del punto **1.5.4 Crear una Credencial Jenkins**.
- **Opción 2:** Presionar el botón **+ Add** y utilizar la opción **Jenkins**



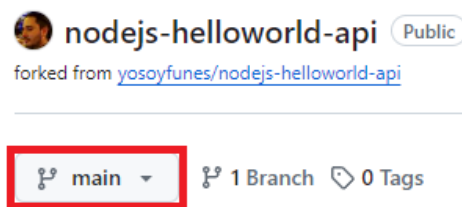
- La misma interfaz para cargar los datos de la credencial como en el punto **(1.5.4 Crear una Credencial Jenkins)**.
- Una vez creada o seleccionada la credencial, se debe colocar el nombre del Branch del repositorio para que tome el código.

Branches to build ?

Branch Specifier (blank for 'any') ?

\*/main

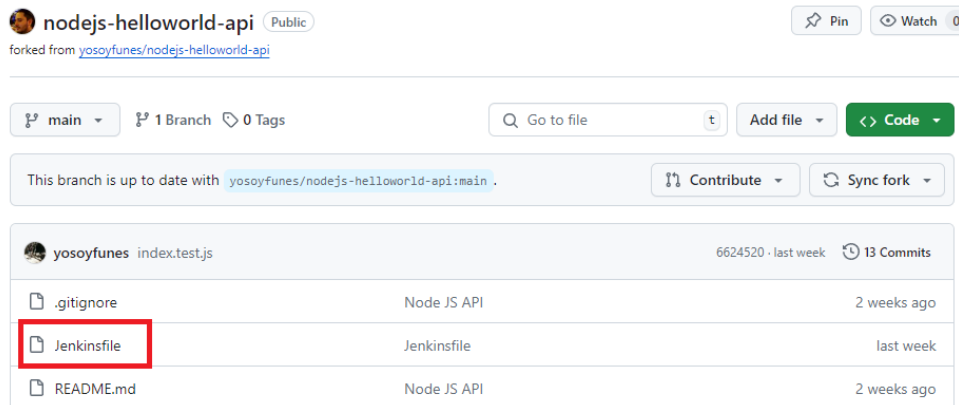
En este caso el nombre del branch es **/main** ya que así está en el repositorio.



- Una vez definido el branch, se debe definir el nombre del archivo Jenkinsfile que es el que contiene el código a ejecutarse.

Script Path ?

El nombre a colocar debe obtenerse del repositorio.



- Una vez configurado todo, se hace clic en Guardar.

De esta forma se crea y configura el pipeline correspondiente a Desafio2.

## 2. Glosario

Contiene una definición sobre los términos posibles que no conozcan.

### 2.1. Branch:

Las branch son ramas que permiten desarrollar características, corregir errores, o experimentar con seguridad las ideas nuevas en un área contenida de tu repositorio.

### 2.2. Dashboard

Es una herramienta de gestión de la información que monitoriza, analiza y muestra de manera visual.

### 2.3. Fork

Es un nuevo repositorio que comparte código y configuraciones de visibilidad con el repositorio original.

### 2.4. Github

Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git.

### 2.5. Jenkins

Es un servidor open source de integración continua que permite compilar y probar proyectos de software de forma continua.

### 2.6. Log

Es un registro secuencial y cronológico de las operaciones realizadas.

### **2.7. Multipass**

Es una herramienta flexible y potente, se puede utilizar para crear y destruir rápidamente máquinas virtuales (instancias) de Ubuntu en cualquier máquina host.

### **2.8. Ngrok**

Es una aplicación multiplataforma que crea túneles (rutas) seguros a la máquina host local. Permite a los desarrolladores exponer un servidor de desarrollo local a Internet con un mínimo esfuerzo.

### **2.9. NVM**

Node Version Manager, es una herramienta de línea de comandos que facilita la instalación, administración y cambio de versiones de Node. js en un sistema. NVM permite a los desarrolladores instalar y gestionar varias versiones de Node. js en una máquina sin interferencias entre ellas.

### **2.10. Pipeline**

Es una serie de complementos que permiten integrar elementos relacionados con la entrega continua en el sistema.

### **2.11. Plugins**

Es una aplicación (o programa informático) que permite extender las funciones de otra aplicación o programa sin tener que modificar el código

### **2.12. Push**

El comando push (git push) sube el contenido de un repositorio local (tu pc) a un repositorio central (github).

### **2.13. Pull Request**

Una solicitud de cambios es una propuesta para combinar un conjunto de cambios de una rama con otra. En una solicitud de cambios, los colaboradores pueden revisar y analizar el conjunto propuesto de cambios antes de integrar los cambios en el código base principal.