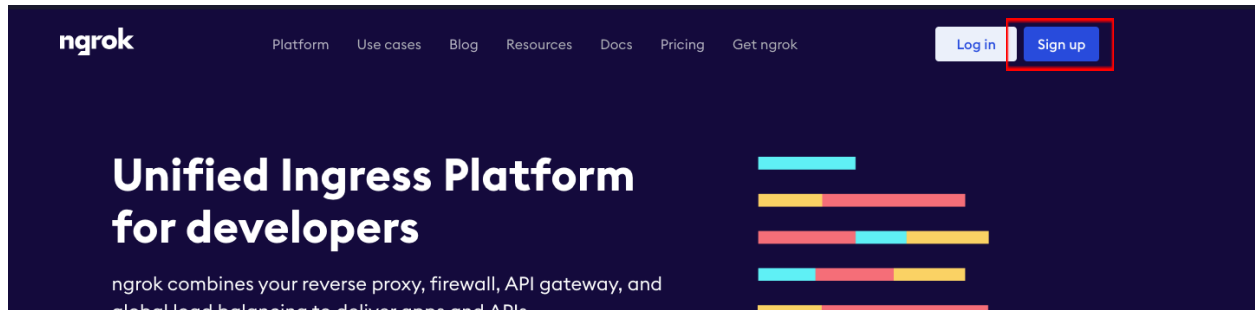


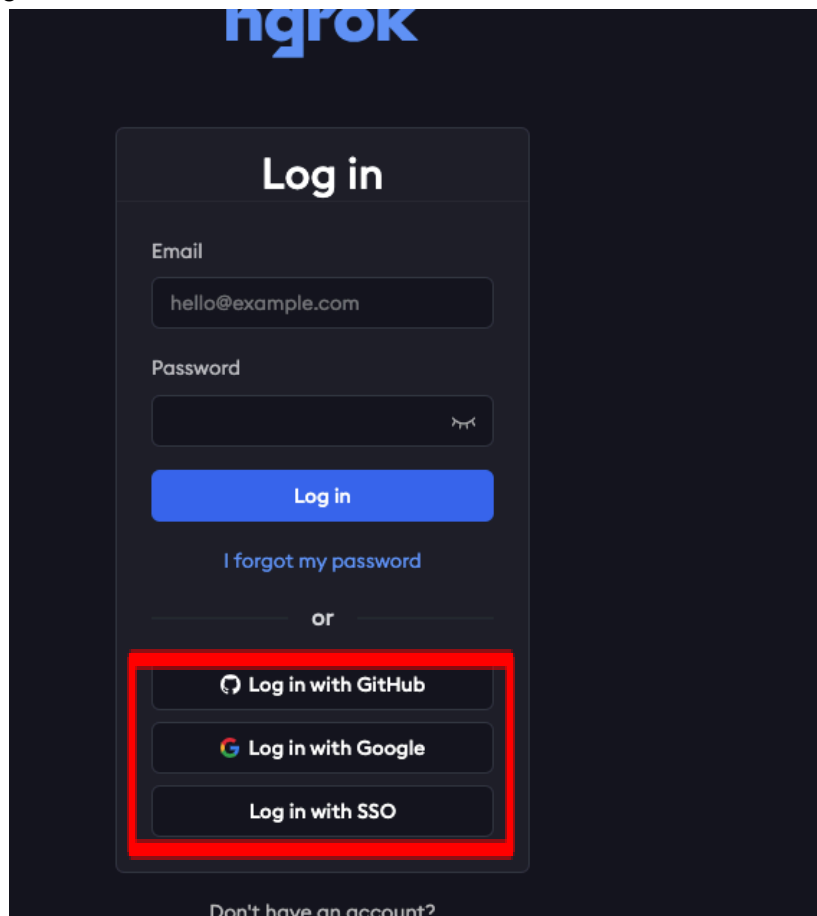
Entrega 2

Instalación de Ngrok:

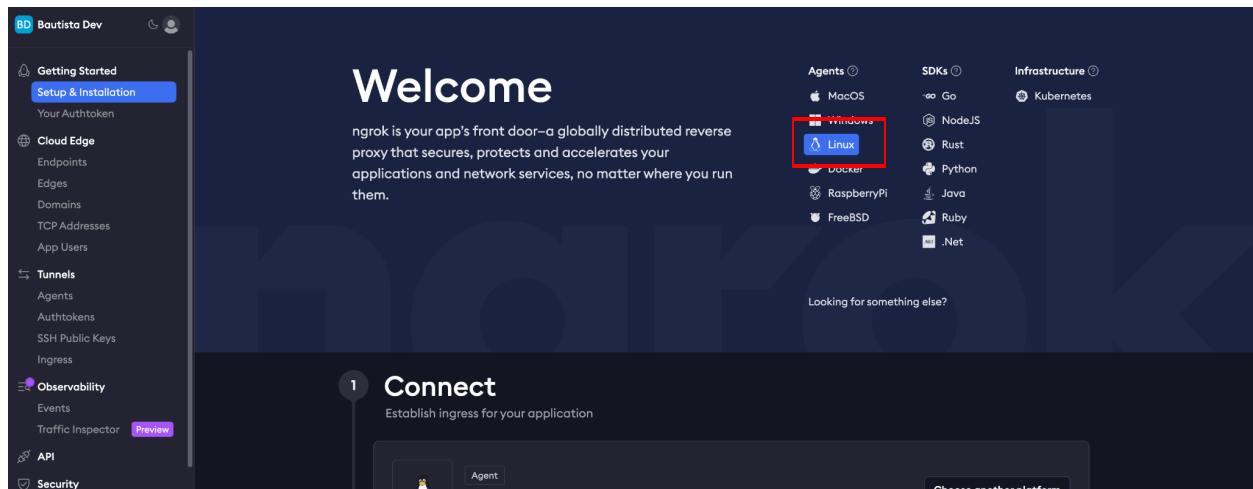
Ingresamos a la página principal de Ngrok y creamos una cuenta en el servicio:



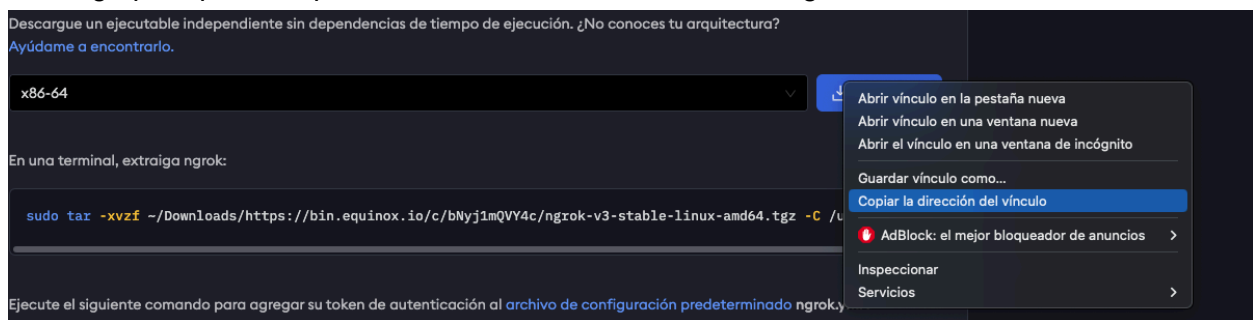
Ngrok nos facilita el registro, dándonos la oportunidad de poder registrarnos con nuestras cuentas de Google o Github:



Una vez dentro del menú principal de Ngrok, debemos elegir un sistema operativo en el cual configuraremos e instalaremos Ngrok. En este caso se elegirá Linux, preferentemente la distribución de Debian.

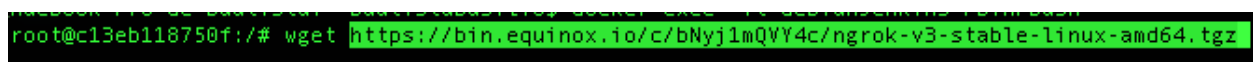


Nos dirigimos hacia la pestaña de descarga y hacemos click derecho sobre el botón azul de descarga para poder copiar la dirección del vínculo de descarga.



Con el uso del comando wget (paquete que debe ser preinstalado), descargamos los archivos de instalación que la página nos provee:

wget <https://bin.equinox.io/c/bNyj1mQVY4c/ngrok-v3-stable-linux-amd64.tgz>



Una vez descargado el archivo .tar, debemos descomprimirlo y redirigir el archivo ejecutable hacia el directorio de archivos binario del usuario “/usr/local/bin”.

```
tar -xvzf ~/Downloads/https://bin.equinox.io/c/bNyj1mQVY4c/ngrok-v3-stable-linux-amd64.tgz -C /usr/local/bin
```

Por último debemos agregar el token de autenticación, que enlaza nuestro entorno local con nuestra cuenta de Ngrok. Ejecutamos el siguiente comando:

Ejecute el siguiente comando para agregar su token de autenticación al [archivo de configuración predeterminado ngrok.yml](#).

```
ngrok config add-authtoken 2f0qLmjwnZPFYbhD6oJ9LUEIrBi_6g5Kzz5Mn6H1nzMmNBuJ6
```

Ya terminado los pasos anteriores, podremos colocar nuestra aplicación en línea en un dominio efímero y poder compartilo con otros usuario/personas.

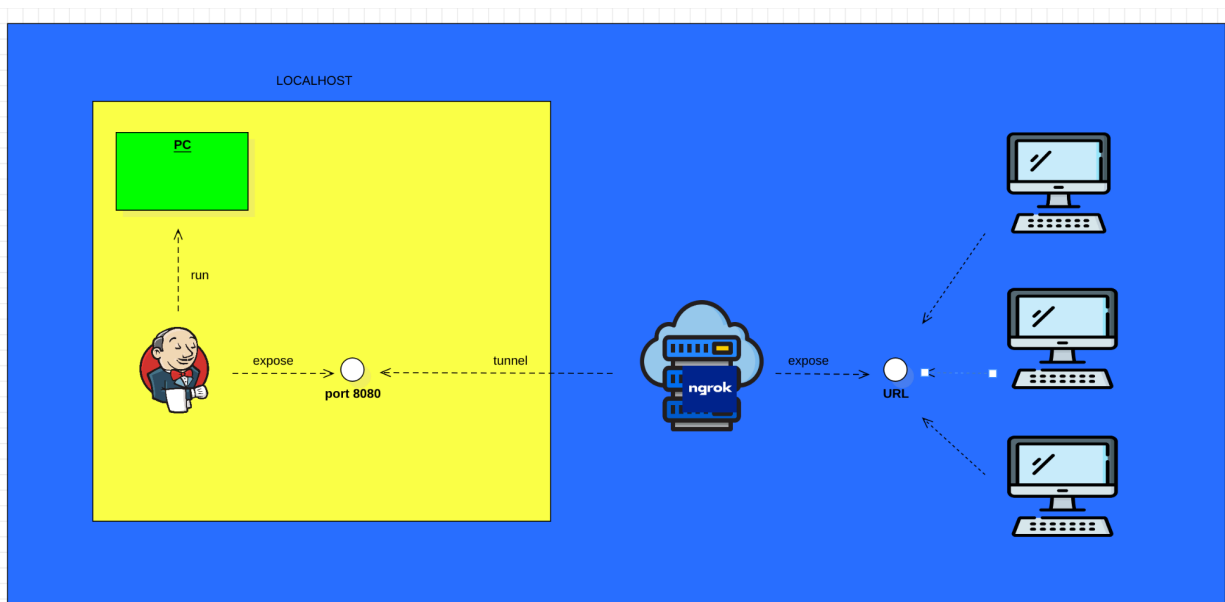
```
ngrok http http://localhost:8080
```

```
ngrok
Try our new Traffic Inspector Dev Preview: https://ngrok.com/r/ti

Session Status      online
Account             Bautista Dev (Plan: Free)
Version             3.8.0
Region              South America (sa)
Latency              45ms
Web Interface        http://127.0.0.1:4848
Forwarding            https://c925-198-188-227-229.ngrok-free.app -> http://localhost:8080

Connections          ttl    opn    rt1    rt5    p50    p99
                     101    0      0.00   0.00   1.60   55.82

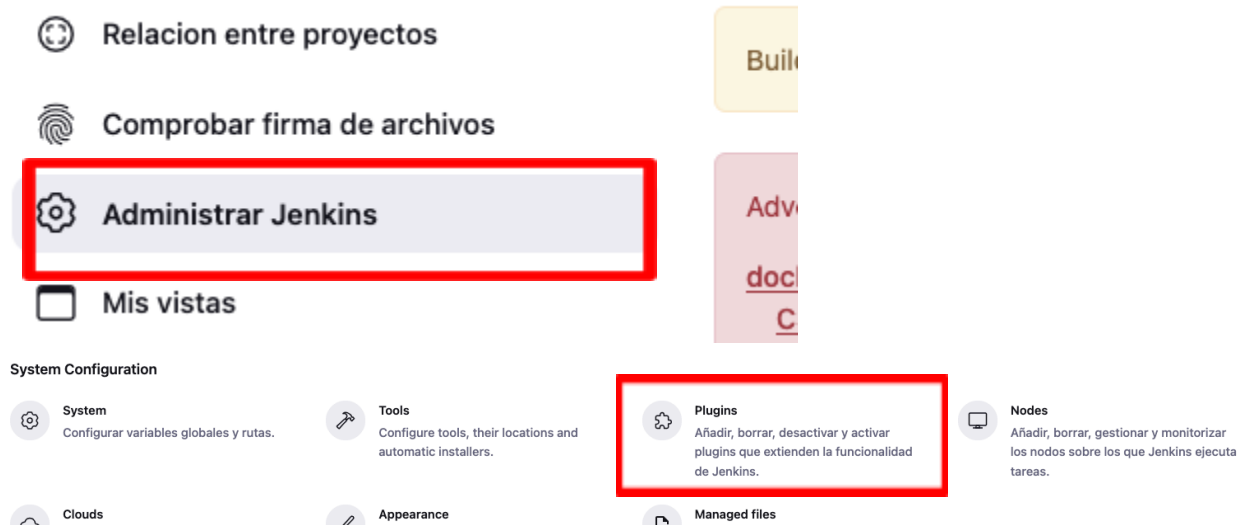
HTTP Requests
-----
GET /job/NodeJsPipeline/buildHistory/ajax 403 Forbidden
POST /manage/pluginManager/updates/body 200 OK
GET /job/NodeJsPipeline/wfapi/runs 200 OK
GET /job/NodeJsPipeline/wfapi/runs 200 OK
POST /manage/pluginManager/updates/body 200 OK
POST /manage/pluginManager/updates/body 200 OK
GET /job/NodeJsPipeline/wfapi/runs 200 OK
POST /manage/pluginManager/updates/body 200 OK
GET /job/NodeJsPipeline/wfapi/runs 200 OK
GET /job/NodeJsPipeline/wfapi/runs 200 OK
```



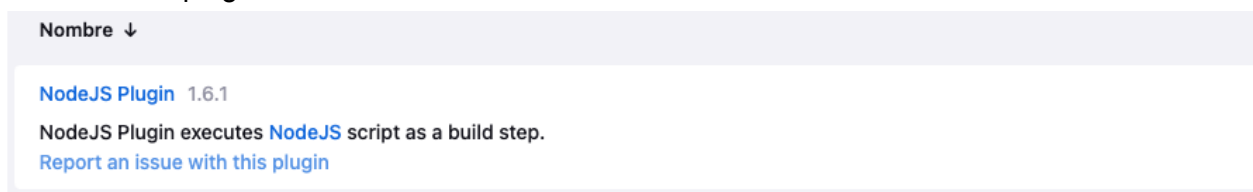
Creación del pipeline

Para poder crear nuestro pipeline, primero debemos instalar los plugins o extensiones que necesitamos, los cuales sean compatibles con el código de la aplicación que queremos testear en el pipeline.

Ingresamos en el administrador de Jenkins > Plugins

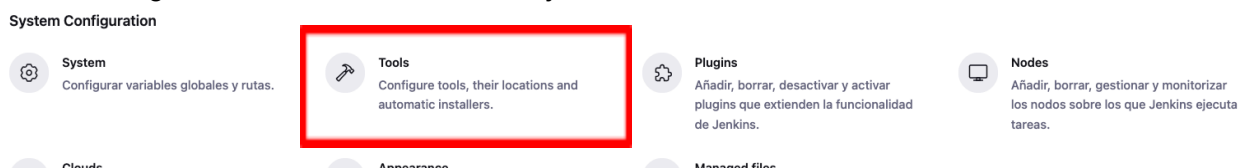


Instalamos el plugin de NodeJS:



Una vez instalada la extensión, debemos descargar una versión de NodeJS el cual utilizaremos para ejecutar los comandos necesarios en nuestro pipeline.

Para ello ingresados en Administrador de jenkins > Tools



Nos dirigimos hacia la sección de NodeJS, Ingresamos un nombre de esa instancia del paquete y la versión del paquete y damos guardar

instalaciones de NodeJS

instalaciones de NodeJS ^ Edited

Añadir NodeJS

≡ NodeJS

Nombre

nodeJs

☒ Instalar automáticamente ?

≡ Install from nodejs.org

Versión

NodeJS 21.7.3

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax "packageName@version"

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

72

Una vez terminada la descarga e instalación de la versión de nodeJS, vamos a crear el pipeline.

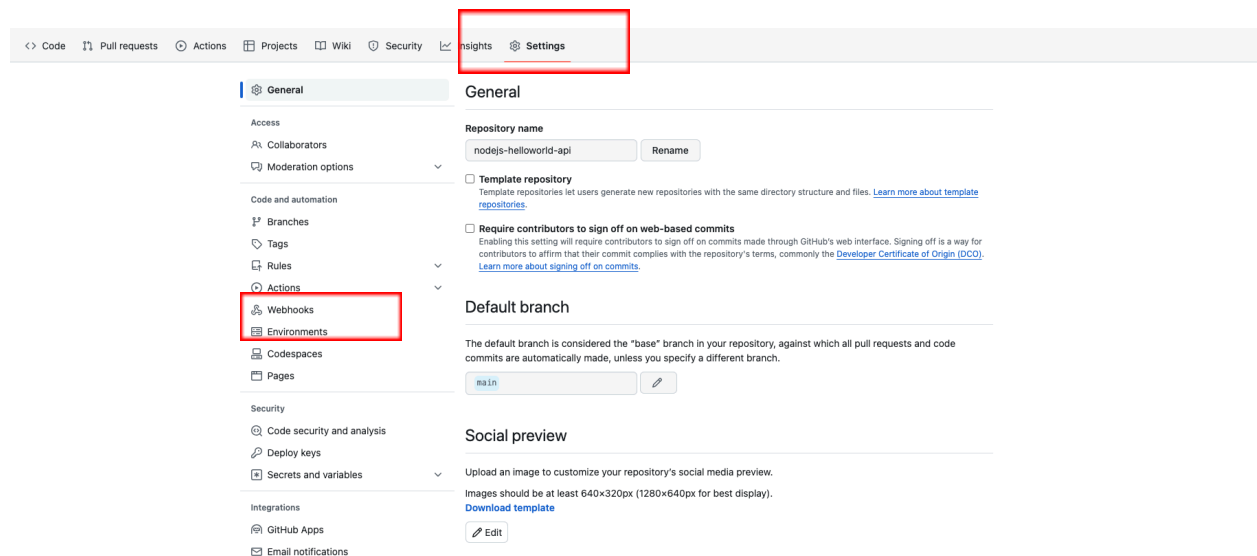
```
pipeline{
  agent any
  tools{
    nodejs "nodeJs" // definimos cual es la versión de node que vamos a usar
  }
  stages{
    stage('clone repository'){
      steps{
        //clonamos el repositorio
        git branch: 'main', credentialsId: 'Bautistadev', url: 'https://github.com/Bautistadev/nodejs-helloworld-api.git'
      }
    }
    stage('install'){
      steps{
        // instalamos las dependencia que utiliza nuestra aplicación
        sh 'npm install'
      }
    }
    stage('test'){
      steps{
        //Testeamos la aplicacion
        sh 'npm test'
      }
    }
    stage('docker build'){
      steps{
        script{
          //Construimos la imagen
          docker.image("nodejs-helloworld-api:v${BUILD_NUMBER}")
        }
      }
    }
  }
  post{
    always{
      echo 'Pipeline finalizado'
    }
    success{
      echo 'Ejecucion exitosa !!'
    }
    failure{
      script{
        if(currentBuild.result == 'FAILURE'){
          echo 'EL STAGE : '+env.STAGE_NAME+ ' HA FALLADO'
        }
      }
    }
  }
}
```

En la sección de construcción de triggers (Build Triggers). Seleccionamos la opción “GitHub hook trigger for GITScm polling”

Build Triggers

- ☐ Construir tras otros proyectos ?
- ☐ Ejecutar periódicamente ?
- ☐ Generic Webhook Trigger ?
- ☒ GitHub hook trigger for GITScm polling ?

Para poder ejecutar nuestro pipeline, cada vez que alguien realice un cambio en nuestro repositorio por medio de una sentencia push o commit sobre cierta rama/branch. Una de las soluciones es utilizar Github Webhook. Para ello debemos dirigirnos a nuestro repositorio distribuido en Github y en el apartado de Setting, seleccionar la opción de WebHooks.



Hacemos click en “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](#).

✓ <https://c925-190-188-227-229.n...> (push)

Edit

Delete

Y una vez en el menu de creacion de webhooks, debemos ingresar la url arrojada por el servidor de Ngrok o dominio que estemos utilizando

```
Try our new Traffic Inspector Dev Preview: https://ngrok.com/r/ti

Session Status      online
Account             Bautista Dev (Plan: Free)
Version             3.8.0
Region              South America (sa)
Latency             46ms
Web Interface        http://127.0.0.1:4040
Forwarding           https://c925-190-188-227-229.ngrok-free.app -> http://localhost:8080
Connections
  t1      opn      t1      t5      p50      p90
329       0       0.00    0.01    1.66     50.75
```

(Recomendado eliminar la S de la sentencia https al final de la cadena)

Copiamos la url y le concatenamos la cadena “/github-webhook/”

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify the format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [documentation](#).

Payload URL *

Content type

Y dependiendo de la situación tenemos la posibilidad de configurar el tipo de evento de nuestro disparador.

Which events would you like to trigger this webhook?

☐ Just the push event.

☐ Send me everything.

☒ Let me select individual events.

☐ **Branch or tag creation**
Branch or tag created.

☐ **Branch or tag deletion**
Branch or tag deleted.

☐ **Branch protection configurations**
All branch protections disabled or enabled for a repository.

☐ **Branch protection rules**
Branch protection rule created, deleted or edited.

☐ **Check runs**
Check run is created, requested, rerequested, or completed.

☐ **Check suites**
Check suite is requested, rerequested, or completed.

☐ **Code scanning alerts**
Code scanning alert created, fixed in branch, or closed.

☐ **Collaborator add, remove, or changed**
Collaborator added to, removed from, or has changed permissions for a repository.

☐ **Commit comments**
Commit or diff commented on.

☐ **Dependabot alerts**
Dependabot alert auto_dismissed, auto_reopened, created, dismissed, reopened, fixed, or reintroduced.

☐ **Deploy keys**
A deploy key is created or deleted from a repository.

☐ **Deployment statuses**
Deployment status updated from the API.

☐ **Deployments**
Repository was deployed or a deployment was deleted.

☐ **Discussion comments**
Discussion comment created, edited, or deleted.

☐ **Discussions**
Discussion created, edited, closed, reopened, pinned, unpinned, locked, unlocked, transferred, answered, unanswered, labeled, unlabeled, had a new comment, or had a new discussion.

☐ **Forks**
Repository forked.


Por último damos en guardar. Y hacemos una prueba modificando el repositorio, agregando un Dockerfile a este y veremos como nuestro servidor Jenkins a la escucha de este cambio, ejecuta el pipeline correspondiente.


 **nodejs-helloworld-api** Public


forked from [edgaregonzalez/nodejs-helloworld-api](#)


 Pin


 Watch 0

 main


 1 Branch


 0 Tags







 Add file


 Code

This branch is **3 commits ahead of** [edgaregonzalez/nodejs-helloworld-api:main](#).


 Contribute


 Sync fork


 Bautistadev Update Jenkinsfile	ec3db1f · 3 hours ago	 15 Commits
 .gitignore	Init project	2 weeks ago
 Dockerfile	FEAT:Dockerfile implementation	3 hours ago
 Jenkinsfile	Update Jenkinsfile	3 hours ago
 README.md	Update README.md	6 hours ago


 **#9**


(pending—En el periodo de gracia. Termina en 4.9 Seg)


 Borrar Pipeline


 Full Stage View


 Rename



 Pipeline Syntax

 GitHub Hook Log

 Historia de tareas Tendencia

 **#9** 13 abr. 2024 22:01

 Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	clone repository	install	test	docker build	Declarative: Post Actions
Average stage times: (Average full run time: ~16s)	1s	293ms	1s	3s	4s	1s	188ms
 #9 abr 13 19:01 1 commit	1s	486ms	1s	2s			
 #8 abr 13 18:29 No Changes	1s	195ms	1s	4s	4s	1s	208ms