Bootcamp DevOps Engineer by EducacionIT 2024~2025

realizado por Hernán Fernández Brando

Desafío#02

Objetivo

Este desafío tiene como objetivo realizar la configuración de un webhook en un repositorio de Github y también crear un simple pipeline para ejecutar un build y testear un proyecto NodeJS.

Escenario

Nuestra organización nos encomendó la tarea de realizar una prueba de concepto para crear un CI/CD de nodejs, necesitan entender cómo se debe realizar el proceso de build para una API hecha en nodejs. Por este motivo, nos entregaron una aplicación de ejemplo hecha en esta tecnología, este proyecto cuenta con una prueba integrada hecha en la herramienta Jest propia de este framework de desarrollo. El referente de desarrollo, nos encargó realizar la configuración de un webhook que permita hacer un build automático cada vez que se produzca un push o un pull request, esto puede ayudar para automatizar la ejecución del job para este CI/CD.

Requisitos

1. Crear un fork del siguiente proyecto en nuestro cuenta de Github: nodejs-helloworld-api
2. Realizar la configuración del github webhook para inicializar el job cada vez que se produce un push o un se crea un PR.
3. Exponer mediante ngrok el servicio de Jenkins para que este pueda ser accedido por Github.
4. Elaborar un jenkins pipeline para que ejecute los pasos para desarrollo, tomar las instrucciones del README.md del proyecto.

Documentación de referencia

Titulo	Descripción	URL
Ngrok developer ingress	Permite exponer un servicio local hacia internet.	https://ngrok.com/
Github Webhooks	Documentación oficial sobre los webhook.	https://docs.github.com/en/webhooks/about-webhooks
Jenkins Github	Plugin utilizado para integrar los webhook de github a Jenkins.	https://plugins.jenkins.io/github/

plugin		
Instalar nvm para nodejs	Guia como instalar nvm en un sistema linux, nvm permite correr distintas versiones de nodejs en nuestro entorno local.	https://nodejs.org/en/download/package-manager/

Entregables

Los entregables establecidos para este proyecto con:

1. Código fuente del pipeline de Jenkins publicado en un repositorio de Github (El repositorio debe ser público).
2. Guía detallada de cómo utilizar el job de Jenkins en un archivo formato DOC o PDF (Adicional al documento y como algo opcional puede crear un archivo README.MD como parte del repositorio).
3. El documento debe contener un apartado de evidencia de las pruebas con resultado exitoso.

Entregable 0. Preparación del entorno

Previo a la ejecución de los pipeline fue necesario realizar las siguientes acciones:

Instalar ngrok

Instalado en clase, incluso se habilitó la url fija disponible para las cuentas gratuitas.
La salida por linea de comandos, obtenida al disponibilizar en internet el recurso de la máquina multipass con jenkins instalado, es la siguiente:

```
ngrok
(Ctrl+C to quit)

Sign up to try new private endpoints https://ngrok.com/new-features-update?ref=

Session Status                online
Account                       hfbrando (Plan: Free)
Update                        update available (version 3.18.4, Ctrl-U to updat
Version                       3.18.1
Region                        South America (sa)
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://super-refined-gelding.ngrok-free.app -> h
Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Instalación de NodeJs para las automatizaciones de Jenkins

Como la automatización de jenkins se va a ejecutar en la instancia de multipass, se va a instalar un programa de node js y exponer el mismo en internet en un puerto determinado. Hay varias formas de compilar el proyecto de nodejs desde jenkins, una de ellas sería mediante comandos bash y sh. En este caso para compilar y ejecutar se requiere que esté instalado node js en la instancia multipass. Aunque si se instala el plugin de nodejs desde jenkins esto no es necesario. No obstante los pasos a seguir son los siguientes:

- Actualiza los paquetes de tu sistema:

sudo apt update
sudo apt upgrade -y

- Instala Node.js. Puedes instalar Node.js utilizando nvm (Node Version Manager) para tener más flexibilidad al instalar diferentes versiones. Primero, instala nvm:

  curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | bash

- Luego, carga nvm en tu sesión actual:

  export NVM_DIR="$HOME/.nvm" && [-s "$NVM_DIR/nvm.sh"] && . "$NVM_DIR/nvm.sh"

- Ahora, instala la versión de Node.js que necesites (por ejemplo, la LTS):

  nvm install --lts

- Verifica que Node.js y npm se hayan instalado correctamente:

  node -v
  npm -v

Instalar plugin `nvm` para `nodejs` en Jenkins

Siguiendo la misma explicación previa a la instalación de NodeJs, también se deben instalar los plugins de Node, debido a que Jenkins trabaja con plugins. Así como también es necesario instalar plugins para traer un repositorio de github, para utilizar un jenkins file, para utilizar Groovy, para comunicarse con slack, para mandar mails.

Desde el sitio web oficial de la documentación de jenkins (https://plugins.jenkins.io/nodejs/) recomiendan tres alternativas para instalar dicho plugin de node:

- Using the GUI: From your Jenkins dashboard navigate to Manage Jenkins > Manage Plugins and select the Available tab. Locate this plugin by searching for nodejs.
- Using the CLI tool: `jenkins-plugin-cli --plugins nodejs:1.6.2`
- Using direct upload. Download one of the releases and upload it to your Jenkins controller.

En la instancia de multipass con la que estoy trabajando opté por la opción 1, desde el entorno visual de jenkins.

- Luego se configuró el Plugin de NodeJS
- En Manage Jenkins --> Tools --> sección NodeJS installations.
  Añadí una nueva instalación, llamada `NodeJS`. Seleccioné la versión `23.3.0`.
- Guardé los cambios.

Configurar webhook en proyecto `github`

1. voy al repositorio
2. voy a settings --> Webhooks --> New
3. en Payload URL ingreso la url que expone el servicio de jenkins en la instancia de multipass y

agrego `/github-webhook/` según la documentación de github.
En mi caso quedaría `https://super-refined-gelding.ngrok-free.app/github-webhook/`

4. En la sección `Which events would you like to trigger this webhook?` marco "let me select individual events" y selecciono: `Pushes` y `Pull requests` tal y como lo indica el enunciado.
5. Guardo los cambios.

Configurar dependencia con webhook en `jenkins`

1. Para eso creamos un nuevo job del tipo `pipeline`.
2. Marcamos el checkbox `GitHub project` completo la url del repositorio público en el que se configuró el webhook hacia la instancia de jenkins que está expuesta con ngrok.
3. Luego en la sección `Build Triggers` marcamos `GitHub hook trigger for GITScm polling`.
4. En la sección Pipeline se selecciona `pipeline script` y se completa el script indicado en el readme del proyecto de nodejs con el que se va a trabajar.

Primeras ejecuciones

En la primera ejecución manual que el pipeline funcionó correctamente, ocurrió que en las siguientes ejecuciones incluyendo las ejecuciones automáticas fallaba.
Se me ocurrió hacer un "clean" del proyecto antes de hacer un install y afortunadamente eso solucionó el problema que había:

```
stage('Clean Previous Installations') {
    steps {
        sh 'rm -rf node_modules package-lock.json || true'
    }
}
```

este es el pipe agregado para que funcione. A continuación en la siguiente sección se muestra el código fuente final.

Posterior a últimas ejecuciones

La instancia multipass empezó a tornarse lenta, y jenkins empezó a fallar. Tuve que cerrar la ejecución de ngrok y al volver a ejecutar aparecía el siguiente mensaje.

```
Microsoft Windows [Versión 10.0.19044.1889]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>multipass shell devopsbootcamp
shell failed: [ssh client] channel creation failed: ''

C:\Windows\system32>multipass stop devopsbootcamp
Stopping devopsbootcamp /
```

Y se quedaba colgado/pensando en la última linea, por mucho tiempo. Con lo cual no pude copiar la prueba de ejecución automática, pero si la ejecución manual (que eran iguales, solamente diferían de que una decía en la primera línea que había sido ejecutada automáticamente por un push).

Entregable 1. Códigos Fuente Jenkins

```
pipeline {
    agent any

    tools {
        nodejs 'NodeJS'
    }

    stages {
        stage('Clone Repository') {
            steps {
                git url: 'https://github.com/hernan-fb/testUseForJenkins', bran
            }
        }
        stage('Clean Previous Installations') {
            steps {
                sh 'rm -rf node_modules package-lock.json || true'
            }
        }
        stage('Install Dependencies') {
            steps {
                sh 'npm install'
            }
        }
        stage('Run Tests') {
            steps {
                sh 'npm test'
            }
        }
        stage('Start Server') {
            steps {
                sh 'npm start &'
            }
        }
    }
    post {
        success {
            echo 'El proceso se ejecutó correctamente.'
        }
        aborted {
            echo "El proceso fue abortado correctamente"
        }
        failure {
            echo 'Hubo un error durante la ejecución del proceso'
        }
    }
}
```

Entregable 2. Guía de cómo utilizar el job

Readme Ejecución automática CI a partir de un push

A continuación puede verse una animación de un push y la ejecución automática y satisfactoria en jenkins (**solo visible desde archivos `.md`**)

Ejecucion Automática en GIF

Se adjunta guía de utilización del job en el formato indicado por el enunciado.

Entregable 3. Evidencia de las pruebas con resultado exitoso

Ejecución CI

```
Started by user Hernán F.B.

[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins
 in /var/lib/jenkins/workspace/desafio02/nodejs_CI_job@2
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Clone Repository)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/hernan-fb/testUseForJenkins
 > git init /var/lib/jenkins/workspace/desafio02/nodejs_CI_job@2 # timeout=10
Fetching upstream changes from https://github.com/hernan-fb/testUseForJenkins
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
 > git fetch --tags --force --progress -- https://github.com/hernan-fb/testUseF
 > git config remote.origin.url https://github.com/hernan-fb/testUseForJenkins
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # t
Avoid second fetch
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 0ae49ebed09faf348f6a6f2c6c93f69a617a298b (refs/remotes/or
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 0ae49ebed09faf348f6a6f2c6c93f69a617a298b # timeout=10
 > git branch -a -v --no-abbrev # timeout=10
 > git checkout -b main 0ae49ebed09faf348f6a6f2c6c93f69a617a298b # timeout=10
Commit message: "test for devopsbootcamp course"
 > git rev-list --no-walk 0ae49ebed09faf348f6a6f2c6c93f69a617a298b # timeout=10
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Install Dependencies)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
```

```
[Pipeline] sh
+ npm install
(node:25102) ExperimentalWarning: CommonJS module /var/lib/jenkins/tools/jenkin
Support for loading ES Module in require() is an experimental feature and might
(Use `node --trace-warnings ...` to show where the warning was created)

added 367 packages, and audited 368 packages in 42s

45 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 low, 1 moderate, 5 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Run Tests)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ npm test

> nodejs-helloworld-api@1.0.0 test
> jest --forceExit

  console.log
    Server is listening on port 3000

      at Server.log (index.js:10:13)

PASS ./index.test.js
  GET /
    ✓ responds with hello world message in JSON format (84 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.078 s
Ran all test suites.
Force exiting Jest: Have you considered using `--detectOpenHandles` to detect a
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Start Server)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ npm start
```

```
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
El proceso se ejecutó correctamente.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```