Imagen que contiene Logotipo

Descripción generada automáticamente**Trabajo Practico N° 2**

Nombre: Ramiro Nahuel Flores

Fecha:4/2/2024

Trabajo Practico: n°2

Titulo: nodejs-helloworld-api

Tabla de contenido

[Requisitos previos: 2](#_Toc189596632)

[Software necesarios: 3](#_Toc189596633)

[Requisitos para ejecutar el pipeline: 3](#_Toc189596634)

[Explicación del Pipeline: 4](#_Toc189596635)

[Resultado: 5](#_Toc189596636)

[Conclusión: 5](#_Toc189596637)

[Imágenes del resultado final: 6](#_Toc189596638)

[Resumen del Pipeline 7](#_Toc189596639)

[Log Completo (Versión Compacta) 7](#_Toc189596640)

**Desafío #2 - Educacion IT**

Bootcamp DevOps

**Introducción:**

El desafío consistió en la creación y configuración de un pipeline en Jenkins para automatizar el proceso de instalación de dependencias, ejecución de pruebas y despliegue de una API Node.js. El pipeline, utilizando Jenkins y GitHub, integra la automatización del flujo de trabajo para una aplicación Node.js, asegurando que el código esté libre de errores y listo para ser desplegado en producción.

Este informe describe los pasos seguidos para configurar el pipeline, los requisitos previos, la explicación detallada del pipeline y el resultado final.

### **Requisitos previos:**

Antes de crear y ejecutar el pipeline en Jenkins, se deben cumplir ciertos requisitos para asegurar que la configuración sea exitosa. Estos son los requisitos clave:

1. **Sistema operativo y entorno:**
   * **Ubuntu Server 24.04** o una distribución Linux similar instalada como servidor.
   * **Jenkins** instalado y funcionando en el servidor. Jenkins es una herramienta de integración continua que facilita la automatización de tareas repetitivas en el ciclo de vida del desarrollo de software.
2. **Configuración de Jenkins:**
   * Asegúrate de tener **Jenkins** instalado en el servidor y accesible desde un navegador web para poder configurar y gestionar los pipelines.
   * Los siguientes **plugins** deben ser instalados en Jenkins:
     + **Pipeline: Nodes and Processes**: Este plugin permite la ejecución de pipelines que requieren herramientas como Node.js, Maven, etc.
     + **NodeJS Plugin**: Este plugin es crucial para gestionar la instalación de Node.js en Jenkins.
3. **Repositorio en GitHub:**
   * Tener un repositorio de GitHub donde se aloje el código fuente de la API Node.js. Este repositorio debe estar configurado correctamente con los archivos de configuración necesarios para ejecutar el pipeline.
   * **Webhook de GitHub a Jenkins**: Es necesario configurar un webhook en GitHub para que, cada vez que se realice un push de código al repositorio, Jenkins ejecute automáticamente el pipeline.
4. **Configuración de Node.js:**
   * El pipeline utiliza una versión específica de **Node.js** (en este caso, la versión 21.0.0), por lo que es esencial que esta versión esté configurada correctamente en Jenkins.

### **Software necesarios:**

1. **Ubuntu Server 24.04**: Sistema operativo base para ejecutar Jenkins.
2. **Jenkins**:
   * Debe tener instalados los siguientes plugins:
     + **Pipeline: Nodes and Processes**.
     + **NodeJS Plugin**.
3. **GitHub**: Para alojar el código y configurar el webhook de integración con Jenkins.
4. **Node.js 21.0.0**: Versión de Node.js requerida para ejecutar el proyecto.

### **Requisitos para ejecutar el pipeline:**

Para ejecutar el pipeline correctamente, se deben seguir estos pasos de configuración:

1. **Instalar los plugins necesarios en Jenkins**:
   * Ingresar a **Manage Jenkins** en el dashboard de Jenkins.
   * Seleccionar **Manage Plugins** e instalar los siguientes plugins:
     + **Pipeline: Nodes and Processes**.
     + **NodeJS Plugin**.
2. **Configurar Node.js en Jenkins**:
   * Ir a **Manage Jenkins** -> **Tools** -> **NodeJS installations**.
   * Añadir la versión **Node.js 21.0.0** y asignarle el nombre nodejs21. Este nombre es importante porque se usará en el archivo **Jenkinsfile** para referenciar la instalación de Node.js.
   * Para que Jenkins reconozca la versión correctamente, también se debe configurar la variable de entorno PATH para Node.js.
3. **Configurar Webhook de GitHub a Jenkins**:
   * Ingresar al repositorio de GitHub del proyecto.
   * Ir a **Settings** -> **Webhooks** y añadir un nuevo webhook apuntando a la URL de Jenkins. Este webhook debe estar configurado para dispararse en eventos de push, de modo que cada vez que se suba un nuevo commit al repositorio, Jenkins ejecute el pipeline automáticamente.

### **Explicación del Pipeline:**

El pipeline está diseñado usando un archivo **Jenkinsfile** con una sintaxis declarativa que define el flujo de trabajo a seguir en Jenkins. El pipeline se divide en varias etapas (stages), cada una con una tarea específica:

1. **Stage - Checkout SCM**:  
   En esta etapa, Jenkins clona el repositorio desde GitHub y obtiene la última versión del código. Esto asegura que el código más reciente se utilice para las siguientes etapas del pipeline.
   * **Comando**:

groovy

CopiarEditar

checkout scm

1. **Stage - Tool Install**:  
   Jenkins instala la versión de **Node.js 21.0.0** especificada en la configuración de Jenkins. Este paso es necesario para que el pipeline ejecute correctamente las siguientes etapas del flujo de trabajo con la versión correcta de Node.js.
   * **Comando**:

groovy

CopiarEditar

tool name: 'nodejs21', type: 'NodeJS'

1. **Stage - Build**:  
   En esta etapa, se ejecuta el comando npm install para instalar todas las dependencias necesarias del proyecto. Esta acción garantiza que el entorno de desarrollo esté preparado para ejecutar el proyecto correctamente.
   * **Comando**:

sh

CopiarEditar

npm install

1. **Stage - Test**:  
   Se ejecutan las pruebas automáticas configuradas en el proyecto utilizando npm test. Las pruebas permiten validar que el código no tenga errores y funcione como se espera.
   * **Comando**:

sh

CopiarEditar

npm test

1. **Stage - Deploy (Opcional)**:  
   En esta etapa, se ejecuta el comando npm start, lo que pone en marcha el servidor de la aplicación. Este comando mantiene el servidor en ejecución de manera continua hasta que se decida detenerlo manualmente. Esta etapa es útil en un entorno de producción o de prueba.
   * **Comando**:

sh

CopiarEditar

npm start

### **Resultado:**

El pipeline fue ejecutado exitosamente, y se completaron todas las etapas sin problemas. El repositorio fue clonado correctamente, las dependencias fueron instaladas con éxito, las pruebas automatizadas pasaron sin errores y el servidor fue levantado correctamente. Los resultados del **test** fueron satisfactorios, lo que indica que la API funciona correctamente antes de ser desplegada.

### **Conclusión:**

Este pipeline de Jenkins facilita la automatización del flujo de trabajo en el desarrollo de software para una API Node.js. La integración de Jenkins con GitHub y Node.js asegura que el proceso de construcción, pruebas y despliegue sea eficiente y libre de errores. La configuración de este pipeline ayuda a mejorar la productividad del equipo de desarrollo, permitiendo realizar pruebas y despliegues de manera rápida y confiable.

## Imágenes del resultado final:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Resultado de la ejecución

**Informe del Pipeline - Jenkins**

**Pipeline: Node.js Hello World API**

**Usuario**: Ramiro Nahuel Flores  
**Repositorio**: [nodejs-helloworld-api](https://github.com/ramiro0987/nodejs-helloworld-api.git)  
**Fecha de ejecución**: [Fecha de la ejecución]

## Resumen del Pipeline

1. **Checkout del Repositorio**
   * **Acción**: Clonación del repositorio
   * **Resultado**: Repositorio clonado correctamente desde GitHub
   * **Commit**: *"Update nodejs version"*
2. **Instalación de Herramientas**
   * **Acción**: Desempaquetando Node.js v21.0.0
   * **Resultado**: Node.js instalado correctamente en Jenkins.
3. **Instalación de Dependencias**
   * **Acción**: Ejecutando npm install
   * **Resultado**:
     + 367 paquetes instalados.
     + 9 vulnerabilidades detectadas (3 bajas, 1 moderada, 5 altas).
     + Recomendación: Ejecutar npm audit fix para resolver vulnerabilidades.
4. **Ejecución de Pruebas**
   * **Acción**: Ejecutando npm test
   * **Resultado**:
     + **Pruebas**: 1 prueba ejecutada, 1 pasada correctamente.
     + **Mensaje de la prueba**: El servidor está escuchando en el puerto 3000.
     + **Detalles**:
       - **GET /**: Responde con el mensaje "Hello World" en formato JSON.

**Resultado Final**

* **Estado del Pipeline**: *Éxito*
* **Duración Total**: 0.387 segundos

## Log Completo (Versión Compacta)

Started by user ramiro nahuel flores

Cloning repository https://github.com/ramiro0987/nodejs-helloworld-api.git

Node.js v21.0.0 installed

npm install completed:

- 367 packages added

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

npm test completed:

- Test passed successfully

- Server listening on port 3000