



No __ s19070104 __

INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ

Libramiento Fresnillo-Tepetongo, Fracc. Los Cardos,
99863 Jerez de García Salinas, Zac.

GARCIA FLORES ANAHI

anahi.gf@tecjerez.edu.mx

Nallely Morales

INGENIERÍA EN SISTEMAS
COMPUTACIONALES

DevSecOps - Desarrollo, Seguridad y Operaciones de Software

Prof. Salvador Acevedo

9° SEMESTRE

EVALUACIÓN PARCIAL 3 [T3]

Fecha:12/11/25

REPORTE DE PRÁCTICA: Implementación de PIPELINES para CI/CD con Jenkins

REPORTE DE PRÁCTICA: Implementación de PIPELINES para CI/CD con Jenkins.....	2
1. OBJETIVO GENERAL.....	2
2. OBJETIVOS ESPECÍFICOS.....	2
3. DESARROLLO DE LA PRÁCTICA.....	2
3.1 Configuración inicial del entorno.....	2
3.3 Creación del Pipeline en Jenkins.....	5
3.4 Ejecución automática del pipeline por evento push.....	7
3.5 Reporte de análisis de código.....	9
3.6 Enlace del repositorio en GitHub.....	9
CONCLUSIÓN.....	10
Referencias:.....	10

1. OBJETIVO GENERAL

Implementar un proceso automatizado de Integración Continua (CI) y Entrega Continua (CD) utilizando Jenkins, con conexión a un repositorio de GitHub y webhook (Ngrok), para desplegar y analizar un proyecto en Laravel.

2. OBJETIVOS ESPECÍFICOS

- Configurar un servidor Jenkins en entorno local.
- Establecer una conexión segura entre Jenkins y GitHub mediante Ngrok.
- Crear un Pipeline declarativo que automatice las etapas de build, test y despliegue.
- Ejecutar automáticamente el pipeline al detectar un evento push en GitHub.
- Analizar el código fuente con herramientas de calidad y generar un reporte.

3. DESARROLLO DE LA PRÁCTICA

3.1 Configuración inicial del entorno

- Se instaló Jenkins en un servidor local con Java 17.
- Se creó un usuario administrador y se configuraron los plugins:
 - GitHub Integration Plugin
 - Pipeline
 - NodeJS Plugin (para React)
 - PHP Plugin (para Laravel)

El proyecto a integrar fue el sistema web de tutorías institucional, alojado en el repositorio:
<https://github.com/anhigf/Proyecto.git>

The screenshot shows the Jenkins Security configuration page. At the top, there is a header with tabs for 'Security - Jenkins'. Below the header, the URL is 'localhost:8080/manage/configureSecurity/'. The main content area has two sections: 'Permitir que los usuarios se registren.' (checkbox) and 'Autorización'. Under 'Autorización', it says 'Usuarios autenticados tienen privilegios para todo' and has a checkbox for 'Allow anonymous read access' which is checked. Below this is a 'Markup Formatter' section with a 'Plain text' dropdown. At the bottom are 'Guardar' and 'Apply' buttons.

3.2 Conexión NGROK con GITHUB (Creación del Webhook)

Para exponer Jenkins en un entorno local y recibir eventos de GitHub, se utilizó Ngrok, que genera una URL pública temporal.

Pasos realizados:

Ejecutar en consola:

```
ngrok http 8080
```

1. Copiar la URL generada

```
C:\Program Files\WindowsApps> + < Ctrl+C to quit >
ngrok
diamond Using ngrok for OSS? Request a community license: https://ngrok.com/r/oss
Session Status      online
Account            anhiGF (Plan: Free)
Update             update available (version 3.33.0, Ctrl-U to update)
Version            3.24.0-msix
Region             United States (California) (us-cal-1)
Latency            138ms
Web Interface     http://127.0.0.1:4040
Forwarding         https://dilative-jeniffer-spiritualistically.ngrok-free.dev -> http://localhost:8080
Connections        ttl     opn      rt1      rt5      p50      p90
                  0       0       0.00    0.00    0.00    0.00
```

En GitHub → Settings → Webhooks:

The screenshot shows a Microsoft Edge browser window with several tabs open. The active tab is 'Add webhook' under 'Settings' in the GitHub 'Proyecto' repository. The left sidebar shows various project settings like General, Access, Collaborators, and Webhooks (which is selected). The main content area is titled 'Webhooks / Add webhook' and contains fields for 'Payload URL' (set to 'https://example.com/postreceive'), 'Content type' (set to 'application/x-www-form-urlencoded'), and 'Secret'. A note at the top explains that GitHub will send a POST request to the specified URL with event details.

2. Confirmar que el webhook devuelva Status: 200 OK.

A Windows PowerShell window titled 'ngrok' is displayed. The output shows the configuration of an ngrok session and a list of incoming HTTP requests. The session status is 'online' with the URL `http://127.0.0.1:4040` mapped to `https://dilative-jeniffer-spiritualistically.ngrok-free.dev`. Below this, a table shows connection statistics. The final part of the output lists several POST requests to the GitHub webhook endpoint, all returning a status of 200 OK.

```
Windows PowerShell x + v

ngrok
(Ctrl+C to quit)

◆ Decouple policy and sensitive data with vaults: https://ngrok.com/r/secrets

Session Status          online
Account                 anhiGF (Plan: Free)
Version                3.24.0-msix
Region                 United States (California) (us-cal-1)
Latency                58ms
Web Interface          http://127.0.0.1:4040
Forwarding             https://dilative-jeniffer-spiritualistically.ngrok-free.dev -> http://localhost:8080

Connections            ttl     opn      rt1      rt5      p50      p90
                        8       0       0.00    0.00    30.07   31.79

HTTP Requests
-----
21:47:25.835 CST POST /github-webhook/          200 OK
21:39:58.527 CST POST /github-webhook/          200 OK
21:22:58.100 CST POST /github-webhook/          200 OK
21:13:20.052 CST POST /github-webhook/          200 OK
21:03:38.198 CST POST /github-webhook/          200 OK
20:54:06.900 CST POST /github-webhook/          200 OK
20:51:33.511 CST POST /github-webhook/          200 OK
20:06:38.307 CST POST /github-webhook/          200 OK
```

Resultado: GitHub puede enviar eventos a Jenkins cada vez que se hace git push.

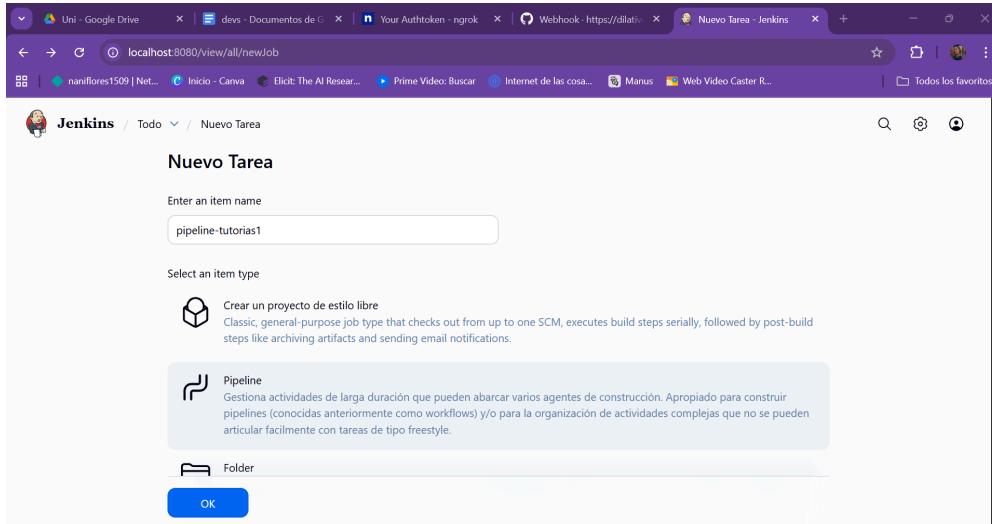
3.3 Creación del Pipeline en Jenkins

Se creó un nuevo trabajo tipo Pipeline en Jenkins

Definición: Pipeline script from SCM

SCM: Git

Repository URL: <https://github.com/anhiGF/Proyecto>



Jenkins / Todo / Nuevo Tarea

Nuevo Tarea

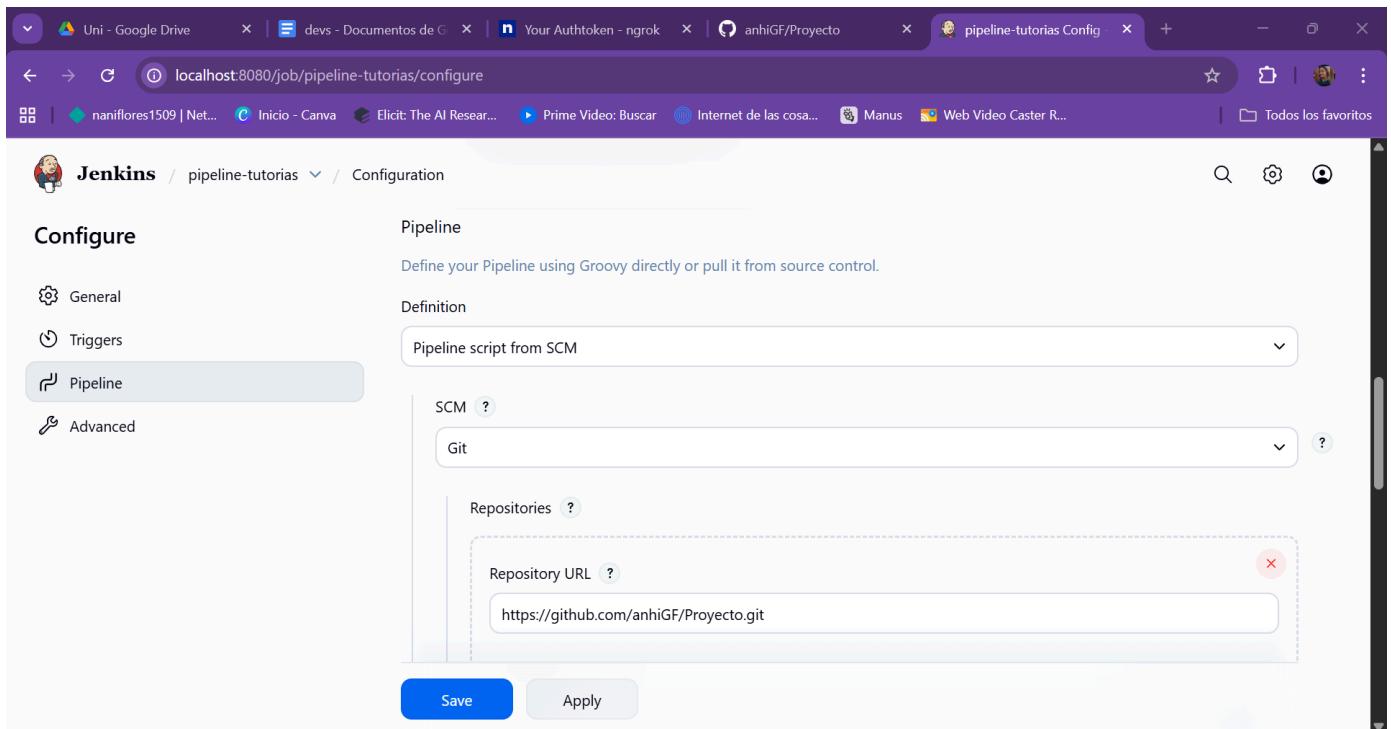
Enter an item name
pipeline-tutorias1

Select an item type

Pipeline Crear un proyecto de estilo libre
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Folder

OK



Jenkins / pipeline-tutorias / Configuration

Configure

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

General

Triggers

Pipeline

Advanced

Definition

Pipeline script from SCM

SCM ?

Git

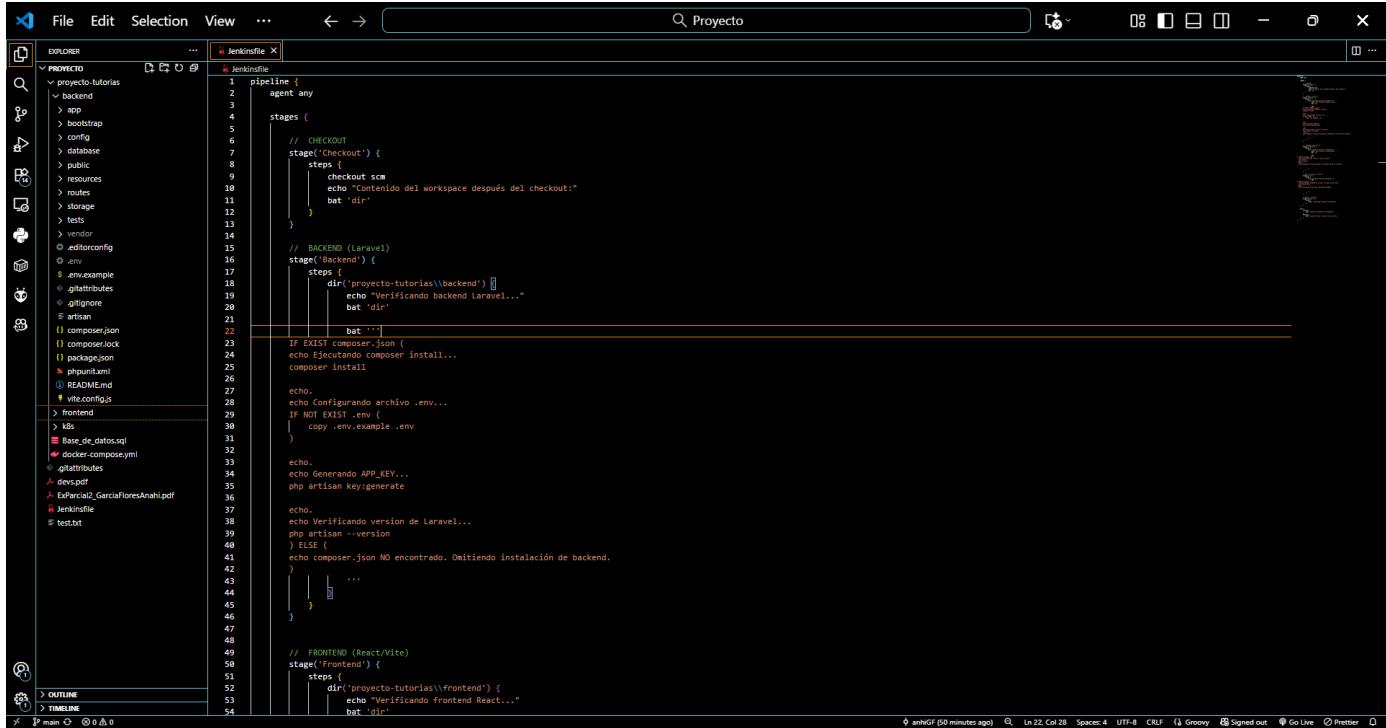
Repositories ?

Repository URL ?

https://github.com/anhiGF/Proyecto.git

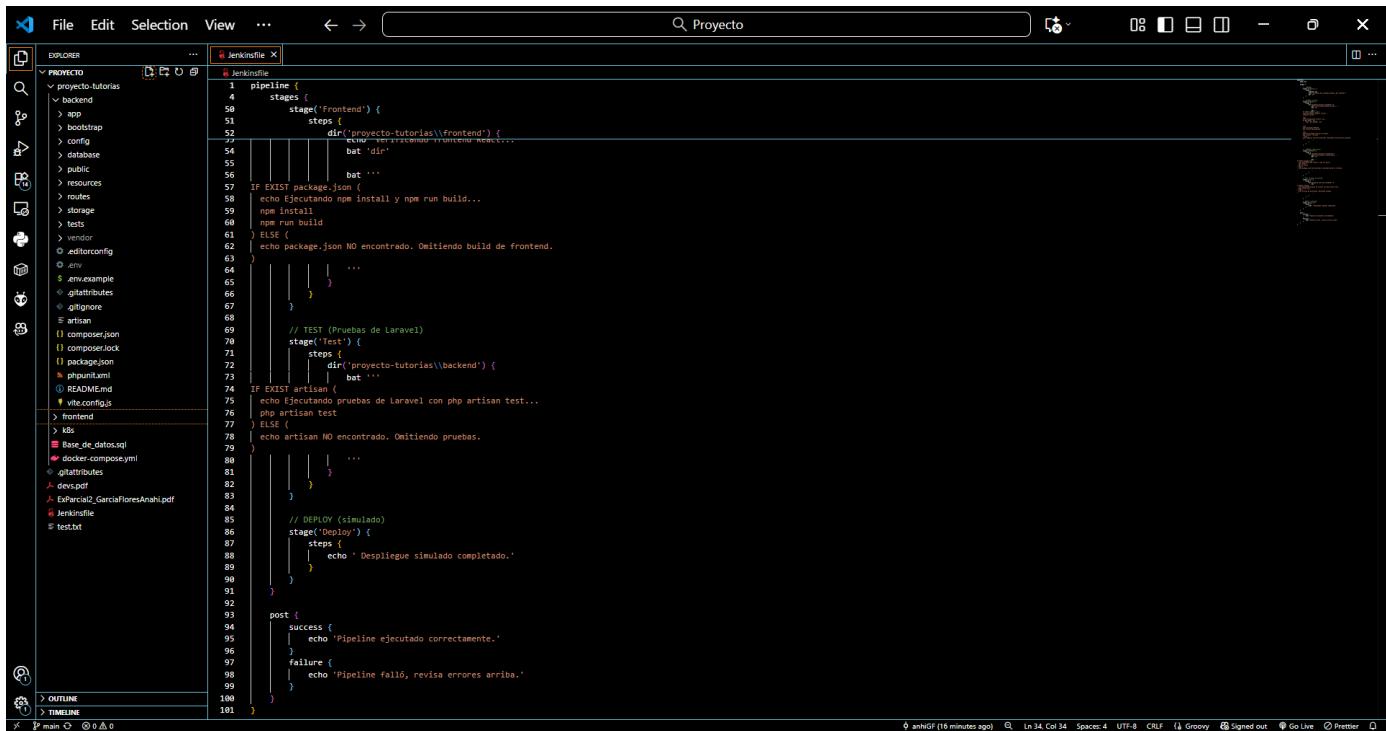
Save Apply

Contenido del archivo Jenkinsfile:



```
File Edit Selection View ... < > Jenkinsfile Proyecto

Jenkinsfile
1 pipeline {
2   agent any
3
4   stages {
5     // CHECKOUT
6     stage('Checkout') {
7       steps {
8         checkout scm
9         echo "Contenido del workspace después del checkout:"
10        bat "dir"
11      }
12    }
13
14    // BACKEND (Laravel)
15    stage('Backend') {
16      steps {
17        dir('proyecto-tutorias\\backend')
18        echo "Verificando backend Laravel..."
19        bat 'dir'
20
21        bat ''
22
23        if EXIST composer.json {
24          echo Ejecutando composer install...
25          composer install
26
27          echo.
28          echo Configurando archivo .env...
29          if NOT EXIST .env {
30            copy .env.example .env
31          }
32
33          echo.
34          echo Generando APP_KEY...
35          php artisan key:generate
36
37          echo.
38          echo Verificando versión de Laravel...
39          php artisan --version
40        } ELSE (
41          echo composer.json NO encontrado. Omitiendo instalación de backend.
42        )
43
44        bat ''
45
46      }
47
48
49    // FRONTEND (React/Vite)
50    stage('Frontend') {
51      steps {
52        dir('proyecto-tutorias\\frontend')
53        echo "Verificando Frontend React..."
54        bat 'dir'
55
56      }
57
58      if EXIST package.json {
59        echo Ejecutando npm install y npm run build...
60        npm install
61        npm run build
62      } ELSE (
63        echo package.json NO encontrado. Omitiendo build de frontend.
64
65        bat ''
66
67      }
68
69      // TEST (Pruebas de Laravel)
70      stage('Test') {
71        steps {
72          dir('proyecto-tutorias\\backend')
73          bat ''
74
75        if EXIST artisan {
76          echo Ejecutando pruebas de Laravel con php artisan test...
77          php artisan test
78        } ELSE (
79          echo artisan NO encontrado. Omitiendo pruebas.
80
81        bat ''
82
83      }
84
85      // DEPLOY (simulado)
86      stage('Deploy') {
87        steps {
88          echo 'Despliegue simulado completado.'
89
90        }
91
92      }
93
94      post {
95        success {
96          echo 'Pipeline ejecutado correctamente.'
97        }
98        failure {
99          echo 'Pipeline falló, revisa errores arriba.'
100      }
101    }
102
103  }
104
105}
```



```
File Edit Selection View ... < > Jenkinsfile Proyecto

Jenkinsfile
1 pipeline {
2   stages {
3     stage('Frontend') {
4       steps {
5         dir('proyecto-tutorias\\frontend')
6         echo "Verificando Frontend React..."
7         bat 'dir'
7
8         if EXIST package.json {
9           echo Ejecutando npm install y npm run build...
10          npm install
11          npm run build
12        } ELSE (
13          echo package.json NO encontrado. Omitiendo build de frontend.
14
15          bat ''
16
17        }
18
19      }
20
21      if EXIST artisan {
22        echo Ejecutando pruebas de Laravel con php artisan test...
23        php artisan test
24      } ELSE (
25        echo artisan NO encontrado. Omitiendo pruebas.
26
27      bat ''
28
29    }
30
31    // TEST (Pruebas de Laravel)
32    stage('Test') {
33      steps {
34        dir('proyecto-tutorias\\backend')
35        bat ''
36
37        if EXIST artisan {
38          echo Ejecutando pruebas de Laravel con php artisan test...
39          php artisan test
40        } ELSE (
41          echo artisan NO encontrado. Omitiendo pruebas.
42
43        bat ''
44
45      }
46
47      if EXIST artisan {
48        echo Ejecutando pruebas de Laravel con php artisan test...
49        php artisan test
50      } ELSE (
51        echo artisan NO encontrado. Omitiendo pruebas.
52
53      bat ''
54
55    }
56
57    // DEPLOY (simulado)
58    stage('Deploy') {
59      steps {
60        echo 'Despliegue simulado completado.'
61
62      }
63
64    }
65
66    post {
67      success {
68        echo 'Pipeline ejecutado correctamente.'
69      }
69
70      failure {
71        echo 'Pipeline falló, revisa errores arriba.'
72
73      }
74
75    }
76
77  }
78
79}
```

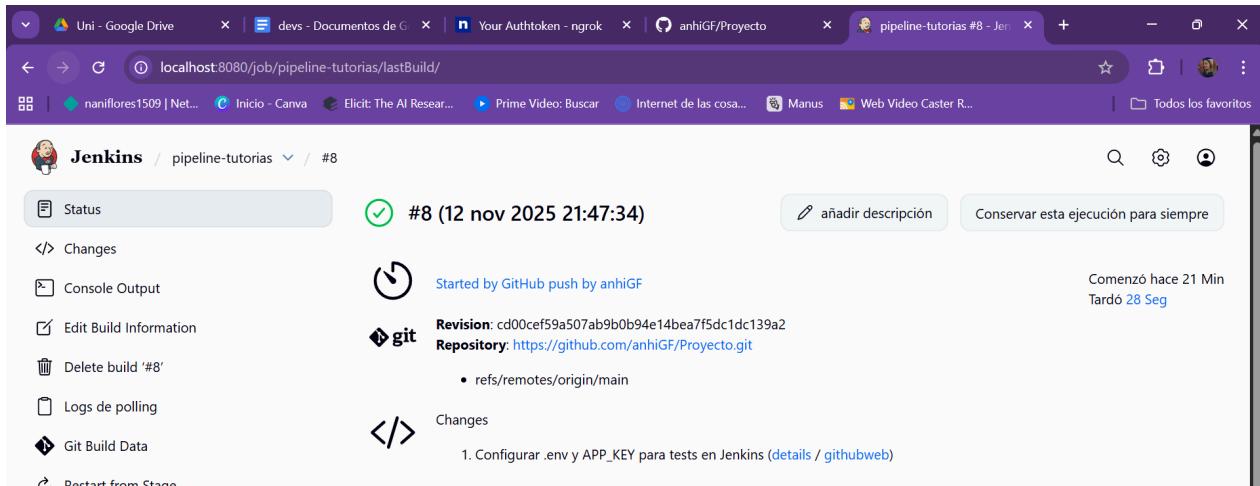
3.4 Ejecución automática del pipeline por evento push

Cada vez que se realiza un git push al repositorio, el webhook de GitHub activa automáticamente Jenkins.

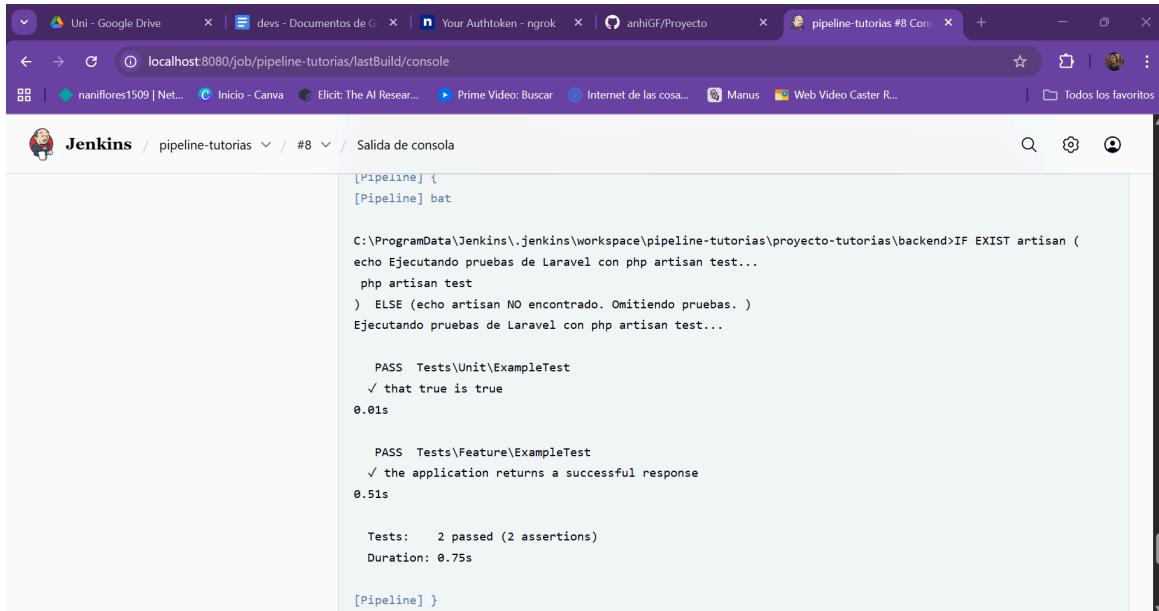
Evidencias:

- El pipeline se ejecuta con todas las etapas (Checkout → Backend → Frontend → Test → Deploy).
- En el historial de Jenkins se registran builds automáticos con #1, #2, etc.

Resultado: Integración continua confirmada.



The screenshot shows the Jenkins Pipeline #8 build status page. The build was started by GitHub push at 12 nov 2025 21:47:34. It took 28 Seg and completed 21 minutes ago. The build was triggered by a git commit (cd00cef59a507ab9b0b94e14bea7f5dc1dc139a2) from the repository https://github.com/anhigF/Proyecto.git. The changes log lists a single item: "1. Configurar .env y APP_KEY para tests en Jenkins". The sidebar includes links for Status, Changes, Console Output, Edit Build Information, Delete build '#8', Logs de polling, Git Build Data, and Download from Stash.



The screenshot shows the Jenkins Pipeline #8 console output. The output shows the execution of a pipeline script. It starts with "[Pipeline]` [Pipeline] bat". The script then runs "php artisan test" in the C:\ProgramData\Jenkins\.jenkins\workspace\pipeline-tutorias\proyecto-tutorias\backend directory. If "artisan" exists, it executes "php artisan test". Otherwise, it prints "Ejecutando pruebas de Laravel con php artisan test...". The output then shows the results of the unit and feature tests: "Tests: 2 passed (2 assertions)" and "Duration: 0.75s". The pipeline ends with "[Pipeline]`" and a closing brace "}".

```
[Pipeline]`  
[Pipeline] bat  
  
C:\ProgramData\Jenkins\.jenkins\workspace\pipeline-tutorias\proyecto-tutorias\backend>IF EXIST artisan (  
echo Ejecutando pruebas de Laravel con php artisan test...  
php artisan test  
) ELSE (echo artisan NO encontrado. Omitiendo pruebas. )  
Ejecutando pruebas de Laravel con php artisan test...  
  
    PASS  Tests\Unit\ExampleTest  
      ✓ that true is true  
    0.01s  
  
    PASS  Tests\Feature\ExampleTest  
      ✓ the application returns a successful response  
    0.51s  
  
Tests: 2 passed (2 assertions)  
Duration: 0.75s  
  
[Pipeline]` }
```

The screenshot shows a browser window with multiple tabs open. The active tab is titled "localhost:8080/job/pipeline-tutorias/lastBuild/console". The page displays the Jenkins pipeline console output for build #8. The output shows the execution of a Jenkinsfile, which includes stages for deployment and post actions. The final message indicates a successful execution.

```
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Despliegue simulado completado.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Pipeline ejecutado correctamente.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

The screenshot shows a browser window with multiple tabs open. The active tab is titled "localhost:8080/job/pipeline-tutorias/". The page displays the Jenkins pipeline job details for "pipeline-tutorias". The left sidebar shows various job management options like Status, Changes, and Configurar. The main content area shows the pipeline name "pipeline-tutorias" and a section for "Enlaces permanentes" (Permanent links) containing a list of previous executions. Below this is a "Builds" section with a table showing the last completed build.

Status

✓ pipeline-tutorias

añadir descripción

Changes

Construir ahora

Configurar

Borrar Pipeline

Rename

Pipeline Syntax

GitHub Hook Log

Credentials

Builds

Build	Estado	Tiempo
Última ejecución (#8)	correcta	6 Min 0 Seg
Última ejecución estable (#8)	correcta	6 Min 0 Seg
Última ejecución correcta (#8)	correcta	6 Min 0 Seg
Última ejecución fallida (#7)	fallida	13 Min
Última ejecución fallida (#7)	fallida	13 Min
Last completed build (#8)	correcta	6 Min 0 Seg

Filter

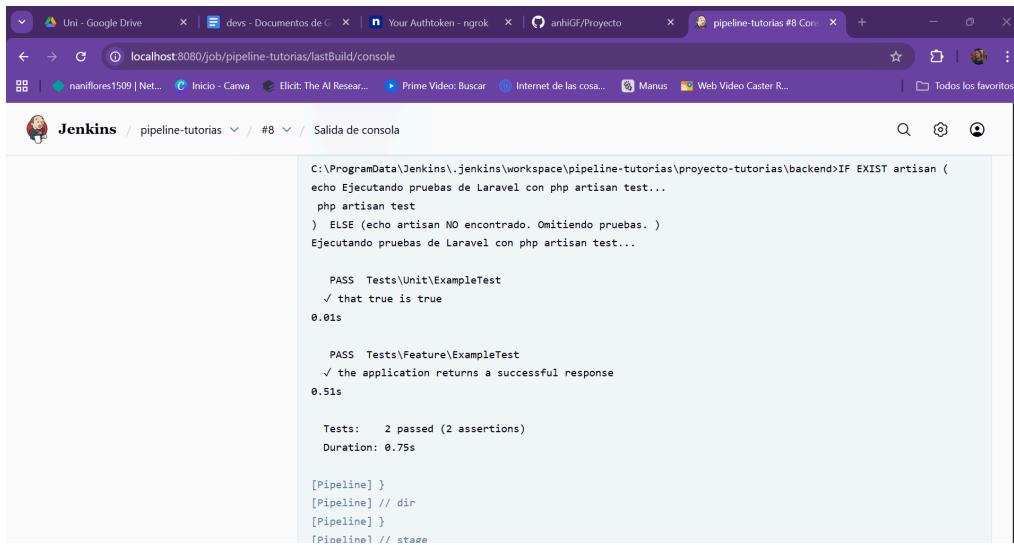
3.5 Reporte de análisis de código

Se agregaron pruebas utilizando php artisan test.

Gracias a la etapa Backend que genera automáticamente:

- .env
- APP_KEY

Las pruebas pudieron ejecutarse correctamente:



```
C:\ProgramData\Jenkins\jenkins\workspace\pipeline-tutorias\proyecto-tutorias\backend>IF EXIST artisan (
echo Ejecutando pruebas de Laravel con php artisan test...
php artisan test
) ELSE (echo artisan NO encontrado. Omitiendo pruebas. )
Ejecutando pruebas de Laravel con php artisan test...

PASS Tests\Unit\ExampleTest
✓ that true is true
0.01s

PASS Tests\Feature\ExampleTest
✓ the application returns a successful response
0.51s

Tests: 2 passed (2 assertions)
Duration: 0.75s

[Pipeline]
[Pipeline] // dir
[Pipeline] }
[Pipeline] // stage
```

Resultado:

- Esto demuestra que el backend está correctamente configurado dentro del entorno CI.

3.6 Enlace del repositorio en GitHub

Repositorio oficial del proyecto:

<https://github.com/anhiGF/Proyecto.git>

El repositorio contiene:

- Código fuente completo (frontend y backend)
- Script SQL Base_de_datos.sql
- Jenkinsfile

CONCLUSIÓN

La práctica permitió configurar un flujo CI/CD completo y funcional usando Jenkins, GitHub y Ngrok. El pipeline implementado logró automatizar exitosamente:

- Descarga del código (Checkout)
- Instalación de dependencias de Laravel y React
- Construcción del frontend
- Generación del archivo .env y APP_KEY
- Ejecución de pruebas unitarias y funcionales
- Un proceso de despliegue simulado

Esto garantiza una entrega continua confiable y con control de calidad, reduciendo tiempo y errores humanos durante el desarrollo del proyecto.

Referencias:

Download and deploy. (n.d.). Download and Deploy. <https://www.jenkins.io/download/>

ANHIGF/Proyecto. (n.d.). GitHub. <https://github.com/anhigf/Proyecto/tree/main>

Ngrok, & Ngrok. (n.d.). *Download ngrok.* <https://ngrok.com/download/windows>

Laravel LLC. (2025). *Testing: Getting Started.* <https://laravel.com/docs/testing>

Jenkins Project. (2023). *Building a JavaScript and PHP application with Jenkins Pipeline.* <https://www.jenkins.io/doc/tutorials/>

GitHub Docs. (2025). *Triggering CI/CD With Webhooks.* <https://docs.github.com/en/webhooks-and-events/webhooks/creating-webhooks>

Laravel Docs. (2025). *Configuring Environment Variables & APP_KEY.* <https://laravel.com/docs/configuration>