



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

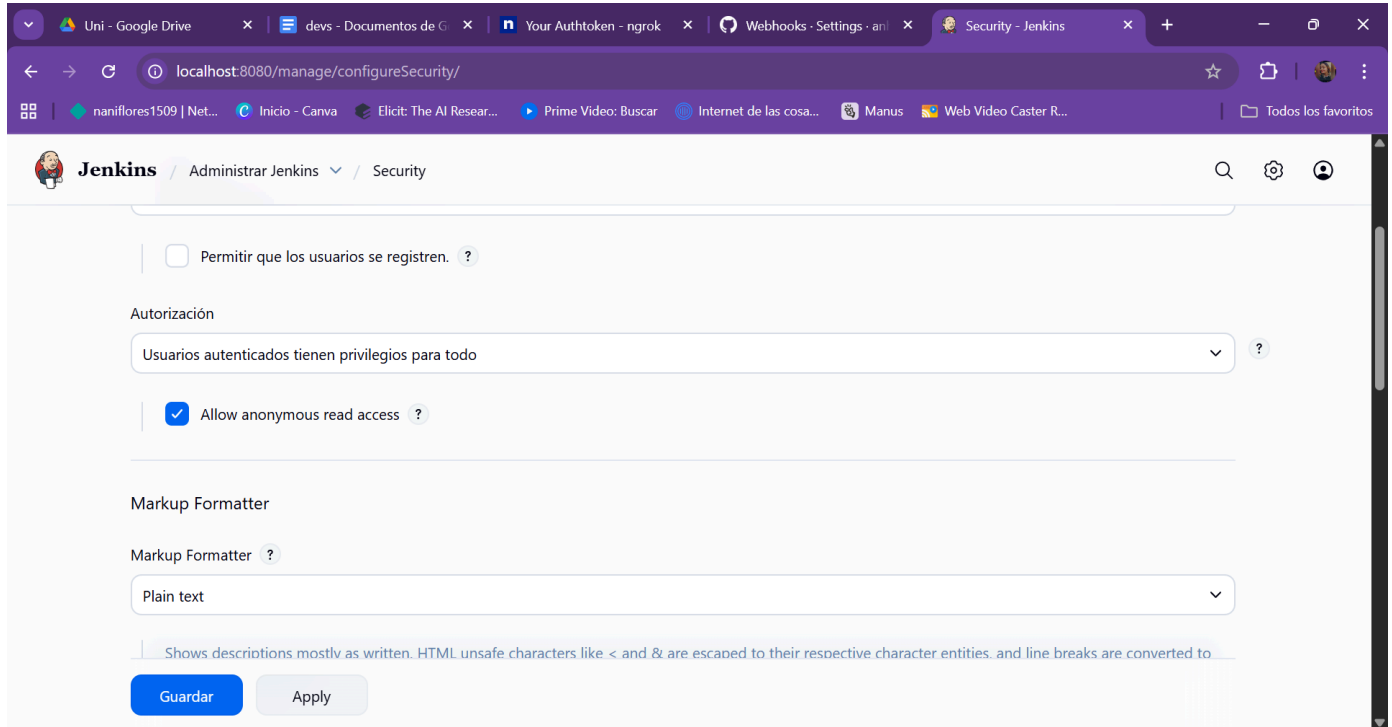
- Configurar un servidor Jenkins en entorno local.
- Establecer una conexión segura entre Jenkins y GitHub mediante Ngrok.
- Crear un Pipeline declarativo que automatice las etapas de build, test y despliegue.
- Ejecutar automáticamente el pipeline al detectar un evento push en GitHub.
- Analizar el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



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
```

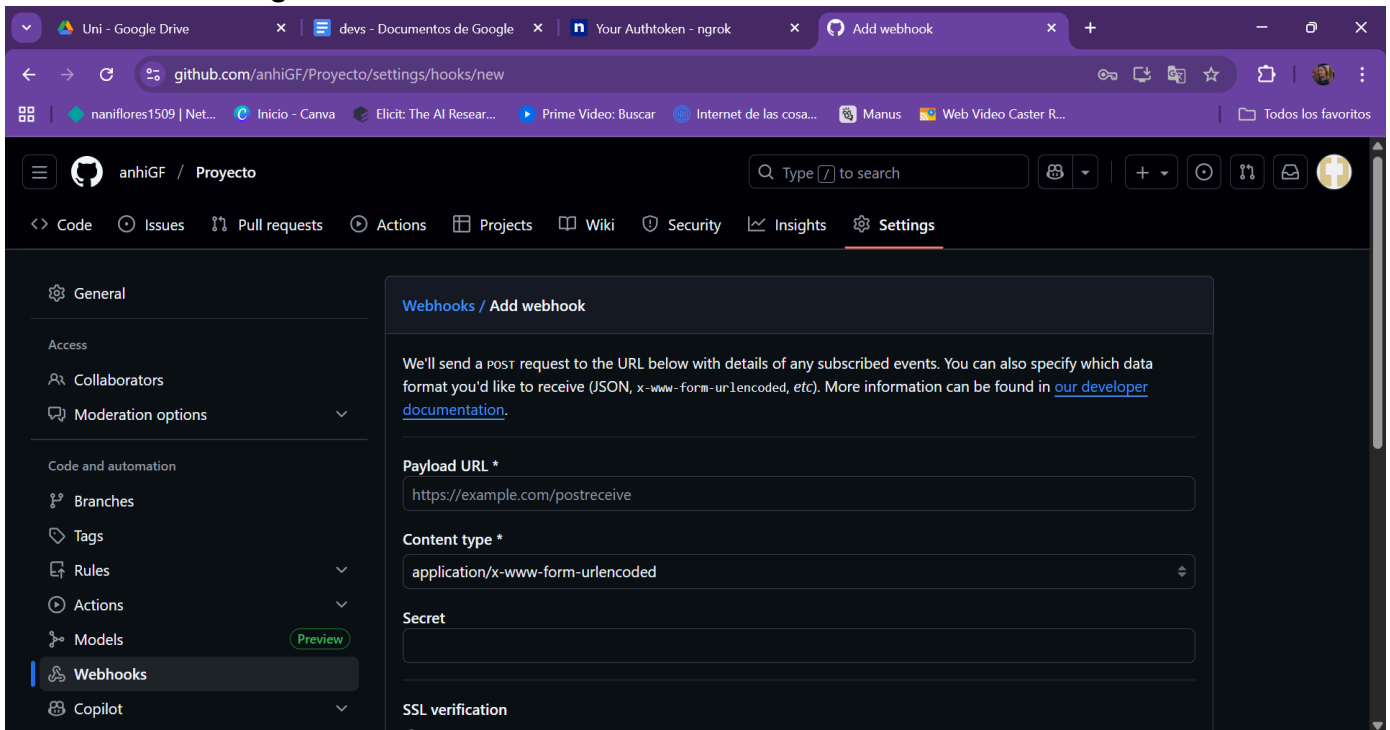
1. Copiar la URL generada

```
C:\Program Files\WindowsAp  x  +  v  (Ctrl+C to quit)
ngrok
♦ Using ngrok for OSS? Request a community license: https://ngrok.com/r/oss

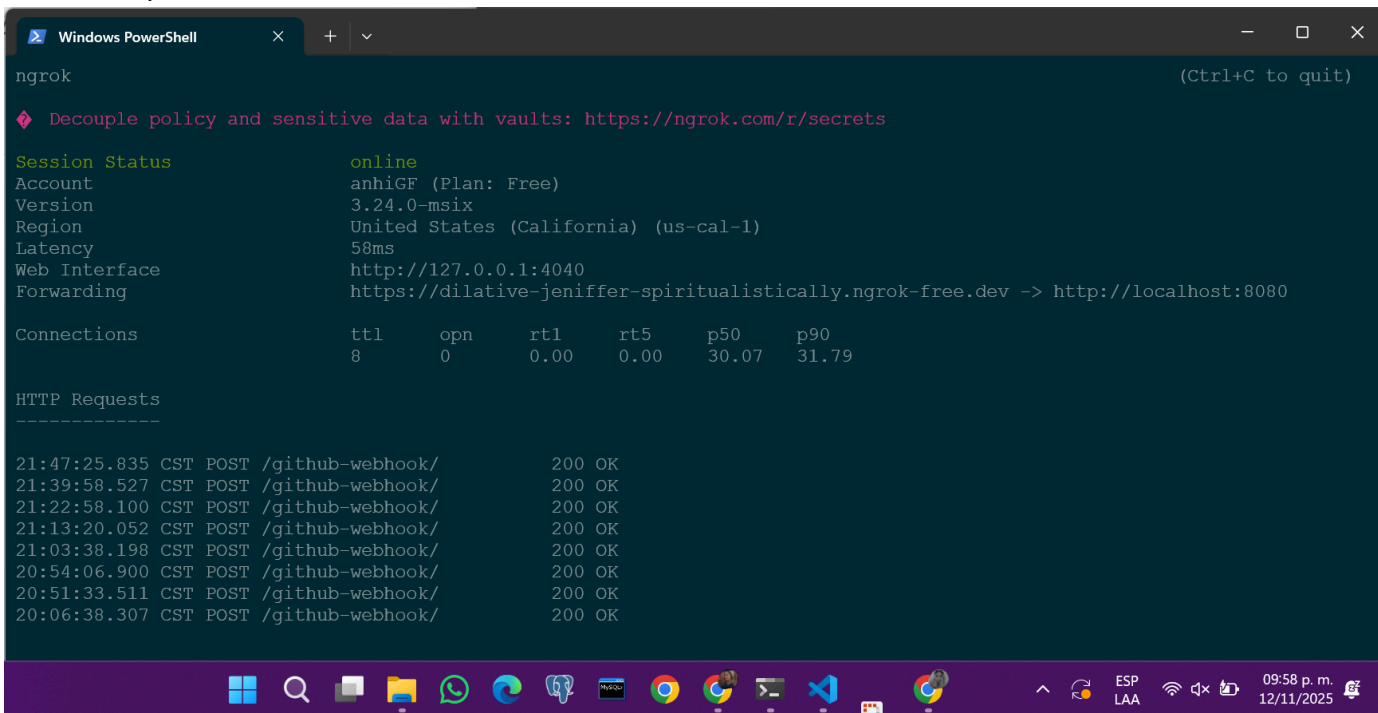
Session Status      online
Account             anhiGF (Plan: Free)
Update              update available (version 3.33.0, Ctrl-U to update)
Version             3.24.0-msix
Region              United States (California) (us-cal-1)
Latency             138ms
Web Interface       http://127.0.0.1:4040
Forwarding           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:



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



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

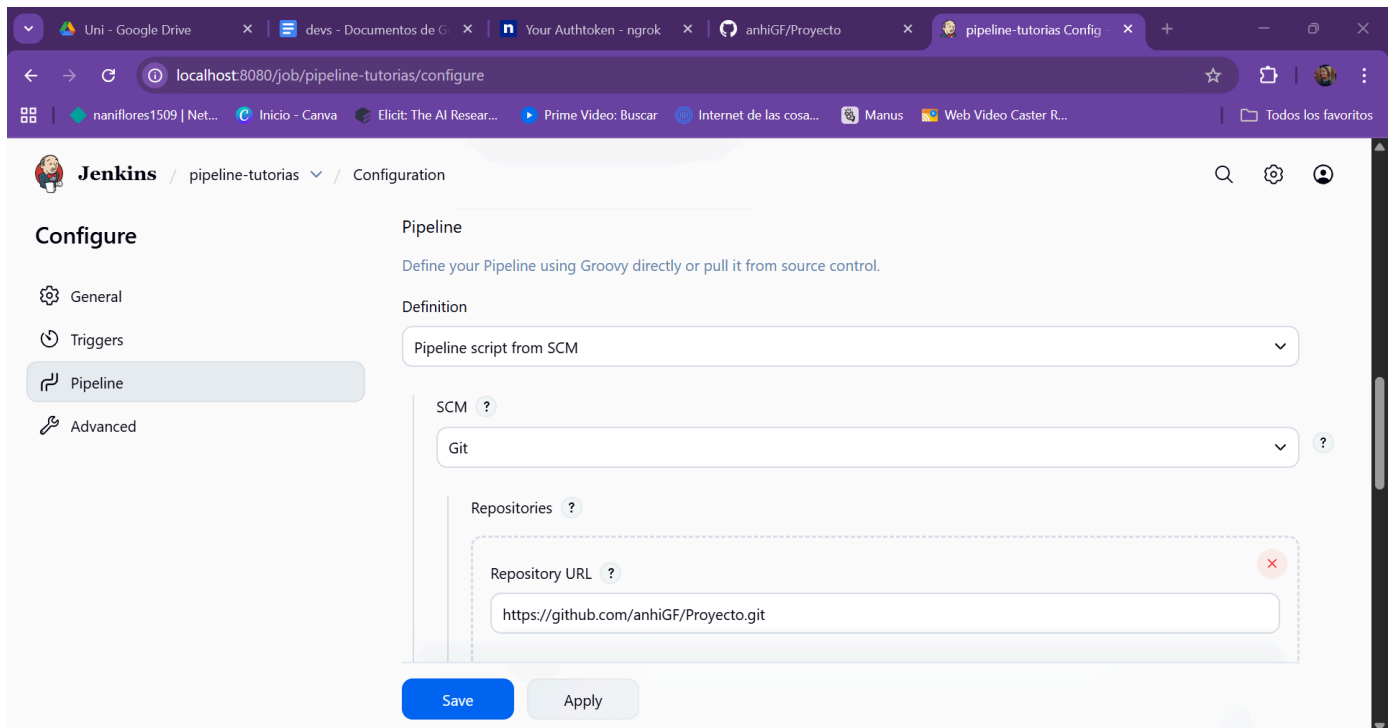
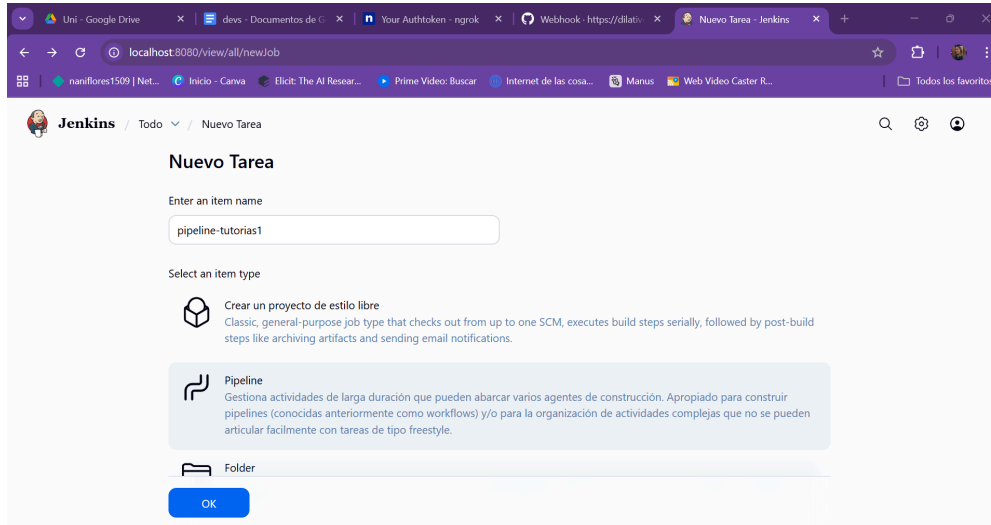
3.3 Creación del Pipeline en Jenkins

Se creó un nuevo trabajo tipo Pipeline en Jenkins

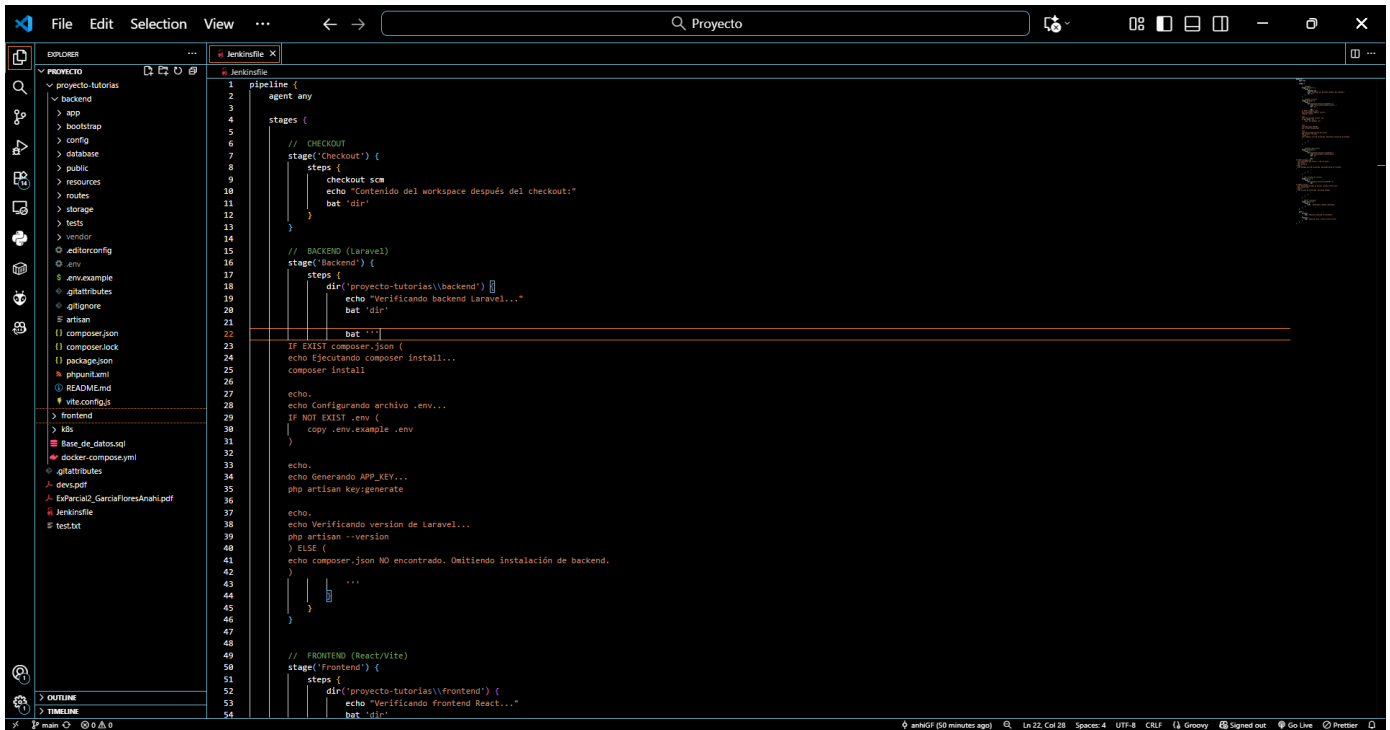
Definición: Pipeline script from SCM

SCM: Git

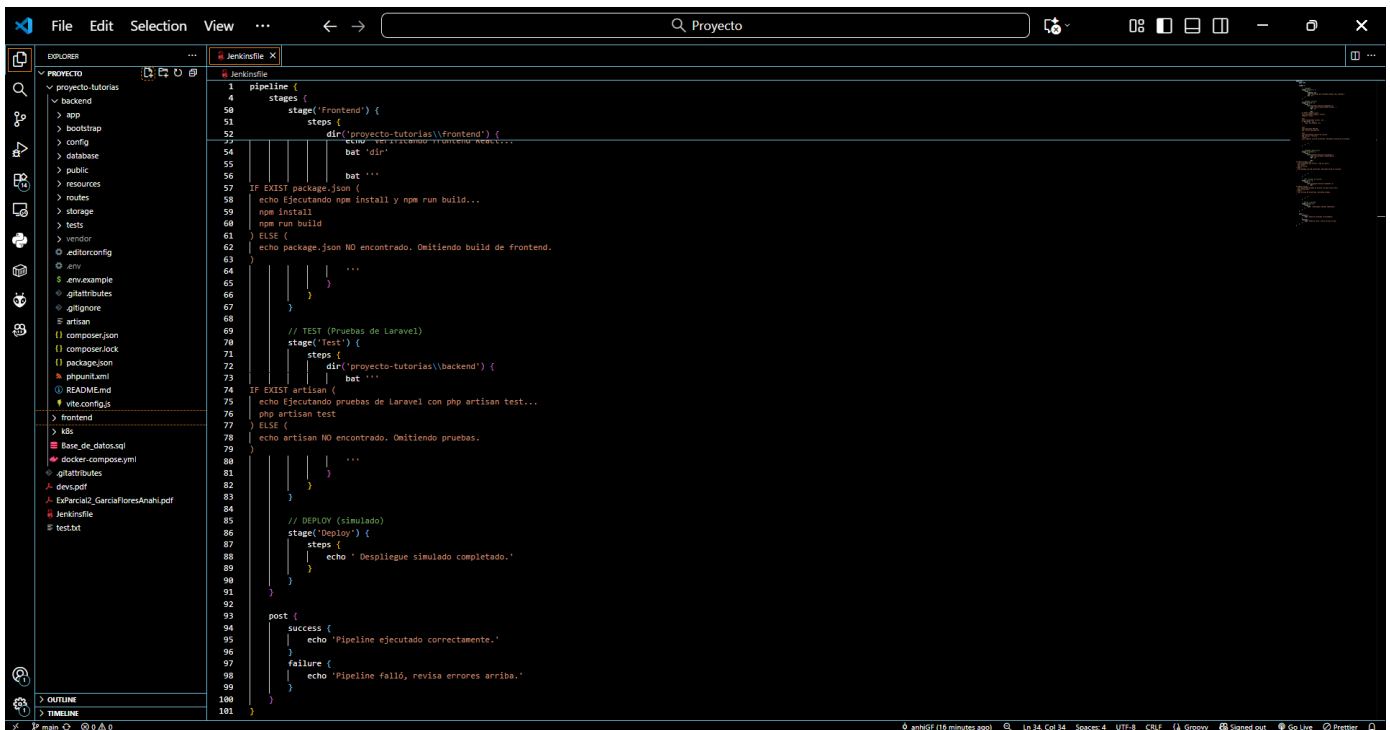
Repository URL: <https://github.com/anhigf/Proyecto>



Contenido del archivo 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      }
22    }
23
24    IF EXISTS composer.json {
25      echo Ejecutando composer install...
26      composer install
27
28      echo Configurando archivo .env...
29      IF NOT EXISTS .env {
30        copy .env.example .env
31      }
32
33      echo Generando APP_KEY...
34      php artisan key:generate
35
36      echo Verificando version de Laravel...
37      php artisan --version
38    } ELSE {
39      echo composer.json NO encontrado. Omitiendo instalación de backend.
40    }
41
42    // FRONTEND (React/Vite)
43    stage("Frontend") {
44      steps {
45        dir('proyecto-tutorias\\frontend') {
46          echo "Verificando frontend React..."
47          bat "dir"
48        }
49      }
50    }
51  }
52}
```



```
53
54
55    // TEST (Pruebas de Laravel)
56    stage("Test") {
57      steps {
58        dir('proyecto-tutorias\\backend') {
59          bat "dir"
60        }
61      }
62    }
63
64    IF EXISTS artisan {
65      echo Ejecutando pruebas de Laravel con php artisan test...
66      php artisan test
67    } ELSE {
68      echo artisan NO encontrado. Omitiendo pruebas.
69    }
70
71    // DEPLOY (simulado)
72    stage("Deploy") {
73      steps {
74        echo "Despliegue simulado completado."
75      }
76    }
77
78    post {
79      success {
80        echo "Pipeline ejecutado correctamente."
81      }
82      failure {
83        echo "Pipeline falló, revisa errores arriba."
84      }
85    }
86  }
87}
```

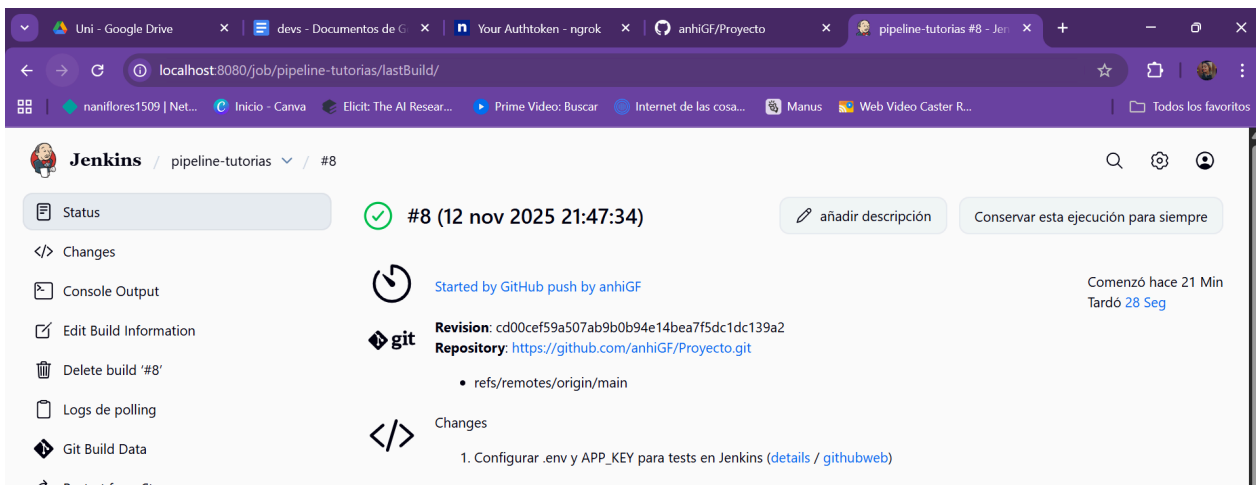
3.4 Ejecución automática del pipeline por evento push

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

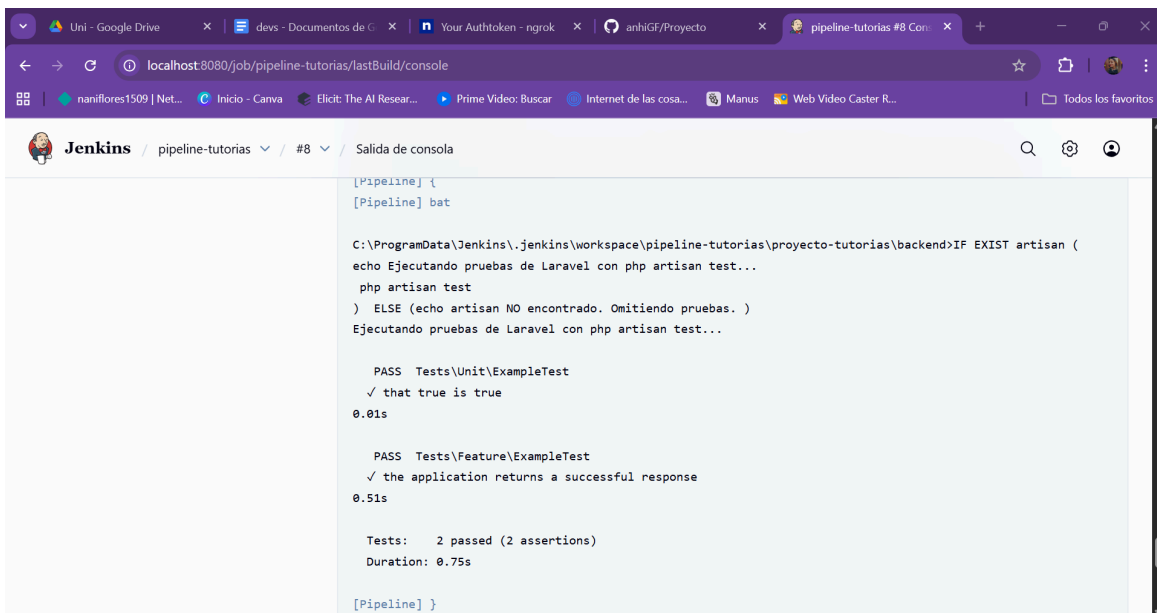
Evidencias:

- 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 web interface for a pipeline named 'pipeline-tutorias'. The build #8, dated 12 nov 2025 21:47:34, is shown as successful (green checkmark). It was started by a GitHub push by user 'anhiGF'. The revision is 'cd00cef59a507ab9b0b94e14bea7f5dc1dc139a2' from the repository 'https://github.com/anhiGF/Proyecto.git', specifically the 'refs/remotes/origin/main' branch. The build details show a 'Changes' section with one change: '1. Configurar .env y APP_KEY para tests en Jenkins (details / githubweb)'. The interface includes a sidebar with links to Status, Changes, Console Output, Edit Build Information, Delete build '#8', Logs de polling, and Git Build Data. The top navigation bar shows the Jenkins logo and the pipeline name.



The screenshot shows the Jenkins web interface for the same pipeline, but displaying the console output for build #8. The output is a JSON object representing the pipeline execution. It shows the pipeline starting with a 'bat' command, followed by a shell script that checks for the existence of 'artisan' and runs 'php artisan test'. The output indicates that the tests passed (2 passed, 2 assertions) and the duration was 0.75s. The console output is displayed in a light blue box with a dark blue border. The top navigation bar shows the Jenkins logo and the pipeline name.

Uni - Google Drive | devs - Documentos de G | Your Authtoken - ngrok | anhiGF/Proyecto | pipeline-tutorias #8 Con | +

localhost:8080/job/pipeline-tutorias/lastBuild/console

naniflores1509 | Net... | Inicio - Canva | Elicit: The AI Resear... | Prime Video: Buscar | Internet de las cosa... | Manus | Web Video Caster R... | Todos los favoritos

Jenkins / pipeline-tutorias / #8 / Salida de consola

```
[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
```

Uni - Google Drive | devs - Documentos de G | Your Authtoken - ngrok | anhiGF/Proyecto | pipeline-tutorias - Jenkins | +

localhost:8080/job/pipeline-tutorias/

naniflores1509 | Net... | Inicio - Canva | Elicit: The AI Resear... | Prime Video: Buscar | Internet de las cosa... | Manus | Web Video Caster R... | Todos los favoritos

Jenkins / pipeline-tutorias

Status

Changes

Construir ahora

Configurar

Borrar Pipeline

Rename

Pipeline Syntax

GitHub Hook Log

Credentials

pipeline-tutorias

añadir descripción

Enlaces permanentes

- "Última ejecución (#8) hace 6 Min 0 Seg"
- "Última ejecución estable (#8) hace 6 Min 0 Seg"
- "Última ejecución correcta (#8) hace 6 Min 0 Seg"
- "Última ejecución fallida (#7) hace 13 Min"
- "Última ejecución fallida (#7) hace 13 Min"
- "Last completed build (#8) hace 6 Min 0 Seg"

Builds

Filter

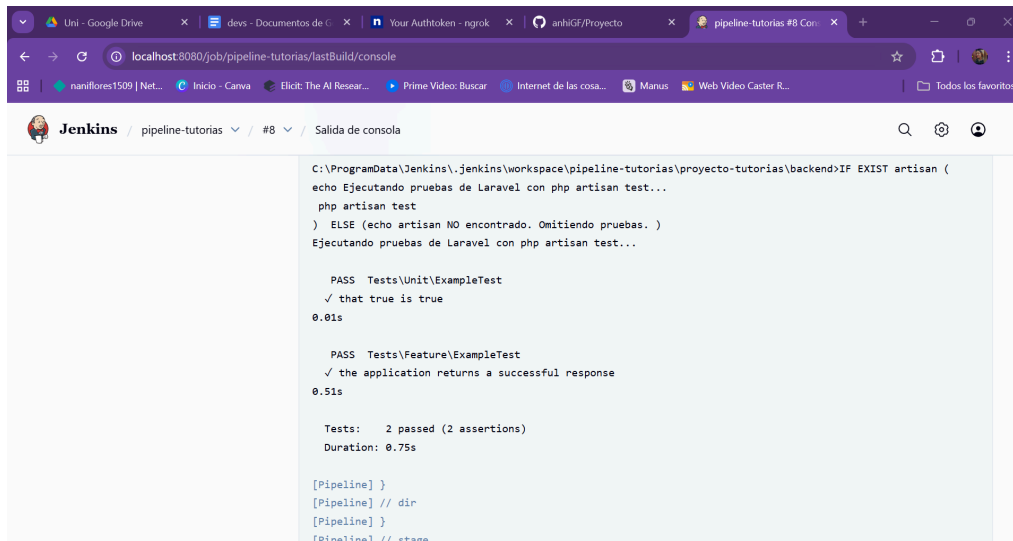
3.5 Reporte de análisis de código

Se agregaron pruebas utilizando php artisan test.

Gracias a la etapa Backend que genera automáticamente:

- .env
- APP_KEY

Las pruebas pudieron ejecutarse correctamente:

A screenshot of a web browser displaying the Jenkins console output for a build named 'pipeline-tutorias #8'. The browser's address bar shows 'localhost:8080/job/pipeline-tutorias/lastBuild/console'. The Jenkins interface shows the console output in a light blue box. The output text is as follows:

```
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>