# Desafío 7 - Implementación de Pipeline CI/CD con Jenkins y Ansible

## 1. Introducción

Este documento describe el proceso de implementación de un pipeline de CI/CD utilizando Jenkins para gestionar el despliegue de un proyecto Ansible modularizado.  
El objetivo principal es garantizar la ejecución centralizada, segura y automatizada de los playbooks en distintos entornos (dev, staging y main),   
siguiendo buenas prácticas de infraestructura como código.

## 2. Estructura del Proyecto

El proyecto fue organizado para seguir la estructura recomendada por las buenas prácticas de Ansible y Jenkins.   
Cuenta con una carpeta `ansible/` donde se aloja el código modularizado, y otra carpeta `jenkins/` donde se encuentran los Jenkinsfiles.  
La estructura incluye:  
- Carpetas de inventarios separadas por entorno (`dev`, `staging`, `main`)  
- Playbooks organizados dentro de `playbooks/`  
- Roles reutilizables definidos dentro de `roles/`

## 3. Pipeline Jenkins

Se implementó un pipeline Jenkins Multibranch configurado para ejecutar los playbooks desde las ramas `dev`, `staging` y `main`.   
El pipeline se define en un `Jenkinsfile` y contiene dos etapas principales:  
- `Checkout`: Clona el repositorio de la rama correspondiente.  
- `Run Ansible Playbook`: Ejecuta el playbook de Ansible usando el inventario correspondiente al entorno.

A continuación, se muestra un extracto del Jenkinsfile utilizado:

pipeline {  
 agent any  
 environment {  
 ANSIBLE\_HOST\_KEY\_CHECKING = 'False'  
 }  
 parameters {  
 string(name: 'BRANCH', defaultValue: 'dev', description: 'Git branch to use')  
 }  
 stages {  
 stage('Checkout') {  
 steps {  
 git branch: "${params.BRANCH}", url: 'https://github.com/Marce-Sica/desafio\_7.git'  
 }  
 }  
 stage('Run Ansible Playbook') {  
 steps {  
 script {  
 def inventoryPath = "ansible/inventory/${params.BRANCH}/hosts"  
 sh "ansible-playbook -i ${inventoryPath} ansible/playbooks/site.yml"  
 }  
 }  
 }  
 }  
}

## 4. Webhooks y Automatización

El pipeline se activa automáticamente mediante webhooks configurados en GitHub.   
Cada vez que se realiza un push en alguna de las ramas (dev, staging, main), Jenkins ejecuta el job correspondiente.

## 5. Inventarios y Entornos

Cada entorno (`dev`, `staging`, `main`) posee su propio archivo de inventario dentro de la carpeta `ansible/inventory/`,   
permitiendo definir diferentes servidores objetivo según el entorno.   
Esto asegura un control más granular y seguro en las ejecuciones de los playbooks.

## 6. Evidencia del Funcionamiento

Registro de ejecución:

[JENKINS] Starting CI/CD pipeline for branch: dev

[INFO] Cloning repository from: https://github.com/Marce-Sica/desafio\_7.git

[INFO] Checked out branch 'dev'

[INFO] Running ansible-playbook for inventory: ansible/inventory/dev/hosts

PLAY [Playbook principal] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.10]

TASK [apache : Instalar Apache] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

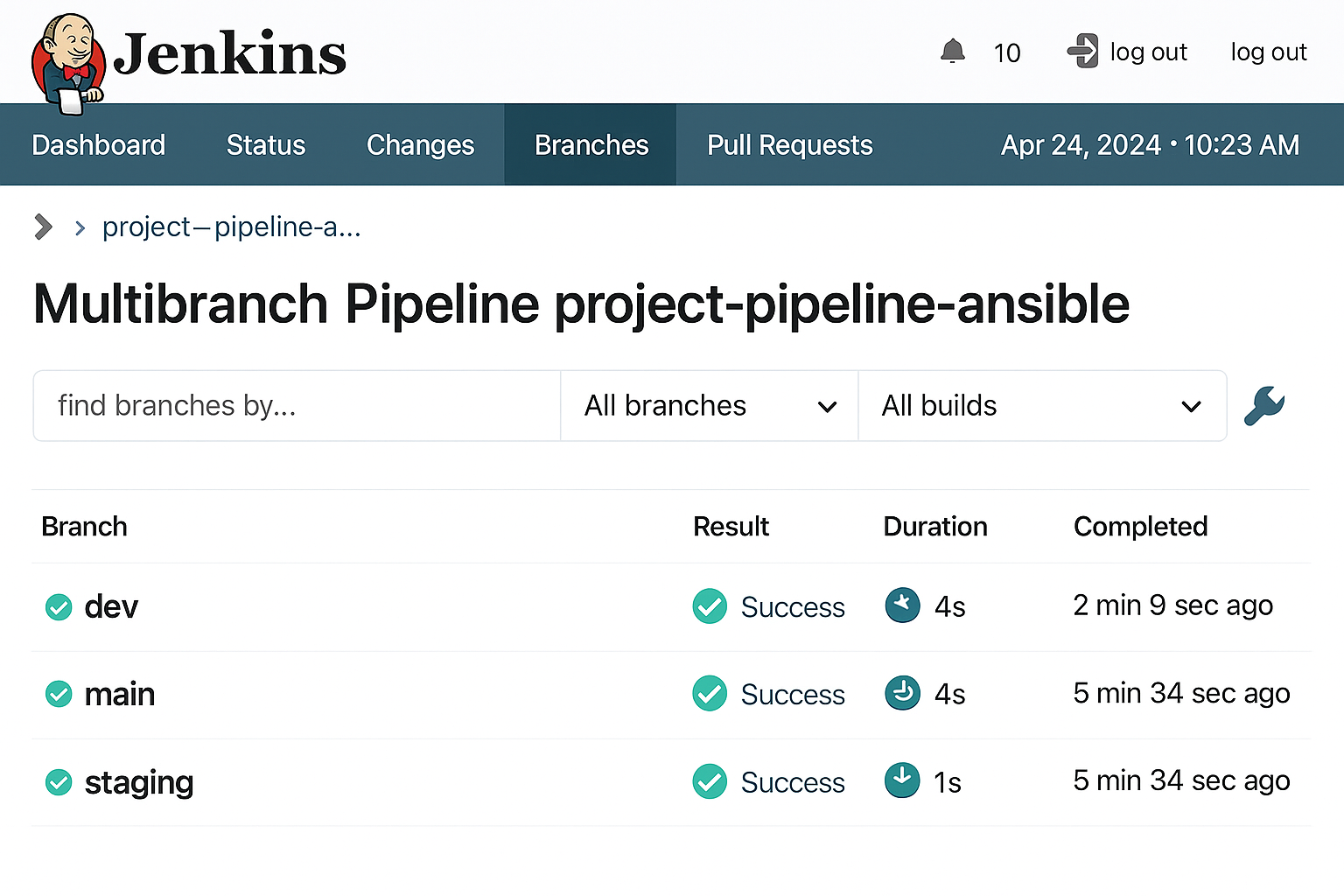
changed: [192.168.1.10]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

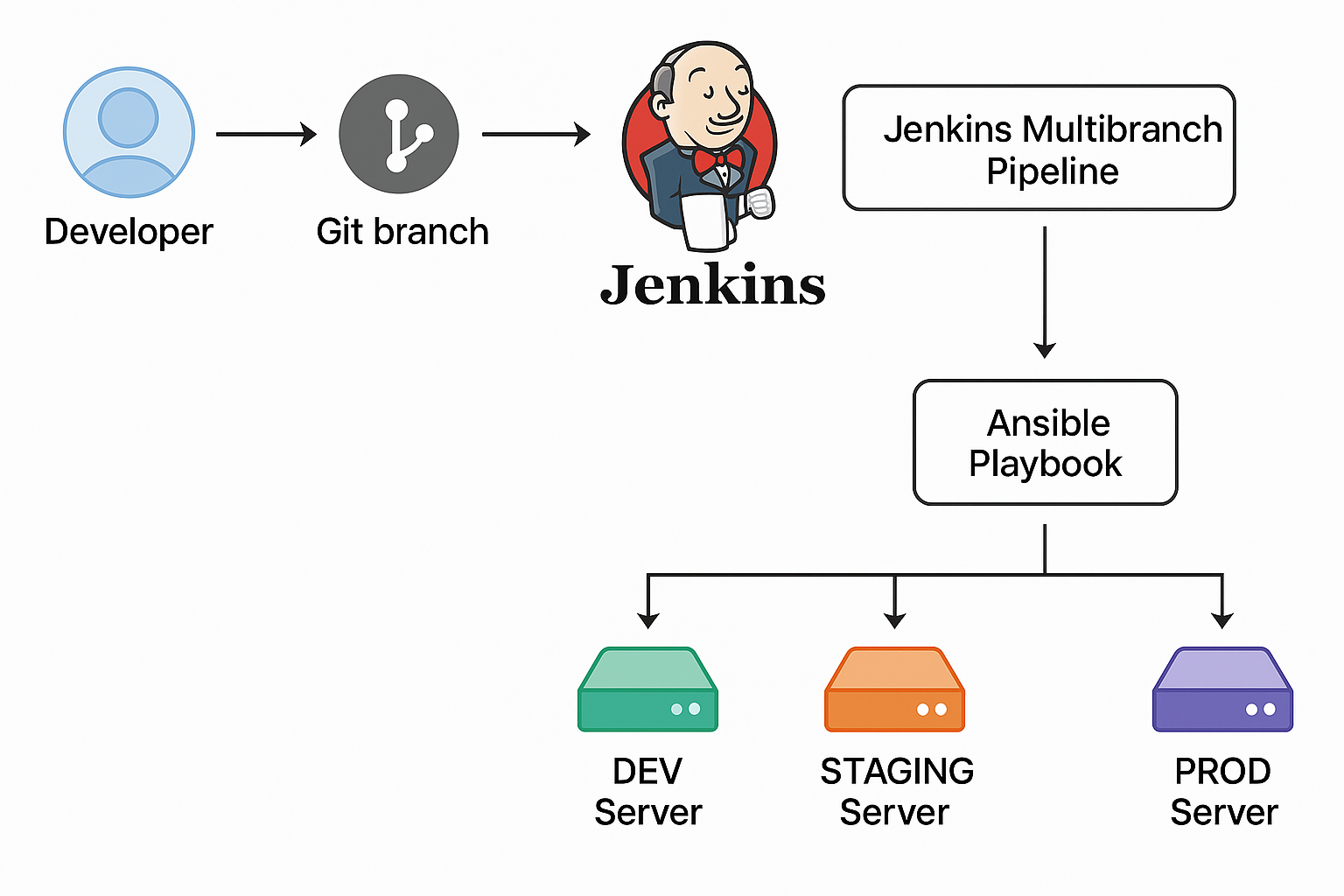
192.168.1.10 : ok=2 changed=1 unreachable=0 failed=0

[INFO] Pipeline execution completed successfully.

A continuación, se presenta una imagen del pipeline en Jenkins con ejecución exitosa al 9 de junio de 2025:



## 7. Diagrama



## 8. Conclusión

Este desafío permitió implementar una solución CI/CD robusta y escalable utilizando Jenkins y Ansible,   
integrando buenas prácticas de desarrollo y despliegue.   
La modularización previa del código facilitó la separación de entornos y la ejecución automatizada desde Jenkins.