DESARROLLO DE PIPELINE DE JENKINS

David Recio Arnés

ENTORNOS INTEGRACIÓN Y ENTREGA CONTINUA

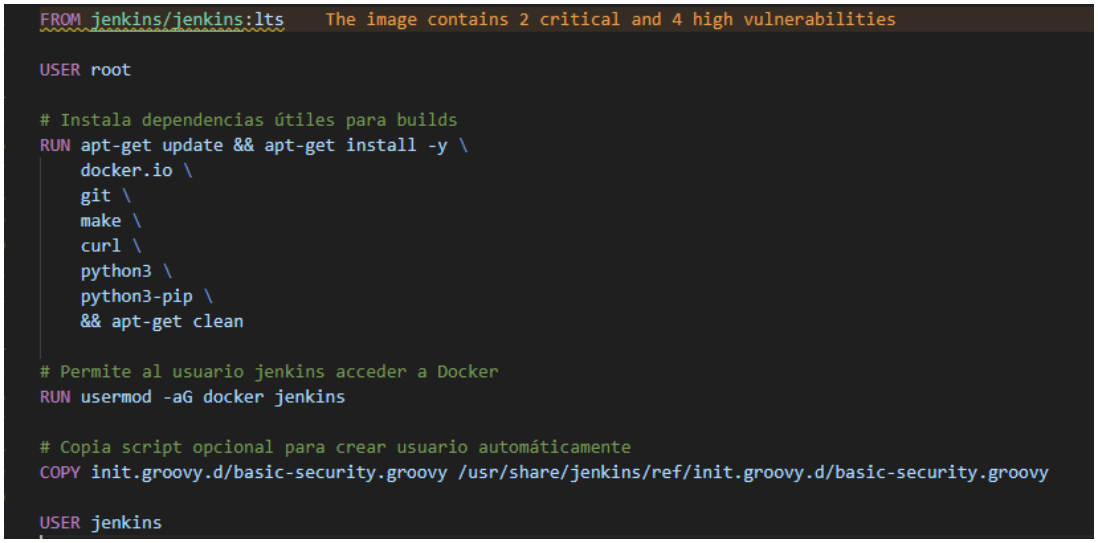
Contenido

Introducción

En esta práctica se ha desarrollado un pipeline en Jenkins con el objetivo de automatizar tareas dentro del proceso de integración continua, abarcando etapas como la construcción del proyecto, la ejecución de pruebas unitarias, pruebas de API y pruebas de extremo a extremo (E2E). Para ello, se ha tomado como base el repositorio público srayuso/unir-cicd, el cual ha sido ampliado para incorporar nuevas funcionalidades. Entre las mejoras implementadas se encuentran la inclusión de etapas específicas para la ejecución de pruebas de API y E2E, el archivado automático de los resultados en formato XML, la visualización de informes generados por dichas pruebas y la configuración de un sistema de notificaciones por correo electrónico, el cual se activa únicamente cuando la ejecución del *pipeline* finaliza con errores, incluyendo en el mensaje el nombre del trabajo y el número de ejecución correspondiente.

Entorno de Trabajo

Para la elaboración de la práctica se ha creado un proyecto sencillo que permite desplegar y configurar un servidor Jenkins utilizando Docker. Es importante destacar que se ha empleado una imagen personalizada de Jenkins, ya que la imagen oficial no incluye todos los componentes necesarios para el desarrollo completo de la actividad. Para levantar el entorno, se ha utilizado la herramienta Docker Desktop como plataforma de virtualización y gestión de contenedores.

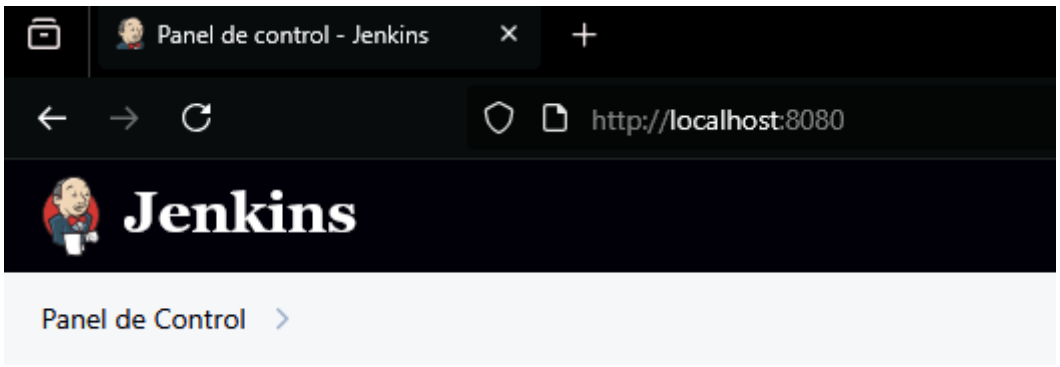


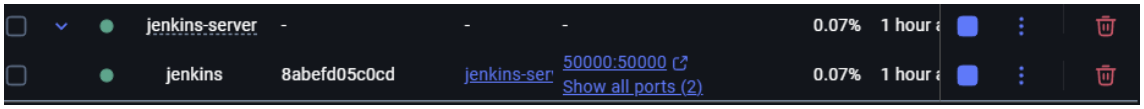
```
FROM jenkins/jenkins:lts    The image contains 2 critical and 4 high vulnerabilities

USER root

# Instala dependencias útiles para builds
RUN apt-get update && apt-get install -y \
    docker.io \
    git \
    make \
    curl \
    python3 \
    python3-pip \
    && apt-get clean

# Permite al usuario jenkins acceder a Docker
RUN usermod -aG docker jenkins

# Copia script opcional para crear usuario automáticamente
COPY init.groovy.d/basic-security.groovy /usr/share/jenkins/ref/init.groovy.d/basic-security.groovy

USER jenkins
```

Tal como se observa en la imagen, se instalan una serie de dependencias necesarias para el proceso de *build*, además de copiar la configuración inicial de Jenkins para asegurar que el entorno esté correctamente preparado desde el primer arranque.

```groovy
#!groovy

import jenkins.model.*
import hudson.security.*

def instance = Jenkins.getInstance()

println "--> Configurando seguridad básica..."

def hudsonRealm = new HudsonPrivateSecurityRealm(false)
hudsonRealm.createAccount("admin", "admin123")

instance.setSecurityRealm(hudsonRealm)

def strategy = new FullControlOnceLoggedInAuthorizationStrategy()
strategy.setAllowAnonymousRead(false)
instance.setAuthorizationStrategy(strategy)

instance.save()

println "--> Usuario admin configurado con éxito."
```

En esta imagen se muestra la configuración de una cuenta de usuario predefinida para acceder al panel de Jenkins, facilitando así el inicio de sesión sin necesidad de crear credenciales manualmente tras el despliegue.

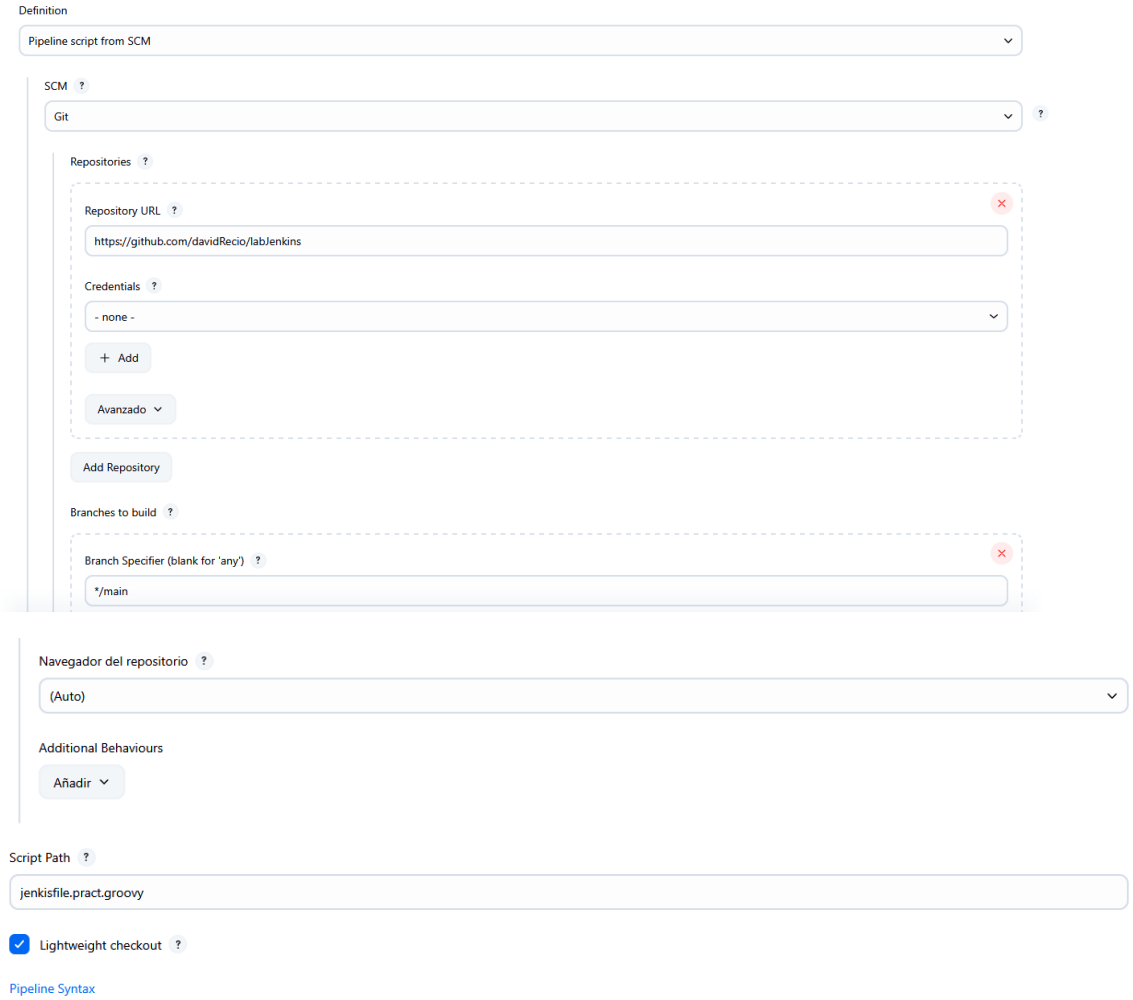




Cabe mencionar que, debido a que en la entrega únicamente se permite incluir el Jenkinsfile y esta documentación, en el apartado de bibliografía se proporcionarán todos los enlaces a los repositorios públicos utilizados, tanto el correspondiente al servidor Jenkins como el del proyecto desarrollado para la práctica.

Configuración de Jenkins

Una vez dentro del panel de Jenkins, se procedió a la instalación de los plugins sugeridos por defecto. Tras completar este paso, se creó un *job* de tipo *Pipeline* llamado **“práctica_despliegue”**. Esta *pipeline* está configurada para obtener su definición desde un sistema de control de versiones (SCM), seleccionando la opción *Pipeline script from SCM* y utilizando Git como sistema de origen, ya que estará asociado a un repositorio público. Es importante señalar que, en caso de tratarse de un repositorio privado, sería necesario configurar las credenciales adecuadas para permitir el acceso. Se especifica la rama desde la cual se leerá el archivo Jenkinsfile, en este caso "*/main", y finalmente se indica el nombre del fichero que debe ejecutarse: jenkinsfile.pract.groovy.



Jenkinsfile explicado

Para la elaboración de l fichero de jenkis se ha hecho uso de la plantilla establecida en la practica de la cual se ha completado en base a lo requerido en la misma.

```
pipeline {

    agent any

    environment {
        JOB_NAME_ENV = "${env.JOB_NAME}"
        BUILD_NUMBER_ENV = "${env.BUILD_NUMBER}"
    }
    stages {
        stage('Source') {
            steps {
                git 'https://github.com/srayuso/unir-cicd.git'
            }
        }       You, 9 minutes ago • Update jenkisfile.pract.groovy …
        stage('Build') {
            steps {
                echo 'Building stage!'
                sh 'make build'
            }
        }
        stage('Unit tests') {
            steps {
                sh 'make test-unit'
                archiveArtifacts artifacts: 'results/*.xml'
            }
        }
        stage('API tests') {
            steps {
                sh 'make test-api'
                archiveArtifacts artifacts: 'results/api_*.xml', allowEmptyArchive: true
            }
        }
        stage('E2E tests') {
            steps {
                sh 'make test-e2e'
                archiveArtifacts artifacts: 'results/e2e_*.xml', allowEmptyArchive: true
            }
        }
    }
```

Respecto a los workers se ha establecido a cualquier agente haciendo uso de agent any.

Después se han establecido las siguientes etapas:

- Source: en esta etapa se realiza el clonado del repositorio del código
- Build: en esta etapa se construye el aplicativo
- Unit test: en esta etapa se realizan las pruebas unitarias
- Api Test: en esta etapa se realizan las pruebas de aplicación
- E2E test: en esta etapa se realizan las pruebas end to end

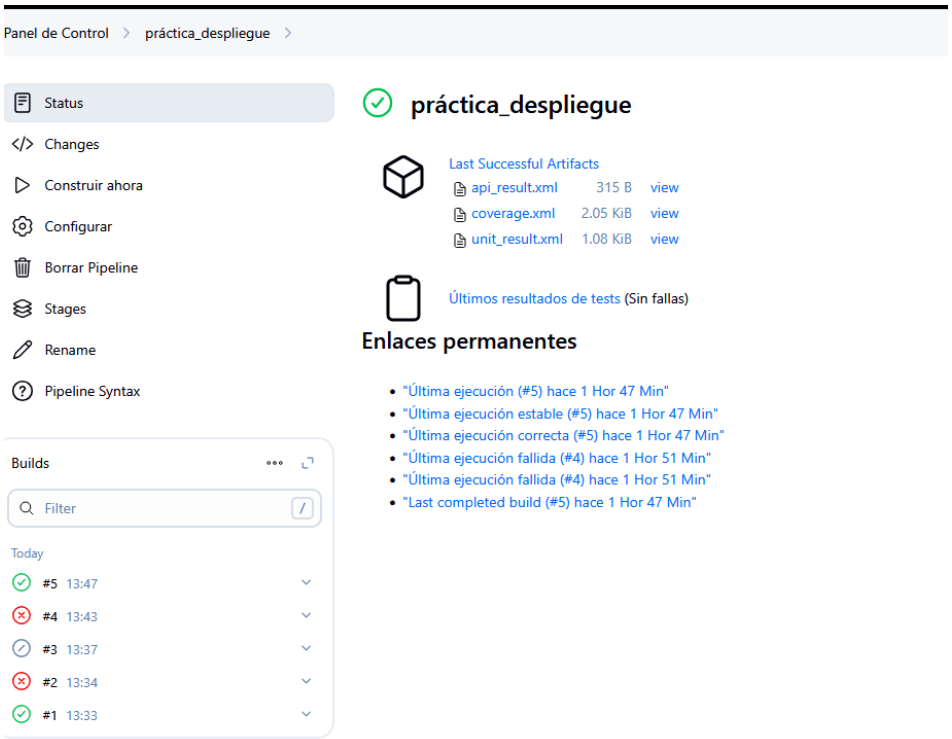
```
    post {
        always {
            echo 'Publishing test reports...'
            junit 'results/*_result.xml'
            cleanWs()
        }
        failure {
            echo "Sending failure notification for ${env.JOB_NAME} build #${env.BUILD_NUMBER}"
            // mail to: 'devops@fintechsolutions.com',
            //      subject: "FAILED: Job ${env.JOB_NAME} [#${env.BUILD_NUMBER}]",
            //      body: "Build failed. Check Jenkins for details."
        }
    }
}
```

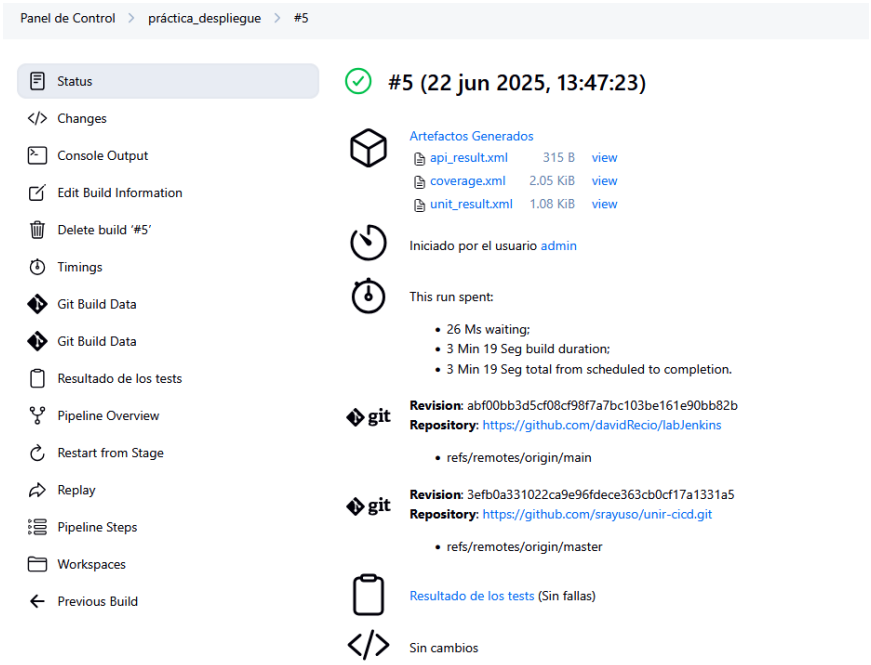
Por último, se define una sección de post-ejecución que se activa una vez finalizadas las fases principales del pipeline. En esta etapa, se realiza la publicación de los resultados de las pruebas ejecutadas y, en caso de que la ejecución haya fallado, se envía automáticamente un correo electrónico notificando el fallo. Dicho correo incluye el nombre del job y el número de ejecución que ha generado el error, facilitando así la identificación y el seguimiento del problema.

Resultados

En esta sección se mostrarán las imágenes que corroboran que se realizo correctamente la practica atendiendo a los objetivos de esta.

```
docker network rm calc-test-e2e || true
calc-test-e2e
[Pipeline] archiveArtifacts
Archiving artifacts
'results/e2e_*.xml' doesn't match anything: 'results' exists but not 'results/e2e_*.xml'
No artifacts found that match the file pattern "results/e2e_*.xml". Configuration error?
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Publishing test reports...
[Pipeline] junit
Recording test results
[Checks API] No suitable checks publisher found.
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```




Panel de Control
práctica_despliegue
#5
Status
Changes
Console Output
Edit Build Information
Delete build '#5'
Timings
Git Build Data
Git Build Data
Resultado de los tests
Pipeline Overview
Restart from Stage
Replay
Pipeline Steps
Workspaces
Previous Build
#5 (22 jun 2025, 13:47:23)
Artefactos Generados
api_result.xml 315 B view
coverage.xml 2.05 KiB view
unit_result.xml 1.08 KiB view
Iniciado por el usuario admin
This run spent:
26 Ms waiting;
3 Min 19 Seg build duration;
3 Min 19 Seg total from scheduled to completion.
Revision: abf00bb3d5cf08cf98f7a7bc103be161e90bb82b
Repository: https://github.com/davidRecio/labJenkins
refs/remotes/origin/main
Revision: 3efb0a331022ca9e96fdece363cb0cf17a1331a5
Repository: https://github.com/srayuso/unir-cicd.git
refs/remotes/origin/master
Resultado de los tests (Sin fallas)
Sin cambios

Bibliografía

- Recio, D. [@davidRecio]. (s.f.). *labJenkins* [Repositorio GitHub]. GitHub. https://github.com/davidRecio/labJenkins
- Afraileni, J. [@jafraileni]. (s.f.). *unir-cicd* [Repositorio GitHub]. GitHub. https://github.com/jafraileni/unir-cicd
- Recio, D. [@davidRecio]. (s.f.). *servidor_jenkin* [Repositorio GitHub]. GitHub. https://github.com/davidRecio/servidor_jenkin